INVESTIGATION AND APPLICATION OF ARTIFICIAL INTELLIGENCE ALGORITHMS FOR COMPLEXITY METRICS BASED CLASSIFICATION OF SEMANTIC WEB ONTOLOGIES

GIDEON KIPROTICH KOECH

217250661

This dissertation is submitted in fulfillment of the requirement for the Magister Technologiae (MTech): Information Technology

Vaal University of Technology, Main Campus.

Private Bag X021, Vanderbijlpark~1900, South Africa

Supervisor: Dr. J. V Fonou-Dombeu

November 2019

Declaration

I, Gideon Koech, declare that, to the best of my knowledge, my dissertation is the result of my original work except otherwise stated. It has not been submitted in candidature for any degree in any university or institution. Due references in literature were stated and acknowledged wherever other sources were involved according to the standard referencing practices.

Gideon Koech **30th November 2019**

Acknowledgements

Special thanks go to my supervisor Dr. J.V Fonou-Dombeu, for his encouragement and for devoting his time to supervise my thesis. For my welfare throughout my stay here in South Africa, I thank my brother Mr. Lawrence Koech and Mr Robert Tewo. I could not have accomplished my study program without their constant support, encouragement and being my mentors. The constant and continuous support from my parents and friends who helped me throughout the whole study period is highly appreciated.

Abstract

The increasing demand for knowledge representation and exchange on the semantic web has resulted in an increase in both the number and size of ontologies. This increased features in ontologies has made them more complex and in turn difficult to select, reuse and maintain them. Several ontology evaluations and ranking tools have been proposed recently. Such evaluation tools provide a metrics suite that evaluates the content of an ontology by analysing their schemas and instances. The presence of ontology metric suites may enable classification techniques in placing the ontologies in various categories or classes. Machine Learning algorithms mostly based on statistical methods used in classification of data makes them the perfect tools to be used in performing classification of ontologies.

In this study, popular Machine Learning algorithms including K-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forest, Naïve Bayes, Linear Regression and Logistic Regression were used in the classification of ontologies based on their complexity metrics. A total of 200 biomedical ontologies were downloaded from the Bio Portal repository. Ontology metrics were then generated using the OntoMetrics tool, an online ontology evaluation platform. These metrics constituted the dataset used in the implementation of the machine learning algorithms.

The results obtained were evaluated with performance evaluation techniques, namely, precision, recall, F-Measure Score and Receiver Operating Characteristic (ROC) curves. The Overall accuracy scores for K-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forest, Naïve Bayes, Logistic Regression and Linear Regression algorithms were 66.67%, 65%, 98%, 99.29%, 74%, 64.67%, and 57%, respectively. From these scores, Decision Trees and Random Forests algorithms were the best performing and can be attributed to the ability to handle multiclass classifications.

Publications

1. Koech, G.K. and Fonou-Dombeu, J.V. (2019) K-Means Clustering of Ontologies Based on Graph Metrics," *IEEE International Multidisciplinary Information Technology and Engineering Conference*, November 21 – 22, Vanderbijlpark, South Africa, Accepted for presentation and publication.
2. Koech, G.K. and Fonou-Dombeu, J.V. K-Nearest Neighbors Classification of Semantic Web Ontologies, *46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020)*, January 20-24, Limassol, Cyprus, Submitted.
3. Koech, G.K. and Fonou-Dombeu, J.V. Binary Classification of Semantic Web Ontologies with Logistic Regression, *23rd International Conference on Business Information Systems (BIS2020)*, June 8 -10, Colorado Springs, USA, Submitted.
4. Koech, G.K. and Fonou-Dombeu, J.V. A Probabilistic Classification of Ontologies with Naïve Bayes, *2020 International Conference on Information Communications Technology and Society (ICTAS)*, March 11-12, Durban, South Africa, Submitted.
5. Koech, G.K. and Fonou-Dombeu, J.V. Support Vector Machine Based Classification of Ontologies, *33th International Conference on Industrial, engineering & Other Applications of applied Intelligent Systems (IEA/AIE 2020)*, July 21-24, Kitakyushu, Japan, Submitted.

Table of Contents

List of Figures

List of Tables

CHAPTER 1. BACKGROUND AND INTRODUCTION

1.1 INTRODUCTION

The semantic web is an extension of the current web, where information is represented more meaningfully both for humans and computers alike (Taye, 2010). Ontologies are the basic building blocks of the semantic web. An ontology is an explicit, formal specification of a shared conceptualization of a domain of interest (Horrocks, 2008). Here, 'formal' implies that the ontology should be machine-readable and 'shared' implies that it should be accepted by a group or community (Zhao et al., 2009). Ontologies play a role in solving the problem of interoperability between applications across different domains, by providing a shared understanding of common information (Gruber, 1993). This is in line with the goals of the semantic web since it promises to achieve better data automation, reuse, and interoperability (Taye, 2010).

Ever since the introduction of the semantic web, ontologies have been growing in large numbers (Zhao et al., 2009). Many ontology libraries currently exist, hosting various ontology files such as, Ontolingua, the DAML library, the Protege OWL library, etc (Sridevi & Umarani 2013). Sridevi & Umarani (2013) also point out that the ontology search facilities provided by these libraries are only limited to keyword searches that makes it difficult for a developer to select the relevant or the best ontology to work with.

To achieve the effective level of knowledge reuse, a requirement is that the search engines be capable of finding the specific ontologies that the users are looking for (Horrocks, 2008). Classification of ontologies makes a significant contribution in providing an understanding that will help in the selection of ontologies (Netzer et al., 2009). Various ranking and classification efforts including classifying ontologies according to the minimal description logic covering all the constructors used have been made (Fábio et al., 2006). Although some ontology search engines classify the ontologies only according to specific search terms more is required (Sridevi & Umarani, 2013). Artificial intelligence (AI) algorithms have not been used in the classification of ontologies even though data mining and machine learning techniques have been used in the classification of text documents (Aurangzeb et al., 2010).

1.2 RATIONAL AND MOTIVATION

The internet is currently the main source of information for most users (Khani et al., 2012). There has been an increase in the number of documents stored on the internet in electronic form (Bilski, 2011). A point of consideration is that machines do not understand the published

documents that are stored on the web. This makes it difficult for a specific user to extract only specific information from the web (Bonino, 2005). Machine learning, which is a branch of AI, involves a set of methods that can automatically detect patterns in data, and then use these patterns to perform other kinds of decision making under uncertainty, or to predict future data (Vandana & Mahender, 2012). Data mining and Machine learning techniques have proven to be powerful classification methods (Bilski, 2011).

With the rise of the Semantic Web, ontologies are becoming the most suitable way of representing, dealing and reasoning with large volumes of information in several domains (Brewster & O'Hara, 2007). As more information is being converted and integrated into Resource Description Framework (RDF) and Web Ontology Language (OWL), carefully designed OWL ontologies are essential for the effective management, reuse, and integration of these data (Sridevi & Umarani, 2013). The creation, evaluation, and maintenance of ontologies have, therefore, become an engineering process that needs to be managed and measured using sound and reliable methods (Netzer et al., 2009).

The evaluation of ontologies entails checking some of its various aspects, including measuring its design complexity which occurs as more ontologies are being developed in real-world applications (Zhang et al., 2010). Ontology classification work has mainly focused on classifying them by considering its expressivity and complexity of reasoning based on the description logic ontology (Fábio et al., 2006).

Various researchers have pointed out the need to classify ontologies to ascertain their selection, reuse, maintenance as well as understand them better (Yang et al., 2006a). As pointed out by Aurangzeb et al. (2010), data mining and machine learning techniques work in conjunction to automatically classify and discover existing patterns in electronic documents. With the existence of ontology metrics that specifically checks for the complexity of ontologies, existing AI algorithms will, therefore, work best in classifying ontologies based on their complexity metrics.

1.3 PROBLEM STATEMENT

Recently there has been an expansion of the semantic web. In order to represent and integrate knowledge and data in real-world applications, more large-scale ontologies have been developed (Gruber, 1993). However, with the production of more ontologies, arises the need to measure their complexity to enable the developers to better understand, maintain, reuse and select the best-suited ontologies (Zhang et al., 2006b).

According to Vandana & Mahender (2012) Classification systems have proved to play a big role in the advent of the evolution of the internet since the meaning and accessibility of text documents and electronic information have increased (Bilski, 2011). Proper classification of the ontology metrics at the ontology-level and class-level requires applications of AI algorithms so that the required and desired ontology is retrieved (Srinivasulu et al., 2014).

This research investigates AI algorithms that are best suited for the classification of semantic web ontologies. The algorithms will then be applied to compute metrics to classify the ontologies in a domain based on their complexity metrics. The successful classification of the ontologies would help ontology developers in understanding and selecting the best ontologies to work with.

1.4 AIM AND RESEARCH OBJECTIVES

The aim of this research is to investigate AI algorithms for classifying semantic web ontologies based on their complexity metrics.

The objectives of this research are:

- To investigate AI algorithms for classification of ontologies.
- To investigate existing complexity metrics of ontologies.
- To compute the complexity metrics of selected semantic web ontologies.
- To implement AI algorithms to classify ontologies based on their complexity.
- To evaluate the performance of the implemented AI algorithms.

1.5 RESEARCH METHODOLOGY

1.5.1. Data Collection

In this study, a literature search was used to collect the data. Journal articles, conference papers, and books that focus on AI classification techniques and design complexity of ontologies were targeted. The components making up the dataset were metrics of ontologies obtained from available public repositories. Here, public domain ontologies from online repositories was downloaded and thereafter their metrics were generated using reliable platforms.

1.5.2. Experiments

1.5.2.1. Implementation

The computation of ontology metrics for the selected ontologies and application of AI classification algorithms were done using existing semantic web and Machine Learning

platforms. Some of the most popular of these platforms include: Sesame, Ontomentrics, Jena API and Protégé, etc. (Magkanaraki et al., 2002; Ramanujam et al., 2009; Zhou, 2010).

In this study, the OntoMetrics platform was used in generating ontology metrics. OntoMetrics is a web-based ontology validation platform that allows users to measure the suitability of existing ontologies, regarding the requirements of their systems (Lozano-Tello & Gómez-Pérez, 2004). The platform is therefore very interactive and user-friendly. The implementation of Machine Learning algorithms was done using Python's Jupyter Notebook environment which is a web application containing live code, visualizations, and narrative texts. Jupyter Notebooks works by encapsulating both documentation and source code along with the source code output. The live cells allow text editing as well as the interactive execution of the source code cells (Schroder et al., 2019).

1.5.2.2. Evaluation

Various metrics were investigated to evaluate the performance of the implemented AI algorithms for classifying semantic web ontologies. Some of the evaluation techniques that were used in this study included the Confusion Matrix; precision, recall, F-measure and the Receiver Operating curve (ROC) (Maynard et al., 2006).

- **Confusion Matrix -** This is a matrix table for measuring the performance of supervised Machine Learning algorithms. The rows of the confusion matrix represent the instances of the actual class used and the columns, the instances of the predicted class (Luquea et al., 2019). The elements of a confusion matrix represent the: number of positives correctly identified or True Positive (TP), number of negatives identified correctly or True Negative (T N), number of negatives incorrectly identified as positive or False Positive (FP) and number of positives incorrectly identified as negatives or False Negative (FN). Other evaluation techniques that can be obtained from the confusion matrix table include accuracy, precision and recall and F-measure (Trajdos & Kurzynski, 2017; Luquea et al., 2019).
- **Accuracy -** The accuracy is defined as the ratio of the total number of correct predictions to the total number of predictions made. On the other hand, the error rate is the difference in accuracy scores.
- **Precision -** Precision is defined as the ratio of True Positives (correctly identified items) to the sum of True Positives and False Positives. It is a measure of the capacity of the correct information returned by the model in percentages.

- **Recall -** Recall is the quotient of True Positives (Correctly identified items) and the sum of True Positives and False Positives. It, therefore, measures the capacity of all the relevant information that a model has extracted in percentages.
- **F-Measure -** F-Measure combines both precision and recall and can be defined as the harmonic mean of precision and recall measures.
- **Receiver Operating Characteristic (ROC) Curve -** Some classification problems cannot be measured with classification scores, mainly when dealing with datasets with heavy class imbalance. Receiver Operating Characteristic (ROC) curve is useful in such cases, offering the best alternatives (Nazrul, 2018). ROC is a plot of True Positive Rate (TPR), which is recall or sensitivity on the x-axis against False Positive Rate (FPR) also known as 1-specificity on the y-axis.

  After plotting the curve, the model performance is determined by looking at the area under the ROC curve (AUC). AUC value of 1 is considered as the best possible that can be obtained in a model while the worst is 0.5. The AUC value that is less than 0.5, would call for reversing the recommendations of the model with the aim of getting a value that is above 0.5. Using ROC curves to check the performance of a model allows for comparison of curves of different models directly or for different thresholds, whereas, the AUC can be used as a summary of the model skill (Brownlee, 2018.).

1.6 DISSERTATION OUTLINE

The rest of this paper has been structured as follows: Chapter 2 is Literature Review and presents a background of Semantic Web and Ontologies, Machine Learning Algorithms and OntoMetrics platform. The chapter also presents work that are related to classification with the most popular Machine Learning algorithms. Chapter 3 is Materials and Methods and Machine learning algorithms and performance evaluation techniques are discussed in detail. Chapter 4 discusses the System Architecture for the classification models built. Experimental results for the implementation of the algorithms in classification of ontologies are presented in chapter 5. The dataset used in the experiments as well as the software environment used in running the experiments are also outlined. Chapter 6 provides the Conclusion and Future Work.

1.7 ORIGINAL CONTRIBUTIONS

The contributions of this study are the design and implementation of various artificial intelligence models for the cluttering and classification of semantic web ontologies:

1. A K-Means model for the clustering of ontologies based on the Graph metrics presented in Chapter 5. This work was accepted for presentation and publication in the *IEEE International Multidisciplinary Information Technology and Engineering Conference*, Vanderbijlpark, South Africa, November 21 – 22, 2019.
2. A K-Nearest Neighbours model for classification of ontologies. This work has been submitted for review at the *46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020)*, Limassol, Cyprus, January 20-24, 2020.
3. A Logistic Regression model for classification of ontologies. This work has been submitted for review to *23rd International Conference on Business Information Systems (BIS2020)*, Colorado Springs, USA, June 8 -10, 2020.
4. A Naïve Bayes model for classification of ontologies. This work has been submitted for review to *2020 International Conference on Information Communications Technology and Society (ICTAS)*, Durban, South Africa, March 11-12, 2020.
5. A Support Vector Machine model for classification of Ontologies. This work has been submitted for review to *33rd International Conference on Industrial, engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2020)*, Kitakyushu, Japan, July 21-24, 2020.
6. Decision Trees and Random Forest models for classification of ontologies. These are presented in Chapter 5.

1.8 PUBLICATIONS

1. Koech, G.K. and Fonou-Dombeu, J.V. (2019) K-Means Clustering of Ontologies Based on Graph Metrics," *IEEE International Multidisciplinary Information Technology and Engineering Conference*, November 21 – 22, Vanderbijlpark, South Africa, Accepted for presentation and publication.
2. Koech, G.K. and Fonou-Dombeu, J.V. K-Nearest Neighbors Classification of Semantic Web Ontologies, *46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020)*, January 20-24, Limassol, Cyprus, Submitted.
3. Koech, G.K. and Fonou-Dombeu, J.V. Binary Classification of Semantic Web Ontologies with Logistic Regression, *23rd International Conference on Business Information Systems (BIS2020)*, June 8 -10, Colorado Springs, USA, Submitted.

4. Koech, G.K. and Fonou-Dombeu, J.V. A Probabilistic Classification of Ontologies with Naïve Bayes, *2020 International Conference on Information Communications Technology and Society (ICTAS),* March 11-12, Durban, South Africa, Submitted.
5. Koech, G.K. and Fonou-Dombeu, J.V. Support Vector Machine Based Classification of Ontologies, *33rd International Conference on Industrial, engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2020),* July 21-24, Kitakyushu, Japan, Submitted.
6. Koech, G.K. and Fonou-Dombeu, J.V. Decision Trees and Random Forest Classification of Ontologies: A Comparative Study, Manuscript under preparation.

CHAPTER 2. LITERATURE REVIEW

2.1 Introduction

In this chapter, a background of the semantic web is discussed. Ontologies, being the basic building blocks of the semantic web and the tools and languages used in building the semantic web are discussed. Ontology metrics are also introduced in this chapter and the specific metrics used in conducting this study, OntoMetrics, is also introduced. The chapter points out the issue regarding the complexity of ontologies brought about by the growing number of ontologies and how ontology metrics are used in understanding the functionalities of an ontology. The chapter further discusses the concept of Classification of ontologies based on their metrics using Machine Learning algorithms.

2.2 The Semantic Web and Ontology

2.2.1. Semantic Web

The evolution of the first web known as web 1.0 to web 2.0, the current web, was motivated by the increasing need to share resources electronically. Contents on the current web, however, are designed specifically for humans and cannot be understood by a computer. Computers, therefore, have no reliable way to process the semantics on the web but rather only process their syntax (Gruber, 1993). This, with the increasing need for interoperability between different web pages, has led to the introduction of a new version of the web known as the semantic web.

The semantic web was introduced by Tim Berners-Lee and is an extension of the current web. It seeks to bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can easily carry out sophisticated tasks for users (Berners-Lee, 2011). This is achieved by providing a description of its contents and services in a way that is machine-readable and one that allows services to be annotated, discovered, published, advertised and composed automatically (Taye, 2010).

The semantic web is not separate from the current web since both are categorized as a set of resources that are uniquely identified by the URI. A major dissimilarity between the two, however, is that whilst the current web uses HTTP to display contents of a page, the semantic web semantically represents data in resources and therefore ensuring machine readability (Gerber et al., 2008).

2.2.2. Ontology

The concept of ontology originated from metaphysics and philosophical science. Early philosophers defined it as "the science of being" as they sought answers to underlying factors like what existence is and the properties that can best describe existence (Gruber, 1993). Recently, ontologies have become more formalized conceptual models applied in artificial intelligence, information systems, natural language processing, database integration, information query systems, agent software systems and web technologies (Yao et al., 2005). According to Gruber (1995), ontologies provide an explicit specification of a conceptualization and can formally be defined as a representation of knowledge of a specific domain and consists of vocabulary representing the domain in form of classes or concepts, properties or relationships existing between these classes.

Formal languages have been established for the encoding of ontology knowledge. The languages fall into three broad categories: vocabularies of natural languages, object-based, and description logics (Stevens et al., 2000). Natural languages-based ontology vocabularies are loosely structured hierarchies of terms. Object-based languages also known as frame-based ontology languages are rigidly structured with each frame (concept) described by a collection of slots (attributes) (Fernández, 1999). Description Logics (DL) languages are based on concepts and relations used to automatically classify taxonomies (Gonzalez-Castillo et al., 2001).

As the Semantic Web is gaining momentum, more ontologies are being developed to represent and integrate knowledge and data. The growth of ontologies in size and numbers makes them more complex and difficult to understand. Assessing the quality of an ontology is therefore essential to ontology developers since it allows them to identify the areas that may require extra work (Jose et al., 2011). This also enables ontology users to know the exact section of the ontology that may lead to problems if not addressed and after making comparisons with various ontologies, they may be able to choose and work on the best ontologies (Tartir et al., 2005).
Existing methodologies for building ontologies propose a phase of reusing the ontology as indicated by Fernández (1999). However, there are no frameworks that indicate to the user the process of choosing ontologies for a new project and no methodologies quantifying the suitability of these ontologies for the system (Fernández, 1999). A metric quantifying the suitability of an ontology would alleviate the problem by showing how appropriate the ontology is to the user.

2.2.3. Semantic Web Architecture

Tim Berbers-Lee (2001), proposed a layered architecture, which adheres to software engineering principles and the fundamental aspects of layered architectures, for the semantic web The architecture guides ontology developers in the development of the semantic web as well as solving any issues regarding the implementation of the semantic web applications. Figure 1 shows the semantic web architecture proposed by Tim Berners-Lee.

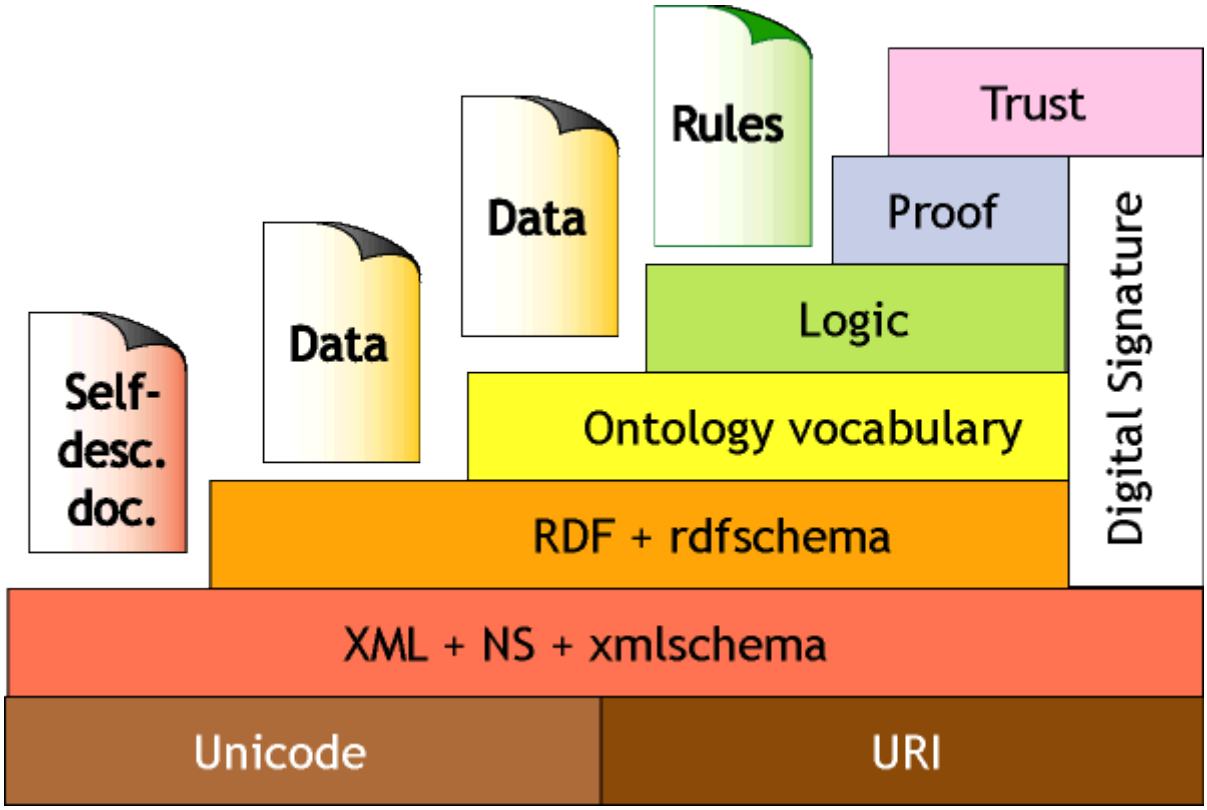

Figure 2:1: Semantic Web Architecture proposed by Tim Berners-Lee (2001)

The URI and Unicode occupies one layer that identifies and locates resources on the web with the help of the URIs. The resources can then be identified by these unique names and the Unicode is a standard for computer character representation. The second layer is the Extensible Markup Language (XML). It is a machine-readable markup language that has its own format. Due to its flexible text format, XML describes data in the WWW community and exchanges different types of data on the web.

Resource Description Framework (RDF) is the third layer of the semantic web that provides a framework for describing the semantics of information about the resources on the web in a way that is understood by machines. Identification of web resources is done with the help of URIs

and a graph model is deployed to describe the relations that exist between these resources. The fourth layer is Ontology vocabulary that delivers grammar and vocabulary for data on the web by breaking down the semantics of data used in keeping ontologies. It, therefore, provides a consistent way of communication and common understanding by different parties.

The fifth layer is represented by Logic and proof that provides logic constraints that checks the structure of an ontology. Consistency problems and redundancy are some of the issues that may arise, and a reasoner is used to check for these. The last layer of the architecture is Trust. This layer checks for the reliability of the information on the web such that it can ensure its quality.

2.2.4. Semantic Web Tools and Languages

A classical ontology contains a hierarchical description of important concepts and their relations in a domain, task or service. The formality extent that is used in describing intuitively these descriptions may vary (Narzary & Nandi, 2014). However, increasing the formality and regularity will enable for machine understanding. A powerful ontology language that can be used effectively in formalizing the structure of the web is greatly useful and of importance to the development of the Semantic Web (Li & Horrocks, 2004).

The process of developing an ontology consists of some rules that are strictly stipulated by the various ontology languages used in developing it. Such guidelines are useful to the ontology developers and users in developing the best and effective ontologies and selecting the best among the public ontologies to work. Some of the most common guidelines are stated below (Berners-Lee, 2011):

- A language should be well designed for the intuition of human users while maintaining its adequate expressive power.
- A language should be well defined with clearly specified syntax and formal semantics.
- A language should also be compatible with existing web standards, etc.

There has been an increase in the development of ontologies within the last decade and to cope with this trend various ontology tools and languages have been developed that can suit different application domains. Some of these languages are based on the extensible Makeup Language (XML) (Connolly et al., 2001). Resource Description Framework (RDF) and Resource Description Framework Schema (RDF Schema) languages were created by World Wide Web Consortium (W3C) group members. The union of RDF and RDF Schema has enabled the creation of supplementary languages which are Ontology Inference Layer (OIL) and Darpa

Agent Markup Language (DAML + OIL) and this has led to the improvement of their features (Li & Horrocks, 2004).

XML, RDF, RDFS, DAML+OIL and Web Ontology (OWL) languages have the capability of linking content documents and grouping them in a logical and relevant way and are mainly used in organizing, integrating and navigating through the Web. They create context-aware computing systems whereby users can search for information in an instinctive way (Borgida & Patel-Schneider, 1994). A brief description of these languages has been discussed below:

- **eXtensible Markup Language (XML)**

XML is a tag-based language for describing tree structures with a linear syntax. The language provides users with a platform for defining their own tags needed for describing the structure of the web documents (Klein et al., 2000). In this way, the content of a web document can be processed automatically. XML also provides a means for exchanging information over the web making it a basic language for the semantic web (Narzary & Nandi, 2014).

- **Resource Description Framework (RDF)**

The structure of information is the main area of focus of the XML language. However, information is not only about the structure but also the semantics that lies in it. XML does not define the semantics of information in a way that can be understood and processed by a machine. Resource Description Framework (RDF) is a language that performs these functions. It is an XML application that has been tailored to add meta-information to documents on the Web (Lassila & Swick, 1999). It defines a model of data used for describing the semantics of data. RDF defines object-property-value-triples that represent the semantics of web resources and introduces a standard syntax for them (Narzary & Nandi, 2014). Since RDF statements are also resources, statements can be alternatively applied to statements, therefore, allowing for nesting properties.

- **Resource Description Framework Schema (RDF Schema)**

RDF Schema is a language encrusted on top of the RDF language. This approach referred to as “Semantic Web Stack” has been presented by the W3C organization and Tim Berners-Lee and puts together the various concepts that are all related to each other. RDFS also expresses class-level relations that describe instance-level relations that belong to the various concepts (Haase et al., 2004; Berners-Lee, 2011). RDF Schema uses the same data model used by the

various object-oriented paradigms. An example of this is the programming language like Java. The model allows for the creation of some specific information within a domain (Brickley & Guha, 1999).

- **Ontology Inference Layer (OIL)**

OIL has been developed in the context of the European IST project On-To-Knowledge. OIL provides modelling primitives used in frame-based and Description Logic oriented ontologies, along with a simple and clean semantics (Li & Horrocks, 2004). The syntax of OIL makes use of RDF(s) and XML(s) such that it can maintain backward compatibility. The language, according to Li & Horrocks (2004), merges three crucial aspects that are provided by different communities. First is the Formal semantics and efficient reasoning support provided by Description Logics. Second is the Epistemologically rich modelling primitives provided by the Frame-based community, and finally a standard proposal for syntactical exchange notations as provided by the Web community.

OIL provides the means for describing structured vocabulary using well-defined semantics. This is the main impact that has been brought about by OIL. OIL distinguishes three different layers to describe ontologies. These layers are the object level where concrete instances of an ontology are described. The first meta-level provides the actual ontological definitions where the terminology that may be instantiated at the object level is defined (Horrocks, 2000). The second meta-level is the final layer and is concerned with describing the features of an ontology such as the name, author, subject, etc. OIL adds a feature, reasoning support on the RDF where expressiveness of an ontology is increased to allow functionalities such as automatically checking for consistency on ontology data.

- **DARPA Agent Mark-up Language - Ontology Inference Layer (DAML+OIL)**

This is a mark-up language created jointly by the American and European ontology communities for the Semantic Web after they merged DAML-ONT and OIL (Horrocks, 2002). This language operates by using the standards of the languages that are already in existence which are XML and RDF (Connolly et al., 2001). It adds the ontological primitives of object-oriented and frame-based systems and the formal rigor of expressive description logic.
DAML+OIL implements an object-oriented approach with the structure of the domain being described in classes and properties terms and the set of axioms that assert characteristics of the respective classes and properties. The meaning of DAML+OIL is defined by a standard model-theoretic semantics based on interpretations, where an interpretation consists of a domain of

discourse and an interpretation function (Arroyo et al., 2014). DAML+OIL language has been accepted and widely used as an ontology language with enough expressiveness throughout the research community.

- **Web Ontology Language (OWL)**

OWL is a Web ontology language developed by the W3C Web Ontology Working Group. This language aims at making Web resources to be processed more easily in applications by adding information about the resources. OWL is mainly applied to situations where information on the web documents need to be processed by applications, instead of only being displayed to humans. OWL makes it is possible to represent the meaning of terms that are used in vocabularies and relationships between those terms (Dean et al., 2003).

OWL consists of three main components (Arroyo et al., 2014). Firstly, Ontologies are defined as a sequence of axioms and facts, plus inclusion references to other ontologies, which are included in the ontology. Secondly, Axioms are used to associate class and property IDs with either partial or complete specifications of their characteristics and to give other logical information about classes and properties. Lastly, Facts state information about individuals in the form of a class that the individual belongs to, together with its properties and values. Individuals can either be given an individual ID or be anonymous, referred to as blank nodes in RDF terms.

OWL uses additional vocabulary together with formal semantics to provide more facilities than XML, RDF, and RDF Schema in expressing meaning and semantics. It, therefore, has greater abilities for representing machine-interpretable contents as compared to other languages (Arroyo et al., 2014).

OWL strives to meet the needs of various users. To ensure this usability, OWL has been categorized into three sublanguages which are (Rajaei, 2007; Narzary & Nandi, 2014):

1. OWL-Lite. This consists of RDFS plus equality and 0/1-cardinality. Layered and easy-going language for tool builders. Developed to capture many of the commonly used features of DAML+OIL. It attempts to provide more functionality than RDFS, which is important in order to support web applications.
2. OWL DL. It contains the whole OWL vocabulary that is interpreted under a number of simple constraints. Primary among these can be found by the type separation. Class identifiers cannot simultaneously be properties or individuals. Similarly, properties cannot be individuals.

3. OWL Full. This is composed of the complete vocabulary but interpreted more broadly than in OWL DL. A class can be treated simultaneously as a collection of individuals and as an individual.

Besides the RDF style syntax, OWL specification also includes an abstract syntax that provides a higher level and a way that is much easier in writing ontologies. The language has the advantage of allowing a more concise statement of the semantics. An important observation is that the OWL abstract syntax has reverted to grouping axioms into frame structures (Rajaei, 2007). Semantic web languages aim at representing ontologies to allow computer programs to inter-operate without pre-existing, outside-of-the-web agreements and since OWL has a degree of effective reasoning mechanism, then computer programs can manipulate this interoperability information themselves (Narzary & Nandi, 2014).

2.3 Ontology Metrics

Ontology metrics are expected to give some insight for ontology developers and users. The complexities of ontologies should be managed both in engineering and at the display level to make the ontologies more tractable. Recently, real-world ontologies from public domains have been collected and used in the computation of metrics from various ontology metrics that have been proposed (Horrocks, 2002). Several metrics have proven to have the capability of revealing internal structures of ontologies and therefore can be successfully used in ontology quality control.

2.3.1. OntoMetrics

With the increase in the use of ontologies in various domains, arises a problem of selecting an appropriate ontology that can fit a domain of choice. Issues regarding the complexity of an ontology and knowledge of its structure pose a great challenge in making selections for ontologies. The methodologies for creating an ontology has always had a stage proposing for reuse of an ontology but lacks a way for indicating to the users the process followed in choosing an appropriate ontology (Fernandez, 1999). Ontology developers have been tasked with examining the characteristics of an ontology and basing on their knowledge and intuition, they then make selections of ontologies. To manage this, various ontology technological platforms have been created recently (Lozano-Tello & Gómez-Pérez, 2004).

Within the last 2 decades, various platforms that are aimed at managing the task of checking the functionality of an ontology have been developed. The Knowledge Systems Laboratory

(KSl) of Stanford University developed the Ontolingua Server in 1996 (Farquhar, 1996). Ontosaurus was developed in 1997 by the Information Sciences Institute (ISI) from the University of South Carolina (Swartout, 1997). WebOnto was developed in the Knowledge Media Institute (KMI) of the Open University (UK) (Domingue, 1998); In 2001, IST OntoKnowledge project developed OILed (Bechhofer, 2001). OntoEdit platform was then developed by the AIFB of the Karlsruhe University (Staab, 2000); Protégé2000 used for ontology development and checking ontology metrics was developed by the Stanford Medical Informatics (SMI) in the Stanford University (Noy, 2001) amongst others.

Lozano-Tello & Gómez-Pérez (2004) proposed the Ontometric method which is based on an Analytical Hierarchy Process (AHP) and adapts different processes for the reuse of ontologies. The Analytic Hierarchy Process (AHP) was developed by Saaty (1977) and is a powerful and flexible tool used for decision-making in complex multi-criteria problems. This method allows people to gather knowledge about a problem, to quantify subjective opinions and to force the comparison of alternatives in relation to established criteria.

OntoMetrics is a web-based ontology validation platform that allows users to measure the suitability of existing ontologies, regarding the requirements of their systems (Lozano-Tello & Gómez-Pérez, 2004). The metrics are divided into two categories: schema and instance metrics. The quality metrics as an example that can be used to evaluate the success of a schema in modelling real-world domains and can aid in reusing an ontology (Gómez-Pérez & Rojas-Amaya, 1999). In this case, the depth, breadth, and height balance of the schema inheritance tree can play a role in a quality assessment. Besides this, the quality of a populated ontology (Knowledgebase Metrics) can be measured to check whether it is a rich and accurate representation of real-world entities and relations. Lozano-Tello & Gómez-Pérez (2004), discussed the following detailed structures of Ontometric platform.

2.3.1.1. Schema Metrics

Schema metrics address the design of the ontology. The metrics falling under this category indicates richness, width, depth, and inheritance of an ontology schema so that it can be used to check whether the ontology design correctly models the knowledge it represents.

2.3.1.1.1. Relationship Richness.

It reflects the diversity of relations and their respective placements in the ontology. In this case, an ontology that contains many relations besides class-subclass relations is richer than a taxonomy that only contains class-subclass relationships. The relationship richness (RR) of a

schema is defined as the ratio of the number of relationships (P) defined in the schema, divided by the sum of the number of subclasses (SC) (which is the same as the number of inheritance relationships) plus the number of relationships.

$$\boldsymbol{PR} = \frac{|\boldsymbol{P}|}{|\boldsymbol{SC}| + |\boldsymbol{P}|}$$

From the formula, the results obtained will be a percentage signifying the number of connections that exist between classes that are rich relationships as compared to all the possible connections that can include rich relationships and inheritance relationships.

2.3.1.1.2. Attribute Richness.

The number of attributes that are defined for each class can indicate the quality of ontology design and the amount of information relating to the instance data. The assumption here is that the knowledge the ontology conveys is more when the slots are more. The metric is generated as the number of attributes for all classes (ATT) divided by the number of classes (C).

$$\boldsymbol{AR} = \frac{|\boldsymbol{ATT}|}{|\boldsymbol{C}|}$$

The value obtained here will be a real number representing the average number of attributes per class. This value provides an insight into the total amount of knowledge about the classes that are in the schema.

2.3.1.1.3. Inheritance Richness.

Inheritance Richness outlines the distribution of information across different levels of the ontology's inheritance tree or the fan-out of parent classes. This measure will indicate how well knowledge is grouped into different categories and subcategories in the ontology. It is defined as the average number of subclasses per class and the number of subclasses is defined as:

$$|\boldsymbol{H^C}(\boldsymbol{C_1 C_i})|$$

Where H is the number of inheritance relationships. The number of subclasses (C_1) for a class C_i is defined as:

$$\boldsymbol{IR} = \frac{\sum \boldsymbol{C_i} \in \boldsymbol{C}\ |\boldsymbol{H^C}(\boldsymbol{C_1 C_i})|}{|\boldsymbol{C}|}$$

2.3.1.1.4. Attribute-Class Ratio.

The relation between classes containing attributes and all classes is represented in this metric. The number of attributes are counted to ascertain whether it has enough class attributes or not. It is therefore defined by:

$$AttributeClassRatio = \frac{ClassesWithAttributes}{NumberOfClasses}$$

2.3.1.1.5. Equivalence Ratio.

It calculates the ratio between the similar classes and all the classes in ontology. It is represented as follows:

$$EquivalenceRatio = \frac{SameClasses}{NumberOfAllClasses}$$

2.3.1.1.6. Axiom / Class Ratio.

The metric calculates the ratio between axioms and classes. It is therefore generated as the average amount of axioms per class:

$$AxiomClassRatio = \frac{Axioms}{Classes}$$

2.3.1.1.7. Inverse Relations Ratio.

It describes the ratio between inverse relations and all the other relations within an ontology and it is calculated as follows:

$$InverseRelationsRatio = \frac{InverseObjectProperties + InverseFunctionalDataProperties}{AllObjectProperties + AllFunctionaldataProperties}$$

2.3.1.1.6. Class / Relations Ratio.

This metric is a ratio between the classes and the relations in the ontology. It is calculated as follows:

$$ClassRelationRatio = \frac{Classes}{Relationships}$$

2.3.1.2. Knowledgebase Metrics.

The way data is organized in an ontology is a significant measure of ontology quality because it can indicate the effectiveness of the ontology design and the amount of real-world knowledge represented by it.

2.3.1.2.1. Average Population

This metric indicates the number of instances compared to the number of classes. The metric is useful if the ontology developer is not sure if enough instances were extracted compared to

the number of classes. The average population (AP) of classes in a knowledgebase is defined as the number of instances of the knowledgebase (I) divided by the number of classes defined in the ontology schema (C).

$$\boldsymbol{AP = \frac{|I|}{|C|}}$$

The output of the above is a real number which shows how well the data extraction process was performed to populate the knowledgebase.

2.3.1.2.2. Class Richness

The distribution of instances across classes is the focus of this metric. The number of classes that have instances in the knowledgebase is compared with the total number of classes. This comparison will give an idea of how well the knowledgebase utilizes the knowledge modelled by the schema classes. It is therefore defined as the percentage of the number of non-empty classes (classes with instances) (C') divided by the total number of classes (C) defined in the ontology schema.

$$\boldsymbol{CR = \frac{|C'|}{|C|}}$$

2.3.1.3. Graph Metrics

2.3.1.3.1. Average Depth

The depth in graphs is a property related to the cardinality of paths existing in the graph. The Average depth indicates the degree to which the ontology has vertical modelling of hierarchies and is defined by:

$$\boldsymbol{m = \frac{1}{n_{p \subseteq g}} \sum_{J}^{P} N_{J \in P}}$$

Where: $N_{J \in P}$is the cardinality of each path J from the set of paths P in a graph g and $n_{p \subseteq g}$ is the cardinality of$P$.

2.3.1.3.2. Average Breadth

Breadth is the cardinality of levels. The Average breadth indicates the degree at which the ontology has horizontal modelling of hierarchies and is defined as follows:

$$\boldsymbol{m = \frac{1}{n_{L \subseteq g}} \sum_{j}^{L} N_{J \in L}}$$

Where: $N_{J\in L}$ is the cardinality of each generation j from the set of generations L in a directed graph g and $n_{L\subseteq g}$ is the cardinality of$L$.

2.3.1.3.3. Average Number of Paths

This metric is a representation of the quotient of the total number of distinct paths and the number of paths of graphs. It is generated as follows:

$$\boldsymbol{m = \frac{n_{DP\subseteq g}}{j}}$$

Where: $n_{DP\subseteq g}$is the cardinality of the set DP in the directed graphs g and the number of graphs$j$.

2.4. Related Study

2.4.1. Ontology Metrics

Lantow (2016) provided a theoretical background and the possible scenarios for using the Onto-Metrics platform. Lantow points out the need to have an automatically calculated metrics to be used in evaluating the quality of ontology to avoid the use of major resources in ascertaining its quality. Rule-based ontology evaluation technique is used in detecting modelling errors and cases of violation of ontology modelling guidelines. Metrics, on the other hand, are used as indicators for any quality problems that cannot be detected by the rule-based method. Since OntoMetrics platform gives free access to metric calculation and definition then it is an important tool evaluation of ontologies through the generation of its metrics.

A study by Gangemi et al. (2005) investigated the existing ontology-evaluation methods based on the perspective of their integration in a single framework. They then set up a formal model based on qualitative and quantitative measures for ontologies. The proposed model consists of a meta-ontology (O^2) which characterizes ontologies as semiotic objects and is complemented with an ontology of ontology evaluation and validation - oQual. Basing on O^2 and oQual, three main types of measures for ontology evaluation are identified. First is structural measures which are typical of ontologies represented as graphs. The second is functional measures which are related to the intended use of an ontology and of its components. Lastly are usability-related measures, that depend on the level of annotation of the considered ontology

Gavrilova et al. (2012) presented an approach aimed at creating teaching strategies for e-learning based on the principles of ontological engineering and cognitive psychology. The framework proposed seeks to develop a methodology where the design of ontology is evaluated by assessing its structure with several quantitative metrics. Visual mind-mapping and concept

mapping are used as powerful learning tools where balance clarity and beauty are integrated into ontology evaluation procedures. This approach has high applicability in education processes and can aid educators and students in creating high-quality ontologies.

Tartir et al. (2005) introduced OntoQA, which is an approach that analyses ontology schemas together with their populations and works by describing them through a well-defined set of metrics. Their study was motivated by the fact that the growing number of ontologies in different domains have made it difficult for ontology users and developers to determine if a certain ontology is suitable for their needs. The set of metrics can highlight key characteristics of an ontology schema as well as its population thereby enabling ontology users and developers to make an informed decision quickly.

2.4.2. Machine Learning Classification Techniques

Machine learning techniques based on statistical methods have been used in handling large volumes of data. Machine learning algorithms perform classification of data, which is a process of predicting an unknown category label with the aim of creating a distinct boundary between the objects based on the attributes and its features (Khan et al., 2010.). Some of the popular machine learning algorithms include Naïve Bayes, Support Vector Machines, Decision Trees, Random Forests, Logistic Regression, and K-Nearest Neighbours algorithm.

2.4.2.1. K-Nearest Neighbors

The research that was done by Destercke (2012) who proposed an approach based on k-Nearest Neighbors algorithm that utilizes the imprecise probabilities and lower previsions. The approach is robust in the sense that the uncertain data is handled in a generic way and the decision rules that have been proposed in the theory enables the user to handle any conflicting information that exists between the neighbors. This also applies when there are no existing close neighbors.

The work of Amores et al. (2006) introduced a distance estimation technique by boosting and applied it to the KNN classifier. The study did not apply the AdaBoost as it is done usually in classification problems. Instead, the proposed technique is used in the learning of the distance function which is further used in the KNN algorithm. The results show that the proposed method outperformed the AdaBoost and the traditional KNN classifier.

In Sun & Houng (2010), an adaptive KNN algorithm that seeks to address the limitations of the traditional KNN algorithm is proposed. The traditional KNN algorithm normally identifies a similar number of nearest neighbors belonging to each sample test set. The algorithm seeks

to find the most suitable value of k which is the number of the least near neighbors that can be used by the training set to obtain the correct class label. In performing the classification of each of the test sets, the value of k is set as the same as the suitable value of k of the nearest neighbors in the training set. The results of the experiments show that the adaptive algorithm is much better than the traditional KNN algorithm.

The study by Agrawal (2014), focuses on the issues surrounding the classification of the uncertain data using the K-Nearest Neighbors algorithm. Uncertain data consists of tuples with different data. Therefore, finding a class of similar tuples is always a complex and tedious process. The data attributes with a higher level of uncertainty should be approached differently as compared to those with a lower level of uncertainty. The existing techniques used in the classification of these data have been underlined and the issues surrounding KNN have also been addressed.

A framework has been introduced by Potamias et al. (2010) that is used in the processing of KNN queries in the probabilistic graphs. The proposed framework is designed based on sampling techniques and during querying; the search space is eliminated with the use of novel techniques. The experimental results perform much better than the previously used techniques in identifying the true neighbors in real-world biological data.

All the research work discussed above have focused on using the KNN algorithm in performing classification of data or improving its performances. Of all these, however, there has never been any research that is aimed at classifying ontologies using The KNN algorithm. The work presented here is the first of its kind in classifying ontologies using the KNN algorithm.

2.4.2.2. Support Vector Machines

The application of SVMs for classification has been of interest to authors in various domains. Ali et al. (2016) presented a classification technique for feature reviews identification and semantic knowledge for opinion mining based on SVM and Fuzzy Domain Ontology (FDO). The proposed system collects reviews about a hotel and its features. Thereafter, SVM is applied to identify these hotel feature reviews and filter out irrelevant features in these reviews. Furthermore, FDO is used to compute the polarity terms of each feature. The experimental results show that the combination of FDO and SVM increases the precision rate of reviews and word extraction as well as the accuracy of opinion mining.

Another study by Vinayagam et al. (2004) developed a large-scale annotation system that performs rapid, reliable and accurate function assignments of gene products. In this work, the annotation was done through the gene ontology terms by applying SVM to classify the correct and false predictions. An organism wise cross-validation technique was done to define the confidence estimates. These results showed that the prediction performance was organism-independent and could manually generate high-quality annotations for other systems.

A combination approach that enhances the existing sentiment classification approaches based on an ontology is proposed in Shein & Nyunt (2010). The main aim of the sentiment classification is to aid in extracting the features on which various reviewers express their opinions and categorizing them into either positive or negative. The classifications of the reviews were made using Natural Language Processing (NLP) techniques, Formal Concept Analysis (FCA) ontology-based and the SVM algorithm.

The authors in Ali et al. (2017), presented an SVM and fuzzy ontology-based semantic knowledge system that filters web content automatically and blocks sites containing pornographic materials. The system uses a blacklist of censored webpages to classify the Uniform Resource Locators (URLs) into adult URLs and medical URLs. Thereafter, the web contents are extracted with a fuzzy ontology to determine the type of the website being viewed. The evaluation results of the system revealed its efficiency in classifying web content and performing automatic detection and blocking.

In Zhou et al. (2017), an ontology-driven framework that supports SVM-based hyperspectral data classification is presented. A dimension reduction algorithm is used to automatically select the prominent spectral characteristics for all the land covers in a hyperspectral image. These characteristics include the ranks and weights, which indicate the wavebands that distinguish a specific land cover mass from others. Thereafter, an ontology called HIC-Ontology is developed to represent the extracted spectral characteristics to support the final training and classification process. The results showed that the proposed technique achieves better performance in the classification of hyperspectral data.

To the best of our knowledge, no previous study has attempted to apply AI techniques to classify ontologies. This is the first study to experiment with the application of an AI algorithm, SVMs in this case, to classify semantic web ontologies based on their complexity/graph metrics.

2.4.2.3. Decision Trees

Authors have applied the decision tree algorithm to classify data in various domains. In Yao et al. (2005), a robust and practical decision tree improved model R-C4.5 and its simplified version are used in the classification of a healthcare dataset to predict inpatient length of stay. The model is based on the C4.5 and the attribution selection and partitioning methods. It avoids the fragmentation appearances by uniting branches that have poor classification effect. The experiments show that the R-C4.5 decision tree model and its simplified version enhances the interpretability of splitting attribute selection, reduces the numbers of insignificant or empty branches and avoids the appearance of overfitting.

Another study by Rastogi & Shim (2000) proposed an improved decision tree classifier called PUBLIC that works by integrating the second "pruning" phase with the initial "building" phase. In PUBLIC, a node is not expanded during the building phase if it is determined that it will be pruned during the subsequent pruning phase. In making this determination for a node before expansion, the classifier computes the lower bound on the minimum cost subtree rooted at the node. The estimate made here shall then be used by the PUBLIC classifier in identifying the nodes that are certain to be pruned. The Experimental results obtained from the implementation of real-life and synthetic data sets have yielded positive results.

The work of Chandra & Paul (2006) presented a novel approach for the choice of the split value of attributes. In their work, the issue of reducing the number of split points has been addressed. This approach was implemented on various datasets that were taken from the UCI machine learning data repository. The results obtained from the experiments conducted showed that this approach gives a better classification accuracy as compared to already existing Decision Tree algorithms such as C4.5, SLIQ, and Elegant Decision Tree Algorithm (EDTA) and at the same time the number of split points to be evaluated is much less compared to that of SLIQ and EDTA.

Mehta et al. (1996) discussed issues surrounding the building of a scalable classifier and some of the issues discussed including the existence of classification algorithms designed only for memory-resident data which limits their suitability for data mining on large datasets. A design of SLIQ was presented in this study, which is a decision tree classifier that can handle both numerical and categorical attributes. This algorithm uses a novel pre-sorting technique in a tree-growth phase which is integrated with a breadth-first tree growing strategy to enable classification of disk-resident datasets. SLIQ algorithm also uses an inexpensive tree pruning

algorithm which results in a compact and accurate tree making it an attractive tool for data mining.

Chandra et al. (2002) proposed an algorithm, Elegant, that aimed at improving the performances of the SLIQ decision tree algorithm proposed by Mehta et al. (1996). A limitation of the SLIQ algorithm is that many Gini indices must be computed at each node of the decision tree. In order to decide which attribute is to be split at each node, the Gini indices must be computed for all the attributes and for each successive pair of values for all patterns which have not been classified. In the proposed algorithm, the Gini index is computed not for every successive pair of values of an attribute but over different ranges of attribute values. Classification accuracy of this technique was compared with the existing SLIQ and the Neural Network technique on three real-life datasets consisting of the effect of different chemicals on water pollution, Wisconsin Breast Cancer Data, and Image data. It was observed that the decision tree constructed using the proposed decision tree algorithm gave far better classification accuracy than the classification accuracy obtained using the SLIQ algorithm irrespective of the dataset under consideration.

A study done by Ruggieri (2002) proposed an analytic evaluation of the runtime behavior of the C4.5 decision tree algorithm. Three strategies for computing the information gain of continuous attributes are implemented. The strategies perform a binary search of the threshold in the training set, starting from the local threshold computed at a node. The first strategy computes the local threshold using the C4.5 algorithm and sorts the cases by quicksort method. The second strategy adopts a counting sort, whereas, the third strategy calculates the local threshold using the main memory version of the RainForest algorithm. The results obtained are seen to have a performance gain that is up by five times the normal C4.5.

2.4.2.4. Random Forest

The Random Forest Algorithm has been used for the classification of data in many domains. In Paul et al. (2018), an improved random forest classifier that performs classification with a minimum number of trees is proposed. The method removes the unimportant features and based on the number of important and unimportant features, a novel theoretical upper limit on the number of trees to be added to the forest is formulated to improve the classification accuracy. The algorithm is seen to converge with a reduced and important set of features. This method was implemented in the classification of histopathological datasets of breast cancer to detect the presence of mitotic nuclei. The method is also applied to the industrial datasets of

dual-phase steel microstructures to classify different phases. On these benchmark data sets, the results obtained show a significant reduction in the average classification error.
Another study by Xu et al. (2012), proposed an improved random forest algorithm for image classification. A novel feature weighting method and tree selection method are developed and synergistically served for making random forest framework well suited to classify image data with many object categories. The new feature weighting method for subspace sampling and tree selection method allows for the reduction of the subspace size thereby improving the classification performance without increasing error bound. Experimental results on image datasets with diverse characteristics have demonstrated that the proposed method could generate a random forest model with higher performance.

In the work of Alam & Vuong (2013) the random forest algorithm is applied on an android feature dataset to classify the applications as either malicious or benign. Since android is the most popular smartphone platform today, it makes it the best choice for malware authors to obtain secure and private data. Besides the classification of these applications, Alam & Vuong, (2013) focused on the detection accuracy of the free parameters of the algorithm. The parameters include the number of trees, depth of each tree and number of the random features selected. On a 5-fold cross-validation, random forest algorithm performed very well with an accuracy of over 99% with an optimal Out-Of-Bag (OOB) error rate (Han et al., 2011) of 0.0002 for forests with 40 trees or more, and a root mean squared error of 0.0171 for 160 trees. An effective classification approach that is based on the random forest algorithm was developed by Zawbaa (2015) to classify fruits. Three fruits including apples, strawberry, and oranges were analysed and features such as the color, shape and scale-invariant feature transform (SIFT) were extracted. Image processing was used in reducing the color indexes. The experimental results showed that the random forest produced better classification accuracy than other machine learning algorithms.

A study by Ali et al. (2012) presented an algorithm based on the random forest that classifies vehicles detected by a multiple inductive loop system, developed for measuring traffic parameters in a heterogeneous and no lane disciplined traffic. Besides the classification of the detected vehicles as a bicycle, motorcycle, scooter, car, and bus, the scheme also counts them accurately under a mixed traffic condition. The evaluation of the algorithm was based on threshold values and signature patterns and the results from the prototype show that the random forest algorithm provides better accuracy compared to the threshold-based and signature-based methods.

2.4.2.5. Logistic Regression

The Logic Regression algorithm has been applied by authors in many domains. Aborisade & Anwar (2018), presented a web-based application for the classification of tweets of netizens using the Logistic Regression technique. The application is built on four main processes, namely, fetching tweets, pre-processing, text feature extraction and machine learning. Labelled tweets are used as training data and in the pre-processing phase involving the removal of URLs, punctuation and stop words, tokenization, and stemming. The application then automatically converts the pre-processed tweets into a set of features vector using Bag of Words and the Logistic Regression algorithm is applied for the classification task. Using Confusion Matrix, the results showed that the accuracy of tweets classification into the selected topics is 92% which is considered very high.

Another study done by Indra et al. (2016) discussed the challenges of developing regression models that can predict fault-prone object-oriented classes in software projects. In fact, the design-complexity metrics have been successful in identifying a fault in software projects; however, the distribution of the metric varies across different projects, making it a difficult task to achieve predictions. In Indra et al. (2016) the authors used simple log transformations to make design-complexity measures more comparable among projects. The findings revealed that these transformations were useful in projects where data is not spread as compared to the data for building the prediction model.

Cruz & Ochimizu (2009) designed a pattern recognition system using a logistic regression model and few mapping functions. The performance of the proposed logistic regression model and mapping functions are assessed with a standard dataset from the UMASS database as well as datasets pertaining to wireless sensor network applications. The findings revealed that in most cases the recognition accuracy is enhanced by using the proposed mapping functions for both binary and multi-class pattern classification problems.

Rao & Manikandan (2016) did a study on the environment, customs, and health status of a community with the aim of finding the critical factors to prevent cerebral infarction. They used a rough set theory in reducing the attributes of the dataset and applied the association rules. The Logistic Regression was finally used in solving the shortcomings of too many rules causing attributes redundancy and reliability framework. The model was very effective in performing classification and the findings where the history of other cerebrovascular diseases, alcohol consumption, and seasonal change are the significant factors of cerebral infractions.

Yang et al. (2010) applied the Logistic Regression technique to classify the electromyography signals originating from the hand. The algorithm was implemented in three subjects using multinomial logistic regression and optimization heuristic based on gradient descent. The results reported an accuracy rate of 90.2%.

A large-scale analysis of the Logistic Regression algorithm by considering the hard classification problem of separating high dimensional Gaussian vectors was performed (Horn & Balbinot, 2015). The asymptotic distribution of the Logistic Regression classifier is evaluated based on the random matrix theory and high dimensional statistics. The findings revealed new insights on the internal mechanism of the Logistic Regression classifier which includes the bias in the separating hyperplane and the hyper-parameter tuning.

Mai et al. (2019) applied Logistic Regression to classify liver patients using gender and laboratory medical test data. The classification results showed that the Logistic Regression achieved better classification accuracy than other machine learning approaches.

2.4.2.6. Naïve Bayes

Authors have implemented the Naïve Bayes algorithm to perform classification in various domains. In Patil & Pawar (2012), the Naïve Bayes algorithm is implemented to classify websites based on the content of their home pages. The contents of the homepage including the title, meta keywords, pages descriptions, anchor labels, and other contents constituted the features were used in the classification.

Weil & Xiang-Yang (2010) proposed a new weighted Naïve Bayesian classifier model based on the information gain theory. The proposed model uses information gained from a set of attributes in the sample space to perform dimensionality reduction and assign the relative weight to each of the classification attributes. This approach strengthens the attributes that have a higher relationship and weakens those with lower relationships in the classification. Furthermore, the Naïve Bayes classifier is simplified by making it effective and improving its classification effect.

Kharya & Soni (2016), investigated the performance criterion of Naïve Bayes classifier considering a new weighted approach in the classification of a breast cancer dataset obtained from the UCI machine learning repository. The ranking performance is done in the decision-making process of the Naïve Bayes algorithm and its performance is improved by incorporating a weighted concept. The experimental results showed that a weighted Naïve Bayes approach outperforms the original Naïve Bayes.

The implementations of the Naïve Bayesian approach for the automatic classification of documents was also carried out (Kini et al., 2015). The documents considered were reports of technical research and the focus was on their text contents and analysis of results. The study aimed to find reliable text extraction techniques that can handle the ever-increasing electronic sources. Another study conducted the task of document classification with the Naïve Bayes approach (Jadon & Sharma, 2017). The study uses linear and hierarchical approaches to improve the efficiency of the classification model. The findings revealed that the hierarchical approach outperformed the linear classification technique.

Permana et al. (2016), used text mining and Naïve Bayes methods as an opinion classifier to measure student's satisfaction for educational institutions. The features used in the research were derived from the student's activities on social media which provides implicit knowledge and perspectives for the educational system. Sentiment analysis was used as a text-mining tool and the Bayesian classifier had an accuracy of 84% and a 16.49% difference from the existing evaluation systems.

Research by Lee et al. (2011) used a Naïve Bayes Multinomial classifier in the classification of twitter trending topics. These trending topics were classified into 18 general categories such as sports, politics, technology, and so forth. The experiments were carried out with two topic classification approaches, namely, bag-of-words for text classification and network-based classification. The text-classification method used word vectors of trending topics definitions and tweets, and the *tf-idf* weights to classify the topics, whereas, the network-based classification method-built categories of similar topics based on the number of common influential users. A similar study was done by Shubham et al. (2018) where the Naïve Bayes algorithm was implemented alongside other classification algorithms to classify tweets into different categories such as sports, politics, technology, and many more.

In light of the above, none of the previous researchers have attempted to classify semantic web ontologies with the Naive Bayes algorithm as is done in this study.

2.5 Conclusion

In this chapter, the Semantic web and ontologies, the hierarchy of the semantic web architecture and the tools and languages used in building ontologies have been discussed. The need for using metrics of an ontology in evaluating ontologies is also outlined. In this regard, the OntoMetrics platform for online generation of ontology metrics is introduced. Thereafter, the

literature on ontology metrics and Machine learning algorithms was presented where work related to ontology metrics and Machine Learning algorithms for classification were discussed.

CHAPTER 3. : MATERIALS AND METHODS

3.1 Introduction

This chapter discusses the Machine Learning techniques used in performing classification in detail. These techniques have been categorized into unsupervised and supervised learning where supervised learning uses an existing class label as a basis of classification. Unsupervised learning takes data as it is and without relying on some existing or provided labels tries to find an existing pattern in the data by learning its inherent structure. In discussing these techniques, mathematical and statistical methods making up the algorithms are discussed. Finally, the performance measures techniques are discussed where the techniques used in checking the performance of an algorithm is presented.

3.2 Machine Learning classification Techniques

3.2.1 Linear Regression

3.2.1.1 Simple Linear Regression

Problems that are in relation with regression techniques seeks to make a prediction of the value of a continuous response variable. Simple linear regression is used to model a linear relationship that exists between a response variable and a single explanatory variable (Richard et al., 2018). Simple Linear regression makes a major assumption that there exists a linear relationship between the response variable and the explanatory variable. It then creates a model of this relationship with a linear surface, referred to as a hyperplane which is a subspace of ambient space that contains it.

Two dimensions exist for simple linear regression, both for the response variable and another for the explanatory variable. The hyperplane has one dimension which is just one-dimension line. Simple linear regression consists of estimators which predict a value based on the data that is observed (Amand & Chris, 2018). It also comprises of the *fit()* and *predicts ()* methods that are used to learn the parameters of the model and to make a prediction of the value of a response variable. The fit method is implemented by the following equation:

$$y = \propto + \beta x \qquad (3.1)$$

Where:

y : The predicted value of the response variable.

x : The explanatory variable and the coefficients $\propto$ and β are the parameters of the model being learned by the learning algorithm.

Ordinary least squares is a technique of using training data to obtain the values of the parameters for simple linear regression by producing the best fitting model. The error created by the model is defined and measured by the term, cost function or a loss function. The difference between the predicted and the observed values are termed as residuals or training errors (Richard et al., 2018).

In order to improve the prediction of the model, the best approach is to minimize the sum of the residuals. In this way, the model will fit if the response variable values that it predicts does not differ much from the observed values of all training examples. The measure is referred to as residual sum squares cost function and it basically assesses the fitness of a model by making a summation of all the squared residuals for all the training examples (Richard et al., 2018). The residual sum of squares is calculated by the following formula:

$$SS_{res} = \sum_{i=1}^{n} (y_i - f(x_i))^2 \qquad (3.2)$$

Where:

y_i = The observed value

$f(x_i)$ = The predicted value

3.2.1.2 Multiple linear regression

Multiple linear regression contains several descriptive variables with each having a coefficient unlike having a single descriptive variable with a single coefficient as in simple linear regression (Amand & Chris, 2018). Multiple linear regression is represented by the model below:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \qquad (3.3)$$

This model can be given by the formula:

$$Y = X\beta \qquad (3.4)$$

Since the values of Y and X from the training data are known, then the values of β can be solved as follows:

$$\beta = (X^T X)^{-I} X^T Y \qquad (3.5)$$

It should also be noted that the value of β minimizes the cost function of the model.

3.2.1.3 Polynomial regression

This is an advanced model of multiple linear regression that adds terms that contain degrees that are more than one to the model. Transforming the training data by adding polynomial terms and then fitting them as the case with multiple linear regression results in obtaining the real-world curvilinear relationships that exists (Richard et al., 2018). A regression that has a third-order polynomial is depicted by the formula below;

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \qquad (3.6)$$

From the formula above, the descriptive variable or the input data is being transformed and then added as a final term to the model so that it can capture the curvilinear relationship. This equation is similar to that of multiple linear regression in a vector notation form. The resulting quadratic regression of this equation is a smooth curve that fits the training data better than for the other regression equations. The quadratic regression model is more accurate than a simple linear regression and since it can take any degree the model can be tried with any degree to draw comparisons. The implementation of Linear Regression model in classification largely depends on the vector space representing the dataset. The dataset might lead to the following issues.

- **Over-fitting**

As the degree increases, it will reach a point that the model will fit the training data almost exactly. The accuracy of the model with a degree that produces such results, however, is low. This is because as the degree increases, the model gets more complex, fitting the training data exactly and thus will not provide a correct approximation of the real relationship. This shortcoming is termed as over-fitting (Park et al., 2018).

- **Regularization**

This comprises several techniques applied to prevent over-fitting. It works by providing more information to an existing problem. This is always in the form of a penalty to act counter to the problem. Tikhonov Regularization (Ridge regression) is a method that deals with ill-posed problems by giving approximate solutions. It is a popular method for computing an approximate solution of linear discrete ill-posed problems with error-contaminated data (Park

et al., 2018). It acts on the residual sum of the least-squares function by adding the L2 norm of the coefficients, as depicted in the formula below:

$$RSS_{ridge} = \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \qquad (3.7)$$

Where:

$\boldsymbol{\lambda}$ = Hyper-parameter that controls the strength of the penalty.

Hyper-parameters are the parameters in a model which are set manually and not automatically generated. $\boldsymbol{\lambda}$ Is directly proportional to the penalty and an increase in its value results in an increase in the cost function. The ridge regression is equal to linear regression when the value of $\boldsymbol{\lambda}$ is equal to zero (Wang et al., 2018).

The Least Absolute Shrinkage and Selection Operator (LASSO) is an implementation in sci-kit learn which penalizes the coefficients by summing their L1 norm and the cost function as in the expression below:

$$RSS_{lasso} = \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \qquad (3.8)$$

Comparing the implementation of the two, LASSO leads to sparse parameters where almost all the coefficients will become zero while the ridge regression only generates coefficients that are small and nonzero (Wang et al., 2018). An implementation known as the elastic net is provided by sci-kit learn where it combines the L1 and L2 penalties in LASSO and ridge regression.

- **Gradient Descent**

This is an algorithm that is used as an optimizer to make an estimate of the local minimum that a function can generate (Sharma, 2018). The residual sum of the squares cost function is, therefore, given by the following equation:

$$SS_{res} = \sum_{i=1}^{n}(y_i - f(x_i))^2 \qquad (3.9)$$

The best parameters that can be used in linear regression are those that will produce the lowest cost function. Gradient descent works by finding the values of those parameters that will minimize the values of the cost function by iteratively updating the values of the model's parameters through the computation of the partial derivatives of the cost function (Theodoridis, 2015).

The plot of gradient descent is a three-dimensional convex like a bowl, consisting of all the possible parameters. The lowest point of the curve will be the local minimum and its points being the parameters of the model (Sharma, 2018). Depending on the number of training instances, there are two varieties of gradient descent:

- Batch gradient descent: this type of gradient descent utilizes all the available training instances to make updates in the model parameters in each round of iteration.
- Stochastic Gradient Descent (SGD): in this type, the parameters are only updated after a single training instance, which are always randomly selected.

3.2.1.4 Correlation Coefficients

- **Pearson correlation Coefficient**

This type of correlation was introduced by Karl Pearson and it works by measuring the linear association between variables that are continuous, displaying the data and fitting a suitable curve in it (Dalinina, 2017). It is the ratio of the covariance of two variables representing a set of numerical data that is normalized to the square root of their variances as shown in the equation below:

$$r = \frac{C_{xy}}{\sqrt{C_{xx}C_{yy}}} = \frac{C_{xy}}{\sigma_x \sigma_y} \qquad (3.10)$$

It, therefore, quantifies the degree under which a relationship between two variables can be described by a line (Chatillon, 1984). Using the two variables X and Y, the formula for calculating the Pearson correlation coefficient is as follows:

$$\rho(X,Y) = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2} \qquad (3.11)$$

- **Spearman's Correlation Coefficient**

A monotonic function is a function that is not affected by the increase in the values of its independent variables (Rodgers & Nicewander, 1988). It does not increase or decrease in this case. The following diagram shows the three states of monotonic function (increasing, decreasing and non-monotonic)

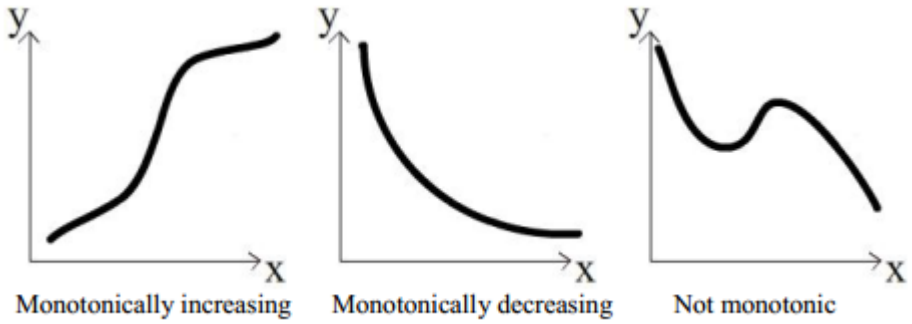

Figure 3:1: States of Monotonic function (increasing, decreasing and non-monotonic) (Wen et al., 2016)

Spearman's correlation coefficient measures the strength of a monotonic relationship in paired data (Chan, 2003). The computation of spearman's correlation is on ranks and not scores and is given by the following equation:

$$\rho(rank_x, rank_y) = \frac{cov(rank_x, rank_y)}{\delta_{rank_x}\delta_{rank_y}} \qquad (3.12)$$

A special case where there are no tied ranks in the dataset applies the following formula:

$$\rho_s = 1 - \frac{6\sum d_i^2}{N(N^2-1)} \qquad (3.13)$$

3.2.2 Logistic Regression

As discussed earlier, linear regression assumes that whenever there is a change in the explanatory variable, then the change results are the same or equivalent amount of change in the value of the response variable (Bewick et al., 2005). This kind of assumption is, however, invalid if the value of the response variable is a representation of a probability. The main difference between non-generalized and generalized linear models is that a generalized model does not carry along this assumption by creating a relation between the linear combination of the explanatory variables to the response variable by using a link function (Park, 2013).

Logistic regression is a linear model that is generalized and has a response variable that describes the probability that the outcome is a positive case. The positive class is only predicted

when the value of the response variable is equivalent to or greater than the discrimination threshold. If this does not hold, then the prediction made will be for the negative class (Bagley et al., 2001). A logistic function is applied in modelling the response variable as a function of a linear combination of the explanatory variables (Peng et al., 2002). The function gives a value that ranges between zero and one, (0, 1) and is given by the equation below:

$$F(t) = \frac{1}{1 + e^{-t}} \qquad (3.14)$$

Whereby, t is equivalent to a combination of explanatory variables in a linear pattern for logistic regression and is given by the following:

$$F(t) = \frac{1}{1 + e^{-(\beta_o + \beta_x)}} \qquad (3.15)$$

The equation above is as a result of:

$$t = \beta_o + \beta_x \qquad (3.16)$$

The inverse of the logistic function is the Logit function and it makes a linkage back to a linear combination of the explanatory variables by:

$$g(x) = ln\frac{F(x)}{1 - F(x)} = \beta_o + \beta_x \qquad (3.17)$$

Besides binary classification, a number of classification problems have more than two classes of interest in classification. The main objective of multi-class classification is to allocate an instance to a single set of classes. Sci-kit-learn library uses one-vs.-all or one-vs.-the-rest strategy to support multi-class classification. In this strategy, a single binary classifier is used in each possible class.

3.2.3 K-Nearest Neighbours

KNN is a supervised learning algorithm that relies on the fact that any objects that are similar do exist in proximity (Harrison, 2018). An appropriate distance function is, therefore, chosen to find the distance between these objects in a given dataset.

- **Distance Function**

The distance function is a real-valued function d, whereby for any given coordinates x, y and z, the following stands [10]:

1. $d(x, y) \geq 0$, and $d(x, y) = 0$ if and only if $x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, z) \leq d(x, y) + d(y, z)$

Property 1 above shows that the distance is always positive and not negative. It can, however, be zero if the coordinates on the plot are the same. Property 2 shows that the distance between two points will always be the same regardless of the origin point. Property 3 represents the triangle inequality, where, if a third point is introduced, then the distance between the two other points cannot be shortened whatsoever.

- **Euclidean distance**

This is the most used distance function in the real world, and it shows how distance is perceived. It is represented in Equation (1).

$$d_{Euclidean}(x, y) = \sqrt{\sum_{i} (x_i - y_i)^2} \qquad (3.18)$$

where $x = x_1, x_2, \ldots, x_m$ and $y = y_1, y_2, \ldots, y_m$ are true representations of the m attribute values of two records. After determining the records that are similar the next step is to determine the process of combining these records to give a classification decision based on the new record. This implemented using a combination function; the most used is the simple Un-weighted voting (Larose, 2005) in the following steps:

1. Decide the value of k, i.e., a value representing the total number of records that will have the most impact in classifying the records, before running the algorithm.
2. Perform the comparison between the new record and the k nearest neighbors. The comparison involves the k records with the minimum number of distances from the new record, based on the Euclidean distance function.
3. After choosing the k records, then the distance between them and the new record is no longer considered since one record will represent one vote.

Getting the k value entails considering some factors that could produce the best classification results. If k is a small value, then the classification results might be inaccurate due to the

presence of outliers or abnormal observations (Larose, 2005). This is because the model classifier outputs the nearest observation value; this may create what is called overfitting. Overfitting is a process whereby, the algorithm tends to memorize the training set rather than performing a generalization (Hand et al., 2001). A bigger k value will also smooth out any distinctive behavior from the training data set. A much bigger value of k will result in overlooking any locally relevant behavior of the training data set.

Choosing the value of k, therefore, involves voting for the best value that would suit the testing samples. Depending on the dataset being used, choosing an even number as the value of k would likely lead to a poor accuracy since it would classify the dataset into equal classes (Zhang & Zhou, 2005). An odd number as the value would, however, draw a comparison line that would make the basis of classification. A dataset can also be used to sort this problem of finding the best value of k, by following a cross-validation procedure. The procedure involves trying out different values of k with different randomly selected sets of training data. The value of k that best minimizes the classification error shall be considered the best value to work with (Potamias et al., 2010).

- **Multi-label Classification**

For Multi-label classification, the instances within the training set are associated with some label sets; the problem is to forecast label sets of unknown instances (Clare & King, 2001). One of the ways of solving the multi-class problem is to decompose it by generating a class of multiple classification features that are independent and binary. This way, the classification algorithm can then be applied to each of the independent classes (Myles & Hand, 2010).

Considering an instance x, and sets of labels associated to it, $Y_x \subseteq y$, and there are *k* nearest neighbors, let $\vec{y_x}$represents a category vector for *x*, and the *l-th* component $\vec{y_x}(l)(l \in y)$ takes the value of 1 if and only if $l \in Y_x$ otherwise 0 (Zhang & Zhou, 2005). Also, let $N(x)$ be an index set of the k nearest neighbours of *x* that is in the training set (Zhang & Zhou, 2005). A counting vector for its membership is defined in Equation (2).

$$\vec{C_x}(l) = \sum_{\alpha \in N(x)} \vec{y_{x_\alpha}}(l), l \in y \qquad (3.19)$$

where $\vec{C_x}(l)$ is the total summation of the neighbors of *x* that belongs to the *l-th* class.

3.2.4 Decision Trees

Decision tree classifiers were first used in the 1960s in the field of artificial intelligence (Li et al., 2001). Other machine learning classifiers are single staged making tree-based classifiers more efficient since decisions are made at multiple stages. Tree classifiers are represented as acyclic graphs containing a root node and successive child nodes that have directional branches connecting them. Decisions are made at each node based on the attribute value contained by the data (Hunt et al., 1996). True or false decisions are made by binary decision tree classifiers and complex decisions are made by non-binary decision trees. For such trees, cases traversing to the left side are true and cases to the right side are false. Cases traverse downwards the tree until it reaches a leaf node and at this point, the data returns a classification result (Lowe, 2015).

The size of the dataset determines the simplicity of the decision tree as it gets more complex with the introduction of more data features. There are several methods for the construction of the decision trees and it mostly depends on the application of the tree classifier. One approach for the construction of a tree is to apply the feature vectors and divide the data at each node in order to get the highest level of information gain at the level of the nodes (Pal & Mather, 2001).

As indicated in the related study section, there are several algorithms used to build decision trees. Some of the famous algorithms include Chi-square–Automatic–Interaction–Detection (CHID), Classification - regression tree (CART), Iterative Dichotomiser 3 (ID3) and C4.5 (Lowe, 2015). CHID is an essential decision tree learning algorithm to handle nominal attributes only. It is a supplementation of the automatic interaction detector and theta automatic interaction detector procedures. CART is the most popular algorithm in statistical method techniques. In the fields of statistics, CART aids decision trees in gaining credibility and acceptance in addition to making binary splits on inputs. ID3 is a simple algorithm that has been used widely and is among the first algorithms that was proposed. The algorithm uses information gain as a splitting criterion and the growth of tree comes to an end when all samples have the same class or information gain is not greater than zero. It, however, fails with numeric attributes or missing values. It combines C4.5, C4.5-no-pruning, and C4.5-rules. This method uses the gain ratio as a splitting criterion and is an optimal choice with numeric attributes or missing values (Alsagheer, 2017).

Test cases are presented at the root node, the start of the tree during its construction and the aim is to assign a set of features to the node which splits the data efficiently. The creation of more nodes down the tree means that the data is being split till each and each case at the nodes

belongs to the same class which is a terminal node (Alsagheer, 2017). There are various methods for splitting the data which carries different conditions and hence splitting the data differently. The most popular methods are entropy, information gain, Gini and twoing (Lowe, 2015).

3.2.4.1 Entropy

This is basic measure of information and is the most widely used splitting condition. It works by splitting the data equally thereby enabling the parent nodes to get the most information gain as possible. It is measured in bits and is given by the following equation (1.0) below:

$$entropy(D) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i) \quad (3.20)$$

Where:

n = number of outcomes

$P(x_i)$ = the probability of the outcome i and

The common values for b are 2, e, and 10. This is because the log of a number that is less than one produces a negative number and therefore adding a negative will make it positive. If all the cases fall in the same class, then $entropy(D)$will result in 0 since there is no information gain. The current node shall, therefore, be the terminal node, indicating that the classification decision has been attained (Han et al., 2012,).

3.2.4.2 Gini

This is the expected error rate when randomly selecting a classification decision from a class distribution (Duda et al., 2001). This technique splits data by extracting the cases into the largest homogenous group that can possibly be achieved and is given by the equation (2.0) below:

$$Gini(t) = 1 - \sum_{i=1}^{j} P(i|t)^2 \quad (3.21)$$

Where:

j = the number of classes

t = the subset of instances for the node and

$P(i|t)$ = the probability of selecting an element of class i from the subset of the nodes

The value of Gini impurity is highest when every class contains an equivalent probability of being selected. The value is, however, zero when all the elements that are in a set comprised of elements of a similar class that is equal to the chance of selecting a single element of that class, equal to one. Maximum Gini impurity value is dependent on the number of classes that exists and can be calculated by the following equation:

$$Gini_{max} = 1 - \frac{1}{n} \qquad (3.22)$$

3.2.4.3 Twoing.

This technique works by grouping cases that have similar characteristics at the top of the tree and then identifies individual classes at the bottom of the decision tree. During the twoing process, the classes are clustered into two super classes having an as-equal-as-possible number of cases. Afterward, the best split of the super classes is then found and used as the split at the current node. This results in a reduction of class possibilities among cases at each child node and a reduction in its impurity (Duda et al., 2001).

Let's denote all the classes at the node by M where $M = \{1, \dots, J\}$. For each node, divide Minto tow classes; $M_1$and $M_2$where $M - M_1 = M_2$. The sole aim here is treating it as a two-class problem and for any splits, at the node, we compute the change in its impurity, given by $\Delta_i(s, D, M_1)$. We shall then take the split $S * (M_1)$ which gives the maximum impurity change, then we find the superclass $M_1 * \max\Delta_i(s * (M_1), D, M_1)$ (Duda et al., 2001). The twoing function is therefore given the equation (4.0) below:

$$twoing(D) = \frac{p_L P_R}{4}\left[\sum_{i=1}^{m} | p(i|D_L) - p(i|D_R)|\right]^2 \qquad (3.23)$$

3.2.5 Random Forest

Random Forest is a supervised learning algorithm, grown using a collaboration of the bagging and ID3 principles and is a collection of many decision trees. For a decision tree, the creation of a node entails defining the associated condition about all the features on the dataset. For random forest, the creation of the node entails using only a fraction of some features which are

randomly selected from a pool of available features, thus the name 'random forest' (Ali et al., 2012). Attaining good classification accuracy is a key and some of the essential features are low bias and low correlation between the constituent trees. In order to attain a low bias, the trees are grown to maximum depth and for a low correlation, the process of randomization is done by selecting random subset features that will split a node and also apply it to training samples in order to build the tree (Verikas et al., 2011).

According to Cutler et al. (2011) for any *p*-dimensional random vector $X = (X_1, \dots, X_P)^T$ as a representation of the real-valued input variables, with a variable *Y*, that is random and is a representation of the real-valued response, then there is an assumption of the existence of an unknown joint distribution$P_{XY}(X, Y)$. The main aim here is finding a prediction function $f(X)$ that will be used in predicting the value of *Y*. This prediction function is determined by a loss function, $L(Y, f(X))$ and is defined solely to minimize the value of the expected loss, represented by the equation below:

$$E_{XY}(L(Y, f(X))) \qquad (3.24)$$

Where:

XY: is the expectation with respect to the joint distribution of *X* and *Y*.

Instinctively, $L(Y, f(X))$ measures the closeness there is between the prediction function $f(X)$ and Y. This is done by penalizing the values of $f(X)$ that distant from Y. For regression, the main choice of L has squared error loss, depicted by the equation below:

$$L\big(Y, f(X)\big) = (Y - f(X))^2 \qquad (3.25)$$

For classification, the choice of L is zero-one loss function, as is depicted by the equation below:

$$L\big(Y, f(X)\big) = I\big(Y \neq f(X)\big) = \begin{cases} 0, & if\ Y = f(X) \\ 1, & otherwise \end{cases} \qquad (3.26)$$

The underlying conditions are minimizing the value of $E_{XY}\Big(L\big(Y, f(X)\big)\Big)$ in regression problems, for squared error loss results in the conditional prospect as a regression function:

$$f(x) = E(Y|X = x) \qquad (3.27)$$

For classification problems, denoting the possible values of Y by$\mathcal{y}$, then by minimizing the value of $E_{XY}\left(L\left(Y, f(X)\right)\right)$ results in the following:

$$f(x) = \underset{y \in \mathcal{y}}{argmax}\, E(Y = y|X = x) \qquad (3.28)$$

This is the *Bayes rule* equation.

An ensemble works by constructing a function f that is in a term called the "base learners"$h_1(x), \ldots, h_j(x)$. A combination of the base learners results in an ensemble predictor, $f(x)$. In a regression problem, the base learners are averaged by the following equation:

$$f(x) = \frac{1}{J}\sum_{j=1}^{J} h_j(x) \qquad (3.29)$$

While in a classification problem, $f(x)$ is the predicted class:

$$f(x) = \underset{y \in \mathcal{y}}{argmax} \sum_{j=1}^{J} I(y = h_j(x)) \qquad (3.30)$$

The j^{th} base learner consists of a tree that is denoted by$h_j(X, \Theta_j)$, and Θ_j consists of a random variable that is independent for $j = 1, \ldots, J$.

3.2.6 Naïve Bayes

Naïve Bayes classifier draws its applications from the Bayes theorem (Bayesian) in statistics and is characterized by assumptions that are fully independent. Its assumption is based on the fact that a certain feature in a class is not related to the presence or absence of another feature (Russek et al., 1983).

- **A probabilistic model of Naïve Bayes.**

Zhang & Gao (2011) looked into a Naïve Bayes classifier for Text classification. Let $D = \langle d_i \rangle\, where\, i = 1, 2, \ldots, n$, represents a dataset to be classified, and d_i is an actual value in the dataset and there exists predefined classes, which is a set of $C = \{c_1, c_2, \ldots, c_k\}$. Here, the classification of the data includes assigning a label of the class $c_j,\ where\, j = 1,2, \ldots k$ from the set C to a dataset. The Bayes classifier is represented by the equation below:

$$P(c_j|D) = \frac{P(c_j)P(D|c_j)}{P(D)} \qquad (3.31)$$

Where:

$P(c_j)$: The preceding information of the appearing probability of the class c_j

$P(D)$: The information that is obtained from the observations that make up the knowledge from the values in the dataset to be classified

$P(D|c_j)$: This is the distribution probability of dataset D in the class spaces.

Bayes classifier works by integrating this information and thereafter computing the *posteriori* separately of the dataset D that falls directly into each of the classes c_j and proceeds by assigning the dataset to the class that has the highest probability, as shown by the equation below:

$$c*(D) = \underset{j}{\arg\max}\, P(c_j|D) \qquad (3.32)$$

An assumption is made here that the components $d_i$of the dataset D are independent with each other since the conditional probability, represented here by $P(D|c_j)$ cannot be directly computed practically (Zhang & Gao, 2011). Therefore, the following will hold:

$$P(D|c_j) = \prod_i P(d_i|c_j) \qquad (3.33)$$

Equation 6.13 is a representation of a naive Bayes model and this will have an impact on equation 6.11, as depicted by the equation below:

$$P(c_j|D) = \frac{P(c_j)\prod_i P(d_i|c_j)}{P(D)} \qquad (3.34)$$

The sample information $P(D)$ is identical to each of the class c_j, $where\ j = 1,2,\dots k$, then equation 6.12 will change to:

$$c*(D) = \left(\underset{j}{\arg max}\right) P(c_j)\prod_i P(d_i|c_j) \qquad (3.35)$$

3.2.7 Support Vector Machines

Support Vector Machines is a Supervised Machine learning algorithm with applications in mathematical and engineering problems. This technique performs classification by

constructing an N-dimensional hyperplane, which optimally separates the data into two categories or classes.

- **Linearly Separable Binary Classification**

Fletcher (2009) explained the linearly separable binary classification technique using Support Vector machines. Assume that there is a set S that comprises of points $X_i \in R^n$ where i = 1, 2,…, N and the points have D attributes. Each point of $X_i$belongs to any of the classes, and given a label $y_i \in \{-1, 1\}$. The training data shall, therefore, be of the form:

$$\{X_i, y_i\} \, where \, i = 1 \ldots L, y_i \in \{-1, 1\}, x \in R^D \qquad (3.36)$$

The assumption made here is that the data is separated linearly and therefore, we can draw a line on a graph of x_1 vs x_2 that will separate the two classes when D =2 and a hyperplane on graphs of $x_1, x_2, \ldots, x_D$ only when D > 2. This hyperplane is represented by the equation $\boldsymbol{w}.x + b = 0$, where:

$\boldsymbol{w}$ Is normal to the plane.

$\frac{\boldsymbol{b}}{||\boldsymbol{w}||}$Represents the perpendicular distance from the hyperplane to the origin.

Support Vectors are closest to the separating hyperplane and Support Vector Machines seeks to position this hyperplane in a manner that will make it as far as possible from the closest members of the two classes (James et al., 2009). The figure below shows a hyperplane that is through two classes that are separable.

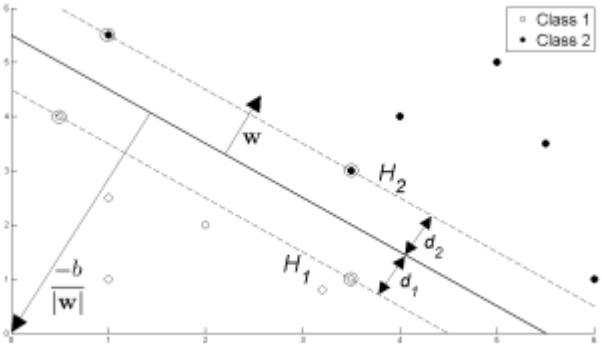

Figure 3:2: A Hyperplane between two separable (Smola & Scholkopf, 2004).

Based on the Figure 3.2, the process of implementing a Support Vector Machine entails selecting the variable w and b. In this way, the training data shall be described by:

$$\boldsymbol{x_i}.\boldsymbol{w} + b \geq +1 \quad for\ y_i = +1 \qquad (3.37)$$

$$\boldsymbol{x_i}.\boldsymbol{w} + b \leq -1 \quad for\ y_i = -1 \qquad (3.38)$$

The equation resulting after combining the two equations is:

$$\boldsymbol{y_i}(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 \geq 0\ \forall i \qquad (3.39)$$

In the figure above, $\boldsymbol{d_1}$ is the distance from the point $\boldsymbol{H_1}$ to the hyperplane and $\boldsymbol{d_2}$ is the distance from $\boldsymbol{H_2}$ to the hyperplane as well. The equidistance of the hyperplane from $\boldsymbol{H_1}$ to $\boldsymbol{H_2}$ is the Support Vector Machine's margin. This margin has to be maximized such that the hyperplane may be positioned to be quite far from the support vectors (Smola & Scholkopf, 2004). Applying vector geometry on this indicates that this margin is equivalent to $\frac{\mathbf{1}}{||\boldsymbol{w}||}$ and maximizing it is same as finding:

$$\min\|\boldsymbol{w}\| \ such\ that\ \boldsymbol{y_i}(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 \geq 0\ \forall i \qquad (3.40)$$

Minimizing $\|\boldsymbol{w}\|$ is same as minimizing $\frac{1}{2}\|\boldsymbol{w}\|^{\mathbf{2}}$ and for that, we have to find the value of the following equation:

$$min\frac{1}{2}\|\boldsymbol{w}\|^{\mathbf{2}}\ such\ that\ \boldsymbol{y_i}(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 \geq 0\ \forall i \qquad (3.41)$$

Finding the values of the constraints that are in the minimization equation above needs the allocation of the Lagrange multipliers $\boldsymbol{\alpha}$, where $\alpha_i \geq 0\ \forall i$:

$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \alpha[\boldsymbol{y_i}(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 \geq 0\ \forall i\ ]$$

$$\equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{L}\alpha_i\,[\boldsymbol{y_i}(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1]$$

$$\equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{L}\alpha_i\,\boldsymbol{y_i}(\boldsymbol{x_i}.\boldsymbol{w} + b) + \sum_{i=1}^{L}\alpha_i \qquad (3.42)$$

The values of w and b minimize while the value of α maximizes. Finding these values entails differentiating L_P with respect to w and b and setting the derivatives to zero. The resulting equations are as follows:

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i . x_j \ such\ that\ \alpha_i \geq 0\ \forall i, \sum_{i=1}^{L} \alpha_i y_i = 0$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i H_{ij} \alpha_j \ where\ H_{ij} \equiv y_i y_j x_i . x_j$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha^T \boldsymbol{H\alpha}\ such\ that\ \alpha_i \geq 0\ \forall i, \sum_{i=1}^{L} \alpha_i y_i = 0 \qquad (3.43)$$

This resulting outcome, L_D is the Dual form of the primaryL_P . This development, shifting from minimizing the value of L_P to maximizing value ofL_D , results in finding the value of:

$$\max_{\alpha} \left[\sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T H \alpha \right] such\ that\ \alpha_i \geq 0\ \forall i\ and \sum_{i=1}^{L} \alpha_i y_i = 0 \qquad (3.44)$$

A data point that is a Support Vector x_s will have the form:

$$y_s(\sum_{m \in S} \alpha_m y_m x_m . x_s + b) = 1 \qquad (3.45)$$

Where S here is the set of indices of the Support Vectors and can be determined by finding the indices I where $\alpha_i > 0$. Multiplying all through by y_s to obtain $y_s^2 = 1$ which is the same as the equation above, we get:

$$y_s^2(\sum_{m \in S} \alpha_m y_m x_m . x_s + b) = y_s \qquad (3.46)$$

$$b = y_s - \sum_{m \in S} \alpha_m y_m x_m . x_s \qquad (3.47)$$

An average over all the Support vectors in S:

$$b = \frac{1}{N_s}\sum_{s \in S} \left(y_s - \sum_{m \in S} \alpha_m y_m x_m . x_s \right) \qquad (3.48)$$

The variables w and b define the separating hyperplane's optimal orientation, which is our Support Vector Machine.

- **Support Vector Machine in Classification**

Fletcher, (2009), points out that the process of implanting Support Vector Machine on a classification problem, entails the following steps:

- First, we create **H**, where$H_{ij} \equiv \ y_i y_j x_i . x_j$.
- The value of α is determined such that:

$$\sum_{i=1}^{L} \alpha_i - \ \frac{1}{2} \alpha^T \boldsymbol{H} \boldsymbol{\alpha}$$

  Is maximized, which is subject to the underlying constraints:

$$\alpha_i \ \geq \ 0 \ \forall i \ and \ \sum_{i=1}^{L} \alpha_i y_i = 0$$

- Compute the value of :

$$w = \sum_{i=1}^{L} \alpha_i y_i x_i$$

- Determine the set of Support Vectors S by finding the indices such that $\alpha_i \ \geq 0$
- Calculate the value of b by:

$$b = \frac{1}{N_s} \sum_{s \in S} \left(y_s - \sum_{m \in S} \alpha_m y_m \, x_m . x_s \right)$$

- Finally, a new point x' is classified by evaluating $y' = sgn(w.x' + b)$

3.3 Performance Measures

3.3.1 Confusion Matrix

This is a matrix table for measuring the performance of supervised Machine Learning algorithms. The rows of the confusion matrix represent the instances of the actual class used and the columns instances of the predicted class (Luquea et al., 2019). These representations of the rows and columns can also be *vice versa*. A classification problem with *M* classes will, therefore, require a confusion matrix size of *M*x*M*.

	P' (Predicted)	N' (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

Figure 3:3: Confusion Matrix Table (Trajdos & Kurzynski, 2017).

The elements of a confusion matrix represent the: number of positives correctly identified or True Positive (TP), number of negatives identified correctly or True Negative (TN), number of negatives incorrectly identified as positive or False Positive (FP) and number of positives incorrectly identified as negatives or False Negative (FN).

Different measures of comparisons of a model can be obtained from the confusion matrix. One such measure is accuracy. The accuracy can be defined as the ratio of the total number of correct predictions made to the total number of predictions (Luquea et al., 2019). It is a popular measure used in checking the accuracy of the models being built; however, it cannot be used in measuring the performance of an imbalanced dataset. In fact, in such a dataset, the model will end up misidentifying the positive classes and yet depict high levels of accuracy. Such a scenario indicates that the accuracy metric does not consider pure randomness. On the other hand, the error rate is the difference in accuracy (Trajdos & Kurzynski, 2017). Equations below represent the accuracy and the error rate, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Error\ Rate = \frac{FP + FN}{TP + TN + FP + FN}$$

3.3.2 Precision and Recall

Precision is defined as the ratio of True Positives (correctly identified items) to the sum of True Positives and False Positives. It is the measure of the capacity of the correct information returned by the model, in percentages.

Recall, on the other hand, is the quotient of True Positives (Correctly identified items) and the sum of True Positives and False Positives. It, therefore, measures the capacity of all the relevant information that a model has extracted, in percentages. A striking feature of the two measures is that they are inversely proportional and the increase in the precision level of the model results in a decrease of its recall measure and vice versa (Luquea et al., 2019). F-Measure combines the two and all the equations are represented in Equations below.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - Measure = \frac{2(Precision)(Recall)}{Precision + Recall}$$

3.3.3 Receiver Operating Characteristic (ROC) Curves

Some classification problems cannot be measured with classification scores, mostly when dealing with datasets with heavy class imbalance. Receiver Operating Characteristic (ROC) curves are useful in such cases, offering the best alternatives (Nazrul, 2018). ROC is a plot of True Positive Rate (TPR), on the x-axis against False Positive Rate (FPR) also known as 1-specificity on the y-axis. These two measures are defined in Equations below.

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{TN + FP}$$

After plotting the curve, the model performance is determined by looking at the area under the ROC curve (AUC). An AUC value of 1 is considered as the best possible that can be obtained in a model while the worst is 0.5. An AUC value that is less than 0.5, would call for reversing the recommendations of the model with the aim of getting a value that is above 0.5. Using ROC curves to check the performance of a model allows for comparison of curves of different models directly or for different thresholds whereas the AUC can be used as a summary of the model skill (Brownlee, 2018).

3.4 Conclusion

This chapter presented the popular Machine Learning algorithm methods that were used in this study in performing the classification of ontologies. The Performance evaluation techniques

methods have also been presented. Each algorithm has been described by discussing its mathematical structure and equations used in building them. The next chapter presents the experimental results that were obtained from the implementation of these algorithms.

CHAPTER 4. : SYSTEMS ARCHITECTURE

4.1 Introduction

This chapter discusses the architectures for the implementation of the AI classification models that will be built in this study. The chapter first introduces the software environment under which the experiments shall be conducted. System architecture for the implementation of AI classification algorithms is then discussed where the steps followed in its implementation are discussed. Flow charts describing the implementation of the classification algorithms are also discussed in the later section of this chapter.

4.2 Architecture for the Implementation of AI Classification Algorithms.

The process of implementing a classification algorithm in a classification problem entails a set of steps. An architecture of a model stipulates the flow of the working of the model from the initial stage to the final stages. Understanding the architecture of a model helps in identifying any arising issues during the working of the model (Mone, 2019). The following steps are followed in the implementation of the AI algorithms for this experiment. Subsections for each of the steps indicated below have been broadly discussed in the section to follow:

1. Data collection: The dataset used in this experiment consists of the complexity metrics, generated from the OntoMetrics platform.
2. Preparation of the input data: Raw data collected cannot be used in implementing the algorithms for classification and should be converted to a format that the algorithms can be able to take it in training the model.
3. Analyzing the input data: Before the implementation of the AI algorithm, the data is explored to check for any patterns that will aid in the implementation. Checking the correlation between the features, for example, allows us to do away with the features that are less correlated and focusing more on features that are highly correlated.
4. Training the Algorithm: The analyzed data is fed to the algorithm after splitting it into training and testing set with a specified ratio. The algorithm will then extract knowledge or information from the data.
5. Testing the algorithm: After extracting the information from the training phase, the algorithm shall then be tested by checking its' accuracy to check whether the results obtained are the desired results.

6. Using the Algorithm: If the algorithm has a higher accuracy and the output results are as desired, then the algorithm is used in making various predictions or making decisions as it was intended to perform. However, if the results obtained are unexpected then the training phase is repeated, but with different ratios.

A flow chart shows a step by step detailed workflow of the implementation of the AI algorithms in classification is shown in the diagram below:

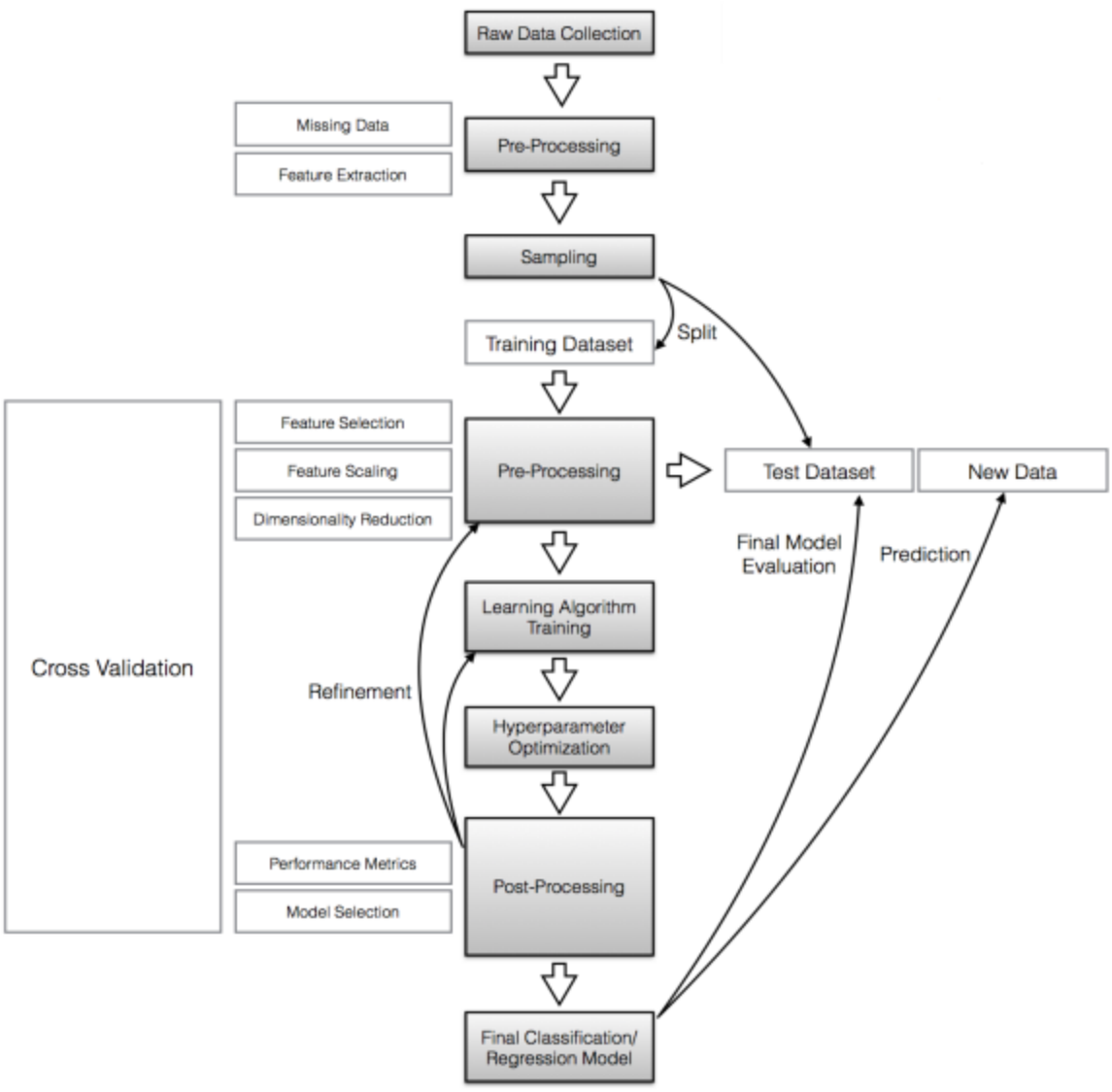

Figure 4:1: Implementation of AI Algorithms workflow

4.2.1. Raw Data Collection / Data Acquisition

This is the first step of the architecture where enough data is collected from the relevant sources and supplied into the classification model. Data can be in different sources such as ERP databases, mainframe devices or Internet of Things (IoT) devices (Sapp, 2017).

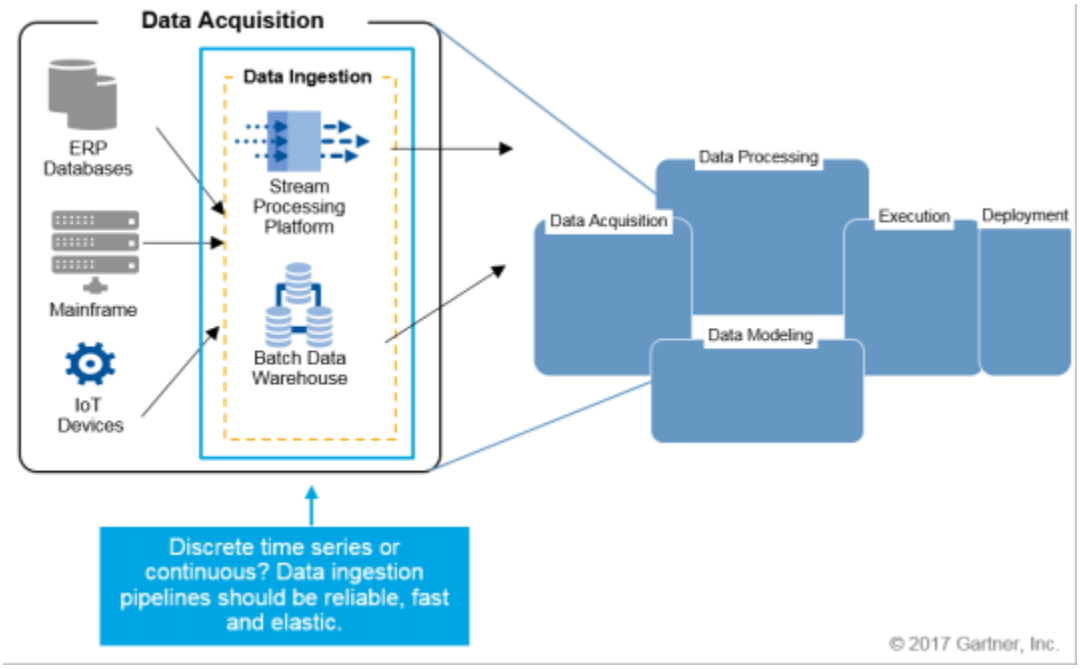

Figure 4:2: Data Acquisition (Garner, 2017)

The data used in the experiment comprises the metrics generated from the OntoMetrics platform and is used as a training data set for the model. The dataset has a total of ten features belonging to the respective ontology complexity metrics. The file containing the dataset, in *CSV* format, is uploaded to the Python's Jupyter notebook platform where the AI algorithms shall be implemented in a classifier model. The classification results are displayed as the output. This (Jupyter Notebook) Integrated Development Environment (IDE) platform is a web-based platform that allows for line by line recall and hence is interactive and it documents all the aspects of the workflow being done (Pandey, 2018).

The IDE uses inbuilt libraries for performing data analysis, implementation of the algorithms and for evaluation of the algorithm performances. The screenshot below shows the libraries that have been used in the experiments.

```
In [*]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.metrics import classification_report, confusion_matrix
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.svm import SVC
        from sklearn.naive_bayes import GaussianNB
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.preprocessing import StandardScaler
        from sklearn import model_selection
        from sklearn.model_selection import train_test_split
        import seaborn as sns
        from sklearn import metrics
        from sklearn.metrics import precision_recall_curve
        from sklearn.metrics import f1_score
        from sklearn.metrics import auc
        from sklearn.metrics import average_precision_score
        from sklearn.metrics import roc_curve
        from sklearn.metrics import roc_auc_score
        from sklearn.cross_validation import StratifiedKFold
        from scipy import interp
        %matplotlib inline
```

Figure 4:3: Libraries for data analysis, implementation of the algorithms and for evaluation of the algorithm performances

The libraries represented by KNeighborsClassifier, SVC, GaussinNB, LogisticRegression, DecisionTreeClassifier, and RandomForestClassifier are inbuilt classification algorithms. Classification_report, confusion_matrix, AUC, roc_curve, and model selection are libraries used for algorithm evaluation performances.

4.2.2. Data Pre-processing

Any data that is generated is not always perfect since some may contain irrelevant variables, a small number of samples, missing values and outliers. Data pre-processing seeks to increase the ability of the classification and predictive models by working on the irrelevant features that are contained in data (Castrounis, 2016). Various approaches exist that aim at sorting out the imperfections: such as imputations of missing values, removing of superimposed noise through smoothing and exclusion of outliers (Hang et al., 2016). These techniques improve the quality of data in its own unique way. The following figure shows the data pre-processing architecture:

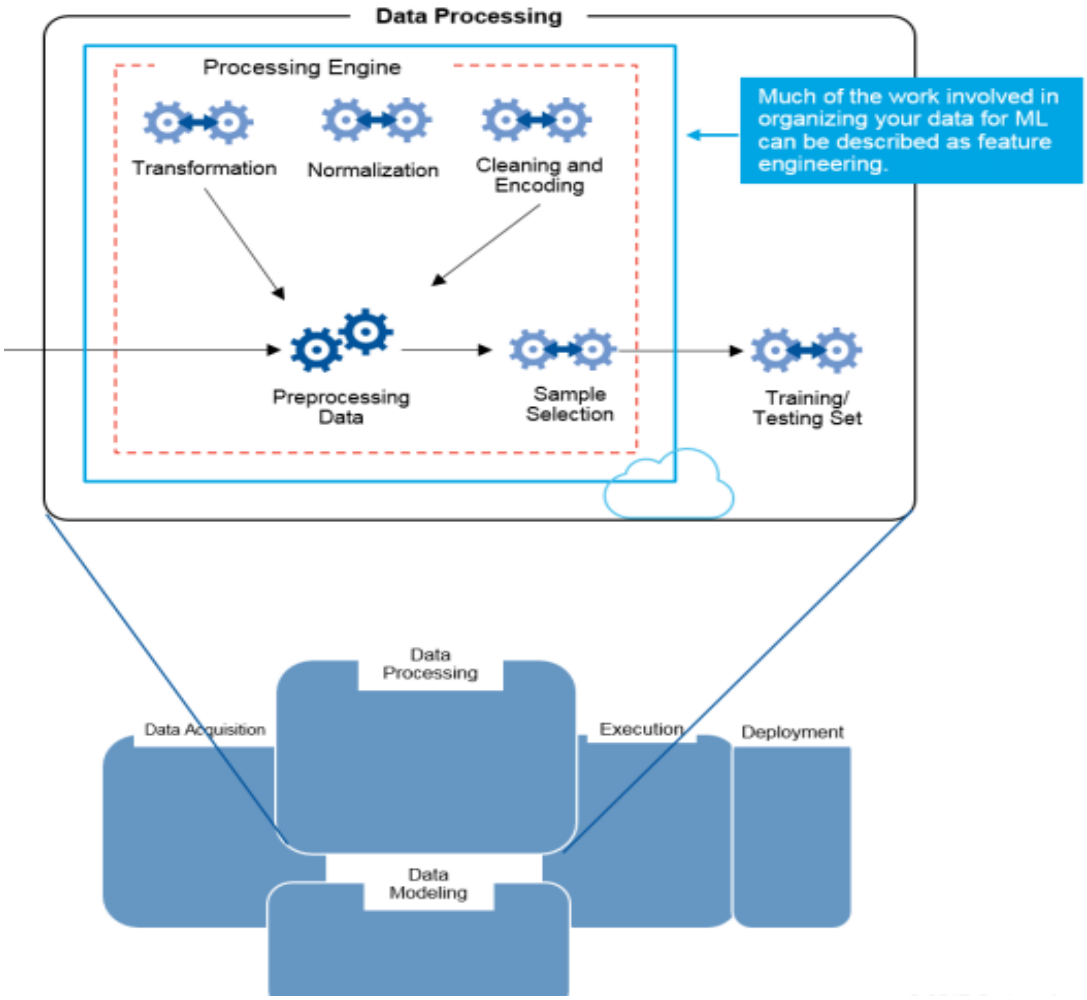

Figure 4:4: Data Pre-processing (Gartner, 2017)

4.2.2.1. Missing Data

Missing values is one of the most common issues faced when doing data analysis. There are several causes of missing values in a dataset and since it affects the performance of a classification algorithm, it is best to handle these missing values before the implementation of the algorithm (Swalin, 2018). The dataset used in the experiment was checked to ascertain whether it had missing values and since there were indeed missing values, appropriate techniques for handling missing values were applied. Some ontologies might not contain all the complexity metrics being generated for this experiment and since the data scarcity is minimal, then the missing values are replaced by some statistics.

4.2.2.2. Feature Extraction

Every feature set distribution type cannot have a perfect method to be used in feature distribution. Most of the supervised algorithms are based on *a priori* knowledge of the cluster distributions (Molder, 2004). Feature Extraction is a preprocessing technique that aims at reducing the feature space dimensions. Through Feature Extraction, the class distributions in a multidimensional space of feature sets are separated distinctively. It also aids in the creation of a connection between the feature data set and the classification systems in the case of a large data set (Lerner et al., 1999). In the experiment, the correlation between the attributes has been checked to determine the attributes that are highly correlated then focus on only them.

4.2.3. Sampling

For any learning algorithm, performance mainly relies on the training set that has been used in building the classifier. The detection rate of classifiers is increased by providing them with a good training sample to work with. These samples also immensely decrease the false positive rate of the classifiers. According to Takizawa & Nakajima (2000) samples that are in the indeterminate area are of great use in the building of a learning algorithm to active learning. Lyhyaoui et al. (1999) also came up with a sample selection approach that uses clustering techniques to choose or select a boundary sample.

4.2.4. Training set

The next step in the architecture is the process of splitting the dataset into training and testing sets according to a specified ratio. In order to train a classification model, the Artificial Intelligence algorithm must be provided with enough data to learn from. In this case, the training data contains all the information being sought after by the classification algorithm. The training set is obtained by dividing the dataset with a given ratio, and the remaining being the test set that will be used to test the performance of the algorithm. Since the learning algorithm must find the underlying patterns in the dataset, then the training set must be larger than the test set (Al-Masri, 2018). In the experiments, different ratios have been used to fit the implementation of the algorithm.

4.2.5. Pre-processing

Any datasets used in the implementation of Classification algorithms can have multiple features or dimensions in its structure. The best method of dealing with this problem is by pre-

processing the data; it entails cleaning up the data either by normalization or scaling techniques such that the data can have a relatively similar magnitude (Zhang et al., 2013).

4.2.5.1. Feature Selection

The process of building a classification model requires using part of the dataset to act as the training set and part of it as the testing set. A given dataset contains some elements of irrelevant or redundant features making up the dataset (Kira & Rendell, 1992). Building the classification model with these features in existence in the dataset will lead to the delay in the learning algorithm's running time as indicated by Hall & Smith (1999). Removing these redundant features, therefore, reduces the running time and produces a more general concept (Dash & Liu, 1997). The underlying concept of the classification problem is attained by this method since it picks up a subset of features that are only of relevance to the desired result (Kohavi & Sommerfield, 1995).

4.2.5.2. Feature Scaling

Data Scaling and Normalization are pre-processing techniques that aim at consolidating data into ranges that are quite appropriate for the implementation of AI algorithms. Classification models that are created from scaled data always portray a higher accuracy performance as compared to the ones obtained from unscaled data (Hang et al., 2016).

4.2.6. Learning Algorithm

The learning algorithm / Data modelling phase of the architecture is a phase where the selected Artificial Intelligence algorithms are implemented. In the experiments, Classification algorithms are built at this stage and the dataset should have been prepared enough for the implementation (Funke, 2019). The learning algorithms here are inbuilt and are therefore always available to perform the classification exercise. The main objective, in this case, is to perform classification of the data being fed to the classification model which has been trained to solve the specific problem.

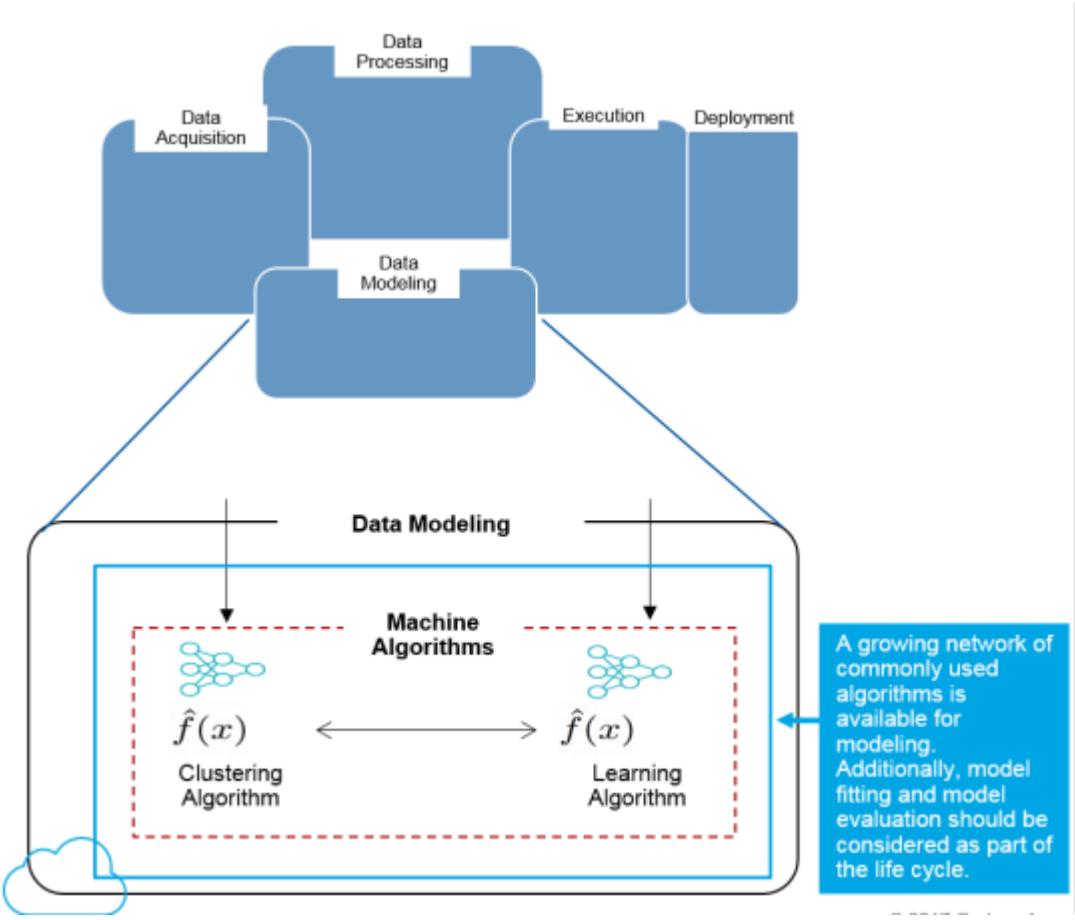

Figure 4:5: Learning Algorithm / Data Modelling phase (Gartner, 2017)

The algorithm being the building block of the model shall execute the classification process repeatedly thereby performing testing and tuning of the experimental results until the desired results are obtained (Sapp, 2017).

4.2.7. Performance Evaluation / Post-Processing

4.2.7.1. Performance Metrics

During the classification of data, evaluation metrics play a role in obtaining optimal or desired results. The metrics measure the performance of the classifier according to a specified measurement (Hossin & Sulaiman, 2015). The existing evaluation metrics are broadly categorized into threshold, probability, and ranking metric types according to its aims. Based on these types of metrics, it can be used to measure and summarize the quality of trained classifier by testing it with unseen data (Caruana & Niculescu-Mizil, 2004). In this way, accuracy and error rates are commonly applied.

4.2.7.2. Model Selection

Evaluation metrics used for model selection is used to determine the best classifier among the different types of the trained classifiers that focuses on the best performance in the future, which will create an optimal model during testing with the unseen data (Caruana & Niculescu-Mizil, 2004).

4.2.7.3. Cross-Validation

The process of model selection by validation is by splitting the training set to be used in evaluating the relative generalization performance of the models. More often, the training data is always less and therefore degrading the performance of the classifier (Guyon et al., 2006). Cross-validation is one of the ways used in splitting the data. The technique entails partitioning the training data into a certain value of disjoint subsets each of roughly equal sizes. In most cases, the variance of the results might be big, and this can be reduced by performing multiple K-fold cross-validations and thereafter averaging the results. When dealing with datasets that have heavy imbalances, classification scores cannot be the most appropriate technique to be used. The Receiver Operating Curve plots the True Positive Rate (TPR), on the x-axis against False Positive Rate (FPR) on the y-axis. The model performance here is determined by checking the area under the curve.

4.3 Artificial Intelligence Classification Algorithms Flow Charts

4.3.1. K-Nearest Neighbors

The K-Nearest neighbor algorithm can be used for either classification, estimation or prediction and is an example of instance-based learning. The algorithm first loads all the data points that are in the dataset in its memory or makes a plotting of the data in an n-dimensional space (Harrison, 2018). Each point on the plotting area represents a label in the dataset. A test sample, which represents a set of training data shall also be plotted within the same n-dimensional space. It will then make a search for its k nearest neighbors based on a specific distance measure out of the training samples (Denoeux, 1995). Figure 4.6 shows the flow chart for the implementation of the K-Nearest Neighbors algorithm in this study.

The implementation of the K-Nearest neighbor algorithm relies solely on the distance between the data points. The flow chart below shows a step by step implementation of the algorithm:

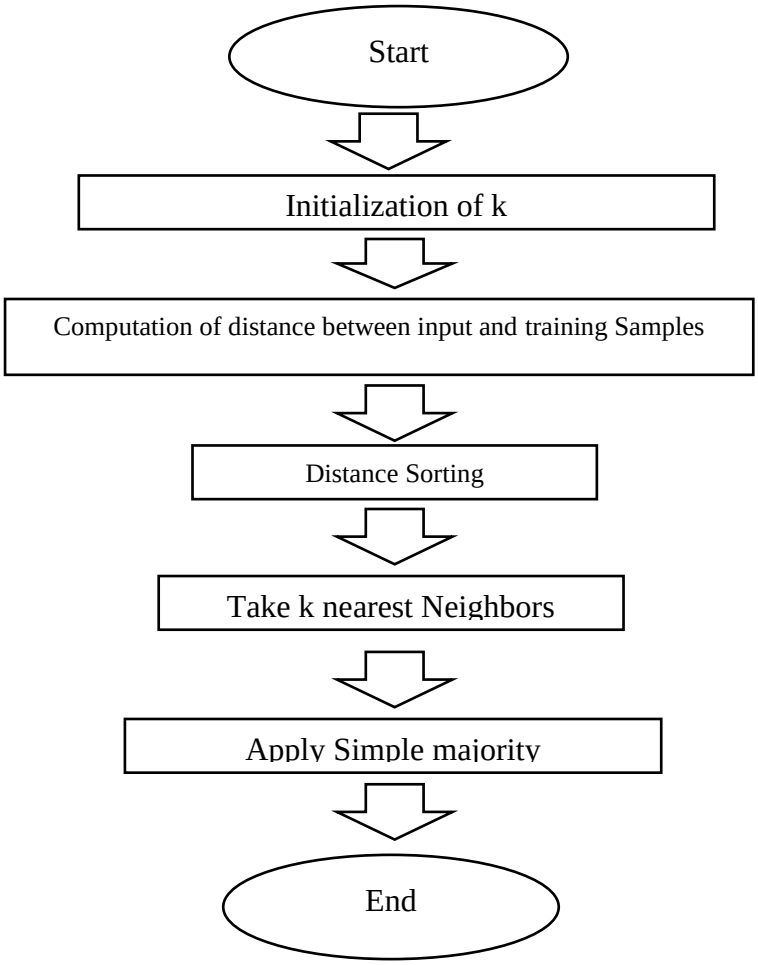

Figure 4:6: Implementation of K-Nearest Neighbors Algorithm for Classification

The importation of the necessary libraries is preceded by splitting the dataset into the training and testing set and here, a ratio of 0.25 is used. The value of k is first initialized and different values of k were tried. The Euclidean distance between the new input and each training sample given in the algorithm is then calculated (Larose, 2005). Sorting out the distance of each training sample from the input data is done and the k nearest neighbors are chosen to the input data. After choosing the neighbors new input data is given various labels according to the classification model, which is majority among the neighbors. Their evaluation accuracy scores were then taken to ascertain the value of k that produces the highest accuracy score (Schaffer, 1994).

4.3.2. Logistic Regression

Logistic regression classifies data by finding the regression coefficients of the data being classified and then uses it to find whether there exists a relationship between it. The flowchart below shows its implementation:

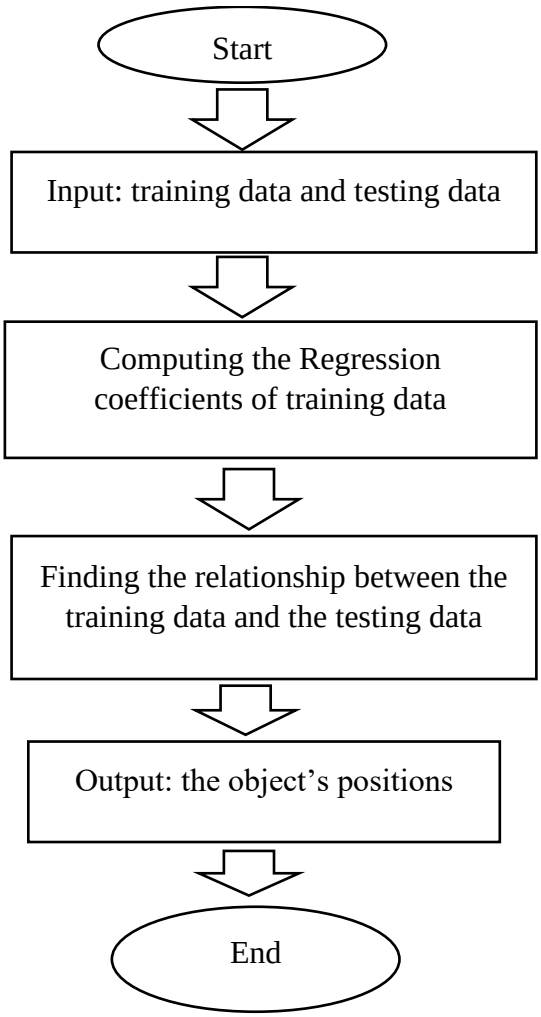

Figure 4:7: Logistic Regression Algorithm Flow Chart

4.3.3. Naïve Bayes

The process of performing classification can be broadly placed in two stages: the training phase and the testing phase. In the training phase, the sample data is provided to the classifier where the classifier shall then learn a class prediction model basing on the labelled data. For the testing phase, the unlabelled features are provided to the classifier where it will apply the classification model in determining the classes of the unseen data (Jadon & Sharma, 2017).

With Naive Bayes, the training data is used in estimating the features and probabilities which will give the ability to create new instances from the Bayes rule extracted from the estimates to find the conditional probability. Conditional independent rule is applied to reduce the parameters modelling the existing conditions (Zhang & Gao, 2011). Building the model requires the calculation of its probabilities which shall be used in calculating the Area Under the Curve (AUC) for its evaluation. Upon evaluation of the model, it is then used in the classification of data. Figure 4.8 below shows the flowchart of the Naïve Bayes classifier.

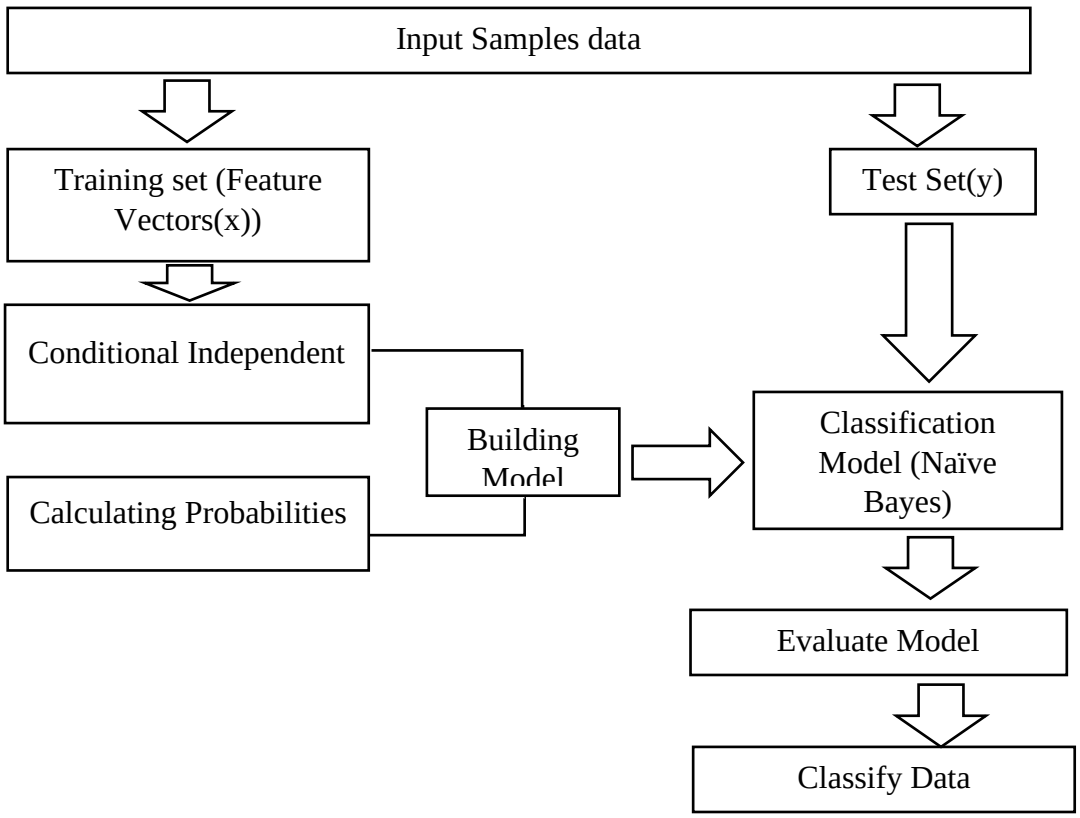

Figure 4:8: Naive Bayes Algorithm Classification Flow Chart

Naive Bayes is a probabilistic model based on the Bayes Theorem. The Model examines the likelihood of the features appearing in the predicted classes. With Naive Bayes, the training data is used in estimating the features and probabilities which will give the ability to create new instances from the Bayes rule extracted from the estimates to find the conditional probability (Zhang & Gao, 2011). Conditional independent is applied to reduce the parameters modelling the existing conditions.

4.3.4. Decision Trees

Decision tree classifiers were first used in the 1960s in the field of artificial intelligence (Li et al., 2001). Other machine learning classifiers are single staged, making tree-based classifiers more efficient since decisions are made at multiple stages. Tree classifiers are represented as acyclic graphs containing a root node and successive child nodes that have directional branches connecting them. Decisions are made at each node based on the attribute value contained by the data (Hunt et al., 1996). True or false decisions are made by binary decision tree classifiers and complex decisions are made by non-binary decision trees. For such trees, cases traversing to the left side are true and cases to the right side are false. Cases traverse downwards the tree until it reaches a leaf node and at this point, the data returns a classification result (Lowe, 2015). Figure 4.9 below is a flow chart for the implementation of a decision tree algorithm.

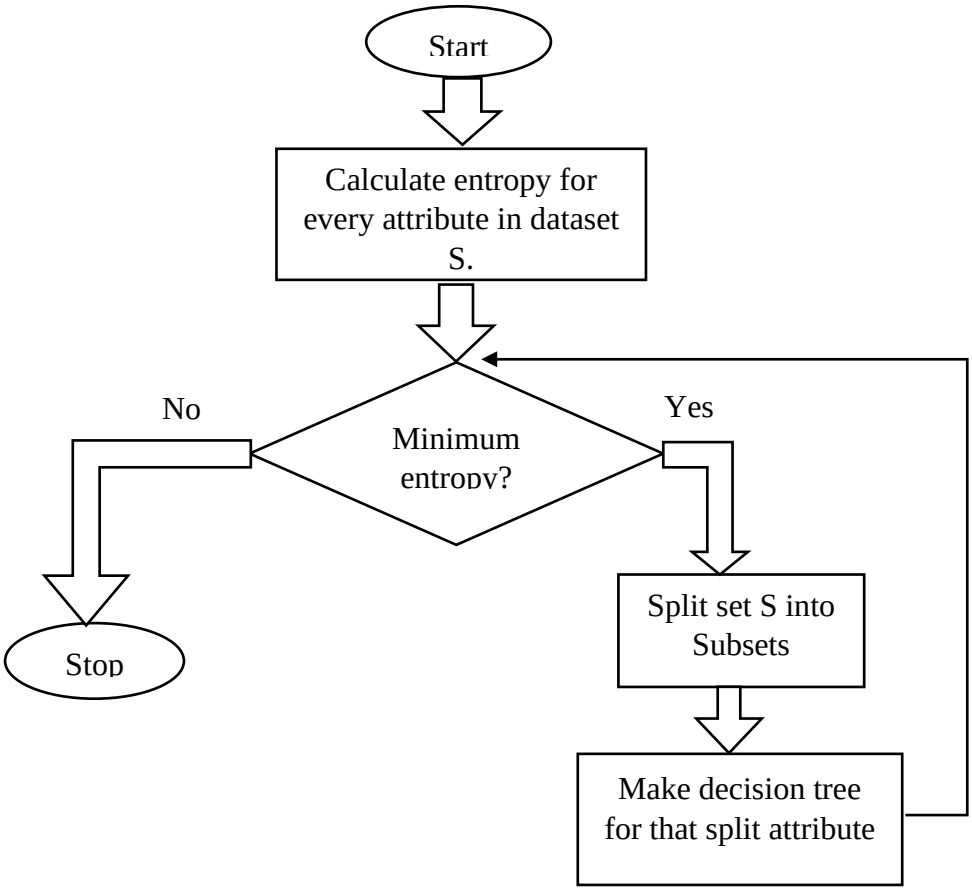

Figure 4:9: A Flow Chart Representation for Decision Tree Algorithm

The implementation of the Decision trees algorithm relies on the learning strategy of the ID3 algorithm. This algorithm performs the process of splitting the data based on the best attribute and placing it on the right leaf nodes. The algorithm learns from the dataset when to stop

performing the splitting process. Information theory is a concept used in determining the best attribute when splitting the dataset and it measures the difference in information right before and after splitting, recording this as information gain (Rastogi & Shim, 1998).

The ID3 algorithm plays a role in deciding the best feature to perform in the splitting of the dataset based on the calculation of its information gain. The set that has a higher information gain is regarded as the best feature to split. The dataset shall then be split into subsets that are based on the best feature and ultimately the process of checking whether all the data in the subset belongs to the same class (Srikant, 1997). If it does, then the process is stopped and the classifer is evaluated for its accuracy. If different classes exist then the system goes back to making decisions of feature and dataset split, producing more branches to the tree.

4.3.5. Random Forest

The Radom forest algorithm uses the bootstrap resampling method to extract multiple samples from the original samples and construct sub-datasets. It then uses the sub dataset to form the base decision tree and train it. In the decision tree training, random forest aids in randomly selecting attributes. It first selects the attributes of the nodes in the random selection of a K attribute subset, and then from the subset to select an optimal attribute for node splitting, which can make each decision tree different from each other. The classification results are then obtained by the voting method, such that the classification performance is enriched (Man et al., 2018). Figure 4.10 below is a flow chart for the Random Forest algorithm.

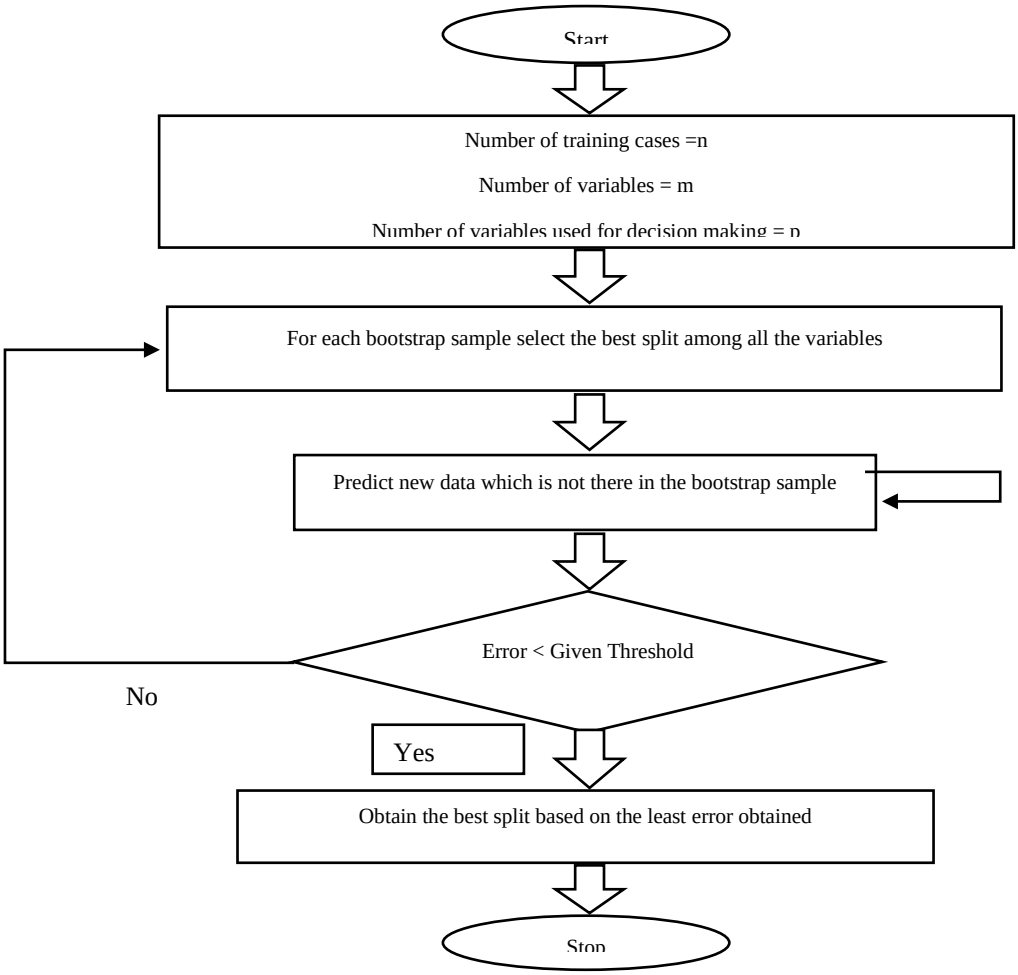

Figure 4:10: A Flow Chart Representation for Random Forest Algorithm

Random forest differs from Decision trees by the number of trees in its model having multiple trees and each of the trees depending on the collection of random variables of certain training points (Segal & Xiao 2011). The step of checking whether all the subsets in the dataset belong to the same class in Decision Trees is preceded by building the next split or branch and in Random forest, a variable subset is chosen at this step. For each variable chosen, sorting is done by the computation of the Gini index at each and every split point. The split with the highest Gini index value is then chosen at this step and the process is repeated (Delgado-Gomez et al., 2018). The process of choosing the best split and its repetition leads to the creation of multiple trees in the model.

4.3.6. Support Vector Machines

The process of building a Support Vector Machine classifier to solve classification problems is a stepwise procedure. After identifying the underlying problem, the data collected shall be

used as input samples. Exploratory analysis of data is then performed to understand all the components of the dataset. Data pre-processing is then done to ensure that only important features are being used as in building the classification model. The data is split into testing and training sets based on a given ratio and then the model is built using the inbuilt libraries. Before it is deployed, the model is evaluated to check whether it is viable in performing the classification of data (Jottrand, 2005). The flow chart in Figure 4.11 shows the stepwise implementation of SVM.

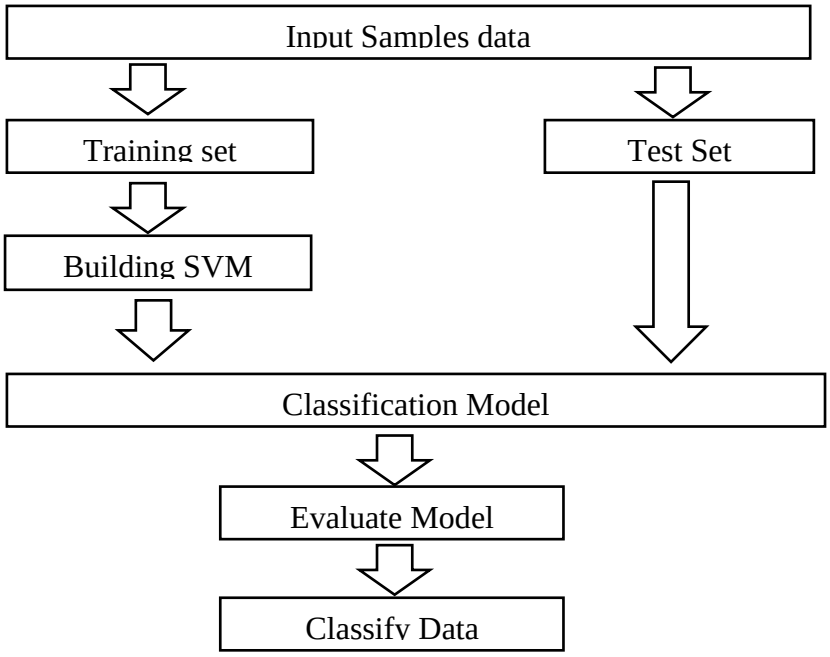

Figure 4:11: Support Vector Machines Algorithm Flow Chart

The inbuilt support vector machines classifier is imported as a library in pythons' Sci-Kit-Learn environment as *'from sklearn.svm import SVC'*. This is imported alongside other libraries such as evaluation and scientific computing libraries. After data preparation, the dataset is split into training and testing sets using a ratio of 0.20. Support Vector Machines classification model is then built by creating a method for the imported library. Since it's a linear SVM model, then the method creates a parameter, indicating that the kernel is of linear type, and setting the probability as true, since it shall be used in testing the model performances (Fletcher 2008).

The built model is then preceded by fitting the model into the training set since the algorithm will learn from the set. After building the classifier, it can now be used to perform classification. A method is therefore created that will use *predict()* function in the data. The model is finally

evaluated to check its performance accuracy and this will be used in deciding whether to use the algorithm as a classifier.

4.4 Conclusion

This chapter presented the system architecture for the implementation of AI algorithms to be used in the classification of ontologies based on ontology metrics. The chapter stipulates the steps followed in the implementation of each AI algorithm from the initial stage of data acquisition, data preparation, building the model, model performance evaluation and finally the use of the final model. In discussing these steps, flowcharts have been used to show the flow of processes and tools used in each step. The next chapter presents Experimental analysis where the results of the implementation of AI algorithms in the classification of ontologies are discussed in detail.

CHAPTER 5. EXPERIMENTS AND DISCUSSION

5.1 Introduction

In this chapter, the process of data acquisition and the method of metrics generation making up the dataset have been outlined. The software environment under which the experiments were conducted is also stated. The experiments conducted, which include the implementation of machine learning algorithms (K-Nearest Neighbors, Support Vector Machines, Naïve Bayes, Logistic Regression, Linear Regression, Decision Trees and Random Forest) are discussed in detail. The performance evaluation techniques for each algorithm implemented are discussed and the results obtained in terms of the accuracy are discussed. Besides the accuracies, the ROC curves used in checking the performance of classification algorithms are also discussed.

5.2 Software Environment

The experiments have been conducted using HP Pavilion dv6 computer, Intel(R) Core (TM) i5-2410M, CPU @ 2.30GHz 2.30 GHz, 6.00 GB RAM, 64-bit Windows 10 operating system, x64-based processor and hard drive size of 500GB. Python software, version 3.7.0 and Jupyter Notebooks version 5.0.0 have been used in running the experiments.

5.3.Dataset

The dataset used in the experiments comprises of 200 ontologies and 17 metrics for each ontology. Ontologies were obtained from the National Center for Biomedical Ontology (NCBO) which is a Bio Portal web platform that provides access to more than 600 ontologies of the biomedical domain.

After randomly downloading the ontologies, their complexity metrics were computed using the OntoMetrics platform. This platform provides a collection of ontology metrics that can be broadly classified into the schema, knowledgebase, and graph metrics. The generated ontology metrics comprises the dataset used in this study. The metrics include: Number of Classes *(noc),* Attribute Richness *(ar),* Inheritance Richness *(ir),* Relationship Richness *(rr),* Equivalence Ratio *(er),* Axiom Class Ratio *(acr),* Inverse Relations Ratio *(irr),* Class Relations Ratio *(crr),* Average Population *(ap),* Class Richness *(cr),* Absolute Root Cardinality *(arc),* Absolute Leaf Cardinality *(alc),* Average Depth *(ad),* Maximal Depth *(md),* Average Breadth *(ab),* Maximal Breadth (mb) and Average Number of Paths *(anp)* as shown in the Appendix (you must label the Appendix such A, B etc.

Before commencing the process of building the classification models, a classification rule is laid down that is in line with this multiclass classification. As mentioned earlier, the metrics generated from the OntoMetrics platform lies in the broad categories of the schema, knowledgebase and graph metrics. However, in evaluating an ontology, the metrics considered are measured against the quality properties of an ontology. There are four quality dimensions use in evaluating and ontology which correlate with the ontology metrics obtained with OntoMetrics (Fonou-Dombeu & Viriri, 2019). The correlation of ontology metrics and the four ontology quality dimensions have been used in defining a classification rule making up the decision class in the classification. The quality dimensions together with the metrics with higher correlation are:

- Accuracy – it indicates the extent to which an ontology represents a real-world domain. This criterion correlates with the following metrics Attribute Richness *(ar),* Inheritance Richness *(ir),* Relationship Richness *(rr),* Equivalence Ratio *(er),* Average Depth *(ad),* Maximal Depth (md), Average Breadth *(ab),* Maximal Breadth *(mb)* and Average Number of Paths *(anp).*
- Understandability – it checks for the comprehensiveness of the concepts, relations and properties in an ontology. Understandability correlates with the Absolute Leaf Cardinality *(alc)* metric.
- Cohesion – This criterion checks for any similarities between the ontologies. This quality dimension is achieved by the metrics Absolute Root Cardinality (arc) and Absolute Leaf Cardinality *(alc).*
- Conciseness – This measures the extent to which an ontology is of importance within the domain it specifies. Conciseness is attained by the Average Population *(ap)* and Class Richness *(cr)* metrics.

The variation in the attainability of these quality dimensions make the metrics in the dataset more suitable to form a classification rule or a class label. In this case, the classification rule has been made by conducting a search of the ontology metrics that correlate with the quality dimensions. Four class labels each representing the quality criteria have been specified. The classes have been labelled 0 for accuracy, 1 for understandability, 2 for cohesion, and 3 for conciseness.

5.4 Results and Discussions

5.4.1. K-Nearest Neighbors

K-Nearest Neighbors (kNN) is an instance-based learning algorithm. Instead of constructing a general internal model the kNN algorithm stores instances of the training data. In the model, classification is computed from a majority vote of the nearest neighbors of each data point.

The *KNeighborsClassifier* Python function is used to build the model and the value of *k* is dependent on the size of the dataset. A large value of k suppresses the noise effects in the classifier but makes classification less distinct. A ratio of 0.3 has been used in splitting the dataset into training and testing sets in the experiment.

The parameters used in building the model include the *Leaf_size*, *metric*, *metric params*, *n-jobs*, *algorithm*, *n_neighbors*, *p* and *weight*. The *algorithm* parameter computes the nearest neighbor and, in this experiment, it's been set to 'auto' which automatically decides on the most appropriate algorithm to be used based on the values passed to the 'fit' method. The *Leaf_size* parameter affects the speed of the model construction and the memory needed to store the tree. The metric specifies the distance function used and, in this experiment, Minkowski metric has been used. The *p* is the power parameter for the Minkowski metric and has been set to 2 in this experiment. The *n_jobs* parameter is the number of parallel jobs to run for neighbors' search and the *non-option* set in this experiment shows that it does not affect the fit method used in constructing the classification model.

Different values of k were used to check the most suitable value that produced the highest accuracy score. The respective precision recall and f-measure score values of the model are given in Table 5.1 which show that, as the value of k increases, the accuracy scores decreases. This is attributed to the increase in outliers as the value of the nearest neighbors increases.

Table 5:1: The Performance Scores for the Different k Values

	K-Nearest Neighbors				
No. of K	**Class Labels**	**Accuracy**	**Precision**	**Recall**	**F- Measure**
5	0	66.67	0.59	0.71	0.65
	1	66.67	0.50	0.59	0.54
	2	66.67	0.75	0.43	0.55
	3	66.67	0.93	0.93	0.93
10	0	63.33	0.56	0.71	0.63
	1	63.33	0.50	0.41	0.45
	2	63.33	0.64	0.50	0.56
	3	63.33	0.93	0.93	0.87
15	0	60.00	0.44	0.50	0.47
	1	60.00	0.50	0.53	0.51
	2	60.00	0.55	0.43	0.48
	3	60.00	0.93	0.93	0.93
20	0	61.67	0.53	0.57	0.55
	1	61.67	0.56	0.59	0.57
	2	61.67	0.45	0.36	0.40
	3	61.67	0.88	0.93	0.90
30	0	56.67	0.45	0.64	0.53
	1	56.67	0.47	0.41	0.44
	2	56.67	0.55	0.43	0.48
	3	56.67	0.86	0.80	0.83

For k=5 in Table 5.1, the model displayed an accuracy of 66.67% and the average precision, recall, and F- Measure scores of 69%, 67%, and 67%, respectively. The confusion matrix for the kNN classification model for k = 5 is shown in Figure 5.1.

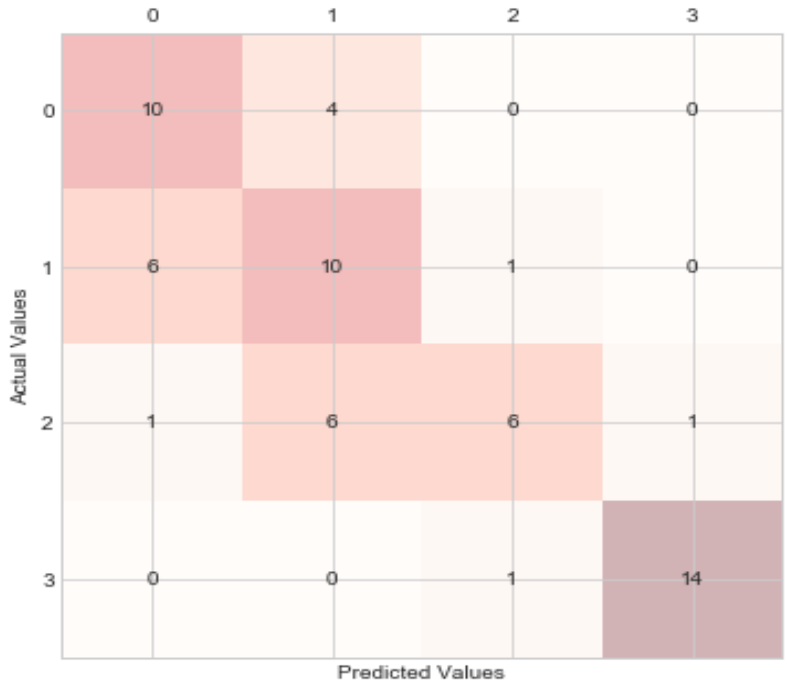

Figure 5:1: Confusion Matrix for kNN

In Figure 5.1, the actual values of the class labels 0, 1, 2 and 3 that were predicted correctly are 10, 10, 6 and 14, respectively. The values on the other cells were misclassified by the model. The total number of the values in the confusion matrix in Figure 5.1 is 60, which is the size of testing set, obtained from a 30% split ratio.

The ROC curves were also used in checking the performance of the kNN model. Two sets of ROC curves were generated where one is the macro average curve for the four class labels and the other is the ROC curves for the individual class labels. The curves are as shown in Figures 5.2 and 5.3.

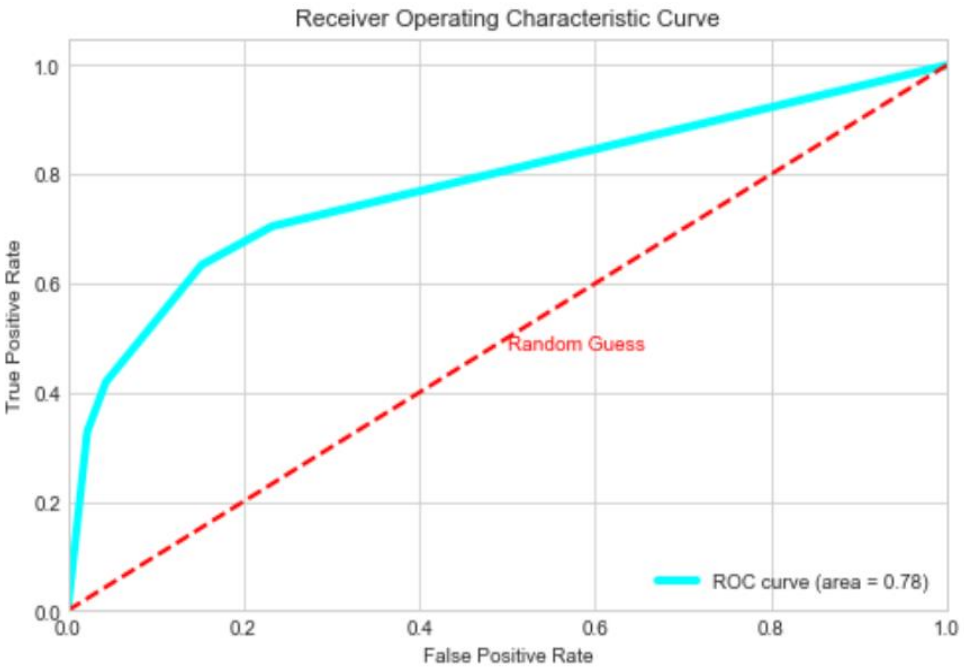

Figure 5:2: Macro-Average ROC Curve for kNN

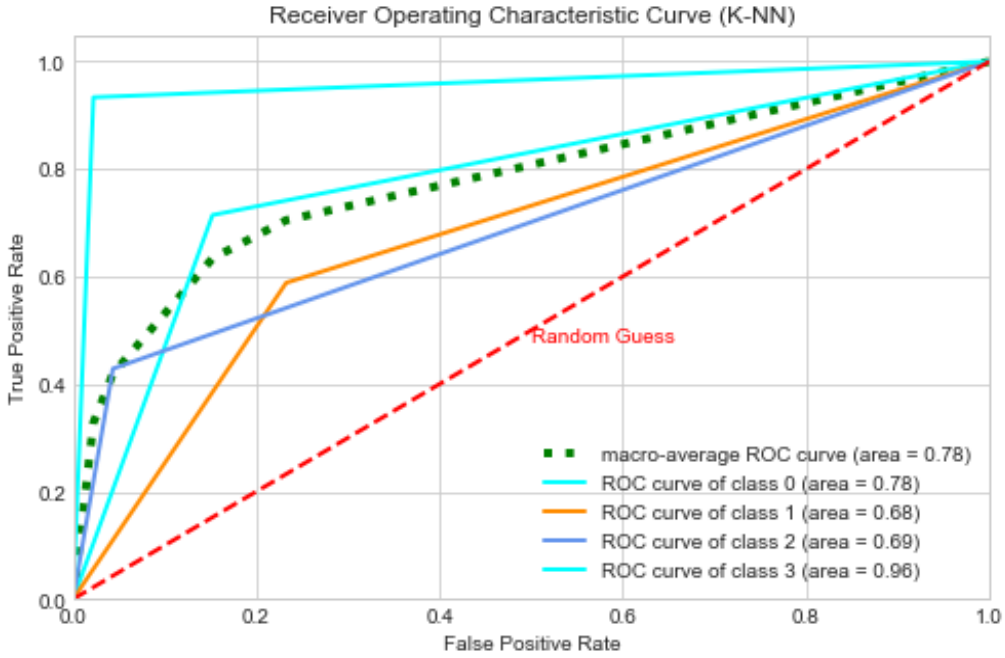

Figure 5:3: ROC Curves for Each Class Labels for kNN

The ROC curves evaluate a model by checking the area under the curve. The macro average area under the curve for all the classes is 0.78. The area under the curves for the class labels 0, 1, 2, and 3 are 0.78, 0.68, 0.69 and 0.96, respectively. Since these scores are evaluated on the

range of 0 to 1, the scores obtained from kNN model show that the algorithm makes a good classification model.

5.4.2. Support Vector Machines

This is a Machine Learning algorithm that can be used for classification, regression and outlier detection. The algorithm performs classification by creating a hyperplane between the vector points and thereafter finding the minimum distance between the hyperplane separating the vector points. It is effective in spaces with high dimensional prospects and consists of different kernels that can be specified for a specific decision function.

The implementation of the algorithm is done using the SVC inbuilt Python function. For this experiment, classification commences by splitting the dataset into the training and testing set using a ratio of 0.30. Some of the parameters utilized include *C* which is the penalty parameter of the error term and has been set to the default value of 1.0. The other parameter is the *random_state* which is the seed of the pseudo-random number generator; it is used when shuffling the data for probability estimates; in the experiment, it has been set to 0 since the model will calculate the probabilities for performing predictions. The *probability* parameter has been set to True. Different kernels were used in implementing Support Vector Machines and include Linear, Polynomial, RBF and Sigmoid kernels. The performance of the classification model according to each kernel is given in Table 5.2.

Table 5:2: The Performance of the different kernels

		Naïve Bayes			
Kernel	**Class Label**	**Precision**	**Recall**	**F-Measure**	**Support**
Linear	0	0.48	0.93	0.63	14
	1	0.60	0.18	0.27	17
	2	0.85	0.79	0.81	14
	3	0.93	0.93	0.93	15
	Average/Total	**0.71**	**0.68**	**0.65**	**60**
Polynomial	0	0.25	0.93	0.40	14
	1	0.00	0.00	0.00	17
	2	0.00	0.00	0.00	14
	3	1.00	0.53	0.70	15
	Average/Total	**0.31**	**0.35**	**0.27**	**60**
RBF	0	0.39	0.43	0.60	14
	1	0.00	0.82	0.72	17
	2	0.89	0.86	0.75	14
	3	0.78	0.87	0.90	15
	Average/Total	**0.52**	**0.61**	**0.58**	**60**
Sigmoid	0	0.34	0.93	0.50	14
	1	1.00	0.06	0.11	17
	2	0.67	0.29	0.40	14
	3	0.87	0.87	0.87	15
	Average/Total	**0.74**	**0.52**	**0.52**	**60**

The Linear kernel displayed the best results amongst the kernels used in the experiment and was selected for performance evaluations with the confusion matrix and ROC curves. The confusion matrix for the classifier is shown in Figure 5.4. In the confusion matrix table, the number of ontologies supplied by the model with the class labels 0, 1, 2 and 3 that were predicted correctly by the model were 13, 3, 11 and 14, respectively. The values tabulated on the other cells were misclassified by the model.

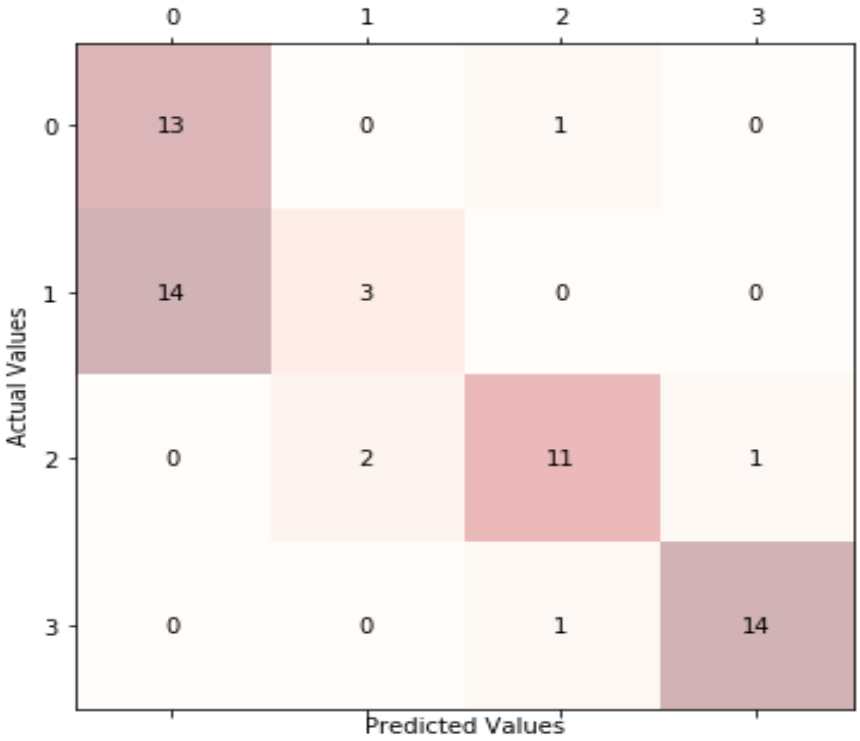

Figure 5:4: Confusion Matrix for SVM Classifier

The model was also evaluated using the ROC curves. Figure 5.5 shows the ROC curves for each of the class labels used in the dataset. The area under the curves for the class labels 0, 1, 2 and 3 are 0.81, 0.56, 0.87 and 0.96, respectively. Furthermore, Figure 5.6 shows that the area under the curve for the macro average of the four-class labels is 0.8.

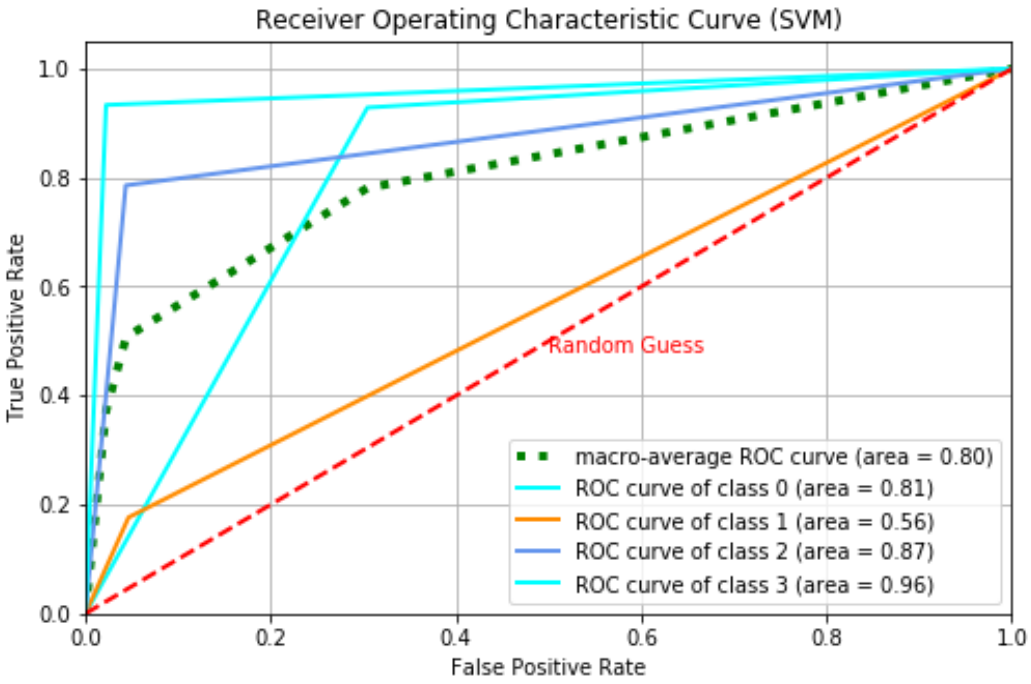

Figure 5:5: ROC Curves for Each Class Labels for SVM Classifier

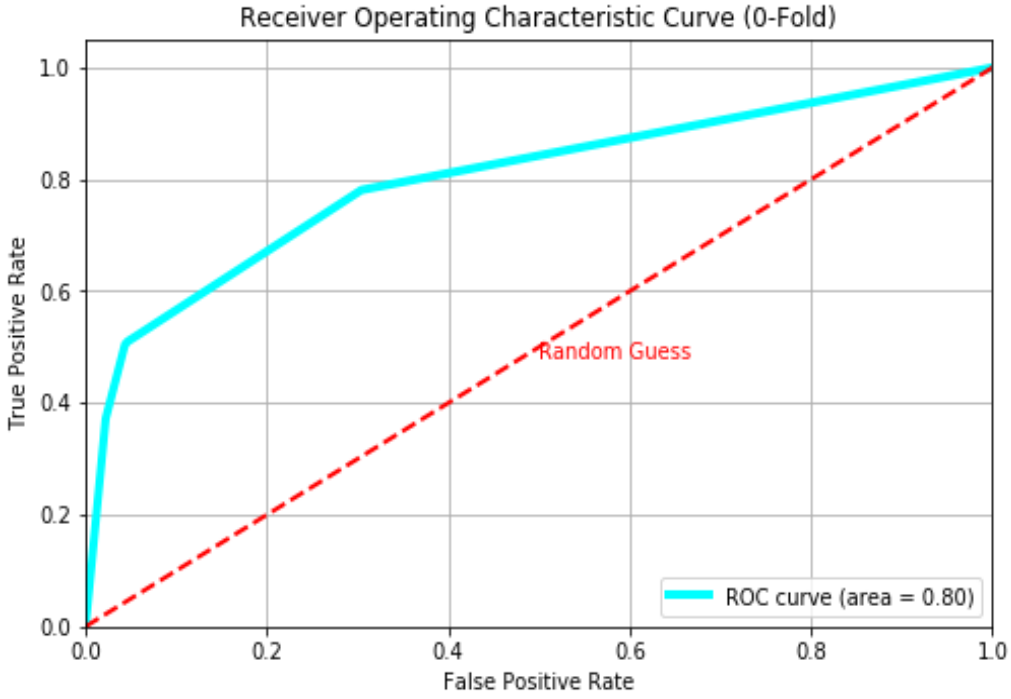

Figure 5:6: Macro Average ROC Curve for SVM

5.4.3. Logistic Regression

Logistic regression is a linear machine learning classification model where the probabilities describing the possible outcomes of a single trial are modelled using a logistic function. The Python *LogisticRegression* function was used in building the model; the function can either be used in binary, One-vs-Rest (OVR) or multinomial logistic regression using an elastic-Net regularization. The Elastic-Net regularization is used in minimizing the cost function and solving optimization problems in the model.

The dataset is first split into training and testing set with a ratio of 0.3 . The parameters used in the Python *LogisticRegression* function in building the model include the *penalty*, which specifies the norm for penalization and allows for regularization. The *C* parameter is the inverse of the regularization strength and must always be a positive float. This is because smaller *C* values specify stronger regularization. Intercept scaling is useful when the solver 'liblinear' is used, therefore, it is set to be true. The synthetic feature weight is dependent on the regularizations made just like other features. The class weight is associated with the classes that are in the form {class_labels: weight} and should have at least one weight. The "balanced" mode uses the y values to automatically adjust weights that are inversely proportional to class frequencies in the input data. These parameters have been outlined in Table 5.3 after a randomized search was conducted to ascertain the best parameters to work with.

Table 5:3: Randomized Search Results showing the performance of the different parameters used

Model Rank	Mean Validation Score	Std	Penalty	Intercept Scaling	Class Weight	C
1	0.753	0.046	11	379190882.0128173	Balanced	20.99578965767907
2	0.747	0.087	11	1933.430800622057	Balanced	80740365128.85052
3	0.547	0.040	12	1.572605035563863e-14	Balanced	118797549.0713333
4	0.253	0.005	12	7025620.200406866	Balanced	1846.4035763068719
5	0.253	0.005	12	3560862102085029.5	Balanced	2.2768568864827288e-07

The best parameters obtained from the search were the penalty: l1, intercept_scaling: 379190882.0128173, class_weight: balanced and C: 20.99578965767907. These parameters were used in building the Logistic Regression model in this experiment. The model took a running time of 0:00:00.566853 and an overall accuracy score of 64% was obtained and upon doing 10-fold cross-validation, the accuracy slightly improved to 64.67%. A detailed performance measurement of the model is depicted by the confusion matrix table in Figure 5.7.

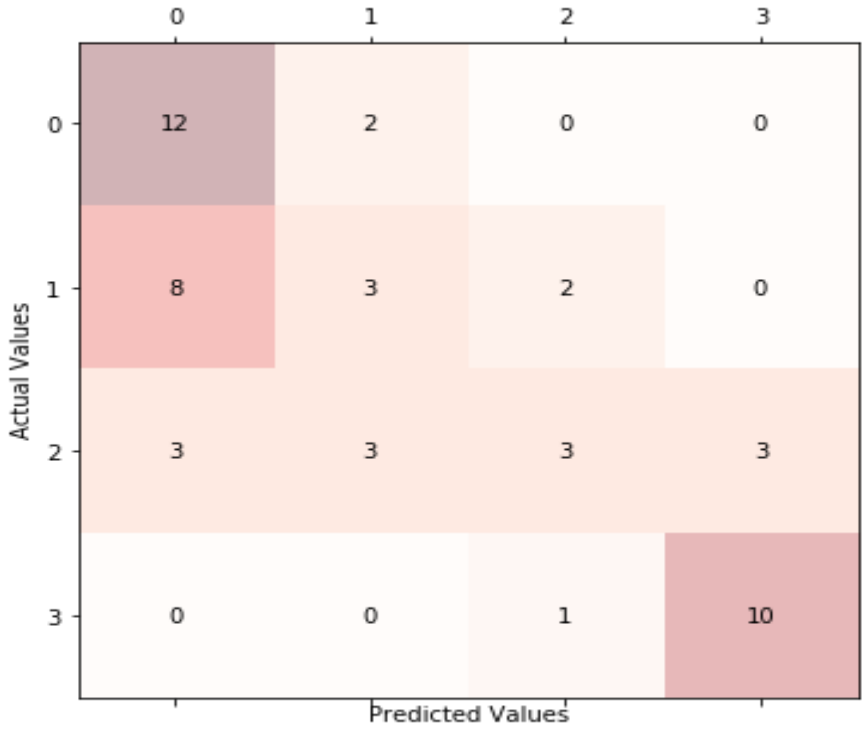

Figure 5:7: Confusion Matrix for Logistic Regression Classifier

From the confusion matrix table (Figure 5.7), the number of actual values of ontologies with the class labels 0, 1, 2 and 3 were predicted correctly by the model are 12, 3, 3 and 10, respectively. The values on the other cells in the table were misclassified by the model. Focusing on these misclassified values, it can be pointed out that there are some cells in the table with rather large numbers such as 8 and 3. These values contribute to the accuracy values depicted by the model. The table sums to a total of 60 which is the representation of the 30% used as the testing set.

The results obtained from the confusion matrix are captured in Table 5.4. It is shown that the average precision, recall and F-Measure scores for Logistic Regression model are 53%, 56% and 52%, respectively.

Table 5:4: Precision, Recall, and F -Measure Scores for Logistic Regression

	Logistic Regression			
Class Label	**Precision**	**Recall**	**F-Measure**	**Support**
0	0.52	0.86	0.65	14
1	0.38	0.23	0.29	13
2	0.50	0.35	0.33	12
3	0.77	0.91	0.83	11
Macro Avg.	0.54	0.56	0.53	50
Average/Total	0.53	0.56	0.52	50

The ROC curves were also used in evaluating the performance of the model. On these curves, a general curve of the average of all the classes are generated and curves for the respective classes are also generated. The area under the curves for the class labels 0, 1, 2 and 3 are 0.78, 0.55, 0.59 and 0.92, respectively are shown in Figure 5.8. The macro average area under the curve for the average of the four-class labels is 0.71 as shown in Figure 5.9.

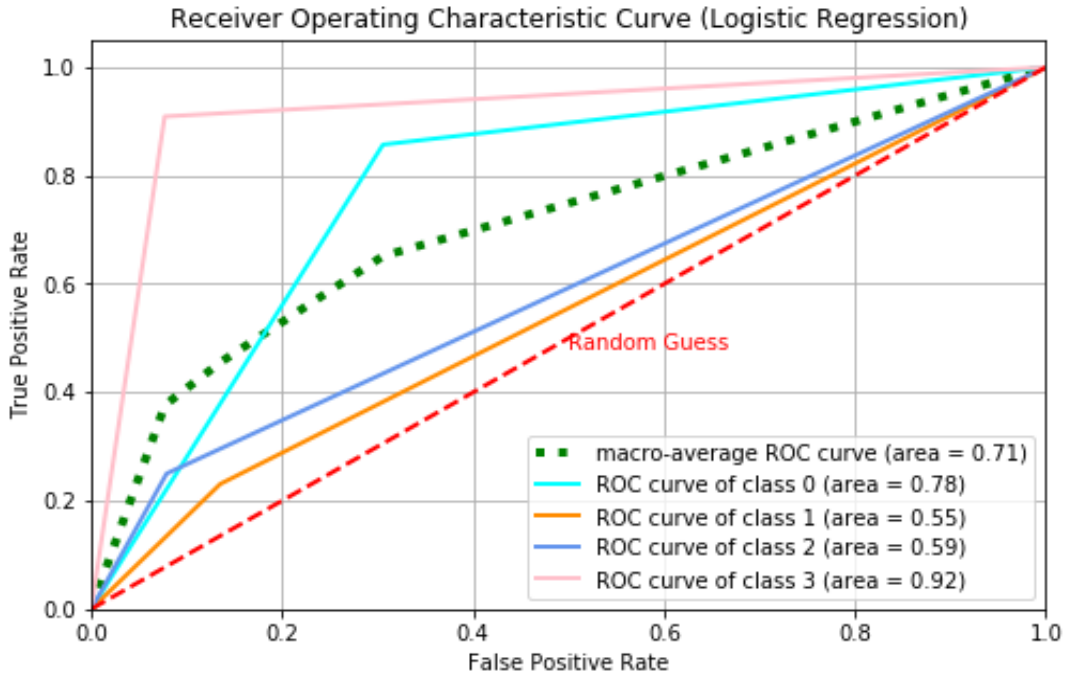

Figure 5:8: ROC Curves for Each Class Label for Logistic Regression Classifier

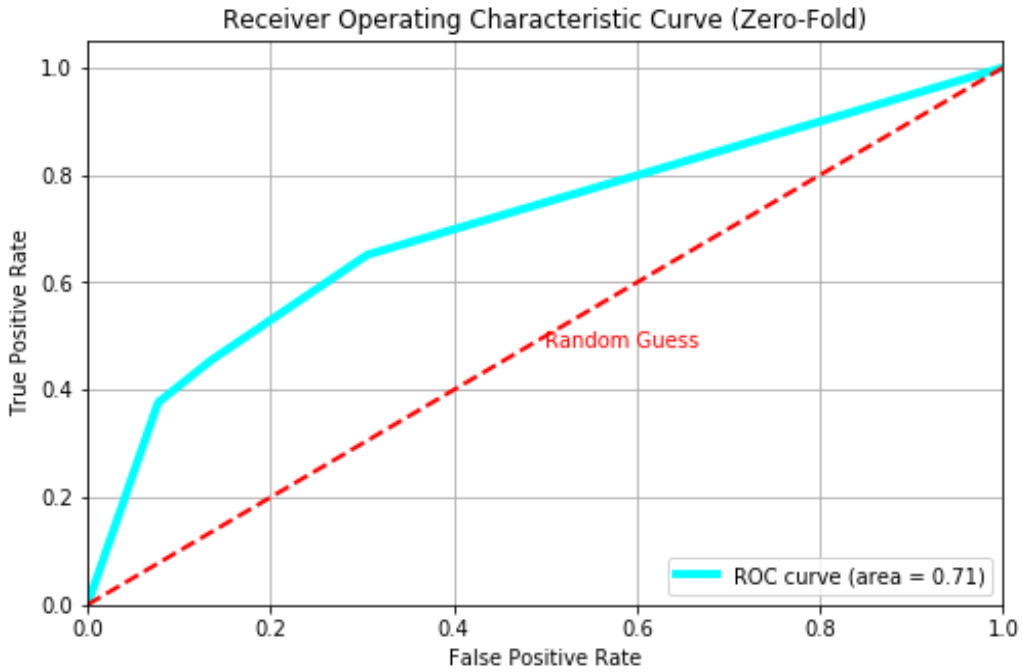

Figure 5:9: Macro Average ROC Curve for Logistic Regression Classifier

5.4.4. Decision Trees

Decision Trees is a non-parametric supervised learning technique that creates a classification model that learns the decision rules from the data then predicts the target variable value. The Python *DecisionTreeClassifier* function is used in this experiment to perform a multiclass classification with four classes.

The *DecisionTreeClassifier* function takes as parameters two arrays X and Y comprising of the training and testing sets, respectively. The training set is of the size [n_samples, n_features] whereas the testing set is of the size [n_samples] and stores the class labels of the training samples. In this experiment, a ratio of 0.3 has been used in splitting the dataset into these two sets where the training set, takes 70% and makes up 140 ontologies on the dataset whereas the testing set is 30% and comprises 60 ontologies.

The building of the model is preceded by fitting it to the training and testing sets. The probabilities of each of the class labels are also predicted and are the fraction of the training samples of a particular class on one leaf. A plot of the tree able to view the specific features that have been used in building the model is shown in Figure 5.10. The total samples indicated at the parent node are 140 which is the size of the training set after splitting the dataset with 70%. The initial entropy for the parent node is 0.742 and the values used are [46, 31, 30, 33].

The decision node takes 10 values and forms a logic test of less than or equals to -0.478. The model takes these values because it's not homogeneous and as it splits the dataset further, metrics with similar characteristics are grouped together and becomes homogenous which explains the zero values of the entropies in the edge nodes.

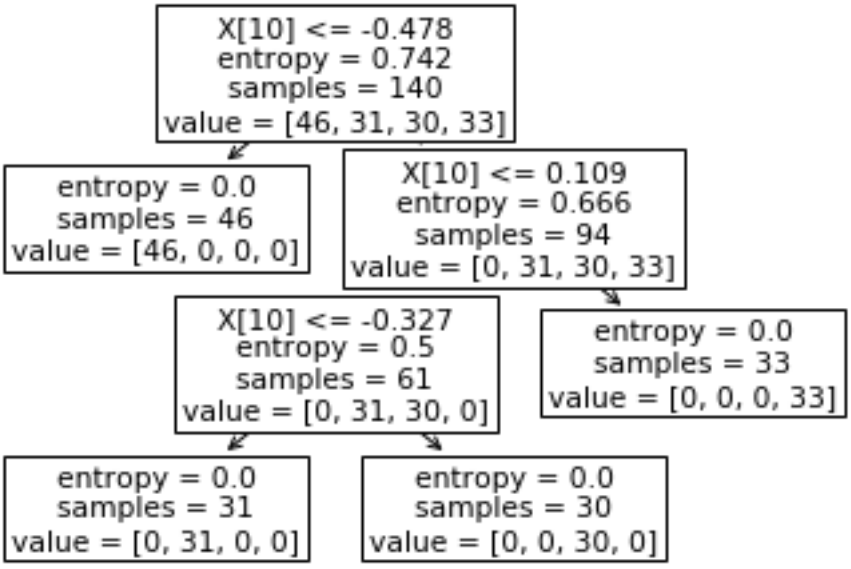

Figure 5:10: Structure of the Decision Tree used in the Classification

A summary of the performance of the Decision Tree model can be viewed using the confusion matrix table. The table gives a summary of the actual values supplied into the model and the values that have been predicted by the model. The Figure 5.11 shows the confusion matrix generated for a Decision Tree model. The actual values with the labels 0, 1, 2 and 3 that have been correctly classified are 13, 17, 14 and 15, respectively. Only one class label of 0 has been predicted by the model as a label with 2 hence misclassified.

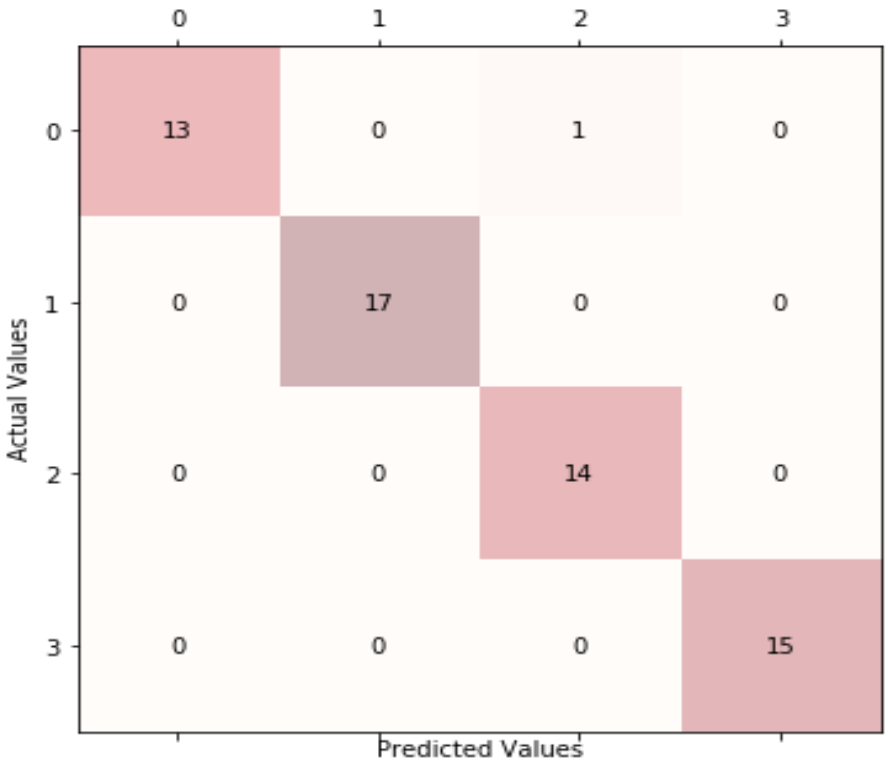

Figure 5:11: Confusion Matrix for Decision Tree Classifier

From the confusion matrix table, more performance evaluation summaries can be deducted. These are the precision, recall and F-Measure scores of each class label used in building the model. These values have been tabulated on Table 5.5 and the overall average of the precision, recall, and F-Measure are all 0.98. Ontologies that have been predicted into the class labels 0, 1, 2 and 3 are 14, 17, 14 and 15, respectively.

Table 5:5: Precision, Recall and F -Measure Scores for Decision Trees

	Decision Trees			
Class Label	**Precision**	**Recall**	**F-Measure**	**Support**
0	1.00	0.93	0.96	14
1	1.00	1.00	1.00	17
2	0.93	1.00	0.97	14
3	1.00	1.00	1.00	15
Macro Avg.	0.98	0.98	0.98	60
Average/Total	0.98	0.98	0.98	60

Testing the functionality of the algorithm using the ROC curve, the overall area under the curve for the model is 0.99. ROC curves for each of the class labels were also generated to further check the model. For the area under the curve for the class labels 0, 1, 2 and 3 are 0.96, 1.00, 0.99 and 1.00, respectively. These values are as shown in the Figures 5.12 and 5.13.

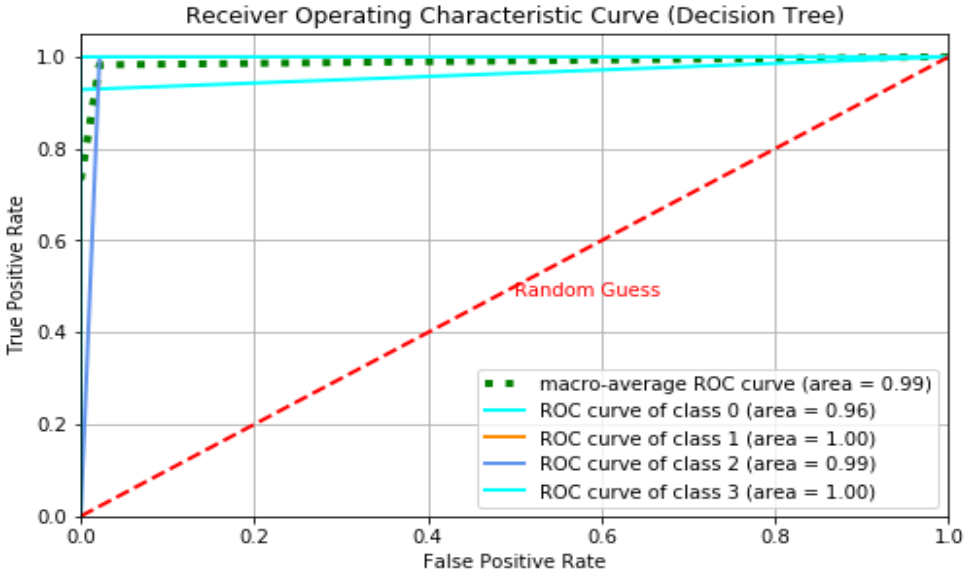

Figure 5:12: ROC Curves for Each Class Label for Decision Tree Classifier

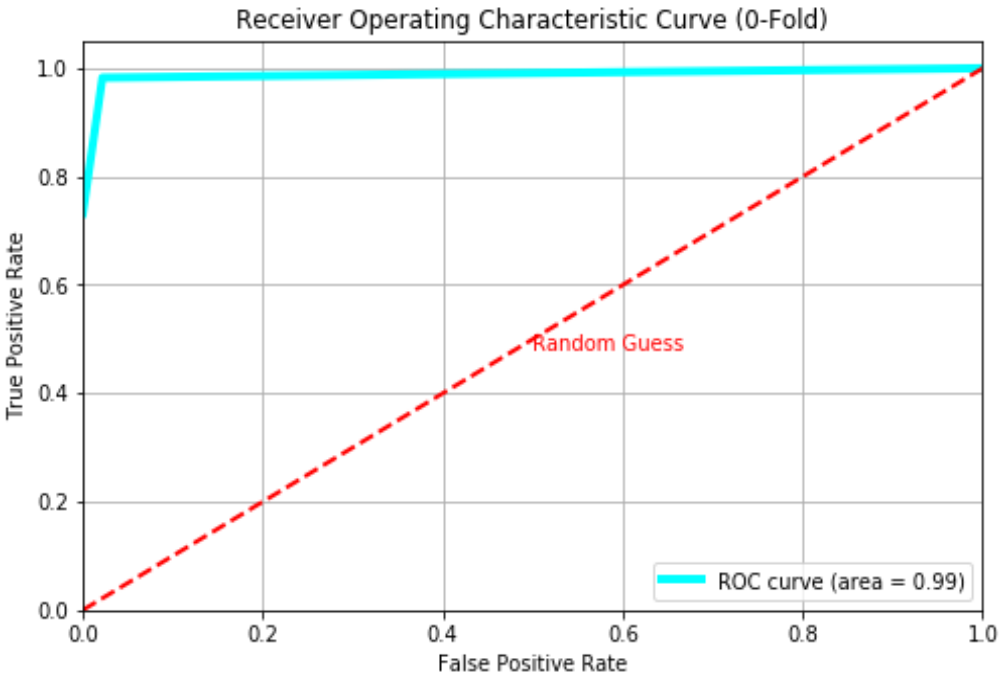

Figure 5:13: Macro Average ROC Curve for Decision Tree Classifier

5.4.5. Random Forest

Random forest is an ensemble machine learning method, that is implemented by building multiple numbers of decision trees during the training phase and giving an output of the average of the individual trees in the forest's prediction. The Python *RandomForestClassifier()* function is used in building the model for classification.

A ratio of 0.3 has been used in splitting the dataset into the training and testing set. From this ratio, a total of 140 ontologies made up the training set whereas 60 ontologies fall into the testing set. The construction of the tree entails splitting the nodes and the best split is obtained from either all input features or a random subset of size *max_features*. The two sources of randomness are used to reduce the variance of the forest estimator. The discrete decision trees have high variance and always tend to overfit. The use of randomness in forests yield decision trees with decoupled prediction errors. The average of these predictions cancels out some errors in the model and by so doing, the model achieves a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. This variance reduction is often significant, hence yielding an overall better model.

The main parameters adjusted in the model are *n_estimators* and *max_features*. The *n_estimators* defines the number of trees in the forest and conventionally, a large number gives

a better model, and, in this experiment, 10 trees have been used in building the model. The *max_features* parameter defines the size of the random subsets of features to be considered when splitting a node. The lower the value, the greater the reduction of variance which will result in a good model and for most empirical good default values, it is set at *max_features=None* as is the case in this experiment.

The *max_depth* parameter is the depth of the tree and has been set to *none* to allow the model to accommodate as more trees as possible. The bootstrap samples are used by default (bootstrap=True) while the default strategy for extra-trees is to use the whole dataset (bootstrap=False) which has been used in this experiment. Since bootstrap has been set as false in this case, the generalization accuracy cannot be estimated on the left out or *out-of-bag* samples and is the reason why the was set to *oob_score=False*.

The confusion matrix is used in evaluating the overall performance of the model. The confusion matrix shows the actual and the predicted classes of ontologies bearing in mind that it is a multiclass classification. Figure 5.14 shows the confusion matrix of the Random Forest classifier.

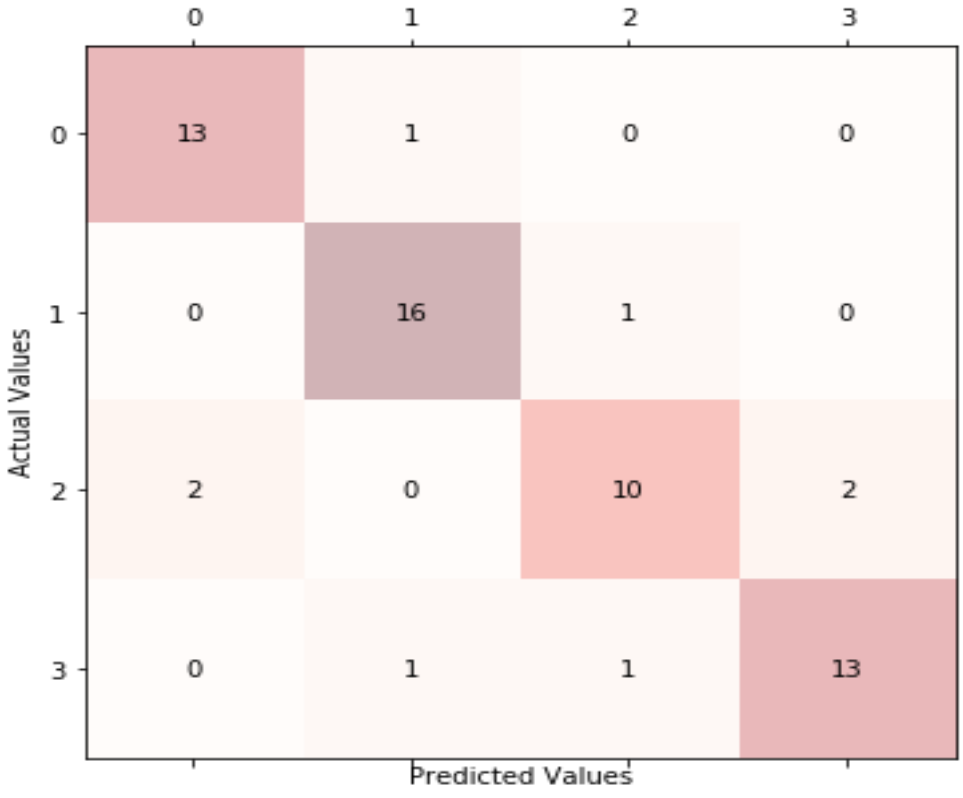

Figure 5:14: Confusion Matrix for Random Forest classifier

From the confusion matrix table in Figure 5.14, the left side shows the actual values whereas the values cutting across the table are predicted values. Figure 5.14 shows that the number of ontologies with the class labels 0, 1, 2 and 3 predicted correctly by the model are 13, 16, 10 and 13, respectively. These numbers of correctly predicted ontologies appear on the right diagonal of the confusion matrix in Figure 5.14. The values on the other cells in Figure 5.14 were misclassified. The confusion matrix table sums to a total of 60 which is the representation of the 30% used as the testing set. Out of the confusion matrix table, precision, recall and F-Measure scores are tabulated. For the Random Forest model, the scores are tabulated in Table 5.6.

Table 5:6: Precision, Recall, and F -Measure Scores for Random Forest

	Random Forest			
Class Label	**Precision**	**Recall**	**F-Measure**	**Support**
0	0.87	0.93	0.90	14
1	0.89	0.94	0.91	17
2	0.83	0.71	0.77	14
3	0.87	0.87	0.87	15
Macro Avg.	0.86	0.86	0.86	60
Average/Total	0.87	0.87	0.86	60

A randomized search has been performed on the model using the *RandomizedSearchCV* function. This entails searching the correct or the desired hyperparameter that will find the highest precision and accuracy of the model. In this search, random combinations of the parameters were used in finding the best solution for the model that was built. As a result of randomly selecting the values at each instance, the whole of the action space will most likely be reached because of randomness since it takes plenty of time to cover all the space available. In this search there are higher chances of finding the most optimal parameter that will give the best results. The *RandomizedSearchCV* took 5.04 seconds for 10 candidates parameter settings. The results of the model are as shown in Table 5.7.

Table 5:7: Randomized Search Results showing the performance for the different parameters used for Random Forest Classifier

Model Rank	Mean Validation Score	Std	Bootstrap	Criterion	Max depth	Max features	Min Samples Leaf	Min Samples Split
1	0.986	0.010	True	Gini	None	7	5	13
2	0.986	0.010	True	Gini	10	8	7	5
3	0.979	0.018	True	Gini	None	8	2	6
4	0.971	0.027	False	entropy	None	8	4	3
5	0.971	0.021	False	entropy	10	6	7	3

After this search, the model took a running time of 0:00:00.146909 and displayed an overall accuracy of 96.67% which is regarded as a good accuracy score. To obtain the best accuracy results, the parameters are cross validated with a specified fold. In this experiment, 10 cross-validation folds were performed, and the overall accuracy reached 99.29%.

Upon testing the functionality of the algorithm using the ROC curves in Figures 5.15 and 5.16, the macro-average area under the curve for the Random Forest Model is 0.89. The ROC curve for each of the classes (0,1,2 and 3) was calculated as in Figure 5.15. The areas under the curves for these classes are 0.96, 0.89, 0.84 and 0.87, respectively. These results indicate that model performs well in classifying ontologies.

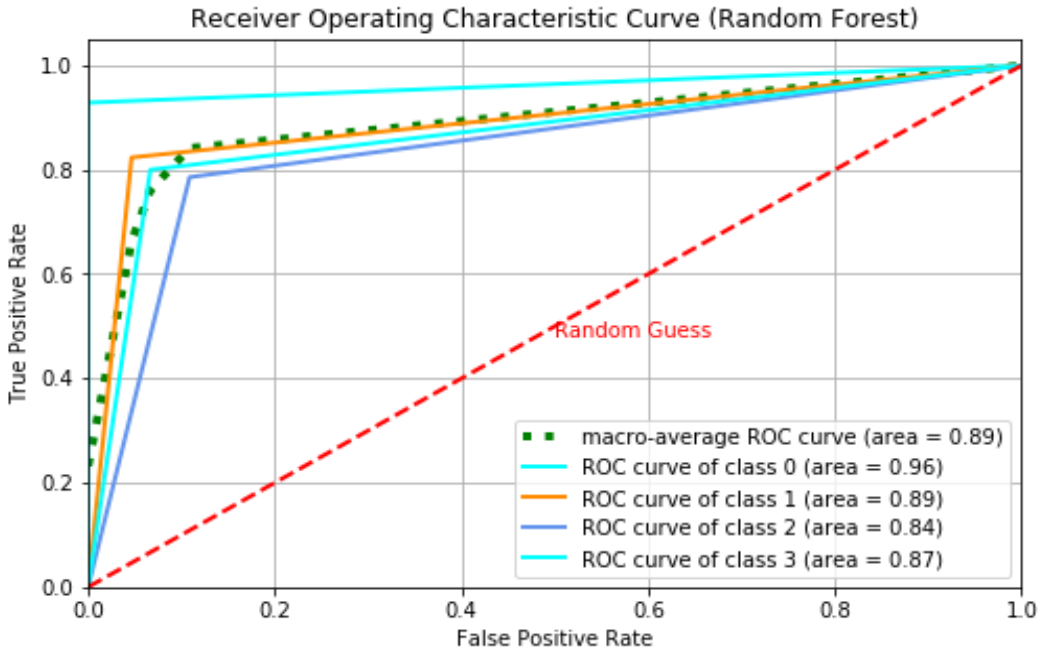

Figure 5:15: ROC Curves for Each Class Label for Random Forest Classifier

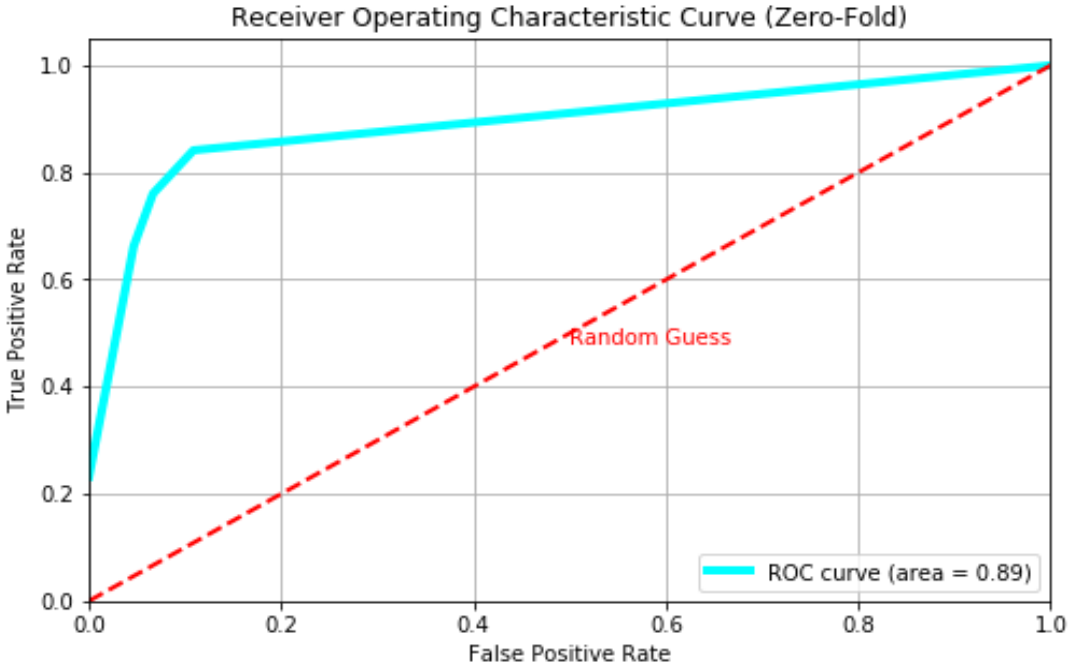

Figure 5:16: Macro Average ROC Curve for Random Forest Classifier

5.4.6. Naïve Bayes

Naïve Bayes is a supervised machine learning algorithm that contains methods based on the Bayes' probabilistic theorem while keeping a 'naive' assumption of conditional independence

between every pair of features given the value of the class variable. The Python *GaussianNB* function is used in building a classification model.

The initial phase of conducting this experiment is splitting the dataset into the training and testing sets. 70% of the dataset, comprising of 140 ontologies has been set as the training set while 30% is the test set which is comprised of 60 ontologies. The model is then built by fitting the training and testing sets into the classifier.

The confusion matrix was used in evaluating the performance of the Naïve Bayes classifier as shown in Figure 5.17. On the confusion matrix table, the number of ontologies with the class labels represented by 0, 1, 2 and 3 that have been correctly predicted by the classifier are 6, 14, 12 and 13, respectively. The values on the other cells represent misclassification by the classifier and explain the reason why the model is not 100% accurate.

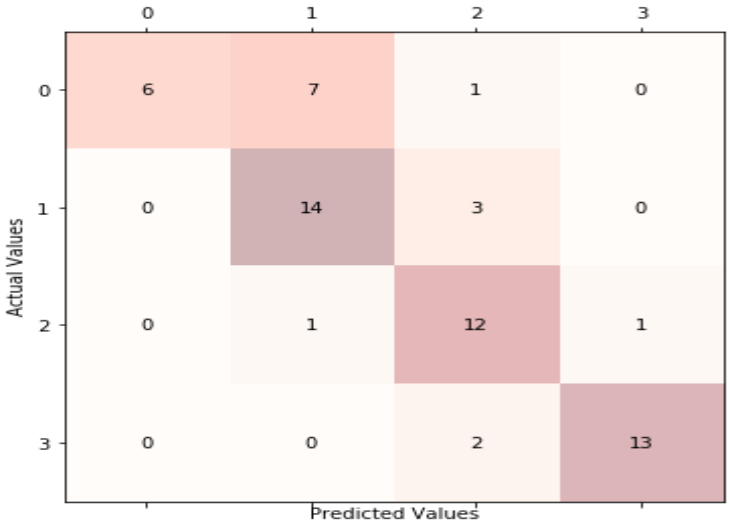

Figure 5:17: Confusion Matrix for Naive Bayes Classifier

Out of the confusion matrix, we can tabulate the precision, recall and F-Measure scores of the model that is shown on Table 5.8. The average precision recall and F-Measure scores of the Naïve Bayes classifier are 0.80, 0.75 and 0.74, respectively and all the 60 ontologies that were set aside for testing set have been predicted here.

Table 5:8: Precision, Recall and F-Measure Scores for Naive Bayes Classifier

	Naïve Bayes			
Class Label	**Precision**	**Recall**	**F-Measure**	**Support**
0	1.00	0.43	0.60	14
1	0.64	0.82	0.72	17
2	0.67	0.86	0.75	14
3	0.93	0.87	0.90	15
Macro Avg.	0.81	0.74	0.74	60
Average/Total	0.80	0.75	0.74	60

Naïve Bayes Classifier is a probabilistic algorithm, and this makes its fit to use a Receiver Operating Characteristic (ROC) Curve in measuring its performances as in Figures 5.18 and 5.19. The overall average of the Area under the Curve for all the classes is 0.83. The individual area under the curves for the class labels 0, 1, 2 and 3 are 0.71, 0.82, 0.86 and 0.92, respectively. Since these values are closer to 1, suggesting that the algorithm performed well in classifying the ontologies using its metrics making it a good classifier.

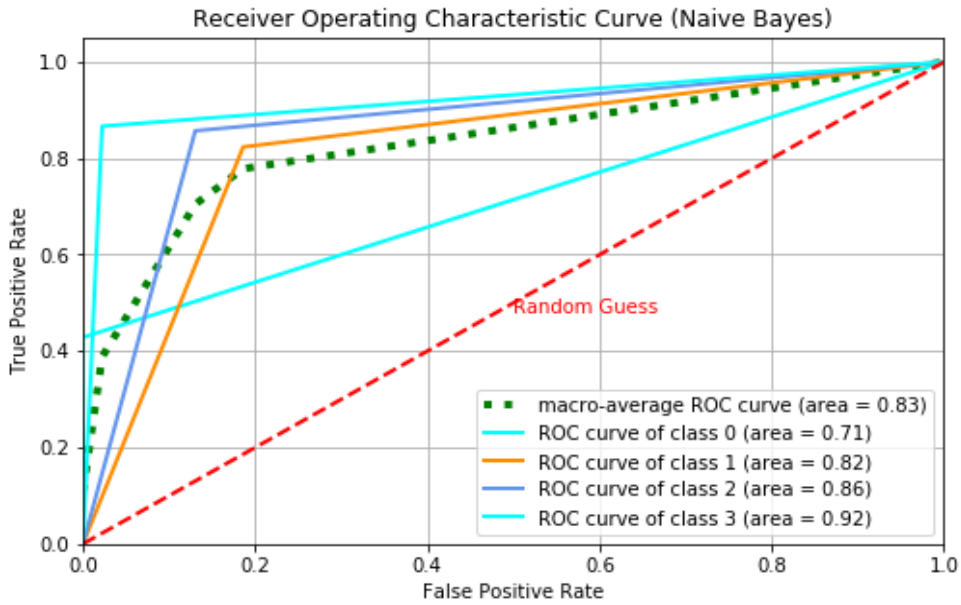

Figure 5:18: ROC Curves for Each Class Label for Naïve Bayes Classifier

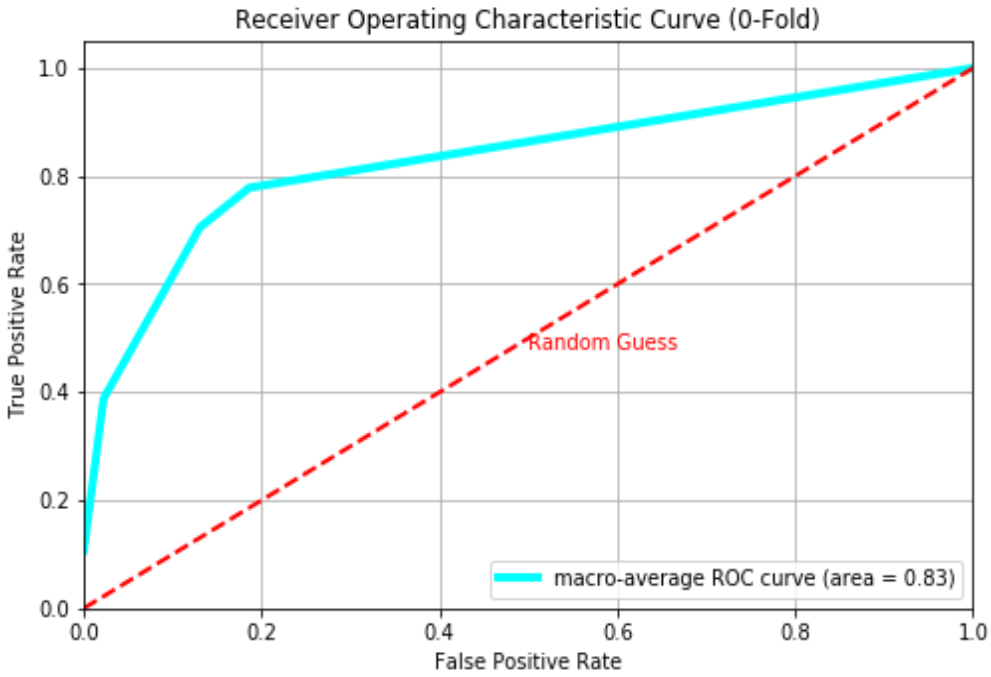

Figure 5:19: Macro Average ROC Curve for Naive Bayes Classifier

5.4.7. Linear Regression

Linear Regression is a machine learning technique, mainly used in regression analysis but also used in solving classification tasks. The algorithm uses the Python *LinearRegression*() function in building a classification model and contains a number of parameters that all perform classification.

In this experiment, a ratio of 0.3 has been used in splitting the dataset into the training and testing sets. Some of the parameters used by the Linear Regression model include *copy_X*, *fit_intercept*, *n_jobs* and *normalize*. The parameter *copy_X* is used to allow for the creation of *X* values and will only copy if it is set to true else will overwrite. Check this in the experiment, it was specified to True. The *Fit_intercept* parameter calculates the intercept of the model and is set to the default true allow for the calculations. The *n_jobs* parameter specifies the number of jobs to be used for computation. The *Normalize* parameter is ignored when *fit_intercept* is set to False. If True, the regressors *X* will be normalized before regression by subtracting the mean and dividing by the *l2-norm*.

Since the algorithm performs classification by building regression models, the confusion matrix and ROC curves cannot be used in evaluating the performance of the algorithm. Some of the techniques that have been used here include Mean Absolute Error (MAE), Mean Squared

Error (MSE), Root Mean Squared Error (RMSE), Median Absolute Error (MAE), Explained Variance Score (EVS) and the R2 Score. These evaluation techniques evaluate the algorithm by finding a center of optimization, thereby, finding the errors that exist in the model. The scores obtained from these performance evaluation techniques are shown in Table 5.9.

Table 5:9: Performance Scores for Linear Regression

Linear Regression	
Performance Metric	**Score**
Mean Absolute Error (MAE)	0.6387547898451156
Mean Squared Error (MSE)	0.6020058170067685
Root Mean Squared Error (RMSE)	0.7992213647326475
Median Absolute Error (MAE	0.6267950618182005
Explained Variance Score (EVS)	0.506690664026025
R2 Score	0.5052006983506012

These scores give the error of a classification model and high values indicate that the model is inaccurate in performing classification. The values in this table, however, are close to zero hence the model is slightly accurate.

5.4.Conclusion

This chapter presented the experimental results for the implementation of Machine learning algorithms. Here, the process of building the model has been stipulated where the ratio used in splitting the data into training and testing sets is. Performance evaluations of each algorithm have also been discussed where the confusion matrix and ROC curve results have been discussed.

CHAPTER 6. : CONCLUSION AND FUTURE WORK

6.1.Summary of the Study

This study aimed at developing Machine Learning models for the classification of ontologies based on their complexity metrics. The main idea in this study is the fact that as ontologies grow in numbers and size, they tend to get more complex. Therefore, they become difficult to select, reuse and maintain them. A total of 200 ontologies obtained from the National Center for Biomedical Ontologies (NCBO) repository were used in the experiments in this study. The metrics of these ontologies have been used to form the dataset for the classifications of ontologies with Machine Learning algorithms.

The implementation of Machine Learning algorithms was done using python environment and specifically Jupyter Notebook in Anaconda Package. Prior to the building of the model, exploratory data analysis was done on the dataset to eliminate noises. The dataset was first split into the training and testing sets with a ratio of 0.3. The building of machine learning models were carried out with existing functions of each algorithm. The functions used in building the models consist of different specific parameters used in achieving classifications. During classification, a randomized search was done on the algorithms with a higher number of parameters to determine the parameters that produce the best results.

The performance of each model was evaluated to ascertain its accuracy scores. Various techniques were used to evaluate the performances of the machine learning models including precision, recall, f-measure scores, and ROC curves.

6.2. Limitations

During this study, some challenges occurred which may have affected the process of classification of ontologies.

- The implementation of classification algorithms was only limited to ontologies with less than 10,000 classes. This was because the OntoMetrics platform used in generating the ontology metrics cannot handle large ontologies of more than 20 000 classes.
- Python's Jupyter notebook used in the implementation of Machine Learning algorithms is constantly being updated with more libraries added to the environment. This affects the results obtained after implementing the algorithms. This includes exploratory data analysis phase, model fitting and data visualizations.

6.3.Recommendations and Future Work

This study focused on the ontologies that were obtained from the NCBO Bio Portal repository. The respective metrics were generated only from OntoMetrics online platform. Future research could focus on using ontologies from different repositories. The implementation of platforms that can generate ontology metrics of very large ontologies could also bring an impact to future research.

The Machine Learning models created in this study from Python language using the Jupyter Notebook environment. Several inbuilt libraries that are integrated into the Notebook environment were used in the process. However, there are other Platforms that can be used in implementing classification using Machine Learning algorithms. These platforms include R, Hadoop, Orange, Weka, etc. and could be used in the future in performing classification of ontologies.

In this study, seven Machine Learning algorithms were implemented in the classification of ontologies. These algorithms have been used widely in performing most classification tasks and have provided excellent results. However, there are more powerful and advanced Machine Learning algorithms that can handle datasets having both multiclass and multilabel attributes. The future direction of this work can investigate the implementation of these algorithms in the classification of ontologies. Some of these algorithms include Apriori, AdaBoost, Gradient Boosting and Deep Learning using Neural Networks among others.

6.4.Conclusion

This study led to the classification of ontologies based on their complexity metrics using machine learning algorithms. Clustering analysis using the K-Means algorithm was first implemented followed by Machine Learning classification algorithms.

The classification models were evaluated using performance evaluation techniques including precision, recall, F-Measure scores as well as The ROC curves. A summary of the results obtained for each algorithm is as follows: the accuracy scores for K-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forest, Naïve Bayes, Logistic Regression and Linear Regression algorithms were 66.67%, 65%, 98%, 99.29%, 74%, 64.67%, and 57%, respectively. From these scores, Decision Trees and Random Forests algorithms were the best performing and can be attributed to the ability to handle multiclass classifications.

References

Aborisade, O. & Anwar, M. 2018. Classification for Authorship of Tweets by Comparing Logistic Regression and Nave Bayes Classifiers. *In Proceeding of IEEE International Conference on Information Reuse and Integration for Data Science*. pp. 269 - 276, Salt Lake City, UT, USA.

Agrawal, R. 2014. K-Nearest Neighbors for Uncertain Data. *International Journal of Computer Applications (0975-8887).* 105(11):13-16.

Ali, F., Khan, P., Riaz, K., Kwak, D., Abuhmed, T., Park, D. & Kwak, K. S. 2017. A Fuzzy Ontology and SVM-Based Web Content Classification System. *International Journal of Sensors,* 19(2): 25781 – 25797.

Ali, F., Kwak, K. S. & Kim, Y. G. 2016. Opinion mining based on fuzzy domain ontology and Support Vector Machine: A proposal to automate online review classification. *International Journal of Soft Computing,* 47: 235 – 250.

Ali, S., Joshi, N., George, B., & Vanajakshi, L. 2012. Application of Random Forest Algorithm to Classify Vehicles Detected by a Multiple Inductive Loop System. *In Proceedings of 15th International IEEE Conference on Intelligent Transportation Systems Anchorage,* Alaska, USA, pp. 491-496.

Al-Masri, A. 2018. What are Training, Validation and Test Data Sets in Machine Learning? Retrieved From: https://medium.com/datadriveninvestor/what-are-training-validation-and-test-data-sets-in-machine-learning-d1dd1ab09bae Accessed 23rd March, 2019.

Alsagheer, R. H. A., Alharan, A. F. H., & Al-Haboobi, A. S. A. 2017. Popular Decision Tree Algorithms of Data Mining Techniques: A Review. *International Journal of Computer Science and Mobile Computing.* 6(6): 133-142.

Amores, J., Sebe, N. & Radeva, P. 2006. Boosting the distance estimation: Application to the K-Nearest Neighbor Classifier. *International Journal of Pattern Recognition Letters.* 27: 201-209.

Arroyo, S., Lara, R., Ding, Y., Stollberg, M. & Fensel, D. 2014. Semantic Web Languages – Strengths and Weakness. Available at

http://info.slis.indiana.edu/~dingying/Publication/Semantic_Web_Languages._Weakness_and_Strengths.pdf Accessed 27th February 2018.

Aurangzeb, K., Baharum, B., Lam, H. & Khairullah, K. 2010. A Review of Machine Learning Algorithms for Text-Documents Classification, *International Journal of Advances in Information Technology*, 1(1): 4-20

Bachir, A. & Benslimane, S. 2013. User-Cantered Approach for Evaluating Ontologies, *In Proceedings of International Conference on Knowledge Engineering and Ontology Development*, Vilamoura, Portugal, 19-22 September, pp. 192-198.

Basili, R., Melo, L. & Briand, C. 1996. A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Journal of Transactions on Software Engineering*, 22(10): 751–761.

Bechhofer, S, Horrocks, I, Stevens, R. 2001, OilEd: A Reason-able Ontology Editor for the Semantic Web. *In Proceedings of CEUR Workshop*, August 1-3, Stanford, CA, USA.

Berners-Lee, T., Hendler, J. & Lassila, O. 2001. The Semantic Web. *The Scientific American Online Article*. Available at https://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf Accessed 10th March 2018.

Bilski, A. 2011. A Review of Artificial Intelligence Algorithms in Document Classification, *International Journal of Electronics and Telecommunications.* 57(3): 263–270.

Bonino, D. 2005. Architectures and Algorithms for Intelligent Web Applications. *PhD Thesis*, Politacnico di Torino. Available at: https://pdfs.semanticscholar.org/170a/69373072485bd80d925e3e5eed33100bdb7c.pdf, Accessed 13/10/2017.

Borgida A. & Patel-Schneider, P. 1994. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic, *International Journal of Artificial Intelligence Research*. 1(2):277-308

Brewster, C. & O'hara, K. 2007. Knowledge Representation with Ontologies: Present Challenges-Future Possibilities. *International Journal of Human Computer* Studies, 65(7): 563–568.

Brickley D. and Guha R. (1999) Resource Description Framework (RDF) Schema Specification. *W3C Proposed Recommendation*. Available from: http://www.w3.org/TR/PR-rdf-schema Accessed on 06/08/2019.

Brownlee, J. 2018. How and When to Use ROC Curves and Precision-Recall Curves for Classification in Python, Available from https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/ Accessed 16 May 2019.

Burton-Jones, A., Storey, V., Sugumaran, V. & Ahluwalia, P. 2005. A Semiotic Metrics Suite for Assessing the Quality of Ontologies. *International Journal of Data Knowledge Engineering*, 55(1): 84–102.

Caruana, R. & Niculescu-Mizil, A. 2004. Data mining in metric space: an empirical analysis of supervised learning performance criteria, *in Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04),* New York, NY, USA, pp. 69-78.

Castrounis, A. 2016. Machine Learning: An In-Depth Guide—Data Selection, Preparation, and Modelling. Retrieved From: https://medium.com/@innoarchitech/machine-learning-an-in-depth-non-technical-guide-pt-2-d13b249007d8 Accessed 21st March, 2019.

Clare, A. & King, R. D. 2001. Knowledge Discovery in Multi-label Phenotype Data, *In Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, Freiburg, Germany, 3-5 September. 2168(1):42-53.

Connolly, D., Harmelen, F., Horrocks, I., Mcguiness, D., Patel-Schneider, P. & Stein, L. 2001. DAML + OIL *Reference Description*. Available at http://www.w3.org/TR/daml+oil-reference Accessed 12th March 2018

Cruz, A. and Ochimizu, K. 2009. Towards Logistic Regression models for predicting fault-prone code across software projects. In Proceedings of 3rd International Symposium on Empirical Software Engineering and Measurement. Pp. 460-463, Lake Buena Vista, Florida, USA.

Cutler, A., Stevens, J. & Cutler, D. 2011. Ensemble Machine Learning; Random Forests," *Springer*; pp. 157 – 176.

Dash, M. & Liu, H., 1997. Feature Selection for Classification. *International Journal of Intelligent Data Analysis*. 1 (1-4): 131-156

Dean M., Connolly D., Van-Harmelen F., Hendler J., Horrocks I., McGuinness D.L., Patel-Schneider P.F., and Stein L.A. 2003. OWL Web Ontology Language 1.0 Reference. W3C Working Draft, Available from https://www.w3.org/TR/owl-ref/ Accessed on 02/08/2019.

Delgado-Gomez, D., Laria, J. & Ruiz-Hernandez, D. 2018. Computerized adaptive test and decision trees: A unifying approach. *Journal of Expert Systems with Applications*, 117: 358-366

Denoeux, T. 1995. A K-Nearest Neighbor Classification Rule Based on Dempster-Shafer Theory," *International Journal on Transactions on Systems, Man and Cybernetics*. 25(5):804-813.

Destercke, S. 2012. A K-nearest neighbors method based on imprecise probabilities. *International Journal of Soft Computing*. 16: 833-844.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V.C. & Sachs, J. 2004. Swoogle: A semantic web search and metadata engine. *In Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*. Washington, DC, USA, November 8-13, pp. 652-659.

Domingue J. 1998. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. *In Proceedings of the 11th Knowledge Acquisition, Modelling and Management Workshop, (KAW'98),* 18-23, April, Banff, Alberta, Canada.

Duda, R.O., Hart, P. E., & Stork, D. G. 2001. Pattern Classification, 2nd ed, John Wiley & Sons Inc., New York, pp. 394–434.

Fábio, K., Christian, P., Joselyto R., Márcio, M., Karina, V., Leliane, N. & Renata, W. 2006. Classifying Ontologies. *In Proceedings of International Workshop on Ontologies and Their Applications*, Ribeirao Preto, Brazil, 27 October, pp. 248-251.

Farquhar A., Fikes R. and Rice J. 1996. The Ontolingua Server: A Tool for Collaborative Ontology Construction. *International Journal of Human-Computer Studies.* 46(6): 707-727.

Fernández M. 1999. Overview of Methodologies for Building Ontologies. *In Proceedings of the IJCAI'99 Workshop on Ontologies and Problem Solving Methods (PSMs).* Stockholm, Sweden, 2 August, pp. 1-13.

Fletcher, T. 2009. Support Vector Machines Explained. Available at https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf Accessed 20/03/2019

Fonou-Dombeu, J.V & Viriri, S. 2019. OntoMetrics Evaluation of Quality of e-Government Ontologies. *In Proceedings of the 8th International Conference on Electronic Government and the Information Systems Perspective,* Linz, Austria, August 26–29.

Funke, M. 2019. Prepare your Data management Architecture for Machine Learning at THINK. Retrieved From: https://www.ibmbigdatahub.com/blog/prepare-your-data-management-architecture-machine-learning-think Accessed 10th April 2019.

Gangemi, A., Catenacci, C., Ciaramita, M. & Lehmann, J. 2006. Modelling Ontology Evaluation and Validation, *In Proceedings of the 3rd European Semantic Web Conference on the Semantic Web: Research and Application*, Budva, Montenegro, 11-14 June, pp. 140–154.

Gangemi, A., Catenacci, C., Ciaramita, M., and Lehmann, J. 2005. A theoretical framework for ontology evaluation and validation. In Proceedings of 2nd Semantic Web Applications and Perspectives, Trento, Italy, 14-16 December, pp. 1-17.

Gavrilova, T., Gorovoy, V., & Bolotnikova, E. 2012. New Ergonomic Metrics for Educational Ontology Design and Evaluation. *In New Trends in Software Methodologies, Tools and Techniques (SoMeT),* 246(361–378).

Gerber, A., Barnard, A. & Merwe, A. 2008. A Functional Semantic Web Architecture. In Proceedings of 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, pp. 273-287.

Gómez-Pérez A. & Rojas-Amaya M. 1999. Ontological Reengineering for Reuse. *In the Proceedings of the 11th European Workshop on Knowledge Acquisition, Modelling and Management.* Dagstuhl Castle, Germany, may, 26 - 29pp. 157-171

Gruber, T. 1993. A Translation Approach to Portable Ontology Specifications. *International Journal of Knowledge Acquisition*, 5(2): 199-220.

Gruber, T. 1993. A Translation Approach to Portable Ontology Specifications. *International Journal of Knowledge Acquisition*, 5(2): 199-220.

Gruber, T. 1995. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human and Computer Studies*, 43(6): 907–928, 1995.

Guyon, I., Safiari, A., Dror, G. & Buhmann, J. 2006. Performance prediction challenge. *In Proceedings of IEEE/INNS conference IJCNN 2006*, Vancouver, Canada, July 16-21.

Haase, P., Broekstra, J., Eberhart, A. & Volz, R. 2004 A Comparison of RDF Query Languages", Proceedings of the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, 7-11 November, pp. 503-517

Hall, M. & Smith, L. 1999. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. *In Proceedings of the 12th International Florida Artificial Intelligence Research Society Conference*, May 1-5. Orlando, Florida, USA. pp. 235-239.

Han, J., Kamber, M. & Pei, J. 2012, Data Mining: concepts and techniques, 3rd ed. Morgan Kaufmann, Waltham.

Hang, H., Stojkovic, I. & Obradovic, Z. 2016. A robust data scaling algorithm to improve classification accuracies in biomedical data. I*nternational Journal of BMC Bioinformatics* Retrieved From https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-016-1236-x Accessed 03rd April, 2019.

Harrison, O. 2018. Machine Learning Basics with the K-Nearest Neighbors Algorithm. Available from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 Accessed May, 15, 2019.

Horn, V. and Balbinot, A. 2015. Upper-Limb Movement Classification Through Logistic Regression sEMG Signal Processing. In: Latin America Congress Intelligence (LACCI). pp. 1-5, Curitiba, Brazil.

Horrocks, I. 2002. DAML + OIL: A description logic for the semantic web, *International Journal of IEEE Data Engineering*, 25(1):4–9.

Horrocks, I. 2008. Ontologies and the Semantic Web. *Journal of Communications of the ACM-Surviving the data deluge*, 51(12): 58.

Hossin, M. & Sulaiman, M. 2015. A Review of Evaluation Metrics for Data Classification Evaluations, *International Journal of Data Mining & Knowledge Management (IJDKP)* 5(2):1-11

Hunt, E., Marin, J. & Stone, P.J. 1996, Experiments in Induction. Academic Press, New York.

Indra, S. T., Wikarsa, L. and Turang, R, R. 2016. Using Logistic Regression method to classify tweets into the selected topics. In Proceedings of International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 385-390, Malang, Indonesia.

Jadon, E. and Sharma, R. 2017. Data Mining: Document Classification using Naive Bayes Classifier, *International Journal of Computer Applications (0975 - 8887)*, 167(6): 13-16.

James, G., Witten, D., Hastie, T. & Tibshirani, R. 2009. An Introduction to Statistical Learning New York: Springer, pp. 238-376.

Jose, F., García, J. & Therón, R. 2011. Analysis of the OWL ontologies: A survey. *International Journal of Scientific Research and Essays*. 6(20): 4318-4329.

Kang, D., Xu, B., Lu, J. & Chu, W. 2004. A Complexity Measure for Ontology Based on UML. *In Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, Washington, DC, USA, 28 May, pp. 222–228.

Kazadi, Y.K. & Fonou-Dombeu, J. 2017. Analysis of Advanced Complexity Metrics of Biomedical Ontologies in the Bio-portal Repository, *International Journal of Bioscience, Biochemistry and Bioinformatics*, 7(1):20-32.

Khan, A. Baharudin, B., Lee, H.L. & Khan, K. 2010. A Review of Machine Learning Algorithms for Text-Documents Classification, *Journal of Advances in Information Technology,* 1(1): 4-20.

Khani, M., Naji, R. & Malakooti, M. 2012. A New Method for Conceptual Classification of Multi-label Texts in Web Mining Based on Ontology. *In Proceedings of 1st International Joint Conference on Advances in Signal Processing and Information Technology*, Amsterdam, The Netherlands, 1-2 December, pp. 43–48.

Kharya, S. and Soni, S. 2016. Weighted Naive Bayes Classifier: A Predictive Model for Breast Cancer Detection, *International Journal of Computer Applications (0975 – 8887)*, 133(9):32-37.

Kini, M., Devi, S., Desai, P. and Chiplunkar, N. 2015. Text Mining Approach to Classify Technical Research Documents using Naïve Bayes, *International Journal of Advanced Research in Computer and Communication Engineering*, 4(7):386-391.

Kira, K. & Rendell, L.A., 1992. The feature selection problem: Traditional methods and a new algorithm. *In Proceedings of 9th National Conference on Artificial Intelligence*, July 12 – 16, San Jose, California, pp. 129–134.

Klein, M., Fensel, D., Harmelen, F. & Horrocks, I. 2000. The relation between ontologies and schema-languages: Translating OIL-specifications in XML Schema. *In proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI'00,* Berlin, Germany, 20-25 August, pp. 3-12

Kohavi, R. & Sommerfield, D., 1995. Feature subset selection using the wrapper method: Over fitting and dynamic search space topology. *In Proceedings of First International Conference on Knowledge Discovery and Data Mining,* August 20 – 21, Montréal, Québec, Canada. Pp. 192–197.

Kwuimi, R. & Fonou-Dombeu, J. 2015. Storing and Querying Ontologies in Relational Databases: An Empirical Evaluation of Performance of Database-Based Ontology Stores. *In the Proceedings of the 9th International Conference on Advances in Semantic Processing*, Nice, France, 19-24 July, pp. 1-7.

Lantow, B. 2016. OntoMetrics: Putting Metrics into Use for Ontology Evaluation. *In Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016),* Porto, Portugal, 9-11, November. pp. 186-191

Larose, D. 2005. Discovering Knowledge in Data: An Introduction to Data mining. Central Connecticut State University Press

Lassila O. & Swick R. 1999. Resource Description Framework (RDF). Model and Syntax Specification. *W3C Recommendations*. Available from: http://www.w3.org/TR/PR-rdf-syntax Accessed on 10/08/2019.

Lee, K., Palsetia, D., Narayanan, R., Patwary, M., Agrawal, A. & Choudhary, A. 2011. Twitter trending topic classification, *In Proceedings of the 11th IEEE International Conference on Data Mining Workshops (ICDMW)*, 11 December, Vancouver, Canada, pp. 251-258.

Lerner, B., Guterman, H., Aladjem, M. & Dinstein, I. 1999. A comparative study of neural network-based feature extraction paradigms. *International Journal of Pattern Recognition Letters* 20:7-14

Li, C. & Park, S. 2009. Combination of Modified BPNN Algorithms and an Efficient Feature Selection Method for Text Categorization, *International Journal of Information Processing and Management*, 45(3): 329–340.

Li, L. & Horrocks I. 2004. A Software Framework for Matchmaking Based on Semantic Web Technology, *International Journal of Electronic Commerce*, 8(4):39-60.

Li, X., Sweigart, J., Teng, J., Donohue, J., & Thombs, L. 2001. A Dynamic Programming Based Pruning Method for Decision Trees. *Informs Journal on Computing* 13(4): 332–344.

Lowe, B, 2015, The Random Forest Algorithm with Application to Multispectral Image Analysis, *Master's Thesis.* The University of Texas at Tyler.

Lozano-Tello, A. & Gómez-Pérez, A. 2004. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*. 2 (15):1-18.

Luquea, A., Carrasco, A., Martín A. & Heras, A. 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix, *International Journal of Pattern Recognition.* 91(1):216-231.

Lyhyaoui A, Ynez, M. & Mora, M. 1999. Sample selection via clustering to construct support vector-like classifiers. *International Journal of IEEE Trans on Neural Networks*, 10 (6): 1474-1480

Magkanaraki, A., Karvounarakis, G., Tuan Anh, T., Christophides, V. & Plexousakis, D. 2002. Ontology Storage and Querying. *Technical Report on Ontologies and Information Systems*, Special Issue 308: 7-13.

Mai, X., Liao, Z. & Couillet, R. 2019. A large-scale analysis of logistic regression: Asymptotic performance and new insights. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). pp. 3357-3361, Brighton, UK.

Markov, A. & Last, M. 2005. A Simple, Structure- Sensitive Approach for Web Document Classification. *In the proceedings of the 3rd International Atlantic Web Intelligence Conference*, Lodz, Poland, 6-9 June, pp. 293–298.

Maynard, D., Peters, W. & Li, Y. 2006. Metrics for Evaluation of Ontology-Based Information Extraction. *In Proceedings of International Workshop on" Evaluation of Ontologies for the Web"*, Edinburgh, Scotland, 22 May, pp. 1045-1049.

Miao, D., Duan, Q., Zhang, H. & Jiao, N. 2009. Rough Set Based Hybrid Algorithm for Text Classification, *International journal of Expert Systems with Applications*, 36(5):9168-9174

Molder. C. 2004. Feature Extraction for Classification: A Survey 1. Linear Methods. Retrieved from: https://www.researchgate.net/publication/272148872_Feature_Extraction_for_Classification_A_Survey_I_Linear_Methods. Accessed on 16th March 2019.

Mone, L, 2019. An Enterprise Architect's Guide to machine Learning Series: Part 1. Available at https://blog.leanix.net/en/an-enterprise-architects-guide-to-machine-learning-series-part-1 Accessed 28/07/2019.

Myles, J. & Hand, D. 1990. The multi-class metric problem in nearest neighbor discrimination rules," *International Journal on Pattern Recognition*, 23(4):1291–1297.

Narzary, A. & Nandi, G. 2014. A Study on Semantic Web Languages and Technologies. *International Journal of Engineering and Computer Science*, 3(11): 9129-9132.

Nazrul, S. 2018. Receiver Operating Characteristic Curves Demystified (in Python), Available from https://towardsdatascience.com/receiver-operating-characteristic-curves-demystified-in-python-bd531a4364d0 Accessed 15 May 2019.

Netzer Y., Gabay D., Adler M., Goldberg Y., Elhadad M. 2009 Ontology Evaluation through Text Classification. *Lecture Notes in Computer Science*, 5731: 210–221.

Niusha, S., Lee, H., Jajkumar, R., Kallimani, V., Nik, A. & Isa, D. 2016 Using Unsupervised Clustering Approach to Train the Support Vector Machine for Text Classification, *International Journal on Expert System with Applications*, 211(3): 4-10

Noy, N., Sintek, M., Decker, S. & Crubezy, M. 2001. Creating Semantic Web Contents with Protege-2000. *IEEE Intelligent Systems* 16(2):60-71.

Pal, M. & Mather, P.M. 2001. Decision Tree-Based Classification of Remotely Sensed Data. *In Proceedings of the 22nd Asian Conference on Remote Sensing*. Singapore. pp. 1-4.

Pandey, P. 2018. Bringing the best out of Jupyter Notebooks for Data Science. Retrieved From https://towardsdatascience.com/bringing-the-best-out-of-jupyter-notebooks-for-data-science-f0871519ca29 Accessed 16th March 2019

Patil A. S. and Pawar, B. V. 2012. Automated Classification of Web Sites using Naive Bayesian Algorithm, *In Proceedings of the International Multiconference of Engineers and Computer Science*, 14-16 March, Hong Kong, pp. 1-5.

Permana, F. C., Rosmansyah, Y. and Abdullah, A. S. 2016. Naive Bayes as opinion classifier to evaluate student's satisfaction based on student sentiment in Twitter Social Media, *The Asian Mathematical Conference (AMC 2016)*, 25-29, July, Bali, Indonesia, pp. 1-10.

Potamias, M., Bonchi, F., Gionis, A. & Kollios, G. 2010. K-Nearest Neighbors in Uncertain Graphs. Journal of the VLDB Endowment. 3 (1): 997-1008.

Poveda-Villal´On, M., Su´Arez-Figueroa, M.C., G´omez-P´erez, A. 2012. Validating ontologies with oops! *In Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management*. Galway, Ireland, October, 8 - 12, pp. 267 – 281

Ramanujam, S., Gupta, A., Khan, L., Seida, S. & Thuraising, B., 2009. R2D: A Bridge between the Semantic Web and Relational Visualization Tools. *In the Proceedings of the third IEEE international Conference on Semantic computing (ICSC),* Berkeley, CA, USA, 14-16 September, pp. 303-311.

Rao, P. and Manikandan, J. 2016. Design and evaluation of logistic regression model for pattern recognition systems. In Proceeding of IEEE Annual India Conference (INDICON). pp. 1-6, Bangalore, India.

Rastogi, R. & Shim, K. 1998. Public: A decision tree classifier that integrates building and pruning. *In the Proceedings of the 24th International Conference on Very Large Databases (VLDB'98)*, 24-27 August, San Francisco, USA. pp. 405–415.

Russek, E., Kronmal, R.A., & Fisher, L.D. 1983. The effect of assuming independence in applying Bayes' theorem to risk estimation and classification in diagnosis. *The Journal of Computers and Biomedical Research*, 16:537–552.

Saaty, T. 1990. How to Make a Decision: The Analytic Hierarchy Process. *European Journal of Operational Research,* 48:(9-26).

Sapp, C. 2017. Preparing and Architecting for Machine Learning. Retrieved From: https://www.gartner.com/doc/3573617/preparing-architecting-machine-learning Accessed 10th April, 2019

Schaffer, C. 1994. Cross-validation, stacking and bi-level stacking: Meta-methods for Classification learning. In Selecting Models from Data: Lecture Notes in Statistics, 89(5): 51 – 59.

Schroder, M., Kruger, F. & Spors, S. 2019. Reproducible Research is more than Publishing Research Artefacts: A Systematic Analysis of Jupyter Notebooks from Research Articles, Available from https://arxiv.org/abs/1905.00092 Accessed 24 September 2019.

Segal, M., & Xiao, Y., 2011. Multivariate Random Forests. Wiley Interdisciplinary Reviews: *Journal of Data Mining and Knowledge Discovery* 1(1) pp. 80-87,

Shein, K. P. P. and Nyunt, T. T. S. 2010. Sentiment Classification based Ontology and SVM Classifier. *In Proceedings of 2nd International Conference on Communication Software and Networks,* 26-28 February, Singapore, pp. 169-172.

Shubham, S., Dixit, S. Kumar, S. and Kumar, P. 2018. Classification of tweets into various categories using classification methods, *International Journal of Advance Research, Ideas and Innovations in Technology,* 4(3): 937 – 939.

Smola, A. J. & Scholkopf, B. 2004. A Tutorial on Support Vector Regression, *Journal of Statistics and Computing*, 14(1):199–222.

Song, W., Spencer, B. & Du, W. 2011. Hybrid Reasoning for Ontology Classification, Advances in Artificial Intelligence, *In Proceeding of 24th Canadian Conference on Artificial Intelligence*, St. John's, NF, Canada, 25-27 May, pp. 372-376.

Sridevi, K. & Umarani, R. 2013. Ontology Ranking Algorithms on Semantic Web: A Review, *International Journal of Advanced Research in Computer and Communication Engineering*, 2(9): 3471-3476

Srikant, R., Vu, Q. & Agrawal, R. 1997. Mining association rules with item constraints. In the Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97), Menlo Park, CA, USA. pp. 67–73.

Srinivasulu, S., Sakthivel, P. & Balamurugan, E. 2014. Measuring the Ontology Level and Class Level Complexity Metrics in the Semantic Web. *International Journal of Advanced Computational Engineering and Networking*, 2(5): 68–74.

Staab S. & Maedche A. 2000. Ontology Engineering Beyond the Modelling of Concepts and Relations. *Proceedings of the ECAI'00 Workshop on Applications of Ontologies and PSMs. Berlin.*

Stevens R, Goble C. & Bechhofer S. 2000. Ontology-based Knowledge Representation for Bioinformatics. *Briefings in Bioinformatics*. 1(4): 398-414.

Sun, S. & Houng, R. 2010. An Adaptive k-Nearest Neighbor Algorithm. *In Proceedings of 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010)*, 10 – 12 August, Yantai, China. pp. 91-94.

Swartout B., Patil R., Knight K. & Russ T. 1997. Towards Distributed Use of Large-Scale Ontologies. *AAAI'97 Spring Symposium Series on Ontological Engineering*. 138-148.

Takizawa, H. & Nakajima, T. 2000. An active learning algorithm based on existing training data, *International Journal of IEICE Transactions on Information and Systems* 83(1):91-99.

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., & Aleman-Meza, B. (2005). OntoQA: Metric-Based Ontology Quality Analysis. *Proceedings of IEEE ICDM Workshop on*

Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources, Houston, TX, 27, November, pp. 45 – 53

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., & Aleman-Meza, B. 2005. OntoQA: Metricbased ontology quality analysis. *In IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources,* Houston Texas, USA, 27 November, pp. 1-10

Tartir, S., Arpinar, I., Moore, M., Sheth, A. & Aleman-Meza, B. 2005. OntoQA: Metric-Based Ontology Quality Analysis. *In Proceedings of the Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources.* Houston, TX, USA, 27 November, pp. 407-416.

Taye, M. 2010. Understanding Semantic Web and Ontologies: Theory and Applications. *International Journal of Computing*, 2(6): 182–192.

Taye, M. M. 2010. Understanding Semantic web and ontologies: Theory and applications, *Journal of Computing*, 2(6):182-192.

Trajdos, P. & Kurzynski, M. 2017. Weighting scheme for a pairwise multi-label classifier based on the fuzzy confusion matrix, *International Journal of Pattern Recognition Letters,* 103(1):60-67.

Vandana, K. & Mahender, N. 2012. Text Classification and Classifiers: A Survey, *International Journal of Artificial Intelligence & Applications,* 3(2): 85-89

Verikas, A., Gelzinis, A. & Bacauskiene, M. 2011. Mining data with random forests: a survey and results of new tests. *Pattern Recognition*, 44(2): 330-349.

Vinayagam, A., Knig, R., Moormann, J., Schubert, F., Eils, R., Glatting, K. H. & Suhai, S. 2004. Applying Support Vector Machines for Gene ontology-based gene function prediction. *International Journal of BMC Bioinformatics*, 5(116):1-14.

Weil, D. & Xiang-Yang, L. 2010. Weighted naive Bayesian classifier model based on information gain, *In Proceedings of the International Conference on Intelligent System Design and Engineering Application,* 13-14 October, Changsha, Huan, China, pp. 819-822.

Yang, L., Xu, D., Li, C. & Tian, W. 2010. An application of logistic regression in cerebral infraction disease detection based on association rules with pre-rough classifier. In Proceedings of 17th International Conference on Industrial Engineering and Engineering Management. pp. 304-306.

Yang, Z., Zhang, D. & Ye, C. 2006a. Ontology Analysis on Complexity and Evolution Based on Conceptual Model. *In Proceedings of the third International Workshop on Data Integration in the Life Sciences*, Hinxton, UK, 20-22 July, pp. 216-223.

Yao, H., Orme, A. & Etzkorn, L. 2005. Cohesion Metrics for Ontology Design and Application. *International Journal of IEEE Software*, 1(1):107-113

Zhang, D., Ye, C. & Yang, Z. 2006b. An Evaluation Method for Ontology Complexity Analysis in Ontology Evolution. *In Proceedings of International Conference on Knowledge Engineering and Knowledge Management,* Podebrady, Czech Republic, 2-6 October, pp. 214-221.

Zhang, H., Li, Y. & Tan, H. 2010. Measuring Design Complexity of Semantic Web Ontologies. *The Journal of Systems and Software*, 83(3):803-814.

Zhang, M. & Zhou, Z. 2005. A k-Nearest Neighbor Based Algorithm for Multi-label Classification, *In the Proceedings of IEEE International Conference on Granular Computing,* 25-27 July, Beijing, China, pp. 718-721.

Zhang, W. & Gao, F. 2011. An Improvement to Naive Bayes for Text Classification, *International Journal of Advanced in Control Engineering and Information Science*, 15(1):2160 – 2164.

Zhang, Z., Zhou, Z. & Shen, D. 2013. Sample Selection in Supervised Learning Based on Adaptive Estimated Threshold. *In Proceedings of the 2013 International Conference on Machine Learning and Cybernetics,* July 1. Tianjin, China, pp. 14-17.

Zhao, Y., Dong, J. & Peng, T. 2009. Ontology Classification for Semantic Web Based Software Engineering. *Journal of IIEE Transactions on Services Computing,* 2(4): 303–317.

Zhou, S. 2010. Relational Databases Access based on RDF View. *In proceedings of the First International Conference on E-Business and E-Government*, Guangzhou, China, 7-9 May, pp. 5486-5489.

Zhou, X., Li, W., Wu, S. and Wang, S. 2017. An Ontology-driven, SVM approach for Hyperspectral image Classification. *International Journal of Image and Data Fusion*, 8 (2): 1-18.

APPENDIX: Complexity Metrics of Biomedical Ontologies forming the dataset.

Index	noc	ar	ir	rr	Acr	er	a/cr	irr	c/rr	ap	cr	arc	alc	ad	md	ab	mb	anp	QR
O_1	93	0	1.0332	0.0204	0	0	9.8495	1	0.9490	0.0000	0	1	66	3.1546	5	3.4643	31	19.4000	2
O_2	410	0	1.3585	0.0559	0	0.03415	4.3659	0.05263	0.6949	0.0585	0	86	301	2.0929	4	4.0357	86	113.0000	3
O_3	1086	0.0313	1.0617	0.0368	0	0.0046	6.4715	0.025	0.9073	0.0000	0	11	829	5.2694	12	4.1724	109	100.8333	1
O_4	19	0	0.9474	0.0000	0	0	8.9474	0	1.0556	0.8421	0.10526	1	17	2.789474	3	6.3333	16	6.333333	4
O_5	533	0	0.9493	0.0000	0	0	8.4822	0	1.05336	0.0000	0	27	483	2.2439	3	10.4510	27	177.6667	3
O_6	14	0	0.0000	1.0000	0	0	2.7143	0	14.0000	0.0000	0	14	14	14.0000	1	14.0000	14	14.0000	2
O_7	636	0	0.9984	0.0031	0	0.00315	9.2736	0	0.9984	0.0000	0	1	496	5.1053	8	4.5106	53	79.5000	3
O_8	116	0	1.0000	0.0000	0	0	6.7759	0	1.0000	1.0948	0.87931	1	84	5.3	8	3.5294	37	15.0000	4
O_9	295	0.0847	1.2780	0.0981	0	0.00678	4.8949	0.30435	0.7057	0.0000	0	3	212	4.7257	8	3.7895	16	36.0000	1
O_{10}	12	1	0.8333	0.5455	0	0.33333	77.8333	0.5	0.5455	13.4167	0.58333	2	7	2.5833	4	2.0	3	3.0000	4
O_{11}	245	0	2.9714	0.0055	0	0	12.5265	0	0.3347	0.0000	0	3	243	1.9883	2	85.3333	242	128.0000	2
O_{12}	466	0.0064	6.0193	0.0046	0	0	13.7060	0	0.1654	1.2339	0.01502	26	447	1.9756	2	53.2000	119	532.0000	4
O_{13}	414	0.5242	1.5048	0.2773	0	0.09662	11.9300	0.26761	0.4803	0.0338	0.00725	16	298	5.2024	11	3.5714	35	38.6364	4
O_{14}	1712	0.0035	1.1454	0.5090	0	1.17056	7.1507	0.03448	0.4286	0.0403	0.00935	1	1328	7.2350	14	4.8115	74	155.0000	4
O_{15}	17	0.7059	3.7059	0.1923	0	0	16.2353	0.46667	0.2179	0.0000	0	6	9	2.6296	4	2.0769	6	6.7500	1
O_{16}	35	0	0.9714	0.3462	0	0	16.4286	0	0.6731	0.0000	0	1	21	4.3143	6	2.3333	4	5.8333	2
O_{17}	118	0.0254	1.1017	0.5357	0	0.13559	5.1017	0.375	0.4214	0.0000	0	17	80	5.1422	8	3.4590	17	26.3750	1
O_{18}	41	0	0.4146	0.4516	0	0	209.0732	0.5	1.3226	11.5854	0.17073	24	24	1.4146	2	2.2778	24	20.5000	4
O_{19}	295	0	1.9559	0.0103	0	0.00339	3.9220	0	0.5060	0.0000	0	9	218	3.5014	5	4.5309	35	73.4000	2
O_{20}	80	0	1.7750	0.2565	0	0.0375	25.0000	0.23913	0.4188	0.0000	0	2	42	5.7875	10	2.0513	15	8.0000	2
O_{21}	2154	0	1.2498	0.0633	0	0.05618	9.5399	0	0.7495	0.0000	0	59	1058	6.5636	10	2.0779	59	707.3000	3

O_{22}	4530	0	1.0987	0.0000	0	0	5.3024	0	0.9102	0.0000	0	7	3626	5.2839	10	5.2785	80	606.5000	3
O_{23}	113	0	2.6903	0.0098	0	0	13.1062	0	0.3681	0.0000	0	5	112	1.9558	2	56.5000	108	56.5000	2
O_{24}	5	0	0.8000	0.4286	0	0	5.2000	0.33333	0.7143	0.0000	0	1	3	2.0	3	1.6667	3	1.6667	1
O_{25}	501	0	0.9860	0.0000	0	0	1.0000	0	1.0142	0.0000	0	7	494	1.9860	2	62.6250	170	250.5000	3
O_{26}	148	0.0473	1.3986	0.1375	0	0.00676	4.8108	0	0.6167	0.0135	0.01351	5	81	4.4298	6	3.3529	15	19.0000	4
O_{27}	134	0.2239	1.8582	0.4196	0	0.15672	7.4030	0.33824	0.3124	0.0597	0.07463	6	18	2.0000	4	3.8333	6	5.7500	4
O_{28}	121	0	0.9917	0.0000	0	0	5.0826	0	1.0083	0.0000	0	1	87	4.6281	6	3.4571	9	20.1667	2
O_{29}	32	0.3125	1.6563	0.3205	0	0.25	9.2500	0	0.4103	0.4063	0.125	6	11	1.5000	2	6.0000	6	6.0000	4
O_{30}	37	0	0.9730	0.0000	0	0	2.9730	0	1.0278	0.0000	0	1	31	2.8108	3	5.2857	11	12.3333	2
O_{31}	155	0	1.0645	0.0179	0	0	4.2774	0.33333	0.9226	0.0000	0	1	95	5.0934	8	2.7164	15	22.7500	1
O_{32}	399	0.0025	1.2757	0.1148	0	0.08521	4.0226	0.06061	0.6939	0.1153	0.05263	21	262	4.7884	8	3.1844	25	56.1250	4
O_{33}	243	0	1.2469	0.6513	0	0.11523	25.5761	0.18251	0.2796	0.0823	0.01646	2	100	9.3420	37	1.7953	9	8.2973	4
O_{34}	131	0	1.0305	0.0000	0	0	7.1374	0	0.9704	0.0000	0	6	102	3.3688	7	4.3243	18	22.8571	2
O_{35}	117	0.1026	1.1453	0.3300	0	0.13675	7.6239	0.35088	0.5850	0.1453	0.08547	36	74	2.5364	5	2.7455	36	30.2000	4
O_{36}	2	5.5	0.0000	1.0000	0	0	146.0000	0.06897	0.0690	4.5000	1	2	2	1.0000	2	2.0000	2	2.0000	2
O_{37}	576	0	1.0538	0.0000	0	0	3.6910	0	0.9489	0.0000	0	1	545	3.6003	6	19.0000	240	101.3333	3
O_{38}	104	0.7404	1.1250	0.6366	0	0	10.2404	0.13158	0.3230	0.0000	0	2	78	4.6827	9	3.8519	22	11.5556	2
O_{39}	300	0.05	0.9067	0.2381	0	0.03	8.0867	0	0.8403	0.7033	0.14	32	220	3.2609	7	3.7375	32	42.7143	4
O_{40}	2270	0.0026	1.7762	0.0771	0	0.09912	8.7533	0.2549	0.5196	0.0203	0.00308	1	1418	12.1887	20	2.5172	129	306.6000	4
O_{41}	265	0.0679	1.3132	0.2073	0	0.15849	4.9472	0.41667	0.6036	0.0000	0	3	214	5.2325	7	5.2115	23	38.7143	1
O_{42}	2098	0	1.0381	0.0000	0	0	27.0119	0	0.9633	0.0000	0	2	1462	7.0782	12	3.3581	35	199.2500	3
O_{43}	246	0.065	0.9919	0.2673	0	0	8.1057	0	0.7387	0.0000	0	2	180	3.7439	6	3.6716	14	41.0000	1
O_{44}	322	0.1087	0.9969	0.0000	0	0	20.7547	0	1.0031	0.0000	0	1	255	4.7391	5	4.7353	29	64.4000	2

O_{45}	1388	0.0029	1.3429	0.0661	0	0.0353	4.9820	0.31818	0.6954	0.0000	0	11	831	11.1169	21	2.7370	80	75.3333	1
O_{46}	243	0	0.9835	0.0245	0	0	7.6543	0	0.9918	0.0000	0	7	130	3.3951	7	2.1316	20	34.7143	2
O_{47}	243	0.3909	0.9588	0.0934	0	0.04938	4.3663	0	0.9455	0.0000	0	10	171	2.4495	4	7.0714	29	49.5000	1
O_{48}	1056	0.0009	1.4612	0.0134	0	0.00663	4.3987	0.07143	0.6752	0.0000	0	3	937	6.0928	9	8.8000	360	117.3333	1
O_{49}	308	0	1.6071	0.0120	0	0	21.6526	0	0.6148	0.0000	0	8	34	2.4727	4	3.4375	8	13.7500	2
O_{50}	289	0.0761	0.9965	0.0619	0	0	17.3391	0	0.9414	0.7128	0.43599	1	214	8.6747	13	3.8026	22	22.2308	4
O_{51}	1018	0.0069	0.9912	0.0059	0	0	10.7318	0	1.0030	0.9322	0.00098	9	789	2.6699	6	4.4261	429	169.6667	4
O_{52}	27	0.1852	1.0741	0.3556	0	0.07407	5.2593	0.14286	0.6000	0.0000	0	15	17	2.1176	4	2.8333	15	8.5000	2
O_{53}	280	0.1107	0.9964	0.2828	0	0	3.9250	0	0.7198	0.0750	0.01429	1	228	3.8036	7	5.2830	19	40.0000	4
O_{54}	3359	0	2.7681	0.0425	0	0.08961	18.9443	0.10714	0.3459	0.0030	0.0003	1	1716	15.7023	27	2.2093	165	1068.0741	4
O_{55}	227	0	0.9648	0.0000	0	0	1.0000	0	1.0365	0.0000	0	8	219	1.9648	2	25.2222	57	113.5000	2
O_{56}	448	0	0.9888	0.0000	0	0	1.0000	0	1.0113	0.0000	0	5	443	1.9888	2	74.6667	155	224.0000	3
O_{57}	250	0	1.4640	0.0292	0	0	7.6800	0	0.6631	0.0000	0	3	157	5.9340	12	2.8235	13	24.0000	2
O_{58}	483	0	0.8054	0.1432	0	0.03727	7.2919	0.02174	1.0639	0.0062	0.00414	119	262	2.2451	6	3.9444	119	59.1667	4
O_{59}	230	0	0.9217	0.0320	0	0.01739	3.3435	0	1.0502	0.0043	0.00435	21	183	3.2251	7	4.8125	25	33.0000	4
O_{60}	10031	0.0001	1.1931	0.0155	0	0.01117	8.0726	0.06667	0.8251	0.0000	0	64	7369	11.8479	18	3.8664	115	738.0556	1
O_{61}	431	0	1.2320	0.0797	0	0	11.3341	0.32609	0.7470	0.0000	0	1	287	6.2082	12	3.1534	26	42.8333	2
O_{62}	6623	0	1.5556	0.0006	0	0	7.0314	0	0.6424	0.0000	0	2	4903	7.4353	16	3.8317	258	495.2500	3
O_{63}	388	0	1.0773	0.0302	0	0	12.9716	0.23077	0.9002	0.0000	0	1	256	5.1158	8	3.0870	26	52.8750	3
O_{64}	124	0	1.1290	0.1463	0	0	3.0806	0	0.7561	0.0000	0	37	102	3.1969	6	5.7727	37	21.1667	2
O_{65}	16	0	0.8750	0.0667	0	0	2.1875	0	1.0667	0.0000	0	2	13	2.6250	3	4.0000	12	5.3333	3
O_{66}	280	0	1.2500	0.0000	0	0	5.0036	0	0.8000	0.0000	0	1	230	3.0123	7	4.6782	204	58.1429	2
O_{67}	224	0.0089	1.7634	0.1203	0	0.04018	5.1473	0.45455	0.4989	0.2634	0.07589	10	186	4.2731	6	5.8205	38	37.8333	4

O_{68}	92	0	1.6196	0.0132	0	0	9.0652	0	0.6093	0.0000	0	1	58	4.6644	7	4.1714	12	20.8571	2
O_{69}	1864	0.0005	1.4211	0.0275	0	0.01019	4.6786	0.05128	0.6843	0.0300	0.00215	176	1471	3.2610	7	5.9618	176	289.5714	4
O_{70}	114	0	0.9298	0.0000	0	0	1.0000	0	1.0755	0.0000	0	8	106	1.9298	2	12.6667	36	57.0000	3
O_{71}	149	0	1.0470	0.0877	0	0.05369	2.5101	0	0.8713	0.0000	0	20	93	3.3029	7	3.0702	20	25.0000	2
O_{72}	96	0	0.9479	0.0000	0	0	7.2604	0	1.0549	0.0000	0	5	67	3.1979	6	3.2000	14	16.0000	3
O_{73}	337	1.2997	3.0119	0.1563	0	0.04154	21.3561	0.01149	0.2801	2.5638	0.10089	16	98	2.2923	4	3.9394	34	32.5000	4
O_{74}	25	0.52	0.3600	0.7273	0	0	32.4000	0	0.7576	4.0000	0.08	16	20	1.4400	3	4.1667	16	8.3333	4
O_{75}	12	0.3333	0.0000	1.0000	0	0	8.2500	0	0.7059	0.4167	0.08333	12	12	1.0000	1	12.0000	12	12.0000	4
O_{76}	480	0	0.9813	0.0000	0	0	1.0000	0	1.0191	0.0000	0	9	471	1.9813	2	48.0000	123	240.0000	3
O_{77}	188	0.3351	0.6064	0.5366	0	0	11.6755	0.18182	0.7642	0.0000	0	75	169	1.6720	3	9.4500	75	63.0000	1
O_{78}	2082	0.0014	0.0000	0.0000	0	0	5.0029	0	0.0000	0.0000	0	2082	2082	1.0000	1	2082.0000	2082	2082.0000	3
O_{79}	736	0.0611	1.2432	0.6391	0	0.00544	10.5530	0.17143	0.2903	0.0000	0	14	488	4.5136	8	2.9558	34	92.0000	1
O_{80}	107	0	0.9626	0.3869	0	0.16822	371.5888	0	0.6369	0.0000	0	31	65	3.3706	8	2.6481	31	17.8750	2
O_{81}	1575	0	3.2248	0.0105	0	0	12.2489	0	0.3068	0.0076	0	1	1141	6.4848	12	3.8460	34	172.7500	3
O_{82}	813	0	1.6384	0.2653	0	0.55105	26.0074	0	0.4484	0.0000	0	7	325	6.9151	13	1.6485	22	216.4615	2
O_{83}	160	0	0.9813	0.2341	0	0.0375	7.9313	0	0.7805	0.0000	0	18	113	2.9600	5	3.9474	18	30.0000	2
O_{84}	46	0	0.9783	0.0000	0	0	2.9130	0	1.0222	0.0000	0	9	29	2.6901	5	2.2188	9	14.2000	3
O_{85}	295	0.0068	1.0407	0.1050	0	0	2.7966	0	0.8601	0.0000	0	6	220	3.4058	5	4.1184	23	62.6000	1
O_{86}	2274	0.0022	0.9897	0.1612	0	0.14565	7.1701	0.14583	0.8475	0.0451	0.01413	21	1199	8.6450	31	2.8692	25	59.4194	4
O_{87}	710	0.0014	2.7268	0.0593	0	0.04085	14.0761	0.22581	0.3450	0.1775	0.00704	11	144	3.1574	8	3.6481	18	24.6250	4
O_{88}	18	0	0.5714	0.1579	0	0	2.2143	0	1.4737	0.0000	0	14	16	1.6071	4	2.1538	14	7.0000	2
O_{89}	407	0	0.9803	0.0245	0	0	7.8919	0	0.9951	0.0000	0	8	355	3.0098	5	7.6792	76	81.4000	2
O_{90}	102	0.0098	1.1275	0.3072	0	0	8.5098	0	0.6145	0.0000	0	3	65	3.7059	7	2.6842	8	14.5714	1

O_{91}	414	0	0.9976	0.0000	0	0	5.8578	0	1.0024	0.0000	0	1	310	4.2072	9	3.9151	31	46.1111	3
O_{92}	483	0.0683	1.0207	0.0257	0	0	5.1553	0	0.9545	0.1077	0.0021	8	388	4.1282	8	5.1296	65	69.2500	4
O_{93}	253	0	1.0909	0.0417	0	0	11.9368	0.5	0.8785	0.0000	0	11	184	3.5176	6	3.9020	35	66.3333	2
O_{94}	375	0	4.7680	0.0022	0	0	15.6933	0	0.2093	0.0000	0	3	370	2.0053	3	62.5000	365	125.0000	3
O_{95}	125	0.48	0.9440	0.0992	0	0	4.4320	0	0.9542	0.2720	0.016	7	105	2.5680	5	5.9524	50	25.0000	4
O_{96}	3398	0.0024	1.5986	0.1447	0	0.2298	10.8808	0.2347	0.5350	0.0883	0.0147	1	2617	8.7522	18	4.4846	78	210.7778	4
O_{97}	279	0	0.9892	0.8292	0	0	71.1147	0.2542	0.1726	0.8602	0.0108	3	44	1.9348	2	15.3333	41	23.0000	4
O_{98}	3580	0.0025	0.9978	0.0000	0	0	11.2838	0	1.0022	0.0000	0	8	2362	13.4950	41	2.9368	66	87.3171	3
O_{99}	5221	0	2.1628	0.0212	0	0.0349	6.9213	0	0.4526	0.0142	0.0002	420	4315	4.5569	12	6.0586	420	430.6667	4
O_{100}	2948	0.0078	4.0468	0.0374	0	0.1174	19.1859	0.1102	0.2379	0.0302	0	1	1575	10.1197	18	2.4021	58	254.8889	1
O_{101}	1662	0.0096	1.2365	0.1034	0	0.0126	3.9284	0.0939	0.7251	0.0578	0.0072	1	1147	10.5305	17	3.8298	36	171.4118	4
O_{102}	390	0.0026	1.2051	0.3535	0	0.2564	5.9538	0.3537	0.5365	0.0000	0	64	202	3.4308	11	2.7895	64	28.9091	1
O_{103}	625	0	0.9056	0.0471	0	0.0368	6.5184	0	1.0522	0.0000	0	109	505	5.8481	10	5.1803	109	63.2000	3
O_{104}	779	0.0077	1.5340	0.0940	0	0.1053	9.8973	0.2087	0.5906	0.0321	0.0103	1	564	8.5381	12	3.7838	27	70.0000	4
O_{105}	23	0.0435	0.8696	0.1667	0	0	3.6087	0	0.9583	0.0000	0	8	12	2.3478	5	1.9167	8	4.6000	1
O_{106}	100	0	1.0400	0.0370	0	0	2.8100	0	0.9259	0.0000	0	1	66	5.4752	8	2.8857	7	12.6250	3
O_{107}	858	0	1.6970	0.2349	0	0.4242	10.0478	0.253	0.4509	0.0140	0.0023	2	624	10.2029	19	3.1953	153	64.5789	4
O_{108}	1251	0	1.1103	0.0746	0	0.044	5.8409	0.0769	0.8334	0.0000	0	8	1003	6.8268	12	5.0400	123	115.5000	3
O_{109}	127	0.0315	1.1654	0.1345	0	0.0236	9.1969	0.125	0.7427	0.0236	0.0079	23	97	3.6850	6	4.0968	23	21.1667	4
O_{110}	108	0	1.0000	0.0000	0	0	3.0556	0	1.0000	0.0000	0	1	76	3.7248	5	3.2647	10	21.8000	2
O_{111}	6567	0	2.0649	0.0065	0	0.0005	14.0641	0.2361	0.4811	0.0012	0	1	5338	12.7558	21	3.9140	266	361.9524	3
O_{112}	3214	0.0006	2.0737	0.0802	0	0.0364	10.4182	0.2244	0.4436	0.0078	0.0025	4	2539	9.6996	36	5.1438	441	104.3056	4
O_{113}	228	0.0132	1.9825	0.0642	0	0.0307	9.8289	0.375	0.4721	0.1491	0.0526	1	131	9.8333	35	2.3265	34	6.5143	4

O_{114}	4566	0.0007	1.4807	0.0111	0	0.012	8.6577	0	0.6678	0.0000	0	142	3517	7.0284	12	4.3893	142	381.5000	1
O_{115}	1555	0.0013	2.7008	0.0796	0	0.1428	15.4727	0.1319	0.3380	0.0141	0.0006	13	82	3.0500	7	2.3729	15	20.0000	4
O_{116}	1617	0.0025	1.2202	0.1278	0	0.1033	9.9270	0.3077	0.7149	0.1373	0.0192	1	1292	7.7411	17	4.8696	93	98.8235	4
O_{117}	2965	0	1.1133	0.0362	0	0.0331	9.1646	0	0.8657	0.0000	0	5	2022	7.0858	12	3.2143	65	266.2500	3
O_{118}	4427	0.0005	1.2268	0.0046	0	0	8.3183	0	0.8114	0.0000	0	8	3626	4.3160	10	5.5245	1044	754.1000	1
O_{119}	632	0	0.9842	0.5079	0	0.9968	6.4367	0	0.5000	0.0000	0	10	515	4.4557	8	5.3559	43	79.0000	3
O_{120}	2270	0.0026	1.7762	0.0771	0	0.0991	8.7533	0.2549	0.5196	0.0203	0.0031	1	1418	12.1887	20	2.5172	129	306.6000	4
O_{121}	2920	0.0017	1.0764	0.1308	0	0.099	8.0555	0.2529	0.8075	0.0051	0.0003	130	2168	8.3331	14	4.0825	187	215.5000	4
O_{122}	519	0	1.1503	0.4215	0	0.0619	8.9884	0.3846	0.5029	0.0385	0.0077	1	410	5.7458	11	4.7411	173	48.2727	4
O_{123}	1234	0.0016	1.2091	0.0681	0	0.0567	6.7723	0.225	0.7708	0.0332	0.0065	1	908	8.6035	17	3.7105	119	91.2353	4
O_{124}	655	0	1.0458	0.0015	0	0	6.8382	0	0.9548	0.0015	0	22	422	7.0303	11	2.8788	26	69.0909	3
O_{125}	128	0.0938	1.1172	0.5719	0	0.1719	9.2969	0.0455	0.3832	1.2891	0.4688	33	81	3.4688	8	2.6667	33	16.0000	4
O_{126}	455	0	0.7099	0.1739	0	0.0901	24.9429	0	1.1637	0.0022	0	152	327	1.6017	3	11.9667	152	119.6667	3
O_{127}	1544	0.0006	1.2429	0.1509	0	0.0317	8.1729	0.434	0.6832	0.0000	0	9	1032	7.3995	13	3.0205	118	124.0000	1
O_{128}	1565	0	1.4479	0.0535	0	0.046	5.3252	0.0833	0.6537	0.0000	0	7	1323	6.8258	12	7.1057	147	156.9167	3
O_{129}	840	0.0012	1.0679	0.0197	0	0.0131	5.5869	0	0.9118	0.4476	0.0048	57	651	3.8036	7	5.0567	69	203.7143	4
O_{130}	5475	0.0005	2.4504	0.0134	0	0	3.4853	0	0.4026	0.0000	0	4	4095	4.2990	10	4.7117	2155	519.7000	1
O_{131}	1596	0.0163	0.9981	0.1531	0	0.0006	16.8371	0.0508	0.8485	3.0119	0.0094	4	1313	6.6510	9	5.6218	20	343.5556	4
O_{132}	1307	0.0015	1.0803	0.1437	0	0.0145	6.1163	0.243	0.7926	0.0000	0	5	691	7.8355	12	3.4490	118	104.3333	1
O_{133}	1771	0.004	0.9938	0.0000	0	0	8.9842	0	1.0063	0.0000	0	11	1353	4.3473	9	4.2267	50	196.7778	3
O_{134}	1551	0	0.8124	0.0889	0	0.0393	5.5500	0	1.1215	0.0000	0	348	943	4.4464	11	4.4021	348	114.4545	3
O_{135}	2261	0	1.4534	0.2387	0	0.2834	4.6799	0	0.5238	0.0000	0	3	336	2.9974	4	7.3076	69	97.0000	2
O_{136}	22	4.9091	0.2727	0.8500	0	0	35.3182	0.0593	0.5500	0.0000	0	16	20	1.2727	2	7.3333	16	11.0000	1

O_{137}	399	0.0023	1.2757	0.1148	0	0.0852	4.0226	0.0606	0.6939	0.1153	0.0526	21	262	4.7884	8	3.1844	25	56.1250	4
O_{138}	374	0	0.9947	0.0700	0	0	9.4064	0	0.9350	0.0000	0	2	320	5.2112	8	6.8000	206	46.7500	2
O_{139}	162	0	0.9815	0.0063	0	0	8.8951	0	1.0125	0.0000	0	3	113	4.4630	8	3.2400	14	62.5000	2
O_{140}	143	0.1329	1.3147	0.1005	0	0	4.7413	0	0.6842	0.1888	0.02797	8	103	2.9580	5	3.4878	18	28.6000	4
O_{141}	308	0	1.1266	0.0086	0	0	5.9351	0	0.8800	0.0000	0	1	1	1.0000	1	1.0000	1	1.0000	1
O_{142}	525	0	2.1010	0.0063	0	0	7.0076	0	0.4730	0.0000	0	47	409	2.9854	5	14.9375	158	95.6000	3
O_{143}	567	0	1.5979	0.1100	0	0.00529	3.3139	0	0.5570	0.0000	0	6	393	6.1190	9	4.5559	40	144.7778	2
O_{144}	2265	0	1.4534	0.2387	0	0.28344	4.6799	0	0.5238	0.0000	0	3	336	2.9974	4	7.3208	69	97.0000	2
O_{145}	162	0.0062	0.9506	0.1492	0	0	4.9877	0.22222	0.8950	0.0000	0	41	70	2.5385	8	2.4375	41	14.6250	1
O_{146}	822	0	0.9903	0.0000	0	0	1.0000	0	1.0098	0.0000	0	8	814	1.9903	2	91.3333	324	411.0000	3
O_{147}	1812	0	0.9950	0.0050	0	0.0011	4.0364	0	1.0000	0.0000	0	11	1364	4.7674	10	4.0401	98	181.4000	3
O_{148}	442	0.0181	1.0068	0.0389	0	0	4.7557	0.04	0.9546	0.0000	0	4	297	5.1236	11	3.0530	36	41.9091	1
O_{149}	12	0	0.0000	0.0000	0	0	5.3333	0	0.0000	0.0000	0	12	12	1.0000	1	12.0000	12	12.0000	2
O_{150}	138	0.0362	0.9928	0.0352	0	0	5.3696	0	0.9718	0.0000	0	1	113	4.0435	6	5.3077	28	23.0000	1
O_{151}	408	0	1.2206	0.2709	0	0.27696	3.3652	0.05714	0.5974	0.0000	0	48	265	5.0119	10	3.8550	48	50.5000	2
O_{152}	199	0	1.1960	0.0518	0	0.00503	7.9196	0	0.7928	0.0151	0.0201	8	181	3.1910	4	10.4737	81	49.7500	4
O_{153}	3639	0	2.2473	0.0007	0	0	8.1434	0	0.4446	0.0000	0	6	3508	2.3776	8	27.5682	1467	454.8750	3
O_{154}	285	0	1.2877	0.0000	0	0	5.6386	0	0.7766	0.0000	0	4	207	5.3481	8	3.8966	17	84.7500	2
O_{155}	83	0	1.0120	0.0562	0	0	4.2771	0	0.9326	0.0000	0	4	59	4.0341	6	3.5200	8	14.6667	1
O_{156}	276	0.0217	1.1087	0.1429	0	0	2.5435	0.125	0.7731	0.0000	0	8	193	3.6737	7	3.3140	15	40.7143	1
O_{157}	46	0.2826	0.8261	0.4063	0	0	13.4565	0	0.7188	1.3696	0.15217	8	35	2.3478	4	3.8333	8	11.5000	4
O_{158}	1097	0	0.9927	0.0000	0	0	4.8560	0	1.0073	0.0000	0	9	767	4.0437	8	3.3172	18	137.2500	2
O_{159}	54	0	2.6852	0.0203	0	0	8.5185	0	0.3649	0.0000	0	1	53	1.9815	2	27.0000	53	27.0000	2

O_{160}	51	0.1961	0.3529	0.6538	0	0.05882	6.2941	0.06061	0.9808	1.1373	0.5098	34	44	1.5400	3	7.1429	34	16.6667	4
O_{161}	756	0	1.0066	0.0000	0	0	4.1429	0	0.9934	0.0000	0	5	553	4.8893	8	3.7463	28	96.0000	3
O_{162}	552	0	1.3297	0.0081	0	0	6.5453	0	0.7459	0.0000	0	1	1	1.0000	1	1.0000	1	1.0000	2
O_{163}	1897	0	0.9995	0.0000	0	0	6.0026	0	1.0005	0.0000	0	1	1556	5.8292	9	5.5468	79	210.7778	3
O_{164}	23	0.0435	1.5217	0.4167	0	0	8.2174	0	0.3833	0.0000	0	21	21	1.0000	1	21.0000	21	21.0000	2
O_{165}	127	0.0315	1.1654	0.1345	0	0.02362	9.1969	0.125	0.7427	0.0236	0.00787	23	97	3.6850	6	4.0968	23	21.1667	4
O_{166}	235	0.0085	1.2298	0.1790	0	0.11064	5.6340	0.13889	0.6676	0.0000	0	33	156	2.9955	7	3.4375	33	31.4286	1
O_{167}	22	0	1.0455	0.1154	0	0	2.3636	0	0.8462	0.0000	0	5	11	2.7576	5	1.9412	5	6.6000	2
O_{168}	140	0	0.9786	0.0000	0	0	1.0000	0	1.0219	0.0000	0	3	137	1.9786	2	35.0000	87	70.0000	2
O_{169}	543	0	1.0552	0.0172	0	0	15.0773	0	0.9314	0.0000	0	4	369	3.9264	9	3.2443	90	63.4444	2
O_{170}	32	0.0938	0.7813	0.1935	0	0	16.9375	0	1.0323	7.1250	0.8125	7	31	1.7813	2	16.0000	25	16.0000	4
O_{171}	822	0.7774	1.0207	0.4541	0	0.0219	14.1484	0	0.5348	1.2032	0.05839	26	665	4.1183	6	5.7797	56	170.5000	4
O_{172}	264	0.0682	1.3182	0.2127	0	0.15909	4.1856	0.38462	0.5973	0.0000	0	2	213	5.1593	7	5.1923	23	38.5714	1
O_{173}	173	0.2197	2.2659	0.2403	0	0	6.2890	0	0.3353	0.1329	0.03468	28	94	3.3822	6	3.1690	28	37.5000	4
O_{174}	227	0	0.9648	0.0000	0	0	1.0000	0	1.0365	0.0000	0	8	219	1.9648	2	25.2222	57	113.5000	2
O_{175}	123	0	1.0244	0.1871	0	0.00813	3.8293	0	0.7935	0.0000	0	3	90	3.5814	5	3.7941	13	25.8000	2
O_{176}	188	0	1.3191	0.0120	0	0	10.3351	0	0.7490	0.8032	0	1	155	4.1186	7	5.5429	20	27.7143	2
O_{177}	325	0	0.0000	0.0000	0	0	7.2954	0	0.0000	0.0000	0	325	325	1.0000	325	325.0000	325	325.0000	3
O_{178}	201	0	0.0000	0.0000	0	0	3.1493	0	0.0000	0.0000	0	201	201	1.0000	201	201.0000	201	201.0000	2
O_{179}	292	0.024	2.8630	0.0933	0	0.01712	0.0000	0.16129	0.3167	0.0000	0	0	0	1.0000	292	292.0000	292	292.0000	1
O_{180}	2177	0	1.2660	0.2942	0	0.51631	6.6863	0	0.5575	0.0000	0	1	1631	7.5017	10	4.5058	101	271.7000	3
O_{181}	418	0	0.9976	0.0000	0	0	2.9928	0	1.0024	0.0000	0	1	416	2.9928	3	139.3333	416	139.3333	2
O_{182}	239	0.0041	0.9672	0.0923	0	0	21.5287	0.22222	0.9385	5.4221	0.90574	4	223	2.8648	3	14.3529	48	81.3333	4

O_{183}	1597	0	1.1897	0.0276	0	0.00125	5.6838	0.13462	0.8173	0.5917	0.54665	2	908	7.0942	16	2.3142	293	100.8125	4
O_{184}	1831	0.0016	0.0688	0.2759	0	0	11.8241	0.08889	10.5230	0.9973	0	1701	1781	1.2534	8	39.8043	1706	228.8750	1
O_{185}	78	0.1154	1.0000	0.3760	0	0	6.4487	0.09302	0.6240	0.0000	0	1	2	1.0256	2	39.0000	76	39.0000	1
O_{186}	38	0.0263	0.7632	0.2927	0	0.05263	2.6579	0.33333	0.9268	0.0000	0	22	22	1.0833	2	8.0000	22	12.0000	1
O_{187}	18	0.3333	0.8333	0.5161	0	0	48.3333	0	0.5806	5.8889	0.88889	3	17	1.8333	2	9.0000	15	9.0000	4
O_{188}	894	0.021	1.0863	0.0363	0	0.02212	8.4181	0.17647	0.8871	0.3816	0	20	768	4.6292	5	10.0349	321	172.6000	1
O_{189}	86	0	1.4535	0.2647	0	0.05814	14.2907	0.46429	0.5059	1.5930	0.03488	5	38	4.4444	8	2.1818	8	9.0000	4
O_{190}	80	0	1.5875	0.0000	0	0	9.0000	0	0.6299	0.0125	0	1	49	4.6519	7	3.6486	10	19.2857	2
O_{191}	2385	0	1.2734	0.0187	0	0.01468	6.2143	0.08696	0.7706	0.0055	0.0109	3	1647	7.4040	14	3.3475	79	350.6429	4
O_{192}	650	0	0.9954	0.0000	0	0	4.0631	0	1.0046	0.0000	0	6	521	3.1976	6	5.0000	39	108.8333	3
O_{193}	44	0	1.8636	0.4570	0	0.65909	9.2500	0	0.2914	0.0000	0	1	9	3.0667	5	2.1429	3	3.0000	2
O_{194}	46	0.0435	0.3913	0.1429	0	0.06522	19.8261	0	2.1905	5.7391	0.6087	28	45	1.3913	2	23.0000	28	23.0000	4
O_{195}	2961	0	3.9875	0.1306	0	0.51604	1.6315	0.20476	0.2180	0.0014	0	1	973	11.2033	17	1.9186	148	1047.2353	1
O_{196}	18	0.3333	0.8333	0.5161	0	0	48.3333	0	0.5806	5.8889	0.88889	3	17	1.8333	2	9.0000	15	9.0000	4
O_{197}	132	0.0545	1.2848	0.3003	0	0.09091	5.6424	0.13044	0.5446	0.1576	0.02424	5	64	2.8504	5	4.8846	38	25.4000	4
O_{198}	389	0.0154	1.6144	0.2629	0	0.10283	11.3419	0.1958	0.4566	0.1465	0.03085	10	29	2.0909	4	3.6667	21	13.7500	4
O_{199}	502	0	1.1793	0.1662	0	0	11.1554	1	0.7070	0.0000	0	1	362	3.2998	6	3.9673	93	101.1667	2
O_{200}	238	0.0042	0.7479	0.7454	0	0.30252	7.8740	0.34491	0.3405	0.0504	0.05042	68	203	2.4612	6	6.4500	68	43.0000	4