

# CLOUD-BASED MACHINE LEARNING AND SENTIMENT ANALYSIS

by

EMMANUEL CHUKWUYEREM OPARA

(Under the Direction of Hayden Wimmer)

## ABSTRACT

The role of a Data Scientist is becoming increasingly ubiquitous as companies and institutions see the need to gain additional insights and information from data to make better decisions to improve the quality-of-service delivery to customers. This thesis document contains three aspects of data science projects aimed at improving tools and techniques used in analyzing and evaluating data. The first research study involved the use of a standard cybersecurity dataset and cloud-based auto-machine learning algorithms were applied to detect vulnerabilities in the network traffic data. The performance of the algorithms was measured and compared using standard evaluation metrics. The second research study involved the use of text-mining social media, specifically Reddit. We mined up to 100,000 comments in multiple subreddits and tested for hate speech via a custom designed version of the Python Vader sentiment analysis package. Our work integrated standard sentiment analysis with Hatebase.org and we demonstrate our new method can better detect hate speech in social media. Following the sentiment analysis and hate speech detection, in the third research project, we applied statistical techniques in evaluating the significant difference in text analytics, specifically the sentiment-categories for both lexicon-based software and cloud-based tools. We compared the three big cloud providers, AWS, Azure, and GCP with the standard python Vader sentiment analysis library. We utilized statistical analysis to determine a significant difference between the cloud platforms utilized as well as Vader and demonstrated that each platform is unique in its analysis scoring mechanism.

INDEX WORDS: Machine learning, Cloud computing, Lexicon-based software, Application programming interface, Sentiment analysis, Cybersecurity analysis, Predictive analysis

# CLOUD-BASED MACHINE LEARNING AND SENTIMENT ANALYSIS

by

EMMANUEL CHUKWUYEREM OPARA

M.S, Georgia Southern University, 2022

A Thesis Submitted to the College of Graduate Studies at Georgia Southern  
University

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE  
INFORMATION TECHNOLOGY

STATESBORO, GEORGIA

© 2022

EMMANUEL CHUKWUYEREM OPARA

All Rights Reserved

# CLOUD-BASED MACHINE LEARNING AND SENTIMENT ANALYSIS

by

EMMANUEL CHUKWUYEREM OPARA

Major Professor: Hayden Wimmer

Committee members: DeJarvis Oliver

Jongyeop Kim

Electronic Version Approved:

December 2022

## DEDICATION

All thanks and glory be to God, for his guidance, understanding of certain topic areas through my master's program. Special appreciation to my family for the financial support in this thesis completion. Finally, I want to thank my faculty advisor for the intellectual support needed to complete this thesis.

## ACKNOWLEDGMENT

A thesis cannot be completed without the help of many people. The author would like to acknowledge the efforts of my committee professors, Dr. Hayden Wimmer, Dr DeJarvis Oliver, Jongyeop Kim and my fellow graduate students.

1	Contents	
2	LIST OF TABLES .....	6
3	LIST OF FIGURES .....	6
4	Introduction.....	7
5	Literature Review.....	8
5.1	Study 1- Auto-ML Cybersecurity Data Analysis Using Google, Azure, and IBM cloud Platforms.....	8
5.2	Study 2- Detection of Hate Speech on social media using an Integrated Lexicon-based software.....	13
5.3	Study 3- Sentiment Analysis using Cloud-based tools and Lexicon-based Software.....	17
6	Auto-ML Cybersecurity Data Analytics Using Google, Azure, and IBM cloud platforms.....	21
6.1	Introduction.....	21
6.2	Background.....	21
6.2.1	Binary Classification.....	22
6.2.2	Multiclass Classification .....	22
6.2.3	Regression.....	22
6.2.4	Anomaly/Intrusion Detection.....	22
6.2.5	Google Cloud Console (GCC) .....	23
6.2.6	Microsoft Azure Console Services .....	23
6.2.7	IBM Watson Cloud .....	23
6.3	Method .....	24
6.3.1	Dataset.....	24
6.3.2	Google Auto-ML.....	29
6.3.3	Azure Auto-ML.....	30
6.3.4	Watson Auto-ML .....	30
6.4	Results.....	31
6.4.1	Google Auto-ML Result .....	31
6.4.2	Azure Auto-ML Result .....	32
6.4.3	IBM watson Auto-ML Result .....	33
6.5	Conclusion .....	37
7	Study 2 – Detection of Hate Speech on social media using Lexicon-based software.....	38
7.1	Introduction.....	38
7.2	Method .....	38
7.2.1	Data Collection .....	39
7.2.2	Data Pre-processing .....	40

7.2.3	Vader Lexicon and Scoring Mechanism .....	41
7.2.4	Vader Lexicon Integration with Hate base Lexicon.....	41
7.2.5	Research Hypothesis .....	41
7.2.6	Preliminary Hypothesis Testing.....	41
7.3	Results.....	42
7.4	Conclusion .....	45
8	Study 3 – Sentiment Analysis using Cloud-based tools and Lexicon-based Software.....	46
8.1	Introduction.....	46
8.2	Background.....	46
8.3	Method .....	46
8.3.1	Data Research .....	47
8.3.2	Lexicon-based Software Sentiment Analysis.....	47
8.3.3	Cloud-based Sentiment Analysis .....	48
8.3.4	Data Normalization.....	50
8.3.5	Research Hypothesis .....	51
8.4	Results.....	52
8.5	Conclusion .....	55
9	Conclusion .....	55
10	References.....	56



## 2 LIST OF TABLES

Table 6-1:Dataset Statistics of UNSW-NB15 showing the distribution of the classes of attacks. ....	25
Table 6-2: Flow Features .....	25
Table 6-3: Basic Features .....	26
Table 6-4: Content Features.....	26
Table 6-5: Time Features.....	27
Table 6-6: Additional generated features.....	27
Table 6-7: Labelled Features .....	29
Table 6-8: Algorithm and cloud platform performance.....	35
Table 6-9: Comparative Analysis .....	36
Table 7-1: Dataset Description .....	39
Table 7-2:Sum and Average of Disney sentiment scores Table 7-3:Sum and Average of 4chan sentiment scores ....	43
Table 7-4:Sum and Average of sentiment scores. Table 7-5: Sum and Average of sentiment scores .....	43
Table 7-6: Paired samples t-test results .....	43
Table 7-7:Sum and Average of sentiment scores. Table 7-8:Sum and Average of sentiment scores .....	44
Table 7-9:Paired sample t-test results for Wikipedia slurs .....	44
Table 7-10:Paired sample t-test results for Gran Torino.....	45
Table 8-1: Dataset Description .....	47
Table 8-2:T-test for negative percentage. Table 8-3:T-test for negative percentage .....	52
Table 8-4: T-test for positive percentage. Table 8-5:T-test for compound percentage. ....	53
Table 8-6:Descriptive analysis showing the statistical values .....	54
Table 8-7: Analysis of Variance test between the cloud platforms and lexicon-based software .....	54
Table 8-8:Sentiment-category comparisons .....	55

## 3 LIST OF FIGURES

Figure 6-1 Creating and Enabling the Auto-ML Environment .....	30
Figure 6-2: Process involved in setting up Auto-ML Environment .....	30
Figure 6-3: Creating the Auto-ML environment .....	31
Figure 6-4: Feature importance and values derived after analysis .....	32
Figure 6-5: Endpoint for batch predictions.....	32
Figure 6-6:Configuration settings in the auto-ml environment.....	33
Figure 6-7: Completed experiment training showing the graph of the Precision-Recall and ROC .....	33
Figure 6-8: Experiment setting in the Auto-ML environment .....	34
Figure 6-9: Algorithm selection for training in experiment environment.....	34
Figure 6-10: Completed training pipeline. ....	34
Figure 6-11: Metric comparison .....	35
Figure 7-1: Data-drop function .....	40
Figure 7-2: Source code for cleaning comment data .....	40
Figure 8-1:Data cleaning in the command line interface.....	47
Figure 8-2:Source code for vader sentiment analysis .....	48
Figure 8-3:Sentiment score in command line .....	50

## 4 Introduction

This thesis includes three research studies which demonstrate the application of descriptive analysis, predictive analysis, and sentimental analysis by utilizing the necessary data analytics tools and techniques to achieve significant statistical results. This thesis includes a detailed investigation into cloud computing and the performance of the evaluating metrics as compared to developed software programs. The first research study focuses on the prediction of vulnerabilities in a network. The network security dataset was extracted, and cloud-based services were utilized in the training of the model. Different attacks were found in the network and a model was trained to predict similar attacks in a test network. Some of the major cloud providers were used in evaluating the performance of the various modelling algorithms selected. The best performing algorithm was evaluated based on the accuracy, precision and recall performance metrics (Opara, Wimmer, & Rebman, 2022). The second research study is based on the application of sentimental analysis on social media data. This was done by extracting reddit data in the form of comments related to hate speech. Also, the use of lexicon-based software was incorporated in this research and the integration with a second lexicon dictionary to further enhance the detection level of hate speech on this microblogging application was performed in this study. The lexicon-based software was implemented on the python programming language because it was fully equipped with machine learning libraries needed for this study. The third study focused on sentiment analysis in cloud computing as compared to lexicon-based software. This study was extended from the second study to further give a descriptive analysis of the sentiment scores derived from the analysis of these different platforms. The aim of this study was to investigate and show evidence that the sentiments derived after analysis from these platforms are significantly different from each other. The cloud-based analysis involved the use of the API endpoints which were used to pull requests. Although, these platforms operate using the same data, pre-processing, and algorithm selection, they predict sentences based on certain unique features. The project implementation phase involved data research and collection, data preprocessing, data normalization and sentiment analysis. To determine significant difference in the sentiments, the use of SPSS tool was incorporated. Analysis of variance test was used to test for significance in this study. In this research thesis, the is sectioned as follows, the literature review for the first, second and third study is outlined. Then this is followed by the detailed procedures followed in each study, starting from study one, study two and study three.

## 5 Literature Review

### 5.1 Study 1- Auto-ML Cybersecurity Data Analysis Using Google, Azure, and IBM cloud Platforms.

Butt et al. (2020) discussed the Machine learning and Deep learning technique used in presenting analysis for Cloud Computing (CC) threats, attacks, and issues, but in this paper, we will be using different ML algorithms to provide a solution to these issues including supervised, unsupervised, and semi-supervised learning. Then we analyzed the different ML algorithms used to solve the security issues and improve the performance in distributed computing. In this review paper, the authors established that using supervised ANN technique and testing their models using the KDD-CUP and NSL-KDD datasets, they were able to detect security attacks and intrusion by unauthorized users. It has an accuracy of 97% in detecting these attacks. Also, the Supervised SVM classifier which finds information for regression analysis, sorting and decision tree, improves the datasets and provides 99.7% accuracy in detecting security threats. What has been achieved here is that using the ANN based technique, there is an important level of data privacy, and it ensures cloud workload protection. Although, there is some functionality in the specialized client-server applications. However, the SVM technique provides a trust-based access control model is an efficient method for security issues in CC and grants data privacy protection. The performance of the network issues can be improved on by research for future scientist (Butt et al., 2020).

Truong et al. (2019) evaluated and compared the distinct types of Auto-ML techniques and tools in Open-Source ML. They highlighted the performance of various Open-source Auto-ML tools. Due to the considerable growth and interest in industrial applications of machine learning (ML) in recent years. That is why Machine Learning (ML) engineers have decided to venture into AutoML in the cloud. They evaluated and selected various AutoML tools and trained and evaluated them on over 300 datasets collected from OpenML, which allows users to query data for different use cases. All AutoML tools are applied to the same training and testing proportions of all datasets. For all evaluations to be done, the following tools and associated versions are used: Darwin 1.6, Auto-sklearn 0.5.2, Auto-keras 0.4.0, Auto-ml 2.9.10, Ludwig 0.1.2, H2O-Automl 3.24.0.5, TPOT 0.10.1. Three different types of supervised learning were used: binary classification, multiclass classification, and regression. All experiments are run on Amazon EC2 p2.xlarge instances, which provide 1 Tesla K80 GPU, 4 vCPUs (Intel Xeon E5-2686, 2.30Ghz) and 61 GiB of host memory. The results of this survey will show that most AutoML tools obtain reasonable results in terms of their performance and efficiency across various datasets. After the various evaluations and benchmarks tested, H2O-Automl, Auto-keras and Auto-sklearn performed better than Ludwig, Darwin, TPOT and Auto-ml. H2O-Automl slightly outperforms the rest for binary classification and regression, and quickly converges to the optimal results (Truong et al., 2019).

Pliuhin, Pan, Yesina, and Sukhonos (2018) used machine learning in cloud technology to solve the issue of electric machines optimization. Microsoft Azure Machine Learning Studio was used to predict the best configuration of an electric motor. Azure ML enabled the computer to learn from the data and trains it to act without being explicitly programmed. They developed a pipeline on the Azure Machine Learning Studio application, the following algorithms Boosted Decision Tree, Regression and Multiclass Neural Network were chosen. To determine the level of optimization, the maximum efficiency was selected. There are some basic steps taken to achieve the model for electric motors optimization in Azure ML; firstly, a dataset in .csv format is uploaded to the Azure ML studio, then a target column is selected after uploading the dataset. The studio automatically splits the data into; initial dataset of 70% for model training in left port and 30% for model test analyses using original data in right port. Then the authors selected the different algorithms, Boosted Decision Tree, Multiclass Neural Network. For each unique combination of the dataset values, cross-validation is conducted and based on the error metric you define. After evaluation, the authors choose

the best-performing model. The results of this survey show that the calculation time was only 1 min 45 s! to Build a model in AML Studio and the metrics quality show a mean absolute error 0.000702, RMS error 0.005926, relative absolute error 0.164582 and relative squared error 1.011483 (the lower value is the better). AML Studio can be used for calculating related size of data and the subsequent optimization procedure (Pliuhin et al., 2018).

Duong and Sang (2018) said the process of training large and complex datasets, require high performance computing infrastructure, which is too expensive to purchase. The interest in cloud computing is because the cloud providers offer these computing services at a cost which is considerable. This is better than purchasing the infrastructure at an excessive cost. As a result, data scientists have turned to the cloud for on-demand and elastic resource provisioning capabilities. Therefore, the authors decided to address the issue facing configuration of the machine learning ML training model on the cloud. This configuration involves the system setup, resource allocation, model, and algorithm development. The authors proposed and implemented a FC2: a web service for fast, convenient, and cost-effective distributed ML model training over public cloud resource. Duong and Sang (2018) proposed the FC2 approach which was aimed to resolve the issue of cloud resource selection and configuration for distributed ML training. They trained large ML models using stochastic gradient descent, which is the standard technique used in training a wide variety of models such as logistic regression or deep learning networks. AWS EC2 was the public cloud platform used for the ML training clusters which composed of virtual machines (VMs) acquired on-demand. A mix of both open-source tools and frameworks was used to implement the FC2 system. The web interface has been implemented using Python/Django. Boto3 and Paramiko are used for interfacing with AWS EC2 and to control cloud instances with SSH. After completion of the ML training task, a Python script triggers a HTTP request from the task's cluster to update the web interface. The CIFAR-10 dataset and the TensorFlow ML framework were used to conduct the experiments. The dataset is a collection of small images which are normally used to evaluate ML and computer vision algorithms. The dataset has sixty thousand color images which are classified into ten classes. Fifty thousand images are used for training, and the rest are test images. The AWS EC2 instance used in this paper was the m4. large instance which is to be the parameter server in cluster setups. The results showed that the Scala-Opt algorithm can effectively make use of scalability properties and provides similar training performance in terms of execution time compared to the Time-Opt algorithm with much lower resource cost (up to 80% cost reduction). It is also observed that Scala-Opt could work well with either GPU or CPU based cloud instances; and it was able to select the lower cost but higher-performing resource given the diverse options from public cloud providers (Duong & Sang, 2018).

Zouhair Chiba (2019) proposed an intelligent approach to build an efficient Deep Neural Network (DNN) based network IDS using the Improved Genetic Algorithm (IGA) and the Simulated Annealing Algorithm (SAA). The Intrusion Detection System (IDS) built here was based on machine learning hence the name MLIDS. The framework was built using the hybrid optimization of the Genetic Algorithm and the Simulated Annealing Algorithm, this results in an intrusion detection system based on machine learning. The platform used for simulation and validation of the method is the Cloud Sim 4.0 simulator. The datasets used in comparison are: CICIDS2017, NSL-KDD version 2015 and CIDDS-001. The experiments were conducted using a Windows 10 – 64 bits PC with 32 GB RAM and CPU Intel(R) Core-i7 2700 K CPU. The genetic algorithm was improved through optimization process which are Parallel Processing and Fitness Value Hashing. The results gotten from the above implementation of the models prove that their model can detect intrusions with higher detection accuracy and precision rate, compared to common model methods developed in previous research. The proposed IDS was placed on the Front-End and Back-End of the Cloud, which can detect and stop attacks in real-time impairing the security of the Cloud Datacenter (Zouhair Chiba, 2019).

Chanthakit and Rattanapoka (2020) proposed a design and implementation of a campus IoT cloud platform. The platform provides four main services: MQTT broker, NodeRED, Apache Spark, and Databases. In the proposed architecture, when a user logs in to the platform, then each user can create and manage MQTT broker, Node-RED, Apache Spark, MongoDB, MySQL, and InfluxDB instances independently. This design runs on only one container per database type and shares among the users to reduce the resources used in the system. Hence, each user can create databases, but the name of the database cannot be repeated by other users. The diagram shows the architecture of the platform. To evaluate the proposed framework, the authors developed an IoT system for plant watering. The system consists of a temperature sensor, and the moisture sensor which serves as the endpoint device used to gather the temperature and moisture from the environment in real-time. Then, the temperature and moisture values are sent to the MQTT broker with the topic tam. Apache spark and Node-RED will get that data. On the Apache spark side, there is a Python script to predict whether the plant watering pump should be turned on or off depending on the received temperature and moisture values. On the Node-RED side, when the Node-RED receives the temperature and moisture data, it will store that data into the InfluxDB database as well as forward that data to its dashboard for visualization, the predicted result from the Python with machine learning script that is executed on Apache spark either on or off is published to the MQTT broker in the topic action. Node-RED subscribes to the topic action to display plant watering pump status on the dashboard. Also, the plant watering device subscribes to the MQTT broker on the topic action to turn on or turn off the plant watering pump according to the received command (Chanthakit & Rattanapoka, 2020).

Ping Li (2018) presented a novel model that protects the data sets of different providers and the data sets of cloud. To protect the privacy requirement of different providers, certain level of encryption (DD-PKE) will be implemented and for the cloud datasets we will use the  $\epsilon$ -differential privacy to be implemented. To resolve the objectives above the named privacy-preserving machine learning under multiple keys (PMLM) will be used to achieve the best encrypted privacy key. Compared to the secure multi-party computation (SMC) only supports the computation on the data encrypted under the same public key and the efficiency and accuracy of the computation need to be improved. The results have established that the model and cloud infrastructure used is proven to be secure in the security model. The experiments also demonstrate the efficiency of our protocol with different classical machine learning algorithms (Ping Li, 2018).

Yeung, Wong, Tam, and So (2019) used machine learning in cloud technology to solve the issue faced by e-commerce developers and users on cloud platforms. They proposed an approach to manage the issue of building data analytics on cloud platforms. They used the Amazon Sage-Maker to illustrate how machine learning models are integrated into data analytic processes. The authors developed a pipeline on the Amazon Sage-Maker application, aimed to address the business IT strategy of implementing the next-generation marketplace, and taking advantage of the available technologies on cloud like machine learning and data warehouse. The authors try to tackle the business need of better understanding customer behaviors and market trends. Four stages were considered when developing the pipeline used to train and evaluate the data. They are as follows, Collecting data from data storage, either from the cloud storage or local PC storage. The file format must be in .csv, processing and transforming data in two modes: Batch Processing and Real-time Processing. Storing data to specific databases after processing on the cloud platform. Consuming data with business intelligence tools. The results of this survey show that cloud platforms have been designed and developed for business cases that require more complicated ML algorithms in doing analysis and predictions. The authors of this paper highlighted how ML is integrated into the data analytics architecture on cloud platform and used Amazon Sage-Maker to illustrate the issue using a real-life e-commerce case for analytic and personalization purposes (Yeung et al., 2019).

Ferreira, Pilastri, Martins, Pires, and Cortez (2021) analyzed eight recent open-source Auto-ML tools (Auto-Keras, Auto-PyTorch, Auto-Sk-learn, Auto-Gluon, H2O Auto-ML, R-miner, TPOT and TransmogrifAI) and they used twelve popular Open-ML datasets and as the benchmark they divided them into regression, binary and multi-class classification tasks. Then, a comparison with different Machine Learning (GML), Deep Learning (DL) and XGBoost (XGB) models was conducted. To select the best tool, we used a lexicographic approach, considering first the average prediction score for each task and then the computational effort. [9] developed all experiments using an Intel Xeon 1.70GHz server with fifty-six cores and 2TB of disk space. For the three scenarios, the same measures were used to evaluate the performance of the external 10-fold test set predictions. Popular prediction measures were selected: regression – Mean Absolute Error (MAE) ( $\in [0.0, \infty[$ , where 0.0 denotes a perfect predictor); binary classification - Area Under the receiver operating characteristic Curve (AUC) ( $\in [0.0, 1.0]$ , where 1.0 denotes the ideal classifier); multi-class classification – Macro F1-score ( $\in [0.0, 1.0]$ , where 1.0 denotes the perfect model). When the Auto-ML tool allowed specifying a time limit for training, the chosen time was one hour (3,600 s). The results of this survey show the best tools for each scenario, after adopting a lexicographic approach, which considers first, for each task, the best average predictive score and then the lowest computational effort. For GML, the lexicographic selection favors Transmogrif-AI for binary classification, Auto-Gluon for multi-class classification and R-miner for regression. For DL, the selection is H2O for the binary and regression tasks and Auto-Gluon for regression. As for the XGB scenario, R-miner is the best overall option for binary and regression tasks, while H2O is recommended for multi-class tasks (Ferreira et al., 2021).

When developing the machine learning pipeline, the process could be so tedious when using open-source ML. Yang, Fan, Wu, and Udell (2020) designed a new Auto-ML system Tensor-Oboe to address the long process in automated supervised learning pipeline. The Tensor-Oboe designed uses a low rank tensor decomposition as a surrogate model for efficient pipeline search. During the development phase a new greedy experiment design protocol to gather information about a new dataset efficiently was developed. The Tensor-obe was developed in two phases, the online and offline phase. The computing performance of pipelines on meta-training datasets was done by the offline phase, to build the tensor surrogate model, while the online phase ran a small number of pipelines on the meta-test dataset to analyze the surrogate model and identify promising pipelines. The experiment was conducted on a Linux machine with 128 Intel® Xeon® E7-4850 v4 2.10GHz CPU cores and 1056GB memory. The authors collected cross-validated pipeline performance on 215 Open-ML meta-training classification datasets with number of data points between 150 and 10,000 that are chosen alphabetically (Yang et al., 2020).

Chahal, Ojha, Choudhury, and Nambiar (2020) proposed the migration of a recommendation system to the cloud using ML workflow. They implemented a recommendation system which was deployed on-premises. The recommendation system was called I-Prescribe, and a method was developed on how to move the system to the cloud. The recommender interface extracts the user context from the incoming message, fetches its feature for the model, builds concatenated one-hot vector and inference from the model to get the best offer for the user. Then the implementation on the cloud was done using MXNet with AWS sagemaker python SDK. The on-premises server used for deploying iPrescribe had twenty-eight physical cores. Instacart consists of three million grocery orders obtained from 200,000 Instacart users. It was observed that with the AWS sage maker, the throughput was improved because of the increased number of active cores and workload (Chahal et al., 2020).

H. Zhang et al. (2020) built the MLModelICI which helps users publish models. The models are automatically converted to optimized formats for production purposes and then profiled under batch-size and hardware settings. The profiling information can be used as guidelines for balancing the trade-off between performance and cost of MLaaS. Finally, the system dockerizes the models for ease of deployment

to cloud environments. MLModelCI was developed to keep the entry barrier as low as possible, so as to make the integration into existing toolchains as efficient and seamless as possible (H. Zhang et al., 2020).

Biswas, Majumdar, Nandy, and El-Haraki (2015) presented a technique to auto-scaling of cloud resources provided by an intermediary enterprise which services requests from a client enterprise. A broker is deployed by the intermediary enterprise to manage client requests with service level agreements (SLAs). A reactive auto-scaling algorithm is activated on request arrival and achieves auto-scaling by acquiring novel resources for serving the recently arrived request. This system will profit intermediary enterprise as well as a reduction of cost for client enterprise (Biswas et al., 2015).

Zhang, Xu, Zhao, Zhang, and Wang (2020) designed and implemented a time prediction model for the LS-DYNA cloud computing platform. The correlation between the hardware conditions of the relevant historical model and the calculation time is set as the training set. regression analysis is adopted, and four core steps are required to establish the regression equation, including curve estimation, automatic linear modeling, regression analysis and Durbin-Watson test. With the minimum and the maximum of relative error to be 102.1% and 582.0%, respectively. On the whole, the average relative error is about 360.1%, which means the prediction error by LS-DYNA is very high (H. Zhang et al., 2020).

Liu, Li, Niu, and Cao (2014) discussed on the development of a cloud-based experiment platform. They customized an IaaS cloud for the experiment context and proposed solutions for software-based experiment and hardware-related simulation, respectively. The prototype system developed showed virtualization efficiency and user experience. The prototype which was developed has six physical servers. The adopted server is DELL PowerEdge T110 II, with the hardware configuration of one E3 CPU, 32GB memory and 4TB hard disk. With this approach, teachers can effectively manage and control the experimental environment easily. Adapting more physical equipment's and optimizing the performance of the platform are the further work (Liu et al., 2014).

## 5.2 Study 2- Detection of Hate Speech on social media using an Integrated Lexicon-based software.

The Internet has become the main source for news acquisition and the use of various social media platforms and social networking applications has become an essential part of our daily life. Its growth is facilitated by the continuous use of the internet. Social media is an inspiring platform for online learning, exchanging ideas and sharing opinions. It is also used by researchers, scientists, and industry to conduct research on different fields ranging from social, political, medicine and various other fields. Social media involves individuals sharing opinions and ideas in the form of texts, posts, status, and blogs. The process of analyzing each text in a sentence written in the tweet, text, or post to get certain information in the form of opinion is called sentiment analysis.

Ruwandika and Weerasinghe (2018) developed a model to detect hate speech using machine learning techniques. The author utilized both supervised and unsupervised machine learning techniques. The techniques used were Logic regression, Navies bayes, Decision tree and K-means. The author compared the results gotten from the algorithms with lexicon-based approaches, and the Navies Bayes classifier performed best with an F-score of 0.719. The dataset used were comments from News articles posted on the Colombo Telegraph website. 1500 preprocessed comment dataset were used (Ruwandika & Weerasinghe, 2018). Researchers and scientists lack a general understanding on what type of content attracts hateful discussions and the possible effects of social networks on the commenting activity on news articles.

Pereira-Kohatsu, Quijano-Sánchez, Liberatore, and Camacho-Collados (2019) proposed a system called Hater Net, which has taken a gold standard. This system identifies and monitors the evolution of hate speech in twitter. The system is currently utilized by the Spanish National Office Against Hate Crimes of the Spanish State Secretariat for Security. The dataset used was an initial corpus of more than two million tweets. Twitter's API Rest was used to download these tweets. The dataset was manually tagged and carefully filtered to produce both the training and testing datasets. For the Feature extraction, an NLP preprocessing pipeline was followed. This involved tokenization, POS tagging and lemmatization. The author applied filter method-based features for the feature selection combined with a LASSO model (Least Absolute Shrinkage and Selection Operator). After all necessary pre-processing was done, a newly developed dataset consists of six thousand expert-labeled tweets. Supervised machine learning classifiers were used to develop a model in which the dataset was trained. The best approach amongst the classifiers consists of a combination of a LTSM+MLP neural network that takes as input the tweet's word, emoji, and expression tokens' embeddings enriched by the tf-idf and obtains an area under the curve (AUC) of 0.828 on our dataset, outperforming previous methods presented in the literature (Pereira-Kohatsu et al., 2019).

Zampieri et al. (2019) proposed a prediction monitor to target offensive posts on social media. In the paper, the authors compiled an Offensive Language Identification Dataset (OLID), that is a manually curated dataset. The Twitter API was used to extract these comments from tweets. The tweets were annotated for offensive words, this was achieved using a three-layer annotation scheme. The authors conducted a round of trial annotation of three hundred instances with six experts using nine keywords. The aim of the annotation was to evaluate the proposed tag set and data retrieval method. It creates a gold standard with instances that were used as test questions to ensure the quality of the annotators for the rest of the data. This new dataset was used to compare with pre-existing datasets for hate speech identification. The authors highlighted the similarities and differences that occur when training both datasets. Further experiment was



carried out to compare the performance of different machine learning models on OLID (Zampieri et al., 2019).

Kumaresan and Vidanage (2019) proposed a system that aims at improving the detection of hate speech. The social medium platform used for the experiment was twitter. The authors utilized both ontologies and fuzzy logic approaches combined with sentimental analysis to determine hate speech and deconstruct the ambiguity present. The tweets were classified into hateful, offensive, and neutral. The dataset used was derived from previous research work which has been published. The system achieved an F1-score higher than previous research work. The F1-score achieved was 0.677 for the Hate class and 0.9805 for the offensive classification. The Vader sentiment library was utilized for the sentiment analysis. It was developed to assist companies and research organizations moderate online conversations and use them as a hate speech predictor. Reddit has its own principles and standards, organized around its communities called subreddits. Subreddits differ from each other in many ways, especially in four specific dimensions: topic, audience, moderation, and style (Kumaresan & Vidanage, 2019).

Pitsilis, Ramampiaro, and Langseth (2018) developed a detection scheme using Recurrent Neural Network (RNN) classifiers and incorporated user-related information. These data were fed as input to the above RNN classifier along with the word frequency vectors derived from the textual content. Long Short-Term Memory Network (LSTM) was the RNN classifier used. The model used consisted of four layers. The input layer, the hidden layer, the dense layer, and the output layer. In the input layer, the number equals the size to the word vector plus the number of additional features. The word vector dimension was set to 30 so that to be able to encode every word in the vocabulary used. The hidden layer was made up of sigmoid activation, which is connected to the input layer and the dense layer. The dense layer was just an additional layer used to obtain a more stable result. The ReLU activation function was utilized in this layer. The output layer provided output in the form of probabilities for each of the three classes Neutral, Racism, and Sexism. The SoftMax activation function was used for this layer. The author's scheme was evaluated on a publicly available corpus of 16k tweets, and the results demonstrate its effectiveness in comparison to existing state of the art solutions. The results derived from incorporating features related to user's behavior into the classification has provided a significant increase in the performance vs. using the textual content alone,  $F = 0.9295$  vs.  $F = 0.9089$  (Pitsilis et al., 2018).

Gitari, Zuping, Damien, and Long (2015) utilized a lexicon-based approach to detect hate speech in web blogs and comment sections. The aim of their research was to create a model classifier that uses sentiment analysis techniques and in particular subjectivity detection to not only detect that a given sentence is subjective but also to identify and rate the polarity of sentiment expressions. They started by whittling down the document size by removing objective sentences. Then, using subjectivity and semantic features related to hate speech, we create a lexicon that is employed to build a classifier for hate speech detection. To perform subjective sentence detection, they employed a rule-based approach that classified sentences relying on a lexicon of well-established clues. They utilized two known sentiment lexicon resources of (Riloff & Wiebe, 2003) and SentiWordNet package in python. There has not been as much work on sentiment analysis using lexicon-based techniques at the document level. However, the authors of this paper made progress on building lexicons for sentiment analysis (Gitari et al., 2015).

Graumas, David, and Caselli (2019) proposed a method that generated polarized word embeddings using controversial topics on Twitter as proxies for interactions among social media communities that may be liable to use abusive language. The authors obtained their datasets from literature published. They derived three datasets from the literature and used these datasets to train and evaluate the prediction models

developed. Two data sets explicitly consisted of words and sentences in the category of hate speech, while the other words consisted of a broader category of offensive language. The results of their experiments, based on simple linear SVM models, showed that the word embeddings, both generic and polarized, outperform n-grams based models across data sets, showing better generalization capabilities, although they fail to outperform such models in the same data distribution scenario (Graumas et al., 2019).

Tan and Lee (2015) proposed a study which examined three aspects of multi-community engagement in Reddit. There was a sequence of communities whereby users post the language that users employ in those communities, and the feedback that users receive, using longitudinal posting behavior on Reddit platform as the main data source, and DBLP for auxiliary experiments. During the investigation, the authors found that over time, users span more communities every ten posts, “jump” more, and concentrate less. Fairly operational users seem consistently less “adventurous” than continuous operational users even, notably, from the very beginning. Curiously, operational users imitate continuous operational users in the top activity quartile. The authors demonstrated the effectiveness of features drawn from these aspects in predicting users’ future level of activity (Tan & Lee, 2015).

Aggarwal, Gola, and Sankla (2021) proposed an expert model for hate speech detection works towards overcoming the strong user-bias present in the available annotated datasets. The authors utilized A-Stacking hybrid classifier based on ensemble learning that was used clustering to form weak hypotheses that was systematically integrated using a meta-classifier in a later stage. The model was adaptive with the properties of the dataset which was used to generate the hypotheses used as base-classifiers. The Waseem and Hovy dataset were used, and it came with tweet identifiers along with their associated class labels, i.e., sexist, racist and non-hateful. The actual tweets were extracted using any tweet crawler. 16k tweet identifiers constitute the dataset. The results show that the proposed model could adapt to the properties of data and behave accordingly when the test environment is changed. The authors emphasized that the available annotated datasets have a strong bias in them. For the correct assessment of the model, it is necessary to restrict the number of tweets per user (Aggarwal et al., 2021).

Gaydhani, Doma, Kendre, and Bhagwat (2018) proposed an approach that automatically classified tweets on Twitter into three classes: hateful, offensive, and clean. They utilized Twitter dataset to perform experiments considering n-grams as features and passing their term frequency-inverse document frequency (TFIDF) values to multiple machine learning models. They utilized three machine learning algorithms for text classification: Logistic Regression, Naive Bayes, and Support Vector Machines. They utilized the Scikit-learn package in python for the implementation. Logistic Regression had the best performance, so it was used to evaluate the test data. The authors observed that the recall value for offensive tweets was 0.93, which signifies that 7% of the offensive tweets were misclassified by the model. The precision for the hateful class is 0.94, which signifies that 6% of both clean and offensive tweets were classified as hateful. On the other hand, the recall for clean class is 0.98, which is significantly better. The authors performed a comparative analysis of the models considering several values of n in n-grams and TFIDF normalization methods. After tuning the model giving the best results, it was achieved with a 95.6% accuracy upon evaluating it on test data (Gaydhani et al., 2018).

Arulmurugan, Sabarmathi, and Anandakumar (2019) utilized an approach to predict sentiment polarity of text toward a specific aspect. Although existing neural network models show promising performances on ABSA, their capabilities can be unsatisfactory in cases where the amount of training data is limited. The authors used multiple dictionaries and knowledge sources to improve the system which they developed.

The objective of the system was to perform aspect-based sentiment analysis (ABSA), which utilized multiple sources of text knowledge to predict sentiment. BiLSTM modelling layer was utilized as an attention mechanism that calculates important scores after encoding the contextual information of text into representations of words at each time step. Structure knowledge is extracted via clause recognition and fused into the model through the BiLSTM layer. However, not all words are equally important in terms of expressing sentiment toward a specific aspect. That is why the attention mechanism was infused in this approach. Sentiment knowledge is exploited by means of training a general classification model with the sentiment labels of documents and fused through pretraining specific layers to extract contextual features and predict sentiment polarities more accurately (Arulmurugan et al., 2019).

Melton, Olusanya, Ammar, and Shaban-Nejad (2021) conducted research on the social media platform called reddit. They were investigating if the social media platform had any role in the GameStop short squeeze in 2021. The subreddit used for this research was the r/WallStreetBets. The discussion held during the time of the event was used for the analysis of the American online retailer (GameStop). Over 10.8m comments were extracted. The comments were divided into two categories to contain the sentiments. The long messages are threads containing more than 2k comments, while the second group short messages are threads containing less than 2k comments. Sentiment analysis was performed using the lexical approach. The Vader sentiment analyzer was the lexicon dictionary used to assess the sentiment of phrases and sentences in the comments, without the need of looking at anything else. The sentiment analyzer extracts the polarity scores and provides the overall sentiment metrics (compound score) for the comments. The compound score is greater than 0.05, it denotes a positive sentiment. When the compound score is less than -0.05, it denotes a negative sentiment. For the compound score lies between 0.05 and -0.05, it denotes a neutral sentiment. To further assess the connection between Reddit investors sentiments and GME price changes the authors employed the wavelet coherence framework. This method utilized here is powerful for analysis of shorter observation periods and can help to identify time-frequency co-movements between selected variables adding to the regression analysis results. From the results, it shows that both tone and number of comments influence GME intraday returns. Sentiments extracted from longer threads have a greater influence. "Fear" is the dominant sentiment in all comments, while comments that express a "Sad" sentiment show the most significant impact (Melton et al., 2021).

Machova, Mach, and Vasilko (2021) proposed an offensive language detection system to analyze over 50,000 right wing German hate tweets posted between August 2017 and April 2018, at the time of the 2017 German federal elections. The authors used both quantitative and qualitative methods to analyze this large corpus of data. The dataset was divided into 5 categories, First and Second level incitement speech category, insult speech category, First and second level slander category speech. The authors used a combination of qualitative approaches with quantitative techniques from Natural Language Processing (NLP). To evaluate the qualitative approach, they analyzed a random subsample of 2,000 tweets, then compared the results to support quantitative evidence. They focused on a selection of hate speech tweets. They utilized the SentiWS lexicon for German. It assigns scores to words like gut = +0.37, and shecht = -0.77, which was used to compute an average score for a given text. They computed the average score for all tweets resulting in about 32% of the hate tweets predicted as negative, against 22% of the safe tweets, or a 10% difference. They utilized Character trigrams to efficiently model linguistic variation such as spelling errors, word inflections, function words. In order to evaluate the Perceptron algorithm used, they did a cross evaluation with a hold-out set of the 1,000 most offensive examples in the hate speech dataset (Machova et al., 2021).

### 5.3 Study 3- Sentiment Analysis using Cloud-based tools and Lexicon-based Software.

Gajananan et al. (2018) developed a model to extract sentiment polarity changes from sequences of support tickets issued over a period and they utilized machine learning to predict the subscription renewal by the customer. The goal of this study is to categorize the sentiment polarity changes extracted from sequences of support tickets, in combination with other ticket history data, by learning a feature representation that subsequently can be passed to a standard binary classifier to predict subscription renewal. In this work, the authors relied on representation learning to automatically understand the sentiment features with characteristics, such as being able to a) encode the temporality associated to the sequence of tickets b) be treated as a vector of continuous values. In addition, this research study was to seek the impact of learned representations of sentiment polarity changes extracted from sequences of support tickets, in combination with other feature families, on the performance of cloud-based service subscription renewal prediction. The dataset used in this research contained around 90,000 associated support tickets. Some features derived from the tickets combined with other handcrafted features relating to ticket data, were passed through a classifier which estimated the likelihood of service subscription renewal by the customer. A total of seven case features were derived from the ticket data. The IBM Watson cloud platform was utilized, and the natural processing language API was used to extract the sentiments (Gajananan et al., 2018).

White and Rege (2020) performed sentiment analysis using tools on the google cloud platform. The experiment was conducted utilizing two services available on the Google Cloud Platform (GCP) for performing sentiment analysis, which are Natural Language API and Auto-ML Natural Language. Comparisons were made between these two services. The dataset used in this experiment consists of 1.5 million labeled comments which were retrieved from the Kaggle website. 1,675,189 comments were labeled as positive sentiments, while 104,796 were labeled as negative comments. Python scripting language was used in the pre-processing and removal of duplicate comments. When performing sentiment analysis using Natural language API on GCP, the authors created a project on google cloud platform, then they enabled the google Natural Language API. They created a service account on the platform to bill them for whatever service they would be utilizing. A private key was provided to the authors as a JSON file to enable the virtual python development environment. Finally, the authors execute the commands for sentiment analysis using the python development environment. The results obtained when using natural language API were represented in a confusion matrix and an ROC Curve. The overall accuracy of the prediction was only 57%. When utilizing Auto-ML natural language, the author labeled the data as "TRAIN", "VALIDATION", and "TEST". The Auto-ML service will automatically train a model using the training and validation data. After the training is complete, the service uses the testing data to determine the true quality of the sentiment analysis and makes various quality metrics available on the GCP console (White & Rege, 2020).

Qaisi and Aljarah (2016) performed sentiment analysis using the following cloud service providers namely, Amazon and Microsoft Azure to analyze their customers opinions and reviews. To do that, two datasets are extracted which consist of tweets that had either organizations names or cloud names. In this experiment, two datasets were created from the twitter API. Navies Bayes classifier was the algorithm used to train the dataset. Results were analyzed and explained in terms of polarity and emotions classifications, this was to prove the impact of sentiment analysis to support organizations decisions. The total number of tweets extract for each dataset was 1500tweets. After analysis, results show from the emotions classification that Microsoft Azure has 65% positive tweets compared to 45% positive tweets for Amazon. Amazon has 50%

negative polarity compared to 25% negative polarity for Microsoft Azure. Word cloud representation was used to identify the most frequent words in each emotions category (Qaisi & Aljarah, 2016).

Alkalbani, Ghamry, Hussain, and Hussain (2016) proposed a research study which was to investigate consumers experience of using SaaS services. In order to establish this, the authors applied sentiment analysis and classification using NLP and ML on the Blue Pages Reviews dataset (BPR) that was generated by crawling SaaS consumers reviews from different web portals. Alkalbani et al. (2016) extracted four thousand online reviews and they utilized sentiment analysis to identify the polarity of each review, that is, whether the sentiment being expressed is positive, negative, or neutral. This research also develops a model for predicting the sentiment of Software as a Service consumers' reviews using a supervised learning machine called a support vector machine (SVM). The application of both Natural Language Processing (NLP) and Machine Learning (ML) was achieved by using Rapid Miner environment and a SaaS application called Semantria, developed by Lexalytics. The major step of this research was the pre-processing of the text. To accomplish this task the authors made use of Text Processing Plug-in Rapid Miner to perform this task, this tool was called "Process Document from Data." They made use of SVMs with three different approaches to word vectors, namely Binary Occurrence, Term Frequency and Term Frequency-inverse document frequency (TF-IDF). The sentimental results show that 62% of the reviews were positive, which indicated that more consumers were satisfied with SaaS services. The results of the prediction accuracy of the SVM-based Binary Occurrence approach (3-fold cross-validation testing) was 92.30%, which indicated better performance in determining sentiment compared to the other approaches (Term Occurrences, TFIDF) (Alkalbani et al., 2016).

Carvalho and Xu (2021) proposed a strategy to reduce the variability in accuracy by creating ensemble models formed by cloud platform technologies. The experiment performed in this paper was to investigate the performance of different ensembles of cloud-based technologies for sentiment analysis. The authors discovered that the ensemble models performed better on longer texts. Two comments datasets were used in this experiment were obtained from twitter and Facebook. They utilized Natural Language Understanding (NLU) service in IBM cloud to obtain sentiment scores within the interval  $[-1, 1]$ . The Text Analytics cognitive service in Microsoft azure was used to obtain scores inside the interval  $[0, 1]$ . Finally, the authors utilized the Natural Language service in Google cloud to obtain sentiment scores within the interval  $[-1, 1]$ . The final dataset used after data transformation consisted of 10,411 tweets and 3,209 Facebook posts. The Twitter data set had 7,368, 1,532, and 1,511 negative, neutral, and positive tweets, respectively. The Facebook dataset had 1,618 negative, 463 neutral, and 1,128 positive posts. To balance their data sets, they performed stratified sampling. That is each sample size was equaled to the absolute frequency of the least popular sentiment in the dataset. This sampling approach implies that our final Twitter data set has  $3 * 1,511 = 4,533$  observations, whereas the Facebook data set has  $3 * 463 = 1,389$  observations. They created the ensemble models by combining outputs produced by the three sentiment-analysis technologies. The "average model" aggregated the three sentiment scores received by a text by simply averaging them. When the average score was within the interval  $[-1, -0.33]$ , then the assigned label is "negative". When the average score was within the interval  $[-0.33, 0.33]$ , then the corresponding label is "neutral". The label was "positive" when the average score was within the interval  $(0.33, 1]$  (Carvalho & Xu, 2021).

Keijzers, Bartneck, and Kazmi (2019) proposed a study that evaluated whether sentiment analysis tools can accurately gauge sentiment in human-chatbot interaction. So, the authors compared the quality of sentiment analysis obtained from Microsoft, Amazon, and Google. To further the study, they compared their results

with the leading lexicon-based software, as well as with human ratings. The authors wanted to prove whether the cloud-based sentiment analysis tools agree with each other and whether the cloud-based sentiment analysis tools agree with the lexicon-based software. Finally, whether the cloud-based sentiment analysis tools agree with human judgements and does the lexicon-based software output agree with human judgements. The authors of this paper used 285 anonymous conversations between Clever Bot, an online chatbot, and a user to analyze. To make the comparison of these analysis tools, the authors transformed all sentiments to report on a range of 0 to 1. Where zero was negative, one was positive and neutral was .5. Therefore, for the amazon comprehend tool which reports four different scores, the authors only made use of the positive and negative sentiment. The negative was subtracted from the positive and the result was added to .5 to derive a single score in the range of 0 and 1. The results show that although the sentiment analysis tools agree moderately with each other, they do not correlate well with human ratings. While the cloud-based services would be an extremely useful tool for human-agent interaction, their current quality does not justify their usage in human-agent conversations (Keijsers et al., 2019).

Al-Omair and Huang (2018) proposed in this paper a comparison of cloud-based emotion recognition services. The authors compared Amazon, Google, and Microsoft cloud platforms in this study. This service detects the emotion of humans using the computer vision, The technology of facial expression recognition is the ability to portray human emotions by analyzing facial expressions. There are various solutions that provide this technology. Typically, a face was detected then a prediction of an emotion is given. The dataset used in this study came from the Karolinska Directed Emotional Faces Database (KDEF). This dataset consists of seven different emotions: afraid, angry, disgusted, happy, neutral, sad, and surprised. Each of 70 individuals in the images portrayed each emotion in two different sessions, a total of 140 images per emotion. The subjects consisted of thirty-five males and thirty-five females between the ages of 20 and 30. Within each emotion iteration was a nested iterative function that evaluated each image within that emotion. The image was uploaded to the cloud-based emotion recognition service API being evaluated. The API derived the confidence rates for several emotions found in the images after analysis. The confusion matrix was generated for each cloud service and was used to display all the predictions made for a certain emotion. Microsoft had the highest average accuracy rate of 97%, compared to Google and Amazon that achieved 85% and 79% accuracy respectively (Al-Omair & Huang, 2018).

Karyotis, Doctor, Iqbal, James, and Chang (2018) proposed an emotion modeling methodology for incorporating human emotion into intelligent computer systems. The authors developed and evaluated the AV-AT computational model of emotion which included an online survey. This survey provided the data from which the computational emotion model was created. The authors used the following emotion labels of flow, excitement, calm, boredom, stress, confusion, frustration, and neutral in the online survey. These labels were used in comparison with 'arousal,' 'valence', 'prediction', and 'evaluation of the outcome' were explored. The survey was generated with the help of the online tool QuestionPro. Eighty participants of various ethnic origins completed the online survey. They utilized two fuzzy classification systems, one for each stage of the emotion model. The training samples contained three inputs and eight outputs for each stage. In the first stage, the inputs were arousal, valence, and prediction, and in the second stage, they were arousal, valence, and outcome. In both stages, the outputs are values of the eight emotions (flow, excitement, calm, boredom, stress, confusion, frustration, and neutral). All variables take values in the interval [0,100]. The performance of the model was evaluated through the deployment of a personalized learning system, and series of offline and online experiments. A hybrid cloud intelligence infrastructure was utilized, and they conducted a large-scale experiment to analyze user sentiments and associated emotions, using data from a million Facebook users (Karyotis et al., 2018).

Tedeschi and Benedetto (2015) proposed a big data sentiment analysis using cloud services. The data was extracted from twitter, the microblogging application using the API. Then Microsoft Azure was utilized in the development of this system. The system had a client web user interface. The SBM's server provided several services through the defined Web Service RESTful and the Windows Azure platform. For instance, the server allows SBM's users to start a new search sending a specific query to the server, which starts a crawling session and stores all the retrieved data in a shared database. The authors generated a dataset of more than 300,000 tweets on several brands and products tweeted by more than 200,000 authors. The authors created a pie chart which gave a better visualization of the analysis of the twitter comments (Tedeschi & Benedetto, 2015).

Harfoushi, Hasan, and Obiedat (2018) compared the lexicon-based analysis with cloud-based analysis. The dataset utilized in this study was Coachella 2015 Twitter sentiment dataset. It consisted of ten columns and 3800 tweets. It consisted of ten created fields; two columns were used to evaluate sentiment analysis algorithm. Coachella sentiment is the first column and encompasses the sentiment of a tweet. Azure-based Sentiment Analysis was utilized in training model and similar procedures of sentimental analysis were followed. A comparison of algorithms was conducted in this study. Support Vector Machine algorithm achieved the highest accuracy values at 73%, 59% and 60% for the three dataset A, B and C. For logistic regression, the values were 0.53, 0.53 and 0.57 respectively for dataset A, B and C (Harfoushi et al., 2018).

## 6 Auto-ML Cybersecurity Data Analytics Using Google, Azure, and IBM cloud platforms.

### 6.1 Introduction

Machine Learning (ML) is a division of artificial intelligence, it is an approach to data analytics that automates analytical models. The system learns from the data, identify patterns, makes decision and prediction (Salza, Hemberg, Ferrucci, & O'Reilly, 2017). The ability to systematically apply complex mathematical calculations to big data is also referred to as machine learning. The future of Automated Machine Learning (Auto ML) has improved the creativity for data scientists, ML engineers and ML researchers by reducing repetitive tasks in machine learning pipelines. Auto ML is designed to effectively solve problems of classification, pattern identification, complex systems behavior prediction and selecting unknown parameters that relate the characteristics of complex objects (Ding & Hsu, 2018). Various cloud platform providers have introduction machine learning tools and libraries in them, to easily build, train and test models and integrate them into business and industrial solutions. Some of the Cloud Providers who will be using this paper are Microsoft Azure Cloud, Google Cloud Platform, IBM Cloud. These platforms have integrated the core ML algorithm, pipelines, libraries. We will be making use of the free services offered to establish this model. On the Google cloud platform, Vertex AI is used, models can be trained and can be assigned to an endpoint to get predictions, because of the features embedded in it. On the Azure Machine learning studio, we created an Auto-ML pipeline, to run a predictive analysis using the UNSW-NB15 network security dataset. The dataset used is stored in the Azure Blob storage. Create a dataset by uploading the (.csv) file or choosing an existing dataset from the datastore location in Azure Blob storage. After creating the dataset, you configure your experiment design, by selecting a compute target and the column you decide to predict. Auto ML randomly splits the data into two, 80% training, while 20% testing for data validation (Swasey, Murphy, Crary, & Harper, 2006). The algorithms used to model this data set were, Decision tree and Random Forest tree. This experiment process takes time due to the computational environment allocated to your account subscription. Finally, I explore the model created and give explanations to these models. In this paper, the same process was repeated on the other two platforms, respectively. The results were evaluated and compared.

### 6.2 Background

In recent years, data scientist jobs are the most sort after job in the world today. There has been a reasonable growth level in the applications of machine learning. It has found application for different sectors of life. Machine learning can simply be defined as a type of prediction being made, based on the question that is being asked, and the available dataset. Automated machine learning (Auto-ML) has increased the creativity and productivity of data scientists because it saves time and effort on the repetitive tasks in ML training pipeline. Before the introduction of Auto-ML, there were opensource tools which are commonly used for training ML models. Some of these open-sources tools are also automated now, and they are embedded in the cloud provider. Some of the ML libraries and tools which have been developed in past are, Weka (1990s), RapidMiner (2001), Scikit-learn (2007-2010), H2O (2011), Spark MLlib (2013), Tanagra, Orange, KNIME (Konstant Information Miner) and so on (Gangadhar & Shanta, 2018). Automated machine learning has been offered as a service by different cloud providers. Various cloud platforms have developed machine learning studios which are embedded on their different platforms. The benefit of this is that the cloud platforms have embedded in them the necessary virtual machines (VM), CPU & GPU processors, storage needed to perform model training. In this section, we would discuss the technique/tasks used by data scientist for machine learning in the cloud and then we would discuss the various cloud providers



highlighted in the introduction and the Auto-ML tools embedded in them. In this research study, we will discuss the following machine learning tasks which are performed on the different cloud platforms discussed in the following section.

#### 6.2.1 Binary Classification

In machine learning, classification refers to a predictive modeling problem where a class label/target is predicted for a given dataset. Therefore, binary classification can be defined as a supervised machine learning task that is used to predict which of two categories an instance of data belongs to. That is ML is used to predict the target column in each dataset. A popular example of classification is e-mail fraud detection. A binary classification task will split the given email dataset into two categories, it will be to predict spam or not (Jeffers, Reinders, & Sodani, 2016). Another scenario in binary classification is when diagnosing whether a patient has a certain disease or not. To achieve the best result, the training data must be balanced i.e, there should not be missing data. The input target column data must be Boolean (binary variable). The evaluation metrics for binary classifications are accuracy, area under the curve (AUC) which should be close to 1, Area under the curve of a precision (AUCPR) and F1-score (Jeffers et al., 2016).

#### 6.2.2 Multiclass Classification

This is like binary classification as they are both types of supervised learning. In multiclass classification, it is used to predict multiple categories for the response variable in each dataset. The input of a classification algorithm is a set of labeled examples. Each target data normally starts as text. It is then run through the Term-Transform, which converts it to the Key (numeric) type. The output of a classification algorithm is a classifier, which you can use to predict the class of new unlabeled instances (Wimmer, 2016). Some scenarios of multiclass classification are, Categorizing flights into "early", "on time", or "late". Rating of movies such as "positive", "neutral", or "negative". The F1 score is the evaluating metric for classification tasks. It is the weighted average of precision and recall. Precision and recall also make an equal contribution to the F1 ranking. Some algorithms used to train multiclass classification models are, k-Nearest Neighbors, Decision Trees, Naive Bayes, Random Forest, Gradient Boosting.

#### 6.2.3 Regression

This machine learning task is used to predict the value of the target from a set of related features. The target is usually any real value, compared to the classification tasks whereby the target is from a finite set of values. This is a type of supervised learning, that predicts numeric scores or values. A common scenario in which regression is used is in the prediction of stocks prices in the coming days. The root mean square error (RMSE) and error rate are the major evaluating metric for regression (Jeffers et al., 2016). RMSE is the square root of the average squared distance between the actual score and the predicted score. To train a regression model, the following algorithms can be applied, Light gradient boost regression, Fast-forest regression, decision trees, Gradient boosting, Fast-tree regression. The input target column data must be single.

#### 6.2.4 Anomaly/Intrusion Detection

This task uses a principal component analysis (PCA) to build an intrusion detection model. The PCA is frequently used in exploratory data analysis because it reveals the inner structure of the data and explains the variance in the data (Ferreira et al., 2021). It analyzes data that contains multiple variables and looks for correlations among the variables to determine the combination of values that best captures the differences in outcomes. The importance of an intrusion detection model in machine learning is that it helps in learning patterns that show there is a network intrusion. It is quite difficult to get network security sample dataset for modeling and the algorithms used in this category have been designed solely to address the core

challenges of building and training models by using imbalanced data sets. The algorithm used to train an intrusion detection model is the RandomizedPCA. The evaluation metric is Area under ROC curve and values greater than 0.5 shows better efficiency.

The auto-ml tools and framework for each of these platforms is discussed below. This section discusses the different cloud platforms used in this study, where Auto-ML tools are utilized in them.

#### 6.2.5 Google Cloud Console (GCC)

This is a supervised machine learning platform that is used to identify relationships in each dataset that cannot be derived logically by browsing or observing the datasets. Auto-ML on Vertex AI enables you to build a code-free model based on the training data provided. Clustering is used to group categories of data into clusters that contain similar characteristics. A scenario whereby clustering is used is when understanding segments of class students based on habits and characteristics of lecture delivery. The K-nearest neighbor is used to train a clustering model. The evaluation metric for clustering is the average distance and a value close to zero is better.

#### 6.2.6 Microsoft Azure Console Services

When utilizing the Azure machine learning tools in the machine learning studio, workflows are optimized for better performance on the cloud service. The training and deployment of models are similar in process but differ in the different open-source framework embedded in the studio. Some of the frameworks it supports are Py-torch, TensorFlow, Scikit-learn, XGBoost, LightGBM, R, .NET, keras, Ray RLLib, Open Neural Network Exchange (ONNX) and the MLOps tools help monitor and deployment of models (Webber-Cross, 2014). Developers will find familiar interfaces such as Python SDK, Command Line prompt (CLI v2) and robust set of tools, backed by durable Azure Resource Manager APIs, for building advanced ML pipelines/workflows. When working in the Microsoft Azure cloud the security protocol is usually a role-based access control (RBAC) for the infrastructure (Webber-Cross, 2014). The integration of the Azure storage blobs, Azure SQL Database, users can import and automatically save the dataset for later referral. With Azure Synapse analytics, the processing and streaming of data with spark (Mohamed, 2017). When a machine learning workflow is ready for operationalization, users can automatically trigger a schedule or HTTPS request. Models are deployed to the managed “endpoint”, for both real-time and batch deployments (Rezazadeh, 2020). Automated ML (Auto-ML) speeds the process of data featurization and algorithm for training which is done in the studio of the Python SDK environment. Hyperparameter optimization is automated, and the results are visualized in the studio.

#### 6.2.7 IBM Watson Cloud

The machine learning process on the IBM Cloud is like Google and Azure cloud in some features. The platform is mostly used by high-end business companies for building analytical models and neural networks from a particular dataset and can be deployed for use in applications. Scalable open-source platform based on Kubernetes and Docker components are integrated on this cloud platform (D. Miller, 2019). It also supports opensource frameworks like Tensorflow, Scikit-Learn, SPSS, PMML, Hybrid Auto AI, XGBoost, Decision Optimization, Pytorch. An API is assigned to a user based on his location region (D. Miller, 2019). For faster deployment of predictive models, users in certain regions are privileged to have more resources to ensure quick training and deployment of models. When training your model, the hardware configuration setting is categorized based on the capacity type and capacity units per hour. The user gets to choose the setting that fits the computing needs. For all frameworks models can be deployed in batch type, where the job duration is called in seconds for the maximum number of nodes stated. While for the online deployment of models, only the AutoAI, SPSS, Scikit-Learn custom libraries, Tensorflow, RShiny, Spark, PMML, Scikit-Learn, Pytorch, XGBoost framework are supported (D. Miller, 2019).

### 6.3 Method

In this section, we discuss the dataset used, the full description of the dataset, the list of dependent variables and independent variables. Then we discuss the process involved in training a model for each of the cloud platforms discussed in this section.

#### 6.3.1 Dataset

For this experiment, the UNSW-NB15 dataset was used. The raw network packets of the UNSW-NB 15 data set were created by the IXIA Perfect Storm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviors. The training and testing dataset of the UNSW-NB15 dataset were both utilized in this experiment. The number of records in the training set is 175,341 records and the testing set is 82,332 records from several types of attack and normal (Moustafa & Slay, 2015). The dataset contains forty-nine attributes. The purpose or goal of this experiment is to predict attacks in this dataset using the different cloud platforms discussed above. The next step we took was data pre-processing, the dataset set been used here is a large security dataset. There are ten distinct types of attack namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. First, we evaluate the dataset, there are four discrete variables in this dataset which are denoted by the following attributes, Protocol, Service, State, and Attack\_category. These four columns contain categorical data, while the numeric variables are the attributes/columns which contain numerical value. The target variable is '*attack category*.', while the predictor variables or independent variables are (*Protocol, service, and state*). We also checked the number of occurrences for each attack in the attack\_category. There was a label attribute/column in the dataset. We used the Boolean mask to generate a Boolean array for the service and protocol attributes. Using the (.loc) function, this returns a series of true and false values for each row in the data frame. We set the value count for both the service and protocol, which is we set the values for protocol and service to be True when the label = 0, and False when the label = 1. The next step of the data cleaning was to convert the attack\_category multi-class categories to numerical values, using the Sklearn label encoder library to label each attack, i.e (0-9). This column was named Num\_AttackCat. Some of the attributes will not be necessary for our analysis, so we had to drop them. These attributes were the 'id,' 'attack\_category,' 'label.' Then we masked the ('protocol,' 'state,' 'service') data attributes to a floating object, then converted the columns to a list. This enabled us to import dummies, to turn our categorical columns ('protocol,' 'service' and 'state') into indicator columns. Now the dataset is ready for analysis. The algorithms which were utilized for modeling in this experiment are Decision tree, Random Forest, and Gradient Boosting Classifier. When performing Auto-ML on the cloud platform, it is important to clean the dataset and prepare it for the algorithms intended for use on them. The result obtained from this analysis would be wrong. This pre-processed dataset is what is read into the cloud storage and then, the machine learning needed to train this model is chosen or it can be set to Automatic. If set to automatic, the cloud machine learning system chooses the best algorithm to model and delivers an output (Moualla, Khorzom, & Jafar, 2021). In table 6-1 below, the data set statistics are provided which shows the classes of attacks, the training sample numbers, the training sample percentage, the testing sample percentage, the number of normal and abnormal attack records.

Table 6-1: Dataset Statistics of UNSW-NB15 showing the distribution of the classes of attacks.

Class type	Training samples	Training samples percentage	Testing samples	Testing samples percentage
Normal	56000	31.94	37000	44.94
Analysis	2000	1.14	677	0.82
Backdoors	1746	1.00	583	0.71
DoS	12264	6.99	4089	4.97
Exploits	33393	19.05	11132	13.52
Fuzzers	18184	10.37	6062	7.36
Generic	40000	22.81	18871	22.92
Reconnaissance	10491	5.98	3496	4.25
Shellcode	1133	0.65	378	0.46
Worms	130	0.07	44	0.05
<b>Total</b>	<b>175341</b>	<b>100</b>	<b>82332</b>	<b>100</b>

The features found in UNSW-NB15 comma separated value dataset are described in the tables 6-2 to 6-6 below. We categorized these features into four main groups. These groups are Flow, Basic, Content, Time features and additional generated features. These features include different packet-based features and flow-based features. The packet-based features are used to examine the payload beside the headers of the packets (Moustafa & Slay, 2015).

Table 6-2: Flow Features

#	Name	Type	Description
1	srcip	Nominal	Source IP address
2	sport	Integer	Source port number
3	dstip	Nominal	Destination IP address
4	dsport	Integer	Destination port number
5	proto	Nominal	Transaction protocol

Table 6-3: Basic Features

#	Name	Type	Description
6	state	Nominal	The state and its dependent protocol, e.g. ACC, CLO, else (-)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination bytes
9	dbytes	Integer	Destination to source bytes
10	sttl	Integer	Source to destination time to live
11	dttl	Integer	Destination to source time to live
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	Nominal	http, ftp, ssh, dns ..,else (-)
15	sload	Float	Source bits per second
16	dload	Float	Destination bits per second
17	spkts	Integer	Source to destination packet count
18	dpkts	Integer	Destination to source packet count

Table 6-4: Content Features

#	Name	Type	Description
19	swin	Integer	Source TCP window advertisement
20	dwin	Integer	Destination TCP window advertisement
21	stcpb	Integer	Source TCP sequence number
22	dtcpb	Integer	Destination TCP sequence number
23	smeansz	Integer	Mean of the flow packet size transmitted by the src
24	dmeansz	Integer	Mean of the flow packet size transmitted by the dst
25	trans_depth	Integer	the depth into the connection of http

			request/response transaction
26	res_bdy_len	Integer	The content size of the data transferred from the server's http service.

Table 6-5: Time Features

#	Name	Type	Description
27	sjit	Float	Source jitter (mSec)
28	djit	Float	Destination jitter (mSec)
29	stime	Timestamp	record start time
30	ltime	Timestamp	record last time
31	sintpkt	Float	Source inter-packet arrival time (mSec)
32	dintpkt	Float	Destination inter-packet arrival time (mSec)
33	tcprrt	Float	The sum of 'synack' and 'ackdat' of the TCP.
34	synack	Float	The time between the SYN and the SYN_ACK packets of the TCP.
35	ackdat	Float	The time between the SYN_ACK and the ACK packets of the TCP.

Table 6-6: Additional generated features

#	Name	Type	Description
36	is_sm_ips_ports	Binary	If source (1) equals to destination (3)IP addresses and port numbers (2)(4) are equal, this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get

			and Post in http service.
39	is_ftp_login	Binary	If the ftp session is accessed by user and password, then 1 else 0.
40	ct_ftp_cmd	Integer	No. of flows that have a command in ftp session.
41	ct_srv_src	Integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	I	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	I	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ltm	I	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm	I	No. of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).

46	ct_dst_sport_ltm	I	No. of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	I	No. of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).

Finally, the last two columns in the dataset are the labelled category. This attribute was labelled with the IXIA tool which generated a report about the attack data. This report is configured in the shape of the ground truth table to match all transaction records. This data set is labelled as listed in Table VII, attack categories (i.e., attack\_cat) and label for each record either 0 if the record is normal and 1 if the record is attack.

Table 6-7: Labelled Features

#	Name	T	Description
48	attack_cat	N	The name of each attack category. In this data set, there are nine categories (e.g., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms)
49	Label	B	0 for normal and 1 for attack records

### 6.3.2 Google Auto-ML

The Vertex AI embedded in the google cloud console (GCC) is the section used to conduct machine learning pipelines. Auto ML on Vertex AI enables you to build a code-free model based on the training data provided. The GCC accepts the uses of CSV or JSONL files and the entity extraction only supports JSON files. It is possible to select use values for the CSV file upload, to choose what aspect of the dataset would be used for training, testing and validation. If the use values are not specified, Vertex AI is programmed to automatically choose it. When running a sentiment analysis, the sentiment max value must be included in the last column of each row of the CSV file uploaded. Dataset created in the vertex AI section is uploaded to the cloud storage, and charges are collected for this storage. When deploying a model in Vertex AI, you create an “Endpoint” object which has in-built resources for serving online predictions. Then the model is



deployed to this endpoint, then you request prediction, by clicking “*Predict ( )*” on the vertex AI. Figure 4.1 shows the steps taken when the free-tier account is created

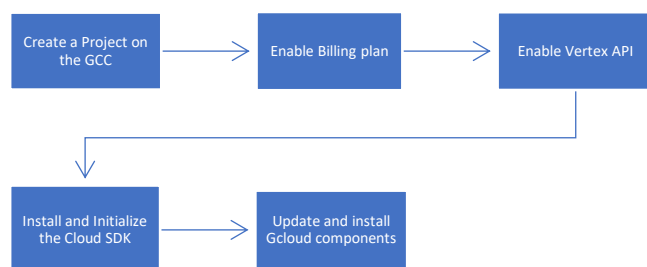


Figure 6-1 Creating and Enabling the Auto-ML Environment

### 6.3.3 Azure Auto-ML

Machine learning studio in Azure cloud allows users with limited code experience and high-level code experience to train and deploy machine learning models. Just like GCC, azure ML studio accepts CSV and JSON file formats. It supports data pre-processing, importing, validating, and cleaning, transformation and normalization. It also performs splitting of the data into testing, training, and validation (cross validation). To efficiently train and validate the data, the azure ML studio allows you to choose the different hardware compute engine, distributed processor, and progress monitoring all at a pay-as-you-use service. Deployment is done by creating an online endpoint to make predictions. Figure 4.2 shows the stages involved after creating an account on Azure cloud.

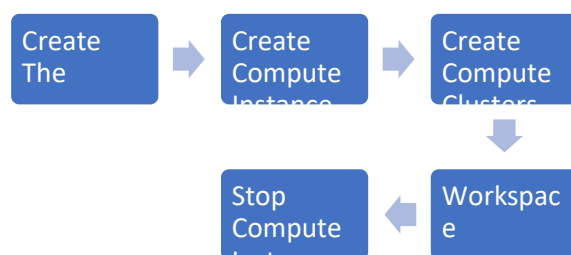


Figure 6-2: Process involved in setting up Auto-ML Environment

### 6.3.4 Watson Auto-ML

The process of training data on IBM is quite different. Firstly, the studio allows the users to load both data gotten from a sole source and data gotten from multiple sources. This is only if the multiple files all have a common key assigned to them. It accepts CSV files, with a minimum of one hundred rows, and in terms of classes or features, nothing more than 20. When creating a project in the Auto-ML experiment section, after loading the dataset set, either through notebook instance or directly uploading it from a file system, the target column to be predicted is selected. Then, based on the subset of the dataset, the platform automatically chooses a default model type and the default metric best suited for the model. The cross-validation that

occurs here divides the training data into folds or groups, to get the best performance model. The runtime settings should be carefully selected (i.e., number of hours, algorithms to be excluded). During the runtime of the experiment, a user can work on other aspects of the project. After the experiment, a detailed tab that contains the projects unique properties are shown, then the user could save the pipelines as a model. Figure 4.3 shows the steps involved after account creation

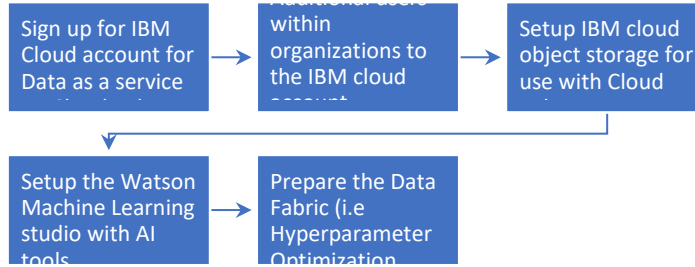


Figure 6-3: Creating the Auto-ML environment

Some of the Evaluating Metrics used in this study will be discussed here.

Mean Absolute Percentage error (MAPE): This measures the average of absolute percentage errors. The formula for calculating it is as follows:

$$MAPE = \frac{\sum \frac{|A-F|}{A} \times 100}{N}$$

Where A = Actual value. F= Forecast value, N= number of observations.

Root Mean Square Error (RMSE): this is the square root of the mean of the square of all of the errors gotten after training. The formula for calculating the RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=0}^n (S_i - O_i)^2}$$

Mean Absolute Error (MAE): this is a measure of errors between double observations expressing the same appearance. The formula used to calculate the MAE is as follows:

$$MAE = \frac{\sum_{i=0}^n |y_i - x_i|}{n}$$

## 6.4 Results

In this section, the prepared dataset is read/uploaded into each of the cloud storage platforms discussed above. The benefits of using a pre-processed dataset are because it reduces the risk of partial data being loaded into the pipeline and guarantees accurate values among the comparisons being made. All the Auto-ML tools mentioned were applied to the different datasets listed.

### 6.4.1 Google Auto-ML Result

The figures below are snapshots of the Google auto-mL cloud platforms during and after the building phase of the model and show batch prediction when deployed to an endpoint. Figure 5.1 shows the feature importance of the dataset after training of the dataset.

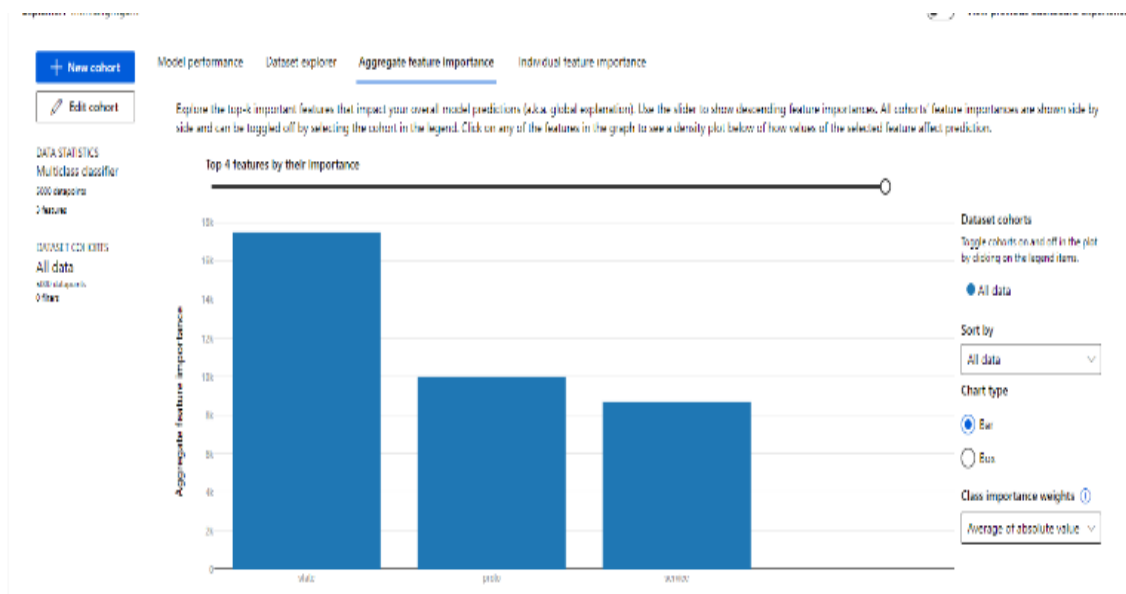


Figure 6-4: Feature importance and values derived after analysis

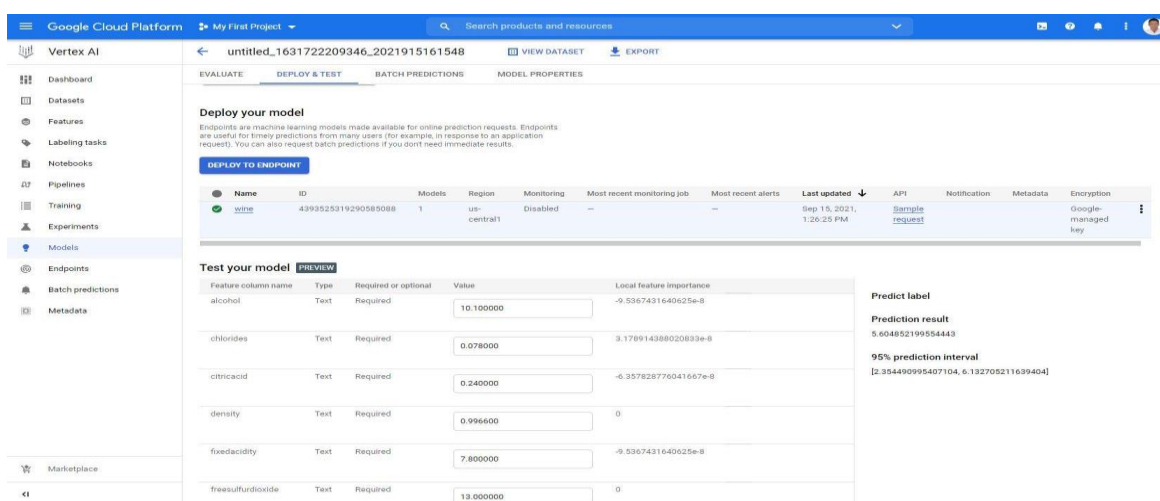


Figure 6-5: Endpoint for batch predictions

#### 6.4.2 Azure Auto-ML Result

When training the model in the Azure auto ML platform, the process as shown above in the methods section is different from the training pipeline in google cloud ML. The figures below show the dataset being uploaded via direct upload from the cloud storage or file repository. While figure 5.3 shows the configuration setting and the training and Figure 5.4 shows the completed run of the dataset training. The duration of the training took 40minutes, this was because of the billing plan I chose and the region where the computing resources were located. The accuracy gotten was 0.7201, the weighted AUC was 0.8926, while the weighted accuracy was 0.90698.

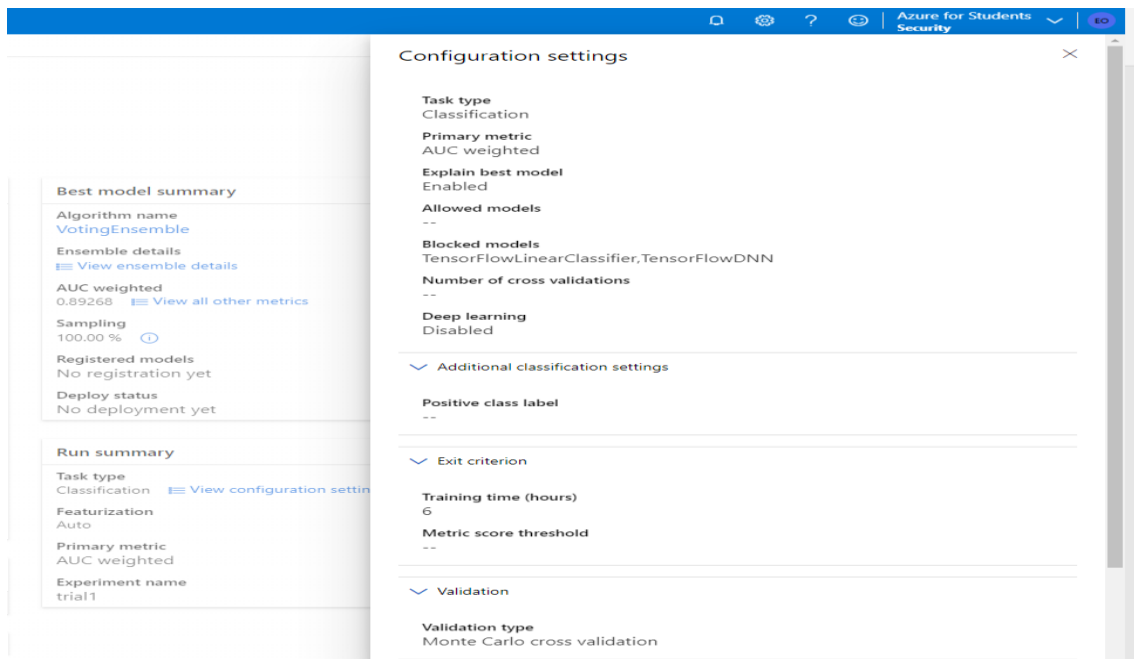


Figure 6-6: Configuration settings in the auto-ml environment

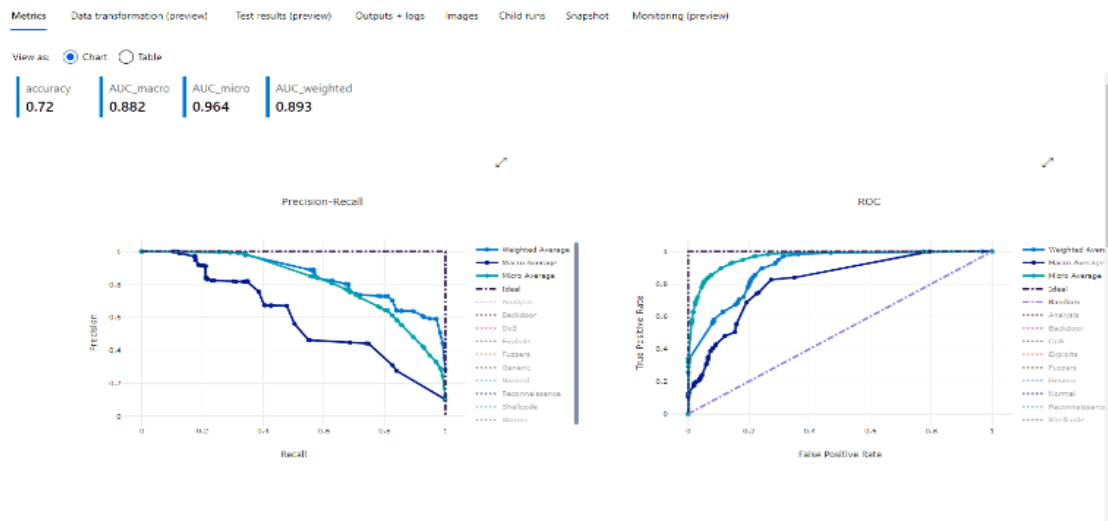


Figure 6-7: Completed experiment training showing the graph of the Precision-Recall and ROC

#### 6.4.3 IBM watson Auto-ML Result

When training the dataset in the IBM Watson studio, you can either use the Jupyter notebook to build your model, or you use a direct upload of the dataset to the Watson machine learning platform. When using direct upload, the dataset should be properly arranged in the excel spreadsheet. In this project we used both the training and testing dataset of the UNSW-NB15 network security dataset. In the experiment settings, once the two CSV files are read into the data storage in the cloud, the user is allowed to join both tables based on the corresponding attributes on the second table. The next step is choosing the prediction column, which is the attack category. The figures below show the steps taken to achieve results. The results achieved are summarized in the table below

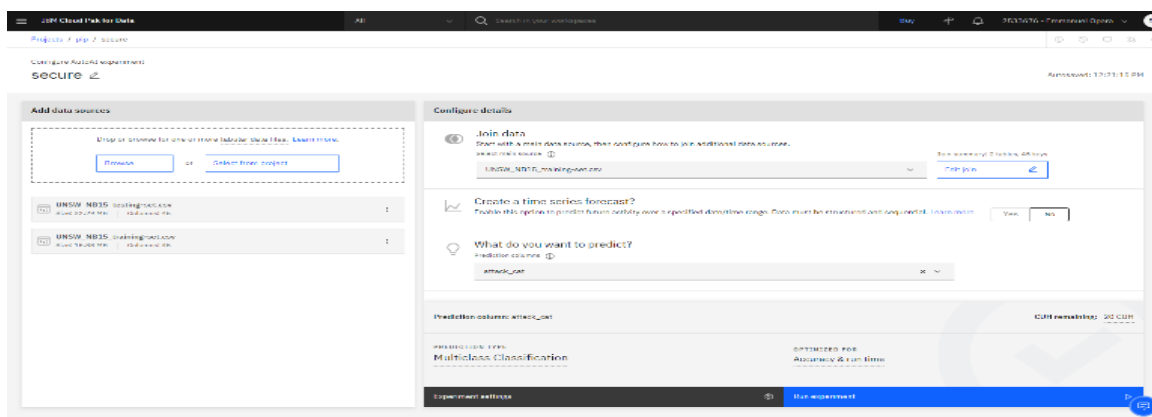


Figure 6-8: Experiment setting in the Auto-ML environment

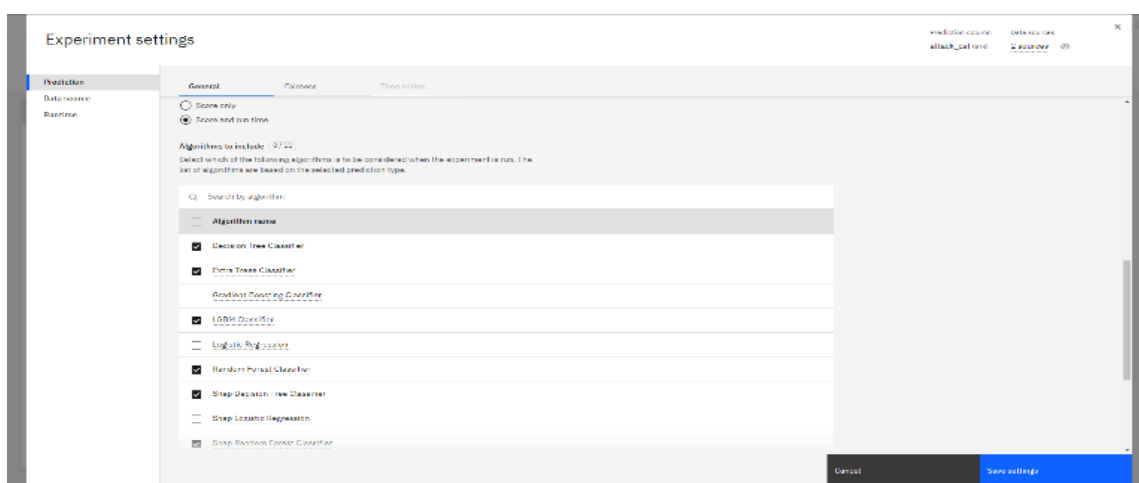


Figure 6-9: Algorithm selection for training in experiment environment

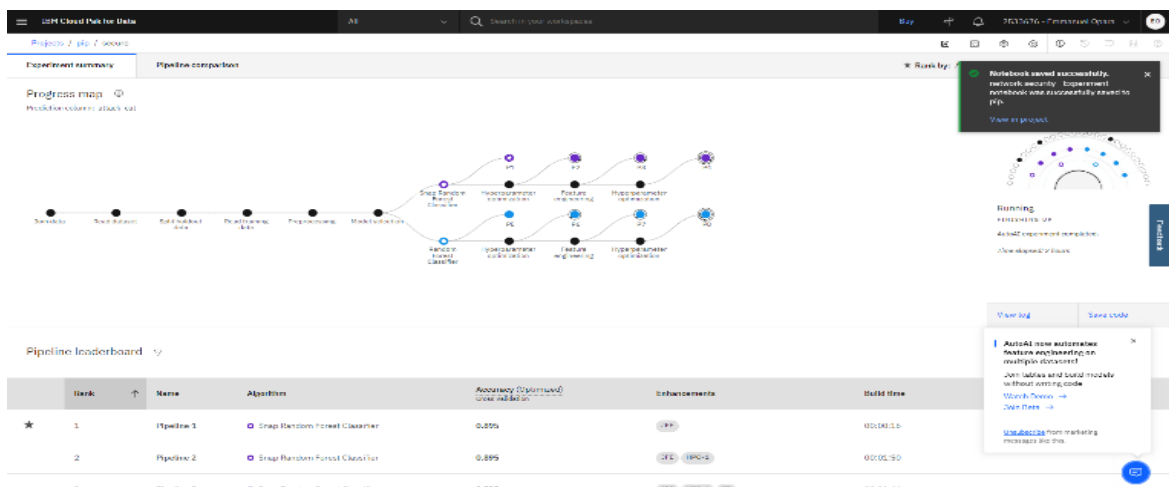


Figure 6-10: Completed training pipeline.

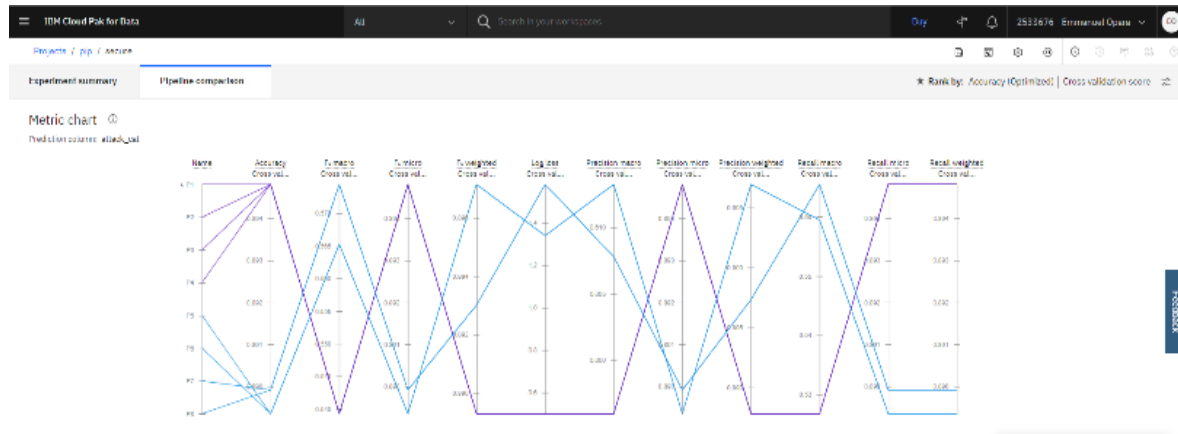


Figure 6-11: Metric comparison

Below is a summary table which shows the accuracy and Area under curve (AUC) weighted values gotten after the analyzes of the datasets through the different cloud platforms. It also shows the machine learning classifier used.

Dataset	Cloud Platform	Algorithm	Accuracy	AUC weighted
UNSW_NB15 Network security dataset	Azure Cloud Platform	Decision Tree Classifier	0.725	0.892
UNSW_NB15 Network security dataset	Google Cloud Console	Random Forest Classifier	0.885	0.890
UNSW_NB15 Network security dataset	IBM Watson Cloud.	Gradient Boost Classifier	0.895	0.899

Table 6-8: Algorithm and cloud platform performance

Below is a comparative summary of some of the features each of these cloud platforms provides. They are all different, in terms of how they process the data and develop the pipeline.

Table 6-9: Comparative Analysis

	Google Cloud Platform	Microsoft Azure Cloud	IBM Watson Studio
<b>Auto-ML supported Tools</b>	Py-torch, Tensor-flow, R, RAPIDS, XGBoost, MXNet, CNTK, Google Auto-ML	Py-torch, Tensor-flow, Scikit-learn, XGBoost, LightGBM, Ray RLLib, Keras, Azure Auto-ML	Tensorflow, Scikit-Learn, SPSS, PMML, Hybrid Auto AI, XGBoost, Decision Optimization, Pytorch
<b>Data types allowed</b>	Numerical, Categorical, Datetime, Time-series	Numerical, Categorical, Datetime, Time-series	Numerical, Categorical, Time-series, Image and Text
<b>Model selection and Hyperparameter optimization</b>	Genetic algorithm, Random search, Bayesian Search Neural Architecture	Ensemble, Random search, Bayesian Search.	Genetic algorithm, Random search, Bayesian Search Neural Architecture Ensemble
<b>ML Tasks</b>	Supervised Learning, Unsupervised Learning	Supervised Learning	Supervised Learning, Unsupervised Learning
<b>Model Evaluation/Results analysis visualization</b>	Model dashboard, Feature Importance, Model Summary,	Model dashboard, Feature Importance Model summary	Model dashboard, Feature Importance Model summary Metric comparison
<b>Feature Engineering</b>	Imbalance data, Feature Selection, Advanced feature extraction	Imbalance data, Categorical Processing, Feature Selection, Advanced feature extraction	Imbalance data, Categorical Processing, Feature Selection, Advanced feature extraction

## 6.5 Conclusion

With the amount of data generated from different sectors, such as healthcare, finance, planning and supply chain management, the need to process these large sums of data must be taken into consideration. With different open-source ML platforms there has been huge advancements in data analytics, but with the high expenses that will be incurred to get the computing resources to analyze these data, organizations and firms are restricted to spending that much. Automated machine learning on cloud platforms is the solution adopted by organizations to analyze their data, due to the availability of these resources which is provided as platform-as-a service. This means users pay for the computing resources needed to analyze any size of data. Three different cloud platforms were used to analyze the UNSW-NB15 network security dataset. After the successful completion of this experiment, we can observe that the best machine learning algorithm utilized here was the Gradient Boost Classifier with an accuracy score of 89.5%. These platforms have open-source libraries built in them, so it chooses the best framework to analyze the data and it processes it at a high speed.



## 7 Study 2 – Detection of Hate Speech on social media using Lexicon-based software.

### 7.1 Introduction

Automatic classification of textual content becomes the only practical method for effective data classification and insight. Although the ability to post comments empowers users to discuss news stories in a creative manner, discussion can also become toxic, leading to racist remarks and hate speech. Researchers have worked on sentiment classification on reviews and comments from social media and other news websites using machine learning techniques at the document level. The main objective of this research study is to investigate hate speech on the reddit platform using the lexicon-based approach. Reddit has its own principles and standards, organized around its communities called subreddits. Subreddits differ from each other in many ways, especially in four specific dimensions: topic, audience, moderation, and style. In our approach, the integration of two lexicon dictionaries was established. The two lexicon dictionaries integrated together were the Hatebase lexicons and the Vader lexicons. This integration created a new dictionary which was used to score comments extracted from four different subreddit and the values derived were used to compare the score derived only from Vader sentiment dictionary. Most studies on reddit comments and posts sentiment analysis are based on binary classification where the reviews are classified into “positive” and “negative.” Moreover, even the best systems currently obtain F1-score, precision, accuracy of only about 85%. There has not been as much work on sentiment analysis using lexicon-based techniques at the document level. However, recently there has been progress on building lexicons for sentiment analysis. Comparing the Vader lexicon-based technique and the integration of Hate base-Vader lexicon-based technique was conducted in this research work. The Vader sentiment lexicon is a library of pre-processed texts. It consists of words that have been categorized as either positive, negative, or neutral. Recently, it has been one of the most used lexicon libraries to perform sentiment analysis. The Hate base lexicon is a collaborative repository of multilingual hate words. It was developed to assist companies and research organizations moderate online conversations and use them as a hate speech predictor. In this work, we investigate top subreddit communities where there might be dissemination of toxic speech. We selected the subreddit communities based on a dataset that has been published in journals by different authors. We collected a large corpus of data, consisting of 100k comments for each of the subreddit used. The data collection process was systematically collected at different time intervals during this research. The results derived from the analysis show a significant difference between our integrated dictionary and the Vader lexicon library

### 7.2 Method

In this paper, the most popular microblogging platform Reddit is used. Just like twitter, reddit is a social media platform where users post about several topics. It has sub-categories called subreddit, this is for different discussions held on the social media platform. Some news websites also use social media platforms to pass across news, and users are allowed to comment on them. However, comment moderation remains an issue. The increasing magnitude of user-generated comment repositories and their continuing fast growth makes it very labor intensive to manually monitor and extract sentiment from user-generated content. The first step in any data analytics research is data collection. Data collection follows a certain procedure, this is a systematic process of extracting the right data needed for your analysis.

### 7.2.1 Data Collection

In this work, we investigate top subreddit communities where there might be dissemination of toxic speech. We selected the subreddit communities based on a dataset referenced by (Tan & Lee, 2015). This dataset contains posts and comments from the inception of reddit in January 2006 to December 2018. Some subreddits are topic specific (r/Nickelodeon, r/PBSkids, r/Disney, r/4chan), while others are from a general topic (r/AskReddit). All subreddits have rules regarding what types of content is allowable in that community, the specificity of the rules and the level of moderation differ from subreddit to subreddit. Even when the written standards are similar, reddit communities attract users with different interests and discussions of different nature (Tan & Lee, 2015). Similar data collection procedure was followed by (Yan & Liu, 2021), in their research the authors extracted the post and comment data from specific subreddit from August to November in 2019 and 2020, respectively, representing the pre-pandemic and pandemic periods, using the Pushshift API. Chen and Sokolova (2021) utilized the API to extract text posts from ‘r/depression’. The subreddit posts conveyed self-expressed contextual aspects of depression and provide a richer context for sentiment analysis than more treatment-oriented posts from r/Anxiety and r/PTSD, the other mental health Reddit subcommunities (Chen & Sokolova, 2021). The PushShift Reddit API was used to extract the comments from the necessary subreddits we needed. The reason for using this API is because it has been utilized in previous research work. The PushShift API was designed and created by the /r/datasets mod team to help provide enhanced functionality and search capabilities for searching Reddit comments and submissions. It provides over four billion comments and submissions since the inception of Reddit. (<https://github.com/pushshift/api>). There are two main endpoints of the API, one of them is for extracting only the comments, while the other is for extracting the submissions. The comment search parameter can be time-based, the reason is that the API allows users to set a period of when the comments will be collected. This function can be activated when writing the code. This is done by using an epoch time for the value of the “after” and “before” parameter, it will return all comments with a created\_utc epoch time greater than that value. The comment search can be extracted by hour of the week, this parameter itself can be used directly to limit comments by a specific hour of the week as well. The range is from 0 to 168 with zero being midnight on Monday and 168 being the 23<sup>rd</sup> hour of Sunday night. The comments can be extracted by hour of the day, using this parameter as an aggregation type, it shows when a subreddit or author is active throughout a typical day (Baumgartner, Zannettou, Keegan, Squire, & Blackburn, 2020). We collected a corpus of 100k comments from each of the subreddit mentioned above. This data came in large files that were put into a csv using panda data frame in python. These comments extracted for analysis were extracted using the *pmaw* package in python. This package ensured an enormous collection of comments at once. We utilized the function after and before for extracting the comments because it was mostly historical comment data. We extracted from the inception year May 2005 to May 2022. The extraction time for each subreddit extraction was about 30minutes. Table 7-1 below shows the description of each column in the dataset extracted for each subreddit.

Table 7-1: Dataset Description

Attribute	Description
Reddit ID tag	This is the user’s unique identification tag number
Comments	This is the user’s generated comment.
Reddit category	This is the category in reddit we selected. For instance, News, comedy, politics category

Subreddit category	This is the sub-category under each category. For instance, under news category, we have conservative news comments, democratic new comments.
Date created	This represents the time the comments were created.
Reddit username	The user's unique username is also extracted along with the comments.

### 7.2.2 Data Pre-processing

In this research study, after collecting a huge corpus of data for each subreddit, the data needs to be arranged and cleaned before further text processing is performed. The data drop function in the panda data frame will be applied to the comment data in the csv. The data drop function applied drops the attribute columns not needed for the analysis. Figure 1 shows source code written in python for the execution. Due to the large amount of data the cleaning process took 1min 30sec to complete each corpus. Figure 2 provided below shows the extracted comments for one of the subreddit and figure 3 shows the source code used to clean the data.



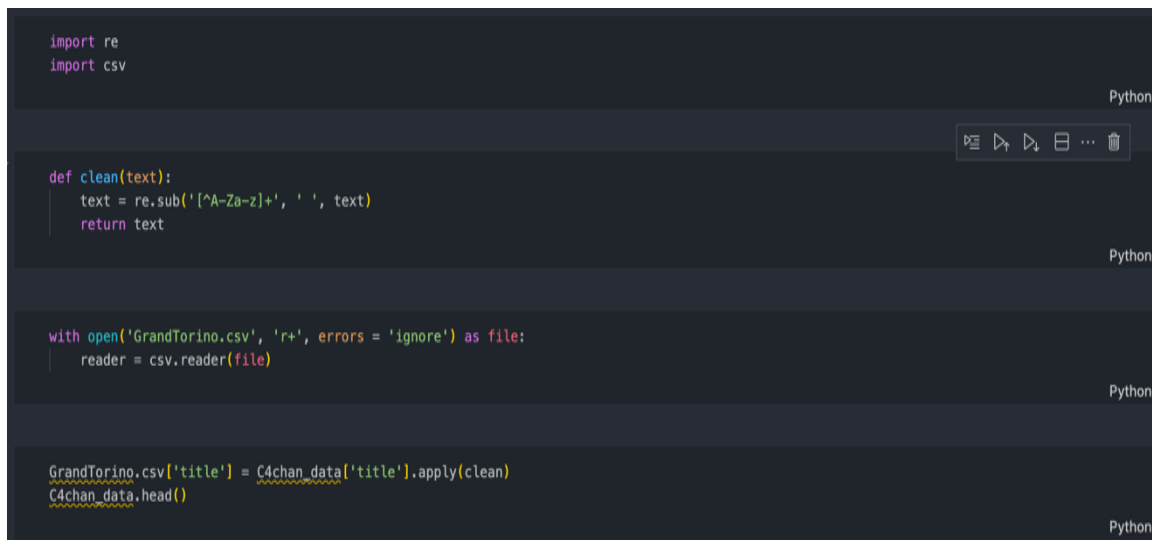
```

[41] CNNDData.to_csv('NickelodeonComments.csv', index=False)

[86] C4chan_data = C4chan_data.drop('body', axis=1)
      C4chan_data.head()

```

Figure 7-1: Data-drop function



```

import re
import csv

def clean(text):
    text = re.sub('[^A-Za-z]+', ' ', text)
    return text

with open('GrandTorino.csv', 'r+', errors = 'ignore') as file:
    reader = csv.reader(file)

GrandTorino.csv['title'] = C4chan_data['title'].apply(clean)
C4chan_data.head()

```

Figure 7-2: Source code for cleaning comment data

### 7.2.3 Vader Lexicon and Scoring Mechanism

Vader sentiment analyzer is a lexicon library which can be installed on python as a module. It is the most common sentiment analysis tool for social media text and text data. This module is based on the lexicon and rule-based methods. We utilized Vader sentiment analyzer because it is fast and accurate, it gives the score of a comment as either “negative,” “positive” or “neutral.” The compound score of Vader is a normalized score between 1 and -1 which is obtained by adding the valence scores of each category score mentioned above and is adjusted depending on the rules. The scoring mechanism of the compound score is explained below

- $0.05 < \text{Compound score} < -0.05 = \text{Neutral Sentiment.}$
- $\text{Compound Score} \geq 0.05 = \text{Positive Sentiment}$
- $\text{Compound Score} \leq -0.05 = \text{Negative Sentiment}$

### 7.2.4 Vader Lexicon Integration with Hate base Lexicon

In this section, the integration of two dictionaries is the novelty of this research project. The lexical approach to sentiment analysis aims to map words to sentiment by building a ‘dictionary of sentiment.’ We used this dictionary to score the sentiment of phrases and sentences, without the need of looking at anything else. The hatebase dictionary contains a corpus of hate words in English that is regularly updated on their website. The hatebase website is a repository of multilingual hate speech. It was developed by the Dark Data project in partnership with the Sentinel project. It was built to assist researchers, organizations and government agencies moderate online conversations and can be used as a predictor for hate speech. It contains over 3894 terms in 98 languages. It is used by various organizations around the world (Dark Data Project). We utilized extraction packages in python with the Hatebase API to extract these words and their allocated scores into a notepad. Then we created another dictionary which we named “VaderHatebase” dictionary. This dictionary was separate from the original Vader dictionary which contained its own text words and scores.

### 7.2.5 Research Hypothesis

In our effort to contribute to the detection of hate speech on social media platforms, this research study seeks to construct a new sentiment dictionary, which was the integration of both the Vader lexicons and the hatebase lexicons and compare the difference in detection among the subreddit chosen for analysis. To further evaluate our dictionary integration, we scored comment texts from two other blog sources and performed t-test on them. The goal of this research study is to compare the sentiment scores derived from both analyses of the different dictionaries. We are trying to prove there is an extra level of detection of hate speech when using our integrated dictionary. In the method section, the tables there show a t-test of the two dictionaries for each subreddit. The null hypotheses (H0) and corresponding alternative hypotheses (H1) are shown below:

H0: There is a difference between the Vader lexicon dictionary score mechanism and Vader+Hatebase dictionary score mechanism the vader+hatebase integration will provide a better score

H1: The Vader + hatebase integration is a lower sentiment score

### 7.2.6 Preliminary Hypothesis Testing

When comparing large corpus of data such as comparing the difference of the means from two separate subreddit, a good method to evaluate the significance is by undergoing preliminary testing with a smaller

size of the dataset. This is also called hypothesis testing (Dietrich, 2015). Forming an assertion and evaluating it with data is the basic concept of hypothesis testing. The common notion when performing hypothesis testing is that there is no difference between two samples. This hypothesis is used for constructing either the null hypothesis (H0) or the alternative hypothesis (H1). The two possible outcomes of a hypothesis test are either Null or Alternative. After the test was conducted, we could either reject the null hypothesis and accept the alternative hypothesis (which would mean that there is a difference between two samples) or we could accept the null hypothesis, which means there is no difference between samples. The result from the t-test is known as t-statistic. The t-test was utilized in this research to compare a set of subreddit groups as it asserts that the data sets came from different subreddit with unequal variances, and it is used to determine whether the two samples are likely to have come from distributions with equal comments means. A two-sample t-test is used when there are distinct subjects in the two samples. The following equation 1 below is used to determine the statistic value t'

Equation 1:

$$t' = \frac{x - y - \Delta_0}{\sqrt{\frac{S_1^2}{m} + \frac{S_2^2}{n}}}$$

If each comment is normally distributed with the same variance and with the same mean ( $\mu_1 = \mu_2$ ), then the t-statistic t, in Equation 2, follows a t-distribution with  $n_1 + n_2 - 2$  degrees of freedom (df).

Equation 2:

$$T = \frac{x_1 - x_2}{sp \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

The t-test for paired two sample means was performed on a sample of comment data extracted from each of the subreddit analyzed in this research. The first sample was the total sentiment scores for Vader lexicon dictionary analysis and the second sample was the total sentiment score for Vader+Hatebase lexicon dictionary.

### 7.3 Results

This section shows the result of the data analysis process performed in the methods section. Table 1 below shows the sum and average of the sentiment score after analysis of comments in the Disney subreddit class. Both scores obtained from the Vader scoring mechanism and Vader-Hate Base scoring mechanism were analyzed. The same process was applied for the following subreddit class which are, Nickelodeon comments, 4chan comments, PBSkids comments. To evaluate the difference in both scoring mechanisms, we employed the statistical sum and average of sentiment scores obtained from each class to further prove that the scoring mechanism developed has a good detection rate and significance. From Tables 2,3,4, we observed that all values for both the sum and averages are positive except for 4chan subreddit. This indicates that there is potential hate speech in this subreddit class.

**Disney Comments**

	Vader Scores	VaderHatebase Scores
Sum	32370.91	29867.67
Average	0.32	0.24

Table 7-2: Sum and Average of Disney sentiment scores

**4chan Comments**

	Vader Scores	VaderHatebase Scores
Sum	-2510.48	-5430.72
Average	-0.02	-0.05

Table 7-3: Sum and Average of 4chan sentiment scores

**Nickelodeon comments.**

	Vader Scores	VaderHatebase Scores
Sum	8429.54	6553.21
Average	0.08	0.06

Table 7-4: Sum and Average of sentiment scores

**PBSkids comments**

	Vader Scores	VaderHatebase Scores
Sum	7395.39	5581.61
Average	0.07	0.06

Table 7-5: Sum and Average of sentiment scores

To determine significance between the Vader scoring mechanism and the VaderHatebase scoring mechanism, we performed statistical analysis of paired sample t-test using SPSS. We utilized the 4chan subreddit class scores for the paired test. Table 5 below shows the Paired sample t-test, which is testing for both the Vader scores and VaderHatebase score for the 100k comments processed through our approach

4chan Paired Samples Test									
		Paired Differences					t	df	Sig. (1-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	VaderHatebase - Vader scores	-0.03	0.42	0.00	-0.03	-0.02	-27.47	166770	0.000

Table 7-6: Paired samples t-test results

In this study, we utilized only the  $P(T \leq t)$  one-tailed of the test because in the study conducted, we were analyzing the detection rate of our own curated dictionary and we evaluated the significance of this dictionary with comments extracted from four sources of subreddit. Among these four sources, three of

them can be classified as non-hateful comments speech, this is because the subreddit communities were mostly discussion held by users with non-hateful intention in their speech. The last subreddit used contains hateful speech which was where we analyzed the results derived from both our dictionary analysis and found significance, the p-value=  $p < 0.05$  for some of the sample sets evaluated. To further prove the efficiency of our approach, we extracted two additional comment datasets and analyzed the comments using Vader sentiment analyzer and the VaderHatebase sentiment analyzer. The sum and average of both dictionary evaluations is shown below. The datasets extracted were (1) Quotes from Gran Torino movie. The comments contained both hateful speech and regular sentences, but hate speech was existence in this comment dataset. This dataset contained 118 comments used in the movie. (2) The Wikipedia ethnic slurs comment dataset, which was the second comment extracted, contains hate speech words. The dataset contained 558 ethnic slur words. Table 6 and Table 7 below show the sum and average of the sentiment scores obtained after analysis. This two comments dataset contains hate speech because the sum and average were negative.

### Wikipedia ethnic slurs comments

	Vader Scores	VaderHatebase scores
Sum	-3.76	-6.12
Average	-0.0067	-0.27

Table 7-7: Sum and Average of sentiment scores

### Grand Torino Comments

	Vader Scores	VaderHatebase Score
Sum	-11.13	-14.35
Average	-0.095	-0.12

Table 7-8: Sum and Average of sentiment scores

From the results derived from the sum and average of the Vader sentiment score and VaderHatebase score, we performed statistical analysis to show there is a significant difference in both dictionaries. The t-test for Wikipedia slurs and Grand Torino comments is shown in tables 8 and 9 respectively below

Wikipedia Ethnic Slurs Paired test									
		Paired Differences					t-stat	df	Sig. (1-tailed)
		Mean	Variance	Pearson Correlat ion					
					P(T<=t) one-tail	T critical one-tailed			
Pair 1	VaderH atebase - Vader scores	-0.265	0.117	0.064	0.00	1.64	17.89	1116	0.000

Table 7-9: Paired sample t-test results for Wikipedia slurs

Gran Torino Paired test				
	Paired Differences			
			t-stat	df

		Mean	Variance	Pearson Correlat ion					Sig. (1- tailed)
					P(T<=t) one-tail	T critical one- tailed			
Pair 1	VaderH atebase - Vader scores	-0.03	0.02	0.94	0.01	1.65	2.21	116	0.01

Table 7-10: Paired sample t-test results for Gran Torino

The results derived from the statistical analysis was evaluated based on the  $P(T \leq t)$  one-tail because its directional. We were proved that our curated dictionary has a better hate detection mechanism. Just like the 4chan dataset the test was used to show that there was a significant difference in both dictionaries.

#### 7.4 Conclusion

Different countries have regulations and polices against hate speech in society. This attention raised the need for automating the detection of hate speech. In this study, we demonstrated the detection of hate speech on Reddit. There have been different hate speech detection mechanisms established by researchers. There has not been as much work on sentiment analysis using lexicon-based techniques at the document level. However, recently there has been progress on building lexicons for sentiment analysis. Our focus was integrating two lexicon dictionaries to improve the detection of hate speech on social media platforms. The two libraries that were integrated were the Vader sentiment dictionary and the Hatebase dictionary. We used the Hatebase API to extract the words into a document and integrated it with the words in the Vader dictionary. After the integration, the python code was written to score the comments extracted from different subreddits and two separate comments dataset to prove that our dictionary integration is better at detecting hate speech. The results for “4chan comments” show that our dictionary integration has significant difference in detecting hate speech, while the “Grand Torino” and “Wikipedia slurs” comments were further used to prove our approach. This approach used in this paper can be integrated into social media platforms to help detect hate speech. It could also be integrated into customer review sections on websites to detect offensive or hateful words from customers when they write a review. It can also be integrated into online workplaces platforms.



## 8 Study 3 – Sentiment Analysis using Cloud-based tools and Lexicon-based Software.

### 8.1 Introduction

Machine learning techniques on cloud platforms offer contemporary artificial intelligence services, with custom models and identified patterns to give accurate predictions. Opinions and comments made by individuals are important in cases where the results of the analysis are used to drive financial decisions made by business executives. In some cases, inappropriate comments made on social media platforms can result in hateful speech or potential threat to society. Cloud-based sentiment analysis is concerned with the automatic extraction of sentiment related information from text. In this research study, we extract and investigate comments from a social media platform to predict hateful sentiments in these comments. This analysis is performed using both cloud-based tools and open-sourced tools. The google cloud platform is utilized as the cloud-based tools in this study and python scripting language and jupyter notebook which is installed on visual studio code are the open-source tools used. The objective of this study is to perform sentiment analysis using cloud-based services that are at the core of modern business interactions and understanding the operating principles of open-source tools in text data collection, text pre-processing and data visualization.

### 8.2 Background

To achieve the objective of this study, we utilize the Natural Language Processing and Auto-ML natural language on Google Cloud Platform(GCP), Microsoft Azure Cloud and Amazon comprehend. For the open-source tools, the Natural Language Toolkit (NLTK) library is installed as a web package on python for text tokenization, tagging, cleaning, and developing a model for training and testing of text data. Finally, to further extend the study, we analyze these comments using the Vader lexicon dictionary to predict the sentiments in these comments. Cloud-based analysis provides prebuilt models that can be invoked as a service allowing developers to perform sentiment analysis, along with other features like entity analysis, content classification, and syntax analysis. In this paper, using the application programming interface(API) we extract comments from reddit social media, python scripting language will be used in the data collection process. The performance of the model on the cloud platform and the model developed using nltk in python will be compared and evaluated. The dataset utilized in this research has been carefully preprocessed removing stopwords, punctuation, and errors. Firstly, we create a project on the platform, then enable the Google Natural Language API, create a service account to perform syntax analysis and set up my virtual python environment for executing sentiment commands. This process takes careful implementation on the cloud. This is because not all services are included in a free tier account, so understanding what a user is entitled to is very essential. Using the Visual Studio Code(VSC), the nltk library is installed as a web package in python. The program codes are written to install each package to the system and install the vader lexicon dictionary for further analysis. In the cloud platform, the comments are scored based on the toxicity and polarity of words mentioned in the sentence. The comments are scored positive, neutral, and negative for each level of toxicity found.

### 8.3 Method

In this paper, the comments utilized was created based on quotes from the movie called “*Gran Torino*.” There are different platforms where individual’s opinions could be extracted from shopping websites to social media platforms which can be utilized to perform natural language processing(NLP) and data mining research. We investigated the scoring mechanism of various cloud analysis services (*Microsoft Azure*,

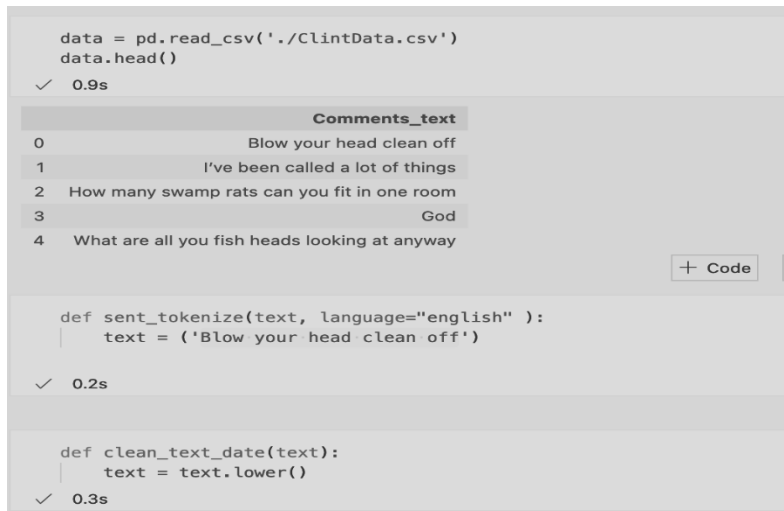
Amazon, and Google Cloud Platform) and the leading lexicon-based software called Vader sentiment analysis. The analysis method for each of these services and software were discussed in detail in this study.

### 8.3.1 Data Research

In this work, we investigated a comment dataset based on the quotes from the movie called “*Gran Torino*” The dataset contained two hundred movie reviews in the format of text. These comments contained both negative and positive sentences. Some of the comments were duplicated and contained stop words and punctuation marks as well as removing URLs and emails. Similar data collection procedure was followed by (Dave, Lawrence, & Pennock, 2003). After obtaining the data, we perform data cleaning. The data drop function in python was applied to the comment data in the comma-separated value (csv)file. The data drop function applied drops the features in columns not needed for the analysis. The comments dropped down to eighty-five unique comments, duplicates were discovered in the dataset. Figure 1 below shows a snippet of the code used for data cleaning. In the dataset, there was only the quotes column. This column contained quotes from the movie. That was the only column or attribute in the dataset. The table below shows the data description of the movie quote dataset

Table 8-1: Dataset Description

Attribute	Description
Movie Quotes	This represents every quote extracted from the movie. These quotes were scored, and the sentiment score was evaluated.



```
data = pd.read_csv('./ClintData.csv')
data.head()
✓ 0.9s
```

	Comments_text
0	Blow your head clean off
1	I've been called a lot of things
2	How many swamp rats can you fit in one room
3	God
4	What are all you fish heads looking at anyway

```
def sent_tokenize(text, language="english" ):
    text = ('Blow your head clean off')
✓ 0.2s
```

```
def clean_text_date(text):
    text = text.lower()
✓ 0.3s
```

Figure 8-1: Data cleaning in the command line interface.

### 8.3.2 Lexicon-based Software Sentiment Analysis

The lexicon-based software(Vader) functions with the linguistic word dictionary which is used to classify through words and phrases in a text. It returns scores on a wide range of variables, from percentage of words over 6 letters and use of pronouns, to informal language markers and psychological construct (Pennebaker, Chung, Ireland, Gonzales, & Booth, 2007). It leverages the parsimonious rule-based modeling to construct a computational sentiment analysis engine which works well on social media style text, and readily generalizes to multiple domains. Therefore, this analysis engine requires no training data, but it is

constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon and does not severely suffer from a speed-performance tradeoff (Hutto & Gilbert, 2014). The VADER sentiment analyzer has a similar correlation co-efficient ( $r = 0.881$ ) as compared to individual human raters ( $r = 0.888$ ) at matching ground truth (Hutto & Gilbert, 2014). VADER has become a common sentiment analysis tool for social media text and text data. We utilized Vader sentiment analyzer because it is fast and accurate, it gives the score of a comment as either “negative,” “positive” or “neutral.” The compound score of Vader is a normalized score between one and -1 which is obtained by adding the valence scores of each category score mentioned above and is adjusted depending on the rules. The scoring mechanism of the compound score is explained below

- $0.05 < \text{Compound score} < 0.05 = \text{Neutral Sentiment}$ .
- $\text{Compound Score} \geq 0.05 = \text{Positive Sentiment}$
- $\text{Compound Score} \leq -0.05 = \text{Negative Sentiment}$

The positive, neutral, and negative scores are ratios for proportions of text that fall in each category, and they add up to be one or close to the float operation. Figure 2 below shows the source code used in the analysis and the loading of the data into data frames.

```

3  import csv
4  import pandas as pd
5  import os
6  from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
7  from os import path
8
9  # initializing all arrays to hold data for processing
10 comment_text, neg_sentiment_value, neu_sentiment_value, pos_sentiment_value, compound_value, overall_result = ([[]] for
11
12 # opening file in read mode
13 with open('CLEAN CLINT DATASET.csv', 'r+', errors = 'ignore') as file:
14     reader = csv.reader(file)
15     for row in reader:
16         comment_text.append([row[0]])
17
18 # initializing analyzer variable
19 analyzer = SentimentIntensityAnalyzer()
20
21 # analyzing each comment and putting respective scores into own arrays
22 for i in comment_text:
23     vs = analyzer.polarity_scores(i)
24     neg_sentiment_value.append(vs['neg'])
25     neu_sentiment_value.append(vs['neu'])
26     pos_sentiment_value.append(vs['pos'])
27     compound_value.append(vs['compound'])
28

```

Figure 8-2:Source code for vader sentiment analysis

### 8.3.3 Cloud-based Sentiment Analysis

Different cloud providers offer services for Natural Language Processing. These services ranges from computer vision, speech recognition to sentiment analysis. The fequently asked question is if these services can be utilized in real-life conversational data and how accurate and precise can they interpret human sentiments in conversations. In this paper we investigate the scoring mechanism and performance of three major cloud providers. These cloud providers are; Microsoft Azure, Amazon, and Google Cloud Platform.

#### 8.3.3.1 Amazon Comprehend

This service uses natural language processing to extract sentiment about a text on a document-based level. That is, this service analyzed each sentence in the column as a document and returned a score. The

document-based analysis can be done using the console or using the comprehend APIs. The comprehend API was utilized in this study. The input document analyzed was in the UTF-8 format. To use this API, a secret client key is generated which is then exported into the terminal editor in python. Using the command (*export\_api\_key= <YOURAPIKEY>*) in the terminal editor, the comprehend API is activated (Guo et al., 2020). The next step is to read the comment-document using the read file command for python, then applying the sentiment analysis command to derive a score. Sentiment analysis returns the scores in the following categories:

Positive – The text expresses an overall positive sentiment.

Negative – The text expresses an overall negative sentiment.

Mixed – The text expresses both positive and negative sentiments.

Neutral – The text does not express either positive or negative sentiments.

Each categories returns a normalized score between the range of 0 to 1, but it returns in percentage and the sum of all categories sum up to 100%. Amazon comprehend already applies a classification algorithm to the sentiment analysis tool (Wickham, 2018). The MXNet deep learning classification algorithm is already applied to the process before a score is returned. This proprietary framework has been adopted by the other cloud providers discussed in this study. It is a highly scalable framework, which allows fast model training, and it supports a flexible programming model and multiple languages. The MXNet library is portable and lightweight. MXNet supports programming in various languages including Python, R, Scala, Julia, and Perl. This also leaves them susceptible to systematic bias (Guo et al., 2020).

#### 8.3.3.2 Azure Text-Analytics

In Microsoft Azure cloud platform, the natural language processing service responsible for sentiment analysis is called Cognitive service. It offers different machine learning algorithms which is combined and utilized in the interpretation of text to provide sentiment. The sentiment analysis feature scores the sentences and returns sentiment labels such as "negative", "neutral" and "positive" based on the highest confidence score found in the sentence and document-level (Harfoushi et al., 2018). This feature also returns scores between 0 and 1 for each document-level & sentences within it for positive, neutral and negative sentiment. Created a password key and an endpoint URL which was utilized to authenticate the API requests. Using the Spark table, we exported the data into a csv and stored it in the Azure Synapse storage. The selected algorithm used to score the text, is a pre-trained model by Azure. The system returned scores for the different texts found in the CSV file. After analysis, the text was categorised into three classes, which was positive, negative and neutral.

#### 8.3.3.3 Google Cloud Natural Language

Just like the other cloud providers, the natural language processing API can be utilized using the python code. The python environment was created using the application credentials, which involved a client key and an endpoint URL which enables requests from the TextAnalytics API. The document format of the text should either be in HTML or plain-text. To evaluate each content of text in the comma-separated value(csv) file, we converted the csv into a text document and included directly in the JSON request in the editor terminal. The sentiment analyzer in this service returns a numerical score and a magnitude value. The numerical score is the sentiment value of the text, and it returns sentiment in the ranges between -1(negative) and 1(positive). The magnitude indicates the overall sentiment strength within the text. It returns a score

between the ranges of 0.0 and +inf . The figure 3 below shows the returned score of a text in the dataset. It returns both the magnitude and overall sentiment score.

```
ecopara97@cloudshell:~ (thematic-nature-365119)$ curl "https://language.googleapis.com/v1/documents:analyzeSentiment?key=${API_KEY}" -s -X POST -H "Content-Type: application/json" --data-binary @request.json
{
  "documentSentiment": {
    "magnitude": 0.5,
    "score": -0.5
  },
  "language": "en",
  "sentences": [
    {
      "text": {
        "content": "Ever notice how you come across somebody once in a while you shouldn't have fucked with That's me",
        "beginOffset": 0
      },
      "sentiment": {
        "magnitude": 0.5,
        "score": -0.5
      }
    }
  ]
}
```

Figure 8-3:Sentiment score in command line

#### 8.3.4 Data Normalization

During analysis, the scoring mechanism of both the lexicon-based software and Google natural language processor returns scores in the ranges of -1 (negative) and +1(positive). Azure Text-Analytics and Amazon comprehend return scores in the ranges of 0 and 1 for each of the different categories. To ensure we were making the right comparisons, the scoring ranges of both the lexicon-based software and the cloud providers were normalized to the range of 0 and 1. We applied a normalization formula to the scores obtained from Vader Sentiment analyser and Google Natural language Processor. The formular can be found in equation 1 below;

Equation 1:

$$\frac{(X - X_{minimum})}{Range\ of\ X}$$

Where  $X$  = Compound score;

$X_{minimum}$  = -1(Negative);

Range of  $X$  = the total range value of the compund score which is (-1,1)

After normalizing the data in the three categories in the lexicon-based software(Vader), we converted the scores to percentage to match the scores in Amazon comprehend and Azure Text-analytics. To achieve this, the normalized score in the negative cateogory was divided by the sum of all normalized score in the different categories. Then, the t-test was utilized in this study to compare the normalized scores in the different categories of Amazon comprehend and Vader. Two-sample t-test is used when there are distinct significance in the two samples. it is used to determine whether the two samples are likely to have come from distributions with equal comments means. In tables 1 to 4, each t-test for the different categories are shown and explained. The equation 2 below is used to determine the statistic value t'. To establish significance, the mean difference must be less than 0.05. That is  $p < 0.05$  shows there is significance between the two groups.

Equation 2:

$$t' = \frac{x-y-\Delta_0}{\sqrt{\frac{s_1^2}{m} + \frac{s_2^2}{n}}}$$

If each comment is normally distributed with the same variance and with the same mean ( $\mu_1 = \mu_2$ ), then the t-statistic  $t$ , in Equation 3, follows a t-distribution with  $n_1 + n_2 - 2$  degrees of freedom (df).

Equation 3:

$$T = \frac{x_1 - x_2}{sp \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

Finally, to compare the scoring mechanism of lexicon-based software with the different cloud service providers, we performed an Analysis of Variance (ANOVA) test. This test is utilized to check if there are any statistical differences between the means of three or more independent groups. If there are no statistical difference found between the groups, the F-ratio should equal to 1. The equation 5 below shows the formula used to calculate Analysis of Variance between groups. This analysis was conducted using SPSS. This is a tool used for descriptive analysis. Equation 4 shows the formula for sum of squares which is then used to derive the F-ratio used in this study.

Equation 4:

$$SST = \sum_{j=1}^n (\bar{X}_j - \bar{X})^2$$

Where;

MSR = Regression mean square (MSR = SSR/df)

MSE = Error mean square (MSE = SSE/df)

SSR = regression sum of squares

SSE = error sum of squares

df = degree of freedom

SST = Total Sum of squares (SST = SSE + SSR)

MST = Total regression mean square  $\left(\frac{SST}{(p-1)}\right)$

p = the p-value that corresponds to  $F_{df}$

Therefore Equation 5;

$$F\text{-ratio} = \frac{MST}{MSE}$$

### 8.3.5 Research Hypothesis

The objective of this research study was to compare the sentiment scores derived from both lexicon-based software and cloud-based services. We established evidence that there is a significant difference between the analysis of comments at document-based level and using a lexicon-based dictionary. We established evidence that there is a significant difference between the analysis of comments using cloud-based services at document-based level. We established evidence that there is a significant difference in each sentiment category for both cloud-based services and lexicon-based software. We investigated and found an overall significant difference between each service utilized. This hypothesis was used to construct the two possible outcomes of the hypothesis test, which are either Null or Alternative hypotheses. After the test was conducted, we rejected the null hypothesis when the p-value was less than 0.05 ( $p < 0.05$ ) and accepted the corresponding alternative hypothesis. The null hypotheses ( $H_0$ ) which signifies that all means are equal and

corresponding alternative hypotheses which signifies at least one mean is statistically different ( $H_1$ ) are shown below in the result section for each table.

## 8.4 Results

This section shows the result of the descriptive analysis process performed in the methods section. The t-test comparing the four categories for both Amazon comprehend and Vader is shown in the tables below. Table 1 shows the t-test for normalized negative percentage value for both amazon comprehend and Vader. Table 2 shows the t-test for normalized neutral percentage value for both amazon comprehend and vader. Table 3 shows the t-test for normalized positive percentage value, while table 4 shows the t-test normalized compound/mixed percentage for both amazon comprehend and vader.

<p><math>H_0</math>: There is no significant difference between the negative sentiment category for Amazon comprehend and that of Vader.</p> <p><math>H_1</math>: There is a significant difference between the negative sentiment category for Amazon comprehend and that of Vader.</p>	<p><math>H_0</math>: There is no significant difference between the neutral sentiment category for Amazon comprehend and that of Vader.</p> <p><math>H_1</math>: There is a significant difference between the neutral sentiment category for Amazon comprehend and that of Vader</p>
--	---

t-Test: Paired Two Sample for Means			t-Test: Paired Two Sample for Means		
	AWSNegPct	VaderNegPct		AWSNeutralPct	VaderNeutralPct
<b>Mean</b>	0.183	0.289	<b>Mean</b>	0.399	0.648
<b>Variance</b>	0.002	0.098	<b>Variance</b>	0.005	0.096
<b>Observations</b>	83.000	83.000	<b>Observations</b>	83.000	83.000
<b>Pearson Correlation</b>	-0.427		<b>Pearson Correlation</b>	-0.268	
<b>Hypothesized Mean Difference</b>	0.000		<b>Hypothesized Mean Difference</b>	0.000	
<b>df</b>	82.000		<b>df</b>	82.000	
<b>t Stat</b>	-2.852		<b>t Stat</b>	-6.757	
<b>P(T&lt;=t) two-tail</b>	0.005		<b>P(T&lt;=t) two-tail</b>	0.000	
<b>t Critical two-tail</b>	1.989		<b>t Critical two-tail</b>	1.989	

Table 8-2:T-test for negative percentage

Table 8-3:T-test for neutral percentage

<p><math>H_0</math>: There is no significant difference between the positive sentiment category for Amazon comprehend and that of Vader</p>	<p><math>H_0</math>: There is no significant difference between the compound sentiment category for Amazon comprehend and that of Vader</p>
---	---

H <sub>1</sub> : There is a significant difference between the positive sentiment category for Amazon comprehend and that of Vader.	H <sub>1</sub> : There is a significant difference between the compound sentiment category for Amazon comprehend and that of Vader.
---	---

t-Test: Paired Two Sample for Means			t-Test: Paired Two Sample for Means		
	AWSPosPct	VaderPosPct		AWSCompPct	VaderCompPct
Mean	0.233	0.059	Mean	0.184	0.005
Variance	0.002	0.019	Variance	0.003	0.000
Observations	83.000	83.000	Observations	83.000	83.000
Pearson Correlation	-0.106		Pearson Correlation	0.020	
Hypothesized Mean Difference	0.000		Hypothesized Mean Difference	0.000	
df	82.000		df	82.000	
t Stat	10.614		t Stat	31.894	
P(T<=t) two-tail	0.000		P(T<=t) two-tail	0.000	
t Critical two-tail	1.989		t Critical two-tail	1.989	

Table 8-4: T-test for positive percentage.

Table 8-5: T-test for compound percentage.

Therefore, for each sentiment category the p-value derived was less than 0.05. This indicated that we rejected the null hypothesis and we concluded that there is a difference in the means. The Analysis of Variance test shows a comparison of the compound score between the lexicon-based software and the three different cloud providers. There is a significant difference between these groups. It also shows the sum of squares and the p-value is less than 0.05. ( $p < 0.05$ ). Table 5 below shows the analysis carried out before generating the ANOVA test. Table 6 shows the statistic values for the ANOVA test.

Descriptive Analysis								
Compound score								
	No. of comments	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
1. (Amazon Comprehend)	85	0.0050	0.0102	0.0011	0.0027	0.0072	0.0000	0.0534
2. (Vader)	85	0.1854	0.0505	0.0055	0.1745	0.1963	0.0317	0.2622
3. (Google NLP)	85	0.4053	0.2062	0.0224	0.3608	0.4498	0.0500	0.9500
4. (Azure Text-Analytics)	85	0.6288	0.2015	0.0219	0.5854	0.6723	0.0732	0.9078





*Table 8-8: Sentiment-category comparisons*

## 8.5 Conclusion

This study demonstrated the systematic process involved in data collection, data pre-processing and data normalization for the efficient analysis of scientific datasets which is utilized in making predictions for business intelligence models. The classification of movie quotes in study three was determined using the sentiment score calculated from the cloud-based analysis and the lexicon-based software analysis. The quotes were converted into comments and classified into positive, neutral, negative, and mixed sentiments. This classification highlights the opinions of individuals on certain subjects and issues, these sentiments are analyzed to develop business intelligence models which deliver innovative insights for businesses and institutions to adjust and make corrective actions to their products and services to better meet the expectations of their customers. This study shows evidence that there is a significant difference in the sentiments derived from cloud-based tools and lexicon-based tools. These tools used the same machine learning algorithms during analysis. We also found significant differences in sentiment scores between the four sentiment-category for each platform. The p-value derived from the analysis is less than 0.05, which shows a significant difference in the sentiment scores.

## 9 Conclusion

In the future, this third study can be extended by developing a web client user interface that would analyze the sentences in real-time. In this thesis, a large corpus of data was extracted from social media and sentiments were derived from these comments. Data mining techniques and algorithms for classification were utilized in this thesis. The research hypothesis in study one was to compare machine learning algorithms why using cloud-based tools. The use of a cybersecurity dataset was to understand the necessary attributes needed to predict network vulnerabilities. In study one, a model was developed that enabled us to view these vulnerabilities in the network. The aim of study two was to present a Python based framework that can be used in implementing sentiment analysis and utilization of machine learning based algorithms on cloud platforms on any given data set. It involved going through the various stages of data analysis, data preprocessing, data normalization, algorithm selection. The data set was acquired using the Push Shift reddit API to extract comments dating back to the inception of reddit. After that, the data set was processed and transformed using normalization techniques to make it compatible with the learning algorithms. The use of different machine learning for text classification can help reduce the workload of analyzing through thousands of texts by making it easier to understand the contents of our data set in a timely manner. From this study, we have also seen the benefits of making use of the python scikit-learn library. It makes data analysis fun as it is quite easy to use and contains all machine learning algorithms. It also contains several optimization routines which have been perfectly integrated with other algorithms. In study three, we utilized a cloud-based tool to perform sentiment analysis. The processes involved are quite similar but differ in terms of categories. Amazon comprehend scores the text in four dissimilar categories, mixed, positive, neutral, and negative, while Google natural language scores the text in magnitude and sentiment scores. Vader scores the sentiment in four dissimilar categories which is like that of Amazon comprehends. After the scores were obtained, we normalized the data, to make effective comparison and derive a significance in each category. This thesis research gives a detailed explanation of cloud-based processes and the effective use of machine learning algorithms. It also gives a sequential procedure on how to utilize machine learning algorithms in other programming languages such as python.

## 10 References

- Aggarwal, A., Gola, B., & Sankla, T. (2021, 2021//). *Data Mining and Analysis of Reddit User Data*. Paper presented at the Cybernetics, Cognition and Machine Learning Applications, Singapore.
- Al-Omair, O. M., & Huang, S. (2018). *A comparative study on detection accuracy of cloud-based emotion recognition services*. Paper presented at the Proceedings of the 2018 International Conference on Signal Processing and Machine Learning.
- Alkalbani, A. M., Ghamry, A. M., Hussain, F. K., & Hussain, O. K. (2016). *Sentiment analysis and classification for software as a service reviews*. Paper presented at the 2016 IEEE 30th international conference on advanced information networking and applications (AINA).
- Arulmurugan, R., Sabarmathi, K. R., & Anandakumar, H. (2019). Classification of sentence level sentiment analysis using cloud machine learning techniques. *Cluster Computing*, 22(1), 1199-1209. doi:10.1007/s10586-017-1200-1
- Baumgartner, J., Zannettou, S., Keegan, B., Squire, M., & Blackburn, J. (2020). *The pushshift reddit dataset*. Paper presented at the Proceedings of the international AAAI conference on web and social media.
- Biswas, A., Majumdar, S., Nandy, B., & El-Haraki, A. (2015). *An auto-scaling framework for controlling enterprise resources on clouds*. Paper presented at the Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Shenzhen, China.  
<https://doi.org/10.1109/CCGrid.2015.120>
- Butt, U. A., Mehmood, M., Shah, S. B. H., Amin, R., Shaukat, M. W., Raza, S. M., . . . Piran, M. J. (2020). A Review of Machine Learning Algorithms for Cloud Computing Security. *Electronics*, 9(9), 1379. Retrieved from <https://www.mdpi.com/2079-9292/9/9/1379>
- Carvalho, A., & Xu, J. (2021). *Studies on the Accuracy of Ensembles of Cloud-Based Technologies for Sentiment Analysis*. Paper presented at the AMCIS.
- Chahal, D., Ojha, R., Choudhury, S. R., & Nambiar, M. (2020). *Migrating a Recommendation System to Cloud Using ML Workflow*. Paper presented at the Companion of the ACM/SPEC International Conference on Performance Engineering, Edmonton AB, Canada.  
<https://doi.org/10.1145/3375555.3384423>
- Chanthakit, S., & Rattanapoka, C. (2020, 21-22 Oct. 2020). *A Campus IoT Cloud Platform for Stream Processing*. Paper presented at the 2020 - 5th International Conference on Information Technology (InCIT).
- Chen, Z., & Sokolova, M. (2021). Sentiment Analysis of the COVID-related r/Depression Posts. *arXiv preprint arXiv:2108.06215*.
- D. Miller, J. (2019). *Hands-On Machine Learning with IBM Watson : Leverage IBM Watson to Implement Machine Learning Techniques and Algorithms Using Python*: Packt Publishing, Limited.
- Dark Data Project, T. S. p. Hatebase Repository. Retrieved from <https://hatebase.org/>
- Dave, K., Lawrence, S., & Pennock, D. M. (2003). *Mining the peanut gallery: Opinion extraction and semantic classification of product reviews*. Paper presented at the Proceedings of the 12th international conference on World Wide Web.
- Ding, P.-W., & Hsu, I.-C. (2018). *A Semantic Internet of Things Framework using Machine Learning Approach based on Cloud Computing*. Paper presented at the Proceedings of the 2nd International Conference on Digital Signal Processing, Tokyo, Japan.  
<https://doi.org/10.1145/3193025.3193055>
- Duong, T. N. B., & Sang, N. Q. (2018, 23-25 Nov. 2018). *Distributed Machine Learning on IAAS Clouds*. Paper presented at the 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS).

- Ferreira, L., Pilastri, A., Martins, C. M., Pires, P. M., & Cortez, P. (2021, 18-22 July 2021). *A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost*. Paper presented at the 2021 International Joint Conference on Neural Networks (IJCNN).
- Gajananan, K., Loyola, P., Katsuno, Y., Munawar, A., Trent, S., & Satoh, F. (2018). *Modeling sentiment polarity in support ticket data for predicting cloud service subscription renewal*. Paper presented at the 2018 IEEE International Conference on Services Computing (SCC).
- Gangadhar, S., & Shanta, R. (2018). Chapter 8 - Machine Learning. *Handbook of Statistics*, 38, 197-228. doi:<https://doi.org/10.1016/bs.host.2018.07.004>
- Gaydhani, A., Doma, V., Kendre, S., & Bhagwat, L. (2018). Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. *arXiv preprint arXiv:1809.08651*.
- Gitari, N. D., Zuping, Z., Damien, H., & Long, J. (2015). A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4), 215-230.
- Graumas, L., David, R., & Caselli, T. (2019). *Twitter-based polarised embeddings for abusive language detection*. Paper presented at the 2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW).
- Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., . . . Zha, S. (2020). GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing. *J. Mach. Learn. Res.*, 21(23), 1-7.
- Harfoushi, O., Hasan, D., & Obiedat, R. (2018). Sentiment analysis algorithms through azure machine learning: Analysis and comparison. *Modern Applied Science*, 12(7), 49.
- Hutto, C., & Gilbert, E. (2014). *Vader: A parsimonious rule-based model for sentiment analysis of social media text*. Paper presented at the Proceedings of the international AAAI conference on web and social media.
- Jeffers, J., Reinders, J., & Sodani, A. (2016). Chapter 24 - Machine learning. In J. Jeffers, J. Reinders, & A. Sodani (Eds.), *Intel Xeon Phi Processor High Performance Programming (Second Edition)* (pp. 527-548). Boston: Morgan Kaufmann.
- Karyotis, C., Doctor, F., Iqbal, R., James, A., & Chang, V. (2018). A fuzzy computational model of emotion for cloud based sentiment analysis. *Information Sciences*, 433, 448-463.
- Keijsers, M., Bartneck, C., & Kazmi, H. S. (2019). *Cloud-based sentiment analysis for interactive agents*. Paper presented at the Proceedings of the 7th International Conference on Human-Agent Interaction.
- Kumaresan, K., & Vidanage, K. (2019). *Hatesense: Tackling ambiguity in hate speech detection*. Paper presented at the 2019 National Information Technology Conference (NITC).
- Liu, Y., Li, B., Niu, J., & Cao, Q. (2014). *A Cloud-Based Experiment Platform for Computer-Based Education*. Paper presented at the Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. <https://doi.org/10.1109/UCC.2014.100>
- Machova, K., Mach, M., & Vasilko, M. (2021). Comparison of Machine Learning and Sentiment Analysis in Detection of Suspicious Online Reviewers on Different Type of Data. *Sensors*, 22(1), 155.
- Melton, C. A., Olusanya, O. A., Ammar, N., & Shaban-Nejad, A. (2021). Public sentiment analysis and topic modeling regarding COVID-19 vaccines on the Reddit social media platform: A call to action for strengthening vaccine confidence. *Journal of Infection and Public Health*, 14(10), 1505-1512.
- Mohamed, W. (2017). *Learning Microsoft Azure Storage*. Birmingham, UK: Packt Publishing.
- Moualla, S., Khorzom, K., & Jafar, A. (2021). Improving the Performance of Machine Learning-Based Network Intrusion Detection Systems on the UNSW-NB15 Dataset. *Computational Intelligence and Neuroscience*, 2021, 5557577. doi:10.1155/2021/5557577

- Moustafa, N., & Slay, J. (2015). *UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)*. Paper presented at the 2015 military communications and information systems conference (MilCIS).
- Opara, E., Wimmer, H., & Rebman, C. M. (2022). *Auto-ML Cyber Security Data Analysis Using Google, Azure and IBM Cloud Platforms*. Paper presented at the 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET).
- Pennebaker, J., Chung, C., Ireland, M., Gonzales, A., & Booth, R. (2007). The Development and Psychometric Properties of LIWC2007 (LIWC. Net, Austin, TX).
- Pereira-Kohatsu, J. C., Quijano-Sánchez, L., Liberatore, F., & Camacho-Collados, M. (2019). Detecting and monitoring hate speech in Twitter. *Sensors*, 19(21), 4654.
- Ping Li, T. L., Heng Ye, Jin Li, Xiaofeng Chen, Yang Xiang. (2018). Privacy-preserving machine learning with multiple data providers,. *Future Generation Computer Systems*, 87, 341-350.  
doi:<https://doi.org/10.1016/j.future.2018.04.076>.
- Pitsilis, G. K., Ramampiaro, H., & Langseth, H. (2018). Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.
- Pliuhin, V., Pan, M., Yesina, V., & Sukhonos, M. (2018, 9-12 Oct. 2018). *Using Azure Maching Learning Cloud Technology for Electric Machines Optimization*. Paper presented at the 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T).
- Qaisi, L. M., & Aljarah, I. (2016). *A twitter sentiment analysis for cloud providers: a case study of Azure vs. AWS*. Paper presented at the 2016 7th International Conference on Computer Science and Information Technology (CSIT).
- Rezazadeh, A. (2020). A Generalized Flow for B2B Sales Predictive Modeling: An Azure Machine-Learning Approach. *Forecasting*, 2(3), 267. doi:10.3390/forecast2030015
- Ruwandika, N., & Weerasinghe, A. (2018). *Identification of hate speech in social media*. Paper presented at the 2018 18th international conference on advances in ICT for emerging regions (ICTer).
- Salza, P., Hemberg, E., Ferrucci, F., & O'Reilly, U.-M. (2017). *Towards evolutionary machine learning comparison, competition, and collaboration with a multi-cloud platform*. Paper presented at the Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany. <https://doi.org/10.1145/3067695.3082474>
- Swasey, D., Murphy, T., Crary, K., & Harper, R. (2006). *A separate compilation extension to standard ML*. Paper presented at the Proceedings of the 2006 workshop on ML, Portland, Oregon, USA.  
<https://doi.org/10.1145/1159876.1159883>
- Tan, C., & Lee, L. (2015). *All who wander: On the prevalence and characteristics of multi-community engagement*. Paper presented at the Proceedings of the 24th International Conference on World Wide Web.
- Tedeschi, A., & Benedetto, F. (2015, 16-18 Sept. 2015). *A cloud-based big data sentiment analysis application for enterprises' brand monitoring in social media streams*. Paper presented at the 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI).
- Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C. B., & Farivar, R. (2019, 4-6 Nov. 2019). *Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools*. Paper presented at the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI).
- Webber-Cross, G. (2014). *Learning Microsoft Azure*. [N.p.]: Packt Publishing.
- White, T. E., & Rege, M. (2020). Sentiment Analysis on Google Cloud Platform. *Issues Inf. Syst*, 21, 221-228.

- Wickham, M. (2018). Leveraging Cloud Platforms. In *Practical Java Machine Learning* (pp. 105-175): Springer.
- Wimmer, H., Powell, L. M. (2016). A Comparison of Open Source Tools for Data Science. *Journal of Information Systems Applied Research*, 9(2), 4-12. Retrieved from <http://jisar.org/2016-9/>
- Yan, T., & Liu, F. (2021). Sentiment Analysis and Effect of COVID-19 Pandemic using College SubReddit Data. *arXiv preprint arXiv:2112.04351*.
- Yang, C., Fan, J., Wu, Z., & Udell, M. (2020). *AutoML Pipeline Selection: Efficiently Navigating the Combinatorial Space*. Paper presented at the Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA. <https://doi.org/10.1145/3394486.3403197>
- Yeung, J., Wong, S., Tam, A., & So, J. (2019, 30-31 July 2019). *Integrating Machine Learning Technology to Data Analytics for E-Commerce on Cloud*. Paper presented at the 2019 Third World Conference on Smart Trends in Systems Security and Sustainability (WorldS4).
- Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019). Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666*.
- Zhang, H., Li, Y., Huang, Y., Wen, Y., Yin, J., & Guan, K. (2020). *MLModelCI: An Automatic Cloud Platform for Efficient MLaaS*. Paper presented at the Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA. <https://doi.org/10.1145/3394171.3414535>
- Zhang, W., Xu, Q., Zhao, M., Zhang, J., & Wang, N. (2020). *Design and Implementation of a Time Prediction Model for LS-DYNA Cloud Computing Platform*. Paper presented at the Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition, Xiamen, China. <https://doi.org/10.1145/3436369.3436493>
- Zouhair Chiba, N. A., Khalid Moussaid, Amina El omri, Mohamed Rida,. (2019). Intelligent approach to build a Deep Neural Network based IDS for cloud environment using combination of machine learning algorithms. *Computers & Security, Volume 86*, 291-317. doi:<https://doi.org/10.1016/j.cose.2019.06.013>.