JOURNAL OF INFORMATION SYSTEMS APPLIED RESEARCH

Volume 17, No. 3
December 2024
ISSN: 1946-1836

In this issue:

iscap
Information Systems & Computing Academic Professionals

The **Journal of Information Systems Applied Research** (JISAR) is a double-blind peer reviewed academic journal published by ISCAP, Information Systems and Computing Academic Professionals. Publishing frequency is three issues a year. The first date of publication was December 1, 2008.

JISAR is published online (https://jisar.org) in connection with the ISCAP (Information Systems and Computing Academic Professionals) Conference, where submissions are also double-blind peer reviewed. Our sister publication, the Proceedings of the ISCAP Conference, features all papers, teaching cases and abstracts from the conference. (https://iscap.us/proceedings)

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the conference. At that point papers are divided into award papers (top 15%) and other submitted works. The non-award winning papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in JISAR. Currently the acceptance rate for the journal is approximately 35%.

Questions should be addressed to the editor at editor@jisar.org or the publisher at publisher@jisar.org. Special thanks to members of ISCAP who perform the editorial and review processes for JISAR.

2024 ISCAP Board of Directors

Journal of Information Systems Applied Research

Network Intrusion Detection System with Machine Learning as a Service

Loma Kangethe
Lk07736@georgiasouthern.edu

Hayden Wimmer
hwimmer@georgiasouthern.edu

Department of Information Technology
Georgia Southern University
Statesboro, GA 30460, USA

Carl M Rebman, Jr.
carlr@sandiego.edu
Knauss School of Business
Department of Supply Chain, Operations, and Information Systems
University of San Diego
San Diego, CA 92110, USA

Abstract

Cloud Computing and Big Data continue to be disruptive forces in computing and has introduced new threats and vulnerabilities to our networks. The paper seeks to demonstrate how an end-to-end network intrusion detection system can be built, trained, and deployed using Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP). We determined the performance of these tools by building a network intrusion detection system (NIDS) and evaluating the performance of each based on precision, accuracy, F1 Score, recall, user experience, cost and computation time for training and predicting the model. Overall, all three platforms performed greater than 90% accuracy with Google Vertex AI having the highest accuracy using the decision tree and Microsoft Azure performing the best based on accuracy, precision, and computation time.

Keywords- Network Intrusion Detection, Machine Learning (ML), Microsoft Azure Machine Learning, Amazon Web Services (AWS), Sage Maker, Vertex AI

Recommended Citation: Kangethe, L., Wimmer, H., Rebman Jr., C.M., (2024). Network Intrusion Detection System with Machine Learning as a Service *Journal of Information Systems Applied Research.* 17(3), pp. 4-15, https://doi.org/10.62273/EWQL5023

Network Intrusion Detection System with Machine Learning as a Service

Koma Kangethe, Hayden Wimmer and Carl M. Rebman, Jr.

1. INTRODUCTION

Cloud computing is an on-demand access to computing resources such as servers (physical servers and virtual servers), data storage, applications, development tools, networking capabilities, analytics, intelligence and networking capabilities. This access is enabled via the internet by hosts as remote data centers that are managed by a cloud services provider (CSP) such as AWS, Microsoft Azure and Google. These CSP are able to offer faster innovation, flexible resources, and economies of scale. Some of the more specific services that CSP provides include r Software as a service (Saas), Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Machine Learning as a Service (MLaaS). MLaaS is a range of services that offer machine-learning tools as part of Cloud Computing Services.

MLaaS is an automated and semi-automated cloud platform that is provided in a cloud marketplace that stores massive amounts of data, has low deployment costs, and has high computing performance. MLaaS providers use their own data center to handle calculations and prevent clients from running their own servers or installing their own software. This introduces cost savings and eliminates the risk associated with having the infrastructure on premises. The platform covers most infrastructure issues such as data pre-processing, model training, and evaluation. Predictions can easily be bridged through REST API to the respective applications. MLaaS services allow for fast model training and deployment with little to no data science expertise. MLaaS providers offer tools that include data visualization, APIs, face recognition, natural language processing, predictive analytics, deep learning and many more.

Figure 1 illustrates how these attacks occur in the network. There are mainly two types of intrusion detection systems (IDS): misuse (signature) based, and anomaly-based systems. A misuse-based system detects already known attacks by use of signature rules defined by network administrators or security specialists. Misuse detection requires frequent updates of signatures to ensure ample detection. If a new attack is reported the security specialists must update the signature database. The limitation with this system is that it cannot be helpful against unknown attacks such as zero-day attacks. An anomaly-based system checks the network behavior and classifies it as normal or abnormal. This system analyzes past data to detect both known and unknown attacks by using statistical based, machine learning based and knowledge-based techniques.

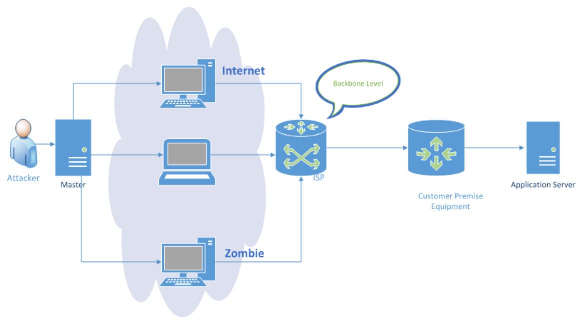


Figure 1: Malware Attack Architecture

The objective of the paper is to use and compare cloud-based machine learning tools for enhanced big data applications to provide data scientists with a set of tools and cloud services that cover the end-to-end machine learning development cycle: ranging from the models' creation, training, validation and testing to the models serving as a service, sharing and deployment. The cloud-based architecture will showcase how data scientists' researchers can develop complex models and train the models in a distributed manner hence automatically parallelizing models and datasets across multiple processors. This research will also compare with building the ML algorithm on-premises.

2. LITERATURE REVIEW

Mäkelä (2022) researched cloud machine learning service providers by comparing the end-to-end machine learning processes on both Amazon Web Services (AWS) and Google Cloud Platform (GCP). The objective was to build an end-to-end machine learning which included two main sections of data pre-processing, and model training and deployment. They then compared the features and performance between the two providers. Data was prepared by organizing and also removed unnecessary variables or information. Jobs were run to prepare the data, and once transformations were completed the dataset was exported back for model training. The GCP command line terminal was used to build the model using TensorFlow. Results show both platforms are equally capable of training an externally created model.

Berg (2022) studied image classification with Machine Learning as a Service seeks to compare Azure, AWS SageMaker, and Vertex AI. The study reviewed MLaaS needs and MLaaS providers and compared various characteristics of MLaaS providers. An image classification algorithm was built, trained, validated and deployed on all the management console on the three cloud platforms using three different datasets. The prediction accuracy, training time, and cost were measured with three different datasets and their features compared. Results indicated that Microsoft Azure ML performed best in terms of prediction accuracy, and training cost, across all datasets. Amazon Web Services SageMaker had the shortest time to train but performed the worst in terms of accuracy and had trouble with two of the three datasets. Google Cloud Platform Vertex AI did achieve the second-best prediction accuracy but was the most expensive platform as it had the largest time to train. It did, however, provide the smoothest user experience. Overall, Azure ML would be the platform of choice for image classification tasks after weighing together the results of the experiments as well as their subjective user experience.

Xhepa and Kanakala (2022) studied Machine Learning Model Computation in AWS and Azure. They examined the machine learning image classification service on cloud to determine the best cloud platform for machine learning projects. For AWS, Amazon SageMaker was used to build, train and deploy the model, S3 bucket for object storage, Amazon EC2 was used to configure instances and IAM for access management. For Microsoft Azure; AzureML was used for the development, training, and deployment of the model and Azure Blob Storage for object storage. Azure Machine Learning workspace was utilized for working with all of the artifacts generated by Azure Machine Learning. Results indicated an artificial intelligent system built on AWS and Azure cloud platforms has the capability to efficiently learn from increasingly complex images with a high degree of generalization using a relatively small repository of data. A highly accurate model was built using TensorFlow with an accuracy of 58%. AWS SageMaker appeared to be an excellent platform for developing simple models and deploying them in the cloud with minimal configuration. However, for predictive analytics, Azure ML may be a more versatile option and this study demonstrated how users can use Amazon Sage Maker and AzureML to easily build, train, and deploy machine learning models.

Opara, Wimmer, and Rebman (2022) studied building an Auto-ML model on the cloud environment. Their objective was to demonstrate the Auto ML functionalities against cyber security threat detection using three different cloud platforms Microsoft Azure, Google, and IBM. They then determined the performance of each platform by evaluating the optimization speed and accuracy results. The UNSW-NB15 dataset and Decision Tree Classifier, Random Forest Classifier and Gradient Boost Classifier algorithms were used and compared in terms of how they processed data and their accuracies. A comparison of the advantages of each of the results from the different platforms were presented and their results showed all three platforms performed greater than 70% accuracy with the IBM Cloud Platform having the strongest performance. They observed that the best machine learning algorithm utilized was the Gradient Boost Classifier algorithm with an accuracy score of 89.5%.

Liberty et al. (2020)’s work centered on scalable machine learning. They looked at the challenges of training ML models on large, continuously evolving datasets and included the following variables; support for incremental training and model freshness, predictability of training costs, elasticity and support for pausing and resuming training jobs for large-scale ML, and the ability to automate hyperparameter optimization and model tuning. Their study built, trained and deployed selected algorithms using Amazon SageMaker, which supports incremental, resumable and elastic learning, as well as automatic hyperparameter optimization. The platform was compared to JVM-based algorithm implementations with regard to computation time and cost. The algorithms selected were Linear Learner, Factorization Machines, K-Means Clustering, Principal Component Analysis (PCA), Neural Topic Model (NTM), Time Series Forecasting with DeepAR. Results indicated that SageMaker can train models both faster and more cost-effective in the majority JVM-based algorithm and is up to 8-times faster than MLlib, as they provided results two to three times cheaper when using the same amount of training time.

Lee (2018) focused on analyzing trends in machine learning as a service, the purpose of the study was to review MLaaS needs and MLaaS providers and compare various characteristics of MLaaS providers. Lee (2018) used four cloud platforms Microsoft Azure Machine Learning Studio, Amazon Machine Learning, Google Cloud Machine Learning Engine, BigML and IBM Watson Studio. Models were built, trained and deployed on the management console and are compared based on the features. Results showed that for AWS, data can be loaded from multiple sources including Amazon RDS, Amazon Redshift, and CSV file.

Kaymakci, et al. (2022) centered on the problems of digital transformation and a transition to cloud-based solutions to use AI/ML by small and medium-sized businesses (SMBs) in the industrial sector. The study presented a systematic selection process of ML cloud services for manufacturing SMB and posed a four-step process to select ML cloud services for SMBs based on an analytic hierarchy process. Their objective was to minimize the hurdles to ML cloud service adoption and to promote digital transformation in manufacturing SMBs. A decision matrix was used to identify the most-suited ML cloud service. The target was focused on IT Security, reliability, cloud management, flexibility, costs, performance, and normalized score. The normalized score showed that AWS SageMaker had a score of 0.3725, Azure ML had a score of 0.38334, and GVP AI platform had a score of 0.2441. Hence, Microsoft's Azure ML had the highest score of the three services and, therefore, is the most fitting for meeting the specific goals for SMB.

Developing classification algorithms using machine learning frameworks is time-consuming, costly, and requires a team with technical capabilities. Noshiri et al. (2021) focused on adopting Machine Learning-as-a-service (MLaaS) cloud delivery model. They used pre-built classification algorithms and evaluated the performance of BigML, Microsoft Azure ML Studio, IBM Watson ML Studio and Google AutoML platforms on the classification of multi-class datasets. They used the metrices of the average-micro-F-score, accuracy, training time, and cost. The purpose of the research was to assist small-to-medium companies in choosing the appropriate platform based on the trade-offs between the average-micro-F-score and training time. A 10 labeled dataset from the UC Irvine ML Repository was used. IBM SPSS Statistics 26, a statistical software was used to extract actionable insights from the data. random subsampling cross-validation procedure was performed on all datasets with different random seeds. Data was split in ratio of 85% to 15% for training and testing data, The data was then fed to the various cloud performs and evaluated. The study found that Google AutoML provided the user with the highest average micro-F score although it is costly and requires more training time.

Yao et al. (2017) focused on researching the complexity vs. performance of Machine Learning as a Service. They evaluated the effectiveness of MLaaS systems from customizable systems to fully automated ones to find out how control relates to risk. UCI machine learning repository was used to get the datasets. The datasets included both numeric and categorical features. Categorical features were converted to numerical values, missing fields with median values, and data samples were split into training and test set by a 70%–30% ratio. The models used were Logistic Regression, Support Vector Machine, Naïve Bayes, Multi-Layer Perceptron, Decision Tree, Bagging, and k-Nearest Neighbor. The platforms used were Amazon Google, Big ML, Prediction IO, Local and Microsoft. Results showed that platforms with higher complexity (more dimensions for user control) achieved better performance. Microsoft provided the highest performance across all platforms, and a highly tuned Microsoft model can produce performance identical to that of a highly tuned local scikit-learn instance. It was observed that server-side optimizations help fully automated systems outperform default settings on competitors, but still lag far behind well-tuned MLaaS systems which compare favorably to standalone ML libraries.

Zhao et al. (2020) used Machine-Learning Based TCP security action prediction in their study which focused on addressing the problems faced in network security. The use of TCP firewalls to either allow or deny traffic according to specific rules is a common exercise of network administrators. However, this is a daunting task due to the huge amount of data on the internet. Machine learning methods of computer security are used to ease this burden. The research sought to predict TCP security action based on TCP transmission characteristics using Machine

Learning techniques. Data Inspection was done and standardized. Importance of each feature analyzed by assessing their permutation importance and fed to the ML engine. Machine models used were AdaBoost, Logistic regression, Support Vector Machine (SVM) neural networks and other ensemble techniques. Results revealed ensemble methods that combine individually reasonable models to make an integrated classifier achieve the best accuracy. The predicted accuracy of TCP security action reached over 98%. Better accuracies can be attained by further preprocessing and fine tuning the algorithms (Zhao et al., 2020).

3. METHODOLOGY

In this section, we present the system setup architecture, the feature attributes of the dataset, the preprocessing techniques, the classification algorithms explored, and the test and evaluation methods used for each of the cloud platforms is described.

System Architecture
This section presents the design architecture of the system and explains the sequence of the process and procedure. This architecture was inspired by the architecture of the three renown cloud platforms to find a solution that is fit for use and purpose based on the use case.

Figure 2 is a system design architecture employed to the three-cloud platform namely AWS, Microsoft Azure, and Google cloud platform. It demonstrates the flow of the designed framework from when the data is loaded to the output of the network intrusion detection system.

The process can be summarized in the following steps:

1. **Data Extraction**: Fetch and Load the data – Data was generated within GSU Southern IT Lab in PCAP format and converted to CSV.
2. **Exploratory data analysis (EDA)**: Analysis was done on the dataset to help identify obvious errors, better understand patterns within the dataset, detect outliers, find relations among variables, and provide insight.
3. **Data Preprocessing**: Data was cleaned by dealing with missing values and the SMOTE technique used to deal with oversampling.
4. **Feature Importance:** This technique assigns a score to input features based on how useful they are at predicting a target variable. This is important for feature selection.

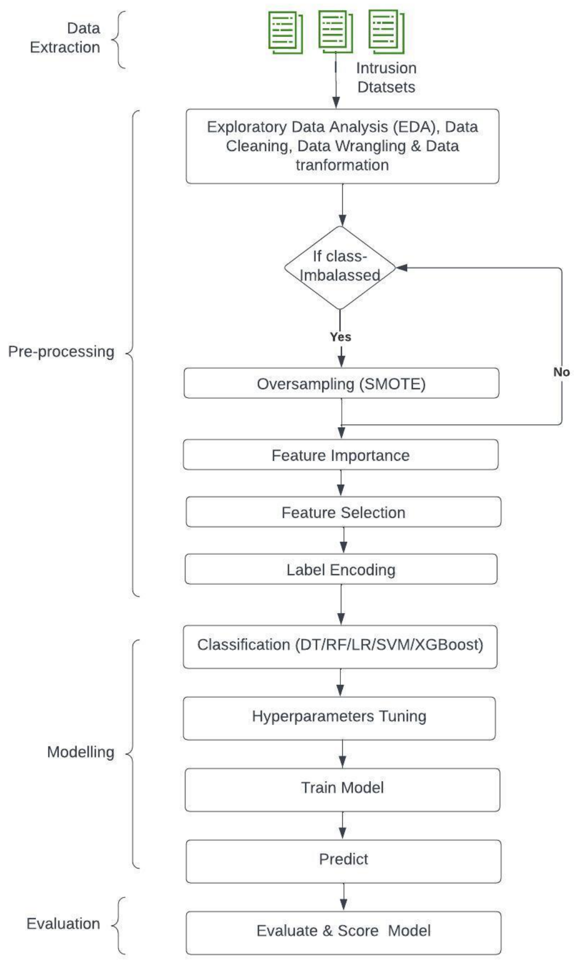


Figure 2: Cloud Machine Learning Flowchart

5. **Feature Selection**: This technique reduces the number of input variables when developing a predictive model. Hence redundant and irrelevant features in the data which may slow down the process of classification are handled. Tree based and Chi-Squared methods are used.
6. **Label Encoding:** converting the labels into a numeric form so as to convert them into the machine-readable form. This is to ensure the Label column is encoded without having any weights or has an ordinal nature.
7. **Train and Test Instance**: the data is split into training dataset and testing dataset randomly. The training dataset is set to 70% while the Test data is set

to 30%. The training data set is fed into different machine learning algorithms to build the model.

8. **Model Classifier**: Different classification models are built used to classify the network traffic in the test data set as normal or attack traffic. The five classification-based algorithms built are: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting Tree (GBT) and Support vector machine (SVM).
9. **Model Evaluation**: This is the process where the models are evaluated based on Accuracy, Recall, Precision and F1 Score.
10. **Compute Time:** This is also evaluated based on the time taken to train the model and predict the model.

DATASET

The dataset was generated within the Georgia Southern University Lab environment. The dataset has real time activities and contemporary attack behavior. The purpose or goal of this research is to predict attacks in this dataset using the different cloud platforms discussed above. The implementation phase initially loads the extracted data into each of the cloud platforms. We shall cover the implementation in each of the cloud performance.

Machine Learning Process and Data Pre-processing

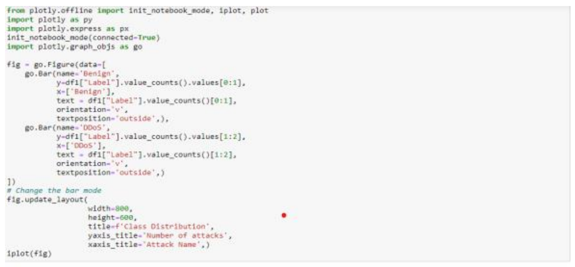
There are different preprocessing steps taken for the removal of unwanted data. The importance of this step is to have a high-quality dataset that will allow the process to be fast and the capture of malware efficient.

- **Missing values**: Features that had more than 70% of the data missing were dropped. The rest of the columns were imputed with the mode based on the descriptive statistics of the features; this ensured that the significance of the dataset was not impacted.
- **Single value columns**: The dataset contained columns with single constant values. The columns had zeros as the integer value. These columns were dropped as they added no value in the model.

Oversampling Technique

Anomaly Synthetic Minority Oversampling Technique (SMOTE) technique was used due to an imbalanced dataset resulting from the anomaly detection case not having a uniform distribution. Figure 3 visualizes the imbalance.

Smote is simply synthesizing duplicating examples from the minority class in the training dataset prior to fitting a model. This can balance the class distribution but does not provide any additional information to the model, hence can effectively learn the decision boundary during prediction.



```
from plotly.offline import init_notebook_mode, iplot, plot
import plotly as py
import plotly.express as px
init_notebook_mode(connected=True)
import plotly.graph_objs as go

fig = go.Figure(data=[
    go.Bar(name='Benign',
           y=df1["Label"].value_counts().values[0:1],
           x=['Benign'],
           text = df1["Label"].value_counts()[0:1],
           orientation='v',
           textposition='outside',),
    go.Bar(name='DDoS',
           y=df1["Label"].value_counts().values[1:2],
           x=['DDoS'],
           text = df1["Label"].value_counts()[1:2],
           orientation='v',
           textposition='outside',)
])
# Change the bar mode
fig.update_layout(
                width=800,
                height=600,
                title=f'Class Distribution',
                yaxis_title='Number of attacks',
                xaxis_title='Attack Name',)
iplot(fig)
```

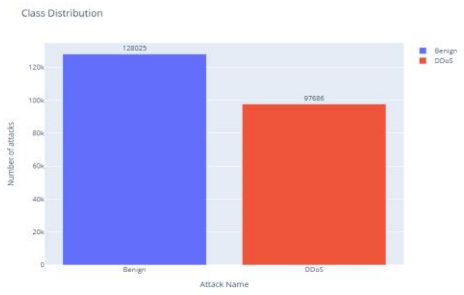
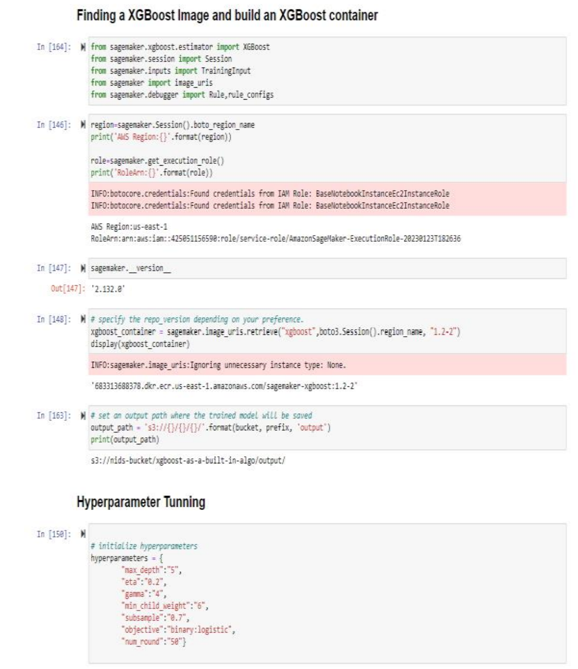


Figure 3: Class Distribution

Feature Importance & Feature Selection

The Random Forest Classifier (RFC) feature importance was used that was able to rank the features based on its relative importance to the predictor. The top 20 features were selected. Figures 4 and 5 show hyper-parameter tuning and XGB Model training respectively.



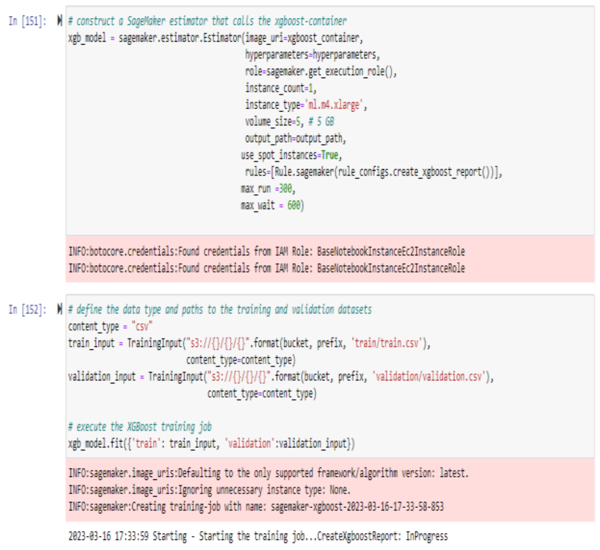
Finding a XGBoost Image and build an XGBoost container

```
In [164]:  from sagemaker.xgboost.estimator import XGBoost
           from sagemaker.session import Session
           from sagemaker.inputs import TrainingInput
           from sagemaker import image_uris
           from sagemaker.debugger import Rule,rule_configs

In [146]:  region=sagemaker.Session().boto_region_name
           print('AWS Region:{}'.format(region))

           role=sagemaker.get_execution_role()
           print('RoleArn:{}'.format(role))

           INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole
           INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole

           AWS Region:us-east-1
           RoleArn:arn:aws:iam::425051156590:role/service-role/AmazonSageMaker-ExecutionRole-20230123T182636

In [147]:  sagemaker.__version__

Out[147]:  '2.132.0'

In [148]:  # specify the repo_version depending on your preference.
           xgboost_container = sagemaker.image_uris.retrieve("xgboost",boto3.Session().region_name, "1.2-2")
           display(xgboost_container)

           INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.

           '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost:1.2-2'

In [163]:  # set an output path where the trained model will be saved
           output_path = 's3://{}/{}/{}/'.format(bucket, prefix, 'output')
           print(output_path)

           s3://nids-bucket/xgboost-as-a-built-in-algo/output/
```

Hyperparameter Tunning

```
In [150]:
           # initialize hyperparameters
           hyperparameters = {
                   "max_depth":"5",
                   "eta":"0.2",
                   "gamma":"4",
                   "min_child_weight":"6",
                   "subsample":"0.7",
                   "objective":"binary:logistic",
                   "num_round":"50"}
```

Figure 4: AWS XGBoost Hyper-parameter-Tuning



```
In [151]:  # construct a SageMaker estimator that calls the xgboost-container
           xgb_model = sagemaker.estimator.Estimator(image_uri=xgboost_container,
                                                     hyperparameters=hyperparameters,
                                                     role=sagemaker.get_execution_role(),
                                                     instance_count=1,
                                                     instance_type='ml.m4.xlarge',
                                                     volume_size=5, # 5 GB
                                                     output_path=output_path,
                                                    use_spot_instances=True,
                                                     rules=[Rule.sagemaker(rule_configs.create_xgboost_report())],
                                                    max_run =300,
                                                    max_wait = 600)

           INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole
           INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole

In [152]:  # define the data type and paths to the training and validation datasets
           content_type = "csv"
           train_input = TrainingInput("s3://{}/{}/{}".format(bucket, prefix, 'train/train.csv'),
                                       content_type=content_type)
           validation_input = TrainingInput("s3://{}/{}/{}".format(bucket, prefix, 'validation/validation.csv'),
                                            content_type=content_type)

           # execute the XGBoost training job
           xgb_model.fit({'train': train_input, 'validation':validation_input})

           INFO:sagemaker.image_uris:Defaulting to the only supported framework/algorithm version: latest.
           INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
           INFO:sagemaker:Creating training-job with name: sagemaker-xgboost-2023-03-16-17-33-58-853

           2023-03-16 17:33:59 Starting - Starting the training job...CreateXgboostReport: InProgress
```

Figure 5: XGB model train

Evaluation

The model is evaluated based on Accuracy, Precision, Recall F1Score, and Compute Time. The Classification Rate measures how accurate the IDS is in detecting normal or anomalous traffic behavior. It is the number of correct predictions made by the model over all kinds of predictions made. Equation 1 represents the formulae for accuracy.

$$\frac{TP + TN}{(TP + TN + FP + FN)}$$

Equation 1: Accuracy Calculation

Compute Time

There were two metrics measured. Time taken to train the model (i.e., the total time taken from starting the training until the training cycle is complete) and the time taken to make predictions.

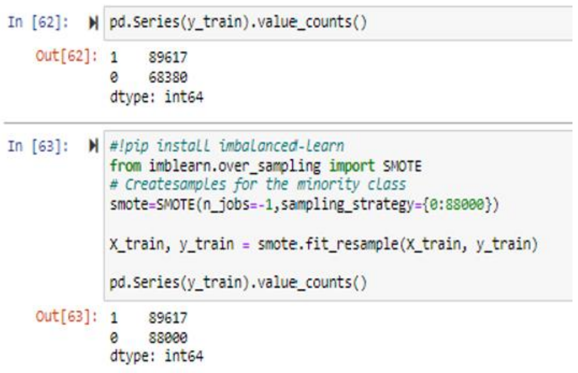
4. EXPERIMENTAL RESULTS

This section describes the results from data preprocessing, the evaluation performance of the network intrusion detection system built on Microsoft Azure, Amazon Web Services and Google Cloud platform. Machine learning models were built, trained, and deployed. The evaluation performance and features of the three platforms are compared amongst themselves and among an on-premise infrastructure.

Explanatory Data Analysis

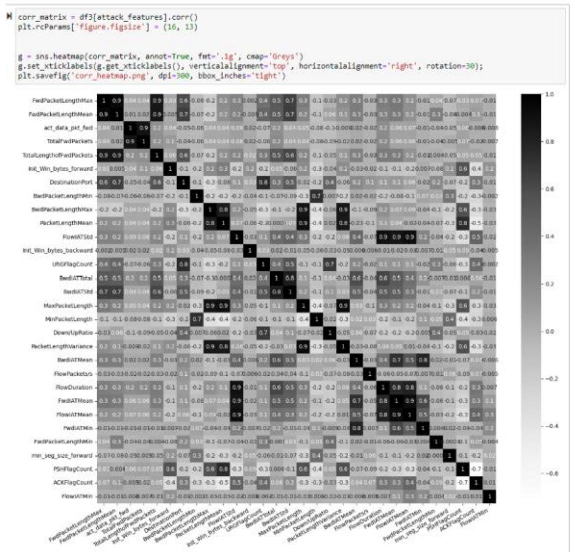
The nature of the dataset based on the class distribution of the target variable indicates an imbalanced dataset this was as the samples of benign outweigh the samples of attack. The dataset is considered not having a normal distribution. The SMOTE technique was applied to overcome this and is illustrated in Figure 6. is a snippet of the SMOTE technique applied.

The Pearson correlation was used to see the correlation between features and was used to reduce the features that were highly correlated. A correlation of greater than 0.95 was used that reduced the features to 30. Fig 7 represents the correlation matrix of features with less than 0.95 similarity. Figure 8 illustrates the importance of features assigned and ranked.



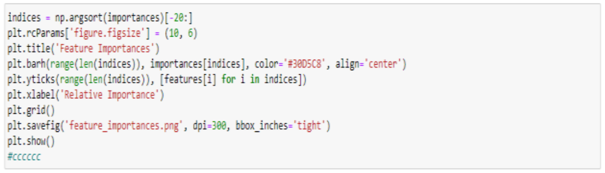
```
In [62]:  pd.Series(y_train).value_counts()

Out[62]:  1    89617
          0    68380
          dtype: int64

In [63]:  #!pip install imbalanced-learn
          from imblearn.over_sampling import SMOTE
          # Createsamples for the minority class
          smote=SMOTE(n_jobs=-1,sampling_strategy={0:88000})

          X_train, y_train = smote.fit_resample(X_train, y_train)

          pd.Series(y_train).value_counts()

Out[63]:  1    89617
          0    88000
          dtype: int64
```

Figure 6: SMOTE Technique



```
corr_matrix = df3[attack_features].corr()
plt.rcParams['figure.figsize'] = (16, 13)


g = sns.heatmap(corr_matrix, annot=True, fmt='.1g', cmap='Greys')
g.set_xticklabels(g.get_xticklabels(), verticalalignment='top', horizontalalignment='right', rotation=30);
plt.savefig('corr_heatmap.png', dpi=300, bbox_inches='tight')
```

Figure 7: Correlation Matrix



```
indices = np.argsort(importances)[-20:]
plt.rcParams['figure.figsize'] = (10, 6)
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='#30D5C8', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.grid()
plt.savefig('feature_importances.png', dpi=300, bbox_inches='tight')
plt.show()
#cccccc
```

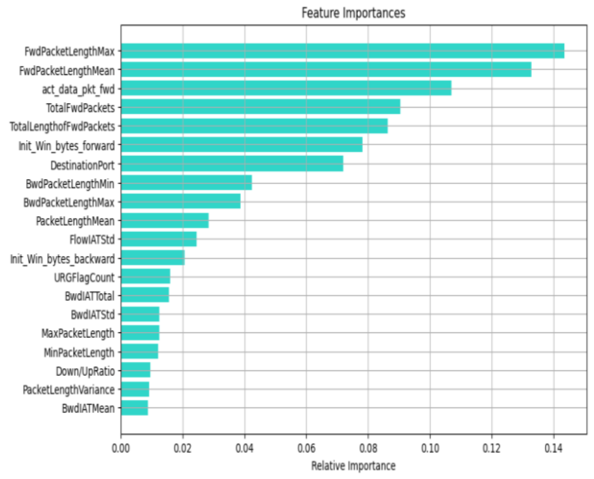


Figure 8: Feature Importance

Table 1 is a comparison of the three cloud platforms based on supported tools, Data types supported, cost etc.

Modeling Results

The models trained and deployed were Logistic Regression, SVM, Decision Tree, Random Forest, and Gradient Boost. Evaluations were compared against each of the three MlaaS and the on-premise set up. The evaluation matrix used was Accuracy, F1 score (overall performance), precision, recall and computation time which reflects on the cost and the time taken to get insights to act on.

	AWS Sage maker	GCP Vertex AI	Microsoft Azure ML
Programming Tools	Python, R studio License required.	REST API, Python, R	Python, R
ML Canvas	Drag and drop Sage maker canvas	Not available	Drag-and-drop UI Azure Ml Studio
ML Frame-works	TensorFlow, PyTorch, Keras ApacheMXNet, XGBoost, Gluon, Caffe2,Chainer, Torch	TensorFlow scikit-learn, XGBoost, Keras	TensorFlow scikit-learn, PyTorch, Microsoft Cognitive Toolkit, Spark ML
Data Types supported	Categorical Numerical Time-series Image Text	Categorical Numerical Time-series Image Text	Categorical Numerical Time-series Image Text
Feature Store	Yes	Yes	No
Built-in Algorithm	Yes	No	No
Schedule	Yes	Yes	Yes
Auto ML	Yes	Yes	Yes
Publish endpoint	Internal only	Yes	Yes
Ease of use	Score (1 -low, 5-high) Documentation – 5 Data Preparation -3 Steps in Training- 3	Score (1 -low, 5-high) Documentation-3 Data Preparation- 4 Steps in Training-4	Score (1 -low, 5-high) Documentation-4 Data Preparation- 5 Steps in Training -4
Cost Analysis	$1.125 per hour	$3.465 per hour	$0.9 per hour

Table 1: Comparison of Cloud Platforms

Results show that gradient boost gave the best parameters with an accuracy score of 93.7%, F1 score of 0.94, precision of 0.91, and recall of 0.92 followed by random forest and decision tree. Support Vector Machine and Logistic regression cave the lowest scores. In regards to computer time, Logistic Regression took the least amount of compute time across all platforms while SVM was noted to take the longest.

The average compute time on premises data warehouse is 77 sec of training and 0.26 sec for predicting time while on cloud is averagely 20 sec for training and 0.21 sec for predicting. Based on the dataset support vector machine took the longest time to train and predict.

Among the 3 MlaaS, Microsoft Azure ML takes the least amount to train and predict. This is followed by AWS. Vertex AI takes the most time to train and predict, however on premise took the longest across all models. Figures 9-14 illustrate the evaluation metrics results.

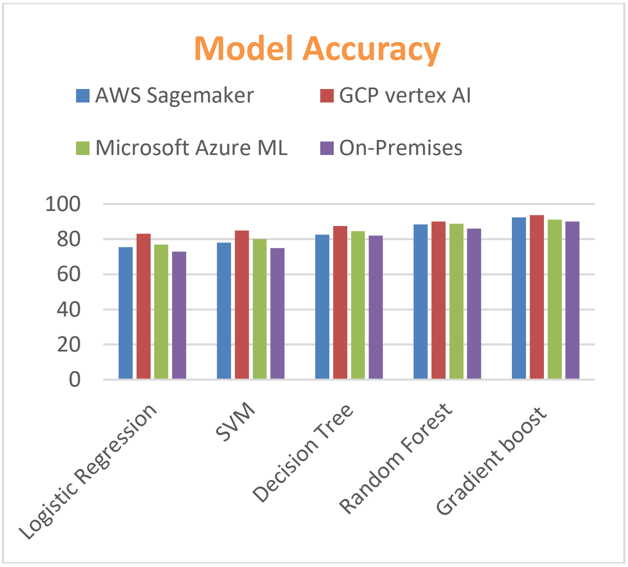


Figure 9: Model accuracy comparison

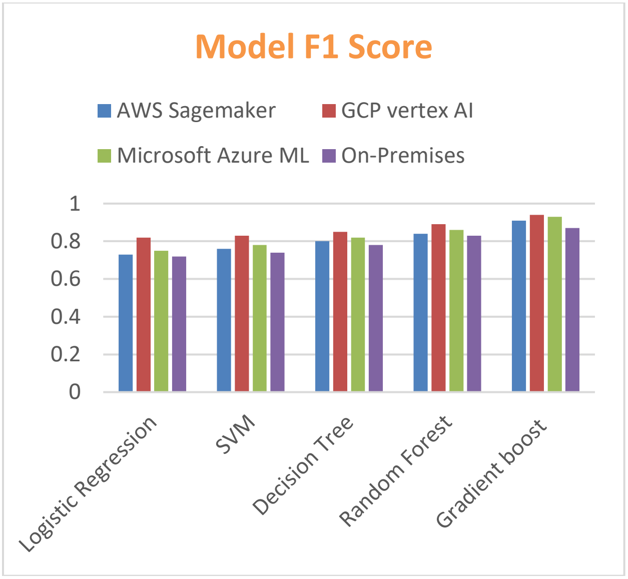


Figure 10: Model F1 score

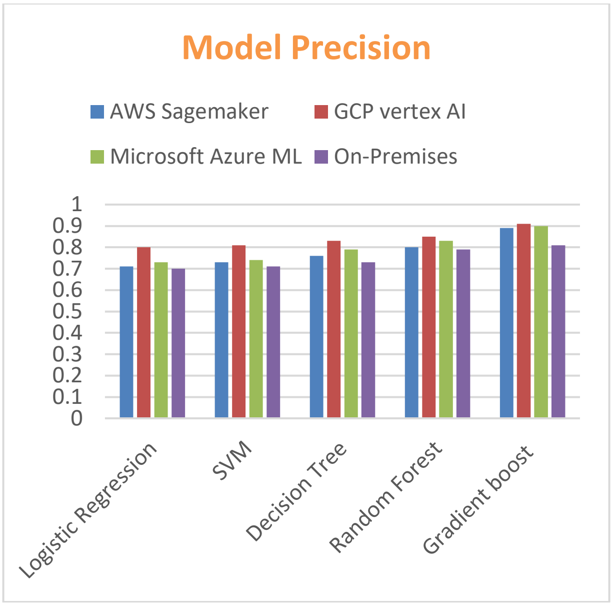


Figure 11: Model precision

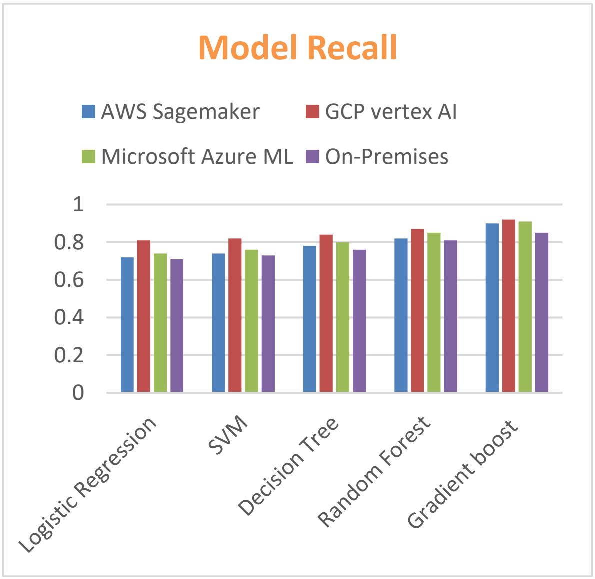


Figure 12: Model Recall

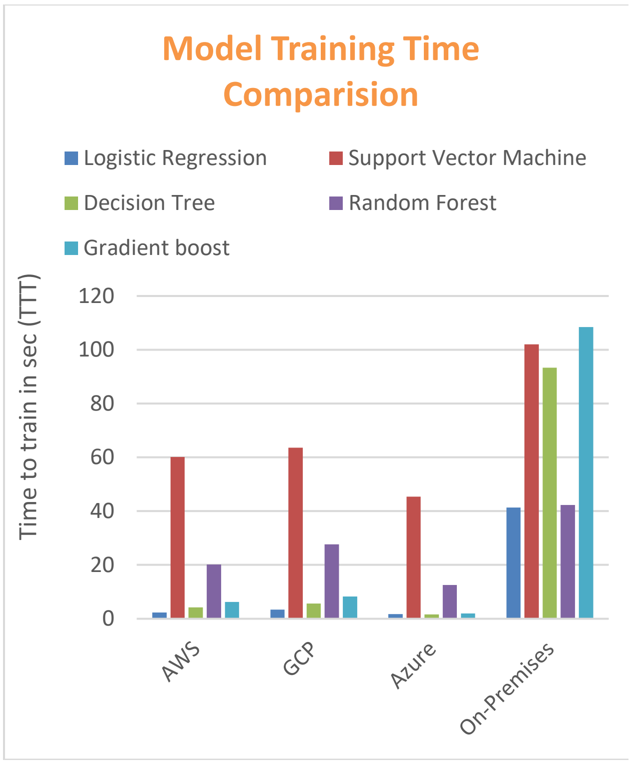


Figure 13: Model Training Time to Train in sec (TTT)

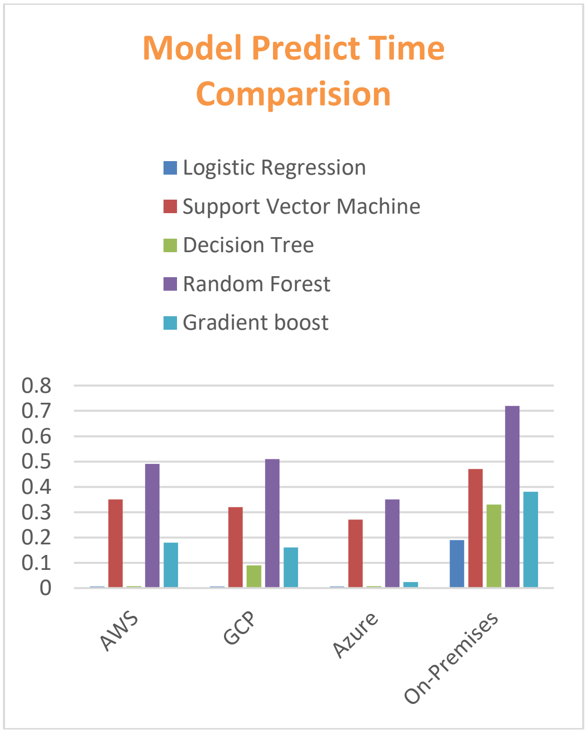


Figure 14: Model Prediction Time in seconds (TTP)

5. DISCUSSION

The MLaaS evaluation was categorized as below

Model Performance
Results show that all the models performed with an accuracy of above 75%. It is noted that ensemble methods namely Gradient Decision Tree, Random Forest and Gradient Boost gave a higher accuracy of greater than 80% with gradient boost giving the highest accuracy of 93.7%. It was observed that GCP Vertex AI gave the best performance with the least number of false positives and negatives.

Training Time and Prediction Time
Logistic regression had the least time to train and predict on all platforms between 3.4 sec-1.73 sec. This is attributed to the model complexity where logistic regression is not very complex, however it was seen to have the poorest performance. Models on cloud took less time to train which indicated that it's better to use MLaaS platforms when dealing with big data and resource intensive models.

User Experience
Microsoft Azure ML gave the best user experience as little coding is required on the canvas as well as it provides the option to code using the Azure terminal, VS code and Notebook hence gives several options to the user. Scheduling and model deployment was also noted to be a smooth and simple process. Azure was observed to provide a lot of flexibility compared to the rest of the platform.

Cost
Vertex AI was the most expensive training at $3.465 per hour, due to that it scales the training job amongst many instances at once. Azure was the cheapest followed by AWS. SageMaker and Azure, the cost is pretty much directly linked to the training time. The SageMaker instance that trained the models ran at a cost of $1.125 per hour and the Azure instance at $0.9 per hour.

Documentation
AWS SageMaker has the most documentation compared to Azure and Vertex AI hence makes it easier to implement AWS and get online solutions to problems frequently encountered.

Networks have been in existence for a long time however with the rapid growth in technology like the use of IoT devices and Edge computing Services, Network data is being produced in

great masses. The threat to networks is increasing and the need for an intelligent intrusion detection system is needed for the cyber security team and hence the framework is built to solve the problem by inspecting traffic traversing the network.
The proposed IDS can be:

- Placed at strategic points in the network as a NIDS (network-based intrusion detection)
- Installed on system computers connected to the network to examine inbound and outbound data on the network.
- Installed on each individual system as a HIDS (host-based intrusion detection)

The proposed framework will help Cyber Security Team to:

- Identify Security threats in the networks within a short period of time and hence reduce the delays and extent of damages that come from identifying malicious attacks late.
- The system provides comprehensive defense against identity theft, information mining, and network hacking. It monitors the network for malicious activity and protects it from unauthorized access with a detection accuracy of 99% and a false positive rate of 0.1%.
- Eliminate of on-premise data center associated risks as fire, faulty equipment etc., hence assurance of uptime and redundancy.
- Cost Savings : Since it is a pay as you go. This ensures you only pay for what you use.
- Faster Deployment : There is faster training and prediction time of the IDS model hence faster deployment.
- Increased Collaboration : Ease to synch files, workspaces in real time hence increase efficiency.

The above shows that our proposed framework can provide high scalability and performance in detecting malicious attacks in masses of network of data being generated in high velocity.

6. CONCLUSION

In this paper we propose the anomaly-based intrusion detection system on 3 different cloud platform (Vertex AI, Azure ML, AWS SageMaker) and compared the performance among them and with an on premise set up.

The system can manage large scale network packet analysis in a short period of time on different cloud platforms. Vertex AI provides the best accuracy and was the least costly. Azure ML performed the best in training and predicting time and offered the best user experience. AWS SageMaker was the fastest to set up due to availability of rich documentation. To test the system, we used the real IDS dataset provided by the Information Technology Department of Georgia Southern University. We built several classification models, and the decision tree performed the best based on accuracy and compute time.

We focused on designing the practical intelligent intrusion detection system with high accuracy of 93% and low false positive rate 7 %. In the future, we plan to use multimodal classifiers, introduce spark clusters and data balancing methods and extend to other cloud service providers, such as IBM.

7. REFERENCES

Berg, G. (2022). Image Classification with Machine Learning as a Service:-A comparison between Azure, SageMaker, and Vertex AI.

Kaymakci, C., Wenninger, S., Pelger, P., & Sauer, A. (2022). A systematic selection process of machine learning cloud services for manufacturing SMEs. Computers, 11(1), 14. https://doi.org/10.3390/computers11010014

Lee, Y.-S. (2018). Analysis on trends of machine learning-as-a-service. International Journal of Advanced Culture Technology, 6(4), 303-308. https://doi.org/10.17703//IJACT2018.6.4.303

Liberty, E., Karnin, Z., Xiang, B., Rouesnel, L., Coskun, B., Nallapati, R., . . . Das, P. (2020). Elastic machine learning algorithms in amazon sagemaker. Paper presented at the Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. https://doi.org/10.1145/3318464.3386126

Mäkelä, R. (2022). End-to-End Machine Learning Leveraging Cloud Service Providers: A Comparison of AWS and GCP (Bachelor's thesis).

Noshiri, N., Khorramfar, M., & Halabi, T. (2021). Machine Learning-as-a-Service Performance Evaluation on Multi-class Datasets. Paper presented at the 2021 IEEE International Conference on Smart Internet of Things (SmartIoT). DOI: 10.1109/SmartIoT52359.2021.00060

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). DOI: 10.1109/ICECET55527.2022.9872782

Xhepa, M., & Kanakala, N. S. (2022). Machine Learning Model Computation in AWS and Azure.

Yao, Y., Xiao, Z., Wang, B., Viswanath, B., Zheng, H., & Zhao, B. Y. (2017). Complexity vs. performance: empirical analysis of machine learning as a service. Paper presented at the Proceedings of the 2017 Internet Measurement Conference. https://doi.org/10.1145/3131365.3131372

Zhao, Q., Sun, J., Ren, H., & Sun, G. (2020). Machine-learning based TCP security action prediction. Paper presented at the 2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE). DOI: 10.1109/ICMCCE51767.2020.00291