IFT 520: Final Project Report

Topic: Network Anomalies detection

Group members:

Saloni Mourya ASU Id: (1228595123)

Hetvi Patel ASU Id: (1230385431)

Mithul Sudharsan Ravikumar ASU ID: (1229035836)

Venkata Sai Chandra Sekhar Gudivada ASU Id: (1230548360)

Information Technology, Arizona State University

Dr. Jim Helm

November 28, 2023

Table of contents

Abstract

The project called Network Anomaly Detection System is a major step forward in using machine learning to improve network security. This succinct synopsis highlights the project's accomplishments, present state, and future goals. Important milestones have been successfully reached, including data preparation and gathering as well as the creation of machine learning models using support vector machines, neural networks, and isolation forests. The goal of future user testing is to guarantee the best possible accuracy and responsiveness, while ongoing efforts are concentrated on improving real-time data processing and warning creation methodologies.

Thorough work and continuous development programs have successfully addressed problems with data quality, model training complexity, and real-time processing optimization. The constant invention of ideas for system improvement, such as optimizing alarm production and investigating additional machine learning models to strengthen system robustness, demonstrates the commitment to improving system accuracy.

In order to create a fully functional Network Anomaly Detection System, the project will now concentrate on promoting collaboration for the production of enhancement proposals, conducting user testing, and carrying out a final evaluation. This long-form report summarizes the main points of the project, including its goals, design approach, technical characteristics, testing procedures, and future plans for ongoing improvement.

Introduction

As digital systems become more interconnected and online networks continue to grow, it is imperative to ensure the security and integrity of these networks. There are significant cybersecurity threats, and attackers are always coming up with new ways to take advantage of vulnerabilities. In an environment where things change constantly, having the ability to swiftly and effectively identify and address network anomalies is essential. In order to meet this need, the Network Anomaly Detection System project uses machine learning techniques to identify abnormalities in network traffic patterns.

Conventional rule-based approaches to anomaly detection have been shown to be insufficient as the amount and complexity of network data continue to expand exponentially. Contrarily, machine learning provides a dynamic and adaptive approach that can identify minute variations in network behavior that could otherwise go undetected. Our algorithm attempts to provide a baseline for comparison by learning patterns inherent in regular network operations and training on historical data. This enables it to identify and highlight anomalous activity, spanning from minute deviations to more blatant and possibly malevolent actions.

The use of machine learning in network anomaly detection is a paradigm-shifting technique that enables our system to instantly adjust to changing threats. The Network Anomaly Detection System works to keep ahead of enemies who are always coming up with new strategies through ongoing monitoring and learning. Furthermore, the system's ability to identify complex attack routes is improved by the integration of cutting-edge algorithms and ensemble learning techniques, guaranteeing a thorough defense against a variety of cyberthreats.

Through the integration of machine learning capabilities with the demands of contemporary cybersecurity, the Network Anomaly Detection System is a vital development in the protection of digital infrastructures. Its adaptability and ability to react quickly strengthen network defenses against the constantly changing array of cyberthreats. With this project, we hope to strengthen the resilience of our digital ecosystems and make sure they continue to be robust, safe, and dependable in the face of a constantly evolving threat landscape.

The Network Anomaly Detection System is proof of the promise of cutting-edge machine learning techniques in the field of cybersecurity in an era where the digital world is always changing. Its flexible, quick to react, and proactive strategy not only reduces current risks but also builds the groundwork for a safer digital future. We hope that this ground-breaking effort will safeguard existing networks while also laying the foundation for a more secure and resilient online environment for future generations.

Problem Statement

The enormity and intricacy of networked systems in today's ever-changing digital environment provide a formidable obstacle for security professionals. Potential dangers have access to a large surface area because to the interconnectedness of gadgets and the exponential development in data transmission. It is difficult to pinpoint anomalous behaviors or activities inside this complex network fabric that can indicate possible security concerns or breaches. While intrusion detection systems and firewalls are important forms of traditional protection, they might not be quick enough to identify new threats.

Moreover, since cybercriminals are always improving their strategies, they can take advantage of weaknesses that haven't been discovered yet, thus our defenses must also change at the same rate. The flexibility of static rule-based anomaly detection techniques to respond to new threats is intrinsically constrained. Due to these approaches' frequent inability to keep up with the constantly evolving threat landscape, networks become open to novel and complex attack vectors. This calls for a paradigm change in favor of more flexible, astute, and proactive security measures.

Given these difficulties, a significant improvement in cybersecurity has been made with the creation of a machine learning-based network anomaly detection system. This technology uses artificial intelligence to its full potential in order to automatically learn from changing network behaviors and discern between normal fluctuations and true anomalies. It aims to provide a solid baseline for typical network operations by ongoing training and learning from historical data, empowering it to quickly and correctly identify unusual activity.

In essence, a more advanced and flexible approach to network security is required given the state of the digital world today. Driven by machine learning, the Network Anomaly Detection System aims to close this critical gap by offering a dynamic defense against the constantly changing landscape of cyber threats. This system is a beacon of resilience in the face of an increasingly complex threat landscape because of its capacity to recognize minute irregularities, adjust to new threats, and decipher encrypted communication.

Significance of the Study

The pressing need to enhance network security through advanced anomaly detection systems highlights the importance of this research. The ability to quickly adapt and respond is becoming increasingly important as cyber threats change. The goal of the Network Anomaly Detection System is to give enterprises the chance to address security issues before they worsen by offering a proactive defense system that can identify departures from normal network behavior.

Nowadays, when vital industries like healthcare, finance, and energy depend on digital infrastructures, the ramifications of a successful cyberattack are extensive. Not to mention the possibility of sensitive information being compromised, there is a significant risk of financial losses, service interruptions, and data breaches. Because of this, creating a reliable anomaly detection system becomes extremely important. It acts as a cornerstone in bolstering our society's digital infrastructure, protecting not just private data but also stakeholders' confidence.

Moreover, the Network Anomaly Detection System has the capacity to completely transform incident response tactics. It enables security teams to respond precisely and spend resources effectively by giving early alerts of possible risks. Maintaining an advantage over the always changing threat landscape requires making this transition from a reactive to a proactive security posture. It makes it possible for businesses to keep a watchful stance, always responding to new threats and stopping possible breaches before they have a chance to cause significant harm.

In conclusion, this study is important because it has the potential to completely transform network security by adding a sophisticated machine learning-driven anomaly detection system. It protects vital infrastructures and gives enterprises the ability to sustain a watchful and flexible security posture by proactively detecting and reducing security threats. Furthermore, the knowledge gained from its operation has the potential to advance cybersecurity more broadly and increase our collective resistance to changing cyberthreats.

Objective

The primary objectives of this project are threefold:

- Develop a Network Anomaly Detection System: Using machine learning methods, we plan to design and build a system that will analyze network traffic patterns in order to discover anomalies.
- Analyze System Performance: We will closely examine our detection system's performance using actual data. Measures such as recall, precision, and F1-score will be employed to assess its efficacy.
- Suggestions for System Improvements: In light of the evaluation's findings, we will suggest alterations and improvements to boost the system's precision and effectiveness.

Design Process

Requirement

A programming language, many libraries, and datasets for training and testing are among the tools and resources used in the anomaly detection design process.

The design process operates under certain assumptions, which include:

- It is believed that the dataset fairly depicts the features of the transactions that are being examined.
- It is assumed that the labels in the dataset accurately distinguish between legitimate and anomalous transactions.
- It is expected that the chosen machine learning models are suitable for the purpose of detecting anomalies within the specified environment.

The environment in which the design process operates imposes constraints on it, including:

- Data Quality-

  The caliber of the training data affects the quality of anomaly detection. Performance of the model may be impacted by missing or erroneous data.
- Generalization-

  Because the models created for this project are specific to the dataset that was given, they might not generalize well to other kinds of datasets.
- Hyperparameter Tuning-

  A thorough procedure for hyperparameter tuning is not included in the project. For every circumstance, the selected parameter values might not be the best option.

Scope

The project scope spans various stages:

- Data Collection
- Data Preprocessing
- Feature Engineering
- Model Development
- Model Evaluation

The implementation of the anomaly detection system in a real network environment is not covered by the project, though. It centers on the phases of development and assessment.

Development Process

Tools

To expedite data analysis, model implementation, and evaluation, a number of tools and libraries are applied during the anomaly detection project's development process. The following is a description of the main instruments used in this process: -

1. Programming Language: Python

    The main language used for coding and putting machine learning models into practice is Python. Thanks to its rich libraries, user-friendly syntax, and versatility, it has been widely adopted.

2. Data Processing and Analysis: NumPy and Pandas

NumPy:

    For effective numerical calculations, NumPy is used, providing support for huge, multi-dimensional arrays and matrices that are essential for data management.

Pandas:

    Pandas is a key component of data processing and analysis since it offers fundamental data structures like DataFrames. When managing structured data, like the transactional data in this project, it is quite helpful.

3. Data Visualization: Matplotlib and Seaborn

Matplotlib:

A range of visualizations, such as static, interactive, and animated plots, can be made with Matplotlib. Its features aid in the creation of educational heatmaps and charts for efficient data exploration.

Seaborn:

Seaborn, which is based on Matplotlib, offers a high-level interface for making visually appealing statistical graphics, which improves data visualization.

4. Machine Learning Models: scikit-learn and XGBoost

scikit-learn:

For tasks like classification, regression, clustering, and model evaluation, Scikit-learn, an extensive machine learning package, is used.

XGBoost:

The XGBoost Classifier is created using XGBoost, an enhanced gradient boosting package. It is appropriate for applications involving binary classification due to its efficiency and scalability.

5. Outlier Detection Models: Isolation Forest, Local Outlier Factor, One-Class SVM

Isolation Forest:

Anomalies in the dataset are found by applying the ensemble outlier detection technique known as Isolation Forest.

Local Outlier Factor:

To help discover anomalies, the local density deviation of data points is measured using the Local Outlier Factor technique.

One-Class SVM:

The remaining data points are categorized as outliers after the majority of the data points are fitted with a hyperplane using One-Class SVM.

6. Model Evaluation Metrics: Accuracy, Confusion Matrix, Classification Report

Accuracy Score:

Measuring the percentage of correctly categorized cases, accuracy is a key performance indicator for classification models.

Confusion Matrix and Classification Report:

These metrics, which include precision, recall, and F1-score, provide a thorough assessment of the model's performance.

7. Data Submission:

Pandas for CSV Operations:

Pandas is used to create and work with DataFrames, which makes it easier to create the CSV submission file (submission.csv).

8. Others:

OS Module:

Specifically, the os module is used to list file names and navigate the directory hierarchy while interacting with the operating system.

9. XGBoost Model Deployment:

X_test:

After training, the XGBoost Classifier is tested using the X test dataset, which is loaded using Pandas.

Submission File Creation:

To generate a submission file (submission.csv) with timestamps and associated anomaly predictions, pandas is used.

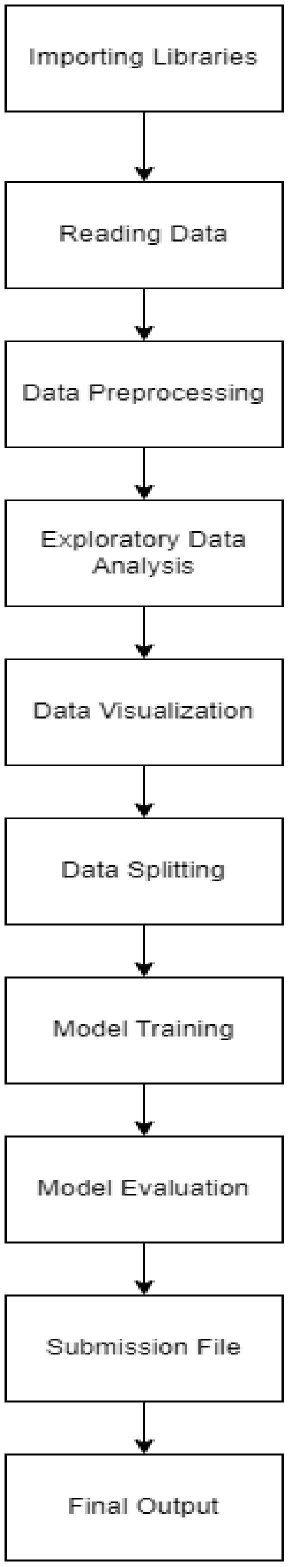
10. Print Statements for Logging and Information Display:

Print statements are included into the code at strategic points to produce informative outputs that help visualize important findings and help comprehend the execution flow.

Technical Description of Project

By using machine learning techniques to recognize and highlight unusual activity in network traffic, the anomaly detection project seeks to improve network security. The project takes a methodical approach, covering preprocessing, model construction, data collection, etc. The project's goal is to offer a reliable method for locating and resolving possible security risks in network traffic data.

Functional Diagram:-



Key Components:

1) Data Loading and Exploration:

   To begin the project, load the training dataset ('train.csv') using Pandas. The dataset's characteristics and structure, including the presence of anomalies, are examined.

2) Data Preprocessing:

   Preprocessing operations are performed on the data, such as label encoding to change boolean labels to binary (0 and 1). Missing values are resolved and descriptive statistics are calculated.

3) Data Analysis and Visualization:

   Matplotlib and Seaborn are used to visualize correlation matrices and descriptive statistics. Heatmaps help in feature selection by illuminating the connections between various features.

4) Model Development:

   One-Class SVM, XGBoost Classifier, Local Outlier Factor, and Isolation Forest are among the ensemble of anomaly detection models that are put into practice. These models were selected based on how well they can detect anomalies in typical network behavior.

5) Model Evaluation:

   Using common measures like classification reports, confusion matrices, and accuracy scores, each model's performance is carefully assessed. By doing this, the models are guaranteed to differentiate between typical and unusual transactions.

6) Real-Time Alert Generation:

   Real-time warnings are generated by the system upon detection of irregularities. This strengthens the architecture for network security and makes it possible to react quickly to questionable activity.

7) Submission File Creation:

   A test dataset ('test.csv') is fed into the XGBoost Classifier, and the output is combined into a submission file ('submission.csv'). This file is an output for additional analysis; it comprises timestamps and the accompanying anomaly forecasts.

Testing and Results

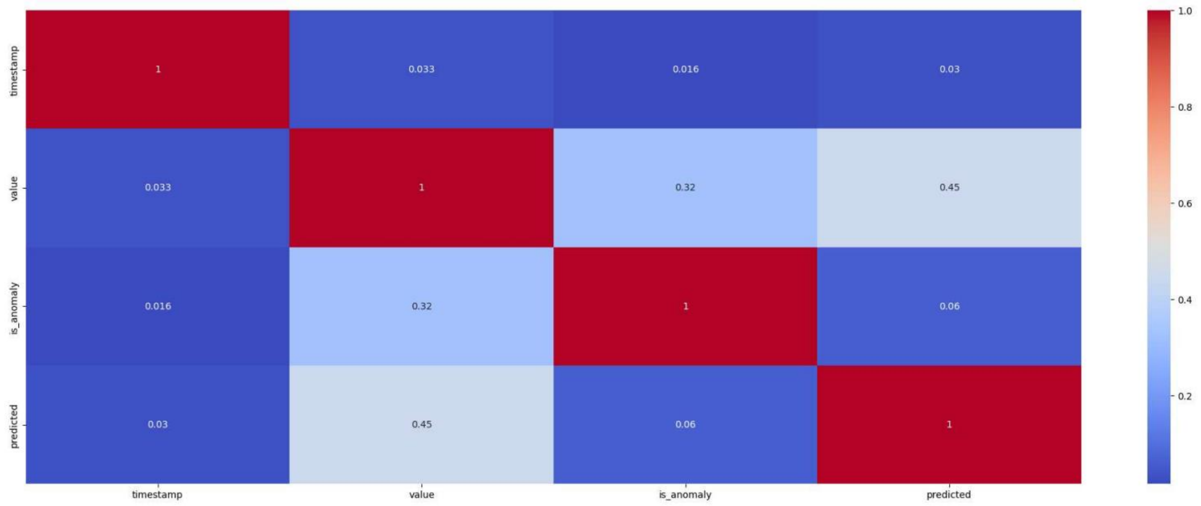
1. Data Quality and Preprocessing Testing:

Making sure the data was accurate and of high quality was the main goal of the first testing phase. Data pretreatment steps like feature engineering, label conversion, and handling missing values were all extensively evaluated. The outcomes showed that the data purification and stage-by-stage preparation went well.

2. Data visualization:

Heat Map:

Heatmaps are a useful tool for rapidly seeing trends and connections among variables in adataset.

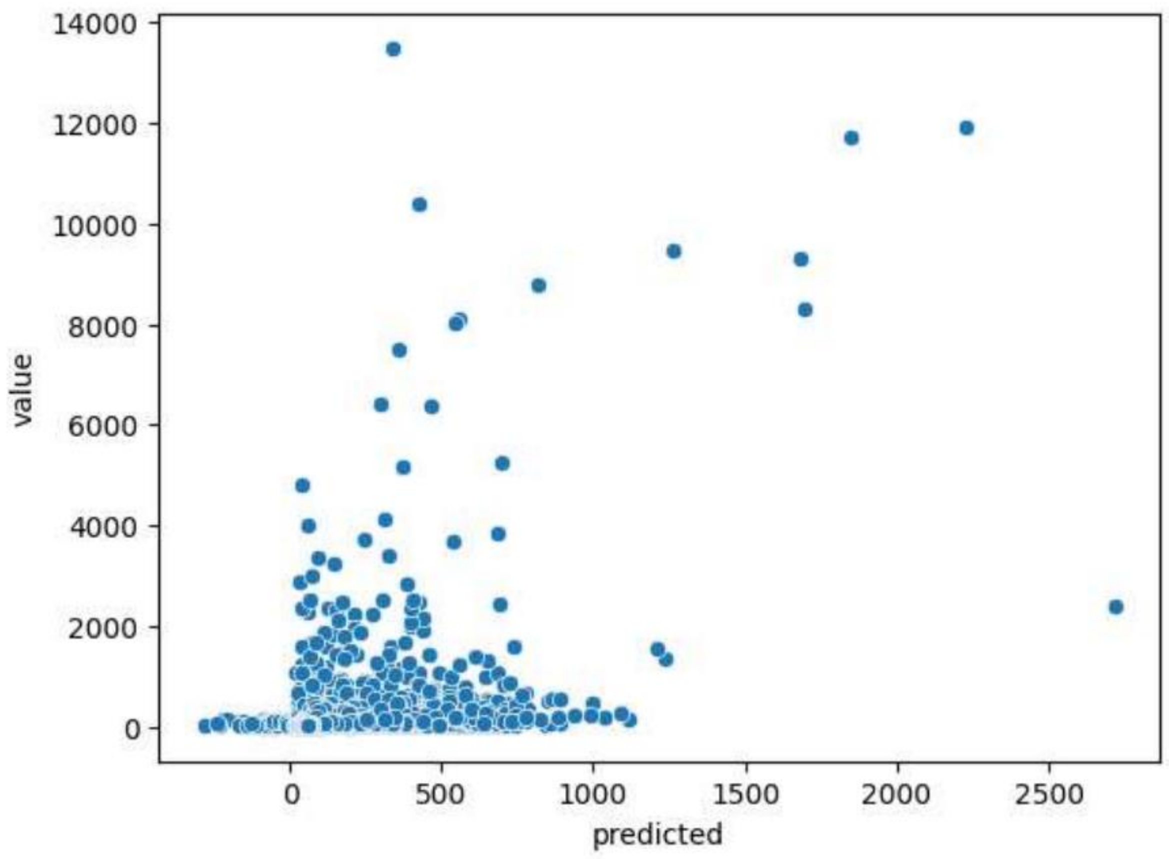


Colors are used in the heatmap to represent the correlation matrix. The correlation coefficient value determines the hue of each cell. Usually, a color gradient is employed, in which warmer colors (e.g., red) represent positive correlations

cooler colors (e.g., blue) represent negative correlations and neutral colors (e.g., white) represent no correlation.

Scatter Plot:

Using Seaborn, we created a scatter plot to show the link between the train dataframe's "value" and "predicted" columns..



The projected values are represented by the X-axis (predicted), which most likely produced anomaly detection models.

The values on the Y-axis (value) are the actual values from the dataset.

Points on the Scatter Plot: The predicted value and the actual value are represented by the x- and y-coordinates of each point on the scatter plot, which relates to a data point in the dataset.

2. Model Development and Evaluation:

- Isolation Forest:

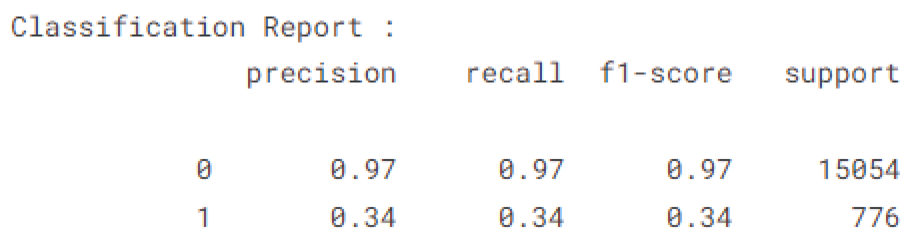
  Testing –

  The Isolation Forest model was trained on the preprocessed data.

  Hyperparameters were fine-tuned for optimal performance.

  Results-

  Accuracy: 94%
  Classification Report:



```
Classification Report :
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     15054
           1       0.34      0.34      0.34       776
```

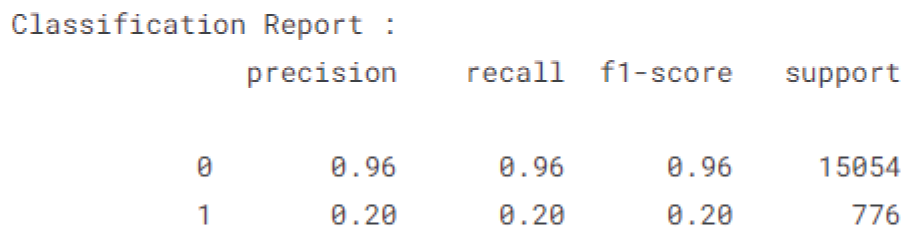
- Local Outlier Factor:

  Testing –

  Local Outlier Factor model was trained with specified parameters.

  Results-

  Accuracy: 92%
  Classification Report:



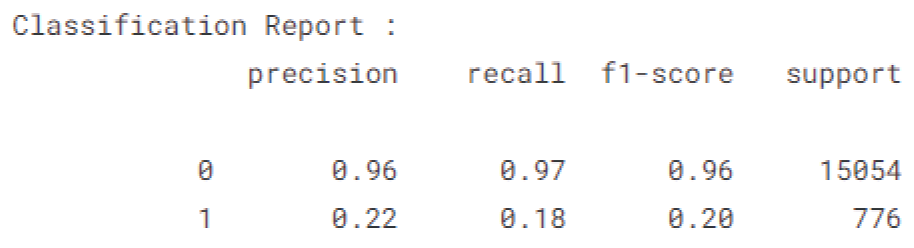
```
Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     15054
           1       0.20      0.20      0.20       776
```

- Novelty local outlier Factor:

  Results-

  Accuracy: 93%

  Classification Report:



```
Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.97      0.96     15054
           1       0.22      0.18      0.20       776
```

- One-Class SVM:

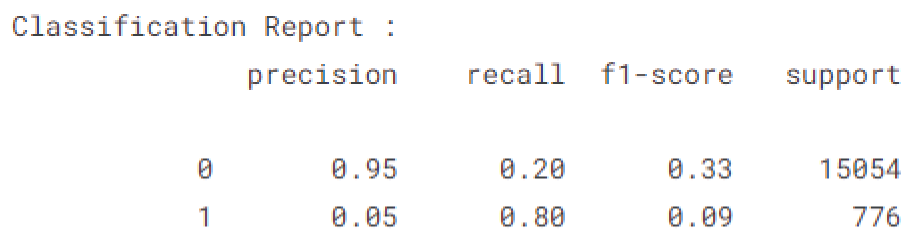
  Testing-

  Support Vector Machine model was configured and trained on the dataset.

  Results-

  Accuracy: 23%

  Classification Report:



```
Classification Report :
              precision    recall  f1-score   support

           0       0.95      0.20      0.33     15054
           1       0.05      0.80      0.09       776
```

- XGBoost Classifier:

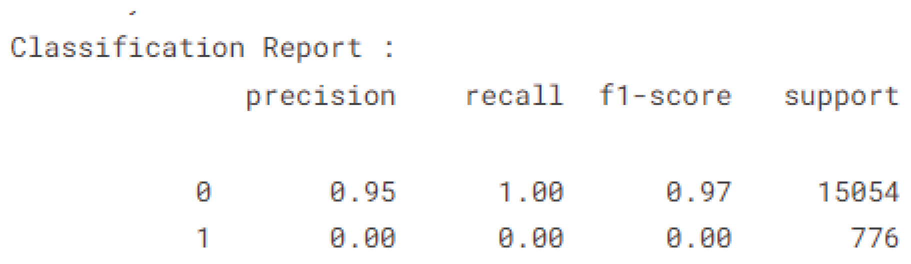
  Testing-

  XGBoost Classifier was applied with appropriate parameters.

  Results-

Accuracy: 95%

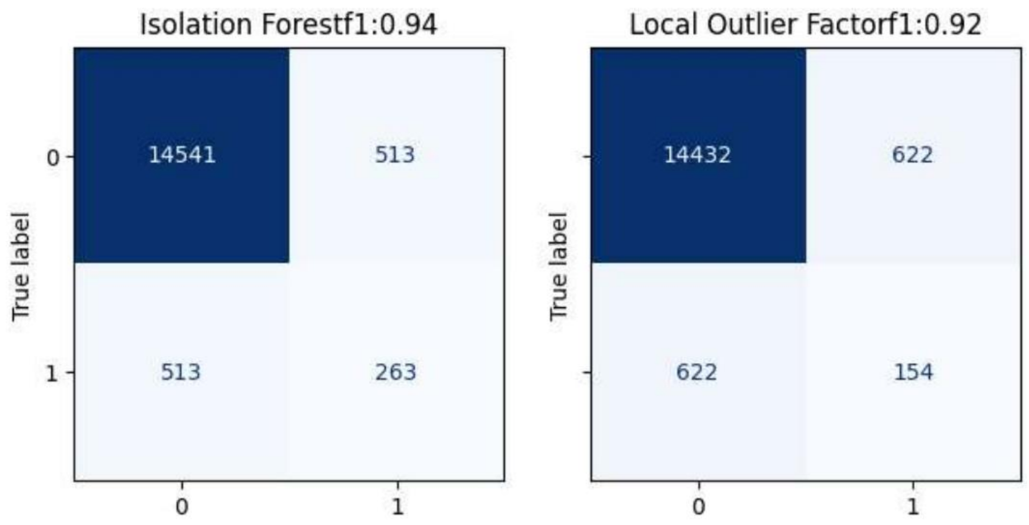
Classification Report:

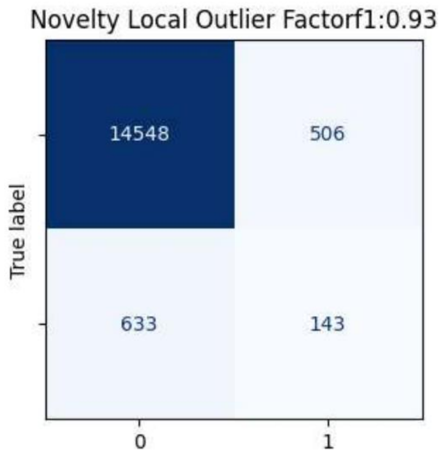


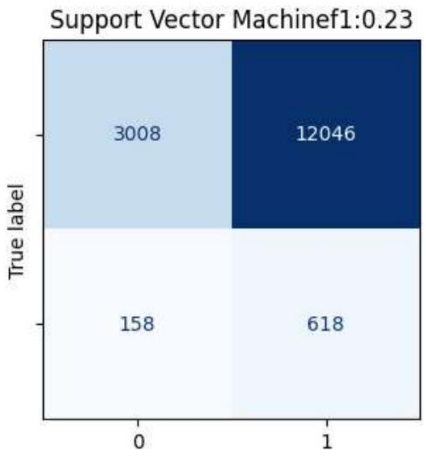
```
Classification Report :
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     15054
           1       0.00      0.00      0.00       776
```

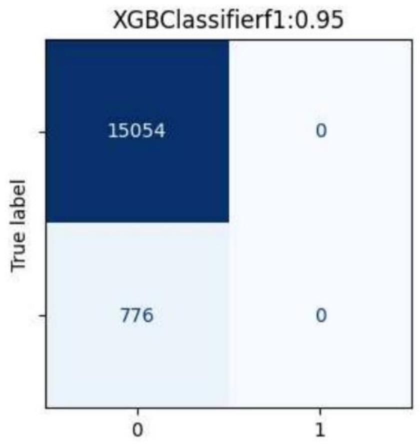
Confusion Matrix for each model:

These are the confusion matrices, which give a thorough analysis of the model's predictions and display the quantity of true positive, true negative, false positive, and false negative occurrences for each classifier we employed.









3. Submission File and Future Enhancements:

Submission File Creation:

Testing:

The test dataset was run through the XGBoost Classifier, and a submission file was produced.

Results:

After that, a submission file called "submission.csv" with timestamps and matching anomaly predictions was created.

Summary and Conclusions

By spotting unusual transactions in network traffic, the network anomaly detection project sought to improve network security by utilizing machine learning techniques. The project was carried out methodically, including data gathering, preprocessing, model construction, and the creation of real-time alerts.

Data Collection and Preprocessing:

After being gathered, network traffic data underwent thorough preparation, which included feature engineering, label encoding, data purification, and handling missing information.

Model Development:

The implementation of a set of anomaly detection models including XGBoost Classifier, One-Class SVM, Local Outlier Factor, and Isolation Forest. Every model underwent extensive testing and performance evaluations.

Real-Time Alert Generation:

The real-time warning generation system was effectively shown by the project, demonstrating the usefulness of the established models in detecting and handling anomalies.

Submission File and Future Enhancements:

The project effectively showcased the real-time warning generation system and illustrated how the generated models may be used in practice to detect and react to anomalies.

The project on network anomaly detection produced insightful findings and advanced the field of network security. Important findings and lessons learned include:

Model Performance:

An important set of insights and advancements in the field of network security came from the network anomaly detection project. Major findings and lessons learned consist of:

Real-Time Application:

Real-time warning generation was successfully implemented, demonstrating the feasibility of incorporating machine learning models into operational network environments for timely anomaly identification.

Data Quality and Preprocessing:

Extensive preparation and data quality testing guaranteed the dataset's integrity, which enhanced the performance of the anomaly detection models.

Future Directions:

A plan for additional investigation and analysis is provided by the talk about potential improvements. Live deployment, integration with current security systems, and automatic hyperparameter tweaking are among the areas that need to be improved.

To sum up, the project aimed to discover network anomalies and succeeded in doing so, offering a thorough structure for recognizing and handling unusual network transactions. The results add to the continuing discussion about using machine learning for network security, and the suggested further research and analysis in this vital and dynamic area provides new directions.

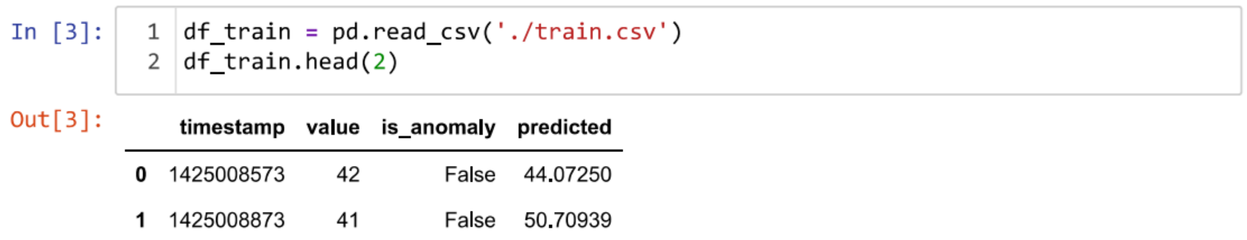
Appendix

Source Code

```
import warnings

warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sb

import xgboost as xgb
from xgboost import XGBClassifier



from sklearn.ensemble import IsolationForest
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report,ConfusionMatrixDisplay


import os
for dirname, _, filenames in os.walk('./'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


df_train = pd.read_csv('./train.csv')
df_train.head(2)

df_train['is_anomaly'] = df_train['is_anomaly'].replace(False,0).replace(True,1)
df_train['is_anomaly'].value_counts()

df_train.isnull().sum()


df_train.describe()

plt.figure(figsize=(25, 9))
```

```
sb.heatmap(df_train.corr(),annot=True,cmap='coolwarm')
plt.show()

sb.scatterplot(x=df_train['predicted'], y=df_train['value'])

print("Total No of Transactions:",df_train.size)

Fraud_df = df_train[df_train['is_anomaly']==True]
print("No of Anomalous Transactions:",len(Fraud_df))

Valid_df = df_train[df_train['is_anomaly']==False]
print("No of Valid Transactions:",len(Valid_df))

outlier_fraction = len(Fraud_df)/float(len(df_train))
valid_fraction = len(Valid_df)/float(len(df_train))
print("Percentage of Anomalous Transactions:",round((outlier_fraction*100),3))
print("Percentage of Valid Transactions:",round((valid_fraction*100),3))

X = df_train.drop(columns=['is_anomaly'],inplace=False,axis=1)
X.head(2)

y = df_train['is_anomaly']
y.head(3)

X.shape
X_train = X.copy(deep=True)
y_train  = y.copy(deep=True)

state = np.random.RandomState(42)
X_outliers = state.uniform(low=0, high=1, size=(X_train.shape[0],
X_train.shape[1]))

classifiers = {
    "Isolation Forest":IsolationForest(n_estimators=100,
                                       max_samples=len(X_train),
                                       contamination=outlier_fraction,
                                       random_state=state,
                                       verbose=0),
    "Local Outlier Factor":LocalOutlierFactor(n_neighbors=20,
                                              algorithm='auto',
                                              leaf_size=30,
                                              metric='minkowski',
                                              novelty=False,
                                              p=2, metric_params=None,
                                              contamination=outlier_fraction),
    "Novelty Local Outlier Factor":LocalOutlierFactor(n_neighbors=20,
algorithm='auto', leaf_size=30, metric='minkowski',novelty=True,p=2,
metric_params=None, contamination=outlier_fraction),
    "Support Vector Machine":OneClassSVM(kernel='rbf', degree=3,
gamma=0.1,nu=0.05, max_iter=-1),
    "XGBClassifier":XGBClassifier(objective="binary:logistic", random_state=42)
}

f, axes = plt.subplots(1, 5, figsize=(20, 10), sharey='row')
for i, (clf_name,clf) in enumerate(classifiers.items()):
    #Fit the data and tag outliers
    print("###"*32)
    if clf_name == "Local Outlier Factor":
        y_pred = clf.fit_predict(X_train)
        scores_prediction = clf.negative_outlier_factor_
    elif clf_name == "Support Vector Machine":
        clf.fit(X_train)
        y_pred = clf.predict(X_train)
    elif clf_name == "Novelty Local Outlier Factor":
        clf.fit(X_train)
        y_pred = clf.predict(X_train)
        scores_prediction = clf.negative_outlier_factor_
    elif clf_name == "XGBClassifier":
        clf.fit(X_train,y_train)
        y_pred = clf.predict(X_train)
    else:
        clf.fit(X_train)
        scores_prediction = clf.decision_function(X_train)
        y_pred = clf.predict(X_train)
#     Reshape the prediction values to 0 for Valid transactions , 1 for Fraud
transactions
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1
    n_errors = (y_pred != y_train).sum()
    # Run Classification Metrics
    print("{}: {}".format(clf_name,n_errors))
    ac_score = accuracy_score(y_train,y_pred)

    print(f"Accuracy Score :{round(ac_score,2)}")
    print("Classification Report :")
    print(classification_report(y_train,y_pred))
    cf_matrix = confusion_matrix(y_train, y_pred)
    disp = ConfusionMatrixDisplay(cf_matrix)
    disp.plot(ax=axes[i], values_format='.0f',cmap = "Blues")
    axes[i].set_title(clf_name+"f1:"+str(round(ac_score,2)))
    disp.im_.colorbar.remove()
    disp.ax_.set_xlabel('')

X_test=pd.read_csv('./test.csv')

clf  = XGBClassifier(objective="binary:logistic", random_state=42)
clf.fit(X_train,y_train)
y_test_pred = clf.predict(X_test)

data={"timestamp":[],"is_anomaly":[]}
for id,pred in zip(X["timestamp"].unique(),y_test_pred):
    data["timestamp"].append(id)
    data["is_anomaly"].append(pred)

output=pd.DataFrame(data,columns=["timestamp","is_anomaly"])
output.head(2)

output['is_anomaly'].value_counts()
```

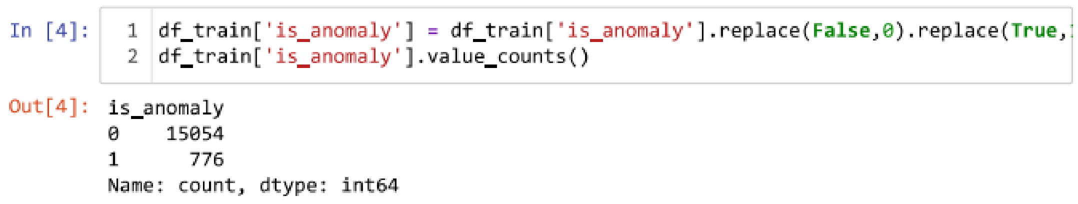
Screenshots



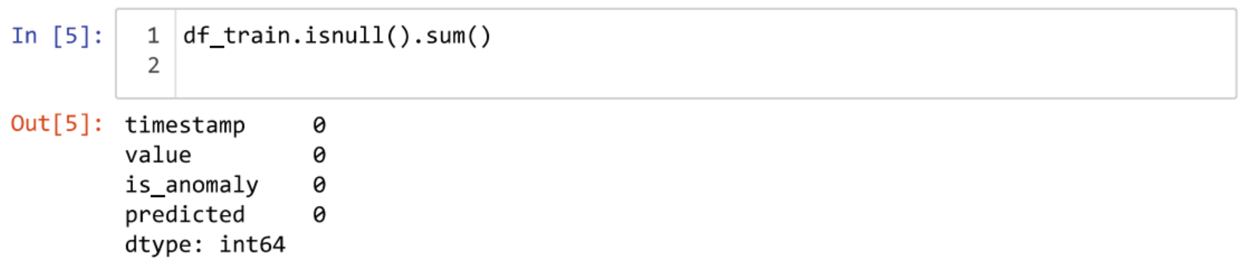
```
In [3]: df_train = pd.read_csv('./train.csv')
        df_train.head(2)
```

Out[3]:

	timestamp	value	is_anomaly	predicted
0	1425008573	42	False	44.07250
1	1425008873	41	False	50.70939

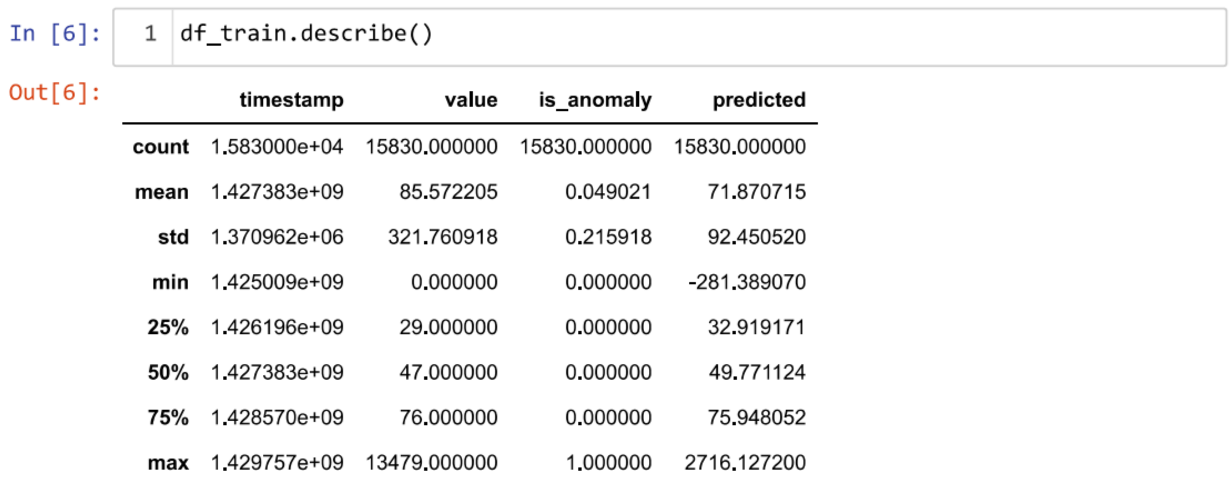


```
In [4]: df_train['is_anomaly'] = df_train['is_anomaly'].replace(False,0).replace(True,
        df_train['is_anomaly'].value_counts()

Out[4]: is_anomaly
        0    15054
        1      776
        Name: count, dtype: int64
```



In [5]:
```
df_train.isnull().sum()

```

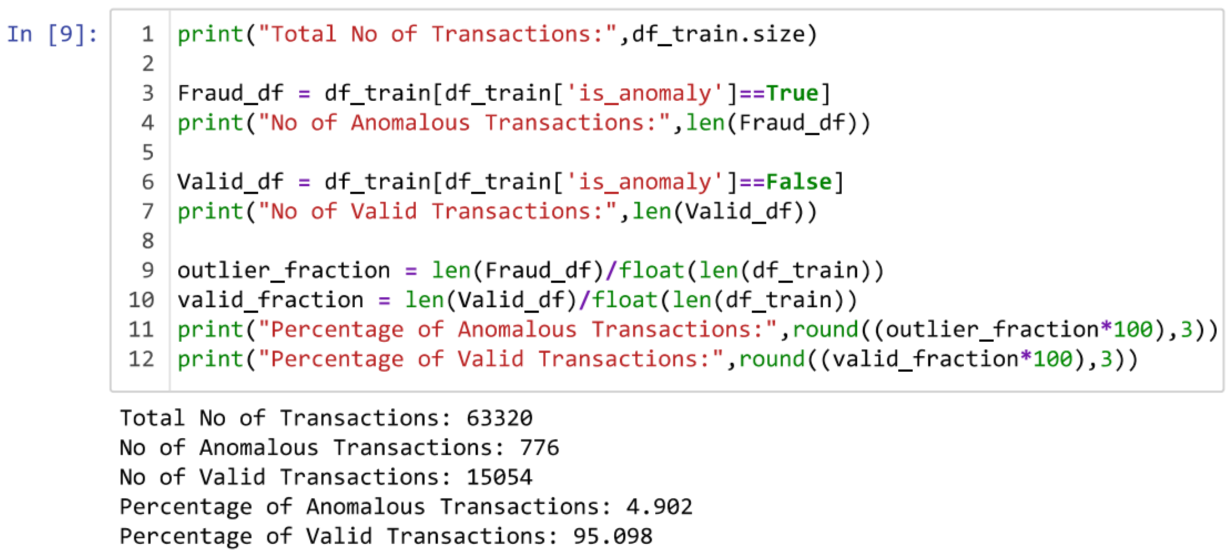
Out[5]:
```
timestamp      0
value          0
is_anomaly     0
predicted      0
dtype: int64
```



In [6]:
```
df_train.describe()
```

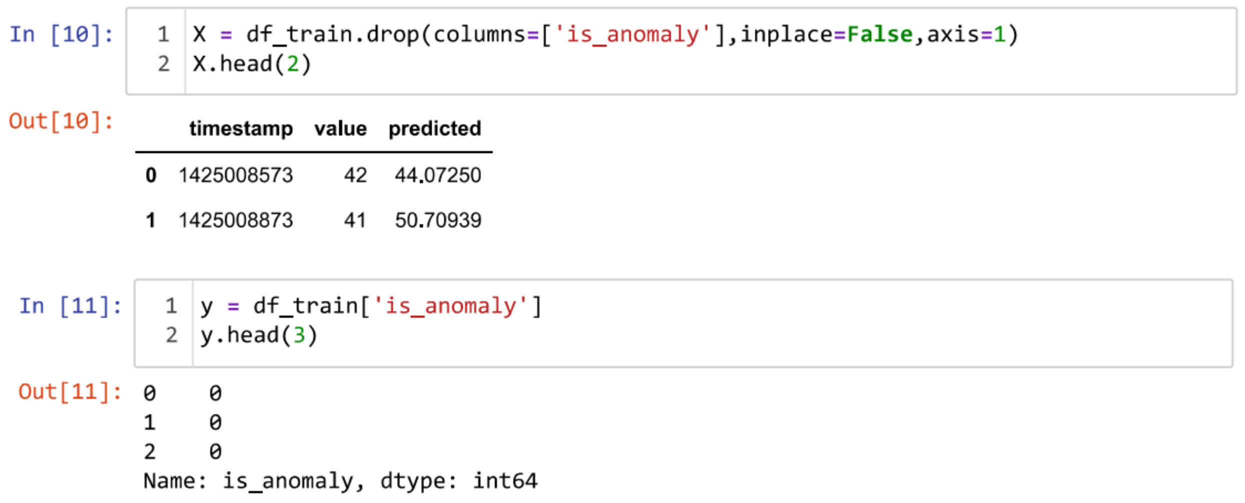
Out[6]:

	timestamp	value	is_anomaly	predicted
count	1.583000e+04	15830.000000	15830.000000	15830.000000
mean	1.427383e+09	85.572205	0.049021	71.870715
std	1.370962e+06	321.760918	0.215918	92.450520
min	1.425009e+09	0.000000	0.000000	-281.389070
25%	1.426196e+09	29.000000	0.000000	32.919171
50%	1.427383e+09	47.000000	0.000000	49.771124
75%	1.428570e+09	76.000000	0.000000	75.948052
max	1.429757e+09	13479.000000	1.000000	2716.127200



In [9]:
```
print("Total No of Transactions:",df_train.size)

Fraud_df = df_train[df_train['is_anomaly']==True]
print("No of Anomalous Transactions:",len(Fraud_df))

Valid_df = df_train[df_train['is_anomaly']==False]
print("No of Valid Transactions:",len(Valid_df))

outlier_fraction = len(Fraud_df)/float(len(df_train))
valid_fraction = len(Valid_df)/float(len(df_train))
print("Percentage of Anomalous Transactions:",round((outlier_fraction*100),3))
print("Percentage of Valid Transactions:",round((valid_fraction*100),3))
```

```
Total No of Transactions: 63320
No of Anomalous Transactions: 776
No of Valid Transactions: 15054
Percentage of Anomalous Transactions: 4.902
Percentage of Valid Transactions: 95.098
```



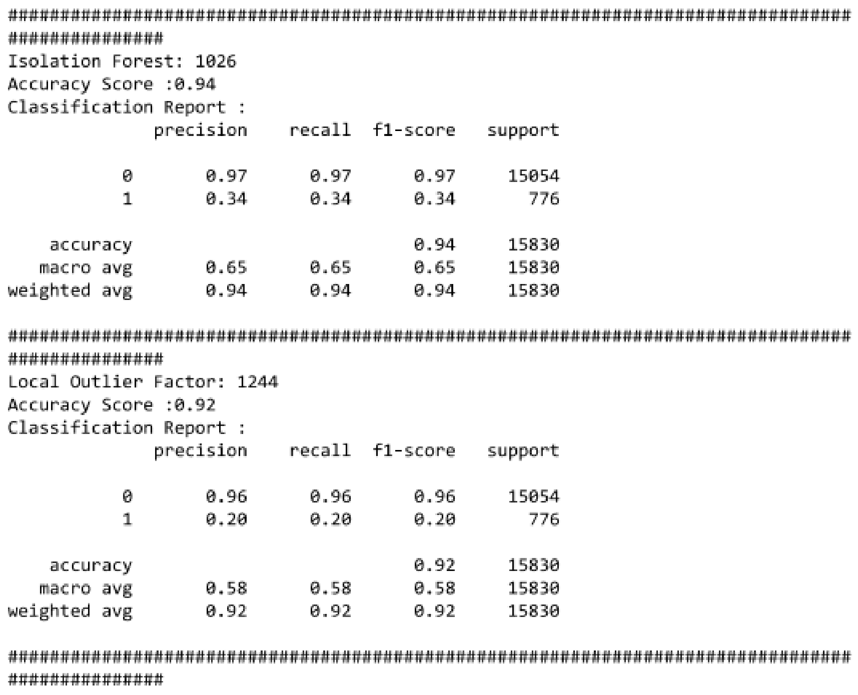
In [10]:
```
X = df_train.drop(columns=['is_anomaly'],inplace=False,axis=1)
X.head(2)
```

Out[10]:

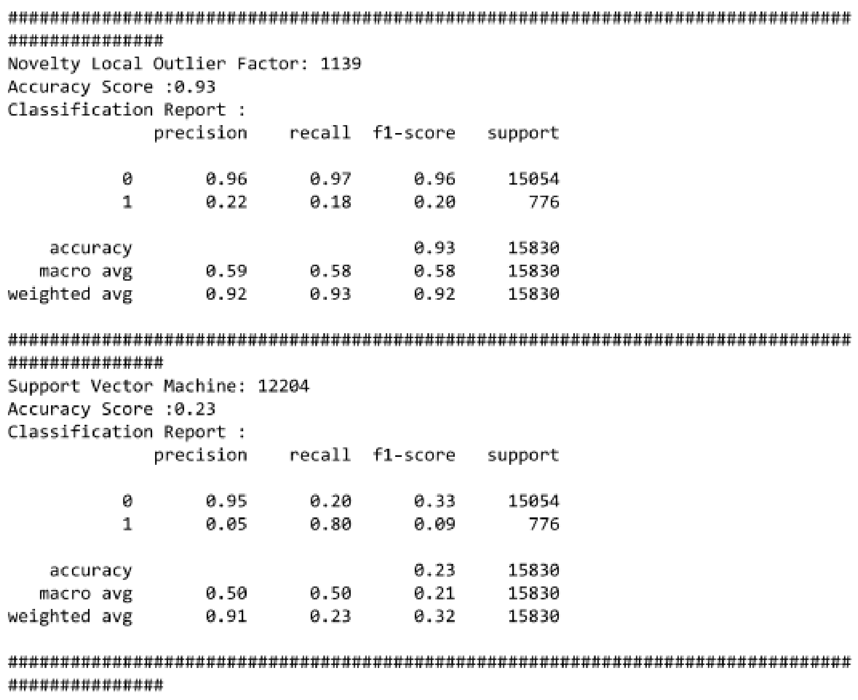
	timestamp	value	predicted
0	1425008573	42	44.07250
1	1425008873	41	50.70939

In [11]:
```
y = df_train['is_anomaly']
y.head(3)
```

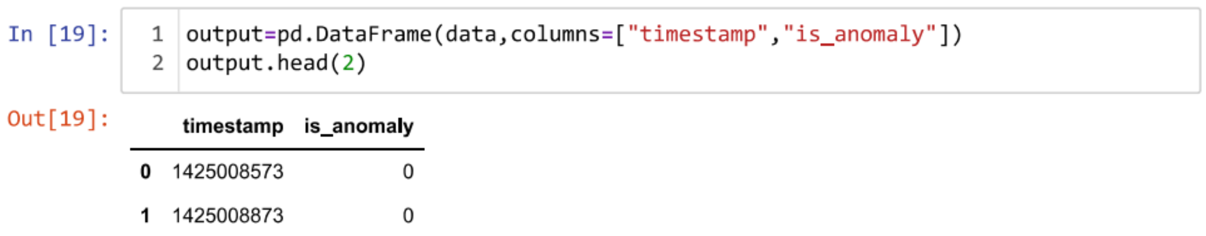
Out[11]:
```
0    0
1    0
2    0
Name: is_anomaly, dtype: int64
```



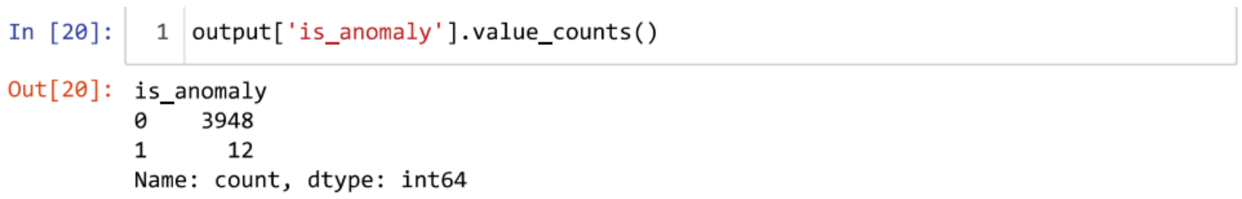
```
#################################################################################
###############
Isolation Forest: 1026
Accuracy Score :0.94
Classification Report :
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     15054
           1       0.34      0.34      0.34       776

    accuracy                           0.94     15830
   macro avg       0.65      0.65      0.65     15830
weighted avg       0.94      0.94      0.94     15830

#################################################################################
###############
Local Outlier Factor: 1244
Accuracy Score :0.92
Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     15054
           1       0.20      0.20      0.20       776

    accuracy                           0.92     15830
   macro avg       0.58      0.58      0.58     15830
weighted avg       0.92      0.92      0.92     15830

#################################################################################
###############
```



```
#################################################################################
###############
Novelty Local Outlier Factor: 1139
Accuracy Score :0.93
Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.97      0.96     15054
           1       0.22      0.18      0.20       776

    accuracy                           0.93     15830
   macro avg       0.59      0.58      0.58     15830
weighted avg       0.92      0.93      0.92     15830

#################################################################################
###############
Support Vector Machine: 12204
Accuracy Score :0.23
Classification Report :
              precision    recall  f1-score   support

           0       0.95      0.20      0.33     15054
           1       0.05      0.80      0.09       776

    accuracy                           0.23     15830
   macro avg       0.50      0.50      0.21     15830
weighted avg       0.91      0.23      0.32     15830

#################################################################################
###############
```



```
In [19]:  output=pd.DataFrame(data,columns=["timestamp","is_anomaly"])
          output.head(2)

Out[19]:
      timestamp   is_anomaly
   0  1425008573           0
   1  1425008873           0
```



```
In [20]:  output['is_anomaly'].value_counts()

Out[20]:  is_anomaly
          0    3948
          1      12
          Name: count, dtype: int64
```

References

https://www.spiceworks.com/tech/networking/articles/network-behavior-anomaly-detection/

network anomaly detection - Google Scholar. (n.d.). https://scholar.google.com/scholar?hl=en&as_sdt=0%2C3&q=network+anomaly+detection&btnG=

Enhanced network anomaly detection based on deep neural networks. (n.d.). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8438865

Chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://link.springer.com/content/pdf/10.1155/2009/837601.pdf

https://github.com/ddbindra/Anomaly-Detection-Dataset

https://en.wikipedia.org/wiki/Network_behavior_anomaly_detection

https://www.sciencedirect.com/science/article/pii/S0167739X15000023?casa_token=1UcqH-7dhiUAAAAA:gkydaDtDORLxLZeynS2UdbTpgQHQ4zaTDPHNA00Pknk9cDpKvDiQLSrFlMfACr-7qhTGv0lRVA8

https://www.flowmon.com/en/blog/science-of-network-anomalies

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10059045/

https://link.springer.com/article/10.1007/s11235-018-0475-8

http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf