**Project Report by**

***Vignesh Sakthivel***

**Deliverable 1:**

* **Project Description and Objective**
* The given dataset contains information related to cybersecurity classification.
* The training data set consists of 200 predictor values and 180,000 observations.
* The testing data set consists of 200 predictor values and 10,000 observations.
* The objective is to classify whether the activity recorded on network nodes is Malicious or not.
* **Data Visualization**

The given data had been explored and for further clarity and understanding we proceeded with visualization. Starting from the number of counts, if it is malicious or just normal activity, all this was visualized using the SNS count plot after dropping all null values from the training data set.

This is the SNS plot representing the count of responses. It depicts the majority and minority of responses registered in the training data set. Shape

Description automatically generated with medium confidence

We needed to see the continuous nature of each node with their respective activities; hence we plotted a histogram for each node to understand their individual characteristics and their nature of skew symmetry follows normal distribution. The histogram plots have been displayed below.

A picture containing text

Description automatically generated

After understanding each node's nature, correlation of those nodes were plotted through the corrMatrix plot. This plot helped us to identify the significance of each node to the response.

Graphical user interface, table

Description automatically generated

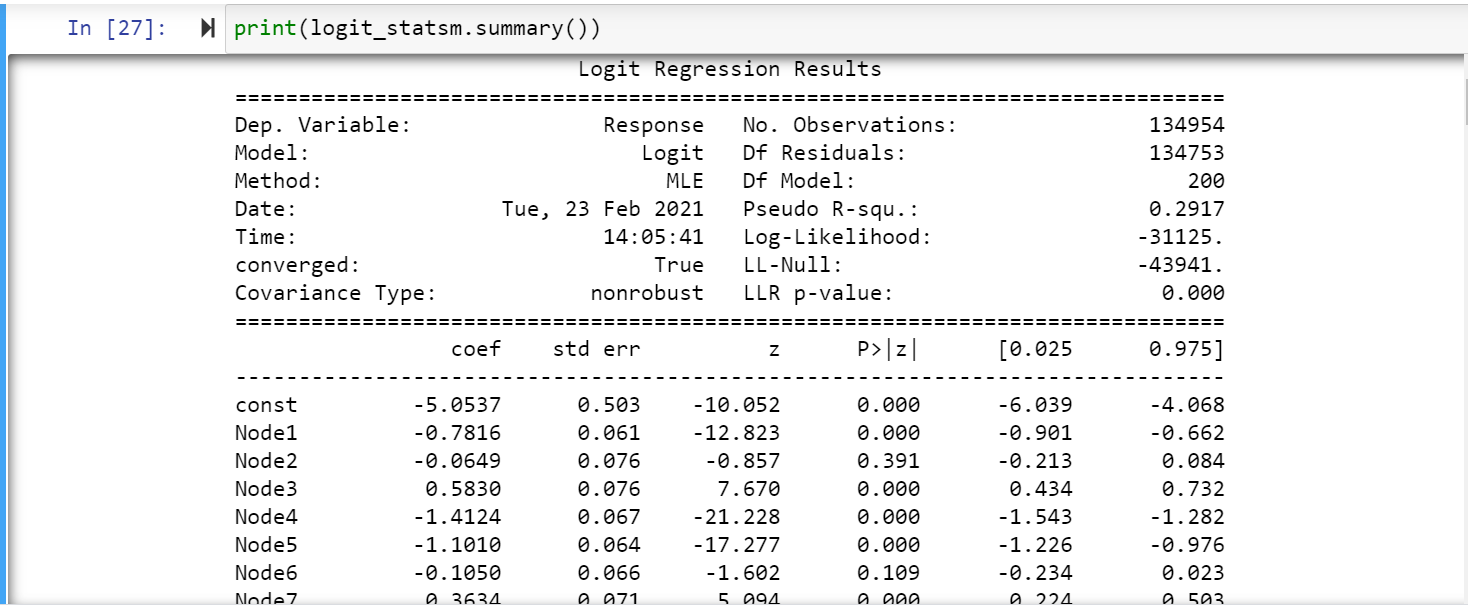
This also helped us to identify and remove highly correlated nodes to get an efficient model.

* **Data Pre-Processing and Splitting techniques**

Data must be processed before making a model, since the dataset is huge and number of missing elements are very less comparatively those are removed.The values of nodes and activities vary on its own measure,so we standardized all the values by Scaling and Transforming the training data. Followed by splitting the data, which is achieved by the Stratify command, train\_test\_split() commend and the data is prepared for the model. For each algorithm, different splitting methods were experimented and best parameters were used for efficiency.

* **Description of the Model used for Analysis and the Visual Representation of Results**

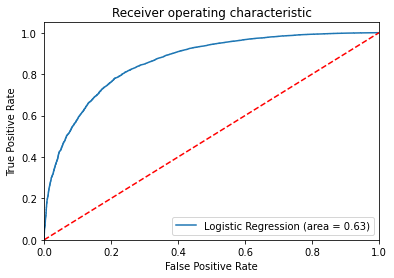
**Model 1.**

After analyzing and working on the data so far, it is easy to decide that, for classification problems with binary results, **Logistic Regression** is good. The dataset was trained for the model by removing correlated variables, standardizing, and scaling the values between 0 and 1. Hence, we started our analysis with Logistic Regression. The Regression Results have been displayed below.

**Confusion matrix, Accuracy Score, classification report** and **ROC curve** were measured appropriately and plotted in the model. Few snaps of the code for the above discussed method have been displayed.

Table

Description automatically generatedText

Description automatically generated

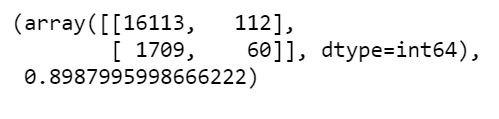
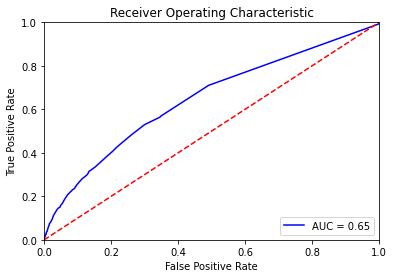
**Sensitivity : 0.986189**

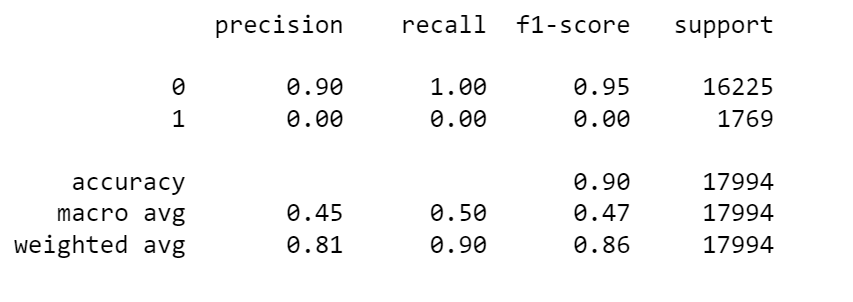
**Specificity : 0.275892**

**Concluding Thoughts:** Logistic Model serves as a good start for the classification model and provides decent performance which could be further improved.

On top of the logistic model, we applied a more predictive model.

**Model 2: Decision Tree Classifier:** We chose the decision tree classifier as it can classify without requiring much computation.The scaled data has been applied on the decision tree classifier.

**Confusion Matrix and the accuracy score** was found. **Classification report** was obtained as below.

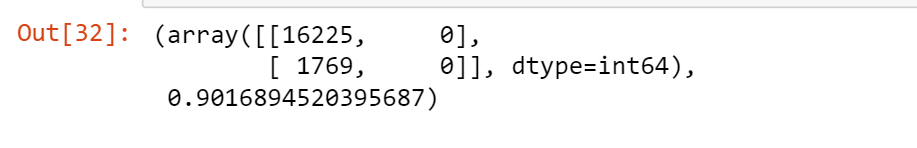
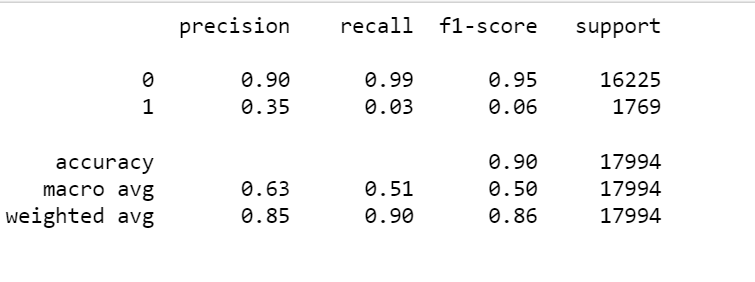
**Accuracy score of 0.898799** was obtained.

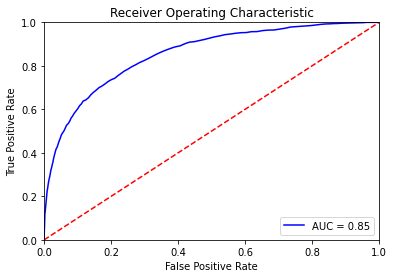
The **Sensitivity** and **Specificity** were found to be **0.904107** and **0.348837 respectively.**

**Concluding Thoughts:** Decision Tree possesses decent accuracy but the performance observed from the classification report results flaws over handling the data efficiently.

**Model 3: Random Forest Classifier:** The next model Random Forest classifier is introduced to the data. We chose the Random Forest classifier as it can handle the missing values and maintain the accuracy of a large proportion of the data.

From the Random forest classifier, we obtain a **Confusion Matrix and accuracy score.** **Classification report** was obtained as below.

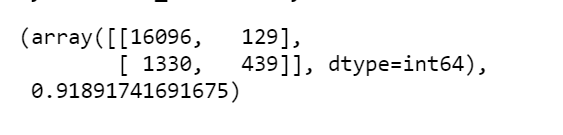
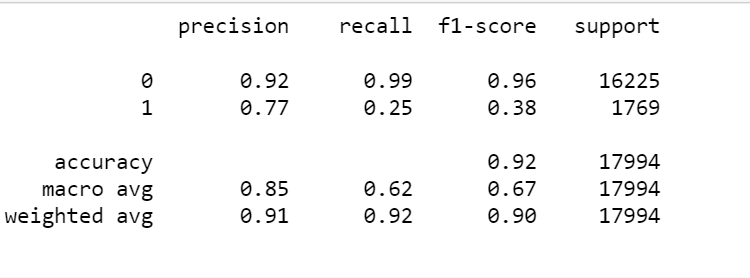
**Accuracy score of 0.901689** was obtained.



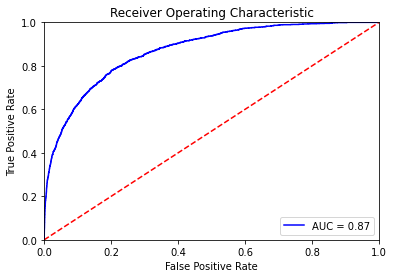
The **Sensitivity** was found to be **0.901689.**

**Random Forest Classifier:** We expected the Random forest model to perform much better than the achieved results with various parameters modifications.

**Model 4: Xtreme Gradient Boost:** The reason we chose XGBoost was because it performs well when you have unbalanced data.We proceeded with the supervised Gradient method to obtain enhanced accuracy. The **Confusion Matrix** obtained is as follows:

As expected, xgboost yields high accuracy comparably, we found a **Classification report.**

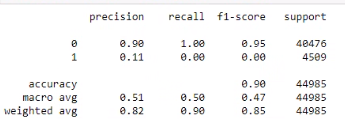
**Accuracy score of 0.9189** was obtained.



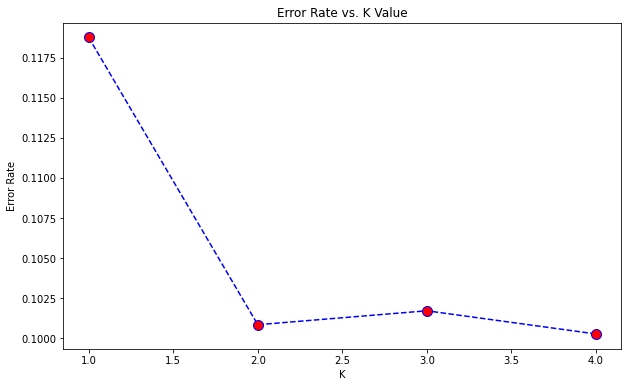
The **Sensitivity** and **Specificity** were found to be **0.923677** and **0.772887 respectively.**

**Concluding Thoughts:** XGBoost can offer better performance on binary classification problems with a severe class imbalance. This model performance is pretty robust.

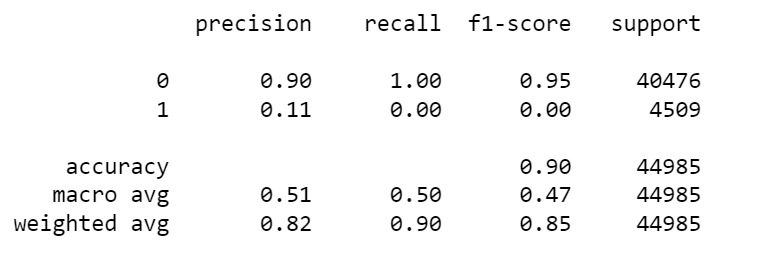
**Model 5 : K-nearest neighbors (KNN) algorithm:** Since we had properly labeled data, we decided to try KNN algorithmThis algorithm was also attempted successfully after overcoming all computational complexity.



**Accuracy of 0.898277203 At k=3.**



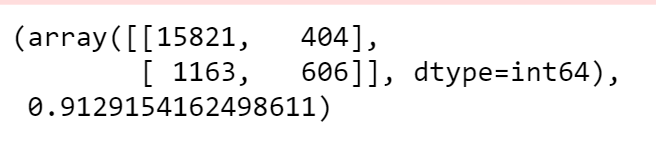
From the graph above we can see that the error rate for the k value is least for **k=2 or k=4**. The classification report for **k=2** is given below:



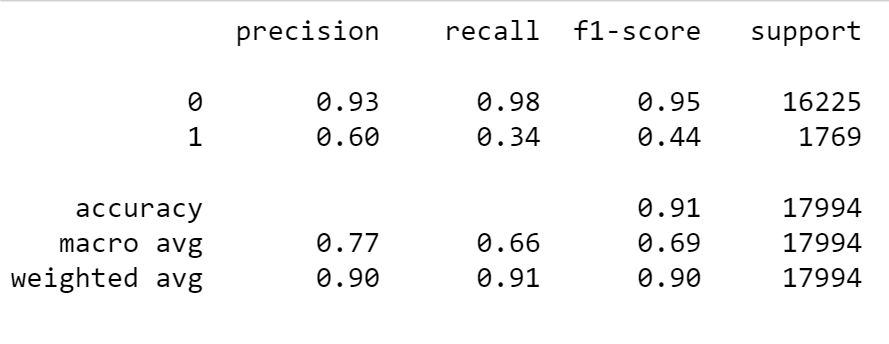
Then from the computational results we got that the error rate is lowest at k=4 which is 0.100277 at **k=4**. The **accuracy** for the model was found out to be **0.899723.**

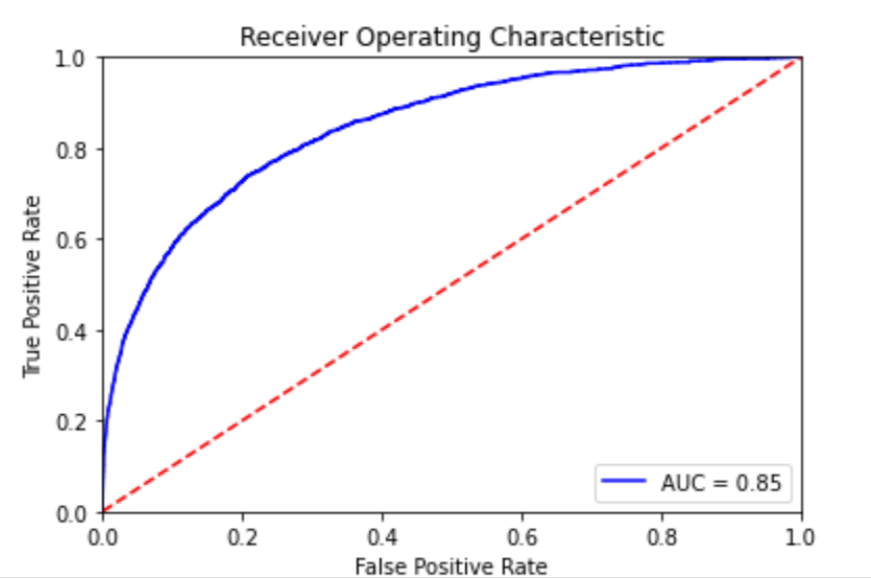
**Concluding Thoughts:** Knn algorithm is easy to implement and understand, but has a major drawback of becoming significantly slower as the size of that data in use grows.

**Model 6 : MLP Classifier:** MLPClassifier stands for Multi-layer Perceptron classifier where the name itself relates to a Neural Network. We decided to use this method as they are very flexible and can be used generally to learn mapping from inputs to outputs. This flexibility allows them to be applied to other types of data.



**Classification Report**





The **Sensitivity** and **Specificity** were found to be **0.931523** and **0.6 respectively.**

**Concluding Thoughts:** As expected MLP Classifier performs comparatively very well, the accuracy and the performance metrics results are efficient since they classify an unknown pattern with other known patterns that share the same distinguishing features.

**Testing**

A new, highly imbalanced test data set was provided. As we've shown, accuracy isn't the only right criterion to use for assessing imbalanced datasets because it can be deceiving. The performance metrics will help get a deeper understanding. Xg-boost and Neural Networks MLP Classifier gives better performance metrics comparatively than others. We decided to go with Xg-boost since it's slightly more reliable in imbalanced data to consider with the adjustable scale\_pos\_weight hyperparameter.

**5. Conclusion:**

* The achieved accuracy score and performance metrics provide evidence that the Xgboost performs robustly compared to other models.
* It proves that **XGBoost** and gradient boosting machines are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture.
* It repetitively leverages the patterns in residuals, strengthens the model with weak predictions, and make it **better**.
* From the experience of this project, it is significant to note that applying a **suitable model/algorithm saves** a lot of computational complexity and **time**.