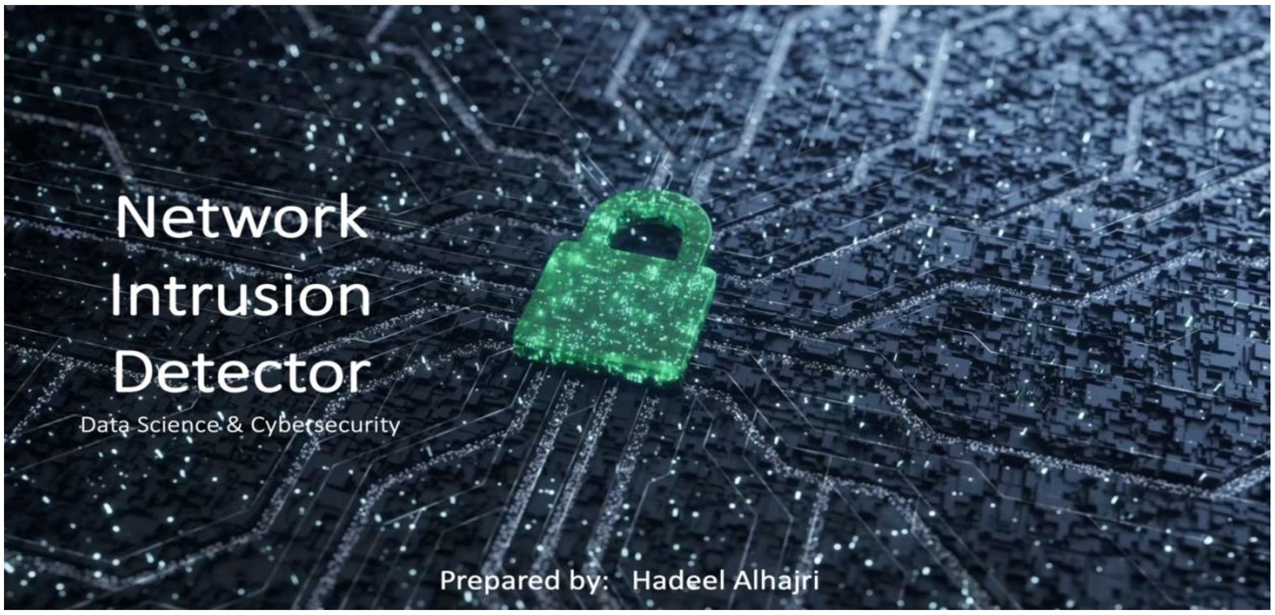


Project Definition

Project Overview

Nowadays, in the digital world, the field of cybersecurity has became more important, and the need of anomaly-detection real time systems has extremely increased. These systems can allow the ability to perform fast real-time defensive operations to protect the cyber world. This project is to build a classifier that can detect network intrusions by classifying network connections to “normal” or “attack” classes. This predictive model will be helpful to the field of network security as an automated network intrusion detector. Classifying connection’s types accurately via machine learning models would enable the Security Operations Center (SOC) officials to prevent malicious events efficiently and can allow to perform fast real-time defensive operations to protect the cyber world.

I worked on data that was provided for KDD Cup 1999[1] leveraging feature engineering along with some classification models such as Logistic Regression, Decision Tree, and Neural Network to achieve very excellent results for this binary classification problem.
Moreover, I tried to deal with the class imbalance and high dimensionality challenges, so I used Random Forest to select the important features, and I tried under-sampling method to balance the data.

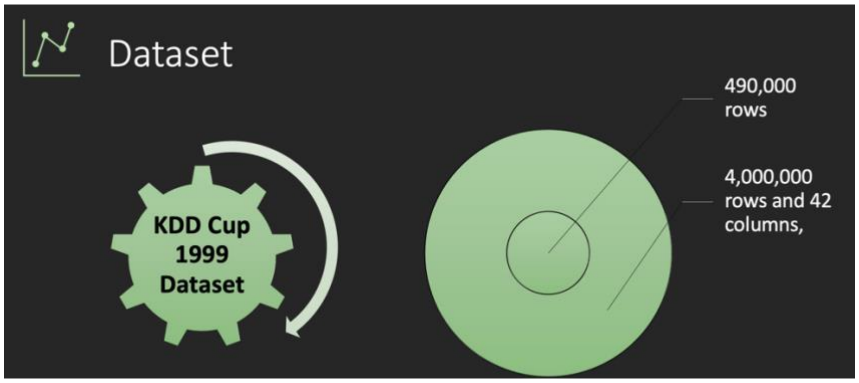
Problem Statement

This project is to build a predictive model capable of distinguishing between bad connections, called intrusions or attacks, and good normal connections. This is a binary classification model that can classify network traffic by analyzing different features of network connection and classify it as “normal” if it is legitimate or “attack” if it is malicious.

[1] https://kdd.org/kdd-cup/view/kdd-cup-1999

Data Description

The **KDD Cup 1999 Dataset** is very huge, as mentioned in the UCI website[2], the full dataset contains around 4,000,000 rows and 42 columns, that makes it difficult to use this huge dataset to build a classifier using only a personal computer's capabilities, a high-performance computer with more computational resources may be needed to process all these observations for more accurate results in shorter time.
However, a portion of 10% of this dataset was used for such project. Moreover, that portion of KDD Cup 1999 Dataset contains around **490,000 rows** and can be obtained easily by **Kaggle** website[3].



Metrics

Because of the high cost associated with false negatives (attack connections that were classified as normal), then **recall metric** is the metric I used to measure the results.

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$$

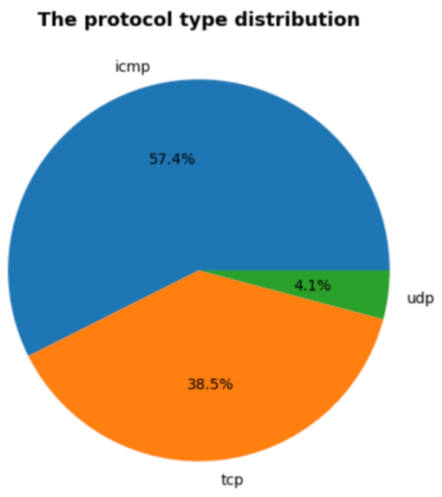
Analysis

Data Exploration and Visualization

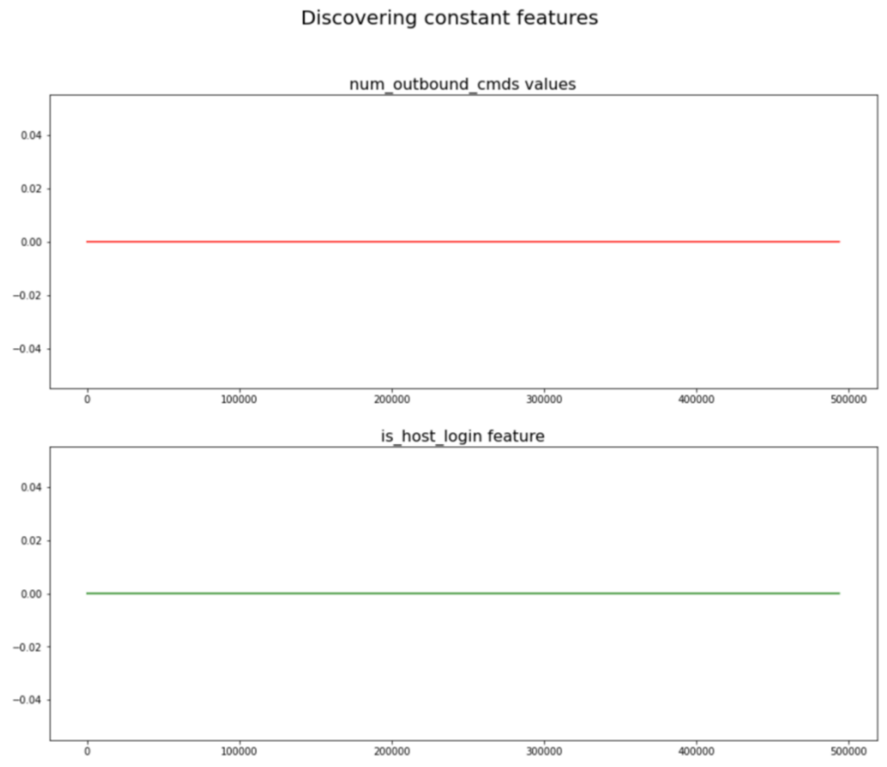
The data set contains 494020 observations and 41 features (excluding the class label column), each observation represents a connection with many features such as: duration, protocol type, service type, number of data bytes, number of wrong fragments, number of urgent packets, number of failed login attempts, successfully logged in flag, connection status, …
Most features are numerical, while 3 of which are categorical. Categorical features include protocol type, service, and connection flag. The following figure represents the protocol type distribution, as

[2] https://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data
[3] https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data?select=kddcup.data_10_percent.gz

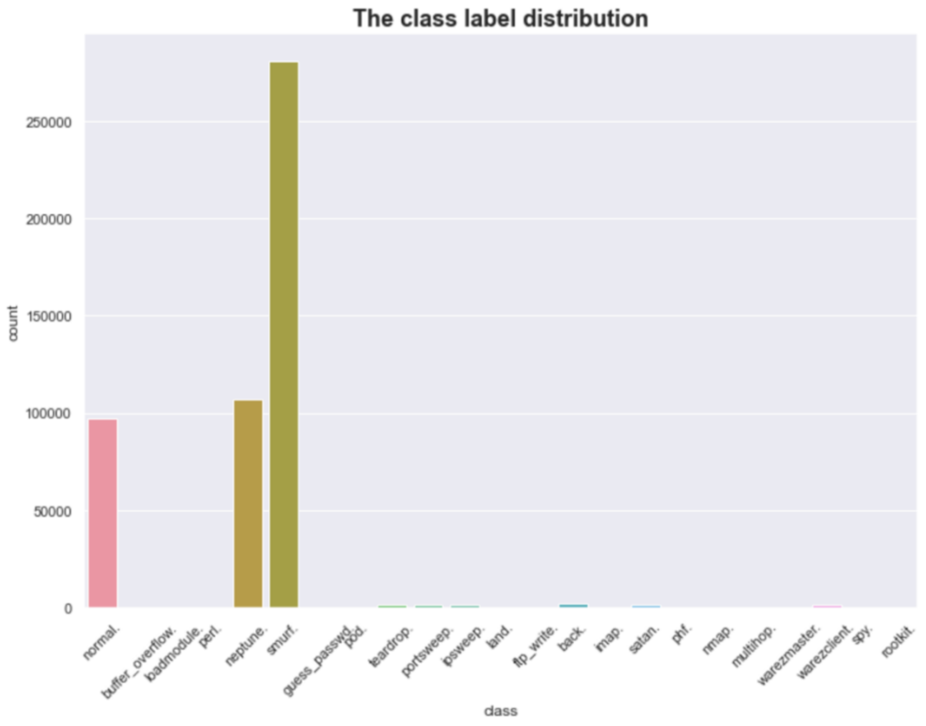
noticed, the majority of the connections used the ICMP protocol, and the minority used UDP protocol.



Moreover, after analyzing the statistics of the numerical features and finding their maximum values, I noticed that *num_outbound_cmds* and *is_host_login* features are seemed to have constant values, so I plot them to make sure.



Additionally, KDD Cup 1999 Dataset contains 23 classes, 22 of them are considered as different attack's types and 1 class represents the "normal" connection (legitimate connection). The following figure illustrates the class label distribution.



Methodology

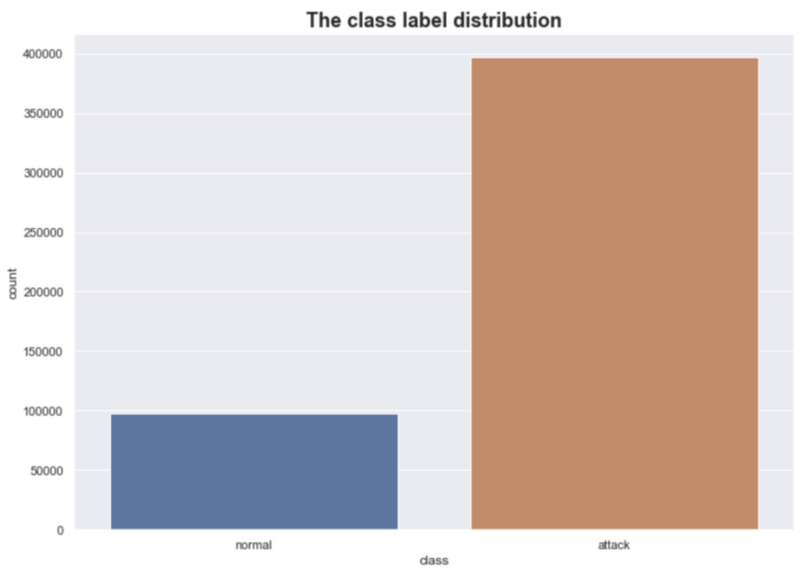
Data Preprocessing

- **Removing Constant Features**

As mentioned above in the Data Exploration and Visualization section, num_outbound_cmds and is_host_login features have constant zero value (no variance) which indicates that they do not have any importance in the data set, so I simply dropped them.

- **Formatting The Class Label Column**

Because I chose to consider the problem as a binary classification problem, the class labels need to be restricted to be only 2 classes. One suggestion is to make the classifier more general by giving it the ability to only detect the presence of attacks without deeply specifying their types. So, the newly formatted class label is "normal" if the connection is legitimate and "attack" if it is malicious. The following figure illustrates the class label distribution after re-format the class label column by replacing and attack type with "attack" label.



- **Feature Engineering**

1. <u>Categorical Features One Hot Encoding:</u>
   As I am planning to build multiple classification models for this task, some of the models cannot work with categorical features (such as decision trees), so I need to make some features encoding. First, as mentioned above, the dataset has 3 categorical features include protocol type, service, and connection flag.
   Depending on the number of possible values for each feature, the encoding method is determined, here is a good reference for encoding methods[4].
   The protocol type has only 3 possible values, while the connection flag has 11 possible values. It seems that the one-hot encoding can be used for these two features since the number of their possible categories is relatively small.

2. "<u>Service" Feature Label Encoding</u>
   After exploring the "service" feature, I found that it has huge number of categories (66 possible values) which will affect the dimensionality negatively if one-hot encoding is used! So, I decided to use label encoding for the "service" feature.

3. <u>Target Column "Class" Encoding</u>
   The same label encoding technique is used on the target column ==> 'attack': 0, 'normal': 1

Implementation

- **Splitting The Data to Train and Test Data Sets**

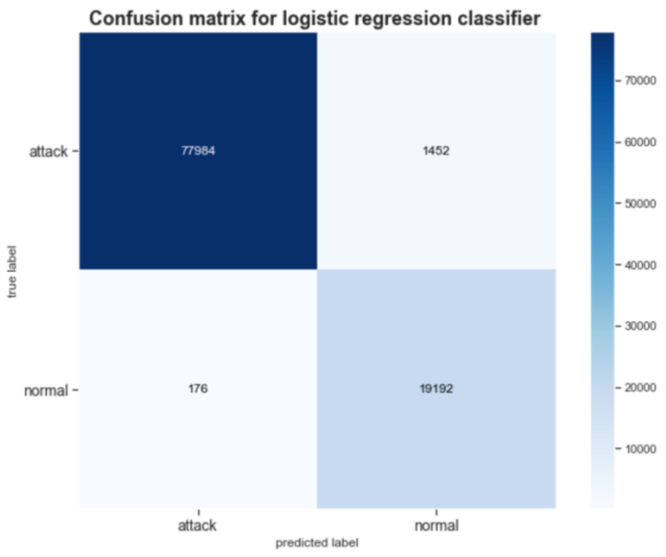
After the preprocessing steps, the data is splitted into two sets: 80% for train set, and 20% for test set. The train set is used to train the models while the test set is used to evaluate them.

[4] https://www.statology.org/label-encoding-vs-one-hot-encoding/

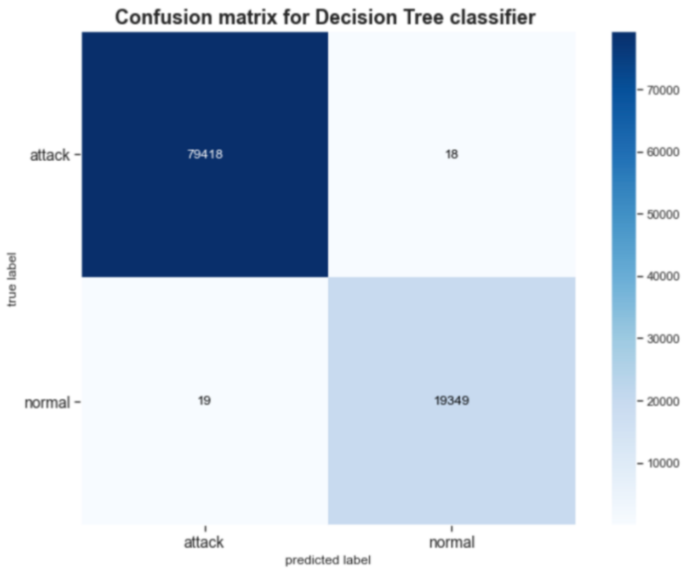
- **Building Classification Models**

Sklearn library features various classification algorithms including Logistic Regression and Decision Tree. In addition, Keras library provides good implementations to facilitate building neural networks. Using the previously mentioned libraries, both Logistic Regression, Decision Tree, and Neural Network classifiers were tried for this binary classification task.

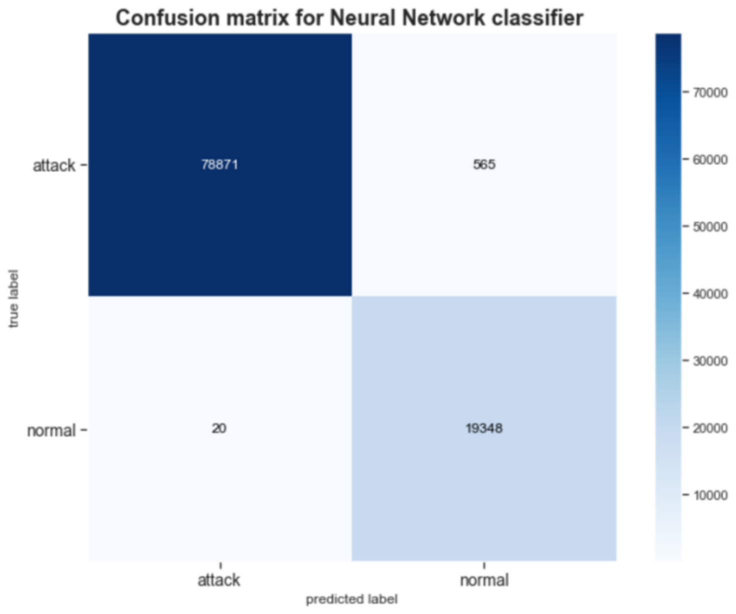
1. Logistic Regression
   The following figure shows the confusion matrix after passing the test set to a Logistic Regression model that is trained on the train set.



2. Decision Tree
   The following figure shows the confusion matrix after passing the test set to a Decision Tree model that is trained on the train set.



3. Neural Network
   I implemented build_neural_network() method that gets benefit of Keras layers to build and return a neural network as a collection of sequentially stacked layers (Dense, and Dropout), I chose the activation function for the last fully connected layer to be *sigmoid*, since I preferred to deal with this network intrusion detection problem as a binary classification task. Also, I used binary_crossentropy as a loss function and Adam as an optimizer.
   Because Neural Networks' training loops for many epochs, it needs a validation set (subset of the training set that is not seen by the model), to validate model's performance after each epoch. I chose subset 25% of the previous training set as a validation set, then trained the model for 20 epochs.
   The following figure shows the confusion matrix after passing the test set to a Neural Network model that is trained on the train set.

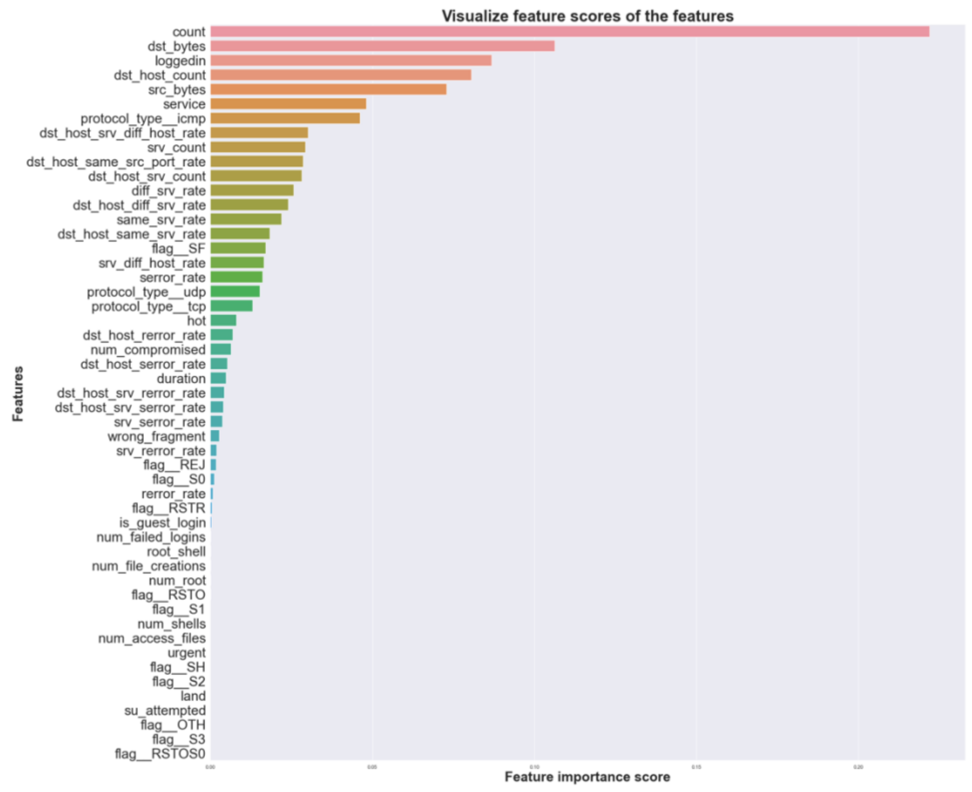


Refinement

- **Dimensionality Reduction**

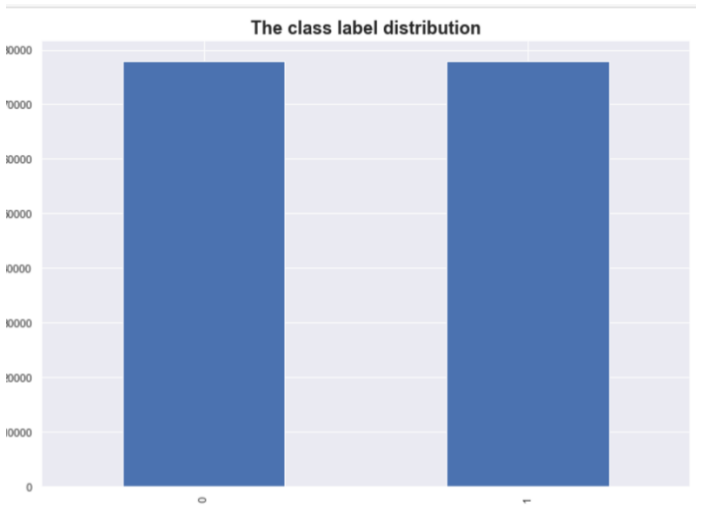
Because the problem is to detect network intrusion, I need to optimize our classifier by making it faster, otherwise, it will not be helpful in real-time usage for the SOC officials. Moreover, if I came up with a wonderful super accurate classifier that takes some time to predict the presence or absence of network attacks, then these accurate results will not be taken in the consideration as if it is fast!
That is why it is worthy to at least try my best to reduce the high dimensionality that the data suffers from to reduce the processing time while making sure to affect the accuracy badly.
So, I thought to try to reduce the high dimensionality in the dataset by determining features importance and removing less important features. Random forest feature importance ranking was used to guide the selection and order of variables to be included as the model underwent refinement to solve the problem of high dimensionality.

The following figure shows the features importance score associated with each feature; top 10 important features were used to prepare the REDUCED dataset; this reduced the features space from 51 to 10.



- **Class Imbalance**

As previously shown in the class distribution figure that is represented in the data preprocessing section, it is clear that the dataset suffers from the class imbalance problem, since the "attack" class samples are much more than the "normal" class. Therefore, I tried the solution of **undersampling** the majority class "attack" and see if it will improve the results.
I know that it is not a good way to lose records! since more training data leads to more accurate results, however, I am just trying this available option because the data is huge which allows for under-sampling. Below is the new class distribution after under-sampling the majority class.



- **Cross Validation**

Now, I need to find out how well the model performs. Is it accurate enough to be used? There are several evaluation methods to determine this. One such method is a K-fold cross validation. 10-fold cross-validation technique is used as a last step in this project to find out how well the model can predict the outcome of unseen data. This technique shuffles the dataset and split into 10 number of subsamples, then will iterating 10 times choosing one split as a test set and the other 9 as train set each time, so 10 scores will be obtained. The average of all these individual evaluation scores is the final score and I will discuss that in the Results section.

Results

Model Evaluation and Validation

- **Hold-out**

    - **Original Data**

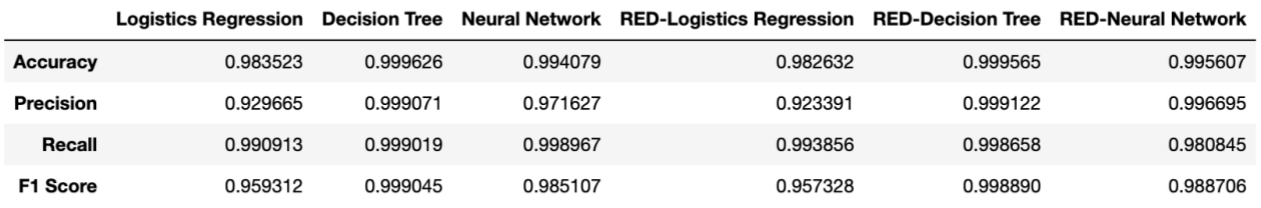
The entire dataset of 494020 records was split into 80/20 train vs. holdout, and all scores reported below were calculated on the testing portion. As mentioned in Metric section, I will consider recall score as a trustworthy metric to evaluate the models for this problem.

	Logistics Regression	Decision Tree	Neural Network
Accuracy	0.983523	0.999626	0.994079
Precision	0.929665	0.999071	0.971627
Recall	0.990913	0.999019	0.998967
F1 Score	0.959312	0.999045	0.985107

As noticed the Decision Tree achieved the highest scores according to all metrics; accuracy, precision, recall, and f1_score metrics.

▪ Reduced-Dimensions Data

The columns that start with 'RED-'mean results on data after dimensionality reduction, in other words, the data with only 10 top important features as explained in the Refinement section.

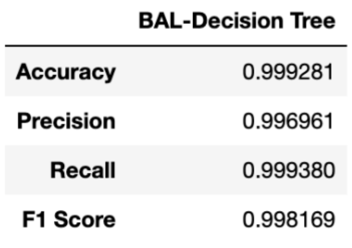


	Logistics Regression	Decision Tree	Neural Network	RED-Logistics Regression	RED-Decision Tree	RED-Neural Network
Accuracy	0.983523	0.999626	0.994079	0.982632	0.999565	0.995607
Precision	0.929665	0.999071	0.971627	0.923391	0.999122	0.996695
Recall	0.990913	0.999019	0.998967	0.993856	0.998658	0.980845
F1 Score	0.959312	0.999045	0.985107	0.957328	0.998890	0.988706

As noticed that Decision Tree recall scores were degraded slightly. On the other hand, the Logistics Regression and Neural Network models' recall score increased after dimension reduction. However, according to recall score the best model is still the Decision Tree without dimensionality reduction.

▪ Under-sampled Balanced Data

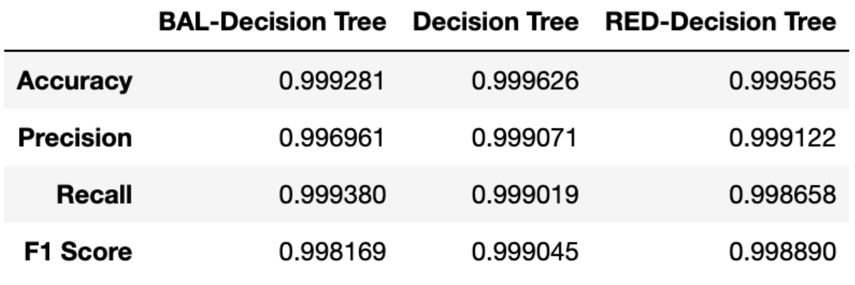
Using under-sampling technique explained previously, the total number of training recording decreased to 155818 samples. The columns that start with 'BAL-' mean results on the balanced data, in other words, the data after removing some "attack" samples as explained in the Refinement section. In addition, after I realized that Decision Tree is the strongest performing model, I decided to exclude other models and continue to determine the best version of Decision Tree.



	BAL-Decision Tree
Accuracy	0.999281
Precision	0.996961
Recall	0.999380
F1 Score	0.998169

▪ Choosing Best Decision Tree Version

The Decision Tree is the model with strongest performance, comparing its results on the original data, reduced-dimensions data, and under-sampled balanced data the results are shown below.



	BAL-Decision Tree	Decision Tree	RED-Decision Tree
Accuracy	0.999281	0.999626	0.999565
Precision	0.996961	0.999071	0.999122
Recall	0.999380	0.999019	0.998658
F1 Score	0.998169	0.999045	0.998890

Although undersampling method may not be the best solution to solve the class imbalance issue, it has proved its ability to enhance the model performance in terms of recall score. Therefore, I decided to choose the winner network traffic classifier to be **Decision Tree model that trained on undersampled training data** and without removing features (without dimensionality reduction).

Justification

- **10-fold cross-validation**

The final Recall score result for Decision Tree network traffic classifier trained on balanced data using 10-fold cross-validation is **0.99786.**

- **Results Discussion**

Decision Tree has proved its ability to deal efficiently and accurately with the high dimensional network traffic data. One reason of Decision Tree that outperforms Logistic Regression is that it bisects the space into smaller and smaller regions, whereas Logistic Regression fits a single line to divide the space exactly into two. Moreover, Neural Network achieved excellent recall score on the original data, and it was very near to the Decision Tree performance. One possible justification is that I trained the neural network for only 20 epoch which may not be enough to understand the task. Additionally, as known that Decision Tree performs some sort of ordering the features in a hierarchy manner during building the tree, this what made Decision Tree super model even when using many features.

Conclusion

Reflection

The project was to build a predictive ML model to distinguish between intrusion and normal network connections as a network traffic classifier. The project includes applying feature engineering techniques and dealing with data challenges such as class imbalance and high dimensionality. Dimensionality reduction is the most interesting part I found in this project, it is a new concept I have learned.

Improvement

Although I got very excellent classification results, here are some ideas I think they can enhance the performance:

1. Using the whole KDD Cup 1999 Dataset with around 4 M records.
2. Trying other classification models such as K-Neighbors.
3. Increase the number of epochs when training the Neural Network.
4. Considering more sophisticated class imbalance solutions such as assigning weight to each class which weakens the majority class high affect.