**Analysis Strategy:**

After studying the distribution of data, we identified that there were different subcategories of DDoS attack based on the layer of the network connection they attempt to attack. The result column had 60% of the normal data. and rest 40% attack types were unevenly distributed. So, clearly the major challenge was to handle the class imbalance problem. Our approach was to implement different sampling techniques to get the classes balanced. Since, for class imbalance problems, accuracy is not an appropriate metric for model evaluation because the accuracy score would be high and heavily biased towards the majority classes (normal class for KDDCup dataset). Hence our main goal was to identify the minority class (attack sub-classes). So, we focused on precision-recall and FPR(fallout rate) for the evaluation of the machine learning models. In other words, our aim was to minimize a bad connection that gets classified as normal.

**Analysis of Code**

**1. Data Exploration**: Since our objective was to cover the majority of the attack types we combined the test and train data from the KDDCup dataset.

1. Identified the dataset for the null values. We found that there were no null values.
2. Checked for the duplicates in the data frame, around 70% of the data was duplicate so we dropped this.
3. Analyzed the attributes of the dataset and worked upon the numerical and categorical features individually.

**2. Feature Selection**

1. After plotting correlation matrix, there were total 9 pairs of highly correlated features, we selected one from each pair. After which there were total 32 numerical attributes.

**3. Data Preprocessing**

1. Numerical attributes: total count=32

We standardized the numerical attributes which had the range greater than 1.

2. Categorical attributes: total count=3 (service, flag, protocol type)

For the columns service and flag had high number of subcategories. On converting numerical value using one hot encoding result would have resulted in the addition of a column per subcategory. In this case it would result in adding 66 + 11 + 3 - 3 = 77 columns. This would have added to the complexity of the model. Hence, we used baseN encoding which highly reduces the dimensionality as the value of N increases.

3. We used SMOTE (Synthetic Minority Oversampling Technique) for balancing the classes.

**4. Model Selection**

We used the following algorithms for the training the model and hyper tuned them.

1. Decision tree
2. Naïve Bayes
3. Random forest
4. Logistic regression
5. XGBoost

**5. Model Comparison**

Hypertuned decision tree performed the best.