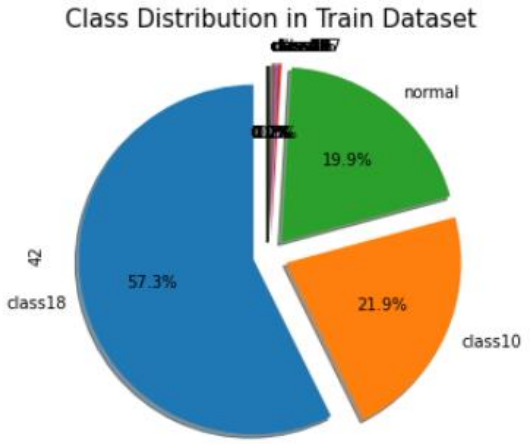
Network Anomaly Detection – A MLPO Challenge by Vbetter

Network Anomaly Detection Challenge

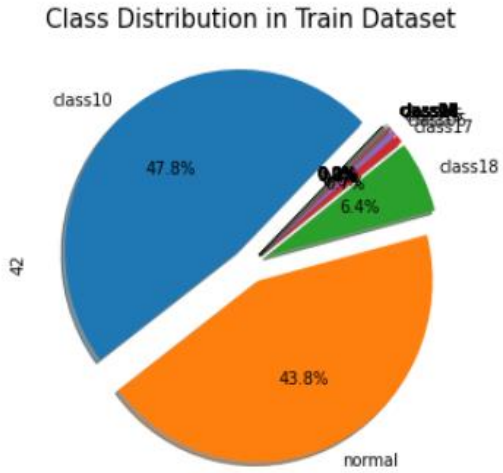


MLPO Challenge: There's been different anomalies in network which are detected and labelled with different class names. With the provided data, we need to train a model in a way that could effectively detect these anomalies on rest of the unseen data, along with it should be able to classify the unseen anomalies as abnormal rather than normal.

Training Set ~ 4.9M rows of labeled tabular data, representing normal network traffic and various types of network anomalies. There are no column headers. The label is the last column.

Rows with a "normal" label indicate this represents normal network activity. Labels of the form "classNN" indicate different types of anomalous network activity. You will notice that some anomaly types are relatively common while others are extremely rare. For this reason, this challenge is a multi-class classification problem with extremely imbalanced classes.

Evaluation Set ~311k rows of additional, unlabeled network traffic data. There are no column headers.




```
Number of classes after removing duplicate rows from train dataset: 23
```

Modelling Methodology

Initials and Data Cleaning

- Import Python Libraries
- Google Collab Engine and GPU for faster processing
- Reading Train and Eval Dataset
- Observing the columns and rows, removed duplicated rows
- Removed features with all zero values etc
- Based on observation, cleaned the features for less complexity
- 1 Min

EDA

- Combined feature 1 for lesser categories
- Plotting Boxplots and bars for observing data trends in each feature
- Derived a new feature Groups
- 5 Mins

Data Pre-processing

- Encoding for categorical features
- Created Test and Validation sets, scaled the data using Quantile scaler
- 1 Min

Feature Selection

- Stats Tests Like Chi, Anova etc
- PCA
- 3 Min

Sampling

- Random Under sampling
- SMOTE for oversample
- 2 Mins

Training Models

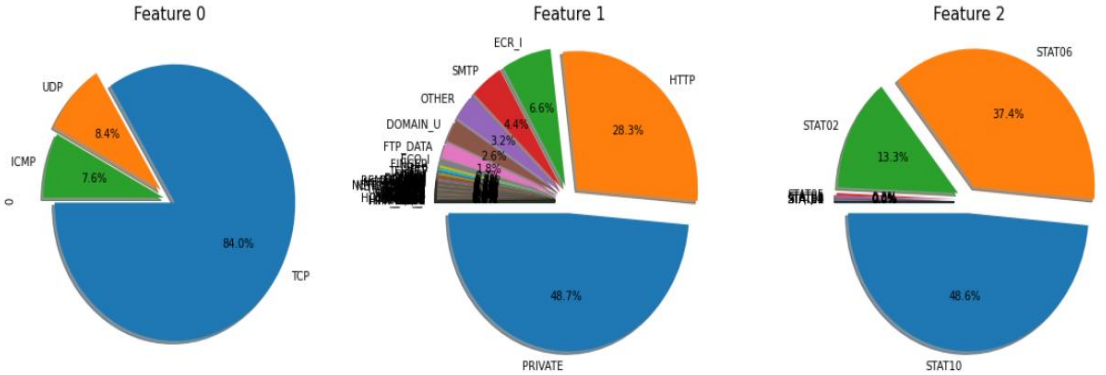
- KNN Model with Stratified K fold
- Random Forest with Stratified K fold
- LGBM
- Ensemble of All
- 15 Mins

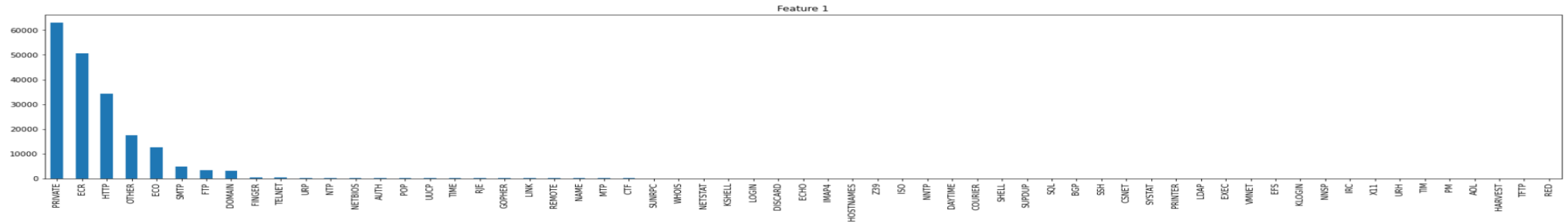
Prediction on Eval

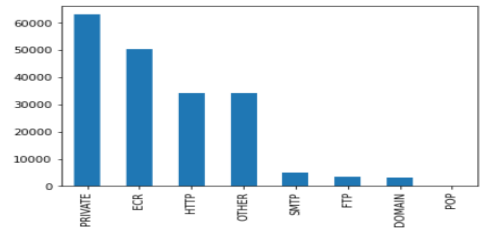
- Predicted the eval data using ensemble
- 5 Mins

Univariate Analysis

- For feature 2 we have 10 type of categories and for 1, only 3 types of categories. That's okay to keep
- For Feature 1, where we had 70 categories, we cleaned the data and grouped the less common application protocols to OTHERS category, so that having a smaller number of categories will make our model little simpler.

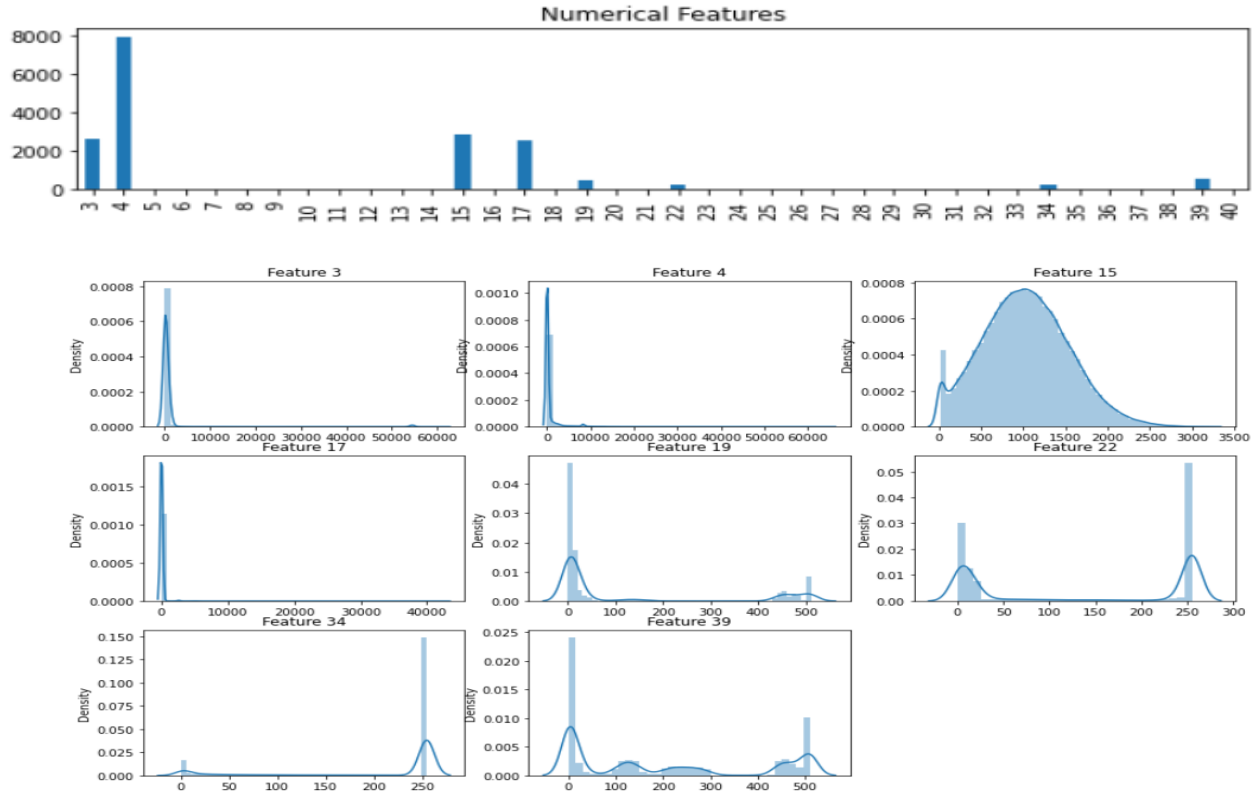






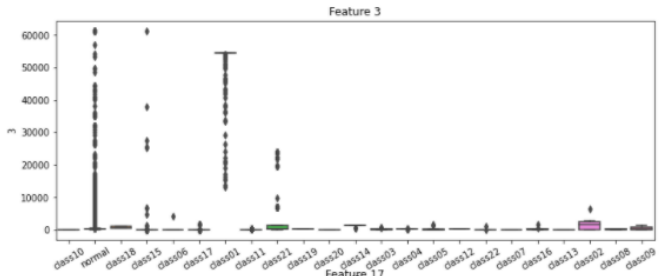
Univariate Analysis

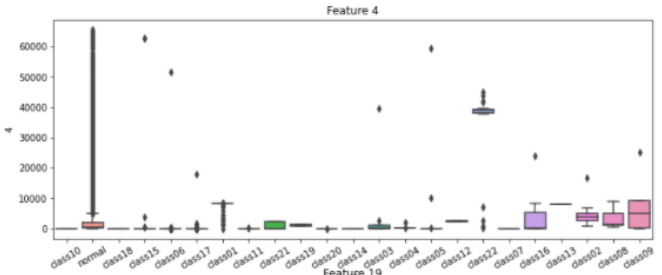
- For numerical features, there are very few columns with high variance, otherwise most of them are having less than 2 unique values as shown in left figure.
- Below figure is for deeper look into variance of high number of unique values features.

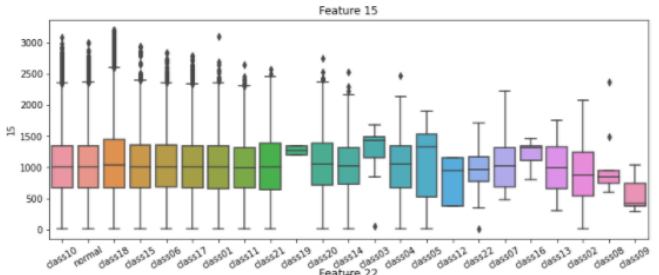


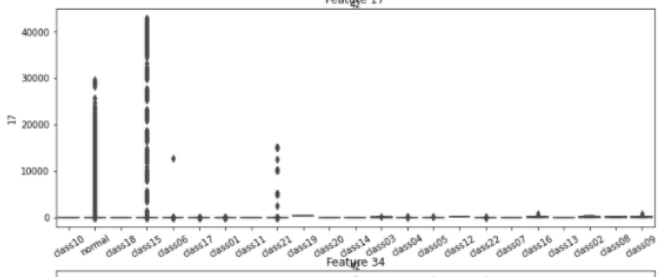
Bivariate Analysis

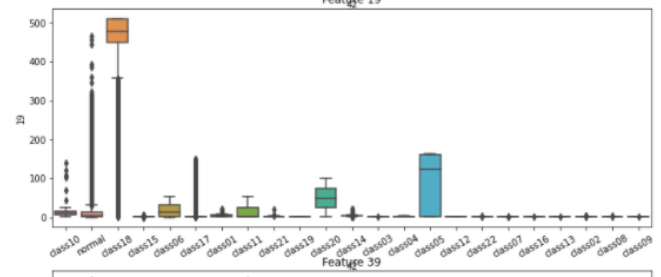
Distribution of high variance features per class to see how they behave in each type of anomalies, clearly there are lot of outliers but also these will help us to distinguish between each type of class.

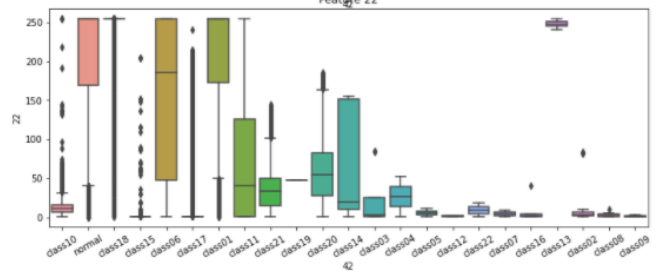


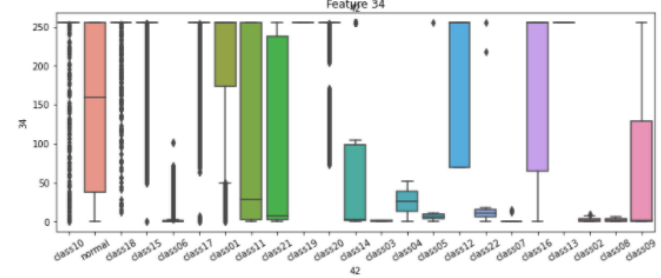


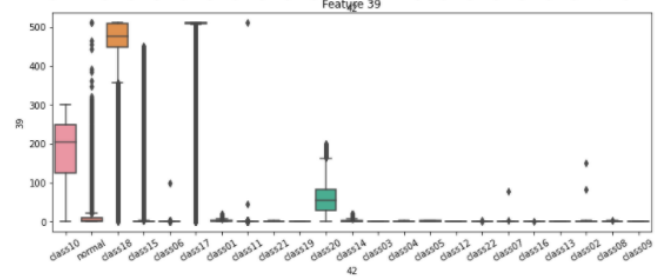








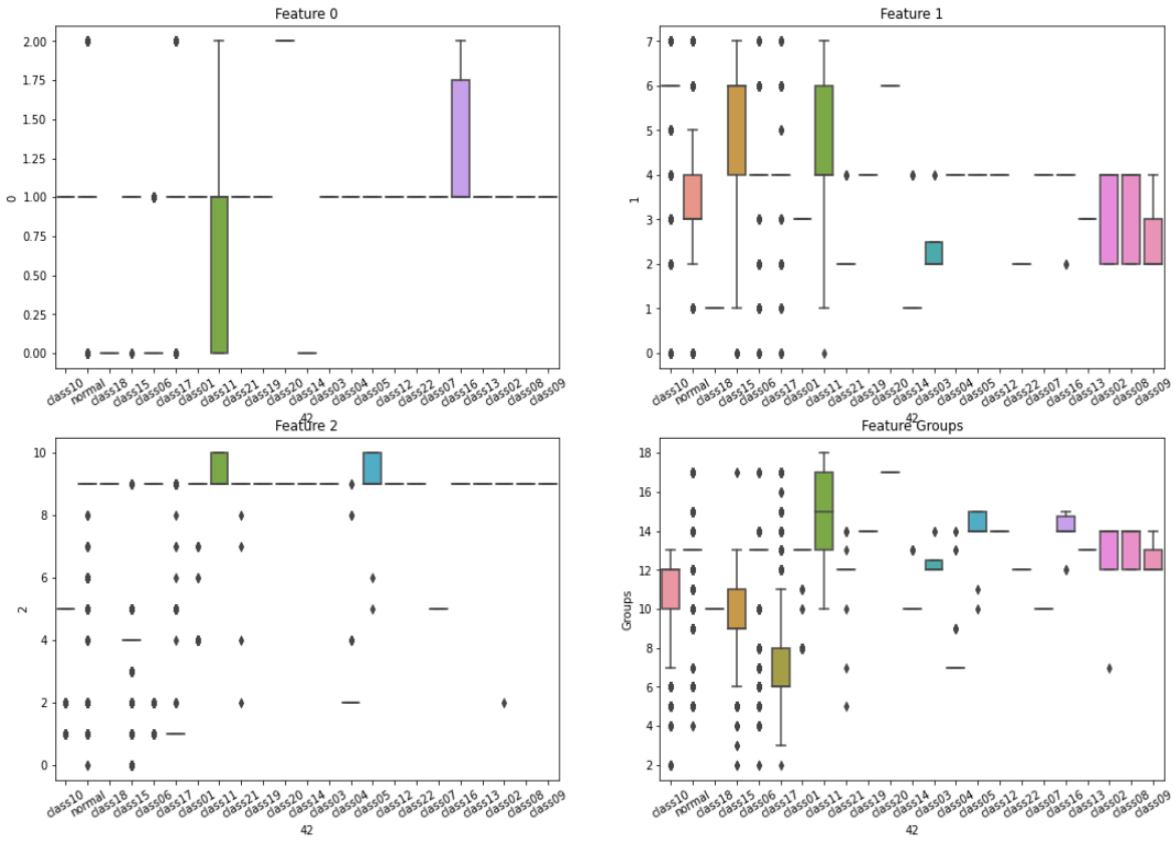




Bivariate Analysis

- Distribution of categorical features per class to see how they behave in each type of anomalies, clearly there are lot of outliers but also these will help us to distinguish between each type of class. Here we have converted the categories to numeric labels for plotting. Due to class imbalance, features have high quartile ranges for majority classes than minority classes.

- We have also derived a new feature, **Groups**, combining all categorical features based on the logic that network is made of layers and each combination of layers will show different behaviors. This will also help us in modelling.



Pre-processing & Feature Selection / Engineering

- **Data Pre-processing:**

Data pre-processing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In simple words, data pre-processing in Machine Learning is a data mining technique that transforms raw data into an understandable and readable format.

- **Data Sourcing and Cleaning** - We started with importing the necessary libraries and dataset, then we proceed further with data cleaning. By dropping duplicates, our dataset became comparatively lighter. We looked at the missing values, fortunately there were none. We did an analysis on the target variable and got to know it is highly imbalanced. We have treated the data using SMOTE after scaling the features
- **Encoding the categorical data** - Better encoding leads to a better model and most of the algorithms cannot handle the categorical variables unless they are converted into a numerical value. We have used Label Encoder for Target variable and One-hot encoding for rest of the categorical variables
- **Splitting the dataset-** Once the encoding is done, we can now move further to split our training data into training and validation step. We have kept 10% of the training data for validation and marked the Stratified as 'True', to make sure we have occurrence of minority classes in both train and validation dataset.
- **Feature Scaling-** Scaling refers to putting the values in the same range or same scale so that no variable is dominated by the other. Most of the times, dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Euclidean distance between two data points in their computations, this is a problem. We have scaled data using Quantile transforms, it is a technique for transforming numerical input or output variables to have a Gaussian or uniform probability distribution

- **Feature Selection:**

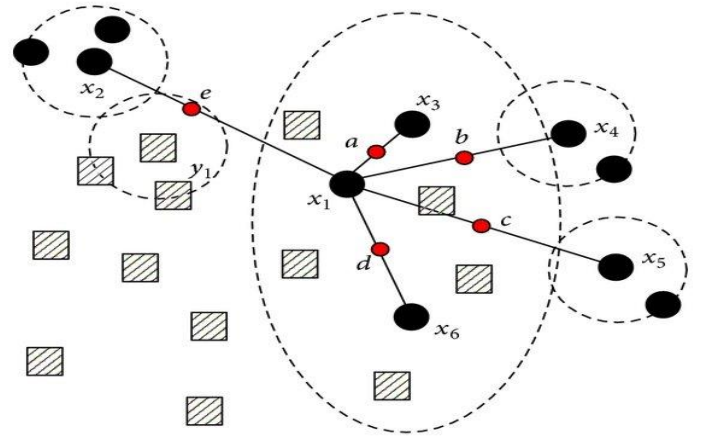
Feature selection is done twice, first by using the combination of Chi-Square, Anova(Analysis of Variance) and Mutual Information Statistics on numerical data, and then combine with our categorical data. And the second time by using Principal Components Analysis(PCA). Let's look at them in detail:

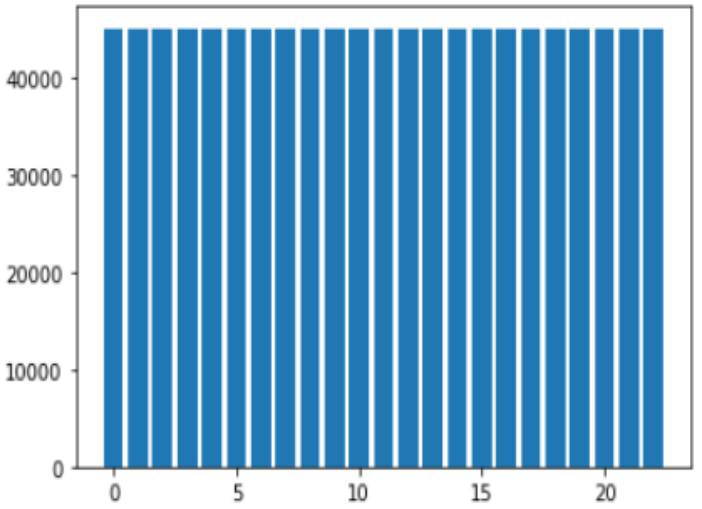
- ***Chi-square Statistics*** – A chi-square ($\chi 2$) statistic is a measure of the difference between the observed and expected frequencies of the outcomes of a set of events or variables. $\chi 2$ depends on the size of the difference between actual and observed values, the degrees of freedom, and the samples size.
- ***Anova Statistics*** – Analysis of variance, or ANOVA, is a statistical method that separates observed variance data into different components to use for additional tests. A one-way ANOVA is used for three or more groups of data, to gain information about the relationship between the dependent and independent variables.
- ***Mutual Information Statistics*** – The mutual information (MI) of two random variables is a measure of the mutual dependence between the two variables. More specifically, it quantifies the "amount of information" obtained about one random variable through observing the other random variable.
- ***PCA*** – Principal component analysis (PCA) is one of the most used dimensionality reduction techniques in the industry. By converting large data sets into smaller ones containing fewer variables, it helps in improving model performance, visualizing complex data sets, and in many more areas.

.

Sampling - SMOTE

- The challenge of working with imbalanced datasets is that most machine learning techniques will ignore minority, and in turn have poor performance on, the minority class, although typically it is performance on the minority class that is most important.
- One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the **Synthetic Minority Oversampling Technique or SMOTE** for short.
- It first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.
- Hyperparameter:
  - K =1
  - Strategy = 45000 samples for each minority anomalies





Training Methodology

KNN

K-Nearest Neighbour is based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories.

Hyperparameter:

K=21 along with PCA for features dimension reduction

Training Time: 5 Mins

Balanced Accuracy : 68% on Vbetter and 78% on Validation

F1 Score : 70% on Validation

Random Forest

Random Forests has been used for training the model, which is a collection of decision trees. The great thing about random forests is that - they almost always outperform a decision tree in terms of accuracy

Hyperparameters:

(n_estimators= 70,

min_samples_split= 5,

min_samples_leaf=1,

max_features='auto',

max_depth= 90,

class_weight='balanced',

bootstrap= True)

Training Time: 7 Mins

Balanced Accuracy : 64% on Vbetter and 88% on Validation

F1 Score : 89% on Validation

LGBM

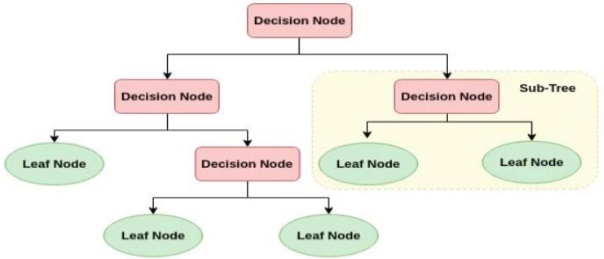
LightGBM, short for Light Gradient Boosting Machine, is a free and open source distributed gradient boosting framework for machine learning originally developed by Microsoft. It is based on decision tree algorithms and used for ranking, classification and other machine learning tasks.

Hyperparameters:
params['learning_rate']=0.1,params['boosting_type']='gbdt' #GradientBoostingDecisionTree,params['objective']='multiclass' ,params['metric']='multi_logloss' ,params['num_class']=23 ,

Training Time: 1 Min

Balanced Accuracy: 85% on validation, 73% on Vbetter

F1 score : 85% on validation



Ensemble

Used custom Voting on various combinations of trained models to achieve accuracy

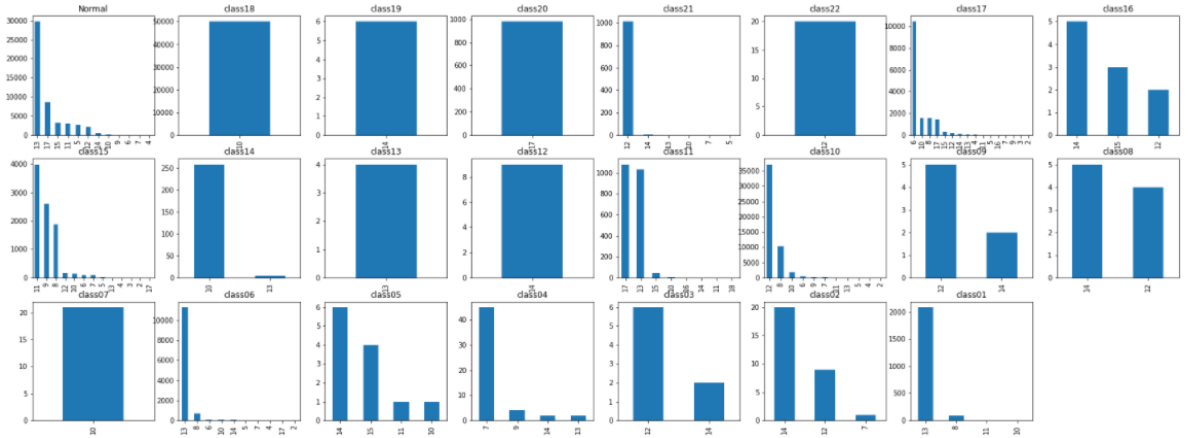
Voting is biased for anomalies and will not count it as normal if all the models has classified it as normal

Balanced Accuracy : 75% on Vbetter and 90% on Validation

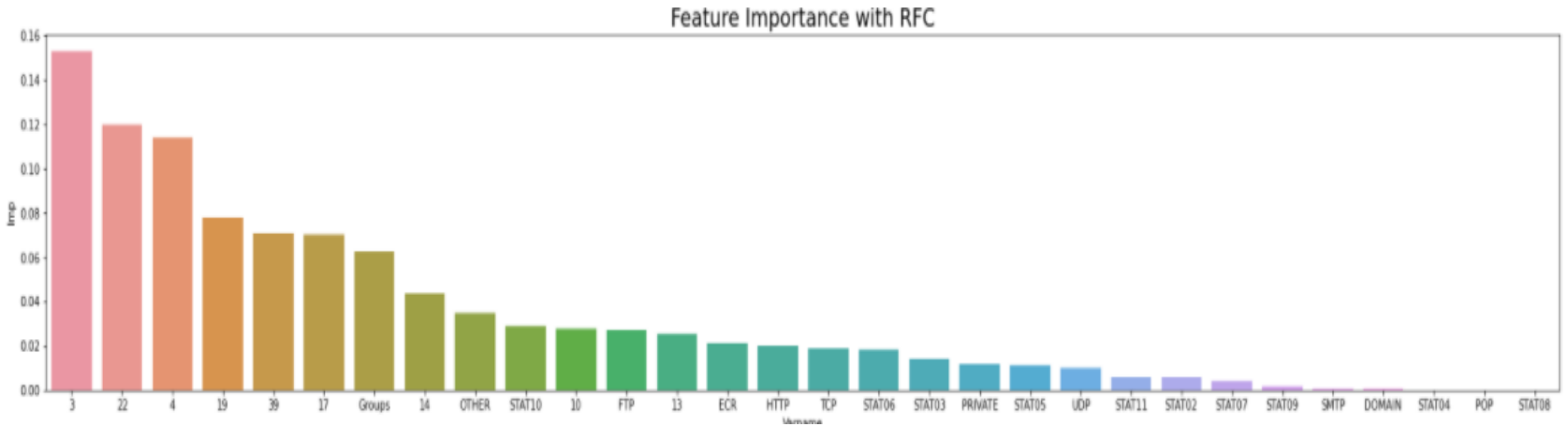
F1 Score : 88% on Validation

Notable Aspects

- **Derived feature Groups** had different variance in each anomaly as shown in left figure, along with during modelling it had really high feature importance.
- Below are **most important features** are after fitting our train dataset for different statistical tests, using these improved my F1 score on validation dataset by 10%.
- Ensemble improved my score on Vbetter by 5%, also on validation data.
- Important features using tree models, might be useful for business while collecting new data for anomalies.



Most Important Features
3
4
39
10
13
14
17
19
22



References

- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://xgboost.readthedocs.io/en/latest/get_started.html
- https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- https://scikit-learn.org/stable/modules/feature_selection.html
- https://machinelearningmastery.com/
- https://towardsdatascience.com/
- https://www.kaggle.com/notebooks
- https://imbalanced-learn.org/stable/