Crowd for Good – Cancer ML Benchmarking Challenge I

Introduction

This document contains the details of the approach followed to solve the “Crowd for Good – ML Benchmarking challenge. This is basically a binary classification problem. Different classification algorithms and their ensembles were considered and finally the XGBoost algorithm was used to obtain the final predictions of test data set.

The following are some details of the dataset:

- The training set has 360 rows
- The testing set has 110 rows.
- The target labels are binary (0 or 1 - indicating False and True)
- There are 25 features
- Column descriptions are intentionally not provided

The document contains detailed explanation of the following:

1. Reports/details of the analysis of the dataset.
2. Details of the approach followed to solve this problem
3. Reasons for going with the approach.
4. Other potential solutions considered.
5. Additional remarks or findings.

Crowd for Good – Cancer ML Benchmarking Challenge I

Analysis of the dataset

The first observation that we find while initially going through the data set is none of the features are categorical data. All the features that will be used for the prediction are numerical. The second obvious observation is that the total number of training examples given for training the models is just 360. This implies that it would be better off to go with classical machine learning algorithms like logistic regression, Adaboost, XGboost, random forest etc. than using state of the art deep learning techniques which might result in overfitting the data. Thus, a decision to start with testing the performance of these classical algorithms was made instead of straight away starting with neural network-based solutions. In the following subsections we will dwell deeper into the other data analysis done to get a better understanding of the data and into the inferences gained.

Testing class imbalance

One of the common issues that we might face in a binary classification problem is the class imbalance of the output. That is the number of training examples from one of the target classes would be very less than the other. This would require us to implement workarounds to deal with this problem as class imbalance can lethally influence the model. The number of examples from each class (0 and 1) was counted and it was found that there is no issue with class imbalance in this dataset.

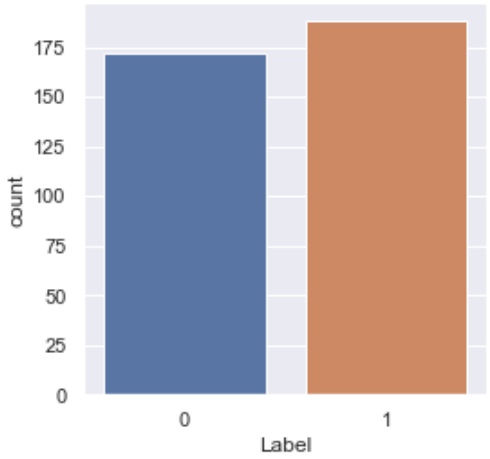


Fig 1: A bar plot showing the number of examples from each class.

INFERENCE:

The number of examples from each class (0 and 1) was counted and it was found that there is no issue with class imbalance in this dataset. Thus, we would need to use techniques to deal with class imbalance while making a model.

Pearson correlation coefficient between the features

The Pearson correlation method is the most common method to use for numerical variables; it assigns a value between – 1 and 1, where 0 is no correlation, 1 is total positive correlation, and – 1 is total negative correlation. This is interpreted as follows: a correlation value of 0.7 between two variables would indicate that a significant and positive relationship exists between the two. A positive correlation signifies that if variable A goes up, then B will also go up, whereas if the value of the correlation is negative, then if A increases, B decreases. The pearson correlation coefficient between the features and the label in the training data set was found out.

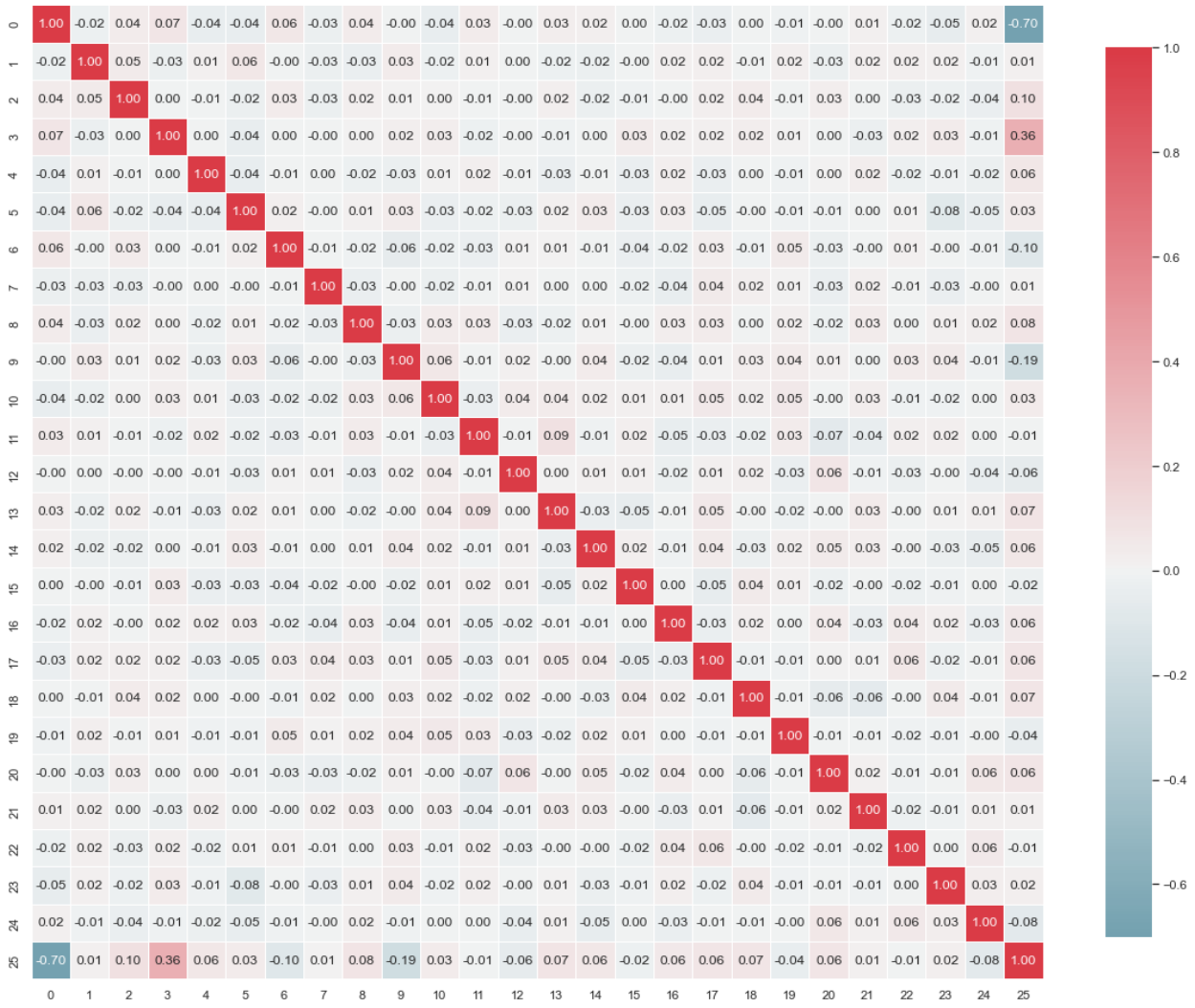


Fig 2: The Pearson correlation coefficient between each feature obtained from training dataset

INFERENCE:

From Fig 2 it is observed that none of the 25 features have considerable Pearson correlation coefficient with each other. Column number 0 has -0.7 correlation coefficient with the label and column 3 has 0.36 correlation coefficient with the labels. This implies that the label might have high linear dependence on the columns 0 and 3.

Testing multicollinearity using variance influencing factor (VIF):

Collinearity occurs because independent variables that we use to build a regression model are correlated with each other. This is problematic because as the name suggests, an independent variable should be independent. It shouldn't have any correlation with other independent variables. If collinearity exists between independent variables, the key point of regression analysis is violated. In regression analysis, we want to isolate the influence of each independent variable to our dependent variable. This way, we can interpret the fitted coefficient of each independent variable as the mean change in the dependent variable for each 1 unit change in an independent variable while keeping the other independent variables constant. Now if we have collinearity, the key point above is no longer valid, as if we change the value of one independent variable, the other independent variables that are correlated will also change.

Variance Inflation Factor or VIF measures the influence of collinearity on the variance of our coefficient estimates. There are a lot of discussions about what would be the appropriate threshold value of VIF before we decide that the collinearity of our independent variables is a cause of concern, but most research papers agree that VIF above 10 indicates a severe collinearity among independent variables. If VIF is close to 1 then it implies that it is not correlated, or multicollinearity doesn't exist. If VIF is close to 5, it implies that it is moderately correlated. If VIF is greater than 7, it is regarded as highly correlated or high multicollinearity exists. I have calculated the VIF values for each of our features.

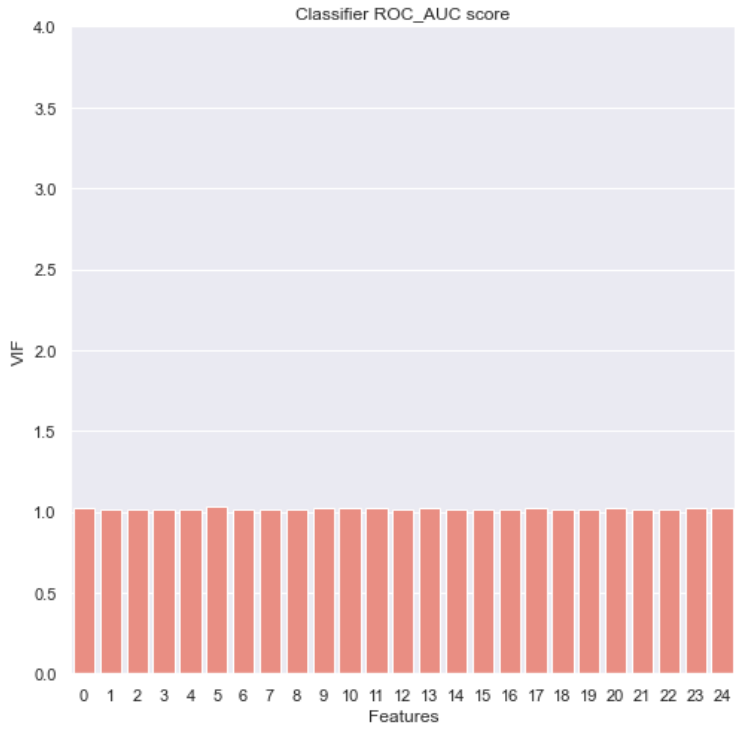


Fig 3: Variance Influencing factor for each of the features

INFERENCE:

From Fig 3 it can be observed that VIF values for each of our features are close to 1. This implies that multicollinearity doesn't exist in out training data set. Thus, we don't have to work or spend time on removing the multicollinearity from our dataset.

Checking the difference in the conditional probability distribution of feature given the label using kernel density estimation. (KDE Plot)

KDE Plot described as Kernel Density Estimate is used for visualizing the probability density of a continuous variable. It depicts the probability density at different values in a continuous variable. KDE plots were generated conditioned on the label (0/1) to study the difference in the probability distributions of each label.

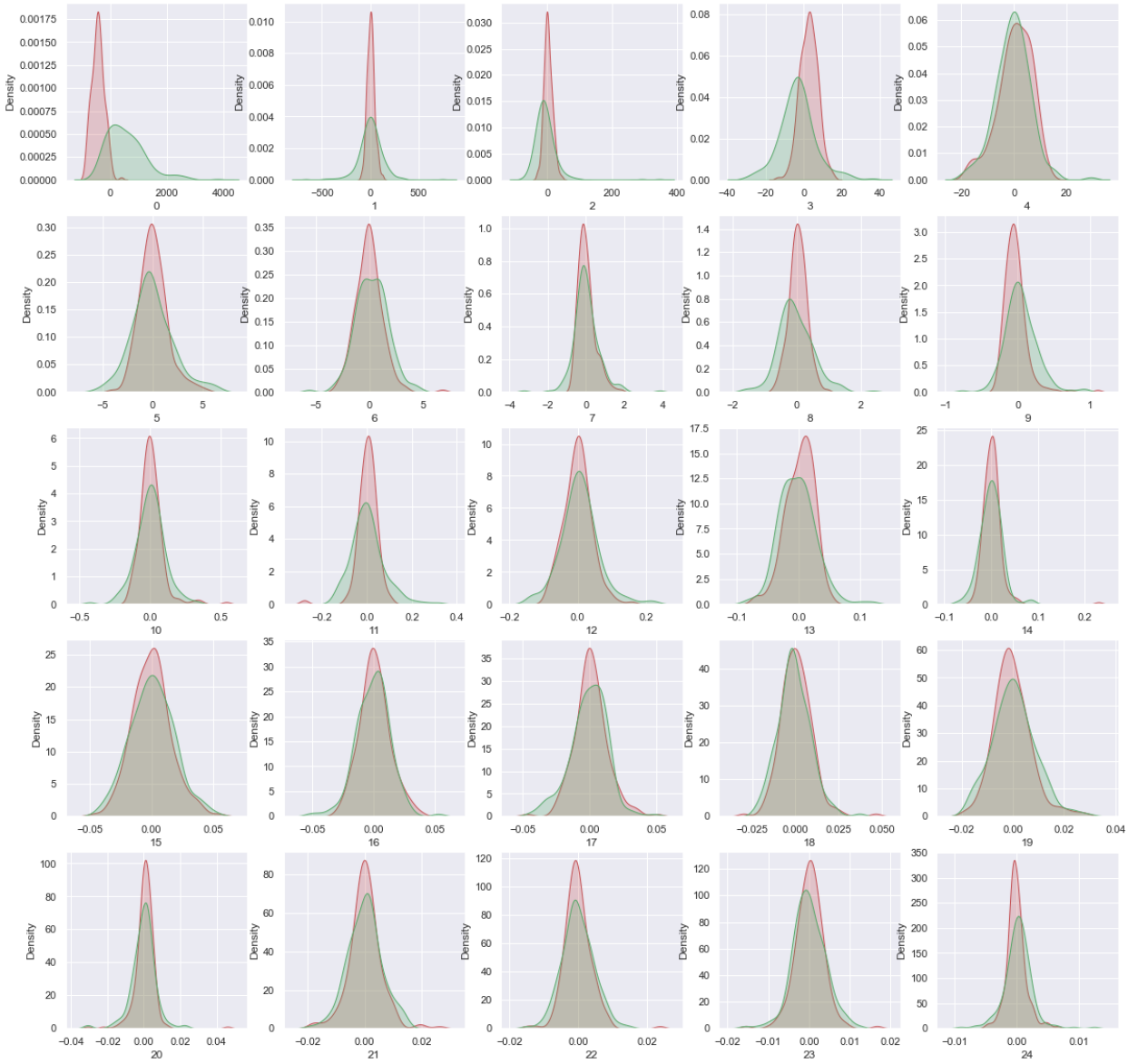


Fig 4: KDE plots for each feature. Here the red color stands for the positive examples and green color for the negative examples.

INFERENCE:

From the KDE plots it can be observed that the probability distribution of features given the label changes significantly for the features in column number 0, 1, 2,3,5,8. Thus these features might be having more influence on deciding the label. This information will be useful if we are required to perform feature engineering or feature selection to improve the model during training.

Using Pair Plots to understand the significance of each feature and pairs of features.

A pair plot allows us to see both distribution of single variables and relationships between two variables. Pair plots are a great method to identify trends for follow-up analysis and, fortunately, are easily implemented in Python A pair plot was generated to understand the influence of each of our features.

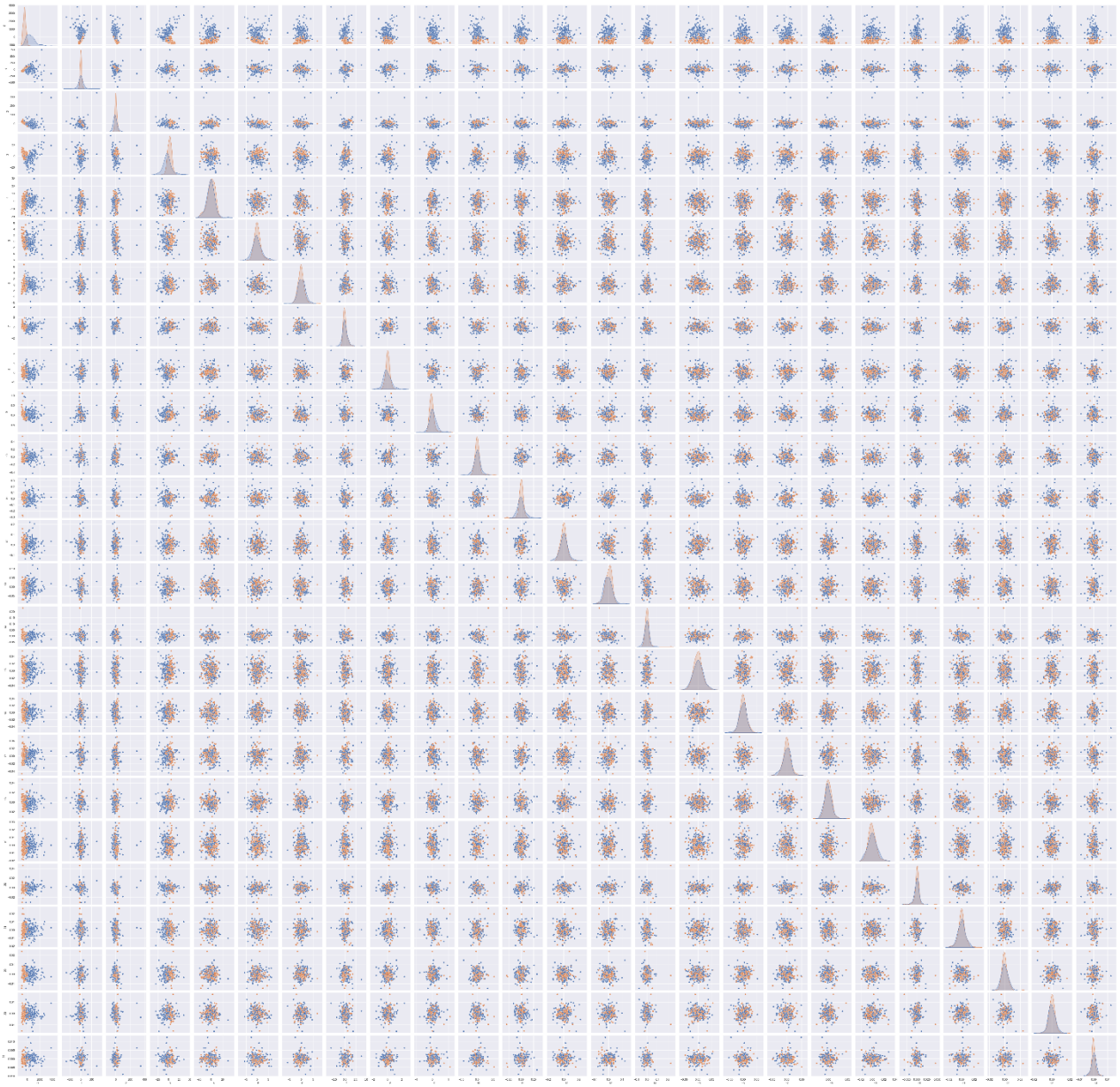


Fig 5 : pair plot made using all 25 features. Blue color for points in the feature space with label 0 and red color for points in the feature space with label 1.

INFERENCE

The pair plot can be used to decide the influence of each feature by looking at the separation of the red and blue points. If there is a significant pattern or difference in the distribution of red and blue points in the feature space, then those pairs of features are significant in the determining the labels.

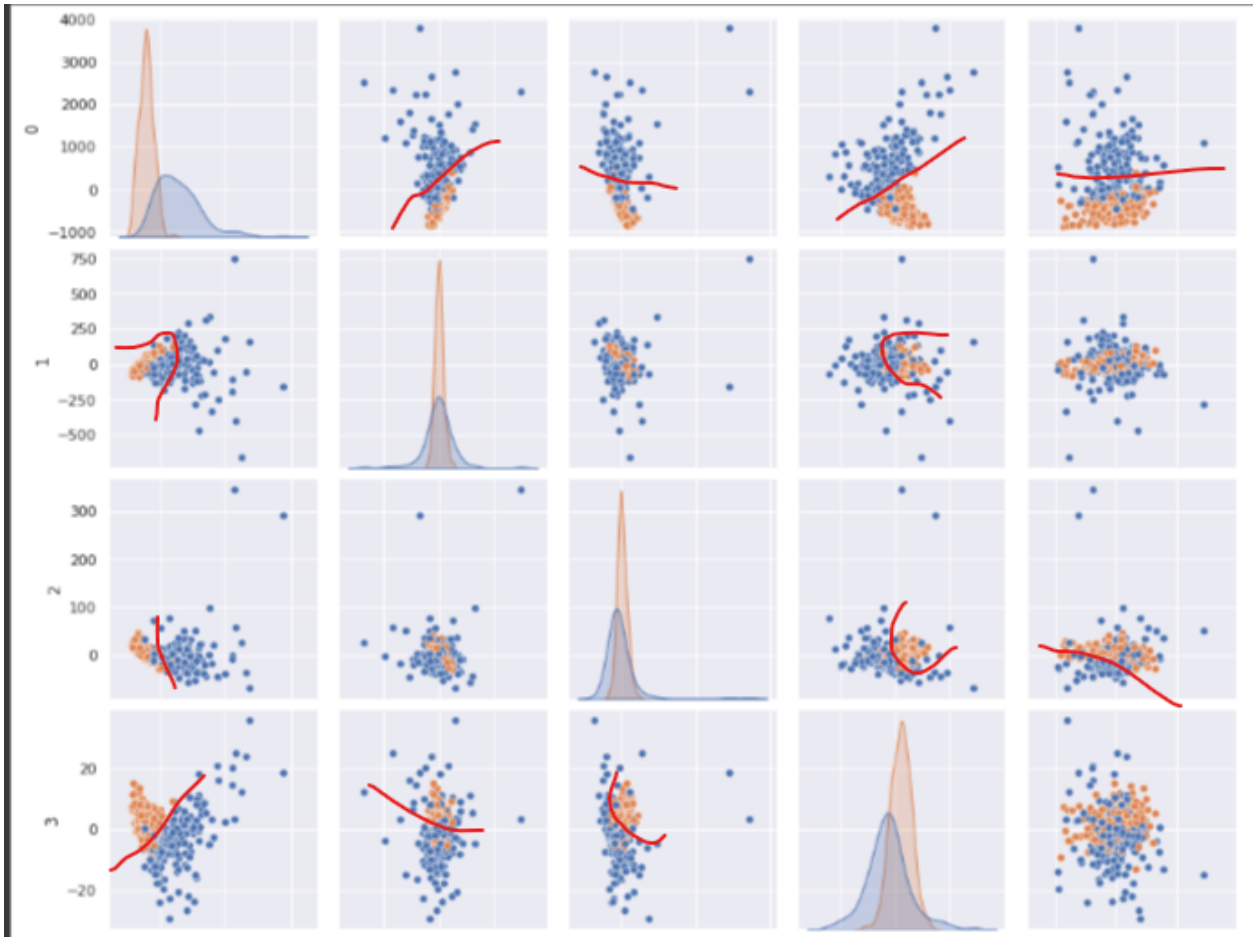


Fig 6: A snippet of the above pair plot which shows features spaces with obvious decision boundaries marked with bright red curves.

Model Training

In this section we will be looking at the following:

1. The XGBoost model used for the classification problem
2. Comparison with other algorithms used for classification
3. Comparison with the ensemble model that was used for classification
4. Why XGBoost was selected as the final model after considering lots of different algorithms.

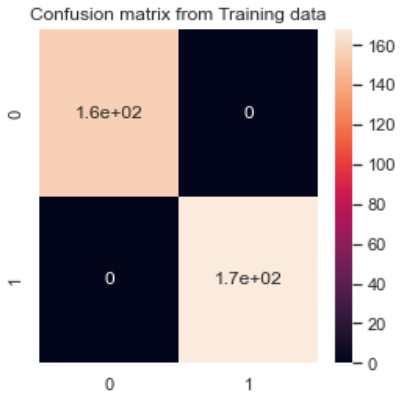
XGBoost model

XGboost was taken as the starting point to attack this binary classification problem. This was done because XGBoost is known to have delivered benchmark results in various ML competitions and is also one of the most used algorithms in Kaggle best models in various challenges. XGBoost is also considered to perform well in the case of tabular data without categorical features which is a condition matching with our problem.

The training dataset provided in the competition was split into training and validation sets with a validation set consisting of 10 percent of the initially provided training dataset. Hyper parameter tuning was performed using grid search method and a suitable set of hyperparameters were obtained.

Performance of the XGBoost model

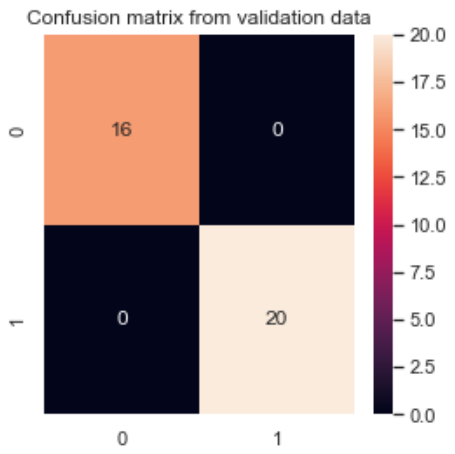
1. **On the Training dataset.**



The confusion matrix plotted from the training dataset shows that the XGBoost model was able to perfectly fit the training data without any false positives or false negatives.

Fig 7 : Confusion matrix from the training data using XGBoost classifier

2. On the Validation dataset:



The confusion matrix plotted from the training dataset shows that the XGBoost model was able to perfectly fit the validation data without any false positives or false negatives. This implies there isn't overfitting in the model.

Fig 8 : Confusion matrix from the validation data using XGBoost classifier

By comparing the performance of the XGBoost model on the training dataset and validation dataset it can be suggested that the trained XGBoost model is able to generalize the classification task to unseen / new data points without overfitting the training data set. Later we will be looking at some other algorithms to compare the results with that of this XGBoost. But looking at these results I am pretty sure that we can directly use this trained XGBoost to obtain good results for the testing dataset of this challenge.

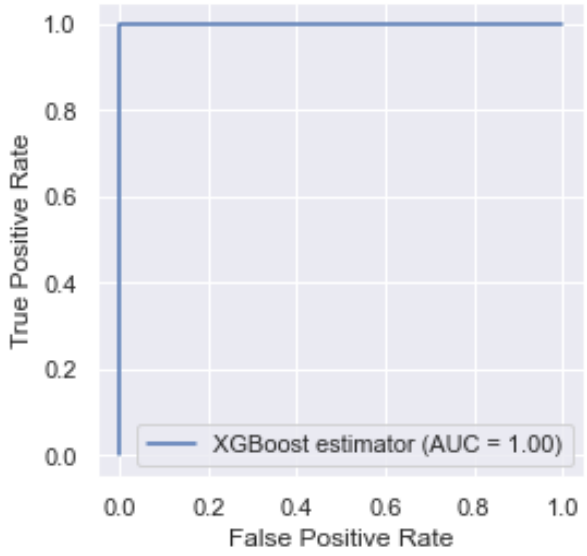


Fig 9: ROC curve plotted from the prediction of validation set using XGBoost estimator

Comparison with other trained classification algorithms.

A few other classical machine learning classification algorithms like K- Neighbors algorithm, Support vector machine, Decision tree classifier, random forest classifier, AdaBoost, Gradient boost classifier, Gaussian naïve bayes, Linear discriminant analysis, quadratic discriminant analysis, Logistic regression were used to train this classification problem. The following plot shows the comparison of these algorithms with the XGBoost algorithm.

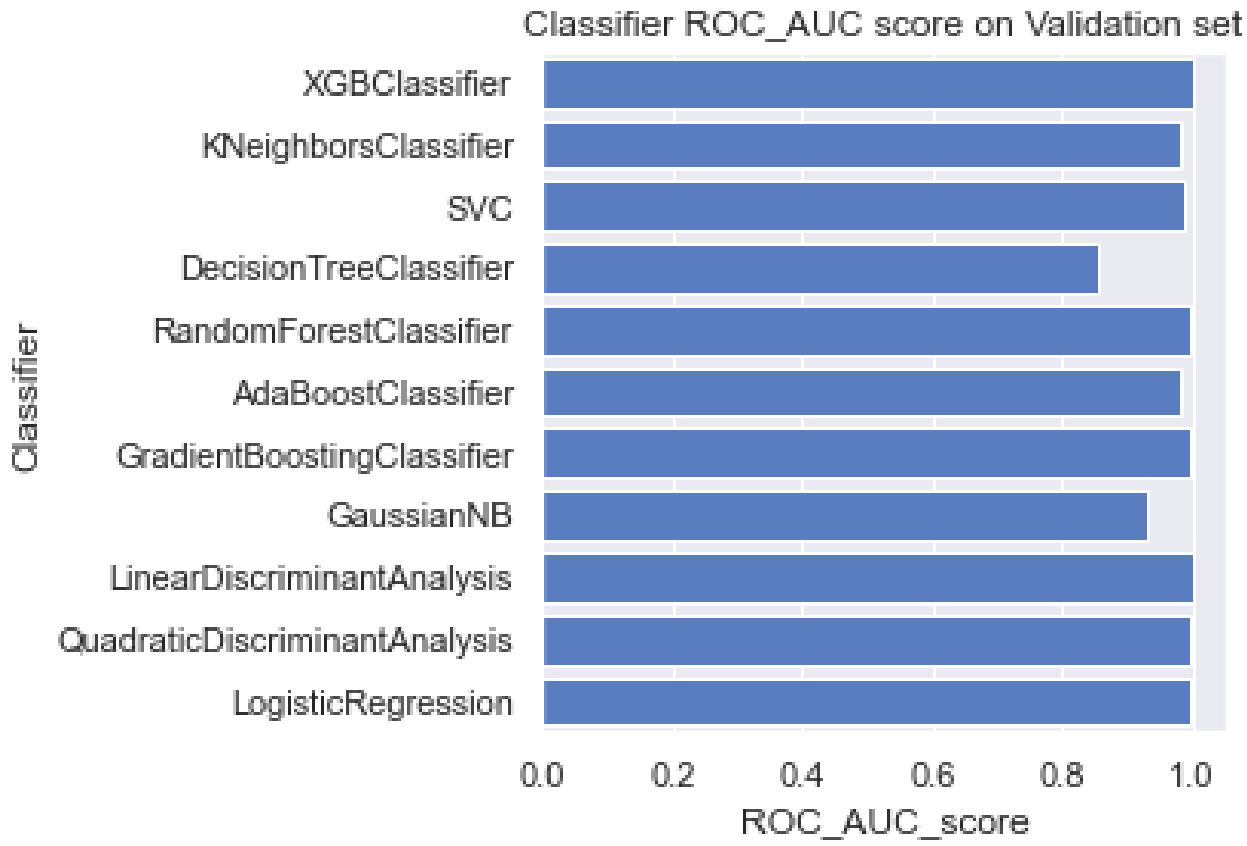


Fig 10: Comparison of different machine learning algorithms on the classification task on the validation set that wasn't used for training.

It was observed that many of these algorithms were able to provide similar classification power for our classification task. This was tested several times by sampling different validation sets and training the models again and it was observed that many of these models were performing on par with the XGB classifier.

Ensemble model:

An Ensemble model by combining the best performing machine learning model was used to compare the results with the trained XGBoost algorithm.
The Ensemble included the following models

1. XGBoost
2. AdaBoost
3. Gradient Boosting
4. Linear discriminative analysis
5. Quadratic discriminative analysis
6. Logistic regression

The average of the predicted probabilities from the above models was used to classify the validation set. The following are the results obtained.

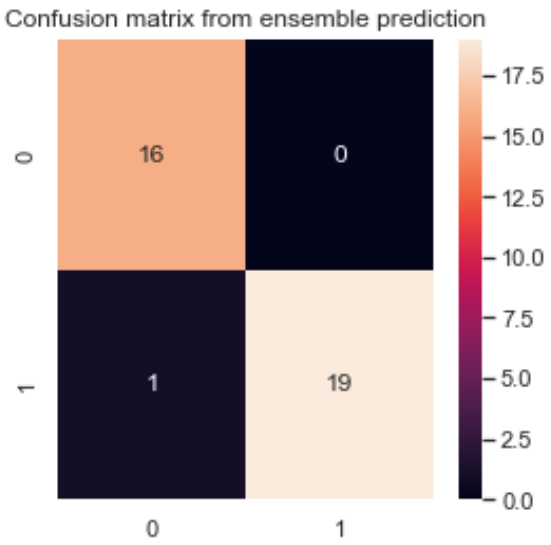


Fig 11: Confusion matrix obtained from the prediction on the validation set using the ensemble.

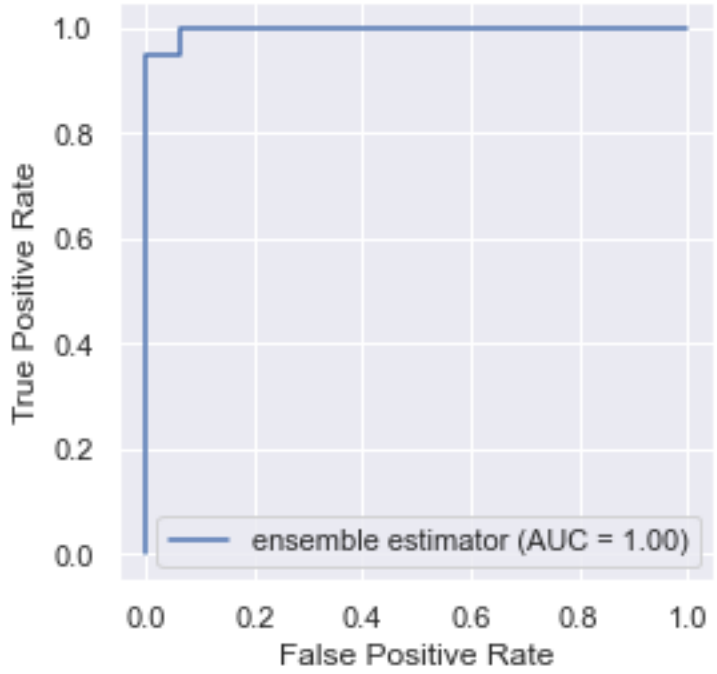


Fig 12: ROC curve obtained from the predictions on validation set using the ensemble model

From Fig 11 and Fig 12, i.e., the confusion matrix and the ROC curve obtained from the ensemble model on the validation set, it was observed that the performance of the trained single XGBoost model is comparable to that of this ensemble model.
This result was verified several times by using different combinations of training dataset and validation set and similar observations were made.

Why was the XGBoost model used for final predictions using test dataset?

Even though it was observed that many different machine learning models were able to perform comparable to the XGBoost model, I have chosen XGBoost as the Final model because I have spent more time on hyperparameter tuning of the XGBoost model than other machine learning algorithms for this classification problem.

XGBoost was chosen over the combined ensemble model even though both of them have shown comparable results because the single XGBoost model is

computationally more efficient than the combined ensemble model and it doesn't make any sense to choose the ensemble model without significant improvement in the classification capabilities.

Thus, the XGBoost with the same hyperparameters were trained again using the entire training dataset provided and were used to predict the classes of the test dataset.

Why do we believe that these trained models will be able to generalize to unseen test dataset?

These models were able to generalize well to unseen validation dataset. But the validation data set is sampled from the training dataset, and we have shown that the classification models are able generalize to unseen validation sets from the same distribution. Thus, if the check whether the testing dataset and training dataset have a similar probability distribution for its features, we would be confident that the trained classification models will be able to generalize well to the testing dataset and the models will give good results on the testing dataset.

From the following plot comparing the probability distributions of individual features of training dataset and testing dataset, it can be observed that both the distributions peaks around the same values of all the features and have comparable shapes.
Thus, we can believe that the training dataset and testing dataset was obtained from the same distribution, and we can believe that the model will generalize well to the testing dataset.

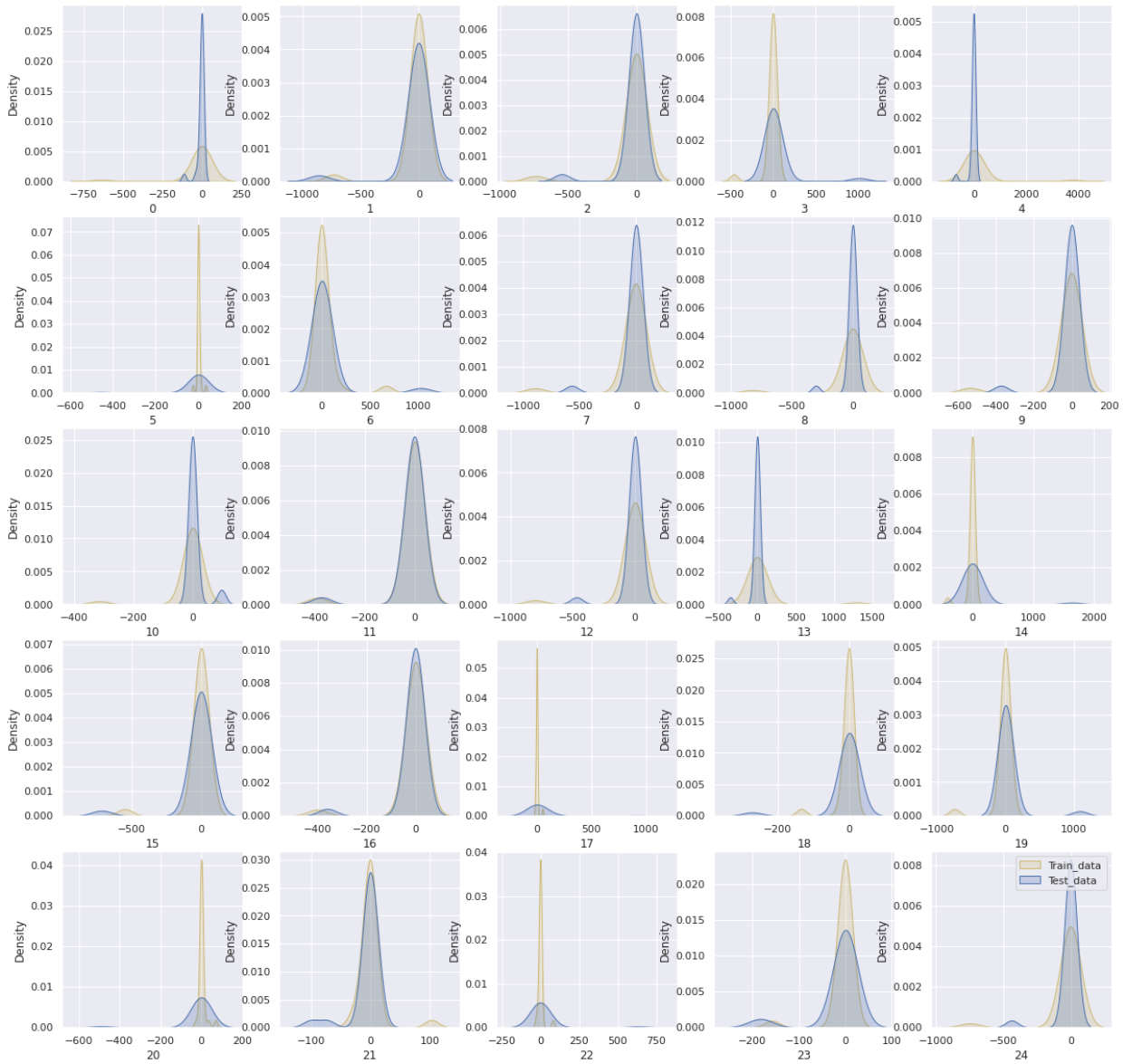


Fig 13: Comparison of the probability distributions of the individual features of the training and testing dataset