DMG Assignment 2 Report

Akanksha Pandey MT20048
Pradeep Kumar MT20036

1. Methodology: Approach and Reason

1. **xgboost classifier.**
   XGBoost stands for eXtreme Gradient Boosting. It is an implementation of gradient boosting machines. This comes under the boosting technique of ensemble methods.

   Advantages of using Xgboosting :
   1. Regularization
   2. High flexibility
   3. Handles missing values
   4. Tree Pruning

   Reason: We tried using a single decision tree for classification, but the score was not that impressive. That's where we decided to go for some advanced techniques like ensemble methods. Since xgboost provides the flavour of boosting technique.

   Preprocessing:
1. We have deleted the 'id' column
2. We have done Under Sampling with the majority because we have imbalanced classes i.e, number of attributes with 1 class are very large then 0 class.

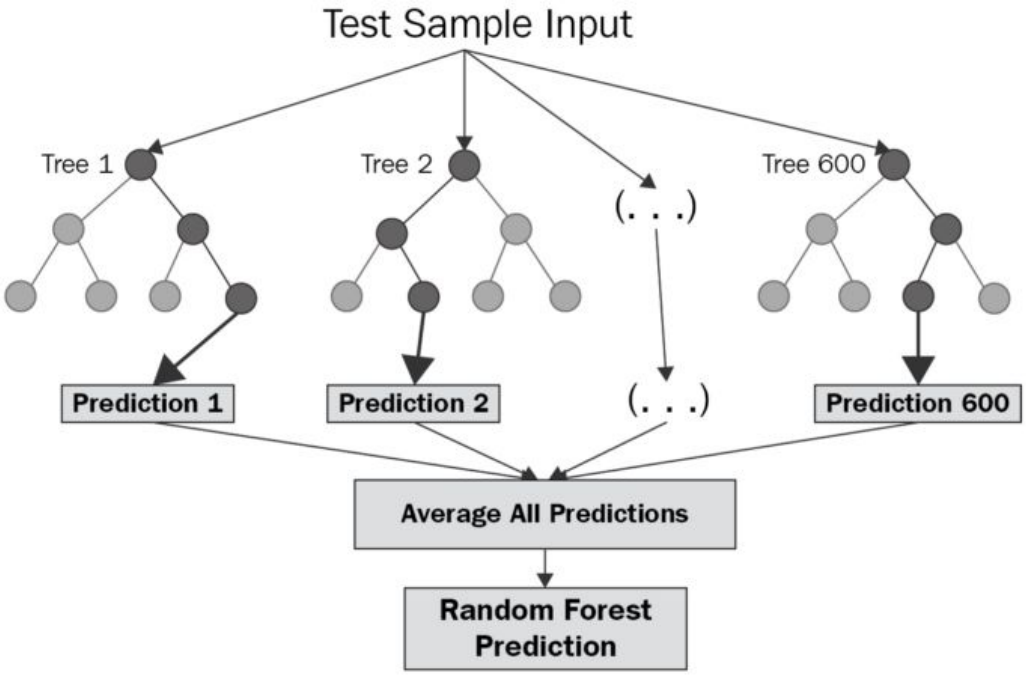
   After removing 'id' column:
   1. Dimension of 'X' Before Undersampling: [2041687 rows x 7 columns]
   2. Dimension of 'X' After Undersampling: [40634 rows x 7 columns]

   Parameters: We have used following parameters for tuning the random forest classifier:
1. **n_estimators :** The number of trees in the forest.
2. **max_depth :** The maximum depth of a tree that is allowed.

   n_estimators=300, max_depth=10

2. **Random Forest Classifier:**

   Random Forest Classifier is an ensemble learning method. It is a collection of Decision Trees known as a forest. It constructs a multitude of decision trees and these decision trees then vote to find the most popular class. This most popular class is given as the output by the Random Forest Classifier.
   Random forest is a bagging technique. It combines the result of multiple predictions.



   Reason: After trying xgboost successfully we were searching for another good technique that can give us comparable results. As random forest decreases the curse of a simple decision tree which is to overfit the training data and provide a model that will less overfit the data and perform better on testing data.

   Preprocessing:

1. We have deleted the ‘id’ column
2. We have done Under Sampling with the majority because we have imbalanced classes i.e, number of attributes with 1 class are very large then 0 class.

   After removing ‘id’ column:

   3. Dimension of ‘X’ Before Undersampling: [2041687 rows x 7 columns]
   4. Dimension of ‘X’ After Undersampling: [40634 rows x 7 columns]

Hyperparameters: We have used following hyperparameters for tuning the random forest classifier:

1. **n_estimators :** ***int, default=100 :*** The number of trees in the forest.
2. **max_features:** The number of features to consider when looking for the best split.
3. **Min_samples_leaf:** ***int or float, default=1:*** The minimum number of samples required to be at a leaf node.
4. **min_samples_split*****int or float, default=2:*** The minimum number of samples required to split an internal node.
5. **Max_depth:** ***int, default=None:*** The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
6. **Random_state:** Controls sampling of the features to consider when looking for the best split at each node.

n_estimators=300, max_features=6, min_samples_leaf = 3, min_samples_split=25, max_depth=50, random_state=80

3. **Random Forest Regressor**

   Similar to the above random forest classifier but the only difference is in the way it combine the results.
   It combines the results by taking the mean predictions coming from different tree sets. Whereas the classifier combines the results by taking the mode of classes.

   Reason: To try out classification problem using a regressor.

   Preprocessing:
   1. We have deleted the 'id' column

Hyperparameters: We have used following hyperparameters for tuning the random forest regressor:

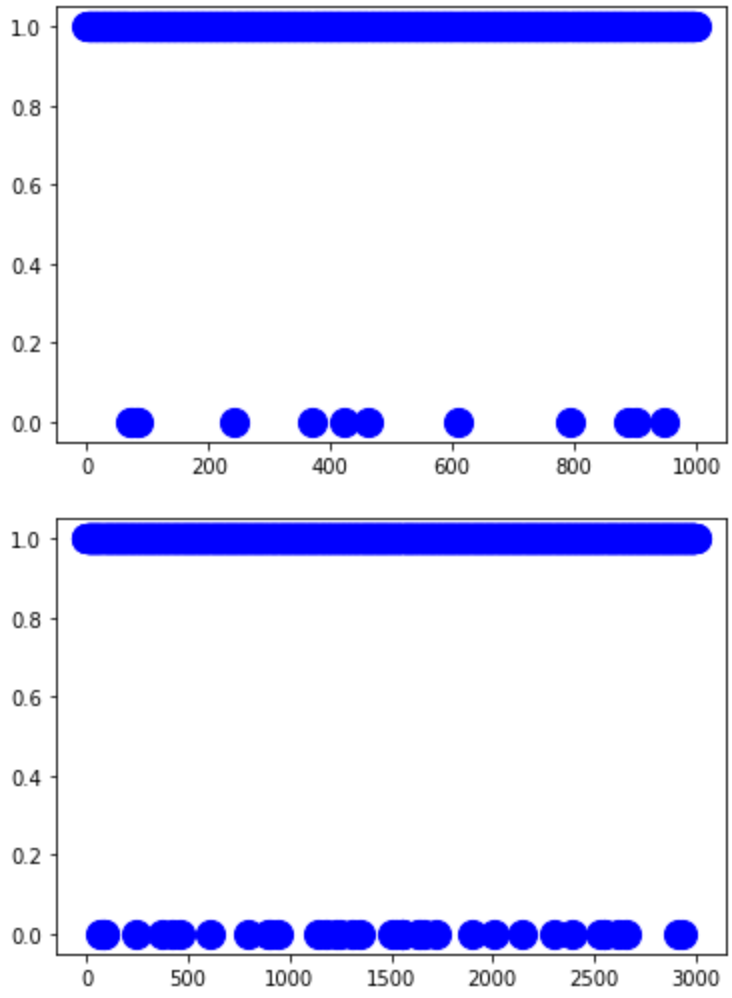
1. **n_estimators :** ***int, default=100 :*** The number of trees in the forest.
2. **max_features:** The number of features to consider when looking for the best split.
3. **Min_samples_leaf:** ***int or float, default=1:*** The minimum number of samples required to be at a leaf node.
4. **Min_samples_split:** ***int or float, default=2:*** The minimum number of samples required to split an internal node.

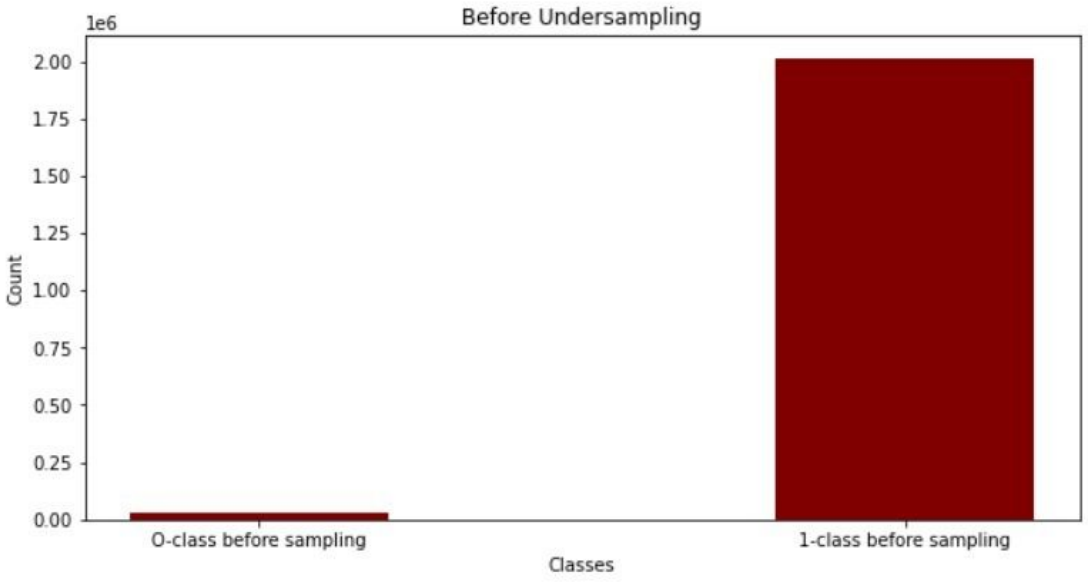
5. **Max_depth:** ***int, default=None:*** The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
6. **Random_state:** Controls sampling of the features to consider when looking for the best split at each node.

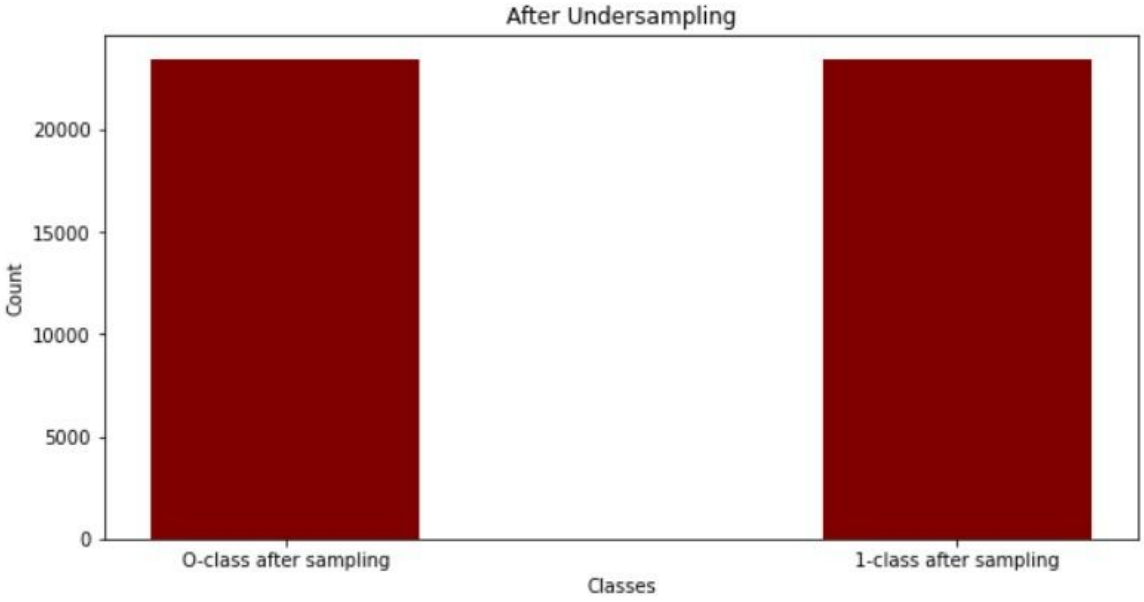
max_depth=15, max_features=6, min_samples_leaf=3, min_samples_split=25, random_state =0

2. Visualization skewness of data before and after preprocessing

Showing class Imbalancing



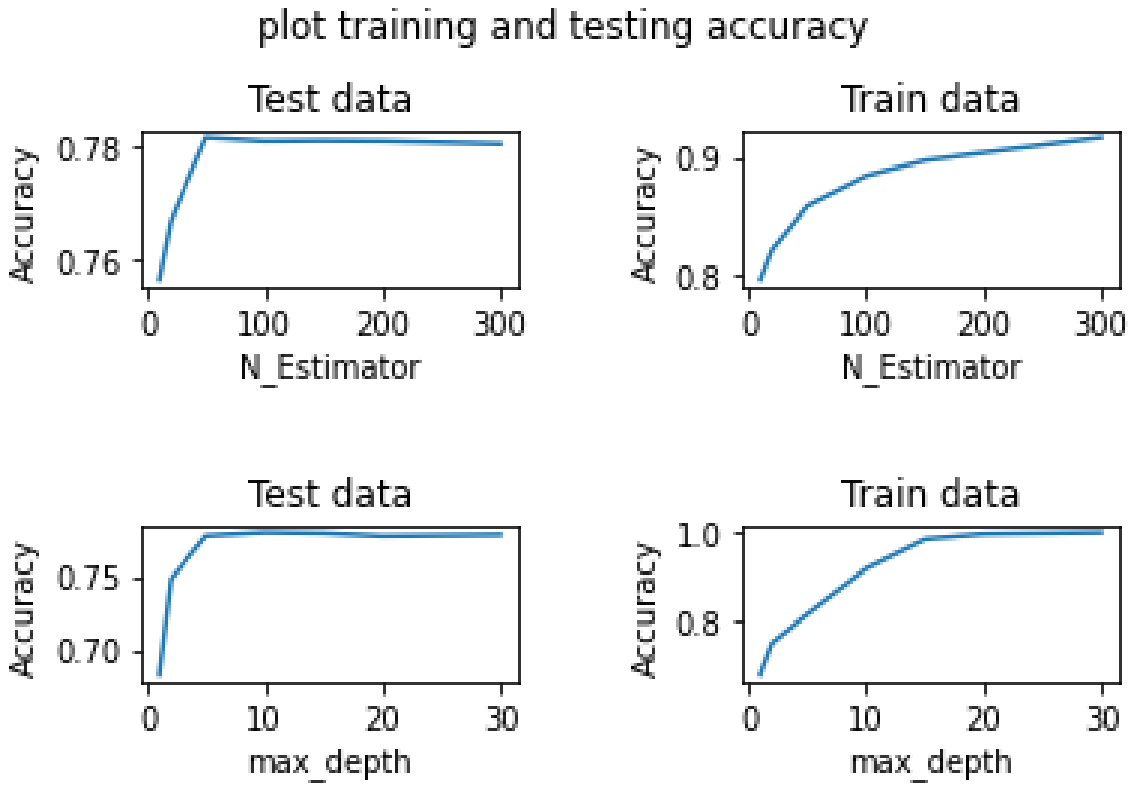




3. Plot training and testing accuracy w.r.t hyperparameters of the model

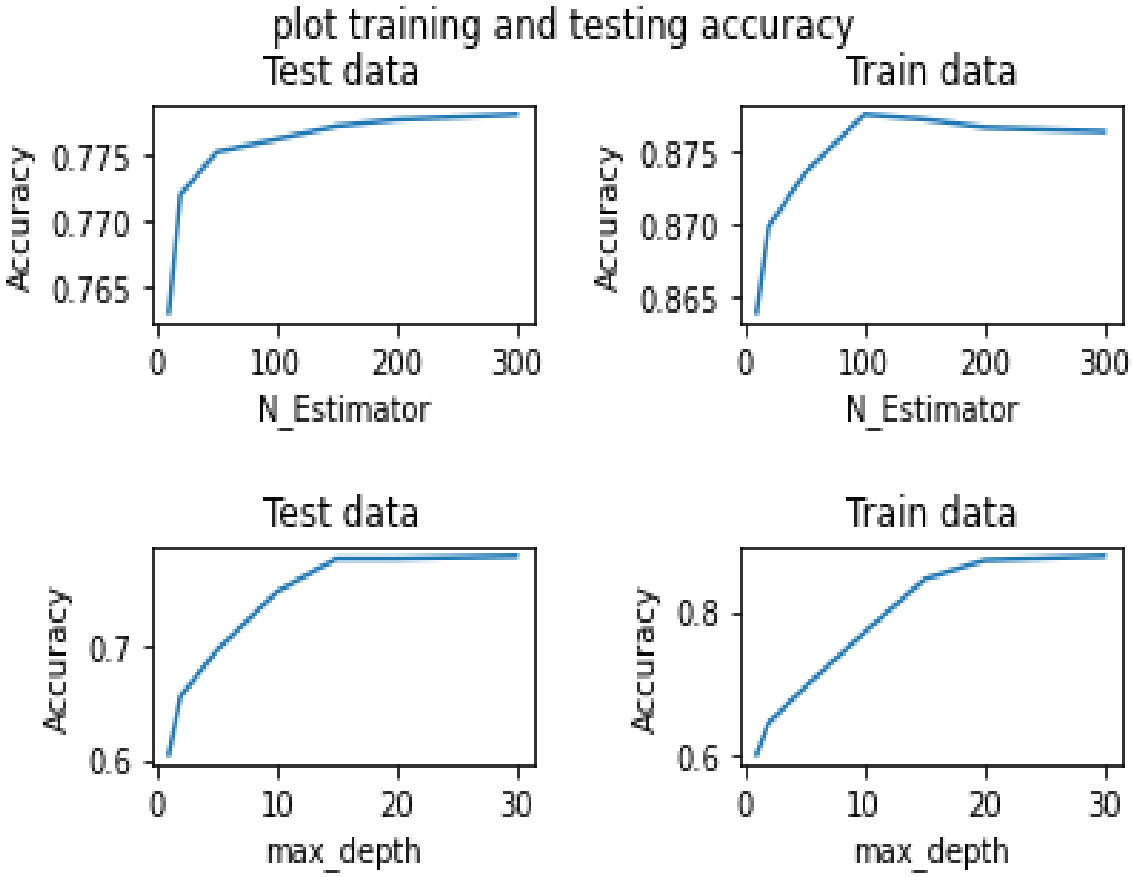
Plottings for xgboost:

Here we have considered **n_estimator** and **max_depth** as hyperparameters and the plotting is done using pyplot.



Plotting for randomForestClassifier

Here we have considered n_estimator and max_depth as hyperparamter for seeing the changes in accuracy which is the y-axis of the plot shown below.



Plotting for randomForestRegressor

For RandomForestRegressor we have considered min_samples_split and max_depth as hyperparameter for plotting the variation in accuracy as we vary the parameters.

Drive Link:

https://drive.google.com/drive/folders/1vpvMcza2n2T9um0LymySiAk3NS71sSdi?usp=sharing

Our Learnings:

In this particular assignment we have various things as listed below:

1. Use of kaggle
2. Various classification Techniques and their implementation on a real data.
3. How to make a decision tree from scratch.
4. How to handle the imbalanced data.
5. Team work
6. Use of colab in a more efficient way