CSCI-P556
Fall 2018
Assignment 3
Due 11:59PM, Nov. 2, 2018
Data Analysis Of Unknown Data Set

Name (prgadugu)

November 5, 2018

1 Introduction to Unknown Data Set.

1.1 What's in the Data?

The given data is from an unknown source; Following are my findings related to the data:

- No Feature or Data related background information.
- Training Set has 2000 rows of data and 500 features.
- Test Set has 600 rows of data and 500 features.
- Target variable is {-1,1}. Hence it is a Binary Classification.

1.2 How to proceed?

- Load the data into DataFrame and clean it.
- Look into the features so as to find something interesting.
- Understand what type of Models needs to be applied for Classification.

2 Exploratory Data Analysis

I have imported the train and test files(including the labels) into a pandas Dataframe, under the names: train_data, test_data, train_label and test_label.

At the outset, I have checked for the unique identifier columns by running a loop and checking for the unique values in each column and checking if it is equal to length of the column. Appended if any into the list, "List_Of_unique_identifier_coloumns" . But it returned out to be empty.

Then, I have used the function, isnull(), to check for any missing values, and found nothing.

There after, I proceeded to check whether the labels are balanced or not using the "seaborn" plot which suprised me with perfectly balanced binary labels as shown below:

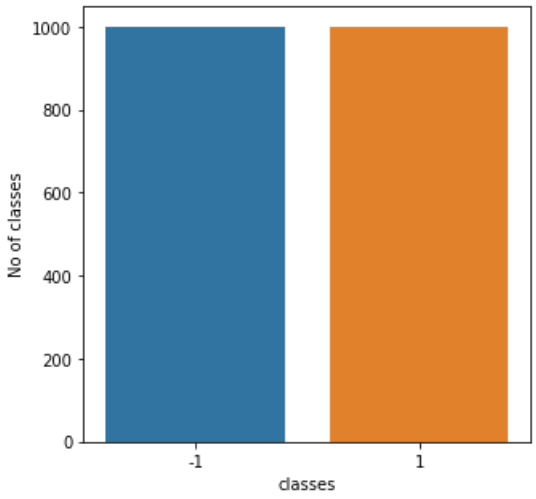

Figure 1: Labels are balanced

The labels are perfectly balanced as shown in the above figure.

3 Baseline Models

I have tried the following Baseline Models,

- Logistic Regression: After the pre-processing of the data, I have trained the Logistic model using the sklearn LogisticRegression module. The accuracy results are as follows:
  Train Accuracy = 74.5
  Test Accuracy = 59
- KNN classifier:I have trained the K Nearest Neighbour model using the sklearn KNeighborsClassifier module. The accuracy results are as follows:
  Train Accuracy = 82.65
  Test Accuracy = 69.16
- Decision Trees: I have trained the Decision Trees model using the sklearn DecisionTreeClassifier module. The accuracy results are as follows:
  Train Accuracy = 100
  Test Accuracy = 75.5
- Random Forests: I have trained the Random Forest Classifier model using the sklearn RandomForestClassifier module. The accuracy results are as follows:
  Train Accuracy = 100
  Test Accuracy = 74

Logistic Regression model gave very low test Accuracy compared to the other models. KNN is better compared to Logistic regression. Both Decision Trees and Random Forests are over-fitting the train set as they are giving a 100 percent train but low test accuracy (better than other models used).

4 Feature Engineering

Firstly I have checked if, there is any correlation between all the features by forming a correlation matrix. Only one feature was dropped when a correlation factor of 0.9 is used. The heat map given in the Jupyter Note Book explains this better.
Finally I have used the Random Forests "Feature importance" module which gives the importance of each feature.

Based on the importance I sorted them and ran a loop with 20 increments(to select the features) and calculated the test accuracy, found that the test accuracy is more for the top 20 features and keeps on decreasing when we select more features. The graph is shown below:

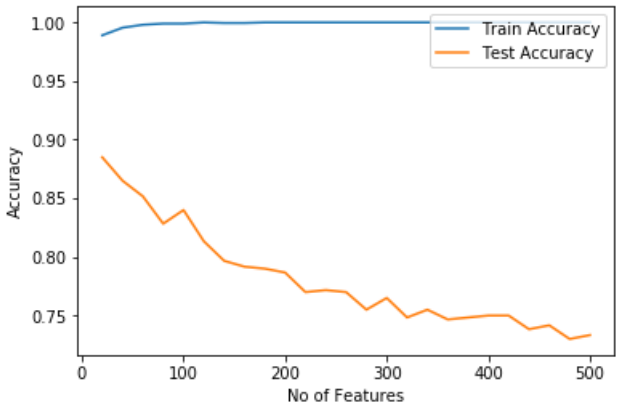

Figure 2: Feature Importance

As shown in the graph the model tries to over-fit, once the features start to increases. Hence the test accuracy drops. So, I took the trade-off point with maximum test accuracy and decent train accuracy. This point comes for top 20 features.

Moving further into the Model Building, I will be using these 20 features which came out to be the best(as the bagging process is involved in the Random Forests) in my model building.

5 Model Building

Every model had its own advantages and disadvantages, but I wanted a model which would have a combined performance of all the models. To do that I started improving each model through hyper parameter tuning.

Firstly, I started with the logistic regression and used the features that I got after Feature Engineering. I tuned the Hyper parameters(penalty and lambda value) and tried to find the performance.

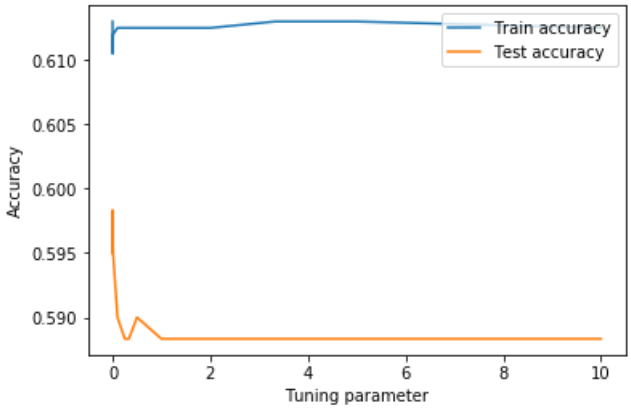

Figure 3: Logistic Regression Hyper Parameter Tuning

Secondly by changing the number of neighbours in the KNN model I calculated the test accuracy, based on which I found that for k = 12 we get the best test accuracy. The results are shown below,

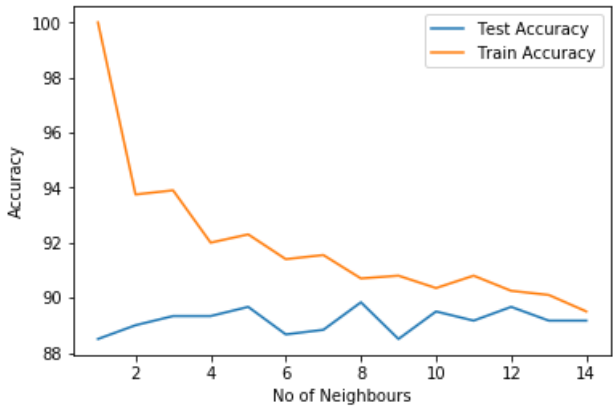

Figure 4: KNN Hyper Parameter Tuning

In the similar fashion I have tuned the Hyper parameters of the Random Forests, using the Randomized search which gave the best parameters. After finding out all the Hyper parameters I created the necessary models with best known hyper parameters to fit them into the Ensemble, which gave me my final model which provided the best results. So when I finally ran my model it gave me a range of test accuracy between 90 to 91.

6 Discussion

Finally, describe how the models compared against each other and the baselines. Was their performance as good as you expected? Were there any challenges? Is there anything that could improve your models performance?

6.1 Comparision Of Models

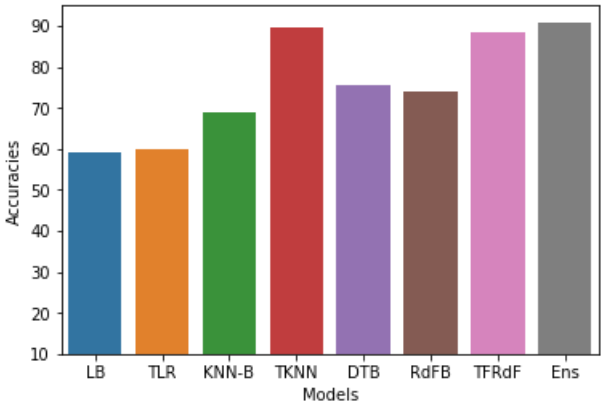

Figure 5: Accuracies Comparisions across different Baselines and the final model.

The above figure gives the information about the baseline models I used and the respective accuracy comparisions. The abbreviations being

- LB(Baseline Logistic Regression) vs TLR (Tuned Logistic Regression):
  There is a very minute change in the accuracy after the tuning of hyper-parameter and training on the feature engineered train set.

- KNN-B(Baseline KNN) vs TKNN(Tuned KNN):
  After tuning, KNN showed a much high improvement with neighbours being 12 for the feature engineered train set. I further used this model in my ensemble model.
- DTB (Decision Trees Baseline) vs :
  The Decision Tree Baseline Model overfits(training accuracy of 100 percent) the original train data as the data is huge. After the feature engineering also scenario is the same, bu the test accuracy has a significant improvement.
- RdfB(Random Forests Baseline) vs TFRdf(Tuned and Feature Engineered Random Forests):
  After the Feature engineering and parameter tuning there is an improvement in the Random Forest Model which gives the test accuracy of 89.6.
- Ens(Ensembled Final Model): After selecting the best sets of all the models and and assembling them using ensemble I got the best test accuracy of 91 percent.

6.2 Challenges Faced

While selecting the best hyperparameters(using Grid Search and RandomisedCV), I faced issues with Jupyter Note Book, hence executed the modules in the "Burrow" server.

The code snippet is given in comments in my Jupyter notebook. After selecting the respective hyperparameters I substituted them in the "ensemble".

6.3 More Improvements

In the whole process of my estimation I have used the principle of "Bagging" and "Max Voting" procedures which have individual models being independent of each other.
Bagging and Boosting are the famous ensemble techniques, Both are ensemble methods to get N learners from 1 learner but, while they are built independently for Bagging, Boosting tries to add new models that do well where previous models fail.
Using the Gradient Boosting and Ada Boosting Models would have significantly improved the performance.