HackStat 2.0 - 2k19

First Round Competition Report

- **Team Leader** : Tharaka Rehan
- **Team Members** : Janith Ganepola
  Sahan Gurusignhe
  Thisun Dayarathna
  Gayangana Leelarathne
- **University** : University of Moratuwa
- **Team Name** : Pathfinders404
- **Kaggle Username** : Pathfinders404
- **Display Name** : Pathfinders404

Introduction

In this problem we are given with a dataset containing data of visitors of a website and the task is to predict whether a given visitor would be a revenue generating customer or not. The dataset consists of multiple dependent variables which are either numerical or categorical. First, we had to observe the user characteristics variation in accordance with the time spent on the website (*Homepage Duration, About us Duration* etc.), the place of origin of the user (*Province, Browser, Operating System*) and the other given features.

We have approached this binary classification problem as a typical supervised learning problem, after observing the relationships exist between 'Revenue' and other dependent variables. After several attempts we realized that logistic regression is not a suitable method of handling this problem as the training dataset is biased towards non-revenue generating customers. Therefore, ***Random Forest Algorithm*** was chosen as the classification model. We got significant improvements on accuracy with this method.

Methodology

1. Selection of language for Programming

Python was used as the basic programming language for developing the solution.
Several python libraries were used including ***NumPy, Pandas, Sci-kit Learn*** to achieve the task.

2. Data Pre-Processing

With the assistance of *SK-learn preprocessing* libraries we fill out missing data in some rows using the library's class *Imputer.* Then the categorical data was encoded to numerical values. The feature column *Month* was transformed into dummy categories to get more unbiased results (using *OneHotEncoder*).

Then we generated the correlation matrix and omitted some feature columns accordingly. After that all the data were scaled and normalized. In addition, we intended to oversample the data using *Smote/Smote-NC* as the dataset was imbalanced one. But as there are a lot of categorical features it was not performed. So, we tried to deal with that selecting a better classification method and algorithm.

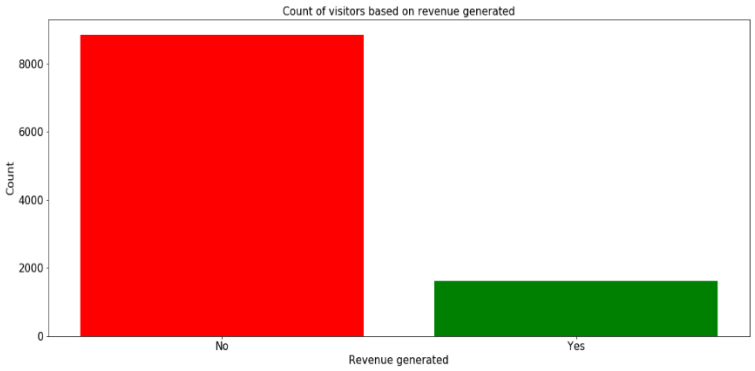

Figure 1 : Histogram of the Dataset (Biased Towards Non-Revenue Generating Customers)

3. Selection of the Classification Model

As mentioned above we have used sk-learn built in library to predict the outputs. Major hyperparameters of the model including *n_estimators, max_feature, min_samples_leaf, min_samples_split* etc. was tuned in order to prevent overfitting or underfitting.
It had been noticed that increasing *n_estimators* parameter for larger values courses overfitting the training data. Therefore, these parameters were carefully tuned for maximum accuracy.

```
rf = RandomForestClassifier(bootstrap=True, class_weight=None,max_depth=8, max_features=10, max_leaf_nodes=None,
                            min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100,
                            n_jobs=1, oob_score=False, random_state=None,verbose=0, warm_start=False)
rf.fit(X_train, y_train)
```

Figure 2: Hyperparameters Used for Random Forest

4. Training and Testing the Model

We have split the given dataset in *'Trainset.csv'* into two parts randomly into a training set and a test set with 80/20 %. Only the training set was used to feed into the model. Therefore, we were able to use the test as a completely independent data set to evaluate the performance of the model. Tuning the model was much easier using this method.
After getting a reasonably well test set accuracy and AUC value we predicted the outcomes on the given test data given in the *'xtest.csv'* file.

Results

These are the results that we got after training the model. Here Test Set is referred to the pre allocated 20% of the training data. Around 91% of accuracy was achieved for the Test Set, which was reflected by the score that we got in the competition.

```
Train Accuracy: 93.559%      AUC: 0.8395
Test Accuracy: 91.842%       AUC: 0.8073
Total Set Accuracy: 93.044%  AUC: 0.8304
Confusion Matrix for Test Set :
 [[1711   53]
 [ 118  214]]
Number of False Positives in Test Set : 53
Number of False Negatives in Test Set : 118
Number of Ones Predicted for Test Data : 227
```

Figure 3: Different Types of Metrics Computed for the Model

Conclusion

According to the observations taken from the initial analysis of dataset and experimenting several possible classification models we have discovered that the ***Random Forest Algorithm*** resulted in the highest AUC and accuracy.