Customer Classifier using Random Forest

By – Rageeni sah

Date – 30th Sept, 2019


Approach Followed
Exploratory Data Analysis
Data Cleaning
Feature Creation
Feature Selection
Model Building and Model Metrics Selection
Model Hyper-parameter tuning

Exploratory Data Analysis

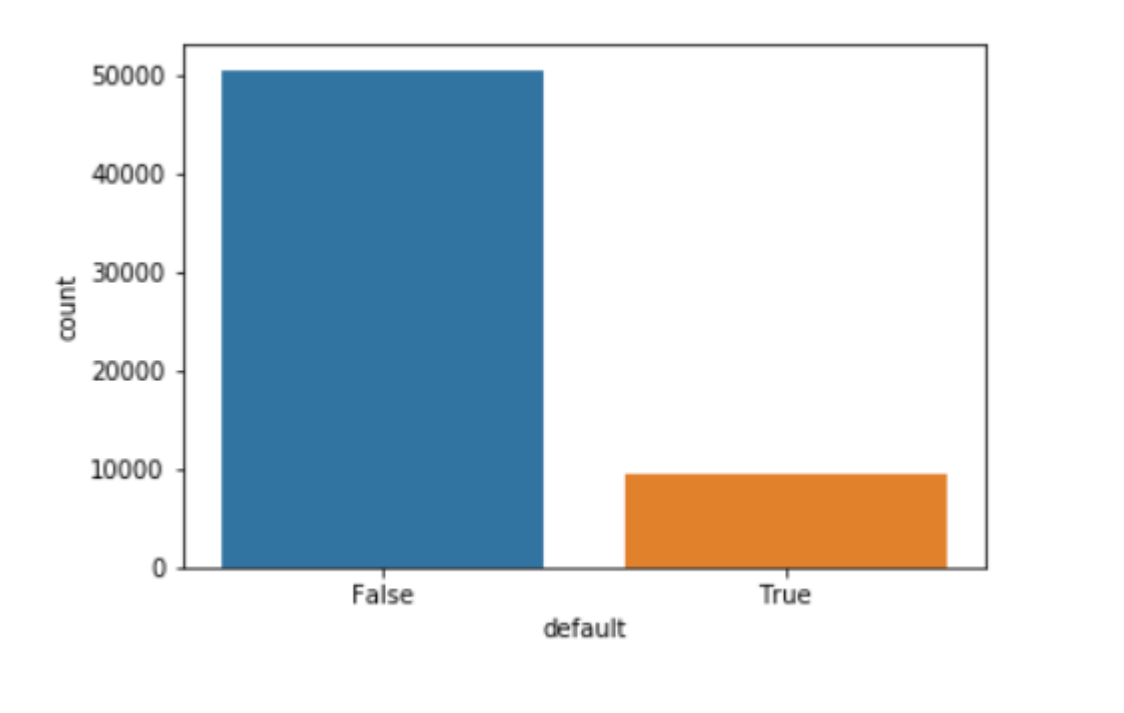
Key observations on train data

- Train Data has 64592 observations and 28 features and 1 target
- Test Data has 35000 observations with 28 features
- Target – ‘default’ (True/False) missing 4626 values
- Missing values in features
- Both numerical, categorical and date features

Exploratory Data Analysis

Key observations on train data

- +4626 missing values



Exploratory Data Analysis
Summary

- Numerical data
  - 'n_defaulted_loans' and 'income' are highly skewed
  - missing_values are filled with median_value of the column
- Categorical Data
  - Number of missing values gender: 2996
  - Number of missing values facebook_profile: 5971
  - Sign
    - virg, libr, cancer have large population
    - sagi, taur have relatively low population
    - All levels are retained for, all of them all default categories > 5 %
- 'score_1', 'reason', 'score_2', 'state','zip', 'channel', ‘job_name', 'real_state‘ dropped
- date_diff_bt_loanEnd_lastPayment - Added

Model Building, Hyper parameter Tuning, Evaluation

- RandomForestClassifier is used for model building for:
  - Feature Dimension is too high and would be more prone to over fitting
  - Random Forest is best selection for high variance
  - SMOTE is used for over-sampling minority class
- There are several parameters for RF model however, model performance is highly influenced by tuning hyper parameters
- Tuned only 'n_estimators' and 'max_depth' parameters while, retaining default values for other parameters
- Accuracy, precision and recall are prime metrics used
- Achieved Model generalization score is 80%
- Model performance can further be improved by tuning all hyper-parameters

Model Building, Hyper parameter Tuning, Evaluation

```
Est--> 150 Depth--> 10, Precision:0.347 --> Recall: 0.813 --> Accuracy: 0.728
Est--> 150 Depth--> 20, Precision:0.36 --> Recall: 0.789 --> Accuracy: 0.744
Est--> 150 Depth--> None, Precision:0.368 --> Recall: 0.785 --> Accuracy: 0.752
Est--> 500 Depth--> 10, Precision:0.348 --> Recall: 0.808 --> Accuracy: 0.73
Est--> 500 Depth--> 20, Precision:0.364 --> Recall: 0.789 --> Accuracy: 0.748
Est--> 500 Depth--> None, Precision:0.362 --> Recall: 0.788 --> Accuracy: 0.746
```

- '1' is positive class , '0' is negative class
  - Recall is very high for this model, while precision is low
  - Recall indicates fraction of true positive from actual positive
  - A low precision essentially means that the classifier returns a lot of false positives (0 is returned as 1)
- Considering precision - recall tradeoff, Low precision should be acceptable because, we cannot misclassify defaulters as non-defaulters, while it is OK to misclassify none defaulters as defaulters

Future Works

- **Model performance can further be improved by tuning following parameters:**
  - min_split_features
  - min_samples_leaf
  - max_features
  - nsamples
  - Outliers handling
  - Removing insignificant features
  - Feature Engineering
  - Using Advanced models

  I have not used them as of now for, tuning each one of them would require ample time to try out several parameter combinations.