Kaggle Competition: Santander Customer Satisfaction

Anonymous Author(s)
Affiliation
Address
email

Abstract

The aim of this project is to develop a machine learning model that helps identifying dissatisfied costumers of Santandar Bank at early stages. In this report we discuss the different stages of the project and describe the various approaches and algorithms we tried. The first section details data pre-processing which deals with redundant and missing data. The second section summarizes the different approaches and algorithms we tried to develop the model, which lead to choosing Extreme Gradient Boosting, xgboost algorithm. Tuning the parameters of this algorithm is detailed in the third section. Finally, the fourth section discusses feature engineering tricks we used to enhance the testing data AUC score.

1 Data Pre-Processing

This project is a worldwide competition presented by Kaggle and initiated by Santandar bank. The aim of this project is to develop a model to identify unsatisfied bank customers early in their relationship. Doing so will allow Santander to detect dissatisfied customers and take proactive steps at early stages to enhance customer's happiness. Real training and testing data were provided in two separate excel sheets: 'train.csv' and 'test.csv'. The training file holds 76020 instances, each having 370 features and accompanied with a label which is a binary variable of value 1 for unsatisfied customers and 0 for satisfied customers. The testing file holds 75818 instances each having 370 features. The training file should be used to learn the model which will be used later to classify the testing data.

The first stage in our project was studying the given data and removing redundancies. To study the data we installed t-SNE matab functions and tried to visualize the high dimensional data in a plane. The result is shown in Figure 1. We can easily see that the data is very complicated and thus conclude that linear models and uncomplicated decision boundaries are bad choices.

After visualizing the data, we searched for features that have a constant value over all instances, i.e. zero standard deviation. These features were removed because they are uncorrelated with the label and thus do not contribute to the target value. Moreover, we searched for equal columns among the data to remove redundancy. In addition to these operations, we also added a column holding the number of zeros in an instance. According to[1] this is one of the important features that improve the AUC score. Also, according to [1] PCA components are important features. For that reason we computed PCA with 4 components on the processed data, and added these 4 features to both training and testing data. The code used for this part is shown in Figure 2.

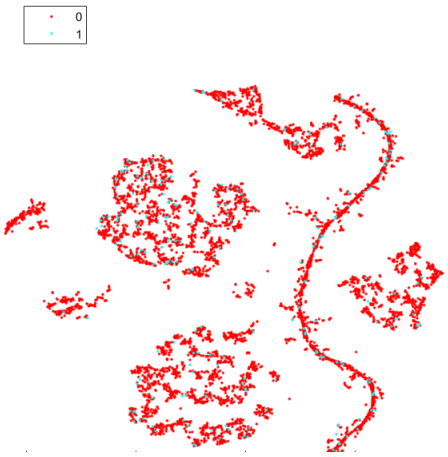

Figure 1: *t-SNE Visualization*

```
pca=PCA(n_components=4)
proc_xtrain=pca.fit_transform(normalize(train[features],axis=0))
proc_xtest=pca.transform(normalize(test[features],axis=0))

train.insert(1,'PCA1',proc_xtrain[:,0])
train.insert(1,'PCA2',proc_xtrain[:,1])
train.insert(1,'PCA3',proc_xtrain[:,2])
train.insert(1,'PCA4',proc_xtrain[:,3])

test.insert(1,'PCA1',proc_xtest[:,0])
test.insert(1,'PCA2',proc_xtest[:,1])
test.insert(1,'PCA3',proc_xtest[:,2])
test.insert(1,'PCA4',proc_xtest[:,3])
```

Figure 2: *PCA Component Code*

Another copy of the pre-processed data was taken, and a matlab code was used to choose a linear independent subset that span the feature space. However, this step did not result in any improvement and thus our final code did not use this copy of the data.

2 Tuning the Parameters

As previously mentioned, we chose to use *xgboost* to predict the label of the testing instances. For that purpose, XGBClassifier python function was used. This function requires various parameters which need to be tuned to get the best possible score. [2] describes a detailed process for tuning the xgboost parameters. GridSearch CV function from sklearn was used to search for the best parameters. An example on using this function is shown in the figure below:

```
param_test1 = {
 'max_depth':[3,5,7,9],
 'min_child_weight':[1,3,5]
}
print(param_test1)
print(type(param_test1))

gsearch1 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, n_estimators=140, max_depth=5,
 min_child_weight=1, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1, seed=27),
 param_grid = param_test1, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch1.fit(train_X,train_Y)
gsearch1.grid_scores_, gsearch1.best_params_, gsearch1.best_score_
```

Figure 3: *Grid Search for best Max depth and Min child Weight Parameters*

Note that this part of the code was run in a separate file to tune the parameters which will be used in our final solution.

3 Feature Engineering

After processing the given data and tuning the xgboost function we were able to get an AUC score of 0.839. To enhance our model, we further studied the given features by using the information provided by [3]. According to [3], var3 represents customer's nationality. Most of the customers are given 2 and rest are given various integers. It is perceived that 2 means the customer is native and other variables represent other nationalities. We replaced this feature by a binary variable indicating whether a customer is a foreigner or native.

$var38$ is one of the most important features in the dataset according to f-test and the importance sequence given by Random Forest and Extreme Gradient Boosted Trees. Since this feature has a skewed normal distribution, we used log transformation to normalize it.

$var36$ has 5 different integer values (0,1,2,3,99), which seems to be a categorical variable. Since Xgboost can't distinguish categorical variables, we transformed this feature into dummy variables consisting of 1's and 0's.

Also, for num_var5, num_var30, num_var35, num_var42, $num\_meses\_var5\_ult3$, and num_var4, we used dummy variables instead of using as integer values in Xgboost algorithm.

4 Other Algorithms Tried

4.1 Random Forest

Random Forest was one of the algorithms we tried to get the best AUC score. We used the transformed data in Random Forest algorithm. Even though we were able to get a good AUC for training data, the AUC for test data was not promising. So, we focused XGBoost algorithm.

```
forest = RandomForestClassifier(n_estimators=1000, criterion='entropy', min_samples_split=100,
                                min_samples_leaf=40, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,
                                bootstrap=True, oob_score=True, n_jobs=1, random_state=None, verbose=1)
```

```
train_pred = forest.predict_proba(train[features])
test_pred = forest.predict_proba(test[features])
```

```
forest = forest.fit(train[features],train.TARGET)
```

```
print('Average Log Loss:', log_loss(train.TARGET.values, train_pred[:,1]))
print('Average ROC:', roc_auc_score(train.TARGET.values, train_pred[:,1]))
print('Finish')
```

```
Average Log Loss: 0.130661896472
Average ROC: 0.872895122038
Finish
```

Figure 4: *Random Forest Algorithm*

4.2 Gradient Boosted Trees Algorithm

Before using XGBoost algorithm, we tried gradient boosted trees algorithm. Before using this algorithm, we had made a feature selection and selected 70 features among all using f-test.

After this, using GridSearchCV function in Python, we tried different tuning options and optimized all parameters. Even under best parameters we get, we couldn't get any Cross-Validation AUC Score over 84.0

Write about num-var4, var36 and var38 Please

References

[1] https://www.kaggle.com/cast42/santander-customer-satisfaction/xgboost-with-early-stopping/output

```
In [133]: gbm_tuned_3 = GradientBoostingClassifier(learning_rate=0.005, n_estimators=1600,max_depth=7,
                                                  min_samples_split=1200,min_samples_leaf=40, subsample=0.80,
                                                  random_state=10, max_features=8)
          modelfit(gbm_tuned_3, X3,y, X3.columns)

Model Report
Accuracy : 0.9606
AUC Score (Train): 0.860386
CV Score : Mean - 0.8359472 | Std - 0.0078459 | Min - 0.8273223 | Max - 0.8493435
```

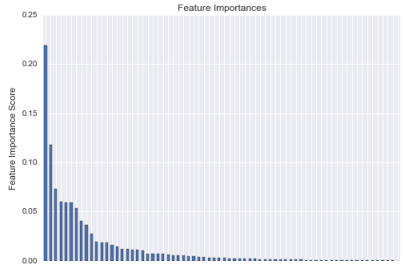

Figure 5: *Random Forest Algorithm*

[2] http://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[3] https://www.kaggle.com/c/santander-customer-satisfaction/forums/t/19291/data-dictionary