Capstone Project Report

Project Description

Which customers are happy customers?

This project is trying to solve a problem from Kaggle sponsored by Santander Bank.
https://www.kaggle.com/c/santander-customer-satisfaction
From frontline support teams to C-suites, customer satisfaction is a key measure of success. Unhappy customers don't stick around. What's more, unhappy customers rarely voice their dissatisfaction before leaving. Santander BanK is asking to help them identify dissatisfied customers early in their relationship. Doing so would allow Santander to take proactive steps to improve a customer's happiness before it's too late. In this competition, participants work with hundreds of anonymized features to predict if a customer is satisfied or dissatisfied with their bank experience.

Get The Data

The data can be downloaded from Kaggle site. The datasets are also included during the project submission.
https://www.kaggle.com/c/santander-customer-satisfaction/data
The data package includes

- train.csv - the training set including the target
- test.csv – the test set without the target
- sample_submission.csv – a sample submission file in the correct format

The anonymized dataset containing a large number of numeric variabls. The “TARGET ” column is the variable to predict. It equals 1 for unsatisfied customers and 0 for satisfied customers.

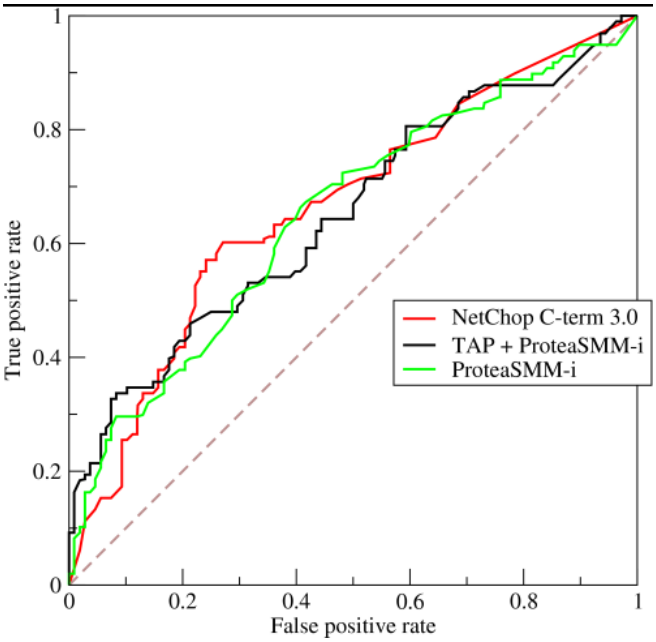
Performance Metrics

Area Under the ROC Curve is selected to evaluate the performance of model in this project. The task for this project is to predict the probability that each customer in the test set is an unsatisfied customer. The final submission.csv file should have the following format.

ID	TARGET
2	0.04663
5	0.062431
6	0.002968
7	0.003628
9	0.001046

Area Under the ROC Curve

ROC curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate against the false positive rate at various threshold settings. Find the sample ROC curve below from (https://en.wikipedia.org/wiki/Receiver_operating_characteristic#/media/File:Roccurves.png).



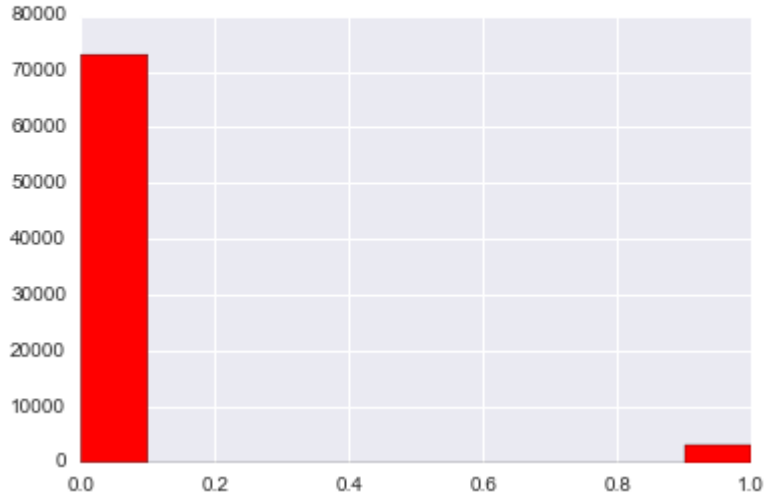
The AUC is a common evaluation metric for binary classification problems. If the classifier is very good, the true positive rate will increase quickly and the area under the ROC curve will be close to 1. If the classifier is no better than random guessing, the true positive rate will increase linearly with the false positive rate and the area under the curve will be around 0.5. One characteristic of the AUC is that it is independent of the fraction of the test population which is class 0 or class 1: this makes the AUC useful for evaluating the performance of classifiers on unbalanced data sets.

PRELIMINARY DATA EXPLORATION

1. Load both training data and testing data. We can see that the training data has 371 columns and the testing data has 370 columns, which means that we have to work on the feature Engineering to decrease the number of columns and find out the golden features to increase the accuracy of our model.

2. Below is the basic information of the data. Total number of training data set is 76020. Number of features is 369 (exclude "TARGET" and "ID"). The histogram below shows the number of satisfied customers vs. number of unsatisfied customers. We can see that the majority of the customers are satisfied customers. Only a small fraction of customers who are not satisfied.

```
number of training data:  76020
number of features:  369
number of satisfied customers:  73012
number of unsatisfied customers:  3008
```



3. Through the preliminary observation of the training dataset, we can find that some columns have constant values (i.e. value of 0). We can find out all these columns and remove them since their information gain is 0 if we use them as features to classify the data. They are not helpful with the classification. Find out all the columns with constant values.

```
#Boolean value to indicate if this is a column with constant value
removed_col = []

# (1)find out columns with constant values
for col in train.columns:
    first = train.ix[0,[col]].values
    if (train[col].values == first * np.ones(len(train)).astype(float)).all():
        removed_col.append(col)
```

We can see that there are 34 columns with constant value. We will remove all these columns from the datasets.

4. We can also check if there are any columns are duplicated. Find out all these duplicated columns and remove them would be helpful to decrease the number of features and lower the computing time.

```
# (2)Check if there are duplicated columns
removed_same_col = []
columns = train.columns[0:-1]
print len(columns)
for i in xrange(len(columns)-1):
    for j in range(i+1,len(columns)):
        if (train[columns[i]].values == train[columns[j]].values).all():
            removed_same_col.append(columns[i+1])

print "number of duplicated columns: ",len(removed_same_col)
print removed_same_col
```

We can see that there are 29 columns are duplicated columns. We will remove all these columns from the datasets.

5. Remove columns found in step 3 and 4 above.

```
#removed duplicaed columns
train.drop(removed_same_col,axis=1,inplace=True)
test.drop(removed_same_col,axis=1,inplace=True)
```

FEATURE SCALING

Display the statistics of the datasets. Please find part of the data description below.

```
print "Display train dataset stats"
display(train.describe())
```

```
Display train dataset stats
```

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3
count	76020.000000	76020.000000	76020.000000	76020.000000	76020.000000	76020.000000
mean	75964.050723	-1523.199277	33.212865	86.208265	72.363067	119.529632
std	43781.947379	39033.462364	12.956486	1614.757313	339.315831	546.266294
min	1.000000	-999999.000000	5.000000	0.000000	0.000000	0.000000
25%	38104.750000	2.000000	23.000000	0.000000	0.000000	0.000000
50%	76043.000000	2.000000	28.000000	0.000000	0.000000	0.000000
75%	113748.750000	2.000000	40.000000	0.000000	0.000000	0.000000
max	151838.000000	238.000000	105.000000	210000.000000	12888.030000	21024.810000

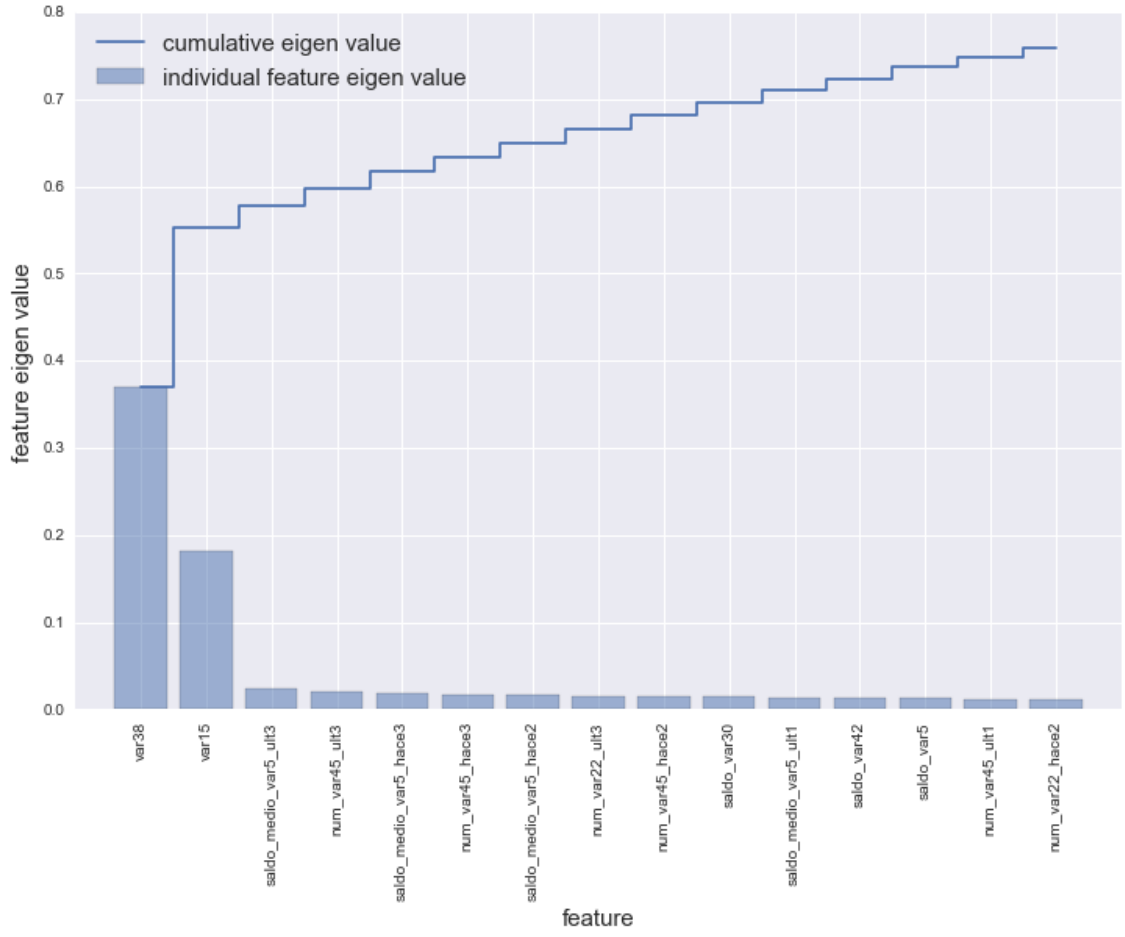
We can find that the mean and median values among all these features vary significantly, which means the data is not normally distributed and indicates a large skew. Perform the feature scaling would be helpful to reduce skewness.

The dataset is then normalized to range (0,1)

	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3
count	60816.000000	60816.000000	60816.000000	60816.000000	60816.000000
mean	0.998288	0.282163	0.000629	0.005623	0.005710
std	0.038432	0.129503	0.011134	0.026540	0.026304
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.999767	0.180000	0.000000	0.000000	0.000000
50%	0.999767	0.230000	0.000000	0.000000	0.000000
75%	0.999767	0.350000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

FEATURE SELECTION AND TRANSFORMATION

Perform a basic Random Forest Classifier to try to classify the dataset and find out the importance of features.



Random Forest Classifier provides a way to determine the importance of each feature. We can find that the most two important features are var38 and var15, which explain 0.370 and 0.182 variance respectively. Features, which ranks lower than 237th, has explain 0 variance. The first 50 features explain over 0.9 variance.

Sklearn SelectFromModel is used for feature selection. According to sklearn documentation, (http://scikit-learn.org/stable/modules/feature_selection.html) SelectFromModel is a meta-transformer that can be used along with any estimator that has a feature_importances_ attribute. If the feature importance value is below the provided threshold parameter, the features are considered unimportant and removed.

The transformed dataset now has only 40 columns.

```
print X_all_norm.shape
```

(76020L, 40L)

MODEL SELECTION

Different classifiers has been tested and fit with the dataset and I compare the performance of these classifiers. Following are the chosen classifiers.

- Decision Tree Classifier
- K-Nearest Neighbors
- Random Forest Classifier
- Logistic Regression
- Gradient Boosting Classifier
- Gaussian Naïve Bayes Classifier
- Adaboost Classifier
- Xtreme Gradient Boosting Classifier

All of these classifiers are untuned. They run on all 76020 training data and calculate AUC score each with 10 cross-validations. Please find the running result below.

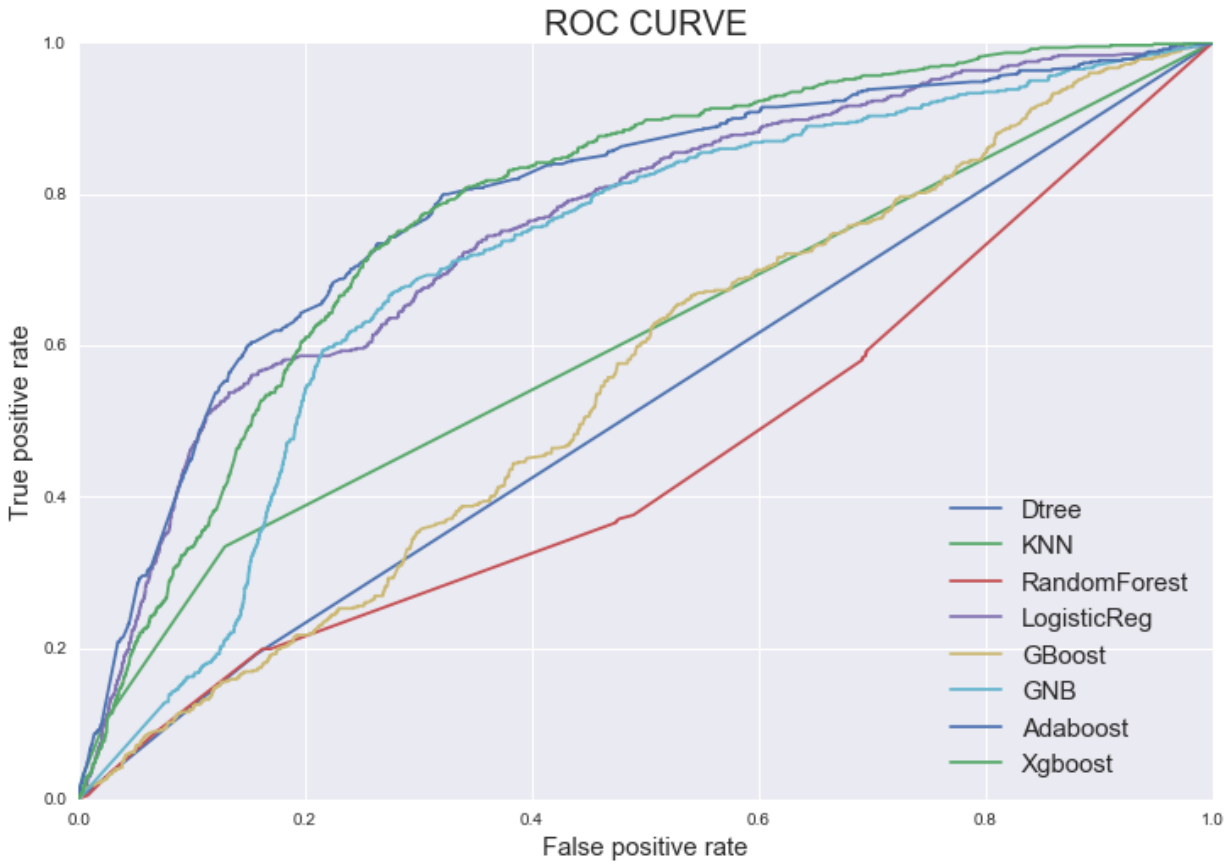
Classifier	Running Time (sec)	AUC Score (training set)
DecisionTreeClassifier	13.626999855	0.571675198697
KNeighborsClassifier	165.298000097	0.630522714608
RandomForestClassifier	14.6019999981	0.670856122994
LogisticRegression	9.41500020027	0.773513394802

GradientBoostingClassifier	354.829999924	0.835911097391
GaussianNB	1.35700011253	0.744050334439
AdaBoostClassifier	58.2100000381	0.825167602286
XGBClassifier	34.8019998074	0.838055048755

According to the result table above, running with 10 cross-validations, I can see that the top three classifiers with good AUC score are Xgboost Classifier, Gradient Boosting Classifier, and Adaboosting Classifier. Xgboost Classifier gets the best AUC score with 0.838.

Decision Tree, Random Forest and GaussianNB have low running time. However, the AUC score for these three classifiers are very low. XGBClassifier has a shorter running time compared to Adaboost Classifier and Gradient Boosting Classifier.

Below is the ROC curve for different model.



Also, from the ROC curve above, we can see that Decision Tree has the worse performance. Xgboost and Adaboost has the relatively good performance.

XGBClassifier is chosen as the classifier for further exploration. Below is the information for basic untuned XGBClassifier.

XGBClassifier(base_score=0.5,colsample_bylevel=1,colsample_bytree=1, gamma=0,learning_rate=0.1,max_delta_step=0,max_depth=3,min_child_weight=1,missing=None,n_estimators=100,nthread=-1,objective='binary:logistic',reg_alpha=0,reg_lambda=1,scale_pos_weight=1, seed=42, silent=False, subsample=1)

AUC score: 0.8380

MODEL TUNING

We need to tune the Xgboost parameters to get a better AUC score. First, I try to apply GridSearch along with cross-validation to find the optimal number of estimators which affects the boosting operation.

```
#Tune number of estimators
param = {'n_estimators':[50,100,150,200,250]}
{'n_estimators': 50} 0.832750002895
{'n_estimators': 100} 0.838055046569
{'n_estimators': 150} 0.838368578154
{'n_estimators': 200} 0.837948082382
{'n_estimators': 250} 0.837355615624
```

We can see that when the number of estimators is 150, the classifier gets the best AUC score of 0.838368

Second, I try to find the best combinations of tree-specific parameters, max_depth and min_child_weight, then find tune the parameters.

```
#Tune max_depth, and min_child_weight
param = {'max_depth':[3,4,5],'min_child_weight':[1,3,5,7,9]}
{'max_depth': 3, 'min_child_weight': 1} 0.838368578154
{'max_depth': 3, 'min_child_weight': 3} 0.838144708612
{'max_depth': 3, 'min_child_weight': 5} 0.83845610541
{'max_depth': 3, 'min_child_weight': 7} 0.838551279446
{'max_depth': 3, 'min_child_weight': 9} 0.83868808744
{'max_depth': 4, 'min_child_weight': 1} 0.838289876529
{'max_depth': 4, 'min_child_weight': 3} 0.838016759815
{'max_depth': 4, 'min_child_weight': 5} 0.837528407461
{'max_depth': 4, 'min_child_weight': 7} 0.838439720514
{'max_depth': 4, 'min_child_weight': 9} 0.838477202969
```

```
{'max_depth': 5, 'min_child_weight': 1} 0.837822254347
{'max_depth': 5, 'min_child_weight': 3} 0.837274965182
{'max_depth': 5, 'min_child_weight': 5} 0.8372595869
{'max_depth': 5, 'min_child_weight': 7} 0.837025308938
{'max_depth': 5, 'min_child_weight': 9} 0.837358376938
```

We can see the with max_depth: 3 and min_child_weight: 9, the classifier get the best AUC score of 0.838368.

Third, I set different values for the subsamples.

```
#Tune subsamples
param = {'subsample': [0.4,0.6,0.7,0.8,0.9,0.95,1]}
```

```
{'subsample': 0.4} 0.838380007171
{'subsample': 0.6} 0.838502326385
{'subsample': 0.7} 0.83929403456
{'subsample': 0.8} 0.838957571655
{'subsample': 0.9} 0.839052001807
{'subsample': 0.95} 0.83876246812
{'subsample': 1} 0.83868808744
```

When subsample parameter is set to 0.7, the AUC score is 0.83929

Last, I try different values for learning rate as follow.

```
#Tune learning rate
param = {'learning_rate':[0.005,0.01,0.02,0.055,0.07,0.08,0.09,0.095,0.1]}
```

```
{'learning_rate': 0.005} 0.818293659344
{'learning_rate': 0.01} 0.822200492782
{'learning_rate': 0.02} 0.827806264971
{'learning_rate': 0.055} 0.838716118865
{'learning_rate': 0.07} 0.838912777691
{'learning_rate': 0.08} 0.838958226387
{'learning_rate': 0.09} 0.8392048561
{'learning_rate': 0.095} 0.838952538045
{'learning_rate': 0.1} 0.83929403456
```

We can see that the best AUC score is still 0.83929 when learning rate is set to 0.1.

The final tuned XGBClassifier is shown as follow

XGBClassifier(base_score=0.5,colsample_bylevel=1,colsample_bytree=1,gamma=0,learning_rate=0.1,max_delta_step=0,max_depth=3,min_child_weight=9,missing=None,n_estimators=150,nthread=-1,objective='binary:logistic',reg_alpha=0,reg_lambda=1,scale_pos_weight=1,seed=42,silent=False,subsample=0.7)

The AUC score is improved from 0.8380 to 0.8392

ANALYZING MODEL PERFORMANCE

We need to check the model's learning and testing error rates on various subsets of training data. See the graph below of the model's performance.

From the graph above, we can see that as the number of data increases, the difference between training set's AUC score and testing set's AUC score is getting smaller, which indicates that the models's bias and variance are low. And it is probably not overfitted.

CONCLUSION

In this project, I try to solve the problem of dissatisfied customer prediction raised by Santander Bank on Kaggle with the datasets provided by Santander Bank. AUC is selected as the performance metrics. I work on the data cleaning, feature Engineering, model selection and model tuning to build a model which can predict the dissatisfied customer. The final tuned XGBClassifier can achieve a relatively good AUC score of 0.8392.

REFLECTION

The data provided by Santander Bank is anonymous, which kind of hard for us to understand what each columns means. And we cannot analyze the data based on any prior industry experience.
When do the feature scaling, some data seem to be outliers. However, since we don't really understand the data, we cannot just remove the data. We have to still include the data, which may affect the accuracy of the final model.

IMPROVEMENT

Even though I try my best to tune the Xgboost Classifier, the result doesn't improve too much. Future improvement would require fully understand the parameters in Xgboost Classifier, and find the parameters combination which could greatly improve the performance.