BATMAN

PROF. PIOTR J. GMYTRASIEWICZ

CS 514 – APPLIED ARTIFICIAL INTELLIGENCE – PROJECT V

04/29/2016

Santander Customer Satisfaction

Which customers are happy customers?

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 Kagglers 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, you'll work with hundreds of anonymized features to predict if a customer is satisfied or dissatisfied with their banking experience.

Model:

I have chosen Random Forest Classifier as my model. The following is the simple Random Forest Classifier implemented initially.

```
clf = RandomForestClassifier(n_estimators=100, max_depth=17,
random_state=1)
scores = cross_validation.cross_val_score(clf, X_train, y_train,
scoring='roc_auc', cv=5)
print(scores.mean())
clf.fit(X_train, y_train)
y_pred = clf.predict_proba(X_test)
```

The reason for choosing Random Forest Classifier was that the evaluated estimator performance for the above simple implementation was 0.821360825487, which is a good one to start with.

I have enhanced the Random Forest Classifier model with n_estimators=800, random_state=1301, n_jobs=-1, criterion='gini', class_weight='balanced', max_depth=10. The parameter class_weight is important. Especially in our case where the variable to predict, TARGET, is very unbalanced (skewed) in our sample. It is better to set class_weight = "balanced_subsample", to have a balanced TARGET for each tree. But if we want weaker trees, class_weight="balanced" must be enough.

```
rfc = RandomForestClassifier (n_estimators=800,
random_state=1301, n_jobs=-1, criterion='gini',
class_weight='balanced', max_depth=10)
```

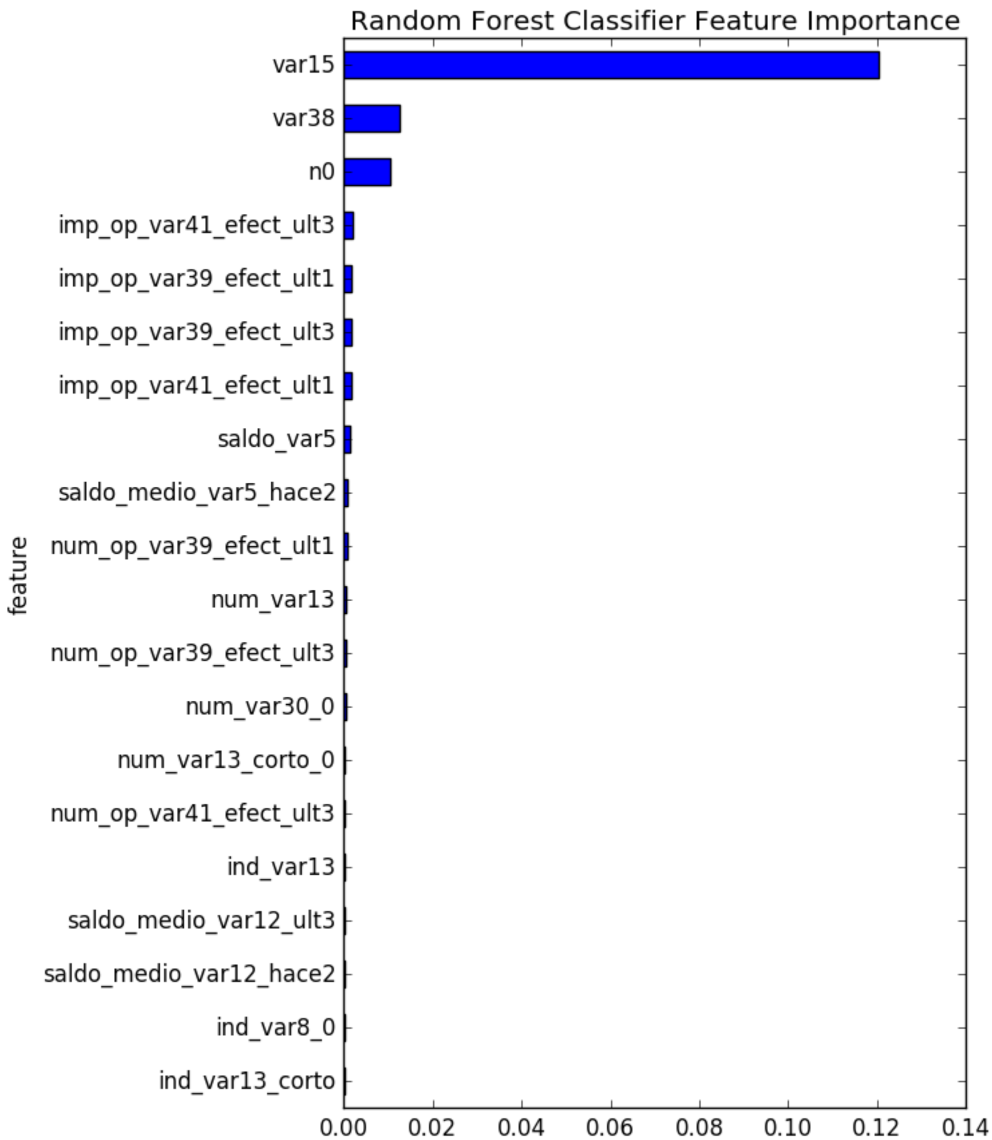
Cross validating the scores on a number of different random splits of the data has said to be an effective selection method for good features. The following link was very useful and gives more detailed explanation on it .

http://blog.datadive.net/selecting-good-features-part-iii-random-forests/

```
for train_idx, test_idx in cross_validation.StratifiedKFold(y, n_folds=5,
shuffle=True, random_state=1301):
        X_train, X_test = X_sel[train_idx], X_sel[test_idx]
        Y_train, Y_test = y[train_idx], y[test_idx]
         r = rfc.fit(X_train, Y_train)
         auc = roc_auc_score(Y_test, rfc.predict_proba(X_test)[:,1])
         for i in range(X_sel.shape[1]):
            X_t = X_test.copy()
             np.random.shuffle(X_t[:, i])
             shuff_auc = roc_auc_score(Y_test, rfc.predict_proba(X_t)[:,1])
                  scores[features[i]].append((auc-shuff_auc)/auc)
       print ("Features sorted by their score:")
print (sorted([(round(np.mean(score), 4), feat) for feat, score in scores.items()],
      reverse=True))
```

Hence we can plot the Feature Importance chart from this information.

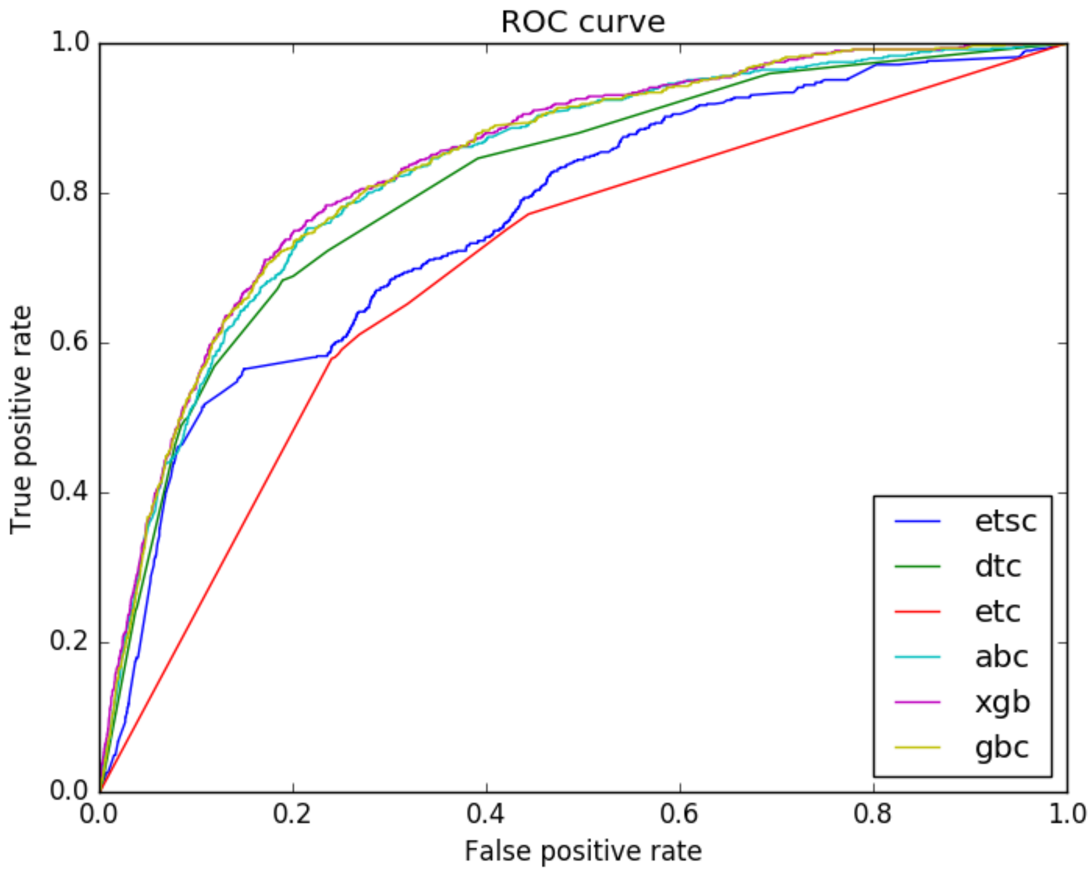
```
featp = ts.sort_values(by='score')[-20:].plot(kind='barh', x='feature', y='score',
        legend=False, figsize=(6, 10))
```


Random Forest Classifier Feature Importance
feature
var15
var38
n0
imp_op_var41_efect_ult3
imp_op_var39_efect_ult1
imp_op_var39_efect_ult3
imp_op_var41_efect_ult1
saldo_var5
saldo_medio_var5_hace2
num_op_var39_efect_ult1
num_var13
num_op_var39_efect_ult3
num_var30_0
num_var13_corto_0
num_op_var41_efect_ult3
ind_var13
saldo_medio_var12_ult3
saldo_medio_var12_hace2
ind_var8_0
ind_var13_corto
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14

ROC (Receiver operating characteristic) Curve:

The following snippet shows the plotting of ROC Curve.

```
plt.figure()
for name,clf in zip(names,clfs):
    clf.fit(X_train,y_train)
    y_proba = clf.predict_proba(X_test)[:,1]
    print("Roc AUC:"+name, roc_auc_score(y_test,
clf.predict_proba(X_test)[:,1],average='macro'))
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
plt.plot(fpr, tpr, label=name)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig('1.png')
plt.show()
```


ROC curve
True positive rate
False positive rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
etsc
dtc
etc
abc
xgb
gbc