Assignment 4 - Random Forest (Binary Classification)

Matthew Dunne

For assignment #4 we will be working with a credit default data set. The data includes various features around financial history and demographic information. The target variable is "default payment next week", which is just a binary flag of whether a customer defaults on a payment in the next week.

You will need to use the Random Forest Classifier from sklearn in order to build a classifier to predict if a customer is likely to default. You will also need to use the GridSearch CV for this assignment.

1. Data Processing

a) Import the data: The target / y variable is "default payment next month" column. Keep all predictors except for "ID."

In [7]:

```
import numpy as np
import os
import pandas as pd
data=pd.read_excel('default of credit card clients.xls')
```

b) Remove any rows that have missing data.

c) Split data into train / test set using an 70/30 split.

In [8]:

```
data=data.dropna(axis=0)
Y=data['default payment next month']
X=data.drop(['default payment next month'], axis=1)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.30, random_state=1234)
```

2. Random Forest Classifier - Base Model:

Start by creating a simple Random Forest only using default parameters.
a) Use the RandomForestClassifier in sklearn. Fit your model on the training data.

In [9]:

```
from sklearn.ensemble import RandomForestClassifier
forest_clf = RandomForestClassifier(random_state=1007)
rf_model=forest_clf.fit(X_train, y_train)
```

b) Use the fitted model to predict on test data. Use the .predict_proba() and the .predict() methods to get predicted probabilities as well as predicted classes.

In [19]:

```
#the class predictions
pred_class=rf_model.predict(X_test)
#the probability predictions
pred_prob=rf_model.predict_proba(X_test)
#the probability predictions return a n x 2 array. First column = prob. of 0. Second column = prob
. of 1 (default next week).
#for the roc_auc_score in step d you can only use one column - the prob. of being 1. So just subse
t to that
prob_of_one=pred_prob[:,1]
```

c) Calculate the confusion matrix and classification report (both are in sklearn.metrics). These are the same tools from HW #3.

In [11]:

```
from sklearn.metrics import confusion_matrix, classification_report
print('Confusion Matrix\n')
print(confusion_matrix(pred_class, y_test))
print('\nClassification Report\n')
print(classification_report(pred_class, y_test))
```

```
Confusion Matrix

[[6564 1340]
 [ 423  673]]

Classification Report

             precision    recall  f1-score   support

          0       0.94      0.83      0.88      7904
          1       0.33      0.61      0.43      1096

avg / total       0.87      0.80      0.83      9000
```

d) Calculate the roc_auc_score for this model. There are many ways to do this, but an example is to use the probabilities from step B and utilize the roc_auc_score from sklearn.

In [12]:

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, prob_of_one)
```

Out[12]:

```
0.73075872721115531
```

3. Random Forest Classifier - Grid Search:

Start by creating a simple Random Forest only using default parameters.
a) Use the RandomForestClassifier along with the GridSearchCV tool. Run the GridSearchCV using the following:
• n_estimators: 50, 750, 1000, 2500
• max_features: 2, 4, 6, 8
Note: Feel free to try out more parameters, the above is the bare minimum for this assignment.
Use 5 cross-fold and for scoring use "roc_auc" (this is the score that will be referenced when identifying the best parameters).

In [13]:

```
from sklearn.model_selection import GridSearchCV
#create of dictionary of parameters as applicable to the underlying random forest model
param_grid={'n_estimators':[50, 750, 1000, 2500], 'max_features':[2,4,6, 8], 'random_state':[1320]}
# create Random Forest model, and random state
rf_obj=RandomForestClassifier()
# Create gridsearch object with various combinations of parameters, n_jobs=-1 means use all proces
sors
rf_Grid = GridSearchCV(rf_obj, param_grid, cv = 5, scoring = 'roc_auc',refit = True, n_jobs=-1, ver
bose = 5)
rf_Grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
```

```
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    3.8s
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed: 10.4min
[Parallel(n_jobs=-1)]: Done  80 out of  80 | elapsed: 18.5min finished
```

Out[13]:

```
GridSearchCV(cv=5, error_score='raise',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
       fit_params=None, iid=True, n_jobs=-1,
       param_grid={'n_estimators': [50, 750, 1000, 2500], 'max_features': [2, 4, 6, 8],
'random_state': [1320]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=5)
```

b) Identify the best performing model:

In [14]:

```
#save the best model to another alias
best_rf=rf_Grid.best_estimator_
#see what those best parameters were
print(rf_Grid.best_params_)
```

```
{'max_features': 2, 'n_estimators': 2500, 'random_state': 1320}
```

c) Use the best estimator model to predict on test data. Use the .predict_proba() and the .predict() methods to get predicted probabilities as well as predicted classes.

In [15]:

```
#make the classification predictions
best_rf_pred_class=best_rf.predict(X_test)
#get just the probabilities of being 1
best_rf_pred_prob=best_rf.predict_proba(X_test)[:,1]
```

d) Calculate the confusion matrix and classification report (both are in sklearn.metrics).

In [16]:

```
print('Confusion Matrix\n')
print(confusion_matrix(best_rf_pred_class, y_test))
print('\nClassification Report\n')
print(classification_report(best_rf_pred_class, y_test))
```

```
Confusion Matrix

[[6604 1251]
 [ 383  762]]

Classification Report

             precision    recall  f1-score   support

          0       0.95      0.84      0.89      7855
          1       0.38      0.67      0.48      1145

avg / total       0.87      0.82      0.84      9000
```

e) Calculate the roc_auc_score for this model. This is where you use the predicted probabilities.

In [17]:

```
roc_auc_score(y_test, best_rf_pred_prob)
```

Out[17]:

```
0.7732920502208664
```

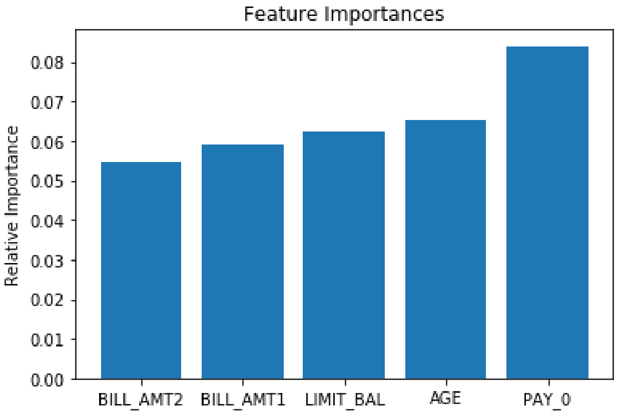
4. What are the best parameters from the Grid Search? Does the Model from #3 outperform Model #2?

The best parameters from the Grid Search are max features=2 and number of trees=2500. Interestingly Model #3 only slightly outperforms Model #2.

5. Create a feature importance plot for your best performing model. What are the top 5 features for this model?

In [18]:

```
import matplotlib.pyplot as plt
features=X.columns
importances=best_rf.feature_importances_
#get the indices of the importances, ordered by the underlying value, take top 5 (np.argsort goes
by ascending to go backwards from end)
indices = np.argsort(importances)[-5:]
plt.title('Feature Importances')
plt.bar(range(len(indices)), importances[indices], align='center')
plt.xticks(range(len(indices)), features[indices])
plt.ylabel('Relative Importance')
plt.show()
```



The top 5 most important features are PAY_0 (most recent repayment status), Age, LIMIT_BAL (amount of credit given), BILL_AMT1 (most recent billing statement), BILL_AMT2 (second most recent billing statement).