# Home Credit default prediction - Adaboost Model

In [1]:

```
import datetime
print(datetime.datetime.now())
```

```
2020-02-26 17:27:27.239467
```

In [2]:

```
import sklearn
print('The scikit-learn version is {}.'.format(sklearn.__version__))
```

```
The scikit-learn version is 0.22.
```

In [3]:

```
import pandas as pd
import numpy as np

import matplotlib as mpl
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.model_selection import GridSearchCV
```

In [4]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
import matplotlib as mpl
import seaborn as sns
```

## read training and testing datasets

In [5]:

```
X_train_final = pd.read_csv(r"C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\T
eam Assignments\Team Project\X_train_redo.csv",sep=',')
#X_val_final = pd.read_csv(r"C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\Te
am Assignments\Team Project\X_test_final.csv",sep=',')
X_test_final = pd.read_csv(r"C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\Te
am Assignments\Team Project\X_test_redo.csv",sep=',')
y_train = np.loadtxt('y_train_redo.txt', dtype=int)
y_test = np.loadtxt('y_test_redo.txt', dtype=int)
#y_val = np.loadtxt('y_val.txt', dtype=int)
```

```
In [6]: X_train_final.shape
Out[6]: (215257, 126)
```

```
In [7]: X_test_final.shape
Out[7]: (92254, 126)
```

```
In [8]: y_train.shape
Out[8]: (215257,)
```

```
In [9]: y_test.shape
Out[9]: (92254,)
```

## Upsampling using SMOTE

```
In [10]: print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1
         )))
         print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train ==
         0)))

         # import SMOTE module from imblearn library
         # pip install imblearn (if you don't have imblearn in your system)
         from imblearn.over_sampling import SMOTE
         sm = SMOTE(random_state = 2)
         X_train_up, y_train_up = sm.fit_sample(X_train_final, y_train.ravel())

         print('After OverSampling, the shape of train_X: {}'.format(X_train_up.shape))
         print('After OverSampling, the shape of train_y: {} \n'.format(y_train_up.shap
         e))

         print("After OverSampling, counts of label '1': {}".format(sum(y_train_up == 1
         )))
         print("After OverSampling, counts of label '0': {}".format(sum(y_train_up == 0
         )))
```

```
Before OverSampling, counts of label '1': 17377
Before OverSampling, counts of label '0': 197880


Using TensorFlow backend.

After OverSampling, the shape of train_X: (395760, 126)
After OverSampling, the shape of train_y: (395760,)

After OverSampling, counts of label '1': 197880
After OverSampling, counts of label '0': 197880
```

## Down sampling using Near Miss

In [11]:

```
print("Before Undersampling, counts of label '1': {}".format(sum(y_train == 1
)))
print("Before Undersampling, counts of label '0': {} \n".format(sum(y_train ==
0)))

# apply near miss
from imblearn.under_sampling import NearMiss
nr = NearMiss()

X_train_down, y_train_down = nr.fit_sample(X_train_final, y_train.ravel())

print('After Undersampling, the shape of train_X: {}'.format(X_train_down.shap
e))
print('After Undersampling, the shape of train_y: {} \n'.format(y_train_down.s
hape))

print("After Undersampling, counts of label '1': {}".format(sum(y_train_down =
= 1)))
print("After Undersampling, counts of label '0': {}".format(sum(y_train_down =
= 0)))
```

```
Before Undersampling, counts of label '1': 17377
Before Undersampling, counts of label '0': 197880

After Undersampling, the shape of train_X: (34754, 126)
After Undersampling, the shape of train_y: (34754,)

After Undersampling, counts of label '1': 17377
After Undersampling, counts of label '0': 17377
```

## Plot ROC_AUC_Curve

In [12]:
```
def plot_roc(clf, X_test_final, y_test, name, ax, show_thresholds=False):
    y_pred_ada = clf.predict_proba(X_test_final)[:, 1]
    fpr, tpr, thr = roc_curve(y_test, y_pred_ada)

    #ax.plot([0, 1], [0, 1], 'k--');
    ax.plot([0, 1], [0, 1]);
    ax.plot(fpr, tpr, label='{}, AUC={:.5f}'.format(name, auc(fpr, tpr)));
    #ax.scatter(fpr, tpr,marker='*');

    if show_thresholds:
        for i, th in enumerate(thr):
            ax.text(x=fpr[i], y=tpr[i], s="{:.2f}".format(th), fontsize=9,
                     horizontalalignment='left', verticalalignment='top', colo
r='black',
                     bbox=dict(facecolor='white', edgecolor='black', boxstyle=
'round,pad=0.1', alpha=0.1));

    ax.set_xlabel('False positive rate', fontsize=18);
    ax.set_ylabel('True positive rate', fontsize=18);
    ax.tick_params(axis='both', which='major', labelsize=18);
    ax.grid(True);
    ax.set_title('ROC Curve', fontsize=18)
```

# ADABOOST STARTS HERE

## Adaboost with no tuning

In [13]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
```

In [14]:
```
# Create adaboost classifer object
abc = AdaBoostClassifier()
# Train Adaboost Classifer
model = abc.fit(X_train_final, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test_final)
```

In [15]:

```
print("Accuracy = {:.3f}".format(accuracy_score(y_test, y_pred)))
print("F1 Score = {:.3f}".format(f1_score(y_test, y_pred)))
auc_abc = roc_auc_score(y_test, y_pred)
print("ROC AUC : %5.5f" %(auc_abc))

ABC_confusion_matrix1 = metrics.confusion_matrix(y_test,y_pred)
ABC_confusion_matrix1
```

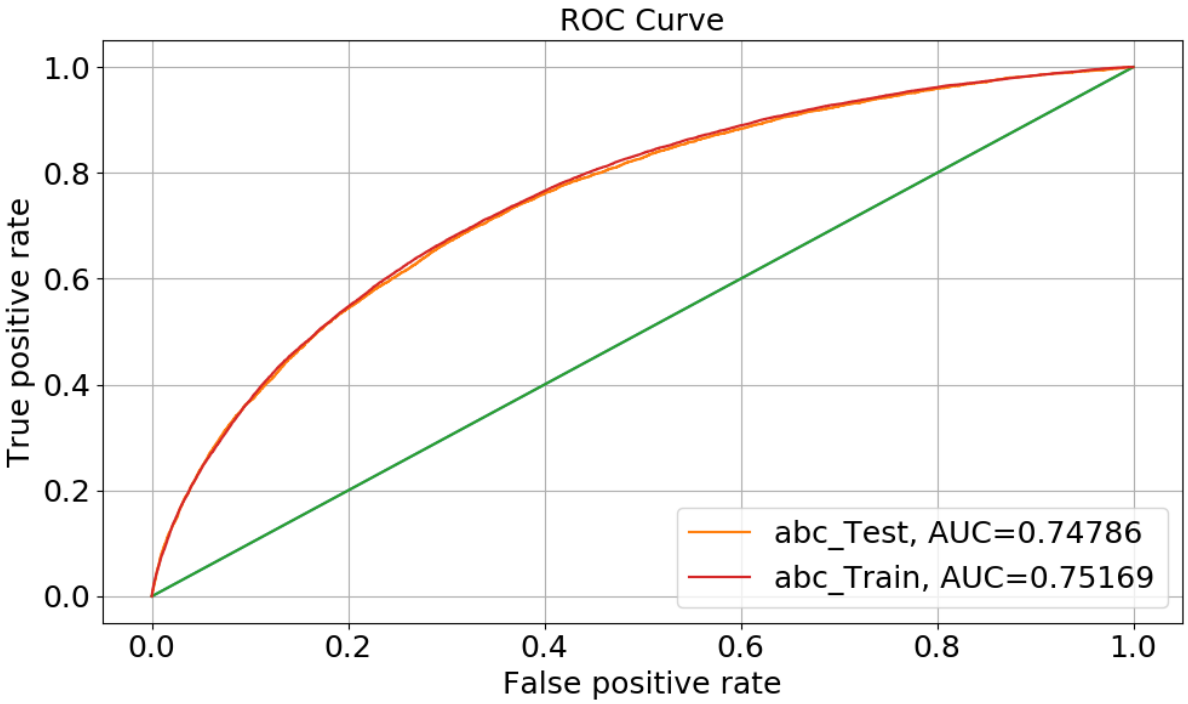
```
Accuracy = 0.919
F1 Score = 0.033
ROC AUC : 0.50763
```

Out[15]:

```
array([[84654,   152],
       [ 7321,   127]], dtype=int64)
```

In [16]:

```
plt.style.use('default');
figure = plt.figure(figsize=(10, 6));
ax4 = plt.subplot(1, 1, 1);
plot_roc(abc, X_test_final, y_test, "abc_Test", ax4)
plot_roc(abc, X_train_final, y_train, "abc_Train", ax4)
plt.legend(loc='lower right', fontsize=18);
plt.tight_layout();
```



## Adaboost with tuning

In [17]:
```
param_dist = {
 'n_estimators': [50, 100],
 'learning_rate' : [0.01,0.05,0.1,0.3,1],
 }

abc2 = GridSearchCV(AdaBoostClassifier(),
 param_grid = param_dist,
 cv=3,
scoring='roc_auc')

abc2.fit(X_train_final, y_train)
y_pred_tuned = abc2.predict(X_test_final)

print("Best Hyper Parameters:\n",abc2.best_estimator_)
```

```
Best Hyper Parameters:
 AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=
1,
                   n_estimators=100, random_state=None)
```

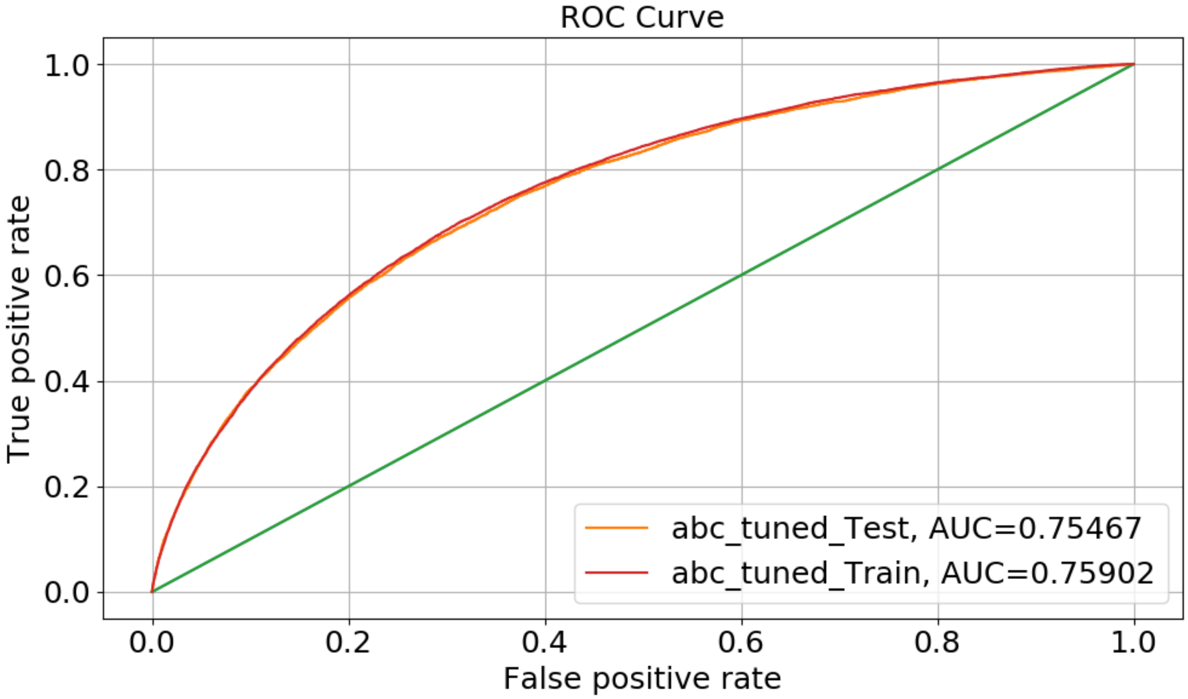
In [18]:
```
print("Accuracy = {:.3f}".format(accuracy_score(y_test, y_pred_tuned)))
print("F1 Score = {:.3f}".format(f1_score(y_test, y_pred_tuned)))
auc_abc2 = roc_auc_score(y_test, y_pred_tuned)
print("ROC AUC : %5.5f" %(auc_abc2))

ABC_confusion_matrix2 = metrics.confusion_matrix(y_test,y_pred_tuned)
ABC_confusion_matrix2
```

```
Accuracy = 0.919
F1 Score = 0.050
ROC AUC : 0.51196
```

Out[18]:
```
array([[84602,   204],
       [ 7252,   196]], dtype=int64)
```

In [19]:

```
plt.style.use('default');
figure = plt.figure(figsize=(10, 6));
ax4 = plt.subplot(1, 1, 1);
plot_roc(abc2, X_test_final, y_test, "abc_tuned_Test", ax4)
plot_roc(abc2, X_train_final, y_train, "abc_tuned_Train", ax4)
plt.legend(loc='lower right', fontsize=18);
plt.tight_layout();
```



## Adaboost upsampled without tuning

In [20]:

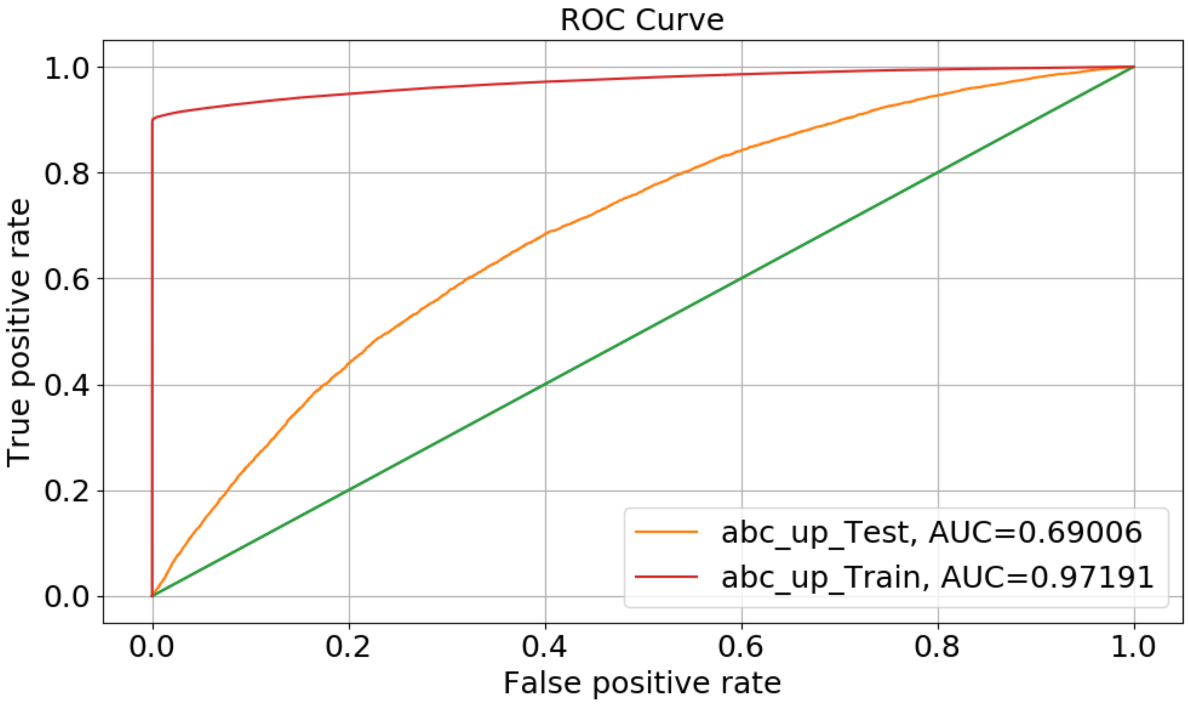
```
# Create adaboost classifer object
abc3 = AdaBoostClassifier()
# Train Adaboost Classifer
model = abc3.fit(X_train_up, y_train_up)

#Predict the response for test dataset
y_pred_up = model.predict(X_test_final)
```

In [21]:
```
print("Accuracy = {:.3f}".format(accuracy_score(y_test, y_pred_up)))
print("F1 Score = {:.3f}".format(f1_score(y_test, y_pred_up)))
auc_abc3 = roc_auc_score(y_test, y_pred_up)
print("ROC AUC : %5.5f" %(auc_abc3))

ABC_confusion_matrix3 = metrics.confusion_matrix(y_test,y_pred_up)
ABC_confusion_matrix3
```

```
Accuracy = 0.917
F1 Score = 0.016
ROC AUC : 0.50253
```

Out[21]:
```
array([[84541,   265],
       [ 7387,    61]], dtype=int64)
```

In [22]:
```
plt.style.use('default');
figure = plt.figure(figsize=(10, 6));
ax4 = plt.subplot(1, 1, 1);
plot_roc(abc3, X_test_final, y_test, "abc_up_Test", ax4)
plot_roc(abc3, X_train_up, y_train_up, "abc_up_Train", ax4)
plt.legend(loc='lower right', fontsize=18);
plt.tight_layout();
```



## Adaboost upsampled with tuning

In [23]:

```
param_dist = {
 'n_estimators': [50, 100],
 'learning_rate' : [0.01,0.05,0.1,0.3,1],
 }

abc4 = GridSearchCV(AdaBoostClassifier(),
 param_grid = param_dist,
 cv=3,
scoring='roc_auc')

abc4.fit(X_train_up, y_train_up)

y_pred_tuned_up = abc4.predict(X_test_final)
print("Best Hyper Parameters:\n",abc4.best_estimator_)
```

```
Best Hyper Parameters:
 AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=
0.3,
                   n_estimators=100, random_state=None)
```

In [24]:

```
print("Accuracy = {:.3f}".format(accuracy_score(y_test, y_pred_tuned_up)))
print("F1 Score = {:.3f}".format(f1_score(y_test, y_pred_tuned_up)))
auc_abc4 = roc_auc_score(y_test, y_pred_tuned_up)
print("ROC AUC : %5.5f" %(auc_abc4))

ABC_confusion_matrix4 = metrics.confusion_matrix(y_test,y_pred_tuned_up)
ABC_confusion_matrix4
```

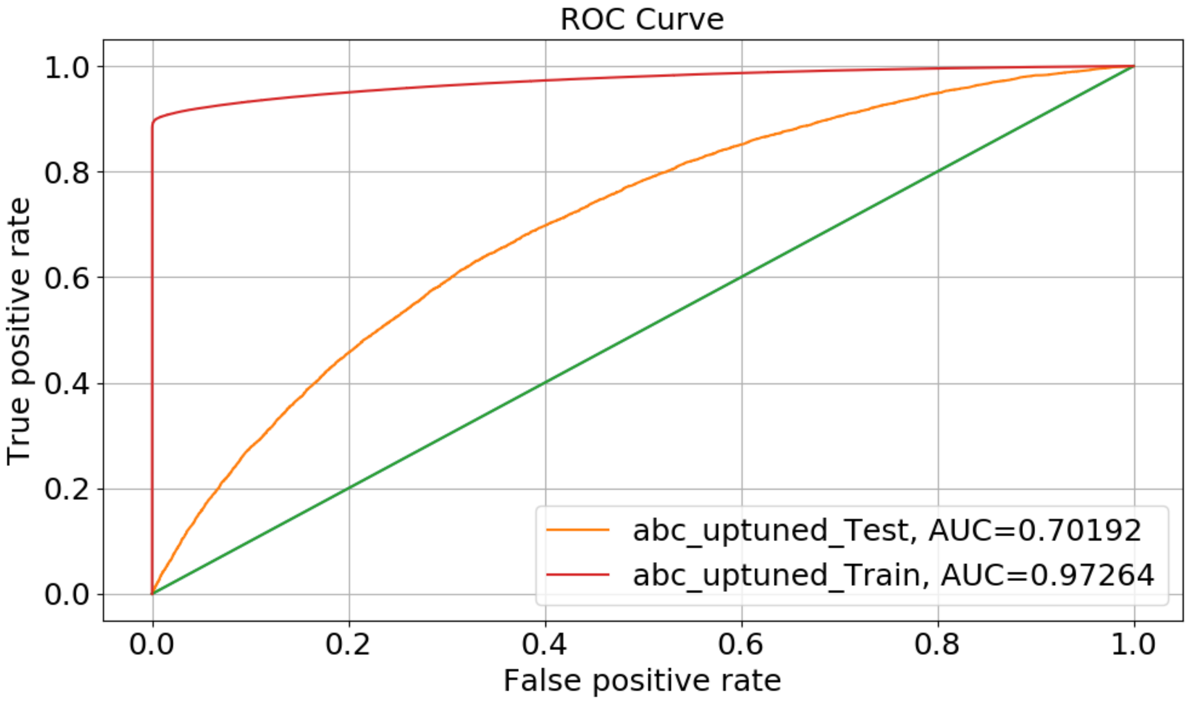
```
Accuracy = 0.918
F1 Score = 0.019
ROC AUC : 0.50375
```

Out[24]:

```
array([[84611,   195],
       [ 7375,    73]], dtype=int64)
```

In [25]:

```
plt.style.use('default');
figure = plt.figure(figsize=(10, 6));
ax4 = plt.subplot(1, 1, 1);
plot_roc(abc4, X_test_final, y_test, "abc_uptuned_Test", ax4)
plot_roc(abc4, X_train_up, y_train_up, "abc_uptuned_Train", ax4)
plt.legend(loc='lower right', fontsize=18);
plt.tight_layout();
```



# Adaboost downsample no tuning

In [26]:

```
# Create adaboost classifer object
abc_down = AdaBoostClassifier()
# Train Adaboost Classifer
abc_down.fit(X_train_down, y_train_down)

#Predict the response for test dataset
y_pred_down = abc_down.predict(X_test_final)
```

In [27]:

```
print("Accuracy = {:.3f}".format(accuracy_score(y_test, y_pred_down)))
print("F1 Score = {:.3f}".format(f1_score(y_test, y_pred_down)))
auc_abc_down = roc_auc_score(y_test, y_pred_down)
print("ROC AUC : %5.5f" %(auc_abc_down))

ABC_confusion_matrix5 = metrics.confusion_matrix(y_test,y_pred_down)
ABC_confusion_matrix5
```

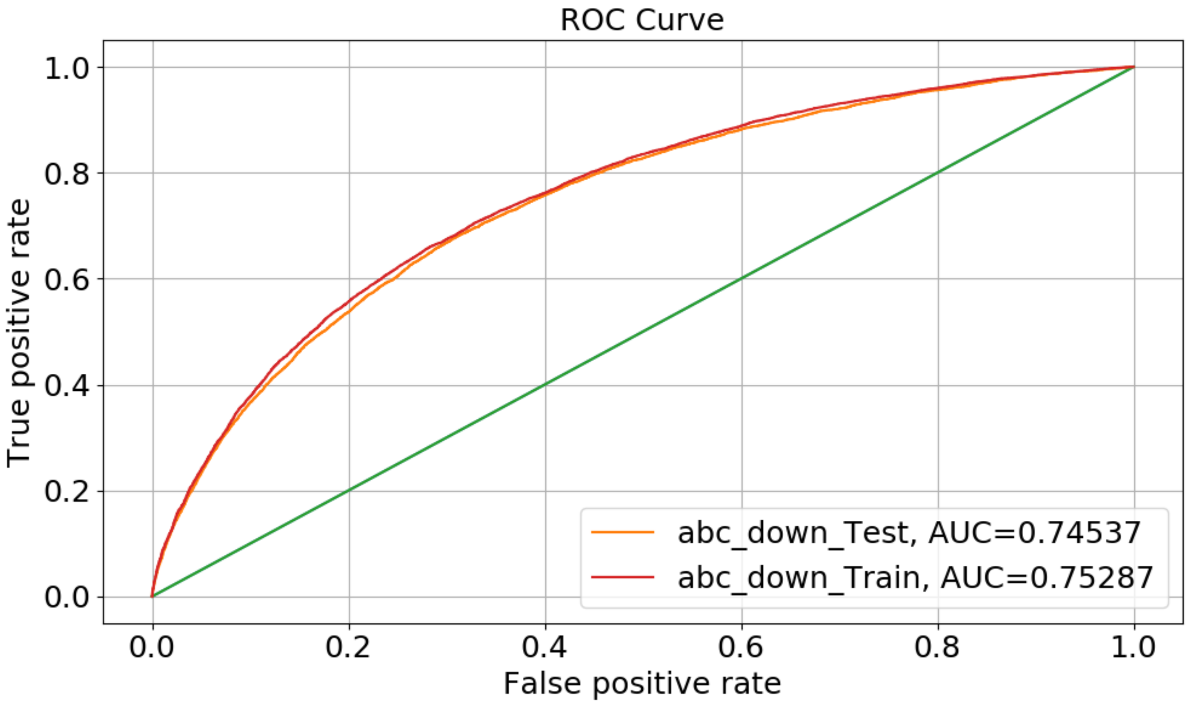
```
Accuracy = 0.690
F1 Score = 0.261
ROC AUC : 0.68428
```

Out[27]:

```
array([[58572, 26234],
       [ 2399,  5049]], dtype=int64)
```

In [28]:

```
plt.style.use('default');
figure = plt.figure(figsize=(10, 6));
ax4 = plt.subplot(1, 1, 1);
plot_roc(abc_down, X_test_final, y_test, "abc_down_Test", ax4)
plot_roc(abc_down, X_train_down, y_train_down, "abc_down_Train", ax4)
plt.legend(loc='lower right', fontsize=18);
plt.tight_layout();
```



## Adaboost downsampled with tuning

In [29]:

```
param_dist = {
 'n_estimators': [50, 100],
 'learning_rate' : [0.01,0.05,0.1,0.3,1],
 }

abc_tuned_down = GridSearchCV(AdaBoostClassifier(),
 param_grid = param_dist,
 cv=3,
scoring='roc_auc')

abc_tuned_down.fit(X_train_down, y_train_down)

y_pred_tuned_down = abc_tuned_down.predict(X_test_final)

print("Best Hyper Parameters:\n",abc_tuned_down.best_estimator_)
```

```
Best Hyper Parameters:
 AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=
0.3,
                   n_estimators=100, random_state=None)
```

In [30]:

```
print("Accuracy = {:.3f}".format(accuracy_score(y_test, y_pred_tuned_down)))
print("F1 Score = {:.3f}".format(f1_score(y_test, y_pred_tuned_down)))
auc_abc_tuned_down = roc_auc_score(y_test, y_pred_tuned_down)
print("ROC AUC : %5.5f" %(auc_abc_tuned_down))

ABC_confusion_matrix6 = metrics.confusion_matrix(y_test,y_pred_tuned_down)
ABC_confusion_matrix6
```

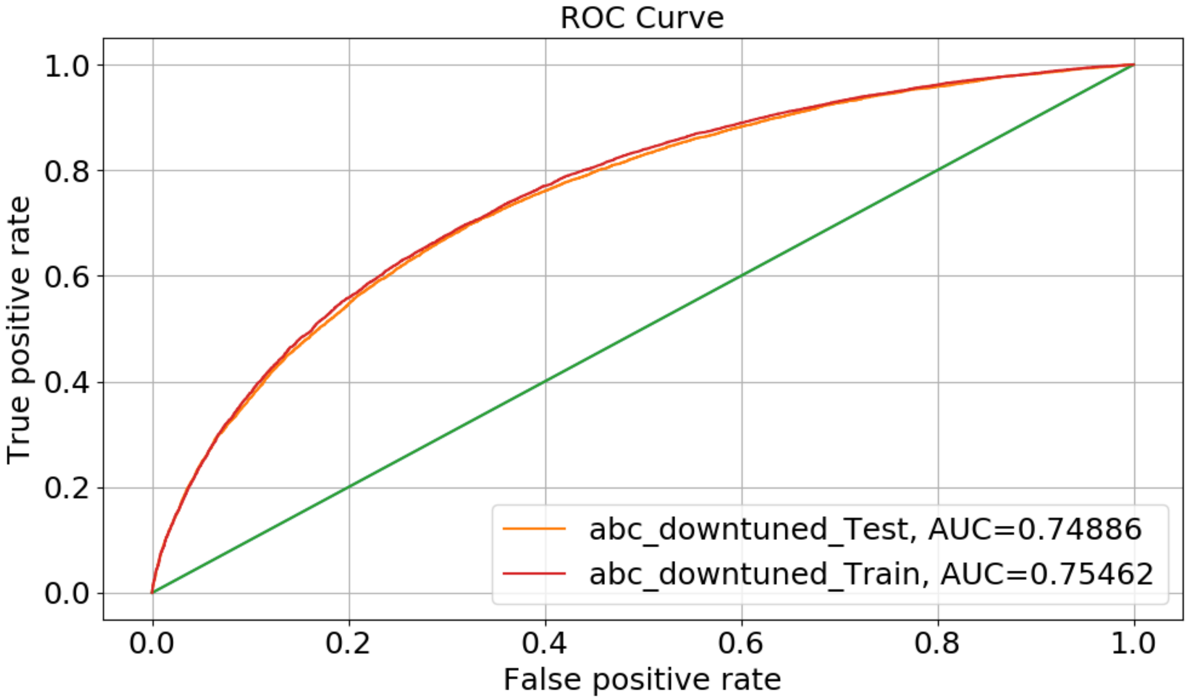
```
Accuracy = 0.689
F1 Score = 0.261
ROC AUC : 0.68565
```

Out[30]:

```
array([[58451, 26355],
       [ 2368,  5080]], dtype=int64)
```

In [31]:

```
plt.style.use('default');
figure = plt.figure(figsize=(10, 6));
ax4 = plt.subplot(1, 1, 1);
plot_roc(abc_tuned_down, X_test_final, y_test, "abc_downtuned_Test", ax4)
plot_roc(abc_tuned_down, X_train_down, y_train_down, "abc_downtuned_Train", ax
4)
plt.legend(loc='lower right', fontsize=18);
plt.tight_layout();
```


ROC Curve
True positive rate
False positive rate
abc_downtuned_Test, AUC=0.74886
abc_downtuned_Train, AUC=0.75462