Lending Club And Prosper Without 2007-2009

July 7, 2016

In [1]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as stats
from scipy.interpolate import interp1d

import matplotlib.pyplot as plt
%matplotlib inline

from sklearn import ensemble, cross_validation, preprocessing
```

```
/home/user/anaconda2/lib/python2.7/site-packages/matplotlib/font_manager.py:273: UserWarning: Matplotlib
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.')
```

In [2]:
```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

1 means the loan was paid. 0 means the loan was defaulted.

In [3]:
```
df1 = pd.read_csv('Lending Club Without 2007-2009.csv')
df2 = pd.read_csv('Prosper Loan Without 2007-2009.csv')

frame = [df1,df2]
df = pd.concat(frame)
df.columns = ['Filler', 'Loan Amount', 'Employment Length', 'Annual Income', 'Debt-To-Income Ra
              'Decision']
df = df.drop(['Filler'],1)
df.describe()
```

Out[3]:
```
       Loan Amount  Employment Length   Annual Income  Debt-To-Income Ratio  \
count  6402875.000        6402875.000     6402875.000           6402875.000
mean     13049.344              2.252      331223.309                 4.457
std      10256.161              2.878     15805890.497               206.817
min          0.000              0.000           0.000                 0.000
25%       5000.000              1.000        7896.000                 0.112
50%      10000.000              1.000       28884.000                 0.238
75%      20000.000              1.000       68000.000                 0.466
max    1400000.000             62.917 13226400000.000            500000.315

          Decision
count 6402875.000
mean        0.150
std         0.357
min         0.000
25%         0.000
```

```
50%          0.000
75%          0.000
max          1.000
```

In [4]:
```
print np.isnan(df).any()
print ''
print np.isinf(df).any()
```

```
Loan Amount              False
Employment Length        False
Annual Income            False
Debt-To-Income Ratio     False
Decision                 False
dtype: bool

Loan Amount              False
Employment Length        False
Annual Income            False
Debt-To-Income Ratio     False
Decision                 False
dtype: bool
```

In [5]:
```
print df[df['Annual Income'] < 1.].count()
print ''

df = df[df['Annual Income'] > 1.]
```

```
Loan Amount              8
Employment Length        8
Annual Income            8
Debt-To-Income Ratio     8
Decision                 8
dtype: int64
```

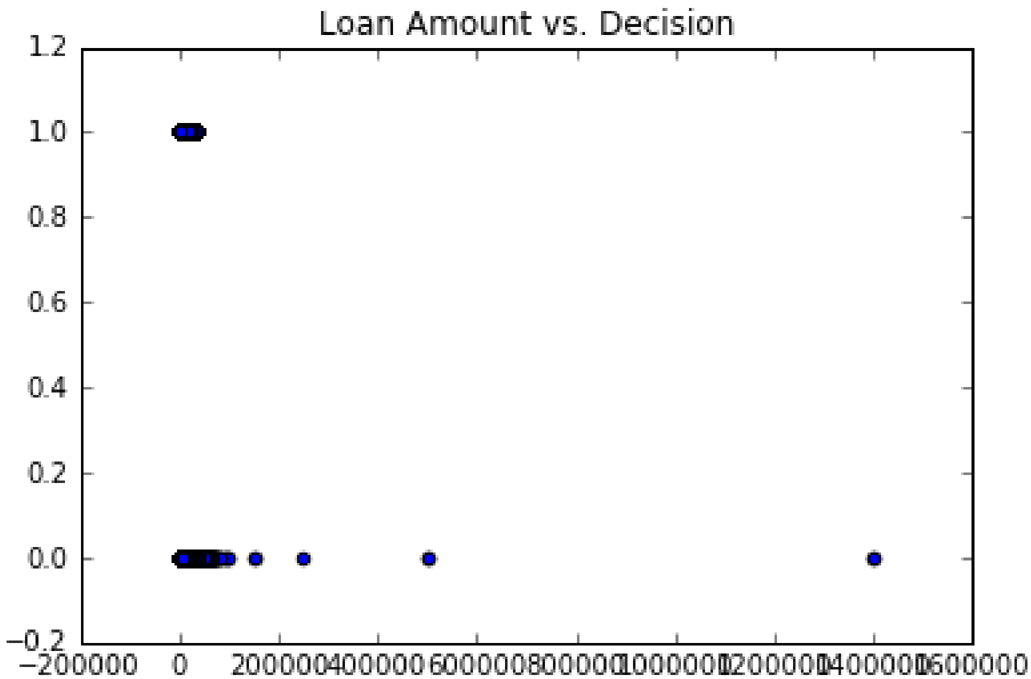
In [6]:
```
df = df[df['Debt-To-Income Ratio'] < 100000]
df.describe()
```

Out[6]:
```
        Loan Amount  Employment Length    Annual Income  Debt-To-Income Ratio  \
count   6402866.000        6402866.000      6402866.000           6402866.000
mean      13049.354              2.252       329818.153                 4.374
std       10256.161              2.878     15400547.126                60.749
min        1000.000              0.000            1.200                 0.000
25%        5000.000              1.000         7896.000                 0.112
50%       10000.000              1.000        28884.000                 0.238
75%       20000.000              1.000        68000.000                 0.466
max     1400000.000             62.917  13226400000.000             44088.000

          Decision
count  6402866.000
mean         0.150
std          0.357
min          0.000
25%          0.000
50%          0.000
75%          0.000
max          1.000
```

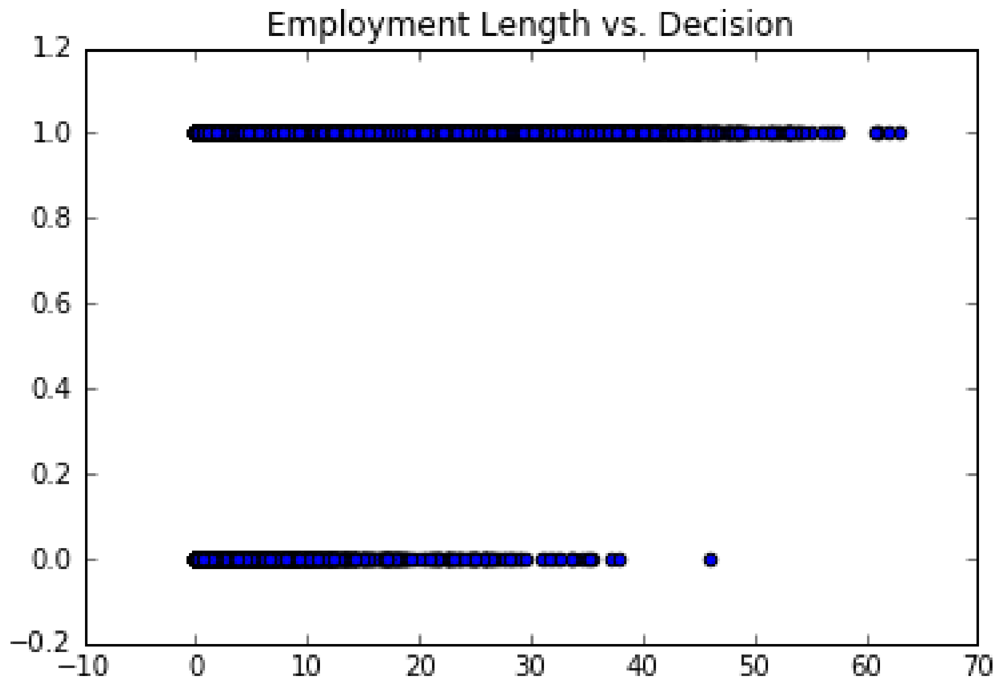
In [7]:
```
plt.scatter(df['Loan Amount'], df['Decision'])
plt.title('Loan Amount vs. Decision')
```

Out[7]: <matplotlib.text.Text at 0x7ff169955f10>



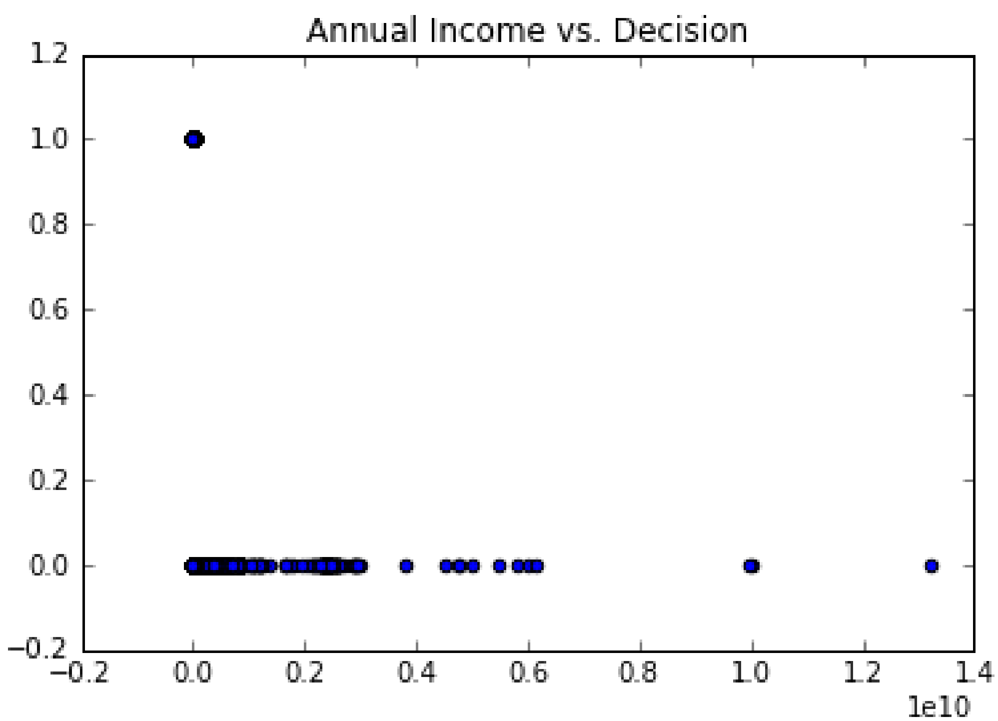
In [8]:
```
plt.scatter(df['Employment Length'], df['Decision'])
plt.title('Employment Length vs. Decision')
```

Out[8]: <matplotlib.text.Text at 0x7ff174704a10>



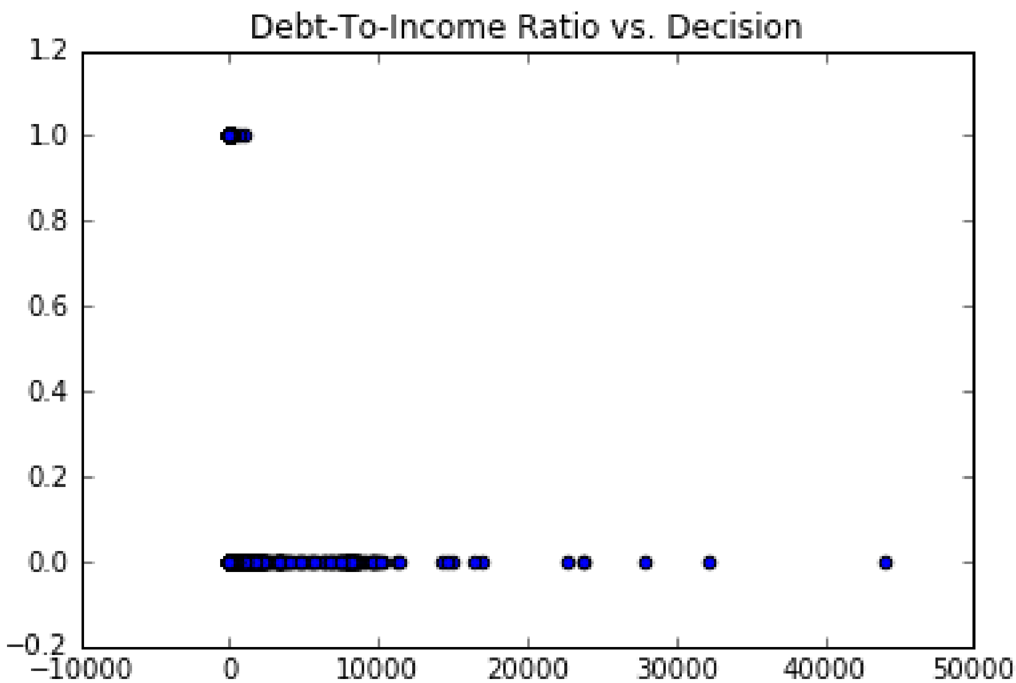

```
In [9]: plt.scatter(df['Annual Income'], df['Decision'])
        plt.title('Annual Income vs. Decision')
```

```
Out[9]: <matplotlib.text.Text at 0x7ff1746ebb90>
```



In [10]:
```
plt.scatter(df['Debt-To-Income Ratio'], df['Decision'])
plt.title('Debt-To-Income Ratio vs. Decision')
```

Out[10]:
```
<matplotlib.text.Text at 0x7ff16e3e6c90>
```



1 Classification Algorithms

In [11]:
```
x = np.array(df.drop(['Decision'],1))
y = np.array(df['Decision'])

x_train, x_test, y_train, y_test = cross_validation.train_test_split(x,y,test_size=0.22)
```

1.1 Gradient Boosting

In [12]:
```
from sklearn import metrics
```

Feature importances tell us which features bear more weight in the decision.

In this case, the debt-to-income ratio is roughly a good indicator whether we should give out a loan or not. This is beneficial for us because that ratio is dimensionless, meaning, currency doesn't matter.

Moreover, both employment length and the annual income play equal parts in a decision.

Lastly, it seems that the loan amount is not as important in this model.

GOAL: Reduce the amount of false negatives. Better to falsely accuse a person of defaulting rather than falsely speculating that a person would pay the debt.

In [13]:
```
clf_gbc = ensemble.GradientBoostingClassifier().fit(x_train,y_train)
print clf_gbc.score(x_test,y_test)
print clf_gbc.feature_importances_
```

```
0.997384694785
[ 0.09745787  0.27157932  0.26687432  0.36408849]
```

```
In [14]: y_pred_gbc = clf_gbc.predict(x_test)
         print metrics.accuracy_score(y_test,y_pred_gbc)
```

```
0.997384694785
```

```
In [15]: print metrics.confusion_matrix(y_test, y_pred_gbc)

         confusion_gbc = metrics.confusion_matrix(y_test, y_pred_gbc)
         TP_gbc = confusion_gbc[1, 1]
         TN_gbc = confusion_gbc[0, 0]
         FP_gbc = confusion_gbc[0, 1]
         FN_gbc = confusion_gbc[1, 0]
```

```
[[1197186     610]
 [   3074  207761]]
```

```
In [16]: # CLASSIFICATION ACCURACY
         print (TP_gbc + TN_gbc) / float(TP_gbc + TN_gbc + FP_gbc + FN_gbc)
         print metrics.accuracy_score(y_test, y_pred_gbc)
```

```
0.997384694785
0.997384694785
```

```
In [17]: # CLASSIFICATION ERROR
         print (FP_gbc + FN_gbc) / float(TP_gbc + TN_gbc + FP_gbc + FN_gbc)
         print 1 - metrics.accuracy_score(y_test, y_pred_gbc)
```

```
0.00261530521478
0.00261530521478
```

```
In [18]: # Sensitivity: How sensitive is the model in predicting positive instances?

         print TP_gbc / float(TP_gbc + FN_gbc)
         print metrics.recall_score(y_test, y_pred_gbc)
```

```
0.985419878104
0.985419878104
```

```
In [19]: # Specificity: When it's actually no, how often does it predict no?
         # True Negative Rate

         print TN_gbc / float(TN_gbc + FP_gbc)
```

```
0.99949073131
```

```
In [20]: # False Positive Rate: When it's actually no, how often does it predict yes?

         print FP_gbc / float(TN_gbc + FP_gbc)
```

```
0.000509268690161
```

```
In [21]: # Precision: When it predicts yes, how often is it correct?

         print TP_gbc / float(TP_gbc + FP_gbc)
         print metrics.precision_score(y_test, y_pred_gbc)
```

```
0.997072529287
0.997072529287
```

```
In [22]: # F1 Score
         print metrics.f1_score(y_test,y_pred_gbc)
```

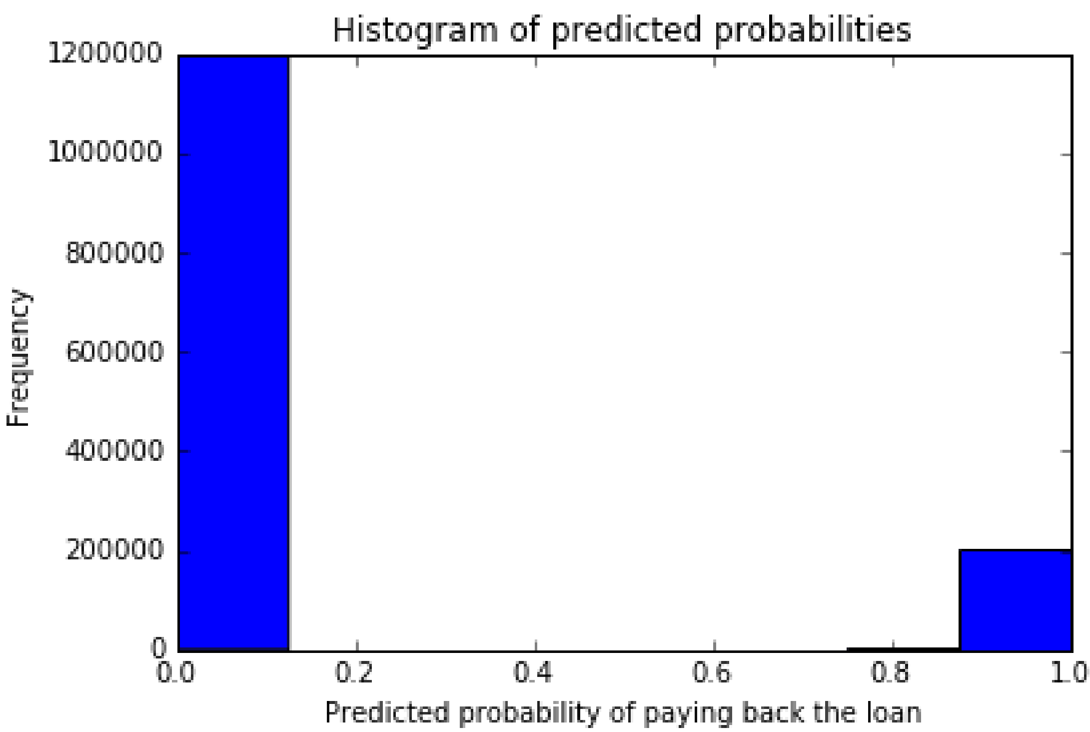
```
0.991211957844
```

1.1.1 Probabilities

```
In [89]: y_pred_prob_gbc = clf_gbc.predict_proba(x_test)
```

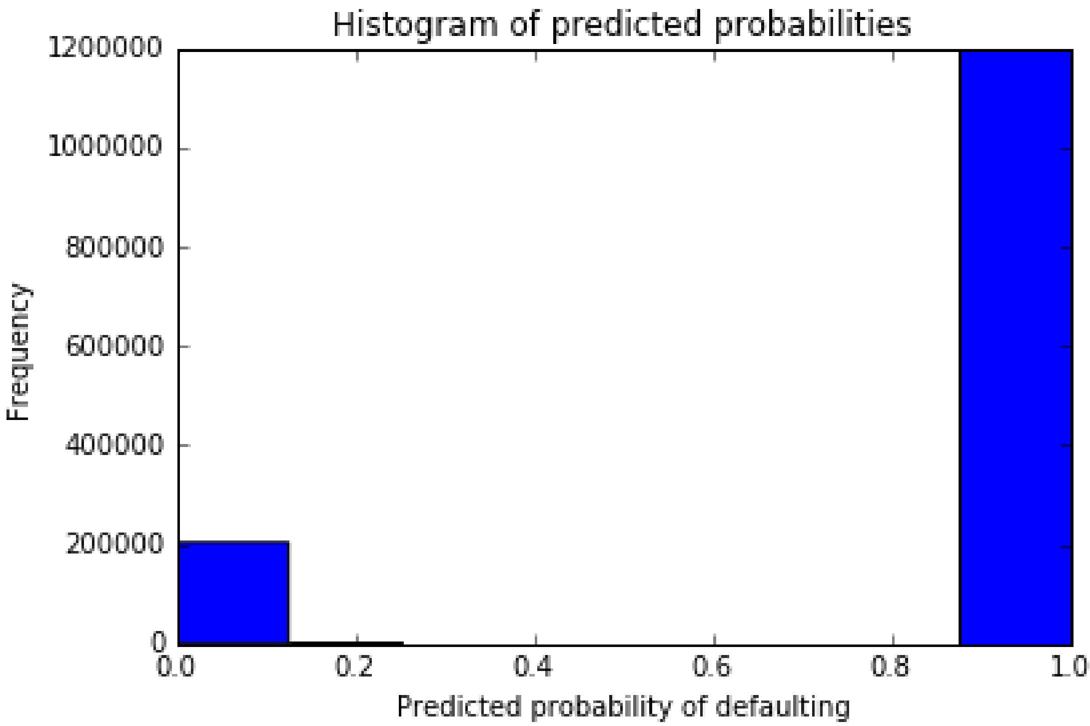
```
In [90]: plt.hist(y_pred_prob_gbc[:,1], bins=8)
         plt.xlim(0, 1)
         plt.title('Histogram of predicted probabilities')
         plt.xlabel('Predicted probability of paying back the loan')
         plt.ylabel('Frequency')
```

```
Out[90]: <matplotlib.text.Text at 0x7ff168cfcc10>
```



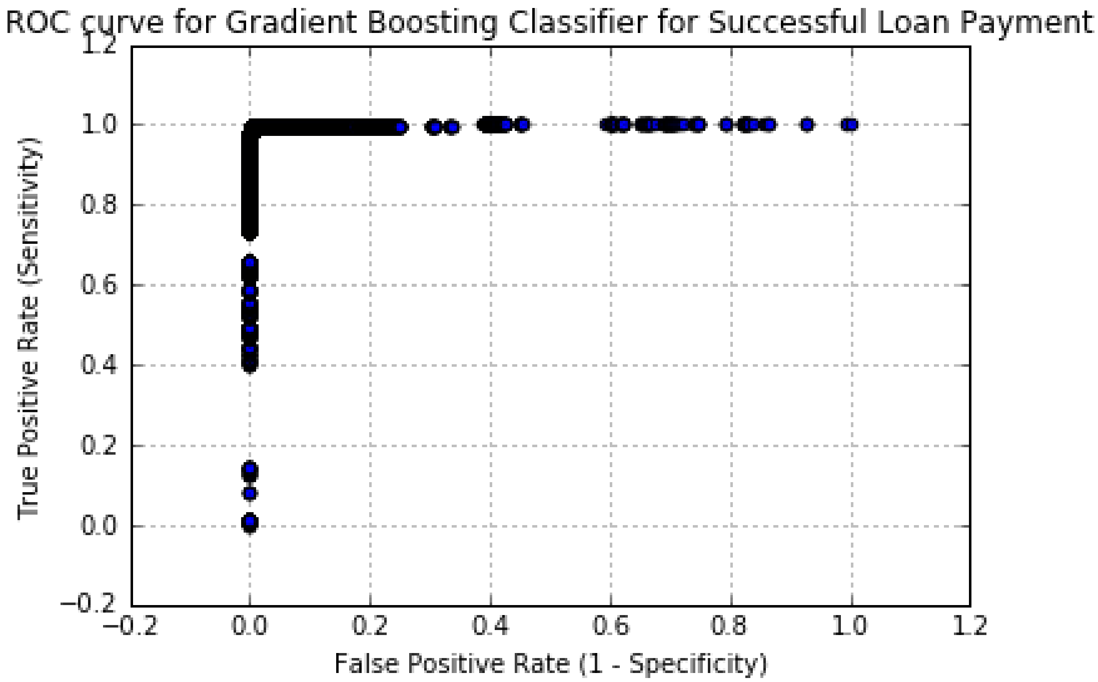

```
In [91]: plt.hist(y_pred_prob_gbc[:,0], bins=8)
         plt.xlim(0, 1)
         plt.title('Histogram of predicted probabilities')
         plt.xlabel('Predicted probability of defaulting')
         plt.ylabel('Frequency')
```

```
Out[91]: <matplotlib.text.Text at 0x7ff168b679d0>
```



1.1.2 ROC and AUC

In [123]:
```
fpr_gbc, tpr_gbc, thresholds_gbc = metrics.roc_curve(y_test, y_pred_prob_gbc[:,1])
plt.scatter(fpr_gbc, tpr_gbc)
plt.title('ROC curve for Gradient Boosting Classifier for Successful Loan Payment')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```




```
In [122]: print metrics.roc_auc_score(y_test, y_pred_prob_gbc[:,1])
```

```
0.999673438921
```

1.2 Random Forest

For the Random Forest model, we see that much like the Gradient Boosting Model, loan amount is given less importance than the rest of the explanatory variables.

Moreover, much like the Gradient Boosting model, the debt-to-income ratio bears the most importance in determining loan defaults. This is again an important conclusion because this ratio is dimensionless.

In this model however, more importance is given to the employment length rather than the annual income.

```
In [28]: clf_rf = ensemble.RandomForestClassifier().fit(x_train,y_train)
         print clf_rf.score(x_test,y_test)
         print clf_rf.feature_importances_
```

```
0.99944840061
[ 0.01845201  0.12237972  0.06705893  0.79210935]
```

```
In [29]: y_pred_rf = clf_rf.predict(x_test)
         print metrics.accuracy_score(y_test, y_pred_rf)
```

```
0.99944840061
```

```
In [30]: print metrics.confusion_matrix(y_test, y_pred_rf)

         confusion_rf = metrics.confusion_matrix(y_test, y_pred_rf)
         TP_rf = confusion_rf[1, 1]
         TN_rf = confusion_rf[0, 0]
         FP_rf = confusion_rf[0, 1]
         FN_rf = confusion_rf[1, 0]
```

```
[[1197512     284]
 [    493  210342]]
```

```
In [31]: # CLASSIFICATION ACCURACY
         print (TP_rf + TN_rf) / float(TP_rf + TN_rf + FP_rf + FN_rf)
         print metrics.accuracy_score(y_test, y_pred_rf)
```

```
0.99944840061
0.99944840061
```

```
In [32]: # CLASSIFICATION ERROR
         print (FP_rf + FN_rf) / float(TP_rf + TN_rf + FP_rf + FN_rf)
         print 1 - metrics.accuracy_score(y_test, y_pred_rf)
```

```
0.000551599389762
0.000551599389762
```

```
In [33]: # Sensitivity: How sensitive is the model in predicting positive instances?

         print TP_rf / float(TP_rf + FN_rf)
         print metrics.recall_score(y_test, y_pred_rf)
```

```
0.997661678564
0.997661678564
```

```
In [34]: # Specificity: When it's actually no, how often does it predict no?
         # True Negative Rate

         print TN_rf / float(TN_rf + FP_rf)
```

```
0.999762897856
```

```
In [35]: # False Positive Rate: When it's actually no, how often does it predict yes?

         print FP_rf / float(TN_rf + FP_rf)
```

```
0.000237102144272
```

```
In [36]: # Precision: When it predicts yes, how often is it correct?

         print TP_rf / float(TP_rf + FP_rf)
         print metrics.precision_score(y_test, y_pred_rf)
```

```
0.998651638449
0.998651638449
```

```
In [37]: # F1 Score
         print metrics.f1_score(y_test,y_pred_rf)
```

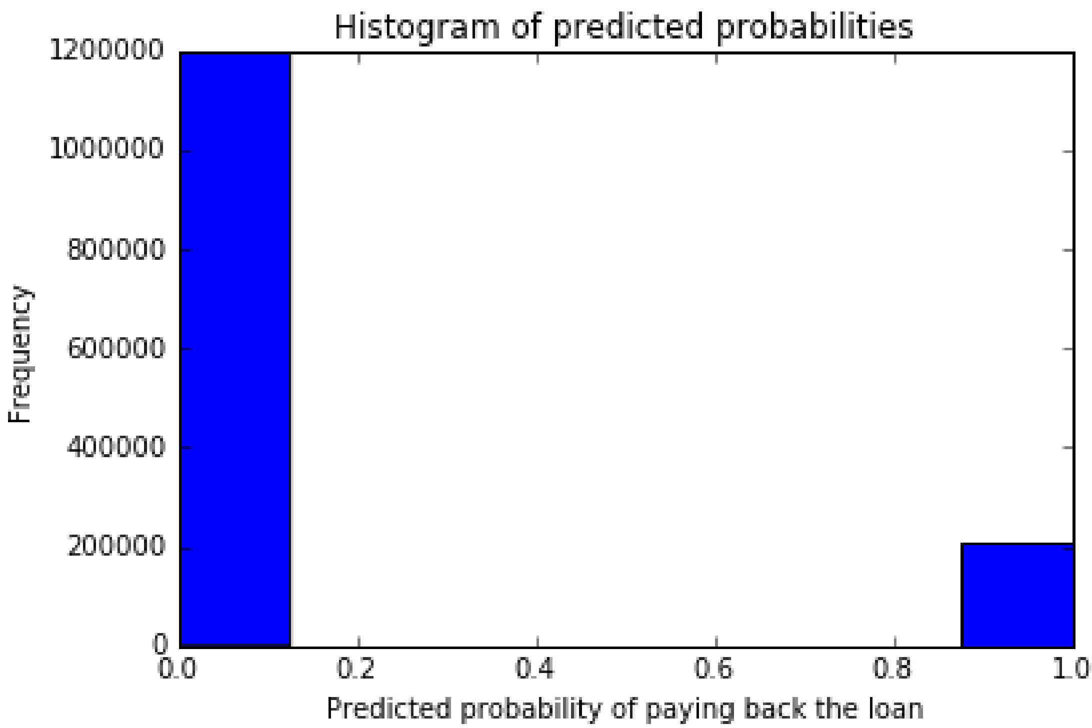
```
0.998156413049
```

1.2.1 Probabilities

```
In [100]: y_pred_prob_rf = clf_rf.predict_proba(x_test)
```

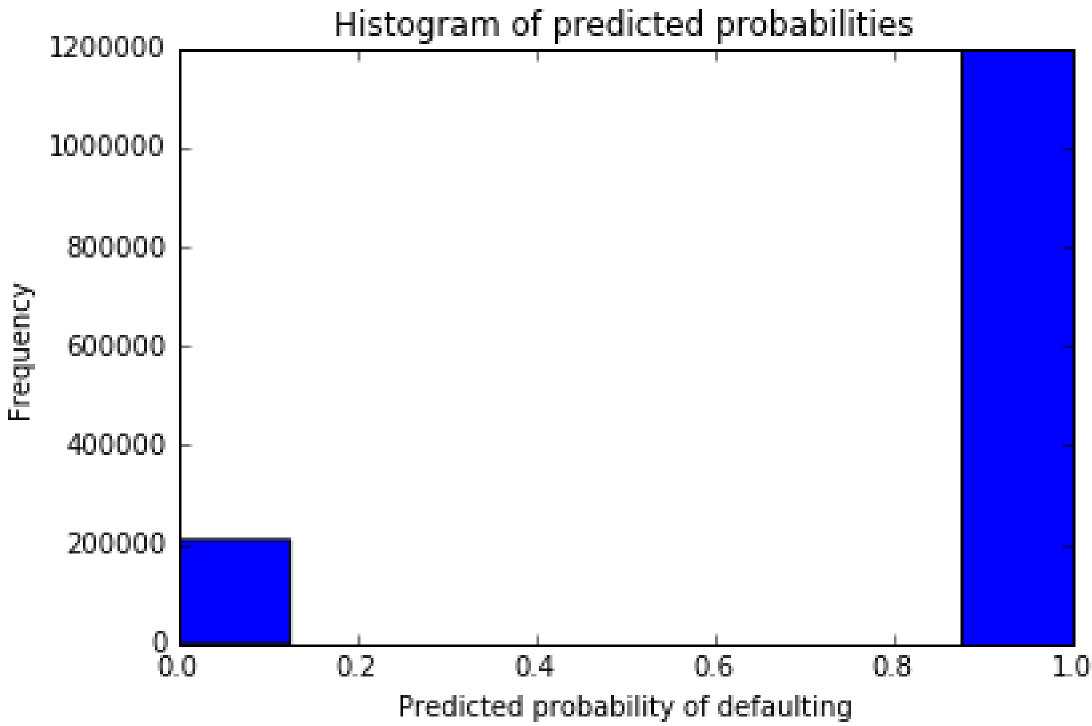
```
In [101]: plt.hist(y_pred_prob_rf[:,1], bins=8)
          plt.xlim(0, 1)
          plt.title('Histogram of predicted probabilities')
          plt.xlabel('Predicted probability of paying back the loan')
          plt.ylabel('Frequency')
```

Out[101]: <matplotlib.text.Text at 0x7ff169177910>



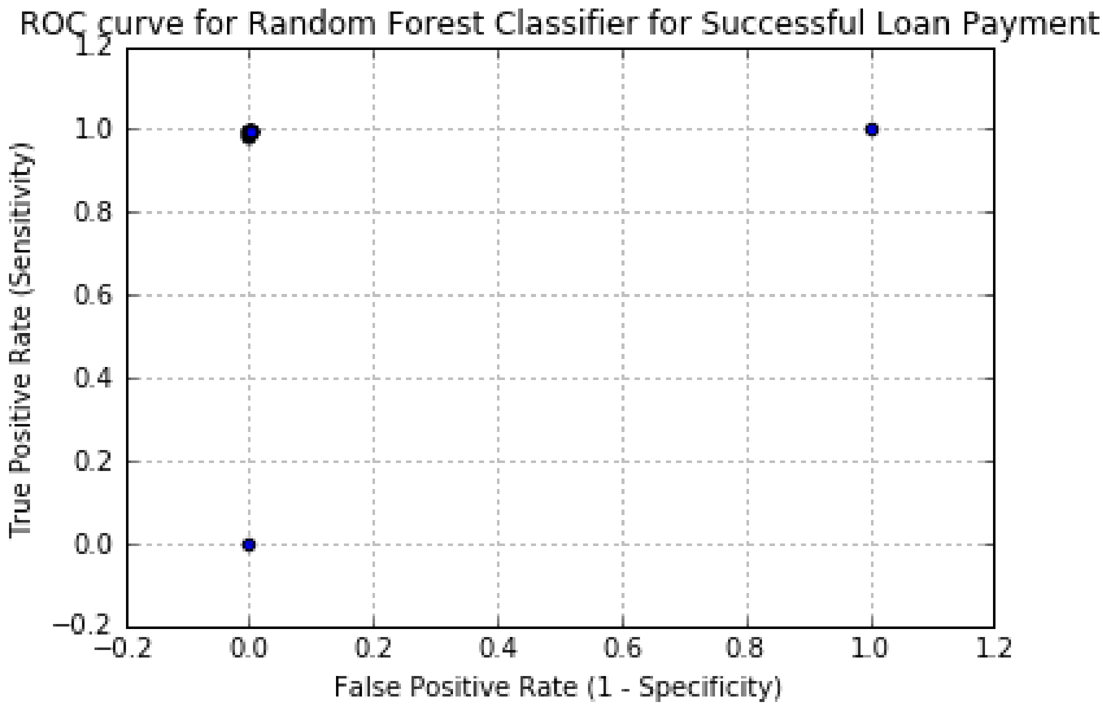

```
In [102]: plt.hist(y_pred_prob_rf[:,0], bins=8)
          plt.xlim(0, 1)
          plt.title('Histogram of predicted probabilities')
          plt.xlabel('Predicted probability of defaulting')
          plt.ylabel('Frequency')
```

Out[102]: <matplotlib.text.Text at 0x7ff169177c50>



1.2.2 ROC and AUC

In [119]:
```
fpr_rf, tpr_rf, thresholds_rf = metrics.roc_curve(y_test, y_pred_prob_rf[:,1])
plt.scatter(fpr_rf, tpr_rf)
plt.title('ROC curve for Random Forest Classifier for Successful Loan Payment')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```




```
In [120]: print metrics.roc_auc_score(y_test, y_pred_prob_rf[:,1])
```

```
0.999744442952
```

1.3 AdaBoost

The observation that the debt-to-income ratio parameter is a good indicator for loan defaults is again proven here in this model.

However, in this model, we see that the employment length also plays an important role in determining loan defaults, more so than that of the annual income.

Lastly, as we can see yet again, the loan amount parameter plays little role in explaining loan defaults.

```
In [43]: clf_ada = ensemble.AdaBoostClassifier().fit(x_train,y_train)
         print clf_ada.score(x_test,y_test)
         print clf_ada.feature_importances_
```

```
0.994604690654
[ 0.06  0.38  0.2   0.36]
```

```
In [44]: y_pred_ada = clf_ada.predict(x_test)
         print metrics.accuracy_score(y_test,y_pred_ada)
```

```
0.994604690654
```

```
In [45]: print metrics.confusion_matrix(y_test, y_pred_ada)

         confusion_ada = metrics.confusion_matrix(y_test, y_pred_ada)
         TP_ada = confusion_ada[1, 1]
         TN_ada = confusion_ada[0, 0]
         FP_ada = confusion_ada[0, 1]
         FN_ada = confusion_ada[1, 0]
```

```
[[1194822    2974]
 [   4626  206209]]
```

```
In [46]: # CLASSIFICATION ACCURACY
         print (TP_ada + TN_ada) / float(TP_ada + TN_ada + FP_ada + FN_ada)
         print metrics.accuracy_score(y_test, y_pred_ada)
```

```
0.994604690654
0.994604690654
```

```
In [47]: # CLASSIFICATION ERROR
         print (FP_ada + FN_ada) / float(TP_ada + TN_ada + FP_ada + FN_ada)
         print 1 - metrics.accuracy_score(y_test, y_pred_ada)
```

```
0.00539530934645
0.00539530934645
```

```
In [48]: # Sensitivity: How sensitive is the model in predicting positive instances?

         print TP_ada / float(TP_ada + FN_ada)
         print metrics.recall_score(y_test, y_pred_ada)
```

```
0.978058671473
0.978058671473
```

```
In [49]: # Specificity: When it's actually no, how often does it predict no?
         # True Negative Rate

         print TN_ada / float(TN_ada + FP_ada)
```

```
0.997517106419
```

```
In [50]: # False Positive Rate: When it's actually no, how often does it predict yes?

         print FP_ada / float(TN_ada + FP_ada)
```

```
0.00248289358121
```

```
In [51]: # Precision: When it predicts yes, how often is it correct?

         print TP_ada / float(TP_ada + FP_ada)
         print metrics.precision_score(y_test, y_pred_ada)
```

```
0.985782783496
0.985782783496
```

```
In [52]: # F1 Score
         print metrics.f1_score(y_test,y_pred_ada)
```

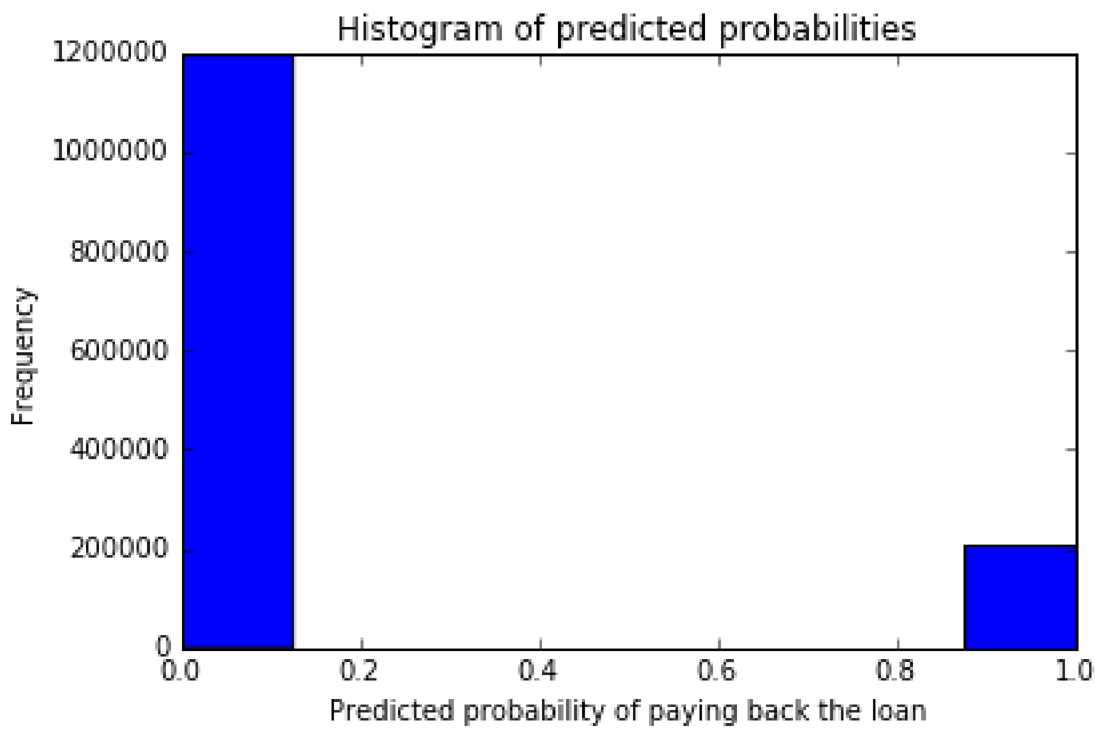
```
0.981905537382
```

1.3.1 Probabilities

```
In [84]: y_pred_prob_ada = clf_rf.predict_proba(x_test)
```

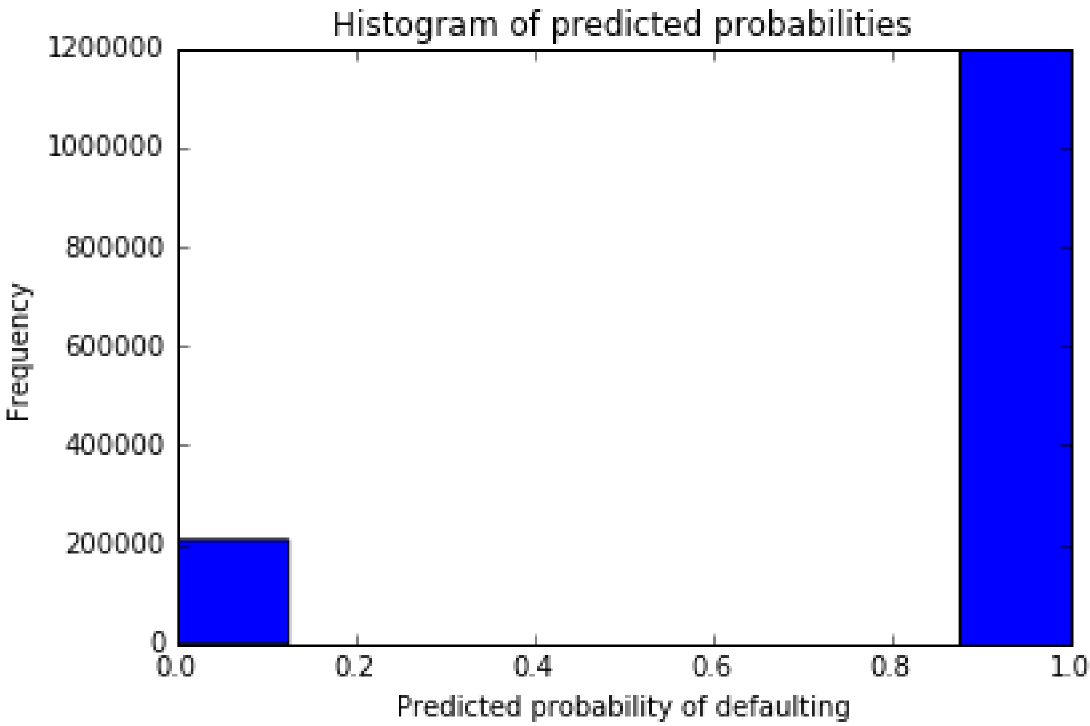
```
In [85]: plt.hist(y_pred_prob_ada[:,1], bins=8)
         plt.xlim(0, 1)
         plt.title('Histogram of predicted probabilities')
         plt.xlabel('Predicted probability of paying back the loan')
         plt.ylabel('Frequency')
```

Out[85]: <matplotlib.text.Text at 0x7ff168e11fd0>



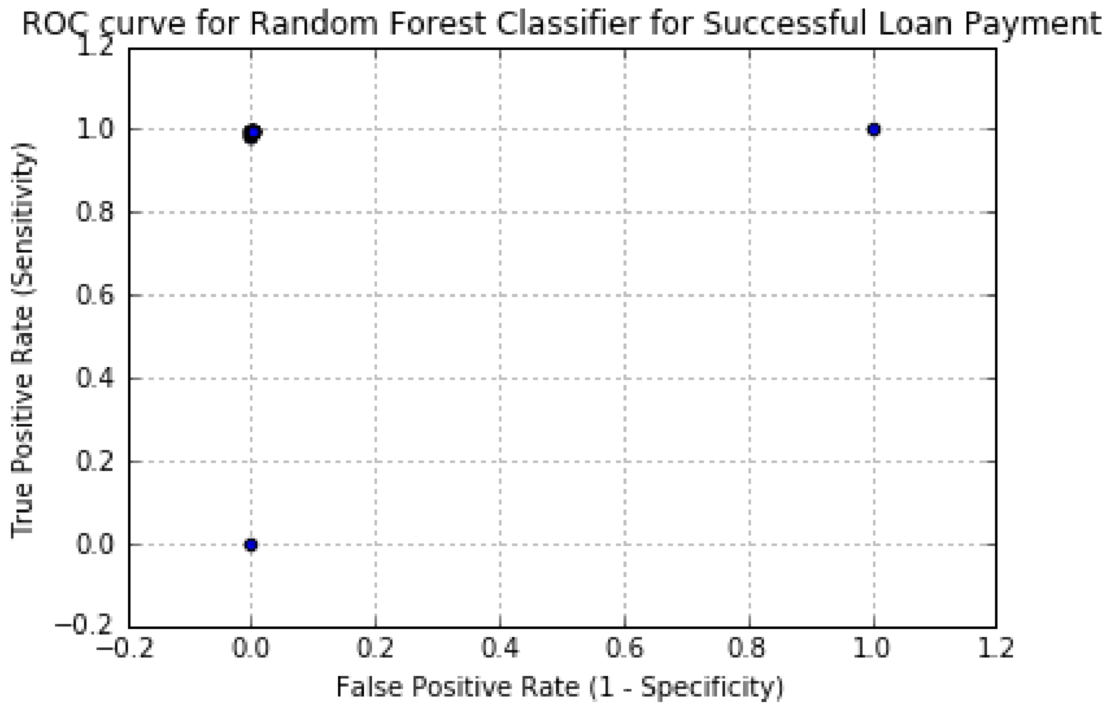
In [86]:
```
plt.hist(y_pred_prob_ada[:,0], bins=8)
plt.xlim(0, 1)
plt.title('Histogram of predicted probabilities')
plt.xlabel('Predicted probability of defaulting')
plt.ylabel('Frequency')
```

Out[86]: <matplotlib.text.Text at 0x7ff168cfce10>



1.3.2 ROC and AUC

In [117]:
```
fpr_ada, tpr_ada, thresholds_ada = metrics.roc_curve(y_test, y_pred_prob_ada[:,1])
plt.scatter(fpr_ada, tpr_ada)
plt.title('ROC curve for Random Forest Classifier for Successful Loan Payment')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```




```
In [118]: print metrics.roc_auc_score(y_test, y_pred_prob_ada[:,1])
```

```
0.999744442952
```

2 Exporting Models

```
In [61]: from sklearn.externals import joblib

         joblib.dump(clf_gbc, 'Gradient Boosting Model/Gradient Boost Model for Credit Default.cls')
         joblib.dump(clf_rf, 'Random Forest Model/Random Forest Model for Credit Default.cls')
         joblib.dump(clf_ada, 'AdaBoost Model/AdaBoost Model for Credit Default.cls')
```

```
Out[61]: ['AdaBoost Model for Credit Default.cls',
          'AdaBoost Model for Credit Default.cls_01.npy',
          'AdaBoost Model for Credit Default.cls_02.npy',
          'AdaBoost Model for Credit Default.cls_03.npy',
          'AdaBoost Model for Credit Default.cls_04.npy',
          'AdaBoost Model for Credit Default.cls_05.npy',
          'AdaBoost Model for Credit Default.cls_06.npy',
          'AdaBoost Model for Credit Default.cls_07.npy',
          'AdaBoost Model for Credit Default.cls_08.npy',
          'AdaBoost Model for Credit Default.cls_09.npy',
          'AdaBoost Model for Credit Default.cls_10.npy',
          'AdaBoost Model for Credit Default.cls_11.npy',
          'AdaBoost Model for Credit Default.cls_12.npy',
          'AdaBoost Model for Credit Default.cls_13.npy',
          'AdaBoost Model for Credit Default.cls_14.npy',
```

```
'AdaBoost Model for Credit Default.cls_15.npy',
'AdaBoost Model for Credit Default.cls_16.npy',
'AdaBoost Model for Credit Default.cls_17.npy',
'AdaBoost Model for Credit Default.cls_18.npy',
'AdaBoost Model for Credit Default.cls_19.npy',
'AdaBoost Model for Credit Default.cls_20.npy',
'AdaBoost Model for Credit Default.cls_21.npy',
'AdaBoost Model for Credit Default.cls_22.npy',
'AdaBoost Model for Credit Default.cls_23.npy',
'AdaBoost Model for Credit Default.cls_24.npy',
'AdaBoost Model for Credit Default.cls_25.npy',
'AdaBoost Model for Credit Default.cls_26.npy',
'AdaBoost Model for Credit Default.cls_27.npy',
'AdaBoost Model for Credit Default.cls_28.npy',
'AdaBoost Model for Credit Default.cls_29.npy',
'AdaBoost Model for Credit Default.cls_30.npy',
'AdaBoost Model for Credit Default.cls_31.npy',
'AdaBoost Model for Credit Default.cls_32.npy',
'AdaBoost Model for Credit Default.cls_33.npy',
'AdaBoost Model for Credit Default.cls_34.npy',
'AdaBoost Model for Credit Default.cls_35.npy',
'AdaBoost Model for Credit Default.cls_36.npy',
'AdaBoost Model for Credit Default.cls_37.npy',
'AdaBoost Model for Credit Default.cls_38.npy',
'AdaBoost Model for Credit Default.cls_39.npy',
'AdaBoost Model for Credit Default.cls_40.npy',
'AdaBoost Model for Credit Default.cls_41.npy',
'AdaBoost Model for Credit Default.cls_42.npy',
'AdaBoost Model for Credit Default.cls_43.npy',
'AdaBoost Model for Credit Default.cls_44.npy',
'AdaBoost Model for Credit Default.cls_45.npy',
'AdaBoost Model for Credit Default.cls_46.npy',
'AdaBoost Model for Credit Default.cls_47.npy',
'AdaBoost Model for Credit Default.cls_48.npy',
'AdaBoost Model for Credit Default.cls_49.npy',
'AdaBoost Model for Credit Default.cls_50.npy',
'AdaBoost Model for Credit Default.cls_51.npy',
'AdaBoost Model for Credit Default.cls_52.npy',
'AdaBoost Model for Credit Default.cls_53.npy',
'AdaBoost Model for Credit Default.cls_54.npy',
'AdaBoost Model for Credit Default.cls_55.npy',
'AdaBoost Model for Credit Default.cls_56.npy',
'AdaBoost Model for Credit Default.cls_57.npy',
'AdaBoost Model for Credit Default.cls_58.npy',
'AdaBoost Model for Credit Default.cls_59.npy',
'AdaBoost Model for Credit Default.cls_60.npy',
'AdaBoost Model for Credit Default.cls_61.npy',
'AdaBoost Model for Credit Default.cls_62.npy',
'AdaBoost Model for Credit Default.cls_63.npy',
'AdaBoost Model for Credit Default.cls_64.npy',
'AdaBoost Model for Credit Default.cls_65.npy',
'AdaBoost Model for Credit Default.cls_66.npy',
'AdaBoost Model for Credit Default.cls_67.npy',
'AdaBoost Model for Credit Default.cls_68.npy',
```

```
'AdaBoost Model for Credit Default.cls_69.npy',
'AdaBoost Model for Credit Default.cls_70.npy',
'AdaBoost Model for Credit Default.cls_71.npy',
'AdaBoost Model for Credit Default.cls_72.npy',
'AdaBoost Model for Credit Default.cls_73.npy',
'AdaBoost Model for Credit Default.cls_74.npy',
'AdaBoost Model for Credit Default.cls_75.npy',
'AdaBoost Model for Credit Default.cls_76.npy',
'AdaBoost Model for Credit Default.cls_77.npy',
'AdaBoost Model for Credit Default.cls_78.npy',
'AdaBoost Model for Credit Default.cls_79.npy',
'AdaBoost Model for Credit Default.cls_80.npy',
'AdaBoost Model for Credit Default.cls_81.npy',
'AdaBoost Model for Credit Default.cls_82.npy',
'AdaBoost Model for Credit Default.cls_83.npy',
'AdaBoost Model for Credit Default.cls_84.npy',
'AdaBoost Model for Credit Default.cls_85.npy',
'AdaBoost Model for Credit Default.cls_86.npy',
'AdaBoost Model for Credit Default.cls_87.npy',
'AdaBoost Model for Credit Default.cls_88.npy',
'AdaBoost Model for Credit Default.cls_89.npy',
'AdaBoost Model for Credit Default.cls_90.npy',
'AdaBoost Model for Credit Default.cls_91.npy',
'AdaBoost Model for Credit Default.cls_92.npy',
'AdaBoost Model for Credit Default.cls_93.npy',
'AdaBoost Model for Credit Default.cls_94.npy',
'AdaBoost Model for Credit Default.cls_95.npy',
'AdaBoost Model for Credit Default.cls_96.npy',
'AdaBoost Model for Credit Default.cls_97.npy',
'AdaBoost Model for Credit Default.cls_98.npy',
'AdaBoost Model for Credit Default.cls_99.npy',
'AdaBoost Model for Credit Default.cls_100.npy',
'AdaBoost Model for Credit Default.cls_101.npy',
'AdaBoost Model for Credit Default.cls_102.npy',
'AdaBoost Model for Credit Default.cls_103.npy',
'AdaBoost Model for Credit Default.cls_104.npy',
'AdaBoost Model for Credit Default.cls_105.npy',
'AdaBoost Model for Credit Default.cls_106.npy',
'AdaBoost Model for Credit Default.cls_107.npy',
'AdaBoost Model for Credit Default.cls_108.npy',
'AdaBoost Model for Credit Default.cls_109.npy',
'AdaBoost Model for Credit Default.cls_110.npy',
'AdaBoost Model for Credit Default.cls_111.npy',
'AdaBoost Model for Credit Default.cls_112.npy',
'AdaBoost Model for Credit Default.cls_113.npy',
'AdaBoost Model for Credit Default.cls_114.npy',
'AdaBoost Model for Credit Default.cls_115.npy',
'AdaBoost Model for Credit Default.cls_116.npy',
'AdaBoost Model for Credit Default.cls_117.npy',
'AdaBoost Model for Credit Default.cls_118.npy',
'AdaBoost Model for Credit Default.cls_119.npy',
'AdaBoost Model for Credit Default.cls_120.npy',
'AdaBoost Model for Credit Default.cls_121.npy',
'AdaBoost Model for Credit Default.cls_122.npy',
```

```
'AdaBoost Model for Credit Default.cls_123.npy',
'AdaBoost Model for Credit Default.cls_124.npy',
'AdaBoost Model for Credit Default.cls_125.npy',
'AdaBoost Model for Credit Default.cls_126.npy',
'AdaBoost Model for Credit Default.cls_127.npy',
'AdaBoost Model for Credit Default.cls_128.npy',
'AdaBoost Model for Credit Default.cls_129.npy',
'AdaBoost Model for Credit Default.cls_130.npy',
'AdaBoost Model for Credit Default.cls_131.npy',
'AdaBoost Model for Credit Default.cls_132.npy',
'AdaBoost Model for Credit Default.cls_133.npy',
'AdaBoost Model for Credit Default.cls_134.npy',
'AdaBoost Model for Credit Default.cls_135.npy',
'AdaBoost Model for Credit Default.cls_136.npy',
'AdaBoost Model for Credit Default.cls_137.npy',
'AdaBoost Model for Credit Default.cls_138.npy',
'AdaBoost Model for Credit Default.cls_139.npy',
'AdaBoost Model for Credit Default.cls_140.npy',
'AdaBoost Model for Credit Default.cls_141.npy',
'AdaBoost Model for Credit Default.cls_142.npy',
'AdaBoost Model for Credit Default.cls_143.npy',
'AdaBoost Model for Credit Default.cls_144.npy',
'AdaBoost Model for Credit Default.cls_145.npy',
'AdaBoost Model for Credit Default.cls_146.npy',
'AdaBoost Model for Credit Default.cls_147.npy',
'AdaBoost Model for Credit Default.cls_148.npy',
'AdaBoost Model for Credit Default.cls_149.npy',
'AdaBoost Model for Credit Default.cls_150.npy',
'AdaBoost Model for Credit Default.cls_151.npy',
'AdaBoost Model for Credit Default.cls_152.npy',
'AdaBoost Model for Credit Default.cls_153.npy',
'AdaBoost Model for Credit Default.cls_154.npy',
'AdaBoost Model for Credit Default.cls_155.npy',
'AdaBoost Model for Credit Default.cls_156.npy',
'AdaBoost Model for Credit Default.cls_157.npy',
'AdaBoost Model for Credit Default.cls_158.npy',
'AdaBoost Model for Credit Default.cls_159.npy',
'AdaBoost Model for Credit Default.cls_160.npy',
'AdaBoost Model for Credit Default.cls_161.npy',
'AdaBoost Model for Credit Default.cls_162.npy',
'AdaBoost Model for Credit Default.cls_163.npy',
'AdaBoost Model for Credit Default.cls_164.npy',
'AdaBoost Model for Credit Default.cls_165.npy',
'AdaBoost Model for Credit Default.cls_166.npy',
'AdaBoost Model for Credit Default.cls_167.npy',
'AdaBoost Model for Credit Default.cls_168.npy',
'AdaBoost Model for Credit Default.cls_169.npy',
'AdaBoost Model for Credit Default.cls_170.npy',
'AdaBoost Model for Credit Default.cls_171.npy',
'AdaBoost Model for Credit Default.cls_172.npy',
'AdaBoost Model for Credit Default.cls_173.npy',
'AdaBoost Model for Credit Default.cls_174.npy',
'AdaBoost Model for Credit Default.cls_175.npy',
'AdaBoost Model for Credit Default.cls_176.npy',
```

```
 'AdaBoost Model for Credit Default.cls_177.npy',
 'AdaBoost Model for Credit Default.cls_178.npy',
 'AdaBoost Model for Credit Default.cls_179.npy',
 'AdaBoost Model for Credit Default.cls_180.npy',
 'AdaBoost Model for Credit Default.cls_181.npy',
 'AdaBoost Model for Credit Default.cls_182.npy',
 'AdaBoost Model for Credit Default.cls_183.npy',
 'AdaBoost Model for Credit Default.cls_184.npy',
 'AdaBoost Model for Credit Default.cls_185.npy',
 'AdaBoost Model for Credit Default.cls_186.npy',
 'AdaBoost Model for Credit Default.cls_187.npy',
 'AdaBoost Model for Credit Default.cls_188.npy',
 'AdaBoost Model for Credit Default.cls_189.npy',
 'AdaBoost Model for Credit Default.cls_190.npy',
 'AdaBoost Model for Credit Default.cls_191.npy',
 'AdaBoost Model for Credit Default.cls_192.npy',
 'AdaBoost Model for Credit Default.cls_193.npy',
 'AdaBoost Model for Credit Default.cls_194.npy',
 'AdaBoost Model for Credit Default.cls_195.npy',
 'AdaBoost Model for Credit Default.cls_196.npy',
 'AdaBoost Model for Credit Default.cls_197.npy',
 'AdaBoost Model for Credit Default.cls_198.npy',
 'AdaBoost Model for Credit Default.cls_199.npy',
 'AdaBoost Model for Credit Default.cls_200.npy',
 'AdaBoost Model for Credit Default.cls_201.npy',
 'AdaBoost Model for Credit Default.cls_202.npy',
 'AdaBoost Model for Credit Default.cls_203.npy']
```

In [ ]: