Random Forest

April 24, 2019

Random Forest Project

In this project I have explored publicly available data from LendingClub.com. Lending Club connects people who need money (borrowers) with people who have money (investors). Hopefully, as an investor you would want to invest in people who showed a profile of having a high probability of paying you back. I have tried to create a model that will help predict this.

I have used lending data from 2007-2010 and be trying to classify and predict whether or not the borrower paid back their loan in full.

Columns represent:

credit.policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.

purpose: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").

int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates.

installment: The monthly installments owed by the borrower if the loan is funded.

log.annual.inc: The natural log of the self-reported annual income of the borrower.

dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income).

fico: The FICO credit score of the borrower.

days.with.cr.line: The number of days the borrower has had a credit line.

revol.bal: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).

revol.util: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).

inq.last.6mths: The borrower's number of inquiries by creditors in the last 6 months.

delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.

pub.rec: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).

Importing required python libraries and importing loan dataset into pandas as dataframe

```
In [1]: import pandas as pd
```

```
In [2]: import matplotlib.pyplot as plt
        import seaborn as sns
        import sklearn as skl
        import numpy as np
```

```
In [3]: from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.model_selection import train_test_split

In [5]: df = pd.read_csv('loan_data.csv')

In [6]: df.head()
```

Out[6]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not.fully.paid
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	0
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	0
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	0
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	0
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	0

Creating dummies for variable "purpose" which had 7 categories.

```
In [8]: df['purpose'].nunique()
```

Out[8]: 7

```
In [12]: df_final=pd.get_dummies(df,columns=['purpose'],prefix='purpose',drop_first=True)

In [13]: df_final.head()
```

Out[13]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not.fully.paid	purpose_credit_card	purpose_debt_consolidation	purpose_educational	purpose_home_improvement	purpose_major_purchase	purpose_small_business
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	0	0	1	0	0	0	0
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	0	1	0	0	0	0	0
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	0	0	1	0	0	0	0
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	0	0	1	0	0	0	0
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	0	1	0	0	0	0	0

```
2        4710.000000       3511        25.6                1            0
3        2699.958333      33667        73.2                1            0
4        4066.000000       4740        39.5                0            1

   pub.rec  not.fully.paid  purpose_credit_card  purpose_debt_consolidation  \
0        0               0                    0                           1
1        0               0                    1                           0
2        0               0                    0                           1
3        0               0                    0                           1
4        0               0                    1                           0

   purpose_educational  purpose_home_improvement  purpose_major_purchase  \
0                    0                         0                       0
1                    0                         0                       0
2                    0                         0                       0
3                    0                         0                       0
4                    0                         0                       0

   purpose_small_business
0                       0
1                       0
2                       0
3                       0
4                       0
```

```
In [14]: df_y = df_final.pop('not.fully.paid')
         df_x = df_final
```

Splitting the data into training and testing data

```
In [18]: train_x,test_x,train_y,test_y = train_test_split(df_x,df_y,test_size =0.3,random_state
```

```
In [28]: type(train_x)
```

```
Out[28]: pandas.core.frame.DataFrame
```

Creating a Decision Tree Classifier that predicts our target variable and predicting target variable for test data.

```
In [29]: from sklearn.tree import DecisionTreeClassifier
```

```
In [30]: DT=DecisionTreeClassifier()
```

```
In [32]: DT.fit(train_x,train_y)
```

```
Out[32]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                     max_features=None, 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, presort=False, random_state=None,
                     splitter='best')
```

```
In [33]: Tree_Pred = DT.predict(test_x)
```

Creating a Random Forest Classifier that predicts our target variable

```
In [34]: from sklearn.ensemble import RandomForestClassifier

In [37]: RF = RandomForestClassifier(n_estimators=200)

In [39]: RF.fit(train_x,train_y)

Out[39]: 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=200, n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)

In [40]: Forest_Pred = RF.predict(test_x)
```

Comparing results from decision tree and random forest

```
In [41]: from sklearn.metrics import confusion_matrix,classification_report

In [43]: print('For Decision Tree:\n')
         print(confusion_matrix(test_y,Tree_Pred))
         print(classification_report(test_y,Tree_Pred))
         print('For Random Forest:\n')
         print(confusion_matrix(test_y,Forest_Pred))
         print(classification_report(test_y,Forest_Pred))

For Decision Tree:

[[1984  447]
 [ 333  110]]
              precision    recall  f1-score   support

           0       0.86      0.82      0.84      2431
           1       0.20      0.25      0.22       443

   micro avg       0.73      0.73      0.73      2874
   macro avg       0.53      0.53      0.53      2874
weighted avg       0.75      0.73      0.74      2874

For Random Forest:

[[2421   10]
 [ 435    8]]
              precision    recall  f1-score   support
```

```
              0       0.85      1.00      0.92      2431
              1       0.44      0.02      0.03       443

   micro avg       0.85      0.85      0.85      2874
   macro avg       0.65      0.51      0.48      2874
weighted avg       0.79      0.85      0.78      2874
```

From the above comparision we can see that random forest classifier with an accuracy of 0.79 performed better than the decision tree which had an accuracy of 0.75

In [ ]: