Loan_Analysis

October 17, 2018

```
In [3]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

```
In [4]: #This dataset contains loan characteristics of a bank's customers.
        #The goal is to clean the dataset, create a model that predicts loan outcome(paid or c
        #find the characteristics that are highly correlated with safe loan applicantions.
```

```
In [5]: #Creating the dataset
        data = pd.read_csv('LoansTrainingSet.csv')
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2
  interactivity=interactivity, compiler=compiler, result=result)
```

```
In [6]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 256984 entries, 0 to 256983
Data columns (total 19 columns):
Loan ID                         256984 non-null object
Customer ID                     256984 non-null object
Loan Status                     256984 non-null object
Current Loan Amount             256984 non-null int64
Term                            256984 non-null object
Credit Score                    195308 non-null float64
Years in current job            245508 non-null object
Home Ownership                  256984 non-null object
Annual Income                   195308 non-null float64
Purpose                         256984 non-null object
Monthly Debt                    256984 non-null object
Years of Credit History         256984 non-null float64
Months since last delinquent    116601 non-null float64
Number of Open Accounts         256984 non-null int64
Number of Credit Problems       256984 non-null int64
Current Credit Balance          256984 non-null int64
Maximum Open Credit             256984 non-null object
Bankruptcies                    256455 non-null float64
```

```
Tax Liens                       256961 non-null float64
dtypes: float64(6), int64(4), object(9)
memory usage: 37.3+ MB
```

```
In [7]: data.head()
```

```
Out[7]:                                 Loan ID                           Customer ID  \
        0  000025bb-5694-4cff-b17d-192b1a98ba44  5ebc8bb1-5eb9-4404-b11b-a6eebc401a19
        1  00002c49-3a29-4bd4-8f67-c8f8fbc1048c  927b388d-2e01-423f-a8dc-f7e42d668f46
        2  00002d89-27f3-409b-aa76-90834f359a65  defce609-c631-447d-aad6-1270615e89c4
        3  00005222-b4d8-45a4-ad8c-186057e24233  070bcecb-aae7-4485-a26a-e0403e7bb6c5
        4  0000757f-a121-41ed-b17b-162e76647c1f  dde79588-12f0-4811-bab0-e2b07f633fcd

          Loan Status  Current Loan Amount        Term  Credit Score  \
        0  Fully Paid                11520  Short Term         741.0
        1  Fully Paid                 3441  Short Term         734.0
        2  Fully Paid                21029  Short Term         747.0
        3  Fully Paid                18743  Short Term         747.0
        4  Fully Paid                11731  Short Term         746.0

          Years in current job Home Ownership  Annual Income             Purpose  \
        0            10+ years  Home Mortgage        33694.0  Debt Consolidation
        1              4 years  Home Mortgage        42269.0               other
        2            10+ years  Home Mortgage        90126.0  Debt Consolidation
        3            10+ years       Own Home        38072.0  Debt Consolidation
        4              4 years           Rent        50025.0  Debt Consolidation

          Monthly Debt  Years of Credit History  Months since last delinquent  \
        0      $584.03                     12.3                          41.0
        1  $1,106.04                      26.3                           NaN
        2  $1,321.85                      28.8                           NaN
        3      $751.92                     26.2                           NaN
        4      $355.18                     11.5                           NaN

           Number of Open Accounts  Number of Credit Problems  Current Credit Balance  \
        0                       10                          0                    6760
        1                       17                          0                    6262
        2                        5                          0                   20967
        3                        9                          0                   22529
        4                       12                          0                   17391

          Maximum Open Credit  Bankruptcies  Tax Liens
        0               16056           0.0        0.0
        1               19149           0.0        0.0
        2               28335           0.0        0.0
        3               43915           0.0        0.0
        4               37081           0.0        0.0
```

```
In [8]: #Get rid of non-integer units in series
        for ch in ['$', ',']:
            data['Monthly Debt'] = [i.replace(ch, '') for i in data['Monthly Debt']]
```

```
In [9]: #Make Monthly Debt numeric through list comprehension
        data['Monthly Debt'] = [float(i) for i in data['Monthly Debt']]
```

```
In [10]: data['Monthly Debt'].head()
```

```
Out[10]: 0     584.03
         1    1106.04
         2    1321.85
         3     751.92
         4     355.18
         Name: Monthly Debt, dtype: float64
```

```
In [11]: data['Loan Status'].unique()
```

```
Out[11]: array(['Fully Paid', 'Charged Off'], dtype=object)
```

```
In [12]: #Change 'Loan Status' unique values to integers, e.g. create dummy variable
         data['Loan Status'] = data['Loan Status'].replace('Fully Paid', 1).replace('Charged O
```

```
In [13]: data['Loan Status'].head()
```

```
Out[13]: 0    1
         1    1
         2    1
         3    1
         4    1
         Name: Loan Status, dtype: int64
```

```
In [14]: #The columns 'Term', 'Home Ownership', and 'Purpose' are all categorical
         #Here we create dummy variables and merge them with the dataset
         data2 = pd.merge(data, (pd.get_dummies(data['Term'], drop_first = True)),left_index =
```

```
In [15]: data3 = pd.merge(data2, (pd.get_dummies(data['Home Ownership'], drop_first = True)), l
```

```
In [16]: data4 = pd.merge(data3, (pd.get_dummies(data['Purpose'], drop_first = True)), left_ind
```

```
In [17]: del data4['Term']
         del data4['Home Ownership']
         del data4['Purpose']
```

```
In [19]: #Next we deal with missing values for 'Bankruptcies' and 'Tax Liens'
         #Correlation map revealed that both variables are correlated with 'Number of Credit P
         #We fill missing values depending on how many credit problems that individual has had
```

```
In [20]: #Knowing how many credit problems someone has can help us predict how many bankruptci
         data4.groupby('Number of Credit Problems')['Bankruptcies'].mean()
```

```
Out[20]: Number of Credit Problems
         0     0.000000
         1     0.822141
         2     0.900234
         3     1.039191
         4     1.058182
         5     1.008000
         6     1.214286
         7     1.187500
         8     0.916667
         9     0.100000
         10    0.666667
         11    0.000000
         Name: Bankruptcies, dtype: float64
```

```
In [21]: #Filling in missing 'Bankruptcies' values by the number of their credit problems
         for i in data4[data4['Bankruptcies'].isnull()].index:
             data4['Bankruptcies'][i] = data['Bankruptcies'][data4['Number of Credit Problems']
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: Deprecatio
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  This is separate from the ipykernel package so we can avoid doing imports until
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [22]: #Filling in missing 'Tax Liens' values by the number of their credit problems
         for i in data4[data4['Tax Liens'].isnull()].index:
             data4['Tax Liens'][i] = data4['Tax Liens'][data4['Number of Credit Problems'] == 
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: Deprecatio
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  This is separate from the ipykernel package so we can avoid doing imports until
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [23]: #Each Loan is suppose to be unique, therefore duplicates are dropped
         data5 = data4.drop_duplicates(['Loan ID'], keep = 'last')
```

```
In [24]: #Max credit score an individual can have is 800 based off of this particular credit s
         #It appears an extra zero was added to these credit scores
         data5['Credit Score'][data5['Credit Score'] > 800].head(10)
```

```
Out[24]: 341     6600.0
         349     6760.0
         420     7460.0
         522     7320.0
         623     7270.0
         846     6690.0
         926     7230.0
         1240    7380.0
         1306    7440.0
         1317    7390.0
         Name: Credit Score, dtype: float64
```

```
In [25]: #We divide these values by ten to get rid of the extra zero
         #This is done through list comprehension
         data5['Credit Score'][data5['Credit Score'] > 800] = [i/10 for i in data5['Credit Sco
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py:7620: Settin
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self._update_inplace(new_data)
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:29
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
In [27]: #There are now no more values above 800
         #Distribution of credit score
         data5['Credit Score'].describe()
```

```
Out[27]: count    167406.000000
         mean        723.028022
```

```
        std          26.648780
        min         585.000000
        25%         713.000000
        50%         732.000000
        75%         742.000000
        max         751.000000
        Name: Credit Score, dtype: float64
```

```
In [28]: #Some individuals had extremely high loan amounts yet low income
         #Delete outliers in 'Current Loan Amount'
         data6 = data5.drop(data5[data5['Current Loan Amount'] > 1000000].index)
```

```
In [29]: #Next 'Years in current job' is made purely numeric
         data6['Years in current job'].head()
```

```
Out[29]: 0    10+ years
         1      4 years
         2    10+ years
         3    10+ years
         4      4 years
         Name: Years in current job, dtype: object
```

```
In [30]: data6['Years in current job'] = [str(i) for i in data6['Years in current job']]
```

```
In [31]: h = [i.split(' ')[0] for i in data6['Years in current job']]
```

```
In [32]: h = [i.replace('<', '.5') for i in h]
         h = [i.replace('n/a', '0') for i in h]
         h = [i.replace('10+', '10') for i in h]
         h = [float(i) for i in h]
```

```
In [33]: data6['Years in current job'] = h
```

```
In [34]: hh = list(data6['Maximum Open Credit'])
```

```
In [140]: #Now 'Maximum Open Credit is cleaned by getting rid of strings
          hh = [str(i).replace('#VALUE!', '0') for i in hh]
          hh = [float(i) for i in hh]
          data6['Maximum Open Credit'] = hh
```

```
In [141]: data6['Maximum Open Credit'] = [float(i) for i in data6['Maximum Open Credit']]
```

```
In [36]: #Everytime 'Annual Income is null, so is 'Credit Score'
         #These null rows are deleted
         data[data['Annual Income'].isnull()].head(3)
```

```
Out[36]:                                   Loan ID  \
         7   0000afa6-8902-4f8f-b870-25a8fdad0aeb
         8   00011dfc-31c1-4178-932a-fbeb3f341efb
         12  00029f9f-0cc5-4d4e-aabc-ea4a7fe74e12
```

```
                             Customer ID  Loan Status  Current Loan Amount  \
7   e49c1a82-a0f7-45e8-9f46-2f75c43f9fbc            0                24613   
8   ef6e098c-6c83-4752-8d00-ff793e476b8c            1                10036   
12  afbc2fa3-3bad-4d48-b691-829aed78bad5            0                17980   

          Term  Credit Score Years in current job Home Ownership  \
7    Long Term           NaN              6 years           Rent   
8   Short Term           NaN              5 years           Rent   
12  Short Term           NaN             < 1 year       Own Home   

    Annual Income              Purpose  Monthly Debt  Years of Credit History  \
7             NaN        Business Loan        542.29                     17.6   
8             NaN   Debt Consolidation        386.36                     17.7   
12            NaN   Debt Consolidation        597.50                      9.9   

    Months since last delinquent  Number of Open Accounts  \
7                           73.0                        7   
8                            NaN                        7   
12                          43.0                        7   

    Number of Credit Problems  Current Credit Balance Maximum Open Credit  \
7                           0                   14123               16954   
8                           0                   11970               16579   
12                          0                    6817               22800   

    Bankruptcies  Tax Liens  
7            0.0        0.0  
8            0.0        0.0  
12           0.0        0.0  
```

```
In [192]: data7 = data6
          del data7['Months since last delinquent']
          data8 = data7.drop(data7[data7['Annual Income'].isnull()].index)
```

```
In [194]: del data8['Loan ID']
          del data8['Customer ID']
```

```
In [196]: #Predictions are created for 'Loan Status' aka whether or not the loan was paid off
          #'Loan Status' is categorical, calls for a classification algorithm
          from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
          from sklearn.ensemble import GradientBoostingClassifier
          gbc = GradientBoostingClassifier()
```

```
In [ ]: gbc.fit(data8.drop('Loan Status', axis = 1), data8['Loan Status'])
```

```
In [770]: #75% accuracy score means
          accuracy_score(data8['Loan Status'], gbc.predict(data8.drop('Loan Status', axis = 1)
Out[770]: 0.75739054131743777
```

```
In [779]: gbc.feature_importances_
```

```
Out[779]: array([ 0.10438967,  0.18943464,  0.04284989,  0.18065884,  0.10127577,
                  0.0714165 ,  0.03295017,  0.00468396,  0.07069356,  0.09599793,
                  0.01642792,  0.00726124,  0.01304626,  0.01581257,  0.00846362,
                  0.        ,  0.02838365,  0.        ,  0.0027543 ,  0.00769752,
                  0.00249725,  0.00094808,  0.        ,  0.00115457,  0.        ,
                  0.0012021 ])
```