FEATURE SELECTION ON LENDING CLUB LOAN DATASET WITH PYTHON

WHAT FACTORS AFFECT LOAN AMOUNT?

DR. ALVIN ANG

CONTENTS

I. STEP 1: READING IN THE DATA

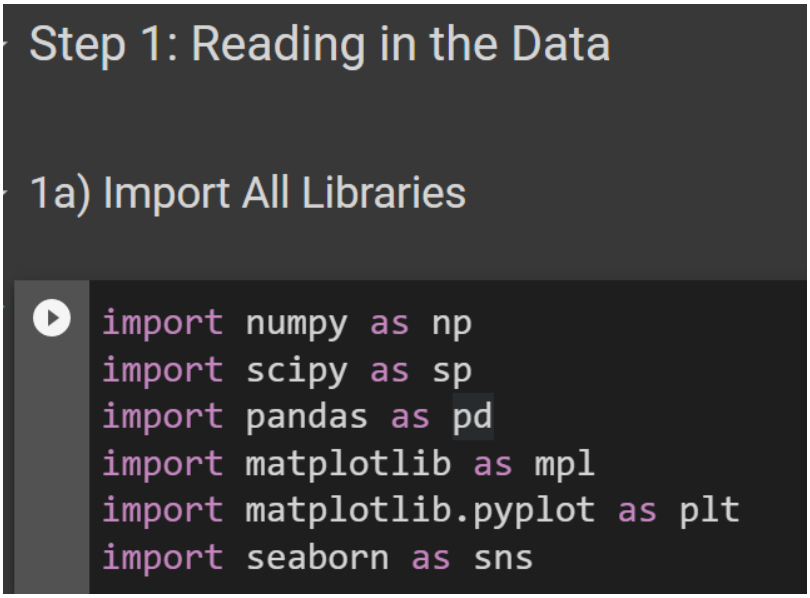
IPYNB:

- https://www.alvinang.sg/s/Feature_Selection_on_Lending_Club_Loan_Dataset_by_Dr_Alvin_Ang.ipynb

FILES:

- https://www.alvinang.sg/s/LendingClubLoan200-rows.csv
- https://www.alvinang.sg/s/LCDataDictionary.xlsx

A. IMPORT ALL LIBRARIES



B. SETTING UP OPTIONS

1b) Setting Up Options

```
#Pandas Options
pd.set_option('display.max_colwidth', 1000,
              'display.max_rows', None,
              'display.max_columns', None)
#preventing the dataframe frrom displaying
#too long a column by setting the
#max_colwidth to 1000

#Plotting Options
%matplotlib inline
mpl.style.use('ggplot')
sns.set(style='whitegrid')
```

C. BROWSING THE COLUMNS

1c) Browsing the Columns

```
[226] loans = pd.read_csv('https://www.alvinang.sg/s/LendingClubLoan200-rows.csv')

[227] loans.info()

    #there are 73 columns! TOO MANY!
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 74 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          199 non-null    int64
 1   member_id                   199 non-null    int64
 2   loan_amnt                   199 non-null    int64
 3   funded_amnt                 199 non-null    int64
 4   funded_amnt_inv             199 non-null    float64
 5   term                        199 non-null    object
 6   int_rate                    199 non-null    float64
 7   installment                 199 non-null    float64
 8   grade                       199 non-null    object
 9   sub_grade                   199 non-null    object
 10  emp_title                   190 non-null    object
 11  emp_length                  198 non-null    object
 12  home_ownership              199 non-null    object
 13  annual_inc                  199 non-null    float64
 14  verification_status         199 non-null    object
 15  issue_d                     199 non-null    object
 16  loan_status                 199 non-null    object
 17  pymnt_plan                  199 non-null    object
 18  url                         199 non-null    object
 19  desc                        129 non-null    object
 20  purpose                     199 non-null    object
 21  title                       199 non-null    object
 22  zip_code                    199 non-null    object
 23  addr_state                  199 non-null    object
 24  dti                         199 non-null    object
 25  delinq_2yrs                 199 non-null    float64
 26  earliest_cr_line            199 non-null    object
 27  inq_last_6mths              199 non-null    object
 28  mths_since_last_delinq      47 non-null     float64
 29  mths_since_last_record      5 non-null      float64
 30  open_acc                    198 non-null    float64
 31  pub_rec                     199 non-null    int64
 32  revol_bal                   199 non-null    int64
 33  revol_util                  199 non-null    float64
 34  total_acc                   199 non-null    float64
 35  initial_list_status         199 non-null    object
 36  out_prncp                   199 non-null    object
 37  out_prncp_inv               199 non-null    float64
 38  total_pymnt                 199 non-null    float64
 39  total_pymnt_inv             199 non-null    float64
 40  total_rec_prncp             199 non-null    float64
 41  total_rec_int               199 non-null    float64
 42  total_rec_late_fee          199 non-null    float64
 43  recoveries                  199 non-null    float64
 44  collection_recovery_fee     199 non-null    float64
 45  last_pymnt_d                198 non-null    object
 46  last_pymnt_amnt             199 non-null    object
 47  next_pymnt_d                14 non-null     object
 48  last_credit_pull_d          198 non-null    object
 49  collections_12_mths_ex_med  199 non-null    object
 50  mths_since_last_major_derog 1 non-null      float64
 51  policy_code                 198 non-null    float64
 52  application_type            199 non-null    object
 53  annual_inc_joint            1 non-null      object
 54  dti_joint                   0 non-null      float64
 55  verification_status_joint   0 non-null      float64
 56  acc_now_delinq              198 non-null    float64
 57  tot_coll_amt                1 non-null      float64
 58  tot_cur_bal                 0 non-null      float64
 59  open_acc_6m                 0 non-null      float64
 60  open_il_6m                  0 non-null      float64
 61  open_il_12m                 0 non-null      float64
 62  open_il_24m                 0 non-null      float64
 63  mths_since_rcnt_il          0 non-null      float64
 64  total_bal_il                0 non-null      float64
 65  il_util                     0 non-null      float64
 66  open_rv_12m                 0 non-null      float64
 67  open_rv_24m                 0 non-null      float64
 68  max_bal_bc                  0 non-null      float64
 69  all_util                    0 non-null      float64
 70  total_rev_hi_lim            0 non-null      float64
 71  inq_fi                      0 non-null      float64
 72  total_cu_tl                 0 non-null      float64
 73  inq_last_12m                0 non-null      float64
dtypes: float64(40), int64(6), object(28)
memory usage: 115.2+ KB
```

A. CREATING THE MISSING FRACTIONS

Step 2: Ridding Columns that have too many NaNs

2a) Creating the Missing Fractions

```
[228] missing_fractions = loans.\
                          isnull().\
                          mean().\
                         sort_values(ascending = False)
```

```
missing_fractions
#so many rows are pure empty!
```

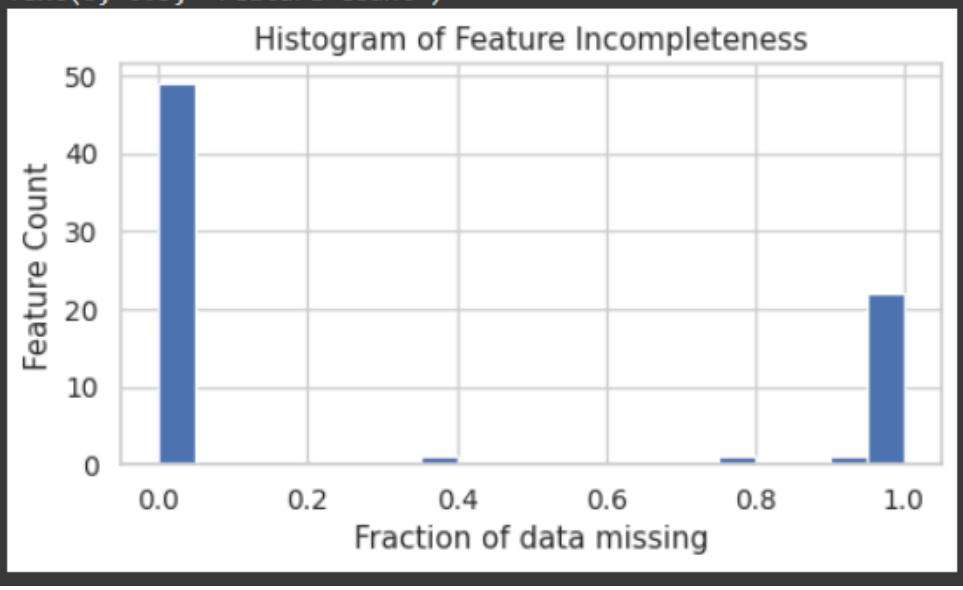
```
inq_last_12m                 1.000000
verification_status_joint    1.000000
tot_cur_bal                  1.000000
open_acc_6m                  1.000000
open_il_6m                   1.000000
open_il_12m                  1.000000
open_il_24m                  1.000000
mths_since_rcnt_il           1.000000
total_bal_il                 1.000000
dti_joint                    1.000000
il_util                      1.000000
open_rv_12m                  1.000000
open_rv_24m                  1.000000
max_bal_bc                   1.000000
all_util                     1.000000
total_rev_hi_lim             1.000000
inq_fi                       1.000000
total_cu_tl                  1.000000
annual_inc_joint             0.994975
mths_since_last_major_derog  0.994975
tot_coll_amt                 0.994975
mths_since_last_record       0.974874
next_pymnt_d                 0.929648
mths_since_last_delinq       0.763819
desc                         0.351759
emp_title                    0.045226
acc_now_delinq               0.005025
last_pymnt_d                 0.005025
last_credit_pull_d           0.005025
open_acc                     0.005025
policy_code                  0.005025
emp_length                   0.005025
total_rec_prncp              0.000000
collections_12_mths_ex_med   0.000000
application_type             0.000000
last_pymnt_amnt              0.000000
collection_recovery_fee      0.000000
recoveries                   0.000000
total_rec_late_fee           0.000000
total_rec_int                0.000000
id                           0.000000
total_pymnt_inv              0.000000
url                          0.000000
loan_amnt                    0.000000
funded_amnt                  0.000000
funded_amnt_inv              0.000000
term                         0.000000
int_rate                     0.000000
installment                  0.000000
grade                        0.000000
sub_grade                    0.000000
home_ownership               0.000000
annual_inc                   0.000000
verification_status          0.000000
issue_d                      0.000000
loan_status                  0.000000
pymnt_plan                   0.000000
purpose                      0.000000
total_pymnt                  0.000000
title                        0.000000
zip_code                     0.000000
addr_state                   0.000000
dti                          0.000000
delinq_2yrs                  0.000000
earliest_cr_line             0.000000
inq_last_6mths               0.000000
pub_rec                      0.000000
revol_bal                    0.000000
revol_util                   0.000000
total_acc                    0.000000
initial_list_status          0.000000
out_prncp                    0.000000
member_id                    0.000000
out_prncp_inv                0.000000
dtype: float64
```

B. PLOTTING THE MISSING FRACTIONS

2b) Plotting the Missing Fractions

```
plt.figure(figsize=(6,3), dpi=90)
missing_fractions.plot.hist(bins=20)
plt.title('Histogram of Feature Incompleteness')
plt.xlabel('Fraction of data missing')
plt.ylabel('Feature Count')
```

```
Text(0, 0.5, 'Feature Count')
```



We see that close to 50 columns are filled with values whlie about 20+ columns have almost 100% NaNs (which means that are literally empty!)

C. CREATING THE DROP LIST

2c) Creating the Drop List

```
[231] drop_list = sorted(list(
          missing_fractions[missing_fractions > 0.3].
          index))
```

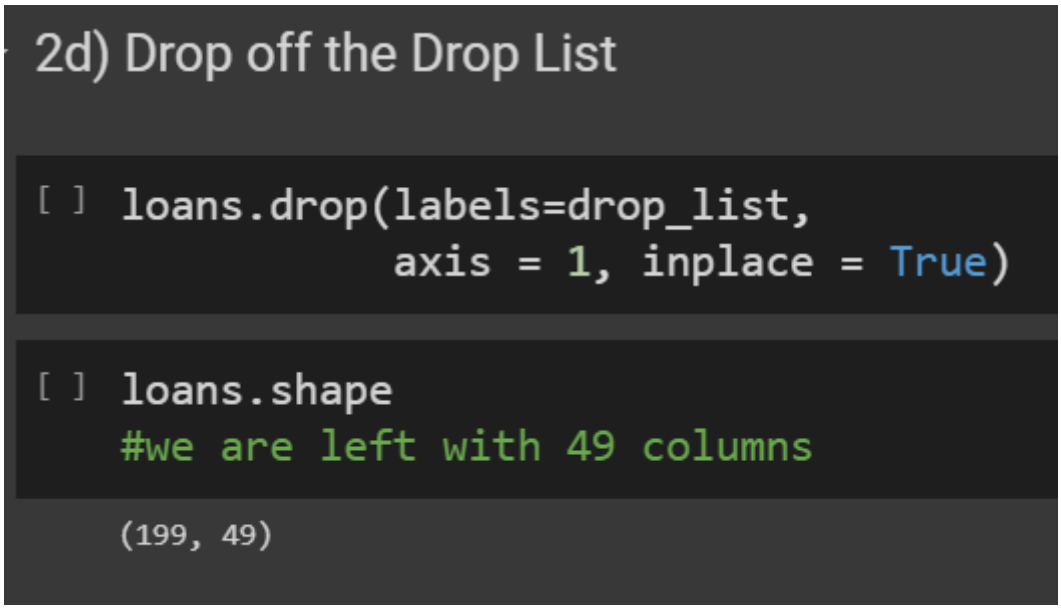
```
import cmd
cli = cmd.Cmd()
cli.columnize(drop_list, displaywidth = 10)
```

```
all_util
annual_inc_joint
desc
dti_joint
il_util
inq_fi
inq_last_12m
max_bal_bc
mths_since_last_delinq
mths_since_last_major_derog
mths_since_last_record
mths_since_rcnt_il
next_pymnt_d
open_acc_6m
open_il_12m
open_il_24m
open_il_6m
open_rv_12m
open_rv_24m
tot_coll_amt
tot_cur_bal
total_bal_il
total_cu_tl
total_rev_hi_lim
verification_status_joint
```

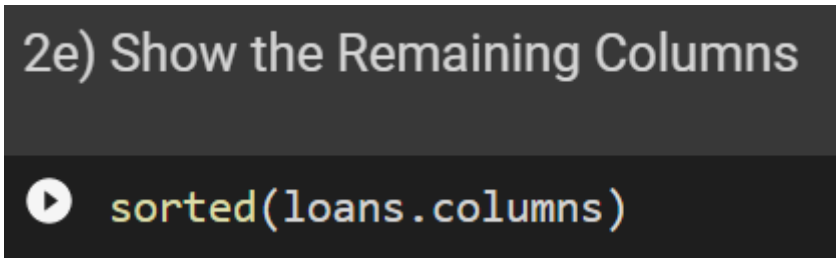
```
[233] len(drop_list)
      #we need to drop off 25 columns because they have
      #too many NaNs
```

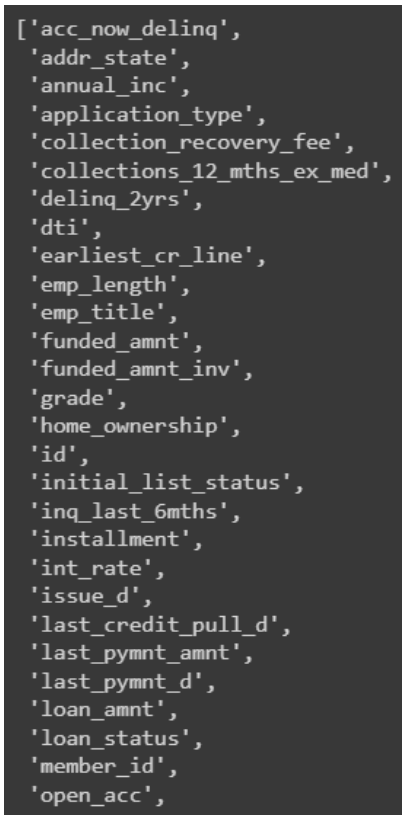
```
25
```

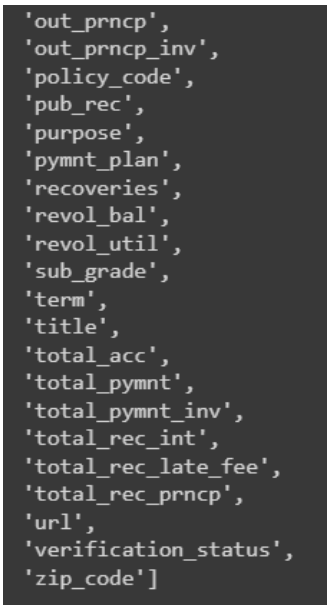
D. DROP OFF THE DROP LIST



E. SHOW THE REMAINING COLUMNS







III. STEP 3: SELECTING IMPORTANT COLUMNS OUT OF THE REMAINING COLUMNS

A. TAKING A PEEK AT THE LENDING CLUB LOAN DICTIONARY

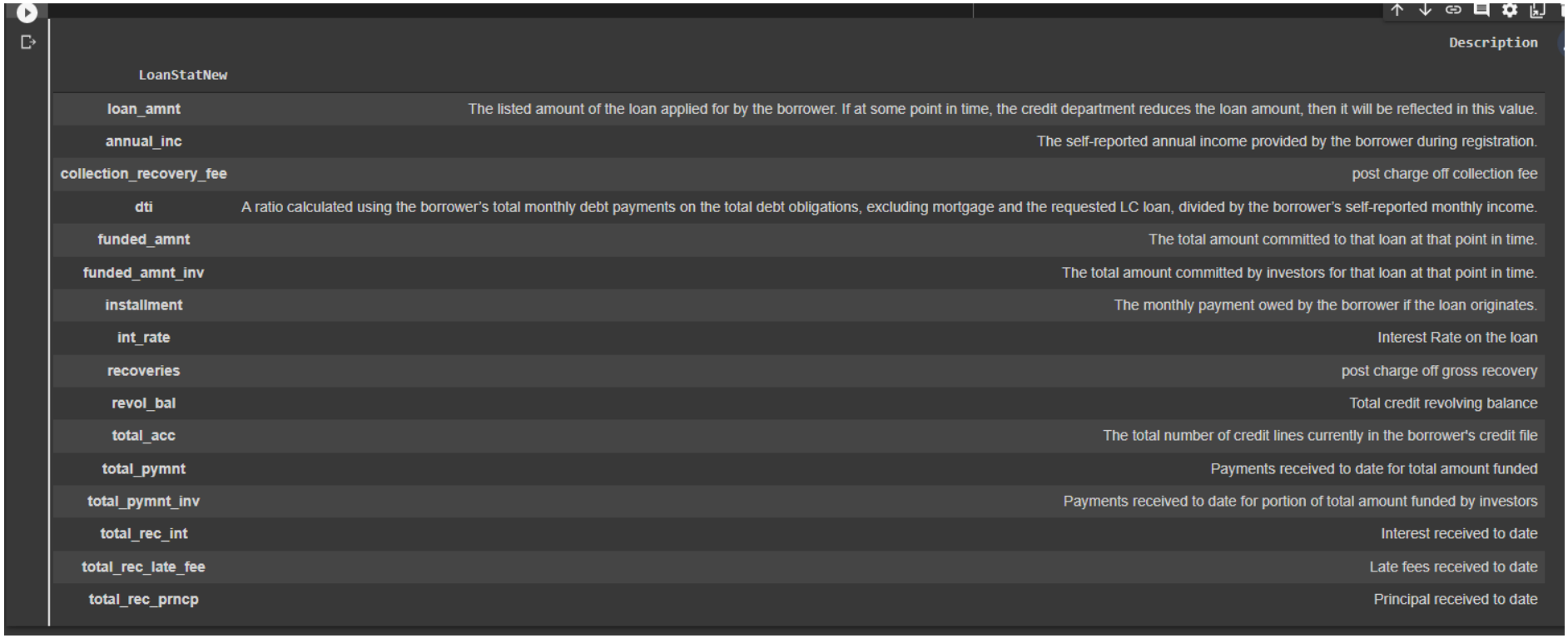
Step 3: Selecting Important Columns out of the Remaining Columns

3a) Taking a peek at the Lending Club Loan Dictionary

```
xls = pd.read_excel('https://www.alvinang.sg/s/LCDataDictionary.xlsx',
                    sheet_name = 'LoanStats',
                    index_col = 'LoanStatNew')
```

```
xls.loc[['loan_amnt',
         'annual_inc',
         'collection_recovery_fee',
         'dti',
         'funded_amnt',
         'funded_amnt_inv',
         'installment',
         'int_rate',
         'recoveries',
         'revol_bal',
         'total_acc',
         'total_pymnt',
         'total_pymnt_inv',
         'total_rec_int',
         'total_rec_late_fee',
         'total_rec_prncp']]

#from the LCL Dictionary, we see the Financial Definitions of the Column Names
#we choose the Columns above because we feel that they impact
#Loan Amount the most.
```



LoanStatNew	Description
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
annual_inc	The self-reported annual income provided by the borrower during registration.
collection_recovery_fee	post charge off collection fee
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
funded_amnt	The total amount committed to that loan at that point in time.
funded_amnt_inv	The total amount committed by investors for that loan at that point in time.
installment	The monthly payment owed by the borrower if the loan originates.
int_rate	Interest Rate on the loan
recoveries	post charge off gross recovery
revol_bal	Total credit revolving balance
total_acc	The total number of credit lines currently in the borrower's credit file
total_pymnt	Payments received to date for total amount funded
total_pymnt_inv	Payments received to date for portion of total amount funded by investors
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
total_rec_prncp	Principal received to date

B. CREATING A NEW KEEP LIST

3b) Creating a New Keep List

```
keep_list = ['loan_amnt',
             'annual_inc',
             'collection_recovery_fee',
             'dti',
             'funded_amnt',
             'funded_amnt_inv',
             'installment',
             'int_rate',
             'recoveries',
             'revol_bal',
             'total_acc',
             'total_pymnt',
             'total_pymnt_inv',
             'total_rec_int',
             'total_rec_late_fee',
             'total_rec_prncp']

len(keep_list)
#we will keep 16 columns out of the 49 remaining
```

```
16
```

C. DROP OFF THE UNIMPORTANT / UNWANTED COLUMNS

3c) Drop Off the Unimportant / Unwanted Columns

```
[ ] new_drop_list = [col for col in loans.columns if col not in keep_list]
```

```
[ ] len(new_drop_list)
    #we want to drop off 33 columns
```

```
33
```

```
[ ] loans.drop(labels=new_drop_list, axis = 1, inplace = True)
```

```
sorted(loans.columns)
#this is what we have left after dropping 33 unwanted columns
```

```
['annual_inc',
 'collection_recovery_fee',
 'dti',
 'funded_amnt',
 'funded_amnt_inv',
 'installment',
 'int_rate',
 'loan_amnt',
 'recoveries',
 'revol_bal',
 'total_acc',
 'total_pymnt',
 'total_pymnt_inv',
 'total_rec_int',
 'total_rec_late_fee',
 'total_rec_prncp']
```

```
loans.shape
#we obtain our 16 Wanted columns
```

```
(199, 16)
```

IV. STEP 4: FIND THE TOP 5 IMPORTANT FEATURES USING RANDOM FOREST CLASSIFIER

Step 4: Find the TOP 5 Important Features using Random Forest Classifier

```
X = loans[['annual_inc',
           'collection_recovery_fee',
           'dti',
           'funded_amnt',
           'funded_amnt_inv',
           'installment',
           'int_rate',
           'recoveries',
           'revol_bal',
           'total_acc',
           'total_pymnt',
           'total_pymnt_inv',
           'total_rec_int',
           'total_rec_late_fee',
           'total_rec_prncp']]

#we take all our wanted 15 columns but we leave out
#'loan_amnt' because that belongs to y
```

```
[ ] X_index = pd.DataFrame (X.columns)
```

```
X_index
```

	0
0	annual_inc
1	collection_recovery_fee
2	dti
3	funded_amnt
4	funded_amnt_inv
5	installment
6	int_rate
7	recoveries
8	revol_bal
9	total_acc
10	total_pymnt
11	total_pymnt_inv
12	total_rec_int
13	total_rec_late_fee
14	total_rec_prncp

```
[ ] y = loans['loan_amnt']
```

A. CHECK THAT ALL COLUMNS ARE OF NUMERIC TYPE BEFORE FEEDING INTO THE CLASSIFIER

4a) Check that All Columns are of Numeric Type before Feeding into the Classifier

```
X.dtypes
#we see that the only column of "string" type is 'dti'
```

```
annual_inc                float64
collection_recovery_fee   float64
dti                        object
funded_amnt                 int64
funded_amnt_inv           float64
installment               float64
int_rate                  float64
recoveries                float64
revol_bal                   int64
total_acc                 float64
total_pymnt               float64
total_pymnt_inv           float64
total_rec_int             float64
total_rec_late_fee        float64
total_rec_prncp           float64
dtype: object
```

```
X['dti'] = X['dti'].astype(float)
#we need to convert 'dti' column to 'float' in order to feed the
#Random Forest Classifier

#however, there seems to an error pop up
#where a word 'CA' is found in one of the rows.....
#(we dunno which row nor which column!)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-250-ab220a19c04e> in <module>()
----> 1 X['dti'] = X['dti'].astype(float)
      2 #we need to convert 'dti' column to 'float' in order to feed the
      3 #Random Forest Classifier
      4
      5 #however, there seems to an error pop up

6 frames
/usr/local/lib/python3.7/dist-packages/pandas/core/dtypes/cast.py in astype_nansafe(arr, dtype, copy, skipna)
   1199     if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
   1200         # Explicit copy, or required since NumPy can't view from / to object.
-> 1201         return arr.astype(dtype, copy=True)
   1202
   1203     return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: 'CA'

SEARCH STACK OVERFLOW
```

```
[ ] X.loc[X['dti'] =='CA']
#Row 36 is dirty....
```

	annual_inc	collection_recovery_fee	dti	funded_amnt	funded_amnt_inv	installm
36	55596.0	0.0	CA	10800	10800.0	348

4b) Cleanse Characters into Numbers

```
[ ] X['dti'].iloc[34: 38]
    #we check the surrounding row numbers values...
```

```
34    6.35
35    11.8
36      CA
37    8.48
Name: dti, dtype: object
```

```
X['dti'].iloc[36]
```

```
'CA'
```

```
X['dti'].iloc[36] = 10
#since 'dti' row 36 is 'CA' and we need to change to a number
#we decide upon 10 because its between 11.8 and 8.48
```

we assign the number 10 into row 36 for column 'dti'...

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

```
[ ] X['dti'].iloc[36]
```

```
10
```

```
[ ] X['dti'] = X['dti'].astype(float)
    #we need to convert 'dti' column to 'float' in order to feed the
    #Random Forest Classifier
```

finally this code runs without any problems!!

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
X.dtypes
#Finally! all columns are now numbers!
```

```
annual_inc                float64
collection_recovery_fee   float64
dti                       float64
funded_amnt                 int64
funded_amnt_inv           float64
installment               float64
int_rate                  float64
recoveries                float64
revol_bal                   int64
total_acc                 float64
total_pymnt               float64
total_pymnt_inv           float64
total_rec_int             float64
total_rec_late_fee        float64
total_rec_prncp           float64
dtype: object
```

C. CREATE RANDOM FOREST CLASSIFIER

4c) Create Random Forest Classifier

```
[ ] import sklearn
    from sklearn.ensemble import RandomForestClassifier

    clf = RandomForestClassifier(n_estimators = 100)
    #n_estimators stands for the no. of trees in the forest
    #we use 100 trees here

    clf.fit(X, y)
```

```
RandomForestClassifier()
```

D. FIND IMPORTANT FEATURES

4d) Find Important Features

```
[ ] clf.feature_importances_
```

```
array([0.06060857, 0.01142957, 0.04984586, 0.15068175, 0.14645094,
       0.10291792, 0.0456518 , 0.0133651 , 0.04882682, 0.04138062,
       0.08186602, 0.0870414 , 0.05942668, 0.00573743, 0.09476953])
```

```
[ ] feature_imp = pd.Series(clf.feature_importances_, index = X_index).sort_values(ascending = False)
```

```
feature_imp
```

```
(funded_amnt,)              0.150682
(funded_amnt_inv,)          0.146451
(installment,)              0.102918
(total_rec_prncp,)          0.094770
(total_pymnt_inv,)          0.087041
(total_pymnt,)              0.081866
(annual_inc,)               0.060609
(total_rec_int,)            0.059427
(dti,)                      0.049846
(revol_bal,)                0.048827
(int_rate,)                 0.045652
(total_acc,)                0.041381
(recoveries,)               0.013365
(collection_recovery_fee,)  0.011430
(total_rec_late_fee,)       0.005737
dtype: float64
```

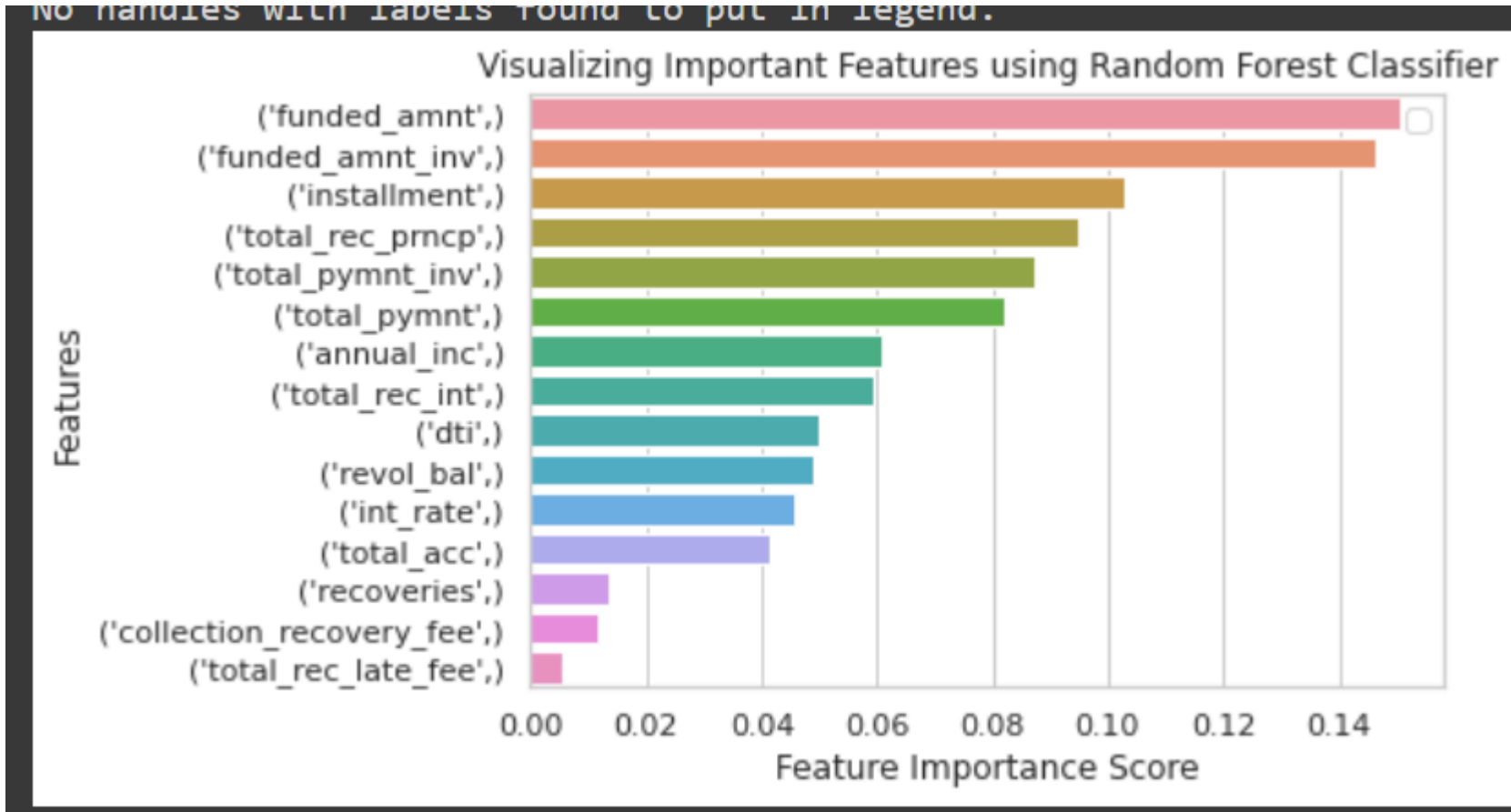
4e) Plot Important Features

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#Creating a bar plot
sns.barplot(x = feature_imp, y = feature_imp.index)

#Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title('Visualizing Important Features using Random Forest Classifier')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



From the Random Forest Classifier, we see that the TOP 5 important columns are:

- Funded Amount
- Funded Amount Inventory
- Installment
- Total Received Principle
- Total Payment

Step 5: Cross check the TOP 5 Important Features using Multiple Regression

5a) Create Multiple Regression Model

```
[ ] from statsmodels.api import OLS
    OLS(y,X).fit().summary()
```

OLS Regression Results

Dep. Variable:	loan_amnt	R-squared (uncentered):	0.986
Model:	OLS	Adj. R-squared (uncentered):	0.985
Method:	Least Squares	F-statistic:	856.5
Date:	Sun, 05 Jun 2022	Prob (F-statistic):	2.24e-161
Time:	08:28:11	Log-Likelihood:	-1748.4
No. Observations:	199	AIC:	3527.
Df Residuals:	184	BIC:	3576.
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	0.025	0.975]
annual_inc	0013	0.005	0.26	0.794	-.009	0.011
collection_recovery_fee	4032	5.495	0.98	0.327	-6.437	16.244
dti	-0.6412	20.285	-1.0	0.310	-60.662	19.379
funded_amnt	3663	0.294	4.64	0.000	.786	1.946
funded_amnt_inv	-.3063	0.284	-1.0	0.282	-.866	0.254
installment	-.2592	2.817	-1.8	0.064	-10.818	0.299
int_rate	4.8507	28.326	1.23	0.220	-21.034	90.735
recoveries	4365	1.566	0.91	0.360	-1.652	4.525
revol_bal	0059	0.013	0.44	0.660	-.021	0.032
total_acc	8305	14.041	0.13	0.896	-25.871	29.532
total_pymnt	-.5189	1.258	-1.2	0.229	-4.001	0.964
total_pymnt_inv	4512	0.248	1.81	0.070	-.038	0.940
total_rec_int	1946	1.290	0.92	0.356	-1.351	3.740
total_rec_late_fee	-3.1294	21.396	-1.0	0.281	-65.342	19.083
total_rec_prncp	1475	1.272	0.90	0.368	-1.362	3.657

Omnibus:	227.998	Durbin-Watson:	2.254
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8010.921
Skew:	4.662	Prob(JB):	0.00
Kurtosis:	32.651	Cond. No.	1.97e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.97e+04. This might indicate that there are strong multicollinearity or other numerical problems.

5b) Create a Dataframe to Store P-Values

```
import pandas as pd
# list of strings
lst1 = ['annual_inc',
        'collection_recovery_fee',
        'dti',
        'funded_amnt',
        'funded_amnt_inv',
        'installment',
        'int_rate',
        'recoveries',
        'revol_bal',
        'total_acc',
        'total_pymnt',
        'total_pymnt_inv',
        'total_rec_int',
        'total_rec_late_fee',
        'total_rec_prncp']
# list of int
lst2 = [0.794, 0.327, 0.31, 0, 0.282, 0.064, 0.22, 0.36, 0.66, 0.896, 0.229, 0.07, 0.356, 0.281, 0.368]
# Calling DataFrame after zipping both lists, with columns specified
df = pd.DataFrame(list(zip(lst1, lst2)), columns =['X', 'P-value'])
```

```
result = df.sort_values(by="P-value", ascending=True)
result
```

	X	P-value
3	funded_amnt	0.000
5	installment	0.064
11	total_pymnt_inv	0.070
6	int_rate	0.220
10	total_pymnt	0.229
13	total_rec_late_fee	0.281
4	funded_amnt_inv	0.282
2	dti	0.310
1	collection_recovery_fee	0.327
12	total_rec_int	0.356
7	recoveries	0.360
14	total_rec_prncp	0.368
8	revol_bal	0.660
0	annual_inc	0.794
9	total_acc	0.896

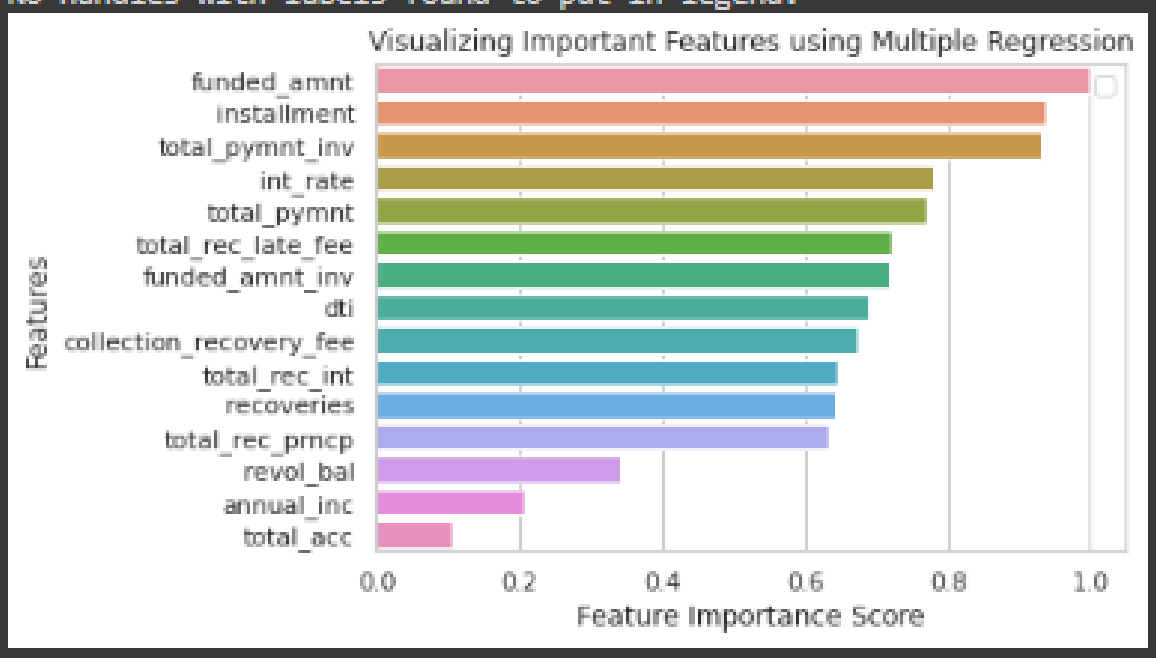
5c) Plot Important Features

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#Creating a bar plot
sns.barplot(x = 1 - result['P-value'], y = result['X'])

#Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title('Visualizing Important Features using Multiple Regression')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



From the Mutliple Regression Model, we see that the TOP 5 important columns are:

- Funded Amount
- Installment
- Total Payment Inventory
- Interest Rate
- Total Payment

Step 6: Fish Out Those Important Columns that are Shared Between "Random Forest Classifier" and "Multiple Regression"

From the Random Forest Classifier, we see that the TOP 5 important columns are:

- Funded Amount
- Funded Amount Inventory
- Installment
- Total Received Principle
- Total Payment

From the Mutliple Regression Model, we see that the TOP 5 important columns are:

- Funded Amount
- Installment
- Total Payment Inventory
- Interest Rate
- Total Payment

```
#The shared TOP Columns are:
#No. 1 --> Funded Amount
#No. 2 --> Installment
#No. 3 --> Total Payment
```

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.