LendingClubLoanPrediction

January 27, 2022

LendingClub is a US peer-to-peer lending company, headquartered in San Francisco, California. It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC), and to offer loan trading on a secondary market. Lending Club operates an online lending platform that enables borrowers to obtain a loan, and investors to purchase notes backed by payments made on loans.

In this project, I will use supervised learning models to identify whether the LC loan will be default in the future and make a prediction of their loan interest rate.

1 Part 0: Setup Google Drive Environment and Load Data

[540]:
```
import pandas as pd
import numpy as np

from tabulate import tabulate

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
```

```
from sklearn import model_selection
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix
```

[478]:
```
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

[479]:
```
link = 'https://drive.google.com/open?id=1VOAhrB5eIhFc8C5qeFx5QYa2OLD7s1eV'
fluff, id = link.split('=')
file = drive.CreateFile({'id':id})
file.GetContentFile('loan-clean-version.csv')
```

load data

[480]:
```
LC_df  = pd.read_csv('loan-clean-version.csv')
LC_df.head()
```

[480]:
```
        id  loan_amnt  funded_amnt  …  total_rec_prncp total_rec_int
loan_status
0  1077501       5000         5000  …          5000.00        861.07   Fully
Paid
1  1077430       2500         2500  …           456.46        435.17  Charged
Off
2  1077175       2400         2400  …          2400.00        603.65   Fully
Paid
3  1076863      10000        10000  …         10000.00       2209.33   Fully
Paid
4  1075269       5000         5000  …          5000.00        631.38   Fully
Paid

[5 rows x 29 columns]
```

2 Part 1: Data Exploration

3 1.1 Raw Data

Get to know the raw data by reading data dictionary

##full description of each fields:

LoanStatNew	Description
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.

LoanStatNew	Description
addr_state	The state provided by the borrower in the loan application
annual_inc	The annual income provided by the borrower during registration.
collection_recovery_fee	post charge off collection fee
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
desc	Loan description provided by the borrower
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.
earliest_cr_line	The month the borrower's earliest reported credit line was opened
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
emp_title	The job title supplied by the Borrower when applying for the loan.
fico_range_high	The upper boundary of range the borrower's FICO belongs to.
fico_range_low	The lower boundary of range the borrower's FICO belongs to.
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.
grade	LC assigned loan grade
home_ownership	The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER.
id	A unique LC assigned ID for the loan listing.
initial_list_status	The initial listing status of the loan. Possible values are – W, F
inq_last_6mths	The number of inquiries by creditors during the past 6 months.
installment	The monthly payment owed by the borrower if the loan originates.
int_rate	Interest Rate on the loan
is_inc_v	Indicates if income was verified by LC, not verified, or if the income source was verified
issue_d	The month which the loan was funded

LoanStatNew	Description
last_credit_pull_d	The most recent month LC pulled credit for this loan
last_fico_range_high	The last upper boundary of range the borrower's FICO belongs to pulled.
last_fico_range_low	The last lower boundary of range the borrower's FICO belongs to pulled.
last_pymnt_amnt	Last total payment amount received
last_pymnt_d	Last month payment was received
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.
loan_status	Current status of the loan
member_id	A unique LC assigned Id for the borrower member.
mths_since_last_delinq	The number of months since the borrower's last delinquency.
mths_since_last_major_derog	Months since most recent 90-day or worse rating
mths_since_last_record	The number of months since the last public record.
next_pymnt_d	Next scheduled payment date
open_acc	The number of open credit lines in the borrower's credit file.
out_prncp	Remaining outstanding principal for total amount funded
out_prncp_inv	Remaining outstanding principal for portion of total amount funded by investors
policy_code	Publicly available policy_code=1, new products not publicly available policy_code=2
pub_rec	Number of derogatory public records
purpose	A category provided by the borrower for the loan request.
pymnt_plan	Indicates if a payment plan has been put in place for the loan
recoveries	post charge off gross recovery
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
sub_grade	LC assigned loan subgrade
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
title	The loan title provided by the borrower

LoanStatNew	Description
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
url	URL for the LC page with listing data.

[481]:
```
LC_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9004 entries, 0 to 9003
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   9004 non-null   int64
 1   loan_amnt            9004 non-null   int64
 2   funded_amnt          9004 non-null   int64
 3   funded_amnt_inv      9004 non-null   float64
 4   term                 9004 non-null   object
 5   int_rate             9004 non-null   float64
 6   installment          9004 non-null   float64
 7   grade                9004 non-null   object
 8   emp_length           8688 non-null   object
 9   home_ownership       9004 non-null   object
 10  annual_inc           9004 non-null   float64
 11  verification_status  9004 non-null   object
 12  purpose              9004 non-null   object
 13  addr_state           9004 non-null   object
 14  dti                  9004 non-null   float64
 15  earliest_cr_line     9004 non-null   int64
 16  inq_last_6mths       9004 non-null   int64
 17  open_acc             9004 non-null   int64
 18  pub_rec              9004 non-null   int64
 19  revol_bal            9004 non-null   int64
 20  revol_util           9001 non-null   float64
 21  total_acc            9004 non-null   int64
 22  out_prncp            9004 non-null   int64
 23  out_prncp_inv        9004 non-null   int64
 24  total_pymnt          9004 non-null   float64
 25  total_pymnt_inv      9004 non-null   float64
 26  total_rec_prncp      9004 non-null   float64
```

```
 27  total_rec_int        9004 non-null   float64
 28  loan_status          9004 non-null   object
dtypes: float64(10), int64(11), object(8)
memory usage: 2.0+ MB
```

There are 9k data, emp_length has over 300 null data, 7 non-numerical features, loan_status is the target variable.

[482]:
```
LC_df.isnull().sum()
```

[482]:
```
id                       0
loan_amnt                0
funded_amnt              0
funded_amnt_inv          0
term                     0
int_rate                 0
installment              0
grade                    0
emp_length             316
home_ownership           0
annual_inc               0
verification_status      0
purpose                  0
addr_state               0
dti                      0
earliest_cr_line         0
inq_last_6mths           0
open_acc                 0
pub_rec                  0
revol_bal                0
revol_util               3
total_acc                0
out_prncp                0
out_prncp_inv            0
total_pymnt              0
total_pymnt_inv          0
total_rec_prncp          0
total_rec_int            0
loan_status              0
dtype: int64
```

[483]:
```
LC_df.nunique()
```

[483]:
```
id                     9004
loan_amnt               604
funded_amnt             681
funded_amnt_inv        1234
term                      2
```

```
int_rate                 70
installment            3871
grade                     7
emp_length               11
home_ownership            3
annual_inc             1555
verification_status       3
purpose                  13
addr_state               45
dti                    2559
earliest_cr_line        458
inq_last_6mths            9
open_acc                 33
pub_rec                   3
revol_bal              7573
revol_util             1023
total_acc                63
out_prncp                 1
out_prncp_inv             1
total_pymnt            8962
total_pymnt_inv        8942
total_rec_prncp        2199
total_rec_int          8838
loan_status               2
dtype: int64
```

4 1.2 Missing data

missing value emp_length: annual_inc emp_length replace null; 3 empty revol_util can just be deleted

[484]:
```
LC_df_nonull = LC_df.dropna(subset=["revol_util"])
```

[485]:
```
LC_df_nonull.isnull().sum()
```

[485]:
```
id                       0
loan_amnt                0
funded_amnt              0
funded_amnt_inv          0
term                     0
int_rate                 0
installment              0
grade                    0
emp_length             315
home_ownership           0
annual_inc               0
```

```
verification_status      0
purpose                  0
addr_state               0
dti                      0
earliest_cr_line         0
inq_last_6mths           0
open_acc                 0
pub_rec                  0
revol_bal                0
revol_util               0
total_acc                0
out_prncp                0
out_prncp_inv            0
total_pymnt              0
total_pymnt_inv          0
total_rec_prncp          0
total_rec_int            0
loan_status              0
dtype: int64
```

[486]:
```
min(LC_df_nonull["annual_inc"]), max(LC_df_nonull["annual_inc"]),␣
↪LC_df_nonull["annual_inc"].nunique()
```

[486]:
```
(6000.0, 1782000.0, 1554)
```

max-min /#unique annual——inc 1554 bucket

[487]:
```
LC_df_nonull["annualInc_bucket"] = pd.cut(LC_df_nonull["annual_inc"],␣
↪LC_df_nonull["annual_inc"].nunique(), precision=0)
```

```
/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
```

[488]:
```
LC_df_nonull.head()
```

[488]:

	id	loan_amnt	…	loan_status	annualInc_bucket
0	1077501	5000	…	Fully Paid	(23143.0, 24286.0]
1	1077430	2500	…	Charged Off	(28857.0, 30000.0]
2	1077175	2400	…	Fully Paid	(11714.0, 12857.0]

```
3  1076863      10000  …   Fully Paid  (48286.0, 49429.0]
4  1075269       5000  …   Fully Paid  (35714.0, 36857.0]

[5 rows x 30 columns]
```

annual incom bucket groupby emp length mean mean fill emp length na

[489]:
```
LC_df_nonull["emp_length"] = LC_df_nonull["emp_length"].astype('str')
```

```
/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
```

[490]:
```
LC_df_nonull["emp_length"] = LC_df_nonull["emp_length"].str.extract('(\d*)')
```

```
/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
```

[491]:
```
LC_df_nonull.emp_length.value_counts()
```

[491]:
```
10    2222
      1160
2      899
3      831
5      780
4      780
1      649
6      629
7      446
8      346
9      259
Name: emp_length, dtype: int64
```

[492]:
```
LC_df_nonull.emp_length
```

[492]:
```
0       10
1
2       10
3       10
4        3
        ..
8999    10
9000     3
9001     2
9002
9003     1
Name: emp_length, Length: 9001, dtype: object
```

[493]:
```
LC_df_nonull.emp_length = pd.to_numeric(LC_df_nonull.emp_length)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:5170:
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
```

[494]:
```
LC_df_nonull.emp_length
```

[494]:
```
0       10.0
1        NaN
2       10.0
3       10.0
4        3.0
        …
8999    10.0
9000     3.0
9001     2.0
9002     NaN
9003     1.0
Name: emp_length, Length: 9001, dtype: float64
```

[495]:
```
LC_df_nonull.emp_length.value_counts()
```

[495]:
```
10.0    2222
2.0      899
```

```
3.0      831
5.0      780
4.0      780
1.0      649
6.0      629
7.0      446
8.0      346
9.0      259
Name: emp_length, dtype: int64
```

[496]:
```
LC_df_nonull["emp_length_helper"] = LC_df_nonull.
↪groupby("annualInc_bucket")["emp_length"].transform(lambda x: x.fillna(x.
↪mean()))
```

```
/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
```

[497]:
```
LC_df_nonull.emp_length_helper = LC_df_nonull.emp_length_helper.round()
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:5170:
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
```

[498]:
```
LC_df_nonull.head()
```

[498]:

	id	loan_amnt	…	annualInc_bucket	emp_length_helper
0	1077501	5000	…	(23143.0, 24286.0]	10.0
1	1077430	2500	…	(28857.0, 30000.0]	5.0
2	1077175	2400	…	(11714.0, 12857.0]	10.0
3	1076863	10000	…	(48286.0, 49429.0]	10.0
4	1075269	5000	…	(35714.0, 36857.0]	3.0

```
[5 rows x 31 columns]
```

[499]:
```
LC_df_nonull.emp_length_helper.isnull().sum()
```

[499]:
```
8
```

[500]:
```
LC_df_nonull["emp_length"] = LC_df_nonull["emp_length"].
↪fillna(LC_df_nonull["emp_length_helper"])
```

```
/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
```

[501]:
```
LC_df_nonull.emp_length.value_counts()
```

[501]:
```
10.0    2225
6.0     1120
5.0     1105
4.0      936
2.0      901
3.0      842
1.0      649
7.0      591
8.0      360
9.0      264
Name: emp_length, dtype: int64
```

[502]:
```
#LC_df_nonull.emp_length = pd.to_numeric(LC_df_nonull.emp_length)
```

[503]:
```
#LC_df_nonull.emp_length = LC_df_nonull.emp_length/100000
```

[504]:
```
#LC_df_nonull.emp_length = LC_df_nonull.emp_length.astype(int)
```

[505]:
```
#LC_df_nonull.emp_length.value_counts()
```

[506]:
```
LC_df_nonull = LC_df_nonull.dropna(subset=["emp_length"])
```

[507]:
```
LC_df_nonull.loan_status.value_counts()
```

[507]:
```
Fully Paid     7478
Charged Off    1515
Name: loan_status, dtype: int64
```

5 1.3 EDA

[508]:
```
sns.set()
```

[509]:
```
numCol = []
for col in LC_df_nonull:
  if (LC_df_nonull[col]).dtype==np.float or (LC_df_nonull[col]).dtype==np.int:
    numCol.append(col)
```

[510]:
```
numCol
```

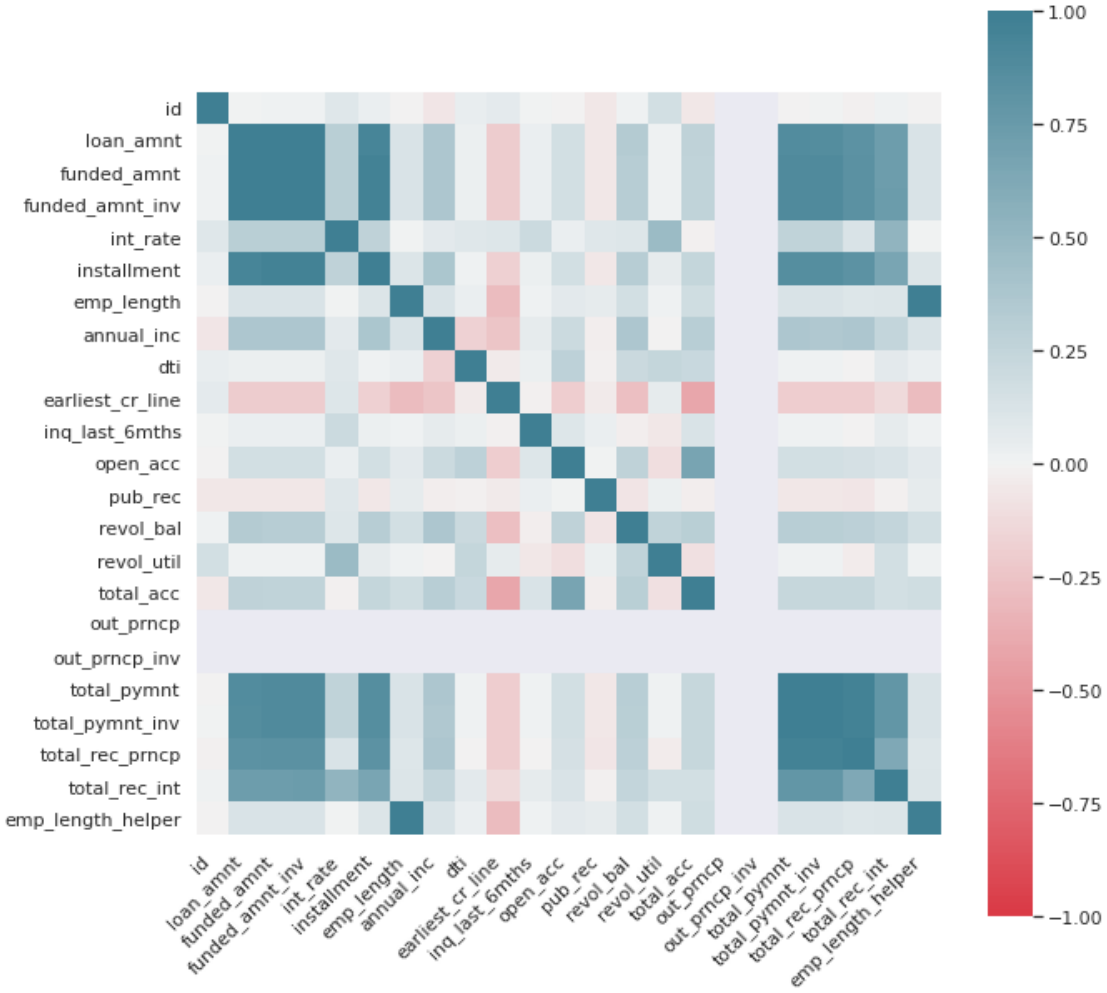
[510]:
```
['id',
 'loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'int_rate',
 'installment',
 'emp_length',
 'annual_inc',
 'dti',
 'earliest_cr_line',
 'inq_last_6mths',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 'out_prncp',
 'out_prncp_inv',
 'total_pymnt',
 'total_pymnt_inv',
 'total_rec_prncp',
 'total_rec_int',
 'emp_length_helper']
```

[511]:
```
len(numCol)
```

[511]:
```
23
```

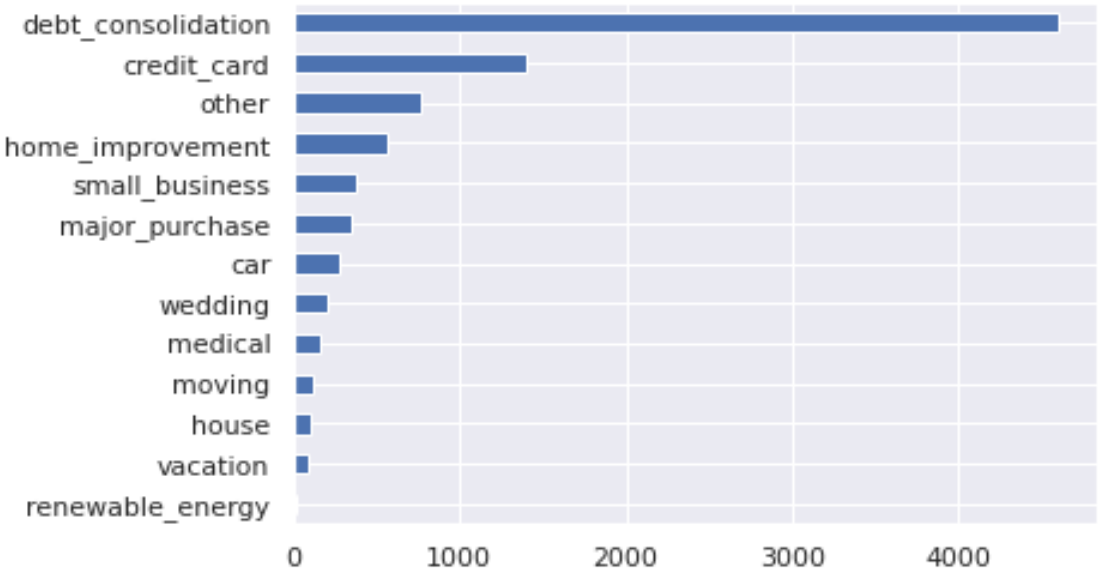
[512]:
```
corr = LC_df_nonull[numCol].corr()
fig, ax = plt.subplots(figsize=(10,10))
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(10, 220, n=200),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right')
```

[512]:
```
[Text(0.5, 0, 'id'),
 Text(1.5, 0, 'loan_amnt'),
 Text(2.5, 0, 'funded_amnt'),
 Text(3.5, 0, 'funded_amnt_inv'),
 Text(4.5, 0, 'int_rate'),
 Text(5.5, 0, 'installment'),
 Text(6.5, 0, 'emp_length'),
 Text(7.5, 0, 'annual_inc'),
 Text(8.5, 0, 'dti'),
 Text(9.5, 0, 'earliest_cr_line'),
 Text(10.5, 0, 'inq_last_6mths'),
 Text(11.5, 0, 'open_acc'),
 Text(12.5, 0, 'pub_rec'),
 Text(13.5, 0, 'revol_bal'),
 Text(14.5, 0, 'revol_util'),
 Text(15.5, 0, 'total_acc'),
 Text(16.5, 0, 'out_prncp'),
 Text(17.5, 0, 'out_prncp_inv'),
 Text(18.5, 0, 'total_pymnt'),
 Text(19.5, 0, 'total_pymnt_inv'),
 Text(20.5, 0, 'total_rec_prncp'),
 Text(21.5, 0, 'total_rec_int'),
 Text(22.5, 0, 'emp_length_helper')]
```



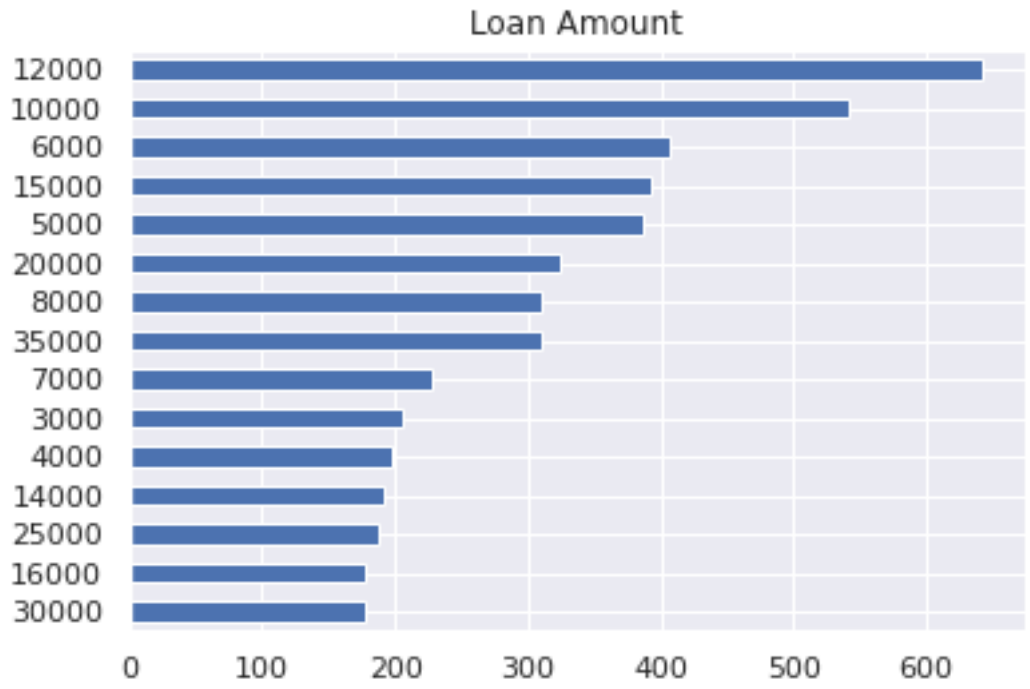
[513]:
```
# explore purpose category
LC_df_nonull["purpose"].value_counts().sort_values().plot(kind = 'barh')
```

[513]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47a463a7d0>



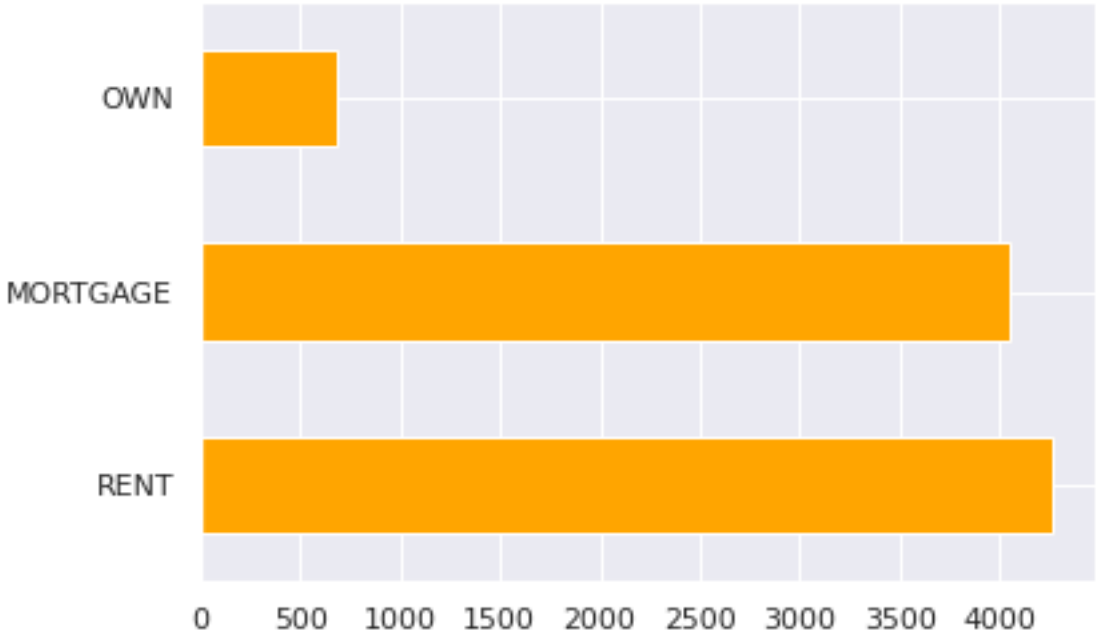
[514]:
```
LC_df_nonull["loan_amnt"].value_counts()[:15].sort_values().plot(kind='barh',␣
↪title="Loan Amount")
```

[514]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47a42a2250>



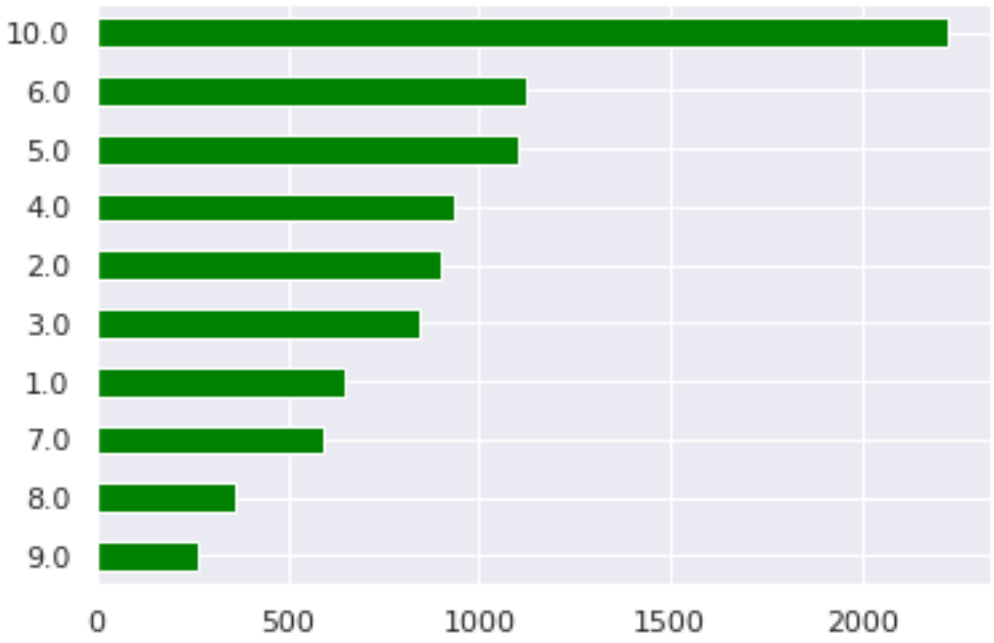
[515]:
```
LC_df_nonull["home_ownership"].value_counts().plot(kind='barh', color='orange')
```

[515]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47a44493d0>



[516]:
```
LC_df_nonull["emp_length"].value_counts().sort_values().plot(kind='barh',␣
↪color='green')
```

[516]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47a43c8d90>



[517]:
```
#total loan amount issued by State
df_location = LC_df_nonull.filter(["addr_state", "loan_amnt"], axis = 1)
df_location = df_location.groupby("addr_state",).sum().reset_index()
df_location = df_location.sort_values("loan_amnt", ascending=False)
#df_location = df_location.sort_values("loan_amnt", ascending=True)
df_location.head()
```

[517]:

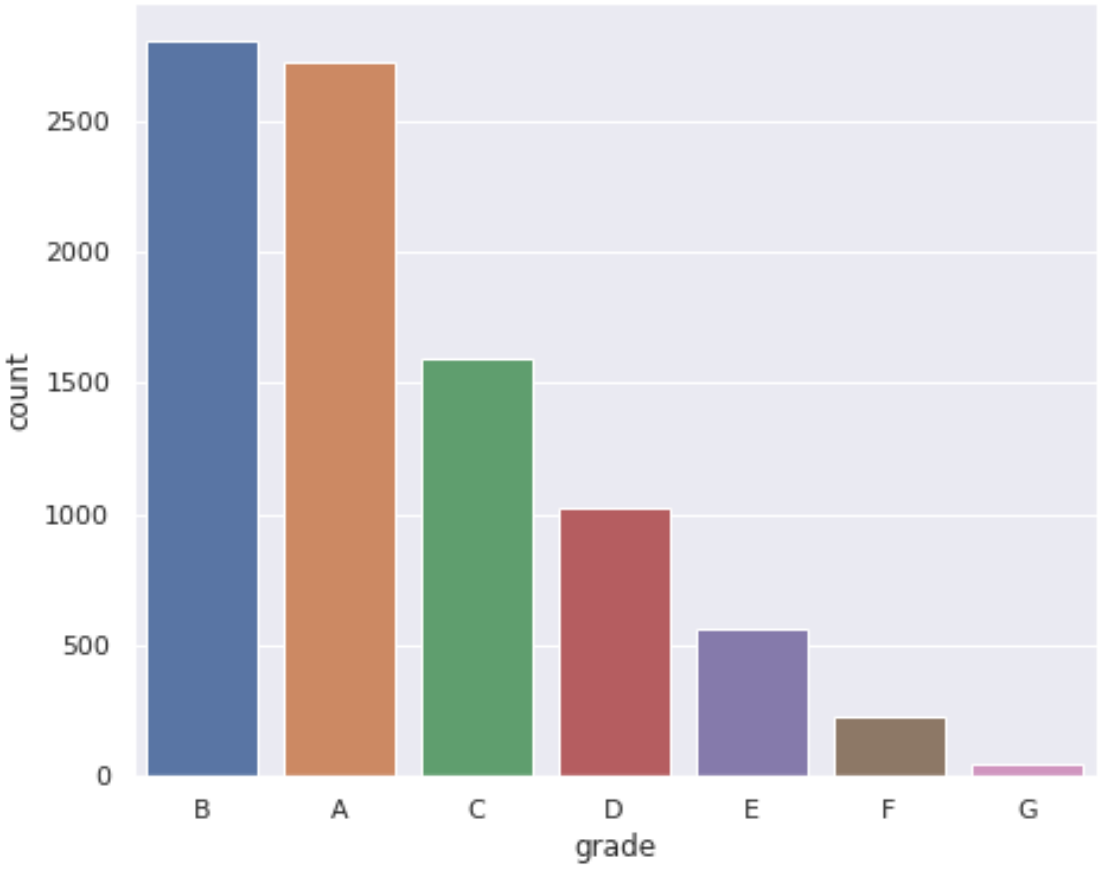
	addr_state	loan_amnt
4	CA	20463225
28	NY	10178275
37	TX	8441150
9	FL	7682875
25	NJ	5297625

[518]:
```
import plotly.graph_objects as go

fig = go.Figure(data=go.Choropleth(
    locations=df_location['addr_state'], # Spatial coordinates
    z = df_location['loan_amnt'].astype(float), # Data to be color-coded
    locationmode = 'USA-states', # set of locations match entries in `locations`
    colorscale = 'Reds',
    colorbar_title = "USD",
))

fig.update_layout(
```

```
    title_text = 'Total amount issued by State',
    geo_scope='usa', # limited map scope to USA
)

fig.show()
```

[519]:
```
plt.figure(figsize=(16, 6))
plt.subplot(1, 2, 1)
sns.countplot(x="grade", data= LC_df_nonull, order = LC_df_nonull['grade'].
 ↪value_counts().index)
```

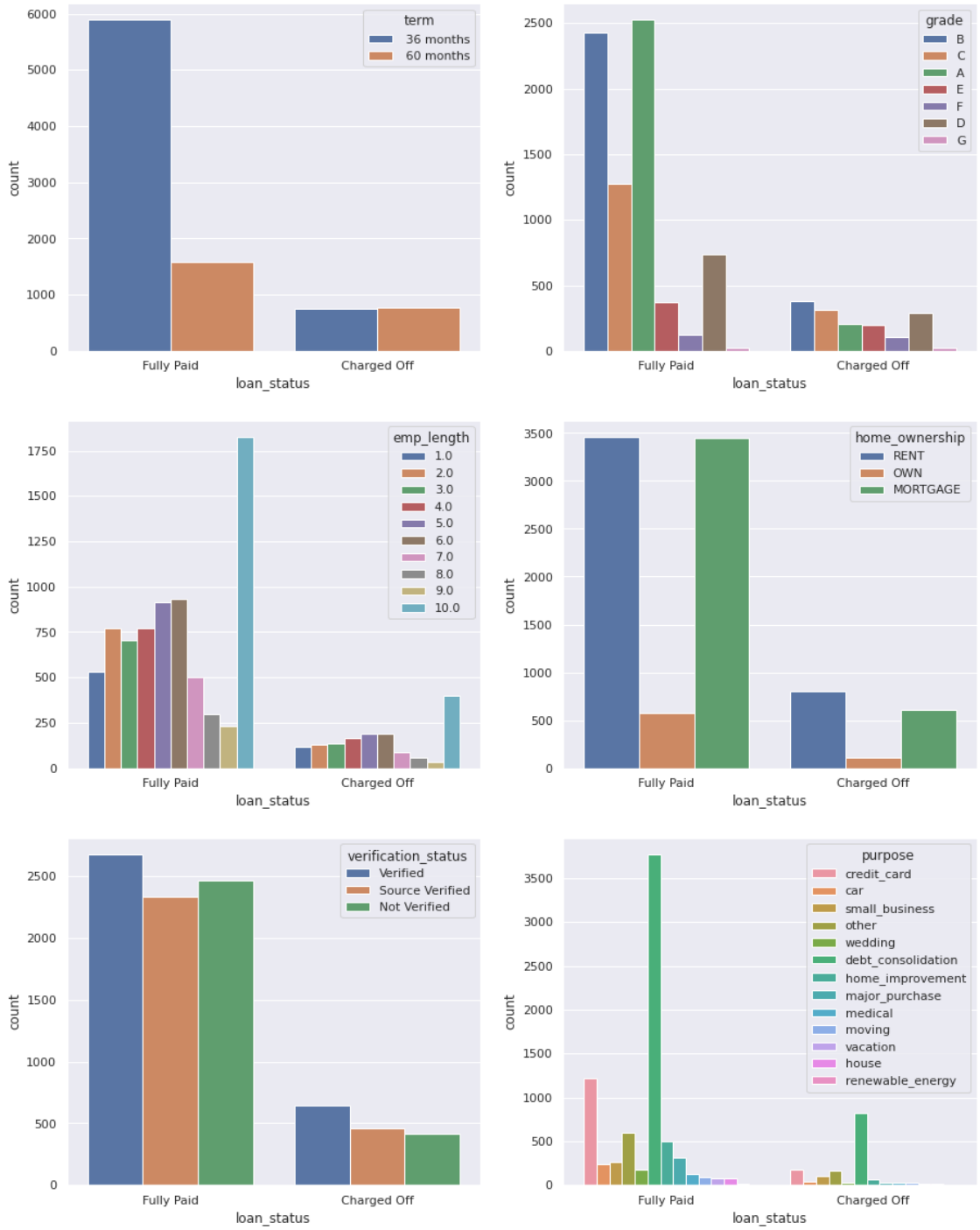
[519]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47a4532ad0>



[520]:
```
_,axss = plt.subplots(3,2, figsize=[15,20])
sns.countplot(x='loan_status', hue='term', data=LC_df_nonull, ax=axss[0][0])
sns.countplot(x='loan_status', hue='grade', data=LC_df_nonull, ax=axss[0][1])
sns.countplot(x='loan_status', hue='emp_length', data=LC_df_nonull,␣
 ↪ax=axss[1][0])
```

```
sns.countplot(x='loan_status', hue='home_ownership', data=LC_df_nonull,␣
 ↪ax=axss[1][1])
sns.countplot(x='loan_status', hue='verification_status', data=LC_df_nonull,␣
 ↪ax=axss[2][0])
sns.countplot(x='loan_status', hue='purpose', data=LC_df_nonull, ax=axss[2][1])
```

[520]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47a40d8e50>



6 Part 2 Feature Preprocessing

One hot to categorical features

[521]:
```
catCol = []
for col in LC_df_nonull:
  if (LC_df_nonull[col]).dtype==np.object and col!="loan_status":
    catCol.append(col)

catCol
```

[521]:
```
['term',
 'grade',
 'home_ownership',
 'verification_status',
 'purpose',
 'addr_state']
```

[522]:
```
def OneHotEncoding(df, enc, categories):
  transformed = pd.DataFrame(enc.transform(df[categories]).toarray(),␣
 ↪columns=enc.get_feature_names(categories))
  return pd.concat([df.reset_index(drop=True), transformed], axis=1).
 ↪drop(categories, axis=1)


enc_ohe = OneHotEncoder()
enc_ohe.fit(LC_df_nonull[['term',
 'grade',
 'home_ownership',
 'verification_status',
 'purpose',
 'addr_state']])
LC_df_nonull = OneHotEncoding(LC_df_nonull, enc_ohe, ['term',
 'grade',
 'home_ownership',
 'verification_status',
 'purpose',
 'addr_state'])
# for col in catCol:
#   categories = np.array(np.array(col))
#   enc_ohe.fit(LC_df_nonull[categories])
#   LC_df_nonull = OneHotEncoding(LC_df_nonull, enc_ohe, categories)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarning:

Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0
and will be removed in 1.2. Please use get_feature_names_out instead.
```

[523]:
```
LC_df_nonull.head()
```

[523]:
```
        id  loan_amnt  funded_amnt  …  addr_state_WI  addr_state_WV
addr_state_WY
0  1077501       5000         5000  …            0.0            0.0
0.0
1  1077430       2500         2500  …            0.0            0.0
0.0
2  1077175       2400         2400  …            0.0            0.0
0.0
3  1076863      10000        10000  …            0.0            0.0
0.0
4  1075269       5000         5000  …            0.0            0.0
0.0

[5 rows x 98 columns]
```

drop useless columns: id

[524]:
```
LC_df_nonull = LC_df_nonull.drop(["id"], axis=1)
LC_df_nonull = LC_df_nonull.drop(["annualInc_bucket"], axis=1)
LC_df_nonull = LC_df_nonull.drop(["emp_length_helper"], axis=1)
```

drop features that are highly correlated with target

[525]:
```
#drop_col = ["total_pymnt", "total_pymnt_inv", "total_rec_int",␣
 ↪"total_rec_prncp"]
LC_df_nonull = LC_df_nonull.drop(["total_pymnt"], axis=1)
LC_df_nonull = LC_df_nonull.drop(["total_pymnt_inv"], axis=1)
LC_df_nonull = LC_df_nonull.drop(["total_rec_int"], axis=1)
LC_df_nonull = LC_df_nonull.drop(["total_rec_prncp"], axis=1)
```

Data Splitting

[526]:
```
Y = LC_df_nonull["loan_status"]
X = LC_df_nonull.drop(["loan_status"], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state=888,␣
 ↪test_size=.25)
```

Standardization

[527]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6744 entries, 8111 to 1946
Data columns (total 90 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   loan_amnt                             6744 non-null   int64
 1   funded_amnt                           6744 non-null   int64
 2   funded_amnt_inv                       6744 non-null   float64
 3   int_rate                              6744 non-null   float64
 4   installment                           6744 non-null   float64
 5   emp_length                            6744 non-null   float64
 6   annual_inc                            6744 non-null   float64
 7   dti                                   6744 non-null   float64
 8   earliest_cr_line                      6744 non-null   int64
 9   inq_last_6mths                        6744 non-null   int64
 10  open_acc                              6744 non-null   int64
 11  pub_rec                               6744 non-null   int64
 12  revol_bal                             6744 non-null   int64
 13  revol_util                            6744 non-null   float64
 14  total_acc                             6744 non-null   int64
 15  out_prncp                             6744 non-null   int64
 16  out_prncp_inv                         6744 non-null   int64
 17  term_ 36 months                       6744 non-null   float64
 18  term_ 60 months                       6744 non-null   float64
 19  grade_A                               6744 non-null   float64
 20  grade_B                               6744 non-null   float64
 21  grade_C                               6744 non-null   float64
 22  grade_D                               6744 non-null   float64
 23  grade_E                               6744 non-null   float64
 24  grade_F                               6744 non-null   float64
 25  grade_G                               6744 non-null   float64
 26  home_ownership_MORTGAGE               6744 non-null   float64
 27  home_ownership_OWN                    6744 non-null   float64
 28  home_ownership_RENT                   6744 non-null   float64
 29  verification_status_Not Verified      6744 non-null   float64
 30  verification_status_Source Verified   6744 non-null   float64
 31  verification_status_Verified          6744 non-null   float64
 32  purpose_car                           6744 non-null   float64
 33  purpose_credit_card                   6744 non-null   float64
 34  purpose_debt_consolidation            6744 non-null   float64
 35  purpose_home_improvement              6744 non-null   float64
 36  purpose_house                         6744 non-null   float64
 37  purpose_major_purchase                6744 non-null   float64
 38  purpose_medical                       6744 non-null   float64
 39  purpose_moving                        6744 non-null   float64
```

```
40  purpose_other                  6744 non-null   float64
41  purpose_renewable_energy       6744 non-null   float64
42  purpose_small_business         6744 non-null   float64
43  purpose_vacation               6744 non-null   float64
44  purpose_wedding                6744 non-null   float64
45  addr_state_AK                  6744 non-null   float64
46  addr_state_AL                  6744 non-null   float64
47  addr_state_AR                  6744 non-null   float64
48  addr_state_AZ                  6744 non-null   float64
49  addr_state_CA                  6744 non-null   float64
50  addr_state_CO                  6744 non-null   float64
51  addr_state_CT                  6744 non-null   float64
52  addr_state_DC                  6744 non-null   float64
53  addr_state_DE                  6744 non-null   float64
54  addr_state_FL                  6744 non-null   float64
55  addr_state_GA                  6744 non-null   float64
56  addr_state_HI                  6744 non-null   float64
57  addr_state_IL                  6744 non-null   float64
58  addr_state_KS                  6744 non-null   float64
59  addr_state_KY                  6744 non-null   float64
60  addr_state_LA                  6744 non-null   float64
61  addr_state_MA                  6744 non-null   float64
62  addr_state_MD                  6744 non-null   float64
63  addr_state_MI                  6744 non-null   float64
64  addr_state_MN                  6744 non-null   float64
65  addr_state_MO                  6744 non-null   float64
66  addr_state_MS                  6744 non-null   float64
67  addr_state_MT                  6744 non-null   float64
68  addr_state_NC                  6744 non-null   float64
69  addr_state_NH                  6744 non-null   float64
70  addr_state_NJ                  6744 non-null   float64
71  addr_state_NM                  6744 non-null   float64
72  addr_state_NV                  6744 non-null   float64
73  addr_state_NY                  6744 non-null   float64
74  addr_state_OH                  6744 non-null   float64
75  addr_state_OK                  6744 non-null   float64
76  addr_state_OR                  6744 non-null   float64
77  addr_state_PA                  6744 non-null   float64
78  addr_state_RI                  6744 non-null   float64
79  addr_state_SC                  6744 non-null   float64
80  addr_state_SD                  6744 non-null   float64
81  addr_state_TN                  6744 non-null   float64
82  addr_state_TX                  6744 non-null   float64
83  addr_state_UT                  6744 non-null   float64
84  addr_state_VA                  6744 non-null   float64
85  addr_state_VT                  6744 non-null   float64
86  addr_state_WA                  6744 non-null   float64
87  addr_state_WI                  6744 non-null   float64
```

```
 88  addr_state_WV                        6744 non-null   float64
 89  addr_state_WY                        6744 non-null   float64
dtypes: float64(80), int64(10)
memory usage: 4.7 MB
```

[528]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

7 Part 3: Modeling

RandomForest/ Kneighbors/ LogisticRegression

[529]:
```
# Logistic Regression
classifier_logistic = LogisticRegression()

# K Nearest Neighbors
classifier_KNN = KNeighborsClassifier()

# Random Forest
classifier_RF = RandomForestClassifier()
```

[532]:
```
classifier_logistic.fit(X_train, y_train)
y_pred = classifier_logistic.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

 Charged Off       0.52      0.11      0.19       385
  Fully Paid       0.84      0.98      0.91      1864

    accuracy                           0.83      2249
   macro avg       0.68      0.55      0.55      2249
weighted avg       0.79      0.83      0.78      2249
```

[533]:
```
classifier_KNN.fit(X_train, y_train)
y_pred = classifier_KNN.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

 Charged Off       0.34      0.10      0.16       385
  Fully Paid       0.84      0.96      0.89      1864
```

```
    accuracy                           0.81      2249
   macro avg       0.59      0.53      0.53      2249
weighted avg       0.75      0.81      0.77      2249
```

[534]:
```
classifier_RF.fit(X_train, y_train)
y_pred = classifier_RF.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

 Charged Off       0.40      0.05      0.09       385
  Fully Paid       0.83      0.98      0.90      1864

    accuracy                           0.82      2249
   macro avg       0.61      0.52      0.50      2249
weighted avg       0.76      0.82      0.76      2249
```

Cross Validation

[536]:
```
model_names = ['Logistic Regression','KNN','Random Forest']
model_list = [classifier_logistic, classifier_KNN, classifier_RF]
count = 0

for classifier in model_list:
    cv_score = model_selection.cross_val_score(classifier, X_train, y_train,␣
 ↪cv=5)
    print(cv_score)
    print('Model accuracy of ' + model_names[count] + ' is ' + str(cv_score.
 ↪mean()))
    count += 1
```

```
[0.82727947 0.83543365 0.84803558 0.82876205 0.84198813]
Model accuracy of Logistic Regression is 0.8362997758533082
[0.80578206 0.8176427  0.81912528 0.82060786 0.81231454]
Model accuracy of KNN is 0.8150944869592379
[0.82802076 0.82950334 0.83765752 0.82876205 0.82863501]
Model accuracy of Random Forest is 0.8305157353617252
```

Logistic Regression has the best performance.

SVM

[538]:
```
 classifier_SVC = SVC()

cv_score = model_selection.cross_val_score(classifier_SVC, X_train, y_train,␣
 ↪cv=5)
print('Model accuracy of SVM is: ' + str(cv_score.mean()))
```

```
Model accuracy of SVM is: 0.830220099293245
```

Neural Networks

[543]:
```
mlp = MLPClassifier(hidden_layer_sizes=(30,30))
mlp.fit(X_train,y_train)
y_pred = mlp.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

 Charged Off       0.33      0.29      0.31       385
  Fully Paid       0.86      0.88      0.87      1864

    accuracy                           0.78      2249
   macro avg       0.59      0.58      0.59      2249
weighted avg       0.77      0.78      0.77      2249


/usr/local/lib/python3.7/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:696:
ConvergenceWarning:

Stochastic Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
```

[544]:
```
classifier_RF.fit(X, Y)
importances = classifier_RF.feature_importances_
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature importance ranking by Random Forest Model:")
for ind in range(X.shape[1]):
  print ("{0} : {1}".format(X.
 ↪columns[indices[ind]],round(importances[indices[ind]], 4)))
```

```
Feature importance ranking by Random Forest Model:
int_rate : 0.0656
annual_inc : 0.0654
revol_util : 0.0637
dti : 0.0621
earliest_cr_line : 0.0595
revol_bal : 0.0588
installment : 0.055
total_acc : 0.0516
funded_amnt_inv : 0.0479
open_acc : 0.0419
loan_amnt : 0.0415
```

```
funded_amnt : 0.0414
emp_length : 0.034
term_ 36 months : 0.0247
inq_last_6mths : 0.0223
term_ 60 months : 0.0164
purpose_debt_consolidation : 0.0102
addr_state_CA : 0.0097
verification_status_Source Verified : 0.0092
home_ownership_RENT : 0.0088
home_ownership_MORTGAGE : 0.0087
verification_status_Verified : 0.0086
verification_status_Not Verified : 0.0081
grade_A : 0.0074
purpose_small_business : 0.007
addr_state_NY : 0.0068
purpose_other : 0.0067
addr_state_FL : 0.0066
purpose_credit_card : 0.0066
grade_D : 0.0063
grade_C : 0.0061
grade_B : 0.0056
pub_rec : 0.0053
grade_F : 0.0052
grade_E : 0.0052
home_ownership_OWN : 0.0049
addr_state_TX : 0.0047
addr_state_NJ : 0.0046
addr_state_IL : 0.0045
addr_state_GA : 0.0043
purpose_home_improvement : 0.0043
addr_state_MD : 0.0041
addr_state_VA : 0.0038
addr_state_AZ : 0.0037
addr_state_PA : 0.0036
addr_state_OH : 0.0036
addr_state_NV : 0.0035
addr_state_WA : 0.0034
purpose_medical : 0.0033
addr_state_NC : 0.003
purpose_car : 0.0028
addr_state_MA : 0.0026
purpose_moving : 0.0024
purpose_wedding : 0.0024
addr_state_CT : 0.0023
addr_state_MI : 0.0022
addr_state_MN : 0.0021
addr_state_OR : 0.0021
purpose_major_purchase : 0.0021
```

```
addr_state_RI : 0.002
addr_state_WI : 0.002
addr_state_MO : 0.0017
addr_state_AL : 0.0017
grade_G : 0.0015
addr_state_KY : 0.0014
addr_state_CO : 0.0014
addr_state_NM : 0.0014
addr_state_OK : 0.0014
addr_state_SC : 0.0013
addr_state_LA : 0.0013
purpose_house : 0.0013
addr_state_AK : 0.0012
purpose_vacation : 0.0012
addr_state_KS : 0.0011
addr_state_HI : 0.0011
addr_state_AR : 0.0009
addr_state_NH : 0.0009
addr_state_DC : 0.0008
addr_state_SD : 0.0008
addr_state_WV : 0.0007
addr_state_UT : 0.0006
purpose_renewable_energy : 0.0005
addr_state_DE : 0.0005
addr_state_MT : 0.0005
addr_state_WY : 0.0002
addr_state_VT : 0.0002
addr_state_MS : 0.0
out_prncp : 0.0
out_prncp_inv : 0.0
addr_state_TN : 0.0
```

[547]:
```
X_with_corr = X.copy()
scaler = StandardScaler()
X_l2 = scaler.fit_transform(X_with_corr)
LRmodel_l2 = LogisticRegression(penalty="l2", C = 0.1, solver='liblinear')
LRmodel_l2.fit(X_l2, Y)

indices = np.argsort(abs(LRmodel_l2.coef_[0]))[::-1]

print ("Logistic Regression (L2) Coefficients")
for ind in range(X_with_corr.shape[1]):
  print ("{0} : {1}".format(X_with_corr.columns[indices[ind]],round(LRmodel_l2.
 ↪coef_[0][indices[ind]], 4)))
```

```
Logistic Regression (L2) Coefficients
int_rate : -0.5656
annual_inc : 0.5101
```

```
term_ 36 months : 0.1983
term_ 60 months : -0.1983
purpose_small_business : -0.1601
revol_util : -0.1587
purpose_credit_card : 0.1262
purpose_other : -0.1152
inq_last_6mths : -0.1104
loan_amnt : -0.1083
grade_A : -0.1039
purpose_major_purchase : 0.0763
grade_E : 0.076
purpose_medical : -0.0684
addr_state_TX : 0.0681
funded_amnt : 0.0664
addr_state_UT : 0.0661
addr_state_KS : 0.065
purpose_moving : -0.0646
addr_state_NV : -0.0617
addr_state_CA : -0.0601
earliest_cr_line : 0.0576
funded_amnt_inv : -0.057
open_acc : -0.0557
total_acc : 0.0555
pub_rec : -0.053
addr_state_RI : -0.0523
addr_state_MD : -0.0508
addr_state_MO : 0.0496
grade_C : 0.0489
addr_state_AR : 0.0487
addr_state_MT : 0.0473
verification_status_Not Verified : -0.0469
addr_state_OK : 0.0438
verification_status_Verified : 0.0403
grade_D : 0.0395
home_ownership_RENT : -0.0393
dti : -0.037
addr_state_AK : -0.0369
addr_state_CO : 0.0364
addr_state_SD : -0.0361
addr_state_FL : -0.0358
purpose_home_improvement : 0.0356
addr_state_LA : 0.0326
addr_state_TN : 0.0323
addr_state_WV : 0.0312
home_ownership_MORTGAGE : 0.03
addr_state_MS : 0.0287
addr_state_VA : -0.028
addr_state_AL : -0.0277
```

```
grade_F : 0.0266
purpose_debt_consolidation : 0.0258
addr_state_CT : 0.0245
addr_state_NM : -0.0242
addr_state_SC : 0.0221
addr_state_OR : -0.0218
purpose_wedding : 0.0216
emp_length : -0.0216
addr_state_MI : -0.0213
addr_state_WI : 0.0185
addr_state_PA : 0.0183
addr_state_NC : 0.0179
home_ownership_OWN : 0.0177
addr_state_AZ : -0.017
purpose_renewable_energy : -0.0153
addr_state_DE : 0.015
purpose_car : -0.0149
addr_state_DC : 0.0149
grade_B : -0.0141
addr_state_VT : 0.0136
addr_state_OH : -0.0123
addr_state_GA : -0.0113
addr_state_HI : 0.0111
addr_state_NH : -0.0097
addr_state_MA : 0.0083
grade_G : 0.0059
addr_state_NY : 0.0059
addr_state_IL : -0.0058
purpose_vacation : -0.0058
installment : 0.0055
addr_state_WA : -0.0055
verification_status_Source Verified : 0.0052
addr_state_WY : 0.005
purpose_house : -0.0043
addr_state_MN : 0.0039
revol_bal : 0.0034
addr_state_KY : 0.0022
addr_state_NJ : -0.0012
out_prncp : 0.0
out_prncp_inv : 0.0
```