```
#Dataset Information  https://drive.google.com/file/d/1VOAhrB5eIhFc8C5qeFx5QYa2OLD7s1eV/view


import pandas as pd
import numpy as np

from tabulate import tabulate#transfer data into table format

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

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
# link='https://drive.google.com/file/d/1VOAhrB5eIhFc8C5qeFx5QYa2OLD7s1eV/view'
# fluff, id = link.split('=')
# file = drive.CreateFile({'id':id})
# file.GetContentFile('loan-clear-version.csv')
Lingyi_LC_df = pd.read_csv('/content/loan-clean-version.csv')
Lingyi_LC_df.head()
```

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	1077501	5000	5000	4975.0	36 months	10.65	162.87
1	1077430	2500	2500	2500.0	60 months	15.27	59.83
2	1077175	2400	2400	2400.0	36 months	15.96	84.33
3	1076863	10000	10000	10000.0	36 months	13.49	339.31
4	1075269	5000	5000	5000.0	36 months	7.90	156.46

5 rows × 29 columns

```
#Part1 Data Exploration


Lingyi_LC_df.describe()
```

	id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	insta
count	9.004000e+03	9004.000000	9004.000000	9004.000000	9004.000000	9004.
mean	9.632337e+05	12291.884163	12154.156486	12076.054639	12.126728	357.
std	7.953238e+04	8285.682170	8096.937145	8033.211335	4.195740	227
min	4.581650e+05	1000.000000	1000.000000	750.000000	5.420000	30
25%	8.778840e+05	6000.000000	6000.000000	6000.000000	8.490000	187.
50%	9.879685e+05	10000.000000	10000.000000	10000.000000	11.710000	312
75%	1.033607e+06	16000.000000	16000.000000	15975.000000	15.230000	469.
max	1.077501e+06	35000.000000	35000.000000	35000.000000	24.110000	1288

8 rows × 21 columns

```
Lingyi_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
```

```
Lingyi_LC_df.nunique()
```

```
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
```

```
Lingyi_LC_df.isnull().sum()
```

```
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
```

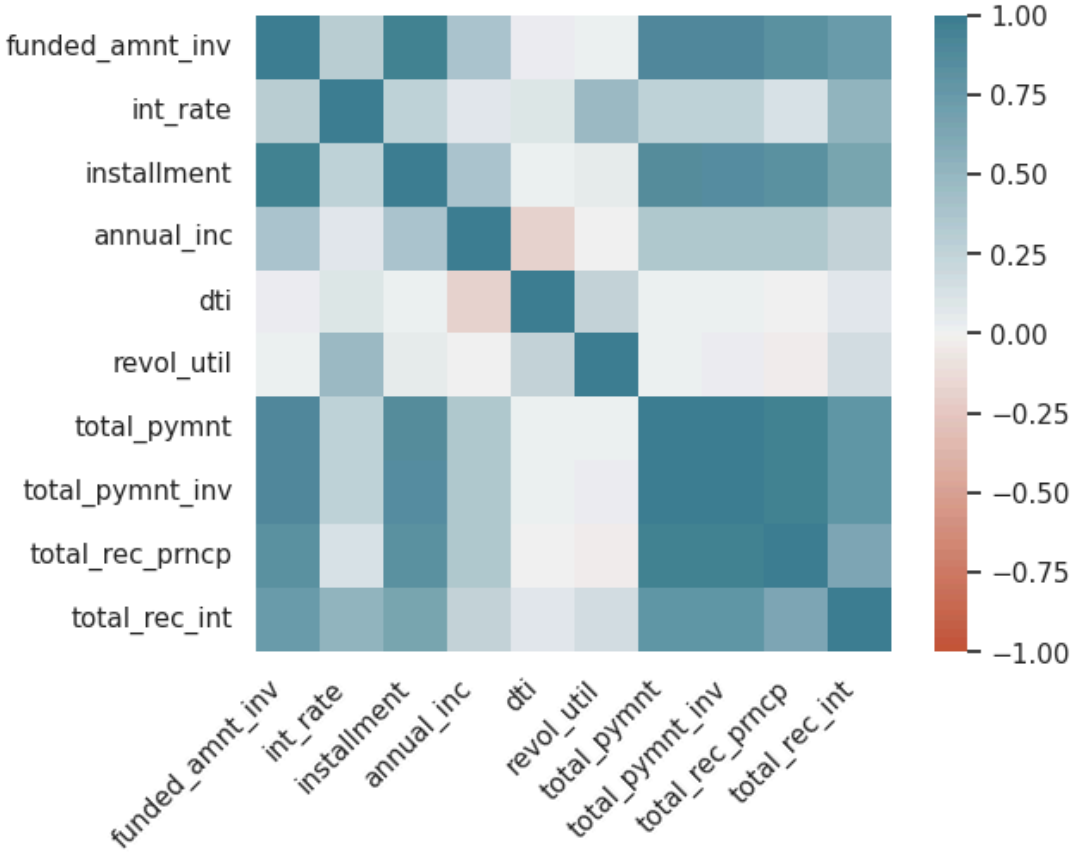
```
Lingyi_LC_df.loc[:,'loan_status'].value_counts()
```

```
Fully Paid     7487
Charged Off    1517
Name: loan_status, dtype: int64
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
numCol = []
for col in Lingyi_LC_df:
  if Lingyi_LC_df[col].dtype == float:
    numCol.append(col)
corr = Lingyi_LC_df[numCol].corr()#pearson
```

```
ax = sns.heatmap(
    corr,
    vmin = -1,vmax = 1,center = 0,
    cmap = sns.diverging_palette(20,220,n=200),
    square = True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation = 45,
    horizontalalignment = 'right')
```

```
[Text(0.5, 0, 'funded_amnt_inv'),
 Text(1.5, 0, 'int_rate'),
 Text(2.5, 0, 'installment'),
 Text(3.5, 0, 'annual_inc'),
 Text(4.5, 0, 'dti'),
 Text(5.5, 0, 'revol_util'),
 Text(6.5, 0, 'total_pymnt'),
 Text(7.5, 0, 'total_pymnt_inv'),
 Text(8.5, 0, 'total_rec_prncp'),
 Text(9.5, 0, 'total_rec_int')]
```



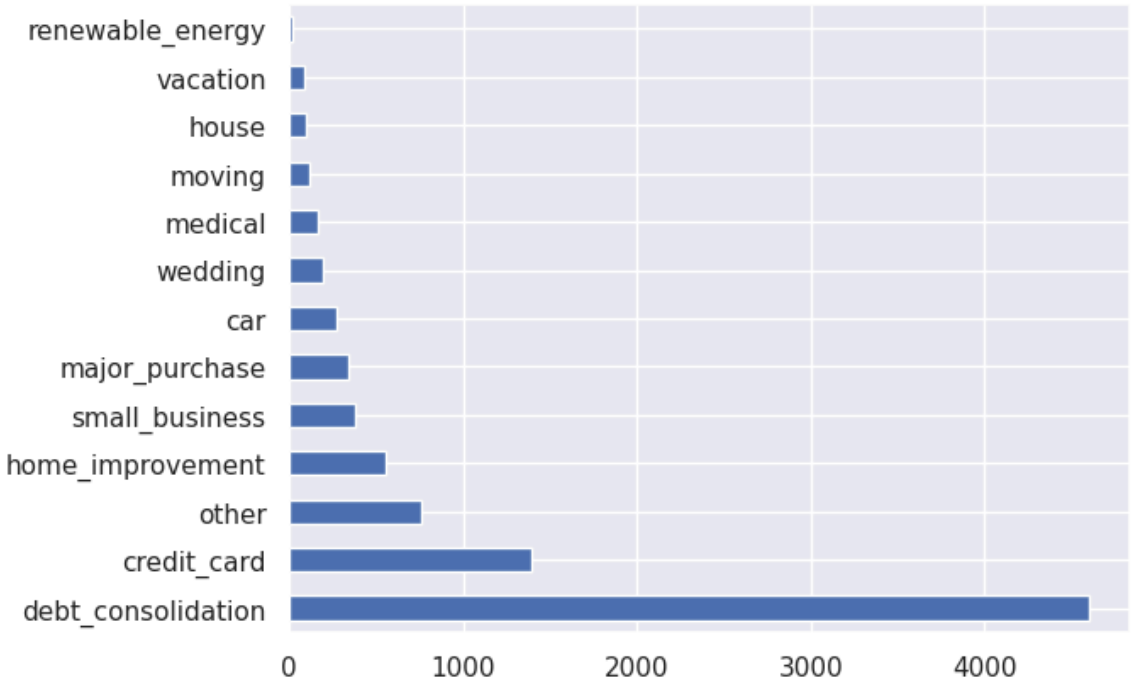

```
corr_score = Lingyi_LC_df[numCol].corr()
corr_score
```

	funded_amnt_inv	int_rate	installment	annual_inc	dti	rev
funded_amnt_inv	1.000000	0.302945	0.959195	0.371554	0.025192	(
int_rate	0.302945	1.000000	0.267857	0.076103	0.091614	(
installment	0.959195	0.267857	1.000000	0.385192	0.017451	(
annual_inc	0.371554	0.076103	0.385192	1.000000	−0.176920	−
dti	0.025192	0.091614	0.017451	−0.176920	1.000000	(
revol_util	0.022395	0.470893	0.056484	−0.005319	0.243479	
total_pymnt	0.884368	0.259296	0.859719	0.366455	0.020130	
total_pymnt_inv	0.885395	0.258493	0.859358	0.364869	0.019860	
total_rec_prncp	0.829373	0.124425	0.827792	0.365577	−0.004552	−(
total_rec_int	0.734486	0.531955	0.660184	0.253143	0.077365	

Next steps: View recommended plots

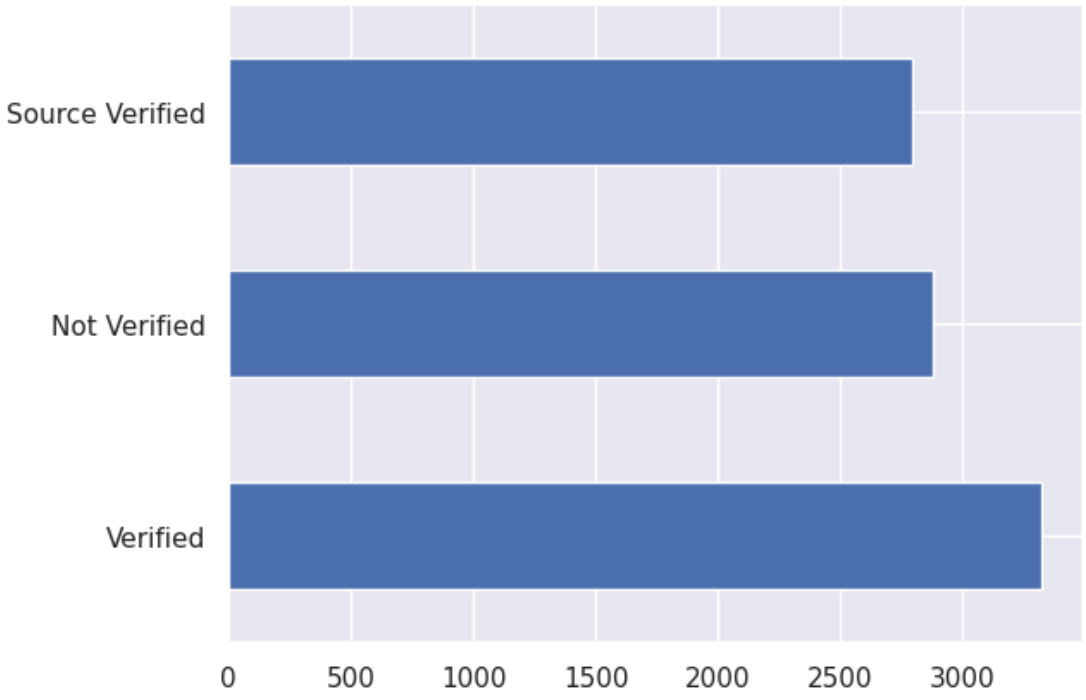
```
Lingyi_LC_df['purpose'].value_counts().plot(kind = 'barh')#horizontal
```

```
<Axes: >
```



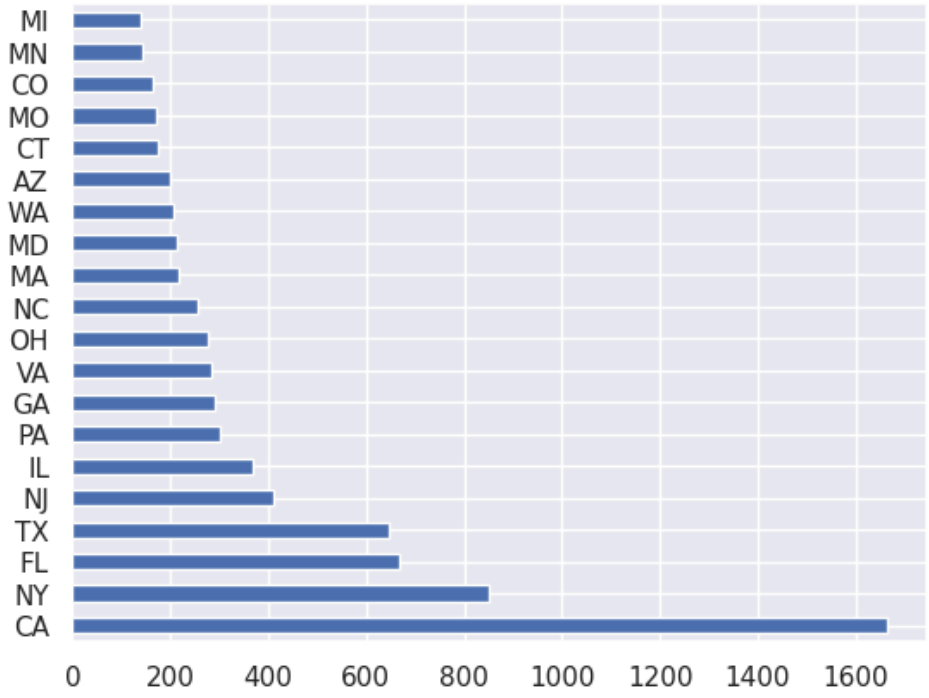

```
Lingyi_LC_df['verification_status'].value_counts().plot(kind = 'barh')
```

```
<Axes: >
```



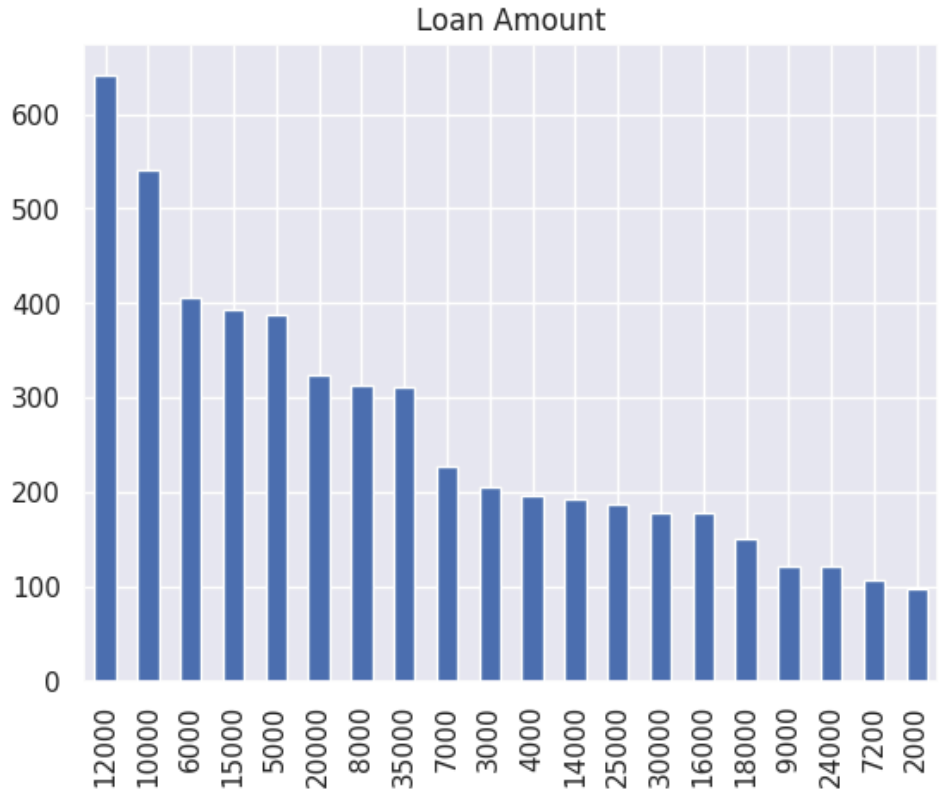

```
Lingyi_LC_df['addr_state'].value_counts()[:20].plot(kind = 'barh')
```

```
<Axes: >
```



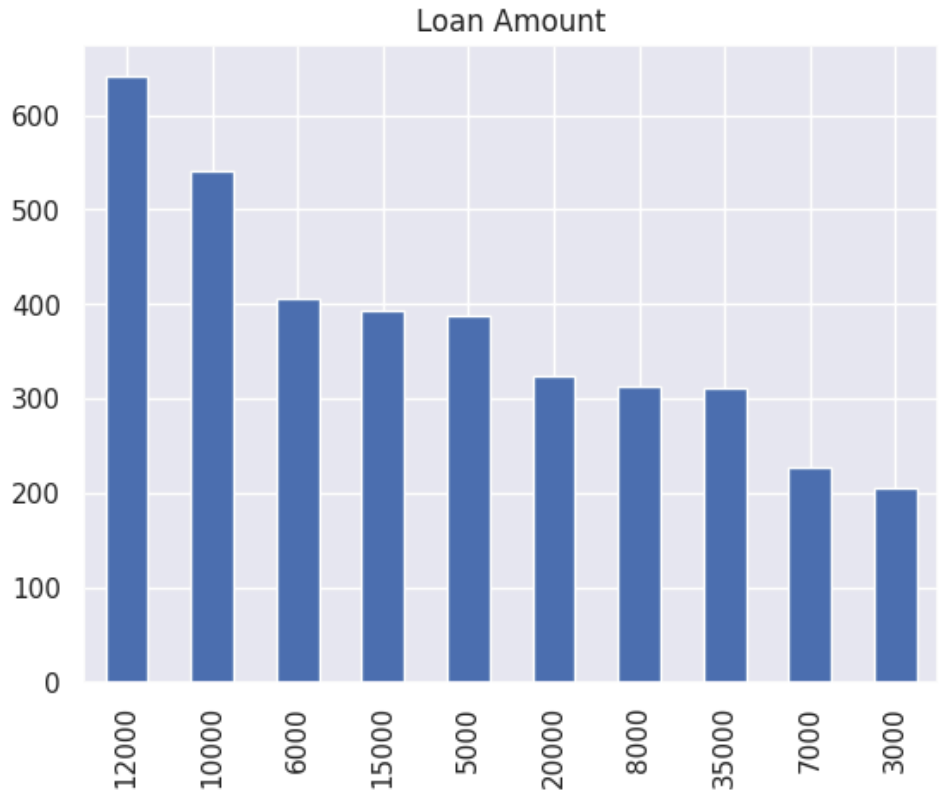

```
Lingyi_LC_df['loan_amnt'].value_counts()[:20].plot(kind = 'bar',title = 'Loan Amount')
```

```
<Axes: title={'center': 'Loan Amount'}>
```



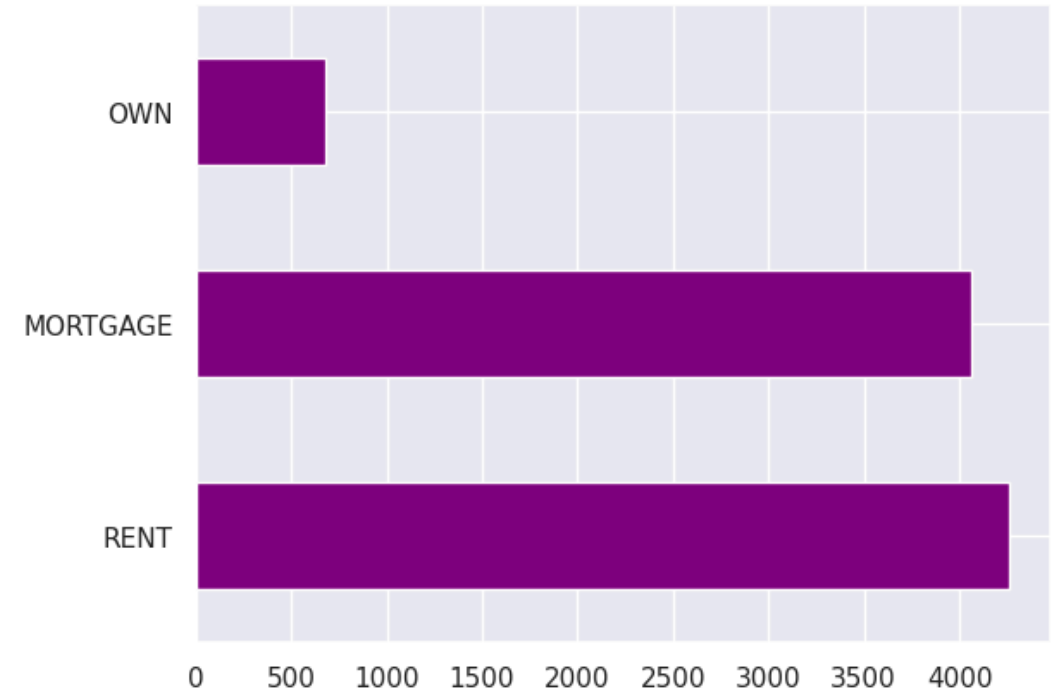

```
Lingyi_LC_df['loan_amnt'].value_counts()[:10].plot(kind = 'bar',title = 'Loan Amount')
```

```
<Axes: title={'center': 'Loan Amount'}>
```



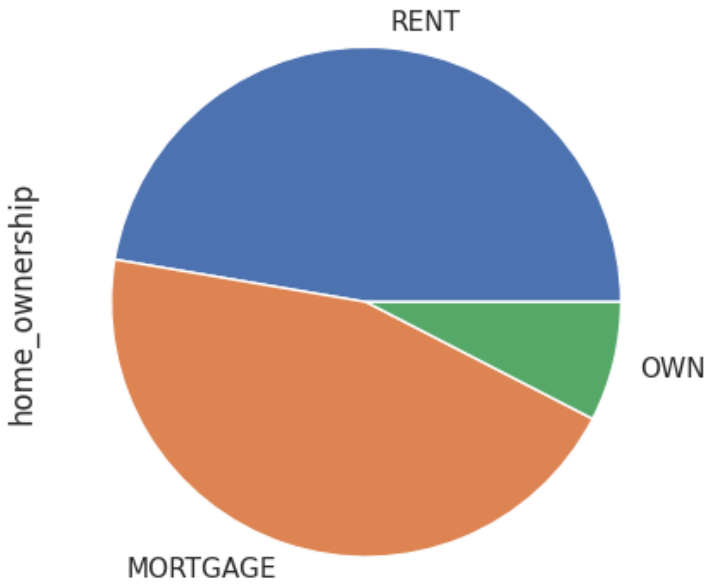

```
Lingyi_LC_df['home_ownership'].value_counts().plot(kind = 'barh',color= 'purple')
```

```
<Axes: >
```



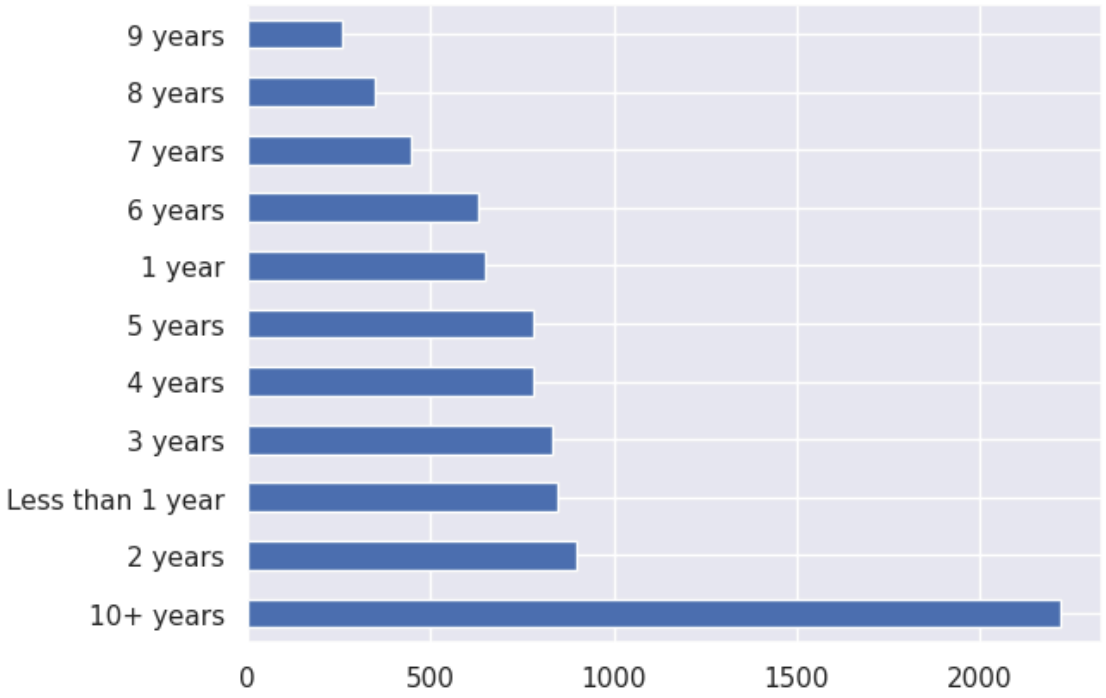

```
Lingyi_LC_df['home_ownership'].value_counts().plot(kind = 'pie')
```

```
<Axes: ylabel='home_ownership'>
```




```
Lingyi_LC_df['emp_length'].value_counts().plot(kind = 'barh')
```

```
<Axes: >
```




```
df_location = Lingyi_LC_df.groupby('addr_state',).sum().reset_index()
df_location = df_location.filter(['addr_state','loan_amnt'],axis = 1)
df_location.head()
```

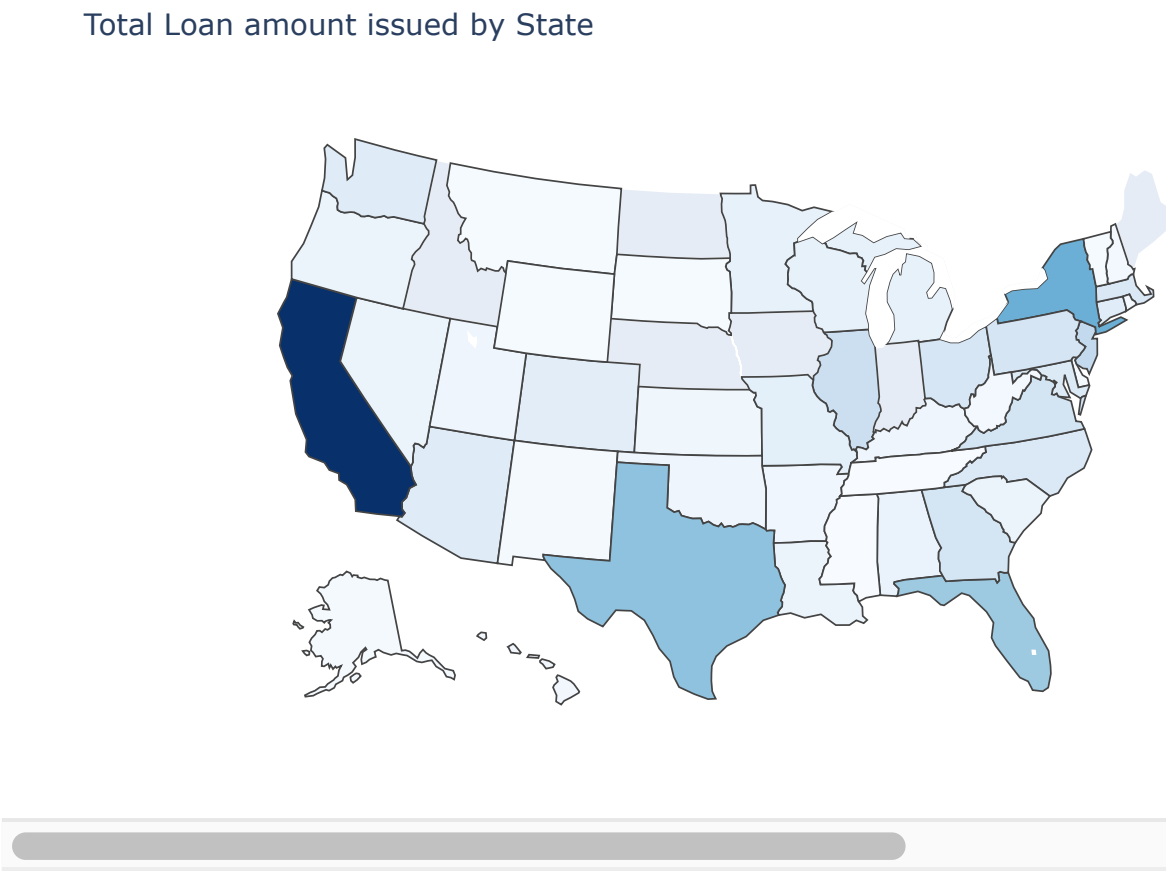
```
<ipython-input-127-4cb65c8ce8c0>:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a fu
```

	addr_state	loan_amnt
0	AK	359675
1	AL	1308425
2	AR	753275
3	AZ	2477200
4	CA	20470425

Next steps: View recommended plots

```
import plotly.graph_objects as go

fig = go.Figure(data=go.Choropleth(
    locations=df_location['addr_state'],
    z = df_location['loan_amnt'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Blues',
    colorbar_title = "USD",
))

fig.update_layout(
    title_text = 'Total Loan amount issued by State',
```

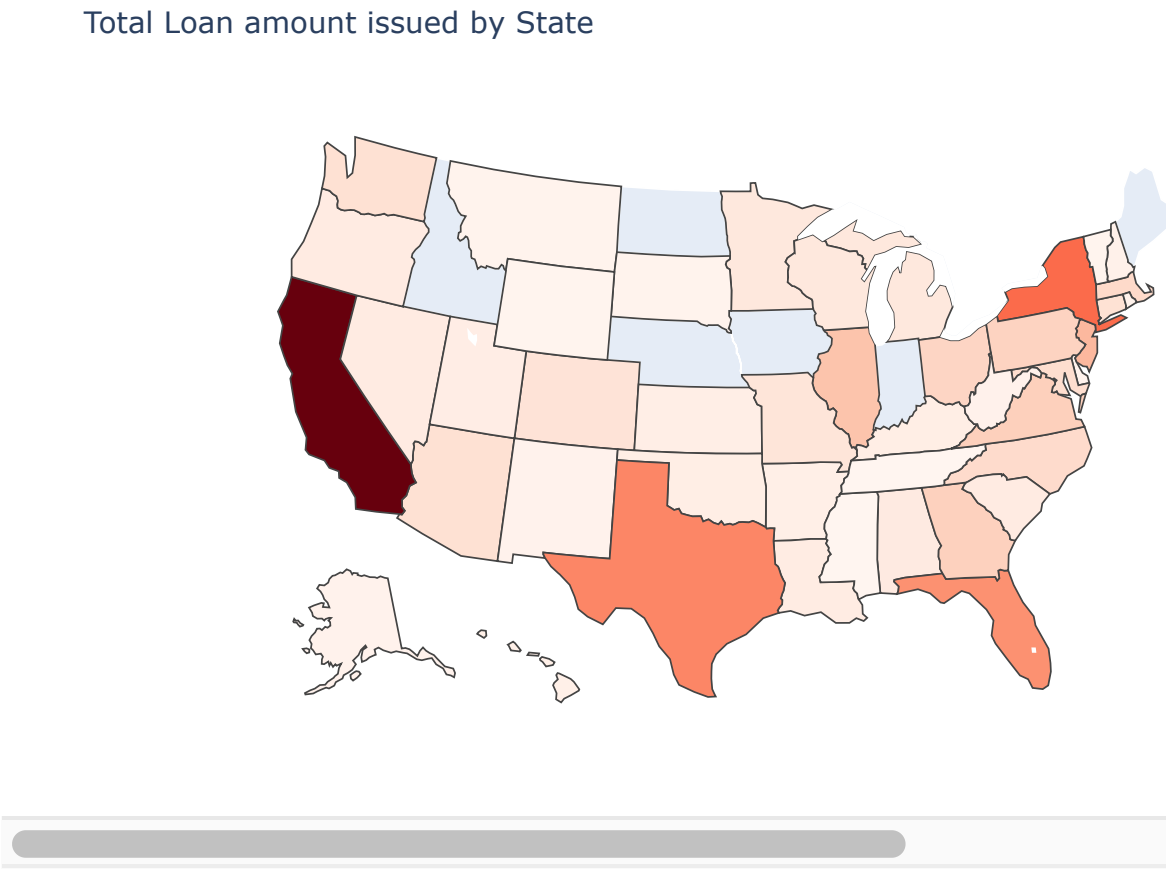
```
    geo_scope='usa',
)

fig.show()
```



双击（或按回车键）即可修改

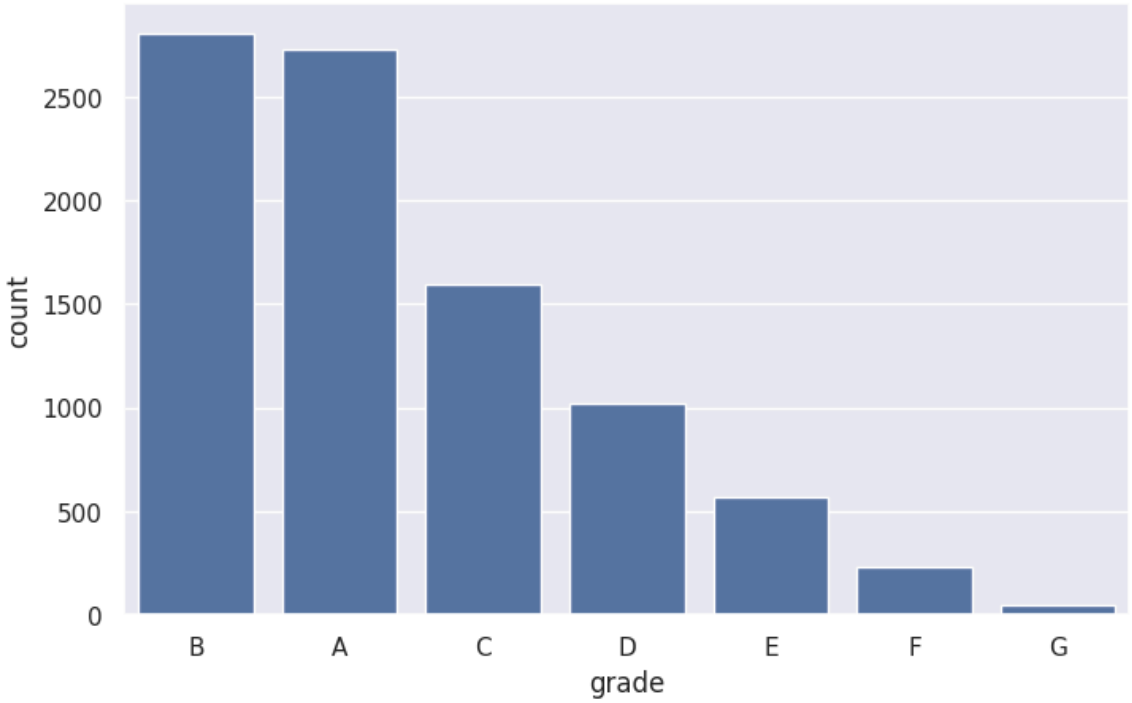
双击（或按回车键）即可修改

```
import plotly.graph_objects as go

fig = go.Figure(data=go.Choropleth(
    locations=df_location['addr_state'],
    z = df_location['loan_amnt'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Reds',
    colorbar_title = "USD",
))

fig.update_layout(
    title_text = 'Total Loan amount issued by State',
    geo_scope='usa',
)

fig.show()
```



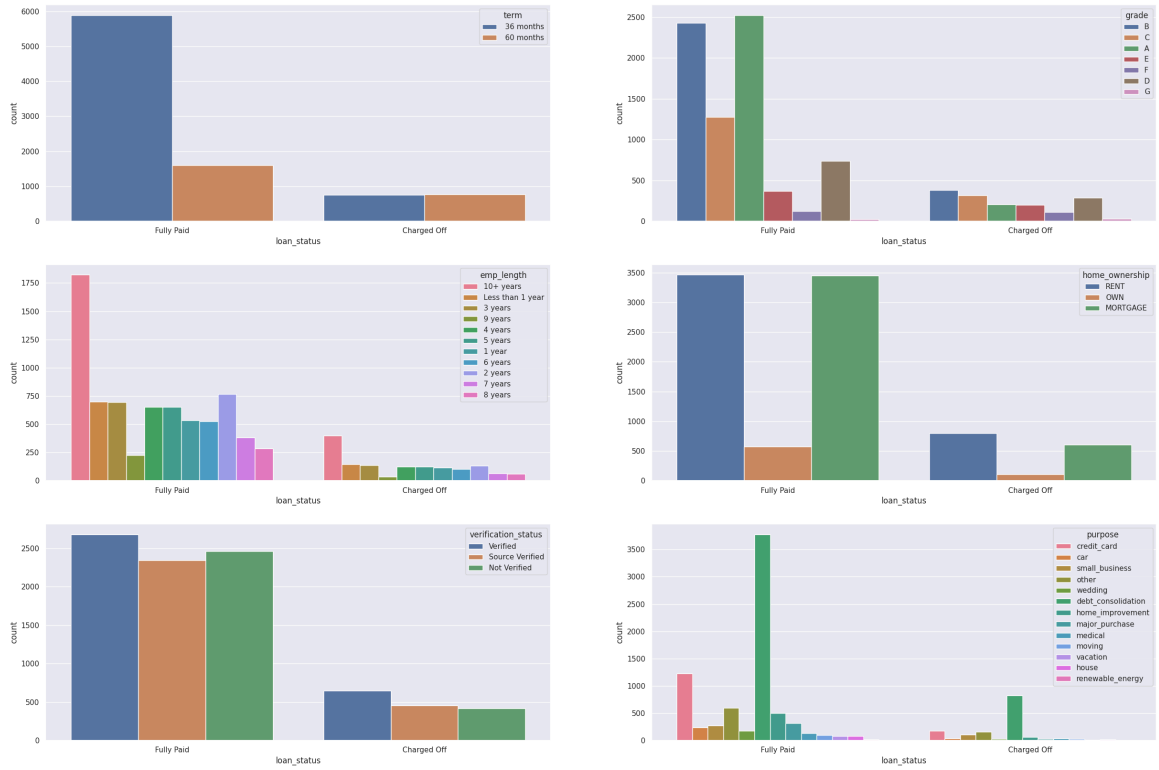

```
plt.figure(figsize = (18,5))
plt.subplot(1,2,1)
sns.countplot(x='grade',data = Lingyi_LC_df,order = Lingyi_LC_df['grade'].value_counts().index)
```

```
<Axes: xlabel='grade', ylabel='count'>
```



```
_,axss = plt.subplots(3,2, figsize=[30,20])
sns.countplot(x='loan_status', hue='term', data=Lingyi_LC_df, ax=axss[0][0])
sns.countplot(x='loan_status', hue='grade', data=Lingyi_LC_df, ax=axss[0][1])
sns.countplot(x='loan_status', hue='emp_length', data=Lingyi_LC_df, ax=axss[1][0])
sns.countplot(x='loan_status', hue='home_ownership', data=Lingyi_LC_df, ax=axss[1][1])
sns.countplot(x='loan_status', hue='verification_status', data=Lingyi_LC_df, ax=axss[2][0])
sns.countplot(x='loan_status', hue='purpose', data=Lingyi_LC_df, ax=axss[2][1])
```

```
<Axes: xlabel='loan_status', ylabel='count'>
```



#Part2:Data Cleaning and Feature Preprocesing

```
Lingyi_LC_df.head()
```

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	1077501	5000	5000	4975.0	36 months	10.65	162.87
1	1077430	2500	2500	2500.0	60 months	15.27	59.83
2	1077175	2400	2400	2400.0	36 months	15.96	84.33
3	1076863	10000	10000	10000.0	36 months	13.49	339.31
4	1075269	5000	5000	5000.0	36 months	7.90	156.46

5 rows × 29 columns

```
Lingyi_LC_df.describe()
```

	id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	insta
count	9.004000e+03	9004.000000	9004.000000	9004.000000	9004.000000	9004.
mean	9.632337e+05	12291.884163	12154.156486	12076.054639	12.126728	357.
std	7.953238e+04	8285.682170	8096.937145	8033.211335	4.195740	227
min	4.581650e+05	1000.000000	1000.000000	750.000000	5.420000	30
25%	8.778840e+05	6000.000000	6000.000000	6000.000000	8.490000	187.
50%	9.879685e+05	10000.000000	10000.000000	10000.000000	11.710000	312
75%	1.033607e+06	16000.000000	16000.000000	15975.000000	15.230000	469.
max	1.077501e+06	35000.000000	35000.000000	35000.000000	24.110000	1288

8 rows × 21 columns

```
class_mapping = {label:idx for idx, label in enumerate(np.unique(Lingyi_LC_df['term']))}
Lingyi_LC_df['term'] = Lingyi_LC_df['term'].map(class_mapping)

class_mapping = {label:idx for idx, label in enumerate(np.unique(Lingyi_LC_df['grade']))}
Lingyi_LC_df['grade'] = Lingyi_LC_df['grade'].map(class_mapping)

class_mapping = {label:idx for idx, label in enumerate(np.unique(Lingyi_LC_df['home_ownership']))}
Lingyi_LC_df['home_ownership'] = Lingyi_LC_df['home_ownership'].map(class_mapping)

class_mapping = {label:idx for idx, label in enumerate(np.unique(Lingyi_LC_df['verification_status
Lingyi_LC_df['verification_status'] = Lingyi_LC_df['verification_status'].map(class_mapping)

class_mapping = {label:idx for idx, label in enumerate(np.unique(Lingyi_LC_df['purpose']))}
Lingyi_LC_df['purpose'] = Lingyi_LC_df['purpose'].map(class_mapping)

class_mapping = {label:idx for idx, label in enumerate(np.unique(Lingyi_LC_df['addr_state']))}
Lingyi_LC_df['addr_state'] = Lingyi_LC_df['addr_state'].map(class_mapping)

class_mapping = {'Fully Paid' : 0, 'Charged Off' : 1}
Lingyi_LC_df['loan_status'] = Lingyi_LC_df['loan_status'].map(class_mapping)

Lingyi_LC_df.head()
```

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	1077501	5000	5000	4975.0	0	10.65	162.87
1	1077430	2500	2500	2500.0	1	15.27	59.83
2	1077175	2400	2400	2400.0	0	15.96	84.33
3	1076863	10000	10000	10000.0	0	13.49	339.31
4	1075269	5000	5000	5000.0	0	7.90	156.46

5 rows × 29 columns

```
Lingyi_LC_df = Lingyi_LC_df.select_dtypes(include=[np.number]).interpolate().dropna()
# drop high correlation and high variance colums
Lingyi_LC_df = Lingyi_LC_df.drop(["total_pymnt"], axis=1)
Lingyi_LC_df = Lingyi_LC_df.drop(["total_pymnt_inv"], axis=1)
Lingyi_LC_df = Lingyi_LC_df.drop(["total_rec_int"], axis=1)
Lingyi_LC_df = Lingyi_LC_df.drop(["id"], axis=1)
Lingyi_LC_df = Lingyi_LC_df.drop(["total_rec_prncp"], axis=1)

Lingyi_LC_df.head(10)
```

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	hc
0	5000	5000	4975.0	0	10.65	162.87	1	
1	2500	2500	2500.0	1	15.27	59.83	2	
2	2400	2400	2400.0	0	15.96	84.33	2	
3	10000	10000	10000.0	0	13.49	339.31	2	
4	5000	5000	5000.0	0	7.90	156.46	0	
5	3000	3000	3000.0	0	18.64	109.43	4	
6	5600	5600	5600.0	1	21.28	152.39	5	
7	5375	5375	5350.0	1	12.69	121.45	1	
8	6500	6500	6500.0	1	14.65	153.45	2	
9	12000	12000	12000.0	0	12.69	402.54	1	

10 rows × 23 columns

```
from sklearn.model_selection import train_test_split
yPredict = Lingyi_LC_df.loan_status
XClean = Lingyi_LC_df.drop(['loan_status'],axis = 1)
X_train,X_test,y_train,y_test = train_test_split(XClean,yPredict,random_state = 42,test_size =0.25
```

```
X_train.head()
```

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade
7442	14400	14400	14400.0	1	19.29	375.85	4
8145	7000	7000	7000.0	0	5.42	211.12	0
1434	1325	1325	1325.0	0	6.62	40.69	0
2777	12000	12000	12000.0	0	6.62	368.45	0
1867	16000	16000	15975.0	1	15.27	382.92	2

5 rows × 22 columns

```
X_test.head()
```

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade
8718	5600	5600	5350.0000	0	7.49	174.17	0
248	12000	12000	12000.0000	0	8.90	381.04	0
6142	16000	16000	16000.0000	0	6.62	491.26	0
1421	14550	14550	14550.0000	0	11.71	481.26	1
7021	20000	20000	19711.1365	1	18.39	512.13	4

5 rows × 22 columns

```
y_train.head()
```

```
7442    0
8145    0
1434    0
2777    1
1867    0
Name: loan_status, dtype: int64
```

```
y_test.head()
```

```
8718    0
248     1
6142    0
1421    0
7021    0
Name: loan_status, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
#Part3: Model training and Selection
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

classifier_logistic = LogisticRegression()
classifier_KNN = KNeighborsClassifier()
classifier_RF = RandomForestClassifier()
```

```
classifier_logistic.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
from sklearn.metrics import classification_report,confusion_matrix
prediction_Regression = classifier_logistic.predict(X_test)
print(classification_report(y_test,prediction_Regression))
```

```
                  precision    recall  f1-score   support

               0       0.84      0.99      0.91      1872
               1       0.64      0.06      0.10       379

        accuracy                           0.84      2251
       macro avg       0.74      0.52      0.51      2251
    weighted avg       0.80      0.84      0.77      2251
```

```
classifier_RF.fit(X_train,y_train)
```

▾ RandomForestClassifier
RandomForestClassifier()

```
prediction_RF = classifier_RF.predict(X_test)
print(classification_report(y_test,prediction_RF))
```

```
                  precision    recall  f1-score   support

               0       0.84      0.99      0.91      1872
               1       0.57      0.07      0.13       379

        accuracy                           0.83      2251
       macro avg       0.71      0.53      0.52      2251
    weighted avg       0.80      0.83      0.78      2251
```

```
classifier_KNN.fit(X_train,y_train)
```

▾ KNeighborsClassifier
KNeighborsClassifier()

```
prediction_KNN= classifier_KNN.predict(X_test)
print(classification_report(y_test,prediction_KNN))
```

```
                  precision    recall  f1-score   support

               0       0.85      0.96      0.90      1872
               1       0.43      0.15      0.22       379

        accuracy                           0.82      2251
       macro avg       0.64      0.55      0.56      2251
    weighted avg       0.78      0.82      0.79      2251
```

```
#Cross Validation 5 folders


from sklearn import model_selection
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
```

```
[0.83641747 0.83271651 0.83641747 0.82666667 0.83851852]
Model Accuracy ofLogistic Regressionis<built-in method mean of numpy.ndarray object at 0x7fe1f
[0.80977054 0.80754996 0.80977054 0.80148148 0.81555556]
Model Accuracy ofKNNis<built-in method mean of numpy.ndarray object at 0x7fe1f8033a50>
[0.83197631 0.8238342  0.82753516 0.82666667 0.83185185]
Model Accuracy ofRandom Forestis<built-in method mean of numpy.ndarray object at 0x7fe1f805219
```

```
#SVM model


from sklearn.svm import SVC

classifier_SVC = SVC()
cv_score =model_selection.cross_val_score(classifier_SVC,X_train,y_train)
print('Model Accuracy of SVM is:'+ str(cv_score.mean()))
```

```
Model Accuracy of SVM is:0.8319266387038409
```

```
#Neural Network
```