Building Machine Learning Model

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv(r'C:\Users\Lenovo\Downloads\ml_new')
```

In [3]:

```
df.drop('Unnamed: 0',axis=1,inplace=True)
df.head()
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	1077501	1296599.0	5000.0	5000.0	4975.0	36	10.65	162.87
1	1077430	1314167.0	2500.0	2500.0	2500.0	60	15.27	59.83
2	1077175	1313524.0	2400.0	2400.0	2400.0	36	15.96	84.33
3	1076863	1277178.0	10000.0	10000.0	10000.0	36	13.49	339.31
4	1075358	1311748.0	3000.0	3000.0	3000.0	60	12.69	67.79

5 rows × 47 columns

In [4]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42350 entries, 0 to 42349
Data columns (total 47 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   id                        42350 non-null  int64  
 1   member_id                 42350 non-null  float64
 2   loan_amnt                 42350 non-null  float64
 3   funded_amnt               42350 non-null  float64
 4   funded_amnt_inv           42350 non-null  float64
 5   term                      42350 non-null  int64  
 6   int_rate                  42350 non-null  float64
 7   installment               42350 non-null  float64
 8   grade                     42350 non-null  object 
 9   sub_grade                 42350 non-null  object 
 10  emp_title                 42350 non-null  object 
 11  emp_length                42350 non-null  object 
 12  home_ownership            42350 non-null  object 
 13  annual_inc                42350 non-null  float64
 14  verification_status       42350 non-null  object 
 15  issue_d                   42350 non-null  object 
 16  loan_status               42350 non-null  object 
 17  purpose                   42350 non-null  object 
 18  title                     42350 non-null  object 
 19  zip_code                  42350 non-null  object 
 20  addr_state                42350 non-null  object 
 21  dti                       42350 non-null  float64
 22  delinq_2yrs               42350 non-null  float64
 23  earliest_cr_line          42350 non-null  object 
 24  fico_range_low            42350 non-null  float64
 25  fico_range_high           42350 non-null  float64
 26  inq_last_6mths            42350 non-null  float64
 27  open_acc                  42350 non-null  float64
 28  pub_rec                   42350 non-null  float64
 29  revol_bal                 42350 non-null  float64
 30  revol_util                42350 non-null  float64
 31  total_acc                 42350 non-null  float64
 32  out_prncp                 42350 non-null  float64
 33  out_prncp_inv             42350 non-null  float64
 34  total_pymnt               42350 non-null  float64
 35  total_pymnt_inv           42350 non-null  float64
 36  total_rec_prncp           42350 non-null  float64
 37  total_rec_int             42350 non-null  float64
 38  total_rec_late_fee        42350 non-null  float64
 39  recoveries                42350 non-null  float64
 40  collection_recovery_fee   42350 non-null  float64
 41  last_pymnt_d              42350 non-null  object 
 42  last_pymnt_amnt           42350 non-null  float64
 43  last_credit_pull_d        42350 non-null  object 
 44  last_fico_range_high      42350 non-null  float64
 45  last_fico_range_low       42350 non-null  float64
 46  pub_rec_bankruptcies      42350 non-null  float64
dtypes: float64(30), int64(2), object(15)
memory usage: 15.2+ MB
```

Data Preparation

In [5]:

```
df.columns
```

Out[5]:

```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title',
       'emp_length', 'home_ownership', 'annual_inc', 'verification_status',
       'issue_d', 'loan_status', 'purpose', 'title', 'zip_code', 'addr_stat
e',
       'dti', 'delinq_2yrs', 'earliest_cr_line', 'fico_range_low',
       'fico_range_high', 'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_ba
l',
       'revol_util', 'total_acc', 'out_prncp', 'out_prncp_inv', 'total_pymn
t',
       'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'last_fico_range_high', 'last_fico_range_low', 'pub_rec_bankruptcie
s'],
      dtype='object')
```

In [6]:

```
col_drop=['id','member_id','funded_amnt','funded_amnt_inv','sub_grade','emp_title','iss
          'out_prncp','out_prncp_inv', 'total_pymnt','total_pymnt_inv','total_rec_prncp
          'total_rec_late_fee','recoveries','collection_recovery_fee','last_pymnt_d','l
```

In [7]:

```
df.drop(columns=col_drop,axis=1,inplace=True)
```

In [8]:

```
df.columns
```

Out[8]:

```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'emp_lengt
h',
       'home_ownership', 'annual_inc', 'verification_status', 'loan_status',
       'purpose', 'title', 'addr_state', 'dti', 'delinq_2yrs',
       'earliest_cr_line', 'fico_range_low', 'fico_range_high',
       'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util',
       'total_acc', 'last_credit_pull_d', 'last_fico_range_high',
       'last_fico_range_low', 'pub_rec_bankruptcies'],
      dtype='object')
```

In [9]:

```
1 df['title'].value_counts()
```

Out[9]:

```
Debt Consolidation         2255
Debt Consolidation Loan    1755
Personal Loan               706
Consolidation               545
debt consolidation          532
                           ...
Health and well being         1
AB Consolidation Loan         1
My Debt Conso Loan            1
GM card re-fi                 1
VISA                          1
Name: title, Length: 21173, dtype: int64
```

In [10]:

```
1 df['addr_state'].value_counts()
```

Out[10]:

```
CA    7398
NY    4045
FL    3093
TX    2896
NJ    1976
IL    1667
PA    1643
GA    1500
VA    1481
MA    1432
OH    1321
MD    1118
AZ     928
WA     886
CO     854
NC     827
CT     812
MI     794
MO     759
MN     648
NV     526
WI     514
SC     489
AL     482
OR     467
LA     460
KY     356
OK     316
KS     296
UT     278
AR     260
DC     222
RI     208
NM     205
NH     188
WV     186
HI     180
DE     135
MT      96
WY      87
AK      86
SD      66
VT      57
TN      32
MS      26
IN      19
IA      12
NE      11
ID       9
ME       3
Name: addr_state, dtype: int64
```

In [11]:

```
col_drop=['title','addr_state']
df.drop(columns=col_drop,axis=1,inplace=True)
```

In [12]:

```
df.columns
```

Out[12]:

```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'emp_lengt
h',
       'home_ownership', 'annual_inc', 'verification_status', 'loan_status',
       'purpose', 'dti', 'delinq_2yrs', 'earliest_cr_line', 'fico_range_lo
w',
       'fico_range_high', 'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_ba
l',
       'revol_util', 'total_acc', 'last_credit_pull_d', 'last_fico_range_hig
h',
       'last_fico_range_low', 'pub_rec_bankruptcies'],
      dtype='object')
```

Checking FICO Score Columns

In [13]:

```
df['fico_range_high'].unique()
```

Out[13]:

```
array([739., 744., 694., 699., 734., 664., 679., 729., 714., 709., 724.,
       669., 674., 764., 689., 759., 684., 704., 794., 754., 719., 769.,
       749., 774., 784., 779., 799., 814., 804., 819., 789., 809., 829.,
       824., 634., 629., 654., 659., 649., 644., 639., 614., 624., 619.])
```

In [14]:

```
df['fico_range_low'].unique()
```

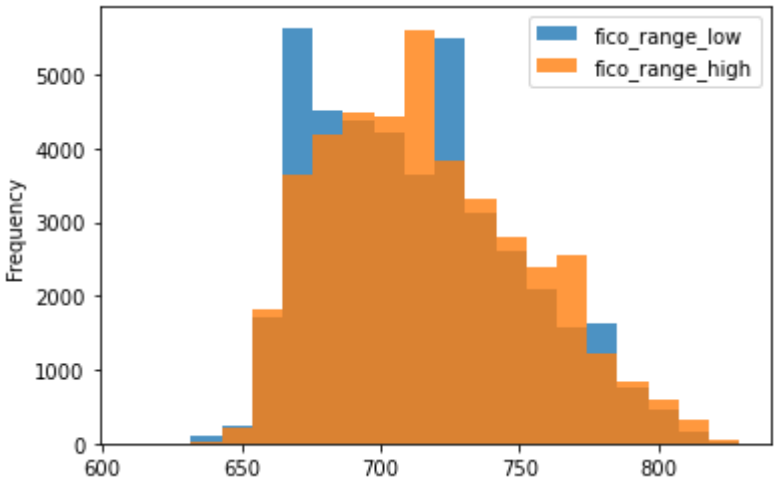
Out[14]:

```
array([735., 740., 690., 695., 730., 660., 675., 725., 710., 705., 720.,
       665., 670., 760., 685., 755., 680., 700., 790., 750., 715., 765.,
       745., 770., 780., 775., 795., 810., 800., 815., 785., 805., 825.,
       820., 630., 625., 650., 655., 645., 640., 635., 610., 620., 615.])
```

In [15]:

```
plt.figure(figsize=(10,8))
df[['fico_range_low','fico_range_high']].plot.hist(alpha=0.8, bins=20)
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



In [16]:

```
#FICO Score Average
df['fico_range_avg']=(df['fico_range_low']+df['fico_range_high'])/2
```

In [17]:

```
1 df[['fico_range_low','fico_range_high','fico_range_avg']].sample(10)
```

Out[17]:

	fico_range_low	fico_range_high	fico_range_avg
8007	695.0	699.0	697.0
23961	705.0	709.0	707.0
29042	775.0	779.0	777.0
33237	665.0	669.0	667.0
24222	665.0	669.0	667.0
12636	695.0	699.0	697.0
6517	705.0	709.0	707.0
19967	690.0	694.0	692.0
22345	780.0	784.0	782.0
2623	710.0	714.0	712.0

In [18]:

```
1 df.drop(columns=['fico_range_low','fico_range_high','last_fico_range_high','last_fico_r
```

In [19]:

```
1 df.columns
```

Out[19]:

```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'emp_lengt
h',
       'home_ownership', 'annual_inc', 'verification_status', 'loan_status',
       'purpose', 'dti', 'delinq_2yrs', 'earliest_cr_line', 'inq_last_6mth
s',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'last_credit_pull_d', 'pub_rec_bankruptcies', 'fico_range_avg'],
      dtype='object')
```

In [20]:

```
1 df.describe()
```

Out[20]:

	loan_amnt	term	int_rate	installment	annual_inc	dti	
count	42350.000000	42350.000000	42350.000000	42350.000000	4.235000e+04	42350.000000	42
mean	11109.211924	42.220165	12.158697	323.124979	6.920405e+04	13.383605	
std	7409.408311	10.516469	3.707855	208.872033	6.413285e+04	6.723341	
min	500.000000	36.000000	5.420000	15.670000	1.896000e+03	0.000000	
25%	5200.000000	36.000000	9.620000	165.740000	4.000000e+04	8.210000	
50%	9800.000000	36.000000	11.990000	278.410000	5.900000e+04	13.480000	
75%	15000.000000	60.000000	14.720000	428.907500	8.250000e+04	18.690000	
max	35000.000000	60.000000	24.590000	1305.190000	6.000000e+06	29.990000	

In [21]:

```
1 col_drop=['earliest_cr_line','open_acc','pub_rec','total_acc','last_credit_pull_d','pub
2 df.drop(columns=col_drop,axis=1,inplace=True)
```

In [22]:

```
1 df['loan_status'].value_counts()
```

Out[22]:

```
Fully Paid                                             33542
Charged Off                                             5567
Does not meet the credit policy. Status:Fully Paid      1952
Does not meet the credit policy. Status:Charged Off      742
Current                                                  513
In Grace Period                                           16
Late (31-120 days)                                        12
Late (16-30 days)                                          5
Default                                                    1
Name: loan_status, dtype: int64
```

In [23]:

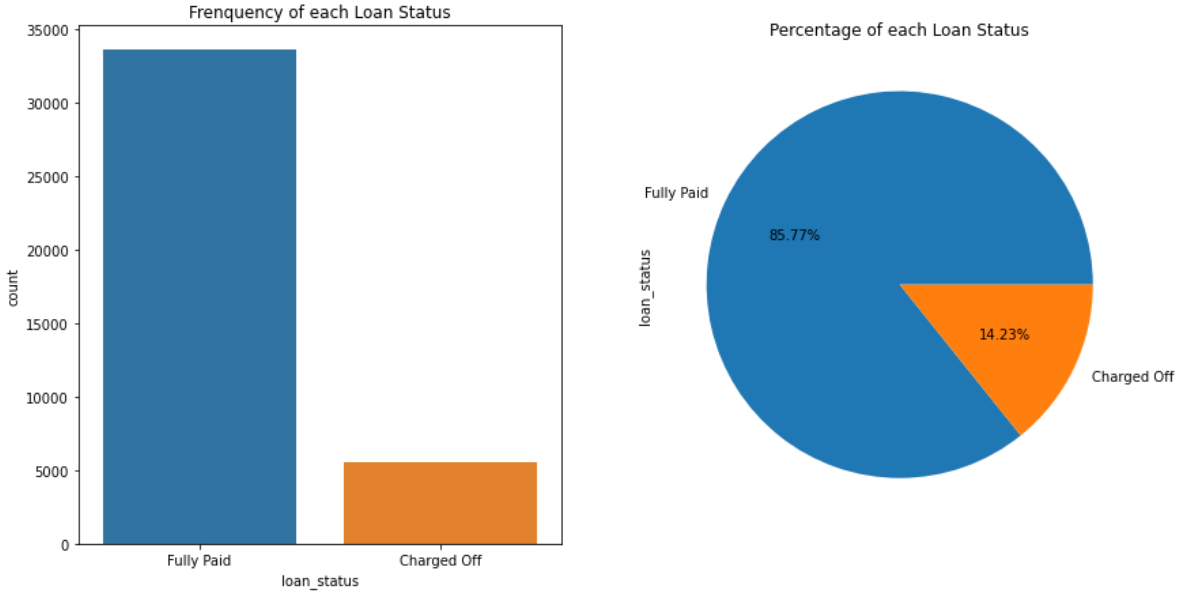
```
1 df = df[(df['loan_status']=='Fully Paid') | (df['loan_status']=='Charged Off')]
```

In [24]:

```
1 df.reset_index(drop=True, inplace=True)
```

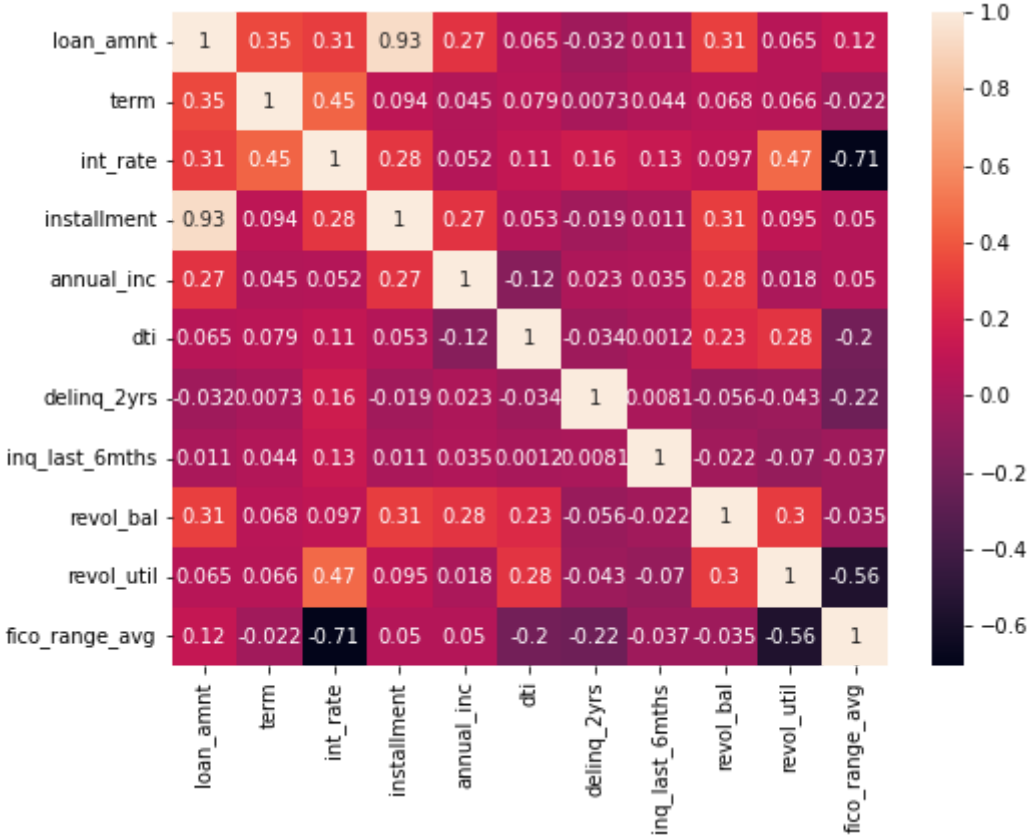
In [25]:

```
fig, ax = plt.subplots(1,2, figsize=(14,7))
sns.countplot(x='loan_status', data=df, ax=ax[0])
ax[0].set_title('Frenquency of each Loan Status')
df['loan_status'].value_counts().plot(kind='pie', ax=ax[1], autopct='%1.2f%%')
ax[1].set_title('Percentage of each Loan Status')
plt.show()
```



In [26]:

```
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



Target Variable

In [27]:

```
x_df = df.drop(columns=['loan_status'], axis=1)
y_df = df['loan_status']
```

Encode Categorical Columns

In [28]:

```
cat_var=[]
for i in x_df.columns:
    if x_df[i].dtype=='object':
        print(i)
        cat_var.append(i)
```

```
grade
emp_length
home_ownership
verification_status
purpose
```

In [29]:

```
for x in cat_var:
    cat_list='var_'+x
    cat_list=pd.get_dummies(x_df[x],prefix=x)
    df1=x_df.join(cat_list)
    x_df=df1
```

In [30]:

```
x_df.shape
```

Out[30]:

(39109, 56)

In [31]:

```
x_df.sample()
```

Out[31]:

	loan_amnt	term	int_rate	installment	grade	emp_length	home_ownership	annual_inc
32536	25000.0	36	12.18	832.5	B	3 years	OWN	100000.0

1 rows × 56 columns

In [32]:

```
x_df.drop(columns=cat_var,axis=1,inplace=True)
```

In [33]:

```
x_df.head()
```

Out[33]:

	loan_amnt	term	int_rate	installment	annual_inc	dti	delinq_2yrs	inq_last_6mths	revol_
0	5000.0	36	10.65	162.87	24000.0	27.65	0.0	1.0	136
1	2500.0	60	15.27	59.83	30000.0	1.00	0.0	5.0	16
2	2400.0	36	15.96	84.33	12252.0	8.72	0.0	2.0	29
3	10000.0	36	13.49	339.31	49200.0	20.00	0.0	1.0	55
4	5000.0	36	7.90	156.46	36000.0	11.20	0.0	3.0	79

5 rows × 51 columns

Training & Testing Data

In [34]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [35]:

```
x_train,x_test,y_train,y_test = train_test_split(x_df, y_df, test_size=0.30)
```

In [36]:

```
x_train.shape
```

Out[36]:

```
(27376, 51)
```

In [37]:

```
x_test.shape
```

Out[37]:

```
(11733, 51)
```

In [38]:

```
y_train.shape
```

Out[38]:

```
(27376,)
```

In [39]:

```
y_test.shape
```

Out[39]:

```
(11733,)
```

In [40]:

```
1 x_train.sample(5)
```

Out[40]:

	loan_amnt	term	int_rate	installment	annual_inc	dti	delinq_2yrs	inq_last_6mths	r
19349	18000.0	60	12.68	406.62	154000.0	16.78	0.0	0.0	
23969	8700.0	36	15.58	304.07	60000.0	4.30	3.0	0.0	
1377	9000.0	36	10.65	293.16	24600.0	19.76	0.0	1.0	
8700	35000.0	60	18.39	896.22	104000.0	10.58	1.0	2.0	
36718	20000.0	36	14.74	690.74	114000.0	15.27	0.0	1.0	

5 rows × 51 columns

In [41]:

```
1 y_train.sample(5)
```

Out[41]:

```
37325    Fully Paid
28715    Fully Paid
26874    Fully Paid
10347    Fully Paid
14983    Fully Paid
Name: loan_status, dtype: object
```

Logistic Regression

In [42]:

```
1 log_reg=LogisticRegression(max_iter=1000)
```

In [43]:

```
1 log_reg.fit(x_train,y_train)
```

Out[43]:

```
LogisticRegression(max_iter=1000)
```

In [44]:

```
1 y_pred=log_reg.predict(x_test)
```

Training Accuracy Score

In [45]:

```
1 print('Accuracy is',log_reg.score(x_train,y_train))
```

```
Accuracy is 0.8573202805376973
```

Testing Accuracy Score ¶

In [46]:

```
1 print('Accuracy is',log_reg.score(x_test,y_test))
```

```
Accuracy is 0.8584334782238132
```

In [ ]:

```
1
```

In [ ]:

```
1
```