Loan_Prediction

June 6, 2023

[30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[31]:
```
df=pd.read_csv("train.csv")  #train data
```

[32]:
```
train_data=df.copy()
```

[45]:
```
df.head(5)
```

[45]:
```
    Loan_ID  Gender  Married  Dependents  Education  Self_Employed  \
0  LP001002       1        0           0        0.0              0
1  LP001003       1        1           1        0.0              0
2  LP001005       1        1           0        0.0              1
3  LP001006       1        1           0        NaN              0
4  LP001008       1        0           0        0.0              0

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0       120.0             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History  Property_Area  Loan_Status
0             1.0            0.0            1
1             1.0            1.0            0
2             1.0            0.0            1
3             1.0            0.0            1
4             1.0            0.0            1
```

[46]:
```
df.shape
```

[46]: (614, 13)

[47]:
```
df.columns
```

[47]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

[34]:
```
df.describe()
```

[34]:
```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       614.000000         614.000000  592.000000         600.00000
mean       5403.459283        1621.245798  146.412162         342.00000
std        6109.041673        2926.248369   85.587325          65.12041
min         150.000000           0.000000    9.000000          12.00000
25%        2877.500000           0.000000  100.000000         360.00000
50%        3812.500000        1188.500000  128.000000         360.00000
75%        5795.000000        2297.250000  168.000000         360.00000
max       81000.000000       41667.000000  700.000000         480.00000

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

[35]:
```
#Filling Missing Values in train Data
```

[36]:
```
df.isnull().sum()
```

[36]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[37]:
```
df["Gender"].fillna(df["Gender"].mode()[0],inplace=True)
df["Married"].fillna(df["Married"].mode()[0],inplace=True)
df["Dependents"].fillna(df["Dependents"].mode()[0],inplace=True)
df["Self_Employed"].fillna(df["Self_Employed"].mode()[0],inplace=True)
df["Credit_History"].fillna(df["Credit_History"].mode()[0],inplace=True)
df["LoanAmount"].fillna(df["LoanAmount"].mode()[0],inplace=True)
df["Loan_Amount_Term"].fillna(df["Loan_Amount_Term"].mode()[0],inplace=True)
```

[38]:
```
df["Education"].fillna(df["Education"].mode()[0],inplace=True)
```

[39]:
```
df.isnull().sum()
```

[39]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

[40]:
```
df.dtypes
```

[40]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

1 describe features

1) Loan_ID : Loan id of applicant

2) gender : The gender of applicant female =0 and male =1
3) Married : married status single or married
4) Dependents =0: Indicates that the loan applicant has no dependents.

   1: Indicates that the loan applicant has one dependent.

   2: Indicates that the loan applicant has two dependents.

   3+: Indicates that the loan applicant has three or more dependents.
5) Education :applicant graduate or not graduate
6) Self_Employed :check applicant has self employee or not : "No":0 and "Yes":1
7) ApplicantIncome : Applicant monthly salary
8) CoapplicantIncome : CoApplicant monthly salary
9) LoanAmount :Loan amount in thousands
10) Loan_Amount_Term:Term of loan in months
11) Credit_History : credit history meets guidelines
12) Property_Area Urban/ Semi Urban/ Rural
13) Loan_Status Loan approved (Y/N)

[41]:
```
#Replacing the categorical values
```

[42]:
```
#converting string values(Categorical Values) to integer
df.Gender=df.Gender.map({"Female":0,"Male":1})

df.Married=df.Married.map({"No":0,"Yes":1})

df.Self_Employed=df.Self_Employed.map({"No":0,"Yes":1})

df.Education=df.Education.map({"Not":1,"Graduate":0})

df.Property_Area=df.Property_Area.map({"Urban":0,"Rural":1})

df.Loan_Status=df.Loan_Status.map({"N":0,"Y":1})

df.Dependents=df.Dependents.map({"3+":3,"0":0,"1":1,"2":2})
```

[43]:
```
df.head(10)
```

[43]:
```
    Loan_ID  Gender  Married  Dependents  Education  Self_Employed  \
0  LP001002       1        0           0        0.0              0
1  LP001003       1        1           1        0.0              0
2  LP001005       1        1           0        0.0              1
3  LP001006       1        1           0        NaN              0
4  LP001008       1        0           0        0.0              0
```

```
5  LP001011       1        1           2        0.0              1
6  LP001013       1        1           0        NaN              0
7  LP001014       1        1           3        0.0              0
8  LP001018       1        1           2        0.0              0
9  LP001020       1        1           1        0.0              0

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0       120.0             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0
5             5417             4196.0       267.0             360.0
6             2333             1516.0        95.0             360.0
7             3036             2504.0       158.0             360.0
8             4006             1526.0       168.0             360.0
9            12841            10968.0       349.0             360.0

   Credit_History  Property_Area  Loan_Status
0             1.0            0.0            1
1             1.0            1.0            0
2             1.0            0.0            1
3             1.0            0.0            1
4             1.0            0.0            1
5             1.0            0.0            1
6             1.0            0.0            1
7             0.0            NaN            0
8             1.0            0.0            1
9             1.0            NaN            0
```

[44]:
```
df.dtypes
```

[44]:
```
Loan_ID               object
Gender                 int64
Married                int64
Dependents             int64
Education            float64
Self_Employed          int64
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area        float64
Loan_Status            int64
dtype: object
```

[48]:
```
df = df.drop(columns=['Loan_ID']) ## Dropping Loan ID
```

[49]:
```
df.head(3)
```

[49]:
```
   Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
0       1        0           0        0.0              0             5849
1       1        1           1        0.0              0             4583
2       1        1           0        0.0              1             3000

   CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                0.0       120.0             360.0             1.0
1             1508.0       128.0             360.0             1.0
2                0.0        66.0             360.0             1.0

   Property_Area  Loan_Status
0            0.0            1
1            1.0            0
2            0.0            1
```

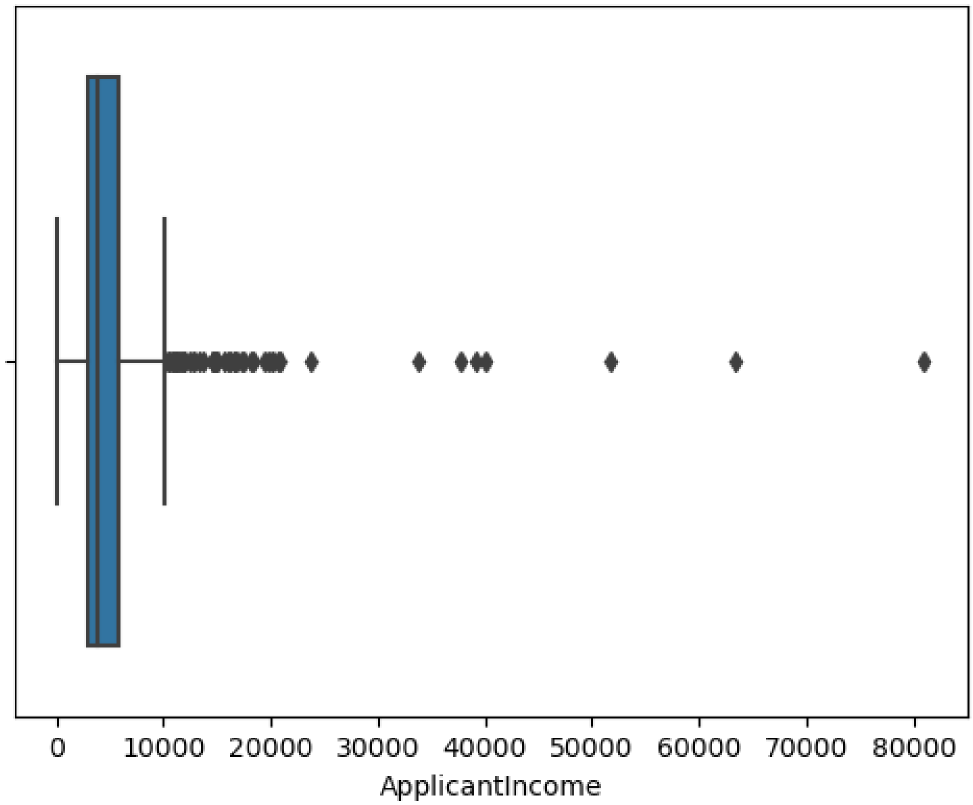
2 EDA

2.0.1 a) Outlers

[ ]:
```
# ApplicantIncome
```

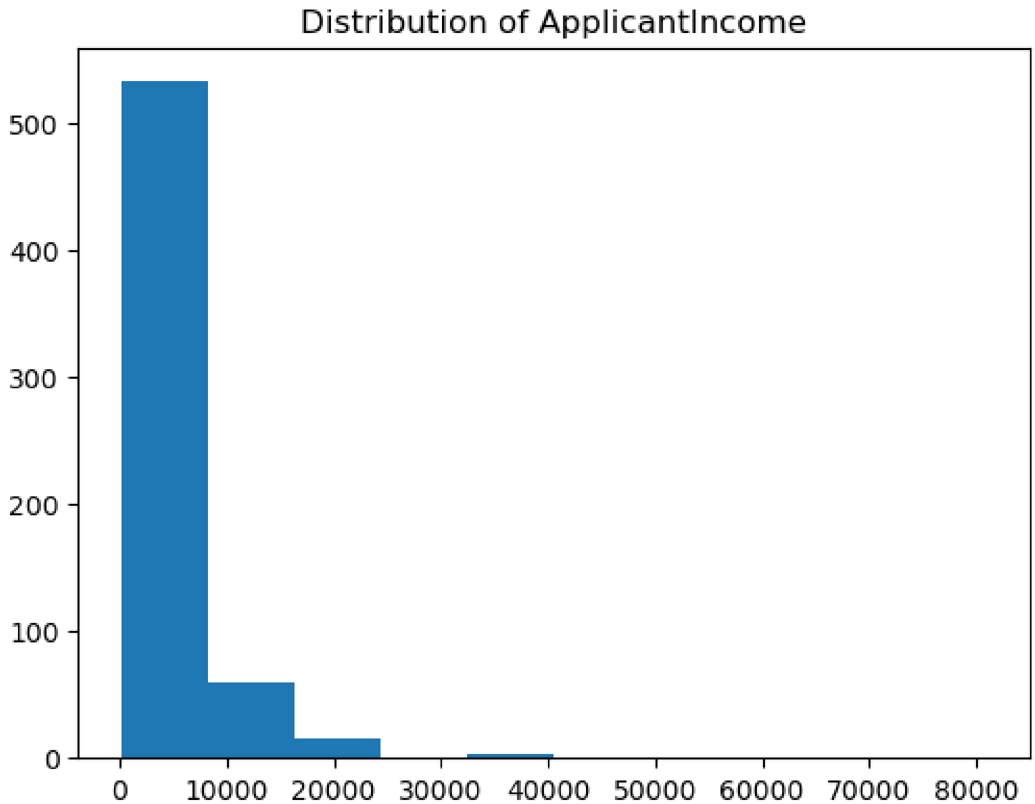
[50]:
```
sns.boxplot(x=df['ApplicantIncome'])
```

[50]:
```
<AxesSubplot:xlabel='ApplicantIncome'>
```



[54]:

```
plt.hist(df['ApplicantIncome'])
plt.title("Distribution of ApplicantIncome")
plt.show()
```



[55]:
```
# observation : ApplicantIncome is + skewed
```

[ ]:
```
# 2) find outlier LoanAmount  using z scores
```

[57]:
```
column_name = 'LoanAmount'
z_scores = np.abs((df[column_name] - df[column_name].mean()) / df[column_name].
 ↪std())
```

[58]:
```
z_score_threshold = 3
```

[59]:
```
outliers = df[z_scores > z_score_threshold]
```

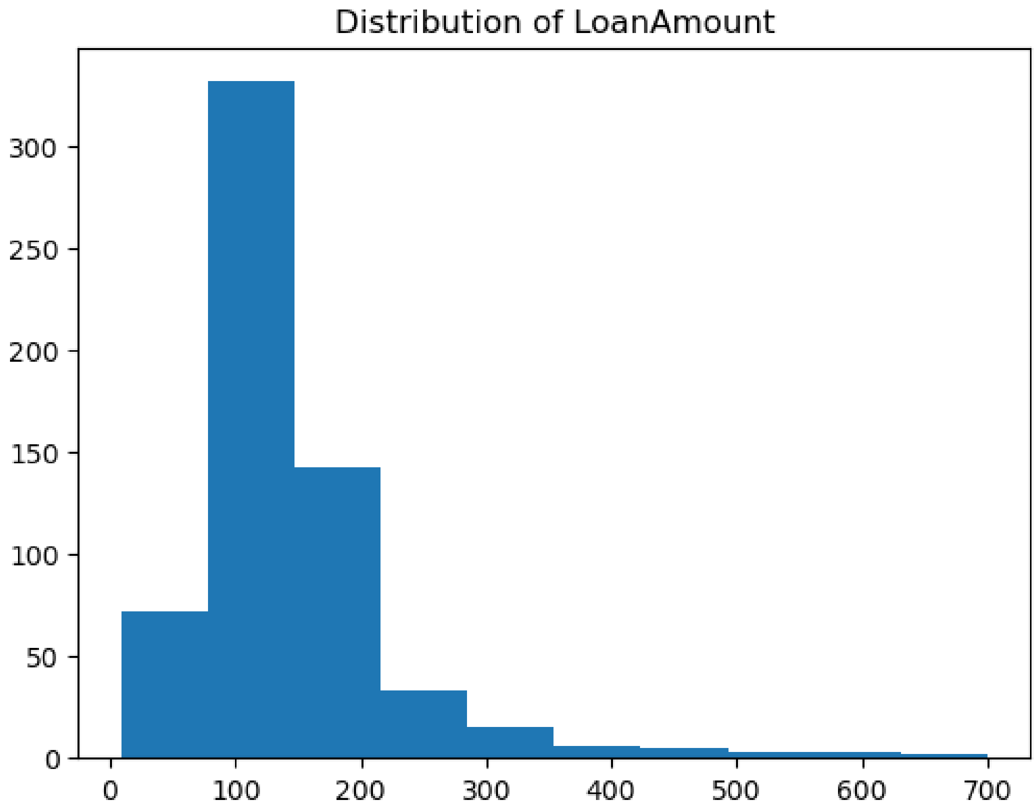
[60]:
```
print(outliers)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
130       1        0           0        0.0              1            20166
155       1        1           3        0.0              0            39999
171       1        1           3        0.0              0            51763
177       1        1           3        0.0              0             5516
278       1        1           0        0.0              0            14583
308       1        0           0        0.0              0            20233
333       1        1           0        0.0              0            63337
```

```
369        1        1          0        0.0              0            19730
432        1        0          0        0.0              0            12876
487        1        1          1        0.0              0            18333
506        1        1          0        0.0              0            20833
523        1        1          2        0.0              1             7948
525        1        1          2        0.0              1            17500
561        0        1          1        0.0              1            19484
604        0        1          1        0.0              0            12000

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
130                0.0       650.0             480.0             1.0
155                0.0       600.0             180.0             0.0
171                0.0       700.0             300.0             1.0
177            11300.0       495.0             360.0             0.0
278                0.0       436.0             360.0             1.0
308                0.0       480.0             360.0             1.0
333                0.0       490.0             180.0             1.0
369             5266.0       570.0             360.0             1.0
432                0.0       405.0             360.0             1.0
487                0.0       500.0             360.0             1.0
506             6667.0       480.0             360.0             1.0
523             7166.0       480.0             360.0             1.0
525                0.0       400.0             360.0             1.0
561                0.0       600.0             360.0             1.0
604                0.0       496.0             360.0             1.0

     Property_Area  Loan_Status
130            0.0            1
155            NaN            1
171            0.0            1
177            NaN            0
278            NaN            1
308            1.0            0
333            0.0            1
369            1.0            0
432            NaN            1
487            0.0            0
506            0.0            1
523            1.0            1
525            1.0            1
561            NaN            1
604            NaN            1
```

[61]:

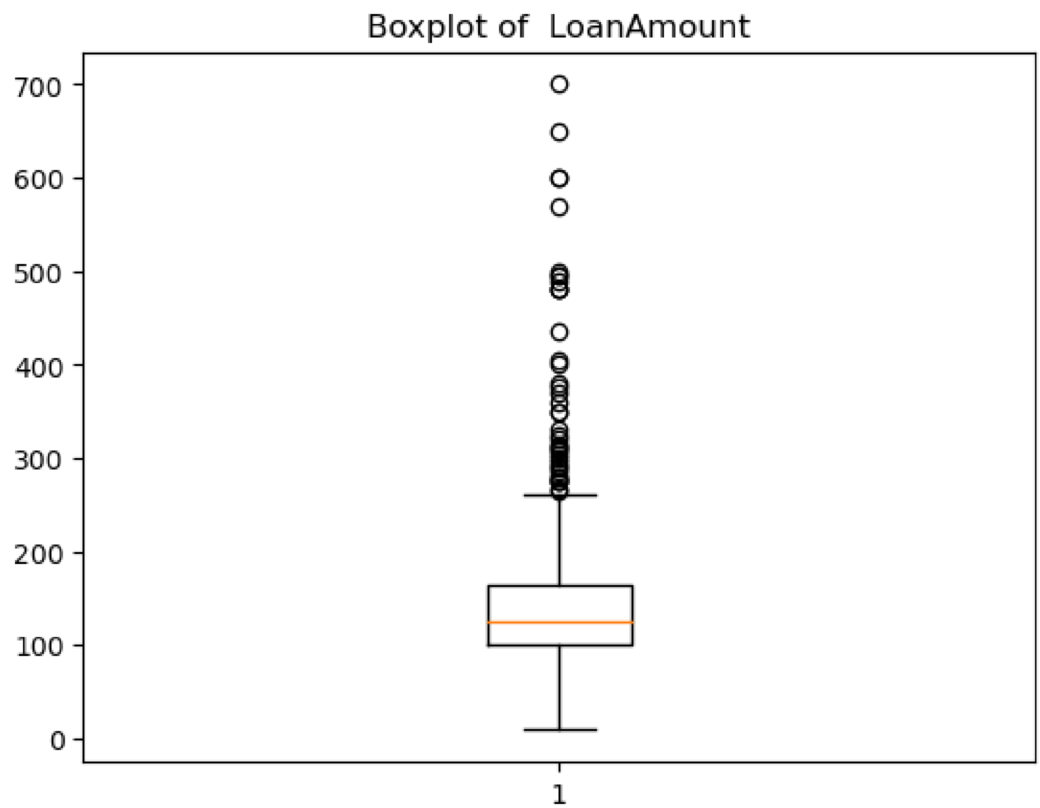
```
plt.hist(df['LoanAmount'])
plt.title("Distribution of LoanAmount")
plt.show()
```



[ ]:
```
# observation :LoanAmount is + skwed data
```

[64]:
```
plt.boxplot(df['LoanAmount'])
plt.title("Boxplot of  LoanAmount")
```

[64]: Text(0.5, 1.0, 'Boxplot of LoanAmount')

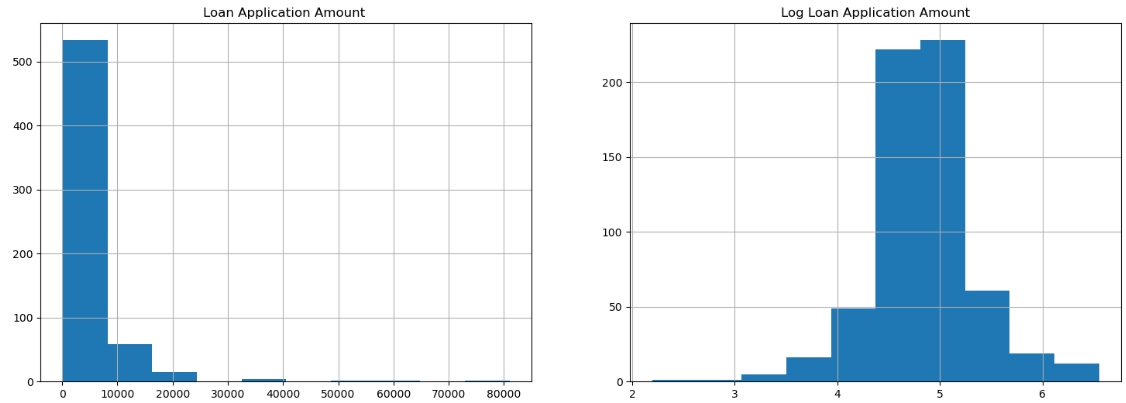


[65]:
```
df.columns
```

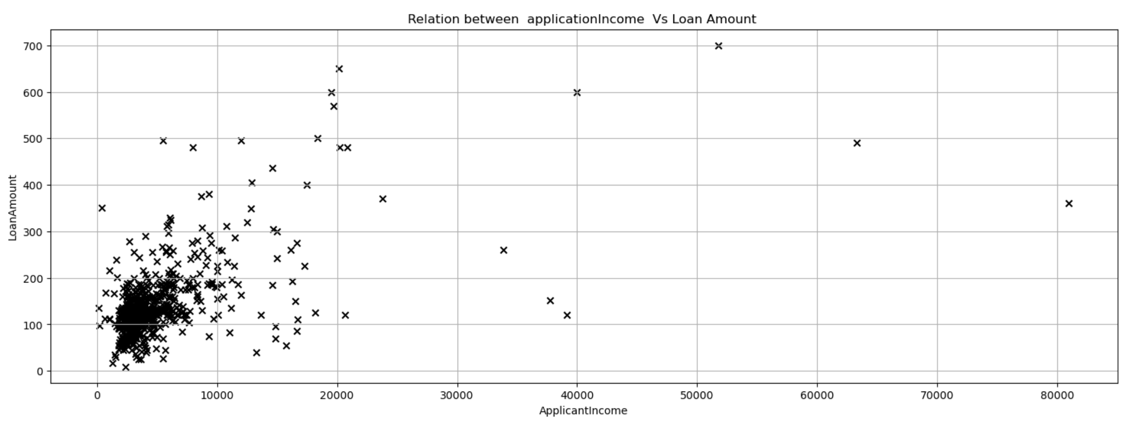
[65]:
```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

2.1 data visualizaton

[157]:
```
plt.figure(figsize=(18,6))
plt.subplot(1,2,1)
df['ApplicantIncome'].hist(bins=10)
plt.title("Loan Application Amount ")
plt.subplot(1,2,2)
plt.grid()
plt.hist(np.log(df['LoanAmount']))
plt.title("Log Loan Application Amount")
plt.show()
```



[159]:
```
plt.figure(figsize=(18,6))
plt.title("Relation between  applicationIncome  Vs Loan Amount ")
plt.grid()
plt.scatter(df['ApplicantIncome'],df['LoanAmount'],c='k',marker='x')
plt.xlabel("ApplicantIncome")
plt.ylabel("LoanAmount")
plt.show()
```

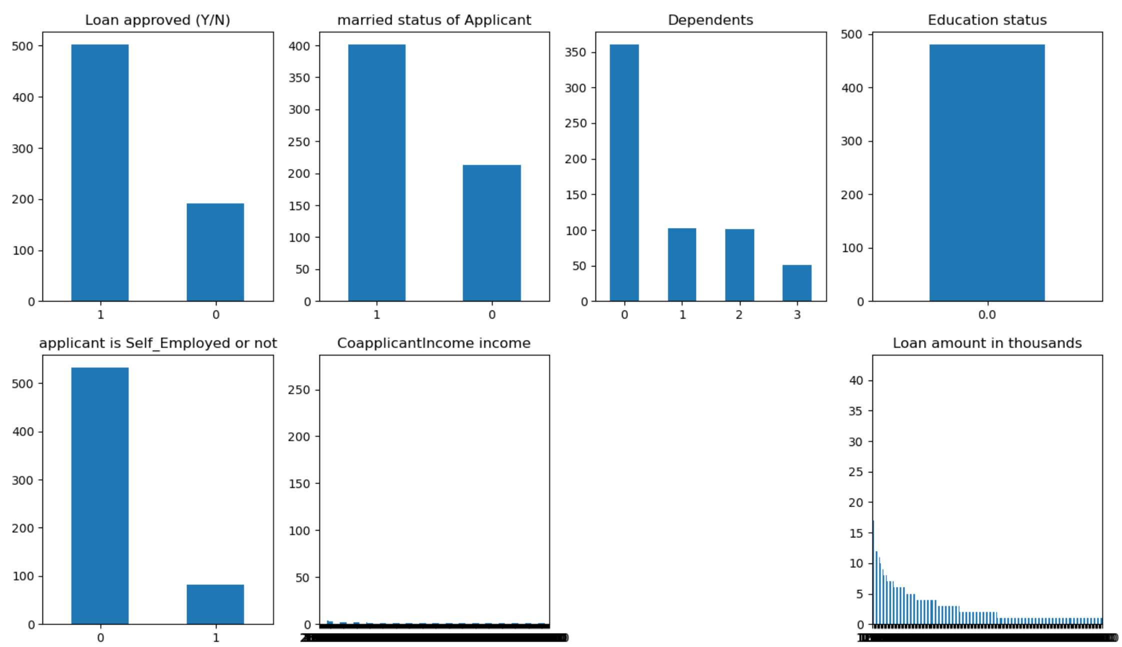


3 Univariate Analysis

[72]:
```
plt.subplot(241)

df['Gender'].value_counts().plot(kind='bar', title='Gender distribution of␣
↪Applicant', figsize=(16,9))

plt.xticks(rotation=0)
```

```
plt.subplot(242)

df['Married'].value_counts().plot(kind='bar', title='married status of␣
↪Applicant', figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(243)

df['Dependents'].value_counts().plot(kind='bar', title='Dependents',␣
↪figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(244)

df['Education'].value_counts().plot(kind='bar', title='Education status',␣
↪figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(245)

df['Self_Employed'].value_counts().plot(kind='bar', title='applicant is␣
↪Self_Employed or not', figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(246)

df['ApplicantIncome'].value_counts().plot(kind='bar', title='Applicant income',␣
↪figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(246)

df['CoapplicantIncome'].value_counts().plot(kind='bar',␣
↪title='CoapplicantIncome income', figsize=(16,9))

plt.xticks(rotation=0)


plt.subplot(248)
```

```
df['LoanAmount'].value_counts().plot(kind='bar', title='Loan amount in␣
↪thousands', figsize=(16,9))

plt.xticks(rotation=0)


plt.subplot(241)

df['Property_Area'].value_counts().plot(kind='bar', title='Urban/ Semi Urban/␣
↪Rural', figsize=(16,9))

plt.xticks(rotation=0)


plt.subplot(241)


df['Loan_Status'].value_counts().plot(kind='bar', title='Loan approved (Y/N)',␣
↪figsize=(16,9))

plt.xticks(rotation=0)



plt.show()
```

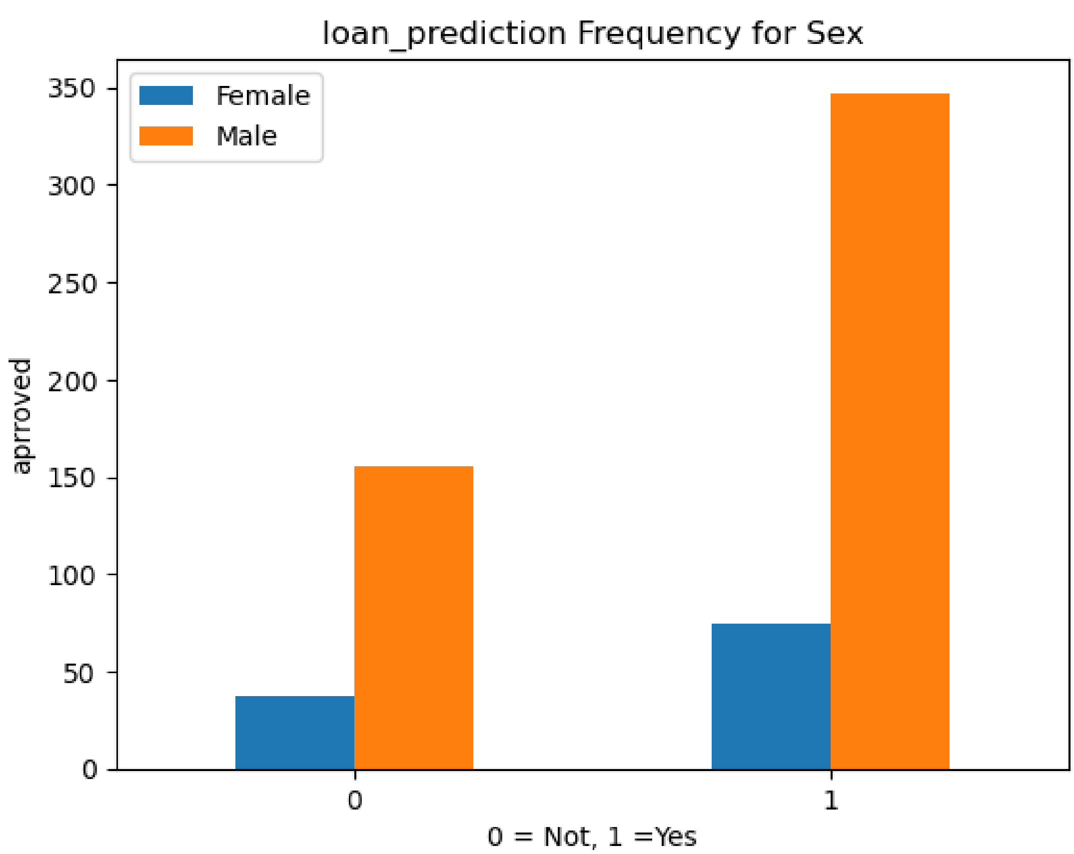


4 observations :

1) Loan_Status: About 2/3rd of applicants have been granted loan.
2) Martial Status: 2/3rd of the population in the dataset is Marred; Married applicants are more likely to be granted loans.
3) Dependents: Majority of the population have zero dependents and are also likely to accepted for loan.
4) Employment: 5/6th of population is not self employed.
5) ApplicantIncome CoapplicantIncome LoanAmount there is no significant relation to Loan_status.

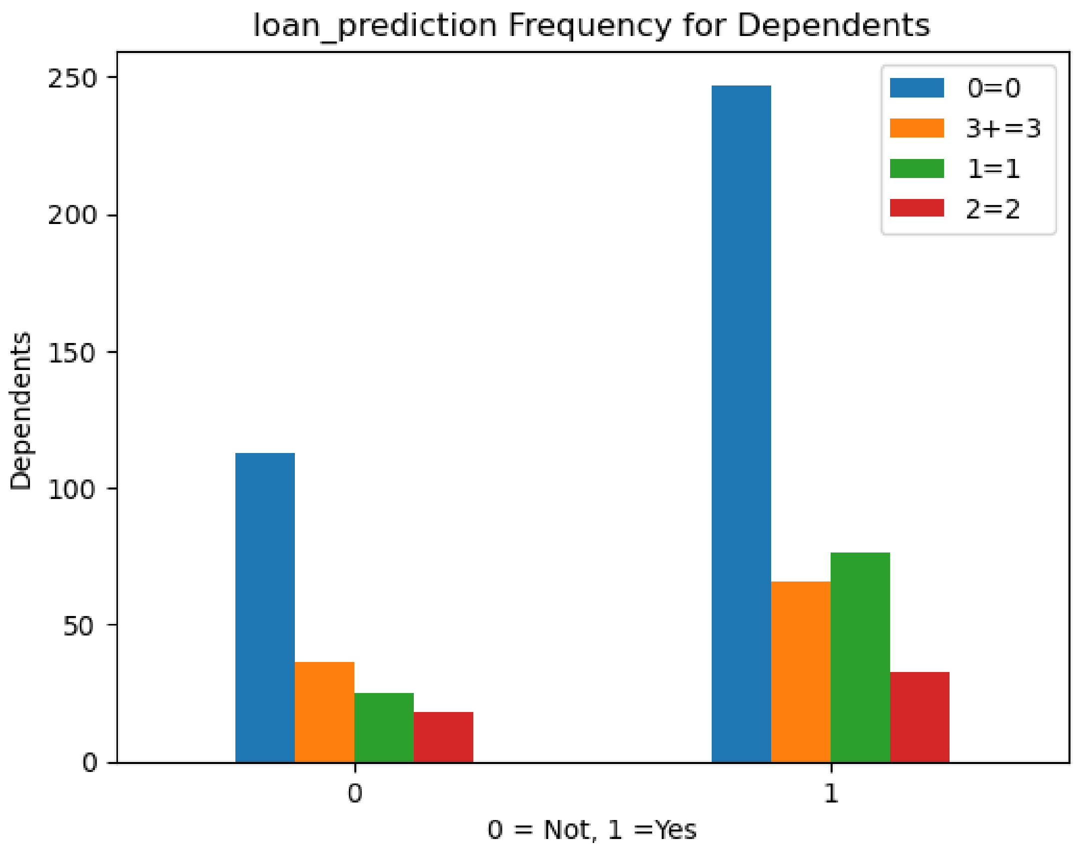
5 Bivariate Analysis

[75]:
```
pd.crosstab(df.Loan_Status, df.Gender).plot(kind='bar')
plt.title("loan_prediction Frequency for Sex")
plt.xlabel("0 = Not, 1 =Yes")
plt.ylabel("aprroved")
plt.legend(["Female", "Male"]);
plt.xticks(rotation=0);
```



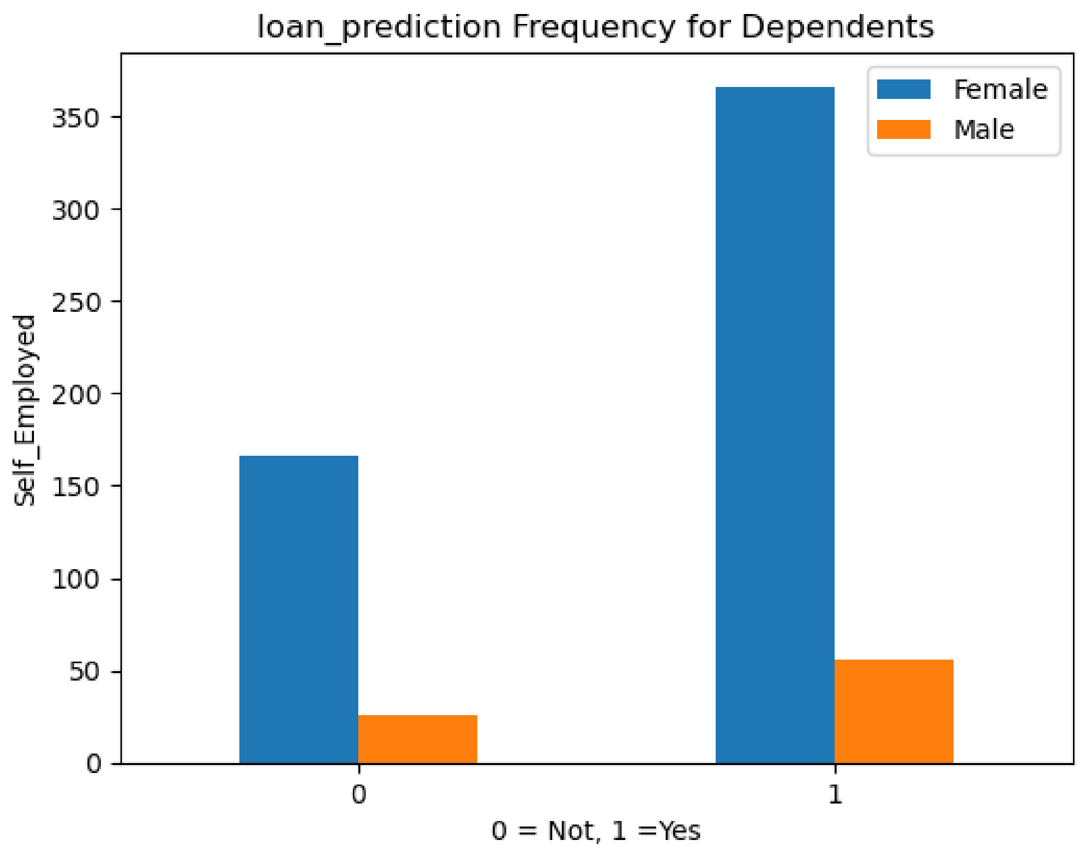
[80]:

```
pd.crosstab(df.Loan_Status, df.Dependents).plot(kind='bar')
plt.title("loan_prediction Frequency for Dependents")
plt.xlabel("0 = Not, 1 =Yes")
plt.ylabel("Dependents")
plt.legend(["0=0","3+=3","1=1","2=2"]);
plt.xticks(rotation=0);
```

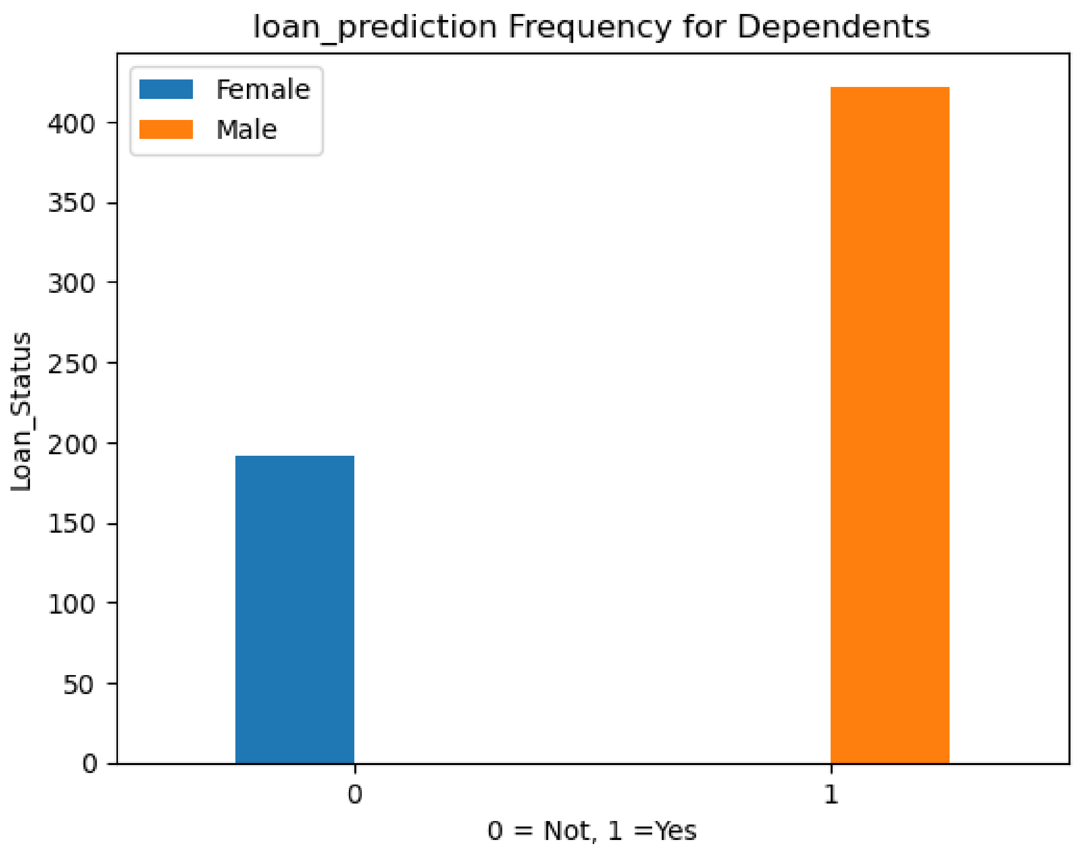


[83]:

```
pd.crosstab(df.Loan_Status, df.Self_Employed).plot(kind='bar')
plt.title("loan_prediction Frequency for Dependents")
plt.xlabel("0 = Not, 1 =Yes")
plt.ylabel("Self_Employed")
plt.legend(["Female", "Male"]);
plt.xticks(rotation=0);
```

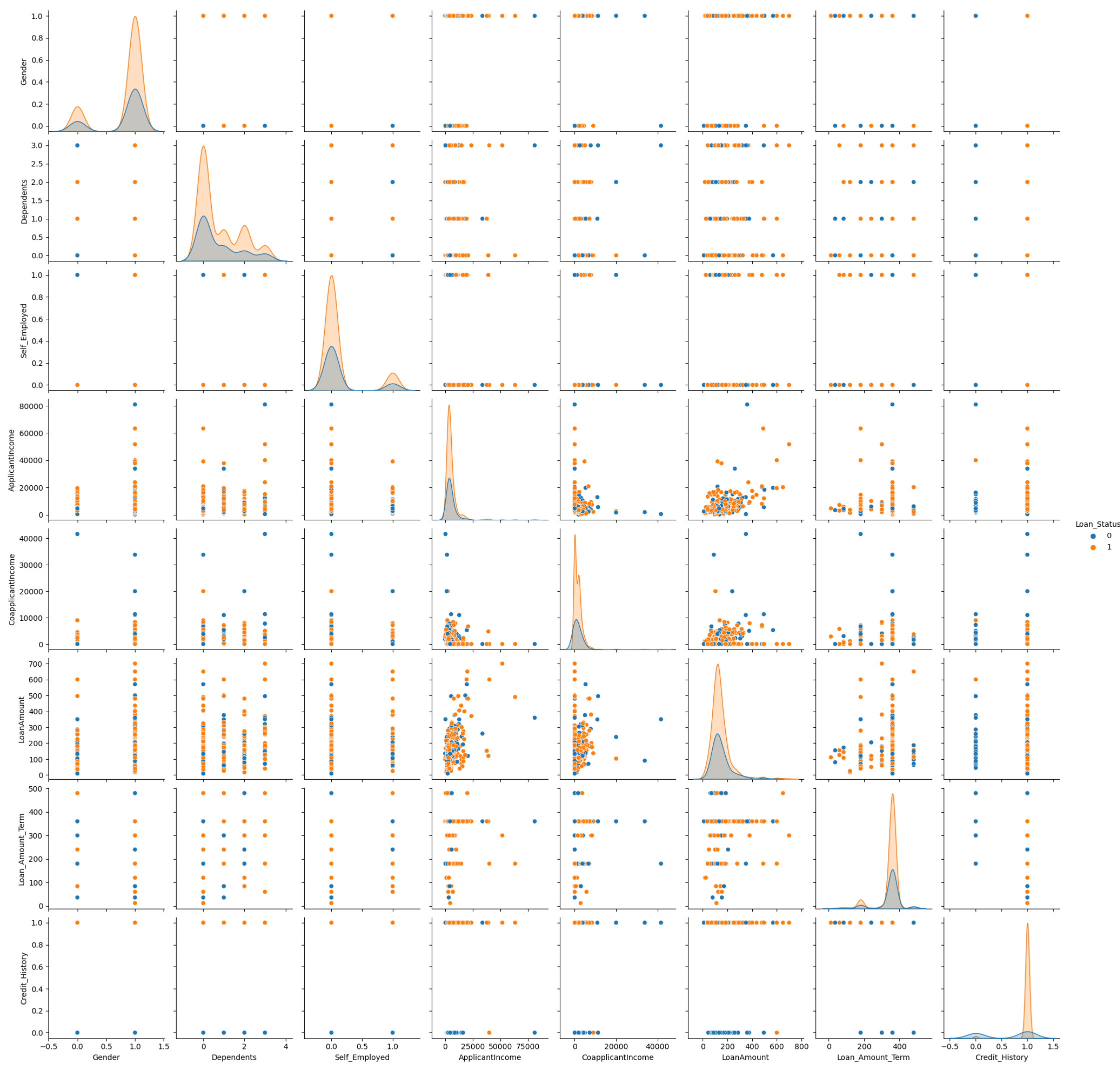


[85]:
```
pd.crosstab(df.Loan_Status, df.Loan_Status).plot(kind='bar')
plt.title("loan_prediction Frequency for Dependents")
plt.xlabel("0 = Not, 1 =Yes")
plt.ylabel("Loan_Status")
plt.legend(["Female", "Male"]);
plt.xticks(rotation=0);
```



6 Multivariate analysis

[93]:
```
# in multivariate analysis   use pairplot
sns.pairplot(df,␣
 ↪vars=['Gender','Dependents','Self_Employed','ApplicantIncome','CoapplicantIncome','LoanAmou
 ↪hue='Loan_Status')
#plt.title('Age, Sex vs. Heart Disease')
plt.show()
```



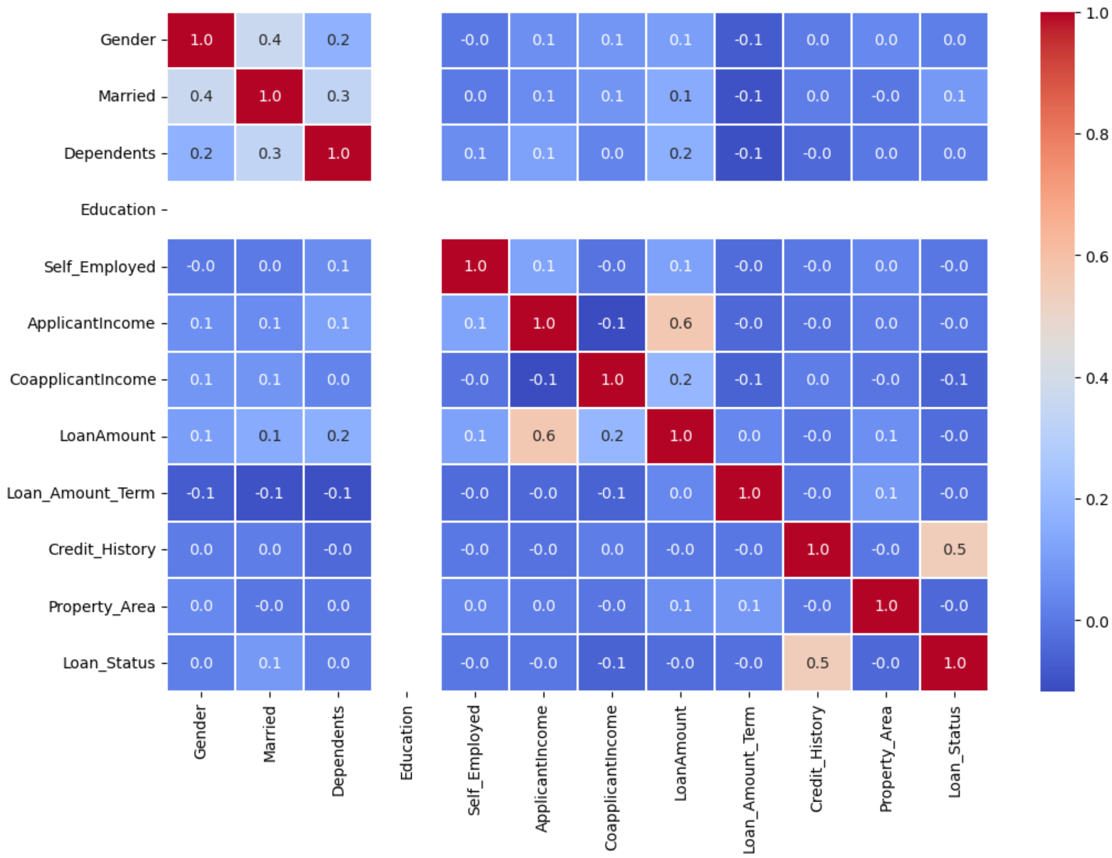
[163]:
```
corr_matrix=df[['Gender','Dependents','Self_Employed','ApplicantIncome','CoapplicantIncome','I
 ↪corr()
print("correlation matrix")
print(corr_matrix)
```

```
correlation matrix
                     Gender  Dependents  Self_Employed  ApplicantIncome  \
Gender             1.000000    0.172914      -0.000525         0.058809
Dependents         0.172914    1.000000       0.056798         0.118202
Self_Employed     -0.000525    0.056798       1.000000         0.127180
ApplicantIncome    0.058809    0.118202       0.127180         1.000000
CoapplicantIncome  0.082912    0.030430      -0.016100        -0.116605
LoanAmount         0.106404    0.163017       0.114971         0.564698
Loan_Amount_Term  -0.074030   -0.103864      -0.033739        -0.046531
Credit_History     0.009170   -0.040160      -0.001550        -0.018615
```

```
                   CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
Gender                      0.082912    0.106404         -0.074030
Dependents                  0.030430    0.163017         -0.103864
Self_Employed              -0.016100    0.114971         -0.033739
ApplicantIncome            -0.116605    0.564698         -0.046531
CoapplicantIncome           1.000000    0.189723         -0.059383
LoanAmount                  0.189723    1.000000          0.037152
Loan_Amount_Term           -0.059383    0.037152          1.000000
Credit_History              0.011134   -0.000250         -0.004705

                   Credit_History
Gender                   0.009170
Dependents              -0.040160
Self_Employed           -0.001550
ApplicantIncome         -0.018615
CoapplicantIncome        0.011134
LoanAmount              -0.000250
Loan_Amount_Term        -0.004705
Credit_History           1.000000
```

[164]:

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), cmap='coolwarm', annot=True, fmt='.1f', linewidths=.1)
plt.show()
```



This matrix shows the linear relationships between these variables, with values ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation).

[97]:
```
#x and y split

target_column = 'Loan_Status'
feature_columns = ['Gender', 'Married',␣
↪'Dependents','Self_Employed','ApplicantIncome', 'LoanAmount',␣
↪'Credit_History']
```

[99]:
```
X = df[feature_columns]
y = df[target_column]
```

[100]:
```
# Print the X (features) DataFrame
print(X.head())
```

```
   Gender  Married  Dependents  Self_Employed  ApplicantIncome  LoanAmount  \
0       1        0           0              0             5849       120.0
1       1        1           1              0             4583       128.0
2       1        1           0              1             3000        66.0
```

```
3        1         1            0               0            2583        120.0
4        1         0            0               0            6000        141.0

   Credit_History
0             1.0
1             1.0
2             1.0
3             1.0
4             1.0
```

[101]:
```
print(y.head())
```

```
0    1
1    0
2    1
3    1
4    1
Name: Loan_Status, dtype: int64
```

[102]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,stratify␣
↪=y,random_state =42)
```

[103]:
```
X_train.shape
```

[103]: (491, 7)

[104]:
```
X_test.shape
```

[104]: (123, 7)

[105]:
```
y_train.shape
```

[105]: (491,)

[106]:
```
y_test.shape
```

[106]: (123,)

7 Model 1: Decision Tree Classifier

[107]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score,f1_score
```

[108]:
```
tree_clf = DecisionTreeClassifier()
tree_clf.fit(X_train,y_train)
y_pred = tree_clf.predict(X_train)
```

```
print("Training Data Set Accuracy: ", accuracy_score(y_train,y_pred))
print("Training Data F1 Score ", f1_score(y_train,y_pred))
```

```
Training Data Set Accuracy:  1.0
Training Data F1 Score  1.0
```

8 Model 2: Random Forest Classifier

[109]:
```
from sklearn.ensemble import RandomForestClassifier
```

[111]:
```
rm= RandomForestClassifier(n_estimators=100,max_depth=3,min_samples_leaf = 10)
rm.fit(X_train,y_train)
y_pred = rm.predict(X_train)
print("Train F1 Score ", f1_score(y_train,y_pred))
print("Train Accuracy ", accuracy_score(y_train,y_pred))
```

```
Train F1 Score  0.8699080157687253
Train Accuracy  0.7983706720977597
```

[112]:
```
print("Validation Mean F1 Score:␣
↪",cross_val_score(rm,X_train,y_train,cv=5,scoring='f1_macro').mean())
print("Validation Mean Accuracy:␣
↪",cross_val_score(rm,X_train,y_train,cv=5,scoring='accuracy').mean())
```

```
Validation Mean F1 Score:  0.7105036634489533
Validation Mean Accuracy:  0.7983714698000413
```

9 Model 3: Logistic Regression

[114]:
```
from sklearn.linear_model import LogisticRegression
```

[115]:
```
LR_model = LogisticRegression(C=0.01).fit(X_train,y_train)
LR_model
```

[115]:
```
LogisticRegression(C=0.01)
```

[116]:
```
yhat = LR_model.predict(X_test)
yhat
```

[116]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

[117]:
```
model = LogisticRegression()
```

[124]:
```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

[139]:
```
numeric_features = ['Gender ', 'Married', 'Education', 'Self_Employed',␣
↪'Property_Area','Loan_Status']
categorical_features = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',␣
↪'Loan_Amount_Term', 'Credit_History']
```

[140]:
```
# Create transformers for imputation, encoding, and scaling
imputer = SimpleImputer(strategy='most_frequent')
encoder = OneHotEncoder(handle_unknown='ignore')
scaler = StandardScaler()
```

[141]:
```
# Define the ColumnTransformer
preprocessor = ColumnTransformer(transformers=[
    ('imputer', imputer, categorical_features),
    ('encoder', encoder, categorical_features),
    ('scaler', scaler, numeric_features)
])
```

[148]:
```
# Create the pipeline with preprocessing steps and a classifier
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('DecisionTreeClassifier', DecisionTreeClassifier())
])
```

[149]:
```
pipeline
```

[149]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('imputer',
SimpleImputer(strategy='most_frequent'),
                                                  ['ApplicantIncome',
                                                   'CoapplicantIncome',
                                                   'LoanAmount',
                                                   'Loan_Amount_Term',
                                                   'Credit_History']),
                                                 ('encoder',
OneHotEncoder(handle_unknown='ignore'),
                                                  ['ApplicantIncome',
                                                   'CoapplicantIncome',
                                                   'LoanAmount',
                                                   'Loan_Amount_Term',
                                                   'Credit_History']),
```

```
                                              ('scaler', StandardScaler(),
                                               ['Gender ', 'Married',
                                                'Education', 'Self_Employed',
                                                'Property_Area',
                                                'Loan_Status'])])),
                ('DecisionTreeClassifier', DecisionTreeClassifier())])
```

10 Observation

[ ]:
```
# the accuracy of the built model using different evaluation
```

DecisionTreeClassifier = accuracy =Training Data Set Accuracy: 1.0 Training Data F1 Score 1.0

RandomForestClassifier = accuracy = Train Accuracy 0.7983706720977597 Train F1 Score 0.8699080157687253