Loan Approval Predication

May 16, 2023

0.0.1 Loan Approval Prediction using Python

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# read the dataset
df=pd.read_csv('loan_prediction.csv')
df.head()
```

[2]:
```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             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         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

[3]:
```
df.shape
```

[3]: (614, 13)

[4]:
```python
# I'll drop the loan id column and move further:
df = df.drop('Loan_ID', axis=1)
```

[5]:
```python
df
```

[5]:
```
    Gender Married Dependents     Education Self_Employed  ApplicantIncome  \
0     Male      No          0      Graduate            No             5849
1     Male     Yes          1      Graduate            No             4583
2     Male     Yes          0      Graduate           Yes             3000
3     Male     Yes          0  Not Graduate            No             2583
4     Male      No          0      Graduate            No             6000
..     …       …          …           …             …               …
609 Female      No          0      Graduate            No             2900
610   Male     Yes         3+      Graduate            No             4106
611   Male     Yes          1      Graduate            No             8072
612   Male     Yes          2      Graduate            No             7583
613 Female      No          0      Graduate           Yes             4583

    CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                 0.0         NaN             360.0             1.0
1              1508.0       128.0             360.0             1.0
2                 0.0        66.0             360.0             1.0
3              2358.0       120.0             360.0             1.0
4                 0.0       141.0             360.0             1.0
..                …          …               …              …
609               0.0        71.0             360.0             1.0
610               0.0        40.0             180.0             1.0
611             240.0       253.0             360.0             1.0
612               0.0       187.0             360.0             1.0
613               0.0       133.0             360.0             0.0

   Property_Area Loan_Status
0          Urban           Y
1          Rural           N
2          Urban           Y
3          Urban           Y
4          Urban           Y
..           …         …
609        Rural           Y
610        Rural           Y
611        Urban           Y
612        Urban           Y
613    Semiurban           N

[614 rows x 12 columns]
```

[6]:
```python
##Let's have a look if the data has missing values or not:
df.isnull().sum()
```

[6]:
```
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
```

[7]:
```python
# The data has missing values in some of the categorical columns and some␣
↪numerical columns. Let's have a look at the descriptive statistics of the␣
↪dataset before filling in the missing values:
```

[8]:
```python
df.describe()
```

[8]:
```
       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
```

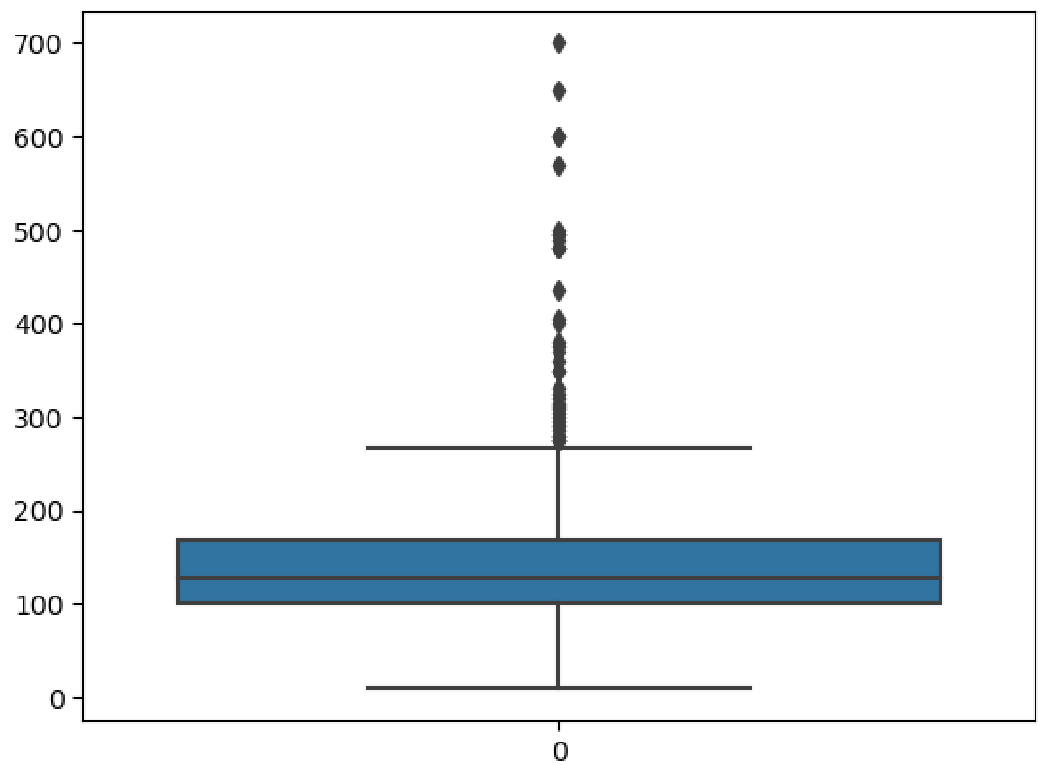
[9]:
```python
# In categorical columns, we can fill in missing values with the mode of each␣
↪column. The mode represents the value that appears most often in the column␣
↪and is an appropriate choice when dealing with categorical data:
```

```
[10]: # Fill missing values in categorical columns with mode

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)
```

1.We can fill in the missing values of the loan amount column with the median value. The median is an appropriate measure to fill in missing values when dealing with skewed distributions or when outliers are present in the data;

```
[11]: # Visualizing Outliers Using Box Plot
import seaborn as sns
sns.boxplot(df['LoanAmount'])
```

```
[11]: <AxesSubplot: >
```




```
[12]: # Fill missing values in LoanAmount with the median
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)
```

2.We can fill in the missing values of the loan amount term column with the mode value of the column. Since the term of the loan amount is a discrete value, the mode is an appropriate metric to use;

[13]:
```
# Fill missing values in Loan_Amount_Term with the mode
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
```

3.We can fill in the missing values of the credit history column with the mode value. Since credit history is a binary variable (0 or 1), the mode represents the most common value and is an appropriate choice for filling in missing values.

[14]:
```
# Fill missing values in Credit_History with the mode
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

[15]:
```
df
```

[15]:
```
     Gender Married Dependents     Education Self_Employed  ApplicantIncome  \
0      Male      No          0      Graduate            No             5849
1      Male     Yes          1      Graduate            No             4583
2      Male     Yes          0      Graduate           Yes             3000
3      Male     Yes          0  Not Graduate            No             2583
4      Male      No          0      Graduate            No             6000
..      …       …          …           …             …              …
609  Female      No          0      Graduate            No             2900
610    Male     Yes         3+      Graduate            No             4106
611    Male     Yes          1      Graduate            No             8072
612    Male     Yes          2      Graduate            No             7583
613  Female      No          0      Graduate           Yes             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                  0.0       128.0             360.0             1.0
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
..                 …          …               …             …
609                0.0        71.0             360.0             1.0
610                0.0        40.0             180.0             1.0
611              240.0       253.0             360.0             1.0
612                0.0       187.0             360.0             1.0
613                0.0       133.0             360.0             0.0

    Property_Area Loan_Status
0           Urban           Y
1           Rural           N
2           Urban           Y
3           Urban           Y
4           Urban           Y
..            …         …
609         Rural           Y
610         Rural           Y
```

```
611         Urban           Y
612         Urban           Y
613     Semiurban           N

[614 rows x 12 columns]
```

0.1 Exploratory Data Analysis

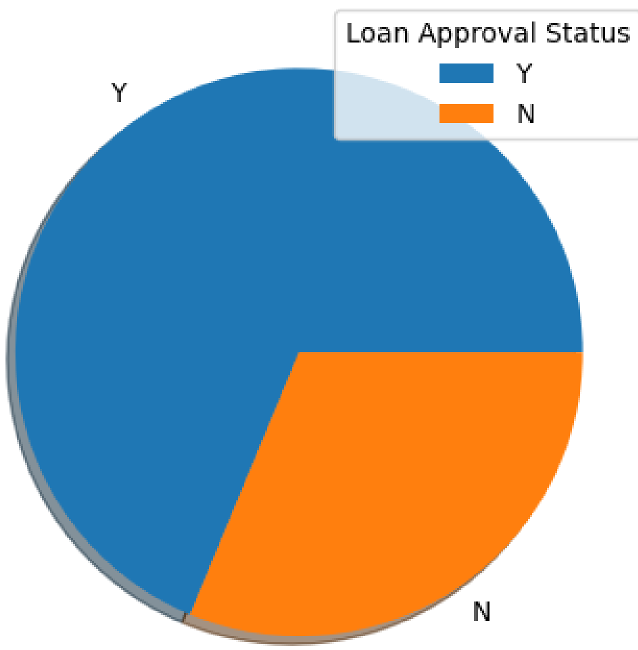
[53]:
```python
#Now let's have a look at the distribution of the loan status column:
```

[16]:
```python
loan_status_count = df['Loan_Status'].value_counts()
```

[52]:
```python
loan_status_count
```

[52]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

[17]:
```python
plt.pie(loan_status_count,shadow=True,labels=loan_status_count.index)
plt.legend(title='Loan Approval Status')
plt.show()
```



[54]:
```python
#Now let's have a look at the distribution of the gender column:
```

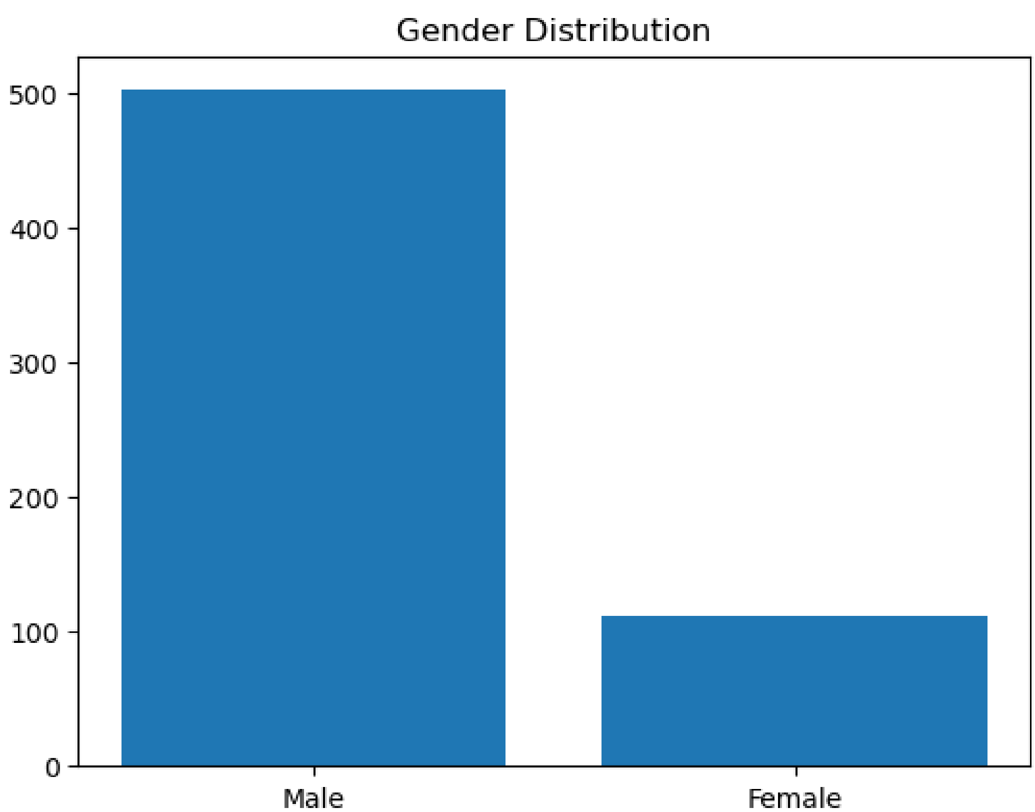
[19]:
```python
gender_count = df['Gender'].value_counts()
```

[20]:
```python
gender_count
```

[20]:
```
Male      502
Female    112
Name: Gender, dtype: int64
```

[21]:
```python
plt.bar(gender_count.index,gender_count.values, align='center')
plt.title('Gender Distribution')
```

[21]:
```
Text(0.5, 1.0, 'Gender Distribution')
```



[55]:
```python
#Now let's have a look at the distribution of the martial status column:
```

[22]:
```python
married_count = df['Married'].value_counts()
```

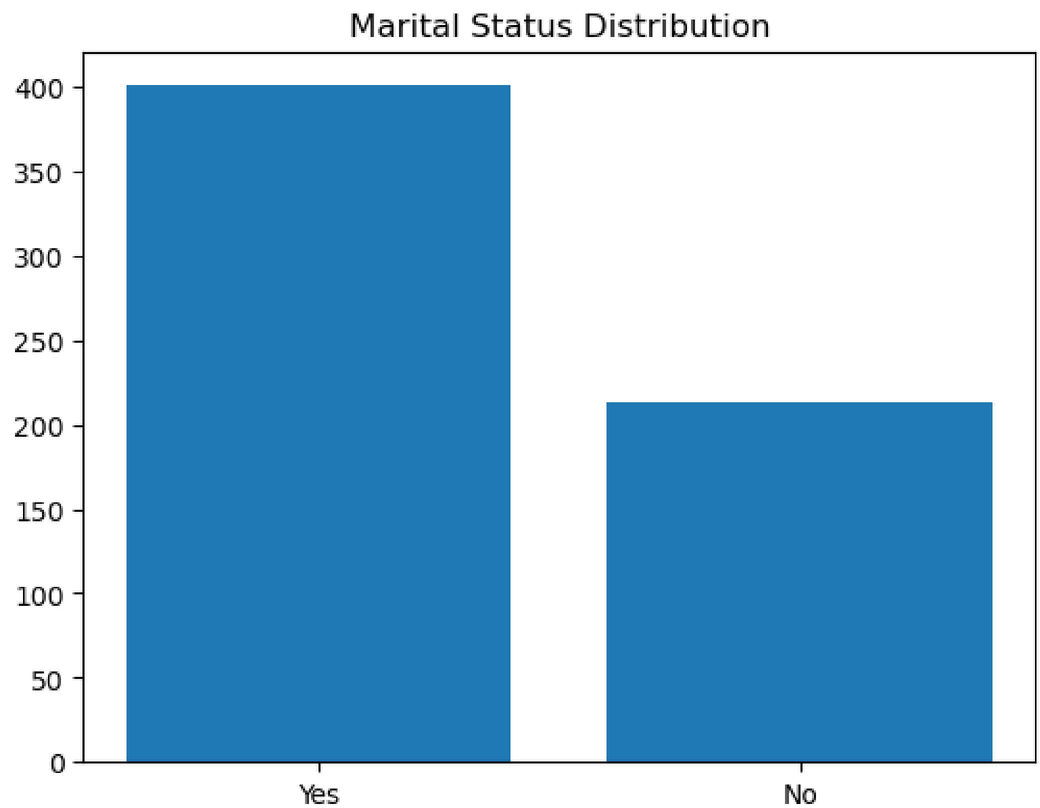
[23]:
```python
married_count
```

[23]:
```
Yes    401
No     213
```

```
Name: Married, dtype: int64
```

[24]:
```
plt.bar(married_count.index,married_count.values, align='center')
plt.title('Marital Status Distribution')
```

[24]:
```
Text(0.5, 1.0, 'Marital Status Distribution')
```



[56]:
```
#Now let’s have a look at the distribution of the education column:
```

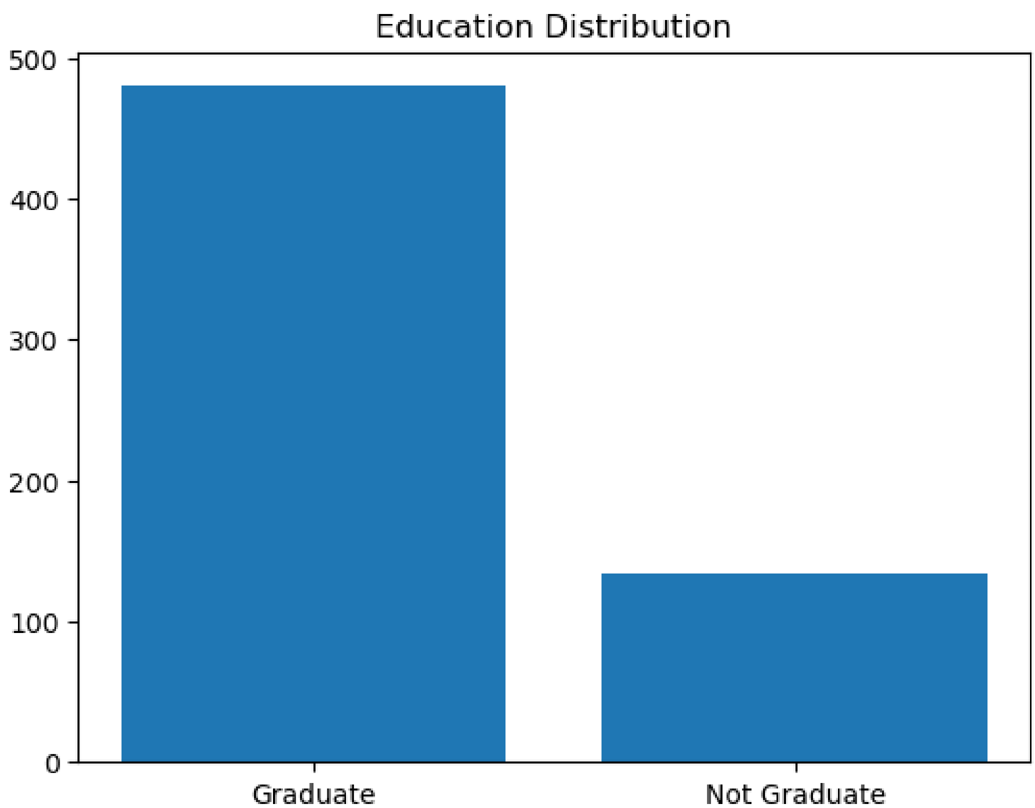
[25]:
```
education_count = df['Education'].value_counts()
```

[26]:
```
education_count
```

[26]:
```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

[27]:
```
plt.bar(education_count.index,education_count.values,align='center')
plt.title('Education Distribution')
```

[27]:
```
Text(0.5, 1.0, 'Education Distribution')
```



[57]:
```
#Now let's have a look at the distribution of the self-employment column:
```

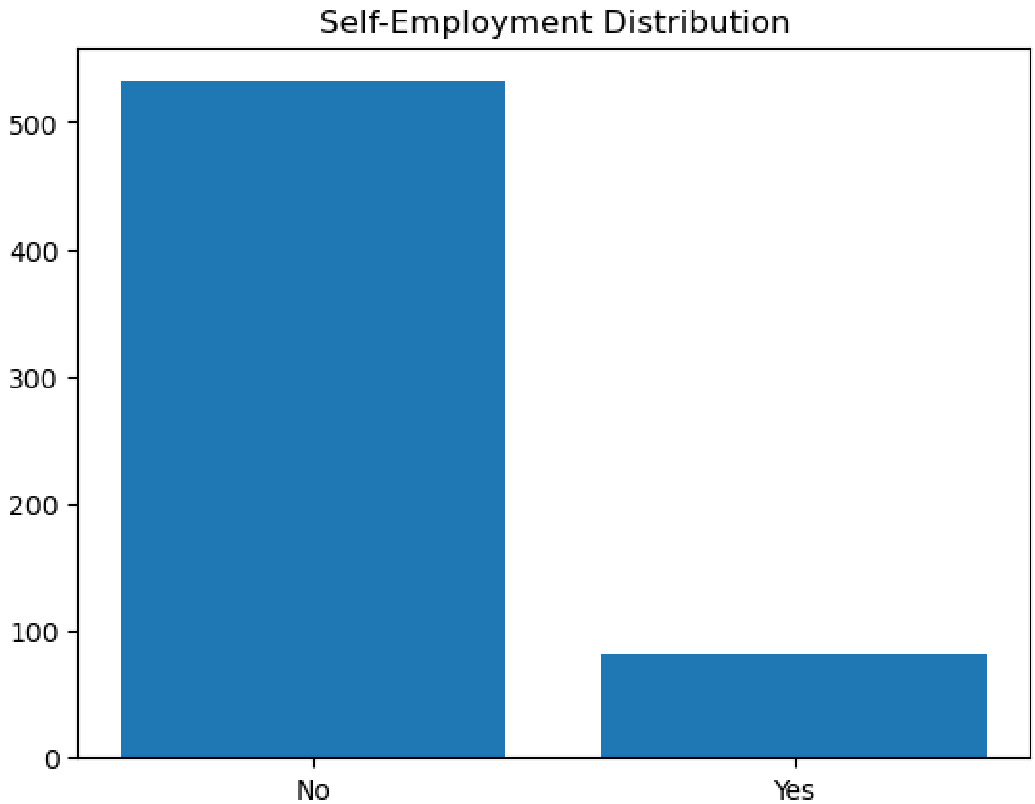
[28]:
```
self_employed_count = df['Self_Employed'].value_counts()
```

[29]:
```
self_employed_count
```

[29]:
```
No     532
Yes     82
Name: Self_Employed, dtype: int64
```

[30]:
```
plt.bar(self_employed_count.index,self_employed_count.values,align='center')
plt.title('Self-Employment Distribution')
```

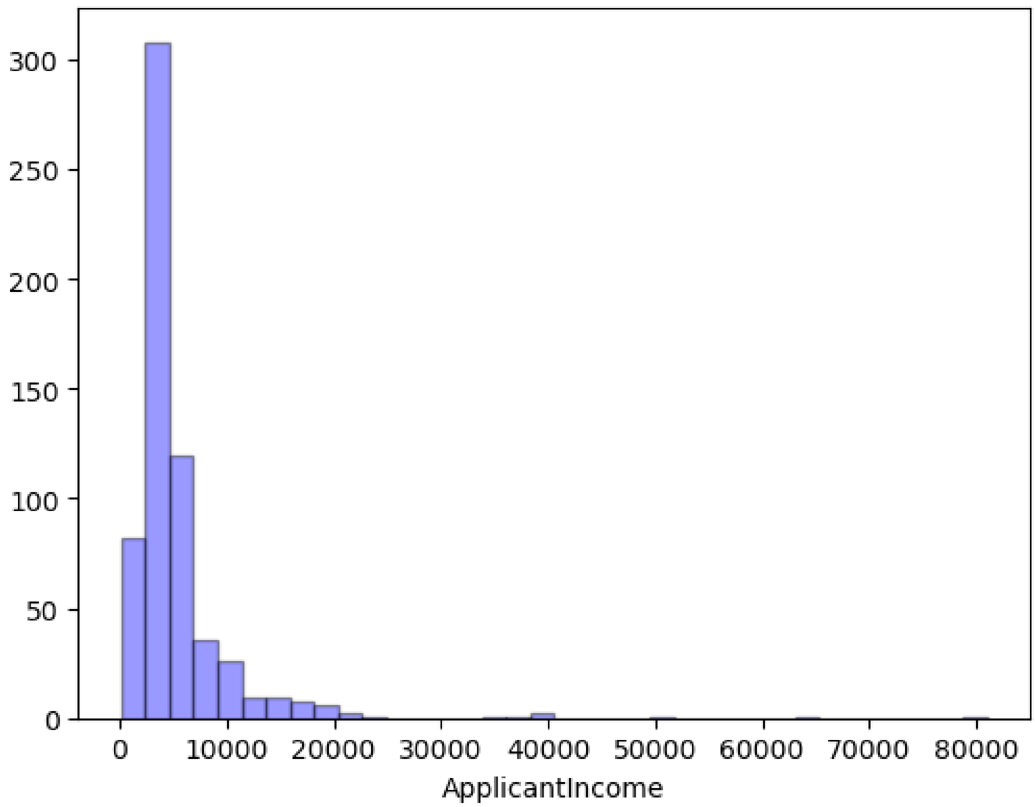
[30]:
```
Text(0.5, 1.0, 'Self-Employment Distribution')
```



[58]:
```
#Now let's have a look at the distribution of the Applicant Income column:
```

[31]:
```
sns.distplot(df['ApplicantIncome'], hist=True, kde=False,
             bins=int(180/5), color = 'blue',
             hist_kws={'edgecolor':'black'})
```

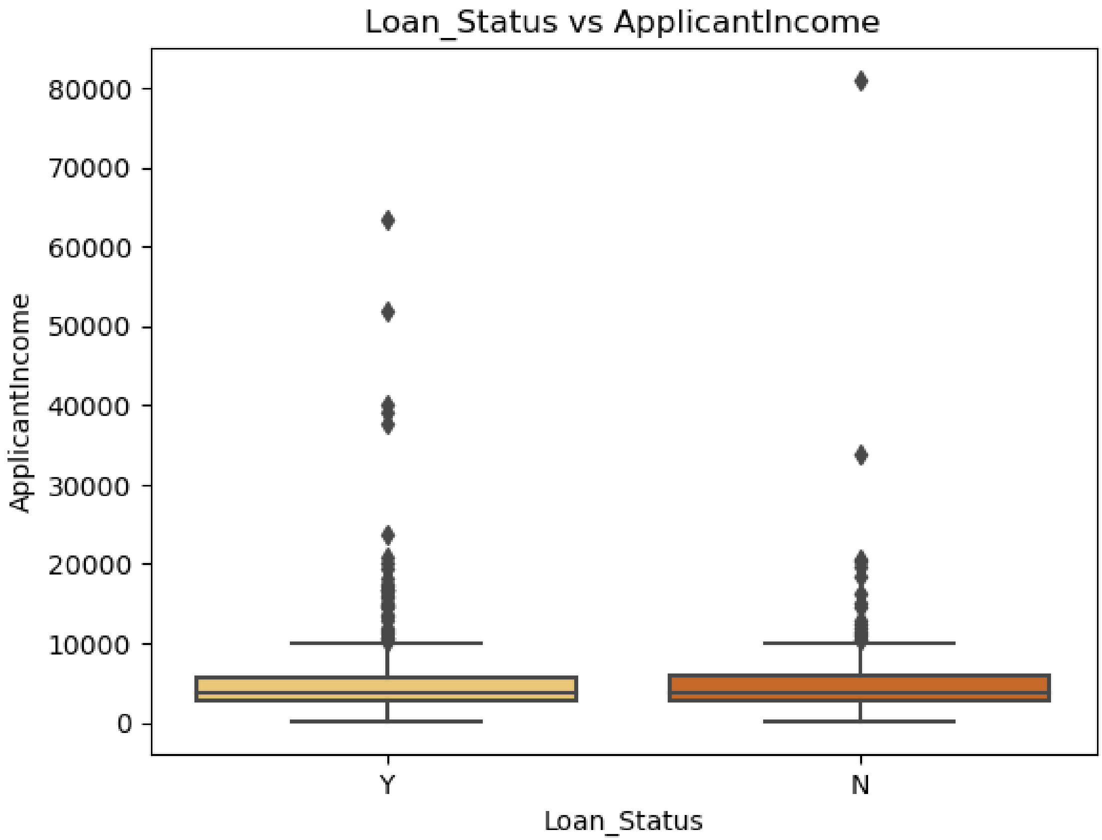
[31]: <AxesSubplot: xlabel='ApplicantIncome'>



[32]:
```
#Now let’s have a look at the relationship between the income of the loan␣
↪applicant and the loan status:
```

[33]:
```
sns.boxplot(x="Loan_Status", y="ApplicantIncome", data=df, palette="YlOrBr");
plt.title('Loan_Status vs ApplicantIncome')
```

[33]: Text(0.5, 1.0, 'Loan_Status vs ApplicantIncome')

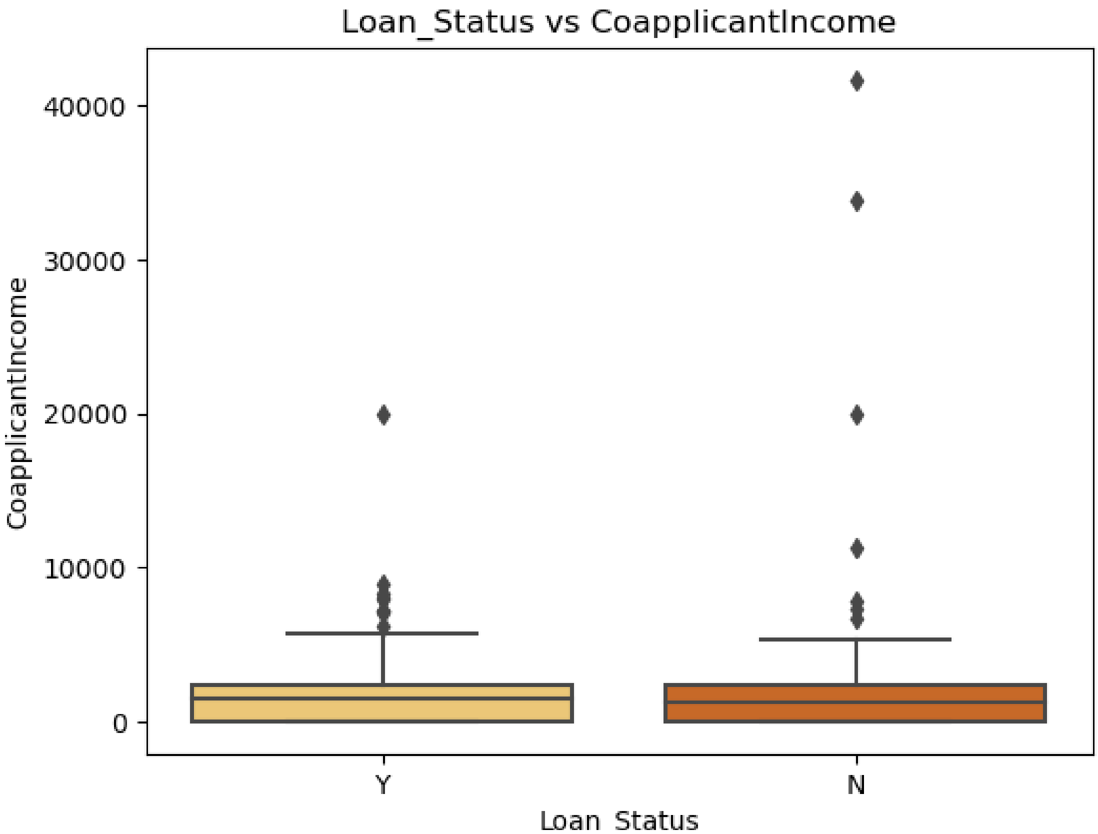


- The “ApplicantIncome” column contains outliers which need to be removed before moving further. Here’s how to remove the outliers:

[34]:
```
# Calculate the IQR
Q1 = df['ApplicantIncome'].quantile(0.25)
Q3 = df['ApplicantIncome'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df = df[(df['ApplicantIncome'] >= lower_bound) & (df['ApplicantIncome'] <=␣
 ↪upper_bound)]
```

[35]:
```
sns.boxplot(x="Loan_Status", y="CoapplicantIncome", data=df, palette="YlOrBr");
plt.title('Loan_Status vs CoapplicantIncome')
```

[35]:
```
Text(0.5, 1.0, 'Loan_Status vs CoapplicantIncome')
```



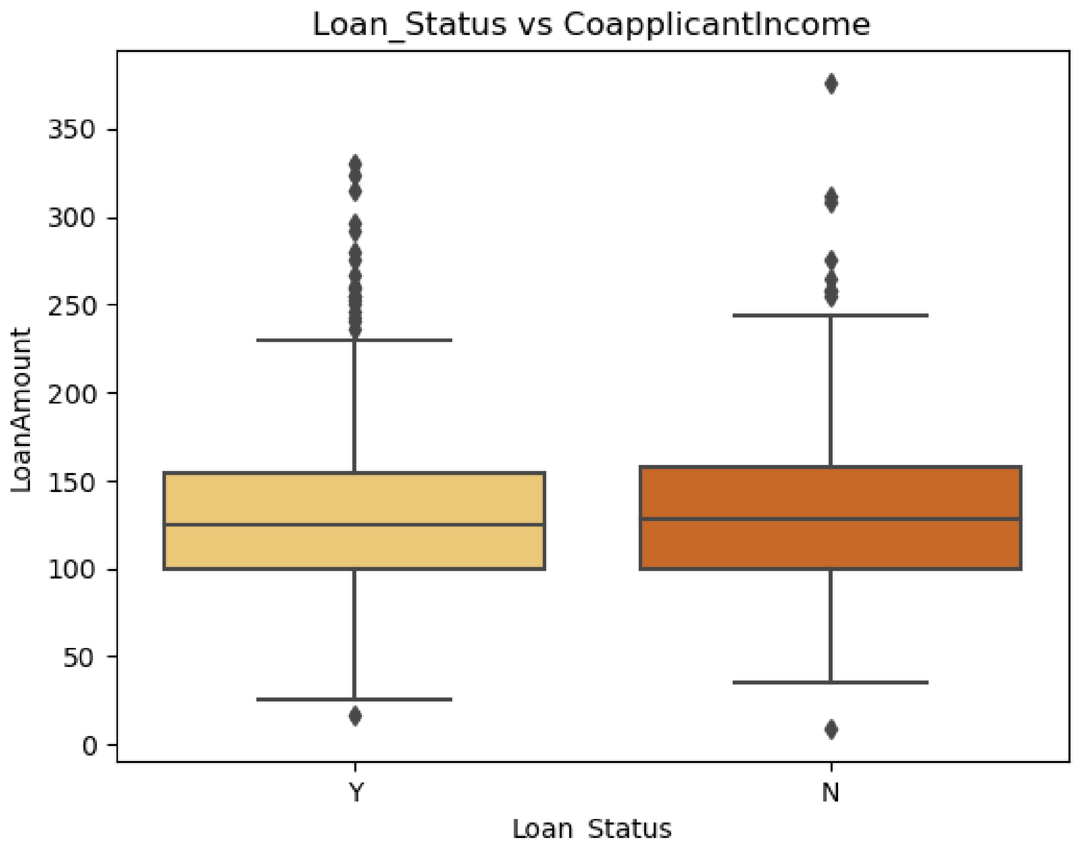
- The income of the loan co-applicant also contains outliers. Let's remove the outliers from this column as well:

[36]:
```
# Calculate the IQR
Q1 = df['CoapplicantIncome'].quantile(0.25)
Q3 = df['CoapplicantIncome'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df = df[(df['CoapplicantIncome'] >= lower_bound) & (df['CoapplicantIncome'] <=
 ↪upper_bound)]
```

[37]:
```
#Now let's have a look at the relationship between the loan amount and the loan
 ↪status:
```

[38]:
```
sns.boxplot(x='Loan_Status', y='LoanAmount', data=df, palette="YlOrBr");
plt.title('Loan_Status vs CoapplicantIncome')
```

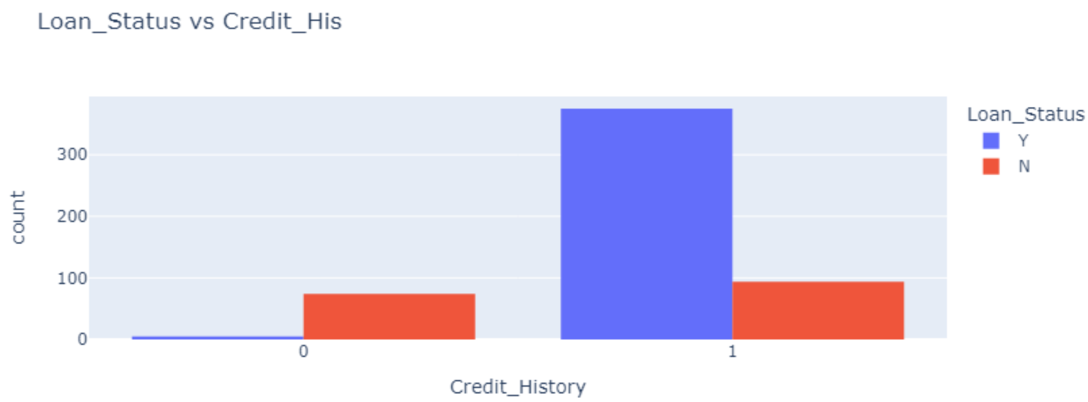
[38]: Text(0.5, 1.0, 'Loan_Status vs CoapplicantIncome')



[39]:
```
#Now let's have a look at the relationship between credit history and loan␣
↪status:
```

[40]:
```
import plotly.express as px
```

[43]:
```
fig_credit_history = px.histogram(df, x='Credit_History', color='Loan_Status',
                                  barmode='group',
                                  title='Loan_Status vs Credit_His')
fig_credit_history.show()
```




```python
[44]: fig_property_area = px.histogram(df, x='Property_Area', color='Loan_Status',
                                       barmode='group',
                                      title='Loan_Status vs Property_Area')
      fig_property_area.show()
```

0.2 Data Preparation and Training Loan Approval Prediction Model

In this step, we will:

1) convert categorical columns into numerical ones;
2) split the data into training and test sets;
3) scale the numerical features;
4) train the loan approval prediction model.

[45]:
```
# Convert categorical columns to numerical using one-hot encoding
cat_cols = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',␣
↪'Property_Area']
df = pd.get_dummies(df, columns=cat_cols)
```

[46]:
```
# Split the dataset into features (X) and target (y)
X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']
```

[47]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[48]:
```
# Scale the numerical columns using StandardScaler
scaler = StandardScaler()
numerical_cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',␣
↪'Loan_Amount_Term', 'Credit_History']
X_train[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])
```

[49]:
```
from sklearn.svm import SVC
model = SVC(random_state=42)
model.fit(X_train, y_train)
```

[49]:
```
SVC(random_state=42)
```

[59]:
```
#Now let's make predictions on the test set:
```

[50]:
```
y_pred = model.predict(X_test)
print(y_pred)
```

```
['Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y']
```

[51]:
```
# Convert X_test to a DataFrame
X_test_df = pd.DataFrame(X_test, columns=X_test.columns)

# Add the predicted values to X_test_df
X_test_df['Loan_Status_Predicted'] = y_pred
print(X_test_df.head())
```

```
     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
```

```
277       -0.544528           -0.037922   -0.983772           0.305159
84        -0.067325           -0.931554   -1.571353          -1.430680
275       -0.734870            0.334654   -0.298262           0.305159
392       -0.824919            0.522317   -0.200332           0.305159
537       -0.267373           -0.931554   -0.454950           0.305159

     Credit_History  Gender_Female  Gender_Male  Married_No  Married_Yes  \
277        0.402248              0            1           0            1
84         0.402248              0            1           0            1
275        0.402248              0            1           0            1
392        0.402248              0            1           0            1
537        0.402248              0            1           1            0

     Dependents_0  …  Dependents_2  Dependents_3+  Education_Graduate  \
277             1  …             0              0                   1
84              0  …             0              0                   1
275             0  …             0              0                   1
392             1  …             0              0                   1
537             0  …             1              0                   1

     Education_Not Graduate  Self_Employed_No  Self_Employed_Yes  \
277                       0                 1                  0
84                        0                 1                  0
275                       0                 1                  0
392                       0                 1                  0
537                       0                 1                  0

     Property_Area_Rural  Property_Area_Semiurban  Property_Area_Urban  \
277                    0                        0                    1
84                     0                        0                    1
275                    0                        1                    0
392                    0                        0                    1
537                    0                        1                    0

     Loan_Status_Predicted
277                      Y
84                       Y
275                      Y
392                      Y
537                      Y

[5 rows x 21 columns]
```

- So this is how you can train a Machine Learning model to predict loan approval using Python.