Loan_Approval_Prediction_Model_Using_Neural_Networks (3)

March 23, 2025

[132]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
from keras import Sequential
from keras.layers import Dense
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

[133]:
```
from google.colab import auth
auth.authenticate_user()

import gspread
from google.auth import default
creds, _ = default()

gc = gspread.authorize(creds)

worksheet = gc.open('bankloan').sheet1

# get_all_values gives a list of rows.
rows = worksheet.get_all_values()
# print(rows)

# Convert to a DataFrame and render.
import pandas as pd
df = pd.DataFrame.from_records(rows)
```

Data Cleaning

[134]:
```
# setting first row as column headers
# droping old index and reseting new index
# displaying the first 5 rows
df.columns = df.iloc[0]
```

```
df = df.iloc[1:].reset_index(drop=True)
df.head()
```

[134]:
```
0   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

0 ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term  \
0            5849                 0        300              360
1            4583              1508        128              360
2            3000                 0         66              360
3            2583              2358        120              360
4            6000                 0        141              360

0 Credit_History Property_Area Loan_Status
0              1         Urban           Y
1              1         Rural           N
2              1         Urban           Y
3              1         Urban           Y
4              1         Urban           Y
```

[135]:
```
# checking duplicates
# no duplicates found
df[df.duplicated(keep=False)]
```

[135]:
```
Empty DataFrame
Columns: [Loan_ID, Gender, Married, Dependents, Education, Self_Employed,
ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term,
Credit_History, Property_Area, Loan_Status]
Index: []
```

[136]:
```
# checking for missing values
# no dupliactes found
df.isnull().sum().sum()
```

[136]:
```
np.int64(0)
```

[137]:
```
df.isna().sum()
```

[137]:
```
0
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
```

[138]:
```python
# checking all columns that have an empty string
for col in df.columns:
  print(col, df[df[col] == ''].shape[0])
```

```
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
```

[139]:
```python
df.dtypes
```

[139]:
```
0
Loan_ID              object
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome      object
CoapplicantIncome    object
LoanAmount           object
Loan_Amount_Term     object
Credit_History       object
Property_Area        object
Loan_Status          object
dtype: object
```

[140]:
```
# changing to the correct data type
df['Dependents'] = df['Dependents'].astype(int)
df['ApplicantIncome'] = df['ApplicantIncome'].astype(float)
df['CoapplicantIncome'] = df['CoapplicantIncome'].astype(float)
df['LoanAmount'] = df['LoanAmount'].astype(float)
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].astype(int)
```

[141]:
```
# displaying the shape of the data
df.shape
```

[141]:
```
(614, 13)
```

[142]:
```
# descriptive statistics for the DataFrame
df.describe()
```

[142]:
```
0      Dependents  ApplicantIncome  CoapplicantIncome  LoanAmount  \
count  614.000000       614.000000         614.000000  614.000000
mean     0.786645      5403.459283        1621.245798  152.257329
std      1.017827      6109.041673        2926.248369   89.356494
min      0.000000       150.000000           0.000000    9.000000
25%      0.000000      2877.500000           0.000000  100.250000
50%      0.000000      3812.500000        1188.500000  129.000000
75%      2.000000      5795.000000        2297.250000  175.000000
max      3.000000     81000.000000       41667.000000  700.000000

0      Loan_Amount_Term
count        614.000000
mean         335.228013
std           78.182120
min           12.000000
25%          360.000000
50%          360.000000
75%          360.000000
max          480.000000
```

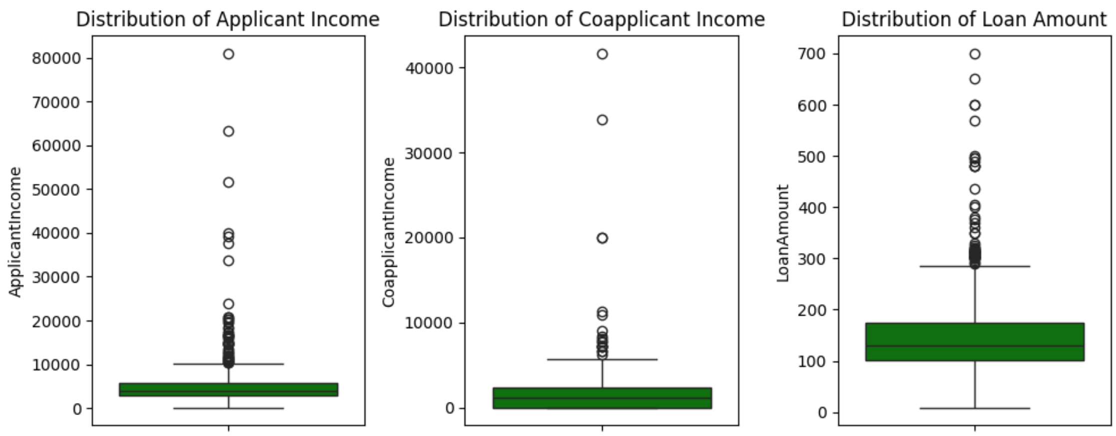
0.1 Exploratory Data Analysis

[143]:
```
# box plots showing ditribution of Applicantincome, CoapplicantIncome and Loan␣
↪Amount
fig, ax = plt.subplots(1,3, figsize=(10,4))
sns.boxplot(y = df["ApplicantIncome"], ax= ax[0], color='green')
ax[0].set_title("Distribution of Applicant Income")

sns.boxplot(df["CoapplicantIncome"], ax= ax[1], color='green')
ax[1].set_title("Distribution of Coapplicant Income")

sns.boxplot(df["LoanAmount"], ax= ax[2], color='green')
```

```
ax[2].set_title("Distribution of Loan Amount")
plt.tight_layout()
plt.show()
```



0.1.1 Insights

1. Applicant income is higher and more widely distributed than coapplicant income. This means that the primary financial responsibility for the loan likely rests with the applicant.
2. A significant portion of coapplicants have limited or no income. This could imply the coapplicant's role being more about providing additional security or meeting eligibility criteria rather than contributing financially.
3. Loan amounts are concentrated in a lower range, with fewer instances of very large loans. This may be influenced by the income distribution of the applicants and the lending criteria. This is actually a good sign because it may be in the means of the borrower to pay thus mitigating risks.
4. The presence of outliers in all three distributions highlights the diversity in the applicant pool and the range of loan amounts requested and approved.Instead of removal, they will be analyzed separately.

[144]:
```
df_yes = df[df["Loan_Status"] == 'Y']
df_yes.head(2)
```

[144]:
```
0   Loan_ID Gender Married  Dependents Education Self_Employed  \
0  LP001002   Male      No           0  Graduate            No
2  LP001005   Male     Yes           0  Graduate           Yes

0  ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0           5849.0                0.0       300.0               360
2           3000.0                0.0        66.0               360

0 Credit_History Property_Area Loan_Status
0              1         Urban           Y
```

```
2              1         Urban            Y
```

[145]:

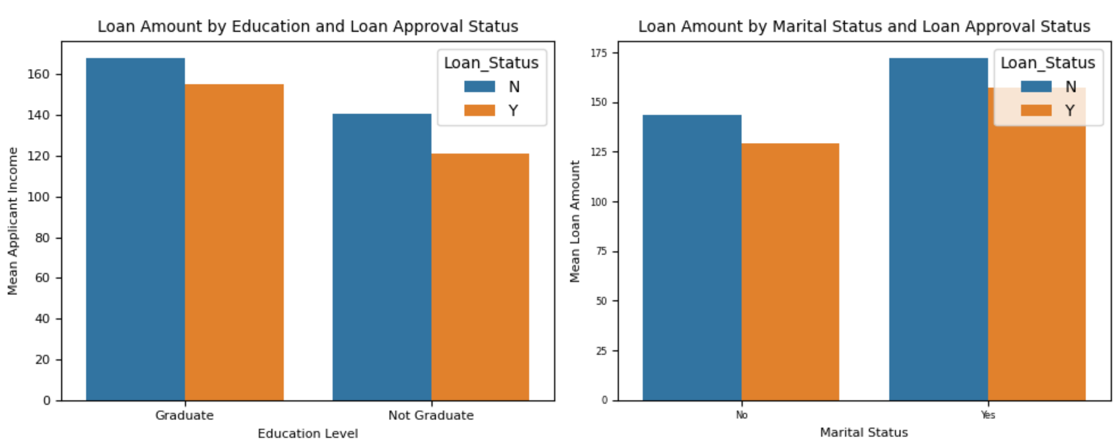
```
# Group by Education, Loan Status, and Marital Status, and calculate the mean␣
 ↪of Applicant Income and Loan Amount
df_educ = df.groupby(['Education', 'Loan_Status'])[['LoanAmount']].mean().
 ↪reset_index()
df_marital = df.groupby(['Married', 'Loan_Status'])[['LoanAmount']].mean().
 ↪reset_index()

# Create a subplot with 1 row and 2 columns
fig, ax = plt.subplots(1, 2, figsize=(10, 4))

# Plotting Applicant Income by Education and Loan Status
sns.barplot(data=df_educ, x='Education', y='LoanAmount', hue='Loan_Status',␣
 ↪ax=ax[0])
ax[0].set_title('Loan Amount by Education and Loan Approval Status',␣
 ↪fontsize=10)
ax[0].set_xlabel('Education Level', fontsize=8)
ax[0].set_ylabel('Mean Applicant Income', fontsize=8)
ax[0].tick_params(axis='both', labelsize=8)

# Plotting Loan Amount by Marital Status and Loan Status
sns.barplot(data=df_marital, x='Married', y='LoanAmount', hue='Loan_Status',␣
 ↪ax=ax[1])
ax[1].set_title('Loan Amount by Marital Status and Loan Approval Status',␣
 ↪fontsize=10)
ax[1].set_xlabel('Marital Status', fontsize=8)
ax[1].set_ylabel('Mean Loan Amount', fontsize=8)
ax[1].tick_params(axis='both', labelsize=6)

# Adjust layout for better spacing
plt.tight_layout()
plt.show()
```



0.1.2 Insights

Married

1. Disapproved loans: Married individuals whose loan was not approved have a higher mean loan amount compared to unmarried individuals whose loan was not approved.
2. Approved loans: Married individuals whose loan was approved have a higher mean loan amount compared to unmarried individuals whose loan was approved.
3. Overall: Married individuals tend to have a higher mean loan amount compared to unmarried individuals, regardless of the loan approval status. For both marital statuses, the mean loan amount for disapproved loans is higher than for approved loans. The married are more likely to have loan approva

Education Status

1. Education level is positively associated with the mean loan amount. Graduates, on average, apply for and are more likely to be considered for larger loans than non-graduates.

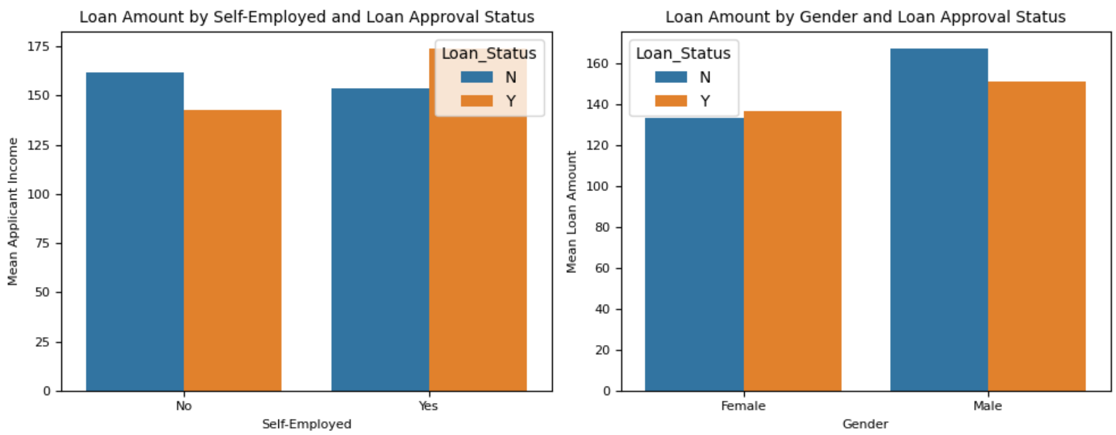
[146]:
```
# Group by Self-Employed, Gender, and Loan Status, and calculate the mean of␣
↪Applicant Income and Loan Amount
df_selfemployed = df.groupby(['Self_Employed', 'Loan_Status'])[['LoanAmount']].
↪mean().reset_index()
df_gender = df.groupby(['Gender', 'Loan_Status'])[['LoanAmount']].mean().
↪reset_index()

# Create a subplot with 1 row and 2 columns for gender and self-employed status
fig, ax = plt.subplots(1, 2, figsize=(10, 4))

# Plotting Applicant Income by Self-Employed and Loan Status
sns.barplot(data=df_selfemployed, x='Self_Employed', y='LoanAmount',␣
↪hue='Loan_Status', ax=ax[0])
ax[0].set_title('Loan Amount by Self-Employed and Loan Approval Status',␣
↪fontsize=10)
ax[0].set_xlabel('Self-Employed', fontsize=8)
ax[0].set_ylabel('Mean Applicant Income', fontsize=8)
ax[0].tick_params(axis='both', labelsize=8)

# Plotting Loan Amount by Gender and Loan Status
sns.barplot(data=df_gender, x='Gender', y='LoanAmount', hue='Loan_Status',␣
↪ax=ax[1])
ax[1].set_title('Loan Amount by Gender and Loan Approval Status', fontsize=10)
ax[1].set_xlabel('Gender', fontsize=8)
ax[1].set_ylabel('Mean Loan Amount', fontsize=8)
ax[1].tick_params(axis='both', labelsize=8)
```

```
plt.tight_layout()
plt.show()
```



0.1.3 Insights

1. Self-employment doesn't show a great impact on the mean loan amount based on approval status. For both self-employed and non-self-employed individuals, the mean loan amount for disapproved loans is similar or slightly higher than for approved loans.
2. For males, a higher mean loan amount is associated with loan disapproval. This could suggest that males might be applying for larger loans that are more frequently rejected, or other factors related to males applying for larger loans lead to higher rejection rates for those amounts.
3. For females, the trend is reversed; a higher mean loan amount is associated with loan approval. This might indicate that when females apply for larger loans, they are more likely to be approved.

0.2 Feature Engineering

[147]:
```
# Droping column Loan ID because its just an indentifier
df.drop('Loan_ID', axis=1, inplace=True)
df.head(2)
```

[147]:
```
0 Gender Married  Dependents Education Self_Employed  ApplicantIncome  \
0   Male      No           0  Graduate            No           5849.0
1   Male     Yes           1  Graduate            No           4583.0

0  CoapplicantIncome  LoanAmount  Loan_Amount_Term Credit_History  \
0                0.0       300.0               360              1
1             1508.0       128.0               360              1

0 Property_Area Loan_Status
0         Urban           Y
```

```
1         Rural            N
```

[148]:
```
# Identifying independent and dependent varibles
pre_x = df.drop('Loan_Status', axis=1)
pre_y = df['Loan_Status']
```

[149]:
```
# Checking for unique values in y
pre_y.unique()
```

[149]:
```
array(['Y', 'N'], dtype=object)
```

[150]:
```
# Convert Loan_Status to binary (1 for 'Y', 0 for 'N')
dum_y = pre_y.map(dict(Y=1,N=0))
```

[151]:
```
# Convert categorical features into dummy variables,
# dropping the first category to avoid multicollinearity
x_dum = pd.get_dummies(pre_x, drop_first=True)
x_dum = x_dum.astype(int)
x_dum.head(2)
```

[151]:
```
   Dependents  ApplicantIncome  CoapplicantIncome  LoanAmount  \
0           0             5849                  0         300
1           1             4583               1508         128

   Loan_Amount_Term  Gender_Male  Married_Yes  Education_Not Graduate  \
0               360            1            0                       0
1               360            1            1                       0

   Self_Employed_Yes  Credit_History_1  Property_Area_Semiurban  \
0                  0                 1                        0
1                  0                 1                        0

   Property_Area_Urban
0                    1
1                    0
```

[152]:
```
# Value counts for dum_y
# Imbalanced distribution so smote will be applied to balance classes
dum_y.value_counts()
```

[152]:
```
Loan_Status
1    422
0    192
Name: count, dtype: int64
```

[153]:
```
smote = SMOTE()
x_smote, y = smote.fit_resample(x_dum, dum_y)
```

```
min_max = MinMaxScaler()
x = min_max.fit_transform(x_smote)
```

[154]:
```
# Counting y values
y.value_counts()
```

[154]:
```
Loan_Status
1    422
0    422
Name: count, dtype: int64
```

[155]:
```
# Spliting the data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,␣
 ↪random_state = 42, shuffle = True)
```

0.3 Training the Model

[156]:
```
# Initialize a Sequential model
classifier = Sequential()
classifier.add(Dense(200, activation='relu', kernel_initializer =␣
 ↪'random_normal', input_dim = x_test.shape[1])) # Input layer
classifier.add(Dense(400, activation='relu', kernel_initializer =␣
 ↪'random_normal')) # Hidden layer
classifier.add(Dense(4, activation='relu', kernel_initializer =␣
 ↪'random_normal'))
classifier.add(Dense(1, activation='sigmoid', kernel_initializer =␣
 ↪'random_normal')) # Out put layer
classifier.compile(optimizer='adam', loss = 'binary_crossentropy', metrics =␣
 ↪['accuracy']) # Compiling the model
classifier.fit(x_train, y_train, batch_size = 20, epochs = 50, verbose = 0) #␣
 ↪Training
eval_model = classifier.evaluate(x_train, y_train) # Evaluating the trained␣
 ↪model
eval_model
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
22/22                 0s 4ms/step -
accuracy: 0.8391 - loss: 0.3440
```

[156]:
```
[0.3388855159282684З, 0.8399999737739563]
```

[157]:
```
y_predicted = classifier.predict(x_test)
y_predicted = (y_predicted > 0.5).astype(float)
```

```
6/6             0s 13ms/step
```

[158]:
```
# Convert probabilities into loan status
# Return both credit score (probability) and loan status
loan_status = np.where(y_predicted > 0.5, 'Y', 'N')
predicted_results = list(zip(y_predicted.flatten(), loan_status))
for credit_score, status in predicted_results:
   print(f"Predicted Credit Score: {credit_score:.2f}, Loan Status: {status}")
```

```
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
```

```
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
```

```
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
```

```
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
Predicted Credit Score: 1.00, Loan Status: ['Y']
Predicted Credit Score: 0.00, Loan Status: ['N']
```

[159]:
```
# Accuracy Score
ac = round(accuracy_score(y_test, y_predicted),2)
ac
```

[159]: 0.81

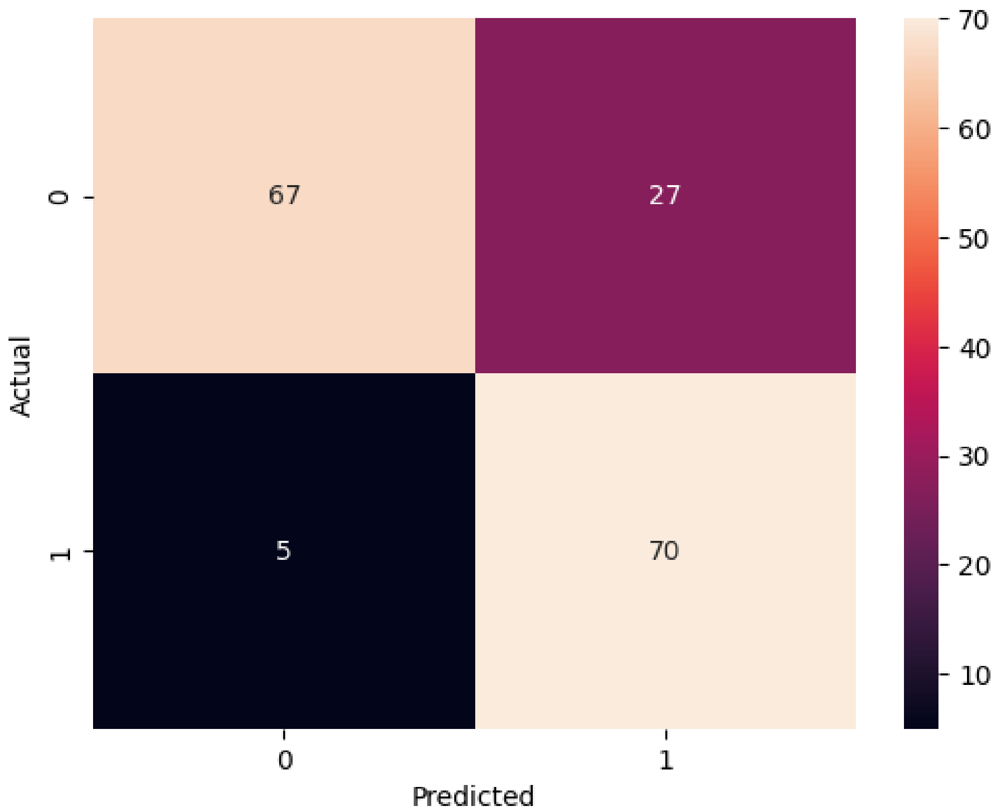
[160]:
```
# confusion matrix
cm = confusion_matrix(y_test, y_predicted)
cm
```

[160]:
```
array([[67, 27],
       [ 5, 70]])
```

1. The major diagonal values indicate how well the model is performing in terms of correct classifications. A high sum of these values means the model is making accurate predictions.
2. The model correctly predicted 73 cases as "No" (0).
3. The model incorrectly predicted 21 "No" (0) cases as a "Yes" (1)
4. The model incorrectly predicted 11 cases as "No" (0) when they were actually "Yes" (1).

5. The model correctly predicted 64 cases as "Yes" (1).

[161]:
```
# ploting a heat map for confusin matrix
sns.heatmap(cm, annot=True, fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



[162]:
```
# Pickling the model
import joblib
joblib.dump(classifier, 'loan_model.pkl')
```

[162]: ['loan_model.pkl']

[162]:
```

```