Loan Status Prediction

May 17, 2024

Author: Saviour Dan (SavyStat)

Aim: To build a logistic regression model that can effectively predict the loan status of customers The metrics for validation of the model used in this analysis include: F1_score, Precision, Accuracy, AUC, and ROC The odd ratio for each feature and their coefficients are also presented in this analysis

[1]:
```python
#import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```python
df = pd.read_csv("loanfile.csv")
```

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

[3]:
```
    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
```

[4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

[5]:
```
df.shape
```

[5]:
```
(614, 13)
```

[6]:
```
df.describe()
```

[6]:
```
       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
```

[7]:
```
df.isnull().sum()
```

[7]:
```
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
```

#Data Cleaning Process: Fill missing values for LoanAmount and credit_History with mean

[8]:
```python
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
```

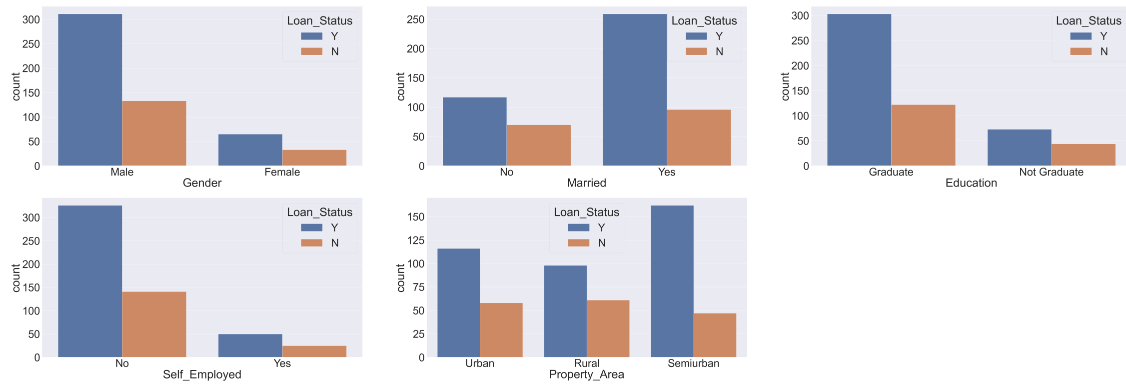
[9]:
```python
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mean())
```

[10]:
```python
df.dropna(inplace=True)
```

[11]:
```python
df.shape
```

[11]:
```
(542, 13)
```

[12]:
```python
plt.figure(figsize=(100, 50))
sns.set(font_scale=5)

plt.subplot(331)
sns.countplot(x='Gender', hue='Loan_Status', data=df)

plt.subplot(332)
sns.countplot(x='Married', hue='Loan_Status', data=df)

plt.subplot(333)
sns.countplot(x='Education', hue='Loan_Status', data=df)

plt.subplot(334)
sns.countplot(x='Self_Employed', hue='Loan_Status', data=df)

plt.subplot(335)
sns.countplot(x='Property_Area', hue='Loan_Status', data=df)

plt.show()
```



[13]:
```
df['Loan_Status'].replace('Y',1, inplace = True)
df['Loan_Status'].replace('N',0, inplace = True)
```

[14]:
```
df['Loan_Status'].value_counts()
```

[14]:
```
Loan_Status
1    376
0    166
Name: count, dtype: int64
```

[15]:
```
df.Gender= df.Gender.map({'Male':1, 'Female':0})
df['Gender'].value_counts()
```

[15]:
```
Gender
1    444
0     98
Name: count, dtype: int64
```

[16]:
```
df.Married= df.Married.map({'Yes':1, 'No':0})
df['Married'].value_counts()
```

[16]:
```
Married
1    355
0    187
Name: count, dtype: int64
```

[17]:
```
df.Dependents= df.Dependents.map({'0':0, '1':1, '2': 2, '3+':3})
df['Dependents'].value_counts()
```

[17]:
```
Dependents
0    309
1     94
2     94
3     45
```

```
Name: count, dtype: int64
```

[18]:
```
df.Education = df.Education.map({'Graduate':1, 'Not Graduate':0})
df['Education'].value_counts()
```

[18]:
```
Education
1    425
0    117
Name: count, dtype: int64
```

[19]:
```
df.Property_Area= df.Property_Area.map({'Rural':0, 'Semiurban':1, 'Urban': 2})
df['Property_Area'].value_counts()
```

[19]:
```
Property_Area
1    209
2    174
0    159
Name: count, dtype: int64
```

[20]:
```
df.Self_Employed= df.Self_Employed.map({'Yes':1, 'No':0})
df['Self_Employed'].value_counts()
```

[20]:
```
Self_Employed
0    467
1     75
Name: count, dtype: int64
```

[21]:
```
df.head()
```

[21]:
```
    Loan_ID  Gender  Married  Dependents  Education  Self_Employed  \
0  LP001002       1        0           0          1              0
1  LP001003       1        1           1          1              0
2  LP001005       1        1           0          1              1
3  LP001006       1        1           0          0              0
4  LP001008       1        0           0          1              0

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0  146.412162             360.0
1             4583             1508.0  128.000000             360.0
2             3000                0.0   66.000000             360.0
3             2583             2358.0  120.000000             360.0
4             6000                0.0  141.000000             360.0

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

4 1.0 2 1

[22]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

[23]:
```
x = df.iloc[1:542, 1:12].values
y = df.iloc[1:542, 12].values
```

[24]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size =0.3,␣
↪random_state =0)
```

[25]:
```
model = LogisticRegression()
model.fit(x_train, y_train)

lr_prediction = model.predict(x_test)
print("Logistic Regression Accuracy: ", metrics.accuracy_score(y_test,␣
↪lr_prediction))
```

```
Logistic Regression Accuracy:  0.7852760736196319
```

[26]:
```
precision = metrics.precision_score(y_test, lr_prediction)
print("Logistic Regression Precision: ", precision)
```

```
Logistic Regression Precision:  0.7591240875912408
```

[27]:
```
f1 = metrics.f1_score(y_test, lr_prediction)
print("Logistic Regression f1_score: ", f1)
```

```
Logistic Regression f1_score:  0.8559670781893004
```

[28]:
```
lr_pred_probabilities = model.predict_proba(x_test)[:, 1]
```

[29]:
```
print(lr_pred_probabilities)
```

```
[0.78543932 0.22759719 0.27160626 0.78246052 0.8719914  0.84118535
 0.72968711 0.8106914  0.85816943 0.81057493 0.85431095 0.61970044
 0.80033625 0.73019305 0.27807681 0.73097375 0.73483661 0.88142099
 0.8482974  0.87083999 0.77584077 0.86288701 0.66726422 0.81755208
 0.81391518 0.69997132 0.80997632 0.81686204 0.83264866 0.17155246
 0.87897616 0.52090033 0.30114606 0.09107134 0.51179919 0.74562209
 0.25733697 0.82075875 0.84907224 0.70393257 0.77021702 0.3000842
 0.19107525 0.86246452 0.81976383 0.69120089 0.76486472 0.59274573
 0.87140219 0.44012109 0.73546742 0.86962499 0.22621137 0.83615654
 0.83349035 0.87113035 0.11685049 0.86038571 0.56924912 0.88418652
 0.86283694 0.67325799 0.86126167 0.2727278  0.73387833 0.63594487
 0.7717784  0.85861377 0.71375275 0.18439935 0.81006039 0.81987571
 0.68805666 0.86650337 0.85887747 0.82759388 0.67926772 0.84008585
 0.81979684 0.84604287 0.80643842 0.73671585 0.24905102 0.82947245
 0.77169345 0.83735722 0.74422214 0.88950841 0.37037516 0.9340723
```

```
 0.38357656 0.79363135 0.84870264 0.68410958 0.86577769 0.86469331
 0.81589544 0.88340907 0.73549345 0.78224669 0.69159609 0.8535461
 0.25779172 0.5489995  0.73695001 0.93320391 0.8729035  0.73324307
 0.71935128 0.73979979 0.63191772 0.74890449 0.8594812  0.65377226
 0.78836098 0.67533502 0.72709258 0.83534665 0.78896751 0.84256784
 0.69497105 0.66227055 0.16205195 0.82974715 0.8740011  0.68434166
 0.83789007 0.9565323  0.68635114 0.80198843 0.85040122 0.84160951
 0.12810264 0.81173082 0.83921473 0.79722851 0.90178145 0.84952998
 0.27368199 0.7567004  0.78496842 0.66246018 0.84181478 0.26269451
 0.835083   0.81085688 0.32123643 0.85262786 0.86354061 0.19020761
 0.66219549 0.30287531 0.76386101 0.75813337 0.7608713  0.78742368
 0.75164874 0.89282867 0.70483684 0.78523206 0.83708409 0.86134259
 0.20587713]
```

[30]:

```
roc_auc = metrics.roc_auc_score(y_test, lr_pred_probabilities)
log_loss = metrics.log_loss(y_test, lr_pred_probabilities)
print("The ROC (Receiver Operating Characteristic): ", roc_auc)
print("The AUC (Area Under the Curve): ", log_loss)
```

```
The ROC (Receiver Operating Characteristic):  0.714498510427011
The AUC (Area Under the Curve):  0.5413461628763725
```

[31]:

```
print('Loan status prediction for trained data: ' , lr_prediction)
```

```
Loan status prediction for trained data:  [1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0
 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1
 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0]
```

[32]:

```
feature_names = df.columns[1:-1]

# Get the coefficients (parameters) of the logistic regression model
coefficients = model.coef_[0]

# Create a data frame to store the coefficients and corresponding feature names
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

# Calculate the odds ratios by exponentiating the coefficients
coef_df['Odds Ratio'] = np.exp(coef_df['Coefficient'])

# Print or display the DataFrame
print(coef_df)
```

```
              Feature  Coefficient  Odds Ratio
0              Gender     0.117942    1.125179
1             Married     0.403391    1.496891
2          Dependents    -0.271102    0.762539
```

```
3           Education    0.657847    1.930632
4       Self_Employed   -0.038987    0.961763
5     ApplicantIncome   -0.000015    0.999985
6   CoapplicantIncome   -0.000069    0.999931
7          LoanAmount   -0.000621    0.999379
8    Loan_Amount_Term   -0.004527    0.995484
9      Credit_History    2.431141   11.371850
10      Property_Area    0.099192    1.104279
```

[33]:
```
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\dell\anaconda3\lib\site-
packages (7.10.0)
Requirement already satisfied: beautifulsoup4 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (4.12.2)
Requirement already satisfied: bleach!=5.0.0 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (4.1.0)
Requirement already satisfied: defusedxml in c:\users\dell\anaconda3\lib\site-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in c:\users\dell\anaconda3\lib\site-
packages (from nbconvert) (3.1.3)
Requirement already satisfied: jupyter-core>=4.7 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (5.5.0)
Requirement already satisfied: jupyterlab-pygments in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: markupsafe>=2.0 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (2.1.3)
Requirement already satisfied: mistune<4,>=2.0.3 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (2.0.4)
Requirement already satisfied: nbclient>=0.5.0 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (0.8.0)
Requirement already satisfied: nbformat>=5.7 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (5.9.2)
Requirement already satisfied: packaging in c:\users\dell\anaconda3\lib\site-
packages (from nbconvert) (23.1)
Requirement already satisfied: pandocfilters>=1.4.1 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (2.15.1)
Requirement already satisfied: tinycss2 in c:\users\dell\anaconda3\lib\site-
packages (from nbconvert) (1.2.1)
Requirement already satisfied: traitlets>=5.1 in
c:\users\dell\anaconda3\lib\site-packages (from nbconvert) (5.7.1)
Requirement already satisfied: six>=1.9.0 in c:\users\dell\anaconda3\lib\site-
packages (from bleach!=5.0.0->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in c:\users\dell\anaconda3\lib\site-
packages (from bleach!=5.0.0->nbconvert) (0.5.1)
Requirement already satisfied: platformdirs>=2.5 in
```

```
c:\users\dell\anaconda3\lib\site-packages (from jupyter-core>=4.7->nbconvert)
(3.10.0)
Requirement already satisfied: pywin32>=300 in c:\users\dell\anaconda3\lib\site-
packages (from jupyter-core>=4.7->nbconvert) (305.1)
Requirement already satisfied: jupyter-client>=6.1.12 in
c:\users\dell\anaconda3\lib\site-packages (from nbclient>=0.5.0->nbconvert)
(8.6.0)
Requirement already satisfied: fastjsonschema in
c:\users\dell\anaconda3\lib\site-packages (from nbformat>=5.7->nbconvert)
(2.16.2)
Requirement already satisfied: jsonschema>=2.6 in
c:\users\dell\anaconda3\lib\site-packages (from nbformat>=5.7->nbconvert)
(4.19.2)
Requirement already satisfied: soupsieve>1.2 in
c:\users\dell\anaconda3\lib\site-packages (from beautifulsoup4->nbconvert) (2.5)
Requirement already satisfied: attrs>=22.2.0 in
c:\users\dell\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\users\dell\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in
c:\users\dell\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in
c:\users\dell\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.10.6)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\dell\anaconda3\lib\site-packages (from jupyter-
client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in c:\users\dell\anaconda3\lib\site-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (25.1.2)
Requirement already satisfied: tornado>=6.2 in c:\users\dell\anaconda3\lib\site-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.3.3)
```