Loan Eligibility Prediction

September 7, 2024

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

1 preform EDA

[3]:
```
df = pd.read_csv('loan-test.csv')
df.head()
```

[3]:
```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

[4]:
```
df.tail()
```

[4]:
```
      Loan_ID Gender Married Dependents     Education Self_Employed  \
362  LP002971   Male     Yes         3+  Not Graduate           Yes
```

```
363  LP002975    Male      Yes           0      Graduate            No
364  LP002980    Male       No           0      Graduate            No
365  LP002986    Male      Yes           0      Graduate            No
366  LP002989    Male       No           0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
362             4009               1777       113.0             360.0
363             4158                709       115.0             360.0
364             3250               1993       126.0             360.0
365             5000               2393       158.0             360.0
366             9200                  0        98.0             180.0

     Credit_History Property_Area
362             1.0         Urban
363             1.0         Urban
364             NaN     Semiurban
365             1.0         Rural
366             1.0         Rural
```

[5]:
```
df.describe(include='all')
```

[5]:
```
         Loan_ID Gender Married Dependents Education Self_Employed  \
count        367    356     367        357       367           344
unique       367      2       2          4         2             2
top     LP002989   Male     Yes          0  Graduate            No
freq           1    286     233        200       283           307
mean         NaN    NaN     NaN        NaN       NaN           NaN
std          NaN    NaN     NaN        NaN       NaN           NaN
min          NaN    NaN     NaN        NaN       NaN           NaN
25%          NaN    NaN     NaN        NaN       NaN           NaN
50%          NaN    NaN     NaN        NaN       NaN           NaN
75%          NaN    NaN     NaN        NaN       NaN           NaN
max          NaN    NaN     NaN        NaN       NaN           NaN

        ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count        367.000000         367.000000  362.000000        361.000000
unique              NaN                NaN         NaN               NaN
top                 NaN                NaN         NaN               NaN
freq                NaN                NaN         NaN               NaN
mean        4805.599455        1569.577657  136.132597        342.537396
std         4910.685399        2334.232099   61.366652         65.156643
min            0.000000           0.000000   28.000000          6.000000
25%         2864.000000           0.000000  100.250000        360.000000
50%         3786.000000        1025.000000  125.000000        360.000000
75%         5060.000000        2430.500000  158.000000        360.000000
max        72529.000000       24000.000000  550.000000        480.000000
```

```
        Credit_History Property_Area
count       338.000000           367
unique             NaN             3
top                NaN         Urban
freq               NaN           140
mean          0.825444           NaN
std           0.380150           NaN
min           0.000000           NaN
25%           1.000000           NaN
50%           1.000000           NaN
75%           1.000000           NaN
max           1.000000           NaN
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             356 non-null    object
 2   Married            367 non-null    object
 3   Dependents         357 non-null    object
 4   Education          367 non-null    object
 5   Self_Employed      344 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

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

[7]:
```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
```

```
Credit_History       29
Property_Area         0
dtype: int64
```

[8]:
```
df['Credit_History'] = df['Credit_History'].fillna( df['Credit_History'].
↪dropna().mode().values[0] )
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna( df['Loan_Amount_Term'].
↪dropna().mode().values[0] )
```

[9]:
```
# fill na with mode fro categorical data and mean for the numerical data
df['Gender']= df['Gender'].fillna(df['Gender'].mode()[0])
df['Dependents']= df['Dependents'].fillna(df['Dependents'].mode()[0])
df['Self_Employed']= df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
df['Credit_History']= df['Credit_History'].fillna(df['Credit_History'].mean())
df['LoanAmount']= df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term']= df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].
↪mean())
df['Property_Area'] = df['Property_Area'].map({'Rural':0,'Semiurban':2,'Urban':
↪1}).astype('int')
df['Dependents'].replace('3+','3',inplace=True)
```

[18]:
```
df['Dependents'] = df['Dependents'].astype('int')
```

[20]:
```
df.drop(columns='Loan_ID',inplace=True)
```

[26]:
```
# creat new column with total income
df['total_income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
```

[28]:
```
# check for NAN
df.isnull().sum()
```

[28]:
```
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
total_income         0
dtype: int64
```

[24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             367 non-null    int64
 1   Married            367 non-null    int64
 2   Dependents         367 non-null    int64
 3   Education          367 non-null    int64
 4   Self_Employed      367 non-null    int64
 5   ApplicantIncome    367 non-null    int64
 6   CoapplicantIncome  367 non-null    int64
 7   LoanAmount         367 non-null    float64
 8   Loan_Amount_Term   367 non-null    float64
 9   Credit_History     367 non-null    float64
 10  Property_Area      367 non-null    int64
dtypes: float64(3), int64(8)
memory usage: 31.7 KB
```

[30]:

```
df.head()
```

[30]:

```
   Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
0       1        1           0          1              0             5720
1       1        1           1          1              0             3076
2       1        1           2          1              0             5000
3       1        1           2          1              0             2340
4       1        0           0          0              0             3276

   CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                  0       110.0             360.0             1.0
1               1500       126.0             360.0             1.0
2               1800       208.0             360.0             1.0
3               2546       100.0             360.0             1.0
4                  0        78.0             360.0             1.0

   Property_Area  total_income
0              1          5720
1              1          4576
2              1          6800
3              1          4886
4              1          3276
```

2 Modeling

[ ]:

```
df.columns
```

[32]:
```python
# split our data in to target and feature
X = df.drop('Married',axis=1)
y= df['Married']
```

[34]:
```python
# scale our data
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
```

[36]:
```python
X=scale.fit_transform(X)
```

[38]:
```python
X
```

[38]:
```
array([[ 0.48547939, -0.75822199,  0.5448117 , …,  0.4376739 ,
        -0.01732564, -0.12618159],
       [ 0.48547939,  0.18187082,  0.5448117 , …,  0.4376739 ,
        -0.01732564, -0.34650636],
       [ 0.48547939,  1.12196363,  0.5448117 , …,  0.4376739 ,
        -0.01732564,  0.08181731],
       …,
       [ 0.48547939, -0.75822199,  0.5448117 , …,  0.4376739 ,
         1.25437613, -0.21804778],
       [ 0.48547939, -0.75822199,  0.5448117 , …,  0.4376739 ,
        -1.28902741,  0.19602411],
       [ 0.48547939, -0.75822199,  0.5448117 , …,  0.4376739 ,
        -1.28902741,  0.54403709]])
```

[40]:
```python
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

[42]:
```python
model_df = {}

def model_val(model,X,y):
    # spliting dataset for training and testing
    X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                      test_size=0.20,
                                                      random_state=42)

    # training the model
    model.fit(X_train, y_train)

    # asking model for prediction
    y_pred = model.predict(X_test)

    # checking model's prediction accuracy
    print(f"{model} accuracy is {accuracy_score(y_test,y_pred)}")
```

```
    # to find the best model we use cross-validation, thru this we can compare␣
↪different algorithms
    # In this we use whole dataset to for testing not just 20%, but one at a␣
↪time and summarize
    # the result at the end.

    # 5-fold cross-validation (but 10-fold cross-validation is common in␣
↪practise)
    score = cross_val_score(model,X,y,cv=3)  # it will divides the dataset into␣
↪5 parts and during each iteration
                                            # uses (4,1) combination for␣
↪training and testing

    print(f"{model} Avg cross val score is {np.mean(score)}")
    model_df[model] = round(np.mean(score)*100,2)
```

[44]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

# passing this model object of LogisticRegression Class in the function we've␣
↪created
model_val(model,X,y)
```

```
LogisticRegression() accuracy is 0.6891891891891891
LogisticRegression() Avg cross val score is 0.6811053356435203
```

[46]:
```
from sklearn import svm

model = svm.SVC()
model_val(model,X,y)
```

```
SVC() accuracy is 0.6756756756756757
SVC() Avg cross val score is 0.6728864009951575
```

[48]:
```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model_val(model,X,y)
```

```
DecisionTreeClassifier() accuracy is 0.6081081081081081
DecisionTreeClassifier() Avg cross val score is 0.6293704740326092
```

[50]:
```
from sklearn.ensemble import RandomForestClassifier

model =RandomForestClassifier()
model_val(model,X,y)
```

```
RandomForestClassifier() accuracy is 0.6486486486486487
RandomForestClassifier() Avg cross val score is 0.6893464836287707
```

[52]:

```python
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier()
model_val(model,X,y)
```

```
GradientBoostingClassifier() accuracy is 0.6891891891891891
GradientBoostingClassifier() Avg cross val score is 0.7056732862410592
```