credit-card-score-1

March 22, 2024

1 Importing Libraries

[29]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import plotly.express as px

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score


import warnings

warnings.filterwarnings("ignore", category=ImportWarning,␣
↪module='specific_module')
# Code that might trigger the warning for specific_module
```

[30]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

2 Importing dataset

[32]:
```
# Load your dataset
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Credit Card Score/Credit-Score-Data.
↪csv')
df
```

[32]:
```
          ID  Customer_ID  Month           Name  Age        SSN Occupation  \
0       5634         3392      1  Aaron Maashoh   23  821000265  Scientist
1       5635         3392      2  Aaron Maashoh   23  821000265  Scientist
2       5636         3392      3  Aaron Maashoh   23  821000265  Scientist
3       5637         3392      4  Aaron Maashoh   23  821000265  Scientist
4       5638         3392      5  Aaron Maashoh   23  821000265  Scientist
…          …            …      …              …    …          …          …
99995 155625        37932      4          Nicks   25   78735990   Mechanic
99996 155626        37932      5          Nicks   25   78735990   Mechanic
99997 155627        37932      6          Nicks   25   78735990   Mechanic
99998 155628        37932      7          Nicks   25   78735990   Mechanic
99999 155629        37932      8          Nicks   25   78735990   Mechanic

       Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  …  \
0           19114.12            1824.843333                  3  …
1           19114.12            1824.843333                  3  …
2           19114.12            1824.843333                  3  …
3           19114.12            1824.843333                  3  …
4           19114.12            1824.843333                  3  …
…                  …                      …                  …  …
99995       39628.99            3359.415833                  4  …
99996       39628.99            3359.415833                  4  …
99997       39628.99            3359.415833                  4  …
99998       39628.99            3359.415833                  4  …
99999       39628.99            3359.415833                  4  …

       Credit_Mix  Outstanding_Debt  Credit_Utilization_Ratio  \
0            Good            809.98                 26.822620
1            Good            809.98                 31.944960
2            Good            809.98                 28.609352
3            Good            809.98                 31.377862
4            Good            809.98                 24.797347
…               …                 …                         …
99995        Good            502.38                 34.663572
99996        Good            502.38                 40.565631
99997        Good            502.38                 41.255522
99998        Good            502.38                 33.638208
99999        Good            502.38                 34.192463

      Credit_History_Age  Payment_of_Min_Amount  Total_EMI_per_month  \
0                    265                     No            49.574949
1                    266                     No            49.574949
2                    267                     No            49.574949
3                    268                     No            49.574949
4                    269                     No            49.574949
…                      …                      …                    …
99995                378                     No            35.104023
```

```
99996              379                      No             35.104023
99997              380                      No             35.104023
99998              381                      No             35.104023
99999              382                      No             35.104023

       Amount_invested_monthly                Payment_Behaviour  \
0                    21.465380   High_spent_Small_value_payments
1                    21.465380    Low_spent_Large_value_payments
2                    21.465380   Low_spent_Medium_value_payments
3                    21.465380    Low_spent_Small_value_payments
4                    21.465380  High_spent_Medium_value_payments
…                         …                               …
99995                24.028477   High_spent_Large_value_payments
99996                24.028477  High_spent_Medium_value_payments
99997                24.028477   High_spent_Large_value_payments
99998                24.028477    Low_spent_Large_value_payments
99999                24.028477  High_spent_Medium_value_payments

      Monthly_Balance  Credit_Score
0          312.494089          Good
1          284.629163          Good
2          331.209863          Good
3          223.451310          Good
4          341.489231          Good
…                 …           …
99995      479.866228          Poor
99996      496.651610          Poor
99997      516.809083          Poor
99998      319.164979      Standard
99999      393.673696          Poor

[100000 rows x 28 columns]
```

[33]:
```
df.shape
```

[33]:
```
(100000, 28)
```

3 Data pre-processing

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
```

```
 0   ID                        100000 non-null  int64
 1   Customer_ID               100000 non-null  int64
 2   Month                     100000 non-null  int64
 3   Name                      100000 non-null  object
 4   Age                       100000 non-null  int64
 5   SSN                       100000 non-null  int64
 6   Occupation                100000 non-null  object
 7   Annual_Income             100000 non-null  float64
 8   Monthly_Inhand_Salary     100000 non-null  float64
 9   Num_Bank_Accounts         100000 non-null  int64
 10  Num_Credit_Card           100000 non-null  int64
 11  Interest_Rate             100000 non-null  int64
 12  Num_of_Loan               100000 non-null  int64
 13  Type_of_Loan              100000 non-null  object
 14  Delay_from_due_date       100000 non-null  int64
 15  Num_of_Delayed_Payment    100000 non-null  int64
 16  Changed_Credit_Limit      100000 non-null  float64
 17  Num_Credit_Inquiries      100000 non-null  int64
 18  Credit_Mix                100000 non-null  object
 19  Outstanding_Debt          100000 non-null  float64
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        100000 non-null  int64
 22  Payment_of_Min_Amount     100000 non-null  object
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   100000 non-null  float64
 25  Payment_Behaviour         100000 non-null  object
 26  Monthly_Balance           100000 non-null  float64
 27  Credit_Score              100000 non-null  object
dtypes: float64(8), int64(13), object(7)
memory usage: 21.4+ MB
```

[35]:
```
#Checking for null values
df.isna().sum()
```

[35]:
```
ID                          0
Customer_ID                 0
Month                       0
Name                        0
Age                         0
SSN                         0
Occupation                  0
Annual_Income               0
Monthly_Inhand_Salary       0
Num_Bank_Accounts           0
Num_Credit_Card             0
Interest_Rate               0
Num_of_Loan                 0
```

```
Type_of_Loan                0
Delay_from_due_date         0
Num_of_Delayed_Payment      0
Changed_Credit_Limit        0
Num_Credit_Inquiries        0
Credit_Mix                  0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Credit_History_Age          0
Payment_of_Min_Amount       0
Total_EMI_per_month         0
Amount_invested_monthly     0
Payment_Behaviour           0
Monthly_Balance             0
Credit_Score                0
dtype: int64
```

[36]:
```python
#Checking for duplicated
df.duplicated().sum()
```

[36]:
```
0
```

[37]:
```python
# Value count for each value
for i in df.columns:
    print(i,'\n',df[i].value_counts())
    print('-'*90)
```

```
ID
 5634      1
105608    1
105642    1
105637    1
105636    1
         ..
55629     1
55628     1
55627     1
55626     1
155629    1
Name: ID, Length: 100000, dtype: int64
--------------------------------------------------------------------------------
----------
Customer_ID
 3392     8
39924    8
23267    8
48794    8
```

```
18548    8
        ..
11956    8
30819    8
40329    8
49221    8
37932    8
Name: Customer_ID, Length: 12500, dtype: int64
--------------------------------------------------------------------------------
----------
Month
 1    12500
2    12500
3    12500
4    12500
5    12500
6    12500
7    12500
8    12500
Name: Month, dtype: int64
--------------------------------------------------------------------------------
----------
Name
 Jessicad              48
Langep                48
Stevex                48
Vaughanl              40
Ronald Groverk        40
                      ..
Breidthardtj           8
Sven Egenterx          8
Antonella Ciancioc     8
Valentina Zan          8
Nicks                  8
Name: Name, Length: 10128, dtype: int64
--------------------------------------------------------------------------------
----------
Age
 38    3070
28    3045
31    3037
26    3025
32    2969
36    2953
25    2952
27    2951
35    2940
39    2927
```

```
34    2922
44    2902
22    2890
19    2875
41    2865
20    2833
37    2832
29    2823
43    2809
30    2807
21    2792
24    2789
23    2719
45    2712
40    2695
42    2643
33    2623
18    2427
46    1670
15    1615
17    1551
16    1505
49    1419
48    1416
55    1395
53    1394
52    1388
54    1342
51    1332
50    1305
47    1265
14    1197
56     379
Name: Age, dtype: int64
--------------------------------------------------------------------------------
----------
SSN
 821000265    8
544050223    8
381365261    8
994731178    8
647449598    8
             ..
936122774    8
91611869     8
576385212    8
281301712    8
78735990     8
```

```
Name: SSN, Length: 12500, dtype: int64
--------------------------------------------------------------------------------
----------
Occupation
 Lawyer          7096
Engineer         6864
Architect        6824
Mechanic         6776
Scientist        6744
Accountant       6744
Developer        6720
Media_Manager    6720
Teacher          6672
Entrepreneur     6648
Doctor           6568
Journalist       6536
Manager          6432
Musician         6352
Writer           6304
Name: Occupation, dtype: int64
--------------------------------------------------------------------------------
----------
Annual_Income
 20867.670    16
9141.630     16
32543.380    16
40341.160    16
22434.160    16
             ..
18317.260     8
14784.450     8
60573.960     8
18413.795     8
39628.990     8
Name: Annual_Income, Length: 12488, dtype: int64
--------------------------------------------------------------------------------
----------
Monthly_Inhand_Salary
 6769.130000     16
6639.560000     16
2295.058333     16
6082.187500     16
6358.956667     16
                ..
1056.522397      1
1573.927963      1
4722.318333      1
611.734883       1
```

```
10823.060060     1
Name: Monthly_Inhand_Salary, Length: 13241, dtype: int64
--------------------------------------------------------------------------------
----------
Num_Bank_Accounts
 6     13175
7     12999
8     12940
4     12343
5     12298
3     12107
9      5503
10     5329
1      4540
0      4417
2      4340
11        9
Name: Num_Bank_Accounts, dtype: int64
--------------------------------------------------------------------------------
----------
Num_Credit_Card
 5     18903
7     17024
6     16932
4     14362
3     13560
8      5073
10     4962
9      4753
2      2196
1      2185
11       36
0        14
Name: Num_Credit_Card, dtype: int64
--------------------------------------------------------------------------------
----------
Interest_Rate
 8     5104
5     5096
6     4832
12    4648
10    4616
7     4584
9     4576
11    4512
18    4192
15    4072
20    4008
```

```
17    3888
16    3800
19    3704
3     2824
1     2744
4     2640
2     2520
13    2432
14    2272
32    1776
22    1752
24    1736
30    1728
23    1720
29    1696
28    1648
27    1640
25    1608
21    1592
34    1528
26    1528
33    1496
31    1488
Name: Interest_Rate, dtype: int64
--------------------------------------------------------------------------------
----------
Num_of_Loan
 3    15752
2    15712
4    15456
0    11408
1    11128
6     8144
7     7680
5     7528
9     3856
8     3336
Name: Num_of_Loan, dtype: int64
--------------------------------------------------------------------------------
----------
Type_of_Loan
 No Data
11408
Not Specified
1408
Credit-Builder Loan
1280
Personal Loan
```

```
1272
Debt Consolidation Loan
1264
                                                       …
Not Specified, Mortgage Loan, Auto Loan, and Payday Loan
8
Payday Loan, Mortgage Loan, Debt Consolidation Loan, and Student Loan
8
Debt Consolidation Loan, Auto Loan, Personal Loan, Debt Consolidation Loan,
Student Loan, and Credit-Builder Loan                         8
Student Loan, Auto Loan, Student Loan, Credit-Builder Loan, Home Equity Loan,
Debt Consolidation Loan, and Debt Consolidation Loan        8
Personal Loan, Auto Loan, Mortgage Loan, Student Loan, and Student Loan
8
Name: Type_of_Loan, Length: 6261, dtype: int64
--------------------------------------------------------------------------------
----------
Delay_from_due_date
 15    3596
13    3424
8     3324
14    3313
10    3281
      …
59     528
39     525
43     502
46     490
37     490
Name: Delay_from_due_date, Length: 63, dtype: int64
--------------------------------------------------------------------------------
----------
Num_of_Delayed_Payment
 19    5982
17    5832
10    5802
16    5768
15    5724
18    5668
20    5584
12    5493
9     5399
8     5300
11    5272
14    4503
13    4332
21    2717
7     2535
```

```
22    2495
6     2491
23    2304
5     2263
25    2241
0     2081
3     2074
2     2071
24    2045
1     2043
4     1981
Name: Num_of_Delayed_Payment, dtype: int64
--------------------------------------------------------------------------------
----------
Changed_Credit_Limit
 8.22     139
11.50    128
11.32    127
7.35     124
10.06    124
        …
28.34      1
22.77      1
29.73      1
27.63      1
21.17      1
Name: Changed_Credit_Limit, Length: 2860, dtype: int64
--------------------------------------------------------------------------------
----------
Num_Credit_Inquiries
 4     11690
3      9188
6      8399
7      8362
2      8335
8      8133
1      7796
0      7190
5      5951
9      5503
11     5261
10     5130
12     4729
13     1564
14     1116
15      866
16      483
17      304
```

```
Name: Num_Credit_Inquiries, dtype: int64
--------------------------------------------------------------------------------
----------
Credit_Mix
 Standard    45848
Good        30384
Bad         23768
Name: Credit_Mix, dtype: int64
--------------------------------------------------------------------------------
----------
Outstanding_Debt
 1109.03    24
1151.70    24
1360.45    24
460.46     24
1058.13    16
           ..
4230.04     8
641.99      8
98.61       8
2614.48     8
502.38      8
Name: Outstanding_Debt, Length: 12203, dtype: int64
--------------------------------------------------------------------------------
----------
Credit_Utilization_Ratio
 26.407909    2
33.163023    2
26.822620    1
30.462162    1
33.933755    1
             ..
38.730069    1
30.017515    1
27.279794    1
27.002436    1
34.192463    1
Name: Credit_Utilization_Ratio, Length: 99998, dtype: int64
--------------------------------------------------------------------------------
----------
Credit_History_Age
 190    488
232    485
191    484
215    483
213    483
      …
3       21
```

```
2       15
403     15
404     15
1        2
Name: Credit_History_Age, Length: 404, dtype: int64
--------------------------------------------------------------------------------
----------
Payment_of_Min_Amount
 Yes    52326
No     35667
NM     12007
Name: Payment_of_Min_Amount, dtype: int64
--------------------------------------------------------------------------------
----------
Total_EMI_per_month
 0.000000      10985
49.574949         8
46.354927         8
68.686857         8
137.778067        8
              …
841.840387        1
82.544594         1
57.268474         1
144.179455        1
859.516628        1
Name: Total_EMI_per_month, Length: 11890, dtype: int64
--------------------------------------------------------------------------------
----------
Amount_invested_monthly
 0.000000      1920
110.634894       8
14.954207        8
82.215338        8
37.771077        8
              …
58.064394        8
132.912602       8
83.539607        8
33.925532        8
24.028477        8
Name: Amount_invested_monthly, Length: 12261, dtype: int64
--------------------------------------------------------------------------------
----------
Payment_Behaviour
 Low_spent_Small_value_payments      28616
High_spent_Medium_value_payments    19738
High_spent_Large_value_payments     14726
```

```
Low_spent_Medium_value_payments     14399
High_spent_Small_value_payments     11764
Low_spent_Large_value_payments      10757
Name: Payment_Behaviour, dtype: int64
--------------------------------------------------------------------------------
----------
Monthly_Balance
 236.241829     8
1183.930696    8
235.451585     7
281.207616     7
1040.212843    7
              ..
469.067198     1
645.980641     1
463.759938     1
623.262089     1
393.673696     1
Name: Monthly_Balance, Length: 98492, dtype: int64
--------------------------------------------------------------------------------
----------
Credit_Score
 Standard    53174
Poor        28998
Good        17828
Name: Credit_Score, dtype: int64
--------------------------------------------------------------------------------
----------
```

4 EDA

[38]:
```
# Create a histogram for age distribution with specified bin size
fig = px.histogram(data_frame=df, x='Age', title='Age Distribution', nbins=20)
fig.show()
```

5 Income Analysis

[39]:
```
import plotly.express as px

# Create individual box plots for annual income and monthly in-hand salary
fig = px.box(data_frame=df, y='Annual_Income', title='Annual Income Box Plot')
fig.show()

fig = px.box(data_frame=df, y='Monthly_Inhand_Salary', title='Monthly In-hand␣
 ↪Salary Box Plot')
fig.show()
```

6 Payment Behavior

[40]:
```
# Create a pie chart for payment behavior
payment_behavior_counts = df['Payment_Behaviour'].value_counts()
fig = px.pie(values=payment_behavior_counts, names=payment_behavior_counts.
↪index,
             title='Payment Behavior')
fig.show()
```

7 Label encoder

[41]:
```
# Encode categorical variables
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    df[column] = label_encoders[column].fit_transform(df[column])
```

[42]:
```
# Sample data (replace with your actual data)
X = df[["Annual_Income", "Monthly_Inhand_Salary",
        "Num_Bank_Accounts", "Num_Credit_Card",
        "Interest_Rate", "Num_of_Loan",
        "Delay_from_due_date", "Num_of_Delayed_Payment",
        "Credit_Mix", "Outstanding_Debt",
        "Credit_History_Age", "Monthly_Balance"]]
y = df["Credit_Score"]
```

[43]:
```
# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[44]:
```
y
```

[44]:
```
0        0
1        0
2        0
3        0
4        0
        ..
99995    1
99996    1
99997    1
99998    2
99999    1
Name: Credit_Score, Length: 100000, dtype: int64
```

[44]:
```

```

8 Logistic Regression

[45]:
```python
# Initialize and train the Logistic Regression model
lr = LogisticRegression()
lr.fit(X_train, y_train)

# Make predictions and calculate accuracy
y_pred_lr = lr.predict(X_test)
accuracy_lr = accuracy_score(y_test, y_pred_lr)

print(f"Logistic Regression Accuracy: {accuracy_lr}")
```

```
Logistic Regression Accuracy: 0.5412
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

9 Decision Tree

[46]:
```python
# Initialize and train the Decision Tree model
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

# Make predictions and calculate accuracy
y_pred_dt = dt.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)

print(f"Decision Tree Accuracy: {accuracy_dt}")
```

```
Decision Tree Accuracy: 0.75245
```

10 Random forest

[47]:
```python
# Initialize and train the Random Forest model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
# Make predictions and calculate accuracy
y_pred_rf = rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)

print(f"Random Forest Accuracy: {accuracy_rf}")
```

```
Random Forest Accuracy: 0.8104
```

11 XG Boost

[48]:
```
# Initialize and train the XGBoost model
xgb = XGBClassifier()
xgb.fit(X_train, y_train)

# Make predictions and calculate accuracy
y_pred_xgb = xgb.predict(X_test)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)

print(f"XGBoost Accuracy: {accuracy_xgb}")
```

```
XGBoost Accuracy: 0.7656
```

12 Comparing model

[49]:
```
import plotly.graph_objects as go

# Accuracy scores for each classifier
accuracies = {
    'Logistic Regression': accuracy_lr,
    'Decision Tree': accuracy_dt,
    'Random Forest': accuracy_rf,
    'XGBoost': accuracy_xgb
}

# Create a bar graph using Plotly
fig = go.Figure(data=[
    go.Bar(name='Accuracy', x=list(accuracies.keys()), y=list(accuracies.
  ↪values()))
])

# Update the layout
fig.update_layout(title='Comparison of Model Accuracy Scores',
                  xaxis_title='Classifiers',
                  yaxis_title='Accuracy Score',
                  yaxis_range=[0.4, 1.0])
```

```
# Show the plot
fig.show()
```

13 New Data Predict

[72]:
```
# Sample data for prediction
sample_data = pd.DataFrame({
    'Annual_Income': [20867.670],
    'Monthly_Inhand_Salary': [6769.130000],
    'Num_Bank_Accounts': [6],
    'Num_Credit_Card': [5],
    'Interest_Rate': [8],
    'Num_of_Loan': [3],
    'Delay_from_due_date': [15],
    'Num_of_Delayed_Payment': [19],
    'Credit_Mix': [2],
    'Outstanding_Debt': [1109.03],
    'Credit_History_Age': [190],
    'Monthly_Balance': [236.241829]
})
```

[73]:
```
# Predict using the trained RandomForestClassifier model
predictions = rf.predict(sample_data)

# Map the predicted values to labels
predicted_labels = ['Standard' if pred == 0 else 'Bad' if pred == 1 else 'Good'␣
↪for pred in predictions]

# Print the predicted labels
print(predicted_labels)
```

```
['Good']
```

[ ]:
```
```