Project – Classification – Credit Score

Problem Statement

You are working as a data scientist in a global finance company. Over the years, the company has collected basic bank details and gathered a lot of credit-related information. The management wants to build an intelligent system to segregate the people into credit score brackets to reduce the manual efforts. Given a person's credit-related information, build a machine learning model that can classify the credit score.

In [118…
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
```

In [119…
```python
df = pd.read_csv('credit_score.csv')
df.drop(columns=['ID','Customer_ID', 'Name', 'SSN', 'Type_of_Loan', 'Credit_History_Age'], inplace=True)
df.head()
```

Out[119…

	Month	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_Loan	Delay_from_due_date	...	Num_Credit_Inquiries	Credit_Mix	Outstanding_Debt	Credit_Utilization_Ratio	Payment_of_Min_Amount	Total_EMI_per_month	Amount_invested_monthly	Payment_Behaviour	
0	January	23	Scientist	19114.12	1824.843333	3	4	3	4	3	...	4.0	_	809.98	26.822620	No	49.574949	80.41529543900253	High_spent_Small_value_payments	312.
1	February	23	Scientist	19114.12	NaN	3	4	3	4	-1	...	4.0	Good	809.98	31.944960	No	49.574949	118.28022162236736	Low_spent_Large_value_payments	284.
2	March	-500	Scientist	19114.12	NaN	3	4	3	4	3	...	4.0	Good	809.98	28.609352	No	49.574949	81.699521264648	Low_spent_Medium_value_payments	33
3	April	23	Scientist	19114.12	NaN	3	4	3	4	5	...	4.0	Good	809.98	31.377862	No	49.574949	199.4580743910713	Low_spent_Small_value_payments	223.
4	May	23	Scientist	19114.12	1824.843333	3	4	3	4	6	...	4.0	Good	809.98	24.797347	No	49.574949	41.420153086217326	High_spent_Medium_value_payments	341.

5 rows × 22 columns

EDA

In [120…
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 22 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   Month                     100000 non-null  object 
 1   Age                       100000 non-null  object 
 2   Occupation                100000 non-null  object 
 3   Annual_Income             100000 non-null  object 
 4   Monthly_Inhand_Salary     84998 non-null   float64
 5   Num_Bank_Accounts         100000 non-null  int64  
 6   Num_Credit_Card           100000 non-null  int64  
 7   Interest_Rate             100000 non-null  int64  
 8   Num_of_Loan               100000 non-null  object 
 9   Delay_from_due_date       100000 non-null  int64  
 10  Num_of_Delayed_Payment    92998 non-null   object 
 11  Changed_Credit_Limit      100000 non-null  object 
 12  Num_Credit_Inquiries      98035 non-null   float64
 13  Credit_Mix                100000 non-null  object 
 14  Outstanding_Debt          100000 non-null  object 
 15  Credit_Utilization_Ratio  100000 non-null  float64
 16  Payment_of_Min_Amount     100000 non-null  object 
 17  Total_EMI_per_month       100000 non-null  float64
 18  Amount_invested_monthly   95521 non-null   object 
 19  Payment_Behaviour         100000 non-null  object 
 20  Monthly_Balance           98800 non-null   object 
 21  Credit_Score              100000 non-null  object 
dtypes: float64(4), int64(4), object(14)
memory usage: 16.8+ MB
```

In [121…
```python
df.describe()
```

Out[121…

	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Delay_from_due_date	Num_Credit_Inquiries	Credit_Utilization_Ratio	Total_EMI_per_month
count	84998.000000	100000.000000	100000.00000	100000.000000	100000.000000	98035.000000	100000.000000	100000.000000
mean	4194.170850	17.091280	22.47443	72.466040	21.068780	27.754251	32.285173	1403.118217
std	3183.686167	117.404834	129.05741	466.422621	14.860104	193.177339	5.116875	8306.041270
min	303.645417	-1.000000	0.00000	1.000000	-5.000000	0.000000	20.000000	0.000000
25%	1625.568229	3.000000	4.00000	8.000000	10.000000	3.000000	28.052567	30.306660
50%	3093.745000	6.000000	5.00000	13.000000	18.000000	6.000000	32.305784	69.249473
75%	5957.448333	7.000000	7.00000	20.000000	28.000000	9.000000	36.496663	161.224249
max	15204.633333	1798.000000	1499.00000	5797.000000	67.000000	2597.000000	50.000000	82331.000000

In [122…
```python
df['Occupation'].replace('_______', np.nan, inplace=True)
```

In [123…
```python
df['Age'] = df['Age'].str.replace('_', '')
```

In [124…
```python
df['Age'] = df['Age'].astype('int')
```

In [125…
```python
df['Annual_Income'] = df['Annual_Income'].str.replace('_', '')
df['Annual_Income'] = df['Annual_Income'].astype('float')
```

In [126…
```python
df['Num_of_Loan'] = df['Num_of_Loan'].str.replace('_', '')
df['Num_of_Loan'] = df['Num_of_Loan'].astype('int')
```

In [127…
```python
df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].str.replace('_', '')
df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].astype('float')
```

In [128…
```python
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].replace('_', np.nan)
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].astype('float')
```

In [129…
```python
df['Credit_Mix'] = df['Credit_Mix'].replace('_', np.nan)
df['Credit_Mix'] = df['Credit_Mix'].replace(['Standard', 'Good', 'Bad'], [1,2,0])
```

In [130…
```python
df['Outstanding_Debt'] = df['Outstanding_Debt'].str.replace('_', '')
df['Outstanding_Debt'] = df['Outstanding_Debt'].astype('float')
```

In [131…
```python
df['Payment_of_Min_Amount'] = df['Payment_of_Min_Amount'].replace('NM', 'No')
df['Payment_of_Min_Amount'] = df['Payment_of_Min_Amount'].replace(['Yes', 'No'], [1,0])
```

In [132…
```python
df['Amount_invested_monthly'] = df['Amount_invested_monthly'].str.replace('_', '')
df['Amount_invested_monthly'] = df['Amount_invested_monthly'].astype('float')
```

In [133…
```python
df['Payment_Behaviour'] = df['Payment_Behaviour'].replace('!@9#%8', np.nan)
```

In [134…
```python
df['Monthly_Balance'] = df['Monthly_Balance'].str.replace('_', '')
df['Monthly_Balance'] = df['Monthly_Balance'].astype('float')
```

In [135…
```python
df['Credit_Score'] = df['Credit_Score'].replace(['Standard', 'Poor', 'Good'], [0,1,2])
```

In [136…
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 22 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   Month                     100000 non-null  object 
 1   Age                       100000 non-null  int32  
 2   Occupation                92938 non-null   object 
 3   Annual_Income             100000 non-null  float64
 4   Monthly_Inhand_Salary     84998 non-null   float64
 5   Num_Bank_Accounts         100000 non-null  int64  
 6   Num_Credit_Card           100000 non-null  int64  
 7   Interest_Rate             100000 non-null  int64  
 8   Num_of_Loan               100000 non-null  int32  
 9   Delay_from_due_date       100000 non-null  int64  
 10  Num_of_Delayed_Payment    92998 non-null   float64
 11  Changed_Credit_Limit      97909 non-null   float64
 12  Num_Credit_Inquiries      98035 non-null   float64
 13  Credit_Mix                79805 non-null   float64
 14  Outstanding_Debt          100000 non-null  float64
 15  Credit_Utilization_Ratio  100000 non-null  float64
 16  Payment_of_Min_Amount     100000 non-null  int64  
 17  Total_EMI_per_month       100000 non-null  float64
 18  Amount_invested_monthly   95521 non-null   float64
 19  Payment_Behaviour         92400 non-null   object 
 20  Monthly_Balance           97132 non-null   float64
 21  Credit_Score              100000 non-null  int64  
dtypes: float64(11), int32(2), int64(6), object(3)
memory usage: 16.0+ MB
```

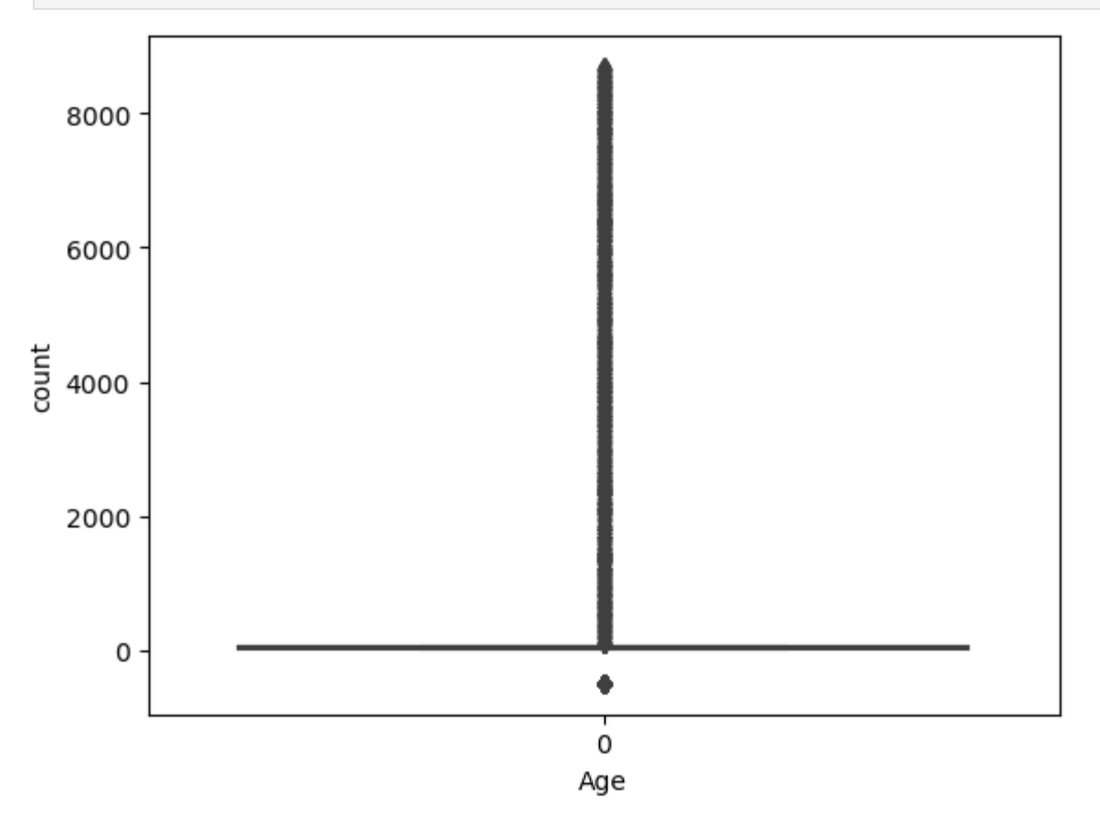
In [137…
```python
df.isna().sum()
```

Out[137…
```
Month                           0
Age                             0
Occupation                   7062
Annual_Income                   0
Monthly_Inhand_Salary       15002
Num_Bank_Accounts               0
Num_Credit_Card                 0
Interest_Rate                   0
Num_of_Loan                     0
Delay_from_due_date             0
Num_of_Delayed_Payment       7002
Changed_Credit_Limit         2091
Num_Credit_Inquiries         1965
Credit_Mix                  20195
Outstanding_Debt                0
Credit_Utilization_Ratio        0
Payment_of_Min_Amount           0
Total_EMI_per_month             0
Amount_invested_monthly      4479
Payment_Behaviour            7600
Monthly_Balance              2868
Credit_Score                    0
dtype: int64
```

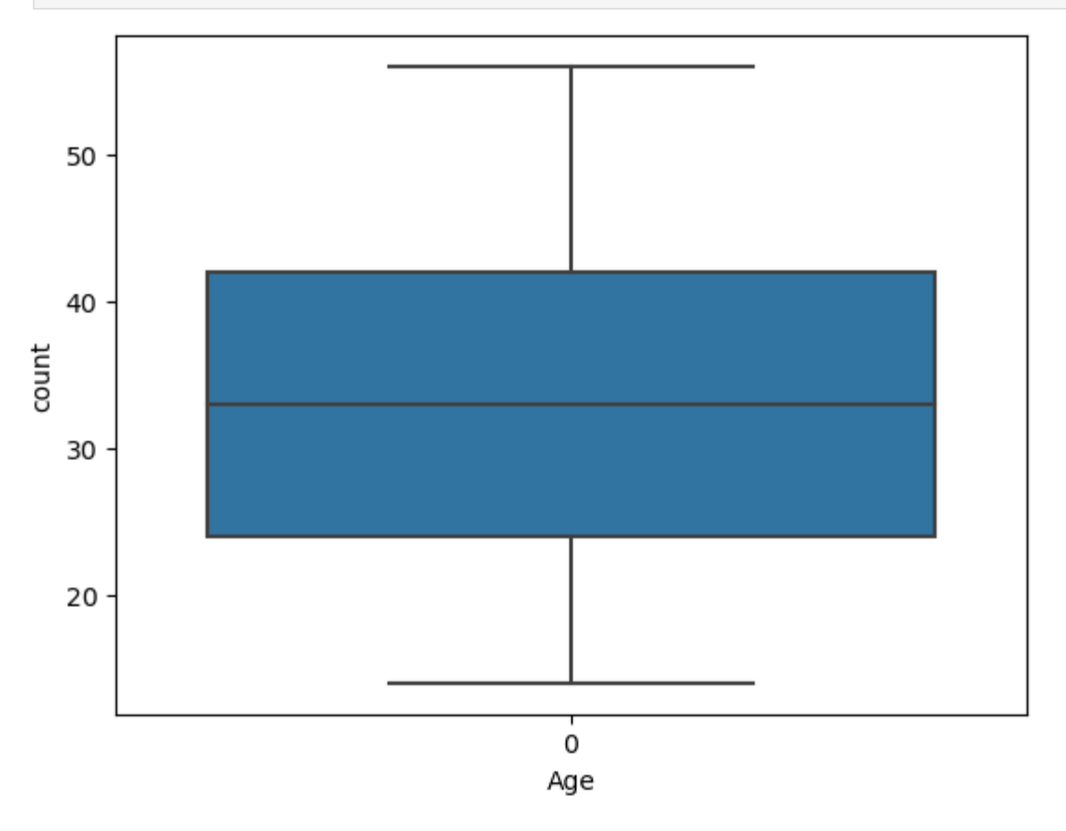
In [138…
```python
df = df.fillna(method='ffill')
df = df.fillna(method='bfill')
df.isnull().sum()
```

Out[138…
```
Month                       0
Age                         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
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
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
```

In [139…
```python
sns.boxplot(df['Age'])
plt.xlabel('Age')
plt.ylabel('count')
plt.show()
```



In [140…
```python
col_names = ['Age']
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1
data = df[(df.Age >= Q1 - 1.5*IQR) & (df.Age <= Q3 + 1.5*IQR)]
sns.boxplot(data['Age'])
plt.xlabel('Age')
plt.ylabel('count')
plt.show()
```



One hot Encoding

In [141…
```python
le = LabelEncoder()
## converting all categorical variables to int
for i in df.columns:
    if df[i].dtype=='object':
        df[i] = le.fit_transform(df[i])
```

In [142…
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 22 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   Month                     100000 non-null  int32  
 1   Age                       100000 non-null  int32  
 2   Occupation                100000 non-null  int32  
 3   Annual_Income             100000 non-null  float64
 4   Monthly_Inhand_Salary     100000 non-null  float64
 5   Num_Bank_Accounts         100000 non-null  int64  
 6   Num_Credit_Card           100000 non-null  int64  
 7   Interest_Rate             100000 non-null  int64  
 8   Num_of_Loan               100000 non-null  int32  
 9   Delay_from_due_date       100000 non-null  int64  
 10  Num_of_Delayed_Payment    100000 non-null  float64
 11  Changed_Credit_Limit      100000 non-null  float64
 12  Num_Credit_Inquiries      100000 non-null  float64
 13  Credit_Mix                100000 non-null  float64
 14  Outstanding_Debt          100000 non-null  float64
 15  Credit_Utilization_Ratio  100000 non-null  float64
 16  Payment_of_Min_Amount     100000 non-null  int64  
 17  Total_EMI_per_month       100000 non-null  float64
 18  Amount_invested_monthly   100000 non-null  float64
 19  Payment_Behaviour         100000 non-null  int32  
 20  Monthly_Balance           100000 non-null  float64
 21  Credit_Score              100000 non-null  int64  
dtypes: float64(11), int32(5), int64(6)
memory usage: 14.9 MB
```

Feature Selection using VIF

In [143…
```python
col_list = []
for col in df.columns:
    if((df[col].dtype!='object') & (col!='Credit_Score')):
        col_list.append(col)


X = df[col_list]
vif_data = pd.DataFrame()
vif_data['feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.columns))]
print(vif_data)
```

```
                     feature       VIF
0                      Month  0.300012
1                        Age  0.974661
2                 Occupation  0.277722
3              Annual_Income  0.985001
4      Monthly_Inhand_Salary  0.365970
5          Num_Bank_Accounts  0.979247
6            Num_Credit_Card  0.970567
7              Interest_Rate  0.976430
8                Num_of_Loan  0.997697
9        Delay_from_due_date  0.332213
10    Num_of_Delayed_Payment  0.981707
11      Changed_Credit_Limit  0.299307
12      Num_Credit_Inquiries  0.979793
13                Credit_Mix  0.321474
14          Outstanding_Debt  0.396141
15  Credit_Utilization_Ratio  0.024506
16     Payment_of_Min_Amount  0.476749
17       Total_EMI_per_month  0.972258
18   Amount_invested_monthly  0.911321
19         Payment_Behaviour  0.310525
20           Monthly_Balance  1.000207
```

Logistic Regression

In [144…
```python
X = df.drop(columns=['Credit_Score'])
y = df['Credit_Score']
```

In [145…
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [146…
```python
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [147…
```python
lr = LogisticRegression()
```

In [148…
```python
lr.fit(X_train, y_train)
```

Out[148…
```
▾ LogisticRegression
LogisticRegression()
```

In [149…
```python
y_pred = lr.predict(X_test)
```

In [150…
```python
accuracy_score(y_test, y_pred)
```

Out[150… 0.61855

In [151…
```python
pd.DataFrame({'actual_value':y_test, 'predicted_value':y_pred})
```

Out[151…

	actual_value	predicted_value
75721	2	2
80184	1	1
19864	2	2
76699	1	1
92991	2	2
...	...	...
32595	0	0
29313	0	0
37862	1	0
53421	0	0
42410	0	0

20000 rows × 2 columns

Decision Tree

In [152…
```python
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Out[152…
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [153…
```python
y_pred = dt.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[153… 0.6965

In [154…
```python
pd.DataFrame({'actual_value':y_test, 'predicted_value':y_pred})
```

Out[154…

	actual_value	predicted_value
75721	2	2
80184	1	1
19864	2	2
76699	1	1
92991	2	2
...	...	...
32595	0	0
29313	0	0
37862	1	0
53421	0	0
42410	0	1

20000 rows × 2 columns

Hyperparameter tuning on DT

In [155…
```python
parameters = {'max_features': ['log2', 'sqrt', 'auto'],
             'criterion': ['entropy', 'gini'],
             'max_depth': [2,3,5,10,50],
             'min_samples_split': [2,3,50,100],
             'min_samples_leaf': [1,5,8,10]}

grid_obg = GridSearchCV(dt, parameters)
grid_obg = grid_obg.fit(X_train, y_train)
dt = grid_obg.best_estimator_
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
acc_dt = round(accuracy_score(y_test, y_pred)*100, 2)
print('Accuracy of Decision Tree model:', acc_dt)
```

```
Accuracy of Decision Tree model: 70.18
```

Random Forest

In [156…
```python
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[156…
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [157…
```python
y_pred = rf.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[157… 0.7975

In [158…
```python
pd.DataFrame({'actual_value':y_test, 'predicted_value':y_pred})
```

Out[158…

	actual_value	predicted_value
75721	2	2
80184	1	1
19864	2	0
76699	1	1
92991	2	2
...	...	...
32595	0	0
29313	0	0
37862	1	0
53421	0	0
42410	0	1

20000 rows × 2 columns

In [ ]: