Credit Banking_Project - 1

EDULYT INDIA

NANDIPATI KUSUMA

Intern Id : EI-3431

May 14, 2025

This project has been implemented using **Python**, leveraging libraries such as **Pandas** for data manipulation and analysis tasks.

1 Project Workflow

1. **Loading the Data:** The data files are loaded into a Jupyter Notebook using **Python** and **Pandas** library. Below is the code used to load the data and the output of the code is given below :

```
import pandas as pd
file_path = 'Credit Banking_Project - 1.xls'
xls = pd.ExcelFile(file_path)
print(xls.sheet_names)
```

```
['Customer Acqusition', 'Spend', 'Repayment']
```

2.Provide a meaningful treatment to all values where age is less than 18.

2.1 Accessing Customer Acquisition data

```
df_acquisition = pd.read_excel(xls, 'Customer Acquisition')
df_acquisition
```

Out[23]:

	Sl No:	Customer	Age	City	Credit Card Product	Limit	Company	Segment
0	1	A1	0.928521	BANGALORE	Gold	500000	C1	Self Employed
1	2	A2	35.534551	CALCUTTA	Silver	100000	C2	Salaried_MNC
2	3	A3	11.559307	COCHIN	Platimum	10000	C3	Salaried_Pvt
3	4	A4	45.820278	BOMBAY	Platimum	10001	C4	Govt
4	5	A5	69.663948	BANGALORE	Platimum	10002	C5	Normal Salary
...	...	...	...	...	...	...	...	...
95	96	A96	29.631637	CHENNAI	Silver	100000	C19	Salaried_Pvt
96	97	A97	20.611833	TRIVANDRUM	Platimum	10000	C20	Govt
97	98	A98	40.538985	CALCUTTA	Platimum	10001	C21	Normal Salary
98	99	A99	21.588666	CALCUTTA	Platimum	10002	C22	Self Employed
99	100	A100	23.607638	COCHIN	Silver	100000	C5	Salaried_MNC

100 rows × 8 columns

2.2 Data cleaning for age column

1. We cant drop the rows where age<18 because those rows has 22% of the entire data .
2. we can go for Mean or Median
3. For Normally or evenly distributed data ,we can go for Mean to fill those age < 18 values.

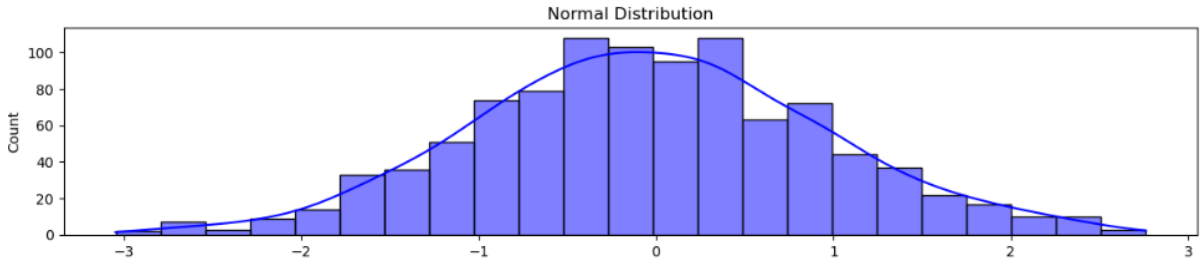

Figure 1: Histogram showing Normal Distribution Bell-Curve

4. For left or right skewed data ,we can use Median to fill those age < 18 values .

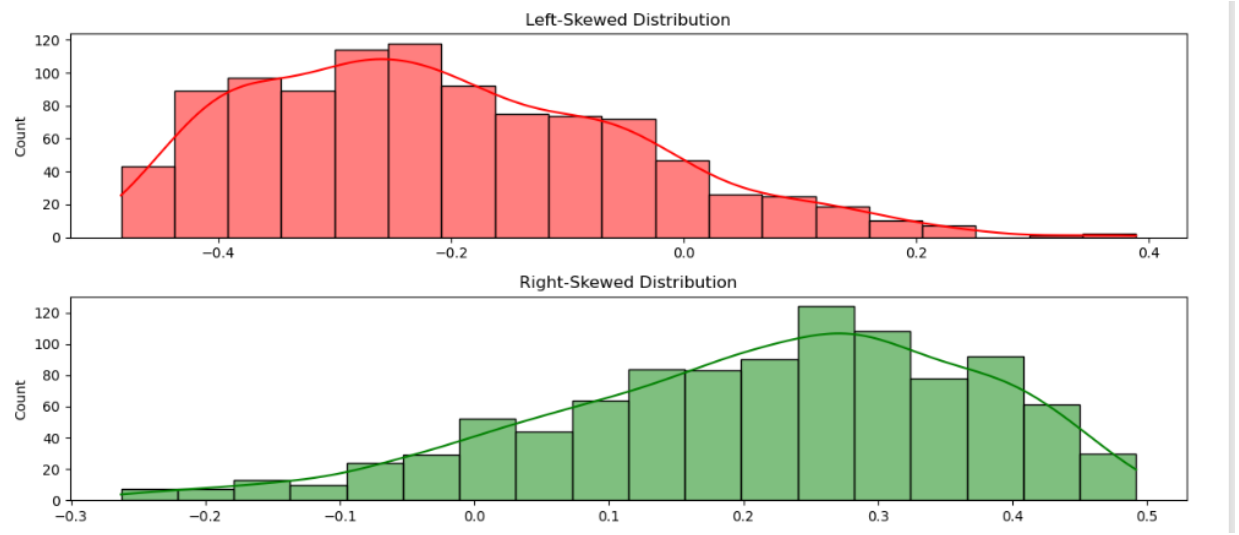

Figure 2: Histogram showing Left and Right Skewed Distribution

5.To know the distribution of 'Age' column ,let's plot Histogram :

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
sns.histplot(df_acquisition['Age'], kde=True, bins=20, color='red',
   edgecolor='black')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

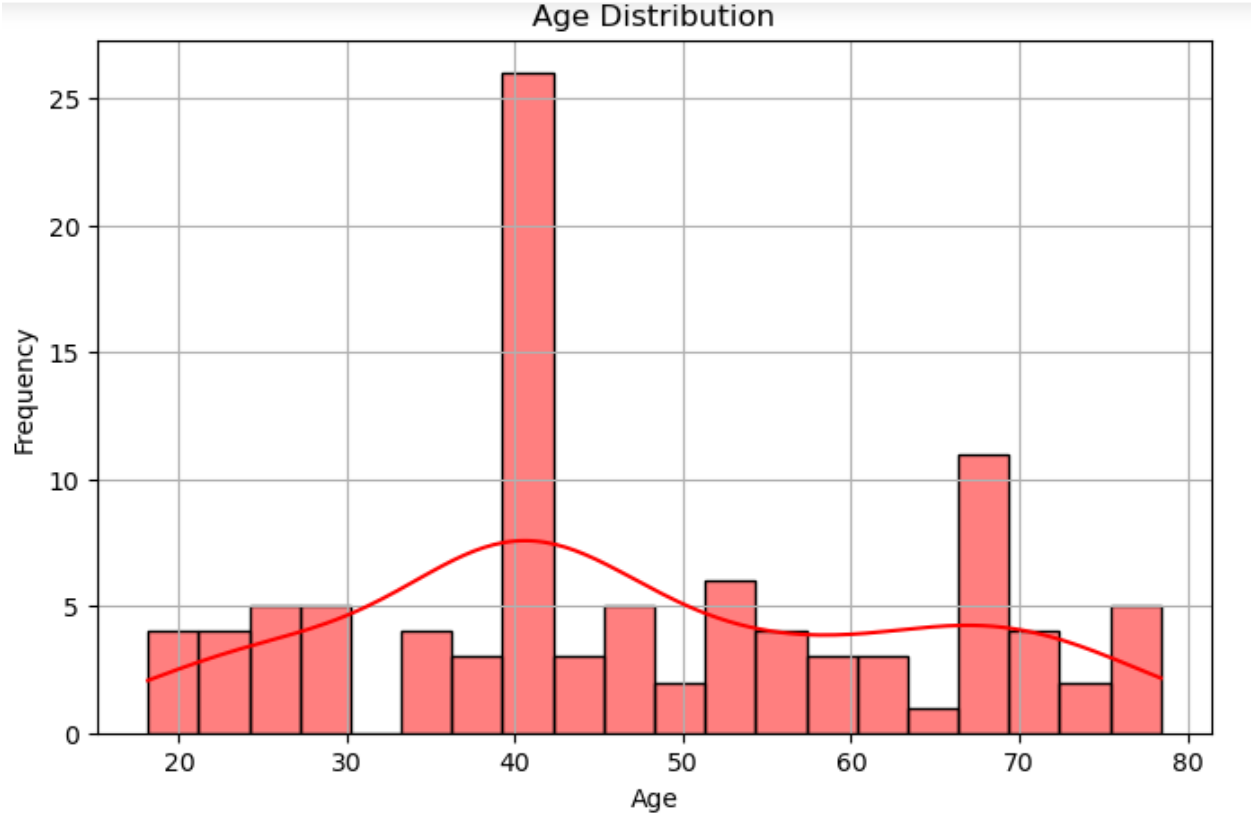

Figure 3: Histogram of 'Age' column forming Bell-Shaped Curve(with outliers)

Filtering out the data into adults and minors :

```
df_adults= df_acquisition[df_acquisition['Age']>=18]
print(df_adults)
df_minors = df_acquisition[df_acquisition['Age']<18]
print(df_minors)
```

	Sl No:	Customer	Age	City	Credit Card Product	Limit	Company	Segment
1	2	A2	35.534551	CALCUTTA	Silver	100000	C2	Salaried_MNC
3	4	A4	45.820278	BOMBAY	Platimum	10001	C4	Govt
4	5	A5	69.663948	BANGALORE	Platimum	10002	C5	Normal Salary
5	6	A6	35.578586	DELHI	Silver	100000	C6	Self Employed
6	7	A7	52.102217	COCHIN	Silver	100001	C7	Salaried_MNC
...	...	...	...	...	...	...	...	...
95	96	A96	29.631637	CHENNAI	Silver	100000	C19	Salaried_Pvt
96	97	A97	20.611833	TRIVANDRUM	Platimum	10000	C20	Govt
97	98	A98	40.538985	CALCUTTA	Platimum	10001	C21	Normal Salary
98	99	A99	21.588666	CALCUTTA	Platimum	10002	C22	Self Employed
99	100	A100	23.607638	COCHIN	Silver	100000	C5	Salaried_MNC

78 rows × 8 columns

Figure 4: Rows of Age > =18

	Sl No:	Customer	Age	City	Credit Card Product	Limit	Company	Segment
0	1	A1	0.928521	BANGALORE	Gold	500000	C1	Self Employed
2	3	A3	11.559307	COCHIN	Platimum	10000	C3	Salaried_Pvt
9	10	A10	4.143754	CALCUTTA	Gold	500000	C10	Normal Salary
11	12	A12	11.182481	BOMBAY	Gold	500000	C12	Self Employed
13	14	A14	6.772363	BANGALORE	Gold	500000	C14	Salaried_Pvt
15	16	A16	5.026450	COCHIN	Gold	500000	C16	Normal Salary
18	19	A19	6.579394	BANGALORE	Platimum	10000	C19	Salaried_Pvt
28	29	A29	0.726493	CALCUTTA	Gold	500000	C11	Salaried_Pvt
29	30	A30	5.537829	BANGALORE	Gold	500000	C12	Govt
36	37	A37	10.683339	BANGALORE	Platimum	100003	C19	Normal Salary
38	39	A39	8.054997	TRIVANDRUM	Platimum	500000	C21	Salaried_MNC
41	42	A42	10.524981	BOMBAY	Gold	500000	C24	Normal Salary
42	43	A43	12.651992	BANGALORE	Gold	500000	C25	Self Employed
52	53	A53	15.339445	DELHI	Platimum	100003	C15	Normal Salary
54	55	A55	4.061398	PATNA	Platimum	500000	C17	Normal Salary
62	63	A63	3.379858	BANGALORE	Gold	500000	C25	Self Employed
63	64	A64	1.840485	DELHI	Gold	500000	C26	Salaried_MNC
64	65	A65	8.985611	COCHIN	Gold	500000	C27	Salaried_Pvt
67	68	A68	16.979379	TRIVANDRUM	Silver	100000	C30	Self Employed
71	72	A72	12.911575	BANGALORE	Silver	100000	C34	Self Employed
74	75	A75	4.234677	BOMBAY	Silver	100003	C37	Self Employed
88	89	A89	7.737083	CALCUTTA	Gold	500000	C12	Govt

Figure 5: Rows of Age < 18

Calculating the mean of adult age :

```
mean_adult_age = df_adults['Age'].mean()
print(f"Mean age of adults: {mean_adult_age}")
```

Mean age of adults: **49.29242405876662**.

Filling up the minors age with the mean_age_of_adults :

```
df_minors['Age'] = mean_adult_age
df_acquisition.loc[df_acquisition['Age']<18,'Age'] =mean_adult_age
df_acquisition
```

Out[20]:

	Sl No:	Customer	Age	City	Credit Card Product	Limit	Company	Segment
0	1	A1	49.292424	BANGALORE	Gold	500000	C1	Self Employed
1	2	A2	35.534551	CALCUTTA	Silver	100000	C2	Salaried_MNC
2	3	A3	49.292424	COCHIN	Platimum	10000	C3	Salaried_Pvt
3	4	A4	45.820278	BOMBAY	Platimum	10001	C4	Govt
4	5	A5	69.663948	BANGALORE	Platimum	10002	C5	Normal Salary
...	...	...	...	...	...	...	...	...
95	96	A96	29.631637	CHENNAI	Silver	100000	C19	Salaried_Pvt
96	97	A97	20.611833	TRIVANDRUM	Platimum	10000	C20	Govt
97	98	A98	40.538985	CALCUTTA	Platimum	10001	C21	Normal Salary
98	99	A99	21.588666	CALCUTTA	Platimum	10002	C22	Self Employed
99	100	A100	23.607638	COCHIN	Silver	100000	C5	Salaried_MNC

100 rows × 8 columns

Figure 6: Customer Acquisition data after applying Data Cleaning to 'Age' Column

Monthly spend of each customer

```
import pandas as pd
df_spend['Month']=pd.to_datetime(df_spend['Month'],format='%d-%b-%y'
    )
df_spend['monthly'] = df_spend['Month'].dt.month
monthly_spend = df_spend.groupby(['Costomer', 'monthly'])['Amount'].
    sum().reset_index()
monthly_spend.rename(columns={'Amount': 'monthly_spend'}, inplace=
    True)
print(monthly_spend)
```

	Costomer	year	month	Amount
0	A1	2004	1	1.511173e+06
1	A1	2004	2	4.138111e+04
2	A1	2004	5	1.311966e+05
3	A1	2005	1	3.984038e+05
4	A1	2005	2	1.404193e+06
..	...	...	...	...
802	A95	2004	1	3.478339e+05
803	A96	2004	1	3.203635e+05
804	A97	2004	1	1.643300e+05
805	A98	2004	1	8.748351e+04
806	A99	2004	1	4.760204e+05

[807 rows x 4 columns]

customer who have spent more than his/her Credit Limit for any particular month

```
#merging with Limit
df_acquisition = df_acquisition.rename(columns={'Customer': '
    Costomer'})
merged_df = grouped.merge(df_acquisition[['Costomer', 'Limit']], on=
    'Costomer', how='left')
print(merged_df)
#Comparing limit and monthly spent of customer
print("customer who have spent more than his/her Credit Limit for
    any particular month")
over_limit = merged_df[merged_df['Amount'] > merged_df['Limit']]
customers_over_limit = over_limit['Costomer'].unique()
customers_over_limit
```

```
customer who have spent more than his/her Credit Limit for any particular month
Out[74]: array(['A1', 'A10', 'A11', 'A12', 'A13', 'A14', 'A15', 'A16', 'A17',
       'A18', 'A19', 'A2', 'A20', 'A21', 'A22', 'A23', 'A24', 'A25',
       'A26', 'A27', 'A28', 'A29', 'A3', 'A30', 'A31', 'A32', 'A33',
       'A34', 'A35', 'A36', 'A37', 'A38', 'A39', 'A4', 'A40', 'A41',
       'A42', 'A43', 'A44', 'A45', 'A46', 'A47', 'A48', 'A49', 'A5',
       'A50', 'A51', 'A52', 'A53', 'A54', 'A55', 'A56', 'A57', 'A58',
       'A59', 'A6', 'A60', 'A61', 'A62', 'A68', 'A69', 'A7', 'A70', 'A71',
       'A72', 'A73', 'A74', 'A75', 'A8', 'A83', 'A84', 'A85', 'A86',
       'A87', 'A9', 'A96', 'A97', 'A98', 'A99'], dtype=object)
```

Monthly repayment of each customer.

```
import pandas as pd

df_repayment['Month'] = pd.to_datetime(df_repayment['Month'])
df_repayment['Month_Num'] = df_repayment['Month'].dt.month
df_repayment['Year'] = df_repayment['Month'].dt.year

monthly_repayment = df_repayment.groupby(['Costomer',
    'Year','Month_Num'])['Amount'].sum().reset_index()
monthly_repayment.rename(columns={'Amount': 'Monthly_Repayment'},
    inplace=True)
```

```
print(monthly_repayment)
```

```
    Costomer  year  month        Amount
0         A1  2004      1  1.362775e+06
1         A1  2005      1  1.581970e+03
2         A1  2004      2  1.911800e+05
3         A1  2005      2  1.199808e+06
4         A1  2006      4  3.712733e+05
..       ...   ...    ...           ...
793      A95  2004      1  7.510949e+04
794      A96  2004      1  1.101390e+05
795      A97  2004      1  1.746064e+05
796      A98  2004      1  9.780260e+04
797      A99  2004      1  3.585899e+05

[798 rows x 4 columns]
```

Highest paying 10 customers.

```
import pandas as pd

df_repayment_sorted = df_repayment.groupby(['Costomer', 'Amount']).
   size().reset_index()
df_repayment_sorted = df_repayment_sorted.sort_values(by='Amount',
   ascending=False)

print(df_repayment_sorted)
```

Out[164]:

	Costomer	Amount
15	A22	9.767171e+06
57	A60	9.262032e+06
58	A61	8.807888e+06
35	A40	8.805085e+06
42	A47	8.529826e+06
38	A43	8.458621e+06
43	A48	8.432804e+06
36	A41	8.374046e+06
44	A49	8.259841e+06
40	A45	8.115210e+06

People in which segment are spending more money

```
total_spent = grouped.groupby('Costomer')['Amount'].sum().
    reset_index()
top_1_customer = total_spent.sort_values(by='Amount', ascending=
    False).head(1)
df_acquisition = df_acquisition.rename(columns={'Customer': '
    Costomer'})
merged_df = top_1_customer.merge(df_acquisition[['Costomer', '
    Segment']], on='Costomer', how='left')
```

Out[163]:

	Costomer	Amount	Segment
0	A22	9.637819e+06	Self Employed

Which age group is spending more money

```
#in which age group people are spending more money
df_acquisition = df_acquisition.rename(columns={'Customer': 'Costomer'})
merged_df = pd.merge(df_acquisition, df_Spend, left_on='Costomer', right_on='Costomer', how='inner')

merged_df['age_group'] = pd.cut(merged_df['Age'],
                                bins=[-float('inf'), 18, 30, 40, float('inf')],
                                labels=['Under 18', '18-29', '30-39', '40 and above'],
                                right=False)
merged_df[['Costomer','age_group','Amount']]
grouped = merged_df.groupby('age_group')['Amount'].sum().reset_index()
grouped = grouped.sort_values(by='Amount', ascending=False).reset_index(drop=True)
value = grouped.loc[0, 'age_group']
print("the highest spending age group is---- ", value)
```

```
the highest spending age group is----  40 and above
```

Which is the most profitable segment?

In [139]:
```
spend_merged = pd.merge(df_acquisition[['Costomer', 'Segment']], df_Spend, on='Costomer', how='inner')
repay_merged = pd.merge(df_acquisition[['Costomer', 'Segment']], df_Repayment, on='Costomer', how='inner')

spend_by_segment = spend_merged.groupby('Segment')['Amount'].sum().reset_index(name='Total_Spend')
repay_by_segment = repay_merged.groupby('Segment')['Amount'].sum().reset_index(name='Total_Repayment')

profit_df = pd.merge(spend_by_segment, repay_by_segment, on='Segment')
profit_df['Profit'] = profit_df['Total_Repayment'] - profit_df['Total_Spend']
most_profitable = profit_df.sort_values(by='Profit', ascending=False).reset_index(drop=True)
print(most_profitable)
value=most_profitable.loc[0,'Segment']
print("the mose profitable segment is----",value)
```

```
         Segment   Total_Spend  Total_Repayment        Profit
0  Self Employed  7.097548e+07     7.055129e+07 -4.241883e+05
1  Normal Salary  1.077071e+08     1.071089e+08 -5.982498e+05
2   Salaried_MNC  6.363949e+07     6.259740e+07 -1.042085e+06
3           Govt  6.732563e+07     6.517141e+07 -2.154222e+06
4   Salaried_Pvt  7.170431e+07     6.577945e+07 -5.924860e+06
the mose profitable segment is---- Self Employed
```

In which category the customers are spending more money?

```
#In which category the customers are spending more money?
grouped=df_Spend.groupby(['Type'])['Amount'].sum().reset_index()
most_spend_category=grouped.sort_values(by='Amount',ascending=False).reset_index(drop=True)
most_spend_category
value=most_spend_category.loc[0,'Type']
print("the most spending Category is----",value)
```

```
the most spending Category is---- PETRO
```

Impose an interest rate of 2.9% for each customer for any due amount.

```
    merged_monthly['due_amount'] = merged_monthly['monthly_spend'] -
        merged_monthly['monthly_repayment']
merged_monthly['due_amount'] = merged_monthly['due_amount'].apply(
   lambda x: x if x > 0 else 0)
merged_monthly['interest'] = merged_monthly['due_amount'] * 0.029
```

	Costomer	year	monthly	due_amount	interest
0	A1	2004	1	4.209483e+06	122075.004360
1	A1	2004	2	0.000000e+00	0.000000
2	A1	2004	5	0.000000e+00	0.000000
3	A1	2005	1	3.339330e+06	96840.574019
4	A1	2005	2	9.947724e+06	288483.988419
..	...	...	...	...	...
802	A95	2004	1	2.727244e+05	7909.007232
803	A96	2004	1	2.102245e+05	6096.511125
804	A97	2004	1	0.000000e+00	0.000000
805	A98	2004	1	0.000000e+00	0.000000
806	A99	2004	1	1.174305e+05	3405.483407

[807 rows x 5 columns]

Monthly profit for the bank

```
merged_df = df_Spend.merge(df_acquisition, left_on='Costomer',
   right_on='Costomer')
merged_df['monthly'] = pd.to_datetime(merged_df['Month']).dt.month

grouped = merged_df.groupby(['monthly', 'Limit'], as_index=False).
   agg(monthly_spend=('Amount', 'sum'))

grouped['bank_profit'] = grouped.apply(
    lambda row: row['Limit'] * 0.02 if row['monthly_spend'] > row['
       Limit'] else 0,
    axis=1
)
print(grouped)
```

	monthly	Limit	monthly_spend	bank_profit
0	1	10000	4.759426e+06	200.00
1	1	10001	3.517977e+06	200.02
2	1	10002	6.002265e+06	200.04
3	1	100000	8.166313e+06	2000.00
4	1	100001	5.125264e+06	2000.02
..	...	...	...	...
90	12	100000	5.156167e+05	2000.00
91	12	100001	7.276435e+05	2000.02
92	12	100002	7.156117e+05	2000.04
93	12	100003	1.873394e+05	2000.06
94	12	500000	2.701744e+06	10000.00

[95 rows x 4 columns]