# Data Visualizatiion - Visionaries

August 1, 2025

[3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from dash import Dash, dcc, html, Input, Output
import plotly.graph_objects as go

# Reading the input Loan Prediction file
df=pd.read_csv("train.csv")

# Head will give the glimpse of top 5 rows in the data
df.head()
```

[3]:

| | UniqueID | disbursed_amount | asset_cost | ltv | branch_id | supplier_id | \ |
|---|---|---|---|---|---|---|---|
| 0 | 420825 | 50578 | 58400 | 89.55 | 67 | 22807 | |
| 1 | 537409 | 47145 | 65550 | 73.23 | 67 | 22807 | |
| 2 | 417566 | 53278 | 61360 | 89.63 | 67 | 22807 | |
| 3 | 624493 | 57513 | 66113 | 88.48 | 67 | 22807 | |
| 4 | 539055 | 52378 | 60300 | 88.39 | 67 | 22807 | |

| | manufacturer_id | Current_pincode_ID | Date.of.Birth | Employment.Type | … | \ |
|---|---|---|---|---|---|---|
| 0 | 45 | 1441 | 01-01-84 | Salaried | … | |
| 1 | 45 | 1502 | 31-07-85 | Self employed | … | |
| 2 | 45 | 1497 | 24-08-85 | Self employed | … | |
| 3 | 45 | 1501 | 30-12-93 | Self employed | … | |
| 4 | 45 | 1495 | 09-12-77 | Self employed | … | |

| | SEC.SANCTIONED.AMOUNT | SEC.DISBURSED.AMOUNT | PRIMARY.INSTAL.AMT | \ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1991 | |
| 2 | 0 | 0 | 0 | |
| 3 | 0 | 0 | 31 | |
| 4 | 0 | 0 | 0 | |

| | SEC.INSTAL.AMT | NEW.ACCTS.IN.LAST.SIX.MONTHS | \ |
|---|---|---|---|
| 0 | 0 | 0 | |
| 1 | 0 | 0 | |
| 2 | 0 | 0 | |

```
3                0                                0
4                0                                0

   DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS  AVERAGE.ACCT.AGE  \
0                                    0         0yrs 0mon
1                                    1        1yrs 11mon
2                                    0         0yrs 0mon
3                                    0         0yrs 8mon
4                                    0         0yrs 0mon

   CREDIT.HISTORY.LENGTH  NO.OF_INQUIRIES  loan_default
0             0yrs 0mon                0             0
1            1yrs 11mon                0             1
2             0yrs 0mon                0             0
3             1yrs 3mon                1             1
4             0yrs 0mon                1             1

[5 rows x 41 columns]
```

[4]:
```
# tail will give the glimpse of bottom 5 rows in the data
df.tail()
```

[4]:
```
        UniqueID  disbursed_amount  asset_cost    ltv  branch_id  supplier_id  \
233149    626432             63213      105405  60.72         34        20700
233150    606141             73651      100600  74.95         34        23775
233151    613658             33484       71212  48.45         77        22186
233152    548084             34259       73286  49.10         77        22186
233153    630213             75751      116009  66.81         77        22186

        manufacturer_id  Current_pincode_ID Date.of.Birth Employment.Type  \
233149               48                1050      01-08-88        Salaried
233150               51                 990      05-12-88   Self employed
233151               86                2299      01-06-76        Salaried
233152               86                2299      26-03-94        Salaried
233153               86                2299      18-02-84        Salaried

        … SEC.SANCTIONED.AMOUNT  SEC.DISBURSED.AMOUNT  PRIMARY.INSTAL.AMT  \
233149  …                     0                     0                4084
233150  …                     0                     0                1565
233151  …                     0                     0                   0
233152  …                     0                     0                   0
233153  …                     0                     0                   0

        SEC.INSTAL.AMT  NEW.ACCTS.IN.LAST.SIX.MONTHS  \
233149               0                             0
233150               0                             0
233151               0                             0
```

```
233152                0                                0
233153                0                                0

        DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS  AVERAGE.ACCT.AGE  \
233149                                    0         1yrs 9mon
233150                                    0         0yrs 6mon
233151                                    0         0yrs 0mon
233152                                    0         0yrs 0mon
233153                                    0         0yrs 0mon

        CREDIT.HISTORY.LENGTH  NO.OF_INQUIRIES  loan_default
233149             3yrs 3mon                0             0
233150             0yrs 6mon                0             0
233151             0yrs 0mon                0             0
233152             0yrs 0mon                0             0
233153             0yrs 0mon                0             0

[5 rows x 41 columns]
```

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

[5]:
```
(233154, 41)
```

This shows that we have 233154 records in our data with 41 attributes required for loan default prediction

[6]:
```
df.columns
```

[6]:
```
Index(['UniqueID', 'disbursed_amount', 'asset_cost', 'ltv', 'branch_id',
       'supplier_id', 'manufacturer_id', 'Current_pincode_ID', 'Date.of.Birth',
       'Employment.Type', 'DisbursalDate', 'State_ID', 'Employee_code_ID',
       'MobileNo_Avl_Flag', 'Aadhar_flag', 'PAN_flag', 'VoterID_flag',
       'Driving_flag', 'Passport_flag', 'PERFORM_CNS.SCORE',
       'PERFORM_CNS.SCORE.DESCRIPTION', 'PRI.NO.OF.ACCTS', 'PRI.ACTIVE.ACCTS',
       'PRI.OVERDUE.ACCTS', 'PRI.CURRENT.BALANCE', 'PRI.SANCTIONED.AMOUNT',
       'PRI.DISBURSED.AMOUNT', 'SEC.NO.OF.ACCTS', 'SEC.ACTIVE.ACCTS',
       'SEC.OVERDUE.ACCTS', 'SEC.CURRENT.BALANCE', 'SEC.SANCTIONED.AMOUNT',
       'SEC.DISBURSED.AMOUNT', 'PRIMARY.INSTAL.AMT', 'SEC.INSTAL.AMT',
       'NEW.ACCTS.IN.LAST.SIX.MONTHS', 'DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS',
       'AVERAGE.ACCT.AGE', 'CREDIT.HISTORY.LENGTH', 'NO.OF_INQUIRIES',
       'loan_default'],
      dtype='object')
```

[7]:
```
df.describe().T
```

[7]:
```
                                  count           mean           std  \
UniqueID                     233154.0  535917.573376  6.831569e+04
disbursed_amount             233154.0   54356.993528  1.297131e+04
```

| | | | |
|---|---|---|---|
| asset_cost | 233154.0 | 75865.068144 | 1.894478e+04 |
| ltv | 233154.0 | 74.746530 | 1.145664e+01 |
| branch_id | 233154.0 | 72.936094 | 6.983499e+01 |
| supplier_id | 233154.0 | 19638.635035 | 3.491950e+03 |
| manufacturer_id | 233154.0 | 69.028054 | 2.214130e+01 |
| Current_pincode_ID | 233154.0 | 3396.880247 | 2.238148e+03 |
| State_ID | 233154.0 | 7.262243 | 4.482230e+00 |
| Employee_code_ID | 233154.0 | 1549.477148 | 9.752613e+02 |
| MobileNo_Avl_Flag | 233154.0 | 1.000000 | 0.000000e+00 |
| Aadhar_flag | 233154.0 | 0.840320 | 3.663097e-01 |
| PAN_flag | 233154.0 | 0.075577 | 2.643201e-01 |
| VoterID_flag | 233154.0 | 0.144943 | 3.520439e-01 |
| Driving_flag | 233154.0 | 0.023242 | 1.506720e-01 |
| Passport_flag | 233154.0 | 0.002127 | 4.607421e-02 |
| PERFORM_CNS.SCORE | 233154.0 | 289.462994 | 3.383748e+02 |
| PRI.NO.OF.ACCTS | 233154.0 | 2.440636 | 5.217233e+00 |
| PRI.ACTIVE.ACCTS | 233154.0 | 1.039896 | 1.941496e+00 |
| PRI.OVERDUE.ACCTS | 233154.0 | 0.156549 | 5.487867e-01 |
| PRI.CURRENT.BALANCE | 233154.0 | 165900.076936 | 9.422736e+05 |
| PRI.SANCTIONED.AMOUNT | 233154.0 | 218503.855323 | 2.374794e+06 |
| PRI.DISBURSED.AMOUNT | 233154.0 | 218065.898655 | 2.377744e+06 |
| SEC.NO.OF.ACCTS | 233154.0 | 0.059081 | 6.267946e-01 |
| SEC.ACTIVE.ACCTS | 233154.0 | 0.027703 | 3.160566e-01 |
| SEC.OVERDUE.ACCTS | 233154.0 | 0.007244 | 1.110789e-01 |
| SEC.CURRENT.BALANCE | 233154.0 | 5427.792819 | 1.702370e+05 |
| SEC.SANCTIONED.AMOUNT | 233154.0 | 7295.923347 | 1.831560e+05 |
| SEC.DISBURSED.AMOUNT | 233154.0 | 7179.997873 | 1.825925e+05 |
| PRIMARY.INSTAL.AMT | 233154.0 | 13105.481720 | 1.513679e+05 |
| SEC.INSTAL.AMT | 233154.0 | 323.268449 | 1.555369e+04 |
| NEW.ACCTS.IN.LAST.SIX.MONTHS | 233154.0 | 0.381833 | 9.551067e-01 |
| DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS | 233154.0 | 0.097481 | 3.844390e-01 |
| NO.OF_INQUIRIES | 233154.0 | 0.206615 | 7.064977e-01 |
| loan_default | 233154.0 | 0.217071 | 4.122523e-01 |

| | min | 25% | 50% | \ |
|---|---|---|---|---|
| UniqueID | 417428.00 | 476786.25 | 535978.5 | |
| disbursed_amount | 13320.00 | 47145.00 | 53803.0 | |
| asset_cost | 37000.00 | 65717.00 | 70946.0 | |
| ltv | 10.03 | 68.88 | 76.8 | |
| branch_id | 1.00 | 14.00 | 61.0 | |
| supplier_id | 10524.00 | 16535.00 | 20333.0 | |
| manufacturer_id | 45.00 | 48.00 | 86.0 | |
| Current_pincode_ID | 1.00 | 1511.00 | 2970.0 | |
| State_ID | 1.00 | 4.00 | 6.0 | |
| Employee_code_ID | 1.00 | 713.00 | 1451.0 | |
| MobileNo_Avl_Flag | 1.00 | 1.00 | 1.0 | |
| Aadhar_flag | 0.00 | 1.00 | 1.0 | |

| | | | |
|---|---|---|---|
| PAN_flag | 0.00 | 0.00 | 0.0 |
| VoterID_flag | 0.00 | 0.00 | 0.0 |
| Driving_flag | 0.00 | 0.00 | 0.0 |
| Passport_flag | 0.00 | 0.00 | 0.0 |
| PERFORM_CNS.SCORE | 0.00 | 0.00 | 0.0 |
| PRI.NO.OF.ACCTS | 0.00 | 0.00 | 0.0 |
| PRI.ACTIVE.ACCTS | 0.00 | 0.00 | 0.0 |
| PRI.OVERDUE.ACCTS | 0.00 | 0.00 | 0.0 |
| PRI.CURRENT.BALANCE | -6678296.00 | 0.00 | 0.0 |
| PRI.SANCTIONED.AMOUNT | 0.00 | 0.00 | 0.0 |
| PRI.DISBURSED.AMOUNT | 0.00 | 0.00 | 0.0 |
| SEC.NO.OF.ACCTS | 0.00 | 0.00 | 0.0 |
| SEC.ACTIVE.ACCTS | 0.00 | 0.00 | 0.0 |
| SEC.OVERDUE.ACCTS | 0.00 | 0.00 | 0.0 |
| SEC.CURRENT.BALANCE | -574647.00 | 0.00 | 0.0 |
| SEC.SANCTIONED.AMOUNT | 0.00 | 0.00 | 0.0 |
| SEC.DISBURSED.AMOUNT | 0.00 | 0.00 | 0.0 |
| PRIMARY.INSTAL.AMT | 0.00 | 0.00 | 0.0 |
| SEC.INSTAL.AMT | 0.00 | 0.00 | 0.0 |
| NEW.ACCTS.IN.LAST.SIX.MONTHS | 0.00 | 0.00 | 0.0 |
| DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS | 0.00 | 0.00 | 0.0 |
| NO.OF_INQUIRIES | 0.00 | 0.00 | 0.0 |
| loan_default | 0.00 | 0.00 | 0.0 |

| | 75% | max |
|---|---|---|
| UniqueID | 595039.75 | 6.710840e+05 |
| disbursed_amount | 60413.00 | 9.905720e+05 |
| asset_cost | 79201.75 | 1.628992e+06 |
| ltv | 83.67 | 9.500000e+01 |
| branch_id | 130.00 | 2.610000e+02 |
| supplier_id | 23000.00 | 2.480300e+04 |
| manufacturer_id | 86.00 | 1.560000e+02 |
| Current_pincode_ID | 5677.00 | 7.345000e+03 |
| State_ID | 10.00 | 2.200000e+01 |
| Employee_code_ID | 2362.00 | 3.795000e+03 |
| MobileNo_Avl_Flag | 1.00 | 1.000000e+00 |
| Aadhar_flag | 1.00 | 1.000000e+00 |
| PAN_flag | 0.00 | 1.000000e+00 |
| VoterID_flag | 0.00 | 1.000000e+00 |
| Driving_flag | 0.00 | 1.000000e+00 |
| Passport_flag | 0.00 | 1.000000e+00 |
| PERFORM_CNS.SCORE | 678.00 | 8.900000e+02 |
| PRI.NO.OF.ACCTS | 3.00 | 4.530000e+02 |
| PRI.ACTIVE.ACCTS | 1.00 | 1.440000e+02 |
| PRI.OVERDUE.ACCTS | 0.00 | 2.500000e+01 |
| PRI.CURRENT.BALANCE | 35006.50 | 9.652492e+07 |
| PRI.SANCTIONED.AMOUNT | 62500.00 | 1.000000e+09 |

```
PRI.DISBURSED.AMOUNT                  60800.00  1.000000e+09
SEC.NO.OF.ACCTS                           0.00  5.200000e+01
SEC.ACTIVE.ACCTS                          0.00  3.600000e+01
SEC.OVERDUE.ACCTS                         0.00  8.000000e+00
SEC.CURRENT.BALANCE                       0.00  3.603285e+07
SEC.SANCTIONED.AMOUNT                     0.00  3.000000e+07
SEC.DISBURSED.AMOUNT                      0.00  3.000000e+07
PRIMARY.INSTAL.AMT                     1999.00  2.564281e+07
SEC.INSTAL.AMT                            0.00  4.170901e+06
NEW.ACCTS.IN.LAST.SIX.MONTHS              0.00  3.500000e+01
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS       0.00  2.000000e+01
NO.OF_INQUIRIES                           0.00  3.600000e+01
loan_default                              0.00  1.000000e+00
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233154 entries, 0 to 233153
Data columns (total 41 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   UniqueID                             233154 non-null  int64
 1   disbursed_amount                     233154 non-null  int64
 2   asset_cost                           233154 non-null  int64
 3   ltv                                  233154 non-null  float64
 4   branch_id                            233154 non-null  int64
 5   supplier_id                          233154 non-null  int64
 6   manufacturer_id                      233154 non-null  int64
 7   Current_pincode_ID                   233154 non-null  int64
 8   Date.of.Birth                        233154 non-null  object
 9   Employment.Type                      225493 non-null  object
 10  DisbursalDate                        233154 non-null  object
 11  State_ID                             233154 non-null  int64
 12  Employee_code_ID                     233154 non-null  int64
 13  MobileNo_Avl_Flag                    233154 non-null  int64
 14  Aadhar_flag                          233154 non-null  int64
 15  PAN_flag                             233154 non-null  int64
 16  VoterID_flag                         233154 non-null  int64
 17  Driving_flag                         233154 non-null  int64
 18  Passport_flag                        233154 non-null  int64
 19  PERFORM_CNS.SCORE                    233154 non-null  int64
 20  PERFORM_CNS.SCORE.DESCRIPTION        233154 non-null  object
 21  PRI.NO.OF.ACCTS                      233154 non-null  int64
 22  PRI.ACTIVE.ACCTS                     233154 non-null  int64
 23  PRI.OVERDUE.ACCTS                    233154 non-null  int64
 24  PRI.CURRENT.BALANCE                  233154 non-null  int64
 25  PRI.SANCTIONED.AMOUNT                233154 non-null  int64
 26  PRI.DISBURSED.AMOUNT                 233154 non-null  int64
```

```
 27  SEC.NO.OF.ACCTS                       233154 non-null  int64
 28  SEC.ACTIVE.ACCTS                      233154 non-null  int64
 29  SEC.OVERDUE.ACCTS                     233154 non-null  int64
 30  SEC.CURRENT.BALANCE                   233154 non-null  int64
 31  SEC.SANCTIONED.AMOUNT                 233154 non-null  int64
 32  SEC.DISBURSED.AMOUNT                  233154 non-null  int64
 33  PRIMARY.INSTAL.AMT                    233154 non-null  int64
 34  SEC.INSTAL.AMT                        233154 non-null  int64
 35  NEW.ACCTS.IN.LAST.SIX.MONTHS          233154 non-null  int64
 36  DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS   233154 non-null  int64
 37  AVERAGE.ACCT.AGE                      233154 non-null  object
 38  CREDIT.HISTORY.LENGTH                 233154 non-null  object
 39  NO.OF_INQUIRIES                       233154 non-null  int64
 40  loan_default                          233154 non-null  int64
dtypes: float64(1), int64(34), object(6)
memory usage: 72.9+ MB
```

[9]:
```
df.isnull().sum()
```

[9]:
```
UniqueID                              0
disbursed_amount                      0
asset_cost                            0
ltv                                   0
branch_id                             0
supplier_id                           0
manufacturer_id                       0
Current_pincode_ID                    0
Date.of.Birth                         0
Employment.Type                    7661
DisbursalDate                         0
State_ID                              0
Employee_code_ID                      0
MobileNo_Avl_Flag                     0
Aadhar_flag                           0
PAN_flag                              0
VoterID_flag                          0
Driving_flag                          0
Passport_flag                         0
PERFORM_CNS.SCORE                     0
PERFORM_CNS.SCORE.DESCRIPTION         0
PRI.NO.OF.ACCTS                       0
PRI.ACTIVE.ACCTS                      0
PRI.OVERDUE.ACCTS                     0
PRI.CURRENT.BALANCE                   0
PRI.SANCTIONED.AMOUNT                 0
PRI.DISBURSED.AMOUNT                  0
SEC.NO.OF.ACCTS                       0
```

```
SEC.ACTIVE.ACCTS                        0
SEC.OVERDUE.ACCTS                       0
SEC.CURRENT.BALANCE                     0
SEC.SANCTIONED.AMOUNT                   0
SEC.DISBURSED.AMOUNT                    0
PRIMARY.INSTAL.AMT                      0
SEC.INSTAL.AMT                          0
NEW.ACCTS.IN.LAST.SIX.MONTHS            0
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS     0
AVERAGE.ACCT.AGE                        0
CREDIT.HISTORY.LENGTH                   0
NO.OF_INQUIRIES                         0
loan_default                            0
dtype: int64
```

[10]:
```
# Converting DisbursalDate and Date.of.Birth columns to date format
df['DisbursalDate'] = pd.to_datetime(df['DisbursalDate'],format='%d-%m-%y',␣
 ↪errors='coerce')
df['Date.of.Birth'] = pd.to_datetime(df['Date.of.Birth'],format='%d-%m-%y',␣
 ↪errors='coerce')

#Considering blank Employment.Type as unemployed
df['Employment.Type'] = df['Employment.Type'].fillna('unemployed')
df['Employment.Type'].head()
```

[10]:
```
0         Salaried
1    Self employed
2    Self employed
3    Self employed
4    Self employed
Name: Employment.Type, dtype: object
```

[12]:
```
df.to_csv("cleaned_data_train.csv", index=False)
```

### 0.0.1 Interactive dashboard for analysing risks

[30]:
```
data = pd.read_csv("cleaned_data_train.csv")

app = Dash(__name__)

app.layout = html.Div(
    style={
        'backgroundColor': 'white',
        'padding': '20px',
        'fontFamily': 'Arial, sans-serif'
    },
    children=[
```

```
        html.H1("Dashboard for Analysing risk", style={'textAlign': 'center',␣
↪'color': '#333'}),

        html.Div([
            html.Label("Filter by State ID:", style={'color': '#555'}),
            dcc.Dropdown(
                id='state-dropdown',
                options=[{'label': state, 'value': state} for state in␣
↪data['State_ID'].dropna().unique()],
                value=None,
                multi=True,
                placeholder="Select State ID",
                style={'backgroundColor': 'white', 'color': '#333'}
            ),
            html.Label("Filter by Current Pincode ID:", style={'color': '#555',␣
↪'marginTop': '10px'}),
            dcc.Dropdown(
                id='Current_pincode_ID-dropdown',
                options=[],
                value=None,
                multi=True,
                placeholder="Select Current Pincode ID",
                style={'backgroundColor': 'white', 'color': '#333'}
            ),
            html.Label("Filter by Branch ID:", style={'color': '#555',␣
↪'marginTop': '10px'}),
            dcc.Dropdown(
                id='branch-dropdown',
                options=[],
                value=None,
                multi=True,
                placeholder="Select Branch ID",
                style={'backgroundColor': 'white', 'color': '#333'}
            ),
        ], style={'marginBottom': '20px'}),

        html.Div([
            html.Label("Select Columns for X-axis:", style={'color': '#555'}),
            dcc.Dropdown(
                id='x-axis-dropdown',
                options=[{'label': col, 'value': col} for col in data.columns␣
↪if data[col].dtype in ['int64', 'float64']],
                value=[],
                multi=True,
                placeholder="Select X-axis columns",
                style={'backgroundColor': 'white', 'color': '#333'}
            ),
```

```
        ], style={'marginBottom': '20px'}),

        html.Button("Apply", id="apply-button", n_clicks=0, style={
            'backgroundColor': '#007BFF',
            'color': 'white',
            'border': 'none',
            'padding': '10px 20px',
            'cursor': 'pointer'
        }),

        dcc.Graph(id='bar-chart', style={'backgroundColor': 'white'}),
    ]
)

# Callback to update Current_pincode_ID options based on selected State_ID
@app.callback(
    Output('Current_pincode_ID-dropdown', 'options'),
    [Input('state-dropdown', 'value')]
)
def update_pincode_options(selected_states):
    if selected_states:
        filtered_data = data[data['State_ID'].isin(selected_states)]
        pincodes = filtered_data['Current_pincode_ID'].dropna().unique()
        return [{'label': pincode, 'value': pincode} for pincode in pincodes]
    return []

# Callback to update Branch ID options based on selected State_ID and␣
 ↪Current_pincode_ID
@app.callback(
    Output('branch-dropdown', 'options'),
    [Input('state-dropdown', 'value'),
     Input('Current_pincode_ID-dropdown', 'value')]
)
def update_branch_options(selected_states, selected_pincodes):
    filtered_data = data.copy()
    if selected_states:
        filtered_data = filtered_data[filtered_data['State_ID'].
 ↪isin(selected_states)]
    if selected_pincodes:
        filtered_data = filtered_data[filtered_data['Current_pincode_ID'].
 ↪isin(selected_pincodes)]
    branches = filtered_data['branch_id'].dropna().unique()
    return [{'label': branch, 'value': branch} for branch in branches]

# Callback to update the graph based on filters and x-axis selection
@app.callback(
    Output('bar-chart', 'figure'),
```

```
    [
        Input('apply-button', 'n_clicks'),
    ],
    [
        Input('x-axis-dropdown', 'value'),
        Input('state-dropdown', 'value'),
        Input('Current_pincode_ID-dropdown', 'value'),
        Input('branch-dropdown', 'value'),
    ]
)
def update_graph(n_clicks, x_cols, states, pincodes, branches):
    # Filter the data based on dropdowns
    filtered_data = data.copy()
    if states:
        filtered_data = filtered_data[filtered_data['State_ID'].isin(states)]
    if pincodes:
        filtered_data = filtered_data[filtered_data['Current_pincode_ID'].
 ↪isin(pincodes)]
    if branches:
        filtered_data = filtered_data[filtered_data['branch_id'].isin(branches)]

    if x_cols:
        filtered_data['Employment.Type'] = filtered_data['Employment.Type'].
 ↪fillna('Unknown')
        aggregated_data = filtered_data.groupby(['Employment.Type'])[x_cols].
 ↪sum().reset_index()

        # Create a stacked bar graph with custom colors
        fig = go.Figure()
        colors = ['#00CC96', '#AB63FA', '#FFA15A', '#19D3F3'] #['#636EFA',␣
 ↪'#EF553B',]

        for i, col in enumerate(x_cols):
            fig.add_trace(
                go.Bar(
                        x=aggregated_data['Employment.Type'],
                        y=aggregated_data[col],
                        name=col,
                        marker_color=colors[i % len(colors)],
                        hovertemplate="Employment Type: %{x}<br>" +
                                      "Amount: $%{y:,.0f}<br>",
                        hoverlabel=dict(
                            bgcolor="white",
                            font_size=12,
                            font_family="Arial"
                        )
                    )
```

```
            )

        fig.update_layout(
            barmode='stack',
            title="Total Disbursed Amount vs Total Asset Cost",
            xaxis_title="Employment Type",
            yaxis_title="Total Amount in USD",
            plot_bgcolor="white",
            paper_bgcolor="white"
        )
    else:
        fig = go.Figure()
        fig.update_layout(title="Please select columns for X-axis")

    return fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=False)
```

```
<IPython.lib.display.IFrame at 0x36417bb90>
```

This interactive dashboard is used to streamline the analysis of financial data by incorporating dynamic filters such as State ID, Pincode ID, and Branch ID, allowing users to customize their exploration based on specific criteria. This helps to identify risks through patterns and insights at various granular levels.

Note: When opened if dashboard isn't appear run the code by importing the files. We also attached screenshots in the submission please see them to know how dashboard will look like

### 0.0.2 Interactive dashboard to analyse patterns and trends

[31]:
```
# Layout of the dashboard
app.layout = html.Div(
    style={
        'backgroundColor': 'white',
        'padding': '20px',
        'fontFamily': 'Arial, sans-serif'
    },
    children=[
        html.H1("Interactive Dashboard", style={'textAlign': 'center', 'color':␣
↪'#333'}),

        # Filters for specific fields
        html.Div([
            html.Label("Filter by State ID:", style={'color': '#555'}),
            dcc.Dropdown(
```

```
                id='state-dropdown',
                options=[{'label': state, 'value': state} for state in␣
↪data['State_ID'].dropna().unique()],
                value=None,
                multi=True,
                placeholder="Select State ID",
                style={'backgroundColor': 'white', 'color': '#333'}
            ),
            html.Label("Filter by Current Pincode ID:", style={'color': '#555',␣
↪'marginTop': '10px'}),
            dcc.Dropdown(
                id='Current_pincode_ID-dropdown',
                options=[{'label': pin, 'value': pin} for pin in␣
↪data['Current_pincode_ID'].dropna().unique()],
                value=None,
                multi=True,
                placeholder="Select Current Pincode ID",
                style={'backgroundColor': 'white', 'color': '#333'}
            ),
            html.Label("Filter by Branch ID:", style={'color': '#555',␣
↪'marginTop': '10px'}),
            dcc.Dropdown(
                id='branch-dropdown',
                options=[{'label': branch, 'value': branch} for branch in␣
↪data['branch_id'].dropna().unique()],
                value=None,
                multi=True,
                placeholder="Select Branch ID",
                style={'backgroundColor': 'white', 'color': '#333'}
            ),
        ], style={'marginBottom': '20px'}),

        # Graph area
        html.Div([
            dcc.Graph(id='plot-inquiries', style={'backgroundColor': 'white',␣
↪'marginBottom': '20px'}),
            dcc.Graph(id='plot-scores', style={'backgroundColor': 'white',␣
↪'marginBottom': '20px'}),
            dcc.Graph(id='plot-employment-type', style={'backgroundColor':␣
↪'white'})
        ])
    ]
)

# Callback to update the graphs based on filters
@app.callback(
```

```
    [
        Output('plot-inquiries', 'figure'),
        Output('plot-scores', 'figure'),
        Output('plot-employment-type', 'figure'),
    ],
    [
        Input('state-dropdown', 'value'),
        Input('Current_pincode_ID-dropdown', 'value'),
        Input('branch-dropdown', 'value'),
    ]
)
def update_graphs(states, pincodes, branches):
    # Filter the data based on dropdowns
    filtered_data = data.copy()
    if states:
        filtered_data = filtered_data[filtered_data['State_ID'].isin(states)]
    if pincodes:
        filtered_data = filtered_data[filtered_data['Current_pincode_ID'].
 ↪isin(pincodes)]
    if branches:
        filtered_data = filtered_data[filtered_data['branch_id'].isin(branches)]

    # Plot 1: Radar Chart of NO.OF_INQUIRIES vs PERFORM_CNS.SCORE.DESCRIPTION
    inquiries_data = filtered_data.groupby('PERFORM_CNS.SCORE.DESCRIPTION')['NO.
 ↪OF_INQUIRIES'].sum().reset_index()
    radar_fig = go.Figure()
    radar_fig.add_trace(go.Scatterpolar(
        r=inquiries_data['NO.OF_INQUIRIES'],
        theta=inquiries_data['PERFORM_CNS.SCORE.DESCRIPTION'],
        fill='toself',
        name="NO.OF_INQUIRIES"
    ))
    radar_fig.update_layout(
        polar=dict(radialaxis=dict(visible=True)),
        title="Radar Chart of NO.OF_INQUIRIES by PERFORM_CNS.SCORE.DESCRIPTION",
        showlegend=True,
        plot_bgcolor="white",
        paper_bgcolor="white"
    )

    # Plot 2: Average of PERFORM_CNS.SCORE vs PERFORM_CNS.SCORE.DESCRIPTION␣
 ↪(Line Chart)
    scores_data = filtered_data.groupby('PERFORM_CNS.SCORE.
 ↪DESCRIPTION')['PERFORM_CNS.SCORE'].mean().reset_index()
    scores_fig = px.line(
        scores_data,
        x='PERFORM_CNS.SCORE.DESCRIPTION',
```

```
        y='PERFORM_CNS.SCORE',
        title='Average of PERFORM_CNS.SCORE by PERFORM_CNS.SCORE.DESCRIPTION'
    )

    # Plot 3: Pie Chart of Employment.Type counts
    employment_data = filtered_data['Employment.Type'].value_counts().
 ↪reset_index()
    employment_data.columns = ['Employment.Type', 'Count']
    donut_fig = go.Figure(data=[go.Pie(
        labels=employment_data['Employment.Type'],
        values=employment_data['Count'],
        marker=dict(colors=['yellow', 'seablue', 'lightblue']),  # Updated␣
 ↪colors
        hole=0.4  # Create a donut chart
    )])
    donut_fig.update_layout(
        title="Distribution of Employment.Type",
        plot_bgcolor="white",
        paper_bgcolor="white"
    )

    return radar_fig, scores_fig, donut_fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=False)
```

```
<IPython.lib.display.IFrame at 0x364a4c950>
```

Note: When opened if dashboard isn't appear run the code by importing the files. We also attached screenshots in the submission please see them to know how dashboard will look like

[14]:
```
data['Date.of.Birth'] = pd.to_datetime(data['Date.of.Birth'], errors='coerce')
data['Year of DOB'] = data['Date.of.Birth'].dt.year.apply(
    lambda year: year if year <= 2000 else year - 100
)

aggregated_data = data.groupby('Year of DOB').agg(
    Sum_Disbursed_Amount=('disbursed_amount', 'sum'),
    Sum_Asset_Cost=('asset_cost', 'sum')
).reset_index()

# Create an area plot using Plotly Graph Objects
fig = go.Figure()

# Add the "Sum of Disbursed Amount" trace
fig.add_trace(go.Scatter(
    x=aggregated_data['Year of DOB'],
```

```
    y=aggregated_data['Sum_Disbursed_Amount'],
    mode='lines',
    fill='tozeroy',
    name='Sum of Disbursed Amount',
    hovertemplate="Year of DOB: %{x}<br>Disbursed Amount: %{y}<extra></extra>"
))

# Add the "Sum of Asset Cost" trace
fig.add_trace(go.Scatter(
    x=aggregated_data['Year of DOB'],
    y=aggregated_data['Sum_Asset_Cost'],
    mode='lines',
    fill='tonexty',
    name='Sum of Asset Cost',
    hovertemplate="Year of DOB: %{x}<br>Asset Cost: %{y}<extra></extra>"
))

# Update layout for better appearance
fig.update_layout(
    title='Year of DOB vs Sum of Disbursed Amount and Asset Cost',
    xaxis_title='Year of DOB',
    yaxis_title='Sum (in Millions)',
    legend_title='Metrics',
    template='plotly_white'
)

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

1. Area Graph - We used Interactive area graph to explore the relationship between Year of Date of Birth and the sum of disbursed amount and asset cost. As it represents the trend over time and providing clear visual representation of the growth and fluctuations in both disbursed amount and asset cost over the years. We feel its better than raditional line plot
2. Radar Chart - The radar chart was employed to compare the number of inquiries encountered for each CNS score description after applying specific filters (State ID, Current Pincode, and Branch ID). The radar chart effectively represents multiple variables (CNS score descriptions) on a single plot, allowing for easy comparison and identification of trends.
3. Donut Graph - Donut graph was deployed along with interactive filters to analyse how employment type is varying geographically.
4. Line Graph - Line graph was used to analyse Average CNS score for each cns description. This will help to easily undertand the trend of cns score also it provides the tooltip when you just hover on it it will show the description type nad average score

Note: When opened if dashboard isn't appear run the code by importing the files. We also attached screenshots in the submission please see them to know how dashboard will look like

[42]:
```
data = pd.read_csv("train.csv")
```

[43]:
```
data.fillna(0, inplace=True)
```

[44]:
```
from datetime import datetime
def calculate_age(dob, reference_date):
    dob = datetime.strptime(dob, '%d-%m-%y')
    age = reference_date.year - dob.year - ((reference_date.month,␣
 ↪reference_date.day) < (dob.month, dob.day))
    return age

reference_date = datetime(2024, 12, 5)
data['Age'] = data['Date.of.Birth'].apply(lambda dob: calculate_age(dob,␣
 ↪reference_date))

data.drop(columns=['Date.of.Birth'], inplace=True)
```

[45]:
```
def convert_to_months(value):
    try:
        years, months = map(int, value.replace('yrs', '').replace('mon', '').
 ↪split())
        return years * 12 + months
    except:
        return 0

data['AVERAGE.ACCT.AGE'] = data['AVERAGE.ACCT.AGE'].apply(convert_to_months)
data['CREDIT.HISTORY.LENGTH'] = data['CREDIT.HISTORY.LENGTH'].
 ↪apply(convert_to_months)
```

[46]:
```
state_id_to_name = {
    1: "Andhra Pradesh", 2: "Arunachal Pradesh", 3: "Assam", 4: "Bihar", 5:␣
 ↪"Chhattisgarh",
    6: "Goa", 7: "Gujarat", 8: "Haryana", 9: "Himachal Pradesh", 10:␣
 ↪"Jharkhand",
    11: "Karnataka", 12: "Kerala", 13: "Madhya Pradesh", 14: "Maharashtra", 15:␣
 ↪"Manipur",
    16: "Meghalaya", 17: "Mizoram", 18: "Nagaland", 19: "Odisha", 20: "Punjab",
    21: "Rajasthan", 22: "Sikkim"
}

data['State_Name'] = data['State_ID'].map(state_id_to_name)

data = data[data['State_Name'].notna()]
```

[47]:
```
statewise_data = data.groupby('State_Name').agg(
    Avg_Disbursed_Amount=('disbursed_amount', 'mean'),
    Avg_Asset_Cost=('asset_cost', 'mean'),
```

```
    Loan_Default_Count=('loan_default', 'sum')
).reset_index()
```

[48]:
```
import json
geojson_path = 'india_state.geojson'
with open(geojson_path) as file:
    geojsonData = json.load(file)

geojson_state_names = [feature['properties']['NAME_1'] for feature in␣
 ↪geojsonData['features']]
statewise_data = statewise_data.set_index('State_Name').
 ↪reindex(geojson_state_names).reset_index()

statewise_data.fillna(0, inplace=True)
```

[49]:
```
import plotly.express as px
from plotly.subplots import make_subplots

fig_disbursed = px.choropleth(
    statewise_data,
    geojson=geojsonData,
    locations="State_Name",
    featureidkey="properties.NAME_1",
    color="Avg_Disbursed_Amount",
    color_continuous_scale="Oranges",
    title="Average Disbursed Amount per State"
)

fig_asset_cost = px.choropleth(
    statewise_data,
    geojson=geojsonData,
    locations="State_Name",
    featureidkey="properties.NAME_1",
    color="Avg_Asset_Cost",
    color_continuous_scale="Blues",
    title="Average Asset Cost per State"
)

fig_loan_default = px.choropleth(
    statewise_data,
    geojson=geojsonData,
    locations="State_Name",
    featureidkey="properties.NAME_1",
    color="Loan_Default_Count",
    color_continuous_scale="Reds",
    title="Loan Default Count per State"
)
```

```
fig = make_subplots(rows=1, cols=1, specs=[[{"type": "choropleth"}]])
fig.add_trace(fig_disbursed.data[0])
fig.add_trace(fig_asset_cost.data[0])
fig.add_trace(fig_loan_default.data[0])

fig.update_layout(
    updatemenus=[
        {
            "buttons": [
                {"label": "Average Disbursed Amount", "method": "update",␣
 ↪"args": [{"visible": [True, False, False]}]},
                {"label": "Average Asset Cost", "method": "update", "args":␣
 ↪[{"visible": [False, True, False]}]},
                {"label": "Loan Default Count", "method": "update", "args":␣
 ↪[{"visible": [False, False, True]}]},
            ],
            "direction": "left",
            "pad": {"r": 10, "t": 10},
            "showactive": True,
            "type": "buttons",
            "x": 0.1,
            "xanchor": "left",
            "y": 1.2,
            "yanchor": "top",
        },
    ],
    title_text="Choropleth Map: Disbursed Amount, Asset Cost, and Loan␣
 ↪Defaults",
    geo=dict(
        showframe=False,
        showcoastlines=False,
        projection_type="mercator",
        scope="asia",
        center={"lat": 20.5937, "lon": 78.9629},
        fitbounds="locations",
    ),
)

fig.show()
```

The choropleth map code creates a geographical visualization to display key loan metrics at the state level focusing average disbursed amount, average asset cost, and loan default count. Each state is color-coded based on the metric being visualized, with darker shades representing higher values. The interactive map allows users to switch between metrics using a dropdown menu.

[50]:
```
columns_of_interest = [
    'disbursed_amount', 'asset_cost', 'ltv', 'PERFORM_CNS.SCORE',
    'PRI.NO.OF.ACCTS', 'PRI.ACTIVE.ACCTS', 'PRI.OVERDUE.ACCTS',
    'PRI.CURRENT.BALANCE', 'PRI.SANCTIONED.AMOUNT', 'PRI.DISBURSED.AMOUNT',
    'PRIMARY.INSTAL.AMT', 'NEW.ACCTS.IN.LAST.SIX.MONTHS',
    'DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS', 'AVERAGE.ACCT.AGE',
    'CREDIT.HISTORY.LENGTH', 'NO.OF_INQUIRIES', 'loan_default'
]
correlation_data = data[columns_of_interest].dropna()
```

[51]:
```
import plotly.express as px

aggregated_data = (
    data.groupby(['State_Name', 'Employment.Type', 'loan_default'])
    .size()
    .reset_index(name='count')
)

aggregated_data['loan_default'] = aggregated_data['loan_default'].map({0: 'No␣
 ↪Default', 1: 'Default'})

fig = px.sunburst(
    aggregated_data,
    path=['State_Name', 'Employment.Type', 'loan_default'],
    values='count',
    color='loan_default',
    color_discrete_map={'No Default': 'green', 'Default': 'red'},
    title='Sunburst chart for Loan Default Counts by State and Employment Type'
)

fig.update_traces(
    textinfo='label+value+percent entry',
    insidetextorientation='radial'
)
fig.update_layout(
    margin=dict(t=50, l=0, r=0, b=0),
    title_font_size=20,
    font=dict(size=12)
)


fig.show()
```

The sunburst chart offers a hierarchical view of loan defaults, enabling a detailed breakdown by state, employment type. Each level of the chart represents a segment of the hierarchy, with proportions depicted visually for easy comparison.Drill down from state-level trends to specific borrower segments contributing to defaults.

[ ]: