#### **EXPERIMENT 1**

## Aim: Introduction to Data science and Data preparation using Pandas steps.

- Load data in Pandas.
- Description of the dataset.
- Drop columns that aren't useful.
- Drop rows with maximum missing values.
- Take care of missing data.
- Create dummy variables.
- Find out outliers (manually)
- Standardization and normalization of columns

Dataset: Financial Risk Assessment

Link: https://www.kaggle.com/datasets/preethamgouda/financial-risk

### Steps:

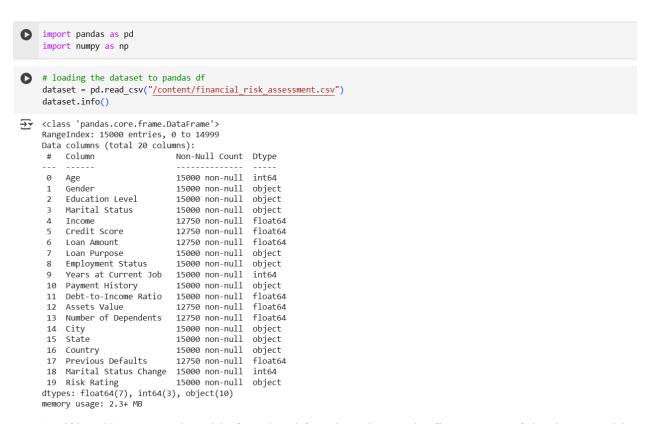
### 1) Loading data in Pandas and extracting information about the dataset.

To load a file onto python for analysis, we need to make use of the pandas library. It gives us functionalities to read a CSV (Comma Separated Values) file and perform various functions on it.

Commands: import pandas as pd (Importing the pandas library onto Google Colab Notebook) df = pd.read\_csv() (Mounts and reads the file in Python and assigns it to variable df for ease of use further)

(Note: Replace with the actual path of the file in "")

dataset.info(): This command gives all the information about the features (columns) of the dataset and the data type of each of these columns. It also gives a summary of all the values in the dataset.



2) df.head(): As mentioned before, head function give us the first 5 rows of the dataset. This allows for the user to get an overview on what values are being listed in the dataset.



3) dataset.shape(): returns the dimensions of the dataset as a tuple (rows, columns), helping to understand its size.

```
[ ] dataset.shape

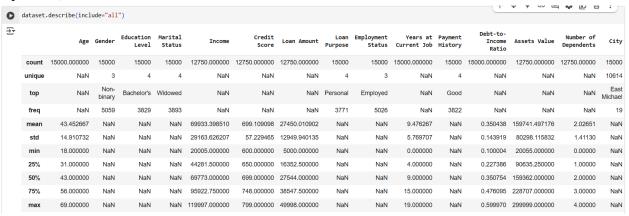
→ (15000, 20)
```

4) Describe the dataset

dataset.describe(): provides statistical summaries of numerical columns, including count, mean, standard deviation, min, max, and quartiles (25%, 50%, 75%).



If the parameter of include="all" is included { df.describe(include="all")}, this includes even the non numeric values and gives some more information on fields such as count of unique values, top value, etc.



## 5) Dropping the columns

dataset.drop() is used to remove specified rows or columns from the dataset.

- dataset.drop(columns=['column name']) → Drops a specific column.
- dataset.drop(index=[row index]) → Drops a specific row.

```
# dropping the columns that aren't useful
    cols = ['Marital Status', 'Marital Status Change', 'Loan Purpose', 'City', 'State']
    df = dataset.drop(cols, axis=1)
<class 'pandas.core.frame.DataFrame'>
    RangeIndex: 15000 entries, 0 to 14999
    Data columns (total 15 columns):
                               Non-Null Count
     # Column
                               15000 non-null
                                                int64
         Age
                               15000 non-null
                                                object
         Education Level
                               15000 non-null
         Income
                               12750 non-null
                                                float64
         Credit Score
                               12750 non-null
         Loan Amount
                               12750 non-null
                                                float64
         Employment Status
                               15000 non-null
                                                object
         Years at Current Job
                               15000 non-null
         Payment History
                                15000 non-null
         Debt-to-Income Ratio
                               15000 non-null
                                                float64
         Assets Value
                               12750 non-null
                                                float64
     10
         Number of Dependents
                               12750 non-null
         Country
                               15000 non-null
                                                object
         Previous Defaults
                               12750 non-null
                                                float64
         Risk Rating
                               15000 non-null
    dtypes: float64(7), int64(2), object(6)
    memory usage: 1.7+ MB
```

### Before Dropping:



As observed here, the columns of 'Marital Status', 'Marital Status Change', 'Loan Purpose', 'City', 'State' have been dropped.

## 6) Drop rows with maximum missing rows

df["missing\_count"] = df.isnull().sum(axis=1)
max\_missing = df["missing\_count"].max()

Here the maximum missing count is 6. So to clean up some of the data, we will remove the rows with 4 or more missing values. df = df[df] missing count" | < 4 |

The above set of commands do the following function:

- i) Create a column called missing\_count where the sum of all the cells having null values is stored.
- ii) The maximum value from this missing count column is considered for deletion
- iii) Finally, we update the dataset by keeping the rows which have missing values less than a particular value

```
df["missing_count"] = df.isnull().sum(axis=1)
    max_missing = df["missing_count"].max()
    print(df.head())
    df = df[df["missing_count"] < 4]</pre>
               Gender Education Level Income Credit Score Loan Amount
                 Male
                                 PhD 72799.0
              Female
                          Bachelor's
   1 57
                                         NaN
                                                      690.0
                                                                33835.0
      21 Non-binary
                            Master's 55687.0
                        Bachelor's 26508.0
Bachelor's 49427.0
                                                                26541.0
                Male
                                                      622.0
   4 25 Non-binary
                                                                36528.0
                                                     766.0
     Employment Status Years at Current Job Payment History \
            Unemployed
                                         19
                                                      Poor
              Employed
                                                       Fair
            Unemployed
    4
            Unemployed
                                         10
      Debt-to-Income Ratio Assets Value Number of Dependents
                                                                   Country \
    a
                  0.154313
                               120228.0
                                                                    Cyprus
                                                         0.0 Turkmenistan
                  0.148920
                                 55849.0
                  0.362398
                               180700.0
                                                              Luxembourg
                  0.454964
                               157319.0
                                                         3.0
                                                                    Uganda
                  0.143242
                                                                   Namibia
                               287140.0
      Previous Defaults Risk Rating missing_count
                    3.0
                             Medium
                             Medium
                    3.0
                            Medium
                               Low
    (14909, 16)
```

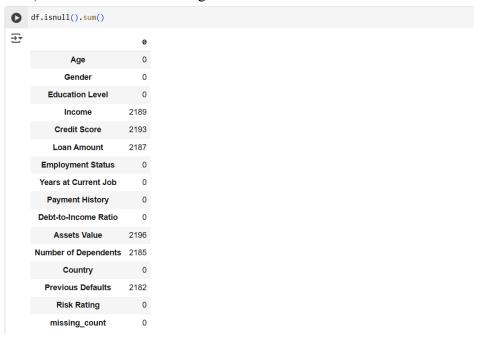
To check the total missing values in each columns.

df.isnull().sum() is used to check for missing values (NaN) in a dataset. Here's how it works:

df.isnull() creates a DataFrame of the same shape as df, where each value is True if it's missing (NaN) and False otherwise.

.sum() then counts the number of True values (missing values) in each column.

7) Take care of the missing values



So, there are many missing values, hence performing the next step.

- To take care of the missing data that has not been removed, one of the 2 methods can be used: If the feature is of a numeric data type, we can use either mean, median or mode of the feature. If the data is normally distributed, use mean, if it is skewed, use median, and if many values are repeated, use mode.
- If the feature contains different categories, there are 2 ways. Either fill it with the mode of the column, or add a custom value such as "Data Unavailable".

```
[ ] # handling the missing data
    df.fillna({'Income':df['Income'].median()},inplace=True)

[ ] df.fillna({'Credit Score':df['Credit Score'].median()},inplace=True)
```

To check the columns with missing values, using the following command df[df.isnull().any(axis=1)] filters and returns all rows that contain at least one missing (NaN) value.



## 8) Creating dummy variables

pd.get\_dummies(df, columns=categorical\_columns, prefix=categorical\_columns, drop\_first=False) is used to convert categorical variables into one-hot encoded format. This transformation helps machine learning models process categorical data.

# Breaking Down the Code:

pd.get\_dummies(df, columns=categorical\_columns, prefix=categorical\_columns,
drop\_first=False)

- Converts each categorical column into multiple binary (0/1) columns, representing unique categories.
- prefix=categorical columns ensures that the new columns have meaningful names.
- drop\_first=False keeps all categories (if True, it drops the first category to avoid multicollinearity).
- for col in categorical\_columns: df\_dummies[col] = df[col]

This restores the original categorical columns back into df\_dummies, so the dataset now contains both original and encoded versions.

Name: Alok Yadav Div: D15C Roll No.: 59

```
categorical_columns = ['Risk Rating', 'Gender', 'Employment Status', 'Payment History']
    df_dummies = pd.get_dummies(df, columns=categorical_columns, prefix=categorical_columns, drop_first=False)
    for col in categorical columns:
         df_dummies[col] = df[col]
    print(df dummies.head())
      Age Education Level Income Credit Score Loan Amount \
            Bachelor's 69773.0
                                              690.0
                                                          33835.0
             Master's 55687.0
Bachelor's 26508.0
                                              600.0
                                                          36623.0
                                        622.0
717.0
                  PhD 69773.0
                                                        15613.0
       Years at Current Job Debt-to-Income Ratio Assets Value
                                           0.154313 120228.000000
                               0.134515
0.148920
                                                      55849.000000
                                           0.362398 180700.000000
0.454964 157319.000000
                                      0.454964 15/313.00001
0.295984 159741.497176
       Number of Dependents Country ... Employment Status_Self-employed \ 0.0 Cyprus ... False
                         0.0 Turkmenistan ...
                                                                              False
                              Luxembourg ...
Uganda ...
                                Iceland ...
       Employment Status_Unemployed Payment History_Excellent \
                               False
                                                            False
       Payment History_Fair Payment History_Good Payment History_Poor \ False False True
                     True
                                                                     False
                                              False
                      False
                                              False
                                             False
       Risk Rating Gender Employment Status Payment History
Low Male Unemployed Poor
Medium Female Employed Fair
                                           Employed
                                                      Excellent
             Medium
                          Male
                                         Unemployed
            Medium Non-binary
                                         Unemployed
```

## 9) Detecting Outlier data

Using IQR Value:

In this method, we find the IQR value for the column; which is the difference between Q1 - 1.5 \* IQR and Q3 + 1.5 \* IQR. This is a standard that is followed, the factor 1.5 can be modified between 1 to 3 based on the requirement.

Command:

```
Q1 = df['Data_Value'].quantile(0.25)
Q3 = df['Data_Value'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df['Data_Value'] < lower_bound) | (df['Data_Value'] > upper_bound)]
```

This method gives the outliers and hence can be removed.

Using Manual method:

Checking for Outlier data in Excel using different value ranges.

And then using the preprocessed data.

```
df.to_csv('financial_risk_preprocessed.csv', index=False)

[ ] cleaned_df = pd.read_csv('/content/financial_risk_preprocessed WITH DEL.csv')
```

- 10) Standardization and Normalization of columns
- StandardScaler: Standardizes features by removing the mean and scaling to unit variance.
- MinMaxScaler: Normalizes features to a fixed range (0 to 1 by default).

### **Standardize Column:**

```
Using formula:

mean_value = df["Data_Value"].mean()

std_value = df["Data_Value"].std()

df["Standardized_Data_Value"] = (df["Data_Value"] - mean_value) / std_value

Using Library:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df['Standardized Data Value Scalar'] = scaler.fit_transform(df[['Data_Value']])
```

### Normalize column:

```
Method 1:
```

```
Formula min_val = df['Data_Value'].min()
max_val = df['Data_Value'].max()
df['Data_Value_Normalized'] = (df['Data_Value'] - min_val) / (max_val - min_val)
Method 2:
Scaler library from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df['Normalized Data Value Scalar'] = scaler.fit transform(df[['Data Value']])
```

Here, the columns, Income, Credit Score, and Loan Amount are standardized and normalized

#### **Conclusion:**

In this experiment, we used pandas and scikit learn to preprocess data, perform normalization and standardization to to make the dataset clean and efficient. Firstly, the dataset in the form of csv file was imported into the Collab and then using df.info(), information related about the features (columns) of the dataset and the data type of each of these columns. Using df.head(), the dataset in the form of dataframe can be viewed in which the top 5 values can be displayed. Then the missing values were detected using df.isnull() and performing handling methods like dropping the rows with missing values, and replacing missing values with mean, median or mode, the missing values were handled.

After handling missing values and encoding categorical variables using one-hot encoding (creating dummy variables), the dataset was further refined by detecting and removing outliers using the Interquartile Range (IQR) method. Outliers were identified based on their deviation from the first (Q1) and third quartile (Q3) thresholds and manual processing on the dataset using Excel, ensuring that extreme values did not affect the model's performance.

StandardScaler was used to transform the columns, Income, Credit Score to standardized and normalized by centering them around a mean of zero with unit variance, making them suitable for models that assume normally distributed data. On the other hand, MinMaxScaler was applied to Loan Amout to a fixed range between 0 and 1.