IMPORT SOME LIBRARIES AND READ THE CSV DATASET

```
import pandas as pd
import numpy as np
import scipy as sp
from sklearn import preprocessing



df = pd.read_csv('loan_approval_dataset.csv')

C:\Users\ashwi\AppData\Local\Temp\ipykernel_12364\679893019.py:1: 
DeprecationWarning: 
Pyarrow will become a required dependency of pandas in the next major 
release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, 
and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at 
https://github.com/pandas-dev/pandas/issues/54466
        
  import pandas as pd
```

HEAD OF THE DATASET

```
df.head()

   loan_id   no_of_dependents      education  self_employed  
income_annum  \
0        1                  2       Graduate             No  
9600000   
1        2                  0   Not Graduate            Yes  
4100000   
2        3                  3       Graduate             No  
9100000   
3        4                  3       Graduate             No  
8200000   
4        5                  5   Not Graduate            Yes  
9800000   

    loan_amount   loan_term   cibil_score  
residential_assets_value  \
0      29900000          12           778                  2400000   

1      12200000           8           417                  2700000   

2      29700000          20           506                  7100000   

3      30700000           8           467                 18200000   
```

```
4      24200000          20          382                    12400000


    commercial_assets_value   luxury_assets_value  
bank_asset_value  \
0                   17600000              22700000            8000000

1                    2200000               8800000            3300000

2                    4500000              33300000           12800000

3                    3300000              23300000            7900000

4                    8200000              29400000            5000000


   loan_status
0     Approved
1     Rejected
2     Rejected
3     Rejected
4     Rejected
```

TAIL OF THE DATASET

```
df.tail()

      loan_id   no_of_dependents      education  self_employed  
income_annum  \
4264     4265                  5       Graduate            Yes   
1000000   
4265     4266                  0   Not Graduate            Yes   
3300000   
4266     4267                  2   Not Graduate             No   
6500000   
4267     4268                  1   Not Graduate             No   
4100000   
4268     4269                  1       Graduate             No   
9200000   

       loan_amount   loan_term   cibil_score  
residential_assets_value  \
4264       2300000          12           317   
2800000   
4265      11300000          20           559   
4200000   
4266      23900000          18           457   
1200000   
4267      12800000           8           780
```

```
8200000
4268      29700000          10          607
17800000

       commercial_assets_value   luxury_assets_value
bank_asset_value  \
4264                    500000               3300000
800000
4265                   2900000              11000000
1900000
4266                  12400000              18100000
7300000
4267                    700000              14100000
5800000
4268                  11800000              35700000
12000000

      loan_status
4264     Rejected
4265     Approved
4266     Rejected
4267     Approved
4268     Approved
```

EXPLORATORY DATA ANAYSIS(EDA)

SHAPE OF THE DATASET

```
print("Rows and Colums of the Dataset : ",df.shape)

Rows and Colums of the Dataset :  (4269, 13)
```

INFO OF THE DATASET

provides the essential details about your dataset, such as the number of rows and columns, the number of non-null values, what type of data is in each column, and how much memory your DataFrame is using.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4269 entries, 0 to 4268
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   loan_id                    4269 non-null   int64 
 1    no_of_dependents          4269 non-null   int64 
 2    education                 4269 non-null   object
 3    self_employed             4269 non-null   object
 4    income_annum              4269 non-null   int64 
 5    loan_amount               4269 non-null   int64 
 6    loan_term                 4269 non-null   int64 
 7    cibil_score               4269 non-null   int64 
 8    residential_assets_value  4269 non-null   int64 
 9    commercial_assets_value   4269 non-null   int64 
 10   luxury_assets_value       4269 non-null   int64 
 11   bank_asset_value          4269 non-null   int64 
 12   loan_status               4269 non-null   object
dtypes: int64(10), object(3)
memory usage: 433.7+ KB
```

DESCRIPTION IF THE DATASET

The pandas.describe function is used to get a descriptive statistics summary of a given dataframe. This includes mean, count, std deviation, percentiles, and min-max values of all the features.

```
df.describe()

           loan_id   no_of_dependents   income_annum   loan_amount  \
count  4269.000000        4269.000000   4.269000e+03  4.269000e+03   
mean   2135.000000           2.498712   5.059124e+06  1.513345e+07   
std    1232.498479           1.695910   2.806840e+06  9.043363e+06   
min       1.000000           0.000000   2.000000e+05  3.000000e+05   
25%    1068.000000           1.000000   2.700000e+06  7.700000e+06   
50%    2135.000000           3.000000   5.100000e+06  1.450000e+07   
75%    3202.000000           4.000000   7.500000e+06  2.150000e+07   
max    4269.000000           5.000000   9.900000e+06  3.950000e+07   

         loan_term   cibil_score   residential_assets_value  \
count  4269.000000   4269.000000               4.269000e+03   
mean     10.900445    599.936051               7.472617e+06   
std       5.709187    172.430401               6.503637e+06   
min       2.000000    300.000000              -1.000000e+05   
25%       6.000000    453.000000               2.200000e+06   
50%      10.000000    600.000000               5.600000e+06   
75%      16.000000    748.000000               1.130000e+07   
max      20.000000    900.000000               2.910000e+07   

        commercial_assets_value   luxury_assets_value  
bank_asset_value  
count              4.269000e+03          4.269000e+03  
4.269000e+03  
mean               4.973155e+06          1.512631e+07  
4.976692e+06  
std                4.388966e+06          9.103754e+06  
3.250185e+06  
min                0.000000e+00          3.000000e+05  
0.000000e+00  
25%                1.300000e+06          7.500000e+06  
```

```
2.300000e+06
50%                3.700000e+06          1.460000e+07
4.600000e+06
75%                7.600000e+06          2.170000e+07
7.100000e+06
max                1.940000e+07          3.920000e+07
1.470000e+07
```

df.isnull().sum()

.isnull().sum() returns a DataFrame where each cell is either True or False depending on that cell's null status.

To count the number of nulls in each column we use an aggregate function for summing:

```
df.isnull()

      loan_id   no_of_dependents   education   self_employed
income_annum  \
0       False              False       False           False
False
1       False              False       False           False
False
2       False              False       False           False
False
3       False              False       False           False
False
4       False              False       False           False
False
...       ...                ...         ...             ...
...
4264    False              False       False           False
False
4265    False              False       False           False
False
4266    False              False       False           False
False
4267    False              False       False           False
False
4268    False              False       False           False
False

       loan_amount   loan_term   cibil_score
residential_assets_value  \
0            False       False         False
False
1            False       False         False
False
2            False       False         False
False
```

```
3            False       False         False
False
4            False       False         False
False
...            ...         ...           ...                        ..
.
4264         False       False         False
False
4265         False       False         False
False
4266         False       False         False
False
4267         False       False         False
False
4268         False       False         False
False

       commercial_assets_value   luxury_assets_value
bank_asset_value  \
0                        False                 False
False
1                        False                 False
False
2                        False                 False
False
3                        False                 False
False
4                        False                 False
False
...                        ...                   ...                ..
.
4264                     False                 False
False
4265                     False                 False
False
4266                     False                 False
False
4267                     False                 False
False
4268                     False                 False
False

       loan_status
0            False
1            False
2            False
3            False
4            False
...            ...
```

```
4264         False
4265         False
4266         False
4267         False
4268         False

[4269 rows x 13 columns]
```

```
df.isnull().sum()
```

```
loan_id                      0
 no_of_dependents            0
 education                   0
 self_employed               0
 income_annum                0
 loan_amount                 0
 loan_term                   0
 cibil_score                 0
 residential_assets_value    0
 commercial_assets_value     0
 luxury_assets_value         0
 bank_asset_value            0
 loan_status                 0
dtype: int64
```

ANALYSIS OF EACH COLUMN

if dropna = False it returns the count of null values and if its True then it returns non-null values

```
df.columns
```

```
Index(['loan_id', ' no_of_dependents', ' education', ' self_employed',
       ' income_annum', ' loan_amount', ' loan_term', ' cibil_score',
       ' residential_assets_value', ' commercial_assets_value',
       ' luxury_assets_value', ' bank_asset_value', ' loan_status'],
      dtype='object')
```

```
df[' education'].value_counts(dropna=False)
```

```
 education
Graduate        2144
Not Graduate    2125
Name: count, dtype: int64
```

```
new_df = df.dropna()
print(new_df)
```

```
      loan_id   no_of_dependents      education   self_employed
income_annum  \
0           1                  2       Graduate             No
9600000
```

```
1           2                  0   Not Graduate            Yes
4100000
2           3                  3       Graduate             No
9100000
3           4                  3       Graduate             No
8200000
4           5                  5   Not Graduate            Yes
9800000
...       ...                ...            ...            ...
...
4264     4265                  5       Graduate            Yes
1000000
4265     4266                  0   Not Graduate            Yes
3300000
4266     4267                  2   Not Graduate             No
6500000
4267     4268                  1   Not Graduate             No
4100000
4268     4269                  1       Graduate             No
9200000

       loan_amount   loan_term   cibil_score
residential_assets_value  \
0         29900000          12           778
2400000
1         12200000           8           417
2700000
2         29700000          20           506
7100000
3         30700000           8           467
18200000
4         24200000          20           382
12400000
...            ...         ...           ...                        ..
.
4264       2300000          12           317
2800000
4265      11300000          20           559
4200000
4266      23900000          18           457
1200000
4267      12800000           8           780
8200000
4268      29700000          10           607
17800000

       commercial_assets_value   luxury_assets_value
bank_asset_value  \
0                     17600000              22700000
```

```
8000000
1                      2200000               8800000
3300000
2                      4500000              33300000
12800000
3                      3300000              23300000
7900000
4                      8200000              29400000
5000000
...                        ...                   ...                  ..
.
4264                    500000               3300000
800000
4265                   2900000              11000000
1900000
4266                  12400000              18100000
7300000
4267                    700000              14100000
5800000
4268                  11800000              35700000
12000000

      loan_status
0        Approved
1        Rejected
2        Rejected
3        Rejected
4        Rejected
...           ...
4264     Rejected
4265     Approved
4266     Rejected
4267     Approved
4268     Approved

[4269 rows x 13 columns]
```

NORMALIZATION

```
arr = np.array(df[' loan_amount'])
#print(arr)
normalized_arr = preprocessing.normalize([arr])
print(normalized_arr)

[[0.02595843 0.01059173 0.02578479 ... 0.02074938 0.01111264
0.02578479]]

from sklearn.preprocessing import MinMaxScaler

normalization_scaling = MinMaxScaler()
```

```
normalized_data = normalization_scaling.fit_transform(df[['
cibil_score']])
print(normalized_data)
```

```
[[0.79666667]
 [0.195     ]
 [0.34333333]
 ...
 [0.26166667]
 [0.8       ]
 [0.51166667]]
```

Split the data into training, testing and validation sets

```
d_col = ['loan_id', ' education', ' self_employed']

df.drop(d_col,axis=1,inplace=True)

df.head()
```

```
   no_of_dependents   income_annum   loan_amount   loan_term   cibil_score
\
0                 2        9600000      29900000          12           778

1                 0        4100000      12200000           8           417

2                 3        9100000      29700000          20           506

3                 3        8200000      30700000           8           467

4                 5        9800000      24200000          20           382


   residential_assets_value   commercial_assets_value
luxury_assets_value  \
0                   2400000                  17600000
22700000
1                   2700000                   2200000
8800000
2                   7100000                   4500000
33300000
3                  18200000                   3300000
23300000
4                  12400000                   8200000
29400000

   bank_asset_value loan_status
0           8000000    Approved
1           3300000    Rejected
```

```
2          12800000    Rejected
3           7900000    Rejected
4           5000000    Rejected
```

```
from sklearn.model_selection import train_test_split
y = df[' loan_status']
X = df.drop(' loan_status',axis = 1)

X_train,X_test,y_train,y_test =
train_test_split(X,y,train_size=0.7,random_state=50)

X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2988 entries, 893 to 1931
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0    no_of_dependents          2988 non-null   int64
 1    income_annum              2988 non-null   int64
 2    loan_amount               2988 non-null   int64
 3    loan_term                 2988 non-null   int64
 4    cibil_score               2988 non-null   int64
 5    residential_assets_value  2988 non-null   int64
 6    commercial_assets_value   2988 non-null   int64
 7    luxury_assets_value       2988 non-null   int64
 8    bank_asset_value          2988 non-null   int64
dtypes: int64(9)
memory usage: 233.4 KB
```

STANDARDIZATION

```
from sklearn.preprocessing import StandardScaler

scaling = StandardScaler()

num_col = X_train.select_dtypes(include=['int64','float64']).columns
num_col
X_train[num_col] = scaling.fit_transform(X_train[num_col])

X_train
```

```
      no_of_dependents  income_annum  loan_amount  loan_term
cibil_score  \
893           1.455023      0.886012     1.061663  -0.506808
0.842310
3749         -1.486923      1.028895     1.239220   1.235818
1.429435
1958          0.866633     -0.292771    -0.048062  -0.855333    -
1.599198
```

```
102          -0.310145     -0.257050    -0.236715   1.584343
0.109858
974          -1.486923     -0.007005     0.373634   0.190242
0.888815
...                ...           ...          ...        ...
...
3330         -0.898534      0.457364     0.240467   0.190242
0.667917
70            0.866633     -1.614437    -1.501803   1.584343    -
1.506189
132           0.278244     -1.150068    -1.268760   1.235818    -
0.779549
2014          0.866633      1.350381     2.127000   0.887293
1.423622
1931          0.278244      0.457364     0.218272   0.887293    -
1.227159

      residential_assets_value  commercial_assets_value
luxury_assets_value  \
893                  -0.896438                -0.307055
0.188991
3749                  1.049592                 2.261685
1.370010
1958                 -0.170538                 0.225207            -
0.362888
102                   0.617141                -0.307055            -
0.396000
974                   1.111371                -0.214488            -
0.495339
...                        ...                      ...
...
3330                  0.416360                -0.168204
1.071996
70                   -1.128108                -1.093876            -
1.444569
132                  -1.050884                -1.024450            -
1.058254
2014                  1.744603                 1.405438
0.630493
1931                  0.663475                -0.654182
0.122765

      bank_asset_value
893           1.338644
3749          0.415647
1958         -0.599649
102          -0.784248
974           0.477180
...                ...
```

```
3330          1.092511
70           -1.338046
132          -0.722715
2014          1.769375
1931         -0.538116

[2988 rows x 9 columns]

X_train.isnull().sum()

no_of_dependents            0
income_annum                0
loan_amount                 0
loan_term                   0
cibil_score                 0
residential_assets_value    0
commercial_assets_value     0
luxury_assets_value         0
bank_asset_value            0
dtype: int64
```

FEATURE SELECTION

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

logreg= LogisticRegression()

rfe = RFE(estimator=logreg, n_features_to_select=5) # running RFE with 
15 variables as output15

rfe = rfe.fit(X_train,y_train)

col = X_train.columns[rfe.support_]

col

Index([' income_annum', ' loan_amount', ' loan_term', ' cibil_score',
       ' luxury_assets_value'],
      dtype='object')

# Assuming you have already fitted the RFE
feature_ranking = rfe.ranking_
selected_features_mask = rfe.support_

# Print the ranking of features
print("Feature Ranking:")
print(feature_ranking)

# Print the selected features based on the boolean mask
selected_features = X_train.columns[selected_features_mask]
```

```
print("\nSelected Features:")
print(selected_features)
```

```
Feature Ranking:
[5 1 1 1 1 3 4 1 2]

Selected Features:
Index([' income_annum', ' loan_amount', ' loan_term', ' cibil_score',
       ' luxury_assets_value'],
      dtype='object')
```

```
logreg_ranking = rfe.estimator_.coef_[0]
print("\nLogistic Regression Coefficient Ranking:")
print(logreg_ranking)
```

```

Logistic Regression Coefficient Ranking:
[ 1.31776459 -1.15086366  0.87859794 -4.1817298  -0.22225772]
```