10_ML - Finding missing value in data.ipynb

In [22]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

10.1 What is missing value

In [2]:
```
dataset = pd.read_csv('loan.csv')
```

In [3]:
```
dataset.head(3)
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [5]:
```
# To know how many rows and columns are pesent in the data
dataset.shape
```

Out[5]:
```
(614, 13)
```

In [6]:
```
# isnull function returns True where missing data is peresent and returns false in
dataset.isnull()
```

Out[6]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
609	False	False	False	False	False	False	False
610	False	False	False	False	False	False	False
611	False	False	False	False	False	False	False
612	False	False	False	False	False	False	False
613	False	False	False	False	False	False	False

614 rows × 13 columns

In [13]:

Out[13]: 149

In [7]:
```
dataset.isnull().sum()
```

Out[7]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [11]:
```
# To determine the percentage null value of each Head
(dataset.isnull().sum()/dataset.shape[0])
```

Out[11]:
```
Loan_ID              0.000000
Gender               0.021173
Married              0.004886
Dependents           0.024430
Education            0.000000
Self_Employed        0.052117
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.035831
Loan_Amount_Term     0.022801
Credit_History       0.081433
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

In [12]:
```
(dataset.isnull().sum()/dataset.shape[0]) * 100
```

Out[12]:
```
Loan_ID              0.000000
Gender               2.117264
Married              0.488599
Dependents           2.442997
Education            0.000000
Self_Employed        5.211726
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           3.583062
Loan_Amount_Term     2.280130
Credit_History       8.143322
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

In [14]:
```
# To determine totall null value in the data
dataset.isnull().sum().sum()
```

Out[14]:
```
149
```

In [18]:
```
dataset.shape
```

Out[18]:
```
(614, 13)
```

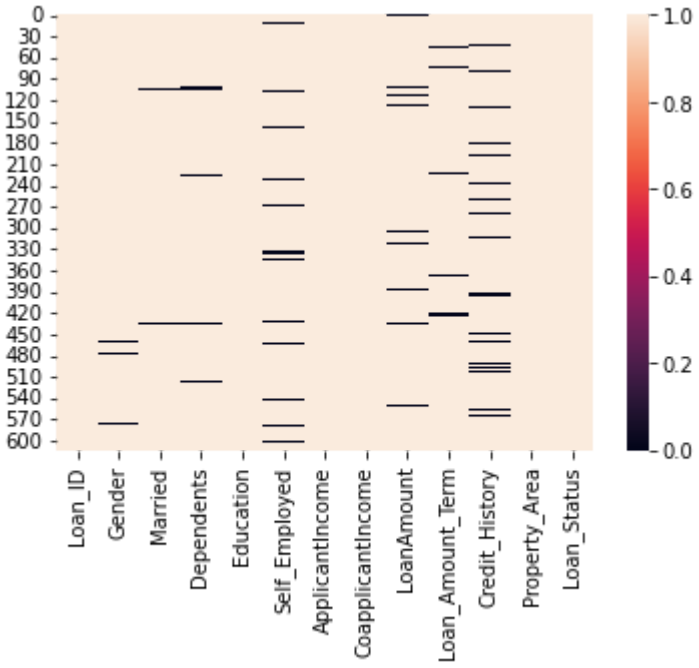
In [20]:
```
# To determine percentage totall null value in the data
# Total number of null data / total number of data * 100
dataset.isnull().sum().sum()/(dataset.shape[0] * dataset.shape[1])*100
```

Out[20]:
```
1.8667000751691305
```

In [21]:
```
# To check not null value in the data
dataset.notnull().sum()
```

Out[21]:
```
Loan_ID              614
Gender               601
Married              611
Dependents           599
Education            614
Self_Employed        582
ApplicantIncome      614
CoapplicantIncome    614
LoanAmount           592
Loan_Amount_Term     600
Credit_History       564
Property_Area        614
Loan_Status          614
dtype: int64
```

In [23]:
```
# To graphically plot the null data
sns.heatmap(dataset.notnull())
plt.show()
```



10.2 How to handle missing values (Dropping)

Deleting in 2 ways:

1. If a column contains 50% missing value, then delete the whole column
2. Only delete the rows which are having missing values, instead of deleting whole column

Deleting a column from the data

In [25]:
```
dataset.isnull().sum()
```

Out[25]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

We will delete Credit_History column as it contains more missing values

In [27]:
```
# Inplace function will enable to make changes in the original datasheet that is da
# It will write changes in the excisting file i.e., load.csv
dataset.drop(columns=['Credit_History'], inplace=True)
```

In [28]:
```
dataset.isnull().sum()
```

Out[28]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Property_Area         0
Loan_Status           0
dtype: int64
```

In [29]:
```
dataset.shape
```

Out[29]:
```
(614, 12)
```

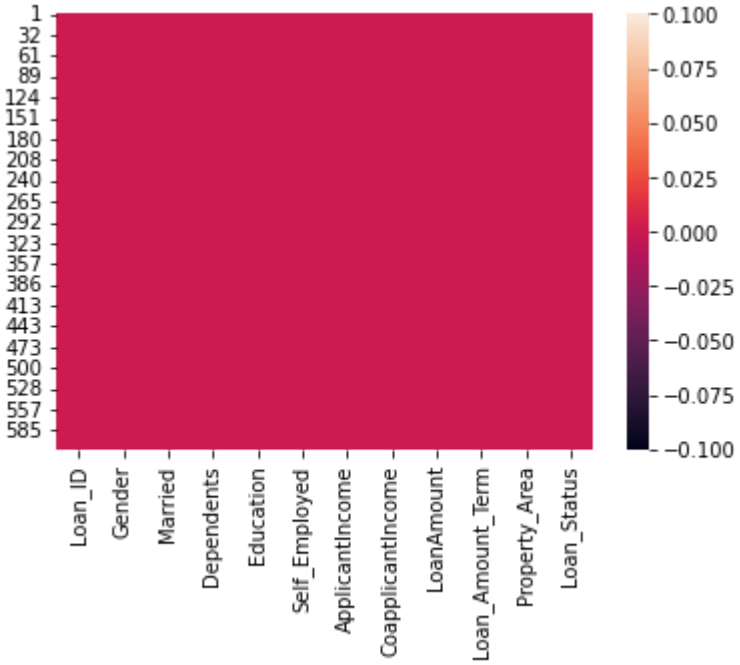
Deleting rows containing null values

In [30]:
```
# Again we use inplace function to write the changes in the same datasheet instead
dataset.dropna(inplace=True)
```

In [31]:
```
dataset.isnull().sum()
```

Out[31]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Property_Area        0
Loan_Status          0
dtype: int64
```

In [32]:
```
sns.heatmap(dataset.isnull())
plt.show()
```



In [34]:
```
dataset.shape
```

Out[34]:
```
(523, 12)
```

To check how much data has been dropped (deleted)

In [37]:
```
((614-523)/614)*100
```

Out[37]:
```
14.82084690553746
```

14% data is lost

10.3 Handling Missing Values (Imputing Category Data)

While dropping the data can be harmful as it may contain essential data, so instead of deleting we will fill the data where the missing values are present

We will import the orginial data of loan.csv, as we have dropped missing data and overwrite the changes in the above data

In [38]:
```
dataset = pd.read_csv('loan.csv')
```

In [39]:
```
dataset.head(3)
```

Out[39]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [40]:
```
dataset.isnull().sum()
```

Out[40]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [42]:
```
# It will fill all the data using the number provided i.e., 10
# this method is not recommended as it will fill dummy data, lead to wrong insight
dataset.fillna(10)
```

Out[42]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

614 rows × 13 columns

So we will fill data wisely, so we will first determine the datatype String data type is called object data in ML Data is of two types:

1. Numerical data
2. Categorical data - string data (object type data) Filling in categorical data:
3. Backward filling
4. Forward filling
5. Mod filling

In [43]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [47]:
```
# Backward filling - Back data will filled , forexample Loan Amount first row is fi
# nichay wala data oper a k fill ho jaye ga
dataset.fillna(method='bfill')
```

Out[47]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

614 rows × 13 columns

In [48]:
```
# Forward filling - oper wala data nicha a k fill ho jaye ga
# by default filling is row wise
dataset.fillna(method='ffill')
```

Out[48]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

614 rows × 13 columns

In [50]:
```
# by default filling is row wise - So fill data column wise, we will use axis
dataset.fillna(method='ffill', axis=1)
```

Out[50]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

614 rows × 13 columns

- In mode data filling, you will most repeatitive data in missing contents
- fill the missing data in Gender column

Fill particular column containing missing value by mode method

In [51]:
```
dataset['Gender'].mode()
```

Out[51]:
```
0    Male
Name: Gender, dtype: object
```

In [52]:
```
dataset['Gender'].mode()[0]
```

Out[52]:
```
'Male'
```

In [54]:
```
dataset['Gender'].fillna(dataset['Gender'].mode()[0], inplace=True)
```

In [55]:
```
dataset.isnull().sum()
```

Out[55]:
```
Loan_ID               0
Gender                0
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Fill all columns containing missing value by mode method

1. First you will collect all object datatype

In [57]:
```
dataset.select_dtypes(include='object')
```

Out[57]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area	L
0	LP001002	Male	No	0	Graduate	No	Urban	
1	LP001003	Male	Yes	1	Graduate	No	Rural	
2	LP001005	Male	Yes	0	Graduate	Yes	Urban	
3	LP001006	Male	Yes	0	Not Graduate	No	Urban	
4	LP001008	Male	No	0	Graduate	No	Urban	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	Rural	
610	LP002979	Male	Yes	3+	Graduate	No	Rural	
611	LP002983	Male	Yes	1	Graduate	No	Urban	
612	LP002984	Male	Yes	2	Graduate	No	Urban	
613	LP002990	Female	No	0	Graduate	Yes	Semiurban	

614 rows × 8 columns

In [58]:
```
dataset.select_dtypes(include='object').isnull()
```

Out[58]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Lo
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
609	False	False	False	False	False	False	False	
610	False	False	False	False	False	False	False	
611	False	False	False	False	False	False	False	
612	False	False	False	False	False	False	False	
613	False	False	False	False	False	False	False	

614 rows × 8 columns

In [59]:
```
dataset.select_dtypes(include='object').isnull().sum()
```

Out[59]:
```
Loan_ID            0
Gender             0
Married            3
Dependents        15
Education          0
Self_Employed     32
Property_Area      0
Loan_Status        0
dtype: int64
```

In [60]:
```
dataset.select_dtypes(include='object').columns
```

Out[60]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [61]:
```
for i in dataset.select_dtypes(include='object').columns:
    print(i)
```

```
Loan_ID
Gender
Married
Dependents
Education
Self_Employed
Property_Area
Loan_Status
```

In [63]:
```
for i in dataset.select_dtypes(include='object').columns:
    #dataset['Gender'].fillna(dataset['Gender'].mode()[0], inplace=True)
    dataset[i].fillna(dataset[i].mode()[0], inplace=True)
```

In [64]:
```
dataset.isnull().sum()
```

Out[64]:
```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

So all object data has been filled and only numerical data is left to be filled

10.4 Handling Missing Values (Scikit-learn)

Import fresh datasheet

In [65]:
```
dataset = pd.read_csv('loan.csv')
```

In [66]:
```
dataset.head(3)
```

Out[66]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [67]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [69]:
```
# To show numerical data type (float)
dataset.select_dtypes(include='float64')
```

Out[69]:

	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.0	NaN	360.0	1.0
1	1508.0	128.0	360.0	1.0
2	0.0	66.0	360.0	1.0
3	2358.0	120.0	360.0	1.0
4	0.0	141.0	360.0	1.0
...	...	...	...	...
609	0.0	71.0	360.0	1.0
610	0.0	40.0	180.0	1.0
611	240.0	253.0	360.0	1.0
612	0.0	187.0	360.0	1.0
613	0.0	133.0	360.0	0.0

614 rows × 4 columns

In [70]:
```
dataset.select_dtypes(include='float64').columns
```

Out[70]:
```
Index(['CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term',
       'Credit_History'],
      dtype='object')
```

Find the missing values using scikit learn

In [71]:
```
from sklearn.impute import SimpleImputer
```

- sklearn provide variety of options to fill the data. for example fill the data by mean, most fequency (mode), median
- We are now filling the data by using mean method

In [74]:
```
si = SimpleImputer(strategy='mean')
si.fit_transform(dataset[['CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term',
       'Credit_History']])
```

Out[74]:
```
array([[0.00000000e+00, 1.46412162e+02, 3.60000000e+02, 1.00000000e+00],
       [1.50800000e+03, 1.28000000e+02, 3.60000000e+02, 1.00000000e+00],
       [0.00000000e+00, 6.60000000e+01, 3.60000000e+02, 1.00000000e+00],
       ...,
       [2.40000000e+02, 2.53000000e+02, 3.60000000e+02, 1.00000000e+00],
       [0.00000000e+00, 1.87000000e+02, 3.60000000e+02, 1.00000000e+00],
       [0.00000000e+00, 1.33000000e+02, 3.60000000e+02, 0.00000000e+00]])
```

In [82]:
```
# To convert this data into csv sheet
si = SimpleImputer(strategy='mean')
```

```
arr = si.fit_transform(dataset[['CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Ter
       'Credit_History']])
```

In [89]:
```
new_dataset = pd.DataFrame(arr, columns=dataset.select_dtypes(include='float64').co
```

In [90]:
```
new_dataset.isnull().sum()
```

Out[90]:
```
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
dtype: int64
```

In [91]:
```
new_dataset
```

Out[91]:

	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.0	146.412162	360.0	1.0
1	1508.0	128.000000	360.0	1.0
2	0.0	66.000000	360.0	1.0
3	2358.0	120.000000	360.0	1.0
4	0.0	141.000000	360.0	1.0
...	...	...	...	...
609	0.0	71.000000	360.0	1.0
610	0.0	40.000000	180.0	1.0
611	240.0	253.000000	360.0	1.0
612	0.0	187.000000	360.0	1.0
613	0.0	133.000000	360.0	0.0

614 rows × 4 columns

In [92]:
```
dataset['LoanAmount'].mean()
```

Out[92]:
```
146.41216216216216
```