Name :- Rishabh Sharma

Registration Number :- 20MAI0082

Github Link :-

https://github.com/rishabh5197/Machine-Learning-Assignments/tree/main/Assignment-2

Loading 1st dataset which is CRX data

In [1]:

```python
# importing essential libraries to do the following task.
import numpy as np
import pandas as pd
```

In [2]:

```python
# crx_data = pd.read_csv("crx.data",names=["col"+str(i) for i in range(0,16)])
crx_data = pd.read_csv("crx.data",names=["col"+str(i+1) for i in range(0,16)])
```

In [3]:

```python
# visualizing the first 5 rows to know whether the data is loaded in correct manner or n
crx_data.head(5)
```

Out[3]:

	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	col1
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	56
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	82
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	

Displaying the last 10 rows of crx_data

In [4]:

```python
# tail is used to denote the values from last
crx_data.tail(10)
```

Out[4]:

	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14
680	b	19.50	0.290	u	g	k	v	0.290	f	f	0	f	g	00280
681	b	27.83	1.000	y	p	d	h	3.000	f	f	0	f	g	00176
682	b	17.08	3.290	u	g	i	v	0.335	f	f	0	t	g	00140
683	b	36.42	0.750	y	p	d	v	0.585	f	f	0	f	g	00240
684	b	40.58	3.290	u	g	m	v	3.500	f	f	0	t	s	00400
685	b	21.08	10.085	y	p	e	h	1.250	f	f	0	f	g	00260
686	a	22.67	0.750	u	g	c	v	2.000	f	t	2	t	g	00200
687	a	25.25	13.500	y	p	ff	ff	2.000	f	t	1	t	g	00200
688	b	17.92	0.205	u	g	aa	v	0.040	f	f	0	f	g	00280
689	b	35.00	3.375	u	g	c	h	8.290	f	f	0	t	g	00000

Replace the ‘?’ with Not-a-Number

In [7]:

```python
crx_data.replace('?',np.nan)
```

Out[7]:

	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	00260
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	00200
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	00200
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	00280
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	00000

690 rows × 16 columns

In [8]:

```python
crx_data.isnull.sum()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-8-de0b73093ffc> in <module>
----> 1 crx_data.isnull.sum()

AttributeError: 'function' object has no attribute 'sum'
```

In [9]:

```python
crx_data.isnull().sum()
```

Out[9]:

```
col1     0
col2     0
col3     0
col4     0
col5     0
col6     0
col7     0
col8     0
col9     0
col10    0
col11    0
col12    0
col13    0
col14    0
col15    0
col16    0
dtype: int64
```

In [10]:
```
crx_data=crx_data.replace('?',np.nan)
```

In [11]:
```
crx_data.isnull().sum()
```

Out[11]:
```
col1     12
col2     12
col3      0
col4      6
col5      6
col6      9
col7      9
col8      0
col9      0
col10     0
col11     0
col12     0
col13     0
col14    13
col15     0
col16     0
dtype: int64
```

Comment on the datatype of variables

In [12]:
```
# the info method of pandas dataframe gives detailed information about the columns an
crx_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 16 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   col1    678 non-null    object 
 1   col2    678 non-null    object 
 2   col3    690 non-null    float64
 3   col4    684 non-null    object 
 4   col5    684 non-null    object 
 5   col6    681 non-null    object 
 6   col7    681 non-null    object 
 7   col8    690 non-null    float64
 8   col9    690 non-null    object 
 9   col10   690 non-null    object 
 10  col11   690 non-null    int64  
 11  col12   690 non-null    object 
 12  col13   690 non-null    object 
 13  col14   677 non-null    object 
 14  col15   690 non-null    int64  
 15  col16   690 non-null    object 
dtypes: float64(2), int64(2), object(12)
memory usage: 86.4+ KB
```

In [18]:

```
for i in crx_data:
    print(type(i))
```

```
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
```

In [17]:

```
# as the above code displayed the names of column it prints column's name type which w
for i in crx_data:
    print(crx_data[i].dtype)
```

```
object
object
float64
object
object
object
object
float64
object
object
int64
object
object
object
int64
object
```

In [19]:

```
float_,int_,object_=[],[],[]
for i in crx_data:
    if crx_data[i].dtype=="object":
        object_.append(i)
    elif crx_data[i].dtype == "float64":
        float_.append(i)
    elif crx_data[i].dtype == "int64":
        int_.append(i)
    else:
        print(i,"something went wrong")
```

In [27]:

```
print("there are", len(float_) ,"columns having datatype as float and they are :- ",end=" '
for i in float_:
    print(i,end=" , ")
print("\nthere are", len(int_) ,"columns having datatype as int and they are :- ",end=" ")
for i in int_:
    print(i,end=" , ")
print("\nthere are", len(object_) ,"columns having datatype as string and they are :- ",er
for i in object_:
    print(i,end=" , ")
```

```
there are 2 columns having datatype as float and they are :-  col3 , col8 ,
there are 2 columns having datatype as int and they are :-  col11 , col15 ,
there are 12 columns having datatype as string and they are :-  col1 , col2 , col4 , col5 , col6 ,
col7 , col9 , col10 , col12 , col13 , col14 , col16 ,
```

The col16 has + and -, replace them ‘P’ and ‘N’ respectively

In [28]:

```
crx_data['col16'].head()
```

Out[28]:

```
0    +
1    +
2    +
3    +
4    +
Name: col16, dtype: object
```

In [30]:

```
# checking though if any another value exits or not
crx_data["col16"].unique()
```

Out[30]:

```
array(['+', '-'], dtype=object)
```

In [31]:

```
# using replace method to replace + with P and - with N
crx_data["col16"] = crx_data["col16"].replace("+","P").replace("-","N")
```

In [32]:

```
crx_data["col16"]
```

Out[32]:

```
0      P
1      P
2      P
3      P
4      P
      ..
685    N
686    N
687    N
688    N
689    N
Name: col16, Length: 690, dtype: object
```

In [33]:

```python
crx_data["col16"].unique()
```

Out[33]:

```
array(['P', 'N'], dtype=object)
```

Find and display the number of variables of type ‘Object’

In [34]:

```python
print("the object types columns are :-")
for i in object_:
    print(i)
```

```
the object types are :-
col1
col2
col4
col5
col6
col7
col9
col10
col12
col13
col14
col16
```

loading 2nd Dataset which is loan.csv

In [35]:

```python
loan_data  = pd.read_csv("loan.csv")
```

In [36]:

```python
loan_data.head(5)
```

Out[36]:

	customer_id	disbursed_amount	interest	market	employment	time_employed	householder
0	0	23201.5	15.4840	C	Teacher	<=5 years	RENT
1	1	7425.0	11.2032	B	Accountant	<=5 years	OWNER
2	2	11150.0	8.5100	A	Statistician	<=5 years	RENT
3	3	7600.0	5.8656	A	Other	<=5 years	RENT
4	4	31960.0	18.7392	E	Bus driver	>5 years	RENT

Display the mean of any two variables with continuous values

In [38]:

```
# there are 3 continous variables that are disbursed_amount,interest and income
# hence printing mean of disbursed_amount and interest
loan_data['disbursed_amount'].mean()
```

Out[38]:

```
14132.2755
```

In [39]:

```
loan_data['interest'].mean()
```

Out[39]:

```
12.678819440000039
```

Print the number of discrete variables

In [40]:

```
loan_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   customer_id            10000 non-null  int64  
 1   disbursed_amount       10000 non-null  float64
 2   interest               10000 non-null  float64
 3   market                 10000 non-null  object 
 4   employment             9389 non-null   object 
 5   time_employed          9471 non-null   object 
 6   householder            10000 non-null  object 
 7   income                 10000 non-null  float64
 8   date_issued            10000 non-null  object 
 9   target                 10000 non-null  int64  
 10  loan_purpose           10000 non-null  object 
 11  number_open_accounts   10000 non-null  float64
 12  date_last_payment      10000 non-null  object 
 13  number_credit_lines_12  238 non-null    float64
dtypes: float64(5), int64(2), object(7)
memory usage: 1.1+ MB
```

In [ ]:

```
# name_of_columns = []
# for i in loan_data:
#     if loan_data[i].dtype =="object":
#         name_of_columns.append(i)
# print("the columns that are discrete in nature are :- ",end=" ")
# for i in name_of_columns:
#     if (i =="date_last_payment" or  i == 'date_issued'): # removing these columns as th
#         pass
#     else:
#         print(i,end=" , ")
```

In [54]:

```
categorical_columns = [i for  i in loan_data.columns if loan_data[i].dtype == "O"]
```

In [56]:

```
print("the columns that are discrete in nature are :- ")
for i in categorical_columns:
    print(i)
```

```
the columns that are discrete in nature are :-
market
employment
time_employed
householder
date_issued
loan_purpose
date_last_payment
```

Display the unique values of two variables with discrete values

In [52]:

```
loan_data["market"].unique()
```

Out[52]:

```
array(['C', 'B', 'A', 'E', 'D'], dtype=object)
```

In [53]:

```
loan_data["employment"].unique()
```

Out[53]:

```
array(['Teacher', 'Accountant', 'Statistician', 'Other', 'Bus driver',
       'Secretary', 'Software developer', 'Nurse', 'Taxi driver', nan,
       'Civil Servant', 'Dentist'], dtype=object)
```

Display the Month with most of loans issued date

In [57]:

```
month = loan_data['date_last_payment'].dt.month
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-57-8720de89590b> in <module>
----> 1 month = loan_data['date_last_payment'].dt.month

~\anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5456             or name in self._accessors
   5457         ):
-> 5458             return object.__getattribute__(self, name)
   5459         else:
   5460             if self._info_axis._can_hold_identifiers_and_holds_name(name):

~\anaconda3\lib\site-packages\pandas\core\accessor.py in __get__(self, obj, cls)
    178             # we're accessing the attribute of the class, i.e., Dataset.geo
    179             return self._accessor
--> 180         accessor_obj = self._accessor(obj)
    181         # Replace the property with the accessor object. Inspired by:
    182         # https://www.pydanny.com/cached-property.html (https://www.pydanny.co
m/cached-property.html)

~\anaconda3\lib\site-packages\pandas\core\indexes\accessors.py in __new__(cls, data)
    492             return PeriodProperties(data, orig)
    493 
--> 494         raise AttributeError("Can only use .dt accessor with datetimelike value
s")

AttributeError: Can only use .dt accessor with datetimelike values
```

In [79]:

```
loan_data['date_issued']=pd.to_datetime(loan_data['date_issued'])
```

In [80]:

```
month = loan_data['date_issued'].dt.month
```

In [81]:

```
month.value_counts()
```

Out[81]:

```
10    1277
7     1066
11    1017
12     882
8      852
4      816
5      749
9      734
1      700
6      700
3      623
2      584
Name: date_issued, dtype: int64
```

In [83]:

```
print(month.value_counts() > 1000)
```

```
10     True
7      True
11     True
12    False
8     False
4     False
5     False
9     False
1     False
6     False
3     False
2     False
Name: date_issued, dtype: bool
```

In [85]:

```
for i in month.value_counts():
    if i > 1000:
        print(i)
```

```
1277
1066
1017
```

In [86]:

```
for i,x in zip(month.value_counts().keys(),month.value_counts()):
    if x > 1000:
        print("Month number :- ",i,"       Month Counts :- ",x)
```

```
Month number :-  10        Month Counts :-  1277
Month number :-  7        Month Counts :-  1066
Month number :-  11        Month Counts :-  1017
```

Display the count of ‘Teacher’ who are ‘Owners’

In [97]:

```
new_df = loan_data[["employment",'householder']]
```

In [98]:

```
new_df = new_df.loc[new_df['employment'] == 'Teacher']
```

In [99]:

```
new_df = new_df.loc[new_df['householder'] == 'OWNER']
```

In [100]:

```
new_df.head()
```

Out[100]:

	employment	householder
71	Teacher	OWNER
85	Teacher	OWNER
171	Teacher	OWNER
672	Teacher	OWNER
1024	Teacher	OWNER

In [102]:

```
print("there are ",new_df.shape[0],"teacher who are owners")
```

```
there are  69 teacher who are owners
```

Display the ‘Employment’ of customers who mostly ‘Rent’

In [107]:

```
new_df_1 = loan_data[["employment",'householder']]
new_df_1 = new_df_1.loc[new_df_1['householder'] == 'RENT']
print("there are ",new_df_1.shape[0],"employee who are on rent")
```

```
there are  4055 employee who are on rent
```

In [108]:

```
new_df_1
```

Out[108]:

	employment	householder
0	Teacher	RENT
2	Statistician	RENT
3	Other	RENT
4	Bus driver	RENT
6	Secretary	RENT
...	...	...
9991	Software developer	RENT
9992	Statistician	RENT
9994	Accountant	RENT
9996	Civil Servant	RENT
9998	Bus driver	RENT

4055 rows × 2 columns