LAB ASSIGNMENT – 7

DATA CLEANING & TRANSFORMATION

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

1Q) Generate 100 random numbers for the salary attribute(100K-1000K) and plot the equal-width (10 bins) and equal frequency (20 values) histograms. Develop user defined functions to perform sampling of the salary attribute: SRSWOR, SRSWR, and stratified sampling. Use samples of size 5 and the strata "low," "medium," and "high"

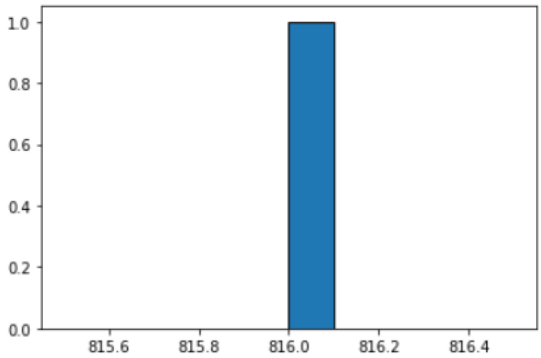
CODE AND OUTPUT :

```
[2] import pandas as pd
    import numpy as np
    import random
```

```
[4] randomlist = []
    for i in range(0,100):
      n = random.randint(100,1000)
    randomlist.append(n)
    print(randomlist)
```

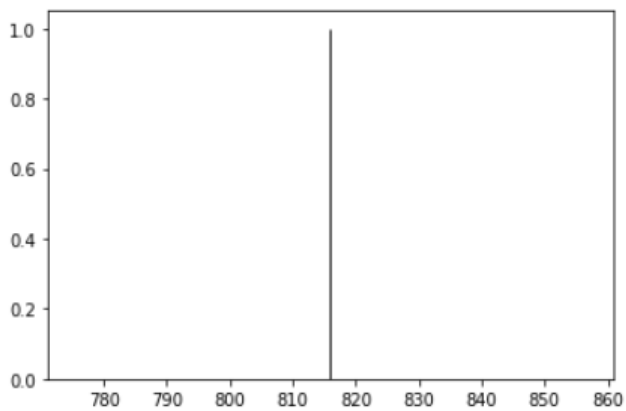
```
[816]
```

```
import matplotlib.pyplot as plt
n, bins, patches = plt.hist(randomlist, edgecolor='black')
plt.show()
```



```
[7] def equalObs(x, nbin):
        nlen = len(x)
        return np.interp(np.linspace(0, nlen, nbin + 1), np.arange(nlen), np.sort(x))
```

```
[8] n, bins, patches = plt.hist(randomlist, equalObs(randomlist, 10), edgecolor='black')
    plt.show()
```




```
from random import choices
strata = ["low","medium","high"]
randomlist2 = []
```

```
[13] for i in range(0,5):
         n = random.randint(100,1000)
         randomlist2.append(n)
```

```
[15] choices(randomlist2, k=5)
```

```
[966, 506, 694, 966, 966]
```

```
[16] from random import sample
     random.sample(randomlist2,5)
```

```
[506, 550, 694, 840, 139]
```

```
strata = ["low","medium","high"]
np.random.choice(["low","medium","high"], size=5, p=[0.3, 0.4,0.3])
```

```
array(['medium', 'high', 'low', 'medium', 'low'], dtype='<U6')
```

2Q)

2.

a. Load ***crx.data*** into a data frame and do the following operations: (The data has no headers)

- Change the column names to A1 to A16
- Replace all '?' marks with np.nan
- Convert A2 and A14 attributes to float data type
- Convert '+' to 1 and "–" to 0 of A16 attribute
- Replace values of "A3, A8, A9, A10" attributes to np.nan in 50 random objects
- Save the file as `Transformed_crx.csv`

b. Ignoring missing values:

- Load the Credit Approval Data Set `Transformed_crx.csv`
- Calculate the percentage of missing values for each variable and sort them in ascending order
- Remove the observations with missing data in any of the variables
- Print and compare the size of the original and complete case datasets

c. Performing mean and median imputation:

- Load the Credit Approval Data Set `Transformed_crx.csv`
- Replace the missing values with the median in five numerical variables 'A2', 'A3', 'A8', 'A11', 'A15' using pandas
- Replace the missing values with the mean in five numerical variables 'A2', 'A3', 'A8', 'A11', 'A15' using pandas
- Use SimpleImputer() of scikit-learn to fill the missing values with median and mean, separately.

d. Performing mode or frequent category imputation:

- Load the Credit Approval Data Set `Transformed_crx.csv`
- Replace the missing values with the mode in the attributes 'A4', 'A5', 'A6', 'A7' using pandas
- Use SimpleImputer() of scikit-learn to fill the missing values with mode.

e. Performing most probable value imputation:

- Load the Credit Approval Data Set `Transformed_crx.csv`
- Replace the missing values with probable values using linear regression in the attributes 'A2', 'A3', 'A8', 'A11', 'A15' using pandas

2a)

CODE AND OUTPUT :

```
[15] df=pd.read_csv('crx.data.csv',header=None)
```

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

```
[54] varnames = ['A'+str(s) for s in range(1,17)]
     df.columns = varnames
```

```
[55] df.columns
```

```
Index(['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'A11',
       'A12', 'A13', 'A14', 'A15', 'A16'],
      dtype='object')
```

```
[57] df= df.replace('?',np.nan)
```

[58]
```
df
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

690 rows × 16 columns

[47]
```
print(df.isnull().sum())
```

```
A1     12
A2     12
A3      0
A4      6
A5      6
A6      9
A7      9
A8      0
A9      0
A10     0
A11     0
A12     0
A13     0
A14    13
A15     0
A16     0
dtype: int64
```

[59]
```
df['A2'] = pd.to_numeric(df['A2'])
```

```
[60] df['A2']
```

```
0      30.83
1      58.67
2      24.50
3      27.83
4      20.17
       ...
685    21.08
686    22.67
687    25.25
688    17.92
689    35.00
Name: A2, Length: 690, dtype: float64
```

```
[61] df['A14'] = pd.to_numeric(df['A14'])
```

```
df['A14']
```

```
0      202.0
1       43.0
2      280.0
3      100.0
4      120.0
       ...
685    260.0
686    200.0
687    200.0
688    280.0
689      0.0
Name: A14, Length: 690, dtype: float64
```

```
[63] df=df.replace({'A16': {'+': 1,'-': 0}})
```

```
df
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

690 rows × 16 columns

```
[52] df.to_csv(' Transformed_crx.csv', index=False)
```

```
[25] df=pd.read_csv('crx.data.csv',header=None)
```

```
[30] df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	00260	0	-
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	00200	394	-
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	00200	1	-
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	00280	750	-
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	00000	0	-

690 rows × 16 columns

2b) CODE AND OUTPUT:

```
[65] df1=pd.read_csv('/content/ Transformed_crx.csv')
```

```
[66] df1
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

690 rows × 16 columns

```
[67] print(df1.isnull().sum().sort_values(ascending=True))
```

```
A3      0
A8      0
A9      0
A10     0
A11     0
A12     0
A13     0
A15     0
A16     0
A4      6
A5      6
A6      9
A7      9
A1     12
A2     12
A14    13
dtype: int64
```

```
[68] df2=df1.dropna()
```

```
df2
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

653 rows × 16 columns

2c) CODE AND OUTPUT :

```
[71] from sklearn.impute import SimpleImputer
     dfMean=df1
```

```
dfMean
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

690 rows × 16 columns

```
[73] imputer = SimpleImputer(missing_values = np.nan,strategy ='mean')
```

```
[74] imputer
```

```
imputer
```

```
SimpleImputer()
```

```
[77] imputer = imputer.fit(dfMean[['A2']])
```

```
[79] dfMean['A2'] = imputer.transform(dfMean[['A2']])
```

```
[80] dfMean['A2']
```

```
0      30.83
1      58.67
2      24.50
3      27.83
4      20.17
       ...
685    21.08
686    22.67
687    25.25
688    17.92
689    35.00
Name: A2, Length: 690, dtype: float64
```

```
[84] dfMean['A3'] = imputer.transform(dfMean[['A3']])
```

```
dfMean['A3']
```

```
0       0.000
1       4.460
2       0.500
3       1.540
4       5.625
        ...
685    10.085
686     0.750
687    13.500
688     0.205
689     3.375
Name: A3, Length: 690, dtype: float64
```

```
[85] dfMean['A8'] = imputer.transform(dfMean[['A8']])
```

```
[86] dfMean['A8']
```

```
0      1.25
1      3.04
2      1.50
3      3.75
4      1.71
       ...
685    1.25
686    2.00
687    2.00
688    0.04
689    8.29
Name: A8, Length: 690, dtype: float64
```

```
dfMean['A11'] = imputer.transform(dfMean[['A11']])
```

```
[89] dfMean['A11']
```

```
0      1.0
1      6.0
2      0.0
3      5.0
4      0.0
      ...
685    0.0
686    2.0
687    1.0
688    0.0
689    0.0
Name: A11, Length: 690, dtype: float64
```

```
[90] dfMean['A15'] = imputer.transform(dfMean[['A15']])
```

```
dfMean['A15']
```

```
0        0.0
1      560.0
2      824.0
3        3.0
4        0.0
       ...
685      0.0
686    394.0
687      1.0
688    750.0
689      0.0
Name: A15, Length: 690, dtype: float64
```

```
[77] from numpy.lib.function_base import median
     import numpy as np
     import pandas as pd
     from sklearn.impute import SimpleImputer
```

```
[78] df1=pd.read_csv('/content/ Transformed_crx.csv')
```

```
[83] dfMedian=df1
```

```
dfMedian
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

```
[79] imputer1 = SimpleImputer(missing_values = np.nan,strategy ='median')
```

```
[80] imputer1
```

```
SimpleImputer(strategy='median')
```

```
[85] imputer1=imputer1.fit(dfMedian[['A2']])
```

```
[86] imputer1
```

```
SimpleImputer(strategy='median')
```

```
[88] dfMedian['A2'] = imputer1.transform(dfMedian[['A2']])
```

```
dfMedian['A2']
```

```
0      30.83
1      58.67
2      24.50
3      27.83
4      20.17
       ...
685    21.08
686    22.67
687    25.25
688    17.92
689    35.00
Name: A2, Length: 690, dtype: float64
```

```
[90] dfMedian['A3'] = imputer1.transform(dfMedian[['A3']])
```

```
[91] dfMedian['A3']
```

```
0       0.000
1       4.460
2       0.500
3       1.540
4       5.625
        ...
685    10.085
686     0.750
687    13.500
688     0.205
689     3.375
Name: A3, Length: 690, dtype: float64
```

```
dfMedian['A8'] = imputer1.transform(dfMedian[['A8']])
```

```
0      1.25
1      3.04
2      1.50
3      3.75
4      1.71
       ...
685    1.25
686    2.00
687    2.00
688    0.04
689    8.29
Name: A8, Length: 690, dtype: float64
```

```
[94] dfMedian['A11'] = imputer1.transform(dfMedian[['A11']])
```

```
[96] dfMedian['A11']
```

```
0      1.0
1      6.0
2      0.0
3      5.0
4      0.0
      ...
685    0.0
686    2.0
687    1.0
688    0.0
689    0.0
Name: A11, Length: 690, dtype: float64
```

```
dfMedian['A15'] = imputer1.transform(dfMedian[['A15']])
```

```
[98] dfMedian['A15']
```

```
0        0.0
1      560.0
2      824.0
3        3.0
4        0.0
       ...
685      0.0
686    394.0
687      1.0
688    750.0
689      0.0
Name: A15, Length: 690, dtype: float64
```

```
dfMedian.isnull().sum()
```

```
A1     12
A2      0
A3      0
A4      6
A5      6
A6      9
A7      9
A8      0
A9      0
A10     0
A11     0
A12     0
A13     0
A14    13
A15     0
A16     0
dtype: int64
```

2d)CODE AND OUTPUT :

```
[115] dfMode=df1
      imputer1 = SimpleImputer(missing_values = np.nan,strategy ='most_frequent')
```

```
[112] dfMode
```

```
SimpleImputer(strategy='most_frequent')
```

```
[117] imputer1=imputer1.fit(dfMode[['A2']])
```

```
[118] imputer1
```

```
SimpleImputer(strategy='most_frequent')
```

```
[119] dfMode['A2'] = imputer1.transform(dfMode[['A2']])
```

```
dfMode['A2']
```

```
0      30.83
1      58.67
2      24.50
3      27.83
4      20.17
       ...
685    21.08
686    22.67
687    25.25
688    17.92
689    35.00
Name: A2, Length: 690, dtype: float64
```

```
[121] dfMode['A3'] = imputer1.transform(dfMode[['A3']])
```

```
[123] dfMode['A3']
```

```
0       0.000
1       4.460
2       0.500
3       1.540
4       5.625
        ...
685    10.085
686     0.750
687    13.500
688     0.205
689     3.375
Name: A3, Length: 690, dtype: float64
```

```
dfMode['A8'] = imputer1.transform(dfMode[['A8']])
```

```
[125] dfMode['A8']
```

```
0      1.25
1      3.04
2      1.50
3      3.75
4      1.71
       ...
685    1.25
686    2.00
687    2.00
688    0.04
689    8.29
Name: A8, Length: 690, dtype: float64
```

```
dfMode['A11'] = imputer1.transform(dfMode[['A11']])
```

```
[127] dfMode['A11']
```

```
0      1.0
1      6.0
2      0.0
3      5.0
4      0.0
      ...
685    0.0
686    2.0
687    1.0
688    0.0
689    0.0
Name: A11, Length: 690, dtype: float64
```

```
[128] dfMode['A15'] = imputer1.transform(dfMode[['A15']])
```

```
[129] dfMode['A15']
```

```
0        0.0
1      560.0
2      824.0
3        3.0
4        0.0
       ...
685      0.0
686    394.0
687      1.0
688    750.0
689      0.0
Name: A15, Length: 690, dtype: float64
```

```
dfMode.isnull().sum()
```

```
A1     12
A2      0
A3      0
A4      6
A5      6
A6      9
A7      9
A8      0
A9      0
A10     0
A11     0
A12     0
A13     0
A14    13
A15     0
A16     0
dtype: int64
```

2e) CODE AND OUTPUT:

```
df1=pd.read_csv('/content/ Transformed_crx.csv')
```

```
[132] df1
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

690 rows × 16 columns

```
[133] df1.isnull().sum()
```

```
A1     12
A2     12
A3      0
A4      6
A5      6
A6      9
A7      9
A8      0
A9      0
A10     0
A11     0
A12     0
A13     0
A14    13
A15     0
A16     0
dtype: int64
```

```
[134] df1=pd.read_csv('/content/ Transformed_crx.csv')
```

```
df1
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

690 rows × 16 columns

+ Code + Text

```
dftestA2=df1[df1['A2'].isnull()]
```

```
[137] dftestA2
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
83	a	NaN	3.500	u	g	d	v	3.000	t	f	0	t	g	300.0	0	0
86	b	NaN	0.375	u	g	d	v	0.875	t	f	0	t	s	928.0	0	0
92	b	NaN	5.000	y	p	aa	v	8.500	t	f	0	f	g	0.0	0	0
97	b	NaN	0.500	u	g	c	bb	0.835	t	f	0	t	s	320.0	0	0
254	b	NaN	0.625	u	g	k	v	0.250	f	f	0	f	g	380.0	2010	0
286	a	NaN	1.500	u	g	ff	ff	0.000	f	t	2	t	g	200.0	105	0
329	b	NaN	4.000	y	p	i	v	0.085	f	f	0	t	g	411.0	0	0
445	a	NaN	11.250	u	g	ff	ff	0.000	f	f	0	f	g	NaN	5200	0
450	b	NaN	3.000	y	p	i	bb	7.000	f	f	0	f	g	0.0	1	0
500	b	NaN	4.000	u	g	x	v	5.000	t	t	3	t	g	290.0	2279	1
515	b	NaN	10.500	u	g	x	v	6.500	t	f	0	f	g	0.0	0	1
608	b	NaN	0.040	y	p	d	v	4.250	f	f	0	t	g	460.0	0	0

```
df2=df1.dropna(inplace=False)
```

```
df2
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	202.0	0	1
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	43.0	560	1
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	280.0	824	1
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	100.0	3	1
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	120.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	b	21.08	10.085	y	p	e	h	1.25	f	f	0	f	g	260.0	0	0
686	a	22.67	0.750	u	g	c	v	2.00	f	t	2	t	g	200.0	394	0
687	a	25.25	13.500	y	p	ff	ff	2.00	f	t	1	t	g	200.0	1	0
688	b	17.92	0.205	u	g	aa	v	0.04	f	f	0	f	g	280.0	750	0
689	b	35.00	3.375	u	g	c	h	8.29	f	f	0	t	g	0.0	0	0

653 rows × 16 columns

```
[140] xtrain=df2[['A3','A8']]
      ytrain=df2[['A2']]
```

[141] xtrain

	A3	A8
0	0.000	1.25
1	4.460	3.04
2	0.500	1.50
3	1.540	3.75
4	5.625	1.71
...	...	...
685	10.085	1.25
686	0.750	2.00
687	13.500	2.00
688	0.205	0.04
689	3.375	8.29

653 rows × 2 columns

[142] ytrain

	A2
0	30.83
1	58.67
2	24.50
3	27.83
4	20.17
...	...
685	21.08
686	22.67
687	25.25
688	17.92
689	35.00

653 rows × 1 columns

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

[144] lr

```
LinearRegression()
```

```
[146] lr.fit(xtrain,ytrain)
```

```
LinearRegression()
```

```
[148] xtest=dftestA2[['A3','A8']]
```

```
xtest
```

	A3	A8
83	3.500	3.000
86	0.375	0.875
92	5.000	8.500
97	0.500	0.835
254	0.625	0.250
286	1.500	0.000
329	4.000	0.085
445	11.250	0.000
450	3.000	7.000
500	4.000	5.000
515	10.500	6.500
608	0.040	4.250

```
[150] ytest=dftestA2[['A2']]
```

```
ytest
```

	A2
83	32.214229
86	28.579438
92	40.050457
97	28.554865
254	27.789247
286	27.658028
329	28.369919
445	29.983657
450	37.533830
500	35.052924
515	38.642917
608	33.088574

```
[152] dftestA2[['A2']]=lr.predict(xtest)
```

```
dftestA2[['A2']]
```

	A2
83	32.214229
86	28.579438
92	40.050457
97	28.554865
254	27.789247
286	27.658028
329	28.369919
445	29.983657
450	37.533830
500	35.052924
515	38.642917
608	33.088574

```
[154] print("Attributes 'A3', 'A8', 'A11', 'A15' have no NAN values")
```

```
Attributes 'A3', 'A8', 'A11', 'A15' have no NAN values
```

```
print(df1.isnull().sum())
```

```
A1     12
A2     12
A3      0
A4      6
A5      6
A6      9
A7      9
A8      0
A9      0
A10     0
A11     0
A12     0
A13     0
A14    13
A15     0
A16     0
dtype: int64
```