Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE & COMMERCE

(EMPOWERED AUTONOMOUS)

Machine Learning Journal

Name : Dheeraj Mishra
Roll No: 712
Class : MSc Data Science and Artificial Intelligence part I

Ramniranjan Jhunjhunwala College of Arts, Science and Commerce

Department of Data Science and Artificial Intelligence

CERTIFICATE

This is to certify Dheeraj Mishra of Msc. Data Science and Artificial Intelligence Roll No 712 has successfully completed the practical of Machine Learning during the Academic Year 2023-2024.

Date :

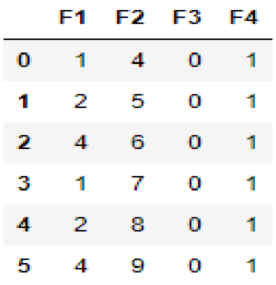
(Prof. Rahul Yadav) **External Examiner**

Prof-In-Charge

INDEX

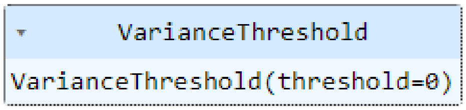
Practical 1

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
df = pd.DataFrame({"F1":[1,2,4,1,2,4],
                   "F2":[4,5,6,7,8,9],
                   "F3":[0,0,0,0,0,0],
                   "F4":[1,1,1,1,1,1]})
df
```



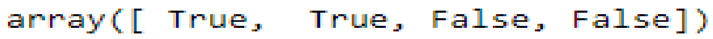
	F1	F2	F3	F4
0	1	4	0	1
1	2	5	0	1
2	4	6	0	1
3	1	7	0	1
4	2	8	0	1
5	4	9	0	1

```
from sklearn.feature_selection import VarianceThreshold
var_thres = VarianceThreshold(threshold=0)
var_thres.fit(df)
```



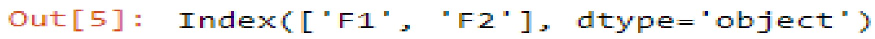
```
VarianceThreshold
VarianceThreshold(threshold=0)
```

```
var_thres.get_support()
```



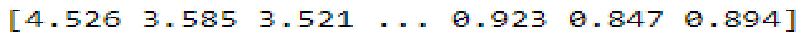
```
array([ True,  True, False, False])
```

```
df.columns[var_thres.get_support()]
```



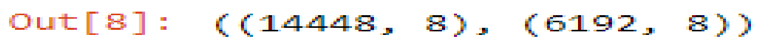
```
Out[5]: Index(['F1', 'F2'], dtype='object')
```

```
from sklearn.datasets import fetch_california_housing
import matplotlib.pyplot as plt
%matplotlib inline
df = fetch_california_housing()
x = pd.DataFrame(df.data,columns = df.feature_names)
y = df.target
print(y)
```



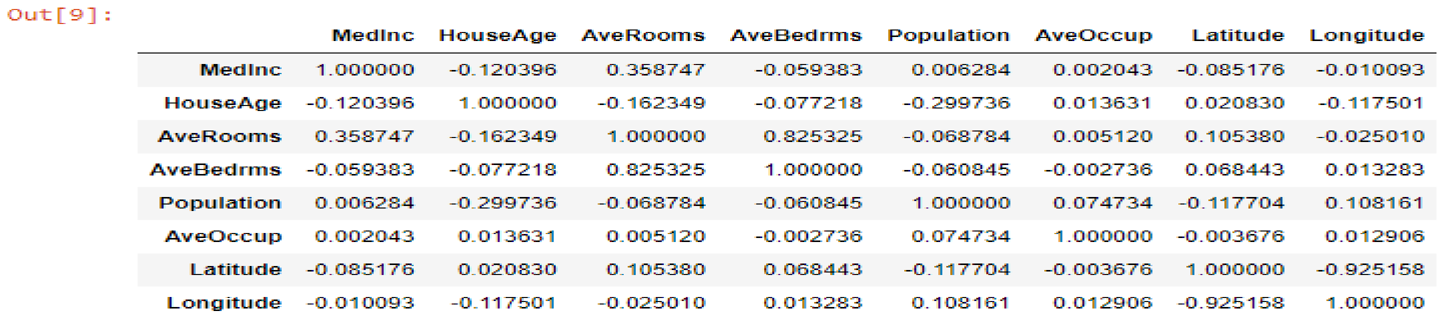
```
[4.526 3.585 3.521 ... 0.923 0.847 0.894]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x,y,test_size=0.3,random_state=0)
X_train.shape, X_test.shape
```



```
Out[8]: ((14448, 8), (6192, 8))
```

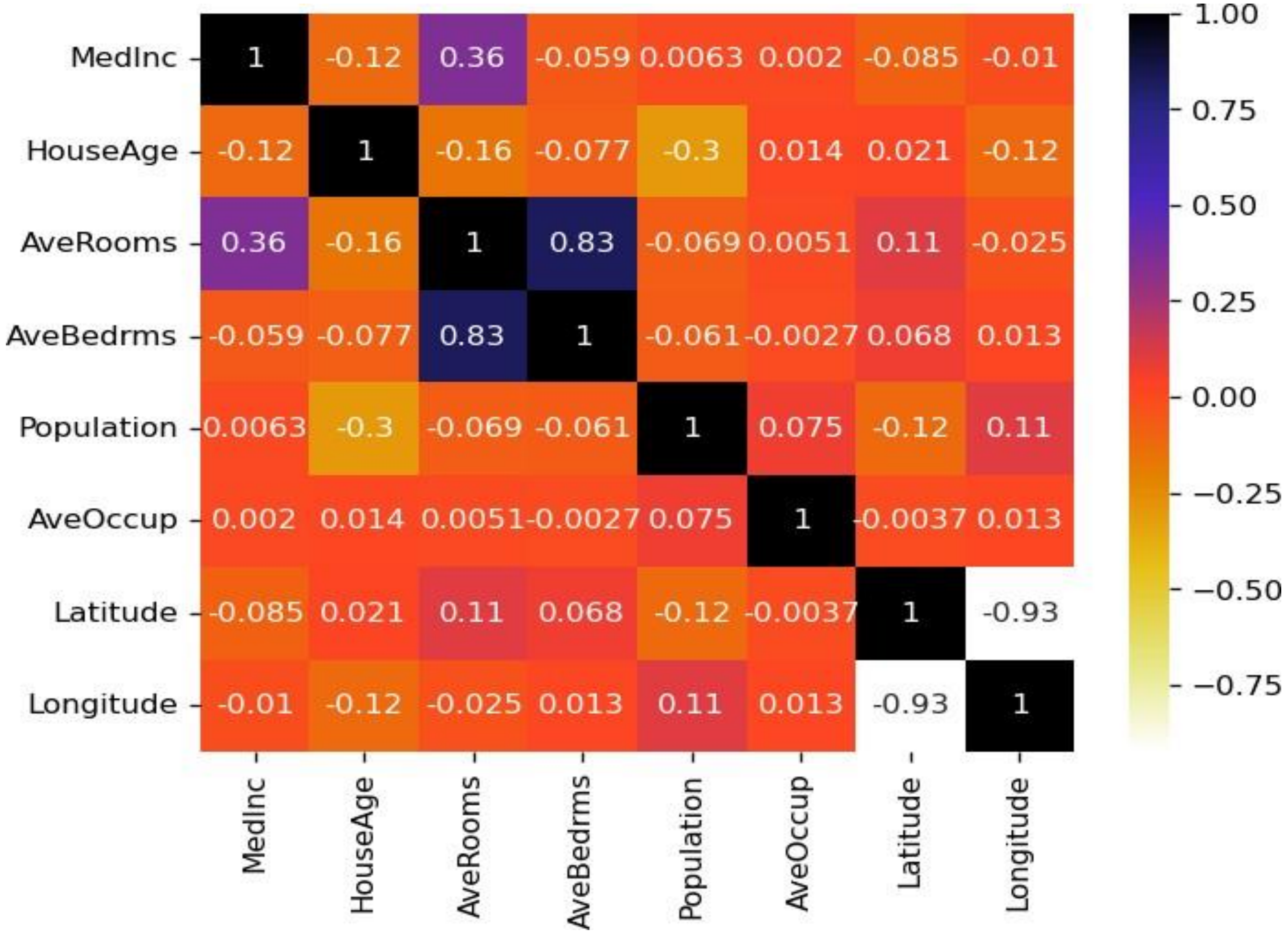
```
X_train.corr()
```



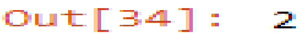
Out[9]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
MedInc	1.000000	-0.120396	0.358747	-0.059383	0.006284	0.002043	-0.085176	-0.010093
HouseAge	-0.120396	1.000000	-0.162349	-0.077218	-0.299736	0.013631	0.020830	-0.117501
AveRooms	0.358747	-0.162349	1.000000	0.825325	-0.068784	0.005120	0.105380	-0.025010
AveBedrms	-0.059383	-0.077218	0.825325	1.000000	-0.060845	-0.002736	0.068443	0.013283
Population	0.006284	-0.299736	-0.068784	-0.060845	1.000000	0.074734	-0.117704	0.108161
AveOccup	0.002043	0.013631	0.005120	-0.002736	0.074734	1.000000	-0.003676	0.012906
Latitude	-0.085176	0.020830	0.105380	0.068443	-0.117704	-0.003676	1.000000	-0.925158
Longitude	-0.010093	-0.117501	-0.025010	0.013283	0.108161	0.012906	-0.925158	1.000000

```
import seaborn as sns
plt.figure(dpi=120)
cor = X_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```

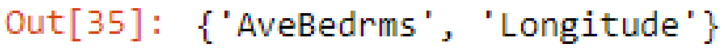


```
def correlation(dataset,threshold):
    col_corr = set()     #set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: #we are intrested in absolute coeff values
                colname = corr_matrix.columns[i]     #getting the name of the column
                col_corr.add(colname)
    return col_corr
corr_features = correlation(X_train, 0.7)
len(set(corr_features))
```



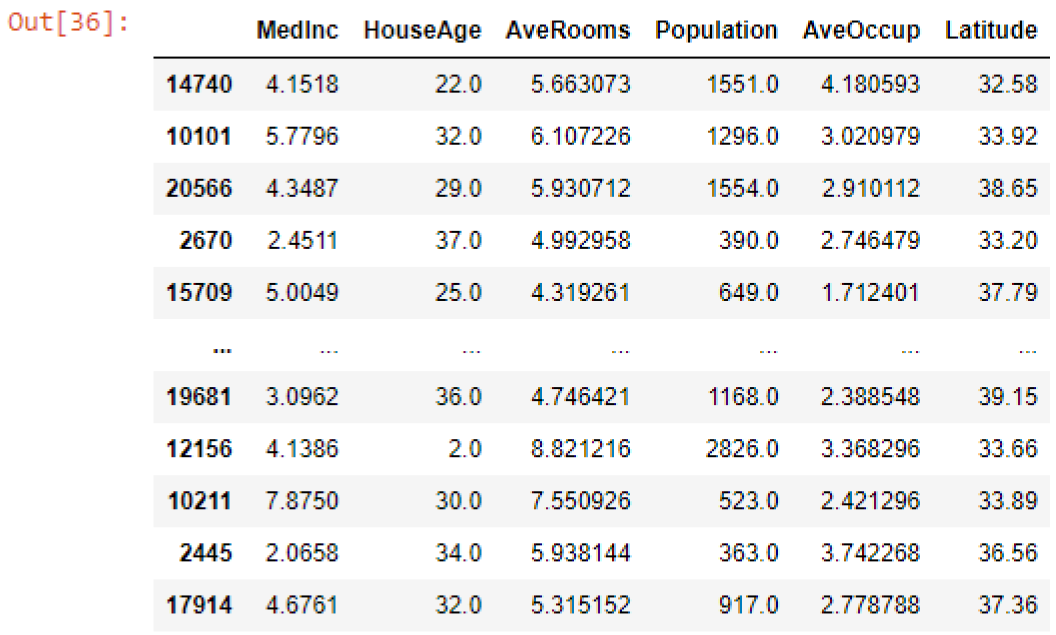
```
Out[34]:  2
```

```
corr_features
```



```
Out[35]:  {'AveBedrms', 'Longitude'}
```

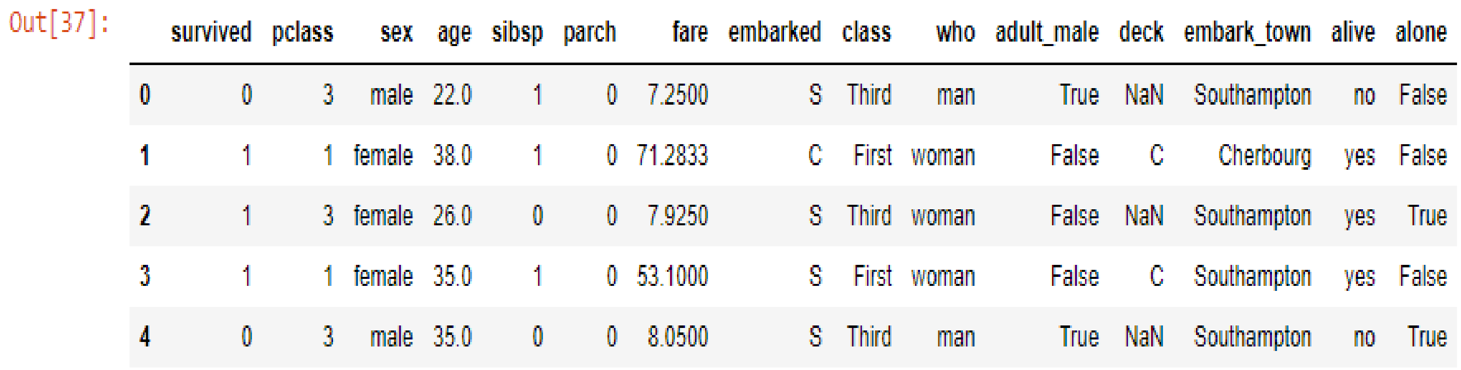
```
X_train.drop(corr_features, axis=1)
X_test.drop(corr_features, axis=1)
```



Out[36]:

	MedInc	HouseAge	AveRooms	Population	AveOccup	Latitude
14740	4.1518	22.0	5.663073	1551.0	4.180593	32.58
10101	5.7796	32.0	6.107226	1296.0	3.020979	33.92
20566	4.3487	29.0	5.930712	1554.0	2.910112	38.65
2670	2.4511	37.0	4.992958	390.0	2.746479	33.20
15709	5.0049	25.0	4.319261	649.0	1.712401	37.79
...	...	...	...	...	...	...
19681	3.0962	36.0	4.746421	1168.0	2.388548	39.15
12156	4.1386	2.0	8.821216	2826.0	3.368296	33.66
10211	7.8750	30.0	7.550926	523.0	2.421296	33.89
2445	2.0658	34.0	5.938144	363.0	3.742268	36.56
17914	4.6761	32.0	5.315152	917.0	2.778788	37.36

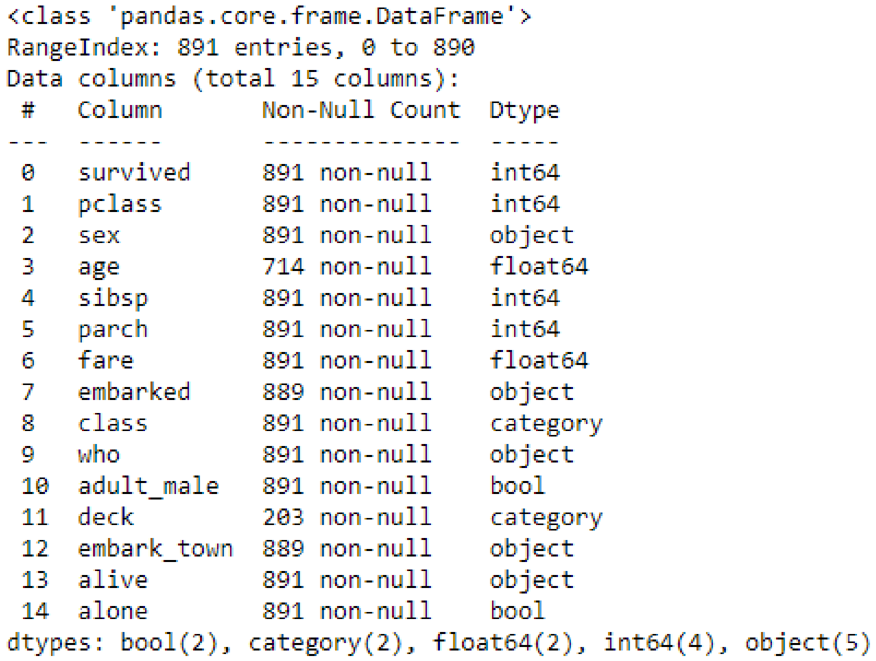
```
import seaborn as sns
import numpy as np
df=sns.load_dataset('titanic')
df.head()
```



Out[37]:

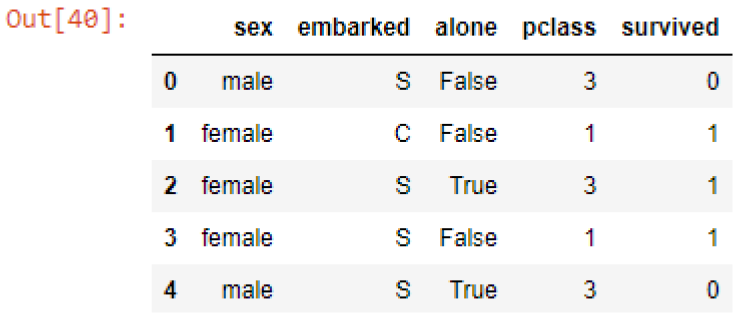
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
```

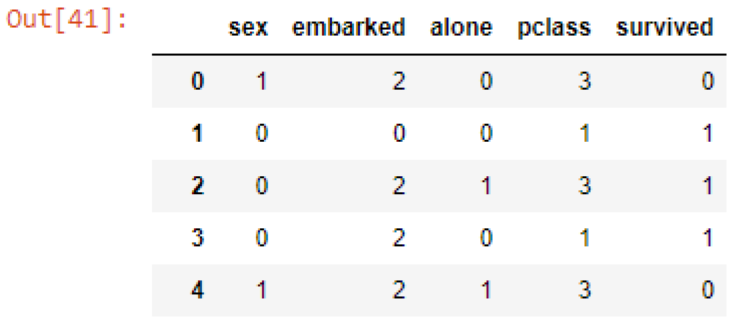
```
df = df[['sex','embarked','alone','pclass','survived']]
df.head()
```



Out[40]:

	sex	embarked	alone	pclass	survived
0	male	S	False	3	0
1	female	C	False	1	1
2	female	S	True	3	1
3	female	S	False	1	1
4	male	S	True	3	0

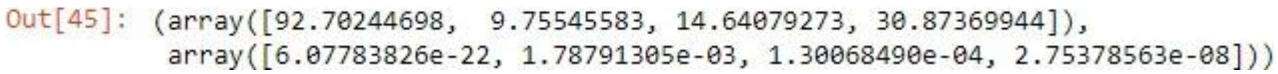
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['embarked'] = le.fit_transform(df['embarked'])
df['alone'] = le.fit_transform(df['alone'])
df['sex'] = le.fit_transform(df['sex'])
df
```



Out[41]:

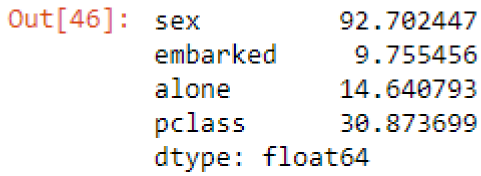
	sex	embarked	alone	pclass	survived
0	1	2	0	3	0
1	0	0	0	1	1
2	0	2	1	3	1
3	0	2	0	1	1
4	1	2	1	3	0

```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
from sklearn.feature_selection import chi2
f_p_values = chi2(x,y)
f_p_values
```



```
Out[45]: (array([92.70244698,  9.75545583, 14.64079273, 30.87369944]),
          array([6.07783826e-22, 1.78791305e-03, 1.30068490e-04, 2.75378563e-08]))
```

```
import pandas as pd
p_values=pd.Series(f_p_values[0])
p_values.index=x.columns
p_values
```



```
Out[46]: sex         92.702447
         embarked     9.755456
         alone       14.640793
         pclass      30.873699
         dtype: float64
```

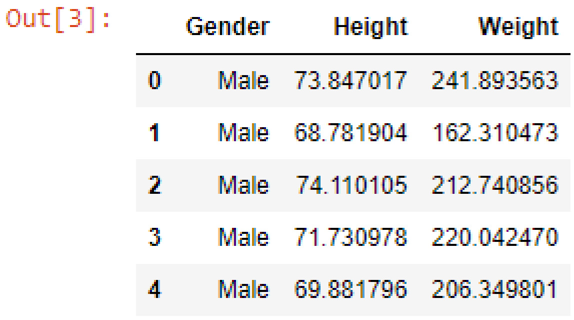
Signature:____________

Practical 2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

SIMPLE LINEAR REGRESSION

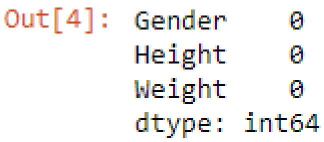
```
data=pd.read_csv(r"C:\Users\rahul\Downloads\archive (2)\weight-height.csv")
data.head()
```



Out[3]:

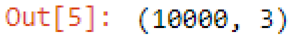
	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

```
data.isnull().sum()
```



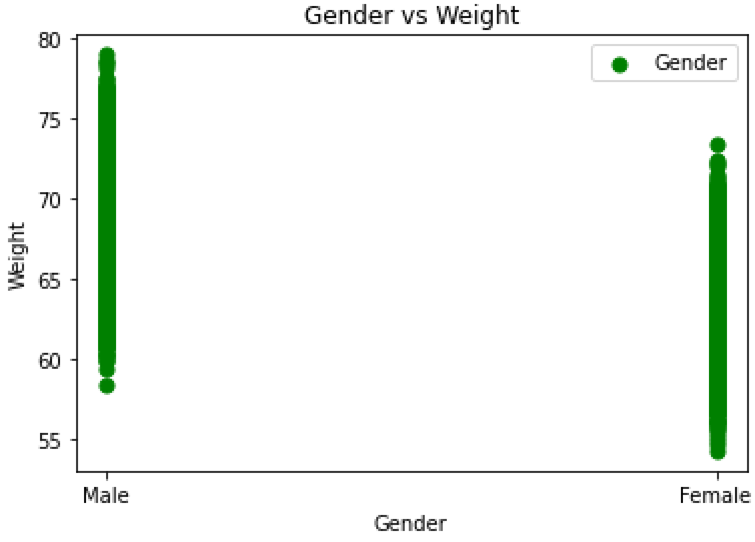
```
Out[4]: Gender    0
        Height    0
        Weight    0
        dtype: int64
```

```
Data.shape
```

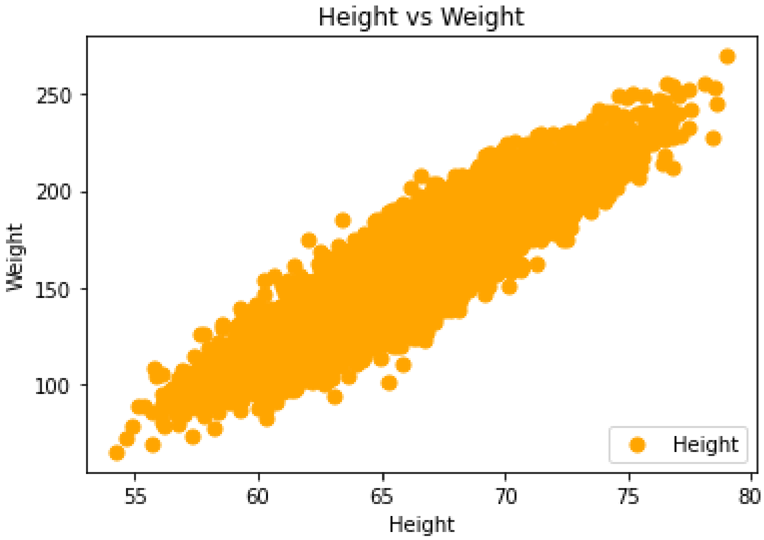


```
Out[5]: (10000, 3)
```

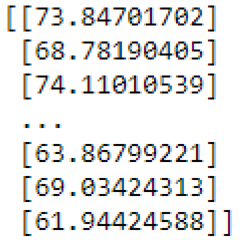
```
x1 = data.iloc[:, 0].values
y1 = data.iloc[:, 1].values
plt.scatter(x1,y1,label='Gender',color='Green',s=50)
plt.xlabel('Gender')
plt.ylabel('Weight')
plt.title('Gender vs Weight')
plt.legend()
plt.show()
```



```
x2 = data.iloc[:, 1].values
y2 = data.iloc[:, 2].values
plt.scatter(x2,y2,label='Height',color='Orange',s=50)
plt.xlabel('Height')
plt.ylabel('Weight')
plt.title('Height vs Weight')
plt.legend(loc="lower right")
plt.show()
```

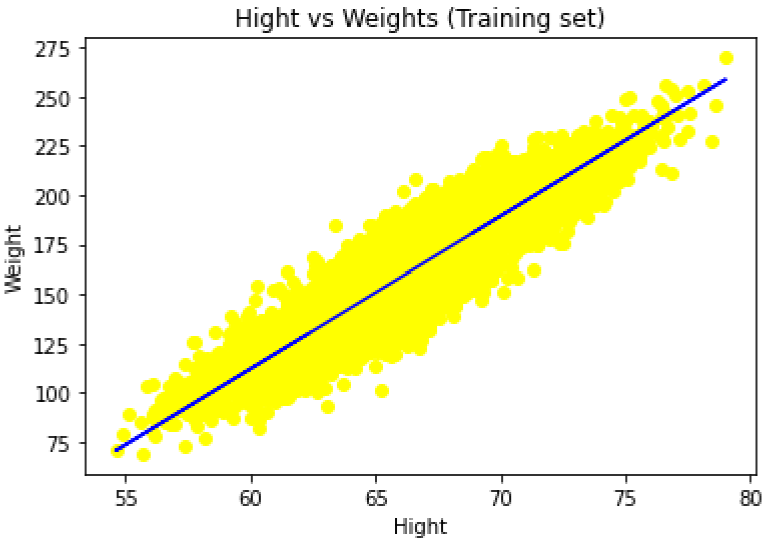


```
X = data.iloc[:, 1:2].values
y = data.iloc[:, 2].values
print(X)
```

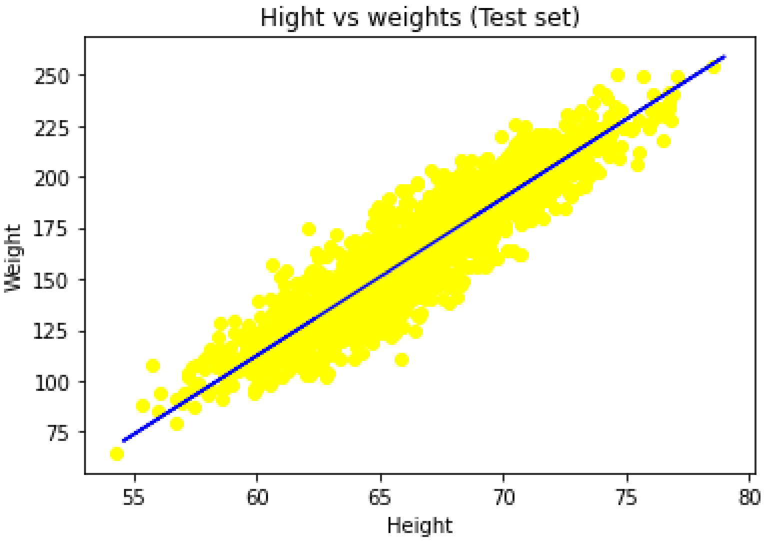


```
[[73.84701702]
 [68.78190405]
 [74.11010539]
 ...
 [63.86799221]
 [69.03424313]
 [61.94424588]]
```

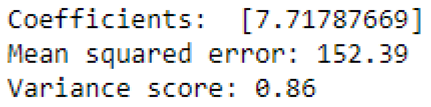
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
plt.scatter(X_train, y_train, color = 'Yellow')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Hight vs Weights (Training set)')
plt.xlabel('Hight')
plt.ylabel('Weight')
plt.show()
```



```
plt.scatter(X_test, y_test, color = 'Yellow')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Hight vs weights (Test set)')
plt.xlabel('Height')
plt.ylabel('Weight')
plt.show()
```



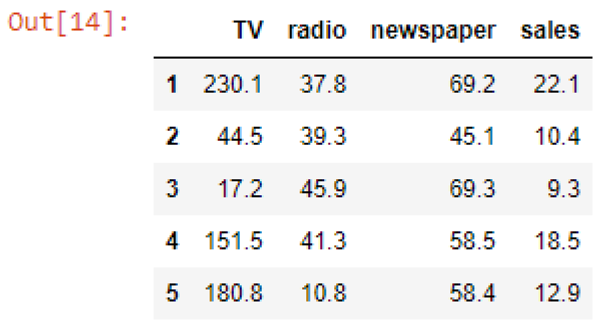
```
y_pred = regressor.predict(X_test)
print('Coefficients: ', regressor.coef_)
print("Mean squared error: %.2f" % np.mean((regressor.predict(X_test) - y_test) ** 2))
print('Variance score: %.2f' % regressor.score(X_test, y_test))
```



```
Coefficients:  [7.71787669]
Mean squared error: 152.39
Variance score: 0.86
```

MULTIPLE LINEAR REGRESSION

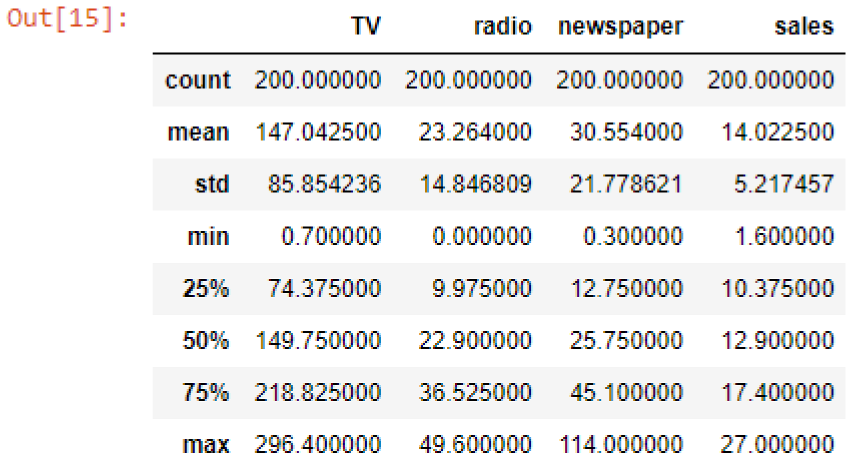
```
data=pd.read_csv(r"C:\Users\rahul\Downloads\Rahul\Advertising.csv",index_col=0,header=0)
data.head()
```



Out[14]:

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

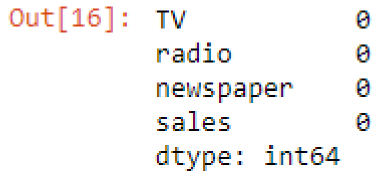
```
data.dtypes
data.shape
data.describe()
```



Out[15]:

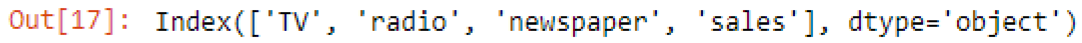
	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

```
data.isnull().sum()
```



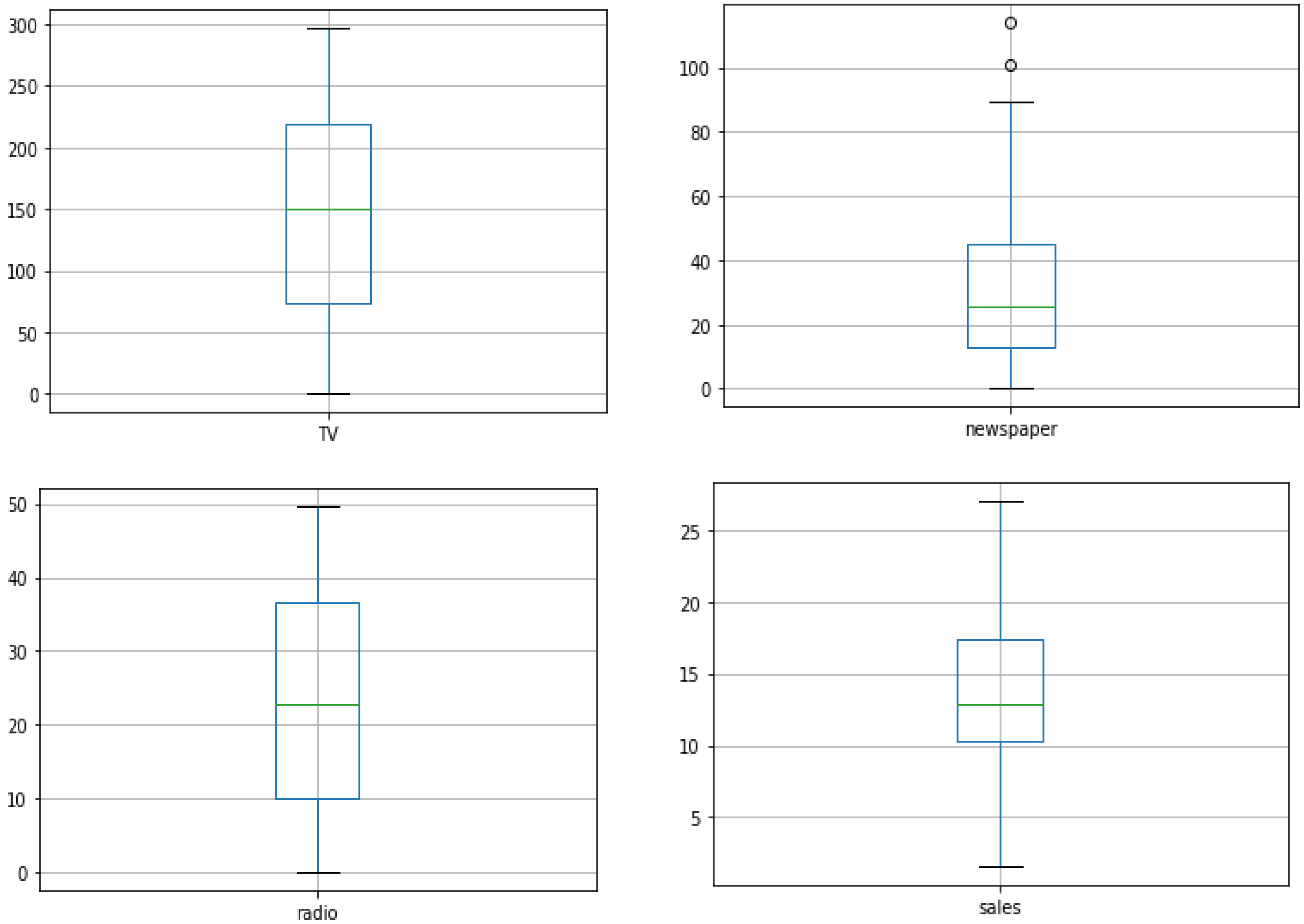
```
Out[16]: TV           0
         radio        0
         newspaper    0
         sales        0
         dtype: int64
```

```
data.columns
```

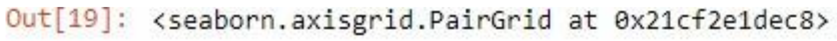


```
Out[17]: Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

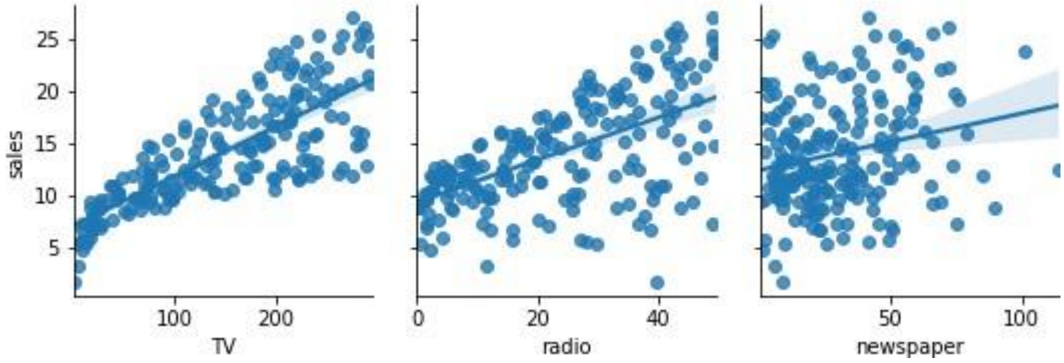
```
for i in data.columns:
    data.boxplot(column=i)
    plt.show()
```



```
sns.pairplot(data,x_vars=['TV','radio','newspaper'],
          y_vars="sales",kind='reg')
```



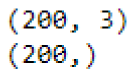
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x21cf2e1dec8>
```



```
data.columns
```

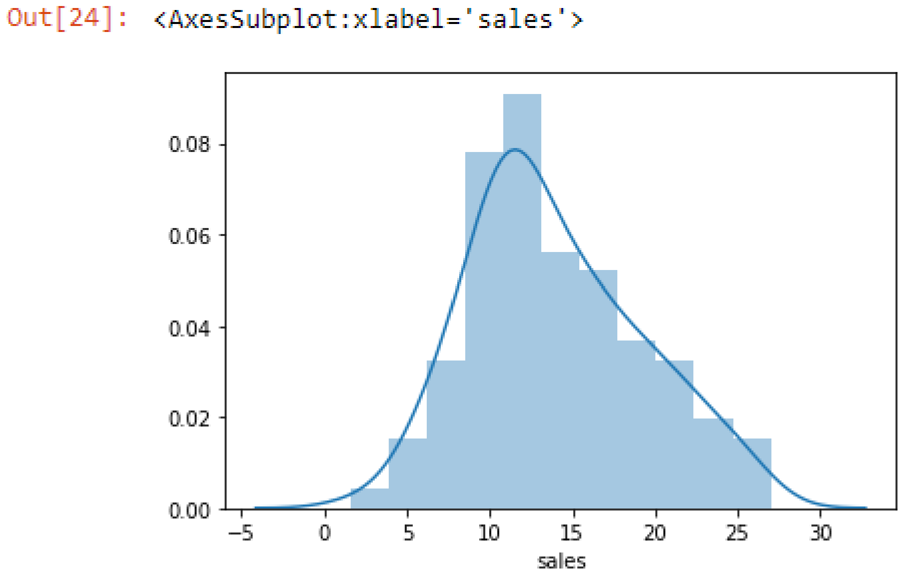
```
Out[20]: Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

```
X=data[['TV','radio','newspaper']]
Y=data['sales'
print(X.shape)
print(Y.shape)
```



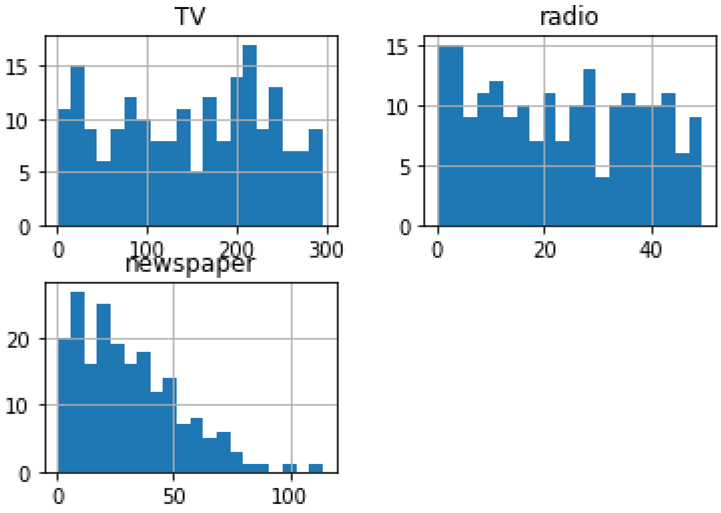
```
(200, 3)
(200,)
```

import warnings
warnings.filterwarnings("ignore")
sns.distplot(Y)



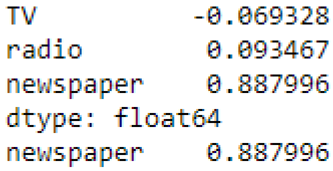
```
Out[24]: <AxesSubplot:xlabel='sales'>
```

X.hist(bins=20)



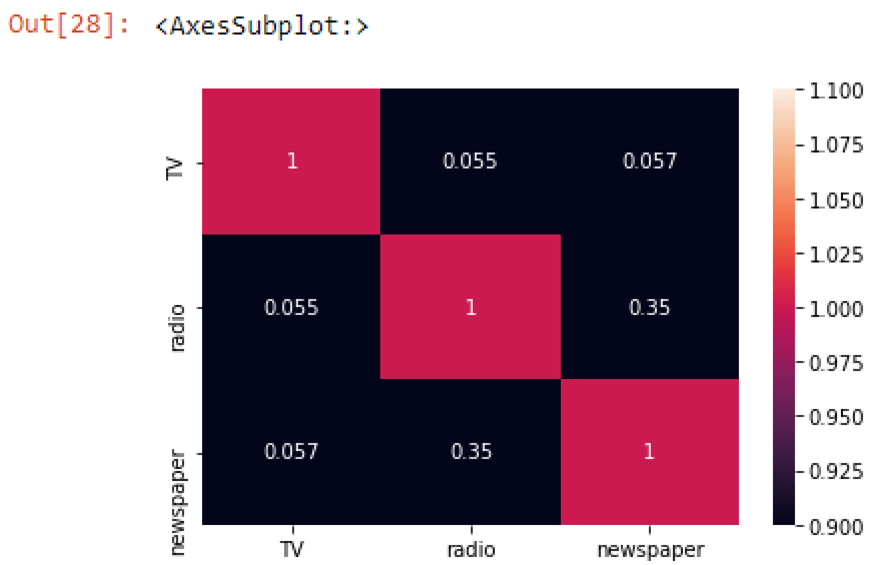
from scipy.stats import skew
data_num_skew = X.apply(lambda i: skew(i.dropna()))
data_num_skewed = data_num_skew[(data_num_skew > .75) | (data_num_skew < -.75)]

print(data_num_skew)
print(data_num_skewed)



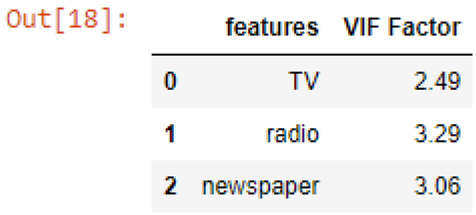
```
TV          -0.069328
radio        0.093467
newspaper    0.887996
dtype: float64
newspaper    0.887996
```

corr_df=X.corr(method="pearson")
print(corr_df)
sns.heatmap(corr_df,vmax=1.0,vmin=1.0,annot=True)



```
Out[28]: <AxesSubplot:>
```

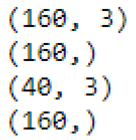
```
from statsmodels.stats.outliers_influence import variance_inflation_factor as vif

vif_df = pd.DataFrame()
vif_df["features"] = X.columns
vif_df["VIF Factor"] = [vif(X.values, i) for i in range(X.shape[1])]
vif_df.round(2)
```



Out[18]:

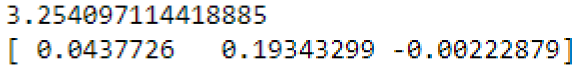
	features	VIF Factor
0	TV	2.49
1	radio	3.29
2	newspaper	3.06

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=10)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_train.shape)
```



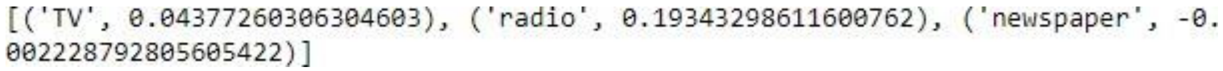
```
(160, 3)
(160,)
(40, 3)
(160,)
```

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(X_train,Y_train)
print(lm.intercept_)
print(lm.coef_)
```



```
3.254097114418885
[ 0.0437726   0.19343299 -0.00222879]
```

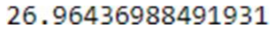
```
print(list(zip(X.columns,lm.coef_)))
```



```
[('TV', 0.04377260306304603), ('radio', 0.19343298611600762), ('newspaper', -0.
002228792805605422)]
```

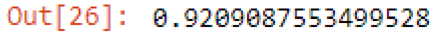
```
X1=100
X2=100
```

```
X3=np.log1p(100)
Y_pred=3.254097114418885+(0.04377260306304603*X1)+(0.19343298611600762*X2)+(-0.002228792805605
422*X3)
print(Y_pred)
```



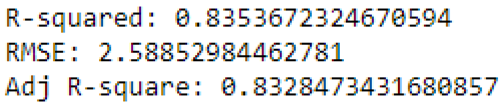
```
26.96436988491931
```

```
Y_pred=lm.predict(X_test)
print(Y_pred)
lm.score(X_train,Y_train)
```



```
Out[26]:  0.9209087553499528
```

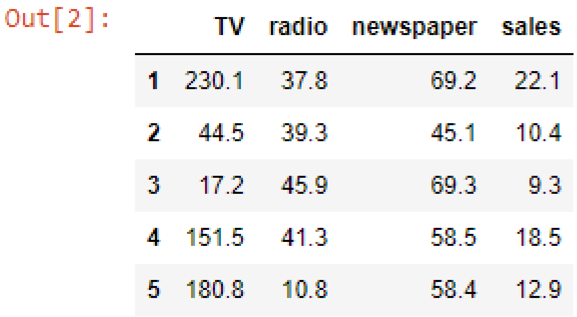
```
new_df=pd.DataFrame()
new_df=X_test.copy()
new_df["Actual sales"]=Y_test
new_df["Predicted sales"]=Y_pred
new_df

from sklearn.metrics import r2_score,mean_squared_error
import numpy as np

r2=r2_score(Y_test,Y_pred)
print("R-squared:",r2)

rmse=np.sqrt(mean_squared_error(Y_test,Y_pred))
print("RMSE:",rmse)

adjusted_r_squared = 1 - (1-r2)*(len(Y)-1)/(len(Y)-X.shape[1]-1)
print("Adj R-square:",adjusted_r_squared)
```



```
R-squared: 0.8353672324670594
RMSE: 2.58852984462781
Adj R-square: 0.8328473431680857
```

Practical 3

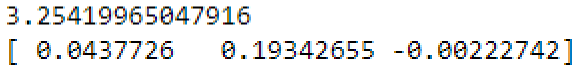
```
import pandas as pd
import numpy as np
data=pd.read_csv(r"C:\Users\rahul\Downloads\Rahul\Advertising.csv",index_col=0,header=0)
data.head()
```



Out[2]:

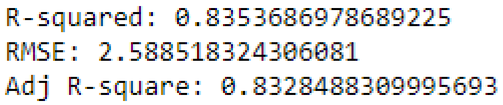
	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

```
X=data[['TV','radio','newspaper']]
Y=data['sales']
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=10)
from sklearn.linear_model import Ridge
lm=Ridge()
lm.fit(X_train,Y_train)
print(lm.intercept_)
print(lm.coef_)
```



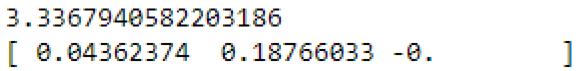
```
3.25419965047916
[ 0.0437726   0.19342655 -0.00222742]
```

```
Y_pred=lm.predict(X_test)
from sklearn.metrics import r2_score,mean_squared_error
import numpy as np

r2=r2_score(Y_test,Y_pred)
print("R-squared:",r2)

rmse=np.sqrt(mean_squared_error(Y_test,Y_pred))
print("RMSE:",rmse)

adjusted_r_squared = 1 - (1-r2)*(len(Y)-1)/(len(Y)-X.shape[1]-1)
print("Adj R-square:",adjusted_r_squared)
```



```
R-squared: 0.8353686978689225
RMSE: 2.588518324306081
Adj R-square: 0.8328488309995693
```

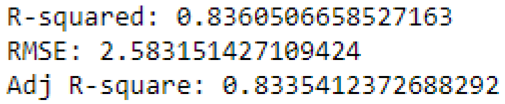
```
from sklearn.linear_model import Lasso
lm=Lasso()
lm.fit(X_train,Y_train)
#print intercept and coefficients
print(lm.intercept_)
print(lm.coef_)
```



```
3.3367940582203186
[ 0.04362374  0.18766033 -0.        ]
```

```
Y_pred=lm.predict(X_test)
```

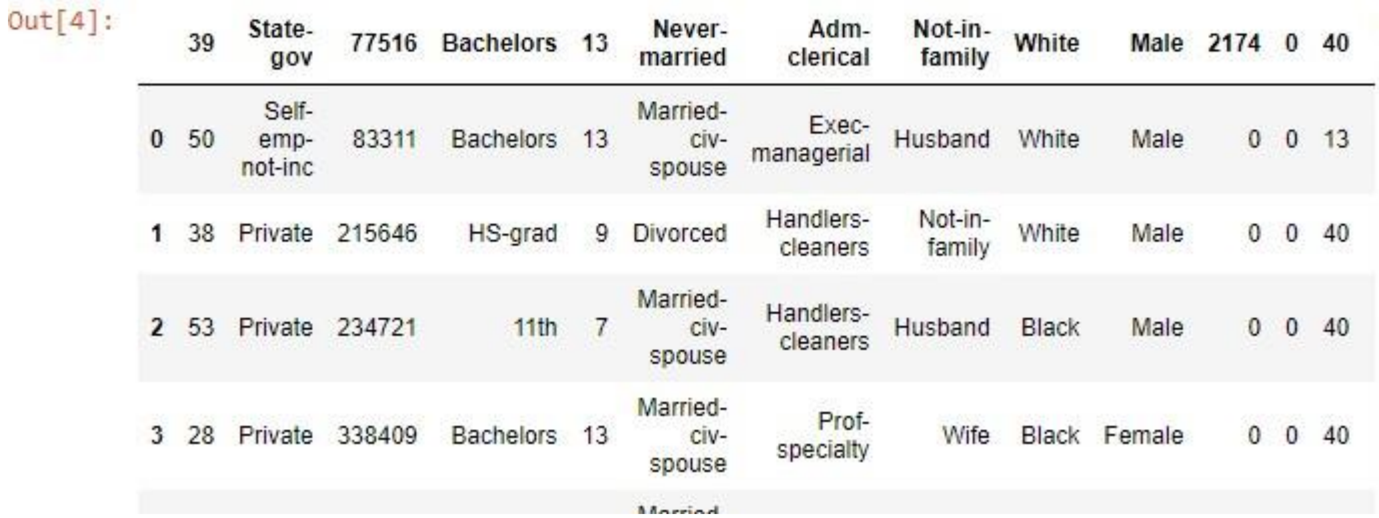
```
from sklearn.metrics import r2_score,mean_squared_error
import numpy as np

r2=r2_score(Y_test,Y_pred)
print("R-squared:",r2)

rmse=np.sqrt(mean_squared_error(Y_test,Y_pred))
print("RMSE:",rmse)

adjusted_r_squared = 1 - (1-r2)*(len(Y)-1)/(len(Y)-X.shape[1]-1)
print("Adj R-square:",adjusted_r_squared)
```



```
R-squared: 0.8360506658527163
RMSE: 2.583151427109424
Adj R-square: 0.8335412372688292
```

Practical 4

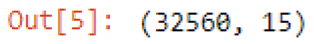
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\User38\Desktop\shraddha\adult_data - adult_data.csv")
df.head()
```



Out[4]:

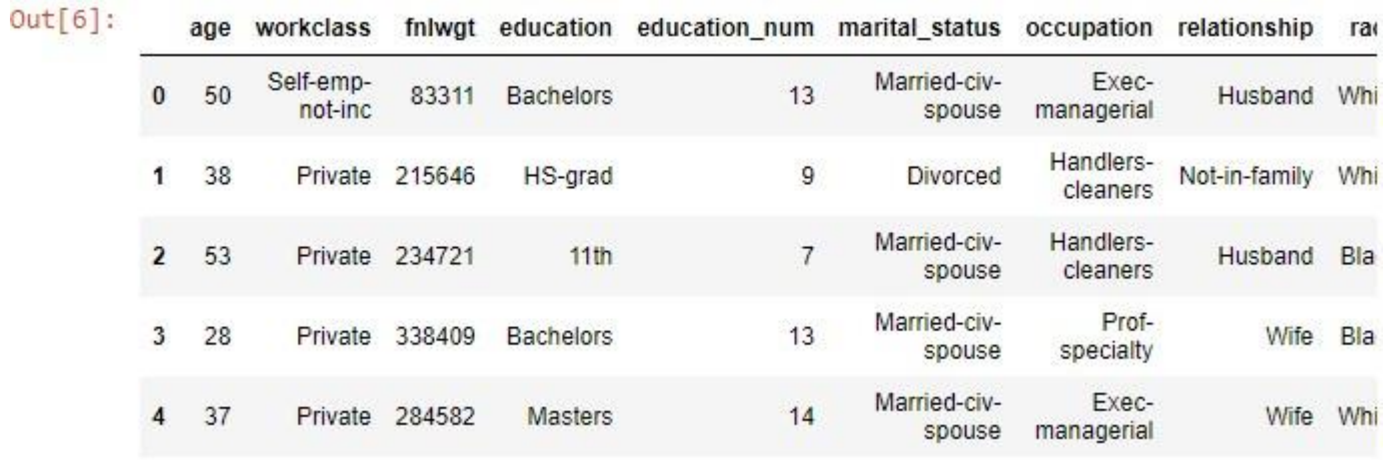
	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40

```
df.shape
```



```
Out[5]: (32560, 15)
```

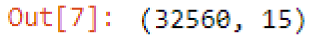
```
df.columns =
['age','workclass','fnlwgt','education','education_num','marital_status','occupation','relationship','race','sex',
           'capital_gain','capital_loss','hours_per_week','native_country','income']
df.head()
```



Out[6]:

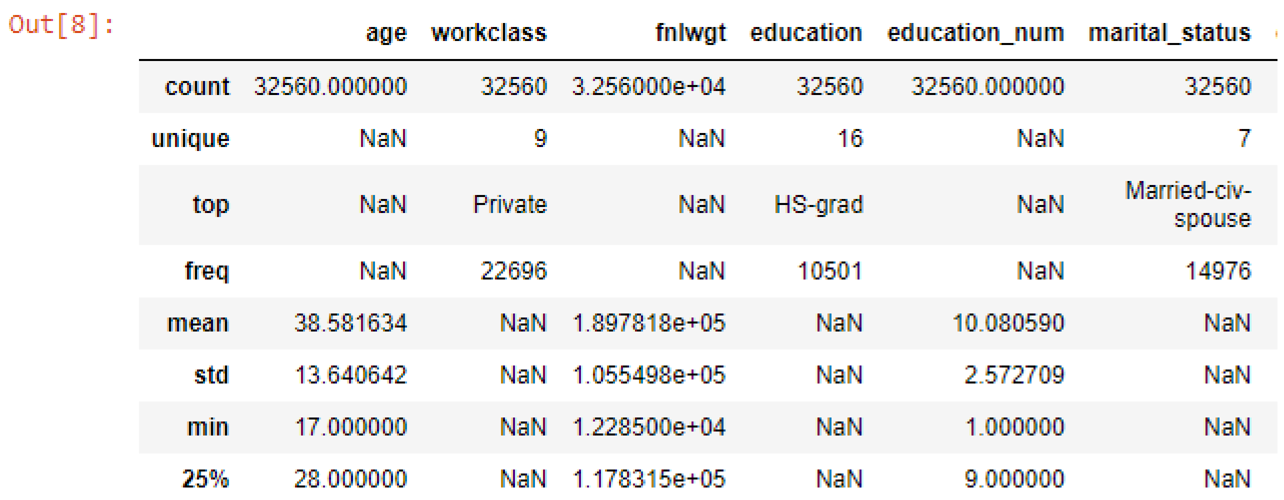
	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	ra
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Whi
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Whi
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Bla
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Bla
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Whi

```
df.shape
```



```
Out[7]: (32560, 15)
```

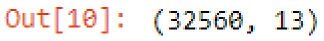
```
df.describe(include='all')
```



Out[8]:

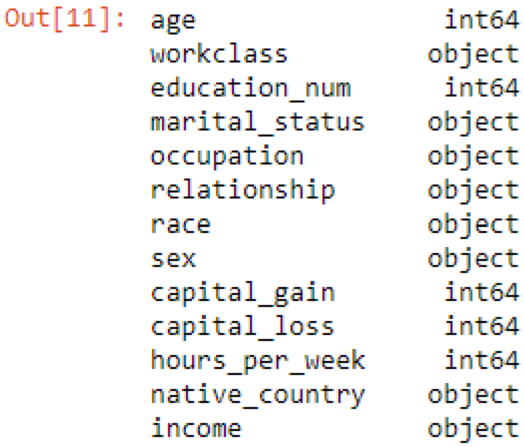
	age	workclass	fnlwgt	education	education_num	marital_status
count	32560.000000	32560	3.256000e+04	32560	32560.000000	32560
unique	NaN	9	NaN	16	NaN	7
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse
freq	NaN	22696	NaN	10501	NaN	14976
mean	38.581634	NaN	1.897818e+05	NaN	10.080590	NaN
std	13.640642	NaN	1.055498e+05	NaN	2.572709	NaN
min	17.000000	NaN	1.228500e+04	NaN	1.000000	NaN
25%	28.000000	NaN	1.178315e+05	NaN	9.000000	NaN

```
df2 = pd.DataFrame.copy(df)
df2.drop(["education","fnlwgt"],axis=1,inplace = True)
df2.shape
```



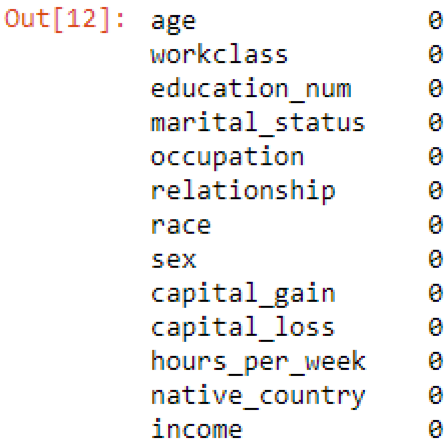
```
Out[10]:  (32560, 13)
```

```
df2.dtypes
```



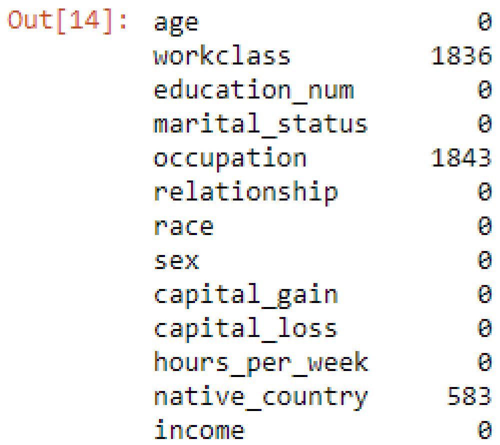
```
Out[11]:  age                int64
          workclass         object
          education_num      int64
          marital_status    object
          occupation        object
          relationship      object
          race              object
          sex               object
          capital_gain       int64
          capital_loss       int64
          hours_per_week     int64
          native_country    object
          income            object
```

```
df2.isnull().sum()
```



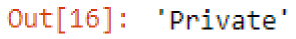
```
Out[12]:  age               0
          workclass         0
          education_num     0
          marital_status    0
          occupation        0
          relationship      0
          race              0
          sex               0
          capital_gain      0
          capital_loss      0
          hours_per_week    0
          native_country    0
          income            0
```

```
df2.replace('?',np.nan,inplace=True)
df2.isnull().sum()
```



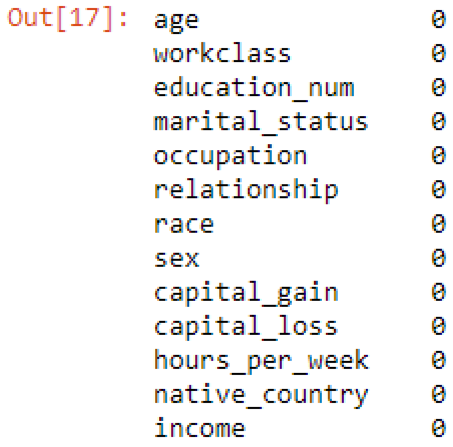
```
Out[14]: age                  0
         workclass         1836
         education_num        0
         marital_status       0
         occupation        1843
         relationship         0
         race                 0
         sex                  0
         capital_gain         0
         capital_loss         0
         hours_per_week       0
         native_country     583
         income               0
```

```
for value in ['workclass','occupation','native_country']:
    df2[value].fillna(df2[value].mode()[0],inplace=True)
df2.workclass.mode()[0]
```



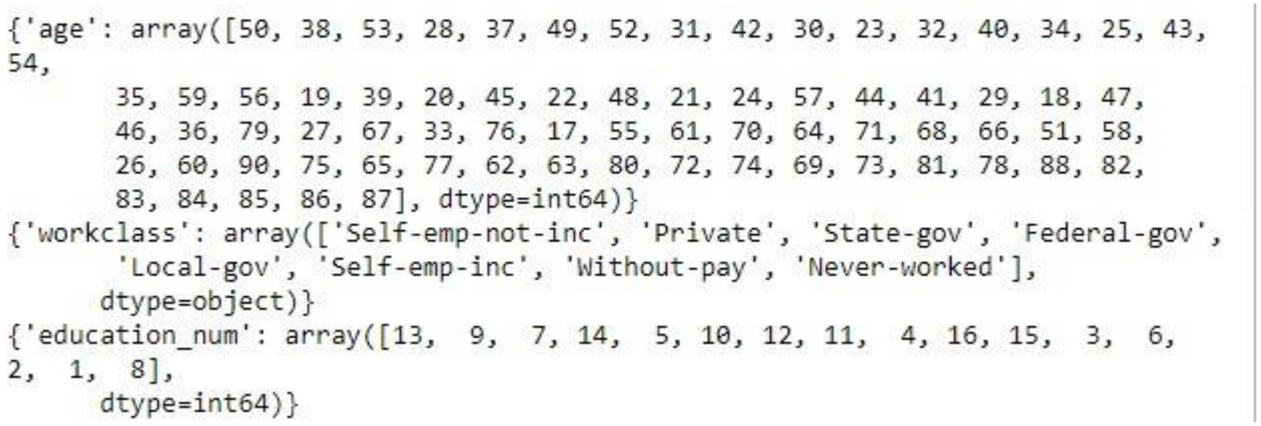
```
Out[16]: 'Private'
```

```
df2.isnull().sum()
```



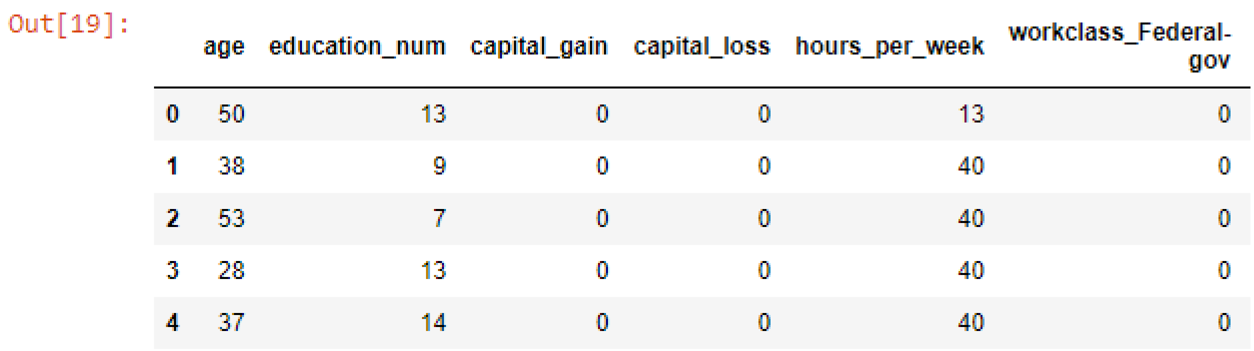
```
Out[17]: age               0
         workclass         0
         education_num     0
         marital_status    0
         occupation        0
         relationship      0
         race              0
         sex               0
         capital_gain      0
         capital_loss      0
         hours_per_week    0
         native_country    0
         income            0
```

```
for i in df2.columns:
    print({i:df2[i].unique()})
```



```
{'age': array([50, 38, 53, 28, 37, 49, 52, 31, 42, 30, 23, 32, 40, 34, 25, 43,
54,
       35, 59, 56, 19, 39, 20, 45, 22, 48, 21, 24, 57, 44, 41, 29, 18, 47,
       46, 36, 79, 27, 67, 33, 76, 17, 55, 61, 70, 64, 71, 68, 66, 51, 58,
       26, 60, 90, 75, 65, 77, 62, 63, 80, 72, 74, 69, 73, 81, 78, 88, 82,
       83, 84, 85, 86, 87], dtype=int64)}
{'workclass': array(['Self-emp-not-inc', 'Private', 'State-gov', 'Federal-gov',
       'Local-gov', 'Self-emp-inc', 'Without-pay', 'Never-worked'],
      dtype=object)}
{'education_num': array([13,  9,  7, 14,  5, 10, 12, 11,  4, 16, 15,  3,  6,
2,  1,  8],
      dtype=int64)}
```

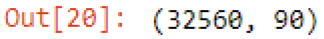
```
df2_new= pd.get_dummies(df2)
df2_new.head()
```



Out[19]:

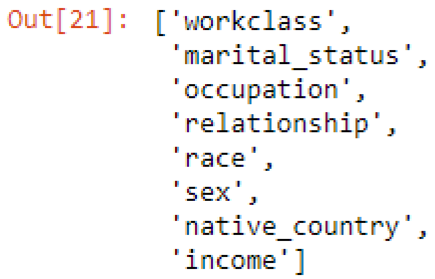
	age	education_num	capital_gain	capital_loss	hours_per_week	workclass_Federal-gov
0	50	13	0	0	13	0
1	38	9	0	0	40	0
2	53	7	0	0	40	0
3	28	13	0	0	40	0
4	37	14	0	0	40	0

```
df2_new.shape
```



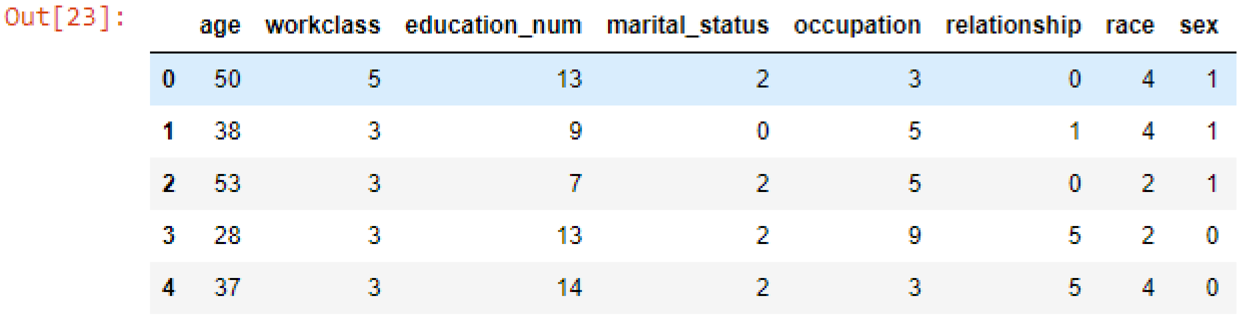
```
Out[20]: (32560, 90)
```

```
colname=[]
for x in df2.columns:
    if df2[x].dtype == 'object':
        colname.append(x)
colname
```



```
Out[21]: ['workclass',
          'marital_status',
          'occupation',
          'relationship',
          'race',
          'sex',
          'native_country',
          'income']
```

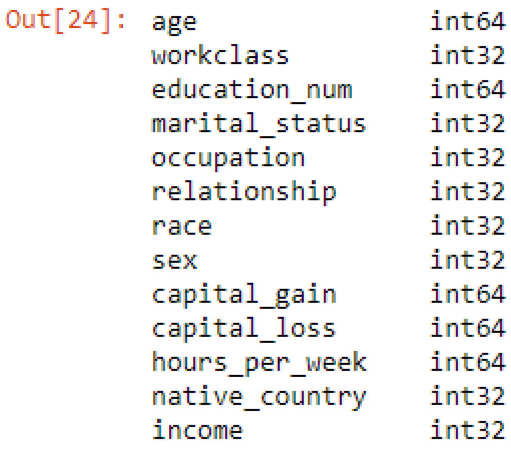
```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

for x in colname:
    df2[x] = le.fit_transform(df2[x])
df2.head()
```



Out[23]:

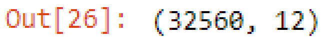
	age	workclass	education_num	marital_status	occupation	relationship	race	sex
0	50	5	13	2	3	0	4	1
1	38	3	9	0	5	1	4	1
2	53	3	7	2	5	0	2	1
3	28	3	13	2	9	5	2	0
4	37	3	14	2	3	5	4	0

```
df2.dtypes
```



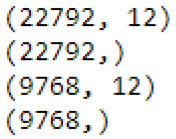
```
Out[24]: age               int64
         workclass         int32
         education_num     int64
         marital_status    int32
         occupation        int32
         relationship      int32
         race              int32
         sex               int32
         capital_gain      int64
         capital_loss      int64
         hours_per_week    int64
         native_country    int32
         income            int32
```

```
X = df2.values[:,0:-1]
Y = df2.values[:,-1]
X.shape
```



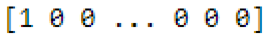
```
Out[26]: (32560, 12)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
print(X)

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size= 0.3, random_state = 10)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```



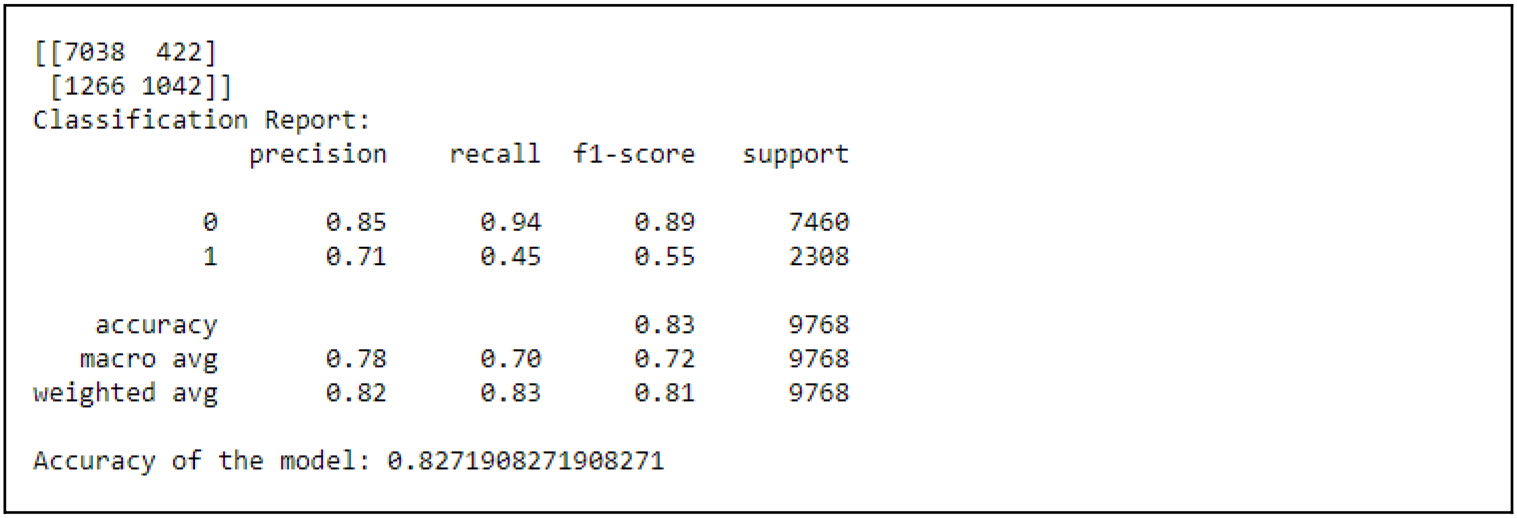
```
(22792, 12)
(22792,)
(9768, 12)
(9768,)
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(x_train,y_train)
Y_pred=classifier.predict(x_test)
print(Y_pred)
```



```
[1 0 0 ... 0 0 0]
```

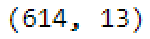
```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
cfm=confusion_matrix(y_test,Y_pred)
print(cfm)

print("Classification Report: ")

print(classification_report(y_test,Y_pred))

acc = accuracy_score(y_test,Y_pred)
print("Accuracy of the model:", acc)
```



```
[[7038  422]
 [1266 1042]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.94      0.89      7460
           1       0.71      0.45      0.55      2308

    accuracy                           0.83      9768
   macro avg       0.78      0.70      0.72      9768
weighted avg       0.82      0.83      0.81      9768

Accuracy of the model: 0.8271908271908271
```

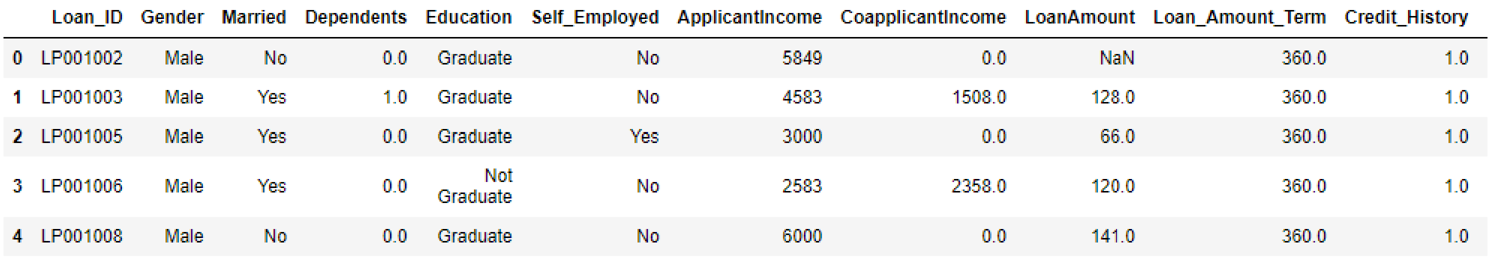
Practical 5

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv(r"C:\Users\User38\Desktop\shraddha\risk_analytics_train.csv")
print(df.shape)
```



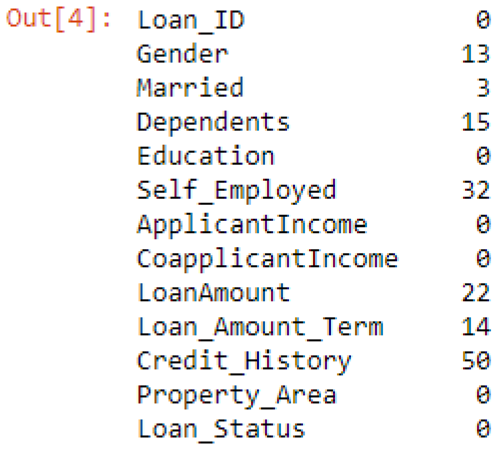
```
(614, 13)
```

```
df.head(5)
```



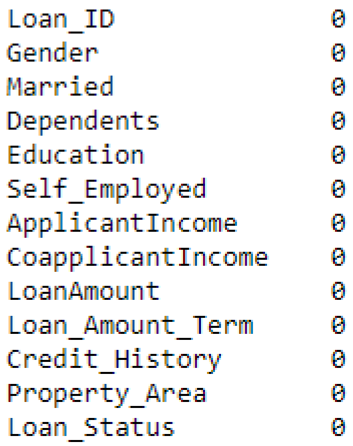
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0

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



```
Out[4]: 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
```

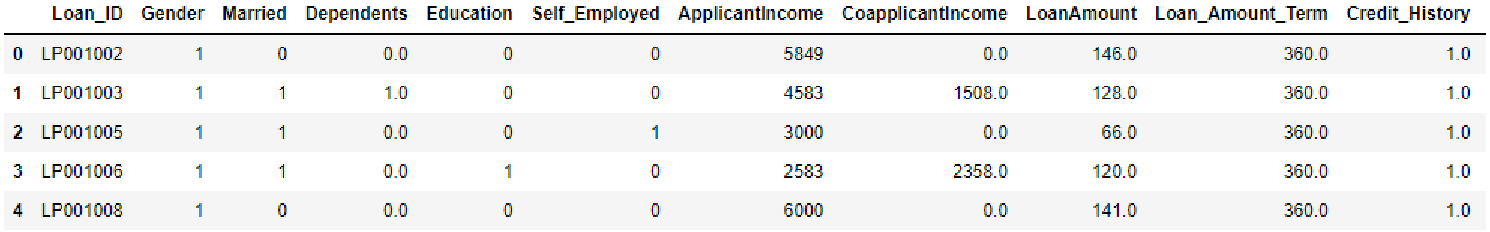
```
for value in ['Gender','Married','Dependents','Self_Employed','Loan_Amount_Term','Credit_History']:
    df[value].fillna(df[value].mode()[0],inplace=True)
print(df.isnull().sum())
df["LoanAmount"].fillna(round(df["LoanAmount"].mean(),0),inplace=True)
print(df.isnull().sum())
```



```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
```

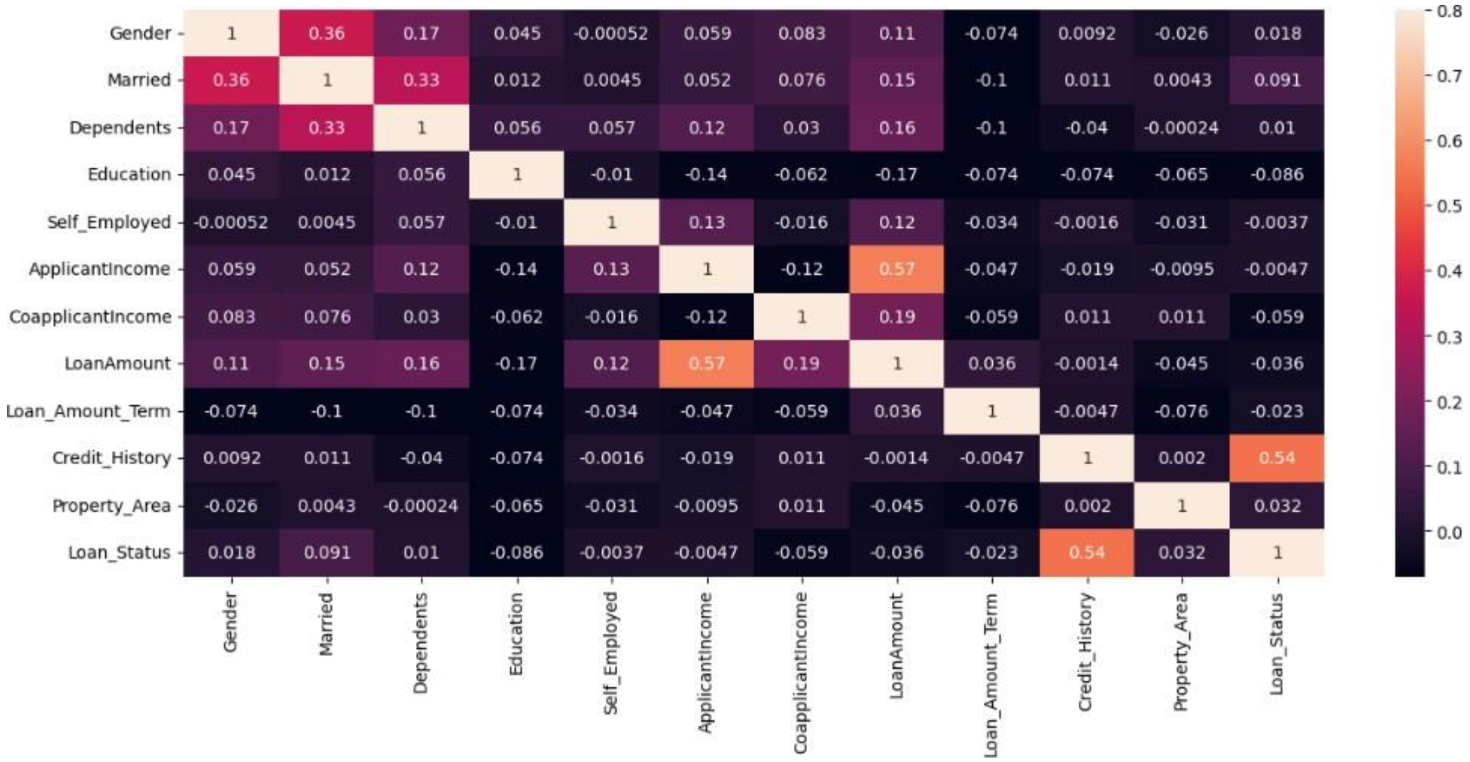
```
from sklearn.preprocessing import LabelEncoder
```

```
colname=["Gender",'Married','Education','Self_Employed','Property_Area', 'Loan_Status']
le= LabelEncoder()
for x in colname:
   df[x]=le.fit_transform(df[x])
df.head()
```

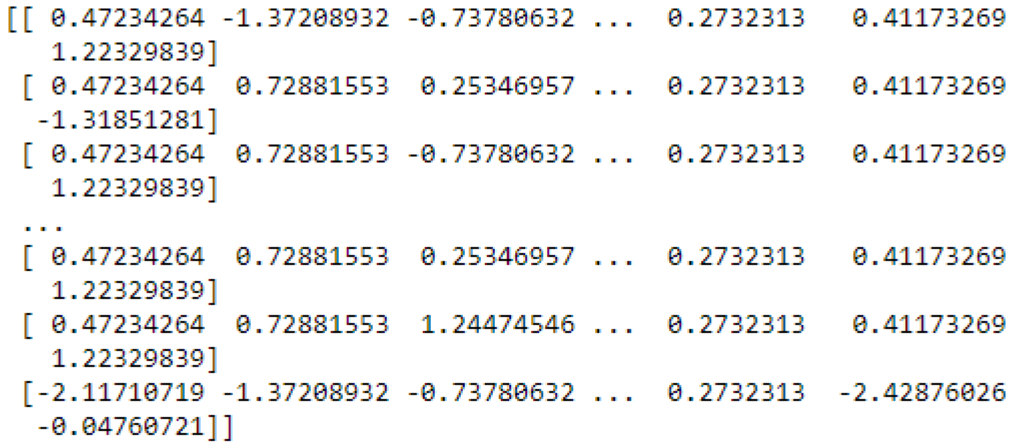


	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	1	0	0.0	0	0	5849	0.0	146.0	360.0	1.0
1	LP001003	1	1	1.0	0	0	4583	1508.0	128.0	360.0	1.0
2	LP001005	1	1	0.0	0	1	3000	0.0	66.0	360.0	1.0
3	LP001006	1	1	0.0	1	0	2583	2358.0	120.0	360.0	1.0
4	LP001008	1	0	0.0	0	0	6000	0.0	141.0	360.0	1.0

```
corr_df = df.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr_df,vmin=-0.07,vmax=0.8,annot=True)
plt.show()
```

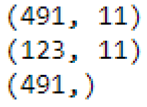


```
X=df.drop(['Loan_Status','Loan_ID'],axis=1)
Y=df.Loan_Status
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X)
x=scaler.transform(X)
print(x)
```



```
[[ 0.47234264 -1.37208932 -0.73780632 ...  0.2732313   0.41173269
   1.22329839]
 [ 0.47234264  0.72881553  0.25346957 ...  0.2732313   0.41173269
  -1.31851281]
 [ 0.47234264  0.72881553 -0.73780632 ...  0.2732313   0.41173269
   1.22329839]
 ...
 [ 0.47234264  0.72881553  0.25346957 ...  0.2732313   0.41173269
   1.22329839]
 [ 0.47234264  0.72881553  1.24474546 ...  0.2732313   0.41173269
   1.22329839]
 [-2.11710719 -1.37208932 -0.73780632 ...  0.2732313  -2.42876026
  -0.04760721]]
```

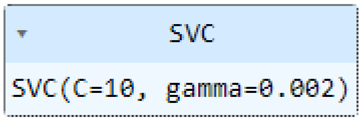
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,Y,test_size=0.2,random_state=10)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
y_test.shape
```



```
(491, 11)
(123, 11)
(491,)
```

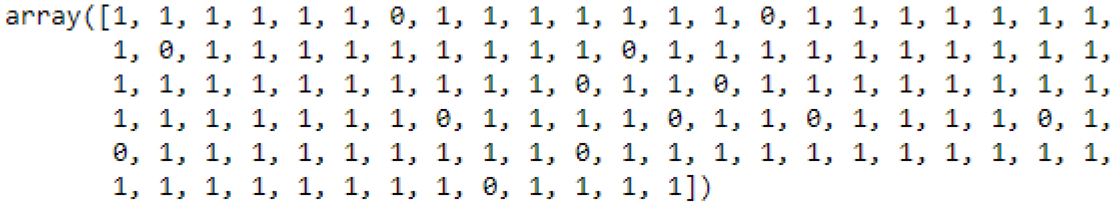
```
(123,)
```

```
from sklearn.svm import SVC
svc_model=SVC(kernel='rbf',C=10,gamma=0.002)
svc_model.fit(X_train,y_train)
```



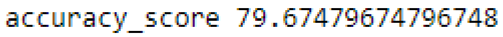
```
SVC(C=10, gamma=0.002)
```

```
y_pred=svc_model.predict(X_test)
y_pred
```



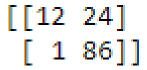
```
array([1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1])
```

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
acc=accuracy_score(y_test,y_pred)
print("accuracy_score",acc*100)
```



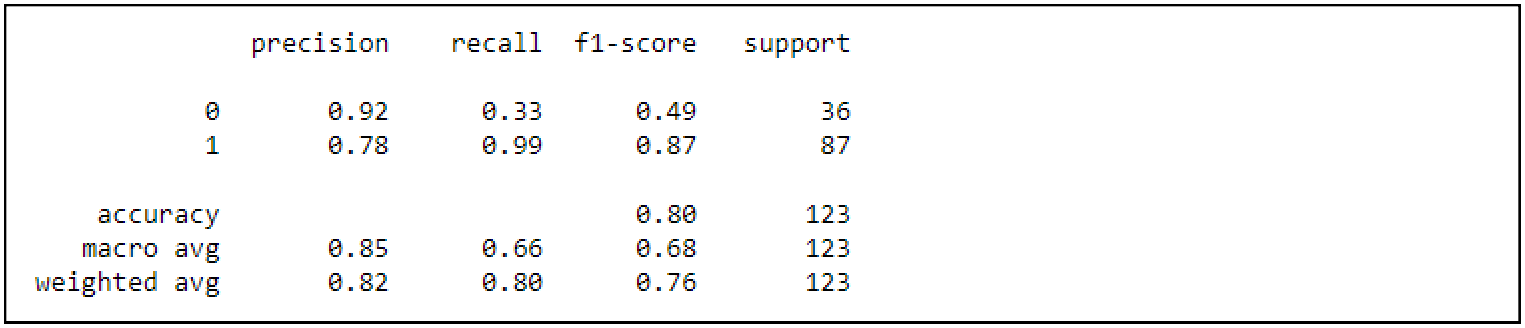
```
accuracy_score 79.67479674796748
```

```
con=confusion_matrix(y_test,y_pred)
print(con)
```



```
[[12 24]
 [ 1 86]]
```

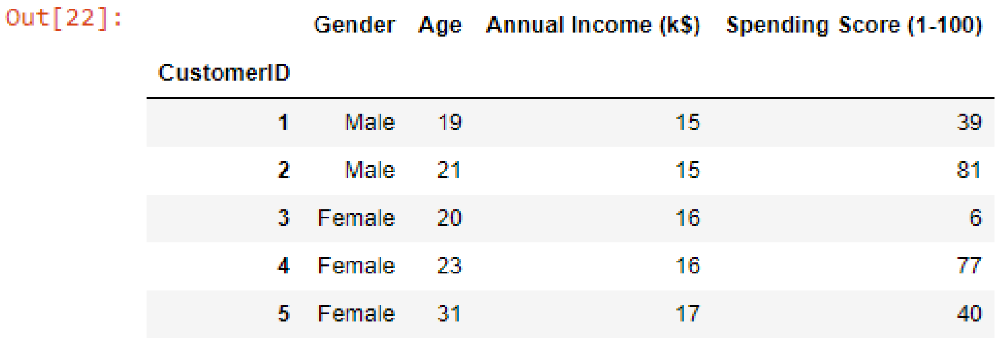
```
print(classification_report(y_test,y_pred))
```



```
              precision    recall  f1-score   support

           0       0.92      0.33      0.49        36
           1       0.78      0.99      0.87        87

    accuracy                           0.80       123
   macro avg       0.85      0.66      0.68       123
weighted avg       0.82      0.80      0.76       123
```

Practical 6

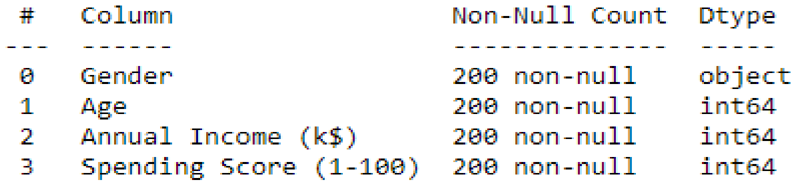
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv(r"C:\Users\User38\Desktop\shraddha\Mall_Customers - Mall_Customers.csv", index_col =0,
header=0)
df.head(5)
```



Out[22]:

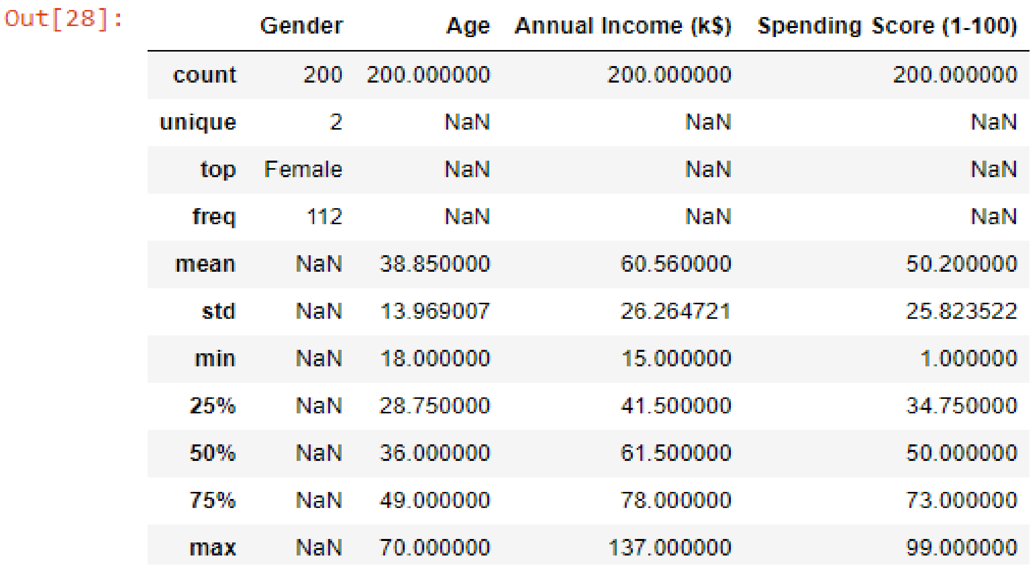
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
CustomerID				
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40

```
print(df.shape)
df.info()
```



```
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Gender                  200 non-null    object
 1   Age                     200 non-null    int64 
 2   Annual Income (k$)      200 non-null    int64 
 3   Spending Score (1-100)  200 non-null    int64 
```

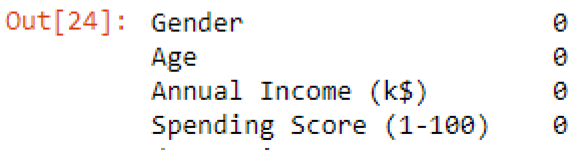
```
df.describe(include="all")
```



Out[28]:

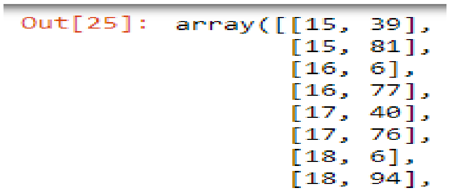
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
count	200	200.000000	200.000000	200.000000
unique	2	NaN	NaN	NaN
top	Female	NaN	NaN	NaN
freq	112	NaN	NaN	NaN
mean	NaN	38.850000	60.560000	50.200000
std	NaN	13.969007	26.264721	25.823522
min	NaN	18.000000	15.000000	1.000000
25%	NaN	28.750000	41.500000	34.750000
50%	NaN	36.000000	61.500000	50.000000
75%	NaN	49.000000	78.000000	73.000000
max	NaN	70.000000	137.000000	99.000000

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



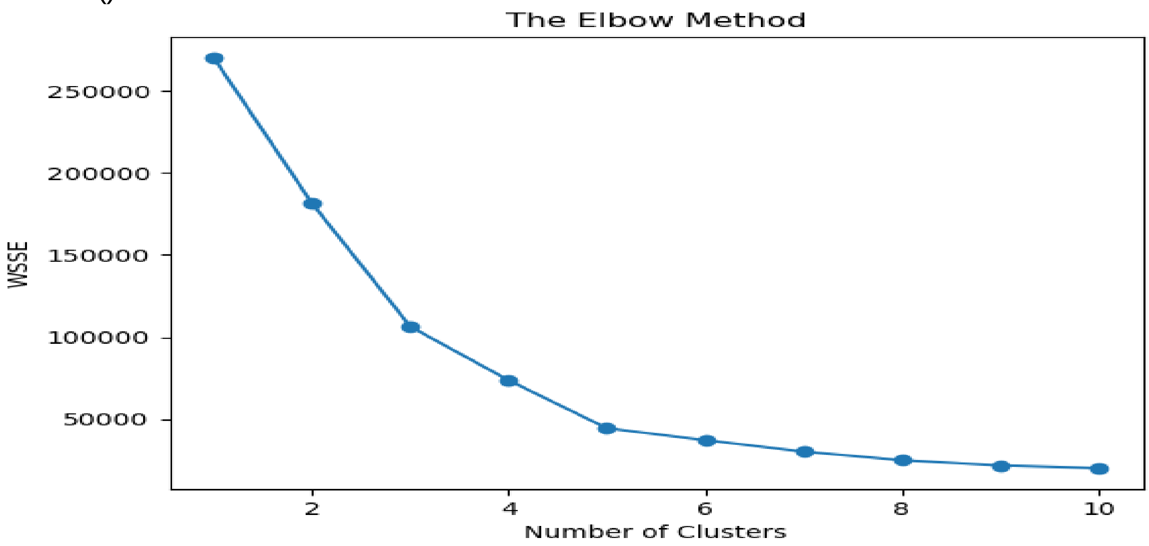
```
Out[24]: Gender                    0
         Age                       0
         Annual Income (k$)        0
         Spending Score (1-100)    0
```

```
x =df.values[:,[2,3]]
x
```

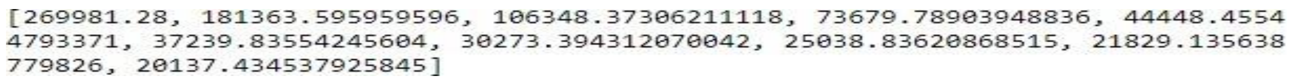


```
Out[25]:  array([[15, 39],
                 [15, 81],
                 [16,  6],
                 [16, 77],
                 [17, 40],
                 [17, 76],
                 [18,  6],
                 [18, 94],
```

```
from sklearn.cluster import KMeans
wsse =[]
for i in range (1,11):
    kmeans = KMeans(n_clusters =i, random_state =10)
    kmeans.fit(x)
    wsse.append(kmeans.inertia_)
plt.plot(range(1,11), wsse)
plt.scatter(range(1,11),wsse)
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WSSE')
plt.show()
```

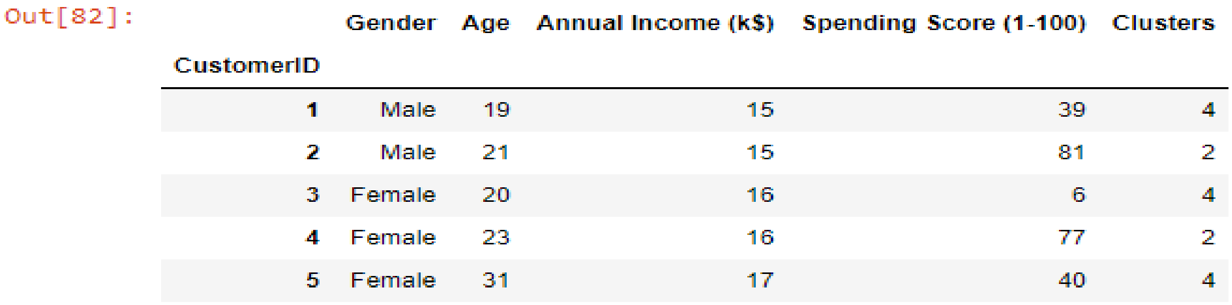


```
print(wsse)
```



```
[269981.28, 181363.595959596, 106348.37306211118, 73679.78903948836, 44448.4554
4793371, 37239.83554245604, 30273.394312070042, 25038.83620868515, 21829.135638
779826, 20137.434537925845]
```

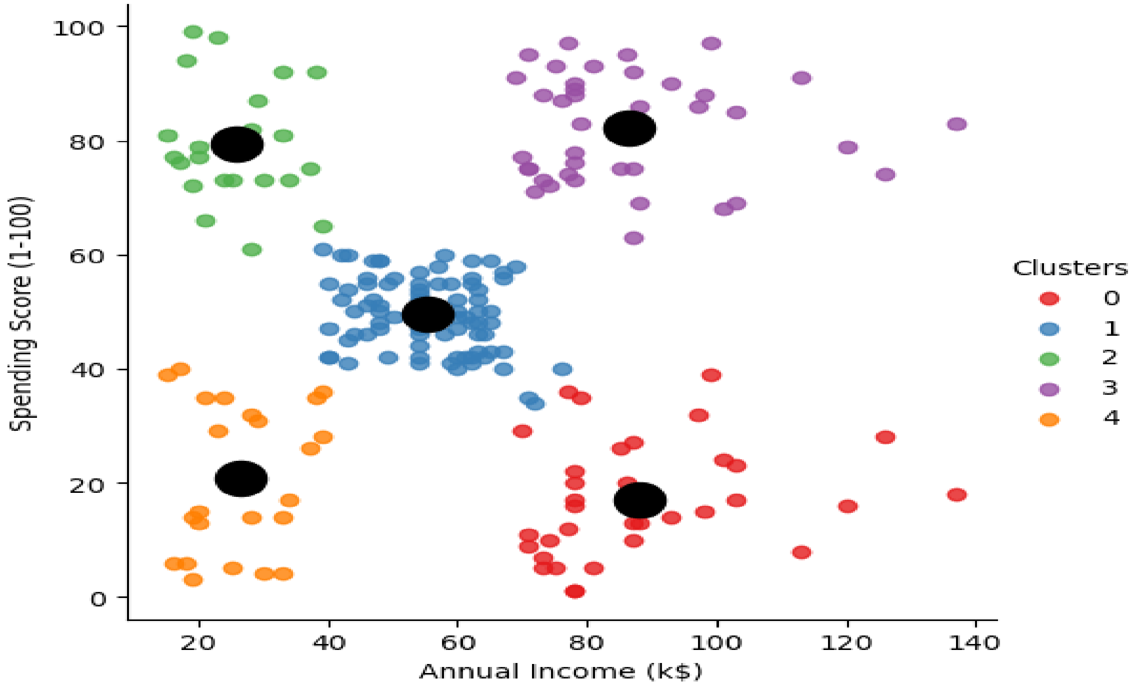
```
kmeans = KMeans(n_clusters=5,random_state=10)
Y_pred = kmeans.fit_predict(x)
Y_pred
kmeans.inertia_
df['Clusters']=Y_pred
df.head()
```



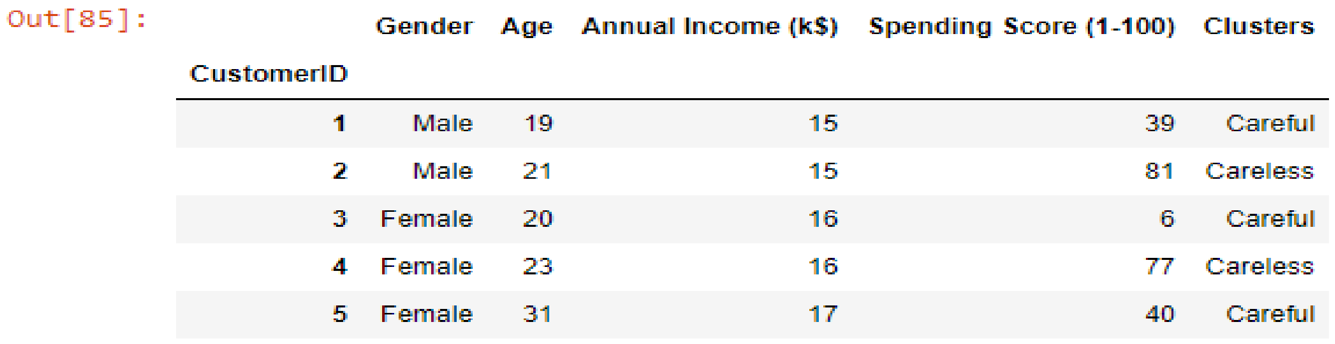
Out[82]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Clusters
CustomerID					
1	Male	19	15	39	4
2	Male	21	15	81	2
3	Female	20	16	6	4
4	Female	23	16	77	2
5	Female	31	17	40	4

```
sns.lmplot(data=df, x='Annual Income (k$)',y='Spending Score (1-100)', fit_reg=False, #No regression line
        hue='Clusters', palette='Set1')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], s = 300, c='black')
plt.show()
```



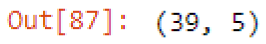
```
df['Clusters']=df.Clusters.replace({1:'Standard',3:'Target',0:'Sensible',2:'Careless',4:'Careful'})
df.head()
```



Out[85]:

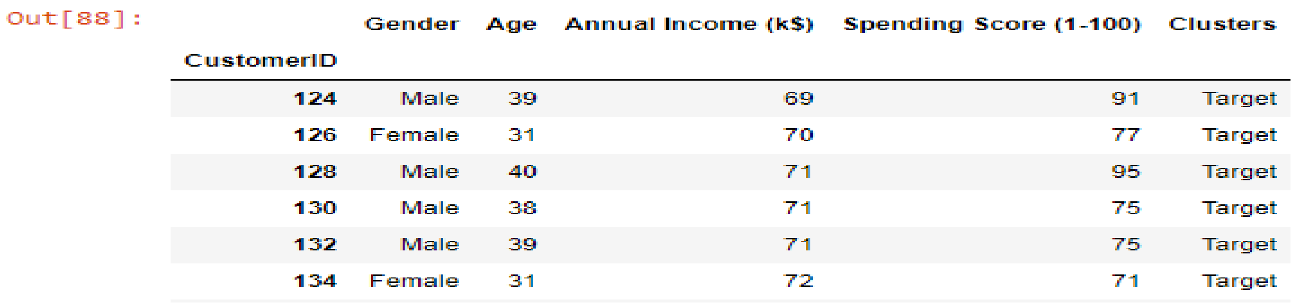
	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Clusters
CustomerID					
1	Male	19	15	39	Careful
2	Male	21	15	81	Careless
3	Female	20	16	6	Careful
4	Female	23	16	77	Careless
5	Female	31	17	40	Careful

```
new_df = df[df['Clusters']=='Target']
new_df.shape
```



```
Out[87]: (39, 5)
```

```
new_df
```



Out[88]:

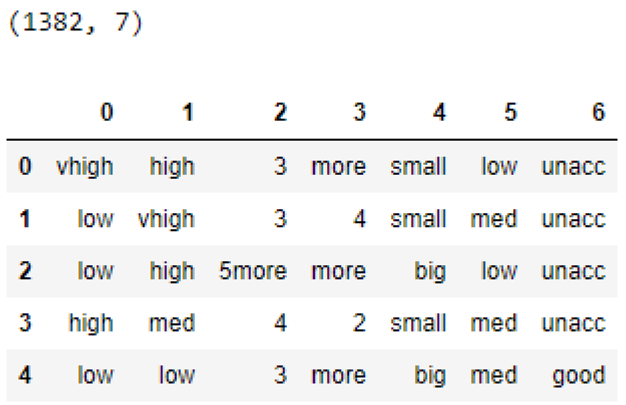
	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Clusters
CustomerID					
124	Male	39	69	91	Target
126	Female	31	70	77	Target
128	Male	40	71	95	Target
130	Male	38	71	75	Target
132	Male	39	71	75	Target
134	Female	31	72	71	Target

```
new_df.to_excel(r"TargetCustomers.xlsx",index=True)
```

Signature:___________

Practical 7

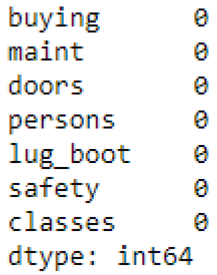
```
import pandas as pd
import numpy as np
car_train = pd.read_csv(r"C:\Users\User38\Desktop\shraddha\cars_train.csv", header=None)
print(car_train.shape)
car_train.head()
```



```
(1382, 7)
```

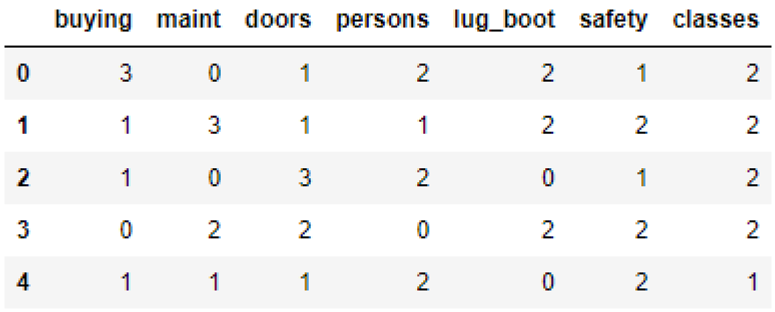
	0	1	2	3	4	5	6
0	vhigh	high	3	more	small	low	unacc
1	low	vhigh	3	4	small	med	unacc
2	low	high	5more	more	big	low	unacc
3	high	med	4	2	small	med	unacc
4	low	low	3	more	big	med	good

```
car_train.columns=["buying","maint","doors","persons",'lug_boot',"safety","classes"]
car_train.head()
```



```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
classes     0
dtype: int64
```

```
colname = car_train.columns
colname
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
for x in colname:
    car_train[x] = le.fit_transform(car_train[x])
car_train.head()
```

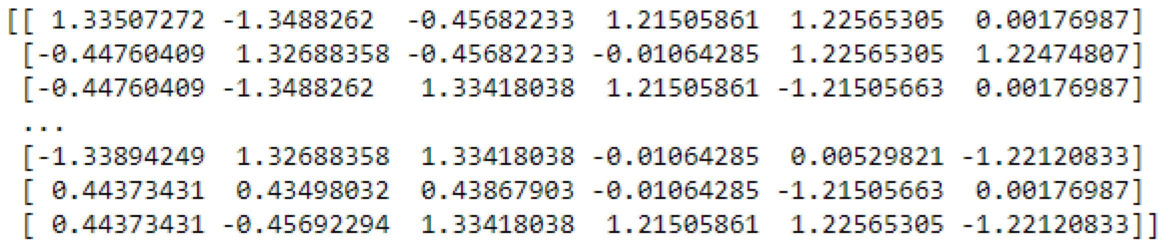


	buying	maint	doors	persons	lug_boot	safety	classes
0	3	0	1	2	2	1	2
1	1	3	1	1	2	2	2
2	1	0	3	2	0	1	2
3	0	2	2	0	2	2	2
4	1	1	1	2	0	2	1

```
X=car_train.values[:,0:-1]
Y=car_train.values[:,-1]
Y=Y.astype(int)
X.shape
```

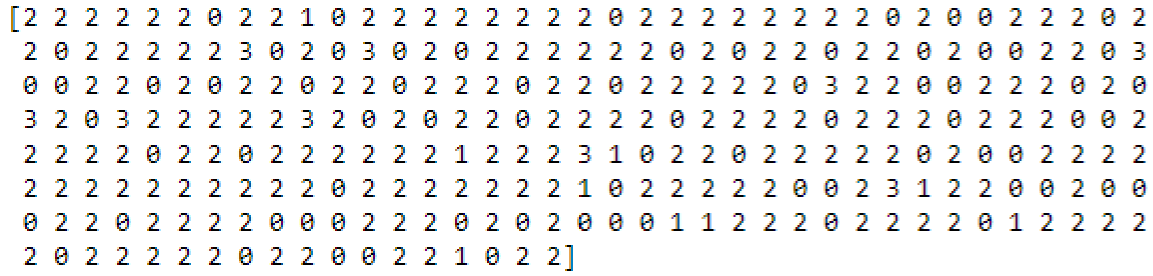
```
Out[12]: (1382, 6)
```

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X=scaler.transform(X)
print(X)



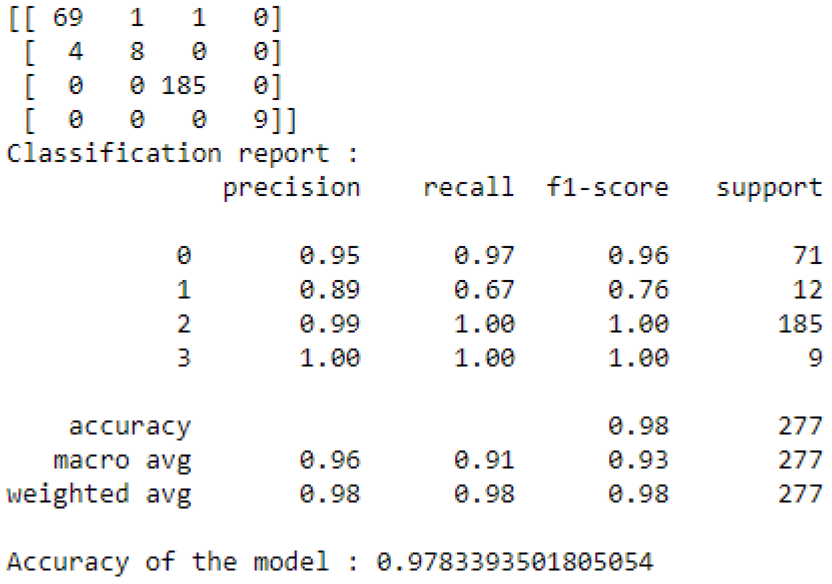
```
[[ 1.33507272 -1.3488262  -0.45682233  1.21505861  1.22565305  0.00176987]
 [-0.44760409  1.32688358 -0.45682233 -0.01064285  1.22565305  1.22474807]
 [-0.44760409 -1.3488262   1.33418038  1.21505861 -1.21505663  0.00176987]
 ...
 [-1.33894249  1.32688358  1.33418038 -0.01064285  0.00529821 -1.22120833]
 [ 0.44373431  0.43498032  0.43867903 -0.01064285 -1.21505663  0.00176987]
 [ 0.44373431 -0.45692294  1.33418038  1.21505861  1.22565305 -1.22120833]]
```

from sklearn.tree import DecisionTreeClassifier
model_DecisionTree = DecisionTreeClassifier(criterion='gini',random_state=10,splitter='best')
model_DecisionTree.fit(x_train,y_train)
Y_pred = model_DecisionTree.predict(x_test)
print(Y_pred)



```
[2 2 2 2 2 2 0 2 2 1 0 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 0 2 0 0 2 2 2 0 2
 2 0 2 2 2 2 2 3 0 2 0 3 0 2 0 2 2 2 2 2 2 0 2 0 2 2 0 2 2 0 2 0 0 2 2 0 3
 0 0 2 2 0 2 0 2 2 0 2 2 0 2 2 2 0 2 2 0 2 2 2 2 2 0 3 2 2 0 0 2 2 2 0 2 0
 3 2 0 3 2 2 2 2 2 3 2 0 2 0 2 2 0 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 0 2
 2 2 2 2 0 2 2 0 2 2 2 2 2 2 1 2 2 2 3 1 0 2 2 0 2 2 2 2 2 0 2 0 0 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 1 0 2 2 2 2 2 0 0 2 3 1 2 2 0 0 2 0 0
 0 2 2 0 2 2 2 2 0 0 0 2 2 2 0 2 0 2 0 0 0 1 1 2 2 2 0 2 2 2 2 0 1 2 2 2 2
 2 0 2 2 2 2 2 0 2 2 0 0 2 2 1 0 2 2]
```

from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cfm=confusion_matrix(y_test,Y_pred)
print(cfm)
print("Classification report :")
print(classification_report(y_test,Y_pred))
acc= accuracy_score(y_test,Y_pred)
print("Accuracy of the model :",acc)

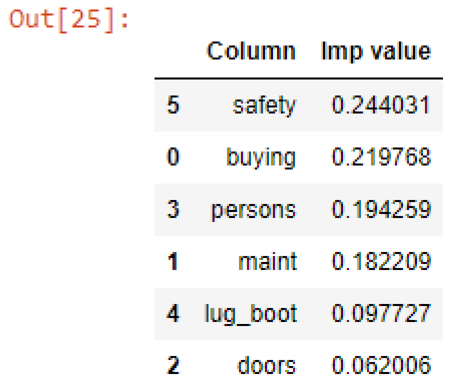


```
[[ 69   1   1   0]
 [  4   8   0   0]
 [  0   0 185   0]
 [  0   0   0   9]]
Classification report :
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        71
           1       0.89      0.67      0.76        12
           2       0.99      1.00      1.00       185
           3       1.00      1.00      1.00         9

    accuracy                           0.98       277
   macro avg       0.96      0.91      0.93       277
weighted avg       0.98      0.98      0.98       277

Accuracy of the model : 0.9783393501805054
```

model_DecisionTree.score(x_train,y_train)

```
Out[23]: 1.0
```

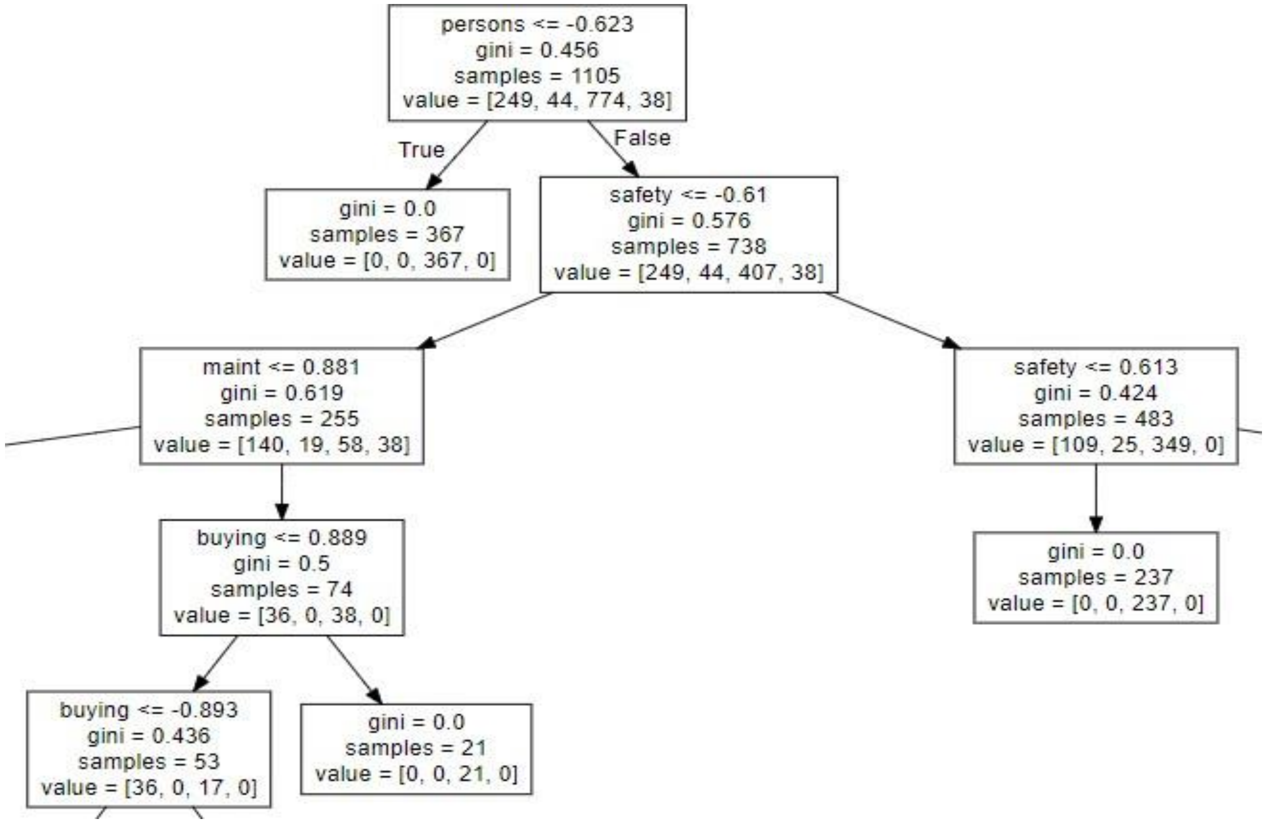
```
print((list(zip(car_train.columns[0:-1],model_DecisionTree.feature_importances_))))
sample=pd.DataFrame()
sample["Column"]= car_train.columns[0:-1]
sample["Imp value"] = model_DecisionTree.feature_importances_
sample.sort_values("Imp value", ascending=False)
```



Out[25]:

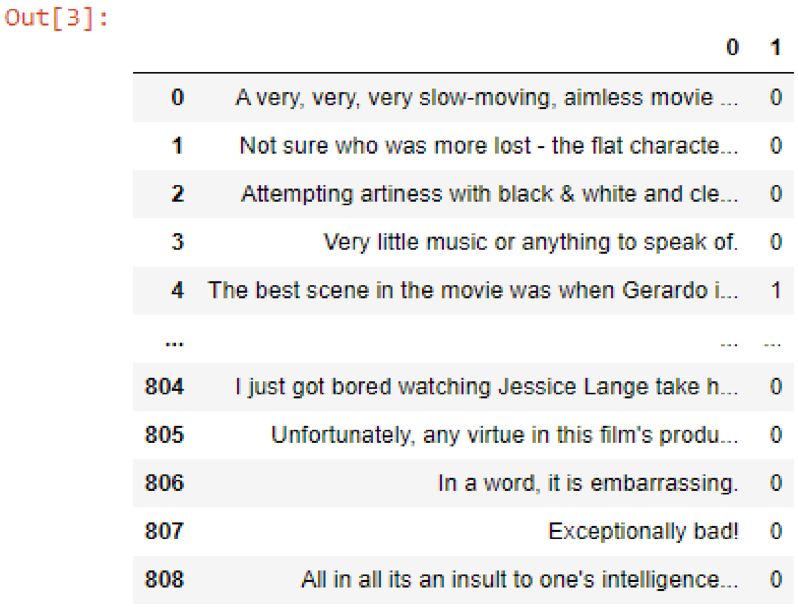
	Column	Imp value
5	safety	0.244031
0	buying	0.219768
3	persons	0.194259
1	maint	0.182209
4	lug_boot	0.097727
2	doors	0.062006

```
from sklearn import tree
with open(r"model_DecisionTree.txt", "w") as f:
    f=tree.export_graphviz(model_DecisionTree,feature_names=car_train.columns[0:-1],out_file=f)
#open Webgraphviz in browser and paste the contents of model_DecisionTree.txt and generate graph
```



Practical 8

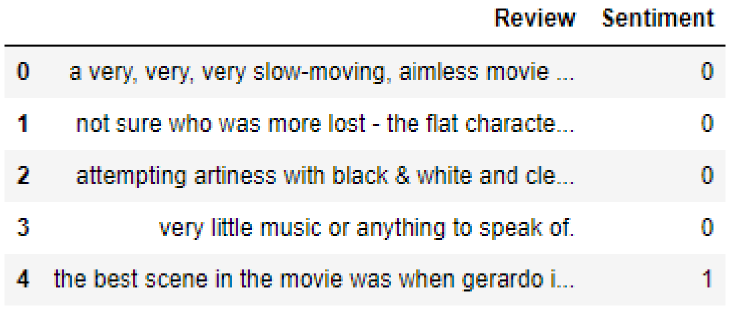
```
import pandas as pd
df=pd.read_csv(r"C:\Users\User38\Desktop\shraddha\imdb_labelled new.txt",delimiter="\t", header=None)
df
```



Out[3]:

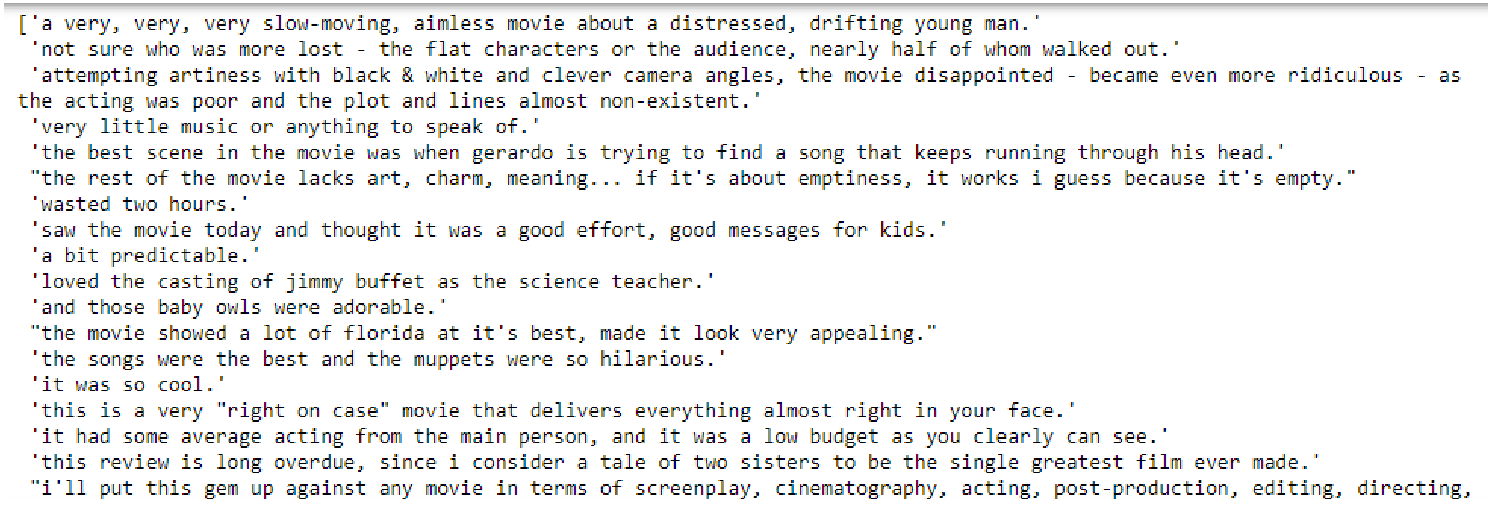
	0	1
0	A very, very, very slow-moving, aimless movie ...	0
1	Not sure who was more lost - the flat characte...	0
2	Attempting artiness with black & white and cle...	0
3	Very little music or anything to speak of.	0
4	The best scene in the movie was when Gerardo i...	1
...	...	...
804	I just got bored watching Jessice Lange take h...	0
805	Unfortunately, any virtue in this film's produ...	0
806	In a word, it is embarrassing.	0
807	Exceptionally bad!	0
808	All in all its an insult to one's intelligence...	0

```
df.columns=["Review","Sentiment"]
df.head()
```

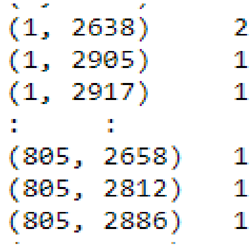


	Review	Sentiment
0	a very, very, very slow-moving, aimless movie ...	0
1	not sure who was more lost - the flat characte...	0
2	attempting artiness with black & white and cle...	0
3	very little music or anything to speak of.	0
4	the best scene in the movie was when gerardo i...	1

```
x = df.values[:,0]
y = df.values[:,1]
y=y.astype(int)
print(x)
```

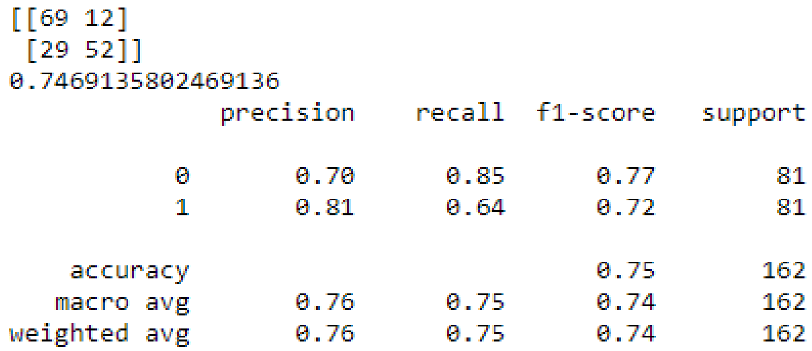


```
['a very, very, very slow-moving, aimless movie about a distressed, drifting young man.'
 'not sure who was more lost - the flat characters or the audience, nearly half of whom walked out.'
 'attempting artiness with black & white and clever camera angles, the movie disappointed - became even more ridiculous - as
the acting was poor and the plot and lines almost non-existent.'
 'very little music or anything to speak of.'
 'the best scene in the movie was when gerardo is trying to find a song that keeps running through his head.'
 "the rest of the movie lacks art, charm, meaning... if it's about emptiness, it works i guess because it's empty."
 'wasted two hours.'
 'saw the movie today and thought it was a good effort, good messages for kids.'
 'a bit predictable.'
 'loved the casting of jimmy buffet as the science teacher.'
 'and those baby owls were adorable.'
 "the movie showed a lot of florida at it's best, made it look very appealing."
 'the songs were the best and the muppets were so hilarious.'
 'it was so cool.'
 'this is a very "right on case" movie that delivers everything almost right in your face.'
 'it had some average acting from the main person, and it was a low budget as you clearly can see.'
 'this review is long overdue, since i consider a tale of two sisters to be the single greatest film ever made.'
 "i'll put this gem up against any movie in terms of screenplay, cinematography, acting, post-production, editing, directing,
from sklearn.feature_extraction.text import CountVectorizer
cv= CountVectorizer()
cv.fit(x)
x=cv.transform(x)
print(x)
```



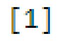
```
(1, 2638)     2
(1, 2905)     1
(1, 2917)     1
:      :
(805, 2658)   1
(805, 2812)   1
(805, 2886)   1
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=10)
from sklearn.naive_bayes import BernoulliNB
model=BernoulliNB(alpha=1.0,binarize=0)
model.fit(x_train,y_train)
Y_pred=model.predict(x_test)
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(y_test,Y_pred))
print(accuracy_score(y_test,Y_pred))
print(classification_report(y_test,Y_pred))
```



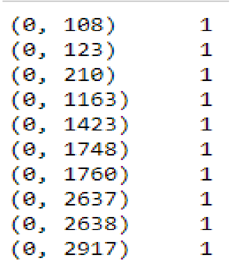
```
[[69 12]
 [29 52]]
0.7469135802469136
              precision    recall  f1-score   support

           0       0.70      0.85      0.77        81
           1       0.81      0.64      0.72        81

    accuracy                           0.75       162
   macro avg       0.76      0.75      0.74       162
weighted avg       0.76      0.75      0.74       162
```

```
test=["that was an awesome movie, the music is also good."]
test=cv.transform(test)
test_pred=model.predict(test)
print(test_pred)
```



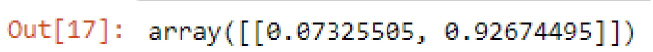
```
[1]
```

```
print(test)
```



```
(0, 108)      1
(0, 123)      1
(0, 210)      1
(0, 1163)     1
(0, 1423)     1
(0, 1748)     1
(0, 1760)     1
(0, 2637)     1
(0, 2638)     1
(0, 2917)     1
```

```
model.predict_proba(test)
```



```
Out[17]:  array([[0.07325505, 0.92674495]])
```

Signature:____________