Experiment-1

Aim:-

Use Advertising data that has Sales (in thousands of units) for a particular product as a function of advertising budgets (in thousands of dollars) for TV, radio, and newspaper media.

1) Apply least squares method model for regression of number of units sold on TV advertising budget for the Advertising data, for least squares coefficient estimates for simple linear regression.

Description:-

→ The method of least squares is a parameter estimation method in regression analysis based on minimizing the sum of the squares of residuals (difference between an observed value and fitted value provided by model) made in the results of each individual equation.

Program:

```
> impolt numpy as up
  impolt pandas as pd
  impolt matplotlib.pyplot as plt

>from google.colab impolt files
  uploaded = files.upload()
```

Choose files | advertising.csv

* 100%. done

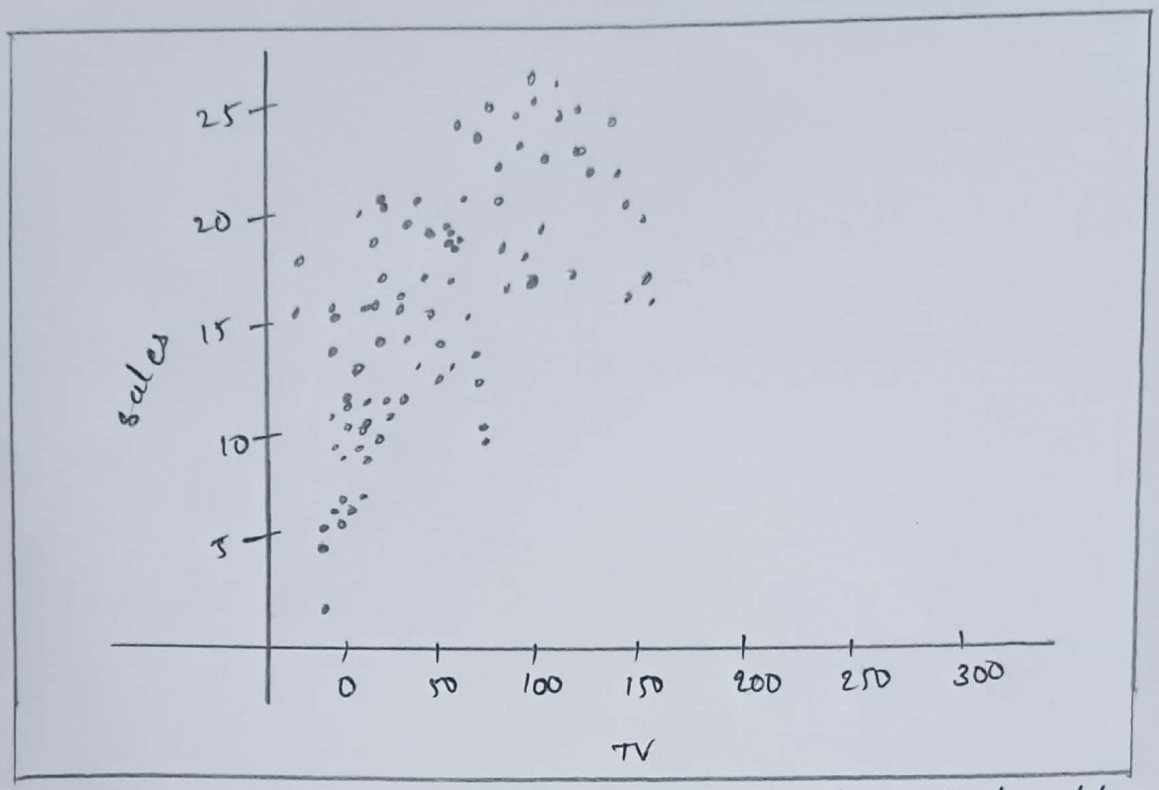
```
> df = pd.read_csv('adversting.csv')
  df.info()
```

#	column	Non-Null count	dtype
0	TV	200 non-null	float64
1	Radio	200 non-null	float64
2	Newspaper	200 non-null	float64
3	Sales	200 non-null	float64

```
>df.shape
```

(200, 4)

```
>impdt seaborn as sns
 sns.pairplot(df, x_vars=['TV'], y_vars='Sales',
                        size=10, aspect=0.5)
X = df.iloc[:,[1,2]].values
Y = df.iloc[:, -1].values
```




```
>from sklearn.model_selection impdt train_test_split
 x_train, x_test, y_train, y_test = train_test_split(x, y,
                                    test_size=0.5)
>impdt statsmodel.api as sm
 x_train_sm = sm.add_constant(x_train)
 model = sm.OLS(y_train, x_train_sm).fit()
 plint("Model coefficients =", model.params)
```

Output:

Model coefficients = [10.67605974 0.11513418 0.04371561]

Sample program

Aim:

Appling linear regression for heart dataset using least square method.

Program:

```
impolt numpy as up
impolt pandas as pd
impolt matplotlib.pyplot as plt

from google.colab impolt files
uploaded = files.upload()

df = pd.read_csv('heart.csv')
df.info()
df.head()
df.shape
impolt seaboln as sns
sns.pairplot(df, x_vars = ['oldpeak', 'trestbps', 'chol'],
             y_vars = 'age', height = 4, aspect = 1, kind = 'scatter')

x = df.iloc[:, [1,2]].values
y = df.iloc [:, -1].values

from sklearn.model_selection impolt train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                   test_size = 0.3)

impolt statsmodels.api as sm
x_train_sm = sm.add_constant (x_train)
model = sm.OLS(y_train, x_train_sm).fit()
```

```
model.params
print("RSE =", model.rsquared)
```

Output :-

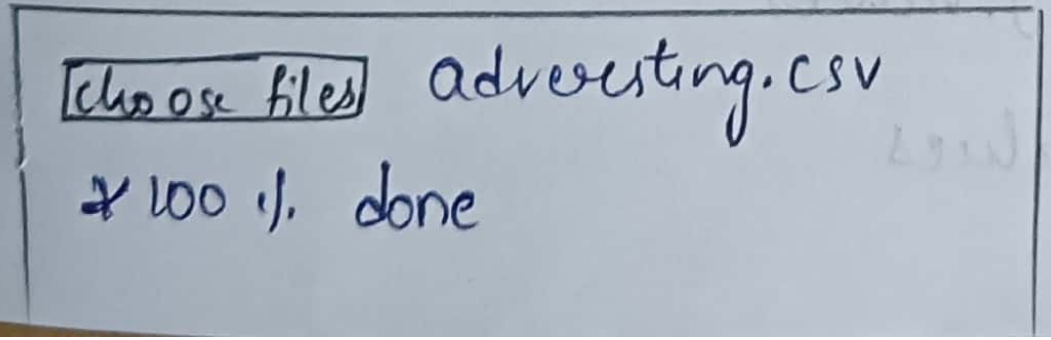
```
[0.524261   -0.293435   0.21369654]
```

RSE = 0.26150138

Program:- (Multiple Regression)

```
>impolt numpy as np
 impolt pandas as pd
 impolt matplotlib.pyplot as plt

>from google.colab impolt files
 uploaded = files.upload()
```



④ df = pd.read_csv('advertising.csv')

df.info()

```
RangeIndex: 200 entries, 0 to 199
Data columns
#   column      Non-Null count   Dtype
--  ------      --------------   -----
0   TV          200 non-null     float64
1   Radio       200 non-null     float64
2   Newspaper   200 non-null     float64
3   Sales       200 non-null     float64
```

⑤ df.head()

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

⑥ df.shape

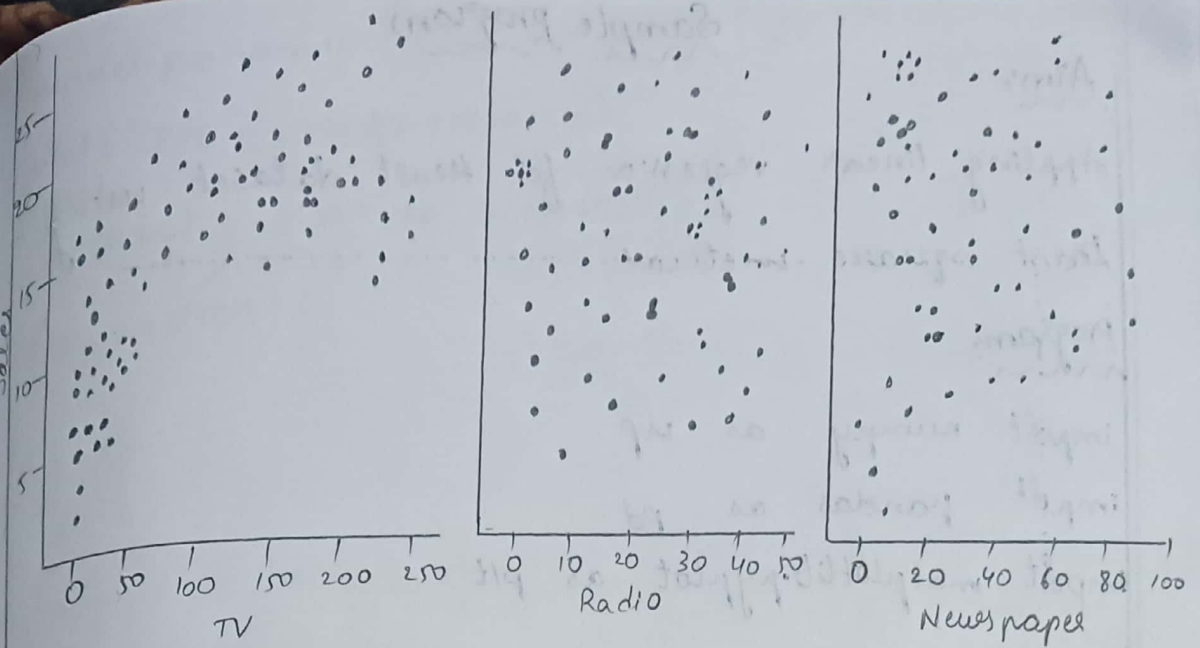
```
(200, 4)
```

⑦ import seaborn as sns

sns.pairplot(df, x_vars=['TV', 'Radio', 'Newspaper'],
y_vars='Sales', size=10, aspect=0.5)

x = df.iloc[:, [1,2]].values

y = df.iloc[:, -1].values



④ from sklearn.model_selection impolt train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.5)

impolt statsmodels.api as sm

x_train_sm = sm.add_constant(x_train)

model = sm.OLS(y_train, x_train_sm).fit()

plint("Model coefficients =", model.params)

plint("Residal standard errol =", model.rsquared)

O/p:-

Model coefficients = [11.71896618 0.10211221 0.01612666]

Residal standard Error = 0.096144033695

Experiment-2

Aim:-

Compute t-statistic, Residual standard error, F-statistic and residual sum of squares (RSS) errors.

Description:-

→ t-statistic is a measure of difference between the means of two groups relative to variability within each group.

1) $t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}}$ $\quad t = \frac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$

→ An F-test is any statistical test used to compare the variances of two samples of the ratio of variances between multiple samples.

Program:-

```
impolt numpy as np
impolt pandas as pd
impolt matplotlib.pyplot as plt

from google.colab impolt files
uploaded =files.upload().

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

df.info()

df.head()

df.shape
```

```
impolt seaborn as sns
sns.pairplot(df, x_vars=['TV', 'Radio', 'Newspaper'],
             y_vars='Sales', size=10, aspect=0.5)
x = df.iloc[:, [1,2]].values
y = df.iloc[:, -1].values
from sklearn.model_selection impolt train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.5)
impolt statsmodels.api as sm
x_train_sm = sm.add_constant(x_train)
model = sm.OLS(y_train, x_train_sm).fit()
model.params
plint("Resuidal standard error=", model.rsquared)
plint("F-statistic=", model.fvalue)
plint("T-statistic=", model.tvalues)
plint("Residual sum of squares=", model.ssr)
```

Output:

```
Residual standard error = 0.0961440336
F-statistic = 5.15899919
T-statistic = [10.75950656  2.63497065  0.65936946]
Residual sum of squares = 2661.45667970
```

Sample

Aim:

Appling KNN for heart data set

Program:

```
impolt pandas as pd
impolt numpy as np
impolt seaboln as sns
impolt matplotlib.pyplot as plt
from sklearn.neighbols impolt kNeighbolsClassifier
from sklearn.metlics impolt accuracy_scole

from google.colab impolt files
uploaded = files.upload()

df=pd.read_scv('heart.csv')

df.head()

sns.countplot(df['target'])

x=df.iloc[:,0:13].values
y=df['target'].values
from sklearn.model_slection impolt train_test_split
x_train,x_test, y_train, y_test=train_test_split(x,y,
                                    test_size=0.25, random_state=0)

from sklearn.preprocesing impolt standardscaler
st_x=Standardscalel()
x_train=st_x.fit_transform(x_train)
x_test=st_x.transfolm(x_test)
```

```
error=[]
for i in range(1,30):
    knn=kNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i=knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))
plt.figure(figsize=(12,6))
plt.plot(range(1,30), error, color='red', linestyle='dashed',
  marker='o', markerfacecolor='blue', markersize=10)
plt.title('Error Rate k value')
plt.xlabel('k value')
plt.ylabel('Mean Error')
print("Minimum error:", min(error), "at k=",
                        error.index(min(error))+1)
```

Output:-

Minimum error :- 0.0 at k=1

```
>>classifier=kNeighborsClassifier(n_neighbors=7)
classifier.fit(x_train, y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred)
>>accuracy_score(y_test, y_pred)
```

Output:-

0.8638132295719845