S.Y. (B.Tech) Semester III

Subject: Statistics for Data Science

Experiment 10

Name: Rishabh Singhvi **SAP ID: 60009210206**

Date:	Experiment Title: Correlation and Regression Analysis
Aim	To implement hypothesis testing on Correlation and Regression Analysis
Software	Google Colab

Theory

```
import numpy as np import
pandas as pd import
scipy.stats as st import
matplotlib.pyplot as plt
```

Q1. *The following table gives the data on weekly family consumption expenditure(Y) and weekly family income(X)*

Y :	70	65	90	95	110	115	120	140	155	150
X :	80	100	120	140	160	180	200	220	240	260

(i) Compute the coefficient of correlation between X and Y.

(ii) Test the significance of the coefficient of correlation between X and Y at 5 percent level of significance.

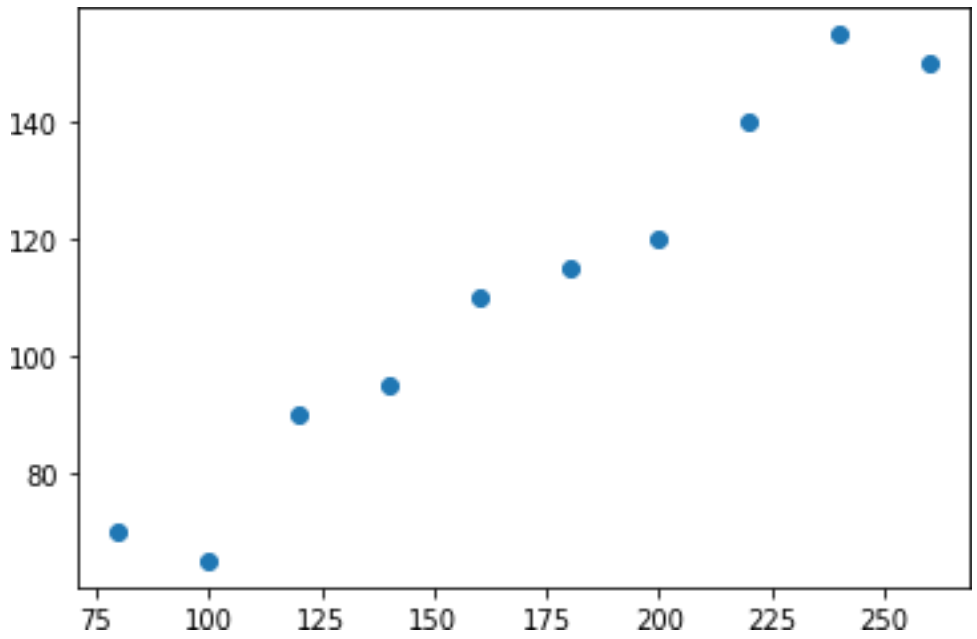
Code:

```
y = np.array([70,65,90,95,110,115,120,140,155,150]) x
=   np.array([80,100,120,140,160,180,200,220,240,260])
print(plt.scatter(x,y))   m_x   =   np.mean(x)   m_y   =
np.mean(y) n = len(x) print("Ux = ",m_x) print("Uy =
",m_y)       print("n       =       ",n)       cov       =
(np.sum(np.array(x)*np.array(y))     -     n*m_x*m_y)/n
print("COV = ",cov)
sx = np.sqrt((np.sum(x*x)-n*m_x*m_x)/n)
sy = np.sqrt((np.sum(y*y)-n*m_y*m_y)/n)
r = cov/(sx*sy) print("r = ",r)
t = r*np.sqrt(n-2)/(np.sqrt(1-r**2))
print('T-statistics =',t) t_cri =
st.t.ppf(0.975,n-2) print('T-
critical =',t_cri) p = (st.t.sf(t,n-
2))*2 print('p-value =',p)
```

Output:

```
<matplotlib.collections.PathCollection object at 0x7fe8a17f5350>
Ux =  170.0
Uy = 111.0 n
= 10 COV =
1680.0
r = 0.9808473685985793
T-statistics= 14.243171154216402
```

```
T-critical= 2.3060041350333704 p-value=
5.752746116732599e-07
```



Ans: Reject the null hypothesis

```
#direct formula for the r and p value
r,p = st.pearsonr(x,y) print("r = ",r)
print("p-value = ",p) r =
0.9808473685985793
p-value = 5.752746116732596e-07
```

Q2. *The following table gives the per capita household expenditure on food (Y) and per capita total household expenditure (X)*

Y :	60	90	110	125	150	170	180	200	220	230	240	250	255	260	260
X :	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800

(i) Compute the coefficient of correlation between X and Y.

(ii) Test the significance of the coefficient of correlation between X and Y at 5 percent level of significance.

Code:

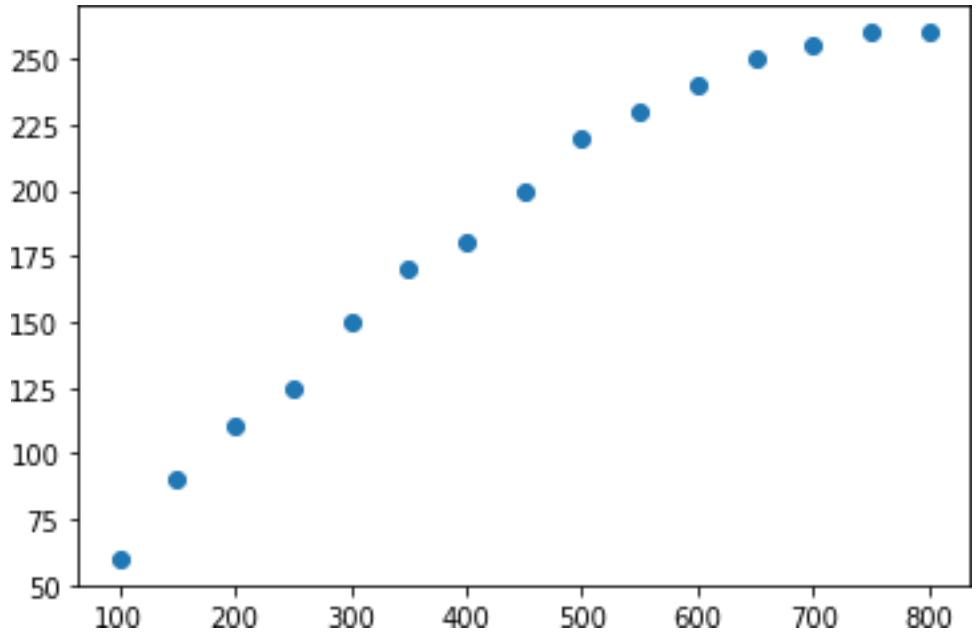
```
y = np.array([60,90,110,125,150,170,180,200,220,230,240,250,255,260,
260]) x =
np.array([100,150,200,250,300,350,400,450,500,550,600,650,700,75
0,800])
plt.scatter(x,y)
m_x = np.mean(x)
m_y = np.mean(y) n
= len(x) print("Ux
= ",m_x) print("Uy
= ",m_y) print("n
= ",n)
cov = (np.sum(np.array(x)*np.array(y)) - n*m_x*m_y)/n print("COV
= ",cov)
```

```
sx = np.sqrt((np.sum(x*x)-n*m_x*m_x)/n)
sy = np.sqrt((np.sum(y*y)-n*m_y*m_y)/n)
```

```
sy r = cov/(sx*sy) print("r = ",r) t
= r*np.sqrt(n-2)/(np.sqrt(1-r**2))
print('T-statistics =',t) t_cri =
st.t.ppf(0.975,n-2) print('T-
critical =',t_cri) p = (st.t.sf(t,n-
2))*2 print('p-value =',p)
```

Output:

```
Ux = 450.0
Uy = 186.66666666666666
n = 15
COV = 13583.333333333334 r
= 0.9765884824912033
T-statistics = 16.368555733381353
T-critical  =  2.1603686564610127
p-value = 4.680215495139796e-10
```



Ans: Reject the null hypothesis

```
#direct formula for the r and p value
r,p = st.pearsonr(x,y) print("r = ",r)
print("p-value    =    ",p)    r    =
0.9765884824912037     p-value     =
4.680215495139356e-10
```

***Q3**. The following table gives the data on weekly family consumption expenditure(Y) and weekly family income(X)*

Y :	70	65	90	95	110	115	120	140	155	150
X :	80	100	120	140	160	180	200	220	240	260

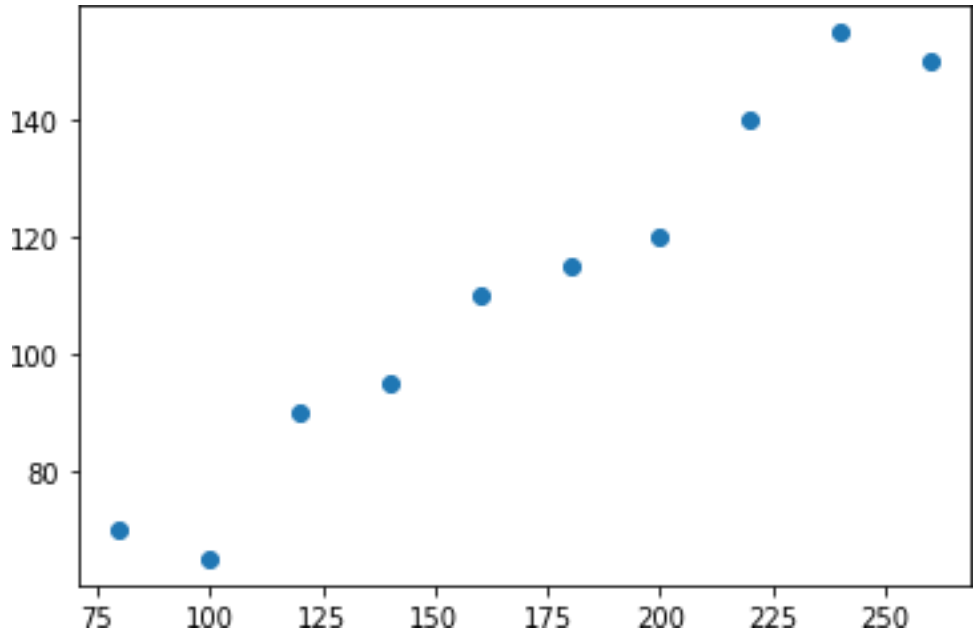
(i) Estimate the consumption function of the family $Y = \beta_0 + \beta_1 X + u$

(ii) Test the significance of the parameters at 5 percent level of significance.

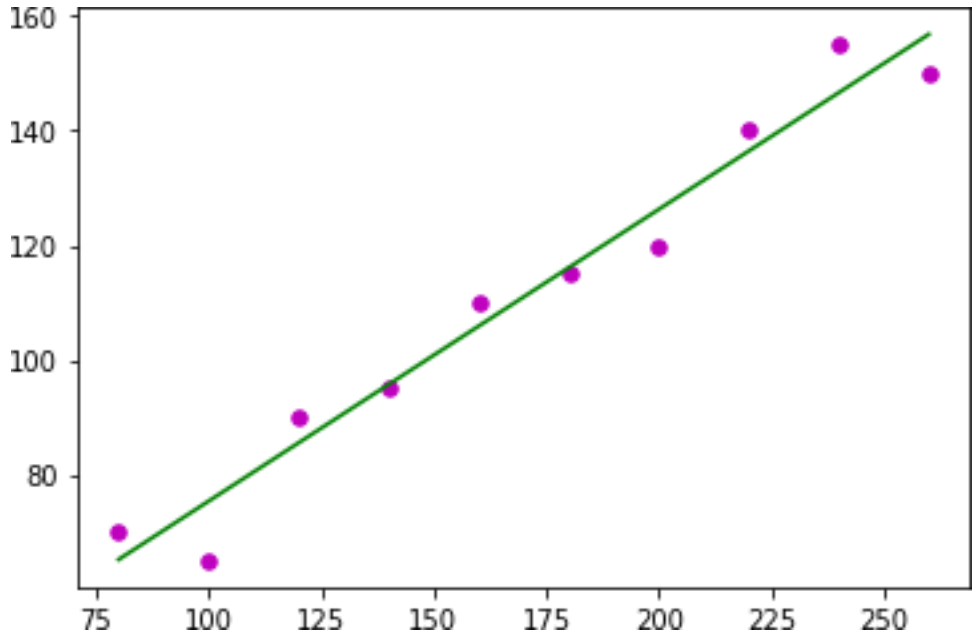
(iii) Find the coefficient of determination.

Code/Output:

```
y = np.array([70,65,90,95,110,115,120,140,155,150]) x
= np.array([80,100,120,140,160,180,200,220,240,260])
print(plt.scatter(x,y))
<matplotlib.collections.PathCollection object at 0x7fe8a0cd3cd0>
```




```
m_x = np.mean(x) m_y =
np.mean(y) n = len(x) sxy =
np.sum(x*y)-n*m_x*m_y sx2 =
np.sum(x*x)-n*m_x*m_x b_1 =
sxy/sx2 b_0 = m_y - b_1*m_x
print("Coefficient β0 = ",b_0) print("Coefficient
β1 = ",b_1)
print(plt.scatter(x,y,color='m',marker='o',s=30)) y_pred
= b_0+(b_1*x)
plt.plot(x,y_pred,color='g')
Coefficient β0 = 24.454545454545467
Coefficient β1 = 0.509090909090909
<matplotlib.collections.PathCollection object at 0x7fe8a0d83a50>
[<matplotlib.lines.Line2D at 0x7fe8a0d2ea50>]
```



Hypothesis test foí β1

- Ho : β1 = 0

```
y_pred = b_0+(b_1*x) num =
np.sum((y-y_pred)**2) den =
np.sum((x-m_x)**2)*(n-2) se_b1
= np.sqrt(num/den) tstats =
b_1/se_b1 print('T-statistics =
',tstats) t_cri =
st.t.ppf(0.975,n-2) print('T-
critical = ',t_cri) p =
(st.t.sf(t,n-2))*2 print('p-
value = ',p)
T-statistics = 14.243171154216384 T-critical
= 2.3060041350333704
p-value = 1.954899670783024e-07
```

Reject the null hypothesis

Hypothesis test foí βo

- Ho: βo = 0

```
ft = (np.sum((y-y_pred)**2))/(n-2) st =
(1/n)+((m_x**2)/(np.sum((x-m_x)**2)))
se_b0 = np.sqrt(ft*st) tstats = b_0/se_b0
print('t-statistics=',tstats) import
scipy.stats as st print('t-critical =
',st.t.ppf(0.975,n-2)) t-statistics=
3.812791090783818
t-critical = 2.3060041350333704
```

Reject the null hypothesis

```
exp_var = np.sum((y_pred-m_y)**2)
tot_var = np.sum((y-m_y)**2) r2 =
exp_var/tot_var
```

```
print('Coefficient of determination = ',r2) Coefficient of determination = 0.9620615604867568 from sklearn.linear_model import LinearRegression y = np.array([70,65,90,95,110,115,120,140,155,150]) x = np.array([80,100,120,140,160,180,200,220,240,260]).reshape((-1,1)) model = LinearRegression().fit(x,y)
print(model.intercept_)
print(model.coef_) r2 = model.score(x,y) print(r2)
24.454545454545467
[0.50909091]
```

***Q4**. The following table gives the per capita household expenditure on food (Y) and per capita total household expenditure (X)*

Y :	60	90	110	125	150	170	180	200	220	230	240	250	255	260	260
X :	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800

(i) Estimate the food expenditure equation $Y = \beta_0 + \beta_1 X + u$

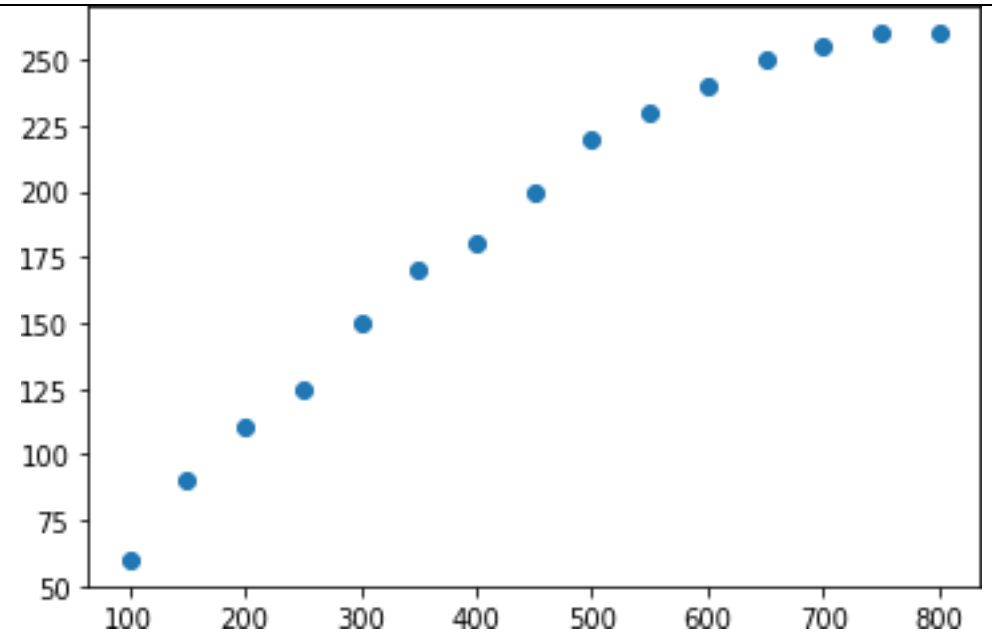
(ii) Test the significance of the parameters at 5 percent level of significance.

(iii) Find the coefficient of determination.

```
0.9620615604867573
```

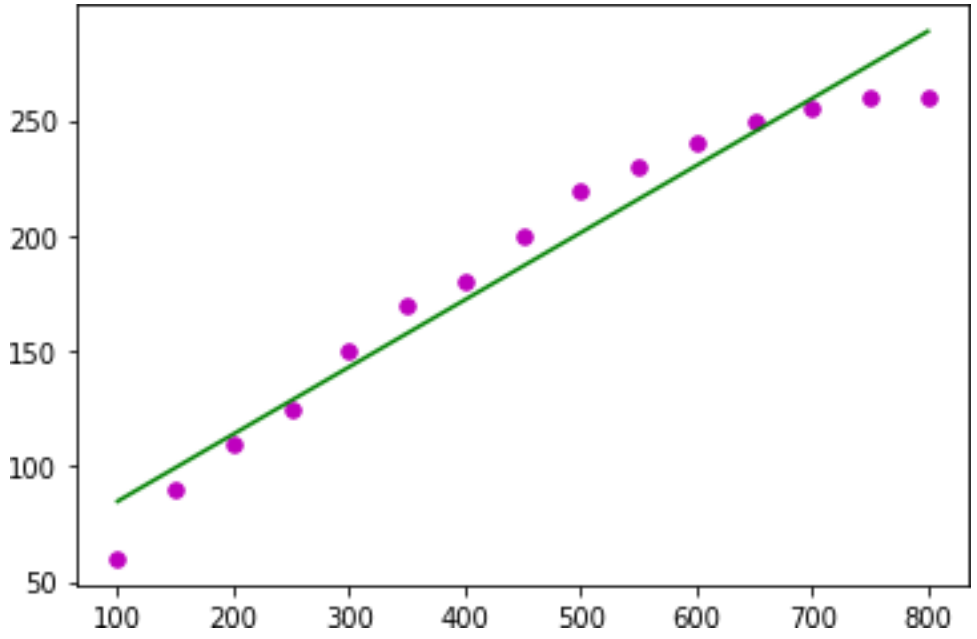
Code/Output:

```
y = np.array([60,90,110,125,150,170,180,200,220,230,240,250,255,260,260]) x = np.array([100,150,200,250,300,350,400,450,500,550,600,650,700,750,800])
plt.scatter(x,y)
<matplotlib.collections.PathCollection at 0x7fe89fff2f90>
```




```
m_x = np.mean(x)
m_y = np.mean(y) n
= len(x)
sxy = np.sum(x*y)-n*m_x*m_y
sx2 = np.sum(x*x)-n*m_x*m_x
b_1 = sxy/sx2
```

```
b_0 = m_y - b_1*m_x print("Coefficient β0 = ",b_0)
print("Coefficient β1 = ",b_1)
print(plt.scatter(x,y,color='m',marker='o',s=30))
y_pred = b_0+(b_1*x) plt.plot(x,y_pred,color='g')
Coefficient β0 = 55.684523809523796
Coefficient β1 = 0.2910714285714286
<matplotlib.collections.PathCollection object at 0x7fe89ff57cd0>
[<matplotlib.lines.Line2D at 0x7fe89ff819d0>]
```



Hypothesis test foí β1

- Ho : β1 = 0

```
y_pred = b_0+(b_1*x) num = np.sum((y-
y_pred)**2) den = np.sum((x-
m_x)**2)*(n-2) se_b1 =
np.sqrt(num/den) tstats = b_1/se_b1
print('T-statistics = ',tstats) t_cri
= st.t.ppf(0.975,n-2) print('T-
critical = ',t_cri) p = (st.t.sf(t,n-
2))*2 print('p-value = ',p)
T-statistics = 16.368555733381445 T-
critical = 2.1603686564610127 p-value =
4.680215495139796e-10
```

Reject the null hypothesis

Hypothesis test foí βo

- Ho: βo = 0

```
ft = (np.sum((y-y_pred)**2))/(n-2) st =
(1/n)+((m_x**2)/(np.sum((x-m_x)**2))) se_b0 =
np.sqrt(ft*st)
```

```
tstats = b_0/se_b0 print('t-statistics =
',tstats)   import   scipy.stats   as   st
print('t-critical = ',st.t.ppf(0.975,n-2))
t-statistics   =   6.273361626206658   t-
critical = 2.1603686564610127
```

Reject the null hypothesis

```
exp_var = np.sum((y_pred-m_y)**2)
tot_var = np.sum((y-m_y)**2) r2 =
exp_var/tot_var
print('Coefficient of determination = ',r2)
Coefficient   of   determination   =   0.9537250641344718   from
sklearn.linear_model      import      LinearRegression      y      =
np.array([60,90,110,125,150,170,180,200,220,230,240,250,255,260,
260]) x =
np.array([100,150,200,250,300,350,400,450,500,550,600,650,700,75
0,800]).reshape((-1,1)) model =
LinearRegression().fit(x,y)
print(model.intercept_)
print(model.coef_) r2 =
model.score(x,y) print(r2)
55.684523809523796
[0.29107143]
0.9537250641344719
```

Conclusion	Hence we have studied and implemented Correlation and Regression Analysis.

Signature of Faculty