

✓ Correlation, Regression and Probability Distribution

Tutorial-3

Name : Ritesh Jha

Roll No : 16010423076

Batch : B3

Branch : IT

✓ Q.1 For the following data set

{(25,70), (28,80), (32,85), (36,75), (38,59), (40,65), (39,78), (42,50), (41,54), (45,66)}

(i) Draw the scatter diagram (ii) Find the correlation coefficients

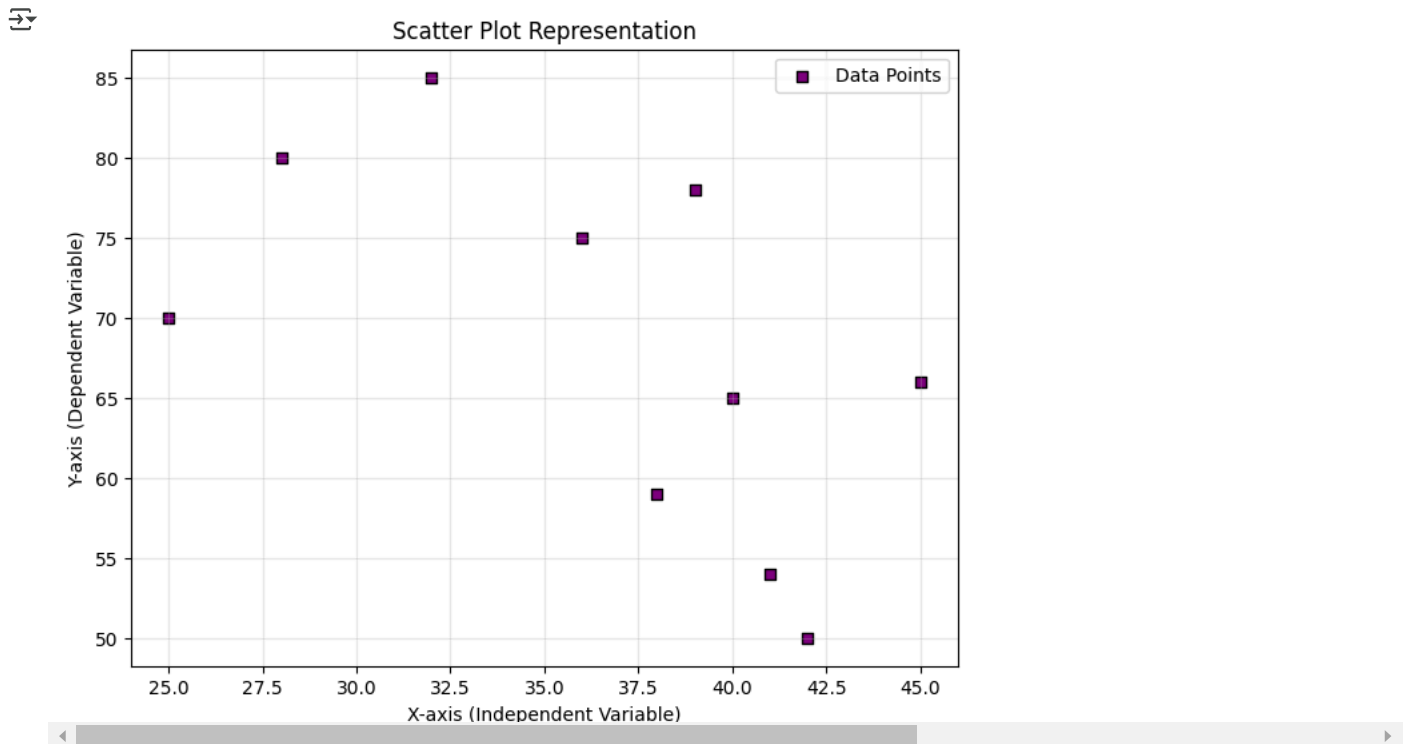
(iii) Find both the regression lines (iv) Plot both regression lines together

(v) Find the error for both regression lines

```
import numpy as np
import matplotlib.pyplot as plt

# (i) Define the dataset and scatter plot
data = [(25,70), (28,80), (32,85), (36,75), (38,59), (40,65), (39,78), (42,50), (41,54), (45,66)]
x = np.array([pair[0] for pair in data]) # Extracting X values
y = np.array([pair[1] for pair in data]) # Extracting Y values

plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='purple', edgecolors='black', marker='s', label='Data Points')
plt.xlabel('X-axis (Independent Variable)')
plt.ylabel('Y-axis (Dependent Variable)')
plt.title('Scatter Plot Representation')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



```
# (ii) Compute and display correlation coefficient
mean_x, mean_y = np.mean(x), np.mean(y)
correlation = np.corrcoef(x, y)[0, 1]
print(f"Correlation Coefficient: {correlation:.6f}")
```

Correlation Coefficient: -0.576431

```
# (iii) Compute regression lines
b_yx = np.sum((x - mean_x) * (y - mean_y)) / np.sum((x - mean_x) ** 2)
c_yx = mean_y - b_yx * mean_x

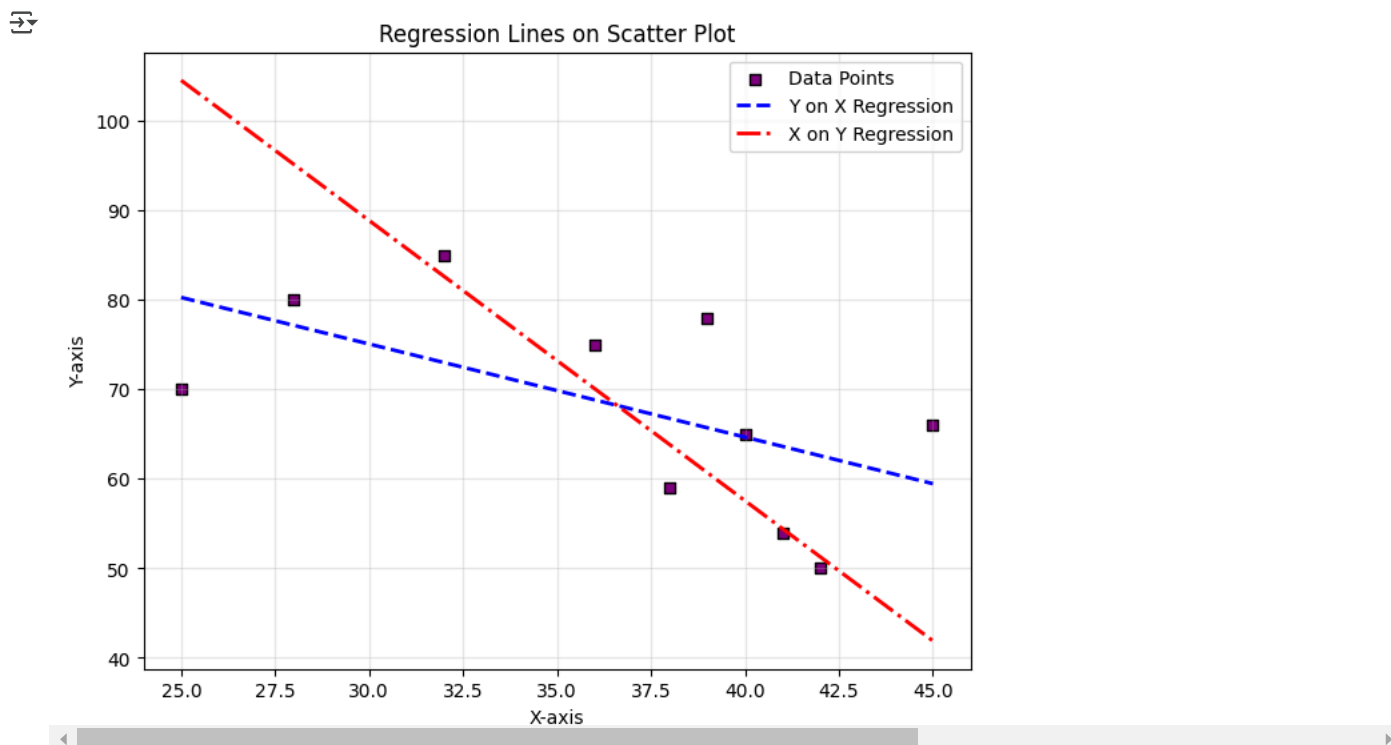
b_xy = np.sum((x - mean_x) * (y - mean_y)) / np.sum((y - mean_y) ** 2)
c_xy = mean_x - b_xy * mean_y
```

```
print(f"Regression Line (Y on X): y = {b_yx:.3f}x + {c_yx:.3f}")
print(f"Regression Line (X on Y): x = {b_xy:.3f}y + {c_xy:.3f}")
```

```
↗ Regression Line (Y on X): y = -1.040x + 106.270
Regression Line (X on Y): x = -0.319y + 58.386
```

```
# (iv) Plot both regression lines
x_range = np.linspace(min(x), max(x), 100)
y_reg_yx = b_yx * x_range + c_yx
y_reg_xy = (x_range - c_xy) / b_xy

plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='purple', edgecolors='black', marker='s', label='Data Points')
plt.plot(x_range, y_reg_yx, 'b--', linewidth=2, label='Y on X Regression')
plt.plot(x_range, y_reg_xy, 'r-.', linewidth=2, label='X on Y Regression')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Regression Lines on Scatter Plot')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



```
# (v) Compute and display Mean Squared Error for both regression lines
pred_y = b_yx * x + c_yx
error_yx = (y - pred_y) ** 2
mss_yx = error_yx / len(data)

pred_x = b_xy * y + c_xy
error_xy = (x - pred_x) ** 2
mss_xy = error_xy / len(data)

df = pd.DataFrame({
    "X": x, "Y": y,
    "Predicted Y (Y on X)": pred_y, "Error (Y on X)^2": error_yx, "MSS (Y on X)": mss_yx,
    "Predicted X (X on Y)": pred_x, "Error (X on Y)^2": error_xy, "MSS (X on Y)": mss_xy
})
print(df.to_string(index=False))

print(f"Total MSS (Y on X) : {np.sum(mss_yx)}")
print(f"Total MSS (X on Y) : {np.sum(mss_xy)}")
```

	X	Y	Predicted Y (Y on X)	Error (Y on X)^2	MSS (Y on X)	Predicted X (X on Y)	Error (X on Y)^2	MSS (X on Y)
	25	70	80.266015	105.391068	10.539107	36.025008	121.550809	12.155081
	28	80	77.145494	8.148204	0.814820	32.830610	23.334795	2.333479
	32	85	72.984799	144.365052	14.436505	31.233411	0.587658	0.058766
	36	75	68.824104	38.141689	3.814169	34.427809	2.471784	0.247178
	38	59	66.743757	59.965769	5.996577	39.538846	2.368048	0.236805
	40	65	64.663409	0.113293	0.011329	37.622207	5.653898	0.565390
	39	78	65.703583	151.201870	15.120187	33.469490	30.586543	3.058654
	42	50	62.583062	158.333447	15.833345	42.413805	0.171234	0.017123
	41	54	63.623236	92.606664	9.260666	41.136045	0.018508	0.001851
	45	66	59.462541	42.738374	4.273837	37.302768	59.247387	5.924739
Total MSS (Y on X) :			80.10054288816502					
Total MSS (X on Y) :			24.599066355451814					

✓ Q2 If X is Binomial Distribution B(n,p) where n=15 p=0.45

Write program to evaluate and print (i) P(X=10) (ii) P(X≤12) (iii) P(X≥9)

```
from scipy.stats import binom

# Parameters
n, p = 15, 0.45

# (i) Probability of exactly X = 10
prob_10 = binom.pmf(10, n, p)
print(f"P(X = 10): {prob_10:.6f}")

# (ii) Probability of X ≤ 12
prob_leq_12 = binom.cdf(12, n, p)
print(f"P(X ≤ 12): {prob_leq_12:.6f}")

# (iii) Probability of X ≥ 9
prob_geq_9 = 1 - binom.cdf(8, n, p)
print(f"P(X ≥ 9): {prob_geq_9:.6f}")
```

✓ Q3 If X is Poisson Distribution with mean 5

Write program to evaluate and print (i) P(X=2) (ii) P(X≤4) (iii) P(1≤X≤3)

```
from scipy.stats import poisson

lambda_value = 5 # Mean of Poisson distribution

# (i) P(X = 2)
prob_x2 = poisson.pmf(2, lambda_value)
print(f"P(X = 2): {prob_x2:.5f}")

# (ii) P(X ≤ 4)
prob_xleq4 = poisson.cdf(4, lambda_value)
print(f"P(X ≤ 4): {prob_xleq4:.5f}")

# (iii) P(1 ≤ X ≤ 3)
prob_x_range = poisson.cdf(3, lambda_value) - poisson.cdf(0, lambda_value)
print(f"P(1 ≤ X ≤ 3): {prob_x_range:.5f}")
```

✓ Q.4 If X is Uniform Distribution over the range (10,90). Write programme to evaluate and print

(i) P(X<29) (ii) P(X>34) (iii) P(70< X<80)

```
from scipy.stats import uniform

# Range parameters
```

```
start, end = 10, 90
```

```
# (i) P(X < 29)
prob_x29 = uniform.cdf(29, loc=start, scale=end - start)
print(f"P(X < 29): {prob_x29:.6f}")
```

```
↩ P(X < 29): 0.237500
```

```
# (ii) P(X > 34)
prob_x34 = 1 - uniform.cdf(34, loc=start, scale=end - start)
print(f"P(X > 34): {prob_x34:.6f}")
```

```
↩ P(X > 34): 0.700000
```

```
# (iii) P(70 < X < 80)
prob_x70_80 = uniform.cdf(80, loc=start, scale=end - start) - uniform.cdf(70, loc=start, scale=end - start)
print(f"P(70 < X < 80): {prob_x70_80:.6f}")
```

```
↩ P(70 < X < 80): 0.125000
```

✓ Q.5 If X is Exponential Distribution with mean 20. Write programme to evaluate and print

(i) $P(X < 10)$ (ii) $P(X > 7)$ (iii) $P(11 < X < 16)$

Find value of k such that $P(X < k) = 0.6$

```
from scipy.stats import expon
```

```
mean = 20
lambda_value = 1 / mean
```

```
# (i) P(X < 10)
prob_x10 = expon.cdf(10, scale=1/lambda_value)
print(f"P(X < 10): {prob_x10:.5f}")
```

```
↩ P(X < 10): 0.39347
```

```
# (ii) P(X > 7)
prob_x7 = 1 - expon.cdf(7, scale=1/lambda_value)
print(f"P(X > 7): {prob_x7:.5f}")
```

```
↩ P(X > 7): 0.70469
```

```
# (iii) P(11 < X < 16)
prob_x11_16 = expon.cdf(16, scale=1/lambda_value) - expon.cdf(11, scale=1/lambda_value)
print(f"P(11 < X < 16): {prob_x11_16:.5f}")
```

```
↩ P(11 < X < 16): 0.12762
```

```
# Find k such that P(X < k) = 0.6
k = expon.ppf(0.6, scale=1/lambda_value)
print(f"Value of k: {k:.5f}")
```

```
↩ Value of k: 18.32581
```

✓ Q.6 If X is Normal Distribution with mean 40 and standard deviation 10. Write programme to evaluate and print (i) $P(X < 38)$ (ii) $P(X > 55)$ (iii) $P(20 < X < 70)$.

Find value of k1 such that $P(X < k1) = 0.3$. Also find k2 such that $P(X > k2) = 0.8$


```
from scipy.stats import norm
```

```
mean, std_dev = 40, 10
```


```
# (i) P(X < 38)
prob_x38 = norm.cdf(38, loc=mean, scale=std_dev)
print(f"P(X < 38): {prob_x38:.6f}")
```

```
↩ P(X < 38): 0.420740
```

```
# (ii) P(X > 55)
prob_x55 = 1 - norm.cdf(55, loc=mean, scale=std_dev)
print(f"P(X > 55): {prob_x55:.6f}")
```


 `P(X > 55): 0.066807`

```
# (iii) P(20 < X < 70)
prob_x20_70 = norm.cdf(70, loc=mean, scale=std_dev) - norm.cdf(20, loc=mean, scale=std_dev)
print(f"P(20 < X < 70): {prob_x20_70:.6f}")
```

 `P(20 < X < 70): 0.975900`

```
# k1 such that P(X < k1) = 0.3
k1 = norm.ppf(0.3, loc=mean, scale=std_dev)
print(f"Value of k1: {k1:.3f}")

# k2 such that P(X > k2) = 0.8
k2 = norm.ppf(1 - 0.8, loc=mean, scale=std_dev)
print(f"Value of k2: {k2:.3f}")
```

 `Value of k1: 34.756`
`Value of k2: 31.584`