Tutorial-3: Correlation, Regression and Probability Distribution Using Python

Name: Aaryan Sharma

Roll No: 16010123012

Batch: A1

Branch: COMPS

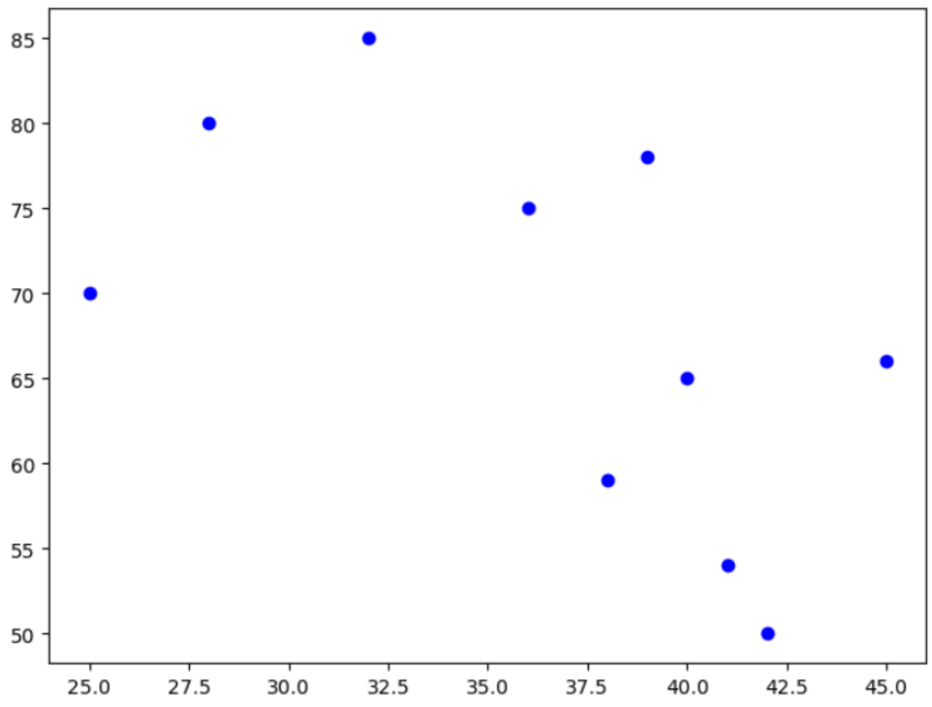
Q.1 For the following data set

{(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

[2]:
```
import numpy as np
import matplotlib.pyplot as plt

# Data
data = [(25,70), (28,80), (32,85), (36,75), (38,59), (40,65), (39,78), (42,50), (41,54), (45,66)]
x = np.array([point[0] for point in data])  # Independent variable
y = np.array([point[1] for point in data])  # Dependent variable

plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='blue', label='Data Points')
```

[2]:
```
<matplotlib.collections.PathCollection at 0x412c118>
```



(ii) Find the correlation coefficients

[3]:
```
# Calculate means
mean_x = np.mean(x)
mean_y = np.mean(y)

# Calculate the correlation coefficient
correlation_coefficient = np.corrcoef(x, y)[0,1]
print("Correlation Coefficient:", correlation_coefficient)
```

```
Correlation Coefficient: -0.5764311756246667
```

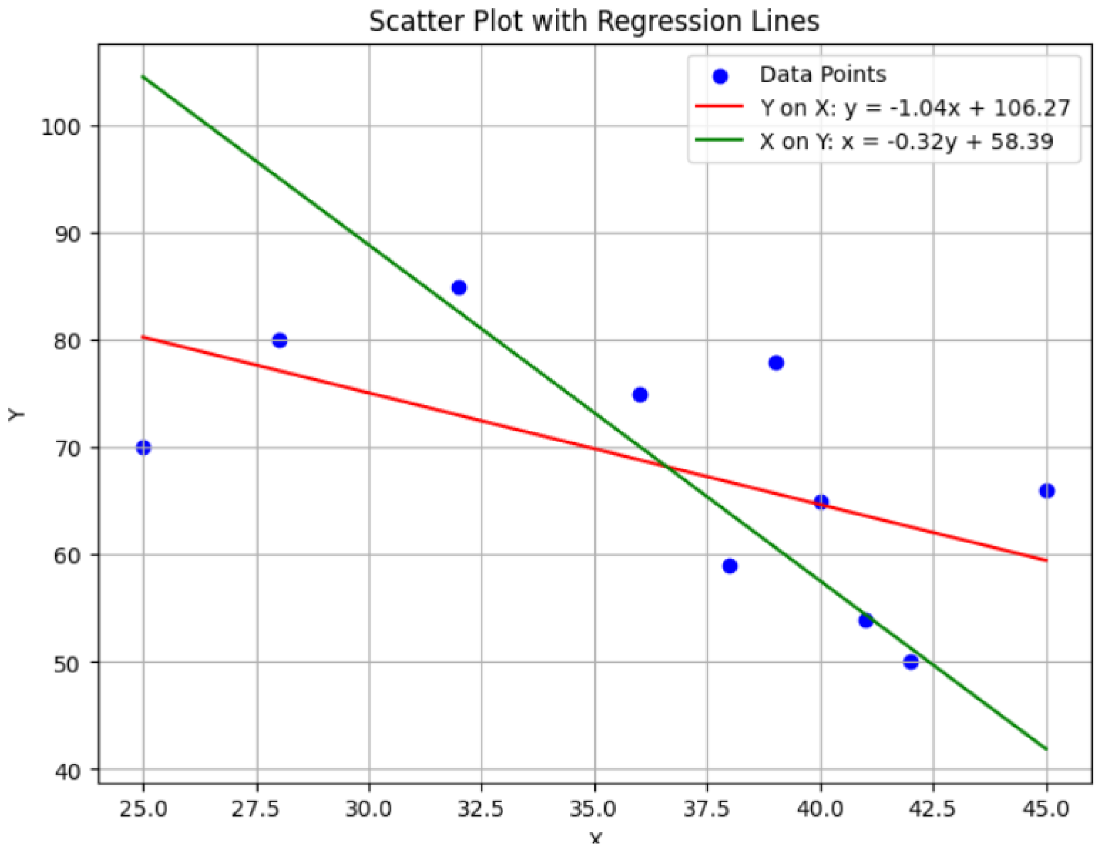
(iii) Find both the regression lines

[4]:
```
# Regression coefficients for Y on X (y = mx + c)
b_yx = np.sum((x - mean_x) * (y - mean_y)) / np.sum((x - mean_x)**2)
c_yx = mean_y - b_yx * mean_x

# Regression coefficients for X on Y (x = my + c)
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 results
print(f"Regression line Y on X: y = {b_yx:.2f}x + {c_yx:.2f}")
print(f"Regression line X on Y: x = {b_xy:.2f}y + {c_xy:.2f}")
```

```
Regression line Y on X: y = -1.04x + 106.27
Regression line X on Y: x = -0.32y + 58.39
```

(iv) Plot both regression lines together

[5]:
```
# Generate points for regression lines
x_vals = np.linspace(min(x), max(x), 100)
y_vals_yx = b_yx * x_vals + c_yx  # Regression line Y on X
y_vals_xy = (x_vals - c_xy) / b_xy  # Regression line X on Y

# Plotting
def plot_regression():
    plt.figure(figsize=(8, 6))

    # Scatter plot of data points
    plt.scatter(x, y, color='blue', label='Data Points')

    # Regression line Y on X
    plt.plot(x_vals, y_vals_yx, color='red', label='Y on X: y = {:.2f}x + {:.2f}'.format(b_yx, c_yx))

    # Regression line X on Y
    plt.plot(x_vals, y_vals_xy, color='green', label='X on Y: x = {:.2f}y + {:.2f}'.format(b_xy, c_xy))


    # Plot labels and legend
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title('Scatter Plot with Regression Lines')
    plt.legend()
    plt.grid()
    plt.show()

plot_regression()
```



(v) Find the error for both regression lines

[6]:

```
# Calculate Mean Sum of Squares (MSS) for each value
import pandas as pd
n = len(data)
# For Y on X
predicted_y = b_yx * x + c_yx
errors_yx = (y - predicted_y)**2
mss_yx_values = errors_yx / n

# For X on Y
predicted_x = b_xy * y + c_xy
errors_xy = (x - predicted_x)**2
mss_xy_values = errors_xy / n



# Create a table showing MSS for each data point
data_table = {
    "X": x,
    "Y": y,
    "Predicted Y (Y on X)": predicted_y,
    "Error (Y on X)^2": errors_yx,
    "MSS (Y on X)": mss_yx_values,
    "Predicted X (X on Y)": predicted_x,
    "Error (X on Y)^2": errors_xy,
    "MSS (X on Y)": mss_xy_values,

}


# Convert to DataFrame for display
results_df = pd.DataFrame(data_table)

# Print results table
print(results_df)
Total_MSS_y_on_x = np.sum(mss_yx_values)
Total_MSS_x_on_y = np.sum(mss_xy_values)

print("Total MSS (Y on X) :",Total_MSS_y_on_x)
print("Total MSS (X on Y) :",Total_MSS_x_on_y)
```

```
    X   Y  Predicted Y (Y on X)  Error (Y on X)^2  MSS (Y on X)  \
0  25  70             80.266015        105.391068     10.539107   
1  28  80             77.145494          8.148204      0.814820   
2  32  85             72.984799        144.365052     14.436505   
3  36  75             68.824104         38.141689      3.814169   
4  38  59             66.743757         59.965769      5.996577   
5  40  65             64.663409          0.113293      0.011329   
6  39  78             65.703583        151.201870     15.120187   
7  42  50             62.583062        158.333447     15.833345   
8  41  54             63.623236         92.606664      9.260666   
9  45  66             59.462541         42.738374      4.273837   

   Predicted X (X on Y)  Error (X on Y)^2  MSS (X on Y)  
0             36.025008        121.550809     12.155081  
1             32.830610         23.334795      2.333479  
2             31.233411          0.587658      0.058766  
3             34.427809          2.471784      0.247178  
4             39.538846          2.368048      0.236805  
5             37.622207          5.653898      0.565390  
6             33.469490         30.586543      3.058654  
7             42.413805          0.171234      0.017123  
8             41.136045          0.018508      0.001851  
9             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)

[7]:
```python
from scipy.stats import binom

# Parameters
n = 15
p = 0.45

# Calculations
a = binom.pmf(10, n, p)          # P(X=10)
b = binom.cdf(34, n, p)          # P(X≤34)
c = 1 - binom.cdf(9, n, p)     # P(X≥9)

# Print results
print(f"P(X=10) = {a}")
print(f"P(X≤34) = {b}")
print(f"P(X≥9) = {c}")
```

```
P(X=10) = 0.051462859925538396
P(X≤34) = 1.0
P(X≥9) = 0.07692871333818019
```

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)

```
[11]: from scipy.stats import poisson

# Parameters
m = 25   # Mean (λ) of the Poisson distribution

# Calculations
a = poisson.pmf(2, m)                     # P(X=2)
b = poisson.cdf(4, m)                     # P(X≤4)
c = poisson.cdf(3, m) - poisson.cdf(0, m)  # P(1≤X≤3)

# Print results
print(f"P(X=2) = {a}")
print(f"P(X≤4) = {b}")
print(f"P(1≤X≤3) = {c}")
```

```
P(X=2) = 4.339982457801251e-09
P(X≤4) = 2.669083424904495e-07
P(1≤X≤3) = 4.085370153610248e-08
```

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)

```
[9]: from scipy.stats import uniform

# Parameters for the uniform distribution
a = 10  # Start of the range
b = 90  # End of the range

# Calculations
p1 = uniform.cdf(10, loc=a, scale=b-a)              # P(X < 10)
p2 = 1 - uniform.cdf(7, loc=a, scale=b-a)           # P(X > 7)
p3 = uniform.cdf(80, loc=a, scale=b-a) - uniform.cdf(70, loc=a, scale=b-a)  # P(70 < X < 80)

# Print results
print(f"P(X < 10) = {p1}")
print(f"P(X > 7) = {p2}")
print(f"P(70 < X < 80) = {p3}")
```

```
P(X < 10) = 0.0
P(X > 7) = 1.0
P(70 < X < 80) = 0.125
```

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.

```
[12]: from scipy.stats import expon

# Parameters for the exponential distribution
mean = 20
lambda_param = 1 / mean  # Rate parameter λ

# Calculations
a = expon.cdf(10, scale=1/lambda_param)                      # P(X < 10)
b = 1 - expon.cdf(7, scale=1/lambda_param)                   # P(X > 7)
c = expon.cdf(16, scale=1/lambda_param) - expon.cdf(11, scale=1/lambda_param)  # P(11 < X < 16)
k = expon.ppf(0.6, scale=1/lambda_param)                     # k such that P(X < k) = 0.6

# Print results
print(f"P(X < 10) = {a}")
print(f"P(X > 7) = {b}")
print(f"P(11 < X < 16) = {c}")
print(f"The value of k is {k}")
```

```
P(X < 10) = 0.3934693402873666
P(X > 7) = 0.7046880897187133
P(11 < X < 16) = 0.1276208462632651
The value of k is 18.3258146374831
```

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

```
[13]: from scipy.stats import norm

# Parameters for the normal distribution
mean = 40
std_dev = 10

# Calculations
a = norm.cdf(38, loc=mean, scale=std_dev)             # P(X < 38)
b = 1 - norm.cdf(55, loc=mean, scale=std_dev)         # P(X > 55)
c = norm.cdf(70, loc=mean, scale=std_dev) - norm.cdf(20, loc=mean, scale=std_dev)  # P(20 < X < 70)
k1 = norm.ppf(0.3, loc=mean, scale=std_dev)           # k1 such that P(X < k1) = 0.3
k2 = norm.ppf(0.8, loc=mean, scale=std_dev)           # k2 such that P(X > k2) = 0.8

# Print results
print(f"P(X < 38) = {a}")
print(f"P(X > 55) = {b}")
print(f"P(20 < X < 70) = {c}")
print(f"Value of k1 such that P(X < k1) = 0.3 is {k1}")
print(f"Value of k2 such that P(X > k2) = 0.8 is {k2}")
```

```
P(X < 38) = 0.42074029056089696
P(X > 55) = 0.06680720126885809
P(20 < X < 70) = 0.9758999700201907
Value of k1 such that P(X < k1) = 0.3 is 34.755994872919594
Value of k2 such that P(X > k2) = 0.8 is 48.41621233572914
```