Module 2: Data Analytics with Python - Statistics

Assignment 1: Basic Statistics & Linear Algebra

Objective: By the end of this lab, you will have a basic understanding of statistics and linear algebra concepts using Python. You will also learn how to work with libraries such as NumPy and SciPy for various statistical and mathematical computations.

Task 1: Setting Up and Installing Required Libraries

Instructions:

```
import sys
# Install numpy
!{sys.executable} -m pip install numpy
# Install scipy
!{sys.executable} -m pip install scipy
```

- Install the necessary Python libraries for statistical analysis: NumPy and SciPy.

Expected Output:

- Confirmation that NumPy and SciPy libraries have been installed successfully.

Task 2: Understanding Measures of Central Tendency

Instructions:

- Create a basic NumPy array x with the values [1, 2, 3, 5, 5].
- Calculate and print the mean of x.
- Calculate and print the median of x.
- Calculate and print the mode of x using both NumPy and SciPy.

```
import numpy as np
from scipy import stats

# Creating a Basic Array in NumPy
X = np.array([1, 2, 3, 5, 5])

# Mean using NumPy
print("Data", X)
print("Mean:", np.mean(X))

# Median using NumPy
print("Median:", np.median(X))

# Mode using NumPy
vals, counts = np.unique(X, return_counts=True)
index = np.argmax(counts)
print("Mode (NumPy):", vals[index])

# Mode using SciPy
print("Mode (SciPy):", stats.mode(X))
```

Expected Output:

- Data: [1 2 3 5 5]
- Mean: 3.2
- Median: 3.0
- Mode (NumPy): 5
- Mode (SciPy): ModeResult(mode=array([5]), count=array([2]))

Task 3: Exploring Measures of Spread

Instructions:

- Use the array x from Task 2.
- Calculate and print the range of x.
- Calculate and print the quartiles and interquartile range (IQR) of x.
- Calculate and print the 20th percentile of x.
- Calculate and print the variance and standard deviation of x.

```
# Range
print("Range:", X.max() - X.min())

# Quartiles and Interquartile Range (IQR)
data1 = np.array([1, 3, 4, 5, 5, 6, 7, 11])
Q1 = np.quantile(data1, .25, interpolation='midpoint')
Q2 = np.quantile(data1, .50, interpolation='midpoint')
Q3 = np.quantile(data1, .75, interpolation='midpoint')
IQR = Q3 - Q1
print("Quartile 1:", Q1)
print("Quartile 2:", Q2)
print("Quartile 3:", Q3)
print("Interquartile Range:", IQR)

# Percentile
percentile_20 = np.percentile(data1, 20)
print("20th Percentile:", round(percentile_20, 2))

# Variance and Standard Deviation
print("Variance:", np.var(data1, ddof=1))
print("Standard Deviation:", np.std(data1, ddof=1))
```

Expected Output:

- Range: 4
- Quartile 1: 3.5
- Quartile 2: 5.0
- Quartile 3: 6.5
- Interquartile Range: 3.0
- 20th Percentile: 3.4
- Variance: 8.785714285714286
- Standard Deviation: 2.9640705601780613

Task 4: Analyzing Covariance and Correlation

Instructions:

- Create two NumPy arrays `x` and `y` with values [1,2,3,4,5,6,7,8,9] and [9,8,7,6,5,4,3,2,1], respectively.
- Calculate and print the covariance between `x` and `y`.
- Calculate and print the correlation coefficient matrix of `x` and `y`.

```
# Covariance
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([9, 8, 7, 6, 5, 4, 3, 2, 1])
print("Covariance:", np.cov(x, y)[0, 1])

# Correlation
print("Correlation Coefficient Matrix:", np.corrcoef(x, y))
```

Expected Output:

- Covariance: -7.5
- Correlation Coefficient Matrix: [[1. -1.], [-1. 1.]]

Task 5: Understanding Kurtosis and Skewness

Instructions:

- Create a sequence of data points and calculate the kurtosis using the scipy.stats.kurtosis function.
- Generate a histogram to visualize skewness in the data.

```
from scipy import stats
import matplotlib.pyplot as plt

# Kurtosis
data2 = np.linspace(-5, 5, 100)
print("Kurtosis:", stats.kurtosis(data2))

# Skewness
numValues = 10000
maxValue = 100
skewness = -5  # Negative values are left skewed
random = stats.skewnorm.rvs(a=skewness, loc=maxValue,
size=numValues)
random = random - min(random)
random = random / max(random)
random = random * maxValue
plt.hist(random, 30, density=True, color='red', alpha=0.1)
plt.show()
print("Skewness:", stats.skew(data2))
```

Expected Output:

- Kurtosis: -1.2002400240024003
- Histogram showing skewness (visual output)
- Skewness: -2.8663736585314984e-17

Task 6: Working with Eigenvectors and Eigenvalues

Instructions:

- Create a 3x3 matrix x using NumPy.
- Calculate the eigenvectors and eigenvalues of x.
- Apply a linear transformation to x and calculate the new eigenvectors and eigenvalues.

```
import numpy as np

x = np.matrix([[4, 2, 3], [6, 7, 5], [3, 6, 1]])
e = np.eye(3)
m = x + 45 * e

# Eigenvectors and Eigenvalues
eig_values, eig_vectors = np.linalg.eig(x)
print("Eigenvalues:", eig_values)
print("Eigenvectors:", eig_vectors)

eig_values_m, eig_vectors_m = np.linalg.eig(m)
print("Transformed Eigenvalues:", eig_values_m)
print("Transformed Eigenvectors:", eig_vectors_m)
```

Expected Output:

- Eigenvalues: [12.76222365, 1.17374428, -1.93596793]
- Eigenvectors: (Displayed as a matrix)
- Transformed Eigenvalues: [57.76222365, 46.17374428, 43.06403207]
- Transformed Eigenvectors: (Displayed as a matrix)