Textual Similarity

Cosine Distance -

```python
import numpy as np
from sklearn.metrics.pairwise import cosine_distances

# Sample vectors A and B
A = np.array([1, 2, 3])
B = np.array([4, 5, 6])

# Calculate cosine distance
cosine_dist = cosine_distances([A], [B])[0][0]

print(f"Cosine Distance: {cosine_dist:.2f}")
```

Edit Distance :

```python
def edit_distance(str1, str2):
    m = len(str1)
    n = len(str2)

    # Create a table to store the edit distances
    dp = [[0] * (n + 1) for _ in range(m + 1)]

    # Initialize the first row and column
    for i in range(m + 1):
        dp[i][0] = i
    for j in range(n + 1):
        dp[0][j] = j

    # Fill in the table using dynamic programming
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            cost = 0 if str1[i - 1] == str2[j - 1] else 1
            dp[i][j] = min(
                dp[i - 1][j] + 1,  # Deletion
                dp[i][j - 1] + 1,  # Insertion
                dp[i - 1][j - 1] + cost  # Substitution
            )

    # The edit distance is in the bottom-right cell of the table
    return dp[m][n]

# Example usage:
str1 = "kitten"
str2 = "sitting"
```

```
distance = edit_distance(str1, str2)
print(f"Edit Distance between '{str1}' and '{str2}': {distance}")
```

SKEWNESS

import numpy as np

from scipy.stats import skew

Sample dataset (replace this with your own data)

data = [10, 20, 25, 30, 35, 40, 45, 50, 60, 70]

Calculate skewness using NumPy and SciPy

mean = np.mean(data)

median = np.median(data)

std_dev = np.std(data)

skewness = skew(data)

print(f"Mean: {mean}")

print(f"Median: {median}")

print(f"Standard Deviation: {std_dev}")

print(f"Skewness: {skewness}")

Pearson Coefficient of Correlation:

```
import numpy as np

# Sample data for X and Y (replace with your own data)
X = [10, 20, 30, 40, 50]
Y = [15, 25, 35, 45, 55]

# Calculate Pearson's correlation coefficient
correlation_coefficient = np.corrcoef(X, Y)[0, 1]

print(f"Pearson's Correlation Coefficient: {correlation_coefficient}")
```

Bowley's coefficient, also known as the Bowley's skewness coefficient, is another measure of skewness used in statistics. It is defined as:

$$B = \frac{Q_1 + Q_3 - 2Q_2}{Q_3 - Q_1}$$

Where:

- Q_1 is the first quartile (25th percentile).
- Q_2 is the second quartile (median).
- Q_3 is the third quartile (75th percentile).

Python Code:

```
import numpy as np

# Sample dataset (replace this with your own data)
data = [10, 20, 25, 30, 35, 40, 45, 50, 60, 70]

# Calculate the quartiles using numpy.percentile
q1 = np.percentile(data, 25)
q2 = np.percentile(data, 50)
q3 = np.percentile(data, 75)

# Calculate Bowley's coefficient
bowley_coefficient = (q1 + q3 - 2 * q2) / (q3 - q1)

print(f"First Quartile (Q1): {q1}")
print(f"Second Quartile (Q2): {q2}")
print(f"Third Quartile (Q3): {q3}")
print(f"Bowley's Coefficient: {bowley_coefficient}")
```

```
Bowleys Coeffcient for Group Data:
```

Bowley's coefficient can also be calculated for grouped data, which means data that is divided into intervals or bins. To calculate Bowley's coefficient for grouped data, you'll need to make some modifications to the formula. Here's the formula for Bowley's coefficient for grouped data:

$$B = \frac{3(M - m)}{Q_3 - Q_1}$$

Where:

- M is the midpoint of the interval containing the median.
- m is the midpoint of the interval preceding the one containing the median.
- Q_1 is the lower quartile, and Q_3 is the upper quartile.

To calculate Bowley's coefficient for grouped data in Python, you'll need the following information:

1. The frequency of each interval (how many data points fall into each interval).
2. The midpoints of the intervals.
3. The lower quartile (Q1) and upper quartile (Q3).

```
# Frequency of each interval
frequencies = [5, 10, 15, 20, 10]

# Midpoints of the intervals
midpoints = [10, 20, 30, 40, 50]

      # Lower quartile (Q1) and upper quartile (Q3)
      q1 = 25  # Replace with your Q1 value
      q3 = 45  # Replace with your Q3 value

      # Calculate M and m based on the interval containing the
median
      median_interval_index = midpoints.index(q3)  # Replace with
the correct index of the median interval
      M = midpoints[median_interval_index]
      m = midpoints[median_interval_index - 1]

      # Calculate Bowley's coefficient
      bowley_coefficient = (3 * (M - m)) / (q3 - q1)

      print(f"M: {M}")
      print(f"m: {m}")
      print(f"Q1: {q1}")
      print(f"Q3: {q3}")
print(f"Bowley's Coefficient: {bowley_coefficient}")
```

Central moments are statistical measures that provide information about the shape and distribution of data around its mean. The central moments of data are typically calculated using the following formulas:

- The first central moment (mean) is simply the mean of the data.
- The second central moment (variance) measures the spread or dispersion of the data.
- The third central moment measures the asymmetry or skewness of the data distribution.
- The fourth central moment measures the kurtosis, which indicates the tails and the presence of outliers in the data.

You can calculate these central moments in Python using NumPy. Here's a Python code snippet to calculate central moments for a given dataset

Python code:

```
import numpy as np
```

```
# Sample dataset (replace this with your own data)
data = [10, 20, 30, 40, 50]

# Calculate the mean (first central moment)
mean = np.mean(data)

# Calculate the variance (second central moment)
variance = np.var(data)

# Calculate the third central moment (skewness)
skewness = np.mean((data - mean) ** 3) / np.power(variance, 3/2)

# Calculate the fourth central moment (kurtosis)
kurtosis = np.mean((data - mean) ** 4) / np.power(variance, 2) - 3

print(f"Mean (First Central Moment): {mean}")
print(f"Variance (Second Central Moment): {variance}")
print(f"Skewness (Third Central Moment): {skewness}")
      print(f"Kurtosis (Fourth Central Moment): {kurtosis}")
      ------------------------------------------------------
```