Genztechs
Assignment 4
Data Science Internship

1. Understand all the calculations of these concepts from any resource and practice them on notebook with any example. Scan all those practice notes from the cam scanner, make pdf and submit it.

2. Code is given for all of the concepts, get understanding of code from here and implement by yourself. Make a .py / .ipynp file for these practice code chunks.

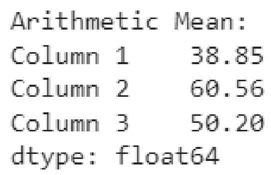
3. In the end questions are given. Implement them by yourself and submit the .py / .ipynb file.

Topics

Statistical Visualization

1. Arithmetic Mean

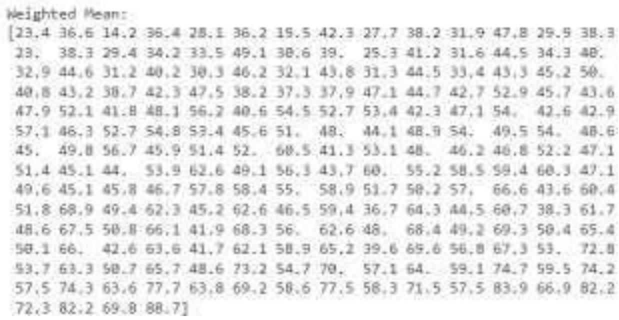
```
import pandas as pd

import numpy as np

from scipy.stats import trim_mean, mode
from sklearn.linear_model import LinearRegression
# Read the CSV file
data = pd.read_csv('/content/numbers.csv')
# Calculate Arithmetic Mean
arithmetic_mean = data.mean()
print("Arithmetic Mean:")
print(arithmetic_mean)
```



```
Arithmetic Mean:
Column 1    38.85
Column 2    60.56
Column 3    50.20
dtype: float64
```

2. Weighted Mean

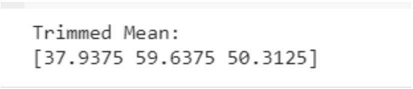
```
weights = [0.3, 0.4, 0.3]  # Example weights for each column
weighted_mean = np.average(data, weights=weights, axis=1)
print("Weighted Mean:")
print(weighted_mean)
```



```
Weighted Mean:
[23.4 36.6 14.2 36.4 28.1 36.2 15.5 42.3 27.7 38.2 31.9 47.8 29.5 38.3
 23.  38.3 29.4 34.2 33.5 49.1 30.6 39.  25.3 41.2 31.6 44.5 34.3 40.
 32.9 44.6 31.2 40.2 30.3 46.2 32.1 43.8 31.3 44.5 33.4 43.3 45.2 50.
 40.8 43.2 38.7 42.3 47.5 38.2 37.3 37.9 47.1 44.7 42.7 52.9 45.7 43.6
 47.9 52.1 41.8 48.1 56.2 40.6 54.5 52.7 53.4 42.3 47.1 54.  42.6 42.9
 57.1 46.3 52.7 54.8 53.4 45.6 51.  48.  44.1 48.9 54.  49.5 54.  48.6
 45.  49.8 56.7 45.9 51.4 52.  60.5 41.3 53.1 48.  46.2 46.8 52.2 47.1
 51.4 45.1 44.  53.9 62.6 49.1 56.3 43.7 60.  55.2 58.5 59.4 60.3 47.1
 49.6 45.1 45.8 46.7 57.8 58.4 55.  58.9 51.7 50.2 57.  66.6 43.6 60.4
 51.8 68.9 49.4 62.3 45.2 62.6 46.5 59.4 36.7 64.3 44.5 60.7 38.3 61.7
 48.6 67.5 50.8 66.1 41.9 68.3 56.  62.6 48.  68.4 49.2 69.3 50.4 65.4
 50.1 66.  42.6 63.6 41.7 62.1 58.9 65.2 39.6 69.6 56.8 67.3 53.  72.8
 53.7 63.3 50.7 65.7 48.6 73.2 54.7 70.  57.1 64.  59.1 74.7 59.5 74.2
 57.5 74.3 63.6 77.7 63.8 69.2 58.6 77.5 58.3 71.5 57.5 83.9 66.9 82.2
 72.3 82.2 69.8 88.7]
```

3. Trimmed Mean

```
trimmed_mean = trim_mean(data, proportiontocut=0.1)
```



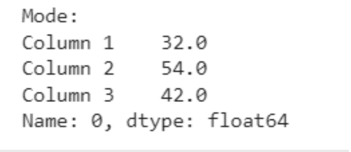
```
Trimmed Mean:
[37.9375 59.6375 50.3125]
```

4. Median

```
# Calculate Median
median = data.median()
print("Median:")
print(median)
```

5. Mode

```
import pandas as pd

from scipy.stats import mode

# Read the CSV file

data = pd.read_csv('/content/numbers.csv')  # Replace with the actual file path

# Calculate Mode

mode_values = data.mode().iloc[0]

print("Mode:")

print(mode_values)
```



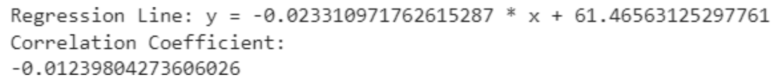
```
Mode:
Column 1    32.0
Column 2    54.0
Column 3    42.0
Name: 0, dtype: float64
```

6. Standard Deviation

```
std_dev = data.std()

print("Standard Deviation:")

print(std_dev)
```

7. Regression and Correlation

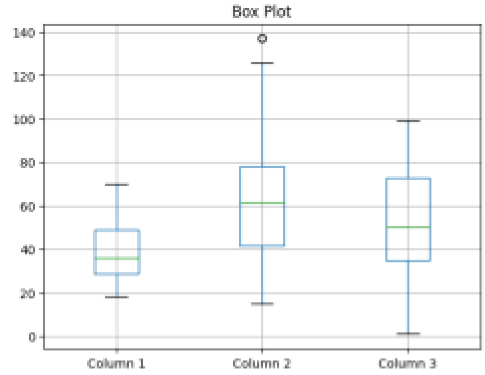
```
x = data.iloc[:, 0]  # Assuming the first column is the independent variable

y = data.iloc[:, 1]  # Assuming the second column is the dependent variable

regression_model = LinearRegression()

regression_model.fit(x.values.reshape(-1, 1), y)

slope = regression_model.coef_[0]

intercept = regression_model.intercept_

print("Regression Line: y =", slope, "* x +", intercept)

correlation_coefficient = data.corr().iloc[0, 1]

print("Correlation Coefficient:")

print(correlation_coefficient)
```



```
Regression Line: y = -0.023310971762615287 * x + 61.46563125297761
Correlation Coefficient:
-0.01239804273606026
```

8. Box Plot

```
import pandas as pd
import matplotlib.pyplot as plt
# Read the CSV file
data = pd.read_csv('/content/numbers.csv')
# Create a box plot
data.boxplot()
plt.title('Box Plot')
plt.show()
```



9. Euclidean distance

```
import numpy as np
from scipy.spatial.distance import euclidean
def calculate_euclidean_distance(vector1, vector2):
    return euclidean(vector1, vector2)

# Example usage
vector_a = np.array([1, 2, 3])
vector_b = np.array([4, 5, 6])

# Calculate Euclidean distance
euclidean_distance = calculate_euclidean_distance(vector_a, vector_b)

# Print the result
```

```
print("Euclidean Distance:", euclidean_distance)
```

```
Euclidean Distance: 5.196152422706632
```

10.Z Score

```
from scipy.stats import zscore


# Read the CSV file

data = pd.read_csv('/content/numbers.csv')

# Calculate z-scores

z_scores = data.apply(zscore)

print("Z-Scores:")

print(z_scores)
```

```
Z-Scores:
     Column 1  Column 2  Column 3
0   -1.424569 -1.738999 -0.434801
1   -1.281035 -1.738999  1.195704
2   -1.352802 -1.700830 -1.715913
3   -1.137502 -1.700830  1.040418
4   -0.563369 -1.662660 -0.395980
..        ...       ...       ...
```

```
195 -0.276302  2.268791  1.118061
196  0.441365  2.497807 -0.861839
197 -0.491602  2.497807  0.923953
198 -0.491602  2.917671 -1.250054
199 -0.635135  2.917671  1.273347
```

11. Manhatan Distance

```
import pandas as pd
from scipy.spatial.distance import pdist, squareform
# Read the CSV file
data = pd.read_csv('/content/numbers.csv')
# Calculate Manhattan distance matrix
manhattan_dist = pdist(data, metric='cityblock')
manhattan_dist_matrix = squareform(manhattan_dist)
# Print the distance matrix
print("Manhattan Distance Matrix:")
print(manhattan_dist_matrix)
```

Output

```
Manhattan Distance Matrix:
[[  0.   44.   35.  ... 159. 156. 177.]
 [ 44.    0.   77.  ... 129. 196. 133.]
 [ 35.   77.    0.  ... 190. 145. 208.]
 ...
 [159. 129. 190.  ...   0.   67.   22.]
 [156. 196. 145.  ...  67.    0.   67.]
 [177. 133. 208.  ...  22.   67.    0.]]
```

12. Minkowski distance

```
from scipy.spatial.distance import cdist
from scipy.spatial.distance import cdist
# Read the CSV file
data = pd.read_csv('/content/numbers.csv')
# Calculate Minkowski distance matrix
```

```
p = 2  # Choose the desired order of Minkowski distance (p=1 for Manhattan distance, p=2 for Euclidean distance)

minkowski_dist = cdist(data, data, metric='minkowski', p=p)

# Print the distance matrix

print("Minkowski Distance Matrix:")

print(minkowski_dist)
```

```
Minkowski Distance Matrix:
[[  0.          42.04759208  33.03028913 ... 117.1110584  124.47489707
  130.15759678]
 [ 42.04759208   0.          75.01333215 ... 111.7631424  137.74614332
  122.34786471]
 [ 33.03028913  75.01333215   0.         ... 129.87686476 122.18428704
  143.77065069]
 ...
 [117.1110584  111.7631424  129.87686476 ...   0.          57.07013229
   14.35270009]
 [124.47489707 137.74614332 122.18428704 ...  57.07013229   0.
   65.03076195]
 [130.15759678 122.34786471 143.77065069 ...  14.35270009  65.03076195
    0.        ]]
```

13. Supremum Distance

```
from scipy.spatial.distance import cdist


# Read the CSV file

data = pd.read_csv('/content/numbers.csv')


# Calculate Supremum distance matrix

supremum_dist = cdist(data, data, metric='chebyshev')


# Print the distance matrix

print("Supremum Distance Matrix:")

print(supremum_dist)
```

```
Supremum Distance Matrix:
[[  0.  42.  33. ... 111. 122. 122.]
 [ 42.   0.  75. ... 111. 122. 122.]
 [ 33.  75.   0. ... 110. 121. 121.]
 ...
 [111. 111. 110. ...   0.  56.  11.]
 [122. 122. 121. ...  56.   0.  65.]
 [122. 122. 121. ...  11.  65.   0.]]
```

14. Chi-Square

```
from scipy.stats import chi2_contingency

# Read the CSV file
data = pd.read_csv('/content/numbers.csv')

# Perform Chi-Square test
chi2_results = chi2_contingency(data)
chi2_statistic = chi2_results[0]
p_value = chi2_results[1]

print("Chi-Square Test:")
print("Chi-Square Statistic:", chi2_statistic)
print("P-value:", p_value)
```

```
Chi-Square Test:
Chi-Square Statistic: 4421.7047235660375
P-value: 0.0
```

15. covariance matrix

```
# Read the CSV file
data = pd.read_csv('/content/numbers.csv')
```

```
# Calculate covariance matrix
cov_matrix = data.cov()

# Print the covariance matrix
print("Covariance Matrix:")
print(cov_matrix)
```

```
Covariance Matrix:
            Column 1    Column 2    Column 3
Column 1  195.133166   -4.548744 -118.040201
Column 2   -4.548744  689.835578    6.716583
Column 3 -118.040201    6.716583  666.854271
```

16. hamming_distance

```
import numpy as np

def hamming_distance(x, y):
    return np.sum(x != y) / len(x)

# Example usage
x = "abcde"
y = "axcye"
hamming_dist = hamming_distance(list(x), list(y))
print("Hamming Distance:", hamming_dist)
```

```
Hamming Distance: 0.2
```

17. jaccard_distance

```
import numpy as np

def jaccard_distance(x, y):
    intersection = len(set(x).intersection(set(y)))
    union = len(set(x).union(set(y)))
    return 1 - intersection / union
```

```
# Example usage
x = [1, 2, 3, 4, 5]
y = [4, 5, 6, 7, 8]
jaccard_dist = jaccard_distance(x, y)
print("Jaccard Distance:", jaccard_dist)
```

```
Jaccard Distance: 0.75
```

18.gower_distance

```
import numpy as np
from scipy.spatial import distance
import pandas as pd


def gower_distance(x, y):
    num_dist = distance.euclidean([x['Column 1']], [y['Column 1']])
    bin_dist = distance.hamming([x['Column 2']], [y['Column 2']])
    ord_dist = distance.cityblock([x['Column 3']], [y['Column 3']])


    dist = weights['Column 1'] * num_dist + weights['Column 2'] * bin_dist + weights['Column 3'] * ord_dist if weights is not None else num_dist + bin_dist + ord_dist


    return dist


# Read data from CSV
data = pd.read_csv('/content/Gowers.csv')


# Sample data from the CSV file
x = {'Column 1': data['Column 1'][0], 'Column 2': data['Column 2'][0], 'Column 3': data['Column 3'][0]}
y = {'Column 1': data['Column 1'][1], 'Column 2': data['Column 2'][1], 'Column 3': data['Column 3'][1]}


# Set weights
weights = {'Column 1': 0.5, 'Column 2': 0.3, 'Column 3': 0.2}
```

```
# Calculate Gower's distance

gower_dist = gower_distance(x, y)


# Print the results

print("Numeric Dissimilarity:", num_dist)

print("Binary Dissimilarity:", bin_dist)

print("Ordinal Dissimilarity:", ord_dist)

print("Gower's Distance:", gower_dist)

print(data.head())
```

```
Numeric Dissimilarity: 2.0

Binary Dissimilarity: 1.0

Ordinal Dissimilarity: 1

Gower's Distance: 1.5
```

19. spearmanr

```
from scipy.stats import spearmanr


# Example usage

x = [1, 2, 3, 4, 5]

y = [5, 4, 3, 2, 1]

spearman_dist, _ = spearmanr(x, y)

spearman_distance = 2 * (1 - spearman_dist)

print("Spearman's Distance:", spearman_distance)
```

```
Spearman's Distance: 2.0
```

Questions:

1. - Create a function that calculates the arithmetic mean of a given list of numbers.

   - Generate a random list of 30 numbers and calculate their arithmetic mean.

2. - Create a function that calculates the weighted mean of a given list of numbers and their corresponding weights.

   - Generate two lists: one with 40 numbers and another with 40 corresponding weights. Calculate their weighted mean.

3. - Create a function that calculates the trimmed mean of a given list of numbers after excluding a specified percentage of outliers from both ends.

- Generate a random list of 100 numbers and calculate their trimmed mean after excluding the top and bottom 10% of outliers.

4. - Create a function that calculates the median of a given list of numbers.

- Generate a random list of 100 numbers and calculate their median.

5. - Create a function that calculates the mode(s) of a given list of numbers.

- Generate a random list of 100 numbers and calculate their mode(s).

6. Implement a linear regression model to predict a target variable based on a set of predictor variables.

- Calculate the correlation coefficient between two variables in a dataset.

7. - Create a function that calculates the standard deviation of a given list of numbers.

- Generate a random list of 100 numbers and calculate their standard deviation.

8. - Create a function that generates a box plot of a given dataset.

- Generate a random dataset and plot its box plot.

9. - Create a function that calculates the z-score of a given list of numbers.

- Generate a random list of 100 numbers and calculate their z-scores.

10. - Create functions that calculate distance matrices for Manhattan, Minkowski, Euclidean, and Supreme distances between a set of points.

- Generate a set of random points and calculate their distance matrices using the different distance metrics.

11. - Create a function that calculates the chi-square statistic and p-value for two categorical variables in a contingency table.

- Generate a contingency table and calculate the chi-square statistic and p-value.

12. - Create a function that calculates the covariance between two variables in a dataset.

- Generate a random dataset and calculate the covariance between two variables.

13. - Write functions that calculate dissimilarity measures (e.g., Jaccard, Dice, Hamming) for nominal, binary, mixed, and ordinal data.

- Generate random datasets of each type and calculate their dissimilarity measures.