**VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES**

**VIVEKANANDA SCHOOL OF INFORMATION TECHNOLOGY**



**BACHELOR OF COMPUTER APPLICATION**

**Introduction to Data Science Lab.**

**BCAP 212**

**Guru Gobind Singh Indraprastha University   
Sector - 16C Dwarka, Delhi – 110078**



**SUBMITTED TO: SUBMITTED BY:**

Karthik Nair

**INDEX**

| S.No. | Practical | Page No. | Signature |
| --- | --- | --- | --- |
| 1. | Suppose you have weather data that tells about the distance and the wind speed. Calculate and generate a new feature, time, from the data. |  |  |
| 2. | Create one, two and three dimensional array using NumPy random function.  Check individual attributes.  i.Print 1D, 2D and 3D array.  ii.NumPy array has its own attribute like datatype, dimension, size, shape find the same for time.  iii.Print first, last and second element of 1D, 2D and 3D array. |  |  |
| 3. | Create an array using arange, print  i.first five elements  ii.element after 5th index  iii.middle subarray  iv.every other element  v.every other element starting at index 1  vi.elements in reversed order |  |  |
| 4. | In question in 2D array, print 2 rows and three columns  i. Create a 2x2 subarray from the original array  ii. Modify subarray (0,0) by 88 and print  iii.Make copy of original array  iv. Print modified and original array |  |  |
| 5. | Create two sample array and  i.combine and print  ii.vertically stack the array and print  iii.horizontally stack the array and print |  |  |
| 6. | Compute the covariance matrix of two given NumPy arrays |  |  |
| 7. | Replace negative value with zero in numpy array |  |  |
| 8. | How to Remove columns in Numpy array that contains non-numeric values? |  |  |
| 9. | How to insert a space between characters of all the elements of a given NumPy  array? |  |  |
| 10. | Create a pandas series from a dictionary of values and an ndarray. |  |  |

| S.No. | Practical | Page No. | Signature |
| --- | --- | --- | --- |
| 11. | Create a Series and Print all the elements that are above 75th percentile. |  |  |
| 12. | Perform sorting on Series data and DataFrames. |  |  |
| 13. | Write a program to implement pivot() and pivot- table() on a DataFrame. |  |  |
| 14. | Write a program to find mean absolute deviation on a DataFrame |  |  |
| 15. | Two Series object, Population stores the details of four metro cities of India and another object AvgIncome stores the total average income reported in four years in these cities. Calculate income per capita for each of these metro cities. |  |  |
| 16. | Create a DataFrame based on E-Commerce data and generate mean, mode, median |  |  |
| 17. | Create a DataFrame based on employee data and generate quartile and variance. |  |  |
| 18. | Program to implement skewness on random data |  |  |
| 19. | Create a DataFame on any Data and compute statistical function of Kurtosis. |  |  |
| 20. | Series objects Temp1, temp3, temp4 stores the temperature of days of week 1, week2, week3, week4, write a script to:-   1. Print average temperature per week. 2. Print average temperature entire month. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

| S.No. | Practical | Page No. | Signature |
| --- | --- | --- | --- |
| 21. | Write a program to read CSV file and create its DataFrame. |  |  |
| 22. | Consider the DataFrame QtrSales where each row contains the term category, item name and expenditure and group the rows by category, and print the average expenditure per category. |  |  |
| 23. | Create a DataFrame having age, name , weight of five students. Write a program to display only the weight of first and fourth rows. |  |  |
| 24. | Write a program to create a Dataframe to store weight, age and name of three people. Print the DataFrame and its transpose. |  |  |
| 25. | Create a data frame in python to produce missing values. The data is printed and missing values are indicated by the value NaN. Handle the missing values by   1. Replacing NaN with 0 2. Dropping the rows with NaN 3. Replacing NaN with the mean/median/mode of the column 4. Replacing NaN with the value of the previous row 5. Replacing NaN with the value of the next row 6. Replacing NaN using interpolation method |  |  |
| 26. | Write a python code for carrying out equal width binning for the price of nine items that are stored in a data frame. For equi- width binning the minimum and maximum price value are used to three equal width bins named low, medium, and high. Plot a histogram for three bins based on the price range. |  |  |
| 27. | Write the python code for outlier detection using the standard deviation method. Here for the randomly generated dataset values, the mean and standard deviation is calculated and then the cut off value is found for identifying outliers by considering thrice the standard deviation value as the threshold value. The outlier can be pictorially represented in the form of a histogram. |  |  |
| 28. | Write the python code to pictorially represent outliers in a histogram. The dataset consists of 94 numerical values containing 2 outliers (the value 10 and 12). The outliers are to be removed from the list and final list of numerical values contain no outliers. |  |  |
| 29. | Write a python code for outlier detection and removal for a given set of data points using an interquartile method. Equi-width bins are created for displaying the data values using a histogram. Q1 and Q3 are calculated using percentile() function used in python, which help in calculating interquartile range by finding differences between Q1 and Q3. Next the lower bound(LB) and upper bound (UB) values are found using the formula = Q1 − �� (1.5 IQR) = Q3 + (1.5 IQR). This helps in forming the final dataset ∗ ∗ contains no outliers by considering data values which are only within the LB and UB. |  |  |
| 30. | Write python code to demonstrates how to perform correlation analysis on a dataset to identify redundant data. • Load the dataset using Pandas read\_csv function. • calculate the correlation matrix using the corr method, iterate over the correlation matrix and compare each correlation value with the threshold (correlation\_threshold). If the absolute correlation value exceeds the threshold, we add the corresponding feature names to the set of highly correlated features. • Finally, print the highly correlated features to identify the redundant data. You can adjust the correlation threshold according to your requirements. • Preprocess data appropriately (e.g., handle missing values, scale numerical features) before performing correlation analysis for accurate results. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 31. | Write a python code for data transformations for four transformation techniques rescaling data, normalizing data, binarizing data and standardizing data. • Initially a data frame is created having three columns as C1, C2 and C3 and four rows or records of numeric values. The original dataset is printed and then various data transformations are applied one by one to view the transformed dataset values. • MinMaxScaler() function is used in case of rescaling data to scale and translate each feature to a given range. • Normalization of data is performed using normalize() preprocessing function that uses the L1 normalization technique. • Binarizer() is used for data transformation for the discretization of continuous feature value. The threshold value considered in the program for binarizing is 5, thus all data above 5 are marked 1 whereas all data qual or below 5 are marked as 0. • The scale() function is used for standardizing data along an axis. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 32. | Write the python code for data transformation using label encoding and one-hot encoding technique. For two techniques, two different datasets have been used and accordingly for each categorial attribute, the values are encoded with numerical values. |  |  |
| 33. | Write the python code used to select features using univariate feature selection method. Consider dataset Pima Indians Diabetes. The dataset has eight features and a class label. The program selects four best features from the diabetes dataset based on the score calculated from each feature |  |  |
| 34. | Write python code of step forward selection for an attribute or feature selection using the step forward selection method. #python code for step forward selection for attribute selection • load the Wine Quality dataset using the load\_wine function from scikit- 5learn and create separate X (features) and y (target variable) data frames. • Then create a linear regression model using LinearRegression from scikit-learn. You can replace this with a different model if needed. • Next, initialize the SequentialFeatureSelector class from scikit-learn, specifying the estimator (our linear regression model) and the direction as ‘forward’. This means that the feature selection process will start with an empty set of features and iteratively add the best feature at each step. • Now fit the selector on the dataset using the fit\_transform method, which performs the step forward feature selection and returns the transformed data with the selected features. • Finally, we obtain the indices of the selected features using the get\_support method and use these indices to get the names of the selected features from the column names of the original dataset. Then print the selected feature names. |  |  |
| 35 | Write python code for Principal Component analysis (PCA) for feature extraction. Consider Breast cancer dataset and after applying PCA only two columns are extracted and displayed in the output. The original dataset consists of 569 records and 30 features. The new dataset formed after feature extraction consists of 569 records and 2 features. • load the Breast Cancer dataset using the load\_breast\_cancer function from scikit-learn and create separate X (features) and y (target variable) data frames. • then initialize the PCA class from scikit-learn, specifying the number of components we want to retain (in this example, n\_components=2). • fit the PCA model on the dataset using the fit\_transform method, which performs the dimensionality reduction and returns the transformed data with the specified number of components. • Next, create a new DataFrame (pca\_df) to store the PCA results. We assign the transformed data to the columns ‘PC1’ and ‘PC2’ (or any other appropriate names). We also include the target variable in the DataFrame for visualization purposes. • print the explained variance ratio using the explained\_variance\_ratio\_ attribute of the PCA object. This shows the amount of variance explained by each principal component. • Finally, print the new DataFrame with the PCA results to inspect the transformed data. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 36. | Write code to perform operations on a data cube in Python, you can use the pandas library for data manipulation and analysis. Create a sample data cube using a dictionary. The data cube contains columns for “Year”, “Quarter”, “Sales”, and “Expenses”. Create a DataFrame using pd.DataFrame and pass the data dictionary. Demonstrates three common operations (aggregation, slicing and dicing) on a data cube. For Aggregation: use the groupby function to group the data by “Year” and “Quarter” columns, and then apply the sum function to calculate the total sales and expenses for each group. For Slicing: use boolean indexing to filter the data for a specific year (2023) and quarter (1). For Dicing: use boolean indexing to filter the data based on a sales range (> 200) and an expenses range (< 120). Finally, print the original data cube, the aggregated data cube, the sliced data cube, and the diced data cube to observe the results of each operation. |  |  |
| 37. | Write code to perform operations on a data cube in Python, you can use the pandas library for data manipulation and analysis. Create a sample data cube using a dictionary. The data cube contains columns for “Year”, “Quarter”, “Sales”, and “Expenses”. Create a DataFrame using pd.DataFrame and pass the data dictionary. Demonstrates three common operations (aggregation, slicing and dicing) on a data cube. For Aggregation: use the groupby function to group the data by “Year” and “Quarter” columns, and then apply the sum function to calculate the total sales and expenses for each group. For Slicing: use boolean indexing to filter the data for a specific year (2023) and quarter (1). For Dicing: use boolean indexing to filter the data based on a sales range (> 200) and an expenses range (< 120). Finally, print the original data cube, the aggregated data cube, the sliced data cube, and the diced data cube to observe the results of each operation. |  |  |

**1. Suppose you have weather data that tells about the distance and the wind speed.**

**Now you are supposed to calculate and generate a new feature from the data, the**

**time.**

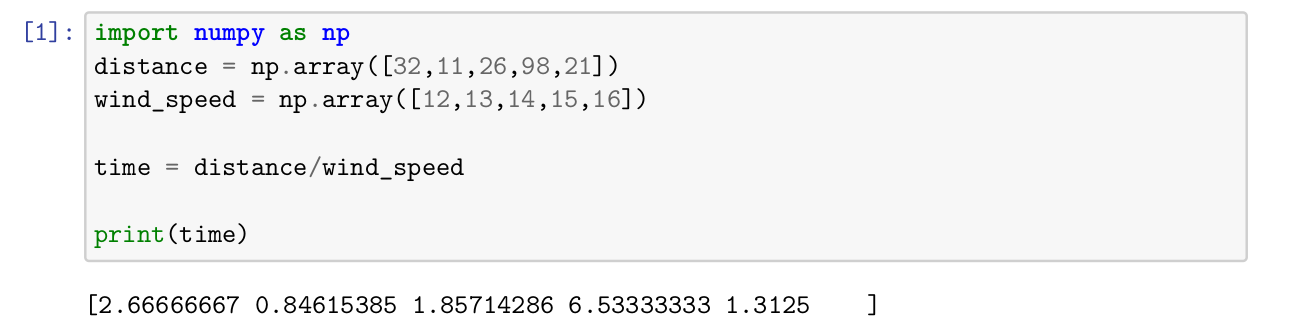
import numpy as np

distance = np.array([32,11,26,98,21])

wind\_speed = np.array([12,13,14,15,16])

time = distance/wind\_speed

print(time)



**2. Create one, two and three dimensional array using NumPy random function check**

**individual attributes.**

**i.Print 1D, 2D and 3D array.**

**ii.NumPy array has its own attribute like datatype, dimension, size, shape find the**

**same for time**

**iii.Print first, last and second element of 1D, 2D and 3D array.**

array0=np.random.randint(100,size=30)

array1=np.random.randint(100,size=(3,15))

array2=np.random.randint(100,size=(2,3,5))

# i

print(array0, array1, array2, sep='\n\n')

# ii

print(f"\ntime \n datatype: {time.dtype}, dimension: {time.ndim}, size: {time.size}, shape: {time.shape}")

print(f"\narray0 \n datatype: {array0.dtype}, dimension: {array0.ndim}, size:{array0.size}, shape: {array0.shape}")

print(f"array1 \n datatype: {array1.dtype}, dimension: {array1.ndim}, size:{array1.size}, shape: {array1.shape}")

print(f"array2 \n datatype: {array2.dtype}, dimension: {array2.ndim}, size:{array2.size}, shape: {array2.shape}")

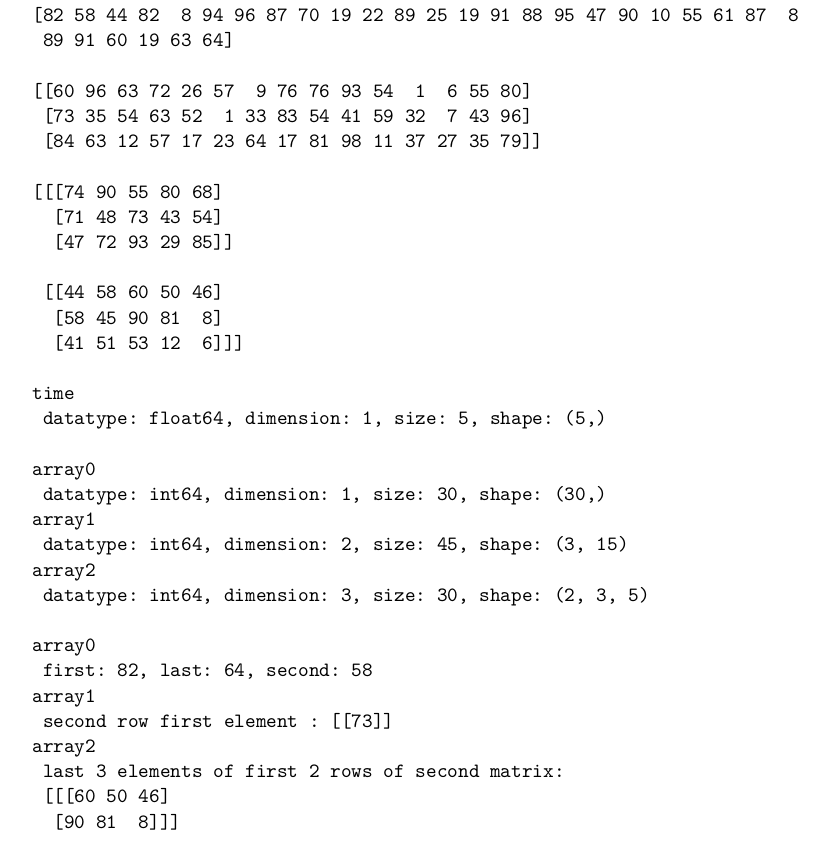
# iii

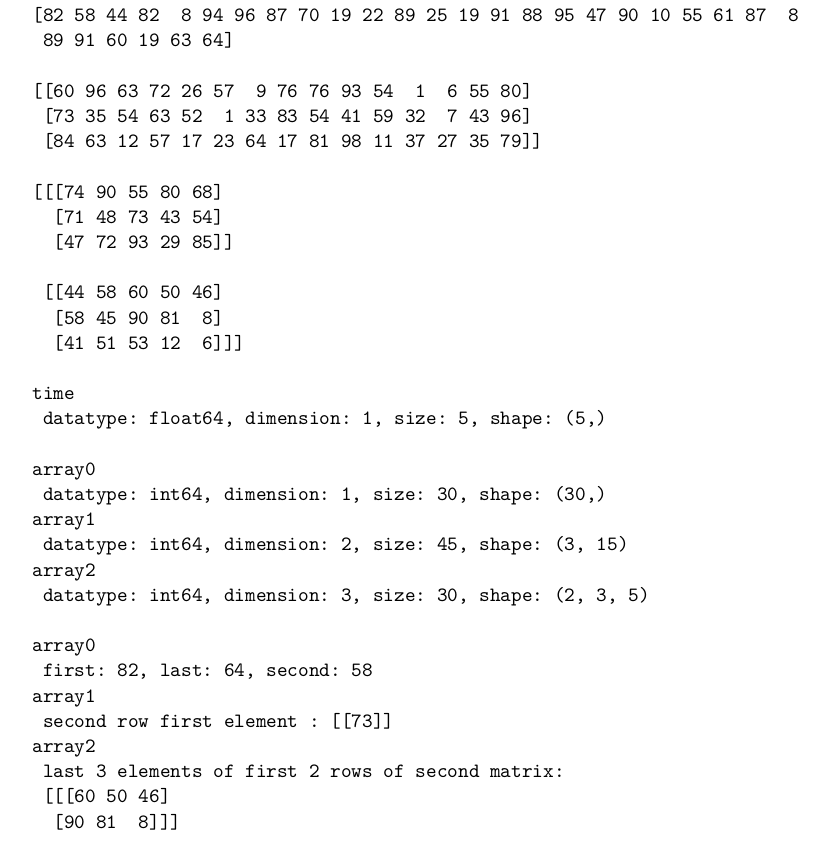
print(f"\narray0 \n first: {array0[0]}, last: {array0[-1]}, second:{array0[1]}")

print(f"array1 \n second row first element: {array1[1:2,0:1]}")

print(f"array2 \n last 3 elements of first 2 rows of second matrix:\n {array2[1:

2, 0:2, -3:]}")



****

**3. Create an array using arange, print**

**i.first five elements**

**ii.element after 5th index**

**iii.middle subarray**

**iv.every other element**

**v.every other element starting at index 1**

**vi.elements in reversed order**

array3=np.arange(0,30)

print(f"\n first 5 elements: {array3[:5]}")

print(f"\n element after 5th index: {array3[6]}")

n = 4

print(f"\n middle subarray with {n} elements: ")

if n%2==0:

print(f"{np.array\_split(array3,2)[0][-1:-((n//2)+1):-1][::-1]} and {np.array\_split(array3,2)[1][0:(n//2)]}")

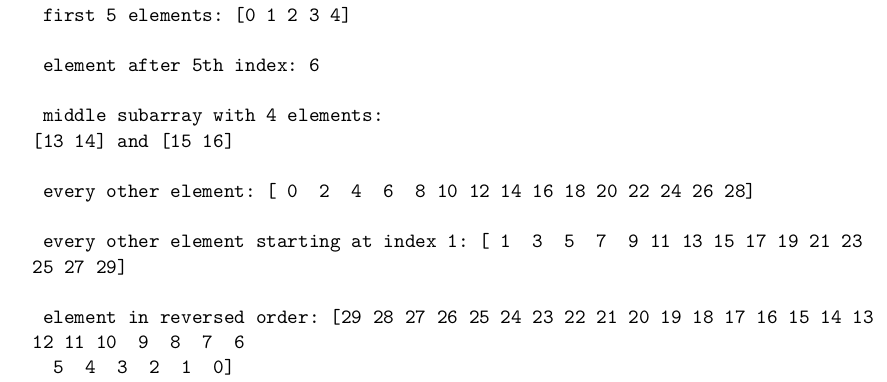
else:

print(f"{np.array\_split(array3,2)[0][-1:-((n//2)+2):-1][::-1]} and {np.array\_split(array3,2)[1][0:(n//2)]}")

print(f"\n every other element: {array3[::2]}")

print(f"\n every other element starting at index 1: {array3[1::2]}")

print(f"\n element in reversed order: {array3[::-1]}")

****

**4. In question in 2D array, print 2 rows and three columns**

**i. Create a 2x2 subarray from the original array**

**ii. Modify subarray (0,0) by 88 and print**

**iii.Make copy of original array**

**iv. Print modified and original array**

print(f"First two rows and three columns of \n\n{array1} is \n\n{array1[:2,:3]}")

array4=np.random.randint(100,size=(3,3))

print(f"\n original array: \n {array4}")

array6 = array4.copy()

array5 = array4[0:2,0:2]

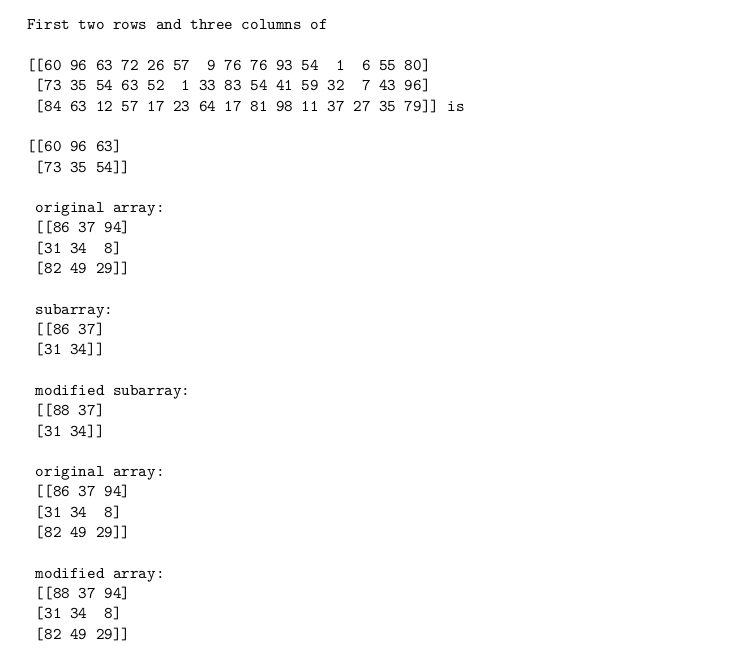
print(f"\n subarray: \n {array5}")

3array5[0,0]=88

print(f"\n modified subarray: \n {array5}")

print(f"\n original array: \n {array6}")

print(f"\n modified array: \n {array4}")

****

**5. Create two sample array and**

**i.combine and print**

**ii.vertically stack the array and print**

**iii.horizontally stack the array and print**

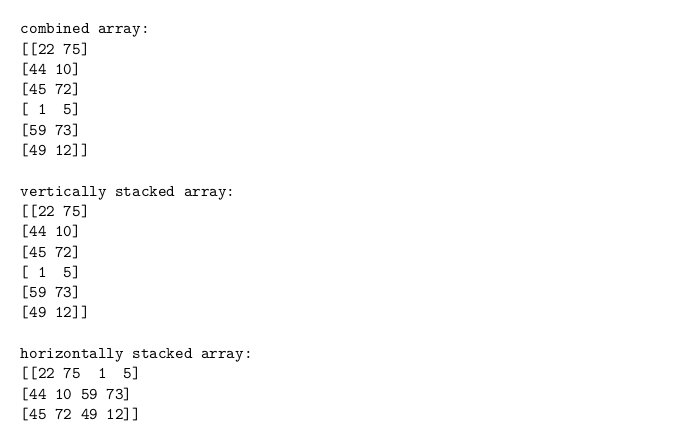
array7 = np.random.randint(100,size=(3,2))

array8 = np.random.randint(100,size=(3,2))

print(f"\n combined array: \n {np.concatenate((array7,array8))}")

print(f"\n vertically stacked array: \n {np.vstack((array7,array8))}")

print(f"\n horizontally stacked array: \n {np.hstack((array7,array8))}")

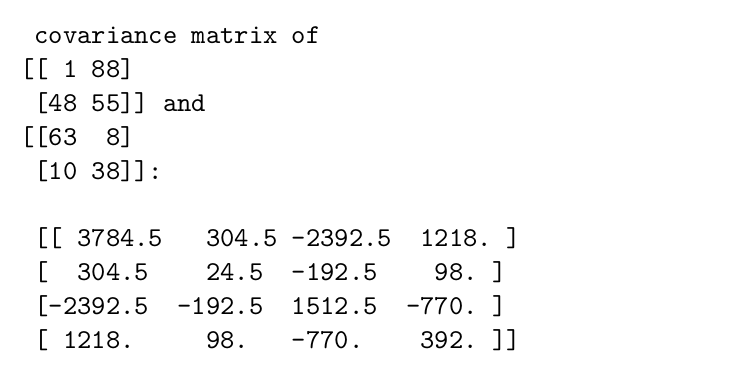


**6. Compute the covariance matrix of two given NumPy arrays**

array9 = np.random.randint(100,size=(2,2))

array10 = np.random.randint(100,size=(2,2))

print(f"\n covariance matrix of \n{array9} and \n{array10}: \n\n {np.cov(array9,array10)}")



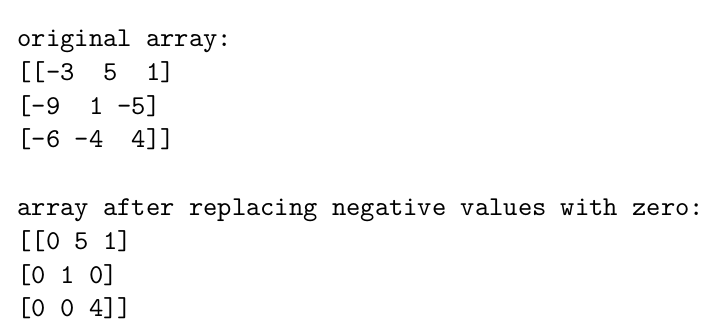
**7. Replace negative value with zero in numpy array**

array11 = np.random.randint(-10,10,size=(3,3))

print(f"\n original array: \n {array11}")

array11[array11<0]=0

print(f"\n array after replacing negative values with zero: \n {array11}")



**8. How to Remove columns in Numpy array that contains non-numeric values?**

import numpy as np

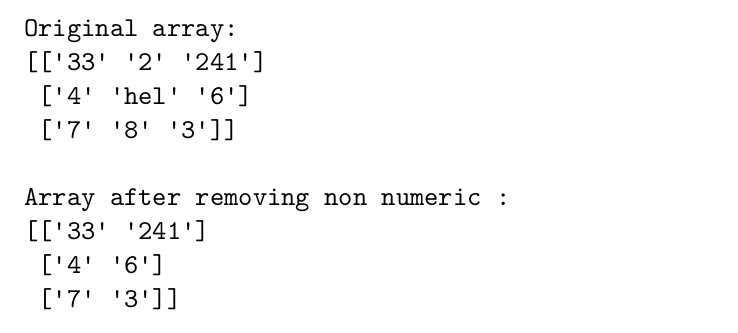
array12 = np.array([["33",2,241],[4,"hel",6],[7,8,3]])

print(f"Original array: \n{array12}\n")

numeric\_columns = np.array([np.char.isnumeric(col) for col in array12.T]).all(axis=0)

array12 = array12[:, numeric\_columns]

print(f"Array after removing non numeric : \n{array12}\n")



**9. How to insert a space between characters of all the elements of a given NumPy**

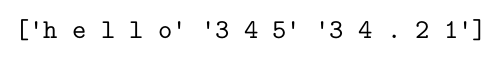
**array?**

array13 = np.array(["hello", 345, 34.21])

# Insert space between characters

array13 = np.char.join(" ", array13)

print(array13)



**10. Create a pandas series from a dictionary of values and an ndarray.**

import pandas as pd

import numpy as np

Andoid\_versions = {"Android 4.4": 19, "Android 5.0": 21, "Android 5.1": 22,

"Android 6.0": 23, "Android 7.0": 24, "Android 7.1": 25, "Android 8.0": 26,

"Android 8.1": 27, "Android 9.0": 28, "Android 10.0": 29, "Android 11.0":

30, "Android 12.0": 31, "Android 12.1": 32}

s0 = pd.Series(Andoid\_versions)

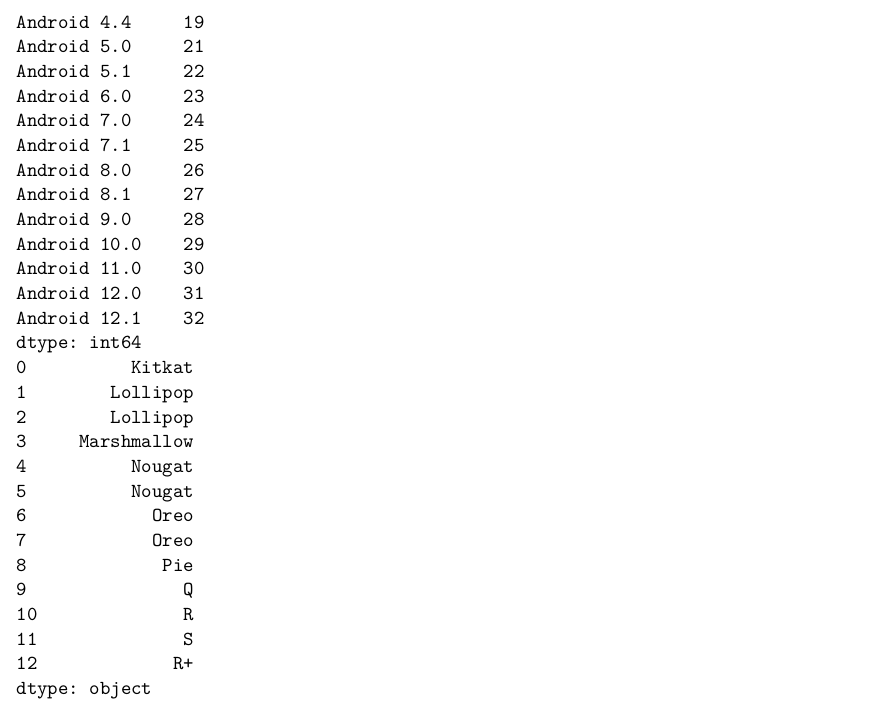
print(s0)

Android\_codenames = np.array(["Kitkat", "Lollipop", "Lollipop", "Marshmallow",

"Nougat", "Nougat", "Oreo", "Oreo", "Pie", "Q", "R", "S", "R+"])

s1 = pd.Series(Android\_codenames)

print(s1)

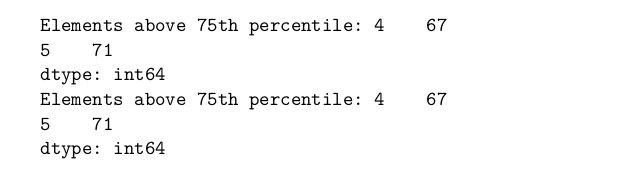


**11. Create a Series and print all the elements that are above 75th percentile.**

s2 = pd.Series([13, 15, 16, 33, 67, 71, 55])

print(f"Elements above 75th percentile: {s2[s2 > s2.quantile(.75)]}")

print(f"Elements above 75th percentile: {s2[s2 > np.percentile(s2, 75)]}")



**12. Perform sorting on Series data and DataFrames**

s3 = pd.Series({"x": 24, "a": 1, "d": 4, "w": 23, "t": 20})

print(s3.sort\_values())

df0 = pd.DataFrame({"x": [24, 1, 4, 23, 20], "y": [1, 2, 3, 4, 5], "z": [6, 7,

8, 9, 10]})

print(df0.sort\_values(by="x"))



**13. Write a program to implement pivot() and pivot-table() on a DataFrame**

df1 = pd.DataFrame({

'codename': ["kitkat", "Lollipop", "Marshmallow", "Nougat"],

'version': [4.4, 5.0, 6.0, 7.0],

'year': [2013, 2014, 2015, 2016]

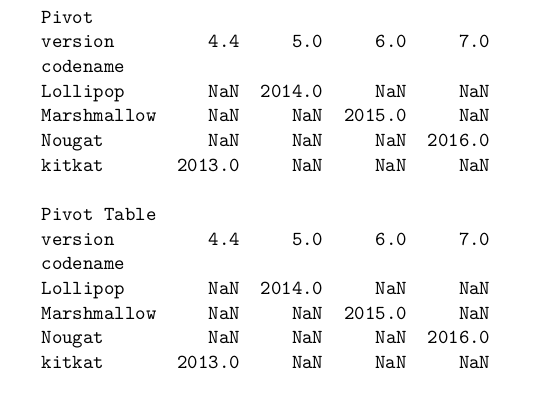
})

pivot0 = df1.pivot(index="codename", columns="version", values="year")

print(f"Pivot\n{pivot0}\n")

pivot1 = df1.pivot\_table(index="codename", columns="version", values="year")

print(f"Pivot Table\n{pivot1}\n")



**14. Write a program to find mean absolute deviation on a DataFrame**

df2 = pd.DataFrame({

'x': [1, 2, 3, 4, 5],

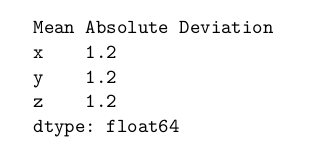
'y': [6, 7, 8, 9, 10],

'z': [11, 12, 13, 14, 15]

})

# mad using abs(df - df.mean()).mean()

print(f"Mean Absolute Deviation\n{abs(df2 - df2.mean()).mean()}\n")



**15. Two Series object, Population stores the details of four metro cities of India and**

**another object AvgIncome stores the total average income reported in four years in**

**these cities. Calculate income per capita for each of these metro cities.**

Population = pd.Series({"Delhi": 29941000, "Mumbai": 23355000, "Chennai":

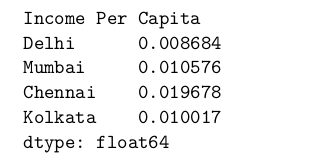
11180000, "Kolkata": 14974000})

AvgIncome = pd.Series({"Delhi": 260000, "Mumbai": 247000, "Chennai": 220000,

"Kolkata": 150000})

IncomePerCapita = AvgIncome / Population

print(f"Income Per Capita\n{IncomePerCapita}\n")

****

**16. Create a DataFrame based on E-Commerce data and generate mean,mode, median**

ecom\_df = pd.DataFrame({

'order\_id': [22311, 22312, 22313, 22314, 22315],

'customer\_id': [1001, 1002, 1003, 1004, 1005],

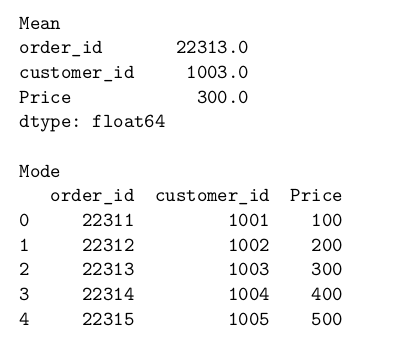
'Price': [100, 200, 300, 400, 500],

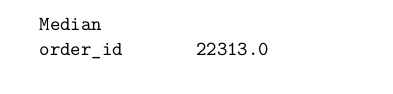
})

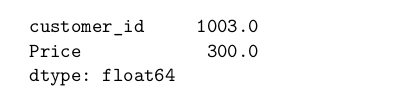
print(f"Mean\n{ecom\_df.mean()}\n")

print(f"Mode\n{ecom\_df.mode()}\n")

print(f"Median\n{ecom\_df.median()}\n")







**17. Create a DataFrame based on employee data and generate quartile and variance**

employee\_df = pd.DataFrame({

'employee\_id': [1001, 1002, 1003, 1004, 1005],

'employee\_name': ["Arjun", "Bhuvan", "Chetan", "Dinesh", "Eshwar"],

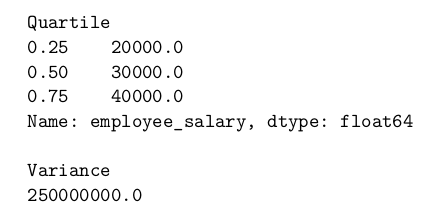
'employee\_salary': [10000, 20000, 30000, 40000, 50000],

})

print(f"Quartile\n{employee\_df['employee\_salary'].quantile([0.25, 0.5, 0.

75])}\n")

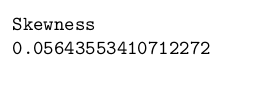
print(f"Variance\n{employee\_df['employee\_salary'].var()}\n")



**18. Program to implement Skewness on Random data**

random\_data = pd.Series(np.random.normal(0, 1, 1000))

print(f"Skewness\n{random\_data.skew()}\n")



**19. Create a DateFrame on any Data and compute statistical function ofKurtosis**

df3 = pd.DataFrame({

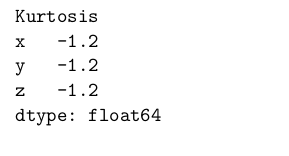
'x': [1, 2, 3, 4, 5],

'y': [6, 7, 8, 9, 10],

'z': [11, 12, 13, 14, 15]

})

print(f"Kurtosis\n{df3.kurtosis()}\n")



**20. Series objects Temp1, temp2, temp3, temp 4 stores the temperature of days of**

**week 1, week 2, week 3, week 4. Write a script to:-**

**a.Print average temperature per week**

**b.Print average temperature of entire month**

temp1 = pd.Series([30.0, 34.2, 31.7, 30.9, 32.5, 33.4, 32.2])

temp2 = pd.Series([32.3, 33.4, 31.2, 32.4, 33.2, 32.4, 31.9])

temp3 = pd.Series([31.2, 32.4, 33.2, 32.4, 31.9, 30.9, 32.5])

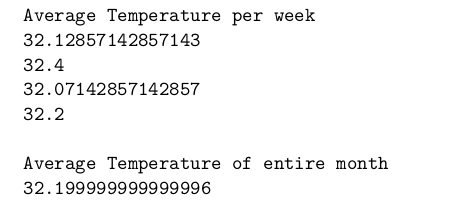
temp4 = pd.Series([33.4, 31.2, 32.4, 33.2, 32.4, 31.9, 30.9])

print(f"Average Temperature per week\n{temp1.mean()}\n{temp2.mean()}\n{temp3.

mean()}\n{temp4.mean()}\n")

print(f"Average Temperature of entire month\n{pd.concat([temp1, temp2, temp3,

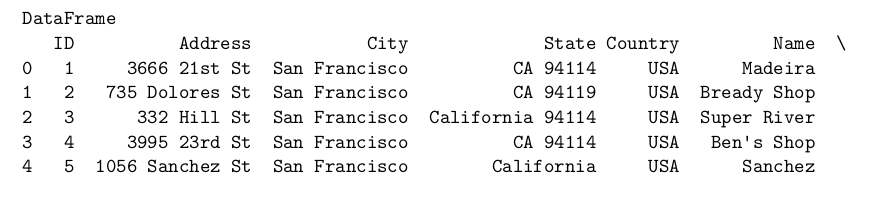
temp4]).mean()}\n")

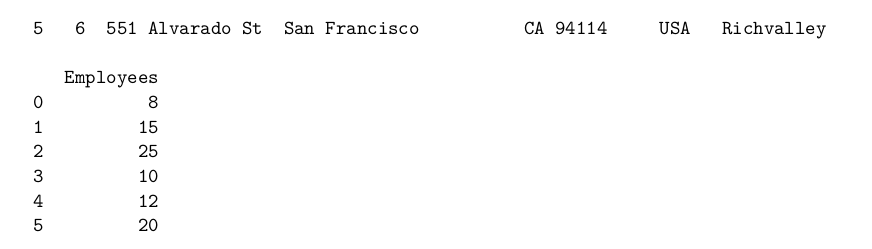


**21. Write a Program to read a CSV file and create its DataFrame.**

df4 = pd.read\_csv("supermarkets.csv")

print(f"DataFrame\n{df4}\n")





**22. Consider the DataFrame QtrSales where each row contains the item category,**

**item name and expenditure and group the rows by category, and print the average**

**expenditure per category.22. Consider the DataFrame QtrSales where each row contains the item category,**

**item name and expenditure and group the rows by category, and print the average**

**expenditure per category.**

QtrSales = pd.DataFrame({

'Category': ["Stationary", "Grocery", "Dairy", "Stationary", "Grocery",

"Dairy", "Stationary", "Grocery", "Dairy"],

'Item': ["Pen", "Bread", "Milk", "Pencil", "Butter", "Cheese", "Eraser",

"Rice", "Curd"],

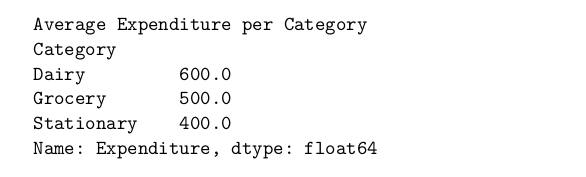
'Expenditure': [100, 200, 300, 400, 500, 600, 700, 800, 900]

})

# group the rows by category and print average expenditure per category

print(f"Average Expenditure per Category\n{QtrSales.

groupby('Category')['Expenditure'].mean()}\n")



**23. Create a DataFrame having age, name, weight of five students. Write a program**

**to display only the weight of first and fourth rows.**

df5 = pd.DataFrame({

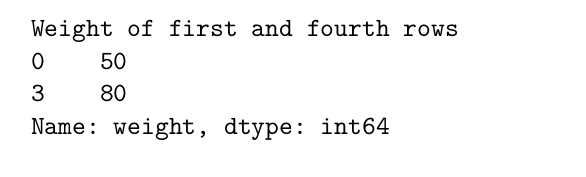
'age': [20, 21, 22, 23, 24],

'name': ["Arjun", "Bhuvan", "Chetan", "Dinesh", "Eshwar"],

'weight': [50, 60, 70, 80, 90]

})

print(f"Weight of first and fourth rows\n{df5.iloc[[0, 3]]['weight']}\n")



**24. Write a program to create a DataFrame to store weight, age and name of three**

**people. Print the DataFrame and its transpose.**

df6 = pd.DataFrame({

'age': [20, 21, 22],

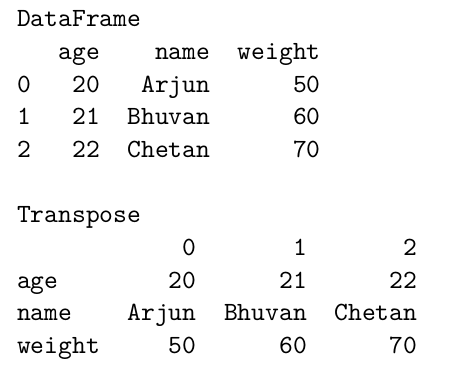
'name': ["Arjun", "Bhuvan", "Chetan"],

'weight': [50, 60, 70]

})

print(f"DataFrame\n{df6}\n")

print(f"Transpose\n{df6.T}\n")



**25. Create a data frame in Python to produce missing values. The data is printed**

**and missing values are indicated by the value NaN. Handle the missing values by**

**a. Replacing NaN with 0**

**b. dropping the rows with NaN**

**c. replacing NaN with the mean/median/mode of the column**

**d. replacing NaN with the value of the previous row**

**e. replacing NaN with the value of the next row**

**f. replacing NaN using interpolation method**

import numpy as np

import pandas as pd

df = pd.DataFrame(np.random.randn(5, 3),

index=['a', 'c', 'e', 'f', 'h'],

columns=['one', 'two', 'three'])

df0 = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

df1 = df0.copy()

df2 = df0.copy()

df3 = df0.copy()

df4 = df0.copy()

df5 = df0.copy()

df6 = df0.copy()

df7 = df0.copy()

print(df)

# Replacing missing values with zeros

print(f"Dataframe after replacing missing values with zeros:\n{df0}\n")

print(df0.fillna(0))

# Dropping rows with missing values

print(f"Dataframe after dropping rows with missing values:\n{df1}\n")

print(df1.dropna())

# Replacing missing values with Mean/Median/Mode

print(f"Dataframe after replacing missing values with Mean:\n{df2}\n")

print(df2.fillna(df2.mean()))

print(f"Dataframe after replacing missing values with Median:\n{df3}\n")

print(df3.fillna(df3.median()))

print(f"Dataframe after replacing missing values with Mode:\n{df4}\n")

print(df4.fillna(df4.mode()))

# Filling NaN values with the value from the previous rows

print(f"Dataframe after filling NaN values with the value from the previous

rows:\n{df5}\n")

print(df5.fillna(method='pad'))

# Filling NaN values with the value from the next rows

print(f"Dataframe after filling NaN values with the value from the next rows:

\n{df6}\n")

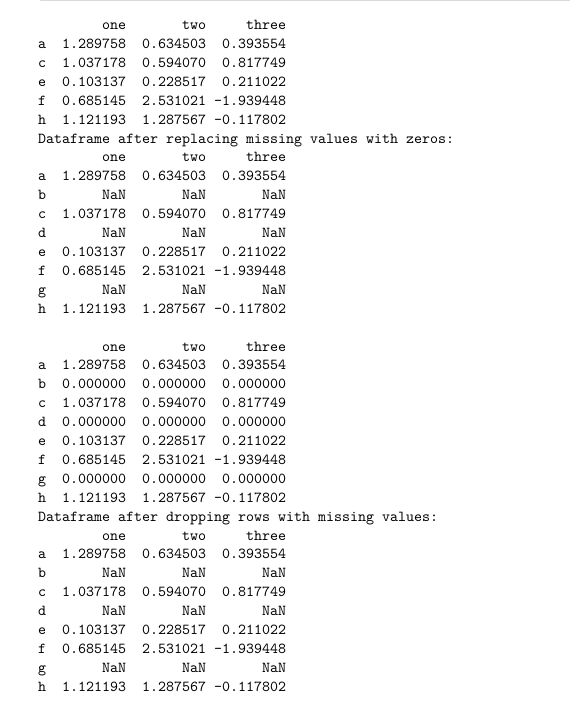
print(df6.fillna(method='bfill'))

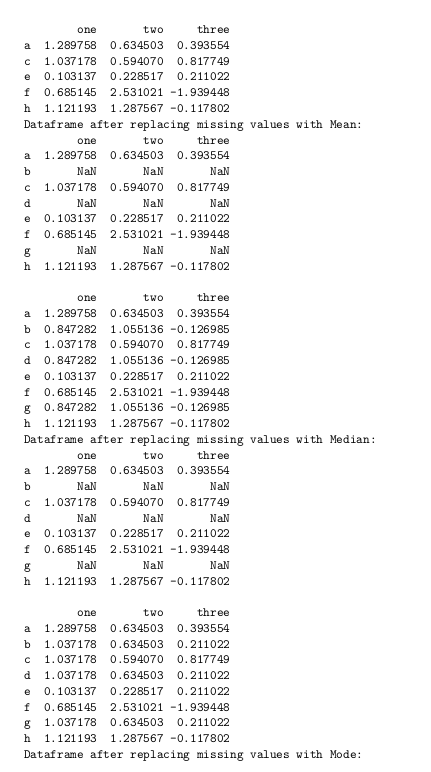
# Filling missing values using interpolation method: Linear

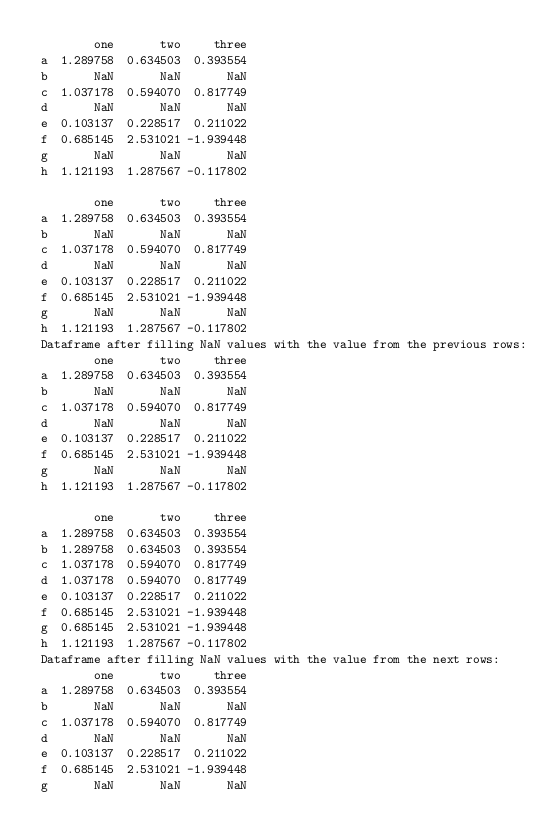
print(f"Dataframe after filling missing values using interpolation method:

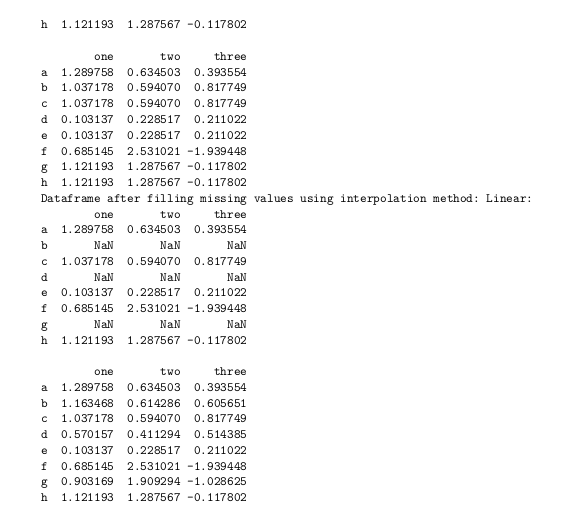
Linear:\n{df7}\n")

print(df7.interpolate(method='linear'))









**26. Write a python code for carrying out equal width binning for the price of**

**nine items that are stored in a data frame. For equi-width binning the**

**minimum and maximum price value are used to three equal width bins named**

**Low, Medium, and High. Plot a histogram for the three bins based on the**

**price range.**

prices = [3.04, 22.93, 3.74, 3.71, 14.25, 3.42, 0.44, 0.44, 0.44]

df8 = pd.DataFrame(prices, columns=['Price'])

print(df8)

# find equi-width bins number from length of the column

bin\_count = int(np.ceil(np.sqrt(len(df8))))

df8['Price'] = pd.cut(df8['Price'], bins=bin\_count, labels=['Low', 'Medium',

'High'])

print(df8)

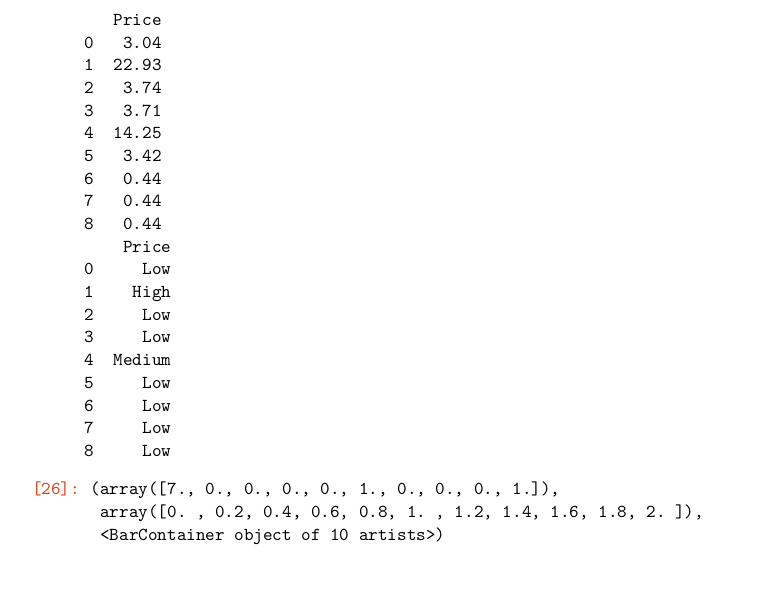
import matplotlib.pyplot as plt

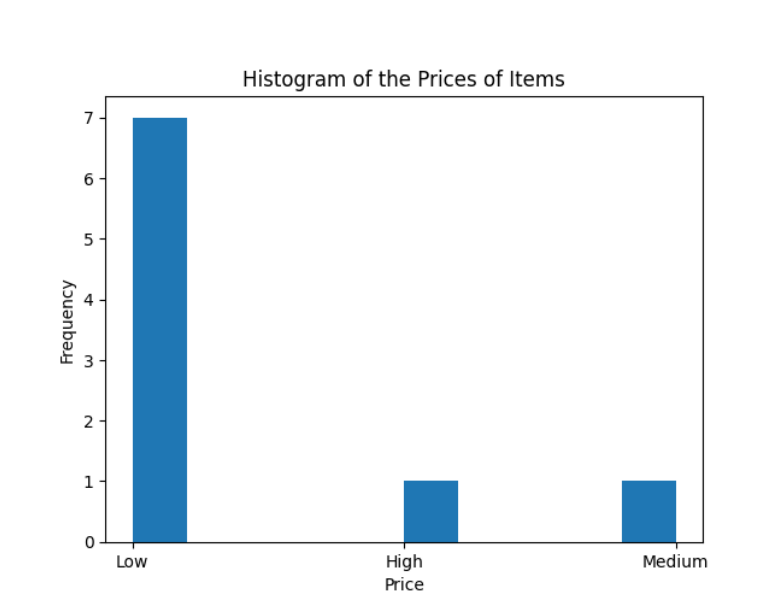
plt.title("Histogram of the Prices of Items")

plt.xlabel('Price')

plt.ylabel('Frequency')

plt.hist(df8['Price'])





**27. Write the python code for outlier detection using the standard deviation**

**method. Here for the randomly generated dataset values, the mean and**

**standard deviation is calculated and then the cut off value is found for**

**identifying outliers by considering thrice the standard deviation value as**

**the threshold value. The outlier can be pictorially represented in form of a**

**histogram**

np.random.seed(21)

data = np.random.normal(loc=0, scale=1, size=1000)

mean = np.mean(data)

std = np.std(data)

threshold = 3 \* std

outliers = []

for value in data:

if abs(value - mean) > threshold:

outliers.append(value)

plt.hist(data, bins='auto', alpha=0.7, rwidth=0.85)

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title('Histogram of Data with Outliers')

plt.axvline(x=np.mean(data), color='red', linestyle='dashed', linewidth=1,

label='Mean')

plt.axvline(x=np.mean(data) + 3 \* np.std(data), color='orange',

linestyle='dashed', linewidth=1, label='Threshold')

plt.axvline(x=outliers[0], color='green', linestyle='dashed', linewidth=1,

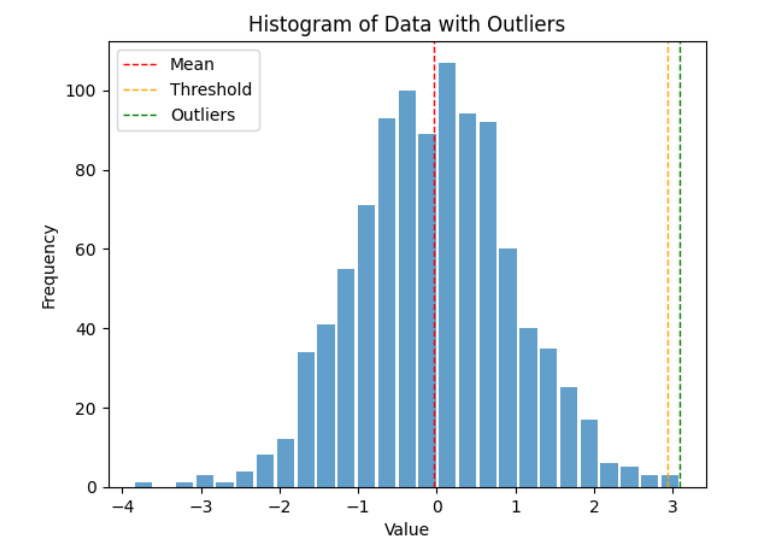
label='Outliers')

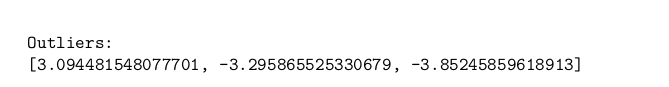
plt.legend()

plt.show()

print("Outliers:")

print(outliers)





**28. Write the python code to pictorially represent outliers in a histogram. The**

**dataset consists of 94 numerical values containing 2 outliers (the value 10**

**and 12). The outliers are to be removed from the list and final list of**

**numerical values contain no outliers.**

import matplotlib.pyplot as plt

dataset = [4, 5, 8, 7, 9, 11, 6, 3, 5, 7, 10, 14, 8, 9, 12, 6, 5, 3, 2, 4, 6,

8, 12, 14, 15, 7, 6, 5, 4, 3, 9, 11, 13, 14, 15, 6, 7, 8, 10, 12, 14, 16,

18, 6, 5, 4, 3, 2, 9, 8, 7, 6, 5, 10, 12, 14, 16, 18, 20, 21, 22, 23, 5, 4,

6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,

44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80,

82, 84, 86, 88, 90, 92, 94]

plt.hist(dataset, bins=10)

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.title('Histogram of Dataset with Outliers')

plt.axvline(x=12, color='red', linestyle='dashed', linewidth=2,

label='Outliers')

plt.axvline(x=10, color='red', linestyle='dashed', linewidth=2,

label='Outliers')

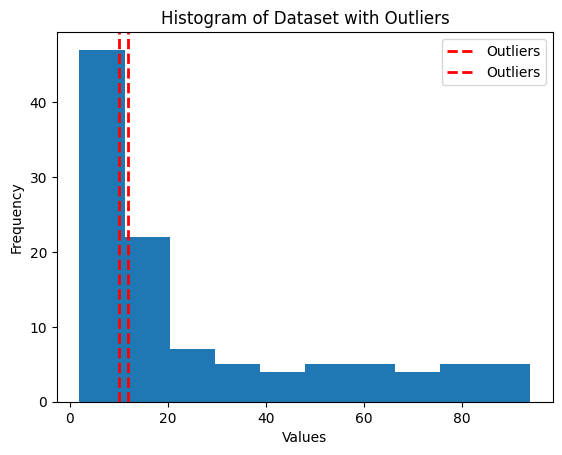
plt.legend()

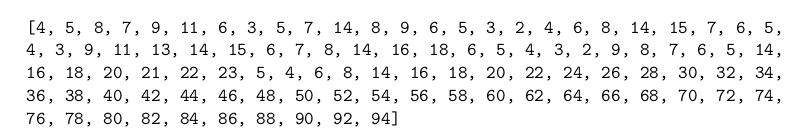
plt.show()

dataset\_without\_outliers = [value for value in dataset if value != 10 and value

!= 12]

print(dataset\_without\_outliers)





**29. Write a python code for outlier detection and removal for a given set of**

**data points using an interquartile method. Equi-width bins are created for**

**displaying the data values using a histogram. Q1 and Q3 are calculated using**

**percentile() function used in python, which help in calculating**

**interquartile range by finding differences between Q1 and Q3. Next the lower**

**bound(LB) and upper bound (UB) values are found using the formula �� = Q1 −**

**(1.5 ∗ IQR) = Q3 + (1.5 ∗ IQR). This helps in forming the final dataset**

**contains no outliers by considering data values which are only within the LB**

**and UB.**

import numpy as np

import matplotlib.pyplot as plt

data\_points = [-210, -4, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 88, 92,

94, 96, 98, 100, 102, 104, 106, 108, 110, 240, 300]

Q1 = np.percentile(data\_points, 25)

Q3 = np.percentile(data\_points, 75)

IQR = Q3 - Q1

LB = Q1 - (1.5 \* IQR)

UB = Q3 + (1.5 \* IQR)

plt.hist(data\_points, bins='auto')

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.title('Histogram of Dataset (With Outliers)')

plt.show()

final\_dataset = [value for value in data\_points if (LB <= value) and (value <=

UB)]

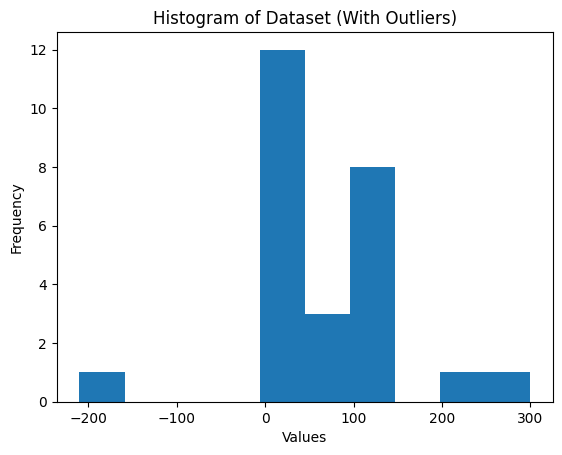
plt.hist(final\_dataset, bins='auto')

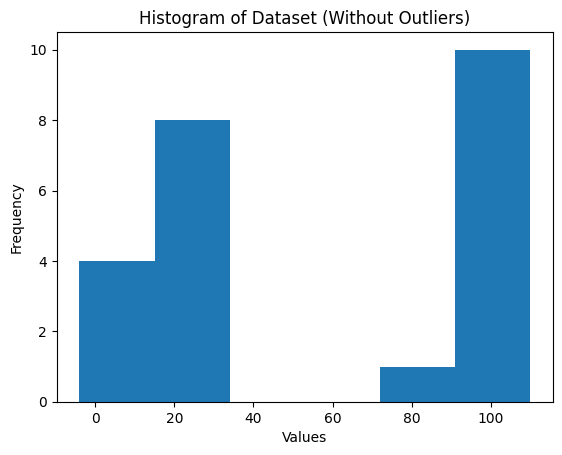
plt.xlabel('Values')

plt.ylabel('Frequency')

plt.title('Histogram of Dataset (Without Outliers)')

plt.show()





**30. Write python code to demonstrates how to perform correlation analysis on a**

**dataset to identify redundant data.**

**• Load the dataset using Pandas read\_csv function.**

**• calculate the correlation matrix using the corr method, iterate over the correlation matrix and**

**compare each correlation value with the threshold (correlation\_threshold). If the absolute**

**correlation value exceeds the threshold, we add the corresponding feature names to the set of**

**highly correlated features.**

**• Finally, print the highly correlated features to identify the redundant data. You can adjust**

**the correlation threshold according to your requirements.**

**• Preprocess data appropriately (e.g., handle missing values, scale numerical features) before**

**performing correlation analysis for accurate results.**

df5 = pd.read\_csv("sample\_dataset.csv")

corr\_matrix = df5.corr().abs()

correlation\_threshold = 0.9

highly\_correlated\_features = set()

# Iterate over the correlation matrix

for i in range(len(corr\_matrix.columns)):

for j in range(i):

# Check if the absolute correlation value exceeds the threshold

if abs(corr\_matrix.iloc[i, j]) > correlation\_threshold:

colname = corr\_matrix.columns[i]

highly\_correlated\_features.add(colname)

print(f"Highly Correlated Features: {highly\_correlated\_features}")



**31. Write a python code for data transformations for four transformation techniques**

**rescaling data, normalizing data, binarizing data and standardizing data.**

**• Initially a data frame is created having three columns as C1, C2 and C3 and four rows or**

**records of numeric values. The original dataset is printed and then various data transforma-**

**tions are applied one by one to view the transformed dataset values.**

**• MinMaxScaler() function is used in case of rescaling data to scale and translate each feature**

**to a given range.**

**• Normalization of data is performed using normalize() preprocessing function that uses the L1**

**normalization technique.**

**• Binarizer() is used for data transformation for the discretization of continuous feature value.**

**The threshold value considered in the program for binarizing is 5, thus all data above 5 are**

**marked 1 whereas all data qual or below 5 are marked as 0.**

**• The scale() function is used for standardizing data along an axis.**

from sklearn.preprocessing import MinMaxScaler, normalize, Binarizer, scale

data = pd.DataFrame({

'C1': [3, 7, 2, 5],

'C2': [10, 2, 8, 6],

'C3': [4, 9, 1, 7]

})

print("Original Dataset:")

print(data)

rescaler = MinMaxScaler(feature\_range=(0, 1))

rescaled\_data = rescaler.fit\_transform(data)

rescaled\_df = pd.DataFrame(rescaled\_data, columns=data.columns)

print("\nRescaled Dataset:")

print(rescaled\_df)

normalized\_data = normalize(data, norm='l1')

normalized\_df = pd.DataFrame(normalized\_data, columns=data.columns)

print("\nNormalized Dataset:")

print(normalized\_df)

binarizer = Binarizer(threshold=5)

binarized\_data = binarizer.transform(data)

binarized\_df = pd.DataFrame(binarized\_data, columns=data.columns)

print("\nBinarized Dataset:")

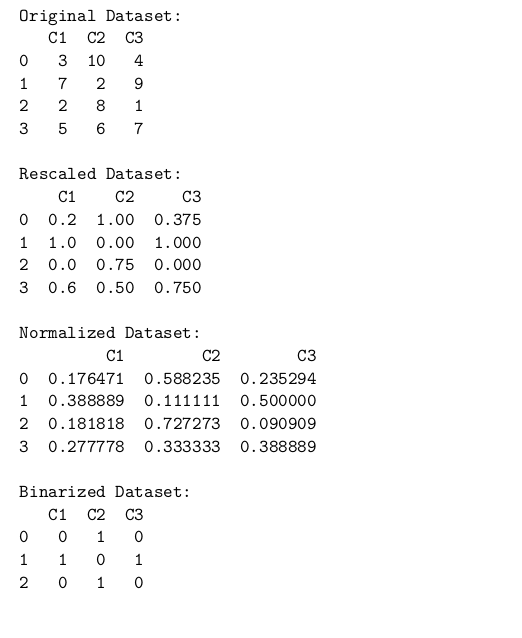
print(binarized\_df)

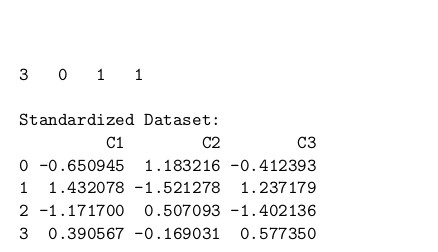
standardized\_data = scale(data)

standardized\_df = pd.DataFrame(standardized\_data, columns=data.columns)

print("\nStandardized Dataset:")

print(standardized\_df)





**32 Write the python code for data transformation using label encoding and one-hot**

**encoding technique. For two techniques, two different datasets have been used and**

**accordingly for each categorial attribute, the values are encoded with numerical values.**

import pandas as pd

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

data1 = pd.DataFrame({

'Color': ['Red', 'Blue', 'Green', 'Red', 'Yellow'],

'Size': ['Small', 'Large', 'Medium', 'Medium', 'Small'],

'Shape': ['Circle', 'Square', 'Triangle', 'Circle', 'Square']

})

print("Dataset 1:")

print(data1)

label\_encoder = LabelEncoder()

data1\_label\_encoded = data1.copy()

for column in data1\_label\_encoded.columns:

data1\_label\_encoded[column] = label\_encoder.

fit\_transform(data1\_label\_encoded[column])

print("\nLabel Encoded Dataset 1:")

print(data1\_label\_encoded)

one\_hot\_encoder = OneHotEncoder(sparse=False, drop='first')

data1\_one\_hot\_encoded = pd.DataFrame(one\_hot\_encoder.fit\_transform(data1))

data1\_one\_hot\_encoded.columns = one\_hot\_encoder.get\_feature\_names\_out(data1.

columns)

print("\nOne-Hot Encoded Dataset 1:")

print(data1\_one\_hot\_encoded)

data2 = pd.DataFrame({

'City': ['New York', 'London', 'Paris', 'London', 'Paris'],

'Temperature': ['Hot', 'Cold', 'Warm', 'Warm', 'Hot'],

'Rain': ['Yes', 'No', 'No', 'Yes', 'Yes']

})

print("\nDataset 2:")

print(data2)

data2\_label\_encoded = data2.copy()

for column in data2\_label\_encoded.columns:

data2\_label\_encoded[column] = label\_encoder.

fit\_transform(data2\_label\_encoded[column])

print("\nLabel Encoded Dataset 2:")

print(data2\_label\_encoded)

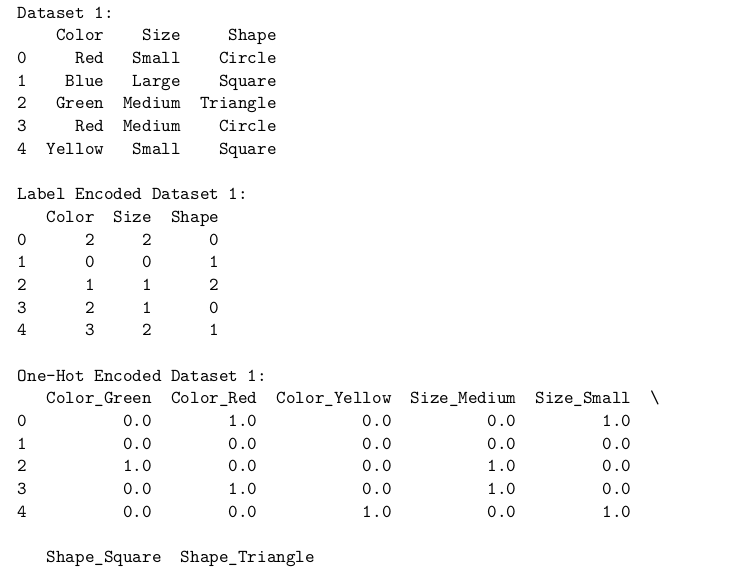
data2\_one\_hot\_encoded = pd.DataFrame(one\_hot\_encoder.fit\_transform(data2))

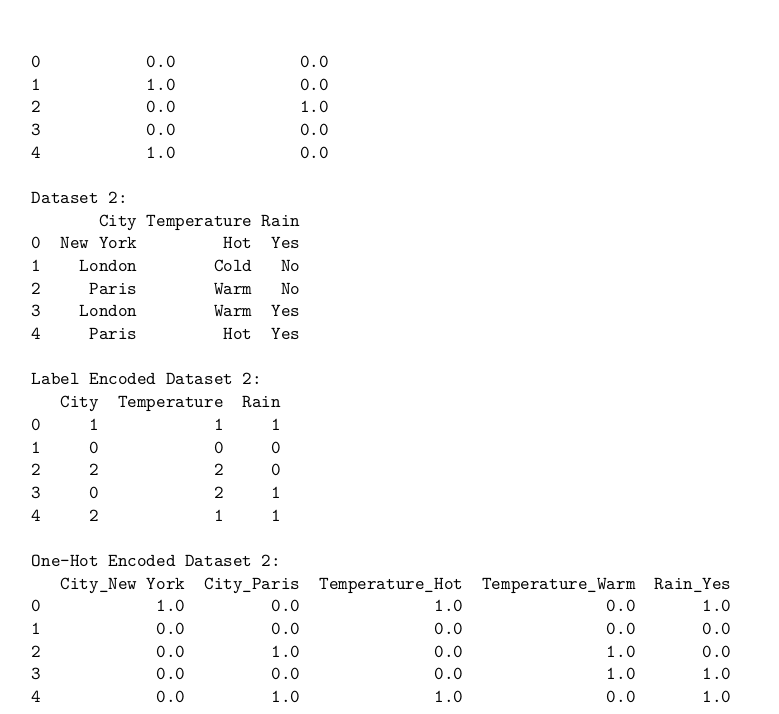
data2\_one\_hot\_encoded.columns = one\_hot\_encoder.get\_feature\_names\_out(data2.

columns)

print("\nOne-Hot Encoded Dataset 2:")

print(data2\_one\_hot\_encoded)





**33. Write the python code used to select features using univariate feature selection**

**method. Consider dataset Pima Indians Diabetes. The dataset has eight features and**

**a class label. The program selects four best features from the diabetes dataset based**

**on the score calculated from each feature**

from sklearn.feature\_selection import SelectKBest, f\_classif

data = pd.read\_csv("diabetes.csv")

X = data.drop("Outcome", axis=1)

y = data["Outcome"]

k = 4

selector = SelectKBest(score\_func=f\_classif, k=k)

X\_new = selector.fit\_transform(X, y)

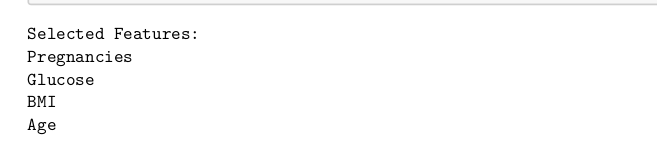
selected\_feature\_indices = selector.get\_support(indices=True)

selected\_feature\_names = X.columns[selected\_feature\_indices]

print("Selected Features:")

for feature in selected\_feature\_names:

print(feature)



**34. Write python code of step forward selection for an attribute or feature selection**

**using the step forward selection method. #python code for step forward selection for**

**attribute selection**

**• load the Wine Quality dataset using the load\_wine function from scikit- 5learn and create**

**separate X (features) and y (target variable) data frames.**

**• Then create a linear regression model using LinearRegression from scikit-learn. You can**

**replace this with a different model if needed.**

**• Next, initialize the SequentialFeatureSelector class from scikit-learn, specifying the estimator**

**(our linear regression model) and the direction as ‘forward’. This means that the feature**

**selection process will start with an empty set of features and iteratively add the best feature**

**at each step.**

**• Now fit the selector on the dataset using the fit\_transform method, which performs the step**

**forward feature selection and returns the transformed data with the selected features.**

**• Finally, we obtain the indices of the selected features using the get\_support method and use**

**these indices to get the names of the selected features from the column names of the original**

**dataset. Then print the selected feature names.**

from sklearn.datasets import load\_wine

from sklearn.linear\_model import LinearRegression

from sklearn.feature\_selection import SequentialFeatureSelector

wine\_data = pd.read\_csv('winequality-red.csv')

X = wine\_data.drop('quality', axis=1)

y = wine\_data['quality']

model = LinearRegression()

selector = SequentialFeatureSelector(estimator=model, direction='forward')

selected\_features = selector.fit\_transform(X, y)

selected\_indices = selector.get\_support(indices=True)

selected\_feature\_names = X.columns[selected\_indices]

print("Selected features:")

for feature\_name in selected\_feature\_names:

print(feature\_name)



**35. Write python code for Principal Component analysis (PCA) for feature extrac-**

**tion. Consider Breast cancer dataset and after applying PCA only two columns are**

**extracted and displayed in the output. The original dataset consists of 569 records and**

**30 features. The new dataset formed after feature extraction consists of 569 records**

**and 2 features.**

**• load the Breast Cancer dataset using the load\_breast\_cancer function from scikit-learn and**

**create separate X (features) and y (target variable) data frames.**

**• then initialize the PCA class from scikit-learn, specifying the number of components we want**

**to retain (in this example, n\_components=2).**

**• fit the PCA model on the dataset using the fit\_transform method, which performs the di-**

**mensionality reduction and returns the transformed data with the specified number of com-**

**ponents.**

**• Next, create a new DataFrame (pca\_df) to store the PCA results. We assign the transformed**

**data to the columns ‘PC1’ and ‘PC2’ (or any other appropriate names). We also include the**

**target variable in the DataFrame for visualization purposes.**

**• print the explained variance ratio using the explained\_variance\_ratio\_ attribute of the PCA**

**object. This shows the amount of variance explained by each principal component.**

**• Finally, print the new DataFrame with the PCA results to inspect the transformed data.**

from sklearn.datasets import load\_breast\_cancer

from sklearn.decomposition import PCA

from sklearn.impute import SimpleImputer

breast\_cancer\_data = pd.read\_csv('breast\_cancer.csv')

X = breast\_cancer\_data.drop(['id', 'diagnosis'], axis=1)

y = breast\_cancer\_data['diagnosis']

# Handle missing values with SimpleImputer

imputer = SimpleImputer(strategy='mean')

X\_imputed = imputer.fit\_transform(X)

pca = PCA(n\_components=2)

transformed\_data = pca.fit\_transform(X\_imputed)

pca\_df = pd.DataFrame(data=transformed\_data, columns=['PC1', 'PC2'])

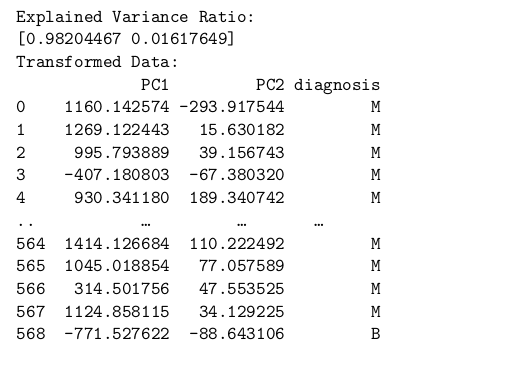
pca\_df['diagnosis'] = y

print("Explained Variance Ratio:")

print(pca.explained\_variance\_ratio\_)

print("Transformed Data:")

print(pca\_df)





**36. Write code to perform operations on a data cube in Python, you can use the**

**pandas library for data manipulation and analysis. Create a sample data cube using**

**a dictionary. The data cube contains columns for “Year”, “Quarter”, “Sales”, and**

**“Expenses”. Create a DataFrame using pd.DataFrame and pass the data dictionary.**

**Demonstrates three common operations (aggregation, slicing and dicing) on a data**

**cube. For Aggregation: use the groupby function to group the data by “Year” and**

**“Quarter” columns, and then apply the sum function to calculate the total sales and**

**expenses for each group. For Slicing: use boolean indexing to filter the data for a**

**specific year (2023) and quarter (1). For Dicing: use boolean indexing to filter the**

**data based on a sales range (> 200) and an expenses range (< 120). Finally, print the**

**original data cube, the aggregated data cube, the sliced data cube, and the diced data**

**cube to observe the results of each operation.**

import pandas as pd

data\_cube = {

'Year': [2022, 2022, 2022, 2023, 2023, 2023],

'Quarter': [1, 2, 3, 1, 2, 3],

'Sales': [100, 200, 150, 250, 300, 350],

'Expenses': [50, 75, 100, 125, 150, 115]

}

df = pd.DataFrame(data\_cube)

print("Original Data Cube:")

print(df)

aggregated\_cube = df.groupby(['Year', 'Quarter']).sum()

print("\nAggregated Data Cube:")

print(aggregated\_cube)

sliced\_cube = df[(df['Year'] == 2023) & (df['Quarter'] == 1)]

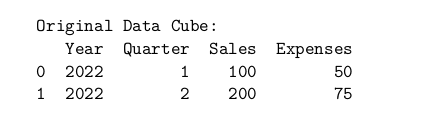
print("\nSliced Data Cube:")

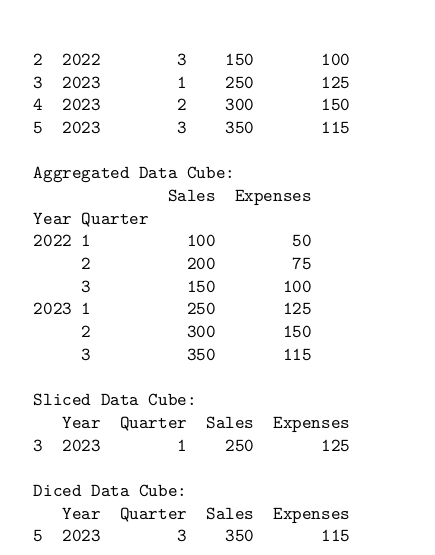
print(sliced\_cube)

diced\_cube = df[(df['Sales'] > 200) & (df['Expenses'] < 120)]

print("\nDiced Data Cube:")

print(diced\_cube)





**37. Write Python code snippet that demonstrates how to perform data discretization**

**using the Pandas library: create a sample dataset with two columns: “Temperature”**

**and “Humidity”. We then create a DataFrame 8using pd.DataFrame and pass the**

**data dictionary. The code demonstrates two types of data discretization:**

**1. Equal-width discretization: We use the pd.cut function to discretize the “Temperature” col-**

**umn into a specified number of bins (num\_bins). The resulting discrete values are represented**

**using numeric labels.**

**2. Equal-frequency discretization: We use the pd.qcut function to discretize the “Humidity”**

**column into a specified number of bins (num\_bins). The resulting discrete values are also**

**represented using numeric labels.**

**Finally, we print the original dataset with the discretized columns to observe the results. Note**

**that you can adjust the number of bins (num\_bins) and apply different discretization techniques**

**based on your specific requirements. Remember to preprocess your data appropriately (e.g., handle**

**missing values, scale numerical features) before performing data discretization for accurate results.**

import pandas as pd

data = {

'Temperature': [25, 30, 22, 20, 35, 28, 32, 18, 26, 29],

'Humidity': [60, 70, 55, 50, 75, 65, 80, 45, 62, 68]

}

df = pd.DataFrame(data)

num\_bins = 3

df['Temperature\_discrete'] = pd.cut(df['Temperature'], bins=num\_bins,

labels=False)

df['Humidity\_discrete'] = pd.qcut(df['Humidity'], q=num\_bins, labels=False)

print("Original Dataset:")

print(df)

