# DAV SAMPLE QUESTIONS

**Module 2 – Regression Analysis 2 Marks – Theory**

1. Explain any two metrics that measure the overall accuracy of the model. 2

Ans]

1. **Accuracy**: Accuracy is a basic metric that measures the overall correctness of a model's predictions. It is calculated as the ratio of the number of correct predictions to the total number of predictions made by the model. Mathematically, it can be expressed as:  
  
 Accuracy = (Number of correct predictions) / (Total number of predictions)

Accuracy is expressed as a percentage, where a higher percentage indicates a higher accuracy of the model. However, accuracy may not always be the most reliable metric, especially when dealing with imbalanced datasets where certain classes may be significantly more frequent than others. In such cases, other metrics such as precision, recall, and F1 score may provide a more comprehensive evaluation of the model's performance.

2. **Confusion Matrix:** A confusion matrix is a tabular representation that provides a detailed breakdown of a model's performance by showing the number of true positive, true negative, false positive, and false negative predictions. It is commonly used to evaluate classification models where the output is discrete, and the goal is to predict the class labels of the input data.

A confusion matrix helps in visualizing the model's performance in terms of both correct and incorrect predictions. From a confusion matrix, several metrics such as precision, recall, and F1 score can be calculated, which provide more insights into the model's accuracy for different classes. The confusion matrix is a useful tool to understand the strengths and weaknesses of a model's predictions and can aid in making improvements to the model's performance.

1. Explain lower t-statistics indicate a predictor should be dropped. 2

Ans)

In a linear regression model, each predictor variable has an associated coefficient estimate and a corresponding t-statistic. The t-statistic is calculated by dividing the estimated coefficient by its standard error. It measures the number of standard errors that the coefficient estimate is away from zero. A higher absolute value of the t-statistic indicates a more significant predictor variable, while a lower absolute value suggests a less significant predictor variable.

A common practice is to use a threshold, such as a significance level (e.g., 0.05 or 0.01), to determine the statistical significance of a predictor. If the absolute value of the t-statistic for a predictor is lower than the threshold, it may indicate that the predictor is not statistically significant in explaining the variation in the dependent variable. In such cases, it may be considered for removal from the model to simplify the model and potentially improve its interpretability and predictive accuracy.

1. What is logistic regression? 1

Ans)

Logistic regression is a statistical method used for modeling the relationship between a binary dependent variable (a variable that can take on one of two possible values, such as 0 or 1) and one or more independent variables (also known as predictor variables or features). It is a type of regression analysis that is commonly used for binary classification problems, where the goal is to predict the probability of an event or outcome belonging to one of two classes.

1. Explain how logistic regression differs from linear regression. 2

Ans)

1. Outcome Variable: In logistic regression, the outcome or dependent variable is binary, taking on one of two possible values (e.g., 0 or 1), representing two classes. On the other hand, in linear regression, the outcome variable is continuous, and can take on any value within a range.
2. Model Output: In logistic regression, the model predicts the probability of an event or outcome belonging to one of the binary classes, and the predicted probabilities are then thresholded to make binary predictions. In contrast, linear regression directly predicts the value of the continuous outcome variable based on the values of the predictor variables.
3. Relationship between Variables: In linear regression, the relationship between the predictor variables and the outcome variable is modeled as a linear relationship, where the coefficients represent the change in the outcome variable per unit change in the predictor variables. However, in logistic regression, the relationship between the predictor variables and the binary outcome variable is modeled using a logistic function (sigmoid function), which maps the predicted values to probabilities.
4. Interpretation of Coefficients: In linear regression, the coefficients represent the change in the outcome variable associated with a unit change in the predictor variable. They can be interpreted as the slope of the linear relationship between the predictor and outcome variables. In logistic regression, the coefficients, also known as log-odds or logits, represent the change in the log-odds of the binary outcome occurring per unit change in the predictor variables. They provide information about the direction and strength of the relationship between the predictor variables and the binary outcome, but cannot be directly interpreted as probabilities.

# Module 3 – Time Series Analysis 2 Marks – Theory

1. What are the components of time series? 2

Ans)

1. The ***trend***refers to the long-term movement in a time series. It indicates whether the observation values are increasing order increasing over time.

Examples of trends are a steady increase in sales month over month or an annual decline in fatalities due to car accidents.

1. The *seasonality* component describes the fixed, periodic fluctuation in the observations over time. As the name suggests, the seasonality component is often related to the calendar. For example, monthly retail sales can fluctuate over the year due to the weather and holidays.
2. A *cyclic* component also refers to a periodic fluctuation, but one that is not as fixed as in the case of a seasonality component. For example, retail sales are influenced *by* the general state of the economy. Thus, a retail sales time series can often follow the lengthy boom-bust cycles of the economy.
3. After accounting for the other three components, the *random* component is what remains. Although noise is certainly part of this random component, there is often some underlying structure to this random component that needs to be modeled to forecast future values of a given time series.

1. Write the steps to perform box-jenkins methodology. 2

Ans)

Developed by George Box and Gwilym Jenkins, the Box-Jenkins methodology for time series analysis involves the following three main steps:

**1.** Condition data and select a model.  
 • Identify and account for any trends or seasonality in the time series. • Examine the remaining time series and determine a suitable model.

**2.** Estimate the model parameters.

**3.** Assess the model and return to Step 1, if necessary.

1. What is the result of the absolute value of ACH(h), when it is closer to 1? 2

Ans)

1. When the absolute value of the autocorrelation function (ACF) at lag h, denoted as ACF(h), is closer to 1, it indicates a strong autocorrelation or a high degree of similarity between the values of a time series at the current time step and the values at the lag h time step.
2. In other words, if ACF(h) is close to 1, it suggests a strong linear relationship between the values of the time series at the current time step and the values at the lag h time step. This indicates that the values of the time series are highly correlated with their past values at the specific lag h.
3. On the other hand, if ACF(h) is closer to 0, it indicates a weak or no autocorrelation between the values of the time series at the current time step and the values at the lag h time step. This suggests that the values of the time series are not significantly correlated with their past values at the specific lag h.
4. The magnitude or closeness of ACF(h) to 1 can provide insights into the strength and direction of the autocorrelation in the time series data. A value of 1 indicates a perfect positive autocorrelation, -1 indicates a perfect negative autocorrelation, and 0 indicates no autocorrelation. Values between 0 and 1 (or 0 and -1) indicate the strength and direction of the autocorrelation, with values closer to 1 (or -1) indicating a stronger autocorrelation.
5. What are the three conditions for stationary time series? 2

Ans)

1. Constant Mean: The mean of the time series remains constant over time. This means that the average of the time series data does not exhibit any trend or systematic change in its central tendency over time.
2. Constant Variance: The variance of the time series remains constant over time. This means that the variability or spread of the time series data does not exhibit any systematic change over time.
3. Constant Autocorrelation: The autocorrelation, which is the correlation between the values of a time series at different time lags, remains constant over time. This means that the degree of similarity or correlation between the values of the time series at different lags does not exhibit any systematic change over time.

# 5 Marks – Sums

1. Explain the application of time series in the following sectors. Finance, Economics, Engineering, Retail and Manufacturing. 2

Ans)

Time series analysis has many applications in finance, economics, biology, engineering, retail, and manufacturing. Here are a few specific use cases:

* Retail sales: For various productlines ,a clothing retailer is looking to forecast future monthly sales. These forecasts need to account for the seasonal aspects of the customer's purchasing decisions. For example, in the northern hemisphere, sweater sales are typically brisk in the fall season, and swimsuit sales are the highest during the late spring and early summer. Thus, an appropriate time series model needs to account for fluctuating demand over the calendar year.
* Spare parts planning: Companies'serviceorganizationshavetoforecastfuturesparepart demands to ensure an adequate supply of parts to repair customer products. Often the spare inventory consists of thousands of distinct part numbers.To forecast future demand, complex models for each part number can be built using input variables such as expected part failure rates, service diagnostic effectiveness, forecasted new product shipments, and forecasted trade-ins/decommission.

However, time series analysis can provide accurate short-term forecasts based simply on prior spare part demand history.

* Stock trading: Some high-frequency stock traders utilize a technique called pairs trading.In pairs trading, an identified strong positive correlation between the prices of two stocks is used to detect a market opportunity . Suppose the stock prices of Company A and Company B consistently move together. Time series analysis can be applied to the difference of these companies' stock prices over time. A statistically larger than expected price difference indicates that it isa good time to buy the stock of Company A and sell the stock of Company B, or vice versa. Of course, th is trading approach depends on theabilitytoexecutethe trade quicklyand be abletodetectwhen thecorrelation inthe stock prices is broken . Pairs trading is one of many techniques that falls in to a trading strategy called statistical arbitrage*.*

1. Which are the models used for forecasting? 2

Ans)

1. Autoregressive Integrated Moving Average (ARIMA): ARIMA is a widely used model for forecasting time series data. It combines three components: autoregression (AR), moving average (MA), and differencing (I) to account for trends, seasonality, and noise in the data. ARIMA models are flexible and can handle a wide range of time series data.
2. Seasonal Autoregressive Integrated Moving-Average (SARIMA): SARIMA is an extension of ARIMA that specifically deals with seasonal time series data. It incorporates seasonal components in addition to the ARIMA components, making it suitable for time series data with seasonal patterns.
3. Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX): ARIMAX is an extension of ARIMA that allows for the inclusion of external or exogenous variables in the forecasting model. These variables can provide additional information to improve the forecast accuracy.
4. Vector Autoregression (VAR): VAR is a multivariate time series model that can forecast multiple variables simultaneously. It models the dependencies and interactions between multiple time series variables, making it useful for forecasting in situations where multiple variables influence each other.
5. Seasonal Decomposition of Time Series (STL): STL is a time series decomposition technique that decomposes a time series into its seasonal, trend, and residual components. These components can be analyzed and modeled separately, and the forecasts can be obtained by combining the forecasts of the individual components.
6. Prophet: Prophet is a forecasting model developed by Facebook that is designed to handle time series data with seasonality, trends, and holidays. It is a flexible and easy-to-use model that incorporates multiple components to capture the patterns in the data.
7. Deep Learning Models: Deep learning models, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, can also be used for time series forecasting. These models are capable of capturing complex patterns in the data and can be especially useful for large-scale and high-dimensional time series data.
8. Explain Auto-correlation function and partial auto-correlation function. 2

Ans)

1. What are the other methods of Time Series Analysis? 2

Ans) Additional time series methods include the following:

* **Autoregressive Moving Average with Exogenous Inputs (ARMAX)** is used to analyze a time series that is dependent on another time series. For example, retail demand for products can be modeled based on the previous dem and combined with weather-related time series such as temperature or rainfall.
* **Spectral analysis** is commonly used for signal processing and other engineering applications. Speech recognition software uses such techniques to separate the signal for the spoken words from the overall signal that may include some noise.
* **Generalized Autoregressive Conditionally Heteroscedastic (GARCH)** is a useful model for addressing time series with nonconstant variance or volatility. GARCH is used for modeling stock market activity and price fluctuations.
* **Kalman filtering** is useful for analyzing real-time inputs about a system that can exist in certain states. Typically, there isan underlying model of how the various components of the system interact and affect each other. A Kalman filter processes the various inputs, attempts to identify the errors in the input, and predicts the current state. For example, a Kalman filter in a vehicle navigation system can process various inputs, such as speed and direction, and update the estimate of the current location.

* **Multivariate time series analysis** examines multiple time series and their effects on each other. VectorARIMA (VARIMA) extendsARIMA by considering a vector of several time-series at a particular time, t . VARIMA can be used in marketing analyses that examine the time series related to a com - pany's price and sales volume as well as related time series for the competitors.

# Module 5 – Data analytics and visualization with R 2 Marks – Theory

1. Explain Kernel density plot in r with proper example. 2

Ans)

A **kernel density plot** is a type of plot that displays the distribution of values in a dataset using one continuous curve.

A kernel density plot is similar to a [histogram](https://www.statology.org/relative-frequency-histogram-r/), but it’s even better at displaying the shape of a distribution since it isn’t affected by the number of bins used in the histogram

### Create One Kernel Density Plot

The following code shows how to create a kernel density plot for one dataset in R:

**#create data**

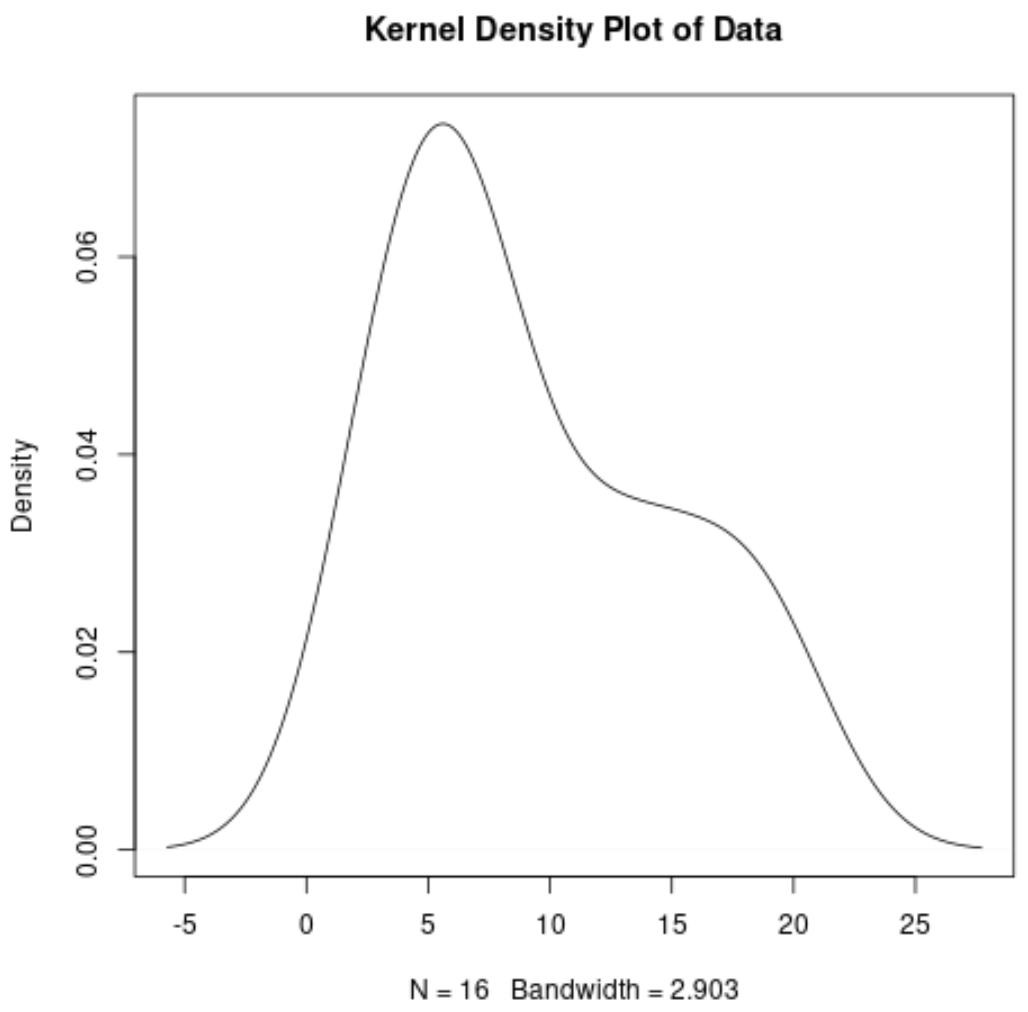
**data <- c(3, 3, 4, 4, 5, 6, 7, 7, 7, 8, 12, 13, 14, 17, 19, 19)**

**#define kernel density**

**kd <- density(data)**

**#create kernel density plot**

**plot(kd, main='Kernel Density Plot of Data')**

****

The x-axis shows the values of the dataset and the y-axis shows the relative frequency of each value. The highest points in the plot shows where the values occur most often.

1. What is main idea for exploratory data analysis and why do you need visualization before analysis? 2

Ans)

* The main idea of exploratory data analysis (EDA) is to visually and analytically examine the main characteristics of a dataset, such as its distribution, central tendency, spread, and relationships between variables. EDA is typically performed as an initial step in the data analysis process to gain insights, identify patterns or trends, and generate hypotheses for further analysis.
* Visualization is an essential part of EDA because it allows us to explore and understand the data in a more intuitive and meaningful way. Here are some reasons why visualization is important before analysis in EDA:

1. Data understanding: Visualization provides a visual representation of the data, allowing us to get a sense of its structure, patterns, and trends.
2. Data exploration: Visualization allows us to visually explore the relationships between variables, identify correlations, and uncover patterns or trends that may not be apparent through other statistical methods.
3. Data validation: Visualization helps in validating data quality by visually identifying any inconsistencies, outliers, or data entry errors
4. Hypothesis generation: It can serve as a starting point for formulating research questions or hypotheses to guide further analysis.
5. Communication and storytelling: Visualization is a powerful tool for communicating findings and insights to stakeholders or team members. It helps in presenting complex information in a clear and visually appealing manner, facilitating better decision-making.

* In summary, visualization is crucial in EDA as it helps in gaining a visual understanding of the data, exploring relationships between variables, validating data quality, generating hypotheses, and communicating findings. It complements statistical analysis by providing a visual context for data exploration and interpretation, and serves as a foundation for further data analysis and decision-making.

# 5 Marks – Theory

1. How would you use facet wrap and facet grid methods of visualization with R and give proper examples. 3

Ans)

1. How will you enhance following R code to display horizontal bar chart and avoid axis labels to overlap? 6

Ans)

# Module 6 – Data analytics and visualization with Python 2 Marks – Theory

1. How would you apply str.cat() to following python code to concatenate address column with name column? 3

Ans)

import pandas as pd

# Create a sample dataframe

df = pd.DataFrame({'name': ['John', 'Jane', 'Alice'],

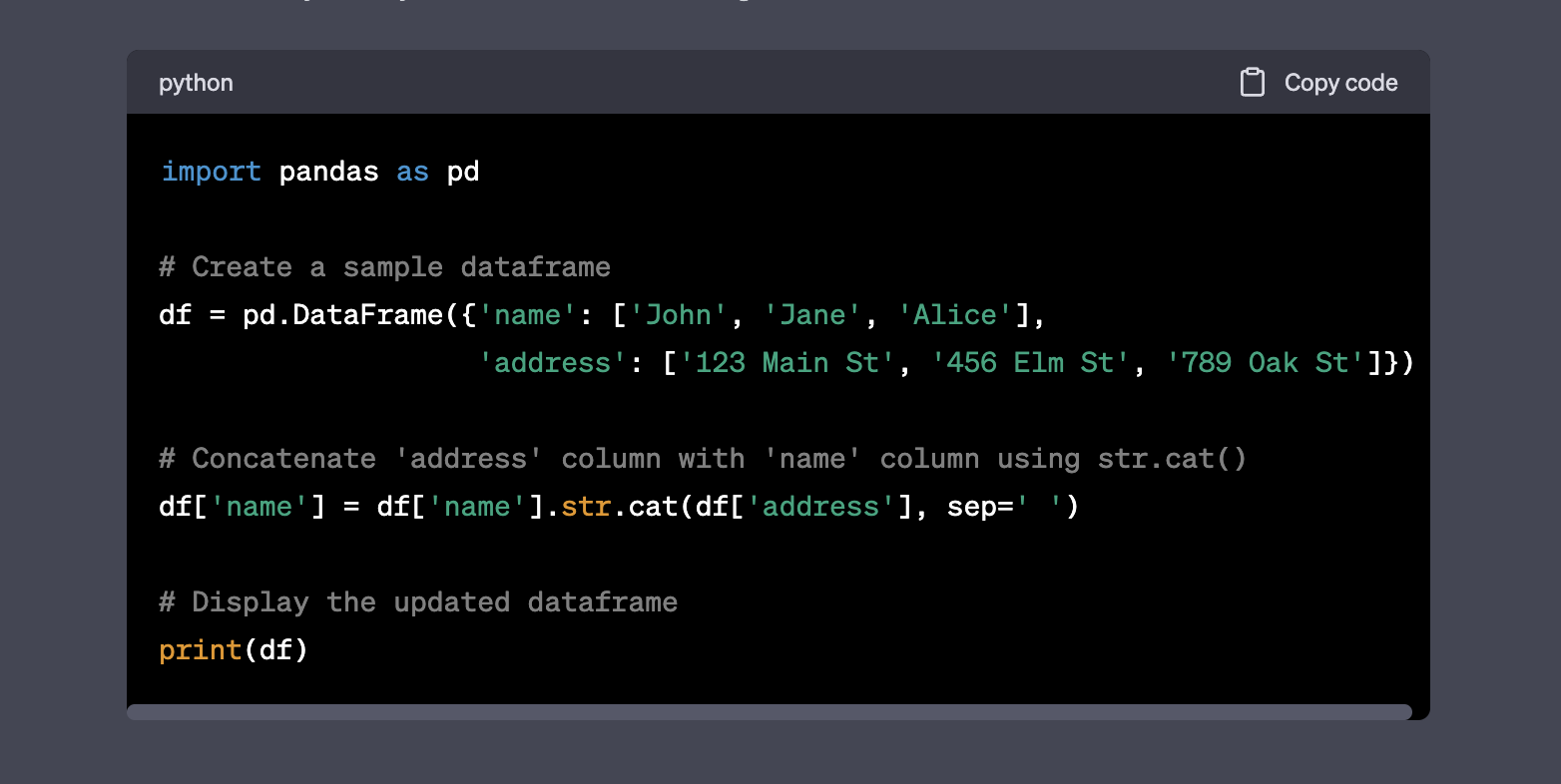
'address': ['123 Main St', '456 Elm St', '789 Oak St']})

# Concatenate 'address' column with 'name' column using str.cat()

df['name'] = df['name'].str.cat(df['address'], sep=' ')

# Display the updated dataframe

print(df)



1. How would you find the determinant and rank for following matrix using python code? 2

Ans)

import numpy as np

# Define the matrix

matrix = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

# Find the determinant of the matrix

determinant = np.linalg.det(matrix)

# Find the rank of the matrix

rank = np.linalg.matrix\_rank(matrix)

# Print the determinant and rank of the matrix

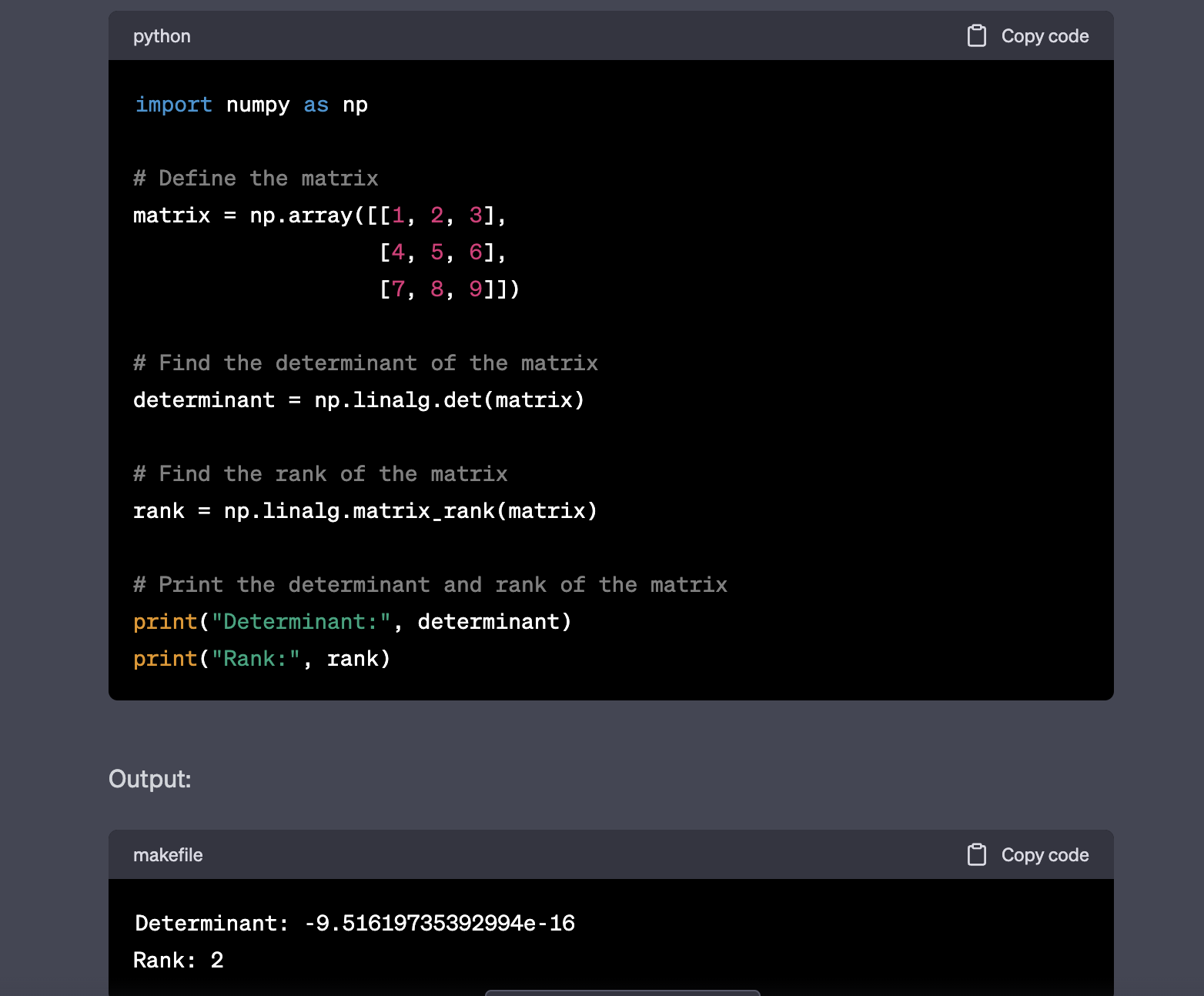
print("Determinant:", determinant)

print("Rank:", rank)

OUTPUT :

Determinant: -9.51619735392994e-16

Rank: 2



# 5 Marks – Theory

1. How would you use CSR and CSC in scipy to handle spare data, explain the methods with examples? 3

Ans)

CSR (Compressed Sparse Row) and CSC (Compressed Sparse Column) are two sparse matrix representations available in SciPy, which are efficient ways to handle large matrices with mostly zero values. Here's an overview of how you can use CSR and CSC in SciPy to handle sparse data, along with examples of their methods:

1. CSR (Compressed Sparse Row):  
   * To create a CSR matrix, you can use the **csr\_matrix()** function from the **scipy.sparse** module. You can pass in the data, indices, and indptr arrays to create the CSR matrix. The **data** array represents the non-zero values, the **indices** array represents the column indices of the non-zero values, and the **indptr** array represents the index pointers to the start of each row in the **data** and **indices** arrays.

Example:  
  
 pythonCopy code

import numpy as np

from scipy.sparse import csr\_matrix

# Create data, indices, and indptr arrays

data = np.array([1, 2, 3, 4, 5, 6])

indices = np.array([0, 2, 3, 0, 1, 2])

indptr = np.array([0, 2, 3, 6])

# Create CSR matrix

csr = csr\_matrix((data, indices, indptr))

print("CSR matrix:")

print(csr)

Output:  
  
 yamlCopy code

CSR matrix:

(0, 0) 1

(0, 2) 2

(0, 3) 3

(1, 0) 4

(1, 1) 5

(1, 2) 6

* + CSR matrix has various methods such as **toarray()** to convert the matrix to a dense NumPy array, **getrow()** and **getcol()** to retrieve rows and columns, **multiply()** for element-wise multiplication with another matrix, and more.

1. CSC (Compressed Sparse Column):  
   * To create a CSC matrix, you can use the **csc\_matrix()** function from the **scipy.sparse** module. You can pass in the data, indices, and indptr arrays to create the CSC matrix. The **data** array represents the non-zero values, the **indices** array represents the row indices of the non-zero values, and the **indptr** array represents the index pointers to the start of each column in the **data** and **indices** arrays.

Example:  
  
 pythonCopy code

import numpy as np

from scipy.sparse import csc\_matrix

# Create data, indices, and indptr arrays

data = np.array([1, 2, 3, 4, 5, 6])

indices = np.array([0, 1, 1, 2, 2, 2])

indptr = np.array([0, 0, 2, 5, 6])

# Create CSC matrix

csc = csc\_matrix((data, indices, indptr))

print("CSC matrix:")

print(csc)

Output:  
  
 yamlCopy code

CSC matrix:

(0, 0) 1

(1, 0) 2

(1, 1) 3

(2, 1) 4

(2, 2) 5

(2, 3) 6

* + CSC matrix has similar methods as CSR matrix, such as **toarray()**

1. How would you use facet grid and pair grid methods of visualization with Python seaborn and give proper examples? 3

Ans)

### seaborn.FacetGrid() :

* FacetGrid class helps in visualizing distribution of one variable as well as the relationship between multiple variables separately within subsets of your dataset using multiple panels.
* A FacetGrid can be drawn with up to three dimensions ? row, col, and hue. The first two have obvious correspondence with the resulting array of axes; think of the hue variable as a third dimension along a depth axis, where different levels are plotted with different colors.
* FacetGrid object takes a dataframe as input and the names of the variables that will form the row, column, or hue dimensions of the grid. The variables should be categorical and the data at each level of the variable will be used for a facet along that axis.

# importing packages

import seaborn

import matplotlib.pyplot as plt

# loading of a dataframe from seaborn

df = seaborn.load\_dataset('tips')

############# Main Section #############

# Form a facetgrid using columns with a hue

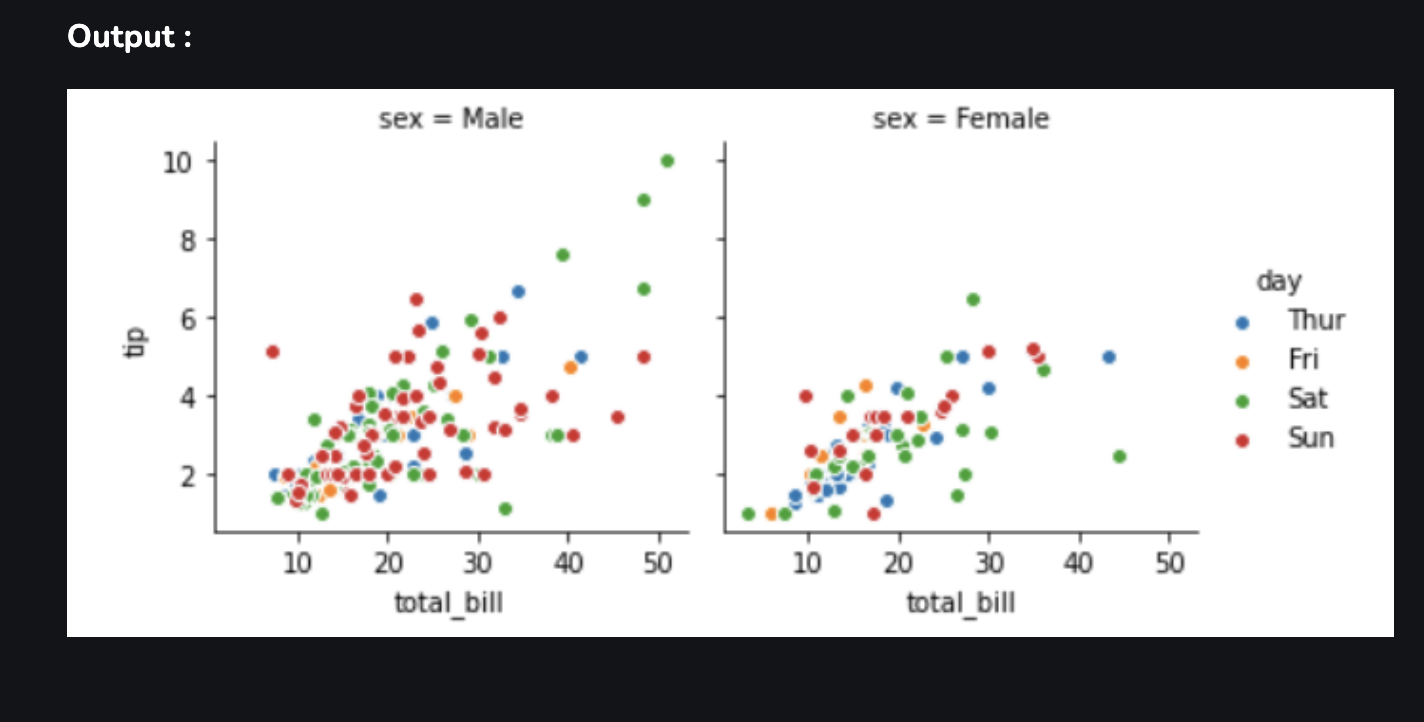
graph = seaborn.FacetGrid(df, col ="sex", hue ="day")

# map the above form facetgrid with some attributes

graph.map(plt.scatter, "total\_bill", "tip", edgecolor ="w").add\_legend()

# show the object

plt.show()



### seaborn.PairGrid() :

* Subplot grid for plotting pairwise relationships in a dataset.
* This class maps each variable in a dataset onto a column and row in a grid of multiple axes. Different axes-level plotting functions can be used to draw bivariate plots in the upper and lower triangles, and the marginal distribution of each variable can be shown on the diagonal.
* It can also represent an additional level of conditionalization with the hue parameter, which plots different subsets of data in different colors. This uses color to resolve elements on a third dimension, but only draws subsets on top of each other and will not tailor the hue parameter for the specific visualization the way that axes-level functions that accept hue will.

# importing packages

import seaborn

import matplotlib.pyplot as plt

# loading dataset

df = seaborn.load\_dataset('tips')

# PairGrid object with hue

graph = seaborn.PairGrid(df, hue ='day')

# type of graph for diagonal

graph = graph.map\_diag(plt.hist)

# type of graph for non-diagonal

graph = graph.map\_offdiag(plt.scatter)

# to add legends

graph = graph.add\_legend()

# to show

plt.show()

