CMR TECHNICAL CAMPUS

Kandlakoya(V), Medchal Road, Hyderabad – 501 401

An UGC Autonomous Institute

Accredited by NBA and NAAC with A Grade

Approved by AICTE, New Delhi and Affiliated to JNTU, Hyderabad

DEPARTMENT OF CSE (AI&ML)

Data Analytics Lab

(22AM604PC)

LAB MANUAL

(R22)

List of Experiments

Sl.No	Programs
1.	Write a python program to perform the following data Preprocessing Methods a) Handling Missing Values b) Noise detection and removal c) Identifying data redundancy and elimination
2.	Write a python to apply any one imputation model on a sample data set.
3.	Write a python to apply Linear Regression Algorithm on a data set.
4.	Write a python to apply Logistic Regression algorithm on a data set.
5.	Write a python to apply Decision Tree Induction for classification on a data set.
6.	Write a python to apply Random Forest Algorithm on a data set.
7.	Write a python to apply ARIMA on Time series Data.
8.	Write a python for Object segmentation using hierarchical based methods.
9.	Write a python programming to perform Visualization techniques (types of Maps-Bar, column, line, Scatter,3D Cubes etc.)
10.	Write a python programming to Descriptive analytics on healthcare data.
11.	Write a python program to perform predictive analytics on Product Sales Data.
12.	Write a python program to Apply predictive analytics for weather forecasting.

WEEK 1:

Write a python program to perform the following data Preprocessing Methods

a) **Handling Missing Values**
b) **Noise detection and removal**
c) **Identifying data redundancy and elimination.**
a) **Handling Missing Values**

```
import pandas as pd

# Read the dataset from a CSV file (replace 'your_dataset.csv' with your actual file)

file_path = r'C:\Users\hi\Downloads\laptop.csv'

df = pd.read_csv(file_path)
# Display the original dataset

print("Original Dataset:")

print(df)

# Handling missing values: Drop or Impute

# Option 1: Drop rows with missing values

df_dropped = df.dropna()

# Option 2: Fill missing values with mean, median, or a specific value

df_filled = df.fillna(df.mean())  # You can use df.median() or a specific value as well
```

 # Display the preprocessed datasets

print("\nDataset after dropping missing values:")

print(df_dropped)

print("\nDataset after filling missing values:")

print(df_filled)

 # Save the cleaned datasets to new CSV files

df_dropped.to_csv('cleaned_dataset_dropped.csv', index=False)

df_filled.to_csv('cleaned_dataset_filled.csv', index=False)

output:

Original Dataset:

```
   Unnamed: 0  Brand       Model Name Processor Operating System Storage  \
0         0.0     HP     15s-fq5007TU   Core i3  Windows 11 Home  512 GB
1         1.0     HP     15s-fy5003TU   Core i3  Windows 11 Home  512 GB
2         2.0  Apple  2020 Macbook Air      NaN   Mac OS Big Sur  256 GB
3         3.0  Apple  2020 Macbook Air       M1   Mac OS Big Sur  256 GB

    RAM         Screen Size Touch_Screen   Price
0   8 GB  39.62 cm (15.6 Inch)         No  38,990
1   8 GB  39.62 cm (15.6 Inch)        NaN  37,990
2   8 GB  33.78 cm (13.3 inch)         No  70,990
3   8 GB  33.78 cm (13.3 inch)         No  70,990
```

Dataset after dropping missing values:

```
   Unnamed: 0  Brand       Model Name Processor Operating System Storage  \
0         0.0     HP     15s-fq5007TU   Core i3  Windows 11 Home  512 GB
3         3.0  Apple  2020 Macbook Air       M1   Mac OS Big Sur  256 GB

    RAM         Screen Size Touch_Screen   Price
0   8 GB  39.62 cm (15.6 Inch)         No  38,990
3   8 GB  33.78 cm (13.3 inch)         No  70,990
```

Dataset after filling missing values:

```
   Unnamed: 0  Brand       Model Name Processor Operating System Storage  \
0         0.0     HP     15s-fq5007TU   Core i3  Windows 11 Home  512 GB
```

1	1.0	HP	15s-fy5003TU	Core i3	Windows 11 Home	512 GB
2	2.0	Apple	2020 Macbook Air	NaN	Mac OS Big Sur	256 GB
3	3.0	Apple	2020 Macbook Air	M1	Mac OS Big Sur	256 GB

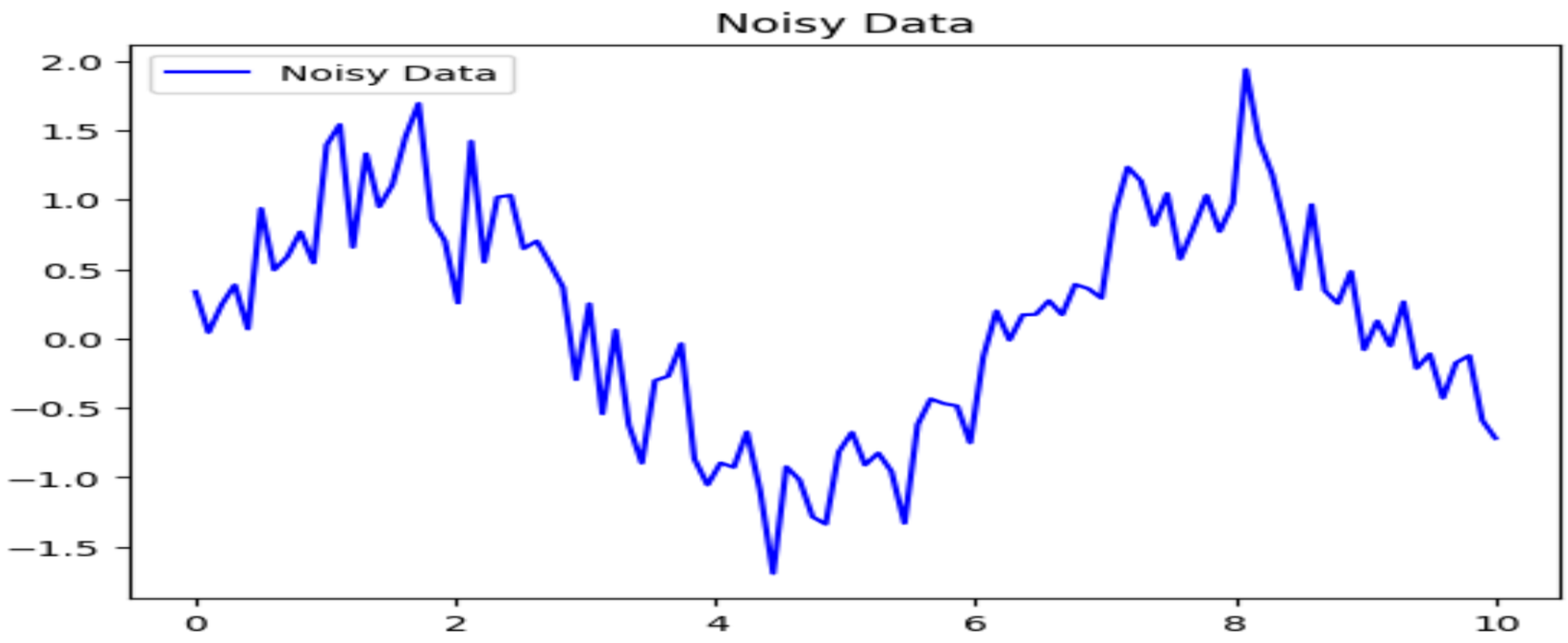
	RAM	Screen Size	Touch_Screen	Price
0	8 GB	39.62 cm (15.6 Inch)	No	38,990
1	8 GB	39.62 cm (15.6 Inch)	NaN	37,990
2	8 GB	33.78 cm (13.3 inch)	No	70,990
3	8 GB	33.78 cm (13.3 inch)	No	70,990

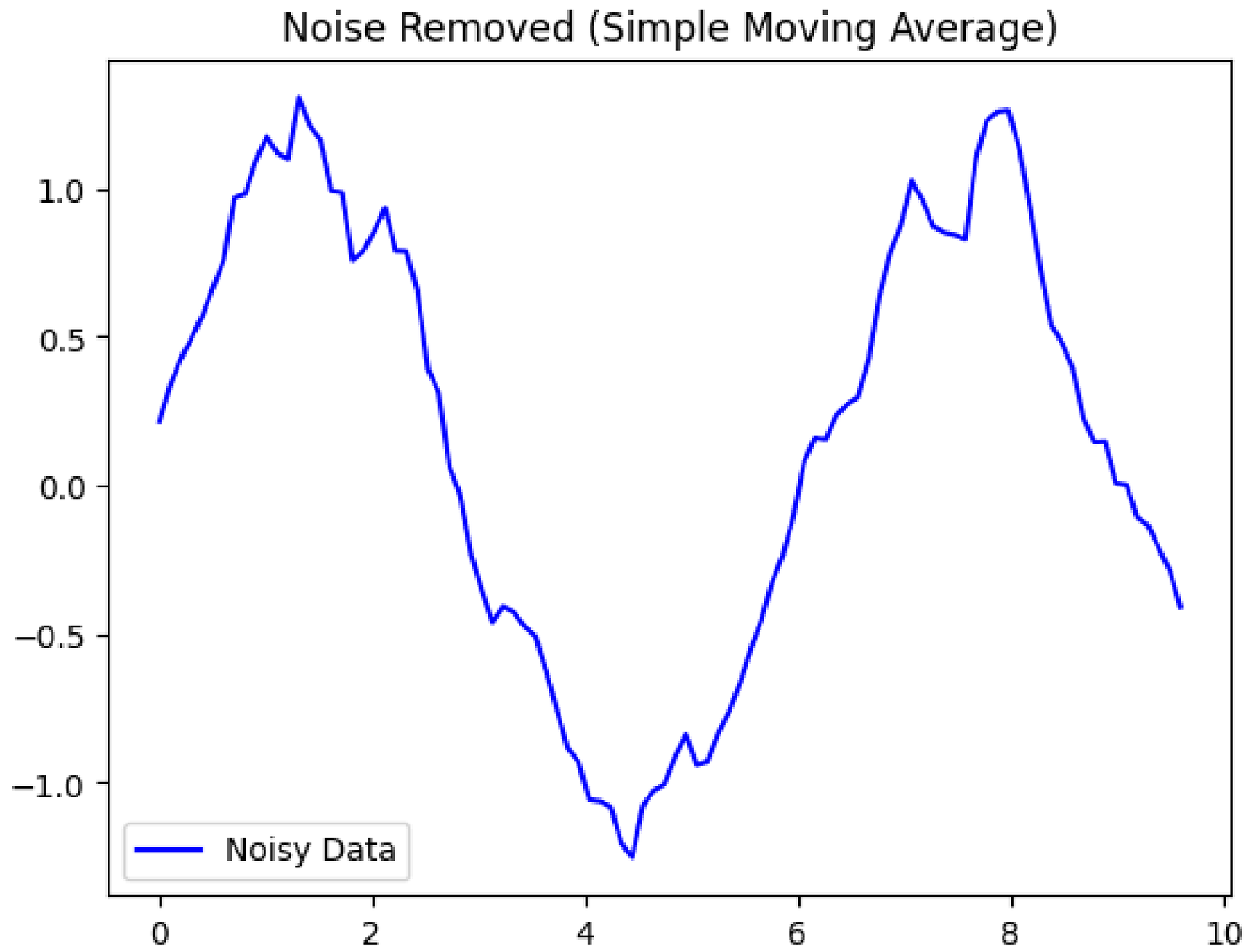
b) Noise detection and removal

```
import numpy as np
import matplotlib.pyplot as plt
def generate_noisy_data():
    # Generate noisy data for demonstration
    x = np.linspace(0, 10, 100)
    y = np.sin(x) + 0.3 * np.random.randn(100)
    return x, y
def plot_data(x, y, title):
    # Plot the original data
    plt.figure()
    plt.plot(x, y, 'b', label='Noisy Data')
    plt.title(title)
    plt.legend()
    plt.show()
def remove_noise_simple(x, y):
    # Simple moving average for noise removal
    window_size = 5
    smoothed_y = np.convolve(y, np.ones(window_size)/window_size, mode='valid')
    return x[:len(smoothed_y)], smoothed_y
```

```
def main():
    # Generate and plot noisy data
    x, noisy_data = generate_noisy_data()
    plot_data(x, noisy_data, 'Noisy Data')
    # Remove noise using a simple moving average
    x_smoothed, smoothed_data = remove_noise_simple(x, noisy_data)
    plot_data(x_smoothed, smoothed_data, 'Noise Removed (Simple Moving Average)')
if __name__ == "__main__":
    main()
```

output





c) Identifying data redundancy and elimination.

```
def eliminate_redundancy(data):
    unique_data = set(data)
    return list(unique_data)
def main():
    # Example data with redundancy
    input_data = [1, 2, 3, 4, 1, 2, 5, 6, 7, 3, 8, 9, 5]
    print("Original Data:", input_data)
    # Perform data processing to eliminate redundancy
    processed_data = eliminate_redundancy(input_data)
    print("Processed Data (Redundancy Eliminated):", processed_data)
if __name__ == "__main__":
    main()
```

output

Original Data: [1, 2, 3, 4, 1, 2, 5, 6, 7, 3, 8, 9, 5]
Processed Data (Redundancy Eliminated): [1, 2, 3, 4, 5, 6, 7, 8, 9]

Sample viva Questions:

1. **List the data processing methods in python.**

There are 4 main important steps for the pre-processing of data.

- Splitting of the data set in Training and Validation sets.
- Taking care of Missing values.
- Taking care of Categorical Features.
- Normalization of data set.

2. **How do you impute missing values pandas?**

If there are way too many missing values in a column, then you can drop that column. Otherwise we can impute missing values with mean, median and mode.

3. **What are the techniques for missing value imputation in Python?**

You can impute missing values in a Python Data Frame using various techniques such as mean, median, mode, or using predictive models. You can use the fillna() method of a Pandas Data Frame to perform imputation.

4. **What are the techniques used in noise detection?**

There are different techniques to remove noise from the digital images, such as filtering and machine learning techniques. This is also termed as denoising. In the denoising process, the initial requirement is to detect and identify the type of noise from the given image.

5. **Which technique is used to remove noise?**

Median Filtering: This is a non-linear method that is used to remove noise from an image by replacing the intensity value of each pixel with the median value of the surrounding pixels. Median filtering is effective at removing salt and pepper noise, but it can also blur edges in the image.

6. **How do you remove noise from a data set?**

Cleaning noisy data in machine learning model is important for the quality results. Noisy data can be handled by missing value imputation, outlier detection, data

standardization&normalization, data encoding, data validation, text processing, cross validation, feature sampling and ensemble methods

WEEK 2:

Write a python to apply any one imputation model on a sample data set.

```
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load a sample dataset (in this case, Iris dataset)

iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['target'] = iris.target

# Introduce some missing values for demonstration purposes
data.iloc[2:5, 1] = None
data.iloc[8:10, 2] = None

# Separate features and target

X = data.drop('target', axis=1)
y = data['target']

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply mean imputation on the training set
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
```

 # Display the original and imputed data

```
print("Original Data:")
print(X_train.head())

print("\nImputed Data:")
print(pd.DataFrame(X_train_imputed, columns=X.columns).head())
```

output:

Original Data:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
22	4.6	3.6	1.0	0.2
15	5.7	4.4	1.5	0.4
65	6.7	3.1	4.4	1.4
11	4.8	3.4	1.6	0.2
42	4.4	3.2	1.3	0.2

Imputed Data:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	4.6	3.6	1.0	0.2
1	5.7	4.4	1.5	0.4
2	6.7	3.1	4.4	1.4
3	4.8	3.4	1.6	0.2
4	4.4	3.2	1.3	0.2

Sample viva Questions:

1. **How to do data imputation in Python?**

Use the SimpleImputer() function from sklearn module to impute the values. Pass the strategy as an argument to the function. It can be either mean or mode or median. The problem with the previous model is that the model does not know whether the values came from the original data or the imputed value.

2. **What is the imputation method of data?**

Data imputation is a method for retaining the majority of the dataset's data and information by substituting missing data with a different value. These methods are employed because it would be impractical to remove data from a dataset each time.

3. What is imputation model for missing data?

Imputation preserves all cases by replacing missing data with an estimated value based on other available information. Once all missing values have been imputed, the data set can then be analyzed using standard techniques for complete data.

4. What are the three types of missing data?

Missing data are typically grouped into three categories:

- Missing completely at random (MCAR). When data are MCAR, the fact that the data are missing is independent of the observed and unobserved data. ...
- Missing at random (MAR). ...
- Missing not at random (MNAR

5. What are four reasons for missing data?

Many reasons for missing data ...

- People do not respond to survey (or specific questions in a survey).
- Species are rare and cannot be found or sampled.
- The individual dies or drops out before sampling.
- Some things are easier to measure than others.
- Data entry errors.
- Many others!

WEEK 3:

Write a python to apply Linear Regression Algorithm on a data set.

 # Import necessary libraries

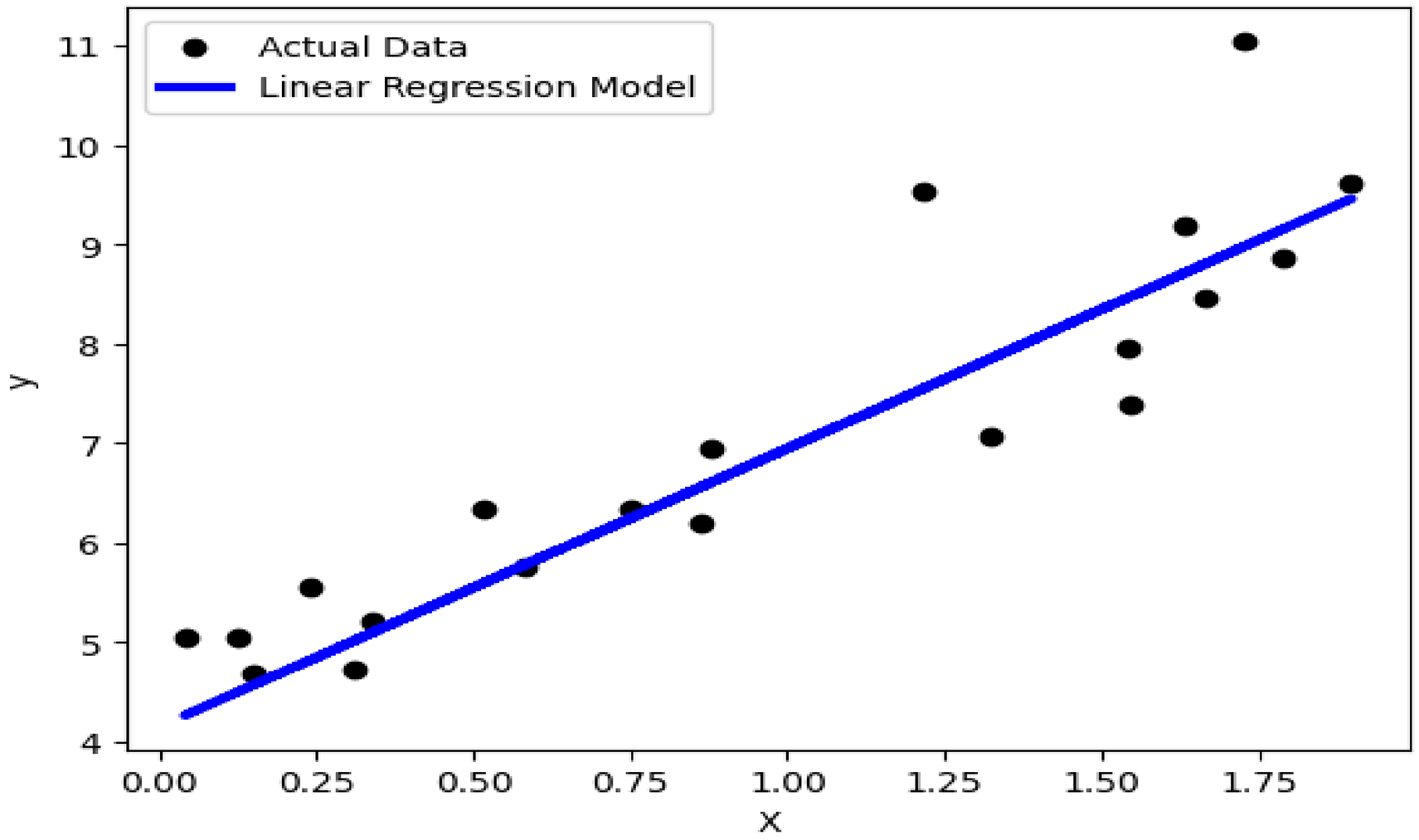
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

 # Generate a sample dataset

```
np.random.seed(42)
X = 2 * np.random.rand(100, 1)
```

```
y = 4 + 3 * X + np.random.randn(100, 1)

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a linear regression model
model = LinearRegression()

# Train the model on the training set
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Plot the results
plt.scatter(X_test, y_test, color='black', label='Actual Data')

plt.plot(X_test, y_pred, color='blue', linewidth=3, label='Linear Regression Model')

plt.xlabel('X')

plt.ylabel('y')

plt.legend()a

plt.show()
```

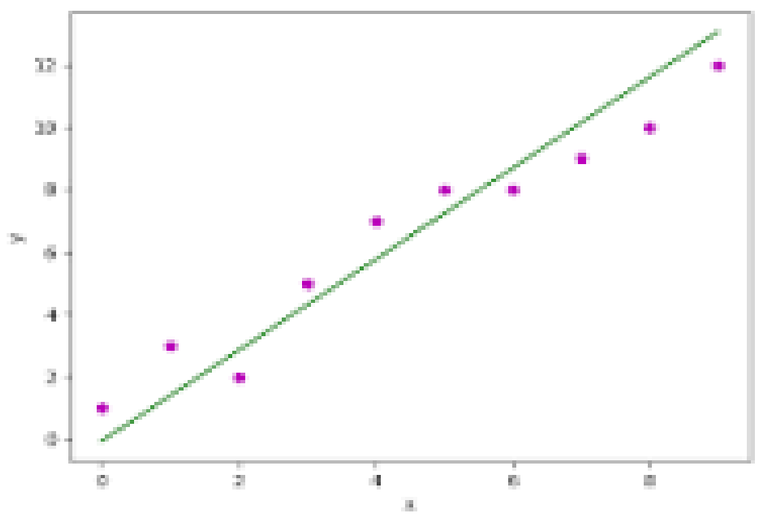
output



Sample viva Questions:

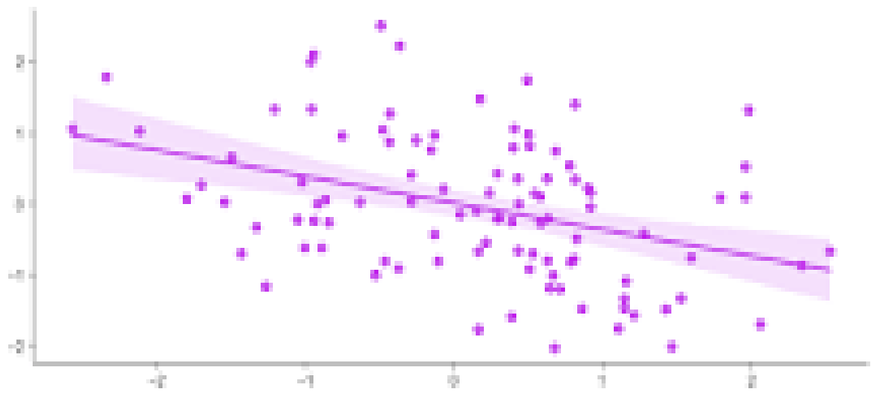
1. **What is linear regression in Python?**

Linear regression is a statistical method for modeling relationships between a dependent variable with a given set of independent variables. Note: In this article, we refer to dependent variables as responses and independent variables as features for simplicity.



2. **What are the regression methods in Python?**

most widely used regression algorithms in Python and Machine Learning, including Linear Regression, Polynomial Regression, Ridge Regression, Lasso Regression, and Elastic Net Regression, Decision Tree based methods and Support Vector Regression (SVR).

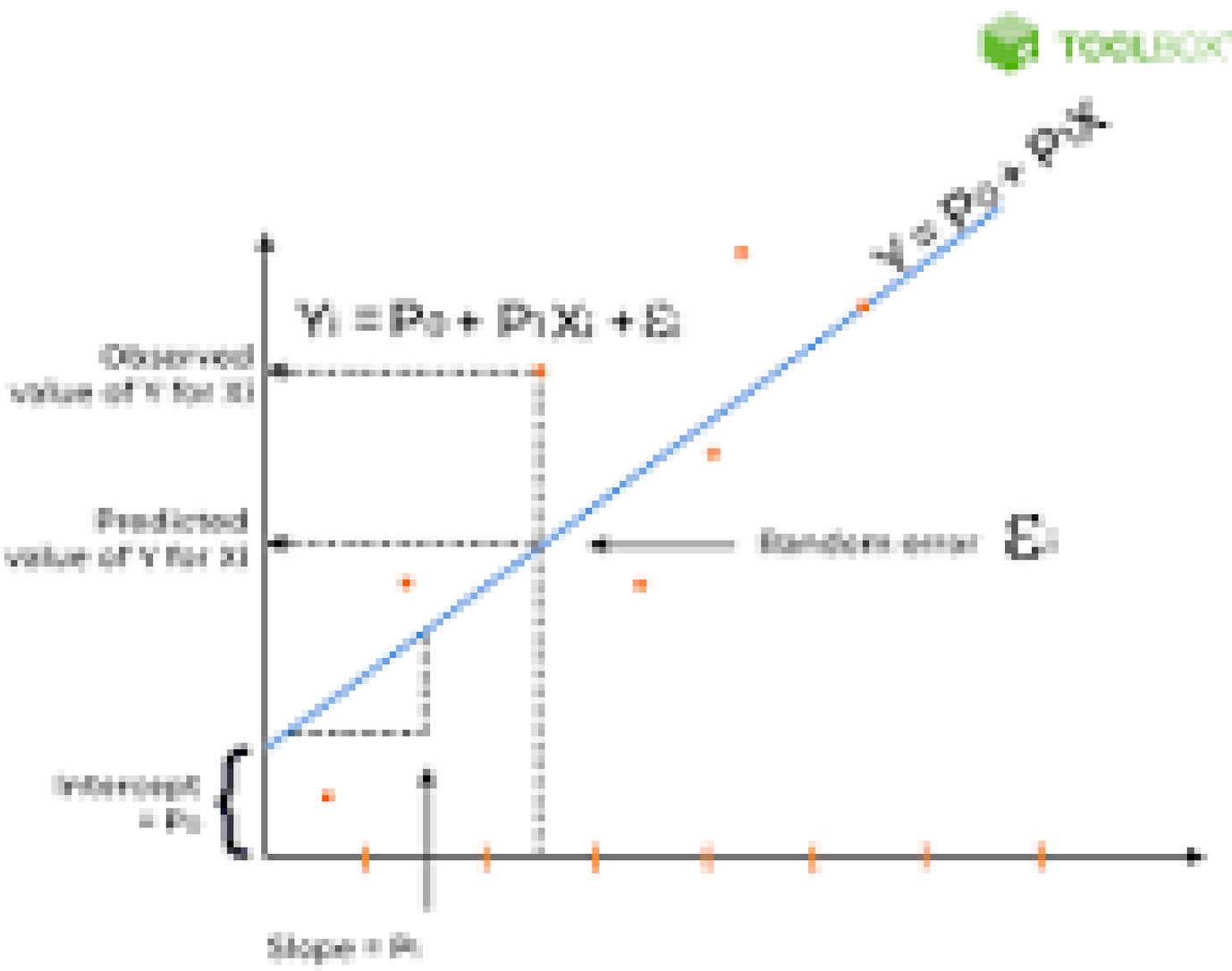


3. What is linear regression used for?

Linear regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. It mathematically models the unknown or dependent variable and the known or independent variable as a linear equation.

4. What is linear regression algorithm?

Linear regression is an algorithm that provides a linear relationship between an independent variable and a dependent variable to predict the outcome of future events. It is a statistical method used in data science and machine learning for predictive analysis.



5. What is a simple linear regression?

Definition. Simple linear regression aims to find a linear relationship to describe the correlation between an independent and possibly dependent variable. The regression line can be used to predict or estimate missing values, this is known as interpolation.

WEEK 4:

Write a python to apply Logistic Regression algorithm on a data set.

 # Import necessary libraries

```
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report

# Generate sample dataset (you can replace this with your own dataset)

np.random.seed(42)

X = np.random.rand(100, 2)  # Features

y = (X[:, 0] + X[:, 1] > 1).astype(int)  # Binary classification label

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model

model = LogisticRegression()

# Train the model

model.fit(X_train, y_train)
```

 # Make predictions on the test set

y_pred = model.predict(X_test)

 # Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

classification_rep = classification_report(y_test, y_pred)

 # Display results

print(f"Accuracy: {accuracy}")

print("Classification Report:")

print(classification_rep)

output

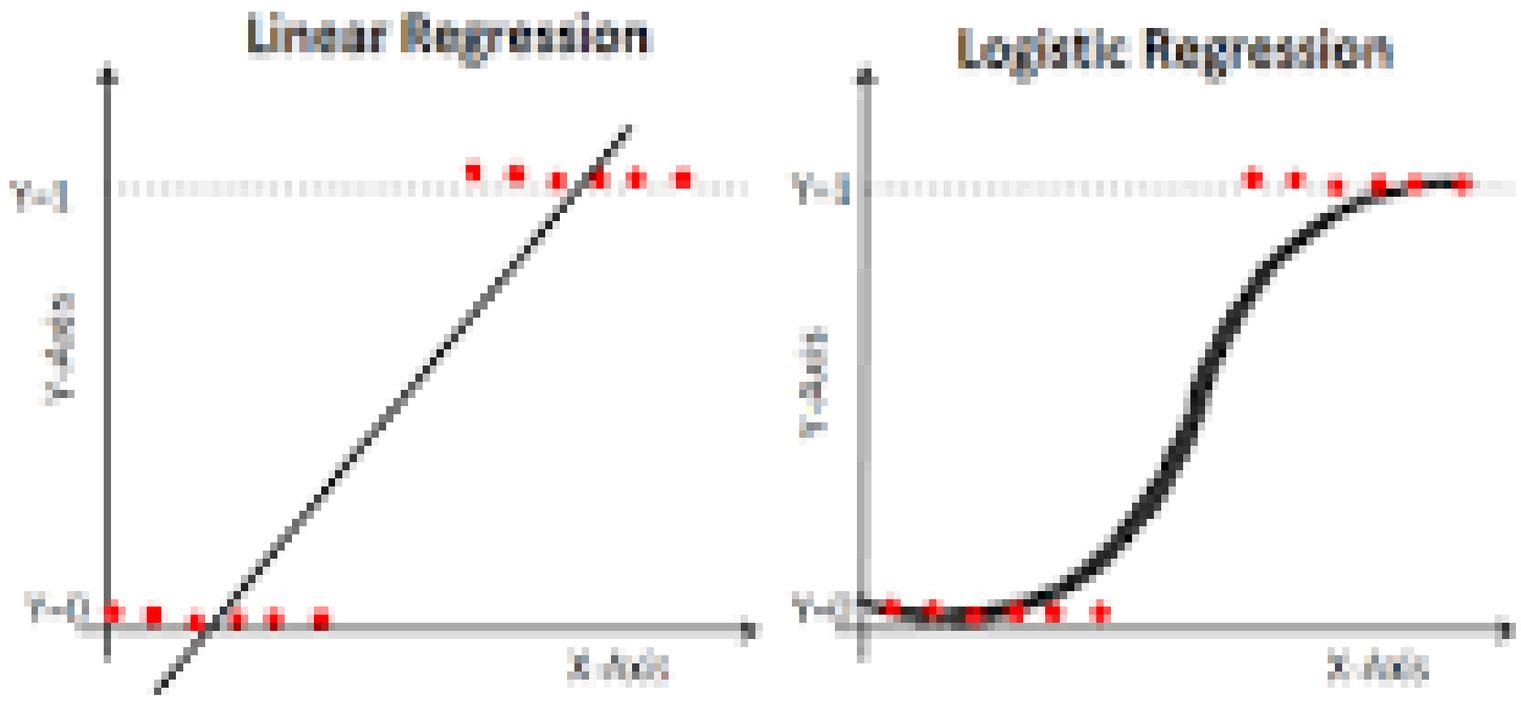
Accuracy: 0.85
Classification Report:

	precision	recall	f1-score	support
0	0.77	1.00	0.87	10
1	1.00	0.70	0.82	10
accuracy			0.85	20
macro avg	0.88	0.85	0.85	20
weighted avg	0.88	0.85	0.85	20

Sample viva Questions:

1. **What is logistic regression in Python?**

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning.



2. **What are the 3 types of logistic regression?**

There are three main types of logistic regression: binary, multinomial and ordinal. They differ in execution and theory.

3. **What are the steps of logistic regression?**

we will dive into the working principles of Logistic regression step-by-step, to understand how it can classify data.

- Step 1: Sigmoid Function. ...
- Step 2: Hypothesis Function. ...
- Step 3: Cost Function. ...
- Step 4: Gradient Descent. ...
- Step 5: Decision Boundary.

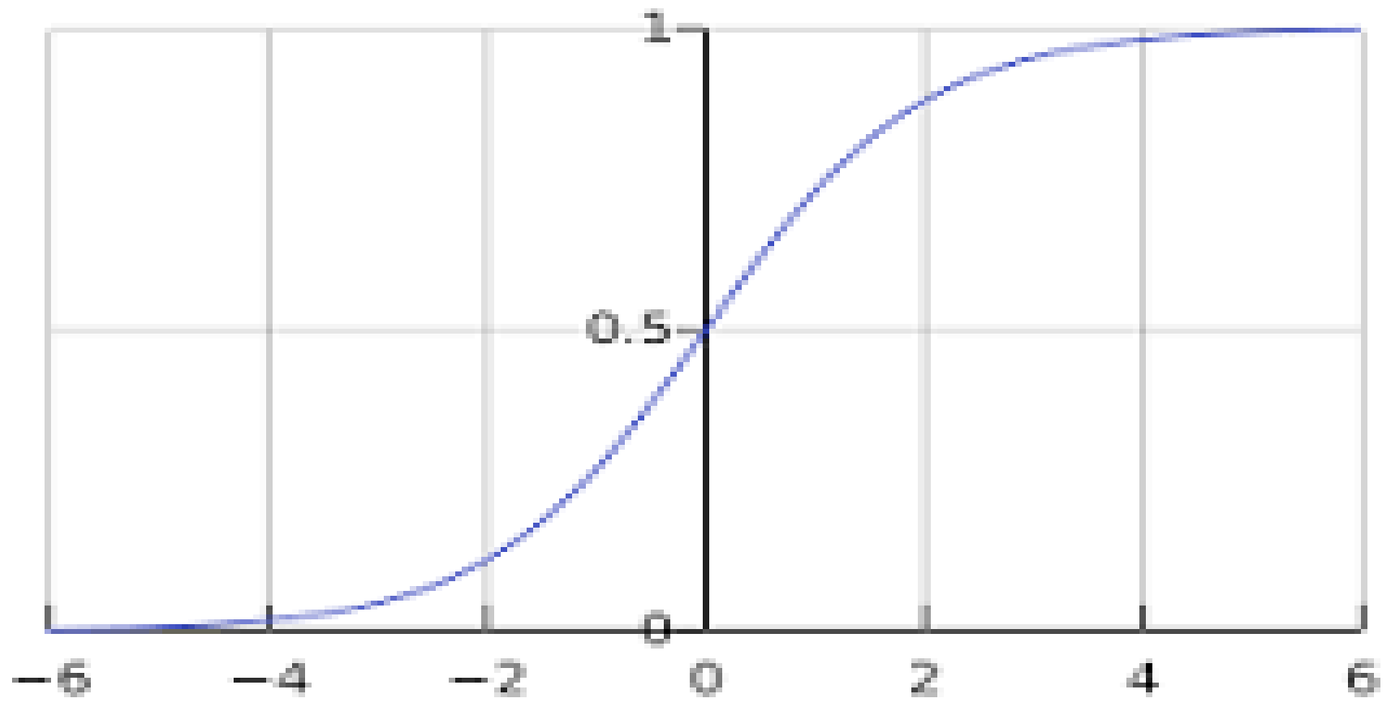
4. **What are the disadvantages of logistic regression?**

Disadvantages of logistic regression are

- Logistic regression fails to predict a continuous outcome. ...
- Logistic regression assumes linearity between the predicted (dependent) variable and the predictor (independent) variables. ...
- Logistic regression may not be accurate if the sample size is too small.

5. **What is a real life example of logistic regression?**

Toxic speech detection, topic classification for questions to support, and email sorting are examples where logistic regression shows good results. Other popular algorithms for making a decision in these fields are support vector machines and random forest.



WEEK 5:

Write a python to apply Decision Tree Induction for classification on a data set.

```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
from sklearn.metrics import accuracy_score

# Load a sample dataset (Iris dataset in this case)
iris = datasets.load_iris()
```

```
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Decision Tree Classifier
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
predictions = clf.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

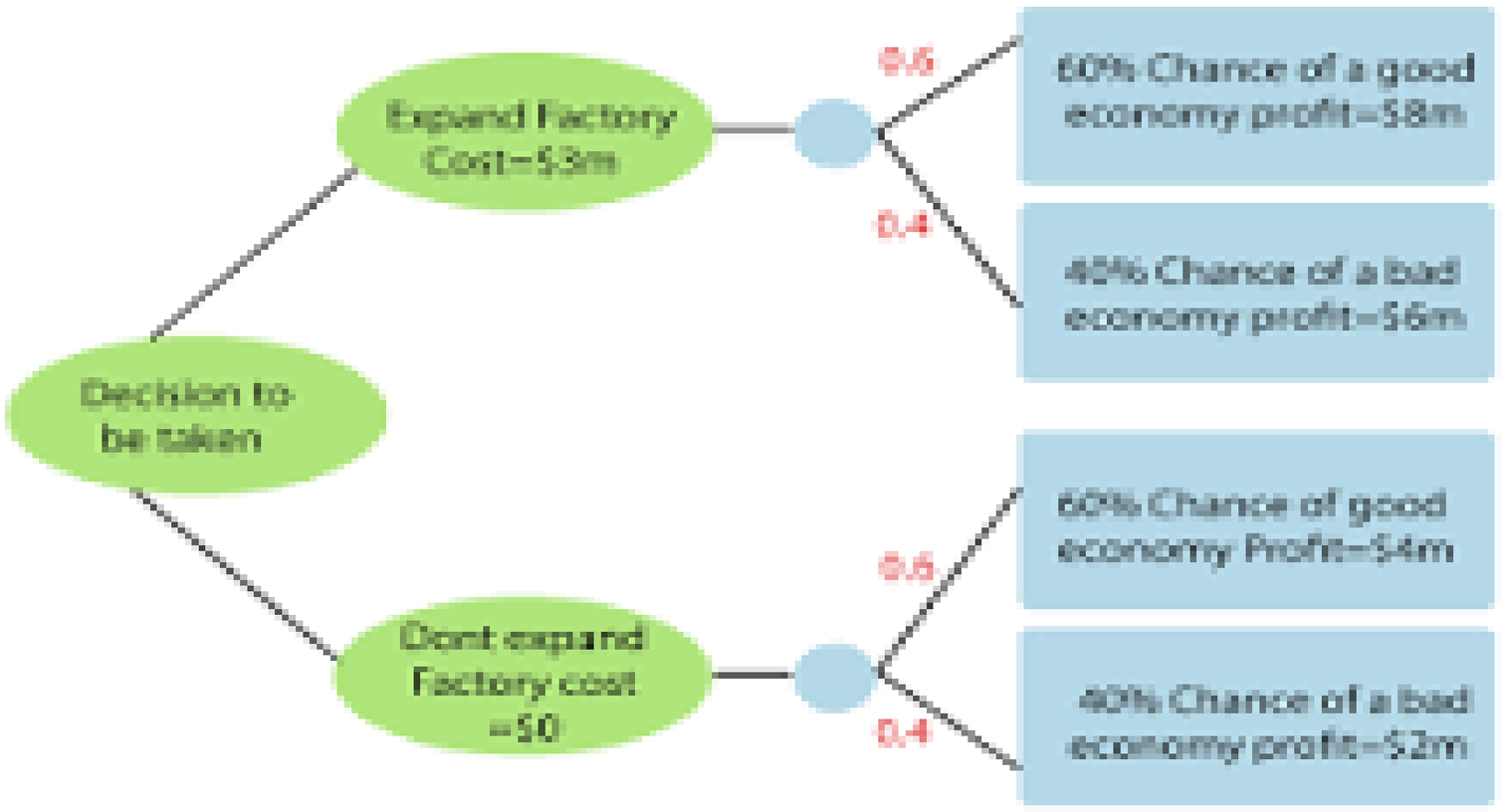
output

Accuracy: 100.00%

Sample viva Questions:

1. **How decision tree induction is used in classification?**

The decision tree creates classification or regression models as a tree structure. It separates a data set into smaller subsets, and at the same time, the decision tree is steadily developed. The final tree is a tree with the decision nodes and leaf nodes. A decision node has at least two branches.



2. How can a decision tree be used to classify data?

Decision tree learning employs a divide and conquer strategy by conducting a greedy search to identify the optimal split points within a tree. This process of splitting is then repeated in a top-down, recursive manner until all, or the majority of records have been classified under specific class labels.

3. How do you find the entropy of a decision tree in Python?

Steps to calculate entropy for a split:

1. Calculate the entropy of the parent node.
2. Calculate entropy of each individual node of split and calculate the weighted average of all sub-nodes available in the split. ...
3. calculate information gain as follows and chose the node with the highest information gain for splitting.

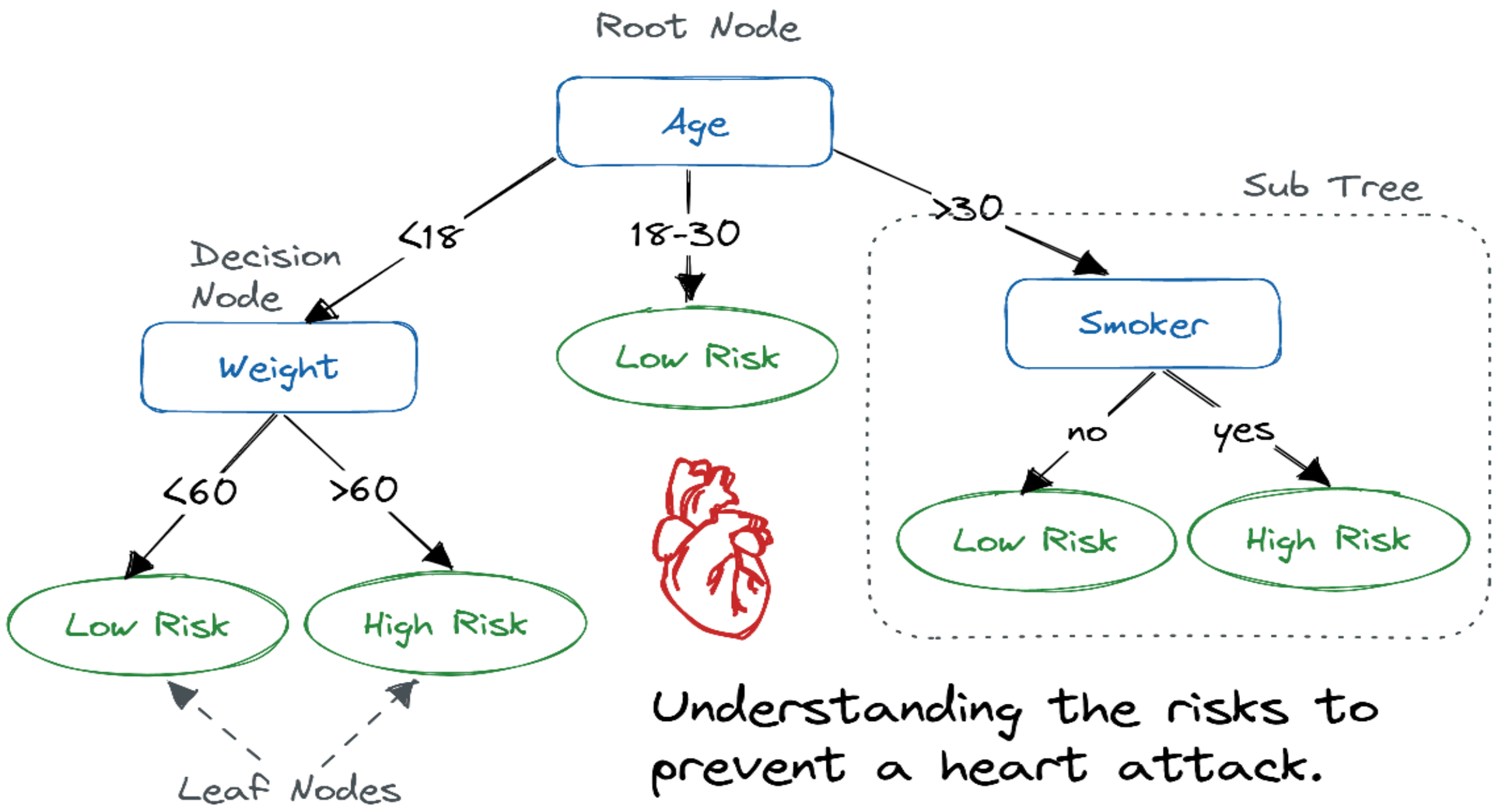
4. Why are decision trees good for classification?

The main advantage of the decision tree classifier is its ability to using different feature subsets and decision rules at different stages of classification. As shown in Figure 4.6, a general decision tree consists of one root node, a number of internal and leaf nodes, and branches.

5. What is decision tree classifier in Python?

A decision tree is a flowchart-like tree structure where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node

represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value.

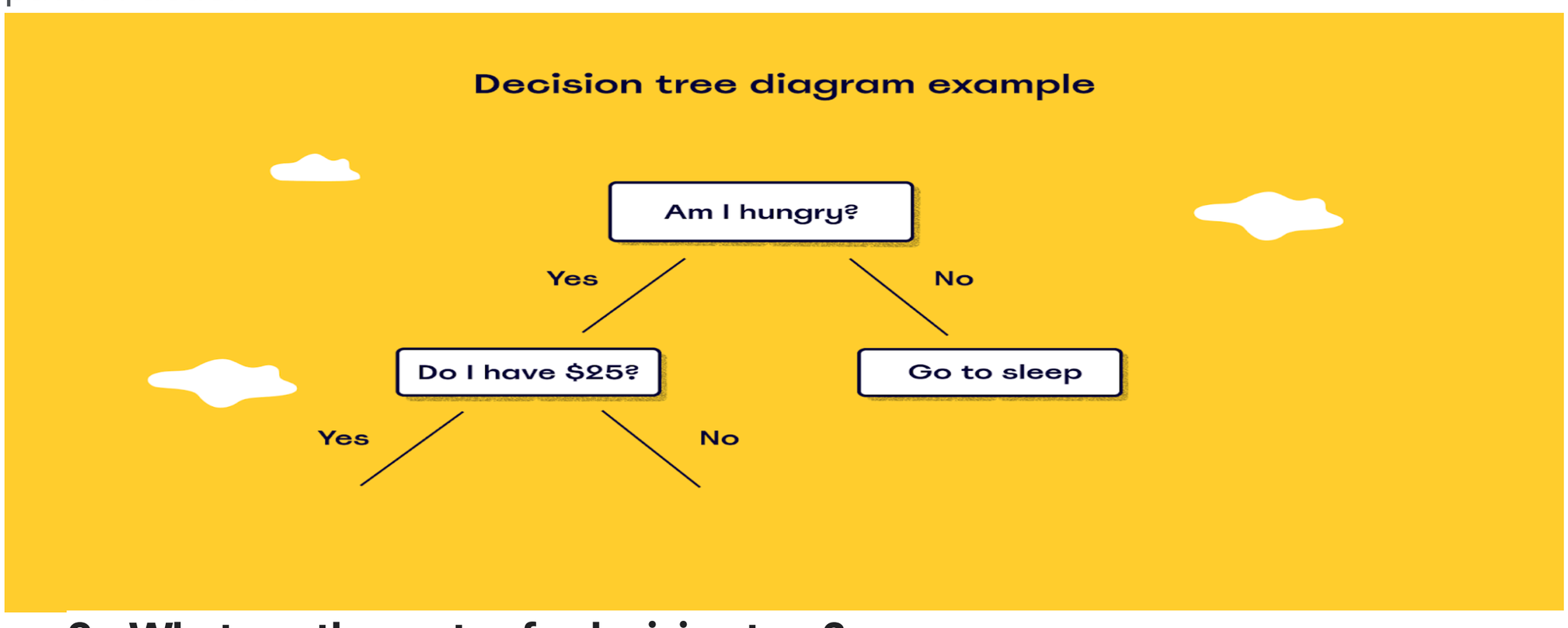


6. **What is the difference between decision tree and classification?**

In the context of decision trees, classification refers to using a decision tree model to assign class labels to instances based on their feature values. The decision tree algorithm uses the features of an instance to traverse the tree from the root to a leaf node, following the decision rules at each internal node.

7. **What is decision tree diagram?**

What is a decision tree diagram? A decision tree diagram is a type of flowchart that simplifies the decision-making process by breaking down the different paths of action available. Decision trees also showcase the potential outcomes involved with each path of action.



8. **What are the parts of a decision tree?**

Decision trees have three main parts: a root node, leaf nodes and branches. The root node is the starting point of the tree, and both root and leaf nodes contain questions or criteria to be answered. Branches are arrows connecting nodes, showing the flow from question to answer

WEEK 6:

Write a python to apply Random Forest Algorithm on a data set.

 # Import necessary libraries

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score, classification_report
```

 # Load the Iris dataset (replace it with your own dataset)

```
iris = load_iris()
X = iris.data
y = iris.target
```

 # Split the dataset into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

 # Create a Random Forest classifier

```
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=4
2)
```

 # Train the model on the training set

```
random_forest_model.fit(X_train, y_train)
```

 # Make predictions on the test set

```
predictions = random_forest_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}")

# Display additional metrics
print("Classification Report:")
print(classification_report(y_test, predictions))
```

OutPut

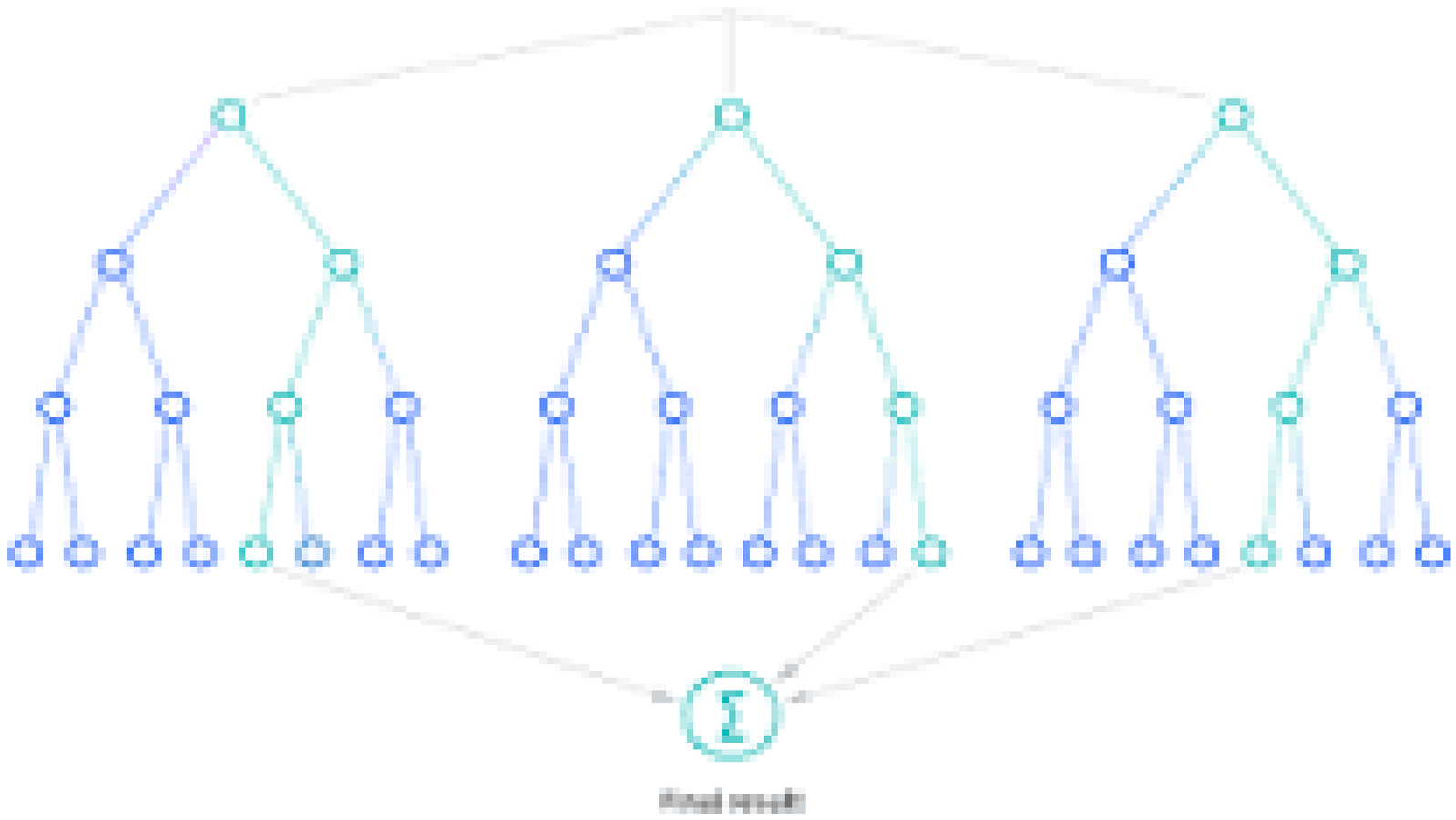
```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Sample viva Questions:

1. **What is random forest dataset?**

Random forest is a commonly-used machine learning algorithm, trademarked by Leo Breiman and Adele Cutler, that combines the output of multiple decision trees to reach

a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.



2. **What data is suitable for random forest?**

It can handle binary, continuous, and categorical data. Overall, random forest is a fast, simple, flexible, and robust model with some limitations. Random forest algorithm is an ensemble learning technique combining numerous classifiers to enhance a model's performance.

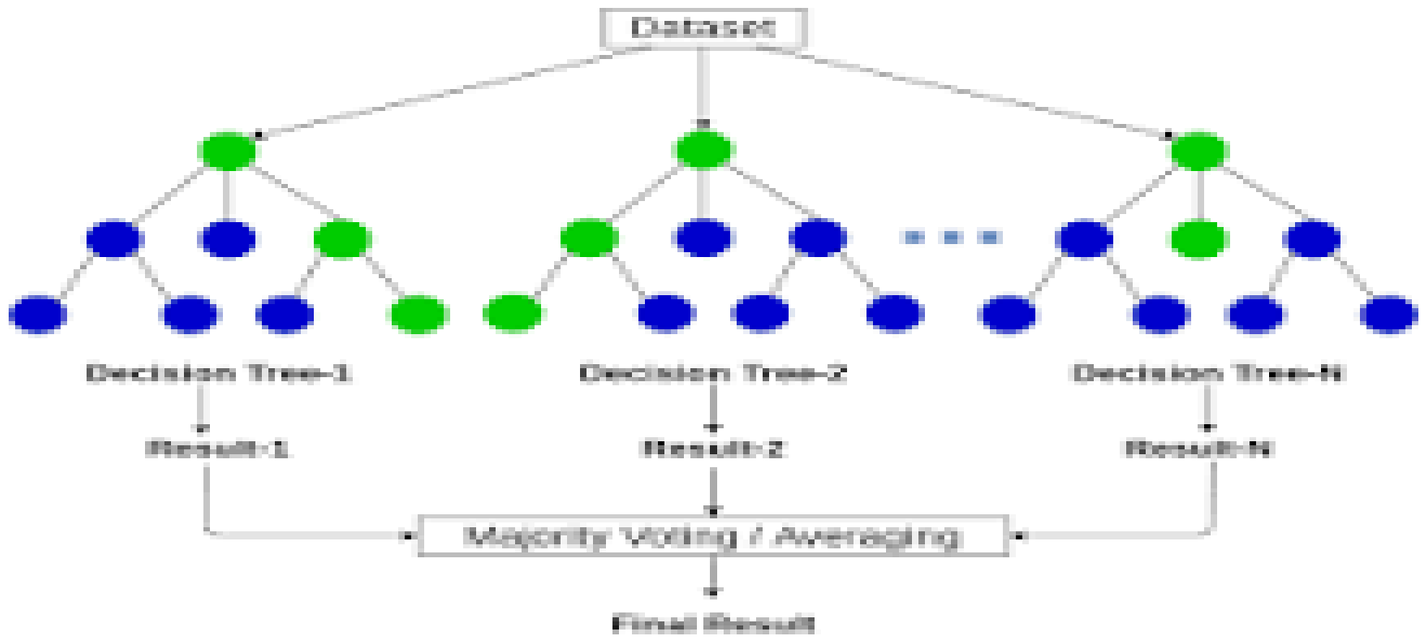
3. **How to train data using random forest?**

Working of Random Forest Algorithm

Step 1: Select random samples from a given data or training set.
Step 2: This algorithm will construct a decision tree for every training data.
Step 3: Voting will take place by averaging the decision tree.

4. **Which algorithm is better than random forest?**

Decision trees are much easier to interpret and understand. We take multiple decision trees in a random forest and then aggregate the result. Since a random forest combines multiple decision trees, it becomes more difficult to interpret.



5. What are the limitations of random forest?

Random forest can be computationally expensive, particularly when working with large datasets. It requires a lot of memory, which can be a constraint when working with limited resources. Although random forest is resistant to overfitting, it can still occur in certain cases, particularly when working with noisy data.

6. Why is it called random forest?

It is called a Random Forest because we use Random subsets of data and features and we end up building a Forest of decision trees (many trees). Random Forest is also a classic example of a bagging approach as we use different subsets of data in each model to make predictions

7. Can random forest do regression?

Random Forest is an ensemble machine learning technique capable of performing both regression and classification tasks using multiple decision trees and a statistical technique called bagging

---

WEEK 7:

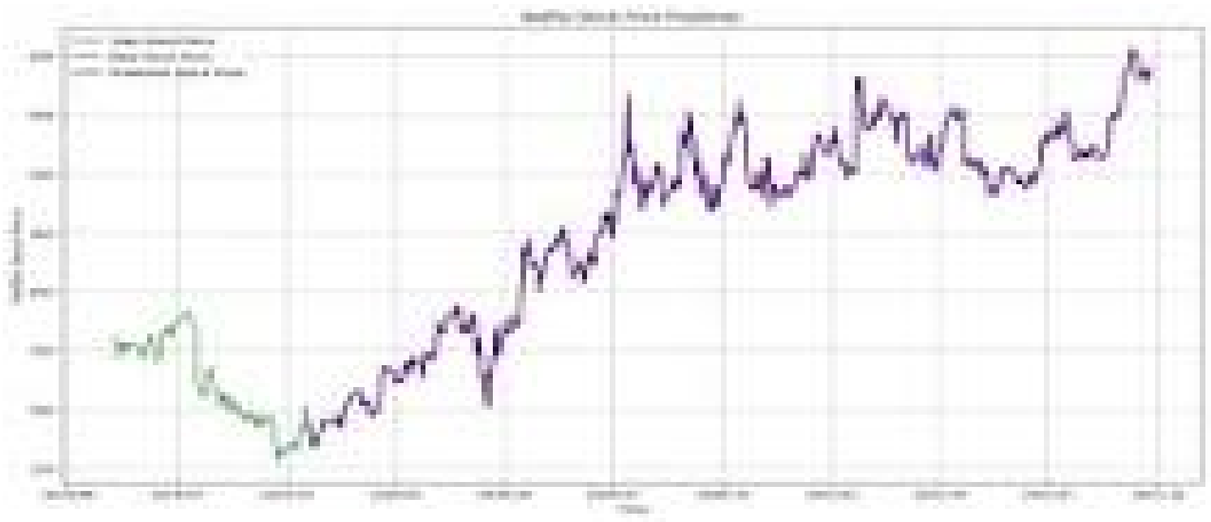
Write a python to apply ARIMA on Time series Data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Function to check stationarity using Dickey-Fuller test

def check_stationarity(data):
    result = adfuller(data)
    print('ADF Statistic:', result[0])
    print('p-value:', result[1])
    print('Critical Values:', result[4])

# Function to plot ACF and PACF

def plot_acf_pacf(data):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
    plot_acf(data, ax=ax1, lags=20)
    plot_pacf(data, ax=ax2, lags=20)
    plt.show()

# Load your time series data (replace 'your_data.csv' with your actual file)

data = pd.read_csv('laptop.csv', parse_dates=['date_column'], index_col='date_colum
n')
time_series = data['target_column']

# Check stationarity
check_stationarity(time_series)

# If the time series is non-stationary, apply differencing
differenced_series = time_series.diff().dropna()

# Check stationarity after differencing
check_stationarity(differenced_series)

# Plot ACF and PACF after differencing
plot_acf_pacf(differenced_series)
```

```
# Determine the order (p, d, q) for ARIMA
# Replace p, d, q values based on ACF and PACF plots
p = 1  # replace with the appropriate lag value from PACF
d = 1  # replace with the number of differences needed to make the series stationary
q = 1  # replace with the appropriate lag value from ACF

# Build ARIMA model
arima_model = ARIMA(time_series, order=(p, d, q))
arima_result = arima_model.fit()

# Forecast future values
forecast_steps = 10  # replace with the number of future steps you want to forecast
forecast = arima_result.get_forecast(steps=forecast_steps)

# Plot original time series and forecasted values
plt.plot(time_series.index, time_series, label='Original Time Series')
plt.plot(forecast.index, forecast.predicted_mean, label='Forecasted Values', color='red
')
plt.legend()
plt.show()
```

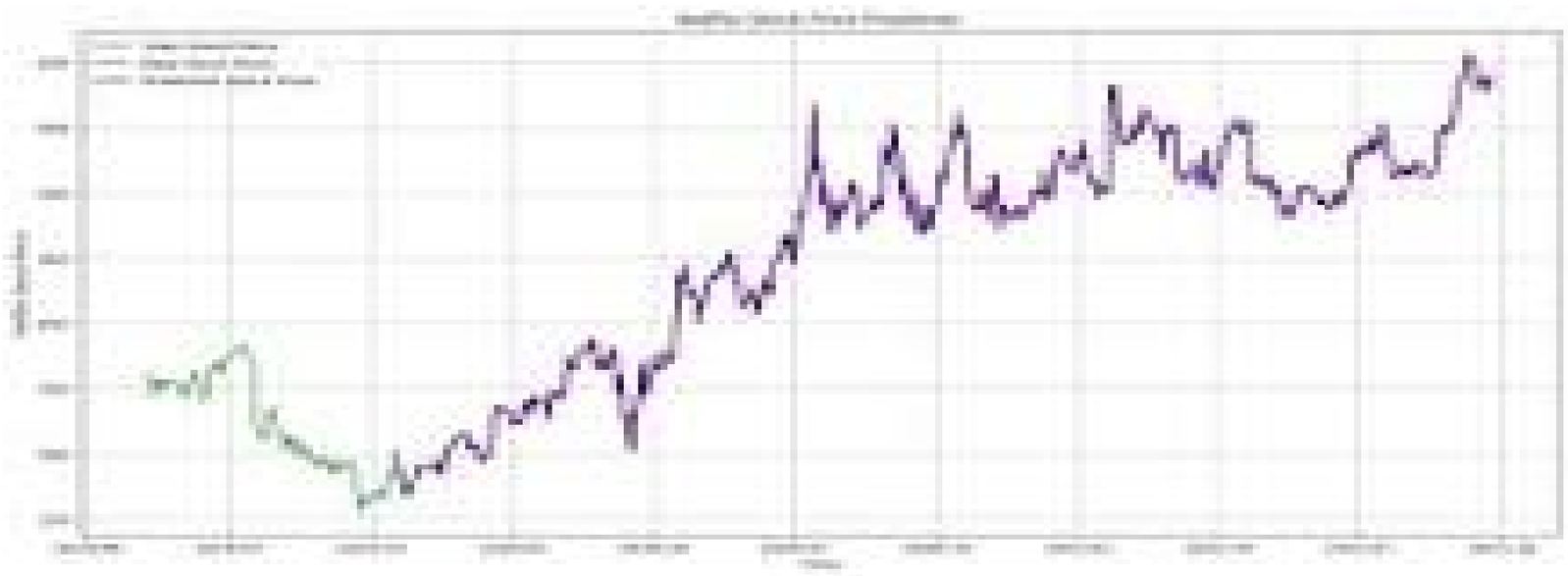
output:



Sample viva Questions:

1. **What is time series analysis in Python ARIMA?**

ARIMA models are a popular tool for time series forecasting, and can be implemented in Python using the `statsmodels` library. Time series analysis is widely used for forecasting and predicting future points in a time series.



2. How to use ARIMA to forecast in Python?

1. Import necessary packages.
2. Prepare time series data.
3. Determine the order of differencing (d), AR (p), and MA (q) terms using ACF and PACF plots.
4. Fit ARIMA model using ARIMA(). ...
5. Generate forecasts using forecast() method.
6. Visualize predictions and evaluate model performance.

3. What is ARIMA used for?

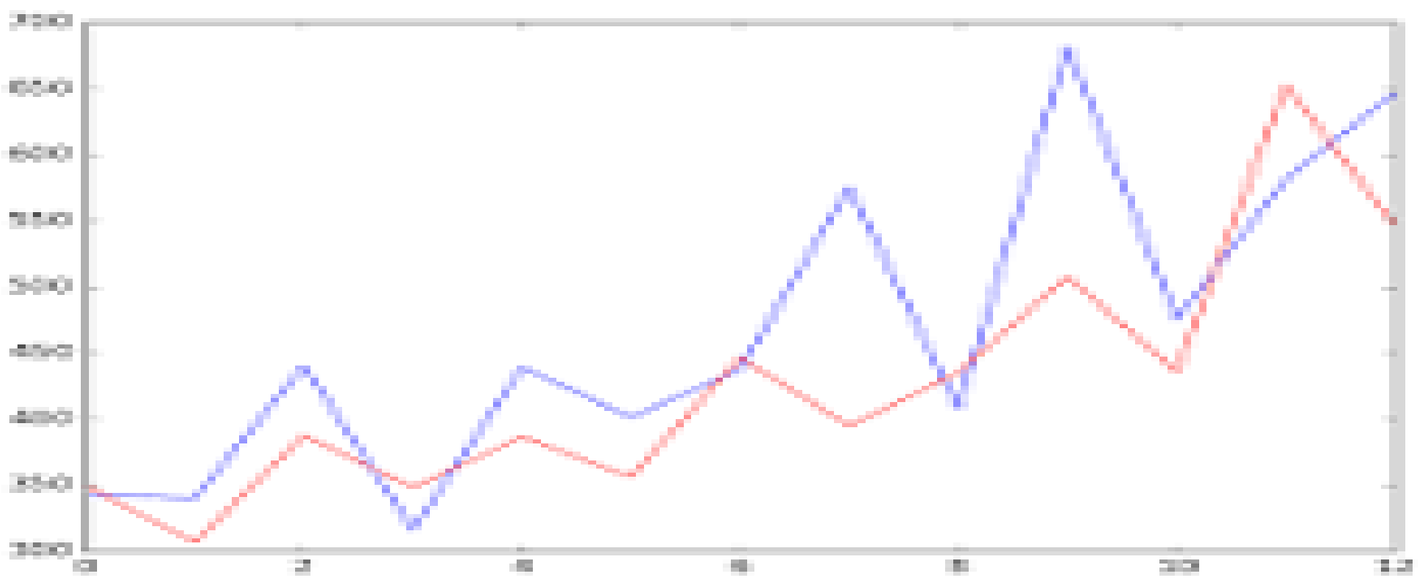
ARIMA is an acronym for "autoregressive integrated moving average." It's a model used in statistics and econometrics to measure events that happen over a period of time. The model is used to understand past data or predict future data in a series

4. Why use ARIMA for time series forecasting?

ARIMA models are a popular and powerful tool for forecasting time series data, such as sales, prices, or weather. ARIMA stands for AutoRegressive Integrated Moving Average, and it captures the patterns, trends, and seasonality of the data using a combination of past values, differences, and errors.

5. What is ARIMA model in Python summary?

The ARIMA (AutoRegressive Integrated Moving Average) model stands as a statistical powerhouse for analyzing and forecasting time series data. It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.



6. What are the advantages of ARIMA?

The advantages of the ARIMA load forecasting model are its easiness, robustness, high efficiency, and practical execution. No disadvantages are mentioned in the abstract. The advantages of the ARIMA load forecasting model include its ability to capture linear relationships and its simplicity

7. How many observations are required for ARIMA model?

The Box and Jenkins method typically recommends a minimum of 50 observations for an ARIMA model. This is recommended to cover seasonal variations and effects

WEEK 8:

Write a python for Object segmentation using hierarchical based methods.

```
import cv2
import numpy as np

def hierarchical_segmentation(image_path, threshold):

    # Read the image

    image = cv2.imread(image_path)

    # Convert the image to grayscale

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to reduce noise

    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Apply hierarchical segmentation using watershed algorithm

    _, markers = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    markers = cv2.connectedComponents(markers)[1]

    # Apply watershed algorithm for segmentation

    segmented = cv2.watershed(image, markers)

    # Create a mask based on the segmentation

    mask = np.zeros_like(gray, dtype=np.uint8)
```

```
mask[segmented == -1] = 255  # Mark watershed boundaries

# Apply thresholding to the mask
_, thresholded_mask = cv2.threshold(mask, threshold, 255, cv2.THRESH_BINARY)

# Bitwise AND to extract the segmented objects

segmented_objects = cv2.bitwise_and(image, image, mask=thresholded_mask)

    return segmented_objects

def main():
    image_path = (r"C:\Users\hi\Downloads\nature-trees-waterfall-water-green-river-4
k-wallpaper-1024x683.jpg")
    print(image_path)
    threshold_value = 100  # Adjust this threshold value according to your needs

    segmented_objects = hierarchical_segmentation(image_path, threshold_value)

    # Display the original and segmented images

    cv2.imshow('Original Image', cv2.imread(image_path))
    cv2.imshow('Segmented Objects', segmented_objects)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

if __name__ == "__main__":
        main()
```

Sample viva Questions:

1. **What is object Segmentation?**

Object segmentation is the process of splitting up an object into a collection of smaller f ixed-size objects to optimize storage and resources usage for large objects.

2. **What is hierarchical segmentation?**

A hierarchical image segmentation is a set of several image segmentations at different levels of segmentation detail in which the segmentations at coarser levels of detail can

be produced from simple merges of regions from segmentations at finer levels of detail.

3. What are the two types of hierarchical clustering methods?

There are two main types of hierarchical clustering:

- Agglomerative: Initially, each object is considered to be its own cluster. According to a particular procedure, the clusters are then merged step by step until a single cluster remains. ...
- Divisive: The Divisive method is the opposite of the Agglomerative method.

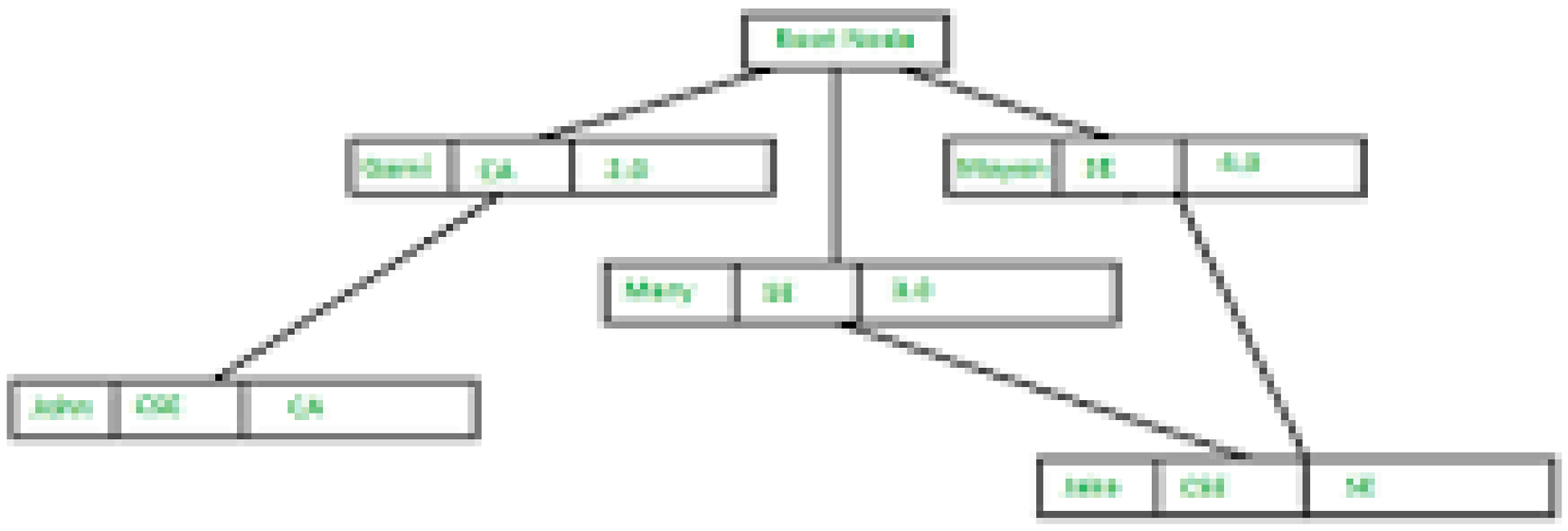
4. What is hierarchical based method for cluster analysis?

Hierarchical clustering starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps: (1) identify the two clusters that are closest together, and (2) merge the two most similar clusters. This iterative process continues until all the clusters are merged together.

5. Where is hierarchical model used?

Applications of hierarchical model :

Hierarchical models are also commonly used as physical models because of the inherent hierarchical structure of the disk storage system like tracks, cylinders, etc. There are various examples such as Information Management System (IMS) by IBM, NOMAD by NCSS, etc.



6. What is application of hierarchical model?

Hierarchical models are highly flexible, accommodating a wide range of data types, such as continuous, categorical, and ordinal data. This flexibility allows researchers to use hierarchical models in various applications, such as clustering, classification, and regression.

7. **What is the difference between object detection and object segmentation?**

Segmentation provides fine-grained information about object boundaries and regions, while detection focuses on identifying specific objects and their locations

---

WEEK 9:

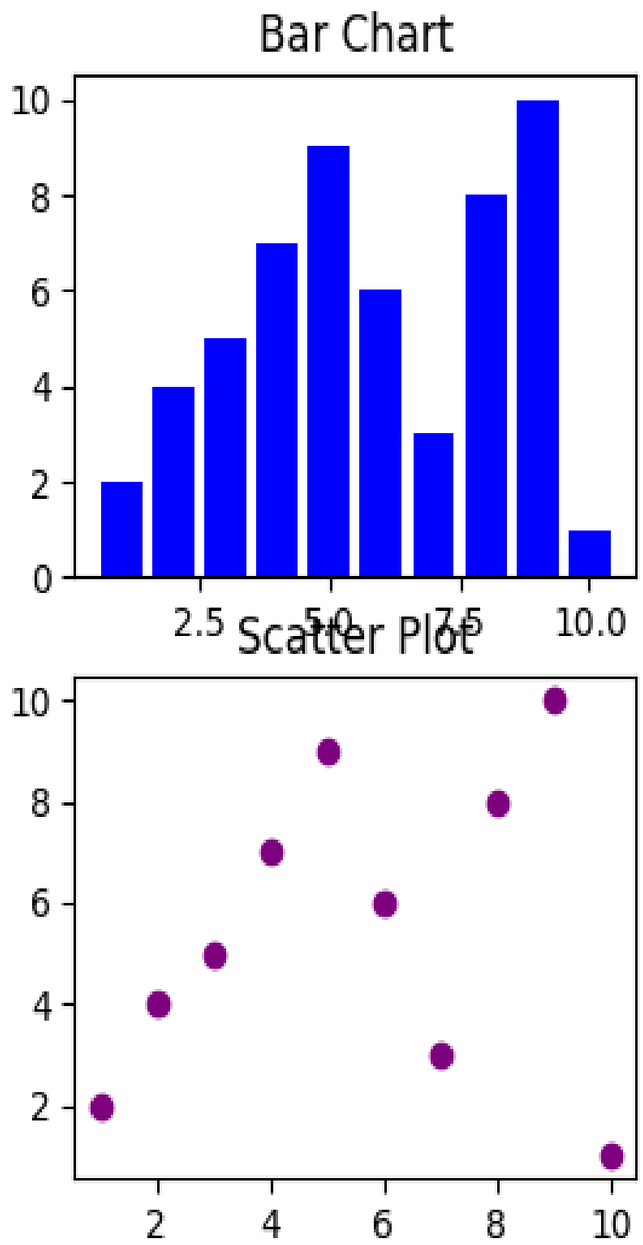
Write a python programming to perform Visualization techniques (types of Maps-Bar, colom, line, Scatter,3D Cubes etc.)

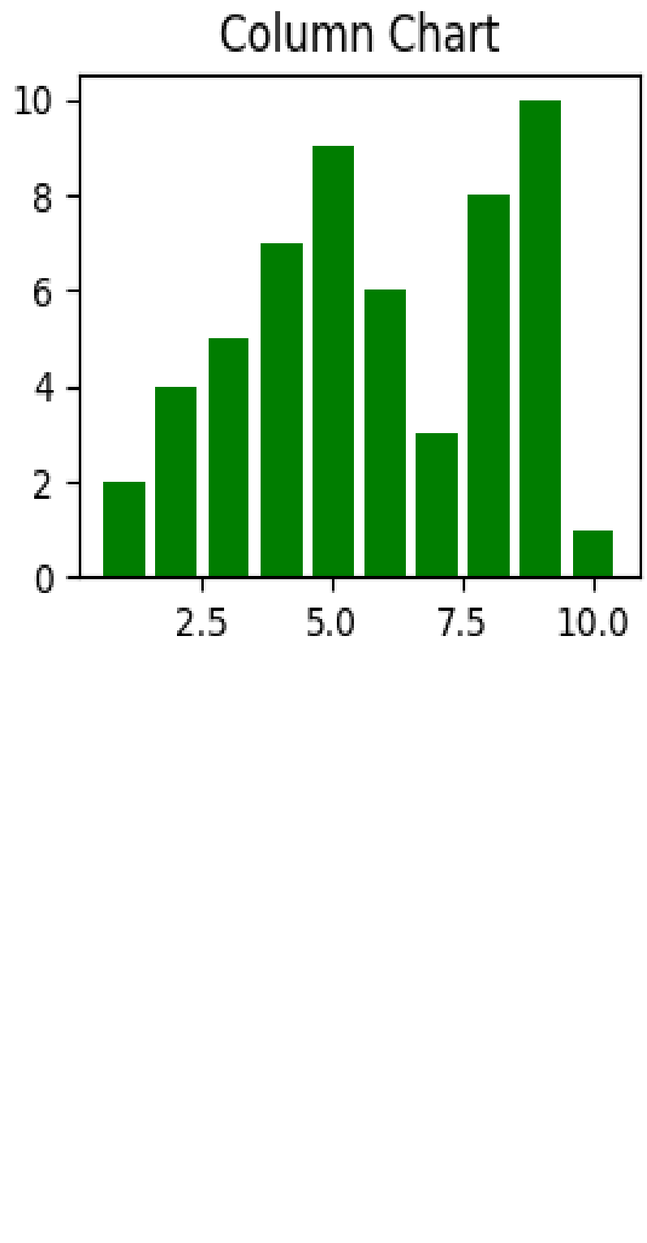
```
import matplotlib.pyplot as plt
import numpy as np
```

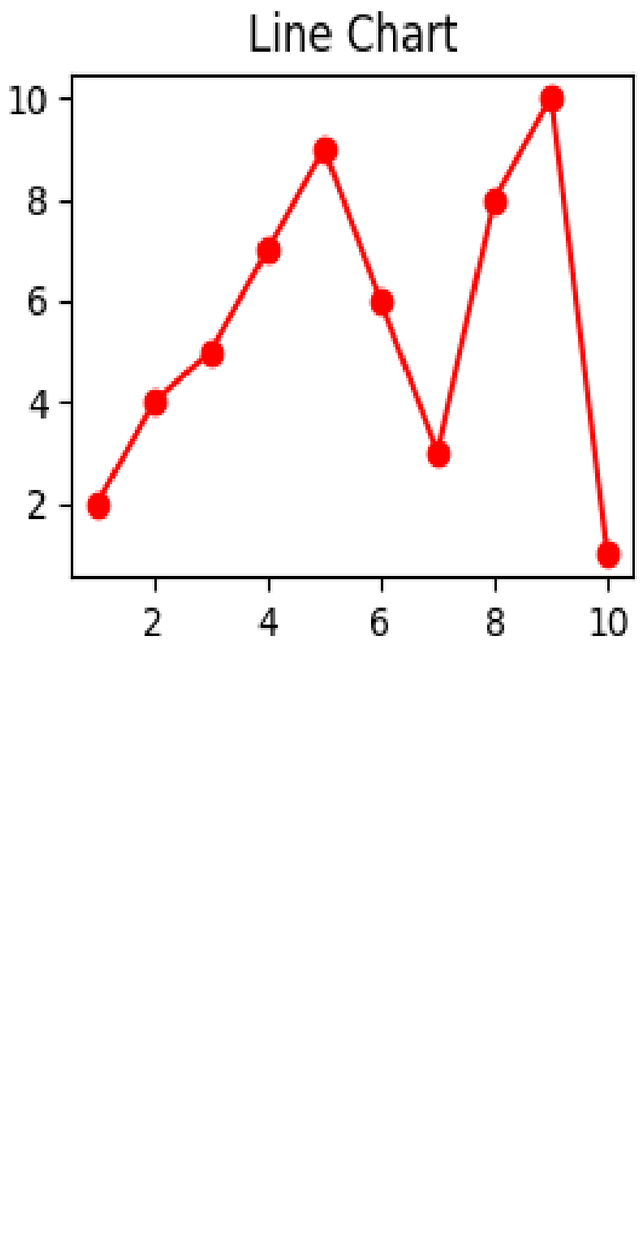
```
from mpl_toolkits.mplot3d import Axes3D

# Sample data for demonstration

x = np.arange(1, 11)
y = np.array([2, 4, 5, 7, 9, 6, 3, 8, 10, 1])
z = np.array([8, 6, 4, 2, 7, 1, 3, 9, 5, 10])

# Bar chart

plt.figure(figsize=(10, 5))
plt.subplot(2, 3, 1)
plt.bar(x, y, color='blue')
plt.title('Bar Chart')

# Column chart

plt.subplot(2, 3, 2)
plt.bar(x, y, color='green')
plt.title('Column Chart')

# Line chart

plt.subplot(2, 3, 3)
plt.plot(x, y, marker='o', linestyle='-', color='red')
plt.title('Line Chart')

# Scatter plot

plt.subplot(2, 3, 4)
plt.scatter(x, y, color='purple')
plt.title('Scatter Plot')

# 3D Cube plot

fig = plt.figure(figsize=(10, 5))
ax = fig.add_subplot(2, 3, 5, projection='3d')
ax.bar3d(x, y, np.zeros_like(x), 0.8, 0.8, z, shade=True, color='orange')
ax.set_title('3D Cube Plot')

# Display the plots

plt.tight_layout()
```

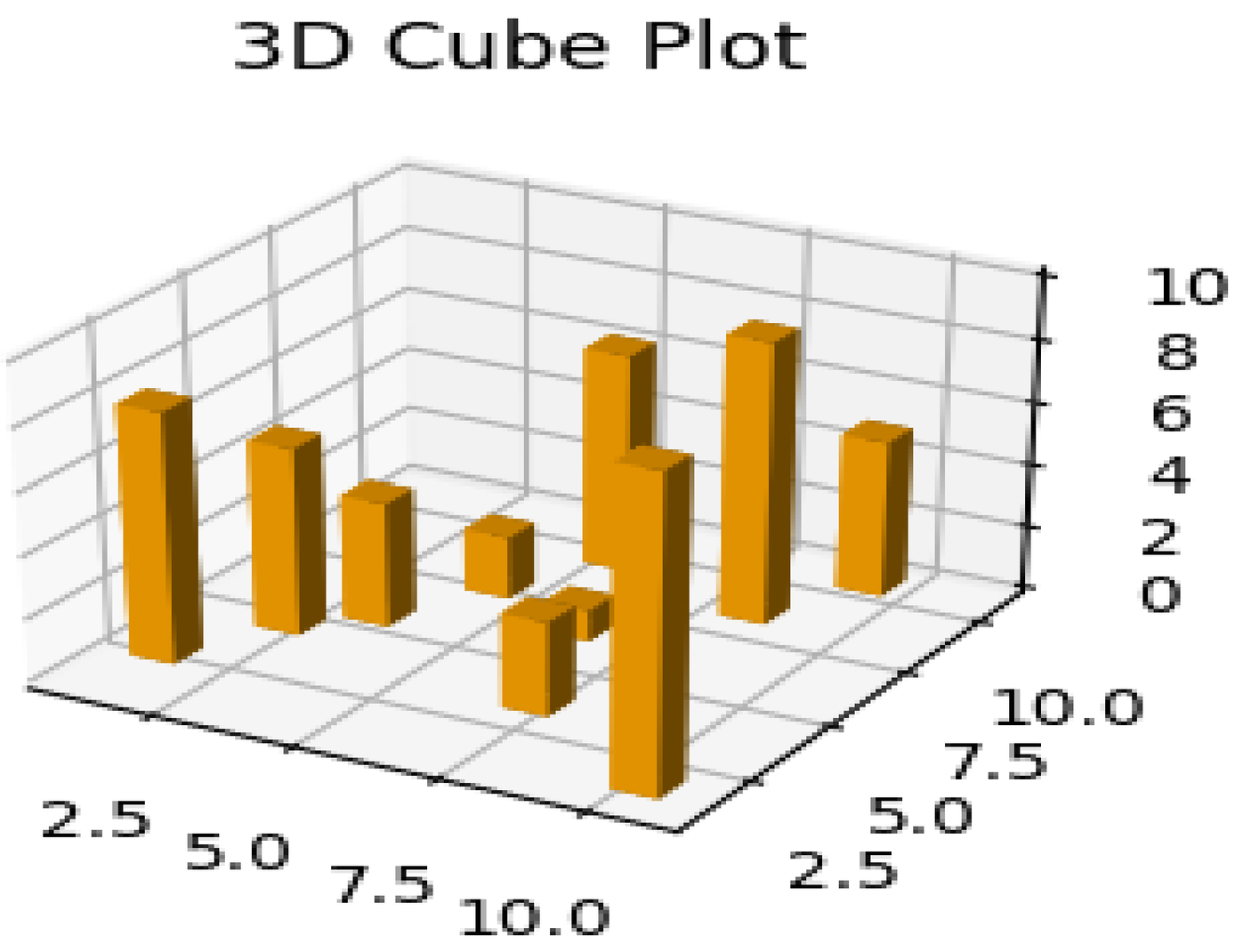
plt.show()

output









Sample viva Questions:

1. **What are the 4 main visualization types?**

Most Common Types of Data Visualization

1. Column Chart. They are a straightforward, time-tested method of comparing several collections of data. ...
2. Line Graph. A line graph is used to show trends, development, or changes through time.
3. Pie Chart.
4. Bar Chart.
5. Heat Maps.
6. Scatter Plot.
7. Bubble Chart.
8. Funnel Chart.

2. **What is visualization technique?**

Visualization is a simple technique that you can use to create a strong mental image of a future event. With good use of visualization, you can practice in advance for the event, so that you can prepare properly for it. And by visualizing success, you can build the self-confidence you need to perform well.

3. **What is visualization in data analytics?**

Data visualization is the process of using visual elements like charts, graphs, or maps to represent data. It translates complex, high-volume, or numerical data into a visual representation that is easier to process

4. **What are the three classes of data visualization techniques?**

The three most common categories of data visualization are graphs, charts, and maps. By choosing the right type of visualization for your data, you can reveal insights, tell a story, and guide decision-making.

5. **What are the five data visualization techniques?**

There are several common techniques used for data visualization: charts (bar, line, pie, etc.), plots (scatter, bubble, box, etc.), maps (heatmaps, dot distribution maps, cartograms, etc.), diagrams and matrices.

6. **Why is data visualization?**

Data visualization allows business users to gain insight into their vast amounts of data. It benefits them to recognize new patterns and errors in the data. Making sense of these patterns helps the users pay attention to areas that indicate red flags or progress

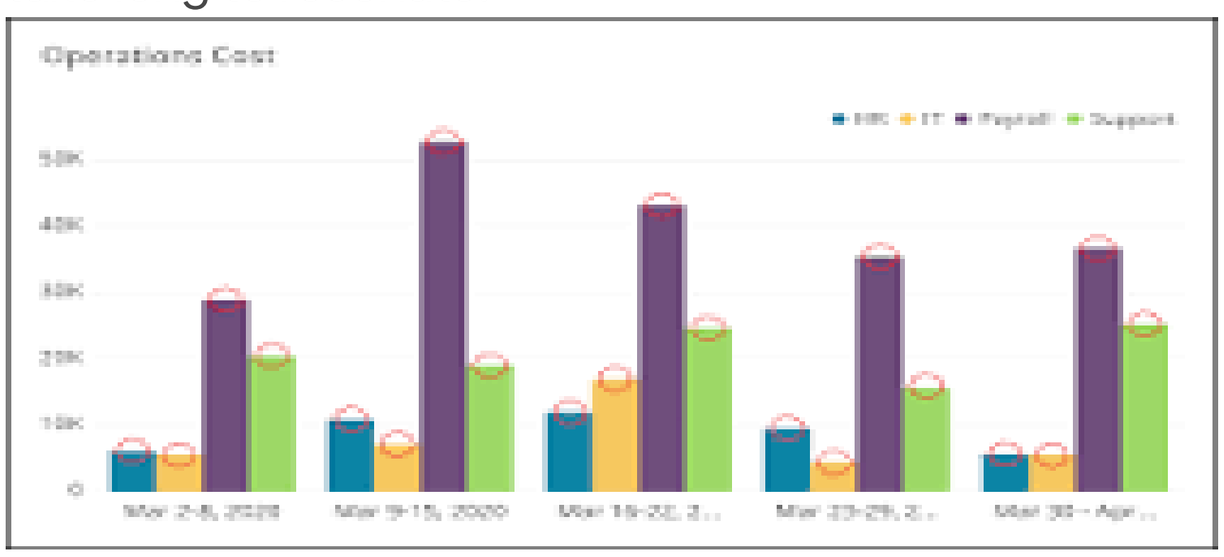
7. **What are the benefits of data visualization?**

Here are its biggest benefits of data visualization:

- Simplifies complex data.
- Reveals patterns and trends.
- Aids in decision making.
- Improves retention and engagement.
- Increases accessibility.
- Real-time monitoring.
- Identify areas that need attention or improvement.
- Predictive analysis.

8. **What are 4 characteristics of data visualization?**

Accurate: The visualization should accurately represent the data and its trends. Clear: Your visualization should be easy to understand. Empowering: The reader should know what action to take after viewing your visualization. Succinct: Your message shouldn't take long to resonate.



9. **What is data visualization and its applications?**

Data visualization is the representation of information and data using charts, graphs, maps, and other visual tools. These visualizations allow us to easily understand any patterns, trends, or outliers in a data set

10. **What is the best data Visualisation tool?**

The Best Data Visualization Software of 2024.

- Microsoft Power BI.
- Tableau.
- Qlik Sense.
- Klipfolio.
- Looker.
- Zoho Analytics.
- Domo.

WEEK 10:

Write a python programming to Descriptive analytics on healthcare data.

```
import pandas as pd

# Create a sample healthcare dataset (replace this with your actual data)

data = {
    'Patient_ID': [1, 2, 3, 4, 5],
    'Age': [25, 30, 45, 60, 35],
    'Gender': ['Male', 'Female', 'Male', 'Female', 'Male'],
    'Disease': ['Flu', 'Diabetes', 'Hypertension', 'Cancer', 'Asthma'],
    'Treatment_Cost': [1000, 5000, 3000, 25000, 800],
    'Treatment_Outcome': ['Recovered', 'Stable', 'Improved', 'In Remission', 'Stable']
}

df = pd.DataFrame(data)

def descriptive_analytics(df):
    # Summary statistics for numerical columns
    summary_stats = df.describe()

    # Count of unique values in categorical columns
```

```
unique_counts = df.nunique()

# Distribution of diseases
disease_distribution = df['Disease'].value_counts()

# Average treatment cost by gender
avg_cost_by_gender = df.groupby('Gender')['Treatment_Cost'].mean()

# Print the results

print("Summary Statistics:")
print(summary_stats)
print("\nUnique Counts:")
print(unique_counts)
print("\nDisease Distribution:")
print(disease_distribution)
print("\nAverage Treatment Cost by Gender:")
print(avg_cost_by_gender)
```

 # Run the descriptive analytics

```
descriptive_analytics(df)
```

output:

```
Summary Statistics:
       Patient_ID       Age  Treatment_Cost
count    5.000000   5.000000        5.000000
mean     3.000000  39.000000     6960.000000
std      1.581139  13.874437    10227.805239
min      1.000000  25.000000      800.000000
25%      2.000000  30.000000     1000.000000
50%      3.000000  35.000000     3000.000000
75%      4.000000  45.000000     5000.000000
max      5.000000  60.000000    25000.000000

Unique Counts:
```

```
Patient_ID        5
Age               5
Gender            2
Disease           5
Treatment_Cost    5
Treatment_Outcome 4
dtype: int64
```

```
Disease Distribution:
Flu           1
Diabetes      1
Hypertension  1
Cancer        1
Asthma        1
Name: Disease, dtype: int64
```

```
Average Treatment Cost by Gender:
Gender
Female   15000.0
Male      1600.0
Name: Treatment_Cost, dtype: float64
```

Sample viva Questions:

1. **What is descriptive analysis in healthcare?**

Descriptive analytics allows us to understand what happened historically and answer questions such as: How many patients were hospitalized last week? What percent of patients dropped home therapy in the last month? What are the average bone mineral metabolism (BMM) laboratory values for the patient population?

2. What are the four types of data analytics of healthcare?

One can employ healthcare data analytics to study past, current, and future patterns through one of the following techniques:

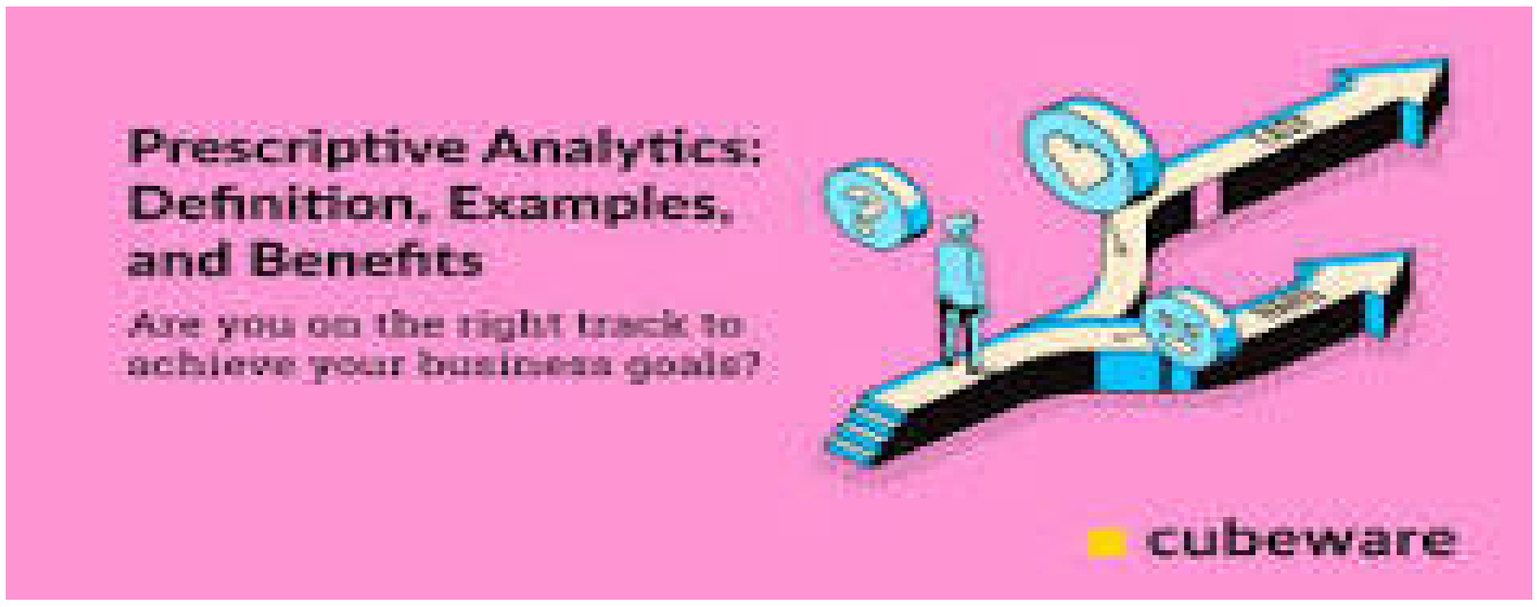
- Descriptive analytics
- Predictive analytics
- Prescriptive analytics
- Discovery analytics

3. How is descriptive research used in healthcare?

Descriptive studies provide knowledge about which populations or subgroups are most or least affected by disease. This enables public health administrators to target particular segments of the population for education or prevention programmes and can help allocate resources more efficiently.

4. What is an example of prescriptive analytics in healthcare?

Healthcare : For example, if a hospital's goal is to reduce their patients' wait time from 30 minutes to 20 minutes, a Prescriptive Analytics model is used to map out the possible courses of action, whereby each action is represented as an adjustable variable.



5. **What is the difference between prescriptive analytics and descriptive analytics?**

There are three types of analytics that businesses use to drive their decision making; descriptive analytics, which tell us what has already happened; predictive analytics, which show us what could happen, and finally, prescriptive analytics, which inform us what should happen in the future.

6. **What is descriptive analytics and exploratory data analytics?**

In contrast to descriptive data analysis, which is a numerical approach to data analysis, exploratory data analysis is a visual approach to data analysis. We will turn to exploratory data analysis once we have a basic comprehension of the data at hand through descriptive analysis

7. **What are the techniques of descriptive data analysis?**

Descriptive techniques often include constructing tables of means and quantiles, measures of dispersion such as variance or standard deviation, and cross-tabulations or "crosstabs" that can be used to examine many disparate hypotheses. Those hypotheses are often about observed differences across subgroups.

WEEK 11:

Write a python program to perform predictive analytics on Product Sales Data.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv(r'C:\Users\hi\Downloads\product.csv')

# Extract features (X) and target variable (y)
X = data[['Price', 'Quantity']]
y = data['Sales']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

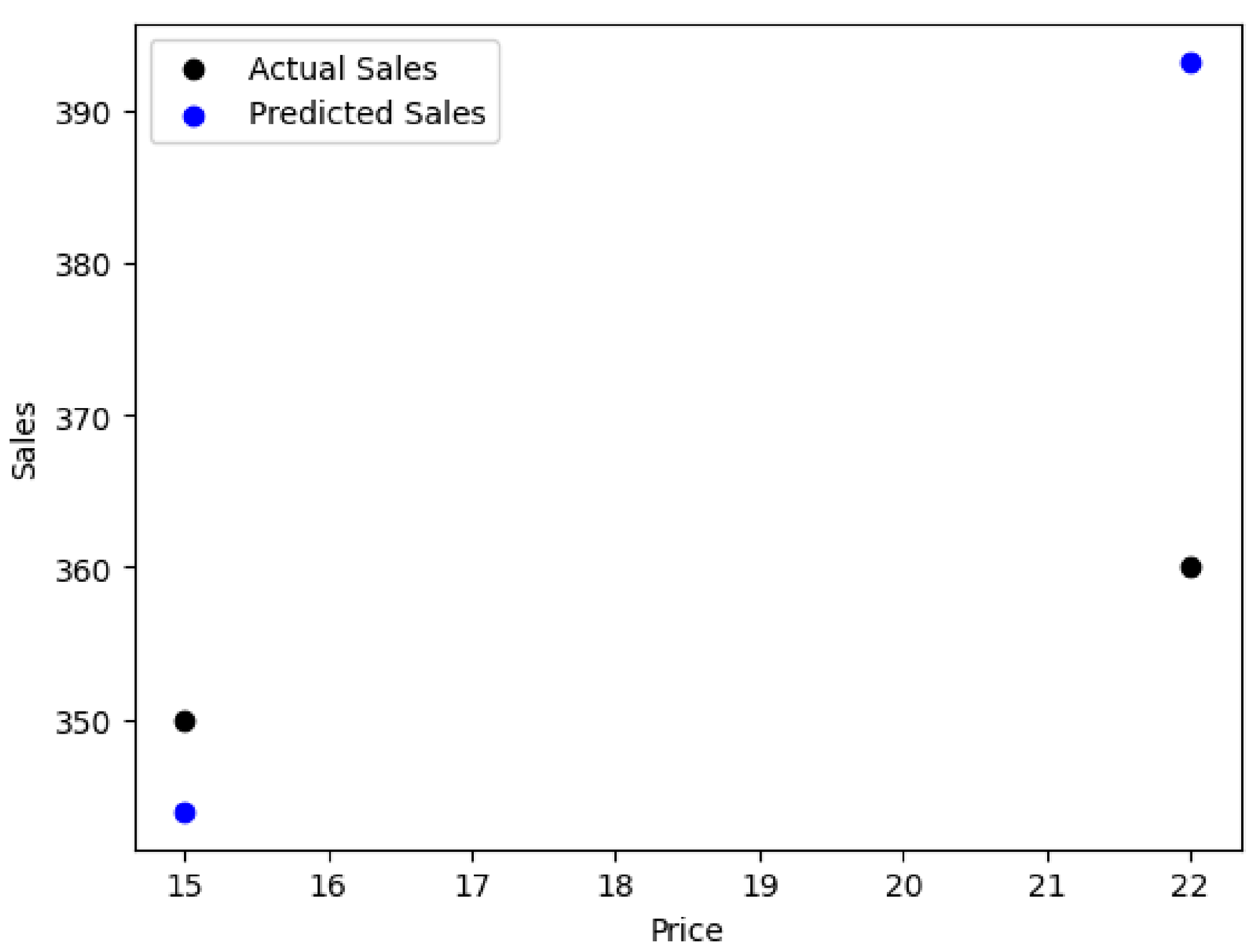
```
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

 # Visualize the predictions

```
plt.scatter(X_test['Price'], y_test, color='black', label='Actual Sales')
plt.scatter(X_test['Price'], y_pred, color='blue', label='Predicted Sales')
plt.xlabel('Price')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

output

Mean Squared Error: 570.0351158816655
R-squared: -21.80140463526662



Sample viva Questions:

1. **Which type of data is used for predictive analytics?**

Predictive analytics uses historical data to predict future events. Typically, historical data is used to build a mathematical model that captures important trends. That predictive model is then used on current data to predict what will happen next, or to suggest actions to take for optimal outcomes.

2. **What is an example of predictive analytics data?**

Predictive analytics models may be able to identify correlations between sensor readings. For example, if the temperature reading on a machine correlates to the length of time it runs on high power, those two combined readings may put the machine at risk of downtime. Predict future state using sensor values

3. **What is predictive analytics on sales data?**

Predictive analytics in sales is the process of using data, various statistical models, algorithms, and machine learning techniques to identify the likelihood of future outcomes based on data you collected in the past.

4. **How is predictive analytics used in retail?**

Through predictive retail analytics, the retailers and company heads can use historic data to generate futuristic insights. They can predict potential sales in the next year, quarter, or the very next day, forecast trends, know expected industry activity, predict customer behavior, and much more.

5. **How does Netflix use predictive analytics?**

Netflix analyzes vast amounts of user data, including viewing history, ratings, and browsing behavior, to make predictions about users' preferences. By applying machine learning algorithms to this data, Netflix can suggest content tailored to individual users, improving user engagement and satisfaction

6. **How to do predictive sales analysis?**

By relating your current transactional and customer interaction data (emails, meetings, phone calls, etc.) to actual sales outcomes, you can predict future revenue to an extremely high degree of accuracy

7. What are the applications of predictive analytics?

Nowadays, it is being used in all major industries, such as healthcare, finance, insurance, retail, etc. Predictive analytics can help predict the future growth of any real-life entity with the help of advanced modern technologies such as machine learning, big data, statistical models, artificial intelligence, etc.

WEEK 12:

Write a python program to Apply predictive analytics for weather forecasting.

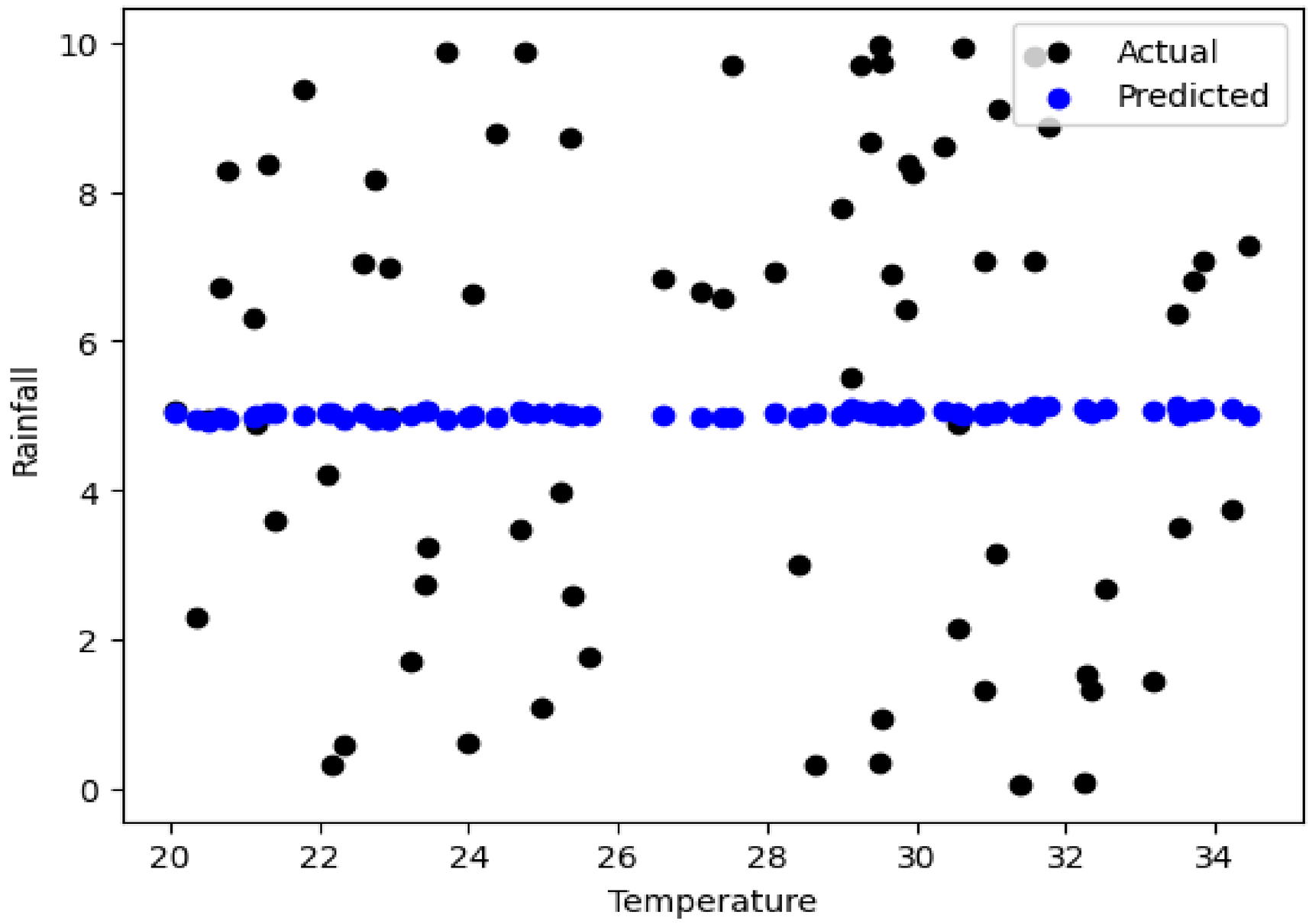
```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta


# Generate a sample dataset

np.random.seed(42)  # For reproducibility

dates = pd.date_range(start='2022-01-01', end='2022-12-31', freq='D')
temperature = np.random.uniform(low=20, high=35, size=len(dates))
humidity = np.random.uniform(low=30, high=80, size=len(dates))
```

```
wind_speed = np.random.uniform(low=5, high=25, size=len(dates))
rainfall = np.random.uniform(low=0, high=10, size=len(dates))

weather_data = pd.DataFrame({
    'Date': dates,
    'Temperature': temperature,
    'Humidity': humidity,
    'Wind_Speed': wind_speed,
    'Rainfall': rainfall
})

# Save the dataset to a CSV file

weather_data.to_csv('weather_dataset.csv', index=False)

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Load the dataset

weather_data = pd.read_csv('weather_dataset.csv')

# Feature selection (for example, using Temperature and Humidity)

features = weather_data[['Temperature', 'Humidity']]
target = weather_data['Rainfall']

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random
_state=42)

# Train a linear regression model
```

```
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set

predictions = model.predict(X_test)

# Evaluate the model

mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')

# Visualize the results (optional)

plt.scatter(X_test['Temperature'], y_test, color='black', label='Actual')
plt.scatter(X_test['Temperature'], predictions, color='blue', label='Predicted')
plt.xlabel('Temperature')
plt.ylabel('Rainfall')
plt.legend()
plt.show()
```

output

Mean Squared Error: 9.897200944942393



Sample viva Questions:

1. **How do you analyse weather forecast?**

Reading and Interpreting Forecast Steps:

1. Get Familiar with the Forecast Layout. ...
2. Understand Weather Symbols and Icons. ...
3. Study Temperature Ranges. ...
4. Analyse Precipitation Probability. ...
5. Consider Wind Speed and Direction. ...
6. Assess Humidity Levels. ...
7. Take Note of Atmospheric Pressure.

2. **What is weather analysis chart?**

The weather map, also known as a synoptic chart, is a simplified depiction of the Earth's surface weather patterns that shows the positions and motions of the various systems

3. What forecasting techniques are used to predict weather patterns?

The main ways we can forecast the weather include looking at current weather conditions, tracking the motion of air and clouds in the sky, finding previous weather patterns that resemble current ones, examining changes in air pressure, and running computer models.

4. What are three major widely used forecast models?

Four common types of forecasting models

a. Time series model.
b. Econometric model.
c. Judgmental forecasting model.
d. The Delphi method.

5. Which machine learning model is best for weather prediction?

Artificial intelligence (AI) firm Google DeepMind has turned its hand to the intensive science of weather forecasting — and developed a machine-learning model that outperforms the best conventional tools, as well as other AI approaches, at the task

6. What is time series analysis in weather forecasting?

Time series techniques have been used for decades to forecast weather patterns, and continue to be a valuable tool for meteorologists and researchers. From daily weather forecasts to long-term climate projections, the study of time series data is essential for understanding and predicting weather patterns.