

**Faculty of Digital Transformation Engineering**

**Department of Educational Technology and Engineering**

**Assignment no:  01**

**Course code : ICTE 4335**

**Assignment On :** KNN, K means clustering and Decision Tree Algorithm.

**Course Title : Introduction to Data Science and Learning Analytics**

|  |  |
| --- | --- |
| Submitted To  **Farhana Islam**  Assistant Professor  Department of  Education Technology and Engineering Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh | Submitted By  **MD Shohedozzaman Basunia [2002033]**  **Farhana Yesmin Ety [2002034]**  **Md. Rony Ahmmed Shah [2002035]**  Session : 2020-21  Department of  Education Technology and Engineering |

**Date of Submission: 20-12-2024**

**Assignment**

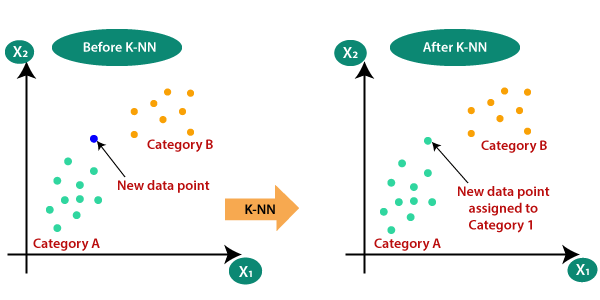
### **KNN, K means clustering and Decision Tree Algorithm**

**KNN Algorithm**

**1. Explain the working principle of the K-Nearest Neighbors (KNN) algorithm in classification tasks.**

**Answer:**

The KNN algorithm operates on the principle of similarity or “nearness,” **predicting the label or value of a new data point by considering the labels or v**alues of its K-nearest (the value of K is simply an integer) neighbors in the training dataset.Again the K-Nearest Neighbors (KNN) algorithm classifies new data points by comparing them to the existing labeled data points in the training set, assigning the new data point to the class that is most common among its "K" nearest neighbors, based on a distance metric like Euclidean distance; essentially, it predicts the class of a new data point by looking at the classes of its closest data points in the training set, assuming that similar data points should be in the same class.



**Training Phase:** KNN doesn’t have an explicit training phase. It simply stores the training dataset.

**Prediction Phase:**

* **Selecting the optimal value of K:**

K represents the number of nearest neighbors that needs to be considered while making prediction.

* **Calculating distance:**

To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

* **Finding Nearest Neighbors:**

The k data points with the smallest distances to the target point are the nearest neighbors.

* **Voting for Classification:**

In the classification problem, the class labels of K-nearest neighbors are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.

2. Discuss the impact of the following factors on the performance of KNN:

a. Choice of k (number of neighbors)

b. Distance metrics (e.g., Euclidean, Manhattan)

c. High dimensionality of data (curse of dimensionality)

**Answer :**

The performance of the K-Nearest Neighbors (KNN) algorithm is highly sensitive to several factors, including the choice of 𝑘, the **distance metric** used, and the dimensionality of the data. Here's how these factors impact KNN:

a. **Choice of k (Number of Neighbors)**

* **Small k**:A very small k (e.g.,k=1) makes the model highly sensitive to noise and outliers in the dataset because predictions are based on the nearest neighbor alone. This can lead to overfitting.
* **Large k**:A large k smooths out noise by considering more neighbors, but it may result in underfitting as the model averages over a broader group, potentially ignoring local patterns.
* **Optimal k**:The choice of k is often determined empirically through techniques like cross-validation. Ideally, k should balance bias and variance.

**b. Distance Metrics**

KNN relies on a distance metric to determine the "closeness" of points. Common distance metrics include Euclidean, Manhattan, Minkowski, and others.

* **Euclidean Distance:**
  + Measures the straight-line (L2 norm) distance between points.
  + Suitable for continuous and well-scaled data.
  + Sensitive to large differences in feature magnitudes, so normalization or standardization of features is critical.
* **Manhattan Distance:**
  + Measures the distaplnce along the axes (L1 norm).
  + More robust to outliers compared to Euclidean distance.
  + Preferred for high-dimensional sparse data or scenarios where grid-like paths are more meaningful.
* **Impact of the choice of metric:**
* The choice of metric directly influences how neighbors are defined and selected.
* For data with varying feature scales, choosing an inappropriate metric without normalization can result in misleading predictions.
* Some metrics may work better depending on the specific geometry of the data.
* **Tuning the metric:**

Experiment with different distance metrics and use validation metrics to assess performance.

**c. High Dimensionality of Data (Curse of Dimensionality)**

**Definition**:

In high-dimensional spaces, the distance between data points tends to become uniform, making it challenging for KNN to distinguish between "close" and "far" neighbors.

**Impact on KNN:**

* Distance Measures Become Less Meaningful: In high-dimensional spaces, the distance between points becomes less meaningful because all points tend to be equidistant from each other.
* Sparsity: Data points in high-dimensional spaces become sparse, making it difficult to find meaningful neighbors.
* Overfitting: With more dimensions, the model may fit the noise in the training data rather than the underlying patterns.

**Challenges**:

Increased computational cost due to the higher number of dimensions.

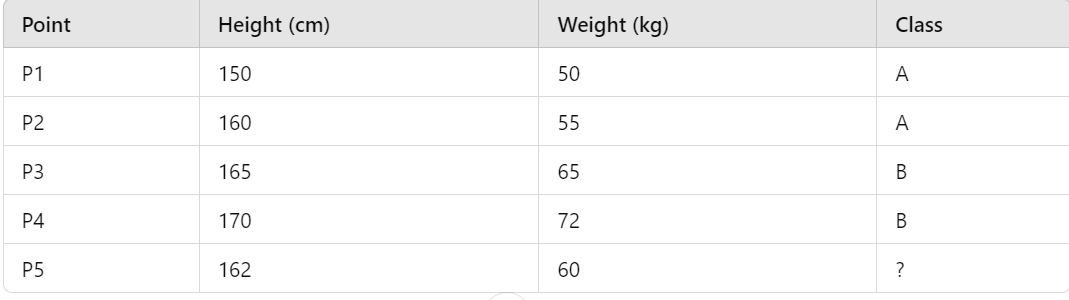
Potential overfitting since the algorithm becomes sensitive to noise in high-dimensional spaces.

**Mitigation strategies:**

* Dimensionality reduction: Techniques like Principal Component Analysis (PCA) or t-SNE can help reduce the number of dimensions while retaining important information.
* Feature selection: Use domain knowledge or feature importance scores to select only the most relevant features.
* Normalize or standardize the data to ensure features contribute equally to the distance calculations.

KNN is highly sensitive to these factors, and careful preprocessing and hyperparameter tuning are essential for achieving optimal performance.

**3.Consider the following dataset with two features (Height and Weight) and two classes (Class A and Class B):**



**(a) Using the Euclidean distance metric, classify point P5 with k = 3. Show your calculations.**

**(b) Explain how the classification would change if k=5.**

**Answer:**

(a)To classify point P5, we'll calculate the Euclidean distance between P5 and all other points (P1 to P4). The dataset contains height and weight features. Let’s proceed step by step:

**1.Euclidean Distance Formula:**

Distance=(𝑥2−𝑥1)2+(𝑦2−𝑦1)2

**2.Calculate distances from P5 to all points**

* P5 to P1:

Distance = sqrt((162-150)^2 + (60-50)^2) = sqrt(144 + 100) = sqrt(244) ≈ 15.62

* P5 to P2:

Distance = sqrt((162-160)^2 + (60-55)^2) = sqrt(4 + 25) = sqrt(29) ≈ 5.39

* P5 to P3:

Distance = sqrt((162-165)^2 + (60-65)^2) = sqrt(9 + 25) = sqrt(34) ≈ 5.83

* P5 to P4:

Distance = sqrt((162-170)^2 + (60-72)^2) = sqrt(64 + 144) = sqrt(208) ≈ 14.42

**3.Sort distances**

* P2: 5.39 (Class A)
* P3: 5.83 (Class B)
* P4: 14.42 (Class B)
* P1: 15.62 (Class A)

**4.Choose**: 𝑘=3 Nearest neighbors:

P2 (A), P3 (B), P4 (B)

**5. Majority voting**

Class A: 1 neighbor

Class B: 2 neighbors

**6.Result**:

P5 is classified as Class B for k=3.

(b) Classifying Point P5 with k = 5

If we consider k = 5, we would include all four points (P1, P2, P3, P4) as neighbors. In this case, we would have 2 points in Class A (P1, P4) and 2 points in Class B (P2, P3). This would result in a tie.

To break the tie, different strategies can be used:

* Increase k: We could try increasing k to a larger odd number to see if a clear majority emerges.
* Use weighted voting: We could assign weights to neighbors based on their distance, giving more weight to closer neighbors.
* Use distance-based weighting: We could divide the vote of each neighbor by its distance to P5, giving more influence to closer neighbors.

4. You are given a dataset containing information about customer purchasing habits (e.g., age, income, and purchase frequency).

1. Describe the preprocessing steps required before applying the KNN algorithm.
2. Discuss the importance of feature scaling and its effect on the results of KNN.

**Answer**:

**1.Preprocessing Steps for KNN Algorithm:**

Before applying the K-Nearest Neighbors (KNN) algorithm to the dataset, the following preprocessing steps are necessary:

a. Handle Missing Data

* Reason: Missing values cadistort distance calculations.
* Methods:Replace missing values with the mean, median, or mode.
* Use advanced techniques like KNN imputation.

b. Encode Categorical Variables

* Reason: KNN works with numerical data,so categorical variables must be transformed into numerical values.
* Methods:

a.Use one-hot encoding for nominal variables.

b.Use label encoding for ordinal variables

c. Feature Scaling

* Reason: Features with different ranges (e.g., age in years vs. income in thousands) can disproportionately affect distance calculations.
* Methods:
  + Standardization: Rescales features to have a mean of 0 and a standard deviation of 1.

z = (x−μ)/σ

* + Normalization: Scales features to a range of [0, 1].

x ′ = (x−min(x))/(max(x)−min(x))

d. Remove Outliers

* Reason: Outliers can skew distance metrics and affect classification results.
* Methods:Use z-scores or interquartile range (IQR) to detect and handle outliers.

e. Feature Selection/Engineering

* Reason: Irrelevant or redundant features can reduce the model's accuracy.
* Method:Use domain knowledge or feature selection techniques like mutual information or recursive feature elimination (RFE).

f. Split the Dataset

* Reason: To evaluate model performance, split the data into training and testing sets.
* Methods:Common ratios are 80:20 or 70:30 for training and testing.

**2.Importance of Feature Scaling in KNN:**

* Equal Contribution of Features: Without scaling, features with larger ranges can dominate the distance calculations, leading to biased results. Scaling ensures that all features contribute equally to the distance calculation.
* Improved Accuracy: Feature scaling can significantly improve the accuracy of KNN, especially when features have different scales.
* Efficient Distance Calculations: Scaling can speed up distance calculations, as it reduces the magnitude of differences between feature values.

**Example**:

Consider a dataset with two features: age (range: 18-80) and income (range: 1000-100000). Without scaling, a small difference in income would have a much larger impact on the distance calculation than a large difference in age. This can lead to inaccurate predictions.

By scaling both features to a common range, we ensure that both features contribute equally to the distance calculation, leading to more accurate predictions.

**Additional Considerations:**

* Categorical Features: Convert categorical features into numerical representations using techniques like one-hot encoding or label encoding.
* Feature Selection: Consider using feature selection techniques to identify the most relevant features, which can improve the accuracy and efficiency of KNN.

In summary, preprocessing steps like handling missing values, encoding categorical variables, and especially feature scaling are crucial for the effective application of the KNN algorithm. Scaling transforms the features to be on a comparable scale, which is essential for accurate distance calculations and overall model performance.

**K means clustering**

**1. Use the K-Means algorithm to cluster a real-world dataset (e.g., Iris dataset, Mall Customers dataset, or any dataset of your choice).**

**a.Visualize the clusters using scatter plots (2D or 3D) and label each cluster.**

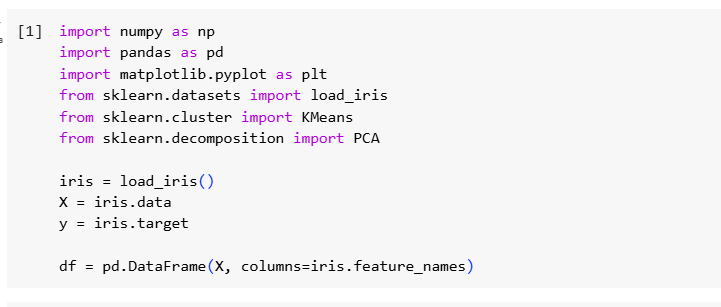
**b.Experiment with different values of k (e.g. k=2, 3, 4, 5) and visualize the clustering results for each case.**

**c.What challenges might arise when selecting k, and how can domain knowledge assist in making this decision? Use the Elbow Method to determine the best value for k.**

**d.Discuss three limitations of K-Means and suggest possible improvements or alternatives to address these limitations.**

**Answer:**

1. Visualize the clusters using scatter plots 3D and label each cluster.

**Code:**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

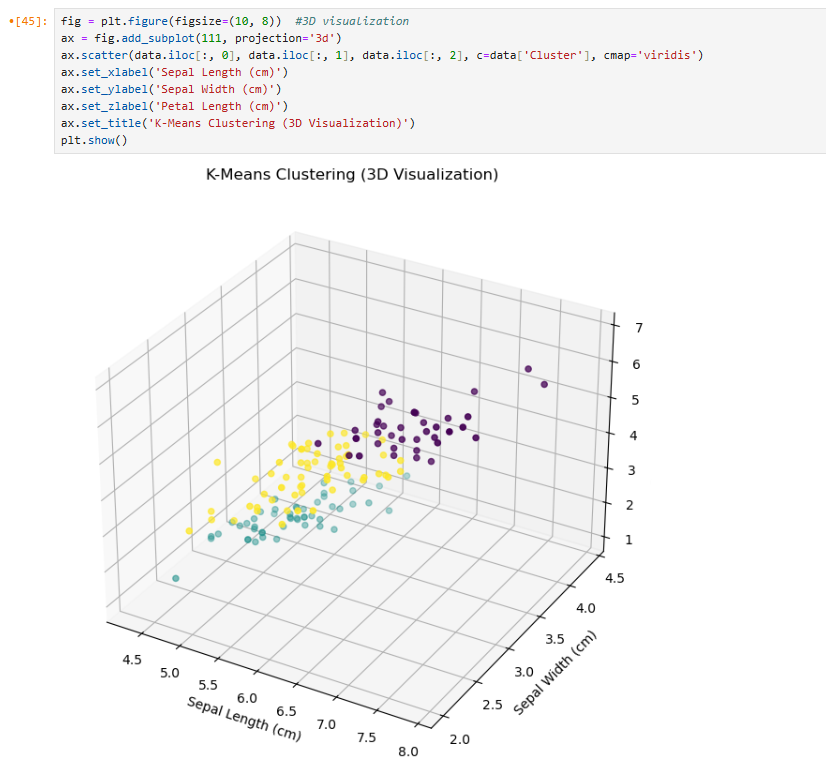
from mpl\_toolkits.mplot3d import Axes3D

# Load Iris dataset

iris = load\_iris()

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

target = iris.target

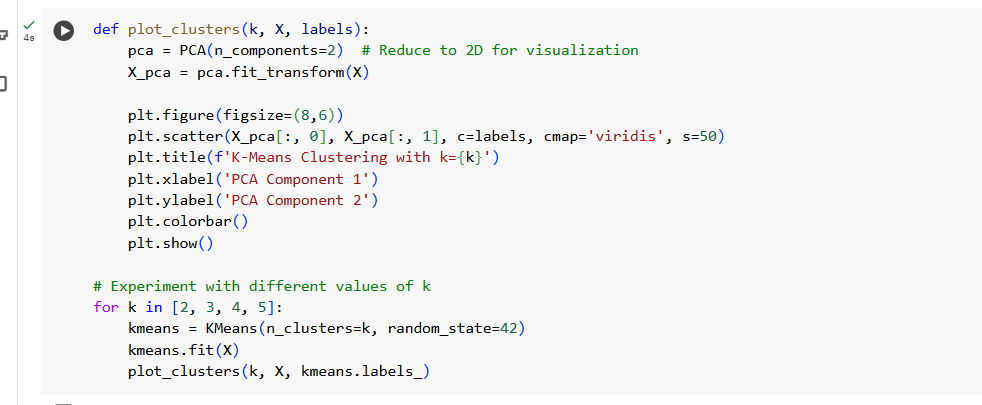
****

b.Let's visualize the clusters using different values of k on a real-world dataset. We'll use the Iris dataset for this example. Here's the step-by-step process:

Step-by-Step Process

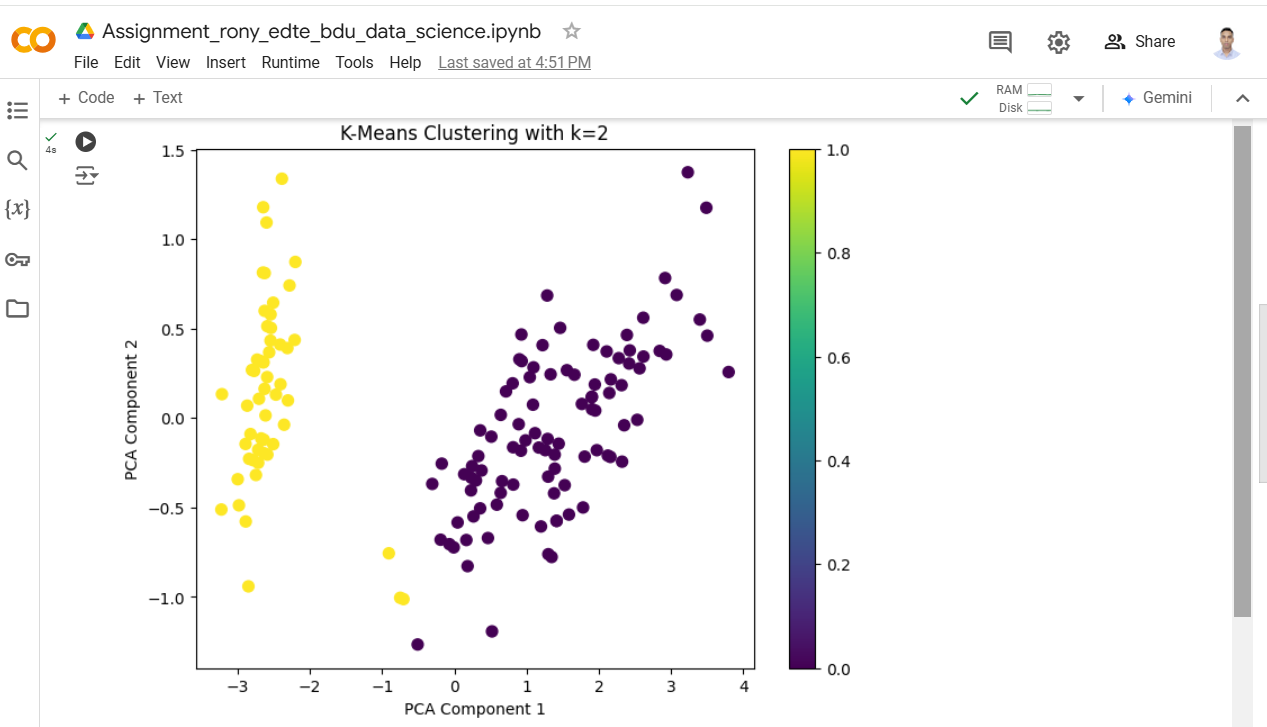
* Load and Standardize the Iris Dataset.
* Apply K-Means Clustering with Different Values of k.
* Visualize the Clustering Results Using Scatter Plots.

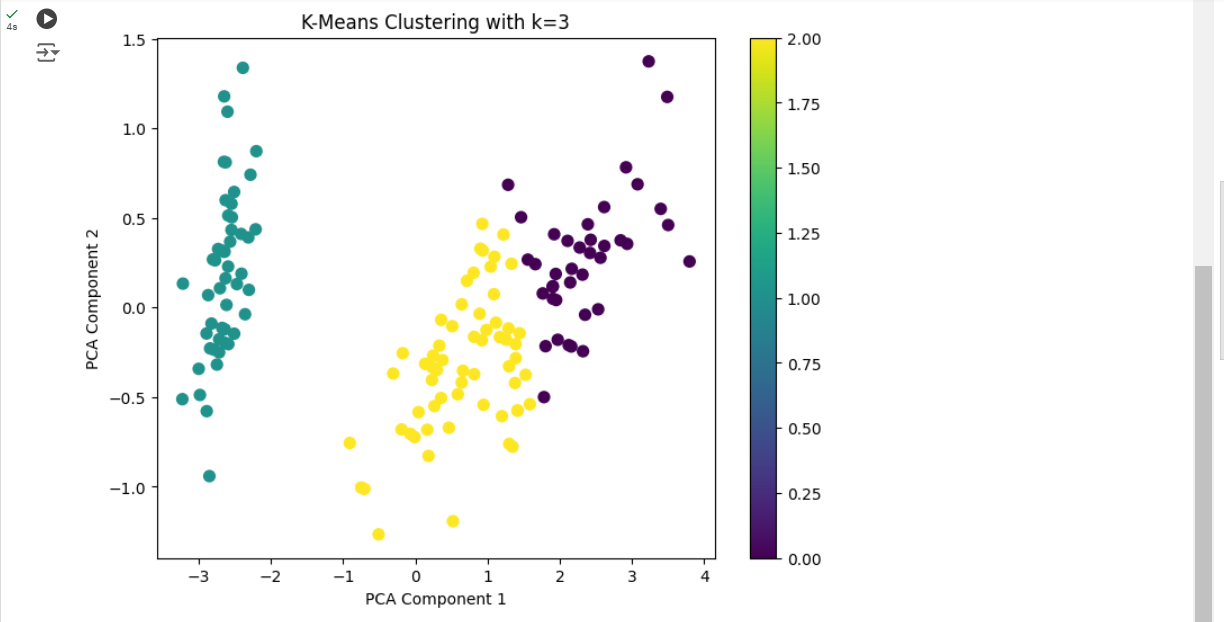
Next, we'll apply K-Means clustering with different values of k (2, 3, 4, 5) and visualize the results:

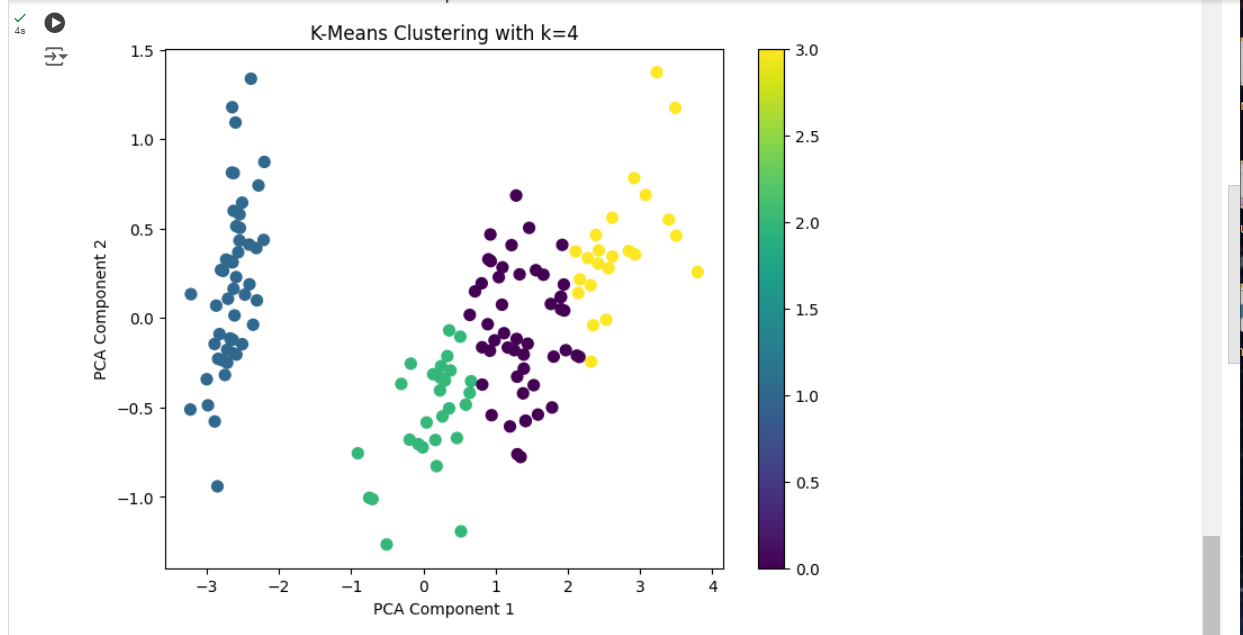


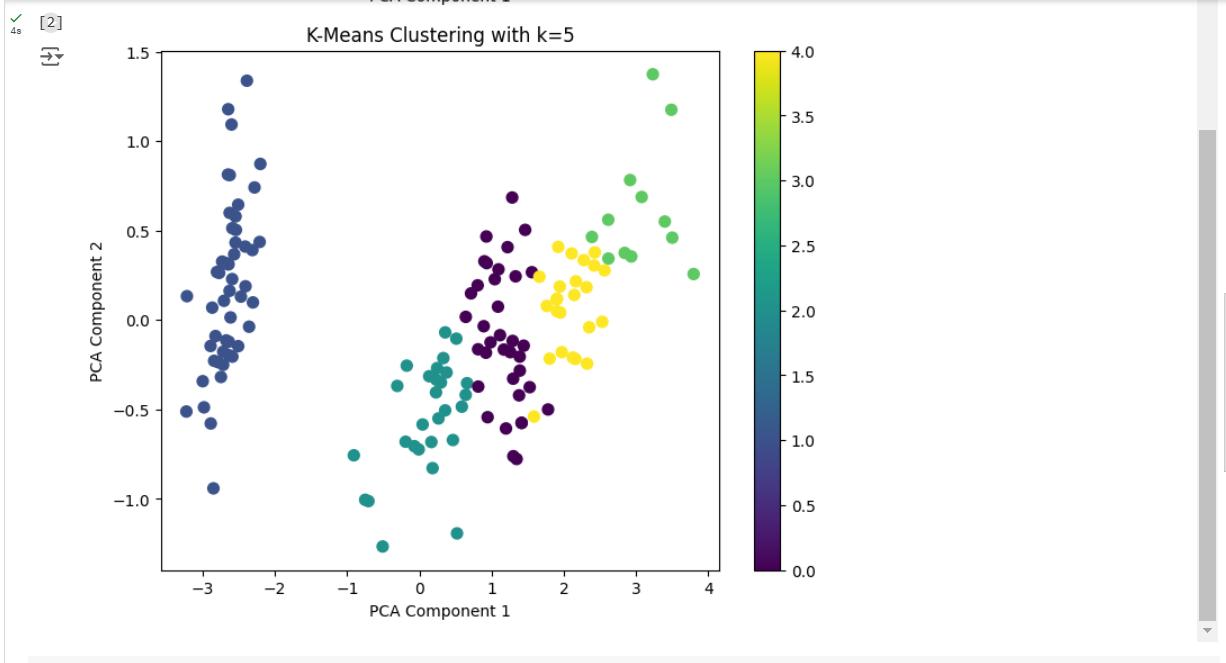
Results

Running the above code will generate scatter plots for k=2, 3, 4, and 5, displaying how the clustering results change with different values of k.









c.**Challenges in Selecting k :**

Selecting the optimal number of clusters 𝑘

, in K-Means clustering involves several challenges:

* Subjectivity: The choice of 𝑘 can be subjective and highly dependent on the dataset. Without domain knowledge, it can be difficult to determine the number of natural clusters in the data.
* Cluster Shape and Size: K-Means assumes spherical clusters of similar sizes, which may not be valid for all datasets. If clusters are of different shapes or sizes, choosing 𝑘 becomes more challenging.
* Overfitting and Underfitting:

1.Underfitting: Choosing too few clusters can lead to underfitting, where the algorithm fails to capture the underlying structure of the data.

2.Overfitting: Choosing too many clusters can lead to overfitting, where the algorithm captures noise rather than meaningful patterns.

* Computational Cost: Evaluating different values can be computationally expensive, especially for large datasets.

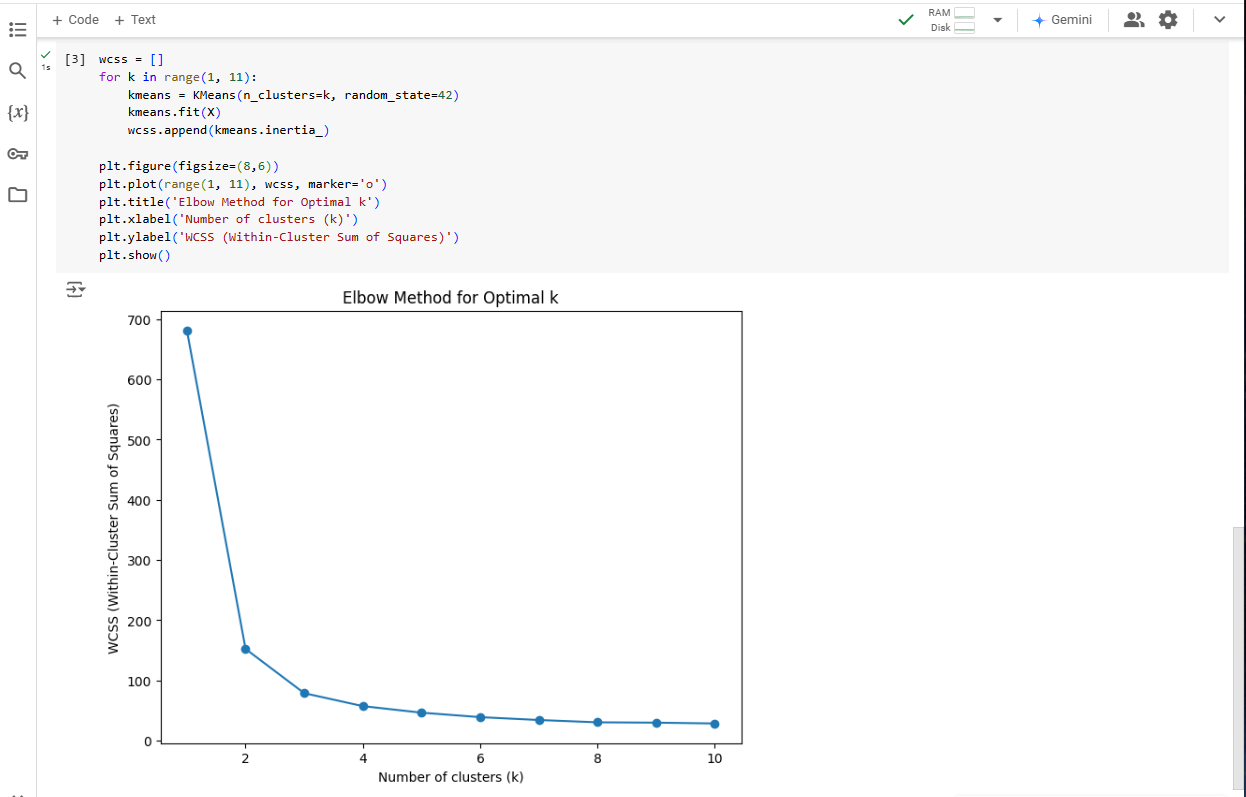
**Role of Domain Knowledge**

Domain knowledge can significantly assist in selecting 𝑘:

* Understanding Data Context: Knowledge about the data's context can provide insights into the expected number of clusters. For example, in a customer segmentation scenario, knowing the different customer personas can guide the selection of 𝑘.
* Interpreting Clusters: Domain knowledge helps in interpreting the clusters and validating whether they make sense. This can prevent the selection of arbitrary or meaningless clusters.
* Initial Estimates: Domain experts can provide initial estimates for k, which can then be refined using quantitative methods like the Elbow Method.

**Using the Elbow Method**

The Elbow Method helps in determining the optimal number of clusters by plotting the within-cluster sum of squares (inertia) against the number of clusters and identifying the point where the rate of decrease sharply changes (the "elbow").



**Interpreting the Elbow Plot**

Identifying the Elbow Point: Look for the point on the plot where the inertia starts to decrease more slowly, forming an "elbow." This point represents the optimal number of clusters. Beyond this point, adding more clusters results in only marginal gains in explaining variance.

Using the Elbow Method helps to

objectively choose 𝑘 by balancing the trade-off between the number of clusters and the within-cluster variance. However, combining this method with domain knowledge ensures that the selected 𝑘

is both statistically and contextually meaningful.

d.Here are three notable limitations of the K-Means algorithm, along with some suggestions for improvements or alternatives to address these issues:

**1. Assumption of Spherical Clusters**

* **Limitation:**

K-Means assumes that clusters are spherical and have similar sizes, which may not hold true for all datasets. As a result, K-Means may struggle to correctly identify clusters that have different shapes or variances.

* Improvement/Alternative:
  + Gaussian Mixture Models (GMM): GMMs extend K-Means by modeling data points as belonging to Gaussian distributions, allowing for more flexibility in the shape and size of clusters.
  + DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN identifies clusters based on the density of data points, allowing for the detection of clusters with arbitrary shapes and varying sizes. It is also robust to noise.

2**. Sensitivity to Initialization**

* Limitation:K-Means is sensitive to the initial placement of centroids. Poor initialization can lead to suboptimal clustering results and may require multiple runs to achieve a good outcome.
* Improvement/Alternative:
  + k-means++ Initialization: This method improves the initial placement of centroids by spreading them out more effectively, reducing the chances of poor initial centroids.
  + Multiple Initializations: Running K-Means multiple times with different random initializations and averaging the results can help achieve more stable and reliable clusters.

3**. Difficulty Handling Outliers**

* Limitation:Outliers can significantly affect the centroids in K-Means, distorting the clusters and leading to poor performance.
* Improvement/Alternative:
  + Preprocessing Steps: Implementing robust preprocessing steps, such as outlier detection and removal or transformation techniques, can mitigate the impact of outliers.
  + Alternative Algorithms: Algorithms like “DBSCAN”and “OPTICS (Ordering Points To Identify the Clustering Structure)”are better suited for datasets with outliers, as they can identify core points and noise separately.

By considering these improvements and alternative methods, you can enhance the effectiveness and robustness of clustering tasks using K-Means.

**Decision Tree**

**The dataset consists of the following columns:**

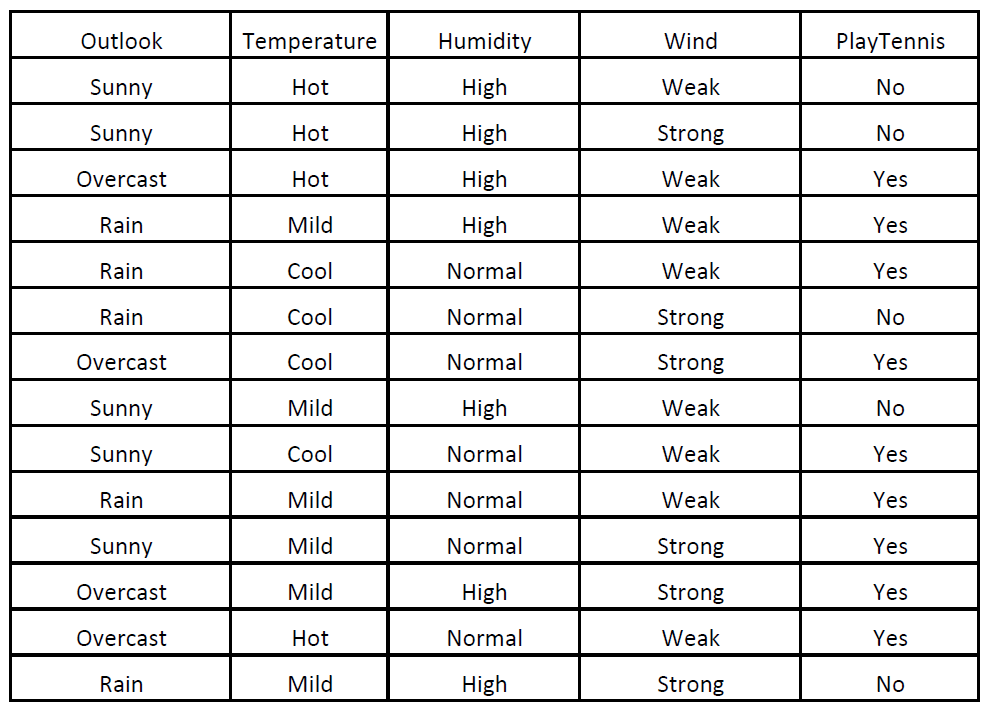
**Outlook: The weather outlook (Sunny, Overcast, Rain).**

**Temperature: The temperature (Hot, Mild, Cool).**

**Humidity: The humidity level (High, Low).**

**Wind: Whether the wind is strong or weak (Weak, Strong).**

**PlayTennis: Target variable (Yes/No) indicating whether tennis can be played.**



**Exploratory Data Analysis (EDA):**

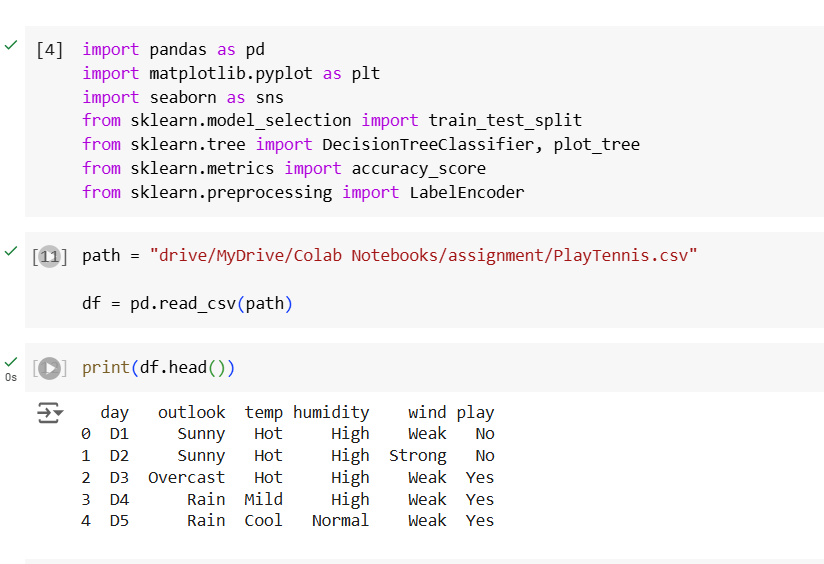
**(a) Load the dataset and display the first few rows.**

**(b) Provide a summary of the dataset. Discuss the number of instances, features, and unique values for each feature.**

**(c) Visualize the distribution of categorical features such as Outlook, Temperature, Humidity, and Wind using bar plots.**

**Answer :**

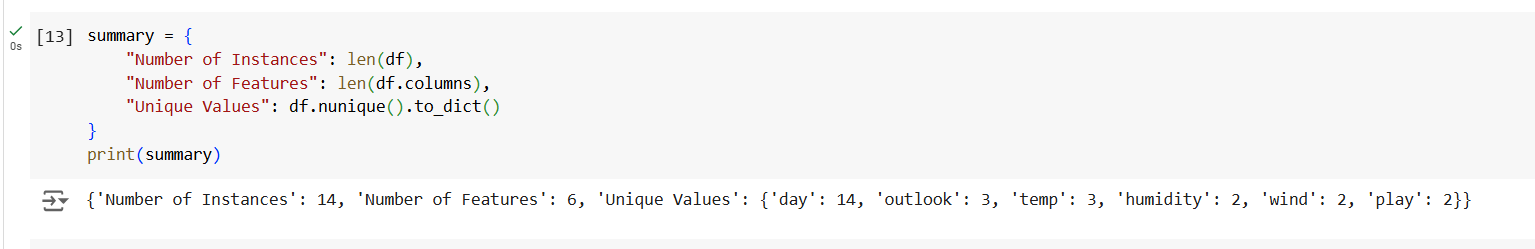
**(a) Load the dataset and display the first few rows**

****

### **(b) Provide a summary of the dataset**

**A summary includes:**

**The number of instances, features, and unique values for each feature.**

****

**Summary Output:**

Number of instances: 14

Number of features: 5 (Outlook, Temperature, Humidity, Wind, Play)

Unique values:

Outlook: 3 (Sunny, Overcast, Rain)

Temperature: 3 (Hot, Mild, Cool)

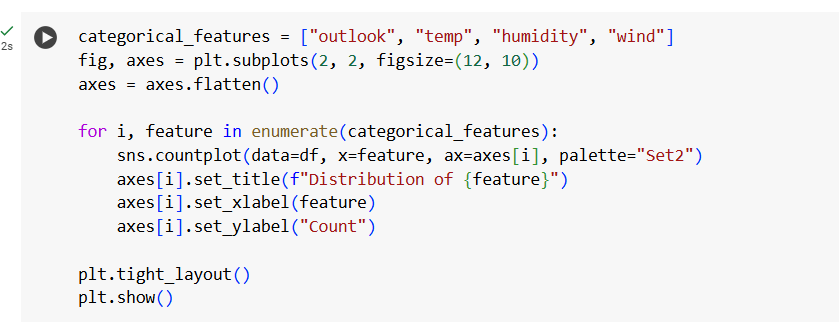
Humidity: 2 (High, Normal)

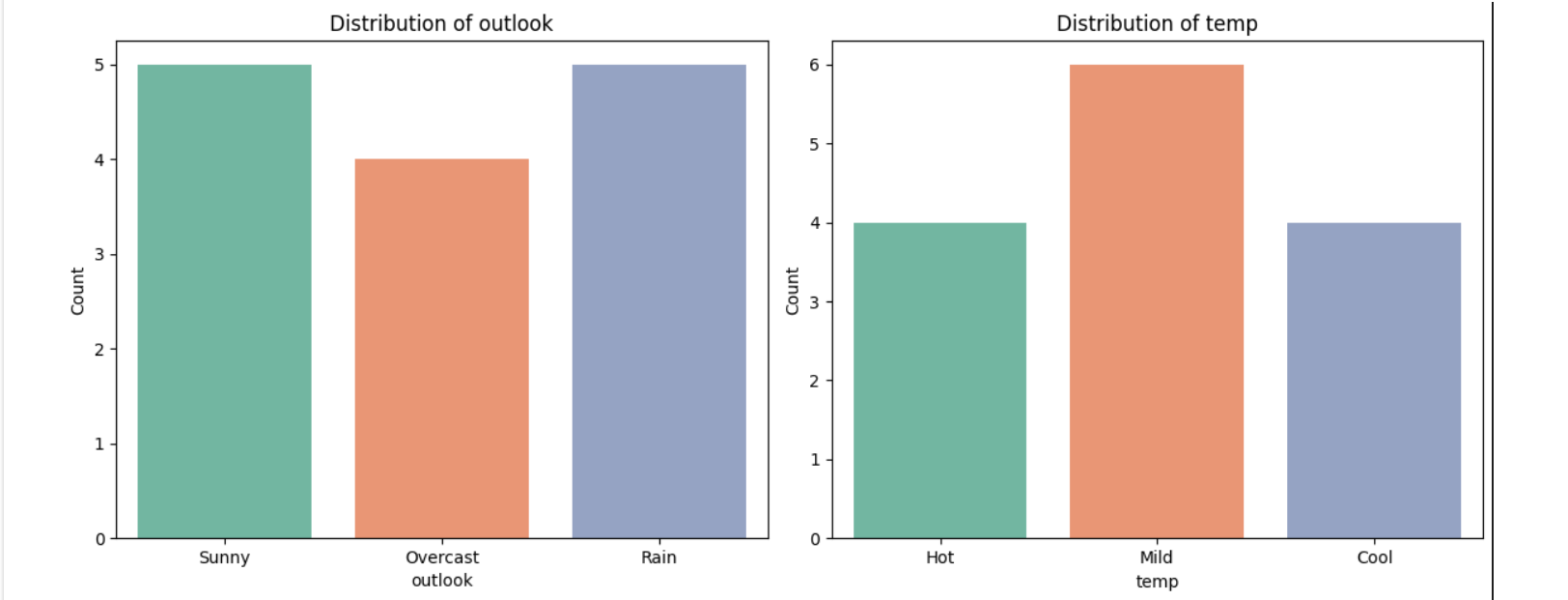
Wind: 2 (Weak, Strong)

Play: 2 (Yes, No)

### **(c) Visualize the distribution of categorical features**

**For features like Outlook, Temperature, Humidity, and Wind, bar plots provide insights into their distributions.**







These bar plots will display:

* Outlook: Count of sunny, overcast, and rainy days.
* Temperature: Count of hot, mild, and cool days.
* Humidity: Count of high and normal humidity days.
* Wind: Count of weak and strong wind days.

These visualizations help in understanding the distribution of categorical features in the dataset.

Data Preprocessing:

(a) Convert categorical features (Outlook, Temperature, Humidity, Wind) into numerical values using one-hot encoding or label encoding.

(b) Split the dataset into training (80%) and testing (20%) sets.

(c) Handle any missing data (if present) or explain why there is no missing data in this case.

Model Building:

(a) Train a Decision Tree classifier using the training dataset.

(b) Visualize the resulting decision tree using a plotting library like matplotlib.

(c) Report the accuracy of the classifier on the test and train dataset.

Model Tuning:

(a) Experiment with the max\_depth and min\_samples\_split hyperparameters. Discuss how adjusting these parameters influences the model’s performance and complexity.

(b) What happens to the tree if max\_depth is set too high? How does it affect overfitting?

Data Preprocessing:

**a) Answer :**  
Code:  
le=LabelEncoder()

df['outlook']=le.fit\_transform(df['outlook'])

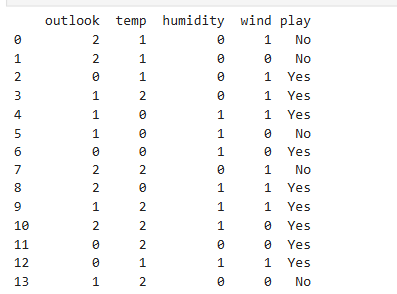
df['temp']=le.fit\_transform(df['temp'])

df['humidity']=le.fit\_transform(df['humidity'])

df['wind']=le.fit\_transform(df['wind'])

print (df)

Input:  
  


Output:  
  
  
  
**b)Answer:**

The code required to split the dataset into training (80%) and testing (20%) sets is given below:

**Code:**

from sklearn.model\_selection import train\_test\_split

x = df.drop("PlayTennis", axis=1)

y = df["PlayTennis"]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

print("Training Set:")

print(x\_train)

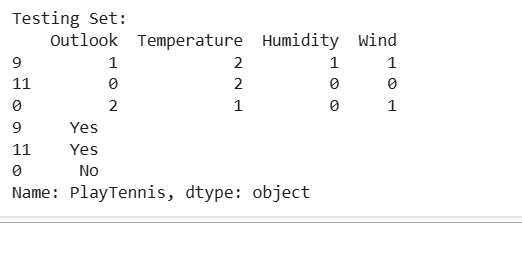
print(y\_train)

print("\nTesting Set:")

print(x\_test)

print(y\_test)

**Output:**



**c)Answer:**

To see if there is any missing data -

**Code:**

# Check for missing values

missing\_data = df.isnull().sum()

# Display columns

print("Missing Data Per Column:")

print(missing\_data)

# Check if there are any missing

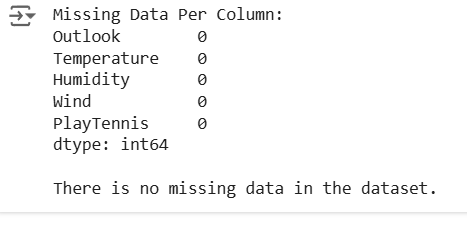
if df.isnull().values.any():

    print("\nThere is missing data in the dataset.")

else:

    print("\nThere is no missing data in the dataset.")

**Output:**



**Model Building:**

**a)Answer:**

The code needed to train a Decision Tree classifier using the training dataset is described below:

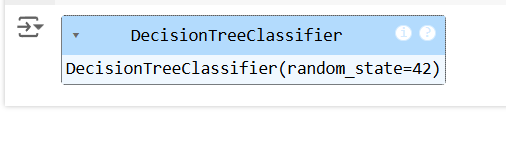
**Code:**

from sklearn.tree import DecisionTreeClassifier

clobj = DecisionTreeClassifier(random\_state=42)

clobj.fit(x\_train, y\_train)

**Output:**



**b)Answer:**

The code required to visualize the resulting decision tree

**Code:**

from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

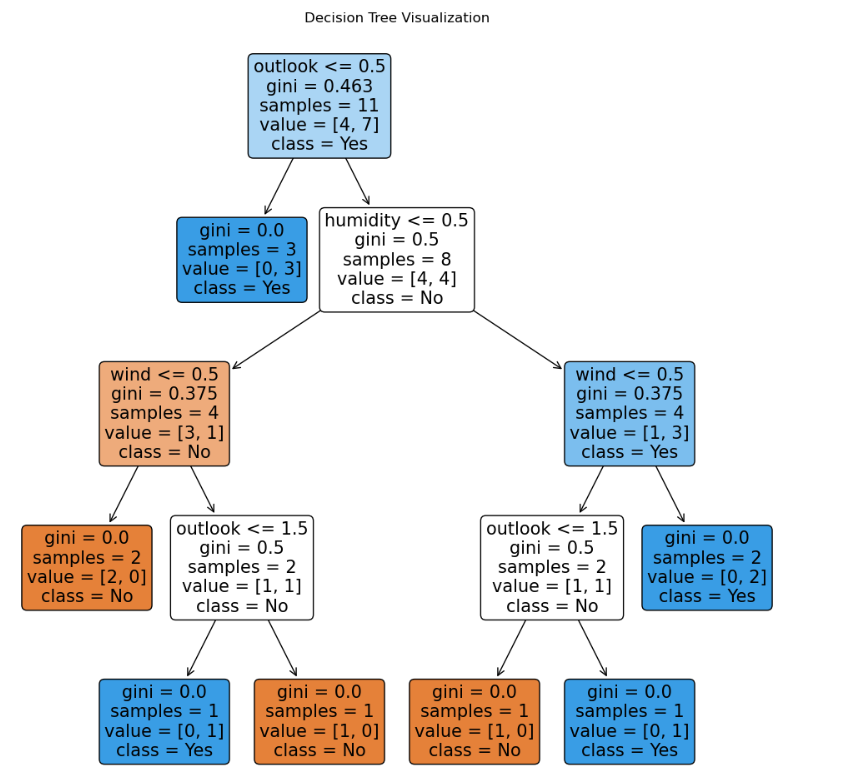
plt.figure(figsize=(12, 12))

plot\_tree(clobj, feature\_names=x.columns.tolist(), class\_names=["No", "Yes"], filled=True, rounded=True)

plt.title("Decision Tree Visualization")

plt.show()

**Output:**



**c)Answer:**

The code needed to measure the accuracy of the classifier

**Code:**

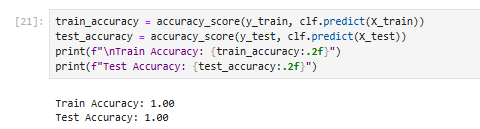
train\_acc = accuracy\_score(y\_train, clobj.predict(x\_train))

test\_acc = accuracy\_score(y\_test, clobj.predict(x\_test))

print(f"\nTrain Accuracy: {train\_acc:.2f}")

print(f"Test Accuracy: {test\_acc:.2f}")

**Output:**



**Model Tuning**

**a)Answer:**

**Code:**

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

max\_depths = [1, 2, 3, 5, 10]

min\_samples\_splits = [2, 5, 10, 20]

results = []

for max\_depth in max\_depths:

for min\_samples\_split in min\_samples\_splits:

clobj = DecisionTreeClassifier(max\_depth=max\_depth, min\_samples\_split=min\_samples\_split, random\_state=42)

clobj.fit(x\_train, y\_train)

y\_pred = clobj.predict(x\_test)

# Computing accuracy

acc = accuracy\_score(y\_test, y\_pred)

results.append({

"max\_depth": max\_depth,

"min\_samples\_split": min\_samples\_split,

"accuracy": acc

})

results\_df = pd.DataFrame(results)

print("Hyperparameter Tuning Results:")

print(results\_df)

# Plotting the results for visualization

for min\_samples\_split in min\_samples\_splits:

subset = results\_df[results\_df["min\_samples\_split"] == min\_samples\_split]

plt.plot(subset["max\_depth"], subset["accuracy"], label=f"min\_samples\_split={min\_samples\_split}")

plt.title("Accuracy vs. max\_depth for different min\_samples\_split values")

plt.xlabel("max\_depth")

plt.ylabel("Accuracy")

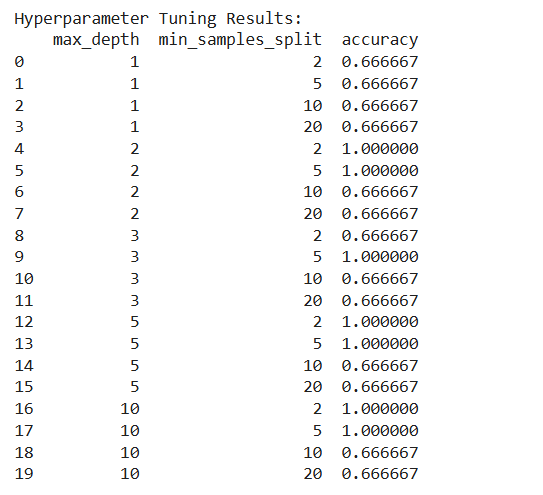
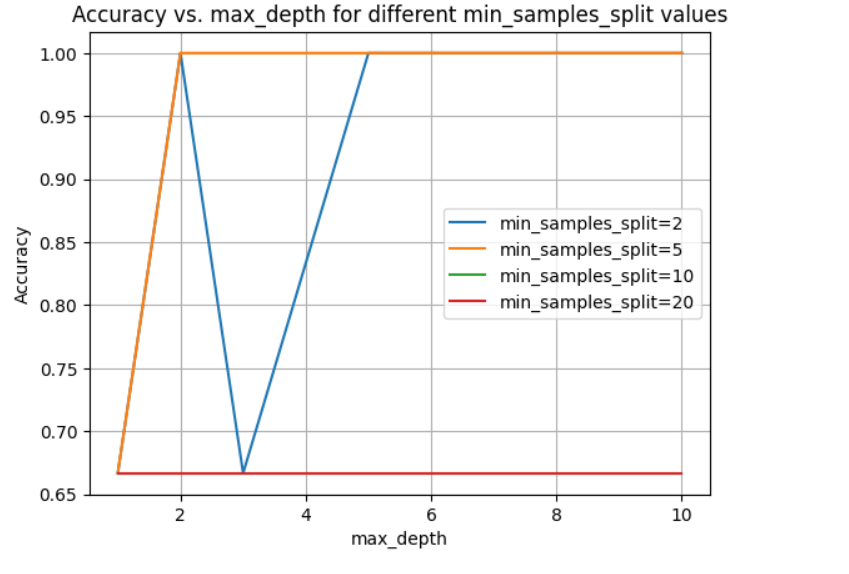
plt.legend()

plt.grid(True)

plt.show()



**Output:**



we can say thatOptimal values for max\_depth and min\_samples\_split strike a balance between underfitting and overfitting. Moderate values often yield the best performance. Use cross-validation for fine-tuning.

**b)Answer:**  
  
When a decision tree's maximum depth (max\_depth) is set too high, it leads to several issues, primarily overfitting, increased complexity, and poor generalization. Let’s break this down in a simple and detailed way:

**1. Tree Grows Too Deep**

* Setting a high max\_depth allows the decision tree to continue splitting until it perfectly fits the training data.
* The tree creates many levels, with each split focusing on small details in the dataset.
* As a result, the tree captures not only the significant patterns but also small variations or even random noise in the data.

**2. Overfitting Occurs**

* **What is overfitting?**  
  Overfitting happens when the model becomes too tailored to the training data and struggles to perform well on new, unseen data.
* **Why does overfitting occur in deep trees?**  
  A deep tree "memorizes" the training data, including irrelevant details, instead of learning the overall trends or patterns. For example:
  + If there’s random noise or outliers in the dataset, the tree tries to fit them as well.
  + This results in a highly specific model that excels on training data but performs poorly on test data, where these minor details might not exist.

**3. Increased Complexity**

* **Interpretability:**  
  A deeper tree means more branches, nodes, and splits. It becomes harder for humans to understand the logic behind the predictions.
  + For example, instead of a simple rule like "If age > 30, classify as X," the tree might create unnecessarily complex rules that are difficult to follow.
* **Computational Costs:**  
  A very deep tree requires more time and resources to:
  + Train, because it has to process numerous splits.
  + Make predictions, as it has to traverse more levels to reach a decision.

**Why does high max\_depth lead to poor generalization?**

* When the tree grows too deep, it not only learns the important patterns in the data but also memorizes the "noise" or random fluctuations.
* This results in:
  + **High training accuracy:** The tree performs perfectly on the training data because it fits every detail.
  + **Low testing accuracy:** The model struggles with new data because it hasn't learned general rules; it only knows how to deal with the specific examples it saw during training.