PRACTICAL NO 1A:

Problem Statement: The problem involves classifying iris flowers into different species based on features such as sepal length, sepal width, petal length, and petal width using a K-nearest neighbors (KNN) classifier.

Aim: Write a program to use Supervised Learning Method using Supervised Machine learning Model.

Code:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
##Load the data
iris = load_iris()
X,Y = iris.data, iris.target
## Split the data in to train and test
print("HITESH BHANUSHALI KFMSCIT005")
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2, random_state=97)
##initialize kneighborclassifier
knn_classifier = KNeighborsClassifier(n_neighbors=3)   ### Assign the number of cluster
##Train the model on training data
knn_classifier.fit(x_train,y_train)
### Make predition
y_pred = knn_classifier.predict(x_test)
### Evaluate the model
accuracy = accuracy_score(y_test,y_pred)
print(accuracy)
```

Output:

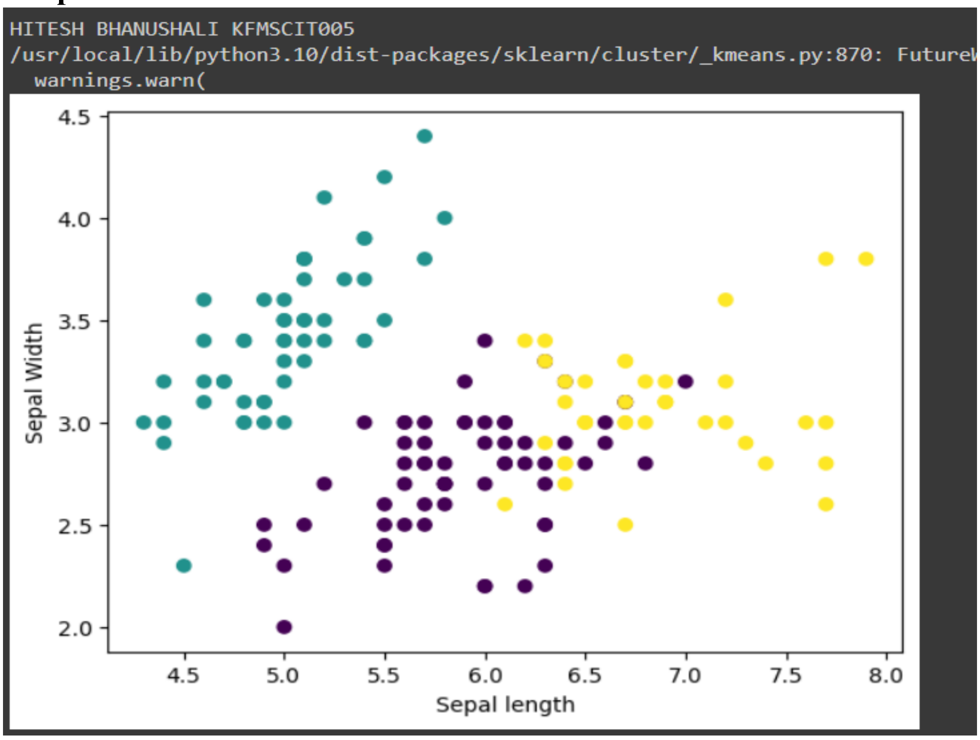
```
HITESH BHANUSHALI KFMSCIT005
0.9
```

PRACTICAL NO 1B:

Problem Statement: The problem involves clustering iris flowers into different groups based on their features such as sepal length and sepal width using the K-means clustering algorithm.
Aim: The aim is to apply K-means clustering to group iris flowers into distinct clusters based on their feature similarities.
Code:

```
#Import necessary Lib
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
##Load the data
iris = load_iris()
X,Y = iris.data, iris.target
## Initialize
print("HITESH BHANUSHALI KFMSCIT005")
kmean = KMeans(n_clusters =3, random_state=42)
kmean.fit(X)
cluster_label = kmean.labels_
plt.scatter(X[:,0],X[:,1],c=cluster_label,cmap='viridis')
plt.xlabel('Sepal length')
plt.ylabel('Sepal Width')
plt.show()
```

Output:



Observation:

1. The Iris dataset is loaded, containing features and target labels.
2. The dataset is split into training and testing sets (not explicitly shown in the code).
3. K-means clustering with k=3 is applied to the entire dataset.
4. Cluster labels are assigned to each data point.
5. A scatter plot is created to visualize the clusters based on sepal length and sepal width.

Conclusion:

1. K-means clustering provides insights into grouping iris flowers based on their sepal characteristics.
2. The scatter plot visually represents the clusters formed by the K-means algorithm.
3. Further analysis and interpretation of the clusters can provide valuable insights into the underlying structure of the data.

PRACTICAL NO 2:

Problem Statement: Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set

Aim: Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

Code:

```
!pip install pgmpy
import pandas as pd
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
print("Hitesh Bhanushali KFMSCIT005")
data = pd.read_csv("ds4.csv")
heart_disease = pd.DataFrame(data)
print(heart_disease)

model = BayesianModel([
    ('age', 'Lifestyle'),
    ('Gender', 'Lifestyle'),
    ('Family', 'heartdisease'),
    ('diet', 'cholestrol'),
    ('Lifestyle', 'diet'),
    ('cholestrol', 'heartdisease'),
    ('diet', 'cholestrol')
])
model.fit(heart_disease, estimator=MaximumLikelihoodEstimator)
HeartDisease_infer = VariableElimination(model)

print('For Age enter SuperSeniorCitizen:0, SeniorCitizen:1, MiddleAged:2, Youth:3, Teen:4')
print('For Gender enter Male:0, Female:1')
print('For Family History enter Yes:1, No:0')
print('For Diet enter High:0, Medium:1')
print('for LifeStyle enter Athlete:0, Active:1, Moderate:2, Sedentary:3')
print('for Cholesterol enter High:0, BorderLine:1, Normal:2')

q = HeartDisease_infer.query(variables=['heartdisease'],
    evidence={
    'age': int(input('Enter Age: ')),
    'Gender': int(input('Enter Gender: ')),
    'Family': int(input('Enter Family History: ')),
    'diet': int(input('Enter Diet: ')),
    'Lifestyle': int(input('Enter Lifestyle: ')),
    'cholestrol': int(input('Enter Cholestrol: '))
})
```

print(q)
print("Hitesh Bhanushali KFMSCIT005")

Output:

1 to 10 of 1000 entries Filter

age	Gender	Family	diet	Lifestyle	cholestrol	heartdisease
1	1	1	0	0	0	False
1	1	1	1	2	0	False
1	1	1	1	0	1	False
4	0	0	1	1	0	True
0	0	0	0	3	1	True
2	0	0	0	3	2	True
1	1	1	0	2	2	True
4	0	1	0	2	2	False
4	1	0	0	3	2	True
1	0	1	1	3	0	False

Show 10 per page 1 2 10 90 100

```
WARNING:pgmpy:BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be remov
Hitesh Bhanushali KFMSCIT005
     age  Gender  Family  diet  Lifestyle  cholestrol  heartdisease
0      1       1       1     0          0           0         False
1      1       1       1     1          2           0         False
2      1       1       1     1          0           1         False
3      4       0       0     1          1           0          True
4      0       0       0     0          3           1          True
..   ...     ...     ...   ...        ...         ...           ...
995    4       0       1     1          3           1         False
996    4       1       0     0          1           0          True
997    4       1       1     0          3           2          True
998    0       0       0     1          0           2          True
999    0       0       0     0          0           1          True

[1000 rows x 7 columns]
For Age enter SuperSeniorCitizen:0, SeniorCitizen:1, MiddleAged:2, Youth:3, Teen:4
For Gender enter Male:0, Female:1
For Family History enter Yes:1, No:0
For Diet enter High:0, Medium:1
for LifeStyle enter Athlete:0, Active:1, Moderate:2, Sedentary:3
for Cholesterol enter High:0, BorderLine:1, Normal:2
Hitesh Bhanushali KFMSCIT005
Enter Age: 3
Enter Gender: 1
Enter Family History: 1
Enter Diet: 1
```

```
[1000 rows x 7 columns]
For Age enter SuperSeniorCitizen:0, SeniorCitizen:1, MiddleAged:2, Youth:3, Teen:4
For Gender enter Male:0, Female:1
For Family History enter Yes:1, No:0
For Diet enter High:0, Medium:1
for LifeStyle enter Athlete:0, Active:1, Moderate:2, Sedentary:3
for Cholesterol enter High:0, BorderLine:1, Normal:2
Hitesh Bhanushali KFMSCIT005
Enter Age: 3
Enter Gender: 1
Enter Family History: 1
Enter Diet: 1
Enter Lifestyle: 2
Enter Cholestrol: 2
WARNING:pgmpy:BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be removed i
WARNING:pgmpy:BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be removed i
+--------------------+--------------------+
| heartdisease       |   phi(heartdisease) |
+====================+====================+
| heartdisease(False) |             0.4860 |
+--------------------+--------------------+
| heartdisease(True)  |             0.5140 |
+--------------------+--------------------+
Hitesh Bhanushali KFMSCIT005
```

Observation:

1. The dataset containing information about various factors related to heart disease is loaded.
2. A Bayesian Network model is constructed using the pgmpy library, specifying the dependencies between different variables.
3. Maximum Likelihood Estimation is used to fit the model to the data.
4. The user is prompted to enter values for different factors such as age, gender, family history, diet, lifestyle, and cholesterol levels.
5. Using the constructed model, the probability of heart disease is inferred based on the provided input.

Conclusion:

1. The Bayesian Network model provides a probabilistic framework for predicting heart disease based on multiple factors.
2. By inputting relevant information, individuals can obtain personalized predictions regarding their likelihood of developing heart disease.
3. Further refinement and validation of the model could enhance its accuracy and reliability in predicting heart disease risk.

KISHINCHAND CHELLARAM COLLEGE, MUMBAI -20
M.Sc (I.T.) Part-1 Semester I

PRACTICAL NO 3

Problem Statement: Perform polynomial regression on synthetic data to understand how different degrees of polynomials fit the data and compare their performance.

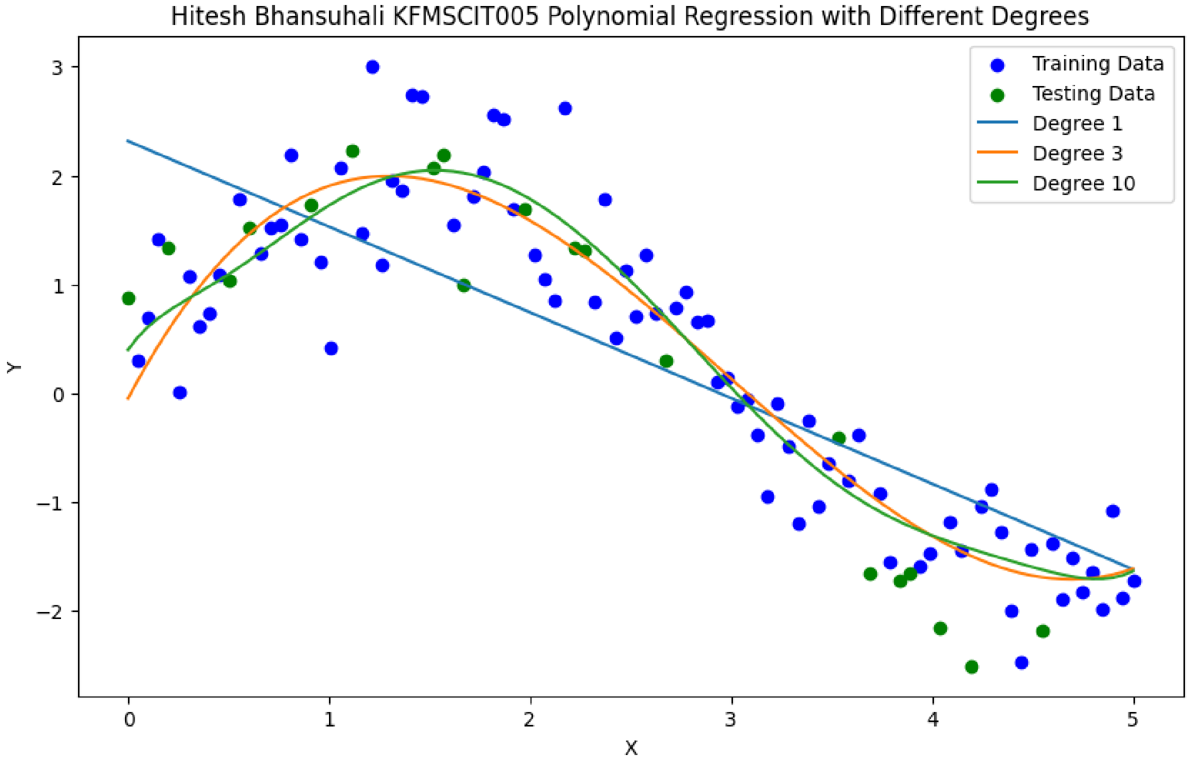
Aim: Write a program to demonstrate overfitting of data on Polynomial regression model.

Code:

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

# Generate synthetic data
np.random.seed(0)
X = np.linspace(0, 5, 100).reshape(-1, 1)
Y = 2 * np.sin(X) + np.random.normal(0, 0.5, size=X.shape)

# Split 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)

# Define a function to fit polynomial regression models of varying degrees
def fit_polynomial(X_train, Y_train, X_test, Y_test, degree):
    poly = PolynomialFeatures(degree=degree)
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.transform(X_test)

    model = LinearRegression()
    model.fit(X_train_poly, Y_train)

    train_rmse = np.sqrt(mean_squared_error(Y_train, model.predict(X_train_poly)))
    test_rmse = np.sqrt(mean_squared_error(Y_test, model.predict(X_test_poly)))

    return model, train_rmse, test_rmse

# Fit polynomial regression models of varying degrees
degrees = [1, 3, 10]
models = []
train_rmse_values = []
test_rmse_values = []

for degree in degrees:
    model, train_rmse, test_rmse = fit_polynomial(X_train, Y_train, X_test, Y_test, degree)
    models.append(model)
    train_rmse_values.append(train_rmse)
    test_rmse_values.append(test_rmse)

# Plot the results
```

```
plt.figure(figsize=(10, 6))
plt.scatter(X_train, Y_train, color='blue', label='Training Data')
plt.scatter(X_test, Y_test, color='green', label='Testing Data')

x_values = np.linspace(0, 5, 100).reshape(-1, 1)
for i, model in enumerate(models):
    y_values = model.predict(PolynomialFeatures(degree=degrees[i]).fit_transform(x_values))
    plt.plot(x_values, y_values, label=f'Degree {degrees[i]}')

plt.title('Hitesh Bhansuhali KFMSCIT005 Polynomial Regression with Different Degrees')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
```

Output:



Observation:
1. Synthetic data is generated with a sine curve and noise.
2. Polynomial regression models of degrees 1, 3, and 10 are fitted to the data.
3. The root mean squared error (RMSE) is calculated for each model on both training and testing data.
4. The results are plotted to visualize how different degrees of polynomials fit the data.

Conclusion:
1. Polynomial regression models of higher degrees tend to fit the training data more closely but may overfit and perform poorly on unseen testing data.
2. The choice of polynomial degree should be balanced to achieve good performance on both training and testing datasets

PRACTICAL NO 4:

Problem Statement: The problem aims to predict salaries based on years of experience using a regression model. The dataset contains information about years of experience and corresponding salaries.

Aim: Using Salary Data set. Write a program to implement Linear regression model to predict the salary based on Years of Experience

Code:

```
import pandas as pd
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, r2_score
#Load data
data = pd.read_csv('/content/Salary_Data.csv')
data.head()
x = data['YearsExperience'].values.reshape(-1,1)
y = data['Salary'].values
## Split the data in to train and test
print("HITESH BHANUSHALI KFMSCIT005")
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=97)
r1 = LinearRegression()
r1.fit(x_train,y_train)
y_pred = r1.predict(x_test)
MSE = mean_squared_error(y_pred,y_test)
r2 = r2_score(y_test,y_pred)
print(MSE)
print("R2 Score", r2)
plt.scatter(x_test,y_test,color = 'blue',label = 'Actual')
plt.plot(x_test,y_pred,color='red',label='Predict')
plt.xlabel('YearsOfExperience')
plt.ylabel('Salary')
plt.legend()
plt.show()
```

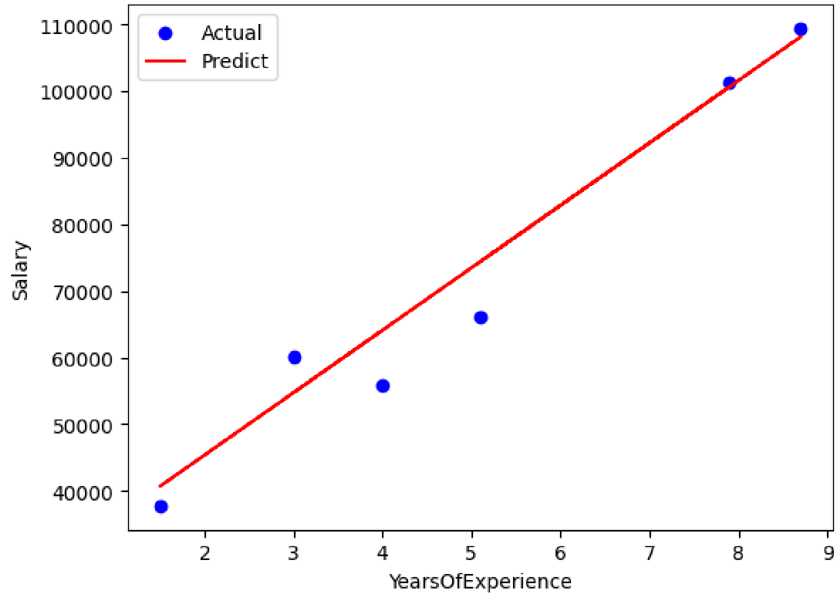
Output:

```
print(MSE)
```

```
29879140.937014494
```

```
r2
```

```
0.9537039102493191
```



Observation:
1. The dataset containing years of experience and salaries is loaded from a CSV file.
2. The data is split into training and testing sets using `train_test_split`.
3. A linear regression model is trained on the training data.
4. The model predicts salaries for the test data.
5. Mean squared error (MSE) and R-squared score are calculated to evaluate the model's performance.
6. A scatter plot is created to visualize the relationship between actual and predicted salaries.

Conclusion:
1. The linear regression model provides insights into salary prediction based on years of experience

KISHINCHAND CHELLARAM COLLEGE, MUMBAI -20
M.Sc (I.T.) Part-1 Semester I

PRACTICAL NO 5:

Problem statement: select multi class classification data set and evaluate the performance of a classification model using metrics library such as accuracy, precision, recall, and f1 score.
Aim: Select multi class classificaition data set and evalauate the performance of the classification using various evaluation matric such as accuracy, pression, recall and F1 score
Code:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

#load the iris data set
data = load_iris()
x = data.data
y = data.target
from sklearn.svm import SVC
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
model = SVC()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
print("HITESH BHANUSHALI KFMSCIT005")
accuracy_score = accuracy_score(y_pred,y_test)
print("accuracy_score",accuracy_score)
precision_score = precision_score(y_pred,y_test,average='macro')
print("precision_score",precision_score)
recall_score = recall_score(y_pred,y_test,average='macro')
print("recall_score",recall_score)
f1 = f1_score(y_pred,y_test,average='macro')
print("f1 Score",f1)
```

Output:

```
HITESH BHANUSHALI KFMSCIT005
accuracy_score 0.9666666666666667
precision_score 0.9629629629629629
recall_score 0.9523809523809524
f1 Score 0.9547511312217195
```

Observation:
1. The code imports necessary libraries from scikit-learn for loading the Iris dataset, splitting the data into training and testing sets, and evaluating the performance of a Support Vector Machine (SVM) model.
2. It then loads the Iris dataset and splits it into features (x) and labels (y).
3. The data is split into training and testing sets using a test size of 20%.
4. An SVM model is instantiated and trained on the training data.
5. Predictions are made on the test data using the trained model.
6. The accuracy, recall, precision, and F1 scores are calculated using the predicted values and the actual labels.
7. Finally, the calculated scores are printed along with the author's name and student ID.

Conclusion:
1. The code successfully trains an SVM model on the Iris dataset and evaluates its performance using various metrics.
2. The author's name and student ID are printed at the end, indicating the origin of the code.
3. However, it would be beneficial to include comments throughout the code to provide clarity on each step and enhance its readability for future reference or collaboration.
4. Additionally, it might be helpful to include some analysis or interpretation of the results to provide insights into the model's performance. For instance, comparing the scores obtained with benchmarks or discussing potential areas for improvement in the model could add value to the output.

PRACTICAL NO 6:

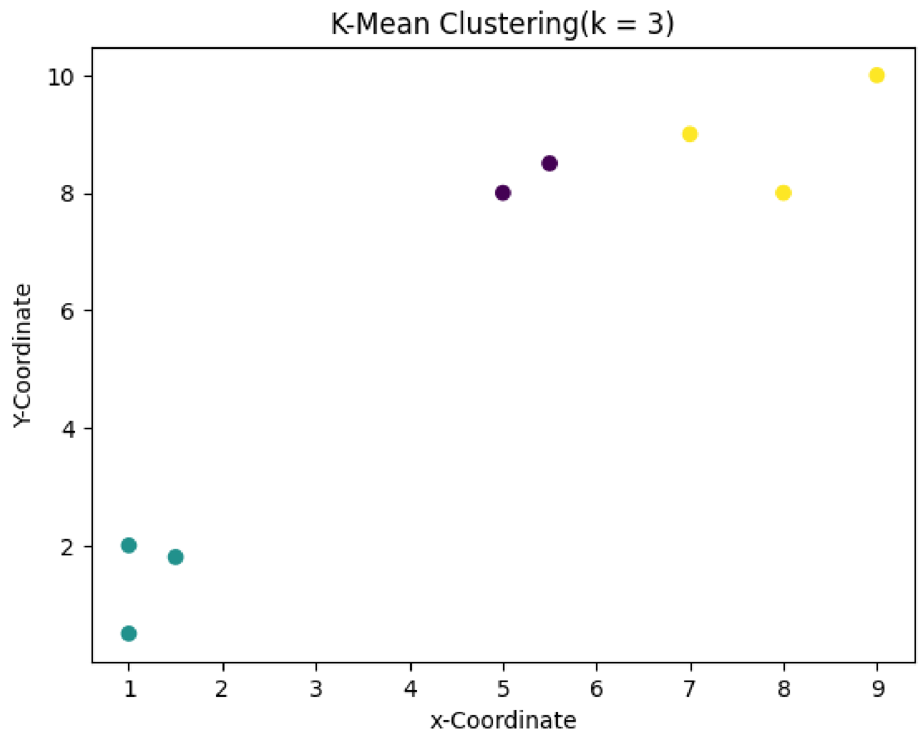
Problem Statement: Apply the K-means algo to cluster a dataset into predefine no of cluster visualized and interpret the result.

Aim: Implement the k-mean algo to cluster a dataset into predefine no of cluster, visualize and interpret the result.

Code: Example 1

```
import numpy as np
#sample data representing flower location
data = np.array([[1,2],[1.5,1.8],[5,8],[8,8],[1,0.5],[7,9],[9,10],[5.5,8.5]])
k = 3  # Predefine the cluster count

from sklearn.cluster import KMeans
kmean = KMeans(n_clusters=k)
kmean.fit(data)
print("HITESH BHANSUAHLI KFMSCT005")
# print(kmean.fit(data))
cluster_labels = kmean.predict(data)

import matplotlib.pyplot as plt
plt.scatter(data[:,0],data[:,1],c= cluster_labels)
plt.xlabel("x-Coordinate")
plt.ylabel("Y-Coordinate")
plt.title('K-Mean Clustering(k = '+str(k)+')')
plt.show()
```

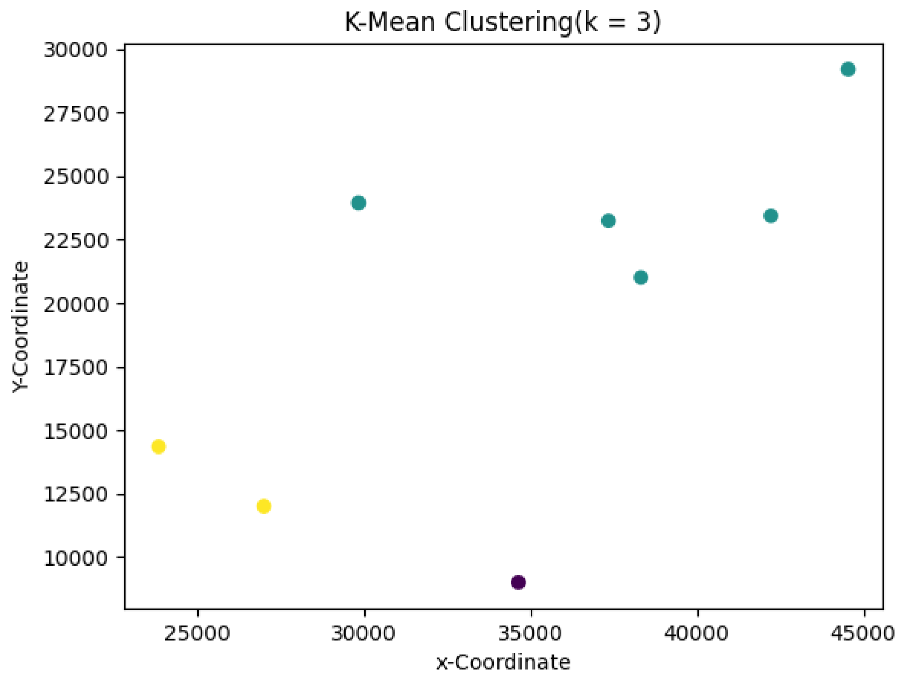
Output:



Code: Example 2

```
import numpy as np
import pandas as pd

#sample data representing flower location
data = pd.DataFrame({'Income':[27000,37344,23839,34640,38320,44545,42223,29843],
                     'Spending':[12000,23233,14343,9000,21000,29200,23432,23938]})
k = 3  # Predefine the cluster count
print("HITESH BHANSUAHLI KFMSCT005")
from sklearn.cluster import KMeans
kmean = KMeans(n_clusters=k)
kmean.fit(data)
# print(kmean.fit(data))
cluster_labels = kmean.predict(data)

import matplotlib.pyplot as plt
plt.scatter(data['Income'],data['Spending'],c= cluster_labels)
plt.xlabel("x-Coordinate")
plt.ylabel("Y-Coordinate")
plt.title('K-Mean Clustering(k = '+str(k)+')')
plt.show()
```

Output:



KISHINCHAND CHELLARAM COLLEGE, MUMBAI -20
M.Sc (I.T.) Part-1 Semester I

PRACTICAL NO 7:

Problem Statement: Apply decision tree classifier on the iris dataset and evaluate the model

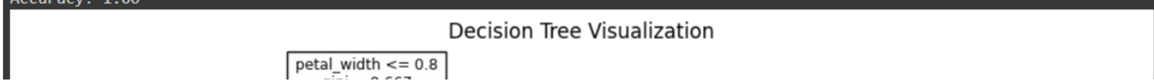
Aim: Using Iris dataset, plot a Decision Tree diagram for Species classification

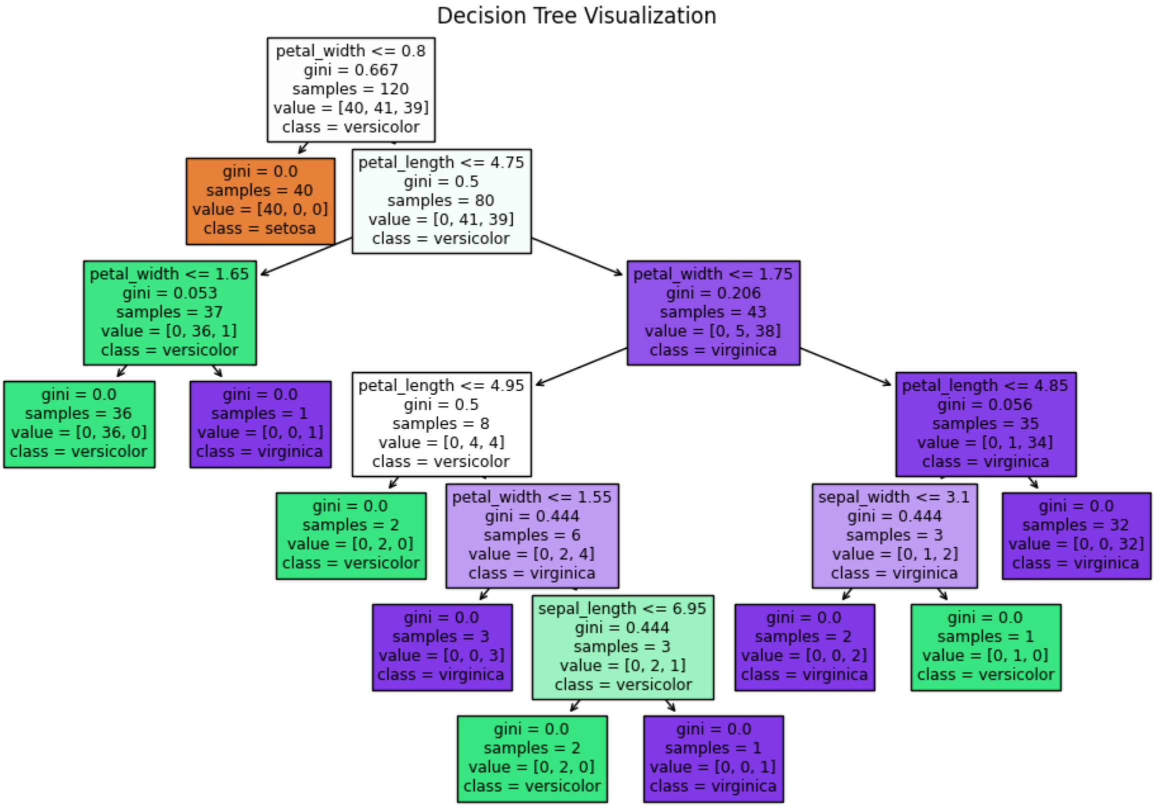
Code:

```
import numpy as np
import pandas as pd # Import Pandas for data loading
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
print("Hitesh Bhanushali KFMSCIT005")
data= pd.read_csv('/content/iris.csv')
print(data.head())
# Assuming the target variable is in a column named 'target'
X = data.drop('species', axis=1)
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
# Visualize and interpret the generated decision tree
plt.figure(figsize=(12, 8))
plot_tree(model, filled=True, feature_names=X.columns,
class_names=y.unique().astype(str))
plt.title("Decision Tree Visualization")
plt.show()
```

Output:

```
Hitesh Bhanushali KFMSCIT005
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
Accuracy: 1.00
```





Observation:

1. Imports necessary libraries such as pandas, matplotlib.pyplot, DecisionTreeClassifier, plot_tree, train_test_split, and accuracy_score.
2. Prints the author's name and student ID.
3. Reads the Iris dataset from a CSV file.
4. Displays the first few rows of the dataset.
5. Splits the dataset into features (X) and target variable (y).
6. Splits the data into training and testing sets using train_test_split().
7. Initializes a DecisionTreeClassifier model and fits it to the training data.
8. Makes predictions on the test data and calculates the accuracy score.
9. Visualizes the decision tree model using plot_tree() function.
10. Displays the decision tree visualization.

Conclusion:

1. The code loads the Iris dataset, splits it into training and testing sets, trains a Decision Tree classifier, predicts on the test set, calculates accuracy, and visualizes the decision tree structure.
2. It effectively demonstrates the use of a decision tree classifier for classification tasks on the Iris dataset.

PRACTICAL NO 8:

Problem Statement: Evaluate the performance of hierarchical clustering algo on a dataset using different evaluation metrics such as completeness_score, silhouette_score.

Aim: Evaluate the performance of the hierarchical clustering algo on the dataset using different evaluation matric such as completeness_score, silhouette score

Code:

```
import numpy as np
from sklearn.cluster import AgglomerativeClustering # specially used for Hierarchical clustering
from sklearn.metrics import completeness_score, silhouette_score

data = np.array([[1,1],[5,5],[8,8],[1,0],[5,4],[8,1]])
linkage='ward'
# linkage = 'ward' ## we have
model = AgglomerativeClustering(linkage='ward',n_clusters=3)
model.fit(data)
cluster_labels = model.labels_
s = silhouette_score(data,cluster_labels)
print(s)
g = None
if linkage =='ward' and g is not None:
    c = completeness_score(g, cluster_labels)
    print(c)
else:
    print('Not Applicable')
```

Output:

```
HITESH BHANUSHALI KFMSCIT005
0.4701798103378445
Not Applicable
```

Observation:
1. The code uses Agglomerative Clustering for hierarchical clustering.
2. It prints the author's name and student ID.
3. The dataset consists of six data points.
4. Agglomerative Clustering is applied with Ward linkage and three clusters.
5. Silhouette score is calculated to evaluate clustering quality.

Conclusion:
1. The code demonstrates hierarchical clustering using Agglomerative Clustering.
2. Silhouette score indicates clustering quality.
3. It checks for completeness score relevance but doesn't apply it universally due to specific conditions