J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD
(Autonomous)

KUKATPALLY, HYDERABAD – 500 085

Certificate

Certified that this is the bonafide record of the practical work done during

the academic year ______________________ by

Name ____

Roll Number _______________________ *Class* __________________________________

in the Laboratory of ____

of the Department of ____

Signature of the Staff Member

Signature of the Head of the Department

*Date of Examination*_________________________

Signature of the Examiner/s

Internal Examiner

External Examiner

J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD
(Autonomous)

KUKATPALLY, HYDERABAD – 500 085

Name ________________________________ *Roll Number* ________________________

Class __________________ *Year*____________ *Laboratory* _________________________

List of Experiments

S.No.	Name of the Experiment	Date of Experiment	Page Number	Marks	Remarks

1. **Write a python program to compute Central Tendency Measures: Mean, Median, Mode Measure of Dispersion: Variance, Standard Deviation**

Code

```
import statistics
# Sample data
data = [12, 15, 21, 15, 19, 15, 18, 21, 17]
# Central Tendency Measures
mean = statistics.mean(data)
median = statistics.median(data)
mode = statistics.mode(data)
# Measures of Dispersion
variance = statistics.variance(data)
std_deviation = statistics.stdev(data)
# Display results
print("Central Tendency Measures:")
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Mode: {mode}")
print("\nMeasures of Dispersion:")
print(f"Variance: {variance}")
print(f"Standard Deviation: {std_deviation}")
```

OUTPUT

```
Central Tendency Measures:
Mean: 17
Median: 17
Mode: 15
Measures of Dispersion:
Variance: 9.25
Standard Deviation: 3.0413812651491097
```

2. Study of Python Basic Libraries such as Statistics, Math, Numpy and Scipy

1. **statistics Library**

```
import statistics
statistics.mean([1, 2, 3])        # Arithmetic mean
statistics.median([1, 3, 2])      # Middle value
statistics.mode([1, 1, 2])        # Most common value
statistics.variance([4, 5, 7])    # Sample variance
statistics.stdev([4, 5, 7])       # Sample standard deviation
```

2. **math Library**

```
import math
math.sqrt(16)              # Square root
math.factorial(5)          # Factorial
math.pow(2, 3)             # Power (2^3)
math.pi                    # Value of π
math.sin(math.radians(30)) # Sine of 30 degrees
```

3. **NumPy Library**

```
import numpy as np
np.array([1, 2, 3])          # Create NumPy array
np.mean([1, 2, 3])           # Mean of array
np.std([1, 2, 3])            # Standard deviation
np.linspace(0, 1, 5)         # Evenly spaced values
np.dot([1, 2], [3, 4])       # Dot product
```

3. Study of Python Libraries for ML application such as Pandas and Matplotlib

Pandas

```
import pandas as pd
pd.read_csv("iris.csv")          # Load data from a CSV file
df.head()                        # View the first 5 rows of a DataFrame
df.describe()                    # Summary statistics
df.dropna()                      # Remove missing values
```

OUTPUT

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

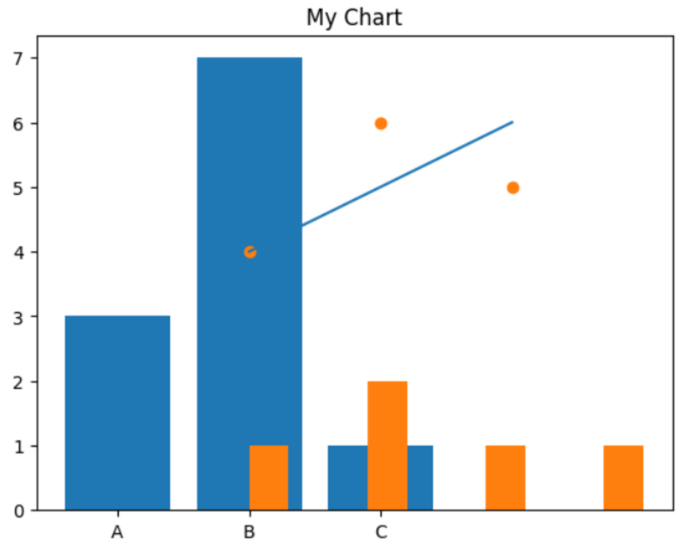
150 rows × 6 columns

Matplotlib

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])      # Line plot
plt.bar(['A', 'B', 'C'], [3, 7, 1]) # Bar chart
plt.hist([1, 2, 2, 3, 4])        # Histogram
plt.scatter([1, 2, 3], [4, 6, 5])   # Scatter plot
plt.title("My Chart")            # Add title to the plot
```

OUTPUT

```
Text(0.5, 1.0, 'My Chart')
```



4. Write a Python program to implement Simple Linear Regression

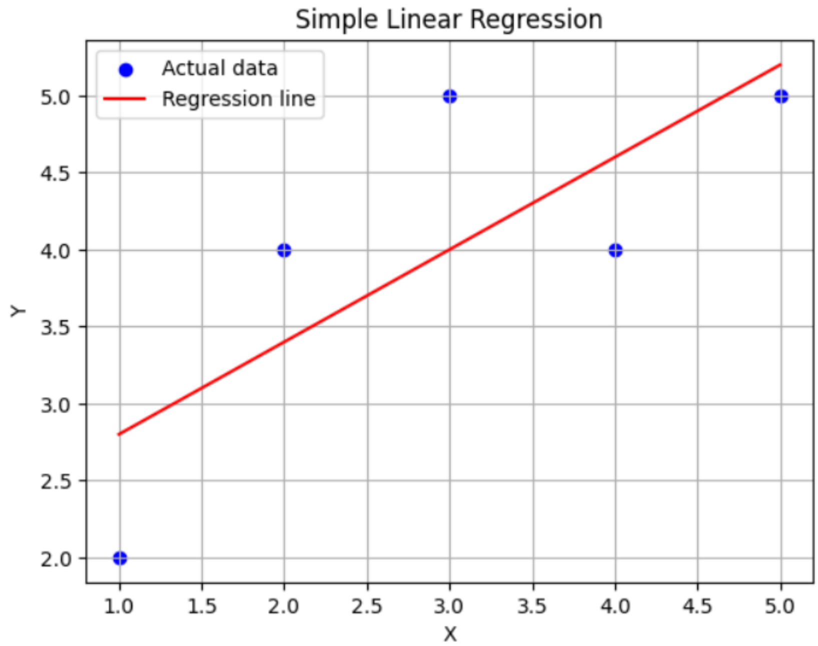
Code

```
import numpy as np
```

```
import matplotlib.pyplot as plt
# Sample data
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 5, 4, 5])
# Calculate means
x_mean = np.mean(x)
y_mean = np.mean(y)
# Calculate coefficients
numerator = np.sum((x - x_mean) * (y - y_mean))
denominator = np.sum((x - x_mean)**2)
slope = numerator / denominator
intercept = y_mean - slope * x_mean
# Predict values
y_pred = slope * x + intercept
# Output model parameters
print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
# Visualization
plt.scatter(x, y, color='blue', label='Actual data')
plt.plot(x, y_pred, color='red', label='Regression line')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Simple Linear Regression')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT

```
Slope (m): 0.6
Intercept (b): 2.2
```



5. **Implementation of Multiple Linear Regression for House Price Prediction using sklearn**

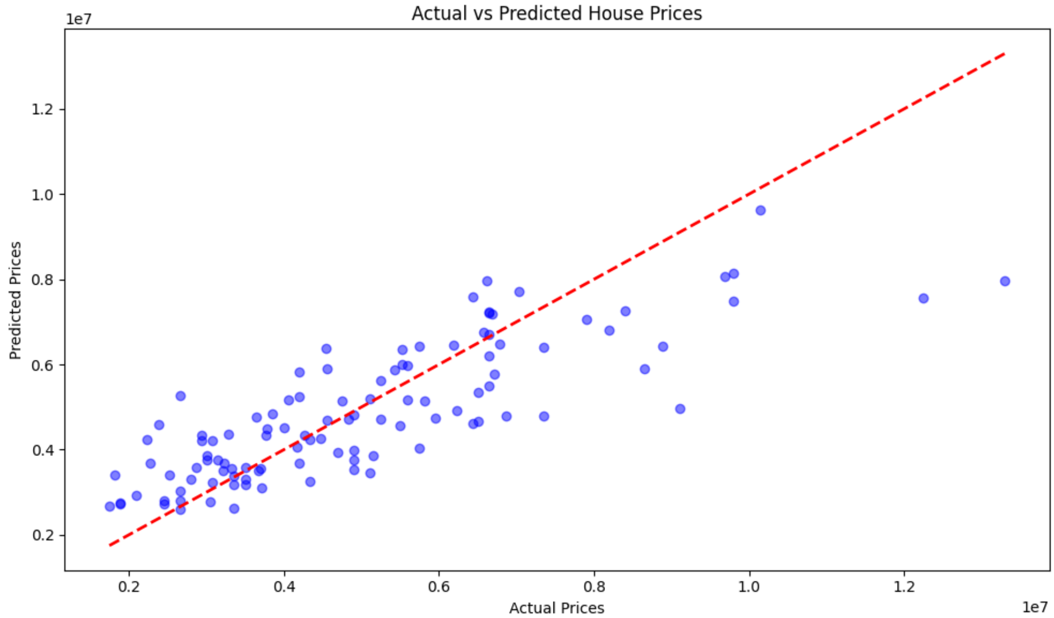
<u>Code</u>

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
# Load the dataset
data = pd.read_csv('Housing.csv')
# Define feature columns and target
numerical_features = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking']
categorical_features = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
'prefarea', 'furnishingstatus']
target_column = 'price'
# Handle missing values (drop rows with missing values in relevant columns)
data = data.dropna(subset=numerical_features + categorical_features + [target_column])
# Encode categorical variables
data = pd.get_dummies(data, columns=categorical_features, drop_first=True)
```

```
# Prepare features (X) and target (y)
feature_columns = numerical_features + [col for col in data.columns if col.startswith(tuple(categorical_features)) and col != target_column]
X = data[feature_columns]
y = data[target_column]
# Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)
# Make predictions
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 model performance
print(f'Mean Squared Error: {mse:.2f}')
print(f'R² Score: {r2:.2f}')
print('Coefficients:', dict(zip(feature_columns, model.coef_)))
print('Intercept:', model.intercept_)
# Visualize actual vs predicted prices
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted House Prices')
plt.tight_layout()
plt.show()
```

<u>OUTPUT</u>

```
Mean Squared Error: 1754318687330.67
R² Score: 0.65
Coefficients: {'area': 511615.56377665815, 'bedrooms': 56615.572457787, 'bathrooms': 549420.501240978, 'stories': 353158.42985603586, 'parking': 193542.7816745456, 'mainroad_yes': 128151.9212953322, 'guestroom_yes': 88590.21346152117, 'basement_yes': 186194.1505056636, 'hotwaterheating_yes': 143233.20624958424, 'airconditioning_yes': 367817.89491558215, 'prefarea_yes': 267018.6608123931, 'furnishingstatus_semi-furnished': -62550.29721128263, 'furnishingstatus_unfurnished': -193987.7810882041}
Intercept: 4737518.175380117
```



6. Implementation of Decision tree using sklearn and its parameter tuning

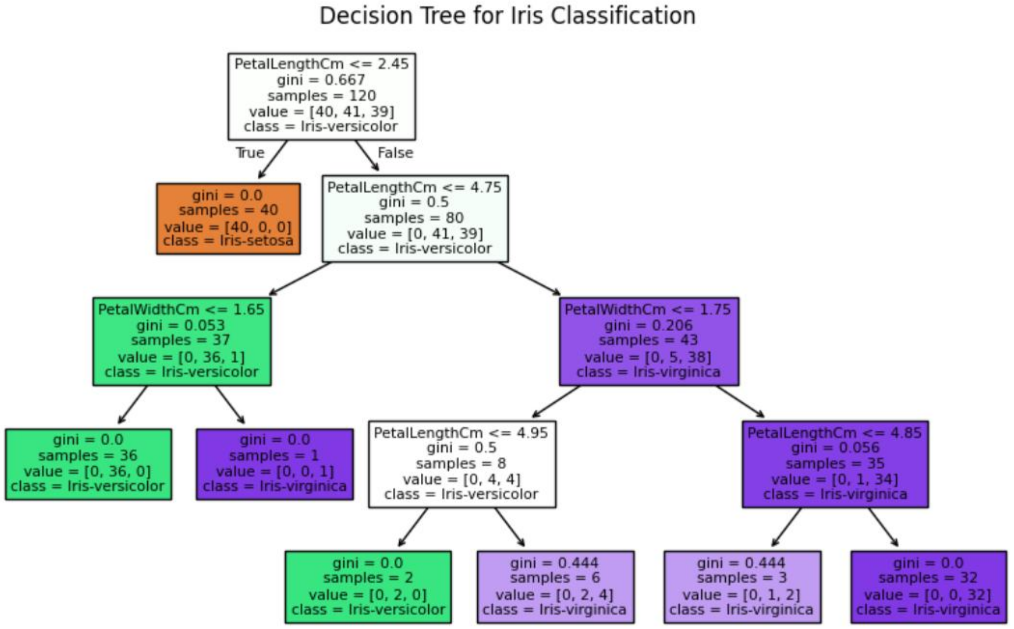
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
# Load the dataset
df = pd.read_csv('Iris.csv')
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = df['Species']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Perform GridSearchCV
grid = GridSearchCV(DecisionTreeClassifier(random_state=42), {
    'max_depth': [2, 3, 4, None],
    'min_samples_split': [2, 3, 4]
}, cv=5).fit(X_train, y_train)
# Get the best model
best_model = grid.best_estimator_
```

```
print(f'Best Parameters: {grid.best_params_}')
# Predict class probabilities for MSE and R² calculation
y_pred_proba = best_model.predict_proba(X_test)
# Convert true labels to one-hot encoding for probability-based MSE and R²
y_test_one_hot = pd.get_dummies(y_test).values
mse = mean_squared_error(y_test_one_hot, y_pred_proba)
r2 = r2_score(y_test_one_hot, y_pred_proba)
# Print evaluation metrics
print(f'Mean Squared Error (based on probabilities): {mse:.4f}')
print(f'R² Score (based on probabilities): {r2:.4f}')
# Visualize the decision tree
plt.figure(figsize=(10, 6))
plot_tree(best_model,feature_names=X.columns,class_names=best_model.classes_, filled=True)
plt.title('Decision Tree for Iris Classification')
plt.show()
```

OUTPUT

```
Mean Squared Error (based on probabilities): 0.0000
R² Score (based on probabilities): 1.0000
```



7. Implementation of KNN using sklearn

Code

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
# Load the dataset
df = pd.read_csv('Iris.csv')
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = df['Species']
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
# Initialize KNN classifier
knn = KNeighborsClassifier()
# Define parameter grid for tuning
param_grid = {'n_neighbors': [3, 5, 7, 9, 11]}
# Perform GridSearchCV
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
# Get the best model
best_knn = grid_search.best_estimator_
print(f'Best Parameters: {grid_search.best_params_}')
# Make predictions
y_pred = best_knn.predict(X_test)
# Calculate accuracy
```

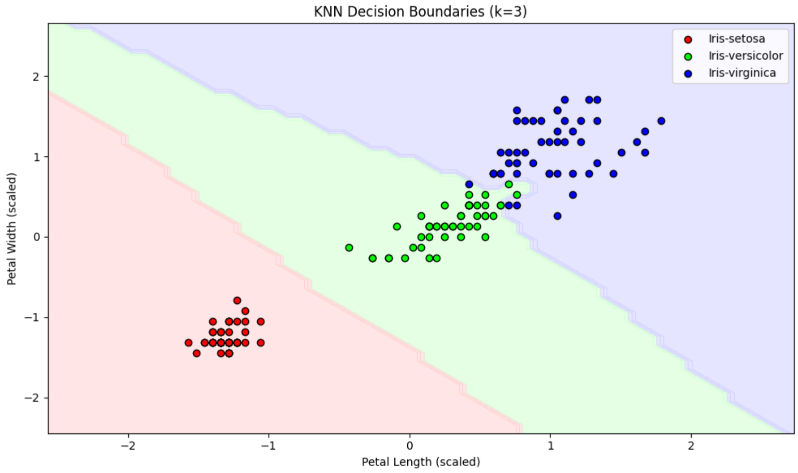
```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.4f}')
# Calculate MSE and R² based on predicted probabilities
y_pred_proba = best_knn.predict_proba(X_test)
y_test_one_hot = pd.get_dummies(y_test).values
mse = mean_squared_error(y_test_one_hot, y_pred_proba)
r2 = r2_score(y_test_one_hot, y_pred_proba)
print(f'Mean Squared Error (based on probabilities): {mse:.4f}')
print(f'R² Score (based on probabilities): {r2:.4f}')

# Visualize decision boundaries using two features (PetalLengthCm and PetalWidthCm)
X_subset = X_scaled[:, [2, 3]]  # PetalLengthCm and PetalWidthCm
X_train_subset = X_train[:, [2, 3]]
X_test_subset = X_test[:, [2, 3]]
# Train KNN on the subset for visualization
knn_subset = KNeighborsClassifier(n_neighbors=grid_search.best_params_['n_neighbors'])
knn_subset.fit(X_train_subset, y_train)
# Create mesh grid for decision boundary
x_min, x_max = X_subset[:, 0].min() - 1, X_subset[:, 0].max() + 1
y_min, y_max = X_subset[:, 1].min() - 1, X_subset[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max, 0.1))
Z = knn_subset.predict(np.c_[xx.ravel(), yy.ravel()])
Z = pd.Categorical(Z, categories=best_knn.classes_).codes
Z = Z.reshape(xx.shape)
# Plot decision boundaries
plt.figure(figsize=(10, 6))
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ['#FF0000', '#00FF00', '#0000FF']
plt.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.3)
for idx, species in enumerate(best_knn.classes_):
    plt.scatter(X_subset[y == species, 0], X_subset[y == species, 1],
                c=cmap_bold[idx], label=species, edgecolor='k')
plt.xlabel('Petal Length (scaled)')
plt.ylabel('Petal Width (scaled)')
plt.title(f'KNN Decision Boundaries (k={grid_search.best_params_["n_neighbors"]})')
plt.legend()
```

```
plt.tight_layout()
plt.show()
```

OUTPUT

```
Best Parameters: {'n_neighbors': 3}
Accuracy: 1.0000
Mean Squared Error (based on probabilities): 0.0049
R² Score (based on probabilities): 0.9776
```



8. Implementation of Logistic Regression using sklearn

Code

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
# Load dataset
df = pd.read_csv('Iris.csv')
# Select features and target
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = df['Species']
# Train-test split
```

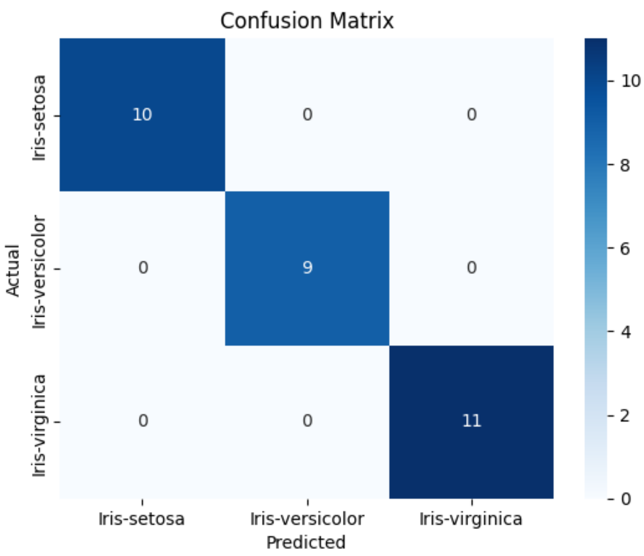
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize and train the Logistic Regression model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
# Predict on the test set
y_pred = model.predict(X_test)
# Evaluation
print("Classification Report:")
print(classification_report(y_test, y_pred))
# Confusion matrix visualization
conf_matrix = confusion_matrix(y_test, y_pred, labels=model.classes_)
sns.heatmap(conf_matrix, annot=True, fmt='d', xticklabels=model.classes_,
yticklabels=model.classes_, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

<u>OUTPUT</u>

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

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       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
```



9. Implementation of K-Means Clustering

<u>Code</u>

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
# Load and preprocess data
df = pd.read_csv('Iris.csv')
df.columns = df.columns.str.strip().str.lower()
X = df[['petallengthcm', 'petalwidthcm']]
# Scale and cluster
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=3, random_state=42).fit(X_scaled)
# Plot clusters and centroids
plt.figure(figsize=(8, 6))
plt.scatter(X['petallengthcm'], X['petalwidthcm'], c=kmeans.labels_, cmap='viridis', s=50)
centers = scaler.inverse_transform(kmeans.cluster_centers_)
plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='x', s=200, linewidths=3)
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('K-Means Clustering of Iris Petal Features')
plt.show()
```

OUTPUT