TOPIC: IRIS FLOWER CLASSIFICATION

NAME: YASH VARSHNEY

UNIVERSITY ROLL NO. :202401100400217

INTRODUCTION

The **Iris dataset** is one of the most famous and widely used datasets in machine learning. It contains **measurements of sepal length, sepal width, petal length, and petal width** for three different species of iris flowers:

- **Iris Setosa**
- **Iris Versicolor**
- **Iris Virginica**

The main objective of this project is to develop a **machine learning model** that can classify iris flowers based on their given measurements. By using classification techniques, we can automate the process of species identification, which can be useful in fields like **botany, agriculture, and environmental science**.

This project uses **Logistic Regression**, a widely used classification algorithm, to train a model that can accurately predict the species of an iris flower based on its physical attributes.

METHODOLOGY

The methodology follows a structured approach, including data collection, preprocessing, visualization, model training, evaluation, and prediction.

1.Dataset Used

- It contains 150 samples with equal distribution among three species.
- The dataset includes four key features:
  - Sepal Length (cm)
  - Sepal Width (cm)
  - Petal Length (cm)
  - Petal Width (cm)

2. Steps Followed

1. Data Loading & Preprocessing

- The dataset was loaded from scikit-learn (sklearn.datasets).
- Converted the dataset into a pandas DataFrame for easy manipulation.
- The species column was mapped from numerical values (0, 1, 2) to their respective species names (Setosa, Versicolor, Virginica).
- Checked for missing values and shuffled the dataset to ensure a variety of samples in training and testing.

2. Exploratory Data Analysis (EDA)

- Visualized species distribution using count plots to understand the data balance.
- Used pair plots to observe feature relationships and differences between species.
- Identified feature importance in distinguishing the three iris species.

3. Data Standardization

- Used StandardScaler to scale the dataset.
- Standardization helps improve model performance by ensuring all features have the same scale.

4. Model Training & Testing

- The dataset was split into 80% training and 20% testing to evaluate model performance.
- Logistic Regression was chosen for classification due to its simplicity and effectiveness in multi-class classification.
- The model was trained on the training dataset and tested on the test dataset.

5. Model Evaluation

- Used Accuracy Score to measure model performance.
- Evaluated results using a Confusion Matrix to analyze misclassifications.
- Used a Classification Report to assess Precision, Recall, and F1-score for each species.

6. Prediction on New Data

- Provided an example input with sepal and petal measurements.
- The trained model was used to classify the species of a new iris flower.

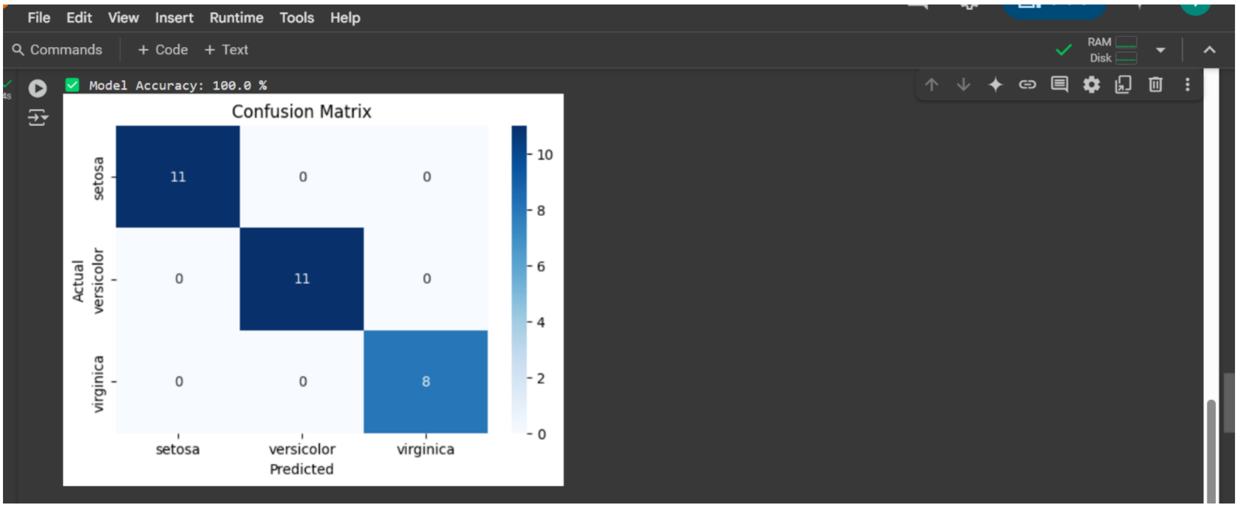
CODE

```
# Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets  # To load the Iris dataset
from sklearn.model_selection import train_test_split  # Splitting data
from sklearn.preprocessing import StandardScaler  # Feature scaling
from sklearn.linear_model import LogisticRegression  # ML model
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix  # Evaluation

# ---------------- Step 1: Load and Prepare Dataset ----------------
# Load dataset from sklearn (No need to download manually)
iris = datasets.load_iris()

# Convert dataset into a pandas DataFrame
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

# Add species labels (0, 1, 2) to the DataFrame
df["species"] = iris.target

# Convert numerical species values into actual names
df["species"] = df["species"].map({0: "Setosa", 1: "Versicolor", 2:
"Virginica"})

# Shuffle the dataset to ensure variety in display
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

# Display first 5 shuffled rows
print("📌 Sample Data (Shuffled):")
print(df.head())

# ---------------- Step 2: Data Exploration ----------------
# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())

# Display statistical summary
print("\nData Summary:\n", df.describe())

# ---------------- Step 3: Data Visualization ----------------
# Count plot of species distribution
plt.figure(figsize=(6, 4))
sns.countplot(x="species", data=df, palette="viridis")
```

```
plt.title("Iris Species Distribution")
plt.xlabel("Species")
plt.ylabel("Count")
plt.show()

#  Pairplot to visualize relationships between features
sns.pairplot(df, hue="species", palette="husl")
plt.show()

# ---------------- Step 4: Prepare Data for Model Training --------------
--
# Separate features (X) and target labels (y)
X = df.iloc[:, :-1]  # All columns except "species"
y = df["species"]  # Target variable (species)

# Split data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# ---------------- Step 5: Standardizing the Features ----------------
# StandardScaler helps normalize the data for better model performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)  # Fit & transform training data
X_test = scaler.transform(X_test)  # Transform test data (without
fitting)

# ---------------- Step 6: Train the Logistic Regression Model ----------
------
# Initialize and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# ---------------- Step 7: Model Evaluation ----------------
# Predict species on the test dataset
y_pred = model.predict(X_test)

#  Accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("\n Model Accuracy:", round(accuracy * 100, 2), "%")

#  Confusion matrix visualization
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d",
xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

```
plt.show()

#  Classification report (Precision, Recall, F1-score)
print("\n Classification Report:\n", classification_report(y_test,
y_pred))

# ---------------- Step 8: Predict New Flower Species ----------------
# Define a new sample with petal & sepal measurements
new_sample = pd.DataFrame([[5.1, 3.5, 1.4, 0.2]], columns=X.columns)

# Scale the new sample using the same StandardScaler
new_sample_scaled = scaler.transform(new_sample)

# Predict the species of the new flower
predicted_species = model.predict(new_sample_scaled)

# Print the predicted species
print("\n🌸 Predicted Species for New Sample:", predicted_species[0])
```

OUTPUT



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

      Setosa       1.00      1.00      1.00        11
  Versicolor       1.00      1.00      1.00        11
   Virginica       1.00      1.00      1.00         8

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


🌸 Predicted Species for New Sample: Setosa
```