NAME : DIVYA AC

USN : 1SV21CS029

TEAM 09

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt

data = pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv')

df = pd.DataFrame(data)

print(df.columns)
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm',
       'Species'],
      dtype='object')
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

```python
data = pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv', sep=';')

df = df.reset_index()

print(df.columns)
```

```
Index(['index', 'Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
       'PetalWidthCm', 'Species'],
      dtype='object')
```

```python
import pandas as pd

# Assuming your CSV file has ';' as the delimiter
data = pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv', sep=';')

# Print out the column names to verify
print(data.columns)
```

```
Index(['Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species'],
dtype='object')
```

```python
# Mount Google Drive if you're using Google Colab
from google.colab import drive
```

```python
drive.mount('/content/drive')

import pandas as pd

# Path to your CSV file in Google Drive
csv_file_path = '/content/drive/MyDrive/archive (10)/iris.csv'

# Reading the CSV file; adjust the separator based on your file
data = pd.read_csv(csv_file_path, sep=',')

# Verify the column names and data
print(data.head())

# Select only two specific columns for independent variables (X)
X = data[['SepalLengthCm', 'SepalWidthCm']]  # Independent variables
(features)

# Dependent variable (target)
y = data['Species']  # Assuming 'Species' is the target variable

# Print first few rows of X and y to verify
print("Features (X):")
print(X.head())

print("\nTarget variable (y):")
print(y.head())
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
   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
Features (X):
   SepalLengthCm  SepalWidthCm
0            5.1           3.5
1            4.9           3.0
2            4.7           3.2
3            4.6           3.1
4            5.0           3.6

Target variable (y):
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
```

```
4    Iris-setosa
Name: Species, dtype: object
```

```
k=3
knn=KNeighborsClassifier(n_neighbors=k)
knn.fit(X,y)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
new_data = np.array([[5.1,3.5]])
prediction = knn.predict(new_data)
if prediction[0] == 1:
    print("Iris-setose")
else:
   print("Iris-versicolour")
```

```
Iris-versicolour
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X
does not have valid feature names, but KNeighborsClassifier was fitted with
feature names
  warnings.warn(
```

```
# Load the dataset
file_path = ('/content/drive/MyDrive/archive (10)/iris.csv')
# Update this path accordingly
df = pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv')

# Define independent variables (features) and dependent variable
X = df[['SepalLengthCm', 'SepalWidthCm']]
y = df['Species']  # Assuming 'Species' is the target variable

# Encode the target variable 'Species' into numerical values
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# 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 the Linear Regression model (for demonstration, though not ideal for
this problem)
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
LinearRegression()

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

```
# Evaluate the model (these metrics might not be appropriate for categorical
predictions)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)

# For classification problems, consider using Logistic Regression, Decision
Trees, or other suitable algorithms.
```

```
Mean Squared Error: 0.16908805917847763
R-squared: 0.7580616005395391
```

```
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Coefficients: [ 0.72039588 -0.66538649]
Intercept: -1.1588138946175666
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
# Load dataset
df =pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv')
# Define independent variables (features) and dependent variable
# (target)  # This line seems to be an unintended command, commenting it out
X = df[['SepalLengthCm', 'SepalWidthCm']]
y = df['Species']

# Encode the target variable 'Species' into numerical values
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y) # Convert target to numerical

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Initialize k-NN regressor (set k=3 for example)
k = 3
knn_regressor = KNeighborsRegressor(n_neighbors=k)
# Train the model
knn_regressor.fit(X_train_scaled, y_train)
# Predict on the test set
y_pred = knn_regressor.predict(X_test_scaled) # Now y_pred will be numerical
# Evaluate performance (e.g., using RMSE and R^2)
```

```python
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print(rmse)
print(r2)
```

```
0.3751542892474251
0.798622151563328
```

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score

# Load dataset
df =pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv')
# Display the first few rows to understand the structure
print(df.head())
```

```
   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
```

```python
X = df[['SepalLengthCm', 'SepalWidthCm']]
y = df['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Initialize logistic regression model
log_reg = LogisticRegression(random_state=42)
# Train the model
log_reg.fit(X_train_scaled, y_train)
LogisticRegression(random_state=42)
# Predict on the test set
y_pred = log_reg.predict(X_test_scaled)
# Evaluate performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{class_report}')
```

```
Accuracy: 0.9
Confusion Matrix:
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       0.88      0.78      0.82         9
 Iris-virginica       0.83      0.91      0.87        11

       accuracy                           0.90        30
      macro avg       0.90      0.90      0.90        30
   weighted avg       0.90      0.90      0.90        30
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
# Load dataset
df =pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv')
# Display the first few rows to understand the structure
print(df.head())
```

```
   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
```

```python
# ... (rest of your code)

X = df[['SepalLengthCm', 'SepalWidthCm']]
y = df['Species']

# Encode the target variable 'Species' into numerical values
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y) # Convert target to numerical

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

# Initialize decision tree regressor
dt_regressor = DecisionTreeRegressor(random_state=42)
```

```
# Train the model
dt_regressor.fit(X_train, y_train)
# ... (rest of your code)
```

```
DecisionTreeRegressor(random_state=42)
```

```
y_pred = dt_regressor.predict(X_test)
# Evaluate performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

```
Mean Squared Error (MSE): 0.29814814814814816
R-squared (R2): 0.5733969263381027
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
# Load dataset
df =pd.read_csv('/content/drive/MyDrive/archive (10)/iris.csv')
# Display the first few rows to understand the structure
print(df.head())
```

```
   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
```

```
# ... (rest of your code)

X = df[['SepalLengthCm', 'SepalWidthCm']]
y = df['Species']

# Encode the target variable 'Species' into numerical values
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y) # Convert target to numerical

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

# Initialize decision tree regressor
rf_regressor = RandomForestRegressor(random_state=42)
```

```python
# Train the model
rf_regressor.fit(X_train, y_train)
# ... (rest of your code)
```

```
RandomForestRegressor(random_state=42)
```

```python
y_pred = rf_regressor.predict(X_test)
# Evaluate performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

```
Mean Squared Error (MSE): 0.192646804138322
R-squared (R2): 0.7243527444761688
```