AI FINAL PROJECT EXPLANATION

The explanation of the AI FINAL PROJECT CODE

FOR THE CODE BELOW : import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

data = pd.read\_csv("accelerometer\_gyro\_mobile\_phone\_dataset.csv")

# Drop the timestamp column as it's not needed for modeling

data = data.drop(columns=['timestamp'])

# Split the data into features and target

X = data.drop(columns=['Activity'])

X.info()

X.head()

X.describe()

print("")

print("")

y = data['Activity']

y.info()

y.head()

y.describe()

print("")

print("")

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Scale the features for KNN

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Initialize the models

knn = KNeighborsClassifier()

decision\_tree = DecisionTreeClassifier(random\_state=42)

naive\_bayes = GaussianNB()

# Train and evaluate KNN

knn.fit(X\_train\_scaled, y\_train)

knn\_predictions = knn.predict(X\_test\_scaled)

knn\_accuracy = accuracy\_score(y\_test, knn\_predictions)

# Train and evaluate Decision Tree

decision\_tree.fit(X\_train, y\_train)

decision\_tree\_predictions = decision\_tree.predict(X\_test)

decision\_tree\_accuracy = accuracy\_score(y\_test, decision\_tree\_predictions)

# Train and evaluate Naive Bayes

naive\_bayes.fit(X\_train, y\_train)

naive\_bayes\_predictions = naive\_bayes.predict(X\_test)

naive\_bayes\_accuracy = accuracy\_score(y\_test, naive\_bayes\_predictions)

knn\_accuracy, decision\_tree\_accuracy, naive\_bayes\_accuracy

THE EXPLANATION :

Importing Libraries: The code imports necessary libraries for data manipulation (pandas), machine learning algorithms (scikit-learn), data preprocessing, model selection, and evaluation.

Loading Data: Your dataset, which contains accelerometer and gyroscope data collected from mobile phones for human activity recognition, is loaded into a pandas DataFrame named data.

Data Preprocessing:

The 'timestamp' column is dropped as it's not relevant for the classification task.

Features (accelerometer and gyroscope readings) are separated from the target variable (activity: standing or walking).

Information about the features and target variable, such as data types, head (top rows), and summary statistics, is printed to understand the structure and distribution of the data.

Train-Test Split: The data is split into training and testing sets with a 70-30 ratio. This allows for training the models on one subset and evaluating their performance on another independent subset.

Feature Scaling: Since K-Nearest Neighbors (KNN) relies on distance metrics, the features are scaled using StandardScaler to ensure they have the same scale. This step is important for achieving optimal performance from the KNN algorithm.

Model Initialization:

Three classifiers are initialized: K-Nearest Neighbors (KNN), Decision Tree, and Naive Bayes. These classifiers will be trained and evaluated using the data.

Model Training and Evaluation:

KNN: The KNN classifier is trained on the scaled training data and evaluated on the scaled testing data. Its accuracy score indicates how well it predicts the activity classes (standing or walking) based on accelerometer and gyroscope readings.

Decision Tree: The Decision Tree classifier is trained on the original training data (without scaling) and evaluated on the original testing data. Its accuracy score reflects its performance in predicting activity classes.

Naive Bayes: The Naive Bayes classifier is trained on the original training data and evaluated on the original testing data. Its accuracy score indicates how well it predicts activity classes based on the provided data.

Model Accuracy: The accuracy scores of all three models (KNN, Decision Tree, Naive Bayes) are printed. These scores provide insights into how well each model performs in classifying activities (standing or walking) based on accelerometer and gyroscope data from mobile phones.

In summary, this code performs machine learning classification tasks on your dataset to predict human activities (standing or walking) using accelerometer and gyroscope data collected from mobile phones. It trains and evaluates three different classifiers and provides accuracy scores to assess their performance.

FOR THE CODE OF THE MULTIVARIATE DATA ANALYSIS

import warnings

warnings.filterwarnings("ignore")

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

# Summary statistics

data = pd.read\_csv('data/accelerometer\_gyro\_mobile\_phone\_dataset.csv')

print(data.describe())

# SELECT ONLY NUMERIC COLUMNS

data\_numeric = data.select\_dtypes(include=[np.number])

# COMPUTE THE CORRELATION MATRIX

corr = data\_numeric.corr()

# PLOT THE CORRELATION MATRIX

sns.heatmap(corr, annot=True, cmap='coolwarm')

# PAIRPLOT

sns.pairplot(data)

# SCATTER MATRIX

pd.plotting.scatter\_matrix(data, figsize=(10, 10))

#DISPLAY THE CORRELATION MATRIX

print(corr)

THE EXPLANATION :

Importing Libraries: Necessary libraries are imported, including pandas for data manipulation, scikit-learn for machine learning, seaborn and matplotlib for visualization, and numpy for numerical computations.

Loading Data: Your dataset is loaded into a pandas DataFrame named data.

Summary Statistics:

data.describe(): This method computes summary statistics of the numerical columns in your dataset, such as count, mean, standard deviation, minimum, and maximum values. It gives you an overview of the distribution of your data.

Selecting Numeric Columns:

data.select\_dtypes(include=[np.number]): This selects only the numeric columns from your dataset. This step is crucial for computing correlations and visualizing relationships between variables.

Computing the Correlation Matrix:

The correlation matrix (corr) is computed using the .corr() method on the numeric data. This matrix shows the pairwise correlations between all pairs of numeric variables in your dataset.

Plotting the Correlation Matrix:

sns.heatmap(corr, annot=True, cmap='coolwarm'): This plots a heatmap of the correlation matrix, where each cell's color represents the correlation coefficient between two variables. The annot=True parameter displays the correlation values in each cell.

Pairplot:

sns.pairplot(data): This creates a grid of scatterplots showing the pairwise relationships between variables. Each scatterplot represents the relationship between two variables, and the diagonal shows the distribution of each variable.

Scatter Matrix:

pd.plotting.scatter\_matrix(data, figsize=(10, 10)): This plots a matrix of scatterplots, where each scatterplot represents the relationship between two variables. It's similar to the pairplot but allows for more customization of the plot layout.

Displaying the Correlation Matrix:

print(corr): This simply displays the correlation matrix in tabular format, showing the numeric values of the correlations between variables.

Explanation:

These multivariate data analysis steps provide insights into the relationships between variables in your dataset. The summary statistics give an overview of the distribution of each variable, while the correlation matrix, heatmap, pairplot, and scatter matrix help visualize the relationships and dependencies between variables.

By examining these visualizations and numerical summaries, you can gain a deeper understanding of how the accelerometer and gyroscope readings relate to each other and how they might influence the prediction of human activities (standing or walking) using machine learning models.