Disclaimer and caution: This script/file was generated via generative AI

Thursday February 13, 2025, without prejudice

Prompt:

Write python code that can perform the following :

1. Perform standard scaling operation on the top 4 attributes identified in the previous code.
2. Train and test this with the most common classifiers on the data and provide a comparative table on their accuracy and performance

Response:

Certainly! Below is a Python code snippet that performs standard scaling on the top 5 attributes of a dataset, trains various classifiers (Logistic Regression, KNN, Naive Bayes, Decision Trees, Random Forests, and Multi-layer Perceptron), and prints a comparative table of their accuracy performance.

Make sure you have the necessary libraries installed. You can install them using pip if you haven't done so already: bash

```
pip install pandas scikit-learn
```

Here is the code:

```python

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neural_network import MLPClassifier

from sklearn.metrics import accuracy_score


# Load your dataset
# Assume df is your DataFrame and 'target' is the name of the target column

# df = pd.read_csv('your_dataset.csv')


# For demonstration, let's create a synthetic dataset

from sklearn.datasets import make_classification

df, target = make_classification(n_samples=1000, n_features=20, n_informative=5,
n_redundant=0, random_state=42)

df = pd.DataFrame(df)

df['target'] = target


# Select the top 5 attributes (for this example, we will just use the first 5
columns)

X = df.iloc[:, :5]

y = df['target']


# 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)


# Perform standard scaling

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)


# Initialize classifiers

classifiers = {

    'Logistic Regression': LogisticRegression(max_iter=1000),

    'KNN': KNeighborsClassifier(),

    'Naive Bayes': GaussianNB(),

    'Decision Trees': DecisionTreeClassifier(),

    'Random Forests': RandomForestClassifier(),
    'Multi-layer Perceptron': MLPClassifier(max_iter=1000)

}


# Train classifiers and evaluate their performance

accuracy_results = {}


for name, clf in classifiers.items():

    clf.fit(X_train_scaled, y_train)

    y_pred = clf.predict(X_test_scaled)

    accuracy = accuracy_score(y_test, y_pred)

    accuracy_results[name] = accuracy


# Create a comparative table for accuracy performance

accuracy_df = pd.DataFrame(list(accuracy_results.items()), columns=['Classifier',
'Accuracy'])

print(accuracy_df.sort_values(by='Accuracy', ascending=False))
```

Explanation:

Data Preparation: The code assumes you have a DataFrame df with features and a target column. For demonstration, a synthetic dataset is created using make_classification. Feature Selection: The first 5 columns are used as features (you can modify this to select your actual top 5 attributes). Data Splitting: The dataset is split into training and testing sets. Standard Scaling: Standard scaling is performed on the features. Classifier Training: Six classifiers are trained and their accuracy is computed. Performance Table: A DataFrame is created to display the accuracy of each classifier, sorted in descending order.

You can replace the synthetic dataset creation with your actual dataset loading and modify the feature selection as needed.

Results:

```
               Classifier  Accuracy
2             Naive Bayes     0.505
3          Decision Trees     0.500
4          Random Forests     0.475
1                     KNN     0.465
0     Logistic Regression     0.460
5  Multi-layer Perceptron     0.460
```