Classify Students Based on Study Methods

A Project Report

Submitted by

RAJA YADAV

University Roll no.: 202401100300193

In partial fulfillment for the award of the degree

Bachelor of Technology

(CSE-AI)

22-April-2025

Introduction to the Problem Statement

Understanding how students learn is crucial for designing effective educational experiences. Students typically exhibit different learning preferences, which are commonly categorized into three main styles: Visual, Auditory, and Kinesthetic. By analyzing questionnaire responses that measure these learning preferences, we can classify students into appropriate learning style categories.

The purpose of this project is to build a machine learning model that can accurately classify students based on their learning preferences. The input data includes scores for visual, auditory, and kinesthetic attributes for each student, along with the actual learning style label. Using this data, a classification model is trained to predict the learning style based on the scores.

This approach can be applied to personalize education, adapt teaching methods to individual needs, and enhance student engagement and learning outcomes.

Methodology & Algorithm Used

To solve the classification problem, the following methodology and machine learning techniques were employed:

2.1 Data Collection and Structure

- **The dataset used contains student responses to a study method questionnaire.**
- **Each entry in the dataset consists of:**
  - **visual_score: Numerical score indicating preference for visual learning.**
  - **auditory_score: Numerical score indicating preference for auditory learning.**
  - **kinesthetic_score: Numerical score indicating preference for hands-on/physical learning.**
  - **learning_style: The actual learning style (visual, auditory, or kinesthetic).**

2.2 Data Preprocessing

- **The dataset was loaded using the pandas library.**
- **Features (X) and target label (y) were extracted.**
- **The data was split into training (80%) and testing (20%) sets.**

2.3 Model Selection: Random Forest Classifier

- **A Random Forest Classifier was chosen for this multi-class classification task.**
- **It is a robust ensemble learning method that combines multiple decision trees and reduces the risk of overfitting.**

2.4 Training and Prediction

- **The model was trained on the training dataset using fit().**
- **Predictions were made on the test dataset using predict().**

2.5 Evaluation Metrics

- **Confusion Matrix: Displays correct and incorrect classifications for each learning style.**
- **Accuracy: Percentage of correctly predicted labels.**

- **Precision: Proportion of correct positive predictions (weighted average for multi-class).**
- **Recall: Proportion of actual positives correctly identified (also weighted average).**

2.6 Visualization

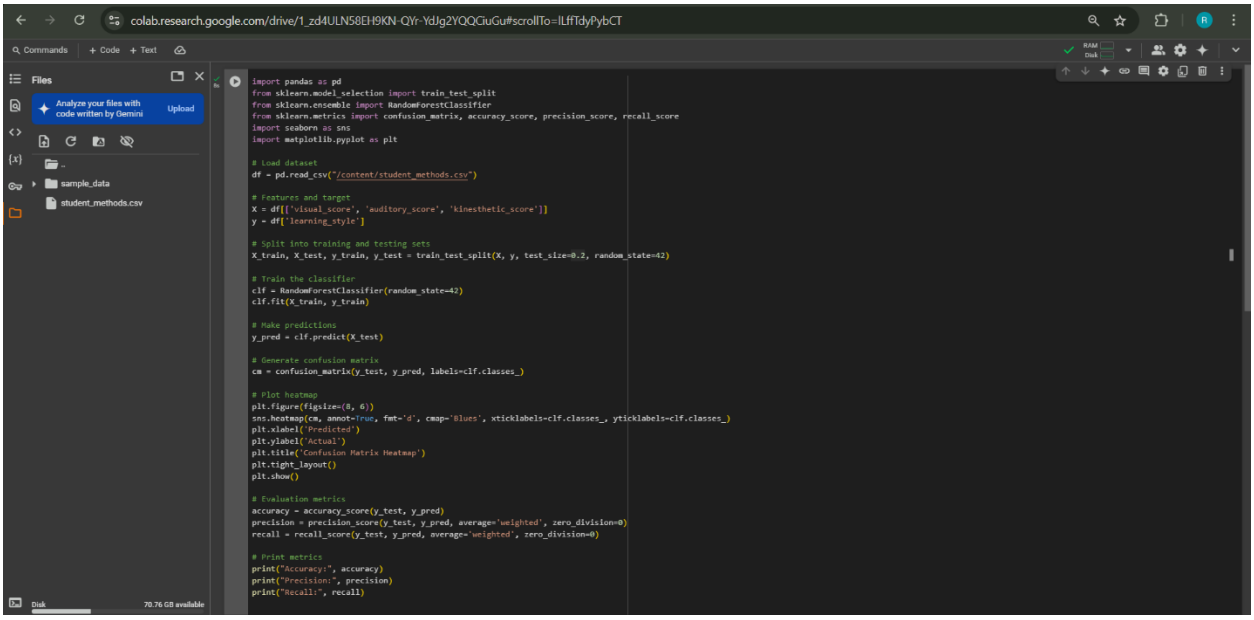
- **A heatmap of the confusion matrix was generated using seaborn for better interpretability.**

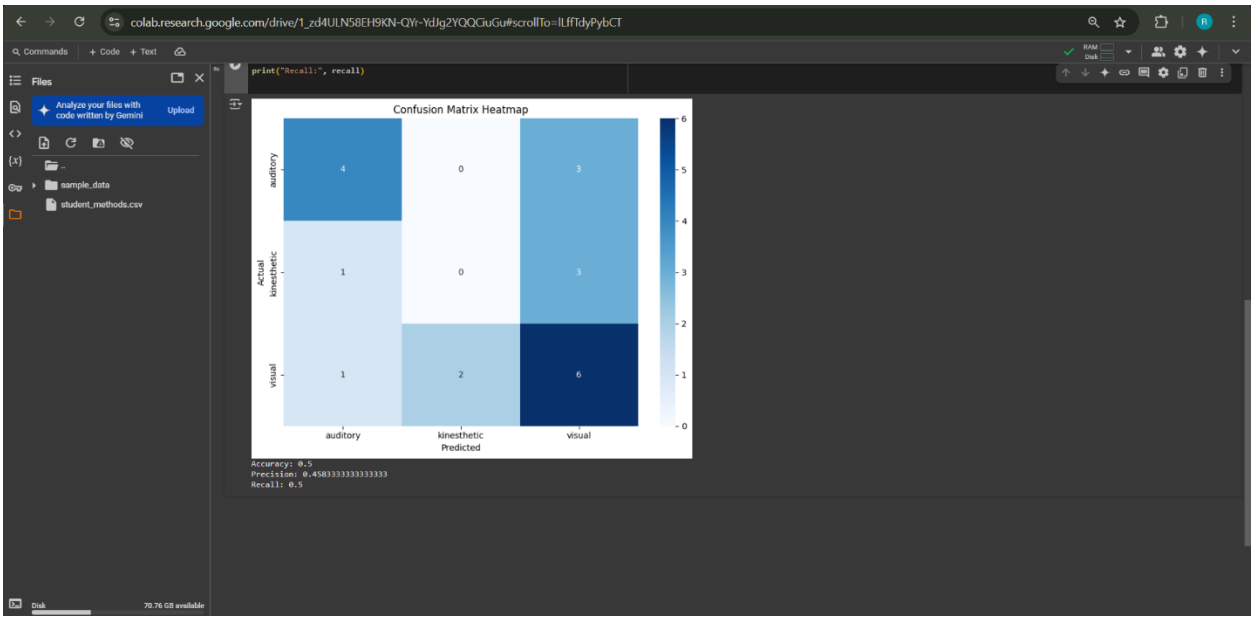
Python Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("/content/student_methods.csv")

# Features and target
X = df[['visual_score', 'auditory_score', 'kinesthetic_score']]
y = df['learning_style']

# Split 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)

# Train the classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Generate confusion matrix
```

```
cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)


# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=clf.classes_,
yticklabels=clf.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.tight_layout()
plt.show()


# Evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)


# Print metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

Screenshot of Output





References & Credit

- **Dataset provided (student_methods.csv)**
- **Libraries Used**:
    - **Pandas**: For data manipulation and analysis.
    - **Scikit-learn**: For implementing the machine learning pipeline, including model training and evaluation.
    - **Seaborn and Matplotlib**: For creating visualizations like the heatmap of the confusion matrix.
- **Machine Learning Concepts**:
    - Ensemble learning techniques such as the **Random Forest Classifier**.
    - Supervised learning and multi-class classification.
    - Evaluation metrics including accuracy, precision, and recall.
- **Development and Guidance**:
    - Developed using Python in a Jupyter notebook or similar environment.
- Google Colab's files.upload() was used for importing CSV files interactively.
    - Reference: Google Colaboratory Documentation.