Predict Students Pass or Fail

Overview

This machine learning project aims to **predict whether a student will pass or fail** based on various academic, personal, and lifestyle-related factors. Using classification algorithms, the model identifies patterns in student behavior and background to determine the likelihood of success in exams.

Goal

To develop a **reliable classification model** that predicts whether a student will **pass or fail**, using features such as study hours, attendance, health, parental education, and internet access.

Source: Kaggle student performance dataset

- load the dataset using pandas and analyze the column.

Exploratory Data Analysis (EDA) Performed to understand the distribution, relationships, and quality of the dataset prior to modeling.

- **Total Features**: 14
- **Numerical** (7): study hours, play hours, attendance, previous grade, daily travel time, sleep hours.
- **Categorical** (7): Student id, gender, pass fail , Parental Education, internet access , likes subject, study group.
- **Correlation heatmap** - To detect multicollinearity and evaluate relationships between features
- **Descriptive Statistics -** exploration of the dataset involved analyzing the central tendency and dispersion of features

Data Preprocessing

- **Missing Values:** Null
- **Data Cleaning**

  Dropped "Student ID "column was removed because it is a unique identifier and does not provide any meaningful information for model training.
- **Encoding**
  - Label encoding for ordinal categories.
  - One-hot encoding for nominal features.
- **Feature Selection**
  - Feature importance (from tree-based models)

Train-Test Split

- Training set and a testing set. The training set, which comprises 80% of the data, was used to train the models, while the remaining 20% was used for model evaluation Model comparsion.

Model Comparison

- **Logistic Regression**: A linear model that applies logistic function to predict the probability of each class.
- **Decision Tree**: A tree-based model that recursively splits the dataset based on different features to create decision rules.
- **Random Forest**: An ensemble model that combines multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting.
- **K-Nearest Neighbors (KNN)**: Predicts the class of a new instance based on the class labels of its nearest neighbors in the training data.

Model: Random Forest Classifier (from Scikit-learn)

Model Evaluation – Classification

Evaluates final model performance on unseen test data

- Accuracy - The ratio of correctly predicted instances to the total instances. Higher accuracy means more correct predictions overall.
- Recall - The ratio of true positive predictions to all actual positives. High recall means fewer false negatives.

Deployment

- Serialized using joblib.
- Integrated into a user-friendly interface with **Streamlit.**

Student Score Predictor

Overview

The *Student Score Predictor* is a machine learning application designed to estimate students' academic performance based on lifestyle and background factors. By leveraging insights from behavioral and demographic features, the tool provides both predictive capabilities and interpretability for educational stakeholders.

Goal

Develop a machine learning model to **predict academic scores** from lifestyle and background data, helping educators, parents, and students make data-driven improvements.

Source: Kaggle student performance dataset

- load the dataset using pandas and analyze the column.

Exploratory Data Analysis (EDA): Performed to understand the distribution, relationships, and quality of the dataset prior to modeling.

- **Total Features**: 16
- **Numerical (9):** study hours, age, attendance, Netflix hours, exercise frequency, mental health rating, sleep hours, social media hours, exam score.
- **Categorical** (7): Student id, gender Parental Education, internet access, extracurricular participation, part time job, diet quality.
- **Correlation heatmap** - To detect multicollinearity and evaluate relationships between features
- **Descriptive Statistics -** exploration of the dataset involved analyzing the central tendency and dispersion of features

Data Preprocessing

- **Missing Values:**
  - **Numerical** – No missing values
  - **Categorical** – Missing values handled using **placeholders**
- **Data Cleaning**
  - Dropped "Student ID "column was removed because it is a unique identifier and does not provide any meaningful information for model training.
  - **Encoding**:
    - Label encoding for ordinal categories.
    - One-hot encoding for nominal features.
- **Feature Selection**
  - Feature importance (from tree-based models)

Train-Test Split

- Training set and a testing set. The training set, which comprises 80% of the data, was used to train the models, while the remaining 20% was used for model evaluation Model comparsion.

Model Comparison:

- **Linear Regression** : Algorithm that models the relationship between input features and a continuous target variable by fitting a straight line. Though more suitable for continuous outputs.
- **Decision Tree**: A tree-based model that recursively splits the dataset based on different features to create decision rules.
- **Random Forest**: An ensemble model that combines multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting.
- **Gradient Boosting Regressor :** A boosting algorithm that builds models sequentially, where each new model attempts to correct the errors made by the previous one
- **K-Nearest Neighbors (KNN)**: Predicts the class of a new instance based on the class labels of its nearest neighbors in the training data.

Model: Gradient Boosting Regressor (from Scikit-learn)

Model Evaluation – Regression

Evaluates final model performance on unseen test data

- **Root Mean Squared Error (RMSE)** Measures the square root of the average squared differences between actual and predicted values.
- **R^2 Score** Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. Range - A value closer to 1.0 a better fit

Deployment

- Serialized using joblib.
- Integrated into a user-friendly interface with **Streamlit.**

Student Learning Style Clustering

Overview

This project is a machine learning-based (Unsupervised)designed to categorize students into learning style groups — **Excellent**, **Good**, or **Average** — based on behavioral and performance-related input metrics. The model leverages **K-Means Clustering** and is deployed using **Streamlit** for interactive use.

Goal

Automatically group students by their learning behavior using clustering, providing educators with quick insights to tailor instruction and support.

Dataset

A synthetic dataset was generated to simulate student learning behavior for clustering purposes. This dataset includes features such as reading speed, interaction time, quiz accuracy, video watched percentage, and topics mastered — enabling unsupervised learning models to identify distinct learning style groups (e.g., Excellent, Good, Average).

Exploratory Data Analysis (EDA): Performed to understand the distribution, relationships, and quality of the dataset prior to modeling.

- **Total Features**: 5
    - **Numerical (5):**
    - Reading speed -Words per minute while reading
    - Interaction time -Time spent interacting with the learning platform
    - Quiz accuracy - Fraction of correct answers (0 to 1)
    - Video watched percent - Percentage of course videos watched (0 to 1)
    - Topics mastered - Count of mastered learning topics
- **Correlation heatmap** - To detect multicollinearity and evaluate relationships between features
- **Descriptive Statistics -** exploration of the dataset involved analyzing the central tendency and dispersion of features

Data Preprocessing

- **Missing Values:**
    - **Numerical** – No missing values.

Feature Scaling

- Applied using StandardScaler to normalize input values before clustering.

Model Comparison

- K-Means is a centroid-based, unsupervised machine learning algorithm used for clustering data into groups based on feature similarity. I experimented with multiple values of **K** using the **K-Means** algorithm and found that three clusters gave the best results.
- DBSCAN is an unsupervised clustering algorithm that groups data points based on density. Unlike K-Means, it does not require the number of clusters (K) to be specified in advance.

Model - KMeans (from Scikit-learn)

Model Evaluation

Silhouette Score Measures how similar a sample is to its own cluster compared to other clusters.

Range: -1 (bad clustering) to 1 (good clustering).

Deployment

- The trained model was **serialized using joblib**.
- A **Pipeline** was used to include both the **scaling** (e.g., StandardScaler) and the **trained classifier** in one integrated object.
- Integrated into a user-friendly interface with **Streamlit.**

Student Dropout Prediction

Overview

This machine learning project aims to **predict whether a student will pass or fail** based on various academic, personal, and lifestyle-related factors. Using classification algorithms, the model identifies patterns in student behavior and background to determine the likelihood of success in exams.

Problem Statement

Many factors—beyond classroom learning—impact a student's academic performance. Lifestyle habits such as sleep, screen time, and physical activity can either enhance or hinder a student's ability to succeed.

Goal

To develop a **classification model** that predicts whether a student will **pass or fail**, using features such as study hours, attendance, health, parental education, and internet access.

Source: Kaggle student performance dataset

load the dataset using pandas and analyze the column.

Exploratory Data Analysis (EDA): Performed to understand the distribution, relationships, and quality of the dataset prior to modeling.

- **Total Features**: 15
    - **Numerical** (7) - Jee main score, jee advanced score, mock test score avg, class_12_percent, attempt count, daily study hours, dropout
    - **Categorical** (8) - School board, coaching institute, family income, parent education, location type, peer pressure level, mental health issues, admission taken
    - **Correlation heatmap** - To detect multicollinearity and evaluate relationships between features
    - **Descriptive Statistics -** exploration of the dataset involved analyzing the central tendency and dispersion of features
    - Data Frame['column'].value_counts() – To obtain the count of each class

Data Preprocessing:

- **Missing Values**
    - **Numerical** – No missing values
    - **Categorical** – Missing values handled using **placeholders**

- **Data Cleaning**

  Dropped "Student ID "column was removed because it is a unique identifier and does not provide any meaningful information for model training.

- **Encoding**:

- Label encoding for ordinal categories.
- One-hot encoding for nominal features.

- **Feature Selection**:
  - Feature importance (from tree-based models)

Train-Test Split

- Training set and a testing set. The training set, which comprises 80% of the data, was used to train the models, while the remaining 20% was used for model evaluation Model comparsion.

Smote – Synthetic Minority Over-sampling Technique

- In dropout prediction problems, the dataset often has **fewer students who actually drop out**, making it hard for machine learning models to learn meaningful patterns from the minority class. This results in **biased models** that predict the majority class more accurately, but perform poorly on the minority class.

Model Comparison

- **Logistic Regression**: A linear model that applies logistic function to predict the probability of each class.
- **Decision Tree**: A tree-based model that recursively splits the dataset based on different features to create decision rules.
- **Random Forest**: An ensemble model that combines multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting.
- **K-Nearest Neighbors (KNN)**: Predicts the class of a new instance based on the class labels of its nearest neighbors in the training data.

Model: Random Forest Classifier (from Scikit-learn)

Model Evaluation – Classification

Evaluates final model performance on unseen test data

- Accuracy - The ratio of correctly predicted instances to the total instances. Higher accuracy means more correct predictions overall.
- Recall - The ratio of true positive predictions to all actual positives. High recall means fewer false negatives.

Deployment

- Serialized using joblib.
- Integrated into a user-friendly interface with **Streamlit.**

Topic Detection(DL-NLP)

Overview

The Topic Detection that uses a deep learning model (LSTM) to classify student-written paragraphs into academic subject categories such as English, Science, or Social Studies. It is developed using Python, TensorFlow/Keras, Streamlit, and Natural Language Processing (NLP) tools.

Dataset Description

- Source: Student-generated text data, labeled by subject.
- Input: Paragraphs written by students.
- Target: Subjects such as literature, social studies, history, math sentences, technology science sentences

Exploratory Data Analysis (EDA)

- Length Analysis: Most paragraphs under 100 tokens – helped define maxlen for padding.

Data Preprocessing

- Tokenizer: Stored and reused for inference.
- LabelEncoder: Encodes class labels and ensures correct label mapping.

Model Architecture

- Type: LSTM (Long Short-Term Memory) - sequential model designed for text classification.
- Framework: TensorFlow / Keras
- Layers:
    - Embedding layer for vector representation of words.
    - LSTM layer for capturing sequential patterns.
    - Dense layer for subject classification with softmax activation.

Model Evaluation

- Evaluates final model performance on unseen test data
- model.evaluate() -This function computes the loss and metrics (like accuracy)

Deployment

- Model Saved: .h5 format using Keras.
- Tokenizer & LabelEncoder: Saved using joblib.
- User Interface: Built with Streamlit for paragraph input and real-time subject prediction.

Digit Recognition

Overview

This project is a web-based digit recognition application, a pre-trained Convolutional Neural Network (CNN). It predicts handwritten digits (0–9) from grayscale images using the MNIST dataset model.

Objective

To build and train a **Convolutional Neural Network (CNN)** model that classifies images based on labeled data. The model is trained on image-like features, normalized and reshaped to match the input requirements of a CNN

Source: Kaggle student performance dataset

Data Preprocessing

- Reshaped into 4D array suitable for CNN input: (samples, height, width, channels)

Train-Test Split

- 80% used for training 20% for evaluating generalization performance.

Model Architecture (CNN)

- Conv2D: Extracts spatial features.
- MaxPooling2D: Reduces spatial dimensions, controlling overfitting.
- Flatten: Converts 2D feature maps into a 1D vector.
- Dense: Fully connected layers.
- Output Layer: softmax activation used for multi-class classification.

Model Compilation

- Optimizer: adam – adaptive learning.
- Loss: categorical_crossentropy – suitable for multi-class classification.
- Metric: Accuracy

Model Training & Evaluation

- Evaluates final model performance on unseen test data.
  - **Loss**: Measure of model error.
  - **Accuracy**: Fraction of correctly classified instances.

Deployment

- Model Saved: .h5 format using Keras.
- Tokenizer & LabelEncoder: Saved using joblib.
- User Interface: Built with Streamlit for paragraph input and real-time subject prediction.

Topic Summarizer

Overview

The TED Talk Tagger App is an interactive web application built using Streamlit that utilizes Hugging Face's facebook/bart-large-mnli model to perform zero-shot classification on TED Talk descriptions. Users input a TED Talk summary, and the app predicts the most relevant tags from a predefined set.

Goal

To automatically generate relevant tags for TED Talks using zero-shot learning.

Model: facebook/bart-large-mnli (Zero-Shot Classification)

facebook/bart-large-mnli

- Type: Zero-shot classifier using natural language inference.
- Purpose: Determines if a label (tag) is entailed by the given description.
- Advantage: Can classify into any category without needing labeled training data for those specific tags.

Library: Hugging Face Transformers

Deployment

- User Interface: Built with Streamlit for paragraph input and real-time subject prediction.