Github Link: **https://github.com/vishnuharichandran66/project-road-safety.git**

Project Title: Enhancing road safety with AI-driven traffic accident analysis and prediction

PHASE-2

- **Problem Statement**

Educational institutions strive to improve student outcomes, but often lack the tools to proactively identify students who may underperform or drop out. Traditional approaches to academic support are reactive and generalized, failing to consider the individual differences among students. This project aims to develop a predictive model that leverages academic records (e.g., grades, attendance) and demographic data (e.g., age, gender, socioeconomic background) to forecast student performance. By accurately identifying students at risk of poor academic outcomes, this model can help educators intervene early, personalize learning strategies, and ultimately enhance student success rates.

- **Project Objectives**
  - To collect academic and demographic data of students.
  - To clean and preprocess the collected data
  - To analyze key factors affecting student performance.
  - To build predictive models using machine learning techniques.
  - To evaluate the models for accuracy and reliability.
  - To identify students at risk of underperforming.
  - To assist educators in making data-driven decisions

.**Flowchart of the Project Workflow**

Data Description

- The dataset used for predicting student performance typically includes a combination of academic and demographic features. These may be collected from institutional records or standardized datasets.

1. Academic Data

- Previous Grades (e.g., exam scores, assignment marks)
- Attendance Records

- Study Hours per Week
- Participation in Class Activities
- Number of Subjects Failed or Passed
- Homework Submission Record
- School Support Programs (e.g., tutoring, mentoring)

2. Demographic Data

- Age
- Gender
- Parental Education Level
- Parental Occupation
- Family Size
- Socioeconomic Status
- Home Internet Access
- Living Area (Urban/Rural)

3. Target Variable

- Final Grade or Academic Performance (classified as: Pass/Fail, Letter Grade, or GPA range)
- Let me know if you'd like help format
- **Dataset Link:https://www.kaggle.com/datasets/sahityasetu/motor-vehicle-collisions-crashes-usa**

- **Data Preprocessing**
  - Data preprocessing is a crucial step to prepare raw data for effective analysis and model building. The following steps are applied:

1. Data Cleaning

- Handling missing values (e.g., filling with mean/mode or removing rows).
- Correcting inconsistent entries (e.g., standardizing categorical values).
- Removing duplicate records

2. Data Transformation

- Encoding categorical variables using techniques like Label Encoding or One-Hot Encoding (e.g., gender, school support).
- Normalizing or scaling numerical features (e.g., grades, study time) to ensure uniformity.

3. Feature Selection

- Identifying and retaining the most relevant features affecting student performance.
- Removing redundant or irrelevant columns.

4. Data Splitting

- Dividing the dataset into training and testing sets (e.g., 70:30 or 80:20 split) to evaluate model performance.

5. Outlier Detection (optional)

- Identifying and addressing outliers that may distort model accuracy

- **Exploratory Data Analysis (EDA)**

EDA helps in understanding the structure, patterns, and relationships in the data before applying any machine learning model. It involves both statistical summaries and visualizations**.**

1. Understanding Data Distribution

Summary statistics (mean, median, mode, standard deviation) for numeric variables like grades and study time.

Value counts for categorical variables (e.g., gender, parental education).

2. Univariate Analysis

Histograms to examine the distribution of individual features (e.g., final grades, age).

Bar plots for categorical data (e.g., gender, school type).

3. Bivariate Analysis

Box plots to compare performance across demographic groups (e.g., gender vs. final grade).

Scatter plots to analyze correlations between study time, attendance, and grades.

4. Correlation Analysis

Heatmaps to visualize correlations between numeric features.

Identifying strongly correlated features that may influence student performance.

5. Outlier Detection

Using boxplots and statistical methods (e.g., IQR) to spot unusual values**.**

- **Feature Engineering**
  - Feature engineering involves creating new input features or modifying existing ones to improve model performance and capture hidden patterns in the data.

1. Encoding Categorical Variables

- Apply Label Encoding or One-Hot Encoding for features like gender, school type, parental education, and internet access.

2. Binning Continuous Variables

- Convert continuous variables (e.g., age, study time, grades) into categories (e.g., low, medium, high) to capture non-linear relationships.

3. Creating New Features

- Total study time per week = study hours × number of study days.
- Attendance ratio = classes attended / total classes.
- Parental involvement score = composite feature from parental education, support, and school meetings.

4. Interaction Features

- Combine related features (e.g., study time × parental education) to capture interaction effects on performance.

5. Normalization/Scaling

- Normalize numerical features like grades and attendance using Min-Max Scaling or Standardization to ensure uniformity.

.

8.Model Bulding

- Model building involves training machine learning algorithms on the processed dataset to predict student performance.

1. Defining the Problem

- **Type:** Classification (e.g., Pass/Fail, Grade A/B/C) or Regression (e.g., predicting exact score or GPA).
- Choose model type based on the target variable**.**

2. Splitting the Dataset

- Split the data into training and testing sets (e.g., 80% train, 20% test).

3. Selecting Machine Learning Algorithms

- Common algorithms used:
- Logistic Regression – for binary classification (e.g., pass/fail).
- Decision Tree – interpretable and handles both types of data.
- Random Forest – ensemble method for higher accuracy
- Support Vector Machine (SVM) – effective in high-dimensional spaces.
- K-Nearest Neighbors (KNN) – simple, distance-based classification.
- Gradient Boosting / XGBoost – for advanced, high-performance predictions.
- Linear Regression – if predicting exact numeric scores.

4. Training the Models

- Train each model using the training dataset.
- Tune hyperparameters using Grid Search or Random Search.

5. Evaluating Model Performance

- Use appropriate metrics:
- Accuracy, Precision, Recall, F1-Score (for classification).
- RMSE, MAE, R^2 Score (for regression).
- Use cross-validation to validate model stability.
-

• Visualization of Results & Model Insights

- Visualizing results helps in understanding model performance and extracting meaningful patterns from data and predictions.

1. Model Performance Visualization

- Confusion Matrix: Shows true vs. predicted classifications (e.g., Pass/Fail).
- ROC Curve & AUC Score: Evaluates classification performance visually.
- Accuracy/Precision/Recall Bar Chart: Compares metrics across multiple models.

- Feature Importance Plot: Displays which features contribute most to predictions (e.g., Random Forest or XGBoost importance chart).

2. Prediction Distribution

- Histogram/Bar Chart of predicted grades or performance levels.
- Scatter Plot of actual vs. predicted values (for regression tasks).

3. Insights from Data Patterns

- Heatmap of Correlations: Shows relationships between features (e.g., study time vs. grades).
- Box Plots: Compare performance across demographic groups (e.g., gender, parental education).
- Pair Plots: Analyze how multiple features interact with each other and the target variable.

4. Interpretability Tools (optional)

- SHAP or LIME Plots: Explain individual predictions and global feature impacts (if advanced interpretation is needed).

- **Tools and Technologies Used**

1. Programming Language

- Python – for data preprocessing, model building, and visualization**.**

2. Libraries and Frameworks

- Pandas – for data manipulation and analysis.
- NumPy – for numerical operations.
- Matplotlib & Seaborn – for data visualization and plotting.
- Scikit-learn – for machine learning algorithms, model training, and evaluation.
- XGBoost – for gradient boosting and advanced predictive modeling.
- SHAP/LIME (optional) – for model interpretability and explaining predictions.

3. Development Environment

Jupyter Notebook / Google Colab – for writing and executing code interactively.

4. Version Control (optional)

- Git & GitHub – for code versioning and project collaboration

- **Team Members and Contributions**

- **1. [M.VISHNU HARI CHANRAN] – Project Manager**
- Responsibilities: Led the project, managed the team, coordinated tasks, and ensured deadlines were met. Maintained communication between team members and stakeholders, and provided regular project updates.

- **2. [D. TAMIL MANI] – Data Engineer**
- Responsibilities: Collected, cleaned, and preprocessed the student performance data. Managed data integration, ensured data quality, and prepared datasets for analysis and model building.

- **3. [R.VITHISH] – Data Scientist**
- Responsibilities: Conducted Exploratory Data Analysis (EDA), visualized data trends and patterns, performed feature engineering, and assisted with model selection. Analyzed the correlation between academic and demographic factors**.**

- **4. [C.SANJAY] – Machine Learning Engineer**
- Responsibilities: Built, trained, and fine-tuned machine learning models, including algorithms like Random Forest, XGBoost, and SVM. Evaluated model performance using various metrics, and selected the best model for prediction.