Student Exam Performance Predictor

End-to-End Machine Learning Project

Python | Flask | Scikit-learn | HTML/CSS | XGBoost

Project Overview

This project predicts student math scores based on various demographic and academic factors. It demonstrates a complete ML pipeline from data ingestion to web deployment using Flask.

Input Features

- **Gender**
- **Race/Ethnicity**
- **Parental Education Level**
- **Lunch Type**
- **Test Preparation Course**
- **Reading & Writing Scores**

Target Variable

Math Score

Predicted score out of 100 points

Project Architecture

```
mlproject/
├── src/
│   ├── components/
│   │   ├── data_ingestion.py      # Data loading and splitting
│   │   ├── data_transformation.py # Feature engineering & preprocessing
│   │   └── model_trainer.py       # Model training and evaluation
│   ├── pipeline/
│   │   └── predict_pipeline.py    # Prediction pipeline
│   ├── exception.py               # Custom exception handling
│   ├── logger.py                  # Logging configuration
│   └── utils.py                   # Utility functions
├── templates/
│   ├── home.html                  # Main prediction form
│   └── index.html                 # Landing page
├── notebook/
│   └── data/
│       └── stud.csv               # Raw dataset
├── artifacts/                     # Generated model artifacts
├── app.py                         # Flask web application
├── setup.py                       # Package installation
└── requirements.txt               # Dependencies
```

Features

Machine Learning Pipeline

- **Data Ingestion:** Automated data loading and train-test splitting
- **Data Transformation:** Comprehensive preprocessing pipeline
- **Model Training:** Multiple algorithm comparison
- **Model Evaluation:** R^2 score-based selection

Web Application

- **Flask-based** user interface
- **Real-time prediction** through web forms
- **Input validation** and error handling
- **Responsive design** for better UX

Supported Machine Learning Algorithms

Linear Regression	Decision Tree	Random Forest	Gradient Boosting
XGBoost	CatBoost	AdaBoost	Auto-Selection

Installation & Setup

Prerequisites

- Python 3.7+
- pip package manager

1. Clone the Repository

```
git clone https://github.com/het004/mlproject.git
cd mlproject
```

2. Create Virtual Environment

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install Dependencies

```
pip install -r requirements.txt
```

4. Install the Package

```
pip install -e .
```

Usage

Training the Model

```
python src/components/data_ingestion.py
```

This will:

- Load the dataset from `notebook/data/stud.csv`
- Split into train/test sets
- Apply data transformations
- Train multiple models with hyperparameter tuning
- Save the best model to `artifacts/`

Running the Web Application

```
python app.py
```

Then navigate to `http://localhost:5000` in your browser.

Making Predictions via API

```
from src.pipeline.predict_pipeline import CustomData, PredictPipeline

# Create input data
data = CustomData(
    gender='male',
    race_ethnicity='group A',
    parental_level_of_education="bachelor's degree",
    lunch='standard',
    test_preparation_course='completed',
    reading_score=85,
    writing_score=90
)

# Get prediction
pipeline = PredictPipeline()
result = pipeline.predict(data.get_data_as_data_frame())
print(f"Predicted Math Score: {result[0]}")
```

Data Schema

Input Features

Feature	Type	Description	Possible Values
gender	Categorical	Student's gender	male, female
race_ethnicity	Categorical	Ethnic group	group A, B, C, D, E
parental_level_of_education	Categorical	Parent's education	associate's degree, bachelor's degree, high school, master's degree, some college, some high school
lunch	Categorical	Lunch type	free/reduced, standard
test_preparation_course	Categorical	Test prep completion	none, completed
reading_score	Numerical	Reading score (0-100)	Integer
writing_score	Numerical	Writing score (0-100)	Integer

Target Variable

math_score: Mathematics score (0-100)

Model Performance

Model Selection

The system automatically selects the best performing model based on R^2 score on the test set. Models with $R^2 < 0.6$ are rejected to ensure minimum performance standards.

Hyperparameter Tuning

GridSearchCV with 3-fold cross-validation is used for optimal parameter selection across all supported algorithms.

Quick Start Example

1 — Install and Run

```
git clone https://github.com/het004/mlproject
cd mlproject
pip install -r requirements.txt
python app.py
```

2 — Open Browser

Navigate to `http://localhost:5000`

3 — Get Predictions

Fill the form with student details and get instant math score prediction!

Project Highlights

Robust Error Handling

- Custom exception classes with detailed error tracking
- Comprehensive logging system with timestamps
- Graceful failure handling in all components

Modular Design

- Separate components for each ML pipeline stage
- Reusable utility functions
- Clean separation of concerns

Production Ready

- Pickle serialization for model persistence
- Scalable Flask application structure
- Environment-agnostic file paths

Future Enhancements

- **Model Interpretability:** Add SHAP values for prediction explanations
- **Advanced Validation:** Implement cross-validation metrics
- **API Documentation:** Add Swagger/OpenAPI documentation
- **Docker Deployment:** Containerize the application
- **Database Integration:** Store predictions and user interactions
- **A/B Testing:** Framework for model comparison in production

Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add some amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

Contact

Author: Het

Email: shahheta1973@gmail.com

GitHub: het004

License

This project is open source and available under the MIT License.

Star this repository

if you found it helpful!

This README provides a comprehensive overview of your machine learning project, including installation instructions, usage examples, architecture details, and future enhancement possibilities. The project demonstrates excellent software engineering practices with modular design, proper error handling, and a complete ML pipeline from data ingestion to web deployment.

Made with Genspark