PRACTICAL-TRAINING-2025

SUBMITTED BY : Parijat Roy ROLL NUMBER : 2228125

SUBMITTED TO : Dr Manas Nayak

Comprehensive Project Report: Student Performance Prediction

1. Introduction & Project Goal

The primary objective of this project was to analyze a dataset on student performance to identify key factors influencing academic scores. Building on this analysis, the project aimed to develop and deploy a robust machine learning model capable of accurately predicting a student's math score based on various demographic and academic attributes. The project follows a complete MLOps lifecycle, from initial data exploration to deploying the final model as an interactive web application.

---

2. Exploratory Data Analysis (EDA) & Initial Findings

Before building the model, an exploratory data analysis was conducted to understand the data's characteristics and uncover initial insights.

- **Data Quality**: The dataset was found to be of high quality, with no missing values or duplicate entries, providing a solid foundation for the analysis.
- **Key Insights**:
    - **Balanced Performance**: Students showed similar average scores across math, reading, and writing, though performance in reading was slightly higher.
    - **Gender Disparity**: On average, female students outperformed male students across all subjects.
    - **Socio-economic Impact**: The type of lunch a student received (a proxy for socio-economic status) was a significant indicator of performance. Students with a "standard" lunch consistently scored higher than those with a "free/reduced" lunch.

---

3. The Machine Learning Pipeline: Architecture and Components

The core of the project is an automated pipeline that processes the data, trains a model, and saves the necessary components for prediction. This pipeline is built upon several interconnected Python scripts.

3.1 Core Utility Modules (The Foundation)

These modules provide essential, reusable functionalities that support the entire pipeline.

- **Custom Exception Handling (exception.py)**: To ensure the application is robust, a custom exception handling system was built. It captures errors and formats them into a detailed message, including the file name and line number where the error occurred, making debugging significantly easier.
- **Logging (logger.py)**: A centralized logging system records every important step, event, and error during the pipeline's execution. Each run generates a timestamped log file, creating a clear audit trail for monitoring and troubleshooting.
- **Utility Functions (utils.py)**: This module contains critical helper functions:
    - save_object & load_object: These functions use dill to save and load Python objects (like the trained model and data preprocessor), ensuring they can be easily persisted and reused.
    - evaluate_models: This automates the process of training multiple machine learning models, performing hyperparameter tuning using GridSearchCV, and evaluating their performance to find the best one.

3.2 Step 1: Data Ingestion (data_ingestion.py)

This is the entry point of the pipeline. Its responsibilities are to:

1. **Load Data**: Read the raw student data from the source CSV file (stud.csv).
2. **Split Data**: Divide the dataset into training (80%) and testing (20%) sets to ensure the model can be evaluated on unseen data.
3. **Store Data**: Save the raw, training, and testing sets into an artifacts directory, making them accessible for the next stages of the pipeline.

3.3 Step 2: Data Transformation (data_transformation.py)

Once the data is ingested, it must be preprocessed to be suitable for machine learning models.

1. **Define Pipelines**: Separate transformation pipelines are created for different data types:
    - **Numerical Pipeline**: Imputes any missing values with the median and scales the features using StandardScaler.
    - **Categorical Pipeline**: Imputes missing values with the most frequent category and then converts the categorical data into a numerical format using OneHotEncoder.

2. **Combine & Apply**: A ColumnTransformer combines these two pipelines into a single preprocessing object. This object is then fitted on the training data and used to transform both the training and testing sets.
3. **Save Preprocessor**: The fitted preprocessing object is saved as preprocessor.pkl using the save_object utility. This is crucial for ensuring that new data for prediction is transformed in exactly the same way as the training data.

3.4 Step 3: Model Training (model_trainer.py)

This component takes the transformed data and finds the best predictive model.

1. **Define Models**: A dictionary of various regression models (e.g., Linear Regression, Random Forest, XGBoost, CatBoost) is defined for evaluation.
2. **Train & Evaluate**: The evaluate_models utility function is called to train each model on the preprocessed training data and evaluate its performance (using the R-squared metric) on the test data.
3. **Select Best Model**: The model with the highest R-squared score is identified as the best-performing model. The analysis showed that **Linear Regression** and **Ridge Regression** were the top performers, both achieving an R-squared score of approximately **0.88**.
4. **Save Best Model**: The best model object is saved as model.pkl for later use in the prediction pipeline.

---

4. Serving Predictions

After training, the saved model and preprocessor are used to make predictions on new data.

4.1 The Prediction Pipeline (predict_pipeline.py)

This script operationalizes the model for inference:

- **CustomData Class**: A helper class that takes new data (e.g., from a web form) and structures it into a pandas DataFrame with the correct column names and data types.
- **PredictPipeline Class**: This class loads the saved preprocessor.pkl and model.pkl. Its predict method takes the new data DataFrame, transforms it using the loaded preprocessor, and feeds the result to the loaded model to get a prediction.

4.2 The Web Application (app.py)

A user-friendly web interface is created using **Flask**:

- **Home Page**: Renders an HTML form where a user can input a student's details.
- **Prediction Route**: When the form is submitted, the application:
    1. Collects the input and creates a CustomData object.
    2. Calls the PredictPipeline to get the math score prediction.
    3. Displays the predicted score back to the user on the web page.

---

5. Packaging and Deployment

To ensure the project is reproducible and easy to deploy, several configuration files are used.

- **requirements.txt**: Lists all the Python libraries the project depends on.
- **setup.py**: Makes the entire project installable as a Python package. This standardizes the project structure and handles the installation of dependencies listed in requirements.txt.
- **Dockerfile**: Contains instructions to build a Docker container. This packages the application, its dependencies, and the necessary environment into a single, portable unit. It ensures that the application runs consistently regardless of the deployment environment. The CMD instruction in the Dockerfile is set to run the Flask application, making the model accessible over the web.

Below attached is the certificate alongside this detailed project report, I'm proud to share that the successful execution of this project was supported by a rigorous training program. The course spanned 63 distinct sections, building a strong foundation from basic Python programming and progressively advancing to complex, state-of-the-art concepts such as Transformers. This structured learning was instrumental in developing the end-to-end pipeline presented here.

udemy

Certificate no: UC-d284c976-ef16-42a0-9994-349c4abe37ef
Certificate url: ude.my/UC-d284c976-ef16-42a0-9994-349c4abe37ef
Reference Number: 0004

CERTIFICATE OF COMPLETION

Complete Data Science,Machine Learning,DL,NLP Bootcamp

Instructors **Krish Naik, KRISHAI Technologies Private Limited**

Parijat Roy

Date **Aug. 19, 2025**
Length **99 total hours**