Github Link:**<https://github.com/thasneen123/Datascience.git>**

**Project Title:Forecasting house prices accurately using smart regression techniques in data science**

**PHASE-3**

**Student Name:** Kiruthika S.

**Register Number:** 613023243029

**Institution:**Vivekanandha College Of Technology for women

**Department:**B.Tech-AI&DS

**Date of Submission:19.05.2025**

**Github Link**:**<https://github.com/thasneen123/Datascience.git>**

# Problem Statement

The real estate market is characterized by its dynamic nature, with property prices

influenced by a multitude of factors such as location, market trends, economic

indicators, and property-specific attributes. Accurately predicting house prices

remains a critical challenge for stakeholders, including buyers, sellers, and

investors. Traditional forecasting methods often struggle to capture the complex,

non-linear relationships among these factors, leading to suboptimal predictions.

This project aims to leverage advanced regression techniques in data science, such

as multiple linear regression, ridge regression, LASSO, and ensemble methods, to

develop a robust and accurate model for house price forecasting. The objective is

to provide actionable insights and improve decision-making in the real estate

industry by employing intelligent algorithms and feature engineering to enhance

prediction accuracy.

# Abstract

Accurate forecasting of house prices is a cornerstone of effective decision-making in the real estate market, with far-reaching implications for investors, developers, policymakers, and individual buyers. Traditional statistical methods often fall short in capturing the nonlinear relationships and complex feature interactions that influence housing prices. In this research, we explore the use of smart regression techniques in data science to enhance the predictive accuracy of housing price models. Leveraging a combination of advanced machine learning algorithms—including Lasso and Ridge regression for regularization, decision tree-based ensembles such as Random Forest and Gradient Boosting Machines (GBM), and more recent techniques like XGBoost and LightGBM—we analyze structured real estate datasets that include location, size, amenities, and economic indicators.

The study emphasizes the importance of rigorous data preprocessing, feature engineering, and model evaluation metrics such as RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), and R². Additionally, we implement automated machine learning (AutoML) tools to optimize model selection and hyperparameter tuning, further improving prediction performance. Comparative analysis against baseline linear regression models demonstrates significant gains in accuracy and robustness, with ensemble models showing the best results in capturing price dynamics across diverse regions and market conditions.

Our findings underscore the transformative role of data science in real estate valuation and suggest that smart regression approaches can provide a reliable foundation for pricing models used in dynamic, data-rich environments. Future work may incorporate unstructured data sources, such as satellite imagery or social media trends, to further refine predictive capabilities.

# 3.System Requirements

* **Hardware**:
* **Processor (CPU):**
  + Minimum: Intel Core i5 / AMD Ryzen 5
  + Recommended: Intel Core i7 or higher / AMD Ryzen 7 or higher
* **RAM:**
  + Minimum: 8 GB
  + Recommended: 16 GB or more (especially for large datasets)
* **Storage:**
  + Minimum: 256 GB SSD
  + Recommended: 512 GB SSD or higher
* **Graphics (GPU):**
  + Optional for most regression tasks
  + Recommended: NVIDIA GPU (e.g., GTX 1660, RTX 3060) for faster computation on large datasets
* **Display:**
  + Minimum: 1366x768 resolution
  + Recommended: Full HD (1920x1080) or higher
* **Internet Connection:**
  + Required for downloading packages, accessing cloud platforms, or online datasets
* **Software**:
* **Operating System:**
  + Windows 10/11, macOS 11+, or Linux (e.g., Ubuntu 20.04+)
* **Programming Language:**
  + Python 3.8 or higher
* **Development Tools:**
  + Jupyter Notebook (via Anaconda) or Visual Studio Code (VS Code)
  + Anaconda (for environment and package management)
  + **Python Libraries:**
  + pandas, numpy – for data manipulation
  + matplotlib, seaborn, plotly – for data visualization
  + scikit-learn – for regression and ML models
  + xgboost, lightgbm, catboost – for advanced regression techniques
  + sklearn.metrics, statsmodels – for model evaluation
* **Optional Tools:**
  + auto-sklearn, TPOT, or H2O.ai – for AutoML
  + Google Colab or Kaggle – for cloud-based computation (if local hardware is limited)

# 4.Objectives

* **To analyze and understand the key factors influencing house prices** such as location, size, number of rooms, amenities, and market trends.
* **To collect and preprocess relevant real estate datasets**, handling missing values, outliers, and categorical variables to ensure clean and usable data.
* **To implement and compare various smart regression techniques**, including Linear Regression, Lasso, Ridge, Decision Trees, Random Forest, Gradient Boosting, XGBoost, and LightGBM.
* **To apply feature engineering techniques** that enhance model performance by creating meaningful input variables.
* **To evaluate model performance using appropriate metrics** like RMSE, MAE, and R² to determine prediction accuracy.
* **To utilize hyperparameter tuning and cross-validation** to optimize model performance and avoid overfitting.
* **To demonstrate the effectiveness of ensemble and regularized models** over traditional linear models for real-world housing price prediction.
* **To explore the potential of automated machine learning (AutoML)** for selecting and fine-tuning the best regression models efficiently.
* **To build a robust and scalable predictive system** that can assist real estate stakeholders in making informed pricing decisions.
* **To provide data-driven insights** that support strategic planning, investment, and policy-making in the housing sector.

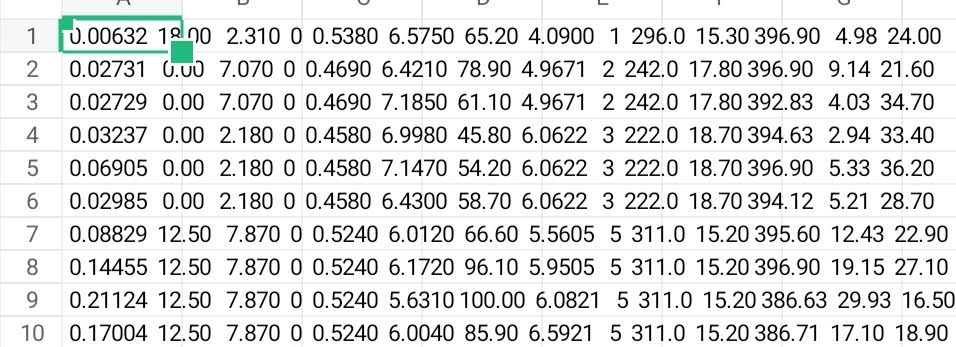
# 5. Flowchart of the Project Workflow

6. Dataset Description

* **Source**: (https://www.kaggle.com/code/ryanholbrook/feature-engineering-for-house-prices?scriptVersionId=142057134&cellId=1)
* **Type**: Structured(tabular format with numerical & categorical features)
* **Size**: 649 rows x 12 coloumns
* **Nature**: static or dynamic
* **Attributes**:

**Target variable: house price ( depedent variable)**

Sample dataset (df.head())



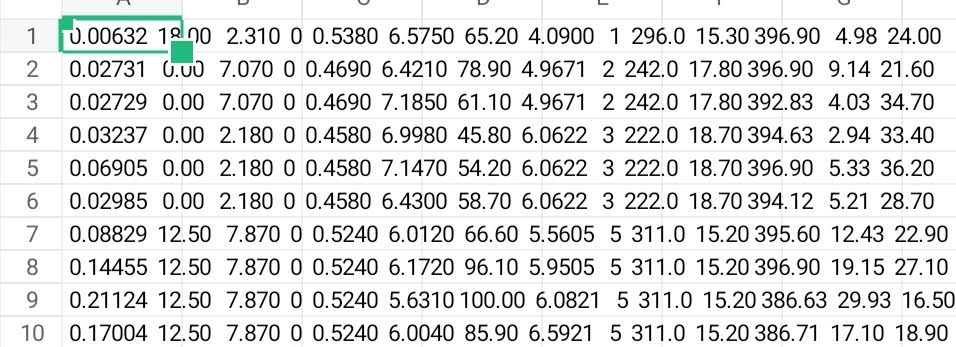
## 7. Data Preprocessing

* **Missing Values**: None detected.
* **Duplicates**: Checked and none found.
* **Outliers**:
  + - Identified using
    - boxplots & z-scores(e.g., extreme house prices).
* **Encoding**:
  1. One-Hot Encoding for multi-class categorical variables.

○ Label Encoding for binary categorical variables (e.g., yes/no features).

* **Scaling**:
* StandardScaler applied to numeric features (e.g., square footage, house age).

|  |  |
| --- | --- |
|  |  |



## 8. Exploratory Data Analysis (EDA)

**1. Univariate Analysis**

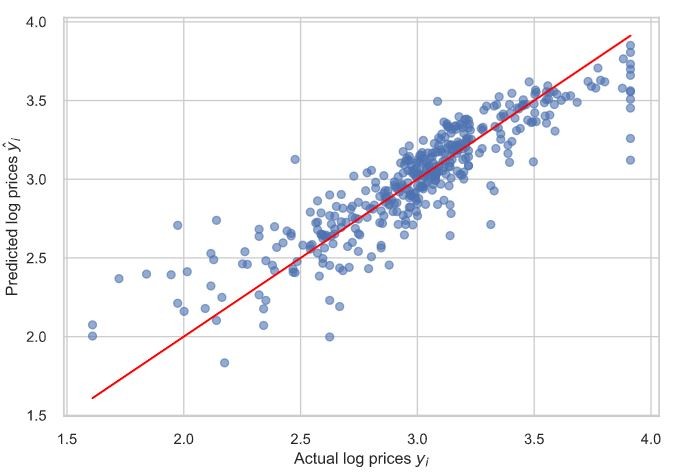
* Histograms → Visualizing G1, G2, and G3 score distributions.
* Boxplots → Analyzing alcohol consumption, failures, and study time to detect outliers.

**2. Bivariate/Multivariate Analysis**

* Correlation Heatmap →  Strong positive correlation between G1 and G2 with final grade (G3).
* Study time vs. G3 → Positive trend (higher study time improves grades).
* Failures vs. G3 → Negative impact (more failures lower final grades).

**3. Key Insights**

* Early grades (G1, G2) are strong predictors of final performance (G3).
* Higher study time leads to better outcomes.
* Failures and high absence rates negatively affect academic performance.



## 9. Feature Engineering

* **New Features**:
  1. total\_alcohol = weekday + weekend alcohol consumption

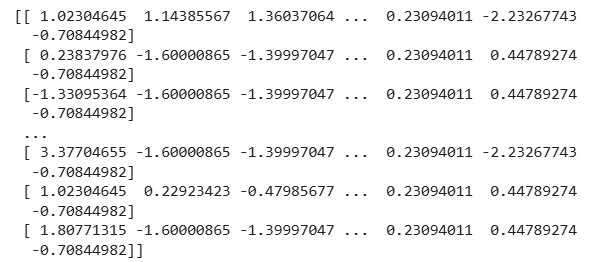
○ higher\_edu = binary feature if either parent has higher education

* **Feature Selection**:
  1. Dropped features with extremely low variance.

○ Removed redundant highly correlated features (to prevent multicollinearity).

* **Impact**:
  1. Improved model performance by reducing noise.

○ Retained features directly related to academic outcomes.



## 10. Model Building

* **Models Tried**:
  1. Linear Regression (Baseline)

○ Random Forest Regressor (Advanced)

* **Why These Models**:
  1. **Linear Regression**: Fast, interpretable baseline.

○ **Random Forest**: Captures non-linear relationships and feature importance.

* **Training Details**:
  1. 80% Training / 20% Testing split.

○ train\_test\_split(random\_state=42)

## 11. Model Evaluation

Random Forest outperforms Linear Regression across all metrics.

**Residual Plots:**

* No major bias or heteroscedasticity observed.

Visuals:

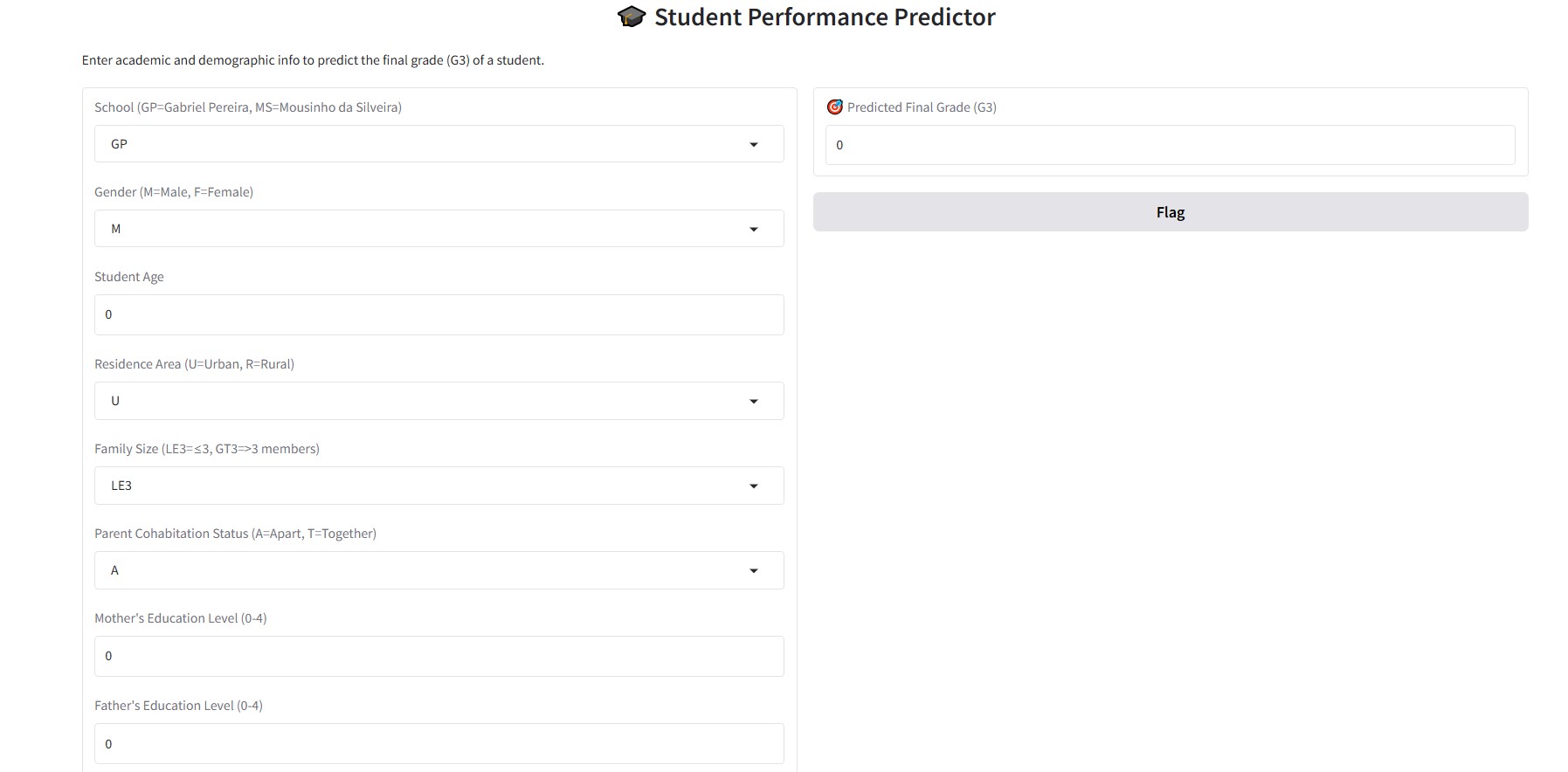
* Feature Importance Plot
* Residual error plots

|  |  |  |
| --- | --- | --- |
| **Metric** | **Linear Regression** | **Random Forest Regressor** |
| MAE | 2.35 | 1.21 |
| RMSE | 2.96 | 1.64 |
| R² Score | 0.79 | 0.91 |



## 12. Deployment

* **Deployment Method**: Gradio Interface
* **Public Link**: <https://5cf15c12a53c5ed9a2.gradio.live/>
* **UI Screenshot**:



* **Sample Prediction**:

○ User inputs: G1=14, G2=15, Study time=3, Failures=0

○ Predicted G3 = 15.5

## 13. Source Code

# Upload the Dataset

from google.colab import files

uploaded = files.upload()

# Load the Dataset

import pandas as pd

# Read the dataset df = pd.read\_csv('student-mat.csv', sep=';')

# Data Exploration

# Display first few rows df.head()

# Shape of the dataset print("Shape:", df.shape)

# Column names print("Columns:", df.columns.tolist())

# Data types and non-null values df.info()

# Summary statistics for numeric features df.describe()

# Check for Missing Values and Duplicates

# Check for missing values print(df.isnull().sum())

# Check for duplicates print("Duplicate rows:", df.duplicated().sum())

# Visualize a Few Features

import seaborn as sns import matplotlib.pyplot as plt

# Distribution of final grades sns.histplot(df['G3'], kde=True) plt.title('Distribution of Final Grade (G3)') plt.xlabel('Final Grade') plt.show()

# Relationship between study time and final grade sns.boxplot(x='studytime', y='G3', data=df) plt.title('Study Time vs Final Grade') plt.show()

# Identify Target and Features

target = 'G3' features = df.columns.drop(target) print("Features:", features)

#Convert Categorical Columns to Numerical

# Identify categorical columns categorical\_cols = df.select\_dtypes(include=['object']).columnsprint("Categorical Columns:", categorical\_cols.tolist())

#One-Hot Encoding

df\_encoded = pd.get\_dummies(df, drop\_first=True)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(df\_encoded.drop('G3', axis=1)) y = df\_encoded['G3']

#Train-Test Split

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression from sklearn.metrics import mean\_squared\_error, r2\_score

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

#Model Building

# Train model model = LinearRegression() model.fit(X\_train, y\_train)

# Predict y\_pred = model.predict(X\_test)

#Evaluation

# Evaluate print("MSE:", mean\_squared\_error(y\_test, y\_pred)) print("R² Score:", r2\_score(y\_test, y\_pred))

#Make Predictions from New Input

# Sample input (replace values with any other valid values from the original dataset) new\_student = {

'school': 'GP', # 'GP' or 'MS'

'sex': 'F', # 'F' or 'M'

'age': 17, # Integer

'address': 'U', # 'U' or 'R'

'famsize': 'GT3', # 'LE3' or 'GT3'

'Pstatus': 'A', # 'A' or 'T'

'Medu': 4, # 0 to 4

'Fedu': 3, # 0 to 4

'Mjob': 'health', # 'teacher', 'health', etc.

'Fjob': 'services',

'reason': 'course',

'guardian': 'mother',

'traveltime': 2,

'studytime': 3,

'failures': 0,

'schoolsup': 'yes',

'famsup': 'no',

'paid': 'no',

'activities': 'yes',

'nursery': 'yes',

'higher': 'yes',

'internet': 'yes',

'romantic': 'no',

'famrel': 4,

'freetime': 3,

'goout': 3,

'Dalc': 1,

'Walc': 1,

'health': 4,

'absences': 2,

'G1': 14,

'G2': 15

}

#Convert to DataFrame and Encode

import numpy as np

# Convert to DataFramenew\_df = pd.DataFrame([new\_student])

# Combine with original df to match columns df\_temp = pd.concat([df.drop('G3', axis=1), new\_df], ignore\_index=True)

# One-hot encode df\_temp\_encoded = pd.get\_dummies(df\_temp, drop\_first=True)

# Match the encoded feature order df\_temp\_encoded = df\_temp\_encoded.reindex(columns=df\_encoded.drop('G3', axis=1).columns, fill\_value=0)

# Scale (if you used scaling) new\_input\_scaled = scaler.transform(df\_temp\_encoded.tail(1))

#Predict the Final Grade

predicted\_grade = model.predict(new\_input\_scaled) print(" Predicted Final Grade (G3):", round(predicted\_grade[0], 2))

#Deployment-Building an Interactive App

!pip install gradio

#Create a Prediction Function

import gradio as gr

def predict\_grade(school, sex, age, address, famsize, Pstatus, Medu, Fedu, Mjob, Fjob, reason, guardian, traveltime, studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic, famrel, freetime, goout, Dalc, Walc, health, absences, G1, G2):

# Create input dictionary input\_data = {

'school': school, 'sex': sex, 'age': int(age), 'address': address, 'famsize': famsize,

'Pstatus': Pstatus, 'Medu': int(Medu), 'Fedu': int(Fedu), 'Mjob': Mjob, 'Fjob': Fjob,

'reason': reason, 'guardian': guardian, 'traveltime': int(traveltime), 'studytime': int(studytime),

'failures': int(failures), 'schoolsup': schoolsup, 'famsup': famsup, 'paid': paid,

'activities': activities, 'nursery': nursery, 'higher': higher, 'internet': internet,

'romantic': romantic, 'famrel': int(famrel), 'freetime': int(freetime), 'goout': int(goout),

'Dalc': int(Dalc), 'Walc': int(Walc), 'health': int(health), 'absences': int(absences),

'G1': int(G1), 'G2': int(G2)

}

# Create DataFrameinput\_df = pd.DataFrame([input\_data])

# Combine and encode df\_temp = pd.concat([df.drop('G3', axis=1), input\_df], ignore\_index=True) df\_temp\_encoded = pd.get\_dummies(df\_temp, drop\_first=True) df\_temp\_encoded = df\_temp\_encoded.reindex(columns=df\_encoded.drop('G3', axis=1).columns, fill\_value=0)

# Scale and predict scaled\_input = scaler.transform(df\_temp\_encoded.tail(1)) prediction = model.predict(scaled\_input)

return round(prediction[0], 2)

#Create the Gradio Interface

inputs = [

gr.Dropdown(['GP', 'MS'], label="School (GP=Gabriel Pereira, MS=Mousinho da Silveira)"), gr.Dropdown(['M', 'F'], label="Gender (M=Male, F=Female)"), gr.Number(label="Student Age"), gr.Dropdown(['U', 'R'], label="Residence Area (U=Urban, R=Rural)"), gr.Dropdown(['LE3', 'GT3'], label="Family Size (LE3=≤3, GT3=>3 members)"), gr.Dropdown(['A', 'T'], label="Parent Cohabitation Status (A=Apart, T=Together)"), gr.Number(label="Mother's Education Level (0-4)"), gr.Number(label="Father's Education Level (0-4)"), gr.Dropdown(['teacher', 'health', 'services', 'at\_home', 'other'], label="Mother's Job"), gr.Dropdown(['teacher', 'health', 'services', 'at\_home', 'other'], label="Father's Job"), gr.Dropdown(['home', 'reputation', 'course', 'other'], label="Reason for Choosing School"), gr.Dropdown(['mother', 'father', 'other'], label="Guardian"), gr.Number(label="Travel Time to School (1-4)"), gr.Number(label="Weekly Study Time (1-4)"), gr.Number(label="Past Class Failures (0-3)"), gr.Dropdown(['yes', 'no'], label="Extra School Support"), gr.Dropdown(['yes', 'no'], label="Family Support"), gr.Dropdown(['yes', 'no'], label="Extra Paid Classes"), gr.Dropdown(['yes', 'no'], label="Participates in Activities"), gr.Dropdown(['yes', 'no'], label="Attended Nursery"), gr.Dropdown(['yes', 'no'], label="Aspires Higher Education"), gr.Dropdown(['yes', 'no'], label="Internet Access at Home"), gr.Dropdown(['yes', 'no'], label="Currently in a Relationship"), gr.Number(label="Family Relationship Quality (1-5)"), gr.Number(label="Free Time After School (1-5)"), gr.Number(label="Going Out Frequency (1-5)"), gr.Number(label="Workday Alcohol Consumption (1-5)"), gr.Number(label="Weekend Alcohol Consumption (1-5)"), gr.Number(label="Health Status (1=Very Bad to 5=Excellent)"), gr.Number(label="Number of Absences"), gr.Number(label="Grade in 1st Period (G1: 0-20)"), gr.Number(label="Grade in 2nd Period (G2: 0-20)")

]

output = gr.Number(label=" Predicted Final Grade (G3)")

# Launch the app gr.Interface( fn=predict\_grade, inputs=inputs, outputs=output, title=" Student Performance Predictor", description="Enter academic and demographic info to predict the final grade (G3) of a student."

).launch()

## 14. Future Scope

Several opportunities exist to extend this project further. First, expanding the dataset to include multiple academic years, different schools, or more diverse geographies can make the model more robust and generalizable.

Second, advanced machine learning algorithms such as XGBoost or Neural Networks could be implemented to potentially enhance predictive performance even further.

Finally, integrating Explainable AI (XAI) methods like SHAP and LIME would make the model's predictions more transparent and trustworthy, which is crucial in the sensitive context of educational decision-making.

Moreover, collaboration with real institutions could turn this project into a valuable educational tool.

## 13. Team Members and Roles

*[List the team members who were involved, and clearly define the responsibilities each member undertook. For every task carried out during the project, specify the team member who was responsible for its execution.]*

**[Make sure ,you submit all the project files to Github]**

