University of Science and Technology of Hanoi
Department of Information and Communication Technology

Machine learning and Data mining I

Report

Prediction of Student GPA from Academic Behaviors

Do Nguyen Gia Nhu - 22BA13248
Nguyen Quang Minh - 22BA13221
Vuong Hong Minh - 22BA13224
Tran Thanh Phat - 22BA13250
Nguyen Quang Anh - BA10002

Hanoi, May 4, 2025

Abstract

This report explores the application of machine learning techniques to predict student's GPA based on their studying behaviors. Various data points, including the duration of the daily study, the study break, the preparation of the exam, the use of social media during the study time, lack of focus in studying,... and the previous academic performance, were collected and analyzed. By utilizing machine learning such as Linear Regression combined Bounded Regressor, this program uses data preprocessing pipelines and feature engineering to ensure accurate predictions. The model performance was evaluated using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2. The results demonstrate significant insights into the correlation between student habits and academic performance, highlighting opportunities for optimization and future research in personalized academic support systems.

Contents

Chapter 1

Introduction

1.1 Context

Academic performance, often measured by Grade Point Average (GPA) has been a focal point in educational research, as it serves as a key metric for evaluating student achievement. While GPA is influenced by multitude of factors, academic behaviors such as class attendance, study duration, physical activity frequency, etc - have emerged as measurable and actionable predictors of student success. Understanding this relationship between behaviors and GPA not only helps the educator in supporting student learning, but also empowers students to adopt strategies that improve their learning outcomes.

This study aims to examine how academic performance is effected by some individual factor and to develop a model using the linear regression method. Linear regression is chosen because of its ability to predict output based on a linear relationship of input.

1.2 Objective

The goal of developing a linear regression model is to have the ability to estimate GPA based on study behaviors. In particular, this study aims to achieve the following specific objectives:
.

1. Identifying factors influencing GPA: Statistically determine which observable academic behaviors (e.g., class attendance, study hours per week, assignment submission timeliness, library resource usage) exhibit the strongest correlation with GPA.

2. Develop a Predictive Linear Regression Model: Construct a robust and interpretable linear regression model to estimate GPA using the identified behavioral predictors, ensuring alignment with statistical assumptions (linearity, homoscedasticity, normality, and independence).

3. Quantify the Relative Impact of Each Behavior: Calculate the regression coefficients for each predictor to assess their individual contributions to GPA, enabling stakeholders to prioritize interventions.

The findings are expected to empower educators to refine student support mechanisms and guide learners in adopting data-backed strategies to enhance their academic outcomes.

Chapter 2

Methodology

2.1 Dataset

We are using the dataset depends on Google Forms to collect the data of the students to determine which observable academic behaviors (e.g., class attendance, study hours per week, assignment submission timeliness, library resource usage) to predict GPA depends on study habits. Then export data to CSV file.

The file contains a dataset with the following features:

- **Timestamp**: When the data was recorded.
- **Gender**: Gender of the participant.
- **Current education level**: Education level of the participant (e.g., University, High school, etc.).
- **Daily study duration**: Time spent studying daily.
- **Preferred study method**: Study methods chosen by the participant (e.g., Self-study, Tutoring, etc.).
- **Use of technology in studying**: Whether technology is used for studying.
- **Most effective study time**: The time of day the participant finds most effective for studying.
- **Study break frequency**: Frequency of study breaks.
- **Exam preparation approach**: When exam preparation begins.
- **Daily sleep hours**: Hours of sleep per day.
- **Exercise frequency**: How often the participant exercises.
- **Diet type**: The participant's dietary preferences (e.g, Balanced diet, No specific diet, etc.)
- **Use of caffeine or stimulants**: Whether caffeine or stimulants are used.
- **Listening to music while studying**: Whether music is played during study sessions.
- **Preferred study environment**: The type of environment preferred for studying.
- **Break duration**: Duration of breaks during study.
- **GPA before adjustment**: GPA prior to adjustment.
- **GPA after adjustment**: GPA after adjustments.

2.2 EDA

2.2.1 Alerts

There are several categories of alerts listed in the report:

- **Latest academic year GPA** is highly overall correlated with **latest semester GPA** and **Use of technology in studying**
- **Latest semester GPA** is highly overall correlated with **Latest academic year GPA** and **Use of technology in studying**
- **Use of technology in studying** is highly overall correlated with **Latest academic year GPA** and **Latest semester GPA**
- **Current education level** is highly imbalanced (66.7%)
- **Use of technology in studying** is highly imbalanced (90.2%)
- **Timestamp** has unique values

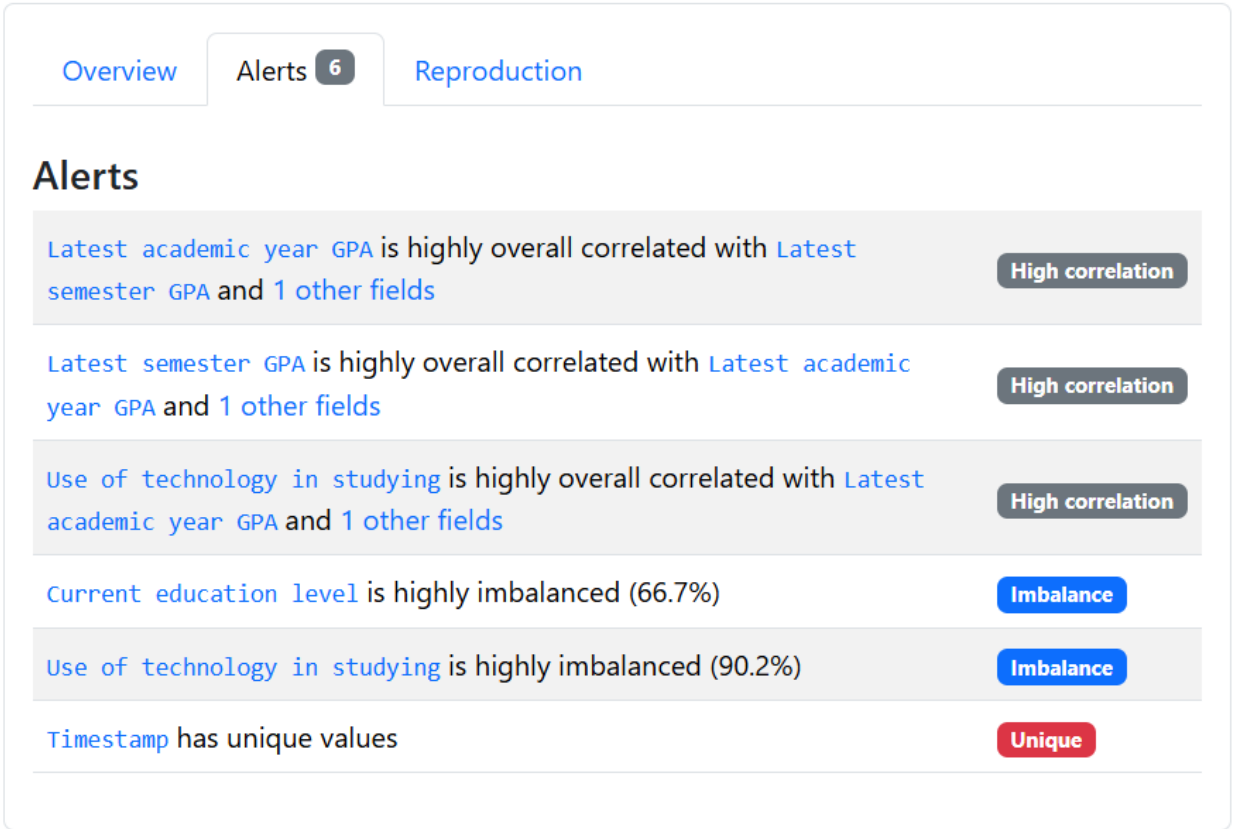

Figure 2.1: Alert report

2.2.2 Interactions

This plot about GPA correlation graph suggests that a student's overall academic performance during the year is a reasonably good indicator of their performance in the latest semester. Due to between correlation variables is high so we choose linear regression model.

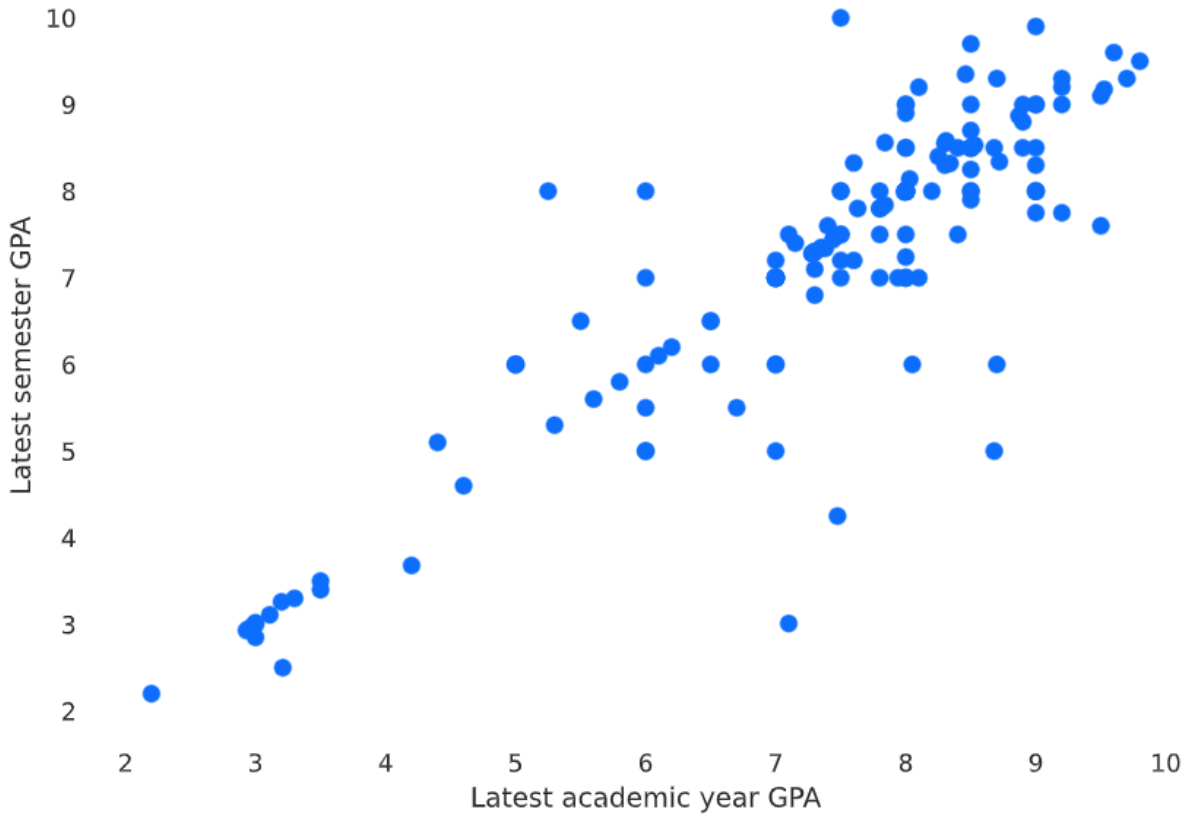

Figure 2.2: Latest GPA correlation graph

2.3 Linear Regression Mathematical Approach

In this problem, we need to predict GPA based on some features (study duration, physical activity frequency, etc). From a visual perspective, it is evident that GPA increases with study time, and that engaging in regular physical activity is correlated with higher academic performance... Based on this observation, we can model the relationship between input and output using a simple linear function:

$$y \approx \hat{y} = f(x) = w_0 + w_1 x_1 + ... + w_n x_n = W^T X$$

where $X = [1, x_1, x_2, ..., x_n]^T$ is feature vector, $W = [w_0, w_1, w2, ..., w_n]^T$ is weight vector, y is actual value and $\hat{y}$ is predicted value.

2.3.1 Cost Function

Prediction Error

In the regression problem, we want the prediction error e, which is the difference between the actual value y and the predicted value $\hat{y}$, to be as small as possible.

$$\frac{1}{2}e^2 = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(W^T X - y)^2$$

Cost Function

It is similar with all data point $(X^{(i)}, y^{(i)}), i = 1, 2, ..., m$ in data set. We aim to minimize the average prediction error, which corresponds to finding w the following function that yields the smallest possible value:

$$E(w) = \frac{1}{2m}\sum_{i=1}^{m}(W^T X^{(i)} - y^{(i)})^2$$

2.3.2 Solution for Linear Regression problem

Normal Equation

The value of W that minimizes E(w) is given in closed form by the equation:

$$W = (X^T X)^{-1} X^T y$$

or in matrix form:

$$\begin{bmatrix} w_0 \\ w_1 \\ ... \\ w_n \end{bmatrix} = \left(\begin{bmatrix} 1 & x_1^{(1)} & ... & x_n^{(1)} \\ 1 & x_1^{(2)} & ... & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & ... & x_n^{(m)} \end{bmatrix}^T \begin{bmatrix} 1 & x_1^{(1)} & ... & x_n^{(1)} \\ 1 & x_1^{(2)} & ... & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & ... & x_n^{(m)} \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & x_1^{(1)} & ... & x_n^{(1)} \\ 1 & x_1^{(2)} & ... & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & ... & x_n^{(m)} \end{bmatrix}^T \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

2.4 Some Mathematical Methodologies

2.4.1 Mean squared error (MSE)

Mean Square Error: measures the average of the squares of the errors- the difference between the real value and the predicted one

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where:

- n is the number of data points.
- Y_i is the actual value.
- $\hat{Y}_i$ is the predicted value.

2.4.2 Mean absolute error (MAE)

Mean Absolute Error is the sum of the absolute differences between predictions and actual values

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

- n is the number of data points.
- y_i is the actual value.
- $\hat{y}_i$ is the predicted value.

2.4.3 R-squared

R Square provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination.

The total sum of squares (proportional to the variance of the data):

$$SS_{tot} = \sum_{i=1}^{m}(y^i - \bar{i})^2$$

The sum of squares of residuals, also called the residual sum of squares

$$SS_{res} = \sum_{y=1}^{m}(y^i - h(x^i))^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

2.5 Implementation

2.5.1 Tool Introduction

Programming Language

In this project, we use Python because of its flexibility and powerful libraries for handling big data and machine learning.

Labraries

Library	Description
`pandas`	Data manipulation library for handling tabular data, such as loading, cleaning, and transforming datasets.
`numpy`	A library for numerical operations and handling multi-dimensional arrays. It provides support for mathematical functions and array manipulations.
`ydata_profiling`	A tool for generating data profiling reports that provide insights into the data distributions, correlations, and other EDA aspects.
`sklearn.model_selection`	Provides utilities for splitting datasets into training and testing sets for model evaluation and validation.
`sklearn.impute`	A module for handling missing data by using different imputation strategies like mean, median, or most frequent values.
`sklearn.preprocessing`	Contains preprocessing techniques such as feature scaling, encoding categorical variables, and transforming features for machine learning.
`sklearn.compose`	Allows the composition of multiple preprocessing steps into a single pipeline for efficient data transformation.
`sklearn.ensemble`	A module that provides ensemble learning methods like Random Forest, which combines multiple weak models to create a stronger model.
`sklearn.pipeline`	Facilitates the creation of a pipeline that chains preprocessing steps and models, making the workflow more organized and reusable.
`sklearn.base`	A base class for creating custom estimators and transformers in scikit-learn, enabling the building of specialized models and data transformations.
`sklearn.linear_model`	Contains algorithms for linear regression, a fundamental method for modeling the relationship between input features and a continuous output variable.
`sklearn.metrics`	Provides various metrics to evaluate the performance of a model, such as mean squared error (MSE), R-squared, and mean absolute error (MAE).

```
#Import necessary labraries
import pandas as pd
import numpy as np
from ydata_profiling import ProfileReport
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler,OrdinalEncoder,
   MultiLabelBinarizer, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin,
   RegressorMixin
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score,
   mean_absolute_error
```

2.5.2 Process

MultiLabelTransformer Class Creation

Class MultiLabelTransformer is created to convert multi-label data from a string of labels separated by ", " into a binary (0/1) matrix suitable for machine learning models. It also includes a function to retrieve feature names for later tasks. It is inherited from the base class **BaseEstimator** and **TransformerMixin** in **Scikit-learn** library.

```
class MultiLabelTransformer(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.mlb = MultiLabelBinarizer()

    def fit(self, X, y=None):
        X_series = pd.Series(X.squeeze())
        X_series = X_series.apply(lambda x: x.split(', ') if isinstance
           (x, str) else x)
        self.mlb.fit(X_series)
        return self

    def transform(self, X):
        X_series = pd.Series(X.squeeze())
        X_series = X_series.apply(lambda x: x.split(', ') if isinstance
           (x, str) else x)
        return self.mlb.transform(X_series)

    def get_feature_names_out(self, input_features=None):
        return [f"Preferred_study_method_{cls}" for cls in self.mlb.
           classes_]
```

BoundedRegressor Creation

This class is also inherited from the base class **BaseEstimator** and **TransformerMixin** in **Scikit-learn** library. Its task is to train on training set, predict the output value based on model parameters and restrict the predicted result to a specific range, which in this case is from 0 to 10 (GPA scale).

```
#Initialize the model with constraints
class BoundedRegressor(BaseEstimator, RegressorMixin):
    def __init__(self, regressor, min_val=0, max_val=10):
        self.regressor = regressor
        self.min_val = min_val
        self.max_val = max_val
#Train and predict
    def fit(self, X, y):
        self.regressor.fit(X, y)
        return self

    def predict(self, X):
        predictions = self.regressor.predict(X)
        return np.clip(predictions, self.min_val, self.max_val)
```

Read and analyze the data

Pandas library is used to read data from file csv. To facilitate analysis, we have used the Profilereport library to observe the correlation between the data. Then, we have to used the **Pandas profiling** library to print the HTML report file. Finally, we remove the white spaces between the columns to ensure consistency and to correctly identify the target variable as **"Latest semester GPA"**, in preparation for building the machine learning model.

```
#Read data
data = pd.read_csv('gpa-collections.csv')
profile = ProfileReport(data, title="Student Score Report")
# profile.to_file('report_csv.html')

target = "Latest semester GPA"
data.columns = data.columns.str.strip()
```

Data Splitting

We split the data into a training set and a test set with a ratio of 80:20, respectively. **"Timestamp"** - Redundant data and **"Latest semester GPA"** - target data columns are removed in the training set.

```
x = data.drop(columns=[target, "Timestamp"], axis=1)
y = data[target]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size
   =0.2, random_state=42)
```

Classification of features

The features are classified into four groups: **Numeric features**: Continuous numerical values,**Ordinal features**: Categorical values with a specific order, **Multilabel categorical features**: A feature that can have multiple values simultaneously,**Single-label categorical features**: Categorical values without any specific order.

```
# Numeric features
num_features = ["Latest academic year GPA"]

# Ordinal features
ord_features = [
    "Current education level",
    "Daily study duration",
    "Study break frequency",
    "Exam preparation approach",
    "Daily sleep duration",
    "Weekly exercise frequency",
    "Social media usage during study time",
    "Gender",
    "Use of technology in studying",
    "Lack of focus in studying",
    "Phone usage while studying"
]

# Multilabel categorical features
multi_nom_features = ["Preferred study method"]

# Single-label categorical features
single_nom_features = [
    "Most effective study time",
    "Dietary Habits",
    "Preferred study environment"]
```

Define the values for the ordinal features

Each list defines the order of possible values for each ordinal feature. This order will be used in the encoding process to maintain the ordinal relationship.

```
education_levels = ["Middle school", "High school", "University", "
   Graduated"]
daily_study_duration = ["< 1 hour", "1-2 hours", "2-4 hours", "> 4
   hours"]
study_break_freq = ["Almost no breaks", "5 min/hour", "10-15 min/hour",
    "15+ min/hour"]
exam_preparation = ["No preparation", "1 day before", "1-2 weeks before
   ", "> 2 weeks before"]
daily_sleep_duration = ["< 4 hours", "4-6 hours", "6-8 hours", "> 8
   hours"]
weekly_exercise_freq = ["Never", "1-2 times", "3-4 times", "5+ times"]
```

```
social_media_usage = ["Never", "< 15 min/hour", "15-30 min/hour", "> 30
    min/hour"]
gender = ["Male", "Female"]
use_technology = ["Yes", "No"]
focus = ["Yes", "No"]
phone_usage = ["Yes", "No"]
```

Define the values for the single-label categorical features

These lists define the possible values for each single-label categorical feature.

```
effective_study_time =["Morning", "Afternoon", "Evening", "Late night"]
diet_habits = ["No specific diet", "Balanced diet", "Special diet (
   Vegan, Keto, etc.)"]
study_env = ["Silent", "Soft background music", "Noisy (cafe, crowded
   library, etc.)"]
```

Creation of a processing pipeline for numeric features

During data collection, the presence of missing values is unavoidable. We handled this issue by imputing missing values by the median and standardizing the data to a normal distribution that has **mean** = 0 and **std** =1 to improve accuracy.

```
num_transformer = Pipeline(steps=[("imputer", SimpleImputer(strategy="
   median")),("scaler", StandardScaler())])
```

Creation of a processing pipeline for ordinal features

It is similar to numeric feature. In the first step, we impute missing values by the most frequent value and in the second step, we convert ordinal value into ordered integers.

```
ord_transformer = Pipeline(steps=[("imputer",SimpleImputer(strategy="
   most_frequent")),
    ("encoder", OrdinalEncoder(categories=[
        education_levels,
        daily_study_duration,
        study_break_freq,
        exam_preparation,
        daily_sleep_duration,
        weekly_exercise_freq,
        social_media_usage,
        gender,
        use_technology,
        focus,
        phone_usage,
    ]))
])
```

Creation of a processing pipeline for multi-label categorical features

We impute missing values by the most frequent value and use class **MultiLabeTransformer** that has been created above to encode into binary matrix.

```
multi_nom_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", MultiLabelTransformer())
])
```

Creation of a processing pipeline for single-label categorical features

The **SimpleImputer** is used to fill missing values with the most frequent value, followed by the conversion of categorical values into a binary matrix using one-hot encoding. When encountering a new value (not present in the predefined categories), the encoder returns an all-zero vector for that label.

```
single_nom_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(categories=[
        effective_study_time,
        diet_habits,
        study_env,
    ], handle_unknown="ignore"))
])
```

Combination of all preprocessing pipelines

The **ColumnTransformer** combines all preprocessing pipelines by applying each transformer to its corresponding set of columns.

```
preprocessor = ColumnTransformer(transformers=[
    ("num_feature", num_transformer, num_features),
    ("ord_feature", ord_transformer, ord_features),
    ("multi_nom_feature", multi_nom_transformer, multi_nom_features),
    ("single_nom_feature", single_nom_transformer, single_nom_features)
        ,
])
```

Creation of complete pipeline

The complete pipeline consists of two main components: a data preprocessing step and a linear regression model with output constrained between 0 and 10.

```
reg = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("model", BoundedRegressor(LinearRegression(), min_val=0, max_val
        =10))
])
```

Model Training

```
reg.fit(x_train, y_train)
```

Retrieve the coefficients corresponding to the feature names.

```
# Get feature and coefficient
feature_names = reg.named_steps['preprocessor'].get_feature_names_out()
coefficients = reg.named_steps['model'].regressor.coef_
intercept = reg.named_steps['model'].regressor.intercept_
# Create coefficient DataFrame
coef_df = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': coefficients
})
```

Print the result

```
# Print coefficients and intercept
print("Feature Coefficients:")
print(coef_df.sort_values(by = "Coefficient",ascending = False))

# Make predictions on the test set
y_predict = reg.predict(x_test)
# Print predicted vs actual values
for i, j in zip(y_predict, y_test):
    print(f"Predicted value: {i:.2f}. Actual value: {j}")

# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_predict)
mse = mean_squared_error(y_test, y_predict)
r2 = r2_score(y_test, y_predict)

# Plot predicted vs actual value
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_predict, color='blue', alpha=0.6)
plt.plot([0, 10], [0, 10], color='red', linestyle='--', linewidth=2)
plt.xlabel("Actual GPA")
plt.ylabel("Predicted GPA")
plt.title("Predicted vs Actual GPA")
plt.grid(True)
plt.tight_layout()
plt.show()
# Print evaluation metrics
print(f"\nEvaluate model:")
print(f"MAE: {mae}")    # Mean Absolute Error
print(f"MSE: {mse}")    # Mean Squared Error
print(f"R2: {r2}")      # R-squared (coefficient of determination)
```

Chapter 3

Results

3.1 Coefficients corresponding to the features

Feature	Coefficient
Latest academic year GPA	1.537
Exam preparation approach	0.433
Preferred study method: Tutoring	0.362
Lack of focus in studying	0.232
Study break frequency	0.144
Preferred study environment: Silent	0.138
Current education level	0.124
Daily study duration	0.104
Preferred study environment: Soft background music	0.081
Most effective study time: Afternoon	0.068
Social media usage during study time	0.064
Dietary Habits: No specific diet	0.036
Daily sleep duration	0.028
Preferred study method: Online learning	0.014
Use of technology in studying	0.000
Dietary Habits: Balanced diet	-0.011
Dietary Habits: Special diet (Vegan, Keto, etc.)	-0.025
Weekly exercise frequency	-0.055
Preferred study method: Self-study	-0.071
Most effective study time: Late night	-0.071
Preferred study environment: Noisy (cafe, crowded library, etc.)	-0.219
Most effective study time: Morning	-0.223
Gender	-0.234
Phone usage while studying	-0.251
Preferred study method: Group study	-0.295

3.2 Predicted vs Actual GPA

As you can see, this comparison shows how close the model's predictions are to the actual GPA values. Ideally, predicted values should be as close as possible actual values.

There are a few points that deviate somewhat from the main cluster. For instance, there's a point around (5, 8) where a student had a moderate academic year GPA but a relatively high latest semester GPA. Conversely, there are points around (8, 6) and (9, 6) where students had high academic year GPAs but somewhat lower latest semester GPAs.

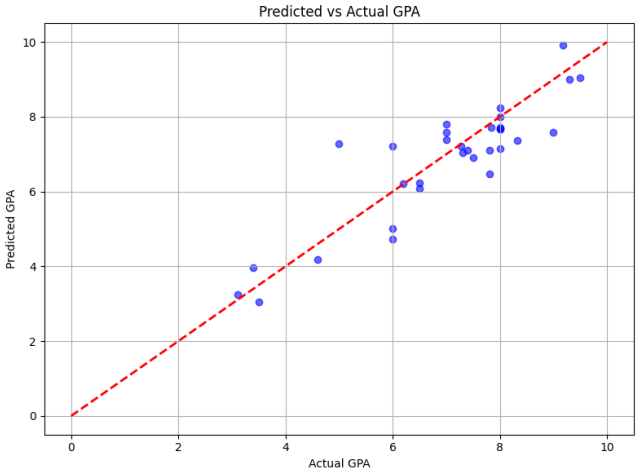

Figure 3.1: Predicted vs Actual GPA

3.3 Model Evaluation

We are evaluate model performance using metrics:

- MAE (Mean Absolute Error)
- MSE (Mean Squared Error)
- R^2 (R-squared): Coefficient of determination, measures the goodness of fit of the model (value from 0 to 1, closer to 1 is better)

Here are the evaluate model results:

- MAE: 0.5972162257393667
- MSE: 0.596014928318773
- R^2: 0.7751393674715464

Chapter 4

Discussion and Future work

The model performed well in term of accuracy, with with quite high R^2 and low MSE. However, there are some limitations to consider. The dataset used in this project may not capture all possible variables that influence GPA. For instance, a candidate's GPA can be affected by their health before the exam, which is not mentioned in data set. Moreover, the relationship between GPA and factors may not be linear. To improve the model's prediction accuracy and generalizability, several steps can be taken in future work: In the term of data set, the quality and quantity of the data could be enhanced by including additional features such as mental health factors, extracurricular activities, or more detailed academic history. This could provide a more comprehensive view of the factors that influence GPA. While linear regression is a good starting point, more advanced models such as Polynomial Regression ,Random Forest Regression, Support Vector Regression or even neural networks could better capture complex relationships between the predictors and GPA. By exploring these avenues, future research could enhance the robustness of GPA prediction models, making them more accurate and applicable to a broader range of students and educational contexts.

Chapter 5

Conclusion

This research aims to analyze the impact of study behaviors on Grade Point Average (GPA). In this research, we can see that coefficients of **Latest academic year GPA** and **Exam preparation approach** are two of the highest, that means these had a more significant influence on GPA. Moreover, some negative factors such as **studying in a noisy enviroment** and **lack of focus in studying** will reduce study effectiveness, leading to a low GPA. This reflects part of the reality.

Although the results are promising, the study has certain limitations, as discussed earlier. We have also proposed some solutions, which we plan to implement and compare with the current model in future research.

Ultimately, this project lays the foundation for further research in this area, offering valuable insights into the factors affecting GPA and the potential for developing more robust and personalized prediction models in educational settings.

Bibliography

[1] Obsie, Efrem & Adem, Seid. (2018). Prediction of Student Academic Performance using Neural Network, Linear Regression and Support Vector Regression: A Case Study. International Journal of Computer Applications. 180. 39-47. 10.5120/ijca2018917057.

[2] Zhu, Weijia. (2024). High school student GPA prediction by various linear regression models. Theoretical and Natural Science. 52. 153-162. 10.54254/2753-8818/52/2024CH0136.

[3] Sarmento, Rui & Costa, Vera. (2017). Introduction to Linear Regression. 10.4018/978-1-68318-016-6.ch006.

[4] Kumari, Khushbu & Yadav, Suniti. (2018). Linear regression analysis study. Journal of the Practice of Cardiovascular Sciences. 4. 33. 10.4103/jpcs.jpcs_8_18.

[5] Falat, Lukas & Piscová, Terézia. (2022). Predicting GPA of University Students with Supervised Regression Machine Learning Models. Applied Sciences. 12. 8403. 10.3390/app12178403.