Project 3

Linear Regression

Student: Nguyen Minh Nhat
Student ID: 22127309
Class: 22CLC05
Applied Mathematics and Statistics
Department of Information and Techonology
University of Science, Viet Nam National University Ho Chi Minh City
July 2024

Abstract

This project undertakes a thorough exploration and analysis of a dataset to develop linear regression models aimed at uncovering variables that may significantly impact the target outcome. The exploratory data analysis (EDA) leverages Python libraries including Pandas, Seaborn, and Matplotlib, while the linear regression models are implemented exclusively with Numpy. This report presents discussion of the EDA methods, the process of model construction, and the evaluation of model performance.

Contents

Acknowledgement

This project has been an incredible learning experience during my time at university. I have learned a lot from this project. I would like to express my gratitude toward my lecterers for this project. I am also grateful to StackOverflow, GitHub, and other websites for their invaluable resources. Special thanks to my classmates, Nguyen Long and Nguyen Trieu Khoang, for their brilliant ideas and insightful suggestions.

1 Introduction

This project focuses on exploring a dataset, analyzing the data, and building linear regression models bases on many aspects of the dataset; and evaluating the models to figure out factors that potentially affect the target variable. EDA is conducted mainly using Python libraries Pandas, SeaBorn, and Matplotlib while Linear Regression models are purely builts using Numpy. This report will provide a detailed explanation of exploratory data analysis, model building, and evaluation.

2 Libraries

2.1 Pandas [1]

Python Data Analysis Library

$(Class)\ Dataframe$:
A 2-dimensional labeled data structure with columns of potentially different types. Similar to a spreadsheet or SQL table, or a dictionary of Series objects.

$(Method)\ DataFrame.iloc[\,]$:
Purely integer-location based indexing for selection by position.

$(Method)\ DataFrame.corr() \rightarrow Dataframe$:
Compute pairwise correlation of columns, excluding NA/null value, with Pearson correlation method as default.

$(Method)\ DataFrame.describe() \rightarrow Dataframe$:
Generate descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values.

$(Function)\ read\_csv() \rightarrow Dataframe$:
Read a comma-separated values (csv) file into DataFrame.

2.2 Matplotlib [2]

Python 2D plotting library

$(Method)\ plt.figure()$:
Create a new figure, or activate an existing figure.

$(Method)\ plt.show()$:
Display all open figures.

2.3 Seaborn [3]

Python data visualization library

$(Method)\ sns.heatmap()$:
Plot rectangular data as a color-encoded matrix.

$(Method)\ sns.joinplot()$:
Draw a plot of two variables with bivariate and univariate graphs.

3 Project Implementation

3.1 Model Construction

The linear regression model is constructed using the following formula:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$$ [4]

In this project, the error term *epsilon* is ommited for simplicity. The basic formula for the linear regression model is:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5$$

where:

- Y is the target variable (Performance Index),
- X_1 is Hours Studied,
- X_2 is Previous Scores,
- X_3 is Extracurricular Activities,
- X_4 is Sleep Hours,
- X_5 is Sample Question Papers Practiced,
- β_0 is the intercept,
- $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ are the coefficients of the features,

3.2 (class) OLSLinearRegression

This class implements Ordinary Least Squares Linear Regression

The OLSLinearRegression class implements Ordinary Least Squares Linear Regression. It stores the model weights and an optional linear transformation function. The class has the following methods:

(method) preprocess(X: np.ndarray) → np.ndarray:

Preprocesses the input data. If a linear function is provided, it applies this function to the input. It then adds a column of ones to the input data to account for the intercept term.

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}$$

(method) fit(X: np.ndarray, y: np.ndarray) → np.ndarray:
Fits the model to the data using the Ordinary Least Squares method. It calculates the optimal weights using the pseudoinverse of the preprocessed input data. Standard formulars for calculating the psuedoinverse are:

$$\begin{cases} \hat{\beta} = (X^T X)^{-1} X^T \\ \hat{\beta} = X^T (X X^T)^{-1} \end{cases} \quad [5]$$

which can face cases where $X^T X$ or $X X^T$ is not invertible. To avoid this, this implementation utilizes numpy's *np.linalg.pinv()* function to calculate the pseudoinverse, which uses the Singular Value Decomposition (SVD) method to provide faster and more stable results.
The inverse is then used to calculate the optimal weights with np.dot() function:

$$self.weights = np.dot(\hat{\beta}, y)$$

(method) predict(X: np.ndarray) → np.ndarray:
Predicts output values for the given input data. It first checks if the model has been fitted, then applies the preprocessing and calculates the predictions using the learned weights byh the following numpy matmul() function:

$$\hat{y} = X \ @ \ self.weights$$

(method) get_formula(self, features: pd.Index) → str:
Generates a string representation of the linear regression formula, including the learned coefficients and feature names.

3.3 Other functions

mean_absolute_error(y_true: np.ndarray, y_pred: np.ndarray) → float:
Calculates the mean absolute error between the true and predicted values.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y - \hat{y}| \quad [6]$$

The numpy implementation of this function is:

np.mean(np.abs(y_true - y_pred))

apply_coef_and_power(x: np.ndarray, coef: np.ndarray, power: np.ndarray) → np.ndarray:
Applies the coefficients and powers to the input data. It calculates the polynomial features of the input data.

Transformation: For each feature i in input x:

$$x_{i,\text{new}} = \text{coef}_i \cdot x_i^{\text{power}_i}$$

Implementation: Using NumPy's element-wise operations:

$$x[:, i] = \text{coef}[i] \cdot \text{np.power}(x[:, i], \text{power}[i])$$

This applies the transformation to each column i of x.

k_fold_cross_validation(X : np.ndarray, y : np.ndarray, k : int = 5, linear_function: typing.Callable |None = None) → np.float64: Performs k-fold cross-validation on the input data. It splits the data into k folds, trains the model on k-1 folds, and evaluates it on the remaining fold. The function returns the average mean absolute error across all folds.

Algorithm:

$$\text{MAE} = \frac{1}{k} \sum_{i=1}^{k} \text{MAE}_i$$

Where MAE_i is the Mean Absolute Error for the i-th fold.

Implementation:

Split data into k folds For each fold i:

- Train on k-1 folds
- Validate on remaining fold
- Compute MAE_i

Return average MAE across all folds

Returns: Average Mean Absolute Error (float)

4 (Question 1)Exploratory Data Analysis

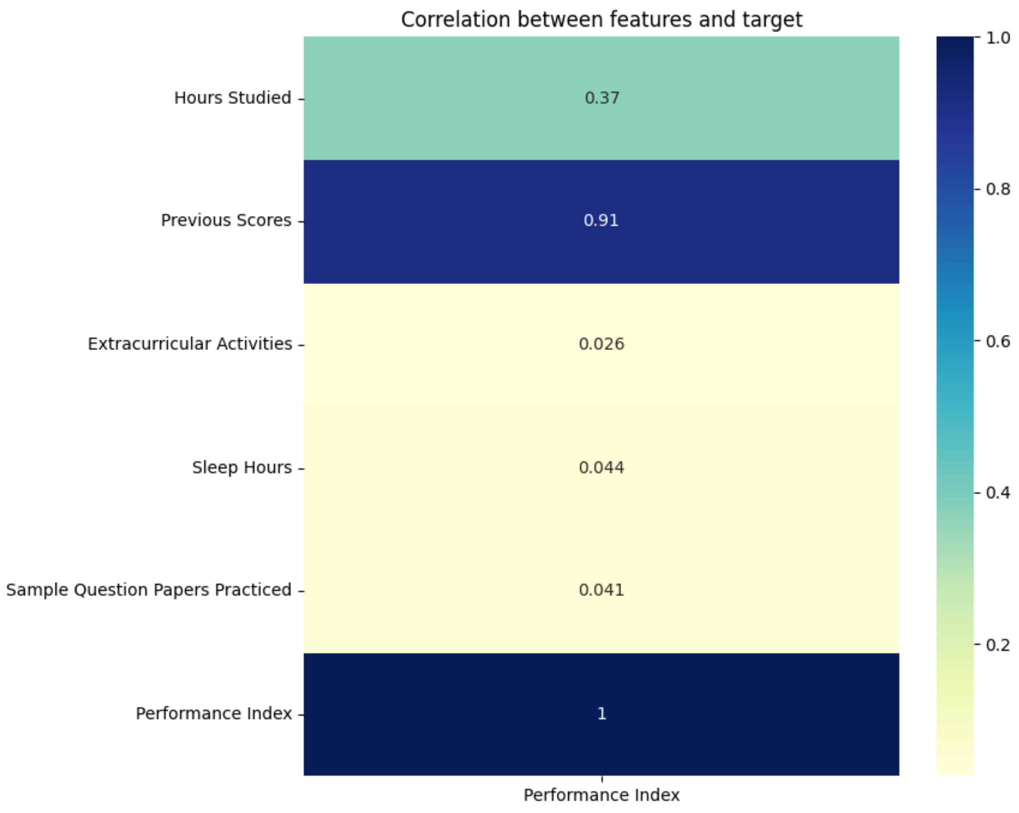
4.1 Descriptive statistics

	Hours Studied	**Previous Scores**	**Extracurricular Activities**	**Sleep Hours**	**Sample Question Papers Practiced**	**Performance Index**
count	9000	9000	9000	9000	9000	9000
mean	4.976444	69.396111	0.493667	6.535556	4.590889	55.136333
std	2.594647	17.369957	0.499988	1.695533	2.864570	19.187669
min	1.0	40.0	0.0	4.0	0.0	10.0
25%	3.0	54.0	0.0	5.0	2.0	40.0
50%	5.0	69.0	0.0	7.0	5.0	55.0
75%	7.0	85.0	1.0	8.0	7.0	70.0
max	9.0	99.0	1.0	9.0	9.0	100.0

Descriptive Statistics of the Dataset

The table suggests that the dataset distribution covers a wide range of values for each variable. The mean values of the variables are close to the median values, indicating that the data is not skewed; this is further supported by high standard deviations compared to data ranges.
The data between Previous Scores and Performance Index has a observable correlation, with all aspects of the dataset pose a degree of similarity to each other.

4.2 Features Correlation with target variable



The correlation matrix above utilizes the Pearson correlation coefficient, which ranges from -1 to 1, where:

- 1 indicating a perfect positive linear relationship,
- -1 indicating a perfect negative linear relationship, and
- 0 indicating no linear relationship between the two variables.

The correlation coefficients for each feature in relation to the target variable are as follows:

- Hours Studied: 0.37 (moderate positive correlation)
- Previous Scores: 0.91 (strong positive correlation)
- Extracurricular Activities: 0.026 (weak positive correlation)
- Sleep Hours: 0.041 (weak positive correlation)
- Sample Questions Papers Practiced: 0.041 (weak positive correlation)

These values suggest that all features have a positive correlation with the target variable. The strongest correlation is observed between previous scores and the target variable, indicating that past performance is a strong predictor of future outcomes. On the other hand, features such as extracurricular activities, sleep hours, and the number of sample question papers practiced show weak correlations, implying a lesser impact on the target variable.

4.3 Feature Distribution with target variable

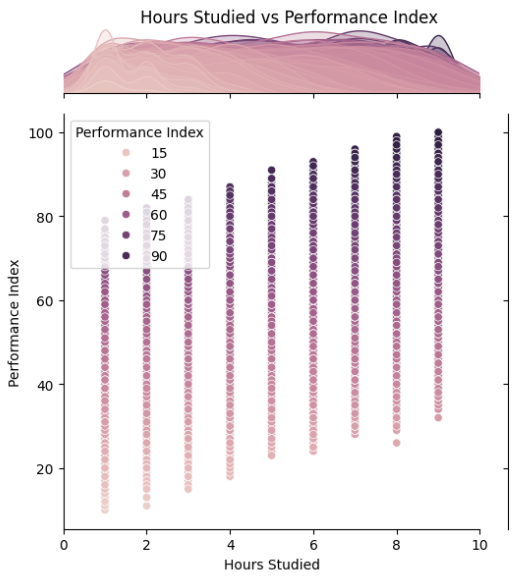
Figure	**Description**
	Each Hours Studied values points are distributed equally across the target variable, with a slight increase in the target variable as Hours Studied increases showing a slight positive correlation, this also be indicated by the color gradient from left to right.

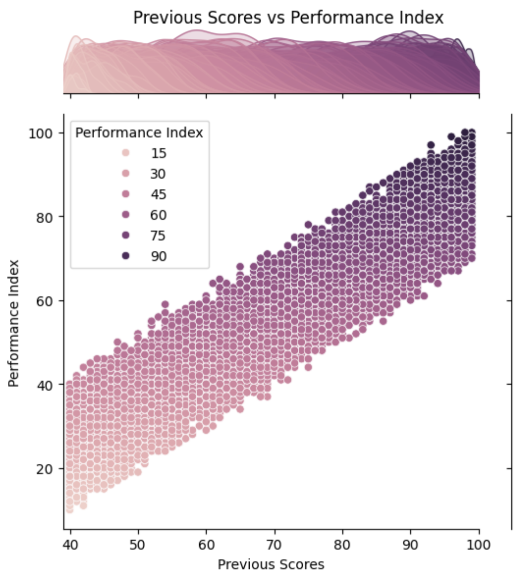
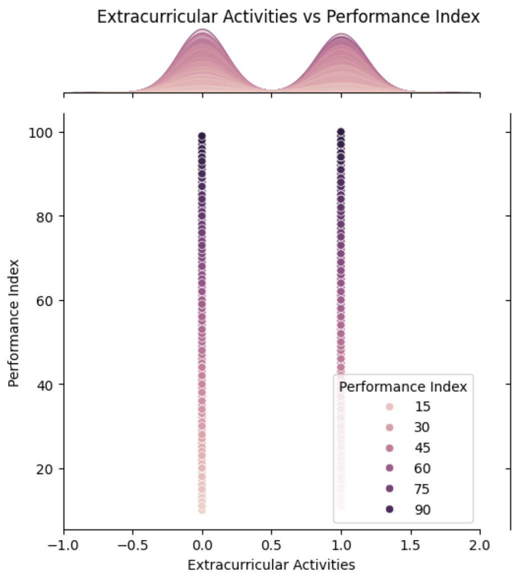
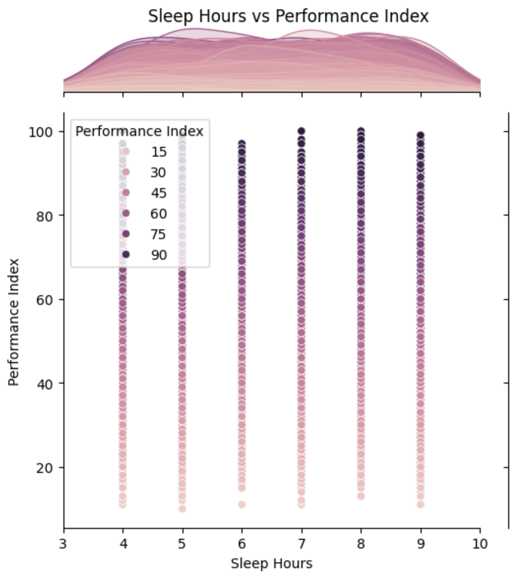
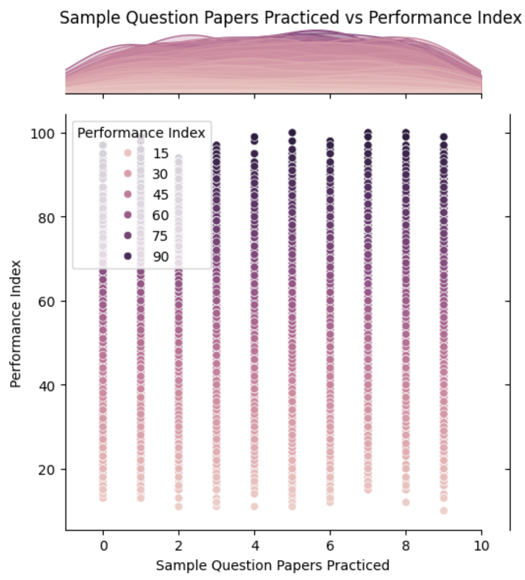
Figure	**Description**
	The Previous Scores values are evenly spread across the target variable, showing a clear increase in the target variable as Previous Scores rise, indicating a strong positive correlation. This is further evidenced by the linear trend of the data points and the visible color gradient from left to right.
	The Extracurricular Activities values are evenly spread, forming two vertical lines of equal size across all target variable values, showing no clear correlation. The color transition from bottom to top also reflects hardly any correlation.
	Like Extracurricular Activities, Sleep Hours values are evenly distributed across the target variable, with no clear correlation. The color gradient from bottom to top further supports this observation.

Figure	Description
	The Sample Question Papers Practiced values are evenly distributed across the target variable, with no clear correlation. The color gradient from bottom to top further supports this observation.

5 (Question 2a) Construct basic linear regression model

The model is construct using basic linear regression model with the following formula:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5$$

use the OLSLinearRegression class to fit the model to the training data and calculate the optimal weights. The model is then used to predict the target variable on the test data, and the mean absolute error is calculated. The formula of the linear regression model is:

$$\begin{aligned} y = {} & 2.852 \cdot \text{Hours Studied} \\ & + 1.018 \cdot \text{Previous Scores} \\ & + 0.604 \cdot \text{Extracurricular Activities} \\ & + 0.474 \cdot \text{Sleep Hours} \\ & + 0.192 \cdot \text{Sample Question Papers Practiced} \\ & - 33.969 \end{aligned}$$

The mean absolute error on the training set is 1.596.
Evaluation: The model has a mean absolute error of 1.596 on the training set, indicating that the model's predictions are, on average, 1.596 units away from the true values. This suggests that the model is relatively accurate in predicting the target variable based on the input features, which is impressive given the simplicity of the model.

6 (Question 2b) Single Feature Linear Regression Models

6.1 Cross-validation on single feature models

Before training the model, the data is shuffled once to ensure that there is no bias in the data. The cross-validation is performed on each feature individually, reshaping the input data to ensure compatibility with the model. The mean absolute error is calculated for each feature, and the results are displayed in a table.
The table below shows the mean absolute error for each feature when used as a single feature in the linear regression model.

Model with 1 feature	**MAE**
Hours Studied	15.452702
Previous Scores	6.618853
Extracurricular Activities	16.199082
Sleep Hours	16.188669
Sample Question Papers Practiced	16.187349

Mean Absolute Error for Single Feature Models

These average MAE values not always the same because of the randomness of the data shuffling. But no significant difference in the MAE values after times of running the code. So that can be confidently said that the order of the features in the table is consistent; and the Previous Scores feature has the lowest MAE, indicating that it is the best single feature for predicting the target variable.

6.2 Retrain the model with the best feature

The model is retrained using the best feature, which is the Previous Scores feature. The formula of the linear regression model with the best feature is:

$$y = 1.011 \cdot \text{Previous Scores} - 14.989$$

The mean absolute error on the test set is 6.544.
Evaluation: The model with the best feature, Previous Scores, has a mean absolute error of 6.544 on the test set. This indicates that the model's predictions are, on average, 6.544 units away from the true values. This is relatively high compared to the basic linear regression model, suggesting that although others feature have weak correlation with the target variable, they still contribute to the model's performance.

7 (Question 2c) Create Custom Linear Transformation

7.1 First Model: Normalization with k-fold cross-validation

The first model is designed to normalize the data using cross-validation based on the best feature, Previous Scores. The normalized data is then used to build a linear regression model.
The table below shows the cross-validation results for each feature, the coefficients, and the powers of the features.

No	Feature	Cross validation	Coefficient
1	Hours Studied	15.449913	0.428
2	Previous Scores	6.618343	1
3	Extracurricular Activities	16.195993	0.409
4	Sample Question Papers Practiced	16.185144	0.409
5	Sleep Hours	16.187000	0.409

Cross-validation Results and Coefficients

The coefficients and powers indices for the features are as follows:

$$\begin{cases} \text{coef} = \begin{bmatrix} 0.428 & 1 & 0.409 & 0.409 & 0.409 \end{bmatrix} \\ \text{power} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{cases}$$

7.2 Second Model: Normalization with Pearson correlation coefficient

The second model is designed to normalize the data using the Pearson correlation coefficient. The normalized data is then used to build a linear regression model.
The table below shows the Pearson correlation coefficient for each feature, the coefficients, and the powers of the features.

No	Feature	Pearson correlation coefficient	Coefficient
1	Hours Studied	0.37	0.407
2	Previous Scores	0.91	1
3	Extracurricular Activities	0.026	0.029
4	Sample Question Papers Practiced	0.041	0.045
5	Sleep Hours	0.041	0.045

Pearson Correlation Coefficient and Coefficients

The coefficients and powers indices for the features are as follows:

$$\begin{cases} \text{coef} = \begin{bmatrix} 0.407 & 1 & 0.029 & 0.045 & 0.045 \end{bmatrix} \\ \text{power} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{cases}$$

7.3 Third Model: Coefficient based on Pearson correlation coefficient rank

The third model is designed to apply coefficients based on the rank of the Pearson correlation coefficient. The normalized data is then used to build a linear regression model.
The table below shows the Pearson correlation coefficient rank for each feature, the coefficients, and the powers of the features.

No	Feature	Pearson correlation	Coefficient/Rank
1	Hours Studied	0.37	3
2	Previous Scores	0.91	4
3	Extracurricular Activities	0.026	1
4	Sample Question Papers Practiced	0.041	2
5	Sleep Hours	0.041	2

Pearson Correlation Coefficient Rank and Coefficients

The coefficients and powers indices for the features are as follows:

$$\begin{cases} \text{coef} = \begin{bmatrix} 3 & 4 & 1 & 2 & 2 \end{bmatrix} \\ \text{power} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{cases}$$

7.4 Fourth Model: Power indices based on Pearson correlation coefficient rank

The fourth model is designed to apply power indices based on the rank of the Pearson correlation coefficient. The normalized data is then used to build a linear regression model.
The table below shows the Pearson correlation coefficient rank for each feature, the power indices, and the coefficients of the features.

No	Feature	Pearson correlation	Power Indices/Rank
1	Hours Studied	0.37	3
2	Previous Scores	0.91	4
3	Extracurricular Activities	0.026	1
4	Sample Question Papers Practiced	0.041	2
5	Sleep Hours	0.041	2

Pearson Correlation Coefficient Rank and Power Indices

The coefficients and powers indices for the features are as follows:

$$\begin{cases} \text{coef} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\ \text{power} = \begin{bmatrix} 3 & 4 & 1 & 2 & 2 \end{bmatrix} \end{cases}$$

7.5 Fifth Model: Brute force to find best coefficients

The fifth model is designed to find the best coefficients from 0 to 7 with a power of all features set to $[0.9, 1, 1.1]$. The normalized data is then used to build a linear regression model.
The table below shows the best coefficients and power indices for the features.

No	Feature	Coefficient	Power indices
1	Hours Studied	2	1
2	Previous Scores	6	1
3	Extracurricular Activities	1	1
4	Sample Question Papers Practiced	1	1
5	Sleep Hours	1	1

Best Coefficients and Power Indices

The coefficients and powers indices for the features are as follows:

$$\begin{cases} \text{coef} = \begin{bmatrix} 2 & 6 & 1 & 1 & 1 \end{bmatrix} \\ \text{power} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{cases}$$

7.6 Evaluation of Custom Linear Transformation Models

The mean absolute error for each custom linear transformation model is calculated using cross-validation. The results are displayed in a table, and the best model is identified based on the lowest mean absolute error.
The table below shows the mean absolute error for each custom linear transformation model.

Model	MAE
Normalized cross-validation	2.248576
Normalized Pearson correlation	2.368353
Coefficient with correlation order	1.620954
Power with correlation order	5.152410
Brute force	1.620954

Mean Absolute Error for Custom Linear Transformation Models

The best model is Brute force with a mean absolute error of 1.620954.

7.7 Retrain the model with the best custom linear transformation

The model is retrained using the best custom linear transformation, which is the Brute force model. The formula of the best model is:

$$\begin{aligned} y = 1.426 &\cdot \text{Hours Studied} \\ &+ 0.170 \cdot \text{Previous Scores} \\ &+ 0.604 \cdot \text{Extracurricular Activities} \\ &+ 0.474 \cdot \text{Sleep Hours} \\ &+ 0.192 \cdot \text{Sample Question Papers Practiced} \\ &- 33.969 \end{aligned}$$

The mean absolute error on the test set is 1.599.
Evaluation: The model with the best custom linear transformation has a mean absolute error of 1.599 on the test set. This indicates that the model's predictions are, on average, 1.599 units away from the true values. This is slightly higher than the basic linear regression model but still relatively accurate in predicting the target variable based on the input features.

References

[1] pandas development team. pandas documentation. https://pandas.pydata.org/docs/reference/, 2024. [Online; accessed 16-August-2024].

[2] Matplotlib Development Team. Using matplotlib — matplotlib 3.9.2 documentation. https://matplotlib.org/stable/index.html, 2024. [Online; accessed 16-August-2024].

[3] seaborn development team. Api reference — seaborn 0.13.2 documentation. https://seaborn.pydata.org/api.html, 2024. [Online; accessed 16-August-2024].

[4] Wikipedia contributors. Linear regression — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Linear_regression&oldid=1239428210, 2024. [Online; accessed 16-August-2024].

[5] Wikipedia contributors. Moore–penrose inverse — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Moore%E2%80%93Penrose_inverse&oldid=1239124321, 2024. [Online; accessed 16-August-2024].

[6] Wikipedia contributors. Mean absolute error — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Mean_absolute_error&oldid=1216949643, 2024. [Online; accessed 16-August-2024].