VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

Report

Linear Regression

Course name: Applied Mathematics and Statistic

MTH00051_22CLC09

Student:
Nguyen Ho Dang Duy
22127085
22CLC09

Lecture:
Nguyen Ngoc Toan
Nguyen Van Quang Huy

Ho Chi Minh city, August 2024

Contents

1 Information

1.1 Student Information

- Full name: Nguyen Ho Dang Duy
- Student ID: 22127085
- Class: 22CLC09

1.2 Abstract

This report details the development and evaluation of predictive models for student performance using various features including study hours, previous academic scores, extracurricular activities, sleep hours, and practice with sample papers. I implement linear regression techniques and employ feature selection methods forward selection and backward elimination to identify the most significant predictors. Through k-fold cross-validation, we compare model performance and select the most accurate model. The findings highlight the key factors influencing student success and validate the effectiveness of different feature selection approaches.

2 Exploratory Data Analysis

This EDA step gives us an initial look at the data. You need to have a certain sense of what you have on hand before devising modeling strategies. EDA helps you visualize the complexity of the problem and outline the first steps that need to be taken.[1]

2.1 Data Overview

2.1.1 Features

1. **Hours Studied:**
   - **Description:** Number of hours student has studied.
   - **Data type:** Integer
2. **Previous Scores:**
   - **Description:** Previous academic scores of the student.
   - **Data type:** Integer
3. **Extracurricular Activities:**
   - **Description:** Binary indicator (0 or 1) of whether the student participated in extracurricular activities.
   - **Data type:** Boolean
4. **Sleep Hours:**
   - **Description:** Number of hours the student sleeps.
   - **Data type:** Integer
5. **Sample Question Papers Practiced:**
   - **Description:** Number of sample question papers practiced by the student.
   - **Data type:** Integer

2.1.2 Target Values

Performance Index

- **Description:** A measure of the student's overall performance. The index ranges from 10 to 100, with higher values indicating better performance.
- **Data type:** Float

2.2 Univariate Analysis

2.2.1 Histograms

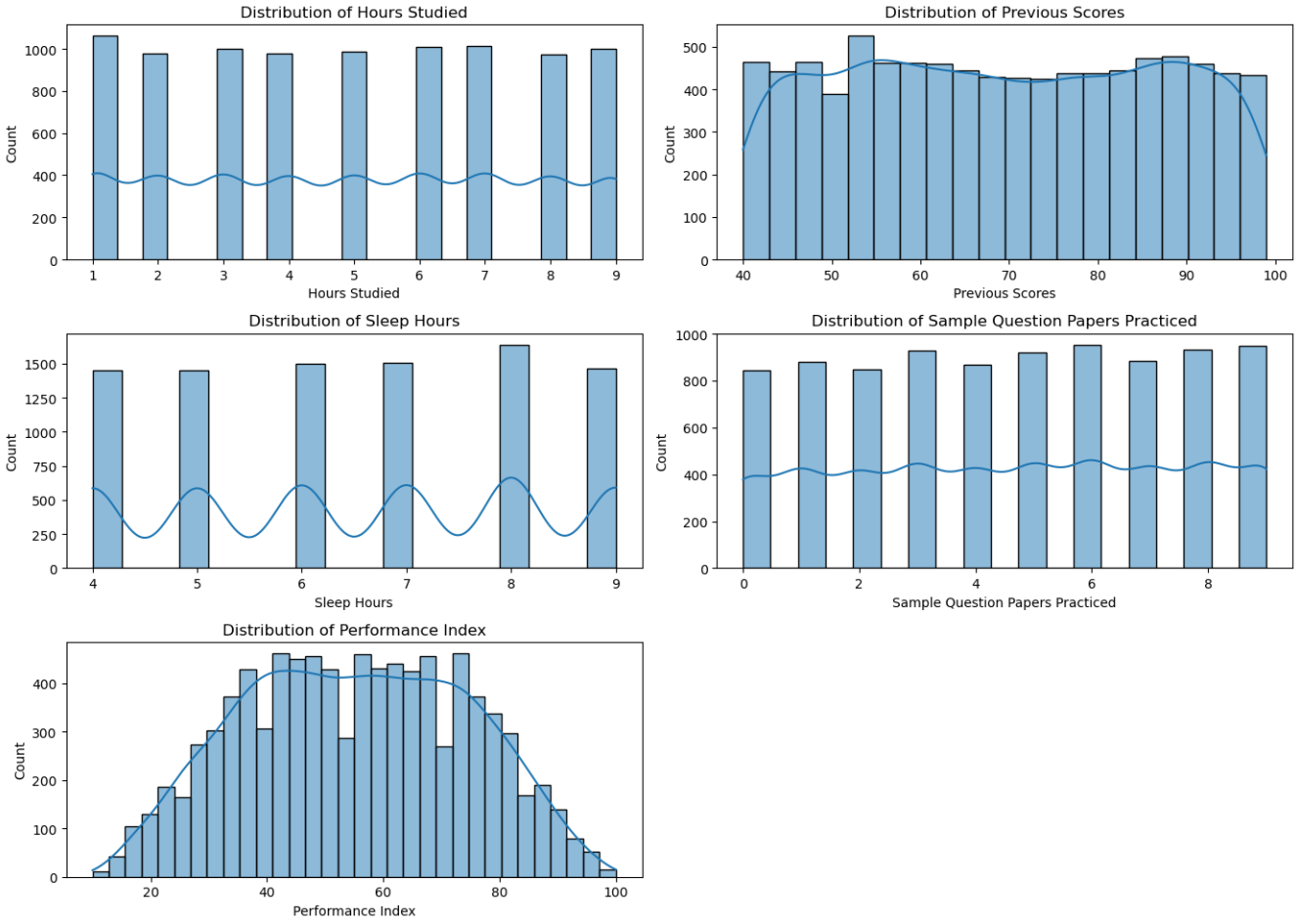


Figure 1: Distibution of numericals features

The distributions for Hours Studied, Previous Scores, Sleep Hours, and Sample Question Papers Practiced all indicate a fairly uniform or evenly spread-out pattern, except for the Performance Index, which follows a normal distribution. This uniformity in study habits and sleep does not seem to create significant outliers in performance, suggesting that other factors might contribute to variations in the Performance Index. The normal distribution of the Performance Index suggests that most students fall into an average performance category, with fewer students excelling or underperforming significantly.

2.2.2 Count Plot

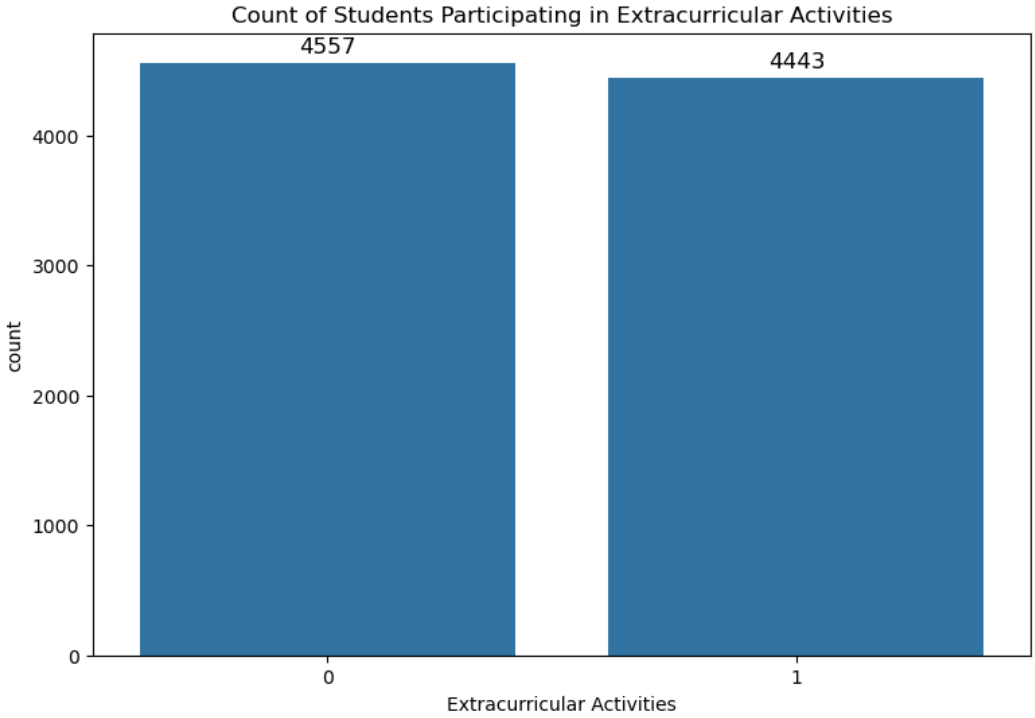


Figure 2: Count plot for Extracurricular Activities

The data suggests that the participation in extracurricular activities among students is quite balanced, with almost an equal number of students engaging in these activities as those who do not. This balance could imply that extracurricular activities are accessible and appealing to a significant portion of the student population, while still leaving room for improvement in encouraging more students to participate.

2.3 Bivariate Analysis

2.3.1 Correlation Matrix (Heatmap)

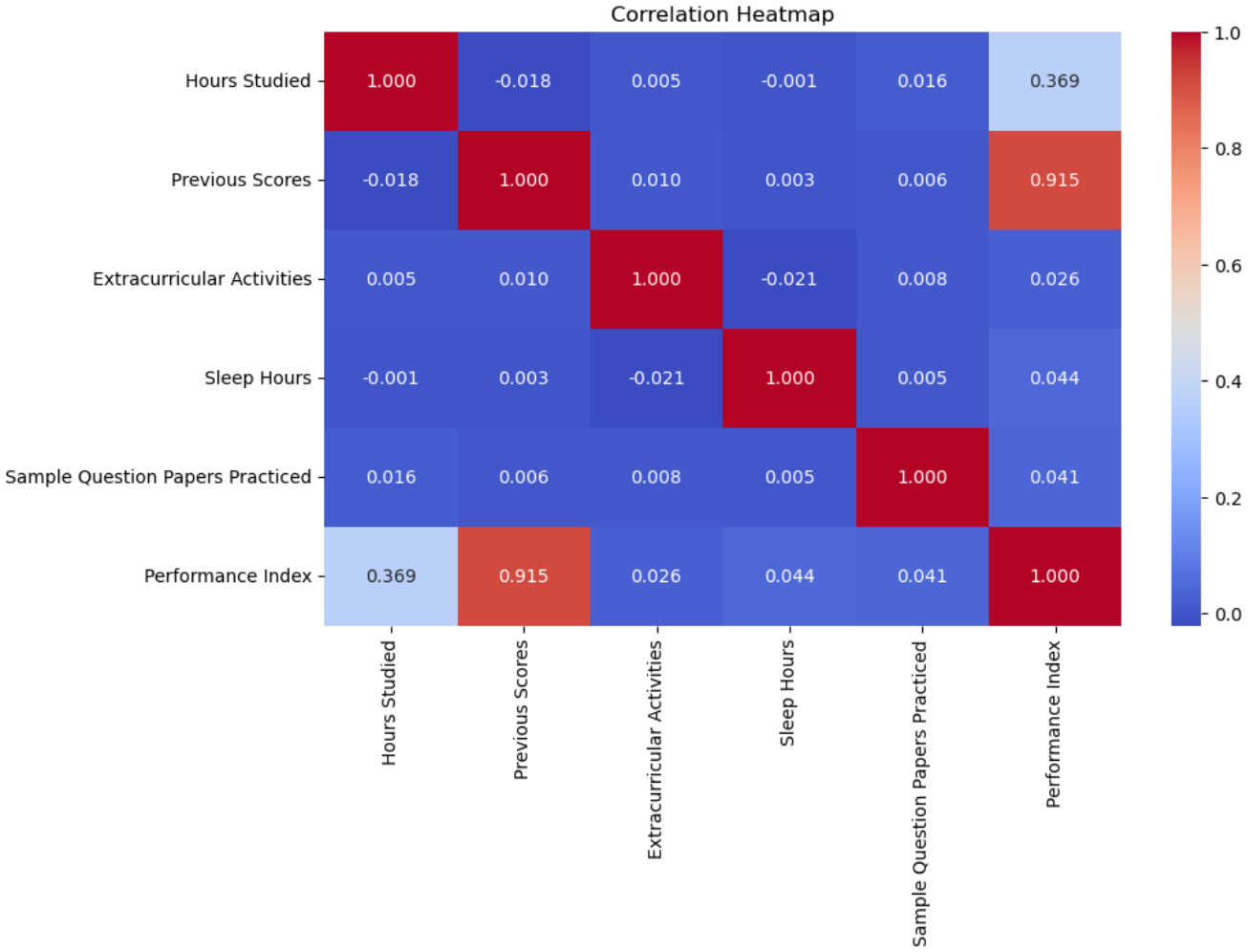


Figure 3: Correlation Matrix between the product of each features

- Previous Scores is the most influential factor on the Performance Index. This emphasizes the importance of maintaining good academic performance over time.
- Hours Studied also has a positive impact on the Performance Index, but it is not as significant as previous academic performance.
- Other factors like extracurricular activities, sleep, and practicing sample papers have little to no noticeable impact on academic performance.

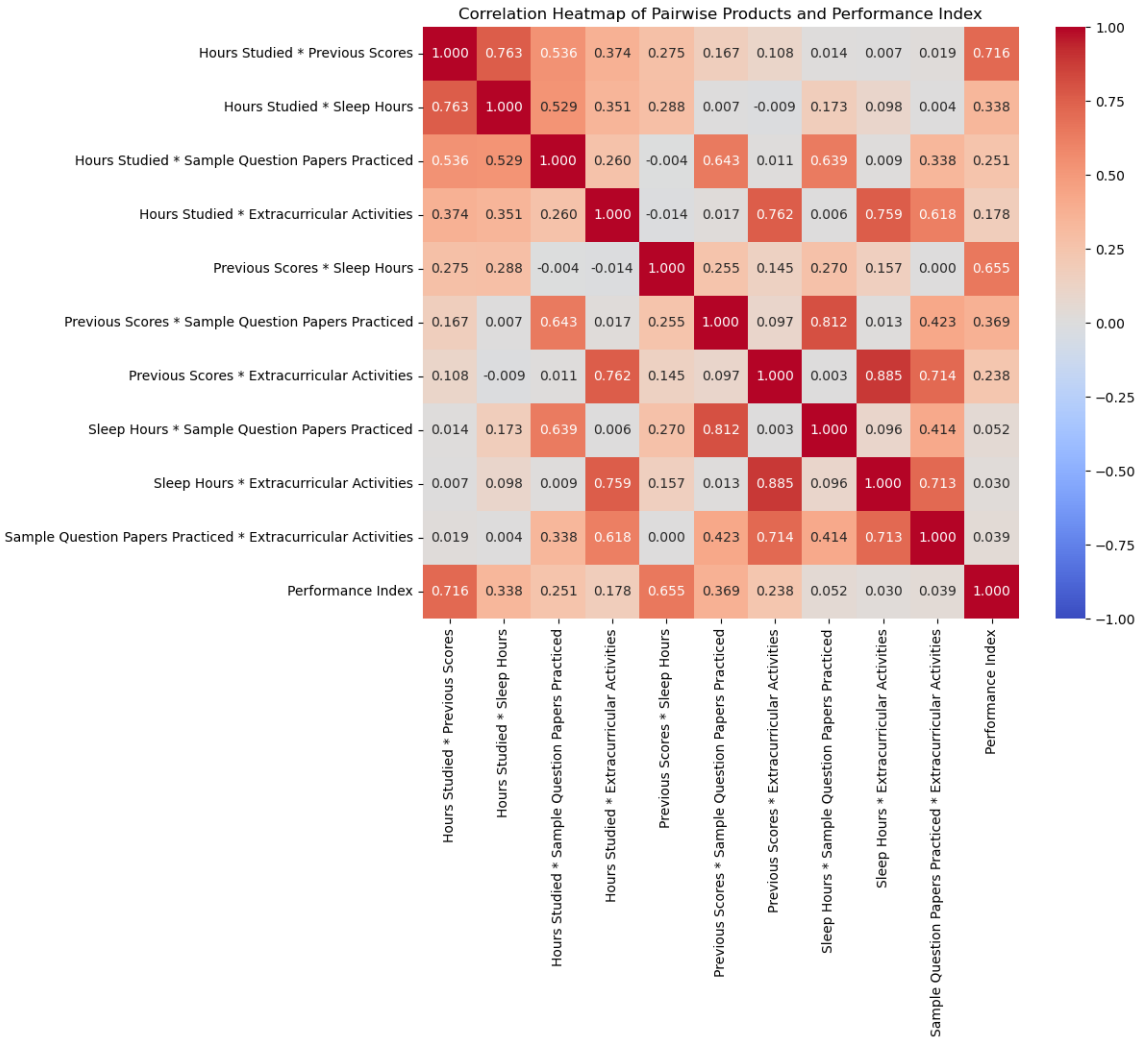


Figure 4: Correlation Heatmap of Pairwise Product Variables

In conclusion, the most important factors that positively influence student performance are hours studied, previous scores, and the practice of sample question papers. Sleep hours and extracurricular activities have supportive but weaker direct effects. Extracurricular activities might be beneficial in other indirect ways, such as fostering better sleep patterns or being correlated with students who already perform well academically.

2.3.2 Box Plots

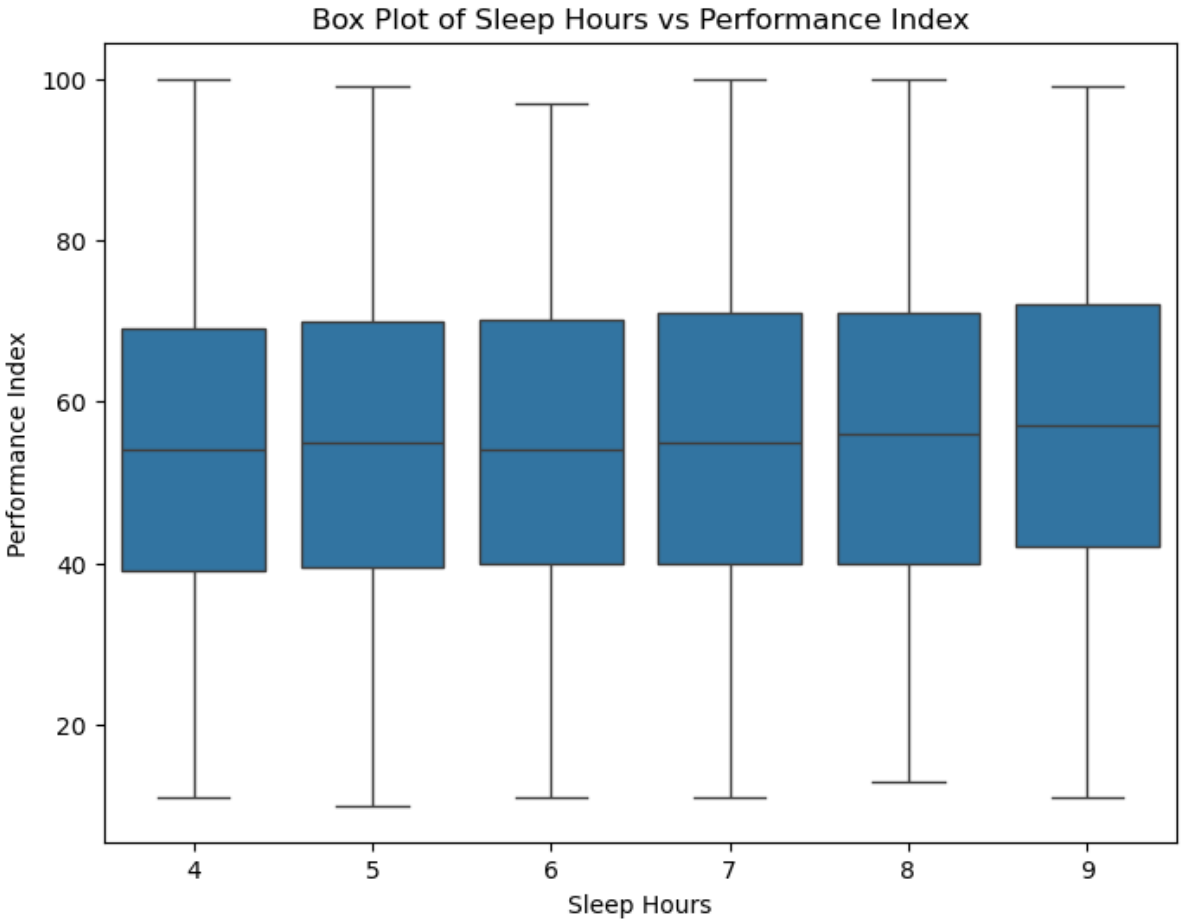


Figure 5: Box Plot of Sleep Hours vs Performance Index

There does not appear to be a strong relationship between sleep hours and performance index. The consistency in the medians and spread across different sleep hours suggests that the number of hours students sleep does not significantly affect their performance index.

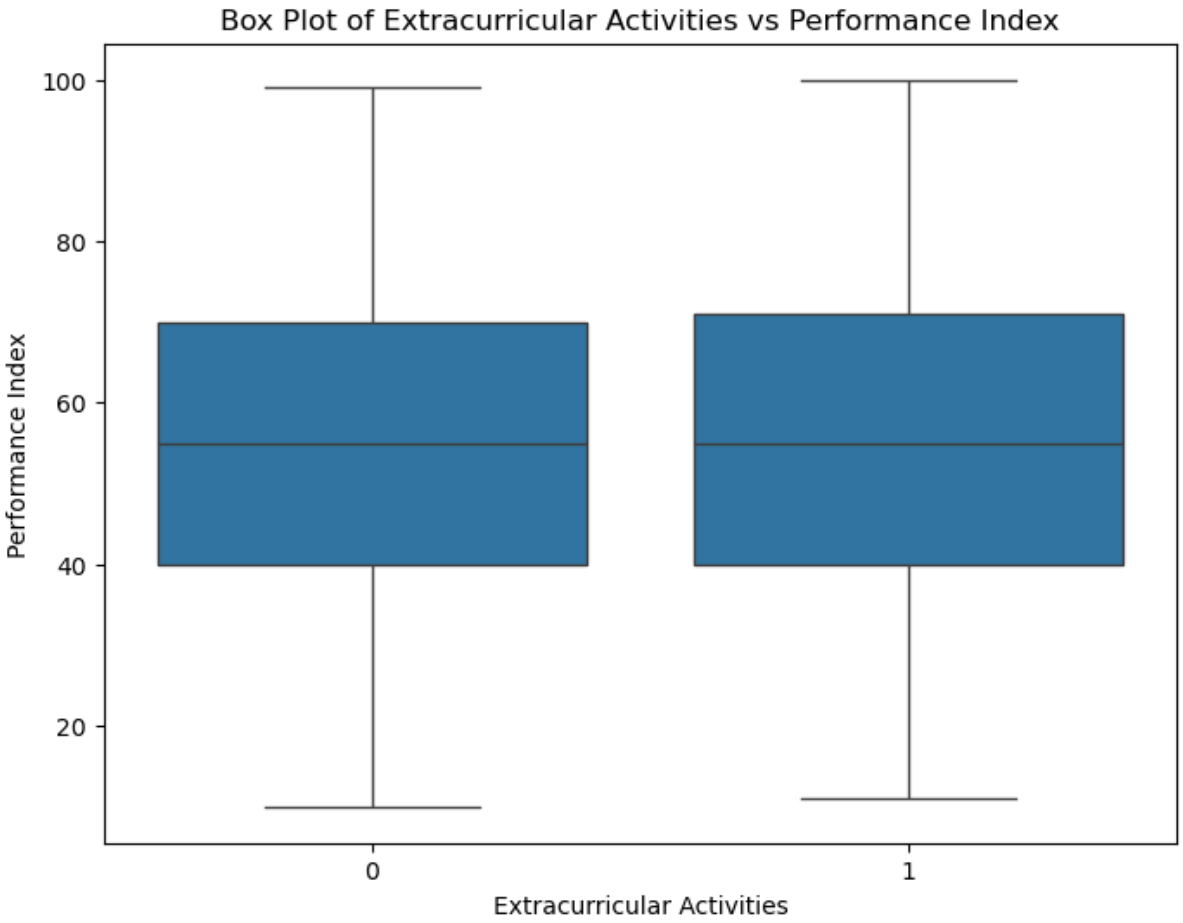
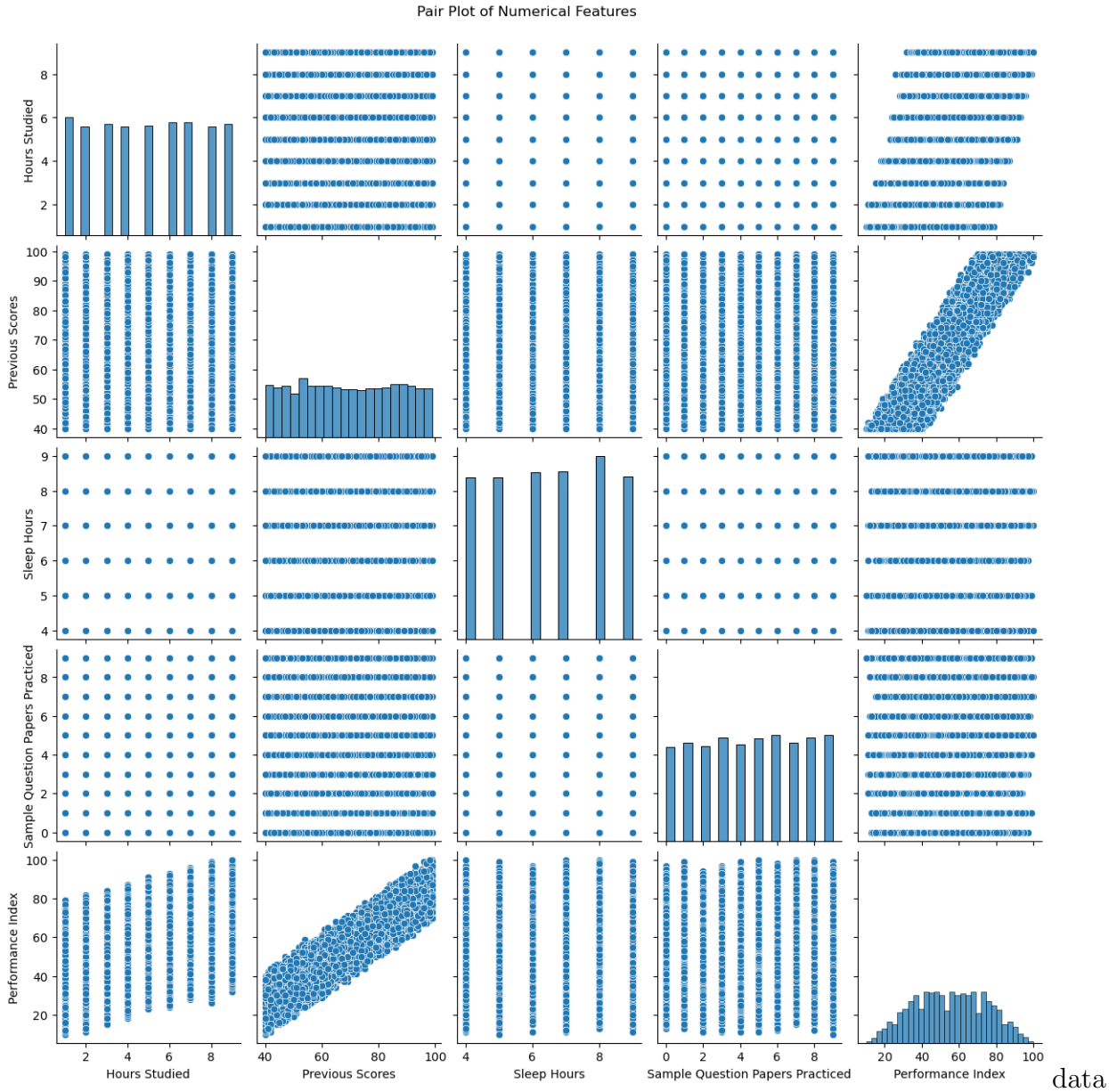


Figure 6: Box Plot of Extracurricular Activities vs Performance Index

Participation in extracurricular activities does not seem to have a significant impact on the performance index. Both groups exhibit very similar performance distributions, implying that being involved in extracurricular activities may not necessarily lead to higher or lower academic performance.

2.4 Multivariate Analysis



data

Figure 7: Pair Plot of Numerical Features [4]

- **Strong Relationship:** The only significant and strong relationship observed is between Previous Scores and Performance Index. This is consistent across all visualizations and confirms that past performance is the most reliable indicator of future performance in this dataset.
- **Weak or No Relationships:** The other variables, such as Hours Studied, Sleep Hours, and Sample Question Papers Practiced, do not show any strong linear relationship with the Performance Index or with each other. This implies that these factors, at least in the context of this dataset, have limited influence on performance outcome
- **Distribution Insights:** The distributions of the variables, especially Performance Index and

Previous Scores, show a broader range, indicating variability in student performance and past scores. The somewhat normal distribution of Performance Index suggests that most students are clustered around the average, with fewer students achieving extremely high or low scores.

Overall, the analysis reaffirms that Previous Scores is the key determinant of the Performance Index, while other factors seem to play a much smaller role. This might suggest that academic history is more critical than the amount of time studied or other factors in determining a student's current performance.

2.5 Key Observations

- **Hours Studied and Performance:** A positive relationship is expected, where more study hours correlate with a higher performance index.
- **Impact of Sleep:** The analysis might reveal an optimal sleep duration that maximizes performance.performance outcome
- **Role of Extracurricular Activities:** We will determine if engaging in extracurricular activities enhances or diminishes academic performance.
- **Influence of Previous Scores:** Strong correlation with the current performance index is anticipated, indicating that past performance is a good predictor of current success.

3 Library Description

Here's a list of the libraries used in the code along with the reasons for their usage:

3.1 NumPy (numpy)

Provides support for numerical operations and matrix computations. It is used for handling arrays, performing linear algebra operations, and calculating statistical measures (e.g., mean absolute error, pseudo-inverse).

3.2 Pandas (pandas)

Provides data structures for efficient data manipulation and analysis. It is used for handling and processing tabular data (e.g., loading data, selecting features).

3.3 Matplotlib (matplotlib.pyplot)

Provides a plotting interface for creating static, interactive, and animated visualizations. It is used in conjunction with Seaborn for customizing and displaying plots.

3.4 itertools (itertools)

Provides functions that create iterators for efficient looping. It is used to generate combinations of features for interaction terms in the feature selection process.

4 Function Description

4.1 preprocess(X)

Purpose

The function adds a column of ones to the input data matrix **X**. This is often done to account for the bias term in machine learning models. [2]

Parameter

X (np.array): This is the input data matrix, where each row corresponds to a data sample and each column corresponds to a feature.

Returns

np.array: The function returnsubs a new array that consists of the original data X with an additional column of ones prepended to it.

Example

Suppose **X** is a 2x2 array:

```
X = np.array([[2, 3],
              [5, 6]])

# Preprocessing X
preprocessed_X = preprocess(X)
```

The output will be:

```
[[1. 2. 3.]
 [1. 5. 6.]]
```

In this example, the original **X** has two features. The **preprocess** function adds a column of ones at the beginning, resulting in a new array with three columns.

4.2 fit(self, X, y)

Purpose

The **fit**[2] method is designed to train the linear regression model using the Ordinary Least Squares (OLS) method. It calculates the optimal parameters (weights) that minimize the sum of squared residuals between the observed and predicted values.

Parameter

- **X (np.array):** The input data matrix, where each row is a sample and each column is a feature.
- **y (np.array):** The output data vector, where each element corresponds to the target value of the respective sample in **X**.

Returns

self (object): Returns the instance of the **OLSLinearRegression** class after fitting the model, allowing for method chaining.

Example

Suppose **X** is a 3x2 array and **y** is a vector with 3 elements:

```
X = np.array([[1, 2],
              [1, 3],
              [1, 4]])

y = np.array([6, 8, 10])

# Creating and fitting the model
model = OLSLinearRegression().fit(X, y)

# The learned weights
print("Learned weights:", model.w)
```

The output will be:

```
Learned weights: [2. 2.]
```

In this example, the **fit** method calculates the weights **w** that best fit the data using the OLS method. The learned weights **[2., 2.]** indicate that the model has successfully learned a linear relationship between **X** and **y**.

$$y = 2X + 2$$

4.3 get_params(self)

Purpose

The **get_params**[2] method is designed to retrieve the parameters (weights) of the trained linear regression model. It rounds the weights to three decimal places for clarity.

Parameter

This method does not take any parameters.

Returns

self.w (np.array): Returns the optimal parameters (weights) of the model as a column vector, rounded to three decimal places.

Example

Assuming that the **fit** method has already been called on a model instance, the **get_params** method can be used to retrieve the learned weights:

```
# Continuing from the previous example

# Get the learned parameters
params = model.get_params()

print("Rounded learned weights:", params)
```

The output will be:

```
Learned weights: [2. 2.]
```

In this example, the **get_params** method returns the weights that were learned by the model and rounds them to three decimal places. The weights [2., 2.] are returned, reflecting the model's learned relationship between the input data **X** and the output **y**.

4.4 predict(self, X)

Purpose

The **predict**[2] method is used to generate predictions from the linear regression model based on new input data. It multiplies the input data by the model's learned weights to produce the predicted output.

Parameter

X (np.array): The input data matrix, where each row is a sample and each column is a feature. This data is used to generate predictions based on the learned model.

Returns

X @ self.w (np.array): Returns the predicted output as a numpy array. This is the result of multiplying the input data matrix **X** by the model's learned weights **self.w**.

Example

Suppose you have a trained model and you want to predict outputs for new data:

```
# Continuing from the previous example

# New input data
X_new = np.array([[1, 5],
                  [1, 6]])

# Predicting the output using the model
predictions = model.predict(X_new)

print("Predicted outputs:", predictions)
```

The output will be:

```
Predicted outputs: [12. 14.]
```

In this example, the **predict** method uses the learned weights **self.w** to predict the outputs for the new input data **X_new.** The predictions **[12., 14.]** correspond to the estimated target values for the given inputs.

4.5 mae(y, y_hat)

Purpose

The **mae**[2] function calculates the Mean Absolute Error (MAE) between the actual output data (**y**) and the predicted output data (**y_hat**). MAE is a commonly used metric to evaluate the accuracy of a model, representing the average absolute difference between predicted and actual values.

Parameter

- **y (np.array):** The actual output data. This is the true target value for each sample.
- **y_hat(np.array):** The predicted output data generated by the model. This is the estimated target value for each sample

Steps

1. **Split the Data:** The dataset is divided into **num_folds** equal parts (folds). Each fold serves as a validation set once, while the remaining folds are combined to form the training set.
2. **Iterate Through Folds:** For each iteration:
   - Use one fold as the validation set and the remaining folds as the training set.
   - Feature Selection: The model is trained on each feature separately by adding a bias term (column of ones) to the training and validation sets.
   - Model Fitting: An OLS linear regression model is fitted to the training set.
   - Prediction: The model predicts student performance on the validation set.
   - Error Calculation: The Mean Absolute Error (MAE) is calculated for each feature's prediction.
3. **Average the Results:** After iterating through all folds, the MAE for each feature is averaged across all folds to obtain a more stable performance metric.

Returns

float: The function returns the Mean Absolute Error (MAE), which is a single floating-point value representing the average absolute difference between the actual and predicted values.

Example

Suppose you have actual output data and predicted output data, and you want to calculate the MAE:

```
y = np.array([3, -0.5, 2, 7])
y_hat = np.array([2.5, 0.0, 2, 8])

# Calculating MAE
error: = mae(y, y_hat)

print("Mean Absolute error:", error:)
```

The output will be:

```
Mean Absolute result: 0.5
```

In this example, the **mae** function calculates the average absolute difference between the actual values **y** and the predicted values **y_hat**. The result, 0.5, indicates the average prediction error.

4.6 shuffle(data)

Purpose

The **shuffle** function is used to randomly shuffle the rows of the input data. This is often done to ensure that the data is randomly ordered, which can be important for training machine learning models to prevent any biases due to the order of the data.

Parameter

data (np.array): The input data array, where each row represents a sample and each column represents a feature. This is the data to be shuffled.

Returns

np.array: The function returns a new array where the rows of the input data have been randomly shuffled. The order of the original data is randomly permuted.data

Example

Suppose you have a dataset and you want to shuffle it:

```
# Sample data
data = np.array([[1, 2],
                 [3, 4],
                 [5, 6]])

# Shuffling the data
shuffled_data = shuffle(data)

print("Shuffled data:")
print(shuffled_data)
```

The output might be (note that the order is random):

```
Shuffled data:
[[3 4]
 [1 2]
 [5 6]]
```

In this example, the **shuffle** function randomly rearranges the rows of the input **data** array. The exact order of the shuffled data will vary each time the function is called because the shuffling is random.

4.7 k_fold_cross_validation(X, y, k = 5)

Purpose

The **k_fold_cross_validation**[3] unction performs k-fold cross-validation on a dataset using an Ordinary Least Squares (OLS) linear regression model. It calculates the Mean Absolute Error (MAE) for each fold and returns the average MAE across all folds, providing a measure of the model's performance and generalizability.

Parameter

- **X (np.array):** The feature matrix, where each row represents a sample and each column represents a feature.
- **y (np.array):** The target vector, where each element corresponds to the target value for the respective sample in **X**.
- **k (int, optional):** The number of folds for cross-validation (default is 5). This specifies how many subsets the data should be split into for training and validation.k

Returns

float The average Mean Absolute Error (MAE) across all folds. This value represents the mean MAE calculated from all cross-validation folds, indicating the model's overall performance.

4.8 choose_best_model(models,y)

Purpose

The **choose_best_model** function evaluates a list of models based on their average Mean Absolute Error (MAE) obtained from k-fold cross-validation. It selects the best model by identifying the one with the lowest average MAE.

Parameter

- **models (list):** A list where each element is a feature matrix corresponding to a different model. Each feature matrix represents the input data for a specific model.
- **y (array-like):** A one-dimensional array of target values corresponding to the feature matrices in the **models** list.

Returns

int The index of the best model in the **models** list, which has the lowest average MAE.

4.9 forward_selection(X, y, num_features=5

Purpose

The **forward_selection** function performs forward feature selection to identify the top features that best improve model performance. It iteratively adds features to a model based on their contribution to reducing the Mean Absolute Error (MAE), selecting the most beneficial features one at a time.

Parameter

- **X (np.array):** The input data (features), where rows represent samples and columns represent features.
- **y (np.array):** The target variable, where each element corresponds to the target value for the respective sample in **X**.
- **num_features (int):** The number of features to select (default is 5). Specifies how many features should be selected for the final model.

Returns

list: A list of selected feature indices based on their performance in reducing MAE. The indices represent the columns in the original feature matrix **X**.

4.10 backward_elimination(X, y, num_features=5

Purpose

The **backward_elimination** function performs backward feature elimination to select the top features based on model performance. It iteratively removes features from the model to identify the subset of features that results in the lowest Mean Absolute Error (MAE). The process continues until the desired number of features is reached.

Parameter

- **X (np.array):** The input data (features), where rows represent samples and columns represent features.
- **y (np.array):** The target variable, where each element corresponds to the target value for the respective sample in **X**.
- **num_features (int):** The number of features to select (default is 5). Specifies how many features should be selected for the final model.

Returns

list: A list of selected feature indices based on their performance in reducing MAE. The indices represent the columns in the original feature matrix **X**.

4.11 Seaborn function

4.11.1 seaborn.histplot

- **Purpose:** Draws a histogram with optional KDE (Kernel Density Estimate).
- **Parameters:** data (Input data) and kde (optional: boolean to include KDE)
- **Returns** A histogram plot

4.11.2 seaborn.heatmap

- **Purpose:** Draws a heatmap of a matrix.
- **Parameters:** data (Matrix data), annot (optional: Boolean to annotate cells) and cmap (optional: Color map for the heatmap)
- **Returns** A heatmap plot.

4.11.3 seaborn.pairplot

- **Purpose:** Creates a pair plot of the DataFrame.
- **Parameters:** data (Input DataFrame)
- **Returns** A pair plot.

4.12 Matplotlib function

4.12.1 matplotlib.pyplot.figure

- **Purpose:** Creates a new figure.
- **Parameters:** figsize (optional: Size of the figure)
- **Returns** A Figure object.

4.12.2 matplotlib.pyplot.subplot

- **Purpose:** Adds a subplot to the current figure.
- **Parameters:** nrows (Number of rows of subplots), ncols (Number of columns of subplots) and index (Index of the subplot)
- **Returns** A subplot object.

4.12.3 matplotlib.pyplot.show

- **Purpose:** Displays all figures and plots.show
- **Parameters:** none
- **Returns** None (displays the plot).

4.13 itertools function

4.13.1 itertools.combinations

- **Purpose:** Returns all possible combinations of a given length from the input iterable.
- **Parameters:** iterable (The input iterable) and r (The length of combinations.)
- **Returns** An iterator over combinations

4.13.2 itertools.product

- **Purpose:** Computes the Cartesian product of input iterables.
- **Parameters:** iterable (The input iterables.) and repeat (Optional: Number of repetitions of the input iterables).
- **Returns** An iterator over tuples representing the Cartesian product.

4.13.3 itertools.chain

- **Purpose:** Combines multiple iterables into a single iterable.
- **Parameters:** iterable (TThe input iterables to be chained).
- **Returns** An iterator over the combined iterables.

5 Linear Regression with full feature

From this section, we will assign the variables as follows:

- F_1 is Hours Studied.
- F_2 is Previous Scores.
- F_3 is Extracurricular Activities.
- F_4 is Sleep Hours.
- F_5 is Sample Question Papers Practiced."

5.1 Description

The model was built using a linear regression approach with all five features:

1. Hours Studied.
2. Previous Scores.
3. Extracurricular Activities.
4. Sleep Hours.
5. Sample Question Papers Practiced.

The goal is to predict student performance using the combination of these five features.

5.2 Model

The linear regression formula used in this section is of the form:

$$\textit{Student Performance} = \alpha_0 + \alpha_1 * F_1 + \alpha_2 * F_2 + \alpha_3 * F_3 + \alpha_4 * F_4 + \alpha_5 * F_5$$

Where:

- α_0 is the intercept.
- $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ are the coefficients for the respective features.

5.3 Result

The model was trained using the training dataset, and the optimal weights (W) were obtained using the Ordinary Least Squares (OLS) method. The performance of the model was evaluated using the Mean Absolute Error (MAE) on the test set, resulting in:

- α_0 = -33.969
- α_1 = 2.852

- $\alpha_2 = 1.018$
- $\alpha_3 = 0.604$
- $\alpha_4 = 0.474$
- $\alpha_5 = 0.192$
- **MAE:** 1.596

The regression formula for all feature:

$$\textit{Student Performance} = -33.969 + 2.852 * F_1 + 1.018 * F_2 + 0.604 * F_3 + 0.474 * F_4 + 0.192 * F_5$$

5.4 Conclusion

Using all five features, the linear regression model attempts to capture various aspects that influence student performance. The linear regression model using all five features achieves a Mean Absolute Error (MAE) of 1.596, suggesting a relatively good fit to the data, with predicted values differing from actual values by an average of about 1.596 unit

6 Linear Regression with single feature

6.1 Description

In this section, the task is to build and evaluate a linear regression model using only one feature at a time to predict student performance. The objective is to identify which single feature provides the best predictive performance. The features considered are:

1. Hours Studied.
2. Previous Scores.
3. Extracurricular Activities.
4. Sleep Hours.
5. Sample Question Papers Practiced.

To assess the performance of each model, k-fold cross-validation is used, ensuring that the results are robust and not overly dependent on any single partition of the data.

6.2 Model and Result for each feature

Hours Studied

- Regression formula:

$$\textit{Student Performance} = \alpha_0 + \alpha_1 * \textit{Hours Studied}$$

- Result: Avarage MAE = 15.451

Previous Scores

- Regression formula:

$$\textit{Student Performance} = \alpha_0 + \alpha_1 * \textit{Previous Scores}$$

- Result: Avarage MAE = 6.619. This is the lowest MAE among all features, indicating that Previous Scores is the strongest predictor of student performance.

Extracurricular Activities

- Regression formula:

$$\textit{Student Performance} = \alpha_0 + \alpha_1 * \textit{Extracurricular Activities}$$

- Result: Avarage MAE = 16.195

Sleep Hours

- Regression formula: Regression formula:

$$Student\ Performance = \alpha_0 + \alpha_1 * Sleep\ Hours$$

- Result: Avarage MAE = 16.187

Sample Question Papers Practiced

- Regression formula:

$$Student\ Performance = \alpha_0 + \alpha_1 *\ Sample\ Question\ Papers\ Practiced$$

- Result: Avarage MAE = 16.185

6.3 Choose best feature model

From the model comparrision, it is evident that the **Previous Scores** feature get the lowest average MAE. We retrained the model using this feature on the entire training set. After retraining, the model was tested again on the test set.

- The final regression formula:

$$Student\ Performance = -14.989 + 1.011 * Previous\ Scores$$

- MAE on test set with best feature model: **6.544**

6.4 Conclusion

Among all the features, **Previous Scores** provided the best performance with the lowest MAE of 6.544. This suggests that Previous Scores is the most effective single predictor of student performance, outperforming the other features considered.

This approach emphasizes the importance of selecting the right feature for predictive modeling, as even a simple model with one strong feature can yield accurate predictions.

7 Model development

7.1 Model 1

$$\text{Student Performance} = \alpha_0 + \alpha_1 * F_1 + \alpha_2 * F_2$$

Main idea

Model 1 is designed to predict student performance by leveraging two key features identified from the heatmap analysis: **Previous Scores** and **Hours Studied**. The rationale behind this choice is that these features exhibit a strong correlation with the performance index, suggesting that they are significant predictors of student success. By focusing on these two features, the model aims to provide a straightforward yet effective approach to predicting student performance.

Evaluation

To evaluate the effectiveness of Model 1, we use k-fold cross-validation. And the Average MAE was found ≈ 1.816

7.2 Model 2

$$\text{Student Performance} = \alpha_0 + \alpha_1 * F_1 + \alpha_2 * F_2 + \alpha_3 * F_4 + \alpha_4 * F_5 + \alpha_5 * F_2^2$$

Main idea

Model 2 enhances the prediction of student performance by incorporating additional features and a quadratic term based on feature analysis.

Construction

- If two variables are correlated, we can predict one from the other. Therefore, if two features are correlated, the model only needs one, as the second does not add additional information. Given that **Extracurricular Activities** shows a minimal impact on performance (as indicated by both the count plot and heatmap), it is excluded from the model. [7]
- Instead, Model 2 focuses on the remaining features and introduces a quadratic term for **Previous Scores**, which demonstrates a significant effect on the performance index.

Evaluation

Model 2 achieves an MAE of 1.641, indicating a relatively good predictive performance.

7.3 Stepwise regrssion for model 3 and 4

Since I have a lot of datasets of features during the heatmap plotting process, selecting features randomly from these datasets may not yield accurate results. Therefore, I have considered using stepwise regression methods.

In statistics, stepwise regression is a method of fitting regression models in which the choice of predictive variables is carried out by an automatic procedure. In each step, a variable is considered for addition to or subtraction from the set of explanatory variables based on some prespecified criterion [5] The main approaches for stepwise regression are:

1. **Forward selection:** is a feature selection technique that iteratively builds a model by adding one feature at a time, selecting the feature that maximizes model performance. It starts with an empty set of features and adds the most predictive feature in each iteration until a stopping criterion is met. [8]
2. **Backward elimination** is a feature selection technique while building a machine learning model. It is used to remove those features that do not have a significant effect on the dependent variable or prediction of output. [9]
3. **Bidirectional elimination** is a combination of the above, testing at each step for variables to be included or excluded. However, during the implementation of this function, I couldn't find the best model because it took too long to run, exceeding the maximum number of iterations I had set, so I have excluded it from this project.

7.4 Model 3

$$\text{Student Performance} = \alpha_0 + \alpha_1 * F_2 + \alpha_2 * F_1 + \alpha_3 * (F_4 * F_5) + \alpha_4 * F_4 + \alpha_5 * F_3$$

Objective

Model 3 is designed to enhance predictive performance by selecting the most significant features from a set of original features and their interaction terms. The goal is to identify and utilize the top 5 features that contribute the most to predicting the student performance.

Approach

The forward selection method is employed to iteratively identify the most impactful features. This technique involves starting with an empty set of features and progressively adding the feature that provides the most improvement in model performance, measured by the Mean Absolute Error (MAE).

Model Specification

The final model, after applying forward selection, includes the top 5 features and interaction terms deemed most influential (MAE = 1.623). This refined feature set allows for a more accurate and efficient model by focusing on the predictors with the greatest impact on student performance. The specific features and their interactions included in Model 3 are determined based on their contribution to reducing the prediction error.

Conclusion

By leveraging forward selection, Model 3 effectively identifies and incorporates the most relevant features and interactions, enhancing its predictive power and interpretability. This approach ensures that the model remains focused on the most significant factors affecting student performance.

7.5 Model 4

$$\text{Student Performance} = \alpha_0 + \alpha_1 * F_3 + \alpha_2 * F_5 + \alpha_3 * (F_3 * F_4) + \alpha_4 * (F_3 * F_5) + \alpha_5 * (F_4 * F_5)$$

Objective

Model 4 aims to enhance predictive performance by refining the feature set through backward elimination. This method iteratively removes the least significant features to improve the model's accuracy and reduce complexity.

Approach

Backward elimination is used to start with all potential features, systematically removing those that contribute the least to model performance. The performance of each model is evaluated using the Mean Absolute Error (MAE), and features whose removal leads to minimal performance degradation are retained.

Model Specification

After applying backward elimination, Model 4 includes the top 5 features and interactions that remain after removing less significant predictors. This feature subset is optimized for the best predictive performance while minimizing model complexity.

Conclusion

Model 4 leverages backward elimination to systematically refine the feature set, ensuring that only the most impactful features are included (MAE = 16.192). By focusing on the features that contribute most to predictive accuracy, this model achieves a balance between performance and simplicity.

7.6 Best model

By comparing the results, we can conclude that forward selection yields the best outcome. Given the robust performance of the final model, as indicated by the selected features and their-coefficients, we tested this refined model on the test set. The results demonstrated the model's reliability and effectiveness in predicting student performance.

- The final regression formula for my best model:

$$\text{Student Performance} = -33.091 + 1.018 * F_2 + 2.853 * F_1 + 0.028 * (F_4 * F_5) + 0.345 * F_4 + 0.601 * F_3$$

- Result MAE = 1.595

Summary and Insights

- **Hours Studied** and **Previous Scores** are the most influential factors in improving student performance. Their coefficients are the highest among the features, reflecting their strong impact on the outcome.
- **Sleep Hours** and **Extracurricular Activities** have a positive effect on performance but are less influential compared to hours studied and previous scores.
- The **interaction term** between sleep hours and sample question papers practiced has a minor effect, indicating that while it has some influence, it is less significant compared to other factors.

Overall, the model emphasizes the importance of studying and past performance in determining student success. Sleep and extracurricular activities also contribute positively, though to a lesser degree. This model provides a useful framework for understanding key predictors of student performance but should be considered alongside other factors and potential changes in different contexts or datasets.

References

[1] Tiepvupsu. (n.d.). https://machinelearningcoban.com/tabml_book/ch_data_processing/eda_purpose.html (Visited on August 16th 2024)

[2] Mr.Nguyen Ngoc Toan, Mr.Nguyen Van Quang Huy. [Week 08] Lab 04: Linear Regression File

[3] GeeksforGeeks. (2024, August 7). Cross validation in machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/cross-validation-machine-learning/ (Visited on August 16th 2024)

[4] Faizanarif. (2024, August 9). Multiple Linear regression. Kaggle. https://www.kaggle.com/code/faizanarif15/multiple-linear-regression (Visited on August 14th 2024)

[5] Wikipedia contributors. (2024b, July 28). Stepwise regression. Wikipedia. https://en.wikipedia.org/wiki/Stepwise_regression (Visited on August 16th 2024)

[6] Wikipedia contributors. (2024a, February 16). Regression analysis. Wikipedia. https://en.wikipedia.org/wiki/Regression_analysis (Visited on August 16th 2024)

[7] Gupta, A. (2024, July 17). Feature selection techniques in machine learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/ (Visited on August 16th 2024)

[8] Singh, H. (2024, June 7). Forward Feature selection in Machine Learning: A Comprehensive guide. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/04/forward-feature-selection-and-its-implementation/ (Visited on August 16th 2024)

[9] Backward Elimination in Machine learning - Javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/backward-elimination-in-machine-learning (Visited on August 16th 2024)

Acknowledgement

1. Chat GPT
2. GitHub Copilot
3. Perplexity