PT-M1: Comparing Logistic Regression with Decision Tree for Multinomial Classification

Kirvin Josh C. Castro

IT Elective III, BSIT

Jose Rizal University

Mandaluyong, Philippines

kirvinjosh.castro@my.jru.edu

***Abstract*— This study explores the application of Machine Learning techniques for mushroom classification, specifically comparing Logistic Regression and Decision Tree models. Using a dataset containing mushroom characteristics, the models were trained to predict whether a mushroom is edible or poisonous. The dataset underwent preprocessing, including handling missing values and encoding categorical variables. Performance metrics such as accuracy, precision, recall, and F1-score were evaluated. The results indicated that the Decision Tree model outperformed Logistic Regression in classification accuracy. This study highlights the importance of feature selection and model interpretability in mushroom classification.**

***Keywords— Mushroom Classification, Logistic Regression, Decision Tree, Machine Learning, Feature Engineering, Predictive Modeling.***

1. Introduction

Mushroom classification has long been a fundamental challenge in the field of mycology. Accurate identification of mushrooms is critical, especially for distinguishing between edible and poisonous species. With thousands of mushroom species existing worldwide, misidentifications can lead to serious health risks, including poisoning and even death. Traditionally, identifying mushrooms has relied heavily on physical characteristics such as cap shape, gill structure, color, and habitat, but this method is labor-intensive and requires significant expertise. Moreover, the sheer diversity of mushrooms and the subtle differences between species make manual identification prone to errors, especially by non-experts.

In recent years, Machine Learning (ML) has emerged as a powerful tool for automating classification tasks, providing more consistent and accurate results. By leveraging computational algorithms to analyze large datasets, ML models can detect patterns and relationships in the data that might be overlooked by human experts. This study applies two widely-used ML models—**Logistic Regression** and **Decision Tree**—to classify mushrooms as either edible or poisonous based on their characteristics.

The dataset used in this study, **"Mushroom Classification"** from Kaggle, contains various attributes that describe the physical and chemical properties of mushrooms. These attributes include characteristics such as cap shape, color, odor, gill size, and habitat. The goal of this research is to explore and compare the performance of Logistic Regression and Decision Tree models in classifying mushrooms based on these features.

Data Collection

1. *Dataset Description*

The study relies on a thorough educational dataset that includes various performance-influencing academic, personal and environmental metrics. The educational data repository served as the reliable source for data acquisition which provided high-value information needed for predictive modeling purposes. The dataset contains 6,054 records which use 20 attributes to measure the factors that affect the performance outcome variable Exam\_Score.

The dataset contains three diverse continuous metrics: Hours\_Studied, Attendance, and Previous\_Scores that demonstrate meaningful potential as predictor variables. The researcher encoded the categorical variables Parental\_Involvement, Access\_to\_Resources, and Motivation\_Level using ordinal and binary schemes so they would maintain their natural relationships. The dataset maintains complete data integrity since all entries follow a reliable structure that avoids missing or duplicate information.

df.info() was also executed, in order to display the number of records, column names, its data types, memory usage, and non-null counts. The command helps the researchers to identify missing values and analyze the dataset structure. Df.dtypes() was also executed to display the data types of each attributes in the Data Frame. This further helps in data preprocessing and analysis as it determines whether the attribute contains numerical or categorical data.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 3. df.info() command execution

1. *Dataset Attributes*

The **Mushroom Classification** dataset from Kaggle contains a total of **8,124 records** and **22 attributes**, describing the characteristics of different mushroom species. The dataset is entirely categorical and includes features such as cap shape, color, gill size, and habitat. The target variable is binary, indicating whether a mushroom is edible (represented as **'e'**) or poisonous (represented as **'p'**).

**Dataset Attributes:**

1. **class**: The target variable (edible or poisonous).
2. **cap\_shape**: Shape of the mushroom cap (bell, conical, flat, knobbed, sunken).
3. **cap\_surface**: Surface of the cap (fibrous, grooves, scaly, smooth).
4. **cap\_color**: Color of the cap (brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow).
5. **bruises**: Whether the mushroom bruises or not (bruises or no).
6. **odor**: Odor of the mushroom (almond, anise, creosote, fishy, foul, musty, none, pungent, spicy).
7. **gill\_attachment**: Attachment of gills (attached, descending).
8. **gill\_spacing**: Spacing between gills (close, crowded).
9. **gill\_size**: Size of gills (broad, narrow).
10. **gill\_color**: Color of gills (black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow).
11. **stalk\_shape**: Shape of the stalk (enlarging, tapering).
12. **stalk\_root**: Type of stalk root (bulbous, club, cup, equal, rhizomorphs, rooted, missing).
13. **stalk\_surface\_above\_ring**: Surface of the stalk above the ring (fibrous, scaly, silky, smooth).
14. **stalk\_surface\_below\_ring**: Surface of the stalk below the ring (fibrous, scaly, silky, smooth).
15. **stalk\_color\_above\_ring**: Color of the stalk above the ring (brown, buff, cinnamon, gray, green, orange, pink, red, white, yellow).
16. **stalk\_color\_below\_ring**: Color of the stalk below the ring (brown, buff, cinnamon, gray, green, orange, pink, red, white, yellow).
17. **veil\_type**: Type of veil (partial, universal).
18. **veil\_color**: Color of the veil (brown, orange, white, yellow).
19. **ring\_number**: Number of rings on the stalk (none, one, two).
20. **ring\_type**: Type of ring on the stalk (evanescent, large, none, pendant).
21. **spore\_print\_color**: Color of spore print (black, brown, buff, chocolate, green, orange, purple, white, yellow).
22. **population**: Population of mushrooms in the habitat (abundant, clustered, numerous, scattered, solitary).
23. **habitat**: Habitat where the mushroom was found (grasses, leaves, meadows, paths, urban, waste).

*Data Cleaning and Data Preprocessing*

The dataset has no missing values, so no imputation is needed. We perform basic checks to confirm this by using the df.isnull().sum() function, which checks for null values in each column.

* 1. *Data preprocessing for further data cleaning*

The output of df.head() displays the first five rows of a dataset containing various factors influencing student performance. Key attributes include Hours\_Studied, Attendance, Parental\_Involvement, and Previous\_Scores, which reflect study habits, school engagement, and past academic performance. Other factors like Access\_to\_Resources, Internet\_Access, and Family\_Income indicate external support and socio-economic conditions. Additionally, Motivation\_Level, Sleep\_Hours, and Tutoring\_Sessions highlight personal and academic behaviors that may impact success. The dataset also includes Teacher\_Quality and School\_Type, which provide insights into educational environments. These variables help analyze student performance through multiple regression analysis.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 4. df.head() command execution

The histograms show numerical data distribution patterns alongside the assessment of both skewness and detection of outliers in the dataset. Most students fall within the normal range when it comes to hours spent studying based on the normal distribution pattern of the Hours\_Studied attribute. The distribution of attendance data is even across all options which indicates different students have different levels of regularity in their presence. Each categorical attribute produces significant spikes in the data which corresponds with the specific available values. Most students maintain a regular sleep pattern by distributing hours they sleep around the 6-to-8-hour mark. The academic performance levels show wide distribution among students in Previous\_Scores. The data attributes Internet\_Access, Family\_Income, and Teacher\_Quality show signs of being categorical since their values are limited and binary or ordinal in nature. Data trends become easier to understand through this visualization method which also helps with preprocessing strategy selection.

A screenshot of a computer screen

AI-generated content may be incorrect.

Fig. 5. histogram command execution

b.) Identifying the missing values

To better understand the dataset, we performed an examination of the number of unique values and missing values in each column. This is crucial in deciding the type of feature it is, whether it is a categorical feature, binary feature, or a continuous feature. It also assists us in making sure that our dataset is complete and free from missing values, thus reducing the imputation process. Columns with many unique values, like Attendance and Previous\_Scores, have a large number of possible values, while columns like Parental\_Involvement and Internet\_Access are mostly categorical. From this examination, we are better placed to determine the proper ways of processing and modeling the data.

A computer code on a black background

AI-generated content may be incorrect.

Fig. 6. Displaying the unique values and the missing values

The below code snippet performs an analysis of a dataset to identify columns that have only unique values for every single row. The script iterates through all columns of the DataFrame in a systematic manner and checks if the number of unique values in a column matches the total number of rows available. If a column meets this criterion, it is then added to an aggregated list. However, as the results show, there are no columns in this dataset that have entirely unique values in every row. This finding suggests that the dataset does not have a clear identifier column, like an ID or unique key.

A screenshot of a computer code

AI-generated content may be incorrect.

Fig. 7. Identifying unique columns where row has a unique value

The given line of code is meant to check for the uniqueness of each column in the dataset. Through the looping of each column and printing their corresponding values, the operation allows for the identification of categorical variables, detection of data anomalies, and checking of the numerical range. This kind of functionality is very critical in exploratory data analysis (EDA), since the knowledge of individual values in each feature is critical in data cleaning, feature generation, and

model preparation.A computer code on a black background

AI-generated content may be incorrect.

Fig. 8. Uniqueness of each column in the dataset.

The code below enables the transformation of categorical data into a numeric format through the assignment of unique integers to each category. The code systematically detects categorical columns, assigns their unique values to integers through mapping, and applies these mappings to the dataset. This transformation facilitates the use of qualitative data in machine learning models.

A screen shot of a computer code

AI-generated content may be incorrect.

Fig 10. Binary Mappings

This script facilitates the encoding of categorical variables into numeric form, hence making them compatible with machine learning models. It goes through all the categorical columns methodically, assigning each unique value an integer and performing the transformation on the whole dataset. For ordinal variables, which have a specific order (e.g., "Low" to "High"), the encoding is from 0 to n, whereas binary variables such as "Yes/No" are assigned values 0 and 1. Nominal variables, which lack a natural order, are assigned random integer values. This is crucial in transforming qualitative data into an appropriate format for efficient algorithmic processing.

The given output specifies the binary encoding parameters for the ordinal variables in the dataset. For ordinal variables, which have a certain order, the encoding is given from 0 to 2. This format is used for variables such as Parental\_Involvement, Access\_to\_Resources, Motivation\_Level, Family\_Income, Teacher\_Quality, Parental\_Education\_Level, and Distance\_from\_Home, where values with labels "Low" are encoded as 0 and those with labels "High" are encoded as 2. Binary encoding is used for dichotomous or binary-category variables, which are Extracurricular\_Activities, Internet\_Access, Learning\_Disabilities, and Gender, where one category is represented as 0 and the other as 1. For nominal variables that do not have any order, such as School\_Type and Peer\_Influence, the classes are represented as 0, 1, and 2.

c. Handling Missing Values

The code effectively identifies and addresses missing values in a DataFrame. It starts by counting the missing values in every column through df.isnull().sum(), saving the output in missing\_values, and then prints the number of missing values in every column. The next is the elimination of columns that have only missing values through df.dropna(axis=1, how='all', inplace=True), thereby ensuring the elimination of fully empty columns. It then removes any rows that contain missing values through df.dropna(inplace=True), thereby safeguarding the integrity of the analysis from missing data. Finally, it prints the modified dimensions of the DataFrame, thereby ensuring the verification of the changes made.

A computer screen shot of text

AI-generated content may be incorrect.

Fig. 12. Handling the missing values

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 13. Output of handling the missing values

The output confirms the lack of any missing values across the columns in the dataset. This fact attests that the dataset is complete and ready to be analyzed, eliminating the requirement for imputation or records elimination. After fixing missing values, the shape of the DataFrame is (6054, 20), indicating there are 6,054 observations and 20 variables. Such a requirement ensures the reliability of the statistical outputs and the training processes of the models.

d.) Identifying Duplicate Values

The code below identifies and removes duplicate rows in a DataFrame. It first calculates the number of duplicate rows using df.duplicated().sum(), which checks for duplicate entries and sums them up. The result is stored in duplicate\_count and printed to inform the user about the number of duplicate rows present. Then, the code removes these duplicate rows using df.drop\_duplicates(inplace=True), modifying the DataFrame directly to ensure that only unique rows remain. This helps in cleaning the dataset by eliminating redundant data, improving the efficiency and accuracy of further analysis.

A screen shot of a computer code

AI-generated content may be incorrect.

Fig. 14. Counting the number of duplicate rows

The result shows that there are no duplicate rows in the dataset. This indicates that all the observations are unique, ensuring the integrity of the data and reducing the risk of bias in the analysis.

A grey background with white text

AI-generated content may be incorrect.

Fig. 15. The number of duplicate rows

e.) Outlier Detection

Here is a code snippet that detects and removes outliers from a DataFrame through the Interquartile Range (IQR) method. The function detect\_outliers\_iqr initially loops through all numerical columns, calculates the first (Q1) and third quartiles (Q3), and calculates the interquartile range (IQR). It then calculates lower and upper bounds by multiplying the IQR by a factor of 1.5 and then identifies values outside this range as outliers. These outliers are counted in a dictionary. For ease of visualization, the code plots boxplots for all columns that have outliers, thus making it easy to visualize the distribution and extreme values involved. The function remove\_outliers\_iqr then goes ahead to remove these outliers, leaving only those values that fall within the calculated IQR range, thus generating a cleaned DataFrame that is free from extreme values. This process helps in improving data integrity by removing extreme values from affecting future analysis and machine learning models.

A screenshot of a computer program

AI-generated content may be incorrect.

Fig. 16. Detecting the outliers

A screenshot of a graph

AI-generated content may be incorrect.

Fig. 17. Boxplot of values (with outliers)

The boxplots of the variables Internet\_Access and Learning\_Disabilities show the presence of extreme outliers. Both variables have a steep concentration of points around zero with some extreme values on the upper end of the distribution. This shows that most of the data points are bunched in one category (presumably indicating minimal or non-access/the absence of the condition) with a few observations in a deviant category. The steep difference between the central cluster and the outliers can be an indicator of data entry problems, rare occurrences, or extreme heterogeneity in these attributes. Additional research needs to be conducted to determine if the outliers are valid observations or spurious observations that need to be cleaned up by data preprocessing techniques, such as transformation, capping, or deletion, depending on the research setting.

III. ATTRIBUTE SELECTION

Feature selection is a central data preprocessing module in machine learning whose aim is to identify the most informative features and eliminate redundant and high-correlation features. We use this technique to improve models, reduce overfitting, and enhance interpretability. There are many ways to competent feature selection, including statistical analysis, domain knowledge, correlation tests, feature vectorization, and dimensionality reduction methods like Principal Component Analysis (PCA). By using these methods wisely, practitioners are able to develop predictive models that are accurate and efficient.

Correlation analysis was used in this study to select the most important attributes from data. Correlation analysis is used to measure the strength of relationships between numerical features and determine the degree to which they affect one another. Feature attributes that have high correlations are likely to lead to redundancy and need to be excluded to preserve the efficiency and objectivity of the model. By focusing on attributes characterized by high and significant correlations, the dataset is reduced to include only the most predictive variables, enhancing the precision of machine learning models while decreasing complexity.

Besides, the use of correlation analysis facilitates data-driven decision-making, ensuring that information obtained is reliable and actionable.

a.) Correlation Analysis Heatmap

A heatmap visualization was utilized in this study to analyze the relationships between attributes in the dataset, providing a correlation matrix for all numerical features. This visualization helps in understanding the degree of association between different variables.

In the heatmap:

* Dark red shades indicate a strong positive correlation (closer to 1), meaning an increase in one variable is associated with an increase in another.
* Dark blue shades represent a strong negative correlation (closer to -1), meaning an increase in one variable is associated with a decrease in another.
* Neutral shades (near 0) indicate weak or no correlation between variables.
* Diagonal values are always 1, as a variable is perfectly correlated with itself.

The visualization assists in the determination of the redundant variables that are responsible for multicollinearity, which negatively impacts predictive modeling. It also determines the features with the highest impact on the target variable, i.e., Exam\_Score. The heatmap reveals that Attendance (0.68) and Hours\_Studied (0.50) have the highest positive correlation with Exam\_Score, and therefore they are the most significant predictors of student performance. Gender (-0.50) is negatively correlated, indicating a gender-based disparity in examination performance.

Through correlation analysis, it is now possible to select the most informative features and reject the redundant or highly correlated features and thereby construct an efficient and effective predictive model.

A graph of heatmap

AI-generated content may be incorrect.

Fig. 18. Correlation Analysis Heatmap

b.) Multicollinearity Check

Multicollinearity is a common issue in regression analysis, where there is high correlation among independent variables. Multicollinearity makes it difficult to interpret model coefficients and produces predictions that are not reliable. Multicollinearity is checked with the Variance Inflation Factor (VIF), which gives a measure of the extent to which the variance of a regression coefficient is inflated by multicollinearity. High VIF value (typically above 10) signifies high collinearity, and the respective variable must be removed or transformed.

The code in question calculates VIF values for all the predictor variables in the data. First, the predictor variables from the cleaned data (df\_cleaned) are extracted. Second, a DataFrame is initialized to store the VIF values for each feature. The variance\_inflation\_factor function from statsmodels is used to calculate the VIF scores for each variable. Lastly, the VIF values are printed, thus making it simple to identify features that could potentially create multicollinearity problems and need to be removed or transformed in the model.

A screenshot of a computer program

AI-generated content may be incorrect.

Fig. 19. Multicollinearity Check

The Variance Inflation Factor (VIF) test reveals potential multicollinearity among the predictor variables. Specifically, the "Attendance," "Previous\_Scores," and "Sleep\_Hours" features have comparatively high VIF values (above 10), which suggests high multicollinearity that can have a negative impact on the stability of the model and coefficient interpretation. Other features have moderate VIF values, generally below 5, suggesting acceptable multicollinearity. NaN values for "Internet\_Access" and "Learning\_Disabilities" can be due to constant or highly correlated features.

A black screen with white text

AI-generated content may be incorrect.

Fig. 20. Output of the Multicollinearity Check

IV. MODEL BUILDING

a) Simple Linear Regression

Q-Q plot of residuals visually verifies whether residuals from the regression model are normally distributed. The red reference line indicates perfect normal distribution, and the blue points indicate actual residuals. If residuals are normally distributed, then points must be very close to the red line. In this scenario, most points are along the line, which indicates that the normality assumption is satisfied to a significant extent. Slight deviations at the tail indicate slight skewness or the presence of outliers. Generally, the residuals are in a pattern consistent with normality, which makes the reliability of the model's inferences and predictions justified.

Normality of residuals is important in the validation of linear regression assumptions because it impacts confidence interval accuracy and hypothesis testing accuracy. If residuals are normally distributed, then the model's prediction is more reliable, and statistical inferences regarding variables' relationships are more reliable. Non-normality, especially in the tails, can indicate potential problems such as heteroscedasticity, omitted variables, or the requirement for data transformation. Although slight deviations are noted in the Q-Q plot, overall compliance of the residuals with the reference line means that the model satisfies the normality assumption to a reasonable extent, which makes its capacity to detect underlying patterns in the data justified.

A graph with a red line

AI-generated content may be incorrect.

Fig. 21. Q-Q Plot of the Residuals

The Q-Q plot of the residuals reveals that the residuals trace a very normal distribution as most of the points fall along the red reference line. This arrangement reveals that the normality assumption of the residuals is generally satisfied, a crucial prerequisite for linear regression models. Minimal deviations are, however, observed at the tails, especially at the top, which reveals minor skewness or the existence of outliers. The residuals generally trace a pattern akin to normal distribution, hence testifying to the reliability of the model's inferences and predictions.

b.) Multiple Linear Regression

Multiple Linear Regression is a sophisticated version of simple linear regression, trying to explain the relationship between a target variable (dependent variable) and several predictor variables (independent variables). The technique is used to investigate the effect of several factors on a particular outcome, thus being a major tool in predictive modeling.

In this research, we will create a multiple linear regression model intended to predict Exam Scores based on a group of determinants such as study hours, parental influence, previous scores, and other determinants. The model will be trained on a data set, and its performance will be evaluated using R-squared measures for the training and test sets.

The following code is required to perform multiple linear regression, including steps such as data preprocessing, splitting data into training and test sets, model training, and evaluating its performance.

A computer screen shot of a program code

AI-generated content may be incorrect.

Fig. 22. Multiple Linear Regression Code

The multiple linear regression output shows the impact of each predictor on the target variable. The intercept is about 38.92, the value at baseline when all the predictors are zero. The coefficients show the direction and strength of the impact of each predictor. "Hours\_Studied" and "Motivation\_Level" have positive coefficients, since higher values produce better outcomes. "Attendance" and "Distance\_from\_Home" have negative coefficients, since there's a reverse relationship. The large values of R-squared for training (0.972) as well as for testing (0.972) show the model fits perfectly, with about 97% of the target variance accounted for by the predictors, showing the power of the model and generalizability.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 23. Results from using the Multiple Linear Regression

# V. DISCUSSION

1. *Model Performance Evaluation of both models*

**Simple Linear Regression Analysis**

The simple linear regression model was used to predict student exam scores from one predictor variable. The Q-Q plot of residuals was inspected to find out if such residuals are normally distributed, which is a basic assumption of regression analysis. The red reference line on the graph represents a perfect normal distribution, and the blue dots on the graph represent the actual residuals. If the residuals are normally distributed, the dots are supposed to closely follow the red line.

In this analysis, most of the dots follow the reference line, confirming that the residuals are close to normally distributed. Divergences at extremes show potential skewness or outliers. Despite such exceptions, the overall trend is as expected under normality, thus confirming the validity of the conclusions and predictions from the model. The high R² values of both the training and test sets also confirm that the chosen predictor has a very high correlation with exam scores, hence a valid single-variable model.

Normality in residuals is essential for the verification of model assumptions, which influence the reliability of confidence intervals and hypothesis testing. Abnormality can imply heteroscedasticity, omitted variables, or the need for transformation. However, since the residuals are generally in accordance with normality, this basic linear regression model provides a good foundation for the prediction of exam scores based on a single factor.

**Multiple Linear Regression Analysis**

Multiple linear regression expands simple regression by using multiple independent variables to forecast exam scores, providing a broader picture of the variables affecting student performance. Predictors here included hours studied, past scores, and other academic characteristics. By considering multiple factors, this model provides a better picture of how individual factors affect exam performance.

The model was trained and tested, and it produced high R² values of 0.972 for training and test sets, indicating that the selected predictors explain about 97% of the variation in exam scores. This implies that the extra variables enhance the accuracy of prediction relative to the simple regression model. The coefficients derived from the model also indicate the importance of each variable. For instance, "Hours Studied" and "Motivation Level" had positive coefficients, implying their direct effect on better exam scores. On the other hand, variables such as "Attendance" and "Distance from Home" had negative coefficients, reflecting an inverse relationship with performance.

Although multiple regression offers greater explanatory power, it is necessary to ensure that extra variables do not introduce multicollinearity, which may mask the relationships between predictors. The similarity of training and test R² values implies that the model generalizes well to new data, reducing overfitting.

Multiple regression is thus a superior technique in capturing the general factors affecting student performance, providing more detailed insights than simple regression. However, attribute selection is still important to ensuring interpretability and preventing unnecessary model complexity.

1. *Reflection on Attribute Selection and Its Impact on Model Performance*

The choice of predictor variables is an important aspect of constructing an efficient regression model. For the multiple linear regression model, the correlation matrix was utilized to choose attributes with high correlations with Exam Scores. The chosen predictors—Hours Studied, Previous Exam Scores, and Motivation Level—were chosen because they were highly correlated with the dependent variable. High R² values of the model (0.972 for both training and test datasets) indicate that the selected variables explain variations in student achievement substantially, explaining 97% of the variance in exam scores.

One of the interesting results of the attribute selection is that although Hours Studied alone was a good predictor for the simple regression model, the addition of Previous Exam Scores and Motivation Level led to a better understanding of student success. Previous Exam Scores had the greatest influence, thus supporting the hypothesis that past performance is a good indicator of future performance. Motivation Level, although with slightly lower influence, added context from the point of view of student effort and contribution.

However, the addition of too many predictors may add complexity without contributing significantly to predictive power. The addition of other variables, such as Parental Education Level or School Resources, only marginally added to R², and thus these variables, although relevant, may not have as direct an influence on individual student scores as study habits and past performance. Moreover, the addition of many predictors can lead to problems such as multicollinearity, making each variable's true effect hard to identify. Thus, the careful choice of the most influential predictors ensures the statistical integrity, interpretability, and utility of the model in facilitating decision-making.

VI. CONCLUSION AND REFLECTION

*a) Conclusion*

The present performance task illustrated the application of simple and multiple linear regression techniques in predicting exam scores based on various academic and behavioral predictors. The simple regression model, which employed only Hours Studied as a predictor, yielded a high R² value of 0.972 for training and test sets. This outcome highlights the close relationship between study time and performance in exams. The multiple regression technique, which incorporated Previous Exam Scores and Motivation Level as predictors, attained a more refined predictive model with an equal R² value of 0.972 for training and test sets. The employment of multiple predictors provides a broader perspective of student performance since it takes into consideration other influential variables beyond study duration.

The results validate that Previous Exam Scores have the greatest effect on current exam performance, followed by Hours Studied and Motivation Level. The equality in R² values between the training and testing processes indicates that the two models are highly generalizable to new data, thus lowering the risk of overfitting. More broadly, these results illustrate that while a single predictor can be very informative, the employment of multiple relevant features significantly enhances the precision and reliability of predictions.

*b) Reflection*

The regression analysis conducted on the data pertaining to student performance yielded useful insights into data preparation, model selection, and performance measurement. The methodology employed in the selection of predictor variables through correlation analysis emphasized the central role of feature selection in the development of precise predictive models. The examination of the impact of various predictors on exam scores was revealing, illustrating that some variables have a direct effect, while others have an effect in a more subtle manner.

Furthermore, this exercise emphasized the inherent trade-offs involved between model simplicity and predictive performance. The simple regression model offered greater interpretability; however, it was deficient in providing an overall explanation of performance. Conversely, the multiple regression model incorporated more variables but needed additional intensive analysis to provide ease of interpretation. This exercise emphasized the necessity of striking a balance between model complexity and practical application to attain optimal predictive results.

*c) Challenges*

One of the most important issues encountered in this endeavor was the selection of the most suitable features for the regression models. Too many predictors in the model threatened redundancy and multicollinearity, which could compromise the model's reliability. Achieving the best balance between including significant variables and having a parsimonious, interpretable model needed a close examination of correlation values in combination with domain expertise.

Another challenge was the interpretation of coefficients in the context of multiple regression. In simple regression, the effect of a single predictor is unambiguous, but in multiple regression, consideration of coefficients in the context of other predictors is necessary. The use of standardized coefficients and graphical plots assisted in gaining a better insight into the relative contribution of each variable to exam performance. Moreover, avoiding overfitting the model to the training data while simultaneously detecting substantive relationships was a significant objective, which necessitated repeated testing and validation.

Overall, this endeavor enhanced the comprehension of regression modeling and highlighted the significance of meticulous data selection in constructing accurate and generalizable predictive models.

VII. REFERENCES

Guyn, L. (2021). *Student performance factors* [Data set]. Kaggle.

<https://www.kaggle.com/datasets/lainguyn123/student-performance-factors>

Al Husaini, Y., & Ahmad Shukor, N. S. (2023). Factors affecting students' academic performance: A review. *ResearchGate*. <https://www.researchgate.net/publication/367360842_Factors_Affecting_Students%27_Academic_Performance_A_review>

Suleiman, I. B., Adekunle, O. O., Dada, E. G., & Ezeanya, C. U. (2024). Key factors influencing students' academic performance. Journal of Electrical Systems and Information Technology, 11(1). <https://www.researchgate.net/publication/384495579_Key_factors_influencing_students%27_academic_performance>

PT-M1: Comparing Logistic Regression with Decision Tree for Multinomial Classification

Kirvin Josh C. Castro

IT Elective III, BSIT

Jose Rizal University

Mandaluyong, Philippines

kirvinjosh.castro@my.jru.edu

Abstract— This study explores the application of Machine Learning techniques for mushroom classification, specifically comparing Logistic Regression and Decision Tree models. Using a dataset containing mushroom characteristics, the models were trained to predict whether a mushroom is edible or poisonous. The dataset underwent preprocessing, including handling missing values and encoding categorical variables. Performance metrics such as accuracy, precision, recall, and F1-score were evaluated. The results indicated that the Decision Tree model outperformed Logistic Regression in classification accuracy. This study highlights the importance of feature selection and model interpretability in mushroom classification.

Keywords— Mushroom Classification, Logistic Regression, Decision Tree, Machine Learning, Feature Engineering, Predictive Modeling.

I. INTRODUCTION

Mushroom classification has long been a fundamental challenge in the field of mycology. Accurate identification of mushrooms is critical, especially for distinguishing between edible and poisonous species. With thousands of mushroom species existing worldwide, misidentifications can lead to serious health risks, including poisoning and even death. Traditionally, identifying mushrooms has relied heavily on physical characteristics such as cap shape, gill structure, color, and habitat, but this method is labor-intensive and requires significant expertise. Moreover, the sheer diversity of mushrooms and the subtle differences between species make manual identification prone to errors, especially by non-experts.

In recent years, Machine Learning (ML) has emerged as a powerful tool for automating classification tasks, providing more consistent and accurate results. By leveraging computational algorithms to analyze large datasets, ML models can detect patterns and relationships in the data that might be overlooked by human experts. This study applies two widely-used ML models—Logistic Regression and Decision Tree—to classify mushrooms as either edible or poisonous based on their characteristics.

The dataset used in this study, "Mushroom Classification" from Kaggle, contains various attributes that describe the physical and chemical properties of mushrooms. These attributes include characteristics such as cap shape, color, odor, gill size, and habitat. The goal of this research is to explore and compare the performance of Logistic Regression and Decision Tree models in classifying mushrooms based on these features.

DATA COLLECTION

1. Dataset Description

The study relies on a thorough educational dataset that includes various performance-influencing academic, personal and environmental metrics. The educational data repository served as the reliable source for data acquisition which provided high-value information needed for predictive modeling purposes. The dataset contains 6,054 records which use 20 attributes to measure the factors that affect the performance outcome variable Exam\_Score.

The dataset contains three diverse continuous metrics: Hours\_Studied, Attendance, and Previous\_Scores that demonstrate meaningful potential as predictor variables. The researcher encoded the categorical variables Parental\_Involvement, Access\_to\_Resources, and Motivation\_Level using ordinal and binary schemes so they would maintain their natural relationships. The dataset maintains complete data integrity since all entries follow a reliable structure that avoids missing or duplicate information.

df.info() was also executed, in order to display the number of records, column names, its data types, memory usage, and non-null counts. The command helps the researchers to identify missing values and analyze the dataset structure. Df.dtypes() was also executed to display the data types of each attributes in the Data Frame. This further helps in data preprocessing and analysis as it determines whether the attribute contains numerical or categorical data.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Fig. 3. df.info() command execution

2. Dataset Attributes

The Mushroom Classification dataset from Kaggle contains a total of 8,124 records and 22 attributes, describing the characteristics of different mushroom species. The dataset is entirely categorical and includes features such as cap shape, color, gill size, and habitat. The target variable is binary, indicating whether a mushroom is edible (represented as 'e') or poisonous (represented as 'p').

Dataset Attributes:

1. class: The target variable (edible or poisonous).

2. cap\_shape: Shape of the mushroom cap (bell, conical, flat, knobbed, sunken).

3. cap\_surface: Surface of the cap (fibrous, grooves, scaly, smooth).

4. cap\_color: Color of the cap (brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow).

5. bruises: Whether the mushroom bruises or not (bruises or no).

6. odor: Odor of the mushroom (almond, anise, creosote, fishy, foul, musty, none, pungent, spicy).

7. gill\_attachment: Attachment of gills (attached, descending).

8. gill\_spacing: Spacing between gills (close, crowded).

9. gill\_size: Size of gills (broad, narrow).

10. gill\_color: Color of gills (black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow).

11. stalk\_shape: Shape of the stalk (enlarging, tapering).

12. stalk\_root: Type of stalk root (bulbous, club, cup, equal, rhizomorphs, rooted, missing).

13. stalk\_surface\_above\_ring: Surface of the stalk above the ring (fibrous, scaly, silky, smooth).

14. stalk\_surface\_below\_ring: Surface of the stalk below the ring (fibrous, scaly, silky, smooth).

15. stalk\_color\_above\_ring: Color of the stalk above the ring (brown, buff, cinnamon, gray, green, orange, pink, red, white, yellow).

16. stalk\_color\_below\_ring: Color of the stalk below the ring (brown, buff, cinnamon, gray, green, orange, pink, red, white, yellow).

17. veil\_type: Type of veil (partial, universal).

18. veil\_color: Color of the veil (brown, orange, white, yellow).

19. ring\_number: Number of rings on the stalk (none, one, two).

20. ring\_type: Type of ring on the stalk (evanescent, large, none, pendant).

21. spore\_print\_color: Color of spore print (black, brown, buff, chocolate, green, orange, purple, white, yellow).

22. population: Population of mushrooms in the habitat (abundant, clustered, numerous, scattered, solitary).

23. habitat: Habitat where the mushroom was found (grasses, leaves, meadows, paths, urban, waste).

Data Cleaning and Data Preprocessing

The dataset has no missing values, so no imputation is needed. We perform basic checks to confirm this by using the df.isnull().sum() function, which checks for null values in each column.

a) Data preprocessing for further data cleaning

The output of df.head() displays the first five rows of a dataset containing various factors influencing student performance. Key attributes include Hours\_Studied, Attendance, Parental\_Involvement, and Previous\_Scores, which reflect study habits, school engagement, and past academic performance. Other factors like Access\_to\_Resources, Internet\_Access, and Family\_Income indicate external support and socio-economic conditions. Additionally, Motivation\_Level, Sleep\_Hours, and Tutoring\_Sessions highlight personal and academic behaviors that may impact success. The dataset also includes Teacher\_Quality and School\_Type, which provide insights into educational environments. These variables help analyze student performance through multiple regression analysis.

Fig. 4. df.head() command execution

The histograms show numerical data distribution patterns alongside the assessment of both skewness and detection of outliers in the dataset. Most students fall within the normal range when it comes to hours spent studying based on the normal distribution pattern of the Hours\_Studied attribute. The distribution of attendance data is even across all options which indicates different students have different levels of regularity in their presence. Each categorical attribute produces significant spikes in the data which corresponds with the specific available values. Most students maintain a regular sleep pattern by distributing hours they sleep around the 6-to-8-hour mark. The academic performance levels show wide distribution among students in Previous\_Scores. The data attributes Internet\_Access, Family\_Income, and Teacher\_Quality show signs of being categorical since their values are limited and binary or ordinal in nature. Data trends become easier to understand through this visualization method which also helps with preprocessing strategy selection.

Fig. 5. histogram command execution

b.) Identifying the missing values

To better understand the dataset, we performed an examination of the number of unique values and missing values in each column. This is crucial in deciding the type of feature it is, whether it is a categorical feature, binary feature, or a continuous feature. It also assists us in making sure that our dataset is complete and free from missing values, thus reducing the imputation process. Columns with many unique values, like Attendance and Previous\_Scores, have a large number of possible values, while columns like Parental\_Involvement and Internet\_Access are mostly categorical. From this examination, we are better placed to determine the proper ways of processing and modeling the data.

Fig. 6. Displaying the unique values and the missing values

The below code snippet performs an analysis of a dataset to identify columns that have only unique values for every single row. The script iterates through all columns of the DataFrame in a systematic manner and checks if the number of unique values in a column matches the total number of rows available. If a column meets this criterion, it is then added to an aggregated list. However, as the results show, there are no columns in this dataset that have entirely unique values in every row. This finding suggests that the dataset does not have a clear identifier column, like an ID or unique key.

Fig. 7. Identifying unique columns where row has a unique value

The given line of code is meant to check for the uniqueness of each column in the dataset. Through the looping of each column and printing their corresponding values, the operation allows for the identification of categorical variables, detection of data anomalies, and checking of the numerical range. This kind of functionality is very critical in exploratory data analysis (EDA), since the knowledge of individual values in each feature is critical in data cleaning, feature generation, and

model preparation.

Fig. 8. Uniqueness of each column in the dataset.

The code below enables the transformation of categorical data into a numeric format through the assignment of unique integers to each category. The code systematically detects categorical columns, assigns their unique values to integers through mapping, and applies these mappings to the dataset. This transformation facilitates the use of qualitative data in machine learning models.

Fig 10. Binary Mappings

This script facilitates the encoding of categorical variables into numeric form, hence making them compatible with machine learning models. It goes through all the categorical columns methodically, assigning each unique value an integer and performing the transformation on the whole dataset. For ordinal variables, which have a specific order (e.g., "Low" to "High"), the encoding is from 0 to n, whereas binary variables such as "Yes/No" are assigned values 0 and 1. Nominal variables, which lack a natural order, are assigned random integer values. This is crucial in transforming qualitative data into an appropriate format for efficient algorithmic processing.

The given output specifies the binary encoding parameters for the ordinal variables in the dataset. For ordinal variables, which have a certain order, the encoding is given from 0 to 2. This format is used for variables such as Parental\_Involvement, Access\_to\_Resources, Motivation\_Level, Family\_Income, Teacher\_Quality, Parental\_Education\_Level, and Distance\_from\_Home, where values with labels "Low" are encoded as 0 and those with labels "High" are encoded as 2. Binary encoding is used for dichotomous or binary-category variables, which are Extracurricular\_Activities, Internet\_Access, Learning\_Disabilities, and Gender, where one category is represented as 0 and the other as 1. For nominal variables that do not have any order, such as School\_Type and Peer\_Influence, the classes are represented as 0, 1, and 2.

c. Handling Missing Values

The code effectively identifies and addresses missing values in a DataFrame. It starts by counting the missing values in every column through df.isnull().sum(), saving the output in missing\_values, and then prints the number of missing values in every column. The next is the elimination of columns that have only missing values through df.dropna(axis=1, how='all', inplace=True), thereby ensuring the elimination of fully empty columns. It then removes any rows that contain missing values through df.dropna(inplace=True), thereby safeguarding the integrity of the analysis from missing data. Finally, it prints the modified dimensions of the DataFrame, thereby ensuring the verification of the changes made.

Fig. 12. Handling the missing values

Fig. 13. Output of handling the missing values

The output confirms the lack of any missing values across the columns in the dataset. This fact attests that the dataset is complete and ready to be analyzed, eliminating the requirement for imputation or records elimination. After fixing missing values, the shape of the DataFrame is (6054, 20), indicating there are 6,054 observations and 20 variables. Such a requirement ensures the reliability of the statistical outputs and the training processes of the models.

d.) Identifying Duplicate Values

The code below identifies and removes duplicate rows in a DataFrame. It first calculates the number of duplicate rows using df.duplicated().sum(), which checks for duplicate entries and sums them up. The result is stored in duplicate\_count and printed to inform the user about the number of duplicate rows present. Then, the code removes these duplicate rows using df.drop\_duplicates(inplace=True), modifying the DataFrame directly to ensure that only unique rows remain. This helps in cleaning the dataset by eliminating redundant data, improving the efficiency and accuracy of further analysis.

Fig. 14. Counting the number of duplicate rows

The result shows that there are no duplicate rows in the dataset. This indicates that all the observations are unique, ensuring the integrity of the data and reducing the risk of bias in the analysis.

Fig. 15. The number of duplicate rows

e.) Outlier Detection

Here is a code snippet that detects and removes outliers from a DataFrame through the Interquartile Range (IQR) method. The function detect\_outliers\_iqr initially loops through all numerical columns, calculates the first (Q1) and third quartiles (Q3), and calculates the interquartile range (IQR). It then calculates lower and upper bounds by multiplying the IQR by a factor of 1.5 and then identifies values outside this range as outliers. These outliers are counted in a dictionary. For ease of visualization, the code plots boxplots for all columns that have outliers, thus making it easy to visualize the distribution and extreme values involved. The function remove\_outliers\_iqr then goes ahead to remove these outliers, leaving only those values that fall within the calculated IQR range, thus generating a cleaned DataFrame that is free from extreme values. This process helps in improving data integrity by removing extreme values from affecting future analysis and machine learning models.

Fig. 16. Detecting the outliers

Fig. 17. Boxplot of values (with outliers)

The boxplots of the variables Internet\_Access and Learning\_Disabilities show the presence of extreme outliers. Both variables have a steep concentration of points around zero with some extreme values on the upper end of the distribution. This shows that most of the data points are bunched in one category (presumably indicating minimal or non-access/the absence of the condition) with a few observations in a deviant category. The steep difference between the central cluster and the outliers can be an indicator of data entry problems, rare occurrences, or extreme heterogeneity in these attributes. Additional research needs to be conducted to determine if the outliers are valid observations or spurious observations that need to be cleaned up by data preprocessing techniques, such as transformation, capping, or deletion, depending on the research setting.

III. ATTRIBUTE SELECTION

Feature selection is a central data preprocessing module in machine learning whose aim is to identify the most informative features and eliminate redundant and high-correlation features. We use this technique to improve models, reduce overfitting, and enhance interpretability. There are many ways to competent feature selection, including statistical analysis, domain knowledge, correlation tests, feature vectorization, and dimensionality reduction methods like Principal Component Analysis (PCA). By using these methods wisely, practitioners are able to develop predictive models that are accurate and efficient.

Correlation analysis was used in this study to select the most important attributes from data. Correlation analysis is used to measure the strength of relationships between numerical features and determine the degree to which they affect one another. Feature attributes that have high correlations are likely to lead to redundancy and need to be excluded to preserve the efficiency and objectivity of the model. By focusing on attributes characterized by high and significant correlations, the dataset is reduced to include only the most predictive variables, enhancing the precision of machine learning models while decreasing complexity.

Besides, the use of correlation analysis facilitates data-driven decision-making, ensuring that information obtained is reliable and actionable.

a.) Correlation Analysis Heatmap

A heatmap visualization was utilized in this study to analyze the relationships between attributes in the dataset, providing a correlation matrix for all numerical features. This visualization helps in understanding the degree of association between different variables.

In the heatmap:

• Dark red shades indicate a strong positive correlation (closer to 1), meaning an increase in one variable is associated with an increase in another.

• Dark blue shades represent a strong negative correlation (closer to -1), meaning an increase in one variable is associated with a decrease in another.

• Neutral shades (near 0) indicate weak or no correlation between variables.

• Diagonal values are always 1, as a variable is perfectly correlated with itself.

The visualization assists in the determination of the redundant variables that are responsible for multicollinearity, which negatively impacts predictive modeling. It also determines the features with the highest impact on the target variable, i.e., Exam\_Score. The heatmap reveals that Attendance (0.68) and Hours\_Studied (0.50) have the highest positive correlation with Exam\_Score, and therefore they are the most significant predictors of student performance. Gender (-0.50) is negatively correlated, indicating a gender-based disparity in examination performance.

Through correlation analysis, it is now possible to select the most informative features and reject the redundant or highly correlated features and thereby construct an efficient and effective predictive model.

Fig. 18. Correlation Analysis Heatmap

b.) Multicollinearity Check

Multicollinearity is a common issue in regression analysis, where there is high correlation among independent variables. Multicollinearity makes it difficult to interpret model coefficients and produces predictions that are not reliable. Multicollinearity is checked with the Variance Inflation Factor (VIF), which gives a measure of the extent to which the variance of a regression coefficient is inflated by multicollinearity. High VIF value (typically above 10) signifies high collinearity, and the respective variable must be removed or transformed.

The code in question calculates VIF values for all the predictor variables in the data. First, the predictor variables from the cleaned data (df\_cleaned) are extracted. Second, a DataFrame is initialized to store the VIF values for each feature. The variance\_inflation\_factor function from statsmodels is used to calculate the VIF scores for each variable. Lastly, the VIF values are printed, thus making it simple to identify features that could potentially create multicollinearity problems and need to be removed or transformed in the model.

Fig. 19. Multicollinearity Check

The Variance Inflation Factor (VIF) test reveals potential multicollinearity among the predictor variables. Specifically, the "Attendance," "Previous\_Scores," and "Sleep\_Hours" features have comparatively high VIF values (above 10), which suggests high multicollinearity that can have a negative impact on the stability of the model and coefficient interpretation. Other features have moderate VIF values, generally below 5, suggesting acceptable multicollinearity. NaN values for "Internet\_Access" and "Learning\_Disabilities" can be due to constant or highly correlated features.

Fig. 20. Output of the Multicollinearity Check

IV. MODEL BUILDING

a) Simple Linear Regression

Q-Q plot of residuals visually verifies whether residuals from the regression model are normally distributed. The red reference line indicates perfect normal distribution, and the blue points indicate actual residuals. If residuals are normally distributed, then points must be very close to the red line. In this scenario, most points are along the line, which indicates that the normality assumption is satisfied to a significant extent. Slight deviations at the tail indicate slight skewness or the presence of outliers. Generally, the residuals are in a pattern consistent with normality, which makes the reliability of the model's inferences and predictions justified.

Normality of residuals is important in the validation of linear regression assumptions because it impacts confidence interval accuracy and hypothesis testing accuracy. If residuals are normally distributed, then the model's prediction is more reliable, and statistical inferences regarding variables' relationships are more reliable. Non-normality, especially in the tails, can indicate potential problems such as heteroscedasticity, omitted variables, or the requirement for data transformation. Although slight deviations are noted in the Q-Q plot, overall compliance of the residuals with the reference line means that the model satisfies the normality assumption to a reasonable extent, which makes its capacity to detect underlying patterns in the data justified.

Fig. 21. Q-Q Plot of the Residuals

The Q-Q plot of the residuals reveals that the residuals trace a very normal distribution as most of the points fall along the red reference line. This arrangement reveals that the normality assumption of the residuals is generally satisfied, a crucial prerequisite for linear regression models. Minimal deviations are, however, observed at the tails, especially at the top, which reveals minor skewness or the existence of outliers. The residuals generally trace a pattern akin to normal distribution, hence testifying to the reliability of the model's inferences and predictions.

b.) Multiple Linear Regression

Multiple Linear Regression is a sophisticated version of simple linear regression, trying to explain the relationship between a target variable (dependent variable) and several predictor variables (independent variables). The technique is used to investigate the effect of several factors on a particular outcome, thus being a major tool in predictive modeling.

In this research, we will create a multiple linear regression model intended to predict Exam Scores based on a group of determinants such as study hours, parental influence, previous scores, and other determinants. The model will be trained on a data set, and its performance will be evaluated using R-squared measures for the training and test sets.

The following code is required to perform multiple linear regression, including steps such as data preprocessing, splitting data into training and test sets, model training, and evaluating its performance.

Fig. 22. Multiple Linear Regression Code

The multiple linear regression output shows the impact of each predictor on the target variable. The intercept is about 38.92, the value at baseline when all the predictors are zero. The coefficients show the direction and strength of the impact of each predictor. "Hours\_Studied" and "Motivation\_Level" have positive coefficients, since higher values produce better outcomes. "Attendance" and "Distance\_from\_Home" have negative coefficients, since there's a reverse relationship. The large values of R-squared for training (0.972) as well as for testing (0.972) show the model fits perfectly, with about 97% of the target variance accounted for by the predictors, showing the power of the model and generalizability.

Fig. 23. Results from using the Multiple Linear Regression

V. DISCUSSION

a) Model Performance Evaluation of both models

Simple Linear Regression Analysis

The simple linear regression model was used to predict student exam scores from one predictor variable. The Q-Q plot of residuals was inspected to find out if such residuals are normally distributed, which is a basic assumption of regression analysis. The red reference line on the graph represents a perfect normal distribution, and the blue dots on the graph represent the actual residuals. If the residuals are normally distributed, the dots are supposed to closely follow the red line.

In this analysis, most of the dots follow the reference line, confirming that the residuals are close to normally distributed. Divergences at extremes show potential skewness or outliers. Despite such exceptions, the overall trend is as expected under normality, thus confirming the validity of the conclusions and predictions from the model. The high R² values of both the training and test sets also confirm that the chosen predictor has a very high correlation with exam scores, hence a valid single-variable model.

Normality in residuals is essential for the verification of model assumptions, which influence the reliability of confidence intervals and hypothesis testing. Abnormality can imply heteroscedasticity, omitted variables, or the need for transformation. However, since the residuals are generally in accordance with normality, this basic linear regression model provides a good foundation for the prediction of exam scores based on a single factor.

Multiple Linear Regression Analysis

Multiple linear regression expands simple regression by using multiple independent variables to forecast exam scores, providing a broader picture of the variables affecting student performance. Predictors here included hours studied, past scores, and other academic characteristics. By considering multiple factors, this model provides a better picture of how individual factors affect exam performance.

The model was trained and tested, and it produced high R² values of 0.972 for training and test sets, indicating that the selected predictors explain about 97% of the variation in exam scores. This implies that the extra variables enhance the accuracy of prediction relative to the simple regression model. The coefficients derived from the model also indicate the importance of each variable. For instance, "Hours Studied" and "Motivation Level" had positive coefficients, implying their direct effect on better exam scores. On the other hand, variables such as "Attendance" and "Distance from Home" had negative coefficients, reflecting an inverse relationship with performance.

Although multiple regression offers greater explanatory power, it is necessary to ensure that extra variables do not introduce multicollinearity, which may mask the relationships between predictors. The similarity of training and test R² values implies that the model generalizes well to new data, reducing overfitting.

Multiple regression is thus a superior technique in capturing the general factors affecting student performance, providing more detailed insights than simple regression. However, attribute selection is still important to ensuring interpretability and preventing unnecessary model complexity.

b) Reflection on Attribute Selection and Its Impact on Model Performance

The choice of predictor variables is an important aspect of constructing an efficient regression model. For the multiple linear regression model, the correlation matrix was utilized to choose attributes with high correlations with Exam Scores. The chosen predictors—Hours Studied, Previous Exam Scores, and Motivation Level—were chosen because they were highly correlated with the dependent variable. High R² values of the model (0.972 for both training and test datasets) indicate that the selected variables explain variations in student achievement substantially, explaining 97% of the variance in exam scores.

One of the interesting results of the attribute selection is that although Hours Studied alone was a good predictor for the simple regression model, the addition of Previous Exam Scores and Motivation Level led to a better understanding of student success. Previous Exam Scores had the greatest influence, thus supporting the hypothesis that past performance is a good indicator of future performance. Motivation Level, although with slightly lower influence, added context from the point of view of student effort and contribution.

However, the addition of too many predictors may add complexity without contributing significantly to predictive power. The addition of other variables, such as Parental Education Level or School Resources, only marginally added to R², and thus these variables, although relevant, may not have as direct an influence on individual student scores as study habits and past performance. Moreover, the addition of many predictors can lead to problems such as multicollinearity, making each variable's true effect hard to identify. Thus, the careful choice of the most influential predictors ensures the statistical integrity, interpretability, and utility of the model in facilitating decision-making.

VI. CONCLUSION AND REFLECTION

a) Conclusion

The present performance task illustrated the application of simple and multiple linear regression techniques in predicting exam scores based on various academic and behavioral predictors. The simple regression model, which employed only Hours Studied as a predictor, yielded a high R² value of 0.972 for training and test sets. This outcome highlights the close relationship between study time and performance in exams. The multiple regression technique, which incorporated Previous Exam Scores and Motivation Level as predictors, attained a more refined predictive model with an equal R² value of 0.972 for training and test sets. The employment of multiple predictors provides a broader perspective of student performance since it takes into consideration other influential variables beyond study duration.

The results validate that Previous Exam Scores have the greatest effect on current exam performance, followed by Hours Studied and Motivation Level. The equality in R² values between the training and testing processes indicates that the two models are highly generalizable to new data, thus lowering the risk of overfitting. More broadly, these results illustrate that while a single predictor can be very informative, the employment of multiple relevant features significantly enhances the precision and reliability of predictions.

b) Reflection

The regression analysis conducted on the data pertaining to student performance yielded useful insights into data preparation, model selection, and performance measurement. The methodology employed in the selection of predictor variables through correlation analysis emphasized the central role of feature selection in the development of precise predictive models. The examination of the impact of various predictors on exam scores was revealing, illustrating that some variables have a direct effect, while others have an effect in a more subtle manner.

Furthermore, this exercise emphasized the inherent trade-offs involved between model simplicity and predictive performance. The simple regression model offered greater interpretability; however, it was deficient in providing an overall explanation of performance. Conversely, the multiple regression model incorporated more variables but needed additional intensive analysis to provide ease of interpretation. This exercise emphasized the necessity of striking a balance between model complexity and practical application to attain optimal predictive results.

c) Challenges

One of the most important issues encountered in this endeavor was the selection of the most suitable features for the regression models. Too many predictors in the model threatened redundancy and multicollinearity, which could compromise the model's reliability. Achieving the best balance between including significant variables and having a parsimonious, interpretable model needed a close examination of correlation values in combination with domain expertise.

Another challenge was the interpretation of coefficients in the context of multiple regression. In simple regression, the effect of a single predictor is unambiguous, but in multiple regression, consideration of coefficients in the context of other predictors is necessary. The use of standardized coefficients and graphical plots assisted in gaining a better insight into the relative contribution of each variable to exam performance. Moreover, avoiding overfitting the model to the training data while simultaneously detecting substantive relationships was a significant objective, which necessitated repeated testing and validation.

Overall, this endeavor enhanced the comprehension of regression modeling and highlighted the significance of meticulous data selection in constructing accurate and generalizable predictive models.

VII. REFERENCES

Guyn, L. (2021). Student performance factors [Data set]. Kaggle.

https://www.kaggle.com/datasets/lainguyn123/student-performance-factors

Al Husaini, Y., & Ahmad Shukor, N. S. (2023). Factors affecting students' academic performance: A review. ResearchGate. https://www.researchgate.net/publication/367360842\_Factors\_Affecting\_Students%27\_Academic\_Performance\_A\_review

Suleiman, I. B., Adekunle, O. O., Dada, E. G., & Ezeanya, C. U. (2024). Key factors influencing students' academic performance. Journal of Electrical Systems and Information Technology, 11(1). https://www.researchgate.net/publication/384495579\_Key\_factors\_influencing\_students%27\_academic\_performance