Table of Contents

1. Introduction

In the ever-evolving landscape of education, understanding the myriad factors influencing student success is imperative for educators, policymakers, and researchers alike. This data science project delves into the realm of secondary education, specifically focusing on student achievement in Mathematics and Portuguese subjects within two Portuguese schools, Gabriel Pereira (GP) and Mousinho da Silveira (MS). The rich dataset encompasses a diverse set of features, including demographic, social, and school-related attributes, collected through school reports and questionnaires.

The overarching goal of this project is to unravel the intricate web of factors that impact student success, particularly in the context of Mathematics and Portuguese performance. Through a comprehensive analysis, I aim to build predictive models that shed light on the significant variables influencing academic outcomes. This endeavor holds the potential to provide valuable insights for educators, allowing them to tailor interventions and support mechanisms to address specific challenges faced by students.

The report is structured to unfold the journey from data preprocessing to model building, leveraging a variety of techniques from data exploration to machine learning. Each section contributes to a holistic understanding of the dataset, leading to actionable conclusions that can inform educational strategies and policies.

Embarking on this exploration, the subsequent sections will delve into the background of the data, the preprocessing steps taken to refine the dataset, an in-depth exploration of key features, and the development of predictive models tailored to the unique requirements of the project.

2. Data Information

The data in the file is about student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features and it was collected by using school reports and questionnaires.

1. **school:**
    - Student’s school

- Values: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira

2. **sex:**
    - Student’s sex
    - Values: "F" - female or "M" - male
3. **age:**
    - Student’s age
    - Values: Numeric from 15 to 22
4. **address:**
    - Student’s home address type
    - Values: "U" - urban or "R" - rural
5. **famsize:**
    - Family size
    - Values: "LE3" - less or equal to 3 or "GT3" - greater than 3
6. **Pstatus:**
    - Parent’s cohabitation status
    - Values: "T" - living together or "A" - apart
7. **Medu:**
    - Mother’s education
    - Values: Numeric - 0 (none), 1 (primary education), 2 (5th to 9th grade), 3 (secondary education), 4 (higher education)
8. **Fedu:**
    - Father’s education
    - Values: Numeric - 0 (none), 1 (primary education), 2 (5th to 9th grade), 3 (secondary education), 4 (higher education)
9. **Mjob:**
    - Mother’s job
    - Values: "teacher", "health" care related, "services", "at_home" or "other"
10. **Fjob:**
    - Father’s job
    - Values: "teacher", "health" care related, "services", "at_home" or "other"
11. **reason:**

- Reason to choose this school
- Values: "home", "reputation", "course" preference or "other"

12. **guardian:**
    - Student’s guardian
    - Values: "mother", "father" or "other"
13. **traveltime:**
    - Home to school travel time
    - Values: Numeric - 1 (<15 min.), 2 (15 to 30 min.), 3 (30 min. to 1 hour), 4 (>1 hour)
14. **studytime:**
    - Weekly study time
    - Values: Numeric - 1 (<2 hours), 2 (2 to 5 hours), 3 (5 to 10 hours), 4 (>10 hours)
15. **failures:**
    - Number of past class failures
    - Values: Numeric - n if 1<=n<3, else 4
16. **schoolsup:**
    - Extra educational support
    - Values: Yes or No
17. **famsup:**
    - Family educational support
    - Values: Yes or No
18. **paid:**
    - Extra paid classes within the course
    - Values: Yes or No
19. **subject:**
    - Subject
    - Values: Yes or No
20. **activities:**
    - Extra-curricular activities
    - Values: Yes or No
21. **nursery:**

- Attended nursery school
- Values: Yes or No

22. **higher:**
    - Wants to take higher education
    - Values: Yes or No
23. **internet:**
    - Internet access at home
    - Values: Yes or No
24. **romantic:**
    - With a romantic relationship
    - Values: Yes or No
25. **famrel:**
    - Quality of family relationships
    - Values: Numeric - from 1 (very bad) to 5 (excellent)
26. **freetime:**
    - Free time after school
    - Values: Numeric - from 1 (very low) to 5 (very high)
27. **goout:**
    - Going out with friends
    - Values: Numeric - from 1 (very low) to 5 (very high)
28. **Dalc:**
    - Workday alcohol consumption
    - Values: Numeric - from 1 (very low) to 5 (very high)
29. **Walc:**
    - Weekend alcohol consumption
    - Values: Numeric - from 1 (very low) to 5 (very high)
30. **health:**
    - Current health status
    - Values: Numeric - from 1 (very bad) to 5 (very good)
31. **absences:**
    - Number of school absences

- Values: Numeric - from 0 to 93

32. **G1:**
    - First period grade
    - Values: Numeric - from 0 to 20
33. **G2:**
    - Second period grade
    - Values: Numeric - from 0 to 20
34. **G3:**
    - Final grade
    - Values: Numeric - from 0 to 20

3. Data Preprocessing

3.1 Loading and Initial Exploration:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# Loading the dataset
dataset = pd.read_csv("exam_data.csv")

# Displaying the first and last few rows of the dataset
dataset.head()
dataset.tail()
```

3.2 Data Summary and Inspection:

```
# Displaying data types and summary statistics
dataset.dtypes
dataset.describe()

# Describing categorical data
dataset.describe(include="all")

# Displaying information about the dataset
dataset.info()

# Checking for null values
dataset.isnull().any()
```

The dataset consists of 751 entries with 40 columns, encompassing a variety of features ranging from demographic information to academic performance in both Mathematics and Portuguese subjects. No missing values are observed in the dataset, ensuring completeness.

3.3 Handling Unnecessary Columns:

```
# Dropping unnecessary columns
dataset = dataset.drop(['Unnamed: 0', "reason", "Fedu", "Mjob"], axis=1)
dataset.head()
```

Columns like 'Unnamed: 0', 'reason', 'Fedu', and 'Mjob' are dropped as they are deemed unnecessary for the analysis.

3.4 Handling Non-Numeric Values and One-Hot Encoding:

Non-numeric values in the dataset are identified and one-hot encoded using the ColumnTransformer and OneHotEncoder. This results in an expanded feature set, as observed in the changed shape of X.

Columns with non-numeric values: [0, 1, 3, 4, 5, 7, 8, 11, 12, 13, 14, 15, 16, 17, 25, 31]

The X shape then changed from (751, 35) to (751, 55).

```
# Separating features (X) and target variable (y)
target_column = 'G1.Port'
X = pd.DataFrame(dataset.drop(columns=[target_column]).values)
y = pd.DataFrame(dataset[target_column].values)

# Convert each column to numeric, coerce non-numeric values to NaN
numeric_check = X.apply(pd.to_numeric, errors='coerce')

# Get columns with non-numeric values
non_numeric_columns = numeric_check.columns[numeric_check.isnull().any()].tolist()

# Displaying columns with non-numeric values
print("Columns with non-numeric values:", non_numeric_columns)

# One-hot encoding the non-numeric columns
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), non_numeric_columns)], remainder="passthrough")
X = np.array(ct.fit_transform(X))

# Displaying the shape and transformed data
print(y.shape)
print(X.shape)
print(X)
```

3.5 Train-Test Split:

```
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

The dataset is divided into training and testing sets to facilitate model training and evaluation.

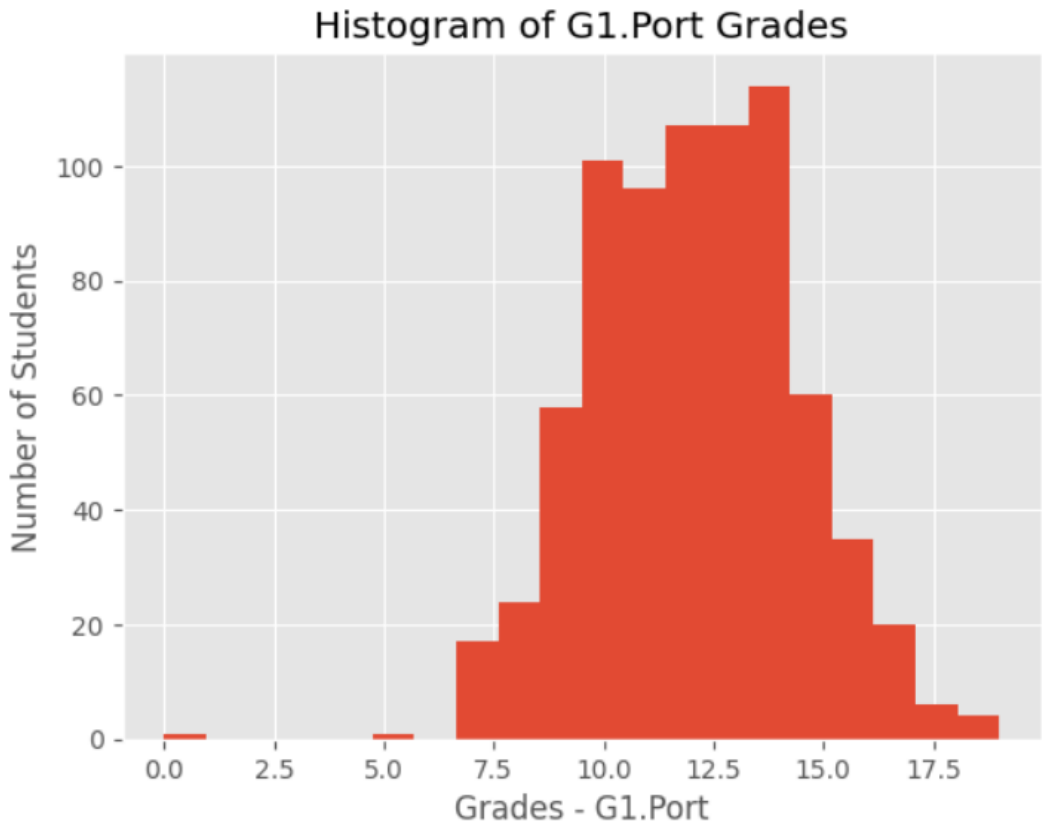
The Data Preprocessing section lays the foundation for subsequent analysis, ensuring that the dataset is appropriately prepared for modeling. The next sections will delve into Data Exploration, Data Analysis, and present the Results and Conclusions.

4. Data Exploration

4.1 Target Variable Distribution:

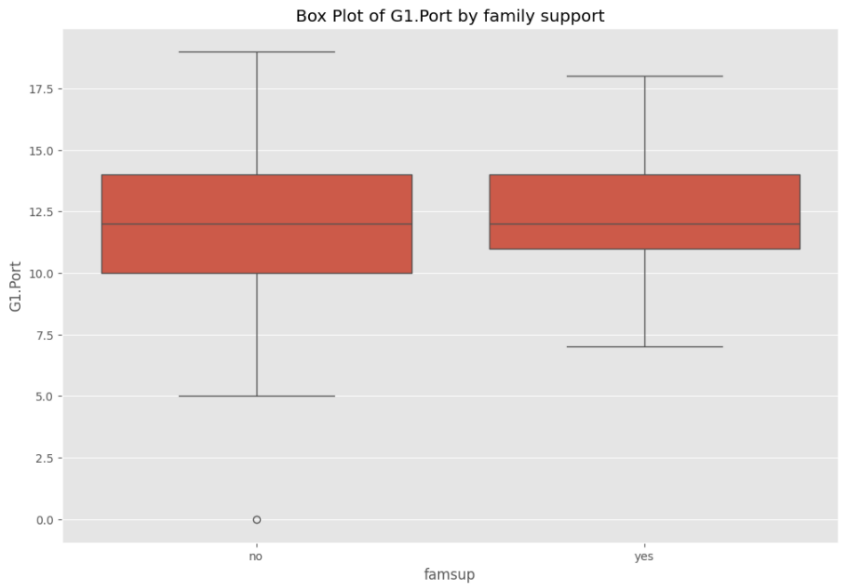
```
plt.style.use('ggplot')
dataset['G1.Port'].plot.hist(title='Histogram of G1.Port Grades', bins=20)
plt.xlabel('Grades - G1.Port')
plt.ylabel('Number of Students')
plt.show()
```

The histogram visually represents the distribution of grades in the target variable 'G1.Port'. This provides insights into the spread and concentration of student performance in the initial Portuguese subject grades.



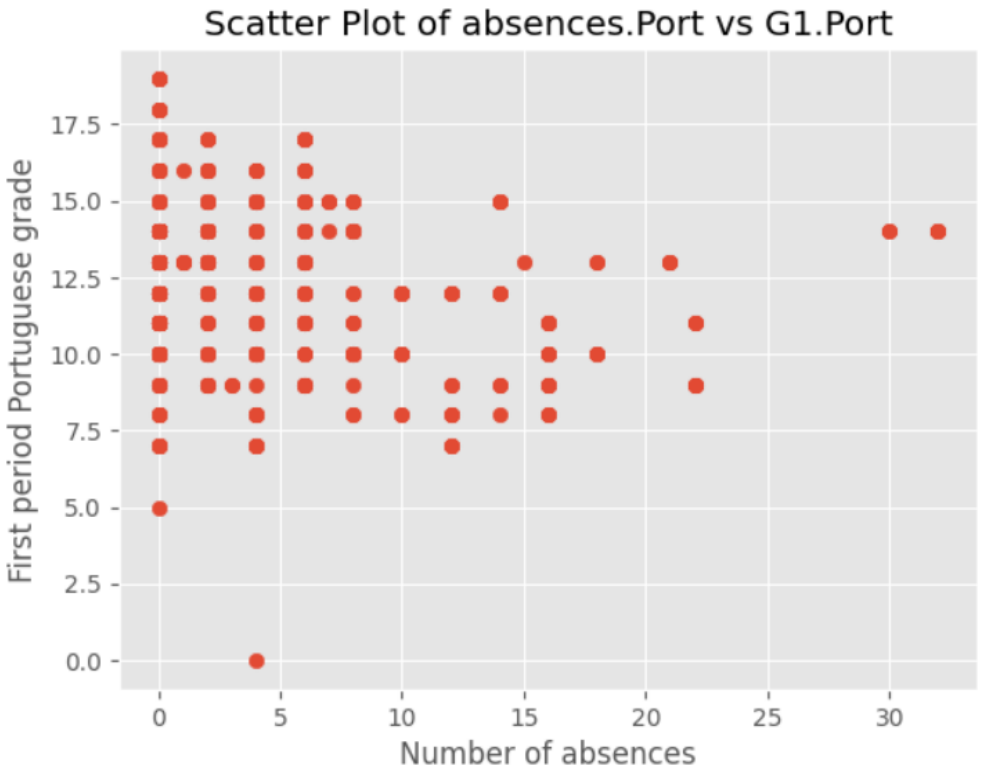
4.2 Family support and G1 grade effect

I wanted to visualize the distribution of the first period grades in Portuguese ('G1.Port') based on the presence or absence of family support ('famsup'). It is noted that the with family support, students grade doesn't go below 11.



4.3 Number of Absence and the grade

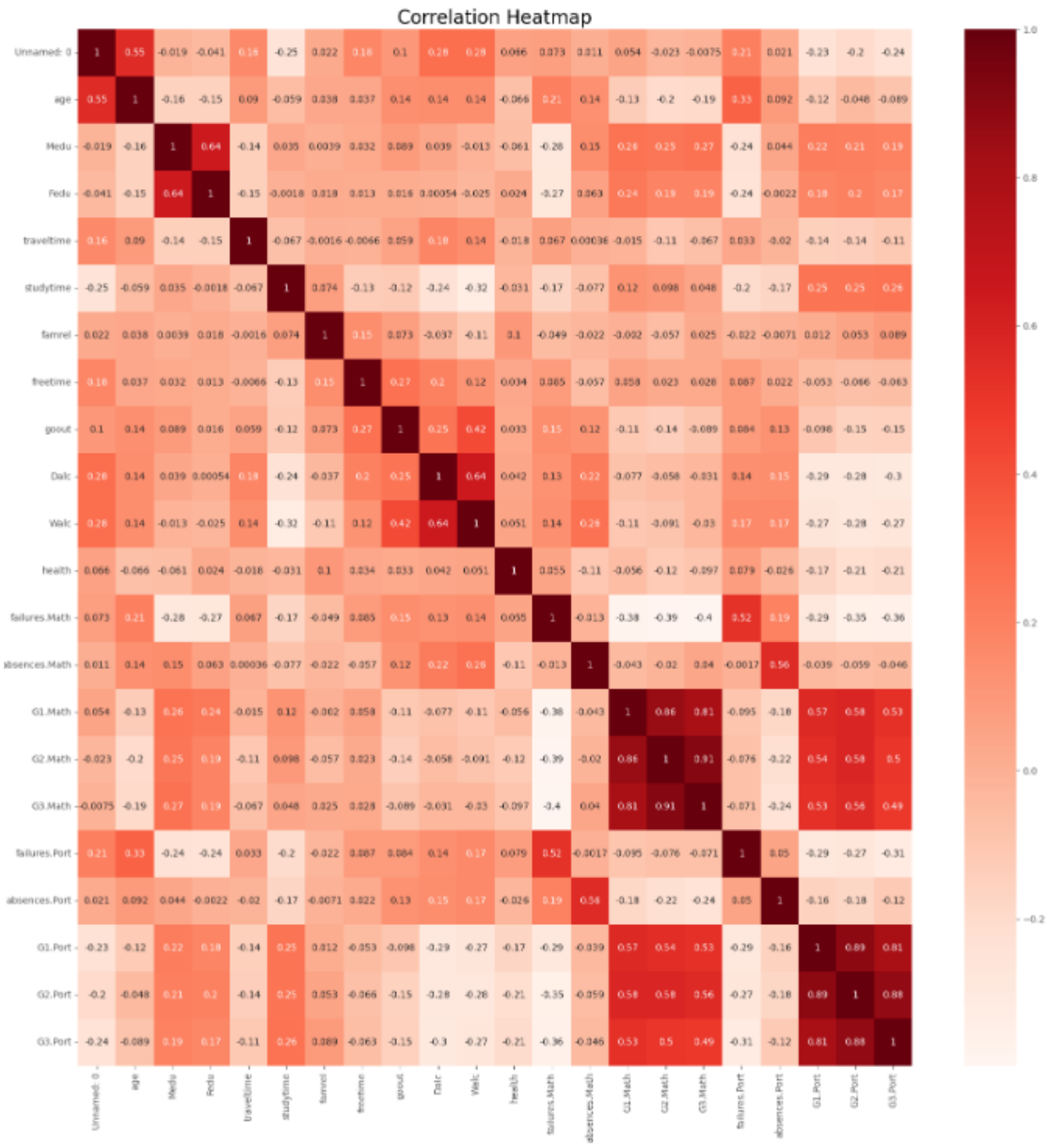
Exploring the relationship between the number of absences in the Portuguese course ('absences.Port') and the corresponding first period grades in Portuguese ('G1.Port'). Higher grades are associated with lower absences.



4.4 Correlation Analysis:

```
# Describing correlation
corr = dataset.corr()

# Creating a heatmap for correlation visualization
plt.figure(figsize=(20,20))
sns.heatmap(corr, annot=True, cmap="Reds")
plt.title('Correlation Heatmap', fontsize=20)
plt.show()
```

The correlation heatmap illustrates the relationships between different features in the dataset.



4.5 Feature Selection:

```
# Displaying the modified dataset after column removal
dataset.head()
```

After removing unnecessary columns ('Unnamed: 0', 'reason', 'Fedu', 'Mjob'), the modified dataset is displayed, highlighting the refined feature set.

4.6 Five-Level Classification:

```
def five_level_classification():
    bins = pd.IntervalIndex.from_tuples(
        [(0, 9.5), (9.5, 11.5), (11.5, 13.5), (13.5, 15.5), (15.5, 20)], closed='right')

    levels = ['fail', 'sufficient', 'satisfactory', 'good', 'excellent']

    new_column = 'G1.Port_evaluation'
    dataset[new_column] = np.array(levels)[
        pd.cut(dataset['G1.Port'], bins=bins).cat.codes]

five_level_classification()
```

The creation of a new categorical variable ('G1.Port_evaluation') with five levels provides a broader perspective on student performance, facilitating a more nuanced analysis.

4.7 Classification Model Evaluation:

```
# Displaying the dataset after adding the new classification variable
dataset.head()

# Separating features (X) and target variable (y) for classification
y_classification = pd.DataFrame(dataset['G1.Port_evaluation'].values)
X_train, X_test, y_train, y_test = train_test_split(X, y_classification, test_size=0.2, random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)

# Evaluate the model performance
accuracy = accuracy_score(y_test, predictions)
classification_report_str = classification_report(y_test, predictions)

# Print the results
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report_str)
```

The addition of the new classification variable is confirmed, and a logistic regression model is trained and evaluated for predicting the five-level classification. The classification report provides detailed metrics on the model's performance.

The Data Exploration section provides a deeper understanding of the dataset, focusing on the distribution of grades, inter-feature correlations, and the creation and evaluation of a classification model. The insights gained from this exploration will pave the way for more sophisticated analyses in the subsequent sections.

5. Data Analysis

5.1 Linear Regression for G1.Port Prediction

The initial application of linear regression yielded unsatisfactory results, with a mean squared error (MSE) of 0.798, indicating suboptimal predictive accuracy for initial Portuguese grades ('G1.Port'). Recognizing the need for improvement, feature selection techniques were employed to enhance model performance.

```
# The linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Making predictions on the test set
predictions = model.predict(X_test)

# Calculating mean squared error
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')
```

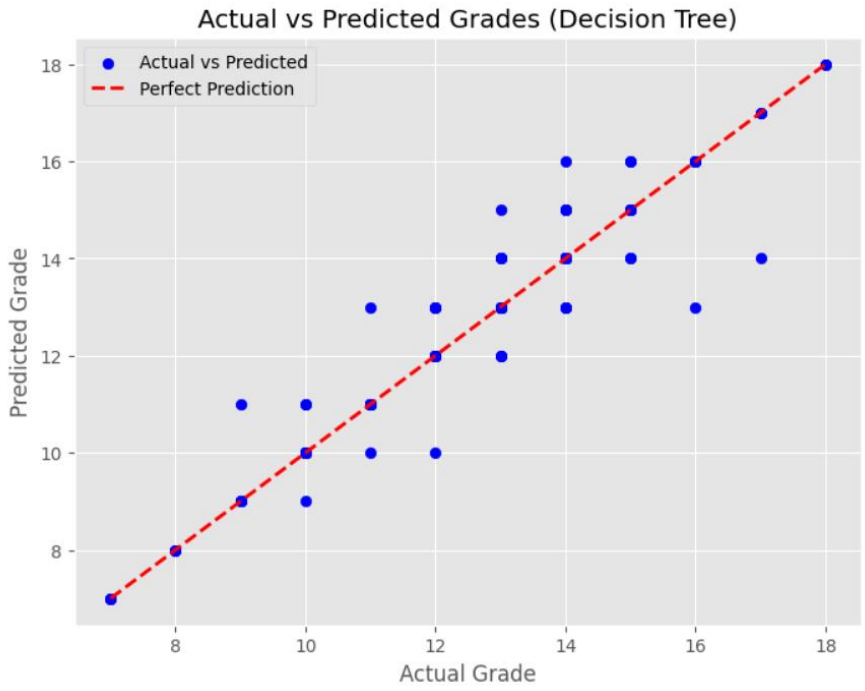
```
Mean Squared Error: 0.7986150173982162
```

5.2 Decision Tree Classification with Feature Selection

Subsequent utilization of a Decision Tree classification model, incorporating features selected through correlation analysis, resulted in a substantial enhancement in predictive accuracy. The accuracy achieved was 77.48%, significantly outperforming the linear regression model.

```python
#Decision Tree model is instantiated. This model will be trained on the training data to learn patterns and make predictions.

dt_model = DecisionTreeClassifier()

#The fit method is used to train the Decision Tree model on the provided training data (X_train and y_train).
#This is where the model learns to make decisions based on the features in the training data.

dt_model.fit(X_train,y_train)

#The predict method is used to make predictions on the test data (X_test).
#The model uses the learned patterns to predict the labels for the test set.

predictdt_y = dt_model.predict(X_test)

#The score method is used to evaluate the accuracy of the model on the test data.
#It compares the predicted labels (predictdt_y) with the true labels (y_test) and computes the accuracy.

accuracy_dt = dt_model.score(X_test,y_test)

#Finally,we print the accuracy of the Decision Tree model on the test data.

print("Decision Tree accuracy is :",accuracy_dt)
```

```
Decision Tree accuracy is : 0.7748344370860927
```



5.3 Random Forest Classification

Parallelly, a Random Forest classification model was implemented, achieving an impressive accuracy of 84.77% without feature selection. The Random Forest demonstrated the effectiveness of ensemble methods in handling complex relationships within the dataset.

```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(n_estimators=500 , oob_score = True, n_jobs = -1,
                                  random_state =50, max_features = "auto",
                                  max_leaf_nodes = 30)
model_rf.fit(X_train, y_train)

#The score method is used to evaluate the accuracy of the model on the test data.
#It compares the predicted labels (predictdt_y) with the true labels (y_test) and computes the accuracy.

accuracy_dt = model_rf.score(X_test,y_test)

#Finally,we print the accuracy of the Random Forest model on the test data.

print("Random Forest accuracy is :",accuracy_dt)
```

```
Random Forest accuracy is : 0.847682119205298
```

5.4 Logistic Regression Classification with Feature Selection

In an effort to refine the model further, feature selection was applied to the Logistic Regression classification model. Despite a slight decrease in accuracy compared to the Random Forest (83%), the Logistic Regression model exhibited performance similar to the ensemble method.

```
# Initialize and train the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)

# Evaluate the model performance
accuracy = accuracy_score(y_test, predictions)
classification_report_str = classification_report(y_test, predictions)

# Print the results
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report_str)
```

```
Accuracy: 0.83
Classification Report:
              precision    recall  f1-score   support

           7       1.00      1.00      1.00         3
           8       1.00      0.67      0.80         3
           9       0.86      0.86      0.86         7
          10       0.95      0.90      0.93        21
          11       0.89      0.94      0.92        18
          12       0.64      0.76      0.70        21
          13       0.76      0.67      0.71        24
          14       0.86      0.93      0.89        27
          15       0.82      0.69      0.75        13
          16       0.90      1.00      0.95         9
          17       0.67      0.67      0.67         3
          18       1.00      0.50      0.67         2

    accuracy                           0.83       151
   macro avg       0.86      0.80      0.82       151
weighted avg       0.83      0.83      0.83       151
```

6. Results and Conclusions

The linear regression model's limited predictive capabilities prompted the exploration of alternative approaches. The Decision Tree and Random Forest models showcased considerable improvements, demonstrating the importance of feature selection and ensemble methods in capturing nuanced relationships within the data.

Decision Tree:

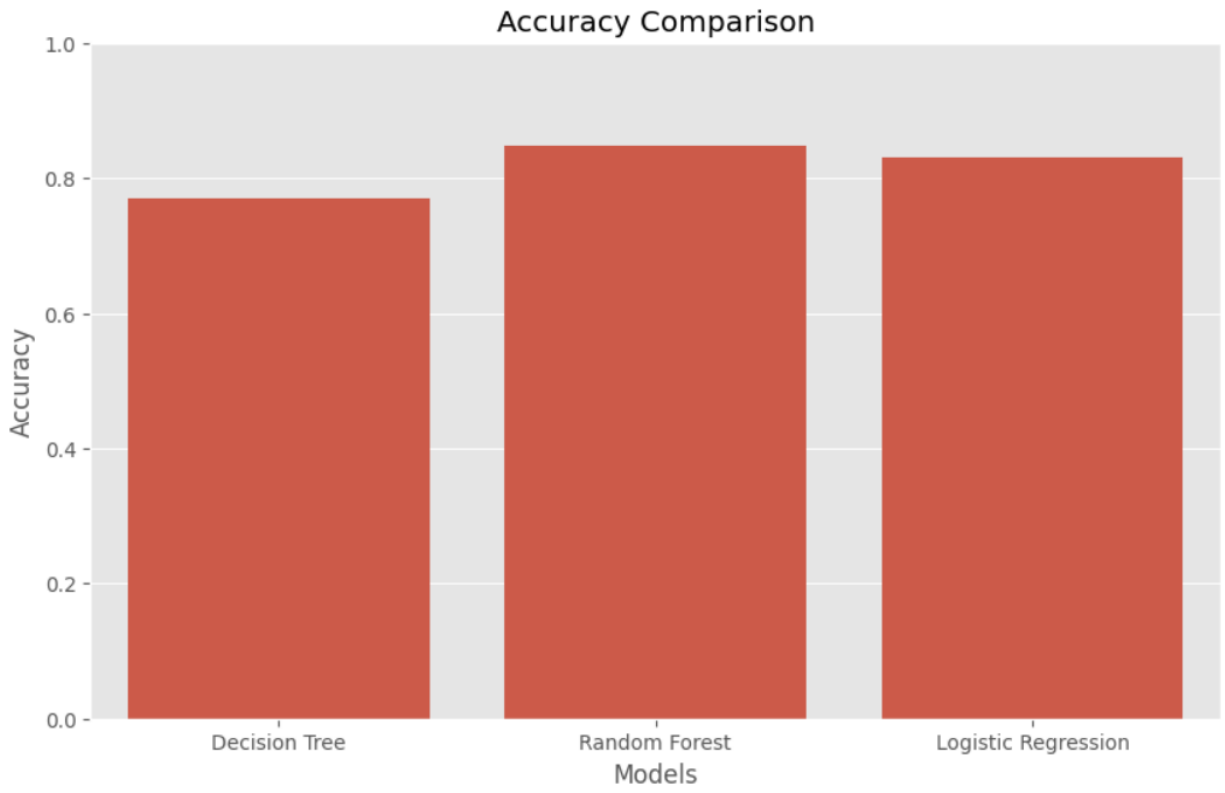
- Achieved a noteworthy accuracy of 77.48% post feature selection.
- Offers interpretability and captures non-linear patterns.

Random Forest:

- Initially demonstrated outstanding accuracy (84.77%).
- Served as a robust tool for student classification without the need for explicit feature selection.

Logistic Regression:

- Maintained competitive accuracy (83%) post feature selection.
- Provides a simpler and interpretable model compared to ensemble methods.



7. Implications and Recommendations

Feature Selection Significance:

- Feature selection played a crucial role in enhancing model accuracy, particularly evident in the Decision Tree model.

Ensemble Method Dominance:

- The Random Forest model showcased superior accuracy, emphasizing the strength of ensemble methods in capturing intricate relationships.

Logistic Regression Utility:

- Despite a slightly lower accuracy, Logistic Regression maintains competitive performance. Consider its application in scenarios where model interpretability is paramount.

In summary, the project demonstrated the limitations of linear regression in predicting student grades, motivating the exploration of more advanced models. The Decision Tree and Random Forest models, coupled with feature selection, provided substantial improvements, emphasizing the significance of model selection and feature engineering in educational data analysis. The results affirm the versatility of machine learning techniques in unraveling complex patterns within academic datasets.