UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga
(Final seminar paper)

Napovedovanje uspeha srednješolskih dijakov s pomočjo strojnega učenja

(Predicting Secondary School Student Performance Using Machine Learning)

Ime in priimek: Lucas Lorenzo Jakin
Študijski program: Računalništvo in informatika
Mentor: izr. prof. dr. Branko Kavšek

Koper, 2024

Ključna dokumentacijska informacija

Ime in PRIIMEK: Lucas Lorenzo JAKIN

Naslov zaključne naloge: **Napovedovanje uspeha srednješolskih dijakov s pomočjo strojnega učenja**

Kraj: Koper

Leto: 2024

Število listov: 42 Število slik: 24 Število tabel: 3

Število referenc: 17

Mentor: izr. prof. dr. Branko Kavšek

Ključne besede: strojno učenje, podatkovno rudarjenje, CRISP-DM, učni uspeh, napovedovanje

Izvleček:
Uporaba strojnega učenja v izobraževalnih okoljih je pridobila veliko pozornosti, zlasti pri napovedovanju modelov, ki podjetjem omogočajo, da na podlagi zgodovinskih podatkov natančno napovedo najverjetnejši odgovor na zastavljeno vprašanje. Namen mojega diplomskega dela je analizirati nabor podatkov iz spletne strni Kaggle, ki se nanaša na dijake srednjih šol na Portugalskem. Sledil bom metodologiji CRISP-DM (Croos Industry Standard Process for Data Mining), kjer bom sistematično raziskal vsako fazo, od razumevanja in priprave podatkov do modeliranja in vrednotenja. V fazi modeliranja bom primerjal različne algoritme strojnega učenja, vsakič z uporabo različnih parametrov, da bi napovedal akademski uspeh dijakov. Nato bom preizkusil in testiral učinkovitost modelov, da bi določil najučinkovitejši pristop.

Key words documentation

Name and SURNAME: Lucas Lorenzo JAKIN

Title of final project paper: **Predicting Secondary School Student Performance Using Machine Learning**

Place: Koper

Year: 2024

Number of pages: 42 Number of figures: 24 Number of tables: 3

Number of references: 17

Mentor: Assoc. Prof. Branko Kavšek, PhD

Keywords: Machine Learning, Data Mining, CRISP-DM, Academic Performance, Prediction

Abstract: The application of machine learning in educational settings has gathered significant attention, particularly in model predictions allowing businesses to make accurate guesses to the likely outcomes of a question based on historical data. My diploma thesis aims to analyze a dataset from Kaggle related to secondary school students in Portugal. By following the CRISP-DM methodology (Cross Industry Standard Process for Data Mining), I will systematically explore each phase, from data understanding and preparation to modeling and evaluation. In the modeling phase I will implement various ML (Machine Learning) algorithms, using different parameters every time, to predict students' academic success. The performance of these models will be then compared and tested to determine the most effective approach.

Zahvala

I want to thank everyone who helped me during this thesis journey. I am especially grateful to my thesis mentor, Dr. Branko Kavšek. Their guidance were key in shaping this work, and their commitment motivated me throughout the project.

I also want to thank my family for their constant emotional support. Your belief in me and feedback were important in helping me overcome the challenges I faced in the past years.

Contents

1 INTRODUCTION 1

2 PROBLEM DESCRIPTION 5

3 DATA UNDERSTANDING 7

3.1 DESCRIPTION OF THE DATASET . . . 7
3.2 ATTRIBUTES . . . 7
3.2.1 Categorical Attributes . . . 10
3.2.2 Numerical Attributes . . . 15
3.3 EXPLORATORY DATA ANALYSIS . . . 18
3.3.1 HeatMap - correlations . . . 20

4 DATA PREPARATION 22

4.1 MISSING VALUES . . . 22
4.2 OUTLIERS . . . 22
4.3 LABEL ENCODING . . . 23

5 MODELING 24

5.1 CLASSIFICATION . . . 24
5.1.1 ZeroR Classifier (0R) . . . 24
5.1.2 OneR Classifier (1R) . . . 25
5.1.3 Random Forest Classifier . . . 25
5.1.4 Support Vector Machine (SVM) . . . 25
5.2 REGRESSION . . . 26
5.2.1 Decision Tree Regressor . . . 26
5.2.2 Random Forest Regressor . . . 27
5.2.3 K-Nearest Neighbors (kNN) . . . 27
5.3 METHODOLOGY . . . 28
5.3.1 ZeroR . . . 28
5.3.2 OneR . . . 29
5.3.3 Random Forest . . . 30
5.3.4 kNN . . . 31

5.3.5 SVM . . . 33
5.3.6 Decision Tree Regressor . . . 34
5.3.7 Random Forest Regressor . . . 35
5.3.8 KNN Regressor . . . 35

6 EVALUATION AND RESULTS 36
6.1 EVALUATION METHODOLOGY . . . 36
6.1.1 Evaluating Classification Models . . . 36
6.1.2 Evaluating Regression Models . . . 38
6.1.3 Cross-Validation . . . 39
6.2 RESULTS . . . 40

7 CONCLUSION AND FUTURE WORK 42

8 BIBLIOGRAPHY 43

List of Tables

1 Attributes and their descriptions 9

2 Classification models' results 40

3 Regression models' results 40

List of Figures

1 CRISPM-DM methodology [15] 4

2 Head of the dataset 8
3 Data summary 10
4 Categorical attributes 11
5 Categorical attributes distribution 14
6 Numerical attributes 15
7 Numeric attributes distribution 17
8 Correlation G1 & G2 18
9 Correlation G2 & G3 19
10 Correlation G1 & G3 19
11 Heatmap - Correlation Matrix 20

12 IQR - Interquantile Range Method 22

13 Support Vector Machine [10] 26
14 Random Forest & Decision Trees 27
15 K-Nearest Neighbors [3] 28
16 Frequency table 28
17 Target distribution 28
18 Random Forest Classification report 31
19 Random Forest Confusion matrix 31
20 KNN Classification report 32
21 KNN Confusion matrix 33
22 SVM Classification report 34
23 SVM Confusion matrix 34

24 Croos-Validation technique [11] 39

Abbreviations

ML	Machine Learning
kNN	k-Nearest Neighbours
SVM	Support Vector Machines
DM	Data Mining
CV	Cross-Validation
CRISP-DM	Cross Industry Standard Process for Data Mining

...

1 INTRODUCTION

Machine Learning (ML) has become a revolutionary technology in recent years, transforming industries and changing the way data-based decisions are conducted. Machine Learning is primarily about creating algorithms that help computers learn from data and make predictions. This ability has enabled companies to discover insights, improve processes, and enhance decision-making in different areas, such as finance, healthcare, marketing, and manufacturing.

The CRISP-DM (Cross-Industry Standard Process for Data Mining) [6] methodology is crucial for the successful implementation of machine learning models. The structured framework provided by CRISP-DM helps guide data scientists and analysts in the complex process of developing data mining and machine learning models. This approach guarantees that projects are carried out in a structured manner, resulting in dependable and understandable results. Published in 1999 to standardize data mining processes across industries, it has since become the **most common methodology** for data mining, analytics, and data science projects.

The CRISP-DM methodology [6] outlines six critical phases in a data mining project:

- **Project/Problem Understanding:** This initial phase focuses on understanding the objectives and requirements of the project. Establishing a strong business understanding is absolutely essential. Most of the tasks performed in this phase are foundational project management activities that are universal to most projects:

  1. **Determine business objectives:** It is crucial to first understand, from a business perspective, what the customer really wants to accomplish.
  2. **Asses situation:** Determine resources availability, project requirements, assess risks and contigencies, and conduct a cost-benefit analysis
  3. **Determine data mining goals:** Moreover, in addition to defining bussiness objectives, it is also important to define what success looks like from a technical data mining perspective.

   4. **Produce project plan:** Select technologies and tools and define detailed plans for each project phase.

- **Data Understanding:** In this phase, data is collected and explored to gain insights into its properties. It drives the focus to identify, collect, and analyze the data sets that can help you accomplish the project goals. This phase has four tasks:

   1. **Collect initial data:** Acquire the necessary data and load it into the analysis tool.
   2. **Describe data:** Examine the data and document its surface properties like data format, number of fields or field identities.
   3. **Explore data:** Query the data, visualize it, and identify relationships among the data.
   4. **Verify data quality:** Document any quality issues. How clean is the data?

- **Data Preparation:** Almost 80% of the project is data preparation.
  This phase, which is often referred to as "data munging", prepares the final data set for modeling. It has five tasks:

   1. **Select data:** Determine which data sets will be used and document reasons for inclusion/exlusion.
   2. **Clean data:** A common practice during this task is to correct, impute, or remove erreneous values. Without it, the quality of the data be innacurate.
   3. **Construct data:** Derive new attributes that will be helpful.
   4. **Integrate data:** Create new data sets by combining data from multiple sources.
   5. **Format data:** Re-format data as necessary, for example converting string values that store numbers to numeric for better mathematical operations.

- **Modeling** Even if this might be the most exciting work is also often the shortest phase of the project. Here various models will likely be built and assessed based on several different modeling techniques. This phase has four tasks:

   1. **Select modeling techniques:** Determine which algoritms to try (e.g. regression, neural network, kNN).
   2. **Generate test design:** Depending on the approach you took, there might be the need to split the data into training set, test, and validation sets.

3. **Build model:** The selected modeling techniques are put into action to create predictive models. Beside running lines of code, this process requires careful consideration of various factors to ensure the models are robust and effective.
4. **Assess model:** This part is critical for determining the effectiveness and reliability of predictive models built in the previous step. It involves a detailed evaluation of the model's performance using various metrics and techniques to ensure it meets the predefined success criteria ans performs well on unseen data.

- **Evaluation** Whereas the Assess Model task of the modeling phase focuses on technical model assessment, the Evaluation phase looks more broadly at which model best meets the business and what to do next. This phase has three tasks:

  1. **Evaluate results:** Checking if the models meet the business criteria.
  2. **Review process:** Review the work accomplished. Summarize findings and correct anything if needed.
  3. **Determine next steps:** Based on the previous three tasks, determine whether to proceed to deployment, iterate further, or initiate new projects.

- **Deployment** The Deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process across the enterprise.

  1. **Plan deployment:** Develop and document a plan for deploying the model.
  2. **Plan monitoring and maintenance:** Develop a thorough monitoring and maintenance plan to avoid issues during the operational phase of a model.
  3. **Produce final report:** Document a summary of the project which might include a final presentation of data mining results.
  4. **Review project:** Conduct a project retrospective about what went well, what could have been better, and how to improve in the future.

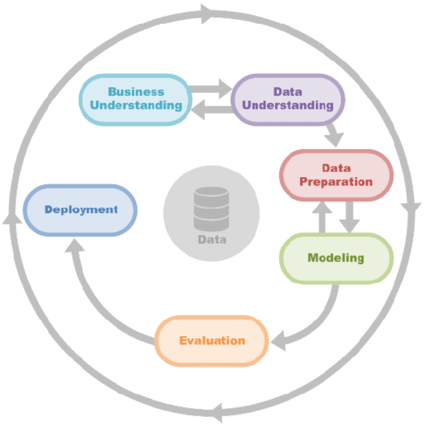


Figure 1: CRISPM-DM methodology [15]

For this diploma thesis, a dataset will be selected to demonstrate the application of the CRISPM-DM methodology. The focus of the thesis will be to systematically apply each phase of CRISPM-DM to the chosen dataset. This will involve selecting an appropriate dataset, understanding its structure and contents, preparing it for analysis, and implementing various machine learning models to predict target values. The models will be then evaluated to determine their accuracy and effectiveness. This diploma thesis aims to provide a thorough and structured approach to data mining and predictive modeling, showcasing the practical application of these techniques on real-world data.

The dataset that I have chosen is about student performance and it is available on both Kaggle [https://www.kaggle.com] and UCI Machine Learning Repository [https://archive.ics.uci.edu]. The dataset approaches student achievement in secondary education in two Portuguese schools [https://www.kaggle.com/datasets/devansodariya/student-performance-data].

2 PROBLEM DESCRIPTION

Education is a key factor for achieving long-term economic progress. During the last two decades, the portuguese educational level has improved. However, the statistics keep Portugal at Europe's tail end due to its high student failure and dropping rates. In particular, failure in the core classes of Mathematics and Portuguese (native language) is extremely serious, since they provide fundamental knowledge for the success in other remaining subjects.
The chosen data set approach student achievements in secondary education of two Portuguese schools. In Portugal, the secondary education consists of 3 years of schooling, preceding 9 years of basic education and followed by higher education. There are several courses that share core subjects as the Portuguese language and Mathematics. It includes a wide range of attributes that capture various aspects of student life and academic achievement, consisting of student grades, demographic information, social factors, and school-related features. The data was collected through school reports and questionnaires, ensuring a detailed and multifaceted view of the factors influencing student performance.

Objectives: The main objective of this thesis is to build and evaluate predictive models that can accurately forecast the final grades of students in Mathematics. The problem will be approached from two angles:

- **Classification:** Categorizing students' final grades into predefined levels or classes on a scale from A to Z.
- **Regression:** Predicting the exact numerical value of a student's final grade based on the input features.

Classification and regression are two important DM (Data Mining) goals. Both require a supervised learning, where a model is adjusted to a dataset made up of K examples, each mapping an input vector $(h_1^k, ..., x_I^k)$ to a given target y_k. The main difference is set in terms of the output representation, discrete for classification and continuous for regression.
The evaluation phase has the importance and the objective to assess the performance and reliability of the predictive models. The goal is to determine the models's accuracy precision and robustness by employing a range of performance metrics such as

accuracy, Mean Squared Error (MSE), F1 Score, etc. Additionally, the evaluation will employ validation techniques such as 10-fold Cross-Validation and random TRAIN-TEST splits to ensure that the models generalize well to unseen data and are not overfitting.

Predicting student performance is of great importance to educators and students themselves. Accurate predictions can help in identifying students at risk of underperforming, allowing for timely interventions and support. By sistematically performing a thorough analysis of student performance data and develop reliable models we can achieve the above mentioned objectives.

The dataset is complemented by an article written by Paolo Cortez and Alice Silva [1], which provides an in-depth explanation of the problem and describes how data scientists approached the analysis and modeling of the data. Their work serves as a benchmark for this diploma thesis.

3 DATA UNDERSTANDING

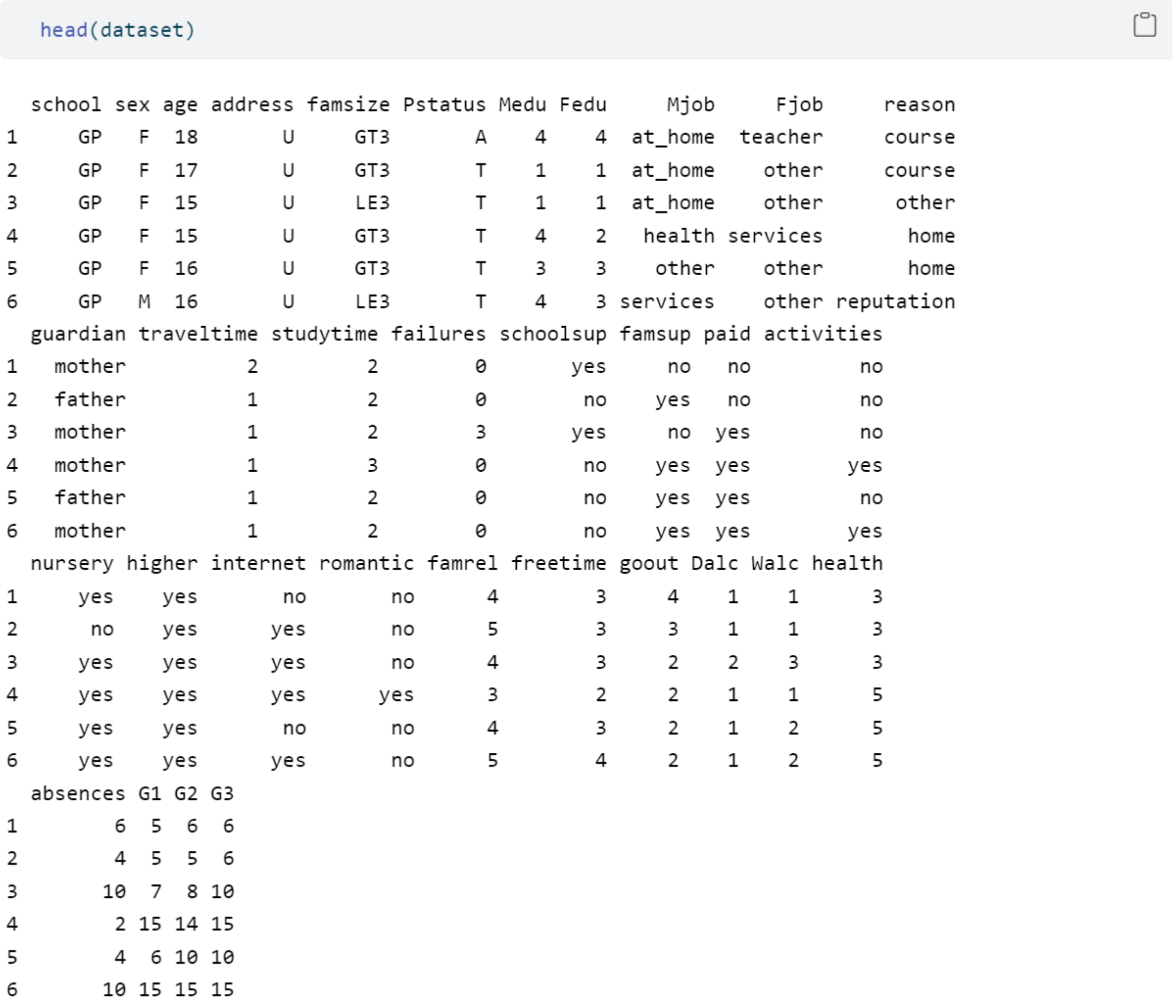
As mentioned before the data can be found in both Kaggle and UCI ML repositories. The UCI ML Repository provides two separate datasets: one focusing on student performance in Portuguese language and another on Mathematics. In contrast the Kaggle version of the dataset contains only the data for Mathematics. Despite the difference, the datasets are fundamentally the same in terms of structure and information. For the purpose of the thesis, the Mathematics dataset has been selected. This choice allows for a focused analysis on a single subject.

3.1 DESCRIPTION OF THE DATASET

The initial version of the dataframe contained scarce information about the students, containing only the grades and the number of absences, it was then complemented with several other information such as demographic (e.g. mother's education, family income), social (e.g. alcohol consumption) and school related attributes (e.g. number of past class failures). The final dataframe consists of 33 columns (attributes) and has 395 rows (instances). Some features were discarded due to the lack of discriminative value, for instance few people answered about their family income probably due to privacy. The dataset contains three very similar attributes, which are G1, G2 and G3 that represent the grade in the first period, second period and final grade, respectively. The target attribute is G3 and is the one that I am trying to predict. G3 (also G1 and G2) contains values that go from 0 to 20, since Portugal has a 20-point grading scale, where 0 is the lowest grade and 20 is the perfect score. The target attribute G3 has a string correlation with attributes G2 and G1. This occurs because G3 is the final grade and it's value depend on the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful.

3.2 ATTRIBUTES

The dataset looks like the following:



```
head(dataset)
```

```
  school sex age address famsize Pstatus Medu Fedu     Mjob     Fjob     reason
1     GP   F  18       U     GT3       A    4    4  at_home  teacher     course
2     GP   F  17       U     GT3       T    1    1  at_home    other     course
3     GP   F  15       U     LE3       T    1    1  at_home    other      other
4     GP   F  15       U     GT3       T    4    2   health services       home
5     GP   F  16       U     GT3       T    3    3    other    other       home
6     GP   M  16       U     LE3       T    4    3 services    other reputation
  guardian traveltime studytime failures schoolsup famsup paid activities
1   mother          2         2        0       yes     no   no         no
2   father          1         2        0        no    yes   no         no
3   mother          1         2        3       yes     no  yes         no
4   mother          1         3        0        no    yes  yes        yes
5   father          1         2        0        no    yes  yes         no
6   mother          1         2        0        no    yes  yes        yes
  nursery higher internet romantic famrel freetime goout Dalc Walc health
1     yes    yes       no       no      4        3     4    1    1      3
2      no    yes      yes       no      5        3     3    1    1      3
3     yes    yes      yes       no      4        3     2    2    3      3
4     yes    yes      yes      yes      3        2     2    1    1      5
5     yes    yes       no       no      4        3     2    1    2      5
6     yes    yes      yes       no      5        4     2    1    2      5
  absences G1 G2 G3
1        6  5  6  6
2        4  5  5  6
3       10  7  8 10
4        2 15 14 15
5        4  6 10 10
6       10 15 15 15
```

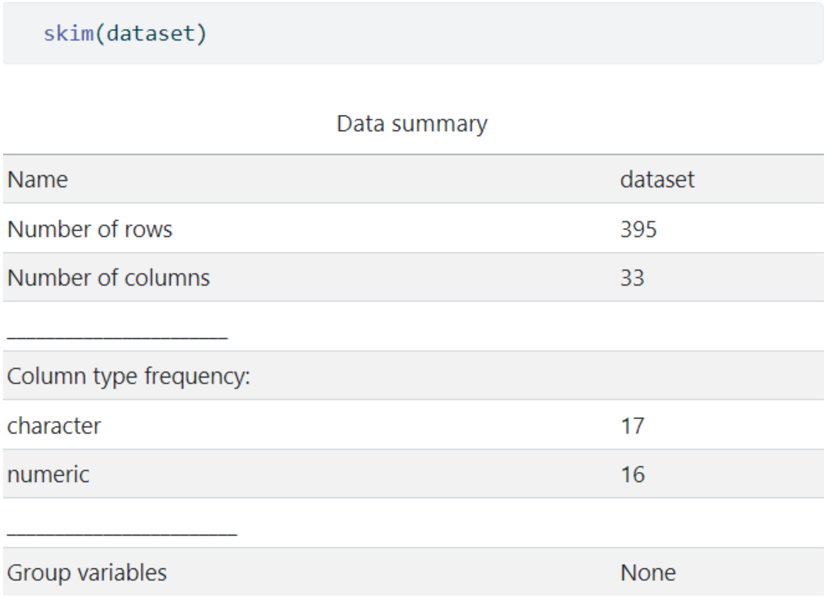
Figure 2: Head of the dataset

There are 33 different attributes, which is a lot so let's get to know them in the following table:

Table 1: Attributes and their descriptions

Attribute	Description
school	Student's school - BINARY(GP or MS)
sex	Student's sex - BINARY(F - M)
age	Student's age - NUMERIC(15-22)
address	Home address - BINARY(Urban or Rural)
famsize	Family size - BINARY (LE3 or GT3)
Pstatus	Parent's cohabitation status - BINARY(Together or Apart)
Medu	Mother's education - NUMERIC(0-4)
Fedu	Father's education - NUMERIC(0-4)
Mjob	Mother's job - NOMINAL
Fjob	Father's job - NOMINAL
reason	Reason to choose this school - NOMINAL
guardian	Student's guardian - NOMINAL(Mother or Father or Other)
traveltime	Home to school travel time - NUMERIC(1-4)
studytime	Weekly study time - NUMERIC(1-4)
failures	Number of past class failures - NUMERIC(n if 1 GE n LT 3, else 4)
schoolsup	Extra educational support - BINARY(y-n)
famsup	Family educational support - BINARY(y-n)
paid	Extra paid classes within the course subject - BINARY(y-n)
activities	Extra-curricular activities - BINARY(y-n)
nursery	Attended nursery school - BINARY(y-n)
higher	Wants to take higher education - BINARY(y-n)
internet	Internet access at home - BINARY(y-n)
romantic	With a romantic relationship - BINARY(y-n)
famrel	Quality of family relationships - NUMERIC(1-5)
freetime	Free time after school - NUMERIC(1-5)
goout	Going out with friends - NUMERIC(1-5)
Dalc	Workday alcohol consumption - NUMERIC(1-5)
Walc	Weekend alcohol consumption - NUMERIC(1-5)
health	Current health status - NUMERIC(1-5)
absences	number of school absences - NUMERIC(0-93)
G1	First period grade - NUMERIC(0-20)
G2	Second period grade - NUMERIC(0-20)
G3	Final grade - NUMERIC(0-20)

The attributes are divided into two distinct data types, which are **Numerical** and **Nominal**. This can be seen with some further analysis by using the skim() function, a method in the R programming language, that provides a neat summary of the dataset:



```
skim(dataset)
```

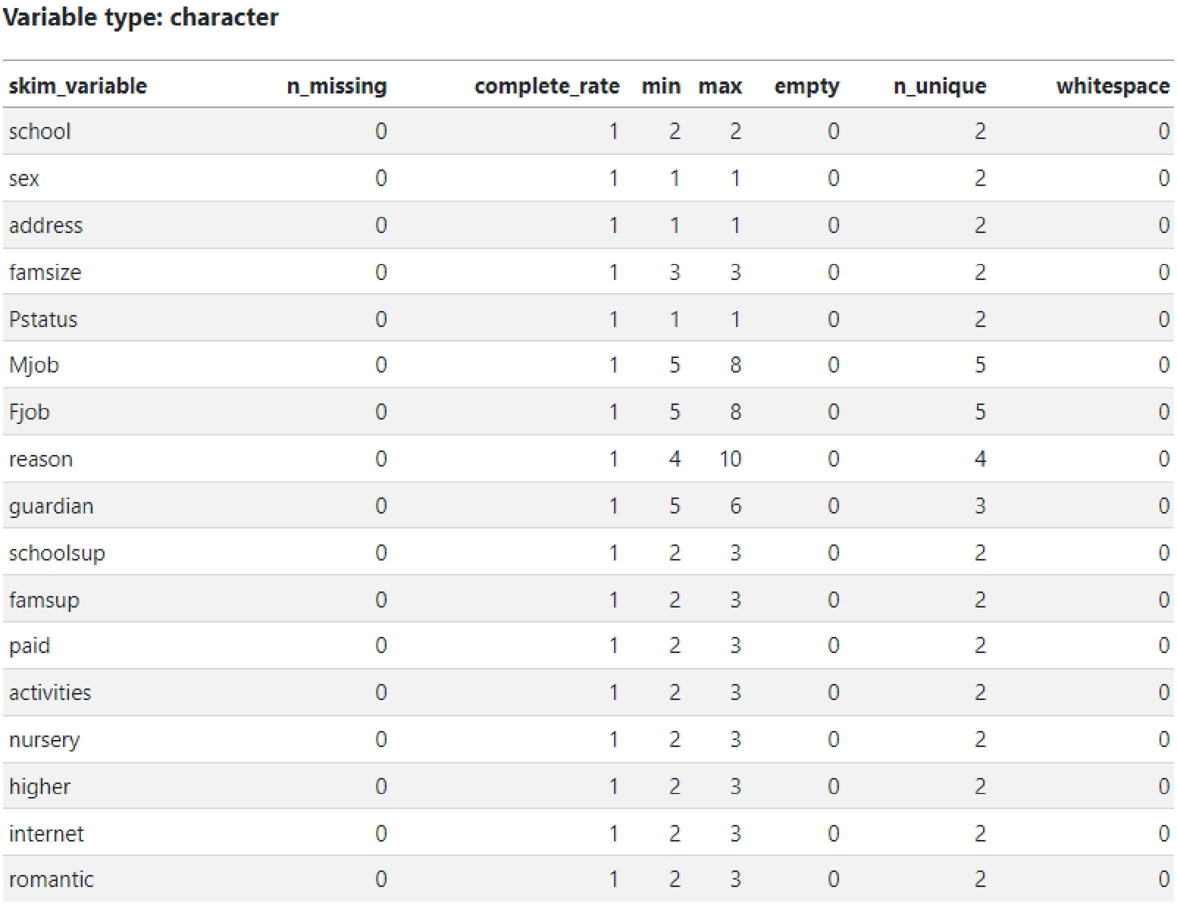
Data summary

Name	dataset
Number of rows	395
Number of columns	33
________________	
Column type frequency:	
character	17
numeric	16
________________	
Group variables	None

Figure 3: Data summary

3.2.1 Categorical Attributes

The dataset contains 17 categorical attributes. Most of them are binary, meaning they consist of only two possible values. The remaining categorical attributes have several values, indicating a broader range of possible categories for those features. These multi-valued attributes might include variables such as mother's job, father's job, or guardian that can occur in different forms. The *skim()* function is very helpful and can provide insightful characteristics about the data. In this case no categorical attribute contains missing values, which is highly advantageous for my analysis. Moreover, characteristics such as min and max value, number of unique values, are printed out.This can be seen in Figure 4.

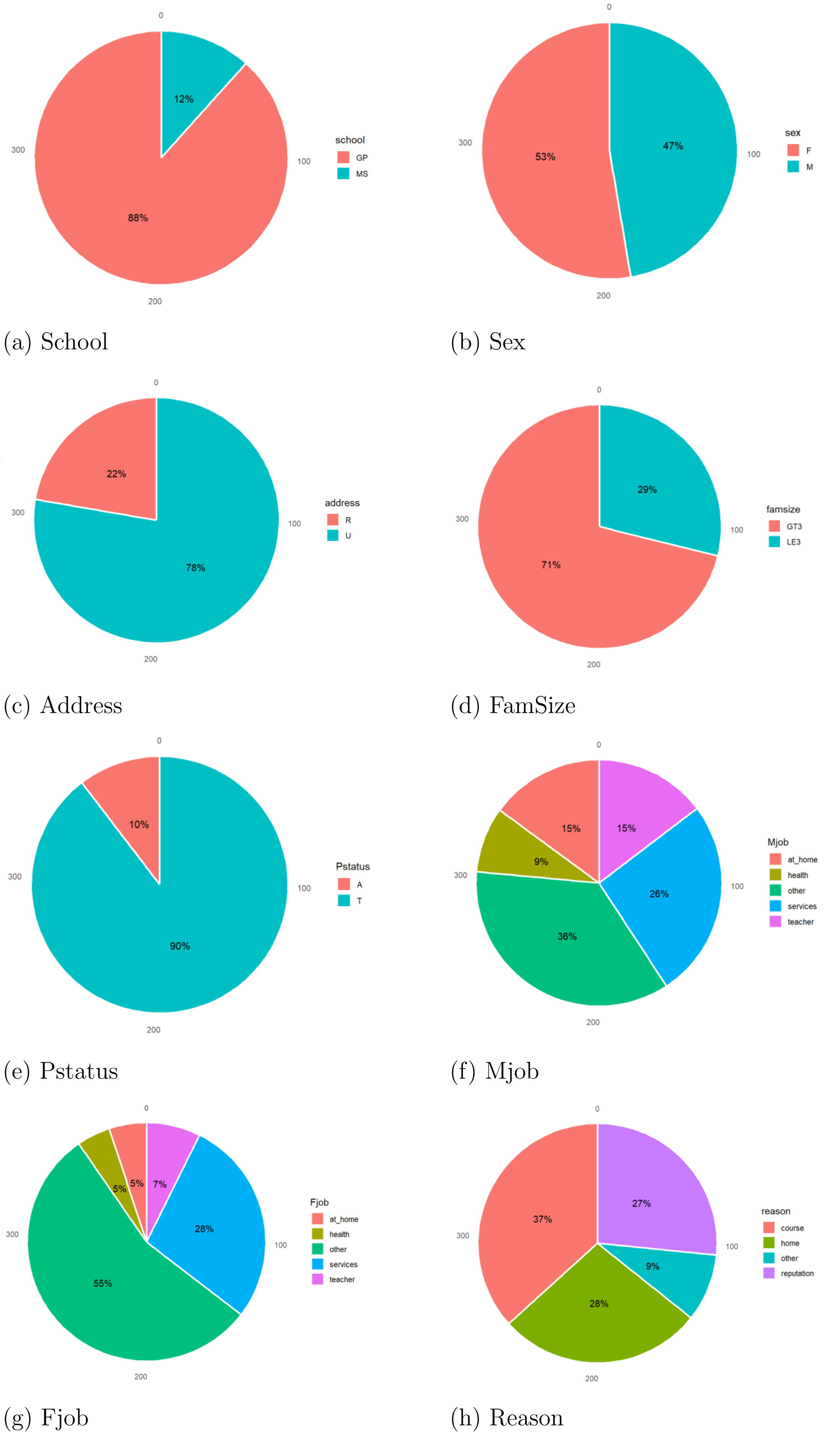


Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
school	0	1	2	2	0	2	0
sex	0	1	1	1	0	2	0
address	0	1	1	1	0	2	0
famsize	0	1	3	3	0	2	0
Pstatus	0	1	1	1	0	2	0
Mjob	0	1	5	8	0	5	0
Fjob	0	1	5	8	0	5	0
reason	0	1	4	10	0	4	0
guardian	0	1	5	6	0	3	0
schoolsup	0	1	2	3	0	2	0
famsup	0	1	2	3	0	2	0
paid	0	1	2	3	0	2	0
activities	0	1	2	3	0	2	0
nursery	0	1	2	3	0	2	0
higher	0	1	2	3	0	2	0
internet	0	1	2	3	0	2	0
romantic	0	1	2	3	0	2	0

Figure 4: Categorical attributes

In Figure 5 we can see the distribution of all categorical attributes using a series of pie charts, which were drawn using R programming language. Each pie chart visualizes the proportion of different categories for a specific attribute provifing characteristics of the data:



(a) School

(b) Sex

(c) Address

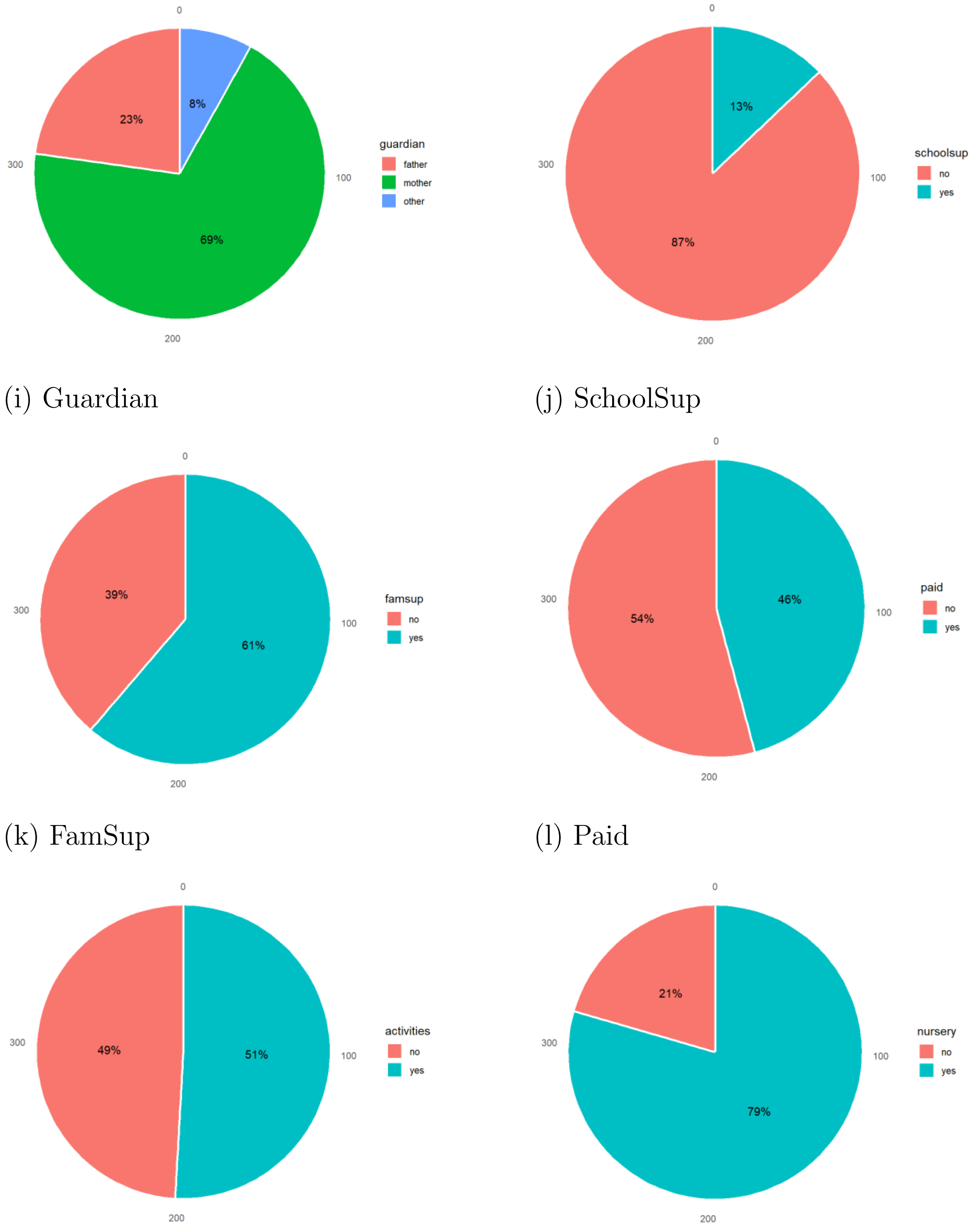
(d) FamSize

(e) Pstatus

(f) Mjob

(g) Fjob

(h) Reason



(i) Guardian

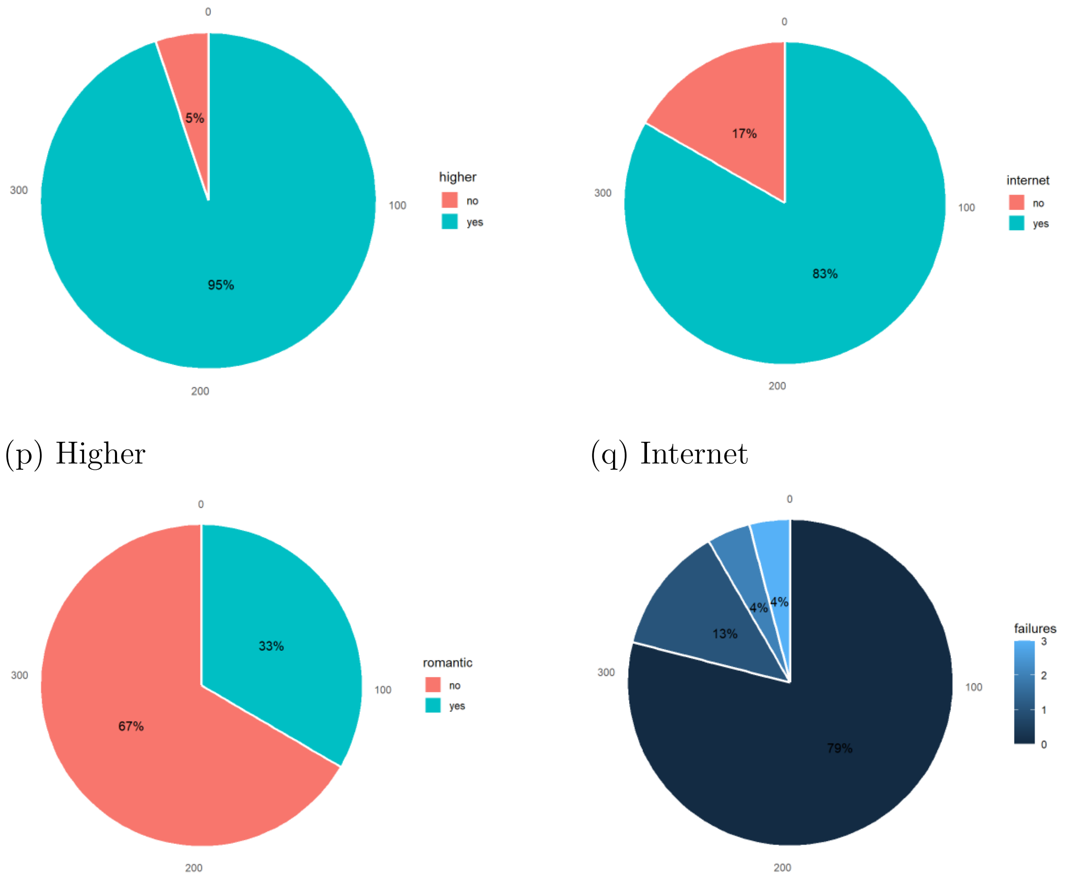
(j) SchoolSup

(k) FamSup

(l) Paid

(m) Activities

(n) Nursery



(p) Higher

(q) Internet

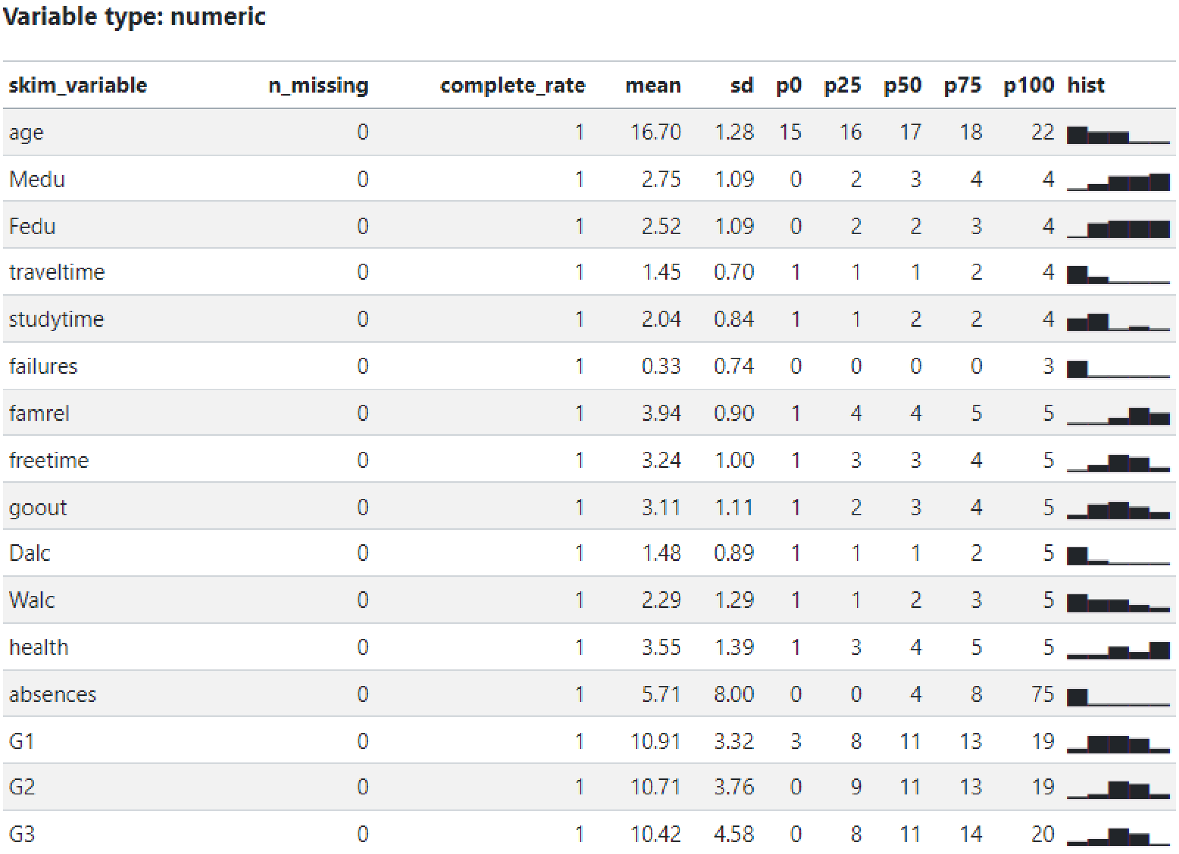
(p) Romantic

(q) Failures

Figure 5: Categorical attributes distribution

3.2.2 Numerical Attributes

The dataset comprises of 16 numerical attributes that also influence the academic outcomes, such as the number of past failures, mother's education, and father's education, among others. To gain a comprehensive understanding of these numerical attributes, after employing the skim() procedure it generated a detailed summary table. The table in Figure 6, similarly as the categorical values, provides essential statistical information about each numerical attribute. Also the numerical values do not contain any missing values, meaning that the whole dataset is complete. The table presents several key pieces such as the average value of each attribute, the standard deviation, the minimum and maximum value and **histograms** that give us a graphical representation of the distribution of each attribute, showing the frequency of different values ranges:



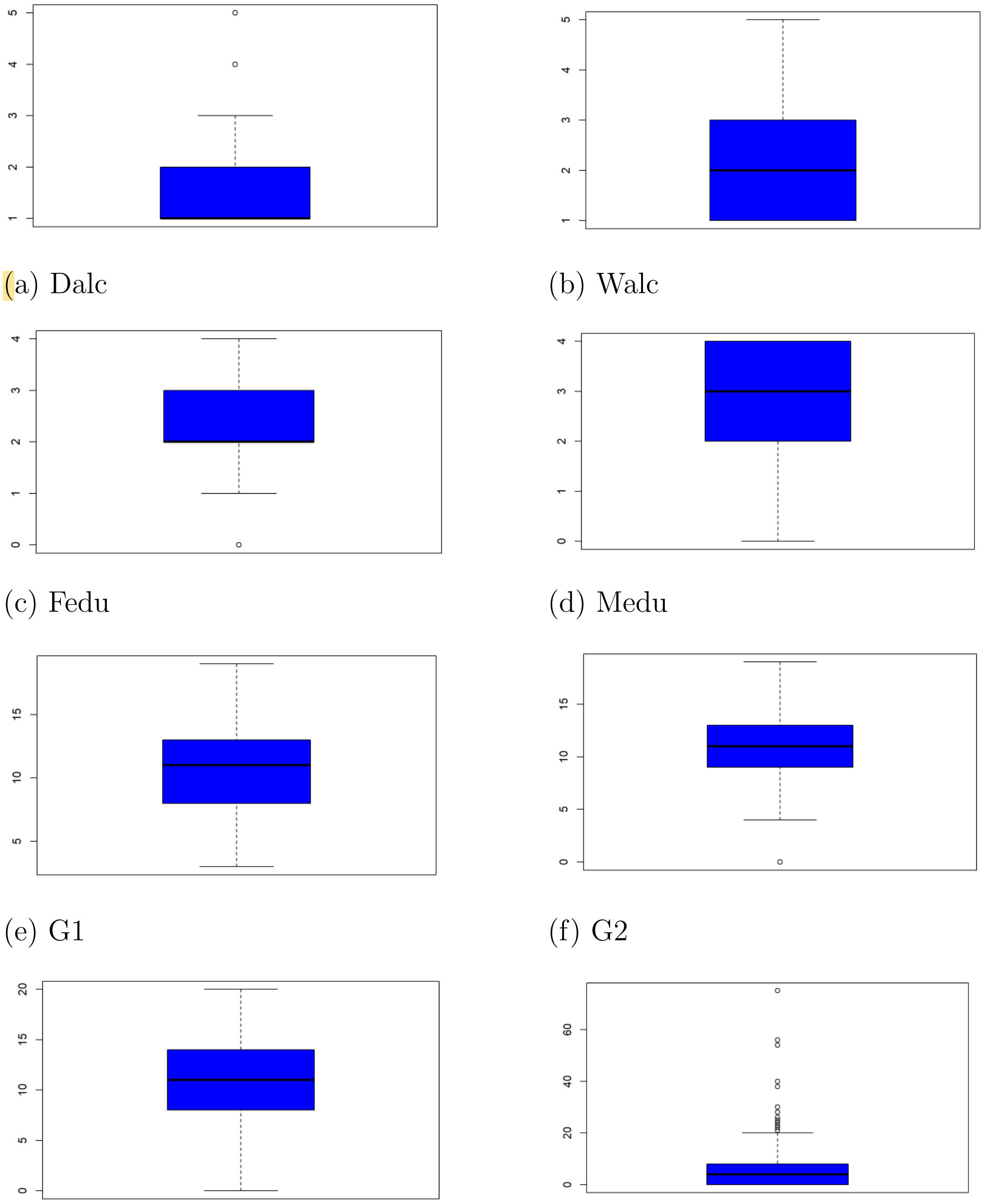
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	16.70	1.28	15	16	17	18	22	▇▅▅▁▁
Medu	0	1	2.75	1.09	0	2	3	4	4	▁▂▆▆▇
Fedu	0	1	2.52	1.09	0	2	2	3	4	▁▇▇▇▇
traveltime	0	1	1.45	0.70	1	1	1	2	4	▇▃▁▁▁
studytime	0	1	2.04	0.84	1	1	2	2	4	▅▇▁▂▁
failures	0	1	0.33	0.74	0	0	0	0	3	▇▁▁▁▁
famrel	0	1	3.94	0.90	1	4	4	5	5	▁▁▂▇▅
freetime	0	1	3.24	1.00	1	3	3	4	5	▁▃▇▆▂
goout	0	1	3.11	1.11	1	2	3	4	5	▂▆▇▅▃
Dalc	0	1	1.48	0.89	1	1	1	2	5	▇▂▁▁▁
Walc	0	1	2.29	1.29	1	1	2	3	5	▇▅▅▃▂
health	0	1	3.55	1.39	1	3	4	5	5	▂▂▅▃▇
absences	0	1	5.71	8.00	0	0	4	8	75	▇▁▁▁▁
G1	0	1	10.91	3.32	3	8	11	13	19	▂▇▇▆▂
G2	0	1	10.71	3.76	0	9	11	13	19	▁▂▇▆▂
G3	0	1	10.42	4.58	0	8	11	14	20	▂▃▇▅▁

Figure 6: Numerical attributes

Below you can find Figure 7 that represents a series of boxplots to visualize the distribution of numeric attributes within the dataset. Boxplots are advantageous for summarizing the central tendency of values. They provide a clear depiction of the median, quartiles, and potential outliers in each numeric attribute. The central box represents the interquantile range (IQR), which contains the middle 50% of the data. The external lines extend to the minimum and maximum values within 1.5 times the

IQR from the first and third quartile, respectively. Data points beyond these external lines are considered to be potential outliers:



(a) Dalc

(b) Walc

(c) Fedu

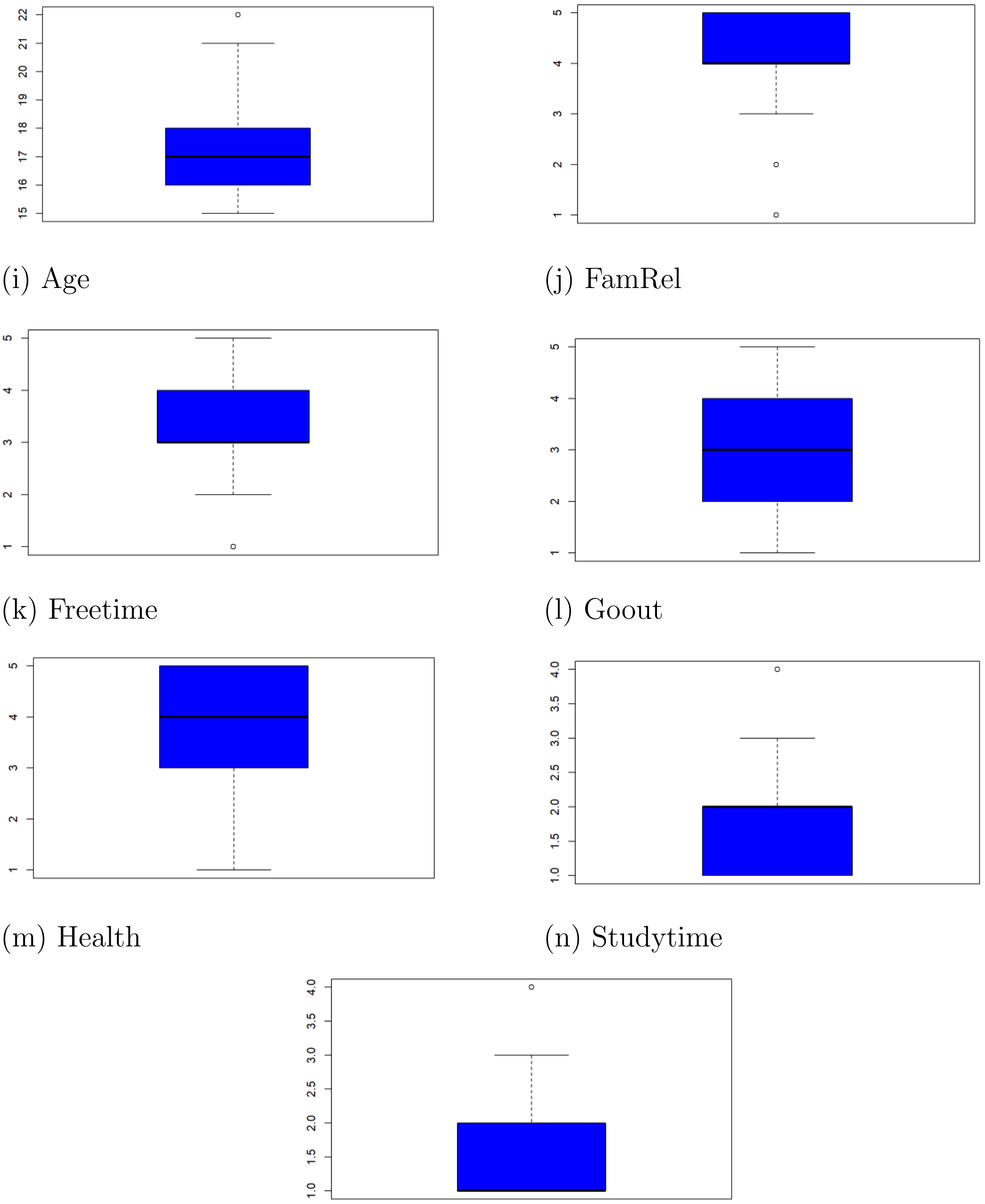
(d) Medu

(e) G1

(f) G2

(g) G3

(h) Absencies



(i) Age

(j) FamRel

(k) Freetime

(l) Goout

(m) Health

(n) Studytime

(r) Traveltime

Figure 7: Numeric attributes distribution

After taking a look at the boxplots, I encountered an “issue” with the Failures attribute. The boxplot contained only three distinct points, making it difficult to interpret. To better understand the distribution of values for this attribute, I decided to draw a pie chart instead. This type of chart provides a clearer visualization of the proportion of different values within “Failures”.

3.3 EXPLORATORY DATA ANALYSIS

Now let's take a look at key insights and patterns within the dataset. The purpose of this analysis is to understand the underlying patterns, distributions, and relationships within the data and to visualize it through scatter plots, histograms and other graphical methods. By providing a clear and comprehensive picture of the dataset, EDA lays the foundation for building effective and accurate predictive models.

Let's analyze the attributes G1, G2 and G3. As described in the dataset description, these three attributes are very similar to each other, since they represent the grades in the first period, second period and final grade, respectively.

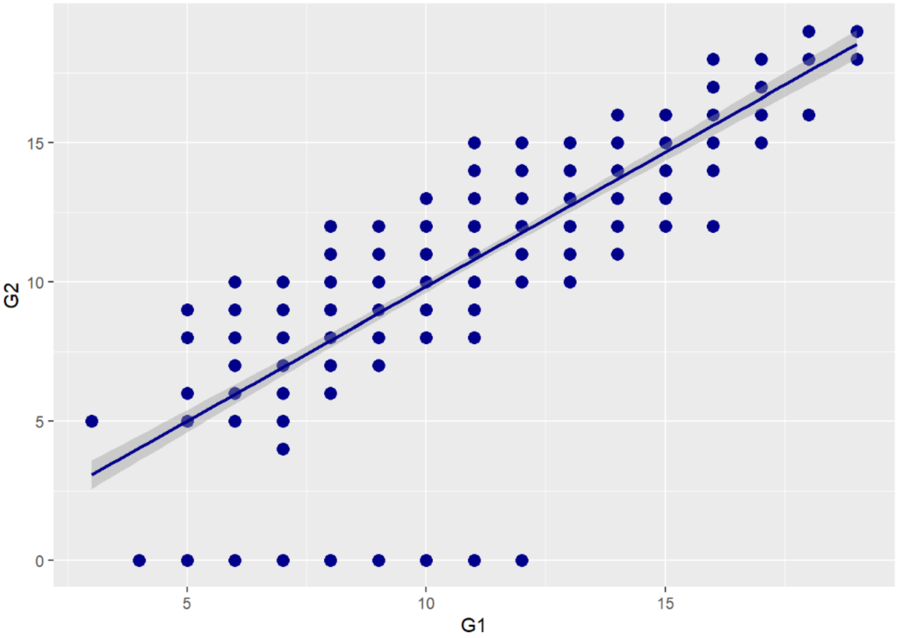


Figure 8: Correlation G1 & G2

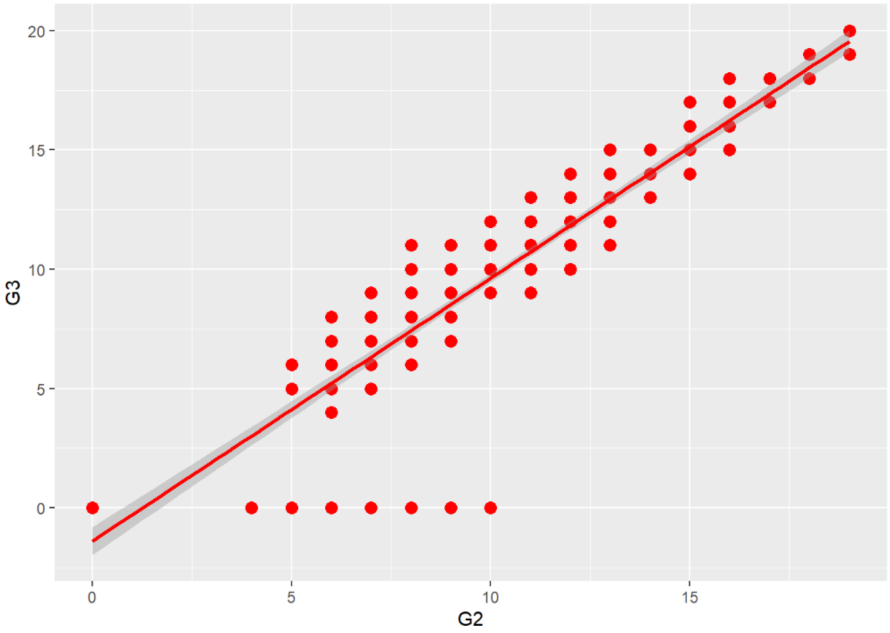


Figure 9: Correlation G2 & G3

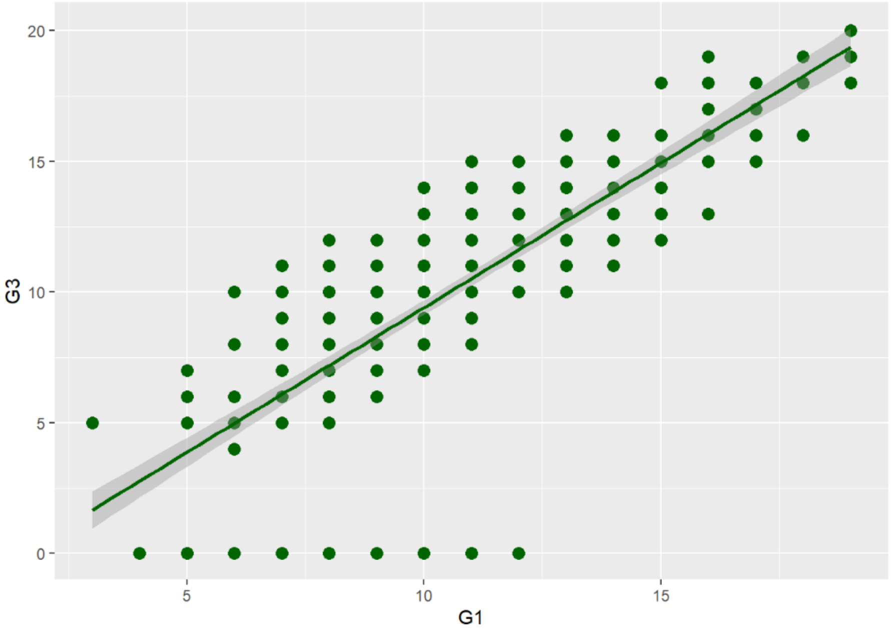


Figure 10: Correlation G1 & G3

From Figures 8, 9, 10, we can see that G1, G2, G3 are correlated to each other and that G3 depends on the grades received from the first and second period (G1 & G2). Because of this correlation I decided to leave out G1 and G2 and to continue with just having the target feature G3.

3.3.1 HeatMap - correlations

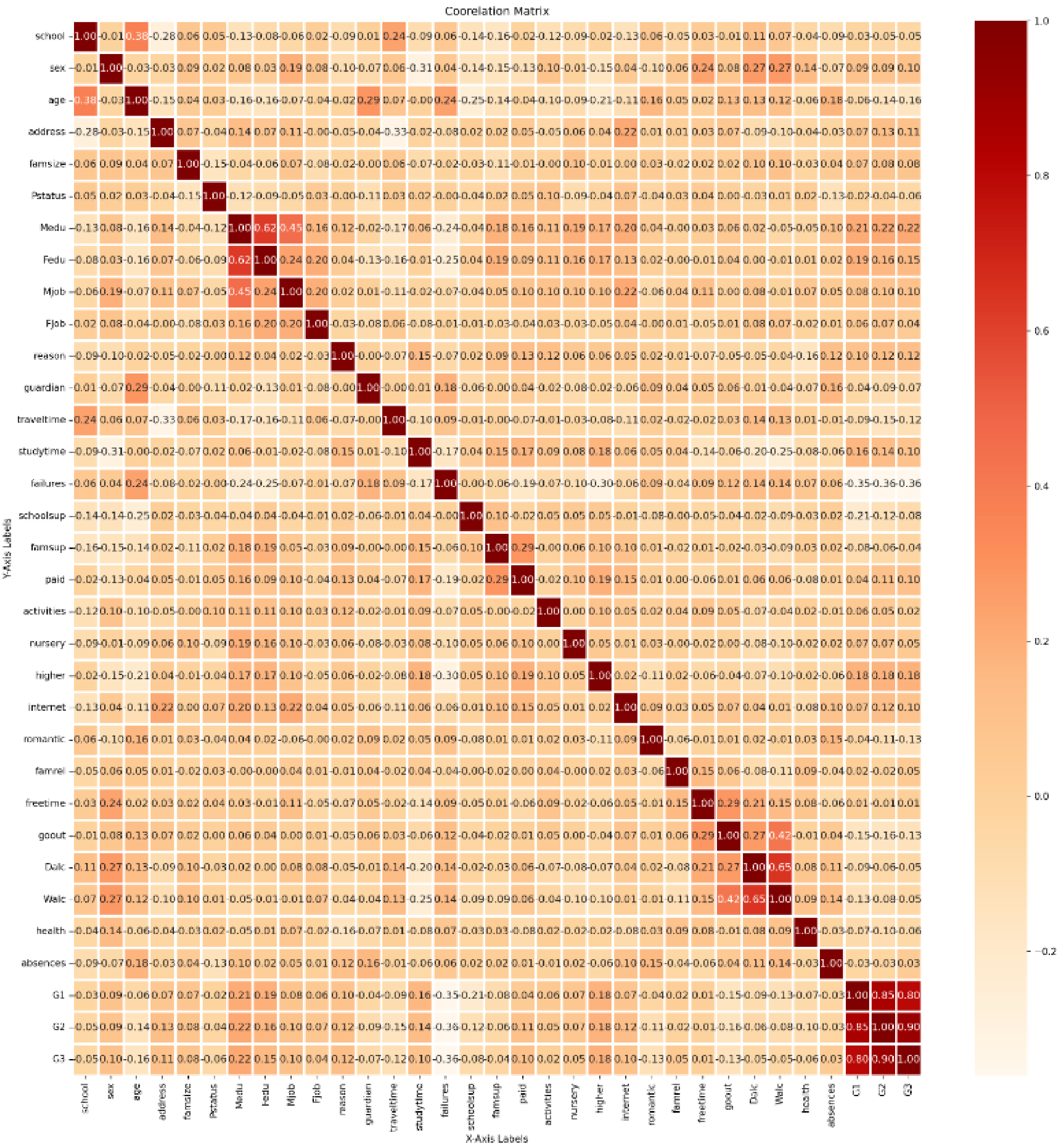


Figure 11: Heatmap - Correlation Matrix

To analyze the relationships between the various attributes in our dataset, I have created a heatmap representing the **correlation matrix**. This matrix visualizes the pairwise correlations between all attributes, helping us understand the strength and direction of linear relationships within the data. A common would be to select features that have a higher positive and negative correlation with *G3*. However, for the purpose of this project, I will use all the attributes in the dataset in order to fully explore the potential influence of each attribute.

4 DATA PREPARATION

4.1 MISSING VALUES

As discussed in the previous chapter, I outlined the fact that the dataset does not contain any missing values across all attributes, which gave a good advantage when preparing the data. The absence of missing data eliminates the need for imputation or deletion strategies, ensuring the integrity of the dataset.

4.2 OUTLIERS

Outliers are data points that deviate significantly from the the majority of observations in a dataset. They can arise due to various reasons such as measurement errors or data entry mistakes. Outliers can have a significant impact on the accuracy of machine learning models. If they are not detected and handled appropriately, they can lead to overfitting, underfitting or biased results. Outliers can be detected by using the **Interquartile Range method (IQR)** [17]. This method can be visualized in the boxplots drawn in Figure 7, where the blue rectangles represent the range between the 75th and 25th percentiles of the data. The data points that fall outside the lower whisker or above the upper whisker are considered as outliers. Moreover, Python offers several methods and libraries for detecting outliers. In some cases, outliers can provide valuable insights into the data and the underlying processes.
In my case the detected outliers are relatively few in number, so I decided not to handle them separately and to include them in the modeling phase.

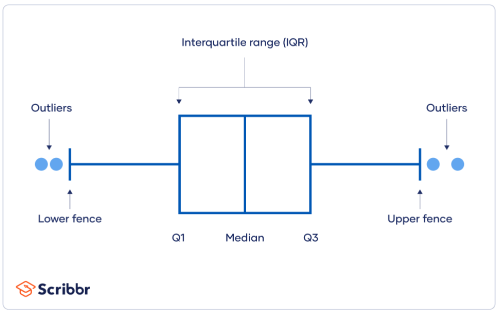


Figure 12: IQR - Interquantile Range Method

4.3 LABEL ENCODING

Label encoding is a technique used in machine learning and data analysis to convert categorical variables into numerical format. It is particularly useful when working with algorithms that require numerical input, as most machine learning models can only operate on numerical data. Most of the machine learning algorithms will not understand or not be able to deal with categorical variables, meaning that they would perform much better in terms of accuracy when **data is represented as a number**.
Deep learning techniques such as the Artificial Neural Network expect data to be numberical. Other models such as Tree-based do a better job in handling **Categorical** variables.

```
le = LabelEncoder()
for column in df.columns:
    if df[column].dtype == 'object':
        df[column] = le.fit_transform(df[column])
```

5 MODELING

5.1 CLASSIFICATION

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.
Even though classification and regression are both from the category of supervised learning, they are not the same. The prediction task is a classification when the target variable is discrete. [8]

In my case, the target attribute G3 is a numeric value ranging from 0 to 20. To facilitate the application of the classification models, I discretized G3 into five classes of grades: A to F. This transformation allows for a categorical interpretation of the final grade, enabling to use classification algorithms. The discretization was made solely on G3 and I additionaly dropped the other two grades columns G1 and G2.

- **Class A** $\leftarrow 16 - 20$
- **Class B** $\leftarrow 14 - 15$
- **Class C** $\leftarrow 12 - 13$
- **Class D** $\leftarrow 10 - 11$
- **Class F** $\leftarrow 0 - 9$

5.1.1 ZeroR Classifier (0R)

ZeroR is the simplest classification method which relies on the target and ignores all predictors. ZeroR classifier simply predicts the majority category and it does not require any training beyond this step. Althought it does not provide any insights into the relationships between features and the target variable, ZeroR serves as a benchmark for other classification methods. ZeroR is only useful for determining a baseline performance for other classification methods. [14]

5.1.2 OneR Classifier (1R)

OneR, short for **"One Rule"**, is a simple, yet accurate, classification algorithm that generates one rule for each predictor in the data, then selects the rule with the smallest total error as its "one rule". It is known for its simplicity and interpretability. It has been demonstrated that OneR generates rules with accuracy that is only marginally lower than that of advanced classification algorithms, while also producing rules that are easy for humans to understand. [13]

5.1.3 Random Forest Classifier

Random forests are a popular supervised machine learning algorithm. Random forests can be used for solving both regression and classification problems. In random forest classification. multiple decision trees are created using different random subsets of the data and features. Each decision tree provides its opinion on how to classify the data. Predictions are made by calculating the prediction for each decision tree, then taking the most popular result.

5.1.4 Support Vector Machine (SVM)

Support Vector Machines is primarly known as a classification technique, but it can be also utilized for both classification and regression tasks. In SVM, a hyperplane is constructed within a multidimensional space to segregate different classes. The central concept of SVM is to find a Maximum Marginal Hyperplane (MMH), which optimally separates the dataset into distinct classes

The main goal of SVM is to separate the given dataset as effectively as possible. The margin refers to the distance between the nearest points of different classes. The aim is to choose a hyperplane that maximizes this margin between the support vectors in the dataset. SVM finds the maximum margin hyperplane through the following steps:

- Generate multiple hyperplanes to separate the classes optimally. On the left hand of Figure 13, four hyperplanes (green, orange, red, purple) are depicted.

- Select the optimal hyperplane with the maximum margin from the nearest data points, as illustrated in the right-hand figure.

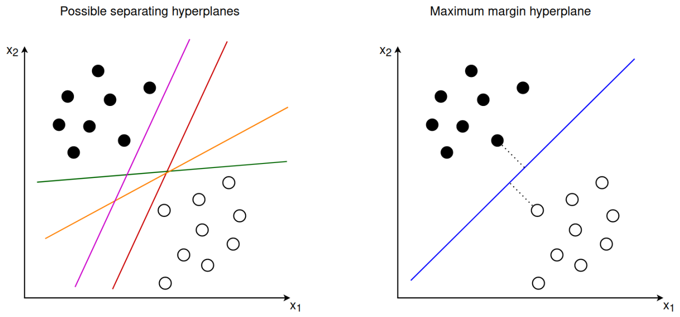


Figure 13: Support Vector Machine [10]

Support Vectors are data points, which are closest to the hyperplane. They play a crucial role in defining the separating boundary by determining the margins. These points are essential for constructing the classfier accurately.
A hyperplane is a decision boundary that separates a set of objects based on their class memberships. [9]

5.2 REGRESSION

In this section I will describe the regression models I selected to predict the student's final grade of the year, known as the target variable G3. By applying different regression techniques and tuning their hyperparameters, I aim to identify the most effective model for predicting the final grade.

5.2.1 Decision Tree Regressor

Decision Tree is one of the most commonly used approaches in supervised learning. It can be used to solve both regression and classification tasks, with classification tasks being particularly common in practical applications. It is a tree-structured model consinsting of three types of nodes. The Root Node is the topmost node that represents the entire dataset and can be split into other nodes. Interior Nodes represent the features of the dataset, with branches denoting decision rules. A decision rule is a simple IF-THEN statement consisting of a condition and a prediction. Leaf Nodes represent the final outcomes. This algorithm is highly effective for addressing decision-making problems. [4]

In the case of this project, the Decision Tree algorithm will be implemented to solve the Regression problem of the dataset.

5.2.2 Random Forest Regressor

Random Forest uses the qualities features of multiple Decision Trees for making decisions, then it is referred as a *Forest*. Decision Trees themselves have a major drawback, since it has the tendency to overfit the data. This issue is mitigated in Random Forest Regression, where the aggregation of multiple trees reduces the risk of overfitting. Additionally, the Random Forest algorithm is faster and more robust compared to many other regression models, making it a powerful tool for regression tasks. [5]

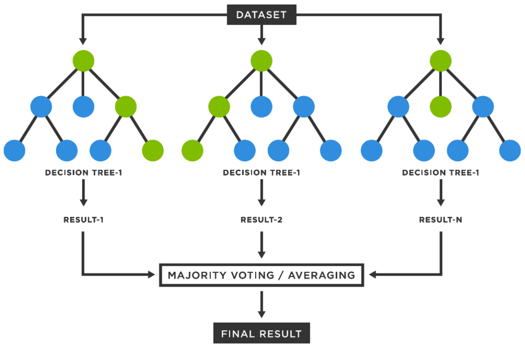


Figure 14: Random Forest & Decision Trees

5.2.3 K-Nearest Neighbors (kNN)

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning classifier that utilizes proximity to make classifications or predictions about the grouping of individual data points. It is one of the most popular and straightforward classifiers used in machine learning today for both classification and regression tasks. The KNN algorithm assumes that similar data points are near each other. For classification problems, it assigns a class label based on a majority vote among the nearest neighbors. In regression problems, the algorithm predicts values by averaging the k nearest neighbors. The key distinction is that classification is used for discrete values, while regression deals with continuous values. Before making any classification, the distance between points must be calculated, with Euclidean distance being the most commonly used method. Other distance possibilities can be the Manhattan distance, the Minkowski distance and the Hamming distance. [7]

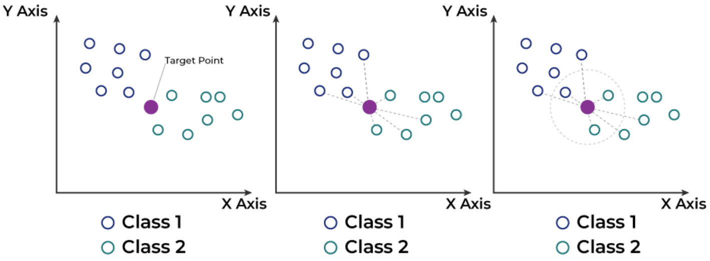
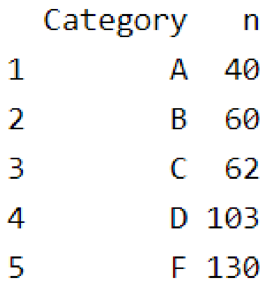


Figure 15: K-Nearest Neighbors [3]

5.3 METHODOLOGY

5.3.1 ZeroR

The first step in ZeroR is to construct a frequency table and identify the majority class (most frequent value):



```
  Category   n
1        A  40
2        B  60
3        C  62
4        D 103
5        F 130
```

Figure 16: Frequency table

These values can be also represented as a histogram:

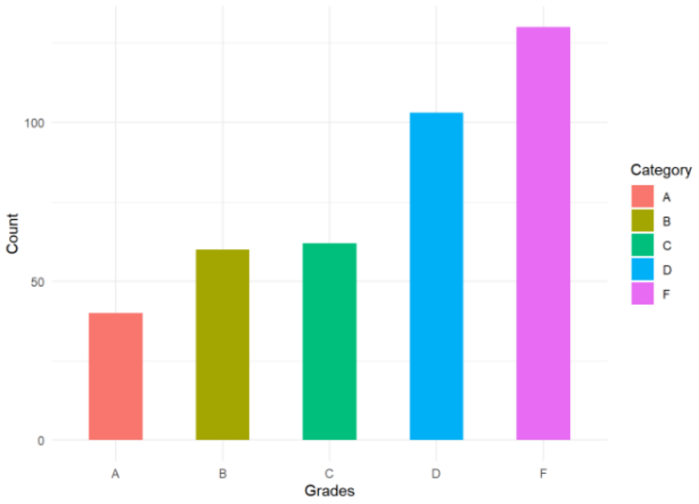


Figure 17: Target distribution

As we can see from Figures 16 & 17, the class value F is the **Majority Class** and it is contained in 130 instances. Since the majority class is F, then for any input, the prediction will be F.
The performance of ZeroR can be typically assessed using accuracy, by constructing a confusion matrix and calculating the accuracy as the number of correct predictions divided by the total number of instances.

$$Accuracy = \frac{CorrectPredictions}{TotalInstasnces} = \frac{130}{395} \approx 0.329 = 32.9\%$$

5.3.2 OneR

The One Rule classifier can be implemented in R programming language by using the OneR library. The code is the following:

```r
library(OneR)

modelOne <- OneR(data_train, verbose = TRUE)

summary(modelOne)

prediction <- predict(modelOne, data_test)

resultsOne <- eval_model(prediction, data_test)

print(resultsOne$confusion_matrix)
print(paste("OneR accuracy is: ", resultsOne$accuracy))
oneRfinal <- (resultsOne$accuracy)
```

OneR() is the main function of the package. It builds a model according to the One Rule machine learning algorithm for categorical data. All numerical data is automatically converted into five categorical bins of equal length.
Predict() is a method for predicting cases or probabilites based on OneR model objects. The second argument can have the same format as used for building the model but must at least have the feature variable that is used in the OneR rules. The default output is a factor with the predicted classes.
Evalmodel() is a simple function for evaluating a OneR classification model. It prints confusion matrices with prediction vs. actual in absolute and relative numbers. Additionally it gives the accuracy, error rate as well as the error rate reduction versus the base rate accuracy together with a p-value. [16]

OneR detected rules are:

- **If absences = (-0.075,0] then categoryClass = F**
- **If absences = (0,7.5] then categoryClass = D**
- **If absences = (7.5,20] then categoryClass = F**
- **If absences = (20,24] then categoryClass = A**
- **If absences = (24,75.1] then categoryClass = F**

$$Accuracy \approx 37.03\%$$

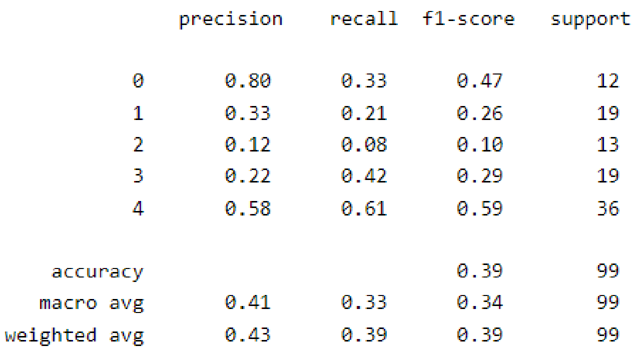
5.3.3 Random Forest

- **Implemented code in Python:**

```python
data = pd.read_csv('discretizedDF.csv')

#Categorical features
cat_features = [feature for feature in data.columns if data[feature].dtype == 'O']

# Numerical attributes
num_features = [feature for feature in data.columns if data[feature].dtype != 'O']



le = LabelEncoder()
for column in cat_features:
    if data[column].dtype == 'object':
        data[column] = le.fit_transform(data[column])

X = data.drop("Category", axis=1)
y = data["Category"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=42)

```

```python
modelR = es.RandomForestClassifier()
kfR = KFold(shuffle=True, n_splits=5)
RandForCvResults = cross_val_score(modelR, X, y, cv=kfR,
    scoring='accuracy')
tmp = modelR.fit(X_train,y_train)
tmp.score(X_test, y_test)
y_test_pred = modelR.predict(X_test)
```

- **Classification report:**



```
              precision    recall  f1-score   support

           0       0.80      0.33      0.47        12
           1       0.33      0.21      0.26        19
           2       0.12      0.08      0.10        13
           3       0.22      0.42      0.29        19
           4       0.58      0.61      0.59        36

    accuracy                           0.39        99
   macro avg       0.41      0.33      0.34        99
weighted avg       0.43      0.39      0.39        99
```

Figure 18: Random Forest Classification report

- **Confusion Matrix:**

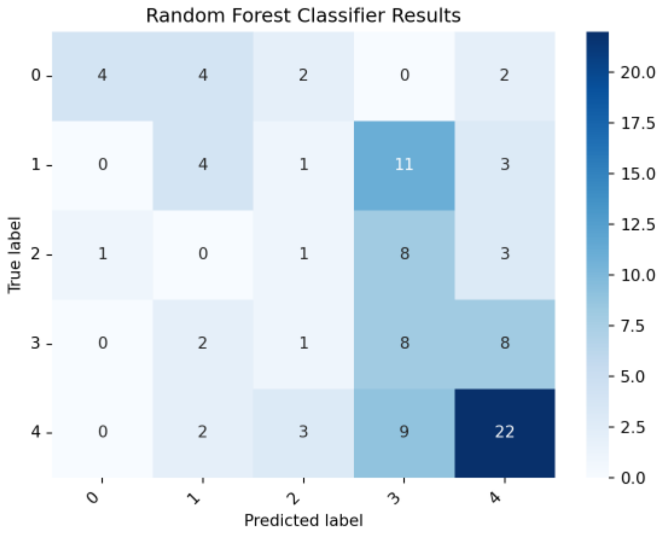


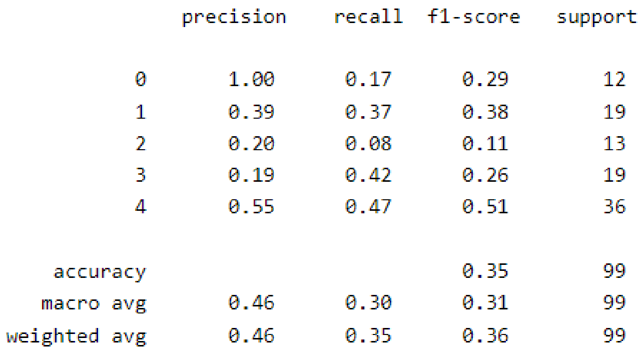
Figure 19: Random Forest Confusion matrix

5.3.4 kNN

- **Implemented code in Python:**

```python
KNclassifier = KNeighborsClassifier(n_neighbors=20)

X = data.drop("Category", axis=1)
y = data["Category"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=42)

kfK = KFold(shuffle=True, n_splits=5)
KnnCvResults = cross_val_score(KNclassifier, X, y, cv=kfK, scoring='accuracy')
tmp1 = KNclassifier.fit(X_train, y_train)
tmp1.score(X_test,y_test)
y_pred = KNclassifier.predict(X_test)
```

- **Classification Report:**



```
              precision    recall  f1-score   support

           0       1.00      0.17      0.29        12
           1       0.39      0.37      0.38        19
           2       0.20      0.08      0.11        13
           3       0.19      0.42      0.26        19
           4       0.55      0.47      0.51        36

    accuracy                           0.35        99
   macro avg       0.46      0.30      0.31        99
weighted avg       0.46      0.35      0.36        99
```

Figure 20: KNN Classification report

- **Confusion Matrix:**

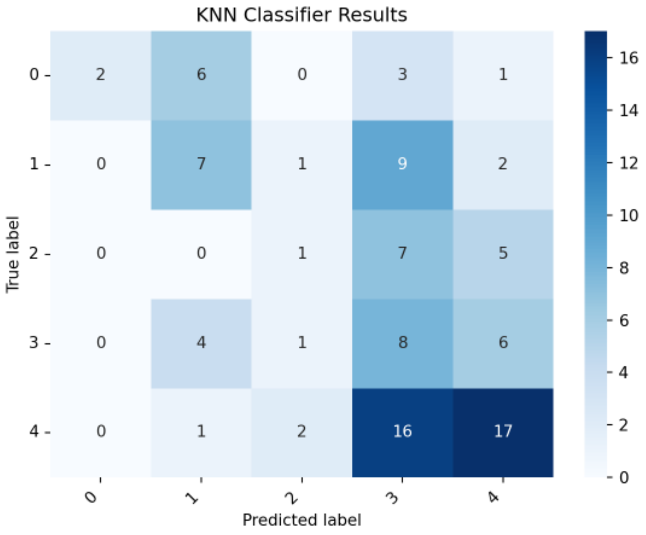


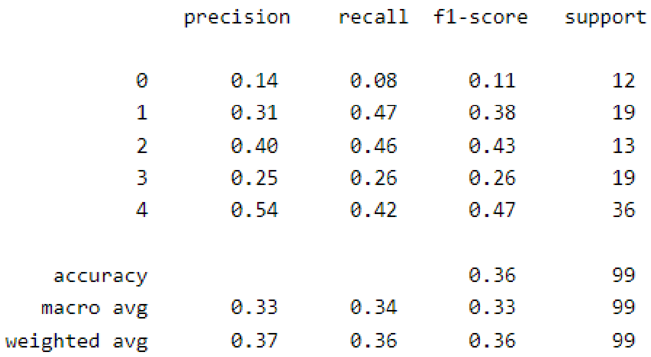
Figure 21: KNN Confusion matrix

5.3.5 SVM

- **Implemented code in Python:**

```python
X = data.drop("Category", axis=1)
y = data["Category"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=42)

SVCclassifier = SVC(kernel='linear', max_iter=251)
kfS = KFold(shuffle=True, n_splits=5)
SVCcvResults = cross_val_score(SVCclassifier, X, y, cv=kfS, scoring='accuracy')
tmp2 = SVCclassifier.fit(X_train, y_train)
tmp2.score(X_test,y_test)
y_pred = SVCclassifier.predict(X_test)
```

- **Classification Report:**



```
              precision    recall  f1-score   support

           0       0.14      0.08      0.11        12
           1       0.31      0.47      0.38        19
           2       0.40      0.46      0.43        13
           3       0.25      0.26      0.26        19
           4       0.54      0.42      0.47        36

    accuracy                           0.36        99
   macro avg       0.33      0.34      0.33        99
weighted avg       0.37      0.36      0.36        99
```

Figure 22: SVM Classification report

- **Confusion Matrix:**

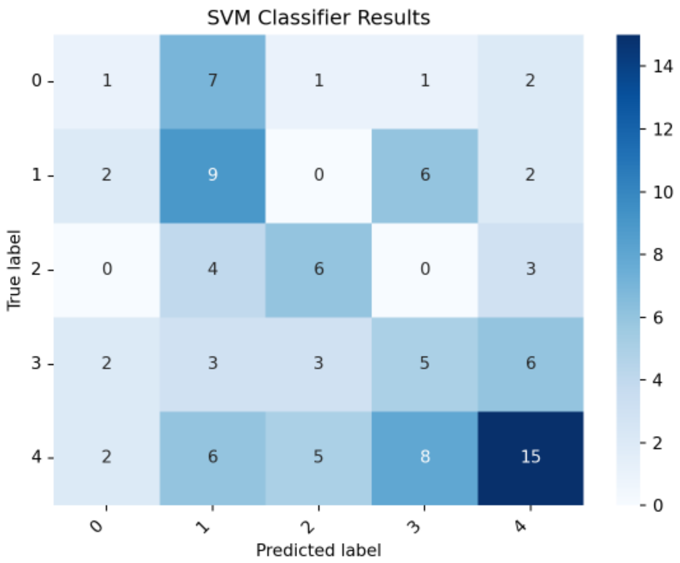


Figure 23: SVM Confusion matrix

5.3.6 Decision Tree Regressor

```
DecisionTree_reg = DecisionTreeRegressor(random_state = 42)
DecisionTree_reg.fit(X_train1, y_train1)
# Making prediction on data
y_pred = DecisionTree_reg.predict(X_test1)

ndf = [Regression_metrics(DecisionTree_reg, X_train1, y_train1, X_test1,
↪ y_test1, y_pred)]

```

```python
dtScore = pd.DataFrame(data = ndf, columns=['R2 Score','Adjusted R2 Score','Cross Validated R2 Score','RMSE', 'MAE', 'MSE'])
dtScore.insert(0, 'Model', 'Decision Tree')
table = dtScore.to_string(index=False)
print(table)
```

5.3.7 Random Forest Regressor

```python
RandomForest_reg = RandomForestRegressor(n_estimators = 10, random_state = 0)
RandomForest_reg.fit(X_train1, y_train1)
# Making prediction on data
y_pred = RandomForest_reg.predict(X_test1)

ndf = [Regression_metrics(RandomForest_reg, X_train1, y_train1, X_test1, y_test1, y_pred)]

rfScore = pd.DataFrame(data = ndf, columns=['R2 Score','Adjusted R2 Score','Cross Validated R2 Score','RMSE', 'MAE', 'MSE'])
rfScore.insert(0, 'Model', 'Random Forest')
table = rfScore.to_string(index=False)
print(table)
```

5.3.8 KNN Regressor

```python
KNeighbors_reg = KNeighborsRegressor(n_neighbors=10)
KNeighbors_reg.fit(X_train1,y_train1)
# Making prediction on data
y_pred = KNeighbors_reg.predict(X_test1)

ndf = [Regression_metrics(KNeighbors_reg, X_train1, y_train1, X_test1, y_test1, y_pred)]

knnScore = pd.DataFrame(data = ndf, columns=['R2 Score','Adjusted R2 Score','Cross Validated R2 Score','RMSE', 'MAE', 'MSE'])
knnScore.insert(0, 'Model', 'KNN Regressor')
table = knnScore.to_string(index=False)
print(table)
```

6 EVALUATION AND RESULTS

6.1 EVALUATION METHODOLOGY

Evaluating machine learning models is a crucial step in the development process to ensure their performance and reliability. Different evaluation techniques are used depending on whether the model is designed for classification or regression tasks. In this section I will describe the evaluation methods that I used and how I have used them to assess my implemented models. Moreover, I have used also utilized cross-validation to ensure robustness.

The primary difference between the evaluation of classification and regression models lies in the nature of the predictions. Classification models deal with discrete outcomes and are evaluated based on their ability to correctly classify instances into categories. Metrics like **accuracy, precision, recall, f1-score, ROC curves** are specifically designed to handle categorical data.
Regression models handle continuous outcomes and are evaluated based on the accuracy of their numerical predictions. Metrics such as **MAE, MSE, RMSE,** R^2 are designed to measure the magnitude of prediction errors and the model's explanatory power.

6.1.1 Evaluating Classification Models

An important part of building classification models is evaluating model performance. This means that data scientists need a reliable way to test approximately how well a model will correctly predict an outcome. Many tools are available for evaluating depending on the problem you're trying to solve, some may be more useful than others. The methods and metrics I used are the following:

- **Accuracy** Accuracy is the simplest evaluation metric. It is defined as the number of correct predictions divided by the total number of predictions multiplied by 100. This metric works great if the target variable classes in the data are approximately balanced.

$$Accuracy = \frac{Number of Correct Predictions}{Total of Predictions}$$

- **Confusion Matrix:** The confusion matrix counts the number of times instances of some class A are classified as some class B. This is a table that is often used to describe the performance of a classification model. It presents a summary of the predictions made by the model against the actual class labels. It contains four different combinations of predicted and actual classes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

- **Precision, Recall and F1 Score:**

  1. *Precision:* Precision measures the accuracy of the positive predictions made by the classifier. Even if the confusion matrix is a great way to evaluate the performance, sometimes there is the need to have a more concise metric.

  $$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

  2. *Recall:* Recall is the proportion of True Positive predictions (those correctly identified by the classifier) to the total number of actual positive instances in the dataset. It measures how in depth the classifier identifies all positive instances.

  $$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

  3. *F1-Score:* The F1-Score is the harmonic mean of precision and recall, emphasizing classifiers that perform well in both metrics. This balance is vital in situations where both metrics are equally important.

  $$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- **ROC Curve:** The **Receiver Operating Characteristic (ROC) curve** visually represents a classification model's performance across different thresholds. It plots the True Positive (TPR) against the False Positive Rate (FPR).

$$TPR = \frac{TruePositives}{TruePositives + FalseNegatives}, FPR = \frac{FalsePositives}{FalsePositives + TrueNegatives}$$

Python offers libraries such as “sklearn” and “metrics” that provide predefined functions for each evaluation metric, making it straightforward to evaluate the performance of the models. By utilizing these tools, I was able to effectively assess them. [2]

6.1.2 Evaluating Regression Models

For the regression models, which predict continuous outcomes, I have focused on the accuracy of these predictions using several metrics:

- **Mean Absolute Error (MAE):** The Mean Absolute Error represents the average of the the absolute differences between the predicted and actual values. This metric provides a straight forward measure of prediction error.

  $$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|$$

  where x_i represents the actual or observed values for the i-th data point, y_i instead represents the predicted value for the i-th data point.

- **Mean Squared Error (MSE):** The Mean Squared Error measures the square root of the average differences between the predicted and actual values. MSE is commonly used in regression problems to evaluate the performance of predictive models.

  $$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$$

- **Root Mean Squared Error (RMSE):** RMSE measures the accuracy of a model's predicted values by comparing them to the actual observed values in the dataset. It represents the square root of the MSE, providing an error metric in the same units as the target variable, making it more interpretable.

  $$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

- **R-Squared (R^2):** The R-squared score, also known as the coefficient of determination, is a common statistical metric used to evaluate how well a regression model fits the data. It measures the proportion of the variance in the dependent variable that can be explained by the independent variables in the model.R^2 provides a useful indicator of the model's overall performance and explanatory power. When R^2 is high we can say that the regression model performs well, on the other hand when it is a low value then this can be a bad thing.

  $$R^2 = 1 - \frac{SSR}{SST}$$

where **SSR** is the sum of the differences between the predicted value and the mean of the dependent variable, and **SST** is the squared differences between the observed dependent variable and its mean.

- **Adjusted R-Squared: Adjusted R-squared** is an improved version of R-squared that accounts for the number of independent variables in the model. It adjusts the R-squared value to reflect the number of predictors, ensuring it is always less than or equal to the R-squared.

$$R^2_{adj.} = 1 - (1 - R^2) * \frac{n-1}{n-p-1}$$

where n is the number of observations in the data and p is the number of independent variables in the data. [12]

- **Cross-Validated R-Squared: Cross-Validated** R^2 is a median value of or R^2 taken from **cross-validation** procedure.

6.1.3 Cross-Validation

Cross-Validation is a resampling technique used to assess the performance of machine learning models on a limited dataset. It is widely favored due to its simplicity and its ability to provide a less biased or overly optimistic evaluation of model performance compared to methods like a straightforward train/test split.

I specifically utilized the cross-validation method applying the k-fold cross-validation technique. In this process, I set k to 10, meaning the dataset was divided into 10 distinct folds. Each fold was shuffled before being used for validation to ensure that the evaluation was both detailed and unbiased. This approach helped in obtaining a more reliable assessment of the model's performance.

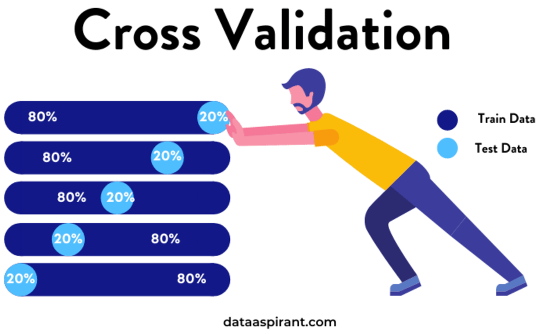


Figure 24: Croos-Validation technique [11]

6.2 RESULTS

Table 2: Classification models' results

Model	**Accuracy**	**10C-V Acc.**	**Precision**	**Recall**	**F1-Score**
Random Forest	0.3939	0.3492	0.4119	0.3306	0.3419
KNN	0.3535	0.3471	0.4647	0.3011	0.3081
SVM	0.3636	0.2812	0.3278	0.3397	0.3268

Table 3: Regression models' results

Model	**R2 Score**	**Adj. R2 Score**	**C-V R2 Score**	**RMSE**	**MAE**	**MSE**
Decision Tree	0.7241	0.5321	0.8175	2.3787	1.3038	5.6582
Random Forest	0.8006	0.6619	0.8764	2.0219	1.1671	4.0881
KNN	0.8010	0.6626	0.8221	2.0198	1.4443	4.0796

From Table**??** we can recognize that the classification models have a moderate performance, with none of them achieving high accuracy or F1-scores (around 30-40%).
The Random Forest classifier has 39.39% chance of making the right predictions. The model's average accuracy over 10-fold-cross validation is lower than the standard accuracy. The precision and recall values are relatively low. The F1-score, which is the harmonic mean of precision and recall, indicates a balance between precision and recall, but overall it is still a low score.
The K-Nearest Neighbors classifier has a low accuracy as well, but it has a precision of 0.4647, which is the highest among the three models, indicating that KNN is the most precise when predicting a class. The recall value is low suggesting that it does not catch as many true positive instances. The F1-score is the lowest among the models.
The Support Vector Machine model has the lowest cross-validated accuracy, showing a significant drop in performance. Both precision and recall are considerably low, leading to a lower F1-score.
Overall, the KNN model has the highest precision but suffers from lower recall, while the Random Forest and SVM models have slightly more balanced performance metrics. These results suggests difficulty in accurately predicting the final grade as a discretized class.

The Decision Tree Regressor reaches a R2 score of 0.7241, which indicates that the model explains 72.41% of the variance in the final grades. The 0.5321 Adjusted

R2 score suggests that the model's explanatory power decreases, indicating a possible overfitting. The model generalizes well during cross-validation. MAE indicates that the Decision Tree model's predictions are off by about 1.3 points on average.
The Random Forest Regressor has higher R2 score than that of the Decision Tree, suggesting better explanatory power and fit. The model mantains a strong performance still after adjusting for the number of predictors and has the highest cross-validated R2 score among the models. RMSE, MAE and MSE are all very low, confirming the Random Forest's better performance.
The KNN Regressor has similar outcomes as the Random Forest model, meaning that also the KNN model performs very well and has a very strong fit.
Overall, among the regression models, Random Forest and KNN perform comparably well, with Random Forest being slightly better. Decision tree stays behind these two in overall predictive accuracy, but still performs decently.

The results show that the regression models perform much better compared to the classification models, suggesting that the initial problem is more suitable when having to predict a continuous value rather than a discretized class.

7 CONCLUSION AND FUTURE WORK

In conclusion, this diploma thesis provided an exploration of the CRISPM-DM methodology, applied to a chosen dataset to gain insights and develop predictive models. Throughout the thesis, each phase of the CRISP-DM was followed, beginning with data understanding and preparation. Key steps included detailed data analysis, visualizing data distributions with boxplots, removing outliers, and handling missing values to prepare the dataset for predictive modeling.

Classification and regression models were implemented to compare their effectiveness in predicting outcomes. These models were then evaluated using various performance metrics, including accuracy, precision, recall, F1 score, MSE, RMSE, and R-squared, giving a clear overview of how well they performed.

For future work, I am really motivated to further test how well these models perform on different datasets. Adjusting the model settings, combining different models, and improving how I select features could also enhance the models' performance. This diploma thesis has provided me a solid foundation for me to further explore machine learning and predictive modeling.

8 BIBLIOGRAPHY

[1] Cortez, P., and Silva, A. Using data mining to predict secondary school student performance. *Dep. Information Systems/Algoritmi RD Centre 1* (2008), 1–8. *(Cited at page 6.)*

[2] GeeksForGeeks. Evaluation metrics for classification model in python, 2024. *(Cited at page 37.)*

[3] GeeksForGeeks. K-nearest neighbor(knn) algorithm, 2024. *(Citirano at pages VIII in 28.)*

[4] Gurucharan, M. K. Machine learning basics: Decision tree regression, 2020. *(Cited at page 26.)*

[5] Gurucharan, M. K. Machine learning basics: Random forest regression, 2020. *(Cited at page 27.)*

[6] Hotz, N. What is crisp-dm, 2024. *(Cited at page 1.)*

[7] IBM. What is the knn algorithm?, 2024. *(Cited at page 27.)*

[8] Keita, Z. Classification in machine learning: An introduction, 2022. *(Cited at page 24.)*

[9] Navlani, A. Support vector machines with scikit-learn tutorial, 2019. *(Cited at page 26.)*

[10] Popkes, A.-L. Extensive guide to support vector machines, 2021. *(Citirano at pages VIII in 26.)*

[11] Rutecki, M. Regression models evaluation metrics, 2022. *(Citirano at pages VIII in 39.)*

[12] Rutecki, M. Regression models evaluation metrics, 2022. *(Cited at page 39.)*

[13] Sayad, S. Oner, 2024. *(Cited at page 25.)*

[14] Sayad, S. Zeror, 2024. *(Cited at page 24.)*

[15] Tounsi, Y. Crisp-dm, 2020. *(Citirano at pages VIII in 4.)*

[16] von Jouanne-Diedrich, H. K. Oner - establishing a new baseline for machine learning classification models, 2017. *(Cited at page 29.)*

[17] Yennhi95zz. The importance of outlier detection in machine learning, 2023. *(Cited at page 22.)*