Predicting Student Performance: A Machine Learning Consultancy Project

Ewa Fojcik and Gabriel Thomas Newton

1. Data Management

This study examines the provided data in order to reproduce a data science consultancy project, where the goal is to build a model capable of predicting whether a student will pass the academic year based on a few characteristics. This research employs machine learning techniques to identify key predictors and construct an accurate classification model.

The features of the data provided were first given numerical, ordinal and nominal labels. In order to understand the data provided, the dataframe .head(), .info() and .describe() functions were used.

The dataset consists of 649 student records with 31 features. It also contains a mix of numerical (14 integer columns) and categorical (17 object columns) features. No null or missing values were found; therefore, no variables were initially removed from the dataset, which simplified the data cleaning process.

The target variable 'Grade' represents student performance, with a wide distribution from 0 to 20, with the mean of 11.9.

The dataset includes a diverse set of categorical features, such as school, sex, address, family size, parental job, and various lifestyle factors. These features require appropriate encoding techniques before being used in machine learning models.

The numerical features were determined to be those with integer values that did not have a fixed domain. Ordinal features had a fixed domain with up to 5 possible options (already encoded as integers). Nominal features were the features which did not have order, and had not previously been assigned integer values. No features were dropped as none seemed irrelevant.

Label	Features
Numerical	'age', 'failures', 'absences'
Ordinal	'Medu', 'Fedu', 'traveltime', 'studytime', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health'
Nominal	'school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic'

The 'Grades' feature was replaced with a binary 'Pass' feature based on whether the value was 12 or higher.

The first step for this project was determining what kind of problem was being faced. Given that the model would have to classify whether or not a student was predicted to pass or fail given other features/data, the problem was determined to be a binary **classification** problem.

In order to avoid introducing unintended ordinal relationships, the next step taken was to **encode** the nominal variables. This was done using SciKit-Learn's LabelEncoder under its preprocessing sub-library.

It was decided that **outliers** would be removed from numerical and ordinal data using the interquartile range (IQR) method. Histograms and box-plots were used for visualisation purposes (see *Figures 1 – 13*). Outliers for the 'failures' data were not removed in this section due to both the high correlation with 'Pass' (seen later and in *Figure 14*) and due to the fact the IQR method would remove all values but 0 failures, making it redundant when logically it is not.

Similar to the 'failures' variable, the ordinal variables' outliers, while extreme, were not considered errors or inconsistencies. All those outliers were within the features' domains. Due to this reasoning, all ordinal variables outliers were initially retained as they represented valid data points and could provide valuable information. It was however observed later on that keeping those values negatively impacted the accuracy of the machine learning models. Consequently, all outliers within the ordinal data were removed.

Next in order to prevent bias in machine learning models, numerical data was **scaled** using SciKit-Learn's StandardScaler function under its preprocessing sub-library. This function applied gaussian normalisation to the numerical data.

In order to perform **feature selection**, correlations were calculated in a correlation matrix (see *Figure 14*). A threshold of 0.2 was chosen, and features whose absolute correlation did not meet the threshold were discarded.

Selected Features	'school', 'address', 'Medu', 'Fedu', 'Mjob', 'studytime', 'failures', 'higher', 'Dalc'
Discarded Features	'sex', 'age', 'famsize', 'Pstatus', 'Fjob', 'reason', 'guardian', 'traveltime', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Walc', 'health', 'absences'

Other thresholds were considered and tested, however 0.2 provided the best balance between feature reduction and model performance. Setting the threshold higher would lead to fewer features being selected and risking underfitting, and selecting too many features would risk overfitting. Additionally, the features selected by this method seemed logically more relevant.

Lastly the data was **randomized**, using the pandas sample method, and **split** into training data (80%) and testing data (20%). **10-fold** cross-validation was chosen for the training of the model. Both of these steps were performed using SciKit-Learn's train_test_split function and Kfold function respectively, both found in its model_selection sub-library.

1. Model Training

The models selected were **Random Forest**, **K-Nearest Neighbors** and **Logistic Regression**. Random Forest was chosen for its ability to handle non-linear relationships and its handling of overfitting. KNN was chosen due to its simplicity to implement, adaptability and its non-parametric nature. Logistic Regression was chosen due to its specialty in classification problems.

Model	Variables	Hyperparameters
Random Forest	n_estimators	100, 150, 200, 250, 300
	max_depth	3, 5, 7, 9, 11, None
KNN	n_neighbours	5, 7, 9, 11
	weights	uniform, distance
	p	1, 2
Logistic Regression	C	0.1, 0.5, 1, 5, 10
	solver	liblinear, lbfgs, saga

The hyperparameter search protocol used was **GridSearch**, which systematically explored hyperparameter combinations and scored outputs by accuracy. 10-fold cross-validation (mentioned in section 1) was used in this step, giving GridSearch multiple splits to evaluate and find the best parameters. This was done using the SciKit-Learn's GridSearchCV function from its model_selection sub-library.

Regarding Random Forests, *the n_estimators variable determines how many decision trees are used. A higher number of n_estimators improves performance at the expense of slower run times. The max_depth variable limits how far each decision tree can branch, the higher the depth the higher the risk of overfitting.*

Hyperparameters, like max_features, min_samples_leaf or bootstrap, were not used, in order to limit the number of hyperparameters for the model to run faster. Using them all would be computationally expensive.

Regarding KNNs, the *n_neighbours variable determines how many neighbours are considered when trying to make a classification (odd numbers were used to prevent ties). The weights variable determines if neighbours all have equal impact when classifying (uniform) or if closer neighbours have a higher impact (distance). The p variable selects whether the model is using Manhattan distance (1) or Gaussian distance (2) in its calculations.*

Regarding Logistic Regression, the *C variable (regularisation strength) determines how flexible the model is when it comes to calculating its output. Low c values can lead to underfitting and higher c values can lead to overfitting.* The **solver** variable determines which algorithm is used by the system.

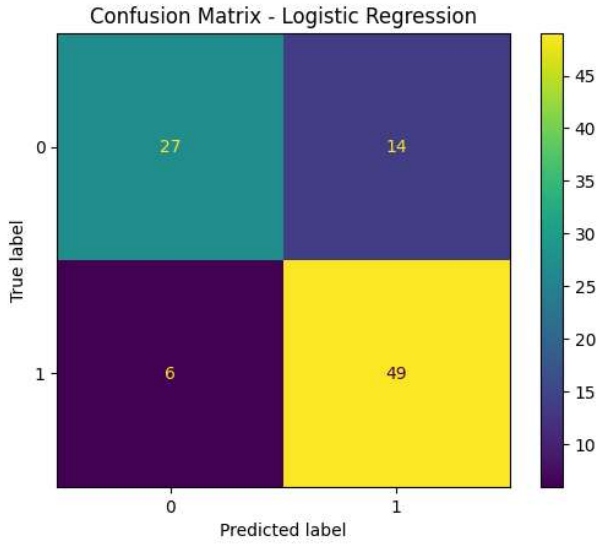
Other hyperparameters were explored but were not found to cause much of a difference in the final results. Examples of this are max_features and min_samples_split for Random Forest; leaf_size for KNN; and penalty for Logistic Regression. These were ultimately left out in order to minimise computational expense/time.

Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.767072	0.720588	0.890909	0.796748
KNN	0.732996	0.685714	0.872727	0.768000
Logistic Regression	0.780297	0.777778	0.890909	0.830508

The best parameters for Random Forest were found to be {'max_depth': 7, 'n_estimators': 150} by GridSearch, with an accuracy score of 0.767072. The best parameters for KNN were found to be {'n_neighbors': 11, 'p': 1, 'weights': 'uniform'} by GridSearch, with an accuracy score of 0.732996. The best model was found to be **Logistic Regression** with parameters **{'C': 1, 'solver': 'liblinear'}** by Grid Search, with an accuracy score of **0.780297**.

2. Evaluation of Model

The following confusion matrix is a visualisation of how well the Logistic Regression classifies the testing data. It provides a breakdown of key findings regarding the model's predictions vs actual outcomes in the test set.



Given the confusion matrix, precision and recall can be calculated to further evaluate the model's performance. Precision showcases high scores for predicting whether a student will pass and Recall scores high in identifying all students who actually passed.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{76}{96} = 0.7917$$

$$Precision = \frac{TP}{TP + FP} = \frac{49}{63} = 0.7778$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{49}{55} = 0.8909$$

$$F1 = F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{49}{59} = 0.8305$$

With a $\frac{27}{27+14} = 0.6585$ accuracy for failures the model is shown to be less reliable when trying to detect students who are at risk of failing.

Confusion Matrix Breakdown		Final Metrics	
True Positives	49	**Accuracy**	0.7917
True Negatives	27	**Precision**	0.7778
False Positives	14	**Recall**	0.8909
False Negatives	6	**F1 Score**	0.8305

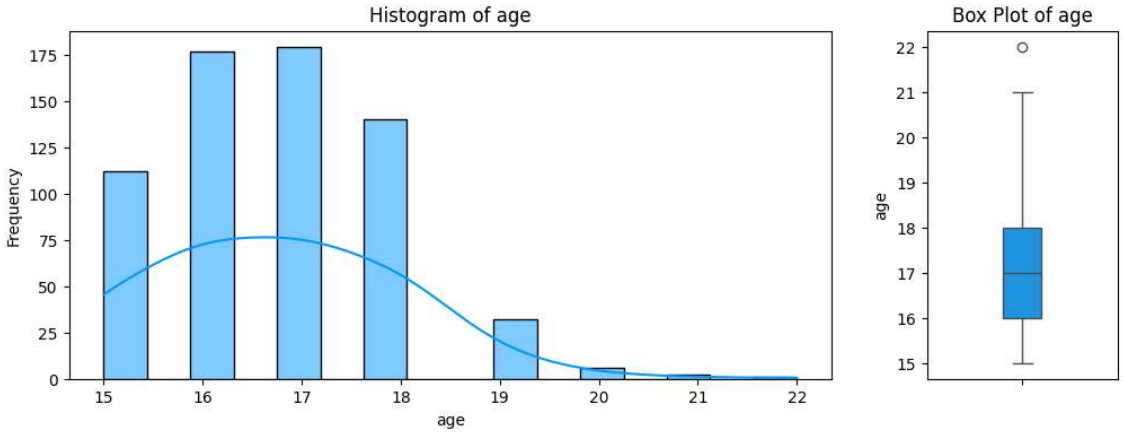


Figure 1 – Histogram and Box-plot of age

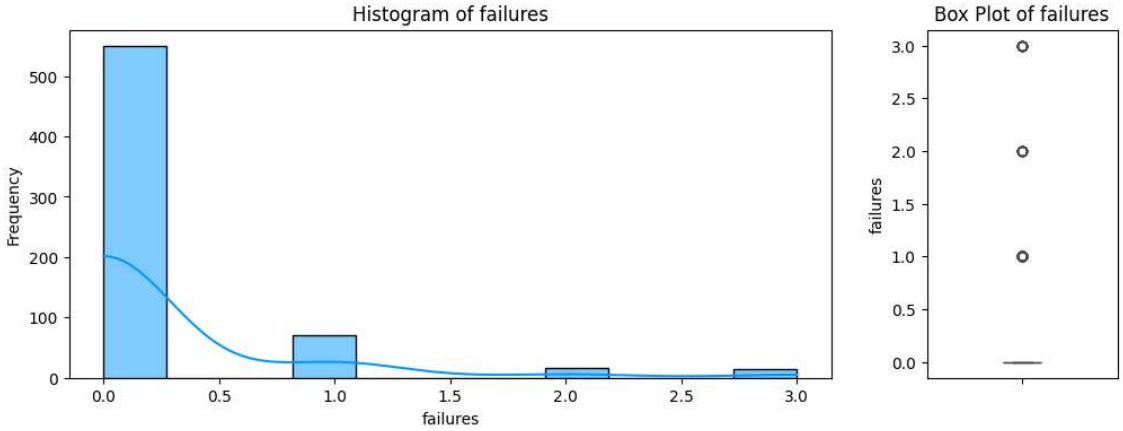


Figure 2 - Histogram and Box-plot of failures

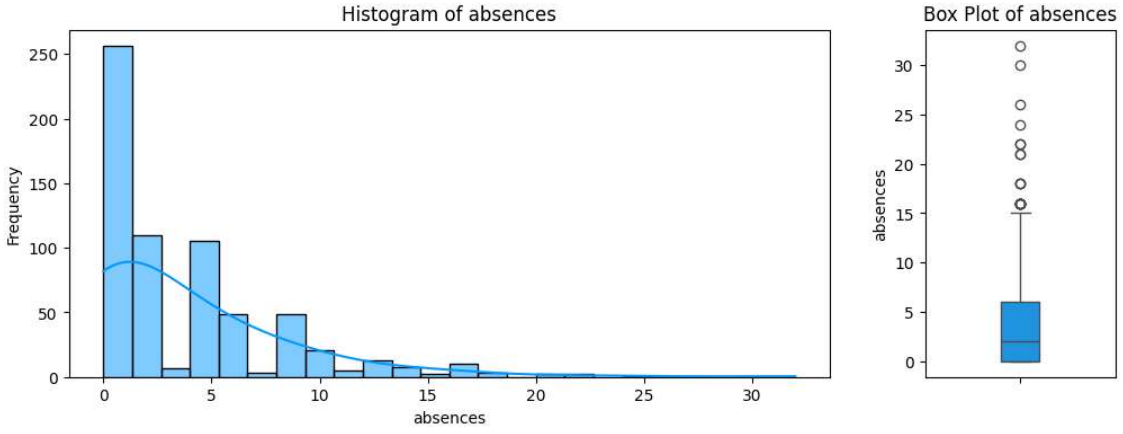


Figure 3 - Histogram and Box-plot of absences

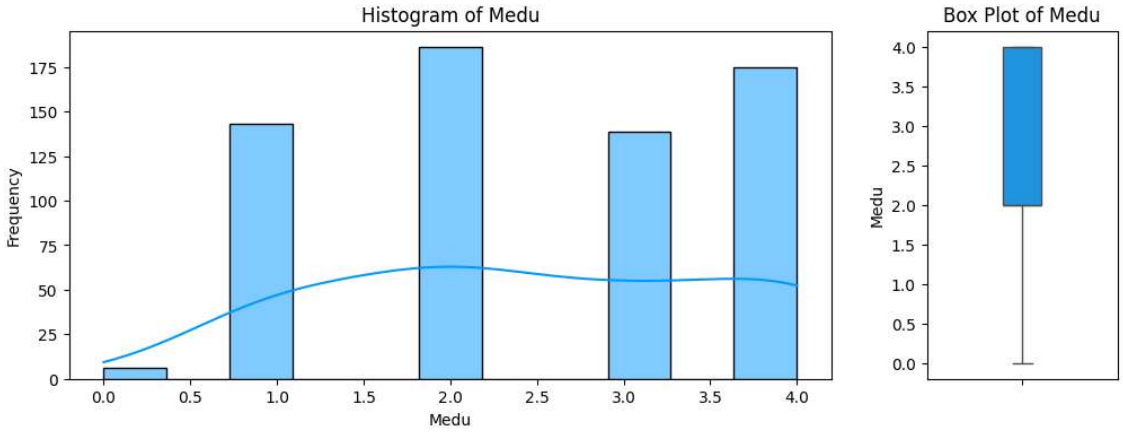


Figure 4 - Histogram and Box-plot of Medu

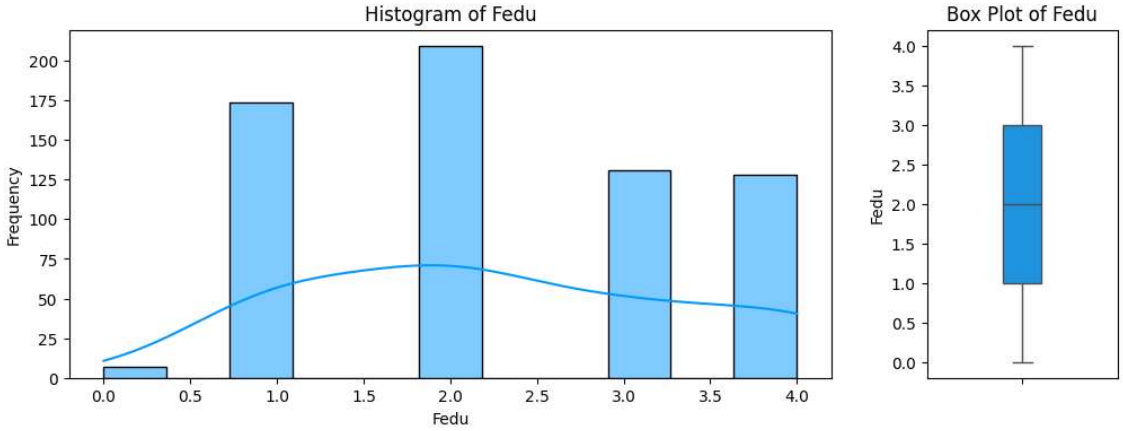


Figure 5 - Histogram and Box-plot of Fedu

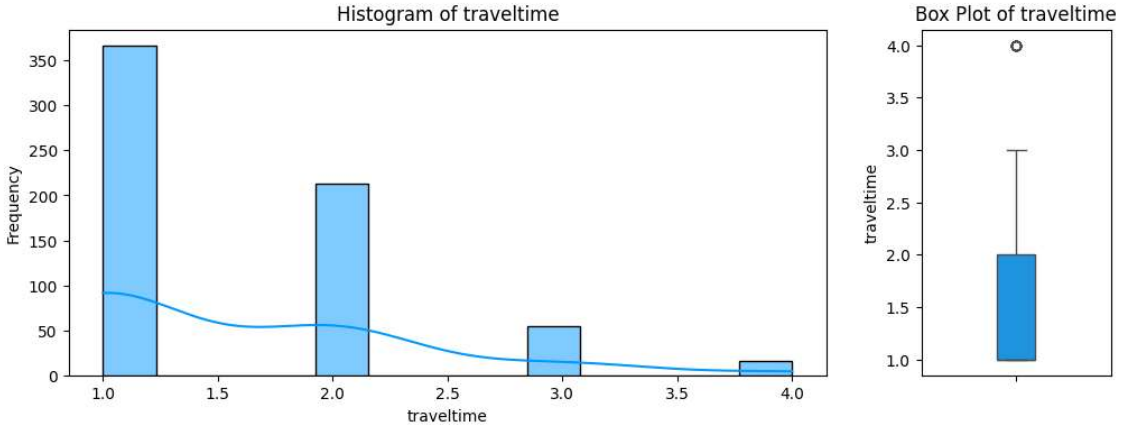


Figure 6 - Histogram and Box-plot of traveltime

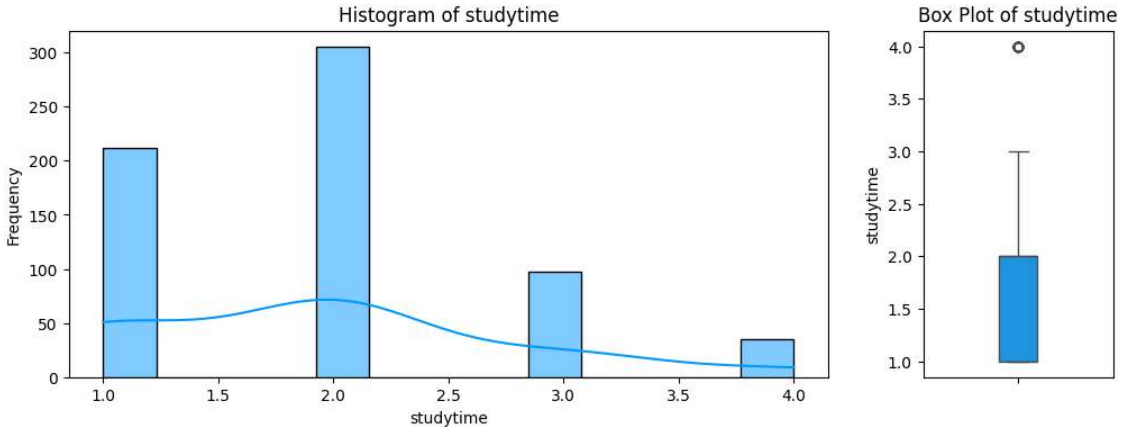


Figure 7 - Histogram and Box-plot of studytime

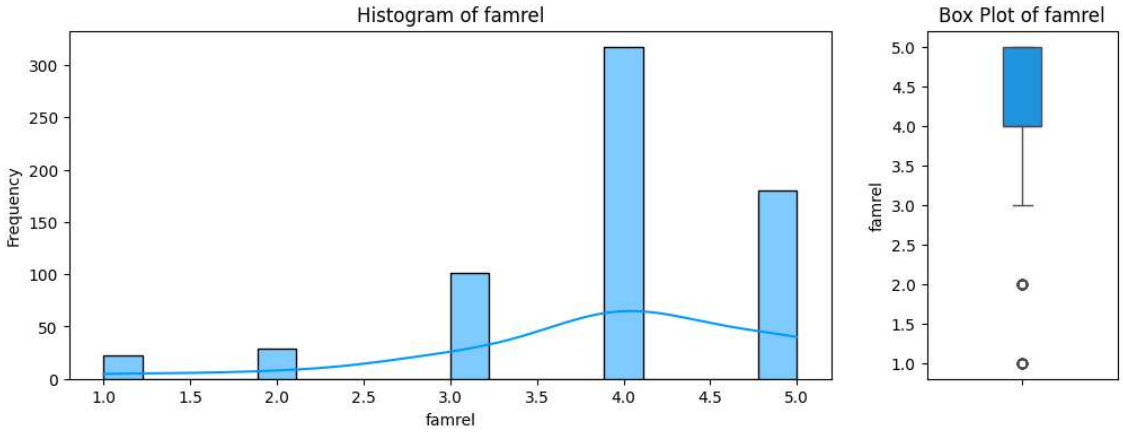


Figure 8 - Histogram and Box-plot of famrel

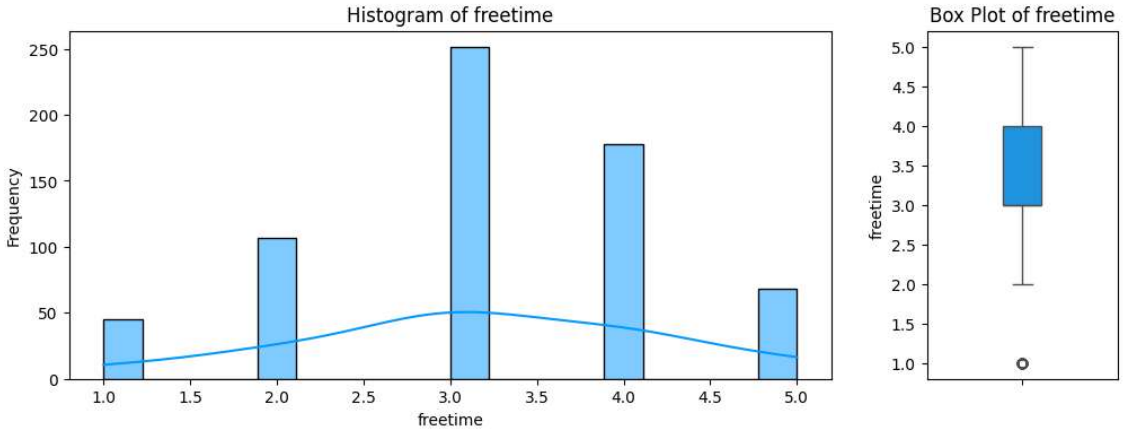


Figure 9 - Histogram and Box-plot of freetime

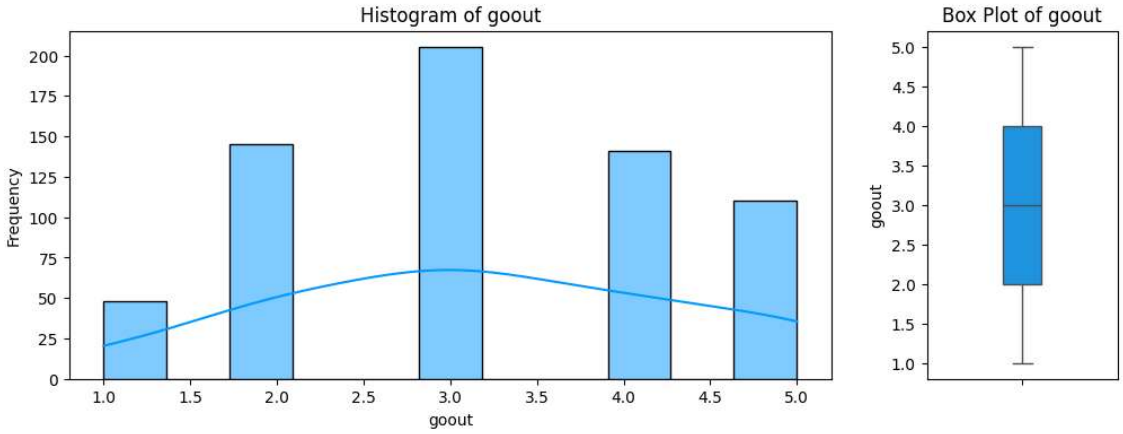


Figure 10 - Histogram and Box-plot of goout

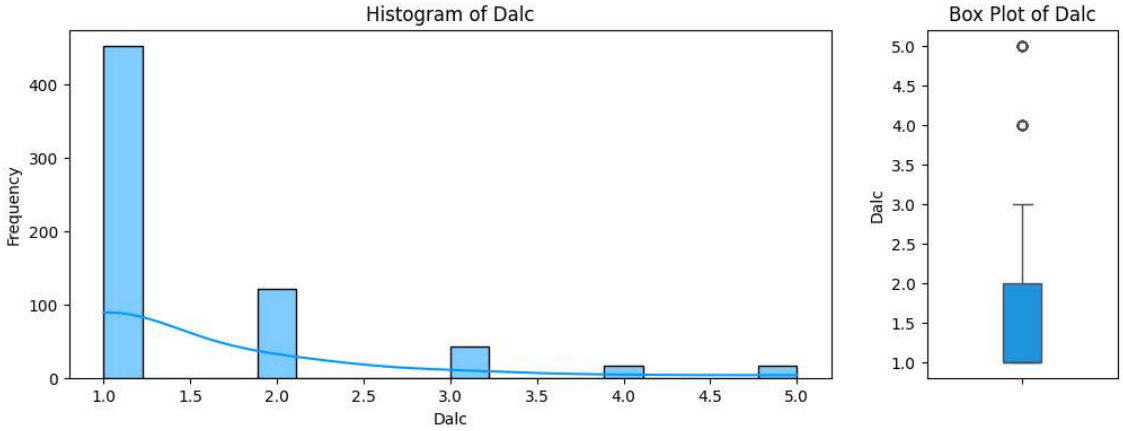


Figure 11 - Histogram and Box-plot of Dalc

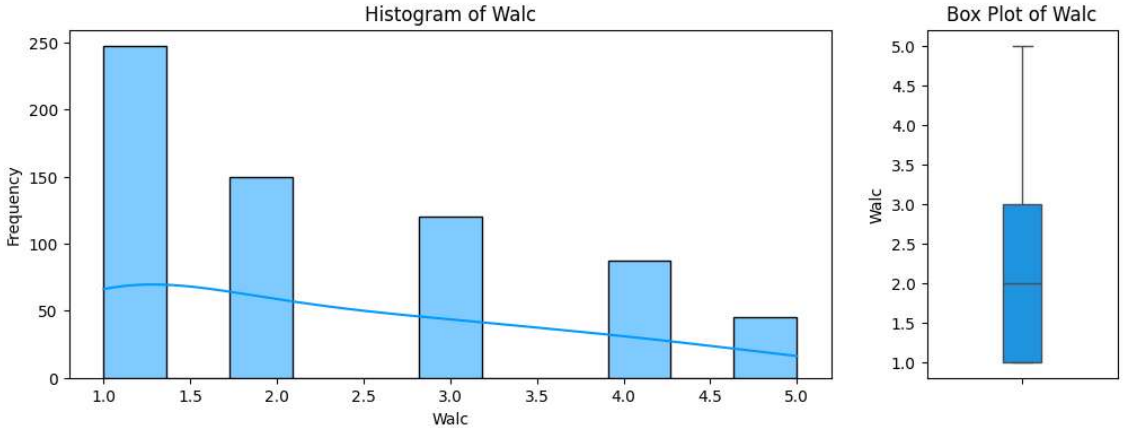


Figure 12 - Histogram and Box-plot of Walc

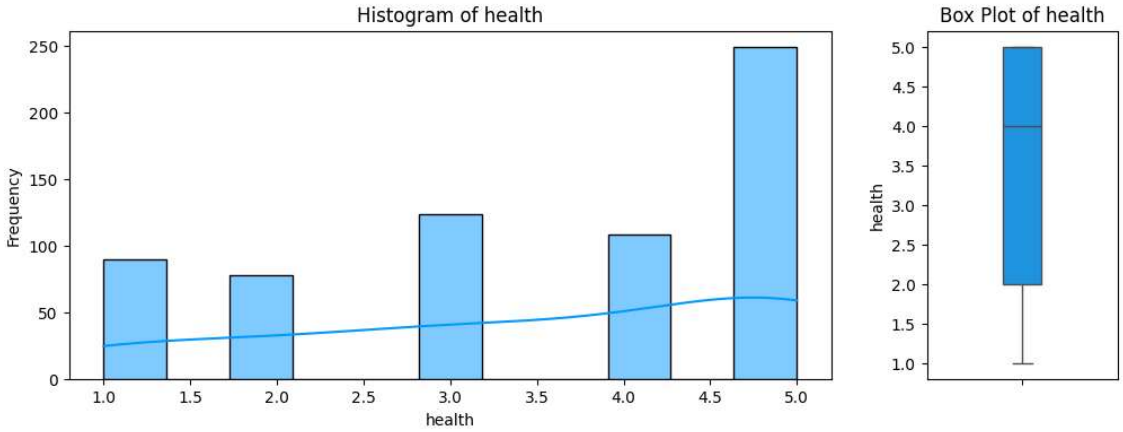


Figure 13 - Histogram and Box-plot of health

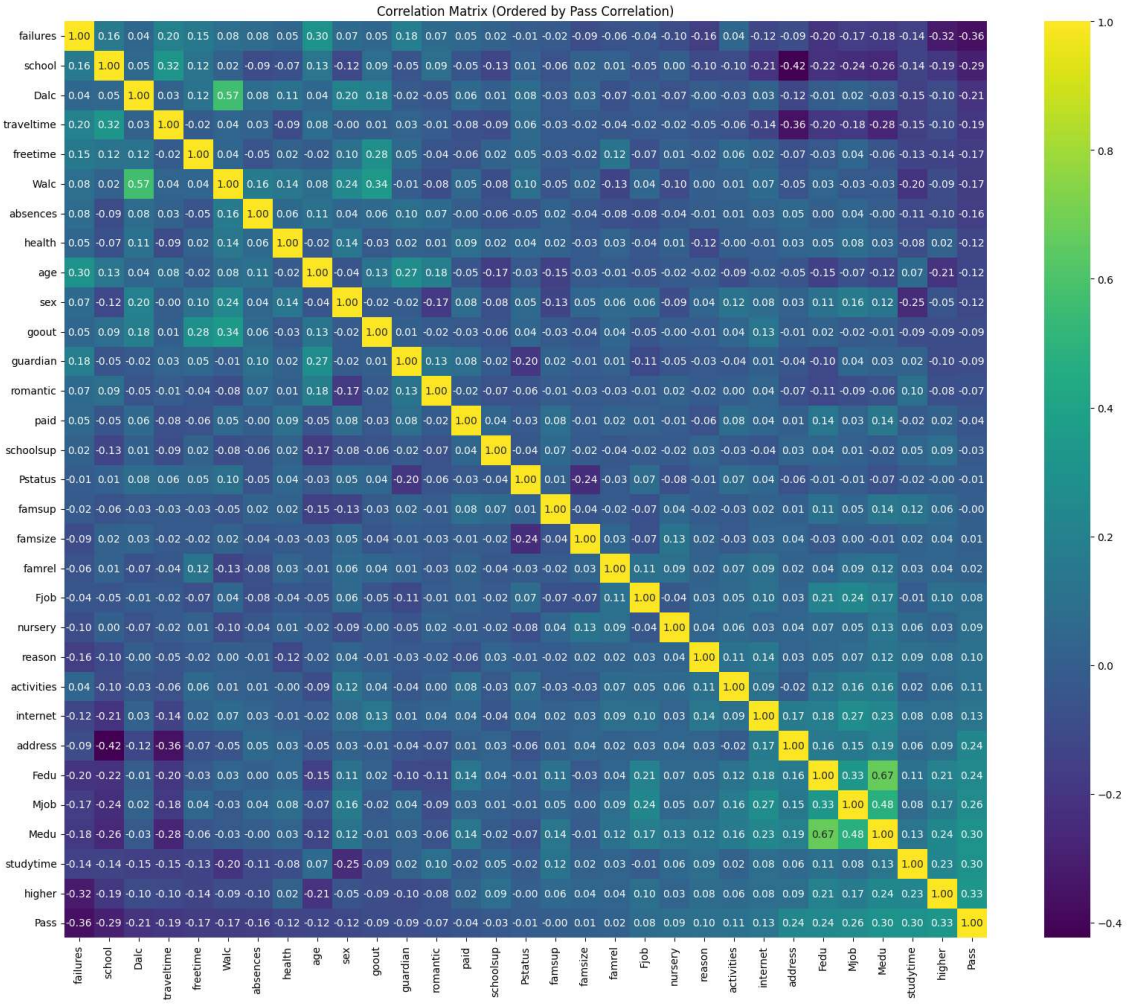


Figure 14 - Correlation Matrix ordered by Pass correlation