A picture containing text

Description automatically generated

**Coursework Final**

**MSc Data Science**

**Student ID:2325490**

**Date: 12/05/2023**

**DS7003 Advanced Decision Making.**

**Name:** **SAMIDHA NEUPANE**

Table of Contents

[**Abstract** 3](#_Toc134823125)

[**Introductions** 3](#_Toc134823126)

[**Literature Review** 4](#_Toc134823127)

[**Methodology** 4](#_Toc134823128)

[**A.** **Data Set Details** 5](#_Toc134823129)

[**B.** **Data preprocessing** 5](#_Toc134823130)

[**C.** **Data set preparation and splitting for input to the model.** 9](#_Toc134823131)

[**D.** **Data Explorations (Exploratory data analysis)** 11](#_Toc134823132)

[**E.** **Hypothesis Definitions** 16](#_Toc134823133)

[**F.** **Model Development and Evaluations.** 18](#_Toc134823134)

[**G.** **Performing metric analysis and Discussion** 22](#_Toc134823135)

[**Limitation of this project** 23](#_Toc134823136)

[**Conclusion** 23](#_Toc134823137)

[**References** 24](#_Toc134823138)

[**Appendix** 25](#_Toc134823139)

[**1.** **Appendix for data processing and cleaning** 25](#_Toc134823140)

[**2.** **Appendix for Model building** 50](#_Toc134823141)

**Using Learning Analytics to Improve Student Performance: Predicting Weighted Scores Based on Engagement and Timeliness.**

## **Abstract**

This study examines the influence of several factors on the weighted score, including activity level, previous education experience, and many more. Results show that active student engagement and timely assignment submission are important for higher scores. The study also examines the performance of various regression models in predicting the weighted score and finds that the Random Forest model seems to have the highest accuracy and R-squared value. The analysis suggests that institutions can use these findings to develop strategies to improve student performance and make informed decisions about course content, evaluation procedures, and student support services.

## **Introductions**

Courses for students and assessments are critical components of the educational system. Courses are designed to provide students with information and skills that will help them succeed in their future careers, while assessments are intended to track their progress and comprehension of the course material. Students' learning and evaluation techniques may vary greatly depending on the course, school, and subject matter. Traditionally, students have learned through lectures, textbooks, and assignments. Learning analytics, a field that analyses student learning data using data mining and machine learning approaches, may also be utilized to give insights into student learning and evaluation. Learning analytics may assist in identifying areas of difficulty for students and providing individualized recommendations for development. It may also be used to assess the efficacy of various learning techniques and evaluation procedures. The data set includes information about students' demographic characteristics, academic background, and engagement with course materials, as well as their performance on various assessments and their final grades in the courses. There are many machine learning techniques that can be applied to this data set depending upon the specific research questions. Like classification which can predict the student performance based on demographic and course related features, Clustering that can identify the groups of student with similar behaviors or characteristics using unsupervised learning like k-means clustering or hierarchical clustering, association rule mining that can find the pattern in the data that can be used to improve student performance or course design, Time series analysis that can analyze the pattern in student behavior over time, and not the least recommendation systems that can suggest the relevant course materials or resources to student based on their demographic and interaction data, regression analysis can be used to model the relationship between the performance of student and other demographic and course-related features. For example, one could use regression analysis for the prediction of student's final grade based on their age, gender, previous education, number of clicks on course materials, and forum participation.

## **Literature Review**

(Pendyala, 2022) has applied several machine learning algorithms to predict students at risk of dropping out. The models used include Logistic Regression, Random Forest, Gradient Boosting, and XGBoost. Overall, the author has provided a thorough and well-documented approach to predicting at-risk students using machine learning techniques. (kaggle.com, n.d.) evaluates the performance of machine learning methods such as Random Forest, Logistic Regression, and XGBoost in predicting student outcomes. (kaggle.com, n.d.) uses various data visualization techniques and statistics to analyze the data and gain insights into the various features of the dataset. Topics such as student demographics, course modules, assessment scores, and the relationships between these features were experimented.

## **Methodology**

The very first step in the data analysis is to prepare data for model pipelining. Open learning analytic datasets are quite messy data sets. It has lots of missing values, duplicate columns, data type errors. To get good accuracy with model results and have meaningful output we need to have clean and good dataset with proper formats. In this project we are going to apply the different process and methods which are mentioned below:

* Data set Details
* Data preparations.
* Data set preparation and splitting for input to the model.
* Data Explorations (Exploratory data analysis)
* Hypothesis definition
* Model Development.
* Performing metric analysis.

Here we are going to explore all the files and tables including all variables to explore about the dataset. We are going to visualize the datasets variables and relationships.

### **Data Set Details**

Dataset link: <https://archive.ics.uci.edu/ml/datasets/Open+University+Learning+Analytics+dataset>. (analyse.kmi.open.ac.uk, n.d.)

This data set contains data about courses, students, and their interactions with Virtual Learning Environment for seven selected courses and more than 30000 students. This data set includes information about students' demographic characteristics, academic background, and engagement with course materials, as well as their performance on various assessments and their final grades in the courses.

Data set consists of various csv files which are mentioned below:

1. Assessments: it contains fields related to course and assessment criteria information.
2. Courses: It contains information about the course and its length.
3. student\_assesment: It contains the information for student assessment and its submission and scores.
4. StudentInfo: It is the csv file containing code module, code presentations, student id and their information like age, region, highest educations, etc.
5. StudentRegistration: It is a csv file with information for student registrations.
6. studentVle: It is a csv file that contains the information about student engagement with the course.

### **Data preprocessing**

In this process We are going to check every variable in useful files about missing values, duplicate values, data formats, data integrations.

1. **Assessment.csv**: This file gives an idea about the module-presentation assessments. Every presentation contains several assessments and exams. I have checked the missing values and duplicate values. There are some duplicate columns, but one student has more assessment and same with same student id number. Assessments IDs are given as integers which are not correct. IDs are categorical. I have included some of the data from assessment tables. Also, we can see some missing values in date columns and weights for every assessment don’t quite add up. Apart from module CCC, which has a weight of 300, and module GGG, which has a weight of 100, most module presentations have a total weight of 200. We need to fix this weight problem and it’s different with the GGG module as it doesn’t have any weights for CMA and TMA. We will just assign 100 total weights to TMA assignment because CMA assignment is frequently weight 0. In addition, we will check if assessments with column assessment id is in the student assessment table and get all the matching and missing values.

Graphical user interface, application

Description automatically generated

Fig (1.1): Assessments data set

A screenshot of a computer

Description automatically generated

Fig (1.2): Module GGG with all assessment types and its respective weights

1. **Student Assessments**: No results are recorded if the student does not give assessment. If the results of the assessments are not kept in the system, the final exam submissions are missing. The following columns are present in this file: **id\_assessment, id\_student, date\_submitted, is\_banked, score.** We have found that there are more assessments in the student assessments table than there are in the Assessments table. Null scores can be filled up with zeros since they might be read as not submitted. However, it seems odd that evaluations with null scores have documented submission dates. Result 1336190 and 1777834 both don't have date un-registration and 1777834 has no date registration too. It might be a clerical error.

A screenshot of a computer

Description automatically generated

Fig (1.3): Showing the NaN values in the score column.

1. **Student Information**: This file includes the students’ achievements and demographic data about them.

Graphical user interface, application, table

Description automatically generated

Fig (1.4): Student Information

1. **Course Information:** This contains the modules list and its information.
2. **Student Registrations**: This file contains information about students, its registrations process, and dates. Also, the date of unregistrations. This section is blank for students who successfully finished the course. In the studentInfo.csv file, the final\_result column's value for students who dropped out is Withdrawal. Here, we verified that the student assessments table had records for all student IDs listed in the Registration tables. Furthermore, we looked at whether any of the students listed in the Students Information field were included in the Assessment Results table. We also checked whether those above both results give the same student ID, and it turns out to be true. This is inconsistent because the date\_unregistration field ought to be filled in for all withdrew students. The Student Information data indicates that 4648 students have withdrawn, yet the Student\_registration table indicates that 4594 students have unregistered. 54 withdrew students are now left without a date to unregister.
3. **VLE resources:** The csv file includes details on the resources that are accessible through the VLE. These often include html pages, pdf files, etc. These resources are available to students online, and interactions with them are documented. We checked for null values and missing values.
4. **Student VLE interactions:** Information regarding how each student interacted with the VLE's materials is contained in the studentVle.csv file. The system most likely captures the clicks at various times on the same day, resulting in duplicates, therefore duplication is quite okay in this case.

### **Data set preparation and splitting for input to the model.**

#### **Data Cleaning and filling NaN values**

We are going to split the data set into 80-20 splits for regression analysis task. Again, after the data is split, we are going to fill some **NaN** values in the above data columns. To address missing values, the most frequent band for each region will be used to fill in null imd\_band values. Any students with null total click values did not interact with the VLE and will be replaced with 0s. For students with NaN late submission rates or failing rates, it will be assumed that they did not submit any assignments, and their rates will be set to 100%. The un-registration date metric has been removed, as it duplicates information in the Withdrawal column. For regression analysis, withdrawals and failures cannot be distinguished due to the continuous nature of the target variable, so all withdrawals will be treated as failures with a score less than 40%.

#### **Feature Engineering**

**Activity\_level\_score:** This is the new column created to measure the activity level of each student within the different course module and presentations. We have assigned the weights to each activity type considering some importance given. For example, activity type “quiz” and “forumg” have a little more weights than “homepages”. Again, we have multiplied it with the “sum\_click” column to calculate the relative weight per module and per student.



Table (1): Activity\_level\_score calculations

#### **Data Integrations**

Next step in the data processing is the merging of different tables from above processed results. We have joined all the tables on various conditions. At first, we joined Student\_assessments table and assessment table with inner join on id\_assessment column and call it Table1. Most final examinations were absent from the Student Assessments table, which was taken into consideration when calculating the overall weight of all modules. assignment's score and weight were added together. Used the sum function to combine the data frame for each weight\*score, module, and presentation. And calculated each module's total recorded weight. Dividing the sum of the weight and the score by the module's total recorded weight gives the weighted scores. Again, we joined the student information, student\_Registrations and Courses table with inner join on code\_module, code\_presentation and student\_id column and call it as a table2. At last, we have joined VLE and StudenVLE table on code\_module, id\_site and code\_presentation column. Before joining these two VLE tables we found that the resources without a task for any student offer no information, total clicks per student per module presentation was determined and call it Table3. After merging all these three tables we got the final data frame which is given below.



Table (1.2): Weighted\_score per module, per presentations and per student

A screenshot of a computer

Description automatically generated with medium confidence

Fig (1.5): Final data after integration.

### **Data Explorations (Exploratory data analysis)**

**Training data descriptions – summary statistics for variables.**

**A screenshot of a computer

Description automatically generated with medium confidence**

Table (1.3): Summary statistics for train dataset

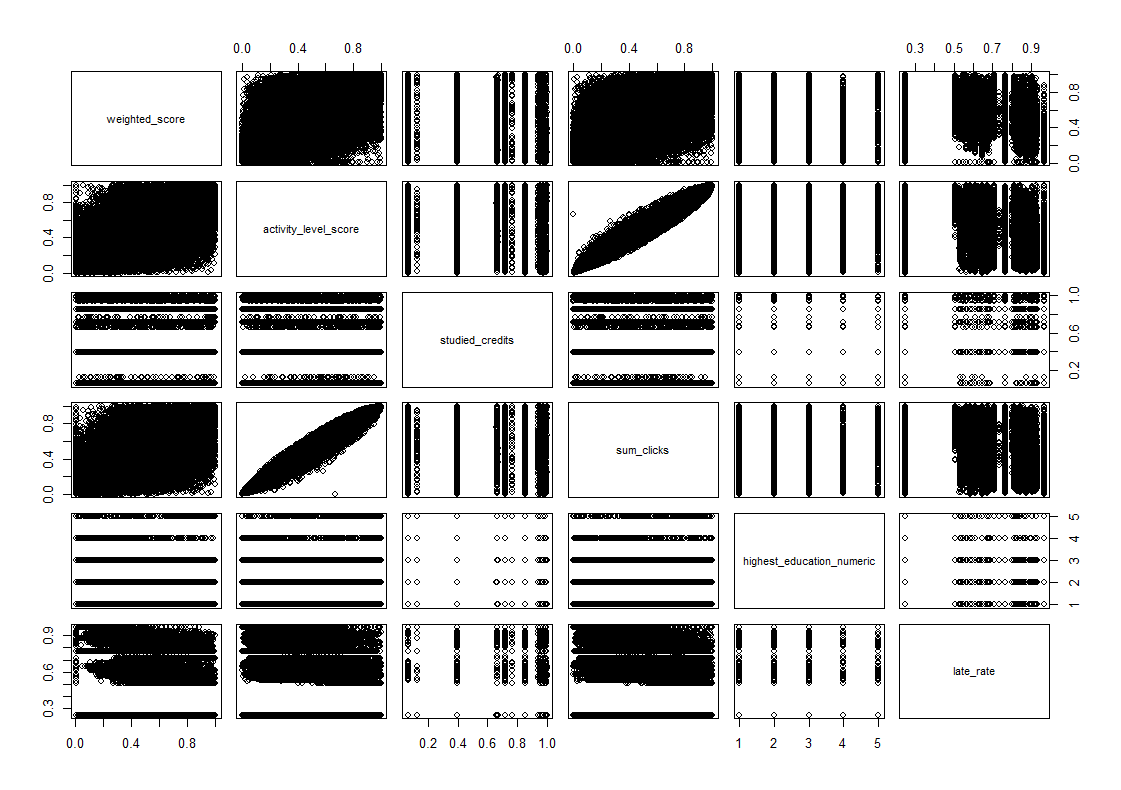
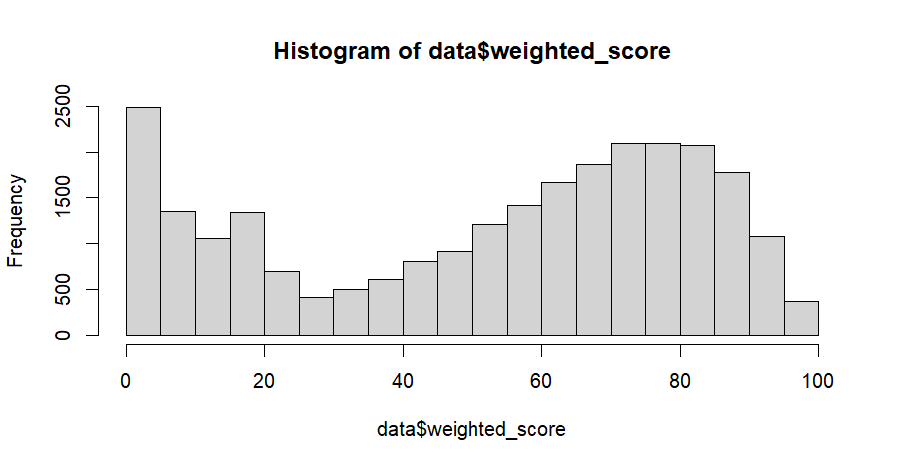


Fig (1.6): Scatter plot of dependent and independent variables.

  
Fig (1.7): Histogram of dependent variables: weighted\_score before normalizations.

In this dataset, there are several skewed variables. Considering normal distributions, linear models should be used with caution.

1. **Data Normalizations**

Normalization is a method of adjusting numerical variables in a way that they fall into a uniform range. It is especially beneficial when dealing with variables that have distinct units of measurement or scales. There are several techniques available to perform normalization such as min-max, Z-score, and SoftMax normalization. However, during this project, we experimented with various normalization methods and found that the rank-based normalization algorithm was the most effective for our data. Initially, we identified outliers by applying the Z-score method and then removed them from the dataset. Afterward, we applied rank-based normalization, and to evaluate the normality of our data, we used the KS-test.

The first step in every task after normalizing and preparing data is to calculate the correlation between variables.

**Results from KS-test**

****

Table (1.4): KS-test for normality check for variables

**Boxplot for dependent variable after normalizations.**

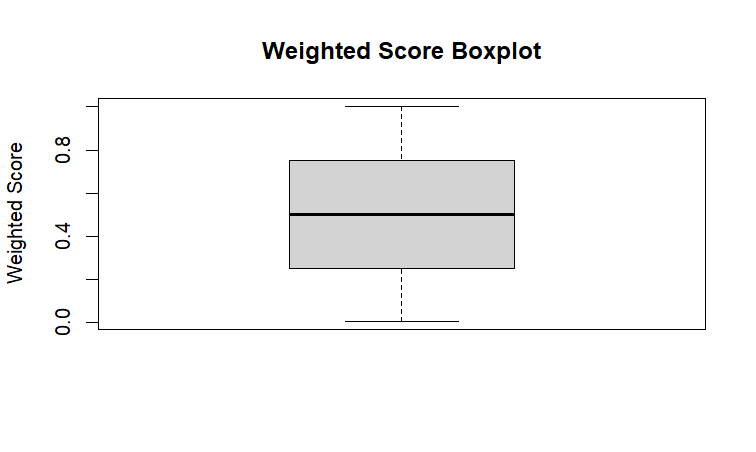
****

Fig (1.8): Boxplot of weighted\_score

**Boxplot for independent variables after normalizations.**

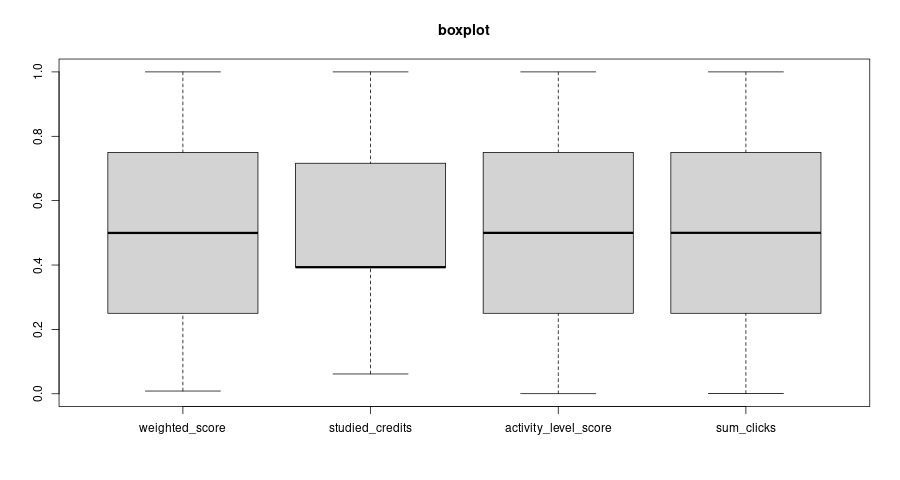


Fig (1.9): Boxplot of independent variables and dependent variables.

**Code module, code presentations, Gender, and Region wise representations of students**

Graphical user interface, website

Description automatically generated

**Students’ distributions for previous qualifications, imb band, Age, and Disability**

Graphical user interface, website

Description automatically generated

1. **Correlation between Variables**

A screenshot of a graph

Description automatically generated with low confidence

Fig (2): correlation heatmap of every variable from dataset

A picture containing text, screenshot, diagram, design

Description automatically generated

Fig (2.1): Correlation matrix of variables independent variable with dependent variable.

Based on the correlation matrix provided, it is apparent that the dependent variable, weighted\_score\_rinorm, has a positive correlation with activity\_level\_score\_rinorm, sum\_clicks\_rinorm, and highest\_education\_numeric\_rinorm. Conversely, it has a negative correlation with late\_rate\_rinorm.

1. **VIF Calculations**

> print(vif\_values)

sum\_clicks\_rinorm highest\_education\_numeric\_rinorm

1.085043 1.001705

late\_rate\_rinorm

1.083295

Multicollinearity check between independent variables

These VIF values suggest that there is no serious multicollinearity issue between the independent variables. Generally, a VIF value greater than 5 is considered as an indication of multicollinearity issue. In our case, all the VIF values are less than 5, which is a good sign, indicating that the selected independent variables are not highly correlated. From the analysis we have taken the variables like weighted\_score, late\_rate, highest\_education, studied\_credits, final\_result, activity\_level\_score, sum\_clicks variables. We have checked the correlation values, scatterplot and VIF values for independent variables multicollinearity.­

### **Hypothesis Definitions**

In this project we are going to define the problem/hypothesis. The hypothesis is that the weighted score is influenced by several factors including the activity level, previous education experience, course module, studied credit, and late rate. By analyzing the relationships among these variables, we can make predictions about the expected weighted score given a certain combination of the independent variables. For example, we can determine which combination of independent variables is most likely to result in a high or low weighted score, and how much impact each variable has on the overall score. This information can be useful for predicting student performance and identifying areas where interventions may be needed to improve outcomes.

### **Model Development and Evaluations.**

In this section we have tried different machine learning models. We have developed various regression modeling techniques. 80% of our datasets were used for training the model and 20% for testing. We have experimented with 5 different models which results and assumptions are discussed below.  
Rechecking the null values again before proceeding for model development.

A screenshot of a computer

Description automatically generated with medium confidence

Fig (2.2): Rechecking for null values

#### **Multiple Linear Regression**

Multiple linear regression is a statistical approach for modelling the relationship between two or more predictor variables and a response variable. It assumes a linear relationship between the predictors and the response and estimates the model parameters using the least squares approach.

> MAE <- mean(abs(data.test$weighted\_score - predictions))  
> # Print the model performance metrics  
> cat("RMSE:", RMSE, "\n")  
RMSE: 0.2130817  
> cat("R-squared:", R\_squared, "\n")  
R-squared: 0.491245  
> cat("MAE:", round(MAE, 2), "\n")  
MAE: 0.17

Fig (2.3): Training metrics output from Multiple linear regression

#### **Random Forest Model**

Random Forest is an ensemble learning method that constructs a multitude of decision trees and combines their predictions to improve the overall accuracy and reduce overfitting. It is commonly used in classification and regression tasks and has shown high performance in various applications.

1. With 10% data

> model.rf

Random Forest

1724 samples

4 predictor

No pre-processing

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 1724, 1724, 1724, 1724, 1724, 1724, ...

Resampling results across tuning parameters:

mtry RMSE Rsquared MAE

2 0.7475296 0.3574576 0.5788954

3 0.7576527 0.3456141 0.5862859

4 0.7640114 0.3392965 0.5906203

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was mtry = 2.

1. With 80% data: In this case we have taken 80% for training.

A screenshot of a computer

Description automatically generated with medium confidence

Fig (2.6): Random Forest model.

A screenshot of a computer

Description automatically generated with medium confidence

Fig (2.5): Random Forest model: Precision, Recall and F1 score.

#### **Ridge Regression**

Ridge regression is a type of linear regression that adds a penalty term to the cost function to address multicollinearity, which occurs when independent variables are highly correlated. This helps to improve the model's stability and reduce overfitting.

A screenshot of a computer

Description automatically generated

Fig (2.7): MAE, RMSE AND R-squared value for ridge regression

A screenshot of a computer

Description automatically generated

Fig (2.8): Precision, recall and F1score value from ridge regression.

#### **Lasso Regression**

Lasso regression is a type of linear regression that uses L1 regularization to reduce the impact of irrelevant features, thus improving model interpretability and performance.

A screenshot of a computer

Description automatically generated

Fig (2.9): Metrics: MAE, RMSE, R-squared, precision, recall, f1-score form Lasso Regression.

#### **Support Vector Machine**

Support Vector Machine (SVM) regression is a powerful algorithm that can handle non-linear relationships between features and target variables. It works by finding the optimal hyperplane that maximizes the margin between the support vectors and the decision boundary.

**A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated**

Fig (3): SVM model Fig (3.1): SVM metrics

Comparative Table for 5 models

 Table (1.5): Metrics comparison of 5 different models.

### **Performing metric analysis and Discussion**

Looking at the results of the models, we can see that the linear regression model has a very low accuracy of 0.17 and low R-squared value of 0.21, indicating that it may not be the best model to predict the weighted score. On the other hand, the Random Forest model has the highest accuracy of 0.14 and R-squared value of 0.78, indicating that it may be the most appropriate model to use for predicting the weighted score. When examining the correlation matrix, we can see that the dependent variable, weighted\_score, has the strongest positive correlation with activity\_level\_score and sum\_clicks, indicating that these variables have a greater influence on the weighted\_score. Additionally, we can see that the weighted\_score has a negative correlation with late\_rate, indicating that as the late\_rate increases, the weighted\_score decreases. Other variables such as studied\_credits and highest\_education\_numeric also have a moderate positive correlation with the weighted\_score, suggesting that these factors could also be important in predicting the weighted\_score. The linear regression model showed the lowest accuracy and R-squared value, indicating that it may not be the best model for this dataset.

The Ridge regression and SVM models showed similar performance in terms of accuracy and R-squared value. However, the SVM model had a slightly lower MAE and RMSE value, suggesting that it may be slightly better at predicting the target variable. The Lasso regression model had a similar performance to the Ridge regression and SVM models in terms of accuracy and R-squared value, but it had the lowest precision value. This means that it may not be the best model for identifying true positive cases. The Random Forest model had the highest accuracy and R-squared value among all the models, indicating that it may be the best model for this dataset. However, it had the highest RMSE value, indicating that its predictions may have higher errors than the other models. Overall, these insights suggest that students' engagement levels and timeliness in submitting their work may be important factors to consider when predicting their weighted scores. The Random Forest model may be the most suitable for predicting these scores, but the Lasso model could also provide valuable insights.

## **Limitation of this project**

We have huge data. We were able to train model up to 30-80% of data only. Also, there are lots of algorithms that implement folding and using data multiple times to make models more robust but, in our case, due to resources and time limitations we are not able to achieve high performing model. To make these models best there are techniques like fine-tuning the hyperparameters, increasing the size or quality of the dataset, using feature engineering, trying a different model architecture, Ensemble mismodelling. In some extent we have tried by using feature engineering like adding “activity\_level\_score” column in data set, using the dataset up to 80% of whole dataset (contains more than seventeen thousand rows).

## **Conclusion**

Learning analytics can provide valuable insights into student learning and evaluation, and the weighted score is influenced by factors such as student engagement, timeliness of assignment submission, and educational background. The Random Forest model was found to be the most suitable for predicting weighted scores. Students' engagement levels and timeliness in submitting their work were identified as important factors in predicting their weighted scores. The linear regression model was found to be the least suitable for predicting weighted scores. The SVM model had a slightly lower MAE and RMSE value than the Ridge regression model, and the Lasso model had the lowest precision value. Institutions can use these findings to develop strategies to improve student performance and make informed decisions about course content, evaluation procedures, and student support services.

## **References**

1. Kuzilek J., Hlosta M., Zdrahal Z. [Open University Learning Analytics dataset](https://www.nature.com/articles/sdata2017171)Sci. Data 4:170171 doi: 10.1038/sdata.2017.171 (2017).
2. kaggle.com. (n.d.). *VLE-DT*. [online] Available at: https://www.kaggle.com/code/chounta/vle-dt [Accessed 11 May 2023].
3. Pendyala, H. (2022). *LearningAnalytics\_Predicting\_students\_at-risk*. [online] GitHub. Available at: https://github.com/Heramb001/LearningAnalytics\_Predicting\_students\_at-risk [Accessed 11 May 2023].
4. kaggle.com. (n.d.). *OULAD - Exploratory Data Analysis*. [online] Available at: https://www.kaggle.com/code/wanderscheidegger/oulad-exploratory-data-analysis [Accessed 12 May 2023].
5. K, G.M. (2020). *Machine Learning Basics: Decision Tree Regression*. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda.
6. Brownlee, J. (2021). *Regression Metrics for Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/regression-metrics-for-machine-learning/.
7. ‌Agrawal, R. (2021). *Evaluation Metrics for Your Regression Model*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/.
8. ‌Wikipedia Contributors (2019). *Regression analysis*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Regression\_analysis.
9. Uyanık, G.K. and Güler, N. (2013). A Study on Multiple Linear Regression Analysis. *Procedia - Social and Behavioral Sciences*, 106(1), pp.234–240. doi:https://doi.org/10.1016/j.sbspro.2013.12.027.
10. Ritvo, A.R., Volkmar, F.R., Lionello-Denolf, K.M., Spencer, T.D., Todd, J., Yirmiya, N., Yaari, M., Newman, D.B., Lestrud, M., Ritvo, E., Dipuglia, A., Ness, M., Lyons, M., Moyle, M., Long, S., Moyle, M., Manjarrés, M., Allen, E., Murdoch, J.D. and Walker, M.F. (2013). Regression Analysis. *Encyclopedia of Autism Spectrum Disorders*, pp.2538–2538. doi:https://doi.org/10.1007/978-1-4419-1698-3\_251.

## **Appendix**

### **Appendix for data processing and cleaning**

# set the working directory to where the CSV file is located

setwd(dirname(file.choose()))

getwd()

#------------ load the data sets to to data Preprocessing -----------------#

# load the CSV file using the read.csv() function

assesments <- read.csv("assessments.csv")

course <- read.csv("courses.csv")

std\_asses <- read.csv("studentAssessment.csv")

std\_info <- read.csv("studentInfo.csv")

std\_regs <- read.csv("studentRegistration.csv")

std\_Vle <- read.csv("studentVle.csv")

Vle <- read.csv("vle.csv")

# check for missing values using is.na() function

missing\_values <- is.na(assesments)

# count the number of missing value

col\_missing <- colSums(missing\_values)

col\_missing[col\_missing > 0]

*# Convert to factors*

assesments$id\_assessment <- as.factor(assesments$id\_assessment)

std\_asses$id\_assessment <- as.factor(std\_asses$id\_assessment)

#---------------ANALYZING TABLES AND JOINING TABLES-------#

#checking the weights of CMA, TMA and Exam

filtered\_data\_CMA <- assesments %>% filter(assessment\_type == "CMA")

filtered\_data\_TMA <- assesments %>% filter(assessment\_type == "TMA")

filtered\_data\_Exam <- assesments %>% filter(assessment\_type == "Exam")

# grouping the data by code\_module and code\_presentation

#and summarizing the weight column by summing it

grouped\_data <- assesments %>%

group\_by(code\_module, code\_presentation) %>%

summarize(total\_weight = sum(weight))

# for exam only calculate and group by

# select rows where assessment\_type is "Exam",

#group by code\_module and code\_presentation,

#and summarize the weight column by summing it

grouped\_data\_exam <- assesments %>%

filter(assessment\_type == "Exam") %>%

group\_by(code\_module, code\_presentation) %>%

summarize(total\_weight = sum(weight))

# we saw that CCC has 2 exams from the group\_data1 table

#for assesments only group by and aggregate

grouped\_data\_Ass <- assesments %>%

filter(assessment\_type != "Exam") %>%

group\_by(code\_module, code\_presentation) %>%

summarize(total\_weight = sum(weight))

grouped\_data\_Ass

# we saw GGG has 0 weights this explains this does not have any assignments

#Again, reorganizing to see from different perspectives

grouped\_data\_GGG <- assesments %>%

filter(code\_module == "GGG") %>%

group\_by(code\_module, code\_presentation, assessment\_type) %>%

summarize(weight\_by\_type = sum(weight))

# values from this table shows that there are no assessments weights only exam weights are present

# filter the data to select only rows where the assessment\_type is CMA or TMA and the weight is 0

#also exclude GGG as we know the result

filtered\_data\_CandT <- assesments %>% filter(assessment\_type %in% c("CMA", "TMA"), weight == 0, code\_module != "GGG")

##### To find the total number of assignment in GGG module

# filter the data to select only rows where the code\_module is GGG

filtered\_data\_GGG <- assesments %>% filter(code\_module == "GGG")

# group the data by code\_module, code\_presentation, assessment\_type, and id\_assessment, and count the number of occurrences of each id\_assessment

grouped\_data <- filtered\_data\_GGG %>% group\_by(code\_module, code\_presentation, assessment\_type, id\_assessment) %>% summarize(count = n())

# --------------assigning the weight to TMA and CMA and GGG module-------------------#

assesments$weight[(assesments$code\_module == 'GGG') & (assesments$assessment\_type == 'TMA')] <- (100/3)

assesments$weight[(assesments$code\_module == 'GGG') & (assesments$assessment\_type == 'CMA')] <- (0)

assesments

#Check if Assessments information are in Student Assessments table

# Subset the data and extract the "weight" column

column\_CMA <- assesments[assesments$assessment\_type == "CMA" & assesments$code\_module != "GGG", "weight"]

counts <- table(column\_CMA)

result\_CMA <- as.data.frame(counts)

colnames(result\_CMA) <- c("Weight", "Count")

result\_CMA

column\_TMA <- assesments[(assesments$assessment\_type == "TMA") & (assesments$code\_module != "GGG"), "weight"]

counts <- table(column\_TMA)

result\_TMA <- as.data.frame(counts)

colnames(result\_TMA) <- c("Weight", "Count")

result\_TMA

# From here we are going to calculate the shared value between

#two dataframe with the same column name

Comp\_columns <- function(df1, df2) {

# Get the names of the columns in each dataframe

df1Columns <- colnames(df1)

df2Columns <- colnames(df2)

# Find the columns that are shared between the two dataframes

diffList <- intersect(df1Columns, df2Columns)

cat("Shared columns: ", diffList, "\n")

# Check if the values in the shared columns match

for(col in diffList) {

x <- table(df1[,col] %in% df2[,col])

cat("Check if values are present in both dataframes for column ", col, ":\n")

print(x)

cat("\n")

}

}

# find the values present in one dataframe and not present not in the other

find\_diff\_values <- function(df1, df2, col) {

# Find all unique values of col in df1 and df2

df1\_IDs <- unique(df1[[col]])

df2\_IDs <- unique(df2[[col]])

# Find values in df1 that are not in df2

diff <- setdiff(df1\_IDs, df2\_IDs)

# Count how many are different

numberDiff <- length(diff)

cat(paste("Values from df1 not in df2: ", toString(diff), "\n"))

cat(paste("Number of missing values: ", numberDiff, "\n"))

}

print\_diff\_values <- function(df1, df2, col) {

# Pull out all unique values of col

df1\_IDs <- unique(df1[[col]])

df2\_IDs <- unique(df2[[col]])

# Compare the two lists

diff <- setdiff(df1\_IDs, df2\_IDs)

# (a) Make a list of missing values

missingList <- as.list(diff)

# (b) Find these IDs in df2

missingDf <- df1[df1[[col]] %in% missingList,]

return(missingDf)

}

Comp\_columns(assesments, std\_asses)

find\_diff\_values(assesments, std\_asses, "id\_assessment")

print\_diff\_values(assesments, std\_asses, 'id\_assessment')

# now we need to calculate which student ids are missing with the id\_assessment missing we got

# List of ids missing

missing\_Lists <- c(30723, 1763, 34885, 15014, 37444, 14990, 30713, 37424, 15025, 34898, 37434, 40087, 34872, 40088, 15002, 1757, 30718, 34911)

# From assessments table find all the rows with weight 100

weight\_with100 <- assesments[assesments$weight == 100, ]

# Get all unique assessment IDs

List\_weight\_with100 <- unique(weight\_with100$id\_assessment)

# Compare this list with the list of all assessment IDs missing from student assessment table

compare\_value <- setdiff(List\_weight\_with100, missing\_Lists)

number\_Compare <- length(compare\_value)

cat("100 weighted assessments in the student assessment able (that are not missing exams): ", compare\_value, "\n")

cat("Number of 100 weighted assessments (that are not missing exams) in the studen assessment table: ", number\_Compare, "\n")

# (a) Make a list of IDs to look

matched\_list <- c(24290, 25354, 24299, 25361, 25368, 25340)

# (b) Find these IDs in the Assessments table

matched\_Df <- assesments[assesments$id\_assessment %in% matched\_list, ]

matched\_Df

# STUDENT ASSESSMENT INFORMATIONS.

# convert id\_assessment column to factors using as.factor() function

std\_asses$id\_assessment <- as.factor(std\_asses$id\_assessment)

std\_asses$id\_student <- as.factor(std\_asses$id\_student)

std\_asses[is.na(std\_asses$score),]

std\_asses$score[is.na(std\_asses$score)] <- 0

head(std\_asses)

#STUDENT REGISTRATION TABLES

#We need to check if all the students id are in student result tables

Comp\_columns(std\_regs, std\_asses)

# 5847 students are missing

# we need to check if any student information are missing from student assessment tables

Comp\_columns(std\_info, std\_asses)

# 5847 students are also missing from student assessments tables too

# we need to check if these are same students

#calculate all the unique value

# Pull out all unique values id\_assessments

id\_regs <- unique(std\_regs$id\_student)

id\_info <- unique(std\_info$id\_student)

# Compare the two lists

diff <- setdiff(id\_regs, id\_info)

# (b) Count how many are different

numberDiff <- length(diff)

numberDiff

# we found there are no same ids

Comp\_columns(std\_regs, std\_info)

# these are same students

#checking student information not present in student assessments table

not\_in\_stdAss = print\_diff\_values(std\_info, std\_asses, 'id\_student')

head(not\_in\_stdAss)

# What are their final results?

column <- not\_in\_stdAss$final\_result

result\_counts <- table(column)

#student registration not present in student assessments table

not\_in\_stdAss\_from\_StdReg = print\_diff\_values(std\_regs, std\_asses, 'id\_student')

head(not\_in\_stdAss\_from\_StdReg)

#4594 students have unregistered

sum(!is.na(not\_in\_stdAss\_from\_StdReg$date\_unregistration))

#calculating the 2 pass students date\_unregistration status

# Show rows with passes

#take the columns with pass and pass it through not\_instdAss list to get the 2 student ids

not\_in\_stdAss[not\_in\_stdAss$final\_result == 'Pass', ]

#Then check indivisually those ids unregistration date

not\_in\_stdAss\_from\_StdReg[not\_in\_stdAss\_from\_StdReg$id\_student=='1336190',]

not\_in\_stdAss\_from\_StdReg[not\_in\_stdAss\_from\_StdReg$id\_student=='1777834',]

#Result 1336190 and 1777834 both don't have date unregistration and 1777834 has no date registration too.

#It might be clerical error

#VIRTUAL LEARNING ENVIRONMENT RESOURCES

Vle\_nunique <- sapply(Vle, function(x) n\_distinct(x, na.rm = TRUE))

Vle\_nunique

str(Vle)

Vle$id\_site<- as.factor(Vle$id\_site)

std\_Vle$id\_site<- as.factor(std\_Vle$id\_site)

std\_Vle$id\_student<- as.factor(std\_Vle$id\_student)

std\_info$id\_student<- as.factor(std\_info$id\_student)

#----------------------- preparing tables integration ----------------------------#

#FROM HERE WE ARE GOING TO TEST HYPOTHESIS

#There is a correlation between the activity

#level of students (such as time spent on learning activities,

#number of activities completed, etc.) and their final assessment scores.

#CHECKING IF activity\_type present in studentVle from Vle table

Comp\_columns(Vle, std\_Vle)

diff\_value\_data<-print\_diff\_values(Vle, std\_Vle, 'id\_site')

# Inner join Vle and stdVle tables on id\_site

ac\_data <- inner\_join(std\_Vle, Vle, by = "id\_site") %>%

select(id\_site, code\_module.x, code\_presentation.x, id\_student, activity\_type, date, sum\_click)

head(ac\_data)

# define the weightage for each activity type

activity\_weights <- c(resource = 0.2,

oucontent = 0.3,

url = 0.1, homepage = 0.1,

subpage = 0.1,

glossary = 0.1,

forumng = 0.1,

oucollaborate = 0.05,

dataplus = 0.05,

quiz = 0.2,

ouelluminate = 0.05,

sharedsubpage = 0.05,

questionnaire = 0.05,

page = 0.05,

externalquiz = 0.1,

ouwiki = 0.05,

dualpane = 0.05,

repeatactivity = 0.05,

folder = 0.05,

htmlactivity = 0.05)

#best way to calculate the activity\_level

activity\_level\_data <- ac\_data %>%

group\_by(code\_module.x, code\_presentation.x, id\_student, activity\_type) %>%

summarize(sum\_clicks = sum(sum\_click)) %>%

mutate(weight = activity\_weights[activity\_type]) %>%

group\_by(code\_module.x, code\_presentation.x, id\_student) %>%

summarize(activity\_level\_score = sum(sum\_clicks \* weight) / sum(weight),

sum\_clicks = sum(sum\_clicks))

head(activity\_level\_data)

#Data integrations steps

#joining student registration student informations and courses tables together

Comp\_columns(std\_regs, course)

#join student registration and course table with inner join

std\_reg\_Andcourse <- inner\_join(std\_regs, course, by = c("code\_module", "code\_presentation"))

std\_reg\_Andcourse$id\_student<- as.factor(std\_reg\_Andcourse$id\_student)

#join the above table with student\_info table

std\_reg\_Andcourse\_Info <- inner\_join(std\_reg\_Andcourse, std\_info, by = c("code\_module", "code\_presentation", 'id\_student'))

#now join Assessment table and student assessment result tables

Ass\_StdAssResult<- inner\_join(assesments, std\_asses, by = c('id\_assessment'))

# calculating weighted score for final score

# Make a copy of dataset

New\_score\_ass <- Ass\_StdAssResult

# Count how many exams there are in Results for every module presentation

New\_score\_ass[New\_score\_ass$assessment\_type == 'Exam', c('code\_module', 'code\_presentation', 'id\_assessment')] %>%

group\_by(code\_module, code\_presentation) %>%

summarise(n\_exams = n\_distinct(id\_assessment))

New\_score\_ass$score <- as.integer(New\_score\_ass$score)

# create acolumn by multiplying score and weight.

New\_score\_ass <- New\_score\_ass %>% mutate(weight\_score = weight \* score)

# Aggregate recorded weight\*score per student per module presentation

sum\_scores <- New\_score\_ass %>%

group\_by(id\_student, code\_module, code\_presentation) %>%

summarise(weightByScore = sum(weight\_score)) %>%

ungroup()

# Calculate total recorded weight of module

# Get total weight of modules

total\_weight <- assesments %>%

group\_by(code\_module, code\_presentation) %>%

summarise(total\_weight = sum(weight)) %>%

ungroup()

# Subtract 100 to account for missing exams

total\_weight$total\_weight <- total\_weight$total\_weight - 100

# Mark module DDD as having 200 credits

total\_weight$total\_weight[total\_weight$code\_module == 'DDD'] <- 200

#-----------------Calculate weighted score--------------------#

#Merge sum\_scores and total\_weight tables

score\_weights <- inner\_join(sum\_scores, total\_weight, by = c("code\_module", "code\_presentation"))

# (b) Calculate weighted score

score\_weights$weighted\_score <- score\_weights$weightByScore / score\_weights$total\_weight

# (c) Drop helper columns

score\_weights <- score\_weights %>% select(-c(weightByScore, total\_weight))

head(score\_weights)

################################# calculation of late submission ####################################

str(Ass\_StdAssResult)

names(Ass\_StdAssResult)

#need to convert char to integer for date column

Ass\_StdAssResult$date <- as.integer(Ass\_StdAssResult$date)

# Calculate the difference between the submission dates

late-Submission <- Ass\_StdAssResult %>%

mutate(submission\_days = date\_submitted - date)

# Make a column indicating if the submission was late or not

lateSubmission <- lateSubmission %>%

mutate(late\_submission = submission\_days > 0)

head(lateSubmission)

#late submission for exams

lateSubmission[lateSubmission$assessment\_type == 'Exam' & lateSubmission$late\_submission == TRUE, ]

#----------------- per student per module presentation-------------------#

total\_late\_per\_student <- lateSubmission %>%

group\_by(id\_student, code\_module, code\_presentation) %>%

summarize(total\_late\_submission = sum(late\_submission))

# late submission per assessment per module per code presentation

lateSubmission <- lateSubmission %>%

group\_by(id\_student, code\_module, code\_presentation, id\_assessment) %>%

summarise(total\_late\_submission = sum(late\_submission))

total\_count\_assessments <- lateSubmission %>%

group\_by(id\_student, code\_module, code\_presentation) %>%

summarize(total\_assessments = n()) %>%

ungroup()

head(total\_count\_assessments)

# data merging

late\_rate\_per\_student <- merge(total\_late\_per\_student, total\_count\_assessments,

by=c('id\_student', 'code\_module', 'code\_presentation'), all=TRUE)

late\_rate\_per\_student$late\_rate <- late\_rate\_per\_student$total\_late\_submission / late\_rate\_per\_student$total\_assessments

late\_rate\_per\_student <- late\_rate\_per\_student[, !(names(late\_rate\_per\_student) %in% c("total\_late\_submission", "total\_assessments"))]

head(late\_rate\_per\_student)

# pass fail rate calculations

passRate <- Ass\_StdAssResult %>% mutate(fail = score < 40)

#aggregate per student per module

total\_fails\_per\_student <- passRate %>%

group\_by(id\_student, code\_module, code\_presentation) %>%

summarize(total\_fails = sum(fail, na.rm = TRUE)) %>%

ungroup()

# Merge df with total fails and total count assessments

fail\_rate\_per\_student <- merge(total\_fails\_per\_student, total\_count\_assessments, by=c('id\_student', 'code\_module', 'code\_presentation'))

# Make a new column with late submission rate

fail\_rate\_per\_student$fail\_rate <- fail\_rate\_per\_student$total\_fails / fail\_rate\_per\_student$total\_assessments

# Drop helper columns

fail\_rate\_per\_student <- fail\_rate\_per\_student[, !names(fail\_rate\_per\_student) %in% c('total\_fails', 'total\_assessments')]

######### Merge all the above tables Ass\_StdAssResult, weighted\_score, late\_submission\_rate and fail\_rate #####

# Merge score\_weights and late\_rate\_per\_student data frames

Asses <- merge(score\_weights, late\_rate\_per\_student, by=c("id\_student", "code\_module", "code\_presentation"))

# Merge fail\_rate\_per\_student data frame with the above merged data frame

Asses <- merge(Asses, fail\_rate\_per\_student, by=c("id\_student", "code\_module", "code\_presentation"))

#Merge all the tables

common\_cols <- intersect(names(std\_reg\_Andcourse\_Info), names(activity\_level\_data))

common\_cols

final\_merged\_file1 <- merge(std\_reg\_Andcourse\_Info, activity\_level\_data, by = c("id\_student", "code\_module", "code\_presentation"), all.x = TRUE)

head(final\_merged\_file1)

final\_merged\_file2 <- merge(Asses, final\_merged\_file1, by = c("id\_student", "code\_module", "code\_presentation"), all.x = TRUE)

head(final\_merged\_file2)

write.csv(final\_merged\_file2, file = "sampletable.csv", row.names = TRUE)

Whole\_final\_data<- final\_merged\_file2

head(Whole\_final\_data)

#------------------------------DATA CLEANING---------------------#

#check for null values

sapply(Whole\_final\_data, function(x) sum(is.na(x)))

Whole\_final\_data$date\_registration <- replace(Whole\_final\_data$date\_registration, Whole\_final\_data$date\_registration == "?", NA)

Whole\_final\_data$date\_unregistration <- replace(Whole\_final\_data$date\_unregistration, Whole\_final\_data$date\_unregistration == "?", NA)

Whole\_final\_data$imd\_band<- replace(Whole\_final\_data$imd\_band,Whole\_final\_data$imd\_band =="?", NA)

#data cleanings for further quality

sapply(train, function(x) sum(is.na(x)))

#sum\_click, activity\_level\_score, weighted\_score, late\_rate,imb\_band, date\_registration, date\_unregistration These have null values need to remove

#forimb\_band

#find out the most frequent imb\_band

# Find what is the most frequent band in each region

# Find the most frequent band in each region without missing values

regions\_list <- unique(Whole\_final\_data$region)

for (i in regions\_list) {

result <- table(Whole\_final\_data[Whole\_final\_data$region == i & !is.na(Whole\_final\_data$imd\_band), "imd\_band"])

max\_band <- names(result[result == max(result)])

print(paste(i, "IMD band: ", max\_band))

}

# Replace all null values with respective most frequent imd\_bands

regions\_list <- unique(Whole\_final\_data[Whole\_final\_data$imd\_band == "" | is.na(Whole\_final\_data$imd\_band),]$region)

regions\_list

for (i in regions\_list) {

mode\_band <- names (which.max(table(Whole\_final\_data$imd\_band[Whole\_final\_data$region == i])))

Whole\_final\_data$imd\_band[Whole\_final\_data$imd\_band == "" & Whole\_final\_data$region == i] <- mode\_band

Whole\_final\_data$imd\_band[is.na(Whole\_final\_data$imd\_band) & Whole\_final\_data$region == i] <- mode\_band

}

#checking null values in imb\_band

sapply(train, function(x) sum(is.na(x)))

# Make a new dataframe just with rows that have null values for the registration date

reg\_date\_nulls\_in\_reg <- Whole\_final\_data[Whole\_final\_data$date\_registration %in% c("", "NA", "NULL"),]

reg\_date\_nulls\_in\_reg

# What are their final results?

column <- reg\_date\_nulls\_in\_reg$final\_result

counts <- table(column)

counts

str(Whole\_final\_data$date\_registration)

#chnage into numeric

Whole\_final\_data$date\_registration <- as.integer(Whole\_final\_data$date\_registration)

Whole\_final\_data$date\_unregistration <- as.integer(Whole\_final\_data$date\_unregistration)

median(Whole\_final\_data$date\_registration, na.rm=TRUE)

# Replace NaN values with date\_unreg minus the median (note, the median is negative)

Whole\_final\_data$date\_registration <- ifelse(is.na(Whole\_final\_data$date\_registration),

Whole\_final\_data$date\_unregistration + median(Whole\_final\_data$date\_registration, na.rm = TRUE),

Whole\_final\_data$date\_registration)

# Replace remaining NaNs with -57

Whole\_final\_data$date\_registration <- ifelse(is.na(Whole\_final\_data$date\_registration),

median(Whole\_final\_data$date\_registration, na.rm = TRUE),

Whole\_final\_data$date\_registration)

#For sum\_click

#Replace with NaN value

Whole\_final\_data$sum\_clicks[is.na(Whole\_final\_data$sum\_clicks)] <- 0

Whole\_final\_data$weighted\_score[is.na(Whole\_final\_data$weighted\_score)] <- 0

Whole\_final\_data$late\_rate[is.na(Whole\_final\_data$late\_rate)] <- 0

Whole\_final\_data$fail\_rate[is.na(Whole\_final\_data$fail\_rate)] <- 0

##### For activity\_level\_score

# calculate the mean of the column

mean\_val <- mean(Whole\_final\_data$activity\_level\_score, na.rm = TRUE)

# replace NaN values with the mean value

Whole\_final\_data$activity\_level\_score <- ifelse(is.na(Whole\_final\_data$activity\_level\_score), mean\_val, Whole\_final\_data$activity\_level\_score)

########## For date\_unregistrations

# calculate the mean of the column

mean\_val\_unreg <- mean(Whole\_final\_data$date\_unregistration, na.rm = TRUE)

# replace NaN values with the mean value

Whole\_final\_data$date\_unregistration <- ifelse(is.na(Whole\_final\_data$date\_unregistration), mean\_val\_unreg, Whole\_final\_data$date\_unregistration)

#checking null values in every columns

sapply(Whole\_final\_data, function(x) sum(is.na(x)))

write.csv(Whole\_final\_data, file = "Final\_olad\_dataset.csv", row.names = TRUE)

### **Appendix for Model building**

#------------------------------eda again--------------------------------------------------#

# set the working directory to where the CSV file is located

setwd(dirname(file.choose()))

getwd()

#------------Libraries -----------------#

library(tidyverse)

library(lubridate)

library(dplyr)

library(ggplot2)

library(caret)

library(corrplot)

library(e1071)

library(ROCR)

library(rpart)

library(rpart.plot)

library(glmnet)

#reload dataset

data <- read.csv("Final\_olad\_dataset ".csv)

#--------------------------- summary statistics and visualizations ----------------------------------#

# Calculate summary statistics for the numerical variables

summary\_stats <- summary(data[, num\_var\_names])

# View the summary statistics

summary\_stats

# Create a histogram for each numerical variable

hist\_list <- lapply(num\_var\_names, function(var) {

ggplot(data, aes(x = !!as.name(var))) +

geom\_histogram(binwidth = 10, color = "black", fill = "lightblue") +

ggtitle(paste("Histogram of", var)) +

xlab(var) + ylab("Frequency") +

theme\_minimal()

})

# Combine the histograms in a grid

grid <- do.call("grid.arrange", c(hist\_list, ncol = 3))

grid

# get the summary statistics of the categorical variables

# Calculate summary statistics for the categorical variables

for (var in cat\_vars) {

cat\_summary <- table(data[[var]])

print(cat\_summary)

}

#------------------------------- OUTLIER DETECTIONS --------------------------------------#

# Since we have converted categorical variable into numeric variables

#we need to do outlier detections on this dataset

library(ggplot2)

#Data Columns names

xdata\_ref <- data.frame(

late\_rate = rnorm(100),

fail\_rate = rnorm(100),

studied\_credits = rnorm(100),

activity\_level\_score = rnorm(100),

sum\_clicks = rnorm(100),

code\_module\_numeric = rnorm(100),

code\_presentation\_numeric = rnorm(100),

module\_presentation\_length\_numeric = rnorm(100),

gender\_numeric = rnorm(100),

highest\_education\_numeric = rnorm(100),

age\_band\_numeric = rnorm(100),

num\_of\_prev\_attempts\_numeric = rnorm(100),

disability\_numeric = rnorm(100),

final\_result\_numeric = rnorm(100),

weighted\_score= rnorm(100)

)

# create a list of variables to plot

my\_vars <- c("late\_rate", "fail\_rate", "studied\_credits",

"activity\_level\_score", "sum\_clicks", "code\_module\_numeric", "code\_presentation\_numeric",

"module\_presentation\_length\_numeric", "gender\_numeric", "highest\_education\_numeric",

"age\_band\_numeric", "num\_of\_prev\_attempts\_numeric", "disability\_numeric",

"final\_result\_numeric", "weighted\_score")

# create boxplots and store in a list

boxplots <- lapply(my\_vars, function(col) {

ggplot(data = mydata, aes(x = "", y = mydata[[col]])) +

geom\_boxplot() +

ggtitle(col) +

ylab(col)

#-----------------------------------------------Reassign data---------------------------------------------#

olad<- data

# Convert columns to numeric

olad$late\_rate <- as.numeric(olad$late\_rate)

olad$studied\_credits <- as.numeric(olad$studied\_credits)

olad$activity\_level\_score <- as.numeric(olad$activity\_level\_score)

olad$sum\_clicks <- as.numeric(olad$sum\_clicks)

#convert columns to factor

olad$final\_result <- factor(olad$final\_result)

olad$highest\_education <- factor(olad$highest\_education)

#---------------------------------VIF calculations------------------------------------#

#before calculating EFA we have to connvert all factor to numeric data types

olad $weighted\_score <- as.numeric(olad $weighted\_score)

olad $studied\_credits <- as.numeric(olad $studied\_credits)

olad $activity\_level\_score <- as.numeric(olad $activity\_level\_score)

olad $sum\_clicks <- as.numeric(olad $sum\_clicks)

# Perform exploratory factor analysis

efa\_model <- factanal(olad, factors = 1, rotation = "varimax")

#factoranaly<-factanal(olad, factors = 1)

# Print the factor loadings

print(efa\_model$loadings)

#--------------------------------Normalizations ------------------------------------------#

# Assign numerical values using as.numeric()

olad$highest\_education\_numeric <- as.numeric(olad$highest\_education)

olad$final\_numeric <- as.numeric(olad$final)

class(olad$final\_numeric)

# Apply rank-based normalization to the numeric variables

olad[, c("weighted\_score", "late\_rate", "studied\_credits", "activity\_level\_score", "sum\_clicks")] <- apply(olad[, c("weighted\_score", "late\_rate", "studied\_credits", "activity\_level\_score", "sum\_clicks")], 2, function(x) (rank(x)-1)/(length(x)-1))

#----------------------------- Correlation analysis --------------------------------------------#

# correlations matrix will all variables

# create correlation matrix

corr <- cor(olad[,c("weighted\_score", "activity\_level\_score", "studied\_credits", "late\_rate", "highest\_education\_numeric", "sum\_clicks")

])

#for few variables

# plot correlation matrix

corrplot(corr, method = "color", type = "upper", tl.col = "black", tl.srt = 45, addCoef.col = "black", diag = FALSE)

corr

cor\_matrix <- cor(olad)

# plot a heatmap of the correlation matrix

library(ggplot2)

ggplot(data = reshape2::melt(cor\_matrix)) +

geom\_tile(aes(x = Var1, y = Var2, fill = value)) +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +

labs(title = "Correlation Matrix Heatmap")

#-----------------------------------Scatter plot -------------------------------------------------#

pairs(olad[, c("weighted\_score", "activity\_level\_score", "studied\_credits","sum\_clicks", "highest\_education\_numeric", "late\_rate")])

model\_data<- olad

names(olad)

model\_data<- select(model\_data, -id\_student)

model\_data<- select(model\_data, -final\_result, -highest\_education, -id\_student, -final\_numeric, -highest\_education\_numeric, -studied\_credits)

head(model\_data)

str(model\_data)

#------------------------------------------ Building model-------------------------------------#

# Split the data into training and testing sets

set.seed(123) # Set the seed for reproducibility

trainIndex <- createDataPartition(model\_data$weighted\_score, p = 0.8, list = FALSE)

data.train <- model\_data[trainIndex, ]

data.test <- model\_data[-trainIndex, ]

fitControl <- trainControl(method="cv", number=10)

#-------------------------- 1. Multiple LINEAR REGRESSION -------------------------------------#

# Build the model

model.lm <- lm(weighted\_score ~ late\_rate + sum\_clicks + activity\_level\_score + highest\_education\_numeric + studied\_credits + final\_numeric , data = data.train)

print(model.lm)

# Make predictions on the test set

predictions <- predict(model.lm, newdata = data.test)

print(predictions)

# Evaluate the model performance

RMSE <- sqrt(mean((data.test$weighted\_score - predictions)^2))

R\_squared <- summary(model.lm)$r.squared

# calculate MAE

MAE <- mean(abs(test$actual - predictions))

# Print the model performance metrics

cat("RMSE:", RMSE, "\n")

cat("R-squared:", R\_squared, "\n")

cat("MAE:", round(MAE, 2), "\n")

#---------------------- 2. Random Forest --------------------------------#

#training the random forest model

# 10 fold Cross-validation

model.rf <- train(weighted\_score ~ .,

data = data.train,

method = "rf",

trControl = fitControl)

print(model.rf)

# Make predictions on the test set

predictions1 <- predict(model.rf, newdata = data.test)

#test the random forest

preds1 <- predict(model.rf, data.test)

class(preds1)

class(data.test$weighted\_score)

# visualizing the predicted and actual values in R

ggplot(data = data.frame(predictions1, data.test$weighted\_score), aes(x = preds1, y = data.test$weighted\_score)) +

geom\_point() +

geom\_smooth(method = "lm", se = FALSE) +

labs(title = "Predicted vs Actual Weighted Score", x = "Predicted Score", y = "Actual Score")

# Convert predictions to factor type

preds1\_factor <- ifelse(predictions1 > 0.5, "1", "0")

preds1\_factor <- factor(preds1\_factor, levels = c("0", "1"))

# Convert test data to factor type

test\_factor <- ifelse(data.test$weighted\_score > 0.5, "1", "0")

test\_factor <- factor(test\_factor, levels = c("0", "1"))

# Create confusion matrix

confusion\_mtx <- confusionMatrix(preds1\_factor, test\_factor)

# Print precision, recall, and F1-score

precision <- confusion\_mtx$byClass['Pos Pred Value']

recall <- confusion\_mtx$byClass['Sensitivity']

f1\_score <- 2 \* ((precision \* recall) / (precision + recall))

cat("Precision:", precision, "\n")

cat("Recall:", recall, "\n")

cat("F1-score:", f1\_score, "\n")

#------------------------------ 3. Ridge Regression ----------------------------------#

model.rr <- train(weighted\_score ~ ., data = train, method = "ridge", trControl = fitControl)

print(model.rr)

# compute predictions using the ridge regression model

preds <- predict(model.rr, test)

# compute mean absolute error (MAE)

mae <- mean(abs(test$weighted\_score - preds))

print(paste0("MAE: ", mae))

# compute root mean squared error (RMSE)

rmse <- sqrt(mean((test$weighted\_score - preds)^2))

print(paste0("RMSE: ", rmse))

# compute R-squared (R2)

r2 <- 1 - sum((test$weighted\_score - preds)^2) / sum((test$weighted\_score - mean(test$weighted\_score))^2)

print(paste0("R-squared (R2): ", r2))

#--------------------------- 4. SVM ------------------------------------------#

model <- svm(weighted\_score ~ ., data = train)

# Create the SVM regression model

svmModel <- svm(weighted\_score ~ late\_rate + studied\_credits + activity\_level\_score + highest\_education\_numeric +final\_numeric+ sum\_clicks,

data = trainData,

kernel = "radial",

cost = 10,

gamma = 1)

# Make predictions on the testing set

predictions <- predict(svmModel, testData)

# Evaluate the model performance

RMSE <- sqrt(mean((predictions - testData$weighted\_score)^2))

R\_squared <- cor(predictions, testData$weighted\_score)^2

# Print the model performance metrics

cat(paste("RMSE: ", RMSE, "\n"))

cat(paste("R-squared: ", R\_squared, "\n"))

# Convert predictions to factor type

predictions\_factor <- ifelse(predictions > 0.5, "1", "0")

predictions\_factor <- factor(predictions\_factor, levels = c("0", "1"))

# Convert test data to factor type

test\_factor <- ifelse(testData$weighted\_score > 0.5, "1", "0")

test\_factor <- factor(test\_factor, levels = c("0", "1"))

# Create confusion matrix

confusion\_mtx <- confusionMatrix(predictions\_factor, test\_factor)

# Calculate precision, recall, and F1-score

precision <- confusion\_mtx$byClass['Pos Pred Value']

recall <- confusion\_mtx$byClass['Sensitivity']

f1\_score <- 2 \* ((precision \* recall) / (precision + recall))

# Print precision, recall, and F1-score

cat("Precision:", precision, "\n")

cat("Recall:", recall, "\n")

cat("F1-score:", f1\_score, "\n")

#-------------------------- 6.Lasso Regression ------------------------------------------#

# fit the Lasso regression model using glmnet

model.lasso <- train(

weighted\_score ~ .,

data = train,

method = "glmnet",

trControl = trainControl(method = "cv", number = 10),

tuneGrid = expand.grid(alpha = 1, lambda = seq(0.01, 1, length = 100))

)

print(model.lasso)

# print the model coefficients

coef(model.lasso$finalModel, s = model.lasso$bestTune$lambda)

# make predictions on the test set

preds <- predict(model.lasso, newdata = test)

# compute performance metrics

mae <- mean(abs(test$weighted\_score - preds))

rmse <- sqrt(mean((test$weighted\_score - preds)^2))

r2 <- 1 - sum((test$weighted\_score - preds)^2) / sum((test$weighted\_score - mean(test$weighted\_score))^2)

# print the performance metrics

cat("MAE:", mae, "\n")

cat("RMSE:", rmse, "\n")

cat("R-squared (R2):", r2, "\n")

#---------------------------------------End-----------------------------------------------------#