HARDWARE AND SOFTWARE PER BIG DATA Mod B

By Prof. Flora Amato

Prepared by:

Syed Muhammad Muneeb	D03000038
Hassan Muhammad Siraj Zafar	D03000033
Muhammad Sameer Kazi	D03000032

Università degli Studi di Napoli Federico II

1. Introduction

1.1 BACKGROUND INFORMATION

In today's data-driven world, educational institutions are increasingly leveraging data analytics to enhance student outcomes and optimize administrative processes. Our project aimed to harness the power of data analytics to gain insights into student performance and engagement, utilizing a comprehensive dataset consisting of seven interconnected tables: **assessments**, **courses**, **student assessment**, **student info**, **student registration**, **student vle**, and **vle**.

To achieve this, we implemented an Exploratory Data Analysis (EDA) using Streamlit, a powerful and user-friendly application framework for data science and machine learning projects. The EDA process involved computing missing values and conducting a thorough analysis to understand the underlying patterns and relationships within the data.

Following the EDA, we employed various modelling techniques to draw meaningful conclusions and provide actionable insights. These techniques included:

Linear Regression: Used to model the relationship between continuous dependent and independent variables, helping us predict student performance based on various factors.

Decision Tree Regressor: Employed to capture non-linear relationships in the data and provide interpretable models for predicting student outcomes.

ARIMA: Utilized for time series analysis to forecast trends and patterns in student performance over time.

Dropout Analysis with Random Forest: Applied to identify key factors contributing to student dropout rates, leveraging the robustness and feature importance capabilities of Random Forest.

Through this comprehensive approach, our goal was to provide a detailed and insightful analysis that could inform data-driven decision-making in educational settings, ultimately contributing to improved student success and institutional efficiency.

About the dataset:
This dataset belongs to Open University Online Learning Platform (Also called as "Virtual Learning Environment (VLE)") that off-campus students use for accessing the course content, forum discussions, sending assessments and checking out assignment marks etc. It consists of 7 selected courses (mentioned as modules in the dataset). Different presentations indicated with letters "B" and "J" after year for semester 2 and semester 1 respectively.

Additionally, the dataset includes student demographics such as location, age group, disability, education level, gender etc. Student assessment marks, interactions with the Virtual Learning Environment (VLE) are also included.

It contains data about courses, students and their interactions with Virtual Learning Environment (VLE) for seven selected courses (called modules). Presentations of courses start in February and October - they are marked by “B” and “J” respectively. The dataset consists of tables connected using unique identifiers.

1.2 SIGNIFICANCE AND MOTIVATION

The significance and motivation of using the approach outlined in our project lie in several key factors:

1. **Comprehensive Data Analysis:** By scraping historical stock data from Yahoo Finance and utilizing advanced data processing techniques, we gain access to a vast repository of information on the automotive manufacturing industry. This allows for a comprehensive analysis of stock performance over time, enabling us to identify trends, patterns, and outliers that may not be apparent through manual analysis alone.
2. **Scalability and Efficiency:** Leveraging distributed computing frameworks like Spark enables us to scale our analysis to handle large volumes of data efficiently. This approach ensures that we can process and analyse extensive datasets without being limited by hardware constraints, thereby facilitating more in-depth and accurate insights into the automotive manufacturing industry's stock data.
3. **Advanced Analytics:** By employing clustering algorithms like K-means, we can uncover hidden structures and relationships within the stock data, enabling us to segment stocks based on their performance characteristics. This facilitates the identification of distinct market trends, investment opportunities, and risk factors within the automotive manufacturing industry, empowering investors and stakeholders to make informed decisions.
4. **Real-time Data Management:** Utilizing Apache Kafka for data storage and messaging provides a robust and scalable solution for managing real-time data streams. This approach enables us to capture, store, and process streaming data from various sources, ensuring that our analysis remains up-to-date and responsive to changing market conditions.
5. **Holistic Approach:** By integrating web scraping, data preprocessing, distributed computing, clustering analysis, and data storage using a combination of Python libraries and Apache Kafka, our approach provides a comprehensive framework for analyzing and understanding the automotive manufacturing industry's stock data. This holistic approach enables us to address various aspects of data analysis and management, resulting in more accurate and actionable insights.

Overall, the significance of our approach lies in its ability to provide a scalable, efficient, and comprehensive solution for analysing historical stock data from the automotive manufacturing industry. By leveraging advanced technologies and techniques, we can uncover valuable insights that can inform investment strategies, risk management practices, and strategic decision-making processes.

1.3 METHODOLOGIES

To deal with the aims and objectives of the project we chose to use the following tools and technologies for our project. They are as follows:

1.3.1 HARDWARE

The development process was done by the following laptops as shown:

Description	Configuration
Laptops	HP Pavilion Laptop 15

1.3.2 SOFTWARE

The development process was done by the following operating systems and software as shown:

Description	Technology
Operating Systems	Windows 11
Editor	Google Collab

1.3.3 DEVELOPMENT SIDE TECHNOLOGIES

The development process was done by the following technologies as shown:

Description	Technology
Framework	Streamlit
Collaboration	Local host
Summarizer Engine	VS Code v1.86.1
UML Diagrams	Star UML v5.0.1

2. Literature Review

2.1 INTRODUCTION

Since the inception of this project, we have taken everything under consideration with respect to research. From gathering the prerequisites of the project to deploying it, the complete guidance for each module of the project was taken under study.

2.1 BIG DATA

Big data has become increasingly important in recent years due to its ability to unlock valuable insights and drive advancements across various sectors. It is now widely used in many organizations and fields such as:

Finance:

- Fraud detection and prevention using machine learning algorithms.
- Risk assessment and mitigation through analysis of market trends and customer behaviour.
- Algorithmic trading for automated decision-making in stock markets.
- Customer segmentation and targeted marketing based on financial transaction data.

Retail:

- Demand forecasting to optimize inventory management and supply chain operations.
- Customer analytics for personalized marketing and recommendation systems.
- Market basket analysis to identify patterns and correlations between products.
- Sentiment analysis of customer reviews and social media data for brand perception.

Manufacturing:

- Predictive maintenance to reduce equipment downtime and optimize maintenance schedules.
- Quality control and defect detection through real-time monitoring of production processes.
- Supply chain optimization for efficient procurement and logistics management.
- Energy management and resource utilization optimization to reduce costs and environmental impact

2.2 STREAMLIT

Streamlit is an open-source framework designed to facilitate the creation of web applications for machine learning and data science projects. It's popular because it allows developers to quickly build interactive and data-driven web interfaces with minimal code. Here are some of the main uses and reasons for its popularity:

Data Visualization:

- Create interactive dashboards and visualizations.
- Display plots from libraries like Matplotlib, Plotly, Altair, and more.

Machine Learning Applications:

- Deploy machine learning models as web applications for easy access.
- Provide interactive interfaces for model input and output visualization.

Prototyping:

- Quickly prototype ideas and see results without extensive frontend development.
- Iterate rapidly by modifying code and seeing changes in real-time.

2.3 PANDAS

Pandas is a popular open-source Python library used for data manipulation and analysis. It provides high-performance, easy-to-use data structures and tools for working with structured data, such as tabular data, time series, and heterogeneous data. Some of the key features and uses of Pandas include:

- Data structures
- data loading and sstorage
- data exploration and cleaning
- data manipulation and transformation

2.4 LINEAR REGRESSION

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The simplest form, called simple linear regression, involves one dependent variable (Y) and one independent variable (X). Multiple linear regression involves one dependent variable and two or more independent variables.

- **Understanding Relationships:**
  Determining the strength and type of relationship between dependent and independent variables. For example, understanding how changes in marketing spend impact sales.
- **Predictive Modelling:**
  Predicting outcomes such as sales forecasts, stock prices, and other continuous variables based on predictor variables.
- **Trend Analysis:**
  Identifying and understanding trends in data over time.
- **Risk Management:**
  Quantifying the risk or effect of variables on a particular outcome, such as credit scoring in finance.
- **Optimizing Processes:**
  In operations and manufacturing, linear regression can be used to optimize processes by understanding the impact of different factors on production efficiency.

2.5 DECISION TREE REGRESS

A decision tree regressor is a machine learning model used for regression tasks, which involve predicting a continuous output variable. Unlike linear regression, which models the relationship between the dependent and independent variables as a straight line, decision tree regression models the data using a tree-like structure of decisions based on the features.

- **Interpretability:**
  Decision trees are easy to visualize and interpret, making it straightforward to understand how decisions are made.
- **Non-Linear Relationships:**
  They can capture non-linear relationships between the features and the target variable.
- **No Need for Feature Scaling:**
  They do not require scaling or normalization of the features.
- **Handling Missing Values:**
  Decision trees can handle missing values in the data.

2.6 ARIMA

ARIMA (AutoRegressive Integrated Moving Average) is a popular statistical method used for time series forecasting. It combines three key components:

AutoRegressive (AR) part: This component uses the relationship between an observation and a number of lagged observations (previous time steps) to model the time series.

Integrated (I) part: This involves differencing the time series data to make it stationary, which means that the mean and variance of the series are constant over time.

Moving Average (MA) part: This component models the relationship between an observation and a residual error from a moving average model applied to lagged observations.

2.7 RANDOM FOREST

Random Forest is an ensemble machine learning method primarily used for classification and regression tasks. It operates by constructing multiple decision trees during training and outputting the mean prediction (for regression) or the mode of the classes (for classification) of the individual trees.

Accuracy:
Generally provides high accuracy by reducing overfitting compared to individual decision trees.

Robustness:
Less sensitive to noisy data and outliers.

Feature Importance:
Can rank the importance of features in prediction.

3. Architecture

3.1 INTRODUCTION

This section provides an outline of the project, its integral parts and major data objects involved and their relationship that flows in the system. It gives us basic understanding about system architecture, how the data objects interact with the system and the overall workflow. The overall running process and the data and interface description are briefly described in this section below.

3.2 SYSTEM ARCHITECTURE

In order to build a scalable, efficient and maintainable software system, layered architecture was built for separation of concern, scalability and maintenance. Each layer is independent of each other as they are configured. separately for each process. If need arises, these layers can be replaced by various implementations.

3.3 ERD

An Entity-Relationship Diagram (ERD) is a visual representation of the data and its relationships within a system. It is a key tool used in database design to illustrate the logical structure of databases. Here’s how an ERD is used in the context of your project:

Entities and Attributes

Assessments:
- Attributes: assessment_id, course_id, type, date, weight

Courses:
- Attributes: course_id, course_name, course_code, department

Student Assessment:
- Attributes: student_id, assessment_id, score

Student Info:
- Attributes: student_id, name, age, gender, major

Student Registration:
- Attributes: student_id, course_id, registration_date, status

Student VLE:
- Attributes: student_id, vle_id, interactions

VLE (Virtual Learning Environment):
- Attributes: vle_id, course_id, resource_type, resource_name

Relationships

Courses and **Assessments**: One-to-many relationship (one course can have multiple assessments).

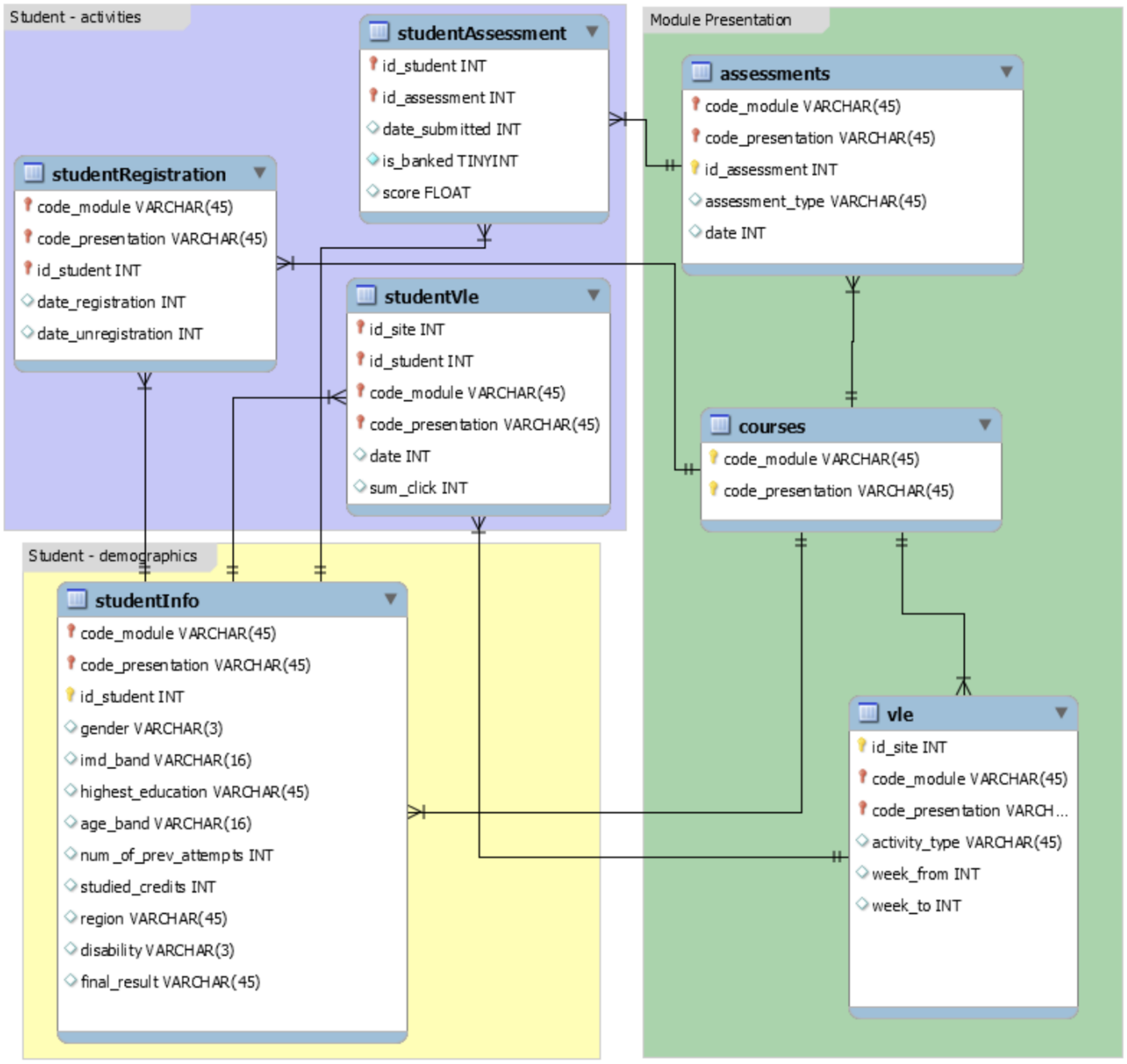
Students and **Student Assessment**: One-to-many relationship (one student can have multiple assessments).

Courses and **Student Registration**: Many-to-many relationship (one student can register for multiple courses, and one course can have multiple students).

Students and **Student VLE**: One-to-many relationship (one student can interact with multiple VLE resources).

Courses and **VLE**: One-to-many relationship (one course can have multiple VLE resources).

3.3 ARCHITECTURE MODEL



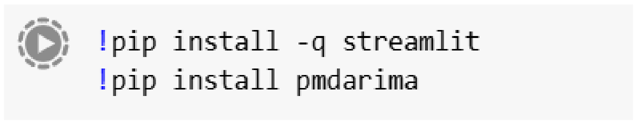
1. Functional and Data Description

. 4.1 INTRODUCTION

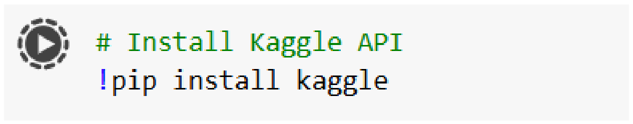
This section describes the main modules of this application, sub-modules and their components that are implemented in order to achieve required results.

. 4.2 DATA LOADING

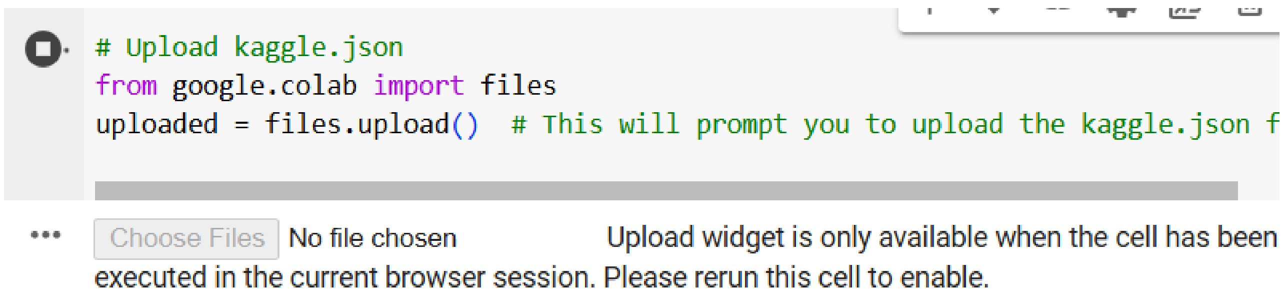
In this section the data from the dataset is loaded from Kaggle. We first run the streamlit and pmdarima after it the dataset is loaded and uploaded from kaggle.



```
!pip install -q streamlit
!pip install pmdarima
```

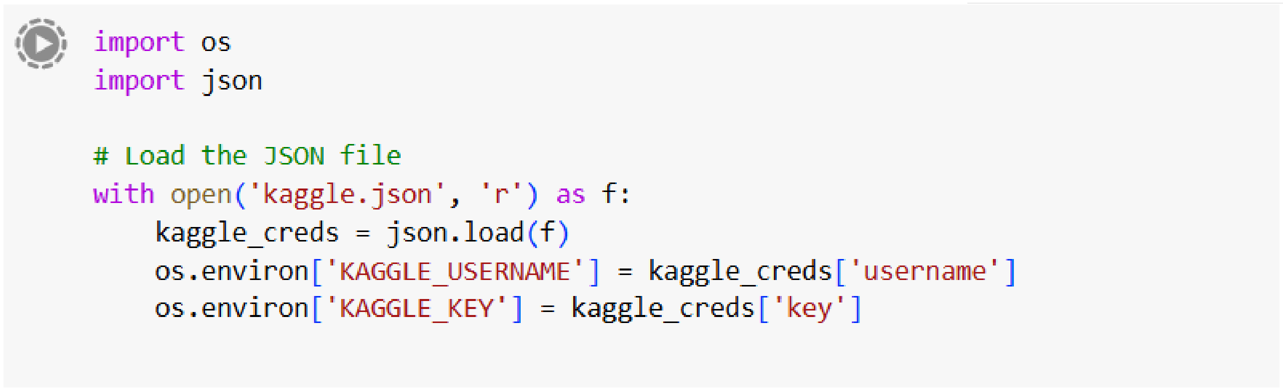


```
# Install Kaggle API
!pip install kaggle
```

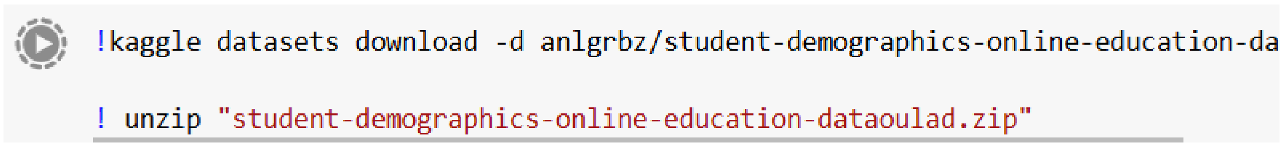


```
# Upload kaggle.json
from google.colab import files
uploaded = files.upload()  # This will prompt you to upload the kaggle.json f
```

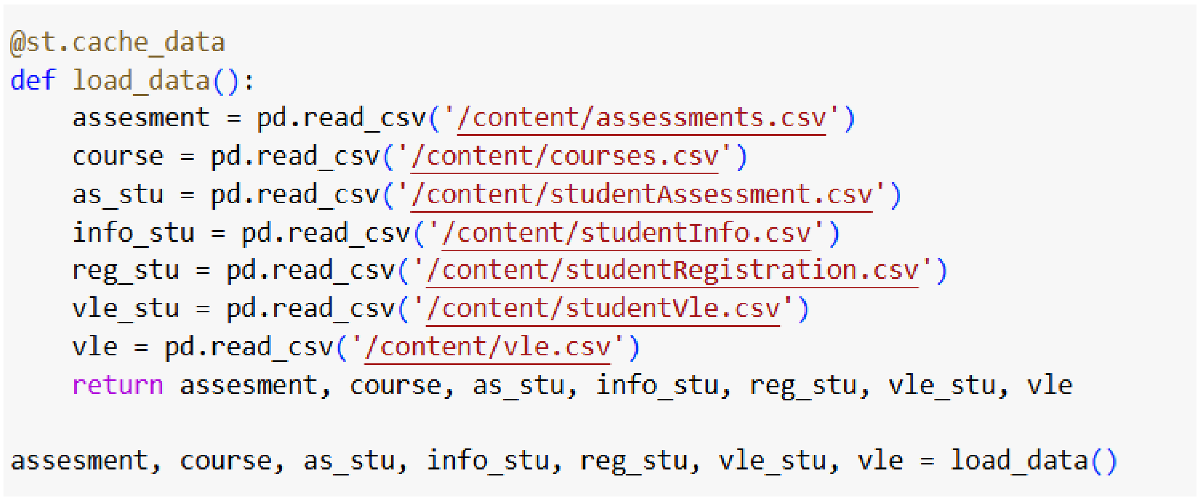
```
Choose Files No file chosen          Upload widget is only available when the cell has been
executed in the current browser session. Please rerun this cell to enable.
```



```
import os
import json

# Load the JSON file
with open('kaggle.json', 'r') as f:
    kaggle_creds = json.load(f)
    os.environ['KAGGLE_USERNAME'] = kaggle_creds['username']
    os.environ['KAGGLE_KEY'] = kaggle_creds['key']
```

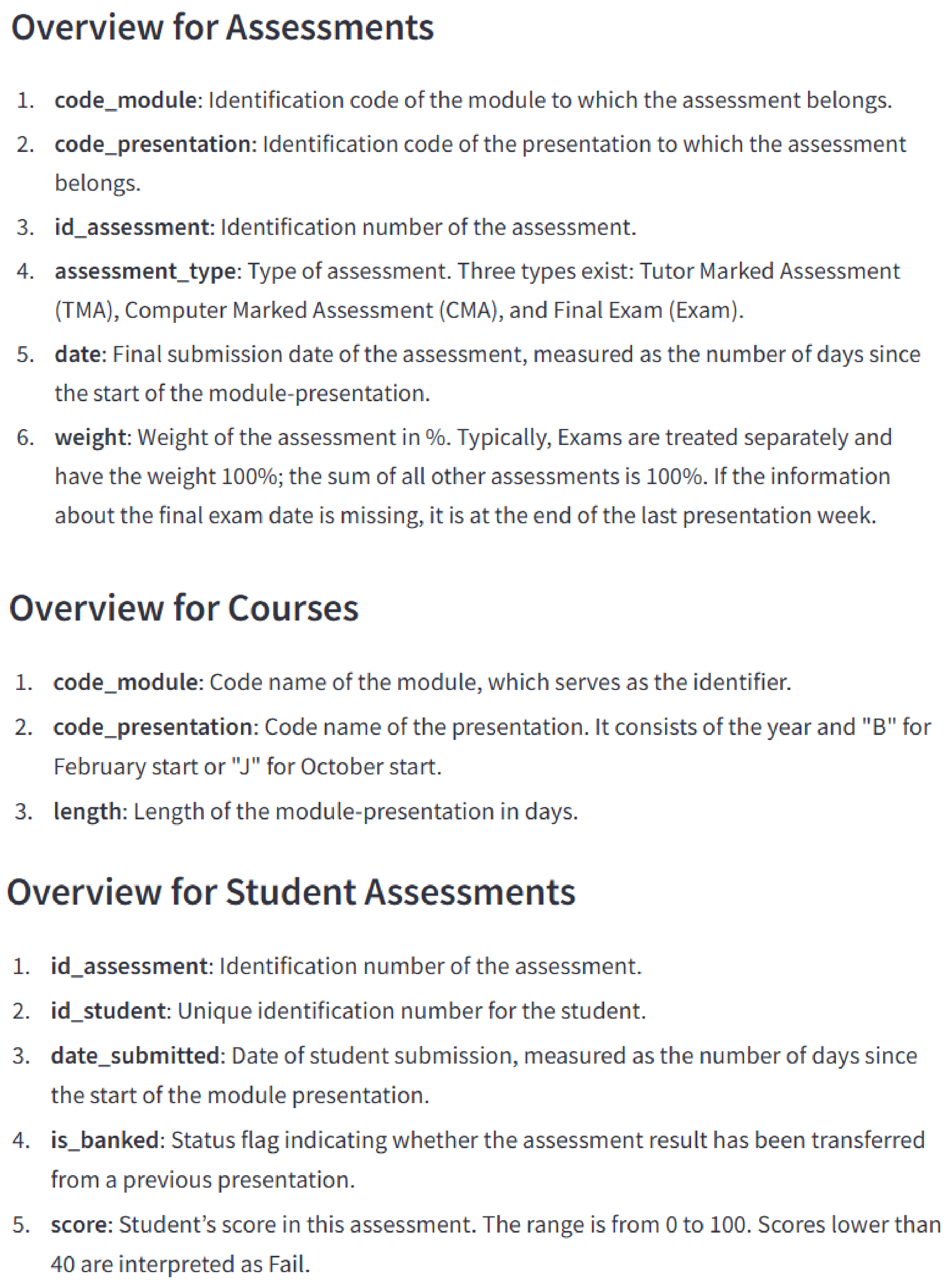


```
!kaggle datasets download -d anlgrbz/student-demographics-online-education-da

! unzip "student-demographics-online-education-dataoulad.zip"
```



```
@st.cache_data
def load_data():
    assesment = pd.read_csv('/content/assessments.csv')
    course = pd.read_csv('/content/courses.csv')
    as_stu = pd.read_csv('/content/studentAssessment.csv')
    info_stu = pd.read_csv('/content/studentInfo.csv')
    reg_stu = pd.read_csv('/content/studentRegistration.csv')
    vle_stu = pd.read_csv('/content/studentVle.csv')
    vle = pd.read_csv('/content/vle.csv')
    return assesment, course, as_stu, info_stu, reg_stu, vle_stu, vle

assesment, course, as_stu, info_stu, reg_stu, vle_stu, vle = load_data()
```

4.2.1 DATASET OVERVIEW



Overview for Assessments

1. **code_module**: Identification code of the module to which the assessment belongs.
2. **code_presentation:** Identification code of the presentation to which the assessment belongs.
3. **id_assessment:** Identification number of the assessment.
4. **assessment_type**: Type of assessment. Three types exist: Tutor Marked Assessment (TMA), Computer Marked Assessment (CMA), and Final Exam (Exam).
5. **date**: Final submission date of the assessment, measured as the number of days since the start of the module-presentation.
6. **weight**: Weight of the assessment in %. Typically, Exams are treated separately and have the weight 100%; the sum of all other assessments is 100%. If the information about the final exam date is missing, it is at the end of the last presentation week.

Overview for Courses

1. **code_module**: Code name of the module, which serves as the identifier.
2. **code_presentation**: Code name of the presentation. It consists of the year and "B" for February start or "J" for October start.
3. **length**: Length of the module-presentation in days.

Overview for Student Assessments

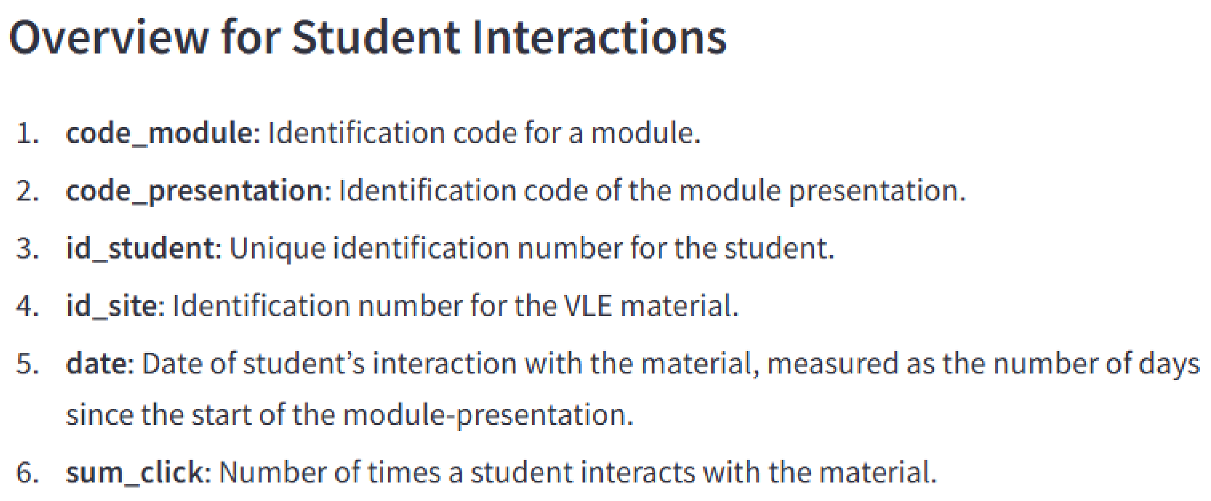
1. **id_assessment**: Identification number of the assessment.
2. **id_student**: Unique identification number for the student.
3. **date_submitted:** Date of student submission, measured as the number of days since the start of the module presentation.
4. **is_banked**: Status flag indicating whether the assessment result has been transferred from a previous presentation.
5. **score**: Student's score in this assessment. The range is from 0 to 100. Scores lower than 40 are interpreted as Fail.

Overview for Student Info

1. **code_module**: Identification code for the module.
2. **code_presentation**: Identification code of the presentation.
3. **id_student**: Unique identification number for the student.
4. **gender**: Student’s gender.
5. **region**: Geographic region where the student lived during the module presentation.
6. **highest_education**: Highest education level of the student upon entry to the module presentation.
7. **imd_band**: Index of Multiple Deprivation band of the place where the student lived during the module presentation.
8. **age_band**: Age band of the student.
9. **num_of_prev_attempts**: Number of times the student has attempted this module.
10. **studied_credits**: Total number of credits for the modules the student is currently studying.
11. **disability**: Indicates whether the student has declared a disability.
12. **final_result**: Student’s final result in the module presentation.

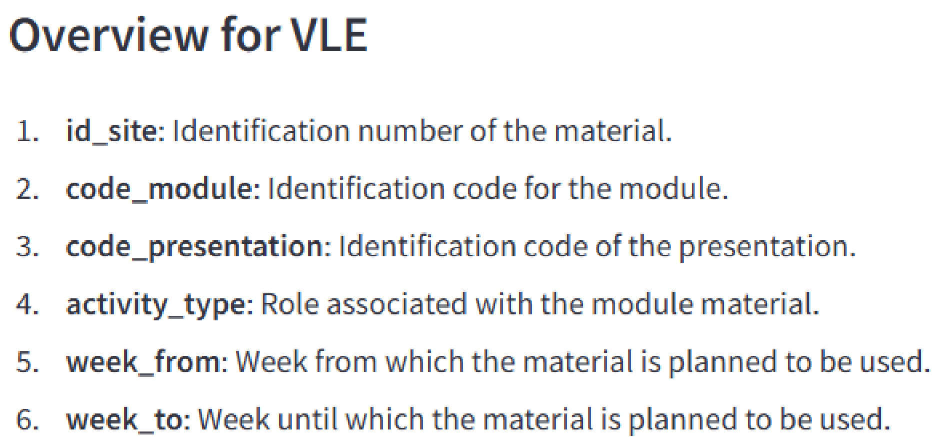
Overview for Student Registration

1. **code_module**: Identification code for a module.
2. **code_presentation**: Identification code of the presentation.
3. **id_student**: Unique identification number for the student.
4. **date_registration**: Date of student’s registration on the module presentation, measured as the number of days relative to the start of the module-presentation. Negative values indicate registration before the start.
5. **date_unregistration**: Date of student’s un-registration from the module presentation, measured as the number of days relative to the start of the module-presentation.



Overview for Student Interactions

1. **code_module**: Identification code for a module.
2. **code_presentation**: Identification code of the module presentation.
3. **id_student**: Unique identification number for the student.
4. **id_site**: Identification number for the VLE material.
5. **date:** Date of student's interaction with the material, measured as the number of days since the start of the module-presentation.
6. **sum_click**: Number of times a student interacts with the material.



Overview for VLE

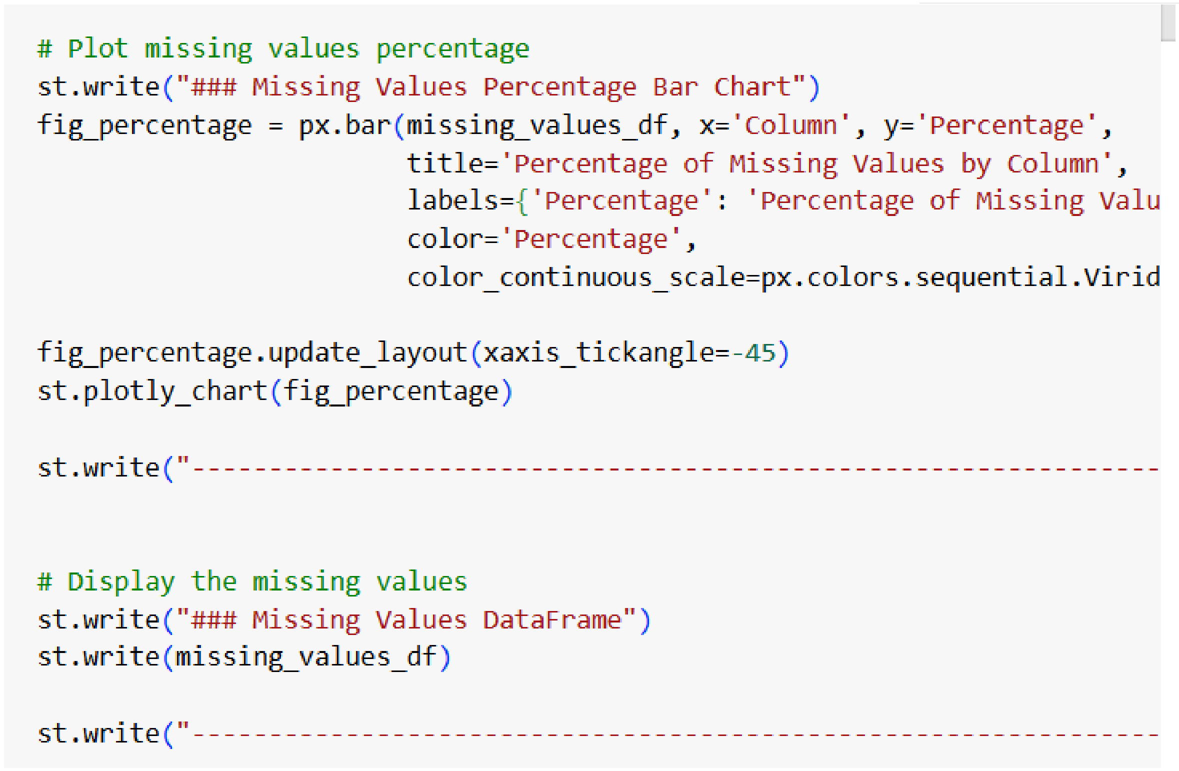
1. **id_site**: Identification number of the material.
2. **code_module**: Identification code for the module.
3. **code_presentation**: Identification code of the presentation.
4. **activity_type:** Role associated with the module material.
5. **week_from**: Week from which the material is planned to be used.
6. **week_to:** Week until which the material is planned to be used.

4.3 MISSING DATA

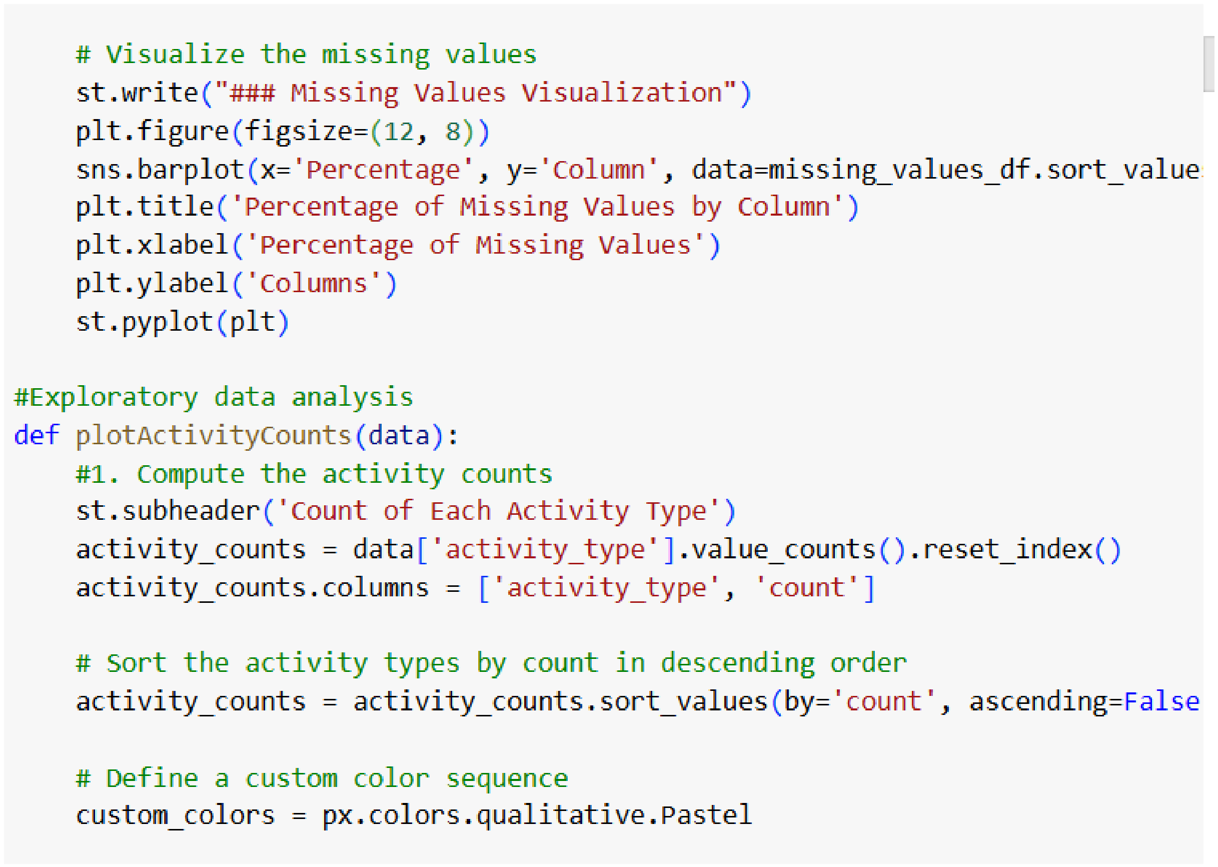
In this section the missing data is computed and cleaned using the necessary steps possible.

```python
def missingValueAssessment(data):
    # Display data information
    st.title("Missing Value Assessment")

    st.write("## Data Information")
    buffer = io.StringIO()
    data.info(buf=buffer)
    s = buffer.getvalue()
    st.text(s)

    st.write("### Dataframe Shape")
    st.write(f"**Rows**: {data.shape[0]}, **Columns**: {data.shape[1]}")

    st.write("---------------------------------------------------------------

    # Calculate missing values
    missing_values = data.isnull().sum()
    missing_values_percentage = (missing_values / len(data)) * 100
```

```python
    # DataFrame to display missing values
    missing_values_df = pd.DataFrame({
        'Column': missing_values.index,
        'Missing Values': missing_values.values,
        'Percentage': missing_values_percentage
    }).sort_values(by='Missing Values', ascending=False)

    # Display missing values in a table
    st.write("### Missing Values")
    st.dataframe(missing_values_df)

    # Plot missing values
    st.write("### Missing Values Bar Chart")
    fig = px.bar(missing_values_df, x='Column', y='Missing Values',
                 title='Missing Values by Column',
                 labels={'Missing Values': 'Count of Missing Values'},
                 color='Missing Values',
                 color_continuous_scale=px.colors.sequential.Viridis)

    fig.update_layout(xaxis_tickangle=-45)
    st.plotly_chart(fig)

    # Display missing values percentage
    st.write("### Missing Values Percentage")
    st.dataframe(missing_values_df[['Column', 'Percentage']])
```



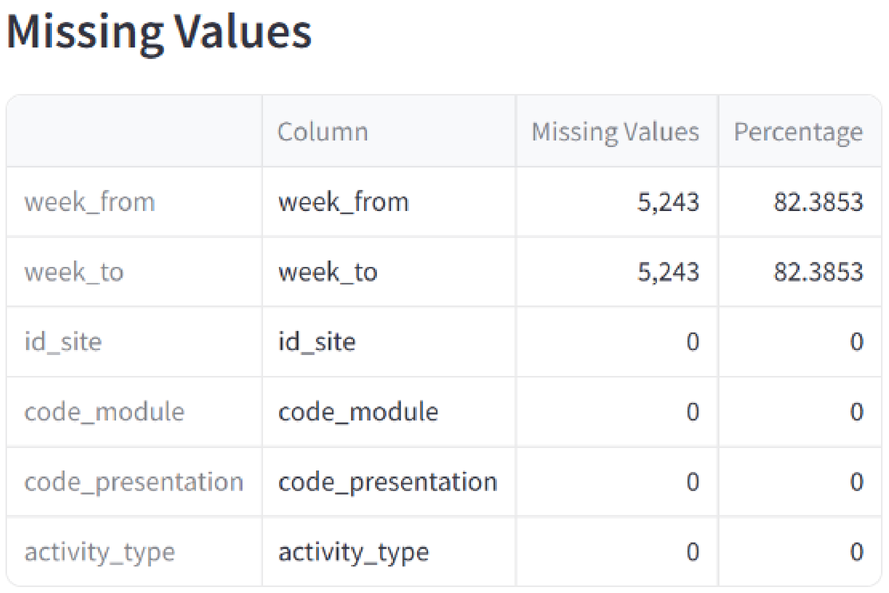
```
# Plot missing values percentage
st.write("### Missing Values Percentage Bar Chart")
fig_percentage = px.bar(missing_values_df, x='Column', y='Percentage',
                        title='Percentage of Missing Values by Column',
                        labels={'Percentage': 'Percentage of Missing Valu
                        color='Percentage',
                        color_continuous_scale=px.colors.sequential.Virid

fig_percentage.update_layout(xaxis_tickangle=-45)
st.plotly_chart(fig_percentage)

st.write("---------------------------------------------------------------


# Display the missing values
st.write("### Missing Values DataFrame")
st.write(missing_values_df)

st.write("---------------------------------------------------------------
```



```
    # Visualize the missing values
    st.write("### Missing Values Visualization")
    plt.figure(figsize=(12, 8))
    sns.barplot(x='Percentage', y='Column', data=missing_values_df.sort_value
    plt.title('Percentage of Missing Values by Column')
    plt.xlabel('Percentage of Missing Values')
    plt.ylabel('Columns')
    st.pyplot(plt)

#Exploratory data analysis
def plotActivityCounts(data):
    #1. Compute the activity counts
    st.subheader('Count of Each Activity Type')
    activity_counts = data['activity_type'].value_counts().reset_index()
    activity_counts.columns = ['activity_type', 'count']

    # Sort the activity types by count in descending order
    activity_counts = activity_counts.sort_values(by='count', ascending=False

    # Define a custom color sequence
    custom_colors = px.colors.qualitative.Pastel
```

4.3.1 GRAPHICAL REPRESENTATION

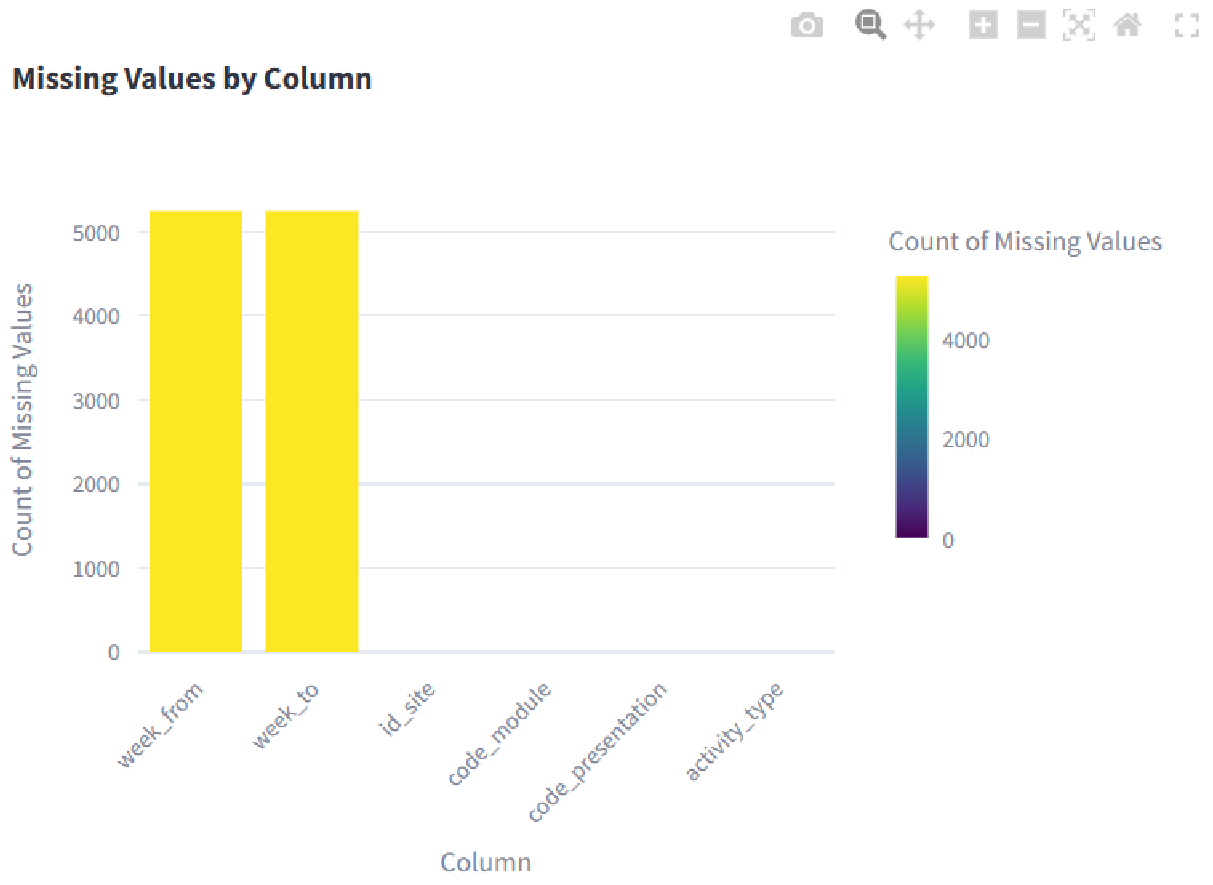
4.3.1.1 VLE



Missing Values

	Column	Missing Values	Percentage
week_from	week_from	5,243	82.3853
week_to	week_to	5,243	82.3853
id_site	id_site	0	0
code_module	code_module	0	0
code_presentation	code_presentation	0	0
activity_type	activity_type	0	0

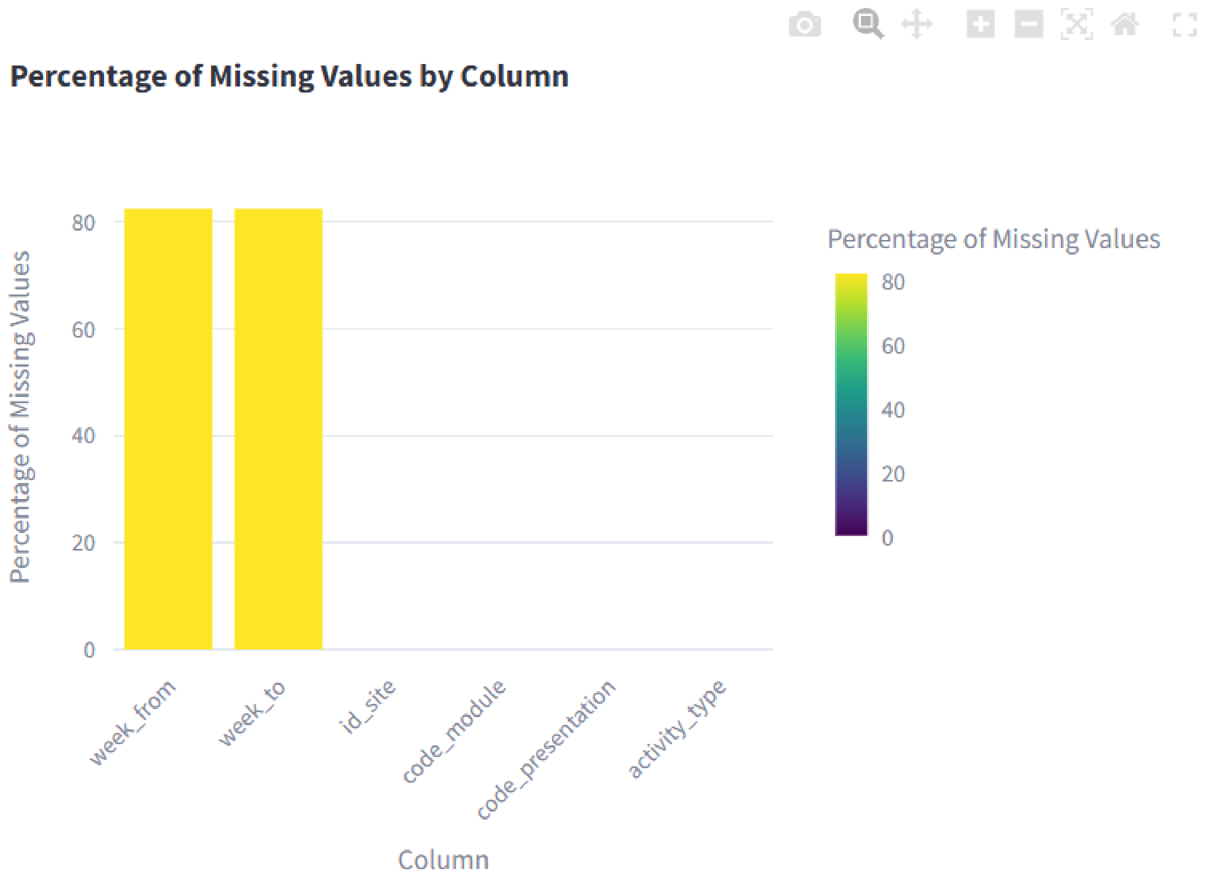
Missing Values Bar Chart



Missing Values Percentage

	Column	Percentage
week_from	week_from	82.3853
week_to	week_to	82.3853
id_site	id_site	0
code_module	code_module	0
code_presentation	code_presentation	0
activity_type	activity_type	0

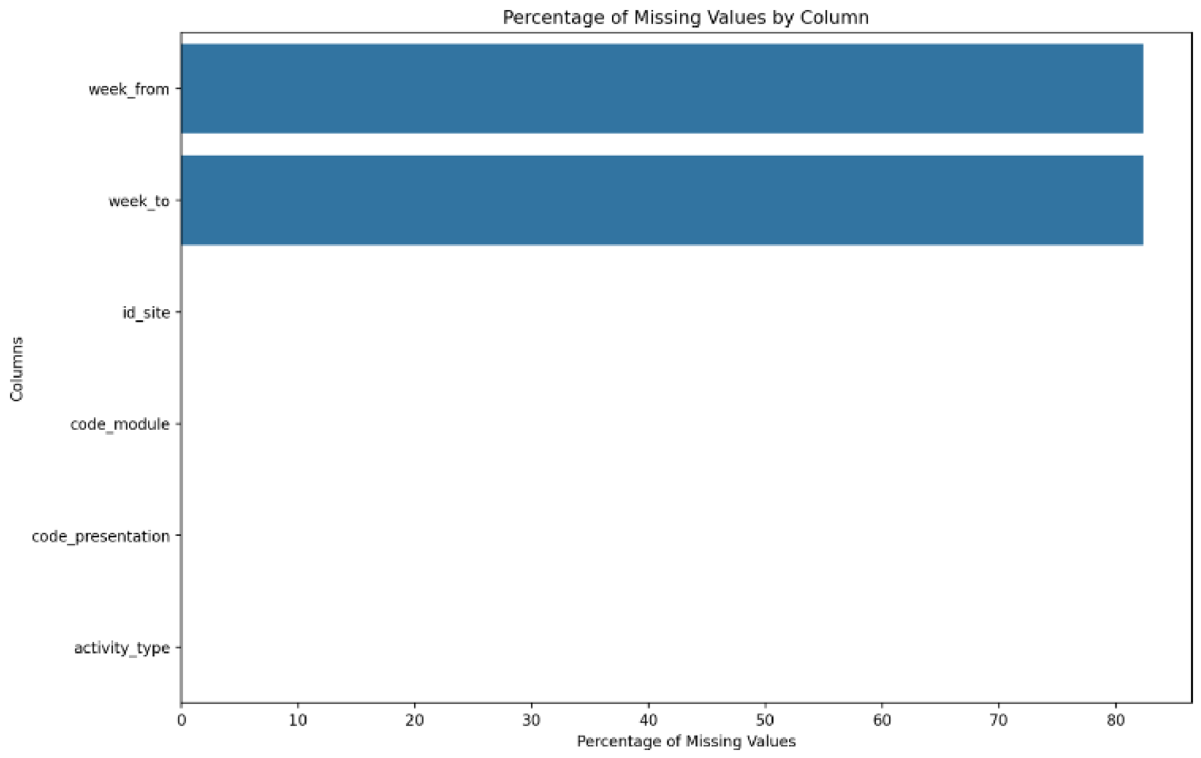
Missing Values Percentage Bar Chart



Missing Values DataFrame

	Column	Missing Values	Percentage
week_from	week_from	5,243	82.3853
week_to	week_to	5,243	82.3853
id_site	id_site	0	0
code_module	code_module	0	0
code_presentation	code_presentation	0	0
activity_type	activity_type	0	0

Missing Values Visualization



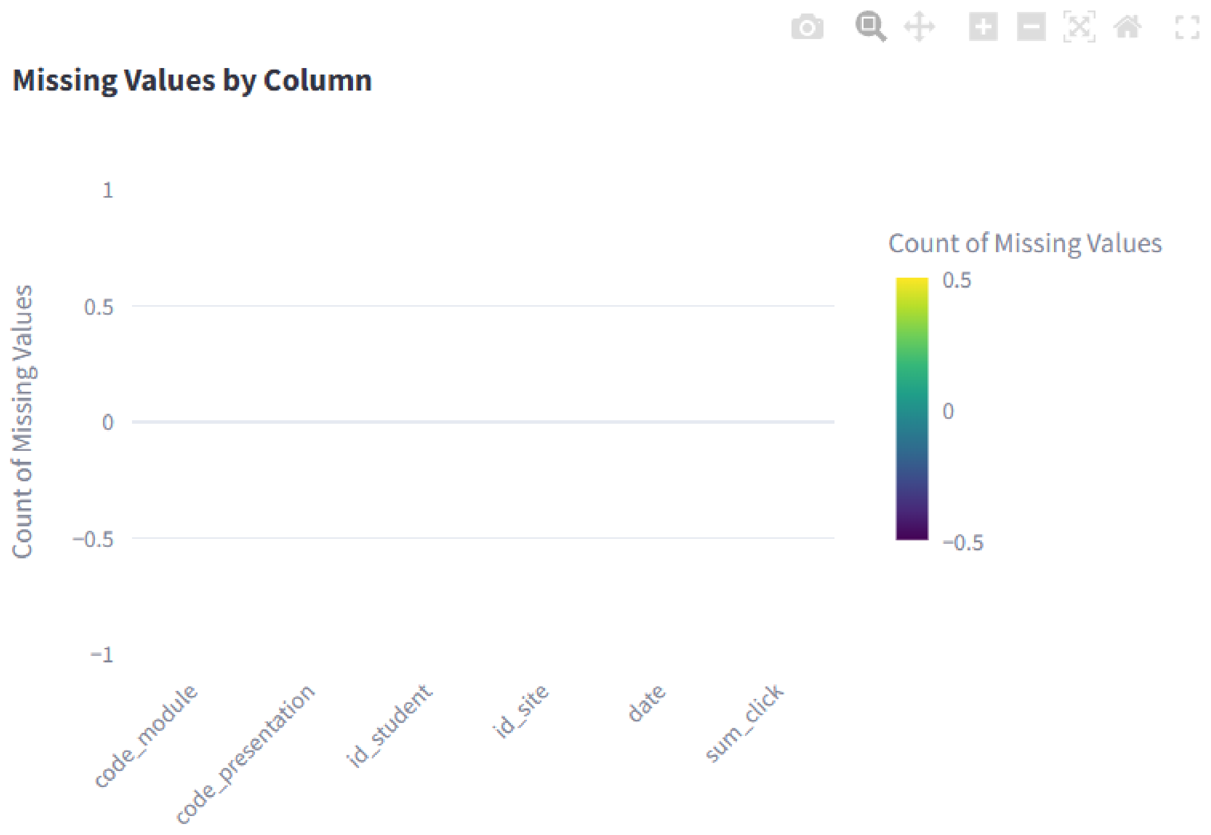
Missing Value Treatment for VLE

Since the 'Week From' and 'Week to' has around 82% missing values, it is best to drop these columns.

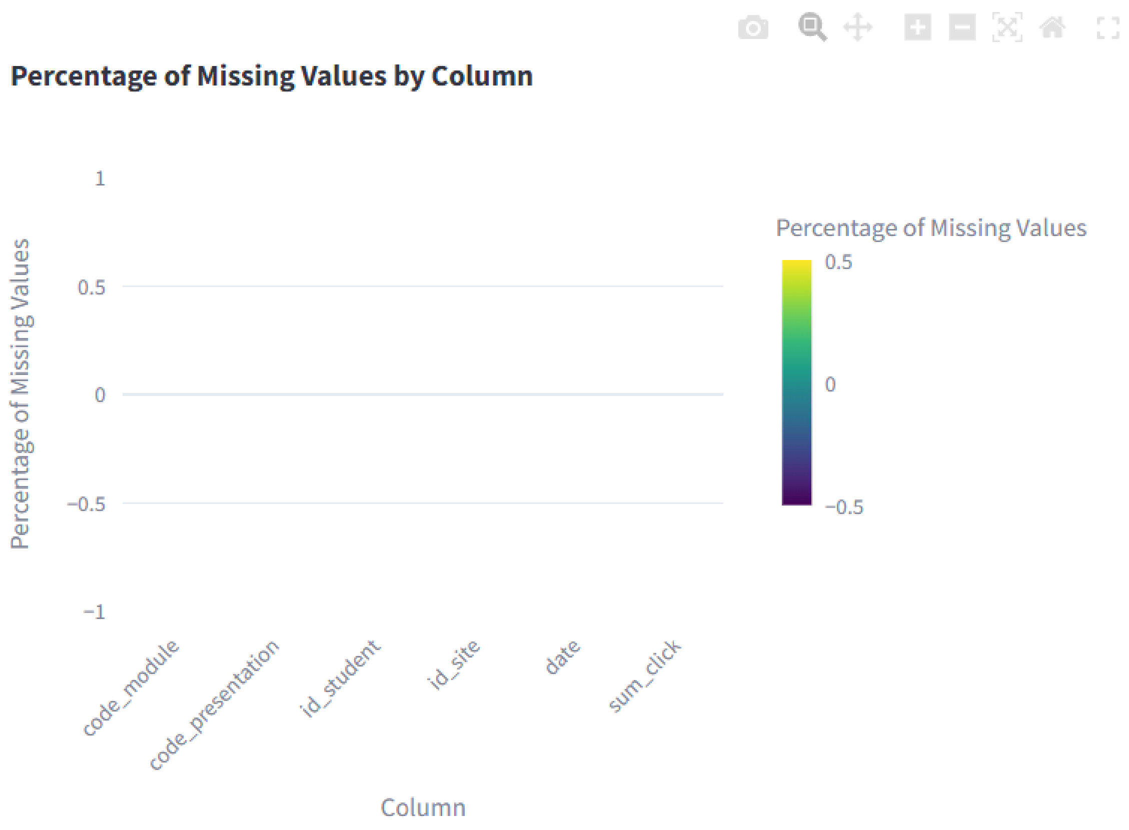
vle.drop(columns=['week_from','week_to'],inplace=True)

4.3.1.2 Student Interactions

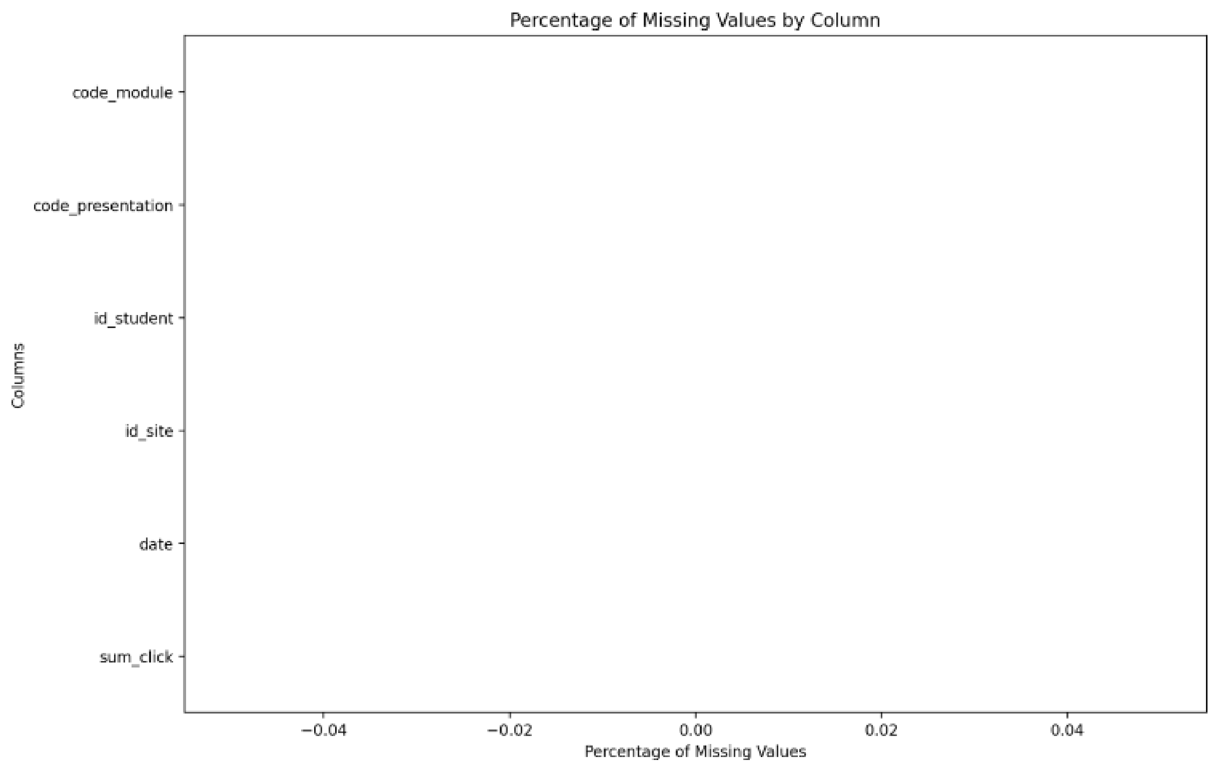
Missing Values Bar Chart



Missing Values Percentage Bar Chart



Missing Values Visualization

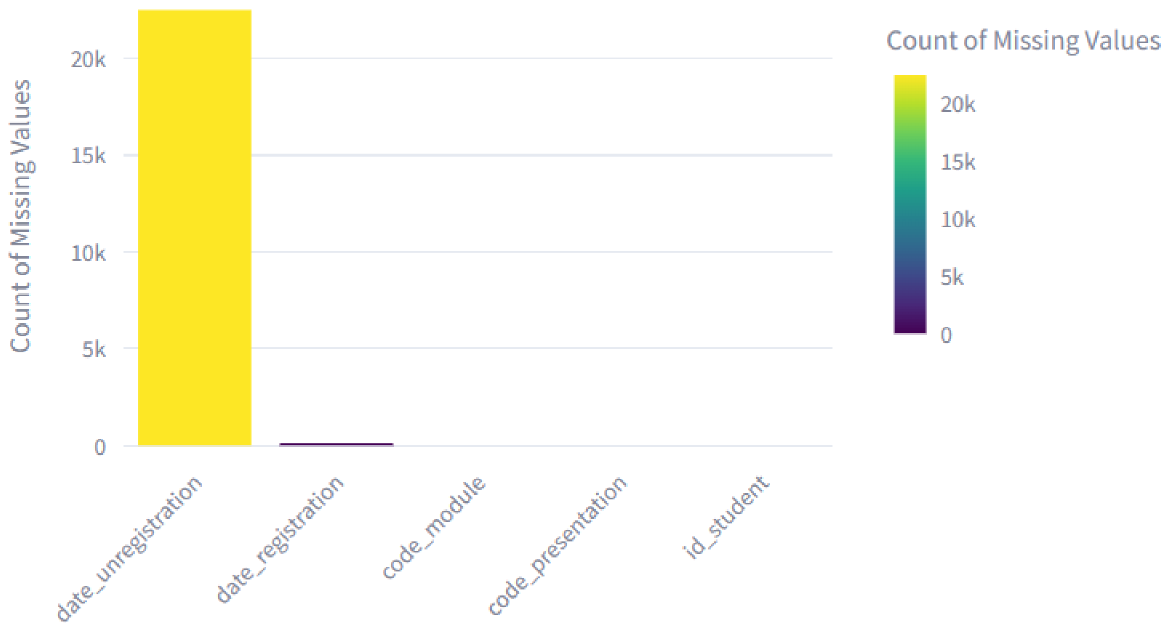


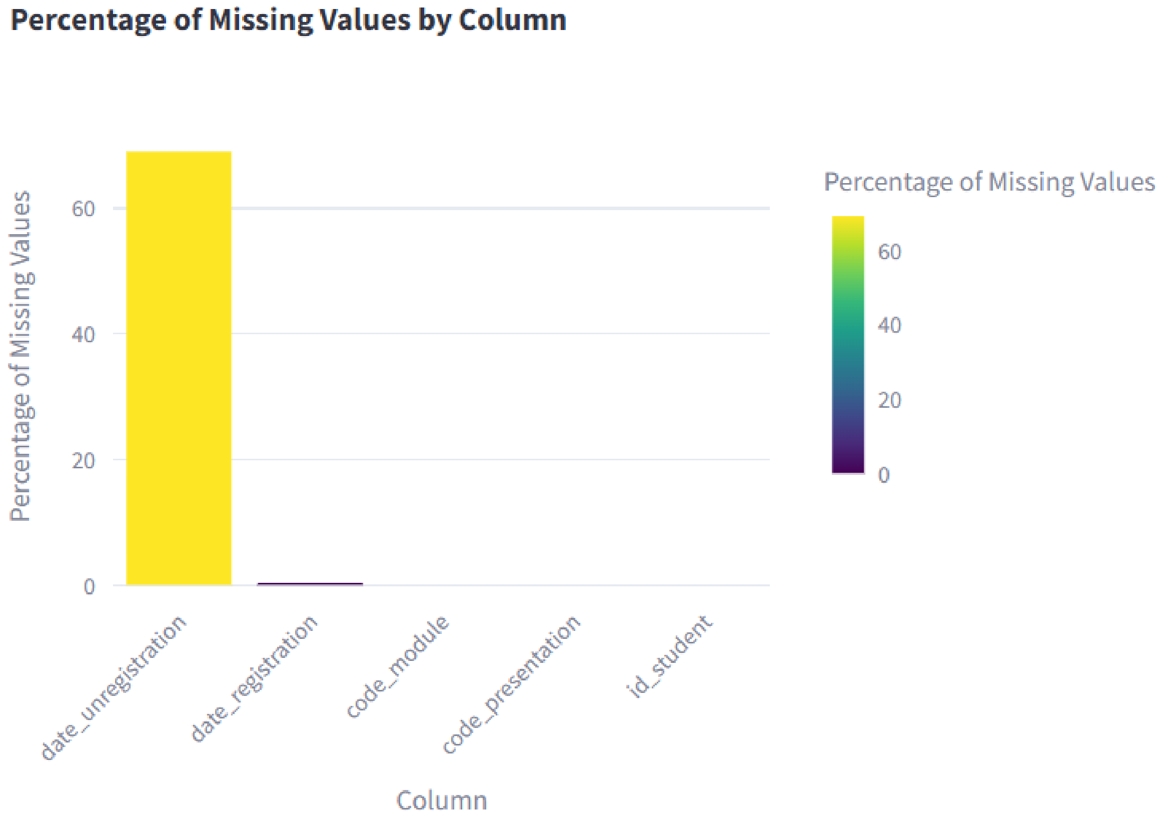
No Missing Values

Transforming date to datetime

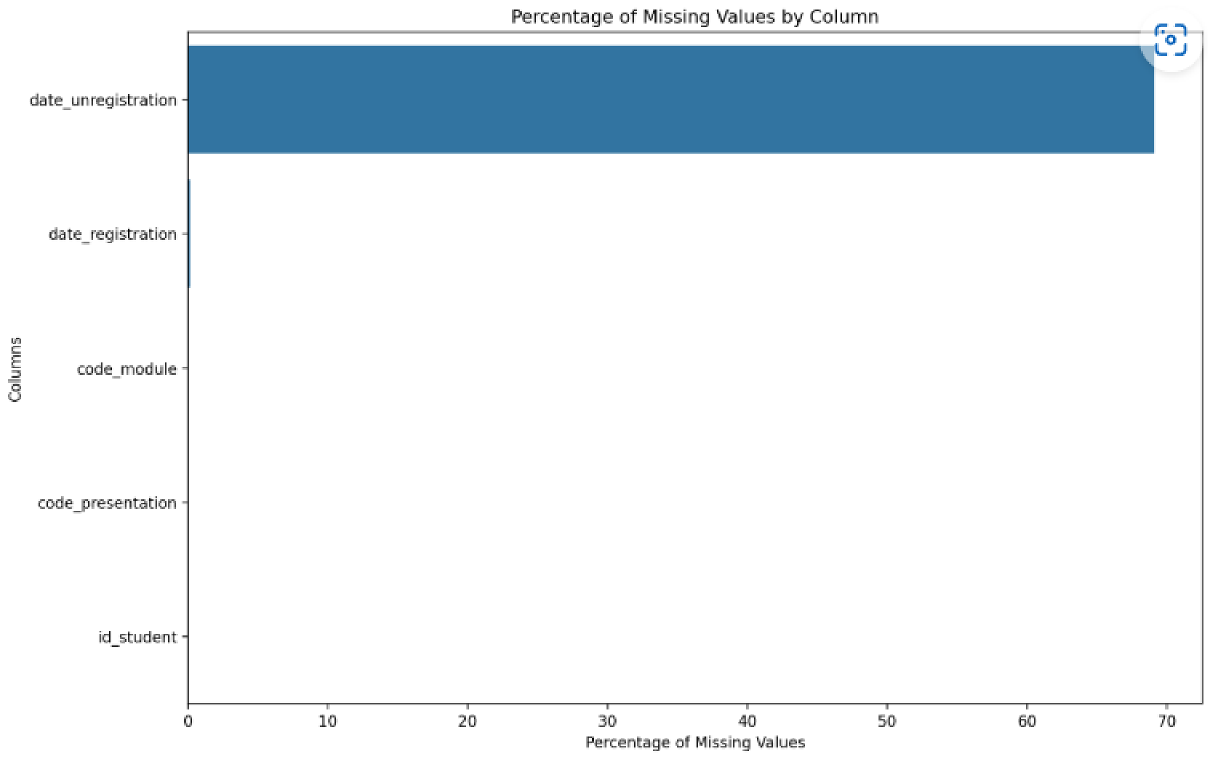
- Define the start date start_date = pd.Timestamp('2000-01-01')
- Convert 'date' to datetime vle_stu['date'] = vle_stu['date'].apply(lambda x: start_date + pd.Timedelta(days=x-1))

4.3.1.3 Student Registrations





Missing Values Visualization



Missing Value Treatment for Student Registration

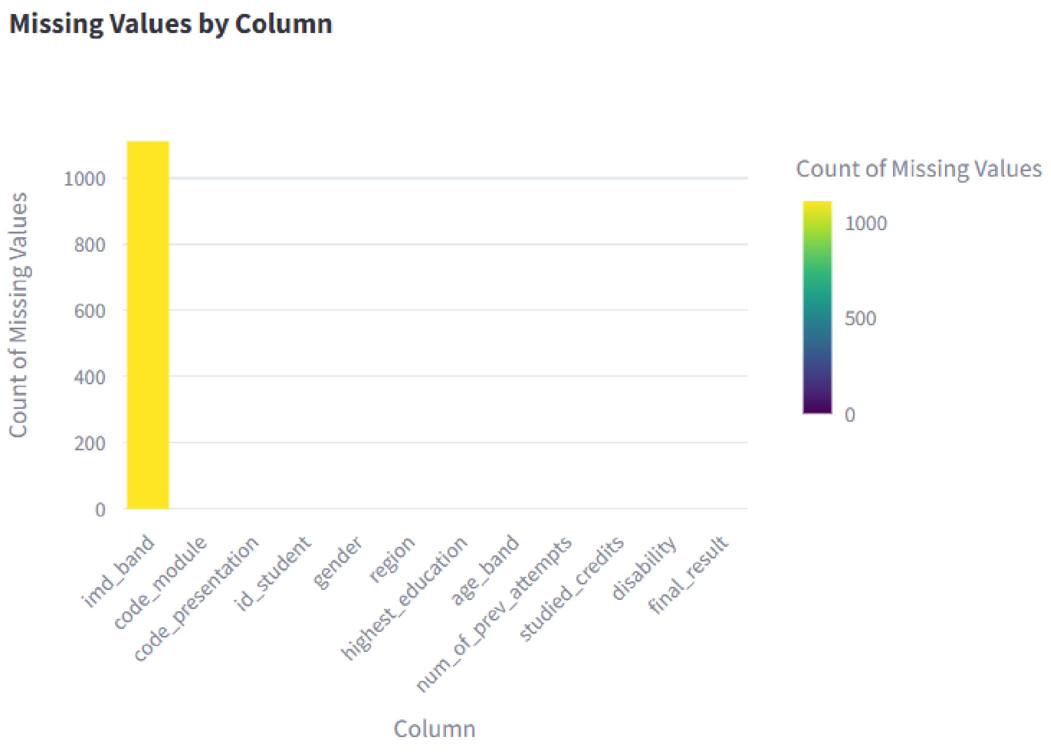
Since the 'date_unregistration' has around 65% missing values, it is best to drop these columns.

Transforming date_registration to datetime

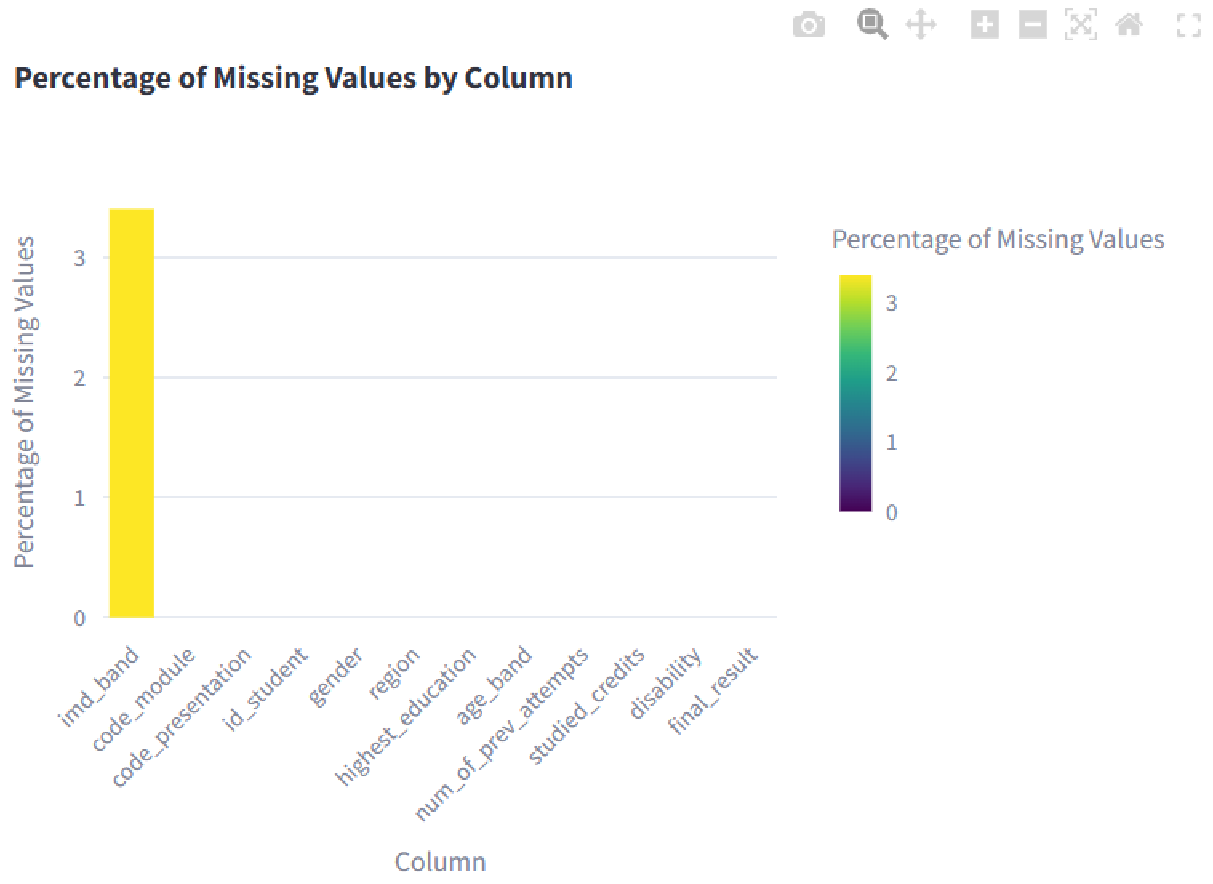
Define the start date start_date = pd.Timestamp('2000-01-01')

Convert 'date_submitted' to datetime reg_stu['date_registration'] = reg_stu['date_registration'].apply(lambda x: start_date + pd.Timedelta(days=x-1))

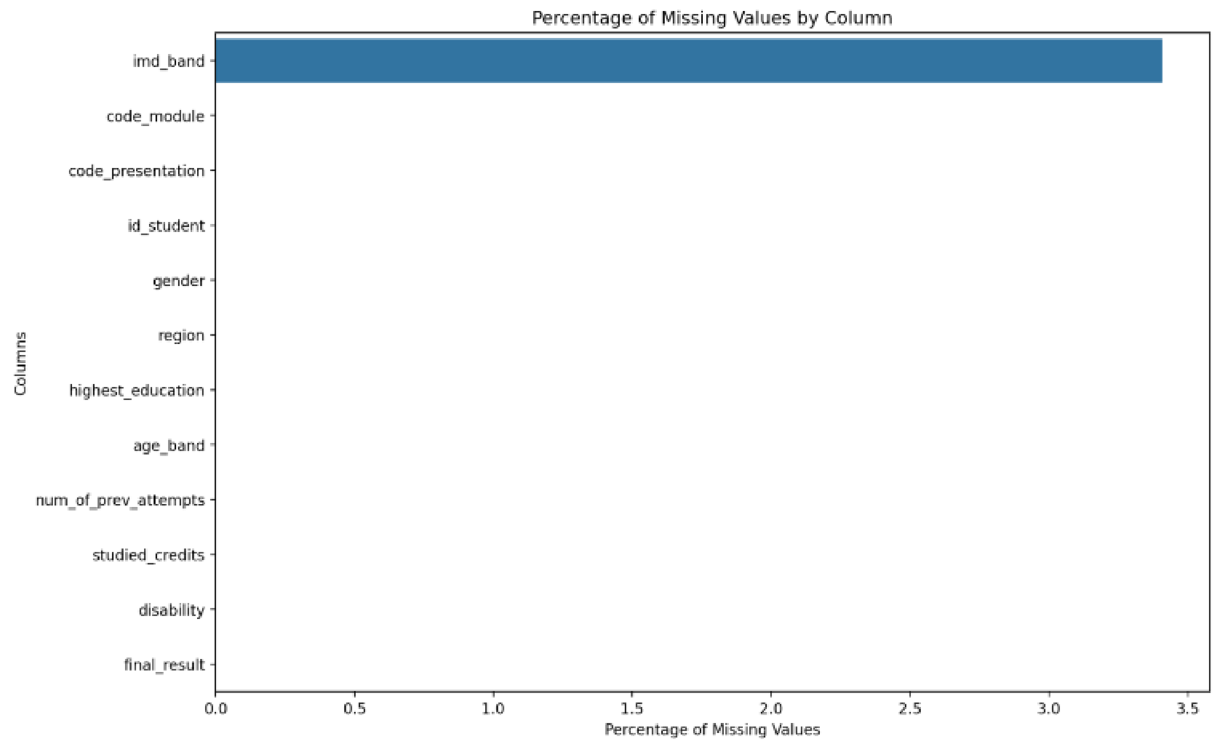
4.3.1.4 Student Info



Missing Values Percentage Bar Chart



Missing Values Visualization

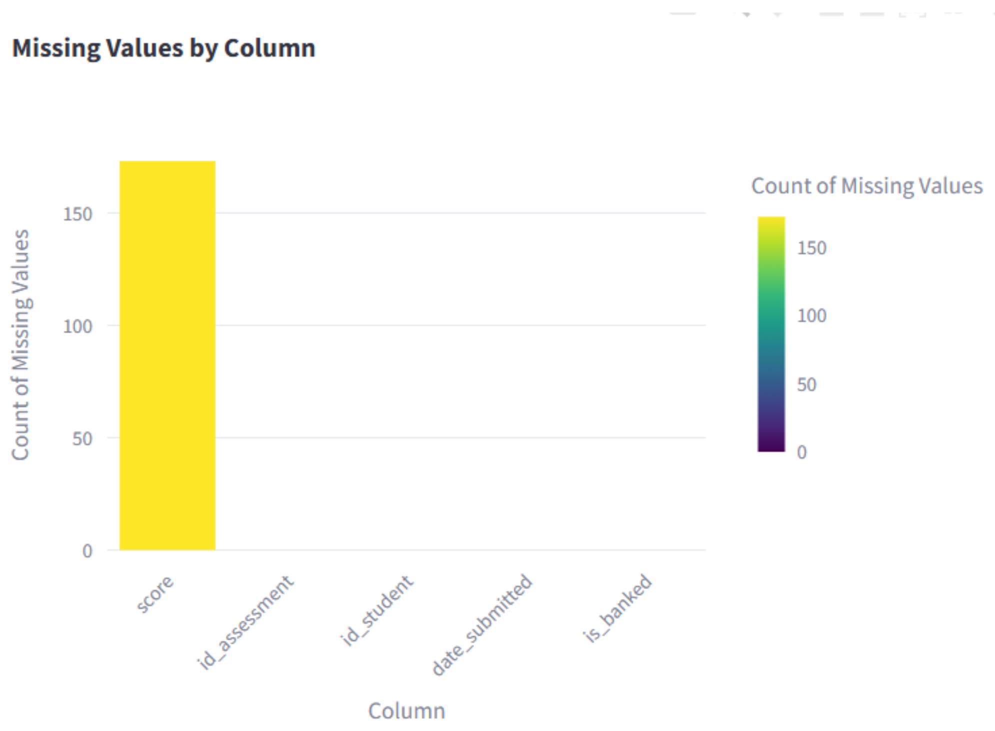


Missing Value Treatment for Student Info

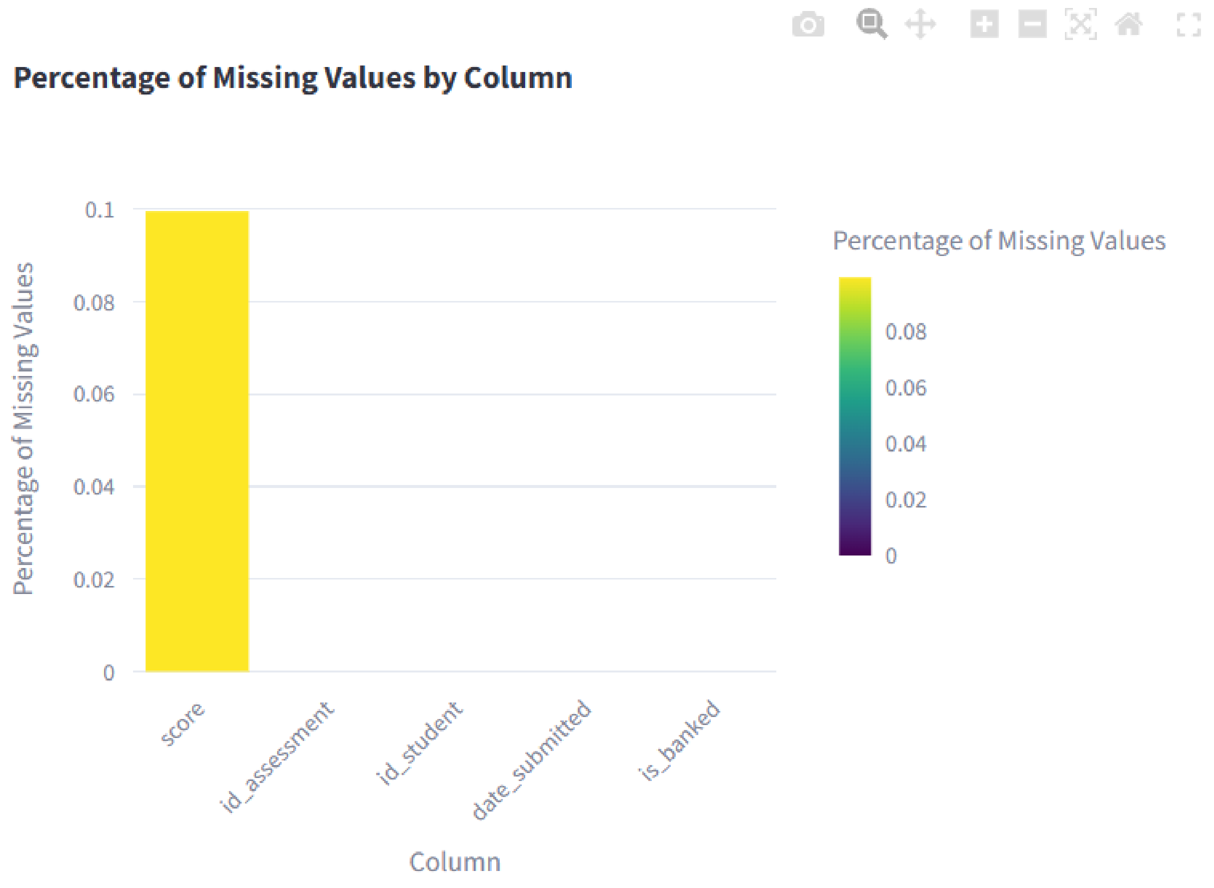
Since 'imd_band' has missing value it's better to impute it with dummy value

info_stu['imd_band'].fillna('unknown', inplace=True) # Filling missing values with 'unknown'

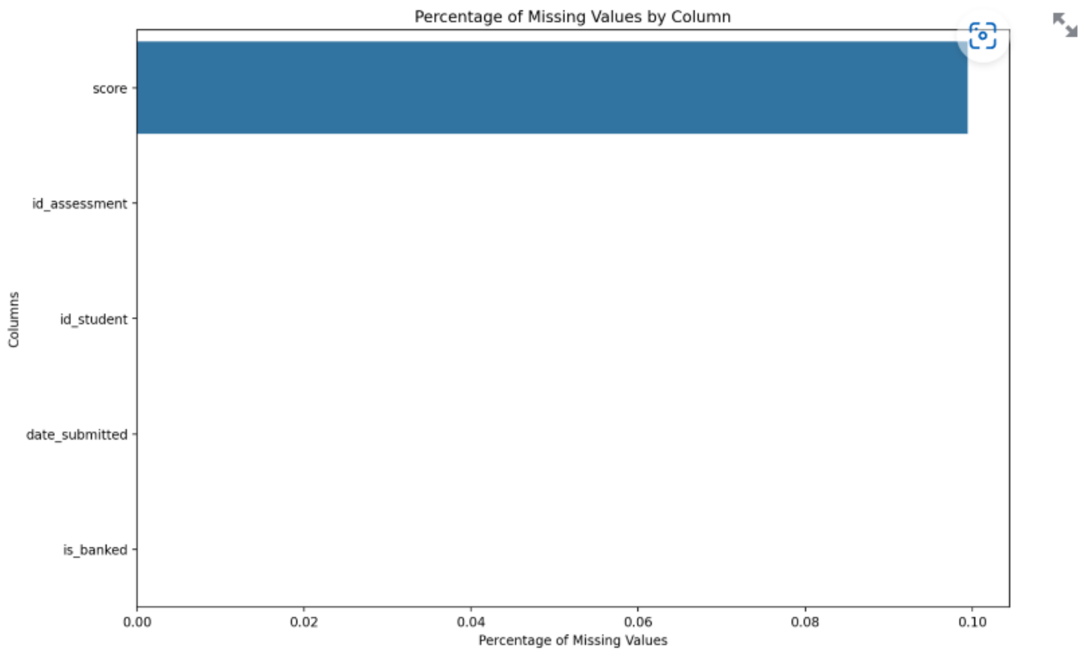
4.3.1.5 Student Assessments



Missing Values Percentage Bar Chart



Missing Values Visualization



Missing Value Treatment for Student Assessments

Score has missing values. Imputing the missing value with the mean of the score achieved that particular student

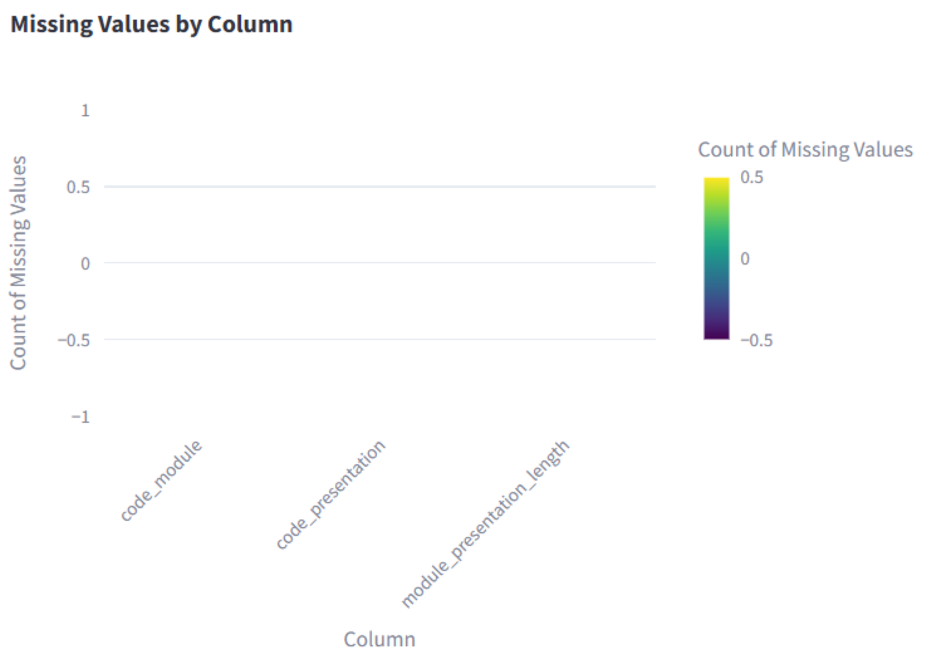
student_means = as_stu.groupby('id_student')['score'].transform('mean')
as_stu['score'].fillna(student_means, inplace=True)

Transforming date_submitted to datetime

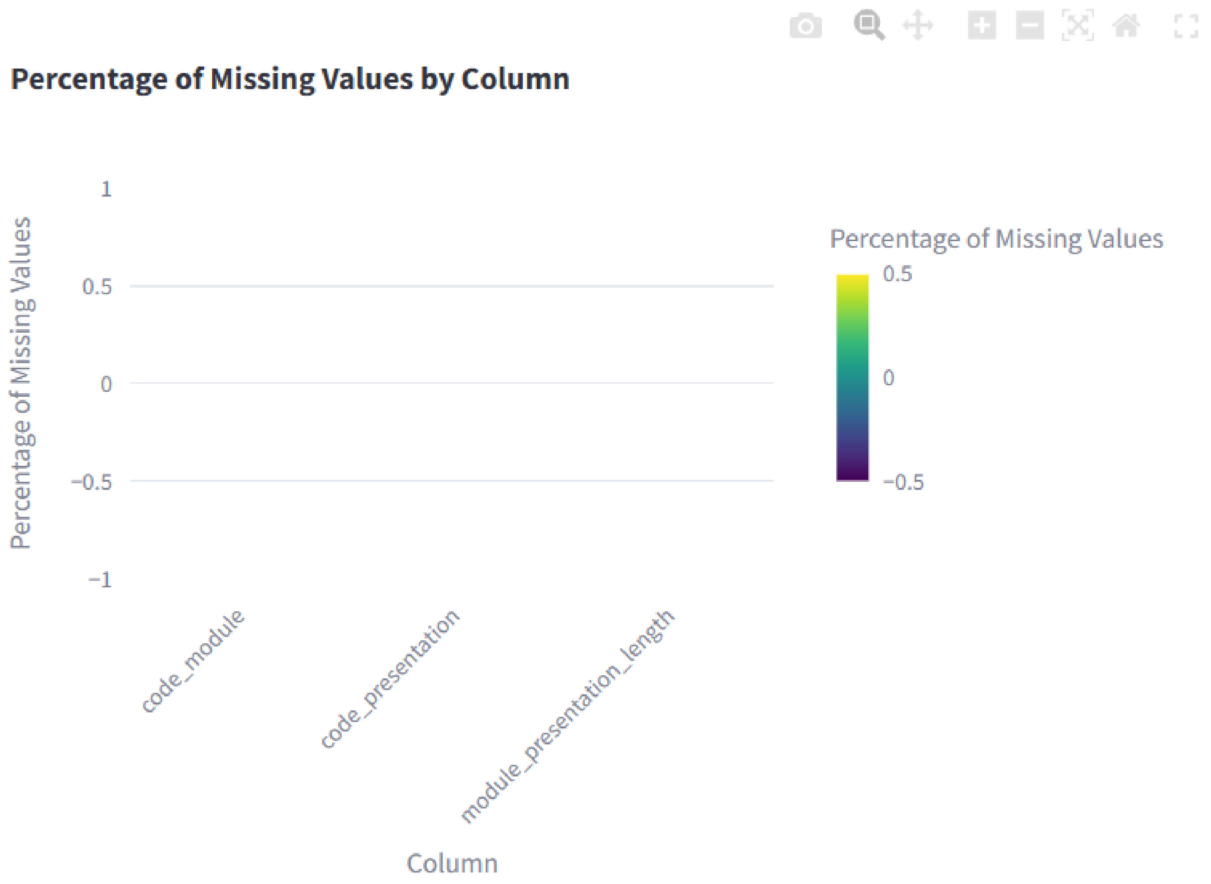
Define the start date start_date = pd.Timestamp('2000-01-01')

Convert 'date_submitted' to datetime as_stu['date_submitted'] = as_stu['date_submitted'].apply(lambda x: start_date + pd.Timedelta(days=x-1))

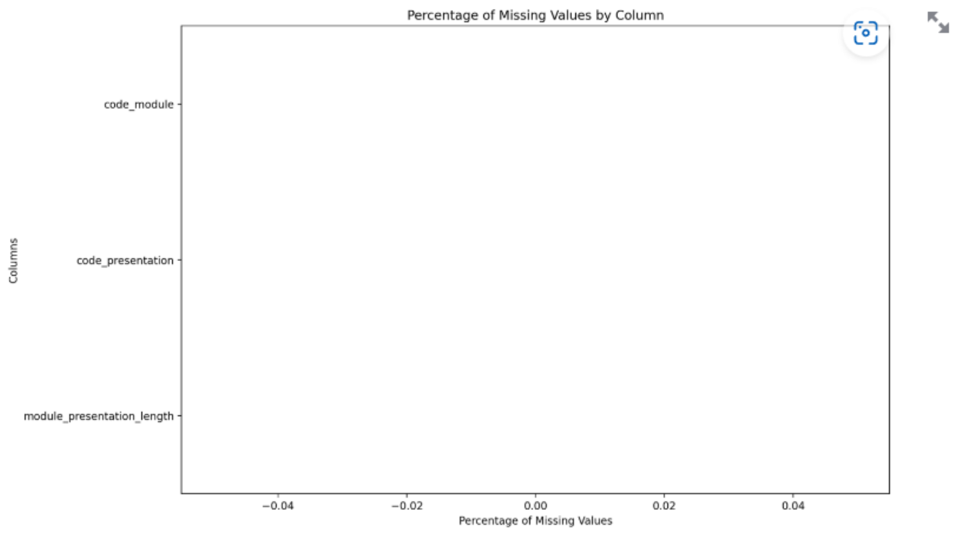
4.3.1.6 Courses



Missing Values Percentage Bar Chart



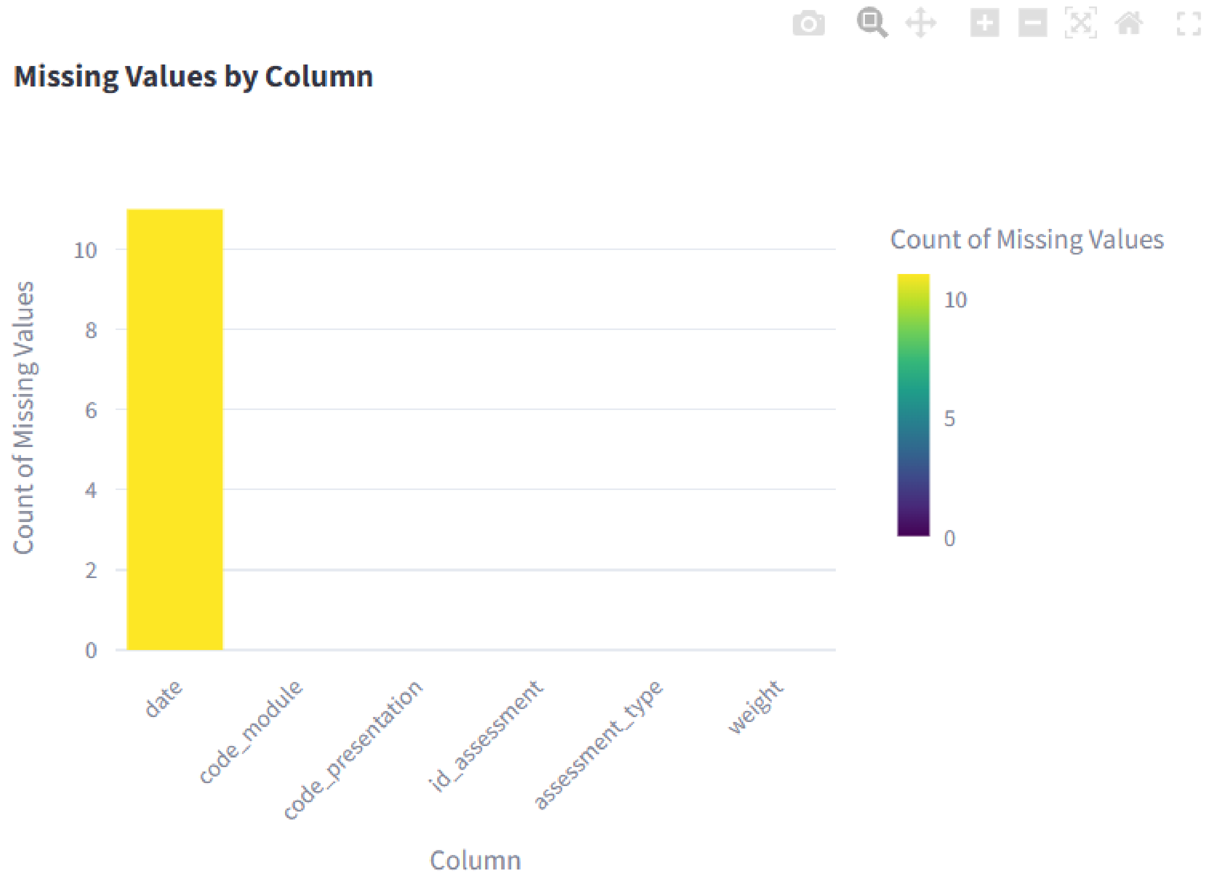
Missing Values Visualization



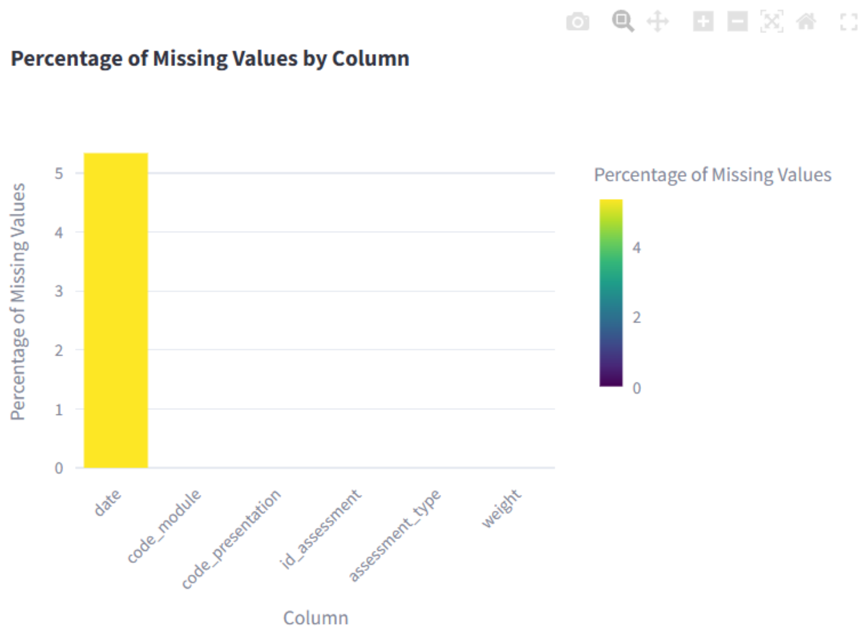
No Missing Values

4.3.1.7 Assessments

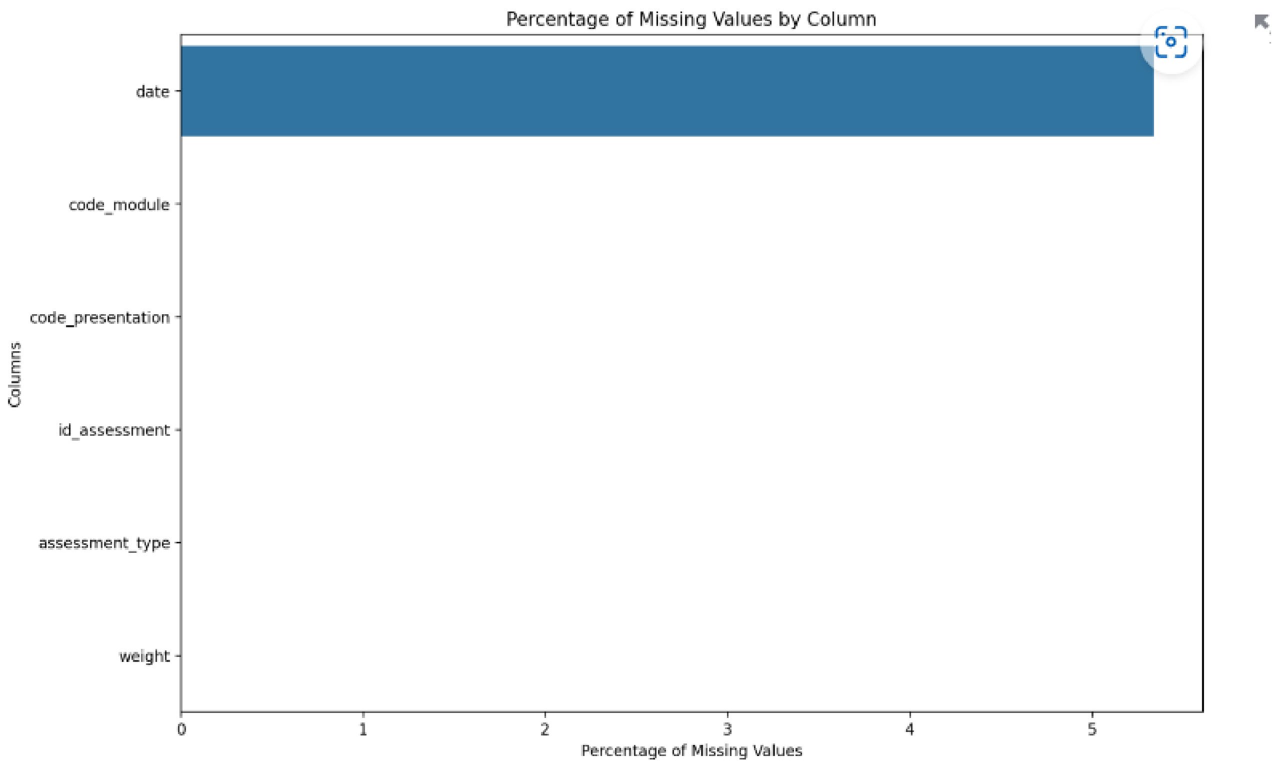
Missing Values Bar Chart



Missing Values Percentage Bar Chart



Missing Values Visualization

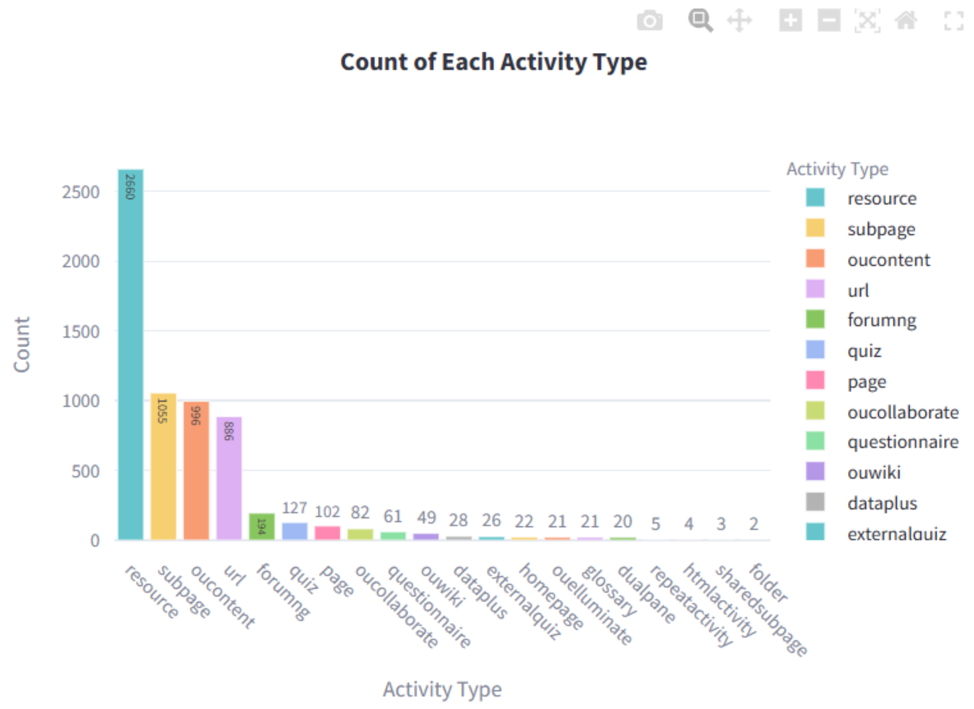


'Date' has around 5% missing values so it's better that we drop these missing values as we don't if this exam has been conducted or not

4.4 EXPLORATORY DATA ANALYSIS

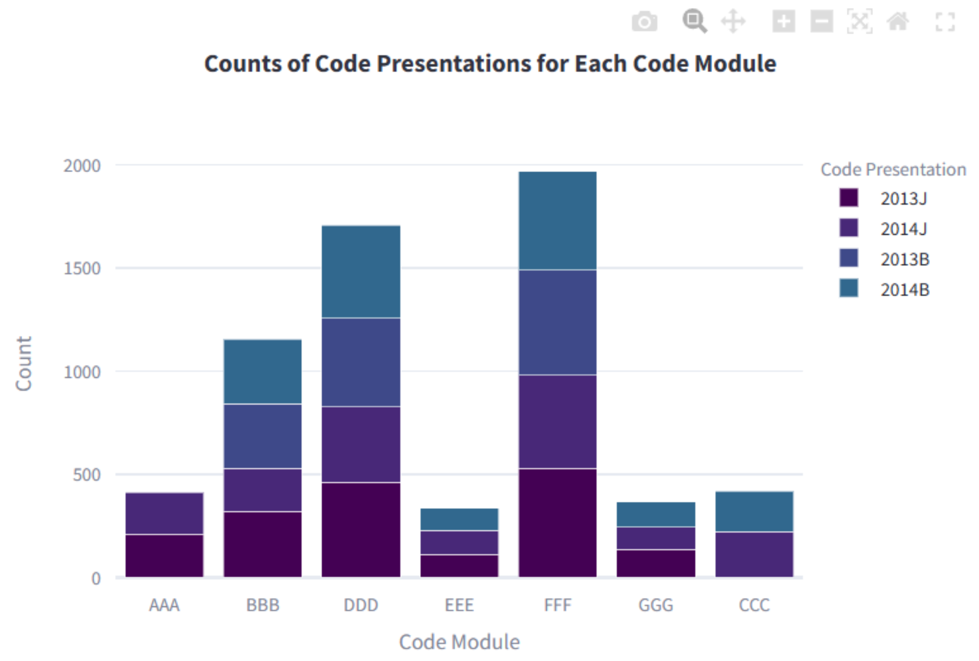
In this section the analysis was made on each table representing the information and concluding some meaningful results.

4.4.1 VLE

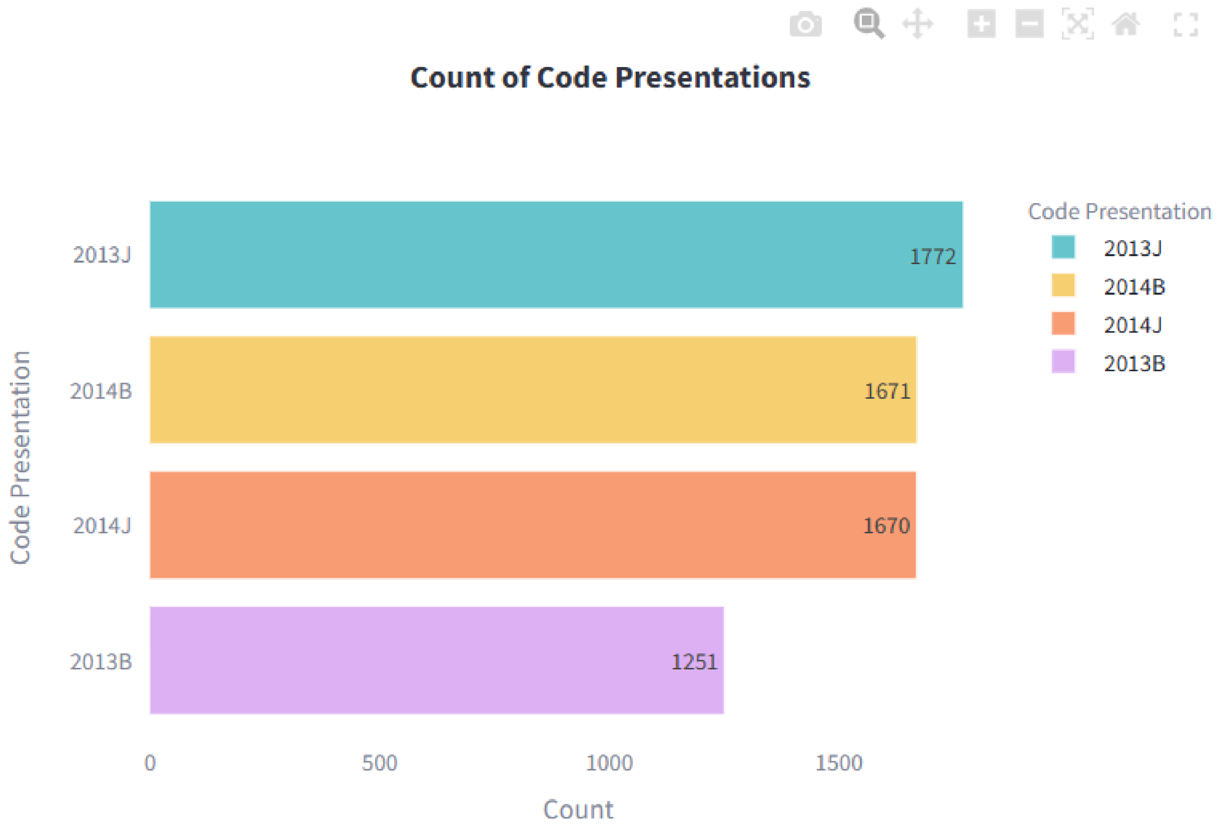


Università degli Studi di Napoli Federico II

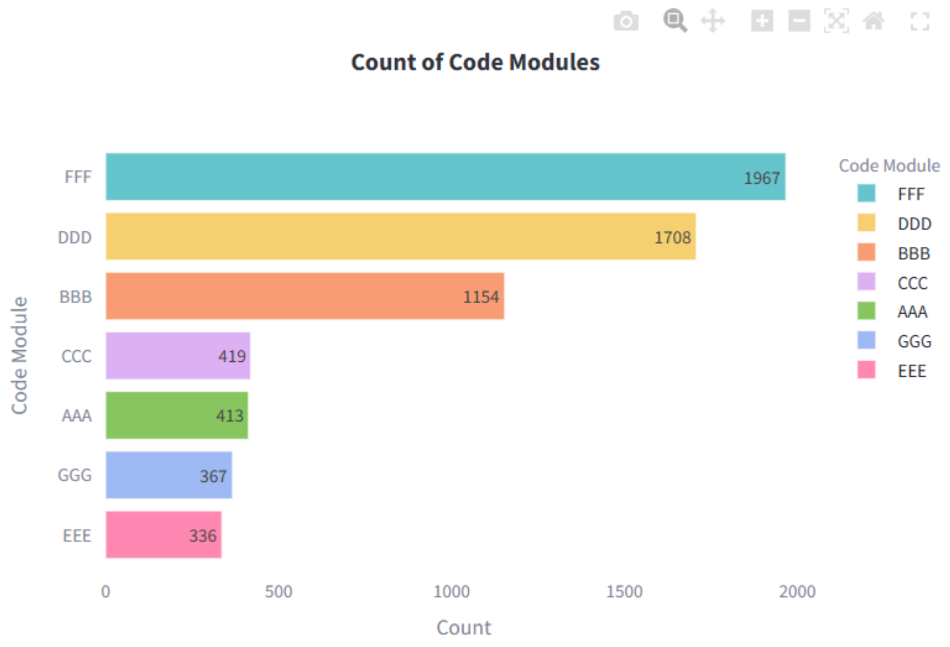
Counts of Code Presentations for Each Code Module



Count of Code Presentations

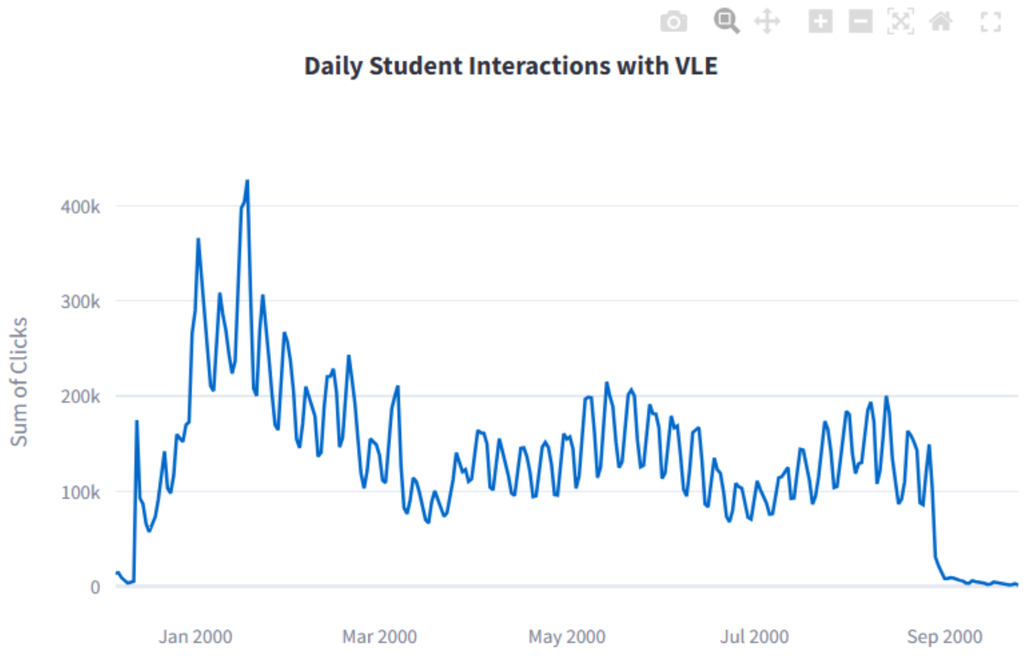


Count of Code Modules



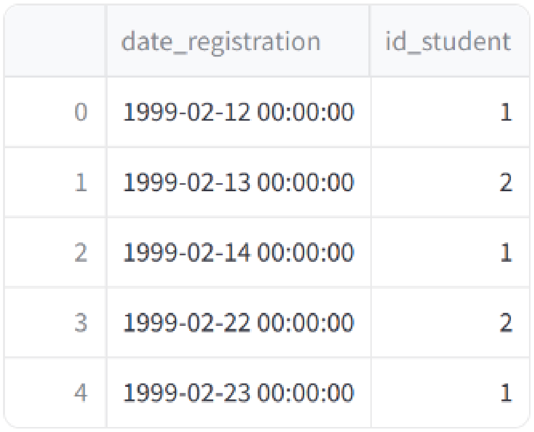
4.4.2 Student Interactions

Time Series Plot of Daily Student Interactions with VLE

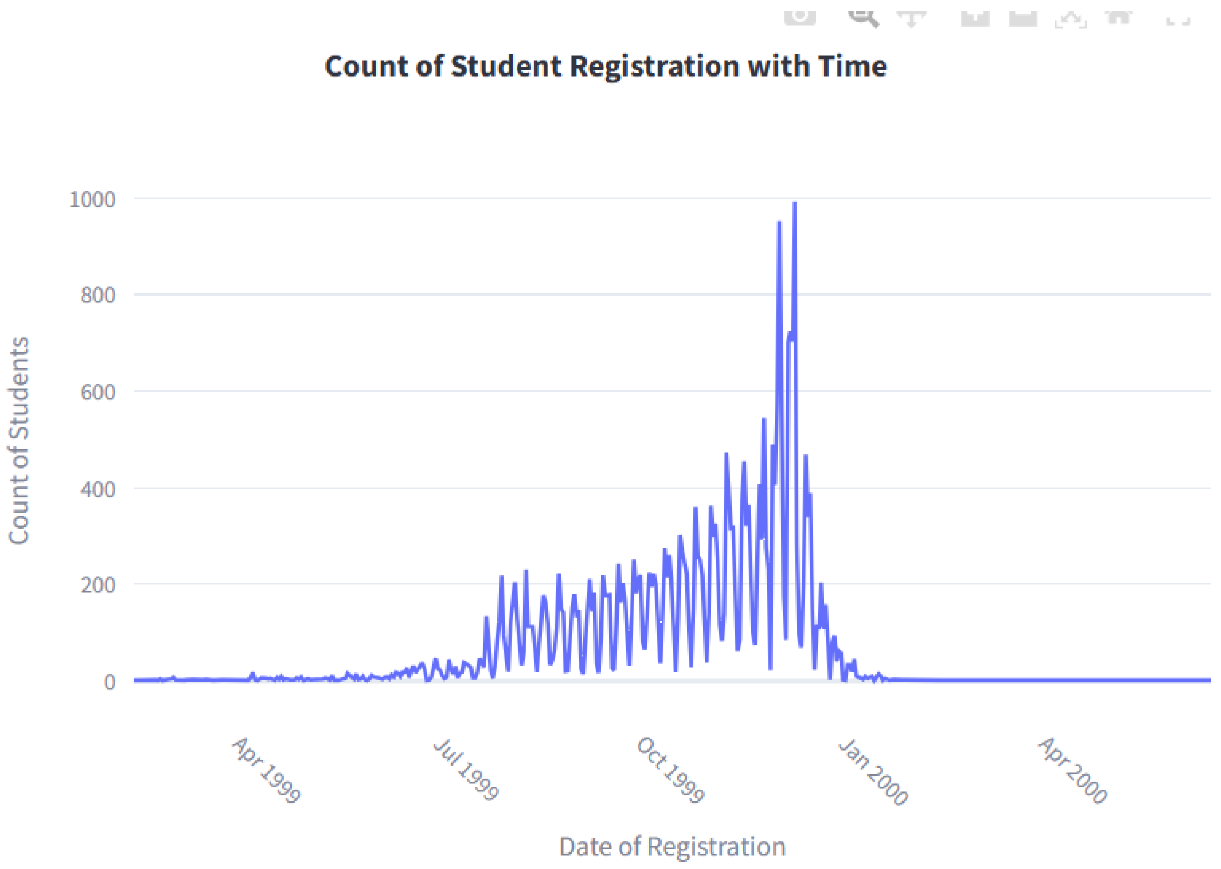


4.4.3 Student Registration

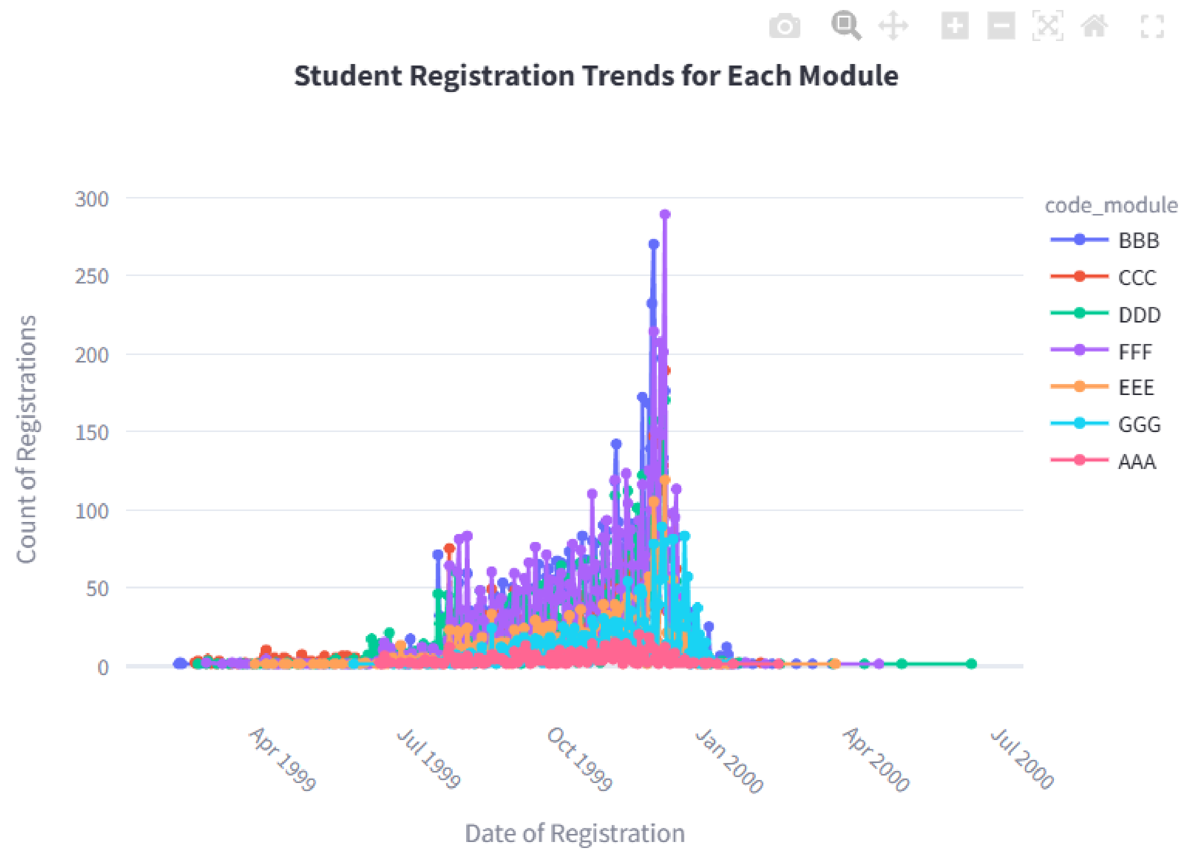
Registration Count of Students With Time



	date_registration	id_student
0	1999-02-12 00:00:00	1
1	1999-02-13 00:00:00	2
2	1999-02-14 00:00:00	1
3	1999-02-22 00:00:00	2
4	1999-02-23 00:00:00	1

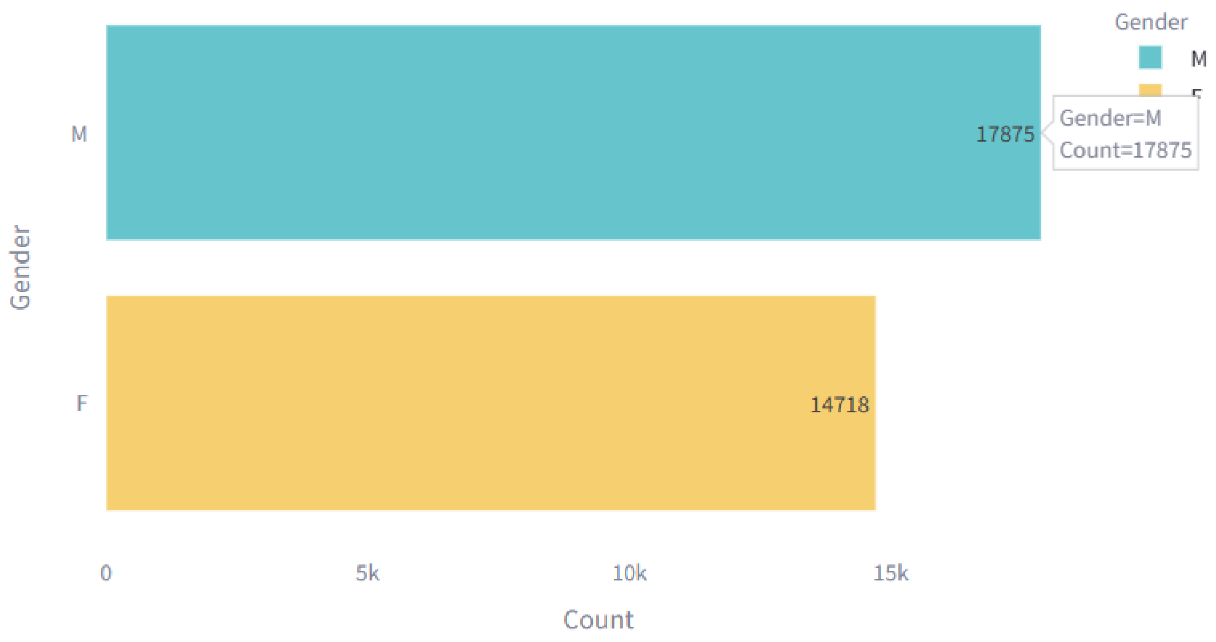


Registration Count Per Module of Students With Time

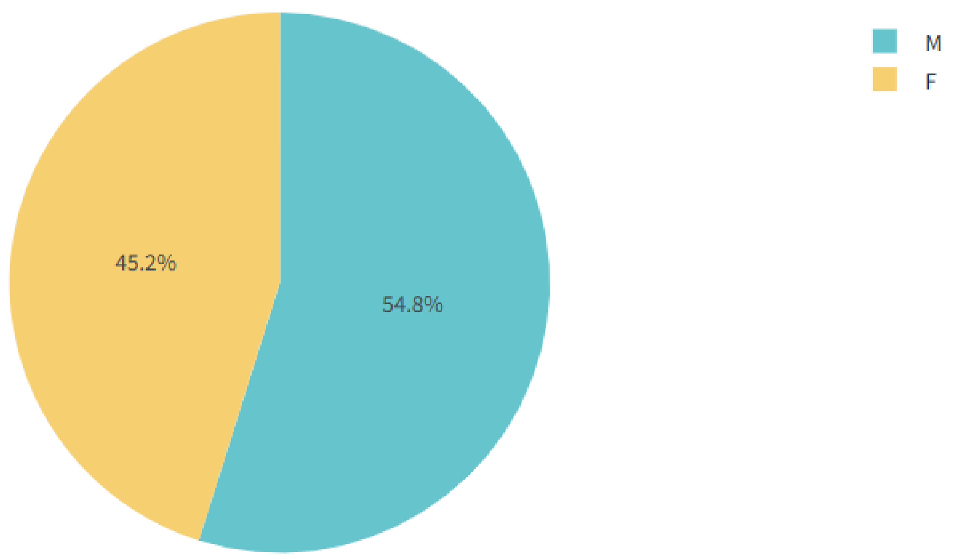


4.4.4 Student Info

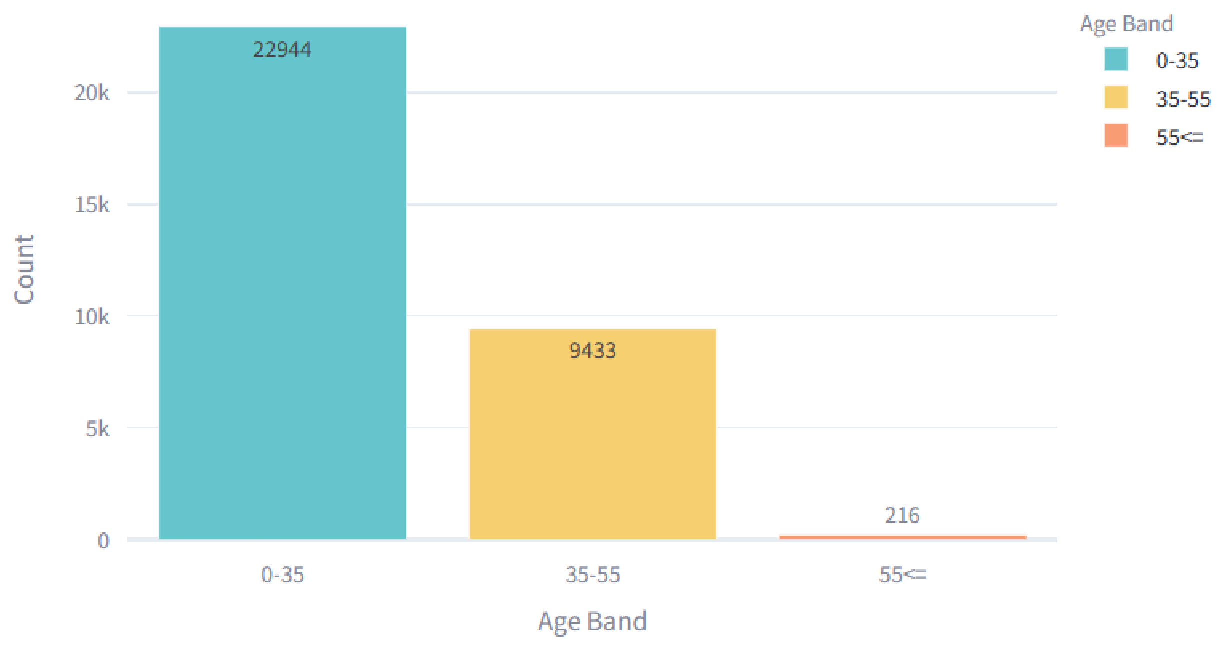
Count of Gender



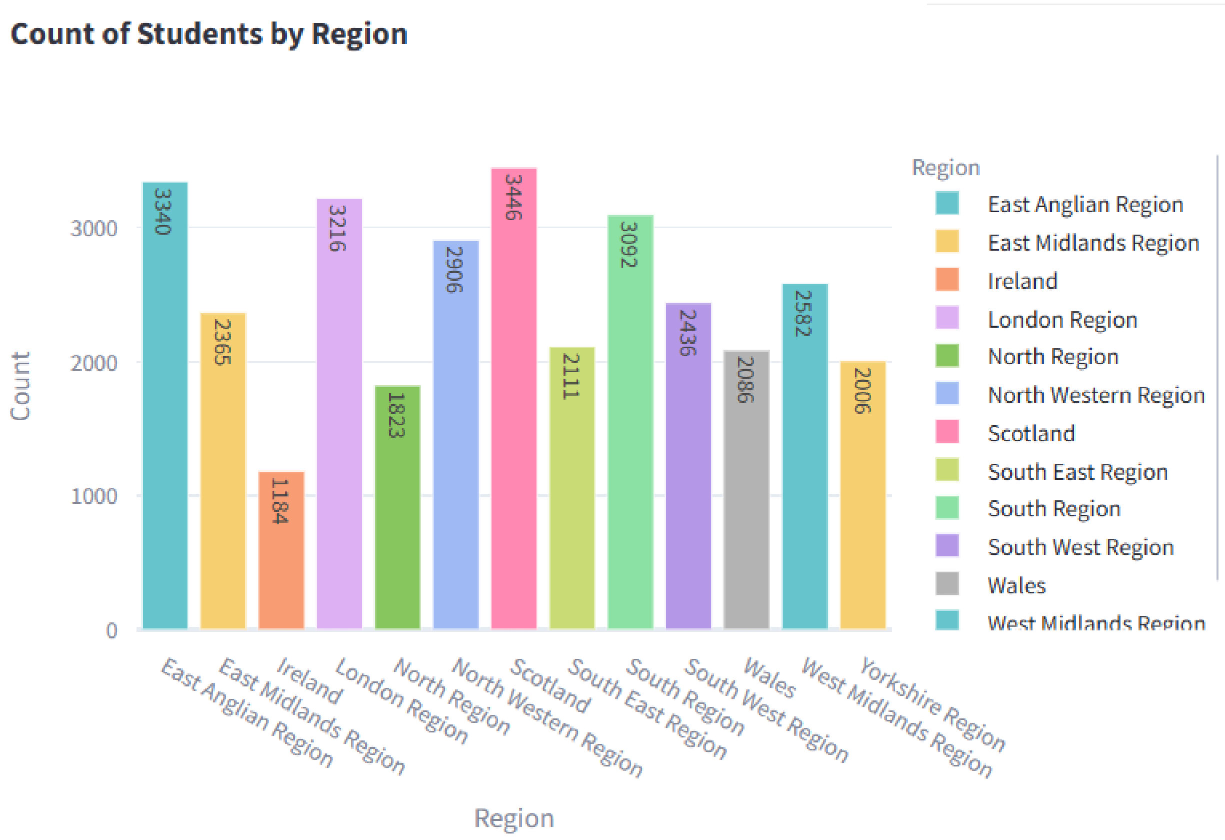
Distribution of Gender



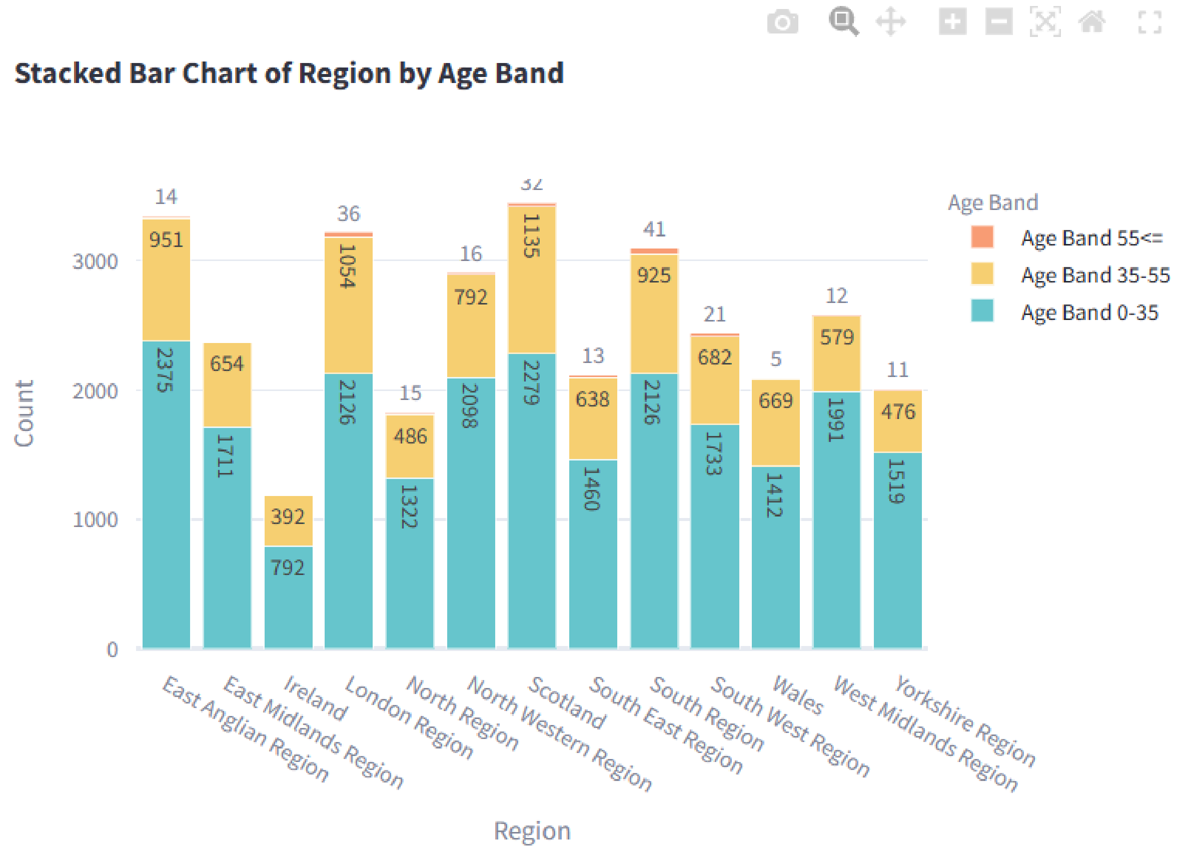
Count of Students by Age Band



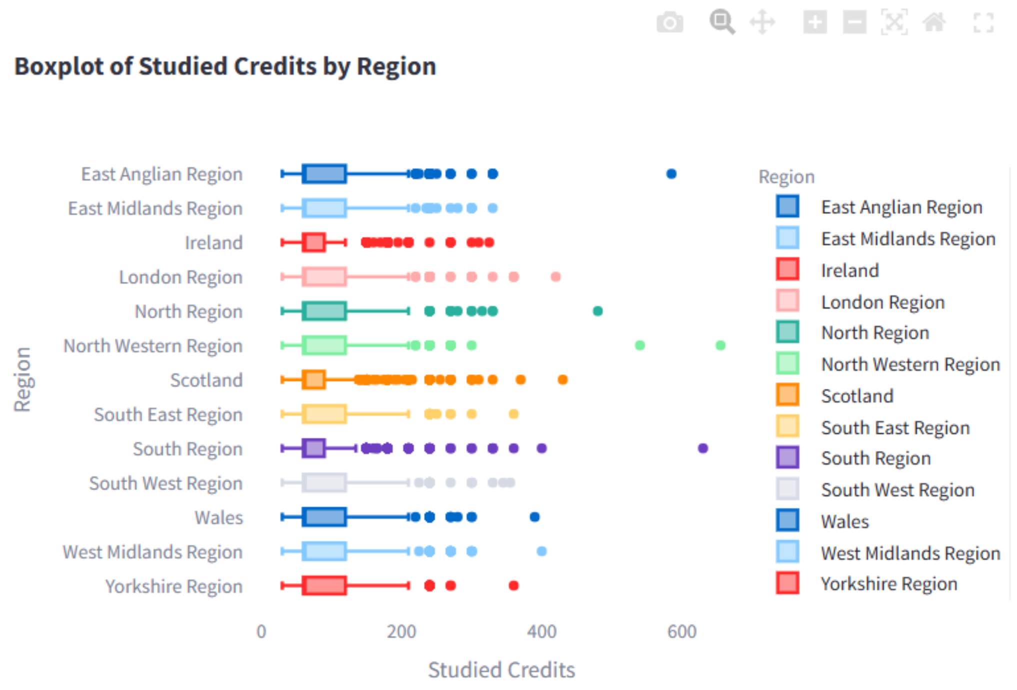
Università degli Studi di Napoli Federico II



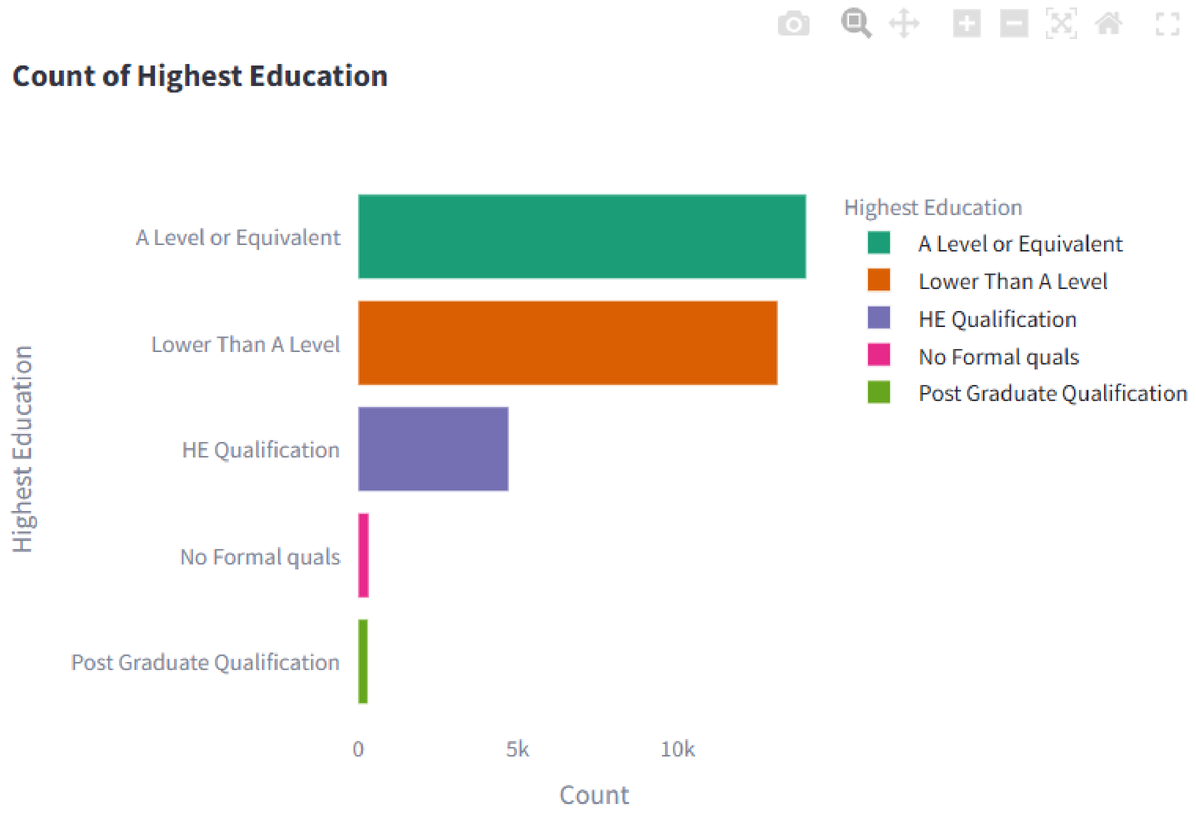
Stacked Bar Chart of Region by Age Band



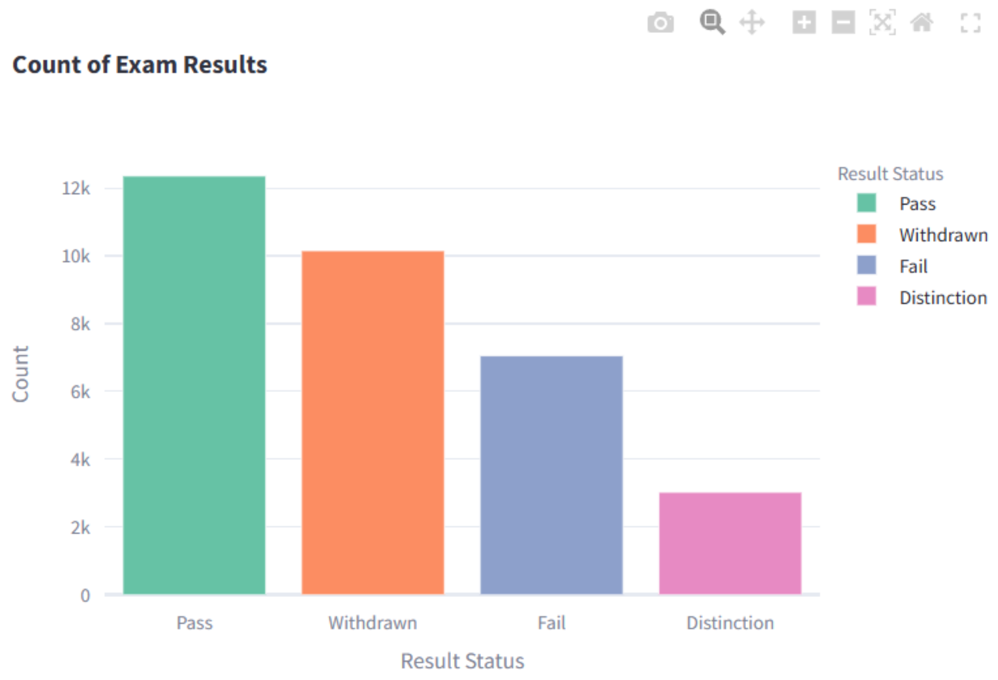
Box Plot of Studied Credits by Region



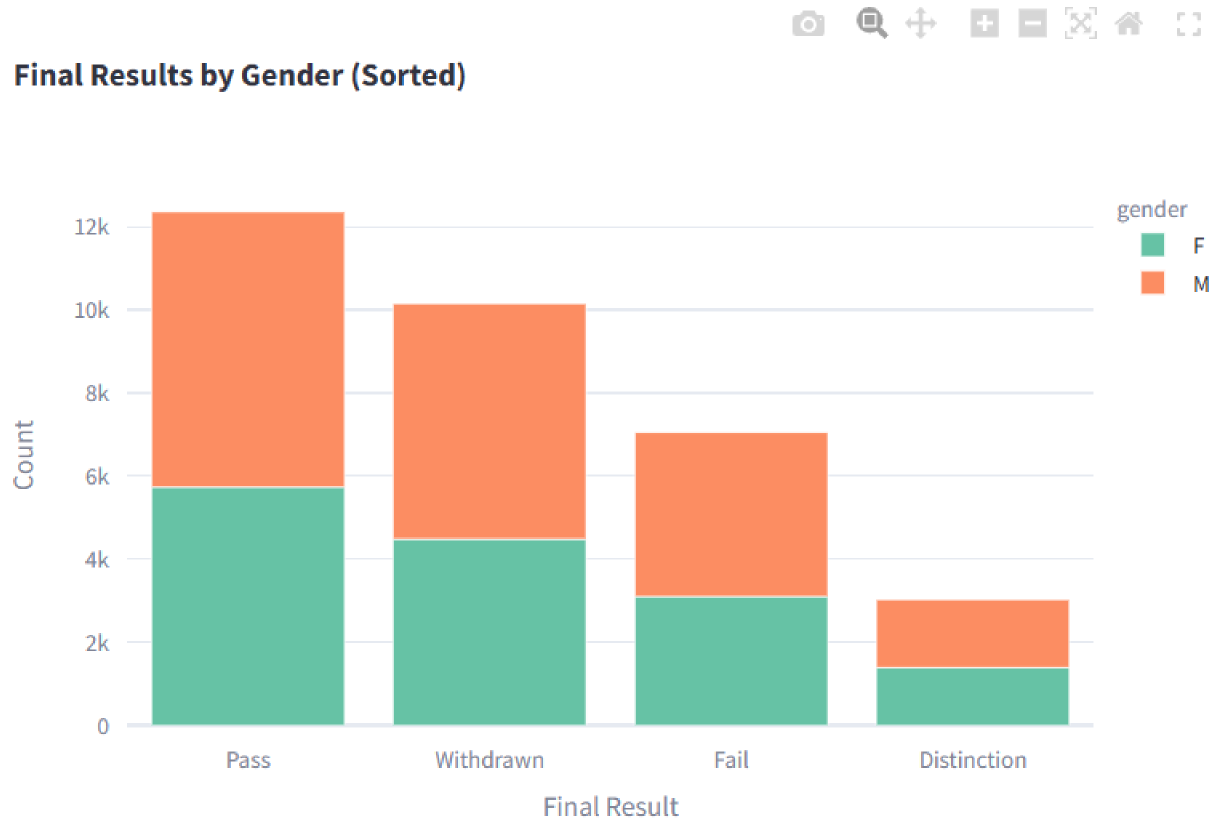
Count Plot of Highest Education



Count Of Exam Results

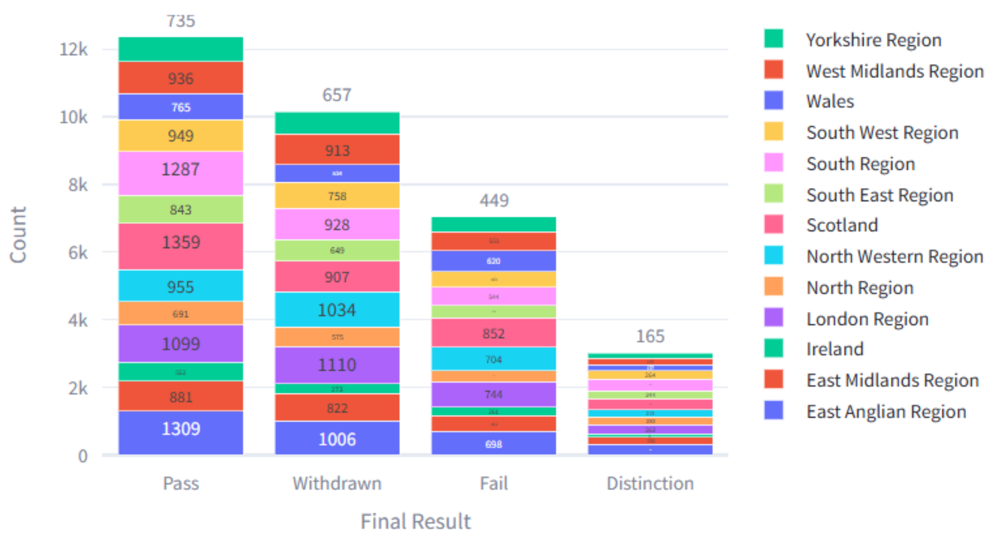


Final Results by Gender



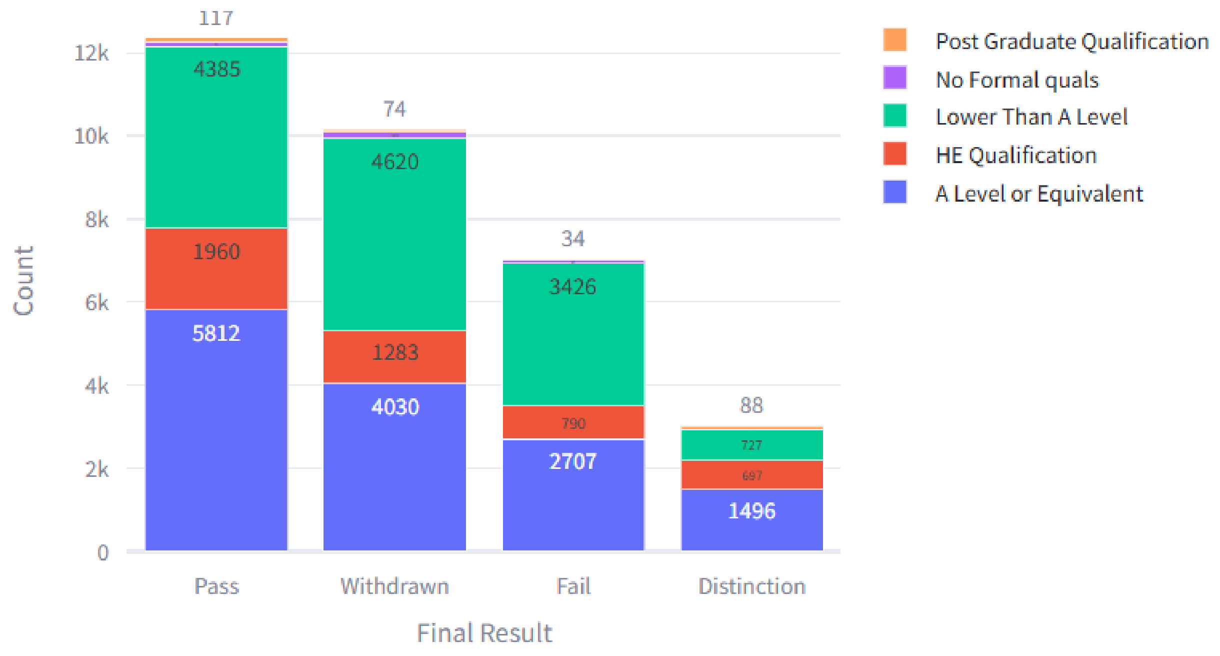
Final Results by Region

Final Results by Region (Stacked)

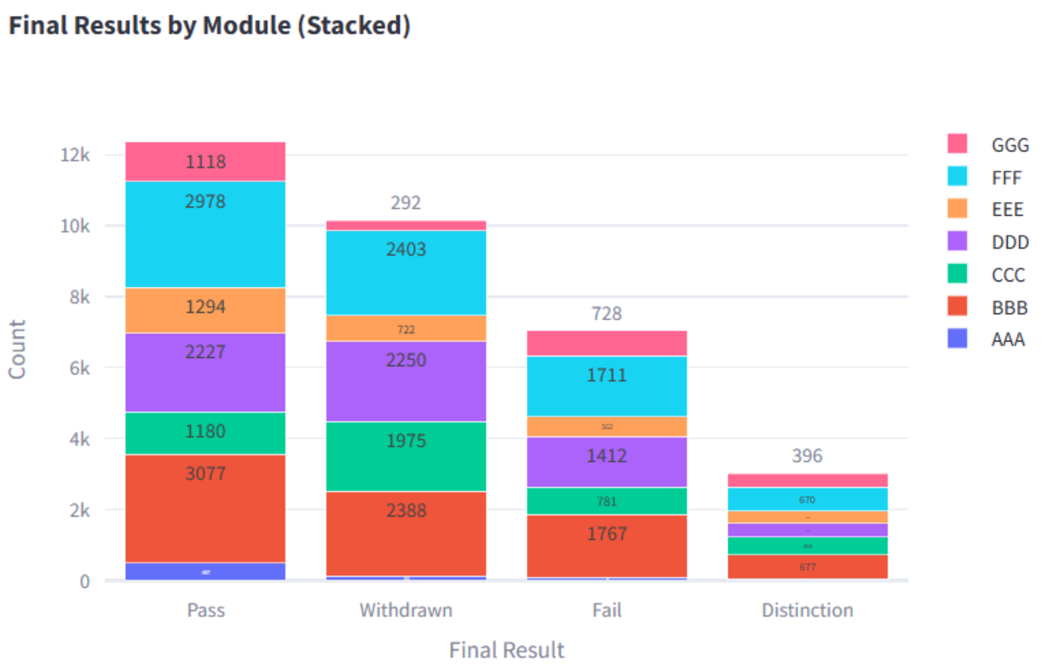


Final Results by Highest Education

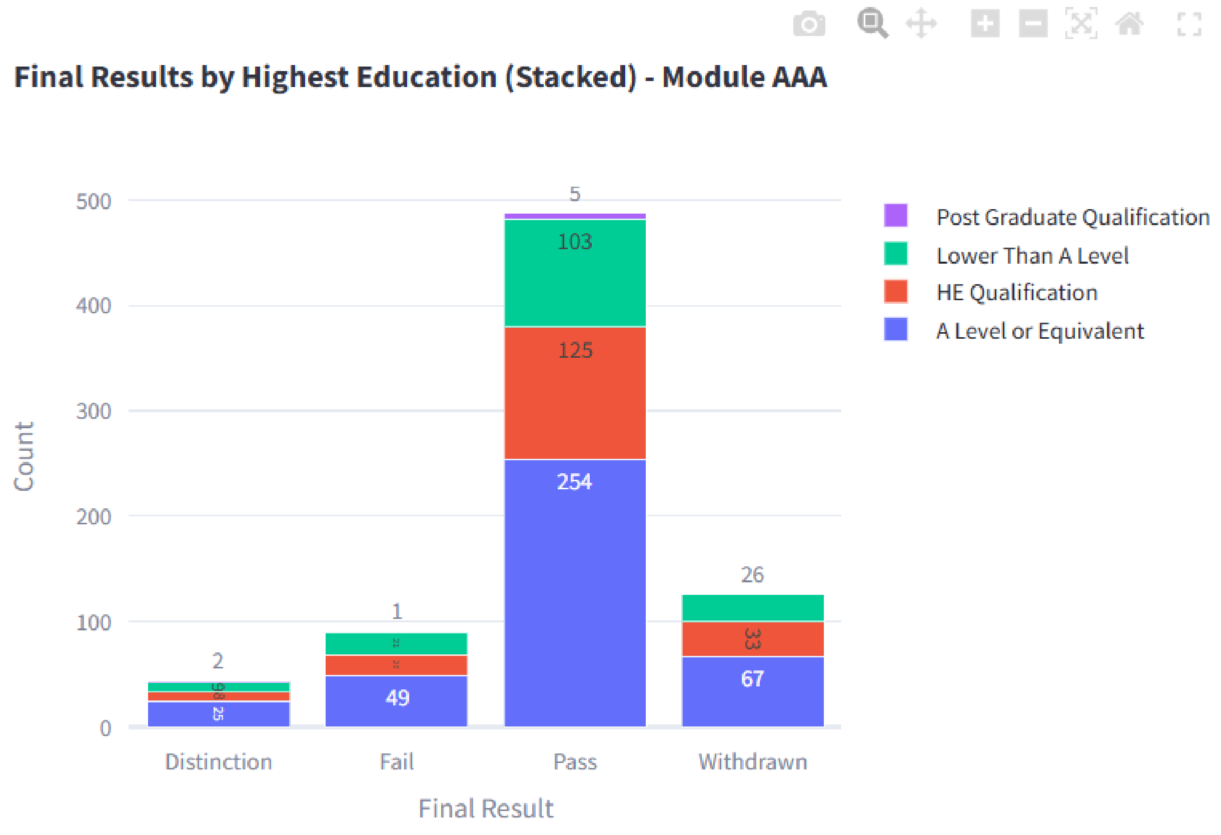
Final Results by Highest Education (Stacked)



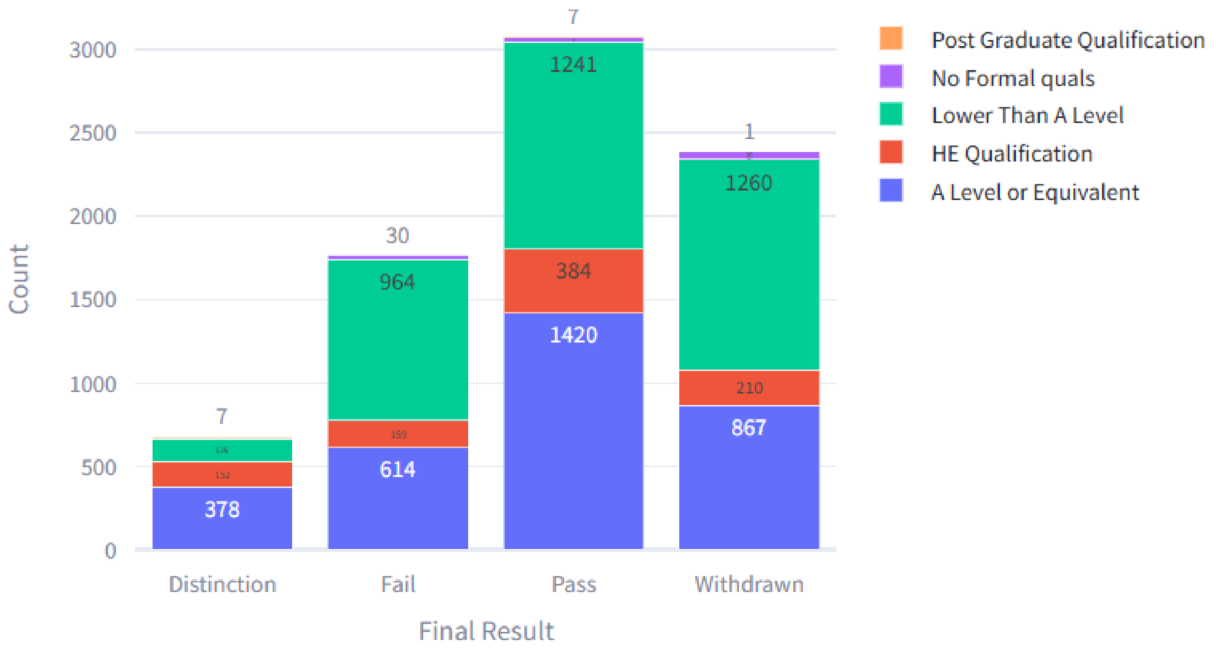
Final Results by Module



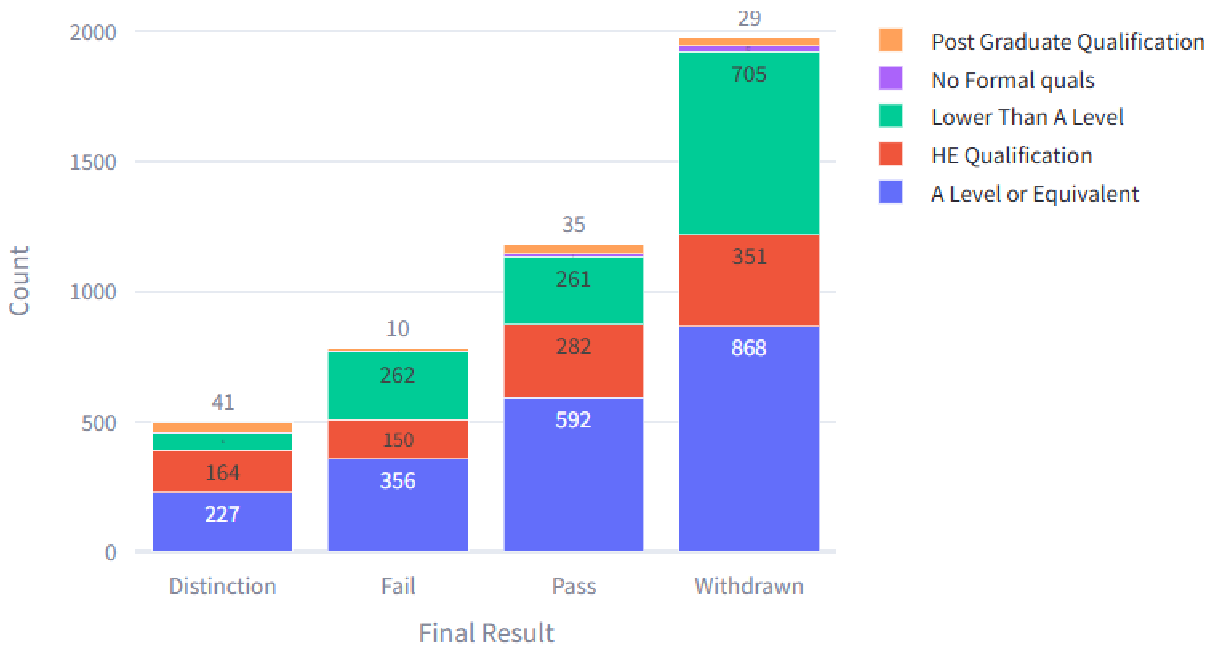
Final Results by Module Grouped by Module



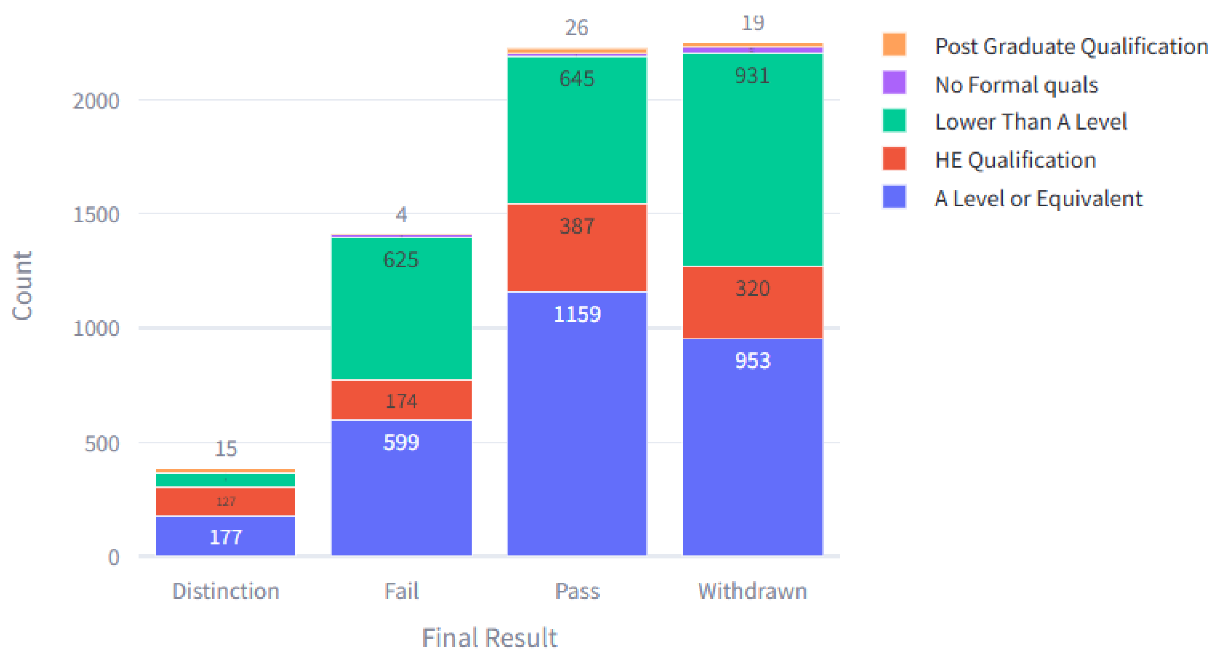
Final Results by Highest Education (Stacked) - Module BBB



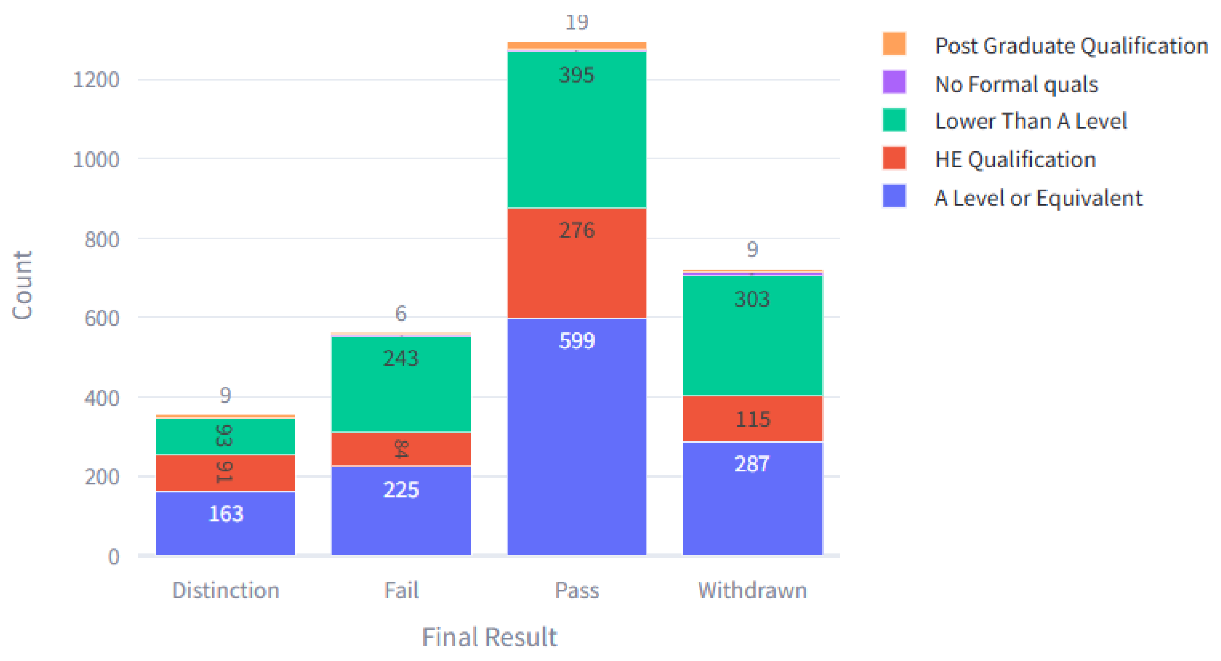
Final Results by Highest Education (Stacked) - Module CCC



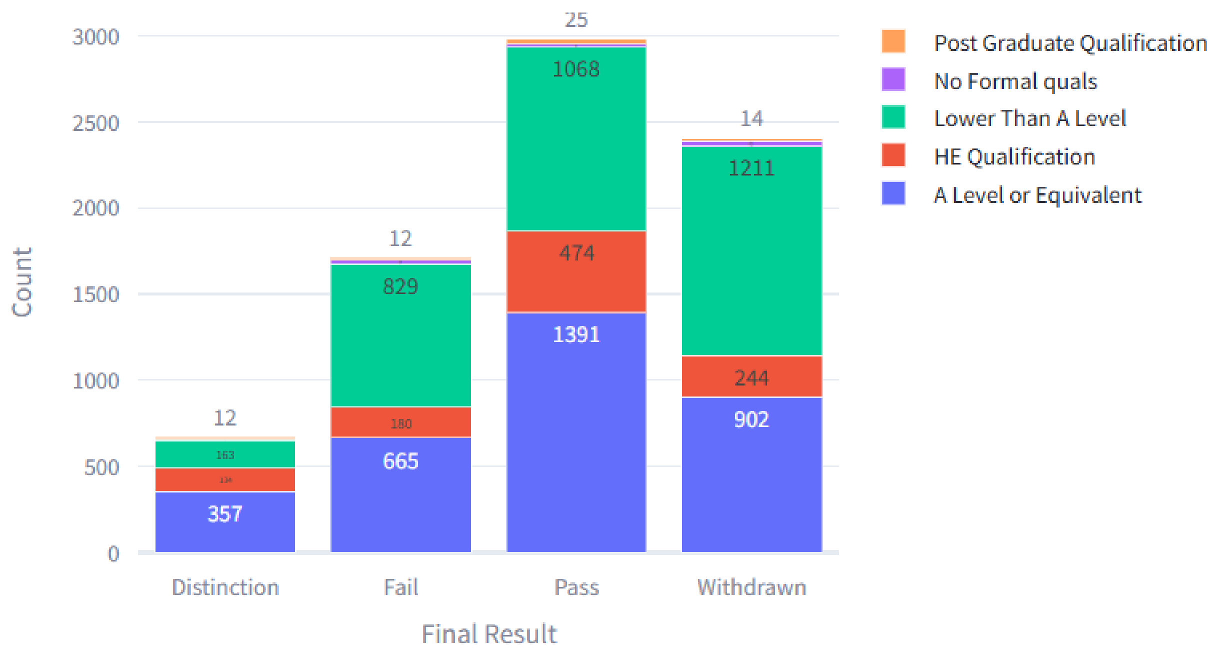
Final Results by Highest Education (Stacked) - Module DDD



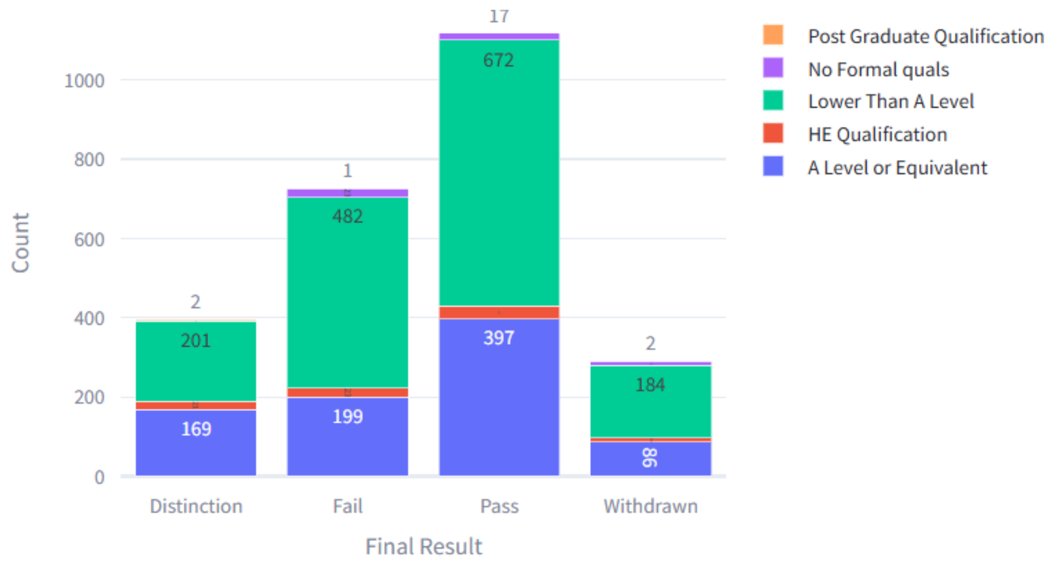
Final Results by Highest Education (Stacked) - Module EEE

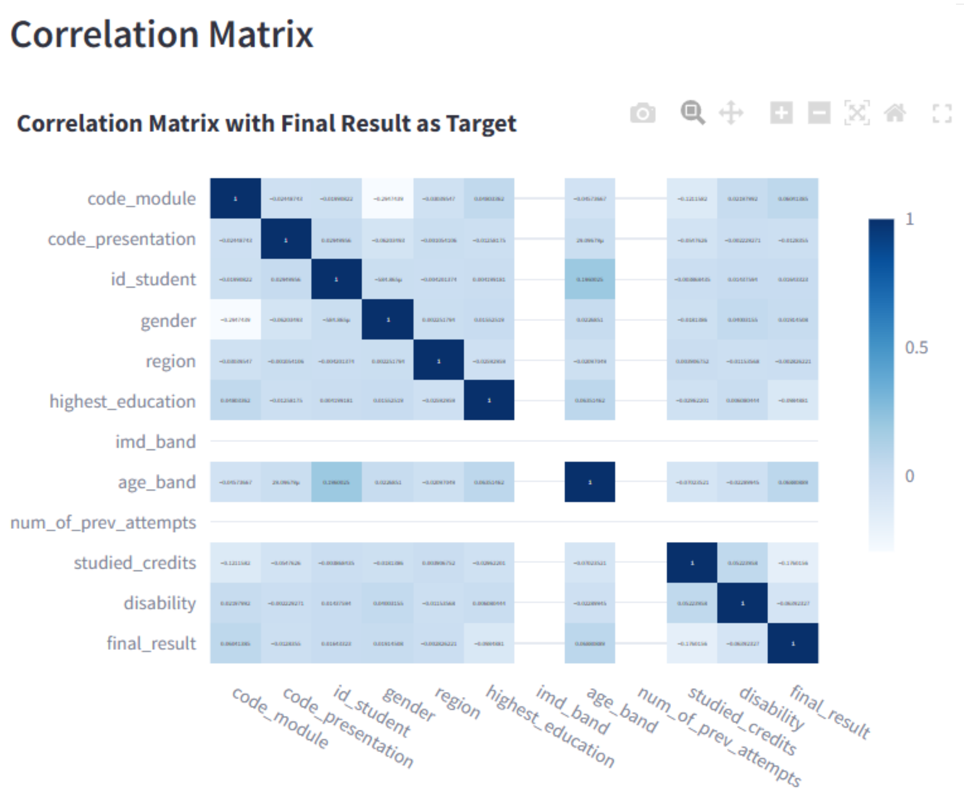


Final Results by Highest Education (Stacked) - Module FFF

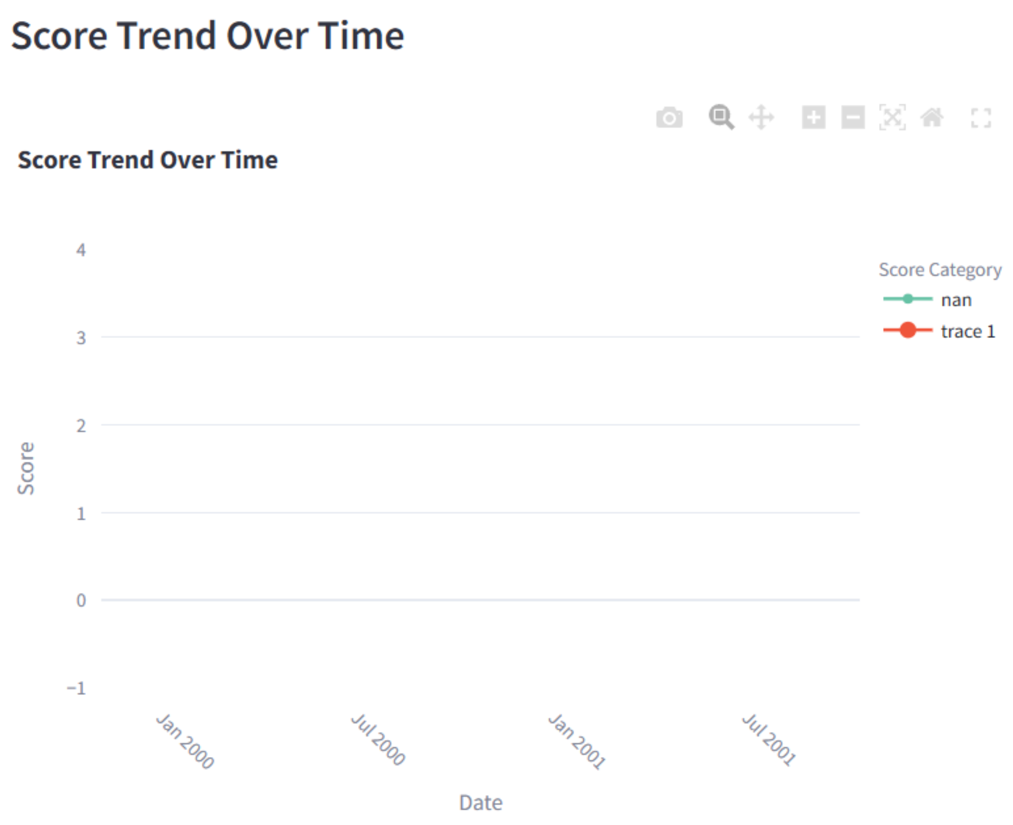


Final Results by Highest Education (Stacked) - Module GGG





4.4.5 Student Assessments



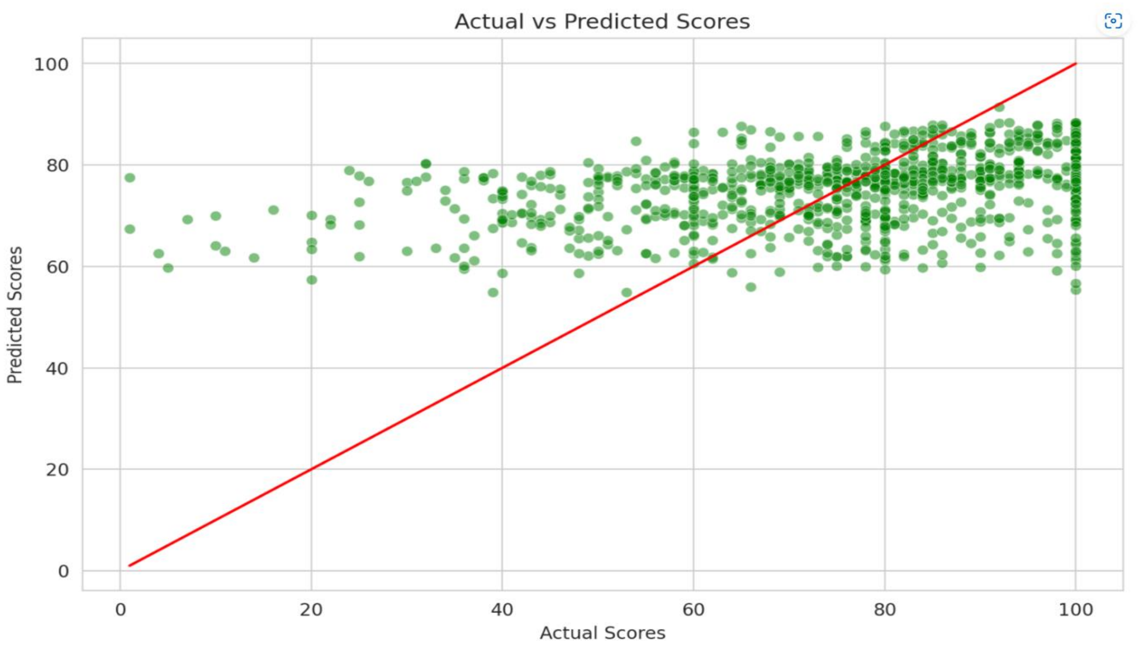
4.5 LINEAR REGRESSION

In this section the data was shaped into the following data:

The predicted scores and the actual scores

To analyze student scores using linear regression, we merge the *student assessment table* and the *student information table* based on student_id. This creates a single table containing both student details and their scores. The merged table is then split into two sets: 80% for training and 20% for testing, with the split being reproducible by setting the random state to 42. We use the score from the *student assessment table* as the target variable. Linear regression is applied to discover the relationship between the scores and the other features. The prediction was then made using the test data and compared with trained data. The results are as follows:

MSE 322.11024475868174
MAE 13.670166595453376
R2_score 0.119344493849133 81

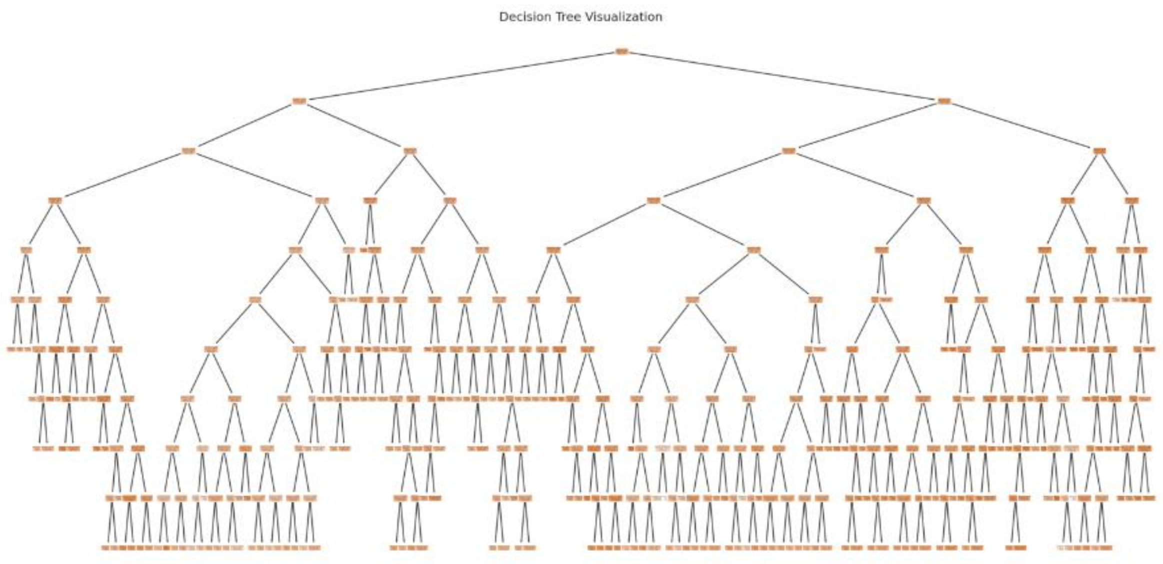


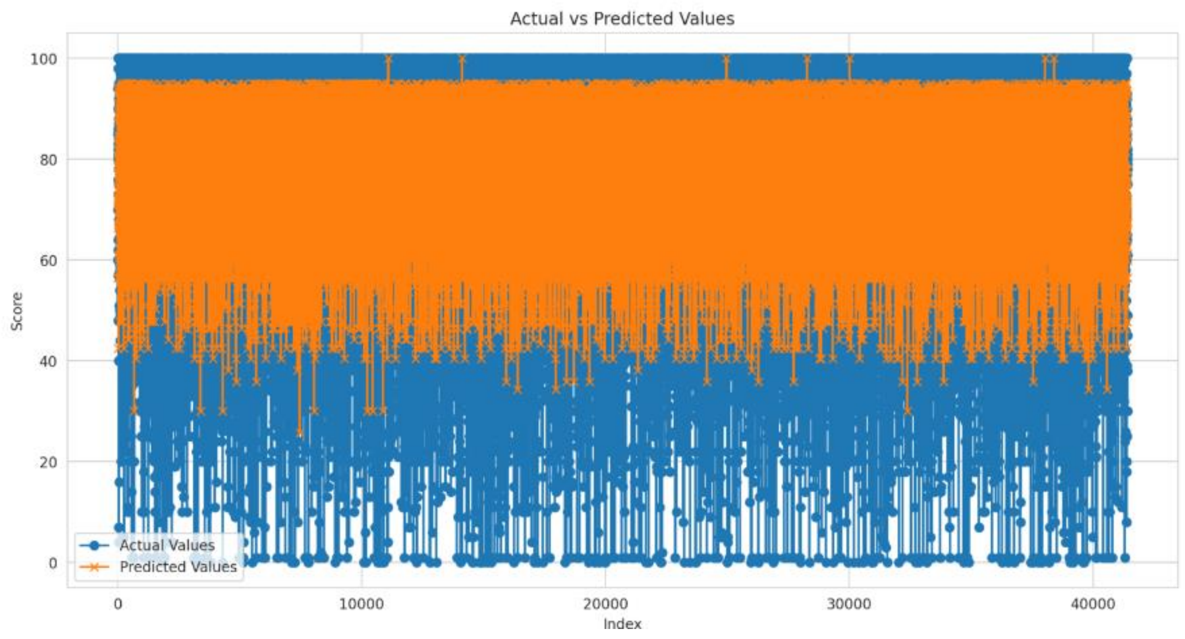
4.6 DECISION TREE REGRESSION

Similar to linear regression decision tree regression was followed the following steps:

- We merged the *student assessment table* and the *student information table* on student_id. This combined table contains both student details and their assessment scores.
- The combined table was split into a training set (80%) and a test set (20%) using a random state of 42 for reproducibility.
- The score from the *student assessment table* was used as the target variable for the decision tree model.
- A decision tree regressor was used to model the relationship between the student details (features) and their scores (target).
- The decision tree was trained using the training dataset.
- The trained decision tree model was used to predict scores on the test dataset.

Using a decision tree regressor provides a way to model complex relationships and interactions between student details and their scores, potentially capturing non-linear patterns that might not be detected by linear regression.





Pruning of tree was also performed but it does not enhance the result of the model.

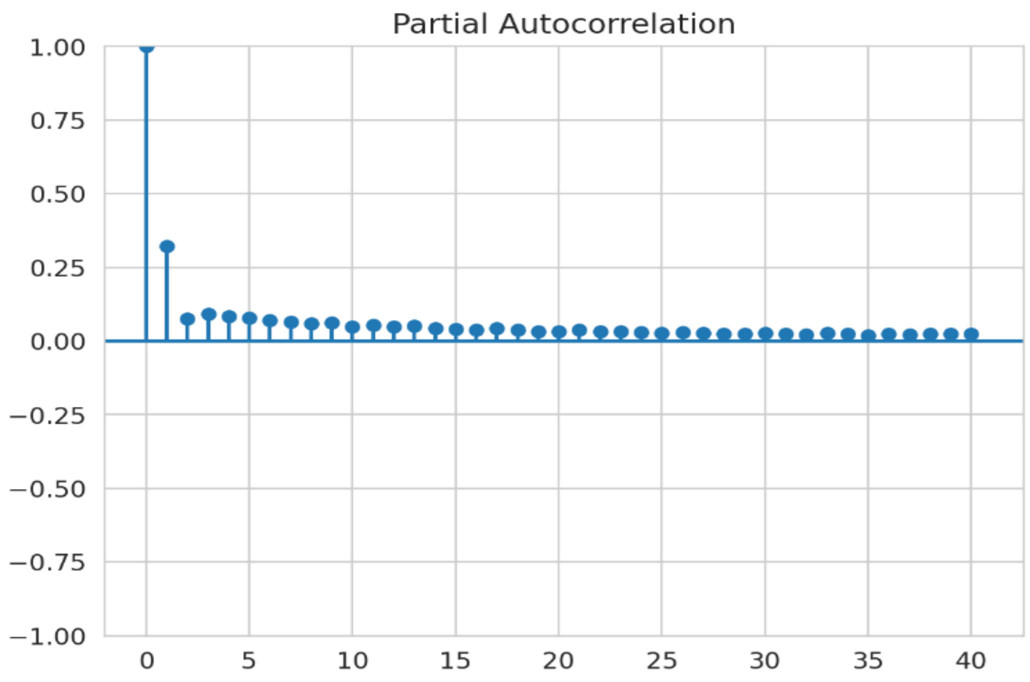
4.7 ARIMA

In the realm of time series analysis, the ARIMA (AutoRegressive Integrated Moving Average) model serves as a foundational tool for forecasting sequential data points. This model integrates three primary components—AutoRegressive (AR), Integration (I), and Moving Average (MA)—to effectively capture the temporal dependencies inherent in time series data such as student scores.

In Arima same data is not used but only subset of the columns from student information and assessment is used. The date and scores are taken and trend of score along with time is predicted. Following is the description of how it is done:

- **Model Fit**: The ARIMA model is fitted to historical student score data to identify and capture underlying patterns and trends.
- **Forecasting**: Once trained, the ARIMA model facilitates the prediction of future student scores based on the identified temporal dependencies.
- **Parameter Selection**: Determining the optimal values of ppp, ddd, and qqq typically involves iterative testing and validation procedures to optimize the model's predictive performance.

Columns:: score || d:: 0

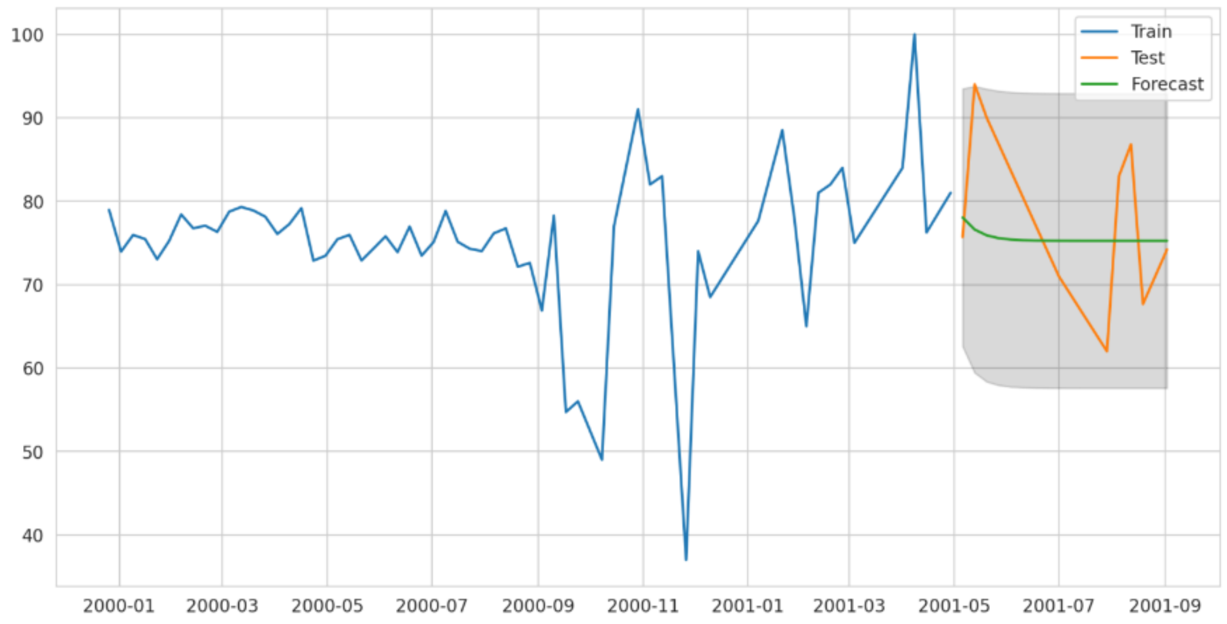


```
ADF Statistic: -5.348287

p-value: 0.000004

Since p-value < 0.05, stationarity doesnt exist (d = 0)
```

Forecasting

MAE: 7.649318753715154

RMSE: 8.940583140784193



4.8 RANDOM FOREST

Here the analysis is done on the trend of dropout of students rather than their score, the for this analysis is made with the combination of student assessment data, assessment data, student information data, course data, student course data, student registration data and VLE student data.

Data Preprocessing

1. **Handling Missing Values**:
   - Missing values in the dataset (df_student_full) are filled with appropriate strategies:
   - Numerical columns are filled with the mean value.
   - Categorical columns are filled with the most frequent value.

1. **Encoding Categorical Variables**:
   - Categorical columns (gender, region, highest_education, imd_band, age_band, num_of_prev_attempts, final_result) are converted into numerical labels using LabelEncoder.

2. **Imputing Missing Values**:
   - Separate imputation strategies (SimpleImputer) are applied for numerical and categorical features to ensure completeness in the dataset.

Feature Engineering

1. **Creating New Features**:
   - **Total Assessment Score**: Aggregate score from assessments (df_assessments_full) grouped by student ID.
   - **Interaction per Module**: Mean of sum_click interactions grouped by code_module and code_presentation.
   - **Dropout Label**: Created based on date_unregistration (1 if unregistered, otherwise 0).

Dropout Prediction Features

- **Features Used**:
  - gender, region, highest_education, imd_band, age_band, num_of_prev_attempts, studied_credits, sum_click, total_assessment_score

Machine Learning Model

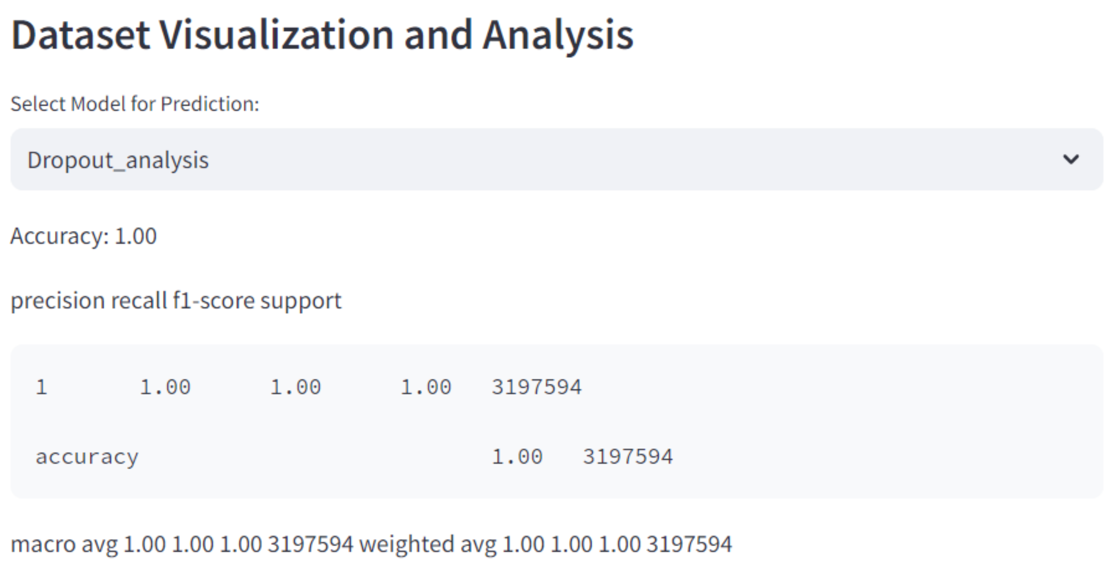
1. **Model Selection**:
   - RandomForestClassifier is chosen for predicting student dropout due to its ability to handle non-linear relationships and feature importance extraction capabilities.

2. **Training and Evaluation**:

    - The dataset is split into training and testing sets (train_test_split) with a ratio of 70:30 for model training and evaluation.
    - Features are standardized (StandardScaler) to ensure all features contribute equally to the model.
    - RandomForestClassifier (n_estimators=100) is trained on the standardized training data (X_train) and corresponding dropout labels (y_train).

3. **Performance Evaluation**:

    - **Accuracy**: The accuracy score of the model on the test set (X_test) is calculated to evaluate its overall predictive performance.
    - **Classification Report**: Provides precision, recall, F1-score, and support metrics for each class (dropout vs. non-dropout).
    - **Confusion Matrix**: Visual representation of the model's predictions compared to actual outcomes, showcasing true positives, true negatives, false positives, and false negatives.



Dataset Visualization and Analysis

Select Model for Prediction:

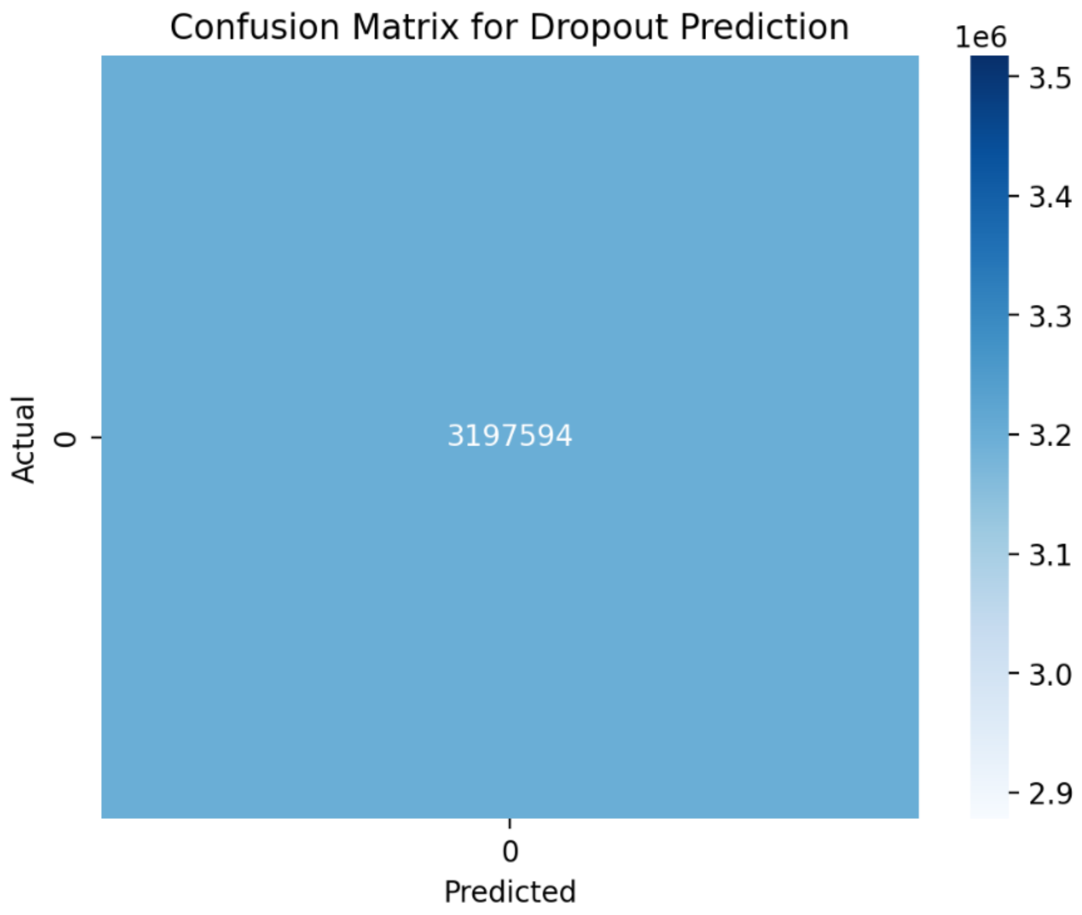
Dropout_analysis

Accuracy: 1.00

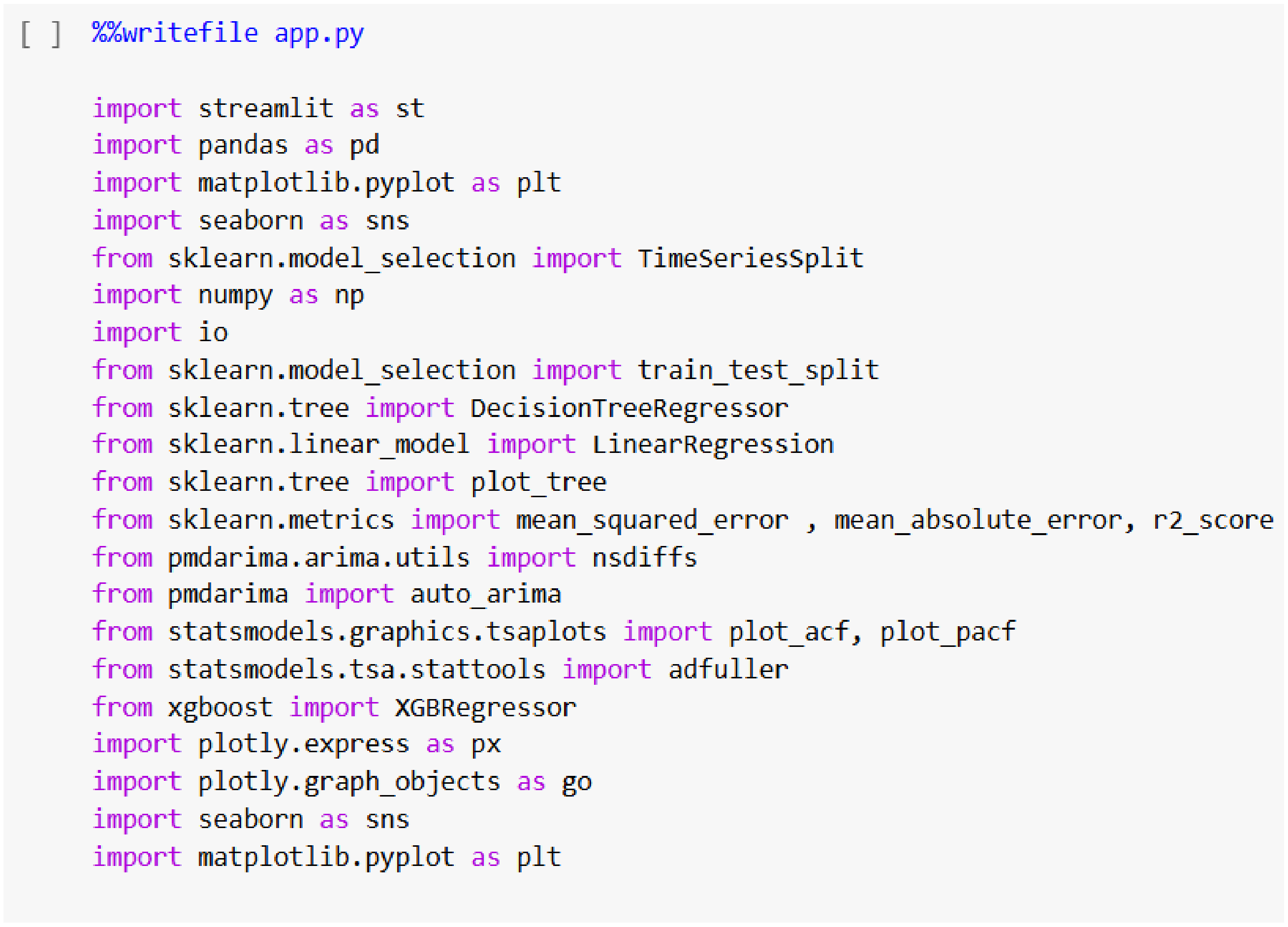
precision recall f1-score support

```
1        1.00       1.00       1.00    3197594

accuracy                               1.00    3197594
```

macro avg 1.00 1.00 1.00 3197594 weighted avg 1.00 1.00 1.00 3197594


Confusion Matrix for Dropout Prediction
1e6
3.5
3.4
3.3
3.2
3.1
3.0
2.9
Actual
0
3197594
0
Predicted

4.9 LIBRARIES



```
[ ]  %%writefile app.py

     import streamlit as st
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import TimeSeriesSplit
     import numpy as np
     import io
     from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.linear_model import LinearRegression
     from sklearn.tree import plot_tree
     from sklearn.metrics import mean_squared_error , mean_absolute_error, r2_score
     from pmdarima.arima.utils import nsdiffs
     from pmdarima import auto_arima
     from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
     from statsmodels.tsa.stattools import adfuller
     from xgboost import XGBRegressor
     import plotly.express as px
     import plotly.graph_objects as go
     import seaborn as sns
     import matplotlib.pyplot as plt
```

2. Conclusion

. 5.1 SUMMARY

In this report, we addressed the missing values in seven interrelated datasets and performed exploratory data analysis (EDA) and predictive modelling using various statistical and machine learning techniques. The datasets included Assessments, Courses, Student Assessments, Student Info, Student Registration, Student VLE, and VLE (Virtual Learning Environment). After handling the missing values with pandas techniques, we analyzed five tables with missing data and employed various models to derive insights and make predictions.

Handling Missing Values

The missing values in the datasets were handled using appropriate imputation techniques. We used the Simple Imputer from sklearn to replace missing values with mean, median, or mode as required by the context of the data. This ensured that the datasets were complete and ready for subsequent analysis.

Exploratory Data Analysis (EDA)

EDA was performed to understand the distribution, relationships, and patterns within the data. We utilized libraries such as pandas, matplotlib, seaborn, and plotly for visualization. Key insights from the EDA included:

- The distribution of scores in student assessments.
- The relationship between course registrations and student demographics.
- The interaction patterns of students with VLE resources.
- Trends and patterns in assessments and their impact on student performance.

Predictive Modelling

Linear Regression

Linear regression was applied to predict student scores based on various attributes. The performance metrics obtained were:

- Mean Squared Error (MSE): 322.11
- Mean Absolute Error (MAE): 13.67
- R2 Score: 0.119

These results indicate that the linear regression model had a limited ability to explain the variance in the student scores, suggesting the need for more complex models or additional features.

Decision Tree Regression

Decision tree regression provided better performance metrics compared to linear regression:

- MSE: 264.61
- MAE: 12.20
- R2 Score: 0.277

This model captured more complexity in the data and provided a better fit, as reflected by the improved R2 score.

ARIMA

ARIMA modelling was employed for time series forecasting. The stationarity test results were:

- ADF Statistic: -5.333599
- p-value: 0.000005

Since the p-value was less than 0.05, the data was deemed stationary (d=0). The forecasting results showed:

- MAE: 7.45
- RMSE: 8.76

These metrics indicated a reasonable accuracy for the time series predictions.

Random Forest Classifier

For classification tasks, the random forest classifier showed excellent performance:

- Accuracy: 1.00
- Precision: 1.00
- Recall: 1.00
- F1-score: 1.00

The analysis and modelling conducted in this report provided valuable insights into the data and demonstrated the effectiveness of various predictive techniques. While linear regression offered baseline performance, decision tree regression and random forest classifier showed significant improvements in predictive accuracy. The ARIMA model effectively handled time series forecasting, indicating potential applications for predicting future trends in student performance and interactions.

The successful handling of missing values and the comprehensive EDA laid a strong foundation for building accurate and reliable models. Future work could involve exploring additional features, employing more advanced modeling techniques, and conducting further validation to enhance the robustness of the predictions.

Overall, the findings from this report can inform decision-making in educational settings, aiding in the optimization of course design, assessment strategies, and resource allocation to improve student outcomes.

3. References

Books and Academic Journals

Pandas Documentation

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

- Available online at: https://pandas.pydata.org/docs/

Machine Learning and Data Science Techniques

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

- Available online at: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

Streamlit Framework

Streamlit Team. (2023). *Streamlit Documentation*. Streamlit Inc.

- Available online at: https://docs.streamlit.io/

Time Series Analysis

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts.

- Available online at: https://otexts.com/fpp3/

Decision Tree and Random Forest Algorithms

Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5-32.

- DOI: 10.1023/A:1010933404324

Online Articles and Tutorials

Handling Missing Data in Pandas

VanderPlas, J. (2016). *Python Data Science Handbook*. O'Reilly Media.

- Chapter available online at: https://jakevdp.github.io/PythonDataScienceHandbook/03.04-missing-values.html

Linear Regression

Ng, A. (2023). *CS229 Lecture Notes: Linear Regression*. Stanford University.

Available online at: https://cs229.stanford.edu/notes2023fall/cs229-notes1.pdf

ARIMA Model for Time Series Forecasting

Brownlee, J. (2017). *Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future*. Machine Learning Mastery.

- Available online at: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

Dataset Reference

Open University Learning Analytics Dataset

Kaliteevsky, D. et al. (2021). *The Open University Learning Analytics Dataset*. The Open University.

- Available online at: https://analyse.kmi.open.ac.uk/open_dataset