Excelerate

STUDENT ENGAGEMENT

POWERED DATA ANALYSIS

BY

TEAM 10A

COLLABORATION

TEAM MEMBERS

TEAM 10A

EDMAR BEATINGO
Team Leader

Industrial Engineer
Philippines

FARWA FAYYAZ
Project Manager

Doctor of Pharmacy
Pakistan

MARYAM MOHAMMED
Project Scribe

Digital Marketer
Kenya

VISHAAL DAYASHANKER
Project Lead

Computer Science & Engineering Student
India

INTERNSHIP OBJECTIVES

TEAM 10A

GENERAL

- Analyze Student Engagement Data
- Understanding which opportunities attract the most signups
- Identifying patterns that lead to drop-offs

Week-1

- Pre-processed Dataset
- Feature Engineering

Week-2

- Exploratory Data Analysis
- Data Visualization

Week-3

- Predictive Data Model
- Churn Analysis

Week-4

- Comprehensive Report
- Presentation Slide and Recording

RAW DATASET OVERVIEW

TEAM 10A

File Name

Copy of SLU Opportunity Wise Data-1710158595043(raw)

File Edit View Insert Format Data Tools Extensions Help

Database of every user who has ever created an account on Excelerate

The dataset contains 8,558 rows and 33 columns

Identifiers and Personal Info

Profile Id, First Name, Last Name, Date of Birth, Gender, Mobile Number, and address details (City, State, Country, Zip Code)

Opportunity Details

Opportunity Id, Opportunity Name, Opportunity Category, Opportunity Start Date, and Opportunity End Date

Educational Background

Institution Name, Graduation Date, Current Student Status, and Current/Intended Major

Status and Progress Tracking

Entry created at, Status Description, Status Code, Apply Date, Reward Awarded Date, Completion Date

Achievements

Reward Amount, Badge Id, Badge Name, Skill Points Earned, and Skills Earned.

WEEK-1 HIGHLIGHTS

TEAM 10A

DATA CLEANING PROCESS

- No.1 Standardizing Formats
- No.2 Handling Inconsistent Categorical Data
- No.3 Correcting Errors
- No.4 Handling Outliers
- No.5 Dealing with Duplicates
- No.6 Deleting Empty Columns

FEATURE ENGINEERING

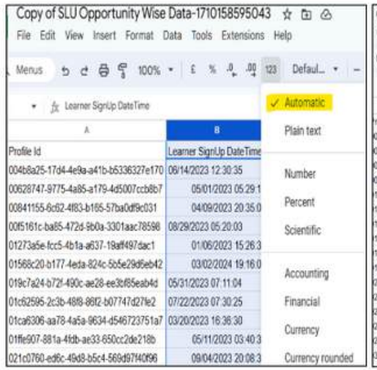
- No.1 Creating New Features
- No.2 Transforming Existing Features
- No.3 Extracting Useful Components
- No.4 Combining Features

DATA CLEANING

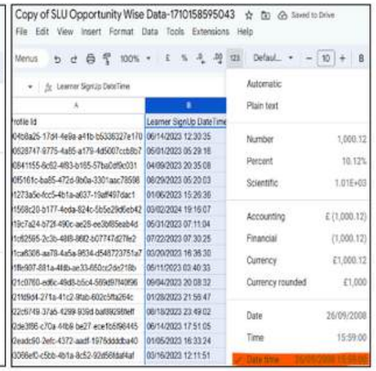
TEAM 10A

Standardizing Formats

Column B (Learner SignUp Date Time) data type formatting was changed to long date and time, then justified to the left.



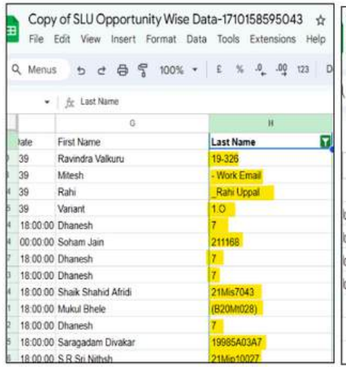
BEFORE



AFTER

Correcting Errors

Column H (Last Name) corrected erroneous entries of numeric names and replaced blank unverifiable cells with "Null."

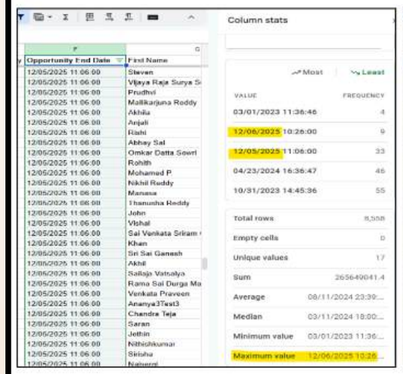


BEFORE

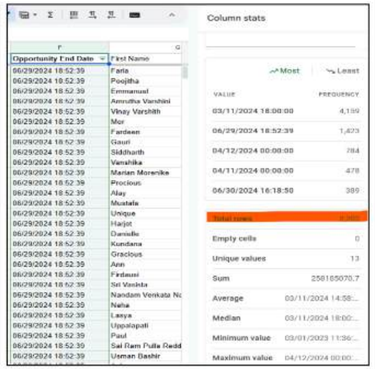
AFTER

Handling Outliers

Column F (Opportunity End Date) check column statistics and removed outliers in year 2025 consisting (355) row entries that are not possible for a year-long program duration.



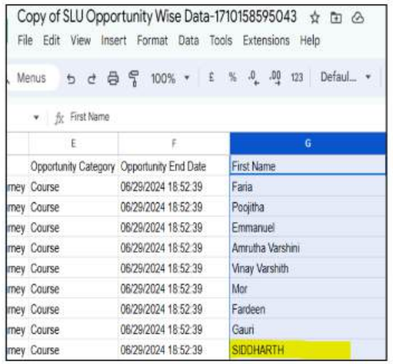
BEFORE



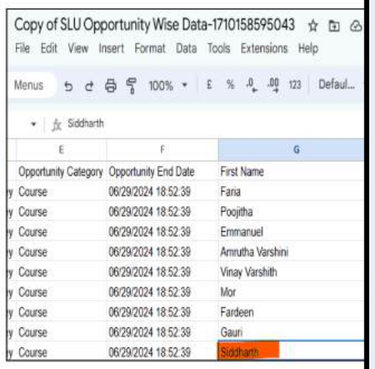
AFTER

Handling Inconsistent Categorical Data

Column G (First Name) changed formatting using proper text and trimmed spaces.



BEFORE



AFTER

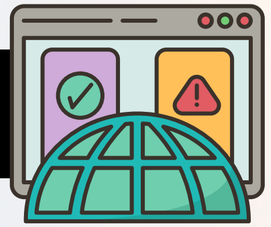
Dealing with Duplicates

No duplicate entries were found where the same Profile Id is registered for the same Opportunity Id in the dataset. This means each profile has only unique program registrations, based on these columns. No duplicate rows were found when considering all columns in the dataset, indicating that each entry is unique across all provided fields.

Deleting Empty Columns

Deleted **columns AH and AI** to reduce clutter, improve processing efficiency, avoid potential errors and enhances data quality. Empty columns consume memory without adding value.It becomes simpler to locate and analyze meaningful information.

DATA CLEANING



TEAM 10A

Standardizing Formats

- Column B (Learner SignUp Date Time)
- Column F (Opportunity End Date)
- Column V (Entry created at)
- Column Y (Apply Date)
- Column Z (Opportunity Start Date)
- Column I (Date of Birth)
- Column S (Graduation Date)
- Column K (Mobile Number)
- Column X (Status Code)

Correcting Errors

- Column G (First Name)
- Column H (Last Name)
- Column L (Address Line 1)
- Column M (Address Line 2)
- Column N (City)
- Column O (State)
- Column P (Country)
- Column Q (Zip Code)
- Column R (Institution Name)
- Column U (Current/Intended Major)
- Column Z (Opportunity Start Date)
- Column AA (Completion Date)
- Column AC (Reward Amount)
- Column AD (Badge ID)
- Column AE (Badge Name)
- Column AC (Reward Amount)
- Column AG (Skills Earned)

Handling Outliers

- Column F (Opportunity End Date)
- Column I (Date of Birth)

Handling Inconsistent Categorical Data

- Column G (First Name)
- Column H (Last Name)

Deleting Empty Columns

Deleted **columns AH and AI** to reduce clutter, improve processing efficiency, avoid potential errors and enhances data quality. Empty columns consume memory without adding value.It becomes simpler to locate and analyze meaningful information.

Dealing with Duplicates

No duplicate entries were found where the same Profile Id is registered for the same Opportunity Id in the dataset. This means each profile has only unique program registrations, based on these columns. No duplicate rows were found when considering all columns in the dataset, indicating that each entry is unique across all provided fields.

FEATURE ENGINEERING

CREATING NEW FEATURES

Age Calculation

Copy of SLU Opportunity Wise Data-1710158595043

File Edit View Insert Format Data Tools Extensions

Menus 100%

15 Ifedayo Street Off Ekoro Road Abule Egba

J	K	L
Last Name	Date of Birth	Age Calculation
Khan	01/12/2001	23
Bembadi	08/16/2000	24
Thompson	01/27/2002	22
Anagurthi	11/01/1999	24
Pechetti	04/19/2000	24
Samb	05/12/1996	28
Shroff	09/09/2001	23
Menon	02/27/2006	18
Rajawat	12/22/2005	18
Varshney	06/26/1998	26
Abubakar	10/01/1998	26
Christopher	06/30/1997	27
Patel	09/09/1998	26
Nafee	05/06/1997	27

Feature:
Age Calculation
(added Column L)

Function:
Calculates the number of full years between a date in cell K2 and today's date. The date in K2 is stored as text in the format MM/DD/YYYY, and the formula extracts the year, month, and day to create a valid date.

Syntax:
=IFERROR(DATEDIF(DATE(VALUE(MID(K2, 7, 4)), VALUE(LEFT(K2, 2)), VALUE(MID(K2, 4, 2))), TODAY(), "Y"), "Invalid Date")

Opportunity Duration

Copy of SLU Opportunity Wise Data-1710158595043

File Edit View Insert Format Data Tools Extensions Help

Menus 100% Defaul...

Null

AB	AC	AD	AE
Status Code	Apply Date	Opportunity Start Date	Opportunity Duration (Days)
1080	06/14/2023 12:36:09	11/03/2022 18:30:39	604
1080	05/01/2023 06:08:21	11/03/2022 18:30:39	604
1080	05/11/2023 00:21:29	11/03/2022 18:30:39	604
1070	10/09/2023 22:02:42	11/03/2022 18:30:39	604
1080	01/06/2023 15:40:10	11/03/2022 18:30:39	604
1040	03/02/2024 20:30:35	Null	0
1110	06/14/2023 05:29:41	11/03/2022 18:30:39	604
1070	07/22/2023 07:44:24	11/03/2022 18:30:39	604
1080	05/24/2023 13:44:09	11/03/2022 18:30:39	604
1070	12/17/2023 02:58:44	11/03/2022 18:30:39	604

Feature:
Opportunity Duration
(added Column AE)

Function:
Calculates the difference between two dates stored as text in H2 and AD2 (in the format MM/DD/YYYY) and returns the difference in days. If there is an error (e.g., if either H2 or AD2 has an invalid date format), it returns 0.

Syntax:
=IFERROR((DATEVALUE(DATE(VALUE(MID(H2,7,4)),VALUE(LEFT(H2,2)),VALUE(MID(H2,4,2)))))-(DATEVALUE(DATE(VALUE(MID(AD2,7,4)),VALUE(LEFT(AD2,2)),VALUE(MID(AD2,4,2))))), VALUE(0))

FEATURE ENGINEERING

TRANSFORMING EXISTING FEATURES

TEAM 10A

Normalized Reward Amount

Copy of SLU Opportunity Wise Data-1710158595043

AG	AH	AI
Completion Date	Reward Amount	Normalized Reward Amount
Null	0.00	0
Null	0.00	0
Null	0.00	0
Null	0.00	0
Null	0.00	0
Null	0.00	0
Null	0.00	0
Null	0.00	0
03/05/2024	200.00	1
Null	0.00	0

Feature:
Normalized Reward Amount
(added Column AI)

Function:
Normalizes the value in column AH to a scale between 0 and 1, based on the minimum and maximum values in the range AH to bring all values within the same range, making them easier to compare.

Syntax:
=(AH2-MIN(AH$2:AH$8203))/(MAX(AH$2:AH$8203)-MIN(AH$2:AH$8203))

Normalized Skills Earned

Copy of SLU Opportunity Wise Data-171015859504

AL	AM
Skill Points Earn	Normalized Skill Points Earn
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
0.00	0
1,052.00	1

Feature:
Normalized Skills Earned
(added Column AM)

Function:
Normalizes the value in column AM to a scale between 0 and 1, based on the minimum and maximum values in the range AM to bring all values within the same range, making them easier to compare.

Syntax:
=(AL2-MIN(AL$2:AL$8203))/(MAX(AL$2:AL$8203)-MIN(AL$2:AL$8203))

Gender Data Encoding

Copy of SLU Opportunity Wise Data-1710158595043

M	N	O
Gender	Gender Data Encoding	Mobile Number
Female	1	+13147230321'
Female	1	+21655415108'
Female	1	+2347065480177'
Male	2	+13148185811'
Male	2	+2348079220552'
Female	1	+919167257003'
Female	1	+16365296996'
Female	1	+13144500235'
Female	1	+917097995522'
Female	1	+18317107216'
Male	2	+918929172802'
Male	2	+13143030487'
Male	2	+13147285704'
Others	3	+918688612411'

Feature:
Gender Data Encoding
(added Column N)

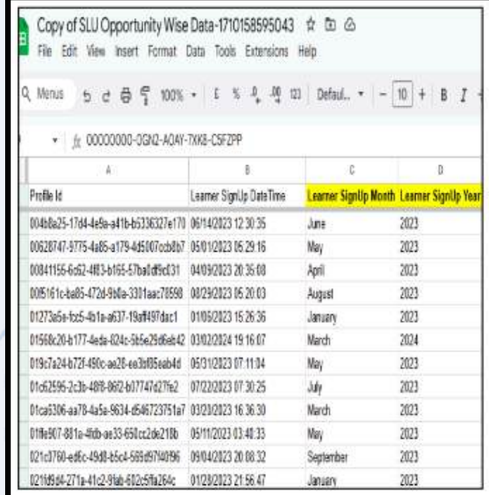
Function:
Uses the IFS function to assign a numerical value based on the specified gender to convert categorical gender data into numeric codes for analysis.

Syntax:
=IFS(M2="Female",value(1),M2="Male",value(2),M2="Others",value(3))

FEATURE ENGINEERING

EXTRACTING USEFUL COMPONENTS

Learner SignUp Month and Year



Copy of SLU Opportunity Wise Data-1710158595043

File Edit View Insert Format Data Tools Extensions Help

00000000-0GN2-AQ4Y-7XK8-C5FZPP

	A	B	C	D
	Profile Id	Learner SignUp DateTime	Learner SignUp Month	Learner SignUp Year
	004b8a25-17d4-4e9a-a41b-b5336327e170	06/14/2023 12:30:35	June	2023
	00628747-9775-4a85-a179-4d5007ccb0b7	05/01/2023 05:29:16	May	2023
	00841155-6c52-4ff3-b165-57ba1df9cf31	04/09/2023 20:35:08	April	2023
	0095161c-ba85-472d-9b0a-3301aac78598	08/29/2023 05:20:03	August	2023
	01273a5a-fcc5-4b1a-a637-19aff497dac1	01/05/2023 15:26:36	January	2023
	01568c20-b177-4eda-024c-5b5e29d6eab42	03/02/2024 19:16:07	March	2024
	019c7a24-b72f-490c-ae28-ee3bf85eab4d	05/31/2023 07:11:34	May	2023
	01c62595-2c3b-48f8-0662-b07747d27fe2	07/22/2023 07:30:25	July	2023
	01ca6306-aa78-4a5a-9634-d546723751a7	03/23/2023 16:36:30	March	2023
	01fle507-881a-4fdb-ae33-650cc2de218b	05/11/2023 03:48:33	May	2023
	021c3760-ed6c-49d8-b5c4-5685d97f4896	09/04/2023 20:08:32	September	2023
	021fd964-271a-41c2-91ab-602c5ffa264c	01/28/2023 21:56:47	January	2023

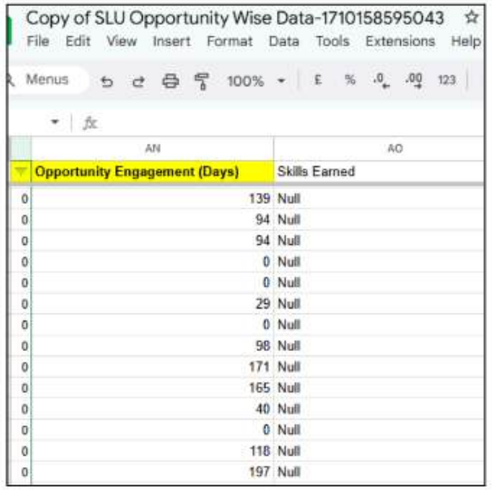
Feature:
Learner SignUp Month (added Column C)
Learner SignUp Year (added Column D)

Function:
Extracts the month name from a date stored as text in cell B2 (formatted as MM/DD/YYYY) and returns the full month name (e.g., "June") based on the date in B2.

Extracts the year as a numeric value from a date in B2. This is useful when B2 contains a date as text in the format MM/DD/YYYY and returns 2024 as a numeric value, making it useful for calculations involving the year extracted from B2.

Syntax:
=TEXT((DATEVALUE(DATE(VALUE(MID(B2,7,4)),VALUE(LEFT(B2,2)),VALUE(MID(B2,4,2))))), "MMMM")
=VALUE(MID(B2,7,4))

Opportunity Engagement (Days)



Copy of SLU Opportunity Wise Data-1710158595043

File Edit View Insert Format Data Tools Extensions Help

	AN	AO
	Opportunity Engagement (Days)	Skills Earned
0	139	Null
0	94	Null
0	94	Null
0	0	Null
0	0	Null
0	29	Null
0	0	Null
0	98	Null
0	171	Null
0	165	Null
0	40	Null
0	0	Null
0	118	Null
0	197	Null

Feature:
Opportunity Engagement (Days)
(added Column AN)

Function:
Calculates the difference in days between two dates stored as text in cells AD2 and AC2 (in the format MM/DD/YYYY). If either of the dates is invalid, the formula returns 0 instead of an error.

Syntax:
=IFERROR((DATEVALUE(DATE(VALUE(MID(AD2,7,4)),VALUE(LEFT(AD2,2)),VALUE(MID(AD2,4,2)))))-(DATEVALUE(DATE(VALUE(MID(AC2,7,4)),VALUE(LEFT(AC2,2)),VALUE(MID(AC2,4,2))))), VALUE(0))

FEATURE ENGINEERING

COMBINING FEATURES

Age and Duration Interaction

Copy of SLU Opportunity Wise Data-1710158595043

AO	AP
Skills Earned	Age and Duration Interaction
Null	13892
Null	14496
Null	13288
Null	14496
Null	14496
Null	0
Null	13892
Null	10872
Null	10872

Feature:
Age and Duration Interaction
(added Column AP)

Function:
Create a feature that combines Opportunity Duration with Age to provide deeper insights into how different factors interact.

Syntax:
=L2*AE2

Engagement Scores

Copy of SLU Opportunity Wise Data-1710158595043

AQ	AR	AS	AT
Duration Weight (50%)	Age Weight (30%)	Gender Weight (20%)	Engagement Scores
302	6.9	0.2	309.10
302	7.2	0.2	309.40
302	6.6	0.4	309.00
302	7.2	0.2	309.40
302	7.2	0.4	309.60
0	8.4	0.4	8.80
302	6.9	0.4	309.30
302	5.4	0.2	307.60
302	5.4	0.4	307.80
302	7.8	0.2	310.00
302	7.8	0.2	310.00
302	8.1	0.2	310.30
302	7.8	0.4	310.20
302	8.1	0.4	310.50
302	7.8	0.4	310.20

Feature:
Duration Weight (50%) (added Column AQ)
Age Weight (30%) (added Column AR)
Gender Weight (20%) (added Column AS)
Engagement Score (100%)(added Column AT)

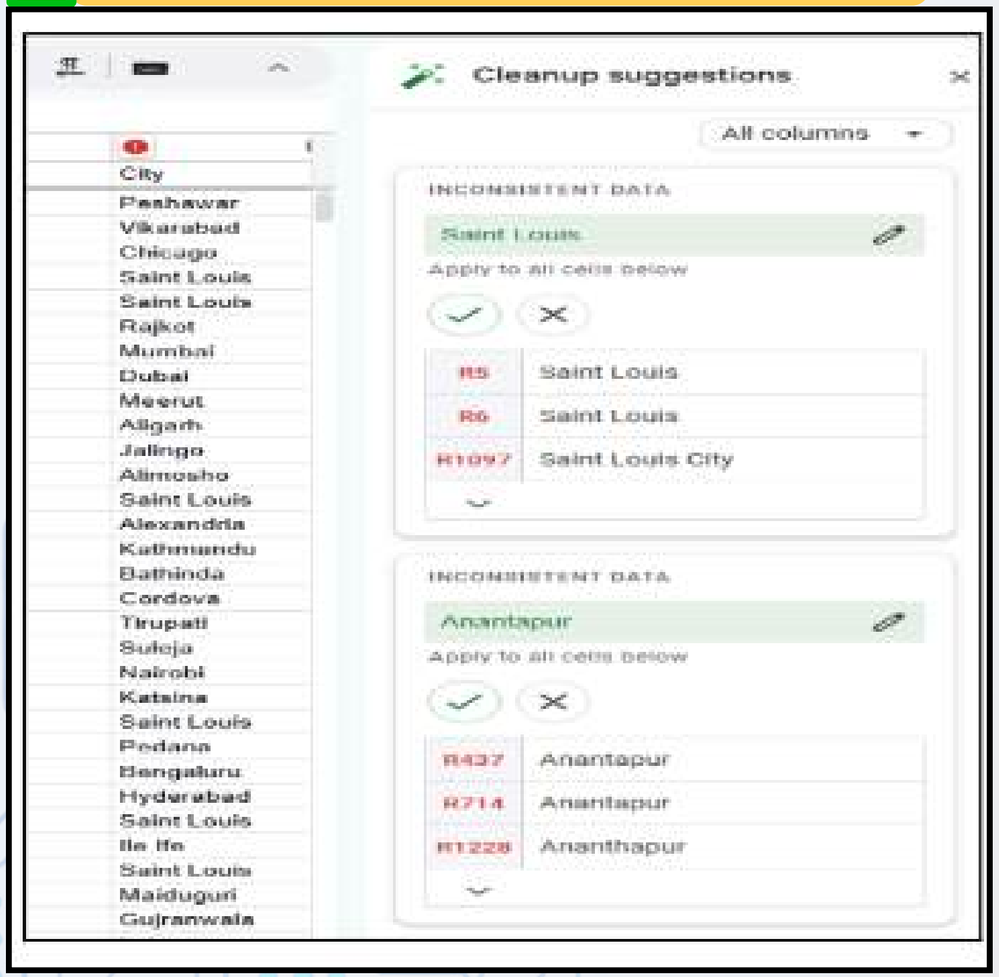
Function:
Calculate a weighted average for combining features like Opportunity Duration, Age, and Gender variables. Weights are assigned to each feature based on its importance, then the weighted sum is computed.

Syntax:
=AE2*0.5, =L2*0.3, =N2*0.2, =sum(AQ2:AS2)/sum(0.5,0.3,0.2)

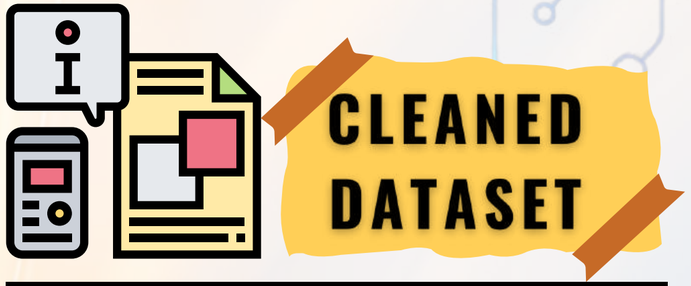
DATA VALIDATION

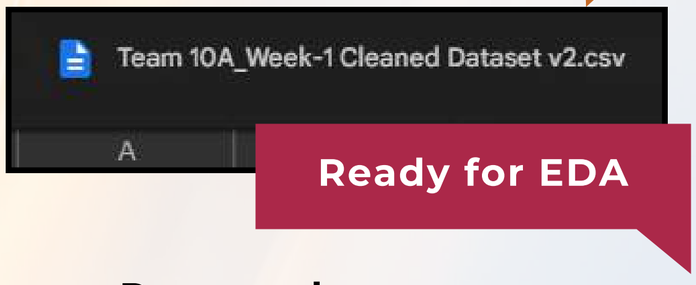
TEAM 10A

Google Sheet Clean-up Suggestion



Google Sheet Column Stat





Ready for EDA

Renamed as:
learner_engagement.csv

WEEK-2 HIGHLIGHTS

TEAM 10A

EXPLORATORY DATA ANALYSIS

- No.1 Further Data Cleaning
- No.2 Summary Statistics
- No.3 Correlation Matrix
- No.4 Result Interpretation

DATA VISUALIZATIONS

- No.1 Sign-up Trends
- No.2 Completion Trends
- No.3 Patterns and Correlations
- No.4 Outliers and Anomalies

PRE-PROCESSED DATASET OVERVIEW

TEAM 10A

File Name

learner_engagement.csv

A B C

Renamed file from Week-1 cleaned raw dataset

The dataset contains 8,203 rows and 46 columns

FILE ORIGIN:
Derived from Excelerate online education platform that tracks user activities, such as signups, course completions, and engagement scores.

KEY COLUMNS

- Learner ID
- SignUp Date
- Opportunity End Date
- Engagement Scores
- Completion Status

KEY FEATURES

- Signup Data
- Completion Information
- Completion Time
- Engagement Scores
- Segmenting Fields

FURTHER DATA CLEANING

Python Codes

```python
import pandas as pd

file_path = 'learner_engagement.csv'
learner_engagement_df = pd.read_csv(file_path)

# Remove duplicates
learner_engagement_df.drop_duplicates(inplace=True)

# Handle missing values
# Fill missing categorical values with 'Unknown' and numeric with median or zero
for column in learner_engagement_df.columns:
    if learner_engagement_df[column].dtype == 'object':
        learner_engagement_df[column].fillna('Unknown', inplace=True)
    else:
        learner_engagement_df[column].fillna(learner_engagement_df[column].median(), inplace=True)

# Correct data types
# Convert date columns to datetime format
learner_engagement_df['Learner SignUp DateTime'] = pd.to_datetime(learner_engagement_df['Learner SignUp DateTime'],
errors='coerce')
learner_engagement_df['Opportunity End Date'] = pd.to_datetime(learner_engagement_df['Opportunity End Date'], errors='coerce')

# Drop unnecessary columns
columns_to_drop = ['Profile Id', 'First Name', 'Last Name']
learner_engagement_df.drop(columns=columns_to_drop, inplace=True, errors='ignore')

# Save the cleaned dataset
cleaned_file_path = 'cleaned_learner_engagement.csv'
learner_engagement_df.to_csv(cleaned_file_path, index=False)
```

The new dataset contains 8,203 rows and 44 columns

cleaned_learner_engagement.csv

Key Points

- Remove duplicates
- Handle missing values
- Correct data types
- Drop unnecessary columns
- Save the cleaned dataset

SUMMARY STATISTICS

Python Codes

```python
# Calculate Descriptive Statistics (Mean, Median, Mode)
completion_time_mean = cleaned_df['Completion Time (Days)'].mean()
completion_time_median = cleaned_df['Completion Time (Days)'].median()
completion_time_mode = cleaned_df['Completion Time (Days)'].mode()[0]

print(f"Completion Time Mean: {completion_time_mean:.2f} days")
print(f"Completion Time Median: {completion_time_median} days")
print(f"Completion Time Mode: {completion_time_mode} days")
```

```
Total Signups: 8202
Completed Opportunities: 27
Completion Rate: 0.33%
Completion Time Mean: 228.22 days
Completion Time Median: 328.0 days
Completion Time Mode: 17.0 days
```

KEY POINTS

- No.1 Low Completion Rate
- No.2 Dispersed Completion Times
- No.3 Long Average Completion Time

CORRELATION MATRIX AND RESULT

TEAM 10A

Python Codes

```python
# Extract signup year and month for correlation analysis
cleaned_df['SignUp Year'] = cleaned_df['Learner SignUp DateTime'].dt.year
cleaned_df['SignUp Month'] = cleaned_df['Learner SignUp DateTime'].dt.month

# Correlation between SignUp Year, Month, and Completion Time
correlations = cleaned_df[['SignUp Year', 'SignUp Month', 'Completion Time (Days)']].corr()
print("Correlations:\n", correlations)
```

```
Correlations:
                        SignUp Year  SignUp Month  Completion Time (Days)
SignUp Year                1.000000     -0.775176                0.307476
SignUp Month              -0.775176      1.000000               -0.081290
Completion Time (Days)     0.307476     -0.081290                1.000000
```

A correlation matrix shows the correlation coefficients between different variables. The correlation coefficient ranges from -1 to 1:

1: Perfect positive correlation (as one variable increases, the other increases proportionally)
-1: Perfect negative correlation (as one variable increases, the other decreases proportionally)
0: No correlation (no linear relationship between the variables)

INTERPRETATION

No.1 SignUp Year and SignUp Month

strong negative correlation

No.2 SignUp Year and Completion Time

moderate positive correlation

No.3 SignUp Month and Completion Time

weak negative correlation

DATA VISUALIZATIONS

Sign-up Trends

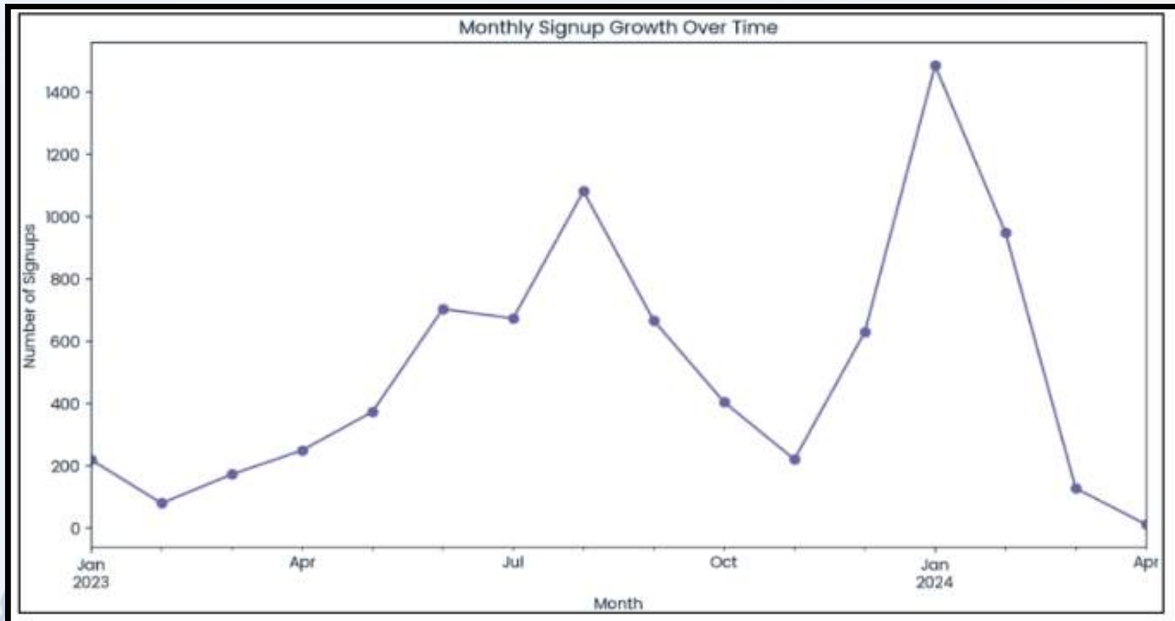


Image 3.1.1 Shows the line graph of monthly signup growth over time per month and year.

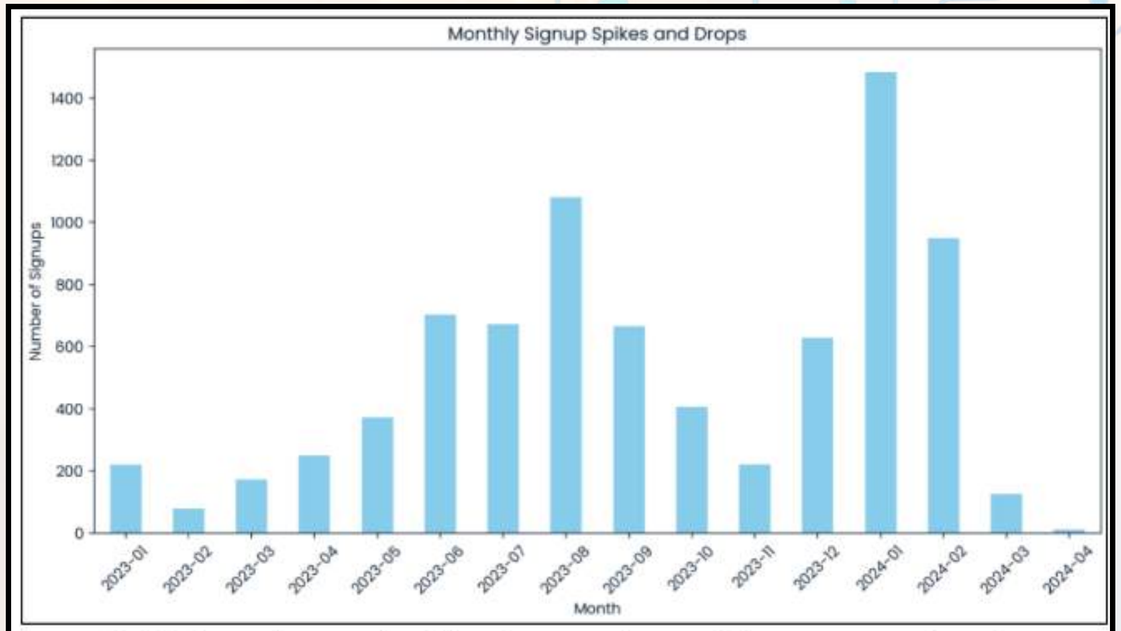


Image 3.1.3 Shows bar graph of the signup spikes and drops per month and year.

Python Codes

```python
# 2. Signup Trends Analysis
# 2.1 Growth (Signup Growth Over Time)
signups_by_month = cleaned_df['Learner SignUp DateTime'].dt.to_period("M").value_counts().sort_index()

plt.figure(figsize=(12, 6))
signups_by_month.plot(kind='line', marker='o', title='Monthly Signup Growth Over Time')
plt.xlabel('Month')
plt.ylabel('Number of Signups')
plt.show()
```

Python Codes

```python
# 2.3 Spikes and Drops in Signup Trends
plt.figure(figsize=(12, 6))
signups_by_month.plot(kind='bar', color='skyblue')
plt.title('Monthly Signup Spikes and Drops')
plt.xlabel('Month')
plt.ylabel('Number of Signups')
plt.xticks(rotation=45)
plt.show()
```

DATA VISUALIZATIONS

Sign-up Trends

TEAM 10A

Python Codes

```python
# 2.2 Seasonality (Signup Peaks Heatmap)
# Extract year and month for heatmap plotting
cleaned_df['Year'] = cleaned_df['Learner SignUp DateTime'].dt.year
cleaned_df['Month'] = cleaned_df['Learner SignUp DateTime'].dt.month

# Heatmap for signups by month and year
heatmap_data = cleaned_df.groupby(['Year', 'Month']).size().unstack()

# Month names for x-axis
month_names = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]

plt.figure(figsize=(10, 8))
sns.heatmap(
    heatmap_data,
    cmap='YlGnBu',
    annot=True,
    fmt=".0f",
    cbar_kws={'label': 'Number of Signups'},
    linewidths=0.5,       # Set linewidth for grid
    linecolor='gray'      # Set color for grid lines
)

# Set title and labels
plt.title('Signup Seasonality Heatmap')
plt.xlabel('Month')
plt.ylabel('Year')

# Set x-ticks with month names centered
plt.xticks(ticks=[x + 0.5 for x in range(12)], labels=month_names, rotation=45, ha='center')
# Set y-ticks with integer years and centered alignment
plt.yticks(ticks=[y + 0.5 for y in range(len(heatmap_data.index))], labels=heatmap_data.index.astype(int), rotation=0)

plt.show()
```

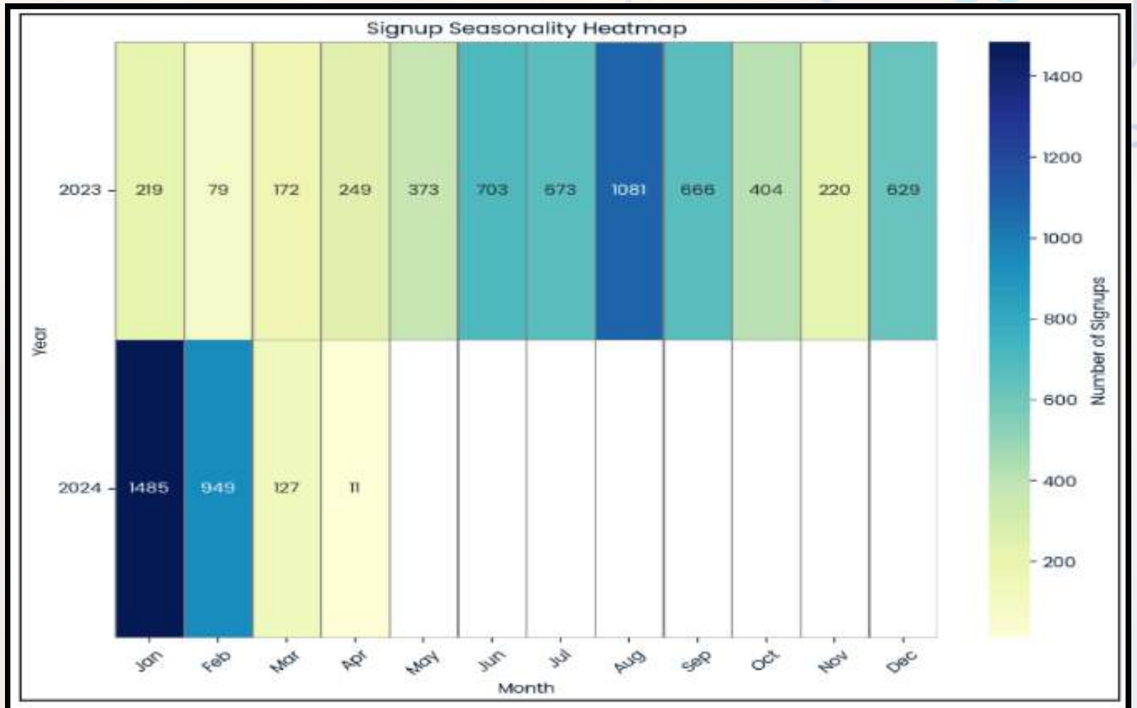


Image 3.1.2 Shows the heatmap of signup seasonality per month and year.

DATA VISUALIZATIONS

Completion Trends

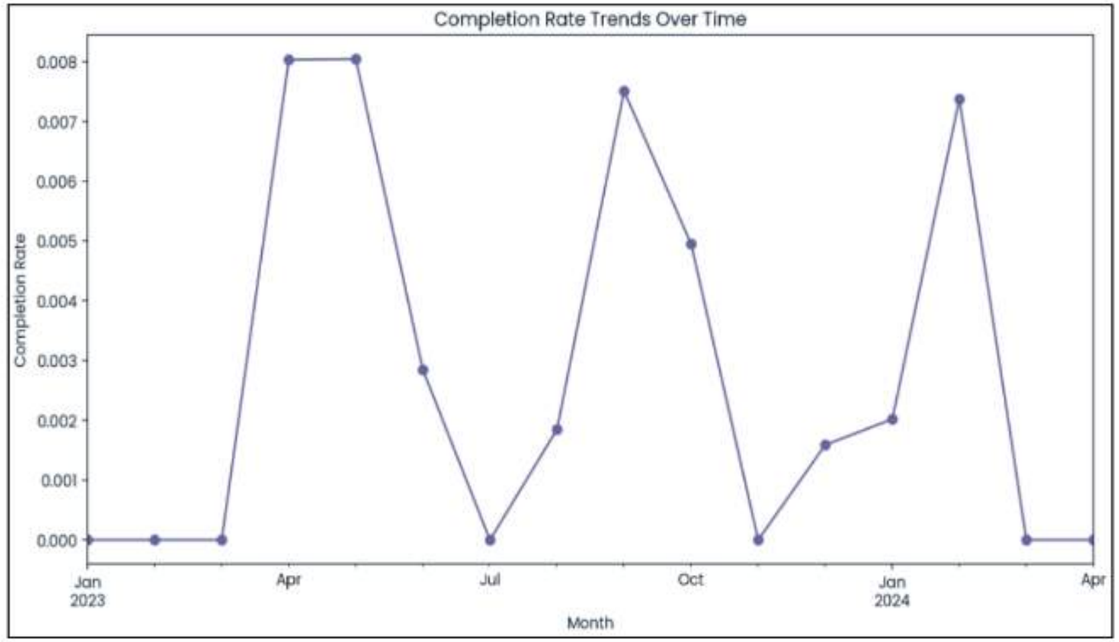


Image 3.2.1 Shows line graph of the completion rate trends over time per month and year.

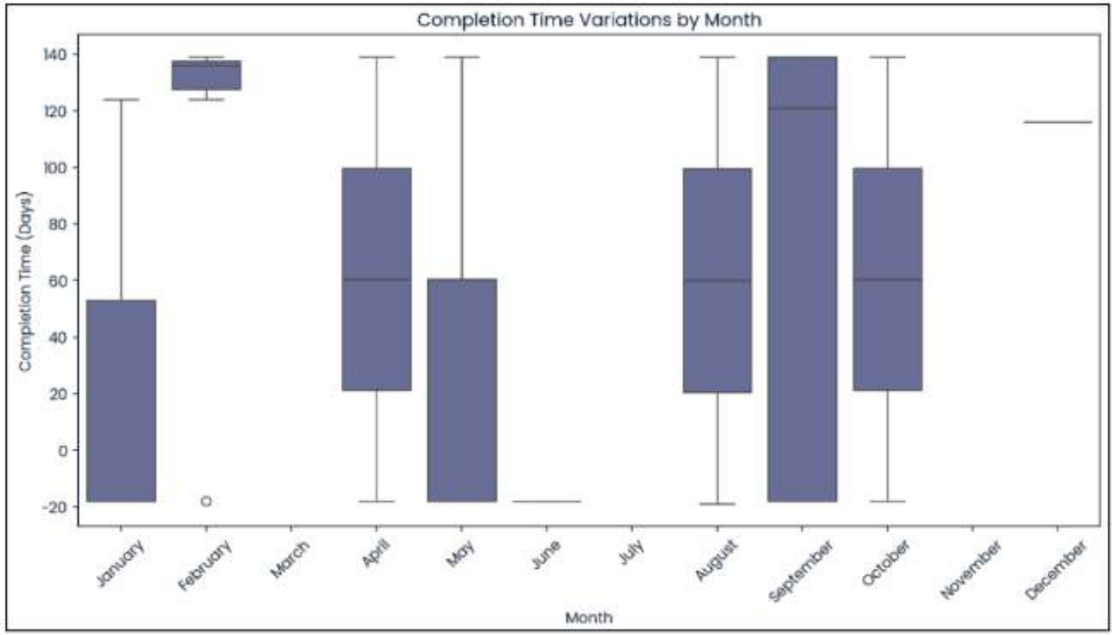


Image 3.2.2 Shows box plot of the completion time variations in days by month.

Python Codes

```python
# 3. Completion Trends Analysis
# 3.1 Stability in Completion Rates
completion_rate_by_month = cleaned_df.groupby(cleaned_df['Learner SignUp DateTime'].dt.to_period("M"))['Completion'].mean()

plt.figure(figsize=(12, 6))
completion_rate_by_month.plot(kind='line', marker='o', title='Completion Rate Trends Over Time')
plt.xlabel('Month')
plt.ylabel('Completion Rate')
plt.show()
```

Python Codes

```python
# 3.2 Variations in Completion Times
plt.figure(figsize=(12, 6))
sns.boxplot(x=cleaned_df['SignUp Month'], y=cleaned_df['Completion Time (Days)'], order=['January', 'February', 'March',
                                                                                          'April', 'May', 'June', 'July',
                                                                                          'August', 'September', 'October',
                                                                                          'November', 'December'])

plt.title('Completion Time Variations by Month')
plt.xlabel('Month')
plt.ylabel('Completion Time (Days)')
plt.xticks(rotation=45)
plt.show()
```

DATA VISUALIZATIONS

Patterns and Correlations

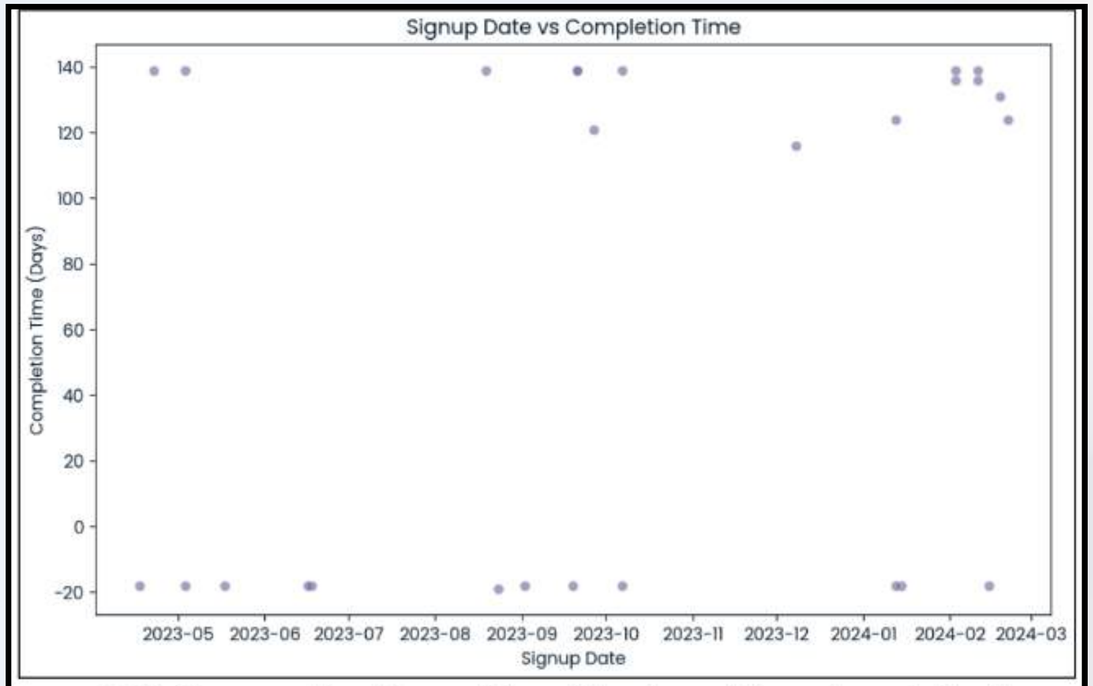


Image 3.3.1 Shows scatterplot correlation of the signup date and completion time.

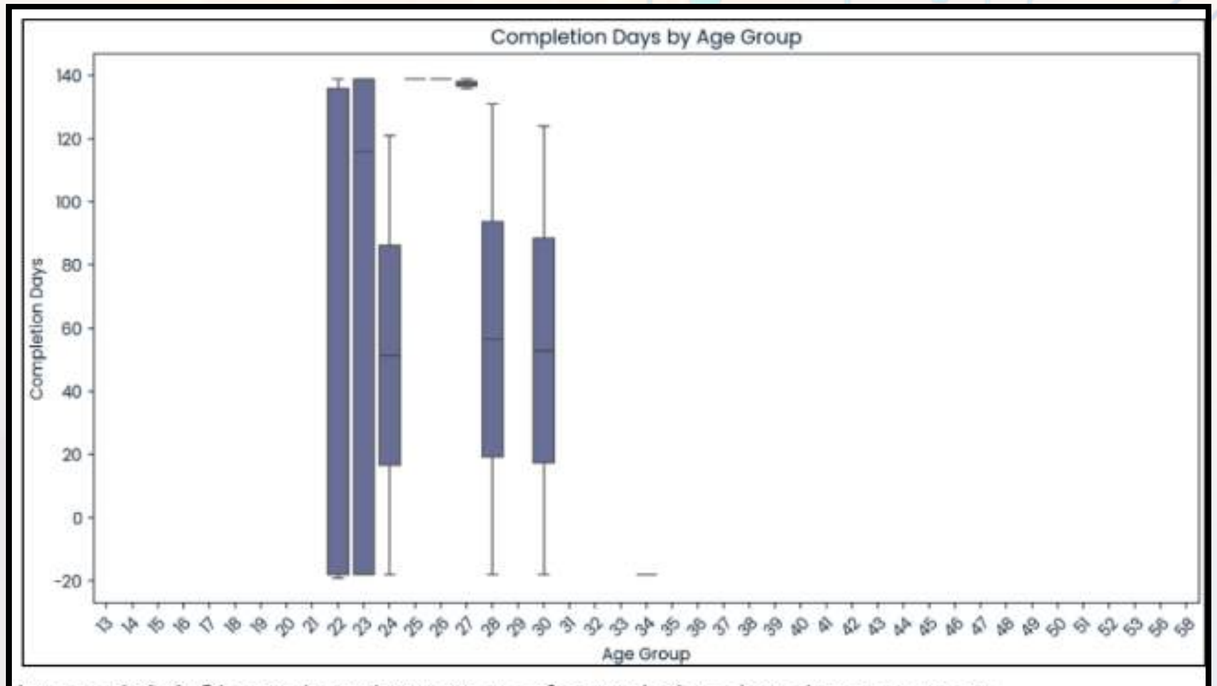


Image 3.3.2 Shows boxplot pattern of completion days by age group.

Python Codes

```python
# 4.1 Signup vs. Completion
plt.figure(figsize=(10, 6))
sns.scatterplot(x=cleaned_df['Learner SignUp DateTime'], y=cleaned_df['Completion Time (Days)'], alpha=0.6)
plt.title('Signup Date vs Completion Time')
plt.xlabel('Signup Date')
plt.ylabel('Completion Time (Days)')
plt.show()
```

Python Codes

```python
# 4.2 Age demographic analysis
plt.figure(figsize=(12, 6))
sns.boxplot(x='Age Calculation', y='Completion Time (Days)', data=cleaned_df)
plt.title('Completion Days by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Completion Days')
plt.xticks(rotation=45)
plt.show()
```

DATA VISUALIZATIONS

Outliers and Anomalies

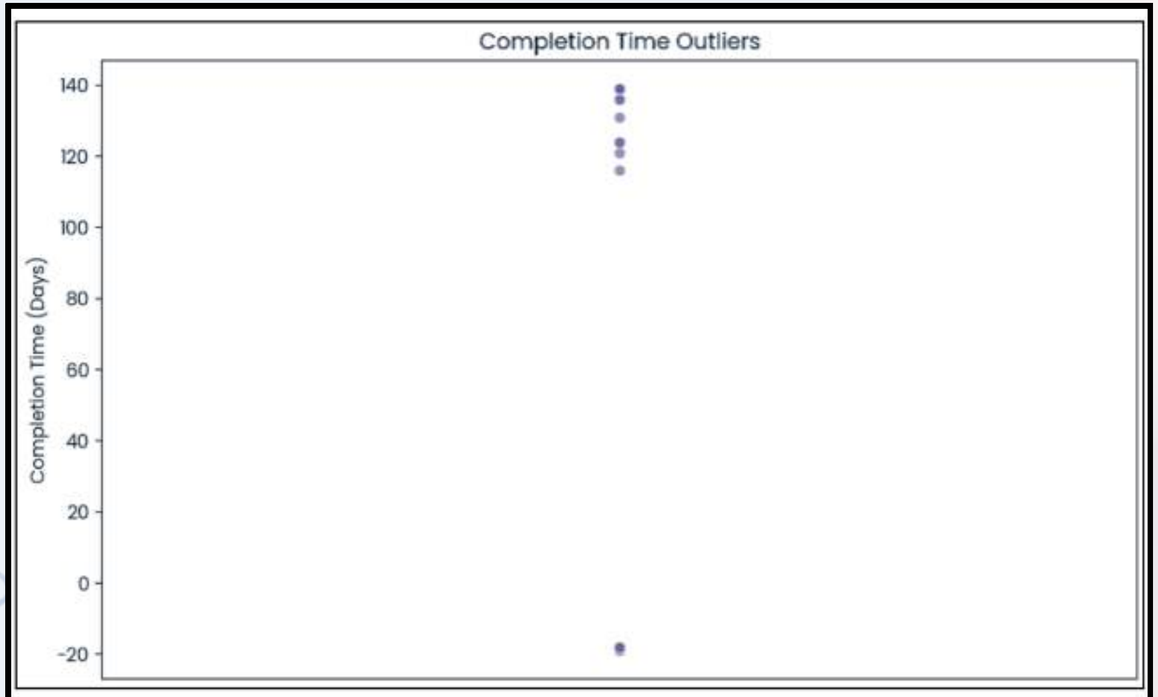


Image 3.4.1 Shows scatterplot pattern of completion time in days outliers.

Python Codes

```python
# 5.1 Completion Time Outliers
plt.figure(figsize=(10, 6))
sns.scatterplot(x=[0] * len(cleaned_df), y=cleaned_df['Completion Time (Days)'], alpha=0.7)
plt.title('Completion Time Outliers')
plt.ylabel('Completion Time (Days)')
plt.xticks([])  # Hide x-axis as it's not meaningful here
plt.show()
```

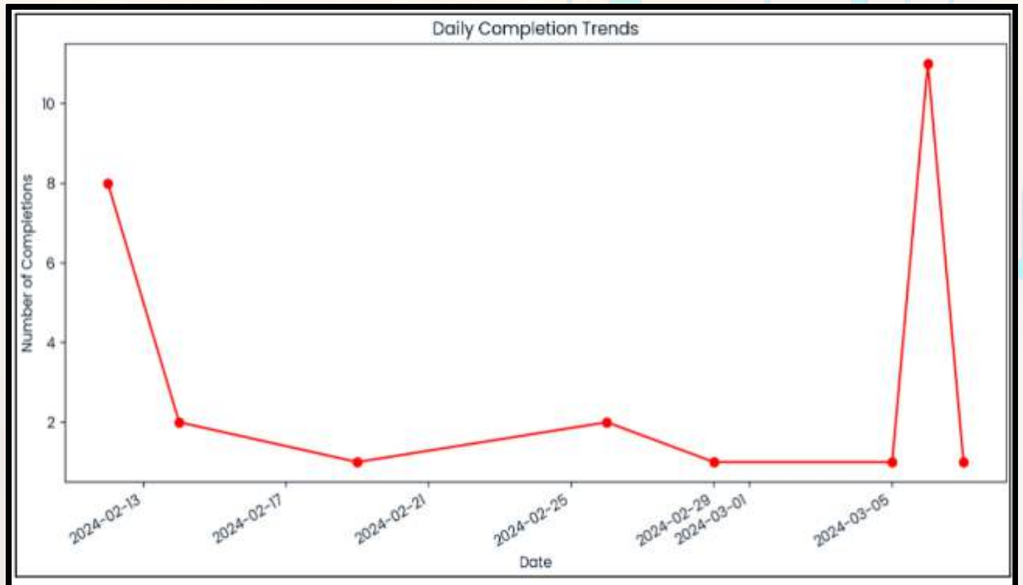


Image 3.4.2 Shows line graph pattern of completion trends over time.

Python Codes

```python
# Identify low completion days
import matplotlib.dates as mdates
completions_by_day = cleaned_df.groupby(cleaned_df['Completion Date'].dt.date)['Completion'].count()

plt.figure(figsize=(12, 6))
completions_by_day.plot(kind='line', marker='o', color='red')
plt.title('Daily Completion Trends')
plt.xlabel('Date')
plt.ylabel('Number of Completions')

# Format x-axis with actual dates
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.gcf().autofmt_xdate()  # Slant dates for readability
plt.show()
```

WEEK-3 HIGHLIGHTS

TEAM 10A

PREDICTIVE DATA MODEL

- No.1 Data Preparation
- No.2 Exploratory Data Analysis
- No.3 Data Visualization
- No.4 Model Selection and Training

CHURN ANALYSIS

- No.1 Performance Metrics
- No.2 Feature Importance
- No.3 SHAP Values
- No.4 Churn Probability

DATA PREPARATION IN PYTHON

TEAM 10A

Reference Dataset

learner_engagement.csv

A B C

FILE ORIGIN

Derived from Excelerate online education platform that tracks user activities, such as signups, course completions, and engagement scores.

The dataset contains 8,203 rows and 46 columns

KEY COLUMNS

- Learner ID
- SignUp Date
- Opportunity End Date
- Engagement Scores
- Completion Status

KEY FEATURES

- Signup Data
- Completion Information
- Completion Time
- Engagement Scores
- Segmenting Fields

DATA PREPARATION IN PYTHON

TEAM 10A

Cleaned Dataset

eda_engagement.csv

A	B	C	D

Python Codes

```python
import pandas as pd
import numpy as np

# Load the dataset
file_path = 'learner_engagement.csv'
learner_engagement_df = pd.read_csv(file_path)

# Remove duplicates
learner_engagement_df.drop_duplicates(inplace=True)

# Handle missing values
# Fill missing categorical values with 'Unknown' and numeric with median or zero
for column in learner_engagement_df.columns:
    if learner_engagement_df[column].dtype == 'object':
        learner_engagement_df[column].fillna('Unknown', inplace=True)
    else:
        learner_engagement_df[column].fillna(learner_engagement_df[column].median(), inplace=True)

# Correct data types
# Convert date columns to datetime format
learner_engagement_df['Learner SignUp DateTime'] = pd.to_datetime(learner_engagement_df['Learner SignUp DateTime'], errors='coerce')
learner_engagement_df['Opportunity End Date'] = pd.to_datetime(learner_engagement_df['Opportunity End Date'], errors='coerce')
learner_engagement_df['Opportunity Start Date'] = pd.to_datetime(learner_engagement_df['Opportunity Start Date'], errors='coerce')

# Drop unnecessary columns if needed (example with placeholder columns)
# Modify the list of columns to drop as needed
columns_to_drop = ['Opportunity Id', 'First Name', 'Last Name', 'Mobile Number', 'Address Line 1', 'Address Line 2', 'Zip Code', 'Badge Id',
'Badge Name', 'Opportunity Duration (Days)', 'Opportunity Engagement (Days)', 'Age and Duration Interaction', 'Duration Weight (50%)', 'Age
Weight (30%)', 'Gender Weight (20%)', 'Engagement Scores', 'Normalized Reward Amount', 'Gender Data Encoding', 'Status Code', 'Normalized
Skill Points Earned']  # Replace with any columns you consider irrelevant
learner_engagement_df.drop(columns=columns_to_drop, inplace=True, errors='ignore')

# Save the cleaned dataset
eda_cleaned_file_path = 'eda_engagement.csv'
learner_engagement_df.to_csv(eda_cleaned_file_path, index=False)
print(f"Cleaned data saved to {eda_cleaned_file_path}")
```

CLEANING POINTS

- Duplicate removal
- Missing value handling
- Data type correction
- Unnecessary column removal
- Renaming and saving new dataset

DATA PREPARATION IN PYTHON

TEAM 10A

Dataset with New Features

features_and_scores.csv

A	B	C	D

Python Codes

```python
# Activity Duration in days (intended duration of the activity)
eda_cleaned_df['Activity Duration (Days)'] = (eda_cleaned_df['Opportunity End Date'] - eda_cleaned_df['Opportunity Start Date']).dt.days

# Completion Rate (binary: 1 if completed, 0 otherwise)
eda_cleaned_df['Completion Status'] = eda_cleaned_df['Completion Date'].notna().astype(int)

# Reward Engagement (normalize the Reward Amount)
if 'Reward Amount' in eda_cleaned_df.columns:
    eda_cleaned_df['Normalized Reward Engagement'] = (eda_cleaned_df['Reward Amount'] - eda_cleaned_df['Reward Amount'].min()) / (eda_cleaned_df['Reward Amount'].max(
- eda_cleaned_df['Reward Amount'].min())

# Frequency of Participation (e.g., in the last year) - requires unique user ID
eda_cleaned_df['Participation Frequency'] = eda_cleaned_df.groupby('Profile Id')['Profile Id'].transform('count')

# Time to Completion (for those who completed)
eda_cleaned_df['Time to Completion (Days)'] = (eda_cleaned_df['Completion Date'] - eda_cleaned_df['Opportunity Start Date']).dt.days
eda_cleaned_df['Time to Completion (Days)'] = eda_cleaned_df['Time to Completion (Days)'].fillna(eda_cleaned_df['Activity Duration (Days)'])

# New Engagement Score Calculation - Combine the above metrics
# A simple weighted sum can serve as a new engagement score
eda_cleaned_df['Engagement Score'] = (
    eda_cleaned_df['Completion Status'] * 0.4 +               # 40% weight on completion
    eda_cleaned_df['Normalized Reward Engagement'].fillna(0) * 0.3 +  # 30% weight on reward engagement
    (1 / (1 + eda_cleaned_df['Time to Completion (Days)'])) * 0.2 +   # 20% weight on time to completion (lower time, higher score)
    (eda_cleaned_df['Participation Frequency'] / eda_cleaned_df['Participation Frequency'].max()) * 0.1  # 10% weight on participation frequency
)

# Replace blank cells in 'Opportunity Start Date' and 'Completion Date' with 'Null' string
eda_cleaned_df['Opportunity Start Date'] = eda_cleaned_df['Opportunity Start Date'].fillna('Null')
eda_cleaned_df['Completion Date'] = eda_cleaned_df['Completion Date'].fillna('Null')

# Replace blank cells in 'Activity Duration (Days)' and 'Time to Completion (Days)' with zero integer
eda_cleaned_df['Activity Duration (Days)'] = eda_cleaned_df['Activity Duration (Days)'].fillna(0).astype(int)
eda_cleaned_df['Time to Completion (Days)'] = eda_cleaned_df['Time to Completion (Days)'].fillna(0).astype(int)

# Replace blank cells in 'New Engagement Score' with zero float
eda_cleaned_df['Engagement Score'] = eda_cleaned_df['Engagement Score'].fillna(0.000000).astype(float)

# Display the updated DataFrame with new features
eda_cleaned_df[['Activity Duration (Days)', 'Completion Status', 'Normalized Reward Engagement',
      'Participation Frequency', 'Time to Completion (Days)', 'Engagement Score']].head()

# Save the cleaned dataset
features_and_scores_file_path = 'features_and_scores.csv'
eda_cleaned_df.to_csv(features_and_scores_file_path, index=False)
```

FEATURE SELECTION

- Activity Duration (Days)
- Completion Status
- Normalized Reward Engagement
- Participation Frequency
- Time to Completion (Days)
- Engagement Score

DATA EXPLORATION IN PYTHON

TEAM 10A

Cleaned data saved to eda_engagement.csv

Table | Chart | Filter | Columns | Search

	Age Calculation	Reward Amount	Activity Duration (Days)	Completion Status	Participation Frequency	Time to Completion (
count	8202	8202	8202	8202	8202	
mean	24.5598634479	0.2926115582	131.0164594001	0.00329188	4.111680078	130.7
std	4.3339882437	6.3366681529	230.9502846787	0.0572838863	3.381229902	230.5
min	13	0	-24	0	1	
25%	22	0	0	0	1	
50%	24	0	0	0	3	
75%	26	0	94	0	6	
max	58	200	604	1	15	

Table No. 1 shows the descriptive summary statistics of the post-processed dataset in python

STATISTICS CALCULATED

- Count
- Mean
- Standard Deviation (Std)
- Minimum
- 25% (Q1): First quartile
- 50% (Median or Q2)
- 75% (Q3): Third quartile
- Maximum

Python Codes

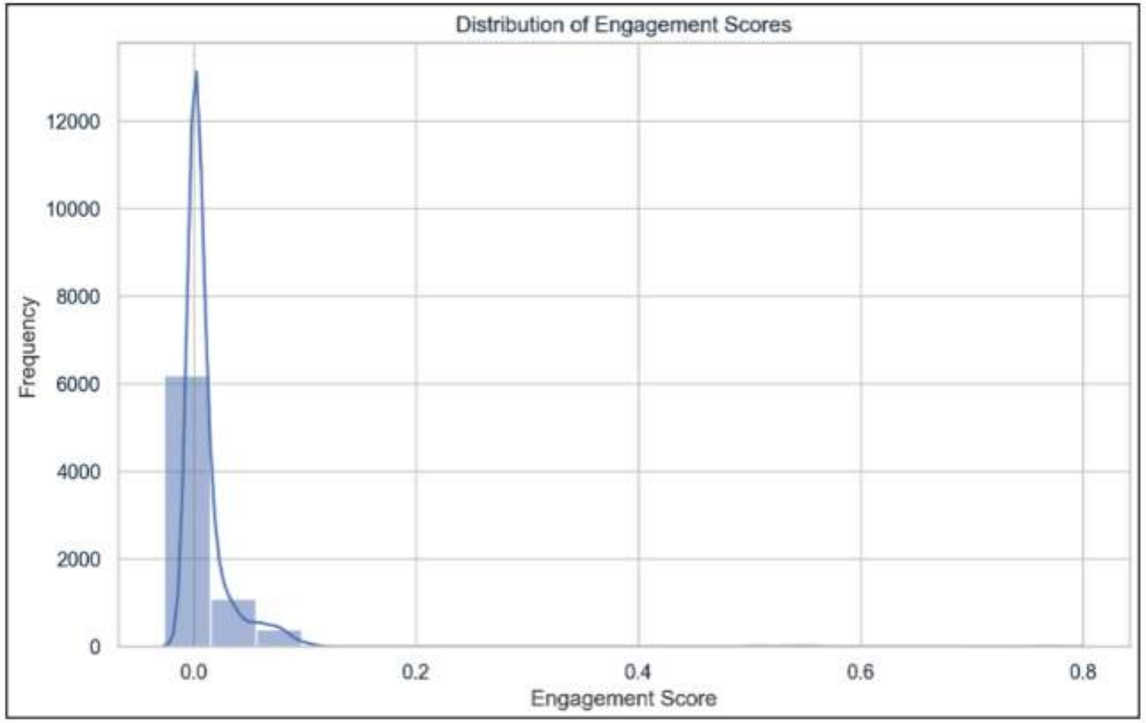
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the features and scores data
features_and_scores_df = pd.read_csv(features_and_scores_file_path)

# Calculate descriptive statistics for numerical columns excluding 'Learner SignUp Year' and 'Normalized Reward Engagement'
descriptive_stats = features_and_scores_df.drop(columns=['Learner SignUp Year', 'Normalized Reward Engagement']).describe(include=[float, int])
descriptive_stats
```

NOTE:
excluding 'Learner SignUp Year' and Normalized Reward Engagement' in the features_and_scores_df DataFrame.

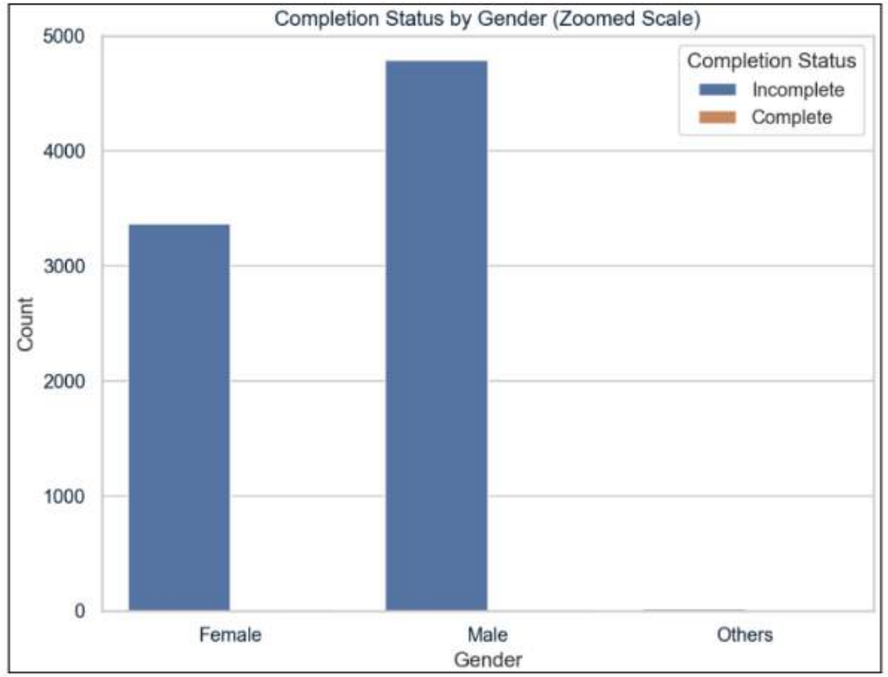
DATA VISUALIZATION IN PYTHON



Graph No. 1 shows the histogram that displays the distribution of engagement scores

Python Codes

```python
# Histogram for New Engagement Score with bars breakdown in granular view
plt.figure(figsize=(10, 6))
histplot = sns.histplot(features_and_scores_df['Engagement Score'], bins=20, kde=True)  # Increased number of bins for granular view
plt.title('Distribution of Engagement Scores')
plt.xlabel('Engagement Score')
plt.ylabel('Frequency')
plt.show()
```

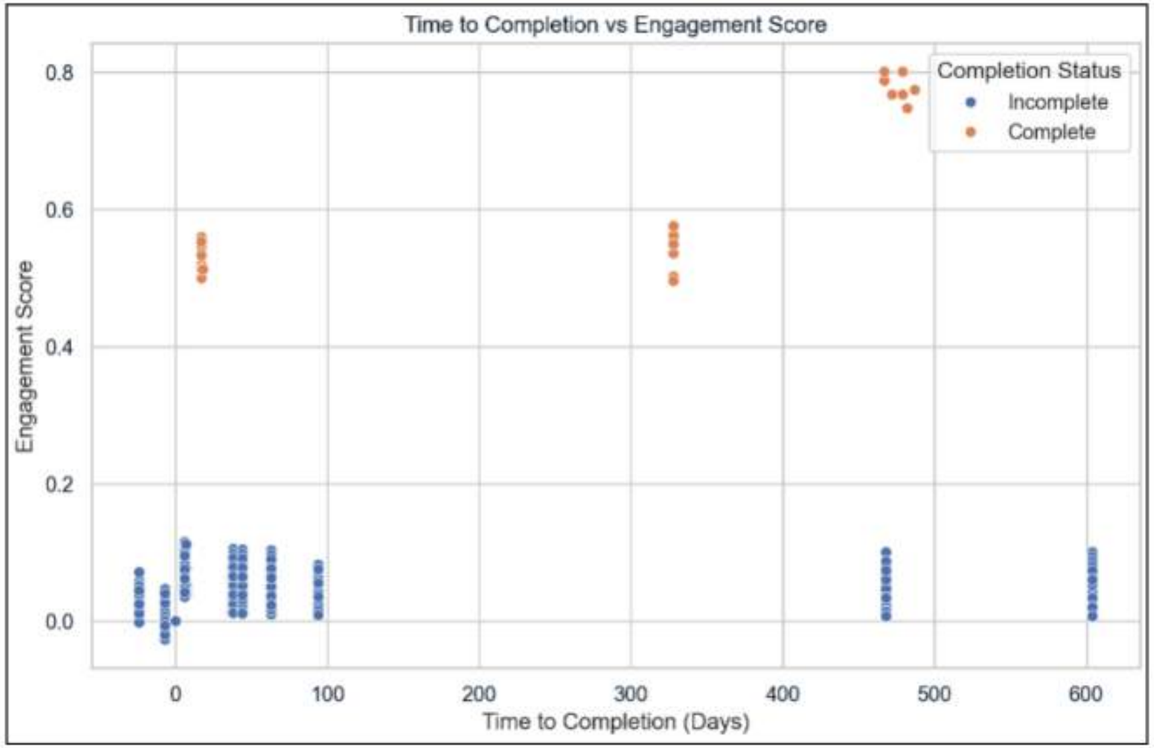


Graph No. 2 shows the distribution of completion status for different genders

Python Codes

```python
# Completion Status by Gender for Both Completed and Incomplete Count in Zoomed Scale
plt.figure(figsize=(8, 6))
sns.countplot(x='Gender', hue='Completion Status', data=features_and_scores_df)
plt.ylim(0, 5000)  # Adjust the y-axis limit to zoom in on low counts
plt.title('Completion Status by Gender (Zoomed Scale)')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Completion Status', labels=['Incomplete', 'Complete'])
plt.show()
```

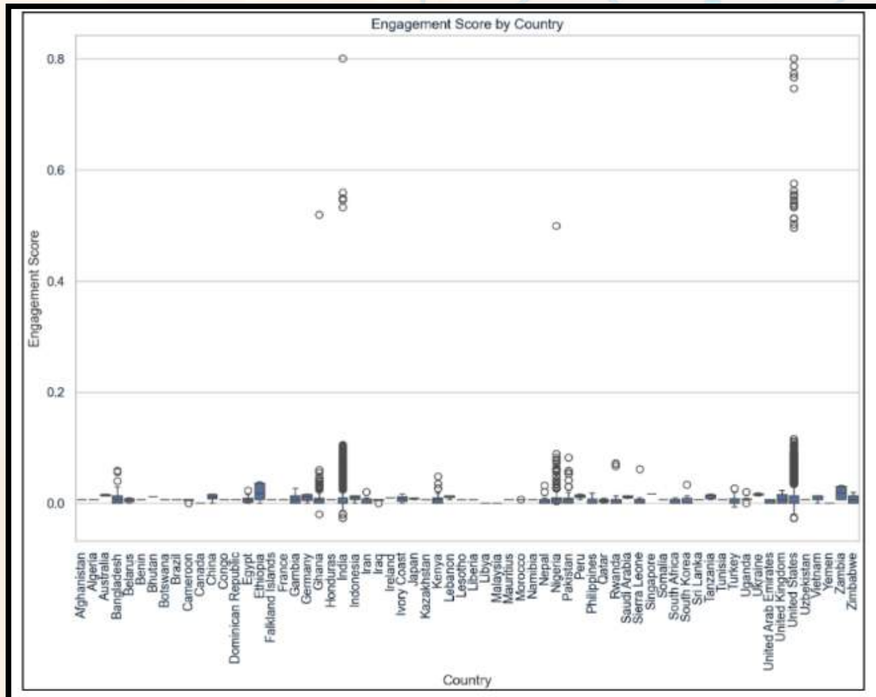
DATA VISUALIZATION IN PYTHON



Graph No. 3 shows the relationship between two variables, the Time to Completion (Days) and Engagement Score

Python Codes

```python
# Scatter plot of Time to Completion vs. Engagement Score
plt.figure(figsize=(10, 6))
scatterplot = sns.scatterplot(x='Time to Completion (Days)', y='Engagement Score', hue='Completion Status', data=features_and_scores_df)
plt.title('Time to Completion vs Engagement Score')
plt.xlabel('Time to Completion (Days)')
plt.ylabel('Engagement Score')
```



Graph No. 4 shows a box plot that displays the distribution of engagement scores across different countries.

Python Codes

```python
# Boxplot for new Engagement Score by City (to see city-wise engagement)
plt.figure(figsize=(12, 8))
# Sort the dataframe by 'Country' to arrange country names in alphabetical order
features_and_scores_df_sorted = features_and_scores_df.sort_values('Country')
sns.boxplot(x='Country', y='Engagement Score', data=features_and_scores_df_sorted)
plt.xticks(rotation=90)
plt.title('Engagement Score by Country')
plt.xlabel('Country')
plt.ylabel('Engagement Score')
plt.show()
```

DATA VISUALIZATION IN PYTHON

TEAM 10A

3.3 Patterns and Trends

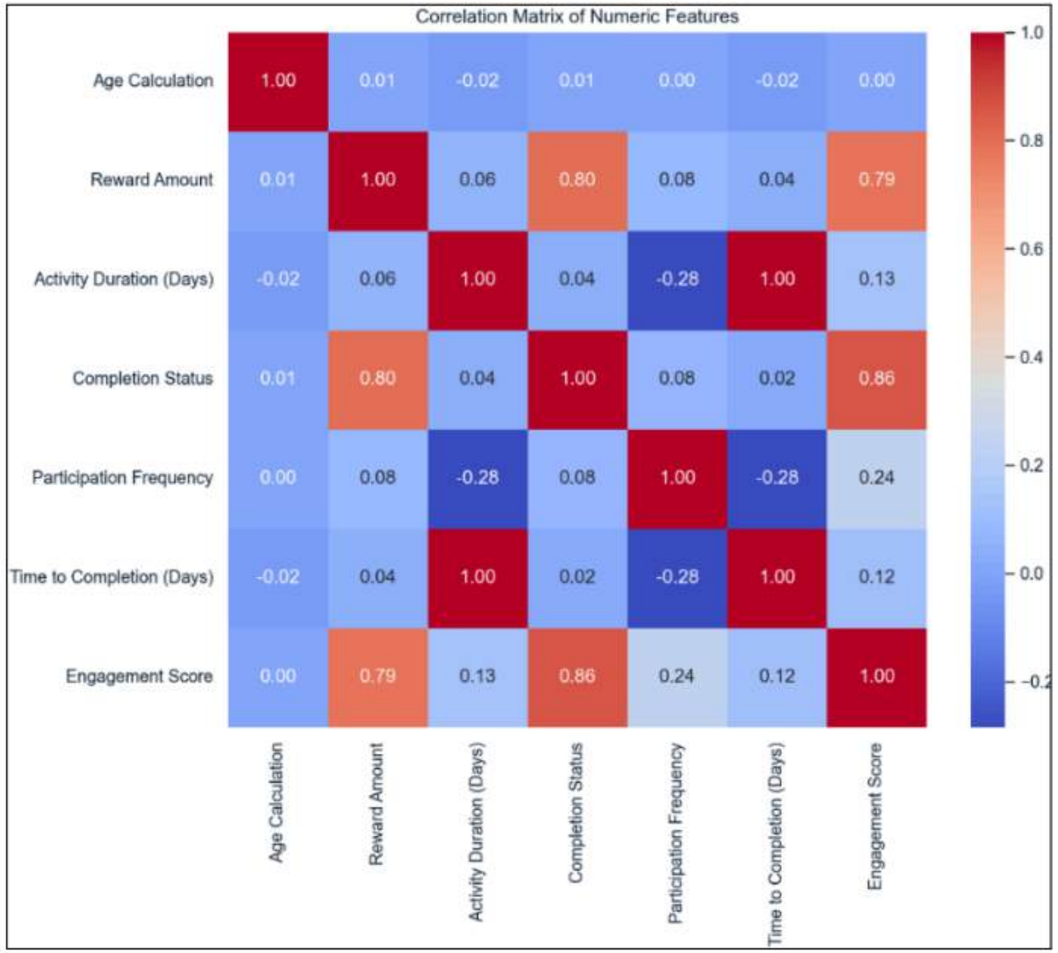


Image No. 1 is a correlation matrix, which visually represents the relationships between different numerical variables.

ANOMALIES CHECKING

```
Anomalies in Engagement Scores:
             City  Gender  Engagement Score  Completion Status
0        Peshawar  Female          0.006997                  0
1       Vikarabad  Female          0.006997                  0
2         Chicago    Male          0.006997                  0
3     Saint Louis  Female          0.006997                  0
4     Saint Louis    Male          0.006997                  0
...           ...     ...               ...                ...
8197  Addis Ababa  Female               inf                  0
8198  Saint Louis    Male               inf                  0
8199       Kosofe    Male               inf                  0
8200  Saint Louis  Female               inf                  0
8201   Hanamkonda    Male               inf                  0

[8177 rows x 4 columns]
```

Python Codes

```python
# Select only the numeric columns for correlation calculation, excluding 'Learner SignUp Year' and 'Normalized Reward Engagement'
numeric_data = features_and_scores_df.drop(columns=['Learner SignUp Year', 'Normalized Reward Engagement']).select_dtypes(include=['number'])

# Calculate the correlation matrix for numeric columns
correlation_matrix = numeric_data.corr()

# Plot the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numeric Features')
plt.show()

#Check for anomalies in engagement scores (e.g., very low or high scores)
anomalies = features_and_scores_df[(features_and_scores_df['Engagement Score'] < 0.2) | (features_and_scores_df['Engagement Score'] > 0.8)]
print("Anomalies in Engagement Scores:\n", anomalies[['City', 'Gender', 'Engagement Score', 'Completion Status']])
```

MODEL SELECTION AND TRAINING IN PYTHON

TEAM 10A

- LOGISTIC REGRESSION
- DECISION TREE
- RANDOM FOREST

Python Codes

```python
# Model Selection and Training
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42)
}

# Dictionary to store model performances
model_performance = {}
```

Python Codes

```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder

file_path = 'features_and_scores.csv'
features_and_scores_df = pd.read_csv(file_path)

# Identify categorical columns
categorical_columns = features_and_scores_df.select_dtypes(include=['object']).columns

# Encode categorical columns if necessary
for col in categorical_columns:
    features_and_scores_df[col] = LabelEncoder().fit_transform(features_and_scores_df[col])

# Separate features (X) and target (y)
X = features_and_scores_df.drop(columns=['Engagement Score'])  # 'Engagement Score' is the target variable
y = features_and_scores_df['Completion Status']

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Check for any NaN or infinite values in X_train and X_test
X_train = X_train.replace([np.inf, -np.inf], np.nan)  # Replace infinite values with NaN
X_test = X_test.replace([np.inf, -np.inf], np.nan)
X_train = X_train.fillna(X_train.mean())  # Replace NaNs with column mean in X_train
X_test = X_test.fillna(X_train.mean())  # Replace NaNs in X_test with mean from X_train

# Scale features for better model performance (optional but recommended for Logistic Regression)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

MODEL PERFORMANCE METRICS

- ACCURACY
- PRECISION
- RECALL
- F1 SCORE
- CLASSIFICATION REPORT

Classification Report for Decision Tree:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2453
1	1.00	1.00	1.00	8
accuracy			1.00	2461
macro avg	1.00	1.00	1.00	2461
weighted avg	1.00	1.00	1.00	2461

Image No. 3 shows the Classification Report for Decision Tree

Classification Report for Logistic Regression:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2453
1	1.00	1.00	1.00	8
accuracy			1.00	2461
macro avg	1.00	1.00	1.00	2461
weighted avg	1.00	1.00	1.00	2461

Image No. 2 shows the Classification Report for Logistic Regression

Classification Report for Random Forest:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2453
1	1.00	1.00	1.00	8
accuracy			1.00	2461
macro avg	1.00	1.00	1.00	2461
weighted avg	1.00	1.00	1.00	2461

Image No.4 shows the Classification Report for Random Forest

Python Codes

```python
# Train and evaluate each model
for model_name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Calculate performance metrics
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='binary', pos_label=y_test.unique()[0])
    recall = recall_score(y_test, y_pred, average='binary', pos_label=y_test.unique()[0])
    f1 = f1_score(y_test, y_pred, average='binary', pos_label=y_test.unique()[0])

    # Store the results
    model_performance[model_name] = {
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1
    }

    # Print classification report
    print(f"Classification Report for {model_name}:\n")
    print(classification_report(y_test, y_pred))
    print("\n")

# Display the model performance summary
print("Model Performance Summary:\n", model_performance)
```

FEATURE IMPORTANCE

```
Feature Importance for Random Forest Model (Selected Features):
Skill Points Earned: 0.1602
Skills Earned: 0.1288
Reward Amount: 0.1217
Time to Completion (Days): 0.0067
Participation Frequency: 0.0034
Current/Intended Major: 0.0003
Activity Duration (Days): 0.0001
Opportunity Category: 0.0000
Age Calculation: 0.0000
Gender: 0.0000
Country: 0.0000
```

Image No.5 shows the Feature Importance for Random Forest Model

Python Codes

```python
# Feature Importance for tree-based models
selected_features = [
    'Skill Points Earned', 'Skills Earned', 'Reward Amount', 'Time to Completion (Days)',
    'Participation Frequency', 'Age Calculation', 'Gender', 'Opportunity Category',
    'Country', 'Activity Duration (Days)', 'Current/Intended Major'
]

feature_importance = {}
if 'Random Forest' in models:
    rf_model = models['Random Forest']
    importance_values = rf_model.feature_importances_
    feature_importance = dict(zip(X.columns, importance_values))
    filtered_feature_importance = {k: v for k, v in feature_importance.items() if k in selected_features}
    sorted_features = sorted(filtered_feature_importance.items(), key=lambda x: x[1], reverse=True)

    print("\nFeature Importance for Random Forest Model (Selected Features):")
    for feature, importance in sorted_features:
        print(f"{feature}: {importance:.4f}")

# Visualize Feature Importance (requires matplotlib)
import matplotlib.pyplot as plt
```

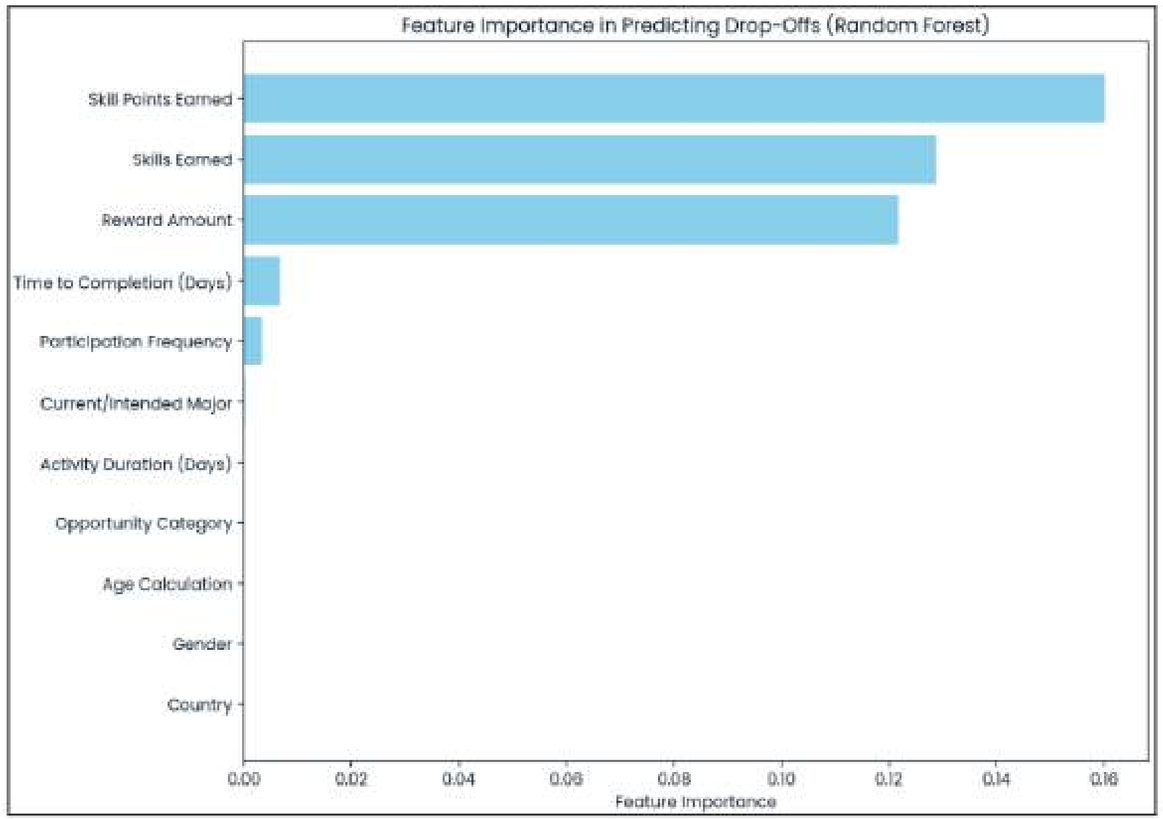


Image No. 6 shows the relative importance of different features in predicting whether a user will drop off from a particular course.

SHAP (SHAPLEY ADDITIVE EXPLANATIONS)

```
Top Factors Contributing to Churn:
                    Feature  Average Impact on Model Output
3    Participation Frequency                       0.128613
1              Reward Amount                       0.082596
0            Age Calculation                       0.032329
2   Activity Duration (Days)                       0.025478
4  Time to Completion (Days)                       0.001631
```

Image No. 8 provides a ranking of factors that contribute to student drop-off, as determined by a machine learning model. A higher average impact on model output indicates a stronger influence on predicting whether a student will drop out.

Python Codes

```python
# Scale features for better model performance (especially for Logistic Regression)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train a classifier (using logistic regression for interpretability)
model = LogisticRegression(random_state=42, max_iter=1000)
model.fit(X_train_scaled, y_train)

# Initialize SHAP for interpretability
explainer = shap.LinearExplainer(model, X_train_scaled, feature_names=X.columns)
shap_values = explainer.shap_values(X_train_scaled)

# Plot SHAP summary to highlight primary factors for churn
shap.summary_plot(shap_values, X_train_scaled, feature_names=X.columns)
```

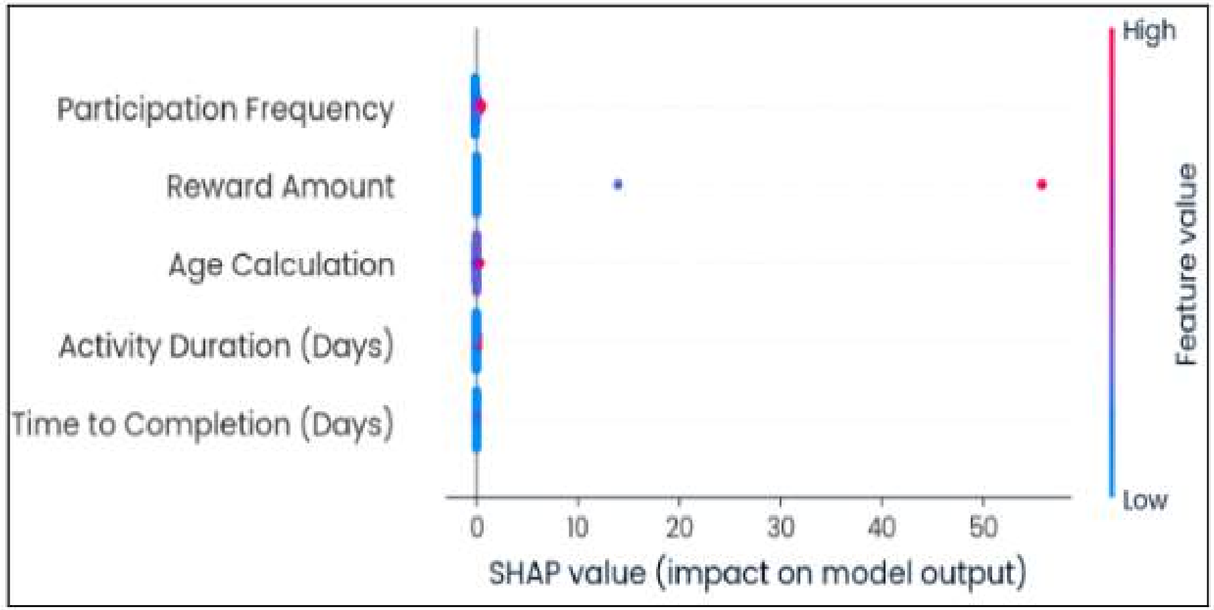


Image No.7 shows a SHAP (SHapley Additive exPlanations) summary plot SHAP is a technique used to understand the contribution of each feature in a machine learning model's prediction

FILTERING LEARNERS AT RISK OF CHURN

TEAM 10A

List of Learners Dataset

at_risk_learners.csv

	Age Calculation	Reward Amount	Activity Duration (Days)	Participation Frequency	Time to Completion (Days)	Churn Probability	At Risk
204	23	200	604	11	487	1	True
5455	23	50	-1	8	17	0.9840564374	True
1115	22	200	604	15	467	1	True
5568	23	50	-1	4	17	0.9808313311	True
5307	22	50	468	15	328	0.9890257044	True

Python Codes

```python
# Train a classifier (using logistic regression for interpretability)
model = LogisticRegression(random_state=42, max_iter=1000)
model.fit(X_train_scaled, y_train)

# Predict churn probability for the test set
churn_probabilities = model.predict_proba(X_test_scaled)[:, 1]  # Probability of churn (assuming class 1 indicates churn)

# Define a threshold for identifying at-risk learners (e.g., 0.7 or higher indicates high churn risk)
threshold = 0.7
at_risk_learners = X_test.copy()
at_risk_learners['Churn Probability'] = churn_probabilities
at_risk_learners['At Risk'] = at_risk_learners['Churn Probability'] >= threshold

# Filter out the learners who are at risk
at_risk_learners = at_risk_learners[at_risk_learners['At Risk']]

# Display the at-risk learners with relevant details
print("List of learners needing attention based on churn analysis:")
print(at_risk_learners[['Churn Probability']].sort_values(by='Churn Probability', ascending=False))

# Optionally, save the list to a CSV file for further review
at_risk_learners.to_csv("at_risk_learners.csv", index=False)

# Display the head of the at-risk learners with relevant details
at_risk_learners.head()
```

```
List of learners needing attention based on churn analysis:
      Churn Probability
204            1.000000
1115           1.000000
5307           0.989026
5615           0.984775
5455           0.984056
5194           0.982562
5285           0.981533
5568           0.980831
```

RECOMMENDATIONS AND CONCLUSIONS

TEAM 10A

STRATEGIES AND INTERVENTIONS

- No.1 Encourage Active Participation
- No.2 Optimize Reward Systems
- No.3 Identify and Support At-Risk Students
- No.4 Monitor Student Progress and Provide Timely Feedback
- No.5 Create a Supportive Learning Environment

FACTORS CONTRIBUTING CHURN

- No.1 Participation Frequency
- No.2 Reward Amount
- No.3 Age Group
- No.4 Activity Duration (Days)
- No.5 Time to Completion (Days)

FUTURE WORKS

- No.1 Deeper Dive into Age and Churn
- No.2 Impact of Activity Type and Difficulty
- No.3 Long-Term Impact of Rewards
- No.4 Effectiveness of Support Interventions
- No.5 Continuous Monitoring and Adaptation