Excelerate AI-Powered Virtual Internship

May - June 2025

Week 1: Data Cleaning and Feature Engineering Report

Date of Submission: May 19, 2025

Team E Members

Sr NO.	Name	Designation	Email
1	Diya Kharel	Team Lead	diyakharel4@gmail.com
2	Hasna Hena Borsha	Project Manager	hasnahenaborsharu@gmail.com
3	Dunisha De Silva	Project Scribe	desilvadunisha17@gmail.com
4	Iqra Shaikh	Project Lead	sh.iqratasleem@gmail.com
5	Faith Odhe	Team Member	Faith.t.odhe@gmail.com
6	Bindu N Gowda	Team Member	bindungowda2004@gmail.com

Table of Contents

1. Introduction

a) Purpose

This report provides comprehensive documentation of the data cleaning and feature engineering activities carried out during **Week 1**. The primary goal was to prepare the raw dataset for in-depth analysis by addressing various data quality issues. These included:

- Handling **missing values**
- Resolving **inconsistent date formats**
- Correcting **illogical chronological sequences**
- Standardizing **text fields**
- Removing **duplicate records**

Additionally, several **new features** were engineered to enhance the dataset's analytical potential and enable more meaningful insights in subsequent stages. This documentation ensures that the dataset is accurate, logically structured, and ready for use in further analysis and modeling.

b) Data Description

The dataset captures **learner registrations for educational opportunities** such as courses, internships, and scholarships. Each record documents key demographic, institutional, and application-related attributes for individual learners.

It originates from a **professional learning platform** and was provided by the **Excelerate team** under the title **"SLU Opportunity Wise Dataset"**.

c) Key Information Captured

Learner Details:

- **First Name**: Learner's given name
- **Gender**: Male, Female, or Other
- **Date of Birth**: For age calculation
- **Country**: Learner's country of residence

Application & Enrollment:

- **Learner SignUp DateTime**: When the learner registered

- **Apply Date**: When the learner applied for the opportunity
- **Opportunity Id**: Unique identifier for each opportunity
- **Status Description**: Describes enrollment status (e.g., Registered, Completed)
- **Status Code**: Integer representing application status

Opportunity Metadata:

- **Opportunity Name**: Title of the course or opportunity
- **Opportunity Start Date & End Date**: Start and end of the opportunity duration
- **Opportunity Category**: Type of opportunity (e.g., Course, Internship)
- **Institution Name**: Name of the educational institution
- **Current/Intended Major**: Field of study
- **Entry Created At**: Timestamp when the record was first created in the system

d) Dataset Composition

The dataset includes **16 columns** with the following data types:

- **Datetime (6 columns)**:
  Learner SignUp DateTime, Opportunity End Date, Date of Birth, Entry Created At, Apply Date, Opportunity Start Date
- **String (9 columns)**:
  Opportunity Id, Opportunity Name, Opportunity Category, First Name, Gender, Country, Institution Name, Current/Intended Major, Status Description
- **Integer (1 column)**:
  Status Code

This structured data foundation is essential for performing high-quality data analysis, trend evaluation, and predictive modeling in the weeks ahead.

2. Data Cleaning Process

To ensure the dataset was accurate, consistent, and ready for analysis, we applied a series of cleaning operations using Python and the Pandas library. We performed the following steps to clean the dataset:

a) Handling Missing Values

We identified missing values in important fields such as 'Institution Name' and 'Apply Date', which are essential for downstream analysis like institutional participation trends and application timing.

- For 'Institution Name', missing values were imputed with the placeholder 'Unknown' to preserve the record while indicating the absence of data.
- For critical date fields (like 'Apply Date', 'Opportunity Start Date', etc.), rows with missing values were **dropped** because such missing timestamps would hinder calculations like durations and signup timelines.

Code:

```
df_cleaned = df_cleaned.assign(
    **{'Institution Name': df_cleaned['Institution Name'].fillna('Unknown')}
)
```

```
# Drop rows with critical missing dates (cannot compute features without them)
df_cleaned = df_cleaned.dropna(subset=[
    'Opportunity Start Date', 'Opportunity End Date', 'Apply Date', 'Learner SignUp DateTime'
])
```

b) Handling Outliers in Date Fields

To ensure all date fields were valid and consistent, we parsed them using pd.to_datetime() with errors='coerce'. This safely converts any invalid or incorrectly formatted date strings into NaT (Not a Time), which allows us to filter out or inspect those entries easily.

The fields cleaned this way included:

- Learner signup and application timestamps
- Dates of birth
- Start and end dates of opportunities
- Record creation timestamps

Code:

```
date_fields = [
        'Learner SignUp DateTime', 'Opportunity End Date', 'Date of Birth',
        'Entry created at', 'Apply Date', 'Opportunity Start Date'
]
for col in date_fields:
        df_cleaned[col] = pd.to_datetime(df_cleaned[col], errors='coerce')
```

c) Standardizing Text Formats

Inconsistent use of casing, leading/trailing spaces, and formatting in string fields can fragment the data (e.g., "nepal" vs "Nepal"). To avoid such issues, we standardized the following columns:

- 'Gender'
- 'Country'
- 'Institution Name'
- 'Current/Intended Major'

We used .str.strip() to remove spaces and .str.title() to enforce consistent capitalization.

Code:

```
df_cleaned['Gender'] = df_cleaned['Gender'].str.strip().str.title()
df_cleaned['Country'] = df_cleaned['Country'].str.strip().str.title()
df_cleaned['Institution Name'] = df_cleaned['Institution Name'].str.strip().str.title()
df_cleaned['Current/Intended Major'] = df_cleaned['Current/Intended Major'].str.strip().str.title()
```

d) Correcting Errors and Manual Mappings

To fix known inconsistencies in naming conventions, we applied mappings using .replace():

- "St. Louis" and "Saint Louis University" were both mapped to "Saint Louis" for uniformity.

- The 'Current/Intended Major' field was further cleaned by removing non-alphabetic characters using regex (not shown here, but part of overall process).

Code

```
df_cleaned['Institution Name'] = df_cleaned['Institution Name'].replace({
    'St. Louis': 'Saint Louis',
    'Saint Louis University': 'Saint Louis'
```

e) Dealing with Duplicates

To ensure each record in the dataset was unique and not repeated, we used .drop_duplicates() to eliminate duplicate rows based on all columns. This is essential to avoid double-counting or skewed analysis.

Code:

```
df_cleaned = df_cleaned.drop_duplicates()
```

f) Handling Inconsistent Categorical Data

Categorical fields like 'Gender' had inconsistent encoding—e.g., 'M', 'F', full words, and even blanks.

- First, we mapped shorthand entries ('M' → 'Male', 'F' → 'Female').
- Then we used .where() to validate values, setting anything outside 'Male' or 'Female' as 'Other'. This ensures clean category buckets for gender-based analysis.

Code:

```
df_cleaned['Gender'] = df_cleaned['Gender'].replace({'F': 'Female', 'M': 'Male'})
df_cleaned['Gender'] =
df_cleaned['Gender'].where(df_cleaned['Gender'].isin(['Male', 'Female']), 'Other')
```

3. Feature Engineering

After cleaning the dataset, we created new features to enrich the information and prepare the data for further analysis and modeling. These features were engineered using date calculations, normalization techniques, and encoding strategies.

a) Age

Purpose:
To estimate the current age of each learner based on their date of birth.

Explanation:
We applied a lambda function that subtracts the year in 'Date of Birth' from the current year (datetime.now().year). If the date is missing (NaT), the function returns np.nan, leaving the value as missing.

Code:

```
df_cleaned['Age'] = df_cleaned['Date of Birth'].apply(
        lambda x: datetime.now().year - x.year if pd.notnull(x) else np.nan
)
```

b) Engagement Duration (Days)

Purpose:
To calculate the number of days between when the learner applied and when the opportunity started. This gives insight into how early or late learners applied.

Explanation:
We subtract 'Opportunity Start Date' from 'Apply Date' using pandas datetime subtraction, which returns a time delta. We then extract the number of days using .dt.days. A negative value indicates the application was submitted **after** the opportunity started (i.e., late application).

Code:

```
df_cleaned['Engagement Duration (Days)'] = (df_cleaned['Apply Date'] -
df_cleaned['Opportunity Start Date']

).dt.days
```

c) Time in Opportunity (Days)

Purpose:
To calculate the full duration of each opportunity, which shows how long the opportunity lasted.

Explanation:
By subtracting the 'Opportunity Start Date' from the 'Opportunity End Date', we obtain a time delta. Using .dt.days, we extract the number of days. This helps identify the typical length of programs or internships.

Code:

```
df_cleaned['Time in Opportunity (Days)'] = (
        df_cleaned['Opportunity End Date'] - df_cleaned['Opportunity Start Date']
).dt.days
```

d) SignUp Month and Year

Purpose:
To extract temporal patterns by identifying when the learner signed up (e.g., peak months, seasonal trends).

Explanation:
We use the .dt accessor to extract the **month** and **year** from 'Learner SignUp DateTime'. These features are useful in trend and time series analysis.

Code:

```
df_cleaned['SignUp Month'] = df_cleaned['Learner SignUp DateTime'].dt.month

df_cleaned['SignUp Year'] = df_cleaned['Learner SignUp DateTime'].dt.year
```

e) Normalized Fields

Purpose:
To scale numeric fields like age and durations between 0 and 1, making them suitable for machine learning models and visualizations.

Explanation:
We use MinMaxScaler from sklearn.preprocessing, which transforms the features to a

common scale. This avoids dominance of one feature over others due to different units or ranges.

Code:

```
scaler = MinMaxScaler()

norm_cols = ['Age', 'Engagement Duration (Days)', 'Time in Opportunity (Days)']

df_cleaned[norm_cols] = scaler.fit_transform(df_cleaned[norm_cols])
```

f) Gender_Male (Binary Encoding)

Purpose:
To convert the categorical gender column into a numerical format suitable for modeling.

Explanation:
This feature encodes:

- 'Male' as **1**
- 'Female' as **0**
- 'Other' as **-1**

This allows for quick gender-based segmentation and is essential for machine learning models that require numeric inputs.

Code:

```
df_cleaned['Gender_Male'] = df_cleaned['Gender'].apply(
        lambda x: 1 if x == 'Male' else (0 if x == 'Female' else -1)
```

4. Feature Examples

This section demonstrates how engineered features were calculated using Python, supported by real-world-style examples to explain their interpretation and analytical value.

a) Age at Signup

```
df_cleaned['Age'] = df_cleaned['Date of Birth'].apply(
    lambda x: datetime.now().year - x.year if pd.notnull(x) else
np.nan
)
```

Explanation:
Calculates each learner's age at the time of processing by subtracting the birth year from the current year. If the birth date is missing, the function returns NaN.

Example:

- **Learner**: Nirmal
- **Date of Birth**: 1998-04-20
- **Assumed Current Date**: 2025-03-23
- **Result**: 26 years
- **Excel Equivalent**: =DATEDIF(1998-04-20, 2025-03-23, "Y")
- **Interpretation**: Nirmal is 26 years old. This feature allows grouping learners by age (e.g., <18, 18–24, 25–34), useful for targeted program design.

b) Engagement Duration (Days)

```
df_cleaned['Engagement Duration (Days)'] = (
    df_cleaned['Apply Date'] - df_cleaned['Opportunity Start Date']
).dt.days
```

Explanation:
Calculates the number of days between a learner's application date and the opportunity start date. Positive values suggest late applications, while negative values suggest early ones.

Example:

- **Learner**: Sujita
- **Apply Date**: 2023-08-10
- **Opportunity Start Date**: 2023-08-05
- **Result**: +5 days
- **Interpretation**: Sujita applied 5 days after the program began. This may be flagged as a late application and can be used to analyze learner responsiveness.

c) Time in Opportunity (Days)

```
df_cleaned['Time in Opportunity (Days)'] = (
    df_cleaned['Opportunity End Date'] - df_cleaned['Opportunity Start Date']
).dt.days
```

Explanation:
Calculates the total number of days an opportunity lasts by subtracting the start date from the end date.

Example:

- **Opportunity**: Digital Skills Bootcamp
- **Start Date**: 2022-01-01
- **End Date**: 2022-06-30
- **Result**: 180 days
- **Interpretation**: The program runs for 6 months. This metric helps compare opportunity durations and group them as short-term or long-term learning experiences.

d) Signup to Start Delay (Days)

```
df_cleaned['Signup to Start Delay'] = (
    df_cleaned['Opportunity Start Date'] - df_cleaned['Learner SignUp DateTime']
).dt.days
```

Explanation:
Measures how many days in advance a learner signed up before the opportunity began.

Example:

- **Learner**: Kritika
- **Signup Date**: 2023-04-01
- **Opportunity Start Date**: 2023-04-20
- **Result**: 19 days

- **Interpretation**: Kritika signed up 19 days early, indicating proactive behavior. This feature can help predict learner commitment and allow for early outreach.

e) Gender_Male Encoding

```
df_cleaned['Gender_Male'] = df_cleaned['Gender'].apply(
    lambda x: 1 if x == 'Male' else (0 if x == 'Female' else -1)
)
```

Explanation:
Converts categorical gender data into a numeric format:

- Male → 1
- Female → 0
- Other/Unknown → -1

Example:

- **Learner Gender**: Female
- **Result**: 0
- **Interpretation**: This binary encoding allows easy filtering and inclusion in models for gender-based analysis while maintaining data standardization.

5. Data Validation

The dataset was thoroughly validated to ensure its **accuracy**, **logical consistency**, and **completeness** before proceeding to analysis. Several checks were conducted across dates, categories, text values, duplicates, and engineered features.

a) Date Consistency

We verified that all date-related columns followed a consistent structure and made logical sense within the context of application and program timelines.

- **Format Standardization:** All date columns (e.g., Apply Date, Opportunity Start Date, Opportunity End Date, Learner SignUp DateTime) were converted to datetime64 using pd.to_datetime(), ensuring uniform YYYY-MM-DD format.
- **Chronological Logic Checks:**

    - Opportunity Start Date ≤ Opportunity End Date
    - Learner SignUp DateTime ≤ Opportunity End Date
    - Apply Date logically aligns between SignUp and Start date
- **Result:** No inconsistencies were found after coercing and dropping invalid date rows.

b) Status Alignment

We ensured that every Status Description was correctly paired with its corresponding Status Code.

- Example:
    - "Started" consistently had Status Code = 1080
    - "Withdrawn" and "Waitlisted" had their correct, unique codes
- This was verified by checking for unexpected code-description mismatches using groupby() and value counts.

c) Text Normalization

Text columns like Institution Name, Country, and Major were standardized to improve grouping and reduce data fragmentation.

- Applied .str.strip() to remove leading/trailing spaces.
- Applied .str.title() to enforce capitalization (e.g., "nepal" → "Nepal").
- Removed symbols and special characters from Current/Intended Major using regex.
- Merged variations of the same institution: e.g., "St. Louis" and "Saint Louis University" → "Saint Louis".

d) Gender Consistency

- Coded all Gender entries as 'Male', 'Female', or 'Other'.
- Shorthand entries ('M', 'F') were mapped to full words.
- Invalid or missing values were defaulted to 'Other'.

This ensures uniform gender-based segmentation and accurate analysis during modeling.

e) Duplicate Assessment

- .drop_duplicates() was applied across all columns to eliminate exact duplicates.
- For Opportunity Id, repeated entries were manually reviewed:
    - If duplicates referred to **multiple learners per opportunity**, they were retained.
    - If the entire row was duplicated (learner + opportunity + timestamps), it was removed.

f) Feature Accuracy

We manually verified a sample of engineered features for correctness.

- **Age at Signup**: 10 learners were sampled; ages were recalculated and matched.
- **Engagement Duration**: Cross-checked date subtraction logic.
- **Opportunity Duration**: Validated length in days across different program types.
- **Signup to Start Delay**: Ensured both negative (late signup) and positive (early signup) cases existed.
- **Gender_Male encoding**: Checked mapping logic: Male → 1, Female → 0, Other → -1.

g) Completeness Check

After cleaning, we reviewed the dataset for any remaining missing values:

- All critical fields (Opportunity Start Date, Apply Date, Date of Birth, SignUp DateTime) were 100% complete.
- Non-critical fields like Current/Intended Major had ≤1 missing entry, which was either imputed or left as 'Unknown'.

Validation Outcomes

- All features were validated for logic and accuracy.
- Date relationships are consistent and free of anomalies.
- No duplicate records or illogical values remain.
- Dataset is now **fully cleaned, validated, and ready** for analysis and modeling in Week 2.

6. Issues Encountered and Resolutions

During the Week 1 cleaning process, the following data quality issues were identified and systematically addressed to improve dataset reliability:

- **Inconsistent Date Formats:**
  Columns such as Date of Birth, Apply Date, and Opportunity Start Date contained varied or malformed datetime entries. These were standardized using pd.to_datetime(errors='coerce') to ensure consistent and parseable formats across all records.
- **Missing Data in Key Fields:**
  Missing values were treated with targeted strategies. For example, Institution Name was imputed using the placeholder 'Unknown'. Rows missing critical temporal fields like Apply Date or Opportunity Start Date were excluded to ensure feature integrity. A single missing value remains in Current/Intended Major, retained as it does not impact core computations.
- **Categorical Inconsistencies:**
  Values in categorical columns such as Gender, Country, and Current/Intended Major were standardized through string normalization techniques (e.g., .str.strip(), .str.title()) and mapping functions. This ensured uniform labeling across categories for accurate grouping and encoding.

7. Conclusion

Summary of Work

In Week 1, we executed a structured data preparation process on the learner sign-up dataset, transforming raw, inconsistent data into a well-formatted, analysis-ready asset.

- **Data Cleaning:**
  We standardized date fields, resolved missing values using context-aware strategies, normalized categorical values, and removed duplicate entries. These operations ensured that the dataset maintained logical consistency and completeness.
- **Feature Engineering:**
  A total of six meaningful features were engineered:
  - Age: Age of the learner at signup, normalized.
  - Engagement Duration (Days): Gap between application and opportunity start.
  - Time in Opportunity (Days): Duration of each opportunity.

    - SignUp Month and SignUp Year: Temporal features for time-series trends.
    - Gender_Male: Numeric encoding of gender.
- **Data Validation:**
  Each engineered and cleaned field was verified through manual sampling and programmatic validation. Date logic was enforced, categorical values were standardized, and the dataset was confirmed to be free from major anomalies or critical missing data.
- **Modeling Preparation:**
  With these cleaning and enrichment steps complete, the dataset is now in optimal condition for further exploration, visualization, and predictive modeling.

Next Steps

1. **Exploratory Data Analysis (EDA):**
    - Discover patterns in engagement by age, gender, and academic major.
    - Analyze trends across signup months and opportunity durations.
    - Investigate completion vs. withdrawal status distributions.
2. **Data Visualization:**
    - Generate insightful charts (e.g., bar plots, line graphs, heatmaps).
    - Build dashboards to visualize demographic segments and learner timelines.
3. **Predictive Modeling:**
    - Train machine learning models to predict engagement levels, dropout risk, or success likelihood.
    - Utilize the engineered features as predictors for modeling learner behavior and program effectiveness.

8. Cleaned Dataset and Code file link

Cleaned Dataset Link:-

https://docs.google.com/spreadsheets/d/1Q5PHeshurrcLhz-2aNbk1GVM5VPSFicm/edit?gid=492492482#gid=492492482

Google Collab .ipynb file and pdf:-

https://drive.google.com/drive/folders/1nrWWq44-xAl2_ZmXjKWaFkBxDKvgYmaM?usp=drive_link