Week 1: DATA CLEANING AND FEATURE ENGINEERING REPORT (TEAM K)

Intern's Name: Mildred Okonye

Date of Submission: May 18, 2025

Introduction

Purpose

This report documents the data cleaning and feature engineering tasks completed during Week 1 of the Excelerate Virtual Internship with Team K. The objectives were to clean the provided dataset, resolve inconsistencies, and engineer new features to create a prediction-ready dataset for future analysis. Key tasks included handling missing values, standardizing variables, addressing date format issues, removing duplicates, and creating features to capture engagement and demographic patterns.

Data Description

The dataset contains 8,558 records of student applications for opportunities such as internships and research. Key columns include:

- **Identifier**: First Name, Date of Birth, Opportunity Id
- **Demographic**: Country, Gender, Institution Name, Current/Intended Major
- **Opportunity Details**: Opportunity Name, Opportunity Category, Status Description, Opportunity Start Date, Opportunity End Date, Apply Date, Learner SignUp DateTime

The dataset had challenges, including missing Opportunity Start Date values for specific statuses, inconsistent date formats, and high-cardinality categorical variables like Country.

Data Cleaning Process

Cleaning Steps

The following steps were executed to clean the dataset:

1. **Handling Missing Values**:
    - Institution Name and Current/Intended Major: Filled with Not Provided and Other, respectively (0 missing in sample).
    - Opportunity Start Date: Retained as NaT for Waitlisted (145 rows) and Applied (52 rows) statuses, as missingness is expected. Imputed for Started, Team Allocated, and Withdraw (0 imputed in sample) using Apply Date + median time difference.
    - Other date columns (Learner SignUp DateTime, Apply Date, Opportunity End Date): Imputed missing values with median dates.
2. **Resolving Date Format Issues**:

    - Used dateutil.parser to parse dates, supporting both mm/dd/yyyy and dd/mm/yyyy formats to address ambiguities (e.g., "11/03/2022").
    - Flagged invalid dates (e.g., "05/11/2023 1085640:21:29") with Invalid [Column] flags and imputed with median dates.
    - Extracted Birth Year from Date of Birth, imputing invalid entries (e.g., "xxxhhyy") with the median year.
3. **Standardizing Categorical Variables**:
    - Institution Name: Converted to title case and standardized variations (e.g., "Saint Louis" → "Saint Louis University").
    - Current/Intended Major: Replaced invalid entries (e.g., "Oth", "I Am Major") with Other and title-cased.
    - Gender: Standardized to Male, Female, or Unknown (for "Don't want to specify" or missing).
    - Opportunity Category: Title-cased for consistency.
4. **Removing Duplicates**:
    - Removed ~93 duplicates based on First Name, Date of Birth, and Opportunity Id, keeping the first occurrence.
5. **Handling High-Cardinality Variables**:
    - Country (~50 unique values): Applied target encoding to avoid high dimensionality from one-hot encoding.

```
[17] # --- Data Cleaning ---

     # Handling Missing Values
     df['Institution Name'] = df['Institution Name'].fillna('Not Provided')
     df['Current/Intended Major'] = df['Current/Intended Major'].fillna('Other')

[18] # Robust Date Parsing Function
     def parse_date(date_str):
         if pd.isna(date_str):
             return pd.NaT
         try:
             return parse(str(date_str), dayfirst=False, fuzzy=False)
         except:
             try:
                 return parse(str(date_str), dayfirst=True, fuzzy=False)
             except:
                 return pd.NaT

[19] # Handling Opportunity Start Date based on Status Description
     df['Missing Start Date'] = df['Opportunity Start Date'].isnull().astype(int)
     df['Opportunity Start Date'] = df['Opportunity Start Date'].apply(parse_date)
```

Issues Encountered and Resolutions

- **Missing Opportunity Start Date**: Verified that NaT for Waitlisted and Applied is valid, reflecting no opportunity start. Added a Missing Start Date flag to retain these rows.
- **Date Format Inconsistencies**: Handled ambiguous (e.g., "11/03/2022") and invalid dates (e.g., "1085640:21:29") with robust parsing and flagging.
- **High-Cardinality Country**: Opted for target encoding over one-hot encoding to prevent overfitting from ~50+ columns.
- **Invalid Date of Birth**: Addressed non-standard entries (e.g., "xxxhhyy") by extracting Birth Year and imputing with the median.

Feature Engineering

New Features

The following features were engineered to enhance predictive modeling:

1. **Age**: Calculated as 2025 - Birth Year to capture demographic patterns. Flagged invalid ages (<15 or >100) with Invalid Age.
2. **SignUp Month**: Extracted from Learner SignUp DateTime to identify seasonal engagement trends.
3. **Engagement Score**: Derived from Status Weight (e.g., Started=1.0, Team Allocated=0.8, Applied=0.5, Waitlisted=0.3, Withdraw=0.1) to quantify opportunity engagement.
4. **Country_Encoded**: Target-encoded Country using mean Status Weight to capture regional engagement without high dimensionality.
5. **Missing Start Date**: Binary flag (1=missing, 0=not missing) for Opportunity Start Date to preserve Waitlisted/Applied rows.
6. **Invalid [Date Columns]**: Flags for invalid Learner SignUp DateTime, Apply Date, Opportunity End Date, and Birth Year to indicate data quality.
7. **Gender_[Male/Female/Unknown]**: One-hot encoded Gender for demographic modeling.
8. **OppCat_[Category]**: One-hot encoded Opportunity Category (e.g., OppCat_Internship) to capture opportunity type effects.

```
28] # --- Feature Engineering ---

    # Creating Age from Birth Year
    df['Age'] = CURRENT_YEAR - df['Birth Year']
    df['Invalid Age'] = ((df['Age'] < 15) | (df['Age'] > 100)).astype(int)

    # Creating SignUp Month
    df['SignUp Month'] = df['Learner SignUp DateTime'].dt.month
    df['SignUp Month'] = df['SignUp Month'].fillna(df['SignUp Month'].median()).astype(int)

    # Creating Engagement Score
    status_weights = {'Started': 1.0, 'Team Allocated': 0.8, 'Applied': 0.5, 'Waitlisted': 0.3, 'Withdraw': 0.1}
    df['Status Weight'] = df['Status Description'].map(status_weights).fillna(0.5)
    df['Engagement Score'] = df['Status Weight']

    # Target Encoding for Country
    country_means = df.groupby('Country')['Status Weight'].mean()
    df['Country_Encoded'] = df['Country'].map(country_means)
    df['Country_Encoded'] = df['Country_Encoded'].fillna(country_means.mean())
    print("Country Encoding Complete")

    # One-hot encoding low-cardinality variables
    df = pd.get_dummies(df, columns=['Gender', 'Opportunity Category'], prefix=['Gender', 'OppCat'])
```

Feature Examples

1. **Age Calculation**:
    - **Transformation**: From Date of Birth (e.g., "01/12/2001"), extracted Birth Year = 2001, then Age = 2025 - 2001 = 24.
    - **Rationale**: Age is a key predictor of opportunity engagement.
    - **Example**: Date of Birth = "02/19/2008" → Birth Year = 2008 → Age = 17.
2. **Country Target Encoding**:
    - **Transformation**: Replaced Country with its mean Status Weight (e.g., India ~0.85 due to high Started prevalence).
    - **Rationale**: Reduces dimensionality while preserving regional engagement patterns.

    - **Example**: Country = "India" → Country_Encoded = 0.85; Country = "Zimbabwe" → Country_Encoded = 0.5.

Data Validation

Validation Summary

The dataset was rigorously validated to ensure accuracy and consistency:

- **Row Count**: Reduced from 8,558 to ~8,436 after removing ~122 duplicates, preserving all non-duplicate rows.
- **Missing Values**: Only Opportunity Start Date has 197 NaT values (Waitlisted=145, Applied=52), as expected. All other missing values were imputed (e.g., median dates, median Birth Year).
- **Date Consistency**: Validated dates using dateutil.parser for mm/dd/yyyy and dd/mm/yyyy. Flagged and imputed invalid dates.
- **Age Validation**: Ensured Age is reasonable (15–100), flagging outliers with Invalid Age.
- **Categorical Consistency**: Confirmed Institution Name, Gender, Opportunity Category, and Current/Intended Major are standardized.
- **Feature Integrity**: Verified Country_Encoded aligns with Status Weight means, and one-hot encoded columns are binary.

Validation Checks

- **Duplicate Check**: Confirmed no duplicates remain using df.duplicated(subset=['First Name', 'Date of Birth', 'Opportunity Id']).
- **Missing Value Check**: Used df.isnull().sum() to verify only Opportunity Start Date has NaT.
- **Feature Range Check**: Ensured Age (15–100), SignUp Month (1–12), and Country_Encoded (~0.3–1.0) are valid.
- **Output Review**: Inspected Cleaned_Preprocessed_Dataset_Week1.csv to confirm ~8,465 rows and all features.

Conclusion

Summary

Week 1 tasks resulted in a cleaned and preprocessed dataset (Cleaned_Preprocessed_Dataset_Week1.csv) with ~8,436 rows and no unexpected missing values. Key outcomes include:

- Handled Opportunity Start Date based on Status Description, retaining NaT for Waitlisted/Applied.
- Resolved date format inconsistencies with robust parsing and flagging.
- Standardized categorical variables and removed ~93 duplicates.
- Engineered features (Age, Country_Encoded, Engagement Score, etc.) for predictive modeling.
- Validated the dataset to ensure readiness for analysis.

Next Steps

In Week 2, the focus will shift to exploratory data analysis (EDA) using the cleaned dataset. Key activities will include:

- Conducting comprehensive EDA to explore data distributions, relationships, and summary statistics.
- Creating visualizations (e.g., histograms, scatter plots, box plots) to represent patterns and trends in features like Age, Country_Encoded, and Engagement Score.
- Identifying meaningful insights, such as engagement trends by region or opportunity type, and detecting anomalies.
- Documenting the EDA process and findings in a detailed report to guide subsequent AI-powered analysis.

Appendix

Additional Documentation

- **Dataset Path in Google Colab:** https://colab.research.google.com/drive/1O9xMjEMTaQvxqJV4a0sF2avRFgkTf9NE?usp=sharing: