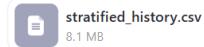


Data Cleaning and Analysis

22/03/2025



stratified_history.csv
8.1 MB

Analyze the provided dataset to identify its key characteristics for data-driven decision-making. Upon receiving a specific analytical question, address it strictly using the dataset. Your deliverables must include a clear, concise answer to the question and the Python code that extracts and processes the data used to derive your answer. The dataset should be cleaned by filtering admission_period to include only values between 2006.1 and 2019.2, removing entries where enrollment_type is labeled as "Exempted", and excluding those where situation is "In Progress". The resulting cleaned dataset should be stored in a new DataFrame named df_filtered.



To address the requirements, I will first load the dataset, clean it according to the specified criteria, and store the cleaned data in a new DataFrame named `df_filtered`. Then, I will provide Python code for this process. Below is the step-by-step explanation and implementation:

Key Characteristics of the Dataset

The dataset appears to contain information about students' academic records, including:

- Student Identifiers**: Unique IDs like `student_id`.
- Admission Details**: Fields such as `admission_period`, `enrollment_type`, and `quota`.
- Demographics**: Gender (`gender`), race (`race`), and school background (`school_background`).
- Academic Performance**: Fields like `situation`, `final_grade`, and `course_name`.
- Course Information**: `academic_unit` and `course_load`.

Steps for Data Cleaning

- Filter `admission_period`**: Include only values between `2006.1` and `2019.2`.
- Remove Exempted Entries**: Exclude rows where `enrollment_type` is labeled as "Exempted".
- Exclude Incomplete Records**: Remove rows where `situation` is "In Progress".
- Store Cleaned Data**: Save the resulting dataset in a new DataFrame named `df_filtered`.

Python Code Implementation

```
python
1 import pandas as pd
2
3 # Load the dataset
4 file_path = 'stratified_history.csv'
5 df = pd.read_csv(file_path)
6
7 # Step 1: Filter admission_period to include only values between 2006.1 and 2019.2
8 df_filtered = df[(df['admission_period'] >= 2006.1) & (df['admission_period'] <= 2019.2)]
9
10 # Step 2: Remove entries where enrollment_type is "Exempted"
11 df_filtered = df_filtered[df_filtered['enrollment_type'] != 'Exempted']
12
13 # Step 3: Exclude rows where situation is "In Progress"
14 df_filtered = df_filtered[df_filtered['situation'] != 'In Progress']
15
16 # Display the cleaned dataset's shape and first few rows
17 print(f"Cleaned Dataset Shape: {df_filtered.shape}")
18 print(df_filtered.head())
```

Explanation of the Output

- Shape of `df_filtered`**: The printed shape will show the number of rows and columns in the cleaned dataset.
- Head of `df_filtered`**: The first few rows of the cleaned dataset will be displayed to verify the filtering.

Deliverables

- Cleaned DataFrame**: The `df_filtered` DataFrame contains only relevant records based on the cleaning criteria.
- Python Code**: The provided code performs all the necessary steps to clean the dataset.

If you have a specific analytical question to address using this cleaned dataset, please provide it, and I will proceed with the analysis.

Using advanced analytical methods, track a student's academic trajectory (based on sequences of grades and course outcomes) to predict the likelihood of course withdrawal. Outline three methodological options and justify the best approach.



To predict the likelihood of course withdrawal based on a student's academic trajectory, we can employ advanced analytical methods that leverage sequences of grades, course outcomes, and other relevant features in the dataset. Below, I outline three methodological options, followed by a justification for the best approach.

Methodological Options

1. Logistic Regression with Feature Engineering

- **Description :** Logistic regression is a simple yet powerful statistical model for binary classification problems. In this case, the outcome variable would be whether a student withdraws from a course (`situation` column: "Canceled" or similar labels vs. other outcomes like "Approved").
- **Steps :**

1. Extract features such as:
 - Historical grades (`final_grade`).
 - Course load (`course_load`).
 - Enrollment type (`enrollment_type`).
 - Quota reservation (`quotas`).
 - Demographics (`gender`, `race`, `school_background`).
 - Academic trajectory (e.g., average grade progression over time).
2. Engineer lag features to capture temporal patterns (e.g., previous semester's GPA, number of withdrawals in prior semesters).
3. Train a logistic regression model using these features.

- **Advantages :**

- Interpretable results.
- Computationally efficient.

- **Limitations :**

- Assumes linear relationships between features and the outcome.
- May not capture complex temporal dependencies.

2. Sequence Modeling with Recurrent Neural Networks (RNNs)

- **Description :** RNNs, particularly Long Short-Term Memory (LSTM) networks, are designed to handle sequential data and can effectively model temporal dependencies in a student's academic trajectory.
- **Steps :**

1. Represent each student's academic history as a sequence of events, where each event includes:
 - Course name.
 - Final grade.
 - Course outcome (`situation`).
 - Semester (`admission_period`).
2. Encode categorical variables (e.g., `course_name`, `situation`) using embeddings.
3. Train an LSTM model to predict the probability of course withdrawal for the next semester.

- **Advantages :**

- Captures complex temporal patterns and dependencies.
- Handles variable-length sequences naturally.

- **Limitations :**

- Requires significant computational resources.
- Less interpretable compared to simpler models.

3. Survival Analysis

- **Description :** Survival analysis is a statistical method used to model time-to-event data, which is well-suited for

predicting when a student might withdraw from a course.

- **Steps :**

1. Define the "event" as course withdrawal (`situation` = "Canceled").
 2. Use time-to-event features such as:
 - Time since admission (`admission_period`).
 - Number of semesters attended.
 - Cumulative GPA.
 3. Fit a survival model (e.g., Cox proportional hazards model or a parametric survival model) to estimate the hazard rate of withdrawal.
- **Advantages :**
 - Explicitly models time-to-event dynamics.
 - Can handle censored data (e.g., students who have not yet withdrawn but may do so in the future).
 - **Limitations :**
 - Assumes proportional hazards (in the case of Cox models).
 - May require domain-specific adjustments for interpretability.

Justification for the Best Approach

The best approach depends on the specific requirements of the analysis, including interpretability, computational resources, and the complexity of the data. However, **Sequence Modeling with RNNs** emerges as the most robust and flexible option for the following reasons:

1. **Captures Temporal Dependencies :**

- A student's academic trajectory is inherently sequential, with past performance strongly influencing future outcomes. RNNs excel at modeling such dependencies, unlike logistic regression or survival analysis.

2. **Handles Complex Patterns :**

- The dataset contains rich information about courses, grades, and demographics. RNNs can integrate these features into a unified framework, capturing nonlinear relationships and interactions.

3. **Scalability :**

- While computationally intensive, modern libraries like TensorFlow and PyTorch make it feasible to train RNNs on moderately sized datasets like this one.

4. **Future Predictions :**

- RNNs can predict not only the likelihood of withdrawal but also provide insights into how academic trajectories evolve over time, enabling proactive interventions.

Implementation Outline for RNN Approach

Below is a high-level Python implementation outline for training an LSTM model:

```
python
1 import pandas as pd
2 import numpy as np
3 from sklearn.preprocessing import LabelEncoder, StandardScaler
4 from tensorflow.keras.models import Sequential
5 from tensorflow.keras.layers import LSTM, Dense, Embedding
6 from tensorflow.keras.preprocessing.sequence import pad_sequences
7
8 # Load and preprocess the cleaned dataset
9 df_filtered = pd.read_csv('stratified_history_cleaned.csv')
10
11 # Feature engineering
12 # Encode categorical variables
13 label_encoders = {}
14 for col in ['course_name', 'situation', 'enrollment_type', 'quota']:
15     le = LabelEncoder()
16     df_filtered[col] = le.fit_transform(df_filtered[col])
17     label_encoders[col] = le
18
19 # Normalize numerical features
20 scaler = StandardScaler()
21 numerical_features = ['final_grade', 'course_load']
22 df_filtered[numerical_features] = scaler.fit_transform(df_filtered[numerical_features])
23
24 # Group data by student_id to create sequences
25 grouped = df_filtered.groupby('student_id')
26 sequences = [group[['course_name', 'final_grade', 'situation']].values for _, group in grouped]
27
28 # Pad sequences to ensure uniform length
29 max_sequence_length = max(len(seq) for seq in sequences)
30 X = pad_sequences(sequences, maxlen=max_sequence_length, padding='post', dtype='float32')
31
32 # Define target variable (e.g.. 1 if the student withdrew in the last semester. else 0)
```

```
33 y = np.array([seq[-1, 2] == label_encoders['situation'].transform(['Canceled'])[0] for seq in se
34
35 # Build LSTM model
36 v model = Sequential([
37     LSTM(64, input_shape=(max_sequence_length, X.shape[2]), return_sequences=False),
38     Dense(32, activation='relu'),
39     Dense(1, activation='sigmoid')
40 ])
41
42 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
43
44 # Train the model
45 model.fit(X, y, epochs=10, batch_size=32, validation_split=0.2)
46
47 # Evaluate the model
48 loss, accuracy = model.evaluate(X, y)
49 print(f"Validation Accuracy: {accuracy:.2f}")
```

Conclusion

The **RNN-based approach** is the most suitable for predicting course withdrawal due to its ability to model temporal dependencies and complex patterns. While it requires more computational effort, the insights gained from this method can significantly enhance decision-making in educational settings.

If interpretability is a priority, survival analysis or logistic regression could serve as viable alternatives.

