

Data Cleaning and Filtering

22/03/2025

stratified_history.csv
0.1 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.

qwen-max-latest

To provide an analysis based on your request, I'll first outline the steps to clean and filter the dataset according to the criteria specified. Then, I will present sample Python code that performs these tasks and prepares the cleaned DataFrame `df_filtered`. Afterward, you can ask a specific analytical question, and I'll address it using the filtered dataset.

Step-by-Step Plan:

- Filter by Admission Period** : Retain only rows where the `admission_period` is between 2006.1 and 2019.2 (inclusive).
- Exclude Exempted Enrollments** : Remove entries where `enrollment_type` is labeled as "Exempted".
- Exclude In-Progress Situations** : Exclude rows where `situation` is marked as "In Progress".
- Store Cleaned Data** : Save the resulting data into a new DataFrame called `df_filtered`.

Here's how this can be implemented in Python:

```
python
1 import pandas as pd
2
3 # Load the dataset from the provided CSV file
4 file_path = 'stratified_history.csv'
5 df = pd.read_csv(file_path)
6
7 # Displaying basic info about the dataset
8 print("Initial Dataset Info:")
9 print(df.info())
10
11 # Cleaning and filtering the dataset
12 df_filtered = df[
13     (df['admission_period'] >= 2006.1) &
14     (df['admission_period'] <= 2019.2) &
15     (df['enrollment_type'] != 'Exempted') &
16     (df['situation'] != 'In Progress')
17 ]
18
19 # Displaying the cleaned dataset information
20 print("\nFiltered Dataset Info:")
21 print(df_filtered.info())
22
23 # Saving the filtered DataFrame to a new CSV file if needed
24 df_filtered.to_csv('filtered_dataset.csv', index=False)
```

Explanation of the Code:

- Loading the Data** : The dataset is loaded using `pd.read_csv()`.
- Filtering Conditions** :
 - `(df['admission_period'] >= 2006.1) & (df['admission_period'] <= 2019.2)` ensures we only include admission periods within the specified range.
 - `(df['enrollment_type'] != 'Exempted')` removes rows where the enrollment type is "Exempted".
 - `(df['situation'] != 'In Progress')` excludes rows where the situation is "In Progress".
- New DataFrame** : The filtered data is stored in `df_filtered`.
- Output** : Basic information about the original and filtered datasets is printed to observe changes after filtering.

Next Steps:

Please provide a specific analytical question or task you'd like addressed using the cleaned dataset (`df_filtered`). For example:

- What is the distribution of students by gender across different admission periods?
- How many students dropped out versus graduated in each year?
- What is the average grade for courses offered by the Academic Unit of Systems and Computing?

Once you specify the question, I'll use the filtered dataset to derive the answer and provide the corresponding Python code.

Using advanced analysis of grade sequences and academic records, identify key periods in a student's academic trajectory where the dropout risk is significantly elevated. Propose three analytical approaches and select the optimal method.

qwen-max-latest

To identify key periods in a student's academic trajectory where the dropout risk is significantly elevated, we can employ advanced analytical approaches that leverage the cleaned dataset (`df_filtered`) to uncover patterns and trends. Below are three analytical approaches, followed by a recommendation for the optimal method:

1. Survival Analysis

Survival analysis is a statistical method used to model time-to-event data, such as the time until a student drops out. It allows us to estimate the probability of dropout at different points in a student's academic trajectory.

Steps:

- Define the Event** : Dropout status (`situation == 'Dropout'`) is the event of interest.
- Time Variable** : Use the difference between `admission_period` and `last_enrollment_period` (or equivalent fields) as the time variable.
- Covariates** : Include features like grade sequences, enrollment type, demographic information, and academic unit.
- Kaplan-Meier Estimator** : Visualize dropout probabilities over time.
- Cox Proportional Hazards Model** : Identify factors that significantly influence dropout risk.

Python Code:

```
python
1 from lifelines import KaplanMeierFitter, CoxPHFitter
2
3 # Calculate time-to-event (time until dropout or last enrollment)
4 df_filtered['time_to_event'] = df_filtered['last_enrollment_period'] - df_filtered['admission_pe
5 df_filtered['event'] = df_filtered['situation'].apply(lambda x: 1 if x == 'Dropout' else 0)
6
```

```
7 # Kaplan-Meier Survival Curve
8 kmf = KaplanMeierFitter()
9 kmf.fit(durations=df_filtered['time_to_event'], event_observed=df_filtered['event'])
10 kmf.plot_survival_function()
11
12 # Cox Proportional Hazards Model
13 cph = CoxPHFitter()
14 cph.fit(df_filtered[['time_to_event', 'event', 'grade', 'enrollment_type', 'gender', 'race']], c
15 cph.print_summary()
```

2. Sequence Pattern Mining

Sequence pattern mining identifies common patterns in grade sequences or academic trajectories that precede dropout events.

Steps:

- **Grade Sequences** : Extract sequences of grades for each student across semesters.
- **Pattern Discovery** : Use algorithms like PrefixSpan or GSP to find frequent grade patterns associated with dropouts.
- **Visualization** : Highlight critical periods where specific grade patterns (e.g., consecutive failures) are linked to higher dropout risk.

Python Code:

```
python
1 from prefixspan import PrefixSpan
2
3 # Create grade sequences for each student
4 sequences = df_filtered.groupby('student_id')['grade'].apply(list).tolist()
5
6 # Apply PrefixSpan to find frequent patterns
7 ps = PrefixSpan(sequences)
8 patterns = ps.frequent(minsup=0.1) # Adjust minsup as needed
9 print(patterns)
```

3. Machine Learning Classification

A supervised machine learning approach can predict dropout risk based on historical data. This method identifies critical periods by analyzing feature importance and decision boundaries.

Steps:

- **Target Variable** : Dropout status (`situation == 'Dropout'`).
- **Features** : Include admission period, grade history, demographic information, and academic performance metrics.
- **Model Training** : Train models like Random Forest, Gradient Boosting, or Logistic Regression.
- **Feature Importance** : Analyze which features (e.g., grades in specific semesters) contribute most to dropout risk.
- **SHAP Values** : Use SHAP (SHapley Additive exPlanations) to interpret model predictions and identify critical periods.

Python Code:

```
python
1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn.model_selection import train_test_split
3 import shap
4
5 # Prepare features and target
6 X = df_filtered[['admission_period', 'grade', 'enrollment_type', 'gender', 'race', 'time_to_event']]
7 y = df_filtered['event']
8
9 # Train-test split
10 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
11
12 # Train Random Forest Classifier
13 clf = RandomForestClassifier(random_state=42)
14 clf.fit(X_train, y_train)
15
16 # Feature Importance
17 importances = clf.feature_importances_
18 print(importances)
19
20 # SHAP Analysis
21 explainer = shap.TreeExplainer(clf)
22 shap_values = explainer.shap_values(X_test)
23 shap.summary_plot(shap_values, X_test)
```

Optimal Method Recommendation

The **optimal method** depends on the specific goals and available resources, but **Survival Analysis** is likely the best choice for this task because:

1. **Temporal Insights** : It explicitly models time-to-event data, providing clear insights into when dropout risk is highest.
2. **Interpretability** : The Kaplan-Meier curve and Cox model offer interpretable results about dropout probabilities and contributing factors.
3. **Flexibility** : It accommodates censored data (students who haven't dropped out yet) and allows for the inclusion of covariates.

While sequence pattern mining and machine learning classification are powerful, they require more preprocessing and may not provide as direct an answer to the question of "when" dropout risk is elevated.

Conclusion

Using survival analysis, we can pinpoint key periods (e.g., early semesters or after a significant grade drop) where dropout risk is highest. This approach provides actionable insights for targeted interventions, such as academic advising or support programs, during these critical periods.

If you'd like, I can refine the analysis further or focus on implementing one of these methods in greater detail. Let me know!

