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.

qwen-max-latest

Analysis and Data Cleaning

The dataset appears to contain historical academic records of students, including their admission details, course enrollments, performance metrics, and outcomes. To ensure the dataset is suitable for data-driven decision-making, we will clean it according to the specified criteria:

1. **Filter** `admission_period` : Include only values between `2006.1` and `2019.2` .
2. **Remove entries with** `enrollment_type` **labeled as "Exempted"** .
3. **Exclude entries where** `situation` **is "In Progress"** .

After cleaning, the resulting dataset will be stored in a new DataFrame named `df_filtered` .

Python Code for Data Cleaning

Below is the Python code to clean the dataset and store the cleaned data in `df_filtered` :

```python
import pandas as pd

# Load the dataset
df = pd.read_csv('stratified_history.csv')

# Step 1: Filter admission_period to include only values between 2006.1 and 2019.2
df_filtered = df[(df['admission_period'] >= 2006.1) & (df['admission_period'] <= 2019.2)]

# Step 2: Remove entries where enrollment_type is labeled as "Exempted"
df_filtered = df_filtered[df_filtered['enrollment_type'] != 'Exempted']

# Step 3: Exclude entries where situation is "In Progress"
df_filtered = df_filtered[df_filtered['situation'] != 'In Progress']

# Display the first few rows of the cleaned dataset
print(df_filtered.head())

# Optionally, save the cleaned dataset to a new CSV file
df_filtered.to_csv('cleaned_dataset.csv', index=False)
```

Explanation of the Code

1. **Loading the Dataset** : The dataset is loaded into a Pandas DataFrame using `pd.read_csv()` .
2. **Filtering Admission Period** : The `admission_period` column is filtered to include only values within the range `[2006.1, 2019.2]` using a logical condition.
3. **Removing Exempted Entries** : Rows where `enrollment_type` is "Exempted" are removed using a negation condition (`!=`).
4. **Excluding In Progress Situations** : Rows where `situation` is "In Progress" are excluded similarly.
5. **Saving the Cleaned Dataset** : The cleaned dataset is optionally saved to a new CSV file for future use.

Key Characteristics of the Cleaned Dataset

After cleaning, the dataset (`df_filtered`) will have the following key characteristics:

- **Timeframe** : Only records from `2006.1` to `2019.2` are included.
- **Enrollment Type** : No records with "Exempted" enrollment type are present.
- **Situation** : All records have a resolved situation (i.e., no "In Progress" cases).

This cleaned dataset can now be used for further analysis or to address specific analytical questions.

Next Steps

If you have a specific analytical question, please provide it, and I will address it using the cleaned dataset (`df_filtered`).

Analyze the relationship between extra-curricular participation (with no failures or dropouts) and academic success using correlation tests (Pearson, Spearman), regression models, and propensity score matching. Compare the effectiveness of each approach.

qwen-max-latest

To analyze the relationship between **extra-curricular participation** (with no failures or dropouts) and **academic success** , we will use the cleaned dataset (`df_filtered`) and apply three analytical approaches: **correlation tests** , **regression models** , and **propensity score matching (PSM)** . Each method has its strengths and limitations, which we will compare at the end.

Step 1: Define Variables

- **Independent Variable (Exposure)** : Extra-curricular participation (`extracurricular` column, e.g., "Extracurricular").
- **Dependent Variable (Outcome)** : Academic success, measured by:
  - GPA (`gpa` column).
  - Course completion rate (`course_completion_rate` column).
  - Dropout status (`situation` column, binary variable: `Dropout` vs. `Graduate`).

We will filter the dataset to include only rows where there are **no failures** (`failures == 0`) and **no dropouts** (`situation != "Dropout"`).

Step 2: Data Preparation

Below is the Python code to prepare the dataset for analysis:

```python
import pandas as pd
import numpy as np
from scipy.stats import pearsonr, spearmanr
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.neighbors import NearestNeighbors

# Load the cleaned dataset
df = pd.read_csv('cleaned_dataset.csv')

# Filter for no failures and no dropouts
df_success = df[(df['failures'] == 0) & (df['situation'] != 'Dropout')]

# Encode categorical variables
df_success['extracurricular'] = df_success['extracurricular'].apply(lambda x: 1 if x == 'Extracu
df_success['situation'] = df_success['situation'].apply(lambda x: 1 if x == 'Graduate' else 0)

# Drop rows with missing values
df_success = df_success.dropna(subset=['gpa', 'course_completion_rate', 'extracurricular'])

# Display the first few rows of the filtered dataset
print(df_success.head())
```

Step 3: Correlation Tests

We will calculate **Pearson** and **Spearman** correlation coefficients between extra-curricular participation and academic success metrics.

Code:

```python
# Pearson correlation
pearson_gpa = pearsonr(df_success['extracurricular'], df_success['gpa'])
pearson_completion = pearsonr(df_success['extracurricular'], df_success['course_completion_rate'

# Spearman correlation
spearman_gpa = spearmanr(df_success['extracurricular'], df_success['gpa'])
spearman_completion = spearmanr(df_success['extracurricular'], df_success['course_completion_rat

print("Pearson Correlation (GPA):", pearson_gpa)
print("Pearson Correlation (Completion Rate):", pearson_completion)
print("Spearman Correlation (GPA):", spearman_gpa)
print("Spearman Correlation (Completion Rate):", spearman_completion)
```

Interpretation:

- **Pearson** : Measures linear relationships.
- **Spearman** : Measures monotonic relationships (non-linear but consistently increasing/decreasing).

Step 4: Regression Models

We will use two regression models:

1. **Linear Regression** : Predict GPA based on extra-curricular participation.
2. **Logistic Regression** : Predict graduation status (`situation`) based on extra-curricular participation.

Code:

```python
# Linear Regression for GPA
X_gpa = sm.add_constant(df_success['extracurricular'])  # Add intercept
y_gpa = df_success['gpa']
model_gpa = sm.OLS(y_gpa, X_gpa).fit()
print(model_gpa.summary())

# Logistic Regression for Graduation Status
X_grad = df_success[['extracurricular']]
y_grad = df_success['situation']
model_grad = LogisticRegression()
model_grad.fit(X_grad, y_grad)
grad_prob = model_grad.predict_proba(X_grad)[:, 1]
auc_score = roc_auc_score(y_grad, grad_prob)
print("AUC Score for Graduation Prediction:", auc_score)
```

Interpretation:

- **Linear Regression** : Coefficients indicate the effect of extra-curricular participation on GPA.
- **Logistic Regression** : AUC score measures the model's ability to distinguish graduates from non-graduates.

Step 5: Propensity Score Matching (PSM)

PSM balances the treatment group (extra-curricular participants) and control group (non-participants) based on covariates like age, gender, and prior academic performance.

Code:

```python
```

```
# Propensity Score Calculation
X_psm = df_success[['age', 'gender', 'admission_period', 'enrollment_type', 'failures', 'gpa']]
y_psm = df_success['extracurricular']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_psm)

# Train logistic regression for propensity scores
ps_model = LogisticRegression()
ps_model.fit(X_scaled, y_psm)
propensity_scores = ps_model.predict_proba(X_scaled)[:, 1]

# Match participants and non-participants
participants = df_success[df_success['extracurricular'] == 1]
non_participants = df_success[df_success['extracurricular'] == 0]

knn = NearestNeighbors(n_neighbors=1)
knn.fit(non_participants[['age', 'gender', 'admission_period', 'enrollment_type', 'failures', 'g
distances, indices = knn.kneighbors(participants[['age', 'gender', 'admission_period', 'enrollme

matched_non_participants = non_participants.iloc[indices.flatten()]
matched_df = pd.concat([participants, matched_non_participants])

# Compare outcomes
mean_gpa_treated = matched_df[matched_df['extracurricular'] == 1]['gpa'].mean()
mean_gpa_control = matched_df[matched_df['extracurricular'] == 0]['gpa'].mean()

print("Mean GPA (Treated):", mean_gpa_treated)
print("Mean GPA (Control):", mean_gpa_control)
```

Interpretation:

- PSM reduces selection bias by creating comparable groups.
- Compare mean GPA or graduation rates between matched groups.

Step 6: Compare Approaches

METHOD	STRENGTHS	LIMITATIONS
Correlation Tests	Simple, quick to compute.	Only measures association; cannot infer causality.
Regression Models	Can control for confounders; predicts outcomes.	Assumes linearity/logistic relationship; sensitive to omitted variable bias.
Propensity Score Matching	Balances groups; reduces selection bias.	Requires large sample size; assumes no unmeasured confounders.

Final Answer

The most effective approach depends on the research question:

- Use **correlation tests** for exploratory analysis.
- Use **regression models** for prediction and controlling confounders.
- Use **PSM** for causal inference when groups need balancing.

For this dataset, **PSM** provides the most robust causal estimate of the effect of extra-curricular participation on academic success.

Boxed Answer :

$$\boxed{\text{Propensity Score Matching is the most effective approach for causal inference.}}$$