Assignments (Py03)

Problem 1: Predicting Hospital Readmission Rates

Hard

Scenario: A hospital wants to reduce its 30-day readmission rates, a key quality measure in healthcare. The hospital has historical data on patients, including demographics, diagnoses, treatments, discharge instructions, and whether or not they were readmitted within 30 days.

Objective: Analyze the data to identify patterns and predictors of readmission.

Pandas Tasks:

1. **Data Cleaning:** Use Pandas to clean the dataset by handling missing values, removing duplicates, and standardizing the format of the data.
2. **Feature Engineering:** Create new features, such as the length of stay, number of comorbidities, or discharge destination, that could be relevant to predicting readmission.
3. **Data Analysis:** Perform exploratory data analysis (EDA) to identify correlations between patient characteristics and readmission rates. Visualize trends using Pandas' plotting functions.

```python
# Example usage (dummy data)
# Load the data
data = pd.read_csv('fictitious_hospital_data.csv')
# Data cleaning
data_cleaned = clean_data(data)
# Feature engineering
data_fe = engineer_features(data_cleaned)
# Data analysis
analyze_data(data_fe)
```

If you stop here, and code this function, by yourself, you will learn a lot. (HARD)

Medium

1. **Data Cleaning:**
    - Check and handle missing values and duplicates.
        - Check for missing values in each column, by using the `isnull()` method followed by `sum()`.
    - Standardize the format of categorical data (e.g., convert to lowercase).
        - Convert the `Gender`, `Primary_Diagnosis`, and `Discharge_Destination` columns to lowercase using the `str.lower()` method.

2. **Feature Engineering:**

- Create new features like `Length_of_Stay_Category`, `Age_Group`, and `Comorbidity_Burden` based on existing columns.
    - Create a new feature `Length_of_Stay_Category` by binning the `Length_of_Stay` column into categories like 'Short', 'Medium', 'Long', and 'Very Long'.
    - Create a new feature `Age_Group` by binning the `Age` column into categories like 'Young', 'Middle-aged', 'Senior', and 'Elderly'.
    - Create a new feature `Comorbidity_Burden` by binning the `Num_Comorbidities` column into categories like 'None', 'Low', 'Medium', and 'High'.

3. **Data Analysis:**

    - Calculate the readmission rate by different categorical features such as gender, primary diagnosis, length of stay, age group, discharge destination, and comorbidity burden.
        - Group the data by each categorical feature and calculate the mean of the `Readmitted_Within_30_Days` column.
    - Generate summary statistics for the cleaned data.
        - Calculate descriptive statistics like mean, standard deviation, min, max, etc., for numerical columns using the `describe()` method.
    - Calculate the correlation matrix to understand relationships between numerical features.
        - Use the `corr()` method to calculate the correlation between numerical columns.

If you stop here, and code this function, with the help of the instructions above, you still learn a lot. (MEDIUM)

Easy

Fill the TODOs in the code below to complete the function.

Here's some dummy code to solve Problem 1 using Pandas, covering data cleaning, feature engineering, and data analysis tasks:

```
import pandas as pd
import numpy as np

# Load the data
data = #TODO, read the data from 'fictitious_hospital_data.csv'

# 1. Data Cleaning
# -----------------

# Check for missing values
missing_values =  #TODO check for missing values in each column
print("Missing values in each column:")
print(missing_values)

# Drop rows with missing values (if any)
```

```python
data_cleaned =  #TODO drop rows with missing values

# Check for duplicate rows
duplicate_rows =  #TODO, check for duplicate rows
# print(f"Number of duplicate rows: {duplicate_rows}")

# Drop duplicate rows (if any)
data_cleaned = #TODO

# Standardize categorical data (convert to lower case or consistent
format)
data_cleaned['Gender'] = #TODO
data_cleaned['Primary_Diagnosis'] = #TODO
data_cleaned['Discharge_Destination'] = #TODO

# 2. Feature Engineering
# ----------------------

# Create a new feature: Length of Stay Category
bins = [0, 3, 7, 14, np.inf]
labels = ['Short', 'Medium', 'Long', 'Very Long']
data_cleaned['Length_of_Stay_Category'] =  #TODO

# Create a new feature: Age Group
age_bins = [0, 30, 50, 70, np.inf]
age_labels = ['Young', 'Middle-aged', 'Senior', 'Elderly']
data_cleaned['Age_Group'] =  #TODO

# Create a new feature: Comorbidity Burden
data_cleaned['Comorbidity_Burden'] =  #TODO

# 3. Data Analysis
# ----------------

# Analyzing Readmission Rates by Various Factors

# Readmission rate by Gender
readmission_by_gender =  #TODO
print("Readmission rate by Gender:")
print(readmission_by_gender)

# Readmission rate by Primary Diagnosis
readmission_by_diagnosis = #TODO
print("\nReadmission rate by Primary Diagnosis:")
print(readmission_by_diagnosis)

# Readmission rate by Length of Stay Category
readmission_by_los =  #TODO
print("\nReadmission rate by Length of Stay Category:")
print(readmission_by_los)

# Readmission rate by Age Group
readmission_by_age_group = #TODO
```

```
print("\nReadmission rate by Age Group:")
print(readmission_by_age_group)

# Readmission rate by Discharge Destination
readmission_by_discharge = #TODO
print("\nReadmission rate by Discharge Destination:")
print(readmission_by_discharge)

# Readmission rate by Comorbidity Burden
readmission_by_comorbidity = data_cleaned.groupby('Comorbidity_Burden')
['Readmitted_Within_30_Days'].mean()
print("\nReadmission rate by Comorbidity Burden:")
print(readmission_by_comorbidity)

# Summary Statistics for Data
summary_stats = #TODO
print("\nSummary statistics of cleaned data:")
print(summary_stats)

# Count of patients in each group (e.g., Age Group, Length of Stay
Category)
age_group_counts = #TODO
length_of_stay_counts =
data_cleaned['Length_of_Stay_Category'].value_counts()

print("\nCount of patients by Age Group:")
print(age_group_counts)

print("\nCount of patients by Length of Stay Category:")
print(length_of_stay_counts)

# Correlation Matrix
# Convert categorical features to numerical using one-hot encoding
data_cleaned = pd.get_dummies(data_cleaned)
# Correlation matrix for numerical features
correlation_matrix = data_cleaned.corr()
print("\nCorrelation matrix: of values abs(corr) > 0.8")
# format the output only show the values that are greater than 0.8 in
absolute value
# Output: feature 1 vs feature 2 : correlation value and only show the
values that are greater than 0.8 in absolute value
print(correlation_matrix[(correlation_matrix.abs() > 0.8) &
(correlation_matrix < 1)].stack().dropna())
```

Code Explanation

1. **Data Cleaning:**
    - Check and handle missing values and duplicates.
    - Standardize the format of categorical data.

2. **Feature Engineering:**
    - Create new features like `Length_of_Stay_Category`, `Age_Group`, and `Comorbidity_Burden` based on existing columns.
3. **Data Analysis:**
    - Calculate the readmission rate by different categorical features such as gender, primary diagnosis, length of stay, age group, discharge destination, and comorbidity burden.
    - Generate summary statistics for the cleaned data.
    - Calculate the correlation matrix to understand relationships between numerical features.

This code is designed to provide insight into patterns in the data and help identify factors associated with readmission without delving into model creation or visualization.

You finished this problem, if you complete the code above, and test it with the example usage. (EASY)

Problem 2: Analyzing Medication Adherence in Chronic Disease Patients

Hard

Scenario: A clinic is concerned about medication adherence among its chronic disease patients (e.g., diabetes, hypertension). Poor adherence can lead to worsening health outcomes and increased healthcare costs. The clinic has data on patient visits, prescriptions, and self-reported adherence levels.

Objective: Identify patients at high risk of non-adherence and the factors associated with poor adherence. Create a system to flag patients who may need additional support.

Pandas Tasks:

1. **Data Merging:** Combine multiple datasets, such as tables from patients and visits with prescription records, into a single DataFrame for analysis.
2. **Adherence Calculation:** Calculate medication adherence rates for each patient by comparing the number of prescribed doses to the number of doses reported as taken.
3. **Risk Stratification:** Group patients by adherence levels (e.g., high, medium, low) and analyze the characteristics of each group (e.g., age, number of medications, comorbidities).

```
# Example usage (dummy data)
# Load the datasets
patients_df = pd.read_csv('fictitious_patients.csv')
visits_df = pd.read_csv('fictitious_visits.csv')

# 1. Data Merging
# Combine patients and visits data into a single DataFrame
# 2. Adherence Calculation
# Calculate medication adherence rates
# 3. Risk Stratification
```

```python
# Classify patients into risk categories based on adherence rate
# Analyze characteristics of each group

# Output the results
print(grouped_df)

# Example: Flag patients with low adherence for additional support
print("Patients at high risk of non-adherence:")
print(low_adherence_patients[['Patient_ID', 'Age', 'Condition',
'Adherence_Rate']])
```

If you stop here, and code this function, by yourself, you will learn a lot. (HARD)

Medium

1. **Data Merging:**
   - Merge the `patients_df` and `visits_df` DataFrames on the common column `Patient_ID` using an inner join.
     - Use the `pd.merge()` function to combine the two DataFrames based on the `Patient_ID` column.
     - Choose the appropriate join type (e.g., inner, outer, left, right) based on the data requirements.
     - Store the merged DataFrame in a variable like `merged_df`.
     - Check the resulting DataFrame to ensure the merge was successful.
     - Handle any missing or duplicate values that may arise during the merge.
2. **Adherence Calculation:**
   - Calculate the medication adherence rate for each patient.
     - Create a new column `Adherence_Rate` in the `merged_df` DataFrame by dividing the `Reported_Doses_Taken` by the `Prescribed_Doses` and multiplying by 100.
     - Ensure that the calculation is accurate and handles any potential division by zero errors.
     - Round the adherence rate to two decimal places for better readability.
     - Check the resulting DataFrame to verify the correctness of the adherence rate calculation.
3. **Risk Stratification:**
   - Classify patients into different adherence levels based on their adherence rates.
     - Define a function `classify_adherence(rate)` that takes an adherence rate as input and returns a category (e.g., 'High', 'Medium', 'Low') based on predefined thresholds.
     - Apply this function to the `Adherence_Rate` column in the `merged_df` DataFrame to create a new column `Adherence_Level`.
     - Check the distribution of patients across different adherence levels.

If you stop here, and code this function, with the help of the instructions above, you still learn a lot. (MEDIUM)

Easy

Here's a dummy Python code using Pandas to tackle the outlined tasks for the medication adherence scenario:

```python
import pandas as pd

# Load the datasets
patients_df = #TODO
visits_df = #TODO

# 1. Data Merging: Combine patients and visits data into a single
DataFrame
# Assume there's a common column 'Patient_ID' in both datasets
merged_df = #TODO

# 2. Adherence Calculation: Calculate medication adherence rates
# Adherence Rate = (Reported Doses Taken / Prescribed Doses) * 100
merged_df['Adherence_Rate'] = #TODO

# 3. Risk Stratification: Classify patients into risk categories based
on adherence rate
def classify_adherence(rate):
    #TODO

merged_df['Adherence_Level'] = #TODO

# Analyze characteristics of each group
# Grouping by Adherence Level and aggregating other characteristics
grouped_df = merged_df.groupby('Adherence_Level').agg({
    'Age': 'mean',
    'Num_Medications': 'mean',
    'Condition': pd.Series.mode,
    'Patient_ID': 'nunique'  # Count of unique patients in each
adherence group
}).reset_index()

# Output the results
print(grouped_df)

# Example: Flag patients with low adherence for additional support
low_adherence_patients = #TODO
print("Patients at high risk of non-adherence:")
print(low_adherence_patients[['Patient_ID', 'Age']].drop_duplicates())
```