Part_3: Continued Analysis

Farhana Islam

1. The DI Requisition database extracted the data into 3 separate tables on 3 csv files.   You will need to combine these files into a single table. (2 Marks)

```
import pandas as pd
from datetime import datetime
df1 = pd.read_csv("DI - Visits 1.3.csv")
df2 = pd.read_csv("DI - Visits 2.3.csv")
df3 = pd.read_csv("DI - Visits 3.3.csv")
# Concatenate the DataFrames into one DataFrame
df = pd.concat([df1, df2, df3], ignore_index=True)
df
# Drop columns 7 and 8 from the DataFrame
df = df.drop(["Unnamed: 6", "Unnamed: 7"], axis=1)
# Save the combined DataFrame as a new CSV file
df.to_csv("combined_DI_data.csv", index=False)
```

2. The DI department does not have information on why patients were seen there. You will need to use the OR Booking data to determine which records from DI were from patients who were in the care pathway for an Appendectomy or Cholecystectomy.

```
import pandas as pd
from datetime import datetime
df_b = pd.read_csv("ORBooking.csv")
df_DI = pd.read_csv("combined_DI_data.csv")
df_DI
# Delete the last 3 rows from df_DI dataset
df_DI.drop(df_DI.tail(3).index, inplace=True)
df_DI
```

3. If you are using a merge command to put these two datasets together, which one would you want to put on the “left”? Explain why. (1 Mark)

I would merge the OR Booking data with the DI data (containing information about patients seen in the Diagnostic Imaging department) using the OR Booking data as the left dataset. This is because the OR Booking data contains information about patients who are scheduled for surgery, including details about the specific procedures they are undergoing (such as Appendectomy or Cholecystectomy). Therefore, by merging the DI data onto the OR Booking data, we can identify which patients seen in the Diagnostic Imaging department were on the care pathway for these specific surgeries.

4. Merge the two datasets in a way that will allow you to answer the below questions. (2 Marks)

To merge the two datasets in a way that allows us to answer the questions, we will use the OR Booking data as the left dataset and merge it with the DI data based on a common identifier such as the patient's ID or Healthcare ID (HCID).

```
# Check column names and data types
print("Columns and data types in OR Booking data:")
print(df_b.dtypes)
print("\nColumns and data types in combined DI data:")
print(df_DI.dtypes)
# Preprocess the data
df_DI['DI Req - Time'] = pd.to_datetime(df_DI['DI Req - Time'])  # Convert DI Req
- Time to datetime
df_b["OR Booking Req DT/Tm"] = pd.to_datetime(df_b["OR Booking Req DT/Tm"])
# Convert HCID column to float
df_b['HCID'] = df_b['HCID'].astype(float)
# Check column names and data types
print("Columns and data types in OR Booking data:")
print(df_b.dtypes)
print("\nColumns and data types in combined DI data:")
print(df_DI.dtypes)
```

```
# Merge the datasets based on the HCID column
merged_df = pd.merge(df_b, df_DI, on="HCID", how="left")
```

5. What is the percentage of Appendectomy patients going through DI on their pathway to the OR? (1 Mark)

```
# Filter the DataFrame to get only Appendectomy patients
appendectomy_df = merged_df[merged_df['Proc Descr Mod'] == 'Laparoscopy
Appendectomy']
# Count the total number of Appendectomy patients
total_appendectomy_patients = len(appendectomy_df)
total_appendectomy_patients
# Count the number of Appendectomy patients who went through DI
di_appendectomy_patients = appendectomy_df['DI - Pt in Suite'].notnull().sum()
di_appendectomy_patients
# Calculate the percentages
percentage_di_appendectomy = (di_appendectomy_patients /
total_appendectomy_patients) * 100
# Round the percentage to two decimal places
rounded_percentage = round(percentage_di_appendectomy, 2)
```

```
# Print the result
print("Percentage of Appendectomy patients going through DI on their pathway to
the OR:", rounded_percentage)
```

6. What is the percentage of Cholecystectomy patients going through DI on their pathway to the OR? (1 Mark)

```
# Filter the DataFrame to get only Cholecystectomy patients
cholecystectomy_df = merged_df[merged_df['Proc Descr Mod'] == 'Laparoscopy
Cholecystectomy']
# Count the total number of Cholecystectomy patients
total_cholecystectomy_patients = len(cholecystectomy_df)
total_cholecystectomy_patients
# Count the number of Cholecystectomy patients who went through DI
di_cholecystectomy_patients = cholecystectomy_df['DI - Pt in
Suite'].notnull().sum()
di_cholecystectomy_patients
percentage_di_cholecystectomy = (di_cholecystectomy_patients /
total_cholecystectomy_patients) * 100
# Round the percentage to two decimal places
rounded_percentage = round(percentage_di_cholecystectomy, 2)
# Print the result
print("Percentage of Cholecystectomy patients going through DI on their pathway
to the OR:", rounded_percentage)
```

7. For the given attributes of the patients, is there any group more likely to go through DI prior to the OR? If you cannot find anything, show you what you attempted. (2 Marks)

```
# Calculate the average age of patients undergoing DI and those in the OR booking
dataset
average_age_di = df_DI['Pt Age'].mean()
average_age_or = df_b['Pt Age'].mean()

# Print the average ages rounded to two decimal places
print("Average age of patients undergoing DI: {:.2f}".format(average_age_di))
print("Average age of patients in the OR booking dataset:
{:.2f}".format(average_age_or))
```

Given the attributes provided in the datasets (patient age, procedure description, procedure time, patient priority, OR booking request date/time, patient location, procedure date, patient check-in time in the OR, patient in OR time, OR, patient transfer, and OR number), it is difficult to determine which group is more likely to undergo diagnostic imaging (DI) prior to the operating room (OR).

However, I attempted to analyze the provided data by comparing the average age of patients undergoing DI and those in the OR booking dataset. Unfortunately, without additional relevant attributes such as gender, symptoms, medical history, or case type, it would be difficult to draw meaningful conclusions regarding which group is more likely to undergo DI prior to the OR.

To further analyze and identify any group more likely to undergo DI prior to the OR, we would need additional data or attributes that could potentially correlate with the likelihood of undergoing DI. Without such data, the analysis remains inconclusive.

8. Are there particular days of the week where wait times to be seen by DI are significantly higher or lower than the overall average for Appendectomy or Cholecystectomy patients? (2 Marks)

```
import pandas as pd

# Read the merged dataset
merged_df = pd.merge(df_b, df_DI, on="HCID", how="left")

# Filter the dataset to include only Appendectomy and Cholecystectomy patients
appendectomy_df = merged_df[merged_df['Proc Descr Mod'] == 'Laparoscopy
Appendectomy'].copy()
cholecystectomy_df = merged_df[merged_df['Proc Descr Mod'] == 'Laparoscopy
Cholecystectomy'].copy()

# Convert the 'DI Req - Time' column to datetime
appendectomy_df['DI Req - Time'] = pd.to_datetime(appendectomy_df['DI Req -
Time'])
cholecystectomy_df['DI Req - Time'] = pd.to_datetime(cholecystectomy_df['DI Req -
Time'])

# Extract the day of the week from the 'DI Req - Time' column
appendectomy_df['Day_of_Week'] = appendectomy_df['DI Req - Time'].dt.dayofweek
cholecystectomy_df['Day_of_Week'] = cholecystectomy_df['DI Req -
Time'].dt.dayofweek

# Calculate the average wait time for DI for each day of the week
avg_wait_time_appendectomy = appendectomy_df.groupby('Day_of_Week')['DI Req -
Time'].mean()
avg_wait_time_cholecystectomy = cholecystectomy_df.groupby('Day_of_Week')['DI Req
- Time'].mean()

# Compare these averages to the overall average wait time for DI
overall_avg_wait_time_appendectomy = appendectomy_df['DI Req - Time'].mean()
overall_avg_wait_time_cholecystectomy = cholecystectomy_df['DI Req -
Time'].mean()
```

```
# Print the results
print("Average wait time for DI for each day of the week (Appendectomy
patients):")
print(avg_wait_time_appendectomy)
print("\nOverall average wait time for DI (Appendectomy patients):",
overall_avg_wait_time_appendectomy)

print("\nAverage wait time for DI for each day of the week (Cholecystectomy
patients):")
print(avg_wait_time_cholecystectomy)
print("\nOverall average wait time for DI (Cholecystectomy patients):",
overall_avg_wait_time_cholecystectomy)
```

9. Are there particular times of the day where wait times to be seen by DI are significantly higher or lower than the overall average for Appendectomy or Cholecystectomy patients? (2 Marks)

```
# Extract the hour of the day from the 'DI Req - Time' column
appendectomy_df['Hour_of_Day'] = appendectomy_df['DI Req - Time'].dt.hour
cholecystectomy_df['Hour_of_Day'] = cholecystectomy_df['DI Req - Time'].dt.hour

# Calculate the average wait time for DI for each hour of the day
avg_wait_time_appendectomy = appendectomy_df.groupby('Hour_of_Day')['DI Req -
Time'].mean()
avg_wait_time_cholecystectomy = cholecystectomy_df.groupby('Hour_of_Day')['DI Req
- Time'].mean()

# Compare these averages to the overall average wait time for DI
overall_avg_wait_time_appendectomy = appendectomy_df['DI Req - Time'].mean()
overall_avg_wait_time_cholecystectomy = cholecystectomy_df['DI Req -
Time'].mean()
print("Average wait time for DI for each hour of the day (Appendectomy
patients):")
print(avg_wait_time_appendectomy)
print("Overall average wait time for DI (Appendectomy patients):",
overall_avg_wait_time_appendectomy)
print("\nAverage wait time for DI for each hour of the day (Cholecystectomy
patients):")
print(avg_wait_time_cholecystectomy)
print("Overall average wait time for DI (Cholecystectomy patients):",
      overall_avg_wait_time_cholecystectomy)
```

10. Sometimes data can be used to find quality improvement opportunities. The idea has been considered that the time difference between the patient being seen by DI (DI - Pt in Suite) and the time a patient is booked into the OR (OR Booking Req DT/Tm) is reflective of a doctor's ability to manage their workload. Is there a doctor who is significantly better than their peers in this area for one or both of the procedures? (2 Marks)

```python
from scipy.stats import f_oneway

# Calculate the time difference between DI and OR booking
merged_df['DI_Pt_in_Suite'] = pd.to_datetime(merged_df['DI - Pt in Suite'])
merged_df['OR_Booking_Req_DT'] = pd.to_datetime(merged_df['OR Booking Req
DT/Tm'])
merged_df['Time_Difference'] = (merged_df['OR_Booking_Req_DT'] -
merged_df['DI_Pt_in_Suite']).dt.total_seconds() / 60  # Convert to minutes

# Group the data by the requesting physician
grouped_by_physician = merged_df.groupby('Requesting Physician')

# Calculate the average time difference for each physician
avg_time_difference_by_physician = grouped_by_physician['Time_Difference'].mean()
avg_time_difference_by_physician
# Perform statistical tests (e.g., ANOVA) to compare the average time differences
between physicians
f_statistic, p_value =
f_oneway(*[grouped_by_physician.get_group(group)['Time_Difference'] for group in
grouped_by_physician.groups])

# significance level
alpha = 0.05

print("ANOVA Test Results:")
if p_value < alpha:
    print("There is a statistically significant difference in the average time
differences between physicians.")
else:
    print("There is no statistically significant difference in the average time
differences between physicians.")
# Identify physicians whose average time difference is significantly lower than
their peers
better_physicians =
avg_time_difference_by_physician[avg_time_difference_by_physician <
avg_time_difference_by_physician.mean()]

print("Physicians with significantly lower average time difference:")
print(better_physicians)
```