

Big Data Analytics in Healthcare using PySpark

Table of content

B. Description of the Scenario and Justify Big Data Classification	1
1. Select a Healthcare Scenario:	1
2. Justify Big Data:	1
C: Define Three Analytical Tasks	2
Task 1 - Average Recovery Time Using Demographics Factors	2
Task 2 - Maximum Recovery Time Using Surgery Details Factors	3
Task 3 -Average Recovery Time Using Lifestyle Factors	3
D: The data collection that has been generated.	5
Data Description:	5
E: Implement the Three Analytical Tasks in PySpark	7
1- Set Up PySpark with RDDs:	7
2- Implement Each Task:	8
F: Results of the tasks when executed over generated data collection	10
Run the Analytical Task and Document the Results:	10

Description of the Scenario and Justify Big Data Classification

1. Select a Healthcare Scenario:


Scenario: Recovery time after surgery

Description: For this project, we are examining factors that influence recovery time after surgery. This includes analysing various patient attributes, such as diabetes status, age, gender, smoking status, alcohol use type of surgery performed, surgery duration, and anaesthesia type. By studying these factors, the goal is to identify patterns and insights that could lead to faster recovery times, improved patient outcomes, and more efficient post-operative care.

2. Justify Big Data:

Below is how each characteristic of big data applies to our analysis:

- **Volume:** The healthcare domain produces substantial data volumes, encompassing patient records, surgical histories, and sensor data collected during and after surgery. This large dataset allows for comprehensive analysis across numerous variables, providing a rich foundation for insight. Our dataset contains 100,000 records for samples because of the limitation of the dataset generator tool.
- **Velocity:** Data in healthcare, particularly in post-surgical monitoring, is generated continuously and often in real-time. Patient vitals, sensor readings, and other metrics are tracked throughout recovery, allowing us to update models dynamically and identify trends promptly.
- **Variety:** The data involved includes a variety of formats as structured records, unstructured texts, numerical data from sensors, and imaging data, each providing unique insights into post-operative recovery. This diversity enables a holistic view of patient health and recovery patterns.
- **Veracity:** Data quality is crucial. Given that healthcare data often contains inaccuracies, significant processing and cleaning are required to ensure reliability. Since much of the data is captured by professionals or automated devices, we can enhance data trustworthiness, enabling accurate conclusions for better patient care.
- **Value:** This analysis offers both economic and social value. Economically, by improving recovery times and patient outcomes, healthcare providers may reduce costs associated with extended care. Socially, these insights contribute to



advancements in healthcare research, potentially benefiting patient recovery protocols globally.

- **Volatility:** As patient health conditions evolve, the relevance of specific data points may change. However, some captured data, particularly demographic and historical health data, remains valuable long-term, supporting ongoing improvements in post-surgical care.
- **Vulnerability:** Security is paramount, as our data is sensitive and involves patient health details. Ensuring stringent security measures will protect patient privacy and maintain compliance with healthcare data regulations.

Define Three Analytical Tasks

All tasks will try to predict the recovery time after surgery but focus on different features.

Task 1 - Average Recovery Time Using Demographics Factors

The first task aims to predict the recovery time for patients after undergoing surgery, based on several key factors. This analysis helps identify which attributes contribute most to recovery duration, enabling healthcare providers to plan post-operative care more effectively.

The features used in this prediction include:

- **Age:** Older patients might have slower recovery times due to decreased resilience and possible pre-existing conditions.
- **Diabetes Status:** Diabetes can slow down recovery due to factors like reduced immune function and slower wound healing.

Purpose: The goal is to utilize these factors to build a basic predictive model that can estimate recovery time. This can assist clinicians in identifying patients at risk of prolonged recovery, helping them make more informed decisions about post-surgical interventions and resources needed for optimal patient care.

Analytical Approach: This task utilizes PySpark RDDs to process and analyze the data. By aggregating recovery times based on key features, this task allows for an understanding of how each attribute influences recovery, providing valuable insights into post-operative care planning. The output will show average recovery times based on different surgery types, helping identify procedures that generally require more extended recovery periods.

Task 2 - Maximum Recovery Time Using Surgery Details Factors

This task aims to analyze and compare the combination of surgery details that result in the longest recovery time.


Purpose: Analyzing the maximum recovery times after surgery is crucial for optimizing patient care and outcomes. By considering factors such as the type of surgery, its duration, and the type of anaesthesia used, healthcare providers can better predict recovery trajectories. This analysis helps in tailoring post-operative care plans, managing patient expectations, and minimizing complications. For instance, more invasive surgeries or those with longer durations typically require extended recovery periods, while the type of anaesthesia can influence immediate post-operative recovery. Ultimately, this detailed analysis supports personalized patient care, enhancing overall recovery experiences and outcomes.

- **Type of Surgery:** Specific procedure performed (knee replacement 0, heart bypass 1).
- **Surgery Duration:** Length of the surgical procedure in hours (1-10 hours)
- **Anesthesia Type:** Type of anesthesia used (general 0, local 1).

Analytical Approach: The analytical approach involves parsing a dataset of surgical records to extract relevant features such as type of surgery, surgery duration, anaesthesia type, and recovery time. By mapping these features to their respective recovery times, the data is aggregated to identify the maximum recovery time for each unique combination of features. Finally, the results are collected and analyzed to determine the specific combination that yields the highest recovery time, allowing for insights into factors affecting patient recovery. This method leverages data processing techniques to facilitate informed decision-making in surgical practices.

Task 3 -Average Recovery Time Using Lifestyle Factors

This analysis explores how lifestyle factors, specifically smoking and alcohol use, impact recovery times after surgery. It is well established that a healthy lifestyle, both before and after surgery, can significantly influence healing speed and reduce complications. Smoking and alcohol use are particularly important risk factors, as both have been associated with increased complications post-surgery, including infection, wound healing delays, and cardiac or respiratory issues. By analyzing recovery times across different lifestyle profiles, this study aims to quantify how these factors affect post-surgical recovery.



Purpose: The objective is to offer insights that aid healthcare providers in planning post-operative care and resource allocation based on patients' lifestyle factors, specifically smoking and alcohol use. Understanding the correlation between these lifestyle choices and recovery duration allows for more tailored care strategies, such as enhanced monitoring, targeted wound care, or additional follow-up visits. This information supports surgical planning and patient counselling, helping to set realistic recovery expectations and identify patients who may benefit from additional support to reduce complications and promote optimal healing.

Analytical Approach: Categorization and Analysis of Smoking and Alcohol Use. Patients are grouped based on their smoking and alcohol use. In this analysis:

- **Smoking Status:** 1 for non-smokers and 0 for smokers.
- **Alcohol Use:** 1 for non-users and 0 for those who use alcohol.

Using PySpark RDDs, the dataset is divided into four groups based on these categories, and the average recovery time for each group is calculated. PySpark's efficiency in processing large datasets makes it ideal for handling extensive healthcare data, enabling a quick and comprehensive analysis.

Data Output: The analysis will compare the average recovery times for the following groups:

1. Non-smokers who do not use alcohol
2. Smokers who use alcohol
3. Smokers who do not use alcohol
4. Non-smokers who use alcohol

If, for instance, patients who smoke and consume alcohol experience significantly longer recovery times, healthcare providers could implement preemptive measures to support these individuals. These measures may include additional post-operative visits, specialized wound care, or tailored discharge protocols.

The data collection that has been generated.

Data Description:

We use the tool provided, "Online Data Generator", with these features:

- **Patient ID**

- **Patient Name**
- **Patient LastName**

Patient Demographics

- **Age Group:** Age group at the time of surgery.
Group 0: 18-30 years
Group 1: 31-45 years
Group 2: 46-60 years
Group 3: 61-75 years
Group 4: 76+ year
- **Gender:** Male, female, or other gender identities.
- **Location:** Country (0-190)
- **Body Mass Index (BMI):** Calculated based on height and weight (12-65) .
- **Diabetes Status:** Presence or absence of diabetes (0 , 1).
- **Blood Pressure Systolic:** Preoperative blood pressure readings (3-20).
- **Blood Pressure Diastolic:** Preoperative blood pressure readings (3-20).
- **Heart Rate:** Preoperative heart rate (40-190).

Life Style:

- **Smoking Status:** Current smoker, or non-smoker (0 ,1)
- **Alcohol Use:** Frequent alcohol consumption, or no alcohol consumption (0 ,1).
- **Exercise Level:** How many hours of exercise per week (0-40).

Surgery Details:

- **Type of Surgery:** Specific procedure performed (knee replacement 0,heart bypass 1).
- **Surgery Duration:** Length of the surgical procedure in hours (1-10 hours)
- **Anesthesia Type:** Type of anesthesia used (general 0, local 1).

Post-Surgery Recovery:

- **Recovery Time:** Total time to full recovery in days (1-30).
- **Length of Hospital Stay:** Number of days hospitalized post-surgery (0-30).
- **Pain Levels:** Post-surgery pain levels recorded over time (0-10).
- **Physical Therapy Sessions:** Number of physical therapy sessions attended in a week (0-7).
- **Infection Occurrence:** Whether the patient experienced any postoperative infections (0,1).

Implement the Three Analytical Tasks in PySpark

1- Set Up PySpark with RDDs:

Here is the setup PySpark environment and data loading steps on Google Colab:

```
[1] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[2] #Setup Spark Environment:
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://archive.apache.org/dist/spark/spark-3.0.1/spark-3.0.1-bin-hadoop2.7.tgz
!tar xf spark-3.0.1-bin-hadoop2.7.tgz
!pip install -q findspark
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.1-bin-hadoop2.7"
import findspark
findspark.init()

[15] # Initialize SparkContext:
from pyspark import SparkContext
try:
    sc = SparkContext("local", "HealthcareRecoveryAnalysis")
except ValueError:
    print("SparkContext already exists.")

[9] # Load the Data and Verify:
file_path = "/content/drive/MyDrive/Colab Notebooks/Data transformation/ExportCSV.csv"
data_rdd = sc.textFile(file_path)

# Preview the data to confirm structure
print("Data preview:", data_rdd.take(5))

Data preview: ['Patient ID, Patient Name, Patient Last Name, Gender, Age group, Location, BMI, Diabetes, Blood pressure Systolic, Blood pressure Diastoli

[7] pip install pyspark

Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)

[10] # Filter Out the Header and Parse Rows:
header = data_rdd.first() # Extract the header row
data_rdd = data_rdd.filter(lambda row: row != header) # Filter out the header
```

Note: There are no null values in the dataset or any outlier so no need for data preprocessing.

2- Implement Each Task:

- **Task 1 - Using Patient Demographics**

The generated Google Colab codes is shown in the figure below

```
[11] # Extract Features and Handle Errors:
def extract_features(record):
    fields = record.split(",")
    try:
        age_group = int(fields[4].strip())
        diabetes_status = int(fields[7].strip())
        recovery_time = int(fields[17].strip())
        return (diabetes_status, age_group, recovery_time)
    except (ValueError, IndexError) as e:
        print(f"Skipping row due to error: {e}")
        return None

features_rdd = data_rdd.map(extract_features).filter(lambda x: x is not None)

[12] # Aggregate and Calculate Averages:
mapped_rdd = features_rdd.map(lambda x: ((x[0], x[1]), (x[2], 1)))
reduced_rdd = mapped_rdd.reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))
average_recovery_rdd = reduced_rdd.mapValues(lambda x: x[0] / x[1])

# Display Results
age_group_labels = {0: "18-30", 1: "31-45", 2: "46-60", 3: "61-75", 4: "76+"}
results = average_recovery_rdd.collect()

for (diabetes_status, age_group), avg_recovery_time in results:
    diabetes_text = "Diabetic" if diabetes_status == 1 else "Non-Diabetic"
    age_group_text = age_group_labels.get(age_group, "Unknown")
    print(f"Diabetes Status: {diabetes_text}, Age Group: {age_group_text}, Average Recovery Time: {avg_recovery_time:.2f} days")

Diabetes Status: Diabetic, Age Group: 76+, Average Recovery Time: 15.56 days
Diabetes Status: Non-Diabetic, Age Group: 61-75, Average Recovery Time: 15.49 days
Diabetes Status: Non-Diabetic, Age Group: 31-45, Average Recovery Time: 15.56 days
Diabetes Status: Non-Diabetic, Age Group: 46-60, Average Recovery Time: 15.45 days
Diabetes Status: Diabetic, Age Group: 18-30, Average Recovery Time: 15.46 days
Diabetes Status: Diabetic, Age Group: 61-75, Average Recovery Time: 15.43 days
Diabetes Status: Non-Diabetic, Age Group: 18-30, Average Recovery Time: 15.54 days
Diabetes Status: Diabetic, Age Group: 31-45, Average Recovery Time: 15.51 days
Diabetes Status: Non-Diabetic, Age Group: 76+, Average Recovery Time: 15.50 days
Diabetes Status: Diabetic, Age Group: 46-60, Average Recovery Time: 15.57 days

[14] sc.stop()
```

- **Task 2 - Maximum recovery time based on Surgery Details**

The generated Google Colab codes are shown in the figure below


```
✓ 4s ▶ # Function to parse the CSV line
def parse_line(line):
    fields = line.split(',')
    try:
        # Extracting values (Type of Surgery, Surgery Duration, Anesthesia Type, Recovery Time)
        #Note that to reach better combination I round the "Surgery Duration"
        return (int(fields[14]), round(float(fields[15])), int(fields[16]), int(fields[17]))
    except (IndexError, ValueError) as e:
        print(f"Error parsing line: {line} - {e}")
        return None

# Map and filter out None values
mapped_rdd = data_rdd.map(parse_line).filter(lambda x: x is not None)

# Extract relevant info for max recovery time
info_rdd = mapped_rdd.map(lambda record: ((record[0], record[1], record[2]), record[3]))

max_recovery_rdd = info_rdd.reduceByKey(lambda a, b: max(a, b))

# Collect results
results = max_recovery_rdd.collect()

# Find the combination with the maximum recovery time
max_combination = max(results, key=lambda x: x[1])

# Show the result
print(f"Combination with Maximum Recovery Time:")
print(f"Type of Surgery: {max_combination[0][0]}, Surgery Duration: {max_combination[0][1]}, Anesthesia Type: {max_combination[0][2]}, Max Recovery Time: {max_combination[1]} days")

Combination with Maximum Recovery Time:
Type of Surgery: 0, Surgery Duration: 5, Anesthesia Type: 0, Max Recovery Time: 30 days
```

● Task 3 - Using Lifestyle Factors

```
▶ # Define a function to extract features and ensure binary values
def extract_features(record):
    fields = record.split(",")
    try:
        smoking_status = int(fields[11].strip()) # Smoking Status
        alcohol_use = int(fields[12].strip()) # Alcohol Use
        recovery_time = float(fields[17].strip()) # Recovery Time

        # Ensure binary values for Smoking Status and Alcohol Use
        if smoking_status in {0, 1} and alcohol_use in {0, 1}:
            return ((smoking_status, alcohol_use), (recovery_time, 1))
        else:
            print(f"Skipping row due to non-binary values: Smoking Status={smoking_status}, Alcohol Use={alcohol_use}")
            return None
    except (ValueError, IndexError) as e:
        print(f"Skipping row due to error: {e}")
        return None

# Apply extraction and filter out None results
features_rdd = data_rdd.map(extract_features).filter(lambda x: x is not None)

# Aggregate by key to calculate the total recovery time and count for each combination
aggregated_rdd = features_rdd.reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))

# Calculate the average recovery time for each combination
average_recovery_rdd = aggregated_rdd.mapValues(lambda x: x[0] / x[1])

# Collect and print results to verify the number of unique combinations
results = average_recovery_rdd.collect()
print("Number of unique combinations:", len(results))
```

```
# Interpret and display the results
for (smoking_status, alcohol_use), avg_recovery_time in results:
    smoking_text = "Smoker" if smoking_status == 1 else "Non-Smoker"
    alcohol_text = "Alcohol User" if alcohol_use == 1 else "Non-Alcohol User"
    print(f"Smoking Status: {smoking_text}, Alcohol Use: {alcohol_text}, Average Recovery Time: {avg_recovery_time:.2f} days")

# Stop the SparkContext
sc.stop()
```

```
Number of unique combinations: 4
Smoking Status: Smoker, Alcohol Use: Alcohol User, Average Recovery Time: 15.52 days
Smoking Status: Non-Smoker, Alcohol Use: Non-Alcohol User, Average Recovery Time: 15.51 days
Smoking Status: Non-Smoker, Alcohol Use: Alcohol User, Average Recovery Time: 15.52 days
Smoking Status: Smoker, Alcohol Use: Non-Alcohol User, Average Recovery Time: 15.48 days
```

Results of the tasks when executed over generated data collection

Run the Analytical Task and Document the Results:

We Executed our PySpark script for each of the 3 tasks and captured the results.

- **Task 1 - Using Patient Demographics**

The results show no significant difference between Diabetes Status and Age Group for Average Recovery time. It is important to note that this conclusion is based on synthetic data.

Diabetes Status	Age Group	Average Recovery time
Diabetic	18-30	15.46
Diabetic	31-45	15.51
Diabetic	46-60	15.57
Diabetic	61-75	15.43
Diabetic	76+	15.56
Non-Diabetic	18-30	15.54
Non-Diabetic	31-45	15.56
Non-Diabetic	46-60	15.45
Non-Diabetic	61-75	15.49
Non-Diabetic	76+	15.5

Table 1: Output using patient demographics

• Task 2 - Maximum and Minimum recovery time based on Surgery Details

The results indicate that for Type of Surgery 0, which corresponds to knee replacement, a Surgery Duration of 5 days and General Anesthesia Type is associated with the highest maximum recovery time of 30 days.

Combination with Maximum Recovery Time
Type of Surgery: 0, Surgery Duration: 5, Anesthesia Type: 0, Max Recovery Time: 30 days

Table 2: Output using Surgery Details

• Task 3 - Using Lifestyle Factors

The results show no significant difference between Smoking Status and Alcohol use for Average Recovery time. It is important to note that this conclusion is based on synthetic data.

Smoking Status	Alcohol use	Average Recovery time
Smoker	Alcohol user	15.52
Smoker	Non-alcohol user	15.51
Non-smoker	Alcohol user	15.52
Non-smoker	Non-alcohol user	15.48

Table 3: Output using Lifestyle Factors