MAY 21, 2025

LIVERPOOL HOPE UNIVERSITY 1844

BIG DATA AND CLOUD COMPUTING COURSEWORK 2

DATA CLASSIFICATION IN HEALTHCARE

AHMED ABDULLAH SHAHID
24012028

Table of Contents

Introduction

This project is about building a strong and reliable Java program that works with patient record CSV files on a Hadoop YARN cluster. The main goal is to make a system that can read and write files using the Hadoop Distributed File System (HDFS) and sort patient data in different ways. The application creates three separate CSV files, each showing the data grouped by a different type: NHS number, patient symptoms and NHS Trust region. It also includes a feature where users can search through the data by entering a patient's NHS number, symptoms (in any letter case) or region. This makes searching flexible and easier. The project runs on a Linux virtual machine in Microsoft Azure, taking advantage of Hadoop's ability to handle large amounts of data and work well even if something goes wrong. This report explains how the system was designed and built and how it can help with managing healthcare data and supporting medical research. It shows how using distributed computing can make it easier and faster to work with healthcare data today.

Project Objectives

The primary aims of the project were to:

- **Develop a distributed application:** Create a Java application that processes patient record CSV files and runs on a YARN cluster.
- **Integrate HDFS for I/O:** Read the input CSV file from HDFS and write the output files (filtered CSVs) directly to HDFS.
- **Implement multiple data classifications:** Generate three output CSV files after reordering fields according to NHS numbers, symptoms and NHS Trust region.
- **Enable interactive querying:** Incorporate an interactive MapReduce query routine where the user can specify classification criteria and view formatted output.
- **Deploy on YARN:** Package and deploy the application on YARN, ensuring that the application is scalable and efficient under a distributed computing environment.

Development Environment

The system is developed using:

- **Programming Language:** Java
- **Big Data Framework:** Apache Hadoop (YARN)
- **Cloud Platform:** Microsoft Azure (Linux-based virtual machine)
- **HDFS (Hadoop Distributed File System)** for storing and processing large datasets
- **Linux Terminal Commands** for execution and system interaction

System Design and Architecture

1. **Component Overview**

The application works directly with the Hadoop Distributed File System (HDFS) by using the Hadoop FileSystem API. This allows it to read a CSV file called *MedicalFiles.csv* from HDFS. Before starting the main task, it checks if the input file exists and deletes any old output files to make sure the system starts fresh. The program then creates three new CSV files, each one grouped by a different type: NHS number, patient symptoms and NHS Trust region. These new files are saved back into HDFS using tools that help keep the format and text encoding correct. In each file, the main category (like NHS number or symptom) is moved to the front to make it easier to understand.

To make sure the raw data is easy to work with, the program has a special part for cleaning and organizing the data. It includes two helpful methods. The first one, called *parseCSVLine*, makes sure lines are split correctly even when commas appear inside quotes. The second one, *adjustFields*, fixes common problems like date fields being broken into two parts (for example: "12-Apr", "2024"). It also fills in missing information when needed and makes sure each row has exactly 12 fields. This cleanup is very important for the rest of the program to work properly.

The MapReduce part of the application has two main parts: *TokenizerMapper* and *ClassificationReducer*. The mapper changes its behavior depending on what the user wants to search for like patient ID, symptom or region. It picks the right field from each line, cleans it up (for example, by removing extra quotes or changing the letter case) and sends it along with the rest of the row. The reducer then puts similar rows together, removes any duplicates and prepares the results in a nice format for the user to read. After each search, the results are shown on the screen and then cleaned up. The whole program works in two main steps, one to classify and save the data, and one to let users search through it.

2. **Pre-processing Phase:**

In the **Pre-processing Phase**, the application begins by reading a file called *MedicalFiles.csv*, which is stored in the Hadoop Distributed File System (HDFS). It first checks if the file exists. Then, it gets the system ready by cleaning up any old output data. After that, it reads the CSV file and fixes problems in the data especially with admission dates that are sometimes split into two columns.

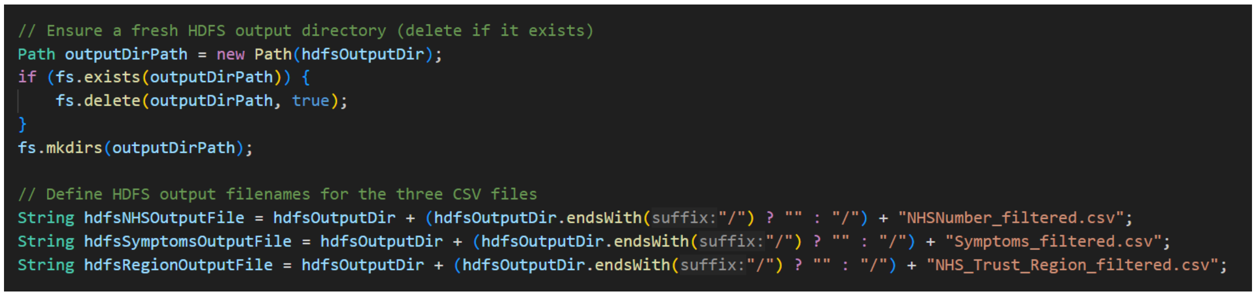
Next, the program creates three new CSV files with updated formats:

- **NHSNumber_filtered.csv**
  This file keeps the original column order and includes all patient records. It is mainly used for sorting or searching by NHS numbers.

- **Symptoms_filtered.csv**
  This file moves the Chief_Complaint (the main symptom) to the first column, making it easier for users to sort or search by symptoms. Additionally, the file is sorted by symptoms, so that all patients with the same symptom appear together, followed by the next symptom group and so on. This organization improves readability and supports faster symptom-based analysis.

- **NHS_Trust_Region_filtered.csv**
  In this file, the NHS_Trust_Region field is moved to the first column. This helps users sort and search the data by region more easily. The file is also sorted by region, so it shows one region and all the patients from that region, then the next region and its patients. This makes it easier to study and compare data from different areas.

All three files are saved back to HDFS using tools that keep the text format and structure correct (*FSDataOutputStream* and *BufferedWriter*).



```java
// Ensure a fresh HDFS output directory (delete if it exists)
Path outputDirPath = new Path(hdfsOutputDir);
if (fs.exists(outputDirPath)) {
    fs.delete(outputDirPath, true);
}
fs.mkdirs(outputDirPath);

// Define HDFS output filenames for the three CSV files
String hdfsNHSOutputFile = hdfsOutputDir + (hdfsOutputDir.endsWith(suffix:"/") ? "" : "/") + "NHSNumber_filtered.csv";
String hdfsSymptomsOutputFile = hdfsOutputDir + (hdfsOutputDir.endsWith(suffix:"/") ? "" : "/") + "Symptoms_filtered.csv";
String hdfsRegionOutputFile = hdfsOutputDir + (hdfsOutputDir.endsWith(suffix:"/") ? "" : "/") + "NHS_Trust_Region_filtered.csv";
```

3. Interactive MapReduce Query Phase:

In the Interactive MapReduce Query Phase, users can search patient records using their own chosen criteria. After the preprocessing step is done the program asks the user in the console to choose one of the following options:

- 1. Search by Patient NHS Number
- 2. Search by Symptom
- 3. Search by Region

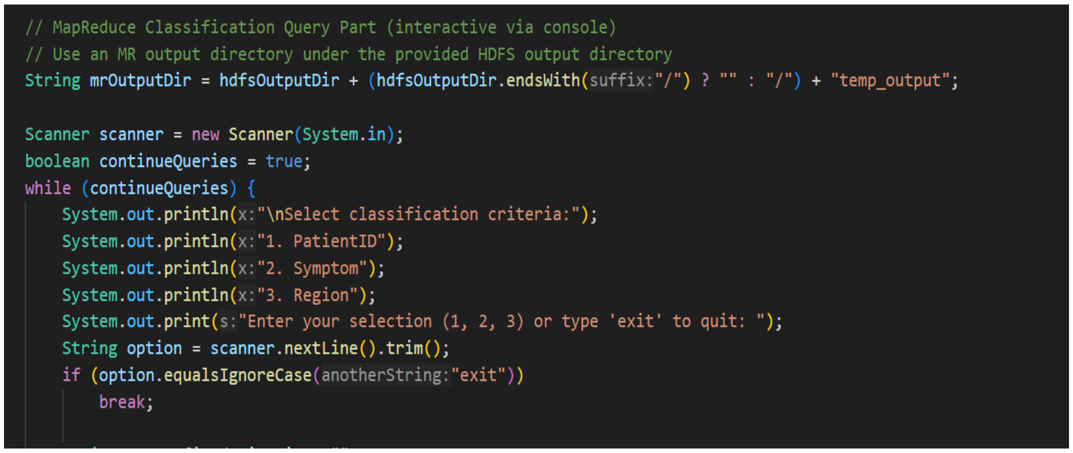
After picking one of the options, the user types a value to search for, for example a patient ID, a symptom like "fever" or a region name like "South West".

The program then sends this value, along with the chosen option, into a MapReduce job:

- The Mapper reads each line in the CSV file and fixes any problems, using the same logic as before. It checks the correct field (for example, field 7 for symptoms or field 11 for region) and compares it with the user's input. For symptoms and regions, the program ignores upper/lower case and can find partial matches.

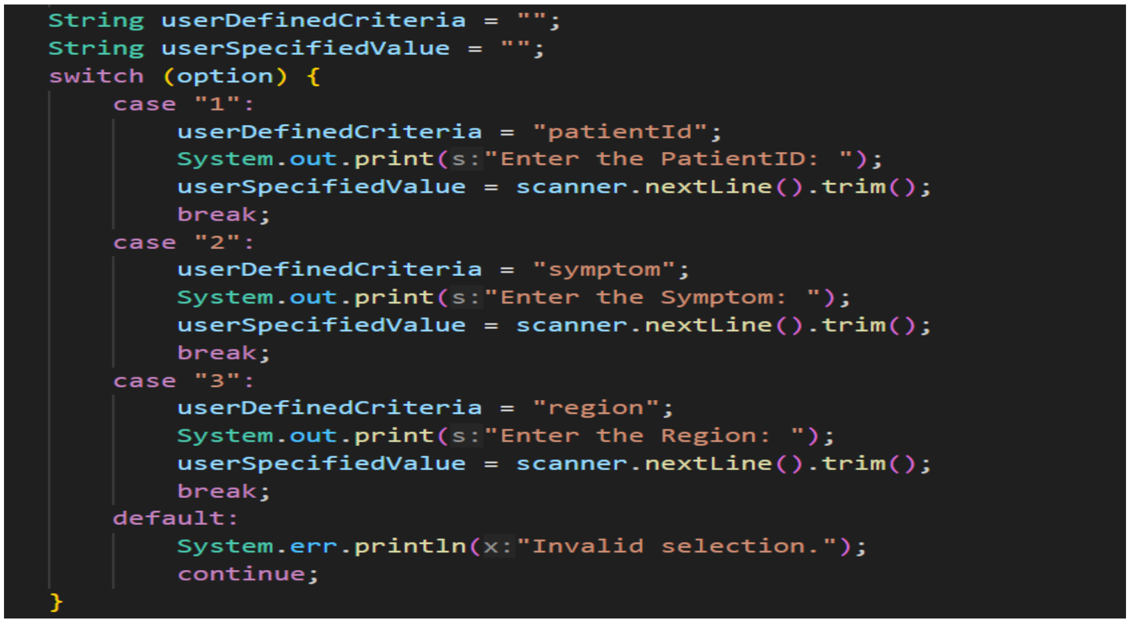
- The Reducer removes duplicates and prepares the matched records to look nice when shown.

The results of the search are stored temporarily in HDFS (in a folder called *temp_output*) and are also printed on the screen. Users can repeat the process and do as many searches as they want in the same session.



```
// MapReduce Classification Query Part (interactive via console)
// Use an MR output directory under the provided HDFS output directory
String mrOutputDir = hdfsOutputDir + (hdfsOutputDir.endsWith(suffix:"/") ? "" : "/") + "temp_output";

Scanner scanner = new Scanner(System.in);
boolean continueQueries = true;
while (continueQueries) {
    System.out.println(x:"\nSelect classification criteria:");
    System.out.println(x:"1. PatientID");
    System.out.println(x:"2. Symptom");
    System.out.println(x:"3. Region");
    System.out.print(s:"Enter your selection (1, 2, 3) or type 'exit' to quit: ");
    String option = scanner.nextLine().trim();
    if (option.equalsIgnoreCase(anotherString:"exit"))
        break;
```

The mapper then performs conditional logic based on this configuration:



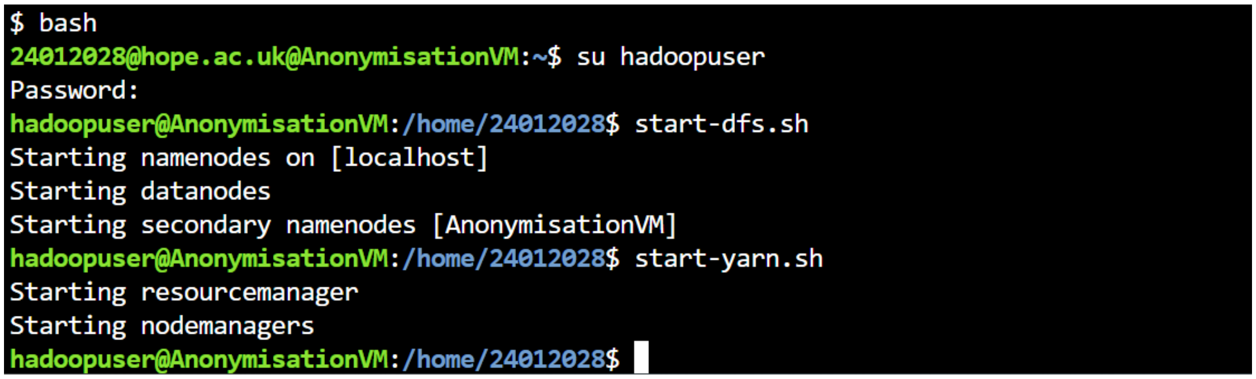
```
String userDefinedCriteria = "";
String userSpecifiedValue = "";
switch (option) {
    case "1":
        userDefinedCriteria = "patientId";
        System.out.print(s:"Enter the PatientID: ");
        userSpecifiedValue = scanner.nextLine().trim();
        break;
    case "2":
        userDefinedCriteria = "symptom";
        System.out.print(s:"Enter the Symptom: ");
        userSpecifiedValue = scanner.nextLine().trim();
        break;
    case "3":
        userDefinedCriteria = "region";
        System.out.print(s:"Enter the Region: ");
        userSpecifiedValue = scanner.nextLine().trim();
        break;
    default:
        System.err.println(x:"Invalid selection.");
        continue;
}
```

YARN Deployment

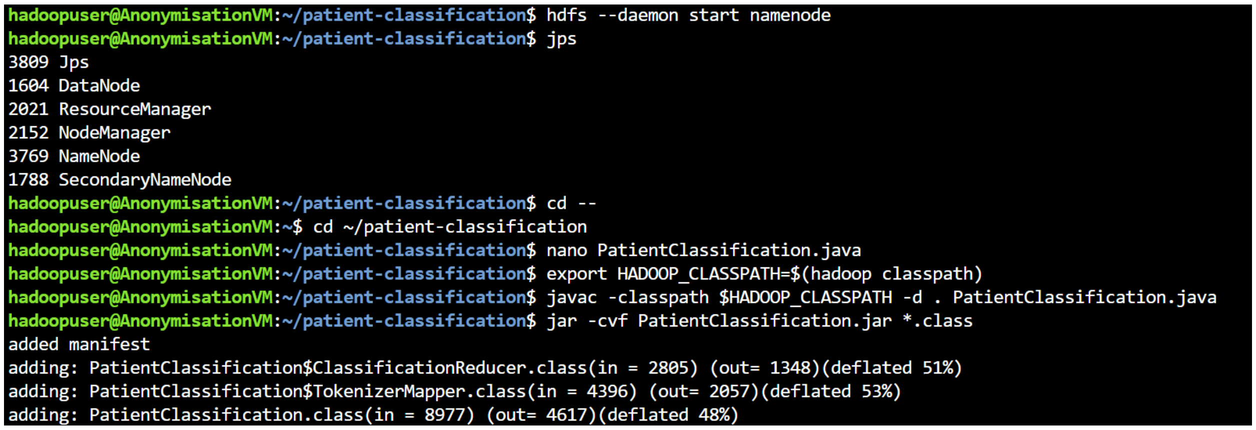
To run the Java-based patient classification program on a Hadoop YARN cluster, several steps were followed in a Linux system. First, the Hadoop services started using the commands start-dfs.sh and start-yarn.sh. This made sure that both the file system (HDFS) and the job manager (YARN) were ready to work.

Next, a new folder called patient-classification was created in the home directory by using the command mkdir ~/patient-classification. Then, the user moved into this folder by typing cd ~/patient-classification.

After that, the main Java file called PatientClassification.java was created or edited using the text editor nano, by typing nano PatientClassification.java.



```
$ bash
24012028@hope.ac.uk@AnonymisationVM:~$ su hadoopuser
Password:
hadoopuser@AnonymisationVM:/home/24012028$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [AnonymisationVM]
hadoopuser@AnonymisationVM:/home/24012028$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoopuser@AnonymisationVM:/home/24012028$
```



```
hadoopuser@AnonymisationVM:~/patient-classification$ hdfs --daemon start namenode
hadoopuser@AnonymisationVM:~/patient-classification$ jps
3809 Jps
1604 DataNode
2021 ResourceManager
2152 NodeManager
3769 NameNode
1788 SecondaryNameNode
hadoopuser@AnonymisationVM:~/patient-classification$ cd --
hadoopuser@AnonymisationVM:~$ cd ~/patient-classification
hadoopuser@AnonymisationVM:~/patient-classification$ nano PatientClassification.java
hadoopuser@AnonymisationVM:~/patient-classification$ export HADOOP_CLASSPATH=$(hadoop classpath)
hadoopuser@AnonymisationVM:~/patient-classification$ javac -classpath $HADOOP_CLASSPATH -d . PatientClassification.java
hadoopuser@AnonymisationVM:~/patient-classification$ jar -cvf PatientClassification.jar *.class
added manifest
adding: PatientClassification$ClassificationReducer.class(in = 2805) (out= 1348)(deflated 51%)
adding: PatientClassification$TokenizerMapper.class(in = 4396) (out= 2057)(deflated 53%)
adding: PatientClassification.class(in = 8977) (out= 4617)(deflated 48%)
```

After setting up the local environment, folders were also created in HDFS by using the commands hdfs dfs -mkdir -p /user/hadoop/input and hdfs dfs -mkdir -p /user/hadoop/output.

These folders are needed for storing the input data and saving the output results in the Hadoop system.

To prepare for compiling the Java program, the Hadoop classpath was added to the environment using this command:
export HADOOP_CLASSPATH=$(hadoop classpath).
This helped the Java compiler find all the necessary Hadoop files.

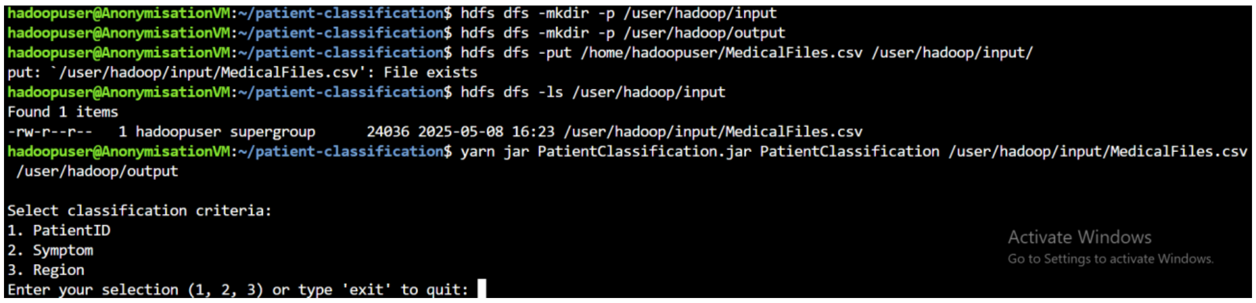
The Java file was compiled with this command:
javac -classpath $HADOOP_CLASSPATH -d . PatientClassification.java,
and the compiled files were packed into a JAR file using:
jar -cvf PatientClassification.jar *.class.

To give the program some data, the file **MedicalFiles.csv** was uploaded to HDFS using:
hdfs dfs -put /home/hadoopuser/MedicalFiles.csv /user/hadoop/input/.
This upload was checked with:
hdfs dfs -ls /user/hadoop/input.

Then, the program was run on the Hadoop YARN cluster using:
yarn jar PatientClassification.jar PatientClassification /user/hadoop/input/MedicalFiles.csv /user/hadoop/output.
This command sent the job to YARN to be processed.

After the program finished, the output files were checked with:
hdfs dfs -ls /user/hadoop/output,
to make sure the results were created successfully.

To show that everything worked correctly, screenshots of the terminal commands are included.



```
hadoopuser@AnonymisationVM:~/patient-classification$ hdfs dfs -mkdir -p /user/hadoop/input
hadoopuser@AnonymisationVM:~/patient-classification$ hdfs dfs -mkdir -p /user/hadoop/output
hadoopuser@AnonymisationVM:~/patient-classification$ hdfs dfs -put /home/hadoopuser/MedicalFiles.csv /user/hadoop/input/
put: `/user/hadoop/input/MedicalFiles.csv': File exists
hadoopuser@AnonymisationVM:~/patient-classification$ hdfs dfs -ls /user/hadoop/input
Found 1 items
-rw-r--r--   1 hadoopuser supergroup      24036 2025-05-08 16:23 /user/hadoop/input/MedicalFiles.csv
hadoopuser@AnonymisationVM:~/patient-classification$ yarn jar PatientClassification.jar PatientClassification /user/hadoop/input/MedicalFiles.csv
 /user/hadoop/output

Select classification criteria:
1. PatientID
2. Symptom
3. Region
Enter your selection (1, 2, 3) or type 'exit' to quit:
```

Results and Observations

The system works in two main parts. First, in the **pre-processing phase**, it reads the input CSV file and changes the data into three new CSV files. Each file focuses on one thing: NHS numbers, symptoms or regions. This helps organize the data in a better way. In the second part, called the **interactive MapReduce query phase**, the user can choose how they want to search the data. They can enter a patient ID, a symptom (like "fever") or a region. The system then runs a MapReduce job based on this input. The code is made to read files from HDFS, clean and organize the data correctly and allow easy searching. It also handles different text formats and letter cases, so the searches work even if the input is typed differently.

1. Pre-Processing Phase Results

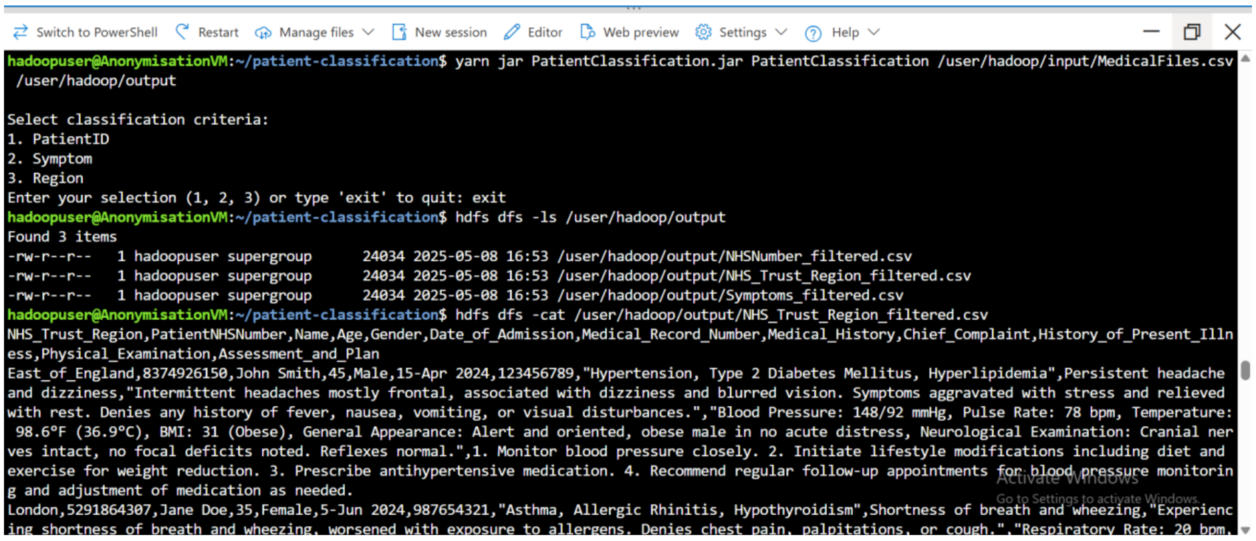
During the **pre-processing stage**, the application reads the CSV file from HDFS and uses regular expressions to correctly split and clean the data. Two helper methods, parseCSVLine and adjustFields, play an important role. They fix problems like split date fields and make sure each record has the right number of columns.

The program then creates three different output files:

- **NHS Classification File**: This file keeps all columns in their original order. It keeps the NHS number first and keeps the raw patient data unchanged.

- **Symptoms Classification File**: This file moves the **Chief_Complaint** (symptom) field to the front. This helps users easily search based on symptoms.
- **Region Classification File**: This file moves the **NHS_Trust_Region** field to the front. This makes it easier to search data by region.

Each file is saved back to HDFS using the correct format and encoding. This helps keep the data clean and ready for use in a distributed system like Hadoop.



```
hadoopuser@AnonymisationVM:~/patient-classification$ yarn jar PatientClassification.jar PatientClassification /user/hadoop/input/MedicalFiles.csv
 /user/hadoop/output

Select classification criteria:
1. PatientID
2. Symptom
3. Region
Enter your selection (1, 2, 3) or type 'exit' to quit: exit
hadoopuser@AnonymisationVM:~/patient-classification$ hdfs dfs -ls /user/hadoop/output
Found 3 items
-rw-r--r--   1 hadoopuser supergroup      24034 2025-05-08 16:53 /user/hadoop/output/NHSNumber_filtered.csv
-rw-r--r--   1 hadoopuser supergroup      24034 2025-05-08 16:53 /user/hadoop/output/NHS_Trust_Region_filtered.csv
-rw-r--r--   1 hadoopuser supergroup      24034 2025-05-08 16:53 /user/hadoop/output/Symptoms_filtered.csv
hadoopuser@AnonymisationVM:~/patient-classification$ hdfs dfs -cat /user/hadoop/output/NHS_Trust_Region_filtered.csv
NHS_Trust_Region,PatientNHSNumber,Name,Age,Gender,Date_of_Admission,Medical_Record_Number,Medical_History,Chief_Complaint,History_of_Present_Illn
ess,Physical_Examination,Assessment_and_Plan
East_of_England,8374926150,John Smith,45,Male,15-Apr 2024,123456789,"Hypertension, Type 2 Diabetes Mellitus, Hyperlipidemia",Persistent headache
and dizziness,"Intermittent headaches mostly frontal, associated with dizziness and blurred vision. Symptoms aggravated with stress and relieved
with rest. Denies any history of fever, nausea, vomiting, or visual disturbances.","Blood Pressure: 148/92 mmHg, Pulse Rate: 78 bpm, Temperature:
 98.6°F (36.9°C), BMI: 31 (Obese), General Appearance: Alert and oriented, obese male in no acute distress, Neurological Examination: Cranial ner
ves intact, no focal deficits noted. Reflexes normal.",1. Monitor blood pressure closely. 2. Initiate lifestyle modifications including diet and
exercise for weight reduction. 3. Prescribe antihypertensive medication. 4. Recommend regular follow-up appointments for blood pressure monitorin
g and adjustment of medication as needed.
London,5291864307,Jane Doe,35,Female,5-Jun 2024,987654321,"Asthma, Allergic Rhinitis, Hypothyroidism",Shortness of breath and wheezing,"Experienc
ing shortness of breath and wheezing, worsened with exposure to allergens. Denies chest pain, palpitations, or cough.","Respiratory Rate: 20 bpm,
```

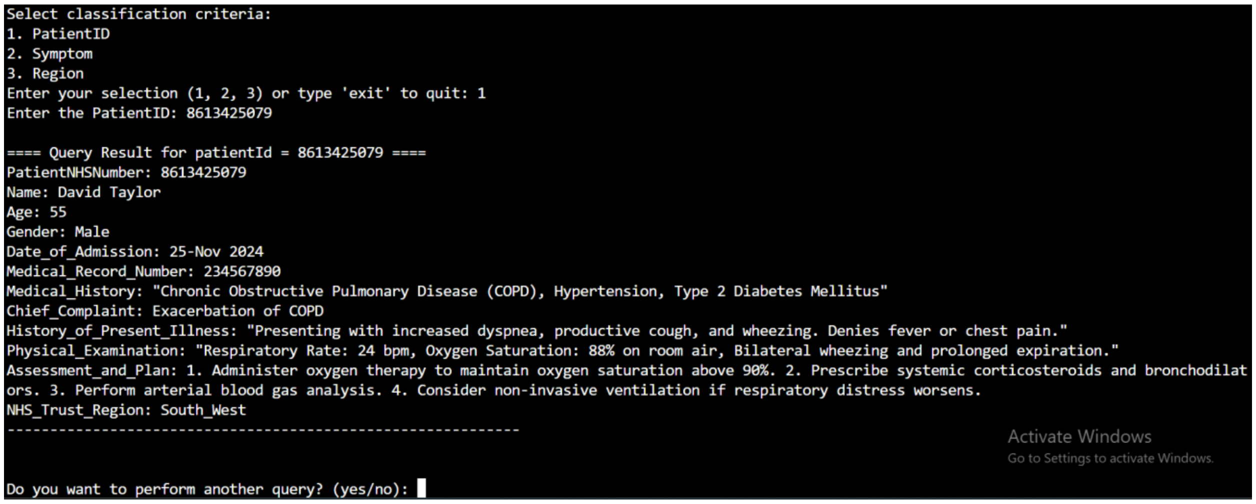
2. Interactive MapReduce Query Phase Results

2.1 Querying by NHS Number

When the user chooses to search by NHS number, the program goes into an interactive mode. It uses a special part of the code called TokenizerMapper, in the "patientId" section.

- **How It Works:** The program reads each line from the CSV file. It checks the first field (the NHS number) to see if it matches the number the user typed. It ignores upper and lower case, so it works even if the letters are not the same case.

- **What Happens Next:** If the NHS number in the file matches the user's number, that full record is saved and sent to the next step. The Reducer then organizes the information and shows it in a clear format with labels for each field (like name, age, symptoms etc.). This helps the user easily see the patient's details and check if the record is correct.



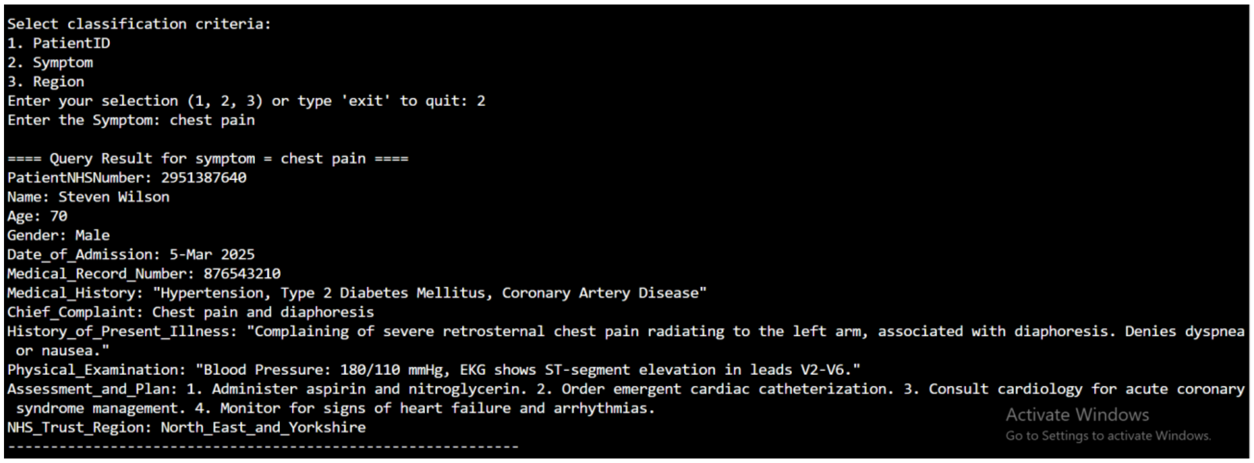
```
Select classification criteria:
1. PatientID
2. Symptom
3. Region
Enter your selection (1, 2, 3) or type 'exit' to quit: 1
Enter the PatientID: 8613425079

==== Query Result for patientId = 8613425079 ====
PatientNHSNumber: 8613425079
Name: David Taylor
Age: 55
Gender: Male
Date_of_Admission: 25-Nov 2024
Medical_Record_Number: 234567890
Medical_History: "Chronic Obstructive Pulmonary Disease (COPD), Hypertension, Type 2 Diabetes Mellitus"
Chief_Complaint: Exacerbation of COPD
History_of_Present_Illness: "Presenting with increased dyspnea, productive cough, and wheezing. Denies fever or chest pain."
Physical_Examination: "Respiratory Rate: 24 bpm, Oxygen Saturation: 88% on room air, Bilateral wheezing and prolonged expiration."
Assessment_and_Plan: 1. Administer oxygen therapy to maintain oxygen saturation above 90%. 2. Prescribe systemic corticosteroids and bronchodilat
ors. 3. Perform arterial blood gas analysis. 4. Consider non-invasive ventilation if respiratory distress worsens.
NHS_Trust_Region: South_West
------------------------------------------------------------
                                                                                                   Activate Windows
                                                                                                   Go to Settings to activate Windows.

Do you want to perform another query? (yes/no):
```

2.2 Querying by Symptom

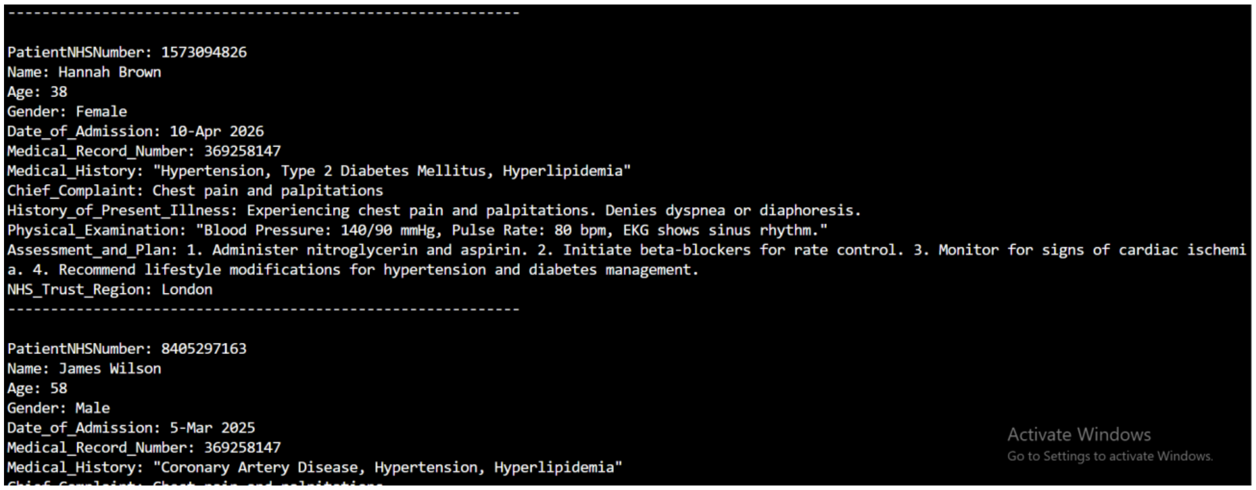
For questions about symptoms, the system follows a similar process:

How it works: In the "symptom" part, the Mapper looks at the "Chief_Complaint" field. It removes any extra quotation marks and changes all the text to lowercase. This helps the system ignore differences in letter case. Then, it checks if the symptom the user typed is found anywhere in that field.

What happens next: If it finds a match, the system saves that record. The Reducer then organizes the results. This method makes sure that symptoms like "fever" are found, no matter if they are written as "Fever", "feVer" or "FEVER".



```
Select classification criteria:
1. PatientID
2. Symptom
3. Region
Enter your selection (1, 2, 3) or type 'exit' to quit: 2
Enter the Symptom: chest pain

==== Query Result for symptom = chest pain ====
PatientNHSNumber: 2951387640
Name: Steven Wilson
Age: 70
Gender: Male
Date_of_Admission: 5-Mar 2025
Medical_Record_Number: 876543210
Medical_History: "Hypertension, Type 2 Diabetes Mellitus, Coronary Artery Disease"
Chief_Complaint: Chest pain and diaphoresis
History_of_Present_Illness: "Complaining of severe retrosternal chest pain radiating to the left arm, associated with diaphoresis. Denies dyspnea
 or nausea."
Physical_Examination: "Blood Pressure: 180/110 mmHg, EKG shows ST-segment elevation in leads V2-V6."
Assessment_and_Plan: 1. Administer aspirin and nitroglycerin. 2. Order emergent cardiac catheterization. 3. Consult cardiology for acute coronary
 syndrome management. 4. Monitor for signs of heart failure and arrhythmias.
NHS_Trust_Region: North_East_and_Yorkshire
------------------------------------------------------------
```



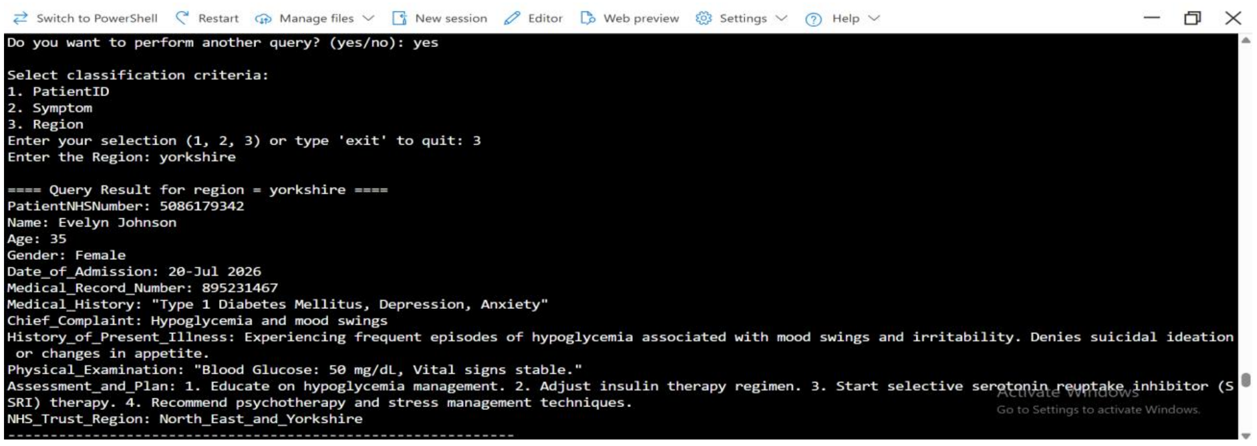
```
------------------------------------------------------------

PatientNHSNumber: 1573094826
Name: Hannah Brown
Age: 38
Gender: Female
Date_of_Admission: 10-Apr 2026
Medical_Record_Number: 369258147
Medical_History: "Hypertension, Type 2 Diabetes Mellitus, Hyperlipidemia"
Chief_Complaint: Chest pain and palpitations
History_of_Present_Illness: Experiencing chest pain and palpitations. Denies dyspnea or diaphoresis.
Physical_Examination: "Blood Pressure: 140/90 mmHg, Pulse Rate: 80 bpm, EKG shows sinus rhythm."
Assessment_and_Plan: 1. Administer nitroglycerin and aspirin. 2. Initiate beta-blockers for rate control. 3. Monitor for signs of cardiac ischemi
a. 4. Recommend lifestyle modifications for hypertension and diabetes management.
NHS_Trust_Region: London
------------------------------------------------------------

PatientNHSNumber: 8405297163
Name: James Wilson
Age: 58
Gender: Male
Date_of_Admission: 5-Mar 2025
Medical_Record_Number: 369258147
Medical_History: "Coronary Artery Disease, Hypertension, Hyperlipidemia"
```

2.3 Querying by Region

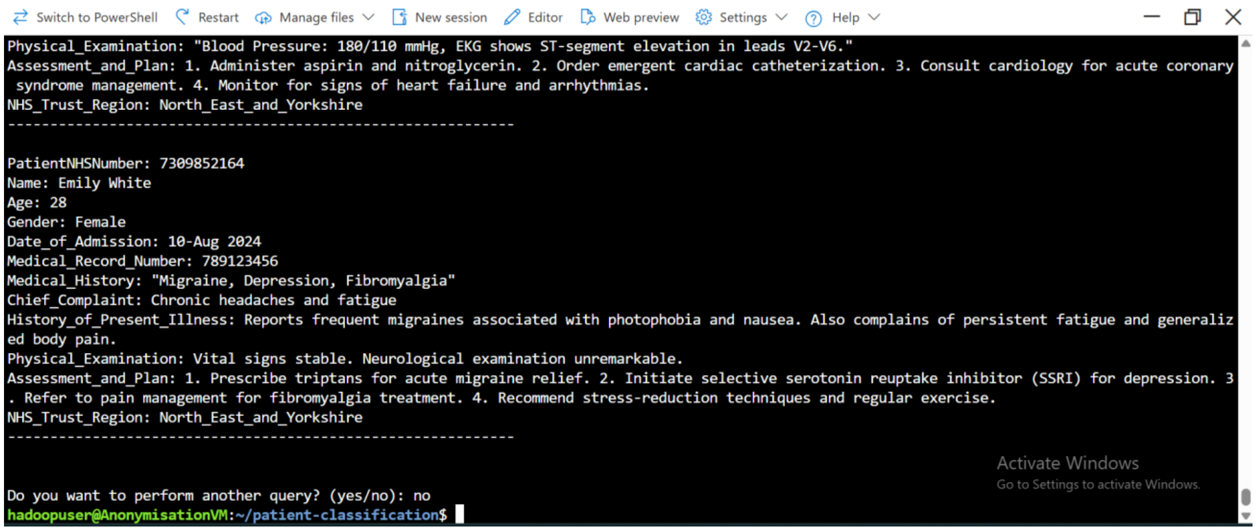
For questions about regions, the system adds an extra step to make sure it works well:

How it works: In the "region" part of the Mapper, the system changes both the region in the record and the user's input to the same format. It replaces underscores (_) with spaces and changes all letters to lowercase. This helps the system treat things like “South_West” and “south west” as the same.

What happens next: If the cleaned-up region in the record includes the user’s input, that record is saved. The Reducer then puts these results together, removes any repeated records and adds clear titles before showing them to the user.



```
Do you want to perform another query? (yes/no): yes

Select classification criteria:
1. PatientID
2. Symptom
3. Region
Enter your selection (1, 2, 3) or type 'exit' to quit: 3
Enter the Region: yorkshire

==== Query Result for region = yorkshire ====
PatientNHSNumber: 5086179342
Name: Evelyn Johnson
Age: 35
Gender: Female
Date_of_Admission: 20-Jul 2026
Medical_Record_Number: 895231467
Medical_History: "Type 1 Diabetes Mellitus, Depression, Anxiety"
Chief_Complaint: Hypoglycemia and mood swings
History_of_Present_Illness: Experiencing frequent episodes of hypoglycemia associated with mood swings and irritability. Denies suicidal ideation
 or changes in appetite.
Physical_Examination: "Blood Glucose: 50 mg/dL, Vital signs stable."
Assessment_and_Plan: 1. Educate on hypoglycemia management. 2. Adjust insulin therapy regimen. 3. Start selective serotonin reuptake inhibitor (S
SRI) therapy. 4. Recommend psychotherapy and stress management techniques.
NHS_Trust_Region: North_East_and_Yorkshire
------------------------------------------------------------
```



```
Physical_Examination: "Blood Pressure: 180/110 mmHg, EKG shows ST-segment elevation in leads V2-V6."
Assessment_and_Plan: 1. Administer aspirin and nitroglycerin. 2. Order emergent cardiac catheterization. 3. Consult cardiology for acute coronary
 syndrome management. 4. Monitor for signs of heart failure and arrhythmias.
NHS_Trust_Region: North_East_and_Yorkshire
------------------------------------------------------------

PatientNHSNumber: 7309852164
Name: Emily White
Age: 28
Gender: Female
Date_of_Admission: 10-Aug 2024
Medical_Record_Number: 789123456
Medical_History: "Migraine, Depression, Fibromyalgia"
Chief_Complaint: Chronic headaches and fatigue
History_of_Present_Illness: Reports frequent migraines associated with photophobia and nausea. Also complains of persistent fatigue and generaliz
ed body pain.
Physical_Examination: Vital signs stable. Neurological examination unremarkable.
Assessment_and_Plan: 1. Prescribe triptans for acute migraine relief. 2. Initiate selective serotonin reuptake inhibitor (SSRI) for depression. 3
. Refer to pain management for fibromyalgia treatment. 4. Recommend stress-reduction techniques and regular exercise.
NHS_Trust_Region: North_East_and_Yorkshire
------------------------------------------------------------

Do you want to perform another query? (yes/no): no
hadoopuser@AnonymisationVM:~/patient-classification$
```

3. Handling Case Insensitivity and Data Normalization

The system is designed to handle different types of input problems:

Ignoring capital letters: The system changes all text (like symptoms and regions) to lowercase. It also changes the user's input the same way. This makes sure that it doesn't matter if the words have capital letters or not.

Fixing underscores: For region searches, the system changes underscores (_) into spaces. This helps match regions even if they are written in different ways, like “South_West” or “south west”.

Making sure the data is correct: The system uses helper functions to check that every record has the correct number of fields. Even if the input file has mistaken (like extra commas or broken dates), the system fixes it so there are always 12 fields in each record.

4. **Output Formatting and Error Handling**

Formatting the results: The Reducer makes each matching record easy to read. It shows each field with a clear label, like “PatientNHSNumber:” or “Name:”, followed by the correct value. A line is added between records to keep them separate and neat. This makes the results easier to understand on the screen.

Handling errors: If there are no matches for a search, the system tells the user clearly. It also checks bad or empty records and skips them.

Potential Applications

The patient record classification system is made to read, organize and search through large amounts of patient data. It can be very useful for working with healthcare data in a fast and smart way. Here are some important ways it can be used:

1. Healthcare Data Integration and Management

This application is great for hospitals and healthcare centers that need to manage a lot of patient records. It reads CSV files from HDFS and creates three different output files, one for NHS numbers, one for symptoms and one for trust regions. This helps bring different types of data together in a simple and organized way. It can be used for:

• **Centralized Patient Databases:** Collecting patient records from different places into one clear and complete database.

• **Data Consistency:** Making sure all records follow the same format, which is important when combining data from different departments or old systems (Shaban-Nejad, Michalowski and Buckeridge, 2018).

2. Real-Time Clinical Data Analysis

The system uses an interactive MapReduce query phase, which helps it answer questions quickly. You can search by patient ID, symptom or region. This fast search feature is useful for:

• **Clinical Decision Support:** Doctors and healthcare staff can quickly find patient records during emergencies or regular check-ups. This helps them make faster and better decisions.

• **Epidemiological Surveillance:** Health teams can use region-based data to watch for disease trends or outbreaks. They can even find early signs by matching part of the region name (Rose et al., 2019).

3. Quality Control and Error Detection

In the preprocessing step, the system not only changes the format of the data but also checks if each record has the right number of fields. This helps in the following ways:

• **Audit and Compliance:** It can find and mark records that are incomplete or broken, which helps keep data quality high.

• **Data Cleansing:** It removes mistakes in the data before using it for analysis. This makes sure that the results from the data are correct and trustworthy (Han, Pei and Kamber, 2011).

4. Big Data and Distributed Processing

This system runs on Hadoop YARN, which helps manage resources and keeps things running even if some parts fail. Because of this, it works well when handling large amounts of data. It is useful for:

• **Large-Scale Medical Record Analysis:** Hospitals and clinics in different areas can use it to study patient records together. This makes it a good choice for national or regional healthcare systems.

• **Integration with Cloud-Based Analytics:** When used with cloud platforms like Microsoft Azure, the system can use big storage and fast computers to study large datasets without slowing down or crashing (Hashem et al., 2015).

5. Research and Epidemiological Studies

The system gives clean and organized data and lets users search in different ways. This makes it a strong tool for research. It can be used for:

• **Medical Research:** Researchers can easily find data about certain symptoms or regions. This helps in looking at past cases or building models to predict future health problems.

• **Public Health Studies:** Data scientists can use the results to study how diseases spread, how treatments work or how healthy a population is. This helps in making better health plans and decisions (Ristevski & Chen, 2018).

6. Clinical Decision Support Systems (CDSS)

Adding this classification system to larger clinical decision support tools can help doctors take a more active role in patient care. It can be used for:

• **Preprocessing for AI Models:** The cleaned and organized data, especially when sorted by symptoms or regions, can be used by machine learning models. This can help predict how patients will do, suggest diagnoses or decide which treatments to focus on first.

• **Better Access to Data:** The system makes the data easy to read, which is useful for creating dashboards or user-friendly interfaces in hospitals. This helps doctors and staff quickly understand patient information (Sutton et al., 2020).

Conclusion

In conclusion, this project successfully creates a system for classifying and searching patient records using a distributed approach. The system works in two main parts: First, the preprocessing phase reads a CSV file from HDFS, fixes data problems by parsing and normalizing it and creates three organized output files. The second part uses an interactive MapReduce query phase, which allows users to quickly search for records based on patient ID, symptoms or regions. It also handles errors well and shows results in a clear format. By running the system on a YARN cluster in Microsoft Azure, the project shows how distributed computing can handle large amounts of healthcare data. It also sets the stage for future uses in clinical decision support, research and public health studies. This approach makes the system ready for modern healthcare data analysis, helping to integrate data and provide useful insights in a world with a lot of big data.

References:

- Shaban-Nejad, A., Michalowski, M. and Buckeridge, D.L., 2018. Health intelligence: how artificial intelligence transforms population and personalized health. *NPJ Digital Medicine*, 1, p.53. [online] Available at: https://doi.org/10.1038/s41746-018-0058-9.
- Rose, E., Rube, S., Kanter, A.S., Cardwell, M. and Naeymi-Rad, F., 2019. Integration of postcoordination content into a clinical interface terminology to support administrative coding. *Journal of the American Medical Informatics Association*, [online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6344336/
- Han, J., Pei, J. and Kamber, M., 2011. *Data mining: concepts and techniques*. 3rd ed. Burlington, MA: Morgan Kaufmann.
- **Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A. and Khan, S.U., 2015.** The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47, pp.98-115. [DOI: https://doi.org/10.1016/j.is.2014.07.006]
- Ristevski, B. and Chen, M., 2018. *Big data analytics in medicine and healthcare*. Journal of Integrative Bioinformatics, 15(3), pp.1-15. [DOI: https://doi.org/10.1515/jib-2017-0030]

- Sutton, R.T., Pincock, D., Baumgart, D.C., Sadowski, D.C., Fedorak, R.N. and Kroeker, K.I., 2020. *An overview of clinical decision support systems: benefits, risks, and strategies for success*. NPJ Digital Medicine, 3(1), pp.1-10. [DOI: https://doi.org/10.1038/s41746-020-0221-y]