Yelp Data Analysis - Phase I

Aayushi Pandey
SEAS
University at Buffalo
Buffalo, NY, USA
pandey25@buffalo.edu

Om Nankar
SEAS
University at Buffalo
Buffalo, NY, USA
omnankar@buffalo.edu

Sushrut Gaikwad
SEAS
University at Buffalo
Buffalo, NY, USA
sushruti@buffalo.edu

***Abstract*—This project focuses on building a big data pipeline using the Yelp Open Dataset, which contains data about millions of reviews, business records, users, tips, and checkins. In the first phase, we set up a Hadoop cluster and implement a Java-based script to ingest the data. After ingestion, we carried out exploratory data analysis using Python and pandas to uncover patterns in the data. We also identify three potential machine learning use cases that can be solved using this data: sentiment analysis of reviews, personalized recommendation system, and fake review detection. Additionally, we defined five data analysis objectives to guide our exploratory work. These early steps help lay the foundation for deeper modeling and insights in later phases of the project.**

***Index Terms*—Hadoop, Yelp Dataset, Data Ingestion, Exploratory Data Analysis, Machine Learning**

I. Introduction

Yelp is a popular online application that publishes crowd-sourced reviews about businesses. Users can leave reviews, ratings, and tips about local businesses, making it a rich source of large real-world data. Analyzing this data can reveal insights about user preferences, business trends, regional patterns, and more. However, working with such a large dataset calls for tools and systems that can handle scale.

Our project aims to build a complete end-to-end big data pipeline to process and analyze the Yelp Open Dataset. In this first phase, we focused on setting up the infrastructure and preparing the data for analysis. We created a Hadoop cluster on our local machine and wrote a Java-based script to ingest the data into the Hadoop Distributed File System (HDFS). In parallel, we used Python and the pandas library to carry out exploratory data analysis (EDA) to get familiar with the structure of the data and uncover the basic trends.

In addition, we also outlined three potential machine learning problems: review sentiment classification, recommendation systems, and fake review detection. We also defined five data analysis objectives to study user behavior, business popularity, and geographic patterns. These tasks gave us a better understanding of the dataset and also set the direction for future phases, where we plan to implement the machine learning methods and advanced analytics.

II. Dataset Description

As mentioned earlier, we are using the Yelp Open Dataset, a publicly available collection of user generated data shared by Yelp for academic purposes. It is a rich source of real-world data involving users, businesses, reviews, and more. It is especially suitable for building end-to-end big data pipelines and building machine learning models due to its size, structure, and variety.

A. Individual Datasets

The data is in the form of JSON files. The five main JSON files are the following:

- `business.json`: It contains information about each business, such as name, location (city, state, latitude, longitude), star rating, number of reviews, categories, and various business attributes (e.g., parking availability, takeout options).
- `review.json`: It includes full user-written reviews with associated star ratings, review dates, and metadata like the number of "useful", "funny", and "cool" votes each review received.
- `user.json`: It provides metadata about users including number of reviews written, average star rating, number of fans, compliments received, and a list of friends (as user IDs).
- `checkin.json`: It records check-in activity on businesses, with timestamp data.
- `tip.json`: It is similar to reviews but shorter in length, these are quick suggestions or comments written by users for businesses.

Each file is stored in JSON format with one object per line. Such files can be easily read using pandas by using the `read_json` method. The entire data is relatively large in size with millions of rows. Further, the data is also relational. We can use columns like `user_id` and `business_id` like keys to connect different JSON files.

Note that the `review.json` file was too large for us to load in our memory. Hence we used a sample of the first 1,000,000 lines from this file and did EDA on that.

III. Exploratory Data Analysis

Some insights that we have gathered using EDA are the following:

A. Business Data

The `business.json` file contains metadata more than 150,000 businesses, including name, location, rating, and operational status.

- **Geographical Insights:** Most businesses are located in Philadelphia, Pennsylvania, with over 1,400 unique cities and 27 states represented. Latitude and longitude distributions revealed distinct city clusters on the map.
- **Rating Distribution:** Star ratings are concentrated around 4.0, and most businesses are still open (about 80%).
- **Review Count:** The `review_count` column showed an extremely heavy right skew, indicating a few highly-reviewed businesses and a long tail of lesser-reviewed ones.
- **Missing Values:** The `attributes` and `hours` columns had 9% and 15.5% missing values respectively, while `categories` had very few missing entries.
- **Multivariate Analysis:** Scatterplots and barplots suggested that businesses with more reviews tended to have slightly higher average ratings, but business status (open/closed) did not have a strong influence on average rating.

B. Check-in Data

The `checkin.json` file contains timestamped records of customer visits to businesses. Each record corresponds to a business and includes a string of check-in timestamps.

- **Coverage:** Check-in data is available for approximately 38,000 unique businesses, indicating a subset of all businesses in the dataset.
- **Total Check-ins:** By parsing the timestamp strings, we found the total number of check-ins to be approximately 1.9 million.
- **Data Quality:** The dataset contains no missing values, which makes it reliable for further analysis.
- **Potential Use:** Check-ins can be used in future work to analyze business popularity trends over time, identify peak business hours, or track seasonal patterns in customer engagement.

C. Tip Data

The `tip.json` file contains short, user-written comments about businesses. Unlike reviews, tips are typically brief suggestions or remarks and are often written in a more informal style.

- **Content and Structure:** Each tip includes a `user_id`, `business_id`, `text`, `date`, and a `compliment_count`, which indicates how many users complimented the tip.
- **Volume and Activity:** The number of tips steadily increased between 2011 and 2019, with the highest volume in 2012. Monthly tip volume appears to be evenly distributed, indicating consistent usage throughout the year.
- **User and Business Engagement:** The most active tip givers and the most "tipped" businesses were identified, showing a similar skew to that of reviews—a small number of users contribute disproportionately.
- **Compliment Analysis:** The majority of tips received zero compliments, indicating that while tips are useful, they do not attract as much engagement as full reviews.

D. User Data

The `user.json` file provides rich metadata about individual Yelp users, including review counts, average ratings, compliment types, and friend connections. This data is essential for understanding user engagement and behavior on the platform.

- **User Growth Over Time:** The number of users steadily increased between 2011 and 2018, peaking in 2015. This trend reflects Yelp's growing popularity in that time period.
- **Review Counts:** The distribution of user review counts is highly right-skewed—most users have written only a few reviews, while a small number of users are extremely active.
- **Average Ratings:** Users tend to give high ratings, with the distribution of average stars centered around 4.0.
- **Social Network:** The majority of users have zero friends on the platform. This suggests that while Yelp includes social features, many users interact independently.
- **Compliment Analysis:** Compliment columns such as `compliment_hot`, `compliment_funny`, and `compliment_writer` are all highly correlated, suggesting that active or well-received users tend to be complimented in multiple ways.
- **Top Users:** We also identified the top-10 most complimented users and broke down their compliment types.

Instructions on how to run our EDA notebook is given in the Appendix, i.e., Section VII-E.

IV. System Setup

The provided `docker-compose.yml` file is used for the Hadoop cluster setup and the steps are followed using the `README.md` file. The main requirements are to have Docker and Maven installed on the system, the installation check for which can be verified using the two commands:

```
docker --version
# Output: Docker version 27.3.1, build ce12230
```

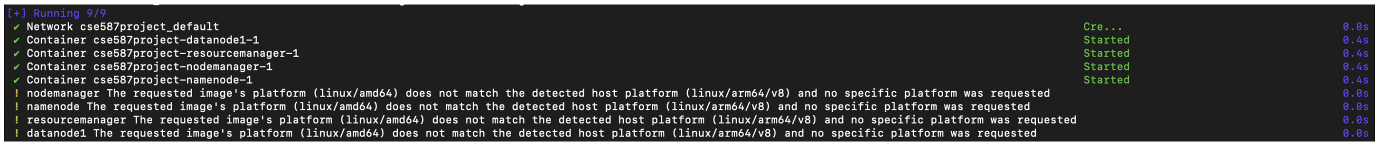
```
hadoop version
# Output: 3.4.1
```

Next, the following command starts the local Hadoop cluster inside Docker.

```
docker compose -f docker-compose.yaml up -d
```

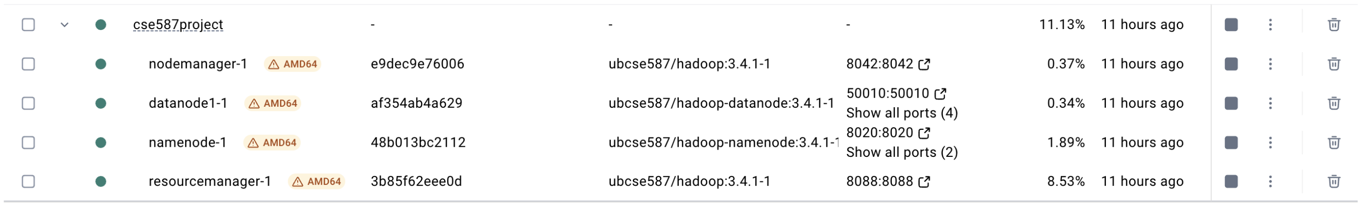
Also, the NameNode, DataNode, ResourceManager, and NodeManager start in Docker. Their creation and verification of whether they are running properly can be seen from the screenshot shown in Figure 1. The Docker desktop application also shows the cluster. Its screenshot is shown in Figure 2.

Next, to test the system setup, we add a `README.txt` file which has the following content:



```
[+] Running 9/9
 ✔ Network cse587project_default                                                                                   Cre...                        0.0s
 ✔ Container cse587project-datanode1-1                                                                             Started                       0.4s
 ✔ Container cse587project-resourcemanager-1                                                                       Started                       0.4s
 ✔ Container cse587project-nodemanager-1                                                                           Started                       0.4s
 ✔ Container cse587project-namenode-1                                                                              Started                       0.4s
 ! nodemanager The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested       0.0s
 ! namenode The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested          0.0s
 ! resourcemanager The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested   0.0s
 ! datanode1 The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested         0.0s
```

Fig. 1. Creation of NameNode, DataNode, ResourceManager, and NodeManager.



●	cse587project	-	-	-	11.13%	11 hours ago
●	nodemanager-1 ⚠ AMD64	e9dec9e76006	ubcse587/hadoop:3.4.1-1	8042:8042	0.37%	11 hours ago
●	datanode1-1 ⚠ AMD64	af354ab4a629	ubcse587/hadoop-datanode:3.4.1-1	50010:50010 Show all ports (4)	0.34%	11 hours ago
●	namenode-1 ⚠ AMD64	48b013bc2112	ubcse587/hadoop-namenode:3.4.1-1	8020:8020 Show all ports (2)	1.89%	11 hours ago
●	resourcemanager-1 ⚠ AMD64	3b85f62eee0d	ubcse587/hadoop:3.4.1-1	8088:8088	8.53%	11 hours ago

Fig. 2. Creation of NameNode, DataNode, ResourceManager, and NodeManager as shown by the Docker desktop application The green dot indicates that the containers are running properly.

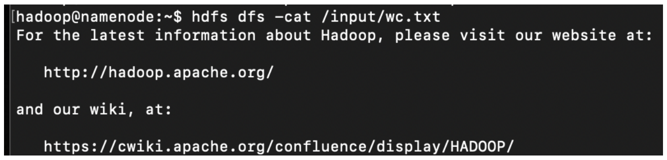
```
For the latest information about Hadoop, please visit our
↪  website at:
    http://hadoop.apache.org/
and our wiki, at:
    https://cwiki.apache.org/confluence/display/HADOOP/
```

This content is written to the NameNode using the following commands:

```
docker exec -it project-namenode-1 bash
hdfs dfs -mkdir /input
hdfs dfs -put README.txt /input/wc.txt
```

The first command is to get in the NameNode, the second is to create a directory in the NameNode, and the third command creates a file `wc.txt` with the content of `README.txt`.

The creation of the files and directories can be seen on `http://localhost:9870`. The creation of the file can also be verified inside the terminal, as is shown in Figure 3.



```
hadoop@namenode:~$ hdfs dfs -cat /input/wc.txt
For the latest information about Hadoop, please visit our website at:

    http://hadoop.apache.org/

and our wiki, at:

    https://cwiki.apache.org/confluence/display/HADOOP/
```

Fig. 3. File creation shown in the terminal.

This confirms that the setup is successful and the Hadoop cluster is running.

V. Data Ingestion

Now, we will discuss how the Yelp dataset (JSON files) are ingested into the Hadoop cluster that was setup in the previous section.

A. *Hadoop Installation and Verification*

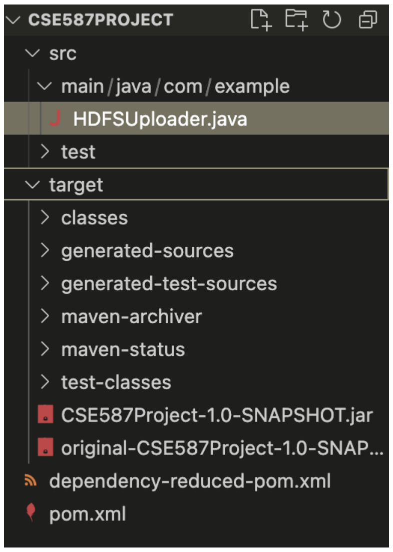
Before beginning data ingestion, Hadoop was installed and its installation was verified using the following command:

```
hadoop version
# Output: 3.4.1
```

This command checks if Hadoop is correctly installed and configured by displaying the installed version details.

B. *Maven Project Setup*

A Maven project was created with the structure shown in Figure 4. The `pom.xml` file contains the necessary dependencies for Hadoop, Woodstox (XML parser), and the Maven Shade Plugin for packaging the project as a JAR file. The dependencies include:



```
CSE587PROJECT
  src
    main / java / com / example
      HDFSUploader.java
    test
  target
    classes
    generated-sources
    generated-test-sources
    maven-archiver
    maven-status
    test-classes
    CSE587Project-1.0-SNAPSHOT.jar
    original-CSE587Project-1.0-SNAP...
  dependency-reduced-pom.xml
  pom.xml
```

Fig. 4. Maven project structure.

- `hadoop-common`
- `hadoop-hdfs`
- `hadoop-client`
- `hadoop-mapreduce-client-core`
- `hadoop-yarn-client`
- `woodstox-core`

The content of the `pom.xml` file is the following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
↪  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
↪  http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>CSE587Project</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>CSE587Project</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <!-- Hadoop Common -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-common</artifactId>
      <version>3.4.1</version>
      <!-- Ensure this matches your Hadoop version -->
    </dependency>
    <!-- Hadoop HDFS -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-hdfs</artifactId>
      <version>3.4.1</version>
    </dependency>
    <!-- Hadoop Client -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-client</artifactId>
      <version>3.4.1</version>
    </dependency>
    <!-- Woodstox Core (StAX XML parser) -->
    <dependency>
      <groupId>com.fasterxml.woodstox</groupId>
      <artifactId>woodstox-core</artifactId>
      <version>6.4.0</version>
    </dependency>
    <!-- Hadoop MapReduce Client (if needed) -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-mapreduce-client-core</artifactId>
      <version>3.4.1</version>
    </dependency>
    <!-- Hadoop Yarn Client (if needed) -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-yarn-client</artifactId>
      <version>3.4.1</version>
    </dependency>
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-hdfs-client</artifactId>
      <version>3.4.1</version>
      <!-- Ensure it matches your Hadoop version -->
    </dependency>
    <!-- Hadoop Client API -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-client-api</artifactId>
      <version>3.4.1</version>
    </dependency>
    <!-- Hadoop Client Runtime -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-client-runtime</artifactId>
      <version>3.4.1</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>3.2.4</version>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>shade</goal>
            </goals>
            <configuration>
              <filters>
                <filter>
                  <artifact>*:*</artifact>
                  <excludes>
                    <exclude>META-INF/*.SF</exclude>
                    <exclude>META-INF/*.DSA</exclude>
                    <exclude>META-INF/*.RSA</exclude>
                  </excludes>
                </filter>
              </filters>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

C. Java Script for HDFS Upload

The `HDFSUploader.java` script uploads the local Yelp JSON files to HDFS. The script does the following:

1) Establishes a connection to HDFS.
2) Ensures that the destination directory exists.
3) Iterates over the local dataset directory to find the JSON files.
4) Checks the existence of the file to avoid redundancy.
5) Copies the JSON files to HDFS.

The content of the `HDFSUploader.java` is the following:

```java
package com.example;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.FileStatus;
import java.io.File;
import java.io.IOException;
import java.net.URI;

public class HDFSUploader {
    public static void main(String[] args) {
        // Local directory containing Yelp JSON files
        //String localDir =
        ↪  "/Users/omnankar/Documents/Spring sem/CSE
        ↪  587/CSE587project/Yelp JSON/yelp_dataset/";
        String localDir = "/tmp/yelp_dataset/";
        String hdfsDestination = "/yelp_dataset/";

        try {
            // Load Hadoop Configuration
            Configuration conf = new Configuration();
            conf.set("fs.defaultFS",
            ↪  "hdfs://namenode:8020"); // Or use the
            ↪  actual Namenode IP
            conf.set("fs.hdfs.impl",
            ↪  "org.apache.hadoop.hdfs.DistributedFileSystem");
            conf.set("fs.file.impl",
            ↪  "org.apache.hadoop.fs.LocalFileSystem");

            // Initialize HDFS FileSystem
            FileSystem fs = FileSystem.get(new
            ↪  URI("hdfs://namenode:8020"), conf);

            // Ensure destination directory exists
            Path hdfsDestPath = new Path(hdfsDestination);
            if (!fs.exists(hdfsDestPath)) {
                fs.mkdirs(hdfsDestPath);
                System.out.println("Created HDFS directory:
                ↪  " + hdfsDestination);
            }

            // Get list of files to upload
            File folder = new File(localDir);
            File[] listOfFiles = folder.listFiles((dir,
            ↪  name) - > name.endsWith(".json"));
            if (listOfFiles == null || listOfFiles.length ==
            ↪  0) {
                System.out.println("No JSON files found in
                ↪  local directory: " + localDir);
                return;
            }

            // Upload each file
            for (File file: listOfFiles) {
                Path srcPath = new
                ↪  Path(file.getAbsolutePath());
                Path destPath = new Path(hdfsDestination +
                ↪  file.getName());

                // Check if file already exists in HDFS
                if (fs.exists(destPath)) {
                    System.out.println("Skipping (already
                    ↪  exists): " + file.getName());
                    continue;
                }

                // Upload file
                fs.copyFromLocalFile(srcPath, destPath);
                System.out.println("Uploaded: " +
                ↪  file.getName());
            }

            fs.close();
            System.out.println("All files uploaded
            ↪  successfully!");

        } catch (Exception e) {
            System.err.println("Error uploading files to
            ↪  HDFS: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

D. Execution Steps

1) Copy Yelp Dataset To NameNode Container:

```
docker cp "/local/path/to/yelp_dataset"
↪  cse587project-namenode-1:/tmp/yelp_dataset
```

This copies the dataset from the local system into the Docker container where the NameNode is running.

2) Build the Maven Project:

```
mvn clean package
```

This compiles the Java code and packages it as a JAR file. The JAR file (`CSE587Project-1.0-SNAPSHOT.jar`) contains the compiled Java code for the script which was run. It is responsible for uploading Yelp dataset JSON files to HDFS.

3) Copy JAR File to NameNode Container:

```
docker cp target/CSE587Project-1.0-SNAPSHOT.jar
↪  cse587project-namenode-1:/tmp/
```

This is shown in Figure 5. This moves the compiled JAR file into the NameNode container for execution.

4) Run HDFSUploader Inside NameNode Container:

```
docker exec -it cse587project-namenode-1 bash -c 'java -cp
↪  /tmp/CSE587Project-1.0-SNAPSHOT.jar
↪  com.example.HDFSUploader'
```

This is shown in Figure 6. This executes the `HDFSUploader` Java program inside the Hadoop container to upload data to HDFS.

5) Verify Uploaded Files:

```
docker exec -it cse587project-namenode-1 bash -c 'hdfs dfs
↪  -ls /yelp_dataset'
```

This is shown in Figure 7.

E. Web Interface Verification

The uploaded files were also verified using the Hadoop NameNode Web UI at `http://localhost:9870`, where the `yelp_dataset` directory and its JSON files are visible. This is shown in Figures 8 and 9.

Another check is to read a few lines from the uploaded data. The first 5 lines of the `business.json` file are read using the following command:

```
docker exec -it cse587project-namenode-1 bash -c 'hdfs dfs
↪  -cat /yelp_dataset/yelp_academic_dataset_business.json |
↪  head -n 5'
```

Figure 10 shows the output.

F. Discussion on Data Ingestion Methods

1) Is `hdfs dfs -put` a good way to write files to an HDFS cluster?: Writing with `hdfs dfs -put` is a good way to write files into an HDFS cluster for small- to medium-sized files and one time or spaced uploads. It is simple, well-proven, and requires less commands. But if massive data ingestion or bulk uploading is required, as in the case of uploading the Yelp dataset (more than 9 GB) it may not be ideal due to its single-threaded approach and lack of advanced features like compression or multi-threaded uploading.

2) How to improve `hdfs dfs -put` speed?: The speed on `hdfs dfs -put` while writing `README.txt` file to input directory was satisfactory and quick but for data ingestion it will be very slow. Some ways of improving it include:

- Use the `-p` parameter for parallel copying.
- Increase the replication factor temporarily during upload (default is 3).
- Implement a custom solution using the HDFS API for parallel uploads.

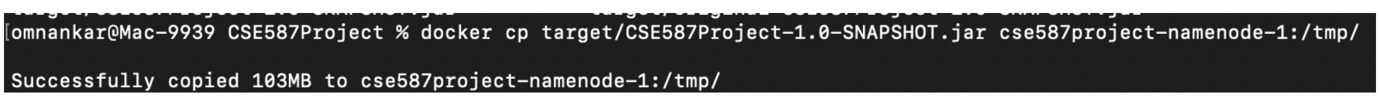
3) Best Data Formats for HDFS Storage: Some formats to store data in HDFS for easy analysis afterwards are:

- *Parquet:* Columnar storage file format that uses effective compression and encoding schemes, often used when Excel or CSV file reach their row limit.
- *JSON:* For semi-structured data, especially when human readability is essential.

The choice of format primarily depends on the specific analysis needs, query patterns, and the tools which will be used to analyze the data.

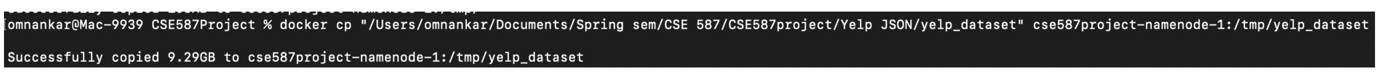
VI. Machine Learning Problem Statements

Based on our exploration of the Yelp Open Dataset, we came up with three machine learning problems that not only fit well with the data but also address real-world use cases. These problems aim to improve user experience and enhance the trustworthiness of the platform.



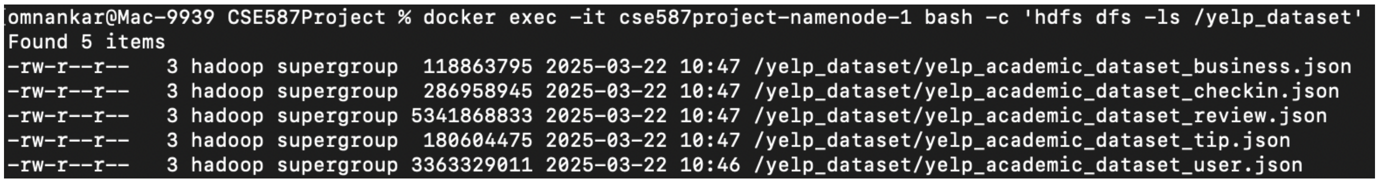
```
[omnankar@Mac-9939 CSE587Project % docker cp target/CSE587Project-1.0-SNAPSHOT.jar cse587project-namenode-1:/tmp/

 Successfully copied 103MB to cse587project-namenode-1:/tmp/
```

Fig. 5. Copying JAR file to NameNode container.



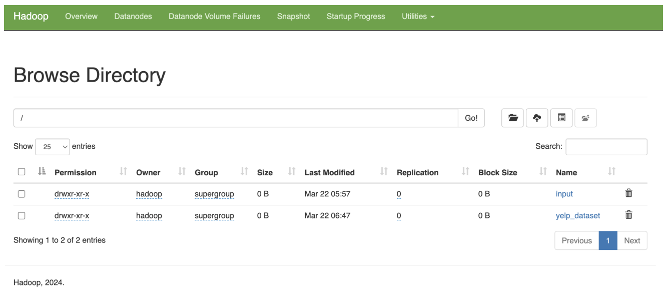
```
[omnankar@Mac-9939 CSE587Project % docker cp "/Users/omnankar/Documents/Spring sem/CSE 587/CSE587project/Yelp JSON/yelp_dataset" cse587project-namenode-1:/tmp/yelp_dataset

Successfully copied 9.29GB to cse587project-namenode-1:/tmp/yelp_dataset
```

Fig. 6. Running HDFSUploader inside NameNode Container.



```
[omnankar@Mac-9939 CSE587Project % docker exec -it cse587project-namenode-1 bash -c 'hdfs dfs -ls /yelp_dataset'
Found 5 items
-rw-r--r--   3 hadoop supergroup  118863795 2025-03-22 10:47 /yelp_dataset/yelp_academic_dataset_business.json
-rw-r--r--   3 hadoop supergroup  286958945 2025-03-22 10:47 /yelp_dataset/yelp_academic_dataset_checkin.json
-rw-r--r--   3 hadoop supergroup 5341868833 2025-03-22 10:47 /yelp_dataset/yelp_academic_dataset_review.json
-rw-r--r--   3 hadoop supergroup  180604475 2025-03-22 10:47 /yelp_dataset/yelp_academic_dataset_tip.json
-rw-r--r--   3 hadoop supergroup 3363329011 2025-03-22 10:46 /yelp_dataset/yelp_academic_dataset_user.json
```

Fig. 7. Verification of uploaded files.



Browse Directory

/

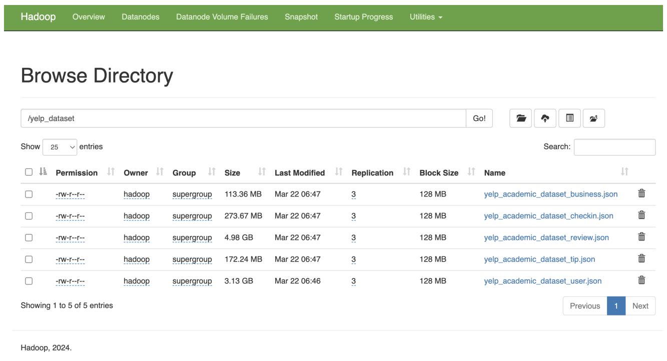
Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Mar 22 05:57	0	0 B	input
drwxr-xr-x	hadoop	supergroup	0 B	Mar 22 06:47	0	0 B	yelp_dataset

Showing 1 to 2 of 2 entries

Hadoop, 2024.

Fig. 8. Web interface verification of the root directory.



Browse Directory

/yelp_dataset

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	113.36 MB	Mar 22 06:47	3	128 MB	yelp_academic_dataset_business.json
-rw-r--r--	hadoop	supergroup	273.67 MB	Mar 22 06:47	3	128 MB	yelp_academic_dataset_checkin.json
-rw-r--r--	hadoop	supergroup	4.98 GB	Mar 22 06:47	3	128 MB	yelp_academic_dataset_review.json
-rw-r--r--	hadoop	supergroup	172.24 MB	Mar 22 06:47	3	128 MB	yelp_academic_dataset_tip.json
-rw-r--r--	hadoop	supergroup	3.13 GB	Mar 22 06:46	3	128 MB	yelp_academic_dataset_user.json

Showing 1 to 5 of 5 entries

Hadoop, 2024.

Fig. 9. Web interface verification of the `yelp_dataset` directory.

A. *Sentiment Analysis of User Reviews*

The goal here is to predict the sentiment of a review—whether it is positive, neutral, or negative—based on the review text. While the dataset already includes star ratings, this task helps extract deeper insight from the actual text users write. Sometimes, the sentiment expressed in the review does not match the star rating, so this can give businesses a clearer picture of how customers feel. This is a classic text classification problem that can be solved using supervised learning, where the review text is used as input and the rating (or derived sentiment label) is the target.

B. *Business Recommendation System*

We aim to build a recommendation system that suggests businesses to users based on their past activity—mainly the reviews they have written and the ratings they have given. This could help users discover new places they have likely to enjoy, making the platform more useful and personalized. We can approach this using collaborative filtering (based on user-item interactions), content-based filtering (using business attributes), or a hybrid of both.

C. *Detecting Fake or Spam Reviews*

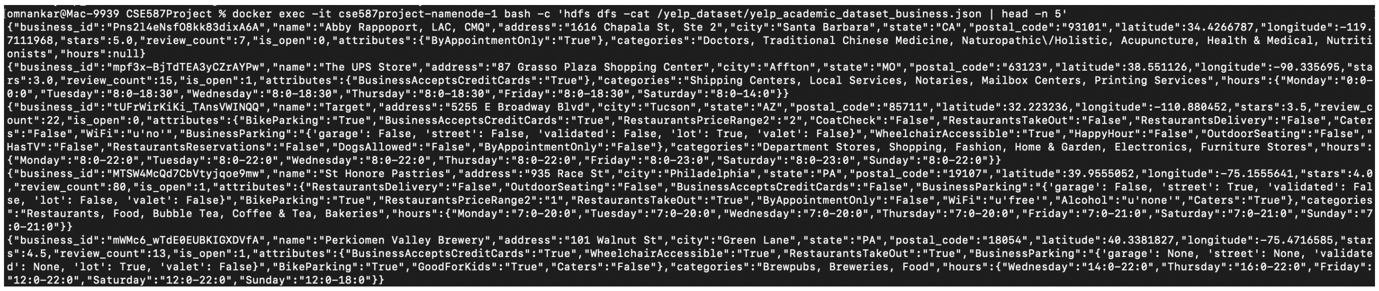
Another problem we identified is the detection of fake or suspicious reviews. Some reviews may be overly positive or negative in a way that does not align with a user's history or the overall business reputation. Spotting these can improve the credibility of the platform and help users trust the information they see. This can be treated as an anomaly detection or binary classification problem, depending on how we define and label fake reviews. We can use features like review text, timing, voting patterns, and user behavior to train a model that flags potentially fake reviews.

VII. Data Analysis Objectives

We have also defined five key objectives that will guide our deeper exploration of the Yelp Open Dataset. These objectives are designed to help us uncover trends, patterns, and relationships that can add value to both users and businesses on the platform. The insights gained from these analyses will also help support and validate our proposed machine learning tasks.

A. *User Trend Analysis*

We plan to study how the user base on Yelp has grown and evolved over time. By analyzing the `yelping_since` field in the `user.json` file, we will track the number of



```
omnankar@Mac-9939 CSE587Project % docker exec -it cse587project-namenode-1 bash -c 'hdfs dfs -cat /yelp_dataset/yelp_academic_dataset_business.json | head -n 5'
{"business_id":"Pns2l4eNsfO8kk83dixA6A","name":"Abby Rappoport, LAC, CMQ","address":"1616 Chapala St, Ste 2","city":"Santa Barbara","state":"CA","postal_code":"93101","latitude":34.4266787,"longitude":-119.
7111968,"stars":5.0,"review_count":7,"is_open":0,"attributes":{"ByAppointmentOnly":"True"},"categories":"Doctors, Traditional Chinese Medicine, Naturopathic\/Holistic, Acupuncture, Health & Medical, Nutriti
onists","hours":null}
{"business_id":"mpf3x-BjTdTEA3yCZrAYPw","name":"The UPS Store","address":"87 Grasso Plaza Shopping Center","city":"Affton","state":"MO","postal_code":"63123","latitude":38.551126,"longitude":-90.335695,"sta
rs":3.0,"review_count":15,"is_open":1,"attributes":{"BusinessAcceptsCreditCards":"True"},"categories":"Shipping Centers, Local Services, Notaries, Mailbox Centers, Printing Services","hours":{"Monday":"0:0-
0:0","Tuesday":"8:0-18:30","Wednesday":"8:0-18:30","Thursday":"8:0-18:30","Friday":"8:0-18:30","Saturday":"8:0-14:0"}}
{"business_id":"tUFrWirKiKi_TAnsVWINQQ","name":"Target","address":"5255 E Broadway Blvd","city":"Tucson","state":"AZ","postal_code":"85711","latitude":32.223236,"longitude":-110.880452,"stars":3.5,"review_c
ount":22,"is_open":0,"attributes":{"BikeParking":"True","BusinessAcceptsCreditCards":"True","RestaurantsPriceRange2":"2","CoatCheck":"False","RestaurantsTakeOut":"False","RestaurantsDelivery":"False","Cater
s":"False","WiFi":"u'no'","BusinessParking":"{'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet': False}","WheelchairAccessible":"True","HappyHour":"False","OutdoorSeating":"False","
HasTV":"False","RestaurantsReservations":"False","DogsAllowed":"False","ByAppointmentOnly":"False"},"categories":"Department Stores, Shopping, Fashion, Home & Garden, Electronics, Furniture Stores","hours":
{"Monday":"8:0-22:0","Tuesday":"8:0-22:0","Wednesday":"8:0-22:0","Thursday":"8:0-22:0","Friday":"8:0-23:0","Saturday":"8:0-23:0","Sunday":"8:0-22:0"}}
{"business_id":"MTSW4McQd7CbVtyjqoe9mw","name":"St Honore Pastries","address":"935 Race St","city":"Philadelphia","state":"PA","postal_code":"19107","latitude":39.9555052,"longitude":-75.1555641,"stars":4.0
,"review_count":80,"is_open":1,"attributes":{"RestaurantsDelivery":"False","OutdoorSeating":"False","BusinessAcceptsCreditCards":"False","BusinessParking":"{'garage': False, 'street': True, 'validated': Fal
se, 'lot': False, 'valet': False}","BikeParking":"True","RestaurantsPriceRange2":"1","RestaurantsTakeOut":"True","ByAppointmentOnly":"False","WiFi":"u'free'","Alcohol":"u'none'","Caters":"True"},"categories
":"Restaurants, Food, Bubble Tea, Coffee & Tea, Bakeries","hours":{"Monday":"7:0-20:0","Tuesday":"7:0-20:0","Wednesday":"7:0-20:0","Thursday":"7:0-20:0","Friday":"7:0-21:0","Saturday":"7:0-21:0","Sunday":"7
:0-21:0"}}
{"business_id":"mWMc6_wTdE0EUBKIGXDVfA","name":"Perkiomen Valley Brewery","address":"101 Walnut St","city":"Green Lane","state":"PA","postal_code":"18054","latitude":40.3381827,"longitude":-75.4716585,"star
s":4.5,"review_count":13,"is_open":1,"attributes":{"BusinessAcceptsCreditCards":"True","WheelchairAccessible":"True","RestaurantsTakeOut":"True","BusinessParking":"{'garage': None, 'street': None, 'validate
d': None, 'lot': True, 'valet': False}","BikeParking":"True","GoodForKids":"True","Caters":"False"},"categories":"Brewpubs, Breweries, Food","hours":{"Wednesday":"14:0-22:0","Thursday":"16:0-22:0","Friday":
"12:0-22:0","Saturday":"12:0-22:0","Sunday":"12:0-18:0"}}
```

Fig. 10. Verification by printing the first 5 lines of the `business.json` file.

new users joining the platform each year. This will help us identify growth patterns and understand how user engagement has shifted across different time periods.

B. Business Category Performance

We aim to evaluate how different categories of businesses perform in terms of user engagement and satisfaction. Using data from `business.json`, including the `categories`, `review_count`, and `stars` fields, we will identify the most reviewed categories and compare their average ratings. This analysis will help us understand which types of businesses tend to attract more user attention and how they are perceived by customers.

C. User Engagement Patterns

Another key objective is to investigate how frequently users contribute to the platform and how that engagement influences their perceived credibility. We will analyze `review_count`, `average_stars`, and compliment-related metrics from the `user.json` file, alongside voting patterns from `review.json`. This will allow us to distinguish between casual and highly active users and assess their influence within the community.

D. Geographic Distribution of Businesses

We also plan to explore the geographic distribution of Yelp activity. By visualizing the latitude and longitude coordinates in the `business.json` file, we will identify which cities or regions have the highest concentration of reviewed businesses. This spatial analysis will give us a clearer picture of Yelp's reach and user engagement across different locations.

E. Relationship Between Review Length and Rating

Finally, we want to examine whether there is a connection between the length of a review and the sentiment it conveys. In this analysis, we will compare the character count of reviews (`text` field) with the corresponding star ratings from the `review.json` file. This will help us understand if longer reviews tend to reflect stronger opinions—either positive or negative—which can be useful for both sentiment analysis and review quality assessment.

Appendix: Running the EDA Notebook

We have created an Anaconda environment for EDA. The `environment.yml` file for our environment is provided in our submission. Please follow these steps to run our EDA notebook:

1) Using Anaconda prompt, navigate into the root directory (i.e., the directory you get after unzipping our submission).
2) Create an environment using our `environment.yml` file by running the following commands in the Anaconda prompt:

```
conda create -p ./venv --file environment.yml
```

3) The environment will be created in this root directory, inside the folder `venv`. Once this is done, activate this environment by running the following command:

```
conda activate ./venv
```

4) Now, run the following command to open a Jupyter notebook server:

```
jupyter notebook
```

Once this server opens, you can click on our submitted EDA notebook to open it.