Machine Learning Big Data Assignment

Aritra Das Ray s423160

November 2023

Contents

1 Introduction

1.1 Aim

The Aim of this project is to process a Covid-19 data-set using PySpark and implement supervised and unsupervised machine learning algorithms.

1.2 Objective

In order to achieve the aim, we will be carrying out the following tasks:

- Setup the PySpark environment
- Import necessary PySpark libraries
- Read Data Using PySpark
- Process the data and make it suitable for the tasks
- Build PySpark Queries to derive the relevant information

2 Methodology

2.1 Theory

For our project we will be using using Apache Spark through a Python API called PySpark. We will also be doing performing Linear regression and K-Means Clustering, which are supervised and unsupervised machine Learning Algorithms on the data set.

Apache Spark
Apache Spark is a fast, general-purpose, unified engine for large-scale data processing and analytics. It is a leading platform for large scale SQL/ETL operations, batch data processing, streaming data processing and implementing Machine Learning. It has in-memory data storage for faster data processing and is approximately 10x faster than Hadoop-MapReduce. It provides high-level APIs in Java, Scala, Python and R and is also compatible with Hadoop Storage APIs i.e. it can read/write any Hadoop-Supported Systems.[1] [2]

Pyspark
PySpark is the Python API for Apache Spark. It enables us to perform real-time, large-scale data processing in a distributed environment using Python. PySpark combines Python's ease of use with the power of Apache Spark to enable processing and analysis of data at any size for everyone familiar with Python. PySpark supports all of Spark's features such as Spark SQL, DataFrames, Structured Streaming, Machine Learning (MLlib) and Spark Core.[3]

Supervised and Unsupervised Machine Learning

Supervised Learning is a type of Machine Learning Algorithm in which the Algorithm is trained with a labeled dataset, where the input data is paired with corresponding output labels. The goal is for the model to learn a mapping from inputs to outputs, making predictions or classifications on new, unseen data.[4] In Contrast to this, Unsupervised learning deals with unlabeled data, where the algorithm tries to find patterns, relationships, or structure within the data without explicit guidance on the output. The goal is often to discover hidden patterns, group similar data points, or reduce the dimensionality of the data.[5]

Linear Regression

Linear regression, one of the most basic form of supervised Learning, is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. The most common method to perform linear regression is the least-squared method in which the goal is to find the best-fit line that minimizes the sum of the squared differences between the observed and predicted values. For this assignment we will be using the Linear Regression library already available with PySpark instead of developing my own implementation of linear Regression Code.[3]

K-Means Clustering

K-means clustering is unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping subsets (clusters). The algorithm aims to group data points that are similar to each other and assigns them to clusters based on feature similarity.[7] The algorithm for K-Means Clustering is as follows:

1. Initialization: Choose K initial cluster centroids randomly.
2. Assignment: Assign each data point to the cluster whose centroid is closest.
3. Update Centroids: Recalculate the centroids of the clusters based on the mean of the data points in each cluster.
4. Repeat: Repeat steps 2 and 3 until convergence (when assignments no longer change significantly).

2.2 Environment Setup

Setting up the PySpark Environment has been extremely challenging. I have tried setting up the PySpark Docker environment on my Windows System, however for computationally expensive queries and code the system kept crashing. I have also tried setting up Spark locally in my Windows System however there seems to be an incompatibility with Windows and the latest version of PySpark.

Eventually I settled with setting up PySpark in a Virtual environment inside Anaconda Distribution. To Successfully setup the PySpark environment, I followed the following steps:

1. Download and install the Anaconda Distribution
2. Create New environment for Pyspark
3. Install PySpark packages inside the environment. This Installs all the default python packages
4. Install OpenJDK package inside the environment for Java Dependencies
5. Install findspark using the Anaconda prompt.

   ```
   conda install -c conda-forge findspark
   ```

   This is required to help Jupyter notebooks to find the PySpark packages.
6. Once all the packages are installed, Open Jupyter Notebooks and initialize the find spark with the below code.

   ```
   import findspark
   findspark.init()
   findspark.find()
   ```

7. Now the PySpark environment is successfully setup in Jupyter Notebook.

2.3 Data preprocessing

We are dealing with a time series data on the number of confirmed cases of covid-19 infected people. The first 4 columns have geographical information - state, country, latitude and longitude. These Act as column indexes. The remaining columns have the no of confirmed cases with the date as a header. While this format of data helps in transferring the data over the web as CSV, working with the data and deriving meaningful information out of is not easy.

Therefore we need to transform the data and create a dataframe with which we can write PySpark Query much more efficiently. To this effect, we used the following steps:

1. **Column Renamed:** We changed the column names from "Province/State", "Country/Region" to "State" and "Country". This was done to remove "/", which is problematic because we try to pass in a column as a variable in a method the "/" throws an error. To resolve this and for the sake of simplifying the process, it was decided to rename the column.

2. **Columns Dropped:** The Latitude and Longitude Columns in the dataframe were dropped because these columns were not being used in the code for any of the questions. There was an attempt made to use the Latitude and Longitude values to determine the continent, However this did not materialize, because of the complexity of the logic involved.

3. **Rows Dropped:** There were 4 locations that were also being tracked in the dataset, namely "Diamond Princess", "MS Zaandam", "Summer Olympics 2020" and "Winter Olympics 2022". While "Diamond Princess" and "MS Zaandam" are Ships, while "Summer Olympics 2020" and "Winter Olympics 2022" are events. Since these Locations are not countries or regions and did not correspond to any continent. This would pose a problem in the 2nd Question hence it was decided to drop the 4 rows.

4. **Pivot Date and Cumulative Increase** To make the data more useful, it was evident that we need one column with all the dates and one column with all the values. To achieve this we used the **stack()** method available in PySpark.

5. **Column No of Days** Date datatype cannot be vectorized in PySpark. Instead we derive the Number of days elapsed as a column to be vectorized and input as a feature in the Machine Learning algorithms. To achieve this we implement a query to calculate no of days.

6. **Column Daily Increase** Additionally the data show daily cumulative cases of Covid 19. Using this does not give a detailed idea of the spread of disease spread. Hence a new column is derived where by subtracting the previous day value from the column value.

7. **Column Zone** Creating the Zone Column which stores the "Region" values, however if "Region" is null, then it takes the "Country" values.

2.4 Code Architecture

Three different notebooks were used to answer each of the three Questions, keeping in mind that answer to each question would require a different dataframe. However, all the code will have similar code elements as listed below:

1. **Initiate findspark** This connects the Jupyter Notebook to spark engine.
2. **Import Required Python and PySpark Libraries** We import necessary Python Libraries like date-time, time and PySpark Libraries like SQL.SparkSession, SQL.functions etc. to build code.
3. **Initialize Spark Session** This is necessary to build the PySpark code and dataframe
4. **Import Data** The CSV file is imported and the data is read automatically into a dataframe with inferred schema.
5. **Build Data Processing Pipeline** Raw data is processed and made Consumable
6. **Implementing Machine Learning Algorithms** This step is only applicable for Questions 2 and 3.
7. **Implementing the results query and Export Results** A final Query is implemented to answer the questions. The result is then exported into a CSV.

The time module in Python is used to determine the performance of the entire Code. Start time is noted down since the beginning of the Spark session and the end time is noted down once the final result is obtained. The total time to execute the code is obtained by subtracting the start time from the end time. The CSV file export is left out of the calculations.

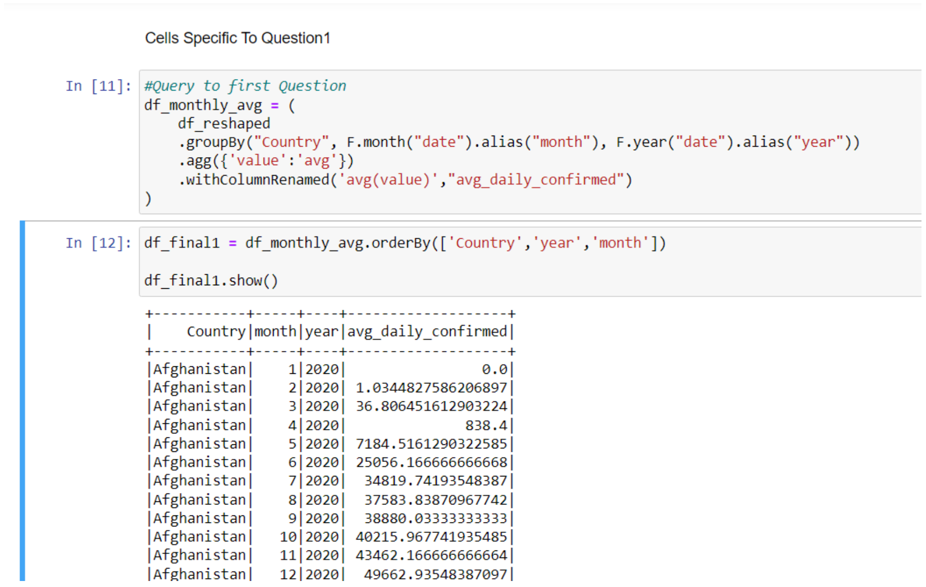
2.5 Dataset

The data was downloaded from the GitHub Repository as directed in the assignment. The database has been archived on 10th March 2023. Hence the dataset has Covid data up to 9th March 2023. The entire dataset, ranging from 22nd January 2020 to 9th March 2023 has been used in the Assignment.

3 Question 1

The interpretation of the first Question is that it asks us to find the average daily cumulative confirmed cases per month per Country. To do that we group by the total number of cumulative confirmed cases by Country, Year and Month. Then we use the average aggregate function to calculate the mean of each groupings.

The results of the query is stored in the CSV file and shared along with code submissions.



Cells Specific To Question1

In [11]:
```
#Query to first Question
df_monthly_avg = (
    df_reshaped
    .groupBy("Country", F.month("date").alias("month"), F.year("date").alias("year"))
    .agg({'value':'avg'})
    .withColumnRenamed('avg(value)',"avg_daily_confirmed")
)
```

In [12]:
```
df_final1 = df_monthly_avg.orderBy(['Country','year','month'])

df_final1.show()
```

```
+-----------+-----+----+-------------------+
|    Country|month|year|avg_daily_confirmed|
+-----------+-----+----+-------------------+
|Afghanistan|    1|2020|                0.0|
|Afghanistan|    2|2020| 1.0344827586206897|
|Afghanistan|    3|2020| 36.806451612903224|
|Afghanistan|    4|2020|              838.4|
|Afghanistan|    5|2020| 7184.5161290322585|
|Afghanistan|    6|2020| 25056.166666666668|
|Afghanistan|    7|2020|  34819.74193548387|
|Afghanistan|    8|2020|  37583.83870967742|
|Afghanistan|    9|2020|  38880.03333333333|
|Afghanistan|   10|2020| 40215.967741935485|
|Afghanistan|   11|2020| 43462.166666666664|
|Afghanistan|   12|2020|  49662.93548387097|
```

Figure 1: Question 1 Query and Output

4 Question 2

The 2nd Question has many parts to it and as a results has been decomposed into multiple steps post data processing:

1. We first create a list mapping the countries to their continents. The list is then converted to PySpark DataFrame. This dataFrame is left joined to the main DataFrame thereby mapping every country to their respective DataFrame.

2. Importing Machine Learning Libraries in PySpark and initializing the VectorAssembly and LinearRegression Objects.

3. Extracting distinct Zones from the Zones column.

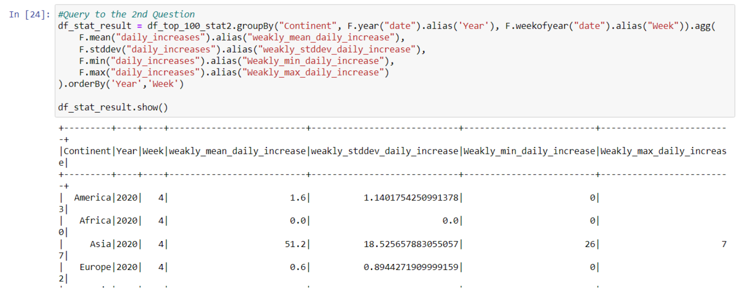
4. Iterating through Zones Column and implementing the Machine Learning Algorithm for each Zone. With this we derive the trendline coefficient for each zone and store it in a list.

5. At the end of the loop we get a list mapping the zone to the trendline coefficient. The list is converted to a dataframe and the top 100 zones are taken and inner joined with the main dataframe

6. The combined dataframe is ordered and except for Continent, date and daily increases all columns are dropped.

7. DataFrame is then grouped by Continent and date, while the daily increase is summed up.

8. Then the Query is to answer Question 2 is executed. In this Query we group the data by "Continent", "Year", "Week" and use the aggregate functions to calculate the results for each group.



```
In [24]: #Query to the 2nd Question
df_stat_result = df_top_100_stat2.groupBy("Continent", F.year("date").alias('Year'), F.weekofyear("date").alias("Week")).agg(
    F.mean("daily_increases").alias("weakly_mean_daily_increase"),
    F.stddev("daily_increases").alias("weakly_stddev_daily_increase"),
    F.min("daily_increases").alias("Weakly_min_daily_increase"),
    F.max("daily_increases").alias("Weakly_max_daily_increase")
).orderBy('Year','Week')

df_stat_result.show()
```

```
+---------+----+----+--------------------------+----------------------------+-------------------------+------------------------
-+
|Continent|Year|Week|weakly_mean_daily_increase|weakly_stddev_daily_increase|Weakly_min_daily_increase|Weakly_max_daily_increas
e|
+---------+----+----+--------------------------+----------------------------+-------------------------+------------------------
-+
|  America|2020|   4|                       1.6|          1.1401754250991378|                        0|
3|
|   Africa|2020|   4|                       0.0|                         0.0|                        0|
0|
|     Asia|2020|   4|                      51.2|          18.525657883055057|                       26|                       7
7|
|   Europe|2020|   4|                       0.6|          0.8944271909999159|                        0|
2|
```

Figure 2: Question 2 Query and Output

The results of the query is stored in the CSV file and shared along with code submissions.

5 Question 3

The Question 3 is uses similar data processing and Linear Regression code. However, the continent data is not appended to the DataFrame. Additionally, the K-Means clustering is a new feature of the code for Question 3.

To Answer Question 3 the following steps were followed post data processing:

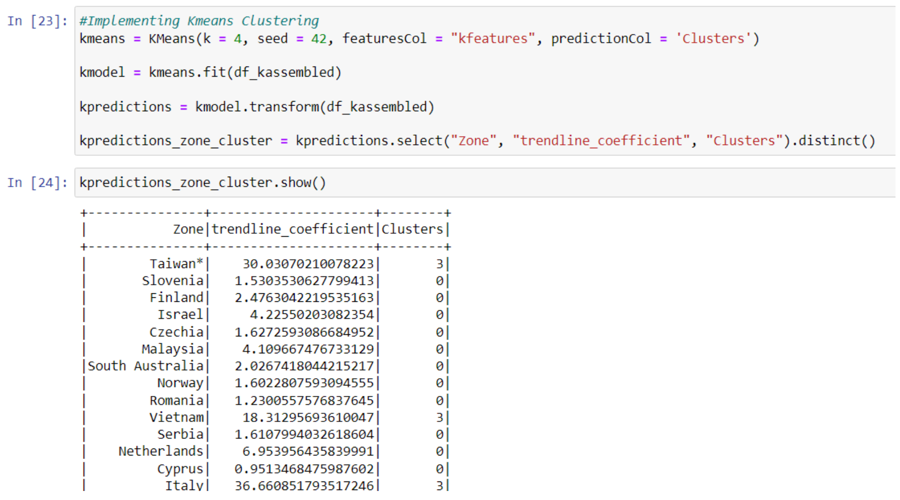
1. The Machine Learning Libraries were imported and the VectorAssembly and LinearRegression Objects were initialized.

2. The trendline Coefficient were calculated in the same manner as in Question number 2 and joined with the processed dataframe.

3. Then the top 50 zones with the highest trendline coefficient is filtered out in a dataframe.

4. This trendline Coefficient is again vectorized and served as input to the K-Means clustering Method.

5. The clustering Algorithm then outputs the cluster number for each of the 50 zones.



```
In [23]: #Implementing Kmeans Clustering
         kmeans = KMeans(k = 4, seed = 42, featuresCol = "kfeatures", predictionCol = 'Clusters')

         kmodel = kmeans.fit(df_kassembled)

         kpredictions = kmodel.transform(df_kassembled)

         kpredictions_zone_cluster = kpredictions.select("Zone", "trendline_coefficient", "Clusters").distinct()
```

```
In [24]: kpredictions_zone_cluster.show()
```

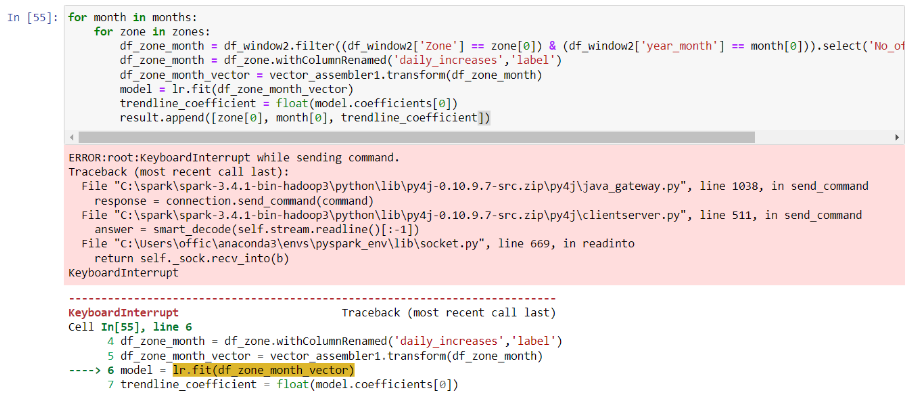
```
+--------------+---------------------+--------+
|          Zone|trendline_coefficient|Clusters|
+--------------+---------------------+--------+
|       Taiwan*|    30.03070210078223|       3|
|      Slovenia|   1.5303530627799413|       0|
|       Finland|   2.4763042219535163|       0|
|        Israel|     4.22550203082354|       0|
|       Czechia|   1.6272593086684952|       0|
|      Malaysia|    4.109667476733129|       0|
|South Australia|   2.0267418044215217|       0|
|        Norway|   1.6022807593094555|       0|
|       Romania|   1.2300557576837645|       0|
|       Vietnam|    18.31295693610047|       3|
|        Serbia|   1.6107994032618604|       0|
|   Netherlands|    6.953956435839991|       0|
|        Cyprus|   0.9513468475987602|       0|
|         Italy|   36.660851793517246|       3|
```

Figure 3: Question 3 Query and Output

The results of the query is stored in the CSV file and shared along with code submissions.

Note:-

The implementation of this Question was done with the entire dataset for the entire time period. However an attempt was made to implement a logic where the trendline coefficient be calculated per month per country but this proved to be computationally expensive. The code ran for 40 hours straight and ultimately had to be terminated before completion to prevent the overheating of the laptop.



```
In [55]: for month in months:
             for zone in zones:
                 df_zone_month = df_window2.filter((df_window2['Zone'] == zone[0]) & (df_window2['year_month'] == month[0])).select('No_of
                 df_zone_month = df_zone.withColumnRenamed('daily_increases','label')
                 df_zone_month_vector = vector_assembler1.transform(df_zone_month)
                 model = lr.fit(df_zone_month_vector)
                 trendline_coefficient = float(model.coefficients[0])
                 result.append([zone[0], month[0], trendline_coefficient])

ERROR:root:KeyboardInterrupt while sending command.
Traceback (most recent call last):
  File "C:\spark\spark-3.4.1-bin-hadoop3\python\lib\py4j-0.10.9.7-src.zip\py4j\java_gateway.py", line 1038, in send_command
    response = connection.send_command(command)
  File "C:\spark\spark-3.4.1-bin-hadoop3\python\lib\py4j-0.10.9.7-src.zip\py4j\clientserver.py", line 511, in send_command
    answer = smart_decode(self.stream.readline()[:-1])
  File "C:\Users\offic\anaconda3\envs\pyspark_env\lib\socket.py", line 669, in readinto
    return self._sock.recv_into(b)
KeyboardInterrupt

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[55], line 6
      4 df_zone_month = df_zone.withColumnRenamed('daily_increases','label')
      5 df_zone_month_vector = vector_assembler1.transform(df_zone_month)
----> 6 model = lr.fit(df_zone_month_vector)
      7 trendline_coefficient = float(model.coefficients[0])
```

Figure 4: Attempt at calculating trendline coefficient per month per zone

6 Results

As part of measuring the performance of the code, the time module was used to capture the system time at the start of the program and at the end of the program. Below table captures the performance of the Queries of each code:

Question	Start Time	End Time	Total Time(s)
One	1700429038.3589716	1700429087.102088	48.74311637878418
Two	1700395061.3030522	1700398859.233296	3797.930243730545
Three	1700402941.680053	1700403832.8686442	891.1885912418365

Table 1: Query Performance

From this we can conclude the machine learning modules took a longer duration to compute.

7 Ethical issues concerning Machine Learning and Big Data

While working with data, especially large volumes of data involving Analysis of Covid cases, a number of Ethical and Legal issues come up.
The foremost among them being the Availability of Quality data. Analysis of Covid and other health and disease data is often used by Governments and other important institutions to make important decisions. Therefore it is important that the data is of highest quality as possible.
Secondly many advanced Machine Learning Models such as deep neural networks, are often seen as black boxes, making it challenging to understand how they reach specific decisions. This lack of interpretability is a concern, especially in critical applications like healthcare.
Finally, handling large amounts of sensitive data especially PHI data raises security issues. Unauthorized access or breaches can lead to serious consequences and loss of privacy of all stakeholders. Therefore Data Security and Privacy is a major Ethical concern

References

[1] The Apache Software Foundation, Apache Spark Documentation [Internet]. Wilmington (Delaware, USA): Apache Spark; 2023 Nov 18[cited 2023 Nov 18]. Available from: https://spark.apache.org/docs/latest/#apache-spark-a-unified-engine-for-large-scale-data-analytics

[2] Dr. Jun Li.Apache Spark Lecture - Spark_1.pdf[Internet]. 2023 Oct 19 [cited 2023 Nov 18]. Cranfield University, Cranfield. Available from: https://canvas.cranfield.ac.uk/courses/27803/pages/lecture-4-apache-spark?module_item_id=501871

[3] The Apache Software Foundation, PySpark Documentation [Internet]. Wilmington (Delaware, USA): Apache Spark; 2023 Nov 18[cited 2023 Nov 18]. Available from: https://spark.apache.org/docs/latest/api/python/index.html

[4] Wikimedia Foundation, Inc. Supervised Learning[Internet]. San Francisco (California): Wikipedia; 2023 Oct 11[cited 2023 Nov 18]. Available from: https://en.wikipedia.org/wiki/Supervised_learning

[5] Wikimedia Foundation, Inc. Supervised Learning[Internet]. San Francisco (California): Wikipedia; 2023 Oct 26[cited 2023 Nov 18]. Available from: https://en.wikipedia.org/wiki/Unsupervised_learning

[6] Wikimedia Foundation, Inc. Linear Regression[Internet]. San Francisco (California): Wikipedia; 2023 Oct 26[cited 2023 Nov 18]. Available from: https://en.wikipedia.org/wiki/Linear_regression

[7] Wikimedia Foundation, Inc. Linear Regression[Internet]. San Francisco (California): Wikipedia; 2023 Oct 26[cited 2023 Nov 18]. Available from: https://en.wikipedia.org/wiki/K-means_clustering