Module 3 Practice lab

Module 3 - Exploring Apache Spark Through DataBricks/Google Colab

Harsh Patel

College of Professional Studies, Northeastern University Toronto

Prof. Mohammad Islam

Table of Contents

Introduction

The purpose of this lab was to apply practical big data techniques using **PySpark** within **Google Colab** to analyze the Boston Housing dataset. This project allowed me to practice loading, cleaning, transforming, and visualizing data using Spark's distributed processing capabilities. While the dataset itself is relatively small, the goal of the assignment was not only to explore the data but also to understand how big data tools such as Spark can be used in real-world data processing scenarios. This reflection discusses the steps I performed in this project, the insights gained, and how data visualization helps interpret Spark-processed data more effectively.

Data Loading and Initial Exploration

The first step in the project involved downloading the Boston Housing dataset and loading it into a Spark DataFrame. While loading the file, I initially faced path-related issues because the dataset was not located in the expected directory. After confirming the file location using shell commands in Colab, the CSV was successfully read using the spark.read.csv() function with schema inference enabled.

Once loaded, I displayed the **first five rows** using df.show(5) to confirm that the data was read correctly. To understand the overall data size, I used df.count(), which returned the number of observations in the dataset. I then used df.printSchema() to examine field names, data types, and ensure the dataset structure aligned with typical Boston Housing attributes.

These initial steps helped verify data quality and confirmed that Spark correctly recognized all numerical columns. Although the Boston dataset is small, using Spark provided a simulation of how distributed tools handle data ingestion and schema inspection at scale.

Data Cleaning and Transformation

The transformation phase included several specific tasks. First, I dropped the column **"b"**, as required in the assignment. This was done using the df.drop("b") function, which removed the column from the DataFrame and simplified the dataset for further analysis.

Next, I rounded all numeric columns to two decimal places. This step ensured consistency, especially because Spark often reads floating-point values with many decimal places. By applying the round() function across all numeric fields, the dataset became cleaner and easier to interpret.

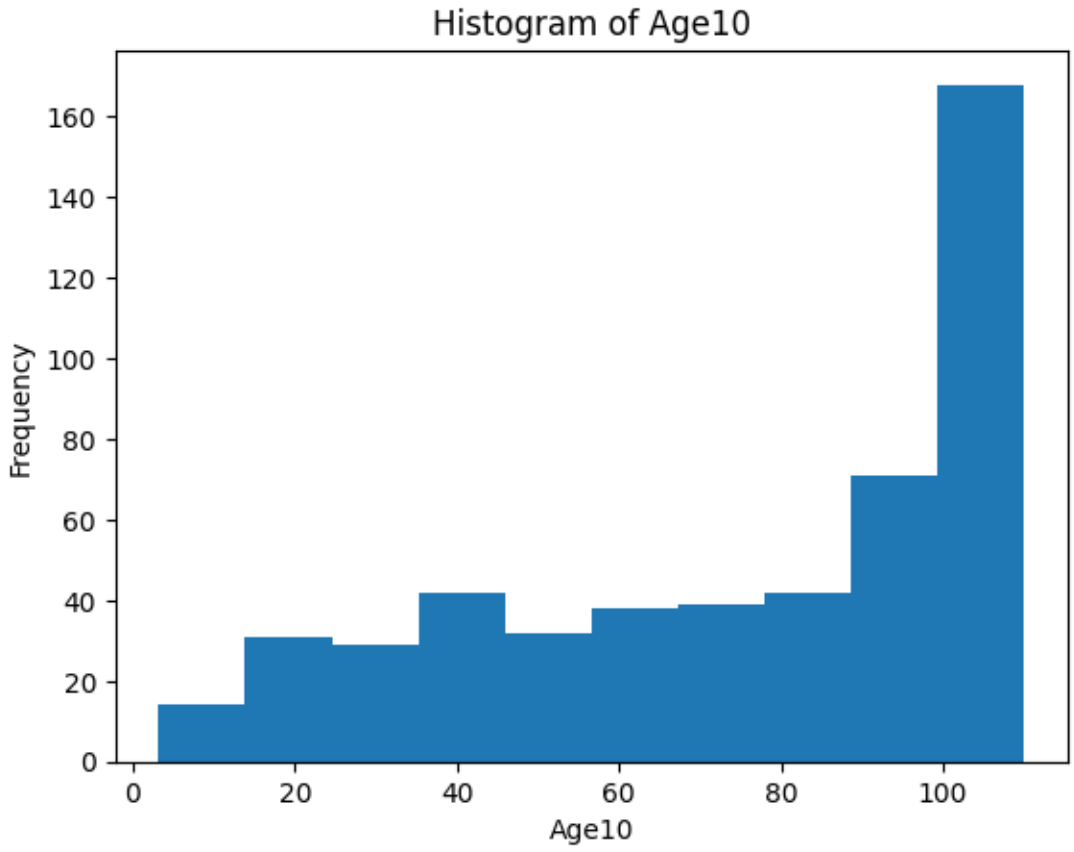
Another transformation involved creating a new feature named **Age10**, which represented a 10% increase in the values of the existing "age" column. This was performed using a simple arithmetic operation: col("age") * 1.10. Creating derived columns such as Age10 demonstrates how Spark can be used for feature engineering—an essential step in machine learning pipelines.

These transformations highlighted Spark's ability to perform column-wise operations efficiently, even though the dataset itself was small. In a real big data scenario, such transformations would allow analysts to manipulate millions of rows while maintaining high performance.

Visualization and Insights

Although Spark is powerful for computation, it is less suitable for direct visualization. Therefore, after transforming the dataset, I converted the Spark DataFrame into a Pandas DataFrame using df.toPandas(). This conversion enabled the use of Matplotlib for plotting.

One key visualization required in the lab was a **histogram of the Age10 column**. This 2D plot revealed the overall distribution of housing age values after the 10% increase. Histograms are especially useful for understanding skewness, modes, and spread of numeric features. In this case, the Age10 column reflected a distribution similar to the original age feature, indicating that only the scale—not the shape—was altered.



How Visualizing Spark-Processed Data Helps Understanding

Visualizing data after Spark processing makes patterns much easier to interpret. Spark excels at calculations, but visualizations bring those numbers to life. For example, trends such as skewed distributions, correlations, or outliers become more obvious when plotted. In this project, visualizing Age10 helped me immediately understand how the distribution of housing age changes with transformations. Without charts, these observations would not be as intuitive by looking at only numerical statistics.

Visualization acts as a bridge between large-scale computation and human interpretation. It turns Spark's processed output into something that can be visually and quickly understood. This is especially useful when explaining findings to stakeholders who may not be familiar with technical details.

Descriptive Statistics and Final Dataset Review

Spark's .describe() function provided summary statistics such as count, mean, standard deviation, minimum, and maximum values for all columns. These metrics are essential in verifying data consistency and understanding the main characteristics of the dataset.

For example, the summary statistics showed differences in scale between features such as crime rate, number of rooms, and median home values. Observing these variations is a key step before applying machine learning algorithms, particularly those sensitive to feature scaling.

Finally, after converting to Pandas, I displayed the **last five rows** using pdf.tail(5). This confirmed that the transformations, rounding, and new column creation were applied consistently throughout the dataset.

Overall Reflection

This project reinforced several important concepts about big data processing. First, even simple datasets benefit from the structure and reliability of Spark operations. Although the Boston Housing dataset is not large, performing transformations with Spark allowed me to experience workflows similar to real-world big data environments where datasets may contain millions of records.

Second, the project highlighted the usefulness of combining tools: Spark for computation and Pandas/Matplotlib for visualization. This hybrid workflow ensures the best of both worlds—efficient processing and intuitive visual understanding.

Lastly, the assignment improved my confidence in using PySpark commands for loading, cleaning, transforming, and analyzing data. I also learned how critical it is to verify file paths and schema definitions before analysis. Overall, this lab provided hands-on practice that strengthened my ability to handle data using distributed computing tools and present meaningful insights through visual and statistical analysis.

Conclusion

The Boston Housing lab provided a practical opportunity to apply big data skills using PySpark. Through data loading, cleaning, transformation, visualization, and interpretation, I gained valuable experience with Spark workflows inside Google Colab. The project emphasized the importance of understanding both the technical and analytical aspects of data processing. Most importantly, it demonstrated how visualization enhances comprehension of Spark-processed data and supports effective communication of findings. This lab helped build a strong foundation for future work in analytics, big data processing, and machine learning applications.