

**Nasrullah Rami Hamad**

**21110510**

**Big Data**

**Section 2**

**Dr. Balqis Al-dabaybeh**

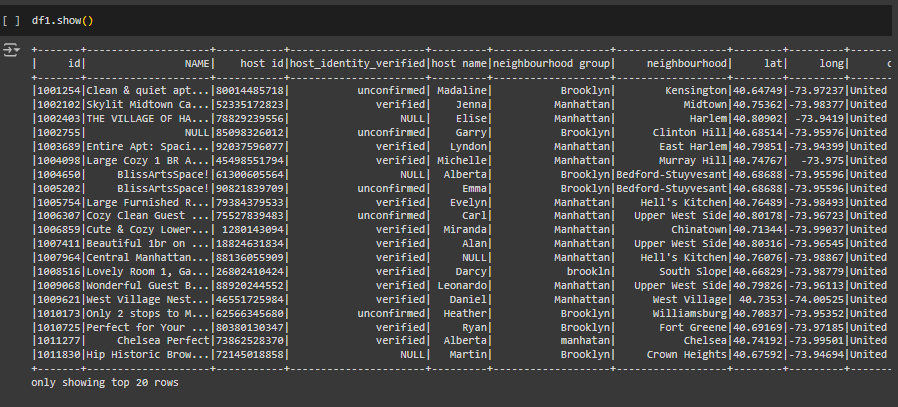
**Eng. Mohammad Al-Jarrah**

**Question 1.**

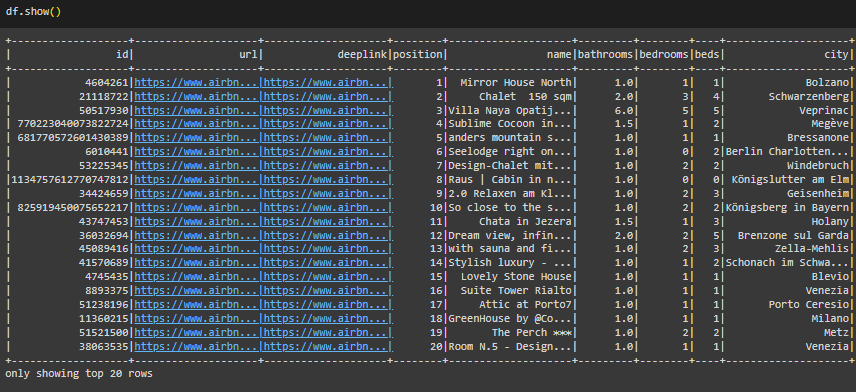
1. The data collection phase that took place, was done with two types of data. Csv (Comma Separated Values), and JSON (JavaScript Object Notation) datasets where we collected the CSV file from Kaggle, and the JSON file was from an API. The datasets studied Airbnb and how hotels and places were reserved by hosts for guests by some characteristics. The datasets were saved in a structured Spark data frame that contains rows (for instances) and columns (for characteristics). These steps were done in Apache Spark, which allows me to create a spark session and create my program inside it. I used a Spark Data frame to contain both the CSV and JSON datasets in order to prepare them and clean them for the modeling phase afterwards.

1. The data was prepared using Spark, and preprocessing techniques that contained some basic techniques that allowed good analysis for the data, and concluded with a machine learning model that predicted the rating category. The steps stated cleaning the dataset from irrelevant values, erasing the null values or replacing them with meaningful data, removing duplicated rows, feature extraction, dropping, and selection. As following:

**CSV FILE Preprocessing:**



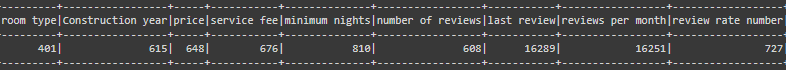
This (Show) function showed the first 20 rows of the Csv data frame, which contained some features to be studied.

****

This (Show) function showed the first 20 rows from the JSON data frame, and some features which will be studied and analyzed.

Then I checked the null values inside the columns, to see how many there are, and the result was as the following:



****

****

The null values in each column were dealt with in numerous ways, either by dropping the rows, or the feature itself, or filling them with relevant values that make sense.

**- country and country code**: I made sure to eliminate all irrelevant values from the columns, doing that made the country be resided only in the United States.

**- license**: the license feature was dropped because the null values exceeded 90% of the column’s data.

- **House Rules**: this column had half its data missing, but I decided not to drop it, instead I filled the null values with (No rules provided) given that some people may have forgotten to include it in the hosting form.

- **Last review**, reviews per month, and number of reviews: These three rows had some correlation where they had missing values, I decided to eliminate the rows where all three were null values.

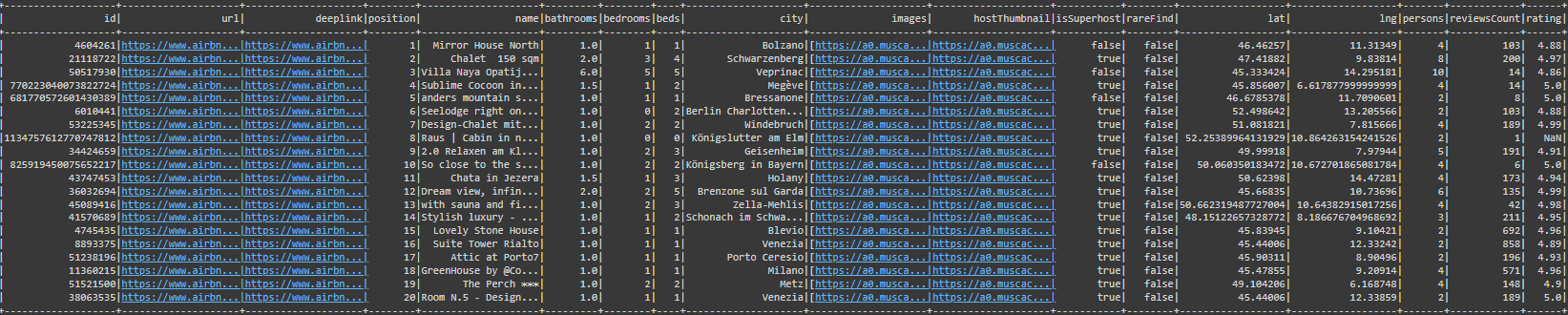
- **Review rate number**: I filled the null values of this feature with the mean, then casted it to Integer, so that it stays between 1 and 5.

After handling the missing values in these columns, I transformed the numerical values from string to integer, the challenge here was that there were two columns (price & service fee), which had values starting with the dollar sign ($). To handle this, I removed the dollar sign from the values inside the features using the Regular Expression library, so that they don’t turn out to be NULLs when I cast them to integer.

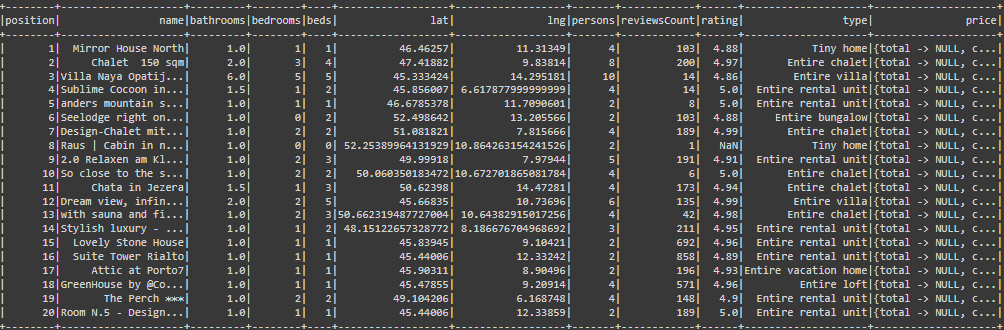
**JSON FILE Preprocessing:**

The way I handled the JSON file started with dropping the features that included IDs, URLs, and other features. Since they will be irrelevant to the data analysis.

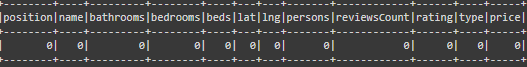
**Before**

****

**After**

****

After handling the irrelevant features, I checked for missing values, which were not present.



Then I started studying the data, and found that the Price feature had unexplainable values, so I handled it by checking the CSV dataset’s price column, I collected the min and max values from it, and made the JSON’s price column have random values between the min and max values from the CSV’s price column.

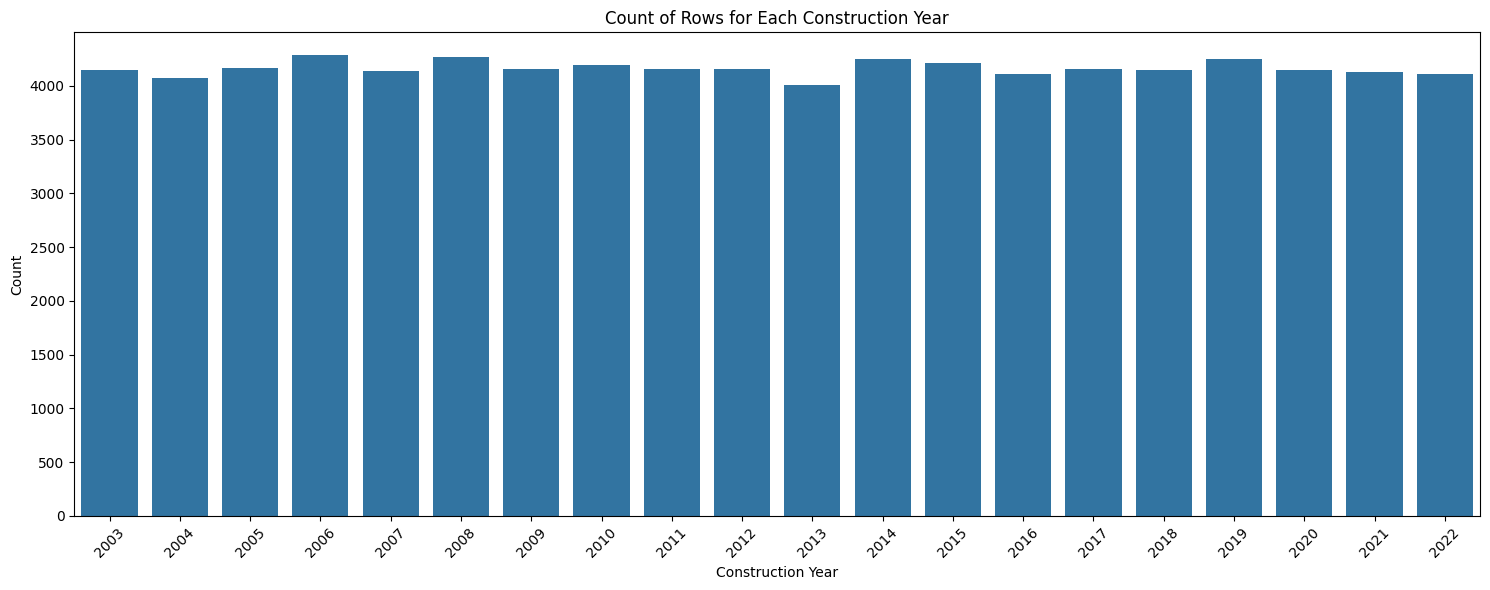
I also rounded up the rating feature to the nearing 10th. So that it gives a value between 1 and 5, same as the CSV’s rating feature. After that I casted the string numerical columns to integers and doubles. So that they can be analyzed effectively and efficiently.

1. After the cleansing and preparation steps for the data. Merging and aggregating both Data frames was challenging and impossible, due to the number of rows in each (80,000 & 300). I made sure to get them closer by performing oversampling, and under sampling techniques on both CSV and JSON data frames. So that they can be aggregated efficiently.
   * Over sampling of JSON DF.

First, I duplicated the JSON df twice, while adding and changing the numbers inside the features, so that they aren’t fully duplicated. Doing this, let the JSON df reach 900 Rows. Then, I added 100 more rows by studying the features and selecting a good subset of data. This led to the JSON data frame to reach 1,000 rows.

* + Under sampling of CSV DF.

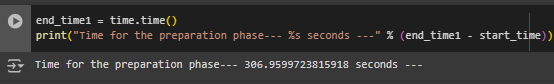
First, I started looking for a row that can help me reduce the number of values for the CSV df, which happened to be the construction year feature. With the help of seaborn, I checked for the count of each unique value for this feature, to study its ability to be reduced equally.



Looking at this plot, the data was distributed evenly for all years. My plan was to take 50 rows of each year to have a dataset of 1,000 rows for the CSV file. This will mean that the data was taken in a balanced and efficient way with no bias towards any year.

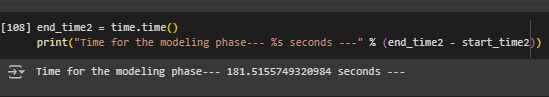
After that, I joined both data frames in a single data frame (Combined DF) which had all the data from both the JSON and CSV file, ready to be used for predictions.

1. I have provided some time and storage computations that will show how long the program takes to execute and the computational resources necessary to run such code. I have divided the time to three phases.
2. The time to run the preparation phase (Preprocessing + Aggregating)



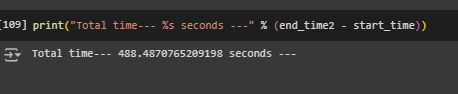
This shows that it takes 306 seconds to apply all the necessary missing value handling and transforming into numerical and aggregation for both data frames. (Nearly 5 minutes)

1. The time to run the modeling phase (Machine learning + Metrics)

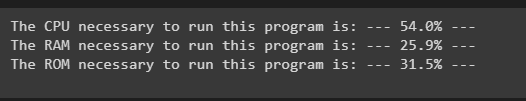


This shows that it needs 181 seconds to run the Logistic Regression I’m using to predict the rating of the hosting. (Nearly 3 minutes)

1. The total time to run the program all together (Both A and B)

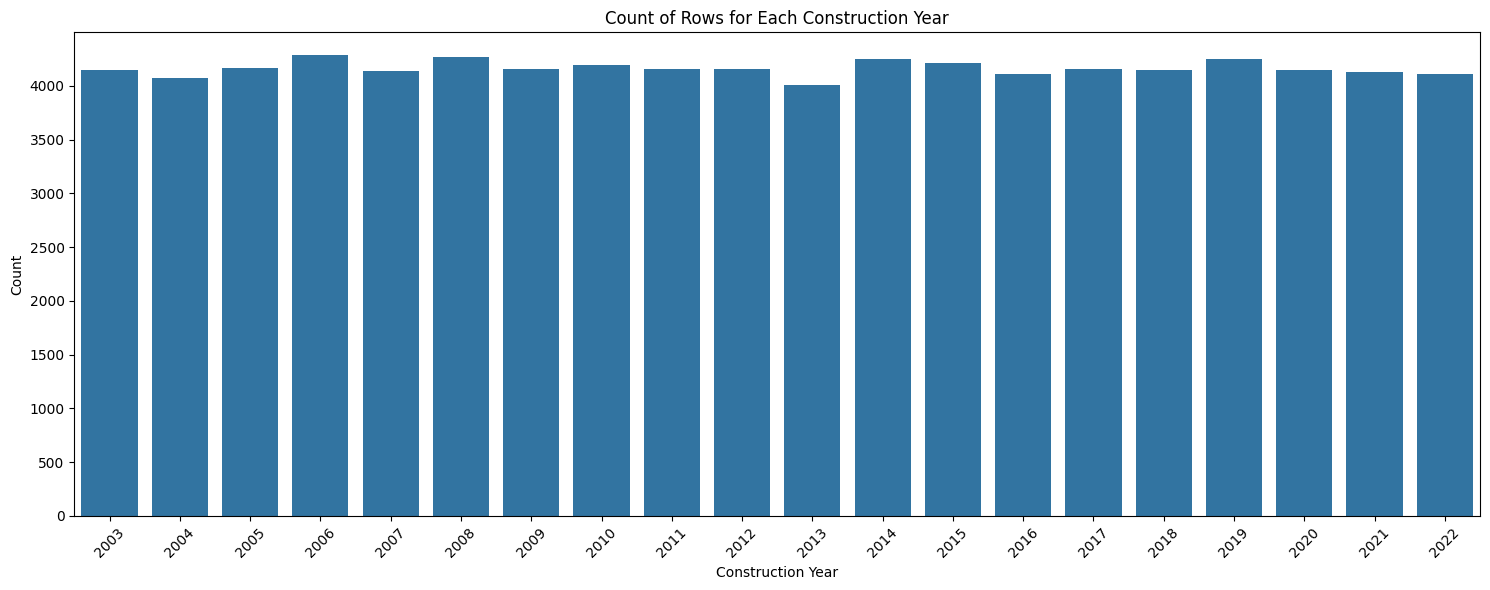


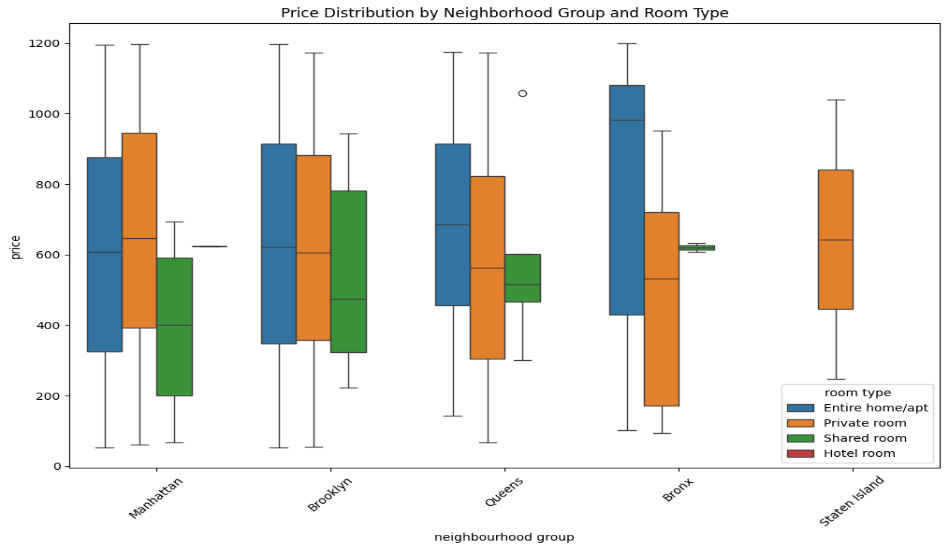
This shows that the total time to run both the preparation phase and the modeling phase is 488 seconds (Nearly 8 minutes)

And the storage and resources necessary were divided into three computations. CPU, RAM, and ROM.

This computation shows that to run such program on Google Colab from the beginning of the collection and preparation of the data (CSV and JSON) to the modeling phase. Requires 54% of CPU, 25% of the RAM, and 31% of ROM of Google colab.

These resources are considered high and computationally expensive since we’re working with Big Data (various sources of data).

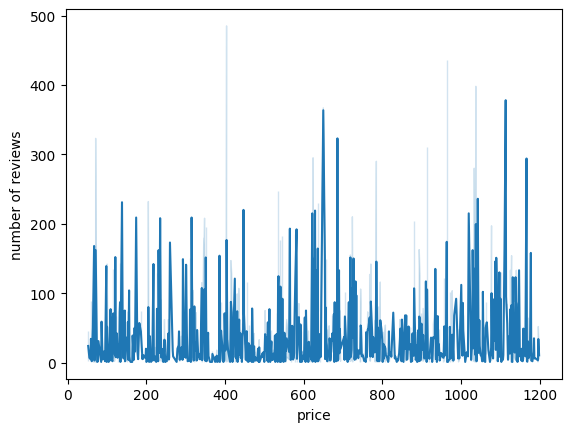
1. To analyze and study both the datasets I used multiple statistical techniques with the use of plotting and visualization, they helped get a better look at the data in a statistical view, and to guarantee a more effective way of thinking and decision making.
2. First, I used the Min and Max functions that show the minimum value and the maximum value inside a feature, I used it to check whether there are outliers in some features such as the rating review feature.
3. I also used the mean (which is the average of the values) to handle missing values inside features, since it gives a more statistical and clear reading for the data.
4. I used a bar plot to show the count of the distinct years inside the data, which looked like this
5. I used Box Plots to see how the prices were distributed based on the room type and the neighborhood the hosting was in, and it looked like this



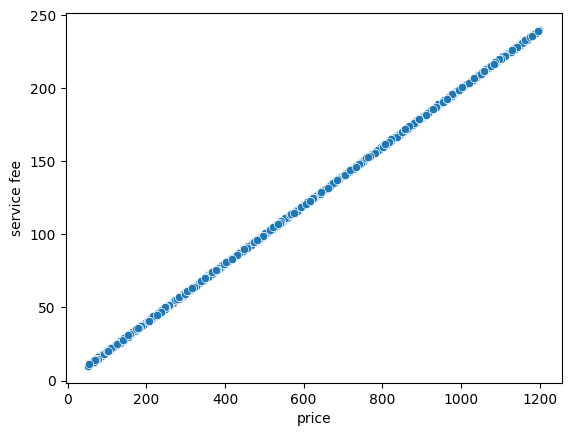
I used seaborn and Matplotlib libraries to create the visualizations, they are some of the best plot creators in Python, since they allow users to interact and modify the plots how they like and how their task requires.

I used the visualizations in multiple phases, first phase was in the preparation phase, where I used the year to find whether it is balanced to proceed with the aggregation process. The second phase was to outline insights and findings to the goal of the program, which is to predict if the rating of the hosting based on the characteristics of the hosting is good or bad.

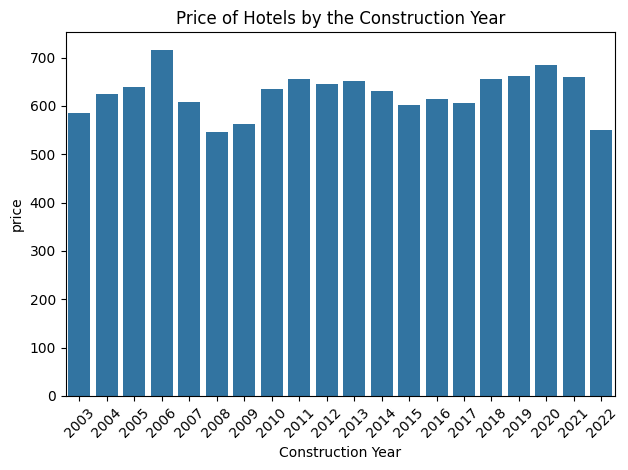
1. Here are some of the visualizations that helped me through the analysis phase to further enhance the understanding of the dataset I have:



This line plot shows the number of reviews that occur with the increase of the price. It states that when the price is between 600 and 1200, which is considered high, the number of reviews increase. Which shows that people who pay more, give an honest review to the place they stayed in.



This scatter plot shows a linear increase with the service fee and the price of the hosting, which indicates that when the price increases, the service fee also increases, and from the numbers in the plot, it looks to be a 5:1 relationship between both features.



This bar plot shows the relationship between the construction year and the price of the hosting. Where it shows that the hotels made in 2008 and 2022 cost the least while the hotels made in 2006 cost the most.

I have used numerous tools in my program to help provide a result that can help me succeed and reach a goal for the problem statement

The problem statement:

The use for Airbnb has been significant these past years, and its still growing due to the fact of frequent internships and family vacations outside of their residing country. While some users have had a great experience selecting a hotel and living in it, some have not had the same experience, which would lead to reporting for the host and closing the hotel on Airbnb immediately. This is due to the problem that some hosts believe that the hotels they host are welcoming for visitors and have the best characteristics people can live in. How can the hosts know that their hotel will grant the visitors a suitable and happy experience without getting reported and most probably banned?

The solution:

The solution for this problem is a predicting system that allows the hosts to write in the characteristics of their hotel in a system, so that it can predict whether the rating will be Good or Bad based on historical data collected from multiple sources of data (JSON and CSV).

Some of these tools are the open-source platform that the program was conducted on (Google Colab). Apache Spark, the tool that was used to code both sources of data and aggregate them together. Matplotlib and Seaborn the visualization libraries that were used to help provide interactive plots to further get an understanding of the data.

1. **Google Colab:** This tool provides a platform for interactive coding for machine learning projects. It is suitable for providing a solution for the problem of the hosts, by the aid of model development based on analysis, exploration, preparation of historical data collected and stored inside Google Colab itself. It facilitates those files of data and allows programmers to make codes that help solve problems daily.

**Pros:**

1. It allows cloud computing to save your work, this means that you can work on your programs from any device as long as you have internet access, this is useful for collaborating with teams for projects.
2. It provides the resources necessary to run your programs effectively (CPU), and it offers a GPU service which allows programmers to run their highly-computational codes quicker and more efficiently.
3. It can be integrated with other Google applications such as Google Drive which allows programmers to load their files from it, and work on it directly.

**Cons**:

1. The sessions that users use for their programs can run out and reset eventually which would require the users to run their programs again from the start.
2. It contains a limited customization system, since it is a server environment, it restricts the customizations that users may need for their systems and codes.
3. It requires internet access, unlike other tools (Jupyter Notebooks & Visual Studio), it needs the devices to be connected to WI-FI to continue working on the programs and execute the necessary programs run by users.
4. **Apache Spark:** Spark is another Apache product that consists of multiple distributed computing systems, that help process big data and complex transformation between data sources, it provides a guide from preparation to modeling for the data collected.

**Pros:**

1. Spark provides scalability in handling big data and different sources of data, which is perfect for the solution for the problem provided.
2. Spark is known for its quickness and speed in performing tasks, making it faster than traditional tools like Hadoop.
3. It provides an API such as PySpark which can be integrated with the commonly used python libraries such as Pandas and Seaborn.

**Cons:**

1. Spark can be complex to set up, since to manage it, you need to have an active session which might need a lot of resources to work.
2. The errors encountered in Spark can be challenging to debug, since most times, it cannot be understood clearly, unlike Python.
3. **Seaborn:** Seaborn is one of the two visualization libraries that is built on top of the other one, Matplotlib. This library is known for its aesthetic plots and for its high-level of interactive plots and graphics. It simplifies the creation of plots needed to understand the features inside the dataset of CSV and JSON.

**Pros:**

1. Seaborn is easy to use and requires no detailed coding techniques to comprehend. It offers a high-level interface for users to create their plots
2. It is known for its aesthetic plots and attractiveness of the plots; it uses color palettes to provide appealing plots which can be useful to perform effective storytelling to stakeholders.

**Cons:**

1. Seaborn suffers from limited customization, unlike Matplotlib which is significantly detailed for plots, seaborn can be limited when it is required to customize plots in detail.
2. It is often used for small data representation, using it for big data such as the task I’m working on, can be challenging and not effective.
3. **Matplotlib:** The second visualization library I used for creating plots was Matplotlib, it is known for its detail in creating interactive, animated, and beautiful visualizations for data, It provides capabilities that are needed to understand the data I have and helps generate findings that are used to reach a solution to the host problem in Airbnb.

**Pros:**

1. Matplotlib is known for its widely-usage, since it is commonly used in Python to provide plots and graphs, allowing it to have a whole community of interactive documentations and support for plots.
2. It is flexible for controlling the whole plot, which allows customizations for the plots created by the users and programmers.

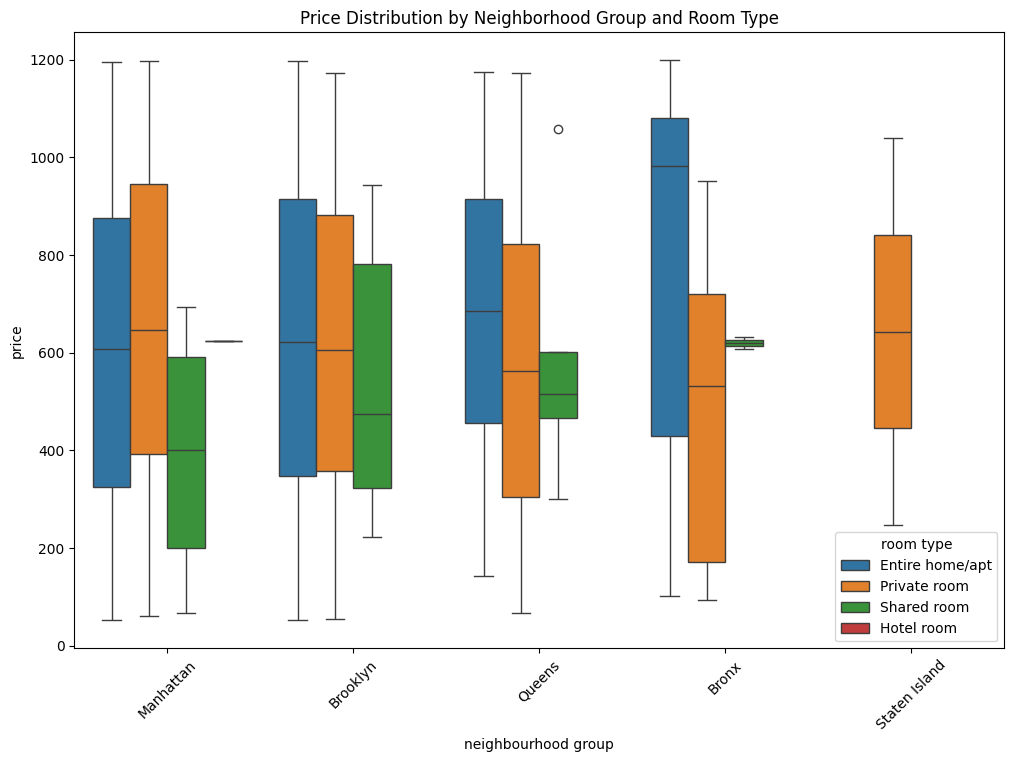
**Cons:**

1. It has a complex syntax when it comes to coding, since you need to initialize every aspect of the plot, unlike seaborn where you only initialize the data and the features needed to plot, Matplotlib needs details for each step.
2. It uses a default style in plotting visualizations so it may not be as appealing as Seaborn, but it can be customized to be better than the plots Seaborn offers.

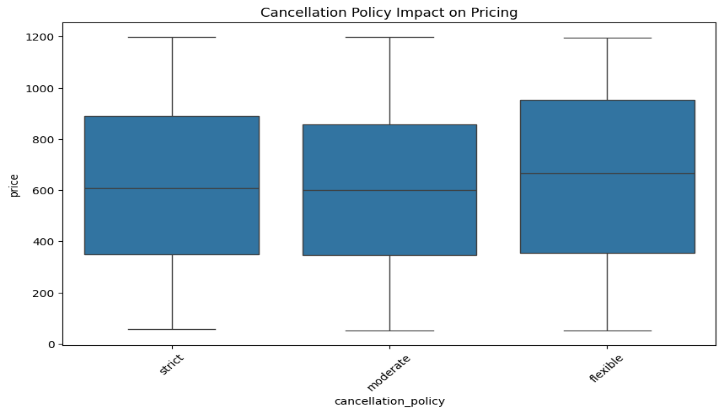
These tools were used to provide a detailed approach from collecting and preparing the data, to performing modeling processes and generating outcomes that can be used for the solution of the problem statement above. These tools help provide an overview of a program that works with big data and multiple sources, and can communicate results to stakeholders accordingly with the help of narrative storytelling.

**Question 2.**

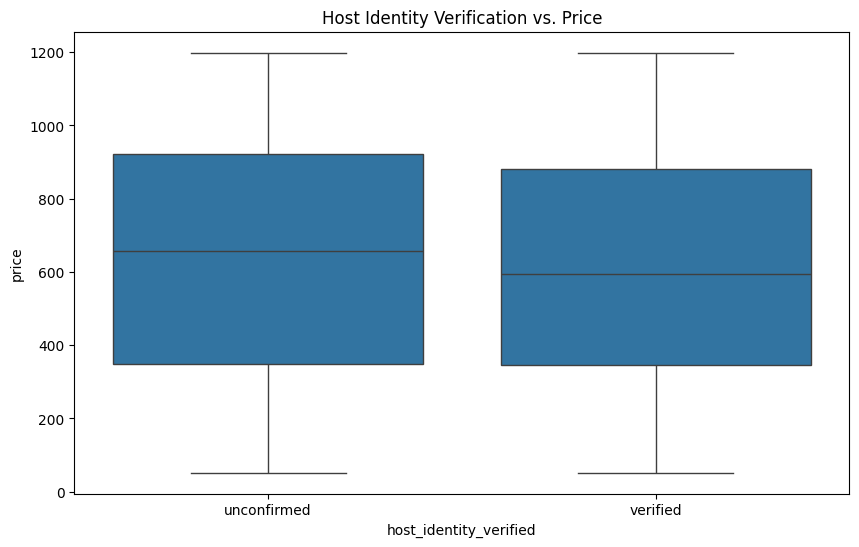
Many appealing plots and relevant visualizations were done using the Seaborn and Matplotlib libraries, and they were done in order to collect insights and help understand the data more effectively. Here are some of the visualizations used for the CSV file.

This plot shows the price distribution by the room type and the neighborhood group that the hosting place is in.

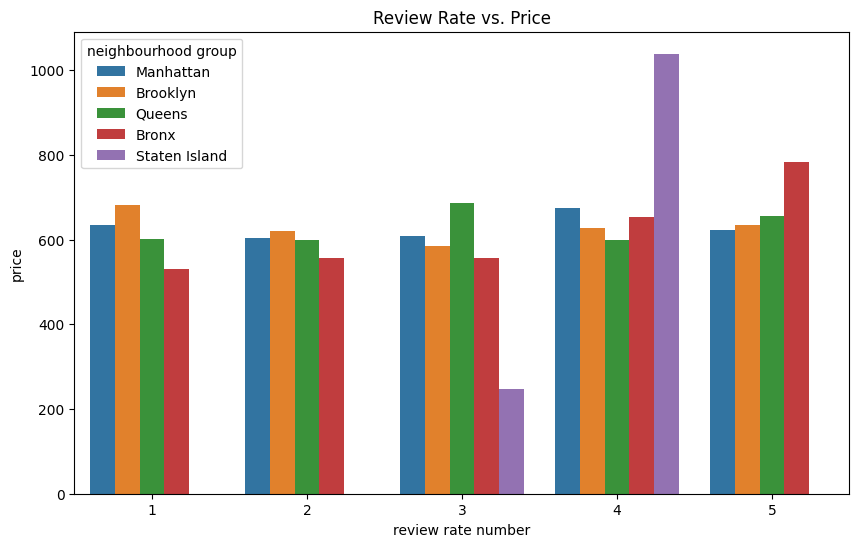
This plot shows that the Bronx neighborhood group contains the highest price for an apartment. It also states that no neighborhood group had hotels built in them.

This Box plot shows that the price of the hosting institute differs slightly when the cancellation policy is flexible, where it can reach $1000.

This shows that when the cancellation policy is set to flexible, people can pay more for the rooms, which states that people are more comfortable and acceptable with the host.

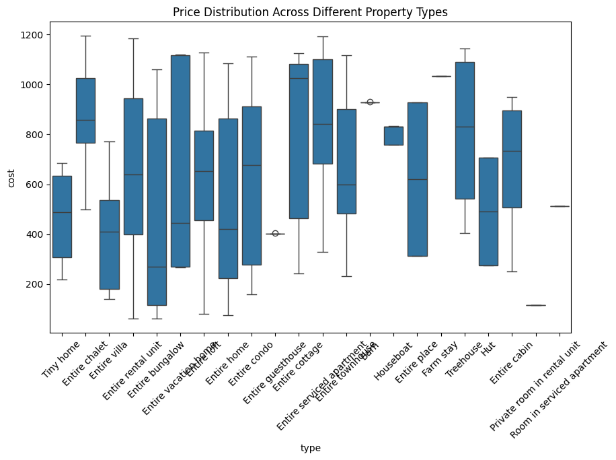


This box plot shows that when hosts who are not confirmed to be verified inside Airbnb can ask for more prices, which means that they cannot be trusted completely unlike verified hosts. This states that the host needs to be verified to be trusted since asking for more money for a room can be worrying and uncomforting for the visitors.



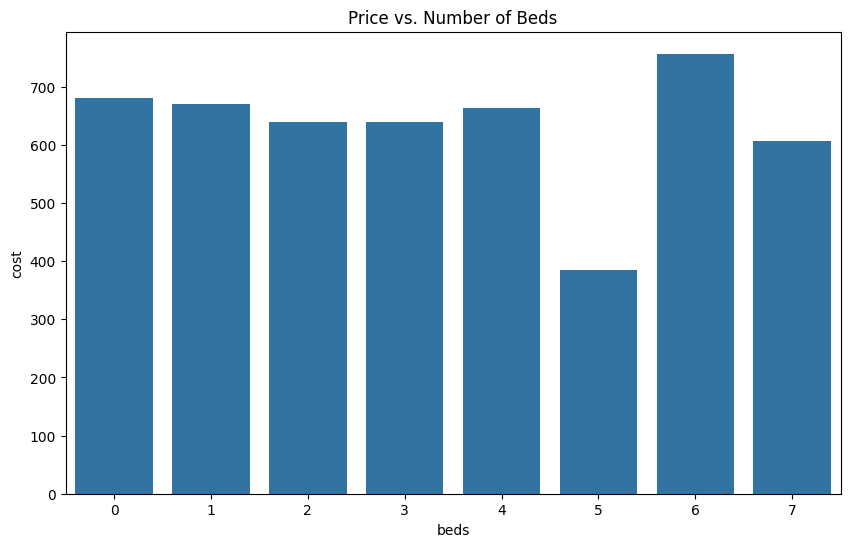
This bar plot shows the review number for the hosting places located in neighborhood groups and their prices.

The plot shows that Staten Island never had any 5-star ratings, this means that visitors do not visit that neighborhood often. But it states that the most paid for hosting was also located in Staten Island (4-Star).



As for the JSON file, visualizations done on it helped understand it more clearly to collect insights for business opportunities.

This boxplot shows the prices distributed by the room type. It shows that the highest paid room type is the entire serviced apartment, and the lowest is a private room in rental unit. This states that people pay more for an apartment and lower for a resold room.

This bar plot showed the relationship between the price and the number of beds. This shows that 6 beds in an apartment is paid for more than others, and a 5-bed room is the least paid for.

This can help hosts to create rooms that have 6 beds instead of 5. Other bed counts are acceptable.

Seaborn as a tool was used because it offers a simple way of creating interactive and appealing plots, and Matplotlib was used for its detailed customization that can be necessary in some of the plots. I used a mixture of both libraries to enhance the ability of customization for plots while having an aesthetic feeling to the visualizations created for the data.

I used an oversampling process for the JSON data frame so that I can aggregate it with the CSV data frame which I also conducted an undersampling process for. I chose to get both data frames to reach 1,000 records. Because this way it can be handled effectively by the model. And so that it can be clear to read. The reason why they need to be in the same length is to avoid having missing values occur in the merging process.

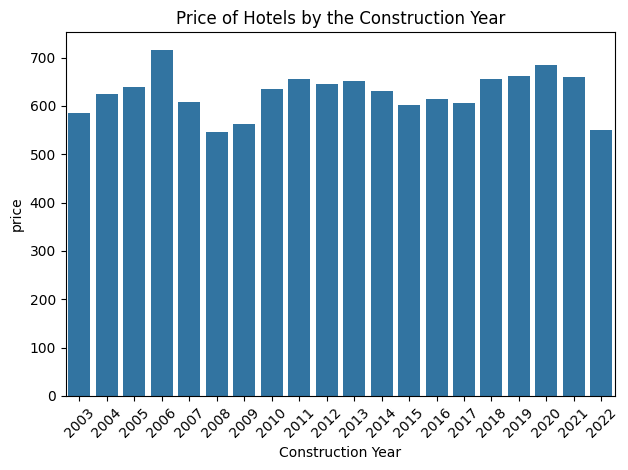
I dropped the unnecessary features so that the model can be trained effectively with no irrelevant data. And transformed the necessary features and handled them efficiently so that I can use them for the model to generate good accuracy and meaningful insights.

I used String Indexer to encode the categorical variables into numerical variables so that they can be fit correctly in the model. I used Vector assembler to combine all the features needed to be fit in the model into a vector called Features, this vector will contain all the features to be trained into the model.

I used a logistic regression model for the prediction of the rating category (Good or Bad), because it handles binary target classes in an efficient way. By using the logistic formula to classify the probability of all predictions, and I used a pipeline to combine the features, target, and model to act as a facilitator for the modeling phase.

I split the data into training and testing data with a ratio of 8:2 for training data. Then I fit the training data to the model, and compared the predictions with the test data.

The model performance was good but not the best. It resulted in 60% in accuracy, 55% for precision, 53% for recall, and 58% for F1-score. This performance can be the result of the generation of the data, since some insights from the data made no sense, this can only mean it was generated and wasn’t collected from the real Airbnb website.



This plot is the evidence that the data was generated, the years are fairly balanced and distributed evenly, and the prices for older hotels are lower than newly constructed hotels which isn’t considered logical in real life. This means that the data was generated and it can not be reliable for insights and decision making.

It was fairly balanced and had no errors in its balancing.

This can indicate that 60% as accuracy is great for the data, and can be used but it wouldn’t be considered reliable.

**Question 3.**

The implementation of Hadoop instead of Spark in this given scenario will change the process of the program in several different ways. Such as the time needed to complete such program, and the storage of the data, while considering the data processing done, and the overall environment.

1. Dealing with data preparation

Hadoop: Hadoop can use the MapReduce feature for batch processing, it can be highly computational since the data needs to be written to the disk when switching from map to reduce levels, this can slow down processing and machine learning processes.

Spark: Spark however uses in-memory computations to deal with the processing of the data, which makes it faster than Hadoop especially with machine learning processes. It also can support real-time generation of data, and can use many APIs that are easy to use instead of the MapReduce.

1. Data Storage

Hadoop: The HDFS of Hadoop is used to store large datasets from multiple sources, which can make it compatible with many sources of data with high volumes, but it usually deals with unstructured data and may require tools to manage other types of data.

Spark: Spark can also use the HDFS of Hadoop since it can be used as an add-on for Hadoop, but spark itself doesn’t have a storage system, it normally uses data frames to store data, so that they can be dealt with using different operations that help work with structured data.

1. Speed and performance

Hadoop: Hadoop is slower for algorithms because of the need to write results for the disk with MapReduce, it can be used for huge datasets and process them entirely in a single process.

Spark: It is faster than Hadoop generally, since its use of in-memory calculations. It can be significantly high in performance for machine learning models, and it can be used for analysis purposes for some tasks and projects.

Generally, Apache spark is better than Apache Hadoop in this case, because it offers better performance due to its in-memory processing skills, it can provide significantly faster machine learning insights, while integrating many more libraries and other skills using the APIs that it provides to help give a more suitable solution for the task at hand. Hadoop, on the other hand, would’ve been better than spark in this task if the focus was on the storage of the data and the batch processing capabilities. But, for building a predicting system for hosting rating category, spark is a better fit for a more efficient approach.

**Question 4.**

There were many merits and limits with the preparation and sourcing of the data, they were taken into consideration while programming the solution for the Airbnb problem. Some of these advantages were:

- Variety of data: Since the data was found in multiple sources, it was found holding infinite amounts of information to help the solution, this included structured data (CSV), and semi-structured data (JSON), this variety allowed collection of a richer dataset that helped build the predictive model that predicted the hosting’s rating category.

- Accuracy of the model: By applying the numerous feature engineering techniques and the meaningful cleaning of the datasets, an improved model accuracy was generated, some feature engineering examples were to convert categorical features into numerical features to handle the data fitting process efficiently into the model. The techniques used in the program reduced the noise for the model and led to better model performance and reliable predictions.

- Scalability and ability: Since Airbnb’s data can be complex and hard to process, tools like Spark helped maintain a comprehensive environment for processing the datasets from multiple sources, this ensures that the model can handle continuous volumes of data without the need of external add-ons. Spark also helped with its ability to integrate with machine learning algorithms, because the solution of the problem deemed it necessary to use ML models.

Yet these advantages came with some challenges along the way, some which were:

- Data inconsistency and quality: It was shown that some of the data collected from the sources were poor, since there were inconsistencies with the data types, the schema structure, and the levels of generality and logic. Aggregating the DFs was challenging and consumed a lot of time, the data itself also had many missing values that were taken care of, so it may not be reliable to take into consideration. Since it can negatively impact the model’s performance.

- Integration of the data: The aggregation process had many challenges to begin with, such as finding the right column to merge both DFs into, which can make it challenging to aggregate the data clearly. It led to challenges in dealing with duplicated data while merging the DFs, since it can happen frequently if the data was duplicated to reach a certain level which I followed.

- Generality and privacy of data: It was shown that the data was deemed inconsistent and illogical, which is why the data was generated automatically, but the reason of this process can be the privacy of such data. Airbnb may not include sensitive information about its hosts and visitors, so that it complies with the regulations of GDPR or other privacy regulations for data.

Overall, the preparation and sourcing of the datasets had many limits and merits. While the merits included collecting for richer information and model accuracy and scalability, the limits included some challenges in the quality of the data, privacy concerns, and the integration of the data frames. Studying these statements and taking them into consideration can lead to effective results for the problem and good decision making for stakeholders.

**Question 5.**

The big data lifecycle is a process of stages that when followed, the performance of such programs will be significantly effective and highly reliable and strategic. The program I made followed the stages of the big data lifecycle to reach the full potential of the solution implemented. The big data lifecycle has six stages:

1. **Discovery (Problem Statement):** This is the first stage in the big data lifecycle and it states the problem statement, it ensures that the problem objective is clear and concise. It focuses on collecting the data from numerous sources, such as CSV files, images, videos, or more. In this stage, communication with stakeholders must be held to reach a well-defined problem statement.

The problem statement I stated was clear and concise, and it solves an actual problem that hosts are facing in Airbnb regarding the ratings. It uses some of the characteristics of Big Data (the 5 Vs), it was challenging to reach a problem statement that measured a measurable solution at first, but with the meaningful collection of data (Various sources of data), it was deemed possible.

1. **Data preparation**: This stage includes all the cleaning done, the transformation conducted between data types, and the numerous preparations for analysis. This step is critical and necessary for the modeling stage later on. It can involve handling of missing values, converting the data to a suitable format so that it can be fit into the models. It also can include any integration done for the data. It can include some challenges at first, like having to deal with a huge dataset or a large file. And some inconsistencies in the data may be unreadable or inconvertible to other formats.

Preparation for the datasets I had included handling missing values by dropping, adding, and modifying the features I had. It included many transformations between formats, from strings to integers and doubles. And it also included the aggregation between the two datasets, since I have added a JSON data frame to a CSV data frame for more enhanced data collection and analysis. So that it can be fit into the model later on.

1. **Planning the model:** This stage involves all the visualizations and studying of relationships between the features, while also planning the modeling techniques to be used in the prediction phase. It includes all the EDAs conducted using many libraries of visualizations, and feature engineering for the features that can be fit in the model, while including some of the model planning that will take place in Stage 4. This stage can have limits such as the unbalancing of the data, and selecting the model technique used for the prediction phase.

In my project, I have included many visualizations and plots that studied how the data was distributed and the correlations between the dataset features, by the help of Seaborn and Matplotlib (Visualization Libraries), I was able to create many interactive and meaningful plots to get several relevant solutions for the problem statement mentioned above. Based on the dataset, which was balanced, I chose the Logistic Regression machine learning model due to the format of the target I chose, which happened to be a classification problem. This model is known for its simplicity and performance against binary class classification problems, and its high accuracy.

1. **Building the model:** This stage involves the development, training, and validation of the model selected in Stage 3. It involves the splitting of the data, and training the model on historical data collected in Stage 1, so that it can be validated with many techniques such as cross-validation or other techniques if necessary. Finally, it needs to be able to generalize well on new data (Avoiding Overfitting and Underfitting). This stage can involve many challenges regarding the resources needed to run the modeling phase, since it can be highly computational depending on the volume and format of the dataset fit into the model.

The development of the Logistic regression model in Spark happened to be different to the deployment I usually do using Python. It involves vector assembling the independent features that are fit into the model, which turns them into a vector, it involves encoding the variables using a String Indexer function that is used in Spark. It involves adding a pipeline that includes the features, target, and the model itself. I split the data 80% for training, and 20% for testing. I made sure to adjust the parameters for a high model performance.

1. **Communicating Results:** This stage involves informing the stakeholders of the results that the models obtained, using powerful insights and visualization tools. Storytelling is necessary in this step so that results can be communicated effectively for the stakeholders without any errors in the results and for further deployment for the model. With poor storytelling and bad result communication, the stakeholders (Mainly non-technical ones) may not take your consideration and would lead to inefficient reporting.

The results that resulted from the model, were generally good for the quality of the data collected and its volume, making it necessary to provide a valuable and narrative storytelling for the results and insights collected. The results were collected and interpreted in a visualization plot so that they can be communicated effectively to stakeholders, to reach a solution through continuous and meaningful decision-making processes. This stage involves talking about the necessary parts with good comprehension of the problem the hosts are facing in Airbnb to avoid any conflicts and poor decisions and judgments.

1. Measuring effectiveness and applying live: This stage concludes the big data lifecycle; it involves monitoring the performance of the model deployed overtime and reporting any maintenance issues or improvements that can be done to the model, so that it leads to better model performance and real-time decisions. Feedback must be continuous in this stage, and decision-making upon the model’s performance will be strictly proposed and made. Challenges with integrating the model to a live environment are possible, but the outcome is necessary for evolution and improvement.

After looking at the results and interpreting them with the stakeholders, It can be seen that applying such model to the environment can’t be reliable at first, but enhancements with continuous reporting and collection of more robust and clean data is possible. The model would need to improve to be deemed worthy of implementation into the real environment of predictions for Airbnb. It can be sufficient at first, but monitoring is needed for more valuable decision-making for the model.

**Question 6.**

**Apache HIVE** is a querying tool that is similar to SQL, it was developed by Facebook, and taken by Apache. It processes the structured data stored in the HDFS of Hadoop, and is efficient for batch processing. The queries inside HIVE can be converted into MapReduce tasks. It is an OLAP tool and is suitable for high latency tasks, but isn’t efficient for fast responses or unstructured data processing. Apache HIVE consists of many components that build its architecture for query processing, it involves:

- It uses **JDBC and ODBC drivers** as HIVE clients to send HiveQL queries that can be written in different programming languages such as Java and python.

- It contains many **services** such as a **CLI, a HIVE server, and a web interface**, they are then integrated into a Driver that acts as the lifecycle of Hive’s processing stage, the lifecycle involves a Parser, a Planner, an Optimizer, and an execution. These lifecycle components are used to process the data and execute the jobs so that they can be used for MapReduce tasks later on.

- Hive’s architecture contains a **Metastore**, which acts as the storage for metadata for data frames and their columns, rows, and data types. **A file system**, which stores the files themselves, (can be HDFS). And **the Job client**, which acts as the job monitor for the tasks required to be executed.

**- Hive’s storage** stores the metadata inside a database, and the file systems and job clients inside the Hadoop cluster or the HDFS.

**Apache Spark** however, is an open-source framework that uses distributed computing and processing to provide real-time data analytics, it uses in-memory computations to process the data, it supports multiple sources of data, and uses APIs to integrate python’s libraries such as Pandas and Scikit-learn into Spark. It is written in Scala, which is a programming language that supports object-oriented and functional programming. It consists of many components that help perform the processing of real-time data, with multiple sources, which are:

**- Spark’s SQL+ Data frames** support different formats of data, such as structured and semi-structured data files (CSV & JSON). It uses the SchemaRDD abstraction which consists of Resilient distributed datasets. This system works on collecting fault-tolerant elements that were partitioned from many nodes that are capable of parallel processing.

- **Spark’s APIs** are the main core of Spark, which help users integrate many libraries and programming languages into Spark for more efficient processing of large and different sources of data. These APIs involve Java, Python, Scala, SQL, and R.

**- Spark’s streaming** component allows spark to maintain fast scheduling performance while performing analytical operations in-memory. It transforms those streaming fast data and analyzes them effectively using the RDD transformations.

**- Spark’s MLlib** is a machine learning API that allows users to use machine learning libraries such as python libraries to perform different ML operations such as classification and clustering. It provides a distributed memory-based architecture to run those models efficiently with high-computational power.

**- GraphX** is the last of spark’s components, it allows for creating high aesthetic graphs using APIs that are used for expressing the data, it allows optimized runtime for the graphical plots and maintains a high computation for them.

Hadoop and Spark can offer many advantages for organizations when it comes to decision-making and leveraging their competition level across all applications. Some key considerations when it comes to working with both is scalability of work, flexibility and real-time processing of data, and the challenges of security for data and resource management. While considering the integration of both tools into the organizational work environment. And the continuous maintenance and reporting that needs to be done.

**Advantages of Hadoop and Spark:**

1. **Scalability:**

* Hadoop uses the horizontal scaling technique to store and process large volumes of data by adding more nodes to the cluster of Hadoop. This is necessary to the rapid growth in volume, velocity, and variety of data.
* Spark however uses distributed computing across nodes to process larger datasets easily, it uses the in-memory computing characteristic to enhance the scalability of using algorithms for analysis of data.

1. **Cost efficiency:**

* Hadoop allows using hardware disks and drivers which can be a cost-effective solution for some storage and data processing tasks, it can be easy and effective for organizations to adopt such tool for their daily tasks to enhance the performance.
* Spark’s usage of in-memory computations reduces the need of an I/O disk for operating data analysis and processing. Which reduces the time and cost of implementing models and generating insights.

1. **Real-time Processing:**

* Hadoop is used for batch processing, which can be slower than the real-time processing that Spark does, but it can be integrated with many programs and tools to find its way capable of performing real-time processing in no time. It is efficient for organizations that collect their data in batches and can be implemented effectively there.
* Spark, on the other hand, works on processing real-time data that is generated in streams, it is 100 times faster than Hadoop’s MapReduce system, which can be integrated faster in organizations that work with streaming lines of data daily.

**Challenges of Hadoop and Spark:**

1. **Implementation complexity:**

* Hadoop finds implementing its cluster complex and it would require intensive knowledge in distributed computing and management for clusters. Writing the MapReduce codes inside the system can also be complex since it is written in Java.
* Spark may be easy to use, but it is much harder to implement, since the need for distributed computing is necessary, and it requires many resources management and optimizations to use its full potential.

1. **Highly computational:**

* Hadoop’s performance can require a lot of resources to work on large data sets, especially if the clusters aren’t tuned efficiently, it would require some applications to integrate into Hadoop for better resource allocation processes inside the cluster.
* Spark can be resource-intensive too, since it works with in-memory calculations and processing for data, which can lead to challenges faced by organizations when needing spark usage on large datasets with high variety and volume.

1. **Governance and Privacy:**

* Hadoop operates in distributed computation environments which can lead to some data security challenges and measures, by data breaches or unauthorized access to the data. It can lead to putting the organization at risk with the global data regulations (GDPR).
* Spark also operates in a distributed environment, its in-memory calculations may leave some data to be visible to the public in case of overload by the volume of data and speed. It may lead to some sensitive data leaked. This must be protected by insisting on organizations to align with compliances such as HIPAA and GDPR to mitigate being at risk financially and reputationally.