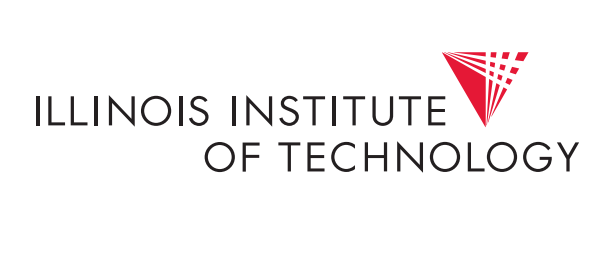
****

**BIG DATA TECHNOLOGIES – CSP 554**

**Choladevi Gheereddy - A20544476**

**NYC Yellow Taxi Data Analysis**

**PROJECT REPORT**

**Instructor**

Prof. Panchal J Jawahar

Table of Contents

[1.Abstract 3](#_Toc152524012)

[2.Introduction 3](#_Toc152524013)

[3.Literature Review 3](#_Toc152524014)

[**3.1 Apache Hadoop Ecosystem** 4](#_Toc152524015)

[**3.2 Apache Spark** 4](#_Toc152524016)

[**3.3 PySpark** 5](#_Toc152524017)

[**3.4 Amazon S3** 5](#_Toc152524018)

[4. Data 6](#_Toc152524019)

[5. Project Workflow 7](#_Toc152524020)

[**5.1 Amazon S3 setup** 7](#_Toc152524021)

[**5.2 PySpark Setup** 8](#_Toc152524022)

[**5.3 Data pre-processing** 9](#_Toc152524023)

[**5.4 Data Analysis** 10](#_Toc152524024)

[**5.4.1 Distribution of Passenger count** 11](#_Toc152524025)

[**5.4.2 Distribution of Trip Distance** 11](#_Toc152524026)

[**5.4.3 Distribution of Payment types** 11](#_Toc152524027)

[**5.4.4 Visualizing trip patterns** 12](#_Toc152524028)

[**5.4.5 Frequent Pickup & drop off location** 13](#_Toc152524029)

[**5.4.6 Frequent pickup and drop off zones** 13](#_Toc152524030)

[**5.4.7 Fare amount distribution** 13](#_Toc152524031)

[**5.4.8 Tip Amount distribution** 14](#_Toc152524032)

[**5.4.9 Analyzing the rate codes** 14](#_Toc152524033)

[6. Conclusion 15](#_Toc152524034)

[7. Source Code 15](#_Toc152524035)

[8.References 15](#_Toc152524036)

# **1.Abstract**

NYC yellow taxi industry have been the subject of various challenges and discussions over the years. These challenges include traffic congestion, fare structures, taxi availability, competition with ride sharing services like Uber/Lyft. This project seeks to explore potential solutions to these issues using big data related tools/techniques and develop a comprehensive insights and recommendations to taxi drivers, considering the interests of passengers, drivers, and the overall sustainability of the industry. This project considers substantial parquet files, employs high-performance computational clusters for PySpark, and constructs diverse analytical charts.

# **2.Introduction**

In New York City, the yellow taxis are an important iconic symbol that represents the city. These are the famous NYC yellow taxis that provide transportation exclusively through street hails. You access this mode of transportation by standing in the street and hailing an available taxi with your hand. The New York City Taxi and Limousine Commission (TLC), created in 1971, is the agency responsible for licensing and regulating New York City’s Medallion (Yellow) taxi cabs, for-hire vehicles, commuter vans, and paratransit vehicles. Over 200,000 TLC licensees complete approximately 1,000,000 trips each day.

However, NYC taxi market has deteriorated for the traditional yellow cabs since the Uber and other ridesharing applications introduced in the city.

# **3.Literature Review**

Big data is the term used to describe datasets that are too big or complex to be handled effectively by traditional data processing application software. Big data is a combination of unstructured, semi-structured or structured data collected by organizations. These data sets can be mined to gain insights and used in machine learning projects, predictive modeling, and other advanced analytics applications.

However, the volume and complexity of this data prevents many widely used software programs and systems from producing results in a timely manner. Big data technologies are precisely those programs, instruments, and frameworks that are designed and put into place with the express purpose of extracting value for the business from massive amounts of data.

Big Data is typically characterized by the five Vs: volume, velocity, variety, veracity, and value.

**Volume:** Volume is like the base of big data, as it's the initial size of data that's collected.  If the volume of data is large enough, it can be considered big data.

**Velocity:** It is used to describe the rate at which data is created and processed in real time.

**Variety:** Refers to the diverse types of data, including structured, semi-structured, and unstructured data, generated from different sources.

**Veracity:** Veracity refers to the quality, accuracy, integrity and credibility of data. Veracity, overall, refers to the level of trust there is in the collected data.

**Value:** Value refers to the usefulness and relevance of data in generating insights, making informed decisions, and creating value for businesses and organizations. This characteristic is critical for organizations to gain a competitive advantage and achieve their goals.

## **3.1 Apache Hadoop Ecosystem**

The Hadoop Ecosystem (fig 3.1.a) is a collection of tools, libraries, and frameworks that help you build applications on top of Apache Hadoop. The Hadoop ecosystem definition extends this functionality with additional tools to make it easier to use Hadoop with other frameworks like Spark or Kafka for real-time processing or machine learning tasks.

A group of logos on a white background

Description automatically generated

***Figure 3.1.a: Hadoop Ecosystem***

## **3.2 Apache Spark**

Apache Spark is an open-source distributed computing system used for big data processing and analysis. It has several features, including SPARK SQL, SPARK Streaming, Mllib, compared to conventional big data processing systems, Spark is much faster because of for its in-memory processing capabilities. Spark is made to handle a variety of workloads including streaming, interactive queries, and batch applications. Spark makes it simple to integrate several processing types by supporting diverse workloads in the same engine. It is often necessary in big data analysis.

**Speed:** Spark performs up to 100 times faster than MapReduce for processing large data.

**Powerful Caching:** These capabilities are offered by a simple programming layer.

**Deployment:** Hadoop via YARN, or Spark’s own cluster manager can all be used to deploy it.

**Real-Time:** Because of its in-memory processing, it offers real-time computation and low latency.

**Polyglot:** In addition to Java, Scala, Python, and R, Spark also supports all four of these languages. Spark also provides a command-line interface in Scala and Python.

A diagram of a company

Description automatically generated

***Figure 3.2.a: Spark***

**Apache Spark Components:**

**Spark Core:** Spark Core includes all of Spark's fundamental features. Supports RDDs.

**Spark Streaming:** To process real-time data streams/streaming data.

**Spark SQL:** Offers a SQL interface to Spark to allow querying and working with structured data. In this project we are using Spark SQL for querying data from S3 data source.

**Mllib:** This library is used for common machine learning (ML) functionalities

**Graphx:** It is used to work with graphs and execute graph-parallel operations.

Some of the key features of Apache Spark include fault tolerance, interactive shell, real-time streaming, and support for multiple programming languages such as Java, Scala, Python, and R.

Spark Core offers Java, Scala, R, and Python APIs for ease of development as well as for creating and modifying these collections.

A screenshot of a computer program

Description automatically generated

***Figure 3.2.b: Spark components***

## **3.3 PySpark**

PySpark is the Python API or interface for Apache Spark, an open-source distributed computing system designed for big data processing and analytics. With PySpark, you can write Python and SQL-like commands to manipulate and analyze data in a distributed processing environment. It was developed to overcome the limitations of the Hadoop MapReduce model by introducing in-memory processing and a more flexible programming model.

The reason to choose to a framework like PySpark is because of how quickly it can process big data. It is faster than libraries like Pandas and Dask and can handle larger amounts of data than these frameworks. If you had over petabytes of data to process, for instance, Pandas and Dask would fail but PySpark would be able to handle it easily.

Furthermore, PySpark provides fault tolerance, which means that it has the capability to recover loss after a failure occurs. The framework also has in-memory computation and is stored in random access memory (RAM). It can run on a machine that does not have a hard-drive or SSD installed. To use PySpark, you need to have Apache Spark installed on your cluster, and then you can interact with it using the PySpark API through Python scripts or Jupyter notebooks.

## **3.4 Amazon S3**

Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services (AWS) that provides object storage through a web service interface. S3 can store any kind of object, it can be used for a variety of purposes, including hybrid cloud storage, Internet application storage, backups, disaster recovery, and data archives and lakes for analytics.

It manages data with an object storage architecture which aims to provide scalability, high availability, and low latency with high durability. The basic storage units of Amazon S3 are objects which are organized into buckets. Buckets can be managed using the console provided by Amazon S3, programmatically with the AWS SDK, or the REST application programming interface. In this project we are use aws-sdk to manage and access the s3 objects.

# **4. Data**

New York City (NYC) Taxi & Limousine Commission (TLC) keeps data from all its cabs, and it is freely available to download from its official website. It primarily keeps and manages data for 4 different types of vehicles but for this project, 2022 Yellow Taxi data was used. It is available in parquet format for each month. It has around 24M rows. To enhance the analysis, taxi zone data from the same website was also incorporated, which consists of information over 250 zones within NYC. This additional dataset will enable us to extract valuable insights at the zone level. These datasets can be downloaded from the following links.

TLC Data: <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

Zone Data: <https://d37ci6vzurychx.cloudfront.net/misc/taxi+_zone_lookup.csv>

Below are the some of the important columns/characteristics of the trip data:

*VendorID:* It is a code indicating the TPEP provider that provided the record.

1= Creative Mobile Technologies, LLC, 2= VeriFone Inc.

*tpep\_pickup\_datetime:* It is the date and time when the meter was engaged.

*tpep\_dropoff\_datetime:* It is the date and time when the meter was disengaged.

*passenger\_count:* It is the number of passengers in the vehicle.

trip\_distance: It is the trip distance in miles reported by the taximeter.

*PULocationID & DOLocationID:* It is the pickup & dropoff of TLC Taxi Zone.

*payment\_type:* It is a numeric code signifying how the passenger paid for the trip.

1= Credit card, 2= Cash, 3= No charge, 4= Dispute, 5= Unknown, 6= Voided trip

*fare\_amount:* It is the time-and-distance fare calculated by the meter.

*tip\_amount:* It is automatically populated for credit card tips. Cash tips are not included.

*total\_amount:*  It is the total amount charged to passengers. Does not include cash tips.

Below figure (fig 4.a) is a sample data screenshot for yellow taxi trip data for the year 2022.

A table with numbers and a number on it

Description automatically generated

***Figure 4.a: Trip data***

To improve the analysis, taxi zone data from the same website was also added, which includes details on over 250 zones in New York City. This will extract beneficial insights at the zone level via to this additional dataset (fig 4.b). In this project, we are going to combine both Trip data (PULocationID and DOLocationID) with taxi zone lookup data (LocationID) to provide more insights and recommendations to taxi drivers to overcome challenges include traffic congestion, fare structures, taxi availability, competition with other taxi providers.

A screenshot of a phone

Description automatically generated

***Figure 4.b: Taxi zone data***

# **5. Project Workflow / Architecture**

The project workflow encompasses a systematic series of steps, seamlessly integrating various tasks essential for successful project execution. Commencing with the creation of an Amazon S3 bucket, this initial step establishes a secure and scalable storage space within the AWS ecosystem. Subsequently, the workflow navigates towards the installation of PySpark, a pivotal framework for large-scale data processing.

The workflow converges on data preprocessing, a critical phase where raw data undergoes meticulous cleaning and transformation. This refined dataset becomes the basis for the ensuing data analysis. The final leg of the project workflow is dedicated to data analysis, where PySpark's powerful capabilities are harnessed to derive meaningful insights. Leveraging the interactive PySpark shell within Jupyter notebooks, the analysis phase allows for the development and execution of PySpark programs. Exploratory data analysis, and pattern identification contribute to a comprehensive understanding of the dataset, ultimately leading to informed decision-making.

## **5.1 Amazon S3 setup**

In this project, Amazon S3 is used for storing all the data. To begin with, one needs to register an account for AWS. Navigate to AWS Management Console to create a S3 bucket. Each object within Amazon S3 is stored within a bucket.

Step 1: In the management console, search for S3 and select the S3 option.

Step 2: Click on the Create Bucket 🡪 Provide all the required information for creating bucket such as bucket name, permissions, ownership details etc. and click done.

Step 3: Once the bucket is created successfully, you will find your bucket on the wizard as shown below (fig 5.1.a).

A screenshot of a computer

Description automatically generated

***Figure 5.1.a: Personalized S3 bucket***

Step 4: To upload the files, select your bucket from the wizard 🡪Click upload 🡪 Add files 🡪 Upload all the files required for the project. In this case, all the 12 months parquet files of Trip data and NYC taxi zone look up data was uploaded to the S3 bucket (fig 5.1.b).

A screenshot of a computer

Description automatically generated

***Figure 5.1.b: S3 Upload***

Step 5: To access the bucket from third party applications, access and secret keys are required which acts as an authentication step. Navigate to IAM and spot the access key section.

Step 6: Click on Create access key 🡪 Copy the access key and secret key credentials from the page as shown below (fig 5.1.c). This access-secret key pair will serve as the authentication from the Jupyter notebook to access all the data files from S3.

A screenshot of a computer

Description automatically generated

***Figure 5.1.c: Copy Access/Secret Key***

## **5.2 PySpark Setup**

In this project, Jupyter Notebook on Anaconda was employed for collaborating with PySpark. The installation of PySpark on Anaconda involves utilizing the conda command, which serves as the package manager underlying the Anaconda distribution. Conda is a versatile package manager, compatible with various platforms and programming languages.

Step 1: Run the below command to install pyspark

A black rectangle with white border

Description automatically generated

***Figure 5.2.a: install pyspark***

Step 2: To execute PySpark within a Jupyter notebook initially, it's necessary to locate the PySpark installation. Findspark package needs to be installed. As it is an external package, it must be installed before being utilized.



***Figure 5.2.b: install findspark***

Step 3: Confirm the PySpark installation by executing the “pyspark” in shell. This initiates the PySpark shell, allowing interactive development of PySpark programs.

Step 4: Initiate spark session with all the required configs to access the S3.

A screenshot of a computer

Description automatically generated

***Figure 5.2.c: Initiate spark session***

## **5.3 Data pre-processing**

Data preprocessing is a crucial stage in the data analysis project, involving a series of steps to ensure that raw data is transformed into a format suitable for analysis. One integral component of this process is data cleaning, which aims to identify and rectify errors, inconsistencies, and missing values within the dataset. Additionally, data preprocessing involves tasks where new features are derived like day, week, month from the timestamps. In this project, various data quality checks were performed on all the columns to make sure no erroneous data is available.

Step 1: From the taxi zone data, Unknown boroughs were looked up and filtered them as they do not give any valuable info. (Fig 5.3.a)

A screenshot of a computer code

Description automatically generated

***Figure 5.3.a: Unknown boroughgs***

Step 2: Records with total\_amount less than zero were detected. There were 255706 such records. These records do not make any sense because trip cannot be 0 dollars. This is probably wrong entry by the drivers. (Fig 5.3.b)

A white rectangular object with red text

Description automatically generated

***Figure 5.3.b: erroneous total\_amount***

Step 3: Trip distance check were performed and found that around 574059 records had trip distance less than or equal to zero. This does not align with our analysis because 0 distance cannot be considered as a trip. (Fig 5.3.c)

A screen shot of a computer

Description automatically generated

***Figure 5.3.c: Unreasonable Trip distance***

Step 4: Passenger counts were checked and since these are trips, it is important to remove all the records where the count is less than or equal to zero.

Step 5: After all the data quality checks as discussed in previous steps, an SQL query was built and filtered out all the unwanted data from the input data as shown below (fig 5.3.d)

A screenshot of a computer code

Description automatically generated

***Figure 5.3.d: Filtering problematic data***

Step 6: After the data cleaning, few additional features are derived that are considered to boost this analysis. These features are day, week, month, trip duration and trip speed from the time stamps. These features will enhance the analysis at granular level.

## **5.4 Data Analysis**

PySpark will serve as the backbone of this analysis, enabling distributed data processing and parallelized operations. Throughout this analysis, the plan is to dive into fare trends over time, examining how prices fluctuate based on different factors such as peak hours, geographic locations, and special events. The investigation into ride distances will shed light on the distribution of trip lengths, allowing for a comprehensive understanding of the service's reach and user preferences. To present the findings in a clear and accessible manner, various Python libraries will be used for data visualization, creating charts, graphs, and interactive displays.

### **5.4.1 Distribution of Passenger count**

The passenger count has values ranging from 0 to 9. The passenger count values are entered manually by the driver. So, the 0s are probably the values where the driver did not enter any information. If we look at the chart (Fig 5.4.1.a), about 85% of the trips have only 1 passenger.

A graph of passenger count

Description automatically generated

***Figure 5.4.1.a: passenger count distribution***

### **5.4.2 Distribution of Trip Distance**

The value for trip distance have been made into several buckets based on the data. The highest being 50+ bucket. About 62% of the overall trips are shorter distances. Most common trip distance usually lies in the range between 1-5 miles. (Fig 5.4.2.a)

A graph of a distribution of trip distance

Description automatically generated

***Figure 5.4.2.a: trip distance distribution***

### **5.4.3 Distribution of Payment types**

From the distribution of payment type, we can see that most of the people prefer paying through credit card. This seems to be an interesting insight, since these are call up taxis unlike uber/lyft, the assumption of cash being highest mode of transport is ruled out. Probably, the drivers started carrying some sort of devices for credit card payments. (Fig 5.4.3.a)

A bar graph with numbers and a number

Description automatically generated

***Figure 5.4.3.a: payment types distribution***

### **5.4.4 Visualizing trip patterns**

Monthly visualizations can reveal long-term trends and seasonality, while daily patterns can highlight peak hours and fluctuations. Hourly analyses will showcase the day's granularity, aiding in optimizing services during specific time intervals for enhanced operational efficiency.

A graph of blue bars

Description automatically generatedA graph of blue bars

Description automatically generated

***Figure 5.4.4.a: monthly trips Figure 5.4.4.b: daily trips***

A graph of a number of blue bars

Description automatically generated

***Figure 5.4.4.c: hourly trips***

There are no major differences between the months (Fig 5.4.4.a). Most of them fall into similar range expect January/February. This could be due to the peak winter season in NYC and it makes harder for drivers to reach the locations in snowy areas. On daily insights chart (Fig 5.4.4.b), people tend to use the taxis more on weekdays rather weekends. This could be to avoid traffic delays, office working days. Hourly analysis (Fig 5.4.4.c) of trips shows that the highest number of trips occur between 5 to 7 pm. As expected, this is the time where people head back to their homes from their offices.

### **5.4.5 Frequent Pickup & drop off location**

In this case, Manhattan being the heart of the NYC, it was observed as the most frequent pickup and drop off borough (Fig 5.4.5.a & Fig 5.4.5.b)

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

***Figure 5.4.5.a: frequent pick-up borough Figure 5.4.5.b: frequent drop off borough***

### **5.4.6 Frequent pickup and drop off zones**

Looking at all the below tables demonstrating top and bottom 5 pickup and drop off zones, Upper East Side South is the most frequent pickup zone whereas Upper East Side North is the most frequent drop-off zone. Alternatively, Crotona Park and Great Kills Park are lease preferred pickup and drop off zones.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

***Figure 5.4.6.a: pickup and drop off zones***

### **5.4.7 Fare amount distribution**

The below plot shows average fare amount on an hourly basis for each day of the week. It is observed the trips taken between midnight to early morning have a slightly higher average fare amount. This might be because of it being an odd time to travel.

A graph of different colored lines

Description automatically generated

***Figure 5.4.7.a: fare amount***

### **5.4.8 Tip Amount distribution**

The plot shows average tip amount on an hourly basis for each day. It is observed that, people tend to give more tip odd hours whereas they give comparatively less during the busiest hours of the day.

A graph of different colored lines

Description automatically generated

***Figure 5.4.8.a: Tip amount***

### **5.4.9 Analyzing the rate codes**

Rate codes 2 and 3 are to JFK and Newark Airports so it is not too surprising that there is extra charge. Rate code 4 is trips to Nassau County which have a median trip distance slightly lower than for rate codes 2 and 3. This is because trips to these counties are charged at the standard city rate within New York City and at twice the metered rate while in Westchester or Nassau County. Rate code 5 is for negotiated fares and the median trip distance for fares with this rate code is much less which suggests that cab drivers are much better off when fares are negotiated.

**A screenshot of a computer

Description automatically generated**

***Figure 5.4.9.a: Rate codes***

# **6. Conclusion**

Overall, this “Big Data” project serves as a comprehensive exploration of the challenges faced by the New York City yellow taxi industry, particularly in response to the disruptive influence of ridesharing services.

Through meticulous steps, including the identification and elimination of unknown boroughs, rectification of erroneous total amounts and trip distances, and the removal of records with unrealistic passenger counts, the dataset is refined to ensure accuracy and reliability. Leveraging SQL queries, the team successfully filters out problematic data, laying the groundwork for meaningful analysis.

The findings are diverse and insightful, addressing various facets of the taxi industry. The analyses encompass the distribution of passenger counts, revealing a predominant occurrence of single-passenger trips. Categorizing trip distances sheds light on the service's reach, with most trips falling within shorter distances. The surprising preference for credit card payments over cash challenges conventional assumptions about payment methods in traditional taxi services.

Visualizations of trip patterns, both monthly and daily, uncover nuanced trends such as heightened taxi usage during weekdays and peak demand during the evening rush hours. Manhattan emerges as the epicenter for both frequent pickups and drop-offs, emphasizing the borough's central role in the taxi ecosystem.

Further, the exploration of fare and tip amount distributions over time provides valuable insights into pricing dynamics, with higher average fare amounts during unconventional travel hours and increased tip amounts during off-peak periods.

# **7. Source Code**

# **8.References**

1. <https://github.com/salahdev8/NYYellowTaxiProject>
2. <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
3. <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page#:~:text=Taxi%20Zone%20Maps%20and%20Lookup%20Tables>
4. <https://aws.amazon.com/getting-started/hands-on/backup-to-s3-cli/>
5. <https://www.carolynlangen.com/blog/2017-11-22-interacting-with-aws-s3-using-python-in-a-jupyter-notebook/>
6. <https://pypi.org/project/matplotlib/>
7. <https://medium.com/geekculture/what-is-a-blob-83e65f590694>