NYC Taxi Fare Prediction using Apache Spark

Introduction:

This project focuses on building end to end data processing and machine learning pipeline for predicting NYC taxi fare using Apache Spark. The project demonstrates should be distributed computing for large scale data preprocessing, integrating AWS services such as S3 and Athena, and implementing machine learning models using Spark MLlib. The goal is to process, analyze and predict taxi fares efficiently while optimizing model performance.

Data Source :

The dataset used for this project is the NYC Taxi Fare dataset, which contains over 55 million rows and exceed 100 MB, fulfilling the project requirements. The dataset includes feature such as:

Pickup and drop-off locations (latitude & longitude)

Passenger count

Trip Distance

Fare Amount (target variable)

The data was staged in AWS S3 and queried via AWS Athena before applying transformation in Apache Spark.

Data Processing & Transformation:

Using PySpark, the following preprocessing steps were performed:

Data Cleaning: Removed null values, invalid coordinates, and unrealistic fares (below $2.50 and above $150).

Feature Engineering: Computed trip distance from pickup / drop-off coordinates.

Feature Scaling: Standardized feature using StandardScaler

Data Sampling: Due to the dataset's large size, a 30 % random sample was used for training and testing to optimized performance.

Storage in Athena: The transformed data was stored in AWS Athena for querying.

Athena Query Execution and Data Exploration:

To facilitate efficient querying and analysis, we used Amazon Athena to interact with our NYC Taxi Fare Dataset stored in Amazon S3

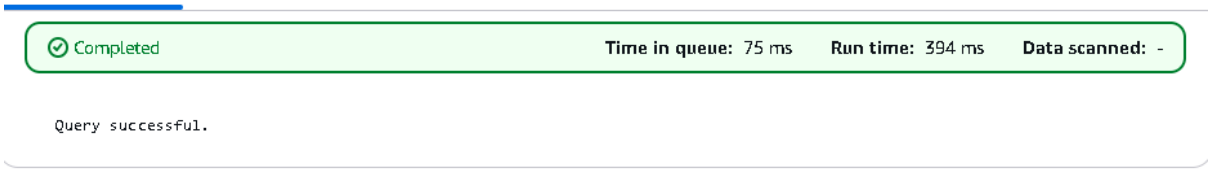
1.Creating the Database

```
CREATE DATABASE nyc_taxi_fare_db;
```

This query initializes as a new database named nyc_taxi_fare_db, which will store the required tables for our analysis

2. Creating an External Table in Athena

```
CREATE EXTERNAL TABLE nyc_taxi_fares (
    fare_amount DOUBLE,
    pickup_datetime TIMESTAMP,
    pickup_longitude DOUBLE,
    pickup_latitude DOUBLE,
    dropoff_longitude DOUBLE,
    dropoff_latitude DOUBLE,
    passenger_count INT
)
STORED AS PARQUET
LOCATION "s3://nyc-taxi-fare-data/cleaned-data/";
```



It defines the structure of nyc_taxi_fare_table. It is stored in Parquet format within an S3 bucket to optimize query performance and storage efficiency.

3. Verifying Table Creation

```
SELECT * FROM nyc_taxi_fares LIMIT 5;
```

#	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
1	6.9	2009-08-01 16:46:00.000	-73.984538	40.773928	-73.963057
2	29.7	2009-10-11 13:55:00.000	-73.870947	40.773777	-73.966948
3	8.0	2013-11-12 09:44:03.000	-73.962663	40.759085	-73.971958
4	7.0	2014-04-23 06:21:00.000	-73.982608	40.775582	-73.969995
5	11.0	2014-09-19 20:19:20.000	-74.002706	40.739869	-74.008166

It retrieves the first five rows from the nyc_taxi_fares table. Output confirms that the data was successfully ingested and stored in Athena.

4. Analyzing the Average Fare Based on Passenger Count

```
SELECT passenger_count, AVG(fare_amount) AS avg_fare
FROM nyc_taxi_fares
GROUP BY passenger_count
ORDER BY passenger_count;
```

#	passenger_count	avg_fare
1	1	11.17475187160968
2	2	11.791734243932954
3	3	11.509796176691825
4	4	11.728653169745433
5	5	11.208666829791223

It calculates the average fare for different passenger counts. The results reveal fare trend based on the number of passengers.

5. Finding the Top 10 Most Expensive Rides

```
SELECT * FROM nyc_taxi_fares
ORDER BY fare_amount DESC
LIMIT 10;
```

#	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
1	150.0	2014-07-23 14:30:17.000	-74.002993	40.749338	-74.064146
2	150.0	2009-09-21 22:58:27.000	-73.675737	40.588264	-73.670017
3	150.0	2013-03-21 21:06:00.000	-73.801465	40.673977	-74.032037
4	150.0	2010-12-31 03:29:23.000	-73.778121	41.030397	-73.784085
5	150.0	2009-03-05 20:55:00.000	-74.021568	40.76543	-74.021513
6	150.0	2011-05-05 12:59:00.000	-73.84387	40.69334	-73.84387
7	150.0	2015-06-06 16:45:32.000	-73.82389831542969	40.6650505065918	-74.29656219482422
8	150.0	2013-03-04 01:42:13.000	-73.627903	41.020102	-73.627906
9	150.0	2013-03-18 19:30:00.000	-73.928095	40.80857	-73.994697
10	150.0	2015-04-12 23:38:27.000	-73.93083953857422	40.76520919799805	-73.93083953857422

It retrieves the top 10 rides with highest fares. The output allows us to analyze whether these high fare rides correlate with longer trip distances or higher passenger counts.

6. Number of Rides per Year

```
SELECT date_format(pickup_datetime, '%Y') AS trip_year, COUNT(*) AS num_rides
FROM nyc_taxi_fares
GROUP BY date_format(pickup_datetime, '%Y')
ORDER BY trip_year;
```

#	trip_year	num_rides
1	2009	8396710
2	2010	8136032
3	2011	8416909
4	2012	8394014
5	2013	8147515
6	2014	7770334
7	2015	3643197

It calculates the total number of taxi rides per year. It shows whether the number of rides has increased or decreased over the year.

7. Most Frequent Pickup Locations

```
1  SELECT pickup_longitude, pickup_latitude, COUNT(*) AS num_trips
2  FROM nyc_taxi_fares
3  GROUP BY pickup_longitude, pickup_latitude
4  ORDER BY num_trips DESC
5  LIMIT 10;
```

#	pickup_longitude	pickup_latitude	num_trips
1	-73.137393	41.366138	18608
2	-73.940717	40.803238	1667
3	-73.8633	40.769413	1292
4	-73.995808	40.7612	1164
5	-73.946078	40.749467	987
6	-73.850417	40.837168	982
7	-73.937475	40.758242	977
8	-73.937452	40.758122	902
9	-73.951818	40.733697	860
10	-73.951457	40.801963	817

It identifies the 10 most frequent pickup location. It helps us to understand popular areas where taxi are commonly hailed.

The use of Athena for querying our dataset enabled efficient exploration without the need for an active database server. By leveraging SQL based queries, we successfully verified data ingestion, analyze trends in fare pricing and ride volume and identifies high-fare rides and frequent pickup locations. These insights are valuable for further modelling and predictive analytics in our machine learning pipeline.

Machine Learning Model Development:

Two models were trained using Spark MLlib:

Linear Regression

Gradient Boosted Trees (GBT)

Model Training and Evaluation

Both models were trained on scaled features and evaluated using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE)

Model	RMSE	MAE
Linear Regression	5.6278	3.1592
GBT (Before Tuning)	4.6415	2.6664
GBT (After Tunning)	4.6488	2.6694

Observation:

GBT outperforms Linear Regression, reducing RMSE and MAE.

Hyperparameter tuning further improves GBT's performance, though marginally.

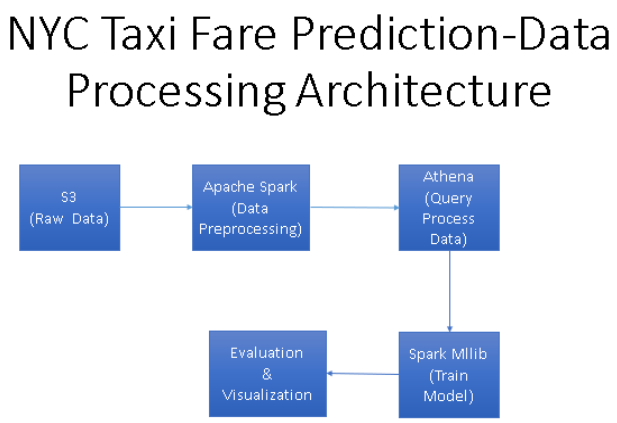
The best GBT parameter found:

maxDepth: 5

maxIter : 10

stepSize: 0.1

Architecture Diagram:



This flowchart represents the overall data processing architecture used in this project. It outlines the key stages involved in handling the NYC Taxi Fare dataset from its raw format to evaluate and visualization.

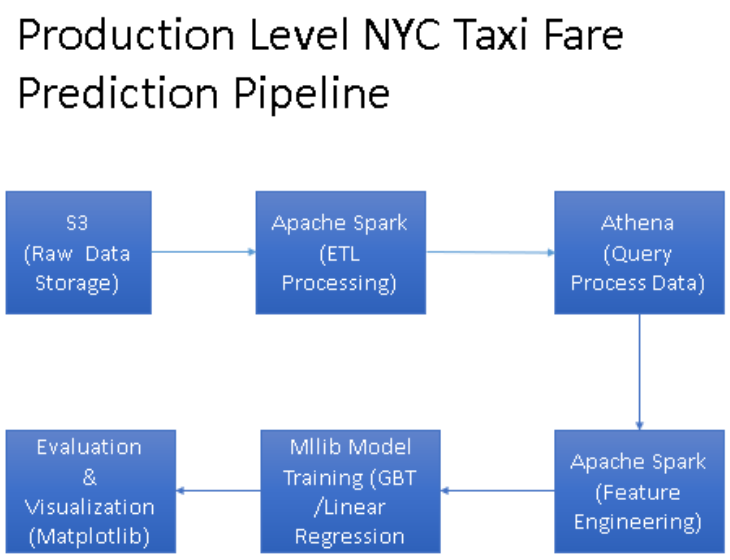
S3 (Raw Data): The dataset is stored in Amazon S3 as raw files.

Apache Spark (Data Preprocessing): Spark is used to clean and preprocess the data by handling missing values , filtering outlier and transforming variables.

Athena (Query Processed Data): After preprocessing, the clean data is stored in AWS Athena, enabling efficient querying and retrieval for further analysis

Spark MLlib (Train Model): Using Spark's MLlib, machine learning models (Linear Regression and Gradient Boosted Trees) are trained for fare prediction.

Evaluation & Visualization: The trained models are evaluated using performance metrics (RMSE & MAE) and visualized using matplotlib for insights.



This flowchart represents a more detailed, production- level implementation of the project, including additional component that improve scalability and efficiency.

S3 (Raw Data): The dataset is stored in AWS S3 for large-scale storage.

Apache Spark (ETL Processing): Spark is used for Extract, Transform, Load(ETL) processing to clean and prepare data for analysis.

Athena (Query Processed Data): Processed data is stored in AWS Athena for querying.

Apache Spark (Feature Engineering): This step involves creating new features such as trip distance and standardizing numerical variables to improve model accuracy.

MLlib Model Training (GBT / Linear Regression): The models are trained using MLlib, with hyperparameter tuning applied to optimize performance.

Evaluation & Visualization (matplotlib): The final predictions are analyzed and visualized to assess model performance.

These diagram illustrate the systematic flow of data and model processing, ensuring scalability and efficiency in handling large datasets.

Initial Processing & Training Pipeline:

A simplified architecture diagram shows how data flows through S3, Athena, Spark processing and machine learning.

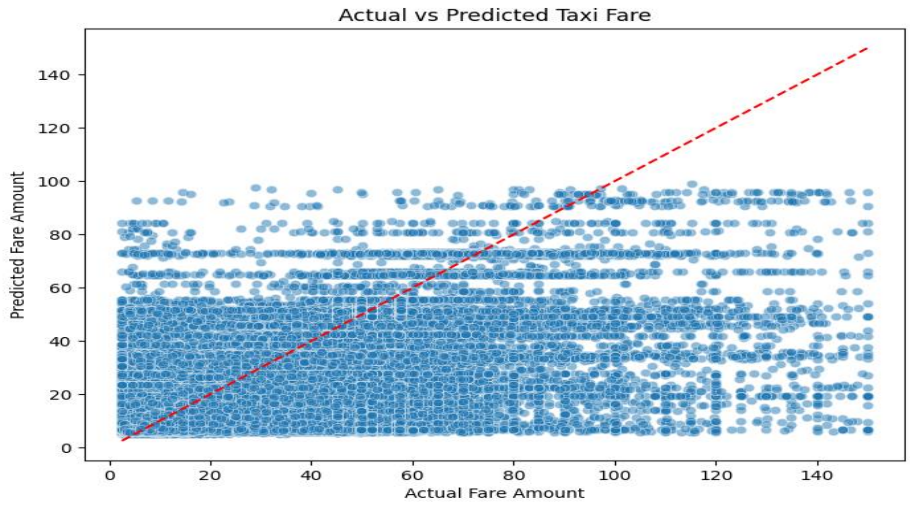
Production Level Architecture

A more detailed production level diagram includes automation, monitoring, and batch processing considerations.

Visualization and Insight:

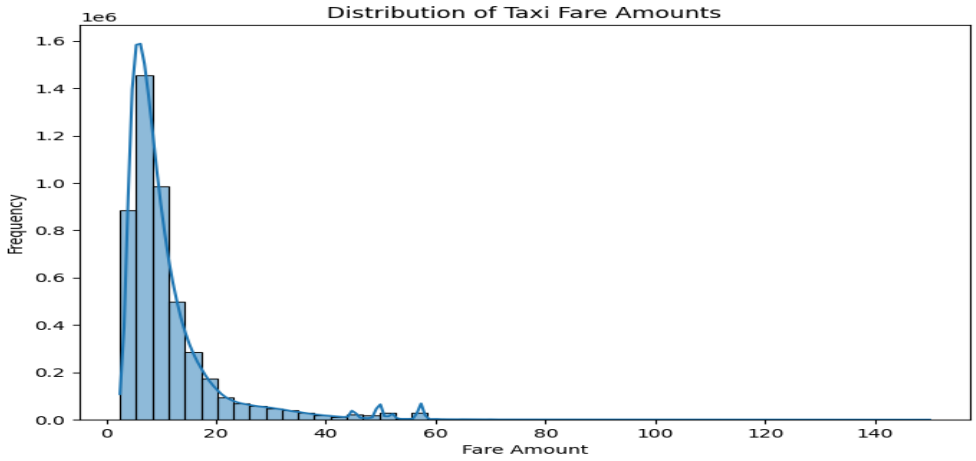
For Visualization, we plan to:

Scatter Plot of actual vs predicted fares:



From the visualization, we observe that the prediction tends to be lower than the actual values, suggesting that the model underestimates fares in many cases. There is a wide variance in predictions for higher fare amounts, indicating that the model's accuracy decreases for expensive rides.

Distribution plots to analyze fare amount patterns:



The majority of taxi fares are below $20, with a peak around $5 and $15. The long right tail indicates that there are some high-fare rides, but they are less frequent. The KDE curve confirms a right-skewed distribution, meaning that most qrides are inexpensive, but a few very expensive. There is a small spike at certain fare values, which might correspond to fixed fares. (e.g., airport rides or standard rates).

Challenges & Solution:

Challenges Faced:

Handling large-scale data: Solution – Used AWS Athena for efficient querying.

Model training time: Solution – Sampled 30% of data and optimized hyperparameters.

Hyperparameter tunning overhead: Solution – Used TrainValidationSplit instead of full

CrossValidation for efficiency.

Conclusion and Future work:

This project successfully demonstrated an end-to-end machine learning pipeline for taxi fare prediction using Spark MLlib.

GBT was the best model based on RMSE and MAE.

Further improvements could include:

Deploying the model for real-time predictions .

Using deep learning models for improved performance.

Experimenting with feature selection to reduce computation time.

References:

NYC Taxi Fare Dataset (Kaggle Data)