EAS587 DIC Phase 3 - Scalable Urban Mobility Analytics: Integrating Apache Spark and Generative AI for NYC Taxi Fare Prediction

Litheesh V Rambabu
Dept. IAD, SEAS
University at Buffalo
Buffalo, NY, USA
litheesh@buffalo.edu

Sai K V Indukuri
Dept. IAD, SEAS
University at Buffalo
Buffalo, NY, USA
sindukur@buffalo.edu

Gandhar Sidhaye
Dept. IAD, SEAS
University at Buffalo
Buffalo, NY, USA
gandhars@buffalo.edu

***Abstract*—This project implements an end-to-end big data pipeline to analyze New York City taxi trip [1] patterns and predict fares. Leveraging Apache Spark, we processed over 700,000 trip records from the 2023 Green Taxi dataset, implementing complex transformations and window functions to derive feature-rich datasets. To enhance predictive accuracy, we integrated a secondary dataset consisting of NOAA daily weather metrics [2] (precipitation, temperature, and snow). We trained and evaluated three distinct machine learning models—Linear Regression, Random Forest, and Gradient Boosted Trees (GBT)—using Spark MLlib's pipeline API. The Gradient Boosted Tree model achieved the highest performance with an R^2 of 0.66 and an RMSE of 10.92. Furthermore, we developed a "Smart Agent" using the Model Context Protocol (MCP), allowing Large Language Models (LLMs) like Claude to interact with our structured data, perform fuzzy location lookups, and execute real-time fare inference tools. This architecture demonstrates a scalable approach to combining traditional big data engineering with modern Generative AI applications.**

***Index Terms*—Data Science, Machine Learning, Urban Mobility, Predictive Modeling, NYC Taxi, Regression, Clustering, Model Context Protocol, Spark, Big Data Engineering**

I. Introduction

Urban mobility systems generate massive volumes of data that, when analyzed effectively, can optimize fleet management, inform urban planning, and improve passenger experiences. Accurately predicting taxi fares is challenging due to the stochastic nature of traffic, which is heavily influenced by external factors such as weather conditions and temporal demand spikes.

This project addresses two core challenges:

- **Scalability**: Processing large-scale trip data requires distributed computing frameworks rather than single-node solutions.
- **Accessibility**: Complex analytical models are often inaccessible to non-technical stakeholders (e.g., policy makers).

We propose a solution that combines Apache Spark [3] for high-throughput data processing and machine learning with an MCP Server to provide a natural language interface for model inference. By integrating NOAA weather data, we also quantify the impact of environmental conditions on transit behavior.

II. Methodology

A. Data Acquisition and Architecture

The pipeline utilizes two primary data sources:

- **Primary Source:** 2023 NYC Green Taxi Trip Records (780,000 rows). Key features include pickup/dropoff times, locations (IDs), trip distance, and total amount.
- **Secondary Source:** NOAA Daily Weather Data (Station JFK), providing precipitation (PRCP), snowfall (SNOW), and average temperature (TAVG).

The architecture follows a three-stage design:

- **Ingestion** & ETL: Spark DataFrames are used for cleaning, casting, and joining data.
- **Modeling:** Spark MLlib is used for feature vectorization and distributed training.
- **Serving:** An MCP Server [4] (Python) exposes the trained models and analytics via tools callable by an LLM.

B. Spark Data Pipeline (Part A)

We implemented a robust pipeline in 01_data_pipeline.py incorporating more than five distinct transformations:

- **Casting Parsing:** Converting string timestamps to Spark Timestamp types.
- **Filtering:** Removing invalid trips (e.g., negative fares, zero distance).
- **Binning:** Categorizing hour into "Rush Hour" vs. "Off Peak" using conditional logic.
- **Complex Joins:** A broadcast join was performed to enrich taxi data with location metadata (taxi_zone_lookup.csv) and weather data on the date key.
- **Window Functions:** We applied a window specification partitioned by Borough to calculate rolling average fares, enabling the detection of price volatility.

C. Multi-Source Integration (Part B)

To justify the enhancement of the analysis, we hypothesized that weather significantly impacts trip metrics. We performed an inner join between the cleaned taxi dataset and the NOAA weather dataset. This allowed us to correlate specific weather events (e.g., heavy snow) with trip volume and pricing.

D. Machine Learning Implementation

We utilized Spark MLlib's Pipeline API to ensure reproducibility. The feature engineering stage included StringIndexer for categorical zones and VectorAssembler for feature consolidation. We implemented three models:

- **Linear Regression:** A baseline parametric model.
- **Random Forest Regressor:** An ensemble method robust to outliers.
- **Gradient Boosted Trees (GBT):** A boosting method to minimize residual errors.

Cross-validation (3-fold) was employed to tune hyperparameters (maxDepth and numTrees).

E. GenAI Agent (MCP Server)

We developed a custom server implementing the Model Context Protocol. The server exposes six tools to the LLM, including predict_fare_basic, predict_fare_weather, and optimize_travel_time. It features a "fuzzy matching" algorithm to resolve natural language location names (e.g., "Soho") to official Zone IDs (e.g., 211) using the lookup dataset.

III. Results & Analysis

A. Performance Benchmarking

We compared the execution time of aggregating average fares by borough using Spark versus Pandas on the 780k row dataset.

- Spark Execution: 2.76 seconds
- Pandas Execution: 1.78 seconds

Analysis: While Pandas was faster for this specific volume (fitting easily in RAM), Spark provides the necessary infrastructure to scale to terabytes of data where Pandas would fail with memory errors.

B. Machine Learning Performance

Table 1 summarizes the performance of the models trained on the dataset.

Model	**RMSE**	R^2
Linear Regression	17.45	0.13
Random Forest (Taxi)	12.18	0.57
GBT Regressor	10.92	0.66
Random Forest (Weather)	15.10	0.51

Table 1: Model Evaluation Metrics

The Gradient Boosted Tree (GBT) was the superior model, explaining 66% of the variance in fares. Interestingly, the Weather-Enhanced Random Forest ($R^2 = 0.51$) performed slightly worse than the baseline RF ($R^2 = 0.57$). This is attributed to the reduced sample size used during the multi-source training phase to ensure memory stability on the local environment.

C. Multi-Source Insights

Integrating weather data yielded significant behavioral insights:

- **Rain Impact:** Rain increases average trip distance by 3.5% (20.63 miles vs. 19.9 miles), likely due to drivers taking alternative routes to avoid congestion.
- **Temperature Demand:** Taxi demand peaks in the 40°F–50°F range (176,525 trips), suggesting users prefer taxis over walking in moderately cold weather.
- **Snow Impact:** Heavy snow events (¿1 inch) reduced trip volume by 15%, yet the average fare remained stable ($\tilde{\$}21.56$), indicating consistent pricing algorithms even during adverse conditions.

IV. Discussion

A. Findings

The project successfully demonstrated that trip distance and location are the strongest predictors of fare. While weather integration provided analytical value (confirming behavioral changes), it did not drastically improve the regression metrics for this specific dataset. This suggests that the pricing formula used by Green Taxis is largely distance/time-dependent and less dynamic regarding weather surges than rideshare apps (e.g., Uber/Lyft).

B. GenAI Integration

The MCP server successfully bridged the gap between raw data and user intent. The agent correctly handled queries like "Compare fares from Soho to Tribeca" by running inference on both trained models and presenting a side-by-side comparison (e.g., calculating a "Rain Tax" of -$2.52 for specific routes).

C. Limitations & Future Work

- **Hardware Constraints:** The local Windows environment required data sampling for the complex join operations. Deploying to a cloud cluster (e.g., AWS EMR) would allow training on the full dataset.
- **Feature Engineering:** Future work could include "Trip Duration" as a predictor, though this requires estimation models since duration is not known pre-trip.

V. Conclusion

We successfully engineered a scalable pipeline using Apache Spark 3.5 to process NYC Taxi data. We fulfilled all objectives: implementing complex transformations, integrating secondary weather data, and training robust ML models ($R^2 = 0.66$). Finally, the implementation of the MCP Server transformed static predictions into an interactive AI agent, showcasing the potential of combining Big Data engineering with Large Language Models.

References

[1] NYC Taxi Limousine Commission, "TLC Trip Record Data," 2023. [Online].
[2] NOAA, "Climate Data Online: Daily Summaries," 2023.
[3] Apache Software Foundation, "Spark MLlib: RDD-based API," 2024.
[4] Anthropic, "Model Context Protocol Specification," 2024.

APPENDIX

A. Appendix A: Data Pipeline Snippet

(01_data_pipeline.py)

```
# Transformation: Window Function for Moving Average
windowSpec = Window.partitionBy("PU_Borough") \
                   .orderBy("parsed_datetime") \
                   .rowsBetween(-5, 0)
df = df.withColumn("Rolling_Avg_Fare", avg("total_amount").over(windowSpec))
```

B. Appendix B: MCP Tool Definition

(server.py)

```
@mcp.tool()
def compare_models(origin: str, destination: str) -> str:
    """Compare Basic vs Weather models to see the 'Rain Tax'."""
    pu, do = resolve_zone(origin), resolve_zone(destination)
    dist = get_dist(pu, do)

    p1 = models["gbt_basic"].predict([[dist, pu, do, 17, 3]])[0]
    p2 = models["rf_weather"].predict([[dist, pu, do, 1, 17, 3, 0.5]])[0]

    return f"Clear: ${p1:.2f} | Rain: ${p2:.2f}"
```