INNOPOLIS UNIVERSITY

Taxi Driver Pay Prediction

Team Members:

Anastasia Pichugina
Arthur Gubaidullin
Israel Adewuyi
Anna Gromova

Introduction to Big Data

May 9, 2025

Contents

1 Introduction

This project focuses on developing a predictive model to estimate driver earnings (`driver_pay`) for Taxi trips in New York City. Using the For-Hire Vehicles Trip dataset, we are analyzing pricing patterns and identifying potential seasonal and other factors that may be hidden in taxi's pricing algorithms.

Traditional pricing models can sometimes not fairly account for different factors and lack transparency. Our aim is to address this critical business problem by creating a prediction model for taxi driver pays. The ability to accurately predict driver pay could help taxi companies optimize their pricing methods and ensure fair pays for drivers as well as competitive rates for passengers. Furthermore, this analysis may reveal inconsistencies or inefficiencies in the current pricing model that could be addressed to improve service quality.

2 Data Description

The Uber NYC for-hire vehicles trip dataset includes trip records from high-volume for-hire vehicles (FHVHVs) in New York City during 2021, specifically focusing on Uber trips. The NYC Taxi and Limousine Commission (TLC) collected and released this data for research purposes.

The main part of the dataset contains 24 columns of detailed trip information including precise timestamps for key events (request, on-scene, pickup, and dropoff times), location data linked to taxi zones, and various trip attributes. The target variable for our prediction model is `driver_pay`. Complementary data includes taxi zone geospatial data (`taxi_zones.*` files), taxi zone lookup table (`taxi_zone_lookup.csv`), and NYC weather records for 2021 (`nyc 2021-01-01 to 2021-12-31.csv`).

The complete dataset has 4.56 GB of information across the total of 49 features of different data types. The trip information from 2021 is stored in Parquet format files organized by month (e.g., `fhvhv_tripdata_2021-01.parquet`), while weather records and taxi zone lookup table are stored in CSV files.

The size and complexity of the dataset require careful consideration of computational resources and processing strategies. Therefore, for the purpose of our project, we decided to use data collected for 3 consecutive months, from January to March, stored in files `fhvhv_tripdata_2021-01.parquet`, `fhvhv_tripdata_2021-02.parquet`, and `fhvhv_tripdata_2021-03.parquet` along with complementary weather data and geospatial information, which allows us for multidimensional analysis that could reveal important patterns in driver earnings.

3 Pipeline Architecture

3.1 Stage 1: Data Collection & Ingestion

- Collecting the dataset from Yandex disk and performing initial preprocessing
- Building a relational model and loading data to the relational database in **PostgreSQL** with tables:
  - `trips`

  - `taxi_zones`
  - `weather`
- Importing the data from the relational database to HDFS with **Sqoop** using:
  - **Avro** row-based data format
  - **Snappy** compression

3.2 Stage 2: Data Storage/Preparation & EDA

- Creating efficient **Hive Tables** and storing the data in the data warehouse Hive for further analytics, while:
  - **Partitioning** `trips_part` by `dispatching_base_num`
  - **Bucketing** by location IDs into 8 buckets
    Optimization: Faster joins on frequent location-based queries
- **Exploratory Data Analysis**

3.3 Stage 3: Predictive Data Analysis

- **Feature engineering**:
  - Joining tables and dropping unnecessary columns
  - Handling missing data (dropping columns with substantial amout of missing values)
  - Cyclical encoding (sine/cosine for day and month)
  - One-hot encoding categoricals (borough, service zone, HVFHS license number)
  - Scaling features for Linear Regression
- Spark **ML models** (Linear Regression, Gradient Boosted Decision Tree) building and training
- Optimizing **hyperparameters**
- **Evaluating** models

3.4 Stage 4: Presentation

- Presenting the analysis and prediction results in **Apache Superset dashboard** with:
  - Data characteristics and description
  - Charts with EDA insights
  - Model results and performance comparisons

Stage	Input	Output
Data Collection & Ingestion	Raw Parquet/CSV files from Yandex Disk (downloaded from Kaggle)	PostgreSQL tables (trips, taxi_zones, weather), Avro files in HDFS (Snappy compressed)
Data Storage and Preparation & EDA	Avro files in HDFS, PostgreSQL schema	Partitioned/bucketed Hive tables, EDA results (8 insights), Query outputs (CSV)
Predictive Data Analysis	Hive tables, Data insights	Processed feature vectors, Trained ML models, Evaluation metrics (RMSE/MAE/R^2)
Presentation	Model results, EDA findings	Interactive Superset dashboard, Data stories, Performance visualizations

Table 1: Inputs and Outputs of Pipeline Stages

4 Data Preparation

4.1 Data Collection and Initial Processing

We started by collecting taxi trip data from three Parquet files (January-March 2021), weather data in CSV, and taxi zone information, and uploading this data to the Yandex disk.

Our preprocessing script (`preprocess.py`) handled several critical tasks. It processed files in 2000-row batches to manage memory usage, taking the first 2,450,000 rows of data. This data collection resulted in obtaining data from 8 consecutive days of January period. The script filtered out records with missing request timestamps, converting the feature to a more convenient format, and ensured all trips occurred in 2021. We also dropped `access_a_ride_flag` column containing no 'Y' values. Finally, we converted Parquet files to CSV format for easier PostgreSQL import.

For the weather data, we kept all 365 days of records but removed unnecessary columns during later stages. The taxi zone data required no preprocessing.

4.2 PostgreSQL Database Design

During the next step of data preparation, we created three main tables in PostgreSQL with proper constraints:

- `trips` - core trip data with foreign keys to two other tables
  - Primary key: `trip_id`
  - Foreign key for `weather`: `request_date` (`date_id` in `weather`)
  - Foreign keys for `taxi_zones`: `pu_location_id` and `do_location_id` (`location_id` in `taxi_zones`)
- `taxi_zones` - for pickup/dropoff locations
  - Primary key: `location_id`

- `weather` - for year-round weather data
  - Primary key: `date_id`

The schema enforced data integrity through primary keys on all tables, foreign key constraints for location and date references, and appropriate data types (TIMESTAMP for times, FLOAT for monetary values).

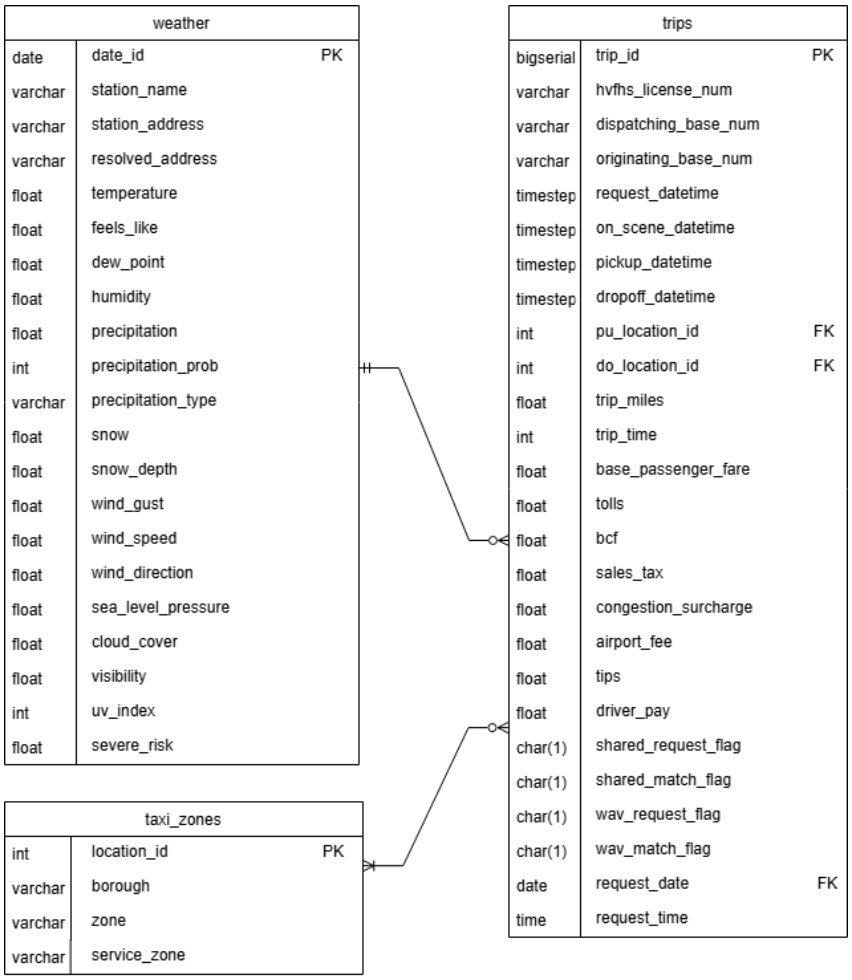

Figure 1: Entity-Relationship Diagram

	location_id [PK] integer	borough character varying (50)	zone character varying (100)	service_zone character varying (50)
1	81	Bronx	Eastchester	Boro Zone
2	220	Bronx	Spuyten Duyvil/Kingsbridge	Boro Zone
3	203	Queens	Rosedale	Boro Zone
4	159	Bronx	Melrose South	Boro Zone
5	178	Brooklyn	Ocean Parkway South	Boro Zone

Figure 2: Taxi zone sample data

	station_name character varying (50)	station_address character varying (100)	resolved_address character varying (100)	date_id [PK] date	temperature double precision	feels_like double precision	dew_point double precisi
1	nyc	nyc	New York, NY, United States	2021-07-17	27.5	29.3	
2	nyc	nyc	New York, NY, United States	2021-09-16	23.3	23.3	
3	nyc	nyc	New York, NY, United States	2021-11-06	8.1	7.3	
4	nyc	nyc	New York, NY, United States	2021-10-11	19	19	
5	nyc	nyc	New York, NY, United States	2021-03-15	-0.2	-5.7	

Figure 3: Weather sample data

4.3 Data Ingestion into Hadoop

We used **Sqoop** to transfer data from PostgreSQL to HDFS with two optimizations techniques. We utilized **Avro** data format, which preserves schema information and supports schema evolution, which makes it an efficient row-based storage. For compression

we have chosen to use **Snappy**, because it provides faster decompression compared to GZIP (better for frequent reads) and minimal CPU overhead during compression. This combination proved great for our analytical workload, balancing storage efficiency with query performance.

The Sqoop command for loading tables to HDFS:

```
sqoop import --connect jdbc:postgresql://... --compression-codec=snappy
--compress --as-avrodatafile --table <table_name>
```

4.4 Hive Data Warehouse Implementation

In **Hive**, we created an optimized data warehouse structure that would support efficient analytical queries. Our implementation included two key optimizations. First, we **partitioned** the trip data by `dispatching_base_num`, which allows the query engine to skip irrelevant data when filtering by dispatch company. Second, we applied **bucketing** to the location IDs, dividing them into 8 buckets to accelerate location-based joins by collecting related records.

The Hive tables were created using the Avro schemas imported from HDFS. This ensured data consistency across all layers of our pipeline. We enabled dynamic partitioning to automatically handle new dispatch bases that might appear in future data updates. The used approaches significantly improves query performance.

Sample Hive Table Creation:

```
CREATE EXTERNAL TABLE trips_part (...)
PARTITIONED BY (dispatching_base_num STRING)
CLUSTERED BY (pu_location_id, do_location_id)
INTO 8 BUCKETS STORED AS AVRO;
```

5 Data Analysis

Our exploratory data analysis focused on understanding patterns in taxi trip data from January 2021 (8 days). We used Hive queries on our optimized data warehouse to reveal several key insights for both business strategy and our modeling approach.

The dataset contained 2.4 million trip records for four companies (Uber, Lyft, Via, and Juno), with weather data for 365 days and 265 taxi zones. We observed an average of 306,000 trips per day during our analysis period, indicating substantial demand. This volume justified our distributed processing approach.

The hourly trip heatmap revealed that the busiest times were during normal commute hours with morning and evening peaks, but also showed sustained demand throughout midday. This insight directly influenced our feature engineering decision to create cyclical time features.

Weather played an important role too. When it rained, prices went up by about 12%, and during colder temperatures, prices were also higher. This finding told us that bad weather makes people willing to pay more for rides. It also validated our decision to include weather features for prediction.

Looking at which companies provided the rides, Uber (HV0003) dominated with 72.5% of trips, followed by Lyft (HV0005) (26.55%) and Via (HV0004) (0.95%). This distribution suggests potential pricing power for the market leader. The company feature may also correlate with pay structures and prices.

The most popular drop-off locations were in Brooklyn and Manhattan. Brooklyn accounted for more than 300k dropoffs alone, with Crown Heights North and East New York being the most popular destinations. Knowing where demand is highest helps taxi companies position drivers where they're needed most. These location patterns guided our feature selection and the use of location-based features to predict driver pay.

We used a radar chart to compare trip characteristics (count, miles, duration) across tip categories (No Tip, Low Tip, Medium Tip, High Tip) and to highlight differences in behavior based on tipping habits. The chart revealed insights into customer behavior and service quality valuable for business stakeholders. For modeling, tip patterns directly affect driver pay, and trip duration/distance are key predictors.

Prices changed throughout the day, peaking at 5 PM when the average trip costs $22.78, compared to just $15.88 at 11 PM. Hourly pricing trends help businesses implement dynamic pricing strategies and optimize revenue. Time-based cost features (e.g., average cost per hour) can enhance predictions by capturing temporal variations in trip earnings.

We also determined that trips to Newark Airport took the longest - 36 minutes on average, which is twice more than any other borough. Understanding trip durations by borough helps businesses optimize fleet allocation.

6 ML Modeling

6.1 Feature Extraction and Data Preprocessing

Our machine learning pipeline starts with comprehensive feature engineering to prepare the data for modeling. All the columns with missing values had a substantial amount of data absent, so to handle this issue we dropped 3 columns from `trips` (`originating_base_num`, `on_scene_datetime`, and `airport_fee`) and 3 columns from `weather` (`precipitation_type`, `wind_gust`, and `severe_risk`). The `taxi_zone` had no missing values.

Day and month were extracted from `pickup_datetime` (`trips`) and `date_id` (`weather`) and used to join `trips` with `weather` on.

We dropped the original zone names (`zone`) due to high cardinality (262 unique values out of 265 rows in `taxi_zone`) which would have created excessive sparse features. Then we joined `taxi_zone` with a combined dataframe of `trips` and `weather` on location IDs of the start and end zones.

For temporal features, we implemented cyclical encoding for day and month using sine/cosine transformations. This approach preserves the cyclical nature of time data while making it interpretable to linear models. The custom transformer we developed converted day numbers into two features (`day_sine`, `day_cosine`) and similarly for months, capturing seasonal patterns in driver payments.

Categorical features were handled through one-hot encoding for `borough` (7 categories), `service_zone` (5 categories), and license numbers (`hvfhs_license_num`) (3 categories).

Finally, we splitted the data into 1,715,526 training samples (70% of data) and 734,798 test samples (30% of data).

The preprocessing pipeline joined trip data with weather and taxi zone information while handling missing values and data formats and dropping irrelevant columns like trip ID, exact timestamps, and location IDs after encoding.

6.2 Model Training and Fine-tuning

We selected two regression approaches to predict driver payments: Linear Regression and Gradient Boosted Trees (GBT). This combination allowed us to compare a simple interpretable model with a more complex ensemble method. The training data (70% of samples) was standardized for Linear Regression but left unscaled for GBT which is scale-invariant.

For hyperparameter optimization, we implemented 3-fold cross-validation with grid search:

- **Linear Regression**: Tested regularization strength (0.001, 0.01, 0.1, 1.0) and elastic net mixing (0.0, 0.25, 0.5, 0.75, 1.0)
- **GBT**: Evaluated tree depths (3, 5) and learning rates (0.05, 0.1)

The grid search evaluated 20 parameter combinations for Linear Regression and 4 for GBT, optimizing for Root Mean Squared Error (RMSE) metric. We parallelized the search across 4 executors to accelerate the process. The best Linear Regression model used L2 regularization ($\alpha = 0.1$) while the optimal GBT used depth=5 trees with learning rate=0.1.

6.3 Model Evaluation

Both best models obtained hyperparameter tuning were evaluated on the 30% test set using three metrics:

Model	**RMSE**	**MAE**	$\mathbf{R^2}$
Linear Regression	3.237	1.404	0.916
GBT	3.988	1.533	0.873

The Linear Regression surprisingly outperformed GBT on all metrics. This suggests that driver payments follow relatively linear relationships with the features in our dataset. The high R^2 values (0.85+) indicate both models explain most variance in payments. The Linear Regression was selected as our final model due to its superior performance and simpler interpretation.

We also analysed the feature importance for two models. Both Linear Regression and GBT had base_passenger_fare, trip_time, and trip_miles among the top-3 most important factors for predicting pay.

The results suggest taxi's payment formula likely incorporates linear combinations of distance, time, and location factors. The model could help identify underpaid trips or optimize pricing by predicting expected driver earnings under different conditions.

7 Data Presentation

We created the Apache Superset dashboard with three main sections to share our findings. The first section describes the three datasets, their columns and number of entries for trips, weather data, and taxi zones in New York.

The second section shows interactive charts of our key findings:

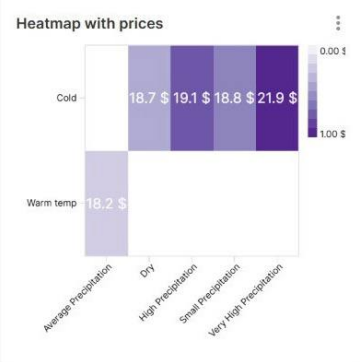

Figure 4: Price Heatmap by Weather Conditions

Weather-Price Relationship (Fig 4) shows how precipitation and temperature affect fares. Empty cells represent missing combinations in our 8-day sample. The darkest cells show that prices are highest when it's both cold (below 5°C) and rainy. This visualization helps taxi provides adjust their pricing algorithms based on weather conditions.

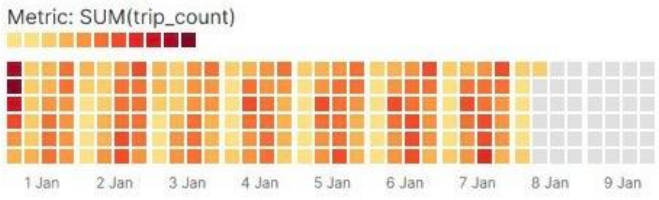

Figure 5: Trip Volume by Hour

Hourly Demand Patterns (Fig 5) shows the number of trips by hour and day with darker cells having more rides. We clearly see the highest demand during morning and evening rush hours. This helps companies know when to have more drivers available.

Competitive Landscape (Fig 6) shows a pie chart showing Uber(HV0003) dominates with 72.5% of trips, Lyft(HV0005) has 26.55%, and Via(HV0004) only 0.95%. This helps understand competition in the market and identify growth opportunities.

Tips (Fig 8) highlights that higher tips do not correlate with longer trips. Medium tips have the highest average trip distance and time. This helps understand customer behavior.

Trip cost per hour (Fig 9) shows that prices peak at 5 PM ($22.78 average) and are lowest at 23 PM ($15.88). This supports dynamic pricing strategies.

Trip duration by borough (Fig 10) bar chart informs that trips to Newark Airport take longest (36 minutes), while other destinations take twice less time.

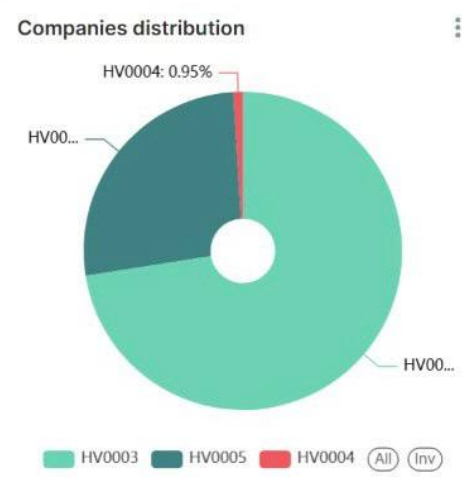

Figure 6: Market Share by Company

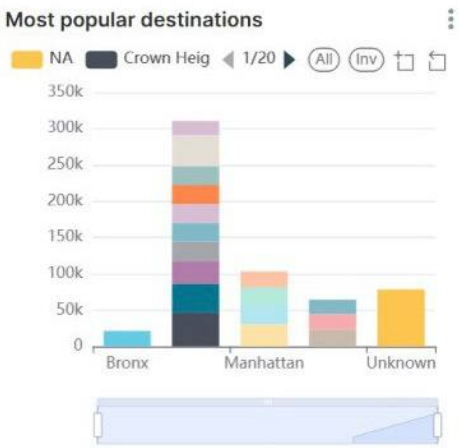

Figure 7: Top Drop-off Locations

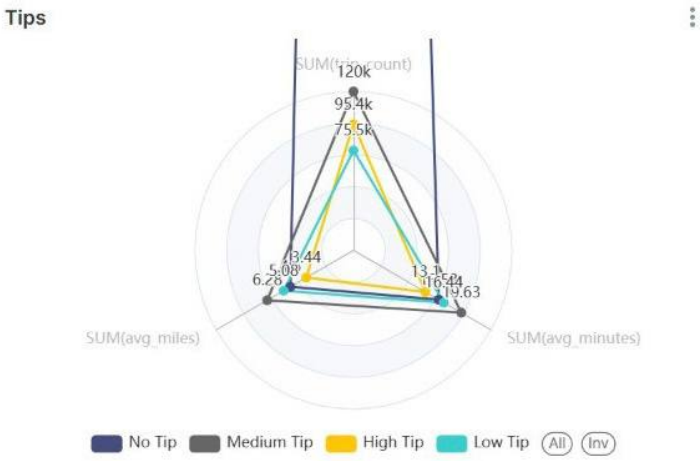

Figure 8: Tips

All visualizations were powered by Hive queries. The 8-day holiday focus provided a manageable sample while capturing unique seasonal patterns valuable for New Year's operational planning.

The last section focused on our modeling and evaluation results. We displayed the performances of our best models along with comparison of their predictions with the actual target values (Fig 11 and Fig 12). We also included the analysis of the importance of the characteristic (Fig 13 and Fig 14) for linear regression and gradient boosted trees, showing the top-3 factors impacting the driver's pay, trip distance, duration, and base fare amount.

These results help businesses to set fair driver payments based on distance, time, and demand, adjust prices during bad weather or rush hours, as well as position drivers in high-demand areas like Brooklyn and understand when customers are most likely to tip well.

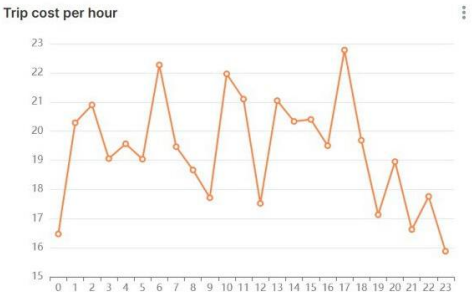

Figure 9: Trip cost per hour

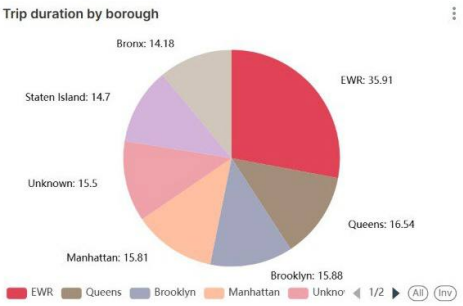

Figure 10: Trip duration by borough

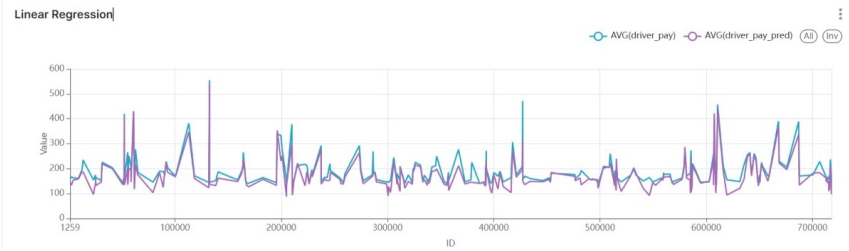

Figure 11: LR Predictions

8 Conclusion

This project successfully implemented an end-to-end big data pipeline for analyzing and predicting taxi driver payments in New York City. We processed over 2.4 million trip records from January 2021 through a robust architecture combining PostgreSQL, HDFS, Hive, and Spark. Our implementation demonstrated how distributed computing can efficiently handle large-scale transportation data while providing actionable business insights.

We developed an optimized data warehouse with partitioned and bucketed Hive tables, engineered meaningful features including cyclical time encodings and weather-related variables that captured usefull patterns in driver payments. We built predictive models that achieved the 0.916 R^2 score, with Linear Regression outperforming Gradient Boosted Trees. To visualize our results, we created an interactive dashboard with data insights and model performances.

The analysis revealed that driver payments follow relatively linear relationships with trip characteristics. Our models can help identify pricing anomalies and optimize driver earnings.

Dashboard is available at the link.

9 Reflections

9.1 Challenges and Difficulties

The project presented several challenges that provided valuable learning experiences. Processing large amounts of raw trip data required careful optimization of our Spark

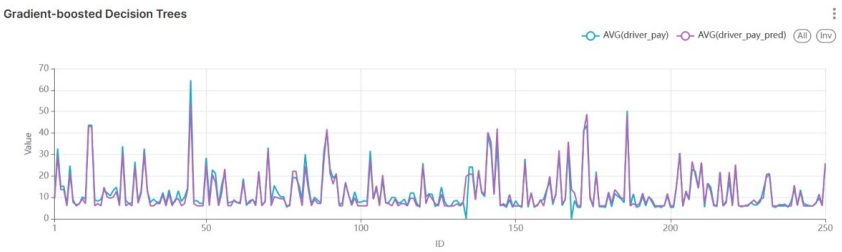

Figure 12: GBT Predictions

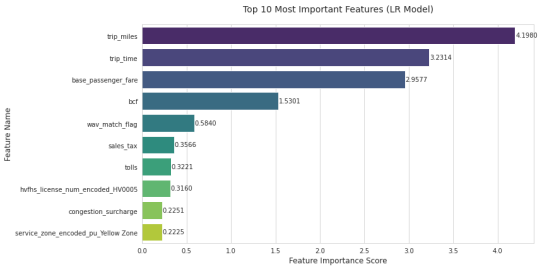

Figure 13: LR Feature Importance

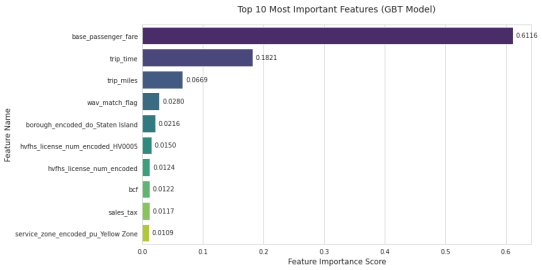

Figure 14: GBT Feature Importance

jobs and Hive table structures. Sparse features and multiple tables required carefull preprocessing and alignment. Shared cluster resources sometimes led to delays, requiring us to optimize our pipeline.

9.2 Recommendations

For future work we recommend investigating pricing on a larger datasets beyond an 8-day period and New York City and incorporating additional data sources like traffic patterns or events calendars. This analysis could reveal more significant trends as well as address the problem on a more global scale.

9.3 Table of Contributions

Team Member Key:

- M1 - Anastasia Pichugina
- M2 - Arthur Gubaidullin
- M3 - Israel Adewuyi
- M4 - Anna Gromova

Project Tasks	**Task Description**	**M1**	**M2**	**M3**	**M4**	**Deliverables**	**Hours**
Data Collection and Preprocessing	Collect data from source and preprocess raw data	100%	0%	0%	0%	trips_month.csv, taxi_zone.csv, weather.csv	2.0

Project Tasks	**Task Description**	**M1**	**M2**	**M3**	**M4**	**Deliverables**	**Hours**
PostgreSQL DB	Creating tables, importing data, testing DB	100%	0%	0%	0%	create_tables.sql, import_data.sql	4.0
Data Ingestion	Creating Hive tables	100%	0%	0%	0%	db.hql, hive_results.txt	2.0
Data analysis	Creating DB queries and 8 charts	100%	0%	0%	0%	Apache Superset charts, q1-8.hql	16.0
Feature Engineering	Prepare data for modeling	0%	100%	0%	0%	Preprocessed train/test data	6.0
ML modeling	Develop ML models	0%	100%	0%	0%	Trained models	10.0
Dashboard	Create visualization dashboard	0%	0%	100%	0%	Dashboard	7.0
Report writing	Compile final report	0%	0%	0%	100%	report.pdf	14.0
Repository	GitHub repo creation and organizing	0%	0%	0%	100%	GitHub repository	1.0
Presentation	Creating slides for defence	0%	0%	0%	100%	presentation.pptx	2.0

Table 2: Team Contributions