DOORDASH

We enable every restaurant to deliver!

Team

Dheeraj

Achintya

Yujata

Agenda

- Problem Statement
- Methodology
    - Infrastructure Setup
    - Dataset Selection
    - Data Split & Preparation
    - Exploratory Data Analysis
    - Model Pipeline & Training
    - Pipeline Working Demo
    - Model Performance Monitoring
    - Feature Modification Impact
- Conclusion
- Contributions

DOORDASH

Problem Statement

DOORDASH

Goal

- DoorDash seeks to enhance customer satisfaction by improving delivery time estimates
- Current inaccuracies cause frustration and resource waste
- This project deploys a machine learning pipeline to predict delivery times accurately and maintain consistency through continuous monitoring

Challenges

- Data Quality Issues
- Target Variable Calculation

Expected Outcomes

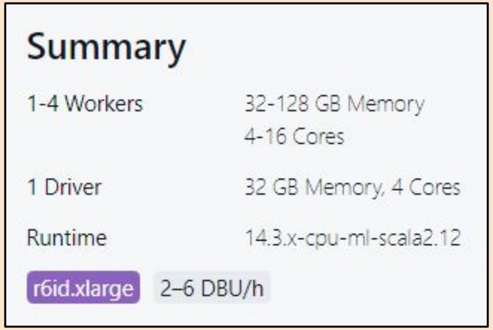
- Improved ETA Accuracy
- Improved Experience and Efficiency
- Model Deployment and Monitoring

Methodology

Infrastructure Setup

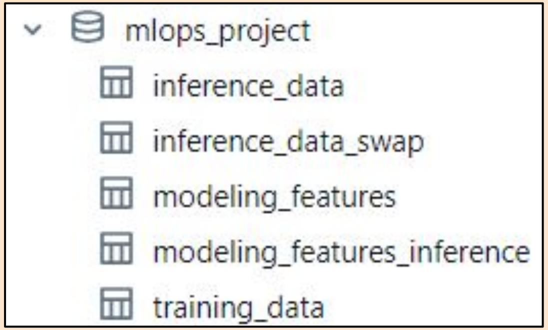
DOORDASH

1. Cluster Setup - for executing AutoML & Workflows



- DataBricks runtime 14.3 ML makes it possible to execute AutoML and WorkFlows
- Runtimes greater than 14.3 ML has Python version 3.11 with which Evidently AI is not compatible.

2. Unity Catalog - Delta Tables - Feature Store



- training_data, version 0 → raw training data
- training_data, latest version → processed training data
- modeling_features (feature store) → final features for modeling
- inference_data, version 0 → raw inference data
- inference_data, latest version → processed inference data
- modeling_features_inference → final features for predicting during inference

Data Overview

DOORDASH

Column Name	# Records	# Missing Records	Missing Records %	Data Type
market_id	186613	943	0.502783	float64
created_at	187556	0	0.000000	object
actual_delivery_time	187549	7	0.003732	object
store_id	187556	0	0.000000	int64
store_primary_category	182883	4673	2.491523	object
order_protocol	186598	958	0.510781	float64
total_items	187556	0	0.000000	int64
subtotal	187556	0	0.000000	int64
num_distinct_items	187556	0	0.000000	int64
min_item_price	187556	0	0.000000	int64
max_item_price	187556	0	0.000000	int64
total_onshift_dashers	172072	15484	8.255668	float64
total_busy_dashers	172072	15484	8.255668	float64
total_outstanding_orders	172072	15484	8.255668	float64
estimated_order_place_duration	187556	0	0.000000	int64
estimated_store_to_consumer_driving_duration	187047	509	0.271386	float64

- **Data Source**: kaggle
- **About**: DoorDash Deliveries
- **Duration**: 4 Months
- **Features**: 16
- **Rows**: ~190K

Data Preparation

Used for EDA and model training

Used for inference

Made a copy of the ***inference data*** *and swapped 'total_items' and 'subtotal' to create* ***inference data swapped****.*

Preprocessed and Engineered 95% of Original Dataset: 34% Reduction from 180K to 118K Records

DOORDASH

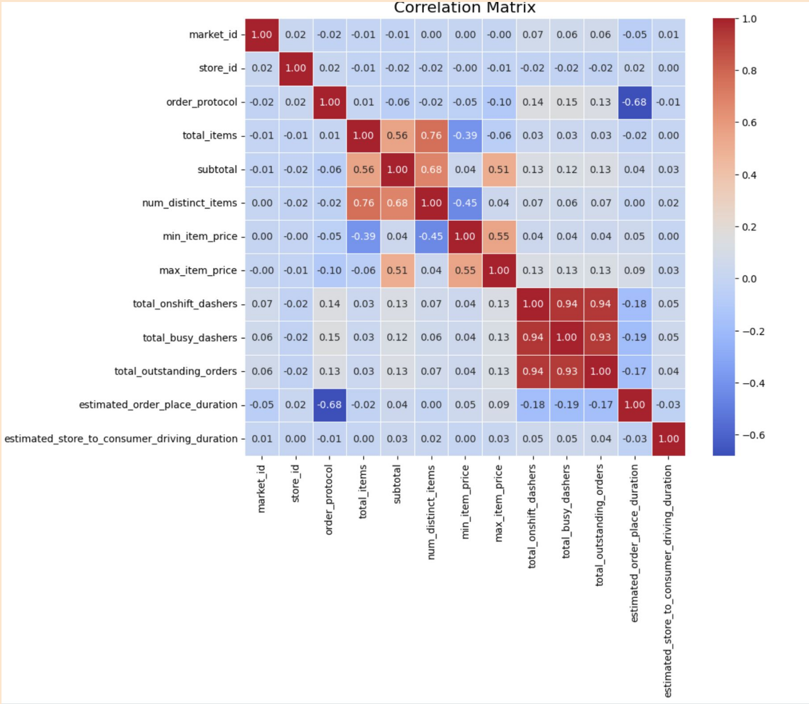
Data Preprocessing

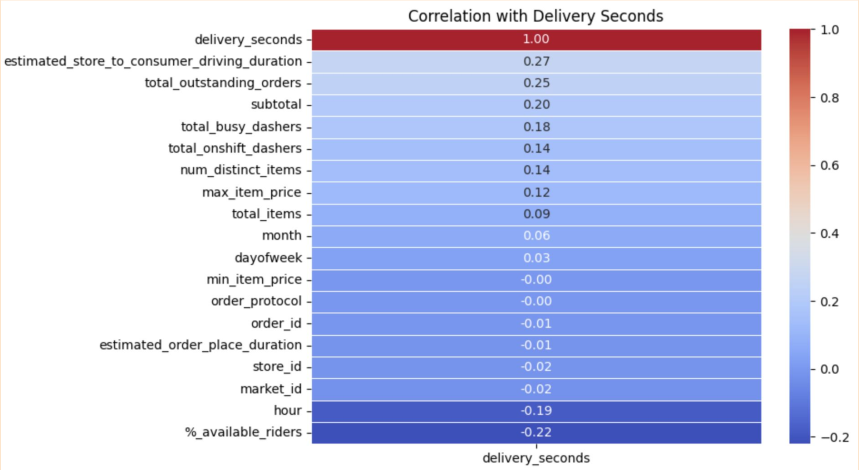
- Missing value treatment
  - Fills missing store_primary_category values based on the most common category per store_id
  - Fills missing market_id values based on the most common market_id per store_id
- Data sanity check
- Outlier removal(*'delivery_seconds', 'subtotal', 'max_item_price'*)
  - Using IQR approach
- Feature engineering
  - (month, dayofweek, hour, %_available_riders, delivery_seconds)

Version	# Outliers	% Outliers
Raw	42463	22.640171
Processed	19364	16.362745

Column Name	# Records	# Missing Records	Missing Records %	Data Type
order_id	118342	0	0.0	int64
market_id	118342	0	0.0	int64
created_at	118342	0	0.0	datetime64[ns]
actual_delivery_time	118342	0	0.0	object
store_id	118342	0	0.0	int64
store_primary_category	118342	0	0.0	object
order_protocol	118342	0	0.0	float64
total_items	118342	0	0.0	int64
subtotal	118342	0	0.0	int64
num_distinct_items	118342	0	0.0	int64
min_item_price	118342	0	0.0	int64
max_item_price	118342	0	0.0	int64
total_onshift_dashers	118342	0	0.0	float64
total_busy_dashers	118342	0	0.0	float64
total_outstanding_orders	118342	0	0.0	float64
estimated_order_place_duration	118342	0	0.0	int64
estimated_store_to_consumer_driving_duration	118342	0	0.0	float64
month	118342	0	0.0	int64
dayofweek	118342	0	0.0	int64
hour	118342	0	0.0	int64
%_available_riders	118342	0	0.0	float64
delivery_seconds	118342	0	0.0	float64

Delivery Seconds: Strongly Affected by Driving Duration and Outstanding Orders





- **Correlation Matrix:**

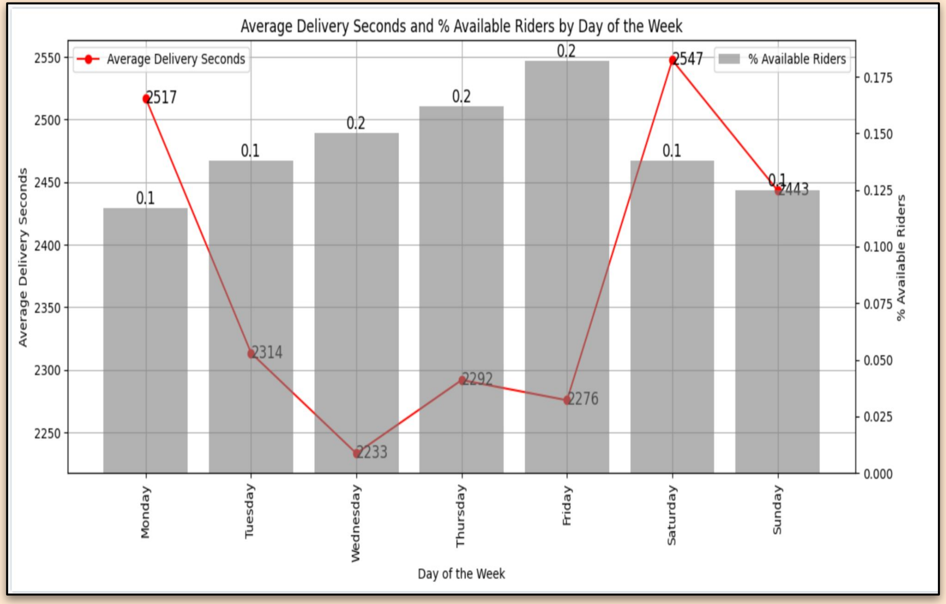
The total_onshift_dashers, total_busy_dashers, and total_outstanding_orders are highly correlated, while estimated_order_place_duration is negatively correlated with order_protocol. Other features have weak correlations

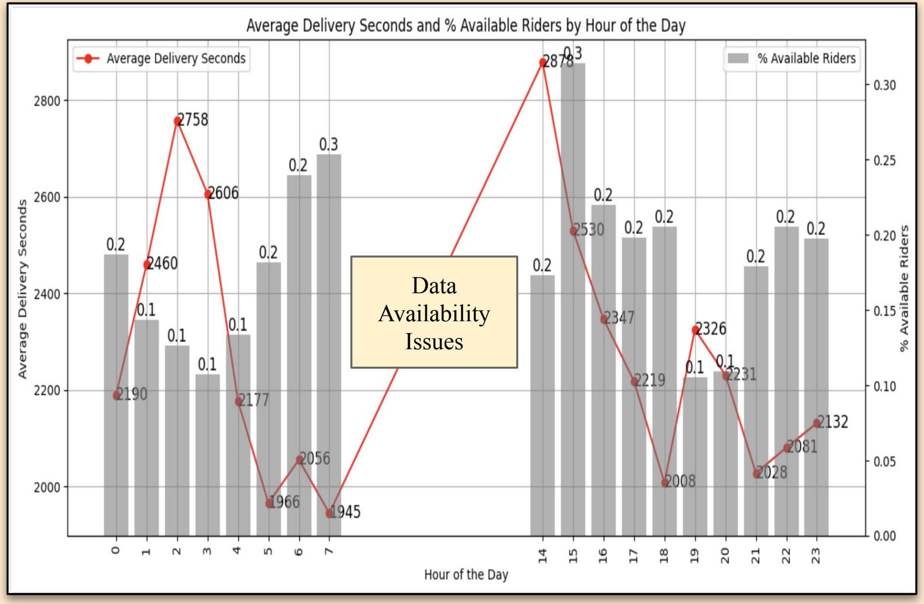
*Deriving **"delivery_seconds":** It is the time from order creation to delivery, excluding the estimated time to place the order*

- **Correlation with Delivery Seconds :**

Delivery time is strongly influenced by driving duration and outstanding orders, while features like minimum item price and order protocol have little impact. Faster deliveries are observed during certain hours and when more riders are available

Peak Times and Delivery Challenges: A Weekday and Hourly Analysis

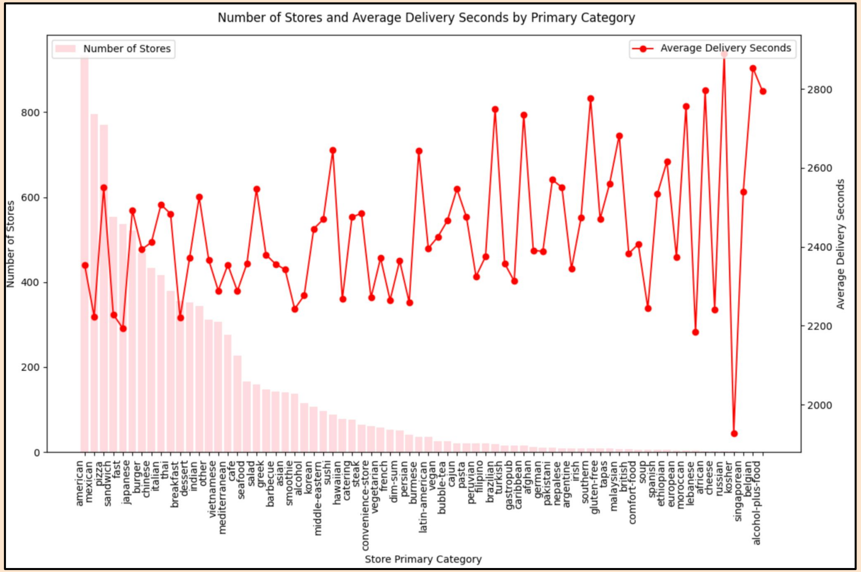


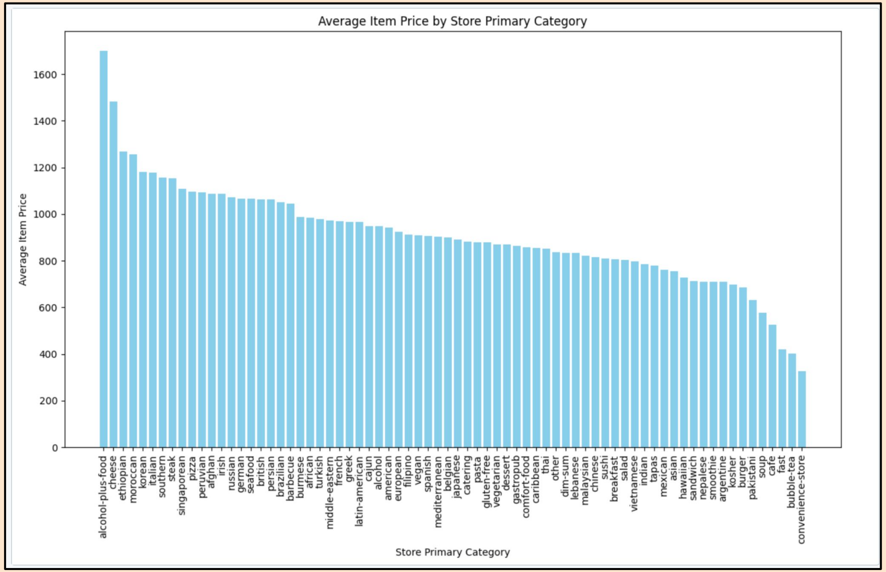


The *lowest average delivery times occur on Wednesdays* (2233 seconds), while the *highest are on Saturdays* (2547 seconds). The significant increase in orders and possible traffic congestion on Saturdays leads to longer delivery times, *despite more riders being available*

Average delivery times spike significantly at hour 14 (2758 seconds) and *decrease sharply by hour 19* (2056 seconds), with *available rider percentages fluctuating*, suggesting peak demand or rider scarcity during certain hours

Distribution of Stores by Category and Service Attributes





Number of Stores

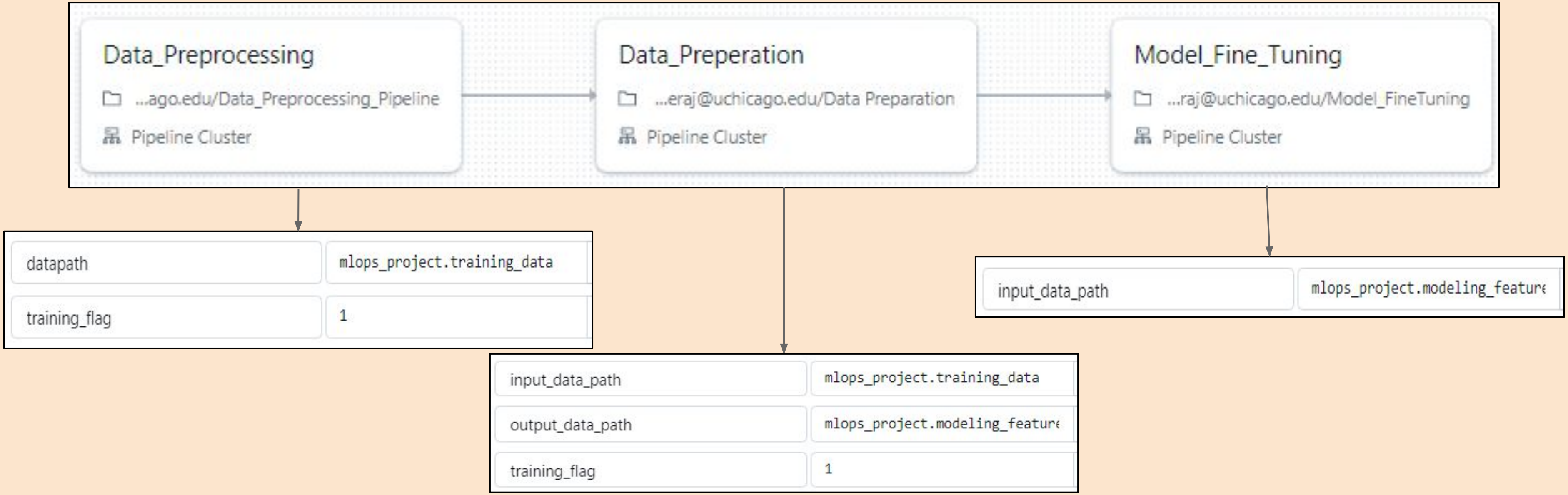
- American, Mexican, Japanese, and Sandwiches have *500+ stores*
- German, Singaporean, Austrian, and Polish have *<100 stores*

Average Delivery Seconds

- Range: Delivery times generally fall between *33 to 50 minutes*
- Higher Times: Southern, Mediterranean, and Thai
- Lower Times: Alcohol-plus-food, Brazilian, and Indian

- Alcohol-plus-food, Ethiopian, and Moroccan have the *highest average item prices*, all over 1400, with Alcohol-plus-food exceeding 1800
- Korean, French, Lebanese, and Japanese fall within the 800 to 1200 price range, indicating a *moderate pricing strategy*
- Bubble Tea, Fast Food, and Convenience Store have the *lowest average item prices*, all under 400, focusing on affordability

Pipeline 1: Training Pipeline



- training_data **version 0** is the input for “Data_Procproessing” and it writes back the processed data to training_data as a new version
- Saved **standard scaler** and **one hot encoder** fitted during training as **pickle files** to use them during inference. This is identified by the ‘**training_flag**’ parameter

AutoML & Metric Used

DOORDASH

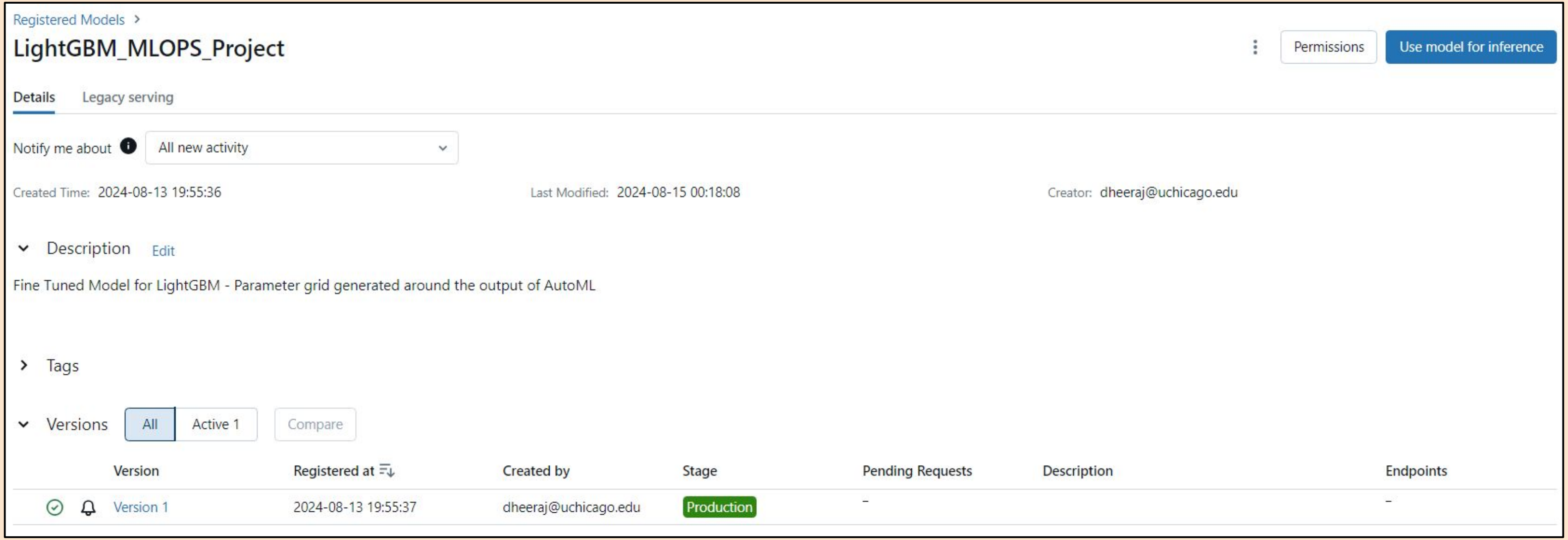
Training dataset: mlops_project.modeling_features
Target column: delivery_seconds
Evaluation metric: val_root_mean_squared_error
Timeout: 60 minutes

Run Name	Created	Dataset	Duration	Source	Models	test_mean_absol	test_root_mean
delicate-smelt-365	1 day ago	dataset (b66600c2) Test_ , data... +2	2.6min	Notebo...	sklearn	554.2652747...	706.4405467...
grandiose-jay-345	1 day ago	dataset (b66600c2) Test_ , data... +2	2.1min	-	sklearn	555.3832982...	707.5719259...
selective-seal-776	1 day ago	dataset (b66600c2) Test_ , data... +2	2.4min	-	sklearn	556.0013963...	708.4737390...
bouncy-skink-529	1 day ago	dataset (b66600c2) Test_ , data... +2	2.3min	-	sklearn	556.1831547...	708.5582089...

- Chosen **RMSE** as the evaluation metric, as it quantifies the average error in delivery time predictions. This directly reflects how closely the model's ETA aligns with actual delivery times, providing a clear measure of potential delays if implemented in production
- AutoML implemented approximately 250 different models from LightGBM, XGBoost, RandomForest, SGD and Decision Trees. However, **LightGBM** outperformed other models, with a test RMSE of **12 minutes** (approx)

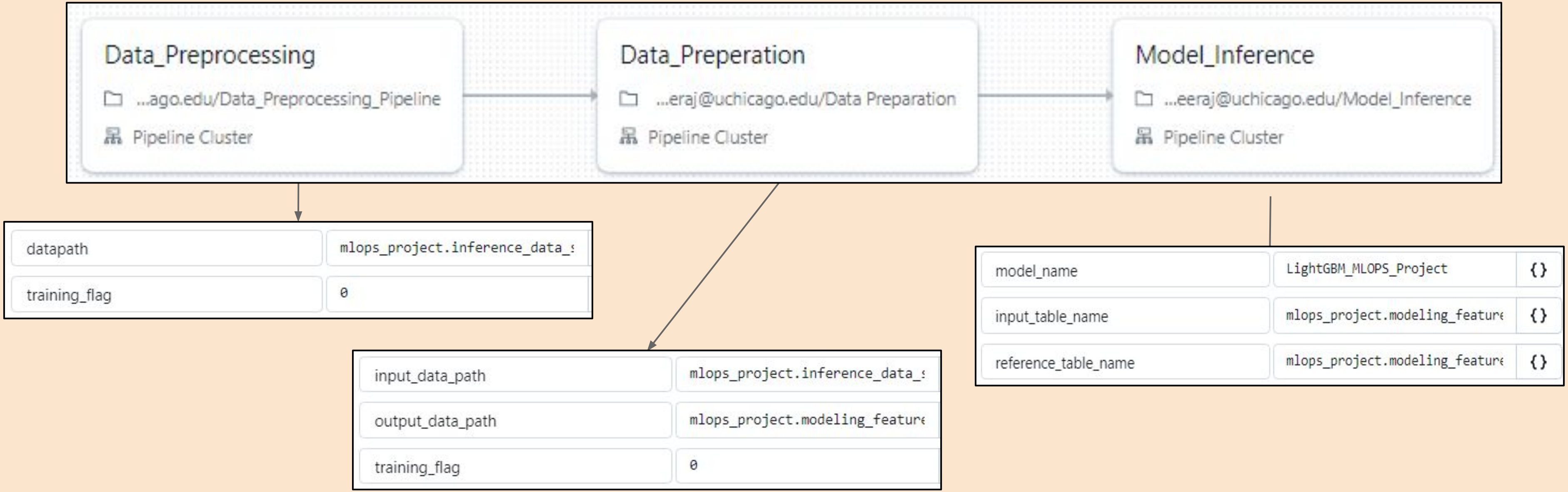
```
LGBMRegressor(colsample_bytree=0.409575686295452,
lambda_l1=141.10516620147297,
              lambda_l2=23.859424872327192,
learning_rate=0.026481749683722605,
              max_bin=267, max_depth=12, min_child_samples=55,
              n_estimators=1504, num_leaves=187,
random_state=211830047,
              subsample=0.5008734724835469)
```

Model Fine Tuning & Deployment



- Created a hyperparameter grid around the hyperparameters given by AutoML for LightGBM
- Performed Random search reducing RMSE from 12 minutes to **11 minutes**
- Logged this model on MLFlow and registered it as part of deployment

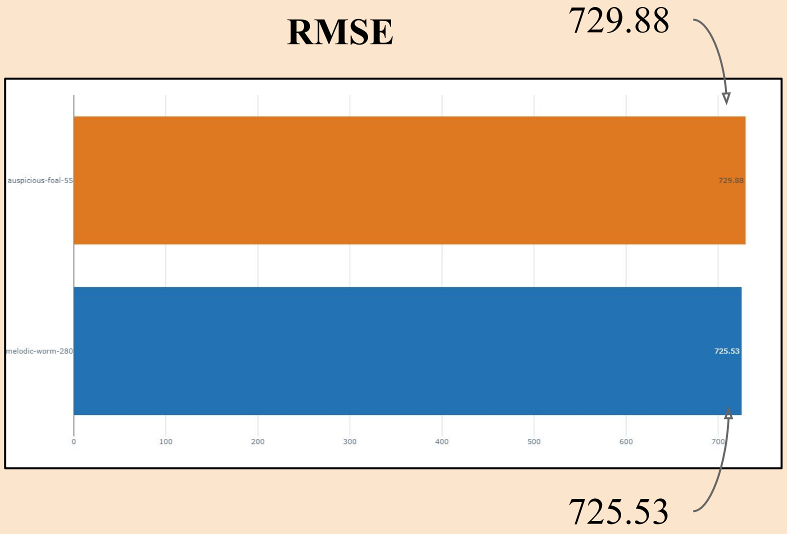
Pipeline 2: Inference Pipeline

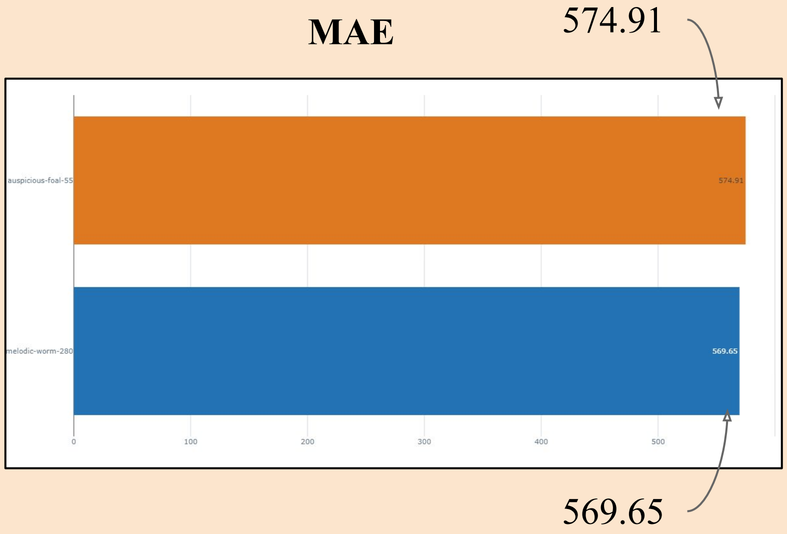


- Loading the pickle files for standard scaler and one hot encoder fitted during training to transform data during inference
- Loads the deployed model and perform batch inference, logs the performance metrics on MLFlow and Data Drift on Evidently

Demo

Model Performance Monitoring





We observe an ***increase*** *in RMSE/ MAE by* ***4 seconds*** *during inference with the same model and data, but with columns swapped*

Inference Set 1

Inference Set 2 (swapped)

Data Drift: Inference Set 1

DOORDASH

Dataset Drift

Dataset Drift is detected. Dataset drift detection threshold is 0.011363636363636364

88	1	0.0114
Columns	Drifted Columns	Share of Drifted Columns

Drift in column 'prediction'

Data drift detected. Drift detection method: Wasserstein distance (normed). Drift score: 0.374

- Covariate shift **identified** in ‘max_item_price’
- Prior probability drift **identified** with a drift score of **0.374**

Data Drift: Inference Set 2

DOORDASH

Dataset Drift

Dataset Drift is detected. Dataset drift detection threshold is 0.011363636363636364

88	3	0.0341
Columns	Drifted Columns	Share of Drifted Columns

Drift in column 'prediction'

Data drift detected. Drift detection method: Wasserstein distance (normed). Drift score: 0.399

- Covariate shift **identified** in 3 columns: columns that were swapped (‘total_items’, ‘subtotal’) and ‘max_item_price’
- Prior probability drift **identified** with a drift score of **0.399** (greater than the score observed with inference set 1)

Conclusion

1

Project Achievements:

- Successfully deployed a machine learning pipeline to predict DoorDash delivery times accurately
- Improved ETA accuracy, enhancing customer satisfaction and operational efficiency
- Implemented robust data preprocessing, reducing the dataset by 34% while maintaining quality

2

Model Performance:

- Utilized AutoML to test ~250 models, with LightGBM emerging as the best performer
- Fine-tuned the model, reducing RMSE from 12 minutes to 11 minutes
- Achieved RMSE of 569.65 seconds (~9.5 minutes) on the first inference set

3

Data Insights:

- Identified key factors influencing delivery times: driving duration and outstanding orders
- Discovered patterns in delivery times across weekdays and hours, informing resource allocation
- Analyzed store categories and service attributes, revealing trends in delivery times and pricing

4

Monitoring and Drift Detection:

- Implemented continuous monitoring of model performance and data drift
- Detected covariate shift in 'max_item_price' and prior probability drift in both inference sets
- Observed slight performance degradation when feature values were swapped

5

Infrastructure and Pipeline:

- Successfully set up Databricks environment with Unity Catalog and Delta Tables
- Developed separate training and inference pipelines for maintainability and scalability

6

Future Directions:

- Further optimize the model based on drift insights
- Expand the feature set to capture more nuanced factors affecting delivery times
- Implement real-time monitoring and retraining strategies to maintain model accuracy

Contributions

- **Achintya**: Auto ML, Inference Pipeline
- **Dheeraj**: Training Pipeline, Model Deployment and Monitoring
- **Yujata**: EDA, Inference Pipeline