DS 5460 Big Data Scaling

Enhancing Bus Punctuality:

A Machine Learning Approach to Binary Classification of Bus Arrival Times

April 14, 2025

Silin Chen, Stella Wu, Xinyi Zhang

Project Description

- **Objective**

Develop a machine learning classifier to predict whether a bus will arrive **on time** or **late**, based on real-time GPS and schedule data.

- **Why This Matters**

NYC buses are frequently delayed due to traffic, weather, and urban complexity.

Unpredictable service impacts passenger satisfaction and operational efficiency.

Accurate prediction can help transit authorities make proactive decisions and improve public trust.

- **Our Solution**

We use a binary classification approach powered by PySpark to process large-scale transit data and train multiple ML models.

The prediction target:

Will the bus be late? → Yes / No

Dataset

- **Source**

New York City Metropolitan Transportation Authority (MTA)

Downloaded from Kaggle: New York City Bus Data (5.54 GB)

- **Data Description**

Size: Over 5 million records, 5.54GB in raw CSV

Interval: Real-time tracking every 10 minutes

Period: October 2017 snapshot of NYC buses

- **Key Features**

OriginLat, OriginLong / DestinationLat, DestinationLong

VehicleLocation_Latitude, VehicleLocation_Longitude

ScheduledArrivalTime, RecordedTime

DistanceFromStop, DirectionRef, PublishedLineName

- **Label Engineering**

We created a binary label LateIndex by comparing scheduled and actual arrival times:

1 = late, 0 = on time

First model: Decision Tree (Baseline Model)

- **Decision Tree Classifier**

Training & Test Split: 70% / 30%

- **Features Included**

DirectionRef, Origin/Destination Latitude & Longitude, Vehicle Location (Lat/Long), Distance from Stop,
Late (binary label: on-time or late)

- **Initial Observation**

Achieved 100% accuracy on both training and test sets, which suggests potential overfitting and a need for further feature engineering and model tuning.

```
Model Evaluation Results
--------------------------------------------------------------------------------
A single decision tree accuracy on training data: 100.00%
A single decision tree accuracy on test data: 100.00%
--------------------------------------------------------------------------------
```

First model - Feature Engineering

1) **Add new feature:** Discrepancy

   A column calculating discrepancies between **scheduled** and **expected arrival times**.

2) **Update feature: Feature Interaction and Polynomial Features**

   Columns showing interactions between 'OriginLat'/'OriginLong' and 'DestinationLat'/'DestinationLong'

3) **Update feature: Binning and Categorization of Continuous Variables**

   A column **Categorizing 'DistanceFromStop'** into 'very_close', 'close', 'moderate', 'far'

Feature Engineering Method	Description	Train Accuracy	Test Accuracy
Add new feature: Adherence	Calculated discrepancy between scheduled and expected arrival times to indicate delays	100.00%	100.00%
Update feature: Feature Interaction & Polynomial	Captured non-linear relationships by combining multiple features	100.00%	100.00%
Update feature: Binning & Categorization	Converted continuous variables into categorical bins for more interpretable patterns	62.77%	61.86%

First model - Hyperparameter Tuning

- **Tuning Method**

Tool: TrainValidationSplit (PySpark)

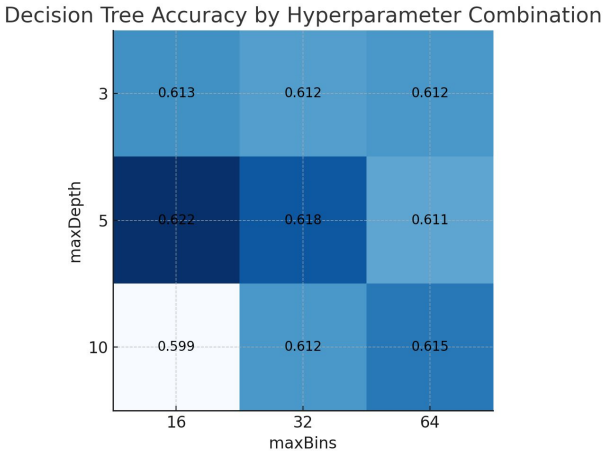
Grid Search over: maxDepth: [3, 5, 10], maxBins: [16, 32, 64]

Total combinations: 9

- **Two-Stage Tuning Strategy**

10% Sample → Pipeline validation

Full Dataset → Final selection



- **Best Configuration**

maxDepth = 5, maxBins = 16

Test Accuracy: 62.35%

- **Observation**

Mid-sized trees generalize best

Tree models are sensitive to depth/bucket size

Spark tuning scales efficiently on GCP

Second model: Random Forest

- **Why Random Forest?**

Builds on top of Decision Trees

Combines multiple trees to reduce overfitting

More robust and stable for real-world data

- **Performance**

Metric	Result
Training Accuracy	100.00%
Test Accuracy	100.00%

- **Model Setup**

Features: Same as first model

Label: LateIndex (binary)

Split: 70% training / 30% testing

Parameters:

numTrees = 20

maxDepth = 5

handleInvalid = "skip"

- **Observation**

Model correctly classified every example

Suggests either:

Extremely strong predictive features

Needs further validation with cross-validation or external data

Third model: Logistic Regression

- **Objective**

Test if Logistic Regression provides significantly better performance than our baseline Decision Tree model.

- **Setup**

Used the same feature set and data pipeline

Applied LogisticRegression model from PySpark ML

Compared on accuracy and generalization

- **Result**

Performance was comparable to Decision Tree，but no immediate improvement without further feature engineering or hyperparameter tuning

```
Model Evaluation Results
--------------------------------------------------------------------------------
Logistic regression accuracy on training data: 100.00%
Logistic regression accuracy on test data: 100.00%
--------------------------------------------------------------------------------
```

Forth model: Gradient Boosted Decision Trees

GBDT is an ensemble learning technique that builds additive models in a forward stage-wise fashion, using decision trees as weak learners. We compared two approaches:

- Baseline GBDT with a fixed number of trees (no early stopping)

```
📌 Baseline GBDT (maxIter=50)
   ➤ Train Accuracy:      0.7321
   ➤ Validation Accuracy: 0.7012
```

- GBDT with Manual Early Stopping, where the optimal number of trees was selected based on validation performance.

```
🔁 maxIter=10 | Train Acc=0.7107 | Val Acc=0.6968
🔁 maxIter=20 | Train Acc=0.7173 | Val Acc=0.6981
🔁 maxIter=30 | Train Acc=0.7226 | Val Acc=0.6984
🔁 maxIter=40 | Train Acc=0.7278 | Val Acc=0.6981
🔁 maxIter=50 | Train Acc=0.7321 | Val Acc=0.7012
🔁 maxIter=60 | Train Acc=0.7350 | Val Acc=0.7012

✅ Best GBDT with Early Stopping:
   ➤ Optimal maxIter:      50
   ➤ Train Accuracy:       0.7321
   ➤ Validation Accuracy:  0.7012
```

Model Comparison

Model	Training Accuracy	Test Accuracy	Notes
Decision Tree (baseline)	100.00%	100.00%	Possible data leakage; perfect results are suspicious
Decision Tree (binned)	62.77%	61.86%	More realistic; shows effect of interpretable feature engineering
Random Forest	100.00%	100.00%	Ensemble approach; likely benefiting from same leakage issue
Logistic Regression	100.00%	100.00%	High separability or redundancy in data
GBDT (no early stopping)	73.21%	70.12%	More balanced and interpretable performance
GBDT (early stopping)	73.21%	70.12%	Validates same configuration; avoids overfitting with fewer trees

- **Decision Tree**, **Random Forest**, **Logistic Regression** models initially achieved 100% accuracy, indicating **overfitting**.
- After applying interpretable feature engineering and hyperparameter tuning to **Decision Tree (tuned)**, performance dropped to **realistic levels**, highlighting the need for robust validation.
- The **Gradient Boosted Decision Tree** model, with and without early stopping, although not fine-tuned, performed **consistently well**.

Thank you!

April 14, 2025

Silin Chen, Stella Wu, Xinyi Zhang