Predictive Analytics for Metro Bus Arrival Times

Vicky Tseng, Aman Ankit, Joey Moran, Brad Luong, Gol-Dann Slater, Muse Wu, and Kai Tan

With special thanks to advisors Dr. Yi Mi, Al-Ahad Ekram, Elizabeth Love, and Patricia Buchanan

Problem
King County Metro (KCM) bus arrival time estimations are often inaccurate. Even though apps like OneBusAway and Google Maps are widely used, they still rely on sourcing their data from the King County Metro data centers.

Current System
The current system uses a linear model which calculates the arrival time by using the distance to the bus stop and speed of the bus. The bus relays its location and speed to the transit data center every 90 seconds, which sometimes can be unreliable.

Time to arrive = Distance remaining / Current speed

122+ million Annual ridership on Metro Transit
23% Of buses are late
4.7 million New Seattle transit riders in 2017
2/3 Of non-riders are more likely to ride with real-time updates*

Data Collection and Cleaning

What data was collected?
- Bus telemetry data (from KCM)
  - scheduled vs. actual times for arrivals/departures
  - passenger boarding/offboarding count
  - nominal and actual miles traveled (from SDOT)
- Traffic data
  - volume, averages and peak traffic in the AM and PM for each street
- Street attribute data (from SDOT)
  - road slope, width, length, speed limit, material
  - highway or not highway
  - one-way or not one-way

How did we clean the data?
- to automatically source SDOT traffic and attribute via API
- to combine traffic, street, and attribute data
- for data cleaning and transformation
- with scalability in mind

Feature Selection

Filter Based Feature Selection (FBFS)

What is FBFS?
Identifies features (variables) in the dataset that have the greatest ability to predict the target column: elapsed travel time between two stops.*

Why use FBFS?
Using identified features allows the model to:
- have higher performance and accuracy
- limit the required computational resources

FBF Results
Using the Mutual Information scoring criteria we filtered our dataset from 67 features to 15

<table>
<thead>
<tr>
<th>Feature</th>
<th>Percentage of Contribution Towards Arrival Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_ELAPSED_ARR</td>
<td>35.4%</td>
</tr>
<tr>
<td>TIME_WKND</td>
<td>14.0%</td>
</tr>
<tr>
<td>TIME_DAYHOUR</td>
<td>12.3%</td>
</tr>
<tr>
<td>VEHICLES_6X8</td>
<td>9.6%</td>
</tr>
<tr>
<td>VEHICLES_4X6</td>
<td>6.9%</td>
</tr>
<tr>
<td>VEHICLES_8X10</td>
<td>6.0%</td>
</tr>
<tr>
<td>VEHICLES_12X14</td>
<td>5.1%</td>
</tr>
<tr>
<td>VEHICLES_14X16</td>
<td>3.6%</td>
</tr>
<tr>
<td>VEHICLES_17X19</td>
<td>2.9%</td>
</tr>
<tr>
<td>VEHICLES_20X22</td>
<td>2.7%</td>
</tr>
<tr>
<td>VEHICLES_24X26</td>
<td>1.7%</td>
</tr>
<tr>
<td>VEHICLES_30X32</td>
<td>0.3%</td>
</tr>
</tbody>
</table>

Clustering

Methodologies
1. Dynamic Time Warping (DTW)
2. Permutation Distribution Clustering (PDC)
3. Boosted Decision Tree (BDT)

What is DTW and PDC?
- DTW: finds similarity between two time series
- PDC: finds the permutation distributions for time series graphs
- BDT: performs agglomerative hierarchical clustering

Why cluster?
- Along any route there are many different regions, from downtown streets to the highways to the neighborhoods
- The city did not want to use predetermined segments but instead wanted the clustering algorithm to self-generate segments using the highest scoring variable from FBFS

Regression Results Incorporating 3 Models

Our results indicate that:
- More than 95% of bus travel time within one minute and at least 30s earlier than actual for Neural Network.
- More than 99% for both Random Forest and Boosted Decision Tree.

Objective
Given the constraints and problem, our objective is to improve the data quality at King County Metro data center by building a scalable machine learning model that can accurately and dynamically predict when the bus will arrive to any bus stop.

Constraints
While we can neither enforce a change in driver behavior nor change real-world events that influence the arrival time, we can improve the experience by increasing the transparency of bus arrivals and communicating the updates to the rider in real-time.

Impact
74% More accurate than current system
22.9 million+ More riders given on-time estimates
6722 Public Transit Systems in the US can incorporate our work in their data centers*

Moving Forward
From our final report, code documentation, and models, our project will be applied to other routes within KCM service.


Appendix

* Methods:
The neural network model is trained using a supervised learning approach with backpropagation algorithm and RMSProp optimizer. The Random Forest regression model is trained using a decision tree algorithm with bagging and random feature selection for each split. The Boosted Decision Tree algorithm is trained using a linear model with AdaBoost as the base learner. The ensemble models are trained using the scikit-learn library in Python.

**Clustering:**
The clustering algorithms are trained using the scikit-learn library in Python. The DTW and PDC algorithms are trained using a decision tree algorithm with bagging and random feature selection for each split. The Boosted Decision Tree algorithm is trained using a linear model with AdaBoost as the base learner. The ensemble models are trained using the scikit-learn library in Python.

**Model Evaluation:**
We evaluate the performance of the models using the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE). The models are trained using a cross-validation technique with k-fold validation. The hyperparameters are tuned using a grid search approach. The models are trained on a local machine with 32GB of RAM and an NVIDIA 3090Ti GPU.

**Data:**
The dataset consists of over 100 million bus arrival time records and includes information such as bus ID, timestamp, bus stop ID, and bus speed. The data is sourced from the King County Metro transit system and includes data from April 2019 to March 2020.

**Deployment:**
The models are deployed on a local machine with 32GB of RAM and an NVIDIA 3090Ti GPU. The models are deployed using the scikit-learn library in Python. The models are trained on a local machine with 32GB of RAM and an NVIDIA 3090Ti GPU. The models are deployed using the scikit-learn library in Python.