### Motivation and Goals
- New York City’s Metropolitan Transportation Authority (MTA) manages one of the busiest bus systems in the world, transporting over 2 million people daily.
- Since 2011, the MTA has released an API that gives real-time bus locations and predicted arrival times for every bus in service in 30 second increments.
- The goal of this project is to develop ML algorithms that provide better bus predictions than the MTA.
- Better predictions lead to better passenger experience and help the MTA more accurately manage passenger demand.

### Dataset and Features
- We scraped the MTA Bus API for 4 days (October 16-19, 2017) to create our dataset. Each prediction instance represented a bus’s distance to a future stop, its estimated time of arrival (ETA) and bus line. We denoted a bus as having arrived at a stop when the API showed that the bus is within 50 meters from the designated stop.
- We added more features to the dataset by adding weather and population density data by hour and geographic location using the Dark Sky API and the Census Bureau.
- Through exploratory data analysis, we reduced the dataset to 16 features due to computational constraints (e.g. too many bus lines and hours in a day to use as features).
- Overall, we had 4.7, 2.1 and 2.9 million bus prediction instances in our training, validation and test sets.

### Machine Learning Models Used

### Linear Regression (with Distance to Stop as only feature)
This served as our baseline model from which to compare.

### Linear Regression w/ All Features and Forward Selection
We used linear regression again, but this time we initiated a forward selection algorithm that selects the next best feature with the best F statistic. We ran this algorithm limiting the number of feature selected (1 to 16), and selected the one with the lowest cross-validation error.

### Linear Regression with Regularization
To minimize overfitting, we experimented with L-1 (Lasso) and L-2 (Ridge) regressions. The minimizing cost function is:

**Lasso:** \[ J(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \sum_{j=1}^{n} |\theta_j| \]

**Ridge:** \[ J(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \sum_{j=1}^{n} \theta_j^2 \]

### Boosting (Adaboost.R2 Algorithm)
We also used ensemble methods, which combines weak learners to create a strong learner. Here, we use Adaboost.R2, a boosting algorithm that runs a regression tree multiple times, each time weighing examples with larger errors more heavily.

### Random Forests
We ran random forests on the dataset; training regression trees multiple time by subsampling the training set and its features.

### Results
- We achieved 18% mean average prediction error (MAPE) and 246 seconds in root mean squared error (RMSE).
- This is slightly higher than the 14.5% MAPE and 203 seconds in RMSE that the MTA predictions had achieved.

### Discussion and Future Work
- Random Forests performed the best among the algorithms tested. Ensemble methods generally performed better than linear regressions and neural networks.
- Future work will include running algorithms with more features and with computational resources. This will enable me to add features such as the route the bus is running on (there are >150 bus routes in the dataset).
- Future work also include experimenting with SVMs.
- Many examples were illogical and were discarded, so capturing data more accurately is a priority.