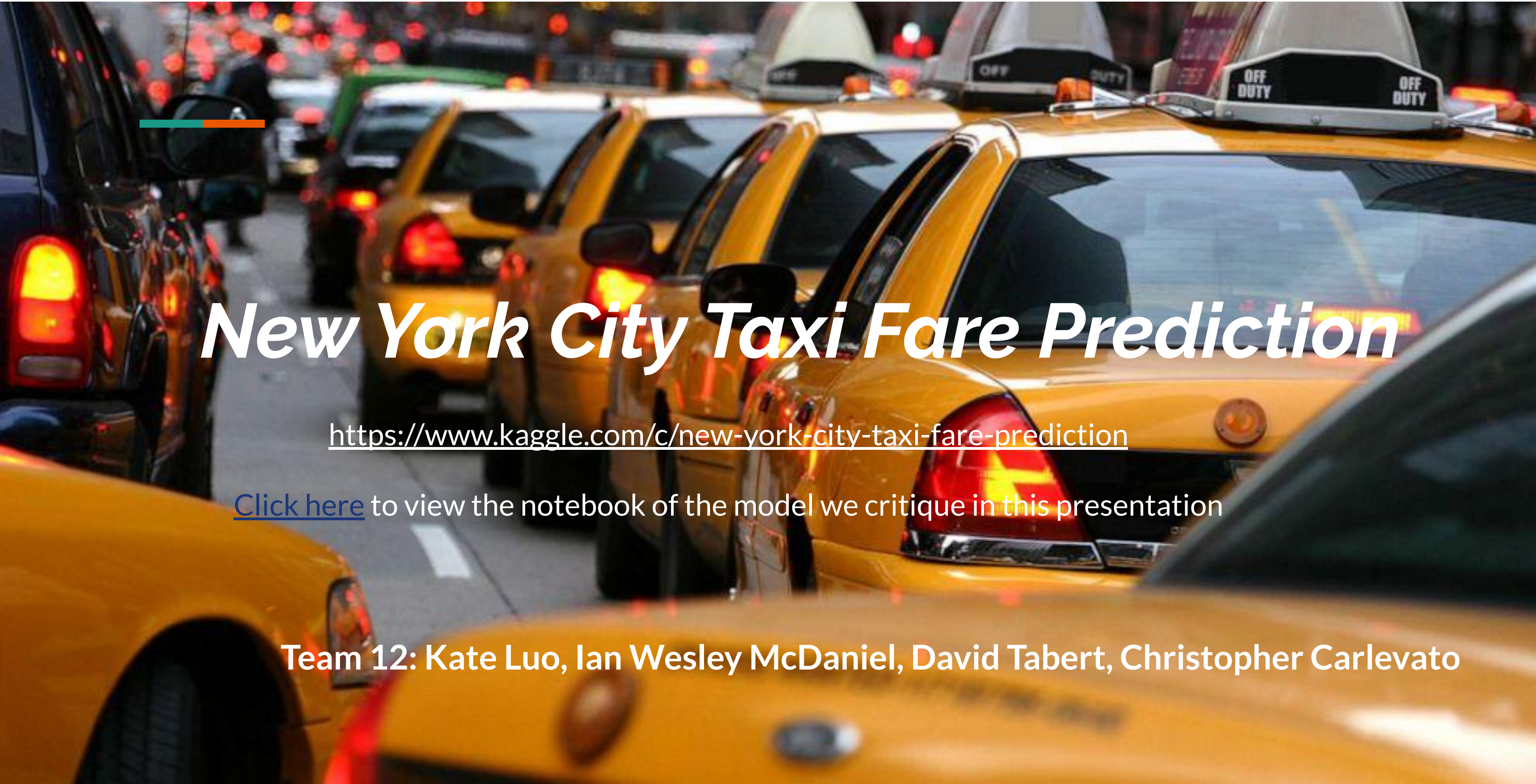

New York City Taxi Fare Prediction

https://www.kaggle.com/c/new-york-city-taxi-fare-prediction

Click here to view the notebook of the model we critique in this presentation

Team 12: Kate Luo, Ian Wesley McDaniel, David Tabert, Christopher Carlevato

Overview

Objective

- Predict fare amount (inclusive of tolls) for a taxi ride in NYC given the pickup and dropoff locations
- Linear model → XGBoost
    - A decision tree algorithm, rather than a simple linear model
    - Beat linear model RMSE: $5.74
        - Model was, on average, off by $5.74
        - Average fare price: $11.33
- Application: Provide riders an accurate cost estimate for their ride
    - Key user feature of ride hailing apps

Objective

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

- Square root of the squared difference between prediction and actual values (residuals) averaged over all of the sample
- Relative to MAE, RMSE penalizes large errors
    - Errors are squared before they are averaged
- Advantage
    - Model prediction error are in the units of the response variable

Dataset

- Separate CSV files for train & test data
  - Big data
    - 55M rows
- Read in 3M rows

```
NY_Train <- fread("train.csv", nrows = 3000000)
```

Features

- pickup_datetime - `timestamp` value indicating when the taxi ride started.
- pickup_longitude - `float` for longitude coordinate of where the taxi ride started.
- pickup_latitude - `float` for latitude coordinate of where the taxi ride started.
- dropoff_longitude - `float` for longitude coordinate of where the taxi ride ended.
- dropoff_latitude - `float` for latitude coordinate of where the taxi ride ended.
- passenger_count - `integer` indicating the number of passengers in the taxi ride.

Target

- fare_amount - `float` dollar amount of the cost of the taxi ride. This value is only in the training set; this is what you are predicting in the test set and it is required in your submission CSV.

NYC Taxi Stats

- NYC taxi trend dashboard
- Average 180 miles per shift
- Decline since launch of ride hailing apps in 2011

Original model

- Strengths:
  - Trained on ~10,000,000 samples
  - Simple
- Weakness:
  - Truth is very rarely linear
    - Large RMSE
  - Multicollinearity
  - Little feature engineering
    - Average lat and lon between pickup and dropoff
  - Data quality
    - Little data cleansing happened
      - Only NaNs removed from lat/lon columns
    - Garbage in → Garbage out

Our Model: XGBoost

- Looked at some published notebooks and found that XBGBoost performed well with this data
    - [Mitchell O'brien's notebook]
- XGBoost advantages
    - Tree-based method which does not assume linearity
    - Slow and weak but effective learners
        - Include a validation stopping rule to avoid overfitting
    - More regularized model to control overfitting
    - Feature importance
- Disadvantage:
    - Computationally expensive

XGBoost Algorithm:

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

XGBoost Model: Added features

- Distance
    - Haversine distance
- Time Features
    - Year
    - Month
    - Day of the week
    - Etc.
- Added distance from pickup location to popular destinations as features
    - Airports and major city landmarks
        - e.g. distance to JFK

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
team12_sub.csv	just now	0 seconds	0 seconds	3.04896

Complete

XGBoost Results

- Learning Rate 0.05 → 0.01
- 8-leaf trees → 6-leaf trees
- 6,000 trees → 500 trees
- Took out certain unimportant features and reran model
- Trained until validation RMSE had not improved in 10 rounds
- RMSE: 3.04896
- Outperformed simple linear model RMSE by 2.69
- Unsurprisingly, distance is the most important feature

Future improvement

- Use entire training dataset
  - Used 5% of all training data
- More feature engineering
- Hyperparameter tuning
- Rerun model with principal components
- Try other models like Random Forest to see if they outperform XGBoost

Sources

- https://www.kaggle.com/obrienmitch94/nyc-taxi-fare-prediction
  - Used this code as a prototype for our model, but improved model by changing hyperparameters and eliminating unnecessary features