New York City Taxi Fare Prediction Android APP

Zhibin Huang
Shilu Wu

Overview

- Project Introduction
- Dataset Processing
- Model Building
- App Implementation

Brief Introduction

In our project, we are tasked with predicting the **fare amount** (inclusive of tolls) for a taxi ride in New York City given the pickup and drop-off locations.

While we can get a basic estimate based on just the distance between the two points, this will result in **an RMSE of $5-$8**, depending on the model used. Our challenge is to **do better** than this using Machine Learning techniques and build and **Android App**.

Dataset Introduction:

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

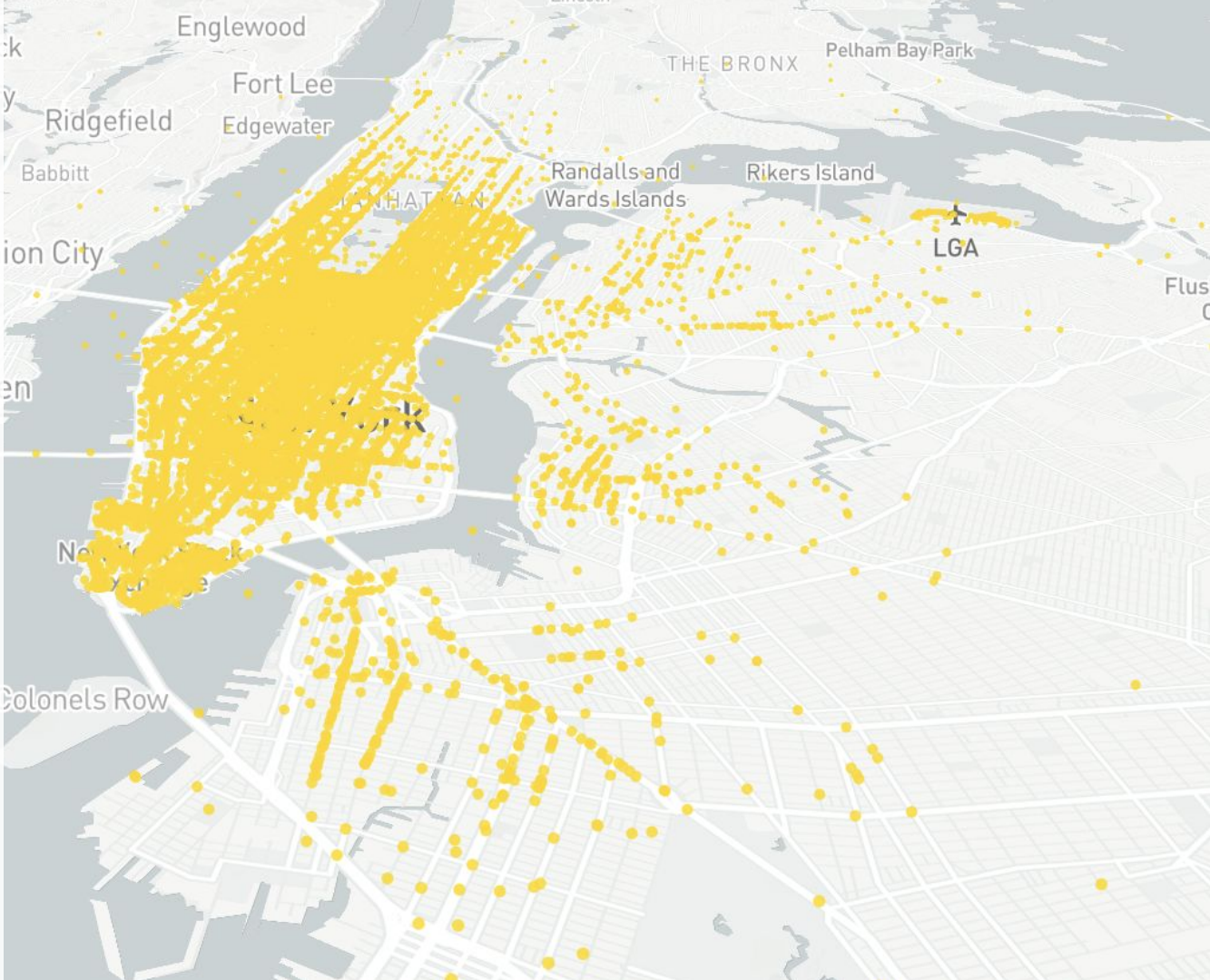
- fare_amount - `float` dollar amount of the cost of the taxi ride. This value is only in the training set; this is what you are predicting in the test set and it is required in your submission CSV.

Source:

- https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/data
- data size is 55M rows but we only use 10M rows since we can’t run all data in our own computer

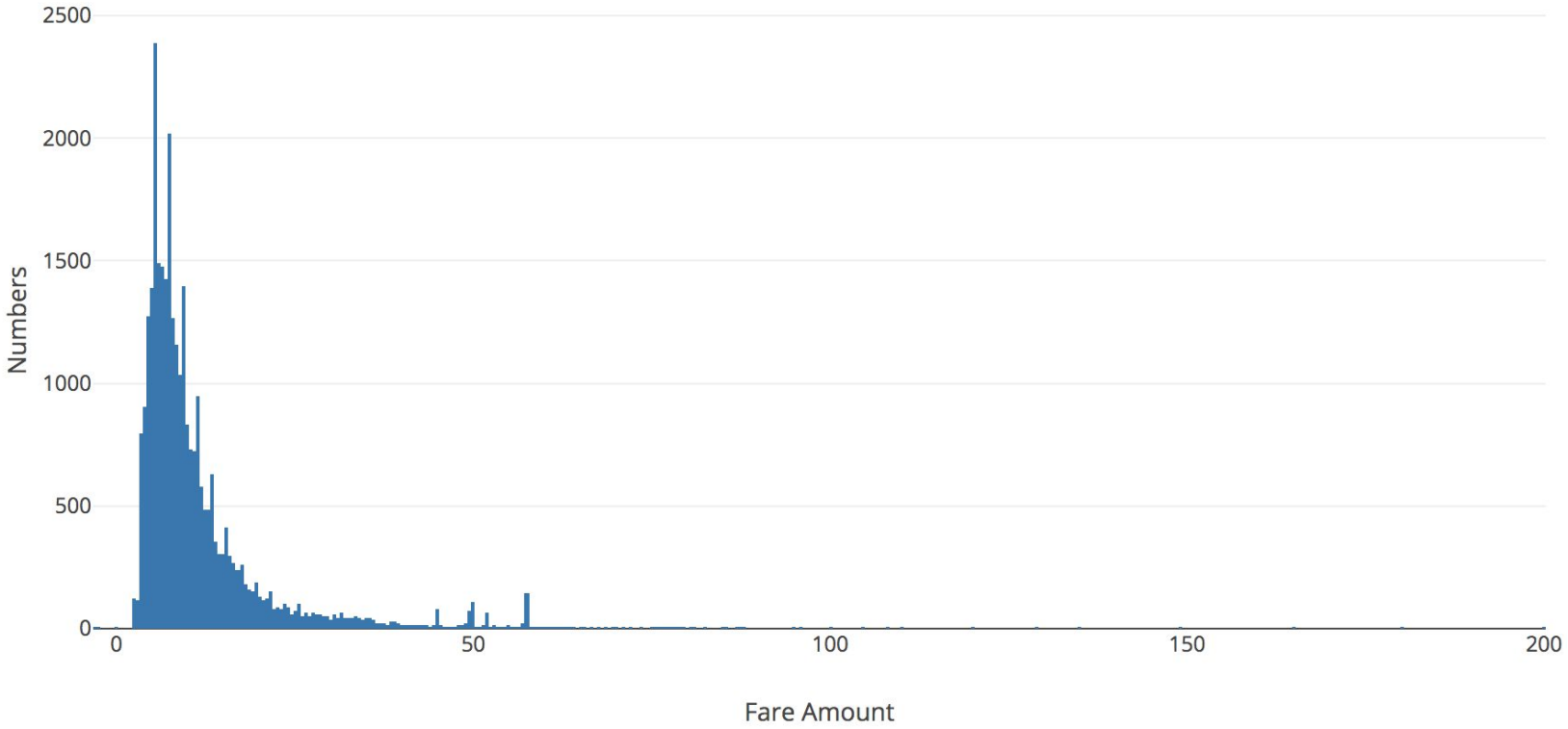
Dataset Introduction:

Pick up locations in NYC



Dataset Introduction:

Fare amount Distribution



Data Cleaning

- Some rows are NaN.
- Some fare amount is negative.
- Some coordinates are far away from New York City.
- Some coordinates are located in water.
- Some passenger counts are greater than 6.

Data Cleaning

- Old size: 10,000,000
- New size(without NaN): 9,999,931
- Old size: 9,999,931
- New size(diff longitude & latitude no to large): 9,979,187
- Old size: 9,979,187
- New size(fare_amount > 0): 9,978,783
- Old size: 9,978,783
- New size(in NYC): 9,787,199
- Number of trips in water: 2025
- Old size: 9,787,199
- New size(Not in Water): 9,785,174
- Old size: 9,785,174
- New size(reasonable passenger count): **9,750,645**

Input Features

- fare_amount float64
- passenger_count int64
- abs_diff_longitude float64
- abs_diff_latitude float64
- euclidean float64
- manhattan float64
- year int64
- month int64
- day int64
- hour int64
- weekday int64

Model Selections

- **Linear Regression**: Based on the absolute longitude and latitude value to predict the fare amount by using linear regression.

- **XGBoost**: It's one of the gradient boost method called eXtreme Gradient Boost used for supervised Machine Learning. The weak learner used in XGBoost is generally the decision tree.

Training Model - Linear Regression

- In our model, we firstly take the absolute longitude/latitude as our input features. By calling **numpy.linalg.lstsq** function to return the least-squares solution to a linear matrix equation. Afterward we take all features as inputs.
- Model parameters: None

Training Model - XGBoost

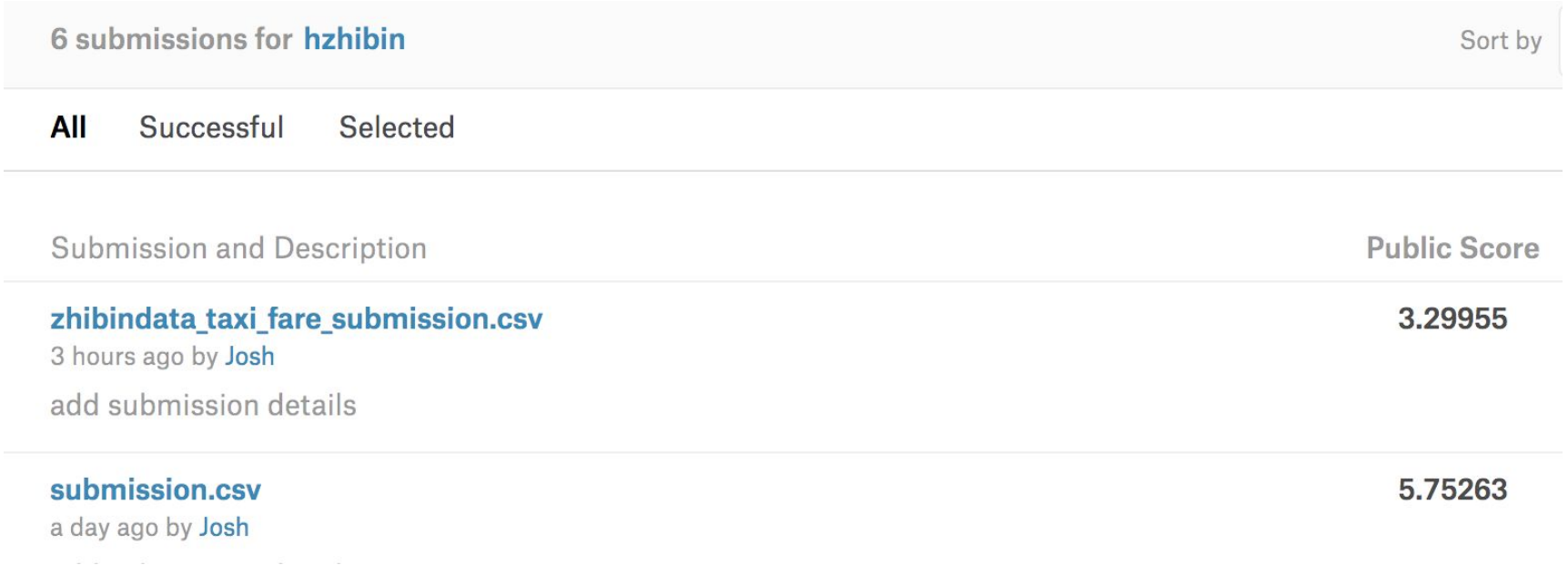
- In our model, we firstly take the absolute longitude/latitude as our input features. By separating the data into 99% training and 1% validation to get the validation loss. Afterward we take all features as inputs.
- Model parameters: learning_rate=0.3,max_depth=6, n_jobs=-1, silent=False

```
%%time
y_XGB_predict = XGB_model.predict(x_test)

XGB_model_error = np.sqrt(mean_squared_error(y_test, y_XGB_predict))

print('XGBoost Root Mean Squared Error - {XGB_model_error}')
print(XGB_model_error)
```

```
XGBoost Root Mean Squared Error - {XGB_model_error}
3.660760959812958
```

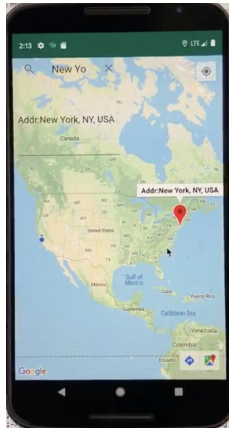
Model Results

Linear and XGBoost Results: (The Best Score on Kaggle is around 3)



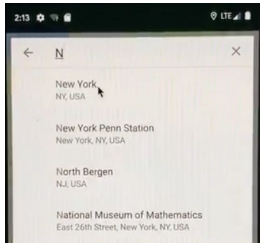
App Implementation - Google Maps SDK for Android

- Add a Map to our application using the Maps SDK for Android.
  The map includes a marker (pin), to indicate a specific location.



- Select current location and display details of the place at that location.

When the user clicks on search box, it will offer the user a list of likely places to choose from.



App Implementation - Google Maps SDK for Android

UI Controls

- Using the Maps SDK for Android to customize the way in which users can interact with the map.The Maps API offers built-in UI controls.
- *My Location button*: this button appears in the top right corner of the screen.
- *Map toolbar*: by default, a toolbar appears at the bottom right of the map when a user taps a marker. The toolbar gives the user quick access to the Google Maps mobile app.
- *Zoom controls*: the built-in zoom controls appears in the bottom right hand corner of the map.

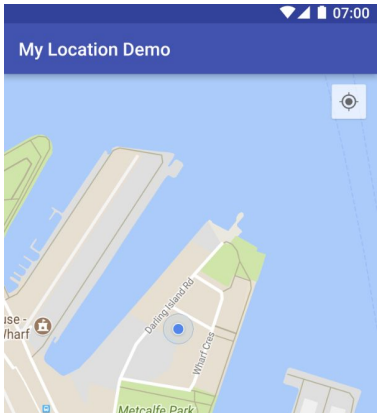
App Implementation - Google Maps SDK for Android

Map Gestures

- Zoom gestures: The map responds to a variety of gestures that can change the zoom level. (Double tap to zoom in/Two finger tap zoom out...)
- Scroll gestures
- Tilt gestures
- Rotate gesture

App Implementation - Google Maps SDK for Android

Location Data



- One of the unique features of mobile applications is location awareness. Adding location awareness to the app offers users a more contextual experience because mobile users bring their device with them everywhere.
- *Location permissions*: Allows the API to determine as precise a location as possible from the available location providers, including the Global Positioning System (GPS) as well as WiFi and mobile cell data.
- *The My Location layer:* Shows the user's current position on the map. Once it's enabled, the My Location button appears in the top right corner of the map. When a user clicks the button, the camera centers the map on the current location of the device. The location is indicated on the map by a small blue dot if the device is stationary, or as a chevron if the device is moving.

App Implementation – Google Places SDK for Android

- The Places SDK for Android allows us to build location-aware apps that respond contextually to the local businesses and other places near the device.
- Concepts
  - *Places*: provides programmatic access to Google's database of local place and business information, as well as the device's current place. A place is defined as a physical space that has a name, or anything you can find on a map. Examples include local businesses, points of interest, and geographic locations.
  - *Autocomplete*: provides pre-made widgets to return place predictions in response to user search queries.
- API overview:
  - Current Place returns a list of places where the user's device is last known to be located along with an indication of the relative likelihood for each place.
  - Place Autocomplete automatically fills in the name and/or address of a place as users type.
  - Place Photos returns high-quality images of a place.
  - Place Details return and display more detailed information about a place.
  - Place IDs stores the unique ID for one or more places for retrieval of place information on demand.

Thank you!