Machine Learning Engineer Nanodegree Capstone Proposal

Akshay Uppal

January 15, 2018

Proposal

Domain Background

With companies like Uber, Lyft and various other cab and cab sharing services, there has been a huge increase in the number of cab rides over the past few years. Cabs have always been a part of our lives in every major city where owning a car is more of a nuisance. As a part of my first project I wanted to touch upon something that impacts everyone.

All the mentioned taxi services must be using some electronic system for dispatching to cater to the growing demand of cabs. This brings to light the requirement of predicting accurately how long each ride takes, making it possible for the system to assign the pickup of next passenger to the right driver. The research paper that intrigued me and made me pursue this is "A Simple Baseline for Travel Time Estimation using Large-Scale Trip Data - Hongjian Wang , Zhenhui Li , Yu-Hsuan Kuo , Dan Kifer" and on some research I found a very relevant competition on Kaggle (A huge online platform for ML and data science related competitions), to predict the duration of a cab ride in New York, one of the busiest cities in the world and decided to use that as my capstone project.

Problem Statement

The dataset being used for the competition 'New York City Taxi Trip Duration' consist of the trip details for about 1.4 million taxi rides between 2009 and 2015 in New York city.

The main aim of the challenge is to predict the duration of a taxi ride, which makes it a supervised regression problem to begin with. I will try to achieve as much as I can on Kaggle leaderboard for this contest based on the Root Mean Squared Log Error, which is defined as the criteria.

Datasets and Inputs

The exact dataset being used cab be found here on Kaggle, the training data contains the following features

- id - a unique identifier for each trip
- vendor_id - a code indicating the provider associated with the trip record
- pickup_datetime - date and time when the meter was engaged
- dropoff_datetime - date and time when the meter was disengaged

- passenger_count - the number of passengers in the vehicle (driver entered value)
- pickup_longitude - the longitude where the meter was engaged
- pickup_latitude - the latitude where the meter was engaged
- dropoff_longitude - the longitude where the meter was disengaged
- dropoff_latitude - the latitude where the meter was disengaged
- store_and_fwd_flag - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
- trip_duration - duration of the trip in seconds

*above points directly from the competition page

Features like trip_duration and dropoff_datetime are excluded since this is what we are trying to predict and are absent in the training database. Since this is a playground competition the data has been sampled and cleaned already.

The training data has about 1.45million samples in it and after removing some outliers (around 5k) I perform a 80-20 split for the training data and the validation data.

"It is good to know that the competition dataset is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on Google Cloud Platform. The data was originally published by the NYC Taxi and Limousine Commission (TLC)."

Solution Statement

My approach to this problem is multifold,

- Firstly, I will do something referred to as Exploratory Data Analysis (EDA) which involves visualization of the data to get a better insight, check for correlation of various features, data cleaning (if required) etc.

- The EDA allows me to do feature engineering, which involves in picking the right features, generating new features from the existing ones, eliminating redundant features based on the correlation and converting the features into the format required by the model.

- Finally, being content with my new feature space, I will use my regressor model. For this problem I have decided to use the XGBoost for regression also sometimes known as the Gradient Boosted Trees. XGBoost being scale invariant and having properties like the random forest can handle nonlinear data as well as reduces the need of preprocessing. Being one of the hottest algorithms for supervised regression on Kaggle made my choice easier.

Benchmark Model

During my final stage, i.e. optimization and fine tuning of hyperparameters I will be using the vanilla implementation of the XGBoost with default parameters as my benchmark, the attempt will be to climb the leaderboard as much as I can.

Evaluation Metrics

The evaluation metrics defined by Kaggle is the Root Mean Squared Log Error which is defined here as :

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

ϵ is the RMSLE value (score)
n is the total number of observations in the (public/private) data set,
p_i is your prediction of trip duration, and
a_i is the actual trip duration for i.
$\log(x)$ is the natural logarithm of x

Being the official metric RMSLE was chosen by me as well, the RMSLE is very similar to the most widely used root mean squared error, only the actual values and predicted values pass through the log function first, the 1 is added to avoid log(0). The log punishes the error for smaller values way more than larger values, example log(100)- log(97) gives a way smaller value when compared to log(10)-log(7). This disparity between larger and smaller values makes more sense when we are trying to predict the trip durations as well, a delta of 3 min is way costlier on a short trip duration than a long one. Therefore, instead of a metric like RMSE where the delta has same priority irrespective of the magnitude of predicted and actual values RMSLE provides us with a relative error.

Project Design

Step 1: Exploration, this is where I make myself comfortable with the data, look up the basic stats like mean, median, info of all the features, their types, find features that are correlated and plot some visuals for a better insight.

Step 2: Clean – Post the exploration, I check the data for missing values and either convert them to NaN or drop them. Post the fixing of missing values its important for us to clean the data of the outliers. The plots during exploration helps a lot on this stage as we can visualize the correlating features. Note: Since we are using XGBoost as our model, being a tree architecture based model, its scale invariant hence deals with a lot of outliers naturally plus here also eliminates PCA for feature transformation as an absolute requirement.

Step 3: Feature Engineering, this is where I drop the redundant data and finally create the useful features, I fetch the pickup_datetime and break it down to its individual components. I try to factor in the holidays

and weekends vs the weekdays, for this I use a database (linked in references) to list out the holidays in 2016 for New York City.

Step 4: Once the features are selected, I train them on the vanilla XBoost to create a benchmark model, pnce the benchmark is created parameters like max_depth, colsample_bylevel, subsample, min_child_weight are modified in an experimental range to achieve better results.

Tools and Libraries Used: Python, Jupyter Notebook, Numpy, Pandas, Seaborn, Maplotlib, Sklearn

Resources

- https://www.kaggle.com/c/nyc-taxi-trip-duration#description (The main competition page on Kaggle, also holds the raw database)
- https://www.kaggle.com/onlyshadow/a-practical-guide-to-ny-taxi-data-0-379/notebook (A very advanced kernel for Exploratory Data analyses, learned a few new techniques from here so decided to mention it).
- A Simple Baseline for Travel Time Estimation using Large-Scale Trip Data - Hongjian Wang , Zhenhui Li , Yu-Hsuan Kuo , Dan Kifer (The paper applies ML to both shanghai cab data and New York cab data and compares them, it also uses APIs by baidu and bing)