Predicting taxi trip duration in New York based on various attributes

Radoš Aćimović, Nevena Lazarević, Filip Jablanović

1. Motivation

New York City is known for its taxi service, so we were interested in finding out details about taxi trips and what is it that impacts the duration of the trip most. So, based on our results we would help people who use taxi service. We live in a rapid world, a world where time is money and we are constantly in a hurry. According to that, the ability to predict how long a taxi trip would last, could present valuable insights to taxi service users in organizing their time and obligations, not thinking about traffic jams anymore.

2. Research questions

In this project, we have applied machine learning concepts that we have covered in lectures of the *Machine Learning* course to create a strategy for predicting duration of taxi trips in New York City.
For our dataset we used the one from NYC Taxi and Limousine Commission from 2013 [1]. The given dataset contains millions of samples that represent real taxi trips for each month of the 2013 year. We reduced this dataset to a more manageable size because of our limited resources and also to minimize the computation time.

Features [2] which are given in the dataset are following:

- medallion: also known as a CPNC is a transferable permit allowing a taxi driver to operate
- hack license: id of the hack license of taxi driver
- vendor id: code that indicates the TPEP provider that provided the record
- rate code: the final rate code in effect at the end of the trip
- store and fwd flag: whether the trip record was held in vehicle memory before sending to the vendor (due to connection failure to the server)
- pickup datetime: exact time when passengers entered taxi vehicle
- dropoff datetime: exact time when passengers left taxi vehicle
- passenger count: number of passengers
- trip distance: distance of the trip in miles
- pickup longitude: longitude of the start point
- pickup latitude: latitude of the start point
- dropoff longitude: longitude of the destination point
- dropoff latitude: latitude of the destination point

3. Related work

Kaggle Challenge: There was a challenge organized by Kaggle that addresses the same problem as ours. Data set was provided by New York City Taxi and Limousine Commission and included pickup time, geo-coordinates, number of passengers, and several other variables. In this solution [3], the author extracted the following features:

- Direction
- Manhattan distance

- Chebyshev distance
- Day of week
- Whether it is night time
- Hour of pick
- Minute of pick
- Day of the week
- etc.

He used the following regressors for the model:

- Random Forest Regressor
- Decision Tree Regressor
- Ada Boost Regressor
- Gradient Boosting Regressor
- SVM Regressor
- XGBoost

The best results were given by XGBoost. The RMSE on the test set was 0.327106.
There were many other entries in the competition that used the similar approach.

Harvard Data Science Final [4]: The objective of this problem was to predict the taxi pickup density. This is a different problem than ours, but the data set is very similar, so it was useful for comparison. Beside using taxi trip data from the NYC government's official website, the authors used the weather data set that was available and it helped them in more accurate prediction.
The main features they used were:

- Discretized latitude/longitude (derived back from geohashes)
- Discretized time - but encoded in 0 to 1
- Other features: number of pickups, day of week, etc

They used two models:

- Random Forest regression
- k-Nearest Neighbors regression

The Random Forest model performed very well with a coefficient of determination (R-squared) on the test data of 0.9505.

4. Methodology

4.1. Pre-processing

All input data was normalized and standardized. Pickup datetime column was used for extracting day of the week, the month and the hour.

4.2. Regression models

Ada Boost Regressor [5] - An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.
We used 50 estimators, with learning rate of 0.001 and square loss function.

A Bagging Regressor [6] is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by

averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.
We used 100 estimators. All samples and all features were used.

Gradient Boosting Regressor [7] builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.
We used 10 estimators with learning rate of 0.0001 and ls loss function.

MLP Regressor [8] optimizes the squared-loss using LBFGS or stochastic gradient descent. The neural network has 3 hidden layers, each having 300 nodes.

Lasso [9] is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer parameter values, effectively reducing the number of variables upon which the given solution is dependent.
We used regularization parameter alpha of 0.01 with a tolerance of 0.0001.

Elastic Net [10] is a linear regression model trained with L1 and L2 prior as regularizer. This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge.
We used regularization parameter alpha of 0.01 with a tolerance of 0.0001.

K Neighbors Regressor [11] is regressor based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. We used a different number of neighbors as parameters: 1, 2, 5, 10, and 20.

4.3. Dimensionality reduction

We used principal component analysis for dimensionality reduction. A couple of approaches were used. We chose to reduce the input to 10 components.

5. Discussion

We had around 40000 samples on which we trained the model and around 3000 that we used for testing model to see whether the regressor predicts well. By increasing the number of training data (up to 1.700.000) we did not get any better results, so we stayed with those that seemed to be the best. We compared the regressors to each other to determine which one offers the best results. K Neighbors turned out to give the best results. We used Root Means Squared Error and R2 Score for evaluation purposes. And the results we came up with showed that RMSE ranges from 250 to 300 while R2 score for test data amounts to 0.76. Finally, the results have shown that a good but not perfect prediction of the duration of driving in New York City is possible.

Regressor	**RMSE**	**R2 SCORE**
Ada Boost	332.55	0.66
Bagging	344.05	0.64
Gradient Boosting	576.25	-0.009
MLP	356.69	0.61
Lasso	345.60	0.63
Elastic Net	347.80	0.63
K Neighbors	**279.22**	**0.76**

6. References

1. https://archive.org/details/nycTaxiTripData2013
2. https://data.cityofnewyork.us/Transportation/2015-Green-Taxi-Trip-Data/gi8d-wdg5
3. https://github.com/mk9440/New-York-City-Taxi-Trip-Duration-challenge-Kaggle
4. https://sdaulton.github.io/TaxiPrediction/
5. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html
6. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html
7. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
8. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
9. http://scikit-learn.org/stable/modules/linear_model.html#lasso
10. http://scikit-learn.org/stable/modules/linear_model.html#elastic-net
11. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html