Weekly Forecast of Fastest Driving Route with Historical Traffic and Weather Data

Seunghwan “Nigel” Kim, Dohoon “Andy” Kim

Advisor: Dr. Sanmay Das

*Spring 2018 CSE 400E `Report*

*Washington University in St. Louis*

[https://github.com/NigelKim/TrafficResearch](https://github.com/NigelKim/SP18MLResearch)

**Introduction**

We live in a world where driving is the standard way of transportation. Accordingly, various map service providers make use of massive traffic data and metadata to predict the real-time fastest driving route between two specific geographic coordinates, often also known as ‘navigation services’. It collects real-time traffic information and optimizes the shortest path to the endpoint with shortest time.

However, predicting a future time’s fastest driving route has not been as fast growing; Google Maps launched a service on December 2017 that predicts the fastest driving route for only a 3-hour future timeframe. If extended to a further timeframe, 24-hour daily prediction or a 7-day weekly prediction becomes possible.

In this study, we integrate historical weather metadata with historical traffic congestion data to train suitable machine learning algorithms and perform a statistical test to assess the performance. Then, with the optimal model, we make prediction about a specified day of the week and ultimately, predict a future 7-day timeframe’s fastest driving routes similarly to a commonly known 7-day weather forecast.

**Data Collection**

For our study, origin and endpoint were arbitrarily selected according to our four initial policies:

1. Selected pair must have more than one competitive output class.
2. Pair must be in a geographical region where we can obtain historical records of fastest driving route.
3. Pair must be in a region where we can access the historically measured congestion information in detail.
4. Pair must be in a region where we can acquire further metadata that can indirectly serve as input features.

Most important of all, we have selected the origin-endpoint pair that can produce multiple competitive fastest driving route(FDR) options. That way, we ensure that our collected historical records have multiple classes of possible FDR outputs instead of a single best FDR option every time, depending on other features specific to a desired day’s condition. As traffic information is a massive data that requires manual measurement or a state government database, our origin-endpoint pair was selected within the city of Chicago, where we have access to a government-provided historical congestion data. Selected origin-endpoint pair and its properties:

|  |  |
| --- | --- |
| Origin (geocoordinates) | John Hancock Center, IL (41.898811, -87.623077) |
| Endpoint (geocoordinates) | Taste of Randolph, IL (41.884276, -87.652300) |
| Distance (km) | 5.168 |
| Duration (s) | 592 |

Table 1. Selected origin and endpoint locations, and their default shortest driving distance with its duration.

To build our training set, we must obtain the true FDR output labels that correspond to each past day in historical time window. Since our experiment was only focusing on a single arbitrary origin-endpoint pair, the exact information is not present in any pre-collected web data. We approached this with two possible methods:

1. Using Google Maps Directions API[1], we integrate Amazon AWS EC2 server-based hourly PHP cronjobs to actively collect FDR output labels throughout the semester.
2. Using Bing Maps REST Services API[2], we integrate Amazon AWS EC2 server and specify past date and time to collect the true FDR output between our origin and endpoint based on past time’s traffic conditions.

The former approach is promising in that we can access real-time traffic information to produce a real-time result, which is iteratively appended to our training set for a semester to obtain sufficient number of data points. However, Google Maps Directions API does not provide us with the road congestion information or roadblocks caused by emergencies and construction[1]. The latter approach using Bing Maps REST Services API was used, for the API provides the total congestion average label of the FDR and the warnings that includes: accident, blocked road, ferry and autorail time table, congestion, disabled vehicle on the road, mass transit incident, planned events, and road hazard[2].

Daily 00:00AM FDR outputs for past dates were iteratively obtained with PHP queries using Amazon EC2 server. Date, time, total average congestion label, warnings, and true output label are extracted from the API response and stored in an SQL database.

From Bing Maps REST Services API response, the total congestion average label outputs only few feature classes: Heavy, Medium, and Mild[2]. By combining it with the fact that our origin-endpoint pair lies in the center of the city of Chicago, we hypothesized that the provided API congestion response was not sufficient to be applied to a busy city area where probabilistically congestion average is classified towards the “Heavy” end of its spectrum and subtle difference between Heavy and Medium congestion is inadequately classified. To incorporate a more powerful means of measurement to measure traffic congestion, we then make use of Historical Congestion Estimates by Segment(2011-2018) data provided from the City of Chicago Data Portal(CCDP)[3]. From the dataset, we can obtain historical daily congestion speed(mi/h) for each pre-allocated Chicago road segments, which is represented as a whole number that can represent the congestion feature more accurately. From CCDP, it is advised to approximately scale the speed in range 0-9 as ‘heavy’, range 10-20 as ‘medium’, and values over 21 are considered as ‘free flow’. However in our study, congestion speed is kept as a real value for better training performance. According to CCDP, there are 1270 traffic segments in Chicago, and each segment has approximately 7 years worth of historical congestion speed data[3].

To assign congestion speed as a feature, we need to manually investigate the output classes of the historical FDR output data collected previously with Bing Maps REST Services API[2]. Each true label is a valid traffic route consisting of a number of unit traffic segments combined. By breaking the route into traffic segments, we matched each segment to a pre-assigned segment ID provided by CCDP[4]. We extracted each segment’s congestion speed of a specific date at 00:00AM and calculated the average congestion speed of all segments that consist a single true label. From this, we can now assign a “congestionSpeed” feature value for every historical date input.

Second part our objective was to integrate metadata that affects the road conditions of past dates. Due to the nature of our study, the metadata must include daily-recorded information for the past timeframe of the historical traffic FDR records. Among many factors that can affect road condition, weather data was chosen, as it directly influences outdoor transportation. Historical daily weather data of Chicago was not accessible to public, and thus a past 5 year chunk historical weather data was obtained from OpenWeatherMap[5]. From the weather data, 3 features—weather condition, temperature, and windspeed—that directly influence the driving were extracted and assigned to each of our main historical FDR dataset. The temperature feature was preprocessed to convert from degrees Kelvin into degrees Celcius before the assignment.

Raw dataset after collection:

Number of data points: 2038 (historical data from 2011/10/01-2018/4/30)

Index:

datetime (format: XX/XX/20XX)

day (format: ‘Mon’, ‘Tue’, etc.)

Features:

congestion (format: label)

congestionSpeed (format: integer)

warningcounts (format: integer)

weather condition (format: string)

temperature (format: real value, in degrees Celcius)

windspeed (format: real value, in mi/h)

Output:

routeoption (format: positive integer multiclass output)

**Data Imputation**

As we concatenated the dataset attained from different sources for congestion speed for segments and weather, we discovered that there existed numerous missing data that we had to handle before applying prediction models. As shown in table 2, Approximately 150 congestionSpeed values were present among 2038 data points, which shows a high percentage of the lack of data we could attain from real source. Weather, temperature and wind speed features were missing 558 data points, which show relatively small percentage of missing feature values compared to the congestionSpeed Feature.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Congestion Speed | Weather | Temperature | Wind speed |
| Missing data | 1938 | 558 | 558 | 558 |
| Total data | 2038 | 2038 | 2038 | 2038 |

Table 2. Number of missing values compared with the total number of data points with respect to congestionSpeed and weather-related features.

Initial thought on deleting the data points that include the missing congestion speed feature was not realized since training with 100 data points was not applicable to machine learning.

Thus, we focused on data imputation to generate values for missing features. Initial implementation was the simple missing-data approach with mean imputation on congestion speed feature and mode imputation on weather categories. But, since we have abundant missing data values, mean imputation method generates identical values for remaining 1938 data points and mode imputation method also fills in with identical values for 558 values, which are huge portion of the data; hence, the latent problem we realized was the possible distortion of distribution of the variable and significant decrement in the variance that would severely affect our training algorithm. Another problem we came up with dealing with mean and mode computation is that single imputation does not generate the uncertainty in the imputations, which means certainty generated through the dataset does hinder the variance of our dataset and further reduces the accurate prediction.

The modern method we came up with dealing with abundant missing data values was Multivariate imputation by chained equations (MICE) that is multiple imputation method and recently emerged as a principled way of dealing with missing data. Multiple imputations enhances multiple iterations on filling in missing values, creating multiple complete datasets. Different from the single imputation method such as mean imputation, multiple imputations generate the uncertainty in the dataset and incorporate precise standard errors. MICE operates under the assumption that the missing data are Missing At Random (MAR), which means the probability of a value missing depends only on observed values, not on unobserved values[6]. Thus, for the application of MICE method, we assume our missing data are Missing At Random with the correlation between the features, congestion speed and weather.

Mice steps are illustrated as below[6]:

1. Single imputation, such as mean imputation, is applied for imputing all the missing values in the dataset. These imputations are treated as place holders for further use.
2. The place holder single imputations for one variable are reset as missing values.
3. Regression model is applied for the observed values from the variable in step 2 on the other variables in the imputation model.
4. The missing values in the variable in step 2 are filled with predictions from regression model set up from step 3. The variable is accounted as a dependent variable and other variables as independent. When the variable is subsequently used as an independent variable in the regression models for other variables, both the observed values and imputed values are used.
5. Steps 2 to 4 are repeated for each variable with missing data and the cycling through each of the variables is one cycle. When one cycle ends, data is filled up with predicted values from regression that show the relationships observed in data.
6. With specific iterations, steps 2 to 4 are repeated with the subsequently updated imputed values. At the last iteration, we get the final imputations set up which should have converged to stable distribution. With multiple iterations, the dependence on the order in which the variables are imputed.
7. The optimal number of iterations is essential to determine. And, our application of 5 iterations and 10 iterations beget the same imputed values in the missing values at the end. Hence, we determined 5 iterations for the process.

Multivariate imputation by chained equation has been applied on our dataset with application of MICE library in R and before we move on to selecting appropriate machine learning prediction models. In figure 1, the correlation of the features in imputed dataset using MICE is shown using heatmap:

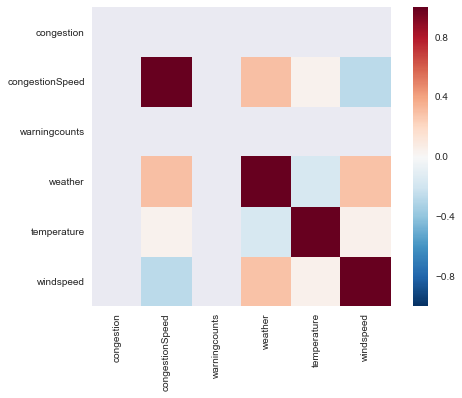


Figure 1. Heatmap showing correlation between the features of imputed dataset.

From figure 1, we can see there is a bit of correlation between congestion speed and weather; however, it is clearly shown that there are almost no correlations between the features, which lead to independence among features.

**Model Selection**

Binary Logistic Regression:

By analysis of true label outputs, there were 2 classes, which means two routes were the most competitively fast driving routes. Hence the problem in reduced to a binary classification problem, and we utilize a classic linear classifier to start with.

We used binary logistic regression suitable for our dataset with dichotomous outputs, for we are unsure of the linearity between independent and dependent variables. Moreover, each observation of our dataset is independent, and each independent variable has low correlation and thus it is valid to apply the binary logistic regression.

Linear SVM:

SVMs are generalized linear classifiers and treated as advanced version of perceptron algorithm[8]. With support vectors and margins, SVM is applied to more flexibly incorporate the data points, hence we decided to implement SVM on our dataset with thought on SVM will perform well applicable to binary classification. For our dataset, among the various kernels, such as polynomial and RBF, we chose to use the linear SVM since we figured our feature space is not high dimensional and there is no need for nonlinear transformation to deal with high dimensional feature space. Thus, we chose linear SVM for binary classification model. For the model parameters, we applied with the l2 penalty and hinge loss because it’s a classification model. Also, tolerance is determined to 0.0001.

Adaboost:

Adaboost is an advanced ensemble technique that learns an accurate strong classifier by combining an ensemble of inaccurate weak learners[7]. The ensemble of weak learners is combined into the strong classifier through the weighted majority vote and the algorithm selects the most pivotal features and give different weights for them. Among the ensemble methods, we selected Adaboost method for its robustness to overfitting and advanced generalization property with important feature selection. Since we focused on binary classification on our route options from relatively independent features we have, iterations over weak classifiers and updated weights on features to build the strong classifier in Adaboost method is thought to be reasonable boosting method.

**Result/Discussion**

After 10 re-runs of 10 fold cross validation, we obtained the mean test accuracy and the mean cross validation accuracy using Python Scikit-Learn library.

The first model was binary logistic regression, which produces below results:

Model Accuracy: 0.863

Accuracy(Mean CV): 0.875 (+/- 0.004)

To check the performance of our model, Receiver Operating Characteristic(ROC) curve and Area Under the Curve(AUC) were measured:

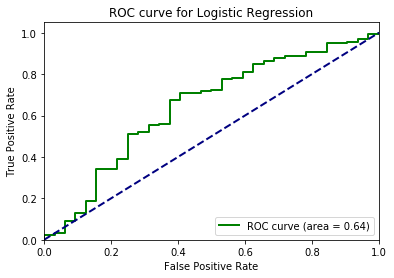


Figure 2. ROC curve and AUC value for Logistic Regression

The next model was Linear SVM, which produces below results:

Model Accuracy: 0.880

Accuracy(Mean CV): 0.875 (+/- 0.004)

To check the performance of our model, Receiver Operating Characteristic(ROC) curve and Area Under the Curve(AUC) were measured:

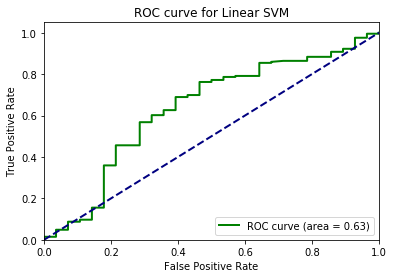


Figure 3. ROC curve and AUC value for Linerar SVM

Our final model was Adaboost, which produces below results:

Model Accuracy: 0.863

Accuracy(Mean CV): 0.875 (+/- 0.004)

To check the performance of our model, Receiver Operating Characteristic(ROC) curve and Area Under the Curve(AUC) were measured:

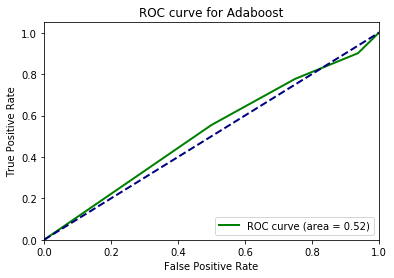


Figure 4. ROC curve and AUC value for Linerar SVM

From the comparison of accuracy and mean cross validation score, we observe very high accuracy values. On the other hand, from figure 2,3, and 4, we can observe the AUC of ROC curve between 0.5-0.7 for all three models. For binary logistic regression and linear SVM we obtained AUC values slightly higher than 0.6, and for Adaboost we obtained AUC of 0.52. From observing the AUC of ROC curve close to 0.5, we hypothesize that our algorithms may rather be classifying the test set randomly, and a probable cause may be the lack of information on our dataset or simply the true label class distribution is disproportionate. Hence, we went back to our dataset and observe the true label classes and found out that approximately 87.5% of the data were in the first true label class, and 12.5% in the second true label class.

Furthermore, we applied K-means clustering with 2 clusters to investigate on grouping of data points without using the output labels.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Congestion | congestionSpeed | Warningcounts | Weather | Temperature | windspeed |
| Centroid 1 | 0.000 | 1.765 | 0.000 | 2.508 | 0.778 | 4.307 |
| Centroid 2 | 0.000 | 1.783 | 0.000 | 1.926 | 12.438 | 4.602 |

Table 3. Means of features when clustered into 2 clusters

From table 3, we can investigate that the two centroids for dataset generated from MICE have relatively distinct means for weather, temperature, and wind speed feature but there is no significant difference between the congestionSpeed means. Thus, we chose two features—congestionSpeed and weather—since they are relatively correlated to each other.

Visualization with comparison of congestioSpeed and weather:

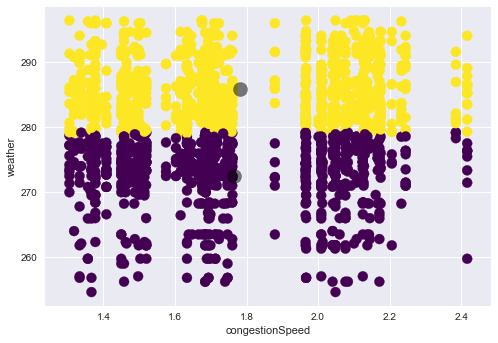


Figure 5. Dataset clustered with respect to features: congestionSpeed and weather

By figure 5 and comparing only congestion speed and weather subject to two centroids in clustering, we conclude that two centroids are centered in similar congestionSpeed means but weather means are evident to have different mean values from each other.

**Conclusion/Limitations**

Machine learning algorithms perform well if there is a “good” dataset to learn from. Most models are not suitable for dealing with missing data, especially for a real world dataset such as the one this study presents. Traffic-related information is particularly difficult to collect, for it is a massive amount of real-time data inter-correlated with social policies and random event; it is very easy to have outliers. To construct a proper dataset from the traffic metadata, we would need both a huge computational power and collectable traffic-related records that accumulate over time. By incorporating MICE, we were able to impute the missing features of our dataset, but the nature of lack of information and high proportion of missing data in traffic information led to a poor prediction model performance. From the analysis of the 3 models—logistic regression, linear SVM, Adaboost—we could clearly see some known behaviors of each algorithm. Because Adaboost is typically weak to outliers, even though the difference is small, it is performing relatively worse from other algorithms; its AUC was around 0.5, indicating that it is bluntly performing random prediction because of the nature of the abundance of outliers in our dataset. Logistic regression and linear SVM tend to predict slightly better than Adaboost, but did not produce a meaningful AUC value over 0.7. By performing a Welch’s T test, there is no significant statistical difference in the efficiency or accuracy of the three models, and we conclude that the dataset still needs improvements in order to properly measure the performance of each investigated model.

Moreover, in our study of predicting future’s Fastest Driving Route particularly brings another problem in the prediction phase. It is very hard to build a working test set in the real world, for the test set must consist only of unmeasured future’s traffic meta-information that are vulnerable to random event.

**Future Work**

First of all, we can improve the disproportionality of our true label outputs by carefully considering the location of our origin-endpoint. For the true labels to be meaningful, we desire to see a more uniform distribution of each class. For that to be possible, our origin-endpoint pair is best to have more than two uniformly probabilistic distribution of true label class occurrence. Second of all, we must reduce the amount of missing values in our features, especially the critical features that influences our prediction importantly. Reiterating from our 2 methods of acquiring data from the data collection stage, the second method was proven to be insufficient due to the abundance of missing data. The first method was discarded in the early stage of our study due to a data race caused by computational overhead of Amazon EC2 Free Tier server and because of the fact that it can only physically collect a small semester-long dataset that will approximately contain 100 data inputs. The most promising direction is to invest at least a year with a massive computational power to actively construct daily real-time input information; we will have an almost complete dataset with a very little percentage of possible missing values. Third, we can extend the collection of traffic-related metadata to not only weather but also both spontaneous and scheduled event that occurs, which often can take place for as little as an hour to several month such as a construction. Last but not least, when applied to real world situation, unobserved set of the near future can be constructed if we utilize Google Map’s “Typical Traffic” information of the day and time of week integrated with the historical average congestionSpeed that can be calculated from the historical traffic data. Weather-related features are relatively easier to collect future information, for weather prediction is in a much bigger timeframe and the future weather predictions are available.

**References**

1. Google Maps Directions API. <https://developers.google.com/maps/documentation/directions/intro>
2. Bing Maps REST Services API. <https://msdn.microsoft.com/en-us/library/ff701713.aspx>
3. Chicago Data Portal. “Chicago Traffic Tracker - Historical Congestion Estimates by Segment - 2011-2018.” <https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Historical-Congestion-Esti/77hq-huss>
4. Chicago Data Portal. “Chicago Traffic Tracker - Congestion Estimates by Segments.” <https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Congestion-Estimates-by-Se/n4j6-wkkf>
5. OpenWeatherMap. *Chicago Weather History Bulk(2012/10-2018/3).* JSON file.
6. Melissa J.Azur, Elizabeth A. Stuart, Constantine Frangakis, Philip J. Leaf. (2011) “Multiple imputation by chained equations: what is it and how does it work?”
7. Divya Ramani, Harshita Kanani, Chirag Pandya. (2013) “Ensemble of classifiers based on association rule mining”
8. S.B. Kotsiantis. (2007) “Supervised machine learning: a review of classification techniques”