Bus Delay Prediction using Transit Information

Akanksha Rajpute(19210135), Akhil Alfons Kodiyan(19210912), Kirthy Francis(19210588), Tejal Vijai Nijai(19210412)
Engineering and Computing, Dublin City University, Dublin, Ireland
Project link : https://github.com/kirthy21/CA683-Bus-Delay-Prediction-using-Transit-Information
Dataset link : https://storage.googleapis.com/bustripdata/RealTimeTripData_v2.csv

***Abstract*—Travel time is a vital component of high-quality infrastructure delivery in road-based transit networks. Prediction of travel time has been largely popular in the last few decades and many models using a variety of data sources, have been proposed. This paper attempts to predict the delay in transit time for a transport network of this kind and tries to compare the forecast quality of various delay models using the recently recorded conduct of the public transit vehicle data. The data used in this analysis was scraped from Dublin-BusRTPIService provided by Transport for Ireland (TFI). The data was cleaned and processed before feature engineering was performed. Then machine learning models were applied to determine the best model for prediction of delay in travel time. The paper also focuses on the data collection and preprocessing as it is a crucial part of which foundation of prediction is laid.**

***Index Terms*—Travel Time, KDD, Dublin Bus, Delay, Feature Extraction, Machine Learning, SVR, Random Forest, XGBoost**

I. Introduction

People around the world depend on public transport for their day to day travel. As the world keeps developing, people need to realize when to anticipate delays, particularly for transport rides, which are inclined to get held up by traffic. In advanced public transport systems, the installation of travel time prediction models will support both transit agencies and passengers. Providing the passengers with correct real-time transit vehicle-arrival time details will boost service quality for the transit. Prediction of travel time is also essential for bus operators as demand for public transport services is strongly related to economic activity. Estimation of travel time between two locations is the main aim of bus travel time prediction.

A variety of methods have been proposed and employed to predict travel time effectively. In this paper, we implement and compare various machine learning models to determine the best model for the data scraped. Before applying the models, we need to clean the data and reduce its dimensions. Dimensionality Reduction has become the focus of the ML process to avoid unnecessary computing power/cost, over-learning and predictive errors. In this regard, redundant features that may have nearly identical predictive value to other features may be excluded without significantly affecting the learning process. Likewise, insignificant elements should also be eliminated. Feature Engineering focuses not just on extracting a subset from the optimal set of features, as well as on constructing new feature sets previously overlooked by ML techniques. This also involves minimizing the greater dimensions to lower dimensions in order to extract the value of the feature [10]. To develop our model, we derived training data from sequences of bus positions over time, as received from Dublin Bus real-time feeds, and cleaned the data using Google Cloud Platform, Tableau Prep builder and Python. Final cleaned data was then feature engineered to make the right assumptions before applying to the model.

II. Related Work

Over the past decades, a number of bus arrival / running-time prediction models have been developed. The proposed work will use bus schedules and live trip data to predict delays in travel time. The work of Comi et al. [3] gives us a good understanding of the various factors that affect the travel times of public buses, which should be taken into consideration for a good delay prediction system. Chen et al. [2] propose an ANN model and a KF algorithm to predict travel time which could be a good choice to predict travel time, as the input required in this model is very similar to that of this study. Whereas Li et al. . [4] use a multistep process using Markov historical transfer model and KF to predict delay time. However, this process of prediction is complex, and this method is said to yield higher accuracy. Karim et al. [5] provide an excellent way to classify multivariant time series that can be used to classify delay patterns along with other contributing factors like weather, time of the week, special occasion, etc as described by Comi et al. [3]. Support Vector Machines is another commonly used method for predicting travel times. Junyou et al. [6] classified public transit GPS data using SVM based prediction algorithms and then verified the algorithm with actual operational data. Although commonly used for classification, the principles of SVM can be used in regression and prediction, as demonstrated by ChiangHong et al. [7]. Bin Yu et al. discusses that the Random Forest model has good performance in non-linear regression and proposed a model that combined Random Forest and near neighbours method. [1]

III. Data Mining Methodology

The Knowledge Discovery in Databases or KDD method helps to operate on large-volume data and mine critical and valuable knowledge by flowing the data through various stages. In our research, we adopted the KDD method that has been effective in seeking a better technique for mining. The

data used in this model has been scraped from DublinBus-RTPIService published by Transport for Ireland (TFI), which provides real-time bus information, via SOAP API.

A. Data Collection

The data was scraped using an online job, which hits soap endpoint periodically in fixed intervals, gathers the responses, and then commits them to an SQL database. This data mainly consists of three categories.

- Route information – i.e. bus stop numbers, direction of the route, the sequence of stops on a Route
- Bus stop information - i.e. bus stop number, address, latitude, and longitude of stop
- Realtime stop information – i.e. expected time of arrival, recorded time of arrival, journey reference, block reference, vehicle reference.

The information was retrieved at a rate of about 7000 records/min and had a size of 10 GB. The fetched information was pushed into tables in an SQL database.

B. Data Preprocessing

The scraped data was separated across tables in the SQL database. This data was too large and challenging to work on; therefore, unnecessary, redundant data was removed, and the tables were merged using Tableau Prep as the system could not process the large amount of data. After the flow processing in Tableau Prep, the merged data was loaded into BigQuery tables for easier and faster access. After exploring the data, which consisted of over 22 million records, routes 42,1,11,43 were prioritized, over other routes with partial route information and used in the model for prediction.

C. Feature Engineering

Feature engineering was performed after cleaning and sorting the data to improve the results. Attributes that were relevant or provided information to our prediction were selected and other attributes that were either redundant or not relevant were dropped. Also, several categorical variables, such as the address of the bus stop and the direction of the route (inbound/outbound) were encoded into numerical variables. Encoding was done using the label encoding feature present in the scikit learn library which encodes the categorical data into respective numerical data, so that they can be used in the machine learning model, and also helps in faster execution. Synthetic data was generated to meet specific needs or certain conditions that were not be found in the original, real data. This was useful when designing our system because the synthetic data was used as a simulation or a theoretical value. This allowed us to take into account the unexpected results and have a basic solution when the results proved to be unsatisfactory. After analyzing the correlation between the attributes available, we constructed synthetic variables multiplying other attributes which later showed a good correlation among the other attributes. TimeWithBusStopAddress, for example, multiplying HourOfDayStepped and BusStopAddressEncoded, showed a significant correlation with our prediction attribute.

D. Regression Models

We have used three different Regression models for prediction. These were selected after performing research, and the criteria were efficiency, fast and better prediction results with the least errors.

1) Support Vector Regression: Support Vector Regression model allows us to fit the error within a fixed threshold. We wanted to minimize the error; therefore, we used SVR so that we can fit error in a specific range. It is also beneficial as it is designed to deal with the data points close to the boundary and our data set is similar to that. The error term is handled within the limits where we set an absolute error less than or equal to the defined margin, called the maximum error, ϵ (epsilon). Epsilon can be adjusted to achieve the optimal accuracy of the model [11].SVR works with continuous data which is well suited for our situation as we are dealing with time-series data.

2) XGBoost Regression: XGBoost is an algorithm that most recently dominated applied machine learning. They are gradient-strengthened decision trees built for speed and performance. XGBoost stands for eXtreme Gradient elevations and is very quick compared to other implementations that improve gradient. It is well suited with our data set as it is highly scalable and very fast compared to any other gradient boosting algorithm, i.e. the next possible model combines with the previous model to minimize the error. It is based on Decision Trees. This algorithm performs very well with tabular data sets.

3) Random Forest Regression: Random forest is a Supervised Learning algorithm and is a bagging technique, i.e. it generates additional data and decreases the variance. It constructs a multitude of decision trees and gives mean prediction (regression) of each tree. We used this algorithm to increase the accuracy of the prediction and to prevent overfitting.

IV. Experiments

For the prediction of arrival delay, we have implemented three Machine Learning models with data obtained from feature extraction and also without feature extraction.

A. SVR

Support Vector Regression (SVR) uses the same principle as SVM but used for regression problems. We implemented GridSearchCV to test and find different SVR model hyperparameters that are most suited to our data. We found, as per our training data, that higher gamma values would attempt to match the data set, i.e. cause generalization error and over-fitting issues. Gamma=0.001 was found to give the best results for our model. Another hyperparameter is epsilon, which denotes how much error you are prepared to enable per instance of training data. We used epsilon=1 as it gave better results.C is another hyperparameter that defines the cost of misclassification. A large C gives you low bias and high variance. Low bias because you penalize the cost of

missclasification a lot. A small C gives you higher bias and lower variance. We found C=100 to be the optimum value for our model.

These hyperparameters enabled the model to boost the prediction with the featured engineered data being applied. The percentage shown rises from 28 per cent to about 55 per cent.

SVR Model execution (Before Feature Engineering):

```
[ ] from sklearn.svm import SVR
    svr_rbf = SVR(kernel='rbf', C=100, gamma=0.001, epsilon=1)
    svr_rbf.fit(train_x, train_y)
    y_pred_svr = svr_rbf.predict(test_x)
    svr_rbf.score(test_x,test_y)
```

```
0.2843515977268084
```

Fig 1 - Support Vector Regression before Feature Engineering.

SVR regressor Model(After Feature Engineering)

```
[ ] from sklearn.svm import SVR
    svr_rbf = SVR(kernel='rbf', C=100, gamma=0.001, epsilon=1)
    svr_rbf.fit(train_x, train_y)
    y_pred_svr = svr_rbf.predict(test_x)
    svr_rbf.score(test_x,test_y)
```

```
0.5468251486414953
```

Fig 2 - Support Vector Regression after Feature Engineering.

B. XGBoost Regression

We have used XGBoost because of the run speed and layout productivity it provides. We experimented with various parameters before arriving at the parameters that gave us good results. One such parameter is Alpha, which is used to reduce pattern overfitting. The hyperparameter: colsample_bytree specifies the percentage of functions (columns) used to construct each tree. The collection of features is likely to be different for each tree (it may be the same due to chance, though it is highly unlikely). The objective parameter was set to reg: squared error to denote linear regression. The n_estimators are used to refer to the size of the trees that are to be created. After applying this model to the data that was feature engineered, the prediction percentage showed an improvement of about 1 per cent.

XGBoost Model Execution (Before Feature Engineering):

```
[ ] import xgboost as xgb
    from sklearn.metrics import mean_squared_error
    xg_reg = xgb.XGBRegressor(objective ='reg:squarederror', colsample_bytree = 0.8, learning_rate = 1,
                    max_depth = 5, alpha = 10, n_estimators = 100)
    xg_reg.fit(train_x,train_y)
    preds_y = xg_reg.predict(test_x)
    rmse = np.sqrt(mean_squared_error(test_y, preds_y))
    print(xg_reg.score(test_x,test_y))
```

```
0.6837655067939301
```

Fig 3 - XGBoost Regression before Feature Engineering.

XGBoost Regressor Model(After Feature Engineering)

```
[ ] import xgboost as xgb
    from sklearn.metrics import mean_squared_error
    xg_reg = xgb.XGBRegressor(objective ='reg:squarederror', colsample_bytree = 0.8, learning_rate = 1,
                    max_depth = 5, alpha = 10, n_estimators = 100)
    xg_reg.fit(train_x,train_y)
    preds_y = xg_reg.predict(test_x)
    rmse = np.sqrt(mean_squared_error(test_y, preds_y))
    print(xg_reg.score(test_x,test_y))
```

```
0.692547970879114
```

Fig 4 - XGBoost Regression after Feature Engineering.

C. Random Forest Regression

Random forest algorithm is an algorithm that groups data trees. One of the hyperparameters that can be defined is the number of trees to be formed i.e. n estimators. We might assume that more trees would be able to generate a more generalized result, but then, the time complexity of the Random Forest model increases drastically with a larger number of trees. In our analysis, we found that it gives almost the same result for n estimators greater than 100, which means that it's not the best choice to use a larger number of estimators in our random forest model. While it won't change the layout, it saved the difficulty of the calculation and avoided the overhead on our CPU. By default, the random_state has been set to 0 to generate the same result every time. We built a model for testing before and after the implementation of feature engineering. From our tests, the prediction percentage score after feature engineering is 66.49 percent which indicates a small quality increase in performance compared to before feature engineering.

V. Results and Findings

Table 1 : Score Percentage

RandomForest Model Execution(Before Feature Engineering):

```
[ ] from sklearn.ensemble import RandomForestRegressor
    regr = RandomForestRegressor(n_estimators = 100, random_state = 0)
    regr.fit(train_x, train_y)
    regr.predict(test_x)
    regr.score(test_x,test_y)
```

```
0.6645777325550246
```

Fig 5 - Random Forest Regression before Feature Engineering.

RandomForest Regressor Model(After Feature Engineering)

```
[ ] from sklearn.ensemble import RandomForestRegressor
    regr = RandomForestRegressor(n_estimators = 100, random_state = 0)
    regr.fit(train_x, train_y)
    regr.predict(test_x)
    regr.score(test_x,test_y)
```

```
0.6649772355300012
```

Fig 6 - Random Forest Regression after Feature Engineering.

Feature Engineering	**Machine Learning Models**		
	SVR	***XGBoost***	***Random Forest***
Before	28.43	68.37	66.45
After	54.68	69.25	66.49

In our research, we adopted the KDD method that has been effective in our analysis, and we tried feature engineering to boost the outcome of our predictive model. Results of before and after the application of feature engineering are recorded in the table and it is evident that feature engineering provided better results. Also, It is observable that the XGBoost model gave better predictions when compared with the SVR and Random Forest models

VI. Conclusion

Data mining and data cleaning is a crucial part of getting good accuracy and predictions from models. Using the web scraped public bus data, we were able to predict transit delay with nearly 70 percent accuracy, which is one of the best seen in related works of this kind. Also, through this study, it is clear that data cleaning plays a vital role in the precise notion of the data, to take the fitting assumptions, and also the need to perform feature engineering appropriately. Comparatively, XGBoost emerged as the better predicting model while compared to SVR or Random Forest even without synthetic features. So it can be argued that XGBoost works well with this type of data, and also that feature engineering can improve results. In the future, we might be able to improve this model using features from diverse sources like historical traffic data, weather patterns, road conditions, proximity to schools, etc.

References

[1] Yu, Bin, Huaizhu Wang, Wenxuan Shan, and Baozhen Yao. "Prediction of bus travel time using random forests based on near neighbors." Computer[U+2010]Aided Civil and Infrastructure Engineering 33, no. 4 (2018): 333-350.

[2] Chen, Mei Liu, Xiaobo Xia, Jingxin2 Chien, Steven I-Jy. (2004). A Dynamic Bus-Arrival Time Prediction Model Based on APC Data. Computer-aided Civil and Infrastructure Engineering - COMPUT-AIDED CIVIL INFRASTR E. 19. 364-376.

[3] Comi, A.; Nuzzolo, A.; Brinchi, S.; Verghini, R. Bus travel time variability: Some experimental evidences. Transp. Res. Procedia 2017, 27, 101–108.

[4] Li, Jinglin, Jie Gao, Yu Yang, and Heran Wei. "Bus arrival time prediction based on mixed model." China Communications 14, no. 5 (2017): 38-47.

[5] Karim, Fazle Majumdar, Somshubra Darabi, Houshang Harford, Sam. (2018). Multivariate LSTM-FCNs for time series classification. Neural Networks. 116.

[6] Junyou, Zhang, Wang Fanyu, and Wang Shufeng. "Application of Support Vector Machine in Bus Travel Time Prediction." International Journal of Systems Engineering 2, no. 1 (2018): 21-25.

[7] Hong, Wei-Chiang, Yucheng Dong, Feifeng Zheng, and Chien-Yuan Lai. "Forecasting urban traffic flow by SVR with continuous ACO." Applied Mathematical Modelling 35, no. 3 (2011): 1282-1291.

[8] Gal, Avigdor, Avishai Mandelbaum, François Schnitzler, Arik Senderovich, and Matthias Weidlich. "Traveling time prediction in scheduled transportation with journey segments." Information Systems 64 (2017): 266-280.

[9] A Study on Multiple Linear Regression Analysis — Elsevier Enhanced Reader. https://reader.elsevier.com/reader/sd/pii/S1877042813046429 (accessed Apr. 21, 2020).

[10] [1]M. F. Uddin, J. Lee, S. Rizvi, and S. Hamada, "Proposing Enhanced Feature Engineering and a Selection Model for Machine Learning Processes," Applied Sciences, vol. 8, no. 4, p. 646, Apr. 2018, doi: 10.3390/app8040646.

[11] T. Sharp, "An Introduction to Support Vector Regression (SVR)," Medium, Mar. 04, 2020. https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2 (accessed Apr. 21, 2020).

[12] Khurana, U., Samulowitz, H. and Turaga, D., 2018, April. Feature engineering for predictive modelling using reinforcement learning. In Thirty-Second AAAI Conference on Artificial Intelligence.

[13] Barse, E.L., Kvarnstrom, H. and Jonsson, E., 2003, December. Synthesizing test data for fraud detection systems. In 19th Annual Computer Security Applications Conference, 2003. Proceedings. (pp. 384-394). IEEE.

[14] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16, San Francisco, California, USA, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[15] J. Brownlee, "A Gentle Introduction to XGBoost for Applied Machine Learning," Machine Learning Mastery, Aug. 16, 2016. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (accessed Apr. 22, 2020).