Prediction & Classification of Anomalies in Transport Network using Dublin Bus Transit Feeds

Akhil Alfons Kodiyan (19210912), Kirthy Francis (19210588)
Dublin City University
Dublin,Ireland
akhil.kodiyan2@mail.dcu.ie, kirthy.francis2@mail.dcu.ie

***Abstract*—Travel time is a vital component of road-based transit networks and delays are a major aspect that affect this transit time. This paper focuses on building a system that collects transit feed in real time and uses that information to predict delays. The predicted delays, along with attributes from transit feed, are used to group delays into classes based on similarities and then further maps these classes to causes. We also try to compare the quality of various forecast and classification models. This study is based on real-time bus transit information published by Dublin Bus. The experimental data used in this analysis was collected using java/spring based batch jobs scheduled based on publicly available bus transit schedule. The collected data, which amounts to about a quarter million records a day, are then pooled in to Google BigQuery, where the data is cleaned and preprocessed. The data collected roughly estimates to be about 5 million records and growing. The delays are then predicted using models based on regression, ensemble and ANNs, and the results are compared. The similarity of patterns in the predicted delays, along with other attributes, are then used to group these delay instances to various probable groups using unsupervised clustering methods like K Means, DBScan and OPTICS algorithms, and these groups are then labelled with causes. The cause-groups are then used to train classification models to predict the cause of new incoming real time transit delay record.**

***Index Terms*—Smart Transport, Delay Prediction,Anomalies classification**

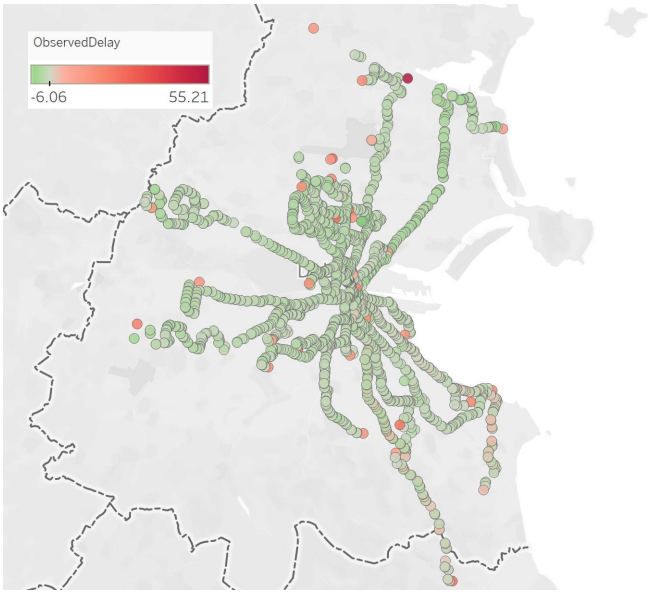

Fig. 1. Average Delay Across Bustops in City

I. Introduction

People around the world depend on public transport for their day to day travel. As the world continues to develop, people need to be equipped with the ability to anticipate delays, particularly for transport rides, which are inclined to get held up by traffic, weather or any other causes. In advanced public transport systems, the installation of travel time prediction models supports both transit agencies and passengers. Providing the passengers with accurate real-time transit vehicle-arrival time details boosts service quality. Prediction of travel time is equally essential to bus operators as demand for public transport services is strongly related to economic activity. Estimation of travel time between two locations is the primary aim of bus travel time prediction.

Dublin Bus is the biggest public transport provider in the Greater Dublin Area operating over 136 routes. Normal services run from 5.00am until midnight with an additional Nitelink service from midnight till 4.00 am on Fridays and Saturdays [1]. Normal services of 17 routes which pass through the City Centre were chosen for our study assuming that the Dublin City Centre would be more susceptible to delays. The data was obtained from the DublinBus RTPIService [2] which provides a SOAP API that regularly publishes route information. A variety of methods have been proposed and employed to predict travel time effectively. In this paper, we implement and compare different machine learning models to determine the best model for transport data. Before applying the models, the data was cleaned and dimensionality reduction was employed to 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. To develop our model, we derived training data from sequences of bus positions over time and cleaned the data using Google BigQuery [3], Tableau Prep builder and Python. Final cleaned data was then feature engineered to make the right assumptions before applying to the model. Clustering methods were applied to this data to obtain groups or patterns among the records classified Late and assign causes to the groups. Once causes

were obtained, they were used to classify and predict delays in incoming unlabelled records.

II. Related Work

Detection and prediction of delays in transport networks is widely researched and studied in the past decades. Delays can be attributed to many aspects. The work of Comi et al. [4] gives us details of the various factors like time of the day, day of week etc. that affect the travel time of transport systems, which could be taken into consideration for a good delay prediction system but they do not consider an important element, that is the history of a trips behaviour which we include in our model. Chen et al. [5] proposes 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, we also plan on extending this by including more feature engineered attributes and try to improve the predictability of the delay. A more recent study by Dairui et al. [6] on Dublin Bus data, used interpolation to predict the arrival time using nearest point search. Whereas Li et al. [7] used a multistep process using Markov historical transfer model and KF to predict delay time, though this process to prediction is complex, this method is known to yield higher accuracy. Karim et al. [8] provides a means of classifying multivariant time series that can be used to classify delay pattern along with other contributing factors like weather, time of the week, special occasion etc. as described by Comi et al. [4] due to limitations of the attributes available in the data set, this flow of work cannot completely be applied in this study. The work of Cristóbal et al. [9] bears some resemblance to our work as it uses historical travel time to distinguish recurring delay and sporadic delay, and group delay patterns and other parameters based on similarity to arrive at cause for the delay. Junyou et al. [10] classified public transit GPS data using XGBoost based prediction algorithms and then verified the algorithm with actual operational data. The HTTP framework described by Wang-Chien et al. [11] uses historical data by identifying and grouping similar trajectories and then combines two hybrid prediction schemes which outperforms state-of-the-art schemes. Whereas Gal et al. [12] segments a bus journey and uses a combination of Machine Learning and Queueing Theory predictors to model travelling time in each segment. They find that the length of the journey does not influence the predictions and supports applying mixed Queue and Machine Learning predictors in similar settings. Several studies include clustering algorithms like K-means to classify transport operations like Kadir et al. [13] did in their research. DBScan is another popular clustering algorithm used for traffic classification as can been seen in the work of Ernman et al. [14].

III. Data Collection

A. Data Source

The data used in this study was sourced from DublinBus RTPIService, which publishes real-time bus transit information for use of public. This data source mainly consist of data such as the real time time status of bus stop, serviced routes, service schedule, operators operating in various routes [2]. Data was collected from these endpoints using periodically scheduled batch jobs and this data was used to update our data pool with the information, for tracking and job scheduling. We used serviced routes and service schedule for scheduling the batch, and to collect the real-time information which is rudimentary for subsequent analysis. We tried to focus our study by sampling about 17 North-South routes (Fig. 1) which traverse through the City Centre, as they pass through common regions around 40% of the stops on the routes, and thus gives us good and frequent data points along these routes. The purpose of the route sampling was to reduce volume of data collected and thereby ease up pressure on the Extract Transform Load (ETL) pipeline and subsequent storage and analysis costs.

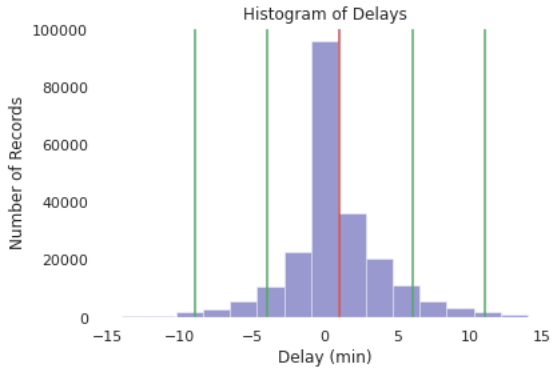

Fig. 2. Distribution of delays

B. System architecture

The ETL system presented in Fig. 3 was used to collect/process and analyse the data, it is composed of four components. The trip schedule creator module is a java based spring batch for creating schedules from the scraped data for the real time information collection. This schedule is used by the real-time trip data scraping module for collecting real-time bus transit information. Real-time trip data scraping module is a high performance data collection approach, created using java, spring batch, quartz scheduler and caffeine cache for accurate status monitoring of multiple trips simultaneously, while reducing data delay as much as possible and also decreasing the burden on the RTPI API. The data obtained are then cleaned and trip matched by the Cleaning and matching module before it is sent to storage in batches. Tracking data, which amounts to about a quarter million records per day, is then streamed in batches to BiqQuery for storage, cleaning and feature engineering. The tracking data is cleaned and feature engineered using Big query analytic queries, due to the sheer volume of the data, which approximates to 5 million rows. After cleaning and feature engineering the data, it is ported to different tables for usage in prediction and classification workbooks and the results were fed back to the original

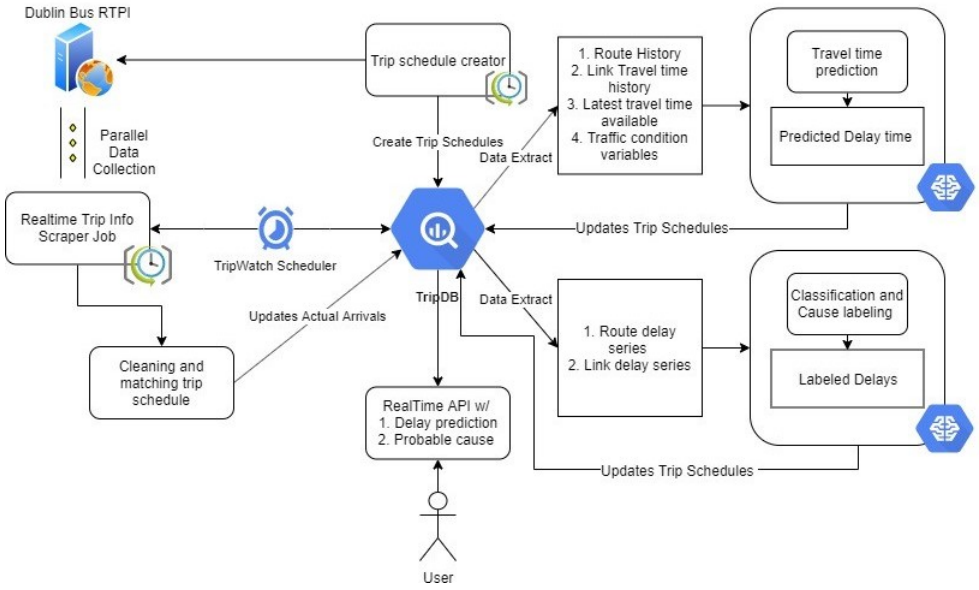

Fig. 3. System Architecture

tables. The final module is the Cause finder module with the prediction and cause of delays which aims to make the data available for the users as an enhanced feed. All ETL components are deployed on AWS Beanstalk where as the data is stored in Google Cloud Platform BigQuery and Storage

C. Data Overview

The data used in the study was scraped during the period of June-August 2020. We choose around 17 routes which traverse through the city centre, roughly from north to south of Dublin so as to ensure maximum common bus stops across these routes. Advantage of this approach is that we could cut down the volume of the data and thereby decrease the pressure on the ETL systems, while getting frequent updates, especially in the common ares. In addition these common areas largely coincide with the Dublin City Centre and the routes also frequent commuter routes which are prone to traffic delays (Fig. 5). The final data snapshot consisted of over 5 million raw records. On prima facie analysis, we observed a large number of missing stops which upon manual inspection was found to be due to the buses skipping stops as a result of lesser number of travellers owing to COVID 19 pandemic.

D. Data Model

Though data items like trip schedule and route plans were used in collection of data, the main data points used in study comprises of real time tracking data for each bus stop. The main attributes of this data are included in (Table II).

The raw attributes were processed to derive synthetic attributes for better performance in prediction and classification. Some of the derived features are DistanceInMeters, ArrivalDelayPrediction, DayofWeek, HourofDay, TripTypeId, HistoricalAverageDelay, details of which can be found in Section IV. Some of these are direct derivatives of the original

TABLE I
NUMBER OF RECORDS AND STOPS IN ROUTES

RouteId	Number of Records	Number of Stops
39	603690	86
41	541767	60
40	511558	83
13	484641	92
16	456056	76
46A	423749	62
4	368027	60
83	333922	67
9	306852	68
7	304473	76
123	212190	46
140	197477	46
11	191719	65
1	175802	42
42	60719	64
44	27054	81
47	17415	54

TABLE II
DATA DESCRIPTION OF THE SCRAPED FIELDS

Field	Description	Type
routeId	Route Id	String
direction	Direction of route	String
destinationId	Id of destination of route	Integer
destinationName	Name of destination of route	String
stopId	Bus stop Id	Integer
dataFrameRef	Date of Journey	String
visitNumber	Sequence of visiting the Stop	Integer
aimedArrivalTime	Aimed time of Arrival of Bus	Timestamp
expectedArrivalTime	Expected time of Arrival of Bus	Timestamp
latitude	Latitude of Bus Stop	String
longitude	Longitude of Bus Stop	String
location	Name of Bus Stop	String
address	Name of Area of Bus Stop	String
recordedAtTime	Time at which data was recorded	Timestamp

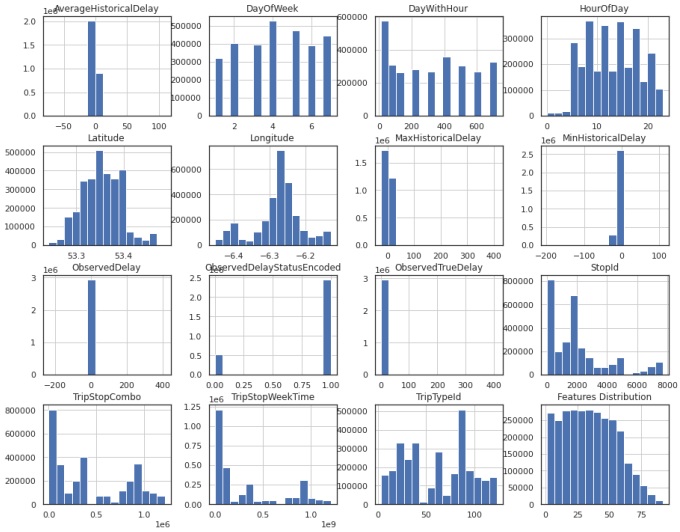

Fig. 4. Distribution of fields

attributes whereas, some are partially processed data, for example, DistanceInMeters represents the haversine distance covered by the bus from the first stop, while some other attributes are the aggregates of prior trip delay experiences formed from data processing, like the HistoricalAverageDelay, which represents the average delay observed, details of which can be found in Section IV.

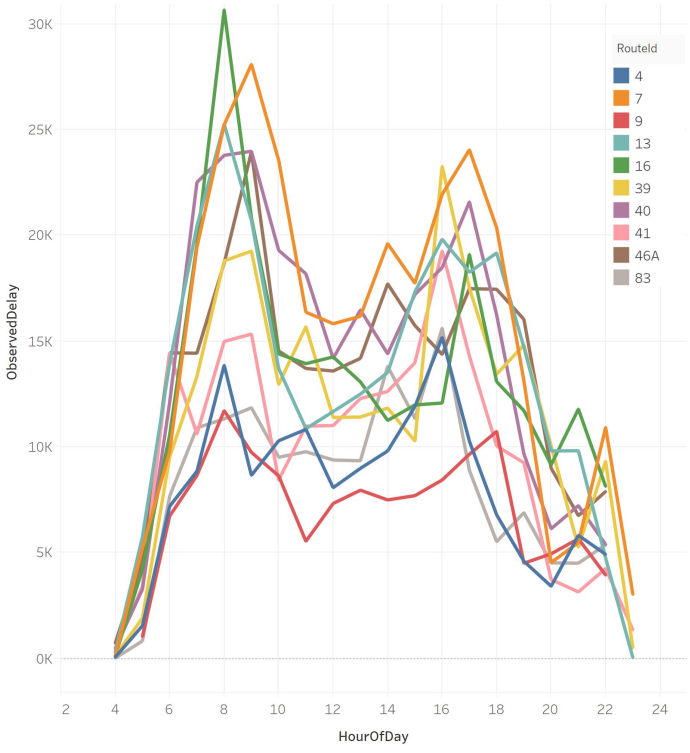

Fig. 5. Delay across a day for different routes

IV. Data PreProcessing

The preprocessing was primarily performed in Google BigQuery due to the data size. The data consisted of significant quantities of total missing data as transit feed of certain bus stops were missing even though they were scheduled by the operator. This, on manual inspection of the route data, was discovered to be due to skipping of stops and in some cases, entire trips, due to lesser passengers owing to the COVID 19 pandemic. The missing data was not imputed as such delay instances where supplemented by other buses operating in same bus stops. Other important missing data cases encountered were due to data issues in feed and data collection pipeline failures and these were handled by data filtering.

Fig. 6. Correlation fields before Feature engineering

The cleaned raw data consists of many identifiers such as routeId, tripId, destinationId, etc which could lead to very poor performance during prediction and classification, and was therefore reduced to to improve correlation significantly and later derive meaningful fields which could be used for the experiments. Part of the data was used to produce more useful fields, for example, observed delay used to identify the trip entries that are Early, OnTime and Late which could serve as a good predictor variable in classification. For this we filtered the outliers using mean and standard deviation method, and binned them based on standard deviations from mean (Fig. 2). Another field would be the hour of the day, previous research [4] has shown that binning the hour based on the time of the day like Early morning, Late morning etc., can reduce

granularity and improve prediction as the traffic pattern would be similar in these hours of the day.

Another important field, would be the distance covered by the bus at a position in the journey, this is calculated by using haversine formula [9] from the origin as bus completes each milestone in the trip. Meanwhile, the aggregate fields are a group of fields that represent the historical behaviour of the bus, relative to the bus stop, route and time. One of the most important fields is the average historical delay, that is, the average delay experienced by the bus in a particular stop in past, excluding outliers. Finally to make the computations easier, label encoding and data transformations like normalisation were also performed on the relevant data attributes. The resultant data and its correlation can be seen in correlation matrix in Fig. 7.

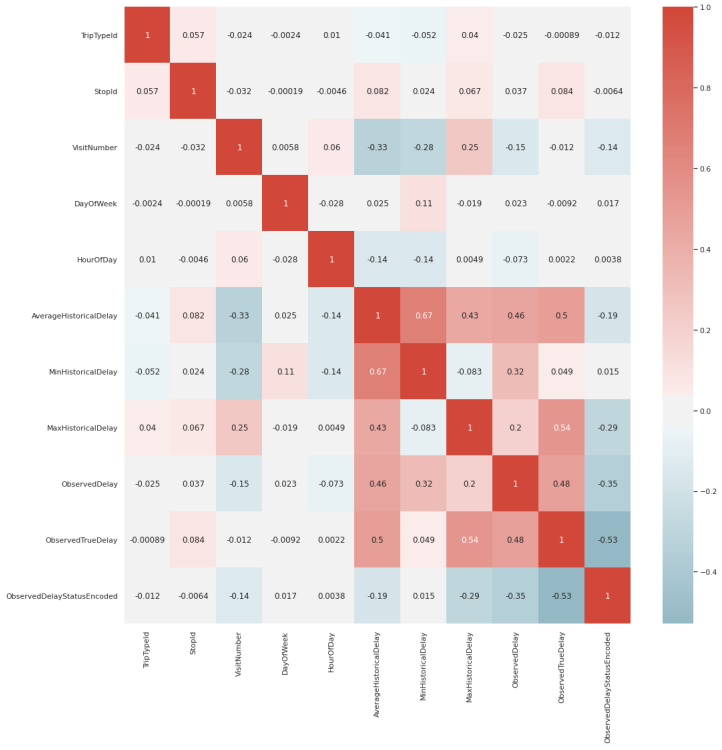

Fig. 7. Correlation fields after Feature engineering

V. Experiments & Evaluation

The cleaned and feature engineered data will be used to perform two set of experiments these are firstly delay prediction and estimation which tries to predict delay time for the trip events. Secondly, a clustering and classification where using the delay time and other trip attributes similar delays would be clustered and labelled and a classifier would be trained to predict the label for new trip event.

A. *Experiment I: Delay Prediction & Estimation*

*a) **Methodology**:* In this experiment, we try to predict whether a scheduled visit to a bus stop might be delayed and if delayed, by how many minutes. Here we used supervised machine learning algorithms to train on the transit feed data which has delay time and then tried to predict the delay. As a part of this research we explored three main approaches - firstly we investigated the performance of various regression algorithms in predicting the delay time. Secondly, we used ANNs to predict the same and compared the neural network model to examine how it scaled up against it's statistical counterparts. Then we used an ensemble approach for deducing the delay time. In this process, we split the prediction model in to two steps - first we tried to predict if a schedule will possibly be delayed, and secondly, we estimate the probable number of minutes of the delay. Finally, we built a machine learning model in Cloud using GCP, using the same data and examined how accurately it predicts the delay time. We also compare and contrast the performance of the models against each other. Our goal is to get the best possible prediction so that it can be fed from our results back to the original tables for the next experiment and to evaluate how each models perform against each other.

TABLE III
Fields and Field Types used for Prediction

Field	Description	Field Type
TripTypeId	Combination of Route Id, Direction of Route and Destination Id	int8
StopId	Bus Stop Id	int16
TripStopCombo	Trip and Stop Id encoded	int32
VisitNumber	Sequence of visiting the Stop	int8
DayOfWeek	Day Of Week	int8
HourOfDay	Hour Of Day	int8
DayWithHour	Day and hour combined	int16
TripStopWeekTime	Trip weekly time line	int32
Latitude	Latitude of Stop	float64
Longitude	Longitude of Stop	float64
AverageHistoricalDelay	Average delay observations	float64
MinHistoricalDelay	Lowest delay encountered	int16
MaxHistoricalDelay	Highest delay encountered	int16
ObservedDelay	Calculated delay	int16
ObservedTrueDelay	Actual delay	int16
ObservedDelay StatusEncoded	Delay status	int8

Evaluation of the regression models in this sections are primarily done using the following metrics - R squared (R^2), also known as the coefficient of determination, which is the square of the Pearson correlation coefficient between the labels and predicted values. This ranges between zero and one, where a higher value indicates a higher- quality model. Secondly, the root-mean-square error (RMSE) deviation which is a frequently used measure of the differences between the values predicted by a model or an estimator and the values observed. This ranges from zero to infinity, where a lower value indicates a higher-quality model. Finally, by the mean absolute error (MAE) which is the average absolute difference between the labels and the predicted values. This ranges from zero to infinity, where a lower value indicates a higher-quality model.

*b) **Regression Approach**:* In this approach we experimented various regression models and tuned their parameters

TABLE IV
REGRESSION MODELS AND SCORES

Model	Model Score	MAE Score	RMSE Score	R^2 Score
Linear Regression	0.22	2.86	42.36	0.22
Random Forest Regressor	0.22	2.86	42.54	0.22
XGBoost Regressor	0.23	2.71	42.56	0.23
SVR Regressor	0.21	2.97	45.56	0.21

so as to get the best possible prediction with feature engineered data. We experimented with several models including XGB Regressor, Random Forest Regressor and SVR models. XGBoost is an implementation of the gradient boosted decision trees, designed for speed and performance, whereas a random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset, and then uses averaging to improve the predictive accuracy and control over-fitting. SVR is the regression version of the SVM, where the fundamentals are to minimise error, individualising the hyper plane which maximises the margin, keeping in mind that part of the error is tolerated. The accuracy of these prediction models are evaluated by R^2, RMSE, MAE. Hyper parameters were tuned using grid search in relevant cases to find the best performing parameters.

The results of this experiment are shown in the table (Table IV). From this it is concluded that the XGBoost Regressor model performs the best in comparison with other regression models with R^2 score of 0.23 and the worst performing model is SVR Regressor with R^2 score of 0.21. Even though the margins are quite thin, it could be said that with the feature engineered data, the models performed equally well. We experimented with various other hyper parameters and the variations were found to be minor, but overall the results tend to oscillate around these numbers and the best results were considered.

*c) **ANN Approach**:* In this experiment we built a neural network model and tuned its hyper parameters to get the best possible prediction with the feature engineered data. We experimented using varying hidden layers, activation functions and kernel state but was unable improve the performance of the ANN beyond a certain point. The best result was obtained for the ANN with an MSE of 57.08. It consisted of 5 hidden layers, a batch size of 100 and trained for 100 epocs, which is very poor when compared to the performance of the other models. Though we tried increasing the depth and width of the hidden layer, it was not feasible beyond an extend due to the limitation on the compute time available.

*d) **Ensemble Approach**:* In this approach we created an ensemble model consisting of two layers of models. (Fig. 8) The first layer is a classifier that is trained on the input to predict if a given trip schedule point is likely to get delayed or not. If the schedule is not likely to be delayed, we do not need to calculate the delay estimation for this point. On the other hand, if the classifier predicts the schedule is likely to get delayed, then move on to the next level of this ensemble model, which is the predictor model based on a regressor or ANN similar to the models used in the previous experiment.

TABLE V
ENSEMBLE FIRST STAGE: CLASSIFIER MODELS & SCORES

Model	Accuracy	Precision	Recall	f1 Score
Naive Bayes	0.82	0.68	0.82	0.75
Decision Tree	0.835	0.81	0.83	0.78
Random Forest	0.83	0.84	0.53	0.77
ANN Binary	0.83	0.82	0.83	0.78

TABLE VI
ENSEMBLE SECOND STAGE:PREDICTION MODELS AND SCORES

Model	Model Score	MAE Score	RMSE Score	R^2 Score
SVR Regressor	0.57	2.16	17.02	0.57
Random Forest Regressor	0.57	2.15	17.02	0.57
XG Boost Regressor	0.59	2.16	17.57	0.59

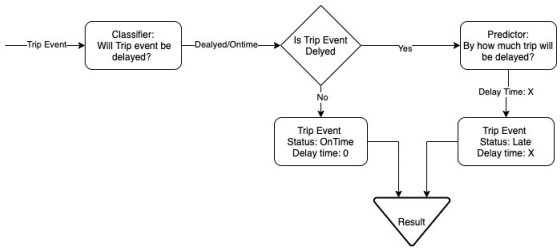

Fig. 8. Ensemble Model

While building the first layer of the ensemble model, we decided to compare classification models like the Native Bayes, Random Forest and ANN to determine the better performing classifier model. The Naive Bayes model is one of the basic supervised learning algorithms that is based on applying the Bayes' theorem with the naive assumption of conditional independence between every pair of features, given the value of the class variable. Whereas a random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and also controls over-fitting. ANN is basically a binary classifier ANN network using reu activation in the first section of the layers and sigoid activation towards the end. The results are fine tuned by splitting through the middle before evaluating.

Results of the first stage classification are tabulated in table (Table V). From the table it is clear that all the models perform similarly and the best performer is the decision tree & ANN. While considering the training time, decision tree model would best choice.

For the second stage we considered same set of predictors from experiments discussed earlier which are Random Forest, SVR and XGBoost. We omitted ANN from the set of the models used because of poor performance. The data used for training is a reduced set which only consist of cases which show delay while onTime cases have been filtered out from data frame.

From the table (Table VI) it can be deduced that the all the regressors performed equally relative to each other with the best model being the XGBoost model. Tying both of the results from first and second stage together it can be concluded that the ensemble models were able to predict the delay and delay time with nearly 60% accuracy.

*e) **Cloud Based ML Approach**:* In this approach, we built a learning model based on BiqQuery and ML framework provided by Google Cloud Platform (GCP). Google provides APIs in GCP that enable a connection to port the data stored in BigQuery to their cloud based ML framework - Cloud AutoML. Using this framework we designed ML models based on the selected data, its predictor features, target feature and other inputs like evaluation method, optimisation method, training time etc. Once these specifications are determined, AutoML tries various ML models and selects the best model for the data and parameters specified. It then trains the model for a given time frame or till no longer improvements can be reached and returns the trained model and score.

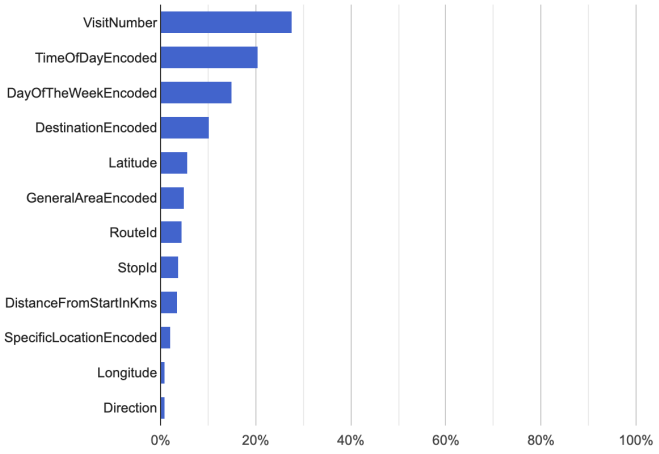

Fig. 9. Feature Importance

We tried varying the time interval of training the model without changing the data set specifications and analysed the quality of the predictions of models. In both cases AutoML chose an ANN with 64 layers and trained them using the data. Results of the prediction is shown in the table (Table VII). From the results, one can see that increasing the training time can impact the performance model to a certain point, beyond which gains seems the be diminishing rapidly. Though these models are not as good as the ensemble models, they perform better than the regression based models and ANN. The importance of each feature in terms of prediction quality is recorded in Fig. 9. It can be seen that the VisitNumber is the most important feature at about 25% feature importance, followed by TimeOfDayEncoded, DayOfTheWeekEncoded and DestinationEncoded all between 20% and 10%. Latitude and Direction features have negligible feature importance.

*f) **Results**:* Comparison of the approaches can be found in the table (Table VIII). From the results, one can very

TABLE VII
CLOUD ML: PREDICTION SCORES

Training Time	MAE Score	RMSE Score	MAPE Score	R2 Score
1 Hour	2.378	4.289	1.0E+100%	0.279
2 Hour	2.264	4.072	1.0E+100%	0.328
3 Hour	2.265	4.083	1.0E+100%	0.329

TABLE VIII
COMPARISON: BEST MODELS AND SCORES

Model Type	Model Name	MAE Score	RMSE Score	R2 Score
Regression	XGBoost	2.71	42.56	0.23
ANN	Sequential	3.093	57.02	0.04
Ensemble	Decision+XGBoost	2.16	17.56	0.59
Cloud ML	NN	2.265	4.083	0.329

easily discern that the ensemble model outperforms all other approaches by a wide margin. Whereas the ANN based model, which scored the lowest, might attribute its lower performance to the duration of training or a depth of the network. Since ensemble performed the best in this experiment we use this model to predict the delay status and delay time for the whole dataset to be used for following experiments.

B. *Experiment II:Delay clustering and Classification*

*a) **Methodology**:* The second experiment attempts to classify the delays, using unsupervised methods, into causes using the attributes to find underlying patterns using clustering. Three different clustering algorithms applied are - Kmeans, DBScan and OPTICS, and their results are compared using the silhouette scores. Once the data is clustered, machine learning models are used to classify delays based on causes and then the classification can be applied on real time feed. Multinomial logistic regression, Decision Tree and XGBoost algorithms were the machine learning models applied. Artificial Neural Network using the Fastai library [15] was also applied. The models were compared using accuracy, precision, recall and f1 score metrics.

*b) **Clustering**:* Unsupervised learning was used to find groups of delays which could be attributed to a cause and to find inherent patterns in the delays. The performance of the clustering models was compared using the silhouette score. The silhouette coefficient can be calculated by the equation (Eq. 1), where 'a' is the intra-cluster distance and 'b' is the inter-cluster distance, for each data point. It can take a value from -1 to +1 where a high value indicates that the clusters that are well apart from each other and clearly distinguished whereas low values have clusters assigned incorrectly. The distance metric used is the euclidean distance.

$$SilhouetteScore = \frac{b-a}{max(a,b)} \tag{1}$$

K-means is one of the most simple and quick partion-based algorithm used for unsupervised clustering [13]. The idea

behind the algorithm is to obtain groups with high similarity within groups and do not overlap with other groups. The optimal number of clusters, which is also the number of centroids k, can be obtained using the elbow method. The elbow method runs the K-Means algorithm on the dataset for a range of values and then computes an average score for all clusters using the inertia method.

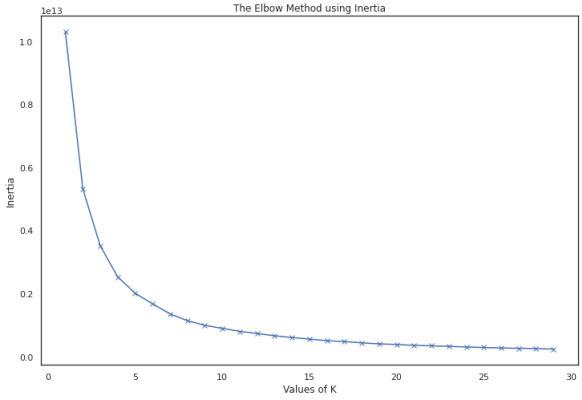

Fig. 10. Elbow Method

From (Fig. 10) it can be seen that the optimal number of clusters is approximately at k=10, after which the inertia starts to decrease in a linear fashion. Once the optimal number of clusters is determined, the K-Means algorithm is applied on the dataset and the resulting cluster label is added as a new column to the dataset. To represent the cluster data in a graph, Principal Component Analysis (PCA) is performed on the dataset to reduce the number of dimensions to two (i.e. x and y axes).

Another clustering algorithm used was the DBSCAN, also known as Density-based Spatial Clustering of Applications with Noise, which is an extremely powerful clustering algorithm that regards clusters as dense areas of data points surrounded by less dense data points [14]. The parameters that regulate this algorithm are 'eps' and 'min_samples'. 'eps' or epsilon is the radius around a point considered neighbourhood , and was set to a value of 0.5, and 'min_samples' is the minimum number of neighbours within the 'eps' radius which was set to 5.

The OPTICS or Ordering points to identify the clustering structure clustering, was also evaluated and is similar to DBScan but overcomes one of the major weaknesses of DBScan i.e. detecting clusters in data of varying density. This is done by linearly ordering the data points such that neighbouring points are closest. The OPTICS algorithm are regulated by 'min_samples' and 'min_cluster_size' which were set to 10 and 0.05 respectively.

The delays were derived from the results of previous experiment on this dataset. The delays classified as 'Late' is used to group the records into clusters, based on historical patterns or behaviours, and label the clusters with causes. The records classified 'Late' were pulled from the BigQuery Table into a Google Collaboratory Notebook. (Fig. 4) where the later phases were implemeted. The data consisted about 2.7 hundred thousand records.

TABLE IX
FIELDS AND FIELD TYPES USED FOR CLUSTERING

Field	Description	Type
ArrivalDelayPrediction	Number of Minutes Bus is Late	int8
DistanceInMeters	Distance from the start of Route	float64
TripId	Combination of RouteId, Direction of Route and DestinationId	int16
Hour_Day	Combination of Hour of Day and Day of Week	int8
Latitude	Latitude of Stop	float64
Longitude	Longitude of Stop	float64

To avoid the clustering algorithms getting fixated on a particular field, minor feature engineering was performed. The RouteId field, which was of the type string, and was converted to int using encoding. The combination of RouteId, Direction and DestinationId was encoded to TripId to uniquely identify trips. The TimeOfDay and DayOfWeek was encoded as Hour_Day to specify the hour of a day. Once feature engineering was performed, unnecessary and redundant fields such as RouteId, Direction, DestinationId, StopId among many others were dropped (Table IX). The remaining fields were downcasted to improve performance and computational costs. Once the data was preprocessed, the clustering algorithms were applied.

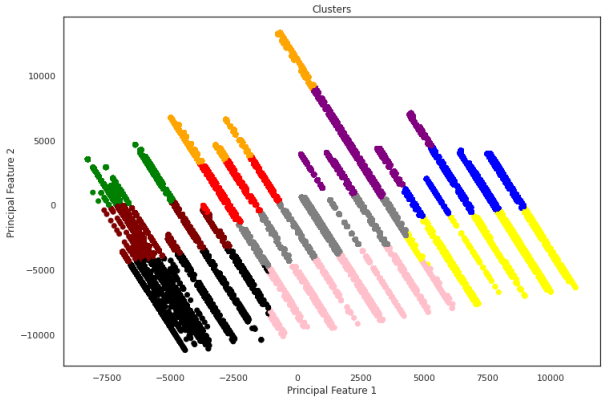

Fig. 11. K-Means Clusters

K-means performed best with a silhouette score of 0.44. The optimal number of clusters was found to be k=10 using the elbow method and was therefore clustered into 10 clusters (Fig. 11). The DBScan algorithm performed adequately with a silhouette score of 0.305 and the data was clustered into 14 clusters with 44 noise points (marked in black (Fig. 12)). The OPTICS algorithm performed the worst with a silhouette score of - 0.15. The DBScan and OPTICS algorithm were both computationally expensive and time-consuming and therefore a subset of three routes were used for the implementation.

*c) **Classification**:* Once the data was clustered into causes, machine learning models were applied and compared to classify the delays of incoming records. The data was split into 80% training and 20% testing sets using the train test

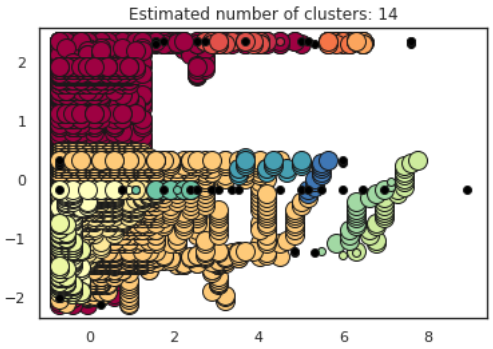

Fig. 12. DBScan Clusters

split function which was stratified based on the clusters to ensure balanced records. The models were implemented and then compared using accuracy (Eq. 2), precision (Eq. 3), recall (Eq. 4) and f1 score (Eq. 5) metrics [16].

$$Accuracy = \frac{TruePositive + TrueNegative}{Total} \tag{2}$$

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4}$$

Logistic regression is one of the basic supervised machine learning algorithms that can be used for classification and was the first model to be applied. Mutlinomial classification was used as the data consists of more than two classes. The 'max_iter' parameter was set to 4000 as the algorithm did not converge at the default maximum 100 iterations. The solver used was 'sag' which is faster for large datasets. The decision tree classifier is another popular classifier that uses a tree like model to mimic human decisions. Since the consists of smaller partitions, information gain is used as the criterion to split the decision trees. The maximum depth of the tree was set to 10. Extreme Gradient Boosted decision tree or XGBoost was also implemented as this algorithm is popular for speed and performance. The maximum depth of the trees was 10 and "multi:softmax" objective was used since the data consists of multiple classes. GridSearch was performed to find the best learning rate and 0.1 was found to be the best learning rate. Artificial neural network using the Fastai library was the final classification model applied. All the fields are categorical and tabular. The data was converted to the data bunch format for Fastai input. Once the data bunch was created, it was input into the learner for training. The neural network consisted of three fully connected hidden layers with 1000, 500 and 15 nodes respectively. The network was trained for 20 epochs.

The multiclass logistic regression model performed poorly with the accuracy of 32% whereas decision tree performed much better with an accuracy of 70%. XGBoost achieved the best results with an accuracy of 97% meanwhile the ANN performed almost comparably good with an accuracy of 92%. The precision, recall and f1 score metrics also follow the same pattern.

$$f1_Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

*d) **Results**:* Of the three clustering algorithms applied, the OPTICS algorithm performed most unfavourably, with a silouette score of -0.15. This implies that the clustering using OPTICS result in incorrect clusters. The DBSCAN algorithm performed fairly well with a silhouette score of 0.305 and produced 14 clusters.The K-means algorithm performed the best, with a silhouette score of 0.44 (Table X) and was finally used to cluster the data into causes for delays.

TABLE X
SILHOUETTE SCORES OF CLUSTERING ALGORITHMS

Method	Silhouette Score
KMeans	0.44
DBScan	0.305
Optics	-0.15

Once the data was clustered using the K-Means algorithm, different classification models were applied and compared for further classification of future real-time incoming records to most probable causes in case of a predicted delay. XGBoost classifier model achieved the best accuracy at 97% while logistic regression obtained the least accuracy at 32%. The Decision tree classifier performed better than Logistic Regression but not as good as the XGBoost model. ANN using Fastai obtained commendable results with an accuracy of 92% (Table XI).

TABLE XI
CLASSIFICATION ALGORITHMS

Model	Accuracy	Precision	Recall	f1 score
Logistic Regression	0.32	0.29	0.30	0.27
Decision Tree	0.70	0.72	0.68	0.67
XGBoost	0.97	0.97	0.97	0.97
ANN	0.92	0.91	0.90	0.90

VI. CONCLUSION & FUTURE WORK

This paper predicts the delays and classifies the delays to similar cause groups using data from Dublin Bus Network Transit feed. To achieve this, we built an ETL pipeline for the collection of real time transit feeds data. This data was streamed in batches to BigQuery, to handle the volume and velocity of the streaming data. The pooled data was then cleaned and feature engineered using Bigquery to prepare the data for subsequent prediction and classification experiments. In the prediction phase, we experimented with various types of supervised learning methods to derive delay time and found that an ensemble approach with decision tree as first layer and XGBoost regressor as second layer would give

better predictions than a standalone regression model or neural network model by a wide margin. Then we used the derived delays with other other relevant parameters to group the data into clusters with the help of unsupervised clustering models based on similarities in delay. These delay clusters are then labelled with causes. Also we found that K Means clustering algorithm provided the best quality of clusters compared to other models. Once the training data is labelled, we used this data to train classification algorithms so that it can classify new incoming trip records with predicted delay to a probable cause based on the cause labels.

Due to time, data availability and scope of study, the data we used was only from one source that is the transit feeds of Dublin bus. Whereas some researches indicate that environmental factors like rainfall, temperature, humidity, etc has some degree of influence on the traffic pattern. Similarly information on spontaneous traffic incidents like accidents, road closures, etc also have unpredictable influence in the traffic. Including such information to this study would have improved the final models capability to some extend but due unavailability of such data owing copyright restrictions we where unable include them in our study. Another area where the results of the study could be improved would be to build a cloud based ensemble model. Yet another area is to find real world causes and prepare pre-trained kernel for each cause pattern, this way, finding cause of the delay would be easier, accurate and reusable.

REFERENCES

[1] "About bus travel," Apr 2020. [Online]. Available: https://www.transportforireland.ie/getting-around/by-bus/about-bus-travel/

[2] T. for Ireland, "Dublinbus." [Online]. Available: http://rtpi.dublinbus.ie/DublinBusRTPIService.asmx

[3] E. Bisong, "Google bigquery," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, pp. 485–517.

[4] A. Comi, A. Nuzzolo, S. Brinchi, and R. Verghini, "Bus travel time variability: some experimental evidences," *Transportation Research Procedia*, vol. 27, pp. 101–108, 2017.

[5] M. Chen, X. Liu, J. Xia, and S. I. Chien, "A dynamic bus-arrival time prediction model based on apc data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 5, pp. 364–376, 2004.

[6] D. Liu, J. Sun, and S. Wang, "Bustime: Which is the right prediction model for my bus arrival time?" in *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. IEEE, 2020, pp. 180–185.

[7] J. Li, J. Gao, Y. Yang, and H. Wei, "Bus arrival time prediction based on mixed model," *China Communications*, vol. 14, no. 5, pp. 38–47, 2017.

[8] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[9] T. Cristóbal, G. Padrón, A. Quesada-Arencibia, F. Alayón, G. de Blasio, and C. R. García, "Bus travel time prediction model based on profile similarity," *Sensors*, vol. 19, no. 13, p. 2869, 2019.

[10] Z. Junyou, W. Fanyu, and W. Shufeng, "Application of support vector machine in bus travel time prediction," *International Journal of Systems Engineering*, vol. 2, no. 1, pp. 21–25, 2018.

[11] W.-C. Lee, W. Si, L.-J. Chen, and M. C. Chen, "Http: a new framework for bus travel time prediction based on historical trajectories," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, 2012, pp. 279–288.

[12] A. Gal, A. Mandelbaum, F. Schnitzler, A. Senderovich, and M. Weidlich, "Traveling time prediction in scheduled transportation with journey segments," *Information Systems*, vol. 64, pp. 266–280, 2017.

[13] R. A. Kadir, Y. Shima, R. Sulaiman, and F. Ali, "Clustering of public transport operation using k-means," in *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*. IEEE, 2018, pp. 427–432.

[14] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, 2006, pp. 281–286.

[15] J. Howard *et al.*, "fastai," https://github.com/fastai/fastai, 2018.

[16] N. Chinchor, "MUC-4 evaluation metrics," in *Fourth Message Uunderstanding Conference (MUC-4): Proceedings of a Conference Held in McLean, Virginia, June 16-18, 1992*, 1992. [Online]. Available: https://www.aclweb.org/anthology/M92-1002