REAL-TIME ANALYTICS FOR URBAN ROAD TRANSPORT

SUMIT KUMAR MUND

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Master of Science by Research

The University of Huddersfield

October 2016

Abstract

Urban road traffic congestion has been a constant problem both in UK and worldwide. Urban road transport authorities collect data from different sources. These data can be effectively utilised with an objective to minimize congestion and its impact. One of the ways for the same can be to find possible congestion in different routes beforehand and then plan accordingly either to reduce the effect or to avoid it entirely.

So this project aims to make effective use of existing data to predict journey time for near future e.g. 15/30/60 minutes ahead for different routes within the urban road traffic network. It also produced a working prototype for the journey time prediction with necessary data visualisations.

A complete data centric approach has been adopted to solve the problem of prediction by building a predictive model using machine learning algorithms with traffic volumes at different points as predictor and journey time for near future as the target. Given the nature and volume of the data, a big data platform (Apache Spark) was chosen as the analytics platform and the work also proposes a high level technical architecture for the end to end solution.

The results for journey time prediction for near future are quite encouraging with a consistency for different routes under the area of consideration.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Urban road traffic congestion in UK has been a constant problem for a number of decades, and with the recent rise of population and rapid growth in economy, the problem is becoming worse. According to a report [37] by INRIX, a leading provider of real-time traffic information and connected driving services, in the year 2014, the UK became the fifth most congested country in the Europe where a commuter wastes around 30 hours on the road annually. London was Europe's most congested city in the year, with drivers spending 96 hours in traffic followed by Grater Manchester with drivers wasting 52 hours.

Such a bad condition of road traffic is costing the economy a lot. In another report [38] by the same agency, it estimates the gridlock will be costing the UK economy around $480 billion cumulatively between years 2013 to 2030. At the individual level, traffic congestion costed drivers on average $1,740 in 2014 alone and if unchecked, this number is expected to grow more than 60% to $2,902 annually by 2030. There is also a cost from damage to the environment due to the increase in road traffic.

In relation to managing the traffic, the Highways Authority [42] has the responsibilities for motorways and trunk roads and the local authorities [43] has management roles for safe and efficient conduct of urban traffic.

Problem Overview

Urban road transport authorities collect data from different kinds of sensors like SCOOT [29] devices, Bluetooth devices etc. Currently most of the collected data are used only to generate static reports. At the same time the authorities are actively looking forward to solve the following problems:

- What is the traffic condition across the network at any given point in time?
- Can traffic incidents and abnormalities be predicted before they happen?
- Can the existing data sources be utilised to make effective traffic plans?

At the moment, a common way to monitor or to have a macroscopic view of a city's traffic is by observing images captured through CCTVs in real-time. This is a manual process and not adequate to provide the information required by traffic controllers to take necessary decisions.

There are different kinds of unexpected incidents that might occur on an urban traffic network e.g. an accident, congestion or traffic delay. If such an unwanted incident occurs, then the authority takes necessary action to mitigate the problem. At the moment, it is mostly reactive in nature – an action is taken only after an incident occurs. But, it will be quite efficient if an authority can predict incidents before they occur then the incident can be managed very efficiently and on certain occasions it can also be avoided. So, authorities can act proactively to manage an incident.

Traffic authorities implement traffic plans to manage the network efficiently and conveniently for the commuters. One example of such plan is that they control different traffic signal times so that there is efficient, effective and safe flow of traffic on the roads. Currently, they run static plans which are predefined. Given that they have now lots of data collected from different sources, transport authorities are interested in making use of the data to come up with more efficient plans. Also, there is an increasing interest in making use of data to build dynamic plan based on changing conditions of a traffic network.

Transport for Greater Manchester or TfGM [39] is the authority responsible for all the urban traffic management for Greater Manchester area. TfGM was kind enough to allow access to one of its information system called C2 [44] and permitted us to use the data in the C2 system for the purpose of this project.

TfGM has installed Bluetooth sensors across the roads transport network. When a vehicle passes through a sensor, it makes a note of the unique identity of Bluetooth device enabled inside the vehicle along with the timestamp. Through these sensors, the traffic volume data at any sensor point or site (as per the convention of C2 system) is available up to every minute. The C2 software system takes this data and computes journey time between any two sites as the start and end of a route. The system also provides a web interface for historical reporting and allows data to be downloaded in excel or csv formats.

Among the different problem areas identified before, the following one has been identified for further investigation:

- Can traffic incidents and abnormalities be predicted before they happen?

There are many abnormalities that can take place, like congestions, accidents, excess emission of pollutants like Nitrogen Oxide etc. For this project, the datasets (including historic ones) were available from the C2 system which TfGM allowed us access to use the datasets for the research.

Motivation

Both traffic volume data and estimated journey time data for routes can used with a data centric approach to solve one of the concerns of the traffic authority: to predict if something is about to go wrong on the network. The traffic congestion or delay in a journey is one of the most important and most frequent undesired happenings on UK road networks including that of Grater Manchester.

Traffic congestion [40] can be defined in terms of the difference between users' expectations of the road network and how it actually performs e.g. suppose it usually takes 5 minutes to commute by car on a particular route and if it takes 7 minutes to commute on that instance then the additional 2 minutes will be contributed to congestion or traffic delay.

So if the journey time for a route can be predicted, then it would be easy to predict the traffic congestion based on the time of the day. This project undertakes this objective to predict the journey time for the short future – 15/30/60 minutes ahead given the historic data and the live data for traffic volume at the time of consideration.

The output of this work can be very useful for the transport authorities. They might be able to take some action if they get to know beforehand that it is going to be congested on a certain route. Also it would help them find out how the entire network is going to do in near future in terms of journey time and congestion. If this project is implemented in production to predict the journey time in real time, the result can be utilised by some other information system in run time.

Real time prediction of journey time is an area of interest in academia. For researchers related to transport, statistics and artificial intelligence, this work may provide basis for further exploration and extension.

Solution Overview

The project, Real-time Analytics for Urban Road Transport aims to achieve following:

- Making effective use of existing data to predict journey time for a route beforehand e.g. before 15 minutes.

- To produce a working prototype for the above prediction and visualisation.

A complete data centric approach has been adopted to solve the problem of predicting the journey time. A network has many routes, but for the scope of this project, a few routes,

including simple and complex ones have been selected for experimenting with the prediction. It was investigated to come up with a model for prediction of journey time which would make use of collected historic data for traffic volumes at different sites.

The following figure (Figure 1) shows the traffic volumes at different sites (point of sensors) at a particular time. The height of the bars depends of the volumes of the traffic, more the height if higher the traffic volume.

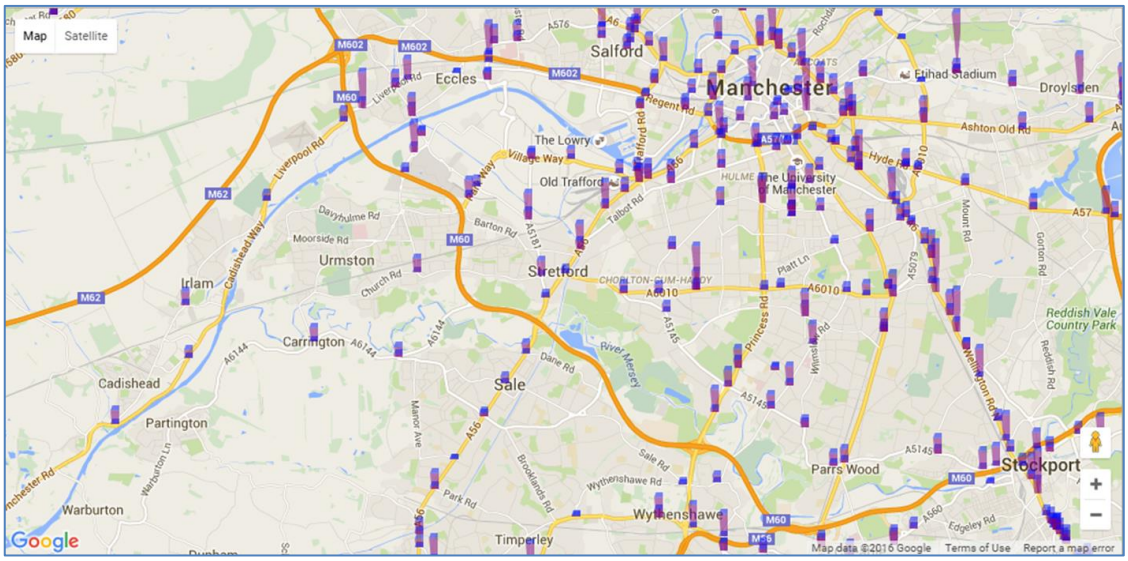

Figure 1 : Traffic volumes at different sites

As part of the project, different approaches were investigated and finally got settled on machine learning based model. In supervised machine learning, the model learns from the examples or the historic data by finding right pattern. It finds the pattern after being trained with the examples. Then the trained model can be used to make prediction given a new dataset. The following diagram (*Figure* 2) explains supervised machine learning visually.

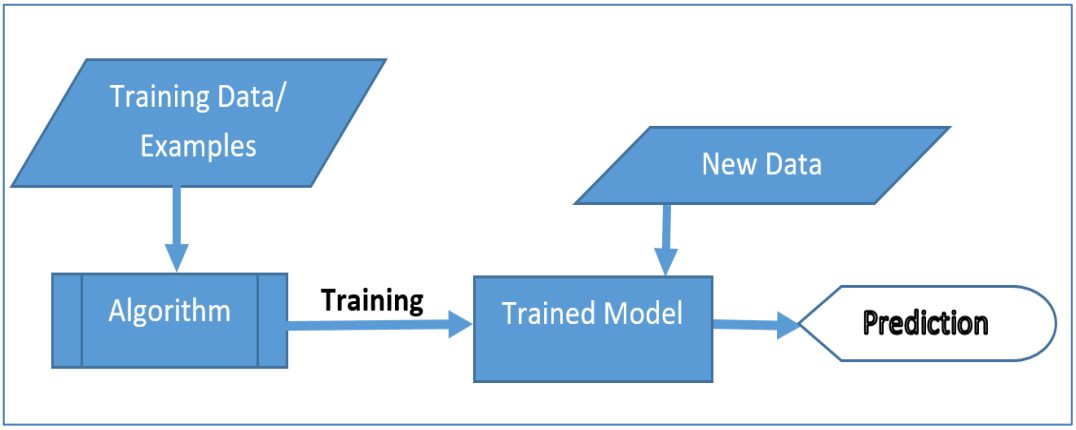

Figure 2 : Supervised Machine Learning

There are different algorithms for supervised machine learning. Decision tree algorithm [57] was selected to build the model. As huge data (traffic volumes for around 100 sites for nearly a year) was processed for training the model. This needed a big data platform as the huge dataset is not possible to be processed in a single machine setting. The Spark [85] implementation was used as the analytics platform for all the data processing.

The entire dataset considered was divided into two sets with 7:3 ratios. The former dataset was used to build or train the model and the rest was used for testing the model and evaluating the result. The final figures were obtained by running the experiments 3 times in a row and by taking the average of the results. The two metrics evaluated are MAE and MAPE. Mean Absolute Error (MAE) [65] is the average of the absolute difference between actual journey time and predicted journey time in seconds. Mean Absolute Percentage Error (MAPE) [66] is the average of the absolute difference between actual journey time and predicted journey time divided by the actual journey time. While MAE depends on the unit of journey time like minutes or seconds, MAPE does not.

Table 1 : Evaluation Result for one of the routes

Prediction Ahead	MAE (seconds)	MAPE
15 min	15	9%
30 min	16	10%
60 min	22	14%

Dissertation Plan

The rest of the dissertation is organized as follows:

Chapter 2 covers the detailed problem analysis and the literature review.

Chapter 3 discusses the solution design in detail and the approach adopted.

Chapter 4 provides a brief of algorithm for scale and how it is designed and implemented.

Chapter 5 discusses the prediction results and explains the evaluation metrics.

The last chapter brings the conclusion and also discusses the next steps.

Chapter 2 Background and Literature Review

Delay in a journey may result from many reasons [70] apart from the increase in traffic flow like events, incidents, weather conditions etc. Few examples are as below:

- An event like a football match organised in the city can cause unexpected flow of traffic which can result in traffic jam and delay.
- An incident like an accident [100] may cause road blockage and result in delay.
- Bad weather like snow fall or good weather like a sunny summer day may impact the traffic flow and can result in traffic delay.

So one ways to predict traffic delay may be to map all such events, incidents and conditions along with traffic flow and come up a solution to predict a delay in journey time as a function of all the enumerations. There are following problems in this approach:

- It is difficult to enumerate all such events, incidents and conditions in real time. These are also interdependent and varies in nature. For example, bad weather may cause an incident which leads to traffic delay.
- If there is a new kind of event or condition or incident which was unknown during the solution design, then it may lead to fail the solution.
- For this project, data was available from one source, C2 system of TfGM. C2 [44] has very limited data about events, incidents and conditions.

So the above approach was not considered and an alternative approach was investigated.
C2 system primarily provides following two kinds of data:

1. *Traffic Volume*: There are Bluetooth based sensors installed throughout the city which capture traffic volumes – count of vehicles passing through a sensor. The traffic volume is captured every minute. The visualisation in *Figure* 1 displays traffic volumes captured at a particular instance.
2. *Journey time*: C2 allows identifying different routes and can mark a route with two sensor points as source and destination. With some algorithms it provides estimated journey time from the source to destination. So it doesn't provide journey time for all the possible routes but the ones already identified in the system or defined by the user. And hence historical data for different journey time is only available for those identified routes. A route can be one of two types:
    - *Normal Route:* A normal route is one which has only two sites – one as its starting point and other as destination point.
    - *Compound Route:* A compound route is one where there are more than two sites on the way.

Journey Time Analysis

Following is the journey time analysis for one of the compound routes with the route id, 9800ZFG0V7IO in C2 system. This route is of length 3 miles stretching from A6042 Trinity Way / Regent Rd, Salford to A665 Great Ancoats St / Pin Mill Brow, via IRR, Manchester. It is a compound route as shown in the following figure (*Figure* 3).

Figure 3 : Route Map for route id: 9800ZFG0V7IO

The journey time (in minutes) for the route has been explored through different plots in the following figures. The histogram and pareto chart in *Figure* 4 shows that the journey time often lies in between 6 to 12 minutes. Nearly 30% of the time it also lies in between 12 to 18 minutes. As it shows in the Box & Whisker plot (*Figure* 5) as well, any journey time above 18 minutes can be considered outliers.

The following are some of the descriptive statistics of the journey time for the route:

- Mean : 12
- Median : 11
- Mode : 9
- Min : 6
- Max : 37

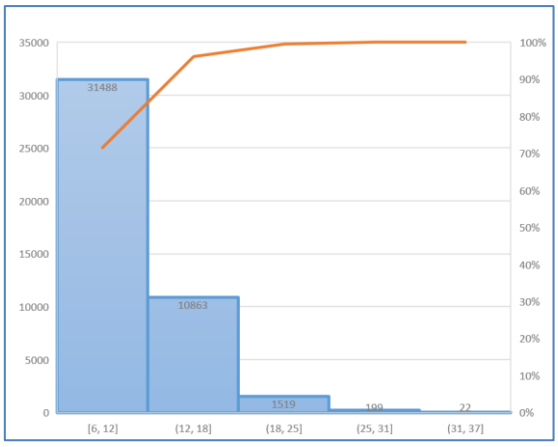

Figure 4 : Journey Time - Pareto Chart

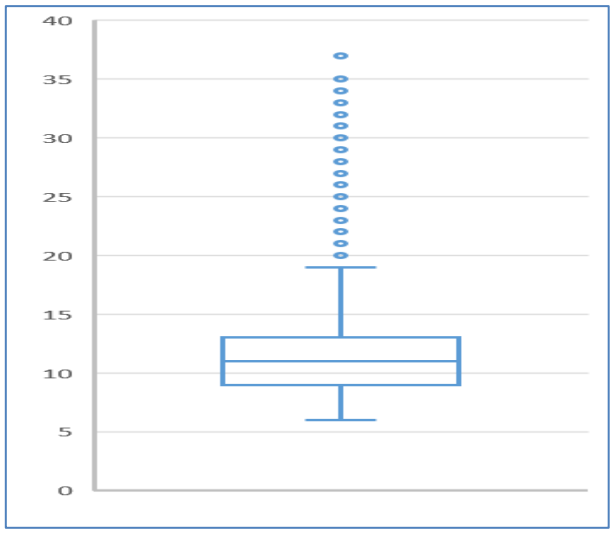

Figure 5 : Journey Time Box & Whisker plot

However, it is typical for any route that the journey time differs at different times of the day. So one average journey may not well describe the journey time for entire day. And also, the perception of congestion [40] depends on the time of the day. So what might be considered congested at 2 PM, might be normal at 8 AM. The following figure (*Figure* 6) plots average journey time, maximum journey time and minimum journey time against different time of the day with an interval of every 15minute. This shows and confirms that at any point of the day, the average journey time limits to 16 minutes. This plot also shows that at different point of time the maximum journey time are different and much above than the average and so also minimum journey time. So just considering historical averages over different point of time may not help in predicting delay in journey time.

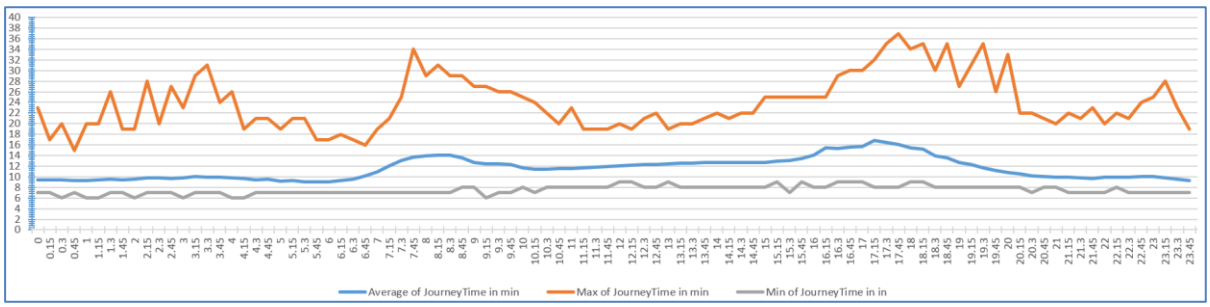

Figure 6 : Average journey time, maximum journey time and minimum journey time plotted for 15minute interval

Journey time related to a route is seasonal in nature. So these seasonal factors like time of the day, day of the week etc. might prove to be strong data points for finding journey time. However, there is a drawback in this approach. Once these seasonal factors are considered, this would have strong influence on the model to find the journey time. But here we are finding journey time with an intention of predicting delay or congestion. So these seasonal factors being strong influencer will generalise the model and would miss the delay in journey time which are corner cases, most of the time. Here is a scenario to illustration the idea: Consider there is a route usually quiet at around 8 pm on a Wednesday. But on a certain Wednesday there is a football match going on a nearby stadium and hence more traffic flow on the route and hence causing congestion or delay. If a model has been built to predict congestion with the seasonal factors like day of the week (Wednesday), time of the day (8 pm) and these would take the average case and bring shadow on the real influencer to the traffic delay (e.g. here, the sudden increase of traffic flow) and may miss to predict the

congestion or delay. So it may not make sense to include these seasonal factors while building a model to predict congestion.

Assumptions and Hypothesis

A hypothesis has been made that there exists a pattern in the traffic volumes from different sensors which correlates to near future journey time involving a route within an area covered through the sensors.

So it was decided to take a pure data-centric approach and predict the journey time of a given route from the historic data - journey time for the given route and the traffic volumes captured from all the sensors. This approach has the following advantages:

1. It does not need any other data points like events or incidents to predict journey time or traffic delay. So the model can work independent of any external factors.
2. The journey time for a given route is being predicted using all the sensor data available or the traffic volumes from the all sensors without selecting a few traffic volumes for a given route. No prior knowledge is being applied about a route with respect to the possible influencing traffic volumes at few sites. So the model can be applied to any identified route within the sensor areas given availability of historic data.
3. If certain traffic volumes are influencing the journey time of a routes more than the rest, then the model won't assume anything to begin with but can find out those influencing points as output.

Problem Definition

The problem of predicting the journey time for a given route can be defined in the following form. The predicted journey time in near future can be described as a function of current journey time and near past traffic volumes of all the sites.

$$jt_{t+15} = f\left(jt_t, v_{i,t}, v_{i,t-1}, v_{i,t-2}, \dots v_{i,t-14}\right)$$

Where,

jt_{t+15} → Journey time at time, (t +15)

jt_t → Journey time at time, t

$v_{i,t}$ → Traffic volume at i^{th} site at time, t

$v_{i,t-1}$ → Traffic volume at i^{th} site at time, t-1

…

$v_{i,t-14}$ → Traffic volume at i^{th} site at time, (t-14)

The above descries a model for predicting journey time before 15 minutes. So the interval here is 15 minutes. The assumption is to look back into the near past for traffic volume data for all the sites for equivalent time interval as here we are looking back 15 minutes' worth traffic volumes data for different sites.

So the model to predict 30 minutes ahead may be as below:

$$jt_{t+\mathbf{30}} = f\left(jt_t, v_{i,t}, v_{i,t-1}, v_{i,t-2}, \dots v_{i,t-\mathbf{29}}\right)$$

However, the duration to look back may differ to get the best result, but currently it has been the default choice. The solution will decide the right interval to look back. Traffic volume for all the sites have been considered so that the model can be generalised for all the routes in the traffic network. It is expected that the model would pick the right data points to make the prediction.

The proposed model as a solution need to solve the function defined as above.

Related Work

A recent study in Huddersfield [1] suggests that all or a section of the data coming from traffic management centres can be input into to a data mining tool leading to discover associative relations between attributes of the data. A prediction model can be formed for different incidents, for example accidents occurring at points in the network, or of patches of congestion.

The TIME project (Transport Information Monitoring Environment) [6] has focussed on urban traffic, using the city of Cambridge as an example. It has shown that traffic data can be used for a number of purposes. Firstly, archived data can be analysed statistically to understand the behaviour of traffic under a range of "normal" conditions at different times, for example in and out of school term. Secondly, periods of extreme congestion resulting from known incidents can be analysed to show the behaviour of traffic over time. Thirdly, with such analysis providing background information, real-time data can be interpreted in context to provide more reliable and accurate information to citizens.

Numerous other research papers are available on short-term traffic flow forecasting. In a classic paper [2], they suggest four models for the freeway traffic flow forecasting problem, which is defined as estimating traffic flow 15 minute into the future. The models were the historical average, time-series, neural network, and nonparametric regression. The nonparametric regression model significantly outperformed the other models. This finding also gets strengthened in a recent study [7] which proposes a k nearest neighbour nonparametric regression (KNN-NPR) forecasting methodology tested against vast quantities of real traffic volume data collected from urban signalized arterials. The results show that the KNN-NPR model is clearly superior to two parametric models, Kalman filtering and seasonal autoregressive integrated moving average (ARIMA), in terms of both prediction accuracy and the construction of the directionality of temporal state evolution without a time-delayed response. It also concludes that KNN-NPR, even though it is very simplified, is able to efficaciously capture the complex behaviour of urban signalized traffic flow.

Another article [3] presents the theoretical basis for modelling univariate traffic condition data streams as seasonal autoregressive integrated moving average processes. In a recent paper [4] it shows how to forecast urban road short-term traffic flow based on the delay and nonlinear grey model. Another study [5] reports on the application and performance of an alternative neural computing algorithm which involves 'sequential or dynamic learning' of the traffic flow process.

Overview of Statistical Learning

The main goal of statistical learning theory [76] is to provide a framework for studying the problem of inference that is of gaining knowledge, making predictions, making decisions or constructing models from a set of data.

It is the process of inductive inference that can be explained in briefly in the steps below:

1. Observe the phenomenon
2. Construct a model of that phenomenon
3. Make predictions using this model

It aims to automate this process and the goal of Learning Theory is to formalize it. In short, Statistical learning theory deals with the problem of finding a predictive function based on data. Most of the statistical learnings can be categorised either as supervised or unsupervised learning.

The supervised learning can be defined as below [77]:

Given a training set of N example input–output pairs

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

where each y_i was generated by an unknown function $y_i = f(x_i)$.
The supervised learning discovers a function, h that approximates the true function, f or $h \approx f$.

Here x and y can be any value; they need not be numbers. The function h is a hypothesis. Learning is a search through the space of possible hypotheses for one that will perform well, even on new examples beyond the training set. To measure the accuracy of a hypothesis a test set of examples are applied that are distinct from the training set. A hypothesis generalizes well if it correctly predicts the value of y for novel examples.

There are different algorithms [80] that can be used to achieve the supervised learning like Neural Networks, decision trees, random forests etc. The following table compares different algorithms.

Table 2 : Comparison of different Supervised Learning Algorithms

Comparisons	**Decision Trees**	**Random Forests & Ensembles**	**Neural Networks**
Easy to interpret	Yes	A little	No
Ease of Training	Yes	Somewhat	No
Training Speed	Fast	Slow	Slow
Prediction Speed	Fast	Moderate	Fast
Amount of Hyperparameter Tuning Needed	Less	Less	A lot
Detect Feature Importance	Somewhat	Yes	No
Availability of libraries for big data scenarios	Yes	Yes	No

Chapter 3 Solution Design

The project aims to achieve following:

- To predict the journey time for short future, e.g. 15/30/60 minutes ahead.
- To produce a working prototype for the above prediction.

As discussed in the previous section, a complete data centric approach was adopted to solve the journey time prediction problem. Statistical learning theory [76] appeared promising for the solution.

As we defined our problem in section 2.3, applying supervised learning we need to find a hypothesis which would approximate, f. For the problem of prediction 15 minute ahead,

$$\mathrm{x} = jt_t\ , v_{i,t}\ , v_{i,t-1}\ , v_{i,t-2}\ , \dots\ v_{i,t-14}$$

$$\mathrm{y} = jt_{t+15}$$

Similarly, we can define x, y pair for both predictions, 30 and 60 minute ahead.

When the output y is one of a finite set of values or categorical values the learning problem is called classification. When y is a number or continuous value, the learning problem is called regression. Technically, solving a regression problem is finding a conditional expectation or average value of y, because the probability that we have found exactly the right real-valued number for y is 0.

Clearly, the problem at the hand is a regression problem which needs to be solved with proper learning algorithms. There are different families of algorithms for solving a regression problem like; Neural Networks, Decision Tree and Ensembles, SVM and others. Different algorithms were tried and finally Decision Tree algorithm was chosen to design the final solution.

Decision Tree

Decision tree induction is one of the simplest and yet most successful forms of machine learning. Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one. For a regression problem, the model fits a value for the region which is average of all the labels or response variables.

The overall process of growing a decision tree [79] can be explained as following. Our data consists of *p* attributes as x and a response as y, for each of N observations: that is, (x_i, y_i) for $i = 1, 2, \dots, N$, with $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$. The algorithm needs to automatically decide on the splitting variables and split points, and also what topology (shape) the tree should have. Suppose first that it partitions into M regions $R_1, R_2, \dots, R_M$, and it models the response as a constant c_m in each region:

$$h(x)\ = \sum_{m=1}^{M} c_m I\{(x, y) \in R_m\}$$

If criterion minimization of the sum of squares or variance is adopted: $\Sigma\big(\ y_i - h(x_i)\big)^2$, then it is easy to see that the best c_m is just the average of y_i in the region, R_M:

$$c_m\ =\ \mathrm{avg}(\ y_i | x_i\ \in R_m)$$

The algorithm adopts a greedy divide-and-conquer strategy. It always tests the most important attribute first. This test divides the problem up into smaller sub problems that can then be solved recursively.

Starting with all of the data, consider a splitting variable j and split point s, and define the pair of half-planes:

$$R_1(j,s) = \{x | x_j \leq s\} \quad \text{and} \quad R_2(j,s) = \{x | x_j > s\}$$

Then it seeks the splitting variable j and split point, s that solves

$$\min_{j,s} [\min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

For any choice of j and s, the inner minimization is solved by:

$$c_1 = \text{avg}(y_i | x_i \in R_1(j,s)) \quad \text{and} \quad c_2 = \text{avg}(y_i | x_i \in R_2(j,s))$$

For each splitting variable, the determination of the split point, s can be done very quickly and hence by scanning through all of the inputs, determination of the best pair (j, s) is feasible. After finding the best split, the data can be partitioned into the two resulting regions and it can be repeated with the splitting process on each of the two regions. Then this process is repeated on all of the resulting regions. At the end it results with the required decision tree.

A very large tree might overfit the data, while a small tree might not capture the important structure. Tree size, depth or level of a tree is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data. One simple approach might be to split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold.

3.1.1 Pros and Cons of Decision Tree

The pros and cons of decision tree based model are as below [78]:

Pros

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in the previous chapter.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.
- Feature normalisation not required.

Cons

- Low bias, high variance
- Not Stable: Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

Methodology

The project follows the steps suggested by the CRIPS DM data mining guide [81] and is planned to proceed in a more iterative way with following phases:

Data Collection

The historic data was collected from the identified source and cleaned. All the data after cleaning stored in a cloud storage for further processing. For the real time data, it was simulated using a part of historic data.

Data Exploration and Visualisation

After data is collected effort will be made to understand it fully. Necessary data visualisation was also made to make the data present in more convincing way.

Investigation of Prediction Technics
In this phase extensive research was done and different machine learning models were evaluated and right algorithm was identified to build the model.

Data Preparation
The data needs to be presented in certain format(s) as required by identified algorithm. Also it may need certain transformation e.g. dealing with null values. The following section 3.3 briefs about the data transformation made for the project.

Prototype (Model) Building
Different models were built and evaluated for the best one. The prototypes of the models were developed using identified platform and programming language.

The stages described above were not executed in isolation to the others. There are dependencies in one stage to one or more of the others. These were executed in an iterative way, each iteration being a small sprint.

The whole process may well be represented in the following diagram:

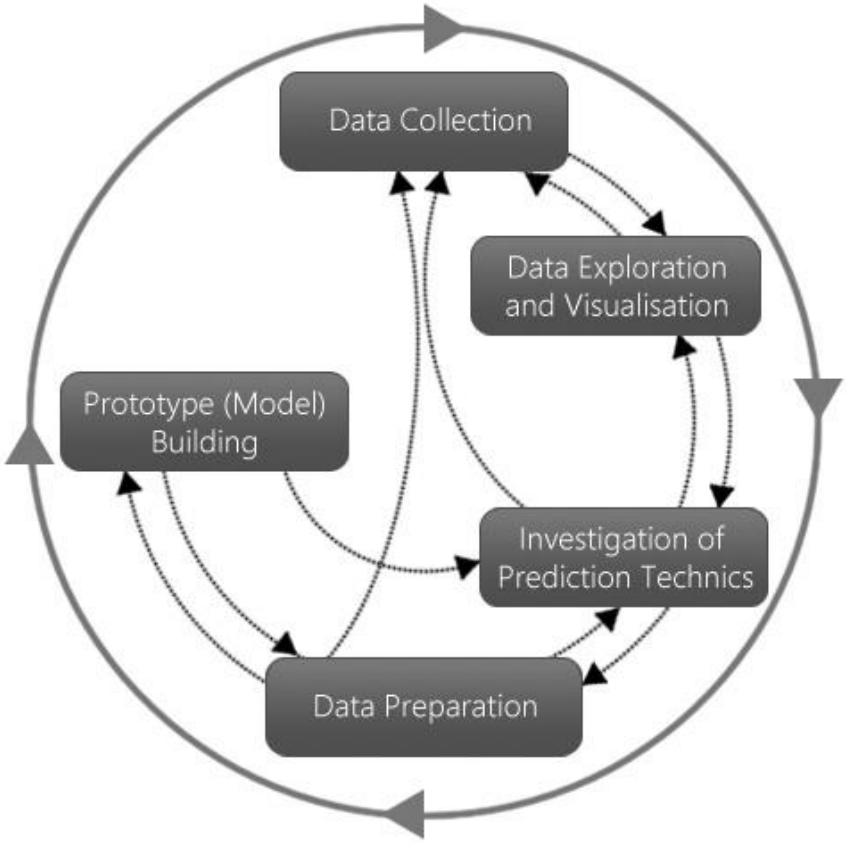

Figure 7 : Different stages of the projects

Data Preparation

The problem analysis in section 2 and solution identified with applying machine learning algorithms need a data set with a flat structure containing the features and label or the target variables. In the analysis before the features are represented by x and label as y. However,

the raw data collected and cleaned are in the temporal structure, so the data needs to be converted. The following figure (*Figure* 8) depicts the data transformation with an example where he data is being transformed with an intention of prediction before 5 minutes. So the entire data for 5 minutes have been converted to one row and journey time 5 minutes ahead is identified as the target variable.

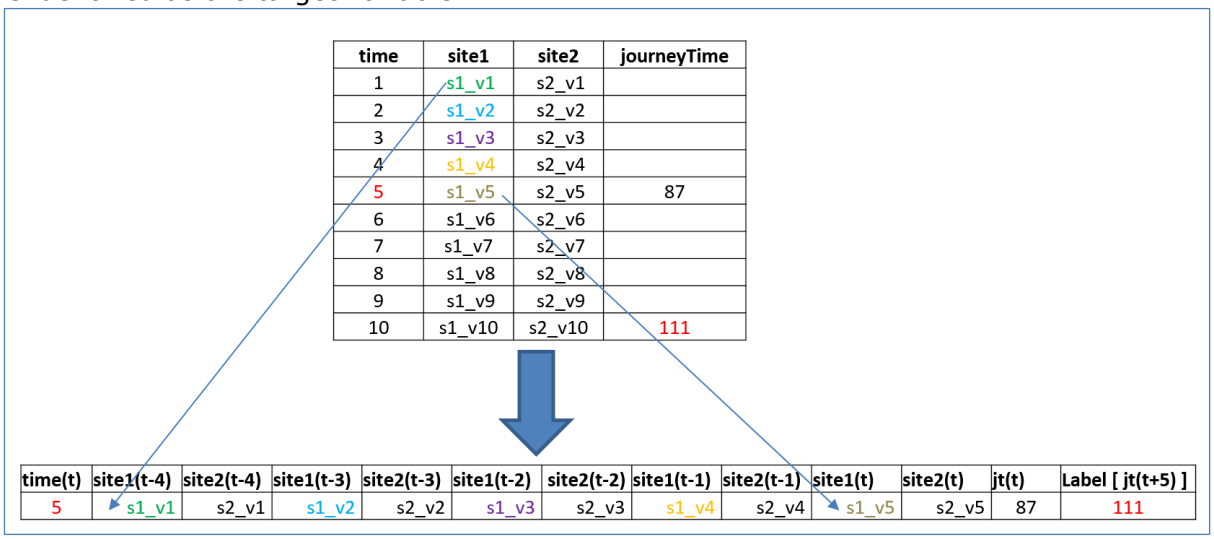

Figure 8 : Data Transformation - Conversion to flat feature vector and label

One-year worth of data was collected (from the C2 system) and prepared. The period taken was August 2014 to August 2015. The cleaned data was further checked for null values and all the records containing null values are deleted. The data set covered data for around 100 sites.

The final dataset was divided into two with 70:30 ratios. The first one was used for training the model and rest for testing evaluation.

Logical Architecture

The following figure (*Figure* 9) depicts the logical architecture for the project.

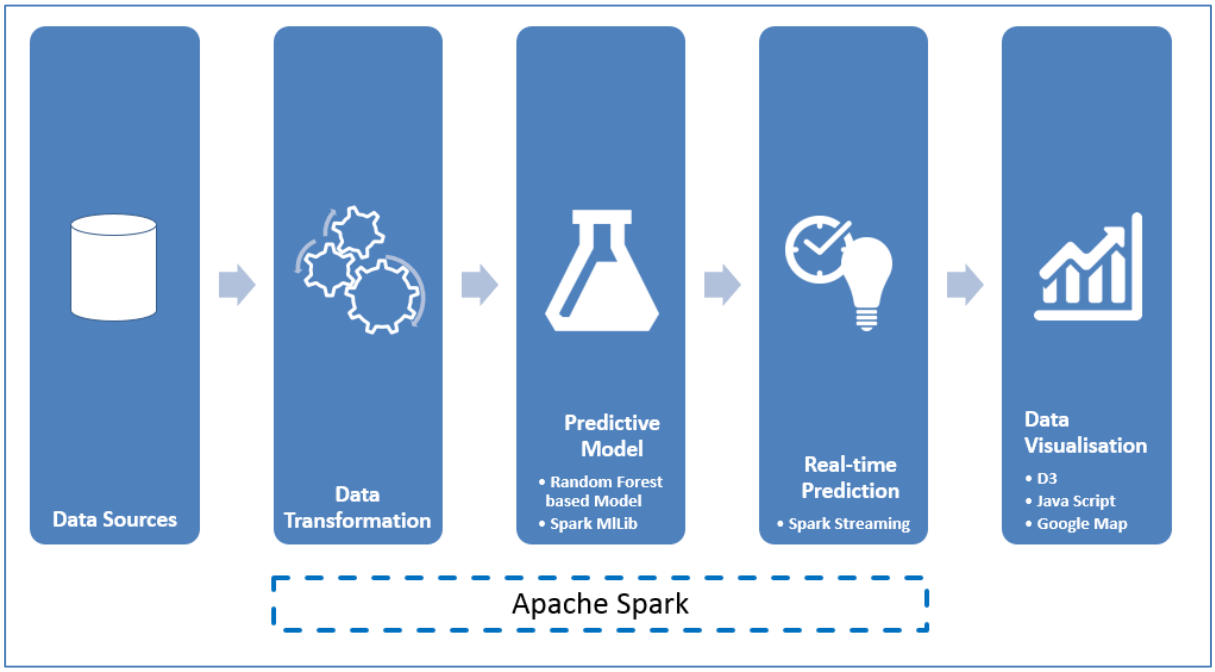

Figure 9 : Logical Architecture

The workflow has been designed to take the raw data and produce the model which can be used to make prediction and visualisation in real time. While the model building happens in the batch mode (offline of making prediction), the prediction can be made in real time provided real time data available.

Proposed Solution Architecture

An overall technical architecture has been proposed for real-time analytics for urban road transport. This architecture is designed to scale out and support any workload for any analytics (real-time or batch) needs related to urban transport or for any organisation for that matter. A system has been designed as a proof of concept implementing the said architecture and tested with a single workload covering a complete use case of predicting traffic delays for a route in the Grater Manchester area in real-time, the solution for which has been described in detail in the following subsection(s).

3.1.2 Data Sources as Publishers

Data can be sourced from many different kinds of sources like SCOOT devices, different kinds of sensors or any other Internet of Things (IoT) sources. These sources would be publishing each data point as a message to messaging system either by themselves or through an external publisher program.

3.1.3 Distributed Messaging System

A distributed messaging system is a *publish-subscribe* service which can scale for millions of messages or events per second and more. The scalability comes through distributed computing where the service is configured to run in a cluster of more than one computer. These are built with fault tolerance. Once messages published to this system, there can be some processing inside and it can pass the raw or processed data to a permanent storage. Any subscriber, ideally a stream processor can subscribe for messages and do the necessary processing as processing capabilities in a messaging system are limited. A distributed messaging system is vital for real-time analytics involving big data.

This system can be set up either on premise or in the cloud. Open source product, *Apache Kafka* [86] provides distributed messaging and is built for high-throughput and high scalability. *Event Hub* [87] and *Kinesis* [88] are the products with similar offerings available as a service on Microsoft cloud and Amazon cloud respectively. Any of these services can be used to fulfil the requirements based on which environment the entire system is deployed.

3.1.4 Analytics Engine

Apache Spark [85], has been chosen as the analytics platform. It provides a robust framework for large scale distributed computing and comes with different subsystem on top of it with capabilities of machine learning, both stream and batch processing, graph processing etc. It is an ideal platform meeting all kinds of needs to perform massive scale and advanced Analytics under one umbrella. It is open source but also available under different commercial distribution by different vendors. Spark is also available as a service by major cloud service providers like Microsoft, Amazon and IBM.

The following section provides a brief overview of Apache Spark and its different components and explains it as an analytics engine in detail.

3.1.5 Storage System

For storage, a traditional RDBMS system or a NoSQL system can be used based on the kinds of further requirement. A distributed and resilient file system can also meet the requirement.

3.1.6 The Visualisation and Other Systems

A system for data visualisations and for other purposes can be fed by APIs exposed in the different layers in the architecture as per the need.

While the complete technical architecture is proposed for the real time analytics, for the purpose of the current project the scope has been reduced to analytics engine and data visualisation. The prototype has been built on top of Apache Spark as the analytics engine and visualisations were made with Google map API [90] and D3 based java script library [89] for web browser rendering. For the real time messaging it was simulated through reading and writing to operating system file system. The prototype was developed and deployed in Microsoft Azure cloud.

The following diagram shows the technical architecture for the entire solution.

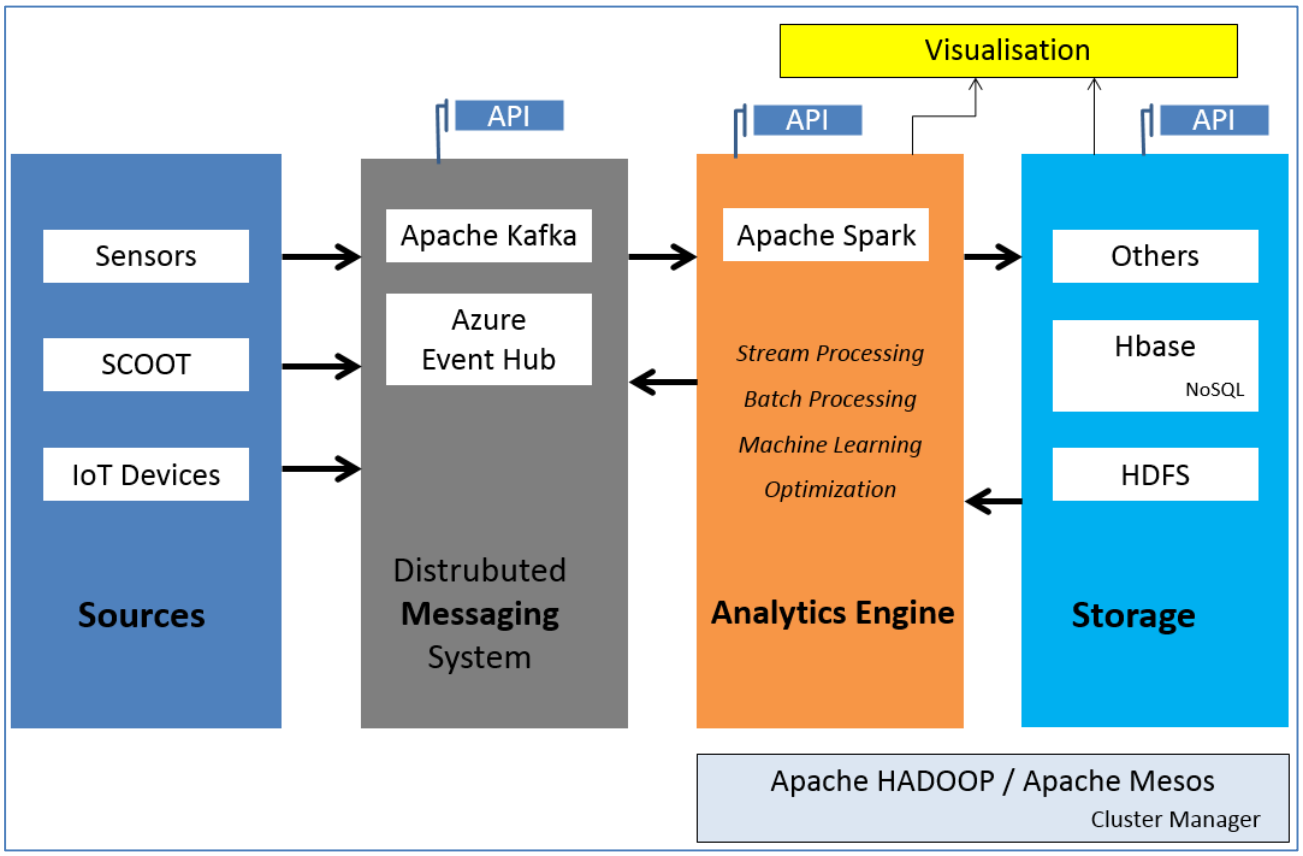

Figure 10 : Solution Architecture

Overview of Apache Spark

Apache Spark [85] is a fast and general engine for large-scale data processing which can be characterised by following attributes:

Speed

Spark extends the popular MapReduce [91] model to efficiently support more types of computations, including interactive queries and stream processing. Speed is important in processing large datasets, as it means the difference between exploring data interactively and waiting minutes or hours. One of the main features Spark offers for speed is the ability to run computations in memory, but the system is also more efficient than MapReduce for complex applications running on disk.

Spark has an advanced DAG execution engine [48] that supports cyclic data flow and in-memory computing resulting speed 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Ease of Use
Spark is written using functional programming language Scala [64], which is also a JVM based language. Spark is designed to be highly accessible, offering simple APIs in Python, Java, Scala, and SQL, and rich built-in libraries. It allows to use it interactively from the Scala, Python and R shells.

Generality
Spark is designed to cover a wide range of workloads that previously required separate distributed systems, including batch applications, iterative algorithms, interactive queries, and streaming. By supporting these workloads in the same engine, Spark makes it easy and inexpensive to combine different processing types, which is often necessary in production data analysis pipelines. In addition, it reduces the management burden of maintaining separate tools by combining SQL, streaming, and complex analytics like machine learning, graph processing and optimisation tasks.

Runs on Different Platform
It can run on both Windows and Linux based operating systems either on standalone mode or on top of a cluster like Hadoop, Mesos etc. Its ecosystem allows it to access diverse data sources including HDFS, Cassandra, HBase, S3 and it is increasing.

3.1.7 Components of Spark
The Spark project contains multiple closely integrated components [93]. At its core, Spark provides an engine for computation that is responsible for scheduling, distributing, and monitoring applications consisting of many computational tasks across many worker machines, or a computing cluster. Because the core engine of Spark is both fast and general-purpose, it powers multiple higher-level components specialized for various workloads, such as SQL or machine learning. These components are designed to interoperate closely, letting you combine them like libraries in a software project.

Different components in Spark stack can be represented as in the following diagram:

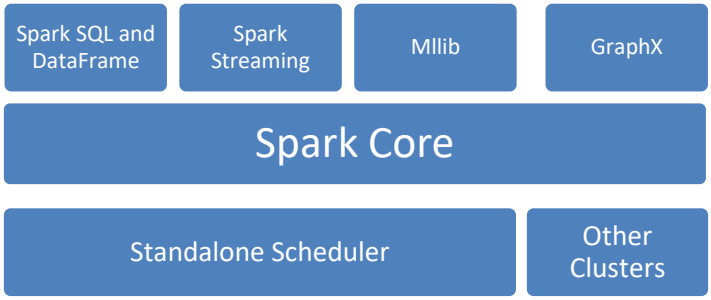

Figure 11: Components of Spark

Apart from the out of the box components, there is also a community driven library or repository of Spark packages [92] where anyone can publish one's open source code as an application on top of Spark stack for others to consume.

Spark Core
Spark Core contains the basic functionality of Spark, including components for task scheduling, memory management, fault recovery, interacting with storage systems, and more. Spark Core is also home to the API that defines resilient distributed datasets (RDDs), which are Spark's main programming abstraction. RDDs represent a collection of items

distributed across many compute nodes that can be manipulated in parallel. Spark Core provides many APIs for building and manipulating these collections.

Spark SQL and DataFrame
Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrames and can also act as distributed SQL query engine. A DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood. DataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs.

MLlib - Machine Learning library for Apache Spark
MLlib is Spark's machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as lower-level optimization primitives and higher-level pipeline APIs.

Spark Streaming
Spark Streaming is a Spark component that enables processing of streaming data. Examples of data streams include logfiles generated by production web servers, or queues of messages containing status updates posted by users of a web service. Spark Streaming provides an API for manipulating data streams that closely matches the Spark Core's RDD API, making it easy for programmers to learn the project and move between applications that manipulate data stored in memory, on disk, or arriving in real time. Underneath its API, Spark Streaming was designed to provide the same degree of fault tolerance, throughput, and scalability as Spark Core.

GraphX
GraphX is a library for manipulating graphs (e.g., a social network's connections graph) and performing graph-parallel computations. Like Spark Streaming and Spark SQL, GraphX extends the Spark RDD API, allowing us to create a directed graph with arbitrary properties attached to each vertex and edge. GraphX also provides various operators for manipulating graphs (e.g., subgraph and mapVertices) and a library of common graph algorithms (e.g., PageRank and triangle counting).

Cluster Managers
Under the hood, Spark is designed to efficiently scale up from one to many thousands of compute nodes. To achieve this while maximizing flexibility, Spark can run over a variety of cluster managers, including Hadoop YARN, Apache Mesos, and a simple cluster manager included in Spark itself called the Standalone Scheduler. When installed on an empty set of machines, the Standalone Scheduler provides an easy way to get started. At the same time when configured on a Hadoop YARN or Mesos cluster, its support for these cluster managers allows the spark applications to also run on them.

3.1.7.1 Advantages of Unified Stack Approach
There are several benefits because of the tight integration. First, all libraries and higher level components in the stack benefit from improvements at the lower layers. For example, when Spark's core engine adds an optimization, SQL and machine learning libraries automatically speed up as well. Second, the costs associated with running the stack are minimized, because instead of running 5–10 independent software systems, an organization needs to run only one. These costs include deployment, maintenance, testing, support, and others. This also means that each time a new component is added to the Spark stack, every organization that uses Spark will immediately be able to try this new component. This changes the cost of trying out a new type of data analysis from downloading, deploying, and learning a new software project to upgrading Spark.

Finally, one of the largest advantages of tight integration is the ability to build applications that seamlessly combine different processing models. For example, in Spark one can write one application that uses machine learning to classify data in real time as it is ingested from streaming sources. Simultaneously, it can be queried the resulting data, also in real time, via SQL (e.g., to join the data with unstructured log files).

Chapter 4 Algorithm for Scale - Decision Trees in MLlib

The size of the dataset considered for this project is in the range of multiple gigabytes. The machine learning algorithms require iterative data processing so the effective size of the data being processed increase many fold. Hence to apply machine learning algorithms for data in gigabytes size is not possible in a single machine environment. It needs distributed computing for which Apache Spark provides an ideal application development framework along with libraries for different kinds of workloads as discussed in previous section.

MLlib library in Spark implements many machine learning algorithms along with decision trees and ensembles. The algorithms have been optimised to scale and make the best of Spark platform.

Overall Algorithm – Decision Tree in MLlib

The overall algorithm for Decision Tree implemented in MLlib can be explained as below [94] [95]:

- *The algorithm partitions data by instances (rows). On each iteration, it splits a set of nodes. In order to choose the best split for a given node, sufficient statistics are collected from the distributed data. For each node, the statistics are collected to some worker node, and that worker selects the best split.*
- *This setup requires discretization of continuous features. This binning is done during initialization, after which each continuous feature becomes an ordered discretized feature with at most maxBins possible values.*
- *The main loop in the algorithm operates on a queue of nodes. These nodes lie at the periphery of the tree being trained. If multiple trees are being trained at once, then this queue contains nodes from all of them. Each iteration works until the node queue is empty:*
  - *On the master node:*
    - *Some number of nodes are pulled off of the queue (based on the amount of memory required for their sufficient statistics).*
  - *On worker nodes:*
    - *The worker makes one pass over its subset of instances.*
    - *For each (node, feature, split) tuple, the worker collects statistics about splitting.*
    - *For each node, the statistics for that node are aggregated to a particular worker. The designated worker chooses the best (feature, split) pair, or chooses to stop splitting if the stopping criteria are met.*
  - *On the master node:*
    - *The master collects all decisions about splitting nodes and updates the model.*
    - *The updated model is passed to the workers on the next iteration.*

4.1.1 Best Split

Following are the high level explanation for the optimisation for finding best split for a decision tree [34] [35] [36].

Level-wise training: Splits for all nodes at the same level of the tree are selected simultaneously. This level-wise optimization reduces the number of passes over the dataset

exponentially - one pass for each level, rather than one pass for each node in the tree. It leads to significant savings in I/O, computation and communication.

Approximate quantiles: Single machine implementations typically use sorted unique feature values for continuous features as split candidates for the best split calculation. However, finding sorted unique values is an expensive operation over a distributed dataset. The MLlib decision tree uses quantiles for each feature as split candidates. It's a standard trade-off for improving decision tree performance without significant loss of accuracy.

Bin-wise computation: The best split computation discretizes features into bins, and those bins are used for computing sufficient statistics for splitting. The binned representations of each instance are precompute, which saves computation on each iteration.

Aggregation over partitions: The assumption is made that the number of splits are known in advance. Thus, the aggregates (at the appropriate indices) are stored in a single array for all bins and rely upon the RDD aggregate method to drastically reduce the communication overhead.

4.1.2 Parameters Tuning for Decision Trees Algorithm in MLlib

The hyperparameters which may be tuned for better performance are explained below [58]:

Table 3 : Hyperparameters

Parameter	Description
maxBins	Number of bins used when discretizing continuous features. ▪ Increasing maxBins allows the algorithm to consider more split candidates and make fine-grained split decisions. However, it also increases computation and communication. ▪ Note that the maxBins parameter must be at least the maximum number of categories M for any categorical feature. ▪ Default value: 32
maxDepth	Maximum depth of a tree. Deeper trees are more expressive (potentially allowing higher accuracy), but they are also costlier to train and are more likely to overfit.
impurity	Impurity measure used to choose between candidate splits. It is a string value and for regression only supported value is "*variance*" as of now.

For predicting journey time for different routes, all the decision tree models were trained with following hyper parameters.

- maxBins = 32
- maxDepth = 5
- Impurity = variance

Chapter 5 Evaluation

The aim of the evaluation is to evaluate the effectiveness of the machine learning algorithm based predictive model using real traffic data which includes (as described in section 2.3) traffic volume data as predictor and the future journey time as the target.

The entire dataset considered was divided into two sets with 7:3 ratios. The former dataset was used to build or train the model and the rest was used for testing the model and evaluating the result. The final figures were obtained by running the experiments 3 times in a row and by taking the average of the results.

Final experiments were performed on the data related to two routes with IDs in C2 system: 9800XUOOTFQD and 9800ZFG0V7IO. The former is a simple route of length 0.7 mile and the other is complex or compound one of length 3 miles. The routes are shown in the *Figure* 12 below on the map. As there are both simple and compound routes in the network, these two routes are representatives of all the routes in the network.

Route: 9800XUOOTFQD, 0.7 miles	Route: 9800ZFG0V7IO, 3 miles
	

Figure 12 : Two Routes on the map

A separate prediction model was built per route per prediction type. The three predictions for which models were built are, prediction before 15 minute, 30 minute and 60 minute ahead. So there were 3 models per route and hence 6 models in total.

Evaluation Metrics

Following evaluation metrics were calculated based on the prediction results for a model.
Mean Absolute Error (MAE) [65] is the average of the absolute difference between actual journey time and predicted journey time in seconds.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Ai - Pi|$$

Mean Absolute Percentage Error (MAPE) [66] is the average of the absolute difference between actual journey time and predicted journey time divided by the actual journey time.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Ai - Pi|}{Ai} X\ 100$$

While MAE depends on the unit of journey time like minutes or seconds, MAPE does not. MAPE, as the name suggests, expresses the error as a percentage.

Prediction Results

Following table (in *Table* 3) shows all the evaluation metrics for the two identified routes.

Table 4 : Prediction Results

Route	**Prediction Ahead**					
	15 min Ahead		30 min Ahead		60 min Ahead	
	MAE	MAPE	MAE	MAPE	MAE	MAPE
9800XUOOTFQD	15 seconds	9 %	16 seconds	10 %	22 seconds	14 %
9800ZFG0V7IO	1 minute	7 %	1.1 minute	8 %	1.45 minute	11 %

The figures in the table were obtained by running an experiment 3 times in a row and by taking the average of the results. For the route 9800XUOOTFQD, journey time was analysed in seconds so we have MAE in seconds. But, for the route 9800ZFG0V7IO, journey time was analysed in minutes so we have MAE in minutes.

The results based on the models for the two routes looks quite encouraging. Even for predicting journey time ahead of 1 hour, we have MAPE, 14 % and 11 % for the short and long routes respectively. This shows that by defining a threshold journey time for a route at certain point of day, we can predict traffic congestion.

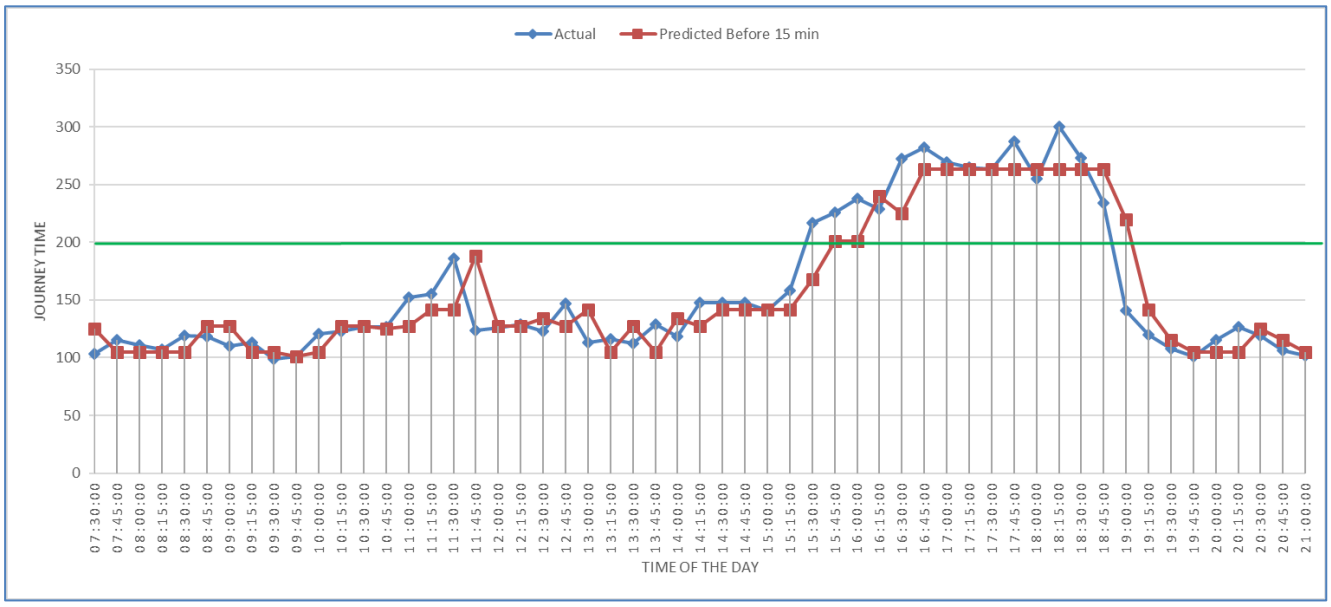

Figure 13 :Prediction 15 minute ahead for the route, 9800XUOOTFQD

The *Figure* 13 shows the prediction by the model for the entire day, a complete new set of data points to demonstrate that it can predict in real time provided the required data is available. On the x-axis it shows the time from 7.30 am to 9 pm, on the y-axis it shows journey time in seconds. The two series on the plot, actual journey time in blue and predicted journey time (15 minutes beforehand) in red shows that the prediction is matching quite closely and in terms of predicting congestions (assuming any journey taking more than 200 seconds), it has predicted almost every time.

Results from Similar Work

While one-on-one comparison is not possible as there are not many studies which predict journey time using real traffic volume data, there are some similar works which can be presented. In one of the recent experiments, the researchers [101] conducted an extensive experimental comparison on publicly available datasets using different methods and published the results. However, the congestion was an area of interest behind the study, it predicts the traffic flow for near future for different stations where a station is an aggregation of detectors from different lanes in the same location and compares different models using the single metric, MAPE. The following table contains the result [101].

Table 5 : Traffic Flow predictions at 16 stations. Error expressed using MAPE

Station	RW	SM	$SARIMA_{ML}$	$SARIMA_{Kal}$	$ARIMA_{Kal}$	SVR_{RBF}	SVR^{S}_{RBF}	SVR^{S}_{lin}	ANN
715916	9.02	9.24	6.75	**5.71**	9.49	6.01	5.99	6.27	6.31
716312	8.22	8.55	5.66	**4.89**	8.15	5.21	5.19	5.80	5.99
716551	9.47	9.58	6.22	**5.35**	7.97	5.80	5.76	6.29	6.20
716841	9.04	8.31	5.58	**4.99**	9.35	5.46	5.36	5.74	5.85
716933	7.66	8.91	5.67	**5.02**	7.59	5.39	5.35	5.62	5.69
717087	8.33	7.83	5.38	**4.80**	8.27	5.21	5.20	5.44	5.51
717123	7.75	7.87	4.98	**4.50**	7.07	4.85	4.77	5.07	5.25
717152	8.51	8.66	5.80	**5.16**	8.18	5.62	5.54	5.87	6.00
717190	8.30	9.30	5.60	**5.08**	8.13	5.46	5.38	5.75	5.75
717269	8.97	11.02	6.67	**5.47**	9.09	6.01	5.81	6.24	6.24
718144	9.31	9.75	6.05	**5.37**	8.00	5.75	5.59	6.09	6.15
737344	8.81	9.16	6.20	**5.14**	8.88	5.86	5.75	5.86	6.35
763626	7.94	8.44	4.75	**4.21**	7.24	4.76	4.55	4.91	5.28
764151	8.34	8.04	5.69	**4.96**	7.93	5.28	5.11	5.46	5.79
767678	9.14	8.42	5.99	**4.94**	9.41	5.59	5.40	5.66	5.82
768682	7.58	7.46	4.56	**3.79**	7.73	4.22	4.12	4.45	4.59

Interpreting Decision Tree Model

As part of the project, custom implementation was made so that a decision tree model can be visualized automatically. A final tree, for a trained decision tree model may look like the one shown in the following figure (*Figure* 14). For an illustration of tree visualisation, it displays the tree for the model to predict 15 minute ahead for the route, 9800XUOOTFQD. The root node is in left most side and leaf nodes on right most.

Every node, apart from the leaf nodes has a condition which results in true or false. The tree can be traversed based the result of the conditions in the nodes – true condition leads to right (green line in the diagram) and false to left (red line in the diagram). The leaf node represents the predictions. The tree is automatically learned by the model after the training with historic data as input.

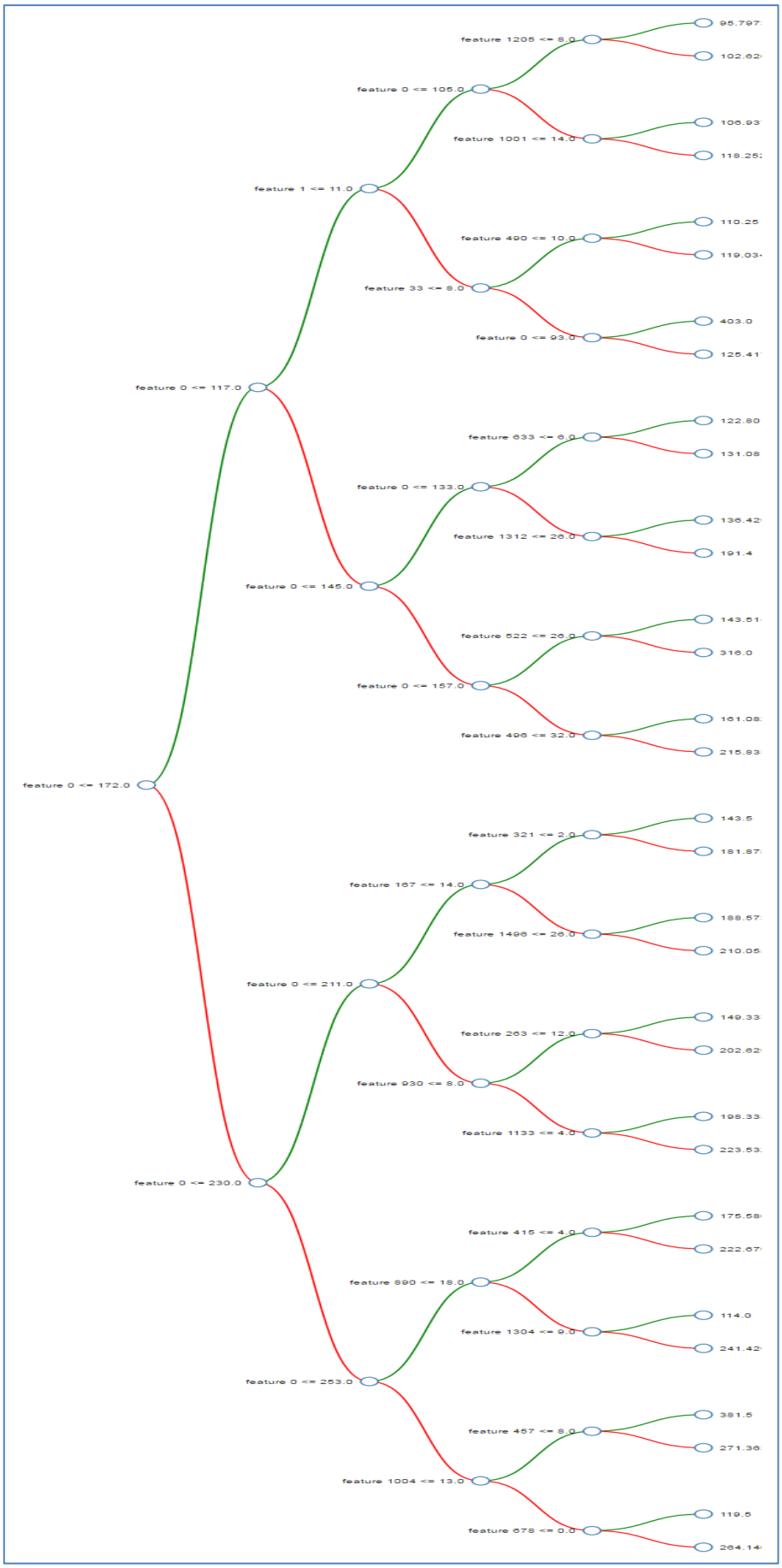

Figure 14 : Trained Decision Tree Model Visualisation

Chapter 6 Conclusion

For this project, historic data was used to build models which can predict the journey time for near future viz. 15/30/60 minutes ahead. Once a model is built or trained, it can make prediction in real time given the data is available in real time. Machine learning algorithms were used to build models with a pure data centric approach. Certain data visualisations were also made as part of the project.

The evaluation in the previous section demonstrates the effectiveness of machine learning based modelling for predicting journey time. In particular, where the journey time is predicted to be high, this indicates that congestion is likely.

Limitations

The scope in the project was limited to decision trees and related algorithms and others were not investigated thoroughly. There is a good chance that other algorithms might give even better results. Another point is that, for building the model the training with examples were made in batch mode and done offline. There is a risk that if the model is not updated regularly or periodically, the model might get obsolete once the traffic pattern changes and prediction might go wrong.

To train and predict effectively, the current prediction model needs a fine tuned lag time and necessary data preparation accordingly. This might prove to be difficult at times.

If the solution is considered to be used a practical purpose, there is another significant limitation. As per the current approach a model needs to be built per route per kind of prediction and there should be historic journey time data available for that route. In an urban transport network, there are different routes. Building a new model for each route might be an expensive affair. Also, a route can be identified in a network, with different combination of source and destinations. So with the current approach, it is not helpful to make prediction for a newly identified route unless we build a fresh model.

Future Work

As identified before as a limitation, further work should be done to build a single model which would be capable of predicting journey time for any route in the network. Also, there is greater scope for trying out different state of the art machine learning algorithms.

Recent studies [96] [97] [98] deep learning based model can be very useful. Specially the deep neural networks with LSTM cells [98] may provide solution to all the identified limitations above:

- It can automatically determine the optimal time lags and ideal to deal with time series data.
- A single model can be trained to predict journey time for all the identified routes.
- It can be trained in both online or offline manner.

Certainly, LSTM based deep learning methods for journey time prediction should be studied and utilised further.

With the current approach, the journey time for a particular route in the transport network can be predicted for the near future successfully. This approach can possibly be extended to make different kinds of prediction given the relevant data available. Air pollution is another acute problem in urban areas. It would be interesting to make prediction for emission of identified pollutants. Similarly, road accidents and other incidents can also be predicted for

near future. It provides greater opportunities even more predictive analytics after combining the urban road transport data with other datasets like weather data, local events data etc.
Further study should also be made to make use the result of this study to improve the urban traffic system. This can be done by making use of real time prediction of journey time and building systems to control traffic flows across the network so that the entire transport network can be optimised. Similar data centric approach could be taken by introducing reinforcement learning [99] to optimize urban traffic system.

Bibliography

1. McCluskey, L., Vallati, M. (2015) Extracting Information from Urban Traffic Data to Improve Road Traffic Management.
2. Smith, B. and Demetsky, M. (1997). "Traffic Flow Forecasting: Comparison of Modeling Approaches." J. Transp. Eng., 10.1061/(ASCE)0733-947X (1997)123:4(261), 261-266.
3. Williams, B. M., & Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. Journal of transportation engineering, 129(6), 664-672.
4. Huan, G. U. O., Xinping, X. I. A. O., & Jeffrey, F. (2013). Urban road short-term traffic flow forecasting based on the delay and nonlinear grey model. Journal of Transportation Systems Engineering and Information Technology, 13(6), 60-66.
5. Chen, H., & Grant-Muller, S. (2001). Use of sequential learning for short-term traffic flow forecasting. Transportation Research Part C: Emerging Technologies, 9(5), 319-336.
6. Bacon, J., Bejan, A. I., Beresford, A. R., Evans, D., Gibbens, R. J., & Moody, K. (2011). Using real-time road traffic data to evaluate congestion (pp. 93-117). Springer Berlin Heidelberg.
7. Yoon, B., & Chang, H. (2014). Potentialities of data-driven nonparametric regression in urban signalized traffic flow forecasting. Journal of Transportation Engineering, 140(7), 04014027.
8. Wu, S., Yang, Z., Zhu, X., & Yu, B. (2014). Improved k-nn for short-term traffic forecasting using temporal and spatial information. Journal of Transportation Engineering, 140(7), 04014026.
9. Vlahogianni, E., & Karlaftis, M. (2013). Testing and comparing neural network and statistical approaches for predicting transportation time series. Transportation Research Record: Journal of the Transportation Research Board, (2399), 9-22.
10. Zhou, H., & Hirasawa, K. (2014). Traffic conduction analysis model with time series rule mining. Expert Systems with Applications, 41(14), 6524-6535.
11. Oh, S., Byon, Y. J., Jang, K., & Yeo, H. (2015). Short-term travel-time prediction on highway: a review of the data-driven approach. Transport Reviews, 35(1), 4-32.
12. Limbach, W. E., & Call, C. A. (1996). http://www. tandfonline. com/doi/abs/10.1080/15324989809381521#. U6sHZBbIv9I. Journal of Range Management, 49(4).
13. NZ Transport Agency. (2014). Travel time predictability. Retrieved from http://www.nzta.govt.nz/assets/resources/research/reports/554/docs/554.pdf.
14. Mai, T., Ghosh, B., & Wilson, S. (2014, August). Short-term traffic-flow forecasting with auto-regressive moving average models. In Proceedings of the Institution of Civil Engineers-Transport (Vol. 167, No. 4, pp. 232-239). Thomas Telford Ltd.
15. Zhang, Y., Zhang, Y., & Haghani, A. (2014). A hybrid short-term traffic flow forecasting method based on spectral analysis and statistical volatility model. Transportation Research Part C: Emerging Technologies, 43, 65-78.
16. Williams, B. M., & Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. Journal of transportation engineering, 129(6), 664-672.
17. Davis, G. A., & Nihan, N. L. (1991). Nonparametric regression and short-term freeway traffic forecasting. Journal of Transportation Engineering, 117(2), 178-188.
18. Huang, M., & Lu, B. (2010, March). Short-term traffic flow parameters prediction based on multi-scale analysis and artificial neural network. In Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on (Vol. 1, pp. 214-217). IEEE.
19. Fusco, G. A. E. T. A. N. O., & Colombaroni, C. H. I. A. R. A. (2013, July). An integrated method for short-term prediction of road traffic conditions for Intelligent Transportation Systems Applications. In 7th WSEAS European Computing Conference, Dubrovnik (pp. 25-27).

20. Chen, H., & Grant-Muller, S. (2001). Use of sequential learning for short-term traffic flow forecasting. Transportation Research Part C: Emerging Technologies, 9(5), 319-336.
21. Haworth, J., & Cheng, T. (2012). Non-parametric regression for space–time forecasting under missing data. Computers, Environment and Urban Systems, 36(6), 538-550.
22. Prasad, K. S. N., & Ramakrishna, S. (2014). An efficient traffic forecasting system based on spatial data and decision trees. Int. Arab J. Inf. Technol., 11(2), 186-194.
23. Duarte, A., Garcia, C., Giannarakis, G., Limão, S., Polydoropoulou, A., & Litinas, N. (2010). New approaches in transportation planning: happiness and transport economics. NETNOMICS: Economic Research and Electronic Networking, 11(1), 5-32.
24. Cenamor, I., Chrpa, L., Jimoh, F., McCluskey, T. L., & Vallati, M. (2014). Planning & scheduling applications in urban traffic management. In Proceedings of the Annual Workshop of UK Planning & Scheduling Special Interest Group.
25. Shah, M. M. S., Chrpa, L., Kitchin, D. E., McCluskey, T. L., & Vallati, M. (2013, August). Exploring Knowledge Engineering Strategies in Designing and Modelling a Road Traffic Accident Management Domain. In IJCAI.
26. North East Combined Authority. (2014). Open Data Service. Retrieved from https://www.netraveldata.co.uk/?page_id=30.
27. North East Combined Authority. (2014). Tyne and Wear Open Data Services Platform [API Specification]. Retrieved from https://www.netraveldata.co.uk/wp-content/uploads/2014/04/333551-OpenDataService-APISpecification.pdf.
28. North East Combined Authority. (2014). Map. Retrieved from http://www.transportnortheast.com/public/map/map.htm.
29. SCOOT. Retrieved from http://www.scoot-utc.com/
30. UMTC. (2009). UTMC Framework Technical Specification. Retrieved from http://www.utmc.eu/SiteAssets/Pages/PageNotFoundError/UTMC-TS003.003%20Framework%20Spec%20Dec09.pdf.
31. Atkins. Integrated urban traffic management and control strategies. Retrieved from http://www.atkinsglobal.com/~/media/Files/A/Atkins-Global/Attachments/sectors/roads/library-docs/technical-journal-3/integrated-urban-traffic-management-and-control-stratigies.pdf.
32. Piatetsky-Shapiro, G. (1996). Advances in knowledge discovery and data mining (Vol. 21). U. M. Fayyad, P. Smyth, & R. Uthurusamy (Eds.). Menlo Park: AAAI press.
33. Marbán, Ó., Mariscal, G., & Segovia, J. (2009). A data mining & knowledge discovery process model. Data Mining and Knowledge Discovery in Real Life Applications, 2009, 8.
34. TheApacheSpark. (2014, Jul 17). Scalable Distributed Decision Trees in Spark MLLib - M. Amde, H. Das, E. Sparks & A. Talwalkar [Video file]. Retrieved from https://www.youtube.com/watch?v=3rRrcPXHu98
35. Zheng, J., & Dagnino, A. (2014, October). An initial study of predictive machine learning analytics on large volumes of historical data for power system applications. In Big Data (Big Data), 2014 IEEE International Conference on (pp. 952-959). IEEE.
36. Gualtieri, M., Rowan Curran, A., TaKeaways, K., & To, M. T. B. P. P. (2013). The Forrester Wave™: Big Data Predictive Analytics Solutions, Q1 2013. Forrester research.
37. Inrix. (2015). Growing economy drives traffic congestion up in over three quarters of UK cities. Retrieved from http://inrix.com/press/scorecard-report-united-kingdom/.
38. Inrix. (2014). Economic & Environmental Impact of Traffic Congestion in Europe & the US. Retrieved from http://inrix.com/economic-environment-cost-congestion/.
39. Transport for Grater Manchester. (2016). Retrieved from http://www.tfgm.com/Corporate/Pages/TfGM.aspx.
40. Department for Transport. (2016). Statistics at DFT. Retrieved from https://www.gov.uk/government/organisations/department-for-transport/about/statistics.
41. Department for Transport. (2014). Road Lengths in Great Britain: 2013 [Statistical Release]. Retrieved from

https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/316680/road-lengths-in-great-britain-2013.pdf.
42. Department for Transport. (2016). Traffic by local authority (TRA89) [Statistical data set]. Retrieved from https://www.gov.uk/government/statistical-data-sets/tra89-traffic-by-local-authority.
43. Highways England. Retrieved from https://www.gov.uk/government/organisations/highways-england.
44. Drakewell C2. (2014, May). Cloud Journey Time Module User Guide.
45. Ranjan, R. (2014). Streaming big data processing in datacenter clouds. IEEE Cloud Computing, 1(1), 78-83.
46. Lorica, Ben, and O'Reilly Radar. (2015). Apache spark: Powering applications on-premise and in the cloud. ACI Information Group.
47. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: cluster computing with working sets. HotCloud, 10, 10-10.
48. Zaharia, M. (2016). An architecture for fast and general data processing on large clusters. Morgan & Claypool.
49. UC Berkeley AMPLab. (2012, Sep 2). Parallel Programming with Spark (Part 1 & 2) - Matei Zaharia [Video file]. Retrieved from https://www.youtube.com/watch?v=7k4yDKBYOcw
50. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2012, April). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 2-2). USENIX Association.
51. Meng, X., Bradley, J., Yuvaz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Xin, D. (2016). Mllib: Machine learning in apache spark. JMLR, 17(34), 1-7.
52. Bradley, J. & Amde, M. (2014, September 29). Scalable Decision Trees in MLlib [Web log post]. Retrieved from https://databricks.com/blog/2014/09/29/scalable-decision-trees-in-mllib.html.
53. Bradley, J. & Amde, M. (2015, January 21). Random Forests and Boosting in MLlib [Web log post]. Retrieved from https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html.
54. Shanahan, J. G., & Dai, L. (2015, August). Large scale distributed data science using apache spark. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 2323-2324). ACM.
55. Richter, A. N., Khoshgoftaar, T. M., Landset, S., & Hasanin, T. (2015, August). A Multi-Dimensional Comparison of Toolkits for Machine Learning with Big Data. In Information Reuse and Integration (IRI), 2015 IEEE International Conference on (pp. 1-8). IEEE.
56. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. (2012). Discretized streams: A fault-tolerant model for scalable stream processing (No. UCB/EECS-2012-259). CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE.
57. Amde, M., Das, H., & Talwalkar, A. (2014). Scalable Distributed Decision Trees in Spark MLLib. [pdf]. Retrieved from https://spark-summit.org/2014/wp-content/uploads/2014/07/Scalable-Distributed-Decision-Trees-in-Spark-Made-Das-Sparks-Talwalkar.pdf.
58. Apache. (2016). Decision Trees - spark.mllib. Retrieved from http://spark.apache.org/docs/1.6.1/mllib-decision-tree.html.
59. Dai, W., & Ji, W. (2014). A mapreduce implementation of C4. 5 decision tree algorithm. International Journal of Database Theory and Application, 7(1), 49-60.
60. Panda, B., Herbach, J. S., Basu, S., & Bayardo, R. J. (2009). Planet: massively parallel learning of tree ensembles with mapreduce. Proceedings of the VLDB Endowment, 2(2), 1426-1437.
61. Morales, G. D. F., & Bifet, A. (2015). Samoa: Scalable advanced massive online analysis. Journal of Machine Learning Research, 16, 149-153.
62. Baldominos, A., Albacete, E., Saez, Y., & Isasi, P. (2014, December). A scalable machine learning online service for big data real-time analysis. In Computational Intelligence in Big Data (CIBD), 2014 IEEE Symposium on (pp. 1-8). IEEE.

63. Murdopo, A. (2013). Distributed decision tree learning for mining big data streams. Master of Science Thesis, European Master in Distributed Computing.
64. École Polytechnique Fédérale de Lausanne (EPFL). (2016). Scala Programming Language. Retrieved from http://scala-lang.org/.
65. Wikipedia. (2016). Mean absolute error. Retrieved from https://en.wikipedia.org/wiki/Mean_absolute_error.
66. Wikipedia. (2016). Mean absolute percentage error. Retrieved from https://en.wikipedia.org/wiki/Mean_absolute_percentage_error.
67. Mori, U., Mendiburu, A., Álvarez, M., & Lozano, J. A. (2015). A review of travel time estimation and forecasting for Advanced Traveller Information Systems. Transportmetrica A: Transport Science, 11(2), 119-157.
68. Li, C. S., & Chen, M. C. (2014). A data mining based approach for travel time prediction in freeway with non-recurrent congestion. Neurocomputing, 133, 74-83.
69. Nikovski, D., Nishiuma, N., Goto, Y., & Kumazawa, H. (2005, September). Univariate short-term prediction of road travel times. In Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005. (pp. 1074-1079). IEEE.
70. Chow, A. H., Santacreu, A., Tsapakis, I., Tanasaranond, G., & Cheng, T. (2014). Empirical assessment of urban traffic congestion. Journal of advanced transportation, 48(8), 1000-1016.
71. Hunter, T., Moldovan, T., Zaharia, M., Merzgui, S., Ma, J., Franklin, M. J., ... & Bayen, A. M. (2011, October). Scaling the Mobile Millennium system in the cloud. In Proceedings of the 2nd ACM Symposium on Cloud Computing (p. 28). ACM.
72. University of California (UC) at Berkeley. (2010). Mobile Millennium Project. Retrieved from http://traffic.berkeley.edu/project.
73. OPTICITIES. (2016). Birmingham - Pilot City. Retrieved from http://www.opticities.com/pilot-cities/birmingham/.
74. Wang, J., Mao, Y., Li, J., Xiong, Z., & Wang, W. X. (2015). Predictability of road traffic and congestion in urban areas. PloS one, 10(4), e0121825.
75. Liang, Z., & Wakahara, Y. (2014). Real-time urban traffic amount prediction models for dynamic route guidance systems. EURASIP Journal on Wireless Communications and Networking, 2014(1), 1-13.
76. Bousquet, O., Boucheron, S., & Lugosi, G. (2004). Introduction to statistical learning theory. In Advanced lectures on machine learning (pp. 169-207). Springer Berlin Heidelberg.
77. Russell, S., & Norvig, P. (2005). AI a modern approach. Learning, 2(3), 4.
78. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 6). New York: springer.
79. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.
80. Caruana, R., & Niculescu-Mizil, A. (2006, June). An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning (pp. 161-168). ACM.
81. Wirth, R., & Hipp, J. (2000, April). CRISP-DM: Towards a standard process model for data mining. In Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining (pp. 29-39).
82. Mund, Sumit. (2015). Microsoft Azure Machine Learning. 1st ed. GB: Packt Publishing.
83. Wu, X., & Kumar, V. (2009). The top ten algorithms in data mining. Boca Raton, Fla;London;: CRC
84. Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63(1), 3-42.
85. Apache. (2016). Spark. Retrieved from http://spark.apache.org/.
86. Apache. (2016). Kafka. Retrieved from http://kafka.apache.org/.
87. Microsoft. (2016). Event Hubs. Retrieved from https://azure.microsoft.com/en-gb/services/event-hubs/.
88. Amazon. (2016). Kinesis. Retrieved from https://aws.amazon.com/kinesis/.
89. D3js. (2016). D3 Java Script Library. Retrieved from http://spark-packages.org/.

90. Google. (2016). Google Maps API. Retrieved from https://developers.google.com/maps/.
91. Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.
92. Databricks. (2016). Spark Packages. Retrieved from http://spark-packages.org/.
93. Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). Learning spark: lightning-fast big data analysis. " O'Reilly Media, Inc.".
94. GitHub. (2016). Decision Tree in Apache Spark MLlib Source Code. Retrieved from https://github.com/apache/spark/tree/master/mllib/src/main/scala/org/apache/spark/mllib/tree.
95. GitHub. (2016). Random Forest in Apache Spark MLlib Source Code . Retrieved from https://github.com/apache/spark/blob/master/mllib/src/main/scala/org/apache/spark/ml/tree/impl/RandomForest.scala.
96. Shang, Q., Lin, C., Yang, Z., Bing, Q., & Zhou, X. (2016). A hybrid short-term traffic flow prediction model based on singular spectrum analysis and kernel extreme learning machine. PLoS one, 11(8), e0161259.
97. Polson, N., & Sokolov, V. (2016). Deep Learning Predictors for Traffic Flows. arXiv preprint arXiv:1604.04527.
98. Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 54, 187-197.
99. Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.
100. Shah, M. M. S., Chrpa, L., Kitchin, D. E., McCluskey, T. L., & Vallati, M. (2013, August). Exploring Knowledge Engineering Strategies in Designing and Modelling a Road Traffic Accident Management Domain. In IJCAI.
101. Lippi, M., Bertini, M., & Frasconi, P. (2013). Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. IEEE Transactions on Intelligent Transportation Systems, 14(2), 871-882. doi:10.1109/TITS.2013.2247040