Smarty city Applications –Aarhus city Road data

Naga Jyothi Kunaparaju

Saint Peter's University nkunaparaju@saintpeters.edu

1-347-341-3307

1 CONTRIBUTION

This report is to detail main steps involved in the creation of Smart City application using Aarhus city dataset. This main aim of the project is to predict future traffic conditions based on the past and current data. Driving situations varies for different places and also different times. Driving conditions based on routes identified by some important features can be studied to make future predictions. These circumstances vary for different transportation systems like public or private. This analysis is helpful in many areas like travel management systems and safety management. Statistical modelling and visualization patterns might reveal some important information about the conditions in the city. In this project, we are going to apply K-means clustering algorithm to create similarity groups to recognize patterns. We can use traffic and road data to design web application which integrates big data into the system. We can integrate this application to another system that will provide Traffic, Parking and event information. The complete application will contain map and GUI frontend developed with JavaScript and JSP. For backend data processing we can use spark engine. Open source tools like Apache Tomcat application server is an efficient way to present information to the user in real time. We plan to host this application in AWS. It provides a cheaper

solution instead maintaining own servers to launch.
Other existing applications related to smart city and big data process information very efficiently. But the apps are lacking user-friendly GUI. It makes a big difference in usage of the application. Even though technology and architecture are up to date, lack of user friendliness is a big concern in most of the cases. The application that we are going design will extend those functionalities and provide the solution in a user-friendly way.

2 STATE OF THE ART

This paper details how technology used in building smart city project efficiently. With this details we know how cities make use online infrastructure that produces Big Data. How such data enables real-time analysis of city life.
For future development of cities smart city concept became an certain pattern. It also provides productive, competitive, open and transparent cities. Smart cities contain huge data stored digitally and numbers of objects connected online. Some Smart services include Smart Travel, Smart Education, Smart Public, Smart Water & Waste, Smart Energy, Smart Tourism, Smart Building Smart Living, Smart safety, smart manufacturing, Smart Travel, Smart event management. The target consumers are citizens and travelers to the city. City Pulse project is one of such projects whose aim is to describe an integrated approach to improve travel, transportation, and parking. City Pulse Project mainly concentrate on three main areas of Aarhus city road data set, which are Smart Parking, Smart Travel Plan and Smart traffic. This project developed by a nine different organizations. These are the tools and applications that City Pulse project provides.

- Journey Planner
- Social media analyser
- City Pulse Dashboard
- Tourism Planner
- Pickup Planner (Vehicle app and Client application)
- Event Publisher

RDF expansion Resource Description Framework, and it is a schema-less data model. With the increase in use of Internet of Things in smart city applications, RDF stream processing is one of the prime technology in Semantic Web and it became present W3C standard

for representing data on the web. Now there are many many RSP engines have created, because of the importance which will process semantically annotated data streams real time.

Here we will talk about some main applications provided by City Pulse project. A smart travel plan is one of the most widely used Smart city strategies. Smartness concept applied to travelers before, during and after their trip. Destinations of tourism attractions could increase their competitiveness level. Another application called Smart Parking which focuses parking problems. Parking problem increases as the numbers of vehicles on the road are increasing with the city growth in population. Issues which arise due to insufficient parking space are driver frustration, air pollution, and traffic congestion. Traffic Dashboard solves some of the problems. Creating Traffic Dash Board will provide many advantages to the user to utilise transport services in the day to day life. It can be identified, performance within a city improves, by combining these areas with other smart city sections such as Event Management, pollution management, and smart transportation services. As our target is to improve Smart Travel, Smart Traffic and Smart parking. Next sections describe closely into these three applications.

3 DATA

Aarhus city traffic data is a collection of datasets of vehicle traffic and street information, observed between two points of a street for a set duration of time over a period of twelve months. There are 449 observation points. The data is available both in semantically annotated format(TTL) and comma separated format using the city pulse information. Which show information of the position of each of the two sensors in the dataset, distance in meters, type of road, etc. Powerful tools and technology are not only available for the wealthiest, but many can get access to it. As technology becomes cheaper, access to data becomes available to everyone

3.1 META DATA:

- POINT_1_STREET
- DURATION_IN_SEC
- POINT_1_NAME
- POINT_1_CITY

- POINT_2_NAME
- POINT_2_LNG
- NDT_IN_KMH
- POINT_2_POSTAL_CODE
- POINT_2_COUNTRY
- POINT_1_STREET_NUMBER
- ORGANISATION
- POINT_1_LAT
- POINT_2_LAT
- POINT_1_POSTAL_CODE
- POINT_2_STREET_NUMBER
- extID
- ROAD_TYPE
- POINT_1_LNG
- REPORT_ID
- POINT_1_COUNTRY
- DISTANCE_IN_METERS
- REPORT_NAME
- RBA_ID
- _id

3.2 DATA:

1. status,
2. AvgMeasuredTime,
3. avgSpeed exitID,
4. medianMeasuredTime,
5. TIMESTAMP,
6. vehicleCount,
7. _id,
8. REPORT_ID

Because we are using Road dataset, we concentrate on the Smart city Dashboard application. This information stored in two data sets. One is meta data other one is actual data. Each record in dataset is observation of how many vehicles passed between two points and its average speed with in five-minute interval.

Meta data consists of geographic details such as Latitude and longitude of each road and exit information. There is a separate metadata file for each individual city. These two datasets can be combined with report_ID.

```
%pyspark

df1.show()
```

```
+------+---------------+--------+-----+-----------------+-------------------+------------+-------+---------+
|status|avgMeasuredTime|avgSpeed|extID|medianMeasuredTime|          TIMESTAMP|vehicleCount|    _id|REPORT_ID|
+------+---------------+--------+-----+-----------------+-------------------+------------+-------+---------+
|    OK|             92|      91|  646|               92|2014-02-13T11:30:00|          35|189978|   158446|
|    OK|             84|     100|  646|               84|2014-02-13T11:35:00|          26|190427|   158446|
|    OK|             78|     108|  646|               78|2014-02-13T11:40:00|          31|190876|   158446|
|    OK|             79|     106|  646|               79|2014-02-13T11:45:00|          28|191325|   158446|
|    OK|             77|     109|  646|               77|2014-02-13T11:50:00|          35|191774|   158446|
|    OK|             83|     101|  646|               83|2014-02-13T11:55:00|          36|192223|   158446|
|    OK|             83|     101|  646|               83|2014-02-13T12:00:00|          24|192672|   158446|
```

Figure 1 Road traffic data

```
%pyspark
df = df1.join(df2, (df1.REPORT_ID == df2.REPORT_ID) ).drop(df1.REPORT_ID)
df.show()
```

```
-------------+--------------------+------+
|status|avgMeasuredTime|avgSpeed|extID|medianMeasuredTime|          TIMESTAMP|vehicleCount|    _id|    POINT_1_STREET|DURATION_IN_SEC|POINT_1_NAME|PO
INT_1_CITY|POINT_2_NAME|      POINT_2_LNG|    POINT_2_STREET|NDT_IN_KMH|POINT_2_POSTAL_CODE|POINT_2_COUNTRY|POINT_1_STREET_NUMBER|ORGANISATION|
  POINT_1_LAT|       POINT_2_LAT|POINT_1_POSTAL_CODE|POINT_2_STREET_NUMBER|POINT_2_CITY| ROAD_TYPE|       POINT_1_LNG|REPORT_ID|POINT_1_COUNTRY|DISTA
NCE_IN_METERS|         REPORT_NAME|RBA_ID|
+------+---------------+--------+-----+-----------------+-------------------+------------+-------+------------------+---------------+------------+--
----------+------------+-----------------+------------------+----------+-------------------+---------------+---------------------+------------+----
-------------+------------------+-------------------+---------------------+------------+----------+------------------+---------+---------------+-----
-------------+--------------------+------+
|    OK|             92|      91|  646|               92|2014-02-13T11:30:00|          35|189978|Østjyske Motorvej|             75|        4321|
     Tilst|        4323|10.12519614484404|Nordjyske Motorvej|       113|               8382|        Denmark|                    0|        COWI|56.2
1731711429131|56.23490211108693|               8381|                    0|    Hinnerup|MAJOR_ROAD|10.10711200000003|   158446|        Denmark|
         2345|AAR_BT_0005 4321 ...|159619|
|    OK|             84|     100|  646|               84|2014-02-13T11:35:00|          26|190427|Østjyske Motorvej|             75|        4321|
     Tilst|        4323|10.12519614484404|Nordjyske Motorvej|       113|               8382|        Denmark|                    0|        COWI|56.2
1731711429131|56.23490211108693|               8381|                    0|    Hinnerup|MAJOR_ROAD|10.10711200000003|   158446|        Denmark|
         2345|AAR_BT_0005 4321 ...|159619|
|    OK|             78|     108|  646|               78|2014-02-13T11:40:00|          31|190876|Østjyske Motorvej|             75|        4321|
```

Figure 2 Merged Dataset

By running analytics on this data we will be able to find out some patterns trends of traffic in past for different situations like different weather conditions, in major city events, peak times, holiday period. If Real time streaming is possible

for more useful information and can be presented to the user which make their travel plans better which ultimately supports city tourism.

4 METHOD:

In this project we are trying to improve three smart city services those are explained below.

4.1 SMART TRAVEL

The Smart Tourism or Smart travel are part development of Smart Cities. With the help Event or Tourism destinations management system travels plans could be determined via maps and Dashboard like city pulse. Users are helped by displaying the routes that they can access and also type of transportation. With this information citizens can schedule of events easily for a complete day activity. The intention is to provide travelers with payment plan, the best travel option and real-time travel information via a web app.

We are going to use a rich set of advanced tools and libraries like Spark, MapReduce and Tableau, which are well suited for data analysis and Visualization.

4.2 SMART PARKING

Parking problems increase every day because the number of vehicles on the road increase day by day. Problems which arise due to insufficient parking space are:

1. Traffic congestion
2. Driver frustration
3. Air pollution.

The price for expanding parking area is extremely high. This kind of system helps to minimize traffic problems by finding a vacant space in a parking garage when busy timeline of the city.

4.3 SMART TRAFFIC

User presented with Real-time information about traffic, parking conditions and transit options. Choosing better options minimizes traffic issues associated with major events. Information presented is like traffic information signs boards, real-time warnings of accidents and detours. The Traffic Planner is a web application for citizens. This is also can be used for retrieving user travel and parking recommendations. Smart transportation systems are applications which, aim to provide services relating to different ways of transport and traffic management. Traffic congestions also determined with a detailed description about chances of being stuck in the traffic. The Idea of the app is to provide options like the type of user and activity they want perform with the application. If the person is tourist want to plan for their day to attend an attraction, the application will guide him with all possibilities. In case of any disaster occur travelers are rerouted and planned for exit. People warned with proper exit routes.

4.4 ARCHITECTURE

The architecture of the system shown in Fig 1. Spark will be used for data processing.

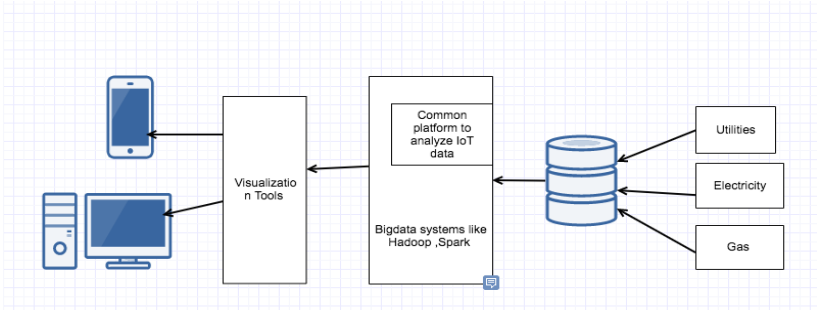

Figure 3

With the Advancement in Semantic Technologies for IoT there is a great opportunity for rendering IoT-enabled services in smart cities. Smart parking

sensor detects vehicles parked in the parking spaces. This information shown in dashboard will provide city tourists also able to park without any trouble.

4.5 PySpark

We are going to use Python with the spark as our work environment. We develop and deploy the application using Zeppelin. Python is easy to learn and a highly productive language where we can do more data analytics tasks quickly. Spark is cluster computing framework designed to handle large datasets. Spark aims to cover wide range of workloads in one place. It reduces the burden of maintaining different tools. RDD is a core data structure in Spark which provides in-memory computation for many general usages and also provides fault tolerance by making RDD coarse-grained. Pyspark is spark framework for Python language. PySpark can be started just like Spark. When PySpark shell opened, spark context created. Objects communicate with the cluster by using Spark Context. Spark, a framework designed for in-memory cluster computing. It mainly supports iterative algorithms. Spark allocate nodes automatically, with only small changes by the user in parametrization part. We implemented the traffic estimation algorithm in Spark

4.6 Feature Evaluation

The model we choose for this data is Unsupervised learning. Since there is no response variable to predict and not sure clusters types, it is the best option at this point of time. Clustering divides data into groups. With the help of predicted groups, we will be able to learn, how the traffic might be in future. Traffic flows are identified by three properties. Traffic

flow speed, traffic flow average time and traffic vehicle count. Traffic flow speed indicates the average velocity of vehicles on the road, in units of km/h. The traffic flow density, namely, the number of vehicles that the road contains. The traffic volume is defined as the number of vehicles traveling through a certain road section in unit time. This project utilize these three properties of traffic flows to find the traffic congestion situation. Steps fallowed.:

4.7 PRE PROCESS DATA.

- Load Meta data and Time variant data
- Remove NA's
- Merge both datasets by REPORT_ID
- Remove duplicate columns
- Convert string to Time stamp.
- Add column hourly which is extracted from timestamp.
- Convert date set to make average speed by hourly.
- Draws graphs to find patterns.

Both data sets need to be merged because Metadata contains geographical information. Road data contain time variant traffic information. In order to make clusters we need both information to be exists in one single dataset.

```
import pandas as
import numpy as np
from __future__ import division,
print_function
#Loading Meta Data
meta_df =
sqlContext.read.format("com.databricks.
spark.csv").option("header",
```

"true").load("/Users/jyothi/Desktop/capstone/meta/trafficMetaData.csv")

Meta data loaded as PySpark data frame. data frames, are an abstraction for tables in Python. Data frames support operations similar to relational tables, and also expose them as functions in a procedural language First data loaded into spark as data frame objects. Spark Data Frame objects can be parallelized for distributed parallel processing. Data types converted from string to float and timestamp. Dataframe API similar to Pandas Dataframe.

```
meta_df = meta_df.drop('extID')

meta_df = meta_df.drop('_id')
```

#Loading Data

```
road_df = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load("/Users/jyothi/Desktop/capstone/*.csv")
```

#Merge Both DF

```
merged_df = road_df.join(meta_df, (road_df.REPORT_ID == meta_df.REPORT_ID) ).drop(meta_df.REPORT_ID)
```

Both data frames contain REPORT_ID which will be used to merge both datasets.

```
rd_df = road_merged_df.select([c for c in road_merged_df.columns if c not in

{'DURATION_IN_SEC','DISTANCE_IN_METERS','POINT_2_NAME'}])
```

Converting String to TimeStamp

```
format = "yyyy-MM-dd'T'HH:mm:ss"

rd_df = rd_df.select('avgMeasuredTime','avgSpeed','TIMESTAMP','vehicleCount','REPORT_ID','POINT_2_LNG','POINT_1_LAT','POINT_2_LAT','POINT_1_LNG', from_unixtime(unix_timestamp('TIMEST
```

AMP',format)).cast("timestamp").alias('date'))

When we load the data, timestamp column is in string format. We need to convert that to timestamp, for further data processing.

Aggregate Hourly data

from pyspark.sql.functions import hour, mean

rd_data_sel = rd_data_selected.select('avgSpeed','POINT_2_LNG','POINT_1_LAT','POINT_2_LAT','POINT_1_LNG', F.hour('date').alias('hour'))

By converting the data frame into timestamp we will be able to abstract data by hour, minute, or day of the week for all kinds of temporal data distributions.

Round data

from pyspark.sql.functions import col, unix_timestamp, round

#rd_data_sel.groupBy("hour").agg(mean("avgSpeed").alias("meanSpeed"),round((mean("vehicleCount").alias("meanveh")).show()),2)

rd_data_set = rd_data_sel.groupBy('hour','POINT_1_LAT' , 'POINT_1_LNG','POINT_2_LAT','POINT_2_LNG').agg(mean('avgSpeed').alias("mean"))

rd_data_set =rd_data_set.withColumn("mean", round(rd_data_set.mean, 3))

rd_data_set =rd_data_set.withColumn("POINT_1_LNG", round(rd_data_set.POINT_1_LNG, 3))

rd_data_set =rd_data_set.withColumn("POINT_2_LNG", round(rd_data_set.POINT_2_LNG, 3))

rd_data_set =rd_data_set.withColumn("POINT_1_L

AT", round(rd_data_set.POINT_1_LAT, 3))

rd_data_set =rd_data_set.withColumn("POINT_2_LAT", round(rd_data_set.POINT_1_LAT, 3))

Data needs to be rounded for a fewer decimal places otherwise clusters might not prove proper results.

root
|-- hour: integer (nullable = true)
|-- POINT_1_LAT: double (nullable = true)
|-- POINT_1_LNG: double (nullable = true)
|-- POINT_2_LAT: double (nullable = true)
|-- POINT_2_LNG: double (nullable = true)
|-- mean: double (nullable = true)

Hour: Hour of the day extracted from timestamp.

Mean: Average of average speed group by hourly.

POINT_1_LAT: Latitude of street starting point.

POINT_1_LNG: Longitude of street stating point

POINT_2_LAT: Latitude of street ending point

POINT_2_LNG: Longitude of street ending point

4.8 FORMATTED DATA SET.

This data set much cleaner and concise. It contains much smaller foot print compared to original data sets. The below figure shows how the sample data for clustering looks like in Zeppelin.

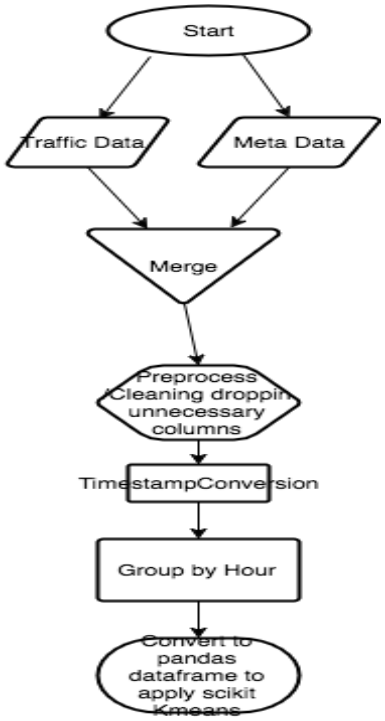

Figure 4: Explains process fallowed, to convert raw data into spark Data Frame object.

Above figure summarizes steps in preprocessing the data.

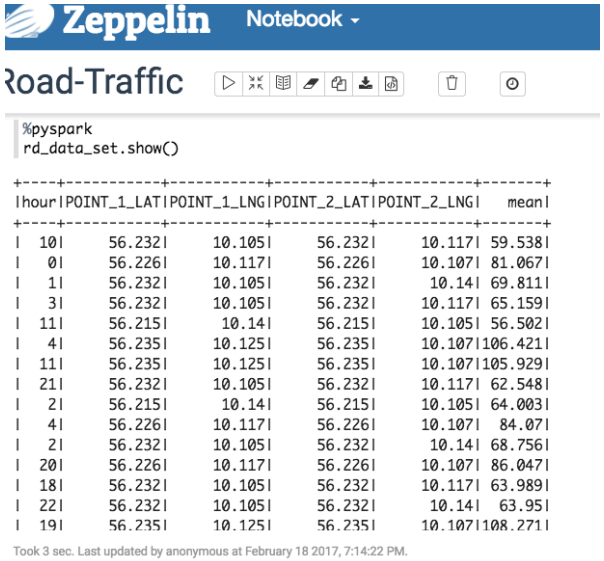

Figure 5 Data set snap shot

Once preprocessing done we start visualizing data to find patterns.

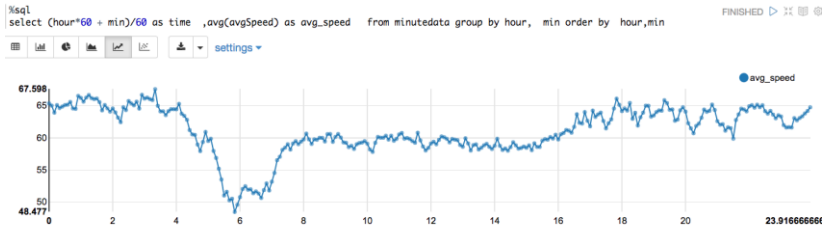

Figure 6 Time series graph on complete data

- Fig 3 shows time series graph for sample data. We can observe, difference in traffic speed for day and night.

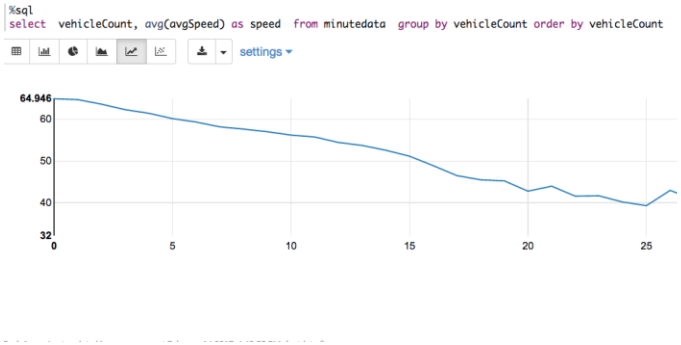

Figure 7 Data aggregated to hourly.

- Fig 4 Shows there is a strong relationship between number of vehicles in the street with in five minutes time period and average speed. We can eliminate either one of the variable to build a model.
- is a heat map. X-axis represents hour of the day. Y-Axis represents Vehicle count. Each cell is for Average speed. By observing we identify clearly, in a day average speed, vehicles travelled in the street are more compared to night.
- Based on research, attributes which are unnecessary are dropped out. Once we preprocess the data, our final feature set looks as fallows.
-

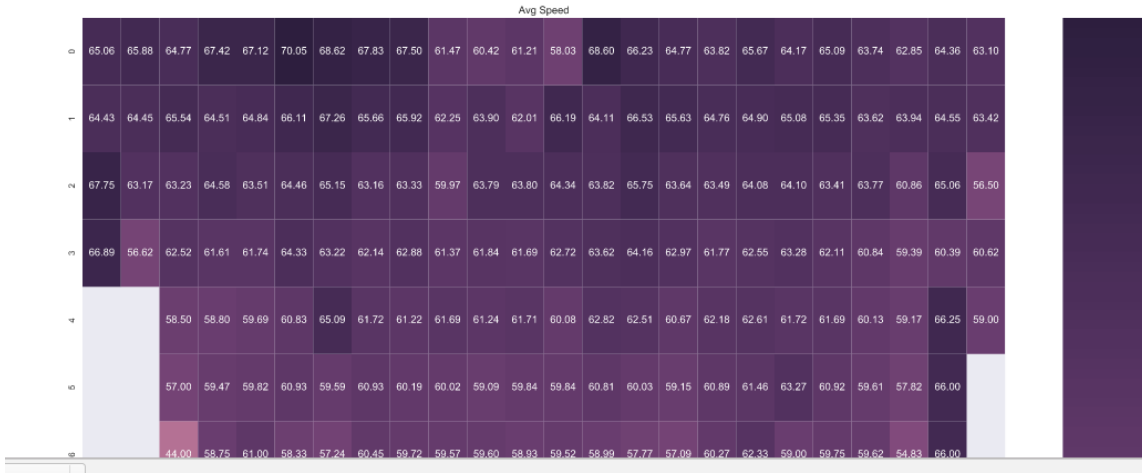

Figure 8 Heat map. Each cell represents average speed. Darker cells with average speed higher than lighter cells. Horizontal axis for number of vehicle's. Verticals time in 24 hour format.

4.9 SKLEARN K-MEANS

K-Means is a popular method for clustering. Clustering methods are

preferred when there is no dependent variable to be predicted but attributes divided into separate areas. The clusters hopefully will represent same characteristics. The data points assigned to the cluster should have a strong resemblance among them. This section describes K-means algorithm. First we will have to choose number of clusters K. One way to find out best value, by running K-means for number of K values and know which K has minimum squared errors. It is the sum of squared distance all points from the centroid. This method is called Elbow method. When we draw a graph between Squared errors to some clusters if there is no significant drop in errors even if we increase K, it represents best K as value. When we apply K-means algorithm it ran iteratively and calculates best cluster representation. In each iteration sum of squared distances from centroids will be calculated and minimized. By doing this, a new centroid determined, and another iteration starts. This process repeats until there is no change in centroid positions. Algorithm converge when it found a best cluster.

```
data_for_clustering = rd_data_set.toPandas()
#del data_for_clustering['REPORT_ID']
#data_for_clustering =
data_for_clustering.drop('TIMESTAMP')
data_for_clustering_matrix =
data_for_clustering.as_matrix()
# investigate alternative numbers of clusters
using silhouette score
silhouette_value = []
k = range(2,20)  # look at solutions between 2
and 20 clusters
for i in k:
    clustering_method = KMeans(n_clusters = i ,
random_state = 9999)
clustering_method.fit(data_for_clustering_matri
x)
```

labels = clustering_method.predict(data_for_clustering_matrix)

silhouette_average = silhouette_score(data_for_clustering_matrix, labels)
silhouette_value.append(silhouette_average)

Right now we are applying sklearn K-means on pandas Data Frame. Future implementation will be using Spark Mlib when we build model based on complete set of data.

K-Means is one of the fastest clustering algorithms. The Silhouette value calculated by using mean within cluster distance x and the mean nearest cluster distance y for each data point. The Silhouette value for a point is (y - x) / max(x, y). y is the distance between a data point and the nearest cluster that the point is not a part of. This function returns the mean Silhouette Coefficient over all samples. The least value is -1 and highest value is 1. 1 indicates the best value and value closer to zero indicate clusters are overlapped. When the points assigned to wrong groups, the values are Negative. In our case when K value 2 or 3 the Silhouette Coefficient value 0.68. It shows that we might consider taking two or three clusters.

Took 0 sec. Last updated by anonymous at February 18 2017, 8:31:07 PM.

```
%pyspark
list(segments)
```

```
[(0,      hour  POINT_1_LAT  POINT_1_LNG  POINT_2_LAT  POINT_2_LNG    mean  cluster
1       0       56.226       10.117       56.226       10.107  81.067        0
9       4       56.226       10.117       56.226       10.107  84.070        0
11     20       56.226       10.117       56.226       10.107  86.047        0
21     12       56.226       10.117       56.226       10.107  84.519        0
25     16       56.226       10.117       56.226       10.107  86.575        0
26     10       56.226       10.117       56.226       10.107  83.775        0
27      2       56.217       10.107       56.217       10.125  90.584        0
29      1       56.217       10.107       56.217       10.125  92.398        0
37     22       56.226       10.117       56.226       10.107  81.359        0
```

5 RESULTS

We used the vehicle trajectory data to identify the traffic problems along with a road. Because data volume is big, Apache Spark with python framework

was used to develop this project. To calculate congestion-related traffic bottleneck issues, we used traffic speed, road geographical and time series data. Performance measures such as annual delay per hour, congestion cost, and the Travel Time Index are produced from this analysis and were used to rank the congested segments across. The traffic congestion identified by some spatial and temporal threshold. When there is no data in the hot-spot area, we can say there are two possibilities. One possibility is that the congestion is so severe that no vehicle can enter. Two there are no cars at that time. The profile of congestion can be derived by analysing deeply operational data.

We implement the method on part of the real-world road network. The results generated surely show these traffic problems are not only consistent with experimental observation, but also provide useful insights. We considered congestion when the travel time exceeds a certain threshold.

In this paper, the traffic congestion state of roads is divided into five time zones (not four clusters), namely, night 12pm - 6am,6am - 10am, 10am - 4 pm, 4 pm - 9pm, 10 pm - 12 pm. According to the definition of traffic congestion, traffic congestion related to certain parameters (such as traffic volume and traffic speed) and also includes many factors. There are two different cases when the vehicle's speed is zero. It may be because traffic blocked by too many vehicles on the road that cannot move or it may also because the street is smooth, with no vehicles driving on the road. By this, a less traffic flow can match two states: the normal flow of heavy traffic or less traffic. However, comprehensive analysis of the three variables of traffic flows (traffic flow velocity, traffic flow density and

traffic volume) can reflect the real situation concerning traffic congestions.

The data is a subset of the complete dataset by average speed less than 20 km/hr and its only 10% of data. Complete data combined, there 18 million rows and 32 columns in this dataset. After preprocessing aggregating the dataset contains Latitude and longitude, hours in 24hour format, day of a year and mean velocity within that period. Dataset divided into different time zones get ready for clustering. Since data is now smaller footprint we plan to use K-Means using these R libraries readr, dplyr,data.table,sqldf,ggplot2,ggmap,maps.We will describe each package one by one. Library 'readr' for reading files and parsing data. Packaged dplyr, data.table, sqldf are provide easy to use data manipulation tasks. Ggplo2, ggmap, maps libraries provide a way to plot geospatial cluster data into map.

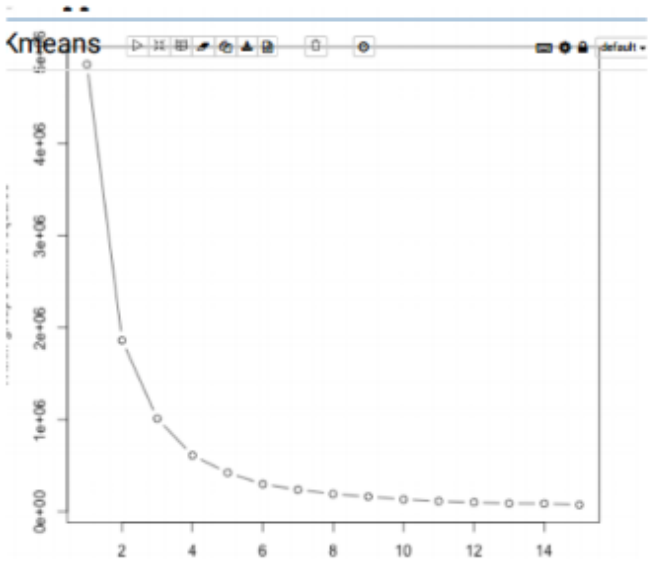

Figure 9 Plot of mean square errors with number of K PLOTTED IN x-axis. The errors drop significantly provide evidence for best K- value.

We started A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters.

Figure 10ZONE -1 Timezone 1AM TO 6 AM. Clusters are scattered in and out of city. Reason might be with less traffic rather than traffic ongestion.

The map shows many points are within the city, closer to each other. In this map, we have taken three clusters which will give right amount mean squared error. And then we started applying K-means model for each time zone. Traffic congestions vary from zone to zone.

Figure 11 Zone 2 Time zone 6AM TO 10 10 AM. Possibility of traffic congestion.

This is because of the fact that before the peak hour, some areas have comparatively higher demand than other areas, mainly the residential communities

rather than the major roads. This can also be observed from Figs, where the number of clusters is presented along with the time domains.

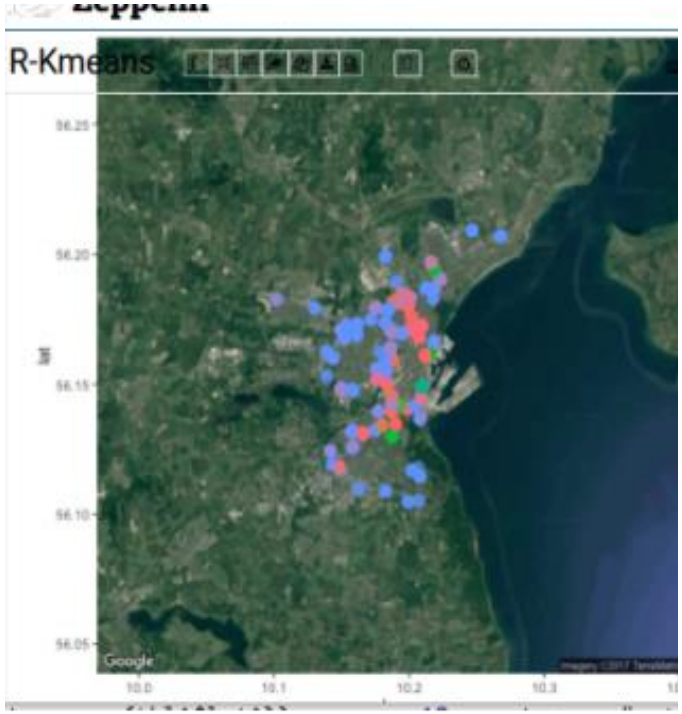

Figure 12 Zone -3 Time zone between 10AM to 3PM. Points are closer indicates traffic congestion with in city.

At the first time domain, i.e. 00:00~06:00,due to the points are all over the city. But it does not mean that the congestion is severe at that moment. But these may be because of low traffic.

Figure 13Zone 4 Time Zone 4PM TO 9PM. Traffic not as close as previous time zones.

Each filtered data with velocity less than 20 km/hr. We partition a large number of

data into different subsets or groups so that the requirements of homogeneity and heterogeneity are clearly explained. Correlation(homogeneity) requires that information in the same cluster should be as similar as possible and heterogeneity means that data in different groups should be as diverse.

Figure 14 Time zone 10 PM to 1 AM . Traffic DISTRIBUTION SIMILAR TO TIMEZONE -1. Reason may be both are in night time.

In Zones 2, 3, 4 clusters are more concentrated in one area. There a might be a chance that cause for this is more are less by traffic. We have to investigate further whether this road jams because of traffic congestion or not.And we have to examine in time zone night 10 PM to morning 6 AM there might less traffic. We need to identify, what is the cause for slower traffic speed. And we also have to distinguish points in location wise which areas are more concentrated with traffic congestions.

6 CONCLUSION

Last few decades, there has been increasing in sensor integration into city traffic. Traffic cameras GPS from vehicles and radar are shared data sources in our daily lives. In addition to these smartphones can act as sensors.

Cell phone carries cell location data every day in the entire city. Citizens can use this data from these sources able to make better travel decisions. Vehicles currently equipped with advanced technology traffic conditions can be exchanged and extract vehicle locations. In much Data-rich transportation systems, sensors collect user information and analysed. When all these advancements are properly used with advanced tools like Spark, Hadoop which will provide so much chance to improve further in future. Also utilising advancements data science improvements in model building in areas such as Machine Learning and Datamining also open doors improvement in citizen's life. Going further smart city concept is one of many such use cases in next decades. In this project, we tried to implement Machine learning algorithm like K-means. Results can be more accurate with a complete analysis of the dataset. Right now we used Sklearn K-means. Spark has machine learning implementation API called Mlib. Using Mlib doing clustering might seems to be the right choice at this point. Because spark automatically runs algorithms in a distributed way. There good support to spark with other cloud computing world like Amazon AWS. Combining these together makes application scales well in future even with tremendous grown in data as well.

Thus generally congestion first begins from bottlenecks and then spread around the nearby areas. Sometimes the process is called cascading failures.

Such daily road traffic network can be checked experimentally from traffic flow data. Exploration of the network traffic flow can provide insights understanding to the management of road network. While static bottlenecks have gained

enough attention and many research results, dynamic network bottlenecks also need to addressed in ongoing studies. Reconstruction of bottleneck requires dynamic traffic flow data, which cannot be in very good time scales such as hourly data, nor on small scales such as seconds, where computation and visualisation are the significant problems. Some questions are still unanswered: how to identify the bottleneck of the urban street network.

7 REFERENCES

1. Ali, Muhammad Intizar, Feng Gao, and Alessandra Mileo. "Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets." *International Semantic Web Conference*. Springer International Publishing, 2015.
2. Ali, Muhammad Intizar, Feng Gao, and Alessandra Mileo. "Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets." *International Semantic Web Conference*. Springer International Publishing, 2015.
3. Tönjes, Ralf, et al. "Real time iot stream processing and large-scale data analytics for smart city applications." *poster session, European Conference on Networks and Communications*. 2014.
4. Kolozali, Sefki, et al. "A knowledge-based approach for real-time iot data stream annotation and processing." *Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), IEEE*. IEEE, 2014.
5. Compton, Michael, et al. "The SSN ontology of the W3C semantic sensor network incubator group." *Web semantics: science, services and agents on the World Wide Web* 17 (2012): 25-32.
6. Anicic, Darko, et al. "EP-SPARQL: a unified language for event processing and stream reasoning." *Proceedings of the 20th international conference on World wide web*. ACM, 2011.
7. Barbieri, Davide Francesco, et al. "C-SPARQL: SPARQL for continuous querying." *Proceedings of the 18th international conference on World wide web*. ACM, 2009.

8. Jin, Jiong, et al. "An information framework for creating a smart city through internet of things." *IEEE Internet of Things Journal* 1.2 (2014): 112-121.
9. Khan, Zaheer, Ashiq Anjum, and Saad Liaquat Kiani. "Cloud based big data analytics for smart future cities." *Proceedings of the 2013 IEEE/ACM 6th international conference on utility and cloud computing*. IEEE Computer Society, 2013.
10. Stutz, Christiane, and Thomas A. Runkler. "Classification and prediction of road traffic using application-specific fuzzy clustering." *IEEE Transactions on Fuzzy Systems* 10.3 (2002): 297-308.
11. Tesema, Tibebe Beshah, Ajith Abraham, and Crina Grosan. "Rule mining and classification of road traffic accidents using adaptive regression trees." *International Journal of Simulation* 6.10-11 (2005): 80-94.
12. [1] W3C Semantic Sensor Networks Incubator Group (SSN-XG). http://www.w3.org/2005/incubator/ssn/.
13. [2] J. Andrea. Envisioning the next-generation of functional testing tools. Software, IEEE, 24(3):58–66, 2007.
14. [3] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic. Ep-sparql: a unified language for event processing and stream reasoning. In Proceedings of the 20th international conference on World wide web, pages 635–644. ACM, 2011.
15. [4] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-sparql: Sparql for continuous querying. In Proceedings of the 18th international conference on World wide web, pages 1061–1062. ACM, 2009.
16. [5] P. Barnaghi, S. Meissner, M. Presser, and K. Moessner. Sense and sens ability: Semantic data modelling for sensor networks. In Conference Proceedings of ICT Mobile Summit 2009, 2009.
17. [A. Bifet, G. Holmes, B. Pfahringer, J. Read, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. Moa: a real-time analytics open source framework. In Machine Learning and Knowledge Discovery in Databases, pages 617–620. Springer, 2011.
18. [7] C. Bisdikian, R. Damarla, T. Pham, and V. Thomas. Quality of information in sensor networks. In 1st Annual Conference of ITA (ACITA07), 2007.
19. [8] A. Bolles, M. Grawunder, and J. Jacobi. Streaming sparql-extending sparql to process data streams. In The Semantic Web: Research and Applications, pages 448–462. Springer, 2008.
20. G. De Francisci Morales. Samoa: a platform for mining big data streams. In Proceedings of the 22nd international conference on World Wide Web companion, pages 777–778. International World Wide Web Conferences Steering

Committee, 2013.

21. M. Fischer and R. Tonjes. Generating test data for black-box testing using genetic algorithms. In Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on, pages 1–6. IEEE, 2012.
22. F. Ganz, P. Barnaghi, and F. Carrez. Information abstraction for heterogeneous real world internet data. 2013.
23. P. Godefroid, M. Y. Levin, D. A. Molnar, et al. Automated whitebox fuzz testing. In NDSS, volume 8, pages 151–166, 2008.
24. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. ACM SIGKDD Explorations Newsletter, 11(1):10–18, 2009.
25. G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy. Smart objects as building blocks for the internet of things. Internet Computing, IEEE, 14(1):44–51, 2010.
26. D. Kuemper, E. Reetz, and R. Tonjes. Test derivation for semantically described iot services. In Future Network and Mobile Summit (FutureNetworkSummit), 2013, pages 1–10. IEEE, 2013.
27. D. J. Russomanno, C. Kothari, and O. Thomas. Sensor ontologies: from shallow to deep models. In System Theory, 2005. SSST'05. Proceedings of the Thirty-Seventh Southeastern Symposium on, pages 107–112. IEEE, 2005.
28. S. Beswick, ''Smart cities in Europe enabling innovation,'' Osborne Clarke, London, U.K., Tech. Rep., 2014. [Online]. Available: http://www.cleanenergypipeline.com/Resources/CE/ResearchReports/Smart%20cities%20in%20Europe.pdf
29. M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, ''Smarter cities and their innovation challenges,'' Computer, vol. 44, no. 6, pp. 32–39, Jun. 2011.
30. T. W. Mills. (Dec. 2015). Intel Corporation—Intel Labs Europe: Open Innovation 2.0. [Online]. Available: http://dspace.mit.edu/handle/ 1721.1/99033
31. Puiu, Dan, et al. "Citypulse: Large scale data analytics framework for smart cities." *IEEE Access* 4 (2016): 1086-1108.