**Smarty city Applications –Aarhus city Road data**

**Naga Jyothi Kunaparaju**

**Saint Peter’s University** [**nkunaparaju@saintpeters.edu**](mailto:jkunaparaju@gmail.com)

**1-347-341-3307**

# 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 functionality and provide the solution in a user-friendly way.

# 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.

# 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

Meta Data:

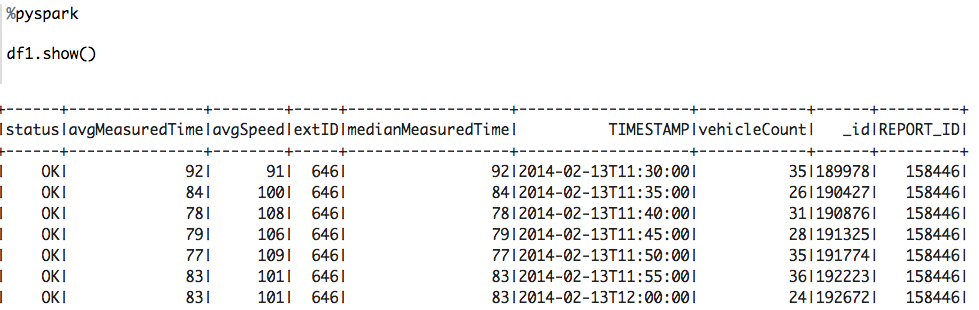
* POINT\_1\_STREET
* DURATION\_IN\_SEC
* POINT\_1\_NAME
* POINT\_1\_CITY
* POINT\_2\_NAME
* POINT\_2\_LNG
* NDT\_IN\_KMH
* POINT\_2\_POSTAL\_CODE
* POINT\_2\_COUNTRY
* POINT\_1\_STREET\_NUMBER
* ORGANISATION
* POINT\_1\_LAT
* POINT\_2\_LAT
* POINT\_1\_POSTAL\_CODE
* POINT\_2\_STREET\_NUMBER
* extID
* ROAD\_TYPE
* POINT\_1\_LNG
* REPORT\_ID
* POINT\_1\_COUNTRY
* DISTANCE\_IN\_METERS
* REPORT\_NAME
* RBA\_ID
* \_id

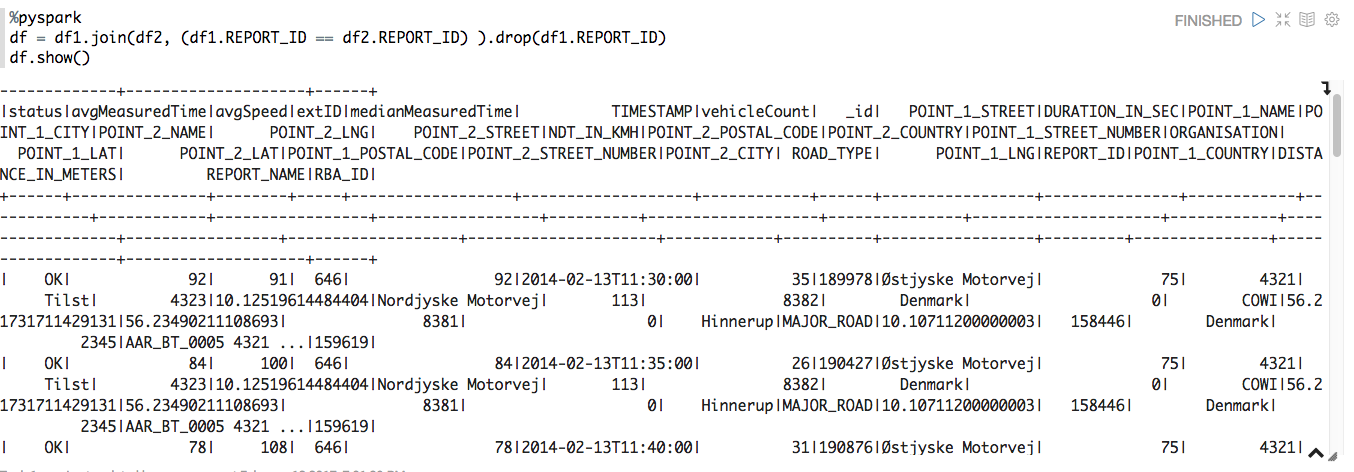
Data:

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.





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.

# Method:

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

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.

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.

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.

Architecture

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

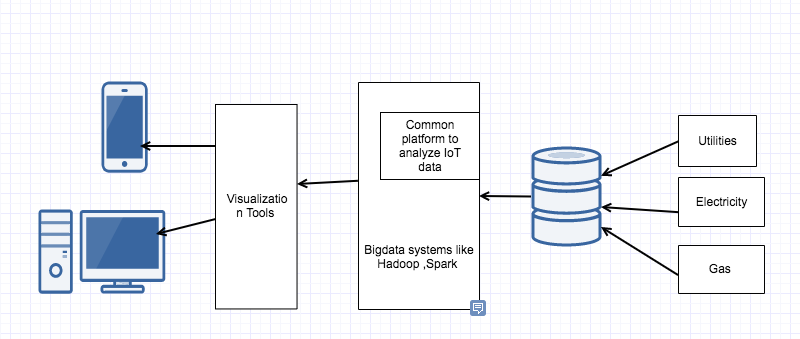


Fig:1

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.

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

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.

Steps fallowed.:

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('TIMESTAMP',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\_LAT", 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

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.



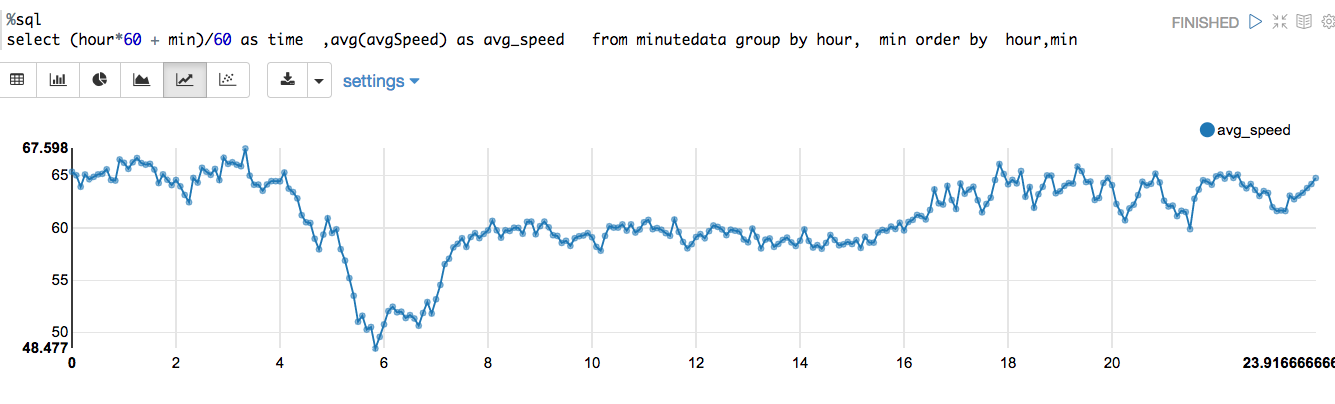
**Fig5**: Explains process fallowed, to convert raw data into spark Data Frame object.

Above figure summarizes steps in preprocessing the data.



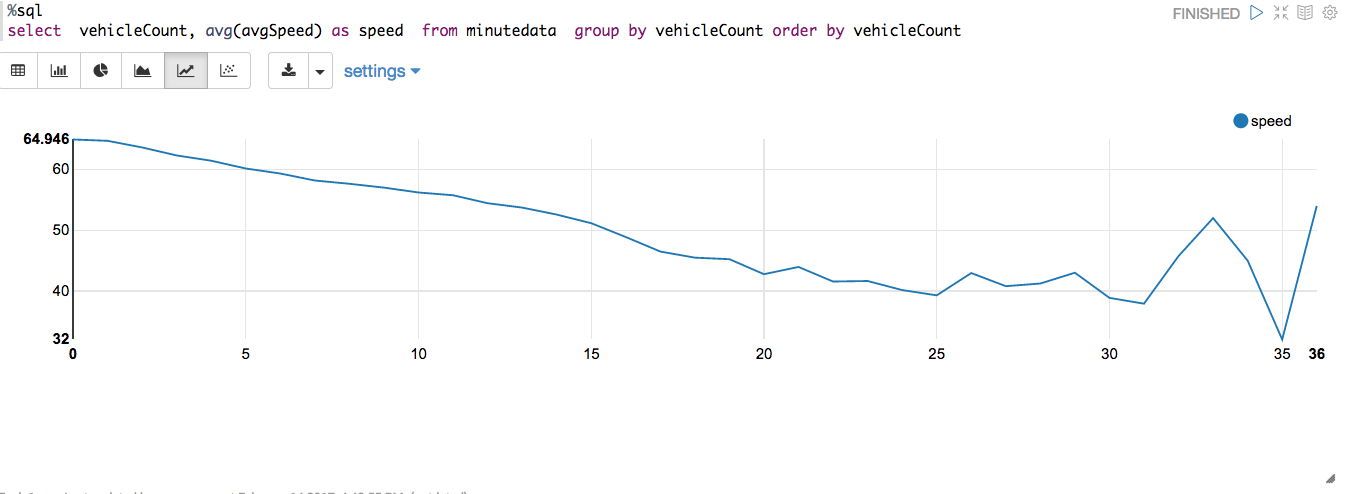
**POINT\_2\_LNG**: Longitude of street ending point

Once preprocessing done we start visualizing data to find patterns.



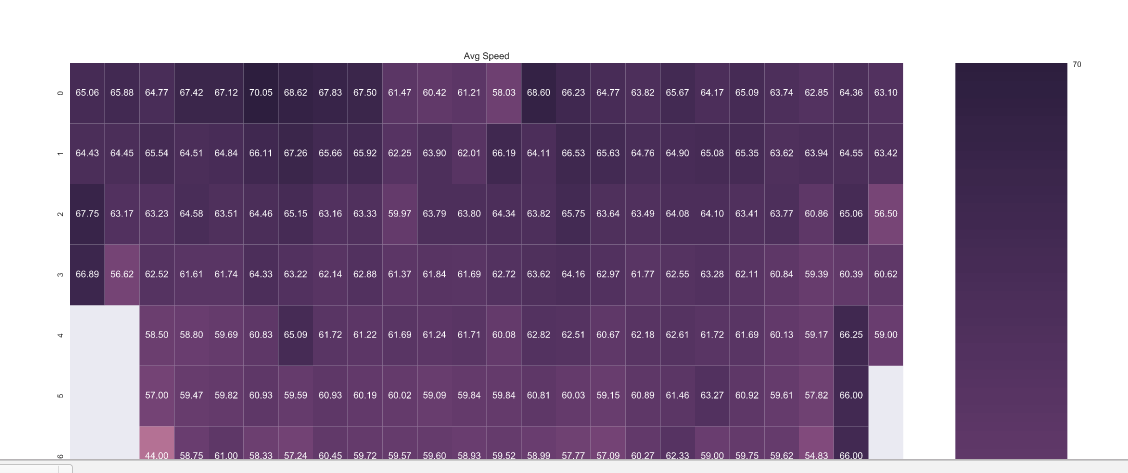
**Fig:1**

* Fig 1 shows time series graph for sample data. We can observe, difference in traffic speed for day and night.



**Fig:2**

* Fig 2 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.



**Fig :3**

* Fig 3 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.

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\_matrix)*

*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.

**

# 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 Kmeans. 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.

# 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.