1. APA-style reference to your IEEE or ACM journal article. Make sure that your article was published in a journal and not in Proceedings.

2. Create a table with 2 columns. Indicate the heading of left column as 'Competitor' and the heading of the right column as 'My algorithm'

3. In the following row, summarize the results of the article in the left column, and the results of your algorithm in the right column

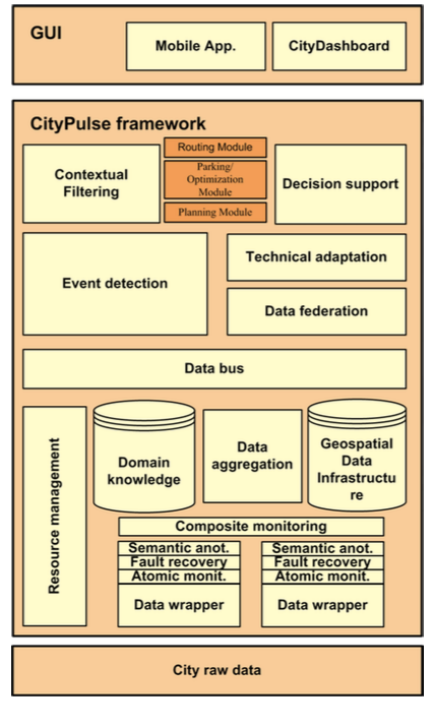
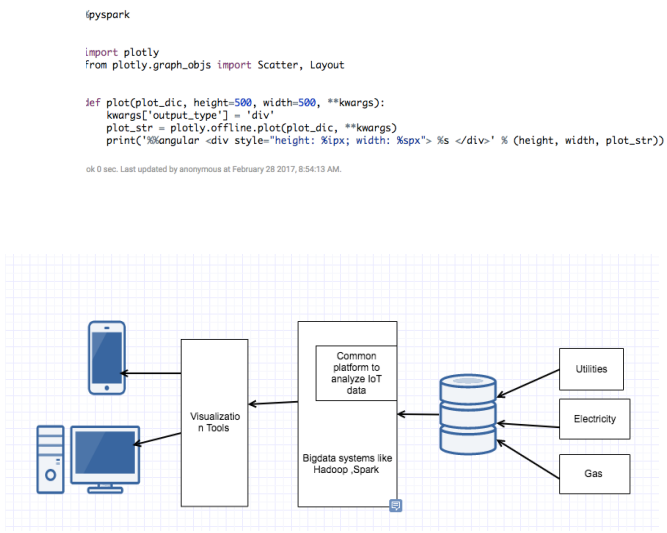
4. In the following 10 rows, compare the article's results and your results **quantitatively**. using 10 different metrics. Explain how you obtained your measurements.

Competitor	My Algorithm

The proposed framework has been demonstrated in the paper by a smart Travel Planer application. The main contributions of this work include integrating of heterogeneous data streams, providing interoperability, quality analysis, (near-) real-time data analytics and application development in a scalable framework.	Since we are using Road dataset, we focus on the Smart city Dashboard application. Road data consists of two different types of data files.Metadata and traffic data.
	My algorithm concentrates mainly on historical data rather than streaming data and structure them
	`%pyspark`
	`import timeit`
	`start = timeit.timeit()`
	`road_df1 = sqlContext.r`
	`road_df2 = sqlContext.r`
	`road_df3 = sqlContext.r`
	`end = timeit.timeit()`
	`print(end - start)`
	`0.001297248003538698`
	19 Million records to process including meta data.

The next step was to configure the *Data aggregation* component to use the Sensor SAX algorithm as aggregation method for the traffic and parking observations.
(r_1) related_city_event(Id) :- filtering_event(Category), city_event(Id, Category, Source).
(r_2) 1 <= {selected_city_event(EventId) : related_city_event(EventId), not expired_event(EventId)} <= 1.
(r_3) value(RankEleName,Value):- selected_city_event(EventId), ranking_city_event_data(EventId, RankEleName,Value).
(r_4) value_with_ranking_type(RankingElementName, M):- value(RankingElementName,Value), ranking_multiplier(RankingElementName,Int), M = Value*Int.
(r_5) sum(C) : value_with_ranking_type("EVENT_LEVEL",Value1), value_with_ranking_type("DISTANCE",Value2), C = Value1+Value2.
(r_6) criticality(C) :- C = M/100, sum(M).
(r_7) critical_city_event(EventId,C):- selected_city_event(EventId), criticality(C).
Listing 3. Logic rules for contextual filtering of events with ranking through linear combination.
	For aggregating data, we are using spark libraries. Using spark Data frame objects aggregating data is much faster than earlier big data applications. It's very fast almost in 5 to 6 sec to union all datasets.
	`%pyspark`
	`import timeit`
	`start = timeit.timeit()`
	`road_df1 = road_df1.unionAll(road_df3)`
	`road_df1.count()`
	`end = timeit.timeit()`
	`print(end - start)`
	`-0.004915146011626348`
	Took 0 sec. Last updated by anonymous at April 07 2017, 11:49:24 AM.
	`%pyspark`
	`rd_data_s.count()`
	`214859`
	Took 42 sec. Last updated by anonymous at April 07 2017, 11:50:10 AM.

All the City Pulse framework components are deployed on a back-end server and are accessible via a set of APIs. As a result of that the application developer has only to develop a user-friendly front-end application, which	Our backend is Apache spark, HDFS, Python Pandas, Numpy, Scipy, MLIB, Sci–Kit learn.For front end we use Zeppelin to create components which is flexible enough to communicate with HTML front. Deployment is much easier and scalable. Spark,

calls the framework	Zeppelin makes rapid development and deployment. Befit from using advanced technologies makes application scales well.
Architecture overview of City Pulse project  FIGURE 1. The components of the CityPulse framework with their APIs.	Architecture of My project 
This scenario aims to provide travel-planning solutions, which go beyond the state of the art solutions by allowing users to provide multi-dimensional requirements and preferences such as air quality, traffic conditions and parking availability. In this way the users receive parking and route recommendations based on the current context of the city. In addition to this, *Travel Planner* continuously monitors the user context and events detected on the planned route. User will be prompted to opt for a detour if the real time conditions on the planned journey do not meet the user specified criteria anymore.	Our target is to predict traffic problems (Congestions) based on historic data. By predicting the congestions, we will be able to detour travelers accordingly. But the difference is, we use only historic data. Another goal of this application is to build analytics platform for domain experts. We are trying to implement Time series model and visualizing the results to understand patterns. Using statistical models explanatory and Predictive analysis which makes deep understanding revealing the patterns of traffic problems
Over sequence diagram of City Pulse project	Sequence diagram of my model

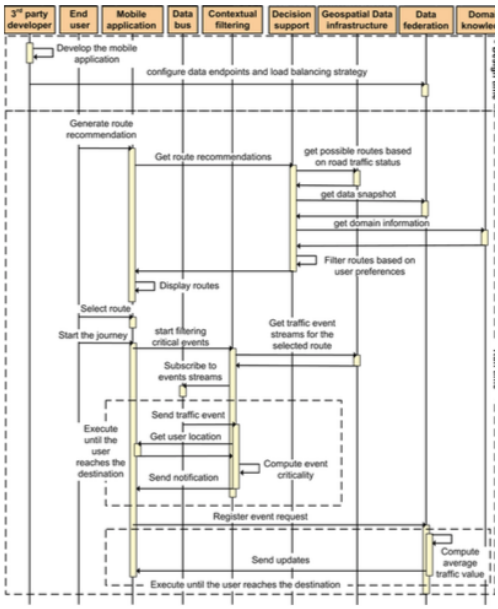
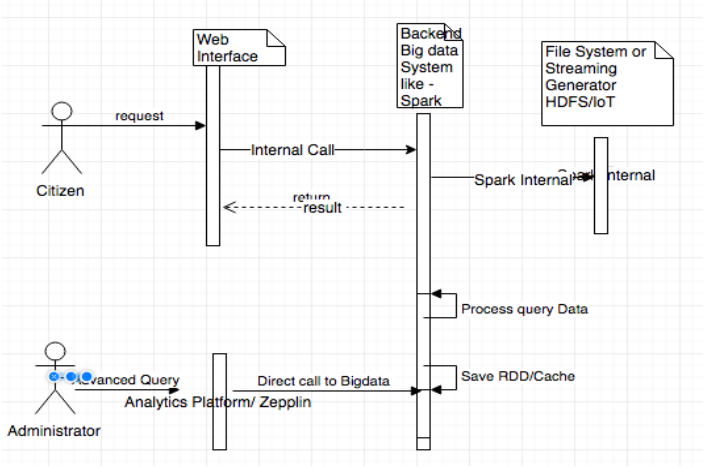


Figure 17. Real-time Adaptive Urban Reasoning workflow for Travel Planner.



For this application, and very probably for most of the City Pulse enabled applications, the framework components can be divided in two different categories. First of all, the large scale data stream processing modules are configured and deployed in order to permanently monitor the status of the Aarhus city. In this way, when a new user logs into the mobile application and generates a request for a recommendation, the back-end application can provide the answer based on the current traffic or pollution situation from the city.

Our application divided into three areas. First one is Data aggregation. Visualizing aggregated data will make us understand the necessity of doing more analysis. Second part of analysis is to create geographical clustering on data and plot these clusters into map to find the patterns from those. And also doing time series analysis might also reveal some more patterns or trends in traffic congestions.
Final part of the application is to build an user interface for users will be able authenticate themselves to the application and by sending their criteria will be able to get maps for their travel plan.

For example, in a travel planner scenario, a reduced list of all possible routes to go from A to B within a geographical area is provided by the *Geospatial Data Infrastructure* component. This does not imply any contextual reasoning, qualitative optimization, or event-driven adaptation, which is instead provided by the *Decision support* component. It basically a sql query optimization on relational data.

My model implementing K-Means clustering technique to group data which will improve accuracy of the results.
We are not implementing any contextual filtering for in this project.

```
%r
tbl1<-subset(data1, data1$hr_grp == 1)
#tbl1<-subset(tbl, data1$mean > 50)
result4<-kmeans(tbl1,3)
tbl1$cluster <- result4$cluster
#View(tbl1)
tbl1$cluster <- as.factor(tbl1$cluster)
```

```
1. input_get_routes(SP, EP, V, 5) :-
    parameter("STARTING_POINT", SP),
    parameter("ENDING_POINT", EP), route_costMode(V).
2. route(@get_routes(SP, EP, V, N)) :- input_get_routes(SP, EP, V, N).
3. route_data(@get_routes_data(SP, EP, V, N)) :- input_get_routes(SP,
   EP, V, N).
4. parameter("STARTING_POINT","10.116919 56.226144").
5. parameter("ENDING_POINT","10.1591864 56.1481156").
6. minimize{AV@2 : valueOf("DISTANCE", AV)}.
7. minimize{AV@1 : valueOf("TRAVEL_TIME ", AV)}.
```

Listing 5. A snapshot of logic decision support rules for the Travel Planner scenario.

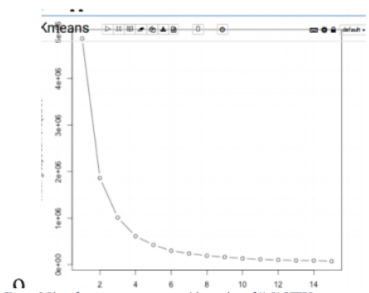


Figure 9 Plot of mean square errors with number of K PLOTTED IN x-axis. The errors drop significantly provide evidence for best K- value.

The ability to record and store historical (cleaned and summarized) data for post-processing makes it possible to analyze the status of the city not only on the go but also at any point in time, enabling diagnosing and ‘‘post mortem’’ analysis of any incidents or relevant situation that might have occurred.

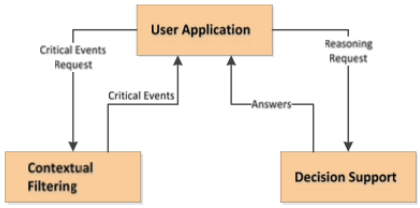
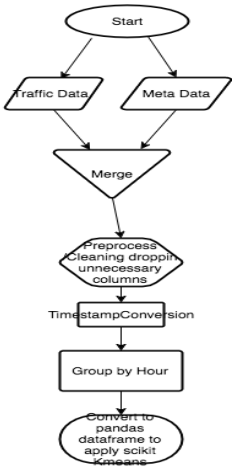


Figure 12. Event-based user-centric decision support.

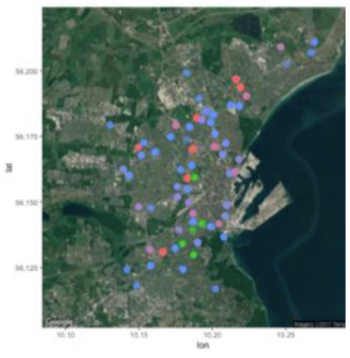
Data flow diagram from raw data to cleaned up to feed to model.



To facilitate that, a dashboard for visualizing the dynamic data of the smart cities is provided on top of the City Pulse framework. Based on this dashboard, the user has the possibility to visualize a holistic and summarized view of data across multiple contexts or a detailed view of data of interest, as well as to monitor the city life as it evolves and as things happens.

My application will allow users choose daily or hourly aggregation and find the traffic congestions. Model will run an algorithm with allow to fallow different distant and similarity metrics to fins best cluster clusters

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. Results obtained are shown below.



Data aggregation using SensorSAX with minimum window size is 1 and sensitivity level is 0.3 for average speed observation of Aarhus traffic data stream.

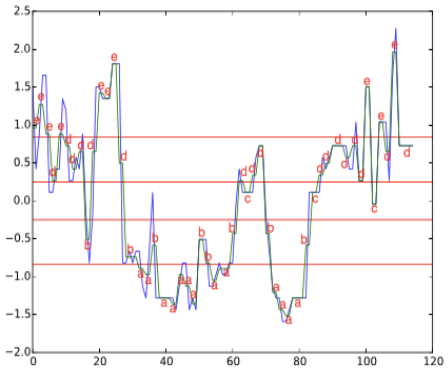


Figure 16. Data aggregation using SensorSAX with minimum window size is 1 and sensitivity level is 0.3 for average speed observation of Aarhus traffic data stream.

My model will allow experts to run analytics like for example heat map of hourly data, Time series for daily or monthly data.

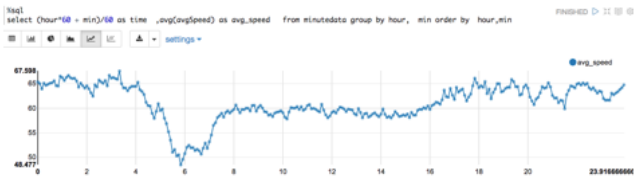


Fig:1

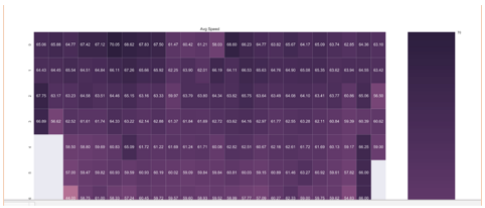


Fig :3

User Reference: uniquely identifies the user that made the request. Such reference is related to user credentials that will be used in the final to manage user logins and instances of the City Pulse framework in different cities. The occurrence of such relevant events is notified to the *Contextual Filtering* component by the *Event detection* component, which provides additional metadata describing the event.

City Pulse DashBoard

Tryng to implement user authentication module rather than having everything in one page try to develop each individual page for different scopes of use.