Final Manuscript

Vehicle Traffic, Provided by City of Aarhus in Denmark

Presented by:	Amit Pandey (in collaboration with Kapil Bastola)
Guided by:	Professor Sylvain Jaume
Course:	DS670 - Capstone

Abstract

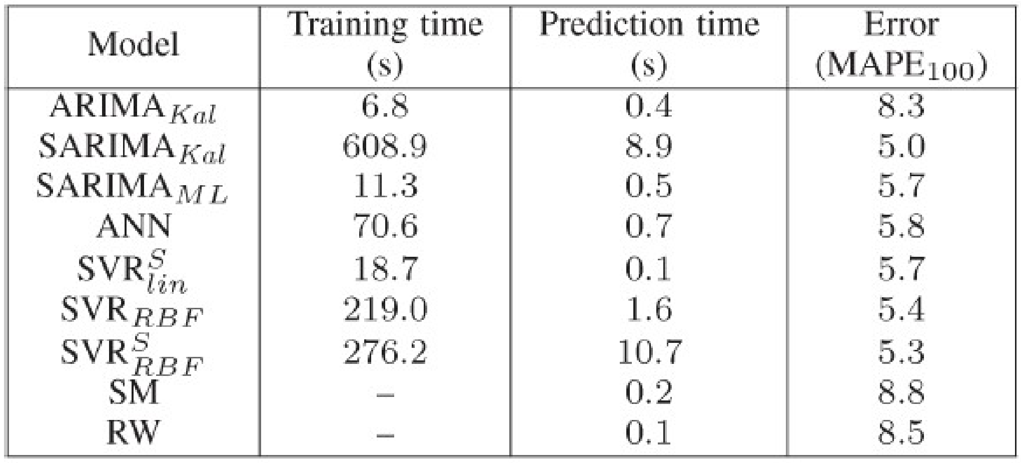
The urban traffic congestion is transforming into an epidemic all over the world. The brisk increase in vehicle traffic has become one of the critical problems faced by cities all over the world. As a result of the constant traffic congestion, transportation cost has increased significantly due to all the time wasted on the road and the corresponding fuel cost. This proposal looks at how the Vehicle Traffic data collected can be used in making Aarhus a smart city in terms of traffic guidance and management. City Pulse project services smart city solution, by means of interpreting big data from Internet of Things and social networks. This paper is presenting a model on how to predict the traffic based on historical data. The data which has been used on this paper is belongs to a smart city on EU which called CityPulse. The model which has been used here is a time series model (ARIMA), and this model has been used to predict the traffic based on the speed as measurement. The higher speed means the lower traffic and the lower speed means the higher traffic. The result was focusing on predicting the traffic on daily and hourly basis. We were able to predict the traffic based on historical data quite accurately.

Work by competitor:

The paper we've selected presents an experimental view of the different statistical & machine-learning approach to short-term traffic-flow forecast. They follow the approach of SARIMA and have proposed 2 new SVR models using a seasonal kernel to determine the similarity with the time-series examples. The results that they present confirm that seasonality is the key feature to achieve a high-accuracy. Though, the more accurate models usually require high computational resources – both while the training and prediction phase. Therefore, the seasonal kernel approach may be a reasonable compromise between the forecast accuracy and computational complexity. The SARIMA version that doesn't include a Kalman filter and the ANNs performed worst than SVR with an RBF kernel. This in-turn is less accurate than seasonal kernel variant. Furthermore, another important direction of the research paper that has been indicated by the experimental results presented in this paper consists of investigating the covariate shift in traffic.

The competitor have presented an extensive experimental review of many statistical and machine learning approaches to short-term traffic flow forecasting. Following the approach in SARIMA, they proposed two new SVR models: employing a seasonal kernel to measure similarity between time-series examples. They presented results confirm that seasonality is a key feature in achieving high accuracy; however, the most accurate models often require high computational resources both during the training phase and at prediction time. For this reason, they presented that seasonal kernel approach might be a reasonable compromise between forecasting accuracy and computational complexity issues. In particular, while SARIMA employed in combination with the Kalman filter ends up being the best model on average, the competitors proposed approach is particularly competitive when considering predictions during highly congested periods. The SARIMA version that does not include a Kalman filter and the ANNs perform consistently worse than SVR with an RBF kernel, which, in turn, is less accurate than the seasonal kernel variant. The competitor result show that the accuracy of the Seasonal Mean (SM) predictor starts degrading when the temporal distance between training and test set grows too much, and for the other predictors, no further improvement is observed when using larger training sets including past months, which are too distant from prediction time.

Even though the competitor have used seven different models, their MAPE is coming out to be much higher, as shown below in table 1:



Model	Training time (s)	Prediction time (s)	Error (MAPE_{100})
ARIMA_{Kal}	6.8	0.4	8.3
SARIMA_{Kal}	608.9	8.9	5.0
SARIMA_{ML}	11.3	0.5	5.7
ANN	70.6	0.7	5.8
SVR^{S}_{lin}	18.7	0.1	5.7
SVR_{RBF}	219.0	1.6	5.4
SVR^{S}_{RBF}	276.2	10.7	5.3
SM	–	0.2	8.8
RW	–	0.1	8.5

Table 1: TRAINING AND PREDICTION TIME FOR ALL THE ALGORITHMS,AVERAGED ON THE 16 NODES CONSIDERED IN THE TEST SET

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation.

Contribution

We did the time-series analysis using ARIMA (Autoregressive integrated moving average) model. ARIMA is a generalization of an autoregressive moving average (ARMA) model, both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step can be applied one or more times to eliminate the non-stationarity.

We predicted the next 5 hours (data points) for the time-series using ARIMA and compared them with the actual.

Data

The dataset assigned for the capstone project is the “Vehicle Traffic, Provided by City of Aarhus in Denmark” from CityPulse dataset collection. A collection of datasets of vehicle traffic, observed between two points for a set duration of time over a period of 6 months (449 observation points in total). The data is available in raw (CSV) and semantically annotated format using the citypulse information model. This proposal looks at how the Vehicle Traffic data collected can be used in making Aarhus a smart city in terms of traffic guidance and management.

A collection of datasets of vehicle traffic in the city called Aarhus in Denmark, observed between two points for a set duration of time over a period certain months (449 observation points in total). The data is available in raw (CSV) and semantically annotated format using the Citypulse information model. There are total four (4) different datasets over different durations available:

1st data set:	February 2014 to June 2014
2nd data set:	August 2014 to September 2014
3rd data set:	October 2014 to November 2014
4th data set:	July 2015 to October 2015

The data is available in raw (CSV) and semantically annotated format using the citypulse information model. For this project, I'm going to use the ".csv" format only as it's easy to manipulate and analyze the data. The data analysis will be done in Apache Zeppelin environment that is being taught in the class. To test out the data, I took the first dataset from February 2014 to June 2014 and saved all the files in a folder "traffic_feb_june" in my Downloads directory. Using Zeppelin Spark, I was able to consolidate all the files in one table called roadtraffic. Using SQL table, I ran a few queries to look into the data set and get a summary. I have only looked at the first dataset (Feb 2014 – June 2014) for now to do the analysis.

The data has the following variables:

"Status": The first column "status" has "OK" in all the 20 rows in the SQL query I ran. After running the distinct value in that column, I noted that there was no other values and the only unique value in that column is "OK". This could potentially mean that the value is meant to represent as a check if there's data collected correctly in that row and "OK" denotes as a "Yes".

"avgMeasuredTime": The second column "avgMeasuredTime" seems to have all numerical values. The two sensors on two purposes of the street measure to what extent it took a vehicle in seconds to achieve the second point from the principal point. This field probably gives us the mean of aggregate time taken in seconds by various vehicles to reach from the main indicate the second point for every data reading.

"avgSpeed": The third column "avgSpeed" also seems to comprise of numerical quantities. The column seems to tell the "average" speed in kilometer per hour (kmh) of the vehicles between the two measurement points.

"extID": The fourth column "extID" seems to contains a 3 and 4 digit numerical value. At the first glance, it looked like a random number but closely looking at the distinct value, "extID" seems like a unique ID for the 449 files first as it contains a total of 449 distinct values in the total table.

"medianMeasuredTime": The fifth column "medianMeasuredTime" column additionally seem to comprise of numerical values and seems to have alike values as the "avgMeasuredTime" in the 20 rows I looked. This column seems to provide us the "median" of total time taken (in seconds) by the various vehicles between the two measured points in the reading.

“**TIMESTAMP**”: The sixth column “TIMESTAMP” seems to contain the data and time of each reading. This column could be helpful in determining which time and days have the frequency of traffic higher or lower.

“vehicleCount”: The seventh column “vehicleCount” seems to contain all numerical values which could potentially tell us the number of vehicles passing simultaneously between the two measured points. This could be another interesting parameter to look at as it would help us identify the flow of traffic during different time zones.

“**_id**”: The eighth column “id” seems to contain all numerical values. After looking at the 20 rows that I initially pulled through SQL, I could see that it contains the unique id number for each row. As the distinct values in the column seems to equal the row in the table, it seems like that “id” could be the unique identified.

“REPORT_ID”: The ninth and the final column also seem to contain all numerical values. After looking at the 20 rows that I initially pulled through SQL, it seems like it’s the same id for the data pulled. Looking at the distinct values, it seems to contain 449 total values representing a unique identifier for each file. Interestingly, the “Report_ID” is also a column in the “metadata” file which would be helpful in linking both the files

MetaData: The metadata file is a “.csv” file with 449 rows and 26 columns (variables). It seems to contain additional information about the vehicle traffic. As the file contains 449 rows, the same as the number of 449 files in the dataset, each row could correspond to one file in the dataset. This file seems to contain information as to the points of measurement in the dataset. It furthermore contains the information for data like: street, city, longitude, latitude, postal code, and country (all in Denmark). As also mentioned earlier, the file contains “REPORTID” variable which could be used to look up and match/merge the information with the dataset, if needed. This file has some information in Danish language so we may have to convert it in English to understand better.

Method: As explained in Figure 1, I plan to use the following method to collect, process and predict the datasets.

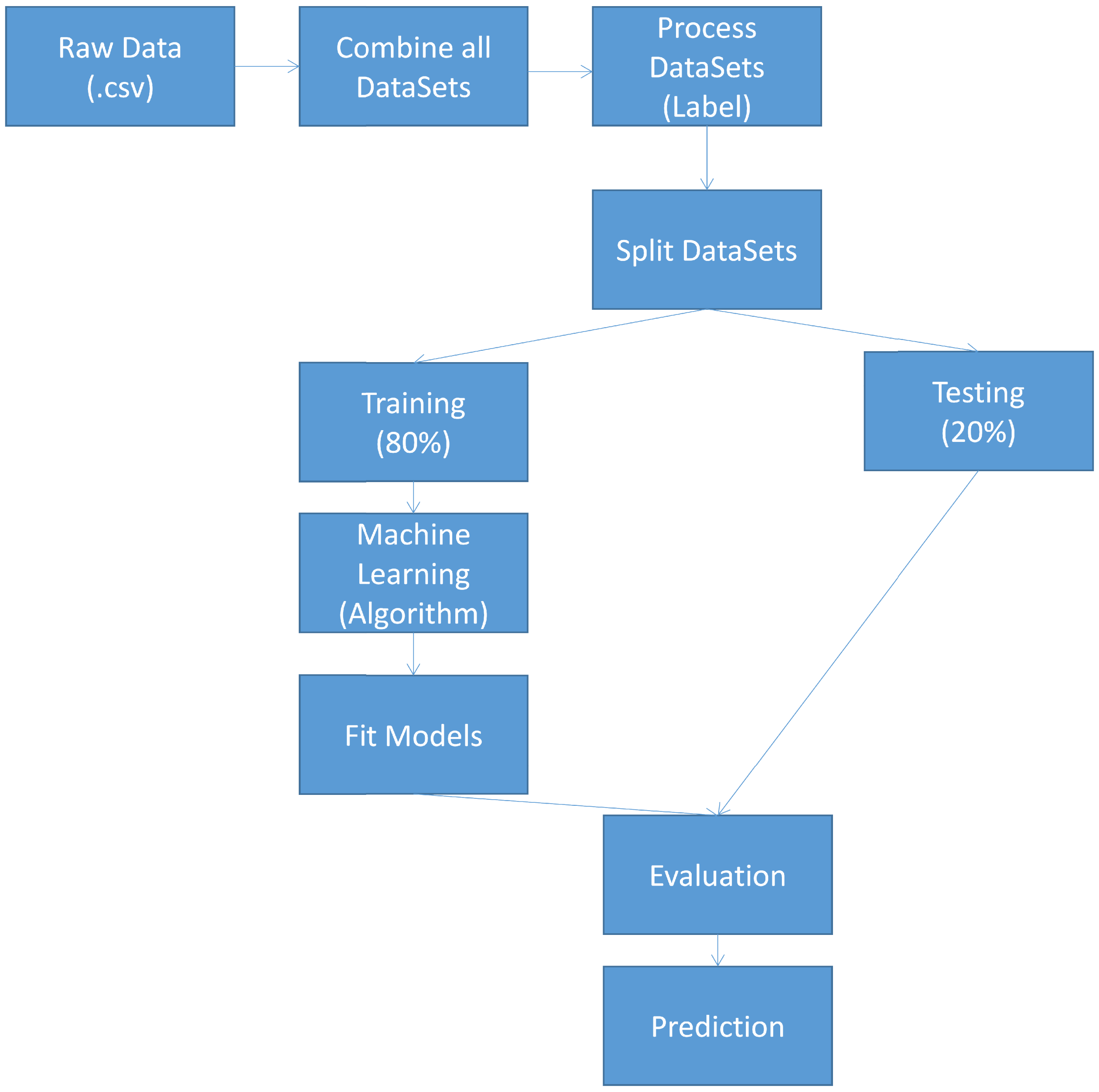


Figure 1 – Method

This is the Process we intended to follow:

1. **Collect and download the datasets into one folder.**
   Based on the data provided for Aarhus, we will download and collect all the datasets in one folder. As the data is available in the raw (CSV) and semantically annotated format using the Citypulse information model, we will use the raw (CSV) format to process and analyze the ata.

2. **Load data and combine all data sets in Zeppelin.**
   Using Zippelin, we will load and combine all the files (449) in the datasets to analyze the information.

3. **Process datasets and label combined data based on velocity.**

   Depending on the programming language (R, Python or SQL), we decide to use on Zeppelin, we may need to import some libraries to do the scientific calculations. We may need the libraries like "NumPy" in Python or "Caret" (short for Classification And Regression Training) in R to attempt to streamline the process for creating predictive models. These libraries contains tools for data splitting, pre-processing, feature selection, model tuning using re-sampling, variable importance estimation, .as well as other functionality.

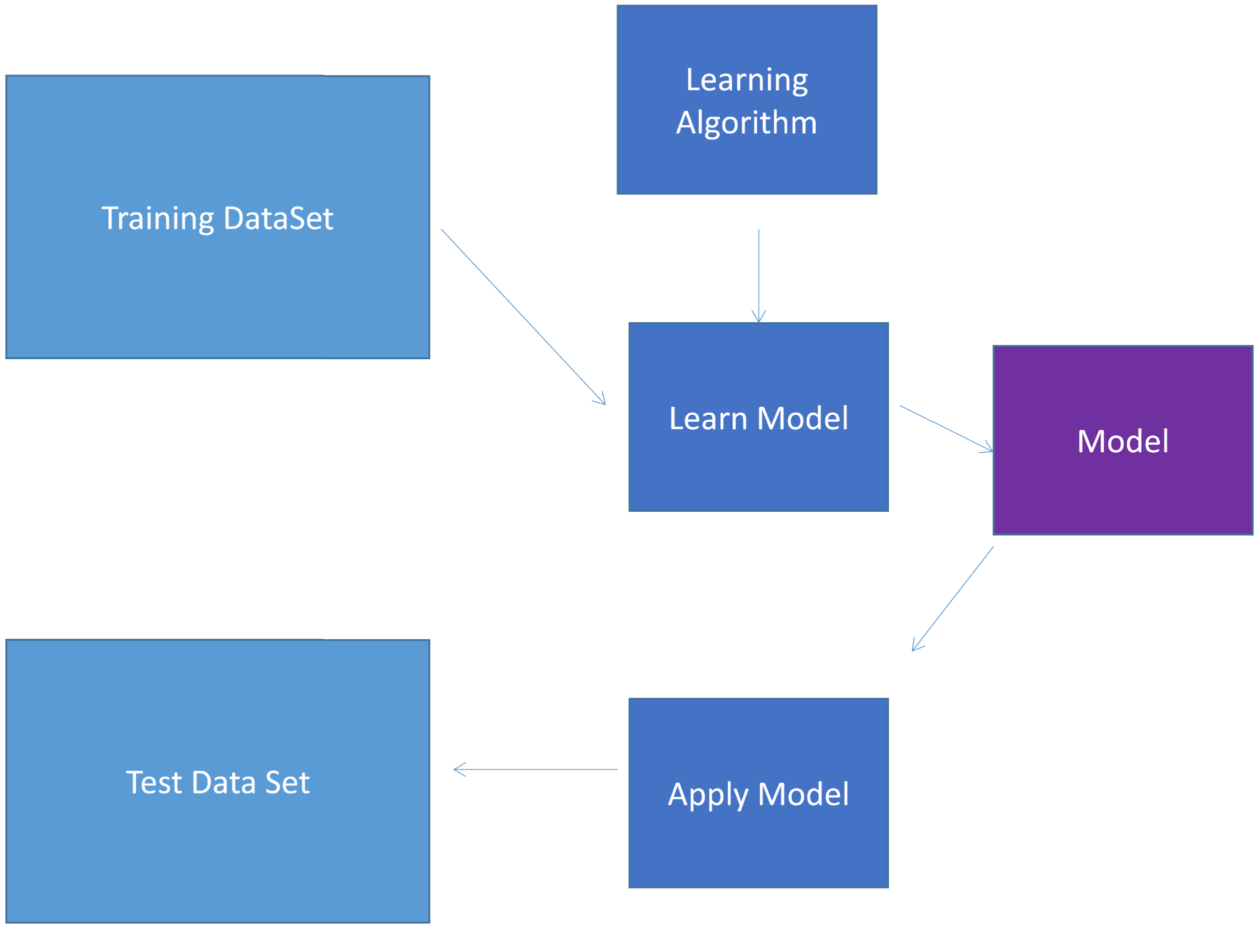
   We'll also do an analysis to see if the data needs to be cleaned and if there are any errors that may corrupt our analysis. We also want to make sure that the null values are set to zero and take care of missing values. We will check for common errors like: missing values, corrupted values, data range errors, etc. We will have to look through file rows and columns and sample test values to see if the values make reasonable sense.

4. **Split Datasets into Training (80%) and Testing Data (20%).**
   As we are going to do some prediction, it would be best to split our combined dataset into two different datasets: Training (80%) and Testing (20%). The Training dataset will be used to build and test different statistical mode and the Testing dataset will be used to evaluate (cross-validate) our model and assumption. We do the splitting and create the Test partition to provide us with a fair assessment of our prediction model build.

   There is no single method for selecting the extent of training and testing data. Some people use 90/10 and some prefer 80/20. In any case, doing as such can cause bias the classification results. For selecting the right split, we can use either "N-Fold cross validation" or "K-fold cross validation". In N-Fold cross validation, we randomize the data and create N equal partition and choose the Nth partition for testing dataset and N-1 partition for the training. In the training set, we can apply "K-fold cross validation" to validate and find the best parameter. This will take out any bias out of our assumptions.

5. **Apply machine learning algorithm and fit models based on the algorithm.**
   Next we'll have to take a look to employ an algorithm model that will best fit the relation among the attributes and class label the data. The model created by the algorithm must fit both the training data and also accurately predict the test data.

Figure 2: Model for building a classification model.



We will use different classification algorithms to classify the data such as:

KNN:

K closest neighbors (KNN) is a basic algorithm that stores all accessible cases and characterizes new cases in view of a closeness measure. KNN has been utilized as a part of measurable estimation and example acknowledgment as of now in the start of 1970 as a non-parametric strategy. KNN is a non parametric lazy learning calculation. When we say a strategy is non parametric, it implies that it doesn't make any assumption on the data distribution. This is quite helpful, as in practical world, the greater part of the practical information does not comply with the ordinary hypothetical suppositions made.

It is additionally a lazy algorithm which means it doesn't utilize the training data to do generalize. Basically, there is very little training phase. This implies the training data is really quick. Absence of speculation implies that KNN keeps all the training information. All the more precisely, all the training information is required amid the testing stage. This is different than algorithm techniques like SVM where you can discard of all non-support vectors with no issue.

Random forest decision tree:

Random forests or random decision forests are a group learning technique for classification, relapse and different undertakings, that work by developing a huge number of decision trees at preparing time and yielding the class that is the method of the classes (classification) or mean forecast (relapse) of the individual trees. Random decision forests adjust for decision trees' propensity for over fitting to their training set. Forests develops numerous classification trees. To classify another question from an information vector, put the info vector down each of the trees in the timberland. Each tree gives a classification, and we say the tree "votes" for that class. The backwoods picks the classification having the most votes (over every one of the trees in the timberland).

Support Vector Machines (SVM) model:

SVMs also support vector networks are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Margin/support to make a group for the training data. A SVM is a discriminative classifier also defined by a separating hyper plane. In other words, in supervised learning, given labeled training data, the algorithm outputs an optimal hyper plane that can categorize new examples.

N**eural Networks:**

Neural networks, sometime also suggested as connectionist systems, are a computational approach, which relies on upon a broad gathering of neural units or fabricated neurons, openly exhibiting the way a natural cerebrum deals with issues with far reaching clusters of normal neurons related by axons. Each neural unit is related with various others, and associations can actualize or inhibitory in their effect on the incitation state of related neural units. Each individual neural unit may have a summation limit which merges the estimations of each one of its wellsprings of information together. There may be an edge limit or confining limit on each affiliation and on the unit itself: to such a degree, to the point that the banner must beat the most distant indicate before causing diverse neurons. These structures are self-learning and arranged, rather than unequivocally altered, and surpass desires in domains where the plan or highlight acknowledgment is difficult to express in a standard PC program.

6. **Evaluate the models on the Testing Data: Cross validate**

When we are done with building of the different algorithms we need to decide which algorithm gives us the best result. In other words, we need to evaluate our models. We use the model on the test dataset to get the prediction from the model and compare them against the expected labels that we had in hand. There are many performance metrics that help us evaluate the performance of our models.

For binary classification, we can look at the number of true positives (tp), which are the correct affirmative predictions, true negatives (tn), which are the correct negative predictions, false positives (fp), which are the incorrect affirmative predictions, and false negatives (fn), which are the incorrect negative predictions. Using these values we can calculate different measures of model performance like accuracy (tp + tn)/ (p + n), precision (tp / (tp + fp)), recall (tp / (tp + fn)), specificity (tn / (fp + tn)), fall-out (fp / (fp + tn)), F1 score (2 * tp / (2*tp + fp + fn)), etc.

We can also look at a receiver operating characteristic curve or ROC curve to evaluate model performance. ROC curve can be plotted for different models to visualize which model is performing the best. The x-axis of the ROC curve is false positive rate while the y-axis is true positive rate. The ROC curve is hence the recall as a function of fall out. We look at area under the curve (AUC) to calculate the probability that a model will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

Using these measures we can evaluate which models performs best on the test dataset to come up with a final model to be used as the prediction model.

We will do a cross validation to check the accuracy of the training model. Cross-validation, sometimes called rotation estimation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. The aim of the cross validation is also to avoid over-fitting of the model.

A confusion matrix is a table that is frequently used to depict the execution of a classification model (or "classifier") on an arrangement of test information for which the genuine qualities are known. The confusion matrix itself is generally easy to see, however the related phrasing can be confusing. Here’s an example of confusion matrix:

n=100	**Predicted: NO**	**Predicted: YES**
Actual: NO	40	10
Actual: YES	15	35

Here are the basic terms used in the confusion matrix:

- **True positives (TP):** These are cases in which we predicted yes (for example, when they have the disease), and they do have the disease.
- **True negatives (TN):** We predicted no, and they don't have the disease.
- **False positives (FP**): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **False negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Even though a confusion matrix will provide the details needed to figure out how well the classification model works, we can figure out the summary of the information with a single number to make it convenient to compare the performance of different models, through the following measures:

- **Accuracy:** Number of correct prediction/total number of predictions to determine the correctness of the classifier. It can be calculated as:
    - (TP+TN)/total = Accuracy
- **Misclassification Rate:** To determine how often the model is wrong:
    - (FP+FN)/total = Misclassification rate.
    - Also, can be determined by 1 minus Accuracy
    - It is also called the "Error Rate"
- **Precision:** When the model predicted yet, how often is that correct.
    - TP/predicted yes = Precision
- **Recall/True Positive Rate:** When the model actually says "yes", how often does it predict yes?
    - TP/actual yes = True Positive Rate:
    - This is also known as the "Sensitivity" or "Recall"
- **False Positive Rate:** When the model returns no, how often does it predict "yes".
    - FP/actual no = False Positive Rate
- **Specificity:** When it's actually no, how often does it predict no.
    - TN/actual no = Specificity
    - Also, equals 1 minus False Positive Rate
- **Prevalence:** How often does the yes condition actually occur in our sample.
    - actual yes/total = Prevalence.

The accuracy of the model can also be calculated through a few other measures like:

- **Null Error Rate:** Null error rate is how often we would be wrong if we always predicted the majority class. This can be a useful baseline metric to compare our classifier against.
- **F Score:** F-score is the weighted average of the true positive rate (recall) and precision.
- **ROC Curve:** This is a commonly used graph that summarizes the performance of a classifier over all possible thresholds. It is generated by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis) as you vary the threshold for assigning observations to a given class.

7. Make a conclusion/prediction.

Based on the confusion matrix and the evaluation measurers (as defined above), we are going to make a model prediction on the accuracy of the model.

Method:
We loaded the data into Zeppelin to clean and analyze it with the idea of stacking the data into the framework and clean it/investigate it. The idea was to invest as much energy as it could to set up the data into a strong base of a domain that could be valuable and simpler to do the model. This way it would be easier to analyze it simply. Thus, in view of that, the data initially has been stacked into Spark, which did not work out as efficient language for the data and the purpose of our model. We, therefore, changed it over the data outline into a SQL as it simple to deal with excessively numerous records. This turned out to be great to just analyze the data. However, we still had to deal with a prediction and analysis part. In this way, we inferred that SQL won't work, therefore we have attempted to stack the data into a PySpark, which is a competitor language amongst Spark and Python, and that where the PySpark originated from. This was a decent decision notwithstanding, because of the absence of data and information in how to deal with the programming part on the grounds that the linguistic structure must conform to both language Spark and PySpark, and because of the long run time that we have confronted on doing the stacking and the analysis, then we have chosen to do only python for loading, clean, analysis and prediction the data and the model.

1. **Collect and download the datasets into one folder.**
   Based on the data provided for Aarhus, we will download and collect all the datasets in one folder. As the data is available in the raw (CSV) and semantically annotated format using the Citypulse information model, we will use the raw (CSV) format to process and analyze the ata.

2. **Load data and combine all data sets in Zeppelin.**
   Using Zippelin, we will load and combine all the files (449) in the datasets to analyze the information. We have utilized numerous languages to the analysis. We have utilized: Spark, SQL PySpark and Python. The languages that we used to fit to the data were both PySpark and Python, in any case we have chosen to run with Python as it was anything but difficult to deal with a sentence structure for one language as opposed to taking care of the structure for two languages. What's more, the outcome for that part, the run time stacking into PySpark took twofold the run time stacking into Python. As Python was more proficiency in this part, why have chosen to finish with python.

   Due to the vast data set we have chosen to take a major measure of the data to test and fabricate the model on it, as taking every one of the data won't be conceivable because of the capacities that we have. It is more than 29 million records and that without mapping and other data.

3. **Clean the data.**

We'll did an analysis to see if the data needs to be cleaned and if there are any errors that may corrupt our analysis. We also wanted to make sure that the null values are set to zero and take care of missing values. We checked for common errors like: missing values, corrupted values, data range errors, etc. We looked through the file rows and columns and sample test values to see if the values make reasonable sense.

We have utilized the cleaning techniques to clean the data of N/A by either supplanting or barring from the example or notwithstanding expelling from the data if the record has an excessive number of missing qualities. What's more, it can't withdraw or supplanted. Likewise, we have examined the data to ensure that the data is not tainted for instance if the data has an immense out of the sudden spike on the chart that implies the data is ruined. Or, on the other hand if the data has some wrong esteem, for example, a period of 99 that implies it is ruined as there are no 99 clock time and the time goes just to 24 hours.

4. **Handling the outliers:**

In order to handle the outliers, what we have done here is we have rejected the anomaly from the data, as this will influence the data as we didn't evacuate it or erase it, as this can be utilized on further analysis. For instance, utilizing the outliers to anticipate the possibility of having an accident on this street. Nonetheless, on this case we truly need to ensure that the outliers are a result of an accident not due to absent or undermined values, or might be a result of street work in the city which cannot be utilized at this stage, with the goal that we have chosen to avoid the outliers from the data and account for the typical data to improve a creation.

5. **Split Datasets into Training (80%) and Testing Data (20%).**

As we are going to do some prediction, it we split our combined dataset into two different datasets: Training (80%) and Testing (20%). The Training dataset was used to build and test different statistical mode and the Testing dataset was used to evaluate (cross-validate) our model and assumption. We did the splitting and create the Test partition to provide us with a fair assessment of our prediction model build.

There was no single method for selecting the extent of training and testing data. Some people use 90/10 and some prefer 80/20. In any case, doing as such can cause bias the classification results. For selecting the right split, we used "N-Fold cross validation" and/or "K-fold cross validation. This will take out any bias out of our assumptions.

6. **Mapping the data:**
   We have mapped the data into the Meta Data to have more data about the record that the framework caught and comprehend the data

7. **Final discovery:**
   We did the time-series analysis using ARIMA (Autoregressive integrated moving average) model. ARIMA is a generalization of an autoregressive moving average (ARMA) model, both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step can be applied one or more times to eliminate the non-stationarity.

Results:
We predicted the next 5 hours (data points) for the time-series using ARIMA and compared them with the actual. Here's the result of the model:

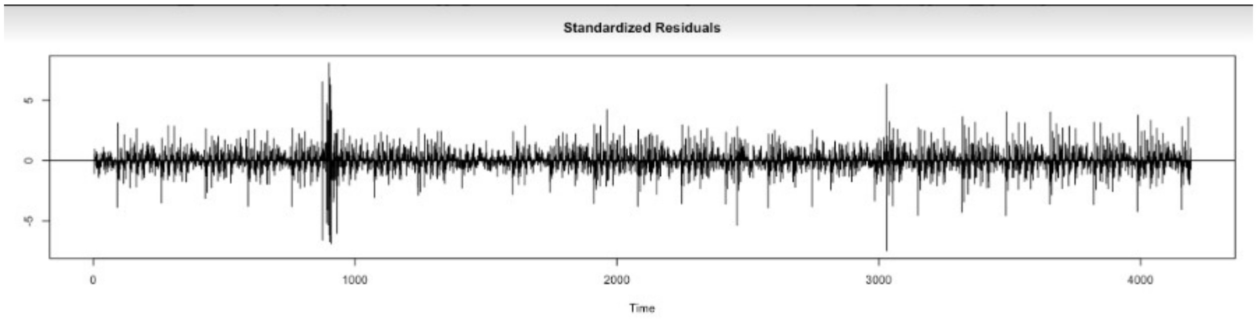


Chart 1:For the existing data, this is the residuals of the predicted fits.

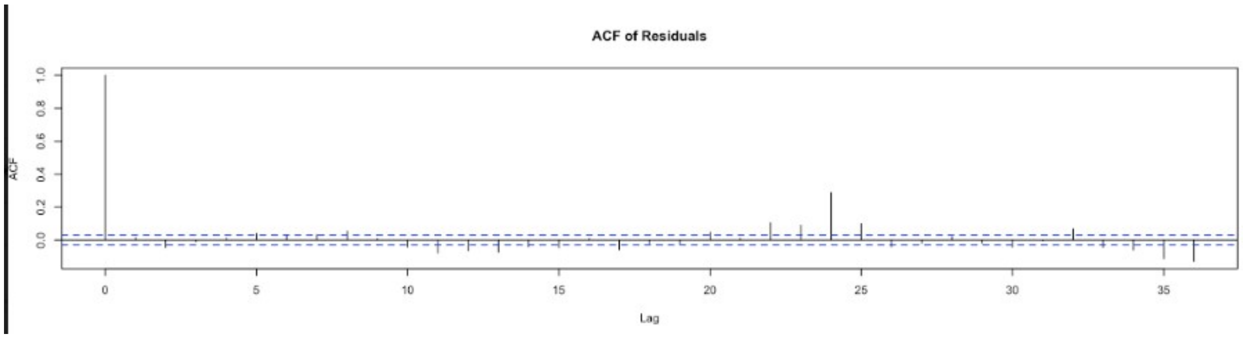


Chart 2: Autocorrelation function (ACF) of residuals shows the correlation of the residuals (as a time series) with its own lags.

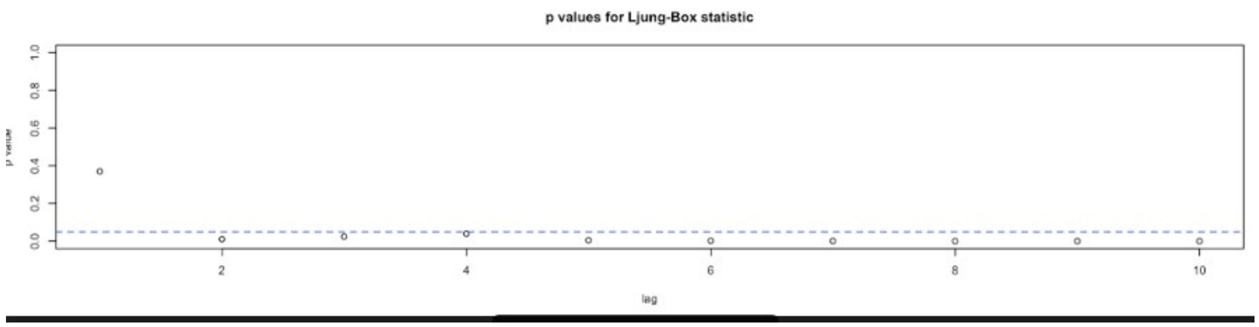


Chart 3: p-value showing the fitting of the model

Discussion:

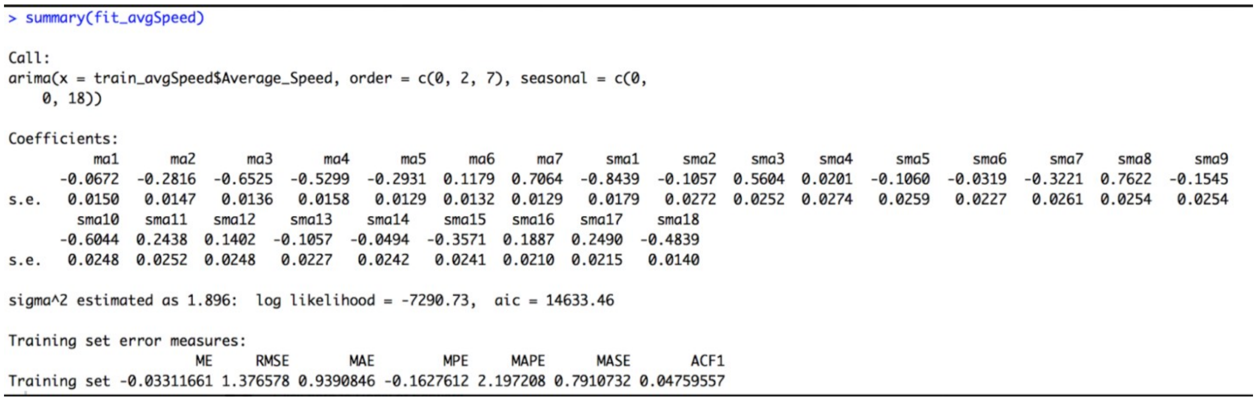
The project here has used the smart city data collected for building up a model to predict the traffic for some certain roads that are covered on the data within that smart city. The data was collected on monthly periods and was captured as a brief record so we need the meta data to unlock the data and make sense on it. We have loaded the data into the system then we have mapped the data into the meta-data and then we got the full information that we needed. The issue that we have faced is to get rid of the not applicable data fields that empty as well as the corrupted data that we faced. The corrupted data is more complicated than not applicable data, as we first need to identify what is corrupted and then decided whether to exclude or substitute. And an example of corrupted is when you see a speed of 600 miles per hour and these definitely means the data is corrupted as the is no way the speed could come to 600 miles per hour in certain roads, maybe all the roads. In order to handle the outliers, what we have done here is we have rejected the anomaly from the data, as this will influence the data as we didn't evacuate it or erase it, as this can be utilized on further analysis. Then we had to load the data into multiple stage then aggregated them tighter to get the full data set in one single frame. Therefore, based on that, we have built up a time series model for predicting the traffic on a giving road.

We did the time-series analysis using ARIMA (Autoregressive integrated moving average) model. ARIMA is a generalization of an autoregressive moving average (ARMA) model, both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step can be applied one or more times to eliminate the non-stationarity. We predicted the next 5 hours (data points) for the time-series using ARIMA and compared them with the actual.

Conclusion:

The conclusion for this paper has built a new model that produces and predicts the traffic on a giving road based on historical data. The model which has been used is a timer series model.

And the model was able to predict the needed information based on the historical data. Even though the competitor have used seven different models, their MAPE is coming out to be much higher, as shown before in “work by competitor”.



```
> summary(fit_avgSpeed)

Call:
arima(x = train_avgSpeed$Average_Speed, order = c(0, 2, 7), seasonal = c(0,
    0, 18))

Coefficients:
          ma1      ma2      ma3      ma4      ma5     ma6     ma7     sma1     sma2    sma3    sma4     sma5     sma6     sma7    sma8     sma9
      -0.0672  -0.2816  -0.6525  -0.5299  -0.2931  0.1179  0.7064  -0.8439  -0.1057  0.5604  0.0201  -0.1060  -0.0319  -0.3221  0.7622  -0.1545
s.e.   0.0150   0.0147   0.0136   0.0158   0.0129  0.0132  0.0129   0.0179   0.0272  0.0252  0.0274   0.0259   0.0227   0.0261  0.0254   0.0254
        sma10   sma11   sma12    sma13    sma14    sma15   sma16   sma17    sma18
      -0.6044  0.2438  0.1402  -0.1057  -0.0494  -0.3571  0.1887  0.2490  -0.4839
s.e.   0.0248  0.0252  0.0248   0.0227   0.0242   0.0241  0.0210  0.0215   0.0140

sigma^2 estimated as 1.896:  log likelihood = -7290.73,  aic = 14633.46

Training set error measures:
                      ME     RMSE       MAE        MPE     MAPE      MASE       ACF1
Training set -0.03311661 1.376578 0.9390846 -0.1627612 2.197208 0.7910732 0.04759557
```

When compared to our model, we are getting a much lower MAPE, which shows that our model is better fitting than the competitor.

References:

1. 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

2. Hu, Y., & Hellendoorn, J. (2013, April). Daily traffic volume modeling based on travel behaviors. In Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on (pp. 639-644). IEEE.

3. Qi, Y., & Ishak, S. (2013). Stochastic Approach for Short-Term Freeway Traffic Prediction During Peak Periods. IEEE Transactions on Intelligent Transportation Systems, 14(2), 660-672. doi:10.1109/tits.2012.2227475

4. He, G. G., & Ma, S. F. (2001). AI-based dynamic route guidance strategy and its simulation. In Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE (pp. 28-32). IEEE.

5. Tao, R., Xi, Y., & Li, D. (2016, July). Simulation analysis on urban traffic congestion propagation based on complex network. In Service Operations and Logistics, and Informatics (SOLI), 2016 IEEE International Conference on (pp. 217-222). IEEE

6. Hu, W., Yan, L., & Wang, H. (2014, October). Traffic jams prediction method based on two-dimension cellular automata model. In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on (pp. 2023-2028). IEEE.

7. Xu, Y., Kong, Q. J., Klette, R., & Liu, Y. (2014). Accurate and interpretable bayesian mars for traffic flow prediction. IEEE Transactions on Intelligent Transportation Systems, 15(6), 2457-2469.

8. Jiang, B., & Fei, Y. (2014, June). On-road PHEV Power management with hierarchical strategies in vehicular networks. In Intelligent Vehicles Symposium Proceedings, 2014 IEEE (pp. 1077-1084). IEEE

9. Yin, H., Wong, S., Xu, J., & Wong, C. K. (2002). Urban traffic flow prediction using a fuzzy-neural approach. Transportation Research Part C: Emerging Technologies, 10(2), 85-98.

10. Yu, Y., Li, J., Guan, H., Wang, C., & Wen, C. (2016). Bag of Contextual-Visual Words for Road Scene Object Detection From Mobile Laser Scanning Data. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3391-3406

11. Zheng, N. N., Tang, S., Cheng, H., Li, Q., Lai, G., & Wang, F. W. (2004). Toward intelligent driver-assistance and safety warning system. IEEE Intelligent Systems, 19(2), 8-11

12. Al-Naima, F. M., & Hamd, H. A. (2012). Vehicle traffic congestion estimation based on rfid. International Journal of Engineering Business Management, 4, 30.

13. Thianniwet, T., Phosaard, S., & Pattara-Atikom, W. (2009, July). Classification of road traffic congestion levels from GPS data using a decision tree algorithm and sliding windows. In Proceedings of the world congress on engineering (Vol. 1, pp. 1-3).

14. Xia, L., Siyi, D., & Shuhua, W. (2016, August). Research on intellisense information service oriented to value network model in smart city. In Information and Automation (ICIA), 2016 IEEE International Conference on (pp. 324-327). IEEE.

15. Singh, D., Vishnu, C., & Mohan, C. K. (2016, December). Visual Big Data Analytics for Traffic Monitoring in Smart City. In Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on (pp. 886-891). IEEE.

16. Abreu, B., Botelho, L., Cavallaro, A., Douxchamps, D., Ebrahimi, T., Figueiredo, P., ... & Trigueiros, M. J. (2000). Video-based multi-agent traffic surveillance system. In Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE (pp. 457-462). IEEE.

17. Kirchner, A., & Ameling, C. (2000). Integrated obstacle and road tracking using a laser scanner. In Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE (pp. 675-681). IEEE.

18. Zhu, K., Li, J., & Zhang, H. (2014, December). Stereo vision based road scene segment and vehicle detection. In Information Technology and Electronic Commerce (ICITEC), 2014 2nd International Conference on (pp. 152-156). IEEE.

19. Finnefrock, M., Jiang, X., & Motai, Y. (2005, June). Visual-based assistance for electric vehicle driving. In Intelligent vehicles symposium, 2005. Proceedings. IEEE (pp. 656-661). IEEE.

20. Pantiruc, C., & Negru, M. (2011, August). FPGA Based CAN Data Visualization. In Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on (pp. 245-252). IEEE.