DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Detection and Classification of Anomalies in Road Traffic using Spark Streaming

NATHAN ADOLFO CONSUEGRA RENGIFO

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

KTH Information and
Communication Technology

Detection and Classification of Anomalies in Road Traffic using Spark Streaming

Nathan Adolfo Consuegra Rengifo

Master's Thesis at KTH Information and Communication Technology
Supervisors: Zainab Abbas and Ahmad Al-Shishtawy
Examiner: Vladimir Vlassov

TRITA-EECS-EX-2018:563

Abstract

Road traffic control has been around for a long time to guarantee the safety of vehicles and pedestrians. However, anomalies such as accidents or natural disasters cannot be avoided. Therefore, it is important to be prepared as soon as possible to prevent a higher number of human losses. Nevertheless, there is no system accurate enough that detects and classifies anomalies from the road traffic in real time. To solve this issue, the following study proposes the training of a machine learning model for detection and classification of anomalies on the highways of Stockholm. Due to the lack of a labeled dataset, the first phase of the work is to detect the different kind of outliers that can be found and manually label them based on the results of a data exploration study. Datasets containing information regarding accidents and weather are also included to further expand the amount of anomalies. All experiments use real world datasets coming from either the sensors located on the highways of Stockholm or from official accident and weather reports. Then, three models (Decision Trees, Random Forest and Logistic Regression) are trained to detect and classify the outliers. The design of an Apache Spark streaming application that uses the model with the best results is also provided. The outcomes indicate that Logistic Regression is better than the rest but still suffers from the imbalanced nature of the dataset. In the future, this project can be used to not only contribute to future research on similar topics but also to monitor the highways of Stockholm.

Keyboards: anomaly detection, traffic flow, accidents, weather, decision tree, random forest, logistic regression, streaming.

Abstrakt

Vägtrafikkontroll har funnits länge för att garantera säkerheten hos fordon och fotgängare. Emellertid kan avvikelser som olyckor eller naturkatastrofer inte undvikas. Därför är det viktigt att förberedas så snart som möjligt för att förhindra ett större antal mänskliga förluster. Ändå finns det inget system som är noggrannt som upptäcker och klassificerar avvikelser från vägtrafiken i realtid. För att lösa detta problem föreslår följande studie utbildningen av en maskininlärningsmodell för detektering och klassificering av anomalier på Stockholms vägar. På grund av bristen på en märkt dataset är den första fasen av arbetet att upptäcka olika slags avvikare som kan hittas och manuellt märka dem utifrån resultaten av en datautforskningsstudie. Dataset som innehåller information om olyckor och väder ingår också för att ytterligare öka antalet anomalier. Alla experiment använder realtidsdataset från antingen sensorerna på Stockholms vägar eller från officiella olyckor och väderrapporter. Därefter utbildas tre modeller (beslutsträd, slumpmässig skog och logistisk regression) för att upptäcka och klassificera outliersna. Utformningen av en Apache Spark streaming-applikation som använder modellen med de bästa resultaten ges också. Resultaten tyder på att logistisk regression är bättre än resten men fortfarande lider av datasetets obalanserade natur. I framtiden kan detta projekt användas för att inte bara bidra till framtida forskning kring liknande ämnen utan även att övervaka Stockholms vägar.

Keyboards: anomalitetsdetektering, trafikflöde, olyckor, väder, beslutsträd, slumpmässig skog, logistisk regression, streaming.

Contents

Chapter 1

Introduction

1.1 Anomaly Detection

The purpose of *anomaly detection* is defined as finding data that does not conform to the notions of normal behavior. Other words that are widely used to reference this type of data are *outliers, exceptions, aberrations, surprises or peculiarities*. The relevance of this problem comes from the need to act upon the discovery of the outliers since such occurrence usually requires a response in many application domains. An example of this can come when examining an anomalous MRI image that may indicate the presence of a malignant tumor, something that requires an immediate action from the doctor in charge. [32] [3] [1]

Such is the importance of *anomaly detection* that the technique has been studied in the statistics community since at least the 19th century, prompting the creation of many techniques in different research communities. To further understand the challenges that come from detecting outliers, the following are the issues related to this process [32]:

- **Definition of normal behavior**: the main difficulty for this task comes when the boundary between normal and abnormal behavior is not precise, something that usually leads to the misclassification of an observation as abnormal when it's actually normal and vice-versa.
- **Malicious actions**: there are cases in which malicious adversaries decide to adapt themselves in order to appear as a normal observation, which is something that ends up complicating the task of defining normal performance. Examples of this can be found in credit card fraud, breakdown of a system, terrorist activity or cyber-intrusion.
- **Notion of anomaly differs for different domains**: every domain has its own needs and data behavior, making it impossible to use a model

defined for one domain in another.

- **Evolving behavior**: multiple domains suffer from data that changes its normal behavior over time and this is something that can turn a well defined model obsolete.
- **Availability of labeled data**: the training of a model with the purpose of classifying anomalies within a dataset can become quite difficult when there is no information that verifies the classes to which different behaviors belong to.
- **Noise**: it is often the case in which data contains noise that is quite similar to the anomalies and therefore are difficult to distinguish and delete.

When dealing with the detection of anomalies, the first thing one should do is understand their nature, which is why there are 3 main categories that outliers can be classified into [32]:

1. **Point Anomalies**: in this category, an outlier is defined as a data point that lies outside the boundary of the normal region of the observations, making them different from normal points.
2. **Contextual Anomalies**: it occurs when the anomalies can be classified as such only within a specific context. Because of this, there are 2 attributes that each data point should have:
    - **Contextual attributes**: indicates the context for a given instance. For example, in time series, time itself is a contextual attribute that determines where each observation belong in the sequence.
    - **Behavioral attributes**: this refers to the characteristics of the observation that are not bound by context. An example of this can be found in a spatial dataset describing the average consumption of a product in the entire world, where the amount at any specific location represents a behavior.
3. **Collective Anomalies**: this kind of outliers arrive when a collection of related observations is anomalous with respect to the entire dataset, but each data point by itself might not be an outlier.

1.2 Traffic Flow Theory

As previously mentioned, each application domain constitutes different behaviors for its dataset. Therefore, the next step is to understand how road traffic data behaves, and for this, one needs to look at the *Traffic Flow Theory*.

Its premise indicates that 3 variables are required to model the behavior of vehicles on the road [12]:

1. **Flow**: refers to the number of vehicles passing per unit of time and it's express as $q(x,t)(veh/min)$.
2. **Density**: indicates the number of vehicles passing per unit of distance and it's defined as $\rho(x,t)(veh/km)$.
3. **Velocity or Speed**: describes the travelled distance per unit of time and it's defined as $v(x,t)(km/min)$.

The relationship among the variables can be understood by looking at the definition of flow in Equation 1.1:

$$q(x,t) = \rho(x,t)v(x,t) \tag{1.1}$$

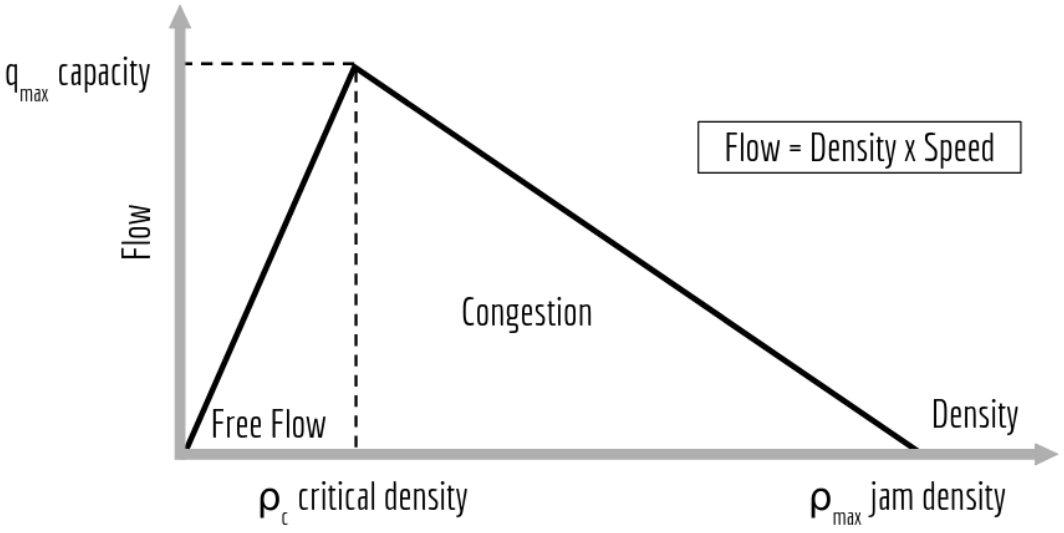

Figure 1.1: Fundamental diagram of Traffic Flow Theory. Adapted from [12]

Furthermore, Figure 1.1 illustrates the behavior of these variables. From it, one can see that as the density increases so does the flow until it reaches its *maximum capacity* (q_{max}), which is the maximum number of vehicles that can transit per unit of time. This also happens to be the same point where the former reaches its *critical density* (ρ_c), which is when the flow starts to decrease and congestion starts to form.

In this figure, the region that defines normal behavior in this domain seems to be quite clear and it refers to the data points that are inside the boundaries form by the triangle under q_{max} and ρ_c. However, if one wishes to not only detect but also understand the reason for the presence of the outliers, then the type of exceptions that can come up in this domain fall into the category of *contextual anomalies*. This occurs because traffic flow is usually quite different

during peak hours, the weekend and time of the year, as well as whether the weather is favorable or not (rain, snow, fog, etc..). These are some of the contexts that need to be taken into account for the proper training of a model capable of detecting anomalies, and their presence increases the difficulty of such task.

1.3 Real Time Streaming

With the rise of Big Data happening a few years ago, a need for consuming and processing large amounts of incoming data led to the development of many stream processing (streaming) technologies, and such is the case of *Apache Storm* [13], *Apache Spark* [35] *and Apache Flink* [5]. In particular, one of the main characteristics of *Apache Spark* is its capability of processing data in-memory, something that leads to an increase in performance. [34]

Putting together all the information gathered so far, an interesting idea rises: set up a model capable of detecting anomalies in road traffic and use it as part of a streaming application that receives incoming observations about the traffic flow and detects an anomaly. The idea behind using a model instead of a set of rules becomes more relevant due to the fact that the former makes streaming processing faster.

1.4 Problem, Purpose and Goals

Road traffic control has been around for a long time to guarantee the safety of vehicles and pedestrians. However, anomalies such as accidents or natural disasters cannot be avoided and it is important to be prepared for this kind of events as soon as possible to prevent a higher number of human losses. Therefore, it is necessary to be able to detect, locate and correctly classify the anomaly right when it happens to be able to send help, if needed, to the right place at the right moment. This begs the question: *is it possible to build an application capable of detecting, in real time, anomalies and their nature in the road traffic from the highways of Stockholm?*

1.4.1 Contributions

- **An analysis of the behavior of road traffic observations**: This study made it possible to understand which observations where anomalies based on contexts of weather, time and place. Moreover, the inclusion of accident data was used to detect the time frame in which a traffic accident took place.

- **An analysis of different machine learning techniques for defining a model capable of correctly classifying incoming data**: The results of the experiments indicate that Logistic Regression is better than Decision Trees and Random Forest in terms of F1-score measure [18]. However, detecting accidents is still not possible due to having an imbalanced dataset.
- **A comparison between over-sampling and under-sampling techniques to deal with imbalanced dataset problems**: The F1-score measure obtained for the experiments were high. However, the under-sampled model suffered from information loss and the over-sampled one was overfitted for the accidents label.
- **Design of streaming application**: Powered by the technology of Apache Spark and integrated with the model that obtained the best performance.

1.5 Ethics and Sustainability

When it comes to the ethical aspect of the project, no human subject was involved in the research and no personal data was collected since all the observations that can be found on the dataset used for training the model only reflect the state of the traffic flow on the highways of Stockholm. and no information about the drivers has been gathered. Furthermore, the project contributes to achieving several *SDG* (Sustainable Development Goals) [20], such as:

- **SDG 3 (Good Health and Wellbeing)**: an application that is not only capable of consuming data in real time about the traffic flow on highways but can also detect anomalies and indicate their nature is a powerful tool that can save many lives if an accident is detected on time. Furthermore, if it can also detect that the main cause of the anomaly was related to the weather, professionals in charge of the situation can be more prepared knowing what the conditions of the road are before heading to the place of the incident.
- **SDG 9 (Industry, Innovation and Infrastructure)**: the development of a system capable of detecting anomalies in traffic road is another step forward in innovation and in the advance of technology. Besides this, the work is a step further in understanding how traffic flow behaves and offers a comparison among different techniques that can do so.
- **SDG 11 (Sustainable Cities and Communities)**: since the system will be monitoring traffic to make sure that the corresponding professionals can take action when an anomaly that harms an individual occurs, cities

will become much safer and citizens will be able to rely on help when needed.

1.6 Methodology

The methodology used for the development of the thesis follows an exploratory approach as well as an empirical method based on literature since the investigation was built on the exploration and study of related papers [4]. Besides this, a data-driven approach and an iterative model were also used because subsequent experiments were performed for training different models taking into account the results of previous experiments and analyzed data. Furthermore, the research can be classified as quantitative since models will be ranked based on their performance results using metrics such as accuracy, precision, recall and F-measure.

Table 1.1: Software libraries that were applied during the project.

Library	**Description**	**Version**
pyspark	Spark API for Python [1]	2.2.0
pandas	Data structure and analysis tools for Python [2]	0.22.0
plotly	Interactive data visualization [3]	2.6.0
gmplot	Renders data on top of Google Maps [4]	1.2.0

The programming language selected for the development of the project was *Python 2.7* [5] and the code was written using *Jupyter Notebook* [6] as the IDE (Integrated Development Environment). For information about the tools that were used, Table 1.1 offers a summary of the required libraries, a short description and their corresponding versions.

1.6.1 Project Environment

The thesis was developed at *RISE SICS*, the Swedish Institute of Computer Science, at the headquarters located in Kista, Sweden. Before starting the development of the thesis at the research group that I worked with, a dataset from *Trafikverket* [7] containing observations from the years 2005 to 2016 for the

[1] https://spark.apache.org/docs/2.3.0/api/python/pyspark.html
[2] https://pandas.pydata.org/
[3] https://plot.ly/
[4] https://github.com/vgm64/gmplot
[5] https://www.python.org/download/releases/2.7/
[6] http://jupyter.org/
[7] https://www.trafikverket.se/en/startpage/operations/Operations-road/

traffic flow was already collected and it became the primary data used to carry out the project. Regarding some of its characteristics, one of the most relevant is its size, which is about *500 GB* of data. This makes it virtually impossible to train a good model on a single machine using all observations. Therefore, all the experiments related to the training of a model were performed in a cluster that was interacted with using *HopsWorks* [8] and that is prepared for running Spark applications that use the full power of said cluster.

1.7 Delimitations

The project will first focus on an analysis of the dataset in order to label the boundaries of what is normal and what is not in traffic flow found on the highways of Stockholm. This will classify the behavior of the data taking into account information from weather and accident datasets. The next step, and with the use of the labels previously defined, will be to focus on supervised anomaly detection techniques such as Decision Trees, Random Forest and Logistic Regression. To apply such algorithms, the project will use the *Machine Learning* libraries that come with *Apache Spark* since they are fully compatible with the data structure used with this tool and the cluster in which the experiments will be executed offers full support for this technology. As for the dataset, the project will use the one that the research group had already collected before the beginning of the work, as well as new datasets containing accident, weather and geolocation information (see Chapter 3 for more details).

1.8 Outline

The rest of thesis is organized as follows: Chapter 2 offers a detailed explanation of the theory needed to understand how the project was done and the decisions that were taken, as well as including a section that highlights some of the most relevant works in the field. After this, Chapter 3 deals with the methodology that was applied when training the models used in this project. Then, Chapter 4 details the experiments performed and the results that were obtained. Chapter 5 provides a discussion about the methods, experiments and results. Finally, Chapter 6 outlines some ideas for a future improvement of the work.

[8] https://github.com/hopshadoop/hopsworks

Chapter 2

Relevant Theory

2.1 Decision Trees

Decision Trees is one of the most popular data mining and machine learning techniques that covers both *classification* (identifying the label that corresponds to an observation) and *regression* (predicting a continuous quantity for an instance). The basic idea behind this algorithm relies on its ability to visually represent decisions and decision making. [11] To further illustrate this concept, let's take a look at Figure 2.1:

Predictors				Target
Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

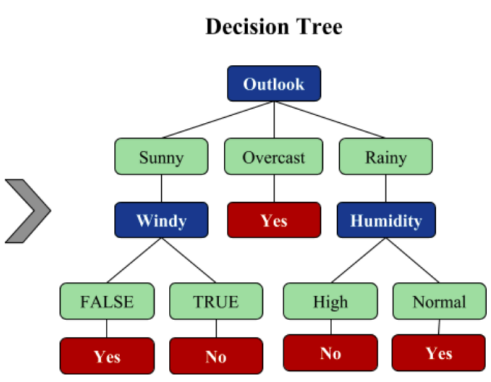

Figure 2.1: From dataset to a Decision Tree. Adapted from [27]

In the table on the left, one can see a dataset with columns (blue header) that contain information about weather and these are defined as *predictors*. Besides this, there is another column (red header) denominated *target* which values determine whether or not is possible to play golf based on the information given by the *predictors* (weather). Of course, trying to understand if it is a good idea to play golf just by looking at the table can be tiring. However, the beauty of this technique allows us to represent the same information presented

in the table on a much more clearer illustration: the decision tree on the right. From it, 3 rules tell us all that is needed to know to make a decision:

- If **sunny**, it is possible to play golf only if is *not* **windy**.
- If **overcast**, it is possible to play golf.
- If **rainy**, it is possible to play golf only if **humidity** is *normal*.

2.1.1 How does Decision Tree work?

It follows a *top to bottom* approach for building the tree, meaning the it first figures out which variable from the dataset is the most important and it puts it as the **root node**. To do this, the algorithm calculates a *cost function* that determines how much *accuracy* is lost by each split [11]. Here is where the concepts of **Entropy** and **Information Gain** come into play [27]:

- **Entropy**: in the context of information theory, entropy determines the average amount of information given by a stochastic source of data. Decision Trees take advantage of this formula and they calculate 2 types of values:
  - **Entropy of the frequency of the target column**: this tells the algorithm how homogeneous the dataset is when divided based on the target column. When this value is close to *0*, the dataset is homogeneous, but if it is close to *1*, the set can be equally divided. Equation 2.1, where S is the target column and c is the total number different values has, details how this operation is performed:

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{2.1}$$

  - **Entropy of the frequency of the target column and another attribute**: the next step is to check how each of the predictors affect the entropy of the label. Equation 2.2 illustrates this process:

$$E(T, X) = \sum_{c \in X} P(c)E(c) \tag{2.2}$$

- **Information Gain**: after the entropy of each predictor with respect to the entropy of the target is calculated, the following steps aim at finding the most homogeneous branch (entropy close to 0), which are also the ones with the highest information gain. The calculation of this value can be seen in Equation 2.3:

$$Gain(T, X) = Entropy(T) - Entropy(T, X) \tag{2.3}$$

After this process the attribute with the highest information gain will be chosen as the **decision node**, which is the same as the **root node** on the first iteration of the algorithm. This successfully divides the dataset in as many partitions as different values can the selected feature has and, for each partition, the process is repeated until all data is classified. The result of the first iteration of this process for the dataset illustrated in Figure 2.1 can be seen in Figure 2.2:

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

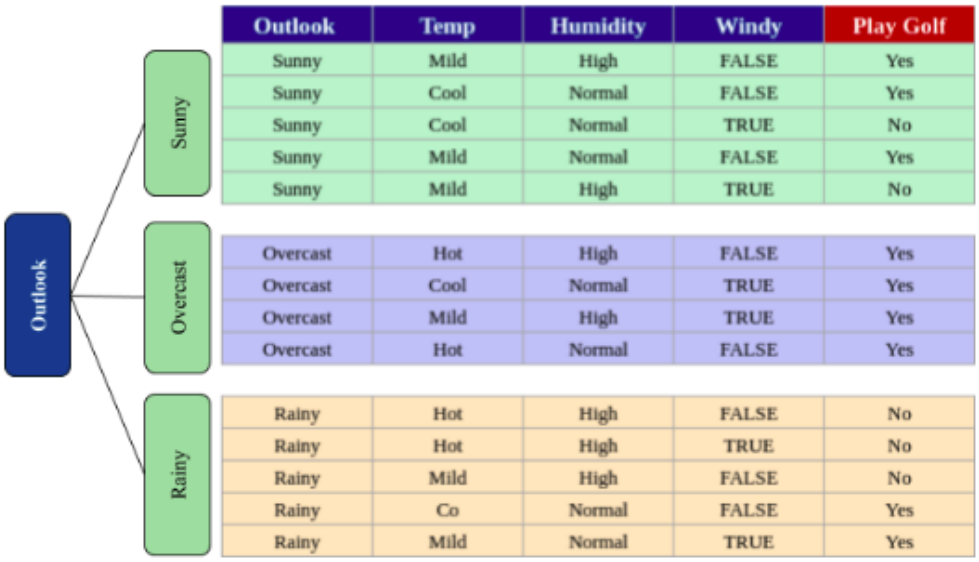

Figure 2.2: Result after the first iteration of the algorithm. Adapted from [27]

2.1.2 Advantages and Disadvantages

The main characteristic of Decision Trees is how simple they are to understand. They achieve this by given a clear visualization of the decision making that leads to predicting the target value that can also be easily interpreted, returning not only a model that can classify observations, but also a model that can give information about the nature of the dataset. They are capable of handling both classification, both binary and multiple, as well as regression

problems and its performance is not affected when features from the dataset are not linearly related. [11]

However, not all datasets are as simple as the one shown in the previous example. Decision Trees can also generate over-complex trees (*overfitting*) that cannot generalize the data as well as the 3 rules obtained in this example. Moreover, two dataset that differ from one another by only small variations can generate 2 completely different trees. There can be a problem with *imbalanced datasets*, which is a situation that occurs when a class represents most of the data and the rest contain only a small portion of it, and this leads to a biased tree towards this dominating label. Lastly, this algorithm is *greedy* and it cannot return a globally optimal tree. [11]

2.2 Random Forests

Now that an explanation of what a Decision Tree is and how it works has been given, understanding what Random Forests is much easier since they are built on top of the former. As mentioned in the previous section, one of the main disadvantages that comes when training a Decision Tree model is that they are prompt to be overfilled. Because of this, Random Forest were made to fix this issue applying the following approaches [8]:

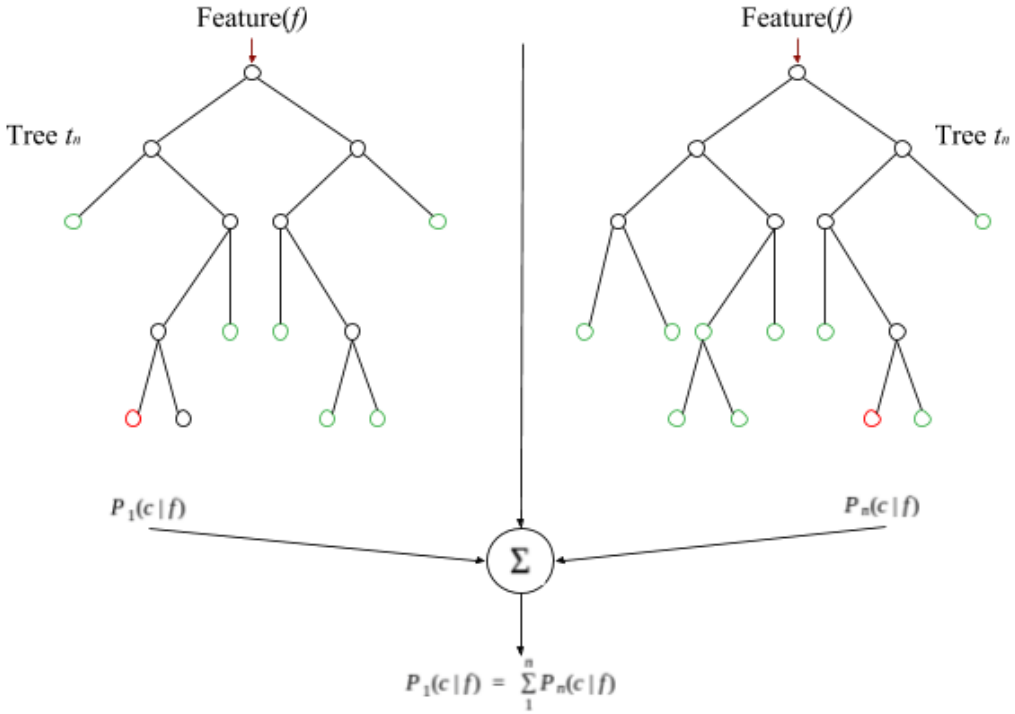

Figure 2.3: Two Decision Trees as part of a Random Forest. Adapted from [8]

- **Random selection of a subset**: Decision Tree main goal, and the main reason that causes overfitting, is to search for the best feature when splitting a node. Random Forest, on the hand, considers only a *random* subset of the features at the moment of splitting a node.
- **Bagging**: after many decision trees have been created, the algorithm has what can be defined as a *forest of decision*, where each of the trees that belong to it have a vote when classifying an incoming observation. The selected label for the new instance will be the one with the majority of the votes. This approach comes from the idea that multiple models put together get a more accurate result. Figure 2.3 is an example of the voting process.

2.2.1 Advantages and Disadvantages

Because of their simplicity, Random Forest are a very popular choice when applying supervised training since the number of hyperparameters it has is not too high, they are easy to understand and their default parameters often generate a good prediction and, just like Decision Trees, it supports both classification and regression problems. If one desires to increase its predictive power, the parameter *max_features* can be modified to limit the amount of features selected by the algorithm at each split. Random Forests are also very good at solving the overfitting problem that Decision Trees suffers as long as there are enough classifiers in the forest.

In general, Random Forests is a very good technique, but there are still some issues that arise when applying this algorithm and the main one is that a large number of trees implies a higher computational cost both when training and when predicting, which is something that might slow down real time applications to the point where using a model trained by this technique may not be adequate.

2.3 Logistic Regression

Logistic Regression is another supervised classification technique that works well with Big data and is also quite different from the ones described in previous sections. This algorithm is based on the *linear regression* model, which is a technique that tries to determine the importance each of the predictors hold for selecting an outcome, as well as determining how each of them impact it [29]. Because of this, Logistic Regressions applies linear regression at the early stages of the algorithm to calculate the *logits (Score)* and it ultimately uses either a *Sigmoid* (Binary classification) or a *Softmax* (Multinomial classification)

function to predict the outcome for the observation [24]. Figure 2.4 illustrates the stages followed by this algorithm.

Figure 2.4: Multinomial Logistic Classifier. Adapted from [24]

1. **Inputs**: just as it was the case for Decision Trees and Random Forests, the inputs for the model are the features that are going to act as predictors and the target attribute that will determine the possible outcomes.
2. **Linear Model**: this is the stage where the algorithm applies the linear regression model. Given a matrix $W = [w_1, w_2, ..., w_i]$, this step takes as input the matrix $X = [x_1, x_2, ..., x_i]$ that contains the values of all the features and outputs their resulting product defined as $Logits = X * W$. These weights will be updated during the training phase, using a loss function, in order to improve the performance since they will be its final output.
3. **Logits**: as mentioned earlier, the *logits* are the resulting product of matrices X and W, and they will be used as the input for *Softmax* function.
4. **Softmax Function**: the goal of this function is to produce a set of probabilities for each of the labels in the target feature that indicate the likelihood the new observation has to fall in any of the target's classes.
5. **Cross Entropy**: given the set of probabilities returned by the softmax function, this step's goal is to calculate the distance between them and the values of the target class. The predicted class will be the one with the shortest distance.
    - **One-Hot-Encoding**: it is important to mention that the values for the target class must be represented as a one-hot-encoded vector.

This kind of vectors take the list of values the target class can have and create a new one where all values are 0 except for the position that corresponds to the selected class. As an example, let's consider a target class with values $X = [animal, plant, human]$. In this case, a one hot encoder for representing the label *human* will be $Y = [0, 0, 1]$.

2.3.1 Advantages and disadvantages

As it was the case with Decision Trees and Random Forests, this technique offers an explanation of the effects the predictors have on the outcome since the weights can be used to explain the importance each of the features represent. [24]

On the other hand, logistic regression also has some limitations. Contrary to the previous technique, this algorithm cannot predict a continuous outcome (regression). If the selected predictors are not the most appropriate for the correct classification of the dataset, the model will have less predictive value, so this model can only be used once those features have been detected. Furthermore, this technique requires that all data points should be independent or otherwise the model will most likely be overfitted. [24]

2.4 Real Time Streaming

This technology became popular during the rise of Big Data thanks to the ever increasing volume of real time information and the need to process it as soon as possible. To further understand what this technology actually is, it is necessary to learn about the 2 concepts that it puts together [30], since only when both characteristics are present a system can actually offer *Real Time Streaming*:

- **Real time**: a system capable of offering this feature should be able to offer a response within a short amount of time, and said shortness is mainly defined by the needs of the application and can range from minutes to milliseconds.
- **Streaming**: this feature offers the capability of continuously processing limitless incoming data by performing a series of transformations to obtain a desired result and whether or not this data is consumed instantly or can be stored for later use will depend on the requirements of the application.

2.5 Related Work

2.5.1 Real-Time Road Traffic Anomaly Detection

This paper describes a model that detects traffic incidents by taking into consideration the speed variation of the cars that are located before and after (upstream and downstream) a certain point on the highway (Figure 2.5) [14].

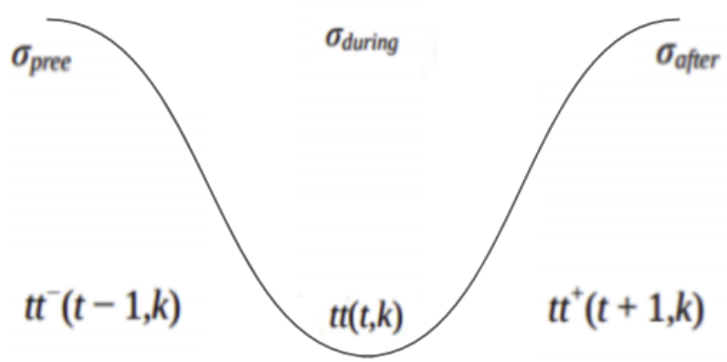

Figure 2.5: Accident characteristics. Adapted from [14]

From the figure, σ is the standard deviation before (upstream), during and after (downstream) the accident, $tt(t,k)$ is the speed computed at time t (moment of the accident) on section k, $tt^{-}(t-1,k)$ is the one computed before and $tt^{+}(t+1,k)$ is the one computed after. Furthermore, the paper introduces an algorithm (see Equation 2.4) for real-time forecasting.

$$tt(t+1,k) = tt^{H}(t+1,k) + \gamma_1 * D + \gamma_2 * UP + \gamma 3 * DS \tag{2.4}$$

The formulas for *D, UP and DS* are expressed as follows:

$$D = desired = [tt^{M}(t,k) - tt^{H}(t,k)]$$
$$UP = upstream = [tt^{M}(t,k-1) - tt^{H}(t,k-1)]$$
$$DS = downstream = [tt^{M}(t,k+1) - tt^{H}(t,k+1)]$$

The algorithm tries to find different traffic modes based on the increase of the upstream (UP) occupation and the decrease of the downstream (DS) occupation where the incident may have happened in order to classify it. Moreover, the algorithm uses a time sequence to describe a traffic state, which is why it includes a moving average algorithm.

The paper also indicates that this implementation is not robust enough to be reliable, but the information presented in the following studies can be added to this model for improvement.

2.5.2 A Real-Time Autonomous Highway Accident Detection Model Based on Big Data Processing and Computational Intelligence

The study proposes a real-time autonomous accident-detection system and compares the performance of 3 different techniques: nearest neighbor model, regression tree, and feed-forward neural network. As for the data, all the models receive as input the average speed, average occupancy and number of cars passing every 2 minutes, as well as the time, taking special consideration to rush hours and whether the event occurs in a weekend or not. [19]

The following table (Table 2.1) shows the performance for each of the models that were trained in this study. The metrics used to evaluate them are the following: $TPR = TP/(TP + FN)$, $TNR = TN/(TN + FP)$, $PPV = TP/(TP+FN)$, $NPV = TN/(TN+FN)$ and $Accuracy = (TP+TN)/(TP+TN+FP+FN)$, where TP is true positives, FN is false negatives, FP is false positives and TN is true negatives, the R at the end of TPR and TNR means *Rate*, PPV is *Positive Predicted Value* and NPV is *Negative Predicted Value*.

Table 2.1: Performance comparison of different models. [19]

Model	Recall TPR (%)	Recal TNR (%)	Precision PPV (%)	Precision NPV (%)	Accuracy (%)
1	92.86	95.12	0.09	100	95.12
2	92.86	96.78	0.15	100	96.78
3	85.71	97.59	0.18	100	97.59
4	78.57	98.27	0.23	100	98.27
5	78.57	98.56	0.28	100	98.56
6	85.71	98.37	0.27	100	98.37
7	50.00	99.51	0.51	100	99.50
8	42.86	99.79	1.02	100	99.79

The results for all the models is very good, but they returned a high number of false positives. This was improved by increasing a bias towards reducing the false positives (models 3, 5, 7 and 8 in Table 2.1), but such action ended up affecting the recall (see section 3.1.2 for definition). For more information about each model, see Table 2.2.

Table 2.2: Models specifications. [19]

Model	Name	Loss	Neurons
1	Nearest Neighbor	0	-
2	Regression Tree with Equal Bias	0	-

Model	Name	Loss	Neurons
3	Regression Tree with Less Alarms	0.5	-
4	Feed-forward Neural Network with Equal Bias	0	20
5	Feed-forward Neural Network with Less Alarms	0.5	5
6	Feed-forward Neural Network with Equal Bias	0	10
7	Feed-forward Neural Network with Less Alarms	0.5	20
8	Feed-forward Neural Network with Less Alarms	0.94	10

At the end of the paper it is suggested that a combination of different models might improve the result, as well as the addition of meteorological parameters, road topology and/or condition of the road.

2.5.3 Anomaly Detection by Combining Decision Trees and Parametric Densities

This paper describes a model that consists of the combination of the following two techniques [17]:

1. **Outlier Detection**: based on analysis for both discrete and continuous uniform distributions, this technique tries to identify outliers presented in the data so that they can be labeled as the anomalous.
2. **Decision Tree**: based on the output of the previous technique, a decision tree model is trained to separate the normal from the anomalous data. The building of the tree reaches a stopping criterion depending on an error limit selected with respect to the prior probability $P(C_A)$, where C_A is the anomalous class. Figure 2.6 shows that a low $P(C_A)$ means less leafs (represented by rectangles in the image) and the brightness show how confident the algorithm is with classifying the points as anomalous.

The first technique is only useful for separating normal from anomalous data, so it cannot be used for classifying traffic events using multiple classes. However, using decision trees once a set of classes has been defined seems to be a good approach for solving the classification problem.

2.5.4 Statistical Anomaly Detection

The paper combines two techniques: Principal Anomaly Detection (PAD) and Bayesian Principal Anomaly [2].

- The former consists on describing a known distribution $P(x|\theta)$ over a set of parameters θ and then defining $A(z|\theta)$, which is the probability of generating a more common example than z. This can be seen in Figure

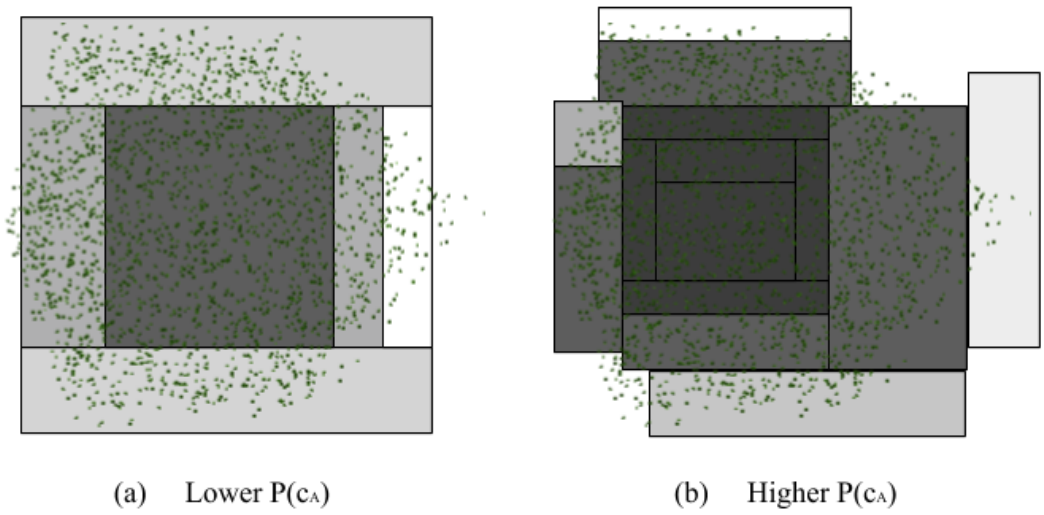

Figure 2.6: Different decision boundaries on synthetic 2D example. Adapted from [17]

2.7, where an anomaly limit is set and every observation z that has a probability $A(z|\theta)$ above such line is considered anomalous.

- The latter is used when the distribution over the parameters of θ is unknown and uses the set of training samples X to find it, which leaves us with the following: $A(z|X) = \int A(z|\theta)P(\theta|X)$

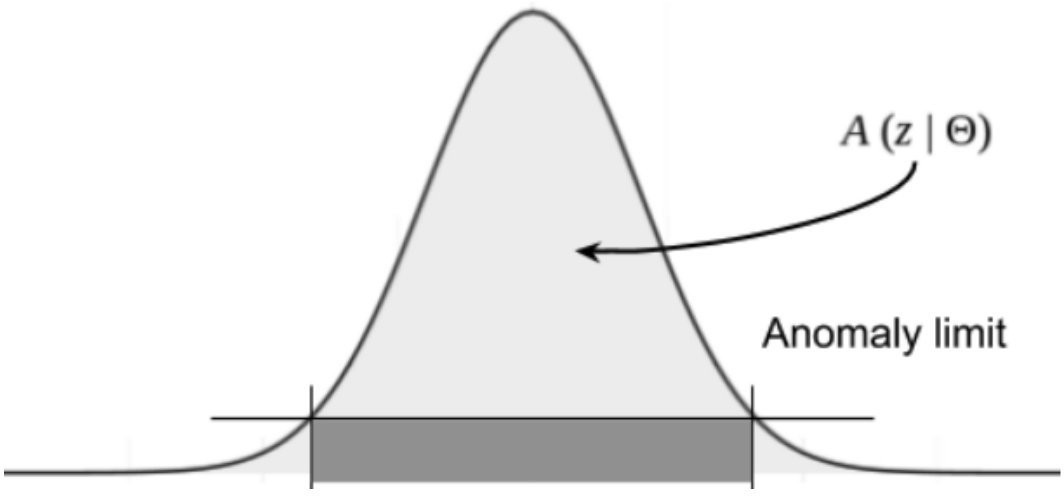

Figure 2.7: Principal Anomaly $A(z|\theta)$. Adapted from [2]

Regarding the calculation of this probability, the example given on the paper follows a Poisson distribution (rate of events per time unit). For traffic anomaly classification, this distribution can be used to detect congestion, but it would be necessary to take into consideration other data and most likely consider the use of a different probability distribution.

As for multiple classes, the paper suggests having different Bayesian Principal Anomaly equations where each one calculates its own probability that an event belongs to a certain class.

2.5.5 Anomaly Detection for Temporal Data using LSTM

The paper proposes the use of a Long Short-Term Memory recurrent neural network that is trained on normal data to predict normal time series patterns and predict future values where the prediction errors are used to detect anomalies. [28]

2.5.6 Naive Bayes and Decision Tree for Adaptive Intrusion Detection

The technique described in this paper proposes to first use Naïve Bayes for classifying the data and then classify the observations. If any example is misclassified, then the technique uses the ID3 algorithm (Decision Tree) to continue with the training until all examples are correctly classified. [7]

Chapter 3

Methods and Datasets

3.1 The Strategy

The process followed during the development of the project can be divided into two sections. The first one deals with an exploration of the dataset [6] where not only a normal behavior region for the observations is defined, but also many subregions within the boundaries of the anomalies are labeled as well. As for the second section, once the labels have been set, the dataset will be used to train model applying all of the techniques described in Chapter 2.

3.1.1 Data Exploration

One of the main contributions this project aims at offering is the understanding of how the traffic flow on the highways of Stockholm behaves. To get an inside look at its conduct, *plotting* is a veritable tool [6] that can be used to show the hidden knowledge within the dataset. Moreover, one can take a look at multiple statistical measures, such as the *mean* and *standard deviation* of a variable, see how it interacts with other features and separate the normal from the abnormal, as well as offering a framework for explaining the reasons behind its nature.

During this section of the work, the exploratory data analysis will focus on understanding how each of the highway points where the sensors are located is affected by the following features:

- **Speed and Density across Time**: the information the sensors provide about the first two variables is key to understand not only the normal behavior for each of the roads, but also the normal behavior at its *different sections*. Moreover, these two variable offer data that has a context within time, so the statistical measures that will be collected from the three

features will be the *mean* and *standard deviation* of the *speed* and *density* per *hour* of the day for all the sensors.

- **Accidents**: the dataset that will be used contains information about the day and hour in which an incident took place, but it does not offer the exact minute in which it started, meaning that an accident starting at *14:55* will be represented as occurring at *14:00*. This is quite unfortunate since the observations at the latter hour do not express the behavior of the traffic flow during the occurrence of this type of event. Therefore, the data exploration regarding this anomaly will take into account the observations within the range of 2 hours before and after the hour indicated by the accidents dataset in order to detect the time frame in which the event actually took place by looking at the speed and density values extracted from the two nearest sensors to the accident location.

3.1.2 Model Training

This section of the project will focus on applying the different supervised classification techniques, described in the previous chapter, for training and comparing multiple models. Furthermore, a method called *k-fold cross validation* [26] will be used during the training phase for each of the chosen techniques. This approach works in the following manner:

1. **Partition the dataset**: the original dataset is randomly partitioned into k sections of equal size.
2. **Selection of the validation set**: *k-1* sections are selected for training the model and the remaining fold is selected as the test set that will be used to measure its performance.
3. **Repetition**: the previous step is performed k times and in each of those iterations a different fold will be used as the validation test.
4. **Average the results**: the performance for all models is put together by taking their average to produce a single estimation.

The main motivation behind applying this method is that all data points will be used both for training and testing the performance of the model, and all instances are part of the validation set *exactly once*.

The metrics that will be used for evaluating the performance of each of the model [18] are *Accuracy* and *F1 Score*:

- **Accuracy**: it determines the percentage of data points that were classified into the right category. Works well when the dataset is balanced, but a high accuracy can be misleading when some categories represent a

small portion of the dataset. Equation 3.1 illustrates how this value is computed:

$$Accuracy = \frac{Number\ of\ correct\ detections}{Total\ number\ of\ detections} \tag{3.1}$$

- **F1 Score, F1 Measure or F-score**: by using the Harmonic Mean, it balances the information given by the *precision*, which indicates how precise the model is for correctly classifying the observations, and *recall*, which measures how robust it is, a characteristic that is achieved when a significant number of instances is not missed. Equation 3.2 indicates how the metric is calculated:

$$F1 = 2 * \frac{1}{1/precision + 1/recall} \tag{3.2}$$

3.2 Datasets

As previously mentioned, the main dataset used to carry out the work was provided by the research laboratory in which the work was carried out. However, in order to detect as many different types of anomalies as possible, more data was looked up for and merged to the one already available. The following subsections detail information about these datasets and from where they were extracted.

3.2.1 Sensor Data

The dataset containing the information about the traffic flow behavior has been extracted from a total of 2000 sensors spread across the highways of Stockholm. Figure 3.1 illustrates how the sensors, represented by purple circles, are distributed across roads.

Table 3.1 presents a summary of the main variables extracted from the main dataset, as well as indicating whether the variable was selected to carry out the project. The descriptions shown in the table were given to the research group by the organization responsible for collecting the data.

Table 3.1: Information recollected in the main dataset.

Variable	Description	Selected
Timestamp	Time at which the event took place	Yes
Road	Name of the highway	Yes

Variable	Description	Selected
KM	Kilometer reference	Yes
Lane	Single line of vehicles (*left to right*)	Yes
Direction	Forward or Counterflow	No
Flow	Number of vehicles per minute	Yes
Average Speed	Average speed per minute (km/h)	Yes
Sign Aid Det Comms	Binary representation of the status	No
Status	Code representation of the status	No
Legend Group	Code for the kind of sign displayed	No
Legend Sign	Values for the sign, if any	No
Legend SubSign	The kind of frame the sign should have	No
Protocol Version	Version of MCS Simone/TOP protocol	No

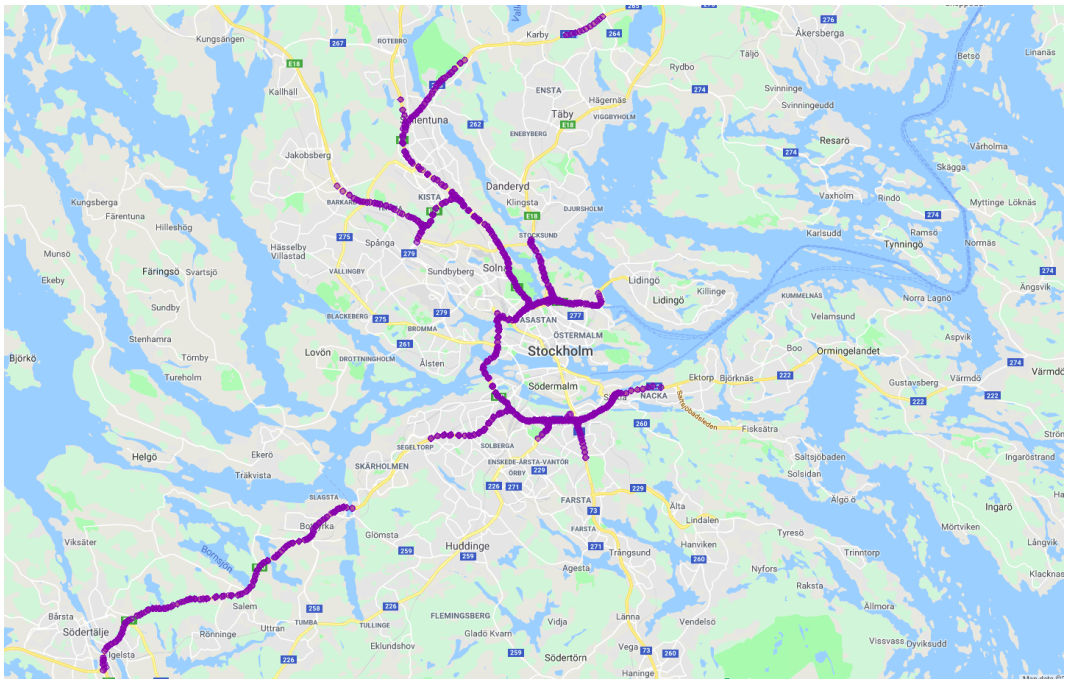

Figure 3.1: Distribution of the sensors across highways.

Each of the variables selected were chosen for certain reasons. The following details those reasons for each of them:

- **Timestamp**: this variable indicates the exact moment in which an observation took place, something that is quite relevant since traffic behaves differently based on the moment of the day, week and month.
- **Road**: since not all roads behave in the same way, it is necessary to take into account and model each of their behaviors.
- **KM and Lane**: these two variables, alongside with the *road*, represent a single **sensor** and also indicate in which order they are placed in the

highway. This information is quite useful when detecting anomalies such as *accidents*.

- **Flow and Average Speed**: they are two of the main variable used in Traffic Flow Theory (described in section 1.2) and are required to understand the behavior of the traffic.

Furthermore, new variables were derived from the selected group. A summary about what these new variables are and how they were extracted from the existent ones can be found in Table 3.2, followed by an explanation for why they are necessary.

Table 3.2: Derived variables.

Variable	**Origin**
Density	Flow / Average Speed
Hour	From the Timestamp
Weekday	From the Timestamp
Day Moment	Grouping of hours. See Table 3.3

- **Density**: this new feature is part of the variables that compose the traffic flow theory and, therefore, is required to understand the behavior of the traffic. Since the *average speed (v(x,t))* is in *km/h* and the *flow (q(x,t))* is in *veh/min*, the equation (3.3) for obtaining the *density* ($\rho(x,t)$) in *veh/km* is as follows:

$$\rho(x,t)(veh/km) = \frac{q(x,t)(veh/min)}{v(x,t)(km/60min)} \tag{3.3}$$

- **Hour and Weekday**: the anomalies that can be found in the traffic flow can be labeled as *contextual anomalies* since it is affected by the moment and type of the day, which is its *contextual attribute*.

- **Average of the previous 20min for Speed and Density**: the need for these 2 variables comes from the fact that anomalies such as accidents are represented by a group of observations instead of a single one. Besides, this type of anomaly is characterized by a sudden drop in speed values and a rise in the case of density, so having the average of the previous 20min will probably help the model in detecting the drop/raise for these features and that they may be caused by an accident.

- **Day Moment**: traffic is constantly affected by the *rush hour* phenomenon, which represents the moment of the day when traffic congestion is at its highest. Because of this, the hours of the day can be grouped to represent both normal behavior and an anomaly caused by the *rush hour*. Table 3.3 details the selected grouping for the hours of the day.

Table 3.3: Values for the variable *Day Moment*.

Value	Range of Hours
Night	[0,2]
Early Morning	[3,8]
Morning	[9,12]
Afternoon	[13,19]
Evening	[20,23]

3.2.2 Geolocation

One of the main goals of this project is to be able to detect anomalies related to traffic accidents in real time. To do this, a historical dataset with information about such events is necessary to understand how this specific type of outliers behave. As it will be seen in the following section, the available dataset for the accidents contains the GPS coordinates of the location for where each of the events took place and, in order to match this information with the main dataset, it is necessary to obtain the GPS coordinates for each *road, km and lane*. A summary for the dataset with such information can be seen in Table 3.4 and it was provided by the research group with whom the work was carried out.

Table 3.4: Dataset with GPS information of the roads.

Variable	Description
Latitude	Specifies *north-south* position on Earth's surface
Longitude	Specifies *east-west* position on Earth's surface
Road	Name of the highway
KM	Kilometer reference
Lane	Single line of vehicles (*left to right*)

3.2.3 Accidents

A dataset containing 9507 accident observations on the traffic roads of Stockholm from 2003 to 2015 was extracted from *Trafikverket* [1] and was used to accurately define the behavior for this kind of anomaly. Figure 3.2 illustrates where each of these accidents took place within the roads of Stockholm.

As it can been seen in the figure (3.2), the dataset also contains observations that occurred outside the highways of Stockholm, so these needed to be filtered

[1] https://www.trafikverket.se/en/startpage/operations/Operations-road/

out. Unfortunately, none of the observations included information about the kind of road where such events took place. See Table 3.5 for a summary about the variables presented in the dataset.

To overcome this issue, an Application Programming Interface (API) from *Open Stree Map* [2] called Nominatin (Latin for *'by name'*) was used [3]. This API has the power to *name* the place that the latitude and longitude from a set of GPS coordinates refer to, making it useful for finding the street and road name of specific location.

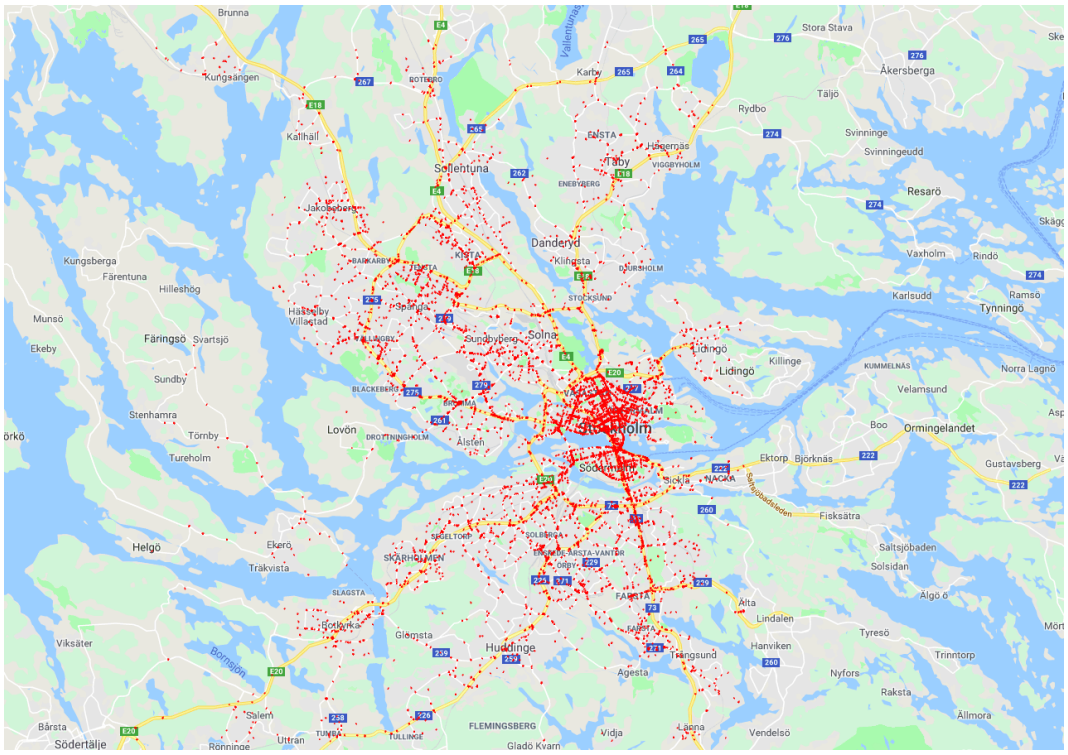

Figure 3.2: Location of the accidents.

Table 3.5: Description of the accident dataset.

Variable	**Description**
Date Time	Date and hour of the event
Latitude	Specifies *north-south* position on Earth's surface
Longitude	Specifies *east-west* position on Earth's surface

Furthermore, the API also has the power to indicate the type of road for any given GPS coordinate, and this feature that was used as the main tool for

[2] https://www.openstreetmap.org/about
[3] https://wiki.openstreetmap.org/wiki/Nominatim

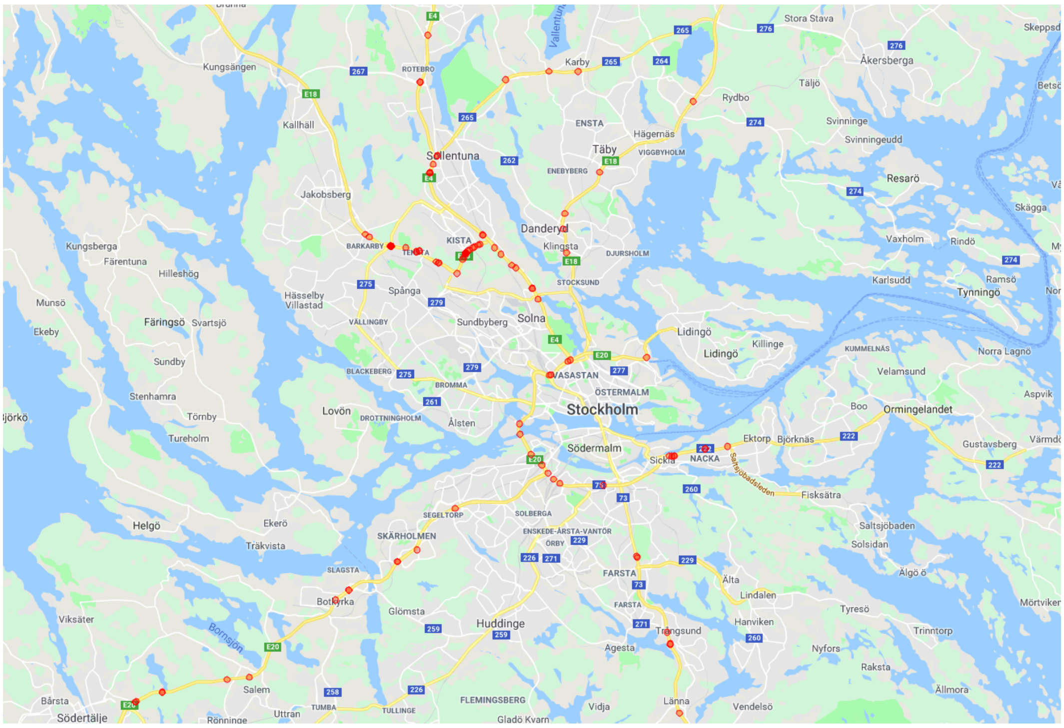

Figure 3.3: Accidents in highways.

filtering out the observations that were not related to a highway. The result of this filtering can be seen in Figure 3.3, where out of the 9507 observations, only 66 occurred in one of the highways of Stockholm.

After the filtering, the next step was to match the accident to the two nearest group of sensors, which are determined by the features *road* and *KM* from the main dataset. In order to do this, the first task that was necessary to perform was the calculation of the distances between the place of the accidents and the nearest sensors. Equation 3.4 illustrates the *Haversine Formula*, which is used to obtain the great circle distance between 2 given points [22]. In it, ϕ represents the latitudes and ψ the longitudes of the given GPS coordinates, whereas r represents the radius of the earth, which is *6371km* [33].

$$d = 2r\sin^{-1}\sqrt{\sin^2(\frac{\phi_1-\phi_2}{2}) + \cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\psi_1-\psi_2}{2})} \tag{3.4}$$

The results for of this task can be seen in Figure 3.4, where the red circle represents the accident, and the green and blue circles are the locations of the 1st and 2nd group of nearest sensors respectively.

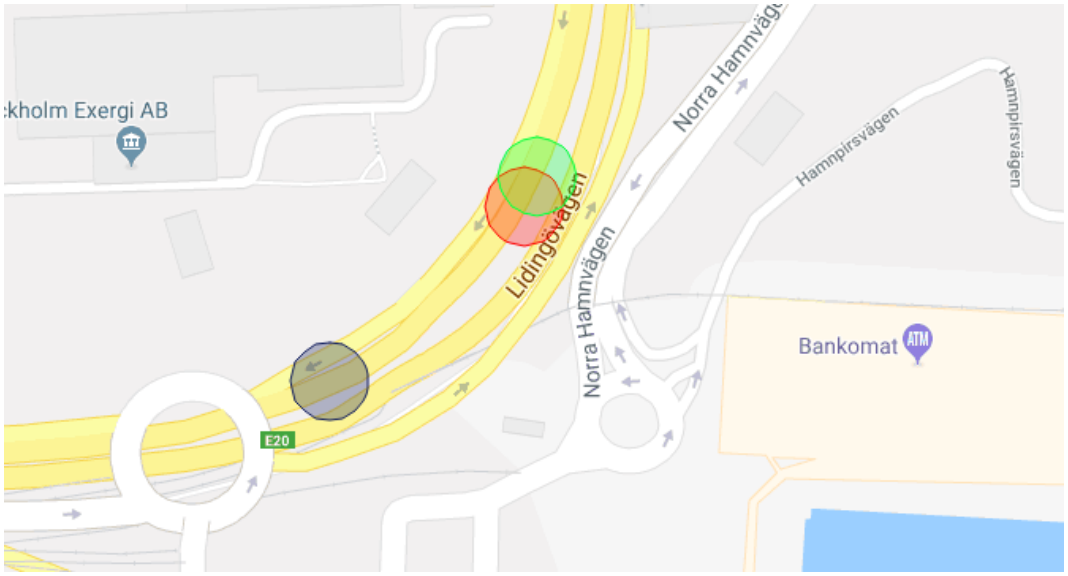

Figure 3.4: Nearest sensors to a located accident.

3.2.4 Weather

The next dataset needed to carry out the project is one that contains weather information. The reason for this is that such dataset will help determine if the anomaly detected by the system is caused by unfavorable atmospheric conditions. Fortunately, the API from *Dark Sky* [4] offers historical information about a certain location, which is defined by its GPS coordinates, and the resulting data is offered by hour of the day.

Regarding the data about the atmospheric conditions returned by this API, each observation contains information about the *temperature, humidity, dew point, wind speed, etc...* However, the most prominent feature is a categorical property that summarizes the atmospheric conditions in the following classes: *clear-day, clear-night, rain, snow, wind, fog, cloudy, partly-cloudy-day, and partly-cloudy-night.* From this, the system will be able to determine if the congestion was due to unfavorable atmospheric conditions when the observation is labeled as *snow, fog or rain.*

3.3 Streaming

The streaming application design can be seen in Figure 3.5. The flow of the data starts by collecting the observations from the sensors located on the highways of Stockholm. When the information enters the streaming application, the data will be grouped by sensor using the variables road, lane and km. Time

[4] https://darksky.net/

windows will be used to calculate the average speed and density values of the past 20min. All this information will arrive at the model node where the observation will be classified according to the best fitted model from this research. Finally, the output will be the category predicted by the model.

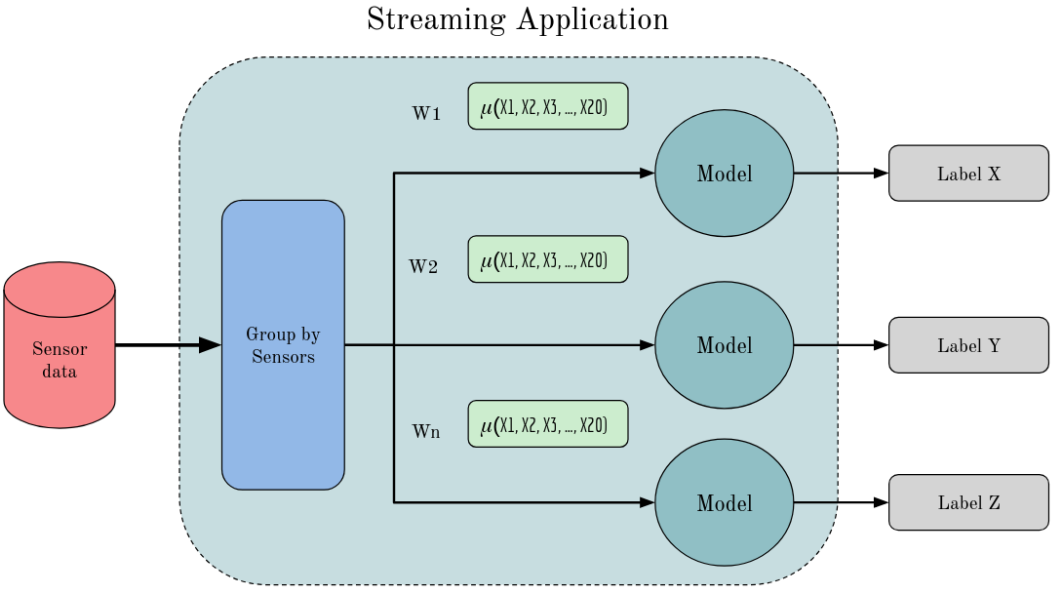

Figure 3.5: Streaming application design.

Chapter 4

Experiments and Results

4.1 Main Goal

This chapter recollects the results obtained by each of the experiments performed, both in the Data Exploration and Model Training phases of the development, as well as offering an analysis for the outcomes.

4.2 Data Pre-processing

For the values of the variables *Speed* and *Flow* to be considered valid, they must fall within a threshold, as any observation outside these intervals has another meaning. Table 4.1 indicates what these intervals are and the meaning for each of them. For both variables, only the values that fall within the category *OK* are taking into account for the data exploration phase but are kept during model training.

Table 4.1: Value ranges for variables Speed and Flow.

Speed	**Category**	**Flow**	**Category**
[2, 250]	OK	[0, 120]	OK
251	Speed is lower than 2 km/h	251	Request failed
252	No vehicles have passed	252	Lanes are not used
253	Lanes are not used	253	Minute speed not possible
254	Minute speed not possible	254	Start Value
255	Start Value	-	-

Furthermore, a randomly selected subset of the observations or *random sampling* [15] of the sensor dataset was taken in order to plot each observation

in some of the experiments that took place during the data exploration phase. This sampling took *5%* for each month from the year 2016 in order to make sure that the sample could represent the dataset.

4.3 Data Analysis

The main goal of this phase is to define the regions within which an observation can be considered *normal* or *not congested* in the traffic flow context. Moreover, the *abonormal* or *congested* regions will be further analysed to understand the reasons behind the occurrence of these anomalies.

4.3.1 Traffic Flow Curve

The first experiment was made to determine the maximum flow (q_{max}) and critical density (ρ_c) shown in Figure 1.1 for the highways of Stockholm. To do this, the *random sample* was used to plot the observations and obtain the traffic flow curve described in mentioned figure.

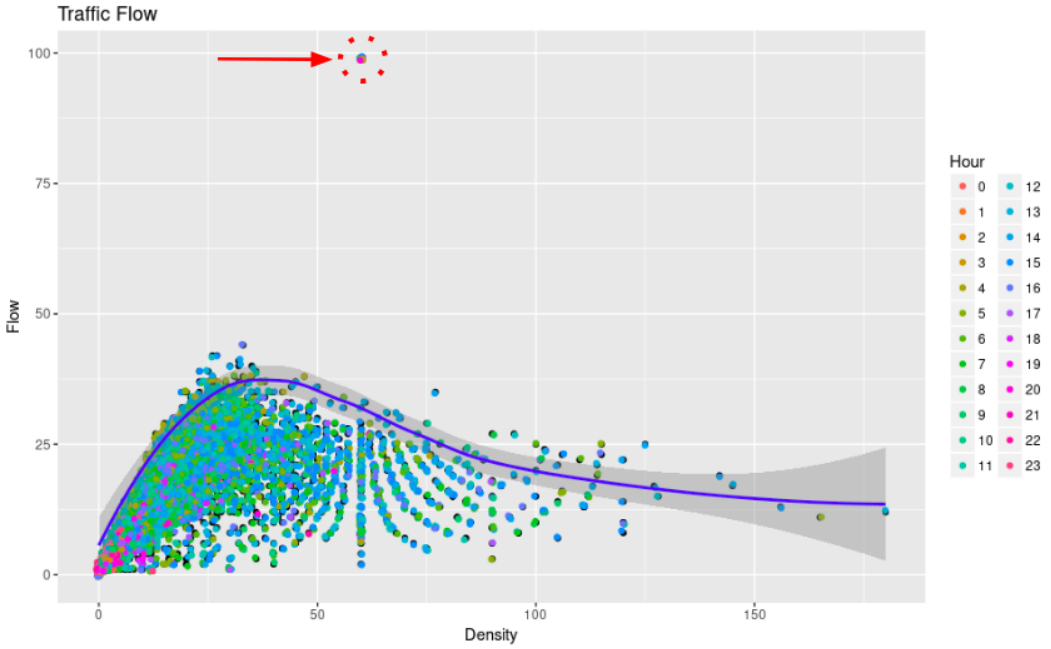

Figure 4.1: Traffic Flow curve for sensor data with sensor errors.

From this experiment, 2 conclusions about the dataset were reached. The first is illustrated in Figure 4.1, where the observations pointed by the red arrow indicate that their $flow = 99\ veh/min$ and their $density = 66\ veh/km$, which implies that their $speed = 90\ km/hr$. These values are *incorrect* since

they suggest that 99 vehicles are passing through the sensor at a speed of just 90 km/hr, while registering 66 vehicles per km, *each minute*. This is physically impossible and the plot clearly shows this since these points are far away from the traffic flow curve. Therefore, this type of observation will be classified as a *Sensor Error* and will be filtered out from the dataset for the rest of the experiments performed in this phase.

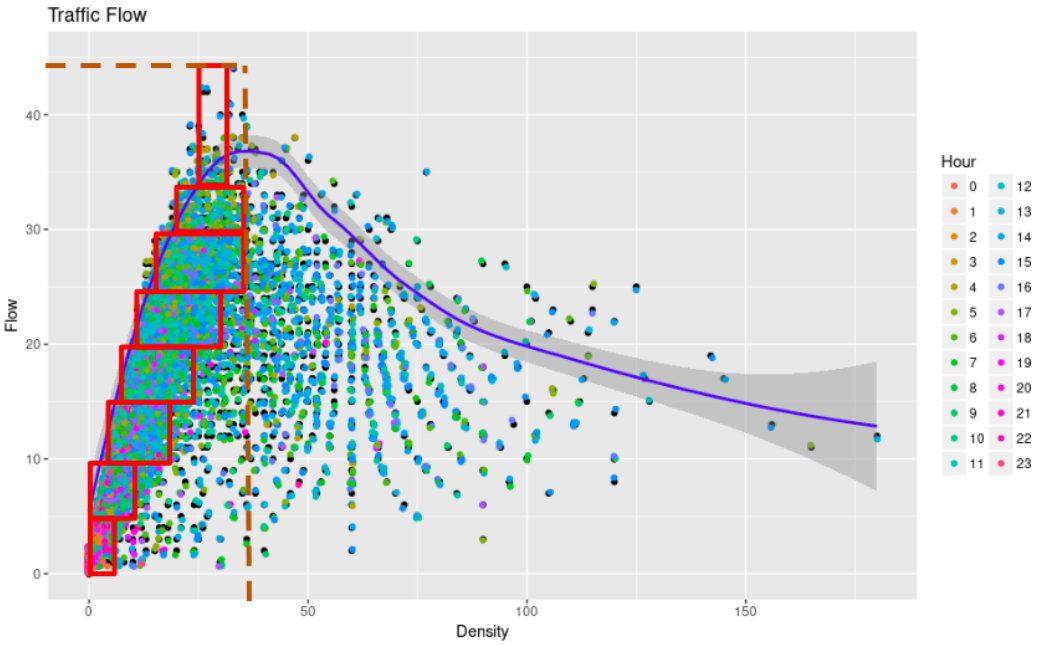

Figure 4.2: Traffic Flow curve for sensor data.

The latter conclusion comes from examining Figure 4.2. First, let's take a look at the discontinued brown lines: they represent the threshold marked by the maximum flow (q_{max}) and the critical density (ρc), which implies that the region symbolizes the *not congested* area. However, if one looks at the observations below *flow = 25*, it can be seen that the critical density also decreases, meaning that the threshold that defines a not congested observation is not wide region defined by the discontinued lines, but a set of multiple sections delimited by the red rectangles, and their definition can be seen in Table 4.2. The aforementioned table indicates the upper limit density value an observation must not surpass to be classified as *not congested* when the flow is within the corresponding range.

Table 4.2: Boundaries that define a *not congested* observation.

Flow	**Density**
[0, 5]	5
[6, 10]	10
[11, 15]	18

Flow	Density
[16, 20]	23
[21, 25]	28
[26, 34]	35
> 34	30

The selection of these boundaries comes with a prices since the set of thresholds are not guaranteed to be the most optimal for the cases that are close to the outside of the defined regions. Nevertheless, from this moment on all experiments will consider this threshold as *ground truth* [23] for the definition of not congested instances.

4.3.2 Day Moments

The next experiment's goal was to be able to detect different moments within the day where the behavior of the traffic flow would behave similarly, and the result of this can be found Figure 4.3. This Figure shows the observations found in the random sample of the main dataset across the day and classifies them to the groups that were defined in Table 3.3.

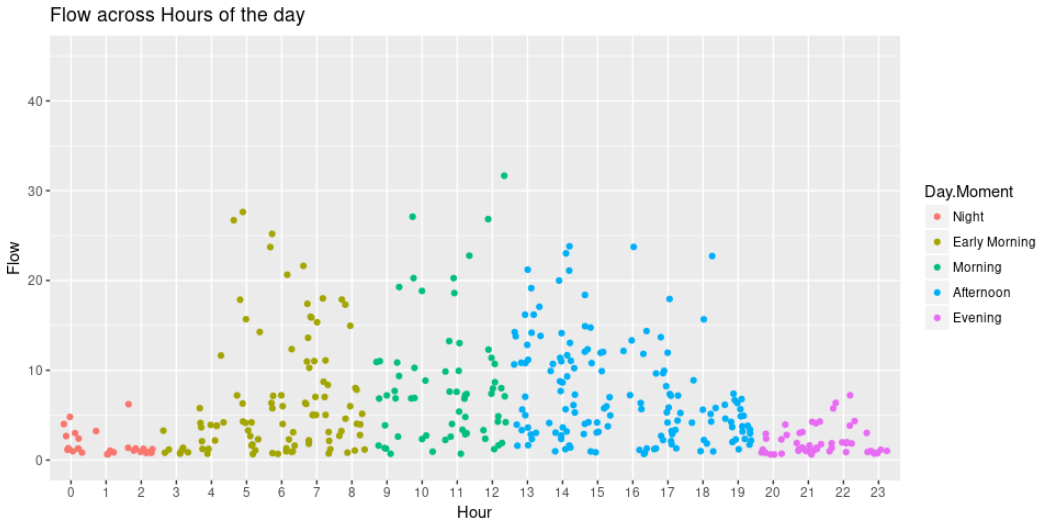

Figure 4.3: Flow across hours of the day.

However, this new variable was ultimately not included as a feature in any of the models since the behavior at any given point on the highway at a certain time t might be completely different from another. For example, road *E4N* might be congested at 8 o'clock in morning but not at 16 o'clock in the afternoon, whereas the observations from road *E4Z* might behave in the opposite manner. If one also takes into consideration the behavior of the

weekend vs. the weekdays, the interaction of among these variable gets more complex and the feature *Day Moment* cannot represent all this information. Therefore, when it comes to the behavior of the traffic flow across time, the new variables that were introduced are the *mean speed* and *mean density* for each sensor per weekday and hour of the day, which makes these features the *contextual attributes* for the traffic flow at both place and time.

4.3.3 Accidents

Since the dataset for accidents only contained information for the hour in which the incident took place but not the minute, it became necessary to plot the speed and density values for each of the accidents in order to determine the actual time frame of the event. In total, there were 66 accidents, but this subsection will only use one of them to illustrate the outcomes of such experiments and the conclusions that were drawn from them.

Taking the information from the nearest sensors to the event, Figure 4.4 shows the values of the speed and density variables for an accident that occurred on the 9th of March, 2012. In such figure, one can clearly see that the actual time frame of the event took place between the 16:10 and 16:30 hours, since this is the range in which the speed and density experience an abnormal drop and raise respectively.

Figure 4.4: Speed and Density for the sensors where an accident took place.

Furthermore, by taking a look at each individual sensor, it is possible to see how the effects of the incident affect each of the lanes in the section of the road where the event took place. To illustrate this, Figures 4.5 and 4.6 show the data for the sensors located on road E75W and lane 1 for the KM references 6880 and 7075. Their behavior is very similar, and this similitude leads us to conclude that the accident occurred on lane 1 of road E75W.

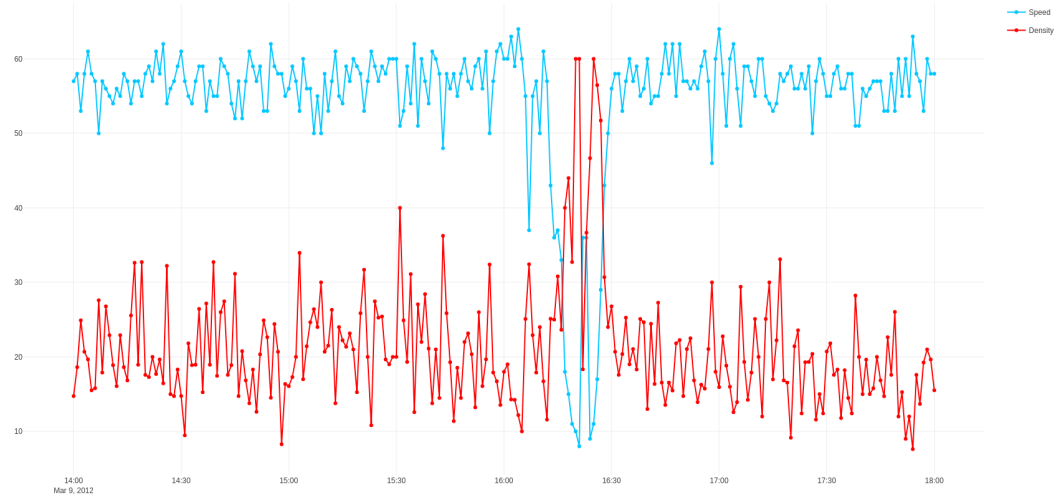

Figure 4.5: Behavior of the accident for lane 1 on road E75W at KM 7075

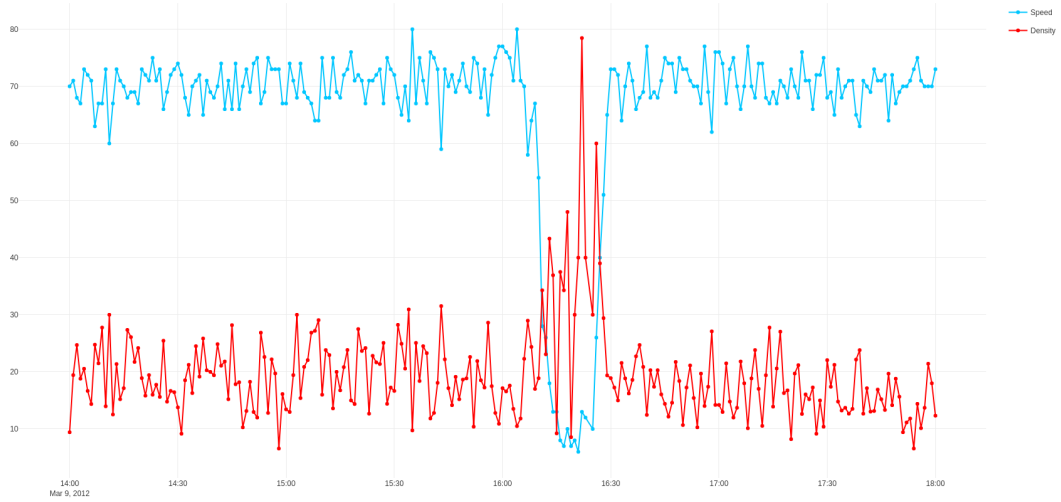

Figure 4.6: Behavior of the accident for lane 1 on road E75W at KM 6880.

Regarding the lanes 2 and 3 from the road E75W at KM reference 7075, and by looking at Figures 4.7 and 4.8, it is clear that there is a congestion within the time frame of the accident. However, the drop and raise that can be seen for the speed and density it's not as notorious as the one shown in the previous figures. The reason behind this is that the vehicles are most likely changing from lane 1 to lanes 2 and 3 in order pass through the accident point.

After analyzing the behavior for each incident, only the time frame from which the accident took place is labeled as an *anomaly caused by and accident*, and a rule was defined in order to label them correctly: *if the speed of the current observation is* ***lower*** *than the mean speed of that particular sensor and hour of the day* ***minus*** *3 times the standard deviation for that sensor at that hour of the day, then it is an accident.* This comes from looking at Figure 4.9,

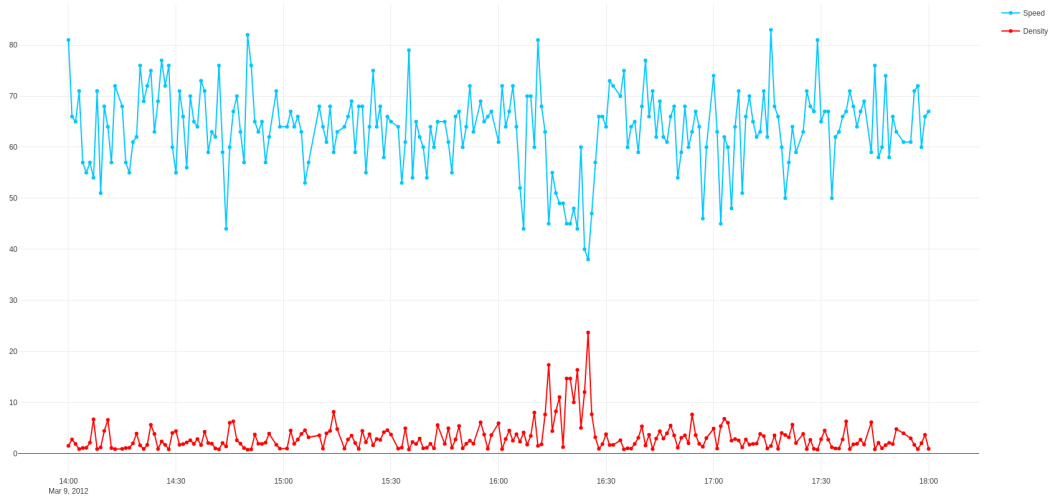

Figure 4.7: Behavior of the accident for lane 2 on road E75W at KM 7075

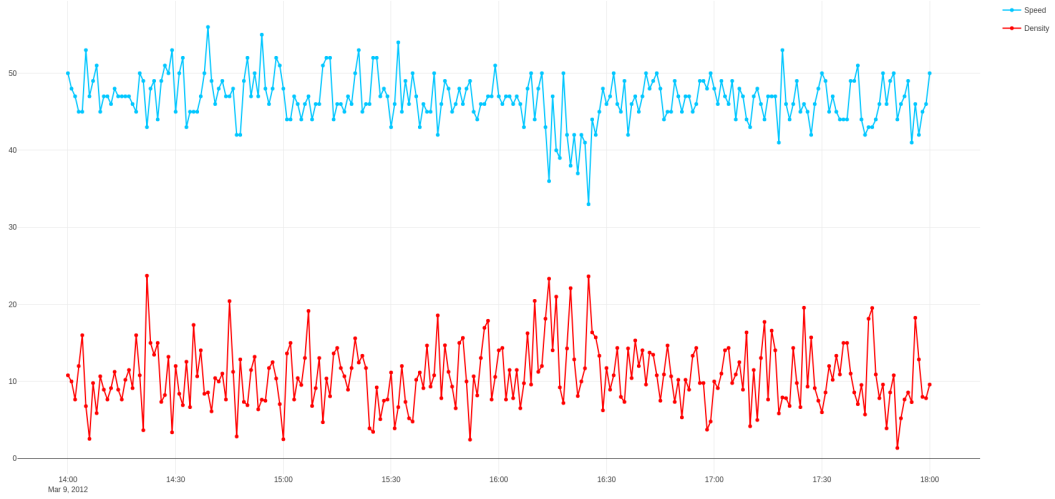

Figure 4.8: Behavior of the accident for lane 3 on road E75W at KM 7075

where it can be seen that at 16 o'clock the mean speed is 79km/hr and its standard deviation is 9.94km/hr, and comparing it with the speed values seen in the time frame of the accident of Figure 4.5, where the values are at least lower than the mean speed minus 3 times the standard deviation of the speed for that sensor and hour of the day.

4.3.4 Chosen Labels

The primary output for the data exploration phase can be found in Table 4.3. It contains a short description for each of the categories defined to separate the dataset observations from abnormal and normal behavior, as well as offering a reason, if available, for the appearance of the former. The observations are

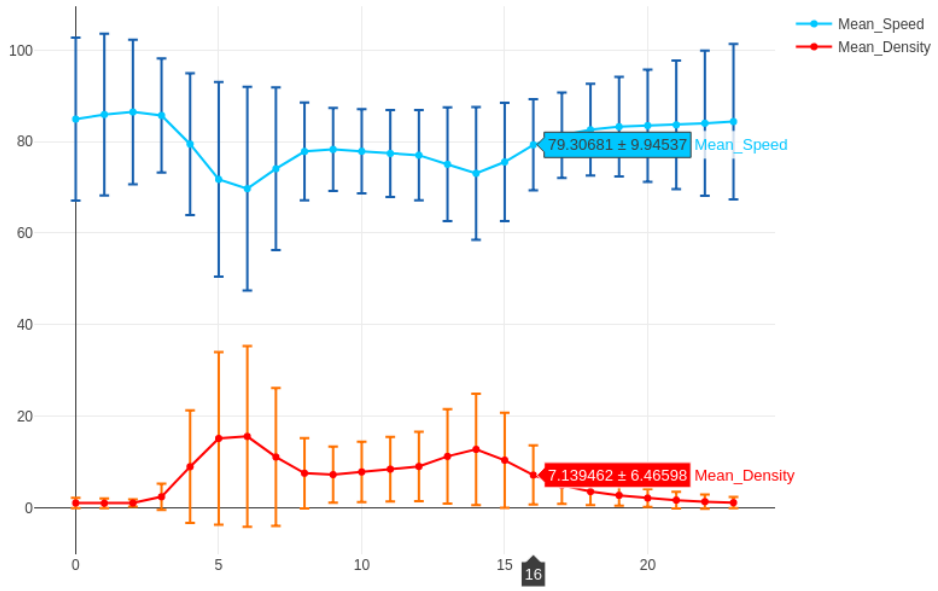

Figure 4.9: Mean and Standard Deviation of the Speed and Density for Road E75W on lane 1 at KM 7075.

classified applying the following criteria:

- If speed and/or flow are not within the OK range values defined in Table 4.1, the observation is labeled as *Error Code*.
- If flow is higher than 50 veh/min, the observation is labeled as *Sensor Error*.
- If flow and density values lie within the not congested region defined by the red area in Figure 4.2, then the observation is labeled as *Not Congested*.
- If the instance lies outside the *Not Congested* boundaries, the following is verified:
  - If the speed and density values are normal for that particular sensor using the mean and standard deviation values by hour and day of the week, the instance is labeled as *Normal Congestion*.
  - If it is not a normal congestion, we check for abnormal behavior. If the data point verifies the conditions for detecting an accident (See section 4.3.3), it is label as *Abnormal Congestion - Accident*. If not an accident and weather is either *snow, fog or rain*, the instance is labeled as *Abnormal Congestion - {weather condition}*. If not, the observation is labeled as *Abnormal Congestion - Other*.

Table 4.3: Description of the labels used to classify the dataset. AC = Abnormal Congestion.

Label	Description
Error Code	Speed or Flow are outside valid range. See Table 4.1
Sensor Error	Any flow value higher than 50
Not Congested	Observation lies in normal behavior range
Normal Congestion	Normal behavior for a particular sensor and time
AC - Accident	Abnormal behavior caused by an accident
AC - Rain	Abnormal behavior caused by rain
AC - Fog	Abnormal behavior caused by fog
AC - Snow	Abnormal behavior caused by snow
AC - Other	Abnormal behavior caused by reasons unknown

4.4 Model Training

Multiple datasets were used for the experiments of this phase. The main one is the full dataset of observations recollected from the sensors plus information about weather and accidents. Another one is a reduced version of the former where the observations labeled as accidents represent 25% of the whole dataset. Table 4.4 presents the percentage of data assign to each of the labels.

Table 4.4: Percentage of observations assign to each of the labels.

Label	Full Dataset	Reduced Dataset
Sensor Error	0.41431%	1.238%
Abnormal Congestion - Other	2.16629%	1.501%
Error Code	23.29142%	3.794%
Abnormal Congestion - Snow	0.07408%	0%
Not Congested	72.58853%	65.261%
Abnormal Congestion - Accident	0.00017%	24.69%
Normal Congestion	1.2504%	3.515%
Abnormal Congestion - Rain	0.15372%	0%
Abnormal Congestion - Fog	0.06106%	0%

The motivation behind the creation of the latter dataset is to fix the imbalanced nature of the full one and help improve the performance of the model regarding the detection of accidents. Abnormal behavior caused by the weather is not included in this version since the main concern was to improve the detection of accidents.

The remaining datasets were also created to solve the imbalanced distri-

bution of the instances across the categories. However, instead of removing observations, this time the accident instances were replicated until they could match either the *abnormal congestion - other* label or the *not congested* one. More details about these datasets can be found in section 4.4.3.

4.4.1 Full Dataset

In section 3.2.1, it was mentioned that an average of the previous 20min for the *Speed* and *Density* features would be added to improve the results for the accident label. The motivation behind this decision can be seen in Figure 4.10. This experiment shows that the F-score value for the *abnormal congestion - accident* label is 0% when these two features are no present in the model. A detailed description of the model, and an analysis of the results obtained for the other classes, can be seen in the following subsection. The results obtained were very similar for both experiments.

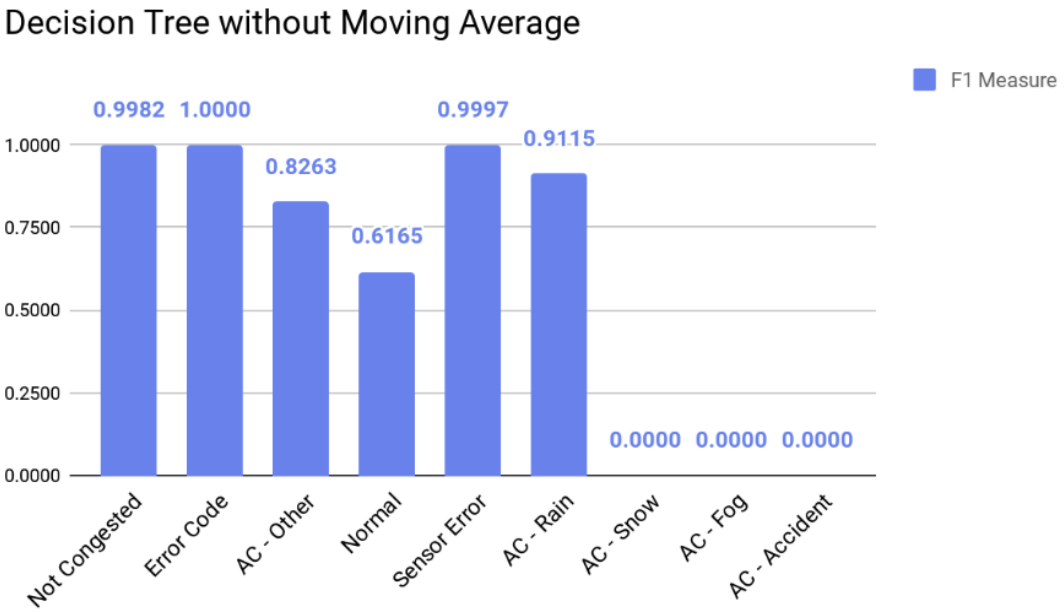

Figure 4.10: Decision Tree without Moving Average results.

4.4.1.1 Decision Trees

The decision tree had a maximum depth of 10. The selected features for predicting the label were density, speed, flow, weather, hour, mean density, mean speed, standard deviation speed, standard deviation density, moving average speed and moving average density. The features road, lane and km represent a single sensor, but they were not included because the means and

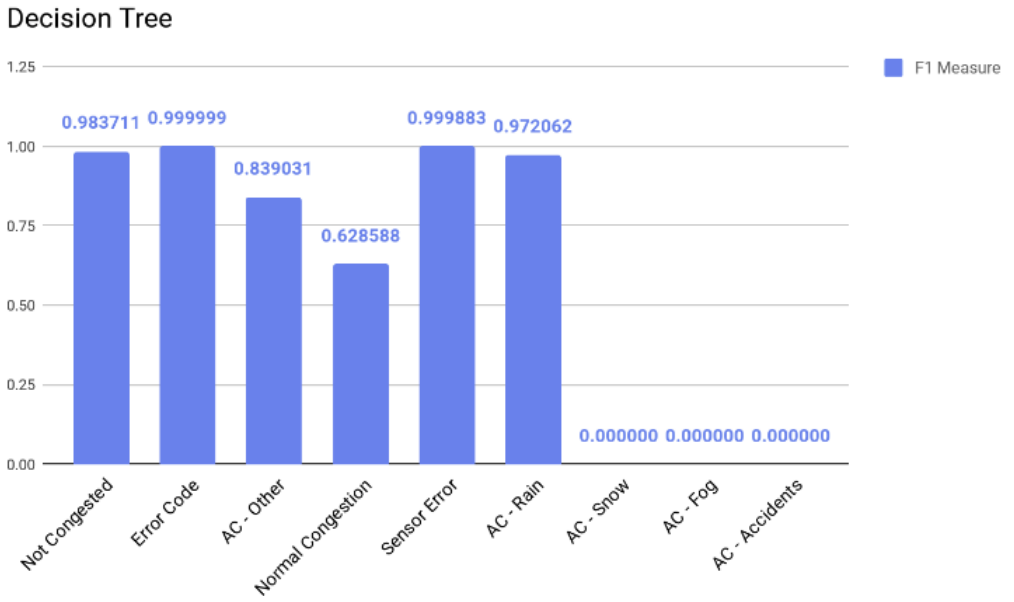

Figure 4.11: Decision Tree results using the full dataset.

standard deviations for speed and density offer the necessary information for each one of them.

Figure 4.11 illustrates the F-score values obtained for each of the classes. The high results obtained for *error code* and *sensor error* were expected because there are clear thresholds that define their boundaries and the tree was capable of detecting them. A similar situation also occurs with the *not congested* category. Even though the boundaries are not as clear (see Figure 4.2), most of the observations reflect a not congested behavior and the tree most likely used a wider threshold to define this region, such as the one defined by the discontinued brown lines in Figure 4.2. This also explains the low score for *normal congestion* since the tree is most likely classifying them as *not congested* or *abnormal congestion - other*. As for the abnormal congestion regarding weather, it is very likely that most of the observations with rain as the value for their weather feature were classified as *abnormal congestion - weather*. This means that the tree classifies instances in this category just by checking if the weather feature is rain. However, the opposite is the case for fog and snow and this might be the reason why the tree is not capable of detecting them. Finally, even with the presence of the moving average features, accidents are most likely not being detected because they only represent 0.0017% of the data and are most likely being classified as *abnormal congestion - other*. Moreover, regarding the accuracy, the model was able to correctly classify 98.27% of the observation. This is explained by the fact that almost 95% of the dataset falls under the categories of *error code, sensor error or not congested* and these labels obtained a high F-score when compared to others.

4.4.1.2 Random Forest

For the training of this model, both the maximum depth and the selection of features are the same as the one used in the Decision Tree with the full dataset experiment. The number of maximum trees was set to 10.

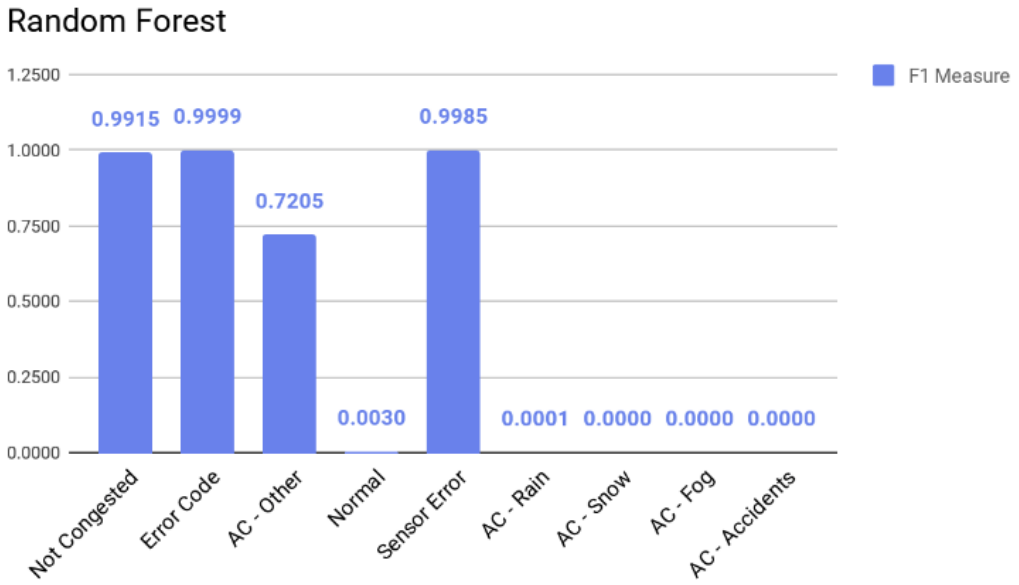

Figure 4.12: Random Forest results using the full dataset.

Figure 4.12 illustrates the F-score values obtained for each of the classes. As it was the case of the decision tree with the full dataset, and for the same reasons, *error code, sensor error and not congested* obtained the highest results. The low score obtained for *normal congestion* can be explained by the fact that this technique creates multiple trees using a random subset of features and this label depends on the means and standard deviations for the speed and density. However, the model probably created different trees where these features are not together. Furthermore, even if a tree correctly classifies an observation in this category, the others will most likely label it as *not congested* or *abnormal congestion - other*. A similar situation also happens to *abnormal congestion -rain* and all of their instances will probably go to *abnormal congestion - other*. The same problems found for labels *fog, snow and accidents* in the decision tree model are also present in this one. Finally, the accuracy for this model is 97.95% because almost 95% of the dataset falls under the categories of *error code, sensor error or not congested* and these labels obtained a high F-score when compared to others.

4.4.1.3 Logistic Regression

The model uses a regularization parameter of 0, a maximum number of iterations of 100, L2 penalty for elastic net and the features are standardized before fitting the model. The selection of features is the same as the one used in the Decision Tree and Random Forest with the full dataset experiments.

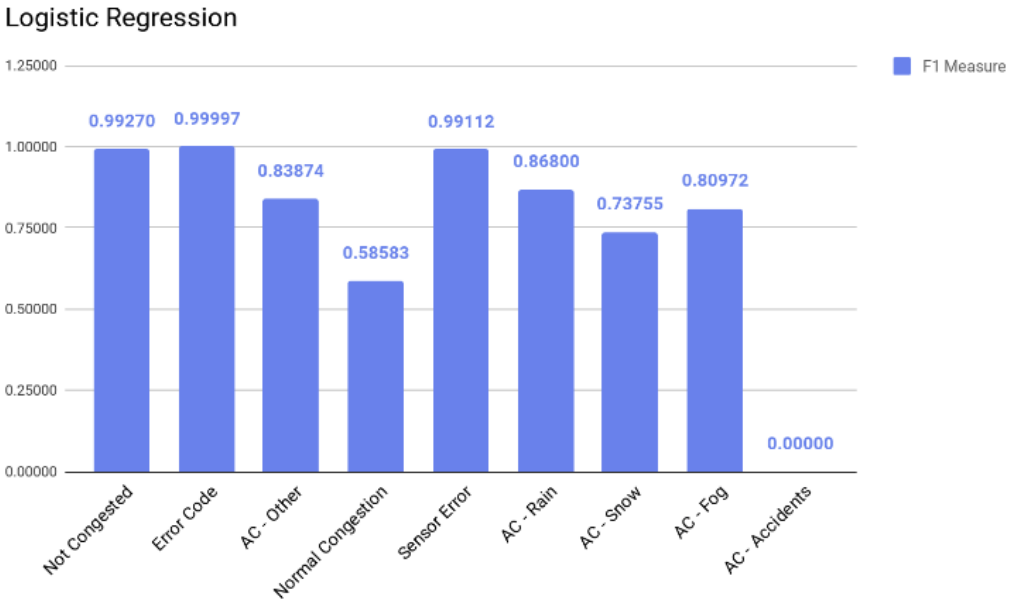

Figure 4.13: Logistic Regression results using the full dataset.

Figure 4.13 illustrates the F-score values obtained for each of the classes. As it was the case of the decision tree and random forest with the full dataset, *error code, sensor error and not congested* obtained the highest results. However, instead of rules, the model calculates a set weights per feature and category to determine the output. Nevertheless, these weights still represent a wide threshold and don't capture the actual boundaries for the *not congested* category, which ends up affecting the *normal congestion* score. Since this model takes into account the interaction between all variables, a better score for *normal congestion* was expected. However, the weights for detecting a *not congested* instance most likely return a higher probability than the ones used for selecting *normal congestion*. Regarding abnormal congestion caused by weather, this model presents a better result than the previous. This is due to the importance the weights for detecting these categories give to the corresponding weather feature. However, this is clearly affecting *abnormal congestion - other* as not all abnormal congestion are due to bad weather. Regarding accuracy, the model reached a 98.64% due to the high scores obtain by the labels that represent the 95% of the dataset: *error code, sensor error or not congested*.

4.4.1.4 Conclusion

Overall, the Logistic Regression model had better results since it was able to take into account the weather feature to classify observations as abnormal congestion caused by its corresponding weather condition. Decision Trees was the second best with similar results to the former and was slightly better at detecting normal congestion. Nevertheless, it could not label all the weather anomalies. Random Forest obtained the worse results and could not detect normal congestion. Finally, none of the models were able to detect accidents.

4.4.2 Relaxed Models

A more simplistic approach for labeling the dataset was defined during the early stages of the project. The criteria for this classification is the following:

- If speed and/or flow are not within the OK range values defined in Table 4.1, the observation is labeled as *Error Code*.
- If flow is higher than 50 veh/min, the observation is labeled as *Sensor Error*.
- If the observation verifies the conditions for detecting an accident (See section 4.3.3), it is labeled as *Accident*.
- If density is higher than 35 and the day is part of the weekend, the observation is labeled as *Normal Congestion*.
- If density is higher than 35 and the day is not part of the weekend, but the day moment (see Table 3.3) is early morning or afternoon, the observation is labeled as *Normal Congestion*.
- If density is higher than 35, the day is not part of the weekend and the day moment is morning night or evening, the observation is labeled as *Abnormal Congestion*.
- If none of the conditions above are met, the observation is labeled as *Not Congested*.

The application of this criteria led to the creation of *relaxed models*. These models were defined with the same parameters used in section 4.4.1. The selected features were the same, but the weather was not included. Day Moment was used and a new binary variable called Is Weekend was added.

Figure 4.14 shows the results obtained from training the models to predict these labels and they are very good. Almost all the categories reach an F-score higher than 90%. However, accidents still get an F-score of 0% due to the imbalanced nature of the dataset. Furthermore, the behavior of the flow for accidents resembles the one for *abnormal congestion*, which leads the classifiers

to label accidents as members of this category. As for the accuracy, all models scored close to 99.9%.

Regardless of the good scores obtained, this approach cannot be used. The thresholds that define *not congested* behavior is too wide, it does not considered the behavior of flow per sensor and it assumes that the traffic flow for all of them is the same.

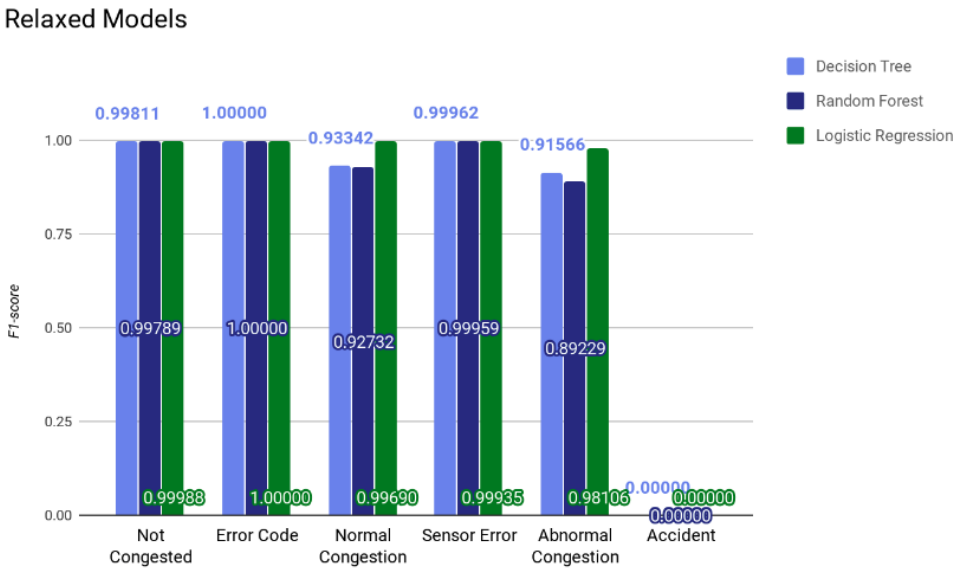

Figure 4.14: Relaxed Models results.

4.4.3 Under-Sampling vs Over-Sampling

One of the main issues encountered was the imbalanced nature of the dataset. To solve it, there are two approaches that are commonly used: *under-sampling* and *over-sampling* [31] [16] [21]. The former aims at removing observations from the majority class while the latter increments the number of observations from the minority label by replicating its instances.

Under-sampling can be useful when run time and storage capacity are limited. However, the resulting class might end up being an inaccurate representation of the population. On the other hand, over-sampling prevents information loss but increases the risk of overfitting the model.

4.4.3.1 Under-Sampling of the Majority Classes

For this technique, enough instances were removed from the majority classes to make sure that 25% of the observations were from the accident

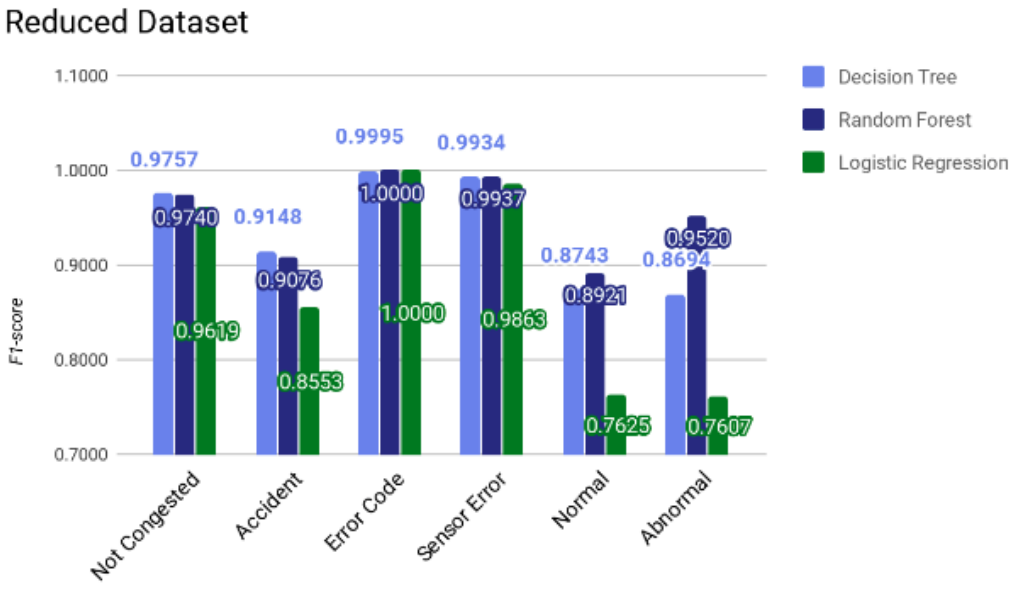

Figure 4.15: Under-sampled dataset results.

category. The relaxed model approach was also used for this experiments. The resulting F-score from the evaluation of each model can be seen in Figure 4.15.

Labels *error code, sensor error and not congested* scored highly across all techniques, but still a bit lower than the relaxed model with the full dataset. This is understandable considering that their combined portion of the dataset is now 70% instead of the usual 95%. The score for *normal congestion* is also reduced across all models. This might be due to the higher presence of anomalies and the models not being able to distinguish normal congestion from abnormal behavior. The same can be said about the *accident* and *abnormal congestion* categories. They still score highly, but since *accidents* represent a bigger portion than *abnormal congestion*, the models are considering it the first choice. However, this is the opposite for Random Forest and it might be due to having trees that only include the features day moment, is weekend and density.

In general, Logistic Regression performs worse than the other models in this experiment. This makes sense because the labels were put using clear thresholds and trees will be better at detecting them. In contrast, the interaction of the variables in the Logistic Regression might create noise in the classification of a congested observation (accident, abnormal or normal). Besides, the reduced dataset contains less than 1% of the full set and there is a higher number of accidents than abnormal congestion. These conditions are surely affecting the classification for the latter label.

4.4.3.2 Over-Sampling of the Accident Data

For this approach, all the features from each of the observations from the *abnormal congestion - accident* label were replicated enough times until its proportion could match that of the *abnormal congestion - other* category. The reason behind this choice is that accidents can be mistaken as abnormal congestions - other since their features show similar values. Table 4.5 shows the distribution of the instances across the categories.

Table 4.5: Percentage of observations assign to each label.

Label	**Proportion**
Sensor Error	3.26%
Abnormal Congestion - Other	15.07%
Error Code	59.96%
Abnormal Congestion - Snow	0.58%
Abnormal Congestion - Accident	15.29%
Normal Congestion	4.14%
Abnormal Congestion - Rain	1.21%
Abnormal Congestion - Fog	0.48%

Furthermore, all instances labeled as *not congested* were removed from the dataset. This was done to create a model that could distinguish an accident from within the abnormal congestion area of the curve as seen in Figure 4.2. Moreover, the instances were classified following the criteria described in section 4.3.4.

The results from this experiment are shown in Figures 4.16 and 4.17. The former includes the moving average of *speed* and *density* features while the latter ignores them. Their results are quite similar. However, the outcome for the *abnormal congestion - fog* label is much better when the moving average features are used. Moreover, and similar to previous experiments, labels *error code and sensor error* scored highly across all techniques. However, *normal congestion* obtained a 0% for the Decision Tree model, 49% for Random Forest and 67% for Logistic Regression. The score for the first model can be explained due to the class only having 4.14% of the total number of instances and the tree not being able to distinguish among this label and abnormal congestions. A similar problem can be seen in the other models but having multiple trees, in the case of Random Forest, and having all variables interact with each other, in the case of Logistic Regression, helps improve this issue. The same can be said about the scores obtained for the labels *abnormal congestion - other* and *abnormal congestion - accident*, which got an average score of 74% and 64%, respectively, across all models. Finally, the weather categories obtained an average score of 95% for the Logistic Regression model. Unfortunately, the

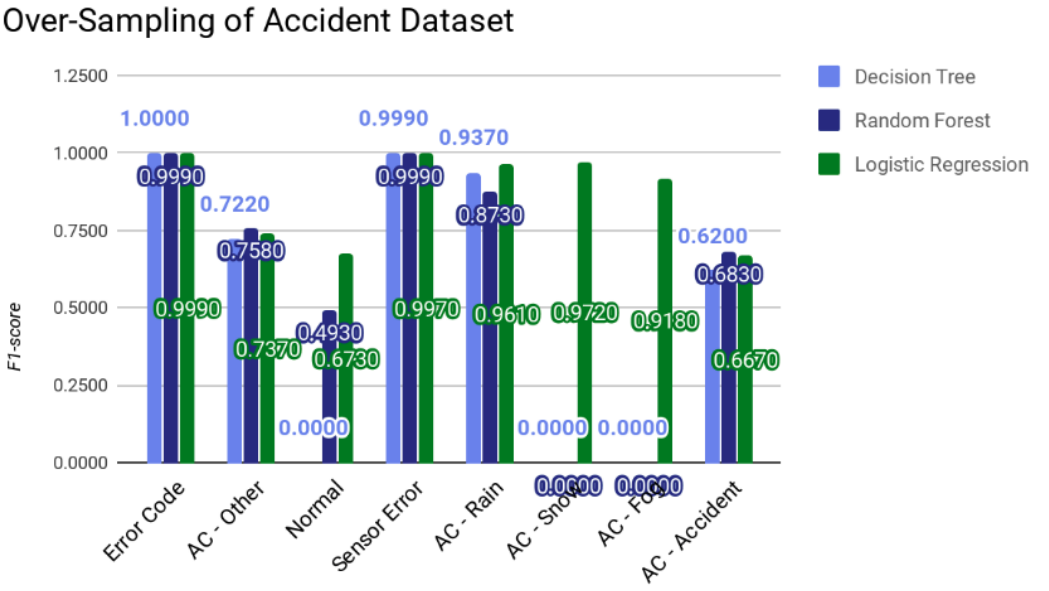

Figure 4.16: Over-sampling of accident dataset results.

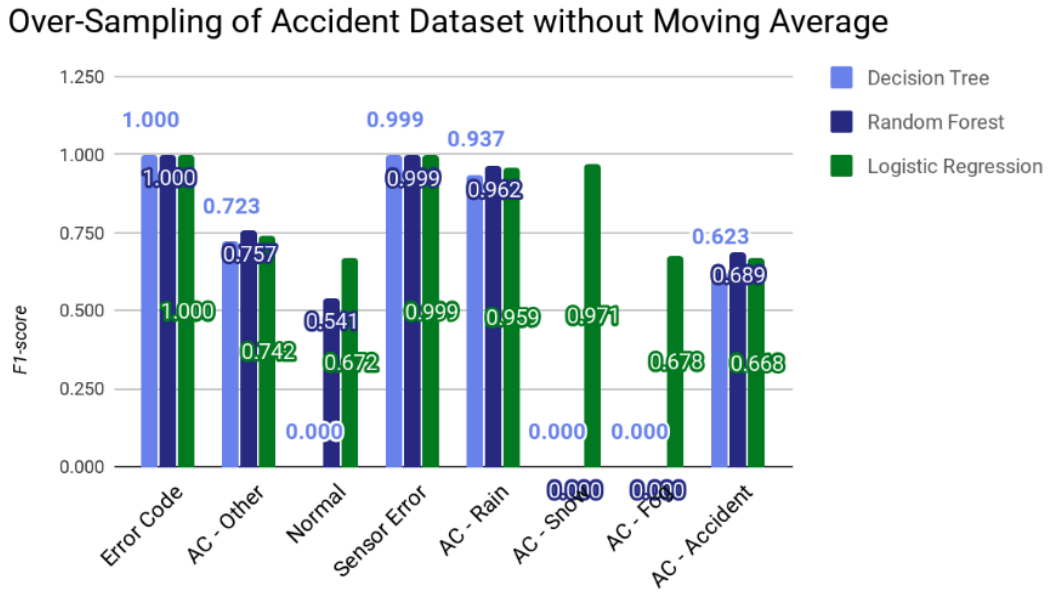

Figure 4.17: Over-sampling of accident dataset without moving average results.

other models could only detect anomalies caused by *rain*, which is very similar result to what was obtained in section 4.4.1.

In contrast to what was seen in the section 4.4.3.1, Logistic Regression turned out to be the better model. This is understandable because the labels were chosen following the criteria explained in section 4.3.4, where the thresholds are not as distinguishable as in the relaxed models. Having increased the amount of accident data improved the results for the model and no information was

lost. However, even if the results for this test are better than previous ones, there is a risk of the model being overfitted.

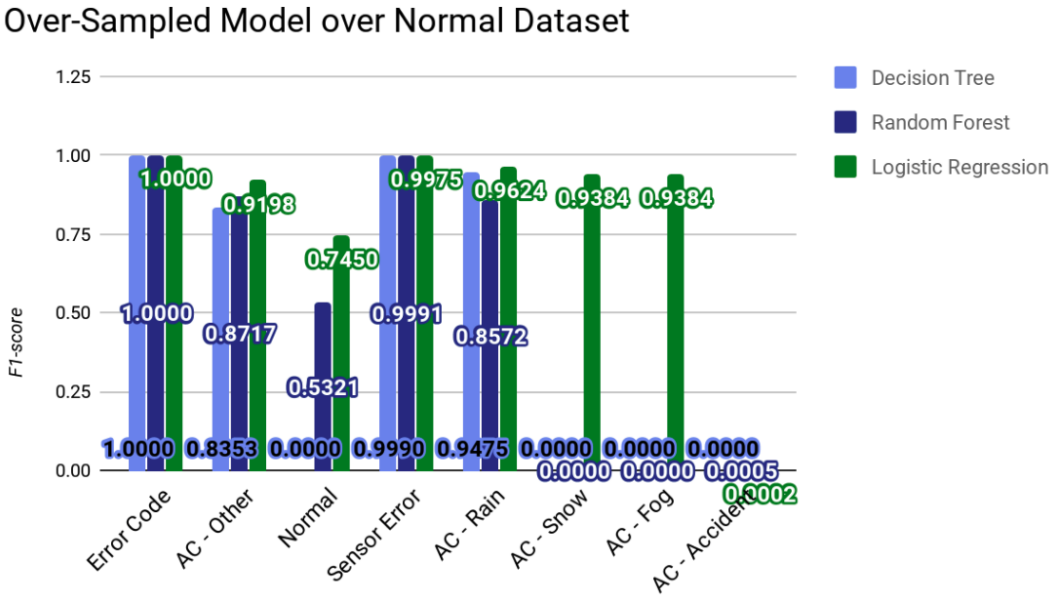

Figure 4.18: Over-sampled model over normal dataset results.

Figure 4.18 illustrates the results of an experiment in which the resulting model was used to classify the observations of the normal dataset. Unfortunately, the overall result shows similar values to those of section 4.4.1. This confirms that the application of over-sampling of the accident dataset resulted in an overfitted model.

4.4.3.3 Inclusion of Not Congested Label

A new experiment, similar to the previous one, was performed. The configuration is the same. However, this time the *Not Congested* observations were included in the dataset. The reason behind this decision was to verify if accidents could be detected even if the presence of the *Not Congested* observations is included. Table 4.6 illustrates the distribution of the dataset for each of the categories.

Table 4.6: Percentage of observations assign to each label.

Label	Proportion
Sensor Error	1.05%
Abnormal Congestion - Other	2.38%
Error Code	24.18%
Abnormal Congestion - Snow	0.04%
Not Congested	69.05%

Label	Proportion
Abnormal Congestion - Accident	2.18%
Normal Congestion	1%
Abnormal Congestion - Rain	0.09%
Abnormal Congestion - Fog	0.03%

The results can be seen in Figure 4.19. The F1-scores were not good. All models were capable of classifying the *Not Congested* instances. However, since the *abnormal congestion - other* and *accident* data were still a small percentage of the dataset (2.5% each) and the *Not Congested* label was 70%, the models were able to correctly classify that latter, but had trouble classifying the former. The low F1-score for the accidents is most likely due to the fact that the models are classifying them as *abnormal congestion - other*.

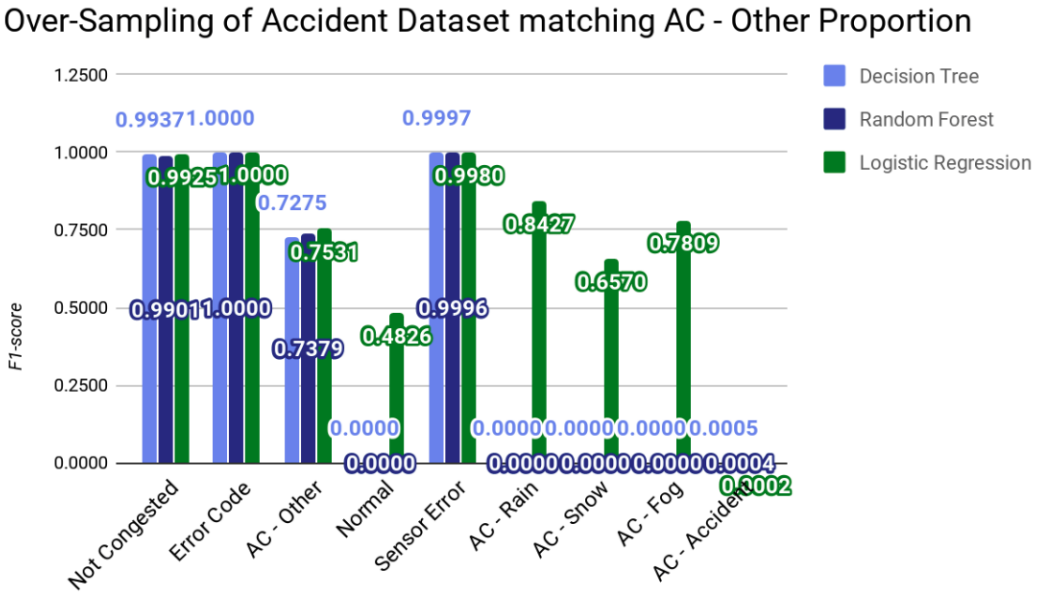

Figure 4.19: Over-sampling of accident dataset matching AC - Other results.

Regarding precision and recall metrics, Tables 4.7 and 4.8 illustrate the obtained scores. The initials of the machine learning techniques are used to distinguish the models in the following way: DT = Decision Tree, RF = Random Forest and LR = Logistic Regression.

Table 4.7: Precision for over-sampled dataset matching AC - Other.

Label	DT	RF	LR
Not Congested	0.9951	0.981	0.9887
Error Code	1	1	0.9999
Abnormal Congestion - Other	0.6023	0.7932	0.9272

Label	DT	RF	LR
Abnormal Congestion - Accident	0.0003	0.0002	0.0001
Sensor Error	0.9999	1	0.9964
Normal Congestion	0	0	0.671
Abnormal Congestion - Rain	0	0	0.8591
Abnormal Congestion - Snow	0	0	0.6706
Abnormal Congestion - Fog	0	0	0.7682

Table 4.8: Recall for over-sampled dataset matching AC - Other.

Label	DT	RF	LR
Not Congested	0.9922	0.9994	0.9963
Error Code	1	1	1
Abnormal Congestion - Other	0.9183	0.6898	0.634
Abnormal Congestion - Accident	0.2229	0.2648	0.576
Sensor Error	0.9995	0.9992	0.9995
Normal Congestion	0	0	0.3769
Abnormal Congestion - Rain	0	0	0.8269
Abnormal Congestion - Snow	0	0	0.6439
Abnormal Congestion - Fog	0	0	0.794

The scores indicate that the models are not precise for accident classification. This means that there were instances incorrectly labeled as accidents. As for the recall, the scores show that the models can detect some of the accidents but not enough to be considered robust.

Another experiment was performed with the same configuration. However, this time the accident instances were replicated until they could match the proportion of Not Congested observations. This resulted in 30% of the proportion for each of the categories. The results were very similar to the previous one, but the *accident* and *abnormal congestion - other* F1-scores were swapped. This is most likely due to the accident label representing a much bigger proportion of the dataset.

4.4.4 Other Experiments

4.4.4.1 Single Sensor Training

Judging by how unfavorable the results of previous experiments have been for the *abnormal congestion - accident* label, a new approach was tested. Instead of training a general model for all sensors, there could be a model for

each one of them. However, there is a total of 2000 sensors, so only a single sensor was used for this approach. The chosen sensor is located on lane 1 of road E75W at KM 7075.

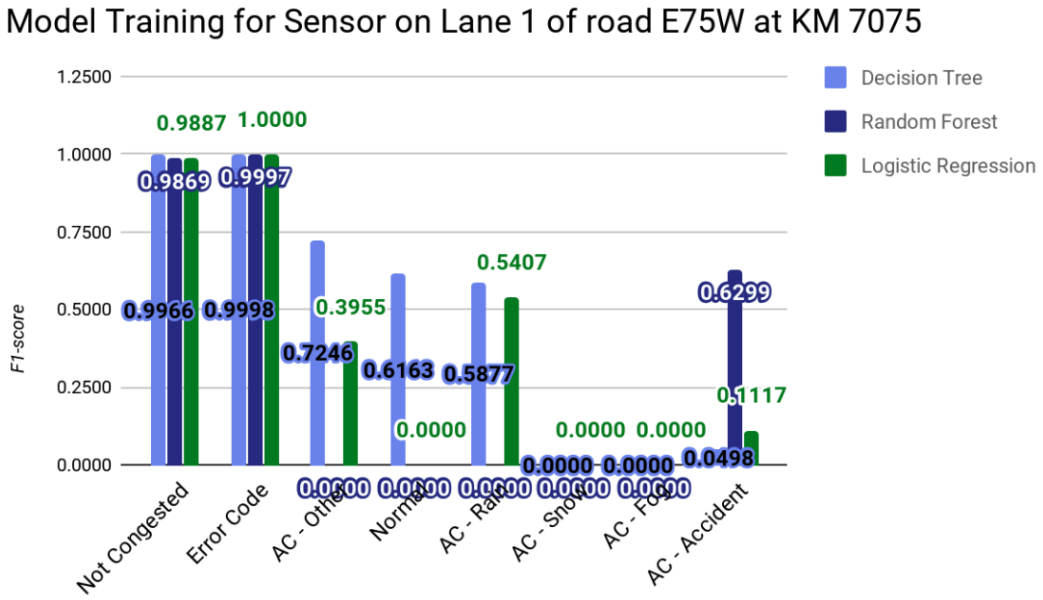

Figure 4.20: Model Training for Sensor E75W - 7075 on Lane 1 results.

The results can be seen in Figure 4.20. *Sensor errors* observations were not present for this sensor. The scores were average across all models. Random Forest was not capable of detecting *normal congestion*, *abnormal congestion - other* and *weather* instances. Decision Tree and Logistic Regression obtained the best results. The *accident* label got an F1-score of 4% for the former and 11% for the latter. This is because the accident data is still too similar to the *abnormal congestion - other* observations. The 0% weather score for *fog* and *snow* is most likely due to not having enough data with those labels. The *normal congestion* and *abnormal congestion - other* categories received a lower score in the Logistic Regression model when compared to the Decision Tree one.

Tables 4.9 and 4.10 illustrate the obtained scores for the recall and precision. Just like in previous sections, the initials of the machine learning techniques are used to distinguish the models in the following way: DT = Decision Tree, RF = Random Forest and LR = Logistic Regression.

Table 4.9: Precision for single sensor training.

Label	**DT**	**RF**	**LR**
Not Congested	0.9959	0.9742	0.9793
Error Code	1	1	1
Abnormal Congestion - Other	0.8026	0	0.7811

Label	DT	RF	LR
Abnormal Congestion - Accident	0.0259	0.8163	0.0621
Normal Congestion	0.5976	0	0
Abnormal Congestion - Rain	0.9135	0	0.8007
Abnormal Congestion - Snow	0	0	0
Abnormal Congestion - Fog	0	0	0

Table 4.10: Recall for single sensor training.

Label	DT	RF	LR
Not Congested	0.9973	1	0.9982
Error Code	0.9997	0.9994	1
Abnormal Congestion - Other	0.6604	0	0.2648
Abnormal Congestion - Accident	0.6795	0.5128	0.5513
Normal Congestion	0.6362	0	0
Abnormal Congestion - Rain	0.4332	0	0.4082
Abnormal Congestion - Snow	0	0	0
Abnormal Congestion - Fog	0	0	0

The scores indicate that Random Forest is precise and quite robust for classifying accidents. However, the other models are not precise and are less robust when compared to Random Forest.

4.4.4.2 Feed-forward Neural Network Model

During the development of the work, an experiment using a different machine learning technique than the previously described was applied. The algorithm is called *Feedforward Neural Network* [10] and it is similar to *Logistic Regresssion* [25]. The stages of the algorithm found in the latter can also be found in the former. However, Feedforward Neural Networks have *hidden layers*, meaning that instead of having just an output layer (see section 2.3), the network can have many where the output of one is the input of the following. Moreover, each layer can have a different amount of neurons where an activation function is applied to the input in order to generate an output. Because of these reasons, defining the best architecture of a neural network for a given problem is a complicated task.

The architecture used for this experiment was a simple one: 2 hidden layers, the first of them with 100 neurons and the second with 50, and sigmoid was used as the activation function. The relaxed model approach and the full dataset were used for training this model.

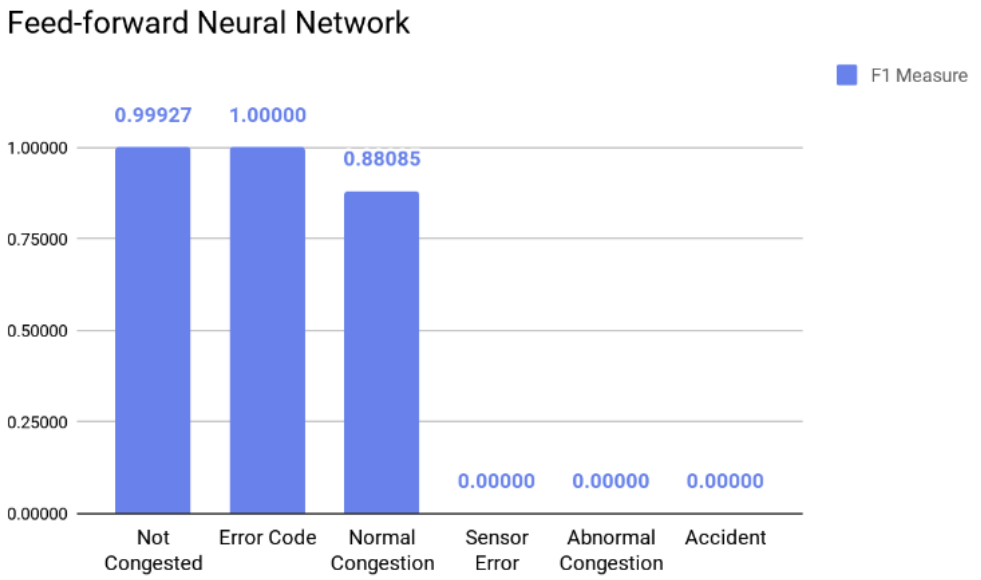

Figure 4.21: Feed-forward Neural Network results.

The results of this experiment can be seen in Figure 4.21. Unfortunately, the output is not favorable. The network scored 99.9% for the *not congested* class and 100% for the *error code* label, just like the previous techniques. However, it scored an 88% for the *normal congestion* category which is worse than what the previous models were able to achieve. As for the other labels, the resulting score was a 0% for all of them.

As it was mentioned earlier, the training of a neural network is a complicated task that requires tuning. The chosen architecture for this experiment was quite simple and was not the best fit for the goal of this thesis. Further development of this model might bring better results.

Chapter 5

Discussion

During the development of the work, one of the most difficult challenges came from having to not only separate the normal from the abnormal observations, but to go a step further and *sublabel* the different abnormal observations based on the nature of their caused. An approach used to deal with this situation was to include datasets containing information about accidents and weather, but even though they were quite useful for representing new types of anomalies, defining the thresholds was not an easy task. The boundaries that were applied still leave room for the misclassification of instances, something that became an important issue when training different supervised classification models, as they struggle to effectively distinguish among the subclasses of anomalies. The training of a model was also important since it helps make stream processing faster than following a set of conditions.

Furthermore, anomalies are uncommon observations that don't appear too often, which is why the dataset turned out to be imbalanced. At the very early stages of the project, this issue made us switch the focus from the *accuracy* to the *F-score* as the main evaluation metric for evaluating performance. It indicates how precise and robust the model is, and can be used to evaluate the classification for each of the classes in order to demonstrate the effectiveness of the approach. Nevertheless, the imbalanced dataset resulted in models that were capable of detecting normal behavior but that were not so effective for distinguishing the different types of abnormal observations.

To fix the imbalanced nature of the dataset issue, under-sampling and over-sampling techniques were applied. They both returned good results, but the former approach reduces the number of instances from the majority classes, leading to information loss. However, the latter replicated the number of accidents, giving the models more information to work with, but it resulted in an overfitted model. Moreover, a feedforward neural network was also trained. However, the results were not favorable because more tuning is required.

Chapter 6

Conclusion

Multiple models for detecting anomalies in the traffic flow of Stockholm were developed using Decision Trees, Random Forest and Logistic Regression methods. The first step was to perform a data exploration analysis on a sensor dataset that processed traffic information from the highways of Stockholm to separate the normal behavior (not congestion) from the abnormal (congestion due to rush hours, accidents or weather) and label it accordingly.

We touched upon the issues that arise both during the data exploration and model training phases. Based on the results, it was concluded that Logistic Regression turned out to be a slightly better model than the rest in terms of how precise and robust it is at classifying data into the different labels. Nevertheless it failed to correctly detect accidents due to the imbalanced nature of the data. This led to the training of a model using a smaller dataset containing 25% of the accidents. The detection of such events improved considerably. However, the dataset was labeled using the relax model approach and could not be considered representative enough for the behavior of the traffic flow. Instead, the accident data was replicated as a way to avoid losing information from other classes. This returned good results but the model turned out to be overfitted.

Finally, a streaming application was designed to: (1) read the information from the sensors, (2) group them based on the road, lane and km reference variables, (3) calculate the average for speed and density of the last 20 minutes and then (4) input the observation and the value of the window into the best model to (5) get its corresponding label.

6.1 Future Work

At the beginning of the project, we wanted to include a dataset containing information regarding important events occurring in Stockholm, such as concerts

or sport games, since this kind of occurrences would have probably cause congestion outside the peak hours. Unfortunately, no such data was found but it would be interesting to add this feature and evaluate the performance of the model.

A technique that would have been interesting to try is Gradient Boosted Trees [9], which is a technique based on Decision Trees but that uses gradient descent to generate the decision rules. Unfortunately, the available API [1] for this tool was designed to work with pandas dataframe which only works locally and cannot be used in the hops cluster. A library called XGBoost4J [2] offers compatibility with Apache Spark dataframes, but it is only available for Java and Scala.

Finally, it would also be interesting to improve the training of the neural network presented in this work since they seem to work well for detecting accidents according to [19]. However, the dataset used in those experiments was balanced and the output was binary (observation is accident or not), so it might still not work as well for the problem presented in this thesis.

[1] https://xgboost.readthedocs.io/en/latest/model.html
[2] https://github.com/dmlc/xgboost/tree/master/jvm-packages

Bibliography

[1] T. Dietterich A. Fern A. F. Emmott, S. Das and W.-K. Wong. *Systematic Construction of Anomaly Detection Benchmarks from Real Data.* Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, 2013.

[2] Jan Ekman Ola Sellin Björn Lindström Stefan Larsen Anders Holst, Markus Bohlin. Statistical anomaly detection for train fleets. *AI MAGAZINE*, 2013.

[3] Guilherme O. CamposArthur ZimekEmail authorJörg SanderRicardo J. G. B. CampelloBarbora MicenkováErich SchubertIra AssentMichael E. Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 2016.

[4] P. Bock. *Getting It Right: R&D Methods for Science and Engineering, 1st ed.* San Diego: Academic Press, 2001.

[5] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[6] Victoria Cox. *Exploratory data analysis. Translating Statistics to Make Decisions.* Apress, 2017.

[7] Nouria Harbi Dewan Md. Farid and Mohammad Zahidur Rahman. Combining naive bayes and decision tree for adaptive intrusion detection. *International Journal of Network Security and Its Applications (IJNSA)*, 2010.

[8] Niklas Donges. The random forest algorithm. https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

[9] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *IMS 1999 Reitz Lecture*, 1999.

[10] Michael Georgiopoulos George Bebis. Optimal feed-forward neural network architectures. *University of Central Florida, Orlando, FL 32816 USA*, 1994.

[11] Prashant Gupta. Decision trees in machine learning. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

[12] Ryan Jay Herring. Real-time traffic modeling and estimation with streaming probe data using machine learning. *UC Berkeley Electronic Theses and Dissertations*, 2010.

[13] Muhammad Hussain Iqbal and Tariq Rahim Soomro. Big data analysis: Apache storm perspective. *International journal of computer trends and technology*, 19(1):9–14, 2015.

[14] Tomer Toledo Jamal Raiyn. Real-time road traffic anomaly detection. *SciRes*, 2014.

[15] Gaurav Jha. 6 sampling techniques: How to choose a representative subset of the population. https://blog.socialcops.com/academy/resources/6-sampling-techniques-choose-representative-subset/.

[16] X. Liu, J. Wu, and Z. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 2009.

[17] Armin Stahl Matthias Reif, Markus Goldstein. Anomaly detection by combining decision trees and parametric densities. *19th International Conference on Pattern Recognition*, 2008.

[18] Aditya Mishra. Metrics to evaluate your machine learning algorithm. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[19] Erdogan Dogdu Murat Ozbayoglu, Gokhan Kucukayan. A real-time autonomous highway accident detection model based on big data processing and computational intelligence. *IEEE International Conference on Big Data*, 2016.

[20] United Nations. Sustainable development goal. https://sustainabledevelopment.un.org/.

[21] Lawrence O. Hall W. Philip Kegelmeyer Nitesh V. Chawla, Kevin W. Bowyer. Smote: Synthetic minority over-sampling technique. *Artificial Intelligence Research 16 (2002) 321–357*, 2002.

[22] Mangesh K. Nichat Nitin R.Chopde. Landmark based shortest path detection by using a* and haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*, 2013.

[23] John Pickles. *Ground Truth. The Social Implications of Geographic Information Systems.* Guilford Press, 1994.

[24] Saimadhu Polamuri. How multinomial logistic regression model works in machine learning. http://dataaspirant.com/2017/03/14/multinomial-logistic-regression-model-works-machine-learning/.

[25] Sebastian Raschka. Machine learning faq. https://sebastianraschka.com/faq/docs/logisticregr-neuralnet.html.

[26] Lei Tang Refaeilzadeh, Payam and Huan Liu. Cross-validation. *Encyclopedia of database systems*, 2016.

[27] Saed Sayad. Decision tree - classification. http://www.saedsayad.com/decision_tree.htm.

[28] Akash Singh. Anomaly detection for temporal data using long short-term memory (lstm). *KTH Royal Institute Of Technology School Of Information And Communication Technology*, 2017.

[29] Statistics Solutions. What is linear regression? http://www.statisticssolutions.com/what-is-linear-regression/.

[30] SQLStream. Real time vs. streaming—a short explanation. http://sqlstream.com/real-time-vs-streaming-a-short-explanation/.

[31] Upasana. How to handle imbalanced classification problems in machine learning? https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/.

[32] Vipin Kumar Varun Cahndola, Arindam Banerjee. Anomaly detection : A survey. *ACM Computing Surveys*, 2009.

[33] Chris Veness. Calculate distance, bearing and more between latitude/longitude points. https://www.movable-type.co.uk/scripts/latlong.html.

[34] Greg Wood. Top streaming technologies for data lakes and real-time data. https://resources.zaloni.com/blog/top-streaming-technologies-for-data-lakes-and-real-time-data.

[35] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

TRITA TRITA-EECS-EX-2018:563