2018

Can Threshold-Based Sensor Alerts be Analysed to Detect Faults in a District Heating Network?

Liam Cantwell
Technological University Dublin, Ireland

Can threshold-based sensor alerts be analysed to detect faults in a District Heating network?

Dissertation submitted in partial fulfilment for a

M.Sc. in Computing *(Data Analytics)*

DT228B

School of Computing

Dublin Institute of Technology

Abstract

Older IoT "smart sensors" create system alerts from threshold rules on reading values. These simple thresholds are not very flexible to changes in the network. Due to the large number of false positives generated, these alerts are often ignored by network operators. Current state-of-the-art analytical models typically create alerts using raw sensor readings as the primary input. However, as greater numbers of sensors are being deployed, the growth in the number of readings that must be processed becomes problematic. The number of analytic models deployed to each of these systems is also increasing as analysis is broadened. This study aims to investigate if alerts created using threshold rules can be used to predict network faults. By using threshold-based alerts instead of raw continuous readings, the amount of data that the analytic models need to process is greatly reduced. The study was done using alert data from a European city's District Heating network. The alerts were generated by "smart sensors" that used threshold rules. Analytic models were tested to find the most accurate prediction of a network fault. Work order (maintenance) records were used as the target variable indicating a fault had occurred at the same time and location as the alert was active. The target variable was highly imbalanced (96:4) with a minority class being when a Work Order was required. The decision tree model developed used misclassification costs to achieve a reasonable accuracy with a trade-off between precision (.63) and recall (.56). The sparse nature of the alert data may be to blame for this result. The results show promise that this method could work well on datasets with better sensor coverage.

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others. Any works of others have been cited and acknowledged.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The data used in this study is real city network data provide for the use of data model development for the IBM IOC product.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: _____

Liam Cantwell

Date: ***15th June 2018***

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Susan McKeever for her guidance, suggestions and encouragement. I learned a great deal in the execution of this project and it was very reassuring to have someone to verify my assumptions and methodologies whilst always remaining positive.

Thank you to my colleague and friend Ray for reviewing this document and discussing model tuning approaches. Thanks also to Brendan who was on hand to discuss spatial query optimisation approaches.

I would also like to thank each of my work colleagues for their support over the 2 years of this part-time masters. In particular, my manager Frances who was very supportive of my studies, approved the funding and made sure that I had everything needed to complete the project.

Thanks to all my classmates and in particular to Pierluca and Ruari with whom I did many of the group projects. A big thanks to all my lecturers in DIT: John McAuley, Brenden Tierney, Damian Gordan, Deirdre Lawless, Luca Longo, Alex Ter-Sarkisov, Diana Carvalho e Ferreira, Sean O Leary, Loran Coyle, David Leonard, Bojan Bozic and Robert Ross.

Lastly, I would like to thank my friends and family who put up with me over the last 2 years. A very special thanks to my brother James for his support in reviewing my work.

Table of Contents

Abstract i

Declaration ii

Acknowledgements iii

Table of Contents iv

Table of Figures ix

Table of Tables x

Glossary xii

1 Introduction 1

1.1 Background 1

1.1.1 How are Work Orders generated? 1

1.2 Research Problem 2

1.2.1 Research question 4

1.3 Research Objectives 4

1.3.1 Hypotheses 4

1.4 Research Methodologies 4

1.5 Scope and Limitations 6

1.5.1 Single network, single city 6

1.5.2 Domain limited to District Heating 6

1.5.3 No explicit target variable in the original dataset 6

1.5.4 Dataset date range 6

1.5.5 No reading value sent with alert notification 7

1.6 Document Outline 8

2 Literature review 9

2.1 Introduction 9

2.2 Shared characteristics of IoT networks 9
2.3 Why IoT sensors? 10
2.4 Fault prediction and prevention 10
2.5 Alerts 11
2.5.1 Measurement thresholding 11
2.5.2 Rule based alerts 12
2.5.3 Thresholding of predicted measurements and classifiers 12
2.5.4 Anomaly prediction 13
2.5.5 Aggregated monitoring (hot spots) 13
2.5.6 Alerts summary 14
2.6 Large sensor data volumes 15
2.7 Modelling pipelines 15
2.8 District Heating challenges 16
2.9 Summary 17
3 Design and methodology 19
3.1 Data 19
3.2 Software used 19
3.3 Stratified partitioning 19
3.4 Model selection 20
3.5 Handling the imbalanced target variable 20
3.5.1 Under sampling 20
3.5.2 Boosting 21
3.5.3 Synthetic Minority Over-sampling Technique (SMOTE) 21
3.5.4 *Misclassification costs* 21
3.6 Evaluation selection (minority class) 21
3.6.1 Accuracy 21
3.6.2 Confusion matrix 22

3.6.3 Precision 22
3.6.4 Recall 22
3.6.5 ROC AUC – (Area Under Curve) 22
3.6.6 Precision-Recall (PR) Curves 23
3.7 Modelling 23
3.8 Tuning 23
3.9 Comparing model performance 23
3.10 Hypothesis testing 24
4 Implementation and results 25
4.1 Business understanding 25
4.1.1 Why is alert data ignored? 25
4.1.2 Work order of type “Fault” and “Accident” 25
4.1.3 How work orders are created? 25
4.2 Data investigation 26
4.2.1 Sparse IoT reading sensor coverage 26
1.1.1 Data variable description 27
1.1.2 Alerts 27
4.2.2 Work orders 32
4.2.3 Invalid data 33
4.3 Data preparation 34
4.3.1 Ignored variables 34
4.3.2 Missing data 34
4.3.3 Duration of an alert 35
4.3.4 Time dimension variables 35
4.3.5 Target variable association 35
4.3.6 Raw dataset 36
4.3.7 Special feature creation (Spatial and temporal aggregation) 36

4.3.8 Dummy variables for categorical input 37
4.3.9 Outliers and extremes 38
4.3.10 Other pre-processing tasks 39
4.4 Modelling 39
4.4.1 Experiment #1 - Raw dataset 39
4.4.2 Experiment 2 # - Special features dataset 41
4.5 Results 41
4.5.1 Experiment #1 - Raw dataset 41
4.5.2 Experiment 2 # - Special features dataset 44
4.6 Comparing Raw versus Special Features datasets 47
4.7 Hypothesis testing 48
5 Analysis, evaluation and discussion 49
5.1.1 Handling the imbalanced target variable 49
5.1.2 Experiment #1 - Raw dataset 49
5.1.3 Experiment 2 # - Special features dataset 50
6 Conclusion 51
6.1 Future work & recommendations 51
6.2 Conflict of Interests 52
Bibliography 53
Appendix 58
8.1 Raw dataset results 58
8.1.1 Evaluation metrics 58
8.1.2 Curves 58
8.1.3 Confusion matrices 59
8.1.4 Predictive importance 60
8.2 Special features dataset results 62
8.2.1 Evaluation metrics 62

8.2.2 Curves 62
8.2.3 Confusion matrices 63
8.2.4 Predictive importance 64
8.3 Comparing Raw dataset with Special features 66
8.3.1 ROC Curves 66
8.4 Outlier data 68
8.5 Zombie classifier with minority class 70
8.6 Top level SQL queries 72
8.6.1 GET_EVENT_WITH_TARGET 72
8.6.2 GET_SPECIAL_FEATURES_DATASET_WITH_TARGET 74
8.6.3 GET_SPECIAL_FEATURES_DATASET_WITH_TARGET 76
8.7 Lower level functions 78
8.7.1 GET_WORK_ORDER_ALERTS 78
8.7.2 GET_WORK_ORDER_ALERTS 79
8.7.3 GET_NUM_RELATED_ALERTS 80
8.7.4 GET_NUM_RELATED_ALERTS (single alert) 82
8.7.5 GET_DISTANCE_BETWEEN_ALL_ALERT_LOCATIONS 83
8.7.6 GET_ALERT_DURATION 84
8.8 Tables to cache recurrent calculations 85
8.8.1 EVENT_LOCATION_DISTANCES 85
8.8.2 EVENT_LOCATION 85
8.9 Modifications to existing system DB tables 86
8.9.1 SRC_EVENTS_MT 86
8.9.2 WORK_ORDER 86

Table of Figures

Figure 1 - Proposed analytical pipeline that filters sensor readings for interesting values from which to generate alerts. Source (author) 3
Figure 2 - Alert sensors form the first part of an analytics pipeline. Source (author) 3
Figure 3 - Data for weeks of the year showing 3 winter months covered. Source (author) 7
Figure 4 - Alerts clustering around a primary network fault. Source (author) 17
Figure 5 - Alerts clustering around a secondary network fault. Source (author) 17
Figure 6 - A cluster of alerts that do not relate to a fault. Source (author) 18
Figure 7 - Stratified partitioning of dataset. Source (author) 19
Figure 8 - Misclassification cost of 3 placed on when the target class = 1 is predicted incorrectly. Source (author) 21
Figure 9 - Histogram of alert creation date with outages visible. Source (author) 28
Figure 10 - Frequency distribution of locations. Source (author) 28
Figure 11 - The top 10 alert sensors by number of alerts issued. Source (author) 28
Figure 12 - Event type distribution. Source (author) 29
Figure 13 - Frequency distribution of the SUBJECT field. Source (author) 29
Figure 14 - Number of alerts per day of the week. Source (author) 30
Figure 15 - The distribution of alerts by each hour of the day. Source (author) 30
Figure 16 - Distances between all alerts. Source (author) 31
Figure 17 - Imbalanced target class with a ratio of 96:4. Source (author) 31
Figure 18 - Number of Work Orders by hour of the day. Source (author) 33
Figure 19 - Linking workorders to alerts that preceded its creation. Source (author) 35
Figure 20 - Visualisation of related alerts grouped by spatial distance. Source (author) 36
Figure 21 - Histogram of NUM_RELATED_ALERTS_10_500. Source (author) 38
Figure 22 - A boxplot and whiskers shows the outliers more clearly. Source (author) 38
Figure 23 - Modelling with multiple techniques. Source (author) 39
Figure 24 - Raw dataset ROC curve. Source (author) 43
Figure 25 - Precision-Recall curves for raw dataset. Source (author) 43
Figure 26 - Predictive importance of the input variables for the Raw C5.0 cost x3 model. Source (author) 44
Figure 27 - ROC curves for special features dataset with C5.0 SMOTE and C5.0 Cost x3 showing the best AUC values. Source (author) 45

Figure 28 - Precision-Recall curve shows most performant model is C5.0 with misclassification costs. Source (author) 46
Figure 29 - Predictive power of input variables for the C5.0 Costx3 model. Source (author) 47
Figure 30 - Comparative ROC curve between raw and special feature datasets for the C5.0 Cost x 3 model. Source (author) 47
Figure 31 - Precision-Recall curves for raw dataset. Source (author) 59
Figure 32 - Predictive importance of input variables for Raw 5.0 Cost x3. Source (author) .. 60
Figure 33 - Predictive importance for Random Forest SMOTE. Source (author) 60
Figure 34 - predictive importance for Random Tree (handling imbalanced) . Source (author) 60
Figure 35 - Predictor importance for Random Tree (handling imbalanced, cost x2) SMOTE. Source (author) 61
Figure 36 - Predictor importance for XGBoost tree SMOTE. Source (author) 61
Figure 37 - ROC curves for special feature dataset. Source (author) 62
Figure 38 - Predictor importance Random Forest SMOTE. Source (author) 64
Figure 39 - Predictor importance for Random Trees (handling imbalance, cost x2) . Source (author) 64
Figure 40 - Predictive importance for Random Trees (handle imbalanced) . Source (author) 64
Figure 41 - Predictor importance for XGBoost Tree SMOTE. Source (author) 65
Figure 42 - Predictive importance for Logistic Regression SMOTE. Source (author) 65
Figure 43 - Predictive importance for Random Trees. Source (author) 66
Figure 44 - Raw and Special features ROC curves for C5.0 with a cost x3. Source (author) . 66
Figure 45 - Raw and Special features ROC curves for XGBoost with SMOTE applied. Source (author) 67
Figure 46 - Raw and Special features ROC curves for Random Tree (handling imbalance) . Source (author) 67
Figure 47 - Raw and Special features ROC curves for Random Forest SMOTE. Source (author) 68

Table of Tables

Table 1 – Partition proportions 19
Table 2 - Alert variable investigation and description 27
Table 3 - Alerts - Interesting statistics 27
Table 4 - Work orders - interesting statistics 32

Table 5 - No missing data for raw model input variables....34
Table 6 - Correlation matrix of all special features....37
Table 7 - Configuration of models for the raw dataset....40
Table 8 - Configuration of models for the special features dataset....41
Table 9 - Compare evaluation metrics of top models on the raw dataset....42
Table 10 - Confusion matrix for the raw dataset....42
Table 11 - ROC Curve AUC values for the raw dataset....43
Table 12 - Compare evaluation metrics of top models on the special features dataset....45
Table 13 - Confusion matrix for the top model....45
Table 14 - ROC Curve AUC values, special features dataset....46
Table 15 - Matrix of baseline model predictions versus C5.0 COSTx3 model predictions in the raw dataset....48
Table 16 - McNemar test of the baseline model versus the C5.0 COSTx3 model....48
Table 17 - Matrix of the raw dataset C5.0 COSTx3 model predictions versus C5.0 COSTx3 model predictions....50
Table 18 - Results of the McNemar test performed on the 2x2 matrix of raw dataset C5.0 COSTx3 and special features C5.0 COSTx3 model predictions....50
Table 19 - Raw dataset evaluation metrics for each model....58
Table 20 - raw dataset ROC curves of all models. Source (author)....58
Table 21 - Confusion matrices for all raw dataset models. Source (author)....59
Table 22 – Special feature dataset evaluation metrics for each model....62
Table 23 - Special features dataset confusion matrix....63
Table 24 - Extreme values table....70

Glossary

District Heating	The industry name for heating networks in cities
DH	District Heating
IoT	Internet of Things
IBM IOC	Intelligent Operations Center An IBM smarter cities product
NPR	Number plate recognition

1 Introduction

1.1 Background

A paper by (Mattern & Floerkemeier, 2010) describes the Internet of Things (IoT) as how

"the Internet extends into the real world embracing everyday objects. Physical items are no longer disconnected from the virtual world, but can be controlled remotely and can act as physical access points to Internet services"

IoT sensors are embedding more and more in our city's public infrastructure. Many energy, transport, waste, water and heating networks are getting IoT upgrades. Some of these sensors simply send reading data from the sensors at regular intervals. Other "smart" sensors have some situational intelligence built in. For example, a smart sensor might send an alert if the pressure in a pipe exceeds a certain value. Early predictions for the success of smarter cities projected huge adoption by cities however it has been noted that "*business around smart cities is having difficulty taking off*" (Vilajosana et al., 2013). There are still many challenges for the vendors of Smarter City products, but the potential benefits are still clear.

IoT sensor data can be used by organisations that run city networks to simply monitor activity in real-time. The next level is to use the data to try better predict and respond to issues that happen such as pipe leaks and bursts. The data is also used to analyse trends and find insights that may lead to efficiencies and cost reductions on the network.

When a significant issue occurs on a network, a network analyst may need to schedule a repair team to resolve that issue. This event is called a **Work Order**.

This paper will focus on fault detection in a smarter city network. A fault in this case is defined as an issue that requires the creation of a Work Order.

1.1.1 How are Work Orders generated?

An operator monitoring a network takes various sources of information into consideration when creating a Work Order. They receive complaints from call centres, feedback from maintenance crews and readings from IoT sensors. Using sensor readings allows for issues to

be resolved proactively rather than responding to complaints and feedback. As previously mentioned, sensor data comes in the form of continuous readings or as events and alerts.

An accurate alerting system should detect or even predict faults. Early detection and prevention of faults leads to reduced time spent monitoring the network and in the deployment of maintenance teams. These are the primary cost factors for a network management company.

Conversely when the alerts being generated include a high proportion of false positives, costs are actually increased. Operator time is wasted, and unnecessary work orders are generated. A loss of confidence in the accuracy of alerts inevitably leads to them being ignored altogether.

1.2 Research Problem

The idea for this study came while working on an IBM IOC District Heating project for a European city. Alerts in the system are generated when pipe pressures exceed high or low threshold values. The customer uses a geospatial dashboard to monitor the network. Surprisingly alert data was not configured to be displayed. When asked why this was the case, the customer said that it had previously being enabled but operators found that most alerts did not correspond to real faults. They made the decision to remove the alerts from the dashboard as they were just "*introducing noise*". The reason why this type of alert is not useful in identifying real faults is that it is overly sensitive to localized spikes in pressure.

While raw alert data was not useful for operators, this study aims to investigate if they can be used as inputs to an analytical model that predicts network faults. If this approach could be validated, it would provide an alternative to existing models that use continuous raw sensor readings as input. By using threshold-based alerts instead of continuous readings, the amount of data that the analytic models need to process would be greatly reduced. This pipeline-based approach also allows this same threshold data to be used as input to multiple other models.

The pipeline diagram in Figure 1 shows how reading data can be filtered using thresholding to extract interesting readings. These readings can then be passed forward in the pipeline to be consumed by one or more downstream models.

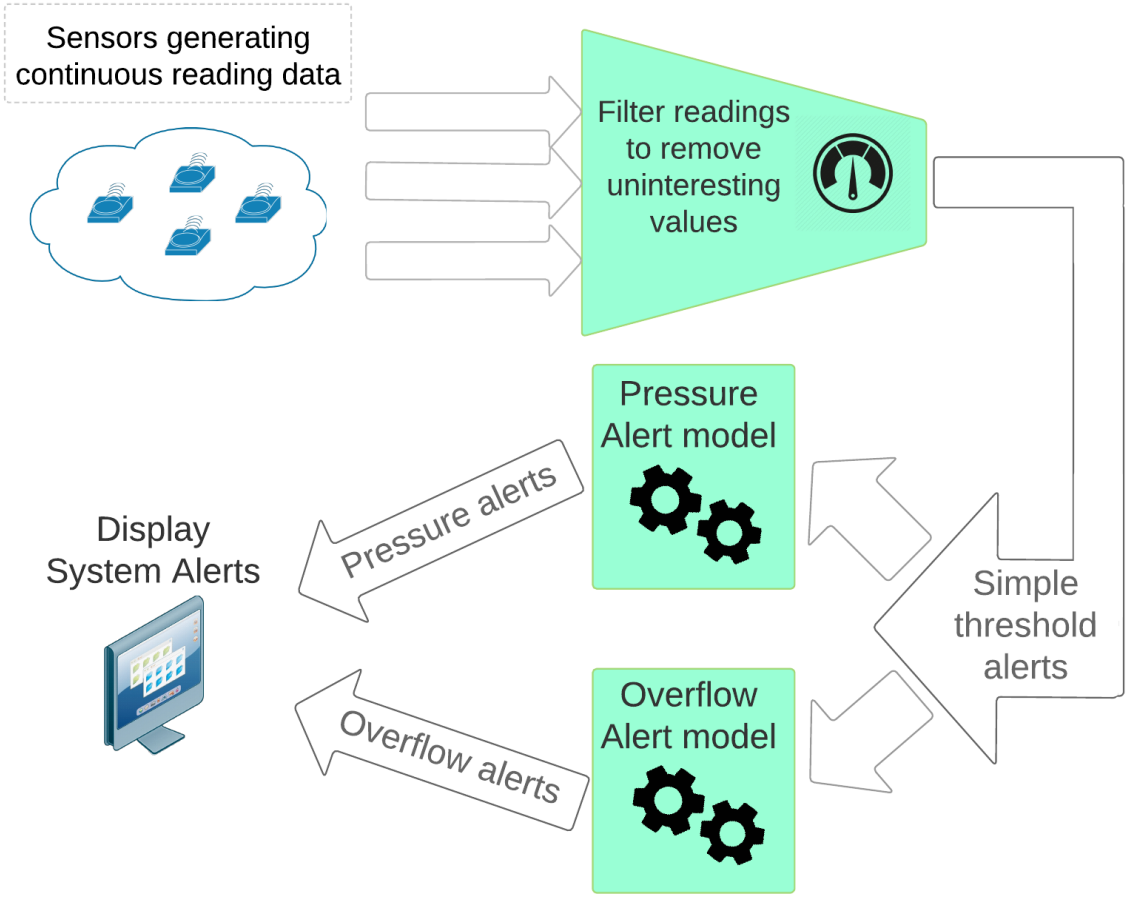

Figure 1 - Proposed analytical pipeline that filters sensor readings for interesting values from which to generate alerts. Source (author)

Interestingly, in this study the data is already filtered to yield interesting readings. These are in the form of "smart sensor" alerts. Figure 2 shows a version of the pipeline diagram from Figure 1. In this version the initial stage of the pipeline has been replaced by smart sensors

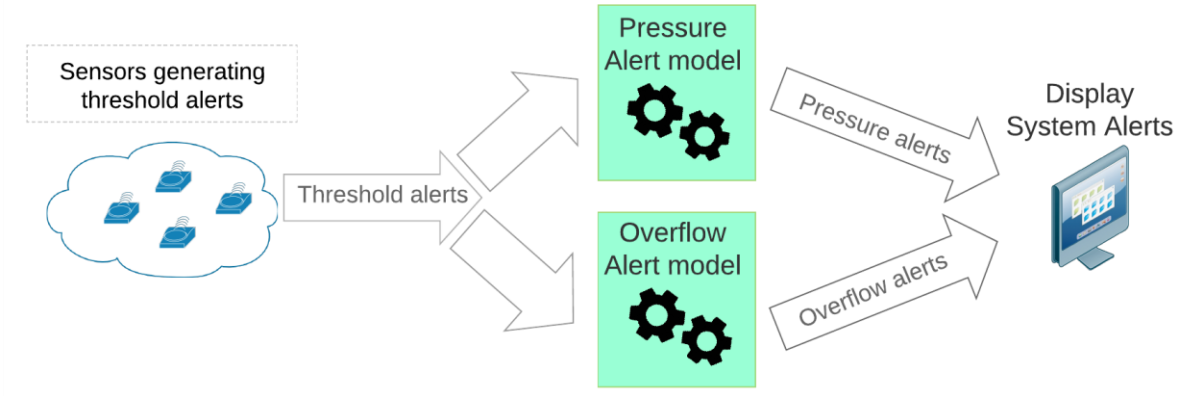

Figure 2 - Alert sensors form the first part of an analytics pipeline. Source (author)

that only send readings when thresholds are exceeded. The possibility of creating accurate fault prediction from threshold-based alerts is the question addressed in this study.

1.2.1 Research question

Can the need for Work Orders in a District Heating network be determined from analysing simple threshold-based alerts?

1.3 Research Objectives

1. Export and clean the required alert and work-order data from an active database of an IBM IOC system.
2. Do business and data understanding work
3. Data preparation: Associate work-orders by type with each alert instance
4. Create all smart features.
5. Partition the dataset into training (70%) and validation (30%)
6. Use different bimodal classification model types to investigate $\mathbf{H_0}$
7. Compare how all the models perform against each other.
8. Accept or reject $\mathbf{H_0}$

1.3.1 Hypotheses

H_0: There is no association between District Heating network simple threshold-based alerts and a need for Work Orders in the network.

1.4 Research Methodologies

The type of research is secondary research as the data used was already collected in the IBM IOC database.

The research objective is Quantitative research as a systematic empirical investigation of the relationships between alert data and work-order data will be performed by confirming hypothesis in a close-ended, stable study.

The research form is empirical research as the feasibility of predicting the need for a work-order will be tested using empirical evidence and experiments.

Deductive reasoning is used to generate a research question, develop hypothesis, gain observations from experiments and then confirm or reject the hypotheses.

The dataset used in the study was acquired from an IBM IOC customer. The customer manages a European cities District Heating network. The data was provided for use in building analytic models.

The study will follow a consistent Data analytics Crisp DM approach (Business understanding, data understanding, data preparation, modelling, evaluation, deployment). This is an iterative feedback process. The dataset will be analysed for its quality.

The first step will be **business understanding**. This step will aim to understand the problems and needs of the domain. This will be achieved by consulting with domain experts and by doing a comprehensive literature review. Insights from this step will inform the **data understanding** stage.

Descriptive analysis tools will be done using R and SPSS to understand the data. Data values ranges, measures of centrality, missing data, outliers will all be examined. Lessons from this stage will lead to more business understanding questions to be asked. Scatter plots and correlation tests will investigate which variables might be useful or superfluous for creating models.

This process will iterate for as long as necessary will be feedback between these first 2 stages before moving on to the next step – **data preparation**.

A very large proportion of the work will be in the data preparation step. Data is extracted into a flat file from a copy of the production database. Temporal and spatial aggregation of data will be done to create "special features". Any required data cleaning will be done. Certain input variable data types will have to be transformed so that in can be used in certain classification models.

The **modelling** will then be done by following a set of clearly described experiments. The dataset will be partitioned into 3 splits using stratification:

1. Training – 70%
2. Validation – 10%
3. Holdout Testing – 20%

These experiments will have **evaluation** criteria defined for them and will be created in such a way to allow the studies hypothesis to be accepted or rejected based on statistically significant results.

In a typical Crisp DM cycle the developed model would then be **deployed** to a production environment to be tested in the field. This was not possible in the time frame of this study.

1.5 Scope and Limitations

1.5.1 Single network, single city

The dataset for this study is for a single European city. It would be preferable to have data for multiple cities to test if results were consistent across networks of varying size and complexity. By having multiple cities any findings would hold greater significance.

1.5.2 Domain limited to District Heating

The District Heating domain is a very interesting one, but it is also much less prevalent than other networks such as water, wastewater and energy. For this reason, the impact of studies on the area are limited to this small domain. The domain also has certain characteristics (described in Section 2.8) that make it more challenging to make fault predictions. However, a new Work Order feature was only being used by a single customer that only managed a District Heating network.

1.5.3 No explicit target variable in the original dataset

When supervised learning methods are used, a target variable is required. In the study's dataset we have alerts and workorder records but there is no explicit link between them. This is a problem as we cannot proceed with a supervised learning approach until this association has been made. To devise a way to make this association between alerts and work orders, domain experts and system analysts were consulted to determine a sensible and informed way to estimate the relationship. The method devised is outlined fully in Section 4.3.3. This limitation means that there will always be a certain proportion of Work Orders that are not related to the alerts at all.

1.5.4 Dataset date range

A multi-year dataset would have been preferred to try and capture seasonal and cultural patterns. However, this was not possible as the dataset was of a city where recording of both work orders and alerts began in November of 2017. Three months of data was subsequently

exported in January of 2018 for use in the creation of models and for this study.

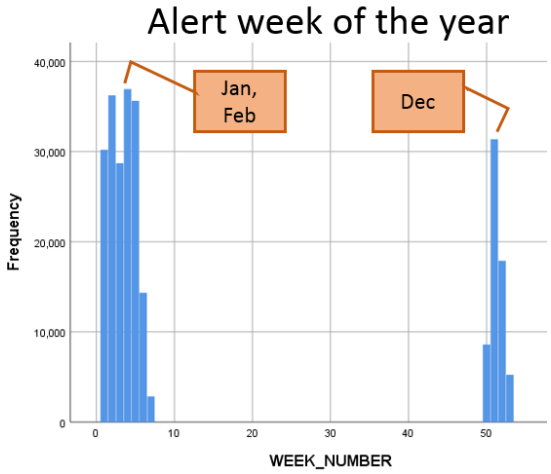

Figure 3 - Data for weeks of the year showing 3 winter months covered. Source (author)

1.5.5 No reading value sent with alert notification

The alerts in this study are generated when a threshold is exceeded. An alert is either created or not. Alert creation does not depend on by how much the threshold has been breached. This is when the actual value of the reading that caused the alert might be useful input to an analytic model. Also, the value of the actual threshold level for that sensor in that location may be of use. Unfortunately, neither of these 2 this information points are contained in the alert notification message for the dataset. Attempts were made to link alert notifications and to get the reading of an asset at the same point in time but due to the co-location of certain assets the results of this attempt were inconsistent and had to be abandoned.

1.6 Document Outline

This paper is organised as follows:

The Chapter 2 - Literature review reviews papers to discover the current state of the art in alert detection. It will review methodologies and issues faced with generating effective alerts. It will also review how other domains handle the problem. It will then present how gaps in the literature lead to the research question.

In Chapter 3 - Design and methodology the process of how the research is conducted is outlined in detail. The reasons for design decisions are given. A description of the how experiments are conducted is outlined. The methods for evaluation are also clearly described.

Chapter 4 - Implementation and results follows the Crisp DM procedure for data analysis. This involves going through the stages of Business Understanding, Data Investigation, Data Preparation, Modelling (running experiments) and listing the results of these stages. As there will be 2 datasets used in the experiments. The results of a comparison between these will also be recorded here.

The results of the previous chapter will then be analysed in Chapter 5 - Analysis, evaluation and discussion. In this chapter the results of the experiments are analysed. The methods of evaluation described in Chapter 3 will used to determine how well the models developed have performed.

Finally, we discuss what all these results mean in Chapter 6 – Conclusion. This section of the paper puts the results in context and discusses the contribution and impact of the findings. It also points to future work that might be undertaken.

2 Literature review

Keywords: *fault detection, IoT sensors, water networks, maintenance, prediction, alert prioritisation, smart meters, temporal aggregation, spatial aggregation, district heating,*

2.1 Introduction

A subdivision of the IoT world is the use of sensors to create "Smarter cities". This encompasses various domains such as water, waste, heating, traffic and energy.

This review will look at how IoT sensors are being used in the "Smarter Cities" domain and will try to identify opportunities to add value to the body of analysis methods. Research will focus on finding ways to assist network operators. The operator's job is to monitor the health of the network and coordinate required actions. The goal is to improve the **accuracy and timeliness of actions** taken by the operator. Better operator decisions lead to a reduction in system costs. Examples could be earlier fault detection, limiting the impact of a fault or even fault prevention.

2.2 Shared characteristics of IoT networks

There are many papers which aim to investigate the current and future challenges of IoT networks (Fletcher, Andrieu, & Hamel, 2013; Niemczynowicz, 1999). These are useful to put in context all the challenges that city management agencies are facing.

Various city IoT network types (energy, water, heating, waste) share certain characteristics. All networks types are monitored via sensors that record time-series measurement data. They share physical characteristics such as:

- connectors (pipes, power lines),
- connection points (valves)
- terminals (sub-stations, tanks, treatment plants, etc)

They also share operational characteristics. Each type of network must deal with challenges such as:

- Load balancing
- Fault monitoring
- Minimizing energy consumption

- Maintenance and repair management

Due to these shared characteristics, this study looks across all the smarter city domains for insights on common problems and solutions. There may be opportunities to gain insights from one domain that can be applied to another.

2.3 Why IoT sensors?

Many papers work backwards from the IoT sensors to find applications for them. Cost reduction is the most common application found. In (Arampatzis, Lygeros, & Manesis, 2005), they examine the possible applications of IoT sensors and make strong cases for how they can help in Water Management. They cite how a monitored system can be proactive rather than reactive. This is primarily how costs are reduced. These ideas are reinforced in (Kanakoudis & Tolikas, 2001; Le Gat & Eisenbeis, 2000).

Another common application of IoT sensors is in the domain of environmental protection and resource shortage (Fang et al., 2014). These issues will at some point become costs for the management agencies or citizens or government. Costs such as fines, loss of profit, losses from in-efficient processes, charges for use of a resource. Resource shortage and uncertain supply will also eventually become costs in the form of fines, loss of profit, losses from in-efficient processes and higher charges for use of a resource. The sole purpose of a paper by (Davis, Sullivan, Marlow, & Marney, 2013) was to find out which if the available market solutions for monitoring water networks actually yielded a cost benefit. They looked for where solutions were *"likely to reflect a rational economic decision"*.

This suggests that in order contribute to this field, further ways to reduce costs should be prioritised. Using sensors to predict and identify faults in a network can result in significant cost savings.

2.4 Fault prediction and prevention

An interesting aspect that this research presented was the topic of maintenance cost for Water Management Agencies. Predicting what leak alerts require a Work Order is closely related to leak and burst detection, but it is not the same. Sometimes you may have a leak alert, but it does not warrant a callout. To efficiently deploy maintenance crews, it is crucial to identify the significant alerts that lead to problems requiring a Work Order.

To get a general insight on the problem of maintenance management, the topic was reviewed in other industries. It was found that different industries were struggling to move away from *"equipment failure-driven and time-based maintenance"* to *"condition-based preventative maintenance"* (Tse, 2002)

This section shows where much of the cost savings could be gained. Prevention of faults is a big cost saver. Earlier detection is a key goal. The identification of what might delay a prediction is something that should be further investigated.

2.5 Alerts

Given the increasing number of sensors being deployed in city networks it is not feasible for an operator to continuously monitor sensor readings. Alerts are used to draw the operator's attention to an issue in the network that may require her intervention.

On reviewing the documentation, there are several approaches to generating alerts:

- Measurement thresholding
- Thresholding of predicted measurements
- Rule based alerts
- Anomaly detection
- Measurement aggregation

The types of data used is important to note. Many solutions use the network infrastructure data along with sensor reading data such as pressure to create their models (Martínez-Codina, Castillo, Gonzalez-Zeas, & Garrote, 2015) whereas others use the infrastructure data which have *"pipe-specific factors, e.g., diameter and length"* paired with maintenance records. This is more usual when the models are simulated (Le Gat & Eisenbeis, 2000).

2.5.1 Measurement thresholding

Creating an alert by measurement threshold is done simply by monitoring each reading value to see if it has exceeded predefined threshold levels.

IBM's product IOC (Intelligent Operations Center) is the source of data for this study. A company journal describes how the product generates alerts (Bhowmick et al., 2012). The out-of-the-box offerings are:

1. Ingestion of externally generated alerts (3rd party services)

2. Generation of alert data based on the threshold set on reading values

The 3rd party in the first instance could be using any method to create alerts. From asking domain experts it is understood that the most common method is through thresholding of predicted measurements and the use of classifiers (Section 2.5.3). The second method is a very simple method of creating an alert which is not sensitive to any other factor other than the reading level.

2.5.2 Rule based alerts

Rule based alerts are often an extension of threshold-based alerts. IBM IOC has rule based alerts where an alert is only issued if the reading exceeds the threshold for a period of time.

2.5.3 Thresholding of predicted measurements and classifiers

Well before the concept of Smarter Cities and IoT was mainstream, measurement data was analysed to try and predict future values so that related decisions could be made ahead of time.

A 1996 study which investigated how the salinity of river water in South Australia could be predicted using Neural Network models. Historically recorded sensor data was used to train a prediction model (Maier & Dandy, 1996). The goal was to predict salinity levels to optimise when water should be extracted from the river. It showed how the water company could make a large optimisation and save costs on damage that high water salinity caused.

This same use case was further worked on by (Kingston, Lambert, & Maier, 2005). They had found that the use of Neural Networks had *"not been adopted by water resources practitioners because of the difficulty in implementing them"*. They proposed some optimisations whereby they changed the training approach to make the implementation of the neural network more straight-forward. Again, their model predicted salinity values.

The various solutions for these problems are broadly to improve management processes and to prevent certain issues arising. The improved management processes center around being more context aware (Perera, Zaslavsky, Christen, & Georgakopoulos, 2014) and using network modelling (Gaddam, 2014) to manage a complex system in a joined up way. In these models there will need to be a direct association of the equipment that failed and the sensor readings. That way the model can have a have a target variable. Typically, this will be a traffic light

status values of Acceptable, Caution and Critical. The target variable for these models would be when the asset has a status of Critical.

The literature strongly promotes prevention of network issues by using prediction models. There is a lot of focus on predicting when leaks and bursts will happen (Martínez-Codina et al., 2015; Mounce S. R., Boxall J. B., & Machell J., 2010). In the water domain there is a strong focus on Water Quality prediction to try and prevent serious events from occurring rather than just responding to them (Kingston et al., 2005; Maier & Dandy, 1996).

This section examined "Smarter City" solutions that are based on monitoring or modelling of sensor reading data. This relies on the accuracy of the individual sensors and making sure that they are calibrated correctly. In a network that contains many sensors that are correctly calibrated this has been seen to be effective. However, a different approach is needed in a network with a sparse distribution of sensors or one that contains less accurate sensors. The next section investigates how data from multiple sensors is aggregated to gain insights.

2.5.4 **Anomaly prediction**

A study of a network in Yorkshire England (Mounce, Mounce, Jackson, Austin, & Boxall, 2014) used a system called AURA (Advanced Uncertain Reasoning Architecture). This system does not detect faults but detects anomalous patterns. These anomalies can then be investigated by a network analyst. It works by using historical sensor reading data the model can "*learn and model the normal operating envelope for a system*". AURA consists of a binary neural network that is built on top of CMMs (Correlation Matrix Memories). This is like a library of system states that have been found during training. If the system comes across a state that it has not seen before it is flagged. At this point it is important to note that it has not detected a fault in the system but has merely detected strange behaviour.

2.5.5 **Aggregated monitoring (hot spots)**

Alerts that are generated by smart sensors are useful but can vary in quality. Sometimes the thresholds configured are no longer relevant and this can lead to many alerts that are disregarded as false positives. However, DH system analysts have given feedback of how clusters of alerts can sometimes be indicative of a real fault in the network. The alerts are both spatially and temporally close. These can be called alert "hot spots".

Hot spot analysis is used in many domains such as crime and service demand (food delivery, taxi apps). When looking at crime hotspots the exact location of the crime varies and hotspots

are identified by some area (neighbourhood/district/postcode/precinct) and over long stretches of time (Sorensen, 1997). The crime hot spots allow law enforcement to see patterns that they would not notice by reviewing individual crimes in isolation. Hotspots were also used in a paper that looked at taxi demand (Chang, Tai, & Hsu, 2010). In this paper hot spot was generated using clustering methods. They tested 3 algorithms (k-means, Agglomerative hierarchical clustering and DBSCAN) and found that "*different clustering methods have different performances on different kinds of data distributions*"

Advanced types of sensor aggregation use multiple sensors to determine the best value. This is a way of improving reading values by using cheaper sensors and aggregating their values. In (Ma, Guo, Tian, & Ghanem, 2011) relative error between sensors is used to determine a dominant value. This shows how aggregation can be a means to mitigate against inaccurate individual sensor data.

The District Heating system analysed in this study has deployed IoT smart sensors that publish an alert notification if there is significantly high or low pressure at a point in the network. Most of these alerts (96% in our dataset) do not indicate the need to take any action. They are false positives. However, on consulting with domain experts it was noted that an analyst may use alert data to identify a fault if some combination of other circumstances exists:

1. The alert has been active for a certain amount of time
2. There is a clustering of alerts in time and space
3. Reading values for certain assets are also anomalous.
4. The alert is active at a crucial location in the network.

These methods of detection effectively use the aggregation of alerts in time and space to identify problems. This work of aggregation could possibly be done using analytical models instead of tedious monitoring.

2.5.6 **Alerts summary**

This section on how alerts are created shows that there are more nuanced means to define errors in the system that by just using blunt thresholds. They can be designed to be more sophisticated and can reduce the need for analysts to consistently monitor individual sensor reading values. However, in practice, many alert systems are quite unsophisticated and are

just based on thresholding. This can lead to many “false positive” alerts, which can in turn lead to analysts dismissing alert notifications.

2.6 Large sensor data volumes

The large volumes of data emitted from IoT sensors is a useful resource but can also be overwhelming. The greater the amount of data that must be processed, the more resources that are needed to process and store this data. Depending on how big a smarter city network is “*the amount of data can be TB (terabytes), even PB (petabytes) and ZB (zettabyte)*” (Chen et al., 2015).

If all analytical modelling is being performed on the reading data that is being ingested, there can be duplication in work done between models. This is where the idea of having an analytical pipeline becomes a useful approach.

2.7 Modelling pipelines

With this large amount of streaming IoT data, it is very important to process it in the most efficient way possible. This is where analytical pipelines could help.

In a scenario described in (Zehnder & Riemer, 2017), IoT cameras were being used to open a gate for vehicle drivers. The pipeline consists of multiple streams of data. One stream has the camera video data. An early stage of the pipeline detects the vehicle registration from this video data using a Number Plate Recognition model. This information can then be fed forward in the pipeline to be combined with other data such as registered users to determine whether the gate should be opened. By splitting up the process, data from one stage can be reused by another. This approach also simplifies each individual component.

Another approach is to offset load on the central processing system by extending the analytical pipeline out into the IoT network. The modelling pipeline can begin in the IoT device itself by doing a certain amount of processing in the sensors. In another camera sensor example, the large transmission bandwidth required for video data poses a problem. A solution is to do the video *“processing at the edge device in order to conserve visual communication bandwidth”* (Chua et al., 2017). This strategy also takes the image processing load off the central system and distributes to the network.

From these examples it can be seen that using machine learning pipelines can be useful to simplify data modelling solutions. Models can be broken down into components that can give outputs that can then be consumed by several different processes or models.

This could be applied in an IoT smarter cities network scenario. Instead of generating all models from the same data, a pipeline could be devised to create intermediate outputs that could be fed forward in the pipeline as input. This would reduce the load on the central system allowing it to scale further by handling larger networks with the same processing power.

One such pipeline could be the thresholding of sensor readings to produce low level alerts which are then further processed by another model. By having the intermediate step, the load on the system is reduced. The more resource-expensive modelling is done on a smaller subset of the streaming data.

2.8 District Heating challenges

District heating (DH) networks have all the usual challenges of smart city IoT networks. These include load management, fault detection and energy consumption.

However, there are problems that are somewhat unique to DH networks as described in (Gadd & Werner, 2015) . They describe how DH networks contain secondary customer heating systems (residential buildings) which have faults that affect the primary supply system (substations and pipes). The primary network has the IoT sensors, but the "secondary network" does not. This means that the buildings cannot be monitored directly for faults. Proposed solutions were to fit the buildings with sensors, but this would require a large investment.

Another issue is that faults "*have no occurrence pattern; thus, they are difficult to predict*" (Gadd & Werner, 2015). There are many factors - including human and weather - which make it difficult to have good fault prediction on a DH network.

These papers introduce the idea that parts of a DH network may not have IoT sensors embedded (secondary network) and yet they may have faults. It raises the question of how we can monitor parts of the system if they do not have sensors applied. Section 2.5.5 discussed how the use of aggregation can mitigate against poor individual sensor data. This approach could also help add some coverage to the secondary networks of a DH network.

2.9 Summary

(Kanakoudis & Tolikas, 2001) notes that there is a *"proven strong relationship between the leaks and breaks"*. This can be generalised to say that there is a strong relationship between alerts and work orders.

This review demonstrated that most of the focus on creating alerts is based on the analysis of continuous measurement data from sensors and maintenance records. Alerts created using these models should have good fault prediction. The alerts are expected to indicate that there is indeed a problem in the network. However, it was also noted that other more basic means of creating alerts are still being used such as the thresholding of reading data and rule engines. Due to the basic design of these methods, they are less flexible than an analytic model and therefore much less accurate.

Figure 4 - Alerts clustering around a primary network fault. Source (author)

Two interesting topics that emerge in the review are those of Large Sensor Data Volumes and Modelling Pipelines. A connection can be made between the pair. In large networks with hundreds of thousands of sensors that emit measurement data every 1 – 30 minutes there is a very large number of reading values to score. The problem of handling large data volumes might be handled by breaking down the work into separate pipeline stages. An example of the proposed pipeline can be seen in Figure 1.

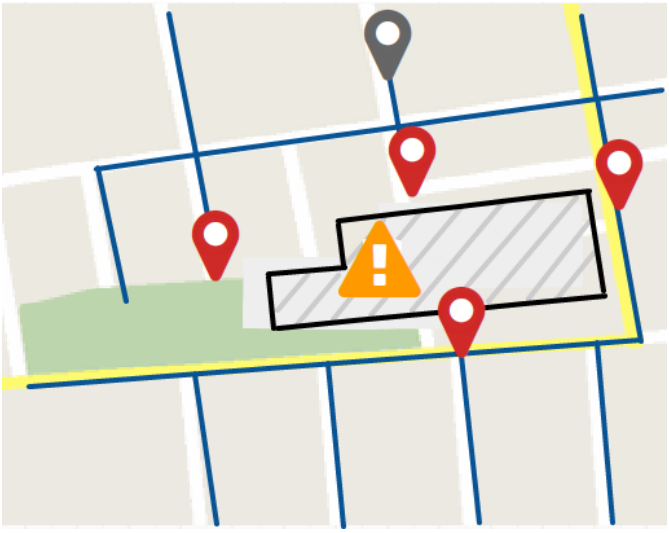

Figure 5 - Alerts clustering around a secondary network fault. Source (author)

As the dataset available to this study contains alerts generated from simple thresholds they generate many false positives. The majority of alerts when taken individually do not result in a Work Order being created. However, it would be interesting to test the viability of using thresholding as a method to filter reading data before it is further processed. The aim would be to make accurate predictions of system faults using these simple alerts as inputs to analytical models.

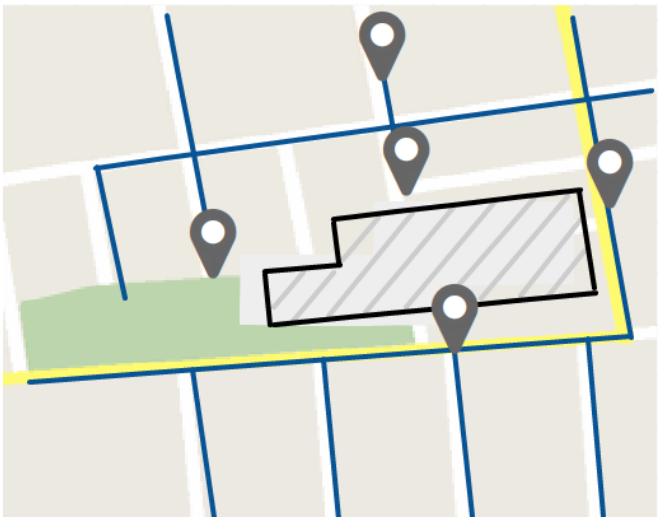

Figure 6 - A cluster of alerts that do not relate to a fault. Source (author)

A final issue to consider is that the dataset being analysed is from a District Heating network. The review highlighted the unique challenges of this type of smarter city network. Often the maintenance records will show that the faulty asset was not part of the IoT monitored network. To solve this problem, it is proposed to use spatial aggregation to compensate for this lack of network coverage. As the system is a network, problems are not always localised. Faults in one part of the network will often affect connected areas. This form of aggregation should aid in the better detection of faults in the monitored part of a DH network as in Figure 4.

More interestingly, if the number of alerts active in the same time and space is recorded, this should aid in the detection of alerts that happen on a secondary network.

The gap found the literature is that modelling is typically done using continuous measurements emitted from sensors. If the same accuracy can be achieved by analysing a filtered subset of that data, it might be useful in reducing system load caused by processing every sensor reading.

It is this link between alerts and Work Orders that I want to investigate and that leads to my research question.

3 Design and methodology

The focus of the experiments in this study is to see if alert data can be used to predict the need for Work Orders (faults and accidents). This will be done using classification models. The models will be compared using the evaluation methods described in Section 3.6. The most performant model will be tested for the significance of its predictions so that the study's null hypothesis can be rejected or fail to be rejected (Section 3.10).

3.1 Data

The data was taken from the database of an IBM IOC instance for a District Heating customer. The database is DB2. This will have to be investigated and exported into a flat file format so that it can be imported by the various software products that are used for modelling.

3.2 Software used

SPSS Statistics 25.0 will be used for data investigation. Initial exploratory tasks such as looking at frequency distributions and generating histograms will be done also.

SPSS modeller 18.0 will be used for the training and validating the models. It will also be used to output confusion matrices, ROC graphs and ROC AUC values.

R will be used for generating Precision-Recall graphs using prediction output from the modelling stage.

3.3 Stratified partitioning

Once data preparation steps are complete the dataset contains a total of 248,018 valid instances. For the purposes of creating models we must partition the data. The split proportions are outlined in Table 1.

	proportion	instances
Training	70.02%	173662
Validation	9.91%	24577
Testing (Hold out)	20.07%	49779
Total		**248018**

Table 1 – Partition proportions

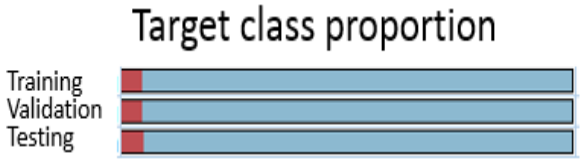

Figure 7 - Stratified partitioning of dataset. Source (author)

As there is a minority target class variable it is essential that the stratified partitioning is performed so that each split is in equal proportions as seen in Figure 7.

3.4 Model selection

The models used will be classification models. The models must also be able to handle both numeric and categorical data. Where a model can only handle numeric data, some pre-processing is required. The use of dummy variables is used.

The principle of Occam's Razor will be used; "*Given two models with the same generalization error, the simpler one should be preferred because simplicity is desirable in itself*" (Domingos, 1999). If a model has the same ability to generalise as a more complicated model it is possible that the more complicated model has some overfitting. A preference will be made for the simple model so long as accuracy and performance are comparable with a more complicated (or larger) model.

Models examined:

1. Decision Tree – C5.0
2. Random Tree
3. Random Forest
4. XG Boost
5. Linear Regression

3.5 Handling the imbalanced target variable

As previously mentioned, the target variable has a minority class. This means that we have to make certain adjustments to handle this fact.

1. Under sampling
2. Boosting
3. Synthetic Minority Over-sampling Technique
4. Misclassification costs

3.5.1 Under sampling

This is a method of handling imbalanced target variables by sampling out values for the majority class variable by a proportion until the classes are balanced. In (Liu, Wu, & Zhou,

2009) it was noted that while under-sampling is an effective method of handling imbalance, "*potentially useful information contained in ignored examples is neglected"*.

3.5.2 Boosting

Boosting is a method to combine weak learners to create a strong learner. If a base classifier such as a decision tree is giving weak results for a certain class, this weak result can be combined with other weak results to give a better combined or "boosted" result. In (Guo & Viktor, 2004) it is found that certain boosting algorithms can yield *"high predictions against both minority and majority classes"*.

3.5.3 Synthetic Minority Over-sampling Technique (SMOTE)

Synthetic minority over-sampling technique (SMOTE) is a method to balance the target class variables. The training partition of the dataset is passed to this process and it balances the target classes by "*creating synthetic minority class examples"* (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

3.5.4 *Misclassification costs*

Misclassification costs are provided by certain algorithms and allow certain results to be penalised. This tells the model to work hard to get that classification right.

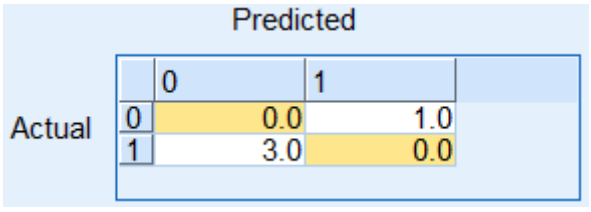

Figure 8 - Misclassification cost of 3 placed on when the target class = 1 is predicted incorrectly. Source (author)

3.6 Evaluation selection (minority class)

3.6.1 Accuracy

Accuracy the ratio of correctly predicted instances to the total number of instances in a dataset. It is a single measurement value based on the formula:

```
Accuracy = (TP+TN)/(P+N) = (TP+TNR)/(Total Data)

  TP = true positive
  TN = true negative
   P = number of positive values
```

```
  N = number of negative values
TPR = true positive rate
```

It is a measurement that is very clear when you have balanced class variables. However, when you have a target class imbalance it can be very misleading. For example, that dataset for this study has a target class with an occurrence ratio of 96:4. If a classifier simply predicted the majority class in all cases it would get an accuracy of 96%. Obviously, this is of no use and is potentially very misleading. For this reason, the metric of accuracy will be largely ignored during evaluation.

3.6.2 Confusion matrix

A confusion matrix is used to investigate the results of a classifier model. It is a useful for investigating datasets with imbalanced target class variables. It shows a 2x2 matrix of the count of instances for each class and the count predicted by the model. The best scenario is to have no False Positives and no False Negatives.

3.6.3 Precision

Precision measures the ratio of correctly predicted positive instances against the total predicted positive instances. In the case of this study, if a classifier scores an alert as *isPresentWhenWorkOrderIsRequired* then precision shows how well does that predict that a work order is required?

```
Precision = TP / (TP + FP)
```

3.6.4 Recall

Recall measures the ratio of correctly predicted positive instances to the all instances with a true value for the target class. For this study, it will give us the percentage of the class where

```
Recall = TP / (TP + FN)
```

3.6.5 ROC AUC – (Area Under Curve)

This study will use the Area Under the receiver operating characteristic (ROC) curve (AUC) as its performance measure, and to compare results from different experiments. The benefits of using this metric are discussed in (Bradley, 1997) and are particularly important when trying to take sensitivity and specificity into account.

However, AUC is a function of sensitivity and specificity, but is not sensitive to imbalance in target class proportions. A very comprehensive breakdown of this is provided in (Haibo He & Garcia, 2009). They note that in "*highly skewed data sets, it is observed that the ROC curve may provide an overly optimistic view of an algorithm's performance*". The paper experiments with an imbalanced dataset to show the preferable use of Precision-Recall curves.

3.6.6 Precision-Recall (PR) Curves

(Saito & Rehmsmeier, 2015) is a paper that solely investigates the performance of Precision-Recall curves versus ROC curves for imbalanced data. It proves that the PR curve "*is more informative than ROC, CROC, and CC plots when evaluating binary classifiers on imbalanced datasets*".

For the evaluation of models in the study, AUC, sensitivity and precision values will be examined, but we will pay close attention to the PR curve to ensure that the imbalanced dataset has not skewed the perceived results.

3.7 Modelling

The modelling process will be done by at first using default model configurations. The results will be compared using the metrics that are outlined in Section 3.6

3.8 Tuning

Different modelling algorithms will have different tuning parameters. Following the initial results from the selected models, additional tuned models will be added. The experiments will be re-run and the evaluation metrics used to alter these parameters in such a way as to improve the performance of the models.

There are many different approaches given for tuning SMOTE in (Zorić et al., 2016) which should be investigated for the tuning of this node. They involve using algorithms to find which tuning parameters might best suit a dataset.

3.9 Comparing model performance

The predictions of the classifiers will either be a 1 or a 0. This is binomial data and as such will not have distributed values. A non-parametric test is needed to examine binomial data as

it is not normally distributed. The McNemar test is suitable for this (Ciechalski, Pinkney, & Weaver, 2002). It will be used to compare the model results of the H_0 models where the dependant variable is *ALERT_WAS_PRESENT_BEFORE_WORK_ORDER.*

3.10 Hypothesis testing

Firstly, for hypothesis testing a significance level, α, is chosen that will be used for all tests. 5%, or $\alpha = 0.05$, is selected as it is the commonly accepted level for this area of study.

The performance measurements are set up so that all models use the same splits. This will give a paired test setup, which is more powerful than unpaired tests.

The relevant test's statistics will be computed; in this case the values of Recall, Precision and Accuracy will be used to see how well each model performs.

Next, the test's statistic (S) is compared to the relevant critical values (CV). The statistic used will be the McNemar test.

Hypothesis $\mathbf{H_0}$ can be rejected or fail to be rejected if any of the models return a statistically significant result that predicts the likelihood of a Work Order being required. The decision rule is to reject H_0 if $S > CV$ and vice versa. Practically, if $P \leq \alpha$ (0.05), we will reject the null hypothesis; otherwise we will fail to reject it.

4 Implementation and results

4.1 Business understanding

Data does not tell the full story of what is going on in a system. The Crisp DM process recommends a strong business understanding phase to make sure the meaning behind the data is understood before it can be properly analysed. The literature review (2) formed a large part of this, but consultation with domain experts was also very important. District heating networks have slightly different characteristics to other smarter city network domains, such as water and energy (see Section 2.8). A number of emails, calls and meetings were required to understand these differences.

4.1.1 Why is alert data ignored?

During these discussions it was discovered that the alerts on the IBM IOC system were not being used to detect a need for work orders. The reason for this was that they were producing too many false positives. Just how the alerts were produced using threshold rules was also discovered.

4.1.2 Work order of type “Fault” and “Accident”

It may seem strange, but when the question of what constitutes an “accident” was raised, the answer was that it was a type of fault. Operators like to use it to describe faults that needed some form of clean-up. As this study aims to detect faults, only Work Orders of type “Fault” and “Accident” will be used.

It had initially been planned to try and predict the “type” of Work Order, in addition to simply the fact that one was required. This was abandoned when it was learned that there was effectively only 1 Work Order Type that was to be detected.

4.1.3 How work orders are created?

It is important to understand the circumstances of how a Work Order is created. This understanding is needed for when Work Orders are being associated with Alerts (Section 4.3.5). The typical flow is:

1. On analysis of the network using the IBM IOC geospatial dashboard, analysts create a Work Order:

a. They set a TARGET_START and TARGET_END_TIME based on the priority of the issue and the resources available.
b. They set a Work Order type
    i. **Fault**
    ii. Prevention
    iii. Modernization
    iv. Retrofitting
    v. Operation
    vi. **Accident**
    vii. Service
    viii. Investment
    ix. Liquidation
c. The Analyst then assigns the Work Order to a piece of infrastructure (sub-station or chamber) that is deemed to be the source of the problem.

2. Next, the team responsible for monitoring that piece of infrastructure act:
    a. When they start work, the starting time is recorded in ACTUAL_START_TIME
    b. They assess and fix the problem.
    c. On completion the completion time is stored in ACTUAL_END_TIME.

4.2 Data investigation

4.2.1 Sparse IoT reading sensor coverage

On querying the database, it was found that there are 124,770 assets recorded on the system. This is normal for a mid-sized city. However, on further examination it was found that only 69 of these have IoT sensors capable of publishing readings. This is indeed low sensor coverage. This could be described as very sparse IoT coverage of the network.

1.1.1 Data variable description

Variable	Data type	Measure	Levels	Description
SUBJECT	string	categorical	241	Event type concatenated with asset specific data
EVENTTYPE	string	flag	2	High or low pressure event
EXTEVENTID	string	flag	2	High or low pressure event
STARTTS	datetime	continuous	-	Datetime the alert is created
ENDTS	datetime	continuous	-	Datetime the alert expires
LOCATION	string	categorical	69	The WKT location of the sensor
WEEK_NUMBER	numeric	ordinal	52	
MONTH_NUMBER	numeric	ordinal	12	
DAY_OF_THE_WEEK	numeric	ordinal	7	
HOUR_OF_THE_DAY	numeric	ordinal	24	
DURATION_IN_MINUTES	numeric	continuous	-	
ALERT_WAS_PRESENT_BEFORE_WORK_ORDER	numeric	flag	2	Target variable

Table 2 - Alert variable investigation and description

1.1.2 Alerts

Alerts – interesting stats	
Total number of alerts in the dataset	248021
Average alert count per day	4509
Average duration of an alert	1733 mins
Number of pressure sensor locations	69
Relevant types	Low pressure, High pressure
Alert subject count	243

Table 3 - Alerts - Interesting statistics

4.2.1.1 Time range of the alerts dataset

A multi-year dataset would have been preferred to try and capture seasonal and cultural patterns. However, this was not possible as the dataset was from a city where recording of both work orders and alerts began in November of 2017. As Figure 3 shows, 3 months of data was subsequently exported in January of 2018 for use in the creation of models. These are all winter months which could have a very different profile to summer month alerts. This is a known limitation and is highlighted in the of the study.

This data was relatively consistent in volume (Figure 9), except for 3 outages which caused gaps in the streaming data. These gaps could influence our models. Intuitively, these gaps could have an impact on the predictive power of the DAY_OF_THE_WEEK input variable.

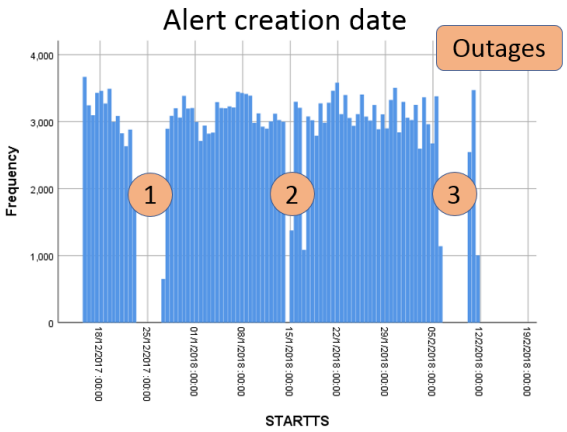

Figure 9 - Histogram of alert creation date with outages visible. Source (author)

4.2.1.2 *Locations of Alerts*

One of the most important numbers affecting this study relates to the LOCATION variable. The first thing that was previously noted in Section 4.2.1 is the small number of locations relative to the number of assets in the city. There are only 69 sensors that can issue alerts. With these low numbers each location may prove to be an indicator of whether an alert is more important or not. Figure 10 shows how alerts have an almost normal distribution the across all locations. There are a few locations that generate the majority of alerts, and even one that is responsible for almost 14% of alerts (Figure 11)

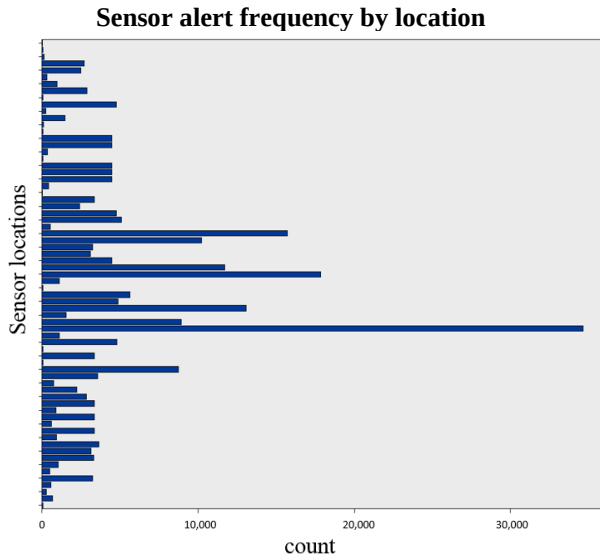

Figure 10 - Frequency distribution of locations. Source (author)

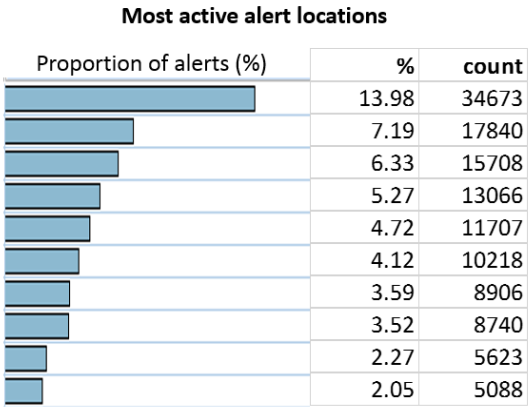

Figure 11 - The top 10 alert sensors by number of alerts issued. Source (author)

4.2.1.3 *Event type*

EVENTTYPE has 2 levels:

- High pressure difference

- Low pressure difference

Both states occur in almost equal proportions, 51.5% : 48.5% (Figure 12). EXTEVENTID is a duplicate of EVENTTYPE.

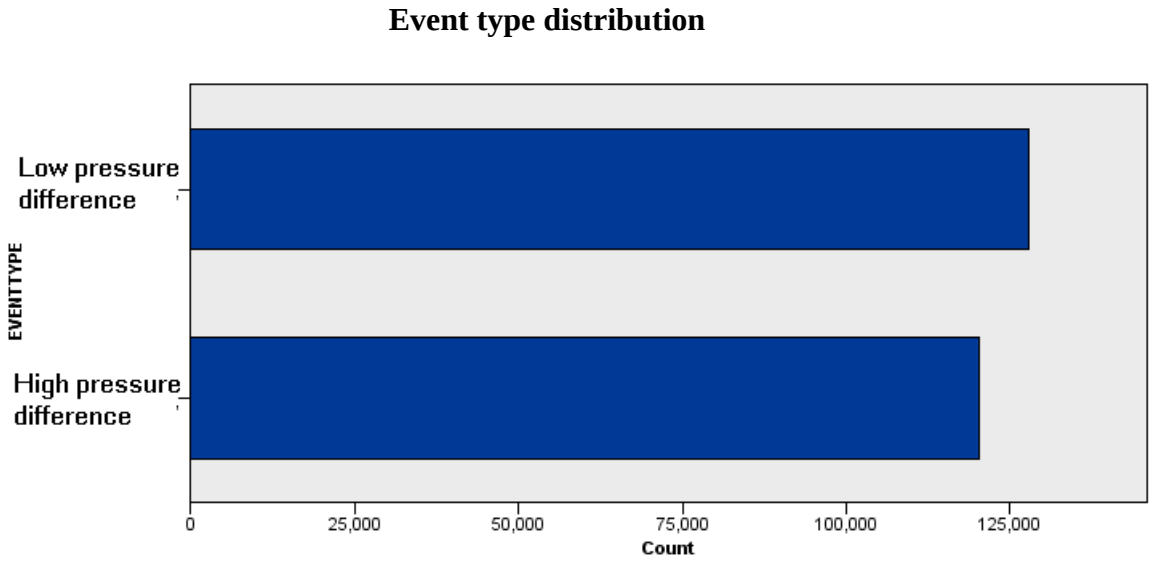

Figure 12 - Event type distribution. Source (author)

4.2.1.4 Subject

The dataset contains 241 levels for the SUBJECT input field. Each value is prefixed by the event type, and then typically some asset identifiers. This is borne out by a strong Pearson

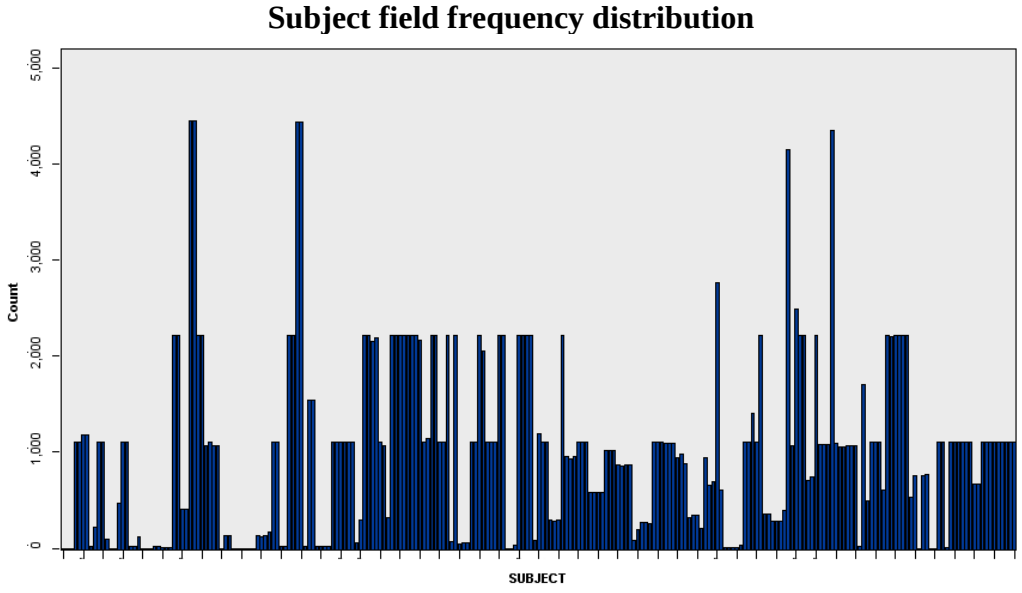

Figure 13 - Frequency distribution of the SUBJECT field. Source (author)

correlation value of 0.4 between LOCATION and SUBJECT. However, the values at a

location tend to change over time. When enquires were made about this, it was noted that the subject fields were changed multiple times during the dataset's time period. The reason for the changes was due to reporting changes. These changes may render this variable of little predictive importance.

4.2.1.5 Day of the week

The day of the week was investigated to see if there was a recognisable pattern or load at the level of a day. Figure 14 shows a noticeable dip in alerts occurring midweek, and that alerts peak at the weekend. This pattern might indicate that this variable will be useful in prediction.

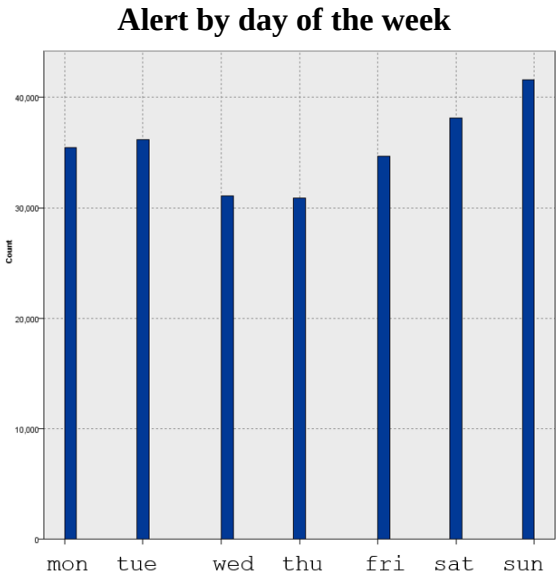

Figure 14 - Number of alerts per day of the week. Source (author)

4.2.1.6 Hour of the day

It would be expected that the state of the network would vary throughout the day. From the distribution graph Figure 15 we can see that, for the most part, the value is steady. There is a noticeable peak for the midday hours of between 10am and 2pm. There is also a very large drop in alerts for the hour of 11pm.

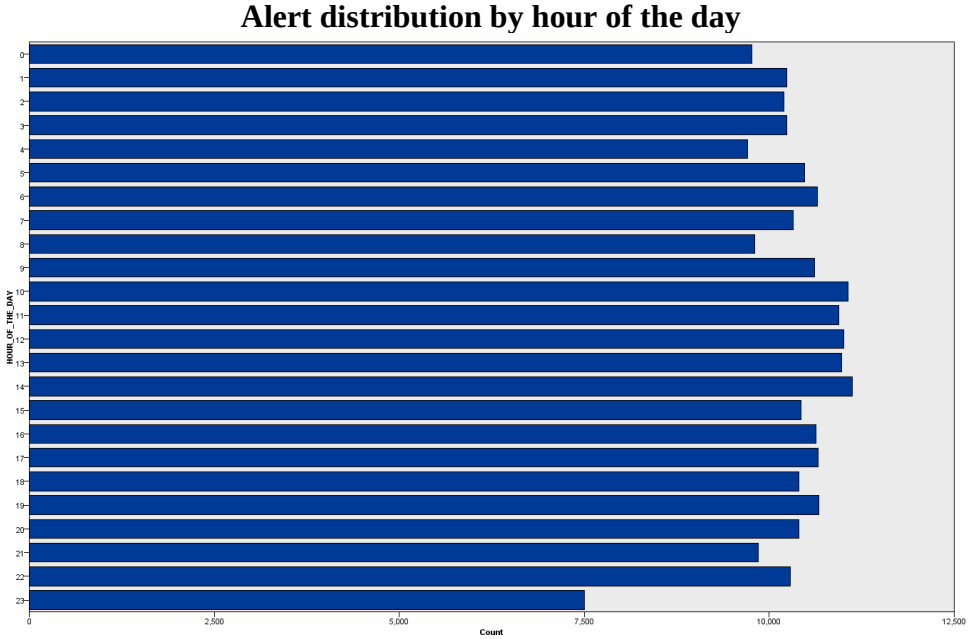

Figure 15 - The distribution of alerts by each hour of the day. Source (author)

4.2.1.7 Distances between alerts

Knowing what kinds of distances there were between values is important when selecting distances for aggregation in Section 4.3.6. The histogram in Figure 16 shows a mean distance of over 6.5km. This data was calculated using the database function GET_DISTANCE_BETWEEN_ALL_ALERT_LOCATIONS in Appendix 8.7.5.

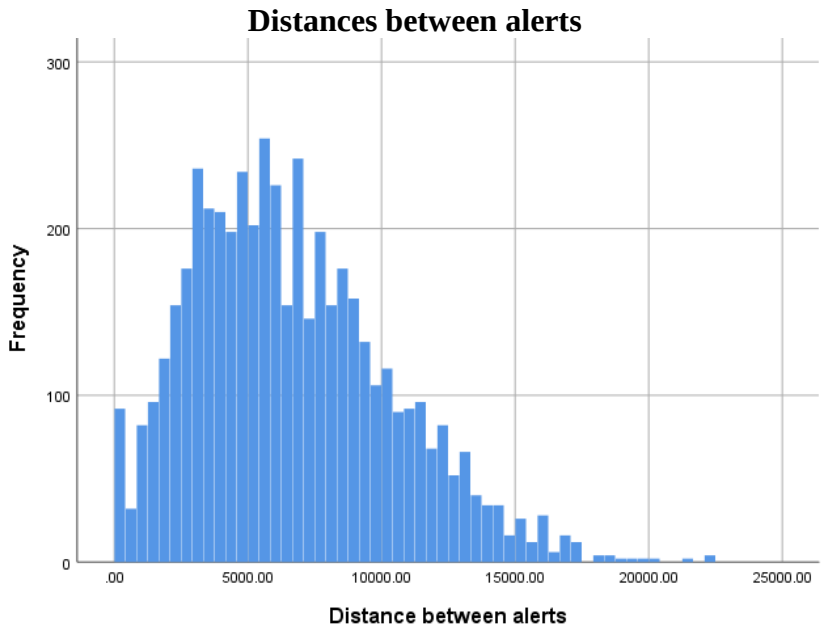

Figure 16 - Distances between all alerts. Source (author)

4.2.1.8 Target variable proportions

A key characteristic of this dataset is the distribution of values for the target variable. It is highly imbalanced at a ratio of 96:4. This can be seen in Figure 17. This has significant implications for the design of our experiments and how they are evaluated. This is discussed in Section 3.5 and Section 3.6.

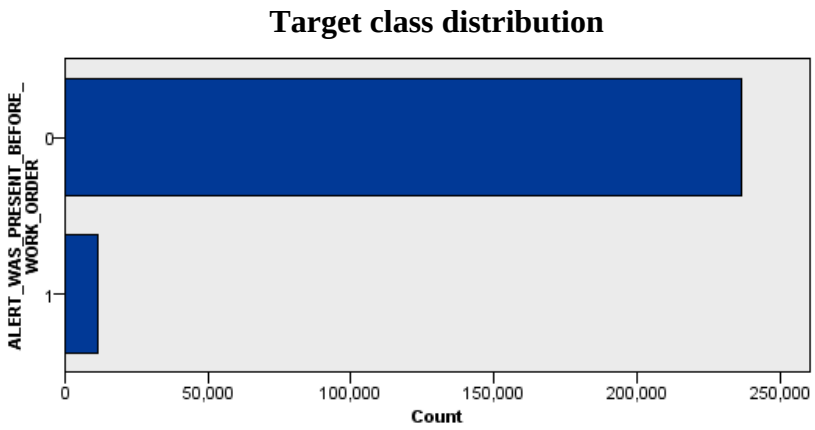

Figure 17 - Imbalanced target class with a ratio of 96:4. Source (author)

4.2.2 **Work orders**

It is important to look at what work order types exist. This study is only interested in work orders created in response to problems in the system. Work order types such as Retrofitting and Moderisation are excluded, as these are not work done to respond to a problem but work that is planned as part of a more long-term maintenance. Only work orders of types "Fault" and "accident" will be used for model training.

Work Orders of type "fault" or "accident" are completed on the same day that they are raised. Table 4 outlines some of the headline statistics about the work orders on the dataset.

Workorders – interesting stats	
Total number of Workorders	4531
Correct creation date set	2701
Number of locations	3076
Relevant types	Fault, accident
Irrelevant types	Prevention, modernisation, retro-fitting, investment
Asset types associated	Chamber, PipeSegment, Substation, Vehicle
Number of Workorders with assets associated	1873

Table 4 - Work orders - interesting statistics

4.2.2.1 Inconsistence creation times

The most relevant discovery made while analysing the Work Order data was when reviewing their creation times. One would assume that faults would occur at random, which seems to be mostly the case for how alerts are distributed (See Figure 15). However, it can be seen in Figure 18 that the distribution is not random. In fact, it seems to demonstrate that the creation of work orders follows a very human pattern. There are hardly any Work Orders created in the small hours of the night and early morning. The numbers pick up as the working day starts. There is also a sharp drop in the number raised at 1pm, which is typically lunchtime for office staff.

One assumption could be that customer load is reduced during night-time at that this reduces the number of faults. However, the drop at lunchtime suggests that the delay is related to operators going on lunch breaks. It seems very possible that the creation times of approximately 25% of Work Orders is affected by human factors. This would have a very big

impact on the accuracy of models create that depend quite a lot on the times that both alerts and Work Orders are created (see Section 4.3.5).

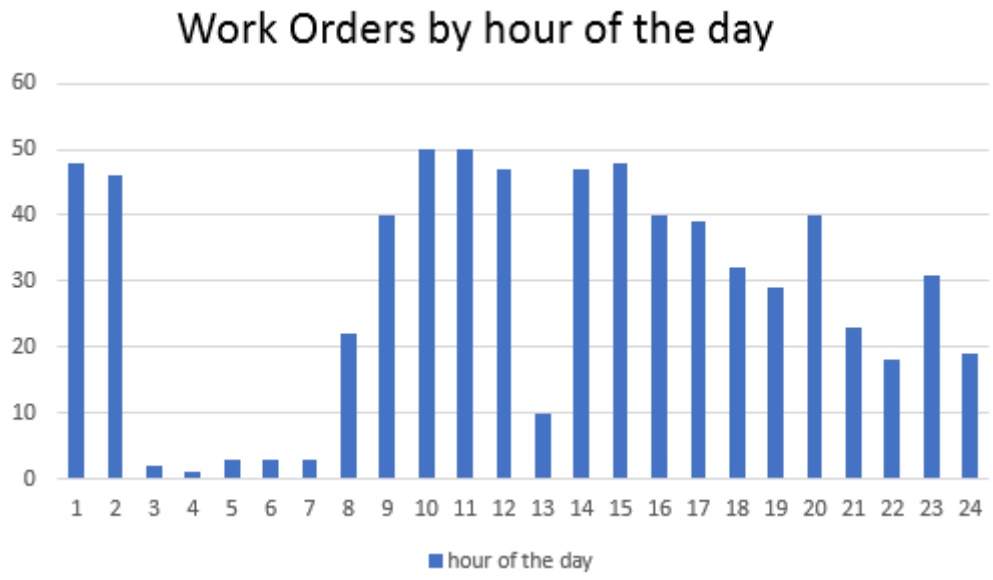

Figure 18 - Number of Work Orders by hour of the day. Source (author)

4.2.3 **Invalid data**

On examining the initial histograms of the data, the most interesting anomalous data was a small number of alert instances that occurred in July. The vast majority of the data occurred in November, December and January. On reviewing this data, it was found to be test data. It was subsequently removed from the dataset.

4.2.3.1 Missing time value in CREATION_DATE field

A large proportion of Work Orders did not have valid creation times set. This was due to a user interface bug. Fortunately, the TARGET_START_DATE is populated with the creation date and time also. This study is only interested in faults that need to be fixed straight away. For that reason, all Work Orders of type "fault" or accident had a TARGET_START_DATE which is equivalent to the CREATION_DATE. The script to fix this issue in detailed in Appendix 8.9.2.1.

4.3 Data preparation

A large amount of the work done on this project was in the pre-processing of data. Characteristics and issues discovered in the data investigation phase need to be addressed so that a dataset can be passed to the modelling phase in the desired format.

4.3.1 Ignored variables

There were many alert variables that were not populated for the dataset used in this study. These columns were included in the database export script, but then filtered out at the variable selection phase.

Discarded variables due to null values:

> DESCRIPTION, CATEGORY, EXTWORKEQUIPMENTID, EXTWORKEQUIPMENTTYPE, ASSET_ID, MEASURE_VALUE, MEASURE_TYPE, MEASURE_UNIT, MEASURE_THRESHOLD_VALUE, MEASURE_THRESHOLD, ADDRESS, ZONE1, ZONE2, ZONE3, OWNER, EVENTSUBTYPE

Discarded variables due to single values:

> DOMAIN, CREATIONTYPE, CREATEDBY, LASTUPDATEDTS, , MODELID, NETWORK, CONTRACTID, EVENT_DATE, ORIGIN_TYPE, ORIGIN_NAME, EVENT_DATA, REMARKS, COSTS, CONSEQUENCES

Discarded as fields are updated after alert creation:

> STATUS, URGENCY, SEVERITY, CERTAINTY, ACK, CASE_DATE, CASE_REFERENCE

4.3.2 Missing data

The columns in Table 5 that will be used in the raw dataset model and none of them have missing data:

	SUBJECT	EVENTTYPE	EVENTID	STARTT	ENDT	LOC	WEEK	MON	DAY	HOUR	DUR
Valid	248018	248018	248018	248018	248018	248018	248018	248018	248018	248018	248018
Missing	0	0	0	0	0	0	0	0	0	0	0

Table 5 - No missing data for raw model input variables

4.3.3 Duration of an alert

Alerts are created to stay active for a fixed amount of time. In this dataset the alerts stay active for 1 hour. This is so that if there are displayed on a dashboard that they stay there for a period of 1 hour to ensure that an operator will be able to take note. If, after this hour-long period, the conditions that caused this alert are still occurring, another alert will be raised. The second alert can be thought of as a continuation of the first alert. The sum of concurrent alerts would seem reasonable as an indicator that might prove predictive in a model. The DB2 function GET_ALERT_DURATION (Appendix 8.7.6) sums up these alert durations to get a cumulative duration value.

4.3.4 Time dimension variables

It is very common practice in data analytics to convert the time variable into as many different time-based values as possible. The DB2 function GET_EVENT_WITH_TARGET (Appendix 8.6.1) also adds fields derived from the STARTTS field:

- WEEK_NUMBER,
- MONTH_NUMBER,
- DAY_OF_THE_WEEK_NUMBER,
- HOUR_OF_THE_DAY

4.3.5 Target variable association

Before supervised learning methods can be applied, there must be a target variable. For this to be the case Alerts and Work Orders need to be linked in some way. In the dataset there is no explicit relation between an alert and a Work Order. At the time of this report, the system user could not create a Work Order from an alert.

This means that, to obtain a training dataset with a target variable, an explicit link must be engineered using domain expert knowledge about how elements in the system are related. Based on consultations with domain experts a method was agreed.

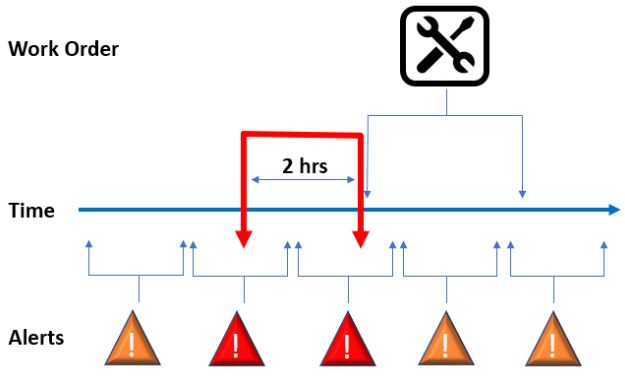

Figure 19 - Linking workorders to alerts that preceded its creation. Source (author)

Association rule:

> An alert will be associated with a Work Order if it occurs within 120 minutes of the Work Order's creation and within a distance of 1000m.

This linking of the data creates a target variable:

> **ALERT_WAS_PRESENT_BEFORE_WORK_ORDER**
>
> This variable denotes whether a Work Order occurred in relation to an alert values: [0,1]

A value of 1 for the target variable means that an alert was present before the work order was created.

4.3.6 Raw dataset

The basic dataset will be the one without any special features. It will contain all of the variables outlined in Section 0. It will also have DURATION_IN_MINUTES which is a derived field. It will have a target variable populated for each instance.

This dataset will be compared with the special features dataset to see if the extra variables help to improve predictions. The creation of this dataset is described next.

4.3.7 Special feature creation (Spatial and temporal aggregation)

Individual alerts may offer much information in themselves as to whether a Work Order will be required or not. However, as they are part of an inter-connected network, the alerts that are active around the same time and space could also provide useful information. This aggregation is particularly useful for sparse sensor data such as what is contained in the dataset of this study.

Figure 20 - Visualisation of related alerts grouped by spatial distance. Source (author)

In Figure 20 it is shown how related alerts are grouped by distance. The alert in the centre is the "alert of interest". The inside circle denotes a radius of 10 metres while the outside circle has a radius of 20 metres.

Therefore, for the alert of interest, the related alerts within a radius of 10m is 2. The related alerts in a radius of 20m is 5.

Related alerts are not only counted in relation to their distance from the alert of interest. They also need to be within a certain time also. This is also varied. When the combinations are done the new variables are:

1. NUM_RELATED_ALERTS_10_500;
2. NUM_RELATED_ALERTS_30_500;
3. NUM_RELATED_ALERTS_60_500;
4. NUM_RELATED_ALERTS_120_500;

5. NUM_RELATED_ALERTS_10_1000;
6. NUM_RELATED_ALERTS_30_1000;
7. NUM_RELATED_ALERTS_60_1000;
8. NUM_RELATED_ALERTS_120_1000

It is a fair assumption to think that many of these new fields will be highly correlated. From the correlation matrix in Table 6 we can see that the value would indicate that they are but it must be remembered that this dataset is imbalanced at a ratio of 96:4. As can be seen in the correlation matrix, many of the values are hovering on mid 90s and there is variance in those correlation values. The predictive importance of these new variables will need to be determined by testing them with models.

	10_500	30_500	60_500	120_500	30_1000	60_1000	120_1000
10_500	0.97	0.95	0.95	0.66	0.64	0.61	0.6
30_500	1	0.95	0.96	0.64	0.66	0.61	0.6
60_500	0.95	1	0.96	0.62	0.62	0.65	0.6
120_500	0.96	0.96	1	0.62	0.62	0.61	0.63
10_1000	0.64	0.62	0.62	1	0.97	0.93	0.93
30_1000	0.66	0.62	0.62	0.97	1	0.94	0.95
60_1000	0.61	0.65	0.61	0.93	0.94	1	0.94
120_1000	0.6	0.6	0.63	0.93	0.95	0.94	1

Table 6 - Correlation matrix of all special features

4.3.8 Dummy variables for categorical input

Some of the input variables in the dataset such as LOCATION, EVENTTYPE and SUBJECT are categorical. For some models such as a logistic regression one, categorical variables are not acceptable. To get around this issue dummy variables are created. The

number of new dummy variables per variable would normally be n-1, where n is the number of levels of the variable.

EVENTTYPE has only levels so I will be transformed to a single dummy variable. However, LOCATION has 69 levels and subject has 241. For these variables the top 10 most frequent values are given dummy variables and the rest are ignored.

4.3.9 Outliers and extremes

The frequency distributions of each input variable were examined. This was to visually check for interesting data instances and outliers. An outlier test was also run for each input variable. When outliers are found they are reviewed to see what way they should be handled. If they are thought to be a flawed instance variable they will be deleted. As the dataset has an imbalanced class variable it was interesting to check if there was any correlation between outliers and the minority class. This would strengthen the case to retain outlier values.

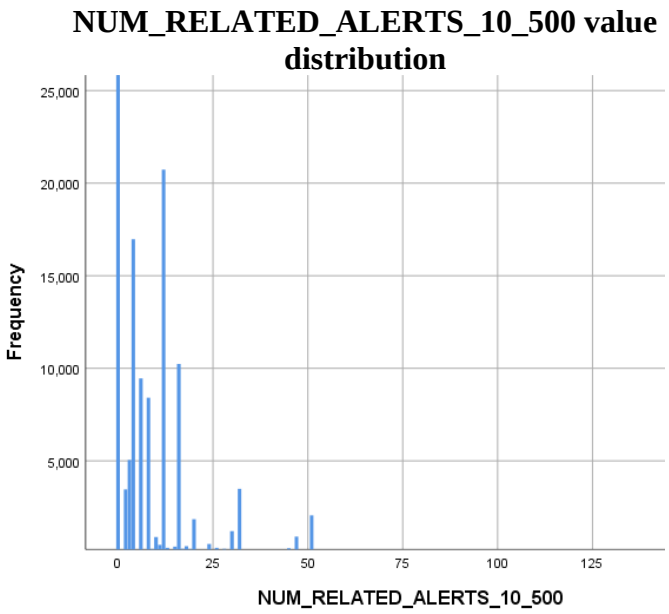

Figure 21 - Histogram of NUM_RELATED_ALERTS_10_500. Source (author)

NUM_RELATED_ALERTS_10_500

This field has top 4 outliers with values of 139. When the histogram for the same feature (Figure 21) is reviewed it can be seen that a value of 139 is indeed and extreme outlier. The boxplot Figure 22 shows the outliers more clearly.

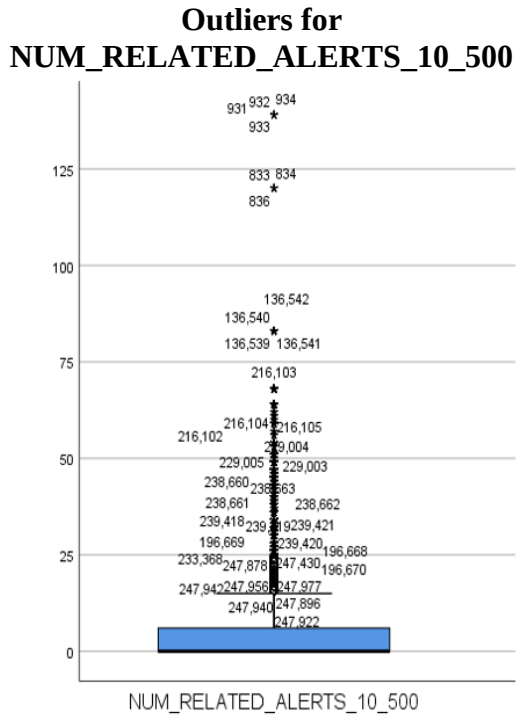

Figure 22 - A boxplot and whiskers shows the outliers more clearly. Source (author)

Having examined these 4 instances no correlation was found between outliers and

minority class. A test with and without these instances on a C5.0 Decision Tree and Random Tree showed no evaluation differences. It was decided to not to delete the outliers. The other special feature variables also had similar outliers. The full outlier report can be seen in the appendix in Section 8.4.

4.3.10 **Other pre-processing tasks**

Some of the other miscellaneous data tasks were:

- Filter out non-fault data such as modernization Work Orders.
- Adding indices to tables
- Creating temporary tables to make certain queries more performant

4.4 Modelling

2 separate experiments will be run. The first will be on the raw dataset where no related alert counts are available. The second experiment will contain these new "special features" the results will be analysed according to Section 3.6 and Section 3.9 and finally the hypothesis will be rejected or fail to be rejected according to Section 3.10.

It is very important that the results from both experiments are evaluated from the same split of the dataset. This is the case for this study were both training, validation and testing partitions are the same for both experiments.

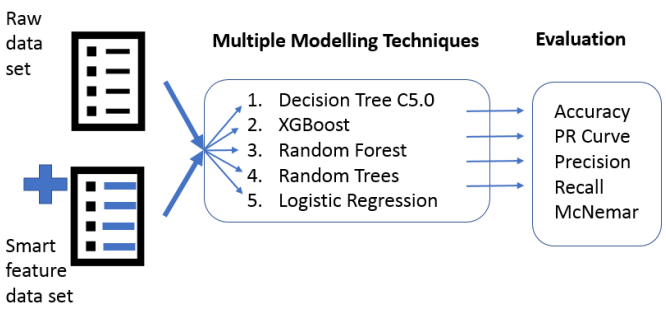

Figure 23 - Modelling with multiple techniques. Source (author)

The lessons learned in the "*No Free Lunch Theorems for Optimization*" (Wolpert & Macready, 1997) are also noted and as such a range of models will be tested.

4.4.1 **Experiment #1 - Raw dataset**

The models tested in this dataset are configured as follows:

	Model type	Algorithm settings	Minority class mitigation
C5.0 Cost x 3	C5.0 Decision Tree	Favour: accuracy	FP = cost x3
Random Forest SMOTE	Random Forest	Num of Trees:10 Max depth: 10	SMOTE Algorithm: Regular K neighbours: 30
Random Tree handle imbalanced SMOTE	Random Tree	Num models: 100 Handle imbal: true Max nodes: 10000 Max depth: 10 Min child node: 5	SMOTE Algorithm: Regular K neighbours: 30
Random Tree handle imbalanced Costx2	Random Tree	Num models: 100 Handle imbal: true Max nodes: 10000 Max depth: 10 Min child node: 5	SMOTE Algorithm: Regular K neighbours: 30 FP = cost x3
XGBoost SMOTE	XGBoost	Tree method: auto Boost round: 10 Max depth: 10 Min child weight: 1	SMOTE Algorithm: Regular K neighbours: 30

Table 7 - Configuration of models for the raw dataset

4.4.2 Experiment 2 # - Special features dataset

	Model type	Algorithm settings	Minority class mitigation
C5.0 Boosting	C5.0 Decision Tree	Favour: accuracy	Use boosting: true
C5.0 SMOTE	C5.0 Decision Tree	Favour: accuracy	SMOTE Algorithm: Regular K neighbours: 30
C5.0 COST x3	C5.0 Decision Tree	Favour: accuracy	FP = cost x3
XGBoost SMOTE	XGBoost	Tree method: auto Boost round: 10 Max depth: 10 Min child weight: 1	SMOTE Algorithm: Regular K neighbours: 30
Random Tree handle Imbal	Random Tree	Num models: 100 Max nodes: 10000 Max depth: 10 Min child node: 5	Handle imbal: true
XGBoost SMOTE	XGBoost	Tree method: auto Boost round: 10 Max depth: 10 Min child weight: 1	SMOTE Algorithm: Regular K neighbours: 30
Reduced Random Forest	Random Forest	Num of Trees:10 Max depth: 10	SMOTE Algorithm: Regular K neighbours: 30
Random Forest SMOTE	Random Forest	Num of Trees:10 Max depth: 10	SMOTE Algorithm: Regular K neighbours: 30
Logistic regression SMOTE	Logistic regression	Method: Enter Procedure: Binomial	SMOTE Algorithm: Regular K neighbours: 30

Table 8 - Configuration of models for the special features dataset

4.5 Results

4.5.1 Experiment #1 - Raw dataset

The first dataset was used to create 5 models. They all used various methods to guard against the imbalanced target class problem. From observing the confusion matrices (Table 10), ROC

curves (Figure 24) and Precision-Recall curves (Figure 25) it can be seen that it is close between the *C5.0 Cost x 3* model and *XGBoost SMOTE* . They both have high values for True Positives and True Negatives on opposite sides of the confusion matrix. The C5.0 Cost x 3 model has more balanced Precision and Recall values.

	C5.0 Cost x 3	XGBoost SMOTE
Accuracy	0.9427	0.7301
Precision	0.4517	0.7597
Recall	0.4101	0.1232
Specificity	0.9723	0.7286

Table 9 - Compare evaluation metrics of top models on the raw dataset

The complete set of evaluation metrics are outlined in Appendix 8.1.1.

4.5.1.1 Confusion Matrix

C5.0 Cost x 3		
	0	1
0	45,853	1,546
1	1,305	1,075
XGBoost SMOTE		
	0	1
0	34,534	12,865
1	572	1,808

Table 10 - Confusion matrix for the raw dataset

The rest of the matrix results are in Appendix 8.1.3

4.5.1.2 Curves and AUC

Figure 24 shows the ROC curve and how the *C5.0 Cost x 3* model and XGBoost SMOTE do better on opposite sides of the curve. The *C5.0 Cost x 3* model has the higher ROC AUC value (Table 11). It also has the more balance Precision-Recall ratio (Figure 25).

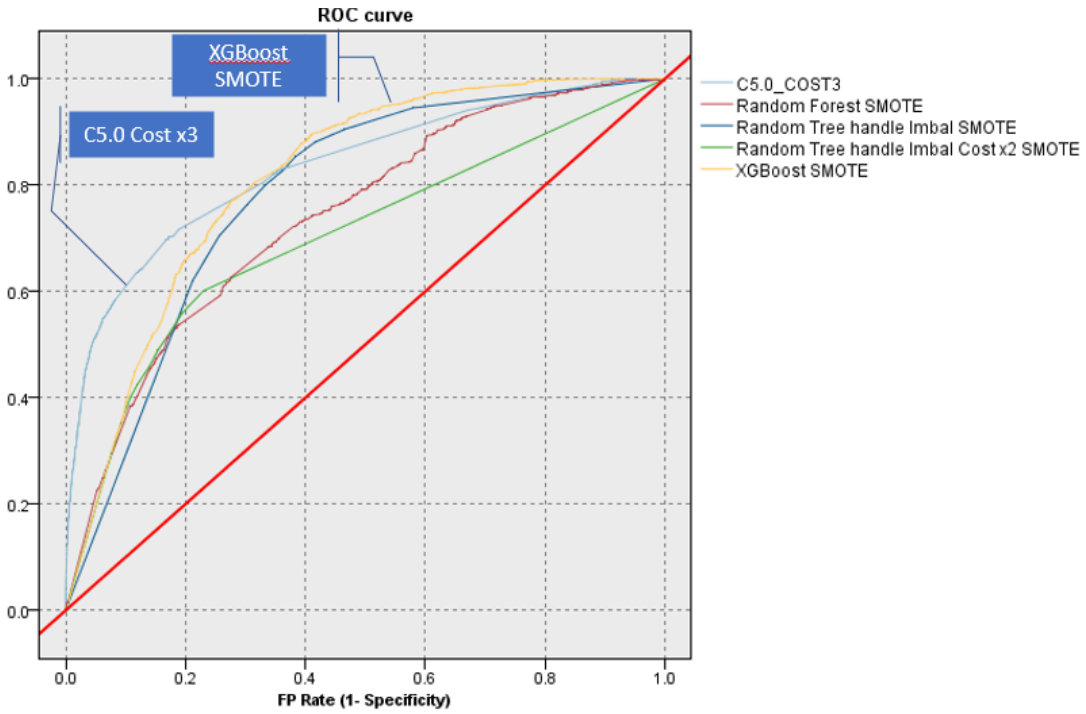

Figure 24 - Raw dataset ROC curve. Source (author)

	AUC	Gini
C5.0 Cost x3	0.834	0.669
XGBoost SMOTE	0.811	0.623
Random Tree handle Imbal SMOTE	0.75	0.5
Random Forest SMOTE	0.739	0.478
Random Tree handle Imbal Cost x2 SMOTE	0.701	0.402

Table 11 - ROC Curve AUC values for the raw dataset

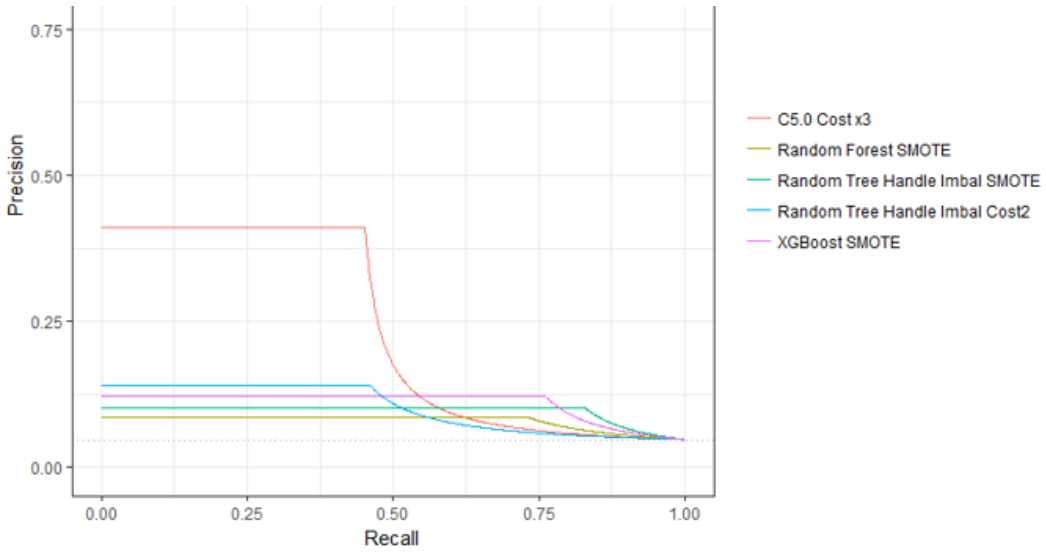

Figure 25 - Precision-Recall curves for raw dataset. Source (author)

4.5.1.3 Predictive importance

HOUR_OF_THE_DAY leads the predictive importance table for the C5.0 Cost x 3 model. The other input variables are making decent contributions also. The rest of the predictor importance graphs are in Appendix 8.1.4.

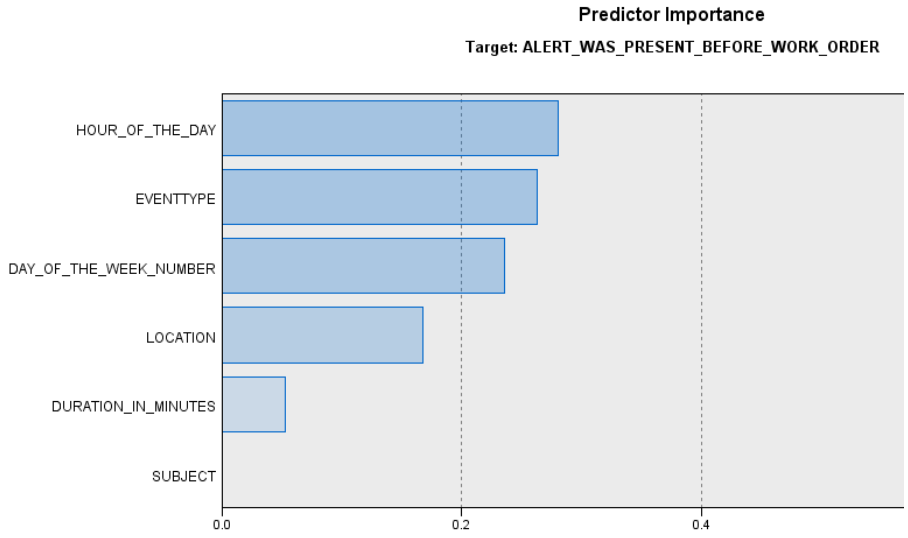

Figure 26 - Predictive importance of the input variables for the Raw C5.0 cost x3 model. Source (author)

4.5.1.4 Tuning

Based in the initial results of the experiments, tuning was performed to improve results. The most effective tuning was achieved using misclassification cost values. A cost of 3 for misclassifying the positive case was found to give the best results for the C5.0 model.

For tree-based models tree depth and pruning settings were varied. The tree depth was set to a maximum of 10. Deeper trees gave results that seemed to be overfitting the data.

4.5.2 **Experiment 2 # - Special features dataset**

The SMOTE algorithm works much better on this dataset for the *C5.0 SMOTE*. The *C5.0 Cost x 3* is performing on the top end of the trialled models also. The ROC curves are very close (Figure 25) to each other and the ROC AUC value (Table 14) is very close. However, in the PR curve (Figure 28) it can be seen that *C5.0 Cost x 3* has the best ratio value.

	C5.0 Cost x 3	C5.0 SMOTE
Accuracy	0.9590	0.9371
Precision	0.5630	0.3966
Recall	0.6349	0.6038
Specificity	0.9753	0.9539

Table 12 - Compare evaluation metrics of top models on the special features dataset

The complete set of evaluation metrics are outlined in Appendix 8.2.1.

4.5.2.1 Confusion Matrix

C5.0 Cost x 3		
	0	**1**
0	46,226	1,173
1	869	1,511
C5.0 SMOTE		
	0	**1**
0	45,213	2,186
1	943	1,437

Table 13 - Confusion matrix for the top model

4.5.2.2 Curves and AUC

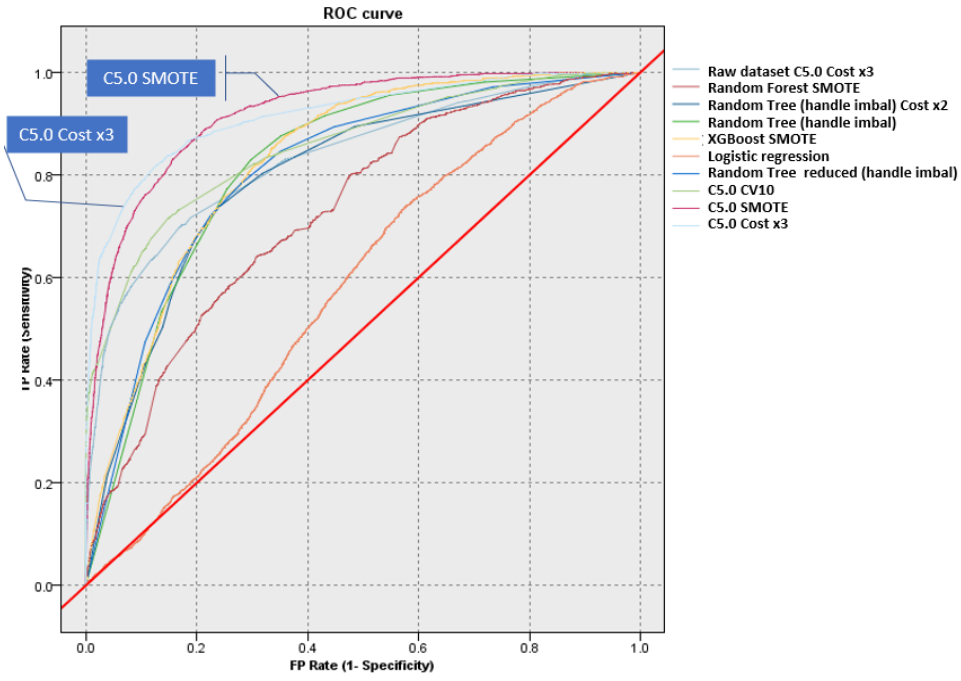

Figure 27 - ROC curves for special features dataset with C5.0 SMOTE and C5.0 Cost x3 showing the best AUC values. Source (author)

	AUC	Gini
C5.0 SMOTE	0.919	0.837
C5.0 COST x3	0.917	0.833
C5.0	0.856	0.712
XGBoost	0.837	0.674
**Raw C5.0 Cost x3*	0.834	0.669
XGBoost SMOTE	0.828	0.656
Random Tree handle Imbal	0.819	0.637
XGBoost SMOTE	0.811	0.623
Reduced Random Forest	0.81	0.62
Random Tree handle Imbal Cost x2	0.796	0.592
**Raw* Random Tree handle Imbal SMOTE	0.75	0.5
**Raw* Random Forest SMOTE	0.739	0.478
Random Forest SMOTE	0.729	0.457
**Raw* Random Tree handle Imbal Cost x2 SMOTE	0.701	0.402
Logistic regression	0.679	0.159

Table 14 - ROC Curve AUC values, special features dataset

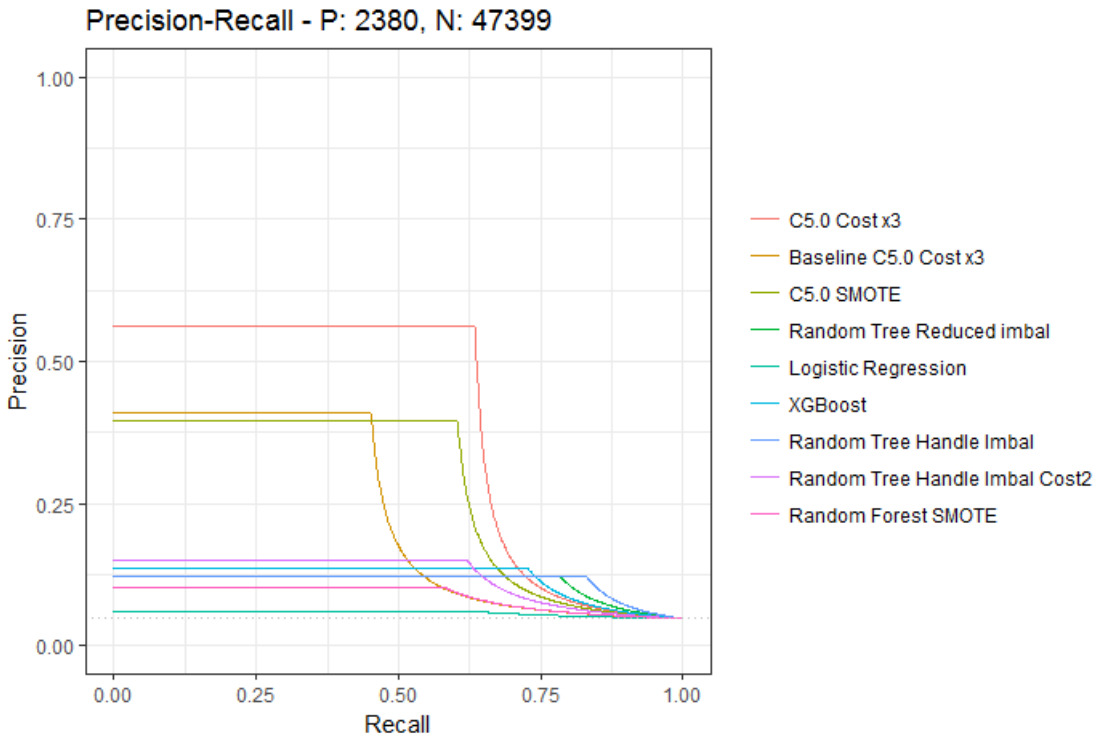

Figure 28 - Precision-Recall curve shows most performant model is C5.0 with misclassification costs. Source (author)

4.5.2.3 Predictive importance

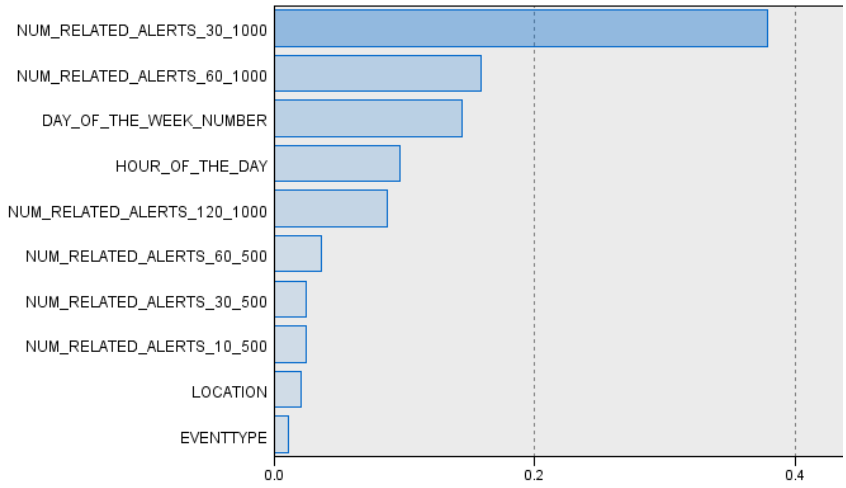

Figure 29 - Predictive power of input variables for the C5.0 Costx3 model. Source (author)

4.5.2.4 Tuning

The tuning on the second dataset was done in a similar way to that of the first dataset.

4.6 Comparing Raw versus Special Features datasets

For the "C5.0 Cost x 3" model there was a noticeable lift (Figure 30) and both the ROC and PR curves when using the dataset with the special features. This was not the case for all models however. For example, for XGBoost in Figure 45 and Random Forest SMOTE in Figure 47 the improvement was minimal to non-existent.

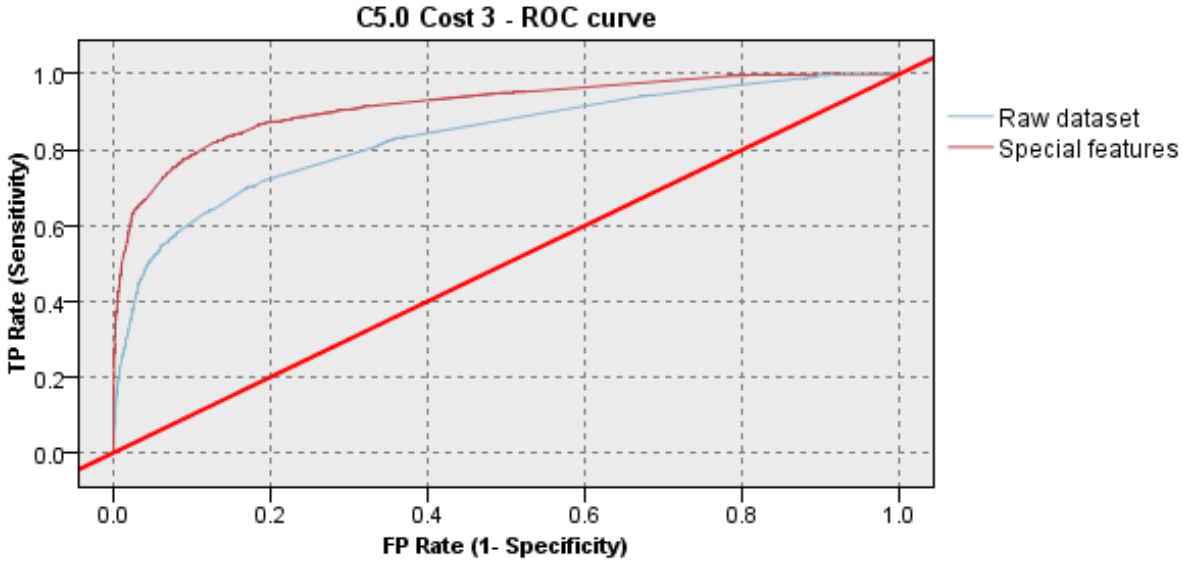

Figure 30 - Comparative ROC curve between raw and special feature datasets for the C5.0 Cost x 3 model. Source (author)

4.7 Hypothesis testing

The McNemar test is used to test the null hypothesis. It is setup against a "zombie" classifier (Appendix 8.5) that guesses false 96% of the time and true 4% of the time. This is the same proportion as the class imbalance.

The McNemar test (Table 16) was performed on the C5.0 Cost x3 model against the baseline classifier. This was to see if the results of the C5.0 Cost x3 are statistically significant.

The significance level $\alpha = 0.05$. A paired McNemar test is done.

The McNemar test's statistic (S) is compared to the relevant critical values (CV). As P = .000 and $P \leq \alpha$ (0.05), the Hypothesis $\mathbf{H_0}$ can be rejected.

		C5.0 COSTx3		
		0	1	Total
Baseline	0	45240	2589	47829
Model	1	1855	95	1950
Total		47095	2684	49779

Table 15 - Matrix of baseline model predictions versus C5.0 COSTx3 model predictions in the raw dataset

Chi-Square Tests

McNemar Test		.000[a]
N of Valid Cases	49779	

a. Binomial distribution used.

Table 16 - McNemar test of the baseline model versus the C5.0 COSTx3 model

5 Analysis, evaluation and discussion

5.1.1 Handling the imbalanced target variable

In the very early tests using algorithms with no allowance for the minority class, there were some very imbalanced results where the negative class has predicted almost 100% of the time. Once the various methods were applied, the best methods to work with this dataset emerged as SMOTE and misclassification costs.

Neither the under-sampling or boosting seemed to have much of a helpful effect on results.

5.1.2 Experiment #1 - Raw dataset

As previously mentioned accuracy values are ignored as they are not a reliable measure when dealing with an imbalanced target variable. The best way to get a quick comparative overview is by looking at the ROC and Precision-Recall curves.

As can be seen in the ROC curve (Figure 24) both the *C5.0_Costx3* (C5.0 Decision Tree with misclassification cost of 3) and the *XGBoost SMOTE* performed the best with ROC AUC values of 0.83 and 0.81. These AUC values are quite different, and this is even more pronounced when comparing them on the Precision-Recall (PR) graph (Figure 25). It is clear that the *C5.0_Costx3* is the best model developed from the raw dataset from these 2 graphs.

Once the *C5.0_Costx3* is identified as the best model the confusion matrix can be examined. As can be seen in Table 10, the model has a little over half a chance of predicting the need for a Work Order when one is required.

It is interesting that the decision tree out performs the other more advanced algorithms. It seems that configuring misclassification costs works well for this imbalanced dataset. SMOTE and algorithms with an option to handle imbalance did not yield results that were nearly as good. However, it was also noted that as the misclassification cost was increased the number of true positives and the number of false negatives traded places. This is acceptable as the main use case is most interested in true positives.

Another surprising finding was the order of predictive importance (Figure 26). Intuitively it was expected that DURATION_MINUTES would have the highest value. However, HOUR_OF_THE_DAY, EVENTTYPE and DAY_OF_THE_WEEK were almost 4 times as

predictive. This would suggest that time that an alert happens is quite important for prediction but the length of time that the alert stays active is much less so.

Next it is examined if the special features added through aggregation yielded much of a performance improvement.

5.1.3 Experiment 2 # - Special features dataset

For the most part the models with the special features performed better except for *Random Forest SMOTE* which had a lower ROC AUC value (Table 14). AUC values for all other models that were run in both had improvements. Doing a McNemar test between the *C5.0_Costx3* model for both the raw and special features will confirm whether the improvement is significant.

With a value P = 0.63, therefore P > α (0.05). This means that any improvement is not statistically significant.

BL_C5.0_COST3 * C5.0 COSTx3 Crosstabulation

Count

		C5.0 COSTx3		
		0	1	Total
Raw_C5.0_COST3	0	46140	1018	47158
	1	955	1666	2621
Total		47095	2684	49779

Table 17 - Matrix of the raw dataset C5.0 COSTx3 model predictions versus C5.0 COSTx3 model predictions

Chi-Square Tests

	Value	Exact Sig. (2-sided)
McNemar Test		.163[a]
N of Valid Cases	49779	

a. Binomial distribution used.

Table 18 - Results of the McNemar test performed on the 2x2 matrix of raw dataset C5.0 COSTx3 and special features C5.0 COSTx3 model predictions

6 Conclusion

This study proved that there is a relationship between alerts (created from thresholding reading values) and work orders. Even in a sparse IoT network the link was statistically significant. However, it would not be advisable to launch this approach in any widespread manner until further investigations are made.

The possible reasons for poor accuracy achieved in this study are many. As mentioned the network contained a small number (69) of IoT sensors. The faults were often occurring in locations were there were no sensor nearby. This makes it very hard to detect issues. Also, as this was a District heating network, the issue was further exasperated by "Secondary networks" which also did not have IoT sensors.

Other possible causes for the low performance of the models could be:

1. Poorly configured thresholds
2. External influences that are not monitored by the system (weather)
3. A human factor causing delays to Work order creation times (Section 4.2.2.1)
4. A small dataset spanning 3 months that was not enough especially when dealing with a minority target class.
5. Outages in the alert streaming service leaving gaps in the dataset

Much of the work involved in this project was around pre-processing of the data and ensuring that the issue of the minority target class variable considered in both training and evaluating the models. It was noted that misclassification costs are a powerful way to handle this imbalance.

There were many limitations on the scope of this project which leave ample room for future work and new methodologies to be employed.

6.1 Future work & recommendations

The most important next step in this process is to test the findings on new datasets. It is particularly difficult to make alert predictions in the District Heating domain. This same process should be tested on water, waste water and energy domains to see how it performs there. Testing the methodology on different city networks would also prove that the solution is not specific to a single network setup.

It would be preferable to have a dataset that is more fully populated with alert input fields (ORIGIN, READING _VALUE) to gain better insight into the true relationship between system alerts and work orders. In the dataset for this study many of the available alert fields were not populated by the customer. If more of the fields available in the alert model are populated it should lead to better predictive results.

Improvements in aggregation methods could also be made. For this study the alert data was aggregated by counting the number of alerts around each individual alert that is active at point in time using database geospatial queries. An improvement on this method would be to use a clustering approach such as K-means described in Section 2.5.5.

Another tweak to the aggregation method would be to create a model that is based on the number of links between sensors instead using distance between alerting sensor. This would be a model that is based on relational distance instead of geospatial distance.

The use of additional external sources such as weather data to augment the alert data would also be an intuitive next step. Weather is particularly interesting in relation to District Heating network as it has a direct impact on customer demand.

The idea of having an intermediate step to processing raw reading data is an interesting one. This study looked at threshold values that were configured to find actual alerts. Another idea would be to broaden these threshold ranges to let more data through as "interesting readings" and then apply modelling techniques to these.

6.2 Conflict of Interests

I wish to acknowledge that while writing this report I was employed by IBM working on the IOC smarter cities product. IBM also provided the funding to complete this research.

Bibliography

Arampatzis, T., Lygeros, J., & Manesis, S. (2005). A Survey of Applications of Wireless Sensors and Wireless Sensor Networks. In *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.* (pp. 719–724). https://doi.org/10.1109/.2005.1467103

Bhowmick, A., Francellino, E., Glehn, L., Loredo, R., Nesbitt, P., & Yu, S. W. (2012). Ibm intelligent operations center for smarter cities administration guide. *IBM International Technical Support Organization*.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Chang, H., Tai, Y., & Hsu, J. Y. (2010). Context-Aware Taxi Demand Hotspots Prediction. *Int. J. Bus. Intell. Data Min.*, *5*(1), 3–18. https://doi.org/10.1504/IJBIDM.2010.030296

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Chen, F., Deng, P., Wan, J., Zhang, D., Vasilakos, A. V., & Rong, X. (2015). Data Mining for the Internet of Things: Literature Review and Challenges. *International Journal of Distributed Sensor Networks*, *11*(8), 431047. https://doi.org/10.1155/2015/431047

Chua, V. S., Esquivel, J. Z., Paul, A. S., Techathamnukool, T., Fajardo, C. F., Jain, N., … Iyer, R. (2017). Visual IoT: Ultra-Low-Power Processing Architectures and Implications. *IEEE Micro*, *37*(6), 52–61. https://doi.org/10.1109/MM.2017.4241343

Ciechalski, J. C., Pinkney, J. W., & Weaver, F. S. (2002). *A Method for Assessing Change in Attitude: The McNemar Test*. Retrieved from https://eric.ed.gov/?id=ED464933

Davis, P., Sullivan, E., Marlow, D., & Marney, D. (2013). A selection framework for infrastructure condition monitoring technologies in water and wastewater networks. *Expert Systems with Applications*, *40*(6), 1947–1958. https://doi.org/10.1016/j.eswa.2012.10.004

Domingos, P. (1999). The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery*, *3*(4), 409–425. https://doi.org/10.1023/A:1009868929893

Fang, S., Xu, L., Pei, H., Liu, Y., Liu, Z., Zhu, Y., … Zhang, H. (2014). An Integrated Approach to Snowmelt Flood Forecasting in Water Resource Management. *IEEE Transactions on Industrial Informatics*, *10*(1), 548–558. https://doi.org/10.1109/TII.2013.2257807

Fletcher, T. D., Andrieu, H., & Hamel, P. (2013). Understanding, management and modelling of urban hydrology and its consequences for receiving waters: A state of the art. *Advances in Water Resources*, *51*(Supplement C), 261–279. https://doi.org/10.1016/j.advwatres.2012.09.001

Gadd, H., & Werner, S. (2015). Fault detection in district heating substations. *Applied Energy*, *157*, 51–59. https://doi.org/10.1016/j.apenergy.2015.07.061

Gaddam, A. (2014, September 14). Designing a Wireless Sensors Network for Monitoring and Predicting Droughts [Paper presented at a conference, workshop or other event, and published in the proceedings]. Retrieved November 26, 2017, from http://researcharchive.wintec.ac.nz/4017/

Guo, H., & Viktor, H. L. (2004). Learning from Imbalanced Data Sets with Boosting and Data Generation: The DataBoost-IM Approach. *SIGKDD Explor. Newsl.*, *6*(1), 30–39. https://doi.org/10.1145/1007730.1007736

Haibo He, & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Kanakoudis, V., & Tolikas, D. K. (2001). The Role of Leaks and Breaks in Water Networks – Technical and Economical Solutions. *Journal of Water Supply: Research and Technology - AQUA*, *50*, 301–311.

Kingston, G. B., Lambert, M. F., & Maier, H. R. (2005). Bayesian training of artificial neural networks used for water resources modeling. *Water Resources Research*, *41*(12), W12409. https://doi.org/10.1029/2005WR004152

Le Gat, Y., & Eisenbeis, P. (2000). Using maintenance records to forecast failures in water networks. *Urban Water*, *2*(3), 173–181. https://doi.org/10.1016/S1462-0758(00)00057-1

Liu, X. Y., Wu, J., & Zhou, Z. H. (2009). Exploratory Undersampling for Class-Imbalance Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550. https://doi.org/10.1109/TSMCB.2008.2007853

Ma, Y., Guo, Y., Tian, X., & Ghanem, M. (2011). Distributed Clustering-Based Aggregation Algorithm for Spatial Correlated Sensor Networks. *IEEE Sensors Journal*, *11*(3), 641–648. https://doi.org/10.1109/JSEN.2010.2056916

Maier, H., & Dandy, G. (1996). The Use of Artificial Neural Networks for the Prediction of Water Quality Parameters. *Water Resources Research - WATER RESOUR RES*, *32*, 1013–1022. https://doi.org/10.1029/96WR03529

Martínez-Codina, Á., Castillo, M., Gonzalez-Zeas, D., & Garrote, L. (2015). Pressure as a predictor of occurrence of pipe breaks in water distribution networks. *Urban Water Journal*, 1–11. https://doi.org/10.1080/1573062X.2015.1024687

Mattern, F., & Floerkemeier, C. (2010). From the Internet of Computers to the Internet of Things. In *From Active Data Management to Event-Based Systems and More* (pp. 242–259). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17226-7_15

Mounce S. R., Boxall J. B., & Machell J. (2010). Development and Verification of an Online Artificial Intelligence System for Detection of Bursts and Other Abnormal Flows. *Journal of Water Resources Planning and Management*, *136*(3), 309–318. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000030

Mounce, S. R., Day, A. J., Wood, A. S., Khan, A., Widdop, P. D., & Machell, J. (2002). A neural network approach to burst detection. *Water Science and Technology*, *45*(4–5), 237–246.

Mounce, S. R., Mounce, R. B., Jackson, T., Austin, J., & Boxall, J. B. (2014). Pattern matching and associative artificial neural networks for water distribution system time series data analysis. *Journal of Hydroinformatics*, *16*(3), 617–632. https://doi.org/10.2166/hydro.2013.057

Niemczynowicz, J. (1999). Urban hydrology and water management – present and future challenges. *Urban Water*, *1*(1), 1–14. https://doi.org/10.1016/S1462-0758(99)00009-6

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys Tutorials*, *16*(1), 414–454. https://doi.org/10.1109/SURV.2013.042313.00197

Ryu, S., Noh, J., & Kim, H. (2016). Deep Neural Network Based Demand Side Short Term Load Forecasting. *Energies*, *10*(1), 3. https://doi.org/10.3390/en10010003

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

Sorensen, S. L. (1997). SMART Mapping for Law Enforcement Settings: Integrating GIS and GPS for Dynamic, Near-real Time Applications and Analysis. In *Crime Mapping and Crime Prevention* (pp. 349–378). Criminal Justice Press.

Tse, P. W. (2002). Maintenance practices in Hong Kong and the use of the intelligent scheduler. *Journal of Quality in Maintenance Engineering*, *8*(4), 369–380. https://doi.org/10.1108/13552510210448540

Vilajosana, I., Llosa, J., Martinez, B., Domingo-Prieto, M., Angles, A., & Vilajosana, X. (2013). Bootstrapping smart cities through a self-sustainable model based on big data flows. *IEEE Communications Magazine*, *51*(6), 128–134. https://doi.org/10.1109/MCOM.2013.6525605

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

Woods, E., & Goldstein, N. (2014). Navigant research leaderboard report: Smart city suppliers. In *Assessment of strategy and execution for 15 smart city suppliers*.

Zehnder, P., & Riemer, D. (2017). Modeling self-service machine-learning agents for distributed stream processing. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 2203–2212). https://doi.org/10.1109/BigData.2017.8258170

Zorić, B., Bajer, D., & Martinović, G. (2016). Employing different optimisation approaches for SMOTE parameter tuning. In *2016 International Conference on Smart Systems and Technologies (SST)* (pp. 191–196). https://doi.org/10.1109/SST.2016.7765657

Appendix

8.1 Raw dataset results

8.1.1 Evaluation metrics

Model	Accuracy	Recall	Specificity	Precision
C5.0 Cost x 3				
	0.9427	0.4101	0.9723	0.4517
Random Forest SMOTE				
	0.6118	0.0853	0.9783	0.7319
Random Tree handle imbalanced SMOTE				
	0.6671	0.7571	0.6626	0.1013
Random Tree handle imbalanced Costx2				
	0.8410	0.4592	0.8601	0.1415
XGBoost SMOTE				
	0.7301	0.7597	0.7286	0.1232

Table 19 - Raw dataset evaluation metrics for each model

8.1.2 Curves

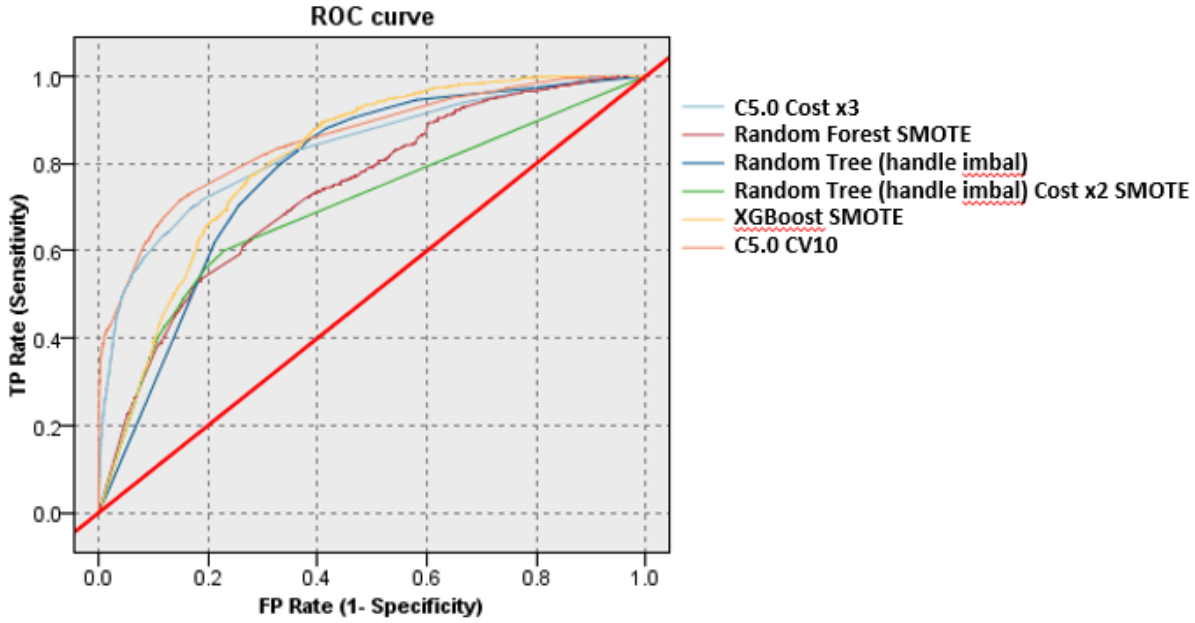

Table 20 - raw dataset ROC curves of all models. Source (author)

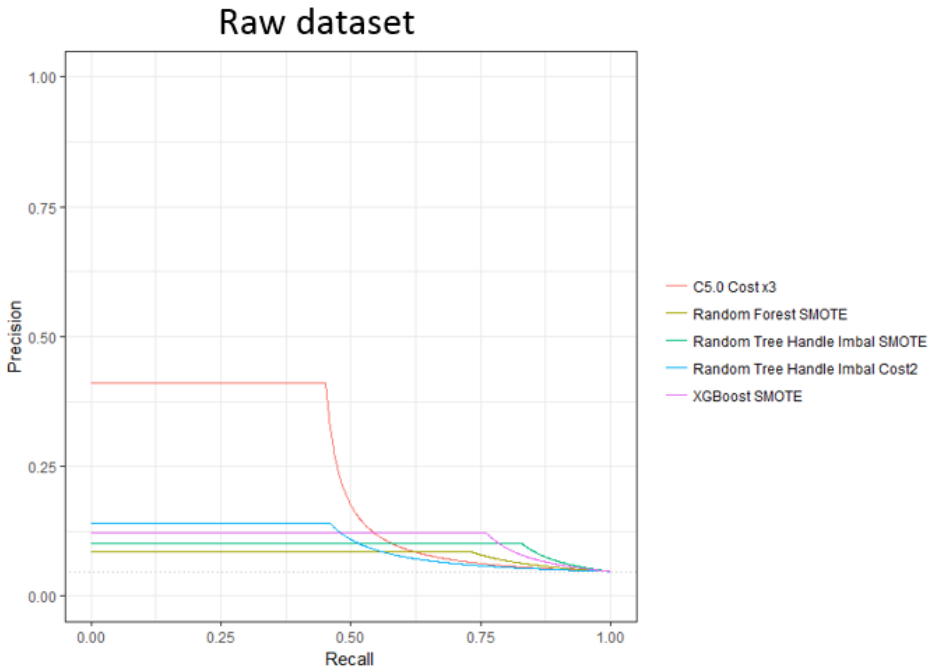

Figure 31 - Precision-Recall curves for raw dataset. Source (author)

8.1.3 Confusion matrices

C5.0 Cost x 3		
	0	1
0	45,853	1,546
1	1,305	1,075
Random Forest SMOTE		
	0	1
0	28,714	18,685
1	638	1,742
Random Tree handle imbalanced SMOTE		
	0	1
0	31,406	15,993
1	578	1,802
Random Tree handle imbalanced Costx2		
	0	1
0	40,770	6,629
1	1,287	1,093
XGBoost SMOTE		
	0	1
0	34,534	12,865
1	572	1,808

Table 21 - Confusion matrices for all raw dataset models. Source (author)

8.1.4 **Predictive importance**

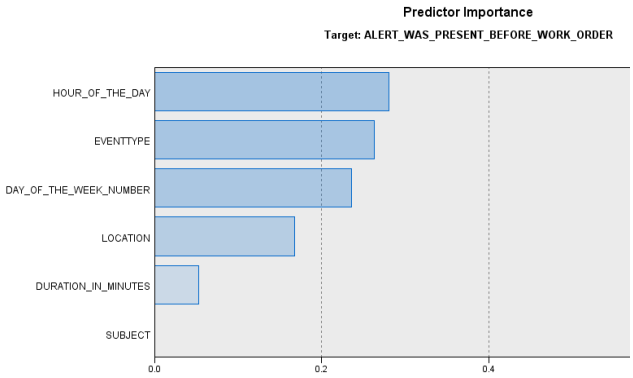

Figure 32 - Predictive importance of input variables for Raw 5.0 Cost x3. Source (author)

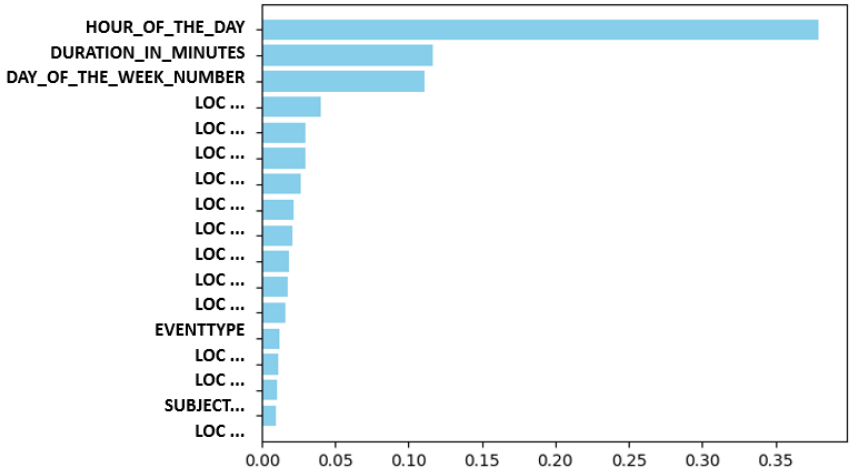

Figure 33 - Predictive importance for Random Forest SMOTE. Source (author)

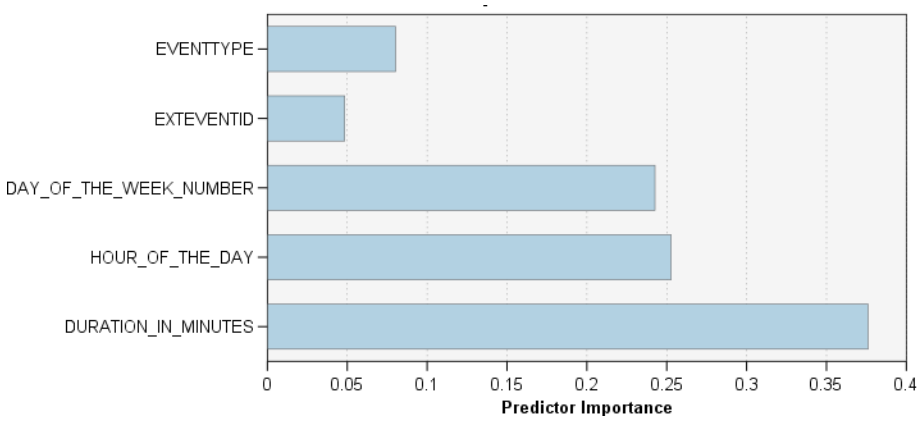

Figure 34 - predictive importance for Random Tree (handling imbalanced) . Source (author)

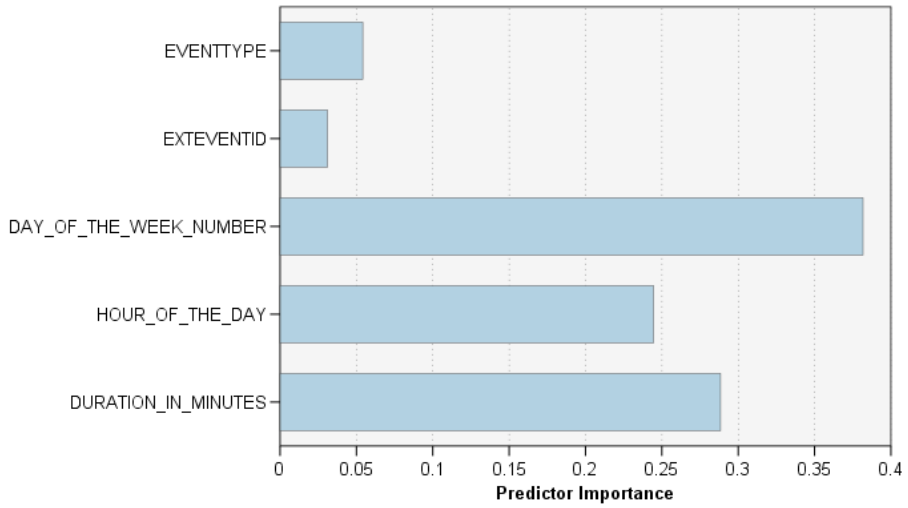

Figure 35 - Predictor importance for Random Tree (handling imbalanced, cost x2) SMOTE. Source (author)

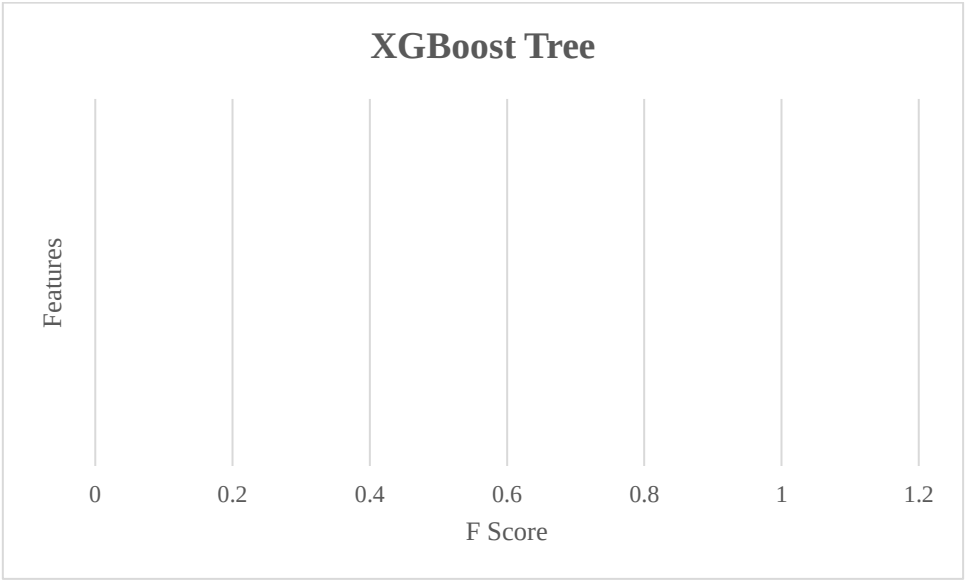

Figure 36 - Predictor importance for XGBoost tree SMOTE. Source (author)

8.2 Special features dataset results

8.2.1 Evaluation metrics

Model	Accuracy	Recall	Specificity	Precision
Random Tree handle imbal cost x2				
	0.8123	0.3442	0.8259	0.0544
Random Tree handle imbal				
	0.7082	0.8298	0.7020	0.1227
XGBoost SMOTE				
	0.7653	0.7277	0.7672	0.1357
Logistic Regression SMOTE				
	0.5085	0.6357	0.5021	0.0603
XGBoost				
	0.9522	0.0017	1.0000	0.8000
C5.0 CV 10				
	0.9660	0.3517	0.9969	0.8489
C5.0 Boosting				
	0.9646	0.3807	0.9939	0.7594
C5.0 SMOTE				
	0.9371	0.6038	0.9539	0.3966
C5.0 Cost x 3				
	0.9590	0.6349	0.9753	0.5630

Table 22 – Special feature dataset evaluation metrics for each model

8.2.2 Curves

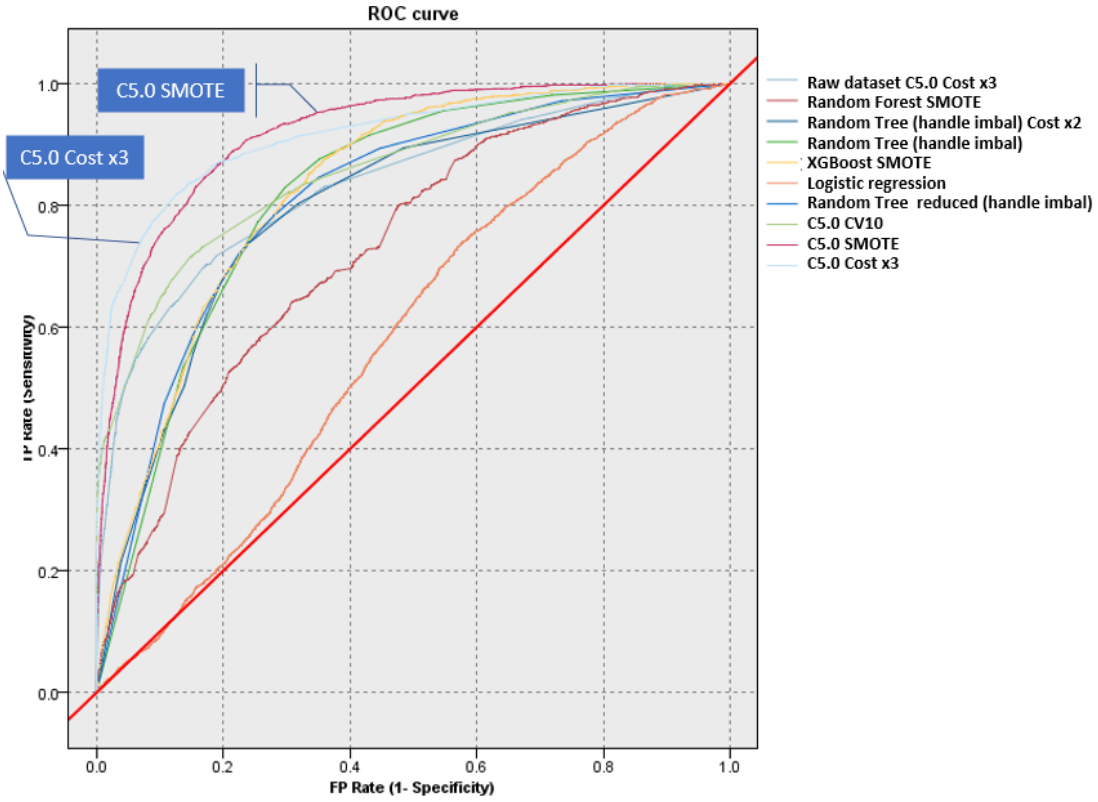

Figure 37 - ROC curves for special feature dataset. Source (author)

8.2.3 **Confusion matrices**

Random Tree handle imbal cost x2		
	0	**1**
0	39,149	8,250
1	905	1,475
Random Tree handle imbal		
	0	**1**
0	33,276	14,123
1	405	1,975
XGBoost SMOTE		
	0	**1**
0	36,366	11,033
1	648	1,732
Logistic Regression SMOTE		
	0	**1**
0	23,801	23,598
1	867	1,513
XGBoost		
	0	**1**
0	47,398	1
1	2,376	4
C5.0 CV 10		
	0	**1**
0	47,250	149
1	1,543	837
C5.0 Boosting		
	0	**1**
0	47,112	287
1	1,474	906
C5.0 SMOTE		
	0	**1**
0	45,213	2,186
1	943	1,437
C5.0 Cost x 3		
	0	**1**
0	46,226	1,173
1	869	1,511

Table 23 - Special features dataset confusion matrix

8.2.4 **Predictive importance**

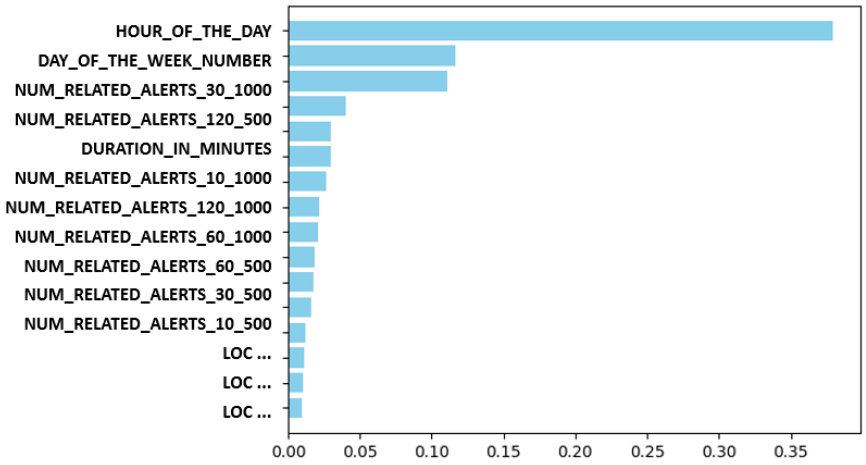

Figure 38 - Predictor importance Random Forest SMOTE. Source (author)

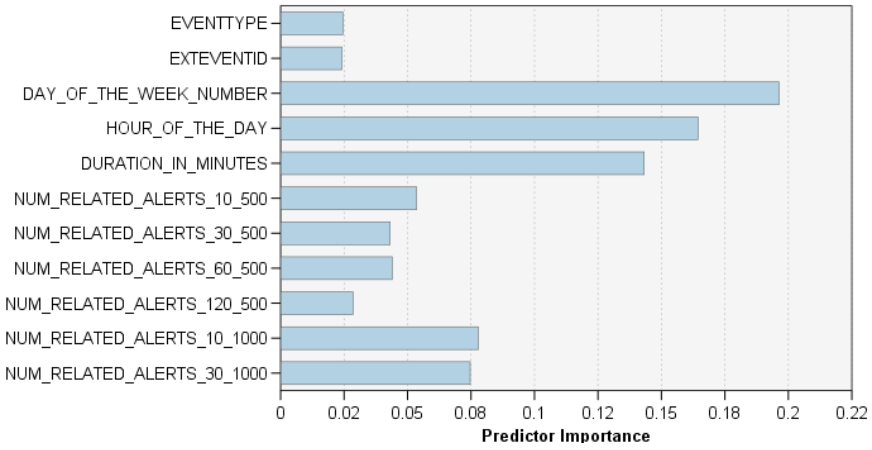

Figure 39 - Predictor importance for Random Trees (handling imbalance, cost x2) . Source (author)

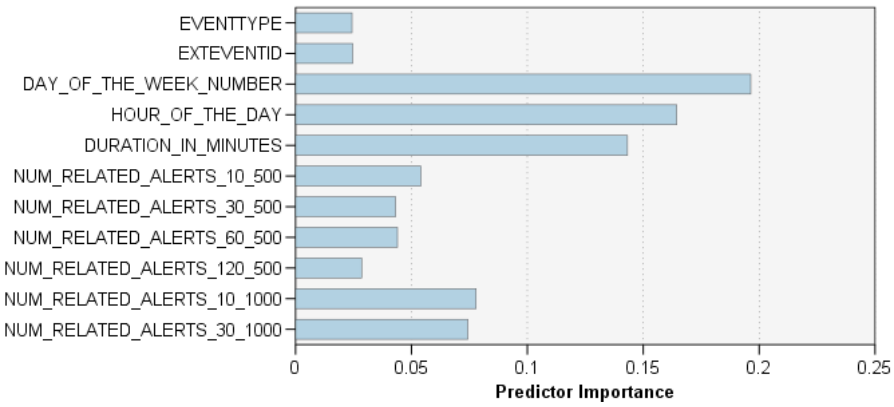

Figure 40 - Predictive importance for Random Trees (handle imbalanced) . Source (author)

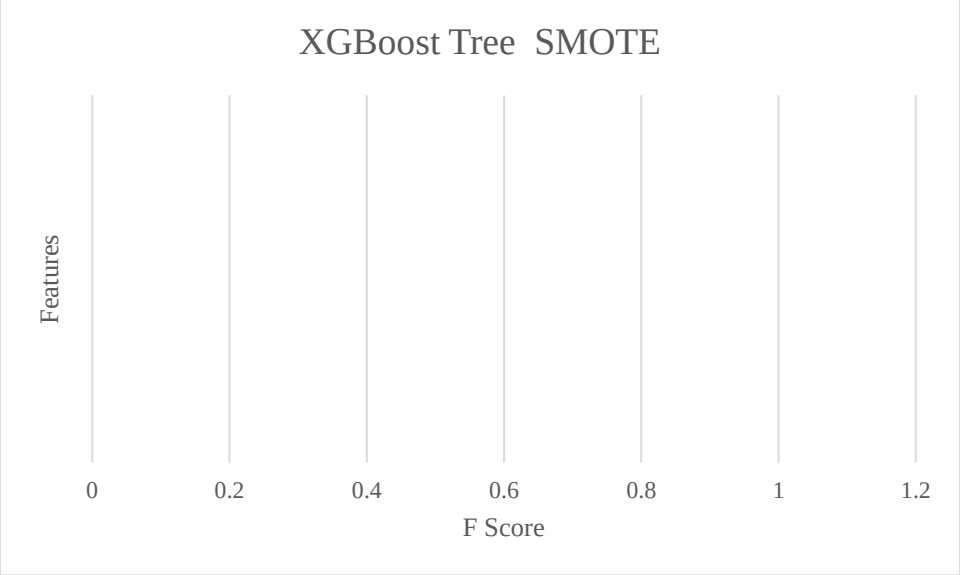

Figure 41 - Predictor importance for XGBoost Tree SMOTE. Source (author)

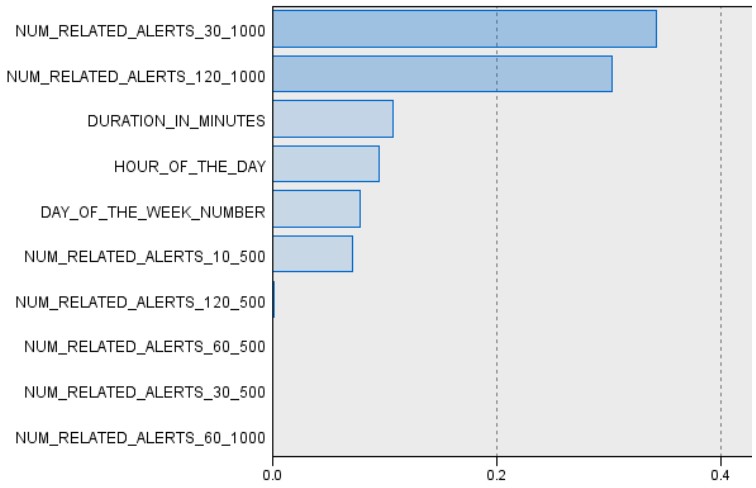

Figure 42 - Predictive importance for Logistic Regression SMOTE. Source (author)

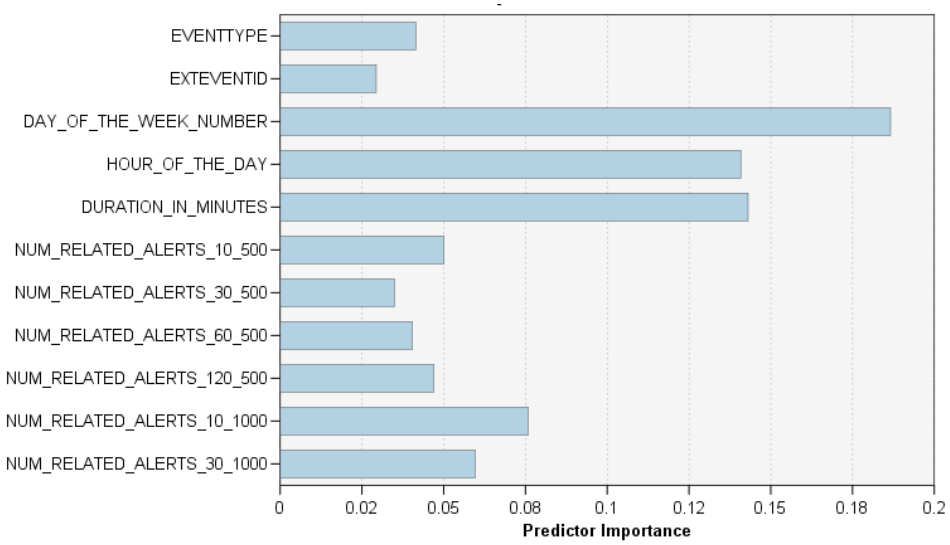

Figure 43 - Predictive importance for Random Trees. Source (author)

8.3 Comparing Raw dataset with Special features

8.3.1 ROC Curves

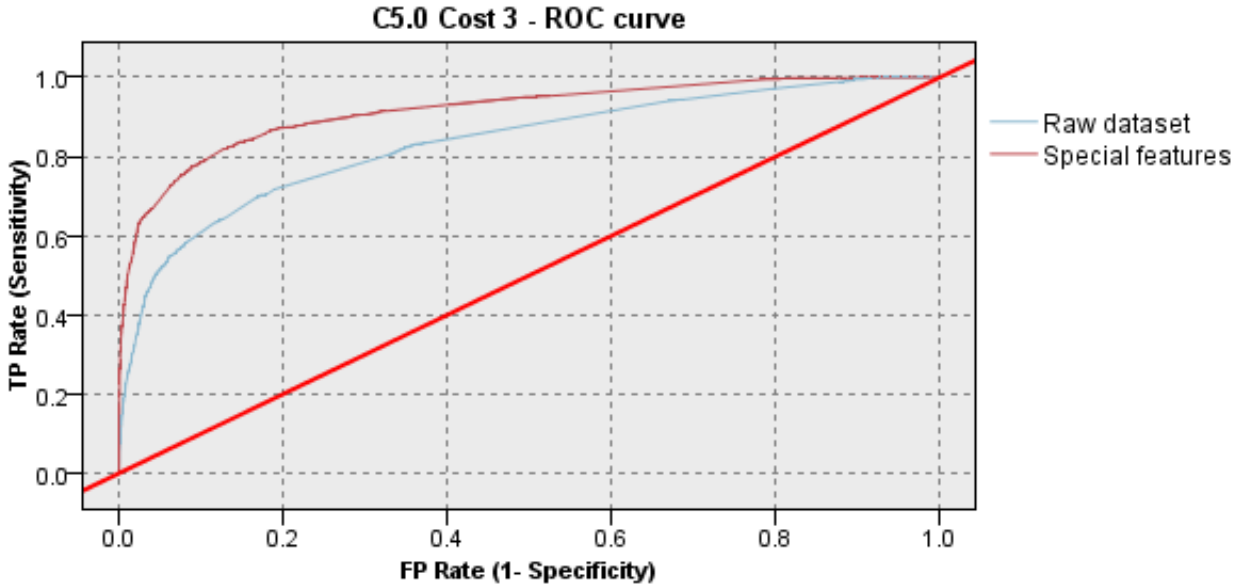

Figure 44 - Raw and Special features ROC curves for C5.0 with a cost x3. Source (author)

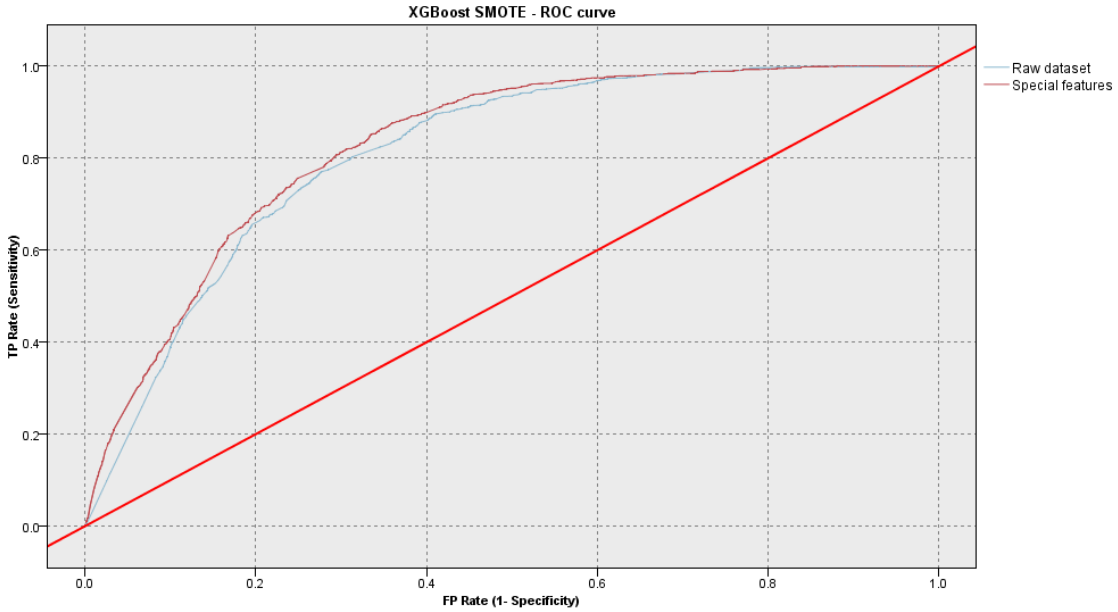

Figure 45 - Raw and Special features ROC curves for XGBoost with SMOTE applied. Source (author)

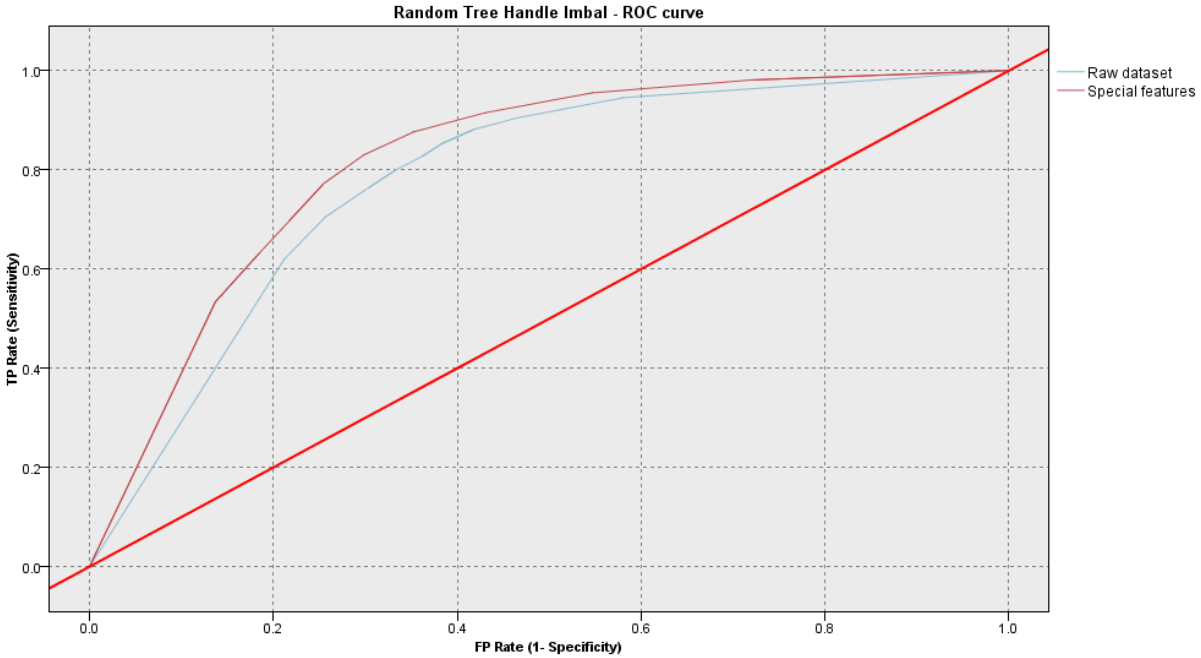

Figure 46 - Raw and Special features ROC curves for Random Tree (handling imbalance) . Source (author)

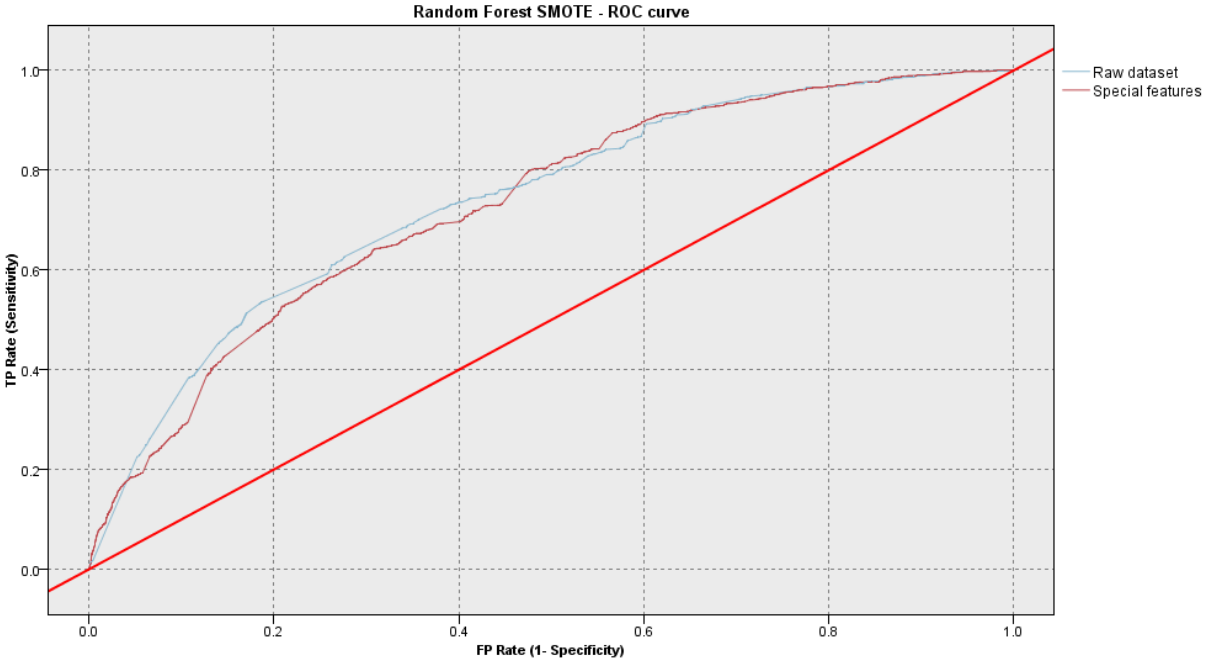

Figure 47 - Raw and Special features ROC curves for Random Forest SMOTE. Source (author)

8.4 Outlier data

Extreme Values

			Case Number	Value
NUM_RELATED_ALERTS_10_500	Highest	1	931	139
		2	932	139
		3	933	139
		4	934	139
		5	833	120[a]
	Lowest	1	248018	0
		2	248017	0
		3	248016	0
		4	248015	0
		5	248014	0[b]
NUM_RELATED_ALERTS_30_500	Highest	1	931	139
		2	932	139
		3	933	139
		4	934	139
		5	833	120[a]
	Lowest	1	248018	0
		2	248017	0
		3	248016	0
		4	248015	0

		5	248014	0[b]
NUM_RELATED_ALERTS_60_500	Highest	1	931	139
		2	932	139
		3	933	139
		4	934	139
		5	833	120[a]
	Lowest	1	248018	0
		2	248017	0
		3	248016	0
		4	248015	0
		5	248014	0[b]
NUM_RELATED_ALERTS_120_500	Highest	1	1138	171
		2	1139	171
		3	1140	171
		4	1141	171
		5	1041	152[c]
	Lowest	1	248018	0
		2	248017	0
		3	248016	0
		4	248015	0
		5	248014	0[b]
NUM_RELATED_ALERTS_10_1000	Highest	1	931	211
		2	932	211
		3	933	211
		4	934	211
		5	833	192[d]
	Lowest	1	248018	0
		2	248017	0
		3	248011	0
		4	248010	0
		5	248008	0[b]
NUM_RELATED_ALERTS_30_1000	Highest	1	931	211
		2	932	211
		3	933	211
		4	934	211
		5	833	192[d]
	Lowest	1	248018	0
		2	248017	0
		3	248011	0
		4	248010	0
		5	248008	0[b]

NUM_RELATED_ALERTS_60_1000	Highest	1	931	211
		2	932	211
		3	933	211
		4	934	211
		5	833	192[d]
	Lowest	1	248018	0
		2	248017	0
		3	248011	0
		4	248010	0
		5	248003	0[b]
NUM_RELATED_ALERTS_120_100	Highest	1	1138	260
0		2	1139	260
		3	1140	260
		4	1141	260
		5	1043	240[e]
	Lowest	1	248018	0
		2	248011	0
		3	248010	0
		4	248003	0
		5	247997	0[b]

a. Only a partial list of cases with the value 120 are shown in the table of upper extremes.

b. Only a partial list of cases with the value 0 are shown in the table of lower extremes.

c. Only a partial list of cases with the value 152 are shown in the table of upper extremes.

d. Only a partial list of cases with the value 192 are shown in the table of upper extremes.

e. Only a partial list of cases with the value 240 are shown in the table of upper extremes.

Table 24 - Extreme values table

8.5 Zombie classifier with minority class

Code to generate random values that are proportional to the binomial target class.

```
package com.thesis;


import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVPrinter;


import java.io.BufferedWriter;
import java.nio.file.Files;
```

```
import java.nio.file.Paths;

import java.util.Random;

import java.util.stream.IntStream;


public class ZombieClassifier {


    public static void main(String[] args) throws Exception {


        final int SAMPLE_SIZE = 49779;

        final double MINORITY_CLASS_PROPORTION = 0.04;

        final int numOfOnes = new Double(SAMPLE_SIZE *
MINORITY_CLASS_PROPORTION).intValue();


        int[] nums = new int[SAMPLE_SIZE];

        Random randomGenerator = new Random();

        IntStream.range(0, numOfOnes)

                .forEach(num -> {

                    int rand = randomGenerator.nextInt(49779);

                    //set a random array value to '1'

                    nums[rand] = 1;

                });

        try (

                BufferedWriter writer =
Files.newBufferedWriter(Paths.get("export.csv"));

                CSVPrinter csvPrinter = new CSVPrinter(writer,
CSVFormat.DEFAULT);

        ) {

            for (int no : nums) {

                csvPrinter.printRecord(no);

            }

            csvPrinter.flush();

        }

    }

}
```

8.6 Top level SQL queries

8.6.1 GET_EVENT_WITH_TARGET

```
CREATE OR REPLACE FUNCTION VDS.GET_EVENT_WITH_TARGET (
        v_number_minutes INTEGER,
        v_distance_in_meters INTEGER
)
        RETURNS TABLE (

ID INTEGER,
SUBJECT VARCHAR(200),
DOMAIN VARCHAR(10),
DESCRIPTION VARCHAR(500),
CATEGORY VARCHAR(100),
EVENTTYPE VARCHAR(100),
EXTEVENTID VARCHAR(200),
EXTWORKEQUIPMENTID VARCHAR(200),
EXTWORKEQUIPMENTTYPE VARCHAR(200),
ASSET_ID INTEGER,
MEASURE_VALUE VARCHAR(200),
MEASURE_TYPE VARCHAR(200),
MEASURE_UNIT VARCHAR(200),
MEASURE_THRESHOLD_VALUE VARCHAR(200),
MEASURE_THRESHOLD INTEGER,
CREATIONTYPE VARCHAR(100),
STATUS VARCHAR(100),
OWNER VARCHAR(100),
CREATEDBY VARCHAR(100),
STARTTS TIMESTAMP,
ENDTS TIMESTAMP,
LASTUPDATEDTS TIMESTAMP,
URGENCY VARCHAR(100),
SEVERITY VARCHAR(100),
CERTAINTY VARCHAR(100),
ACK VARCHAR(3),
MODELID INTEGER,
NETWORK VARCHAR(50),
ADDRESS VARCHAR(500),
ZONE1 VARCHAR(100),
ZONE2 VARCHAR(100),
ZONE3 VARCHAR(100),
CONTRACTID VARCHAR(256),
EVENTSUBTYPE VARCHAR(200),
EVENT_DATE TIMESTAMP,
CASE_DATE TIMESTAMP,
CASE_REFERENCE VARCHAR(100),
ORIGIN_TYPE VARCHAR(100),
ORIGIN_NAME VARCHAR(100),
EVENT_DATA VARCHAR(100),
REMARKS VARCHAR(500),
COSTS VARCHAR(100),
CONSEQUENCES VARCHAR(100),

LOCATION VARCHAR(50),
```

```
WEEK_NUMBER INTEGER,
MONTH_NUMBER INTEGER,
DAY_OF_THE_WEEK_NUMBER INTEGER,
HOUR_OF_THE_DAY INTEGER,

DURATION_IN_MINUTES INTEGER,

WORK_ORDER_ID INTEGER,
ALERT_WAS_PRESENT_BEFORE_WORK_ORDER INTEGER,
WORK_TYPE VARCHAR(128)

)
NO EXTERNAL ACTION
F1: BEGIN ATOMIC

        RETURN
        SELECT E.ID, E.SUBJECT, E.DOMAIN, E.DESCRIPTION, E.CATEGORY, E.EVENTTYPE, E.EXTEVENTID,
            E.EXTWORKEQUIPMENTID, E.EXTWORKEQUIPMENTTYPE, E.ASSET_ID, E.MEASURE_VALUE,
            E.MEASURE_TYPE, E.MEASURE_UNIT, E.MEASURE_THRESHOLD_VALUE, E.MEASURE_THRESHOLD,
            E.CREATIONTYPE, E.STATUS, E.OWNER, E.CREATEDBY, E.STARTTS, E.ENDTS, E.LASTUPDATEDTS,
            E.URGENCY, E.SEVERITY, E.CERTAINTY, E.ACK, E.MODELID, E.NETWORK,
            E.ADDRESS, E.ZONE1, E.ZONE2, E.ZONE3, E.CONTRACTID, E.EVENTSUBTYPE, E.EVENT_DATE,
            E.CASE_DATE, E.CASE_REFERENCE, E.ORIGIN_TYPE, E.ORIGIN_NAME, E.EVENT_DATA, E.REMARKS,
            E.COSTS, E.CONSEQUENCES,

            VARCHAR(db2gse.ST_AsText(db2gse.ST_Centroid(E.LOCATION)), 50) AS LOCATION,

            WEEK(E.STARTTS) WEEK_NUMBER,
            MONTH(E.STARTTS) MONTH_NUMBER,
            DAYOFWEEK(E.STARTTS) DAY_OF_THE_WEEK_NUMBER,
            HOUR(E.STARTTS) HOUR_OF_THE_DAY,


            CASE WHEN
                DUR_E.DURATION_IN_MINUTES IS NOT NULL
            THEN DUR_E.DURATION_IN_MINUTES
                ELSE 0 END AS DURATION_IN_MINUTES,

            WO_E.WO_ID,

            CASE WHEN
                WO_E.WO_ID IS NOT NULL
            THEN 1
            ELSE 0
            END AS ALERT_WAS_PRESENT_BEFORE_WORK_ORDER,


            WO_E.WORK_TYPE

        FROM VDS.SRC_EVENTS_MT E

        LEFT JOIN TABLE(VDS.GET_WORK_ORDER_ALERTS(v_number_minutes, v_distance_in_meters)
                        ) AS WO_E
        ON
            WO_E.EVENT_ID = E.ID

        LEFT JOIN TABLE(VDS.GET_ALERT_DURATION()) AS DUR_E
        ON
```

```
            DUR_E.ID = E.ID


        WHERE STARTTS > '2017-10-01 00:00:00'       -- exclude some test data
                AND EVENTTYPE != 'Pressure Diff'    -- more test data (3 records)
        ;
END F1
```

8.6.2 GET_SPECIAL_FEATURES_DATASET_WITH_TARGET

```
CREATE OR REPLACE FUNCTION VDS.GET_SPECIAL_FEATURES_DATASET_WITH_TARGET (
        v_number_minutes INTEGER,
        v_distance_in_meters INTEGER
)
        RETURNS TABLE (

ID INTEGER,
SUBJECT VARCHAR(200),
DOMAIN VARCHAR(10),
DESCRIPTION VARCHAR(500),
CATEGORY VARCHAR(100),
EVENTTYPE VARCHAR(100),
EXTEVENTID VARCHAR(200),
EXTWORKEQUIPMENTID VARCHAR(200),
EXTWORKEQUIPMENTTYPE VARCHAR(200),
ASSET_ID INTEGER,
MEASURE_VALUE VARCHAR(200),
MEASURE_TYPE VARCHAR(200),
MEASURE_UNIT VARCHAR(200),
MEASURE_THRESHOLD_VALUE VARCHAR(200),
MEASURE_THRESHOLD INTEGER,
CREATIONTYPE VARCHAR(100),
STATUS VARCHAR(100),
OWNER VARCHAR(100),
CREATEDBY VARCHAR(100),
STARTTS TIMESTAMP,
ENDTS TIMESTAMP,
LASTUPDATEDTS TIMESTAMP,
URGENCY VARCHAR(100),
SEVERITY VARCHAR(100),
CERTAINTY VARCHAR(100),
ACK VARCHAR(3),
MODELID INTEGER,
NETWORK VARCHAR(50),
ADDRESS VARCHAR(500),
ZONE1 VARCHAR(100),
ZONE2 VARCHAR(100),
ZONE3 VARCHAR(100),
CONTRACTID VARCHAR(256),
EVENTSUBTYPE VARCHAR(200),
EVENT_DATE TIMESTAMP,
CASE_DATE TIMESTAMP,
CASE_REFERENCE VARCHAR(100),
ORIGIN_TYPE VARCHAR(100),
ORIGIN_NAME VARCHAR(100),
EVENT_DATA VARCHAR(100),
REMARKS VARCHAR(500),
COSTS VARCHAR(100),
```

```
CONSEQUENCES VARCHAR(100),

LOCATION VARCHAR(50),

WEEK_NUMBER INTEGER,
MONTH_NUMBER INTEGER,
DAY_OF_THE_WEEK_NUMBER INTEGER,
HOUR_OF_THE_DAY INTEGER,

DURATION_IN_MINUTES INTEGER,

WORK_ORDER_ID INTEGER,
ALERT_WAS_PRESENT_BEFORE_WORK_ORDER INTEGER,
WORK_TYPE VARCHAR(128),

NUM_RELATED_ALERTS_10_500 INTEGER
,
NUM_RELATED_ALERTS_30_500 INTEGER,
NUM_RELATED_ALERTS_60_500 INTEGER,
NUM_RELATED_ALERTS_120_500 INTEGER,
NUM_RELATED_ALERTS_10_1000 INTEGER,
NUM_RELATED_ALERTS_30_1000 INTEGER,
NUM_RELATED_ALERTS_60_1000 INTEGER,
NUM_RELATED_ALERTS_120_1000 INTEGER



)
NO EXTERNAL ACTION
F1: BEGIN ATOMIC

        RETURN

        SELECT B.*,
                        SF1.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_10_500,
                        SF2.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_30_500,
                        SF3.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_60_500,
                        SF4.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_120_500,
                        SF5.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_10_1000,
                        SF6.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_30_1000,
                        SF7.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_60_1000,
                        SF8.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_120_1000

        FROM TABLE(VDS.GET_EVENT_WITH_TARGET (v_number_minutes, v_distance_in_meters)) B

        LEFT JOIN
                TABLE(VDS.GET_NUM_RELATED_ALERTS (10, 500)) SF1
        ON B.ID = SF1.ID

        LEFT JOIN
            TABLE(VDS.GET_NUM_RELATED_ALERTS (30, 500)) SF2
        ON B.ID = SF2.ID

        LEFT JOIN
            TABLE(VDS.GET_NUM_RELATED_ALERTS (60, 500)) SF3
        ON B.ID = SF3.ID

        LEFT JOIN
```

```
        TABLE(VDS.GET_NUM_RELATED_ALERTS (120, 500)) SF4
    ON B.ID = SF4.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (10, 1000)) SF5
    ON B.ID = SF5.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (30, 1000)) SF6
    ON B.ID = SF6.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (60, 1000)) SF7
    ON B.ID = SF7.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (120, 1000)) SF8
    ON B.ID = SF8.ID ;

END F1
```

8.6.3 GET_SPECIAL_FEATURES_DATASET_WITH_TARGET

```
CREATE OR REPLACE FUNCTION VDS. GET_SPECIAL_FEATURES_DATASET_WITH_TARGET (
        v_event_id INTEGER
)

        RETURNS TABLE (

ID INTEGER,
SUBJECT VARCHAR(200),
DOMAIN VARCHAR(10),
DESCRIPTION VARCHAR(500),
CATEGORY VARCHAR(100),
EVENTTYPE VARCHAR(100),
EXTEVENTID VARCHAR(200),
EXTWORKEQUIPMENTID VARCHAR(200),
EXTWORKEQUIPMENTTYPE VARCHAR(200),
ASSET_ID INTEGER,
MEASURE_VALUE VARCHAR(200),
MEASURE_TYPE VARCHAR(200),
MEASURE_UNIT VARCHAR(200),
MEASURE_THRESHOLD_VALUE VARCHAR(200),
MEASURE_THRESHOLD INTEGER,
CREATIONTYPE VARCHAR(100),
STATUS VARCHAR(100),
OWNER VARCHAR(100),
CREATEDBY VARCHAR(100),
STARTTS TIMESTAMP,
ENDTS TIMESTAMP,
LASTUPDATEDTS TIMESTAMP,
URGENCY VARCHAR(100),
SEVERITY VARCHAR(100),
CERTAINTY VARCHAR(100),
ACK VARCHAR(3),
MODELID INTEGER,
NETWORK VARCHAR(50),
ADDRESS VARCHAR(500),
```

```
ZONE1 VARCHAR(100),
ZONE2 VARCHAR(100),
ZONE3 VARCHAR(100),
CONTRACTID VARCHAR(256),
EVENTSUBTYPE VARCHAR(200),
EVENT_DATE TIMESTAMP,
CASE_DATE TIMESTAMP,
CASE_REFERENCE VARCHAR(100),
ORIGIN_TYPE VARCHAR(100),
ORIGIN_NAME VARCHAR(100),
EVENT_DATA VARCHAR(100),
REMARKS VARCHAR(500),
COSTS VARCHAR(100),
CONSEQUENCES VARCHAR(100),

LOCATION VARCHAR(50),

WEEK_NUMBER INTEGER,
MONTH_NUMBER INTEGER,
DAY_OF_THE_WEEK_NUMBER INTEGER,
HOUR_OF_THE_DAY INTEGER,

DURATION_IN_MINUTES INTEGER,

NUM_RELATED_ALERTS_10_500 INTEGER,
NUM_RELATED_ALERTS_30_500 INTEGER,
NUM_RELATED_ALERTS_60_500 INTEGER,
NUM_RELATED_ALERTS_120_500 INTEGER,
NUM_RELATED_ALERTS_10_1000 INTEGER,
NUM_RELATED_ALERTS_30_1000 INTEGER,
NUM_RELATED_ALERTS_60_1000 INTEGER,
NUM_RELATED_ALERTS_120_1000 INTEGER



)
        NO EXTERNAL ACTION
F1: BEGIN ATOMIC

        RETURN

        SELECT E.*,
                SF1.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_10_500,
                SF2.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_30_500,
                SF3.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_60_500,
                SF4.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_120_500,
                SF5.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_10_1000,
                SF6.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_30_1000,
                SF7.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_60_1000,
                SF8.NUM_RELATED_ALERTS NUM_RELATED_ALERTS_120_1000

        FROM VDS.SRC_EVENTS_MT E

        LEFT JOIN
                TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 10, 500)) SF1
        ON B.ID = SF1.ID

        LEFT JOIN
```

```
        TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 30, 500)) SF2
    ON B.ID = SF2.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 60, 500)) SF3
    ON B.ID = SF3.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 120, 500)) SF4
    ON B.ID = SF4.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 10, 1000)) SF5
    ON B.ID = SF5.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 30, 1000)) SF6
    ON B.ID = SF6.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 60, 1000)) SF7
    ON B.ID = SF7.ID

    LEFT JOIN
        TABLE(VDS.GET_NUM_RELATED_ALERTS (v_event_id, 120, 1000)) SF8
    ON B.ID = SF8.ID

WHERE E.ID = v_event_id;

END
```

8.7 Lower level functions

8.7.1 GET_WORK_ORDER_ALERTS

```
CREATE OR REPLACE FUNCTION VDS.GET_WORK_ORDER_ALERTS (
        v_number_minutes INTEGER,
        v_distance_in_meters INTEGER
)
  RETURNS TABLE (
        WO_ID INTEGER,
        EVENT_ID INTEGER,
        WORK_TYPE VARCHAR(128)
  )
NO EXTERNAL ACTION
F1: BEGIN ATOMIC

  RETURN

        SELECT E_WO.WO_ID, E_WO.ID, UPPER(WO2.WORK_TYPE)
        FROM
  (
    SELECT MAX(WO.WO_ID) WO_ID, E.ID

    FROM
    EAM.WORK_ORDER WO,
```

```
    VDS.SRC_EVENTS_MT E,
    TABLE(
      SELECT db2gse.ST_GeomFromText(LOCATION, 1003) LOCATION
      FROM
        TABLE(
          SELECT VARCHAR(db2gse.ST_AsText(db2gse.ST_Centroid(LOCATION)), 50)  LOCATION
          FROM VDS.SRC_EVENTS_MT
          GROUP BY VARCHAR(db2gse.ST_AsText(db2gse.ST_Centroid(LOCATION)), 50)
        )
      ) AS E_LOC

  WHERE
    WO.CREATION_DATETIME IS NOT NULL AND

    db2gse.ST_Equals(E.LOCATION, E_LOC.LOCATION) = 1  AND

    WO.CREATION_DATETIME > E.STARTTS AND

    TIMESTAMPDIFF(4, CHAR                -- minutes
        (WO.CREATION_DATETIME - E.STARTTS)
    ) < v_number_minutes AND

    db2gse.ST_Intersects(WO.LOCATION,
              db2gse.ST_Buffer(E_LOC.LOCATION, v_distance_in_meters, 'METER')) = 1
            SELECTIVITY 0.00001 AND

    WO.WORK_TYPE IN ('USTERKA', 'AWARIA', 'awaria')

  GROUP BY E.ID
  )
  E_WO,
  EAM.WORK_ORDER WO2

WHERE
  E_WO.WO_ID = WO2.WO_ID;

END F1
```

8.7.2 GET_WORK_ORDER_ALERTS

```
CREATE OR REPLACE FUNCTION VDS.GET_WORK_ORDER_ALERTS (
          v_number_minutes INTEGER,
          v_distance_in_meters INTEGER
)
          RETURNS TABLE (WO_ID INTEGER, EVENT_ID INTEGER, WORK_TYPE VARCHAR(128))
          NO EXTERNAL ACTION
F1: BEGIN ATOMIC

          RETURN

SELECT E_WO.WO_ID, E_WO.ID, UPPER(WO2.WORK_TYPE)
FROM
   (
  SELECT MAX(WO.WO_ID) WO_ID, E.ID

  FROM
```

```
      EAM.WORK_ORDER WO,
      VDS.SRC_EVENTS_MT E,

      TABLE(
        SELECT db2gse.ST_GeomFromText(LOCATION, 1003) LOCATION
        FROM
          TABLE(
            SELECT VARCHAR(db2gse.ST_AsText(db2gse.ST_Centroid(LOCATION)), 50)  LOCATION
            FROM VDS.SRC_EVENTS_MT
            GROUP BY VARCHAR(db2gse.ST_AsText(db2gse.ST_Centroid(LOCATION)), 50)
          )
        ) AS E_LOC

    WHERE
      WO.CREATION_DATETIME IS NOT NULL AND

      db2gse.ST_Equals(E.LOCATION, E_LOC.LOCATION) = 1  AND

      WO.CREATION_DATETIME > E.STARTTS AND

      TIMESTAMPDIFF(4, CHAR                  -- minutes
          (WO.CREATION_DATETIME - E.STARTTS)
      ) < v_number_minutes AND

      db2gse.ST_Intersects(WO.LOCATION,
                  db2gse.ST_Buffer(E_LOC.LOCATION, v_distance_in_meters, 'METER')) = 1
                SELECTIVITY 0.00001 AND

      WO.WORK_TYPE IN ('USTERKA', 'AWARIA', 'awaria')

    GROUP BY E.ID
    )
    E_WO,
    EAM.WORK_ORDER WO2

WHERE
    E_WO.WO_ID = WO2.WO_ID
;

END F1
```

8.7.3 GET_NUM_RELATED_ALERTS

```
CREATE OR REPLACE FUNCTION VDS.GET_NUM_RELATED_ALERTS (
            v_number_minutes INTEGER,
            v_distance_in_meters INTEGER
)
            RETURNS TABLE (
                        ID INTEGER,
                        NUM_RELATED_ALERTS VARCHAR(128)
                        )
            NO EXTERNAL ACTION
F1: BEGIN ATOMIC

RETURN
```

```
SELECT
   E.ID,
   COALESCE(HIT.NUM_RELATED_ALERTS, 0) NUM_RELATED_ALERTS

FROM VDS.SRC_EVENTS_MT E

LEFT JOIN
   (

      SELECT E1_ID, COUNT(E2_ID) NUM_RELATED_ALERTS
      FROM
         (
         SELECT E1.ID E1_ID,
               E2.ID E2_ID,
               E1.STARTTS E1_STARTTS,
               E1.ENDTS E1_ENDTS,
               E2.STARTTS E2_STARTTS,
               E2.ENDTS E2_ENDTS,
               E2.SUBJECT E2_SUBJECT,
               E1.LOCATION_WKT E1_LOCATION_WKT,
               E2.LOCATION_WKT E2_LOCATION_WKT,
               LOC_MAP.METERS_BETWEEN

         FROM (
               SELECT *
               FROM VDS.SRC_EVENTS_MT
               WHERE STARTTS > '2017-10-01 00:00:00' AND
                  EVENTTYPE != 'Presure Diff'
               --FETCH FIRST 150 ROWS ONLY
            ) E1,
            (
               SELECT *
               FROM VDS.SRC_EVENTS_MT
               WHERE STARTTS > '2017-10-01 00:00:00' AND
                  EVENTTYPE != 'Presure Diff'
               --FETCH FIRST 150 ROWS ONLY
            ) E2,
         VDS.EVENT_LOCATION_DISTANCES LOC_MAP


         WHERE

            E1.ID != E2.ID AND

            E1.LOCATION_WKT != E2.LOCATION_WKT AND

            E1.STARTTS > E2.STARTTS AND
            E1.STARTTS < E2.ENDTS + 1 DAYS AND
            E1.STARTTS < E2.ENDTS + v_number_minutes MINUTES AND

            E1.LOCATION_WKT = LOC_MAP.LOC_1_WKT AND
            E2.LOCATION_WKT = LOC_MAP.LOC_2_WKT AND

            LOC_MAP.METERS_BETWEEN  <= v_distance_in_meters
            )

            GROUP BY E1_ID
   ) HIT
```

```
   ON E.ID = HIT.E1_ID

   WHERE STARTTS > '2017-10-01 00:00:00'  -- exclude some test data
      AND EVENTTYPE != 'Pressure Diff'    -- more test data (3 records)
      ;

END F1
```

8.7.4 GET_NUM_RELATED_ALERTS (single alert)

```
CREATE OR REPLACE FUNCTION VDS.GET_NUM_RELATED_ALERTS (
          v_event_id INTEGER,
          v_number_minutes INTEGER,
          v_distance_in_meters INTEGER
)
          RETURNS TABLE (
                    ID INTEGER,
                    NUM_RELATED_ALERTS VARCHAR(128)
                    )
          NO EXTERNAL ACTION
F1: BEGIN ATOMIC

RETURN

   SELECT
      E.ID,
      COALESCE(HIT.NUM_RELATED_ALERTS, 0) NUM_RELATED_ALERTS

   FROM VDS.SRC_EVENTS_MT E

   LEFT JOIN
      (

         SELECT E1_ID, COUNT(E2_ID) NUM_RELATED_ALERTS
         FROM
            (
            SELECT E1.ID E1_ID,
                  E2.ID E2_ID,
                  E1.STARTTS E1_STARTTS,
                  E1.ENDTS E1_ENDTS,
                  E2.STARTTS E2_STARTTS,
                  E2.ENDTS E2_ENDTS,
                  E2.SUBJECT E2_SUBJECT,
                  E1.LOCATION_WKT E1_LOCATION_WKT,
                  E2.LOCATION_WKT E2_LOCATION_WKT,
                  LOC_MAP.METERS_BETWEEN

            FROM (
                  SELECT *
                  FROM VDS.SRC_EVENTS_MT
                  WHERE ID = v_event_id
               ) E1,
               (
                  SELECT *
                  FROM VDS.SRC_EVENTS_MT
                  WHERE STARTTS > '2017-10-01 00:00:00' AND
                     EVENTTYPE != 'Presure Diff'
```

```
            --FETCH FIRST 150 ROWS ONLY
          ) E2,
        VDS.EVENT_LOCATION_DISTANCES LOC_MAP


        WHERE

          E1.ID != E2.ID AND

          E1.LOCATION_WKT != E2.LOCATION_WKT AND

          E1.STARTTS > E2.STARTTS AND
          E1.STARTTS < E2.ENDTS + 1 DAYS AND
          E1.STARTTS < E2.ENDTS + v_number_minutes MINUTES AND

          E1.LOCATION_WKT = LOC_MAP.LOC_1_WKT AND
          E2.LOCATION_WKT = LOC_MAP.LOC_2_WKT AND

          LOC_MAP.METERS_BETWEEN  <= v_distance_in_meters
          )

          GROUP BY E1_ID
    ) HIT

  ON E.ID = HIT.E1_ID

  WHERE ID = v_event_id;

END F1
```

8.7.5 GET_DISTANCE_BETWEEN_ALL_ALERT_LOCATIONS

```
CREATE OR REPLACE FUNCTION VDS.GET_DISTANCE_BETWEEN_ALL_ALERT_LOCATIONS ()
   RETURNS TABLE (
        METERS_BETWEEN DOUBLE,
        LOCATION_1 db2gse.ST_GEOMETRY,
        LOCATION_2 db2gse.ST_GEOMETRY,
        LOC_1_WKT VARCHAR(50),
        LOC_2_WKT VARCHAR(50)
  )
NO EXTERNAL ACTION
F1: BEGIN ATOMIC


        RETURN
        SELECT
            db2gse.ST_Distance(LOC_1.LOCATION, LOC_2.LOCATION, 'METER') AS METERS_BETWEEN,
            LOC_1.LOCATION LOCATION_1,
            LOC_2.LOCATION LOCATION_2,
            LOC_1.LOCATION_WKT LOC_1_WKT,
            LOC_2.LOCATION_WKT LOC_2_WKT

        FROM
            VDS.EVENT_LOCATION LOC_1,
            VDS.EVENT_LOCATION LOC_2;

END
```

8.7.6 GET_ALERT_DURATION

```
CREATE OR REPLACE FUNCTION VDS.GET_ALERT_DURATION ()
        RETURNS TABLE (
                        ID INTEGER,
                        DURATION_IN_MINUTES INTEGER)
        NO EXTERNAL ACTION
F1: BEGIN ATOMIC


        RETURN

        SELECT E1_ID, SUM(BD.DURATION_IN_MINUTES)


        FROM
                (

                SELECT
                        E1.ID E1_ID,
                        TIMESTAMPDIFF(4, CHAR(E2.ENDTS - E2.STARTTS)) "DURATION_IN_MINUTES"

                FROM (
                                SELECT *
                                FROM VDS.SRC_EVENTS_MT
                                WHERE STARTTS > '2017-10-01 00:00:00' AND
                                        EVENTTYPE != 'Presure Diff'
                        ) E1,
                   (
                        SELECT *
                        FROM VDS.SRC_EVENTS_MT
                        WHERE STARTTS > '2017-10-01 00:00:00' AND
                                        EVENTTYPE != 'Presure Diff'
                        ) E2


                WHERE
                        E1.ID != E2.ID AND

                        E1.STARTTS > E2.STARTTS AND
                        E1.STARTTS < E2.ENDTS + 1 DAY AND
                        E1.STARTTS < E2.ENDTS + 120 MINUTES AND

                        E1.LOCATION_WKT = E2.LOCATION_WKT


                ) BD

        GROUP BY E1_ID;
END F1
```

8.8 Tables to cache recurrent calculations

8.8.1 EVENT_LOCATION_DISTANCES

```
CREATE TABLE VDS.EVENT_LOCATION_DISTANCES AS (
    SELECT *
    FROM TABLE(VDS.GET_DISTANCE_BETWEEN_ALL_ALERT_LOCATIONS())
) WITH NO DATA;

INSERT INTO VDS.EVENT_LOCATION_DISTANCES (SELECT *
    FROM TABLE(VDS.GET_DISTANCE_BETWEEN_ALL_ALERT_LOCATIONS ()));

-- Add spatial index to increase spatial query times
-- Using a spatial clusters that are tuned to point geometries
CREATE INDEX
VDS.EVENT_LOCATION_1_DISTANCES_IDX
ON VDS.EVENT_LOCATION_DISTANCES(LOCATION_1)
EXTEND USING DB2GSE.SPATIAL_INDEX(0.0021, 0.011, 0.044) ;
CREATE INDEX
VDS.EVENT_LOCATION_2_DISTANCES_IDX
ON VDS.EVENT_LOCATION_DISTANCES(LOCATION_2)
EXTEND USING DB2GSE.SPATIAL_INDEX(0.0021, 0.011, 0.044) ;

-- Add indices to the location distances table to improve query performance
CREATE INDEX VDS.EVENT_LOCATION_DISTANCES_LOC_1_WKT_X
ON VDS.EVENT_LOCATION_DISTANCES ( LOC_1_WKT )
ALLOW REVERSE SCANS PAGE SPLIT SYMMETRIC COLLECT SAMPLED DETAILED STATISTICS
COMPRESS NO INCLUDE NULL KEYS;

CREATE INDEX VDS.EVENT_LOCATION_DISTANCES_LOC_2_WKT_X
ON VDS.EVENT_LOCATION_DISTANCES ( LOC_2_WKT )
ALLOW REVERSE SCANS PAGE SPLIT SYMMETRIC COLLECT SAMPLED DETAILED STATISTICS
COMPRESS NO INCLUDE NULL KEYS;

CREATE INDEX VDS.EVENT_LOCATION_DISTANCES_MB_IDX
ON VDS.EVENT_LOCATION_DISTANCES ( METERS_BETWEEN )
ALLOW REVERSE SCANS PAGE SPLIT SYMMETRIC COLLECT SAMPLED DETAILED STATISTICS
COMPRESS NO INCLUDE NULL KEYS;
```

8.8.2 EVENT_LOCATION

```
CREATE TABLE VDS.EVENT_LOCATION AS (
    SELECT *
    FROM TABLE(VDS.GET_ALL_ALERT_LOCATIONS ())
) WITH NO DATA;

INSERT INTO VDS.EVENT_LOCATION (
      SELECT *
      FROM TABLE(VDS.GET_ALL_ALERT_LOCATIONS ())
);

CREATE INDEX
VDS.EVENT_LOCATION_IDX
ON VDS.EVENT_LOCATION(LOCATION)
EXTEND USING DB2GSE.SPATIAL_INDEX(0.0021, 0.011, 0.044) ;
```

8.9 Modifications to existing system DB tables

8.9.1 SRC_EVENTS_MT

```
-- Add a Well-Known-Text (WKT) version of the LOCATION field for certain GROUP BY
operations
ALTER TABLE VDS.SRC_EVENTS_MT ADD COLUMN LOCATION_WKT VARCHAR(50 OCTETS);
UPDATE VDS.SRC_EVENTS_MT SET LOCATION_WKT = db2gse.ST_AsText(LOCATION);

CREATE INDEX VDS.SRC_EVENTS_MT_LOC_WKT_X ON VDS.SRC_EVENTS_MT ( LOCATION_WKT )
ALLOW REVERSE SCANS PAGE SPLIT SYMMETRIC COLLECT SAMPLED DETAILED STATISTICS
COMPRESS NO INCLUDE NULL KEYS;
```

8.9.2 WORK_ORDER

```
-- Spatial index optimised for point geometries
CREATE INDEX
EAM.WORK_ORDER_LOCATION_IDX
ON EAM.WORK_ORDER(LOCATION)
EXTEND USING DB2GSE.SPATIAL_INDEX(0.0021, 0.011, 0.044);
```

8.9.2.1 Fix for missing CREATION_DATE time value

```
-- add the new column to be the definitive datetime value

ALTER TABLE EAM.WORK_ORDER ADD COLUMN CREATION_DATETIME TIMESTAMP;

CREATE INDEX EAM.WORK_ORDER_CDT_IDX
       ON EAM.WORK_ORDER ( CREATION_DATETIME )
       ALLOW REVERSE SCANS PAGE SPLIT SYMMETRIC
       COLLECT SAMPLED DETAILED STATISTICS
       COMPRESS NO
       INCLUDE NULL KEYS;

-- update the new column

UPDATE EAM.WORK_ORDER WO

SET CREATION_DATETIME =

CASE WHEN (
              TIME(CREATION_DATE) != '00:00:00' AND
              ACTUAL_START_DATE IS NOT NULL AND
              ACTUAL_START_DATE < CREATION_DATE
              )
              THEN  ACTUAL_START_DATE    -- actual_start is before creation date
       WHEN TIME(CREATION_DATE) != '00:00:00'
              THEN CREATION_DATE         -- creation date is complete
       WHEN (
                     ACTUAL_START_DATE IS NOT NULL AND
                     DATE(CREATION_DATE) =  DATE(ACTUAL_START_DATE) AND
                     TIME(ACTUAL_START_DATE) != '00:00:00'
                )
              THEN ACTUAL_START_DATE     -- use actual start date as creation date
                                         -- is missing
                                         -- the time and date is the same for both
       ELSE
```

```
            UPDATEDATE    -- if updated date is the same as the creation date use
                          -- it
END
WHERE
      CREATION_DATE IS NOT NULL AND
      UPPER(WO.WORK_TYPE) IN ('USTERKA', 'AWARIA') AND
      (
            (UPDATEDATE IS NOT NULL AND DATE(UPDATEDATE) = DATE(CREATION_DATE))
OR
            (
            ACTUAL_START_DATE IS NOT NULL AND DATE(CREATION_DATE) =
            DATE(ACTUAL_START_DATE) AND TIME(ACTUAL_START_DATE) != '00:00:00')
OR
            TIME(CREATION_DATE) != '00:00:00'
      )
;
```