POLITECNICO DI TORINO

Master of Science in Computer Engineering

Master's Degree Thesis

From Cluster Distributions Through Kernel Density Estimate to Driving Behaviour Scores: A Complete Data Science Pipeline

Supervisors:
prof. Luca Cagliero

Co-Supervisor:
prof. Elena Baralis
dott. Giuseppe Attanasio

Candidate:
Lorenzo Vaiani

Academic year 2020 – 2021

Abstract

The study of driving behaviour is an element of great interest for all those companies that provide a fleet management service. Careful observations allow us to find imperfections that, if corrected, entail a reduction of risks and an increase in profits. In order to objectively evaluate this type of behaviour, there are countless indicators to be taken into consideration. In this work, the most significant indicators among those available are used to produce an evaluation of the driving behaviour through a pipeline of operations related to the world of data science.

This thesis focuses on the last stage of the process. Machine learning techniques are used to explore a cluster distribution resulting from previous stages. Then that distribution is exploited to calculate a final Key Performance Indicator (KPI) for each trip to be evaluated. This is not done through conventional techniques, such as comparison with a reference approved by a domain expert, but through the application of a Kernel Density Estimation (KDE) function that allows us to produce an assessment based on proximity to the majority of similar behaviours.

The results of the experiments demonstrated the validity of the automatically selected set to be considered as a reference, thus confirming the reliability of the KPIs. Furthermore, the analyses on the various types of features have highlighted which are the indicators that most influence the evaluation of driving behaviour.

Contents

List of Tables 7

List of Figures 8

1 Introduction 11

2 Theoretical Premises 15
2.1 Data Mining e Machine Learning Techniques 15
2.1.1 Classification 16
2.1.2 Methods for Model Validation 20
2.1.3 Metrics for Performance Evaluation 22
2.1.4 Resampling 23
2.2 Kernel Density Estimation (KDE) 25

3 Case Study 27
3.1 Fleet Management 27
3.1.1 Driving Behaviour 28
3.2 Data Description 29
3.2.1 Data Provided 30
3.2.2 Data Enrichment 31

4 Related Works 33

5 Framework Description 37
5.1 Terminology 37
5.2 Pipeline Structure 38
5.3 Key Ideas and Methods 39
5.3.1 Data Distribution Analysis 40
5.3.2 Exploration of Clustering Outcomes 41
5.3.3 KPI Computation 42

5.3.4 Penalty Score 45

6 Experiments 47
6.1 Route Identification 49
6.2 Supervised Analysis 51
6.3 Driving Behaviour Evaluation 53

7 Results 55
7.1 Classification Results 55
7.2 KPI Computation Results 64

8 Conclusions and Future Works 71

List of Tables

4.1 Related works overview, indicating for each the presence of GPS features, CAN features and context features and the type of approach with which the data were analyzed. 36
5.1 Statistics about CAN bus messages, in details we have attendance % in all messages, attendance % in CAN messages and average attendance % inside a vehicle with CAN messages. 40
7.1 Results obtained for route R3 considering the context features and not rebalancing the data. 56
7.2 Results obtained for route R3 considering the context features and rebalancing the data. 56
7.3 Results obtained for route R3 not considering the context features and not rebalancing the data. 59
7.4 Results obtained for route R3 not considering the context features and rebalancing the data. 59
7.5 Comparison between the KPIs of *Connected* trips that include and not include the CAN features, always considering context feature. . 70
7.6 Comparison between the KPIs of *Connected* trips that include and not include the CAN features, always not considering context feature. 70

List of Figures

2.1 Comparison between a histogram (left) and kernel density estimation (right) constructed using the same data. 26
2.2 Different bandwidth comparison. 26
5.1 Pipeline structure. 38
5.2 2D example of KDE applied on data points. 42
5.3 New data points to be evaluate are added. 43
5.4 Focus on the cluster medoid. Its position and score are highlighted. 43
5.5 Comparison between medoid and trip to be evaluated and their scores after the normalization step. 44
7.1 Permutation importance of the Random Forest classifier for the route R3 considering the context features. 57
7.2 Decision Tree graphical representation built based on data of the route R3 and considering the context features. 58
7.3 Permutation importance of the Random Forest classifier for the route R3 not considering the context features. 60
7.4 Decision Tree graphical representation built based on data of the route R3 and not considering the context features. 61
7.5 Violin charts showing how the values of the context features are distributed within each cluster. 62
7.6 Heatmap showing Pearson's correlation between context and vehicle features. 63
7.7 Heatmap that represents KPI values for each *Connected* trip and the penalty score of each feature, obtained considering context features. 66
7.8 Heatmap that represents KPI values for each *Business* trip and the penalty score of each feature, obtained considering context features. 67
7.9 Heatmap that represents KPI values for each *Connected* trip and the penalty score of each feature, obtained without considering context features. 68

7.10 Heatmap that represents KPI values for each *Business* trip and the penalty score of each feature, obtained without considering context features. 69

Chapter 1

Introduction

Logistics can be defined as the set of activities aimed at transporting products from one place to another on schedule, efficiently and at the lowest possible cost [1]. It is the cornerstone of the economy, from the supply chains of small companies to national and international ones. It is also the meeting point between the world of manufacturing and the final consumer. In particular, land logistics is the most widely used transport system both in Italy and in Europe for various reasons, including the type of territory, reliability, economic advantages, traceability and the possibility of establishing door-to-door connections. Those who deal with logistics and freight transport with professionalism are aware that one of the most important aspects concerns the management of their company fleet.

Fleet management is a category of operations useful for coordinating and monitoring a set of vehicles [2]. Many trucking companies use fleet management, for example, to optimize their movements to save time and increase profits. Among the various functions of fleet management, we find the evaluation of driving behaviour. Nowadays, the interest in this operation is constantly increasing because by monitoring their drivers, companies acquire knowledge that will allow them to reduce the risks and to better safeguard the environment by suggesting a more ecological driving style. This type of monitoring is based not only on information about the actions of the driver himself but also on information relating to the state of the vehicle and the environment in which the driving takes place. The evaluation of driving behaviour is a service that is currently offered by hundreds of companies and as many computer tools.

K-Master [3] is one of the many companies that deal with fleet management and which among all interests also has that of studying driving behaviour and exploiting the knowledge acquired to provide better services to its customers. This company

constantly collects data from the tens of thousands of vehicles monitored. Given the large amount of data in this search field, many of the analysis tools make extensive use of data mining techniques and artificial intelligence algorithms to achieve the goal. To properly assess driving behaviour, it is also necessary to know what is right and what is wrong to do while driving a vehicle in a certain environment. For this reason, a classic approach to the problem is to use references provided by a domain expert on which to base the entire assessment. K-Master is interested in exploring alternative approaches, also to stand out from the competition. To respond to this request, we have decided to use a machine learning-based approach. Furthermore, we look for a data-driven approach that departs from the strategy mentioned above and tries to supplant the involvement of the domain expert by basing the evaluation on how and how many other drivers have previously behaved in a situation similar to the one under analysis.

The result of this work is a framework that fully guides the analyses necessary for the study of driving behaviour, starting from the raw data held by the company up to the generation of an evaluation that can be easily understood and interpreted by any customers. During the process, the data is heavily pre-processed through filterings, enrichments and aggregations phases. They are then divided into subgroups on which to perform more specific analyses. Among these, we find the use of unsupervised learning techniques and statistical functions that lead us to obtain the desired evaluation. This thesis describes in detail the final part of the framework, performing external studies to it to explore and discover the most relevant characteristics related to the data that have completed the previous pre-processing phases. Furthermore, the technique used to produce the final output is also described, explaining the reasons and interpreting the results.

All this work leads to interesting discoveries regarding the study of driving behaviour. We defined a smart solution to identify which trips should be considered as a reference for future evaluations. We have also demonstrated the effectiveness of some classifiers in recognizing these types of trips by solving typical machine learning problems, such as that of class unbalancing. By doing this we came into contact with the huge amount of indicators that can describe driving behaviour and we understood which of them are the most relevant. In particular, we made a clear distinction between indicators relating to the vehicle and its driver and indicators relating to the context in which the journey takes place, observing how the results change as these descriptors change and discovering how the considered indicators are related to each other. To provide an evaluation, a density estimation function was used, capable of providing a simple numerical value that measures the correctness of each trip, based on the previously selected reference set. The scores

thus provided are easily justifiable, providing reasons based on the distance between trips in the multidimensional space of their features. Eventually, observing the final output results, it is possible to demonstrate the superior effectiveness of the highest level services provided by the company, thus discovering further information useful for commercial purposes.

This thesis consists of several chapters, the purpose of which is described below. Chapter 2 provides an overview of the theoretical notions related to the world of machine learning, data mining and statistics necessary to understand what follows. Chapter 3 deepens our case study, taking up the concepts of fleet management and driving behaviour until we get to talk about our specific situation, focusing on the data available to us. Chapter 4 explores what is already present in the literature, focusing on those concepts that inspired us and that helped guide the development of the work. Chapter 5 briefly describes the framework created and sets out the main reasons that justify the solution adopted for calculating the final KPI. Chapter 6 summarizes the setup of the experiments, indicating all those variables that have been adjusted and analyzed to launch the simulations. Chapter 7 reports the results obtained both from the analyses outside the framework and from the assessments of driving behaviour, confirming the validity of the work carried out. Chapter 8 highlights which were the most interesting discoveries both from a scientific and commercial point of view and suggests possible developments that could bring improvements to what has already been done.

Chapter 2

Theoretical Premises

Fleet management is an activity that hundreds of companies around the world deal with. Some of them manage tens of thousands of vehicles scattered over a vast territory and receive millions of geolocated data transmissions every day. Wherever there is a sufficient amount of information, and, in particular, where there is an enormous quantity of it as in the environment just described, machine learning can be exploited to bring benefits to those who request its use. Consequently, it is reasonable to think that artificial intelligence can help in the resolution of most of the fleet management operations and indeed various machine learning techniques have been used in this work to try to reach the prefixed goal. This chapter aims to describe the machine learning techniques and algorithms used to complete the various steps necessary to achieve the final goal.

2.1 Data Mining e Machine Learning Techniques

Over the years, machine learning techniques have been tested from various points of view and the research of where and how to best use them is continuing. This growth is fueled by the constant increase in the production of data in every possible context, thanks to technologies that have recently taken a turn, such as Internet of Things (IoT) and smartphones. Machine learning can be defined as the automatic acquisition of new knowledge through experience. Machine learning algorithms exploit a set of initial data, called train dataset, for the training of models capable of autonomously making predictions on new data based on what has been assimilated during their construction. Data mining [4] exploits machine learning algorithms to automatically extract information from large data sets whose acquisition would be too complex or time-consuming if performed manually. The aim is to visualize

some kind of pattern or correlation that would be difficult to discover without the use of these techniques.

Machine learning techniques can be divided into two macro-categories: supervised and unsupervised techniques. The former is trained using labeled samples and it build models whose predictions are based on the input data-label correspondences. The latter categorize unlabeled input data based on their common characteristics.

2.1.1 Classification

The most common supervised learning task is *classification* [5]. It consists of learning the structure of a data set which samples are already divided into groups called *classes*. This is done through the training of a model that can then be used to estimate the class of one or more previously unseen data examples with unknown class labels. In the majority of cases, each class label has a precise meaning within the context in which it is used but a classifier does not need to know it in order to perform the classification task correctly. Classification is a supervised task because a dataset of already labeled examples is used to learn the structure of groups and it is this additional knowledge that makes this approach more powerful than an unsupervised one, such as clustering. A classification algorithm consists mainly of two phases: a training phase in which part of the available dataset is used for the construction, or training, of the model and a test phase in which the model is evaluated through the predictions that it makes on samples unseen before.

In the progress of our work, several supervised learning algorithms have been used to perform classification and their description is provided below. The need to try multiple classifiers arises from the fact that there is not one that is better than the others in all situations: each has its advantages and disadvantages and it is necessary to test them directly on the data to see which is better.

k-Nearest Neighbor. Neighbours-based classification [6] is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. When new unlabeled data arrives, classification is done by a majority vote of its closest k train elements, called nearest neighbours. The basic nearest neighbours classification uses uniform weights but, under some circumstances, it is better to weight the votes such that nearer neighbours contribute more to the final prediction. There are two parameters that can be regulated: the number of neighbours to consider (k) and the metric with which distance between points is computed. Both affect

the performance of k-NN: the k value affects the shape of the boundaries and sensitivity to outliers while the distance metric defines which samples are similar and which not. The advantages of k-NN are:

- It does not need a lot of basic assumptions to be implemented, unlike other classifiers.
- It is a non-parametric approach that does not train a model, so it can be updated online just by adding a point to the train set.
- It guarantees a good generalization for a large number of samples.

The disadvantages of k-NN are:

- It is computationally expensive because it is necessary to store all training samples and then compute the distance from all of them for every test point. This problem can be solved applying some strategies to make this algorithm scalable:
  - We can simply limit the distance computation using only features that we know are relevant, making the calculation lighter. This strategy requires a previous knowledge on the utility of the features. If we do not have it, we can apply a feature reduction technique to reduce the samples' dimension.
  - We can organize the data points in some structures and compare to the sample to be classified only some elements of each class. It requires to know how to organize the data.
- It suffers the *Curse of Dimensionality* because as the number of dimensions increases, the distance between points increases too and therefore the K nearest neighbors could be very far from the new sample they are evaluating.
- It's not that easy to choose the number of nearest neighbours to consider for the voting.

Support Vector Machine. Support Vector Machines (SVM) [7] constitute a class of "learning machines" that originate from concepts relating to the statistical theory of learning. In classification problems, the basic idea of the SVM is to identify a particular separation hyperplane (called optimal hyperplane) that maximizes the distance of the elements of the dataset from the hyperplane itself. In general, we can say that the greater this distance, the smaller the error made by the classifier. This error can also be null if the data are linearly separable. When linear classification is not possible, SVMs are able to effectively perform non-linear

classification using the kernel method, implicitly mapping the input data into a higher-dimensional space. The advantages of SVM are:

- It is an effective algorithm in high dimensional spaces.
- It is an effective algorithm when the number of dimensions of the training samples is greater than the number of samples themselves.
- The separation hyperplane is built based on a subset of training points called support vectors, so this algorithm is memory efficient.

The disadvantages of SVM are:

- It does not perform well when the training dataset contains a large number of samples because the required training time is higher.
- It does not perform very well when the noise level in the training dataset is high.
- The choice of the kernel is not obvious and it turns out to be crucial for obtaining good results. The most suitable kernel and the value of its hyperparameters can be found during the model validation phase.

Decision Tree. A decision tree [8] is a classifier expressed as a recursive partition of the input space. A decision tree consists of nodes connected by edges, forming a tree-like structure. The nodes have exactly one incoming edge, except for the first one, called root. If a node has outgoing edges, it is called an internal or test node. All other terminal nodes are called leaves. In a decision tree, each internal node tests one of the attributes of the incoming data and then divides them according to their value for the selected attribute in order to obtain a node whose incoming examples belong all or almost all to the same class. The final prediction is given by the most present class in the leaf where the unlabeled sample will arrive after being guided by the value of its attributes through a path that starts from the root. There are several splitting criteria that can be used to choose the best attribute to divide the samples. The number of created branches depends on the type of the chosen attribute: we can create a branch for each value of a categorical attribute or we can select a separation point within the scale of values of a continuous attribute, thus originating only two branches. The separation of the samples contained in a node continues until the remaining ones all belong to the same class or until a stopping criterion is satisfied, based for example on the depth of the tree or on the number of remaining samples. Decision tree advantages are:

- The algorithm is simple and it creates a model easy to understand and interpret, perfect for visual representation.
- It is a very fast technique.
- The feature selection process happens automatically because the most relevant ones are selected during the training thanks to splitting criteria.

Decision tree disadvantages are:

- It tends to overfit easily. The overfitting can be avoided choosing an effective stopping criterion or by applying a ensemble method. This last strategy consist in generating a group of base learners and combine them to achieve higher accuracy on test set, through majority voting. This voting system can be also weighted based on the accuracy achieved by each single base classifier.
- It is unstable, meaning a small change in the data can completely upset the structure of the tree.

Random Forest. Random forest [9] is an ensemble method specifically designed for decision tree classifiers. Random Forests grows many trees where each base classifier classifies a different random vector of attributes from the original training dataset. The final result on classifying a new instance is obtained by voting over all the trees in the forest. This approach introduces two sources of randomness: the former is that each base tree is grown using a subset of the original training dataset, the latter is that at each node of a tree the best split is chosen not from all the attributes, but from a random subset of them. Once the number M of attributes to use for train each tree has been fixed, it is possible to proceed in many ways at each node to add more randomness: we can select the M attributes randomly and then choose the one with the higher score based on the splitting criterion or alternatively we can compute the score for all attributes, select the top M and then choose randomly one of them for the split. Random Forest algorithm advantages are:

- It resolves some of decision tree problems. It is more robust to outliers and the risk of overfitting is lower.
- It works well on large datasets and with non-linear data.

Random forest disadvantages are:

- The training time is higher than decision tree, because it has to train several trees.

- Random forests are found to be biased while dealing with categorical variables because they will favor those with more values which can pose a prediction risk.

2.1.2 Methods for Model Validation

Since there is no classification algorithm that best suits all possible situations, after considering different classifiers or different version of the same classifier and after applying their functioning to our data, we need a criterion that allows us to choose the one that best suits the problem that we are trying to solve. There is a large variety of indicators, but usually the most used are the prediction accuracy or the prediction error. The prediction accuracy can be trivially defined as the ratio of correctly classified examples over the total number of examples to be classified and it is discussed in detail in the following section. The prediction error is its opposite, so it is equivalent to the percentage of examples classified incorrectly. In most real world cases these indicators are unknown because they cannot be calculated, so they must be estimated in some way. For this reason we will talk about *estimated* prediction accuracy/error and typically these indicator are computed dividing the input dataset in train set and test set, then training the model exploiting only the train set samples and eventually estimating its accuracy over the test set elements. There are many other different strategy which help to estimate these indicators in a more robust way than described above, especially in the presence of a limited amount of data. Most common performance evaluation techniques, some of which used in this work, are explained below.

Hold Out. It is the simplest method. It consists in dividing the available dataset into two disjointed sub-sets, one called train set used to build the model and one called test set used to evaluate the model. This division is fixed and the elements of the test must never contribute to the construction of the model in order to not alter the final evaluation. Usually the train set is composed of 70-80% of the data while the test set by the remaining 20-30%.

K-fold Cross Validation. Using K-fold cross validation [10] the input dataset is split in K parts of almost the same size, called folds. Then $K-1$ parts are used as train set and the last one as test set to build and evaluate a model. The process is repeated K times, every time leaving out of the training a different fold, until all subsets were used as a test set. At the end the average accuracy/error is computed considering all the results obtained during the K repetitions. In this way, this technique allows to derive a more accurate estimate of model prediction performance. A variant of this technique consists in dividing the

input dataset into K folds while maintaining the same distribution of class labels in each of them [11]. This operation is called stratification and allows to obtain more balanced partitions from the point of view of the represented classes, in order to obtain a more reliable error estimate.

Leave-One-Out Cross Validation. It is an extremization of K-fold cross validation in which the cardinality of the input dataset is used as the fold number. In this way all the partitions will contain exactly one input element. Leave-one-out cross validation is particularly useful when the cardinality of the input dataset is small. Unlike K-fold cross validation, this technique is also an unbiased estimator of the generalization error, because the entire dataset and the training set used to build the model differ only for one sample, so it is a really good way to understand how will be our model in the future. The time required for the execution of this technique is higher than those of the strategies seen so far because it has to train as many models as there are elements in the input dataset.

Bagging. It is an ensemble strategy to perform model evaluation. We choose a classification algorithm and we use only that one but with different training sets. These training sets are created with bootstrap aggregation that is a technique that perform repeatedly random re-sampling of the original training set (samples are picked with the same probability even for more than one sub-train set). At the end, we obtain the prediction value with the majority voting. This approach decreases error, especially of unstable learners, such as decision trees because they depend a lot on the training set and they can easily overfit.

Boosting. It is another ensemble technique. The idea is to build a strong learner by combining several weak learners. This can be done in many ways but the most popular one is called AdaBoost. In addition to re-sampling, we re-weigh the examples: the sample probability of being picked depends on its weight. At each iteration, a new weak learner is trained, and the training samples are re-weighted to focus the system on the ones that the most recently learned classifier got wrong (i.e. training samples that were misclassified are given a higher weight than the ones correctly classified). For each classifier is also computed its importance, proportional to the accuracy achieved. The final classification is based on weighted voting of weak classifiers (where the weights are the learner importance). When used with decision tree algorithms as base learners, considerably decreases overfitting. On the other hand, it is sensitive to noise and outliers.

2.1.3 Metrics for Performance Evaluation

An evaluation metric can be described as the measurement tool that measures the performance of classifier [12]. In most classifiers, the predicted output is in the form of a label associated with the test instance. In such cases, the predicted class of the test instance is compared with the ground-truth label to calculate a value that indicates the overall classifier goodness. Different metrics evaluate different characteristics of the classifier induced by the classification algorithm. An example can be the use in the test phase to choose the best classifier or the best among the models trained with the same classification algorithm, if we have more than one in comparison. An example of use in the train phase could be for purposes of optimization of the classification algorithm to make sure that the model that is being trained is able to make better future predictions. Below there is a description of the most commonly used metrics with an explanation of the situations to which they are best suited.

Accuracy. Accuracy is the most widely-used metric for evaluating a model. In general, it can be defined as the percentage of correctly classified examples out of the total of examples to be classified.

$$\text{Accuracy} = \frac{\#\text{ correctly classified objects}}{\#\text{ classified objects}} \tag{2.1}$$

However, accuracy is not always the best metric to use, especially when there is an high level of data unbalancing. The basic problem is that when one of the classes is dominant, it is possible to achieve high accuracy simply by predicting that class for all or almost all the test samples. As a consequence, when the classifier makes the prediction for an example of a minority class, even if it has a very high accuracy, it is still likely that the predicted label will be the one of the majority class. Getting such a prediction wrong can be very damaging in some contexts where the minority class is the most important (some examples are described in the following section). In conclusion, accuracy is a metric that analyzes the classifier globally but does not allow to evaluate only the predictions relating to a single class.

Precision. Precision is a metric calculated in reference to a single class. It is computed as the ratio of the number of true positives to the sum of true positives and false positives. The true positives are the examples of the class under analysis that have been correctly classified. False positives are all those examples belonging to other classes but which have been classified with the label of the reference class.

$$\text{Precision} = \frac{\#\text{ objects of class C correctly classified}}{\#\text{ objects classified with the label of class C}} \tag{2.2}$$

Recall. Recall is also a metric calculated in reference to a single class. It is computed as the ratio of the number of true positives to the sum of true positives and false negatives. The true positives are the same as explained for precision. False negatives are all those examples belonging to the class in analysis but which have been classified with the label of another class.

$$\text{Recall} = \frac{\text{\# objects of class C correctly classified}}{\text{\# objects of class C}} \tag{2.3}$$

To have an overall evaluation of the effectiveness of a model, it is better to examine both precision and recall metrics for all the classes involved in the process. For this reason is very common to see these two metrics computed together. Unfortunately, precision and recall are often in opposition because whenever we try to improve one of the two, the other is likely to get worse.

2.1.4 Resampling

One problem that plagues many datasets is that of unbalanced data [13] in which more instances are belonging to some classes than others. Almost all classification algorithms suffer from this problem. Unbalanced datasets usually causes biases in classification and leads to poor generalization performance, due to the fact that learning algorithms assume a balanced distribution of classes. This issue is of great relevance and has been intensively explored because in many fields the minority class is the most important one and it characterizes only an extremely low percentage of input samples. For example, if we consider the field of medical research and in particular the study of rare pathology, there are very little data relating to patients with these diseases compared to the totality of patients observed. In other cases, it may be useful to search for a high detection rate of the minority class by agreeing to allow a greater number of errors in the recognition of the most recurrent class, because the incorrect classification of an example belonging to the majority class has a relatively low cost.

One of the main researches on this topic is the implementation of solutions for handling the imbalance, both at the algorithmic and data levels [14]. The former designs a new classifier or improves the existing algorithms to increase the detection of minority class samples while the latter tries to flatten the dataset unbalance through changes to the distribution of classes. At the algorithmic level, solutions are based on adjusting the costs of the various classes to counter the class unbalance, adjusting the decision threshold or switch to a recognition-based approach from the classical discrimination-based one, that is learning from every single class

instead of learning differences between classes. At the data level, methods for balancing the classes consist of re-sampling the original dataset. This can be done either by over-sampling the minority class or by under-sampling the majority class until the classes are approximately equally represented. Both strategies are performed in the data preprocessing phase, so they can be applied to any machine learning algorithm which will not be aware of the use of these techniques because they are executed before the model training begins. Their usage completely solves the problem of data unbalance, but introduces others, in addition to the need to identify a good criterion with which to add or remove elements from the dataset. The over-sampling generates new elements but the growth of the dataset implies both greater computation times and greater use of resources. The under-sampling could instead eliminate key elements of the majority class, thereby hindering subsequent analyzes. The use of over-sampling is preferred to under-sampling, which is now mainly used where resources are the critical aspect. Sometimes the simultaneous use of these techniques can lead to better results than those obtainable by applying one of the two in isolation. In this work, the oversampling strategy described below was adopted.

Synthetic Minority Over-sampling Technique (SMOTE). SMOTE [15] is an over-sampling technique that generates synthetic training elements belonging to the minority class. The creation of new data starts by selecting one element that belongs to the least popular class, then a new data point is generated somewhere between the selected sample and each of its K nearest neighbor. This process is then repeated for each sample of the minority class. From the geometric point of view, the vector that goes from the element initially chosen to one of the K nearest neighbors is multiplied by a number between 0 and 1 and the result indicates an intermediate point between the two in which the new data point is generated. The feature values of the synthetic examples have an intermediate value between those of the original examples used for their creation. Although this solves the overfitting problem arising from other oversampling techniques in which the elements of the minority class are simply replicated as they are, it could, in some contexts, introduce the problem of the generation of elements whose features have values that are not admissible in real cases. The results show that the SMOTE approach can improve the accuracy of classifiers for a minority class. SMOTE provides a new approach to over-sampling. The combination of SMOTE and under-sampling performs better than plain under-sampling. The experiments conducted on this approach have shown that SMOTE can improve the accuracy of classifiers in all those cases where there is a minority class. Furthermore, it was found that the combination of SMOTE and under-sampling allows obtaining better results than

using only simple under-sampling.

2.2 Kernel Density Estimation (KDE)

The probability density function is a key element of statistical analysis. When there is a dataset whose probability density function is not known it is necessary to adopt techniques to be able to estimate it. The density estimation [16] is the construction of the estimate of the probability density function. This is done by directly exploiting the data itself. This operation is useful and can be applied in various fields. For example, we may want to predict when a natural disaster, such as an earthquake, might occur based on past events. Or we might be interested in finding out the likelihood of a certain disease occurring in a certain category of people given a large amount of subjects' medical records. The techniques of density estimation are close to those of unsupervised learning, feature engineering, and data modeling. Density estimation techniques can be classified in various ways. A possible subdivision is between parametric techniques and non-parametric techniques. The former assumes that sample data comes from a population that can be modeled by a probability distribution, in which parameters are fixed. On the other hand, the latter makes no assumptions about a parametric distribution when modeling the data. One of the most popular and useful is the Kernel Density Estimation (KDE) [17], a non-parametric technique that belongs to the family of the neighbor-based approaches.

KDE is used for pattern recognition and classification by estimating density in metric or feature spaces. For each element X within the feature space, the algorithm allows to calculate the probability of belonging to a class C, considering the density of C in a neighborhood of amplitude K of the point X. The use of KDE is similar to that of histograms but being able to choose the kernel that best suits the situation, the results are smoother and more continuous. The process to construct the estimate of the density function using KDE consists in placing a kernel over each data point and then sum all the kernel together in order to obtain the curve of density estimate. A comparison between histogram and KDE strategies is represented in figure 2.1, in which the red dotted lines represent the single kernels related to each element of the dataset, while the kernel density estimate is shown in blue.

One of the main problems with using KDE is the choice of the bandwidth [18], which is a smoothing parameter of the kernel that can not be predicted precisely. The choice of this parameter strongly influences the result of the estimate, as we can notice observing figure 2.2. The grey curve represents the true density which must be estimated. If a too small bandwidth value is selected, the individual kernels

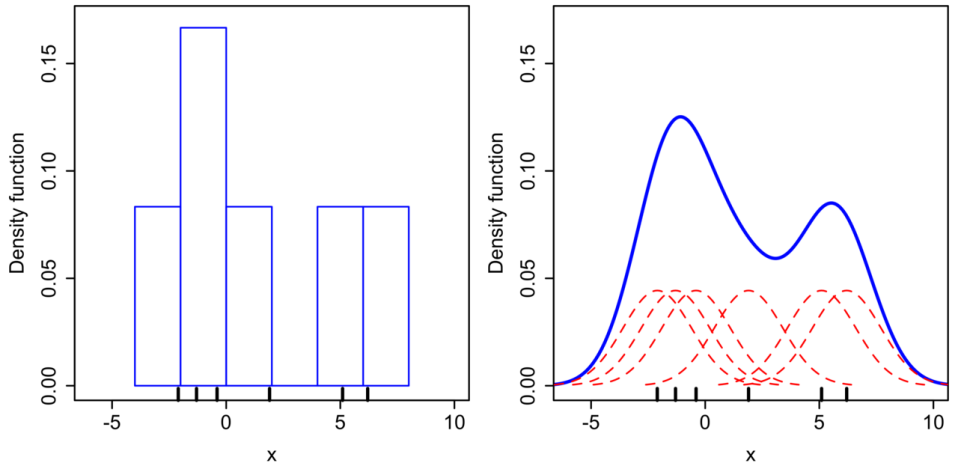


Figure 2.1: Comparison between a histogram (left) and kernel density estimation (right) constructed using the same data.

associated with each point of the dataset will be characterized by a steeper curve and will affect a more restricted area, consequently, the final result of the estimate will be undersmoothed, as can see looking at the red curve in the image. Conversely, if a too high bandwidth value is used the individual kernels will flatten out affecting a larger area and the resulting estimate will be oversmoothed, as shown by the green curve in the figure. The black curve is instead obtained using a correct value of bandwidth: it has the right level of smoothness and manages to get very close to the true density.

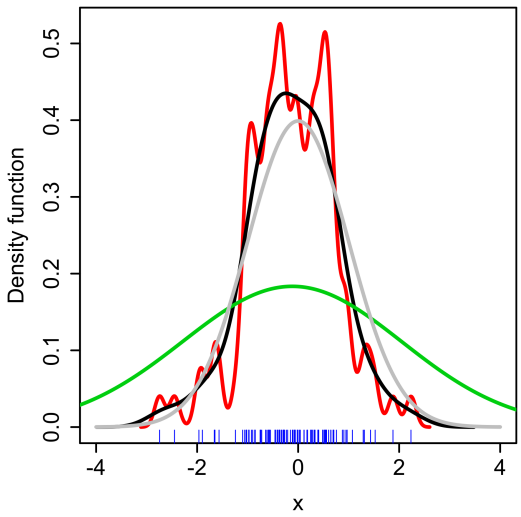


Figure 2.2: Different bandwidth comparison.

Chapter 3

Case Study

This thesis work was developed in collaboration with K-Master Srl, a company subject to coordination and control of Telepass SpA that offers advanced fleet management services dedicated to road haulage companies. K-Master has provided us with their data and guidelines on its semantics. This chapter begins by giving an overview of what fleet management is, focusing on the specific task that interests us most, namely the evaluation of driving behaviour. Next, it analyzes our specific case study describing in detail the data, both those provided by the company and those recovered in other ways.

3.1 Fleet Management

Fleet management is the set of management functions related to a set of vehicles that allows a company to improve efficiency and productivity and to reduce costs and risks. Any operation involving multiple vehicles and which must in some way coordinate or monitor them can be considered a fleet management operation. The concept can be extended to any type of vehicle, including air and naval ones, but K-Master mainly deals with land vehicles, in particular heavy transport vehicles. This area should not be underestimated: for example, in Europe, the exchange of goods between different states takes place 75% by land [19]. A concrete example of fleet management operation can be the management of customer delivery requests arriving randomly over time: since the delivery of the goods can take days or weeks in some cases, it is necessary to ship the equipment that meets the requests even before knowing them in order to be able to satisfy the customer.

3.1.1 Driving Behaviour

K-Master is interested in a particular aspect of fleet management: **driving behaviour**. It is interesting to analyze driving behaviour because some studies have shown how this can be modified, and therefore improved, if the driver receives the right feedback about his driving performance [20] [21]. The meaning of driving behaviour can be interpreted in several ways. A common definition describes the driving behaviour as the set of actions the driver performs to ensure both the safety of people and compliance to the driving regulations. Many studies conducted classify the driving style trying to trace the actions carried out by the driver to his personality or state of mind, as happens in [22]. Such an assessment requires monitoring not only the driving style but also other factors, such as the level of attention of the driver and the precautions to be taken before, during and after the journey, all in relation to the surrounding environment. Other works [23] take into account other factors and divide them in two categories:

- continuous behaviours, which remain stable for a medium or long period, such as over-speeding or under-speeding.
- abrupt behaviours, whose occurrences are sudden and characterize certain moments, such as a sudden change of lane or an unexpected acceleration. These behaviours are more easily detectable at the data level.

However, the company is more interested in a type of driver behaviour whose analysis could provide its customers with some kind of gain or savings, so we can alternatively define the driver behaviour as the set of actions that the driver performs in order to minimize travel times and costs. Several studies have shown that only by positively influencing the driving style is it possible to reduce fuel consumption, in some cases even more than 10% [21]. Moreover, a company may be interested in reducing CO2 emissions, both to safeguard the environment and to comply with any legislative constraints. This aspect can also be influenced by the driving behaviour [24]. The evaluation of what has just been described requires focusing on other key indicators than those mentioned previously, such as travel times, fuel consumption or the choice of the route by the driver.

The quality of the driving behaviour can be calculated as a **Key Performance Indicator (KPI)**, an indicator that evaluates the correctness of a certain action or function. K-Master already provides its customers with a score related to each monitored trip. Our goal is to modernize the strategy by which this value is computed. In particular, it has some disadvantages that need to be contained and we intend to do this by introducing machine learning techniques, which have not

been exploited by the company so far. A common technique used in the fleet management for the calculation of KPIs related to the driving behaviour involves the use of a "golden example" to which all the elements in analysis refer. This golden example represents the KPI with the maximum success rate, that is the one with the highest value, and all the other KPIs are calculated based on a comparison with the latter: the greater the differences with the golden example, the smaller it will be the value of the KPI. Conceptually, the idea is very simple and must be enriched by specifying both which factors will be considered during the comparison and the relevance that will be assigned to each of them, but in any case it is subject to a specific disadvantage: it requires human intervention for the definition of the golden example. This is a disadvantage for two important reasons:

- the characteristics of a golden example are not known a priori and should be defined by a domain expert after careful analysis.
- given the amount of vehicles managed by K-Master, a single golden example is not enough to satisfy all the cases in which we may find ourselves having to calculate a KPI. This fact introduces a trade off between the amount of reference examples provided and the adequacy of the comparison for the correct evaluation of a KPI. Furthermore, the quantity of monitored vehicles is constantly growing and the use that the customer makes of them is constantly changing, therefore periodic human intervention would be required to add the golden examples necessary to satisfy the new cases.

For these reasons it was decided to use an alternative approach that does not require human intervention for the validation of driving behaviour. The strategy used is based on the analysis of the similarities between trips made in order to identify anomalies that can be traced back to incorrect or non-optimal driving behaviour. The purpose of this work is to create a framework that, starting from the raw data collected by the devices installed on the vehicles, is able to autonomously assess, with a KPI, the quality rate of the driving style adopted in each trip. In particular, the KPI we intend to calculate must not be based on a golden reference or on any other type of validation supervised by an operator, but must be relative to the rest of the data available. Furthermore, the system will also be able to provide explanations on why a trip is not close to the identified standard.

3.2 Data Description

There is a lot of information that can be involved in the fleet management process. One of the most important is the vehicle tracking that in its simplest form it is

composed by the GPS coordinates of the vehicle, recorded constantly or periodically during its movements. These coordinates can be registered with GPS devices that the vehicle is equipped with or whit more complex satellite devices that can provide also direction and speed information. Other important information that can affect fleet management are fuel consumption data, load specifications and registry of the vehicle, repair and maintenance data and more. Obviously, the more information you need, the more sensors and devices the vehicle must be equipped with to be able to record it.

3.2.1 Data Provided

The analyses focuses on all the data collected in June 2020, which constitute a considerable amount of information that occupies over 70 GB. The vehicles monitored in that time frame are 30553 and the messages transmitted are hundreds of millions, but not all vehicles transmit the same amount of information. This happens because the company provides its customers with two possible levels of service:

- **Business** service, the vehicles are equipped with a GPS detector that transmits at a rate of approximately one message per minute. The information transmitted are the GPS coordinates, the timestamp and the identifiers of both the vehicle and the message itself. This is the basic service and is installed on all monitored vehicles.

- **Connected** service, the vehicle is equipped with both the same GPS detector provided in the business service and a Controller Area Network (CAN) bus [25]. The CAN device monitors many vehicle components and periodically transmits measurements. The transmission frequency is lower than the one of GPS data and it varies according to which component we are interested in observing. The information transmitted by this device concerns speed, fuel consumption, engine revolutions, engine temperature, brake pressure and many others. This is the advanced service and the vehicles fitted with this device at the time were under 14% (4230 of the total).

K-Master data is stored into MYSQL server database, organized in three main tables:

- table *satmsgPos*. This table stores GPS messages. Each record contains several information collected at the sampling time, such as the current speed, latitude, longitude and others. The *idMsgVpr* attribute is a foreign key for the CAN header table.

- table *satvpr*. The CAN header table. Apart from timestamp attributes, this table is used as bridge table between the GPS messages and the CAN bus messages. The *idMsg* attribute is a foreign key for the CAN bus message table.
- table *satvprItem*. The CAN message table. Each record is associated with a CAN header (from table *satvpr*), and contains the information of a specific CAN message. In this table each record contains a single CAN indicator, therefore different records can correspond to the same transmission and consequently can be associated with a single element of the first table containing the GPS coordinates.

3.2.2 Data Enrichment

The actions that the driver performs while driving and the state of the vehicle itself must be evaluated in relation to the surrounding environment, in order to correctly assess driving behaviours. For this reason, it was decided to associate additional information relating to the location and time in which the measurements are taking place to the data provided by K-Master, which are obtained from sensors that monitor only the vehicle. This should allow us to place the driving actions carried out in a context that can motivate their choice. In order to be taken into consideration, this new information must refer: to a precise geographical position, therefore to a pair of GPS coordinates, and to a precise instant of time, therefore to a timestamp. Another important aspect to consider is that the information we intend to add is specific enough to satisfy the frequency of data transmission from the vehicles. The information of this type, for that it is possible to find a free data history on the web, is not much. The ones we have identified and used in this work are described below.

Weather. Thanks to the API called DarkSky, it was possible to find the weather forecast of a certain position at a certain instant of time in the past. This API provides a large amount of indicators but only 10 of them have been selected in those related to the evaluation of driving behaviour and added to the data already provided by K-Master. These include the percentage of sky coverage, the probability, the intensity and type of precipitation, the temperature, the visibility and the times of sunrise and sunset.

Altitude. Using the API called Jawg it was possible to find the altitude relative to a certain position, indicated by a pair of GPS coordinates. The altitude is expressed in meters above sea level. The value is not added to the various records

as it is reported but we use the variation with respect to the previous record. In the case that the record is the first of a trip, the altitude is set to 0. The calculation of the altitude deltas is relative to the order of the records, so it will be positive in case of uphill and negative in case of downhill.

Chapter 4

Related Works

The business vehicle industry has been changing rapidly in recent years. If until a few years ago this aspect was decidedly marginal, today it is no longer the case. Although the COVID-19 emergency has slowed down the growth of the logistics sector, the study of innovative forms of fleet management has never stopped. The same is true for fleet management software development which is constantly increasing even though there is already a wide range of tools capable of satisfying most needs. What matters most are the costs and the environmental impact of corporate and private vehicles, factors that will be predominant for the work of the fleet managers of tomorrow. Among all the functions that the fleet management deals with, the driver behaviour analysis is one of those in which research is most active, thanks also to the continuous improvement of technologies that allow monitoring vehicles and drivers. Understanding the correct driving style is of particular interest for several reasons, for example for the contributions it makes to environmental protection or to the design of autonomous vehicles.

The usable factors on which to base the estimate of the driving behaviour are many and have been studied both individually and in combination with each other. Among the most common are speed and fuel consumption.
Eboli, Guido, Mazzulla, *et al.* [26] define the driving behaviour of car drivers based on speed values recorded by real-time vehicle tracking. They find out how the speed is related to the characteristics of the driver, such as age and driving experience.
Also, GPS data has been exploited by Guo, Liu, Zhangin, *et al.* [27] in order to develop a hybrid unsupervised deep learning model to study driving behaviour. In this paper, GPS data is seen as time-series and is used in combination with speed changing.

Carpatorea, Nowaczyk, Rögnvaldsson, *et al.* [28] propose a machine learning methodology for quantifying and qualifying driver performance, with respect to a value derived from fuel consumption, called Fuel under Predefined Conditions (FPC). FPC has the purpose to provide a comparison term that captures some of the unmeasured factors and it is more accurate than the fuel consumption raw values.
Another interesting approach is based on Accelerator Pedal Position and Engine Speed (APPES) [29] [30]. It has the intent to qualify driver performance irrespective of the external factors by analyzing driver intention.
The works mentioned above are just some of the possible approaches. Even considering only the information provided by the vehicles of the K-Master business service, it is possible to find a large number of similar studies that try to describe the driving behaviour starting only from the GPS coordinates [31] [32] [33].

Given the presence of a CAN bus device on part of the vehicles monitored by K-Master, it is interesting to observe how these data are exploited and combined with other information in order to evaluate the driving behaviour. It is important to note that the data available regarding multiple vehicles equipped with CAN bus are not necessarily the same, as the CAN bus represents the message based protocol with which measurements are recorded and transmitted, while the content of the transmission depends exclusively on the sensors installed on board in each vehicle.
Choi, Kim, Kwak, *et al.* [34] propose the usage of Hidden Markov Models (HMMs) to capture the sequence of driving characteristics acquired from the vehicle's CAN bus information. In this article, the analyzes are also conducted considering the actions that the driver takes inside the vehicle compartment but which do not directly affect the driving activity, such as the interaction with the radio or with the windows. This choice certainly makes sense but unfortunately, this information is not present in our dataset.
Fugiglando, Santi, Milardo, *et al.* [35] work in a different context, describing the complexity of the driving behaviour through the concept of "Driving DNA". Their study focuses on the use of some specific information, coming from both the CAN bus and not. These include speed, rain and fuel consumption, which we also have in our dataset.
Another example is Ferreira, de Almeida and da Silva [36] where there is a preprocessing step that merges data from the CAN events' dataset with GPS coordinates, weather conditions and altitude. During our work we had to face this step with the same data, paying particular attention to the selection of the most suitable CAN indicators.
The CAN bus is a very popular technology and it is easy to find many other studies that try to combine the transmitted data with information from other sources, such

as GPS, smartphones and other sensors [37] [38] [39].

Since we do not have any type of label that indicates which driving behaviours among those present in our dataset are the correct ones, the approach we are going to apply to the data will be unsupervised.
An example of an unsupervised study is [40] in which clustering techniques, such as Ensemble Clustering Method (ECM) and modified Latent Dirichlet Allocation (LDA), are used to understand different driving styles. It finds out that the different behaviours can be grouped into three main clusters: aggressive, cautious and moderate.
Chen and Chen [41] explore the possibility of identifying driving styles directly from driving parameters through Partitioning Around Medoids clustering method, again identifying three main clusters. Moreover, they do a pre-processing step in which speed data is aggregated in order to produce mean, standard deviation, maximum and minimum values and this idea has also been adopted by us for most of our features.
Marks, Jahangiri and Machiani [42] introduce and define Iterative DBSCAN (I-DBSCAN) as one tool that can be utilized for identifying aggressive driving behaviours within large, unlabeled datasets. Their goal is to retrieve a small subset of outlying driving observations that can be considered as "abnormal" behaviours. We are also interested in identifying these anomalous values in such a way that we can exclude them from those that will define the standard of good driving behaviour, so we perform a clustering phase with the same goal.

Work	GPS	CAN Bus	Contextual Features	Approach
Eboli, *et al.* [26]	No	No	No	Statistical
Guo, *et al.* [27]	Yes	No	No	Unsupervised Deep Learning
Carpatorea, *et al.* [28]	Yes	Yes	No	Statistical
Carpatorea, *et al.* [29]	No	Yes	No	Statistical
Carpatorea, *et al.* [30]	No	Yes	No	Supervised Shallow Learning
Grengs, *et al.* [31]	Yes	No	No	Statistical
Sun, *et al.* [32]	Yes	No	No	Statistical
Brambilla, *et al.* [33]	Yes	No	No	Unsupervised Shallow Learning
Choi, *et al.* [34]	Yes	Yes	No	Supervised Deep Learning
Fugiglando, *et al.* [35]	Yes	Yes	Yes	Graphical
Ferreira, *et al.* [36]	Yes	Yes	Yes	Data Mining
Jackobsen, *et al.* [37]	Yes	Yes	No	Statistical
Fugiglando, *et al.* [38]	No	Yes	No	Unsupervised Shallow Learning
Taha, *et al.* [39]	No	Yes	No	Supervised Shallow Learning
Wu, *et al.* [40]	No	Yes	No	Unsupervised Shallow Learning
Chen, *et al.* [41]	No	Yes	No	Unsupervised Shallow Learning
Marks, *et al.* [42]	Yes	No	No	Unsupervised Shallow Learning

Table 4.1: Related works overview, indicating for each the presence of GPS features, CAN features and context features and the type of approach with which the data were analyzed.

Chapter 5

Framework Description

This thesis work focuses on the final part of the implemented framework. This chapter, after an introduction on the terminology used, briefly describes the entire structure of the pipeline, listing some characteristics of each step, and deepens the explanation on the final part explaining the insights that guided the realization, starting from the exploration of the clustering results obtained in the last step up to get to the calculation of the KPI.

5.1 Terminology

Listed below are some terms with their explanations that are either not specific to the world of transport or that will be used with a specific meaning that must not be misunderstood. The choice of these terms and their meanings were made in agreement with K-Master domain experts in such a way as to reflect the terminology used in their operational field.

Cell. Imagine dividing the world map with a square grid. By *cell* we mean the portion of the territory identified by one of the cells of this grid. Each cell has its unique identifier and a pair of indexes that locate it within the grid. The cells of a grid are all the same big, but the size is a configurable parameter, so it is possible to apply grids with different cell sizes.

Route. It identifies a possible journey that can be traveled both multiple times and by multiple vehicles. The *route* is defined by a pair of cells, those of departure and arrival. It does not take into account the path taken to reach the destination: two trips from point A to point B involving different roads still belong to the same route. Moreover, the route is not bi-directional: a trip from cell A to cell B does not belong to the same route as a trip from cell B to cell A.

Trip. It corresponds to an execution made by a vehicle of the path indicated by a *route*. To distinguish the various journeys made by the same vehicle, we rely on the timestamps of the GPS information transmitted: if two consecutive messages have a temporal distance above a certain threshold, called *Time Delimiter* (TD), then there is a change of trip and therefore the first message corresponds to the termination of the trip in progress while the second corresponds to the start of the new trip.

5.2 Pipeline Structure

The code has been structured as a pipeline of scripts that must be executed sequentially in order to pass from the raw data contained in the input database to the evaluation of the driving behaviour with the use of KPIs. The execution of some scripts is mandatory for the achievement of the final evaluation, while it is optional for others as they provide additional information but not necessary for the final calculation. The pipeline consists of 7 fundamental steps which are: extraction, filtering, type assignment, enrichment, aggregation, route identification and KPI computation.

Figure 5.1: Pipeline structure.

Extraction. The raw data is extracted from database. Its tables are joined together and a single file CSV is created for each vehicle. During the process all the columns not useful in the following phases are removed in order to reduce the size of the output files as much as possible.

Filtering. The records in each vehicle file are aggregated in messages, then the messages are grouped in trip and enriched with other information retrieved knowing the position and the timestamp. During the process only the CAN indicators of interest are kept. Then, each GPS position is associated with one cell, applying the grid model. Next, both single record and entire trips are filtered, if the user has specifically indicated it. The filters applied verify that the information for a certain trip is both truthful and quantitatively sufficient to be able to carry out subsequent analyzes. These controls concern the values of the coordinates, the speed limits, the thresholds of road traveled and time elapsed and the quantity of messages describing a journey. The output is still a file for each vehicle.

Type Assignment. This is an optional script that can be executed on the output files of the previous one. A type between *heavy* and *light* is assigned to the vehicles, based on the registry information, if present. If it is not possible, a machine learning classifier trained on the available registry is used to infer the type. The output is still a file for each vehicle. If this step is performed, in the next phases the analysis can be limited to only one of the two types.

Enrichment. This is an optional script that can be executed on the output files of the previous one. The GPS coordinates and the associated time related to each message are used to enrich the data, adding information from external sources, such as weather conditions and altitude. The output is still a file for each vehicle. Subsequent scripts can be run either the context features are present or not.

Aggregation. The messages in each vehicle file belonging to the same trip are collapsed into a single data point that represents the trip itself. The features that characterize each record are processed to originate new ones relating to the entire journey. In this processing phase some meta information about the trip are computed, such as identifiers, vehicle details and starting and ending coordinates/-timestamps, while the others indicators are treated as in [40], computing mean, standard deviation, maximum and minimum for each of them. The output is a single file containing trips information, one trip per line.

Route Identification. The trips are reunited in groups, one for each route, based on the departure and arrival cell. The data concerning the most populous routes are extracted in order to facilitate the following analysis.

KPI Computation. The trips of a single route are divided into train and test set. Train trips are grouped together using an unsupervised clustering method (DBSCAN). The outliers are removed from the train set and the remaining trips are exploited to train a model that is used to assign the KPI to the test trips. Eventually, a penalty score is assigned at each feature of each test trip that did not reach the maximum KPI, in order to make understandable why.

5.3 Key Ideas and Methods

This section describes and motivates the methodology used to achieve the goal, in particular, describing all the choices made to be able to calculate a KPI satisfying the requirements imposed by K-Master.

5.3.1 Data Distribution Analysis

The company would like that, where possible, the trips taken into consideration as an example of good driving behaviour are only those of vehicles equipped with a CAN device. However, the vehicles of the connected service do not always transmit messages containing CAN indicators. It can happen that sometimes they only transmit their GPS position as if they were vehicles of the business service. Considering only vehicles equipped with a CAN device, transmissions that incorporate at least one CAN indicator are 33.28% of the total (about 24 million out of 72 million). Not all the vehicle are able to monitor and transmit all the CAN features and furthermore the transmission frequency of each CAN descriptor is different both between different vehicles and between different indicators.

There are more than 100 different CAN descriptors present in the database but not all of them have been considered useful for the evaluation of driving behaviour. An analysis relating to the most pertinent (about 20 descriptors chosen in agreement with the K-Master domain experts) was conducted to study their distribution within the data. Table 5.1 shows some of the results obtained: for each reported CAN descriptor (first column) we find alongside the percentage of presence considering all messages (second column), the percentage of presence considering only CAN messages (third column) and the average percentage of presence per vehicle considering only CAN messages (fourth column).

Statistics about CAN bus messages			
CAN descriptor	**Total mes.**	**Total CAN mes.**	**Vehicle CAN mes.**
TimeEngineLife	13.38%	40.19%	41.05 ± 44.14%
TotalFuel	13.34%	40.09%	43.78 ± 44.75%
TotalVehicleDistance	13.33%	40.07%	54.57 ± 36.05%
EngineTemperature	8.54%	25.67%	36.88 ± 34.27%
EngineRpm	4.67%	14.03%	8.72 ± 27.82%
BrakeStatus	4.67%	14.02%	12.99 ± 29.63%
CruiseControlStatus	4.63%	13.90%	12.00 ± 29.63%

Table 5.1: Statistics about CAN bus messages, in details we have attendance % in all messages, attendance % in CAN messages and average attendance % inside a vehicle with CAN messages.

The values shown in the table demonstrate how CAN message types are different from each other. They often contain different descriptors and no CAN indicator appears in at least half of the total messages. Furthermore, the high variance values relative to the averages on connected vehicles indicate that the CAN descriptors are

commonly present in either all the CAN messages of the vehicle or none of them. This statistics lead us to two important conclusions. The first one is that there are very few CAN indicators present within the messages to be able to carry out an accurate analysis. We have in fact decided to consider only those descriptors that appear in at least 40% of CAN messages (see third column of table 5.1), which are *TotalVehicleDistance*, *TimeEngineLife* and *TotalFuel*. The second one is that journeys must not only be made by vehicles equipped with a CAN device but must have actually transmitted a sufficient quantity of CAN descriptors to be able to meet the company's requirement. For this reason, the trips have been divided into two macro-categories using the same nomenclature that the company adopts for their services but with a slightly different meaning:

- **Business trips**, they are all those trips made either by a vehicle associated with the business service or by a vehicle associated with the connected service but with a quantity of messages containing CAN information related to the three selected descriptors below a certain threshold.

- **Connected trips**, they are all those journeys performed by vehicles of the connected service in which the three selected CAN descriptors appear in a quantity of messages higher than a certain threshold.

By quantity of messages we refer to the total of messages transmitted and not only to those containing at least one CAN descriptor. Consequently, a threshold value equal to 10% of the total was chosen, slightly lower than the percentage of presence of the selected descriptors considering all messages (second column of the table).

5.3.2 Exploration of Clustering Outcomes

The clustering step in the last stage of the pipeline assign to each trip a label that allows to distinguish the element of cluster and the outliers. Before proceeding with the KPI calculation, these trips with their new labels are analyzed with a supervised approach to try to validate the clustering results. This analysis involved the use of several different classifiers and several performance evaluation techniques. It also has made possible to understand which are the characteristics of the trips that most guided the division into clusters. This is done in several ways such as through the interpretation of trained models. The analysis helped to understand whether the transition to a fully supervised approach is possible in the future, after a possible validation of the clustering results, and if so which classifier would be better to adopt.

5.3.3 KPI Computation

Given the trips that have passed the clustering phase, and therefore are supposed to be related to correct driving behaviour, it is necessary to find a strategy that uses them to evaluate new trips related to the same route computing a KPI. The basic idea is to allow each element belonging to one of the clusters to provide a contribution for the evaluation of a new example, proportional to the similarity between them. This insight perfectly reflects the mechanism that Kernel Density Estimation uses to estimate the probability density function. Let's consider the case where all the remaining trips belong to a single cluster. We apply to these elements the Kernel Density Estimation function which uses a Gaussian kernel. The result will be a function that maps each point in space with a numerical value proportional to the proximity between that point and the input elements. Figure 5.2 shows a 2D example of applying KDE to a set of 50 elements. The darker the color of the space region, the higher the KDE value at that point.

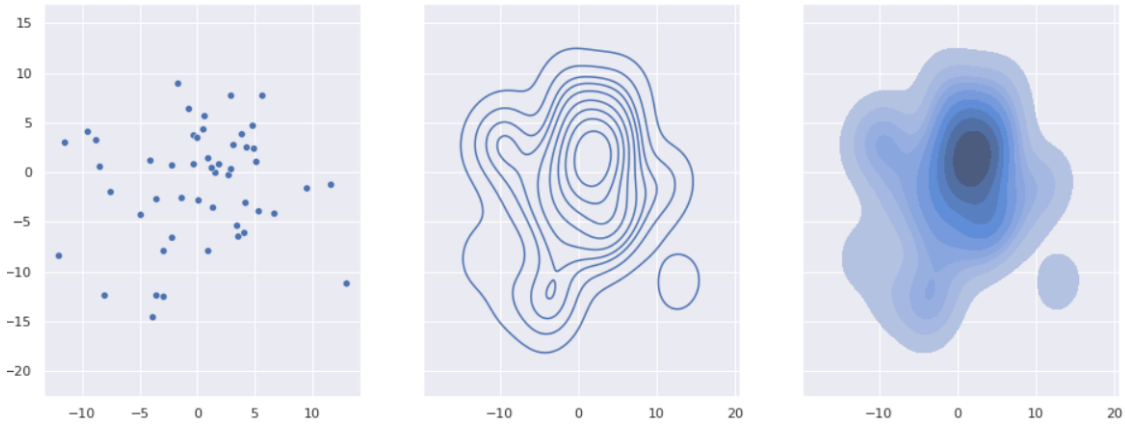


Figure 5.2: 2D example of KDE applied on data points.

Once estimated, the probability density function can be evaluated on new unseen input data points to assign them a numerical value. Figure 5.3 adds to the previous example new points, highlighted in yellow, which receive the value corresponding to their position. As you can see from the image, these additional examples do not contribute to the calculation of the KDE which therefore remains unchanged.

In the current situation, each new trip is assigned to a value which, however, cannot yet be considered the value of the KPI. Although these values are based on proximity to train trips and therefore directly proportional to how good the driving behaviour is, they still do not allow us to understand the distance from what could be defined as perfect driving behaviour. To have a reference point to use as the

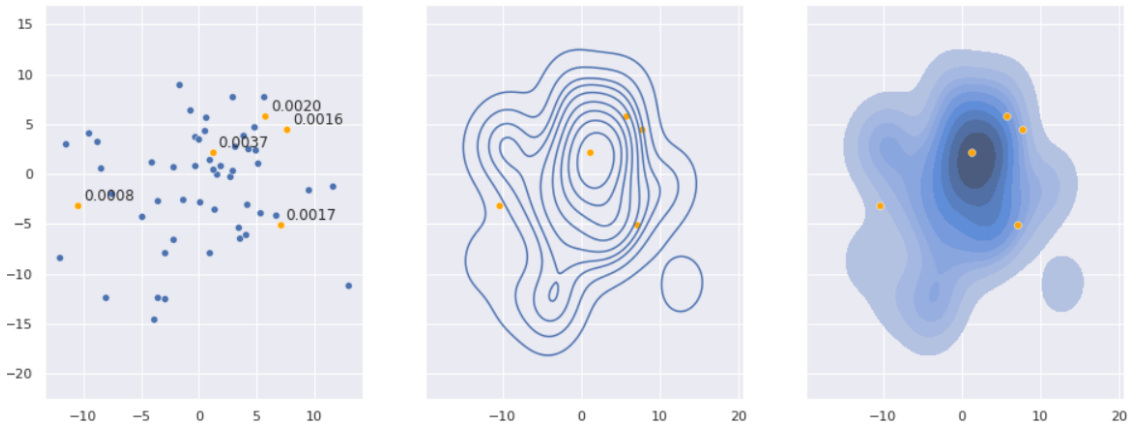


Figure 5.3: New data points to be evaluate are added.

best driving behaviour, we select the cluster medoid and consider its KDE value as the maximum possible score. Figure 5.4 shows which of the initial points is the medoid of the cluster and which is its score. In addition, as imaginable, it is noted that the medoid is located in the region of space with highest values of KDE.

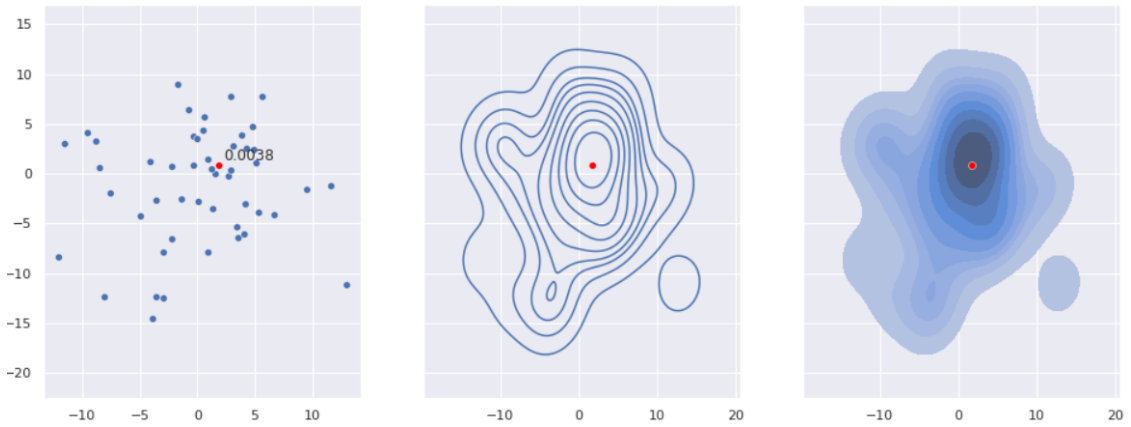


Figure 5.4: Focus on the cluster medoid. Its position and score are highlighted.

Now we can calculate the final KPI value by normalizing all the travel scores between 0 and 1, where 1 corresponds to a score equivalent to that of the medoid found in the previous step. Figure 5.5 shows both the medoid and the trips to be evaluated with relative KPIs, obtained by normalizing the previous scores.

In the event that during the clustering phase the trips were grouped into several clusters, the choice of the medoid to use as a reference must be managed. One

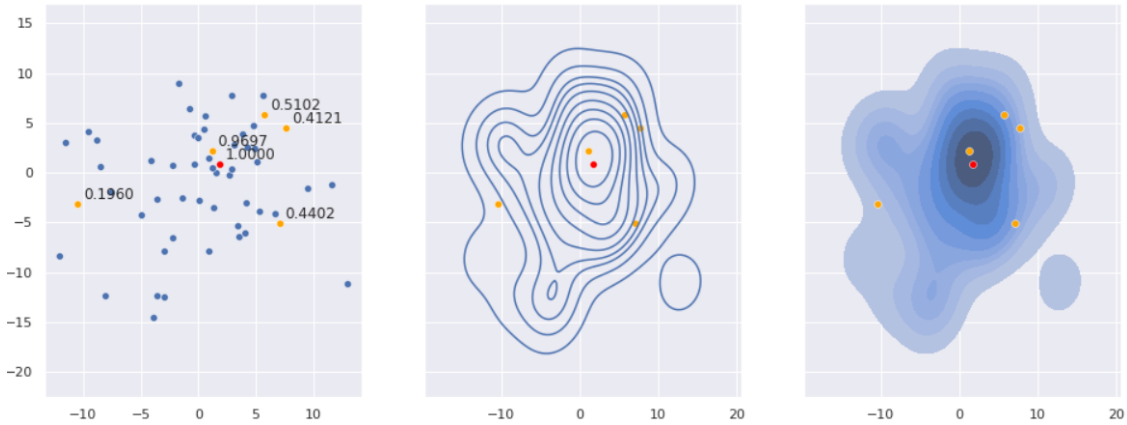


Figure 5.5: Comparison between medoid and trip to be evaluated and their scores after the normalization step.

idea would be to choose the medoid which corresponds to a higher KDE value. This is equivalent to considering the cluster to which that medoid belongs as the only representative of good driving behaviour. Given that the algorithm used for clustering (DBSCAN) originates clusters of different size and density, the medoids of the other clusters could have significantly lower KDE values, thus leading to a lowering of the KPI of all those new trips placed within their clusters. However, we hypothesized that each cluster corresponds to a different good behaviour, each to be adopted in different conditions from those of the other clusters, and that bad behaviours have already been excluded as outliers in the clustering phase. The strategy used is therefore based on choosing as the reference medoid the one closest to the example to be evaluated. In this way, we imagine obtaining a more truthful value of the KPI since it is as if we are taking into account in the evaluation of the conditions in which the guide took place.

Sometimes it can happen that the KDE value of the medoid does not correspond to the maximum value found in the surrounding area. Consequently, it could happen to have to evaluate a new trip that corresponds to a KDE value higher than the reference one. This is not a problem as KDE values greater than that of the reference medoid and also close to that medoid are relatively few. Trips in this situation are assigned the maximum KPI, instead of a value above 1, as they really would get.

In conclusion, calculated in this way, the KPI represents the quality of driving behaviour as a percentage. A KPI of 0 indicates that the trip we are evaluating is outside the range of influence of all the input examples and therefore the driving

behaviour is bad or abnormal. A KPI of 1 is assigned only to journeys with characteristics very similar to those of most of the input ones and indicates an excellent driving behaviour comparable to that of the cluster's medoid journey.

5.3.4 Penalty Score

The final KPI increases as the distance from a reference medoid decreases. Every trip feature contributes to the distance, which results in the sum of all of them. Each feature contribution is hence responsible for a decrease of the KPI, since it effectively *separates* the testing point from the reference. As such, we called this value **penalty score**. While producing the KPI values, we also keep track of the penalty score given by each feature, for each testing trip. These values provide a basic explanation of a KPI that is not optimal (i.e. equal to the reference driving behaviour). By inspecting them, it is possible either to have a first look to features that contributed to the KPI evaluation the most, possibly understanding why a trip obtained a low/high KPI, and to select trips that need to be checked first. The module computes the penalty score for each feature as a percentage contribution to the whole distance from the reference. By doing so, we assume that all the features have the same weight and produce the penalty scores accordingly. However, there is the possibility to provide a list of weights if a non-uniform contribution should be discounted.

Chapter 6

Experiments

This chapter describes the subset of data considered in the analysis and the experiments conducted on it. As already explained, the final goal is to find a valid and innovative strategy for calculating the KPI, but these experiments also have other purposes. First of all, we want to validate the outcome produced by the clustering phase to understand if it can be used as a reference on which to base the computation of the KPI. In particular, we are interested in discovering the reasons that led outliers to be excluded and which features contributed most to this decision. This is done by applying different classification algorithms for the prediction of the class labels assigned by clustering and subsequently analyzing the results obtained and interpreting the trained models. Finally, we test the computation of the KPI with the solution proposed in the previous section, paying attention to simplify the interpretation, so that the results are observable even by non-machine learning users, and analyzing once again the relevance that each feature has in the calculation of the obtained values.

The data used went through a heavy preprocessing phase while running all the pipeline scripts. These steps can be regulated by various parameters whose modification involves obtaining a different output. The experiments were repeated on data coming from different pipeline runs, each with a different parameter configuration, to understand the differences that these parameters provide in the data and if and how they affect the calculation of the KPI. In detail, the parameters that have been modified to produce the various versions of the data used are:

- **Time Delimiter (TD).** The value of this parameter influences the trip definition. GPS messages are processed sequentially and their time stamp is used to group them in separate trips. If the elapsed time between two consecutive messages is greater that **TD**, then the current trip is labeled as "concluded"

and a new one is started. The higher the value, the more the waiting time necessary to consider a journey as completed will increase and therefore the fewer the trips in which the total distance covered by a single vehicle will be broken up. The versions of data used in the experiments were obtained with a value for this parameter equal to 20 or 40 minutes.

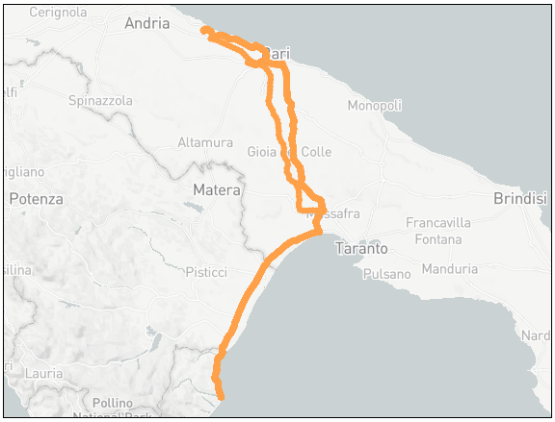
- **Cell neighborhood size (NS).** This parameter indicates the size of the neighborhood of cells that are considered as one to establish the starting or ending point of a route. The higher the value, the wider the area considered and therefore the greater the number of trips that could be included in the route. A value equal to 0 corresponds to a single cell. Each additional neighborhood level surrounds each free side of the cells of the previous level with other cells of the same size. The versions of data used in the experiments were obtained with an integer value between 0 and 2 for this parameter.
- **Contextual Features (CF).** The availability of contextual features (e.g., altitude and meteorological conditions) is indicated by this boolean parameter. By adjusting its value it is possible to choose whether to use data external to the original database or not. The versions of data used in the experiments were obtained both with this feature activated and not.

All tests were done considering each time only trips belonging to the same route. The characteristics that describe a trip are strongly linked to the type of route traveled and therefore, given that the calculation of the KPI is completely based on the similarity between different trips, it was considered sensible to operate at the route level without involving in the same experiment trips belonging to different routes. Furthermore, we remind that the company is interested in evaluating driving behaviour considering only vehicles with an active CAN device as a reference, therefore both our supervised analysis and the clustering results on which it is based take into consideration only connected trips. The experiments conducted belongs to two different types: supervised analyzes and the evaluation of driving behaviour. Although these two categories of expressions operate from the same input data - i.e. the results of clustering - their purpose is quite different. As described in the previous chapter, supervised analysis explores the results obtained from clustering by attempting to interpret the logic behind the subdivisions carried out and to provide any explanations for subsequent experiments. On the other hand, the behaviour assessment aims to calculate a KPIs that can describe the correctness of each trip. Therefore, even if the first part of the experiments does not directly influence the second, it is still useful as a tool for validating and interpreting the results.

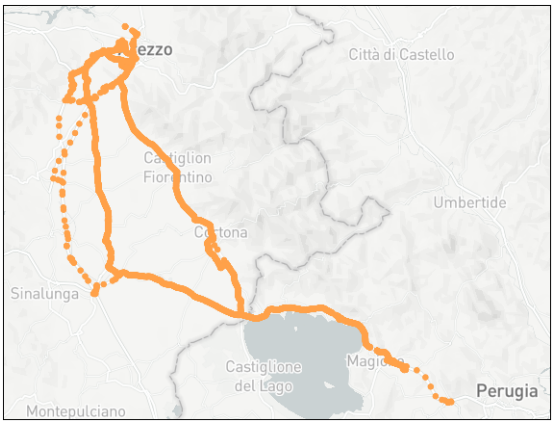
6.1 Route Identification

The routes analyzed were selected on the basis of the combination of parameters used during the execution of the pipeline and the quantity of associated trip connections. Of the three parameters listed above, the value of two of them (**TD** and **NS**) directly affects which and how many trips and routes are created. All possible combinations of the mentioned values have been tested and for each of them the route with the highest number of trip connected have been selected to perform the analyzes. This leads us to 6 different situations, some very similar to each other but never identical (e.g. same route but different amount of trips and consequently different labels assigned by clustering). Moreover, all these routes have been tested both considering and not considering the context features, so we have two possible clustering results for each of them. Eventually, since the trips of the last week of the month have been used as test set on which to evaluate the calculation of the KPI, all the clustering results at the base of the subsequent analyzes are related to trips made in the first three weeks of month.

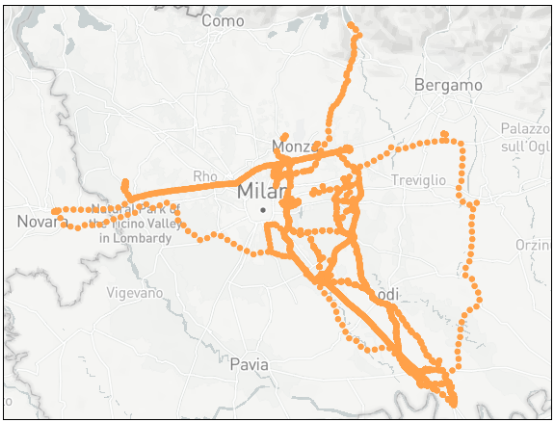
R1: TD = 20 & NS = 0 This route has been traveled 65 times by 7 different vehicles. All the 65 trips belonging to the route are *Connected*. The path of the 46 connected trip used as train set is shown in the adjacent figure. The clustering performed without context features discovered one cluster and one outlier, while the clustering with context features discovered one cluster and two outliers.



R2: TD = 20 & NS = 1 This route has been traveled 106 times by 28 different vehicles. *Connected* trips are 57 while *Business* ones are 49. The path of the 41 connected trip used as train set is shown in the adjacent figure. The clustering performed without context features discovered one cluster and 8 outliers, while the clustering with context features discovered one cluster and 11 outliers.



R3: TD = 20 & NS = 2 This route has been traveled 257 times by 121 different vehicles. *Connected* trips are 113 while *Business* ones are 144. The path of the 70 connected trip used as train set is shown in the adjacent figure. The clustering performed without context features discovered one cluster and 15 outliers, while the clustering with context features discovered one cluster and 4 outliers.



R4: TD = 40 & NS = 0 This route has been traveled 62 times by 7 different vehicles. All the 62 trips belonging to the route are *Connected*. The path of the 43 connected trip used as train set is shown in the adjacent figure. The clustering discovered one cluster and two outliers both including and not including context features.

R5: TD = 40 & NS = 1 This route has been traveled 98 times by 25 different vehicles. *Connected* trips are 50 while *Business* ones are 48. The path of the 37 connected trip used as train set is shown in the adjacent figure. The clustering performed without context features discovered one cluster and 8 outliers, while the clustering with context features discovered one cluster and 10 outliers.

R6: TD = 40 & NS = 2 This route has been traveled 106 times by 27 different vehicles. *Connected* trips are 51 while *Business* ones are 55. The path of the 38 connected trip used as train set is shown in the adjacent figure. The clustering performed without context features discovered one cluster and no outliers, while the clustering with context features discovered one cluster and 10 outliers.

6.2 Supervised Analysis

Clustering assigns a class label to each trip. This label is also used to identify outliers, i.e. those journeys that do not belong to any cluster and therefore are characterized by anomalous driving behavior. Before proceeding with the KPI calculation, we explored the clustering outcome to verify the reasons that led to the exclusion of the outliers. To do this, we applied various supervised learning techniques, observing how different models behave on the labeled data and, when possible, also observing the structure of the trained model itself to interpret and understand which characteristics in data most influence the predictions. Furthermore, statistical techniques were also used to study the correlation between features and between features and labels. Eventually, this type of analysis allowed us to understand if and how a possible supervised solution to our problem could work, highlighting what problems could arise in carrying out a classification on these types of data.

We start the analyses applying four different classifiers: **k-Nearest Neighbor (KNN)**, **Support Vector Machine (SVM)**, **Decision Tree (DT)** and **Random Forest (RF)**. These classifiers were used with the default values for their hyperparameters as provided by the *scikit-learn* library [43], with some exceptions. K-Nearest Neighbor classifier is executed with a K value equal to 10 or to the number of members of the minority class if this is less than 10. The distance used is the Euclidean one and the weights for all the K neighbours are uniform. Support Vector Machine algorithm is used with a radial basis function kernel, a C parameter equal to 1 and a γ parameter equal to $1/\#features$. Decision Tree algorithm splits the elements of its intermediate nodes based on the Gini Index and it proceeds the

construction of the tree without using any stopping criteria. Lastly, Random Forest uses 10 different trees as the one created by Decision Tree algorithm to predict the final label.

Two different evaluation methods are used to estimate how good each classifier is. The first one is a **K-Fold Cross Validation (KFCV)** with a value of K equal to 5. This technique is performed with a stratification approach in such a way to create folders with a constant ratio between elements of different classes. However, it may happen that the minority class, which often turns out to be that of the outliers, does not have a sufficient number of elements for there to be at least one in each folder. In these cases the number of folders is reduced to the number of elements of the minority class (less than 5). The second technique used is the **Leave-One-Out (LOO)**. This one does not require corrections to solve the problem encountered in the previous technique, indeed it is perfect for those cases in which there is a high level of data unbalance. Obviously these techniques are not applicable in those cases where the minority class contains only one element, as this cannot belong to both the train and test data, and consequently even all the rest of the supervised analyzes are not feasible.

An alternative solution to the class imbalance problem is to use a resampling technique that modifies the content of the dataset to have a similar amount of examples for each class. We have used an oversampling technique, called **SMOTE**, in order to increase the number of examples of the minority class by generating new synthetic data consistent with those already present.

To deepen the study of the outliers we tried to compare them with all the other elements belonging to a cluster. In other words, all the elements belonging to a cluster are brought together under a single label so that the only possible distinction is between outliers and non-outliers and consequently the supervised analysis will correspond to a **binary classification**. This approach is obviously not necessary in all those cases where the clustering phase groups all the elements into a single cluster, as there are no different labels to unify and the classification is already binary.

Once the experiments were defined and performed, the classification results were explored. To do this, three different indicators have been calculated: **Accuracy**, **Precision** and **Recall**. Then, the decision trees created are observed in order to understand which features are selected first to perform the split and which are therefore most relevant to discriminate examples belonging to different classes. Next, the importance of features is also analyzed through the permutation importance values obtained from the Random Forest classifier. In the event that the results

in analysis are obtained without the use of context features, these are exploited to display their distribution associated with the trips, separately for each cluster, through the use of violin graphs. Eventually, these distributions are compared to see how weather features could motivate separation when they do not contribute to it. Lastly, even the Pearson Correlation between common features and context features is observed through the use of heatmaps.

6.3 Driving Behaviour Evaluation

Once the data coming from clustering has been thoroughly explored, it is necessary to proceed with the construction of a model that allows to assign a KPI value to new trips. The elements considered as a reference for the evaluations are, as previously mentioned, all *Connected* trips carried out during the first three weeks of the month (from 01-06-2020 to 22-06-2020). All *Business* trips present in this period of time are not necessary, therefore they are excluded from the analyzes. The examples on which the evaluation is carried out are all the trips carried out in the last week of the month (from 23-06-2020 to 30-06-2020). In this case, both *Connected* trips and *Business* trips are used in the evaluations, none excluded.

After preparing all the data groups mentioned above, a KDE function is applied to the training set twice in order to associate a numeric value to each point of the space. The first time is applied to the multidimensional space which also includes all the selected features originating from the CAN device. The result of this application will then be used to evaluate the *Connected* trips belonging to the test set. The second KDE function is instead calculated by excluding the CAN features to make sure that the mapped space is equivalent to the *Business* trips one and consequently the result will be used to evaluate that type of trips. Furthermore, the result of this second KDE execution was used to evaluate *Connected* trips as well, thus neglecting the CAN features. This allowed us to compare both KPIs obtained for this type of trip and observe how and how much the CAN features influence the evaluation.

During each single evaluation, as described in the previous chapter, it is necessary to identify the medoid closest to the element under analysis. This medoid, in addition to providing the reference score necessary to obtain the KPI value, is also used for the calculation of the penalty scores relating to each individual feature. The mechanism by which this calculation is done is as follows. The distances between the two elements considering each axis individually are measured and summed. This is equivalent to adding the differences in the values of the same feature, for each feature. Then, each single distance that contributed to the total is divided by

the total. The result obtained corresponds to the contribution in percentage form that each feature has provided to ensure that the calculated KPI deviates from the optimal value (i.e. equal to 1). The penalty score values of each feature for each example of the test set are then depicted through a heatmap.

Chapter 7

Results

This chapter shows the results in the form of images and tables. The experiments were made for all the sections described in the previous chapter. Considering the huge amount of simulations carried out, the complete analysis of a single reference route, i.e. route **R3**, will be reported below, while for the other routes only the most significant results will be discussed.

7.1 Classification Results

In the first place, all available data, both those of the vehicle and those of context, were used to train the various classifiers mentioned by exploiting the two cited validation techniques and by calculating the previously described metrics each time. Remind that in this case the outliers are just under 6% of the total elements (4 out of 70), therefore the dataset is very unbalanced. The results obtained are shown in the table 7.1. The terms "Class 0" and "Class -1" refer respectively to the elements belonging to the cluster and to the outliers.

Looking at the reported values in detail, we immediately notice that accuracy always achieves excellent results regardless of the classification algorithm and validation technique used, but this is not relevant information given the class unbalance in the dataset. Precision and Recall for cluster elements class are excellent as imaginable. On the other hand, Precision and Recall related to the outlier class are very bad and never exceed 50.00%: this means that at most two outliers out of four are correctly classified. The worst case is that represented by the KNN and SVM algorithms in which a Precision of 100.00% and a Recall equal to 0.00% for outlier class indicate that the classifier labels all the examples as belonging to cluster elements class.

	Classifier	Accuracy	Precision Class 0	Recall Class 0	Precision Class -1	Recall Class -1
KFCV	KNN	94.28%	94.28%	100.00%	0.00%	0.00%
	SVM	94.28%	94.28%	100.00%	0.00%	0.00%
	DT	92.89%	96.92%	95.45%	40.00%	25.00%
	RF	94.19%	95.59%	98.48%	50.00%	50.00%
LOO	KNN	94.28%	94.28%	100.00%	0.00%	0.00%
	SVM	94.28%	94.28%	100.00%	0.00%	0.00%
	DT	91.42%	95.45%	95.45%	25.00%	25.00%
	RF	92.85%	95.52%	96.97%	33.33%	25.00%

Table 7.1: Results obtained for route R3 considering the context features and not rebalancing the data.

This route is one of the most unbalanced in terms of classes, but also in the other routes with a higher number of outliers, about 10, we can observe the presence of the problem. The metrics for the majority class are always excellent, thus confirming good accuracy, while those for the minority class commonly obtain results between 60% and 80% and rarely around 90%. The more outliers are there, and therefore the dataset is balanced, the more the behaviour just described gets thinner.

Trying to solve the problem of class imbalance, using the oversampling technique mentioned in the previous chapters, we obtain the results shown in the table 7.2.

	Classifier	Accuracy	Precision Class 0	Recall Class 0	Precision Class -1	Recall Class -1
KFCV	KNN	96.98%	100.00%	93.93%	94.28%	100.00%
	SVM	100.00%	100.00%	100.00%	100.00%	100.00%
	DT	94.73%	96.82%	92.42%	92.75%	96.96%
	RF	97.72%	100.00%	95.45%	95.65%	100.00%
LOO	KNN	97.72%	100.00%	95.45%	95.65%	100.00%
	SVM	100.00%	100.00%	100.00%	100.00%	100.00%
	DT	93.93%	95.31%	92.42%	92.65%	95.45%
	RF	96.96%	98.43%	95.45%	95.58%	98.48%

Table 7.2: Results obtained for route R3 considering the context features and rebalancing the data.

This strategy completely solves the problem, as in each simulation we obtain excellent Precision and Recall values for outlier class, all above 92.00%, without worsening those for cluster elements class. This also leads to a slight improvement in

Accuracy values which were already very good. Worthy of note is the SVM classifier that succeeds in perfectly classify all the samples in the dataset regardless of the validation technique used. Also in all the other sections, there is a visible performance improvement, especially for the metrics relating to the class of outliers which obtain a score never below 90%.

Returning to the original dataset, therefore without having performed an oversampling technique, we analyze the features that have contributed most to correctly classify the elements. Remembering that there are few outliers, it is reasonable to think that the features needed to separate elements based on their label are also few. In figure 7.1 we can observe the Permutation Importance of the 10 most relevant features of the Random Forest classifier. The plot represents for each feature the importance values obtained by repeating the permutations of the values 15 times. The white block runs from the lower quartile to the upper quartile. The yellow bar indicates the median. The whiskers extend to cover all the values obtained.

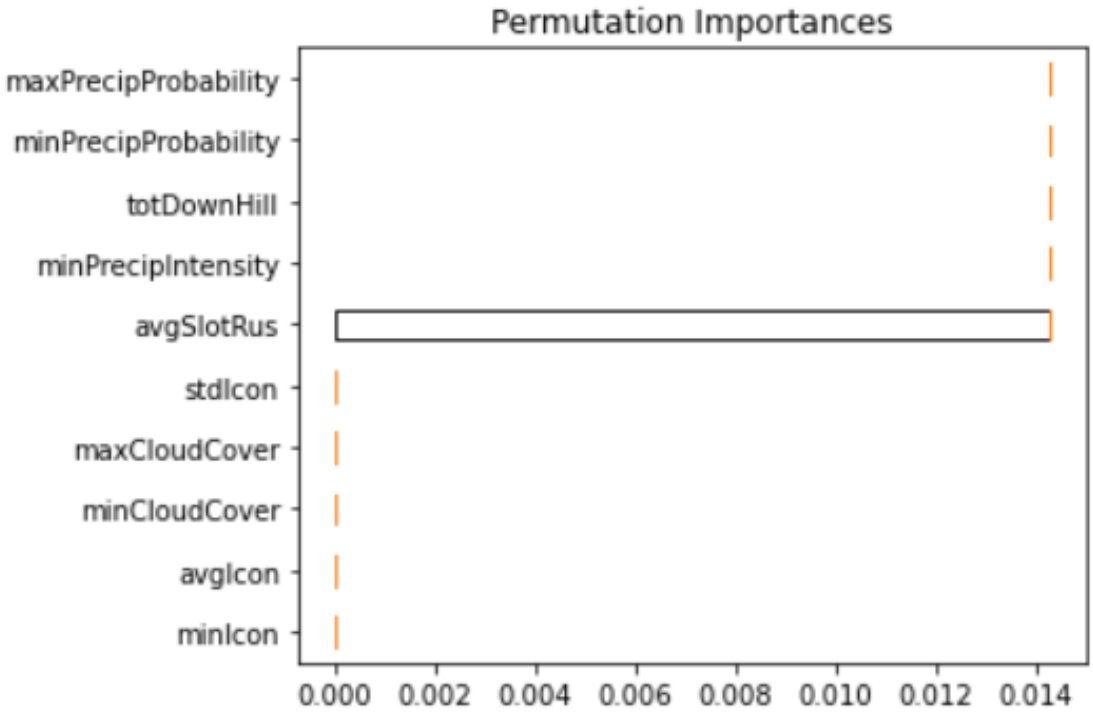


Figure 7.1: Permutation importance of the Random Forest classifier for the route R3 considering the context features.

We note that there are only 5 features whose permutation of values has led to changes to the classification, one of which is only in part of the permutations. More interesting is the fact that the most relevant features are all contextual, such

as the probability and intensity of precipitation.

Observing also the structure of the constructed DT, represented in figure 7.2, it is possible to notice that the features used to correctly classify all the examples are again few, only 2, and contextual. Furthermore, they are two of the most relevant features of the permutation importance calculated earlier.

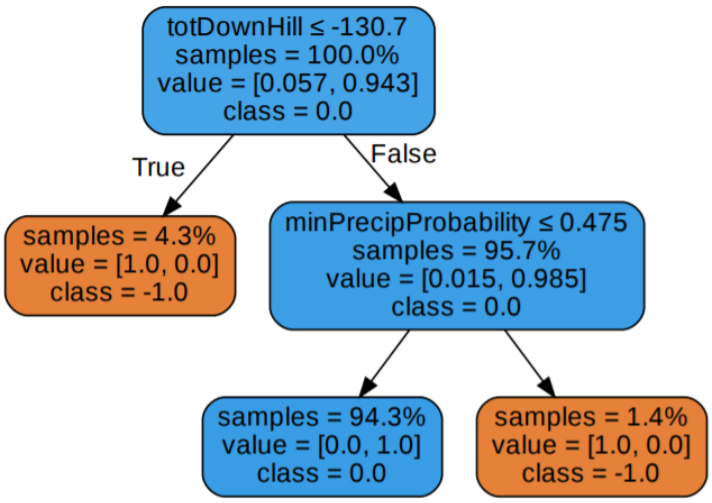


Figure 7.2: Decision Tree graphical representation built based on data of the route R3 and considering the context features.

Also in the other routes, the context features are particularly highlighted. In routes **R1** and **R4**, whose trips connect the same two cells, the features identified as significant are exclusively contextual ones and they are also equal between permutation importance and decision tree. In the **R2**, **R5** and **R6** routes, which also share the departure and arrival cells, the features considered relevant are both contextual or coming from the CAN bus and about half of them appear in both permutation importance and decision tree. It was this result that led us to remove the context features from the analyzes to use them later to describe and motivate the clusters obtained using only the features coming from the vehicles.

The results obtained without using the context features are shown in the table 7.3. Remind that in this case the outliers are about 21% of the total elements (15 out of 70), therefore the dataset is still unbalanced but less than in the previous case. Also in this case excellent accuracy is obtained and the SVM classifier seems to be slightly better than the others. The metrics related to the single class are good except for the Recall for the minority class: this can be interpreted as saying that all the examples of the majority class are correctly classified (Recall for cluster

	Classifier	Accuracy	Precision Class 0	Recall Class 0	Precision Class -1	Recall Class -1
KFCV	KNN	91.42%	90.16%	100.00%	100.00%	60.00%
	SVM	98.57%	98.21%	100.00%	100.00%	93.33%
	DT	97.14%	96.49%	100.00%	100.00%	86.66%
	RF	95.71%	94.82%	100.00%	100.00%	80.00%
LOO	KNN	91.42%	90.16%	100.00%	100.00%	60.00%
	SVM	98.57%	98.21%	100.00%	100.00%	93.33%
	DT	97.14%	96.49%	100.00%	100.00%	86.66%
	RF	94.28%	93.22%	100.00%	100.00%	73.33%

Table 7.3: Results obtained for route R3 not considering the context features and not rebalancing the data.

elements class and Precision for outlier class both 100%) while those of the class minority are not (low Recall for outlier class). All this is presumably due to the amount of outlier slightly higher than the previous case and not to the change in the features used. Once again we apply the SMOTE oversampling technique to try to improve the situation. The results thus obtained are displayed in the table 7.4.

	Classifier	Accuracy	Precision Class 0	Recall Class 0	Precision Class -1	Recall Class -1
KFCV	KNN	95.45%	91.66%	100.00%	100.00%	90.91%
	SVM	100.00%	100.00%	100.00%	100.00%	100.00%
	DT	98.18%	100.00%	96.36%	96.49%	100.00%
	RF	96.36%	94.73%	98.18%	98.11%	94.54%
LOO	KNN	96.36%	93.22%	100.00%	100.00%	92.72%
	SVM	100.00%	100.00%	100.00%	100.00%	100.00%
	DT	100.00%	100.00%	100.00%	100.00%	100.00%
	RF	97.27%	94.82%	100.00%	100.00%	94.54%

Table 7.4: Results obtained for route R3 not considering the context features and rebalancing the data.

The performance increment occurs again for all calculated metrics. SVM and DT obtain excellent results succeeding in perfectly classify all test examples. In particular, in the case of balanced classes, SVM confirms itself as the best classifier on all the sections analyzed, almost always managing to obtain 100% accuracy.

Returning to the original dataset, therefore without the application of oversampling, let's analyze once again the permutation importance relating to the RF classifier.

The result are shown in figure 7.3.

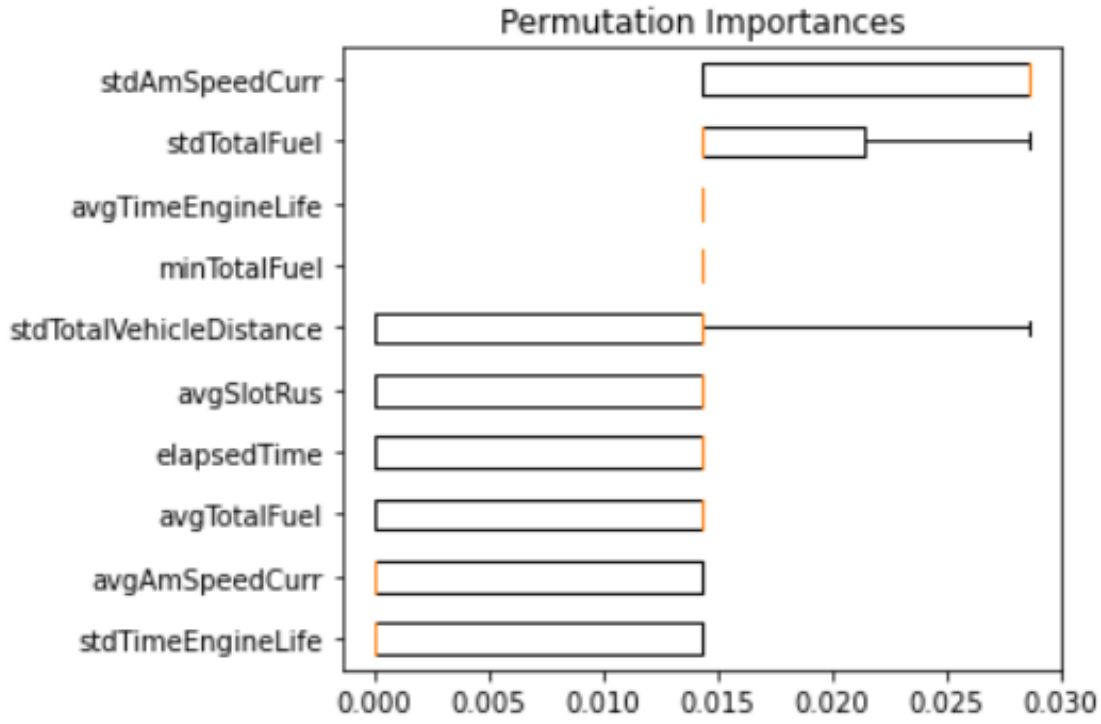


Figure 7.3: Permutation importance of the Random Forest classifier for the route R3 not considering the context features.

This time we can notice that the amount of features that appear to be relevant following the permutations are greater than in the previous case. These include both features coming from the CAN device and features included in the *Business* service, without a clear predominance of one of the two categories. In particular, fuel consumption indicators often appear in 3 of the top 10 positions in terms of importance.

Observing again the structure of the constructed DT, represented in figure 7.4, we see that it has a greater number of intermediate nodes concerning the previous case. The cause of this increase in the number of splits performed does not depend on the lack of context features but probably due to the peculiarity of some of the outliers to be isolated. The first split, the one at the root, manages to correctly separate most of the outliers while the two subsequent splits extract from the group only one outlier each, presumably different from those isolated previously. The features used all appear in the top 10 highlighted by the permutation importance in the previous figure and one of this is again related to fuel consumption.

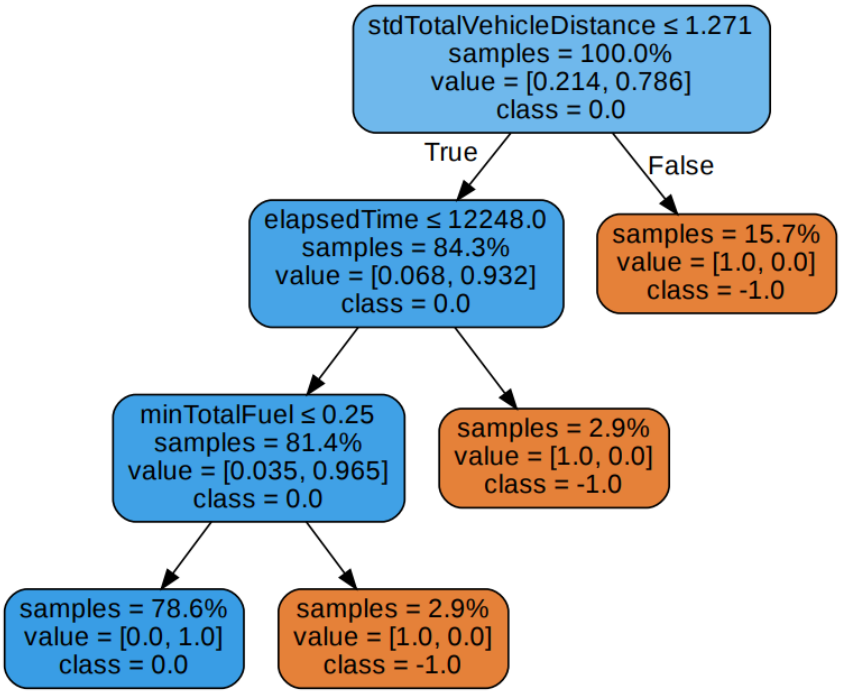


Figure 7.4: Decision Tree graphical representation built based on data of the route R3 and not considering the context features.

As for the other sections, the classification is not possible in **R1** because it has only one outlier. In **R2**, both permutation importance and decision tree indicate distance traveled as the main feature. In **R4** the fuel consumption coupled with the elapsed time is highlighted again. The **R5** and **R6** routes share the same decision tree that mainly makes use of the distance traveled, as happens for the **R2** route similar to them, while the permutation importance highlights in addition to the distance also the 5-minute breaks for both routes and the fuel consumption only for **R5**.

Since the results from clustering were obtained without using context features, we conducted an in-depth analysis to find out if and how a particular weather or altitude feature describes one of the two clusters and differentiates it from the other. To do this, violin graphs were used to represent the distribution of a single feature separately for each cluster. The meteorological features involved in this exploration are only the averages of each indicator, therefore maximums, minimums and standard deviations have been excluded. The obtained results for the 4 features which distribution differs most between clusters are shown in figure 7.5.

It is clearly seen that distributions of the outlier class contain values far outside

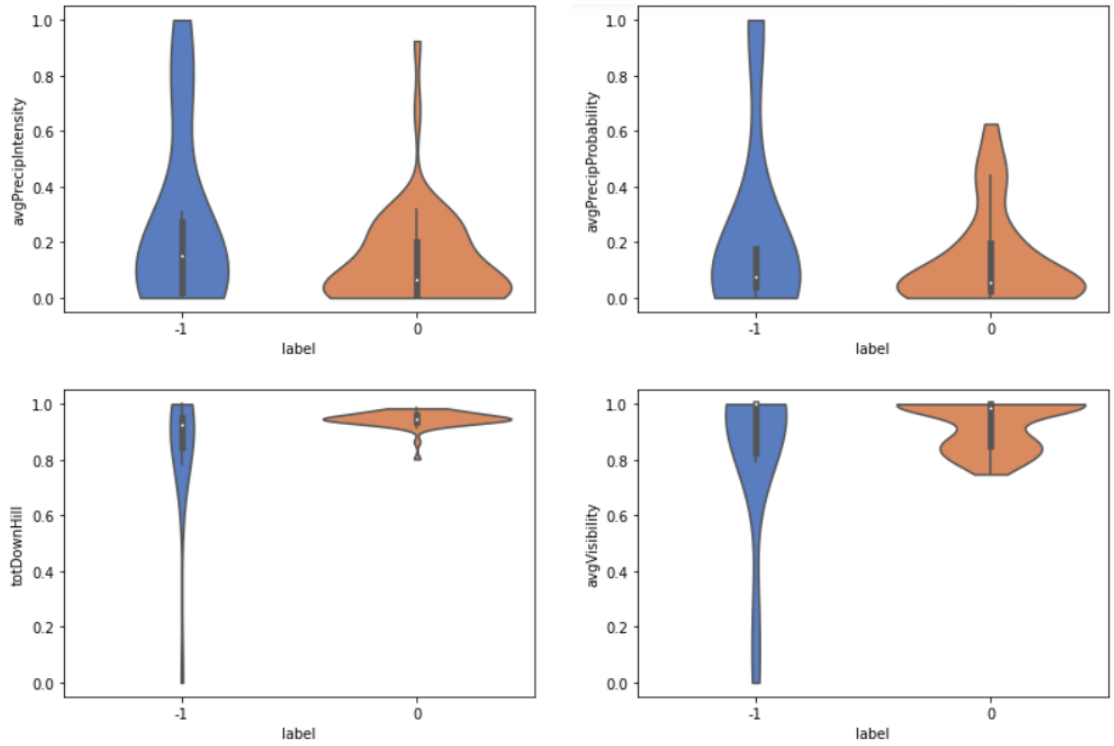


Figure 7.5: Violin charts showing how the values of the context features are distributed within each cluster.

those of the cluster elements. In fact, some of the trips considered outliers are characterized by the presence of rainfall and poor visibility, factors that could justify the anomalies found in the other vehicular features and which led to their classification as outliers. As regards different ascent and descent values, these indicate that an alternative route has been traveled and obviously all the measurements made by the vehicle have been influenced by it. Different distributions between clusters characterize most of the features in all the analyzed sections, in particular, we often see the visibility and intensity of rainfall.

If we intend to observe instead how the individual context features are linked to some of the features coming from the vehicle, such as times, speeds, distances and CAN features, the heatmap in the figure 7.6 shows the Pearson correlation that links the pairs of features. This allows us to find out if and how much the value of a context feature affects that of a base feature.

As anticipated, we note a strong correlation between the difference in altitude that characterized the route and distance, speed and travel time features from the vehicle. This is because some trips have chosen an alternative route with a

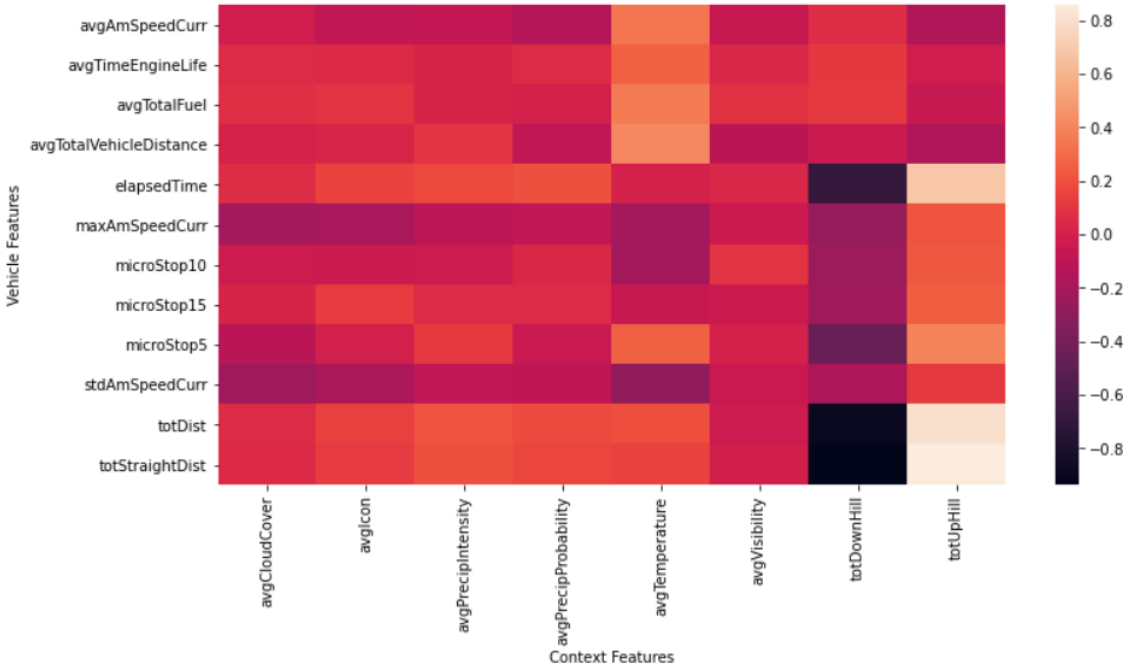


Figure 7.6: Heatmap showing Pearson's correlation between context and vehicle features.

completely different conformation. Observing better we note that there is a slight correlation also between altitude and pauses carried out and the speed. In the case of *totDownHill* the correlation is negative because this feature only has negative values. This behaviour is present in all the other sections analyzed, even if in some cases the correlation is inverted, as in **R2**, **R5** and **R6**.

Another important fact to note is the slight negative correlation between speed and weather features. This means, as imaginable, that the worse the weather, the lower the speed of a trip, to the point of being able to exclude it from the cluster. This happens for each route analyzed and sometimes the speed features are not the only ones affected by the weather.

Furthermore, there also seems to be a slight correlation between the temperature and some of the vehicular features. This type of correlation is not reflected in the other sections and is probably coincidental, especially considering that the data analyzed relates to June and the temperature should not be a critical indicator.

7.2 KPI Computation Results

Once the clustering results have been analyzed, it is time to exploit them to calculate a KPI, or a numerical score, which indicates the correctness level of a new journey outside the cluster. After removing the outliers and calculating the KDE function for each point in space we can proceed with the evaluation of all trips in the test set - i.e. those made in the last week of the month. Among these, we have 43 Connected trips and 43 Business trips. Then, also the penalty score for each feature of each trip is calculated. All the penalty scores have been represented as a heatmap that shows the percentage of negative contribute associated with each feature and each row is flanked by a label that indicates the associated trip information and KPI value. The result can be seen in figure 7.7 for *Connected* trips and in figure 7.8 for *Business* trips.

Observing the KPIs we notice a wide range of values, with trips that are close to perfection (around 95% correctness) and others that turn out to be bad (with ratings of less than 20%). Most of the trips are in any case above 80% therefore the proportion between good trips and possible outliers is consistent with the one identified by clustering. If we look at the heatmaps we can identify several particular and well-defined situations. The best *Connected* trip tests fail to obtain the maximum KPI mainly due to the features relating to the sky coverage and the climatic trend of the day. These features affect a large part of trips. If we look instead at the group of the worst *Connected* trips, we see that the cause of the low score obtained are the features relating to elapsed time, space traveled and height difference crossed. This is a clear sign that these trips have taken an alternative route to the reference ones and obviously have been evaluated as very incorrect. Moving on to *Business* trips, the best ones do not reach perfection due to the speed features that differ from the values of the cluster medoid, while for trips with low ratings the situation is similar to *Connected* trips. Eventually, we can notice some cases in which there is a single feature that stands out among those that differ from the medoid ones. However, looking at the scale it can be seen that there is no case in which a single feature contribute more than 25% to the total penalty score.

In addition to the KPIs shown in the heatmaps, a second KPI has been calculated for *Connected* trips that do not consider CAN features. This is equivalent to simulating an evaluation for a *Business* type trip. The resulting KPI is called **Truncated KPI**. These values are then compared with the original KPIs to evaluate how much the CAN features actually impact the final score. This comparison is shown in the table 7.5. Having excluded the CAN features does not involve significant changes to the KPI values. The small differences that are created can

correspond to both improvements and worsening of the score and even in the most extreme cases these do not exceed 15-16%. This fact was conceivable if we consider the analyzes carried out previously which affirm the great importance of context features in determining the difference between similar trips.

Based on this, the KPI computation it repeated even without taking into account the context features and the results obtained are shown in the figure 7.9 and 7.10.

Again the majority of the trips achieve a correctness indicator above the 80% and this time we also have two trips who have been able to obtain a perfect evaluation (i.e. maximum quality regarding the driving behaviour), in fact for them no penalty score values are indicated. We have a 0 value KPI, which means that a trip is located outside the range of influence of the computed KDE. Moreover, as it happens with context features, most of the trip, both *Connected* and *Business*, with a relative low KPI has been influenced by an alternative path, as the penalty scores for space and time traveled suggest. Regarding the best trips that slightly differ from the perfect KPI, the features that have had the greatest impact are those relating to the time slot, for both *Connected* and *Business*, and to speed, the maximum one for *Connected* and the standard deviation for *Business*. We can note once again the presence of some isolated cases where the penalty score of a single feature far exceeds that of all the others, but this time the scale is wider compared to the case that includes the context feature and these features contribute up to 40% of the overall decrease in the KPI. This fact is justified by the lack of a large number of features, those of context, which previously took on part of the blame.

Eventually, the comparison between the two KPI computed for the *Connected* trips is repeated. The removal of CAN features does not cause significant changes in trips with a high KPI value, while for those at the bottom of the ranking they are strongly influenced. In particular, the removal of the CAN features causes marked improvements in the value of the KPIs. This happens because you lose track of all those incorrect behaviours that are recorded by the sensors of the CAN device, thus causing the wrong attribution of a good evaluation to some trips. This demonstrates the relevance of CAN indicators, the correct use of them in this kind of analysis and the advantage that customers obtain by taking advantage of the *Connected* service.

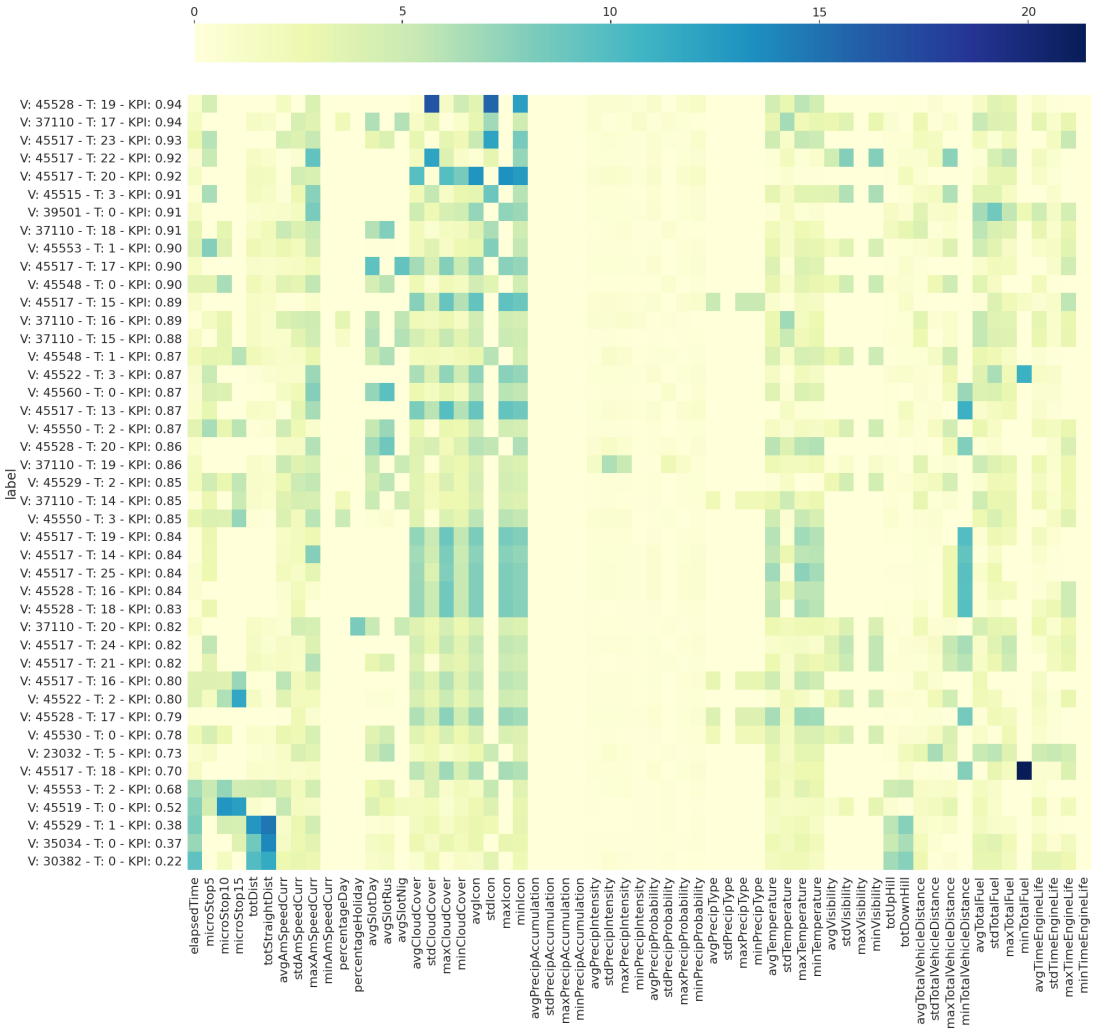


Figure 7.7: Heatmap that represents KPI values for each *Connected* trip and the penalty score of each feature, obtained considering context features.

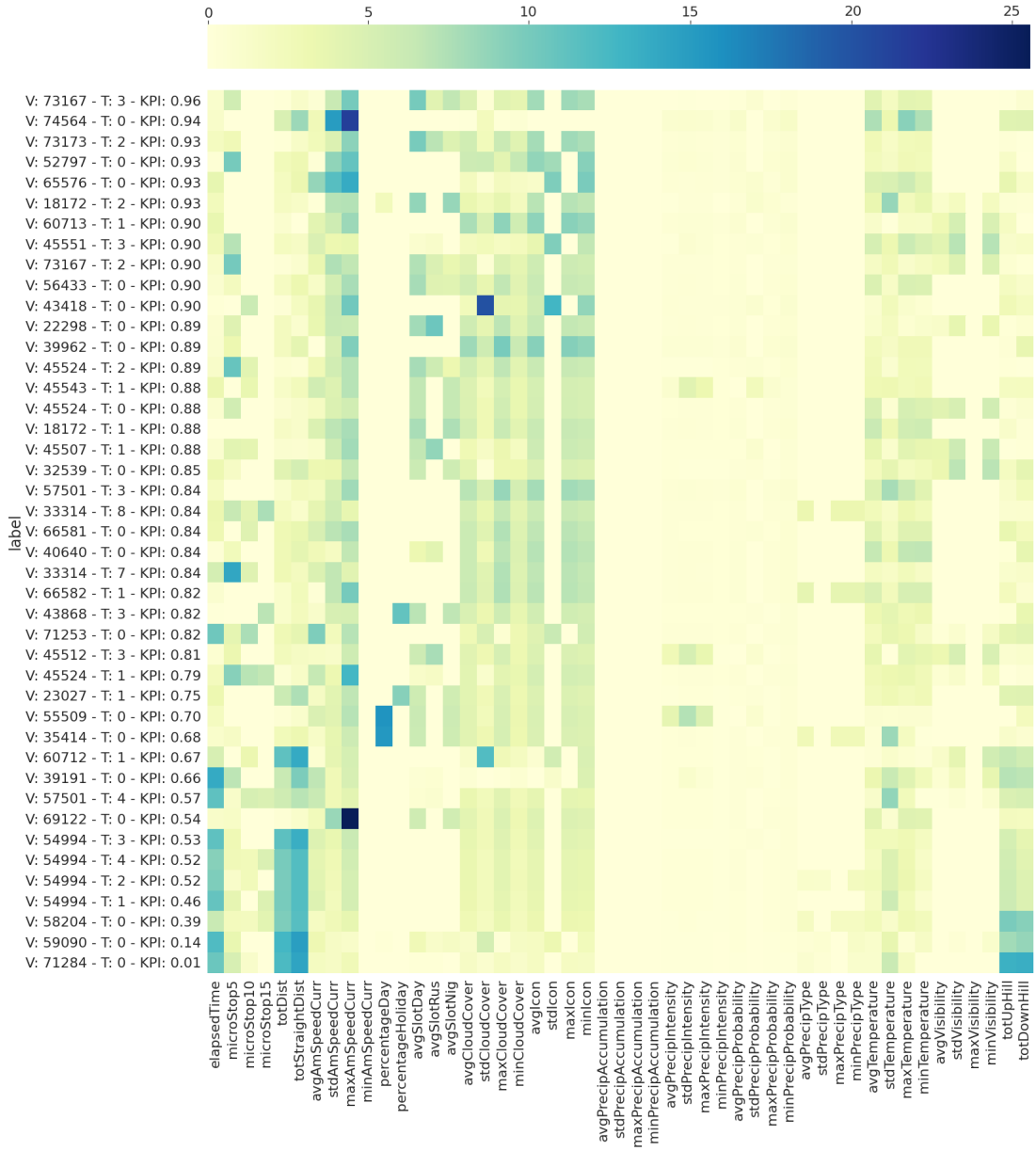


Figure 7.8: Heatmap that represents KPI values for each *Business* trip and the penalty score of each feature, obtained considering context features.

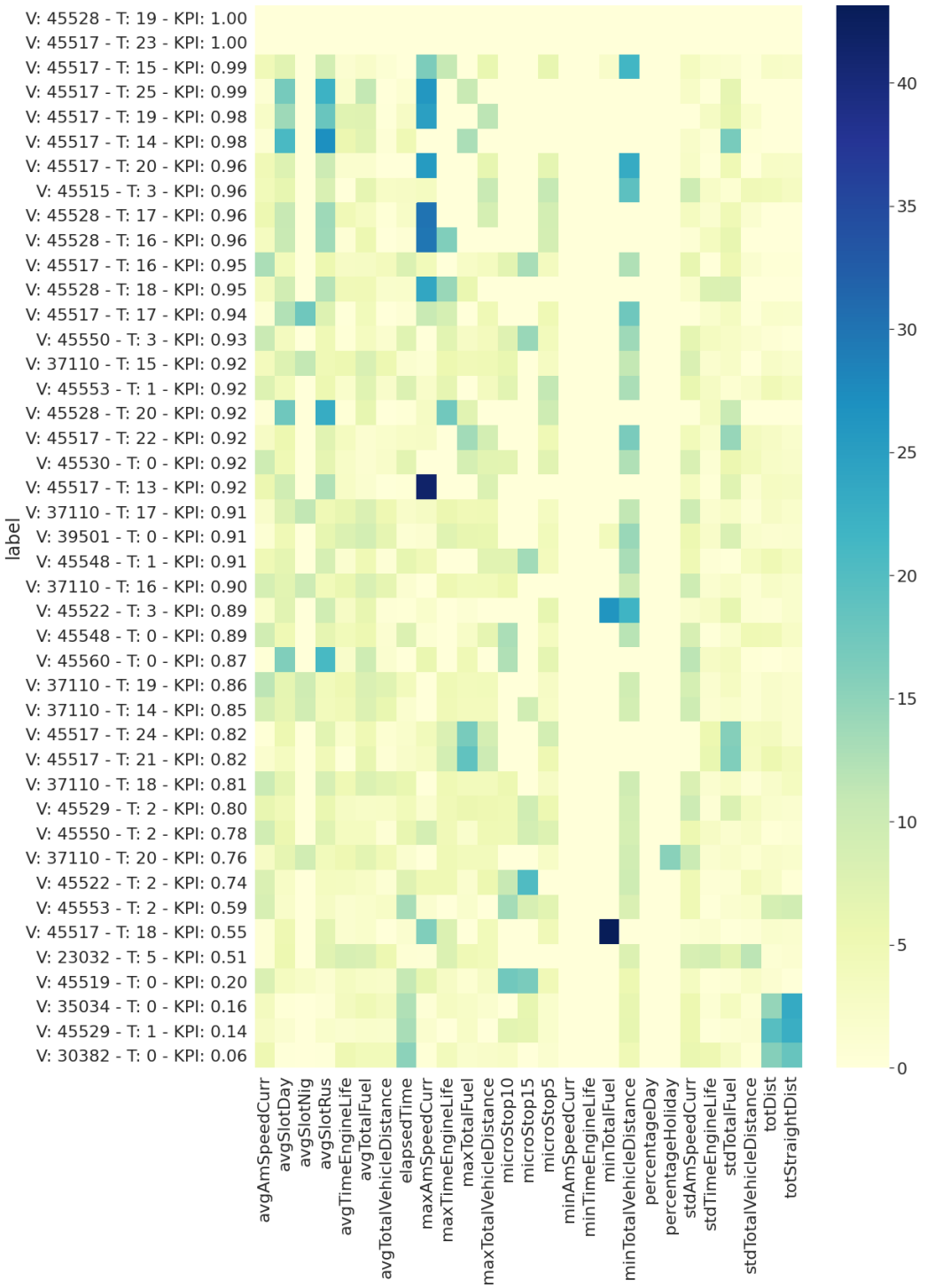


Figure 7.9: Heatmap that represents KPI values for each *Connected* trip and the penalty score of each feature, obtained without considering context features.

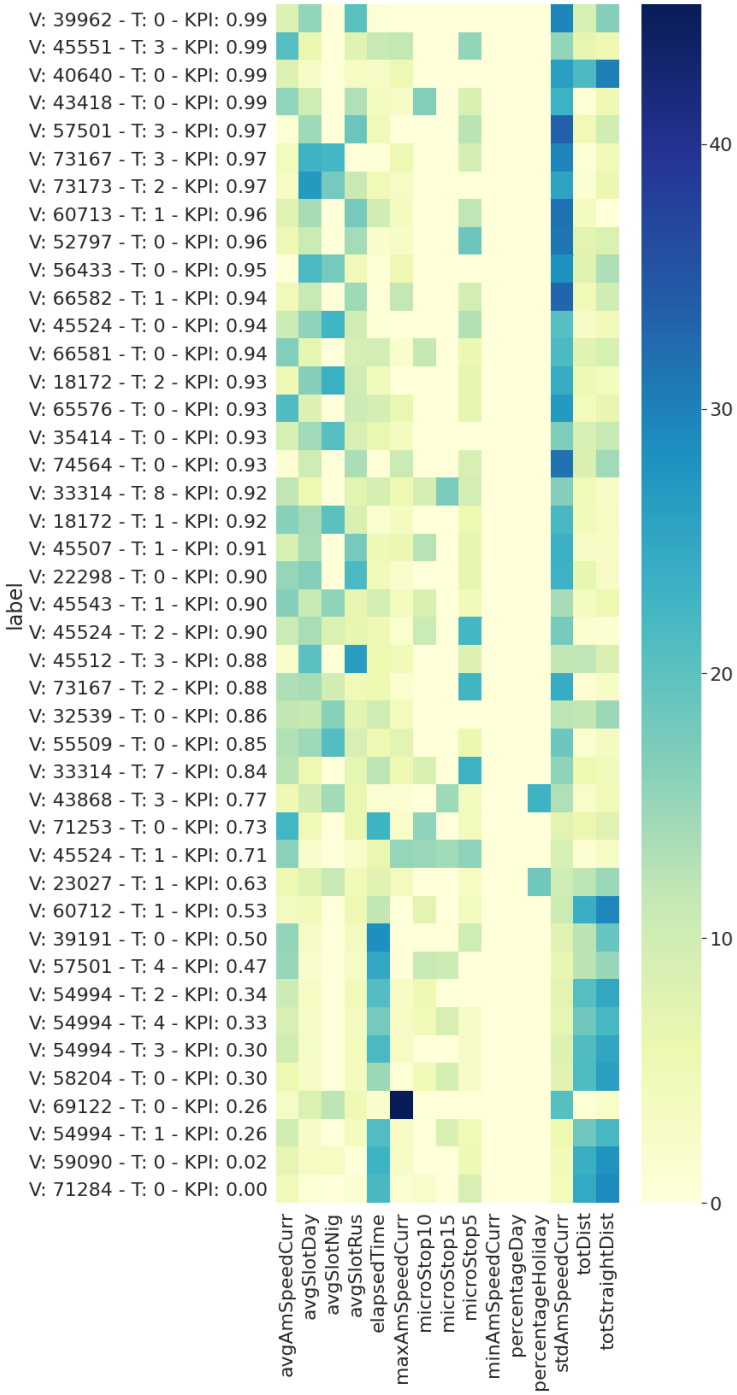


Figure 7.10: Heatmap that represents KPI values for each *Business* trip and the penalty score of each feature, obtained without considering context features.

KPI Comparison			
KPI	**Truncated KPI**	**KPI**	**Truncated KPI**
0.94	0.94	...	
0.93	0.94	0.79	0.79
0.92	0.93	0.79	0.80
0.91	0.94	0.78	0.78
0.91	0.91	0.72	0.85
0.91	0.90	0.70	0.86
0.90	0.93	0.67	0.66
0.90	0.92	0.52	0.52
0.90	0.91	0.37	0.37
0.89	0.89	0.36	0.37
...		0.22	0.24

Table 7.5: Comparison between the KPIs of *Connected* trips that include and not include the CAN features, always considering context feature.

KPI Comparison			
KPI	**Truncated KPI**	**KPI**	**Truncated KPI**
1.00	1.00	...	
1.00	1.00	0.77	0.80
0.99	0.99	0.76	0.78
0.99	0.99	0.73	0.74
0.98	0.99	0.59	0.57
0.98	0.98	0.54	0.96
0.96	0.95	0.50	0.88
0.96	0.97	0.19	0.20
0.96	0.96	0.15	0.17
0.95	0.97	0.13	0.13
...		0.05	0.07

Table 7.6: Comparison between the KPIs of *Connected* trips that include and not include the CAN features, always not considering context feature.

Chapter 8

Conclusions and Future Works

The work described is part of a framework for the evaluation of the driving behaviour of vehicles belonging to a fleet. This framework exploits an innovative approach based on the behaviuor adopted by most pilots and on the similarity between trips, instead of a more classic solution based on the direct involvement of a domain expert in charge of validation. The main objective of this thesis was the analysis of the data coming out of the preprocessing phase and the search for the final strategy to be used to calculate the required KPIs.

The reported results demonstrate how multiple classifiers can easily predict whether a trip is worthy of being used as a reference for evaluating other trips, despite any class unbalancing problems, basing their training on the labels assigned during a preliminary clustering phase. This can be useful if we are interested in increasing the number of examples in the reference set used for the evaluation, using a model trained by a supervised algorithm instead of rerunning the unsupervised learning phase. However, the clustering phase is still useful as it is the only way we have to identify new clusters, different from those already known. Given the small number of similar trips - i.e. belonging to the same route - currently available, this could easily happen in the future. For example, more than one group, each related to different behaviours, could be formed, as documented in the literature, or a group related to the best driving behaviour could be formed for each weather situation. The latter hypothesis could happen regardless of whether or not the context features are used. In-depth analyzes have confirmed that these features have a strong impact on the results when used. When not considered during training they can still be used to justify the results obtained, because the correlation with some of the

vehicular features has been demonstrated.

Regarding the computation of the final KPIs, this work is an example of the usage of a density estimation function applied to tabular data to assign an evaluation to new unseen samples based on similarities with input ones. Starting from a situation in which the data is not labeled and there is no validation provided by a domain expert, this allows reducing the complex phases that constitute the framework to a simple numerical value. Although not directly validated, these KPIs benefit from a sufficient degree of reliability based on the hypothesis that the trips analyzed are performed by expert pilots of several transport companies who know-how and must behave well. Moreover, in some cases, the vehicles of the *Connected* service can detect anomalies that vehicles not equipped with a CAN device could not identify. This is potentially actionable for marketing purposes.

This study could be further investigated, especially about the data analyzed. The number of trips that belong to the same route is very limited. A larger amount would allow for more robust results. This problem will be solved over time thanks to the constant and continuous collection of data and the spread of monitoring devices on board vehicles. Alternatively, aggregating trips of similar routes might be an acceptable solution. Another problem is the absence or scarcity of some information known to be related to driving behaviuor. For example, the CAN bus is capable of monitoring engine revolutions but transmits this information only once a minute, a much lower frequency than that with which the monitored value varies. There are many other useful indicators whose presence in the database is not sufficient. The introduction of this information would certainly bring added value to the analyzes, because this data could make the results more accurate.

Bibliography

[1] C. Ferrozzi and S. Roy, *Dalla logistica al supply chain management*. IlSole24Ore, 2001.

[2] T. G. Crainic and G. Laporte, *Fleet management and logistics*. Springer Science & Business Media, 2012.

[3] *K-Master website*. [Online]. Available: https://www.telepasskmaster.com/.

[4] D. J. Hand and N. M. Adams, "Data Mining", in *Wiley StatsRef: Statistics Reference Online*. American Cancer Society, 2015, pp. 1–7, ISBN: 9781118445112. DOI: https://doi.org/10.1002/9781118445112.stat06466.pub2.

[5] C. C. Aggarwal, "Data Classification", in *Data Mining: The Textbook*. Cham: Springer International Publishing, 2015, pp. 285–344, ISBN: 978-3-319-14142-8. DOI: 10.1007/978-3-319-14142-8_10.

[6] B. V. Dasarathy, "Nearest neighbor (NN) norms : NN pattern classification techniques", *IEEE Computer Society Tutorial*, 1991.

[7] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support Vector Machines", *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998. DOI: 10.1109/5254.708428.

[8] L. Rokach and O. Maimon, "Decision Trees", in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA: Springer US, 2005, pp. 165–192, ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_9.

[9] L. Breiman, "Random Forests", 2001. DOI: 10.1023/A:1010933404324.

[10] M. Stone, "Cross-Validatory Choice and Assessment of Statistical Predictions", *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, pp. 111–133, 1974. DOI: https://doi.org/10.1111/j.2517-6161.1974.tb00994.x.

[11] X. Zeng and T. R. Martinez, "Distribution-balanced stratified cross-validation for accuracy estimation", *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 1, pp. 1–12, 2000. DOI: 10.1080/095281300146272.

[12] M. Hossin and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations", *International Journal of Data Mining Knowledge Management Process*, vol. 5, pp. 01–11, Mar. 2015. DOI: 10.5121/ijdkp.2015.5201.

[13] V. García, J. Sánchez, and R. Mollineda, "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance", *Knowledge-Based Systems*, vol. 25, no. 1, pp. 13 –21, 2012, Special Issue on New Trends in Data Mining, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2011.06.013.

[14] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets", *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, pp. 42–47, Jan. 2012.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique", *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. DOI: 10.1613/jair.953.

[16] B. Silverman, *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics & Applied Probability, London: Chapman and Hall, 2005.

[17] G. R. Terrell and D. W. Scott, "Variable Kernel Density Estimation", *The Annals of Statistics*, vol. 20, no. 3, pp. 1236–1265, 1992, ISSN: 00905364.

[18] B. A. Turlach, "Bandwidth Selection in Kernel Density Estimation: A Review", in *CORE and Institut de Statistique*.

[19] D.-G. for Communication (European Commission), *Trasporti: Collegare i cittadini e le imprese dell'Europa*. 2014, ISBN: 978-92-79-42785-5. DOI: 10.2775/14166.

[20] J.-H. Hong, B. Margines, and A. K. Dey, *A Smartphone-Based Sensing Platform to Model Aggressive Driving Behaviors*, ser. CHI '14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, 4047–4056, ISBN: 9781450324731. DOI: 10.1145/2556288.2557321.

[21] D. Hari, C. J. Brace, C. Vagg, J. Poxon, and L. Ash, *Analysis of a Driver Behaviour Improvement Tool to Reduce Fuel Consumption*. 2012, pp. 208–213. DOI: 10.1109/ICCVE.2012.46.

[22] O. Taubman-Ben-Ari, M. Mikulincer, and O. Gillath, "The multidimensional driving style inventory—scale construct and validation", *Accident Analysis Prevention*, vol. 36, no. 3, pp. 323 –332, 2004, ISSN: 0001-4575. DOI: 10.1016/S0001-4575(03)00010-1.

[23] Y. Ma, Z. Zhang, S. Chen, Y. Yu, and K. Tang, "A Comparative Study of Aggressive Driving Behavior Recognition Algorithms Based on Vehicle Motion Data", *IEEE Access*, vol. 7, pp. 8028–8038, 2019. DOI: 10.1109/ACCESS.2018.2889751.

[24] J. V. Mierlo, G Maggetto, E. V. de Burgwal, and R Gense, "Driving style and traffic measures-influence on vehicle emissions and fuel consumption", *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 218, no. 1, pp. 43–50, 2004. DOI: 10.1243/095440704322829155.

[25] U. Kiencke, S. Dais, and M. Litschel, "Automotive Serial Controller Area Network", Feb. 1986. DOI: 10.4271/860391.

[26] L. Eboli, G. Guido, G. Mazzulla, G. Pungillo, and R. Pungillo, "Investigating Car Users' Driving Behaviour through Speed Analysis", *PROMET - Traffic-Transportation*, vol. 29, p. 193, Apr. 2017. DOI: 10.7307/ptt.v29i2.2117.

[27] J. Guo, Y. Liu, L. Zhang, and Y. Wang, "Driving Behaviour Style Study with a Hybrid Deep Learning Framework Based on GPS Data", *Sustainability*, vol. 10, no. 7, 2018, ISSN: 2071-1050. DOI: 10.3390/su10072351.

[28] I. Carpatorea, S. Nowaczyk, T. Rögnvaldsson, M. Elmer, and J. Lodin, "Learning of Aggregate Features for Comparing Drivers Based on Naturalistic Data", pp. 1067–1072, 2016. DOI: 10.1109/ICMLA.2016.0194.

[29] I. Carpatorea, S. Nowaczyk, T. Rognvaldsson, and M. Elmer, "APPES Maps as Tools for Quantifying Performance of Truck Drivers", Jan. 2014. DOI: 10.13140/2.1.1152.4167.

[30] I. Carpatorea, S. Nowaczyk, T. Rögnvaldsson, and J. Lodin, "Features extracted from APPES to enable the categorization of heavy-duty vehicle drivers", pp. 476–481, 2017. DOI: 10.1109/IntelliSys.2017.8324336.

[31] J. Grengs, X. Wang, and L. Kostyniuk, "Using GPS Data to Understand Driving Behavior", *Journal of Urban Technology*, vol. 15, pp. 33–53, Aug. 2008. DOI: 10.1080/10630730802401942.

[32] Q. C. Sun, R. Odolinski, J. C. Xia, J. Foster, T. Falkmer, and H. Lee, "Validating the efficacy of GPS tracking vehicle movement for driving behaviour assessment", *Travel Behaviour and Society*, vol. 6, pp. 32 –43, 2017, ISSN: 2214-367X. DOI: 10.1016/j.tbs.2016.05.001.

[33] M. Brambilla, P. Mascetti, and A. Mauri, "Comparison of different driving style analysis approaches based on trip segmentation over GPS information", pp. 3784–3791, 2017. DOI: 10.1109/BigData.2017.8258379.

[34] S. Choi, J. Kim, D. Kwak, P. Angkititrakul, and J. Hansen, "Analysis and Classification of Driver Behavior Using in-Vehicle CAN-BUS Information", *Bienn. Workshop DSP In-Veh. Mob. Syst.*, Jan. 2007.

[35] U. Fugilando, P. Santi, S. Milardo, K. Abida, and C. Ratti, "Characterizing the "Driver DNA" Through CAN Bus Data Analysis", CarSys '17, 37–41, 2017. DOI: 10.1145/3131944.3133939.

[36] J. C. Ferreira, J. de Almeida, and A. R. da Silva, "The Impact of Driving Styles on Fuel Consumption: A Data-Warehouse-and-Data-Mining-Based Discovery Process", *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2653–2662, 2015. DOI: 10.1109/TITS.2015.2414663.

[37] K. Jakobsen, S. C. H. Mouritsen, and K. Torp, "Evaluating Eco-Driving Advice Using GPS/CANBus Data", SIGSPATIAL'13, 44–53, 2013. DOI: 10.1145/2525314.2525358.

[38] U. Fugiglando, E. Massaro, P. Santi, S. Milardo, K. Abida, R. Stahlmann, F. Netter, and C. Ratti, "Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment", *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 737–748, 2019. DOI: 10.1109/TITS.2018.2836308.

[39] A. M. Taha and N. Nasser, "Utilizing CAN-Bus and smartphones to enforce safe and responsible driving", pp. 111–115, 2015. DOI: 10.1109/ISCC.2015.7405502.

[40] J. Wu, Y. Du, G. Qi, and M. Xu, "Leveraging longitudinal driving behaviour data with data mining techniques for driving style analysis", *IET Intelligent Transport Systems*, vol. 9, Apr. 2015. DOI: 10.1049/iet-its.2014.0139.

[41] K.-T. Chen and H.-Y. W. Chen, "Driving Style Clustering using Naturalistic Driving Data", *Transportation Research Record*, vol. 2673, no. 6, pp. 176–188, 2019. DOI: 10.1177/0361198119845360.

[42] C. Marks, A. Jahangiri, and S. G. Machiani, "Iterative DBSCAN (I-DBSCAN) to Identify Aggressive Driving Behaviors within Unlabeled Real-World Driving Data", pp. 2324–2329, 2019. DOI: 10.1109/ITSC.2019.8917162.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.