POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Analysis and Development of Methods for Automotive Market Segmentation

Supervisor
prof. Paolo Garza

Candidate
Marco Angelo Guttadauria

Internship Tutor
Dott. Ing. Silvia Delsanto

Accademic year 2020-2021

Summary

In the automotive sector in recent years the use of data has become increasingly important to allow car manufacturers to analyse the market, competing vehicles, their specifications and their prices so that they could use this information to create a product offering that adapts as much as possible to the market. Some of the more significant analyses require vehicle segmentation.

Segmentation represents the division of vehicles into commercially homogeneous groups. The market analysis required by car manufacturers are typically conducted using as comparison vehicles from the same segment. This type of analysis is called basket analysis, where the basket would be the set of vehicles considered most similar to the vehicle being compared.

Jato has developed its own more general definition of segmentation and based on this subdivision catalogues the models present in the databases. The definition of the segments and the choice of the segment to be assigned to the vehicles, however, are not immediate processes.

The first part of the thesis is aimed at defining the characteristics that allow the segmentation to be carried out and to develop a tool capable of carrying out an automatic segmentation of the models that can be used as a support for the segmentation phase. The model developed is based on a random forest algorithm that allows to obtain an automatic classifier to support the choice of the segment.

The second part of the work is aimed at allowing to systematically define the vehicles that are considered similar to a given vehicle in order to be able to define more precise baskets for basket analysis.

Two similarity measures were developed to identify similar vehicles at model level and similar vehicles at version level. Given the presence of categorical and continuous features, the proposed solution is based on the Gower distance measurement that allows to manage both types of variables by combining the use of the Dice coefficient for categorical features and the Manhattan distance for continuous features.

Acknowledgements

Con queste poche righe colgo l'occasione per ringraziare tutte le persone che mi sono state vicine in questo percorso.

Desidero ringraziare il mio relatore, il Prof. Paolo Garza, per avermi seguito con competenza e disponibilità e la Dott.ssa Silvia Delsanto, per avermi guidato in questo percorso di tirocinio in Jato e per avermi aiutato a crescere con i suoi preziosi suggerimenti.

Vorrei ringraziare particolarmente la mia famiglia che mi ha sempre supportato e permesso di raggiungere questo importante obiettivo. So che in alcuni periodi non è stato facile starmi vicino ma anche in quei momenti siete stati pazienti e mi siete stati di grande supporto.

Un pensiero speciale va ai miei "colleghi", ma soprattutto amici, Salvo C., Salvo P., Andrei, Luigi, Giuseppe e Vincenzo, con cui ho condiviso questo periodo universitario fatto di alti e bassi ma sempre tra mille risate, senza di loro non sarebbe stato lo stesso.

E per ultimi, ma non per importanza, vorrei ringraziare i miei amici di una vita, in particolare Aldo, Giorgio Khaleesi, Giorgio Prì, Laura, Monica e Noemi con cui ho condiviso alcune delle esperienze più belle della mia vita. Grazie per esserci sempre anche a distanza, con voi le giornate sono sempre un pò più leggere.

P.S. quasi dimenticavo di ringraziare Ermenegildo per gli ottimi Spritz offerti.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, it has increasingly been heard that the most important assets owned by companies are the data they acquire, manage and store. Nowadays most companies have databases containing various types of data, these databases constitute a potential mine of useful information for various purposes. Analyzing this data correctly can give the company support in decision making and produce a competitive advantage for the company on the market. The type of data and the use made of it changes as the sector of interest changes. The sector of interest for this thesis is the automotive one. We will see how data is important in this area and how **Jato** plays an important role in collecting and analyzing it.

1.1 JATO role in automotive sector

During the 1980s the automotive sector was in its phase of maximum expansion. In those years the economy was changing and the concept of globalization was gaining ground and, as in all sectors, the automotive sector was also involved. Car manufacturers were starting to expand into the international market, to do this they therefore needed to learn more about the global market, competing vehicles, their specifications and their prices so that they could use this information to create a product offering that adapts as much as possible to new markets. **Jato** was therefore born to harness the power of information in this new environment by making data collection and analysis his business. Over the years It has collected various types of data to be able to provide different information and different tools to its custom.

The company plays an important role at various levels of the automotive industry. The customers to which it provides its services are of various types, in particular, **Jato** caters to automotive manufacturer, car dealerships, fleet and leasing companies, agricultural manufacturer and component suppliers. It provides these clients with tools to support their needs, in order to improve the quality of their work and to provide important competitive advantages. To better understand the importance of **Jato** in the automotive sector, here are some of the most important services and information that it provides for its customers:

- **Jato Carspecs**: it provides vehicle comparison and configuration solution which enables to see comprehensively the advantages of one vehicle against another. In terms of advantages and benefits, dealers and manufacturers can use *Carspecs* to demonstrate why a consumer should purchase their vehicle over one of their competitors.

- **Jato Volumes**: through the *ModelMix* dataset it provides access to a set of volumes statistics including prices that allows quality detailed analysis of the market trends.

- **Jato Incentives**: it provides all types of national retail incentives for vehicles. This information helps to position a vehicle competitively on price.

- **Jato V5**: analysis and benchmarking application for Automotive Manufacturers. It allows to analyse the information described in the previous points all in the same place, in order to support the sales strategies of the manufacturers' products.

- **Jato Analysis & Reporting**: It provides bespoke analysis services to individual customers using the data available to the company. These analyses will assist decision making process of clients.

Thanks to the various types of data and services, jato provides transversal support to all the players in the automotive sector and this makes it play a very important role. Before describing the data managed by *Jato* in details, it is important to describe the data in the automotive sector and their problems.

1.2 Automotive data complexity

Reproducing, in the form of data, events or objects from the real world, always presents challenges, which differ depending on the area covered. In the automotive world this process is characterized by various complexities. In particular, the representation of the vehicle and all its attributes requires a very high level of detail. Speaking of cars, an important distinction must be made between models and versions.

1.2.1 Model Vs Version and their data representation difficulties

In the automotive sector the most important entity to represent is the vehicle which can be defined at the model or version level. We can define the model as a set that contains within it a variable number of versions. Models are identified by a car's make, that is the brand of the vehicle, and a model name that refers to the name of the car product. For example, *Alfa Romeo* is the car make and *Stelvio* is the model name. Cars of the same model can vary greatly from each other and knowing the model alone is not enough to identify a vehicle. The various versions of a model differ from each other according to different characteristics, some of the main ones are as follows:

- **Body type** : refers to the shape and model of a particular automobile. Example of body type are *sedan*, *hatchback*, *SUV*
- **Trim level** : refers to the type of equipment and the style of the vehicle. For example *sporty*, *luxury* or *standard* trim.
- **Model year** : this is one of the main ways a vehicle can stand out from another of the same model. It is indicative of the period in which that specific version of the model was released
- **Engine** : refers to the power of the equipped engine.
- **Fuel type** : indicates the type of motor power supply. Example *unleaded*, *diesel*, *electric*
- **Type of traction** : it indicates the type of traction of the vehicle which can be *front*, *rear* and *integral.*

Being able to correctly represent the version of a vehicle presents complexities due to the level of detail necessary to represent it correctly and to the great variability of the data used to represent it. Each version, as mentioned, is characterized by a long series of attributes which therefore require a detailed research of the information and an equally accurate representation of the same. Sometimes it is not easy to find all the necessary information that may be missing or expressed according to a non-standard nomenclature. For example, certain characteristics such as the *trim level* are represented with values that differ from make to make making the representation of the data more complex. The great variability of the data is given by the fact that there are many characteristics that define the version and each of these can assume different values. This means that there may be many versions of the same model that differ from each other for one or more values of one or more of their many attributes. Furthermore, over time, also thanks to technological development, more and more features are added to the existing versions and features may undergo changes and assume new values. In the next paragraph we will see how *Jato* dealt with the problem of representing the data relating to vehicles. First, however, let's analyze the complexities related to another important entity in the automotive sector, the option.

1.2.2 Options, features and their data representation difficulties

Car options are add-ons for a vehicle that a buyer of a new car can choose before purchase [2]. During the purchase phase, the seller offers the buyer a series of options of various types, each of which has a value that will increase the final price of the vehicle.

Options can consist of one or more features. Each feature adds a specific functionality to the vehicle and may be present in one or more options. Examples of options with multiple features may be the technology packages offered with the purchase of the vehicle, these contain various features that improve the technological aspect, for example, by adding tools such as touchscreens or smartphone integration.

There are many features available on the market and their number is constantly increasing. There are various types of features with various purposes, there are features that increase vehicle comfort, features that make the vehicle safer or features that can improve the vehicle driveability. The features can be standard or optional, they are standard when the vehicle mounts them without the need to be added, optional when they must be added during the purchase phase and their addition increases the price of the final vehicle. Some features may be standard in some vehicle versions while in others not or some features may only be available in some versions while in others they cannot be chosen.

From what has been said, features and options are not simple entities to represent given their high variability and their need for detail, moreover the information concerning them must always be updated to cope with the continuous changes in the market. Being able to create a structure that can represent the options comprehensively is therefore a challenge.

1.3 Jato data

In this paragraph we will introduce fundamental concepts for the continuation of this thesis. First we will describe the data collection phase with its peculiarities, then we will introduce the concept of segmentation, and finally we will describe the data structure in *Jato* and their location in the databases.

1.3.1 Data collection complexity

This phase is performed by the **Jato researchers** , who have the task of describing the information regarding the vehicles and everything about them in the greatest possible detail. This information is collected with a descriptive purpose and not with the aim of being used for analysis. The collection phase presents difficulties, first of all, for researchers who often have to catalogue features that could not be available or that, depending on the market or make, have different names and definitions. In particular, the data suffer from critical issues such as the lack of values for certain vehicle features that the researchers were not able to find, or the presence of duplicated identifiers representing the same feature. The duplication of identifiers is caused by the need for researchers to express features with the greatest possible detail, leading them to represent two identical features, which have small differences, in a different way.

1.3.2 Vehicle Segmentation

Car models can be grouped into sets called **segments**. The segment defines the categorization of the car based on various factors which can be physical or market based. Segmentation represents the attempt to divide vehicles into commercially homogeneous groups. *Jato* defined his version of segmentation dividing the vehicles in the following 20 classes:

- **EU A - utility/city cars**
- **EU A SUV**

- **EU B – small**
- **EU B SUV**
- **EU C1 - lower medium -**
- **EU C1 SUV**
- **EU C2 - lower medium +**
- **EU C2 SUV**
- **EU D1 - upper medium -**
- **EU D1 SUV**
- **EU D2 - upper medium +**
- **EU D2 SUV**
- **EU E1 - large and executive**
- **EU E1 SUV**
- **EU E2 – luxury**
- **EU E2 SUV**
- **EU Large MPV**
- **EU Medium MPV**
- **EU Mini MPV**

This subdivision is not immutable over time because it must be continually adapted to changes occurring in the market. For this reason, being able to define the various segments is a critical and not easy task. The assignment of models to the various segments is even more complicated. During the vehicle data collection phase, the **Jato researchers**, in addition to entering information about the vehicles, must also enter the segment of the vehicle model. The choice of which segment to associate with each vehicle is made on the basis of the model. Whenever a new model has to be inserted in the jato databases, the researchers who have the task of inserting it must specify which segment it belongs to. The choice of which market segment is most suitable for a specific model is not at all simple because it is not totally based on objective parameters. To make the researchers' task easier, common physical characteristics were sought within the segments in order to make the assignment as objective as possible. However, the physical characteristics are not always indicative of the segment to which a vehicle belongs, for example, it may happen that the makes specify preferences regarding the segment in which to insert their model, leaving out, in part, the considerations on physical characteristics. In general, therefore, the researchers' task of inserting a new model within a segment could benefit from the support of an automatic segmentation tool based on physical and performance parameters. The development of such a tool will be shown in the next chapter.

1.3.3 Data structure, schema_id hierarchy

The complexity of the data to be represented made it necessary to create an adequately articulated structure to be able to catalog in detail all the information requested. In order to describe in details the versions of vehicles with all their characteristics, *Jato* uses a hierarchical structure. Each vehicle characteristic is identified by an id called *schema_id*. Each *schema_id* can take on specific values. The value they assume is specified through the *data_value* attribute. This attribute can indicate, for example, the presence/absence of that feature, specify its size or, in the case of aesthetic features, the colour or the material of which it is composed. In addition to the *data_value*, *schema_id* also have an attribute that identifies their position inside the vehicle. This attribute is named location and can be null in the case of features that do not need to be located or for which localization makes no sense. The *schema_ids* are organized according to a hierarchy. Each *schema_id* has a *parent_schema_id* that it refers to. The *parent_schema_id* is also a *schema_id*, it typically represents the actual item of the vehicle and its *data_value* indicates whether or not it is present in the vehicle. *The schema_id* children, on the other hand, add characteristics or attributes to the feature represented by the *parent_schema_id* and their *data_values* generally specify the details, for example the size, color, material, etc ...

1.3.4 Databases

Jato stores all the information collected within different databases. Each database has the task of conveying a specific information and they all follow the hierarchical structure of *schema_id* introduced previously. The main datasets of *Jato* which have been employed within the thesis are:

- **Carspecs**: it is the database on which much of the work that will be presented in this thesis is based. It represents one of the most important *Jato*'s databases and contains all the information concerning the versions of vehicles available in the present and past market. It follows the hierarchical model of *schema_id* for the representation of the data. Although this scheme allows a high level of detail, it also presents complications due to the way in which the data are collected. The researchers do not set themselves the goal of collecting data for the purposes of analysis but only to describe the data in the most detailed way possible. This means that the data is not always in a ready form for mass processing. For example, a typical problem is the redundancy of some *schema_id* that represent the same feature. The reason is that the researcher at the time of coding of certain features, in order to provide an accurate representation, coding similar features, which differ only in some details, using two different *schema_id.* Having redundant *schema_id* makes the analysis phase more complex because you must take into account the possibility of having features nearly represented with the same identifier. **Carspecs** is made up of a series of regional databases that all follow the same structure. They are made up of several tables, the ones that will be mainly used in this work will be the **version table** and the **equipment table**:
  - **version table**: Each row of the table represents the version of a vehicle. Each row is uniquely identified by the *vehicle_id* attribute which is a value derived from the unique identifier of version, *UID*, concatenated with the *DataDate* attribute which indicates the date of release on the market of the specific version. Including the attributes just mentioned there are 92 total columns to characterize the version. These attributes are all identified through the *schema_id* hierarchy.
  - **equipment table**: each row represents a feature, specify by a *schema_id*, installed on a specific vehicle with the corresponding *data_value* and *location.* The rows of the table are uniquely identified by the *vehicle_id* and *schema_id* attribute pair. This table contains the description of the complete equipment of all the vehicles present in the version table. Through the *vehicle_id* identifier it is possible to trace all the features installed in a specific vehicle and their corresponding values.

- **ModelMix**: it consists of a set of databases that contain the sales volumes of vehicles from 35 different markets. Source volumes data is distilled to version level, and further enhanced with pricing detail. Version informations are researched by local 'model mix volumes' specialists, responsible for collating data from manufacturers and importers. Secondary sources fulfil any further data, to which *Jato* adds prices.

- **TransactionalDB(TA)**: Transaction Analysis is a recent *Jato* data asset that combines *Jato* Vehicle Specifications with information about actual customers' choices on equipment and final transaction price paid for vehicle and options which allows for new class of strategic and tactical insights.

Chapter 2

Vehicle segment classifier

In the previous chapter we saw how the *Jato* segmentation presents some criticalities. It is not always easy for researchers to identify the right segment to assign to new vehicles. This chapter will describe the steps that led to the creation of a classifier that would allow the automatic segmentation of vehicles based on their physical characteristics. This work has a dual purpose:

- The first is to provide help to researchers in carrying out segmentation. This work, as mentioned earlier, is critical, so the use of an automatic classifier, which relies on vehicle characteristics for segment assignment, can help researchers when registering new vehicle models in the databases.

- The second is related to the aim of this thesis. In order to obtain a method able to define the similarity between vehicles, a first step is to identify which are the characteristics of the vehicle that best characterize it. The features identified to obtain a good automatic segmentation, will probably be the same that characterize the model and therefore can also be used to define a concept of similarity.

2.1 Introduction to classifiers

Before talking about the implementation of the algorithm, automatic classification will be introduced and the algorithms used to achieve it will be briefly described.

2.1.1 Classifiers explained

Classification is one of the most important concepts in data science. It is the process of recognizing, understanding and grouping ideas and objects into predefined categories or 'subpopulations'. Using pre-classified **training sets**, machine learning programs use a variety of algorithms to classify future data sets, **test sets**, into categories. One of the most common uses of classification is to filter e-mails into 'spam' or 'non-spam', for example. In short, classification is a form of *"pattern recognition"*: by applying classification algorithms to training data, the same pattern (similar words or feelings, sequences of numbers, etc.) is searched for in future datasets.

The classification can be divided into two categories according to the number of classes that are taken into consideration:

1. **Binary Classification**: A machine learning algorithm that is used to determine which of two classes (categories) a data instance belongs to. The classification algorithm input is a set of samples with labels (training set), where each label could be 0 or 1. Some examples of binary classification are as follows:

   (a) Understanding the feelings of Twitter comments as 'positive' or 'negative'.

   (b) Diagnosis to establish whether a patient has a certain disease.

   (c) Deciding whether or not to mark an email as *"spam"*.

2. **Multiclass Classification**: A machine learning activity that is used to estimate the class (category) of a data instance. The input of the classification algorithm is a set of examples with labels (training set). The number of labels in this case is greater than or equal to 3. Some examples of multiclass classification scenarios are as follows:

   (a) Determine the breed of a dog such as *'Siberian husky'*, *'golden retriever'*, *'poodle'* and so on.

   (b) Classify film reviews as *'positive'*, *'neutral'* or *'negative'*.

The study of classification is very broad, in machine learning there are many classification algorithms that can be used depending on the type of dataset being worked on. It is important to be able to distinguish machine learning algorithms into other two macro-categories:

1. **Supervised learning**: to train the machine learning algorithm, labeled data is used, i.e. data that has already been classified before. The dataset is used as a basis for predicting the class of other unlabeled data. Such algorithms are used for classification and regression problems.

2. **Unsupervised learning**: to train the machine learning algorithm, unlabeled data is used, i.e. data that has never been classified before. The aim is to model the underlying structure or distribution of the data in order to learn more about the data itself. Such algorithms are used for clustering and association problems.

2.1.2 Steps for building the classifier

The process of obtaining a good classification model can be divided into several steps:

- **Data preprocessing**
- **Choice of model**
- **Training phase**
- **Test/evaluation**
- **Tuning of hyperparameters**
- **Prediction**

Data Preprocessing

Data preprocessing in Machine Learning is a crucial step that helps to improve data quality to facilitate the extraction of meaningful information from the data itself. It is the technique of preparing (cleaning and organising) raw information to make it suitable for building and training Machine Learning models. In simple terms, data preprocessing is a technique that transforms raw data into an understandable and readable format.

This process can be subdivided into different phases:

- **Acquiring the dataset**: To create and develop machine learning models, we must first acquire the relevant dataset. This dataset may be composed of data collected from multiple sources that are then combined in an appropriate format to form a dataset. Dataset formats differ depending on the use case.

- **Identifying and handling missing values**: In data preprocessing, it is crucial to identify and handle missing values correctly, otherwise inaccurate and wrong conclusions may be drawn from the data. Basically, there are two ways to handle missing data:
  - **Delete row**: this method is not 100% efficient and is only recommended when the dataset has adequate samples. It is necessary to ensure that no bias addition remains after deletion.
  - **Compute the mean**: this method is useful for features that have numeric data such as age, salary, year, etc. In this case the mean, median or mode of a particular feature that contains a missing value is calculated and the result is substituted for each missing value corresponding to that feature. This method can add variance to the dataset and any data loss can be reversed efficiently. It produces better results than the first method (omission of rows/columns).
- **Coding of categorical data**: categorical data refers to features whose value represents membership of a category. Machine Learning models are mainly based on mathematical equations. Therefore, keeping categorical data in the equation will cause some problems as you should only have numbers in the equations. Then, the goal is to encode the various categories into numerical values. Some of the coding techniques are:
  - **Integer encoding**: each unique label is mapped to an integer number.
  - **One Hot encoding**: each label is mapped to a binary vector. For N categories in a feature, it uses N binary variables.
  - **Dummy encoding**: is a small improvement over one-hot encoding. It uses N-1 features to represent N labels/categories.
- **Dividing the dataset**: Each dataset used for machine learning models must be divided into two separate sets: **training set** and **test set**. The **training set** represents the subset used to train the machine learning model. The **test set**, on the other hand, is the part of the dataset used to test the model. Usually, the dataset is split with a 70:30 or 80:20 ratio. This means that 70% or 80% of the data is taken for training the model, leaving the remaining 30% or 20% for the testing phase. The splitting process varies according to the shape and size of the data set in question.

- **Feature scaling** : is a method of standardizing the independent variables of a dataset within a specific range. In other words, feature scaling limits the range of values in which the variables move so that they can be compared on a common base. Feature scaling is typically done in one of two ways:
  - **Standardization** : is the process of rescaling one or more attributes so that they have a mean value of 0 and a standard deviation of 1. As a result, the characteristics will be rescaled to have the properties of a normal distribution with zero mean and standard deviation of 1. The Z-standard value is computed as follows:

    $$Z = \frac{x_{\mathrm{i}} - u}{\sigma}$$

    where u is the mean of the training samples, σ is the standard deviation of the training samples and x_i is the value to be standardized.
  - **Normalization**: In normalization, the data is scaled to a fixed range, usually from 0 to 1. It is a good technique to use when one does not know the distribution of the data or when one knows that the distribution is not *Gaussian.* The formula is:

    $$Z = \frac{x - min(x)}{max(x) - min(x)}$$

At the completion of these steps the dataset will be ready to be used to train and test a classification model.

Choice of model

There are many models that researchers and data scientists have created over the years. Some are particularly suitable for images, some for sequences (such as text or music), some for numerical data, and some for text-based data. Below we will look at five of the most common supervised algorithms for classification in machine learning:

- **Logistic Regression**: is a classification algorithm used to predict a binary outcome. The independent variables are analyzed to determine the binary outcome with the outcomes falling into one of two categories. The independent variables can be categorical or numerical, but the dependent variable is always categorical.

- **Naive Bayes Classifier**: given the assumption of conditional class independence, i.e. that the effect of an attribute on a given class is independent of the values of the other attributes, it calculates whether or not a data point belongs to a given category.
- **Support Vector Machines**: is a powerful and flexible machine learning algorithm, used for both classification and regression. But generally, it is used in classification problems that can handle several continuous and categorical variables.
- **Decision tree**: is a machine learning algorithm that is perfect for classification problems. It works like a flowchart, separating data points into two similar categories at a time from *"tree trunk"* to *"branches"* to *"leaves"*, where the categories become more finitely similar. This creates categories within categories, allowing organic classification with limited human oversight.
- **Random Forest**: is a machine learning algorithm for classification, regression and other tasks that operates by constructing a multitude of decision trees at the time of training. The result returned by the Random Forest is nothing more than the average numerical result returned by the different trees in the case of a regression problem, or the class returned by the largest number of trees in the case the Random Forest was used to solve a classification problem.

The models that will be explored further on are the decision tree and the random forest because they are the subject of this work.

Training phase

During the data preprocessing two distinct portions of the dataset were obtained, the training set and the test set. In this phase the training set is used to train the algorithm in order to generate the correct output and therefore be able to correctly predict the data class. A machine learning algorithm has certain settings called *"parameters"*, which can be adjusted to change its behaviour. The training procedure uses the examples provided with the training set to find the correct values for these parameters; each machine learning algorithm has its own training procedure. However, there are some parameters that are not adjusted by the training procedure. These parameters are called *hyperparameters* and we will see later how to set them correctly.

Test phase

In this phase the test set is used to validate the goodness of the classifier. It is important that this phase is done using a dataset that has never been seen by the classifier in order to simulate the functioning of the classifier on data from the real world, i.e. data with which the classifier has not been compared during the training phase. The results obtained from the test phase can be evaluated using the following measures:

- **True Positive (TP)**: when it is predicted that an observation belongs to a class and it actually belongs to that class.
- **True Negative (TN)**: when it is predicted that an observation does not belong to a class and in fact it does not.
- **False Positive (FP)**: when it is predicted that an observation belongs to a class and in reality, it does not belong to that class.
- **False Negative (FN)**: when it is predicted that an observation does not belong to a class and actually does belong to that class.
- **Confusion Matrix**: a matrix in which the TP, TN, FP, FN values of the various classes are shown. In the case of multiclass classification, a confusion matrix like this shown in the figure 2.1 will occur .

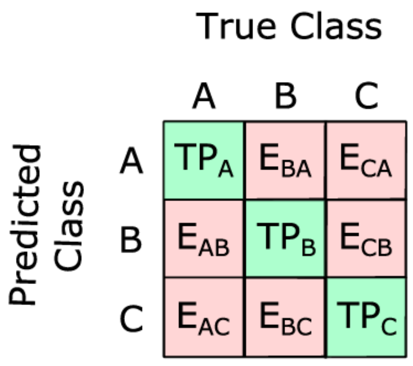


Figure 2.1. Multiclass Confusion Matrix

TP_A is the number of TP in class A and E_{AB} are the samples from class A that have been misclassified as class B. Therefore, the FN in class A (FN_A) is the sum of E_{AB} and E_{AC} ($FN_A = E_{AB} + E_{AC}$) indicating the sum of all samples from class A that have been misclassified as class B or C. Instead, the False Positive of A (FP_A) is calculated as follows, $FP_A = E_{BA} + E_{CA}$.

- **Accuracy**: This is the ratio of the number of correct predictions to the total number of input samples. It only works well if there are an equal number of samples belonging to each class.

$$Accuracy = \frac{True\ Positive + True\ Negative}{All\ Samples}$$

- **Precision**: the ability of a classification model to identify only relevant data points. It is calculated as the correctly predicted positive samples (TP) divided by the total number of predicted positive samples (TP+FP).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- **Recall**: the ability of a classification model to find all relevant cases within a dataset. Recall is calculated as the number of True Positives divided by the total number of True Positives and False Negatives.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- **F1-Score**: F-Measure provides a way to combine precision and recall into a single measure that captures both properties. It is defined as the harmonic mean of the precision and recall of the model.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

In the case of **multiclass classification**, *Precision*, *Recall* and *F1-Score* are more tricky to calculate and are calculated in two ways:

- **Micro average**: all contributions in terms of TP, FP, TN, FN of the available classes are considered and added together in order to calculate the validation measures. This type of measurement is preferable in the case that there is an imbalance in the distribution of the classes.
- **Macro average**: the measures are calculated independently for each class and then the average is made between them. It is used when you want to associate the same importance to all classes.

Tuning of hyperparameters

This phase aims to find the optimal set of *hyperparameters*. *Hyperparameters* are parameters that define the architecture of the model. As mentioned before these parameters cannot be trained during the training phase but must be obtained experimentally. There are several techniques for tuning the parameters, the most important of which are:

- **Grid search**: this is probably the simplest *hyperparameter* optimization method. As many lists of values are provided as there are *hyperparameters* to be optimized, and with this technique, we simply build a model for each possible combination of all the *hyperparameter* values provided, evaluating each model and selecting the architecture that produces the best results.

- **Random search**: differs from grid search in that we no longer provide a discrete set of values to explore for each *hyperparameter*; rather, we provide a statistical distribution for each *hyperparameter* from which values can be randomly sampled to try to identify the architecture that produces the best results.

Cross Validation

I also introduce the concept of cross validation which will be useful later. Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called *k-fold cross-validation.*[15]

Cross validation consists in:

1. Splitting the data into training and test set

2. Further splitting the training set into a larger number of non-overlapping subsets called ‘folds’

3. Then choose one of the folds as validation fold

4. The rest of the folds will be merged and used for training

5. The process can be repeated for k times (where k is the number of folds)

6. Once the best choice of hyperparameters across all folds has been identified, the model can be trained on the whole training set and tested on the data

This procedure can be very time-consuming, but it works well when the dataset is not too big and provides a good estimate of the model's hyperparameters.

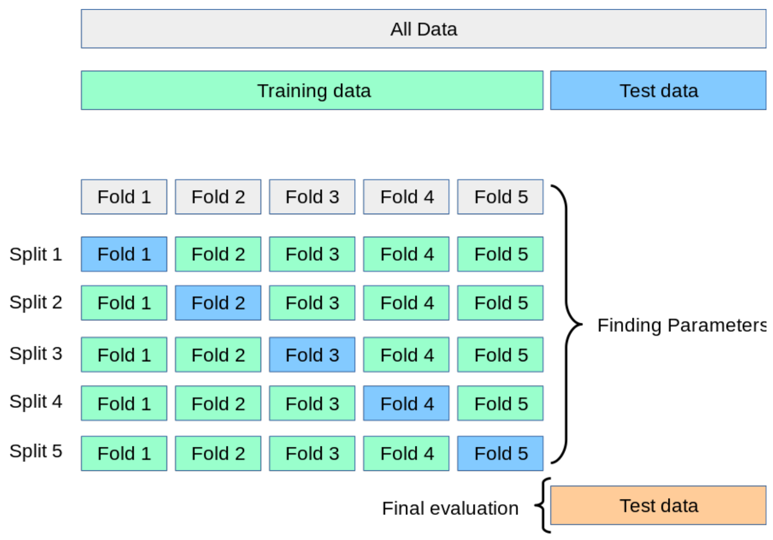


Figure 2.2. Cross validation procedure

Prediction

At this stage, the model is ready to be used for what it was created for. The classification algorithm is ready to make predictions on new data, then to be used to classify data from the real world.

2.1.3 Introduction to Decision tree

In the previous section of this paragraph, we have briefly seen some of the machine learning models that can be used for classification. Among those seen, the *decision tree* and *random forest* were used to solve the problem proposed in this chapter. For this reason, in this and the next section of this paragraph I will go to describe these two algorithms in more detail.

The **Decision Tree** (**DT**) is a non-parametric supervised machine learning algorithm used for classification and regression. Decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules are and the more fitted the model is.

The decision tree builds classification or regression models in the form of tree structures, figure 2.3. It divides a data set into smaller and smaller subsets while at the same time an associated decision tree is developed incrementally. The result is a tree with decision nodes and leaf nodes. A **decision node** has two or more branches. The **leaf node** represents a classification or decision and has no output branches. The highest decision node in a tree that corresponds to the best predictor is called the **root node**. Decision trees can handle both categorical and numerical data.

Important terms when talking about Decision trees are:

- **Root Node**: represents the entire dataset or a sample of the dataset.
- **Splitting**: is the process of dividing a node into sub-nodes.
- **Decision Node**: is a sub-node that splits into other sub-nodes.
- **Leaf / Terminal Node**: are nodes that do not divide further.
- **Pruning**: is the process of eliminating the sub-nodes of a decision node and is the opposite of splitting.
- **Branch / Sub-Tree**: is the sub-part of the whole tree.
- **Parent and Child Node**: a node, which is divided into sub nodes is called a parent node of sub nodes where the sub nodes are the child nodes of the parent node.

How the decision tree works

The decision on how to make the node divisions has a great influence on the accuracy of the tree. The decision criteria differ between a decision tree for classification and one for regression. The creation of subnodes increases the homogeneity of the resulting subnodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree divides the nodes over all available variables and then selects the division that results in more homogeneous subnodes.

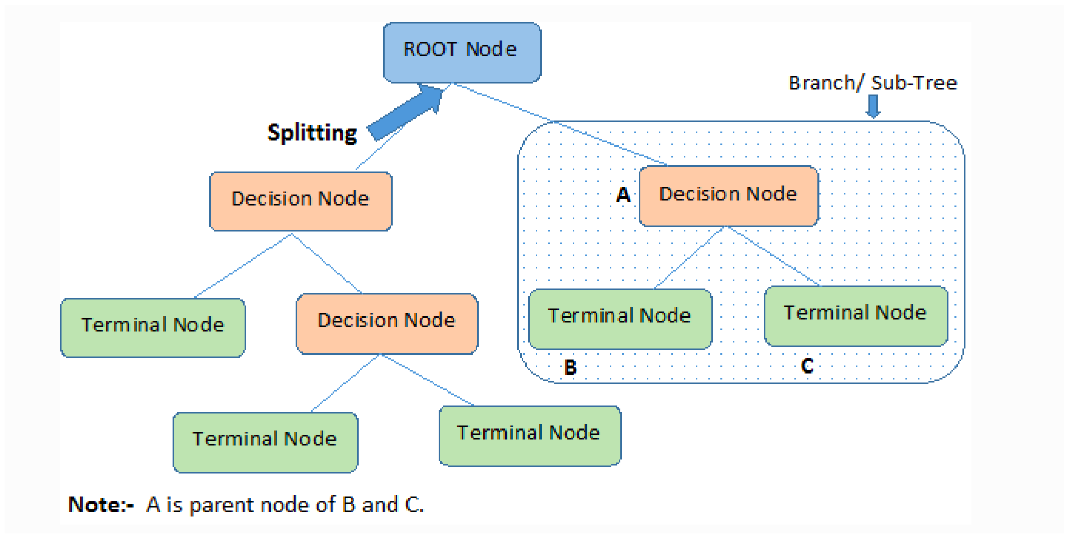


Figure 2.3. Decision tree structure

Attribute selection measures

If the dataset consists of N attributes, deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. Randomly selecting any node as root cannot solve the problem. If we follow a random approach, it may give us negative results with poor accuracy. To solve this attribute selection problem, the researchers worked and devised some solutions by suggesting to use some criteria such as: *Entropy*, *information gain*, *Gini index*, *gain ratio*, *variance reduction* and *chi-squared*. These values are calculated for each attribute and then sorted according to these values. The attributes are inserted into the tree following the order, the attribute with the highest value is placed at the root. For categorical attributes, *information gain* will be used as a criterion. For continuous attributes, the *Gini index* will be used. After building the decision tree, classifying a test record is simple. Starting from the root node, the test condition is applied to the record and it will follow the right branch based on the node's test condition. This leads to another internal node for which a new test condition is applied or to a leaf node. Once at the leaf node, the class label associated with the leaf is then assigned to the record. The *Gini index* will be used in our classifier so we will see how to compute it.

Gini Index

The *Gini index* can be seen as a cost function used to evaluate the splits in the data set. It is calculated by subtracting the sum of the squared probabilities of each class from one.

$$Gini = 1 - \sum_{i=1}^{C} (p_{\mathrm{i}})^2$$

It favors larger and easier to implement partitions, while information gain favors smaller partitions with distinct values. The *Gini index* only generates binary divisions. The higher the value of the *Gini index*, the more homogeneous the divisions will be.

Limiting the overfitting of the decision tree

The common problem with decision trees, especially with a table full of columns, is that they fit a lot of data. Sometimes it looks like the tree has stored the training dataset. If there are no limits set on a decision tree, it is likely to get 100% accuracy on the training set because in the worst case it will end up creating a leaf node for each observation. Therefore, this affects the prediction accuracy of samples that are not part of the training set. Two techniques can be used to limit this problem:

- Pruning of the decision tree
- Random Forest.

Pruning of the Decision Trees

The pruning process results in trees that have grown until the stopping criteria are reached. However, the fully grown tree is likely to overtrain on the training data, leading to poor accuracy on the yet unseen data. In pruning, the branches of the tree are cut off, i.e. the decision nodes are removed starting from the leaf node so that the tree does not grow too large and generate overfitting on the training data.

2.1.4 Introduction to Random Forest

Random Forest is an example of an ensemble machine learning algorithm, in which several machine learning algorithms are combined to achieve better predictive performance. Random Forest therefore consists of a large number of independent decision trees operating as an ensemble. Each individual tree in the random forest outputs a class prediction and the class with the most votes becomes the model prediction. Low correlation between models is the key. Uncorrelated models can produce ensemble predictions that are more accurate than any individual prediction. The reason for this effect is that trees protect each other from their individual errors (as long as they do not go wrong together in the same direction). While some trees may be wrong, many others will make the right prediction, so as a group the trees are able to move in the correct direction. There are two ways of obtaining independent decision tree models:

- **Bagging**: decision trees are very sensitive to the data they are trained on, small changes to the training set can generate significantly different trees. The Random Forest exploits this advantage by allowing each individual tree to sample randomly from the dataset, generating different trees. This process is known as *bagging*.
- **Feature Randomness**: in a normal decision tree, when it is time to split a node, we consider every possible feature and choose the one that produces the best splitting. In contrast, each tree in a random forest can only choose from a random subset of features. This imposes even more variation between trees in the model and ultimately results in less correlation between trees and more diversification.

2.2 Data collection and cleaning

The first and one of the most important phases in the building of a machine learning algorithm is **data preprocessing**. This phase, like all the subsequent ones, was carried out using the Python programming language with its supporting library. I will describe the various steps carried out during the preprocessing phase based on the steps described in the previous paragraph:

- The first step was to identify the database from which to extract the vehicle data. It was decided to use vehicles from the Italian market to create the model and then apply the algorithm to the Russian, German and British markets. The database from which the data was obtained is *Carspecs* for the Italian market. The information necessary for the realisation of the classifier are the vehicles including their physical, performance and content characteristics and, since we want to solve a supervised learning problem, the label relative to the *Jato* segment manually assigned by the researchers when the vehicle is entered in the database. The tables of the *Carspecs* database from which to extract this data are *version* and *equipment*. The *version* table contains all the versions of vehicles on the Italian market with the relative characteristics. In the *equipment* table, as described in chapter 1, each line represents an option installed in a specific vehicle. The features extracted are, for the version table:

– ***vehicle_id***: Vehicle identifier.
– ***JATORegSegment***: European regional segment assigned to the vehicle.
– ***Make***: vehicle manufacturer name.
– ***Model***: vehicle model name.
– ***RetailPrice***: selling price of the vehicle in euro.
– ***bodytype***: encodes the shape and/or function of the vehicle bodywork.

From the equipment table, on the other hand, the information extracted are:

– ***carlength***: vehicle length in *mm*.
– ***carwidth***: vehicle width in *mm*.
– ***carheight***: vehicle height in *mm*.
– ***wheelbase***: the distance between the central point of the front and rear wheels of the vehicle in *mm*.
– ***GVW***: encodes the absolute maximum the vehicle can weigh, including load, passengers, fuel and so on in *kg*.
– ***engineLiters***: encodes engine displacement in *litres*.
– ***groundClearance***: encodes the maximum ground clearance of a vehicle with its empty weight in *mm*

In addition to the listed features, information on the **premiumness** of vehicles has also been added based on the manufacturer to which they belong. Based on characteristics that indicate the market segment in which the car manufacturers are placed, numerical values are assigned ranging from 0 to 3 where 0 indicates the highest *premiumness level* and 3 indicates the lowest. At this point I have obtained a dataset containing

8473 rows and 14 columns. Each row represents a version of a vehicle while the columns represent the features just described. The level of granularity we want to obtain is that of the model so we will have to group the rows representing the vehicles into models. Before doing this, I check the presence of null values within the dataset.

- To check for **null values**, a count of the rows with null values was performed for each of the columns of the dataset. The only feature with null values was *groundClearance* with 2647 out of 8473 null values. As seen previously there are two ways to handle null values and since we do not want to discard the rows to avoid excluding some significant vehicle model, we have opted to replace the null values with the average values of the feature. In particular, the average *groundClearance* value was calculated for each segment and for each vehicle with a null value the average value of the corresponding segment was replaced. In this way, the dataset is free of null values and we can continue with the preparation of the data.

- Another step preceding to grouping the versions into models is to encode the categorical features appropriately. The categorical features in this dataset are: *BodyType* and *premiumness*.

  – **BodyType**: there are 11 different BodyType values; *convertible*(**CA**), *coupe*(**CO**),*hetchback*(**HA**),*micro car*(**MC**), *mini MPV*(**MM**), *Mini-van*(**FW**), *pickup*(**PU**), *sedan*(**SA**), *sport utility vehicle*(**OD**), *targa*(**TA**), *wagon*(**ES**). Given the high number of categories present in this feature I decided to collect the body type grouping the most similar body types in terms of shape and size in 5 macro categories:

    ∗ **0**: **PU**,

    ∗ **1**: **CA**,**CO**,**TA**

    ∗ **2**: **HA**,**MC**,**SA**,**ES**

    ∗ **3**: **FW**,**MM**

    ∗ **4**: **OD**

  – **premiumness**: I apply the dummy econding.

At this point the data are ready to be grouped in order to obtain model level granularity. As a result, I obtain a dataset with 293 rows, representing 293 different models, and 14 columns containing the features obtained by calculating the mode of the values corresponding to each version of the corresponding model. The vehicles represented in the dataset are divided between the segments as follows:

- **EU A - utility/city cars**: 10
- **EU A SUV**: 2
- **EU B – small**: 20
- **EU B SUV**: 28
- **EU C1 - lower medium -**: 15
- **EU C1 SUV**: 22
- **EU C2 - lower medium +**: 16
- **EU C2 SUV**: 13
- **EU D1 - upper medium -**: 10
- **EU D1 SUV**: 15
- **EU D2 - upper medium +**: 12
- **EU D2 SUV**: 15
- **EU E1 - large and executive**: 13
- **EU E1 SUV**: 13
- **EU E2 – luxury**: 12
- **EU E2 SUV**: 8
- **EU Large MPV**: 8
- **EU Medium MPV**: 7
- **EU Mini MPV**: 18

As can be seen, the distribution between the various categories is not homogeneous and this must be taken into account in the following steps.

- The data obtained were then splitted into training set and test set using an 80:20 ratio, obtaining two datasets of 234 and 59 vehicles, respectively. In the split operation, the distribution of the data in the various categories was also considered, maintaining the same distribution of classes in both the training set and the test set.
- In the case of random forest and decision tree, the feature scaling phase is not necessary because the node conditions are evaluated on the individual features so it is not necessary to have all the variables on the same scale to compare them.

2.3 Decision Tree Classifier

One of the goals of this work phase is to obtain information regarding the importance of the features that best characterize the vehicles, in order to be able to define a measure of similarity between them later on, for this reason I chose the **Decision Tree** as the first classifier to be tested. This decision is mainly based on the fact that this classification algorithm is known to be easily understandable and interpretable.

2.3.1 Decision tree implementation

In order to correctly implement the decision tree, some parameters that specify its behavior must be set. In particular, for this implementation of the decision tree classifier I have set the following parameters:

- **Criterion**: this parameter rappresent the function used to measure the quality of the splits. As previously mentioned, the split criterion can be chosen from the following options: *Entropy*, *information gain*, *Gini index*, *gain ratio*, *variance reduction* and *chi-squared.* In this case I have chosen to use the *Gini index* since in my dataset I have for the most part continuous variables.

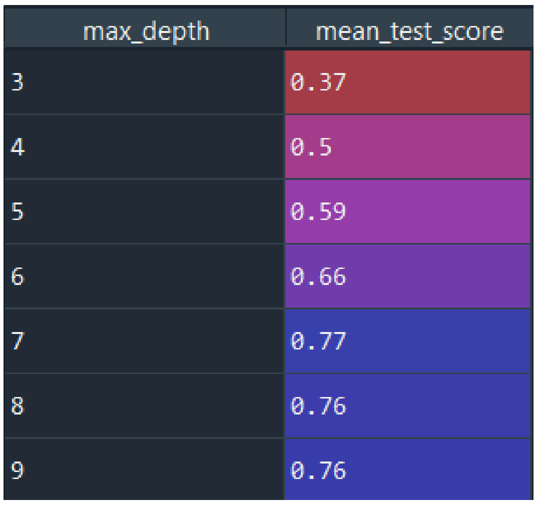
- **Split strategy**: this parameter indicates the strategy to choose the split at each node. There are two possible strategy: *best* to choose the best split and *random* to choose the split randomly. The chosen strategy was the *best* strategy.

- **Max depth**: this parameter indicates the maximum depth of the tree. It needs to be chosen following a parameter tuning phase in order to identify the depth that performs best in terms of accuracy. For this parameter tuning I have adopted the technique of *Grid Search* with *cross validation.*

2.3.2 Decision Tree training and hyperparameter tuning using cross-validation

To train the model and perform the *max depth* parameter tuning phase I used a *5-fold cross validation* with *Grid Search.* For the tuning of the *max depth* I carried out the *grid search* in the range of values from 3 to 9. For each of these values a *5-fold cross validation* was performed using the training set obtained previously as a dataset, in order to identify the value of *max depth* which results in the best average accuracy.

I got as a result a number of decision tree models equal to the number of parameters I passed to the function. Each of these models has an average accuracy value as shown in the figure 2.4.



max_depth	mean_test_score
3	0.37
4	0.5
5	0.59
6	0.66
7	0.77
8	0.76
9	0.76

Figure 2.4. Decision tree models accuracy after cross validation

The value of *max depth* that allows to obtain the best train accuracy value is 7 with a mean accuracy score of 77%.

2.3.3 Decision Tree evaluation

The obtained model must therefore be evaluated on a set of data never seen before. This is the **test phase**. In order to obtain the necessary validation measures, I must first run the classification algorithm on the test set, obtaining the model predictions for the vehicles present in the test set. Once the predictions have been obtained, it is possible to calculate the various validation measures starting from the accuracy on the test set:

Test set accuracy: 0.75

We can see how the accuracy obtained on the test set is in line with that obtained during the training phase. The model seems to work as expected on new samples but still does not get a good result in terms of accuracy. Repeating the prediction process on the training set we see how much the model fitted the dataset. From the obtained results it is evident as the model suffers of overfitting.

Training set accuracy: 0.99

Calculating also the measures of *precision*, *recall* and *F1* in micro and macro average for the test set, shown in table 2.1, we see how, unfortunately, the model does not work very well as classifier.

Precision_{μ}	Precision_{M}	Recall_{μ}	Recall_{M}	F1-score_{μ}	F1-score_{M}
73%	70%	73%	69%	73%	68%

Table 2.1. Decision Tree evaluation metrics

However, in the next section we put the focus on the kind of information that the model gives us by analysing the decision tree structure.

2.3.4 Decision Tree structure analysis

The analyses now move on to interpreting the structure of the tree generated by the model just found. These analyzes aim to identify the features that are important for the model to assign the class label. Identifying these features would give us an estimate of the features that best characterize the vehicle.

Given the size of the tree, analysis will be performed on parts of the diagram that result more interesting.

To better understand the images below I report a legend with the meaning of the nodes' fields:

1. Test condition on a vehicle feature
2. Gini index value
3. Number of samples in that specific node
4. Distribution of samples present in the node between the various categories. The order of the classes is the following: **EU A - utility/city cars**, **EU A SUV**, **EU B - small**, **EU B SUV**, **EU C1 - lower medium -**, **EU C1 SUV**, **EU C2 - lower medium +**, **EU C2 SUV**, **EU D1 - upper medium -**, **EU D1 SUV**, **EU D2 - upper medium +**, **EU D2 SUV**, **EU E1 - large and executive**, **EU E1 SUV**, **EU E2 - luxury**, **EU E2 SUV**, **EU Large MPV**, **EU Medium MPV**, **EU Mini MPV**, **EU Sports**

5. Prevalent class in the node's samples

As mentioned above the root node is the best predictor and in this case, it indicates the feature that best characterizes the vehicle. The root node bases its subdivision on the body type feature, in particular, on whether the vehicle has the body type that belongs to the macro-category 1 or not.

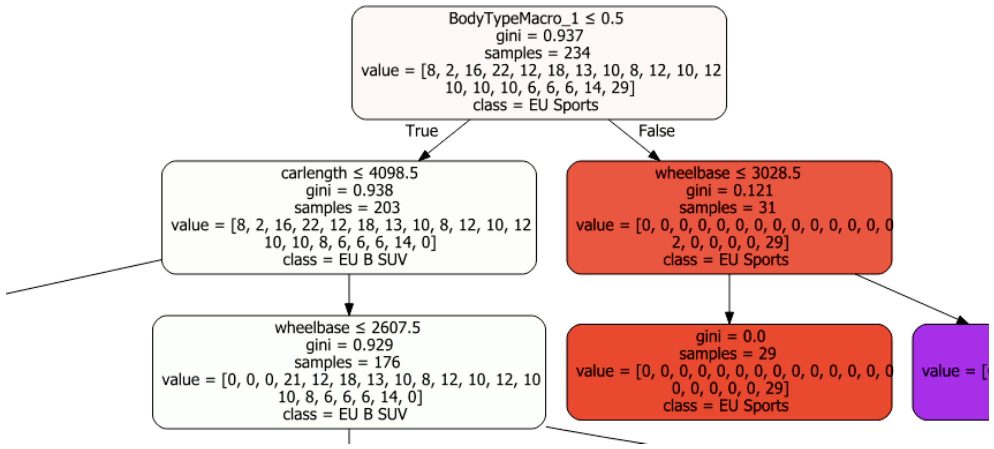


Figure 2.5. Root node with its branches of the best decision tree model

We see, in the figure 2.5, that in the node following the *True* branch, there are 203 samples so most of the training samples. Instead in the node to the right there are only 31 samples of which almost all belong to the *EU Sports segment*. We can say that this branch clearly separates sports vehicles from non-sports vehicles.

The second most relevant node is the following.

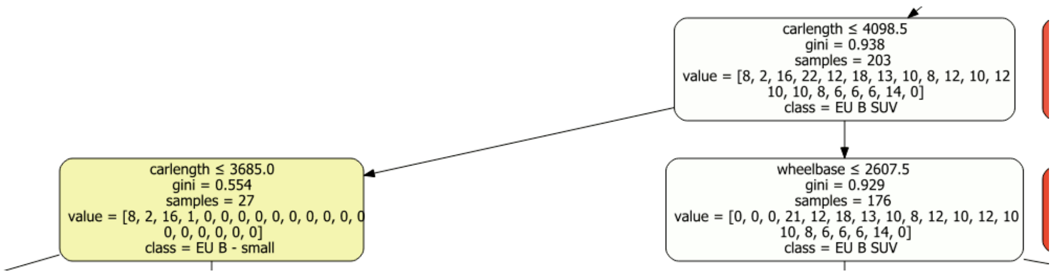


Figure 2.6. First decision tree structure detail

The node in the up right, in the figure 2.6, is the node that follow the root node after the *True* branch. This node apply a test on the *carlength* attribute and, as we can see from the results, it divides the vehicles of small dimensions, like the vehicles in the segments *EU A - utility/city cars*, *EU A SUV* and *EU B - small*, form the other vehicles. This suggests us that also the car length is a relevant feature in the characterization of the vehicle.

Exploring the tree deeper we find other interesting nodes. The top left node, in the figure 2.7, tests the *RetailPrice* attribute. The resulting separation shows that price is fundamental in the distinction between high-end and low-end segment vehicles. Analysing more in detail we see how in this branch there are mainly SUV vehicles and that the price makes a separation between the SUV up to segment C1 and the SUV from segment C2.

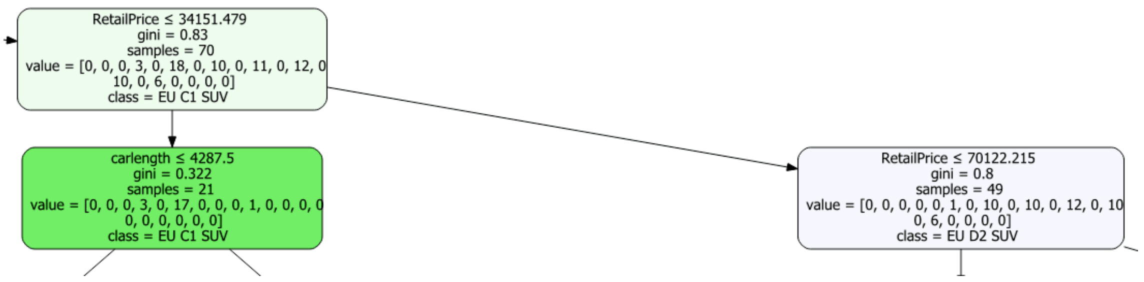


Figure 2.7. Second decision tree structure detail

In the schema, among the features used to develop the classification, the *premiumness* feature, linked to the *premiumness* of the vehicle, does not appear. Probably this is due to the fact that the price of the vehicle represented by *RetailPrice* is already sufficient to distinguish between vehicles of different market ranges.

Although the first model allowed us to extrapolate interesting information, we want to obtain a more accurate classification and more details about importance of the features involved.

2.4 Random Forest Classifier

This time we use the random forest model to classify vehicles. This machine learning algorithm, as mentioned above, manages to achieve better results by basing its decision on several low correlated decision trees.

2.4.1 Implementation

As for the decision tree algorithm, also for the implementation of the random forest algorithm it is necessary to specify parameters that characterize the algorithm's behavior. For the implementation of the random forest, the same parameters adopted in the case of the decision tree were chosen:

- **Criterion**: I used the *Gini Index* since in my dataset there are mainly continuous variables.
- **Splitter**: Also in this case the chosen strategy is the *best* strategy.
- **Max depth**: for the random forest I used the value 7 that in the single decision tree got the best result in terms of accuracy

This type of algorithm is very sensitive to the datasets used for training and test phases. Given the limited amount of data and the high number of classes available, different types of splits generate training and test sets with different class distribution, greatly influencing the quality of the classifier. To manage this problem and get robust quality estimates on the classifier, I managed of doing different splits on the data and create a random forest model for each split. This approach allows me to obtain classifier evaluation measures that are as independent as possible from the process of data splitting.

2.4.2 Training phase

This procedure is based on a cycle of 70 iterations in which for each iteration I generate a new dataset split, getting different training and test sets each time. For each of these splits I train and test a different random forest model using the parameters described above. For each model generated, I have computed and stored five evaluation metrics:

- **Test set accuracy**: the test accuracy value obtained comparing the prediction labels on the test set with the correct labels of the test set itself.
- **Training set accuracy**: the training accuracy value obtained comparing the prediction labels on the training set with the correct labels of the training set itself.

- **Features importance scores**: one of the main reasons why I chose to use the random forest as a classification model is the possibility of obtaining an estimate of the importance of the features in the classification. I store a numeric value for each feature. This value indicates how relevant a feature was in assigning the label.
- **Predicted class probabilities**: for each sample I stored a vector having as many values as classes and each value represents the probability that the sample belongs to a specific class. The predicted class probabilities of an input sample are computed as the mean predicted class probabilities of the trees in the forest. The information related to the prediction confidence are useful to understand how much the classifier is secure about the predictions made.
- **Misclassified vehicles**: I manage a data structure that considers all the vehicles that have been misclassified. This data structure allows me to have more information on the types of errors that the classifier makes at each iteration.
- **Number of predictions**: during the various iterations, the vehicles can be inserted or not within the test set. Each time a specific vehicle appears in the test set, a counter is incremented. Each vehicle has a counter that keeps track of the number of times it is classified.

Having collected all this information at a single iteration level, the next step is to aggregate them to generate measures that characterize the model regardless of the type of data split adopted. The measures are aggregated in this way:

- **Average test set accuracy**: the average value is calculated from the test accuracy list.
- **Average training set accuracy**: the average value is calculated from the train accuracy list.
- **Average feature importance**: The average value of importance is calculated for each feature.
- **Average predicted class probability**: for each vehicle, the average value is calculated by adding the predicted class probability vectors obtained for each forecast, divided by the number of predictions made on the vehicle.

In order to manage all the measures relating to vehicles and generate other derivatives measures, I created a scructure, figure 2.8, that has as many rows as there are vehicles, 293, and the following eight columns:

- **JATORegSegment**: segment assigned by Jato. Represents the correct class label label.
- **Make**: car manufacturer of the vehicle.
- **Model**: vehicle model name.
- **#test**: number of times the vehicle has been inside the test set during the iterations.
- **#missclassified**: number of times the vehicle has been incorrectly classified.
- **Probabilityavg**: Average predicted class probability associated with the most frequently predicted class for that vehicle.
- **Miss_ratio**: ratio between #test/#missclassified.
- **Frqprediction**: most frequently predicted class label for that vehicle.

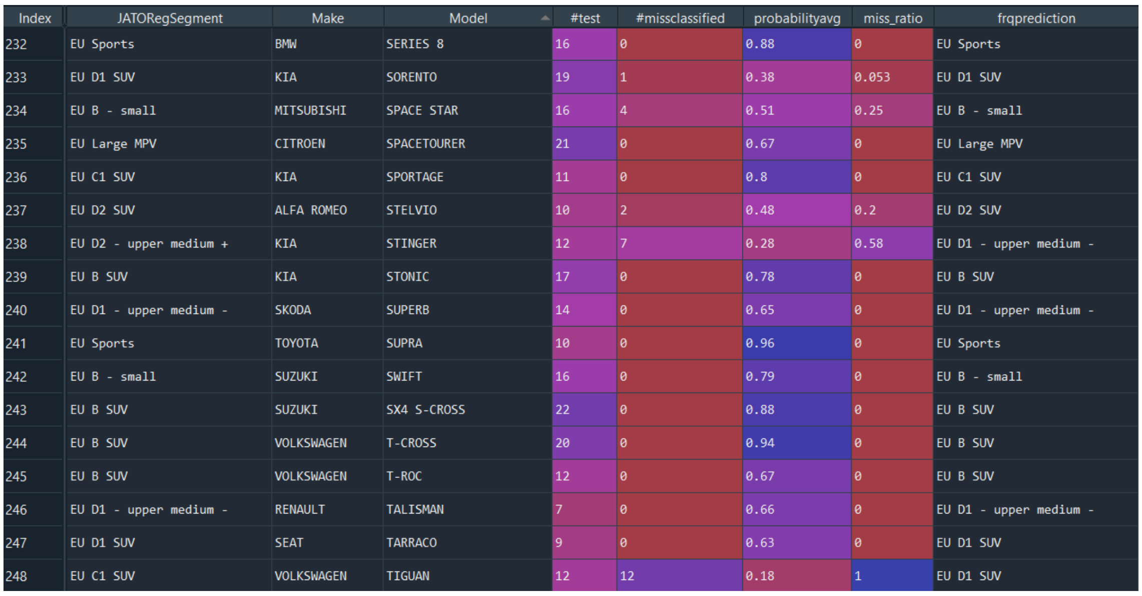
2.4.3 Classifier evaluation

The analyses will start with the accuracy measurements. We have seen how the decision tree reached an accuracy value of the test set equal to 77%, while on the test set we observed an accuracy of 99% symptom of a marked overfitting. The trained random forests reach an average accuracy value on the test set of 84%

Average test set accuracy: 0.84

It is therefore evident how actually the use of an ensemble model has brought advantages in terms of accuracy on the test set. This allows us to have a good classifier that can be used as a decision support for assigning the segment. Looking instead at the accuracy on the training set, we have a result very similar to that of the decision tree:

Average training set accuracy: 0.98



Index	JATORegSegment	Make	Model	#test	#missclassified	probabilityavg	miss_ratio	frqprediction
232	EU Sports	BMW	SERIES 8	16	0	0.88	0	EU Sports
233	EU D1 SUV	KIA	SORENTO	19	1	0.38	0.053	EU D1 SUV
234	EU B - small	MITSUBISHI	SPACE STAR	16	4	0.51	0.25	EU B - small
235	EU Large MPV	CITROEN	SPACETOURER	21	0	0.67	0	EU Large MPV
236	EU C1 SUV	KIA	SPORTAGE	11	0	0.8	0	EU C1 SUV
237	EU D2 SUV	ALFA ROMEO	STELVIO	10	2	0.48	0.2	EU D2 SUV
238	EU D2 - upper medium +	KIA	STINGER	12	7	0.28	0.58	EU D1 - upper medium -
239	EU B SUV	KIA	STONIC	17	0	0.78	0	EU B SUV
240	EU D1 - upper medium -	SKODA	SUPERB	14	0	0.65	0	EU D1 - upper medium -
241	EU Sports	TOYOTA	SUPRA	10	0	0.96	0	EU Sports
242	EU B - small	SUZUKI	SWIFT	16	0	0.79	0	EU B - small
243	EU B SUV	SUZUKI	SX4 S-CROSS	22	0	0.88	0	EU B SUV
244	EU B SUV	VOLKSWAGEN	T-CROSS	20	0	0.94	0	EU B SUV
245	EU B SUV	VOLKSWAGEN	T-ROC	12	0	0.67	0	EU B SUV
246	EU D1 - upper medium -	RENAULT	TALISMAN	7	0	0.66	0	EU D1 - upper medium -
247	EU D1 SUV	SEAT	TARRACO	9	0	0.63	0	EU D1 SUV
248	EU C1 SUV	VOLKSWAGEN	TIGUAN	12	12	0.18	1	EU D1 SUV

Figure 2.8. Validation table detail

Also in this case the classifier suffers overfitting. This is a known issue for Random Forest classifiers.

To analyze the performance of the classifier in more detail, it was useful to extrapolate the *Accuracy*, *Precision*, *Recall* and *F1-score* values relating to the individual segments and the resulting micro and macro average values.

The validation measures for each class are reported in the table 2.2. We can see that the class that is correctly predicted every time is that relating to the *EU Sports* segment which identifies sports cars while the one on which the classifier struggles the most is that of vehicles belonging to the *EU Medium MPV* segment. Two classes in which the classifier struggles are *EU C1-lower medium -* and *EU C2-lower medium +*, probably this behavior is due to the fact that the vehicles belonging to these two classes are very similar between them therefore the classifier tends to mistake them and make errors in the predictions.

In addition to the accuracy values shown above, the micro and macro averages of the measures listed in the table 2.2 have been calculated

From the table 2.3 we see how in terms of micro and macro average we have good *precision* and *recall* values. In particular, given that our dataset has a non-homogeneous distribution of classes, the *micro average* metrics are taken into greater consideration because they allow us to weigh the classes

Segment	Precision	Recall	F1-score	Accuracy
EU D2 - upper medium +	83%	95%	87%	83%
EU C2 SUV	90%	80%	83%	90%
EU Sports	100%	100%	100%	100%
EU Mini MPV	95%	100%	98%	95%
EU E2 - luxury	95%	94%	93%	95%
EU A - utility/city cars	90%	91%	88%	90%
EU C2 - lower medium +	64%	79%	69%	64%
EU B - small	90%	91%	89%	89%
EU B SUV	95%	81%	87%	95%
EU C1 - lower medium -	66%	54%	56%	66%
EU C1 SUV	85%	86%	83%	85%
EU D1 - upper medium -	92%	77%	81%	92%
EU E2 SUV	73%	64%	66%	73%
EU D2 SUV	73%	72%	70%	73%
EU Medium MPV	33%	41%	36%	33%
EU D1 SUV	86%	74%	77%	86%
EU Large MPV	80%	76%	76%	80%
EU E1 SUV	70%	82%	73%	70%
EU Large MPV	97%	90%	92%	97%

Table 2.2. Random Forest segments evaluation metrics

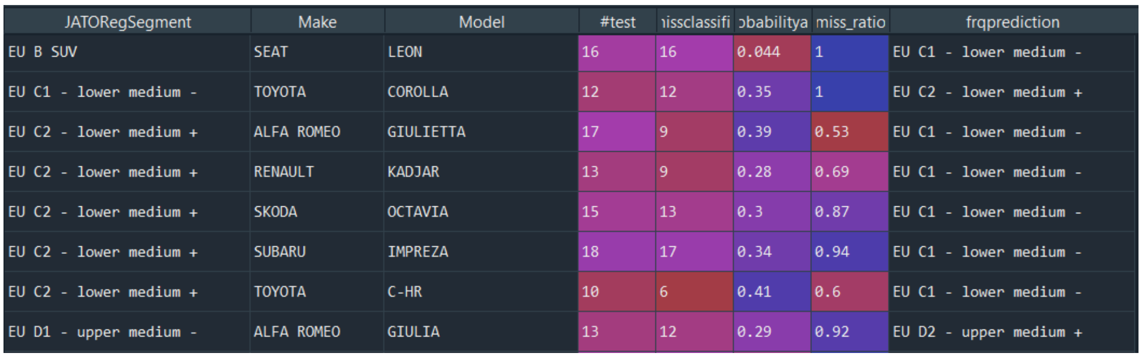
Precision$_\mu$	Precision$_M$	Recall$_\mu$	Recall$_M$	F1-score$_\mu$	F1-score$_M$
84%	82%	86%	80%	84%	80%

Table 2.3. Random Forest evaluation metrics

adequately to the number of elements present in the dataset.

A more in-depth analysis of the quality of the classifier can be made using the measures contained in the structure previously shown. Here, as mentioned before, we have detailed information on how the classifier behaves with each model present in the dataset. Among all the models we look for those that are wrongly classified most of the time. To search for these models, I filter the dataset by keeping only those models that have the attribute *miss_ratio* greater than 0.5.

The resulting dataset has 48 rows. This shows us that 16% of vehicles are badly rated most of the time.



JATORegSegment	Make	Model	#test	iissclassifi	obabilitya	miss_ratio	frqprediction
EU B SUV	SEAT	LEON	16	16	0.044	1	EU C1 - lower medium -
EU C1 - lower medium -	TOYOTA	COROLLA	12	12	0.35	1	EU C2 - lower medium +
EU C2 - lower medium +	ALFA ROMEO	GIULIETTA	17	9	0.39	0.53	EU C1 - lower medium -
EU C2 - lower medium +	RENAULT	KADJAR	13	9	0.28	0.69	EU C1 - lower medium -
EU C2 - lower medium +	SKODA	OCTAVIA	15	13	0.3	0.87	EU C1 - lower medium -
EU C2 - lower medium +	SUBARU	IMPREZA	18	17	0.34	0.94	EU C1 - lower medium -
EU C2 - lower medium +	TOYOTA	C-HR	10	6	0.41	0.6	EU C1 - lower medium -
EU D1 - upper medium -	ALFA ROMEO	GIULIA	13	12	0.29	0.92	EU D2 - upper medium +

Figure 2.9. Frequently misclassified vehicle dataset extract

From this extract of the dataset, figure 2.9, it is possible to extrapolate some interesting information. It can be seen how a typical classifier error is to associate the label of an adjacent segment. This type of error may not be serious if the classifier is used as a decision support for Jato researchers. Another interesting thing is the presence of vehicles that are always misclassified. Among these vehicles there is the *Seat Leon* which in the Jato segmentation is classified as a *B SUV* while for the classifier it should belong to the *C1 segment - lower medium -*. As anticipated in the previous chapter, the Jato segmentation has some problems, the models are not always found in the segment that best suits them. For example, in some cases the vehicles are placed in specific segments under direct request of the manufacturers even if the most correct segment would be another. From this we can understand how a classifier that has the task of carrying out automatic segmentation cannot be perfect.

For this reason, it is interesting to be able to identify a subset of models for which the classification is almost certain, on which a standalone classifier could be applied. I create two subsets by filtering the vehicles based on the confidence value that the classifier has in classifying them. The confidence thresholds used were 50% and 60%, this values were chosen following a heuristic approach using the Italian regional dataset as a training set.

In order to obtain the first subset I filtered the validation dataset by selecting only the vehicles that have the *probabilityavg* feature greater than 0.5. The resulting dataset,that we'll call **subset1**, has 180 rows. Of these 180 vehicles, only 5 are missclassified at least once. This result shows us how by choosing a set of vehicles that are classified with a confidence of at least 50% we obtain 61% of the dataset and on these vehicles the classifier has an accuracy of 97%.

To carry out a test with a second subset, called **subset2**, I used 60% as the confidence threshold. **Subset2** has 129 vehicles, of these vehicles only 1 is missclassified at least once. With a confidence of at least 60% we obtain 44% of the dataset and the classifier can predict these vehicles with an accuracy greater than 99%.

2.4.4 Features importance analysis

We confirmed that also in our case the random forest obtains better results than the decision tree in the classification problems, obtaining a classifier that can be of great help to Jato researchers when entering new vehicles into the database. The last important measure to analyse is relating to the importance of features in the classification. The mean values of importance of the features obtained during the various iterations of the random forest are shown in table 2.4:

These results show us that the most important feature for segment prediction is *carlength* followed by *wheelbase* and *RetailPrice.*

It is important to note how the features generated by the *BodyType* after dummyfication, also have a particular relevance. If we sum the scores for all 4 *BodyTypeMacro* attributes (*BodyTypeMacro_1*+ *BodyTypeMacro_2*+ *BodyTypeMacro_3*+ *BodyTypeMacro_4*), we would have that the *BodyTypeMacro* feature has an importance value equal to 0.23, revealing itself to be the most relevant feature.

The features that are not very relevant are *engineLiters*, with an importance value of 0.03, and the *premiumness* feature, which, as already seen in the decision tree, is of little importance in the prediction of segment.

Feature	Importance
carlength	16%
wheelbase	12%
RetailPrice	11%
carheight	10%
GVW	9%
groundclearance	7%
BodyTypeMacro_1	7%
carwidth	6%
BodyTypeMacro_2	6%
BodyTypeMacro_4	6%
BodyTypeMacro_3	4%
engineLiters	3%
premiumness_1.0	2%
premiumness_2.0	1%
premiumness_0.0	0%
premiumness_3.0	0%

Table 2.4. Features importance values

2.4.5 Random Forest implementation in other markets

Jato being an international company manages the data of different automotive markets. Each market has its own peculiarities and the vehicles within it can change in type and characteristics. The classifier just analysed the data of the *Italian* market obtaining good results. To validate the choice of the confidence thresholds used previously and to verify the robustness of the classifier, I repeat the classification and validation process using data from the *British*, *Russian* and *German* markets.

British Market

The data preprocessing phase follows that already seen for the data of the *Italian* market. The database used is the *British* version of *Carspecs* and the tables from which the data were extracted are version and equipment. The dataset obtained has 293 vehicles and 14 columns:

- **Vehicle_id**: Vehicle identifier
- **JATORegSegment**: European regional segment assigned to the vehicle
- **Make**: vehicle manufacturer
- **Model**: vehicle model
- **RetailPrice**: selling price of the vehicle in *euro*
- **Bodytype**: encodes the shape and/or function of the vehicle bodywork
- **carlength**: vehicle length in *mm*
- **carwidth**: vehicle width in *mm*
- **carheight**: vehicle height in *mm*
- **wheelbase**: the distance between the central point of the front and rear wheels of the vehicle in *mm*
- **GVW**: encodes the absolute maximum the vehicle can weigh, including load, passengers, fuel and so on in *kg.*
- **engineLiters**: encodes engine displacement in *litres*

- **groundClearance**: encodes the maximum ground clearance of a vehicle with its empty weight in *mm*.
- **premiumness**: make premiumness

The split between train and test always follows the same 80:20 ratio obtaining a test set of 234 rows and a test set of 59. The approach followed to analyze the dataset is the same as that used for the *Italian* market and the measurements obtained are of the same type.

The test set accuracy obtained is the same as obtained for the *Italian* dataset:

Average test set accuracy: 0.84

The accuracy on the training set is also the same, showing also in this case overfitting:

Average training set accuracy: 0.98

The analyses on vehicle subsets bring results in line with those obtained for the Italian market:

- **Subset 1**(50% confidence threshold): 65% of the entire dataset (191 vehicles) with accuracy of 97%(5 vehicle misclassified)
- **Subset 2**(60% confidence threshold): 47%of the entire dataset (139 vehicles) with accuracy greater than 99% (1 vehicle misclassified)

Russian Market

The data is obtained from the *Russian Carspecs* database by applying the same procedures used for previous markets. The resulting dataset consists of 269 rows and 13 columns. Compared to the datasets used in the previously analysed, for this market I could not use the *premiumness* feature because it was not valued for 12 different makes for a total of 58 vehicles. Since for this type of information it was not possible to make an average or replace it with arbitrated values, I decided not to use this feature in the classification also because the feature was considered not very important by the previous classifiers. The split between train and test always follows the same 80:20 ratio obtaining a test set of 215 rows and a test set of 54.

Applying the same classification procedures seen so far, the results are as follows:

- **Average test set accuracy**: 0.77
- **Average training set accuracy**: 0.95
- **Subset 1**(50% confidence threshold): 52% of the entire dataset (141 vehicles) with accuracy of 96%(5 vehicle misclassified)
- **Subset 2**(60% confidence threshold): 36% of the entire dataset (105 vehicles) with accuracy of 100% (0 vehicle missclassified)

The results show that the overall classification accuracy has decreased, perhaps due to the lack of the *premiumness* feature which, even if in a small part, helps to carry out the classification. The results obtained on the subsets, on the other hand, are in line with those obtained in the markets previously analysed.

German Market

The data is obtained from the *German* carspecs database by applying the same procedures used for previous markets. The resulting dataset consists of 296 rows and 14 columns. The columns represent the same features present in the datasets of the *Italian* and *British* markets. The split between train and test always follows the same 80:20 ratio obtaining a test set of 236 rows and a test set of 60.

Applying the same classification procedures seen so far, the results are as follows:

- **Average test set accuracy**: 0.83
- **Average training set accuracy**: 0.97
- **Subset 1**(50% confidence threshold): 55% of the entire dataset (163 vehicles) with accuracy of 96%(6 vehicle misclassified)
- **Subset 2**(60% confidence threshold): 36% of the entire dataset (108 vehicles) with accuracy of 100%(0 vehicle missclassified)

The analyses carried out in the *German* market, once again, confirm the goodness of the classifier in obtaining excellent accuracy results in the identified vehicle subsets.

2.5 Classification algorithms comparison

The decision tree and random forest algorithms shown were chosen both for reasons related to the interpretability of the model and for reasons related to the quality of the classifier obtained, in particular for the random forest. During the search for the best classification algorithm to use to approach the segmentation problem, two other classification algorithms were tested:

- **Naive Bayes Classifier**
- **Support Vector Machine**

In this paragraph the implementations of the first two algorithms will be briefly shown and a comparison will be made between all the classification algorithms shown so far.

2.5.1 Naive Bayes classifier implementation

The dataset used is the one obtained by carrying out the steps shown in paragraph 2.2. This dataset was divided into training and test sets using the 80:20 ratio and keeping the distribution of classes equal in both sets. As result I obtained two datasets composed respectively of 234 and 59 vehicles. Unlike the decision tree and random forest cases, in this case the standardization of the features is carried out since the algorithm in question is sensitive to changes in the scale of the features used. Once the standardization is done, the algorithm is trained using the training set and then validated using the test set.

2.5.2 Support Vector Machine implementation

The dataset used is the same one adopted also in the previous cases. Also in this case it is necessary to standardize the features and an 80:20 division is made between training and test set, obtaining two datasets of 234 and 59 vehicles respectively.

The **Support Vector Machine** is a parametric algorithm that requires a phase of *hyperparameters* tuning. In order to perform the tuning, I used a *Grid Search* with *5-fold Cross validation* on the *C* and *gamma* parameters of the algorithm using the following vectors as lists of values:

- **C** = 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000

- **Gamma** = 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000

The **RBF kernel** was used to execute the algorithm since the classification problem does not appear to be a linear classification problem.

At the end of the *hyperparameters* tuning phase, the parameters with which the best result was obtained are:

- **C** = 10000.0
- **Gamma** = 0.0001

2.5.3 Results and Comparisons

The choice of using the Random Forest as a classification algorithm was given, as can be seen from the table 2.5, by the fact that it obtains the best in all the validation measures considered, despite having a high overfitting. Only the measures relating to the micro average data are reported, which is the most significant in the case of a dataset with unbalanced classes.

Algorithm	Test Accuracy	Train Accuracy	Precision$_\mu$	Recall$_\mu$	F1-score$_\mu$
DT	75%	99%	73%	73%	73%
RF	84%	98%	84%	86%	84%
NB	65%	73%	65%	65%	65%
SVM	78%	92%	79%	80%	80%

Table 2.5. Classification algorithms validation measures

Chapter 3

Vehicle similarity

In this chapter we will show how a vehicle similarity metric was defined, initially built at model level and then extended to version level.

3.1 Use case

The main reason for the importance of the *Carspecs* dataset is to allow Jato customers to make comparisons between vehicles. The makes use *Carspecs* data to carry out market research comparing the characteristics of vehicles similar to theirs. These comparisons are made using the technique called basket analysis.

Basket analysis consists in defining a set, 'basket', of vehicles considered similar to the vehicle that they want to compare and make a comparison between them. The choice of which vehicles to include in the basket is made by the makes that specify which ones they think are the competitors of the vehicle they want to compare. The vehicles that may be more similar for certain aspects, e.g. solely depending on physical criteria, may not always be present in the basket, for this reason it is useful to be able to identify an application-dependent similarity measure that is able to define the vehicles.

Basket definition can be a long process, especially at version level and may be operator dependent. The use of a vehicle similarity measure can be used, in another use-case scenario, not only by makes but also by private customers who need to compare similar vehicles in order to decide which one to buy. By defining a specific vehicle, the customer can obtain what vehicles are similar to it, compare them and decide which one is best for him. The possibility of being able to have a comparison at the version level, then, makes this tool even more useful to support the choice, allowing you to have a more detailed and precise comparison.

3.2 Introduction to similarity measures

In this paragraph I define the main distance measures focusing on those that will be used in this chapter. As described in the previous paragraph, the goal is to define a measure that indicates how similar two vehicles are.

The difference between distance and similarity measures is very subtle. The numerical value that returns a distance measure indicates how much an element is different from another, the one returned by a similarity measure indicates how similar an element is to another. In the case of distance measures, to identify similar elements, it is necessary to take into consideration those that return very small distance values, while in the case of similarity measurements, those that return very high values must be taken into consideration. Since both types of measures can be interpreted according to the concept of similarity, I will use the concept of similarity and distance measures interchangeably using mainly the definition of distance measures because it is the most used.

There are different types of distance measurements that can be used and it is important to know them in order to choose the best one according to the problems to be solved. When calculating the distance between two elements of a dataset it is possible that they have features of different data type, for example real values, categorical values or boolean values. In these cases, it is important to choose the distance measures that are able to handle data of different types.

The distances that are important to know for the rest of the chapter are the following:

- **Euclidean Distance**: calculates the distance between two real-valued vectors. It is particularly good for two rows of data that have floating point or integer values. If the features of the 2 vectors have different scales, usually the data are normalized or standardized before carrying out the distance calculation, in this way no features prevail over the others. The formula for calculating the Euclidean distance is:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_\mathrm{i} - q_\mathrm{i})^2 + ... + (p_\mathrm{n} - q_\mathrm{n})^2}$$

- **Manhattan Distance**: calculates the distance between two real-valued vectors. Unlike the Euclidean distance, it is more suitable for vectors of integers. The formula is the following :

$$d(x,y) = \sum_{i=1}^{n} |x_{\mathrm{i}} - y_{\mathrm{i}}|$$

- **Dice distance**: is a distance metric that applies to binary type values. A typical example of use is on categorical variables after the dummyfication process. The formula is:

  $$D = \frac{2x_1}{2x_2 + x_3 + x_4}$$

  Where:

  - $\mathbf{x_1}$ = number of dummies 1 for both individuals
  - $\mathbf{x_2}$ = number of dummies 1 for this and 0 for that
  - $\mathbf{x_3}$ = number of dummies 0 for this and 1 for that
  - $\mathbf{x_4}$ = number of dummies 0 for both

- **Gower Distance**: allows you to measure how different two samples are. Samples can have mixed, categorical, numerical, logical or text data features. The distance is always between 0 and 1. The Gower distance combines different metrics to be able to manage and calculate data of different types:

  - **quantitative**: Manhattan distance normalized by interval
  - **ordinal**: the variable is ranked first, then the *Manhattan distance* is used with a special adjustment for the ties
  - **nominal**: the variables are first converted using dummyfication and then the *Dice coefficient* is used.

  The general formula is :

  $$D_{\mathrm{Gower}}(x_1, x_2) = 1 - \left(\frac{1}{p} \sum_{j=1}^{n} s_{\mathrm{j}}(x_1, x_2) \right)$$

  with $s_j(x_1,x_2)$ as the partial similarity function computed separately for each descriptor.

3.3 Features Selection

The reference database, as in the previous paragraph, is the *Carspecs* data relating to the *Italian* market. As already mentioned, all the features that characterize the vehicle can be extracted from this dataset. The results obtained by the classifier in the previous paragraph, were used as starting points to choose which features to consider in this case. Given that the features used on that occasion led to good results in the subdivision of vehicles into categories, it is conceivable that those same features can provide reliable information in defining a measure of how similar a vehicle is to another. To these features other features have been added both in the case of similarity between models and in the case of similarity between versions. In these two cases, different attribute lists were used since, as mentioned in the first chapter, models and versions need a different level of detail to be uniquely identified, in particular to distinguish one version from another it is necessary to specify many characteristics of the vehicle that are not necessary in the identification of the model, such as the type of equipment or the number of doors. The choice of which features to adopt was also supported by discussion with the **SMEs** (Subject Matter Experts), who showed me which features were the most important in order to identify the vehicle, in particular regarding the identification of the version.

3.3.1 Model features

The goal of defining a similarity measure by considering first the models and then the versions was to be able to define in the first instance a measure based on the physical/performance attributes of the model then refining the concept of similarity to the versions integrating their peculiar characteristics at a later time. Considering the information obtained in the definition of the automatic segmentation tool and after a discussion with *SMEs* in order to define what features best represent the models according to the described requirements, we arrived at this list of features:

- **RetailPrice**: the price of the vehicle in *euro*, includes all national taxes. Local or state taxes are excluded, as are registration taxes, insurance and road taxes. Delivery charges and options price are excluded.
- **JATORegSegment**:European regional segment assigned to the vehicle
- **maxPwrkW**: the maximum engine output, given in *kilowatts (kW)*

- **acceleration**: the time in seconds for the vehicle to reach 100 *km/h* from a standing start.
- **maximumSpeed**: the maximum speed for a vehicle in *km/h*.
- **engineCC**: the engine capacity in cm^3, which is the total swept volume of the pistons.
- **fuelConsumption**: official fuel consumption *(l/100km)* data given in the manufacturer's documentation for the combined cycle (or equivalent).
- **CO2emissions**: the Carbon dioxide level *(g/km)* as published by the manufacturer for combined driving conditions measured in grams per kilometer.
- **carlength**: vehicle length in *mm*.
- **carwidth**: vehicle width in *mm*.
- **carheight**: vehicle height in *mm*.
- **wheelbase**: the distance between the central point of the front and rear wheels of the vehicle in *mm*.
- **FrimDiameter**: the diameter in *inches* of the front rim, without the tyre.
- **GVW**: encodes the absolute maximum the vehicle can weigh, including load, passengers, fuel and so on in *kg*.
- **Kerbweight**: the published weight of the vehicle before load, passengers, or any optional extras are added in *kg*, i.e. the empty weight with all the standard equipment.
- **cargocapacity**: the maximum cargo volume available measured with the rear seat up to bottom of the window.

In addition to the features listed above, information relating to vehicle sales volumes was collected from the *ModelMix* database. The volumes will not be used directly in the distance calculation but will be used as a sorting criterion for the most similar vehicles. The reason for this choice is dictated by the fact that the competitor with the most market is the one that is most interesting for the purposes of comparison.

The data were obtained following the same data preprocessing procedure seen in chapter 2. The database used included 8473 versions of vehicles belonging to 20 different segments. Some of the listed features had null values:

- **cargocapacity**: 108 null values on 8473 vehicle versions
- **fuelConsumption** : 324 null values on 8473 vehicle versions
- **acceleration**: 429 null values on 8473 vehicle versions
- **kerbweight**: 6 null values on 8473 vehicle versions
- **maximumSpeed**: 123 null values on 8473 vehicle versions
- **CO2emissions**: 45 null values on 8473 vehicle versions
- **Volumes**: 2037 null values on 8473 vehicle versions

All the null values, expect for the *Volumes* values, were replaced with the mean value of the feature in the model, if the model had no information on that feature the mean value of the segment was used. The entries relating to vehicles for which sales volumes were not available were eliminated, thus obtaining 6436 vehicle versions. Handled the null values of the dataset, the data were grouped at the model level, obtaining 270 models characterized by 17 features. This features are all continous except for the *JATORegSegment* feature that is categorical.

3.3.2 Version features

In addition to the features used to obtain similarity at the model level, other features have been added to allow to define a measure of similarity at version level:

- **trimClassification**: a classification to identify between similar trim levels across different countries and across different makes. This attribute can take the following values: **Base (B)**, **Medium (M)**, **Luxury (L)**, **Sport (S)**.
- **BodyType**: encodes the shape and/or function of the vehicle bodywork
- **NumberOfDoors**: the number of doors for a vehicle.

- **seatingCapacity**: the number of people that the vehicle can carry in standard configuration.
- **FuelType**: the primary fuel type used by the vehicle
- **powertrain**: the type of powertrain. This comprises the main system that drives / propels the vehicle.
- **TransmissionType**: the type of transmission fitted to a vehicle.
- **ConcludeDate**: indicates the end date of the sale of a specific version
- **DataDate**: indicates the date from which version data is available on the databases. it is indicative of the date the vehicle was placed on the market even if they do not always coincide perfectly.
- **DrivenWheels**: the wheels that drive a vehicle which can be front (F), rear (R) or 4WD(4)
- **TotalRetailAdjustment**: cumulative price in *euro* of the options included in the vehicle
- **ModelPrice**: retail price of the vehicle model in *euro*, calculated by averaging the retail prices of all versions of the same model

These features are of mixed type. There are continuous features and categorical features. Not all will be used directly by the similarity calculation algorithm, the *DataDate* and *ConcludeDate* features will be used in order to identify the versions that are sold in the same sales period as the vehicle being compared. Among the extracted data, no vehicle has null values in the version features listed above. At the end of the extraction, 6436 versions were obtained with 28 features that can be used for the calculation of similarity.

3.4 Selection of distance metrics

In order to define a similarity measure that works correctly it is important to choose the distance metrics that best suits the type of feature (continuous or categorical) we have available. Also in this case it is necessary to distinguish between the metrics chosen to define the similarity between models and the metrics chosen to define the similarity between versions.

3.4.1 Distance metrics per model

To choose the most suitable metric, it is necessary to take into account that the features that characterize this case are both continuous and categorical, for this reason the process of choosing the right distance metric is less immediate. The measure that best suits mixed type of feature is the *Gower Distance*. The choice fell on the *Gower distance* measurement also because in the calculation of the distance between categorical variables it uses the *Dice coefficient*.

The peculiarity of the *Dice coefficient* is that it considers the absence of features in both samples differently from the presence, so it does not consider as similar samples that share the absence of a characteristic but only considers similar those that have the same characteristic. Using dummy encoding of categorical variables, this is precisely the behaviour we want to obtain when we compare two versions of different vehicles since we are interested in considering two vehicles similar only if they have the same characteristics and not if the same characteristics are absent in both.

3.4.2 Distance metrics per version

As in the previous case, in order to choose the most appropriate distance measure, we must take into account the type of variables available. To calculate the similarity between versions we have mixed type features, continuous variables and categorical variables, for this reason also in this case I decided to use the *Gower distance*.

The use of the *Dice coefficient* in the calculation of the distance between categorical features assumes greater relevance in this case given the more numerous presence of categorical features used to identify the version.

3.5 Distance algorithm application

In this paragraph we will see how the distance measure selected in the previous paragraph were used to calculate the similarity between models first, and then between versions.

3.5.1 Model distance algorithm

The model dataset consists of 270 models characterized by 17 features. We want to calculate the similarity of each model with all the others. The concept of similarity is a concept that can be applied on different levels and can have different objectives based on the similarity criterion to be adopted. In this case we want to obtain different types of similarity, one general that takes into account all the features and others limited to subspaces of features of the same type. The features used in the similarity calculation are all those listed in paragraph 3.3.1 except the volumes, therefore 16 features, which have been divided into the following subspaces:

- **Price**: *RetailPrice*
- **Performance**: *maxPwrkW*, *acceleration*, *maximumSpeed*, *engineCC*
- **Emissions**: *fuelConsumption*, *CO2emissions*
- **Dimensions**: *carlength*, *carwidth*, *carheight*, *FrimDiameter*,*wheelbase*
- **Weight**: *GVW*, *kerbweight*
- **Capacity**: *cargocapacity*

Each of these subspaces identifies a coordinate according to which the similarity between vehicles can be defined. For each subspace of features, the *Gower distance* between all vehicles was calculated, obtaining six different similarity measures, each of which indicates similarity for a given type of attributes. The presence of different similarity criteria allows to identify similar vehicles according to different needs allowing customers to compare them on several floors.

To define a general similarity measure that takes into consideration all the vehicle characteristics and therefore indicates an overall similarity between two models, we have chosen to perform the calculation of the *Gower distance* using the 16 features available for our models.

For each of the calculated distances, weights were used in order to specify the importance of the features for the calculation of the similarity. Due to the fact that this is an unsupervised problem, therefore there are no labels that can help us in the training phase. The definition of the algorithm, including the values to be assigned to the various weights, was obtained downstream of a process of qualitative training and testing. For this type of training the measurement algorithm was performed several times on a small fixed set of vehicles. The training set used consisted of 27 models for each of which a list of competitors was provided by the *SMEs*. The models were chosen to represent all the segments available in the dataset. The competitors available for each model varied in number and range from 1 to 16 for a total of 151 competitors available with an average of 5.5 competitors per model. To verify the functioning of the algorithm in order to apply corrections, a **Top 5 accuracy** measure was used, defined as follows:

- the number of competitors available for each model is computed and summed together, if a model has more than 5, only 5 competitors are considered in the sum. For each model, the 5 most similar models are calculated and how many competitors are present are counted. The numbers of competitors found among the 5 most similar models for all models are added up and the result is divided by the sum of competitors calculated before.

$$Top5\ Accuracy = \frac{competitors\ identified\ in\ the\ top5\ similar\ models}{sum\ of\ competitors\ available}$$

At each iteration, the weights of the distance algorithm are modified and the accuracy value calculated. The goal was to optimize the top 5 accuracy value by obtaining the best possible combination of weights. The best accuracies achieved in the training phase are shown in the table 3.1.

Top 5	Top 10	Top 15
85%	90%	95%

Table 3.1. Top5, Top 10, Top 15 training accuracies in model similarity measure

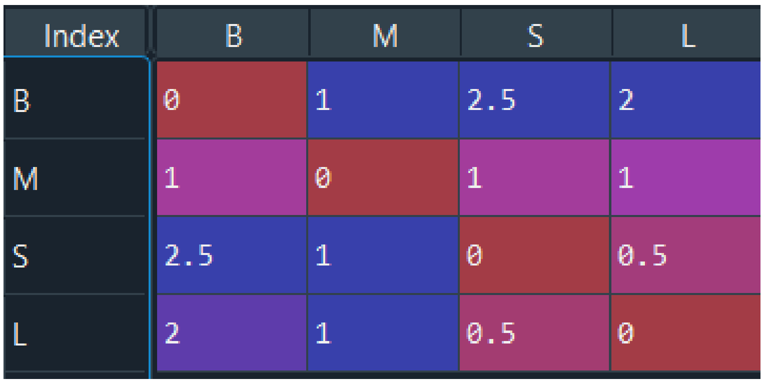
3.5.2 Version distance algorithm

The number of versions available in this case is 6436 characterized by the 28 features listed above. As in the distance calculation for models, the features are divided into subspaces of features in order to calculate the similarity according to different criteria. In addition to those already considered, the following subspaces are added:

- **Price**: *RetailPrice*, *ModelPrice*
- **Equipment Cost**: *TotalRetailAdjustment*
- **VersionFeatures**: *NumberOfDoors*, *seatingCapacity*, *BodyType*, *FuelType*, *powertrain*, *DrivenWheels*, *TransmissionType*, *JATORegSegment*

The calculation of the distance between versions based on these subspaces is carried out using the *Gower measure*, in this way I can simultaneously manage categorical features and continuous features in the same distance calculation, in particular in the *VersionFeatures* subspace I consider categorical all the features except *NumberOfDoors* and *seatingCapacity*. I use the number of doors and the passenger capacity as continuous variables because I not only want to know if both versions have the same value in these two features but it is important to know, in case the vehicles being compared do not have the same value, how much the values differ because, for example, the distance between a vehicle that has 5 seats and one that has 4 is less than that of a 5-seat vehicle with a 2-seat.

The features listed in the subspaces do not include the one relating to the trim level. For this features I defined a matrix that indicates the distances between the various types of trim level, figure 3.1.



Index	B	M	S	L
B	0	1	2.5	2
M	1	0	1	1
S	2.5	1	0	0.5
L	2	1	0.5	0

Figure 3.1. Trim level distance matrix

The procedure for calculating the similarity takes place in several steps:

- For each version for which I have to calculate the distance, I filter the dataset of vehicles with which to make the comparison taking into consideration only vehicles that were sold in the same period by comparing the values of sale and withdrawal from the market indicated by the *DataDate* and *ConcludeDate* attributes.

- I use the *Gower* formula to calculate the similarity according to all the subspaces, obtaining the similarity values for each of them. Weights have been associated with the features within the subspaces to indicate their importance in the calculation of similarity. The values of the weights were assigned following the indications of the domain experts and using the information obtained from the training of the similarity measure of the models. For example, for the features of the *Price* subspace, the *RetailPrice* attribute has an higher weight compared to the *ModelPrice* attribute, this is because the *ModelPrice* helps us to identify models with a similar value to that of the vehicle we are comparing but to obtain the most similar version from the price point of view, the retail price of the version is more important. In the calculation of the similarity in the *VersionFeatures* subspace, the distance relative to the *trim level* has also been added using as distance values those contained in the table 3.1.

- To obtain the overall similarity measure that takes into account all the mentioned characteristics, a linear combination is performed in order to obtain the Version Similarity which therefore includes all the distances of the subspaces seen so far. Also in this case weights are assigned to indicate the importance of the various components in the distance calculation
- At this point it is possible to order the similar versions according to different criteria, as many as the subspaces for which the similarity has been calculated. Once the sorting criterion has been chosen, it is possible to filter the similar versions obtained, keeping only the most similar version of each of the present models.

3.6 Validation

In this paragraph the results obtained by applying the similarity algorithms by model and by version will be shown. For both cases a method based on the identification will be used, within similar vehicles, of those that are considered market competitors.

3.6.1 Model Validation

A subset of vehicles belonging to different segments will be selected in order to see if the measurement works the same way in all segments. The vehicles available are 270 divided into 20 segments according to an uneven distribution.

The validation process conceived is based on the given definition of similar vehicles, i.e. vehicles that appear to be market competitors are considered similar. The validation phase follows the following steps:

- I randomly select an arbitrary number of models defining a validation subset.
- For each vehicle belonging to the validation set I get a list of competitors from a *SME*.
- For each vehicle in the subset, I apply the algorithm in order to create an ordered list of similar models according to the overall Model Similarity measure defined before.
- For each vehicle in the validation subset, I take into consideration the first 5,10 and 15 places on the list of similar vehicles.
- I define *Top 5*, *Top 10* and *Top 15* accuracy measures that specifies how many of the controlled models were actually competitors:

$$Accuracy = \frac{competitors\ found}{competitors\ available}$$

I randomly selected 20 vehicles distributed evenly among the various segments and for each of these a *SME* have defined a list of competitors. The number of competitors for each vehicle is not fixed and varies from 1 to 15 , obtaining 141 total competitors.

The selected models models with their segment and the number of competitors available for each model are listed below:

- **Jeep Grand Cherokee - E1 SUV**: 7 competitors
- **Jaguar XE - D2**: 6 competitors
- **Cupra Formentor - C2 SUV**: 5 competitors
- **Skoda Kamiq - B SUV**: 15 competitors
- **Citroen C4 Spacetourer - Mini MPV**: 3 competitors
- **KIA Niro - C2**: 3 competitors
- **Opel Zafira - Medium MPV**: 3 competitors
- **Porsche Panamera - E2**: 4 competitors
- **KIA Picanto - A**: 6 competitors

- **Peugeot 508 - D1**: 6 competitors
- **Hyundai Tucson - C1 SUV**: 15 competitors
- **Volvo S90 - E1**: 6 competitors
- **Land Rover Discovery Sport - D2 SUV**: 8 competitors
- **Toyota RAV4 - D1 SUV**: 11 competitors
- **Lexus LC - Sports**: 2 competitors
- **Ford Puma - B SUV**: 15 competitors
- **Skoda Scala - C1**: 12 competitors
- **Ford Galaxy - Large MPV**: 1 competitors
- **Nissan Micra - B**: 14 competitors
- **Mercedes GLS - E2 SUV**: 3 competitors

By applying the described validation procedure for this subset, the results described in the table 3.2, which analyzes the accuracy values for the individual models, and in the table 3.3, which represents the general accuracy value of the algorithm, are obtained.

Model	Top 5 Acc	Top 10 Acc	Top 15 Acc
Jeep Grand Cherokee	60%	71%	100%
Jaguar XE	60%	83%	100%
Cupra Formentor	60%	100%	100%
Skoda Kamiq	60%	70%	60%
Citroen C4 Spacetourer	67%	100%	100%
KIA Niro	67%	100%	100%
Opel Zafira	67%	100%	100%
Porsche Panamera	33%	100%	100%
KIA Picanto	80%	100%	100%
Peugeot 508	80%	83%	83%
Hyundai Tucson	80%	80%	100%
Volvo S90	80%	100%	100%
Land Rover Discovery Sport	80%	88%	100%
Toyota RAV4	80%	90%	100%
Lexus LC	100%	100%	100%
Ford Puma	100%	70%	67%
Skoda Scala	100%	90%	92%
Ford Galaxy	100%	100%	100%
Nissan Micra	100%	90%	79%
Mercedes GLS	100%	100%	100%

Table 3.2. Top 5, Top 10, Top 15 accuracy values per model in model similarity validation phase

Top 5	Top 10	Top 15
80%	91%	92%

Table 3.3. Top 5, Top 10, Top 15 accuracy validation model similarity

3.6.2 Version Validation

The validation relating to the measure of similarity between versions is more complex than that relating to models. As already mentioned, a version of a model differs from the others by the presence of many attributes that vary its characteristics, sometimes significantly, leading a version of a specific model to compete with versions of models belonging to different segments or that do not belong to its competitors models. One of the features that can introduce more variation in the models is the *trim level* that allows you to divide the versions into 4 macro categories:

- **B** : Base
- **M** : Medium
- **L** : Luxury
- **S** : Sport

This great variability complicates the validation process which can no longer be based solely on the concept of similarity given by market competitors because the most similar version does not always belong to a competitor model. For this reason the validation of the similarity measure at the version level is carried out on two levels:

1. Validation following the competitor model criterion
2. Validation based solely on version characteristics

Validation following the competitor model criterion

In this first part of the validation I use the same criterion used in the model validation phase by performing the following steps:

- I take into consideration the 20 models randomly selected in the previous phase obtaining a test set of models distributed equally among the various segments.
- For each of the models in the test set, I randomly select a version belonging to each of the trim levels available for that model.
- For each selected version, I run the similarity algorithm.
- I take into consideration the first 5, 10 and 15 most similar versions and check how many of these versions belong to competing models.
- Calculation of Top 5, Top 10 and Top 15 accuracy

I repeat this procedure 10 times, selecting different versions of the test models at each iteration, this is because given the great variability in the characteristics that identify a version, the choice of which version to consider in the validation changes the result obtained considerably. For this reason the process is repeated 10 times, selecting sets of different versions, calculating the average values of the obtained accuracies.

The choice to select a version for each of the available trim levels is dictated by the fact that, as previously mentioned, the models with which the vehicle is compared may also vary as the trim varies. This behavior must be taken into account in order to be able to analyze the results obtained more precisely. The versions obtained from this random selection are 57 divided according to the trim levels in this way:

- **B** = 19
- **M** = 15
- **L** = 12
- **S** = 11

In the table 3.4, we see the *Top 5* accuracy results obtained. It can be seen that for the B and S trims the accuracy values are lower and how only in those cases there are versions in which the accuracy is equal to 0. This behavior is particularly evident in the case of the versions of the S trim, that is the trim more sporty. This behavior is due to the fact that typically the sport trims have characteristics that make the vehicle quite different in terms of performance and value, leading it to compare with versions of models that do not belong to the same segment or that are not considered competitors.

Trim	Mean accuracy	Std accuracy	Min accuracy	Max accuracy
B	65%	27%	0%	100%
L	72%	28%	12%	100%
M	71%	27%	7%	100%
S	59%	32%	0%	100%

Table 3.4. Version similarity Top 5 accuracy divided by trim

In the table 3.5, instead, we see the *Top 10* accuracy results. The accuracy values have grown for all the trims and it can be seen that in this case the results obtained are similar for all four categories analyzed. This shows us that, even for trims B and S on which there is greater difficulty in identifying competitors, if the analysis is expanded to the first 10 vehicles, the performance is in line with the other trims.

Trim	Mean accuracy	Std accuracy	Min accuracy	Max accuracy
B	77%	19%	37%	100%
L	79%	24%	22%	100%
M	79%	21%	25%	100%
S	77%	22%	22%	100%

Table 3.5. Version similarity Top 10 accuracy divided by trim

Finally, the table 3.6 shows the *Top 15* accuracy results. Again, we get higher accuracy and similar values for all trims.

Trim	Mean accuracy	Std accuracy	Min accuracy	Max accuracy
B	86%	14%	58%	100%
L	83%	15%	55%	100%
M	86%	14%	65%	100%
S	86%	16%	53%	100%

Table 3.6. Version similarity Top 15 accuracy divided by trim

Validation based solely on version characteristics

In this validation phase, it is not considered whether or not a vehicle belongs to the competitors of the vehicle being compared, but, for each version considered similar, an attempt is made to understand whether there are other versions of the same model that are closer to the vehicle that he is confronting. To understand if one version is more similar to another, I used some of the most characteristic attributes of the version such as:

- Version Price
- Model Price
- Number of doors
- Seating capacity

- Trim level
- Fuel type
- Power train
- Transmission type
- Body type

Another problem that characterizes this type of validation is the lack of labels and therefore the need to carry out a manual validation. In order to make this manual validation easier, the similarity algorithm is set in such a way as to consider during the calculation only versions of vehicles that have *BodyType*, *PowerTrain* and *TrimLevel* equal to those of the version being compared, in this way the number of versions to be analyzed narrows and the behavior of the algorithm is verified in a more punctual and precise way.

The steps of this validation phase are as follows:

- Starting from the test set of 20 models used in the previous phases, N versions are selected at random
- For each of these versions, the similarity algorithm is performed to identify the most similar versions.
- For the first 10 models that appear among the similar versions, I check whether the version considered most similar for each model is actually the one that is closest to the vehicle being compared among that model versions. The criterion for which a version of a model is considered more similar than another of the same model is given by the presence of a greater number of version features equal to or closer to those of the version with which the comparison is being made.
- I count the number of times in which the most similar version of the model is correctly identified and, for each test version considered, I extrapolate an accuracy value given by the ratio between:

$$Accuracy = \frac{similar\ versions\ correctly\ identified}{models\ considered}$$

- Calculation of the average accuracy value obtained from the analyzed test versions.

Given the need to manually validate the results, 2 versions were taken for each of the 20 models available for the test, obtaining 40 versions, and the steps just shown are applied to these.

The average accuracy value obtained is 86%.

Chapter 4

Conclusions and future works

4.1 Vehicle segment classifier conclusions

The chapter 2 showed the development of a classifier capable of providing support to Jato researchers when assigning the segment to vehicles. The model developed is based on a random forest algorithm that allows to obtain an automatic classifier to support the choice of the segment. A total of 20 classes were predicted. The first model achieves an 84% accuracy on a set of 293 models. The results were limited by the fact that the total dimension of the population (the total model population) is scarce and by the fact that in some cases segmentation may be driven by commercial considerations difficult to capture. The classifier may thus be employed as a support to classification, but not as a standalone classifier. For this reason, it was interesting to identify a subset of models for which the classification is almost certain, on which a standalone classifier could be applied. A threshold on the confidence of the classification was thus set, and automated classification was performed on the high confidence set. The confidence thresholds used were 50% and 60%, these values were chosen following a heuristic approach using the *Italian* regional dataset as a training set. On the high confidence subsets, the accuracy of the model reaches more than 96%. In order to validate the choice made, the same analyses were also applied to the *British*, *Russian* and *German* markets, obtaining similar results demonstrating the robustness of the model.

Important information was also obtained regarding the importance of features in vehicle characterization. We have seen how some features such as *carlenght* and *bodytype* are important for the separation of models into segments, while others such as *premiumness* are a little less. This information were used as a basis for creating the distance measurement between, models first and versions later, in the third chapter.

Tables 4.1 and 4.2 show the accuracy results obtained by the classifier.

Classificator	Dataset Region	Test Accuracy	Train Accuracy
Decision Tree	IT	77%	99%
Random Forest	IT	84%	98%
	GB	84%	98%
	RUS	77%	95%
	D	83%	97%

Table 4.1. DT and RF classification results in the analysed market.

Region	S1 Dimension	Accuracy S1	S2 Dimension	Accuracy S2
IT	180 samples	97%	124 samples	99%
	61% dataset		44% dataset	
GB	191 samples	97%	139 samples	99%
	65% dataset		47% dataset	
RUS	141 samples	96%	105 samples	100%
	52% dataset		36% dataset	
D	163 samples	96%	108 samples	100%
	55% dataset		36% dataset	

Table 4.2. Random Forest results on dataset subsets. S1 is the subset containing vehicles that are classified with greater than 50% confidence, S2 with greater than 50% confidence.

4.2 Vehicle similarity conclusions

The second part of the work was aimed at allowing to systematically define the vehicles that are considered similar to a given vehicle in order to be able to define more precise baskets for basket analysis. Two similarity measures were developed to identify similar vehicles at model level and similar vehicles at version level. Given the presence of categorical and continuous features, the proposed solution is based on the *Gower distance* measurement that allows to manage both types of variables by combining the use of the *Dice coefficient* for categorical features and the *Manhattan distance* for continuous features. The similarity measures obtained were validated separately, however, both based on the same test set of 20 models selected randomly but equally distributed among the available segments.

4.2.1 Model similarity validation

The search for competitors within the similar models identified by the similarity algorithm resulted in the accuracies shown in the table 3.3. The results obtained are satisfactory because they allow to create baskets of 5 vehicles with at least 80% of competing vehicles and baskets of 10 vehicles with at least 91% of competing vehicles.

4.2.2 Version similarity validation

As seen, the validation was based on two different strategies obtaining the results shown in paragraph 3.6.2. The most interesting result concerns the second part of the validation based mainly on the version characteristics. Here the applied validation shows how in 86% of the cases analyzed the algorithm is able to correctly identify for each model the version most similar to the vehicle being compared. This is a satisfactory result that allows us to identify with good precision the versions to be used as a comparison in basket analysis. However, the type of validation used is a bit weak because it appears to be subjective and based on considerations given by the undersigned and not by a domain expert.

4.3 Future works

To improve the accuracy of both tools developed, it would be necessary to work on the data available, trying to expand the number and the quality.

As regards possible future developments only in the case of similarity measures, two improvements can be made:

- **User interface:** create a user interface that allows to select models and versions in order to identify vehicles similar to those selected. This could expand the use cases of these tools.

- **More robust validation phase**: the second part of the validation of the similarity measure between versions should be improved by defining a more robust validation based on more objective parameters or considerations by domain experts

Bibliography

[1] B. Peterson. *"Car Make and Model: What Does it Mean?"*. 2021. URL: https://www.valuepenguin.com/auto-insurance/car-make-model

[2] *"What are Car Options?"*. 2013. URL: https://www.kbb.com/what-is/car-options/

[3] R. Wolff. *"Classification Algorithms in Machine Learning: How They Work"*. 2020. URL: https://monkeylearn.com/blog/classification-algorithms/

[4] K. Goyal. *"Data Preprocessing in Machine Learning: 7 Easy Steps To Follow"*. 2020. URL: https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/

[5] L. Govoni. *"L'importanza del ridimensionamento dei dati nei problemi di machine learning"*. 2020. URL: https://lorenzogovoni.com/ridimensionamento-dei-dati/

[6] G. Yufeng. *"The 7 Steps of Machine Learning"*. 2017. URL: https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e

[7] J. Jordan. *"Hyperparameter tuning for machine learning models"*. 2017. URL: https://www.jeremyjordan.me/hyperparameter-tuning/

[8] *"How to Choose Evaluation Metrics for Classification Models"*. 2020. URL: https://www.analyticsvidhya.com/blog/2020/10/how-to-choose-evaluation-metrics-for-classification-model/

[9] A. Mishra *"Metrics to Evaluate your Machine Learning Algorithm"*. 2018. URL: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning\
-algorithm-f10ba6e38234

[10] W. Koehrsen *"Beyond Accuracy: Precision and Recall"*. 2018. URL: https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

[11] C. Sehra *"Decision Trees Explained Easily"*.

2018. URL: https://chiragsehra42.medium.com/decision-trees-explained-easily-28f23241248

[12] Nagesh Singh Chauhan *"Decision Tree Algorithm, Explained"*. 2020. URL: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[13] T. Yiu *"Understanding Random Forest"*. 2019. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[14] M. Mithrakumar *"How to tune a Decision Tree?"*. 2019. URL: https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680

[15] J. Brownlee *"A Gentle Introduction to k-fold Cross-Validation"*. 2020. URL: https://machinelearningmastery.com/k-fold-cross-validation/

[16] W. Raseman *"Multivariate Distances: Mahalanobis vs. Euclidean"*. 2018. URL: https://waterprogramming.wordpress.com/2018/07/23/multivariate-distances-mahalanobis-vs-euclidean/

[17] J. Brownlee *"4 Distance Measures for Machine Learning"*. 2020. URL: https://machinelearningmastery.com/distance-measures-for-machine-learning/

[18] *"Mahalanobis distance"*. URL: https://en.wikipedia.org/wiki/Mahalanobis_distance

[19] D. Anand *"Gower's Distance"*. 2020. URL: https://medium.com/analytics-vidhya/gowers-distance-899f9c4bd553

[20] *"Dice coefficient"*. 2020. URL: https://stats.stackexchange.com/questions/55798/what-is-the-optimal-distance\
-function-for-individuals-when-attributes-are-nomina/55802#55802