Decoding An Automobile's Technical Specification From Its Identification Number

Mantas Mačiūnas, Viktor Skorniakov

Vilnius University Faculty of Mathematics and Informatics,
Naugarduko str. 24, Vilnius
mantas.maciunas@mif.stud.vu.lt, viktor.skorniakov@mif.vu.lt

Abstract. We propose a method for decoding technical vehicle's parameters (make, model name, body type etc.) from its identification number. Classification is done for the entire specification at once, thus utilising the underlying dependencies between labels. To achieve the goal, several models were used – nearest neighbours, decision tree, extra trees and random forest classifiers.

Keywords: vehicle identification number, multi-label classification, random forest classifier, decision tree classifier, extra trees classifier, k-nearest neighbours classifier.

1. Introduction

In the 1980s, there was a serious attempt at standardising the identification of any particular vehicle by introducing a 17-character vehicle identification number (VIN). Despite the goal being standardisation, nowadays, several VIN standards are in use. Furthermore, even if some manufacturers use the same VIN structure, they are not obligated to encode the same information and the same symbols may carry different information. Therefore, VIN structure is very inconsistent among manufacturers, and there is no easy way to decode it. Although there are plenty of free specification decoder tools on the internet, they frequently are applicable to only one make or a group of makes (for example, "Volkswagen" model VINs are encoded analogously to those of "Audi" or "Seat" because all of these manufacturers belong to "Volkswagen Auto Group"). The most frequently used method of decoding – a VIN lookup table. Using it one can decode certain technical parameters [1]. Although this method can be very accurate, it is tremendously inefficient in the long run since each new make comes with a new VIN structure. Furthermore, every time a manufacturer updates one

DOI: https://doi.org/10.15388/LMITT.2025.15

of its current models, the lookup table has to be updated as well. Moreover, manufacturers are not inclined to freely share their VIN encoding schemas. In most cases, owners of a particular make of vehicle collectively analyse and figure out the meaning of the VIN symbols on their cars. Lastly, although universal VIN decoder tools exist, most of them are closed-source, so it is impossible to figure out how they work or suggest any improvements. This research paper offers an alternative – VIN decoding using machine learning methods. To achieve the goal, multi-label machine learning models were used since they are capable of utilising the underlying correlations and dependencies between the technical parameters, thus enabling the model to make more accurate predictions and avoid impossible parameter combinations. Conducting the literature analysis, we did not encounter articles considering the decoding of VINs. Therefore, the research direction pursued by us exhibits substantial novelty.

The rest of the paper is organized as follows: Section 2 describes the data, Section 3 focuses on the methodology, Section 4 describes the results; the concluding section is devoted to the summary.

2. Data

The VIN of an automobile is an international coding standard that appeared in the 1980s. It is a collection of 17 letters and numbers, consisting of three obligatory parts, in which the encoding of the manufacturer, model and serial number is mandatory. The manufacturer is identified by the first three VIN symbols, often referred to as WMI (world manufacturer identifier). Symbols in positions 4-8 define the model and its specification. However, this requirement is enforced very loosely: some manufacturers only encode the model name, whereas others encode everything from the model name to a specific engine. The standards of the European Union (and the majority of the rest of the world) and North America differ as well. From the 9th symbol onwards European manufacturers must encode the VIN in such a manner that it becomes unique, i.e. encode a serial number and (sometimes) encode more specific technical data. Meanwhile, the North American VINs' ninth symbol is used as a check symbol – ensuring the rest of the VIN is entered correctly; symbols 10-11 contain the model year and the plant in which the vehicle was manufactured; finally, from the 12th symbol onwards follows the serial number of the vehicle.

Data from various European national automotive registries containing the VINs and technical specifications of automobiles was used in this research paper. The manufacturers' list included "Toyota", "Lexus", "Seat", "Cupra", "Mercedes-Benz" and "BMW". The technical parameters chosen for classification training were the ones that were present in all datasets: make, model name, model generation code, body type, fuel type, engine displacement and engine power. Some labels, especially those of model names and generation codes, were tremendously rare. The direct cause of this was an inherent data disbalance in the dataset. The usual methods of solving this issue (for example, data augmentation) were inapplicable, because the data contained many cases where insignificant differences between VIN symbols (for example, different serial numbers) resulted in different label sets. Therefore, the best way to resolve the data disbalance was to create models that are insensitive to outliers and are able to accurately classify even rare data values.

One unit of measurement was picked for each measurable technical parameter. For example, the engine displacement values in some sources were in cubic centimetres, whereas in others – in litres. Because cubic centimetres are a more accurate unit of measurement, they were chosen as the unit for engine displacement. Records having displacement in litres were converted into cubic centimetres. In case of unsuccessful conversion, the data row was dropped. Kilowatts were selected as the unit of measurement for engine power.

Finally, we had to decide how to cope with the rare data instances. As mentioned before, data synthesis was too risky because this method might mislabel generated VINs, thus confusing the models that will be trained on the data. Another frequently utilised method – dataset size reduction to the least popular label – was also inapplicable. This method reduces the dataset by finding the least common label and removing values of all other labels until their frequency is the same as of this label. However, because this problem involves thousands of possible label vector values, leaving just a few of each value would very likely result in underfitted models. Furthermore, these rare VIN values reflect an essential characteristic of the population. Namely, these are either low-production models or very old (at least 30 years old) automobiles. Low-production model VINs are tremendously difficult to find, because any single one of the data sources used in this research (mainly European national automobile registries)

could contain a handful of these VINs at best, and there was insufficient time to expand the dataset to include more sources. Old cars, meanwhile, provide a similar challenge – 30 years after rolling out of the factory very few cars still drive on the road. According to the European Automobile Manufacturers' Association, the mean age of a car registered in Europe was only 12 years old in 2023. Thus, it is entirely likely that vehicles that are 30 or more years old constitute the absolute minority of all vehicles on the road. These challenges left only one solution for the data disbalance – the removal of rare values from the dataset. The rarity threshold was set to 10. After removing these values, the dataset contained 2,611,885 entries.

3. Methodology

3.1. Classifiers

These multi-label classifiers were used for modelling: random forest classifier (RFC), decision tree classifier (DTC), extra trees classifier (ETC) and k-nearest neighbours classifier (MLkNN).

Multi-label random forest classifer differs from its single-label sibling in two ways: 1) different metrics that are used for node splitting during training; 2) predictions are made differently [2]. When splitting a node, this classifier calculates the value of the splitting criterion for each label vector coordinate separately and uses the mean of these values to determine splits. This manner of splitting ensures that the tree is optimised to best predict label vectors rather than separate label vector coordinates [3]. Because the leaves are vectors, the prediction of a single tree is a full technical specification vector. Whenever the forest is supplied a VIN it has never seen before, the prognosis is produced by the process of voting [4].

A decision tree is a special case of a random forest classifier where the forest consists of only one tree. The main difference is that the random forest trees are fitted by using a randomly sampled dataset, thus creating a decorrelated tree ensemble, whereas a decision tree is fitted with the entire training dataset.

Extra trees classifiers work analogously to random forest classifiers, but they use an additional randomisation that is much stronger than those of the previously described classifiers [5].

The multi-label case of the k-nearest neighbours classifier is the same as the single-label one. This research paper utilises two search algorithms for the MLkNN model – K-D tree and ball tree [6], [7], [8].

3.2. Utilised software tools

The code used for this research paper was written using the "Python" programming language, version 3.10.16. All machine-learning methods were coded using the "scikit-learn" library.

3.3. Model accuracy metrics

The classification accuracy metrics for multi-label models differ significantly from those meant for single-label models. We utilised 6 accuracy metrics for models' evaluation and choice. Two of these metrics are meant for separate labels' vector coordinate scoring accuracy evaluation, whereas the remaining four are meant to evaluate the accuracy when classifiying the entire label vector. First, the base formulae that are used for all the aforementioned accuracy metrics need to be defined. The indicator function is given by equation

$$1_{a=b} = \begin{cases} 1, if\ a = b \\ 0, if\ a \neq b. \end{cases}$$

Fractions of true positive predictions (*TP*), false positive predictions (*FP*) and false negative predictions (*FN*) are defined by equations

$$TP_C = \frac{1}{|C|} \sum_{c \in C} 1_{\hat{y}(c)=C'},$$

$$FP_C = \frac{1}{|N \backslash C|} \sum_{c \in N \backslash C} 1_{\hat{y}(c)=C'},$$

$$FN_C = \frac{1}{|C|} \sum_{c \in C} 1_{\hat{y}(c) \neq C'},$$

where *C* is one class of the classified coordinate, *N* is the union of all classes that are present in the dataset, and *C'* is a label denoting that the given data entry belongs to the class *C*.

Accuracy and $F1_{macro}$ for evaluating the single labels' vector coordinate classification, are defined as follows:

$$Accuracy(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} 1_{\hat{y}_i = y},$$

$$F1_{macro}(y_L, \widehat{y_L}) = \frac{(1+1^2)TP_L}{(1+1^2)TP_L + FP_L + 1^2 FN_L},$$

$$F1_{macro}(y, \hat{y}) = \frac{1}{|L|}\sum_{l \in L} F1_{macro}(y_l, \hat{y}_l),$$

The second formula is devoted to a single label vector coordinate class L, whereas the third formula is the combined $F1_{macro}$ for a single coordinate of the label vector.

The remaining metrics are for the evaluation of the overall accuracy of the model. Two of these are the aforementioned accuracy and $F1_{macro}$, modified to fit multi-label classifiers:

$$Accuracy(Y, \hat{Y}) = \frac{1}{n}\sum_{y \in Y^t} Accuracy(y, \hat{y}),$$

$$F1_{macro}(Y, \hat{Y}) = \frac{1}{n}\sum_{y \in Y^t} F1_{macro}(y, \hat{y}),$$

here *y* is a single coordinate column, the amount of coordinates is labeled *n* and *Y* is the set of all label vectors, represented as a matrix, where one column is for a single technical parameter. So, the overall classification accuracy and $F1_{macro}$ are calculated by averaging the metric calculated for each separate technical parameter. Finally, two additional accuracy metrics devoted to multi-label models were used – *Hamming* loss and exact-match score [9]:

$$HammingLoss(Y, \hat{Y}) = \frac{1}{N}\sum_{i=1}^{N} \frac{|y_i \triangle \hat{y}_i|}{n},$$

$$ExactMatch(Y, \hat{Y}) = \frac{1}{N}\sum_{i=1}^{N} 1_{y_i = \hat{y}_i},$$

where *N* denotes the amount of rows in *Y*, i.e. the amount of data instances in *Y*.

4. Results

The full dataset was split into training and testing subsets by iterative stratification, which prioritises the least common labels and starts by

stratifying those first [10]. All models were trained with the same dataset (size – 2,089,508 rows), and all accuracy metrics were calculated after the trained models classified the same testing dataset (size – 522,377 rows). The optimal parameters for the models were chosen by using 10-fold cross-validation on a subset of the full dataset. This subset was created by randomly taking 10 of each unique row in the dataset.

As we can see in the first table, all models reached a quite high overall classification accuracy (>0.88). However, looking at the $F1_{macro}$ it becomes quite obvious that not all models were capable of classifiying outliers equally well. The extra trees classifier in particular is the worst in this regard with only 0.763 $F1_{macro}$. The exact-match score also separates the extra trees classifier as the worst of the bunch – it managed to correctly decode the technical specification of only 59.5% of the testing data VINs. Therefore, for further work it is recommended to skip this model and use one of the three remaining ones – random forest classifier, decision tree or k-nearest neighbours classifier. For further analysis the decision tree was chosen as it beats both of the remaining models with respect to all metrics and manages to correctly predict the technical specification of more than 4 out of 5 VINs in the testing dataset – an exact-match score of 82.2%.

Table 1. The accuracy metrics for all models.

Classifier	Classification accuracy	$F1_{macro}$	Exact-match score	Hamming loss
RFC	0.932	0.870	0.747	0.068
DTC	**0.950**	**0.923**	**0.822**	**0.050**
ETC	0.884	0.763	0.595	0.116
MLkNN	0.930	0.878	0.764	0.070

The second table shows the accuracy metrics for all technical parameters separately when using a decision tree. All parameters are classified with at least 85% accuracy. The classification of an automobile‘s make is especially accurate – the model only made mistakes for “Seat“ and “Cupra“ VINs (99.5% and 91.0% of “Seat“ and “Cupra“ VIN codes respectively had their make labeled correctly). The $F1_{macro}$ of the fuel type is quite a bit lower than that of other technical parameters – while analysing incorrectly labeled VINs it was noticed that the model classified VINs, where fuel type was labeled

as “Gasoline/LPG“, poorly – only 10.2% of such VINs had their fuel type labeled correctly. All remaining fuel types were classified with at least 87.7% accuracy. The lowest individual classification accuracy was achieved with engine power, but even this value of 0.859 should be considered impressive, given that this information is not even encoded in “Seat“ and “Cupra“ VINs.

Table 2. Accuracy metrics for separate technical parameters using the DTC model.

Technical parameter	Classification accuracy	$F1_{macro}$
Make	0.997	0.985
Model name	0.974	0.951
Body type	0.974	0.986
Model generation code	0.994	0.992
Fuel type	0.950	0.802
Engine displacement	0.899	0.866
Engine power	0.859	0.879

5. Conclusions

This research paper proposes several machine-learning methods capable of decoding a vehicle‘s technical parameters from its VIN. The optimal parameters for the models were selected using a small subsample and applying cross-validation to it. The data was split into training and testing subsets by stratifying it by the rarest labels, thus ensuring good data distribution in these subsets. After all models were fitted, the best results were achieved using the decision tree classifier which achieved 95% overall classification accuracy and correctly predicted the full technical specification of 82.2% VINs in the testing dataset. This model classified all individual technical parameters with at least 85.9% accuracy as well as $F1_{macro}$ no smaller than 0.802. Future research will consider the improvement of classification of rare values as well as finding better ways to separate “Seat“ and “Cupra“ VINs.

Acknowledgements. The author is grateful to Assoc. Prof. V. Skorniakov for providing feedback on the final version of the manuscript.

References

[1] W. Bachman, J. Granell, R. Guensler, and J. Leonard (1998). Research Needs for Determining Spatially Resolved Subfleet Characteristics. Transportation Research Record, 1625(1), pp. 139-146.

[2] P. Geurts, L. Wehenkel, and F. d'Alché-Buc (2006). Kernelizing the output of tree-based methods. In Proceedings of the 23rd international conference on Machine learning (ICML ,06). Association for Computing Machinery, New York, NY, USA, pp. 345–352.

[3] M. N. Dumont, R. Marée, L. Wehenkel, and P. Geurts (2009). Fast Multi-class Image Annotation with Random Subwindows and Multiple Output Randomized Trees. International Conference on Computer Vision Theory and Applications, pp. 196-203.

[4] L. Breiman (1998). Arcing Classifiers. The Annals of Statistics 26(3), pp. 801–849.

[5] P. Geurts, D. Ernst, and L. Wehenkel (2006). Extremely randomized trees. Mach Learn 63, pp. 3–42.

[6] J. L. Bentley (1975). Multidimensional binary search trees used for associative searching. Commun. ACM 18(9), pp. 509–517.

[7] J. H. Friedman, J. L. Bentley, and R. A. Finkel (1977). An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS), 3(3), pp. 209-226.

[8] S. M. Omohundro (1989). Five balltree construction algorithms, pp. 1-22.

[9] D. Krstinić, M. Braović, L. Šerić, and D. Božić-Štulić (2020). Multi-label classifier performance evaluation with confusion matrix. Computer Science & Information Technology, 1, pp. 1-14.

[10] K. Sechidis, G. Tsoumakas, and I. Vlahavas (2011). On the stratification of multi-label data. In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III 22, pp. 145-158.