The University of Texas at Austin

Department of Computer Science

Effectively Predicting Kicked Cars

Author:
Jashan Shah

Supervisor:
Professor Junfeng Jiao

Case Studies in Machine Learning

December 5, 2023

ABSTRACT

Automotive dealerships often have to make difficult decisions when purchasing a vehicle in hopes of reselling it and making a profit. There are many risks and factors to consider such as the vehicle's odometer, size, and the price point at different stages of the process. Using these features, we want to find an adequate way of predicting if the automotive dealership should buy the car from the auction. The automotive dealership should have been able to make a profit from selling the vehicle to make it a good buy. A kicked car, a car that has an unforeseen issue and is purchased, can have a negative impact on an automotive dealership. Consistent losses from kicked cars could lead to an automotive dealership closing. The intention of this study is to find a way to possibly improve a RandomForest model by using SHAP values for predicting whether a car is a kicked car.

Table of Contents

1. Introduction

This paper focuses on using SHAP values to find the least significant features and removing them in hopes of improving a model that predicts whether or not a car is a bad buy in terms of making a profit for an automotive dealership. We will be using a dataset from www.kaggle.com. It is known as the "Do not get Kicked!" a dataset that was provided by the Carvana competition.

In the pursuit of developing an effective predictive model, we will explore the effects of removing the least significant features from the dataset. The dataset comprises 32 features, including crucial information such as odometer readings, size, and price points at different stages of the purchasing process. The ultimate goal is to provide automotive dealerships with a tool to assess whether a particular vehicle is worth purchasing from an auction, with the intention of minimizing the likelihood of acquiring kicked cars and sustaining financial losses.

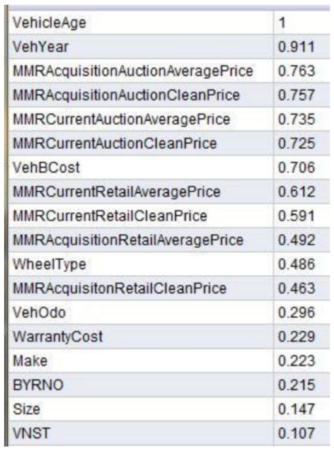
It is important that the automotive dealership tries to get as much information about the car as possible as they are often required only given a limited time to inspect the vehicles thoroughly which can sadly lead to purchasing cars that have hidden issues. Oftentimes auction cars can have incomplete or unclear maintenance and accident histories. This lack of information about the past of a vehicle can give powerful clues on whether or not a car would be a good buy or not.

A few other risks the automotive dealership faces are competitive bidding, hidden mechanical issues, market fluctuations, customer demand, and as-is sales. There are usually multiple automotive dealerships bidding on the same car which would make the cost of buying the auctioned car much higher. This competition is bad for the automotive dealership as it greatly reduces profits and makes it much more likely for a car purchase to be in a loss. If there is a hidden mechanical issue then it will take time and money to repair the issue. The money that the repair costs would come out of the profits and could lead to a loss if it is major enough. The amount of time it takes to repair the vehicle also has a huge impact. As the time it takes to repair the vehicle increases, the more likely that there will be market fluctuations of the value of used cars. Cars are known as depreciating assets, meaning that they lose value over time. A few months could result in thousands of dollars in lost profit. Consumer demand could also change during this time, a new trend might show up and the demand for certain types of vehicles can change due to this. Lastly, many of these auction sales are ‘as-is’. This means that the buyer is responsible for all the risks associated with the vehicle. Without any kind of warranty, if the car ends up not working correctly in any way right after the purchase, the automotive dealership would have to deal with it by themselves.

With all these risks associated with buying an auctioned car, the automotive dealerships are overdue with a better way of deciding whether or not a car is worth buying or not from an auction. My goal is a simple one, help automotive dealerships make better decisions when it comes to buying cars from auctions using machine learning. I will observe different machine learning methods to see which would be best in this scenario. However, in order to effectively utilize these advanced machine learning algorithms, we must first ensure we understand the data and are able to identify the most valuable parts of it.

1.2 Research Background

To select the most relevant and important features for the model's prediction, “Information Gain” and “Chi-square” can be used to see the relevance of the dataset features. The results using “Information Gain” are very similar. It is also important to note that the features with MMR in the beginning are closely related and have predictive power. Below are the results of using Chi-square to identify relevant and important features (Domejean n.d.).



VehicleAge	1
VehYear	0.911
MMRAcquisitionAuctionAveragePrice	0.763
MMRAcquisitionAuctionCleanPrice	0.757
MMRCurrentAuctionAveragePrice	0.735
MMRCurrentAuctionCleanPrice	0.725
VehBCost	0.706
MMRCurrentRetailAveragePrice	0.612
MMRCurrentRetailCleanPrice	0.591
MMRAcquisitionRetailAveragePrice	0.492
WheelType	0.486
MMRAcquisitonRetailCleanPrice	0.463
VehOdo	0.296
WarrantyCost	0.229
Make	0.223
BYRNO	0.215
Size	0.147
VNST	0.107

2. <u>Research and Methods</u>

2.1 Why Use SHAP

While Chi-Square and Information Gain are powerful in their own way, I decided to use SHAP as it has its own set of benefits. SHAP has the ability to individualize feature importance for each prediction. This lets us see the exact impact of each and every feature on each specific prediction. There are also ways to see more generalized trends since we have the detailed data as well. SHAP values are based on Shapley values from cooperative game theory, this ensures that there is a fair allocation of importance to each feature. If features are interrelated then their contributions are distributed in a way that respects their interactions. SHAP values are also model-agnostic, this means that they are able to be used on any machine learning model. Chi-Square and Information Gain are generally used more for decision trees and other tree-based models for feature selection. If we would like to explore the effects of features on different models then we should utilize SHAP as Chi-Square and Information Gain may not be directly applicable to other types of models.

2.2 Overview and Methods of Research

I decided to dive deeper into the RandomForest technique and performed a GridSearch to find the optimal hyperparameter values. However, before I did that I had to prepare the data in a way that would show the results of using SHAP to drop features that were least important and test if that would improve the machine learning model. For more information on how the data was prepared please refer to section 3.3 Data manipulation and Preparation. After the preprocessing and preparation of the data I decided to utilize the GridSearchCV library to get the best version of the model in a reasonable time. This means we would be systematically searching through a set of hyperparameter combinations to find the combination that best maximizes the performance of the model. A GridSearch will be ran twice, once before the least important features are dropped and once after the least important features are dropped.

We will perform GridSearch both times in order to make sure we have the models working at their best both times. This avoids the situation where features were removed but the model uses hyperparameters that were either much better or much worsley optimized, giving us false information on how the removal of the features affected the models performance.

Once we have the hyperparameters for our model that best fits our situation, we will use a set of evaluation metrics such as accuracy, mean fit time, std fit time, mean score time, std score time, etc. These evaluation metrics will be used later to compare with our new metrics to see if there were any improvements to the model. Our new metrics that will be formed by using SHAP values to and will be used to analyze the importance of each of the features for each model. We will then eliminate the least valuable features from the dataset and run another GridSearch on the modified dataset. This would give new optimal hyperparameter values along with the new performance metrics.

We would then compare each model's performance metrics before and after the least important features were eliminated. Using the performance metrics, we will decide if removing the least important features from the dataset is helpful in creating a better machine learning model for predicting whether or not an auctioned car is a good or bad buy.

3. <u>Materials and Data Sources</u>

3.1 Materials Used

In this study, I utilized the 'Don't Get Kicked!' datasets to analyze and draw insights into if it was possible to improve predicting whether a car is a good or bad purchase. The datasets were retrieved from www.Kaggle.com. Data processing and analysis were conducted using Jupyter Notebook, a widely used interactive computing environment.

I utilized several online sources during the research process to effectively determine that I was going in the right direction. I wanted to make an informed decision on what tools and models I should utilize in my research to ensure my work would provide as much value as possible.

3.2 Data Sources

The datasets utilized in this paper are from the Kaggle Competition by Carvana. The data contains missing values and 32 Independent variables (C3-C34). The dependent variable is 'IsBadBuy' which is a binary variable (C2). The dataset was split up to 60% training and 40% testing. Note the testing data only has 33 columns as it is omitting the 'IsBadBuy' column due to the nature of the competition. More information about the data and how it was prepared is in the next section below.

3.3 Data manipulation and Preparation

In this section we will discuss how the data was pre processed and prepared for the use of the machine learning model. Below is the training data info. It was retrieved by first loading the csv using a method from the Pandas library known as read_csv(). Then I simply just printed the info.

DataFrame Information:

- Shape: 72983 rows x 34 columns
- Memory Usage: 18.9+ MB
- Data Types:

  Float64: 10 columns | Int64: 9 columns | Object: 15 columns

Column Details:

#	Column	Non-Null Count	Dtype
0	RefId	72983	int64
1	IsBadBuy	72983	int64
2	PurchDate	72983	object
3	Auction	72983	object
4	VehYear	72983	int64
5	VehicleAge	72983	int64
6	Make	72983	object
7	Model	72983	object
8	Trim	70623	object
9	SubModel	72975	object
10	Color	72975	object
11	Transmission	72974	object
12	WheelTypeID	69814	float64
13	WheelType	69809	object

14	VehOdo	72983	int64
15	Nationality	72978	object
16	Size	72978	object
17	TopThreeAmericanName	72978	object
18	MMRAcquisitionAuctionAveragePrice	72965	float64
19	MMRAcquisitionAuctionCleanPrice	72965	float64
20	MMRAcquisitionRetailAveragePrice	72965	float64
21	MMRAcquisitonRetailCleanPrice	72965	float64
22	MMRCurrentAuctionAveragePrice	72668	float64
23	MMRCurrentAuctionCleanPrice	72668	float64
24	MMRCurrentRetailAveragePrice	72668	float64
25	MMRCurrentRetailCleanPrice	72668	float64
26	PRIMEUNIT	3419	object
27	AUCGUART	3419	object
28	BYRNO	72983	int64
29	VNZIP1	72983	int64
30	VNST	72983	object
31	VehBCost	72983	float64
32	IsOnlineSale	72983	int64
33	WarrantyCost	72983	int64

Note that many of the variables have null values and some more than others. PRIMEUNIT and AUCGUART stand out from the rest since they only have a mere 3419 non-null values. There are 72,983 entities, this means that only 4.68% of the entities have non-null values for these features. Considering how little information we have relative to the amount of data we have, I decided it was important to deal with all these null values. I decided to use transformers that would deal with all the null values in the dataset. I used SimpleImputer for numerical data and categorical data, however, I used different strategies for both. For numerical I went with mean and for categorical I used the category that showed up most frequently. I also decided it would be best to use StandardScaler for the numeric features. What this does is it removes the mean which will standardize all the numeric features. It is important to note that this is done to each column independently.

3.4 Utilizing Encoders

3.41 Utilizing OneHotEncoder

For categorical features, I also used OneHotEncoder. I decided to use this since it would make the features into a format that machine learning algorithms can use to improve predictions. An issue with OneHotEncoder is that it makes more columns. Take for example if you had a categorical variable “Color”. Let's say there are three different categories: red, blue, and green. The one-hot encoding might make that one column into three columns. One for red, one for blue and one for green. This would then make all the values 0 except for the ones that have that specific color. For example, if a color is present then it would instead of having a 0, it would have a 1. Each instance would have only one 1 and the rest would be 0s. This method might seem harmless but it can become a huge amount of features if there are a ton of different categories. With the huge increase in column amount, this could increase the run time of many different methods greatly. This could also make it difficult to use SHAP to see the importance of different features. On the other hand, it could also provide more in depth analysis. If there are too many different categories then we can start wondering what value that feature gives us. Take for example an extreme case, we have a categorical feature and all instances have a different category. The machine learning model would be unable to gain any valuable data from that feature since it would not really help in making predictions. Using this example, we can see why OneHotEncoder can be a useful tool in order to remove unimportant features from our dataset. To show exactly how many more features are created due to OneHotEncoder, consider how many it created in this experiment. The initial datasets training data consisted of 34 features. After the OneHotEncoder it became around 2200 features. That is nearly 64 times the amount of features. Not only does this take a lot of storage but it also takes a massive amount of time to utilize. For example the SHAP value calculations took several hours to compute since there were so many features with only around 70,000 samples. If there had only been 34 features then the SHAP value calculations would only take a few minutes to complete. While one hot encoding is powerful, for some use cases it is not worth utilizing. I believe in the future when our CPUs, RAM, and storage are better, it would be worth taking a look at one hot encoding for this use case again. In the next section we will talk about another type of encoder that I tried out that seemed to bypass the issues faced by one hot encoding.

3.42 Utilizing LabelEncoder

Since OneHotEncoder can increase runtime massively, it is important to know about other encoders that can get the job done. One such encoder is LabelEncoder. I tried the same experiment utilizing LabelEncoder. Like OneHotEncoder, LabelEncoder is used for encoding categorical labels with integer values. It is usually used when the target variable is categorical or when there are categorical features.

When using the fit method, LabelEncoder learns the unique classes or categories that are in the categorical data. It initially gives an integer label to each of the different categories or classes based on the order of appearance. So basically the first unique class would be labeled as 0, the second unique class would be labeled 1, and so on. Once this is done, the ‘classes_’ attribute would contain all the unique classes and the internal mapping would be a dict. For example, going back to our colors example, it could look something like this: {‘red’ : 0, ‘blue’ : 1, ‘green’ : 2, ‘yellow’ : 3}.

After the fitting is completed, the transform method can be used to convert the original labels into the encoded integer labels. ‘inverse_transform’ would do the same thing but the opposite. It would make encoded labels back to the original labels.

While LabelEncoder may be simple and straightforward to use, it can be an important process as many machine learning algorithms work with numerical data. I believe that the fact that it does not create a massive amount of features, outweighs the cons it may have. Of course it does depend on what exactly you need the model to do. One-hot encoding can be a safer option but it can come with tradeoffs.

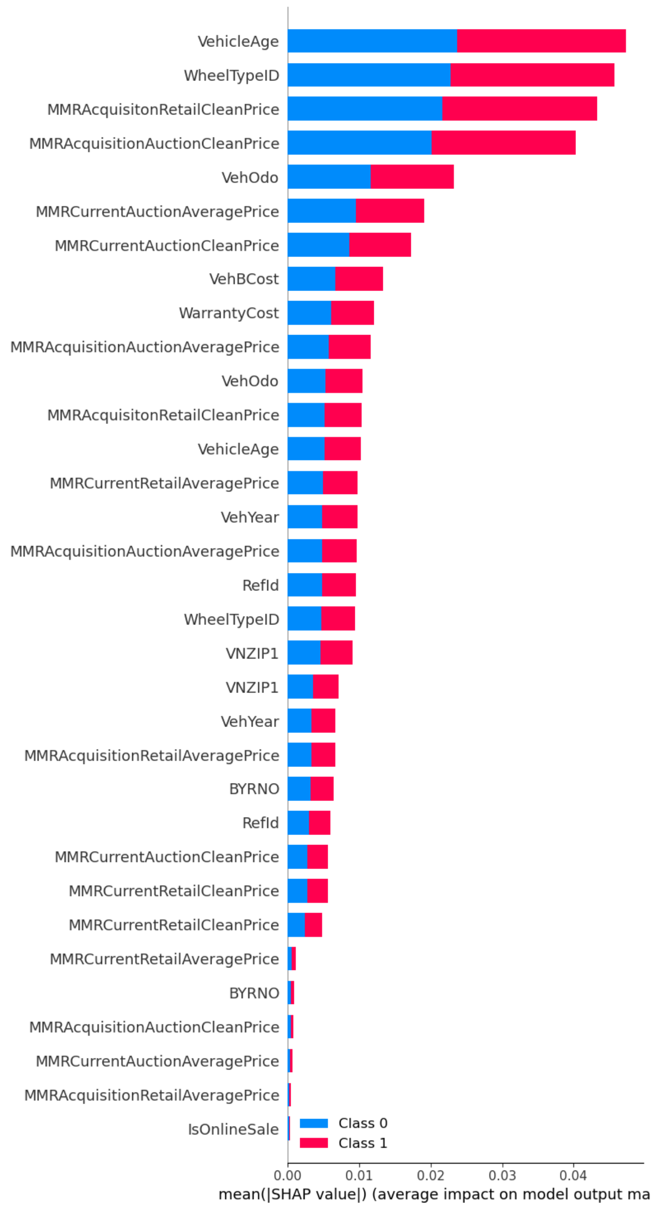
4. <u>Results</u>

The table below shows the results of the GridSearch for the RandomForest algorithm before any of the least important features were removed.

Rank test score	Max depth	N estimators	Mean test score	Mean fit time
15	None	3	0.618288	2.36432
14	None	5	0.628153	3.97783
9	None	50	0.632551	43.49304
5	None	100	0.633236	88.44614
4	None	200	0.633538	167.6955
1	10	3	0.806528	1.023928
2	10	5	0.716107	1.629653
11	10	50	0.63188	16.05501
7	10	100	0.632647	34.5443
8	10	200	0.632606	72.40083
12	20	3	0.629044	2.047769
13	20	5	0.628592	3.349103
10	20	50	0.63251	37.15353
6	20	100	0.633154	83.84591
3	20	200	0.633552	170.5427

The best hyperparameters: {'classifier__max_depth': 10, 'classifier__n_estimators': 3} for mean test score of accuracy. It is interesting to see that the amount of trees created of 3 along with a max depth of 10 is the best for accuracy. It also seems like the mean fit time gets larger as the n_estimators increase. The best hyperparameters model had a mean accuracy of 90.6528%. The std test score was 0.110265 which is really low. This means that performance is relatively stable.

Using the shap library, we were able to calculate the SHAP values. Below is a bar graph showing the values. Class 0 is blue which is a good buy and Class 1, a bad buy, is in red.



The bar graph above suggests that VehicleAge along with WheelTypeID and MMRAcquistionRetailCleanPrice are the top 3 most important features used by the RandomForest model. The least important features for predictions are 'MMRCurrentRetailAveragePrice', 'BYRNO', 'MMRAcquisitionAuctionCleanPrice', 'MMRCurrentAuctionAveragePrice', 'MMRAcquisitionRetailAveragePrice', and 'IsOnlineSale'. These variables seem to be considerably less important so I decided to drop all six of these features. I ran another GridSearch with the modified dataset and got the best parameters as: {'classifier__max_depth': None, 'classifier__n_estimators': 5}. Below shows the results of the GridSearch for the RandomForest algorithm after the least important features were removed.

Rank test score	Max depth	N estimators	Mean test score	Mean fit time
15	None	3	0.620837	2.448334
1	None	5	0.670712	3.965811
12	None	50	0.632428	39.30481
9	None	100	0.632839	79.20016
5	None	200	0.633264	195.956
2	10	3	0.664683	1.579255
4	10	5	0.633757	2.657654
14	10	50	0.632291	26.20842
10	10	100	0.632784	51.91853
7	10	200	0.633017	104.5575
3	20	3	0.646912	3.026155

8	20	5	0.632949	4.965341
11	20	50	0.63262	49.21936
13	20	100	0.632318	98.25469
6	20	200	0.633058	197.6697

The best parameters model had a mean test score accuracy of 67.0712% and a mean fit time of 3.965811. It is important to note that the best mean test score accuracy was considerably lower than before the removal of the least important features. However, there are certain models where the mean test score accuracy increased slightly. In fact there were 6 models where the mean test score was greater after the removal of the least important features. For example, before the removal when there were 5 n_estimators and no max depth the mean test score was 62.8153%. After the removal when there were 5 n_estimators and no max depth the mean test score was 67.0712%. This shows that there is potential to improve a RandomForest model using this method of removing the least significant features from the dataset.

5. **Conclusions**

My study sought to help automotive dealerships decide whether they should buy an auctioned car. I proposed a method to make RandomForest models better at predicting whether a car is a good or bad buy for an automotive dealership. My method consists of first finding the best parameters by utilizing GridSearchCV for the RandomForest model then getting the SHAP values and dropping the least significant features. Once we get the modified data, we do another GridSearchCV with the modified data and compare the evaluation metrics of the two best parameter models. I also explained the pros and cons of using two different encoders, OneHotEncoder and LabelEncoder. The end results show that it is possible for a model to improve using this method.

Lastly, I express my heartfelt gratitude to Professor Jiao and the Teaching Assistants for their guidance and support throughout this semester. This research project has been a significant learning experience for me. It has shaped my understanding of the subject matter and honed my skills as an engineer and developer. I appreciate Professor Jiao's leadership and the dedicated assistance provided by the Teaching Assistants, all of whom have contributed to the success of this academic endeavor.

References

1. Ho, R., Romano, J., & Wu, E. (2012). *Predicting "kicked cars" in an online automotive marketplace*. CS229 Project Report. Stanford University. HoRomanoWu-KickedCarPrediction.pdf (stanford.edu)
2. Whiting, M. E., Badinelli, R. D., & Rood, R. P. (2014). *Data Science with Kaggle's Competition: Don't Get Kicked.* Retrieved from ResearchGate
3. Figueroa-Domejean, O. (n.d.). *Data Science with Kaggle's Competition: Don't Get Kicked* [PDF file]. Retrieved from https://www.researchgate.net/profile/Oswaldo-Figueroa-Domejean/publication/262523736_Data_Science_with_Kaggles_Competition_Dont_Get_kicked/links/02e7e537e4b8e6c97b000000/Data-Science-with-Kaggle-s-Competition-Don-t-Get-kicked.pdf
4. Cutler, A., Cutler, D. R., & Stevens, J. R. (2011, January). Random Forests. *ResearchGate*. https://www.researchgate.net/publication/236952762_Random_Forests
5. Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324
6. Biau, G., & Scornet, E. (2012). A random forest guided tour. *Journal of Machine Learning Research, 13*, 369-450. https://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf
7. Louppe, G. (2014). Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*. Retrieved from https://arxiv.org/pdf/1407.7502.pdf
8. Van den Broeck, G., Lykov, A., Schleich, M., & Suciu, D. (2022). On the tractability of SHAP explanations. *Journal of Artificial Intelligence Research, 74*, 851-886. Retrieved from https://www.jair.org/index.php/jair/article/view/13283/26815
9. Liu, Y., Liu, Z., Luo, X., & Zhao, H. (2022). Diagnosis of Parkinson's disease based on SHAP value feature selection. *Biocybernetics and Biomedical Engineering, 42*(3), 856-869. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S0208521622000638
10. Jiang, D., Lin, W., & Raghavan, N. (2020). A novel framework for semiconductor manufacturing final test yield classification using machine learning techniques. *IEEE Access, 8*, 197885-197895. https://ieeexplore.ieee.org/abstract/document/9244159

11. Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*. Retrieved from https://arxiv.org/pdf/1704.02685.pdf
12. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. Retrieved from https://arxiv.org/pdf/1602.04938.pdf
13. Lv, Z., Ding, H., Wang, L., & Zou, Q. (2021). A convolutional neural network using dinucleotide one-hot encoder for identifying DNA N6-methyladenine sites in the rice genome. *Neurocomputing, 422*, 214-221. https://www.sciencedirect.com/science/article/abs/pii/S0925231220315137
14. Yadav, M. S., & Kalpana, R. (2019, December). Data preprocessing for intrusion detection system using encoding and normalization approaches. In *2019 11th International Conference on Advanced Computing (ICoAC)* (pp. 265-269). IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/9087274
15. Mitic, N., Jankovic, A., & Vukicevic, M. (2016). Predicting risk of buying low-quality car with RapidMiner. Retrieved from https://symorg.fon.bg.ac.rs/proceedings/2016/papers/DATA%20SCIENCE%20AND%20BUSINESS%20INTELEGENCE.pdf#page=24
16. Edmunds. (n.d.). *Confessions of an Auto Auctioneer.* Retrieved from https://www.edmunds.com/car-buying/confessions-of-an-auto-auctioneer.html
17. Nik. (2022, August 8). *One-hot encoding in Scikit-learn with onehotencoder • datagy*. datagy. https://datagy.io/sklearn-one-hot-encode/
18. Kumar, V. (2022, August 8). *Categorical data encoding with Sklearn LabelEncoder and onehotencoder*. MLK - Machine Learning Knowledge. https://machinelearningknowledge.ai/categorical-data-encoding-with-sklearn-labelencoder-and-onehotencoder/
19. Kumar, V. (2022, August 8). *Categorical data encoding with Sklearn LabelEncoder and onehotencoder*. MLK - Machine Learning Knowledge. https://machinelearningknowledge.ai/categorical-data-encoding-with-sklearn-labelencoder-and-onehotencoder/

20. *One hot encoding in Scikit-Learn*. ritchieng.github.io. (2023, September 30). https://www.ritchieng.com/machinelearning-one-hot-encoding/
21. Clayton, M. (2022, December 23). *The best methods for one-hot encoding your data*. Medium. https://towardsdatascience.com/the-best-methods-for-one-hot-encoding-your-data-c29c78a153fd
22. GeeksforGeeks. (2023, April 18). *Label encoding in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/
23. Awan, A. A. (2023, June 28). *An introduction to shap values and machine learning interpretability*. DataCamp. https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability?irclickid=we%3A3toUkqxyPT5CXA%3AzI%3AWnrUkFShrWxp3VCVY0&irgwc=1&im_rewards=1&utm_medium=affiliate&utm_source=impact&utm_campaign=000000_1-2003851_2-mix_3-all_4-na_5-na_6-na_7-mp_8-affl-ip_9-na_10-bau_11-Bing+Rebates+by+Microsoft&utm_content=BANNER&utm_term=EdgeBingFlow
24. Mazzanti, S. (2023, September 15). *Shap explained the way I wish someone explained it to me*. Medium. https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30
25. Dataman, C. K. (2023, January 30). *Explain your model with the shap values*. Medium. https://medium.com/dataman-in-ai/explain-your-model-with-the-shap-values-bc36aac4de3d
26. *Shap: How to interpret machine learning models with python*. Better Data Science. (n.d.). https://betterdatascience.com/shap/
27. Ram, S. (2020, October 18). *Mastering random forests: A comprehensive guide*. Medium. https://towardsdatascience.com/mastering-random-forests-a-comprehensive-guide-51307c129cb1
28. Meltzer, R., Rachel Meltzer Writer for The CareerFoundry BlogRachel is the founder of MeltzerSeltzer, Rachel Meltzer Writer for The CareerFoundry Blog, Meltzer, R., Blog, W. for T. C., & Rachel is the founder of MeltzerSeltzer. (2023, August 31). *What is*

Random Forest? [beginner's guide + examples]. CareerFoundry. https://careerfoundry.com/en/blog/data-analytics/what-is-random-forest/

29. Koehrsen, W. (2018, January 17). *Random Forest in python*. Medium. https://towardsdatascience.com/random-forest-in-python-24d0893d51c0
30. Koehrsen, W. (2018a, January 10). *Hyperparameter tuning the random forest in python*. Medium. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74