Classifying car price ranges with neural networks

Sivert Marius Skarning

April 2020

Contents

1 introduction

This project will try to find data pre-processing methods and a neural network that best predicts the buying price of a car, based on the car evaluation dataset. It will also compare performance and anccuracy between decision trees and neural networks on this dataset.

1.1 related work

There are numerous articles that have studied the performance of different modeling techniques with respect to the car evaluation dataset. The article by Sameer Singh[4] discusses the performance of varying training set sizes for different classification methods for the car evaluation sets. Sameer used artificial neural networks, K-nearest neighbour, decision trees and support vector machines in order to classify the acceptability of each car.

An article[3] also explored the performance of data mining classification methods. Here the authors also focus on the pre-processing of the data. They discuss concepts like data-cleaning, data-transformation and splitting of the data-set.

2 method

2.1 data quality

In order to ensure data quality it is necessary to assess the dataset. In this report we will clean the dataset by removing duplicates and fill missing values. We do this to give the neural network algorithms quality data to analyze[1].

2.1.1 missing values

After running an r script that counts missing values, we foundt out that there are no missing values. This will save us from doing interpolation or fill with mean method to fill missing values.

2.1.2 duplicate values

Duplicate values might give the modeling algorithm an idea that the date counts more than other data. This migh contribute to over-fitting. After running a script that counted duplicates in r, we found out that the dataset did not have any duplicate data.
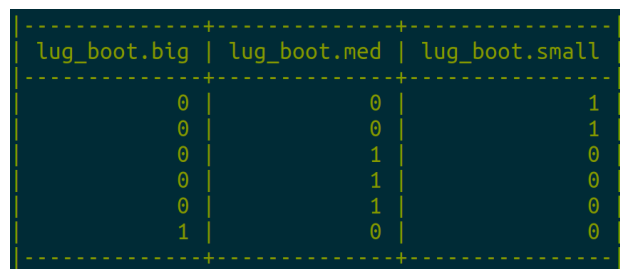
2.2 encoding

There are three main encodings when working with classification of categorical data[2].

- Integer encoding

- One Hot encoding

- Learned embedding encoding

In this project i will explore the performance of One Hot encoding on the car evaluation data set. This encoding method is used when the machine learning algorithm might not be able to understand the relationship between the data. Integer encoding on this dataset gave poor perfomance.

In figure 1 you can see how one hot encoding encodes the lug_boot size category into numerical values by splitting the categories into seperate columns of data.

```
--------------+--------------+----------------
 lug_boot.big | lug_boot.med | lug_boot.small
--------------+--------------+----------------
            0 |            0 |               1
            0 |            0 |               1
            0 |            1 |               0
            0 |            1 |               0
            0 |            1 |               0
            1 |            0 |               0
--------------+--------------+----------------
```

Figure 1: Excerpt of One Hot Encoding on car evaluation dataset

2.3 neural networks in neuralnet

2.3.1 opening remarks

In this project i decided to use neuralnet in r for the neural network prediction. This is a well know package and it has good resources.

2.3.2 training data

After cleaning the dataset and applying one-hot encoding I fit the model to the training data. The calculation of the accuracy of the model was done with the with the result matrix and the original data. This showed an error rate of 64% wich is worse than the error rate of the decition tree. I decided to split the dataset into to different sets. One training set and one testing
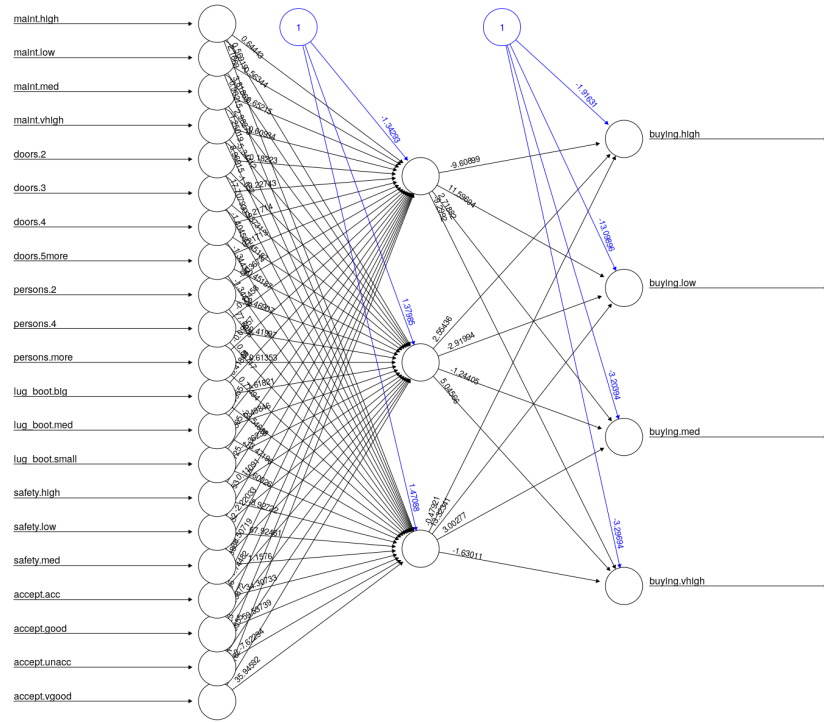
Figure 2: Artificial Neural Network

set. This prediction was done on the training set. This is not accurate of how the model will perform in real life since the model can be highly tuned for this set (over-fitted).

2.3.3 test data

The prediction on the test data will display the accuracy of the model more realistic. A huge gap betweeen accuracy in test data-set and training data-set will be signs of over-fitting.

When running the ANN on the validation data we got an almost identical accuracy. On the training set we got 35.57% accuracy, when running the model on the validation data we got a 36.67%. The fact that the two results are so simulare to eachother is a good indicator that the model is not over-fitted.

2.3.4 K-fold cross-validation

Even tough the model is not over-fitting i decided to try k-fold cross-validation in order to find the accuracy of the model. Cross-validation is a method for

Figure 3: ANN results

testing the model that used all of the data for testing and all of the data for validation. The way it works is that it splits the dataset into k different parts. It then uses k - 1 parts for training and the last part for validation. Then it repeats it k times in such a way that all the k-parts has been used in both testing and validation. Then it takes the average accuracy of these models.

In my project I used 10 fold cross validation. Cross validation gave a 31.6% accuracy wich is worse than the split testing method. This could be a sign that the model is overfitted.

2.3.5 learning curve

A learning curve is a great tool for debugging the model. The learning curve gives you vital information on if the model is learning and if it is the fit is good.
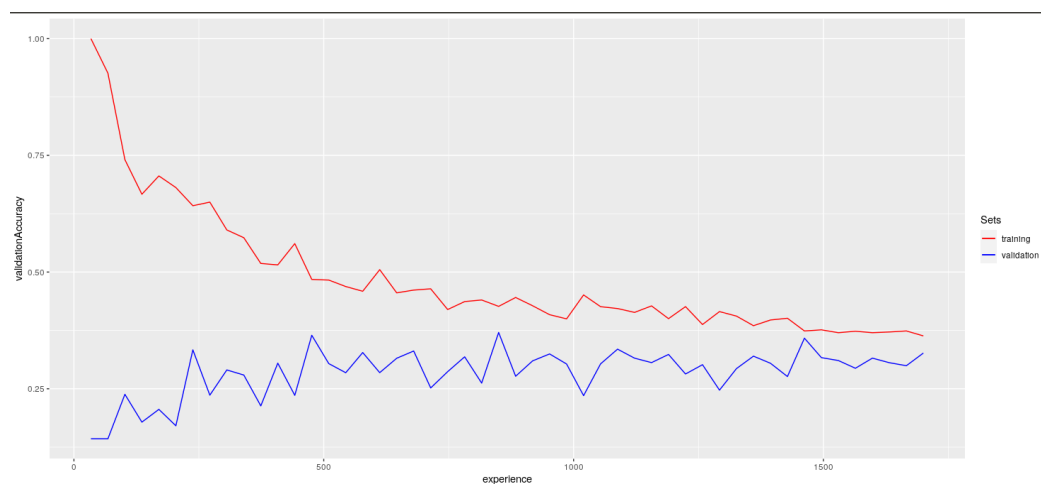

Figure 4: Learning Curve of neural network

As you can see in figure 4 the model is learning. However the training process could benefit from more training examples in order to close the gap between the training accuracy and the validation accuracy. In figure 4 the x-axis shows number of training examples and y-axis shows the accuracy of the model.

2.4 neuralnetworks in h2o

I have been using the neuralnet package for my predictions up to this point. Neuralnet is an easy to use package. It is great for getting into machine learning and for creating basic learning models with. The problem with neuralnet is its lack of parameter tuning options. Customizing the layers and activation functions is limited.

From this point and on I will use a more advanced machine learning package called h2o. This has more options for tuning and more references.

2.4.1 tuning

Hyperparameter tuning is trying to find the most optimal parameters for getting the most accurate neural network. There are many ways at doing hyperparameter tuning. The easiest way is actually trial and error. The problem is that this takes a long time, and it is easy to get lost with no structure.

grid search In grid search we will define what hyperparameters we would like to tune. Then we would construct ranges that the hyperparameters have to be in. Then grid search will try alle the combinations of these parameters and find the best value.

For grid search I created a grid with three values.

- Number of hidden layers and neurons in each layer

- Input_dropout_ratio

- Learning rate

In figure 5 you can se the result of the grid search sorted on what goves the smallest logloss on the set. The figure shows that a network with 2 hidden layers with 10 neurons in each, a input dropout ration of 0.5 and a learning rate of 0.01 gives the best result. When testing the model I get a error rate of 61.3 percent. This is a good improvement from the 62.4 percent i get when i tune the model my self.

```
Hyper-Parameter Search Summary: ordered by increasing logloss
         hidden input_dropout_ratio rate  model_ids           logloss
1       [10, 10]                0.5 0.01   1_model_5 1.2236142156369745
2      [5, 5, 5]                0.5 0.05  1_model_14 1.2470897631603002
3       [10, 10]                0.5 0.05  1_model_13 1.2611977142449862
4      [5, 5, 5]                0.0 0.01   1_model_2 1.2761085921495066
5   [10, 10, 10]                0.5 0.01   1_model_7  1.28758283072634
6      [5, 5, 5]                0.5 0.01   1_model_6 1.3157903971925144
7   [10, 10, 10]                0.0 0.05  1_model_11  1.33090760015359
8      [5, 5, 5]                0.0 0.05  1_model_10 1.3383175829320841
9       [10, 10]                0.0 0.01   1_model_1 1.3508842918639907
10  [10, 10, 10]                0.0 0.01   1_model_3 1.3657055642249858
11      [10, 10]                0.0 0.05   1_model_9  1.381433558116787
12    [100, 100]                0.5 0.01   1_model_8 1.4000534524214154
13    [100, 100]                0.5 0.05  1_model_16 1.4073280537789483
14    [100, 100]                0.0 0.01   1_model_4 1.4809109741389215
15  [10, 10, 10]                0.5 0.05  1_model_15 1.7055866466486607
16    [100, 100]                0.0 0.05  1_model_12 1.7639304365187627
```

Figure 5: Results of grid search

2.4.2 Optimization

gradient decent

RMS Prop Optimization

Adam

References

[1] Data preprocessing concepts. https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825. Accessed: 2020-03-02.

[2] Why one-hot encode data in machine learning? https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. Accessed: 2020-03-02.

[3] Jamilu Awwalu, Anahita Ghazvini, and Azuraliza Abu Bakar. Performance comparison of data mining algorithms: A case study on car evaluation dataset. *Int. Jour. of Computer Trends and Technology (IJCTT)*, 13(2), 2014.

[4] Sameer Singh. Modeling performance of different classification methods: deviation from the power law. *Project Report, Department of Computer Science, Vanderbilt University, USA*, 2005.