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 data set. It will also compare performance and accuracy between decision trees and neural networks on this data set.

1.1 Literature review

There are numerous articles that have studied the performance of different modeling techniques with respect to the car evaluation data set. 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 neural networks, K-nearest neighbor, 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 data set. In this report we will clean the data set 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 found 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 might contribute to over-fitting. After running a script that counted duplicates in r, we found out that the data set did not have any duplicate data.

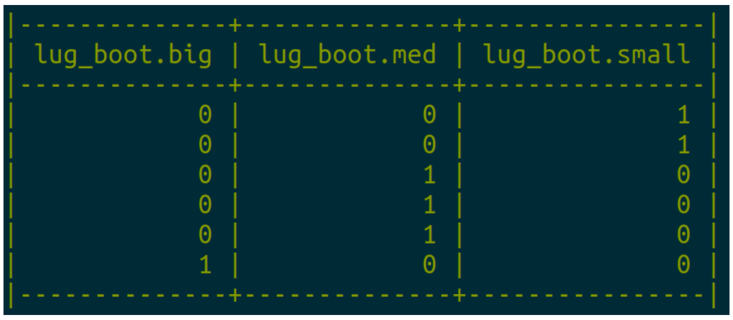
2.2 Encoding

There are three main encoding 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 data set gave poor performance.

In figure 1 you can see how one hot encoding encodes the lug_boot size category into numerical values by splitting the categories into separate 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 data set

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 data set 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% which is worse than the error rate of the decision tree. I decided

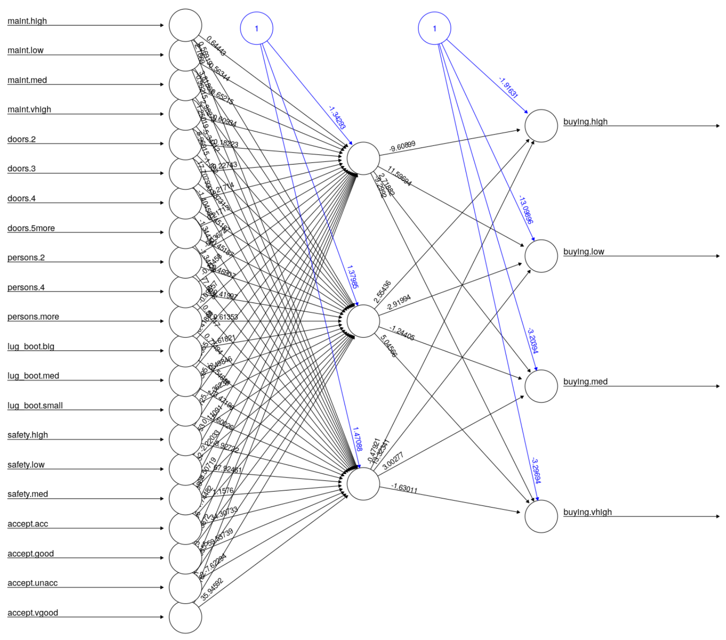
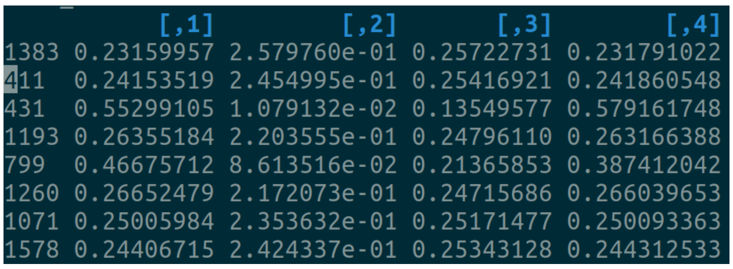


Figure 2: Neural Network



```
           [,1]         [,2]       [,3]         [,4]
1383 0.23159957 2.579760e-01 0.25722731 0.231791022
411  0.24153519 2.454995e-01 0.25416921 0.241860548
431  0.55299105 1.079132e-02 0.13549577 0.579161748
1193 0.26355184 2.203555e-01 0.24796110 0.263166388
799  0.46675712 8.613516e-02 0.21365853 0.387412042
1260 0.26652479 2.172073e-01 0.24715686 0.266039653
1071 0.25005984 2.353632e-01 0.25171477 0.250093363
1578 0.24406715 2.424337e-01 0.25343128 0.244312533
```

Figure 3: ANN results

to split the data set into to different sets. One training set and one testing 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 between 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 simular to each other 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 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 data set 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 which is worse than the split testing method. This could be a sign that the model is over fitted.

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.

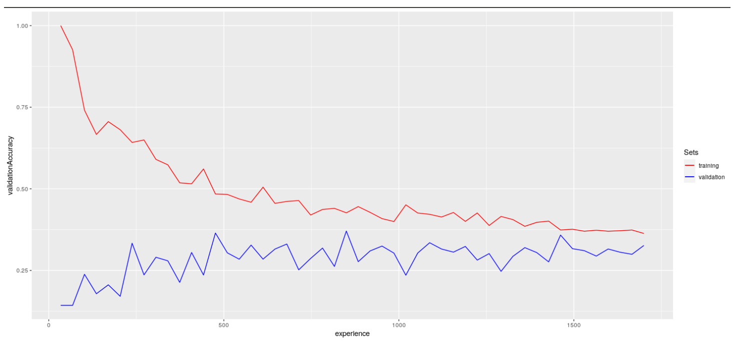


Figure 4: Learning Curve of neural network

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 Neural networks 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.

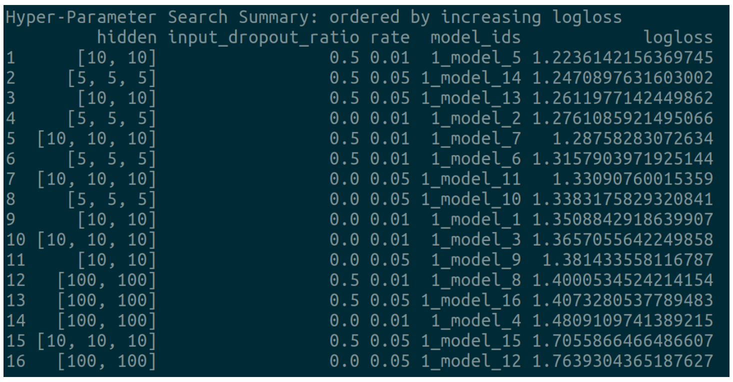
2.4.1 Tuning

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



```
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

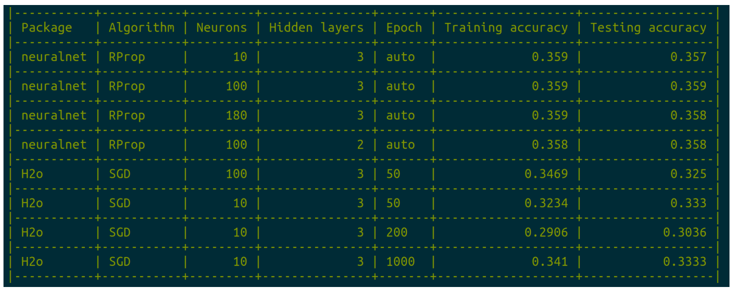
In figure 5 you can see the result of the grid search sorted on what gives the smallest logloss on the set. The figure shows that a network with 2 hidden layers with 10 neurons in each, a input dropout ration of 0.5 and a learning rate of 0.01 gives the best result. When testing the model I get a error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

2.4.2 Optimization

Optimization of the neural network is a way to get better predictions. The concept is about minimizing the chosen error function. Gradient decent is a method where you find the gradients of the loss function and takes steps where the gradient decrease. If the gradient is steep it takes long steps(high step value), if the gradient is less steep it takes a smaller step. In this way it will converge towards the most optimum values for the weights of the neural network. There are many different types of gradient decent algorithms.

In the CRAN neuralnet package an optimization algorithm called resilient backpropegation(Rprop) is used. This is a very fast algorithm, however it is complex to implement. The H2o packages uses stochastic gradient descent(SGD). This algorithm takes on single data set each epoch, and adjusts the weights. The example it chooses is random, and that is why we it is called stochastic[1].

There are not possible to switch optimization algorithms on H2O or neuralnet package. However I can compare how the two models performed. The comparison of the two algorithms will never be accurate because they are run on two different packages. When trying different parameters like the number of epochs, neurons, hidden layers etc, I got the results in figure.



```
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| Package   | Algorithm | Neurons | Hidden layers | Epoch | Training accuracy | Testing accuracy |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| neuralnet | RProp     |      10 |             3 | auto  |             0.359 |            0.357 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| neuralnet | RProp     |     100 |             3 | auto  |             0.359 |            0.359 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| neuralnet | RProp     |     180 |             3 | auto  |             0.359 |            0.358 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| neuralnet | RProp     |     100 |             2 | auto  |             0.358 |            0.358 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| H2o       | SGD       |     100 |             3 | 50    |            0.3469 |            0.325 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| H2o       | SGD       |      10 |             3 | 50    |            0.3234 |            0.333 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| H2o       | SGD       |      10 |             3 | 200   |            0.2906 |           0.3036 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
| H2o       | SGD       |      10 |             3 | 1000  |             0.341 |           0.3333 |
|-----------+-----------+---------+---------------+-------+-------------------+------------------|
```

Figure 6: Results of optimization

[1]https://towardsdatascience.com/demystifying-optimizations-for-machine-learning-c6c6405d3eea

2.4.3 Interpretation

In order to measure how good the model is at predicting output we need to analyze some rows of the data to analyze the output of the data. In the tables below we see the input row and the output the model gives us.

Input

Row	buying	maint	doors	persons	lugboot	safety	accept
4	med	vhigh	5	5	med	med	acc

Output

Row	predict	high	low	med	vhigh
4	med	0.015	0.286	0.698	0.000

2.4.4 Confidence interval

In order to put into perspective how accurate we think our model is, we can use confidence intervals. The problem about using an validation set is that it is just a limited number of rows. We can not test all data that exists.

The confidence interval tells us how sure we are that the acutal error rate is within the interval. In this paper I will find the confidence interval where we are sure that we are 95% sure that the acutal error rate is within the interval [2].

```
Z <- 1.6              # Z-value from table
N <- 173              # Number of rows in validation-set
ERROR <- 0.62         # The given models error-rate

confidence_interval <- function(z_value, n, error_rate) {
    # Calculates the confidence interval for the given error
    return (z_value * sqrt((error_rate * (1 - error_rate)) / n))
}


print(confidence_interval(Z, N, ERROR))
```

The resulting interval was 5.9%. This mean that we are 95% sure that the error rate is between 0.56 and 0.679.

[2]https:machinelearningmastery.comconfidence-intervals-for-machine-learning

3 Conclusion

When starting this project I hoped that I could create a viable model that could categorize what price-range a car would be in based on some predictors. After trying to predict this with both classification trees and neural networks, I would say that the data does not contain the variance to make this possible. More data about the cars would have to be gathered in order to make the prediction accurate.

Both classification trees and neural networks spawned similar results. The classification tree was a lot easier to set up than what the neural network was.

I have tried data preprocessing, feature extraction, encoding, learning curves to try to get better results from the model. Even tough some has improved the accuracy, the total result is not sufficient.

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.