Danmarks Tekniske Universitet

Car Price Prediction - Part 1

Project by
by

Bjarke Erichsen (s204104)
Frederik Sartov (s204118)
William Peytz (s204145)

Introduction to Machine Learning and Data Mining, 02450

Technical University of Denmark

October, 2021

Contents

1 Data description

The dataset [1] we have chosen to work with, is a dataset that contains data on cars and their related price. We might use the dataset to understand which attributes are more or less relevant for pricing. The dataset contains 26 features, including the target variable which is car price and the ID feature which does not carry relevant information to the problem at hand. Furthermore, for the regression part of the next report we want to predict the price of the given car, based on the features. For the classification part we will do the same, but with the simple change of discretizing the price feature into a couple categories of price for example cheap, middling, expensive and very expensive, and then training a model to predict which category a given car belongs to. Importantly, to simplify the classification and regression tasks we will use the understanding gained in the data preparation phase. This will include excluding irrelevant features and employ data preparation techniques to standardize and clean relevant data. Lastly, we will also use PCA by projecting the data into a subspace spanned by the most important principal components, thereby simplifying the problem.

Other analysts have previously worked on this dataset. One of the most popular use cases of the dataset was the ability to try and create a price prediction model. One previous project created by Agata Rutkowska [2], tried to create a model that could predict the price of various cars using multiple linear regression on the dataset. To analyze the data she first added a new column called car brand, to replace the car name category. Then she imported the first word from the car name category. She cleaned the car brand column for spelling errors, such as replacing "toyouta" with "toyota", so all cars from the same brand came in the same category, and afterwards deleted the car name category. She then plotted the mean car price by brand. Then she transforms the car brand column by doing one-hot-encoding. Afterwards she imported two different linear models, ElasticNetCV and LassoCV. She then tested both models in a 75/25 training/test split. From these tests she got the best score with the LassoCV model at 90.17% accuracy.

2 Data attributes

The attribute feature types are showcased in the following table, where D means discrete, B means binary and C means continuous.

Table 1: Attribute and attribute type

CarID	Symboling	CarName	Fueltype	Aspiration	Doornumber	Carbody
nominal/D	ordinal/D	nominal/D	nominal/B	nominal/B	nominal/B	nominal/D
Drivewheel	Enginelocation	Wheelbase	Carlength	Carwidth	Carheight	Curbweight
nominal/D	nominal/B	ratio/C	ratio/C	ratio/C	ratio/C	ratio/D
Enginetype	Cylindernumber	Enginesize	Fuelsystem	Boreratio	Stroke	
nominal/D	ordinal/D	ratio/D	nominal/D	ratio/C	ratio/C	
Compressionratio	Horsepower	Peakrpm	Citympg	Highwaympg	Price	
ratio/C	ratio/D	ratio/D	ratio/D	ratio/D	ratio/D	

When classifying the different features into different attribute types, some of the attributes are very straightforward to classify. However, other attributes do not fit so neatly into a single category thereby making their categorical relationship dependant on perspective and the objective of the ML task. One dichotomy that presented a lot of these ambiguities was the nominal vs ordinal split. The reason for this is that some attributes could be viewed as on a spectrum from less to more and be viewed as simply a set of essentially different categories. An example of this is the attribute "Doornumber", this could be viewed as an ordinal feature ordered from less doors to more doors. However, we chose rather to view it as nominal, as each number of doors seems best thought of as representing its own category, especially when taking into account that we want to predict price. An example of the very non-linear relationship between door number and price could be sports cars with two doors having a higher price than many four-door cars. We applied the same logic to attributes like "Aspiration" and "Enginetype" etc.

We also had to make decisions on whether certain attributes were discrete or continuous, where it matters how you interpret the attributes. An example would be "carweight" which is measured in kilogram and would normally be considered continuous, but in our dataset all the values are integers, we decided to interpret "carweight" as discrete. We did this with attributes "carweight", "enginesize", "citympg" and "highwaympg". An important note to consider is that due to machine learning methods we are going to use for this project it is not going to make much of a difference whether we classify these attributes as continuous or discrete. Specifically the discrete vs. continuous spectrum primarily matters when it comes to target variables.

In regards to the quality of the data there are no missing values in any of the 26 different attributes (see table 1), so there is no reason to exclude data points for this reason. However there are some corrupted data in the form of spelling errors in the car name attribute. This is not a big problem as the spelling mistakes are minor, so

we are still able to determine which car name was actually meant, and therefore we are able to correct all of the spelling mistakes.
The attribute 'car_ID' is spurious and not relevant in regards to the price of the vehicle and should therefore be removed. Another data issue could be outliers, but before doing further analysis it can be hard to tell if these exist. If there are outliers and they seem to be a mistake, it could be preferable to remove these data points from the dataset.

Table 2: Attribute summary statistics

	Symboling	Wheelbase	Carlength	Carwidth	Carheight	Curbweight	Enginesize	Boreratio
Mean	0.83	98.76	174.05	65.91	53.72	2555.57	126.91	3.33
STD	1.0	6.0	12	2.0	2.0	521.0	42.0	0.0
Median	1.0	97.0	173.2	65.5	54.1	2414.0	120	3.31
Max	3	120.9	208.1	72.3	59.8	4066	326	3.94
Min	-2	86.6	141.1	60.3	47.8	1488	61	2.54

Table 3: Attribute summary statistics

	Stroke	Compressionratio	Horsepower	Peakrpm	Citympg	Highwaympg	Price
Mean	3.26	10.14	104.12	5125.12	25.22	30.75	13276.71
STD	0.0	4.0	40.0	477.0	7.0	7.0	7989.0
Median	3.31	9.0	95.0	5200.0	24.0	30.0	10295.0
Max	3.94	23.0	288	6600	49	54	45400.0
Min	2.54	7.0	48	4150	13	16	5118.0

Table 4: Attribute proportion

Engine Location	Proportion (%)
Front	0.99
Rear	0.01
Drivewheel	
Foward	0.59
Rear	0.37
Quad	0.04
Carbody	
Sedans	0.47
Hatchback	0.34
Wagon	0.12
Hardtop	0.04
Convertible	0.03
Doors	
Four	0.56
Two	0.44
Aspiration	
Std	0.82
Turbo	0.18
Fueltype	
Gas	0.9
Diesel	0.1

Fuelsystem	Proportion (%)
mpfi	0.46
2bbl	0.32
idi	0.1
1bbl	0.05
spdi	0.04
4bbl	0.01
spifi	0.0
mfi	0.0
Cylindernumber	
Four	0.78
Six	0.6
Five	0.05
Eight	0.02
Two	0.02
Enginetype	
ohc	0.72
ohcf	0.07
ohcv	0.06
l	0.06
dohc	0.06
rotor	0.02
dohcv	0.0

We transform the CarName attribute into an attribute simply of car brands (see table 5), as this is the information we want from the attribute. Even though many car brands only show up once in the dataset, we still consider this information important because it is the only information we have about the brand value which is presumably a factor that contains a lot of information about the price. The following is a table of the number of data points for each car brand.

Table 5: Data elements by car brand

Toyota	Nissan	Mazda	Mitsubishi	Honda	Subaru	Volvo	Peugeot
32	18	17	13	13	12	11	11
Volkswagen	Dodge	Bmw	Buick	Audi	Plymouth	Saab	
12	9	8	8	7	7	6	
Porsche	Isuzu	Alfa-romero	Chevrolet	Jaguar	Renault	Mercury	
5	4	3	3	3	2	1	

Concerning which attributes (see table 4) we will be continue using and in later reports base our model on, we choose to exclude the following attributes. Firstly, we exclude car_ID because its spurious. We also exclude Engine Location because of a lack of balance in data point, ie. 99% of the engines are in front, thereby making any generalisation from this unfounded. Fuel-type is dropped, because of its unbalanced proportion.

3 Data visualization

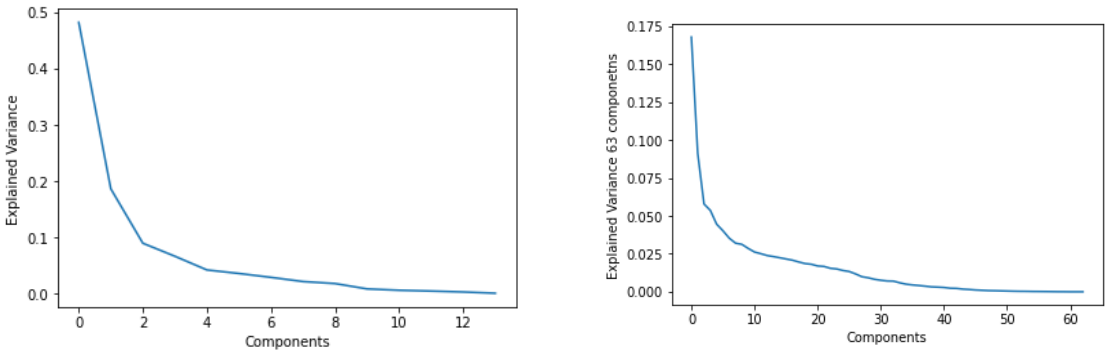
3.1 PCA analysis

Importantly, we have chosen to do PCA analysis on both the whole dataset and the dataset consisting only of quantitative features. These features are the ones described in table 6. The reason for this split is to be able to compare the two results. In short, to get the, informationally richer PCA analysis gotten from all the features, while still getting the safe more easily understandable PCA gotten from the subset of features.

dim 1-7	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize
dim 8-14	boratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg

Table 6: Features

Having standardized these features the PCA gives us 14 principal components each of which contain some proportion of the variance in the dataset. To get an overview, we plotted the sorted proportion of variance and the principal components corresponding to figure 1.a.



(a) Explained variance with 14 principal components (b) Explained variance with 63 principal components

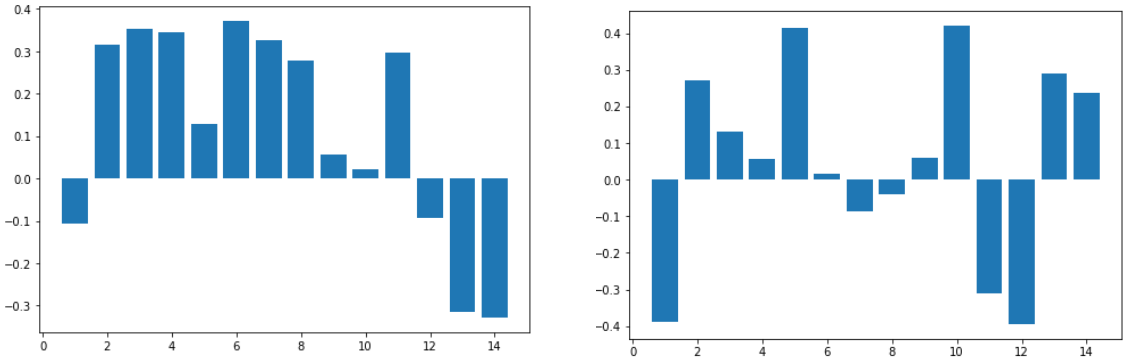
Figure 1: Explained variance: PCA with 14 and 63 principal components respectively.

As can be seen on the plots above, around 90% of the variance is captured by the first two principal components in the first plot and around 94% of the variance is captured in the first two principal components in the second plot. Each principal component of, corresponding to the first plot, is of course a 14 dimensional vector, which means we can visualize them directly in 14D space. Therefore we choose to plot these in the following way as seen on figure 2.

As the data we work with has more than a few categorical attributes, we would then like to know how much of this data is impactful. To see this we used principal component analysis and compared two versions of the same data. In one version we excluded the categorical data ending up at 14 attributes and a version with the categorical data, where we used one-hot-encoding, totalling up to 63 attributes.

3.2 Principal directions of features

Figure 2 shows the weight of each attribute on the first two principal components. This insight into the principal directions gives us a look into which attributes maximises the variance.

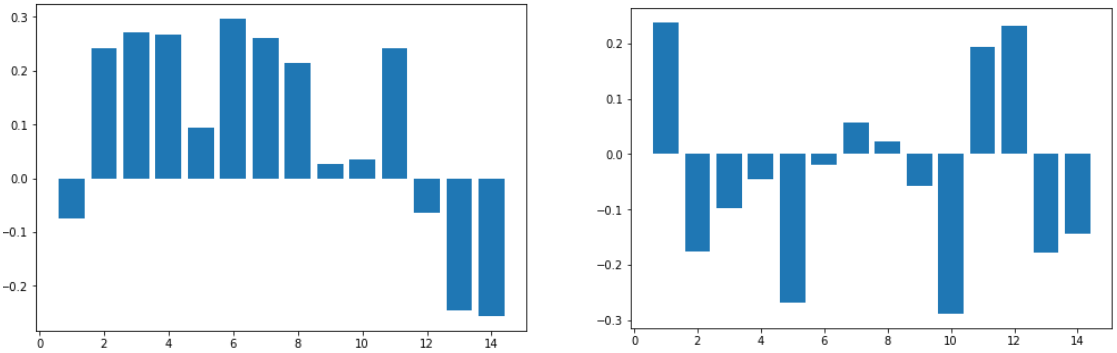


(a) Principal directions of features on the first principal component

(b) Principal directions of features on the second principal component

Figure 2: Principal directions of the first 14 principal components

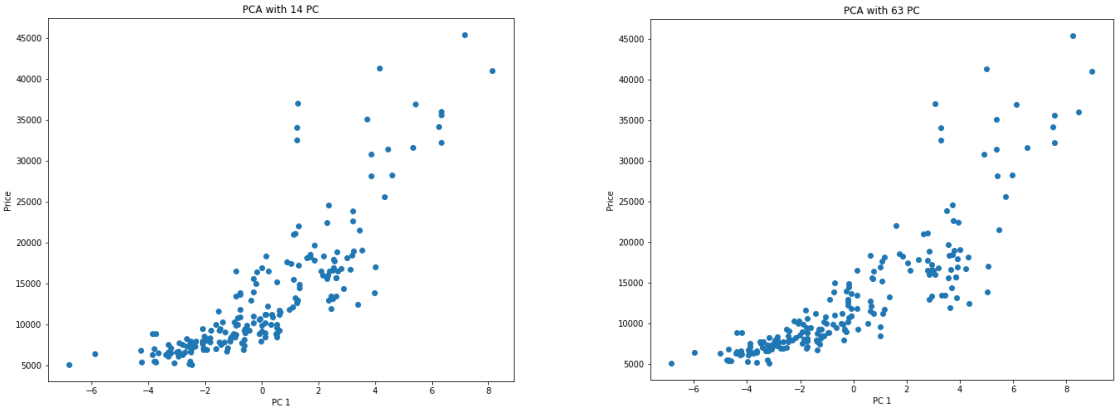
The first 14 principal directions of the attributes on the one-hot-encoded categorical data is shown on the figure 3. The vector representation of the principal directions are located in the appendix.



(a) Principal directions of features on the first principal component

(b) Principal directions of features on the second principal component

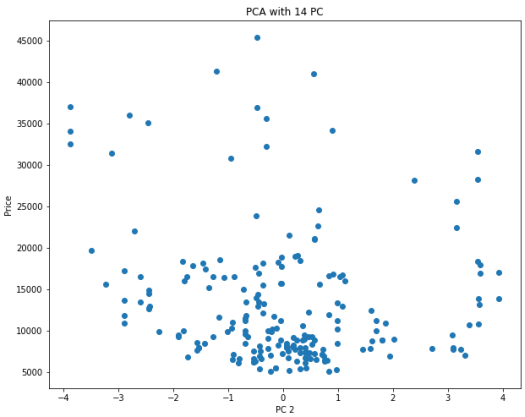
Figure 3: Principal directions of the first 14 principal components of the 63 attributes



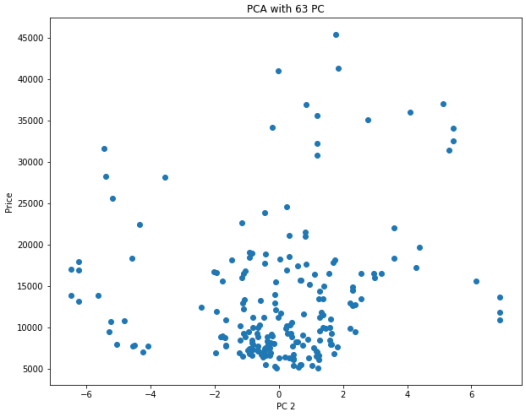
(a) Principal directions of 14 features on the first principal component with price on y-axis

(b) Principal directions of 63 features on the first principal with price on y-axis component

Figure 4: The first principal component against price



(a) Principal directions of 14 features on the second principal component with price on y-axis



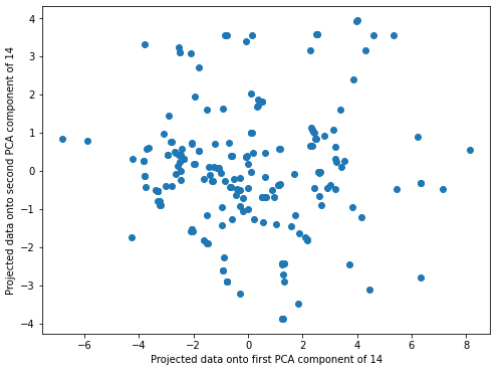
(b) Principal directions of 63 features on the second principal with price on y-axis component

Figure 5: The second principal component against price

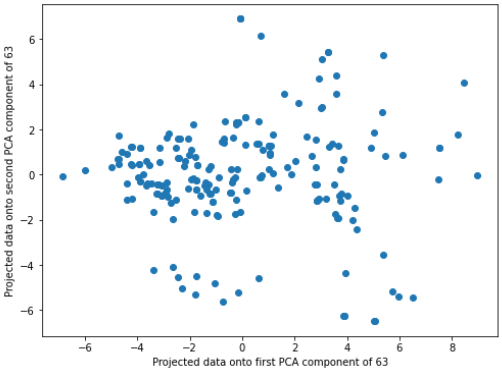
The introduction of categorical data points changes the principal directions only to a limited degree. The length of each principal component being 1, we see that most of this length, when using all attributes, comes from dimensions pertaining to the first 14 dimensions, ie. the quantitative dimensions. These dimensions pertain to respectively around 90% and 94% of the variance and the first two PC's have mutual dependence with price. With these facts we can in all likelihood say that the quantitative dimensions will be the primary factors which go into differentiating prices between cars. Although it must be said that there will be situations where this isn't true, for example if a dimension with very little variance still correlates heavily with price.

3.3 Projected data onto principal components

Figure 6 and 7 shows the data in both 2D and 3D space with the price included on the 3D plot.

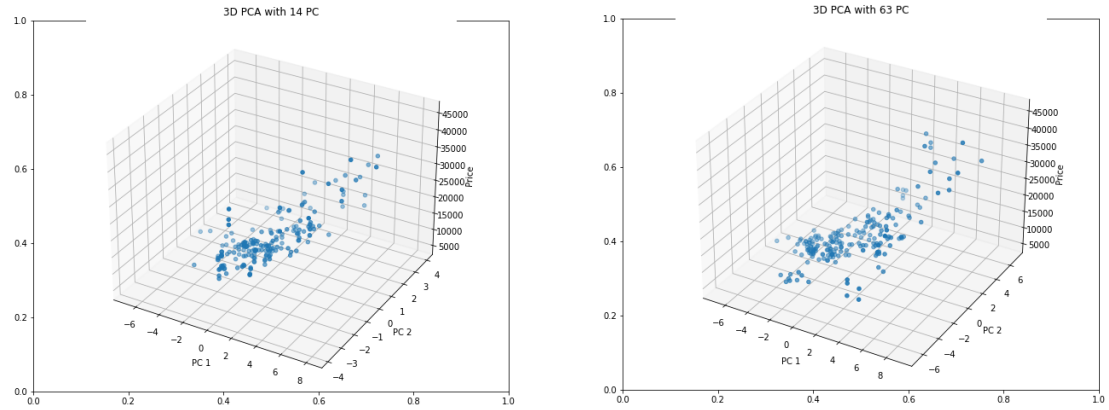


(a) Projected data with 14 principal components in 2D



(b) Projected data with 63 principal components in 2D

Figure 6: 2D PCA plot of both 14 and 63 principal components



(a) Projected data with 14 principal components in 3D

(b) Projected data with 63 principal components in 3D

Figure 7: 3D PCA plot of both 14 and 63 principal components with price attribute on the z-axis

When we take the projected data onto our different principle components it is important that the data is first standardized, before evaluating correlation. When looking at the results, it becomes clear that the principle components and the price are dependent on each other, as there is correlation between the them. This is true both for the 14, as well as the 63 principle components.

3.4 Feature exploration

To explore the features we are working with, we will first see which of our quantitative features are normally distributed. We decided to test this with qq-plots. Furthermore, to evaluate effectively we also plotted a histogram of each attribute, thereby checking the normal distribution from two statistical references.

Table 7: Normal distribution evaluation with qq-plot

Attribute	QQ-plot analysis
Symboling	Right skewed
Wheelbase	Heavy-tailed
Carlength	Normal distributed
Carwidth	Heavy-tailed
Carheight	Normal distributed
Curbweight	Bottom heavy-tailed
Cylindernumber	Heavy-tailed
Enginesize	Heavy-tailed
Boreratio	Right skewed
Stroke	Heavy-tailed
Compressionratio	Top heavy-tailed
Horsepower	Heavy-tailed
Peakrpm	Heavy-tailed
Citympg	Bottom-tailed-heavy
Highwaympg	Slight top heavy-tailed
Price	Heavy-tailed

From analyzing and validating the QQ-plots (See appendix) we concluded that only two of the attributes were normal distributed, these were: "carlength" and "carheight". The conclusion for the other attributes are found in table 7. The histogram analysis was found to be similar where "carlength" appear to be normal distributed and "carheight" and "carwidth" were found to be near normal distributed. From this we conclude that "carlength" is the only true normal distributed attribute. However, as carheight and carwidth are near normal distributed they are accepted as well.

3.5 Correlation between attributes

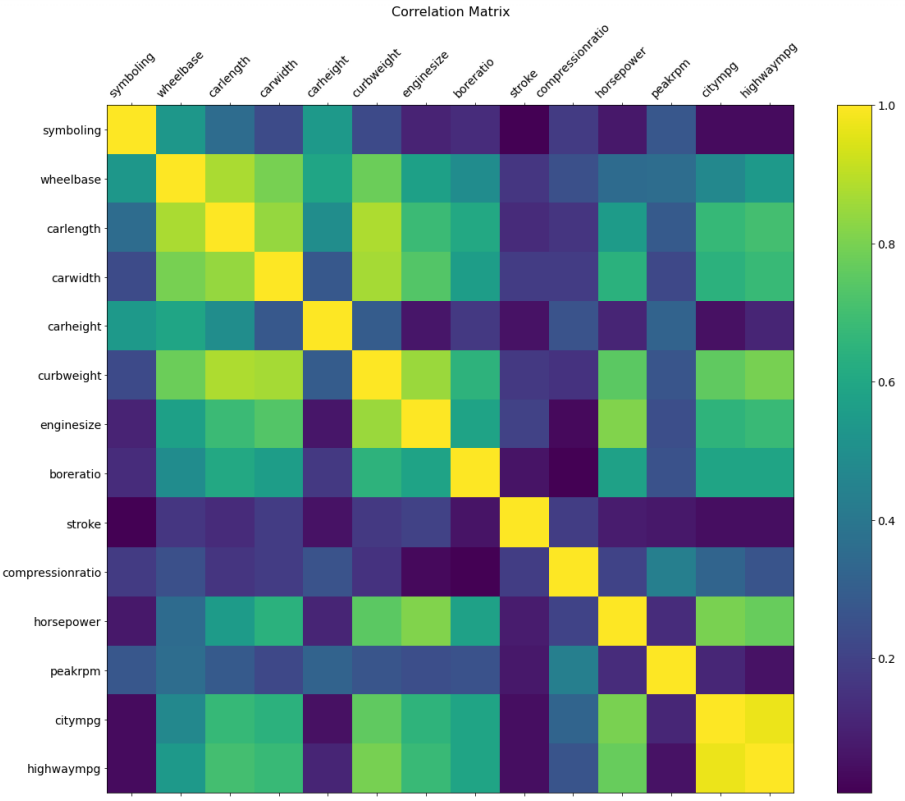


Figure 8: Correlation matrix of attributes

Figure 8 shows the absolute correlation matrix of the different attributes in the dataset. Here it easy to see that there are strong correlations between many of the different attributes, as many have absolute correlations values above 0.5. This makes sense as many cars in the same price range have many of the same features and specs, therefore high absolute correlation values between attributes are expected. In general, it shows us what we would expect given the results of our principal component analysis, specifically since so few dimensions are needed to capture most of the variance, the 14 features must have pretty significant correlations. Lastly, it also confirms some intuitive suspicions about the dataset like citympg and highwaympg would have near perfect correlation.

3.6 Outliers in the data

To investigate outliers in the dataset we created box-plots for all of the different attributes in the dataset (See 7.1). On these box-plots you can find all data points that are more than three standard deviations away from the mean. When looking for outliers among the different attributes, we observe that the amount of extreme observations vary a lot. Some of the attributes only have extreme observations on one side of the median, such as price which only has outliers to the right. One attribute that can look strange at first glance is the cylindernumber attribute. Almost all cars have four cylinders, which makes all cars that do not have exactly four seem like outliers, but we do not interpret this as a problem or mistake. We have decided not to remove any data points from the dataset as we don't believe it is necessary, as none of the points seem to be mistakes and because we do not believe it would change our analysis in a negative way.

In relation to outliers it is also important to look at the PCA, when we take the projected data onto principle components 7. Here we notice that there are no observation which can easily be perceived as an outlier, which lends credence to the perception we got from looking at each dimension individually.

3.7 Primary machine learning objectives

To asses whether the machine learning objectives seem to be feasible we can look at figure 7.a and b. From these images we see a clear non independence between the dimensions of the principal components and the price. For any supervised learning task, this dependence is what the model will use to predict the target feature. Moreover, this relationship seems somewhat straightforward and simple. Specifically most of the relationship seems capturable by linear regression. We believe this because most of the relationship seems linear, and the rest appears polynomial in nature. In short, we would be able to predict the price of a car from its input features, relatively effectively by employing a linear regression model with a polynomial basis function. Therefore, we believe the machine learning task to be quite feasible.

4 Discussion

We have learned two very important things about our dataset. First of all, we have learned that there is significant dependence between our the target feature and the first two principal components and that this relationship appears not too hard to capture. Secondly, we have learned that the vast majority of the variation in the dataset, whether we use all features, or only the quantitative ones, is capturable in merely two principal components. This will also in most cases imply that we can predict price accurately using nearly as accurately using only the first two principal components, thereby simplifying the prediction task enormously. More than these two very central pieces of information we have also gained a much greater understanding of each attribute. This includes understanding the distributions of each quantitative attribute, specifically that most are not normally distributed and the correlations between attributes. Concerning outliers we concluded based on viewing the projection onto principal components and viewing box-plots, that we have minimal reason to remove such extreme observations as they do not constitute anything we could describe as out of the bounds of the given category. Of course, such extreme observations might still have a negative impact on the supervised learning task, however this remains to be seen.

Lastly, we consider the machine learning tasks to be absolutely doable given the correlations, linear and non-linear, which we can see between the price and the first two principal components.

5 References

[1] Manish Kumar. *Car Price Prediction Multiple Linear Regression*. 2019. URL: `https://www.kaggle.com/hellbuoy/car-price-prediction`.

[2] Agata Rutkowska. *Car price prediction*. 2021. URL: `https://www.kaggle.com/arutkowska/car-price-prediction`.

6 Exam Questions

Question 1.

30 minute interval is of the type continuous interval. We assume exactly one of the *A-D* statements to be true, therefore because each of these statements can be distinguished by its categorisation of x_1, we simply use this to infer our answer.
Our answer is therefore that statement D is true, as it answers that x_1 is of type interval.

Question 2. Our assumption is again that there is exactly 1 correct answer. (excluding E of course). Our calculation for A is: max(|26-19|, |2-0|) = 7.
Our calculation for B is: $cubeRoot((26-19)^3 + (2-0)^3 = 7.054$
Our calculation for C is: (|26-19|)+(|2-0|) = 9
Our calculation for D is: $fourthRoot((26-19)^4 + (2-0)^4) = 7.0116$
Our answer is therefore A.

Question 3. Explained variance:
A: $\frac{13.9^2+12.47^2+11.48^2+10.03^2}{13.9^2+12.47^2+11.48^2+10.03^2+9.45^2} = 86.6\%$
B: $\frac{11.48^2+10.03^2+9.45^2}{13.9^2+12.47^2+11.48^2+10.03^2+9.45^2} = 0.4798\%$
C: $\frac{13.9^2+12.47^2}{13.9^2+12.47^2+11.48^2+10.03^2+9.45^2} = 0.5201\%$
D: $\frac{13.9^2+12.47^2+11.48^2}{13.9^2+12.47^2+11.48^2+10.03^2+9.45^2} = 0.7167\%$

Our answer is therefore A, as $0.86 > 0.80$. The rest are false.
Question 6. What is the probability an observation had $\hat{x}_2 = 0$ given light congestion?

P($\hat{x}_2$=0|y=2) = P($\hat{x}_2$=0 and $\hat{x}_7$=0) + P($\hat{x}_2$=0 and $\hat{x}_7$=1|y=2)
We then read the probabilities given by table 2, which gives us the following estimation for the probability of x2=0 given y=2.
P($\hat{x}_2$|y=2)= 0.81+0.03 Therefore our answer is B.

7 Appendix

The omitted principal directions of the 63 principal components are shown in figure 9

```
[-0.07564644  0.24079079  0.27156403  0.26664519  0.09371907  0.29660884
  0.25985144  0.21313852  0.02576023  0.0351583   0.24105849 -0.06437808
 -0.24495804 -0.25543236  0.03658691  0.06438262  0.12784334 -0.06243548
 -0.06037489 -0.07710915 -0.03558677  0.09057137 -0.03899292  0.01851687
 -0.0507785  -0.02414651  0.09319936 -0.04877475  0.04956824 -0.00282388
  0.01014018 -0.02243859 -0.029942   -0.02924686  0.08067819 -0.05230945
  0.08396462 -0.06393706  0.04796632 -0.10897489  0.0522467   0.03866374
 -0.22210728  0.22604137  0.0356189   0.07599126 -0.144581   -0.00321346
  0.12624141  0.00149915  0.08114141 -0.1912498   0.13745036 -0.04539005
  0.05613248  0.00149915 -0.18994922 -0.00092122  0.05230945 -0.00142337
  0.19213159 -0.02250145 -0.0047638 ]
```

(a) Vector representation principal directions on the first principal component

```
[ 0.23829725 -0.17673843 -0.09752211 -0.04472247 -0.26865797 -0.02001644
  0.0570065   0.02227738 -0.05696278 -0.28783739  0.19483067  0.23147904
 -0.17720892 -0.14345971  0.00855825  0.02914262 -0.06715998 -0.00615018
  0.01847947  0.01234135 -0.00225953  0.04582727  0.04671738  0.0189877
  0.05635859  0.02088506 -0.16293738  0.01157327  0.13695952  0.00659295
  0.0231322   0.00615176 -0.02427    -0.08141459 -0.04846664  0.29649975
 -0.10542596  0.2443302   0.05888532  0.19186286 -0.16533221 -0.09870698
 -0.0570867   0.05903939  0.06476765 -0.15668439 -0.10597546  0.05761647
  0.09929802  0.16468498 -0.06869988 -0.14037584  0.12351001 -0.00094378
  0.04980454  0.16468498 -0.02904704  0.14626251 -0.29649975  0.02714678
  0.12981803  0.06453475  0.01968669]
```

(b) Vector representation principal directions on the second principal component

Figure 9

This section will not include all the histogram plots and QQ-plots only those found troublesome to validate.

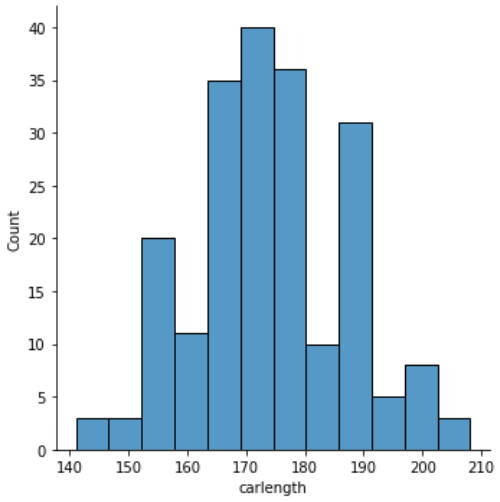


Figure 10: Histogram of carlength

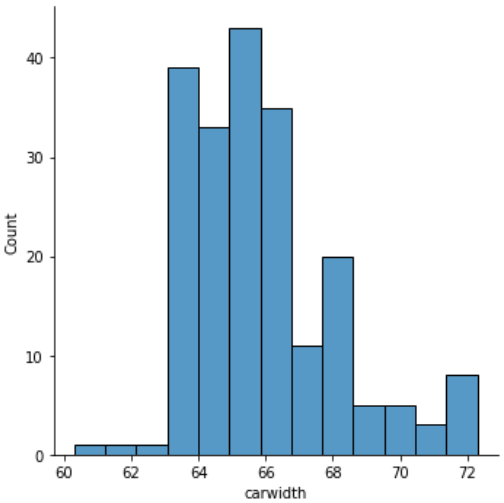


Figure 11: Histogram of carwitdth

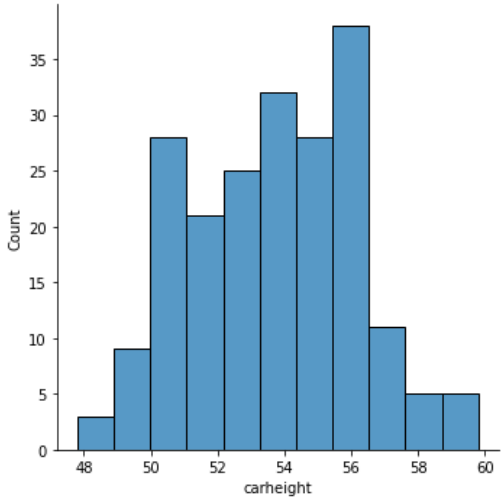


Figure 12: Histogram of carheight

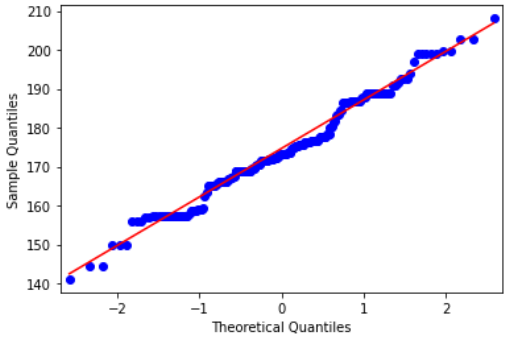


Figure 13: QQ-plot of carlength

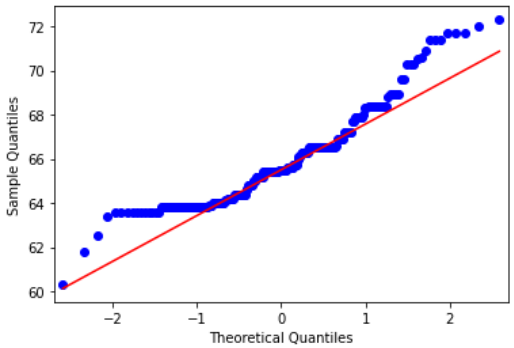


Figure 14: QQ-plot of carwidth

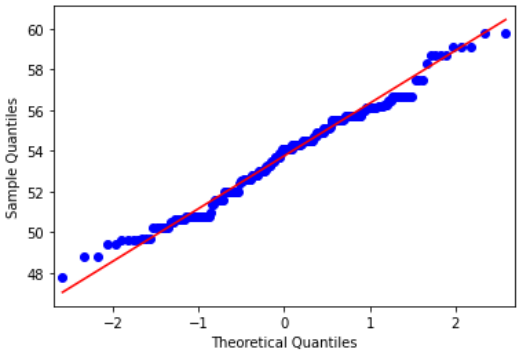


Figure 15: QQ-plot of carheight

7.1 Boxplot

We used boxplot to validate if the dataset has outliers. These are illustrated below:

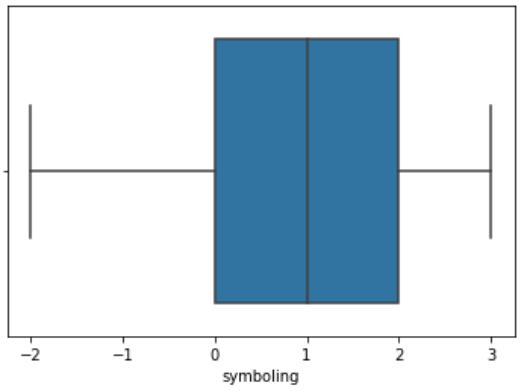


Figure 16: Boxplot of symboling

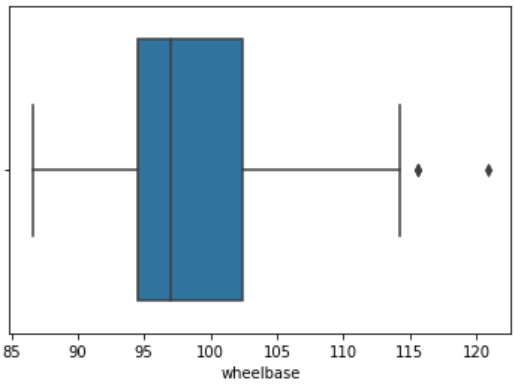


Figure 17: Boxplot of wheelbase

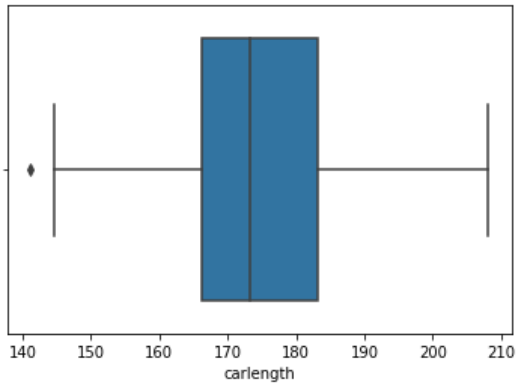


Figure 18: Boxplot of carlength

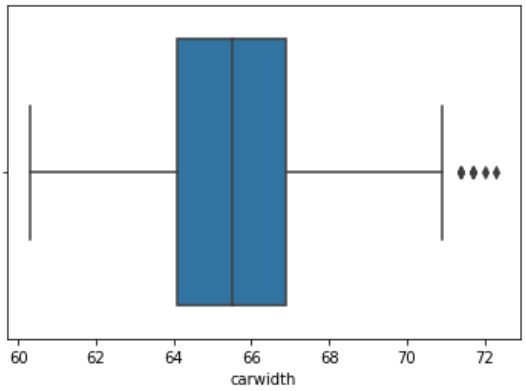


Figure 19: Boxplot of carwidth

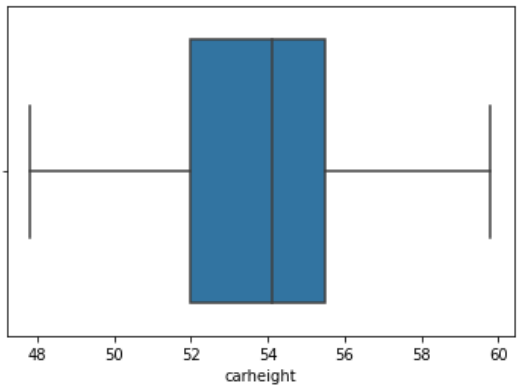


Figure 20: Boxplot of carheight

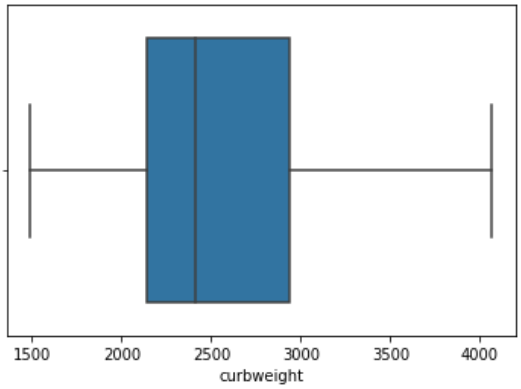


Figure 21: Boxplot of curbweight

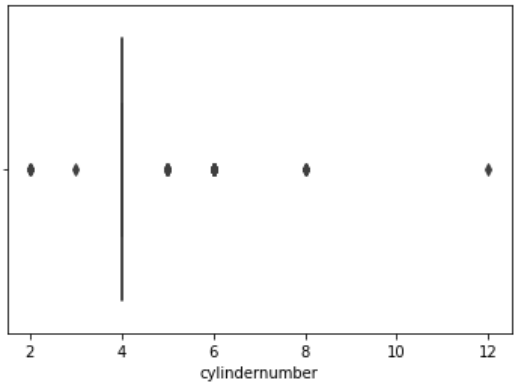


Figure 22: Boxplot of enginenumber

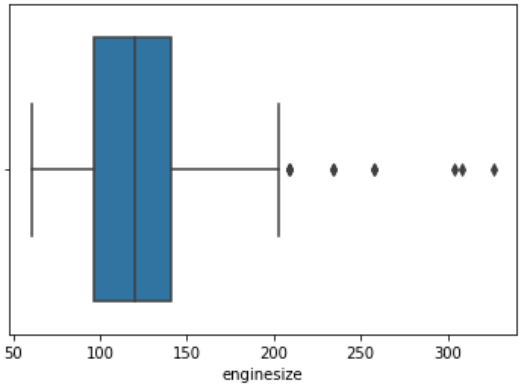


Figure 23: Boxplot of enginesize

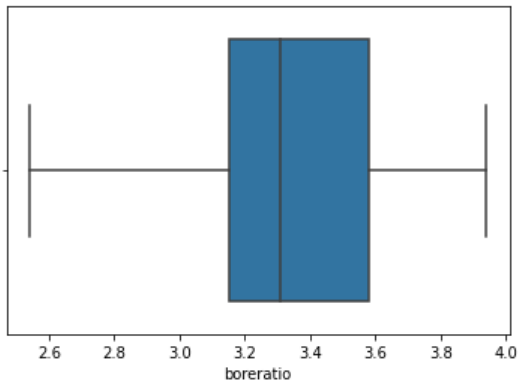


Figure 24: Boxplot of boreratio

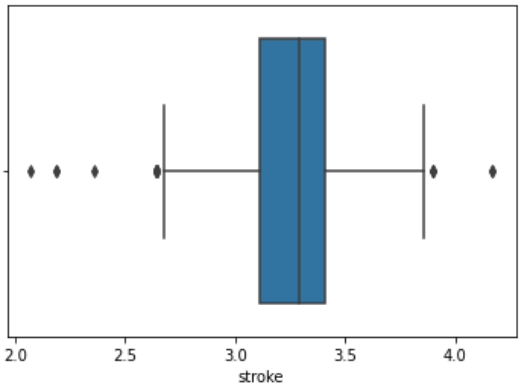


Figure 25: Boxplot of stroke

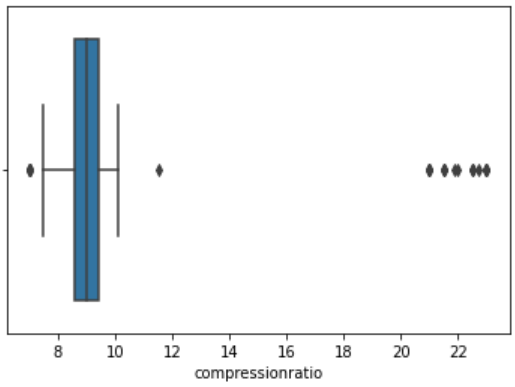


Figure 26: Boxplot of compressionratio

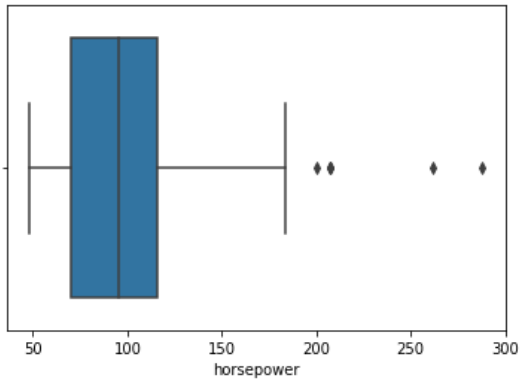


Figure 27: Boxplot of horsepower

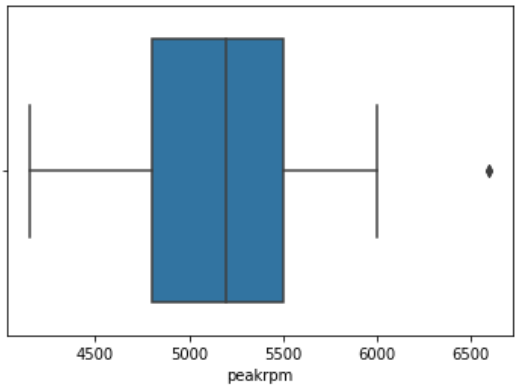


Figure 28: Boxplot of peakrpm

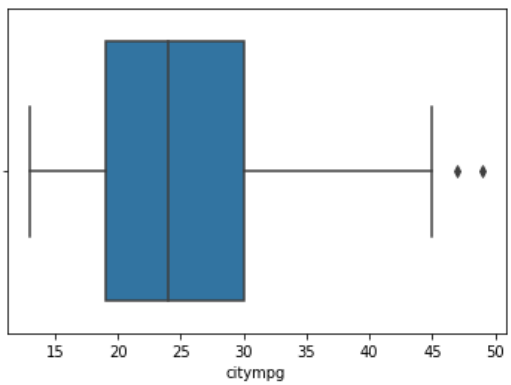


Figure 29: Boxplot of citympg

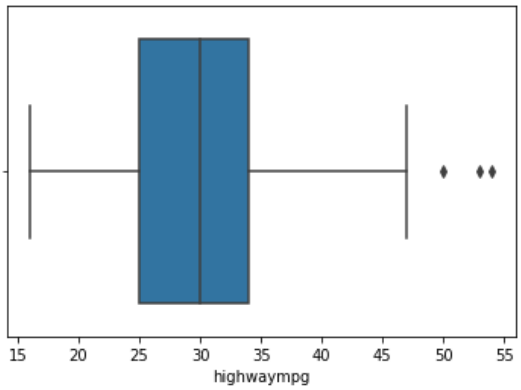


Figure 30: Boxplot of highwaympg

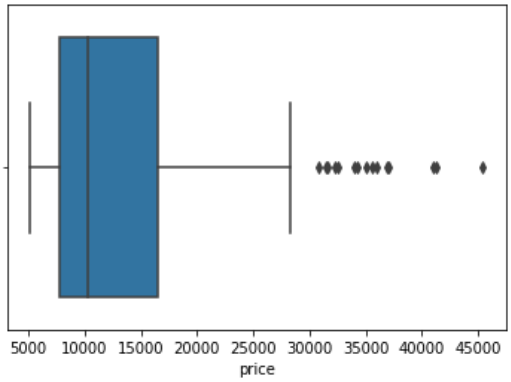


Figure 31: Boxplot of price

Danmarks Tekniske Universitet

Car Price Prediction - Part 2

Project by
by

Bjarke Erichsen (s204104)
Frederik Sartov (s204118)
William Peytz (s204145)

Introduction to Machine Learning and Data Mining, 02450

Technical University of Denmark

October, 2021

Contents

1 Regression, part a

In the last report we used PCA and found that with the first 3 principal components (PCs) we could represent around 96% of the variance in the dataset. We transform the data by projecting it onto the 3 first PCs, on which we will then use linear regression to predict the price. Specifically, with linear regression we will construct a function that maps from values spanned in the 3 dimensional space of the 3 PCs to the target variable which is the price of the car. Its important to note, that every dimension previously included in the dataset, is represented in these 3 PCs, including the categorical ones which where one-of-K coded and then standardized.

We have standardized our data matrix X, so that each feature has a mean of 0 and a variance of 1. We do this so that when a regularisation parameter is introduced it wont be biased in any way. Specifically, introducing the regularisation parameter means we want to also minimize the square of the weight vector ie. the vector consisting of slopes. Some features might simply have larger values and variance associated with them fx. horsepower, the weights associated with such features will be affected in an inverse way, thus making the weights arbitrarily smaller, which is undesirable. Importantly, we do the standardization before doing PCA and after having projected X into the PCA subspace, as this is the dataset we are actually going to be doing regularization on.

Significantly, when performing cross validation to find the optimal λ value, we make sure to project the data into a the subspace spanned by PCs calculated from the relevant subset of the dataset. This is to make sure no information from data-elements in the test set, is carried over to the training set, in short, we keep the test and train sets completely separate. Practically this means we do PCA only on the train set in each fold of the cross validation, we then train and test our model with the data projected onto the PCs created from the train set.

The following figures (see figure 1) plot the generalisation error, which is the mean of the squared residual between the prediction of price and actual price. As can be seen lower regularisation values produce better results, having tested regularisation values between 2^{-6} and 2^{6}, and then plotted logarithmically the values on the x axis. The generalisation errors is however only slightly better with the lower values as no great difference in generalisation error occurs before the testing with the largest regularisation values.

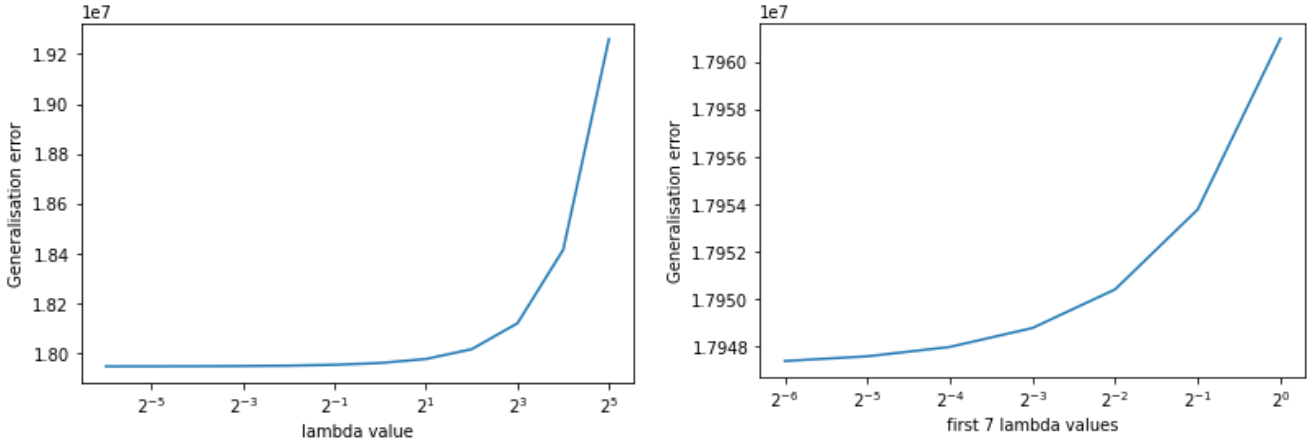


Figure 1: λ values plotted against generalization error

After training the model ie. picking the best weights corresponding to the optimal value of λ, we can make a new prediction. Importantly, to make new observations we need to make the same feature transformations, which where performed on the data we trained the model on. In this specific case, we project onto PCs created from the train dataset, and then standardize the projected input data with scalar values created from the dataset, lastly, we append a column of 1 values. Of course the dataset we compute the feature transformation on should not include the input data we are trying to map onto price. After the feature transformations are done, we can use the linear model, by using the following equation: $y = x \cdot w.T$. In short we matrix multiply the transformed input data with the transposed weights, which outputs the target feature, ie. price. All these steps can be performed with an arbitrary number of input observation at the same time. The actual results of the model is that the mean difference between a prediction of price and actual price, is around 2800 dollars using the best model and a MSE. Considering that the standard deviation of price in the dataset is around 8000, makes this not unrealistically far off what one might expect a linear regression models performance to be. The weights associated with each principal components and the bias term, of the final model, is shown on figure 2.

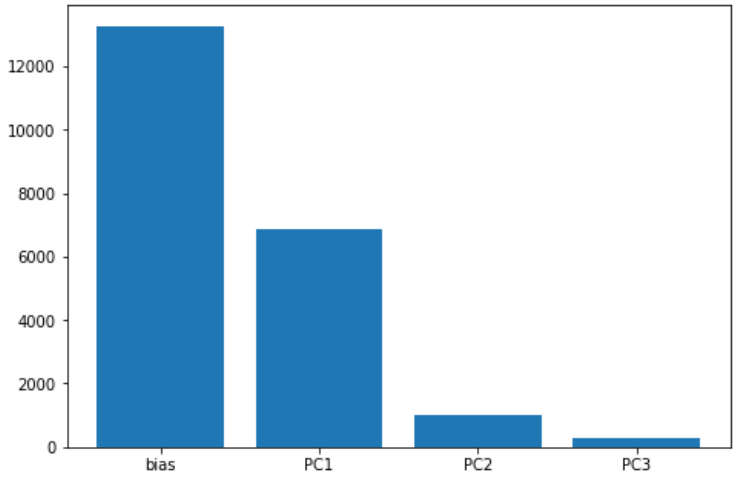


Figure 2: Bias and principal components

As can be seen, on bar plot, the first PC has by far the greatest correlation, while the two others only have slight positive correlations. The bias term of course just corresponds to the mean price, which is 13275 dollars.

2 Regression, part b

Using 3 different models to predict the price from the input features, and also different model parameters, we will find out what model is best. Best in this case means the minimisation of mean squared error, which is of course not the same as minimizing the absolute error, although we sometimes will convert the average mean squared error (MSE) to absolute error, to understand if our results are reasonable. The 3 models are a baseline model, a linear regression model (LR) and an Artificial neural network (ANN). We try out different configurations of the Linear regression and ANN model by varying the number of hidden units in the single hidden layer of the ANN model and by varying the regularisation factor when calculating the LR model.
Importantly, we transform the training and test data by using PCA, and we also standardize the results of PCA after this. The principal components we project the data upon and the standardization, is always calculated from the training set. Before performing 2 layer cross validation we somewhat arbitrarily choose the hyperparameters and parameter values we will explore which are listed in table 1.

ANN				
Number of hidden neurons range (h)	Number of iterations	Learning rate	Activation function	Batch size
1-9	500	0.001	Tanh()	25
Linear Regression				
Regularisation value range (λ)	n			
$2^{-n} - 2^{n}$	0-9			
2 layer Cross validation				
Outer folds ($K1$)	Inner folds ($K2$)			
10	10			

Table 1: Chosen parameters and hyperparameters

As seen on table 1, h and λ both include the value of 1. As neural networks come in many shapes it is appropriate to go into further detail about the exact structure of the NN we use. Doing a single forward pass in the neural network means giving the first 3 neurons the activation corresponding to the value of each of the 3 principal components. These activation's are then passed to the second layer ie. hidden layer with size h, where each connection contains a weight and each neuron a bias. The output of this we then put through the tanh activation function giving us activation in the -1 to 1 range. Lastly, we pass the activation to the output layer by weighing each connection and having a bias for the output, which is only a single neuron corresponding to the prediction of price. We update the parameters with gradient descent, we simply compute the gradient of the loss with respect to each parameter and then manually go through each parameter to update its value by the learning rate multiplied by its grad attribute ie. a single element of the gradient vector. A small note, in relation to the ANN we standardized y, to remove the otherwise reoccurring issue of exploding gradients, and then when testing the model did the reverse transformation

on the output of the model.

Using 2 layer cross validation with the models and parameter values as described, table 3 produced.

Outer fold	ANN		Linear regression		baseline
i	h^*_i	E_i^{test}	λ^*_i	E_i^{test}	E_i^{test}
1.0	1.0	5973577.16	0.25	6694658.4	51369639.79
2.0	5.0	10669399.01	0.0078125	13387455.01	44175864.67
3.0	6.0	30836539.84	0.015625	42611171.3	132622548.42
4.0	6.0	11125024.11	0.125	18083598.38	60110713.17
5.0	6.0	4811685.57	0.015625	8081206.92	39922275.68
6.0	6.0	7007396.37	0.125	9159078.7	38455481.8
7.0	8.0	19914372.89	0.015625	21729496.24	91973469.42
8.0	6.0	14049744.97	0.00390625	14194541.89	65077532.51
9.0	8.0	7315215.66	2.0	9472432.1	39849841.54
10.0	6.0	14360284.17	0.25	19439421.37	82392067.66

Figure 3: Two level cross-validation for the regression models

As the error is defined as the error per observation one might think that these values are too high. However as the price of a car is naturally a high number, even a reasonable predictor will have a high error when the error is squared as it is here. Therefore we can consider these numbers completely reasonable.
Moreover, while the optimum value of λ is small as in the previous section, it is however here much more varied. It seems at a glance that the optimum value of λ is simply less than 0.5 and that as long as this is true, the performance of the linear regression model is not greatly affected. Concerning the number of hidden units, it is clear that somewhere between 5 and 8 is optimal with especial emphasis on 6 being particularly good, although this might simply be a statistical fluke. Lastly, we might at a glance see that at least the baseline model is by far the worst, although both linear regression and the ANN model seem to perform somewhat similarly with the ANN model being, maybe, slightly better.

For the statistical comparison of the models we choose to use setup 1, where we use paired t-tests as described. The output of those comparisons are as described in figure 4.

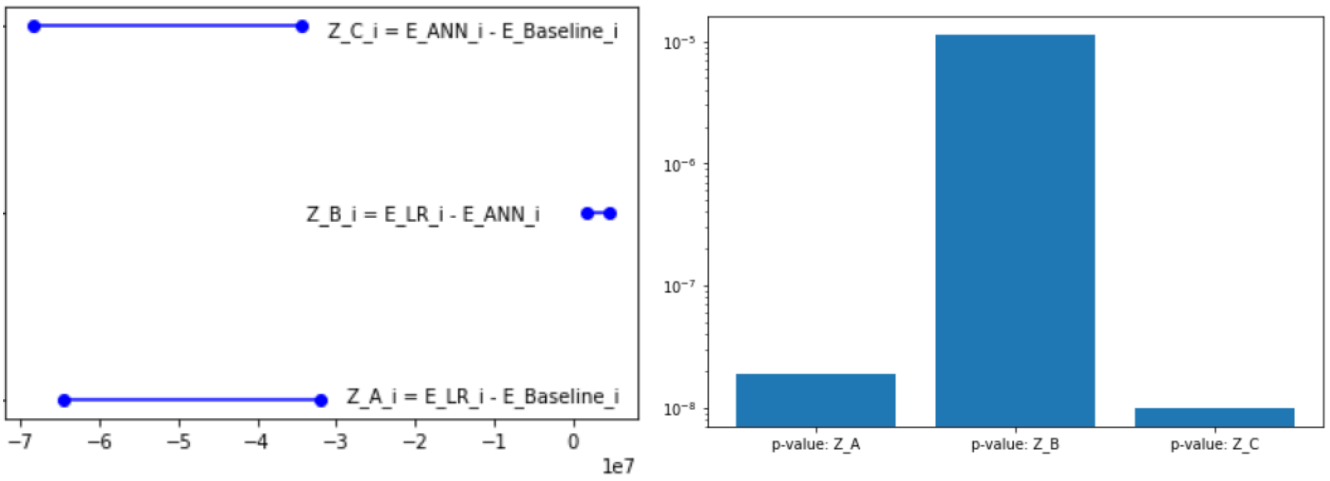


Figure 4: Confidence interval and p-value for regression models

Comparing LR to Baseline	Confidence Interval	p-Value
	[-64491454, -32008565]	$1.870 \cdot 10^{-8}$
Comparing LR to ANN	Confidence Interval	p-Value
	[1755300, 4491431]	$1.136 \cdot 10^{-5}$
Comparing ANN to Baseline	Confidence Interval	p-Value
	[-68313619, -34433131]	$9.942 \cdot 10^{-9}$

Table 2: Model comparison with statistical evaluation values

Interpreting figure 4 and table 2 we denote in the plots how the differences are calculated, so that we can interpret the confidence interval. Moreover, when writing "comparing model a to b" we mean that we subtract the error for a given observation of model b, from the error of model a. More broadly, we can say a negative confidence interval, like when we compare LR to baseline, indicates that Baseline has greater error than LR. Specifically that the confidence interval denotes the likely real average differences in MSE, in this case, that the baseline model has 32008565 to 64401454 more average error than LR.

With the ability to interpret the plot and table, we can see that the ANN model is better than the linear regression model, and that the linear regression model is better than the baseline model. The models thus form a hierarchy of performance with ANN first, LR second and baseline third, with all differences being statistically significant as can be ascertained from the p-value being below 0.05. Specifically, we can see that when comparing each model, the p-value is far below 0.05, meaning the The H_0 hypothesis which states that each pairing of models will have equal performance, is false for all 3 comparisons.
Furthermore, the difference between the linear regression model and the baseline model is considerably bigger than that between the linear regression model and the ANN model, although both differences remain far beyond statistical significance. All of these conclusions are valid for choices of alpha (pertaining to statistical significance) down to around 0.00001, ie. far beyond any reasonable doubt. Moreover, because we used the statistical evaluation techniques pertaining to setup 1, we must stress that the conclusions validity are limited to situations where we are training the models on the same dataset as this.

3 Classification

When talking about cars, the price is often the deciding factor, when it comes to choosing what vehicle to buy. Therefore it can be nice to know what different options you have, depending on your budget. Moreover, car manufacturers might also be interested in knowing what a reasonable price range for a given vehicle is. Therefore a classification of cars in relation to price can be interesting. We have decided to create a multi-class classification with five different car price tiers, which are as follows: cheap, affordable, normal, expensive and luxury. Each of these five classes have their own predefined price range according to the following table:

Class	**Price range**
Cheap	\$0 - \$5.000
Affordable	\$5.000 - \$15.000
Normal	\$15.000 - \$40.000
Expensive	\$40.000 - \$100.000
Luxury	\$100.000 and above

Table 3: Price classes used for target prediction

We encode our original dataset with the specified ranges seen on table 3, so that a car with a price of between 0 and \$5.000 will be encoded as cheap, which we represent by the number 1. This encoding scheme is applied to the entire dataset giving us a target feature consisting of integers in the 1 to 5 range, representing the 5 different classes.

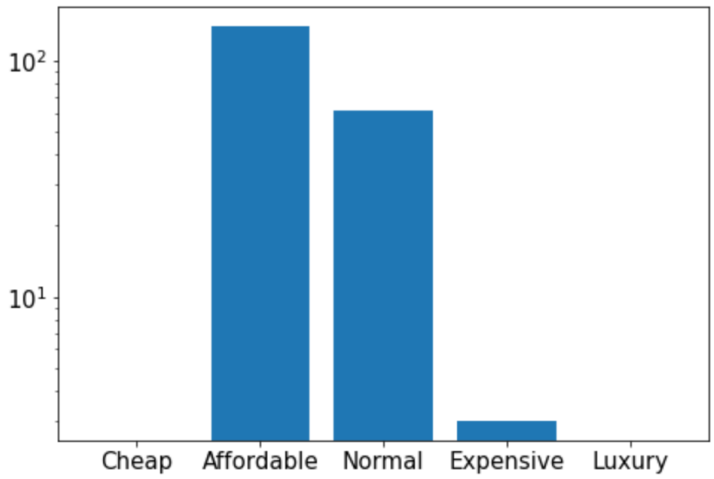


Figure 5: Distribution of categories

As seen on the figure above, the categories are not based on percentiles of the dataset but on our apriori understanding of where the category boundaries should be. Therefore, we would not expect there to be an equal amount of cars in each category from our data set. This makes our dataset very unbalanced as seen on the table 5, and although this will mean that the models will have higher accuracy, it wont affect our ability to compare the models in an unbiased way since we are using two level cross validation. The 3 models we use for classification are a baseline model, a Multinomial Logistic Regression model (MLR) and a K-Nearest Neighbors algorithm (KNN). We try out different values of hyperparameters for the MLR- and KNN-model by varying the regularisation factor λ and the number of neighbors for the respective models. The chosen range of values are represented in table 4. The baseline model is a simple MLR model without any specific λ input, and works by predicting that everything is equal to the largest class in the training set. fx. if there are more cars in the "affordable" class, then it will predict that everything in the validation set corresponds to this class.

K-Nearest Neighbors	
Number of neighbors (k)	k
Range of neighbor values k	1-10
Multinomial Logistic Regression	
Regularisation value range (λ)	n
$2^{-n} - 2^{n}$	0-9
Two level Cross-validation	
Outer folds ($K1$)	Inner folds ($K2$)
10	10

Table 4: Chosen parameters and hyperparameters

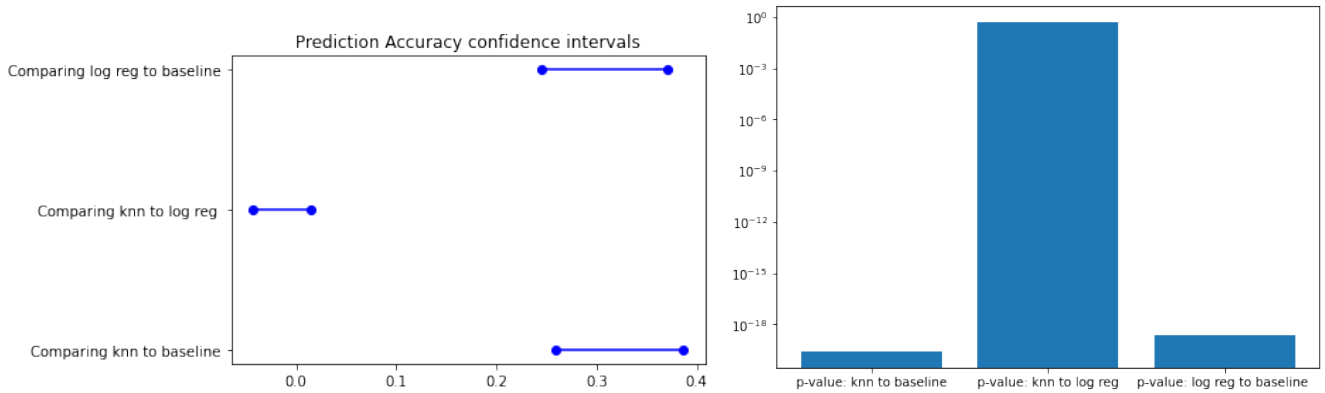
After using the chosen two machine learning models and a baseline model, we can compare their different accuracy's to find out which method is preferred. We see that the worst model by far is the baseline model, as it has an accuracy of approximately 63.4%. Even though it is simple and has a lower accuracy, it is still good to test, as this gives us a baseline of what accuracy we should beat when using other models, as if we got an accuracy of less than this baseline accuracy that model would be useless. The logistic regression model has an accuracy of 85.3% which is a lot better than baseline, but it also means that there is room to grow as the accuracy still is far from perfect. The best of the three model is the k-nearest neighbor model. This is because it has the highest accuracy of 87.8%, which is better than both the baseline, as well as the logistic regression model. One of the key differences between the logistic regression and the k-nearest neighbor model, is that logistic regression is a linear classifier, whereas KNN is a non-linear classifier, and we would therefore expect KNN to be better able to predict price although it will also have a greater tendency to overfit because of its complexity.
Using two level cross validation on the classification models the best parameters were ascertained and the model accuracy calculated. The values are found table 6.

Outer fold	KNN		Logistic regression		baseline
i	k^*_i	E_i^{test}	λ^*_i	E_i^{test}	E_i^{test}
1.0	8.0	0.9	0.125	0.9	0.62
2.0	10.0	0.86	1.0	0.86	0.52
3.0	3.0	0.81	1.0	0.76	0.76
4.0	5.0	0.9	2.0	0.81	0.76
5.0	5.0	0.85	256.0	0.9	0.8
6.0	9.0	0.9	0.125	0.8	0.6
7.0	9.0	0.85	0.125	0.85	0.7
8.0	8.0	0.9	0.25	0.95	0.65
9.0	10.0	0.95	0.5	0.95	0.6
10.0	9.0	0.95	0.5	0.95	0.85

Figure 6: Two level cross-validation for the classification models

From table 6 the models' accuracies are found in E_i^{test} and the best hyperparameters are located in k_i^* and λ_i^* for KNN and MLR respectively. From the 10 outer folds the best fold for KNN is 9 and 10 with a neighbor values of 10 and 9, giving an accuracy score of 95%. The MLR model achieves its highest accuracy of 95% at λ value 0.5 and 0.25, while the most frequent occurring λ value is 0.125. The baseline model's greatest accuracy is 0.85. This accuracy stems from this outer fold containing primarily affordable classes (see table 3) ie. we have a very unbalanced dataset. It must be said that these test values will statistically over estimate the average case for the given parameter value. This is because each test value has been selected because it has the highest average test score of all the parameters, which in non technical language means we also will select a lucky model, that is biased towards a higher test score.

To ascertain whether or not the models significantly differ from each other, we will calculate the confidence interval and p-value with McNemar's test. The statistical results from this test are shown on table and a plot is given for illustration purposes.



Comparing MLR to Baseline	Confidence Interval	p-Value
	[0.2449, 0.3714]	$2.168 \cdot 10^{-19}$
Comparing KNN to MLR	Confidence Interval	p-Value
	[-0.0434, 0.0140]	0.508
Comparing KNN to Baseline	Confidence Interval	p-Value
	[0.2588, 0.3868]	$2.711 \cdot 10^{-20}$

Table 5: Model comparison with statistical evaluation values

From the information on table 5 we can begin to form an image of how well the models compare. The confidence interval shall be interpreted as follows: $\theta = \theta_A - \theta_B$ where subscript A and B denotes the classification model. If $\theta > 0$ model A is preferable over B and so forth.
So from this information it can be ascertained that the KNN model is preferable to the baseline model. For the comparison of the KNN model to the MLR model the confidence interval overlaps with 0, which entails that we cannot say one model performs better than the other, with the required statistical certainty. Comparing the baseline to the MLR model, we see a negative confidence interval, this means the MLR model is better, as intuition would foresee.
Letting the H_0 be the null hypothesis that the models have the same performance and the significance value be $P < 0.05$. Overall, we see from the p-values that we can with a great statistical certainty disprove the H_0 hypothesis when comparing the baseline model and the two other models. In greater detail, when looking at the comparison between the baseline and MLR the statistical significance is far below the significant value, which removes all doubt that MLR is the preferable model. For the comparison between KNN and MLR the p-value is significantly larger than the threshold of 0.05, as the p-value is 0.508, which means that H_0 cannot be rejected. In practice it means that we are not able to say that one model is better than another. This, too, is indicated on the two level cross validation table where the model performances are calculated to be 95% for some instances for both KNN and MLR.
As mentioned in the regression part, these conclusions validity are limited to situations where we are training the models on the same dataset as this. Furthermore, we should also mention that because of our considerably unbalanced dataset, the absolute accuracy's become very large, as can be seen by the baseline being above 50 % accurate. However this unbalance does not affect our ability to come to definitive conclusion about what model is best given we limit these conclusions to the particular encoding scheme and as mentioned, the particular training set we work with.

In order to make the optimal logistic regression model, we first need to choose a suitable λ, for our model. We do this by picking the most frequently occurring λ in our two level cross validation (see table 6) for classification, where the λ value, λ=0.125 had the highest frequency at 3. Other values in a similar size range would also be suitable. In relation to how the logistic regression model actually works, it is based on a sk-learn model called LogisticRegression, which also works for multinomial linear regression. An important distinction to make is the difference between normal logistic regression and multinomial logistic regression. The primary difference is that in multinomial logistic regression, the output does not have to be binary, which means that it can be used to

categorize more than two classes. This is important, because for our case we have five different classes, which means that binary classifiers wouldn't be sufficient. In regards to how logistic regression works, it simply takes the sum of each input variable weighted by a trained weight parameter and then puts the resulting value through a sigmoid function. If the probability of this variable being one class is above 50% the model will output 1 and if it is below 50% it will output 0. This also work with multiple independent variables, where again after taking all the independent variables into account, the model will output 1 if the probability is above 50%. This whole classification process is also know as a maximum likelihood function. For multinomial logistic regression this wont work as there are more than two discrete possible outcomes. Logistic regression can be thought of as a linear neural network with a softmax activation function which makes it capable of multi-class prediction. We add all of the different weights of the different outputs where all the different outputs are then divided by this new found sum. By doing all of this the different outputs will get a new value between 0 and 1, which corresponds to the probability of that given output. All of these new values will of course add up to exactly 1, thus corresponding to a probability. Notably, we chose to use multinomial logistic regression for our ordinal target feature, even though you normally use if for nominal features. However, as can be ascertained from the models performance, the model still performs impressively well.

Since logistic regression corresponds to a linear neural network with no activation function, it should be apparent the multi-class neural network allows us to extend logistic regression to the multi-class setting.

Lastly, the following table describes the weights of the multinomial logistic regression model.

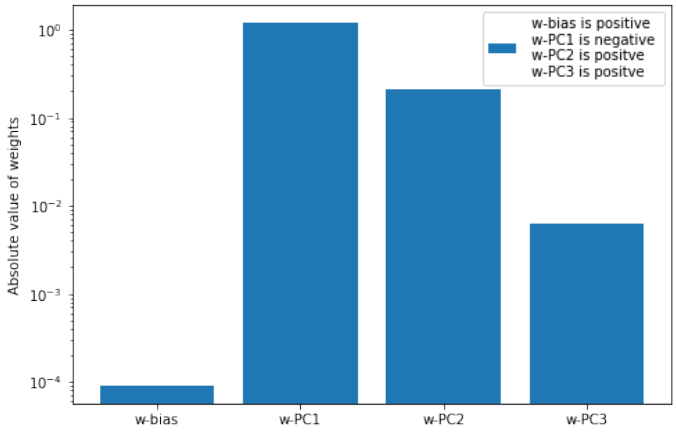


Figure 7: MLR weights

We use the table above to infer if the features that are relevant are the same as in the regression part. Immediately we can say that the importance of features remain relatively similar. However, we see that especially the second input feature ie. PC2 has a greater weight associated with it here than in the linear regression part, thereby implying a greater significance. The hierarchy of PC1 being most important followed by PC2 and then PC3, still remain the same although with much less difference between PC1 and PC2.

4 Discussion

In both the regression and classification parts of our report, we have learned that the baseline models have been greatly outperformed by both the linear models (Multinomial Logistic Regression and Linear Regression) and the non-linear models (KNN or ANN). Moreover, we have also learned that simply increasing the complexity of the model does not necessarily yield a better result, in short, there seems to be a middle ground of complexity that is ideal. This was seen clearly on display, when the optimal neural network had around 6 hidden neurons, and in the number of k in k nearest neighbors that was ideal, although one could argue if a higher or lower value of k that makes the model more or less complex. The ANN model was the optimal model for the regression problem and both KNN or MLR was optimal for the classification problem where we could not decide which of these was best. Furthermore, we learned that the features had very different impact on price, with the first feature PC1 having by far the greatest correlation, followed by PC2 and the then PC3. Moreover, the same hierarchy of importance existed with the features in the multinomial logistic regression however with PC2 and PC3 having comparatively slightly higher weights associated with them. In both classification and regression we also learned that the linear models showed the greatest performance with a relatively low λ value, although for logistic regression it was higher than linear regression.

Our car price dataset has been used in other studies before, and it can therefore be interesting to compare their results with ours, to see how others tackled the dataset and see if used the same or other models than us. One

of the studies about this car price prediction dataset was created by Manish Kumar [1]. Before going into his findings, it is very important to note, that it is very hard, and in many cases misleading to compare accuracy's of models from different studies, because the way the study is set up can bias the result a lot. In our case, our five different classes, with different price ranges, turned out to not be balanced at all between the five classes, which meant that almost all of the different cars in our data set, ended up in either the "Affordable" or the "Normal" category, with almost no cars in the other three categories. Because of this, even the simplest of models, such as the baseline, would get a relatively high accuracy, by just guessing on the most common price range within the training set. Therefore comparison of regression instead of classification models, is to be preferred in the circumstances where the categories are not exactly the same. When it comes to multiple LR a difference between our study and Kumar's preliminary study is that we look at all of the different variables to get the best model, whereas Kumar tries different combinations of just normal linear regression as well as multiple linear regression. As an example he tries doing multiple LR with the correlated variables "horsepower", "curbweight" and "enginesize". To measure the accuracy of his model he uses the r^2 value, which for this specific test was 0.819. This means that with his prelimanry study, Kumar finds out that a lot of the correlation between car features and car price, can be found within just a few correlated variables. After his preliminary study Kumar uses recursive feature elimination, to check if the different variables are dependent or independent. If he finds a variable to be independent, the variable is removed from the training set. After doing RFE, his model gets an r^2 value of 0.936. This means that with this method Kumar was able to get an even better model, which also just confirms that you are able to create a model which predicts the price of a car, if you know enough of the car features.

5 References

[1] Manish Kumar. *CarPrice Prediction MLR+RFE+VIF*. 2019. URL: `https://www.kaggle.com/hellbuoy/carprice-prediction-mlr-rfe-vif/notebook`.

6 Exam Questions

Question 2:
We use the formula for purity gain. $\Delta = I(r) - \sum_{k=1}^{K} \frac{N(v_k)}{N(r)} I(v_k)$ and as described in the problem we use the impurity measure named Class error, given by the formula $1 - \max_c p(c|v)$

We then simply input the numbers, subtracting the impurity of the root with the weighted impurity of each branch. With K being 2 since we are simply splitting into two branches, we end up with 2 expressions we have to add. However one of these expression becomes 0, as the impurity function is 0. Specifically because we only have one element, $p(c = 2|v)$ will end up being exactly 1 for the branch where x7 is present. Putting it all together and excluding the unnecessary computations, we get the following.
$\left(1 - \left(\frac{33+4}{33+28+30+29+4+2+3+5+1}\right)\right) - \left(\frac{33+28+30+29+4+2+3+5}{33+28+30+29+4+2+3+5+1} \cdot \left(1 - \frac{33+4}{33+28+30+29+4+2+3+5}\right)\right) = \frac{1}{135}$ As $\frac{1}{135} = 0.0074$, the answer is therefore C.

Question 3: There are seven inputs, four targets and a single hidden layer containing 10 units. This means we have $7\dot{1}0 + 10$ parameters between input and hidden layer and $10\dot{4} + 4$ parameters between the hidden and output layer. Using this information the amount of parameters is= 124, the answer is therefore A.

Question 4:

In order to find the splitting rules we compare the decision tree with the classification boundary. We notice that if rule A and C both are true, the congestion is level 4. Now we look at the classification boundary. Here we notice that $b_1 > -0.16$ and that there are no rules for b_2 within the given interval. Rule A and C says that $b_1 > -0.76$ and $b_1 > -0.16$. Since these are the only two rules, and since all other options don't give the correct boundary for congestion level 4 the answer must be D.

Question 5:
The time it takes to create the table is calculated by the following expression.
$5 \cdot (4 \cdot (5 \cdot (20 + 5) + 5 \cdot (8 + 1)) + (20 + 5) + (8 + 1)) = 3570$
The answer is therefore C

In short, the calculations above comes from the 5 outer folds which we can see from the table that we are using. Inside each of these 5 folds, we have to calculate, for each of the 4 inner folds, the generalisation error of each model. Since both the logistic regression and ANN models have 5 different parameter values, we have to compute generalisation error 5 times on each model. Lastly, the $(20 + 5) + (8 + 1)$ part is done for each of the 5 outer folds, as we train each model, using the parameter value pertaining to the lowest generalisation error, on D^{par} and test on D^{test}.