# M3 stakeholder report

This stakeholder report is covering the M3 assignment in the social data science course on AAU. The assignment is centered around the use of neural networks and how they can be utilized, trained and tuned. This report will also dip into the process of preparing the data for the network.

## Motivation

We had an idea, that one could create an application for the mobile, where you could take a picture of your car and input various information, such as the year it was manufactured, the manufacturer brand, kilometer count etc. With this information the application would give you a rough estimate of the price that one could expect for the car, when selling it on the market for used cars.From this, we can formulate the following problem statement:

> *"Is it possible to train a neural network, to give a rough price estimate on a used car, given an image and easy to access numerical and categorical data?"*

This means that we wanted to make use of various data types and combine them in order to try and train a neural network to make a price estimate. We have found examples of how to combine data when working with neural networks, as well as a dataset that contains all that we deem to be necessary for the project. We want to tackle the problem, by creating price categories, where each category span a certain price range. By splitting the price estimate into price categories, we turn the pricing into a multinomial problem.

## The dataset

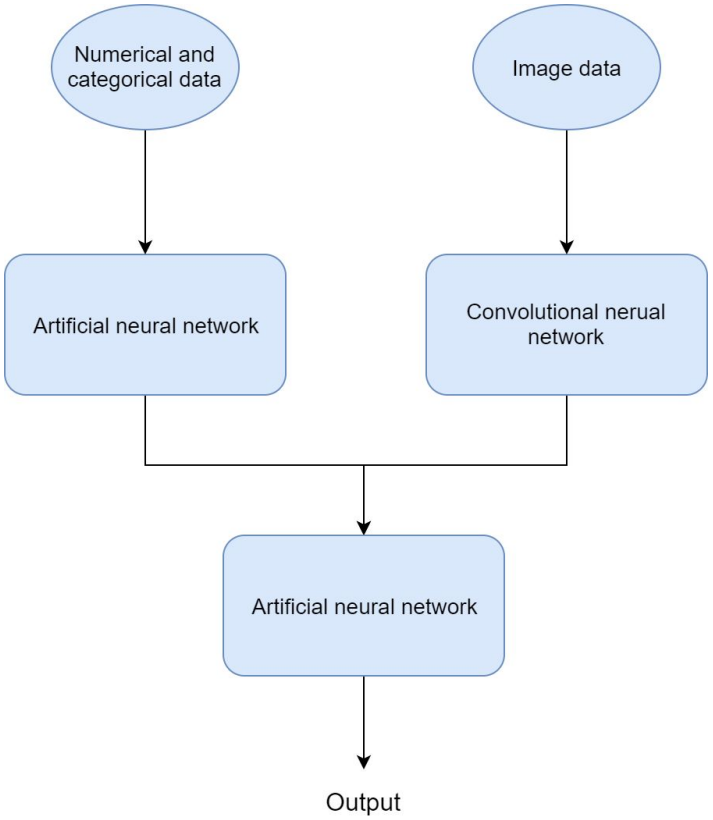
We found a dataset containing over 500.000 entries of used cars for sale on craigslist. each entry had several pieces of information about the car, as well as a url with an image of the car. After removing the entries that were missing what we deemed to be vital information, or contained nonsense, the dataset had shrunk 150.000 entries, which is still more than enough to work with. We discovered that some of the image urls were broken, but this would prove to be a different matter, as we could not process 150.000 entries, including trial and error, within a reasonable timeframe. We therefore decided to pick out 50.000 samples at random and only work with those entries and their corresponding images.

### Gathering images

Checking and gathering the images took a tremendous amount of time when we tried the first time we tried. But with the help of multithreading we managed to check and get the images, and the final set of entries had shrunk down to about 41.500, due to some urls not working. The multithreaded processing of the image urls is reflected in the colab notebook, and due to the extensive code needed to prepare the data, this code have been collected in a separate colab notebook.


## Building the neural network

The neural network is constructed using keras functional API, as we would need to combine 3 different networks in order to process the mixed data. The network that is processing numeric and categorical data is an artificial neural network (ANN) and the network that processes images is a convolutional neural network (CNN). These are concatenated and the output is processed in yet another artificial network



# Training the network

Before we could train, we had to make sure that our dataset was split correctly, so that the images that is being fed to the convolutional network, matches up with the numerical and categorical data, that is being fed to the artificial neural network. Although keras offers many functionalities for working with images, the toolset is much smaller when it comes to working with mixed data. We handle the challenge by making two datasets, the entries would match each other one to one. This allows us to use regular data splitting on both the images and numerical data, and get two datasets that would correspond correctly. We do not perform any further data splitting or random selection of samples.

The combined network is trained with the training data set, until the accuracy no longer seem to be improving. We stop the training using some of the built-in functions in Keras. The training is performed on several different combinations of artificial neural networks and convolutional networks in order to expand the potential for better results.

## Tuning

We wanted to be able to fine tune all sorts of network combination. Tuning a network built with keras sequential API can be done relatively easy, but many of these conveniences are not available for the functional API. There are many different ways of overcoming this challenge, and due to the time constraint of the project, our solution is relatively simple. We create functions that allow for easy implementation of different kinds of networks, both ANN and CNN. We then run a predefined set of network combination, much in the style of the regular hyper parameter tuning of sequential Keras networks. Our function tries all the combinations in a typical brute-force manor, instead of utilizing some of the tricks that the sklearn library offers.

The different network combinations have a direct effect on the result, and so more network combination could be tested in the future.

# Results

The best result of the training of the combined network is not very good, only about 6 percent, which is much to low for any real world application. Our baseline machine learning models, have an accuracy of about 25 percent, without any image data, so we are nowhere close to the performance of the much less computationally intensive machine learning algorithms.

Another aspect could be our knowledge of which features that impacts the price of a car, we are just making a rough guess at what would impact the price, as commoners with no special knowledge about cars.

Interestingly, our models seemed to perform better on their own, then when they are combined. This could be due to the datasets, images / numerical and categorical data, not

really complementing each other and thereby actually performing worse, when evaluated together. Ideas for how this can be improved is discussed in the section on future work.

# Future work

## A different kind of problem

Our attempt at this problem was by splitting the price into price categories of a given price range. We have found other examples that tries to estimate the prices of houses, given some data about the house and 4 images per house, each with the same angles / rooms. The price is then estimated with a linear activation function, thereby making it a regression problem. This approach could be interesting to mimic in our setup, as this could potentially give a more accurate price, instead of just a price range, as our current project is formulated.

## Less noise in the image dataset

The example that estimated house prices where also very consistent with the images that were used. Upon further inspection, we see that many of our images are not representative of the car that was sold, which could explain why our results aren't all that good.

## Reliability of the dataset

A possible issue with the dataset is the reliability. The data is scraped from craigslist, a site that is a multipurpose site, where there are many people who don't make serious posts. An example of a better scraping could be, if bilbasen here in Denmark was scraped. Bilbasen is only for cars and has both new and used cars, and is only for cars, you won't see a Ford Focus to one million kr in there, the prices would be more realistic and less people "having fun" on the site and therefore increase the reliability of the dataset.

## Further network tests

The possibility of trying different kinds of networks with different kinds of layers, with different amount of neurons, are there. Our function setups were rather hardcoded, and could very well be optimized to add and remove different layers and test with different kinds of layers in different sequences and change the amount of neurons in each layer and change the sequence of the neurons on each layer. There are a lot of possibilities, but this also creates the problem of time consumption. Not only do you have to be precise with the function, but also the runtime would be increased exponentially, while also depriving the system of its resources.