UNIVERSITY OF COPENHAGEN

Predicting apartment prices in Copenhagen using Artificial Neural Networks

Amalie Starcke Thorsen (nxv252), Caroline Hedstrøm (zrx592),
Viola Longfors Hansen (pfq529) & Franziska Krug (pnt237)

Exam numbers: 69, 85, 142 & 192

Social Data Science

Summer 2018

Submitted on: September 1st, 2018

Number of pages: 17 (13 normal pages)

Splitting by sections: 1 and 10 was done together.

192: 2 (first half), 4 (first half), 6, 6.1, 7.2, 8.2 (Precision), 9 (second section)

69: 2 (second half), 4 (second half), 6.2, 8.1, 8.2 (Recall), 9 (third section)

142: 3, 3.1, 5 (first half), 6.3, 8.2 (until Confusion matrix), 8.2 (F-score), 9 (fourth and fifth section)

85: 3.2, 3.3, 5 (second half), 7, 7.1, 8.2 (Confusion matrix), 9 (first section)

Contents

1 Introduction 3

2 Literature Overview 4

3 Data sources 5

3.1 Web scraping data from Boliga 5

3.2 “ Hvorlangterder.dk ” API 5

3.3 Merging the Boliga dataset and the distances 6

4 Data cleaning 6

5 Descriptive Statistics 6

6 Methods 10

6.1 Preprocessing 10

6.2 Multilayer perceptron 10

6.3 Model Selection 11

7 Analysis 11

7.1 Target and features 12

7.2 Building the models 13

8 Results 13

8.1 Regression model 13

8.2 Classification model 14

9 Discussion 16

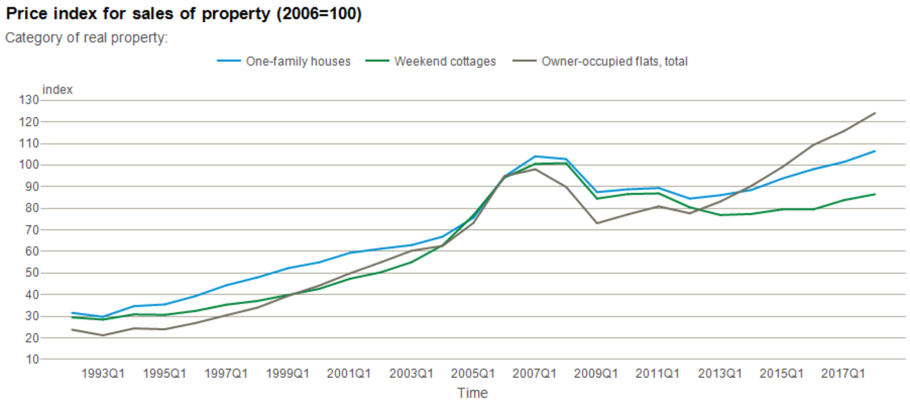
10 Conclusion 18

References 19

1 Introduction

In the past three decades, housing prices in Denmark have been characterized by drastic ups and downs. Figure 1 illustrates these developments: From 1993, there was a persistent increase, becoming even steeper in 2005/2006 and eventually experiencing a sharp drop in 2008/2009. The housing market started to recover in 2013 and despite the burst of the housing bubble during the financial crisis, property prices rose back to high levels by historical standards. Other countries have exhibited a similar pattern of dramatically increasing house prices in the run-up to the international economic crisis, followed by a sharp downturn, while the recovery of the markets in the aftermath of the crisis was more heterogeneous across countries (Bergmann et al., 2015).

Figure 1: Housing prices in Denmark since 1993



Source: Statistics Denmark (2018)

Since developments in the housing market have been shown to be leading indicators for a country's economic activity, predicting house prices has become a tool to forecast the phase of the business cycle as well as output (Plakandaras et al., 2014). Numerous studies have tried to develop models that are able to predict housing prices and could thus not only be used as indicators to stakeholders in the real estate market, but could also assist policy makers to find timely real estate policies and thus to sustain the recovery of the housing markets and of the economy in general (Park and Bae, 2014; Plakandaras et al., 2014).

In our project, we develop such a model for Copenhagen using machine learning. We obtain data on 7,329 apartment sales in Copenhagen from the Danish webpage www.Boliga.dk and combine it with information on distance to various destinations such as to the closest supermarket, school or public transport stations. We thus have a dataset consisting of information on the apartment itself, i.e. address, sales price, number of rooms, building year, and square meters, but also on proximity to destinations that might influence the buying choice. Using a neural network we then try to predict the sales price based on the property's attributes. Accuracy checks reveal that our model correctly predicts 97 percent of the sales prices within an interval of one million.

The remainder of the paper is structured as follows. Section 2 provides an overview of the existing literature on the prediction of house prices. Section 3 describes the data sources and the data collecting process, Section 4 explains the required cleaning of the data and Section 5 gives summary statistics on the variables. In Section 6 we discuss the methodology we apply to predict housing prices in Copenhagen. Section 7 and 8 describes the analysis using machine learning and the results. Finally, Section 9 discusses the results and Section 10 concludes.

2 Literature Overview

The literature on the prediction of house prices is divided into two strands, depending on the method applied. The first strand utilizes hedonic pricing models, which allow to construct housing price indexes based on certain attributes of the property. Thus, the value of one house relative to another is due to differences in the set of attributes, such as number of rooms, age of the house and geographical factors (Bin, 2014). Despite its usefulness in accounting for unobservable housing attributes (such as air pollution or noise), hedonic estimation is prone to certain pitfalls with regard to model assumptions and estimation. One potential problem arising with hedonic estimation are the strict requirements on the functional form. (Selim, 2009) An alternative that does not face these limitations is offered by machine learning algorithms, which we will use in the subsequent analysis. Recent literature has compared the performance of both approaches. Selim (2009) examines determinants of house prices in Turkey employing both hedonic estimation and Artificial Neural Network. His results demonstrate that Artificial Neural Networks perform better than hedonic approaches when there is potential non-linearity in the hedonic functions.

Within the literature, there is also a discussion on which machine learning is best suitable for price predictions. Jirong et al. (2011) use a hybrid of genetic algorithm and support vector machines (G-SVM) and a conventional grey model (GM) to predict national average housing prices in China. Data for the years 1993 to 1998 are employed as training data, while data for the years 1999 to 2002 serve as test data. Comparing the absolute relative errors of both approaches, the results reveal the G-SVM approach as more accurate than GM. Plakandaras et al. (2014) also develop a novel forecasting methodology, EEMD-SVR, based on Empirical Mode Decomposition (EEMD) used in signal processing combined with Support Vector Regression (SVR) from the field of machine learning. Applying their methodology on U.S. house price data spanning the years 1890 to 2012, they find that the autoregressive version of their methodology, EEMD-AV-SVR, outperforms all alternative forecasting tools (e.g, Random Walk models or Bayesian VAR). The EEMD-AV-SVR yields an in-sample-accuracy of 84.615 measured by the Mean Absolute Percentage Error and was able to predict the drop in the U.S. housing market prices in 2006-2009 up to two years ahead. Park and Bae (2014) focus on data from Fairfax County, U.S., and show that for this sample the RIPPER algorithm is superior to other methodologies. Ahn et al. (2012) study the performance of Ridge Regressions when forecasting the Home Sales Index and the Home Rental Index in the South Korean real estate market. The danger of using plain Ridge Regression is that it might turn insignificant variables into significant ones. Applying GA-Ridge, the authors are not only able to mitigate this problem, but also show its superiority in prediction accuracy compared to other algorithms. Other papers have made use of neural networks, which we also apply in our project. Kauko et al. (2002) use two neural network algorithms, SOM and LVQ, to capture house market segmentation in Helsinki. Both algorithms exhibit a high classification accuracy and successfully point towards the location of market segments and their determinants. However, the authors call for a careful exploration of the robustness of the results and emphasize that conventional statistical methods would have performed equally successful. Chiarazzo et al. (2014) focus on how transportation and environmental factors affect real estate prices. The authors feed an Artificial Neural

Network with data from the Italian city of Taranto and find that the algorithm yields a good fit in predicting real estate prices. Sampathkumar et al. (2015) compare the performance of multiple regression techniques and Artificial Neural Networks when forecasting land prices in the Chennai Metropolitan Area, India. While both approaches do well in predicting house prices for the years 2014 and 2015, based on data from 1997 to 2011 and validation data for the period of 2012 and 2013, the Artificial Neural Network methodology is characterized by a slightly higher accuracy. Finally, Wilson et al. (2002) report promising results of the Artificial Neural Network's ability to forecast 2-year house prices in the UK.

3 Data sources

This section presents the data sources used in this project. In order to get data regarding sales prices on apartments in Copenhagen, the Danish website www.boliga.dk was used. To get additional information about distances to e.g. public transport and schools the API for hvorlangterder.dk was used.

3.1 Web scraping data from Boliga

www.boliga.dk is a freely accessible database containing information about every property sold in Denmark since 1992. In order to get the relevant information, the data were scraped. The scraping was done in accordance with `https://www.boliga.dk/vilkaar-og-betingelser` which specifies the disallowances when scraping Boliga data.

Initially, four helper functions were created. The first function was made to obtain the right URL to the wanted data. The second function calculates the maximum number of pages to scrape given the zip codes and time period. This function was made because we wanted to be able to change the zip codes and the time period. The third function scraped a single web page obtaining the wanted information. The scraping was done using the `Request` and `BeautifulSoup` modules. Finally, the data was converted into a dataframe. The fourth helper function creates a for loop over all the pages calculated in the second helper function. After scrapping all the pages into one large dataframe, some cleaning and converting of the data was done.

Finally, the master function was created by wrapping up the four helper functions into one. The master function `get_boliga_data(zipcode, minyear, maxyear)` takes three inputs: i) zipcode: defining which zipcodes we want data for, ii) minyear: defining the starting date for our data and iii) maxyear: defining the latest date we want to obtain data from. This function was the one used to obtain the Boliga data set.

3.2 “ Hvorlangterder.dk ” API

`Hvorlangterder.dk` is a Danish page where you can type in an address and get the distances to the nearest school, doctor, hospital, train station, bus station, s-train station etc. From the API we constructed a function called `get_dist(address)`, to obtain distances in meters using a given address. We chose to get the distances when going by bike. Alternatively we could have picked the distance by car, by foot or by public transportation.

After constructing the function, it was applied to the addresses from Boliga. Since some of the addresses were spelled differently in the Boliga data and the data from hvorlangterder.dk, it was necessary to use the `try-except` function. The `try-except` function prevents the procedure from stopping if it cannot be executed on the specific observation. Instead it just passes the observation and moves on to the next. Fortunately, the data only contained two observations which could not be used in the function.

3.3 Merging the Boliga dataset and the distances

The Boliga data and the distance data were merged together using the variable `address`. The merge was done using an inner join. This is important because otherwise the observations with the misspelled addresses would still be in the data set, but without the added distances.
Because the distances are calculated from the addresses in the Boliga dataset, the two datasets contain the same observations except for the misspelled observations. Furthermore, there were two apartments that were sold twice in the considered time period and therefore created duplicates when merging the two datasets. This problem was dealt with using `.drop_duplicates()`.

The merged data set was saved as a csv file. The csv file was used for the rest of the project, to avoid having to run all of the scraping and API again, and to make sure we worked with the same observations throughout the project.

4 Data cleaning

As mentioned in the Section 1, during the past 30 years house prices have been far from stable. Except for the drop in the mid-2000s, the data show a clear trend towards increasing prices. In order to avoid having to account for these time effects, we restrict our analysis to the years 2017 and 2018, i.e. to the prices from the period ranging from January 1, 2017 to August 24, 2018.
We further limit our data to the dwelling type: Lejlighed (apartment), which is the most common dwelling type in Copenhagen. As sales types, family sales were not included since they were found to occur at a significantly lower price compared to sales among non-family members. Furthermore, the sales types "Auktion" (auction) and "Andet" (other) were not included.
Neither the target, nor the feature variables suffer from missing values. There was one observation which was reported to have 0 square meters, and which was subsequently dropped.
When inspecting the data further, 6 observations were found with a sales price less than 200,000 DKK. By researching the specific addresses online, these were found to be obvious errors and they were removed from the data.
After cleaning the data, the finial data set consists of 7,329 observations. The analysis was done on these observations.

5 Descriptive Statistics

Table 1 presents summary statistics on the observations in our dataset. After merging and cleaning the data from the Boliga webpage and hvorlangterder.dk, the dataset consists of 7,329 observations. The target variable is sales price, measured in Danish Kroner. The average sales price for an apartment in Copenhagen in our dataset is 3,448,998.33 DKK, ranging from 272,000 DKK to 37,403,058 DKK. This corresponds to an average price of 40,525.49 DKK per square meter, while the average property bought consists of roughly 3 rooms with an average of 83.57 square meters. Boliga furthermore provides information on the year the property was built, which was on average, in 1937. Besides the number of rooms, the size of the property in square meters and the building year, the distance information obtained provides a valuable addition to the feature variables since access to public transport, such as bus or metro stations, as well as proximity to supermarkets and to schools are important determinants in the decision-making process. The distance information panel in Table 1 displays details on the distance in meters to 10 destinations that arguably play a role when choosing a property. In order to provide more detailed information on the target variable, sales price, Table 2 shows descriptive statistics broken down by area and quarter of the year. In the time period considered, most sales occurred in the København S and Ø areas. Properties in Nordhavn, København K, København V and København Ø yielded, on average, the highest sale prices. As illustrated in Figure 2, the average price per square meter in these areas exceeds the average of the whole sample and confirms that these four areas are indeed the most expensive ones in Copenhagen. The mean sales price per quarter of year remained roughly constant over time and reassures that our analysis should not be biased by time trends. Since the third quarter of 2018 is still going on when writing this paper, not all sales for the third quarter of 2018 have been reported which is reflected in the low number of observations for this time span.

Table 1: Descriptive Statistics

	N	**Mean**	**Std**	**Median**	**Min**	**Max**
Sales information						
sales_price	7,329	3,448,998.33	2,049,572.8	2,995,000.0	272,000.0	37,403,058.0
m2_price	7,329	40,525.49	12,130.19	39,743.0	3,126.0	5,34329.0
rooms	7,329	2.74	1.07	3.0	1.0	9.0
m2	7,329	83.57	36.45	75.0	16.0	386.0
building_year	7,329	1937.4	53.97	1935.0	1623.0	2018.0
Distance information						
school	7,329	535.18	341.64	452.7	30.23	2,319.78
junction	7,329	3,181.37	1,315.19	3,029.71	165.79	6,764.11
metro	7,329	1,886.52	1,201.83	1,782.48	52.08	5,468.95
daycare	7,329	300.85	181.25	266.41	17.47	1,417.16
doctor	7,329	515.47	410.51	382.32	15.57	2,537.82
hospital	7,329	2,412.34	1,273.78	2,072.93	139.32	6,605.79
stop	7,329	229.87	149.74	195.14	12.65	1,078.51
supermarket	7,329	303.46	222.15	263.0	11.74	1,814.75
strain	7,329	1,846.28	1,589.43	1,060.69	52.08	7,351.26
train	7,329	2,321.39	1,039.05	2,333.62	52.08	4,864.74

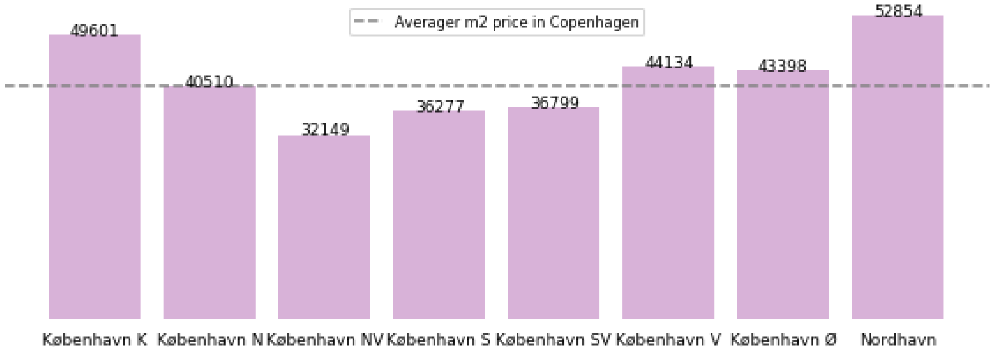
Note: Source: The sales information were obtained from www.Boliga.dk for the period between 01/01/2017 until 24/08/2018. The distance information for the addresses were downloaded from Hvorlangterder.dk. Prices are in DKK and distance is measured in meters."Stop" refers to the closest bus stop and "strain" to the closest stop of the so-called "S-Train".

Table 2: Descriptive Statistics on Sales Price by Area and Quarter of the Year

	N	**Mean**	**Std**	**Median**	**Min**	**Max**
Area						
København K	848	4,918,287.63	2,919,952.92	4,225,000	841,000	37,403,058
København N	848	2,757,632.27	1,413,596.92	2,427,500	300,000	19,500,000
København NV	541	2,149,842.75	780,141.57	1,950,000	458,000	4,805,388
København S	2,284	2,850,581.17	1,343,266.97	2,542,500	272,000	18,500,000
København SV	475	3,009,961.88	1,307,922.21	2,995,000	645,237	11,810,000
København V	655	4,010,862.16	1,767,886.48	3,750,000	5,500,00	17,000,000
København Ø	1,558	3,981,527.14	2,023,664.81	3,495,000	365,652	18,935,619
Nordhavn	120	6,955,587.25	4,106,827.39	5,899,000	460,651	24,155,000
Quarter						
2017_1	1,277	3,326,472.03	1,981,358.8	2,895,000	272,000	24,155,000
2017_2	1,420	3,527,361.6	2,110,448.82	3,147,500	509,590	37,403,058
2017_3	1,204	3,455,141.25	2,048,,404.64	2,985,000	550,000	21,500,000
2017_4	1,160	3,445,779.88	2,036,237.18	2,952,500	300,000	18,995,000
2018_1	1,093	3,588,052.0	2,173,184.67	3,100,000	272,000	3,2600,000
2018_2	1,013	3,393,615.54	1,958,424.38	2,925,000	458,000	18,935,619
2018_3	162	3,113,471.62	1,746,707.32	2,550,000	380,000	1,1600,000

Note: Source: The sales information were obtained from www.Boliga.dk for the period between 01/01/2017 until 24/08/2018. Prices are in DKK.

Figure 2: Average price per m2 in different areas of Copenhagen



To investigate the correlation between the variables, we plotted the correlation in a `sns.heatmap`, which is shown in Figure 3.

The heat map represents the correlations between all of the variables. We can see that rooms and square meters are both highly correlated with sales price (and with each other). This is to be expected, as it means that more square meters in an apartment come along with higher sales price. The same applies for the number of rooms in an apartment.

We can also observe that metro is negatively correlated with junction and s-train. This is also not surprising, as there is only one metro stop in Copenhagen that is in the same station as an s-train stop (Nørreport) and since the metro runs only in central Copenhagen, which is far away from the junction. Month and quarter are, by obvious reasons, perfectly correlated.

Figure 3: Correlation

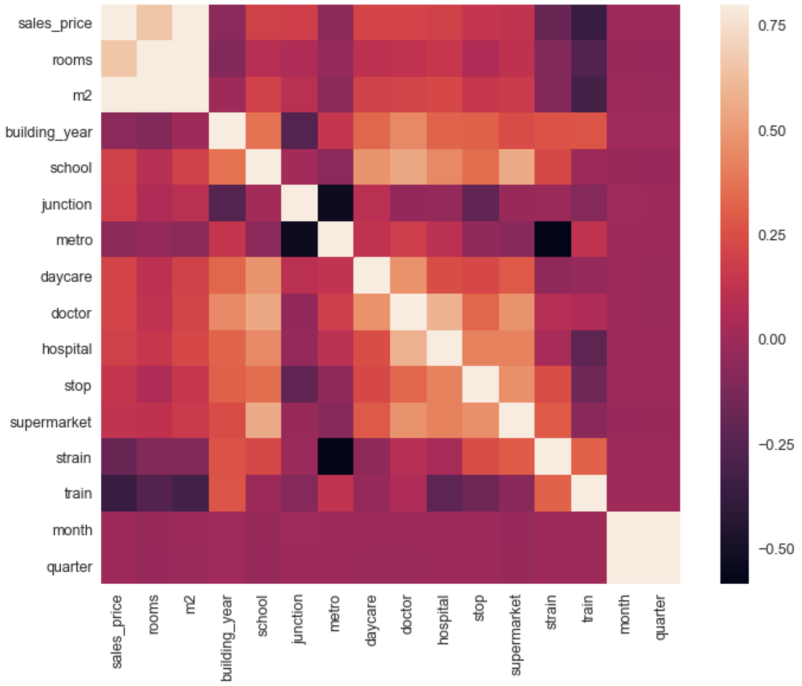
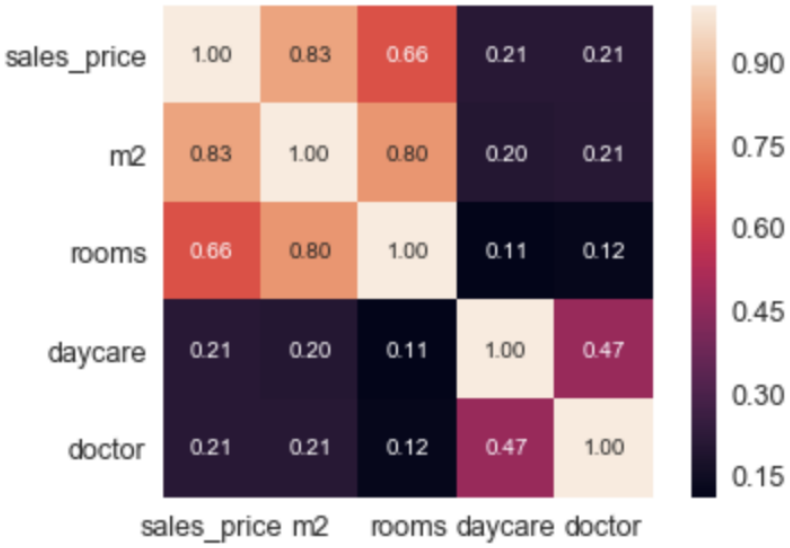


Figure 4 illustrates the four variables that are most correlated with the target variable - sales prices.

Figure 4: Top 5 correlation



6 Methods

To predict the apartment prices, a supervised machine learning algorithm was applied to the collected data, and different parameter values were tried out to find the best model (hyperparameter tuning). For the chosen algorithm to perform well, some preprocessing had to be done with the raw data. The preprocessing of the data and the chosen learning algorithm are described in more detail in the following subsections.

6.1 Preprocessing

To ensure optimal performance, it is important to investigate whether the correct features are selected. If some features are highly correlated, they will, to a certain extent, be irrelevant for the model, and it will be better to not include them. Including them will make the learning algorithm run slower, and in some cases it may also lower the predictive performance of the model. In addition to the feature selection, it is also important that the features of the model are on the same scale. In this project, the chosen scale was a standard normal distribution with zero mean and unit variance. Furthermore, the data should randomly be split into two parts: a *training data set* and a *test data set*. The training and optimization of the model is done solely using the train data, while the test data is kept to evaluate the final model. The parameters for feature scaling should only be obtained from the training data, but should also be applied to transform the test data (Raschka and Mirjalili, 2017, p. 12-13).

6.2 Multilayer perceptron

The first step of a supervised machine learning algorithm is to train a model with labeled data which can afterwards be used to make predictions about unseen or future data. Within supervised machine learning, methods can be applied to predict categorical variables as well as continuous variables (Raschka and Mirjalili, 2017, p. 3). In this project, both approaches have been utilized. We have trained a model to predict the exact price of an apartment, and we have trained another model to predict in what interval the price of the apartment should be. Both tasks have been conducted using a *Multilayer Perceptron* (MPL), which is a multilayer feedforward neural network. Figure 5 illustrates the structure of a MLP with three layers: one *input layer*, one *hidden layer* and one *output layer*. More hidden layers can be added to create a deeper network architecture. This type of network is fully connected. All of the nodes in the input layer are connected to the hidden layer, which is fully connected to the output layer. If more hidden layers are added, they are connected to each other. Nodes in different layers are connected via a weight coefficient. (Raschka and Mirjalili, 2017, p. 384-385).

The first step in the MPL learning procedure is to calculate an output by forward propagating the patterns of the training data through the network. At each node in the hidden layers, the net input is calculated using the weights and the output from the previous layer. The activation function takes the net input as an argument, and calculates the output that is used in the next node.

The second step of the procedure is to use the final output from the network to calculate the error which you wish to minimize using a cost function. In this project we have chosen to optimize

the squared-loss using a stochastic gradient-based optimizer known as "adam". Going into detail about the choice of cost function and optimizer is beyond the scope of the project.
The third and final step of the procedure is to backpropagate the error to update the weights used in the model (Raschka and Mirjalili, 2017, p. 387).

After training the model using the training data, the unseen test data can be used to calculate the network output. If the outcome variable is categorical, a threshold function is then applied to predict class labels (Raschka and Mirjalili, 2017, p. 387). If the outcome variable is continuous, it is predicted using the network output as an argument in a final activation function (Chiarandini, n.d., p. 56).

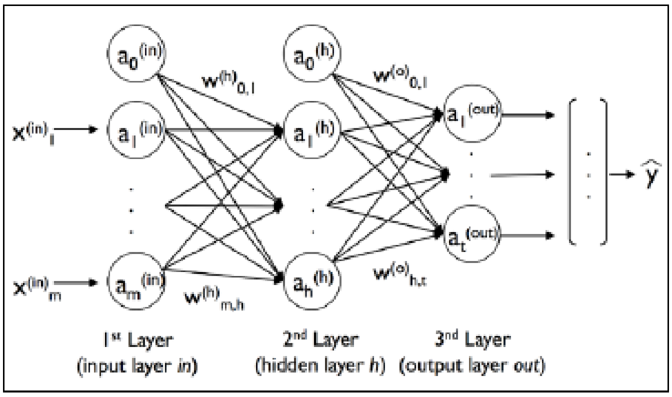
6.3 Model Selection

The model which is trained by the MPL algorithm varies with the parameter inputs, also known as *hyperparameters*, e.g. the number of layers and the number of nodes in each layer. To get the best model, different combinations of hyperparameters should be evaluated. This process is called *Model Selection*. The test data cannot be used to conduct the model selection, as it can lead to overfitting, i.e. the model fits the test data well, but does not perform very well in predicting new data. A solution is to split the raw data into not just a training and a test set, but also a *validation data set*. The new training and validation data set can be seen as two subsets of the original training data set, which will be referred to as the *development data set*. With this method, the training data can be used to train the model, while the validation data set can be used to do the model selection. The test set is still kept to evaluate the model's ability to generalize to new unseen data.
The described method has the disadvantage that it may be very sensitive to how the development data is split into training and validation subsets. A solution can be to do *K-fold Cross-Validation*. When using this method, the development data is randomly split into k folds without replacement. Then $k-1$ folds are used to train the model, and the last fold is used to evaluate the hyperparameters. This is repeated k times, so each fold is used as validation data once, and k performance estimates are obtained. The results are used to calculate an average performance of the models, which is less sensitive to the splitting of the data. Once this is done, and the optimal hyperparameters have been found, the model can be retrained on the complete training set (Raschka and Mirjalili, 2017, p. 190-192). When the chosen learning algorithm contains several hyperparameters, the optimal combination of hyperparameter values can be found via *grid search*, which includes cross-validation. The approach is to specify a list of values for each hyperparameter which should be tuned, and then the computer searches for the combination which returns the best model performance (Raschka and Mirjalili, 2017, p 201).

7 Analysis

In order to build models that can predict the sales prices for apartments sold in the period 01.01.2017 to 24.08.2018 we used the procedures described in Section 6. First, we built a regression model to predict the sales price of an apartment. Second, we built a classification model in order to predict in which price interval the sales price of an apartment were in. Both models were employed using a neural network algorithm. In the following it is described in more detail

Figure 5: Neural network



Source: Raschka and Mirjalili, 2017, p. 384.

how we applied the procedures to our specific data.

7.1 Target and features

In both models the sales prices of apartments in Copenhagen was used as the target variable. In the regression model, the target variable was the realized sales price whereby it was split into intervals for the classification model. We constructed the following price intervals:

- price below $1,000,000$ DKK
- price between $1,000,000$ DKK and $2,000,000$ DKK
- price between $2,000,000$ DKK and $3,000,000$ DKK
- price between $3,000,000$ DKK and $4,000,000$ DKK
- price between $4,000,000$ DKK and $5,000,000$ DKK
- price between $5,000,000$ DKK and $6,000,000$ DKK
- price between $6,000,000$ DKK and $7,000,000$ DKK
- price between $7,000,000$ DKK and $8,000,000$ DKK
- price between $8,000,000$ DKK and $9,000,000$ DKK
- price between $9,000,000$ DKK and $10,000,000$ DKK
- price above $10,000,000$ DKK

As our data only contains 93 observations with a price above $10,000,000$ DKK, these form the highest interval.

In both models we included 16 features - 13 numeric and 3 categorical:

Numerical: rooms, square meters, building year and the distances to school, junction, metro, daycare, doctor, hospital, stop, supermarket, s-train and train.

Categorical: city area, sales quarter and sales year.

Note that we treated rooms and building year as numeric, although they arguably could have been treated as categorical. Note further, that we dropped the variable price per square meter because it is linearly correlated with the sales price. Lastly, price reduction was not included in the model, because this was only indicated for a few apartments.

In light of the comparably small number of features (compared to the 7,329 observations), we did not see the need to perform any further feature selection before making the neural network. Also we did not import all variables from the hvorlangterder.dk API - only the ones we found relevant.

For our model to work, we needed the categorical features to be converted into dummies. This was easily done by applying the command `.get_dummies()` on the data set.

To ensure that our data did not overfit the data, we split it into a development and a test data set. The test set was set aside to be used only for the final testing. For the classification model we specified that the price intervals should be distributed in the subsets as in the original data set.

7.2 Building the models

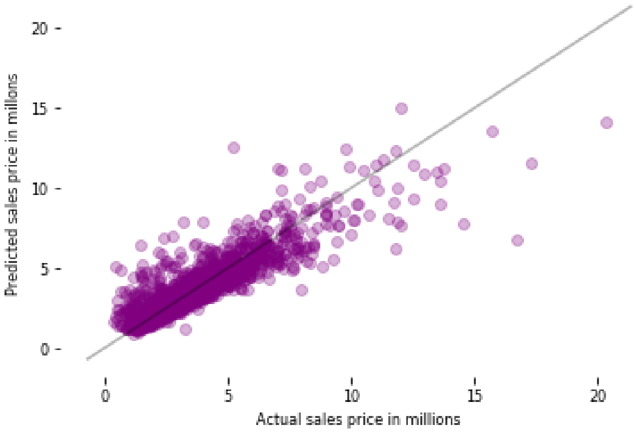
We built both models using a multilayer perceptron as described in 6.2. The algorithm was applied to the data using a pipeline, which also conducted the scaling of the features. The pipeline was used as an argument in a grid search (`GridSearchCV` from sklearn), which optimized the number and sizes of the hidden layers and the number of iterations. We specified twelve different possible combinations: four combinations of the number and sizes of the layers and three different number of iterations. The combinations of number and sizes of the layers were: three layers with eight nodes, three layers with ten nodes, three layers with twelve nodes and four layers with twelve nodes. The number of iterations were: 1,000, 2,000 and 3,000. The gridsearch was done only using the development data, and the number of folds in the cross-validation was set to five. The remaining hyperparameters were set to the default values. For the regression model, we chose to optimize the mean squared error, while the accuracy was optimized for the classification model. After training the models and tuning the hyperparameters, the predictive performance of the model was evaluated on the test data. We used the function `MLPRegressor` for the regression and `MLPClassifier` for the classification - both imported from the Python sklearn package.

8 Results

8.1 Regression model

For the regression model the optimal combination of hyperparameters suggested by the grid search was three hidden layers with eight nodes in each layer and 2,000 iterations. This combination gave an average mean squared error of 1,226,608,349,624.76 from the cross-validation. When using the model to predict the test data, the obtained mean squared error was 772,246,678,090.09. Since the error from the prediction of the test data was smaller than the error from the development data, it does not seem like the model overfitted the training data. The mean absolute error from the prediction on the test data was 522,135.02 DKK. Considering that the mean price of the apartments in the original data set was 3.4 mio. DKK, this seems like quite a big deviation. Figure 6 depicts a scatter plot of actual sales prices and predicted sales prices from the final regression model made on the test data. The main part of the predictions is centered around the 45-degree line, i.e. they are very close to the actual sales price. What seems to cause deviation is that the model has some troubles predicting the relatively high sales prices. More specifically, the model has a tendency to underpredict the sales prices for the most expensive apartments.

Figure 6: Actual sales price vs. predicted sales price



8.2 Classification model

For the classification model, the optimal outcome suggested by the grid search was three hidden layers with eight neurons in each layer and 1,000 iterations. Using this resulted in an average accuracy of 0.94 from the cross validation. The accuracy is calculated as the number of true predictions divided by the total number of predicted observations:

$$\text{Accuracy} = \frac{\textit{number of true predictions}}{\textit{total number of observations}}$$

Since using this number of hidden layers and iterations yielded the highest accuracy, this model was used to evaluate the predictive performance with the test data. Calculating the accuracy for the final predictions conducted with the test data gives:

$$\text{Accuracy} = \frac{2359}{2443} = 0.97$$

The accuracy calculated for the model based on the test data is higher than the accuracy calculated from the model based on the development data. This indicates that the model is not overfitting.

Confusion matrix

A confusion matrix is a table that visualizes the performance of an algorithm. Each row in the matrix represents the actual class while the columns represent the predicted class. The matrix was used to show how the classification model in 8.2 performed. Using the test data of 2,443 observations and the model with three hidden layers and 1,000 iterations, the resulting confusion matrix looks like this:

Table 3: Confusion Matrix for the test data

		Classifier Predictions										
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Actual Value	**0**	27	1	0	0	0	0	0	0	0	0	0
	1	6	495	6	0	0	0	0	0	0	0	0
	2	0	12	699	3	0	0	0	0	0	0	0
	3	0	0	6	504	5	0	0	0	0	0	0
	4	0	0	0	6	298	4	0	0	0	0	0
	5	0	0	0	0	6	157	1	0	0	0	0
	6	0	0	0	0	0	1	83	4	0	0	0
	7	0	0	0	0	0	0	0	46	0	0	1
	8	0	0	0	0	0	0	0	2	13	8	4
	9	0	0	0	0	0	0	0	0	4	8	2
	10	0	0	0	0	0	0	0	0	0	2	29

Sales prices that were correctly predicted are allocated on the diagonal of the matrix and the prediction errors are located in the cells outside the diagonal. For example, we see that six observations were predicted in the interval *below 1 million* even though they truly belonged to the interval *between 1 and 2 millions*. There are different measures that can be used in order to evaluate the performance of the classification model. These are described below.

Precision

The precision ratio is the number of correctly predicted observations in a specific category divided by the total number of observation predicted to be in that category. If the number of observations in the category, which is predicted incorrectly, is high, then the precision is low.

$$\text{Precision} = \frac{\textit{number of true predictions in category } z}{\textit{number of observation with true value in category } z}$$

The lowest precision in our classification model is for the 9th price interval i.e. sales prices *between 9 and 10 millions*. In this interval sales prices for 8 apartments were correctly predicted and 10 sales were predicted to either be in the 8th or 10th price interval. The average precision in our classification model is 0.97, which is the weighted average of the precisions for all price intervals:

$$\text{Total avg. precision} = \frac{0.82 \cdot 28 + 0.97 \cdot 507 + \cdots + 0.81 \cdot 31}{2443} = \frac{2359}{2443} = 0.97$$

A precision of 0.97 means that 97 percent of the predicted values in a price interval are predicted correctly.

Recall/Sensitivity

The *recall ratio*, also known as the *sensitivity*, is the number of correctly predicted observations in a certain category divided by the total number of observations, where the true target value is in that category. The recall is calculated as:

$$\text{Recall} = \frac{\textit{correctly predicted observations in category } z}{\textit{total number of observations with the true target value in category } z}$$

The average recall in our model is 0.97. This means that the model on average is able to predict the specific class of an apartment for 97 percent of the observations:

$$\text{Recall} = \frac{0.96 \cdot 33 + 0.98 \cdot 508 + \cdots + 0.94 \cdot 36}{2443} = \frac{2362}{2443} = 0.97$$

From the confusion matrix in table 3 we can see that 15 sold apartments in the 8th price interval are predicted to the in the 7th, 9th or 10th price interval. This results in a low recall ratio of only 0.48. Likewise, the 9th price interval has a relatively low recall ratio, but overall the model predicts most of the apartments in the correct price interval.

F1-score

The F1-score is the harmonic mean of precision and recall. This is calculated as two times the product of the precision and the recall divided by the sum of the precision and the recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2TP}{2TP + FP + FN}$$

From Table 4 we see the lowest F1-score in the 9th price interval, which is of of course because the recall and precision is lowest at the 9th price interval. Using the average precision and the average recall the F1-score can be found as

$$F_1 = 2 \cdot \frac{0.97 \cdot 0.97}{0.97 + 0.97} = 0.97$$

Table 4: Classification report

	Precision	**Recall**	**F1-score**	**Support**
0	0.82	0.96	0.89	28
1	0.97	0.98	0.98	507
2	0.98	0.98	0.98	714
3	0.98	0.98	0.98	515
4	0.96	0.97	0.97	308
5	0.97	0.96	0.96	164
6	0.99	0.94	0.97	88
7	0.88	0.98	0.93	47
8	0.76	0.48	0.59	27
9	0.44	0.57	0.50	14
10	0.81	0.94	0.87	31
Average total	0.97	0.97	0.97	2443

9 Discussion

Even though our model has proven to perform well in predicting house prices, there are some aspects that could be further improved.

First, we only have the building year of a given apartment. Since, on average, the apartments sold in our time period are relatively old (average building year is 1937), it possibly would have made a difference for our model, if we had included an indicator on the state of renovation of the

apartment, i.e. whether or not the apartment had been renovated and in that case when that happened.
Second, because Copenhagen is a big city with a lot of different "hot spots", we thought it wold be misleading to just choose one "hot spot" (e.g. the town square) and use the distances for every apartment to the given "hot spot" as a feature. Therefore, the model could be improved by defining such "hot spots" for every area of Copenhagen. Examples would include, among others, Sankt Hans Torv in Copenhagen N and Trianglen in Copenhagen Ø. For that purpose, one could use an API from hvorlangterder.dk, which has the feature of measuring the distances between different addresses. Unfortunately, this was beyond the scope of the project since agreeing on such "hot spots" for every district of Copenhagen was too time consuming.

For the hvorlangterder.dk API we had to choose which kind of route the API should use for calculating the distances. We decided to use a bike-route because it is very common for people in Copenhagen to ride their bikes, but we could also have used a car-route which might be more meaningful for some features - e.g. the junction distances. We do not think that choosing a different mode of transport would change the results, however, it would be interesting to explore this further.
When cleaning the data we deleted some outliers from the data set. Those observations were obvious errors caused by Boliga. We thus decided to drop these observations in order to get a better model for sales price prediction. Among these observations was a large villa with a public value of $5,000,000$ DKK, but the sales price according to Boliga was only $50,000$ DKK. There are other, similar cases which we subsequently removed from the data set by deleting all observations with a sales price below $100,000$ DKK. We have not made an outlier analysis to detect all outliers, meaning that presence of other outliers is possible. We only removed observations with obviously erroneous prices, hence we cannot exclude the presence of extreme observations. As pointed out in Section 7.1, in light of the comparably small number of features (compared to the $7,329$ observations) we did not see the need to perform any further feature selection before applying the neural network. This is because we pre-selected those variables from the hvorlangteder.dk API which we considered relevant, and we didn't import all distance variables the API offers.

As seen in Section 8.1, the model does not perform well when predicting the most expensive apartments. A reason for this might be, that there are only few very expensive apartments in the data set compared to cheap and middle priced apartments - i.e., the data set consists of $3,746$ observations with a sales price below or at 3 million DKK, $3,226$ observations with a sales price between 3 and 7 million DKK and only 357 observations with a sales price above 7 million DKK. Possibly, this has a great impact on the mean squared error, as a lot of very expensive apartments that are underfitted would lead to an increase in the mean squared error.
To improve precision for the expensive apartments and lower the mean squared error, more data on very expensive apartments would be necessary. This could be done by including more years, since Boliga offers data on sales starting in 1992. In order to avoid having to control for time effects, we refrained from using all available years. Future analysis could thus improve the model by considering a larger time period.
Another approach could be to focus only on the most common apartments - namely apartments with sales prices between 1 million DKK and 7 million DKK. This way we would be able to train the model to predict these kinds of apartments even better, but then the model would not be

suited for more expensive apartments.

If the project should be extended, it would also be interesting to look at other dwelling types in Copenhagen and in the rest of Denmark. Then, it would be possible to investigate the predictors for sales prices and compare the predictors across cities. We suspect that distance to junction, schools, etc. is likely to be of greater importance when buying a house outside Copenhagen.

When using a neural network, there are many hyperparameters, which can be optimized to achieve the model with the best predictive performance. Due to the limited time available, we chose only to optimize the number and sizes of the hidden layers and the number of iterations. Furthermore we decided to focus on relatively few combinations (four hidden layer combinations and three numbers of iterations), and we only used five-fold cross-validation. If more research was to be done, we recommend to extend this part of the analysis, i.e. to tune more hyperparameters, use a sample from a larger interval and employ more folds in the cross-validation.
Finally, it might be worthwhile to use various algorithms and compare the performances. The existing literature (compare Section 2) is somewhat conflicted when it comes to which algorithm is best suited for house price predictions. Comparing the results using a neural network to other approaches would hence be a promising extension to this project.

10 Conclusion

The aim of this project was to build two neural networks in order to predict the sales prices of apartments in the period January 1, 2017 to August 24, 2018. In order to do that, data from www.Boliga.dk on sales information was scraped and complemented with distance data from the API hvorlangterder.dk. Using this novel dataset, two neural networks were built – a regression model and a classification model. Employing grid search, we explored the optimal number of hidden layers and iterations for the models. That is, using three hidden layers of eight neurons in both models and 2,000 and 1,000 iterations in the regression model and the classification model, respectively.

After training the models, the regression model on average yielded an absolute prediction error of 522,135 DKK. The classification model was able to correctly predict the price intervals of 97 percent of the apartments which exceeded our expectations. Even though the model performed very well, there are some aspects of the project that could have been done differently or which could have been added. It would be worthwhile for further research to take these aspects into account.

References

Ahn, Jae John; Byun, Hyun Woo; Oh, Kyong Joo and Kim, Tae Yoon (2012): Using ridge regression with genetic algorithm to enhance real estate appraisal forecasting. Expert Systems with Applications 39, p. 8369 - 8379.

Bin, Okmyung (2004): A prediction comparison of housing sales prices by parametric versus semi-parametric regressions. Journal of Housing Economics 13, p. 68-84.

Bergmann, U. Michael; Silleman, Bjørn Tangaa and Sørensen, Peter Birch (2015): House Prices in Denmark and Sweden. In: Andersen, Torben M.; Bergmann, Michael and Jensen, Svend E. Hougaard (2015): Reform Capacity and Macroeconomic Performance in the Nordic Countries. Oxford Scholarship Online. Available at: http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780198717102.001.0001/acprof-9780198717102-chapter-12 [Accessed August 28, 2018]

Boliga (2018): https://www.boliga.dk/ [Accessed August 24, 2018].

Chiarandini, Marco (n.d.): Machine Learning and Neural Networks. Slides for the course "DM543 Introduction to Computer Science", Department of Mathematics and Computer Science, University of Southern Denmark. Availabe at: https://imada.sdu.dk/~rolf/Edu/DM534/E16/DM534-marco.pdf. [Accessed August 30, 2018]

Chiarazzo, Vincenza; Caggiani, Leonardo; Marinelli, Mario and Ottomanelli, Michele (2014): A Neural Network based model for real estate price estimation considering environmental quality of property location. Transportation Research Procedia 3, p. 810 - 817.

Hvor langt er der? (2018): http://hvorlangterder.dk/ [Accessed August 24, 2018].

Jirong, Gu; Mingcang Zhu and Liuguangyan, Jiang (2011): House price forecasting based on genetic algorithm and support vector machine. Expert Systems with Applications 38, p. 3383-3386.

Kauko, Tom; Hooimeijer, Pieter and Hakfoort, Jacco (2002): Capturing Housing Market Segmentation: An Alternative Approach based on Neural Network Modelling. Housing Studies 17(6), p. 875-894.

Park, Byeonghwa and Bae, Jae Kwon (2015): Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data. Expert Systems with Applications 42, p. 2928-2934.

Plakandaras, Vasilios; Gupta, Rangan; Gogas, Periklis and Papadimitrou, Theophilus (2015): Forecasting the U.S. real house price index. Economic Modelling 45, p. 259-267.

Raschka, Sebastian and Mirjalili, Vahid (2017): Python Machine Learning. Machine Learning and Deep Learning with Python, skikit-learn, and TensorFlow. Second Edition. Birmingham: Packt Publishing.

Sampathkumar, V.; Santhi, M. Helen and Vanjinathan, J. (2012): Forecasting the land price using statistical and neural network software. Procedia Computer Science 57, p. 112-121.

Selim, Hasan (2009): Determinants of house prices in Turkey: Hedonic regression versus artificial neural network. Expert Systems with Applications 36, p. 2843-2852.

Statistics Denmark (2018): EJEN5: Price index for sales of property (2006=100) by category of real property (quarter). Available at: http://www.statbank.dk/statbank5a/default.asp?w=1536

[Accessed August 27, 2018]

Wilson, I.D.; Paris, S.D.; Ware, J.A. and Jenkins, D.H. (2002): Residential property price time series forecasting with neural networks. Knowledge Based Systems 15, p. 335-341.