Combining Machine Learning and Forecasting to Predict Real Estate Prices

An experimental study of machine learning and forecasting methods applied to Danish real estate sales data

MSc. In Business Administration and Information Systems

Master's Thesis

Jakob Pallesen
Julius Varlid Bech
Hand-In Date 15th of January 2018
Supervisor Raghava Rao Mukkamala
Number of Pages 84 pages / 109,884 characters
Abstract

This master's thesis is concerned with machine learning and forecasting for price prediction in the Danish real estate market. The primary aim of this thesis is to investigate how machine learning can be combined with forecasting methods in order to make more accurate real estate price predictions. The secondary aims of this thesis are to investigate how big data as a concept can assist when combining machine learning with forecasting as well as investigate what challenges needs to be addressed in order to gain the full potential of machine learning in the real estate market.

Operating within the concepts of big data, machine learning and forecasting, we have collected publicly available data in order to conduct an experiment where we build house price predicting model and compare their accuracy. We have used a price index model as our forecasting model, and combined it with our machine learning models by extrapolating previous sales data. The machine learning algorithms used are boosted decision tree regression and decision forest regression. Our results show that combining our forecasting model with any of our machine learning models did not yield a higher accuracy in real estate price prediction. Our machine learning model, using an iteratively tuned boosted decision tree regression showed the highest model fit without being combined with the data extrapolated by the forecasting model. As recommendations, we present how machine learning can be used to predict the price of a house and what challenges to be aware of.

We conclude that the combination of the two models used does not yield higher accuracy when combined by extrapolating historical sales data. Further, we conclude that models where one is far superior in prediction accuracy than the other is not combined. However, we will recommend what can be done next, in order to overcome these challenges and create combined predictive models with a higher accuracy.
Table of Contents

1.0 Introduction .................................................................................................................. 5

1.1 Research Relevance ..................................................................................................... 5

1.2 Research Motivation .................................................................................................... 6

1.3 Research Question ........................................................................................................ 6

1.4 Research Approach ...................................................................................................... 7

1.5 Structure of the Thesis ................................................................................................. 7

2.0 Research Methodology ................................................................................................. 9

3.0 Conceptual Framework ............................................................................................... 11

3.1 Big Data and Data Mining .......................................................................................... 11

3.1.1 Introduction to “Big Data” ..................................................................................... 11

3.1.2 The Four V’s ........................................................................................................... 12

3.1.3 Big Data Approach ............................................................................................... 14

3.1.4 Data Mining .......................................................................................................... 14

3.1.5 Data Mining Approach ......................................................................................... 15

3.2 Machine Learning ....................................................................................................... 17

3.2.1 Supervised Versus Unsupervised ......................................................................... 18

3.2.2 Types of Algorithms ............................................................................................. 18

3.2.3 Overfitting ............................................................................................................. 19

3.2.4 Validation .............................................................................................................. 20

3.2.5 Evaluation .............................................................................................................. 21

3.2.6 Boosted Decision Tree Regression ....................................................................... 22

3.2.7 Decision Forest Regression .................................................................................. 24

3.2.8 Computational Complexity of Learning .............................................................. 24

3.2.9 Machine Learning in Relation to Big Data .......................................................... 24

3.3 Forecasting .................................................................................................................. 25
4.0 Related Work .................................................................................................................. 26

4.1 Articles About the Danish Real Estate Market .......................................................... 26

4.2 Academic Articles where Machine Learning has been used to Predict the Price of a House ......................................................................................................................... 27

4.3 Academic Articles that Combined the use of Machine Learning or Forecasting Methods 27

4.4 Academic Articles that looks into Forecasting Models ................................................. 28

5.0 Data Analysis .................................................................................................................. 29

5.1 Business Understanding ............................................................................................... 29

5.1.1 Private People .......................................................................................................... 30

5.1.2 Realtors .................................................................................................................... 30

5.1.3 Mortgage institution ............................................................................................... 30

5.1.4 Danish Tax Authority .............................................................................................. 31

5.1.5 Organizations Maintaining Datasets ........................................................................ 31

5.2 Boliga Data Analysis .................................................................................................... 31

5.2.1 Data Understanding ................................................................................................ 31

5.2.3 Data Preparation ...................................................................................................... 38

5.2.4 Market Analysis ....................................................................................................... 42

5.3 KMD Data Understanding and Preparation ................................................................. 49

5.4 KMD Data Modelling .................................................................................................. 56

5.4.1 Baselines .................................................................................................................. 56

5.4.2 Price Index Model ................................................................................................... 58

5.4.3 Machine Learning Analysis ..................................................................................... 60

5.4.4 Boosted Decision Tree Regression Configuration .................................................. 65

5.4.5 Decision Forest Regression Configuration .............................................................. 67

5.4.6 Tuning the Algorithms ............................................................................................ 70

6.0 Results .......................................................................................................................... 75

6.1 Meaningful Facts ......................................................................................................... 78

Page 3 of 121
1.0 Introduction

Machine learning has been given a great deal of attention over the past few years. With Artificial intelligence as its super category, machine learning is being applied more and more in practice. In the digital age dominated by big data and ever improving processing technology, machine learning poses a great deal of potential in assisting to resolve some of the challenges related to pattern recognition in large and complex datasets. As organizations strive toward becoming more data driven, machine learning becomes even more popular as a tool to make better decisions in the context of business as well as government.

In the real estate market, traditional methods of forecasting have been utilized for decades to describe current conditions in the market. Machine learning as a pattern recognition tool presents an opportunity to make more accurate assessments with big data. The potential of machine learning applied with data from the real estate market is great, and poses new areas of research when combined with forecasting methods. In this thesis, we will experiment with the possibilities of using machine learning to predict the price of a house in the real estate market. We will also experiment with forecasting methods and how they might be combined with machine learning in order to extract the best of both methods.

1.1 Research Relevance

Market price prediction is not a new topic in real estate market. Valuation of a real estate property is crucial to several actors influencing and infused by the real estate market. Banks have to assess the price of a house before it is bought by a customer. The Danish tax authority ‘Skat’ has to evaluate each piece of real estate in the country in order to collect the correct amount of property tax. Real estate agents need to set the price right in order for the seller to be able to sell at the highest price possible. Finally, private people want to know how much their house or potential new house is worth in order to make a more informed decision on when to sell or buy.
Businesses and organizations as well as ‘Skat’ has an opportunity to utilize emerging
technologies such as machine learning and big data to extract value and knowledge from
data. Machine learning and big data has the potential to improve decision making within
these actors. Regarding real estate price prediction, machine learning has the potential
to aid and assist the decision making foundation for ‘Skat’, banks, real estate agents and
individuals.

These opportunities pose challenges that has to be overcome in order to make accurate
prediction of prices in the real estate market. Data used to make the price predictions
must exist and the methods used to predict the price must be tuned. These challenges
need to be addressed in order to gain the full potential of machine learning applied in the
real estate market.

1.2 Research Motivation

The topic of applying machine learning to predict prices is not new. However, despite the
increased interest in using machine learning to predict prices in a given market, there
seems to exist a gap in the literature regarding the combination of machine learning and
forecasting methods to make predictions. The perception of big data and machine
learning is that it is capable of pattern recognition to a degree yet unheard of. Even so,
there seems not to be much research into how machine learning compares to forecasting
methods in real estate price prediction.

Due to this gap, research in machine learning and forecasting applied to the real estate
market needs to be conducted, in order to understand how machine learning combined
or not combined with forecasting methods present attainable opportunities for actors in
the market and how challenges can be addressed.

1.3 Research Question

On the basis of the relevance and motivation the primary aim of this thesis is to assess
how machine learning and forecasting methods perform comparably, and whether they
can be combined to achieve greater accuracy in price predictions. Our primary research
question has been formulated as follows:
• How can machine learning and forecasting methods be used separately and combined to make real estate price predictions?

In order to answer our primary research question, we have formulated the following secondary research questions, which will be addressed throughout this thesis:

• What is machine learning how does it relate to big data?
• How can the dimensions of big data help combine machine learning and forecasting methods in order to improve real estate price predictions?
• Which challenges needs to be addressed in order to gain the full potential of combining machine learning and forecasting methods?

1.4 Research Approach

This theoretical foundation throughout this thesis has its grounding in big data and machine learning. The experimentation has been conducted using big data and machine learning methods in order to answer our research question. We have obtained data from two sources to conduct our experimentation. We have obtained historical sales data from boliga.dk and via KMD. The data was collected in order to experiment with the application of machine learning and forecasting methods. This was done with the intention to obtain information on how these different methods performed separately and combined.

1.5 Structure of the Thesis

This thesis consists of eight chapters structured as described below.

Chapter 1 - Introduction
Introduces the subject matter of the thesis - predicting real estate prices using machine learning and forecasting and a combination of the two. Further, it is outlines what has motivated this study and what the primary as well as secondary aims for this study is.

Chapter 2 - Research methodology
Presents our approach to research and what we perceive as acceptable knowledge gained through research.
Chapter 3 - Conceptual framework

Elaborates on the concepts used throughout this thesis to create a common understanding of what is meant by each concept and its components. Also, explains the methods we will be using in our analysis. Machine learning, big data and forecasting will in this chapter be presented and outlined with explanations.

Chapter 4 - Related work

In this chapter work related to the topics of machine learning, forecasting, the real estate market and machine learning model combining will be presented.

Chapter 5 - Data analysis

The data analysis presents how we gathered data and performed our descriptive analysis in order to understand the data. Further, this chapter presents how we conducted our experiments with the different models.

Chapter 6 - Results

In this chapter, we have presented and explained the results of our experimentation. This includes the results from our experimentation with machine learning models, a forecasting model, the combination and a set of baselines.

Chapter 7 - Discussion

This chapter discusses the finding of this study. After presenting the findings, we present our suggestions to what future work we believe can improve the knowledge of the topic of applying machine learning and combining models to predict real estate prices.

Chapter 8 - Conclusion

In the final chapter, we conclude how our findings answer our primary as well as secondary research aims.
2.0 Research Methodology

Positivism [30] is a philosophical approach where the researcher believes in empirical science telling the truth. Positivists believe in the objective truth of the reality and avoids religion, feelings and other hard or unmeasurable aspects to have an impact in the research. The approach is usually used within the natural phenomena. We will approach our research with a philosophy of positivism and conduct experiments [30],[71] to answer our research questions.

![Image 1 – Research Onion [30]](image)

Ontology [69],[30] is the view of the researchers on the nature of reality or being. Our ontology makes us focus on objectivity where our logic dominates the course of action as the exclusive source to knowledge when conducting our research. Epistemology [70],[30] is the researchers view regarding what constitutes acceptable knowledge. In this thesis, we will proceed with a quantitative data driven approach. We will gather data and
objectively evaluate the results of our experiments. We consider acceptable knowledge as knowledge based on properties and relations of natural phenomena. Axiology [72] is the researchers view on the role of values in a research. Our view is that our research is undertaken in a value-free way (non-beneficial), where we are independent from the data and maintains an objective stance.
3.0 Conceptual Framework

In this chapter, we will describe the concepts of big data, data mining, machine learning and forecasting as a foundation for the research conducted.

3.1 Big Data and Data Mining

3.1.1 Introduction to “Big Data”

Big data is a concept within data science that covers multiple processes from data collection to other processes such as analyzing, storing, structuring, interpreting data. Gartner's Hype Cycle, a branded graphical presentation developed by Gartner, pointed out in 2012 [46] that Big data would reach “Peak of Inflated Expectations” in 2-5 years, even though the concept was known already in the early 2000s [49]. Today, Big Data is a well heard of term, but the definition of what big data is, differs. We will use the definition by Gartner which is described and analyzed in the “International Journal of Information Management” [51] and specifically from the journal volume 35 “Beyond the hype: Big data concepts, methods, and analytics” [50], where it is defined as: “Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.” Another journal “IEEE Access” has published a paper, “Machine Learning With Big Data: Challenges and Approaches” [56], about the definition of big data. They used Gartner's definition of big data as the approach to analyze the concept. Both journals introduce the three main V's, Volume, Variety and Velocity as the dimensions describing big data, but they also introduced additional V's that other companies have tried to include as dimensions describing big data. In the definition used throughout this thesis we will include IBM's fourth dimension [56], Veracity, to describe big data and the challenges behind big data.
3.1.2 The Four V’s

Volume

Volume refers to the amount, size and scale of the data [56]. When discussing this within a machine learning perspective we talk about the number of rows and attributes, so it is both the vertical and horizontal amount. The volume of a dataset is relative. It depends on the type of data and its complexity. A dataset with few rows but very complex data with various attributes could be considered big data and a dataset with 1.000.000 rows could sometimes not be referred to as big data, if the data is not complex. Imagine 1.000.000 rows with only one boolean attribute that give a value 1 or 0.

The volume dimension introduces numerous challenges. Especially the computational complexity is challenging when it comes to volume. For machine learning, usually, the more data the better. Accordingly, to the higher volume of data, more and more compute power, storage, bandwidth and processing of the data is required. The complexity of data is increasing, it is not necessarily just numeric values in a database that require to be stored, it could also be images, videos or other items, which increases the requirements of the hardware and processing software tools.

Algorithms performance is crucial for value creation of the increasing data size. This is demanding extra resources allocated to architectural infrastructure so companies can scale up and down for both storage and compute power. This leads many companies to use cloud solutions where they can scale up and down accordingly. Furthermore, the data structure is also crucial for the algorithms to work properly. It could be partitioning and placement, data clusters or other methods. It is not only the data size that impacts the performance of the data but more the data structure being used.

Machine learning algorithms relies on the assumption that they can be run from one single machine when they are being designed, but the increasing size of data also increases the requirement of memory allocated to the disks where the algorithms are running on [27]. This challenge is referred to as the curse of modularity [26] and should be taken into consideration when designing an algorithm. If possible, algorithms should be designed to work with multiple nodes and hereby enable multiple compute units to work in parallel execution.
Variety
Variety refers to the number variations in the dataset, not only the number of attributes but also the variety of what the attributes represents. As described datasets does not necessarily consist of letters and numbers in a database but it could also consist of images, videos and other items that requires a lot more storage [28]. The higher requirement to storage can result in situations where the data need to be stored in multiple locations which introduces new challenges, since algorithms now need to select the data from multiple data sources. This is the major challenge for variety, since we know how much data we got today, but the growth of data seems exponential.

Another challenge is the data heterogeneity. It consists of two main heterogeneities, semantic- and syntactic heterogeneity [47]. When multiple parties are working on multiple datasets that at some point need to be joined, there is a challenge with meanings and interpretations inside the different datasets. This is referred to as the Semantic heterogeneity. The Syntactic challenge is based on the difference between the chosen data types, formats, encoding and similar small technicalities in the datasets. This can be a problem when trying to join the datasets and also poses challenges for the algorithm to work with different data types.

Velocity
Velocity refers to the speed, how big data is generated and the rate of the data being analyzed. Smart phones, embedded devices and other real-time sensors are requiring fast reactions from the environment to handle all their data transactions and by that fact the velocity has become an important factor for big data. The increasing volume of data and the increase of the data created every day, requires machine learning algorithms to adapt by processing and transforming data on a daily basis [48] and by that work with incremental learning [90].

Veracity
The last V, veracity, refers to the reliability of data in the dataset and the reliability in the data sources being used. The data provenance is the data mining process where tracking of all changes and methods are being used to processing the data in order to track down errors. This process poses a challenge itself, by the fact that all this metadata needs to
be stored and processed by requiring additional compute units and storage. Another challenge is the uncertainty inside the gathered dataset. An example could be data gathered from social media. This data would be subjective and may not tell the truth and end up making the findings unprecise.

### 3.1.3 Big Data Approach

According to Gartner's definition of big data the general approach of big data is as illustrated in the figure below, “Data Analytics Pipeline”. The three V’s and their challenges are hiding between every step and with the fourth V, veracity, we already have challenges before the data extraction. All this should be taken in consideration when working with big data.

![Image 2 - Data Analytics Pipeline](image-url)

### 3.1.4 Data Mining

Processing data is a known term where we as human beings always tried transforming data into useful information. It could be trying to understand various observations to something useful. There are many similarities with that understanding of data transformation and the newer term data mining, which is defined and described in the book “Data Mining - Practical Machine Learning Tools and Techniques” [52]. The book defines data mining as follows: “Data mining is defined as the process of discovering patterns in data”. The book furthermore states that the process should be automatic or at least semi-automatic. The outcome or discovered patterns must be meaningful for the reader or the user that are going to use the findings. So, data mining is a technical approach where the old data transformation is a more general approach. In this thesis,
we will use the data mining approach where we technically try to transform data in information and knowledge.

3.1.5 Data Mining Approach

Our data mining approach to transform data in this thesis is The Cross Industry Standard Process for Data Mining (CRISP-DM) [53],[54]. The model will not be used as a theoretical approach, but instead as the methodical approach of how we go through the data mining process and experimentation.

![Image 3 – CRISP-DM](53)

**Business Understanding**

The business perspective should first be understood. It is vital to understand the problem, the case and the goal, before trying to find solutions and define which patterns that are wished to be discovered. What is the background of the businesses’ that the data miner wants to mine, in order to clarify the objectives [16]? Which actors could be interested in the project and what is their risks and benefits in it? The business requirements should be determined, the goal for the project should be clarified with success criteria and the preliminary plan should be converted into a data mining perspective and problem definition [55].
Data Understanding
The data miner now needs to investigate what data is achievable to collect that somehow could describe patterns and fulfill the businesses’ demands. The volume and variety of the data [56], [50] being gathered and its challenges as describe in the big data section should be taken into consideration. The data must be described, explored and at least the quality of the data must be verified, before moving to next step - Data Preparation.

Data Preparation
The data miner should now select the required data, integrate it and merge it with all required sources so it is ready for modelling. During the whole data preparation the miner should be aware of the veracity challenge [56], [51] as describe in the big data section. The mining process requires the miner to track metadata in order to track down errors if there are any errors at a later point. This process should secure dataset or datasets that are ready to be modelled. This process will most likely run through multiple iterations with the modeling since the algorithms accommodate data structure requirements.

Modeling
Models should now be selected and created so they can discover a or multiple patterns within the business understanding and its requirements. Initially this might be based on assumptions of which models that could discover patterns within the data, but due to the agile process the models would likely be revised and remade. Nonetheless, the models would be build and attributes will be chosen. The model’s setup and parameter settings should be documented so further changes easily can be implemented.

Evaluation
The models should then be evaluated whether they achieve and fulfill discovering patterns within the data. Else, the further iterations will be considered. Another aspect is to determine whether the findings are truth worthy. Are we confident that the data we gathered and the model we made are valid and reliable, if not, we must go back to step one and restart from there with the new adjustments. In other words, the veracity should be taken into consideration. There are many ways to evaluate the models that have been made, but due to the velocity dimension, it has to be done in a dynamic and fast matter that can handle incremental train data and discover patterns within a valuable time [59].
Deployment
If we are satisfied, we can proceed to deployment. The data miner can now put the models into use to achieve value for the investment. The models should continuously be monitored and maintained.

3.2 Machine Learning

Machine learning is by Provost and Fawcett [53] defined as a method used to create predictive models from data. A predictive model can be explained as a rule that has the ability to predict an output from a set of inputs. As a subcategory of artificial intelligence, Provost and Fawcett [53] describes machine learning as a field which concerns itself with finding patterns in data.

Likewise, Shalev-Shwartz and Ben-David [23] describes it as automated pattern recognition and learning. By analyzing attributes of a given dataset, machine learning techniques facilitate the creation of inductive reasoning [23]. Inductive reasoning is described by Shalev-Shwartz and Ben-David [23] as the process of labeling yet unseen data. This labeling is done using a set of rules figured out by the learner [23]. We will refer to inductive reasoning as scoring.

There are different types of machine learning. The type of machine learning we will be operating within is supervised machine learning as opposed to unsupervised machine learning as described by Ayodele [82]. Supervised machine learning is defined as machine learning done with labeled data. Labeled data could for instance be a dataset containing attributes describing a set of cakes and a label for each cake indicating whether the cake tastes good or not. In the case of real estate price prediction, the label is tied to price of the houses sold and the rest of the data can be attributes describing the house itself.

Unsupervised machine learning is learning without a label set. The learner does not know specifically what prediction is “correct” and cannot correct itself according to labels as supervised learning can.
When creating a machine learning model, the input given to the learner can be divided into different types of data as Shalev-Shwartz and Ben-David [23] describes. The learner is given a training set of data, which contains a domain set of data and a label set of data. The domain set is the data describing attributes believed to be predictive of the label set. The label entries are tied to specific entries in the domain set.

The output of the learner is a prediction rule that can be used to label new data. We will refer to the output of the learner as the trained model. The success of a trained model can be measured as the error of predictions made to new data. The less the error, the better the model. This is done using testing data. The trained model is fed with yet unseen data containing the same kinds of attributes as the domain set used by the learner. The testing data contains labels for each recording of data, but the trained model does not get to see these labels.

### 3.2.1 Supervised Versus Unsupervised

Supervised machine learning as defined in the previous section is machine learning where a model train on predicting a label from a set of attributes. This means that there is a “correct” label for each data entry. By minimizing the error between the predicted label and the actual label, the learner can tune itself to achieve the best results.

Unsupervised machine learning is performed without a label to indicate the correct “prediction” for the learner. Unsupervised learning can be done as a way of categorizing data into clusters. There will not be a correct categorization for each data entry, and the number of clusters is not defined by a label dataset. We will not be using unsupervised machine learning in our data analyses, but the principal of it is important to understand, since we will be mentioning a type of algorithm unsupervised clustering can be mistaken for.

### 3.2.2 Types of Algorithms

We will be explaining the different types of algorithms in relation to supervised machine learning, where data contains an attribute set as well as a label set.
Classification
A classification algorithm is defined by Provost and Fawcett [52] as an attempt to predict which class (also known as a category) in the entire dataset a specific data entry belongs to. The label set contains the correct class for each data entry. A class will be represented as a category. The classification learner will create a rule as output that tries to predict the class of each data entry with as few errors as possible.

Regression
A regression algorithm is defined by Provost and Fawcett [52] as “value estimation” or a learner that creates a rule in order to predict a numeric value. In the case of the real estate market and house price prediction a regression algorithm would be used to predict the price of a house with an error as small as possible.

3.2.3 Overfitting
When creating a model that tries to predict patterns in a given dataset, it is important to ensure the validity of the evaluation of the model. By that we mean to ensure the consistency in error minimization the model exhibits when exposed to new data. A model can be “too perfect” and only deliver good results with the data it was trained on. Shalev-Shwartz and Ben-David [23] defines it as “when a hypothesis (the rule the learner output creates) fits the training data too well”. This means that a model that has not been tested with data outside of the dataset provided for training may suffer from overfitting. Shalev-Shwartz and Ben-David [23] illustrates this very well with an example consisting of a dataset with 10 entries (See the image below).

![Image 4 – Polynomial Regressions [23]](image-url)
Using a model based on a second degree polynomial function (degree 2) the data points do not match up with the regression. Using a model based on a third degree polynomial function the data points get closer to the regression made, and finally using a ten degree polynomial function the data points match the regression perfectly. Now imagine we let the model with the tenth degree polynomial calculate the prediction of a new data entry given. In the image below we have indicated the new data entry with the predicted label as the blue point. In the process of calculating this prediction, the correct label has been hidden from the trained model. Visualized below we have added the correct data point including the label in red and the calculation done by the trained model in blue.

![Image 5 – 10th Degree Polynomial Regression](image)

Since we have used a regression method that suited the first 10 data points perfectly, but not checked to see if new data points would be calculated with an acceptable error, the trained model is overfitting and may have extreme errors when introduced to new data. The predicted blue data point has a visibly large error displayed as the distance to the red data point.

### 3.2.4 Validation

In order to measure the error and validate a trained model, it must be tested. In order to minimize overfitting as much as possible, it is possible to hold out a set of data during training [23]. By holding out a set of data containing attributes and labels, the trained model can be evaluated in relation to, for the trained mode, yet unseen data. By holding out e.g. 20% of the data given, the model can train with the remaining 80% and test the accuracy and precision with the 20% we hold out. This is called splitting the data into a
test set and train set. The split has to be done randomly throughout the data to ensure a good assessment of the trained model.

Another way to use a hold out set is to gather new data after the fact, which the trained model has not seen.

To ensure that the data has not randomly been split in favor of the trained model you can do what is called k-Fold Cross Validation [23]. We will refer to this as cross validation. This is done by splitting the dataset into a given number of sets. E.g. you can split it into 5 sets. By training the model with set 1, 2, 3 and 4 as one set, the trained model is then tested with the 5th set. After training and testing this combination of data, the model is retrained with a combination of four other sets and tested against a different 5th set. This is done till all (in this case five) tests have been calculated. The average of all the tests is then used to indicate how good the model is [23].

3.2.5 Evaluation

By testing the trained model and marking a testset of data with predictions, it is possible to calculate the average model prediction error of the trained model as well as how well the predictions fit the correct values (labels) overall.

The mean absolute error (MAE) is the mean absolute difference between the predicted label and the actual label. The MAE indicates the average error for the trained model. This type of error does not indicate how widespread the errors are. The mean absolute error is calculated as [78]:

\[
MAE = \frac{\sum |Predicted Label - Actual Label|}{n}
\]

Where \(n\) is the number of data entries to be predicted.

The root mean squared error (RMSE) like the MAE is also a measure of error between the predicted label and the actual label. The difference between these two way of measuring the error is that the RMSE takes into consideration the variation in error. This means that if there is a big variation in the error from prediction to prediction, the RMSE gets bigger. This has the effect that RMSE in equal to or (as in most cases) bigger than MAE. The root mean squared error can be calculated as [78]:

\[
RMSE = \sqrt{\frac{\sum (Predicted Label - Actual Label)^2}{n}}
\]
RMSE = \sqrt{\frac{\sum (\text{Predicted Label} - \text{Actual Label})^2}{n}}

Where \( n \) is the number of data entries to be predicted.

The coefficient of determination or r-squared can be used as a measure to explain how well a model performs when introduced to a test dataset. Field [80] defines r-squared as a measure that provides information about the correlation between attributes. Likewise the Academic Skills Kit, Newcastle University [81] describes it as the correlation between the prediction made and the actual value. The coefficient of determination is measured between 0-1 where 1 means the trained model has a perfect fit, and 0 means the trained model is completely random. R-squared is calculated as:

\[ R^2 = 1 - \frac{\sum (\text{Actual Label} - \text{Predicted Label})^2}{\sum (\text{Actual Label} - \text{Mean (Actual Label)})^2} \]

3.2.6 Boosted Decision Tree Regression

Alpaydin [77] describes a decision tree as a hierarchical model for supervised machine learning where the prediction of the label is done by creating a set of binary rules also referred to as decisions. These decisions are called branches. In this way, a decision tree split the data into group more and more fine grained until a leaf node is reached. A leaf node is a final node indicating the prediction of the decision tree. In case of regression the leaf node predicts a numeric value based on the training data contained within the specific grouping indicated by the attributes of the data. This is a simple dataset with a decision tree:
At the root of the tree (in the top) the first rule dictates a split where attribute $x_1$ larger than $w_{10}$. The rule can be understood as a divider based on whether a statement describing the attributes in the data is true or false. At the end of the last branch is a leaf node marked with a C. Each leaf predicts the label of the data. In case of regression this is a numeric value.

Schapire [76] in 1990 introduced a boosting algorithm that uses several iterations of a simple machine learning algorithm in order to strengthen the prediction accuracy in the trained model. Boosting is defined by Alpaydin [77] as using several trained models combined. The combination is done by initially training a model and scoring it according to the label data. The incorrect predictions are used as data for the next trained model. This is done at least three times and the final result is a rule based on a combination of several trained models.

A boosted decision tree regression is by these definitions a combined set of decision trees that predict numeric label data.
3.2.7 Decision Forest Regression

A decision forest regression like boosting uses a combination of several trained models to make a better prediction [23]. The difference between the decision forest regression and the boosted decision tree regression is how the data is used and/or split between the different trees training to predict the numeric value. In the case of the decision forest regression the same data can be used by all the models trained and the final prediction is determined based on the average prediction of all the trees. The data can also be split into the tree and the final prediction is as well determined based on the average prediction of all the trees.

3.2.8 Computational Complexity of Learning

Machine learning algorithms can be tuned in such a way that the computational complexity exceeds the practicality of training a model and using the output rule in the real world. At a certain point of tuning the return of goodness of model fit diminishes compared to the computational complexity of learning that the current data or method has been exhausted. It is therefore important not only to finetune the machine learning model in order to get great results, but also to make sure other aspects of a correct prediction is met. These are aspects such as data quality, data size and data variety amongst other aspects.

3.2.9 Machine Learning in Relation to Big Data

Machine learning is the tool that can handle big data. Traditional analytics tools are not strong enough to handle and capture the full value of big data [91]. Machine learning can be used to exploit opportunities and recognize the hidden patterns inside big data, which could be beneficial for businesses when it comes to data driven decision making. Machine learning can learn and manage the relationship between multiple data sources. The scalability in big data empowers the use of machine learning [92]. The numerous different machine learning models can when modelled, often in real time, give gleaning insights when businesses are using their large amounts of data [50].
3.3 Forecasting

Forecasting is defined by Clements and Hendry [74] as a statement about the future. By this definition forecasting is the act of trying to predict events in the future. In order to understand how we will be using forecasting in this thesis, it is important to define the term now-casting derived from forecasting.

Bańbura et al [75] describes now-casting as “... a contraction for now and forecasting ...” and defines it as “... the prediction of the present, the very near future and the very recent past.”

We will be using now-casting methods as a subcategory of forecasting to create a predictive model as they share methods of prediction.

When mentioning forecasting of house prices in this thesis, we define it as predicting the price of a house when sold in the present or very near future. It is therefore similar to Bańbura et al’s [73] definition of now-casting. We will use the term forecasting throughout this thesis with this definition.

A method of forecasting and now-casting of prices in a given market is by extrapolation using a price index. The type of price index we will be using is similar to the one used by Bailey et al [73] in order to predict what a given house should cost. The price index works by using the average market trend to extrapolate a previous sales price to current time. The index can be described as:

\[
\frac{\text{Previous sales price}}{\text{Previous market index}} = \frac{\text{Current sales price}}{\text{Current market index}}
\]

In this way, we can use the previous sales price of a house and the previous market index to predict the current sales price with the current market index.
4.0 Related Work

4.1 Articles About the Danish Real Estate Market

Skak [13], a lector from Syddansk University, analyzed the financial crisis in 2008 and argued for banks should have warned their customers already in 2005 when the price evolution were more than two percent points higher than the consumer price index. He indicates that the overall evolution of the real estate market in Denmark follows an average two percent point above the consumer price index growth.

Erlandsen, Lundsgaard and Huefner [16] investigated how regulations in the danish real estate market impacted the evolution of prices on the market. The paper looks into the diversity between the flexible labour market and the highly regulated house rental market that hinders the mobility. The paper’s indicates that an increase in the property tax for owner-occupied housing would make the price evolution more natural on the market. At least it recommends to set the rental housing to be more liberalized by making it more open and flexible.

Arbejdernes Landsbank [6] created a report where they dived into the difference in the market development of danish real estate market. The report illustrated the development for houses and apartments in different cities and regions. The report indicated that the danish market can not be identified as one market but needs to be divided in multiple parts. There is a difference in the growth between apartments and houses, furthermore a difference between larger cities and smaller cities and at least a difference between regions are seen.

The Danish tax ministerium “Skatteministeriet” [12] revealed a new tax law in may 2017 for tax on owner-occupied housing that would come into force from 2021. The purpose of the new law is to make the tax distribution more fair so it follows the market evolution where prices are increasing more in the larger cities than the smaller cities. The new model will include actual trade prices in the valuation of houses, where the previous model only used data about the house, like number of square meter inside the house and the
build year. The new valuation method would now follow the supply and demand on the market[83].

4.2 Academic Articles where Machine Learning has been used to Predict the Price of a House

In a previous project study [20] we predicted the price of a house using a K nearest neighbor (k-NN) regression model. The price was based on the average square meter price of three similar houses, where it chose the similar houses by size, number of rooms and the distance from the house, that you were trying to predict price. The paper indicated that the model required additional attributes, especially ground area, before it would work properly and furthermore using machine learning to weight the attributes that chooses the similar houses. They achieved the dataset by scraping boliga homepage, where they used a library called Beautiful Soup and its methods to scrape every single page and changing page by changing the url. Our previous paper used a market index model to extrapolate previous sales data to predict the price of a house today.

Chiarazzo, Caggiani, Marinelli and Ottomanelli [7] made an Artificial Neural Network (ANN) model to predict the sales price of a house in Taranto (Italy). Their model considered the environmental quality of the property's location by including environmental attributes into the dataset. The paper highlighted the importance of location, since they discovered, that the evolution of the market was different depending on the location, when predicting the price.

4.3 Academic Articles that Combined the use of Machine Learning or Forecasting Methods

Feio, Viana-Ferreira and Costa [21] tested the potential of combining machine learning methods in a bioassessment tool as a prediction tool. Using Support Vector Machines (SVM), Multi-Layer Perceptron and k-Nearest Neighbour (kNN) they created a combined model for prediction, which performance was better than any single method alone.
Fernández-Varelaa, Hernández-Pereiraa, Álvarez-Estévez and Moret-Bonillooa [22] combined machine learning models for the automatic detection of EEG arousals. Testing multiple combinations of machine learning models such as Fisher’s Linear Discriminant, Support Vector Machines, Artificial Neural Networks, Classification Trees, k-Nearest Neighbors and Naive Bayes, they tested and found combined models serving as better predictors than any of the standalone models in the specific combination.

4.4 Academic Articles that looks into Forecasting Models

Xu [5] predicted prices of houses by using a repeat sales index model. The prices were multiplied with a factor equal to the area’s evolution and previous sales of the house. The paper highlighted five bias, renovation bias, hedonic bias, trading frequency bias, sample-selection bias and aggregation bias. At least the model couldn’t predict the price of houses that has not been sold before.

Wang, Wen, Zhang and Wang [3] forecasted the real estates prices using three different models. A combined Particle Swarm Optimization (PSO) and Support Vector Machine (SVM) model, a SVM model and a back propagation (BP) neural network model. They tested the models in Chongqing, a city in China, and discovered that the combined PSO and SVM model had the best forecasting accuracy.

Komtesse, Buhl-Andersen, Nilsson, Rebil, Mukkamala, Hussain and Vatrapu [18] investigated the relationship between social media engagement and financial performance for H&M by trying to forecast the sales based on big social data. They estimated values that determined how much of recent observations that should be used to forecast the sales and how much the previous sales should be used. The paper indicated a correlation between social buzz and sales, so by understanding the social buzz it would be easier to forecast sales. When forecasting the price of a house it is important to include other data than just previous sales.
5.0 Data Analysis

When a house is listed for sale, the price is advertised in two different ways. It is advertised as the full price (in kroner) and the average square meter price (in kroner per square meter). This gives us two possible labels of prediction for a house. The first possibility is to predict the price of a house based on the actual price. The second possibility is to predict the square meter price of a house. The full price of the house can then be calculated by multiplying the square meter price by square meter count of the house. We have chosen to predict the square meter price for houses. Any experiment we have made could be done with the full price as well.

5.1 Business Understanding

The goal for this project is to investigate how machine learning and forecasting can be used and combined as methods to predict the price of a house today. If we succeed in predicting the price of a house by using existing data, we can see different actors that can benefit from this scenario. In the CRISP-DM process this is part of the “Business Understanding”
We will go through the different actors that we find interesting when creating a model to predict the price of a house.

### 5.1.1 Private People

A possibility is to make a website similar to the already existing bolighed.dk [84], where you can enter an address and then the homepage will output a price as if the house is to be sold today. It will be easier for people who want to sell their houses, to look up the price, than get in contact to realtors, schedule an appointment, show the house and then the realtor can calculate the expected sales price. Buyers of houses will achieve more transparency when buying a new house since the calculations will be objectively calculated by a system. The people, who want to see how much their house is worth, but not necessarily wants to sell it, will now have a quick method to check the price. Realtors will no longer be the ones who has all the knowledge of the price of a house. This will leave it more decentralized and transparent for sellers and buyers.

### 5.1.2 Realtors

Even though you can argue that realtors may lose their power on the real estate market, since they will not be the ones in possession of all the tacit knowledge anymore, it can be a question of adaptation [85][86]. Realtors can compete by making their own algorithms to predict the price of a house or other ways of adaptation. Nonetheless, they could get a tool to predict the price of a house without they need to spent time investigating it.

### 5.1.3 Mortgage institution

In Denmark, you can loan up to 80 % of the price of a house at a mortgage institution [87]. You can do this because the institutions use your house as a mortgage for their investment. Mortgage institutions are therefore interested in the real value of the house a buyer is considering buying, since they are exposed to the risk where owners cannot pay. A model where the mortgage institutions only are required to know the address of a house to consider the price of a house, can be an easy, fast and inexpensive approach, compared to physically check a house every time.
5.1.4 Danish Tax Authority

Today, the Danish tax authority is collecting a property tax based on the value of a property. They are currently aiming on implementing a new evaluation system in 2019 [83], that would come into force 2021 [12]. If we can get good result with our model, meaning we are close to predict the actual price of a house, then the Danish tax authority may consider using our model or parts of it.

5.1.5 Organizations Maintaining Datasets

If we succeed in creating a model that can predict the price of a house, then our model is depended on the actors that are feeding the datasets with data. The models should run on a daily basis in order to keep our price predictions updated with the latest available sales records. The models are maintained by the organizations that are inserting data into the datasets whenever citizens buying and selling houses and new properties.

Price prediction models already exist and one of them has been made by Bolighed.dk [84]. Below is their evaluation of their price prediction model. It states that they can estimate the price within 5% of the actual price for 46.6% of real estate, which is far from perfect. Therefore, we believe that there are still business opportunities within the area of predicting prices of houses by using public available data only.

![Image 8 - Precision of price predictions for Bolighed [84].](image)

5.2 Boliga Data Analysis

5.2.1 Data Understanding

After investigating the business and achieving an understanding, we can now focus on acquiring a dataset that can fulfill the business needs.
In our previous study [20], we learned it was possible to acquire a dataset that included a lot of sales data, by using a method called Beautiful Soup in Python. The dataset that was acquired contained the following attributes: address, zip code, price, date, type of sale, square meter price, number of rooms, house type, number of square meters and build year.

The script methodically ran through all the pages from the URL:

http://www.boliga.dk/salg/resultater?&minsaledate=2010&p=1

The last number in the URL represents the page number. Initially we wanted to follow this procedure and gather the same attributes and afterwards find another dataset with other attributes to join. We discovered that the homepage had been changed and that the script used in the previous study did not work anymore. The change actually meant that we either had to find another way to use the Beautiful Soup method on the homepage or make a new script using another method. We investigated the Boliga homepage and discovered that it was possible to make a search where we excluded all criteria when searching on the homepage so it did show all sales ever made.
The new list we got from searching also included new attributes that we were not able to acquire with the previous scraping script. In the previous study, it was concluded that the algorithm required additional attributes such as square meter area. Luckily, the new search on Boliga included this square meter area attribute. Unfortunately, the attribute “Type of Sale” was not present. This attribute determined whether it was a family sale or a normal sale and is an important attribute since the previous study concluded that family sales often are lower priced than the market value. This missing attribute could intertwine with the data foundation. Nevertheless, we decided to scrape this new “search result” and then we just needed a way to scrape it.

The URL is set up different compared to the script used in the previous study. It was not possible to change the page using the URL, and we had to find a different way. We investigated different scraping methods and found that the best way to scrape this data was by using the coding language python and the library called Selenium [67] and belonging methods together with a Chrome Driver [66]. Using Selenium as our main scraping method meant that we used a bunch of different selectors to pick the required data. The script is commented and shown in appendix 1.

The picture below shows how we inspected an element [Image 10] so we could find and use the necessary selector to get the data in our script. We found that “Værelser” row was called “room” so we used the below selector to find it [Image 11].

“rooms = results[i].find_element_by_class_name('room').text”
Then we followed the same procedure for the rest of the attributes [Code 1].
We discovered that it was possible to use a JavaScript to change the page [Code 2]. In the console when you are in the webpage, you can write GoToPage(150), which will bring you to page 150. We knew it was possible to execute JavaScript in Python so this solved the problem with changing pages. We have inserted the loop where we solved the page changing issue below.

```python
x = 100
errorcount = 0
for e in range(0, 50238):
    #print("current page: " + str(x))
    x += 50
    if x > 0:
        if e == 0:
            driver.quit()
        #chromedriver = r"C:\UB\chromedriver.exe"
        #time.sleep(5)
        #driver = webdriver.Chrome(chromedriver)
        driver.get("http://www.boliga.dk/search/find/objectid-bc279b4a-7e3d-4526-9b29-3511db62f3f4")
        time.sleep(15)
        driver.execute_script(\"window.scrollTo(0,0)\")
        driver.execute_script(\"window.scrollTo(0,0)\")
```

Our script was set to run through all the 50.238 pages mentioned in the “search” before and there were 20 sales/rows on each page, which is 50.238*20 = 1,004,760 rows. We used a chrome driver to work as a robot. It selected the rows and changed the pages until page 50.238 and row 20. When the Chrome Driver had gone through more than 100 pages it started going very slow until it crashed after approximately 500 pages. Then we had to incorporate a timer in the script that restarted the Chrome Driver every 100 pages to ensure a stable scraping. It is shown in the picture above.

After testing the script, we calculated the average to be 9.2 seconds each page.
Simple calculation to calculate how long it would take to scrape it all:

50.238 pages * 9.2 seconds = 462.189,6 seconds

462.189,6 / 60 = 7.703,16 minutes

7.703,16 / 60 = 128.386 hours

128,386 / 24 = **5.349 days**

We started the script on a virtual computer and it was consistent with the approximately 9.2 seconds each page, but the virtual computer somehow crashed after page 16.400. We restarted the virtual computer and started the script again from page 16.400. This time it ran through every page without further crashes.

We inserted the data into a sQlite database. Initially we wanted to insert the data directly to the main databases that we were using. The main databases were located on a Microsoft Azure Database Server, where we created two different databases, a development database for testing and a live database. Unfortunately, we ran into a problem with encoding special characters when inserted directly to the main database. We tried multiple methods to fix the encoding problem but without success. Instead we made a work around where we inserted the data to a sQlite database and then exported the data as a CSV file to Microsoft Azure Machine Learning Studio (the tool we will be using for the machine learning part) and then again exported it to our main database.

We ended up with 1.005.968 rows and 14 attributes when we inserted it into the database. Explanation of the attributes:

<table>
<thead>
<tr>
<th>Id</th>
<th>We gave every sale a unique number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>road name, road number and floor number</td>
</tr>
<tr>
<td>zipcodeAndCity</td>
<td>The zip code and name of the city</td>
</tr>
<tr>
<td>Last Seen</td>
<td>When the house was seen the last time and also equal to the sales data</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Price</td>
<td>The price when the house was put up for sale</td>
</tr>
<tr>
<td>Expenses</td>
<td>The sum of all expenses to the house.</td>
</tr>
<tr>
<td></td>
<td>Joint expenses for landowners / owners</td>
</tr>
<tr>
<td></td>
<td>Contribution to antenna association or hybrid network</td>
</tr>
<tr>
<td></td>
<td>Contribution to the common facilities and private roads</td>
</tr>
<tr>
<td></td>
<td>Municipal property tax (grundskyld)</td>
</tr>
<tr>
<td></td>
<td>Property tax</td>
</tr>
<tr>
<td></td>
<td>Renovation, Chimney sweep and Rat Fight</td>
</tr>
<tr>
<td></td>
<td>Property insurance</td>
</tr>
<tr>
<td></td>
<td>Payments of any Debts taken over by the buyer outside the purchase price</td>
</tr>
<tr>
<td></td>
<td>Other similar expenses incurred by the property</td>
</tr>
<tr>
<td>Changes</td>
<td>Changes in the prices since the house was on the market</td>
</tr>
<tr>
<td>Rooms</td>
<td>Number of rooms</td>
</tr>
<tr>
<td>sqm House</td>
<td>Number of square meters inside the house</td>
</tr>
<tr>
<td>sqm Area</td>
<td>Number of square meters of the ground area</td>
</tr>
<tr>
<td>time On Market</td>
<td>Number of days the house was on the market</td>
</tr>
<tr>
<td>buildYear</td>
<td>The year the house was build</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>energyLevel</td>
<td>The level of energy-effectiveness the house is valued</td>
</tr>
<tr>
<td>TypeOfHouse</td>
<td>The type of house, apartment, house, terraced house etc.</td>
</tr>
</tbody>
</table>

### 5.2.3 Data Preparation

Explanation to what is coming next in relation to the CRISP-DM model.

![CRISP-DM Boliga Data Preparation](image12)

We have gathered a dataset consisting of 13 attributes and more than a million sales records. The task was then to prepare and clean the data so it could be used in models. We have found and followed one SQL query best practice, SQL Formatting standard [43] and saved all queries in order to secure a good overview of what is happening in code and for backtracking possible errors and find metadata. All queries executed can be found in the appendix.

First step was to clean the data and get rid of what we defined as bad data. Often larger datasets contain outliers that could have been caused by human errors, technical flaws,
system crashes or similar [45]. That is why we started to make lots of select statements to identify bad data in our dataset that possibly would end up being outliers. We cannot just delete outliers if they show the truth, but when it is caused by bad data, then we can delete the outliers (Hodge & Austin, 2004).

What we deleted and why:

<table>
<thead>
<tr>
<th>What we deleted</th>
<th>Why we deleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where sales price was &lt; 10.000 DKK</td>
<td>We don’t consider them at the general real estate market</td>
</tr>
<tr>
<td>House with &lt; 1 room</td>
<td>No house has less than 1 room</td>
</tr>
<tr>
<td>Sales records with missing data, such as no address,</td>
<td>We consider the sales records as bad data that could cause issues with our</td>
</tr>
<tr>
<td>house number, zip code etc.</td>
<td>models</td>
</tr>
<tr>
<td>Sales records where zip code was &lt; 1.000</td>
<td>The zip codes in Denmark starts from 1.000[42]</td>
</tr>
<tr>
<td>Duplicate rows</td>
<td>Bad data</td>
</tr>
<tr>
<td>Sales records where square meter house = 0</td>
<td>To eliminate bad data</td>
</tr>
</tbody>
</table>

After deleting sales records defined as bad, we continued to go through every column in order to clean the dataset.

We discovered that the NULL values we got from scraping were somehow exported as NULL text instead of NULL integers. We replaced all NULL with “0” and after that we converted it to INT datatype so it could be used for calculation.
We cleaned all numbers so they did not contain "." like "1.000.000", that became "1000000".

```
ALTER TABLE boliga2 ALTER COLUMN price INTEGER NOT NULL
```

Code 3 – SQL Changing Data Type

For error reduction in table updates, we used a transaction method, where we could see the outcome before the table was updated with new data [44]. Example is shown here:

```
BEGIN tran
UPDATE boliga2 SET price = REPLACE(price, ',', '')
SELECT TOP 100 * FROM boliga2
ROLLBACK tran
```

Code 5 – SQL Transaction Method Example

If we were satisfied with the outcome, we would simply use the update statement without the transaction method.

We continued this methodology and cleaned our dataset, so it did not consist of symbols and commas and converted the data types so they could be used for calculations in the models. We created five new attributes and took the road name data from the address attribute and inserted into a new attribute [streetName] and the [zipcode] data from [zipcodeAndCity] and inserted it into the zip code.
Code 6 – SQL Splitting Columns Example

A new attribute was created and called [periodOfSale] which was the same as [lastSeen] but without “Sidst set”.

Two new attributes were calculated:

\[
2017 - [\text{buildYear}] = [\text{houseAge}]
\]

\[
\frac{\text{Price}}{\text{Square meters}} = [\text{sqmHousePrice}]
\]

Overview of the new attributes below with data type:

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Data type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>streetName</td>
<td>nvarchar(50)</td>
<td>The road/street name</td>
</tr>
<tr>
<td>zipcode</td>
<td>int</td>
<td>The zip code</td>
</tr>
<tr>
<td>periodOfSale</td>
<td>nvarchar(50)</td>
<td>The month and year where the house was sold</td>
</tr>
<tr>
<td>houseAge</td>
<td>int</td>
<td>The age in years of the house</td>
</tr>
<tr>
<td>sqmHousePrice</td>
<td>int</td>
<td>The price per square meter for a house</td>
</tr>
</tbody>
</table>
5.2.4 Market Analysis

Data is now prepared and structured so it is ready for modelling. For understanding purposes, we choose to go back to the data understanding phase again, to make some basic statistical overview of the real estate market by using our gathered dataset and by that getting a better understanding of the real estate market.

The purpose was to get an understanding of the real estate market in Denmark for accomplishing the best possible conditions for creating a model to predict the price of a house.

We have used our scraped dataset from Boliga to investigate the market. We have used all the sales records between year 2007 and 2016, which is approximately one million sales.

We choose to use Microsoft Power BI [68] as the tool for visualizing the dataset. We imported the Boliga dataset and started modelling charts to visualize the data. The first chart [Image 14], shown below, is a simple overview chart showing the evolution for the square meter price from 2007 to 2016. The black line represents the average square
meter price of all kinds of houses and apartments sold in Denmark in that period. The graph is decreasing after the financial crisis from 2008 until 2012 and after that, it has been increasing, but it has not reached the same level in 2016 as it had before the financial crisis in 2008.

Image 14 - Chart of the average square meter price in Denmark 2006-2016

In the next chart [Image 15], shown below, we have split apartments and houses, with the assumption that these two types of houses are subcategories in the market, so we separately can see how they evolved in the same period. The red line represents apartments and the blue line represents houses. The chart is showing that the overall square meter price for houses has not evolved significant and the average square meter prices is slightly lower in 2016 than it was from the starting point in 2007. For apartments, we see how the financial crisis impacted the prices from 2008 to 2012 and how the prices are increasing almost linearly from 2012 until 2016. Another notable thing is that there is a huge difference between the average square meter price for houses and for apartments. In 2016 the average square meter price for apartments was approximately 28,000 and for houses it was approximately 15,000, which is almost half.
We have now seen a difference in the average square meter price evolution between houses and apartments and how it generally evolved from 2007-2016. In the next section, we will consider the difference between cities, based on an assumption that the prices in the cities have increased and the prices out on the countryside have decreased. First, we will look at the evaluation of the square meter price in Copenhagen, which is zip codes between 1000 and 2500. The chart is shown below [Image 16]. Again, the red line represents apartments and the blue, houses. This time we see two lines that almost follow the same decrease and increase patterns. The square meter price for apartments decreased from 2007 to 2011 and then stagnated for a year and then it is almost increasing linearly until 2016. For houses the price decreased from 2007 to 2009 and then the price slowly increased until 2016. The average square meter price for houses in Copenhagen is higher than apartments. The square meter price average for apartments and houses are higher in 2016 than it was before the financial crisis in 2008.
The next two charts [Image 17 and 18], will first cover just outside central Copenhagen and then some suburban cities to Copenhagen. The first Chart, shown below, shows average square meter price for houses and apartments between zip code 2501 and 2900 which covers all the cities outside of central Copenhagen but is still part of greater Copenhagen. The red line represents apartments and the blue represents houses. The growth pattern looks quite like image 16; the noticeable difference is the larger gap between houses and apartments. For houses and apartments in greater Copenhagen, the average square meter price has not reached the same level in 2016 as they had before the financial crisis. We will now go one further step out of Copenhagen.
We have now moved further to Birkerød, Allerød and Hillerød area, zip code 3400-3499, which is approximately 25-35 km north from Central Copenhagen. Chart shown below, shows the evolution for apartments and houses in this area and again the red line represents apartments and the blue represent the houses. The overall development shows the same pattern as the other charts. A negative growth from 2007 to 2012 and then an increasing growth. The average square meter price is still far from its level in 2007, approximately 5,000 DKK below for both apartments and houses. This is starting to indicate, that the further away from Copenhagen the less is the growth in the average square meter price.
Image 18 - Average square meter price evolution for zip codes between 3400 and 3499

The next chart [Image 19], shows Køge, zip code 4600, which is a city 45 km south from Copenhagen. The red line represents apartments and the blue represents houses. We see a negative growth from 2007 to 2012/2013 and then a steady growth for apartments and a slowly growth for houses. Apartments in 2016 is a little bit above the level in 2007, but the houses in Køge is still approximately 3,000 DKK below.

Image 19 - Average square meter price evolution for zip code 4600

The chart [Image 20], shows the two Danish Iceland’s Lolland and Falster, which is approximately 150 km south from Copenhagen. The red line represents apartments and
the blue represents houses. The development in this region is showing another picture. The average square meter price for both apartments and houses have been decreasing since 2007. For apartments, the price is decreased with almost 50 % since 2007 to 2016 and for houses they have approximately decreased by 30 % from 2007 to 2016.

Image 20 - Average square meter price evolution for zip codes between 4800 and 4999

**Conclusion for the market analysis**

There is a difference between apartments and houses when looking at the average square meter price evolution. In Copenhagen, the average square meter price is higher for houses than apartments. The average square meter price in Copenhagen is higher in 2016 than it was before the financial crisis in 2008, which is different from the cities outside Copenhagen. Our analysis was primarily focusing on the Zealand region and including Lolland and Falster. It indicated that the longer away from Copenhagen, the cheaper is the average square meter price and likewise is the growth also lower or even negative. We can thereby interpret that there may exist different real estate markets within the Danish real estate market.

After finishing this analysis, we discovered that the price we scrape on Boliga, was not necessarily the actual price but instead it could be the latest offer price. This was a minor error source for this analysis but the difference between the actual sales price and the latest asking price was a general error for the whole dataset so the dataset should still be able to indicate the trends on the market.
We realized that we needed to improve our dataset so it at least contains the actual sales price.

5.3 KMD Data Understanding and Preparation

After gathering data from boliga and conducting our descriptive analysis of the data, we acquire a complete extraction of publicly available real estate data from KMD. KMD is the company in charge of operations regarding these databases. Included in the extraction was “Bygnings og boligregister” BBR [62], “Ejendomsstamregister” ESR [63], “Ejendomsdata” OIS [64] and “Statens Salgs- og Vurderingsregister” SVUR [65].

In the CRISP-DM process this section concerns itself with both the data understanding and the data preparation.

We saw two ways to approach this dataset before modelling. We could either select and join the data from various number of tables every time a model should run or we could make a new table with all the attributes we wanted to use. We choose the second way, by inserting all the data into a new table. We started establishing a join that could select and input all our desired attributes into one table.
We choose to store the data on a Microsoft Azure server so we wherever and whenever could access the data. We had initially also decided to use Microsoft Azure Machine Learning as the tool to design our models and run them. By locating the database on a Microsoft Azure server, it was easy and fast to connect and run the models from Microsoft Azure Machine Learning Studio.

We used Lucidchart [60] to get an overview of all the tables and which keys we could use to join tables. Two examples shown below [Image 22 and 23] where the lines represent the keys we used to join tables together. In the first one we join on three keys that are not unique them self, which is why we had to use three keys, VEJ_KODE, HUS_NR and KOMMUNE_NR, meaning road name, house number and municipality number. Some tables were easier to join than others, since they contain a primary key like AdgAdr_id which is the id of the entrance to a house or the main entrance to an apartment. For apartments, this id was not unique since there could be multiple apartments in one entrance, that is why we had to join tables with EnhAdr_id, which is the actual unique identifier for a house or an apartment and not just the entrance.

The second example, shown below, illustrates a non-complex join, since all tables consisted of unique id’s.
We followed this procedure until we had an overview of all the tables we were going to join and which keys to be used. Entity Relationship Diagram of all tables that are being used is shown below [Image 24].
KMD has a documented overview of all the tables in the BBR register named Cognito catalog. It contains information about the attributes data types, how they could be joined and a short explanation of what they are showing [61]. We used this documentation to find the required attributes, since the table names did not give an indication of what the table contained.
We initiated a join where we gathered all the attributes we used in the Boliga dataset in addition to some more we found relevant. We ended up with 23 attributes and the price attribute was the actual price this time. All the selected attributes are shown below and the name of the table they came from is shown on the left.

```
CO42100T.EnhAdr_id as 'ID',
CO42000T.VEJ_NAVN as 'StreetName',
CO11500T.HUS_NR as 'HouseNumber',
CO42100T.SIDE_DOERNR as 'DoorSide',
CO42100T.Etagebetegn as 'Floor',
CO42000T.PostByNavn as 'City',
CO42000T.PostNr as 'Zipcode',
CO43200T.KoorNord as 'CoordinateNorthing',
CO43200T.KoorOest as 'CoordinateEasting',
CO43200T.DDKNcelle100m as 'DDKNcelle100m',
CO43200T.DDKNcelle1km as 'DDKNcelle1km',
CO43200T.DDKNcelle10km as 'DDKNcelle10km',
CO40400T.VAERELSE_ANT as 'Rooms',
CO42000T.KomKode as 'MunicipalityNumber',
CO40400T.ENH_ANVEND_KODE as 'TypeOfHouse',
CO40100T.OPFOERELSE_AAR as 'BuildYear',
CO40100T.OMBYG_AAR as 'RebuildYear',
CO40400T.BEBO_ARL as 'SqmHouse',
CO11500T.EJD_MATR_ARL_SAML as 'SqmArea',
CO15900T.OMREGNINGS_DATO as 'SalesDate',
CO15900T.KOEBESUM_BELOEB as 'Price',
CO15900T.OVERDRAGELSE_KODE as 'TypeOfSale'
```

Code 7 – SQL Selecting Required Attributes

An overview of the attributes is listed below.

<table>
<thead>
<tr>
<th>Name of attribute</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Uniqueidentifier</td>
<td>The house or apartment ID in BBR</td>
</tr>
<tr>
<td>StreetName</td>
<td>Varchar(40)</td>
<td>The name of the street</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HouseNumber</td>
<td>Char(4)</td>
<td>The number of the house or apartment</td>
</tr>
<tr>
<td>DoorSide</td>
<td>Varchar(4)</td>
<td>Telling if the door is on the right or the left side of the floor</td>
</tr>
<tr>
<td>Floor</td>
<td>Varchar(2)</td>
<td>The floor level</td>
</tr>
<tr>
<td>City</td>
<td>Varchar(20)</td>
<td>The name of the city</td>
</tr>
<tr>
<td>Zipcode</td>
<td>Varchar(4)</td>
<td>The zip code</td>
</tr>
<tr>
<td>CoordinateNorthing</td>
<td>Decimal(10,2)</td>
<td>Northing geographical location</td>
</tr>
<tr>
<td>CoordinateEasting</td>
<td>Decimal(10,2)</td>
<td>Easting geographical location</td>
</tr>
<tr>
<td>DDKNcelle100m</td>
<td>Varchar(15)</td>
<td>BBR has divided locations up in 100m cells</td>
</tr>
<tr>
<td>DDKNcelle1km</td>
<td>Varchar(15)</td>
<td>BBR has divided locations up in 1km cells</td>
</tr>
<tr>
<td>DDKNcelle10km</td>
<td>Varchar(15)</td>
<td>BBR has divided locations up in 10km cells</td>
</tr>
<tr>
<td>Rooms</td>
<td>Smallint</td>
<td>Number of rooms</td>
</tr>
<tr>
<td>MunicipalityNumber</td>
<td>Int</td>
<td>The id number of the municipality</td>
</tr>
<tr>
<td>TypeOfHouse</td>
<td>Smallint</td>
<td>Apartment or house</td>
</tr>
<tr>
<td>BuildYear</td>
<td>Smallint</td>
<td>The year the house or apartment were build</td>
</tr>
<tr>
<td>RebuildYear</td>
<td>Smallint</td>
<td>If the house or apartment were registered rebuild</td>
</tr>
<tr>
<td>SqmHouse</td>
<td>Int</td>
<td>Number of square meters inside the house</td>
</tr>
</tbody>
</table>
We started cleaning this dataset like we did with the Boliga dataset. Then we followed the same procedure and used the SQL queries to delete bad data as we did with the Boliga dataset. Surprisingly there were a lot more bad data in the KMD data, an example was the dataset should not consist of sales data prior 1992, but there were records with date stamp from 1753-01-01 which indicated something was wrong with those records.

![Image 25 – Bad Data Example in Database](image)

We choose to delete such records, where sales date was prior year 1992, since we assessed it was bad data, example shown below.

```
DELETE
FROM Dataset6NoBasement
WHERE YEAR(SalesDate) < 1992
```

Code 8 – SQL Deleting Bad Data Example

We ran through all attributes and looked at the minimum and maximum values in order to determine whether there were any more bad data as explained above. After the cleaning, we ended up deleting 336,825 sales records and ended up with 1,695,749 sales records.
in our joined dataset. 1,169,000 were houses and 203,947 apartments and the rest were other types like summerhouses and terraced houses.

The market analysis also indicated two different markets for apartments and houses, which gave us an assumption that there should be used two different models to predict the price whether it was a house or an apartment. As a proof on concept, we delimited us to only make one of them and it ended up being houses, since that type had most sales records.

5.4 KMD Data Modelling

In this section, we will describe how we modelled our price index predictions and how we did the machine learning modeling. We will also describe how we combined both models. In relation to the CRISP-DM process, this is the “Modeling” part.

![Image 26 – CRISP-DM Modelling](image)

5.4.1 Baselines

We will start by creating a set of simple baselines to compare our models with. Done to make sure our models are not worse than or comparable to random guesses. All baselines are made in SQL, since it was simple calculations and only required few lines of code.
Baseline 1 - The average of all Extrapolated Prices.
It is a simple baseline, where we calculate the average of all extrapolated prices and use them as the square meter price. Calculation shown below.

```
SELECT AVG(indexpriceresult) as 'AvgIndexSqmPrice'
FROM indexmodel18
```

Code 9 – SQL Baseline 1

Baseline 2 - The Average Square Meter Price of all sales in 2015.
We used all sales data for year 2015 and calculated the average of the square meter price. Calculation shown below.

```
SELECT AVG(sqmHousePrice)
FROM dataset6nobasement
WHERE YEAR(SalesDate) = 2015 -- Only sales year 2015
AND TypeOfSale = 1 -- Normal Sale
AND TypeOfHouse = 120 -- Only houses
```

Code 10 – SQL Baseline 2

Baseline 3 - The Average Square Meter Price by Zip Code in 2015.
The third baseline required a bit more lines of SQL code but the approach was simple. Grouping sales by the third and fourth in the zip code, we took the average square meter price in the year 2015. After we found the average square meter price for every zip code then we joined the result on zip codes in year 2016. This outputted all sales records for year 2016 with both the actual price and the calculated baseline price for comparison. The calculation and join is shown below [Code 11].
5.4.2 Price Index Model

We are going to make a forecasting model where we used the previous sales data to calculate a market price index model. The formula is the same as described in the literature review from our earlier study [20] and in the forecasting section in the conceptual framework.

\[
\frac{\text{Previous sales price}}{\text{Previous market index}} = \frac{\text{Current sales price}}{\text{Current market index}}
\]

The index model is built on an assumption that the relationship with the price of a house and the market is the same over time. The formula ends up looking like this after using basic algebra.

\[
\frac{\text{Previous sales price}}{\text{Previous market index}} \times \text{Current market index} = \text{Current market price}
\]
We made the model and calculations in SQL. Due to our market analysis, we found out that the evolution is different depending on where you are located in Denmark. Therefore, we used this finding in our model. We choose to split and define the markets as grouped by their zip codes. More specifically we choose to group the sales by the third and fourth digit in the zip code. So, all sales within zip code 2200-2299 were calculated in one index and 2300-2399 in another etc. We wanted the houses sold to be place geographically close in each grouped index [42]. We choose to make zip code 1000-2000 in one index, since the zip code logic does not follow the same logic in the inner Copenhagen area [40],[41] and there are not that many houses in inner Copenhagen. Only 1132 house sales were recorded between year 1992 and 2016. The indexes were calculated on an annually basis starting from year 1992 as market index value 1 and the latest index for a zip code was year 2016.

For testing purposes, we left out sales records from 2016 that have been sold before, since the index model is limited to only predict the price of a house that has been sold before. Therefore, the training set of the index model consist of all sales data between 1992 and 2015 and for 2016 only the sales records that has not been sold before This amounts to 1.671.864 sales records in total. The test set consisted of houses that had been sold in 2016 and sold before in our dataset, which is 23.885 sales records.

We started making simple averages, one general average square meter price per index and one only for inner Copenhagen.

```
,AVG(price/SqmHouse) OVER (PARTITION BY YEAR(SalesDate),MONTH(SalesDate),SUBSTRING(zipcode,1,2)) AS avgM2PriceZipMonth
,AVG(price/SqmHouse) OVER (PARTITION BY YEAR(SalesDate),MONTH(SalesDate),SUBSTRING(zipcode,1,1)) AS avgM2PriceZipMonthInCPH
```

Code 12 – SQL Calculating Averages

As mentioned before, we made an index model for inner Copenhagen and one for the rest of the country. The calculations shown below [Code 13] is for inner Copenhagen and we follow the same structure for the rest of the country, but with minor changes.
At the end, we inserted the results in a new table.

The whole query can be found in the appendix 12.

5.4.3 Machine Learning Analysis

The tool we have chosen to do the machine learning modeling in is Azure Machine Learning Studio (Azure ML). This is an online tool provided by Microsoft and can be accessed here: https://studio.azureml.net/

The principle behind the tool is to provide customizable building blocks to create machine learning models. These building blocks are called modules and technical descriptions of these can be found on the “Reference” site [93] provided by Microsoft.

After preparing and cleaning the data acquired, we fed it into Azure ML and started building our machine learning models. We wanted to determine the performance of a machine learning model alone and the performance of a machine learning model combined with the price index model. The data we are using in the models, where we do not use the extrapolated price index data, consists of 40.956 rows of sales from the year 2015. We will try to predict the price of 24.007 sales in 2016. The combination of the price index model and machine learning is done by feeding extrapolated data (by the price index model).
model) into the machine learning model. This extrapolation makes it possible to feed 1,144,993 rows of sales (up to and including 2015 sales) into the machine learning model, in order to predict the price of 24,007 sales in 2016. This is almost 28 times more training data compared to only using 2015 data. The hypotheses is that a bigger volume of data will improve the predictions of the machine learning model.

We will be modifying the model in order to try out different configurations and test the performance of each machine learning model with extrapolated data and without. We will also train the models using only one attribute to predict the price as well as a set of attributes listed below. When using the term “One attribute” later in this thesis, we refer to models using only the attribute “DDKNcelle10km”. When using the term “Multiple attributes”, we refer to models using all the attributes listed below.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Attribute description</th>
<th>Attribute type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDKNcelle10km</td>
<td>10km location cell</td>
<td>Category</td>
</tr>
<tr>
<td>StreetName</td>
<td>The street name.</td>
<td>Category</td>
</tr>
<tr>
<td>Zipcode</td>
<td>The zip code.</td>
<td>Category</td>
</tr>
<tr>
<td>DDKNcelle1km</td>
<td>1km location cell</td>
<td>Category</td>
</tr>
<tr>
<td>Rooms</td>
<td>The number of rooms.</td>
<td>Numeric</td>
</tr>
<tr>
<td>SqmArea</td>
<td>The square meter area of the land belonging to the estate.</td>
<td>Numeric</td>
</tr>
<tr>
<td>SqmHouse</td>
<td>The square meter area of the livable house.</td>
<td>Numeric</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>MunicipalityNumber</td>
<td>The technical number of the municipality the house belongs to</td>
<td>Numeric</td>
</tr>
<tr>
<td>BuildYear</td>
<td>The year the house was build.</td>
<td>Numeric</td>
</tr>
<tr>
<td>RebuildYear</td>
<td>Indicates the year (if registered) of complete renovation.</td>
<td>Numeric</td>
</tr>
<tr>
<td>SoldBefore</td>
<td>Indicates if the house has been sold before.</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

Houses in Denmark are grouped into different types of location cells. The entire country is divided in different grid structures consisting of square cells. The ones we use are measured in 10 km and 1 km. The attributes “DDKNcelle10km” and “DDKNcelle1km” indicate which of these cells any given house is placed in. These are location based attributes for the house. Besides using all attributes in each model, we will also use one of these (the DDKNcelle10km attribute) as the single attribute for the machine learning models. We will do this to test how much a large amount of attributes performs compared to a small amount of attributes.

Below [Image 27] is an overview of how we set up each machine learning model that we tested. We will go through each step of the process, from importing the data through training the model and to evaluating its performance.
Image 27 – Machine Learning Model Setup with Extrapolated Data
Image 28 – Machine Learning Model Setup without Extrapolated Data

The two models above [Image 27 and 28] are tied together. They both use the same data input (step 1) even though they select different parts of the data. Both models are evaluated (step 6) side by side. This was done for practical reasons and does not influence the models themselves. The lines that go to the edge of the images (from step 1 and 6) are the lines connecting the two models. Step 1 and 6 has been centered on each model image for better visual representation.
The data is first fed into the two different models in step 1. In the images, this data is named “indexModel18_v3”. This name is created by us for practical reasons and has no effect on the dataset. After the data import the data is split into a training set (step 2a and 4a) and testing set (step 3a and 5a). This is done to avoid overfitting. The way we have done the splitting is by using sales data from 2016 as testing data. That is, we try to predict the sales price of any given house in the year 2016. The training data consists of sales before 2016. In the model, we fed with data containing the extrapolated data, the model uses all previous sales before 2016 extrapolated to 2015 in order to predict the price. In the model without the extrapolated sales data, the training set (step 4a) consists of all sales in 2015. This method of splitting is the hold out set method. Since we are using data acquired until a certain date to forecast the new sales prices we do not cross validate our models. After splitting the data, we used the modules 2b, 3b, 4c and 5b to select specifically the data parts we wanted to use. This includes attributes as well as the label. The steps 3c, 4b and 5c are technical modules that simply passes on the attribute data and renames the label data in preparation for training and testing.

For each set of machine learning models build, we chose a regression algorithm. We will be using a boosted decision tree regression and a decision forest regression. In the steps 2c and 4d, we have inserted the algorithm into the model. Each algorithm can be tuned (or configured) in different ways according to its parameters [93].

We decided on the parameters based on a tuning experiment. We will get back to how we did this after the explanation of our model setup.

5.4.4 Boosted Decision Tree Regression Configuration

We have configured our boosted decision tree regression with these parameters:
“Create trainer mode” refers to whether we want the model to try different parameters or if we want a single parameter (manually setting how the algorithm should work). Due to the large time consumption when letting the model try out different parameters, we chose to manually input the parameters and conduct the tuning of parameters separately. The “Maximum number of leaves per tree” is the maximum number of terminal nodes per tree. That is the maximum number of different predictions. The “Minimum number of samples per leaf node” is the minimum number of sales used to predict the price in a given terminal node. The “learning rate” defines the step size while learning. By Microsoft [93] the “learning rate” is defined as: “The learning rate determines how fast or slow the learner converges on the optimal solution. If the step size is too big, you might overshoot the optimal solution. If the step size is too small, training takes longer to converge on the best solution.” This means that a smaller learning rate may cause the model to be better but is computationally more intense. The “Total number of trees constructed” is the total number of decision trees created by the algorithm. Increasing this number may lead to
better performance, but will increase the computational requirement. “Random number of seeds” is a way of ensuring reproducibility across multiple runs that have the same data and configuration [93]. The “Allow unknown values for categorical features” is a setting that determines whether the model will accept yet unseen data or not when scoring testing data. This is very important to allow, since we need to predict the price of houses in the test data that has not been used for training and may contain data with yet unseen values.

5.4.5 Decision Forest Regression Configuration

We decided not to conduct an automatic tuning experiment of the parameters of the Decision forest regression, since it was too computationally intense. A single run though the models with fixed parameters took more than 24 hours, so we decided to manually tune the parameters after each run. We only did a handful of runs before settling on the following configuration:

![Decision Forest Regression Configuration](image30.png)

Image 30 – Decision Forest Regression Configuration
The configuration was tuned manually by trying to improve the coefficient of determination. We will get back to the evaluation of the models and algorithm configuration later on.

The “Resampling method” in the decision forest configuration determines how the data is used by the trees in the “forest”. Bagging means that data is split amongst each tree [93]. Bagging is also referred to as bootstrap aggregating. The “Create trainer mode” is the same as in the boosted decision tree regression. The “Number of decision trees” configures how many trees will be created in the forest. The more trees created, the more computationally intensive the training of the algorithm becomes. More trees may make the predictions better. The “Maximum depth of the decision trees” refers to how deep the decision trees are allowed to be. A deeper tree may make better predictions, but will be more computationally intense. There is also a greater risk of overfitting when making the trees deeper. The “Number of random splits per node” is fairly self explanatory since it refers to the number of random splits in each tree per node. The “Minimum number of samples per leaf node” configures what the minimum number of data entries per terminal node is required in each tree. The “Allow unknown values for categorical features” is the same in decision forest regression as in boosted decision tree regression.

The next step in the model (step 2d and 4e) is to train the model based on the training data and the configured algorithm. The “Train model” module (step 2d and 4e) is configured with which column of data is the label data in order to differentiate between the label and attributes.

The trained model can be visualized as the individual trees constructed during training both for the boosted decision tree regression algorithm and the decision forest regression algorithm. Here is a visualization of a trained model:
Image 31 – Trained Boosted Decision Tree Regression Visualization
The image above shows a single decision tree created by the machine learning model using the boosted decision tree regression algorithm. Many of the branches have been visually collapsed to give a better overview of the beginning of the tree. The tree splits into branches and ends up in a terminal node (marked with thick blue rings) that predict the square meter price of a house. In this case the algorithm created 500 of these trees. A thumbnail image of the first 14 can be seen to the left in the image above.

After training the model, we score the testing data (step 3d and 5d) by hiding the testing data label and feeding the testing data attributes into the trained model. This produces a prediction for each house in the testing data.

### 5.4.6 Tuning the Algorithms

In order to figure out what parameters to use to get the best performing models, we used a tuning module in a separate model. The tuning module called “Tune Model Hyperparameters” is inserted into the model instead of the “Train Model” Module like so:
Instead of training the model with a single configuration for the algorithm, this tuning module makes it possible for the computer to try different combinations of configurations of the algorithm and pass on the best trained model. The tuning module can be configured like so:
“Specify parameter sweeping mode” determines how the configurations for the algorithm are selected. “Random grid” means that the computer selects random samples of the entire grid of possible configuration. We chose not to use the Entire grid since it is very computationally intense.

“Maximum number of runs on random grid” defines how many times the model will tune the algorithm with random grid samples. We have chosen 50 runs due to computational complexity. When running this model to find the best trained model for the boosted decision tree regression, it took approximately 9.5 hours to finish. It only took a bit less than 2 hours to run both models (with and without extrapolated data) with the boosted decision tree regression algorithm, when a single parameter was entered. This single parameter was the best found by the tuning. It could easily take several days or weeks if we tuned the parameters for each model every time and increases the amount of iterations done in the tuning.
“Random seed” in the tuning module is defined by Microsoft [93] as “... a number to use when initializing the parameter sweep.”, “This is optional, but can be useful for avoiding bias introduced by seed selection.”. We will use the default “0”.

The “label column” specifies which part of the training data is the label data.

Since we will be using a regression algorithm, the module will disregard the configuration of “Metric for measuring performance for classification” and only use the configuration “Metric for measuring performance for regression”. This determines how the tuning module will assess the quality of the algorithm configuration. We will be using “Coefficient of determination (or r^2)” to tune the algorithms, since this is a representation of how well the attributes of a house explain the square meter price of a house. The better they can explain the price with the algorithm, the better the predictions. The coefficient of determination is calculated based on the training data. This may introduce overfitting. In order to verify the performance of the best trained model, it is fed with test data to score in the module “Score Model” and then the coefficient of determination as well as MAE and RMSE is calculated for the test dataset in the module “Evaluate Model”.

Finally, we need to evaluate the model. We have chosen to set up our experiments so that two models are evaluated side by side. The “Evaluate Model” module will calculate the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), “Relative Absolute Error”, “Relative Squared Error” and Coefficient of Determination (r^2). We will be using the MAE, RMSE and r^2 to evaluate how well the models perform. This is how the evaluation output looks:

![Decision Forest Regression Evaluation Results](image34.png)

Image 34 – Decision Forest Regression Evaluation Results
We will show our results and evaluate all the presented models in the coming section called “Results”. Here is an overview of all the machine learning models we have build:

<table>
<thead>
<tr>
<th>Model</th>
<th>Attributes</th>
<th>Training data</th>
<th>Test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosted decision tree</td>
<td>One attribute</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Boosted decision tree</td>
<td>Multiple attributes</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Boosted decision tree</td>
<td>One attribute</td>
<td>2015 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Boosted decision tree</td>
<td>Multiple attributes</td>
<td>2015 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision forest</td>
<td>One attribute</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision forest</td>
<td>Multiple attributes</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision forest</td>
<td>One attribute</td>
<td>2015 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision forest</td>
<td>Multiple attributes</td>
<td>2015 sales</td>
<td>2016 sales</td>
</tr>
</tbody>
</table>

Image 35 – Complete Machine Learning Model Overview
6.0 Results

In this section, we will present our results and present the insights gained from our experiment with machine learning and forecasting where we have used real estate sales data. As a part of the CRISP-DM process, this represents the “Evaluation” element.

Image 36 – CRISP-DM Evaluation

As indicator for how well each model performs, we have calculated three measures of error:

- Mean absolute error (MAE) (measured as square meter price)
- Root mean squared error (RMSE) (measured as square meter price)
- Coefficient of determination (R-squared or $r^2$)

We have listed our results for each model below. The first table is sorted by model to give an overview. The second table is sorted by the coefficient of determination with the best
fitting model at the top. The coefficient of determination is calculated as the goodness of fit for a function (the predictions) compared to the actual values (sales prices). $R^2$ can be interpreted as a description of how well the attributes used to create the function explain the label predicted. We have use the $r^2$ to rank our models. From this ranking, we will describe our findings.

In the table below we have listed our results. The top table is sorted by model. The bottom table is sorted by model fit (coefficient of determination). Here is an explanation to the headers in the table below [Image 37]:

- “Attributes” refers to the set of attributes used as part of the training data in the machine learning models.
- “Training data” refers to whether extrapolated data from the forecasting model was used to train the model, or if 2015 sales data was used.
- “Test data” refers to the data used to test the models.
- “MAE” refers to the mean absolute error of the specific model.
- “RMSE” refers to the root mean squared error of the specific model.
- R-squared refers to the coefficient of determination of the specific model.
<table>
<thead>
<tr>
<th>Model name</th>
<th>Attributes</th>
<th>Test data</th>
<th>Training data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2015 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2015 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>One attribute</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>Multiple attributes</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
<tr>
<td>Decision Forest</td>
<td>All sales extrapolated excluding 2016</td>
<td>2016 sales</td>
<td>2016 sales</td>
</tr>
</tbody>
</table>
6.1 Meaningful Facts

In the table where the models are sorted by \( r^2 \), the baseline 1 and 2 rank at the bottom. With a coefficient of determination of just 0,14 we consider them as good as random guesses. This was to be expected, since they both are very simple baselines. The third baseline, based on the average square meter price in each zip code in 2015, ranks above all the models using extrapolated sales. This is a clear indication of that models ranking below the third baseline are not good predictors of real estate prices. All the models created using extrapolated data (excluding baseline 1 and 2) have a coefficient of determination between 0,55 and 0,60. This indicates that 55% - 60% of the square meter price in 2016 sales can be explained with the attributes used to create these models. These attributes include historical data of sales before 2016. By comparison, the models created where the attributes include only data from 2015 have a smaller error in predictions, and these attributes explain between 62% - 76% of the square meter price in 2016 sales. The finding is that our way of combining a market price index model with a machine learning model is not beneficial even though the volume of data is almost 28 times greater in the combined model compared to the standalone machine learning models. The price index model has a model fit of 56% which may explain why the combination of machine learning and the extrapolated sales has a worse fit than all machine learning models without extrapolated sales (using 2015 sales data). The result when using extrapolated sales together with machine learning is as Cruz and Wishart [94] puts it when describing results of poor quality: “garbage in = garbage out”[94].

The machine learning model with the best fit is the boosted decision tree regression using sales data from 2015 and multiple attributes. The calculated coefficient of determination for this model is 0,76. This is a 20 percentage points increase over the market price index model. This may indicate that our forecasting model simply is not advanced enough or good enough at predicting the square meter price to be combined with a machine learning model. The fact that the “baseline 3” performs better than the price index model is also a clear indication that the performance of the forecasting method used in the price index model simply is not good enough to be used in conjunction with models performing better than “baseline 3”.
Now, let us look at the fit of the machine learning models created using 2015 sale. There is a clear difference in the model fit between models using one attribute and models using multiple attributes. The machine learning models using one attribute to predict sales prices has almost the same fit of 64%. The machine learning models using multiple attributes have a fit between 72% - 76%. This is an indication of that a bigger variety in data describing the houses for sale allows more accurate predictions. With a fit of 76% the machine learning model still leaves 24% of the price of a house unaccounted for. We can incorporate the 24% as the part of the price of a house that is determined by other factors than the ones included as attributes in the machine learning model.

On a side note: Interestingly, but unimportant, the decision forest regression model that uses multiple attributes and extrapolated data (where \( r^2 = 0.60 \)) fits better than the price index model on its own (where \( r^2 = 0.56 \)). This is slightly contradictory to the explanation of “garbage in = garbage out”[94]. This is interesting, but unimportant due to the fact that the performance of the two models are still far from the same machine learning model using 2015 training data (where \( r^2 = 0.72 \)).

6.2 Insights

The insights gained through our analysis and the results are as follows.

Extrapolation of data done to enrich the volume of a dataset should not be done without further work. The price index model we have created (our forecasting model) simply does not perform well enough in order to be combined with machine learning models. This means that an alternative model in the future should not be based on the forecasting model that we have build.

The machine learning models on their own however perform fairly well when compared to our third baseline. In order to improve the fit of the machine learning models, we could enrich the dataset used for training with more attributes. It is difficult to acquire attributes describing the condition of a house, but this is an example of what kind of attribute that could be used to improve the fit of machine learning models when predicting the price of a house.
We have had access to a large dataset with a lot of attributes, but besides the size and the variety of the dataset, the quality is equally important. We have encountered some odd data entries and tried to clean the data as much as possible, but we cannot confirm that all the data is 100% correctly entered into the database. We do not have direct control over the quality of data created in the databases where our dataset has been pulled from, but we can get better at cleaning the data in order to remove data of poor quality.

Finally, alternative forecasting methods and machine learning methods could be used to potentially increase the accuracy of the predictions. We have chosen two machine learning algorithms and a single forecasting model. The use of other models may result in different findings.

**6.3 Deployment**

We do not consider the deployment process of the CRISP-DM model as a part of the scope for the thesis. This said, we will still comment on the process since we have used the CRISP-DM process extensively. We have used the CRISP-DM as a framework for our experimentation. We will not make deployment plan, since this is not a key component to answer our research question.

After creating a model with satisfying prediction accuracy, deployment is an option. As a part of the CRISP-DM, deployment is started when satisfying value creation has been validated through the previous process steps of “Business Understanding”, “Data understanding”, “Data preparation”, “Modeling” and “Evaluation”. In order to keep the model used in deployment up to date, it needs to be retrained continuously. Compared to the stock market or the cryptocurrency market, the real estate market is not sensitive on a day-to-day basis. This means that a machine learning model deploying in a production environment does not constantly need to retrain with new data in order to stay up to date. This has the effect on the deployment, that retraining the model with new data only has to happen on daily or weekly basis and still produce predictions with a consistent accuracy. This would of course have to be tested and validated, since it is only an assumption.
In a production environment, it would be possible to use web services to connect to the trained model in Azure Machine Learning and feed predictions to the application using the trained model.
7.0 Discussion

7.1 Machine Learning and Big Data

*Findings that support and extend the literature of machine learning in relation to big data*

The connection between the four V’s in Big data carry over to machine learning, when machine learning is used as the pattern recognition tool in large datasets. We have conducted our experiment and found that a bigger volume of data is not necessarily better if the veracity is being sacrificed. By creating more data for the machine learning models to use, we hamstrung the model by using a method that proved not to have a sufficient standard of veracity and thereby quality in data.

Furthermore, when creating predictive machine learning models and tuning the variety of data used, we found that this too was linked to the accuracy of the model output. A bigger variety of data allows the machine learning model to select the most influential ones and use them accordingly to achieve the most accurate output.

Velocity, as the fourth V is out of scope since it becomes a challenge when the final machine model needs to be deployed and kept up to date with data. Deployment is out of scope of the research in this thesis though it is important to understand in relation to big data and data mining concepts.

7.2 Combining Machine Learning and Forecasting

*Findings that support and extend the literature of machine learning in relation to combination with forecasting methods.*

Through experimentation with forecasting and machine learning we have found that certain challenges need to be addressed in order to gain the full potential of combining these two methods of prediction. As mentioned in relation to big data and veracity, we have found that a simple combination of machine learning and forecasting, will not yield more accurate predictions than the two models separately. This finding means that these two models, forecasting and machine learning, cannot be combined the way we did. This does not eliminate the possibility of combining other models or combining these models in a different way to create a model with more accurate predictions.
7.3 Suggestions for Future Work

In this thesis, we have addressed and answered our primary research question, but by answering this question, it inevitably creates new questions. The results of our analysis and the findings of our research has spawned a number of new areas of interest. In this section, we will suggest how future work might be built upon the findings in this thesis and expand the literature of the subject of machine learning, big data, forecasting and price prediction.

As our findings indicate, machine learning and forecasting cannot be combined into a signal model, as we have done, without further work. This poses the question of what methods of model combination that can be done in order to create a more accurate prediction output. Future work may investigate different methods of combining models with the same accuracy. Machine learning today utilizes combining of methods to achieve more accurate output rules. These are methods such as boosting which we have used as a part of a machine learning algorithm in the thesis. We believe that investigation into the aspects of model combination is important to take advantage of the full potential of machine learning.

Future work and research within this area may be conducted by experimenting with the use of machine learning algorithms to predict what type of model to be used for the final price prediction. An example could be a classification algorithm that, based on the attributes of a house, determines which available machine learning model would be classified as the best predictor of that type of house. Alternatively, this classification algorithm could include forecasting models in the set of models available.

Other interesting future work within the area of machine learning and real estate price prediction is to investigate the effect of variety in the data. By enriching the dataset used for training with other kinds of data, considered to be more or less related to the real estate market, it could be interesting to explore the possibilities and potential of machine learning. Machine learning is excellent at pattern recognition beyond what humans are capable of.
8.0 Conclusion

The primary aim of this thesis was to investigate how machine learning and forecasting can be used separately and combined to make more accurate real estate price predictions. By obtaining data about real estate sales from boliga.dk and via KMD, we have applied methods of big data and machine learning in order to create and compare experimental prediction models. We have analyzed the datasets and gained descriptive insights into the real estate market before building and testing our predictive models. Our descriptive analyses revealed a clear difference in the real estate market when divided into grouping by attributes such as real estate type and location. The findings from our descriptive analysis led us build our forecasting model as well as our comparable baseline with the intention of predicting house prices. Data from “Bygnings og boligregister” (BBR), “Ejendomsstamregister” (ESR), “Ejendomsdata” (OIS) and “Statens Salgs- og Vurderingsregister” (SVUR) was used to create the forecasting models, baselines and machine learning models. The machine learning models created, were based on the two algorithms “Boosted decision tree regression” and “Decision forest regression”. With the forecasting model, we extrapolated historical sales data, in order to increase the volume of the training data fed into the machine learning models. This was done with the hypotheses that more data could increase the accuracy of the machine learning models. As a result, we have calculated and compared the model fit of each model, in order to answer our research question. We found that all our models using extrapolated data had a worse model fit than the models using a smaller but current amount of data (sales data from 2015) to predict the prices of houses sold in 2016. This finding leads us to conclude that forecasting and machine learning should not be combined as done in this thesis to increase the accuracy of the predictions. The machine learning models had a higher model fit on their own than combined with the forecasting model. The model fit of our machine learning models leads us to conclude that machine learning shows great potential when applied in the real estate market. Challenge to overcome include obtaining and maintaining data that has the explanatory correlation with the price of a house.
Our forecasting model had a significantly lower accuracy compared to our most accurate machine learning model. From this, we can conclude that combining forecasting and machine learning methods to predict house prices does not yield higher accuracy as long as one method on its own is dramatically inferior to the other. We found that tuning the boosted decision tree regression iteratively yielded the most accurate prediction model, when excluding extrapolated data from the forecasting model and instead using 2015 data as training data.

Machine learning methods show great potential when applied to the real estate market. There are still challenges to overcome as described in this thesis. Nonetheless, the opportunities of using machine learning within the real estate market to create a more informed foundation for decision making are great.
9.0 References


Seventh edition, Published 2016 by Pearson Education Limited.


[42] PostNr.info ALL. Available here: http://www.postnr.info/


[44] BEGIN TRANSACTION (Transact-SQL), Microsoft, 2016, Available here:  

[45] Mehmed Kantardzic, Data Mining: Concepts, Models, Methods, and Algorithms, IEEE,  
published by John Wiley and Sons Inc Hoboken New Jersey, 2011, Available here:  
https://books.google.dk/books?id=ZZ7l6v0CvRMC&printsec=frontcover&hl=da&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[46] Gartner’s 2012 Hype Cycle for Emerging Technologies Identifies "Tipping Point"  
Technologies That Will Unlock Long-Awaited Technology Scenarios. Available here:  
https://www.gartner.com/newsroom/id/2124315


[49] SAS. Big Data - What it is and why it matters. Available here:  

35, Issue 2, April 2015, Pages 137-144. Available here:  


[53] Foster Provost and Tom Fawcett, Data Science for Business - What You Need to Know About Data Mining and Data-Analytic Thinking, 2013, published by O'Reilly Media, Inc.

[54] Pete Chapman (NCR), Julian Clinton (SPSS), Randy Kerber (NCR), Thomas Khabaza (SPSS), Thomas Reinartz (DaimlerChrysler), Colin Shearer (SPSS) and Rüdiger Wirth (DaimlerChrysler), CRISP-DM 1.0 - Step-by-step data mining guide, 2000.

[55] Ana azevedo and M.F. Santos, KDD, Semma and CRISP-DM: A Parallel Overview


[57] Rüdiger Wirth and Joches Hipp, CRISP-DM: Towards a Standard Process Model for Data Mining, CRISP-DM consortium


[61] KMD - Cognito Catalog. Available here: https://catalog.cognito.dk/


[64] OIS - Din gen1vej til ejendomsdata. Available here: https://www.ois.dk/


[86] Georg Graetz and Guy Michaels. Industrial robots have boosted productivity and growth, but their effect on jobs remains an open question. Published by The London School of Economics and Political Science 2015. Available here: http://eprints.lse.ac.uk/71064/


[89] Azure SQL Database with DirectQuery. Microsoft documentation. Available here:
[90] Yuan Liu. Incremental Learning in Deep Neural Networks. Tampere University of Technology. Available here:
https://pdfs.semanticscholar.org/986d/e273e4b661127ac48177eb600fc78a64061e.pdf

[91] Skytree, Why do Machine Learning on Big Data? Available here:

[92] SAS. Machine Learning - What it is and why it matters. Available here:


https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2675494/
9.0 Appendices

Appendix 1 – Boliga Python Scraping
import sqlite3
import datetime

DATABASE_NAME = "C:\DB\bolig429999.db"
# Creates a connection to the database
conn = sqlite3.connect(DATABASE_NAME)
cursor = conn.cursor()

cursor.execute('''CREATE TABLE bolig429999(address text, zipcodesAndCity text, lastSeen text, price text, sqfsize text, changes text, rooms text, sqmHouse text, sqmArea text, timeOnMarket text, buildYear text, energyLevel text, houseType text)''')
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time

chrome_driver = r"C:\DB\chromedriver.exe"
driver = webdriver.Chrome(chrome_driver)
driver.get("http://www.bolig429999.com/?view=1")
time.sleep(15)
driver.get("https://www.bolig429999.com/#/search?area=10&area=15")
time.sleep(15)
driver.execute_script("window.scrollTo(0,document.body.scrollHeight)")
results = driver.find_elements_by_css_selector(".searchResultList .search-results-row")

i = 0
while i < 5:
    while True:
        try:
            address = results[i].find_element_by_class_name("address").text .splitlines()
            price = results[i].find_element_by_class_name("price").text
            room = results[i].find_element_by_class_name("room").text
            sqmHouse = results[i].find_element_by_class_name("sqmHouse").text
            sqmArea = results[i].find_element_by_class_name("sqmArea").text
            buildYear = results[i].find_element_by_class_name("buildYear").text
            energyLevel = results[i].find_element_by_css_selector("[energyLevel]").get_attribute("title")
            houseType = results[i].find_element_by_css_selector("[houseType]").get_attribute("title")
            sqlString = "INSERT INTO bolig429999 VALUES("+sqlString")"
            conn.commit()
        except StaleElementReferenceException as fail:
            print(fail)
            errorcount += 1
            time.sleep(1)
            results = driver.find_elements_by_css_selector(".searchResultList .search-results-row")
            pass
            break

print("error count: " + str(errorcount))
Appendix 2 – Sql Updating Query

--Using begin tran and rollback tran method to check the outcome before updating--

BEGIN tran

--Calculating sqmHousePrice--

UPDATE Dataset3NoBasement

    SET sqmHousePrice = price / sqmHouse

    WHERE price IS NOT NULL
    AND sqmHouse IS NOT NULL

SELECT TOP 100 *
FROM boliga2

ROLLBACK tran

SELECT TOP 100 *
FROM boliga2

--Inserting the zipcode FROM zipCodeAndCity to the zipcode column--

UPDATE boliga2

SET zipcode = Left(zipcodeAndCity, PatIndex('% %', zipcodeAndCity))

--Inserting the name of the street FROM address to the streetName column--

UPDATE boliga2

SET streetName = Left(address, PatIndex('%[0-9]%', address + '1') - 1)

--Inserting the date FROM the lastSeen variable without "sidst set:" into the periodOfSale column--

UPDATE boliga2

SET periodOfSale = Right(lastSeen, PatIndex('%[0-9][0-9]%', lastSeen + '1') - 3)

--Calculating how many years FROM 2017 the house was build and inserting into houseAge column--

UPDATE boliga2

SET houseAge = ('2017' - buildYear)

--Changing all 'NULL' text to 0 in sqmArea--
UPDATE boliga2 SET sqmArea=0 WHERE sqmArea = 'NULL' -- 299556 rows changed--
--Changing all 'NULL' text to 0 in changes--
UPDATE boliga2 SET changes=0 WHERE changes = 'NULL' -- 603154 row(s) affected
--Changing all 'NULL' text to 0 in expences--
UPDATE boliga2 SET expences=0 WHERE expences = 'NULL' -- 703097 row(s) affected
--Changing all 'NULL' text to 0 in sqmHouse--
UPDATE boliga2 SET sqmHouse=0 WHERE sqmHouse = 'NULL' -- 1548 row(s) affected
--Changing all 'NULL' text to 0 in rooms - it's different method because the value was actually NULL and not 'NULL' text--
UPDATE boliga2 SET rooms=ISNULL(rooms, 0) -- row(s) affected
UPDATE boliga2 SET houseAge=ISNULL(houseAge, 0) -- row(s) affected
UPDATE boliga2 SET spmHousePrice=ISNULL(spmHousePrice, 0) -- row(s) affected
UPDATE boliga2 SET periodOfSaleNumber=ISNULL(periodOfSaleNumber, 0) -- row(s) affected
Appendix 3 – Sql Query deleting rows

-- Deleting rows WHERE the price is bad data / below 10000 for 'ejerlejlighed'--
DELETE FROM boliga2
WHERE price < 10000
and houseType = 'ejerlejlighed'
-- 3 rows DELETED --

-- Deleting rows WHERE the price is bad data / below 10000 for 'Villa'--
DELETE FROM boliga2
WHERE price < 10000
and houseType = 'Villa'
-- 12 rows DELETED --

-- Deleting rows WHERE the number of squaremeter is bad data / below 10 --
DELETE FROM boliga2
WHERE sqmHouse < 10
-- 1350 rows DELETED --

-- Deleting rows WHERE zipcode is below 1000 since there shouldn't exist address's with
zipcode lower than 1000 --
DELETE FROM boliga2
WHERE zipcode < 1000
-- 314 rows DELETED --

-- Deleting rows WHERE rooms is below 1 since there shouldn't exist address's with zero
rooms --
DELETE FROM boliga2
WHERE rooms < 1
-- 2279 rows DELETED --

DELETE FROM Dataset6NoBasement
WHERE len(price) > 8

-----------------------------------------------

DELETE FROM Dataset6NoBasement
WHERE SqmHouse = 0
DELETE FROM Dataset6NoBasement
WHERE Price = 0
DELETE FROM Dataset6NoBasement
WHERE TypeOfSale not like '%1%'
-- 2279 rows DELETed --
DELETE FROM Dataset6NoBasement
WHERE price < 10000
DELETE FROM Dataset6NoBasement
WHERE sqmhouse < 12
price like ''
SqmHouse = 0
SqmHouse like ''
SalesDate like ''
TypeEnum = 0
TypeEnum like ''
Zipcode like ''
DELETE FROM Dataset6NoBasement
WHERE sqmHousePrice < 1000
--46438 rows--
DELETE FROM Dataset6NoBasement
WHERE sqmHousePrice > 10000
--4261--
DELETE FROM Dataset6NoBasement
WHERE SalesDate < 19920101
--29--
DELETE FROM indexfinal
WHERE zipcode < 2000
and sqmhouseprice < 20000
DELETE FROM indexfinal
WHERE zipcode < 2000
and Indexannually > 100000
Appendix 4 – Sql Query Deleting Duplpicates

DELETE FROM CO42000T
WHERE Ophoert_ts not like '' -- 91999--
DELETE FROM CO42100T
WHERE Ophoert_ts not like '' -- 97168 --
DELETE FROM CO40200T
WHERE Ophoert_ts not like '' -- 189790 --
DELETE FROM CO40100T
WHERE Ophoert_ts not like '' -- 747435 --
DELETE FROM CO43200T
WHERE Ophoert_ts not like '' -- 46524 --
DELETE FROM CO40400T
WHERE Ophoert_ts not like '' -- 380050 --
DELETE FROM CO40500T
WHERE Ophoert_ts not like '' -- 255532 --
-- 3 rows DELETED --
-- Deleting rows with house types we dont use--
SELECT count (*) FROM CO40400T
WHERE ENH_ANVEND_KODE like '120' --1114168--
SELECT count (*) FROM CO40400T
WHERE ENH_ANVEND_KODE like '130' --418607--
SELECT count (*) FROM CO40400T
WHERE ENH_ANVEND_KODE like '131' --2326--
SELECT count (*) FROM CO40400T
WHERE ENH_ANVEND_KODE like '132' --696--
SELECT count (*) FROM CO40400T
WHERE ENH_ANVEND_KODE like '110' --1116585--
SELECT * FROM CO40400T--
BEGIN tran
DELETE FROM CO40400T
WHERE ENH_ANVEND_KODE > 141 -- DELETED 544512 rows --
SELECT count (*) FROM CO40400T
ROLLBACK tran
SELECT count (*) FROM CO40400T

BEGIN tran
DELETE FROM CO40400T
WHERE ENH_ANVEND_KODE < 100 -- DELETED 56325 rows --
SELECT count (*) FROM CO40400T
ROLLBACK tran
SELECT count (*) FROM CO40400T

SELECT count (*) FROM CO40100T
BEGIN tran
DELETE FROM CO40100T
WHERE BYG_ANVEND_KODE > 141 -- DELETED 2908951 rows --
SELECT count (*) FROM CO40100T
ROLLBACK tran
SELECT count (*) FROM CO40100T

SELECT count (*) FROM CO40100T
BEGIN tran
DELETE FROM CO40100T
WHERE BYG_ANVEND_KODE < 100 -- DELETED 2908951 rows --
SELECT count (*) FROM CO40100T
ROLLBACK tran
SELECT count (*) FROM CO40100T

-----------------------------------------------------------------------------------
SELECT count (*) FROM CO40400T
BEGIN tran
SELECT * into CO40400T1
FROM (SELECT row_number() over (partition by enhadr_id order by crud_id) as rownr ,*
FROM CO40400T) as bla
WHERE rownr =1
SELECT count (*)FROM CO40400T
ROLLBACK tran
SELECT count (*)FROM CO40400T
---------------------------------------------------------------------------------------
SELECT count (*)FROM CO40100T
BEGIN tran
SELECT * into CO40100T1
FROM (SELECT row_number() over (partition by AdgAdr_id order by crud_id) as rownr ,*
FROM CO40100T) as bla
WHERE rownr =1
SELECT count (*)FROM CO40100T1
ROLLBACK tran
SELECT count (*)FROM CO40100T1
--------------------------------------------------------------------------------------
SELECT count (*)FROM CO40500T
BEGIN tran
DELETE FROM CO40500T WHERE ObjStatus = 1 and NyByg = 1 and SamletAreal = 0

SELECT count (*) FROM CO40500T

ROLLBACK tran

SELECT count (*) FROM CO40500T
Appendix 5 – Sql Query Changing data types

SELECT TOP 1000 *
FROM boliga2

--Setting price datatype as INT
ALTER TABLE boliga2
ALTER COLUMN price INT;

--Setting expences datatype as INT
ALTER TABLE boliga2
ALTER COLUMN expences INT;

--Setting rooms datatype as INT
ALTER TABLE boliga2
ALTER COLUMN rooms INT;

-- Setting sqmHouse datatype as INT
ALTER TABLE boliga2
ALTER COLUMN sqmHouse INT;

--Setting sqmArea datatype as INT
ALTER TABLE boliga2
ALTER COLUMN sqmArea INT;

--Setting changes datatype as INT
ALTER TABLE boliga2
ALTER COLUMN changes INT;

--Setting price datatype as INT
ALTER TABLE Dataset6NoBasement
ALTER COLUMN Price INT;

--Setting price datatype as INT
ALTER TABLE Dataset6NoBasement
ALTER COLUMN Zipcode varchar(4);
Appendix 6 – Sql Query Cleaning the text in data

-- price attribute to be only numbers, no “kr.” and “.”
BEGIN tran
UPDATE boliga2
SET price = REPLACE(price, '.', '')
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2
BEGIN tran
UPDATE boliga2
SET price = REPLACE(price, 'kr', '')
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2
--Cleaning expences attribute to be only numbers, no kr. and .
BEGIN tran
UPDATE boliga2
SET expences = REPLACE(expences, '.', '')
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2
BEGIN tran
UPDATE boliga2
SET expences = REPLACE(expences, 'kr', '')
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2

REM o - Removing m2 FROM sqmHouse
BEGIN tran
UPDATE boliga2
SET sqmHouse = REPLACE(sqmHouse, 'm2', '')
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2

REM o - Removing m2 FROM sqmArea
BEGIN tran
UPDATE boliga2
SET sqmArea = REPLACE(sqmArea, 'm2', '')
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2

REM o - Removing % FROM changes
BEGIN tran
UPDATE boliga2
SET changes = REPLACE(changes, '%', '')
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2
--Removing ',' FROM address
BEGIN tran
UPDATE boliga2 SET address=LEFT(address, LEN(address)-1)
SELECT TOP 100 *
FROM boliga2
ROLLBACK tran
SELECT TOP 100 *
FROM boliga2
--Removing ' 00:00:00' FROM SalesDate
BEGIN tran
UPDATE Dataset6NoBasement SET SalesDate=LEFT(SalesDate, LEN(SalesDate)-9)
SELECT TOP 100 *
FROM Dataset6NoBasement
ROLLBACK tran
SELECT TOP 100 *
FROM Dataset6NoBasement
--Removing both '-' FROM SalesDate
BEGIN tran
UPDATE Dataset6NoBasement
SET SalesDate = REPLACE(SalesDate, '-', '')
SELECT TOP 10 *
FROM Dataset6NoBasement
ROLLBACK tran
SELECT TOP 10 *
FROM Dataset6NoBasement
Appendix 7 – Sql Query Creating Column

--Creating sqmHousePrice column and setting data type = int--
ALTER TABLE boliga2
ADD spmHousePrice int;

--Creating periodOfSale column and setting data type = nvarchar(50)--
ALTER TABLE boliga2
ADD periodOfSale nvarchar(50);

--Creating houseAge column and setting data type = int--
ALTER TABLE boliga2
ADD houseAge int;

--Creating zipcode column and setting data type = int--
ALTER TABLE boliga2
ADD zipcode int;

--Creating sqmArea column and setting data type = int--
ALTER TABLE boliga2
ALTER COLUMN sqmArea INT;

--Creating streetname column and setting data type = nvarchar(50)--
ALTER TABLE boliga2
ADD streetName nvarchar(50);

--Creating periodOfSaleNumber column and setting data type = int--
ALTER TABLE boliga2
ADD periodOfSaleNumber int;

--Creating periodOfSaleYear column and setting data type = int--
ALTER TABLE boliga2
ADD periodOfSaleYear int;

--Creating sqmHousePrice column and setting data type = int--
ALTER TABLE Dataset3NoBasement
ADD sqmHousePrice int;

--Creating MarketIndex column and setting data type = int--
ALTER TABLE Dataset3NoBasement
ADD MarketIndex int;

--Creating MarketIndex column and setting data type = int--

ALTER TABLE CO15900T
ADD OVERDRAGELSESES_KODE Char(1);
Appendix 8 – Sql Query Joining KMD data

SELECT

    CO42100T.EnhAdr_id as 'ID',
    CO42000T.VEJ_NAVN as 'StreetName',
    CO11500T.HUS_NR as 'HouseNumber',
    CO42100T.SIDE_DOERNR as 'DoorSide',
    CO42100T.Etagebetegn as 'Floor',
    CO42000T.PostByNavn as 'City',
    CO42000T.PostNr as 'Zipcode',
    CO43200T.KoorNord as 'CoordinateNorthing',
    CO43200T.KoorOest as 'CoordinateEasting',
    CO40400T1.VAERELSE_ANT as 'Rooms',
    CO42000T.KomKode as 'MunicipalityNumber',
    --CO40200T.Elevator,
    CO40400T1.ENH_ANVEND_KODE as 'TypeOfHouse',
    CO40100T1.OPFOERELSE_AAR as 'BuildYear',
    CO40100T1.OMBYG_AAR as 'RebuildYear',
    CO40400T1.BEBO_ARL as 'SqmHouse',
    --CO40500T.SamletAreal as 'SqmBasement',
    --CO40500T.KaeldAr1LovigBebo as 'LegalSqmBasement',
    CO11500T.EJD_MATR_ARL_SAML as 'SqmArea',
    CO15900T.OMREGNINGS_DATO as 'SalesDate',
    CO15900T.KOEBESUM_BELOEB as 'Price'

INTO Dataset3NoBasement

FROM  CO42000T CO42000T

    INNER JOIN CO11500T CO11500T ON

    CO11500T.VEJ_KODE = CO42000T.Vejkode
AND CO11500T.HUS_NR = CO42000T.HUS_NR
AND CO11500T.KOMMUNE_NR = CO42000T.KomKode
INNER JOIN CO42100T CO42100T ON
CO42100T.Etagebetegn = CO11500T.ETAGE
AND CO42100T.SIDE_DOERNR = CO11500T.SIDE_DOERNR
AND CO42100T.AdgAdr_id = CO42000T.AdgAdr_id
INNER JOIN CO40100T1 CO40100T1 ON
CO40100T1.AdgAdr_id = CO42000T.AdgAdr_id
INNER JOIN CO43200T CO43200T ON
CO43200T.Adressepunkt_id = CO42000T.Adressepunkt_id
INNER JOIN CO40400T1 CO40400T1 ON
CO40400T1.EnhAdr_id = CO42100T.EnhAdr_id
INNER JOIN CO15900T CO15900T ON
CO15900T.KOMMUNE_NR = CO11500T.KOMMUNE_NR
AND CO15900T.EJD_NR = CO11500T.EJD_NR
Appendix 9 – Sql query calculating baseline average square meter price

```
SELECT
    AVG(sqmHousePrice)
FROM
    dataset6nobasement
WHERE
    YEAR(SalesDate) = 2015 -- Only sales year 2015
    AND TypeOfSale = 1 -- Normal Sale
    AND TypeOfHouse = 120 -- Only houses
```

Appendix 10 – Sql Query calculating baseline average index square meter price

```
SELECT
    AVG(indexpriceresult) as 'AvgIndexSqmPrice'
FROM
    indexmodel18
```

Appendix 11 – Sql Query calculating the average square meter price in zipcodes

```
WITH temp AS (  
SELECT
    SUBSTRING(zipcode,1,2) firstdigitsZipcode,
    AVG(sqmHousePrice) AS avgSqmZip -- Calculating the average square meter price in a zipcode in 2015
```
FROM dataset6nobase

WHERE

  YEAR(SalesDate) IN ( '2015'
  AND TipoH = 120

GROUP BY

  SUBSTRING(zipcode,1,2)

)

SELECT -- Joining the average square meter price in a zipcode for 2015 on sales in 2016
  id,
  zipcode,
  price,
  sqmhouseprice,
  sqmHouse,
  avg(sqmHousePrice) OVER (PARTITION BY SUBSTRING(zipcode,1,2),YEAR(SalesDate)) AS avgSqmZip,
  YEAR(SalesDate) AS SdYear,
  temp.avgSqmZip

FROM dataset6nobase

INNER JOIN

  temp ON firstdigitsZipcode = SUBSTRING(zipcode,1,2)

WHERE

  YEAR(SalesDate) IN ( '2016'
  AND TipoH = 120 -- Only Houses
Appendix 12 – Sql Query calculating our index model

WITH TEMP as (  
SELECT  
CASE  
    WHEN zipcode<2000 THEN --Only one index zipcode for inner Copenhagen  
( ((Price/SqmHouse)  
/  
(convert(float,avgM2PriceZipMonthInCPH) / FIRST_VALUE(avgM2PriceZipMonthInCPH)  
over (partition by substring(zipcode,1,1),TypeOfHouse  
    order by YEAR(SalesDate),MONTH(SalesDate))))  
*  
(FIRST_VALUE(avgM2PriceZipMonthInCPH)  
over (partition by substring(zipcode,1,1),TypeOfHouse  
    order by YEAR(SalesDate) desc,MONTH(SalesDate) desc)/convert(float,FIRST_VALUE(avgM2PriceZipMonthInCPH)  
over (partition by substring(zipcode,1,1),TypeOfHouse  
    order by YEAR(SalesDate),MONTH(SalesDate))))  
)  
else (  
( (price/SqmHouse)  
/  
(convert(float,avgM2PriceZipMonth) / FIRST_VALUE(avgM2PriceZipMonth)  
over (partition by substring(zipcode,1,3),TypeOfHouse  
    order by YEAR(SalesDate),MONTH(SalesDate)))))  

(FIRST_VALUE(avgM2PriceZipMonth)  
over (partition by substring(zipcode,1,3),TypeOfHouse  
    order by YEAR(SalesDate),MONTH(SalesDate)))  
)
order by YEAR(SalesDate) desc, MONTH(SalesDate) desc) / convert(float, FIRST_VALUE(avgM2PriceZipMonth)
    over (partition by substring(zipcode,1,3) ,TypeOfHouse
    order by YEAR(SalesDate), MONTH(SalesDate))
) )
)

END as IndexPriceResult
,*

FROM(
SELECT
    YEAR(SalesDate), MONTH(SalesDate), TypeOfHouse, substring(zipcode,1,2)) as avgM2PriceZipMonth
    , avg(price/SqmHouse) over (partition by
    YEAR(SalesDate), MONTH(SalesDate), TypeOfHouse, substring(zipcode,1,1)) as avgM2PriceZipMonthInCPH
    , *

FROM
    Dataset6NoBasement
WHERE
    TypeOfHouse = 120 -- Houses only
    AND SoldBefore IS NULL -- Only train on the train set
) as basen
)

SELECT * into indexModel18 FROM TEMP
Appendix 13 – Azure Machine learning model setup
Appendix 14 – Scored data example: Boosted Decision Tree Regression – Multiple attributes, 2015 training data, 2016 test data