UNIVERSITY OF COPENHAGEN

A Physicist's Approach To Machine Learning

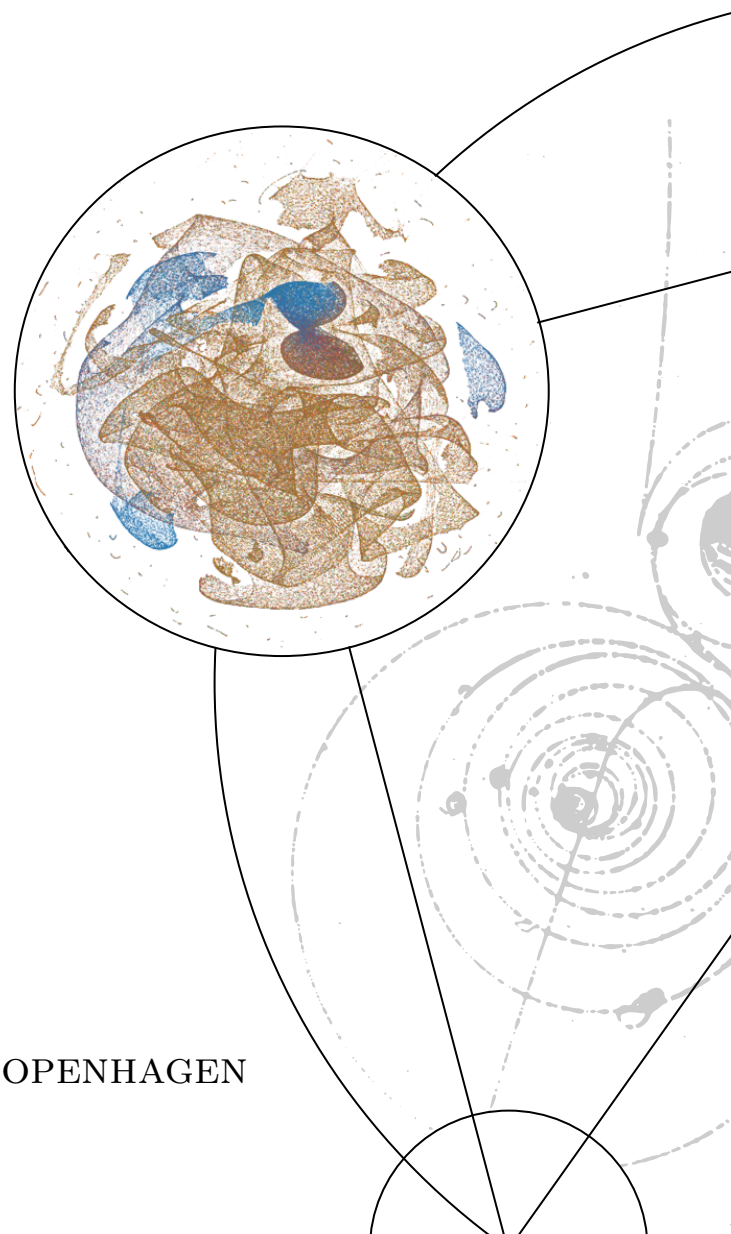
Understanding The Basic Bricks

Christian Michelsen

Master's Thesis (Cand. Scient.)

January 3rd, 2020

Supervised by
Troels Petersen



University of Copenhagen

https://github.com/christianmichelsen

This thesis was inspired by the works of Edward R. Tufte using the Tufte-LaTeX package.

First printing, December 2019

Abstract

Here will be a decent abstract at some point^{TM}.

Contents

Part I

Part I of this thesis covers the introductory theory of machine learning in chapter 2 along with some extra technical aspects of it. In chapter 3 machine learning is applied to estimate Danish housing prices as precisely and accurately as possible.

3. *Danish Housing Prices*

"Buy land, they're not making it anymore."

— Mark Twain

HOUSING MARKETS have always been a playing field for economists, property speculators, real estate agents, and realtors. The author of this thesis is by no metric any of these, not even close to, yet, when the issue of estimating Danish housing prices came up, it was too much of a challenge just lying there to let it go. Estimating housing prices is a classical economical discipline as seen in papers from the Danish National Bank developing a regional model of the Danish housing market [58] to an analysis of the financial crisis in 2008-2009 and its effects on the Copenhagen metropolitan area [72].

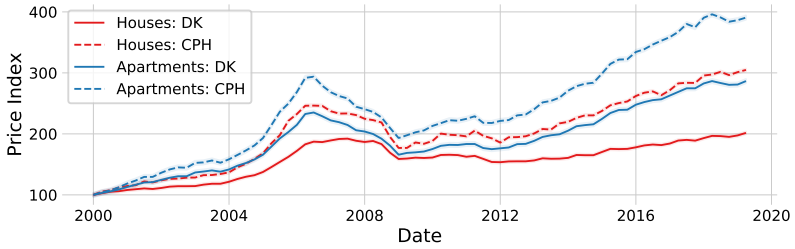


Figure 3.1: Price Index of the Danish housing market. Prince index of Danish one-family houses and owner-occupied apartments where houses are shown in red and apartments in blue, where full lines are for the entirety of Denmark and dashed lines are only for Copenhagen. Errorbars (scaled up with a factor of 2) are shown as colored bands. The price index and its uncertainty is based on numbers from DST [44], however, rescaled to 100 in 2000 (instead of 2006 as it was in the data).

If one takes a look at the time development of the Danish housing market, the Danish governmental organization for statistics, Statistics Denmark, releases a price index [44] for both one-family houses (OFH) and owner-occupied apartments (OOA) quarterly, see Figure 3.1. Here it is easy to see the effect of the financial crisis around 2008, but also the steady increase in the housing market in both Copenhagen and the entire country since then. Housing in this context means both actual houses and privately owned apartments, and will be called residences in general in this project.

The goal of this subproject is not to predict any future collapse as the financial markets as we saw upwards of 10 years ago. Instead, it is to learn patterns in the price of houses in steady times. The goal is training a computer to automatically find these patterns and see if we can improve this model[1].

In section 3.1 the data will be introduced and feature augmented in section 3.2. The evaluation functions will be discussed in section 3.3 and the choice of loss function decided in section 3.4. The

[1] In contrary to Hviid [58], Mulalic et al. [72] and others who base their models on macro-economic principles.

model is fitted and optimized in section 3.5 and the results presented in section 3.6. Finally the model will be further understood in section 3.7, some additional models presented in section 3.8 and at last the models will be discussed in section 3.9.

3.1 *Data Preparation and Exploratory Data Analysis*

> *"80 % of data science is cleaning the data and 20 % is complaining about cleaning the data."*
>
> — Anthony Goldbloom, Kaggle

The first part of any data science project is actually getting the data and being able to read it. This has been an iterative process that has improved over time. The last data transfer we got was September 3rd 2019 which consisted of a 522.4 MB CSV file with dimensions $(711\,212, 171)$. This section will go through the data cleaning process.

Before any further data analysis is performed, all of the data is loaded, except columns which only contain internal information for Boligsiden[2]. To get a better understanding of all of the variables, histograms showing the one-dimensional distributions of all of the variables were made.

[2] The variables `Sag_Kvhx`, `Enhed_GOP_BoligtypeKode`, and `Bygning_GOP_Matrikelnr`.

Four particularly interesting ones are seen in Figure 3.2: the distribution of the date of the sale `SalgsDato` in subfigure (a), the distribution of the type of residence `SagtypeNr` in subfigure (b), the distribution of the longitude of the residence `GisX_WGS84` in subfigure (c), and the distribution of the area of the residence `ArealBolig` in subfigure (d).

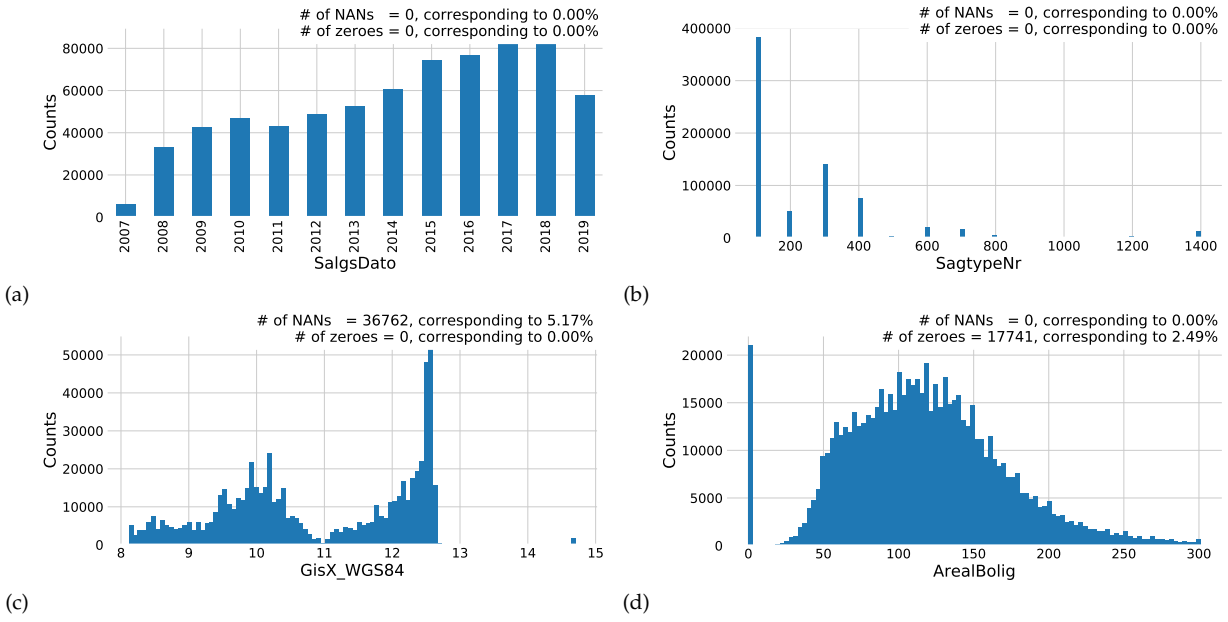


Figure 3.2: Distributions of four out of the 168 input variables. Subplot (a) shows the date of the sale, Subplot (b) shows the type of residence, Subplot (c) shows the longitude, Subplot (d) shows the area og the house.

The distribution of the date of sale, Figure 3.2 (a), is an interesting variable because it shows how Boligsiden has been collecting more

and more data over time. Here 2007 and 2019 are clear outliers since their current database only contains sales from the end of 2007, and 2019 only contains data from the first eight months of the year. The `SagTypeNr` is a discrete code that Boligsiden uses to differentiate between different types of residences. The mapping between code and description can be seen in Table 3.1.

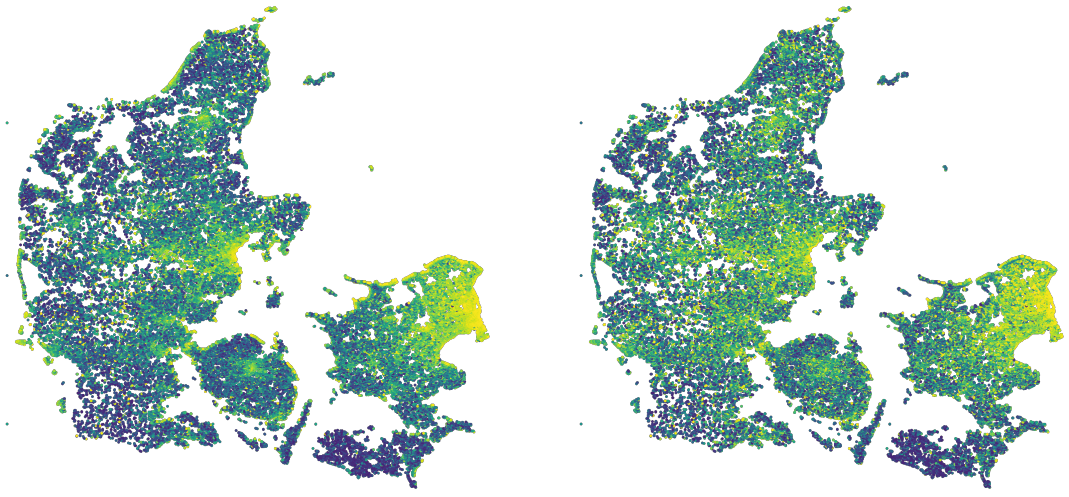
In this project only one-family houses – "Villas" in Danish – with code 100 and owner-occupied apartments – "Ejerlejlighed" in Danish – with code 300 are considered. As can be seen from Figure 3.2 (b) these two types of residences are also the most frequent sales with close to 400 000 and 150 000 sales in total. The longitude distribution, Figure 3.2 (c), is mostly interesting due to fact that it clearly shows how the Great Belt and especially the Baltic Sea separates Denmark into three parts; the Western part, the Eastern part, and then Bornholm. Note that more than 5 % of the residences' locations are unknown values, so-called "Not A Number"s (NANs). The distribution of the area, Figure 3.2 (d), shows that most residences are between 50 m^2 and 200 m^2, as expected in Denmark. However, a relatively large part of the residences, 2.5 %, are listed as having an area of 0 m^2 which are obviously erroneous entries. All of the 1D-distributions can be seen in Figure A.2–A.15.

The geographic distribution of sales can be seen in Figure 3.3. The residences are coloured according the square meter price in Figure 3.3 (a) and according to the sales price in Figure 3.3 (b). Notice the strong correlation between the distance to water and the square meter price, a correlation that is less visible when looking at the sales price. Since these plots each contain 674 647 points[3], over-plotting quickly becomes an issue. To circumvent this the software package called DataShader [8] was used which in a simple, consistent, and not at least computationally efficient manner allows one to plot big data.

Type	Name
100	Villa
200	Rækkehus
300	Ejerlejlighed
400	Fritidsbolig
401	Kolonihave
500	Andelsbolig
600	Landejendom
700	Helårsgrund
800	Fritidsgrund
900	Villalejlighed
1000	Kvæggård
1100	Svinegård
1200	Planteavlsgård
1300	Skovejendom
1400	Lystejendom
1500	Specialejendom

Table 3.1: Mapping between the code in `SagTypeNr` and the type of residence. The two important types of residences are villa (one-family houses) and ejerlejlighed (owner-occupied apartments).

[3] Only sales with a valid GPS-coordinate and area of residence are shown



(a) (b)

Figure 3.3: Geographic distribution of the sold residences. In subplot (a) the sales are colored according to their square meter price and in subplot (b) according to the sales price.

The most important of the features is the sales price, called `SalgsPris` in the dataset. Its distribution is shown in Figure 3.4. This is a pos-

itively skewed distribution that shares visual similarities with a log-normal distribution. The mode[4] of the all sales prices is 1.1 M.kr. and the median is 1.6 M.kr. The mean is 2.0 M.kr. but this value is heavily influenced by a few very high values.

[4] Measured in millions DKK, M.kr.

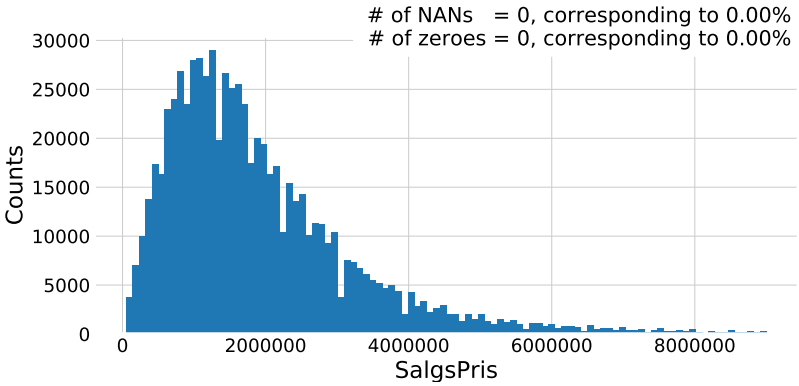


Figure 3.4: Histogram of prices of houses and apartments sold in Denmark.

3.1.1 *Correlations*

Having shown the 1D-distributions of all the different variables in the previous section, the next step would be to look at the correlations between the variables. Since there are 168 input variables, it is almost impossible to understand every inter-variable correlation, however, it is tried in Figure A.16. Here the correlation between all numerical variables that are not obviously related to other variables (like the GPS-coordinates that are in both latitude-longitude and ETRS89[5] format), and with the condition that it has to have one inter-variable correlation higher than $|\rho| > 30\,\%$, are plotted as a (86×86)-dimensional heatmap.

[5] European Terrestrial Reference System 1989.

Even though inter-variable correlations are important in the exploratory data analysis (EDA) phase, what is more important is to get a better understanding of how the input variables correlate with the output variable; the sales price. This is shown in Figure 3.5 for the variables where $|\rho| > 10\,\%$. It is the previous property evaluation, `EjdVurdering_EjendomsVaerdi` that is correlated the most with the sales price, which does not come as any huge surprise. Other positively correlated variables are the cost of ownership[6], area, number of bathrooms, its longitude, and distance to nearest wind mill. In the other end, the local income tax, `Kommune_SkatteProcent`, is the variable that is the most negatively correlated to the sales price, followed by the geographical variables related to province, municipality, and postcode.

[6] This a great example of the fact that correlation does not imply causation

The correlation ρ used above is the linear correlation which only captures linear relationships between variables. All modern machine learning algorithms, however, are also able to capture higher-order correlations and thus a higher-order correlation measure is needed. The maximal information coefficient (MIC), which is a value between 0 and 1, is such a non-linear measure. It finds correlation based on the intuition that if two variables are correlated, it should be possible

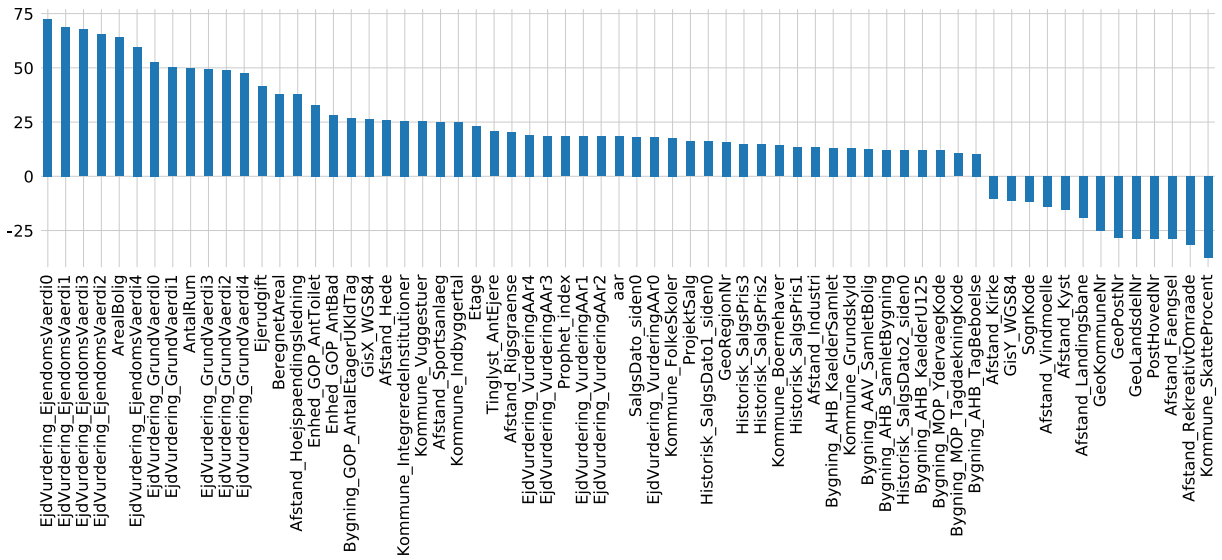


Figure 3.5: Linear correlation between variables and price for variables where the correlation coefficient ρ is $|\rho| > 10\,\%$.

to split the data up into smaller grids where, if they are correlated, the grid that contain points should contain many points and the rest of the grids should be (relatively) empty [79]. This is in comparison to two uncorrelated variables which would simply display noisy behavior and only have few grids with many points in. Albanese et al. [12] extended on this idea and developed the computationally efficient algorithm called MICtools which computes the estimator for MIC: MIC_e. An example of this non-linear correlation is seen in Figure 3.6. Here the relationship between the normal linear correlation ρ and MIC_e can be seen for four synthetic datasets. Notice that MIC_e does it particularly well for the sine wave, and decent for the line and parabola, but only slightly captures the relationship for the exponential growth. The influence of noise on ρ and MIC_e can be see in Figure A.17.

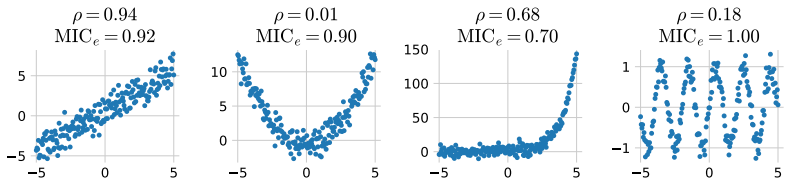


Figure 3.6: Comparison of the linear correlation ρ and the non-linear MIC for a straight line, a parabola, an exponential, and a sine wave, all with noise added. See also Figure A.17.

Using MIC_e as the correlation measure between the numerical variables and the sales prices, the variables with a MIC_e-score higher than 10 % can be seen in Figure 3.7. Again, the previous property evaluations are the most correlated features to the sales price followed by the parish code, `SogneKode`. In general the geographical variables score high here with also the post code, municipality number, and longitude.

3.1.2 *Validity of input variables*

The fact that some of the variables contains considerable amount of invalid values, NANs, requires this to be taken into account before

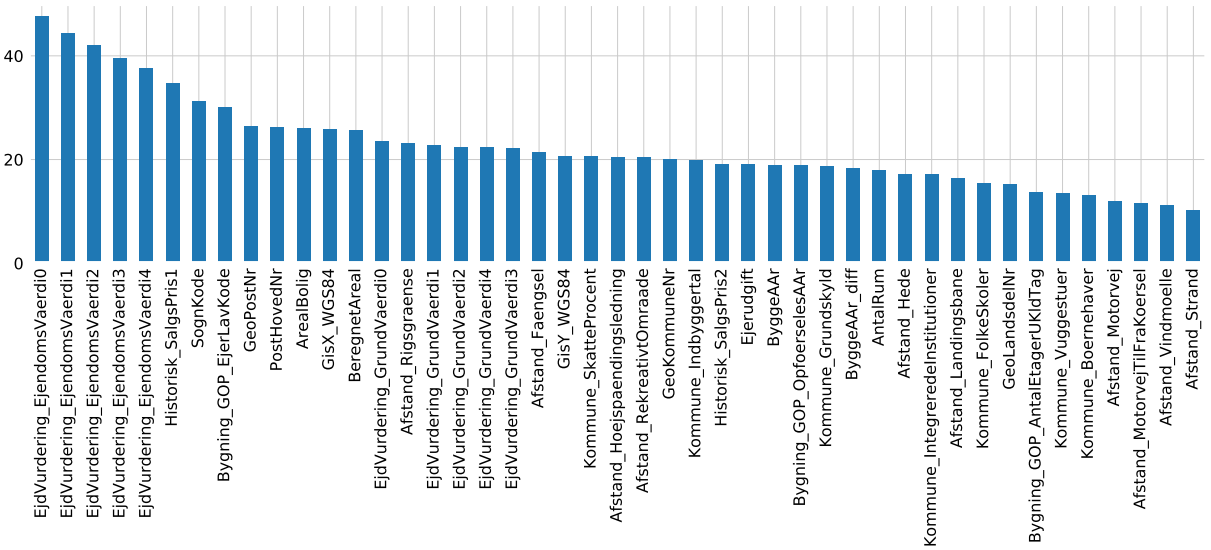


Figure 3.7: Non-linear correlation between variables and price using Maximal Information Coefficient (MIC) for variables where MIC $> 10\%$.

any further analysis. The validity, defined as the percentage of valid observations, of every variable is shown in Figure 3.8. Here the 168 variables variables are grouped together into 25 variables where each group share the same validity. An example of this are all of the 16 different distance-variables[7]. We see that most of the variables have validities around more than 85 %, however, a few of the variables, especially information about the building, `BygningsInfo`, have validities less than 20 %.

[7] Distance to: prison, heath, high-voltage transmission line, industri, visible railroad, church, churchyard, coast, landing strip, motorway, access to motorway, recreational area, border, sports centre, beach, and windmill.

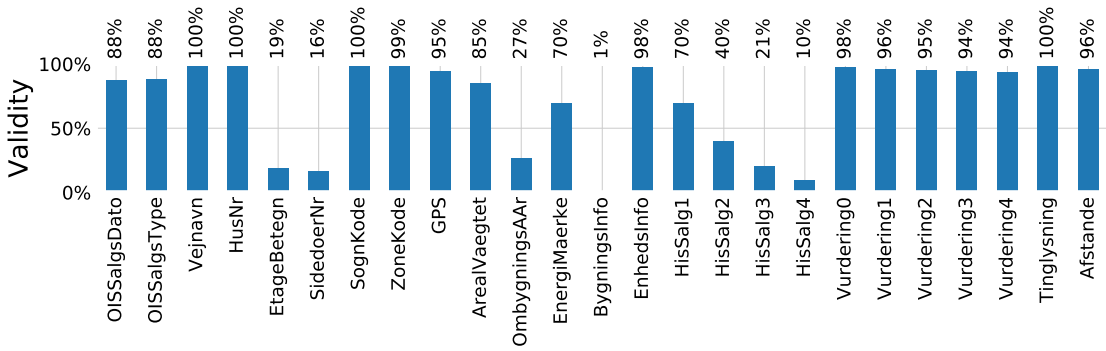


Figure 3.8: Percentage of valid counts for each variable grouped together in categories.

To see how closely related the different validity groups are, one can look at the dendrogram in Figure 3.9. The dendrogram is based on a hierarchical clustering algorithm [98] where the different groups are clustered according to the linear correlation of their validity. This diagram is supposed to be read in a top-down approach, where it can be see that the name of the street, `Vejnavn`, and the number of the residence, `HusNr`, correlate a lot and are thus clustered very early. The year of the last time the residence was rebuilt or greatly modified, `OmbygningsAAr`, does not correlate strongly with any of the other variables and is thus the last variable to be clustered together.

To see a heatmap of the inter-variable correlations, see Figure A.1.

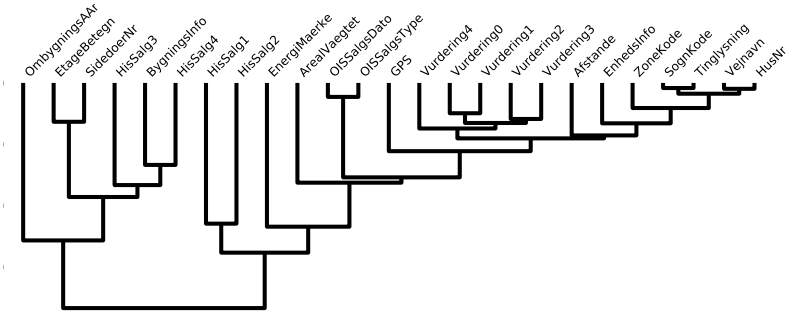


Figure 3.9: Validity Dendrogram based on hierarchical clustering of the linear correlation of validity for the housing price variables clustered together.

3.1.3 *Cuts*

Given the 1D input variable distributions and their validity, we apply some very basic cuts before any further analysis. These cuts are seen in Table 3.2. Sales type, `OISSalgsType`, is a OIS[8] code which describes what type of sale it is: when it is 1 it is considered a normal sale, compared to e.g. forced sales. These cuts are seen as the minimum requirements for what constitutes a curated dataset with no obvious outliers. The reason why the time requirement is applied is to reduce the effect of the financial crisis to creep into the model.

[8] OIS is short for "Den Offentlige Informationsserver", the Danish public information server, and it collects information about Danish residences [9].

	Description	Remaining	Removed
Area	$20\,\mathrm{m}^2 \leq$ Area $\leq 500\,\mathrm{m}^2$	689 140	23 666
Price	0.1 M.kr. $\leq$ Price $\leq$ 100 M.kr.	687 546	1 594
Type	Has sales type 1	605 415	82 131
GPS	Has valid GPS coordinates	578 860	26 555
Private	Only non-business sales	549 140	29 720
Time	Sold in 2009 or later	520 548	28 592

Table 3.2: Overview of the basic cuts which define the minimum information needed to predict the price of a sale.

3.2 *Feature Augmentation*

Until now the analysis have dealt with different types of residences all together. From now on, the rest of the analysis will be applied on single-family houses and owner-occupied apartments independently.

First invalid counts are dropped such that variables which contain more than 10 % NANs are dropped, and duplicate rows are also removed. Then some manual features are added based on existing features. The day of the month, the month, and the year are extracted from the sales date and the sales date is also represented as the numbers of days since January 1st, 2009. From the number of the house, `HusNr`, the number is extracted along with a boolean flag indicating whether or not it includes a letter (eg. "27B"). The number of the side door, `SidedoerNr`, is formatted according to Table 3.3 and the road name is according to Table 3.4. The age of the house is added[9] and the amount of time (in years) since last major modification. The energy rating label, `EnergiMaerke`, is also converted from strings to values according to Table A.2.

Finally, some of the variables in the dataset are not suitable for

String	Explanation	Code
NAN	No side door	0
TH, TV	Right, left	11
MF	Center	12
–	The rest	15

Table 3.3: Side door mapping. If the side door string contains e.g. "TH" this gets the code 11.

Contains	Explanation	Code
Vej	Road	0
Gade	Street	1
Alle, Allé	Avenue	2
Boulevard	Boulevard	3
–	The rest	−1

Table 3.4: Street mapping. If the street name contains e.g. "Vej" this gets the code 0.

[9] In addition to only having the year the house was was built.

machine learning or simply transformations of other variables. These are variables such as the ID, when the house was deleted at Boligsiden, or the cash price. They are dropped since we do not want the model to learn the price of the residence using these variables.

3.2.1 *Time-Dependent Price Index*

In addition to the manual data augmentation in section 3.2, a time-dependent price index is also added. We make use of the open source package called Prophet made by Taylor and Letham [87] at Facebook. It is based on a decomposable time series model [53] with two[10] components; trend $g(t)$ and seasonality $s(t)$:

$$y_{\text{Prophet}}(t) = g(t) + s(t) + \epsilon_t, \tag{3.1}$$

where ϵ_t is a normally distributed error term. Taylor and Letham [87] fit this equation with a generalized additive model (GAM) [54] which they argue has several practical advantages compared to ARIMA [11] models which commonly used in economics [71].

[10] In their paper, Taylor and Letham [87] include a holiday component in their analysis as well which is not included in this project.

[11] AutoRegressive Integrated Moving Average.

We fit the Prophet model on the weekly median price pr. square meter (PPSM) up until (and including) 2017. The results of the prophet model fitted on (owner-occupied) apartments are seen in Figure 3.10 and Figure 3.11. In Figure 3.10 the weakly median price pr. square meter for apartments are shown as black dots with the fitted Prophet model shown in blue. The 1σ uncertainty intervals are shown as the transparent blue band. The Prophet model not only allows predicting previous and future PPSMs, it also return the uncertainty of this prediction. The future predictions for the PPSM are the values after 2018. The trend and seasonality of the model are shown in Figure 3.11 where the top plot is the overall trend $g(t)$ and the bottom plot is the seasonality $s(t)$. Whereas the trend just continues to rise, the seasonality shows that residences are generally sold for a higher price in the Summer months compared to the Winter months. The Prophet model plots for one-family houses can be seen in Figure A.18 and A.19.

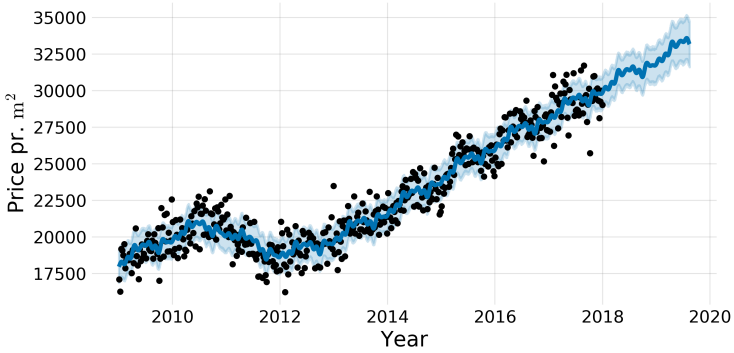


Figure 3.10: The predictions of the Facebook Prophet model trained on square meter prices for owner-occupied apartments sold before January 1st, 2018. The data is down-sampled to weekly bins where the median of each week is used as in input to the Prophet model. This can be seen as black dots in the figure. The model's forecasts for 2018 and 2019 are shown in blue with a light blue error band showing the 1σ confidence interval.

Using the Prophet model, we define the price index (PI) to be the Prophet-predicted PPSM, y_{Prophet}, for each residence normalized by

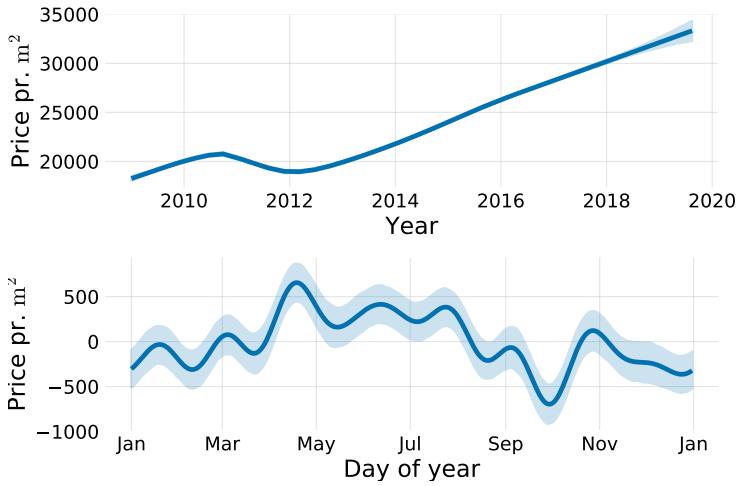


Figure 3.11: The trends of the Facebook Prophet model trained on square meter prices for owner-occupied apartments sold before January 1st, 2018. In the top plot is the overall trend as a function of year and in the bottom plot is the yearly variation as a function of day of year. It can be seen that the square meter price is higher during the Summer months compared to the Winter months, however, compared to the overall trend this effect is minor ($< 10\%$).

the mean to give values around 1:

$$\mathrm{PI}(t) = \frac{y_{\mathrm{Prophet}}(t)}{\langle y_{\mathrm{Prophet}}(t) \rangle}, \tag{3.2}$$

where $\langle \cdot \rangle$ refers to the average. The price index thus works as a measure of the national price for houses or apartments at a given time and is added as a variable to the dataset.

3.3 *Evaluation Function*

The choice of evaluation function f_{eval} is an important decision. The evaluation function will be based on the relative prediction z:

$$z = \frac{\hat{y} - y}{y}, \tag{3.3}$$

where y is true price and $\hat{y}$ the predicted one. The relative prediction is defined such that it is positive when $\hat{y} > y$, due to the outlier cuts made earlier the denominator is made sure to always be positive (and never 0), and $\mathbf{z}$ is expected to approximately follow a normal distribution[12]. Initially the mean of $\mathbf{z}$ was considered as the choice of evaluation function $f_{\mathrm{eval}} \stackrel{?}{=} \mathrm{mean}(\mathbf{z})$ though this only ensures a minimum of bias, not necessarily a low spread. This lead the discussion on to look at the standard deviation of z as $f_{\mathrm{eval}} \stackrel{?}{=} \mathrm{std}(\mathbf{z})$. The mean and the standard deviation, however, are not a very robust estimators since they are heavily influenced by outliers. The mean (and thus also the standard deviation) has a *asymptotic breakdown point* at 0 %, where the breakdown point is defined as the smallest fraction[13] of bad observations that can cause an estimator to become arbitrarily small or large: a single outlier with an arbitrarily large value may cause the mean to diverge to that large value [57]. In comparison, the median has an asymptotic breakdown point of 50 % and is thus a more robust estimator of centrality. A robust measure of the variability or dispersion of a sample $\mathbf{x}$ – compared to e.g. the standard

[12] Where $\mathbf{z}$ is the vector of all relative predictions $\mathbf{z} \in \mathbb{R}^N$.

[13] Where the "asymptotic" in "asymptotic breakdown point" refers to when the number of samples goes to infinity.

deviation σ – is the median absolute deviation (MAD) written as:

$$\mathrm{MAD}(\mathbf{x}) = c \cdot \mathrm{median}\left(|\mathbf{x} - \mathrm{median}(\mathbf{x})|\right), \quad c = \frac{1}{\Phi^{-1}(\frac{3}{4})}, \tag{3.4}$$

where c is a normalization constant to make MAD a consistent estimator of the standard deviation σ assuming normally distributed data and Φ^{-1} is the percent point function[14] [65]. The MAD is thus the median of the absolute differences between the data and the median of the data. We are, however, not just interested in having the distribution of $\mathbf{z}$ as narrow as possible, we also want it centered around 0. We thus continue with the following evaluation function:

$$\begin{aligned} \mathrm{MAD}_0(\mathbf{x}) &\equiv c \cdot \mathrm{median}\left(|\mathbf{x} - 0|\right) = c \cdot \mathrm{median}\left(|\mathbf{x}|\right) \\ f_{\mathrm{eval}}(\mathbf{z}) &\equiv \mathrm{MAD}_0(\mathbf{z}) = c \cdot \mathrm{median}\left(|\mathbf{z}|\right). \end{aligned} \tag{3.5}$$

To get an intuition about the size of a "good" value of MAD_0, one could calculate it comparing the asking price with the actual sales price. Doing so, one finds: $f_{\mathrm{eval}}(\mathbf{z}_{\mathrm{OFH}}) = 11.35\,\%$ for houses and $f_{\mathrm{eval}}(\mathbf{z}_{\mathrm{OOA}}) = 5.72\,\%$ for apartments. In some cases f_{eval} will still be referred to as MAD and it will be mentioned explicably if the form in equation (3.4) is meant.

[14] Inverse of the cumulative distribution function.

The MAD is assuming symmetric distributions and thus non-symmetric robust measures of the variability of a sample have been developed. See Rousseeuw and Croux [81] for more details.

3.4 *Initial Hyperparameter Optimization*

With the initial cleaning and feature adding done, the shapes of the ML-ready datasets are: (291 317, 144) for houses (OFHs) and (114 166, 144) for apartments (OOA), both sharing the same variables. All of the variables which are used from this point on can be seen in Table A.1. The data are split intro training and test sets such that training is defined as every sale from before 2018, every sale from 2018 is the test set, and since more data came after the project started 2019 is a small extra test set.

The number of observations for the different sets can be seen in Table 3.5. Since the dataset has been shown to be quite noisy with a lot of invalid counts a *tight* selection of the data is also applied. The tight selection is defined as residences which are within the 1 % to 99 % quantiles of all[15] numeric variables with more than 3 unique values. The number of observations for the different tight sets can be seen in Table 3.6.

A small study into the effect of some various hyperparameters was performed before any further fitting. This study investigated the effect of the old sales by assigning them a lower weight depending on time. It was investigated whether or not the model would perform better if samples got the time-dependent weight $w(t)$ given by:

$$\begin{aligned} w'(t) &= e^{k \cdot t}, \quad k = \frac{\log 2}{T_{\frac{1}{2}}}, \\ w(t) &= \frac{w'(t)}{\langle w'(t) \rangle}, \end{aligned} \tag{3.6}$$

	Houses	Apartments
Train	240 070	93 115
Test	34 628	14 183
2019	16 619	6868

Table 3.5: Number of observations for houses and apartments in the training, test, and 2019 set.

Tight	Houses	Apartments
Train	143 179	57 795
Test	20 338	8376
2019	9683	4030

Table 3.6: Number of observations for houses and apartments in the training, test, and 2019 set for the tight selection.

[15] Except the variables that contains the words: "aar" (year), "dato" (date), and "prophet".

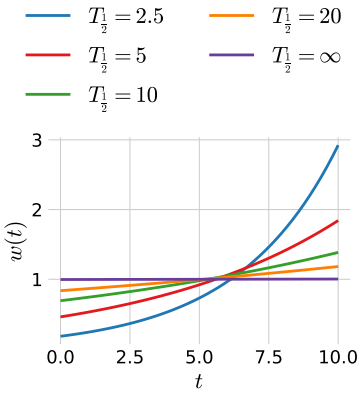


Figure 3.12: The sample weight $w(t)$ as a function of time t where the time is in years after January 1st, 2009, Here seen plotted for different values of the half-life $T_{\frac{1}{2}}$.

where $T_{\frac{1}{2}}$ is the half-life. This is illustrated in Figure 3.12 for the different values of $T_{\frac{1}{2}}$ used in the study. In addition to the weight, it was also investigated whether or not a $\log_{10}$-transformation of the price would increase performance. The reasoning behind this would be that some machine learning methods assume that the dependent variable, y, is normally distributed. Finally the choice of loss function was also added to the study for the five different loss functions defined in section 2.5. A grid search[16] was performed for:

$$\begin{aligned} T_{\frac{1}{2}} &\in \{2.5, 5, 10, 20, \infty\} \text{ years} \\ \log_{10} &\in \{\text{True, False}\} \\ \ell &\in \{\ell_{\text{Cauchy}}, \ell_{\text{Fair}}, \ell_{\text{LogCosh}}, \ell_{\text{SE}}, \ell_{\text{Welsch}}\}. \end{aligned} \tag{3.7}$$

The grid search is run on the training set with 5-fold cross validation, early stopping is applied with a patience of 100, and XGBoost [36] (XGB) is used as the ML model. For apartments the loss function with the lowest f_{eval} was the Cauchy loss with $T_{\frac{1}{2}} = 10$ years and no $\log_{10}$-transformation. This BDT terminated by early stopping after 770 trees, see Table 3.7. For houses the loss function with the lowest f_{eval} was (also) the Cauchy loss with $T_{\frac{1}{2}} = 20$ years and (also) no $\log_{10}$-transformation. This BDT terminated by early stopping after 1514 trees, see Table 3.8. It is interesting to note that both HPO for houses and apartments choose the same loss function and not to do any transformation.

The reason why the $\log_{10}$-transformation was included even to begin with, was that initially it showed better results than no transformation, however, this turned out to be a numerical consequence which was alleviated by dividing all prices with a million, such that y had units of M.kr. instead of kr.. All of the results for the apartments can be seen in Table A.3–A.7 along with all of results for the houses in Table A.8–A.12.

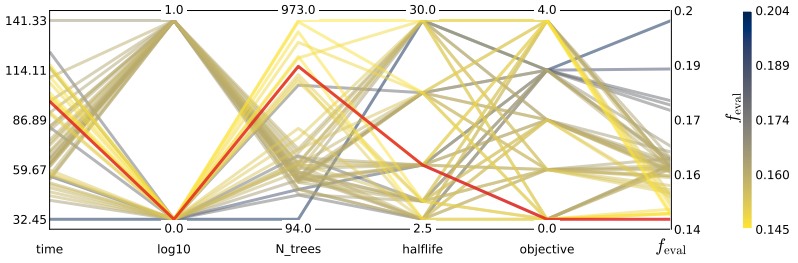


Figure 3.13: Hyperparameter optimization results for apartments. The results are shown as parallel coordinates with each hyperparameter along the x-axis and the value of that parameter on the y-axis. Each line is an iteration of the RS HPO colored according to the performance of that hyperparameter as measured by the MAD from highest in dark blue to lowest (best) in yellow. The single best hyperparameter is shown in red. For the hyperparameter `log10` `0` means False and `1` means True, for `Halftime` ∞ is mapped to `30`, and for `objective` the functions Cauchy (0), Fair (1), LogCosh (2) SquaredError (3), and Welsch (4) are mapped to the integers in the parentheses.

A visualization of the HPO results can be seen as the parallel coordinate plot in Figure 3.13. Here the hyperparameters (along the time taken and the number of trees) of the HPO are plotted along the abscissa (x-axis) and the value of the hyperparameter on the ordinate (y-axis). Every iteration of the HPO is thus a line on the plot. The lines are colored according to their evaluation score; the best hyperparameter is shown in red. For the hyperparameter `log10` `0` means False and `1` means True, for `Halftime` ∞ is mapped to `30`, and for `objektive` the functions Cauchy (0), Fair (1), LogCosh (2)

[16] Grid search was acceptable since the parameter space is small and two of the three dimensions are non-numerical.

Half-life	$\log_{10}$	N_{trees}	f_{eval}
2.5	True	293	0.1598
2.5	False	814	0.1466
5	True	304	0.1610
5	False	923	0.1468
10	True	266	0.1610
10	**False**	**770**	**0.1450**
20	True	288	0.1613
20	False	967	0.1467
∞	True	340	0.1601
∞	False	807	0.1480

Table 3.7: Results of the initial hyperparameter optimization for apartments for the best loss function ℓ_{Cauchy}.

Half-life	$\log_{10}$	N_{trees}	f_{eval}
2.5	True	434	0.1991
2.5	False	1007	0.1872
5	True	350	0.1999
5	False	1130	0.1858
10	True	436	0.1992
10	False	1183	0.1850
20	True	397	0.2003
20	**False**	**1514**	**0.1833**
∞	True	449	0.1992
∞	False	1351	0.1844

Table 3.8: Results of the initial hyperparameter optimization for houses for the best loss function ℓ_{Cauchy}.

SquaredError (3), and Welsch (4) are mapped to the integers in the parentheses. The same plot for houses can be seen in Figure A.20.

What can be concluded from Figure 3.13 is that there is a clear preference to not log-transform the data, that the BDTs with many trees[17] generally performed better than the ones with fewer trees, that it is difficult to see a clear pattern for the half-life, and that there seems to be a tendency for the Cauchy loss to be the best, however, it is still ambiguous. What is not seen in the figure, however, are how the uncertainties of the different iterations also matter, where the uncertainties are the standard deviations (not of the mean) of the 5 folds in the cross validation.

For the half-life and the choice of objective function these can be seen in Figure 3.14 and 3.15. In the first of the two figures it is easily seen that even though $T_{\frac{1}{2}} = 10\,\text{yr}$ is the minimum, the uncertainties are so large that nothing can be concluded definitely with regards to the half-life parameter related to the weights $w(t)$. In contrary, in the second figure there is a clear performance difference between the different loss functions where the Cauchy loss archives the best (lowest) value of f_{eval} and especially the Squared Error is disregarded.

The rest of the machine learning models thus continue with the following hyperparameters:

	$\log_{10}$	Half-life	Loss function
Apartments	False	10 yr	Cauchy
Houses	False	20 yr	Cauchy

3.5 *Hyperparameter Optimization*

With the initial HPO set, the actual training of the models was started. An XGBoost model was fitted to the each of the two training sets, one for apartments and one for houses, where the hyperparameters were optimized using both random search and Bayesian optimization each run for 100 iterations with 5-fold cross validation and early stopping[18]. The hyperparameters to be optimized were the following:

The `subsample` variable controls the row-subsampling and is a number between 0 and 1.

The hyperparameter `colsample_bytree` controls the column down-sampling for each tree, so how many columns (or variables) each tree are allowed to fit to. Is a number between 0 and 1.

The `max_depth` controls the maximum depth of every tree. Is a positive integer (negative values means no maximum depth).

The `min_child_weight` variable controls when the decision tree algorithm should stop splitting a node into further nodes (and will thus turn it into a leaf). Is a positive integer.

[17] Remember that the number of trees were selected by early stopping.

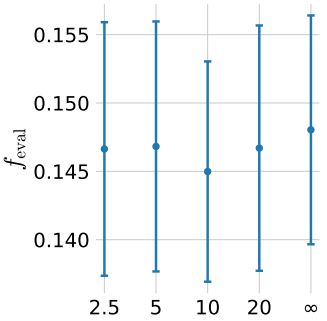


Figure 3.14: Evaluation score as a function of the weight half-life $T_{\frac{1}{2}}$ with the standard deviation over the 5 folds as errorbars for apartments.

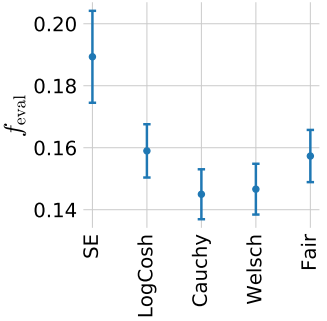


Figure 3.15: Evaluation score as a function of the loss function with the standard deviation over the 5 folds as errorbars for apartments.

[18] This takes a good 4 hours for the RS, 7 hours for the BO, and 90 minutes to optimize the learning rate with early stopping for apartments when run on the local computing cluster, HEP. For the houses this takes a good 24 hours for the RS, 34 hours for the BO, and almost 5 hours optimize the learning rate with early stopping.

`reg_lambda` controls the L2 regularization term. Is a positive number.

`reg_alpha` controls the L1 regularization term. Is a positive number.

The ranges of the hyperparameters were chosen by a manual, iterative process of fitting a subset of the data (1 %–10 %) and making sure that the best hyperparameter is not sufficiently close to the range; if it is, then the range is extended. The final HPO ranges chosen can be seen in Table 3.9. Here $\mathcal{U}(a, b)$ refers to a uniform distribution from a to b, and $\mathcal{U}_{\text{int}}(a, b)$ is the same only an integer distribution.

Hyperparameter	Range
`subsample`	$\mathcal{U}(0.5, 0.9)$
`colsample_bytree`	$\mathcal{U}(0.1, 0.99)$
`max_depth`	$\mathcal{U}_{\text{int}}(1, 20)$
`min_child_weight`	$\mathcal{U}_{\text{int}}(1, 30)$
`reg_lambda`	$\mathcal{U}(0.1, 4)$
`reg_alpha`	$\mathcal{U}(0.1, 4)$

Table 3.9: Probability Density Functions used in the random search to draw new sets of values for the hyperparameters. Each hyperparameter is drawn from the distribution seen in the table.

The fitting pipeline for this subproject is to first run both RS and BO as HPOs to compare their results. The best of the two models is chosen and finally learning rate optimized by early stopping where the learning rate η is reduced from $\eta = 0.1$ to $\eta = 0.01$. In the end one ends up with a model that has been HPO optimized for preprocessing optimizations, loss functions, sample weights, normal BDT hyperparameters and finally the learning rate. I have manually implemented this pipeline in Python since no other packages provide the same flexibility as a custom implementation that works fully automatically.

The results of the RS and BO can be seen in Figure 3.16 and A.22. The corresponding plots for houses can be seen in Figure A.23 and A.24.

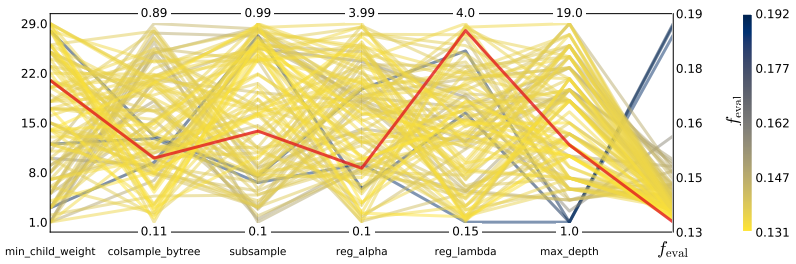


Figure 3.16: Hyperparameter optimization results of XGBoost parameters of the housing model for apartments shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

As with the initial HPO, it is important to compare the results in Figure 3.16 with their uncertainties. This can be seen in Figure 3.17. Here the value of the evaluation function along with its 1σ and 2σ uncertainties are plotted as a function of the iteration along the abscissa. The minimum value of f_{eval} is shown in red. Even though this is the minimum value, notice how most of the other iterations are within 1σ. The plot for BO in Figure A.26 shows the exploration phase in the first half of the iterations where it afterwards converge to a more flat minimum, however, this minimum was still worse than the one found with RS.

The best of the RS and GS models are chosen for subsequent analysis by first reducing the learning rate to $\eta = 0.01$ and then find the best number of estimators by early stopping. The evaluation function as a function of number of estimators, also known as the *learning curve*, is seen in Figure 3.18. This curve is the realization of

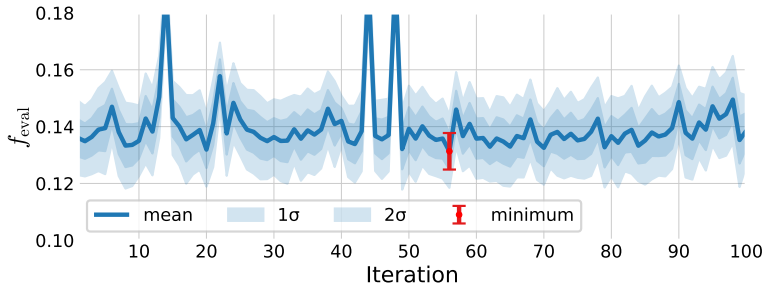


Figure 3.17: The results of running random search (RS) as hyperparameter optimization (HPO) on apartments using the XGB-model. The minimum (mean) loss along with its uncertainty is shown in red, the means for the different iterations of RS in blue, and as light blue bands are the one (and two) standard deviation(s), all as a function of iteration number.

Figure 2.2 in real data, where it first improves a lot and then finds a stable plateau. The minimum is shown in red with its uncertainty. To reduce the risk of overfitting and model complexity – with the further advantage of resulting in a faster model at prediction time – we keep the model with the lowest number of estimators that are still within 1σ of the minimum: see the orange point in the figure. This results in a model that contains less than a fifth of the number of trees and is thus also significantly simpler and faster at inference time. This is the final hyperparameter optimized model that will be used for the further analysis.

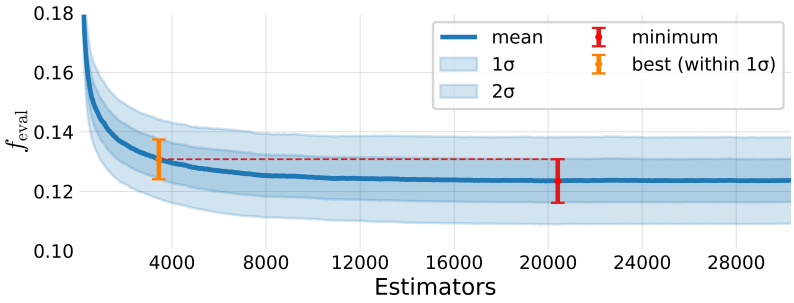


Figure 3.18: The results of early stopping on apartments using the XGB-model. The minimum (mean) loss along with its uncertainty is shown in red, the means for the different iterations of RS in blue, and as light blue bands are the one (and two) standard deviation(s), all as a function of number of estimators (trees). In orange the "best" number of estimators is shown, defined as the lowest number of estimators which are still within 1σ of the minimum value.

3.6 *Results*

The performances of the different models, the random search optimized, the bayesian optimization optimized and the best of the two η optimized with early stopping, are shown in Figure 3.19. Here the distribution of the relative price prediction $\mathbf{z}$ of the model evaluated on the test set, apartments sold in 2018, is shown for the three models. In addition to the distributions, also the performance metrics are shown: the value of the evaluation function[19] along with the fraction of relative price predictions that are within the specified percentage. In this particular instance it is seen that 41.3 % of the predictions by the final early stopping model are less than ±5 % wrong, 69.8 % within ±10 %, and 91.9 % within ±20 %. Note that the differences between the three models are minor and that the distributions almost cannot be distinguished from each other.

An interesting observation from the performance metrics of the test set is the low value of $f_{\text{eval}} = 9.289\,\%$ (`MAD`). By looking at Figure 3.18 one would expect a test loss of around 12 % assuming iid. samples. However, this assumption does not seem to be fulfilled for the test set. The performances of the realtors are also better for

[19] Written as `MAD`.

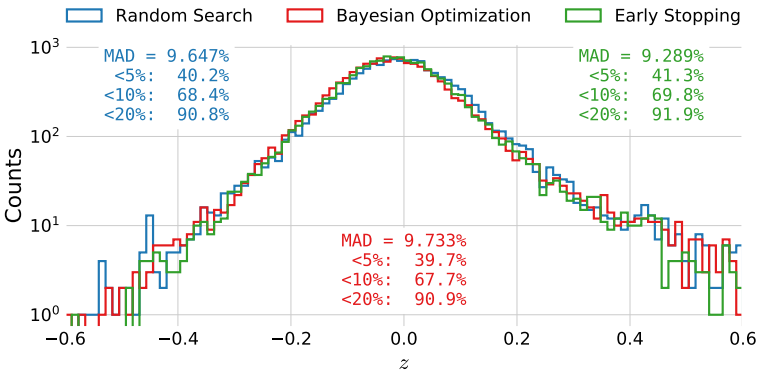


Figure 3.19: Histogram of z-values of the XGB-model trained on apartments. The performance after hyperparameter optimization using random search is shown in blue, for Bayesian optimization in red, and the learning rate optimized with early stopping in green.

the test set than the training set, and by comparing the test set with the extra 2019 set it seems to be 2018 that was an extra homogenous year, see Table 3.10. The "Tight" in the the table corresponds to the realtors performance on the tight version of the different datasets. The performance of the XGB models on the tight test set can be seen in Figure A.27, where it can be seen that the final model has $f_{\text{eval}} = 8.383\,\%$.

	Train	Test	2019
Normal	5.80 %	4.97 %	6.19 %
Tight	5.69 %	4.94 %	6.19 %

Table 3.10: The MAD of the realtors' predictions for the normal and tight selections in the training, test, and 2019 datasets.

To gauge the predictive power of the model over time, we applied the model to the next months data, evaluated the results for that month and continued like that for all the months in the test set (2018) and 2019. We apply two different methods of forecasting: *static* forecasting where the model is only trained once, and *dynamic* forecasting where the model is retrained after each month on all of the previous sales. These predictions allows the relative predictions $\mathbf{z}$ to be calculated and the MAD and the standard deviation (SD) of $\mathbf{z}$ are shown in Figure 3.20.

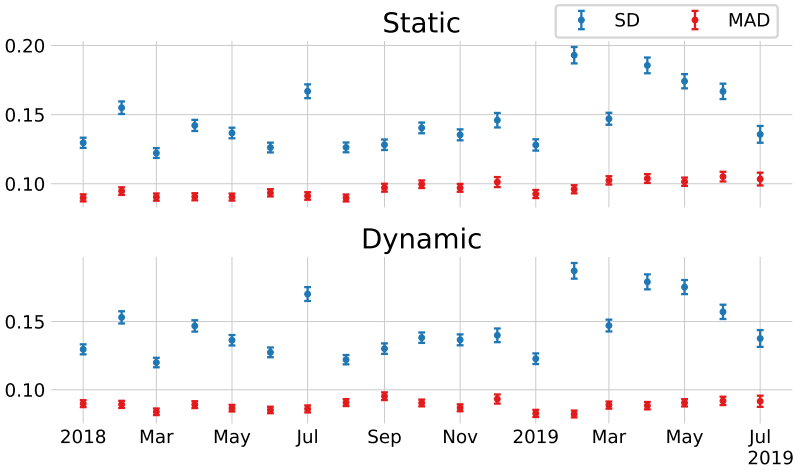


Figure 3.20: Performance of 1-month forecasts for apartments sold in 2018 and 2019. For both plots the XGB model is trained on data up to (but excluding) 2018. Top) The performance of the static model's prediction for both the standard deviation (SD) and MAD of the z-scores. Bottom) Same as above, however, based on a dynamic model, i.e. a model which is retrained after every month to include the previous month's sales.

In the top subplot the results for the static forecast are shown, whereas the dynamic results are shown in the bottom subplot. The errorbars are calculated using the usual variance of the standard deviation:

$$\sigma_\mu = \frac{\sigma}{\sqrt{N}}, \quad \sigma_\sigma = \frac{\sigma}{\sqrt{2N}}, \tag{3.8}$$

where σ_μ is the standard deviation of the mean and σ_σ is the standard

deviation of the standard deviation[20] [18]. Notice the large fluctuations in the standard deviations over time compared to MADs which is an effect of MAD being a robust estimator. What is also interesting to note is that the MAD seems to increase over time for the static model, albeit slowly. In comparison, for the dynamic model this seem less pronounced. This figure not only shows the time dependence of the performance of the model, but also that the model is quite stable over time, at least for the dynamic model.

[20] That this estimator is biased does not matter since we are in the large N limit, $N \sim 1000$.

Using the relative price predictions $\mathbf{z}$, we construct the Market Index, MI. This is an index which measures the overall level of the Danish housing market based on the assumption that if the houses sold in a month are generally sold at a higher price than was predicted by the model, there can be two reasons for it: either the model was wrong or the market simply changed in the time span. With the latter assumption, the ratio between the prediction and the actual price of a residence is thus a measure of the market index:

$$\begin{aligned} z_{\text{mi}} &\equiv \frac{\hat{y}}{y} = 1 - \frac{\hat{y} - y}{y} = 1 - z \\ \text{MI}_{\text{mean}} &= \text{mean}(\mathbf{z}_{\text{mi}}) \\ \text{MI}_{\text{median}} &= \text{median}(\mathbf{z}_{\text{mi}}). \end{aligned} \tag{3.9}$$

Here $\mathbf{z}_{\text{mi}}$ is the vector of ratios between prediction and actual price, and the market index can then be estimated using either the mean MI_{mean} or the median $\text{MI}_{\text{median}}$. The market indices for every month of the forecast described in the previous figure can be seen in Figure 3.21. In the top panel the market index is shown for the static model. Here the median $\text{MI}_{\text{median}}$ is consistently higher than 1, around 1 %. Compare this to the dynamic $\text{MI}_{\text{median}}$ in the lower plot which fluctuates around 1. The mean MI_{mean} is added to show its fluctuations are higher than the medians.

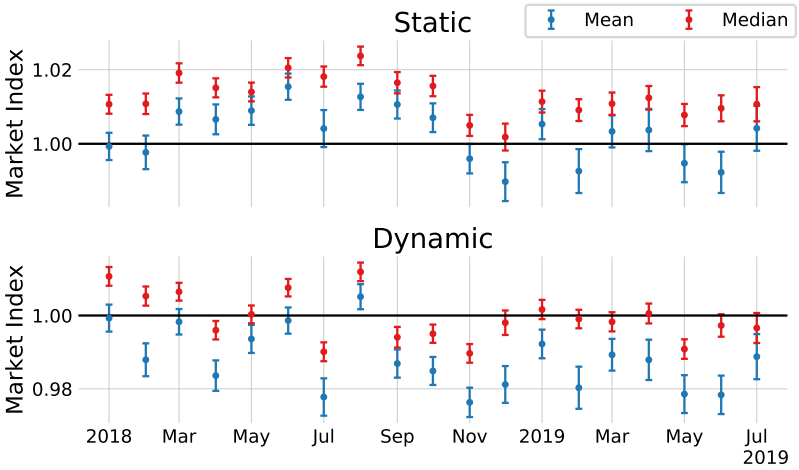


Figure 3.21: The Market Index as defined in equation (3.9) for the static and dynamic 1-month forecasts for 2018 and 2019. Plotted with using the mean in red and the median in blue.

The final results for the model is seen in Table 3.11 for owner-occupied apartments and in in Table 3.12 for one family houses. The MAD is around 9 % for apartments and 16 % for houses, which is still worse than the realtors' prediction, yet still acceptable for a model

that does not have any variables describing the indoor conditions. For apartments around 40 % of all the predictions are within ±5 % and more than 90 % are within ±20 % which is similar to the performance of the professional automated property evaluations from e.g. Bolighed [25]. The performance on the tight cuts can be seen in Table A.13 and A.14.

	MAD (%)	$\leq 5\%(\%)$	$\leq 10\%(\%)$	$\leq 20\%(\%)$
Train	7.83	47.90	75.74	93.97
Test	9.29	41.33	69.77	91.91
2019	9.89	38.85	66.76	90.04

Table 3.11: Performance metrics for the final housing model trained and evaluated on apartments.

	MAD (%)	$\leq 5\%(\%)$	$\leq 10\%(\%)$	$\leq 20\%(\%)$
Train	14.12	28.95	51.89	78.04
Test	15.79	25.61	47.52	75.14
2019	16.50	24.01	45.89	74.22

Table 3.12: Performance metrics for the final housing model trained and evaluated on houses.

3.7 *Model Inspection*

One of the most important aspects of applying advanced machine learning methods, in addition to getting accurate predictions is to understand the model. As mentioned in section 2.8, it is possible to use SHAP values to inspect the trained model for both local predictions and for global feature importances ϕ_i^{tot}. An example of how to use SHAP to better understand a local prediction is seen in Figure 3.22. Here the SHAP values for a particular sale are visualized as a bar chart where the green colors are positive values, red values negative values and the blue is the final prediction $\hat{y}$. Remember that for SHAP values the prediction is the sum of all of the SHAP values, see equation (2.40). Here ϕ_0 is the expectation value denoted as `Bias` in the plot. To show all 143 variables would make the figure excessively large, so two extra bins have been added: the `Overflow` bar which is the sum of the remaining positive SHAP values and likewise with `Underflow` for negative values. The sum of all the green and red bars adds up to $\hat{y} = 6.86$ M.kr. in this particular instance and the actual sold value was $y = 6.35$ M.kr.. Thus, in cases where there is a large discrepancy between the predicted and actual sales prices, one use can use this tool to better understand why the prediction was estimated as it was.

To get an overview of which variables are most important on a global[21] scale, ϕ_i^{tot}, see Figure 3.23. Here the variables are sorted according to the normalized ϕ_i^{tot} which is shown in parentheses after each variable name. In the center of the plot is shown a dot-plot of the dataset plotted with the SHAP value on the abscissa and colored according to the feature value. The way to interpret this plot is as follows. Take a variable of interest, e.g. the area of the apartment `ArealBolig` with $\phi^{\text{tot}}_{\text{ArealBolig}} = 5.35\,\%$. Every dot is a sale plotted

[21] Global here meaning for the entire dataset.

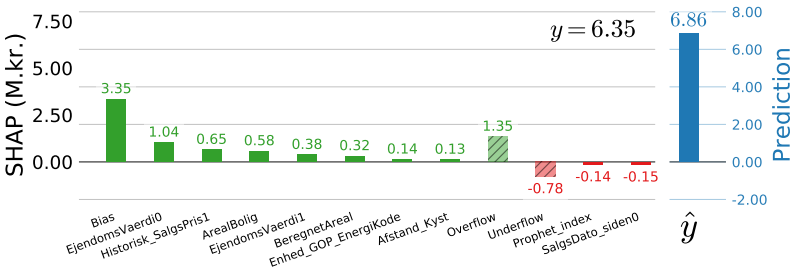


Figure 3.22: Model explanation for XGB model for a specific apartment. The bars are the variables in the dataset that the model found most important sorted after their importance for this particular apartment. The bias bar refers to the expected value of the model, which is simply the mean of the training set which acts as the naive prediction baseline. The "cutoff positive (negative)" bars are the sum of the remaining positive (negative) values that are not shown. On the right hand side of the plot is the model prediction shown. The model prediction is the sum of all of the bars in the left par (6.86 M.kr. in this example). The negative values are shown in red, positive ones in green, and the prediction value in blue.

as a function of their SHAP value with a spread such that the height corresponds to the SHAP distribution of that specific feature. For the area it can be seen that there is a long tail towards high SHAP values, however, most of the samples have a slightly negative SHAP values. The dots are colored according to their feature value and it can thus be seen that large apartments (red) are given a higher SHAP value than small apartments (blue); precisely as expected from the model. In contrary, when the the total days on market (`LiggetidSamlet`) is large it pushes the prediction in the negative direction.

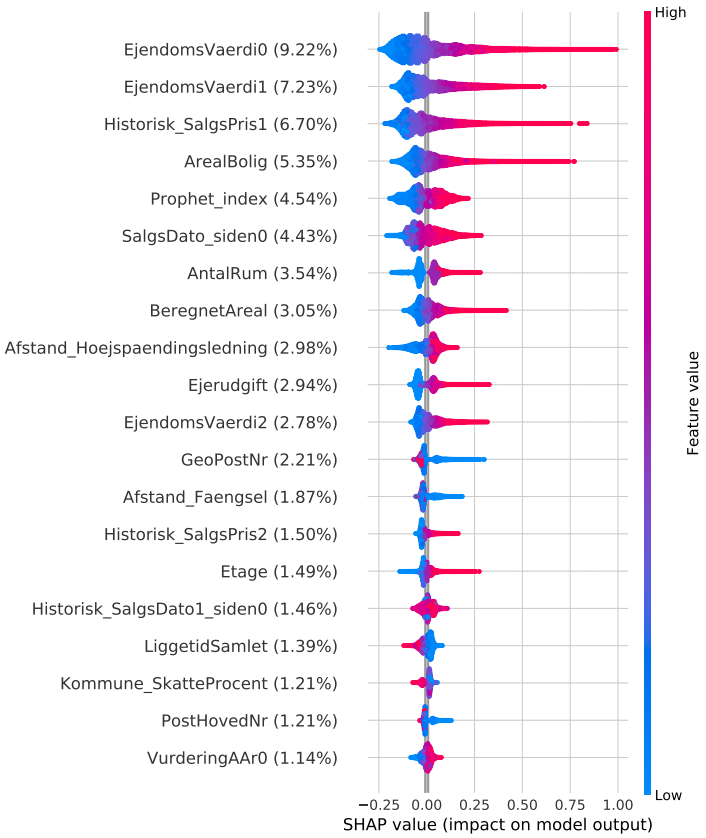


Figure 3.23: Feature importance of apartment prices using the XGB-model. The feature importance is measured using SHAP values. The variables are sorted top to bottom according to their overall feature importance, i.e. the previous public property valuation `EjendomdsVaerdi0` is the most important single feature. Along the x-axis is the impact on model output, in this example the price in M.kr. This axes is colored by the value of the feature, from low (blue) to high (red). In this particular example we see that high values of the previous public property valuation has high, positive impact on the model prediction – exactly as expected. This is exactly opposite the total days on market described by the variable `LiggetidSamlet` where a high value has a negative impact.

The SHAP software [67] not only allows for 1D dependencies to be gauged, it also allows for so-called *interaction plots*. These plots shows the 2D-dependence between the variable and the SHAP value colored according to a second variable. Since the previous public property valuation (PPPV) (`EjendomdsVaerdi0`) is the most important of the features, the interaction plot of this variable is seen in

Figure 3.24.

Here the SHAP value of the `EjendomdsVaerdi0` is plotted as a function of `EjendomdsVaerdi0`. This plot shows that the higher the PPPV, the higher the model output. The colors show how this trend depends on time by the variable `SalgsDato_siden0` which is the number of days since January 1st 2009 that the apartment was. The plot shows that if the apartment has a high PPPV then the newer sales has an even higher SHAP value then older sales, agreeing with the fact that the market has gone up since 2009. On the other hand, for low PPPV apartments the relationship with time is inverse, however, the effect is much smaller here. The SHAP software chose to color by `SalgsDato_siden0` since this is the variable which explains most of the variation for a given PPPV.

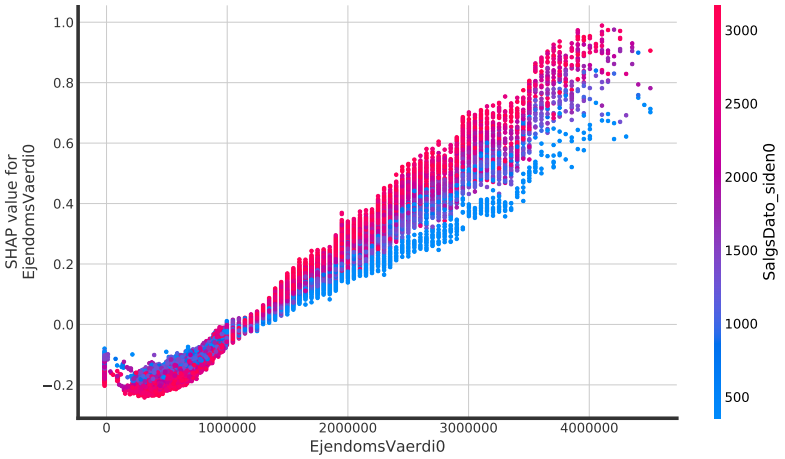


Figure 3.24: Interaction plot of the feature importances for the XGB model on apartments. Here shown for the previous public property valuation (`EjendomdsVaerdi0`) colored according to the sales date (`SalgsDato_siden0`).

What can be concluded from this plot is that for apartments with a high PPPV and that were sold recently were sold for more than apartments with the same PPPV that were sold longer time ago; effectively the time dependence of sales prices. These interaction plots serves as helpful sanity checks to see that the model is actually learning reasonable relationships between the variables.

3.8 *Multiple Models*

In addition to the XGBoost [36] (XGB) BDT model, several other models were also tested. During the sub-project the LightGBM package [61] (LGB), also a BDT model, was released and started gaining traction in the ML community, especially for large-scale data analysis. In comparison to XGBoost, LightGBM implements some extra binning and categorical assumptions that greatly speeds up the fitting process. It was trained in the same was as the XGBoost model with the same HPO-process and range for the hyperparameters.

To compliment the BDTs models, a simple linear model (LIN) with L_2 loss, so-called *ridge regression*, was fitted. Before the fit all of the NaNs were median-imputed which means that all invalid values were replaced with the median along each column. All the features were scaled[22] with a robust scaler from Scipy [98] in the $(25, 75)$

[22] Scaling of input features is generally an important preprocessing step for non-tree based ML models.

quantiles range and the regularization parameter was hyperparameter optimized. This linear model is quick and easy to both implement and fit, and can be seen as the simplest baseline model.

The K-nearest neighbors (KNN) algorithm was fitted to the data with a data preprocessing pipeline similar to the linear model with median imputation and robust scaling of the input features. For the HPO the number of neighbors, K, was optimized for together with the p-norm of the metric, $p \in \{1, 2\}$ (Manhattan, Euclidean, see also subsection 2.4.1).

Finally, support vector machines[23] (SVR) were used with the same preprocessing pipeline as the previous two methods and hyperparameter optimized for the L_2 regularization parameter C and the kernel coefficient γ for the radial basis function (RBF) kernel.

[23] Known as support vector regression when applied to regression problems [17].

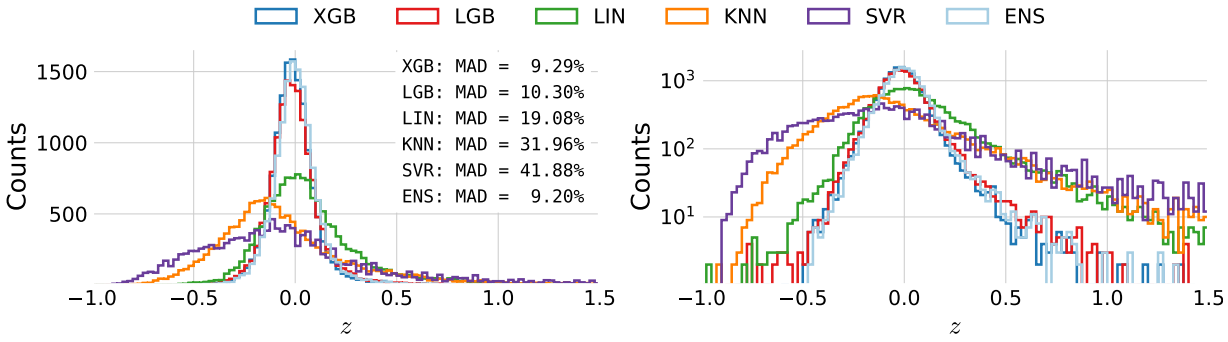


Figure 3.25: Distribution of the relative predictions $\mathbf{z}$ for the five different models and the ensemble model. The left plot shows the normal histogram (with a linear scale), whereas the right one has a log-scaled y-axis to better see the tails of the distributions. The MAD scores for the different models is further shown in the left plot.

The results for the five different models are shown in Figure 3.25, where the two subplots are the same up to a logarithmic scaling of the ordinate axis in the right plot. It is easily seen how XGBoost and LightGBM are the best-performing models together with the ensemble (ENS) of the different models, see paragraph below for further explanation. In the figure the MAD is also shown on the test set, which confirms the visual clue: that XGBoost performs best, followed by the ensemble model.

The five different models – XGB, LGB, LIN, KNN, SVR – should be able to each capture different parts of the hyperdimensional phase space and an ensemble of these models would thus be expected to be as good as or better than the best of the individual models. This kind of ensemble model is sometimes called *super learner* in the statistics community [75, 93]. To make sure that the ensemble model is not just retraining on the training set, and thus end up overfitting, we follow the process from Polley and van der Laan [75]. Using cross-validation for time-series[24] data with 10 folds, the training data is split up into folds sorted by time. Each fold is fitted with all five models, and the prediction of the next fold is made for all five models. This is repeated for the remaining folds until one ends up with a matrix of predictions $\mathbf{Z} \in \mathbb{R}^{(N \times 5)}$ for N training samples. Since all the folds in $\mathbf{Z}$ consists of predictions on unseen data[25] this prevents overfitting. The meta learner then fits Z to the actual predictions of the training data y in the usual way. The combination of a meta

[24] See subsection 2.4.2.

[25] The predictions for the very first fold is based on training data.

learner fitted to the predictions of individual models is called an ensemble model.

At first an XGBoost model was used as the meta learner yielding decent results, however, still performing worse than the single XGB model (MAD = 9.57 %). To better understand the issue, the meta learner was changed to a linear model which would basically just compute a weighted average of the different models:

$$\Psi_{\text{meta}}(\mathbf{x}) = \sum_{i=1}^{5} \alpha_i \Psi_i(\mathbf{x}), \tag{3.10}$$

where $\boldsymbol{\alpha}$ is a vector of the weights for the meta learner and Ψ_i is each of the individual ML models. The linear model performed even worse than using XGB as the meta learner (MAD = 10.48 %), yet it was more transparent. During the debugging process it was realized that none of these models actually optimize the evaluation function, MAD, directly. The XGBoost model was trained using the Cauchy loss found earlier and the linear regression model a simple squared error loss. Since a simple weighted average should work as the meta learner [75], a custom algorithm for finding $\boldsymbol{\alpha}$ according to MAD was implemented.

Given the training data, the evaluation function as a function of $\boldsymbol{\alpha}$ was minimized using of the MINUIT algorithm [59] via the iminuit [88] Python interface. It yielded decent result, yet they were all very dependent on the initial parameter of the fit indicating many local minima. A scan over the 5-dimensional hyperspace in steps of 0.01 was thus conducted and the result of this scan was used a the new initial parameter in the minimization routine. This yielded the following result:

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_{\text{LIN}} \\ \alpha_{\text{KNN}} \\ \alpha_{\text{SVR}} \\ \alpha_{\text{XGB}} \\ \alpha_{\text{LGB}} \end{bmatrix} = \begin{bmatrix} 0.202\,\% \\ 0.002\,\% \\ 0.001\,\% \\ 81.302\,\% \\ 20.002\,\% \end{bmatrix}. \tag{3.11}$$

The fact that it sums to more than 1 just corresponds to an overall scaling. When using the found $\boldsymbol{\alpha}$ in equation (3.10), one gets the ensemble model (ENS) shown in Figure 3.25 with a MAD = 9.20 %. This value is the evaluation loss on the test set based on only the training data and outperforms all of the individual models.

The paragraphs above refer to apartments, however, the intermediate results for houses showed the same pattern. The combined model along with their performance can be seen in Fig XXX (need to be added to appendix).

3.9 *Discussion*

The subproject of estimating housing prices has focussed a lot on experimenting with different machine learning models and how to optimize them. As it can be seen in the previous sections, the choice

of ML model is by far the most important. Actually, the gain from HPO is quite small, especially considering the amount of time spent on it[26]. With the dataset at hand, decent results were made, however, they were nowhere near the performance of the realtors' predictions.

[26] Not only user-time programming it, but also the computational ressources spent.

There are two main reasons for this, the first being that realtors are educated within this field and thus has developed the skills required for estimating the price of a house over many years of hard work. The second reason is the fact that the realtor has access to a lot more information than the ML models have. We are not in possession of any *indoor* variables as we call it. The area of the house, the number of rooms, the name of the street, and the distance to a highway are all variables that are in the data set but none of them describe the overall quality of the house, the maintenance level, the age of the kitchen or bathroom. These features are invisible to the ML model.

During the project it was investigated how to get access to these variables. At first the online images from each sale was suggested, however, it turned out that Boligsiden only have the right to use them while a residence is for sale; when it is sold all rights return to the photographer. The images are not the only thing that provide more information about the condition of the residence, also the descriptions does that.

The descriptions turned out to be available for most of the sales and was investigated for a short period. At that time of the project, the MAD for (a subset of the) apartments was around $\pm 14\,\%$ and $\pm 20\,\%$ for houses. By using methods from the big natural language processing (NLP) community with in the field of machine learning, it was possible to reduce the MAD to around $\pm 12\,\%$ and $\pm 15\,\%$ for apartments and houses respectively. From the improvement in performance it is noticeable how apartments in general are much more uniform compared to houses where the "inside" is more decisive regarding the price.

The methods for translating the text to numerical variables decipherable by classical ML models were for instance simple *bag of words* (BOW) models and term *Term Frequency, Inverse Document Frequency* (TF-IDF) but also slightly more advanced statistical tools such as *Latent Dirichlet Allocation* (LDA). An old example of the learnt text model is seen in Figure 3.26 where a housing-based model was trained with the five numerical variables[27]: the PPPV, postal code, year of construction, distance to shore, and the weighted area. and the text descriptions (encoded with TF-IDF).

[27] With the Danish names: `Ejendomsvaerdi0`, `GeoPostNr`, `ByggeAAr`, `Afstand_Kyst`, and `BeregnetAreal`.

The summary of the trained model is as a SHAP plot in Figure 3.26. As expected, the most important features are the numerical ones, however, the word `flot` ("pretty") was in top five. The model also learnt that `flot` has a positive impact on the price compared to `trænger` ("requires") which has a negative impact.

The descriptions turned out to be more time-consuming to extract for Boligsiden and along with the fact that overall deadline was quickly approaching, the remaining time was focussed on the main part of the project, the quark gluon discrimination. Given more time,

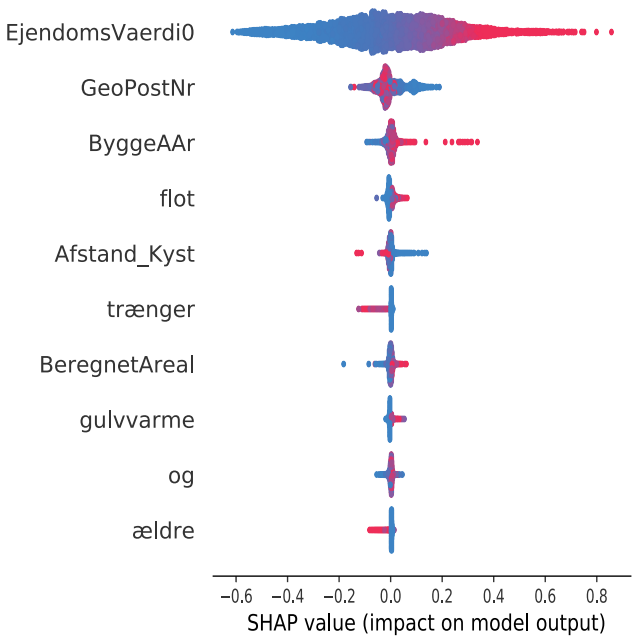


Figure 3.26: Feature importance of villas when the descriptions are also used. The variables are sorted top to bottom according to their overall feature importance, i.e. the previous public property valuation `EjendomdsVaerdi0` is the most important single feature. Along the x-axis is the impact on model output, in this example the price in M.kr. This axes is colored by the value of the feature, from low (blue) to high (red). In this particular example we see that high values of the previous public property valuation has high, positive impact on the model prediction – exactly as expected. This is in contrast to the word "requires" (`trænger`) where a high value has a negative impact.

the text analysis would definitely be the first step for further improving the accuracy and precision of the price predictions.

Another step would be to apply more modern deep learning[28] methods. These methods were briefly experimented with in the initial stages of this subproject but showed inferior performance compared to BDTs. It is generally accepted in the ML community (with some modifications) that neural networks underperform, or at least not outperform, classic ML methods on structured [29] data [63]. Most often they have the inherent complexity needed to perform as well as other ML methods, however, this requires extensive architecture optimization, or, in short; the hypothesis space for neural networks is much larger than for classical ML methods and thus requires more care to avoid overfitting.

[28] Basically advanced neural networks with many layers.

[29] In general data that can be described by a spread sheet, i.e. has a well-defined number of variables and observations which is why it is also known as *tabular data*.

3.10 *Conclusion*

XXX **TODO!**

Part II

Part II of this thesis deals with particle physics and the discriminatory power of machine learning for quark-gluon identification and subsequent analysis. In chapter 4 the theory of the Standard Model is introduced together with a description of the ALEPH detector. Theory is applied in chapter 5 where the types of jets and events in each collision is analysed using machine learning to improve the understanding of how gluon jets hadronizes and splits: simply said how they look and behave.

A. Housing Prices Appendix

Figure A.1: XXXX **TODO!**.

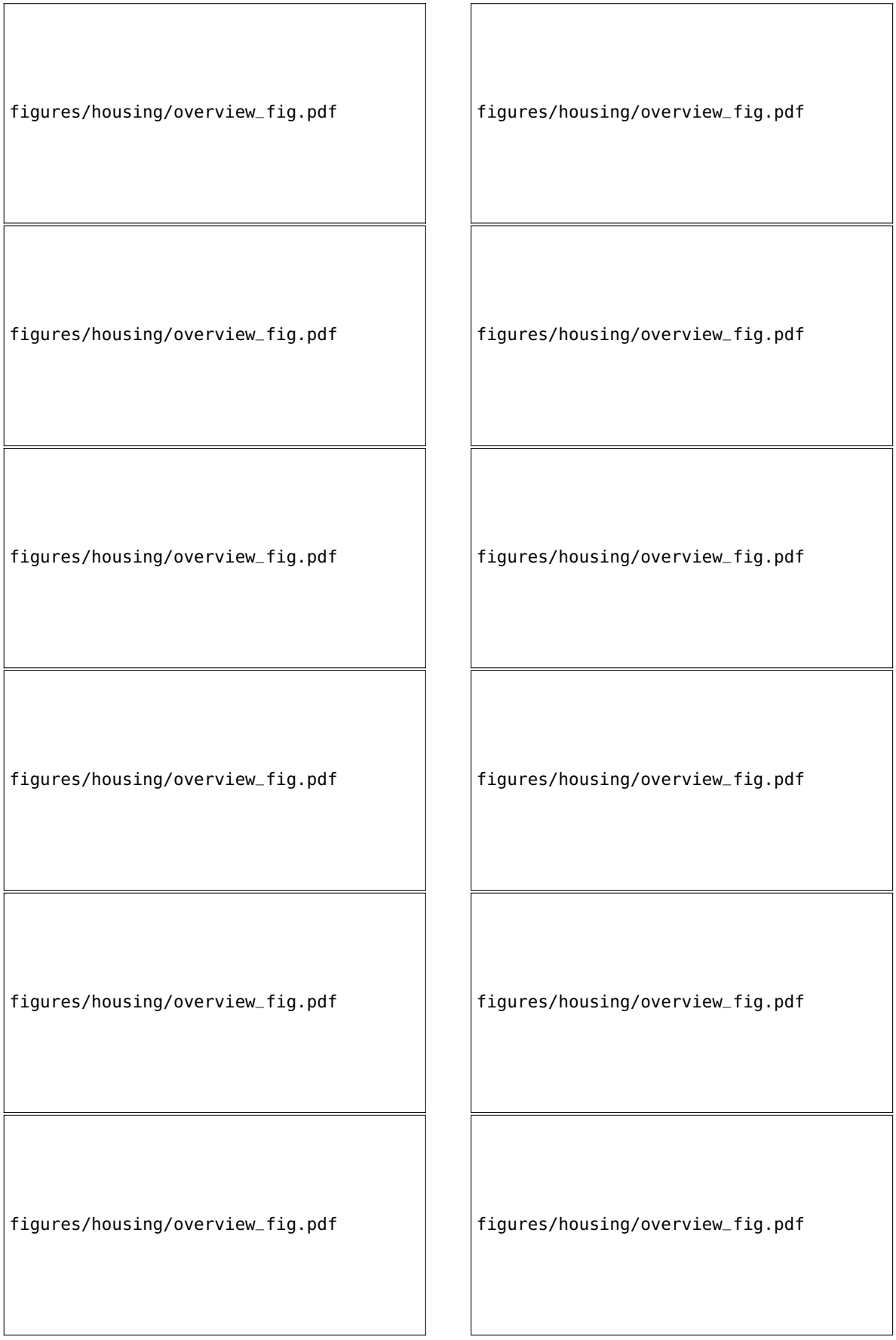


Figure A.2: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

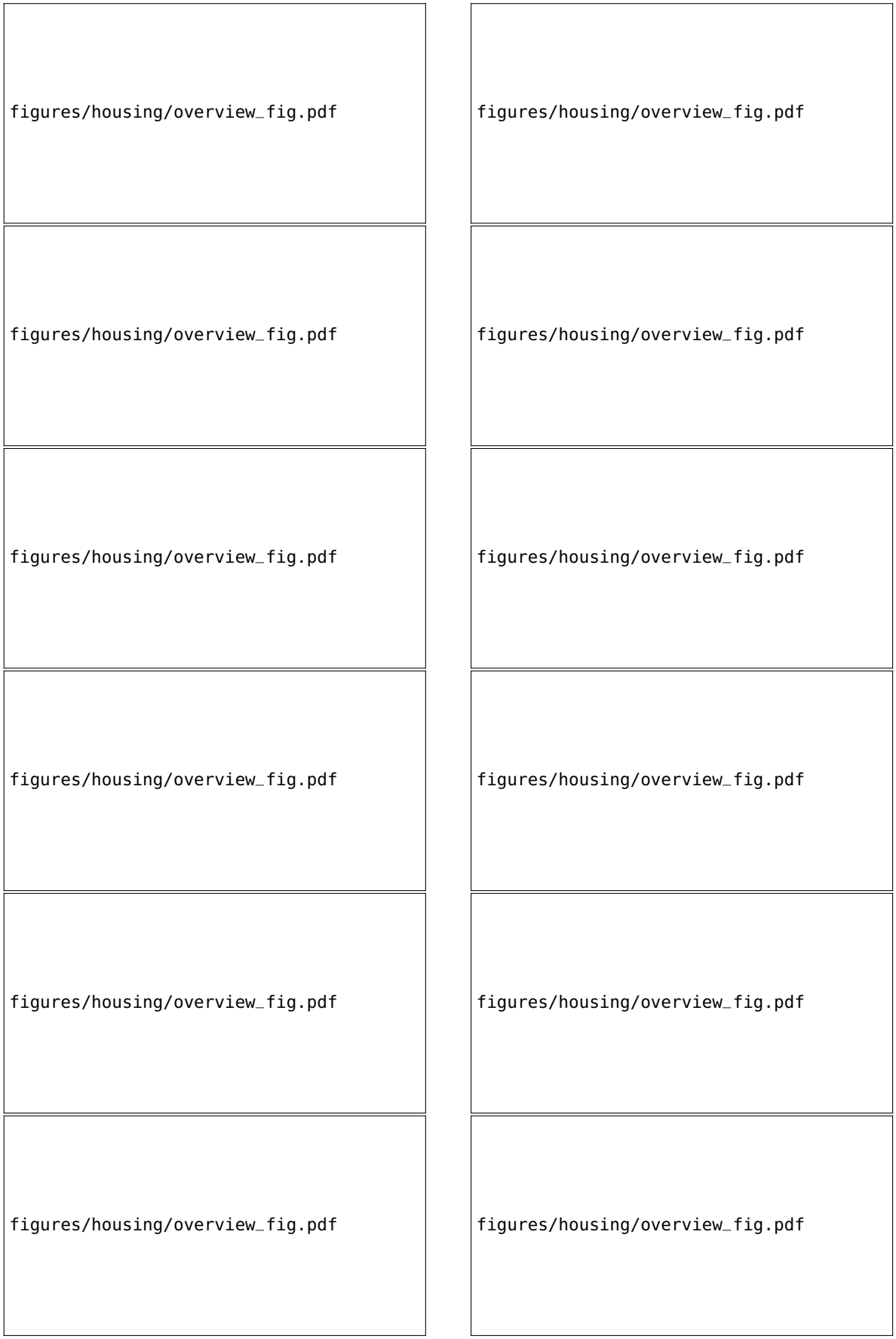


Figure A.3: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

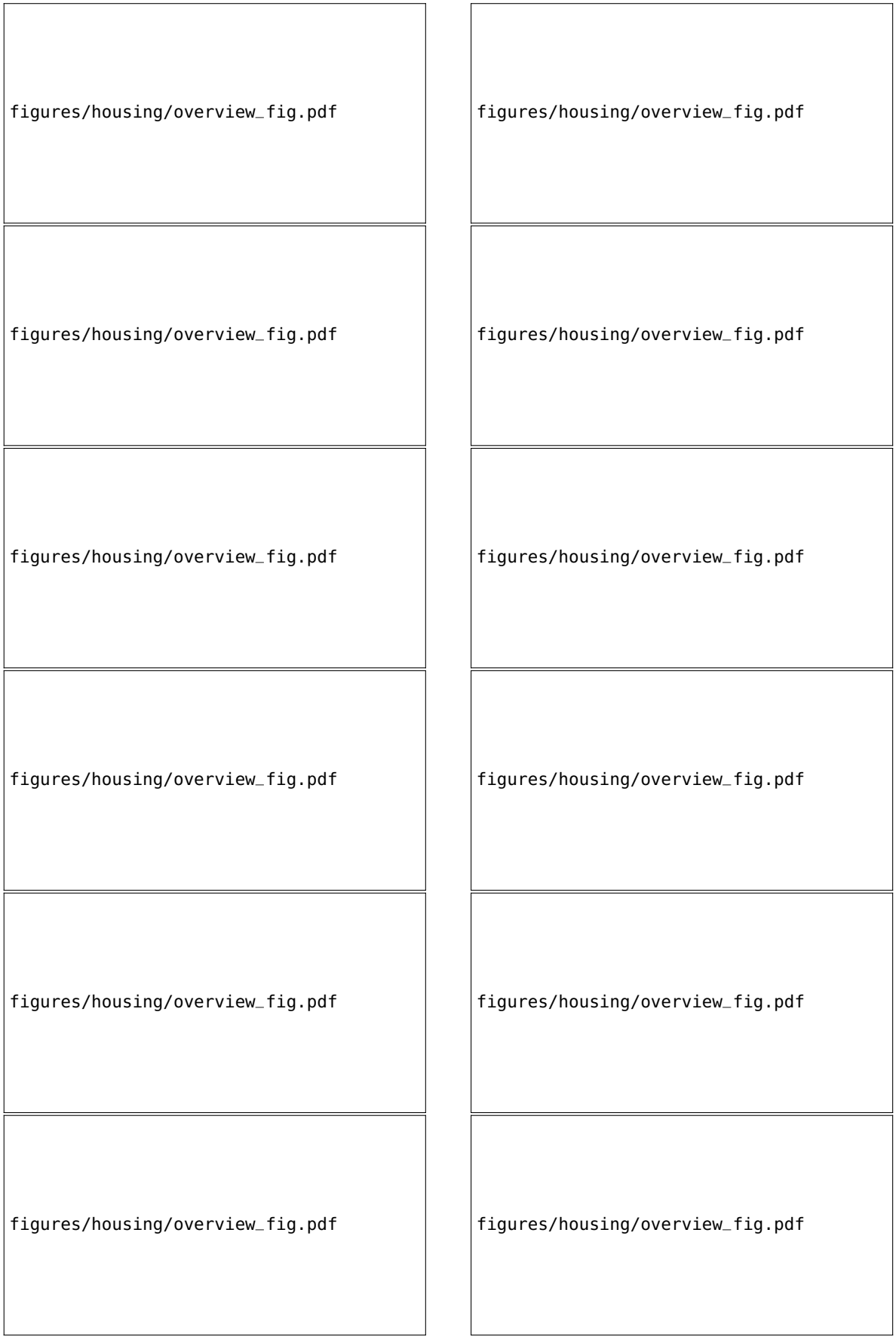


Figure A.4: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

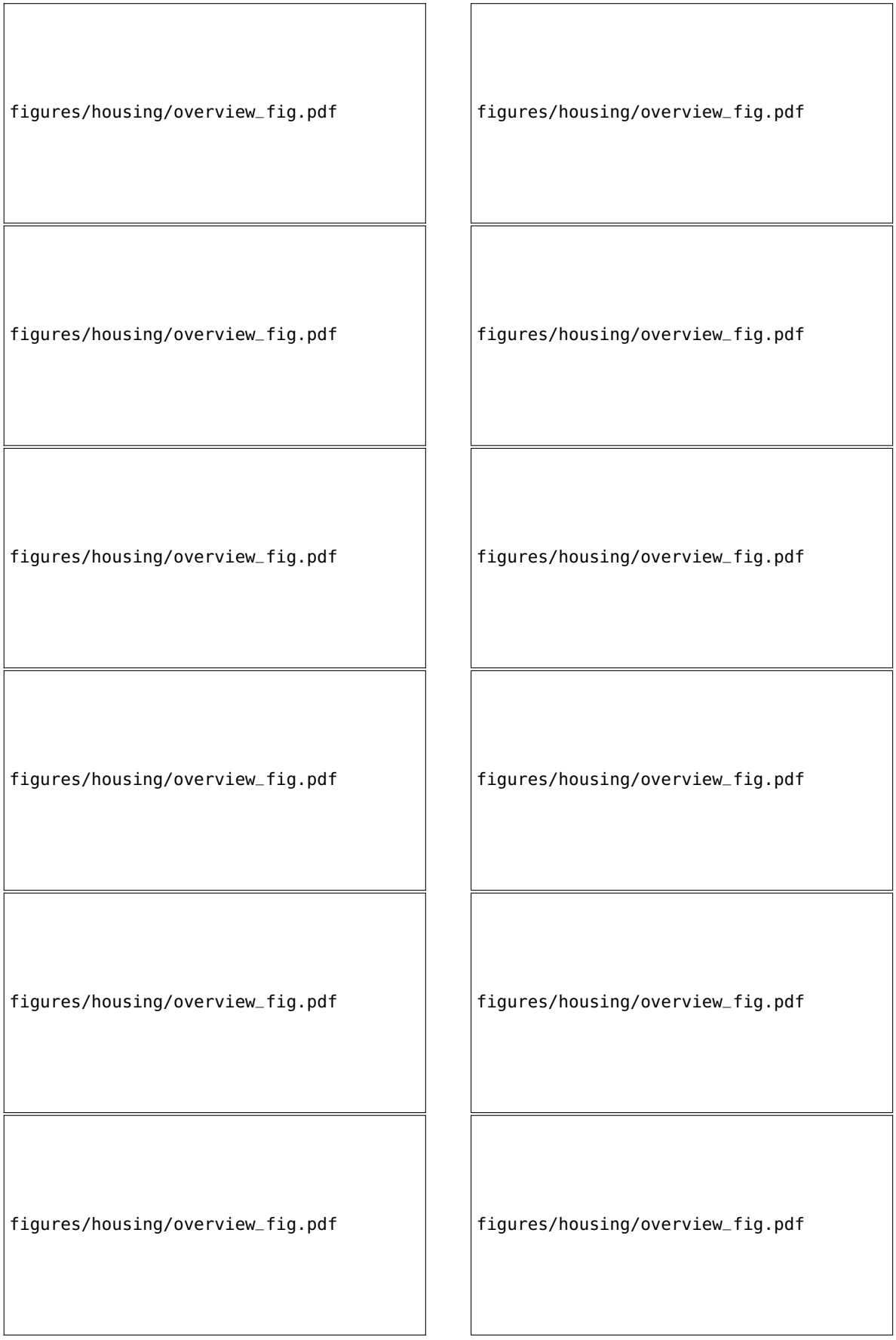


Figure A.5: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

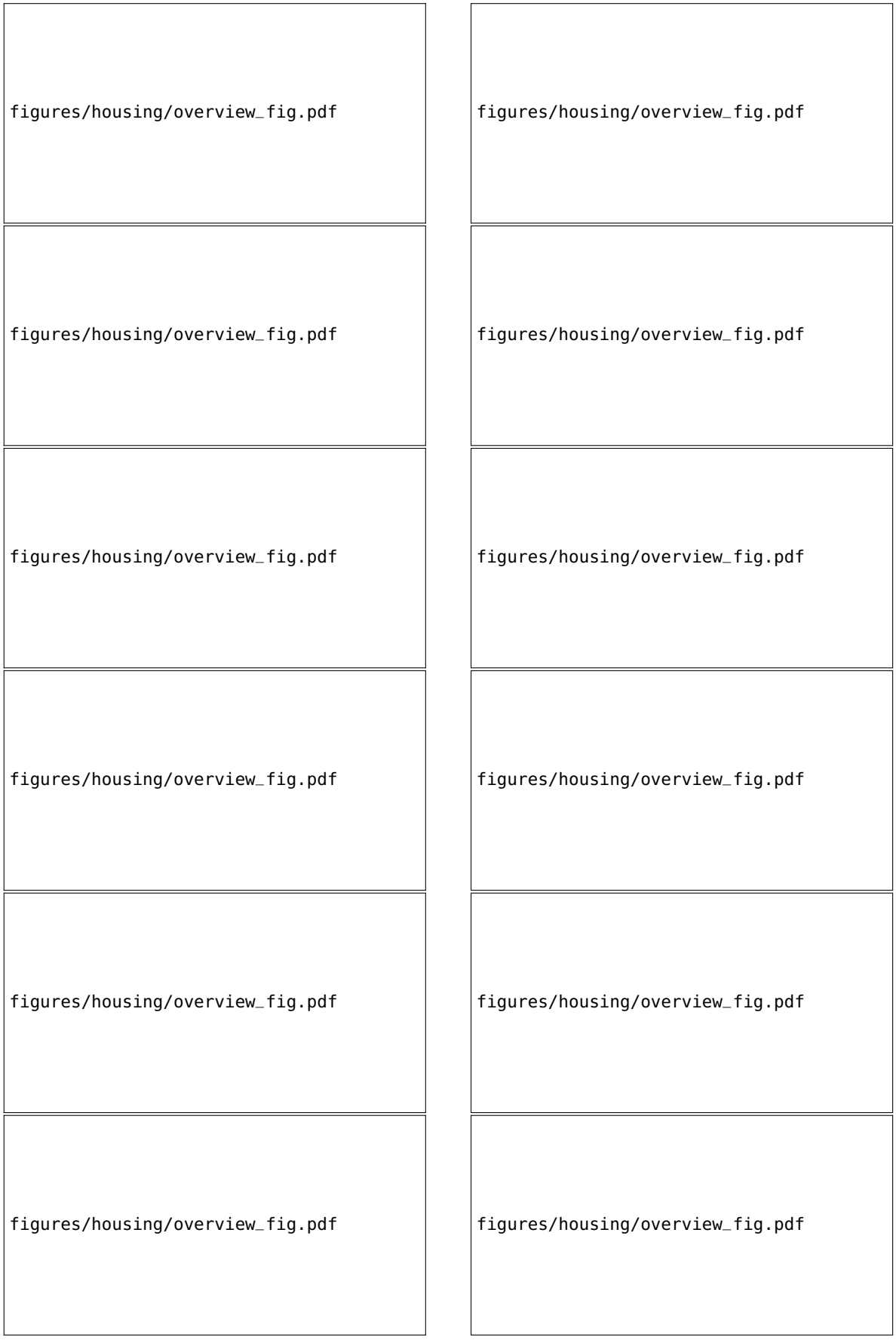


Figure A.6: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

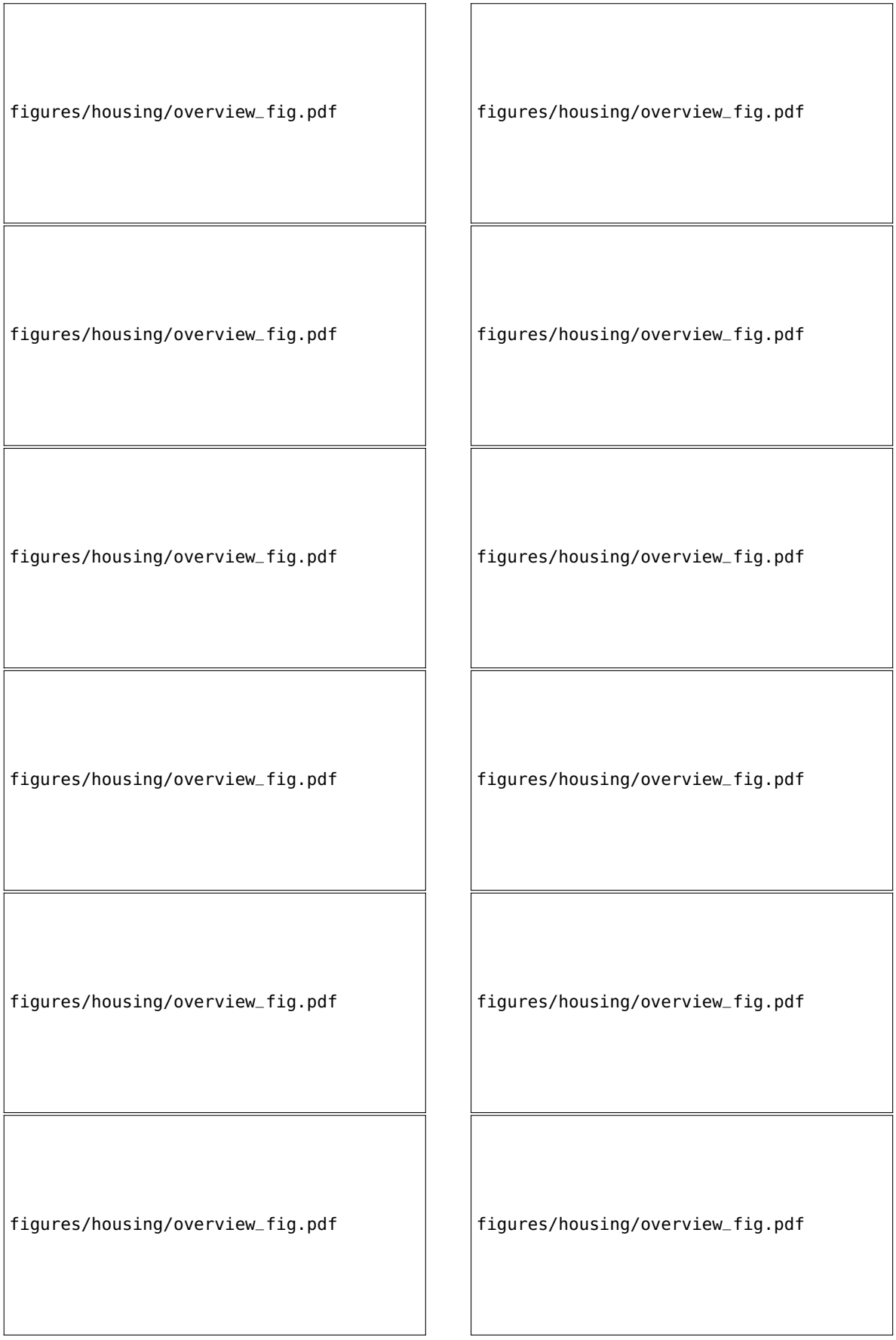


Figure A.7: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

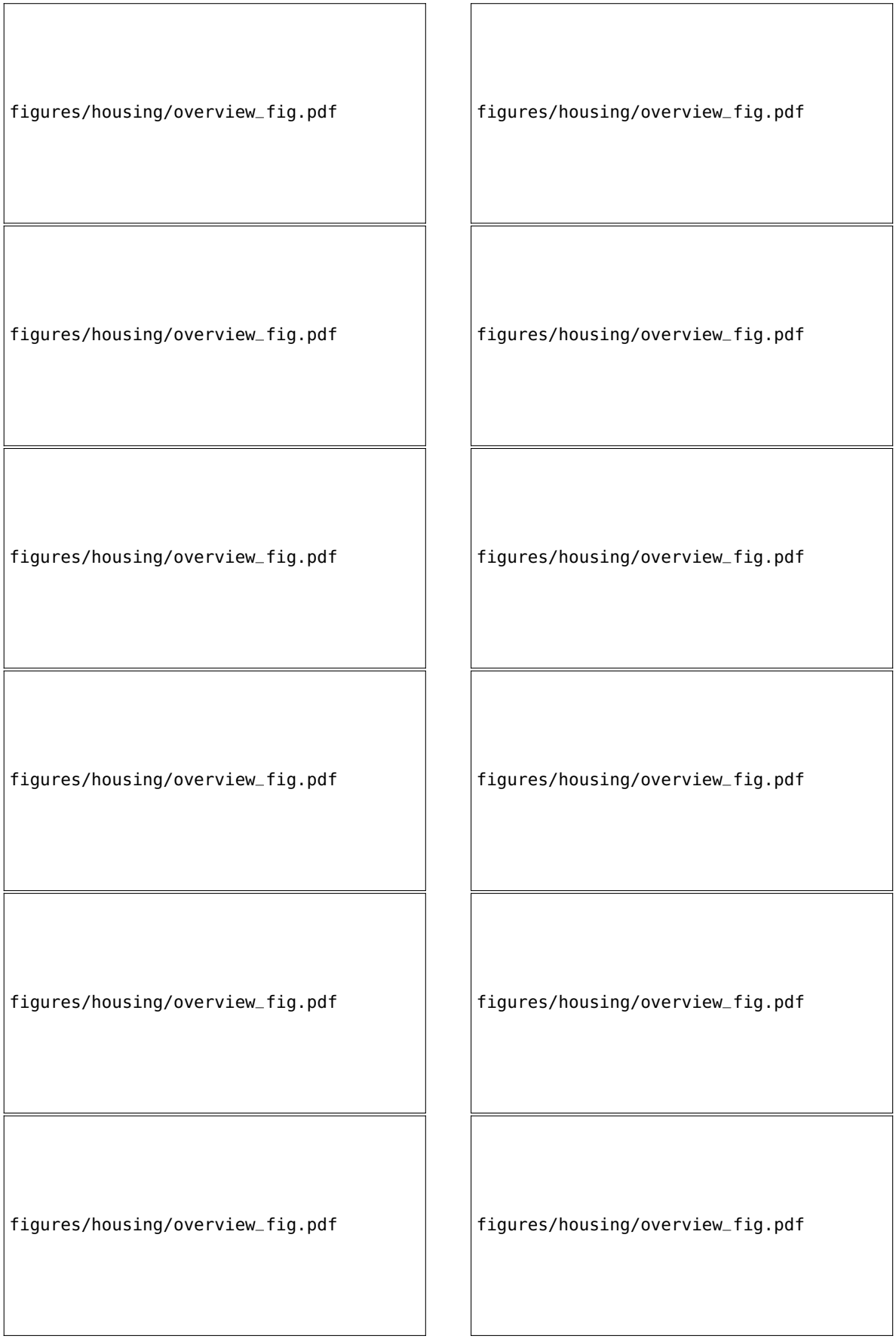


Figure A.8: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

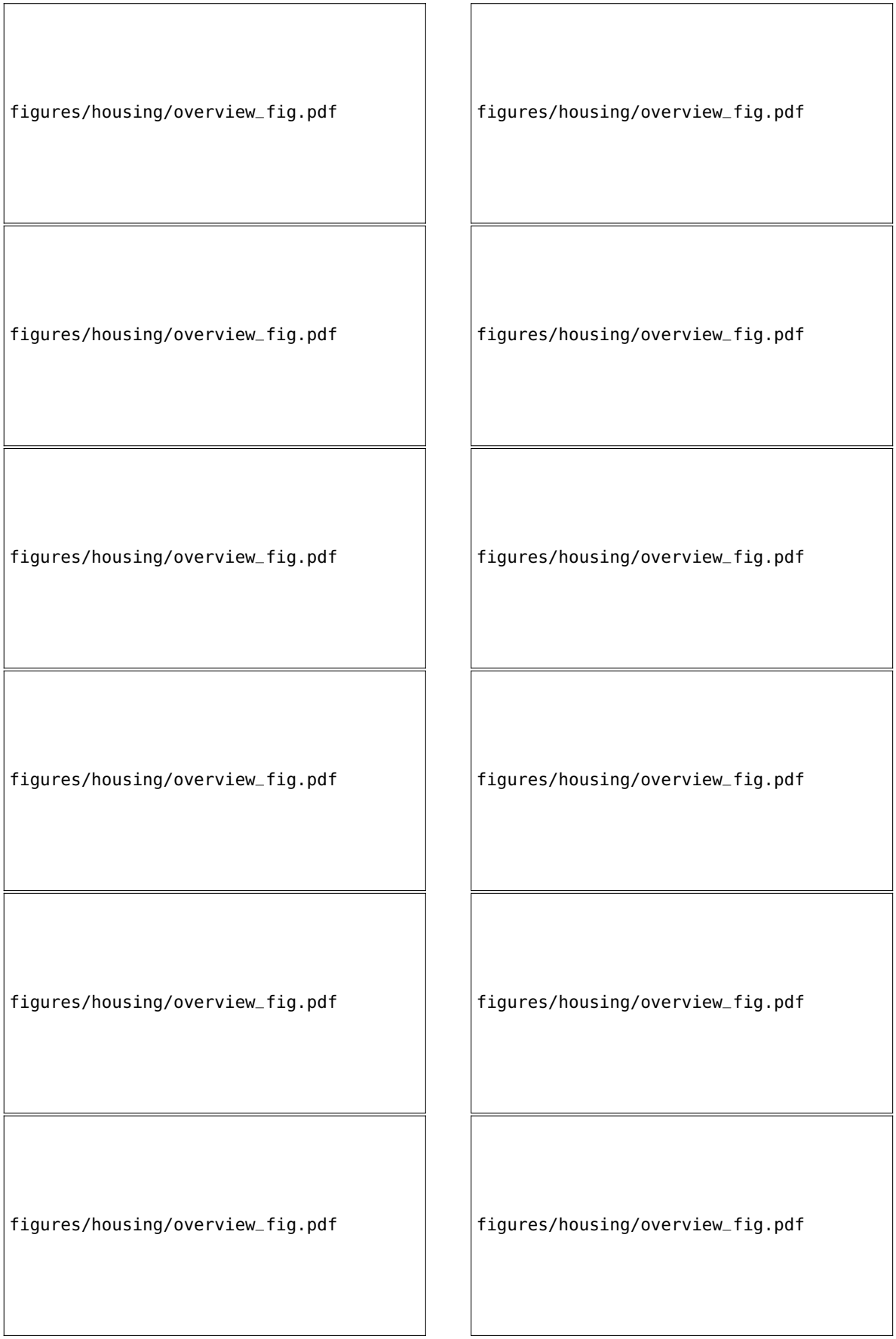


Figure A.9: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

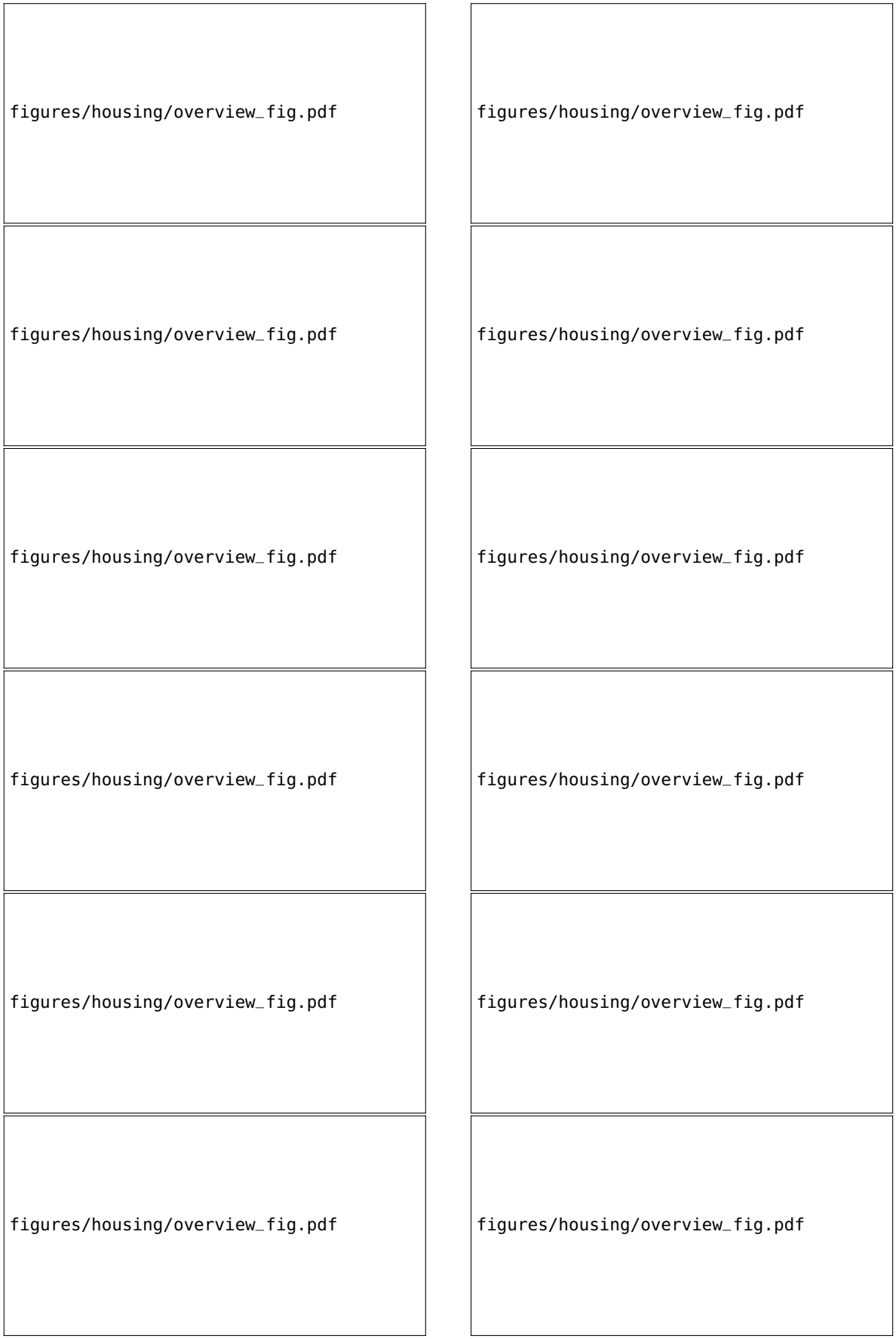


Figure A.10: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

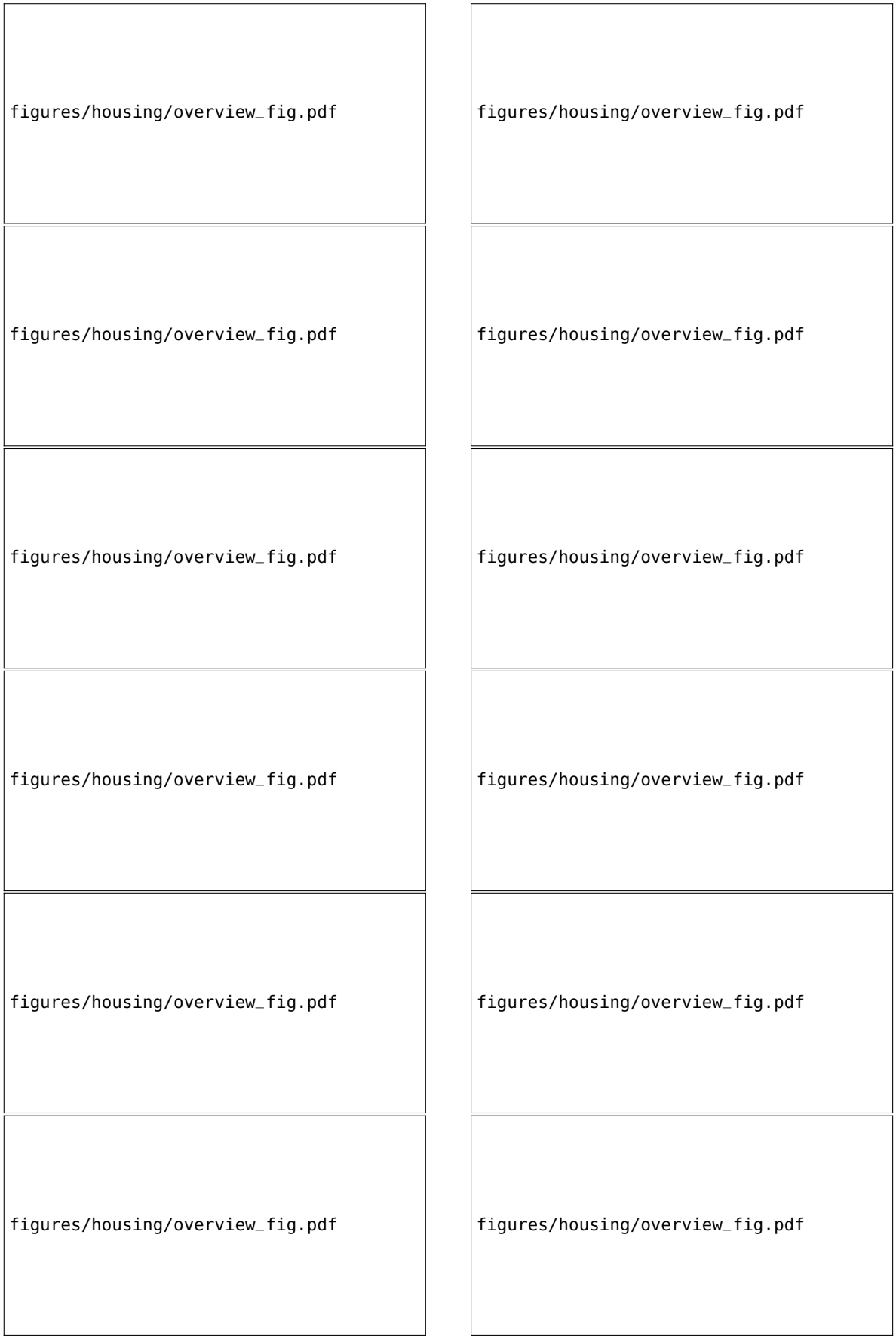


Figure A.11: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

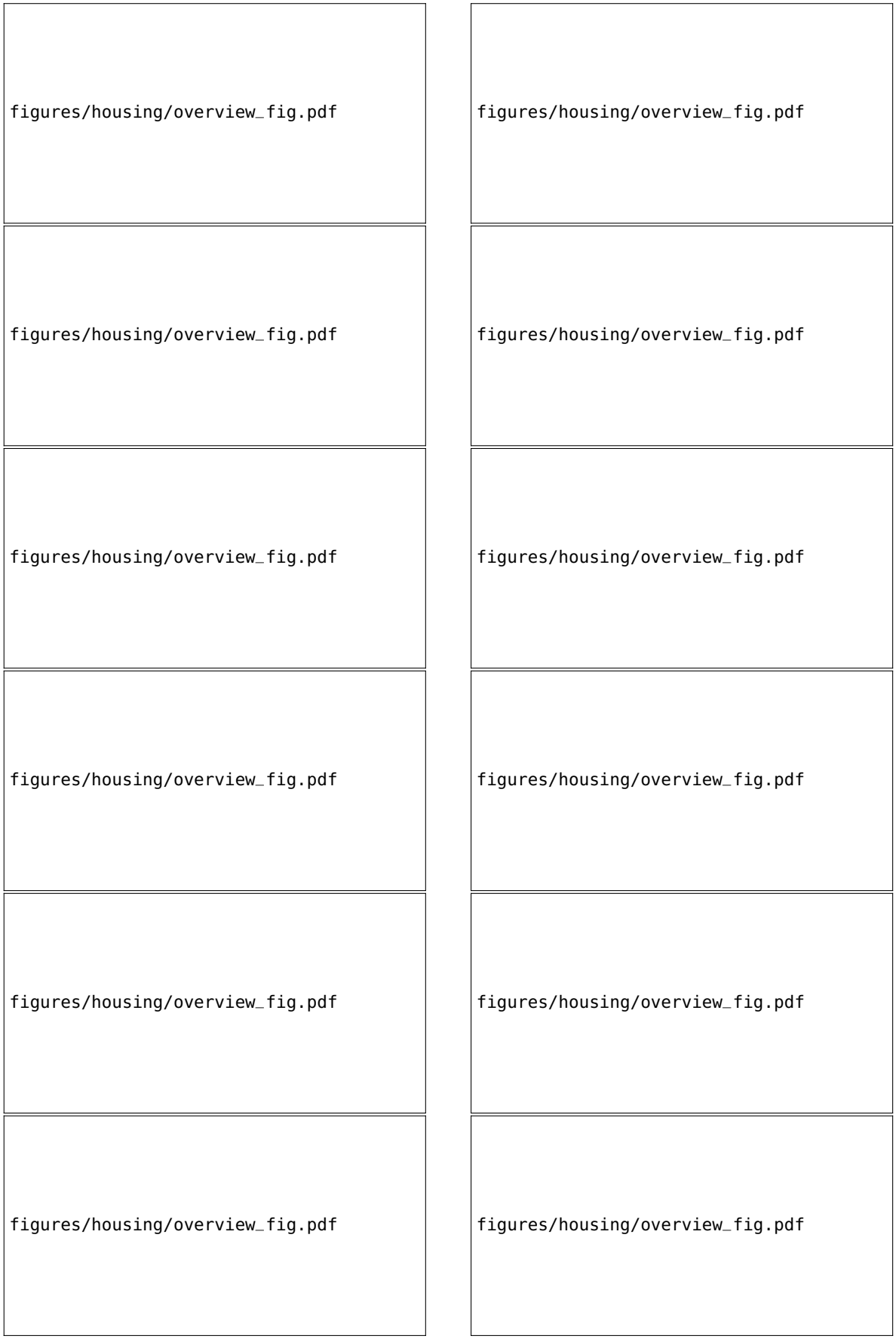


Figure A.12: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

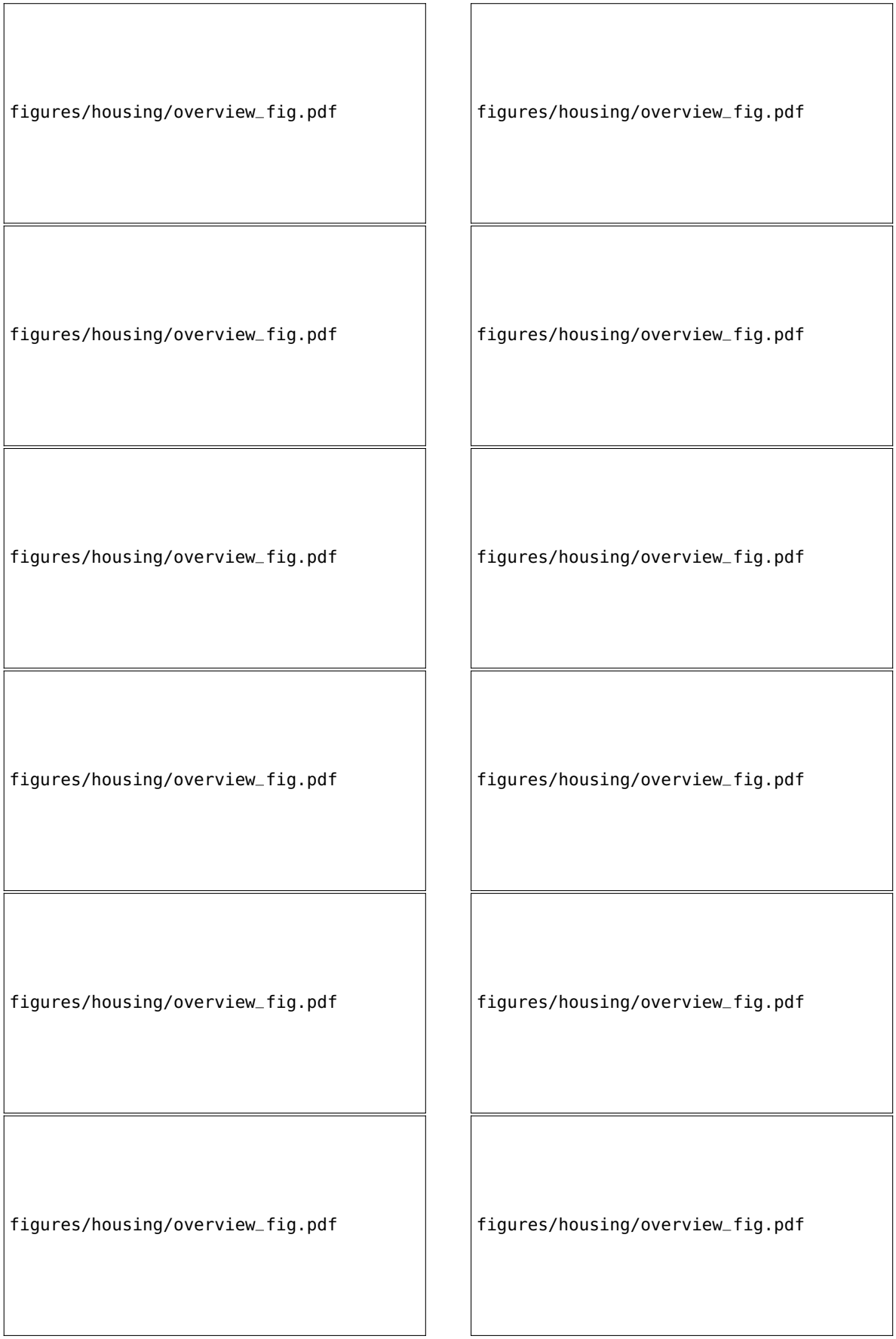


Figure A.13: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

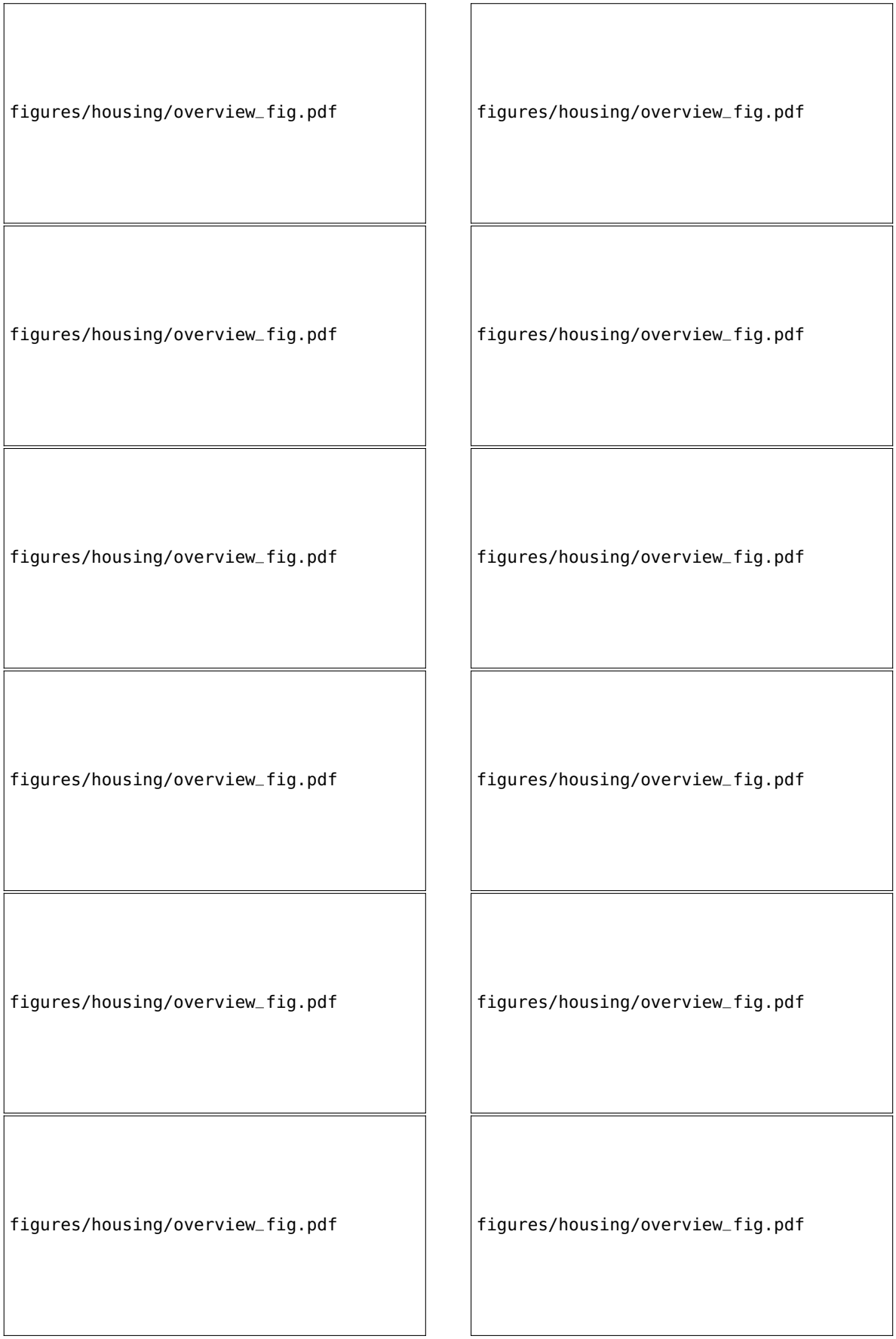


Figure A.14: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

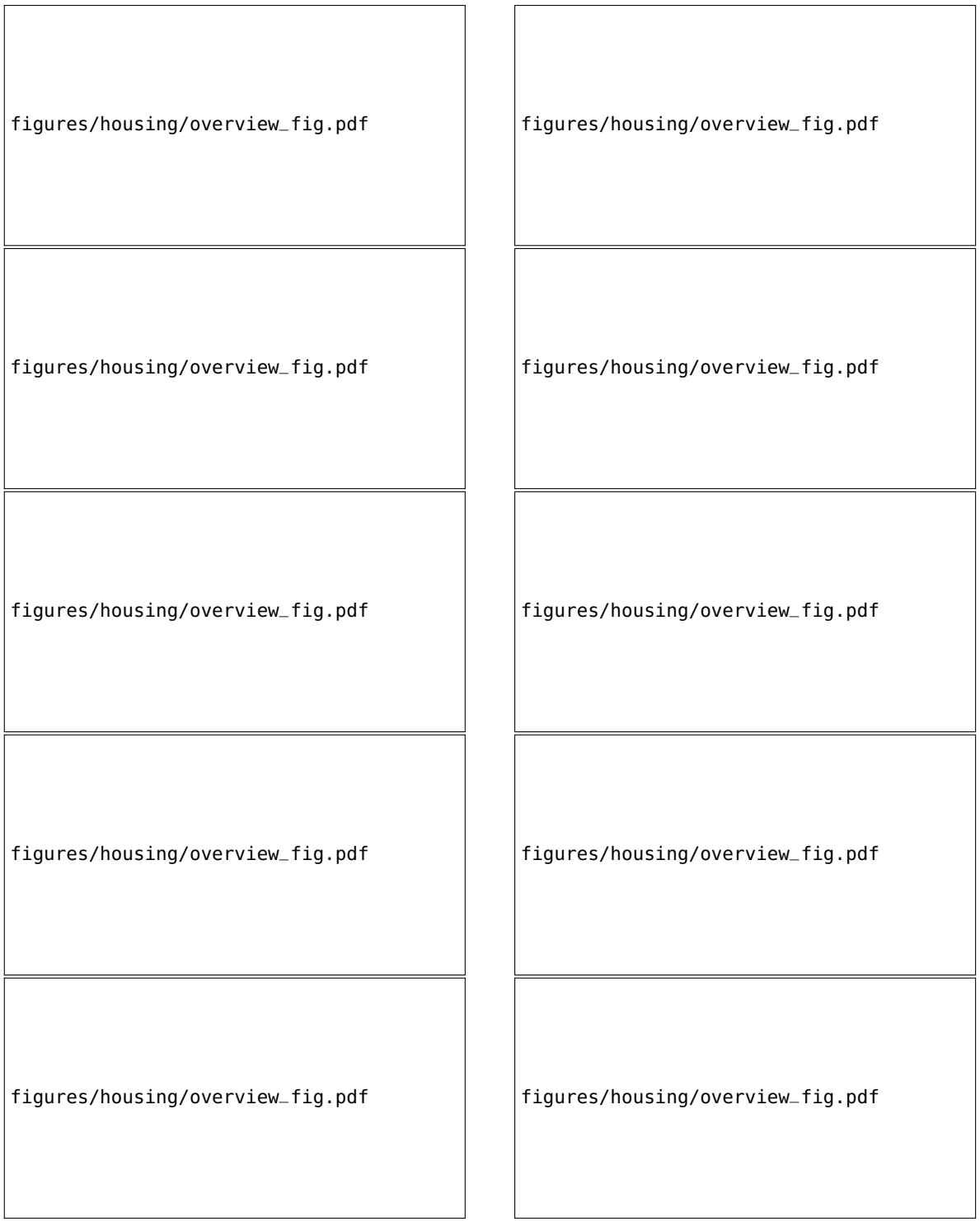


Figure A.15: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

OISSalgsType	GeoLandsdelNr	GeoRegionNr
GeoKommuneNr	GeoPostNr	PostHovedNr
Etage	SognKode	ZoneKode
GisX_WGS84	GisY_WGS84	EjendomsNr
ArealBolig	ArealKaelder	ArealGrund
BeregnetAreal	AntalRum	AntalEtager
ByggeAAr	OmbygningsAAr	EnergiLov
UdenAnnoncering	ProjektSalg	LiggetidAktuel
LiggetidSamlet	Ejerudgift	Bygning_GOP_OpfoerselesAAr
Bygning_GOP_OmbygningsAAr	Bygning_GOP_EjerLavKode	Bygning_GOP_EjerForholdKode
Bygning_GOP_AntalEtagerUKldTag	Bygning_GOP_AntalBoligMedKoekken	Bygning_GOP_AntalBoligUdenKoekken
Bygning_MOP_YdervaegKode	Bygning_MOP_TagdaekningKode	Bygning_MOP_KildeMatrKode
Bygning_IOP_VarmeinstalKode	Bygning_IOP_OpvarmningKode	Bygning_IOP_SuppVarmeKode
Bygning_VOP_VandforsyningKode	Bygning_AGP_Bebygget	Bygning_AGP_HerafAffaldsrum
Bygning_AGP_HerafIndbGarage	Bygning_AGP_HerafIndbCarport	Bygning_AGP_HerafIndbUdhus
Bygning_AGP_HerafUdstue	Bygning_AGP_Overdaekket	Bygning_AHB_SamletBygning
Bygning_AHB_HerafUdvIsolering	Bygning_AHB_KaelderSamlet	Bygning_AHB_KaelderU125
Bygning_AHB_TagSamlet	Bygning_AHB_TagBeboelse	Bygning_AHB_LukkedeOverdaekning
Bygning_AAV_SamletBolig	Bygning_AAV_HerafKaelder	Bygning_AAV_Adgangs
Bygning_AAV_Andet	Bygning_AAV_AAbenOverdaekning	Enhed_Ejendomsnr
Enhed_GOP_AnvendelseKode	Enhed_GOP_AntVaerelseErv	Enhed_GOP_AntToilet
Enhed_GOP_AntBad	Enhed_GOP_EnergiKode	Enhed_AAV_Erhverv
Enhed_AAV_Andet	Enhed_AAV_FeallesAdg	Enhed_AAV_AabenOverdaek
Enhed_AAV_LukketOverdaek	Historisk_SalgsType1	Historisk_SalgsPris1
Historisk_SalgsType2	Historisk_SalgsPris2	Historisk_SalgsType3
Historisk_SalgsPris3	EjdVurdering_VurderingAAr0	EjdVurdering_EjendomsVaerdi0
EjdVurdering_GrundVaerdi0	EjdVurdering_StuehusVaerdi0	EjdVurdering_StueGrundVaerdi0
EjdVurdering_VurderingAAr1	EjdVurdering_EjendomsVaerdi1	EjdVurdering_GrundVaerdi1
EjdVurdering_StuehusVaerdi1	EjdVurdering_StueGrundVaerdi1	EjdVurdering_VurderingAAr2
EjdVurdering_EjendomsVaerdi2	EjdVurdering_GrundVaerdi2	EjdVurdering_StuehusVaerdi2
EjdVurdering_StueGrundVaerdi2	EjdVurdering_VurderingAAr3	EjdVurdering_EjendomsVaerdi3
EjdVurdering_GrundVaerdi3	EjdVurdering_StuehusVaerdi3	EjdVurdering_StueGrundVaerdi3
EjdVurdering_VurderingAAr4	EjdVurdering_EjendomsVaerdi4	EjdVurdering_GrundVaerdi4
EjdVurdering_StuehusVaerdi4	EjdVurdering_StueGrundVaerdi4	Tinglyst_AntEjere
Tinglyst_MindsteAndel	Tinglyst_StoersteAndel	Afstand_Faengsel
Afstand_Hede	Afstand_Hoejspaendingsledning	Afstand_Industri
Afstand_JernbaneSynlig	Afstand_Kirke	Afstand_Kirkegaard
Afstand_Kyst	Afstand_Landingsbane	Afstand_Motorvej
Afstand_MotorvejTilFraKoersel	Afstand_RekreativtOmraade	Afstand_Rigsgraense
Afstand_Sportsanlaeg	Afstand_Strand	Afstand_Vindmoelle
Kommune_Indbyggertal	Kommune_SkatteProcent	Kommune_Vuggestuer
Kommune_Boernehaver	Kommune_IntegreredeInstitutioner	Kommune_FolkeSkoler
Kommune_Grundskyld	dag_i_maaned	maaned
aar	SalgsDato_siden0	Historisk_SalgsDato1_siden0
Historisk_SalgsDato2_siden0	Historisk_SalgsDato3_siden0	HusNr_tal
HusNr_bogstav	SidedoerNummer	Vej
ArealVaegtet_same_as_BeregnetAreal	ByggeAAr_diff	OmbygningsAAr_diff
Energi	Prophet_index	

Table A.1: XXX **TODO!**.

figures/housing/correlations_all.pdf

Figure A.16: XXXX **TODO!**.

Figure A.17: MIC non-linear correlation.

Energy rating label	Code
A2020	2
A2015	4
A2010	6
A2	8
A1	10
A	12
B	20
C	30
D	40
E	50
F2	60
F1	62
F	64
G2	70
G1	72
G	74
H, I, J, K, M	90
NAN	100

Table A.2: Energy rating mapping. If the energy rating is e.g. "A2" this gets the code 8.

figures/housing/Villa_v18_cut_all_Ncols_all_prophet_forecast.png

Figure A.18: The predictions of the Facebook Prophet model trained on square meter prices for owner-occupied apartments sold before January 1st, 2018. The data is down-sampled to weekly bins where the median of each week is used as in input to the Prophet model. This can be seen as black dots in the figure. The model's forecasts for 2018 and 2019 are shown in blue with a light blue error band showing the $1-\sigma$ confidence interval.

figures/housing/Villa_v18_cut_all_Ncols_all_prophet_trends.pdf

Figure A.19: The trends of the Facebook Prophet model trained on square meter prices for owner-occupied apartments sold before January 1st, 2018. In the top plot is the overall trend as a function of year and in the bottom plot is the yearly variation as a function of day of year. It can be seen that the square meter price is higher during the Summer months compared to the Winter months, however, compared to the overall trend this effect is minor ($< 10\%$).

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	226	141	0.1664
2.5	False	201	115	0.1770
5	True	301	90	0.1623
5	False	375	82	0.1786
10	True	318	97	0.1618
10	False	226	56	0.1893
20	True	265	81	0.1626
20	False	687	124	0.1799
∞	**True**	**405**	**110**	**0.1600**
∞	False	94	32	0.2036

Table A.3: Rmse-ejerlejlighed-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	333	75	0.1595
2.5	False	496	57	0.1523
5	True	280	66	0.1606
5	**False**	**734**	**96**	**0.1513**
10	True	367	83	0.1618
10	False	351	52	0.1590
20	True	269	62	0.1609
20	False	333	49	0.1587
∞	True	388	83	0.1595
∞	False	268	42	0.1648

Table A.4: Logcosh-ejerlejlighed-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	293	56	0.1598
2.5	False	814	101	0.1466
5	True	304	68	0.1610
5	False	923	110	0.1468
10	True	266	62	0.1610
10	**False**	**770**	**97**	**0.1450**
20	True	288	65	0.1613
20	False	967	117	0.1467
∞	True	340	72	0.1601
∞	False	807	99	0.1480

Table A.5: Cauchy-ejerlejlighed-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	285	64	0.1628
2.5	False	718	90	0.1517
5	True	257	62	0.1600
5	False	702	91	0.1499
10	True	272	62	0.1601
10	False	771	99	0.1466
20	True	260	61	0.1603
20	False	876	107	0.1486
∞	True	310	69	0.1584
∞	**False**	**973**	**115**	**0.1459**

Table A.6: Welsch-ejerlejlighed-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	229	54	0.1601
2.5	False	304	45	0.1577
5	True	205	54	0.1629
5	False	343	51	0.1549
10	True	257	61	0.1596
10	False	332	47	0.1573
20	True	272	62	0.1608
20	False	403	56	0.1537
∞	True	344	74	0.1578
∞	**False**	**453**	**59**	**0.1527**

Table A.7: Fair-ejerlejlighed-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	458	339	0.1983
2.5	False	844	439	0.1913
5	True	733	478	0.1968
5	False	1126	541	0.1888
10	True	434	310	0.1999
10	False	917	444	0.1884
20	True	398	286	0.2013
20	**False**	**1206**	**575**	**0.1867**
∞	True	730	505	0.1977
∞	False	1264	625	0.1876

Table A.8: Rmse-villa-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	346	223	0.2018
2.5	False	1095	415	0.1877
5	True	618	331	0.1976
5	False	1601	546	0.1847
10	True	506	280	0.1990
10	False	1160	400	0.1873
20	True	445	269	0.2011
20	False	1313	497	0.1876
∞	True	432	258	0.1982
∞	**False**	**2151**	**739**	**0.1842**

Table A.9: Logcosh-villa-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	434	244	0.1991
2.5	False	1007	356	0.1872
5	True	350	208	0.1999
5	False	1130	389	0.1858
10	True	436	240	0.1992
10	False	1183	394	0.1850
20	True	397	242	0.2003
20	**False**	**1514**	**542**	**0.1833**
∞	True	449	257	0.1992
∞	False	1351	470	0.1844

Table A.10: Cauchy-villa-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	867	440	0.1960
2.5	False	835	300	0.1897
5	True	301	184	0.2035
5	False	893	312	0.1878
10	True	341	200	0.2014
10	False	1113	390	0.1869
20	True	338	209	0.2022
20	False	1212	424	0.1875
∞	True	579	321	0.1970
∞	**False**	**1497**	**509**	**0.1837**

Table A.11: Welsch-villa-appendix.

Half-life	$\log_{10}$	N_{trees}	Time [s]	f_{eval}
2.5	True	508	278	0.1956
2.5	False	862	301	0.1882
5	True	506	278	0.1957
5	**False**	**1357**	**462**	**0.1835**
10	True	875	436	0.1946
10	False	954	325	0.1861
20	True	763	402	0.1943
20	False	1256	435	0.1840
∞	True	535	303	0.1973
∞	False	1337	456	0.1844

Table A.12: Fair-villa-appendix.

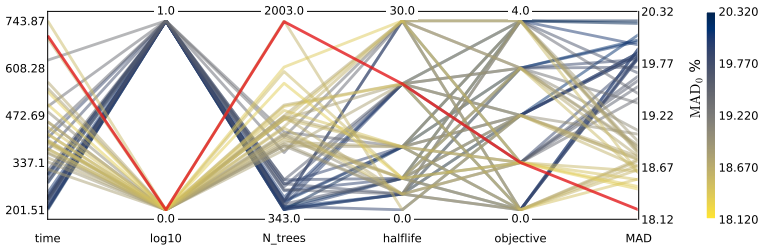


Figure A.20: Hyperparameter optimization results of the housing model for houses. The results are shown as parallel coordinates with each hyperparameter along the x-axis and the value of that parameter on the y-axis. Each line is an event in the 4-dimensional space colored according to the performance of that hyperparameter as measured by MAD_0 from highest MAD in dark blue to lowest AUC in yellow. The single best hyperparameter is shown in red. For the hyperparameter `log10` `0` means False and `1` means True, for `Halftime` ∞ is mapped to `30`, and for `objektive` the functions Cauchy (0), Fair (1), LogCosh (2) SquaredError (3), and Welsch (4) are mapped to the integers in the parentheses.

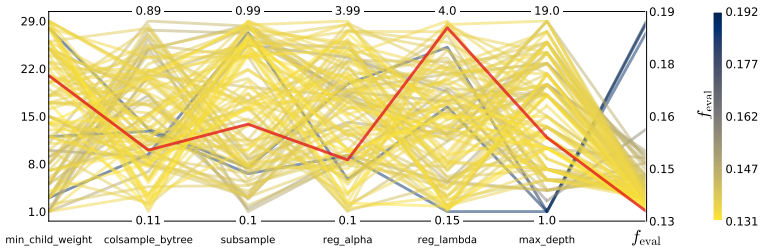


Figure A.21: Hyperparameter optimization results of XGBoost parameters of the housing model for apartments shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

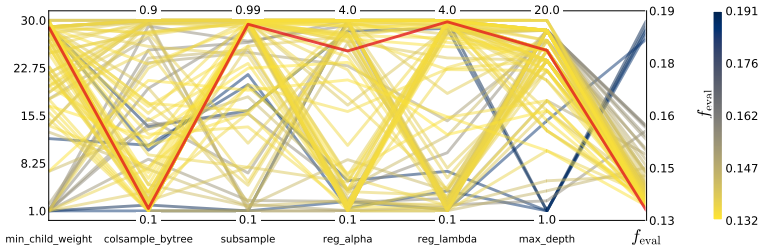


Figure A.22: Hyperparameter optimization results of XGBoost parameters of the housing model for apartments shown as parallel coordinates. Here shown for Bayesian optimization as hyperparameter optimization.

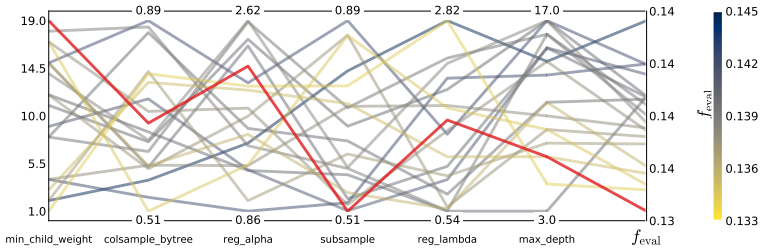


Figure A.23: Hyperparameter optimization results of XGBoost parameters of the housing model for houses shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

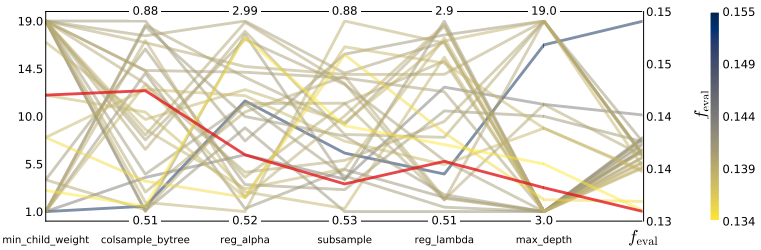


Figure A.24: Hyperparameter optimization results of XGBoost parameters of the housing model for houses shown as parallel coordinates. Here shown for Bayesian optimization as hyperparameter optimization.

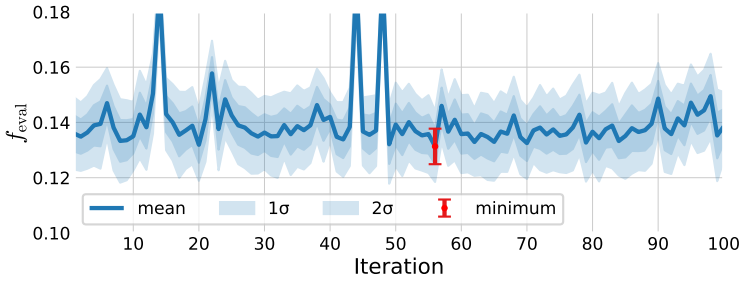


Figure A.25: XXX of the housing model for apartments shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

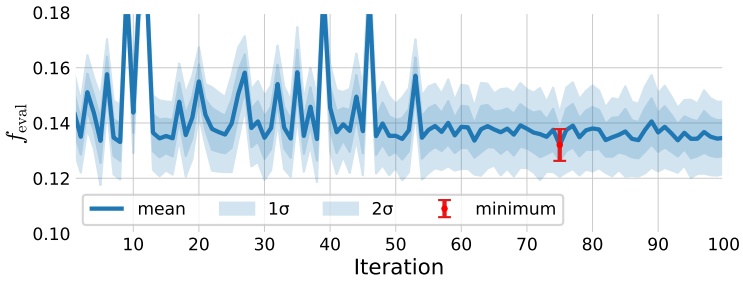


Figure A.26: XXX of the housing model for apartments shown as parallel coordinates. Here shown for Bayesian optimization as hyperparameter optimization.

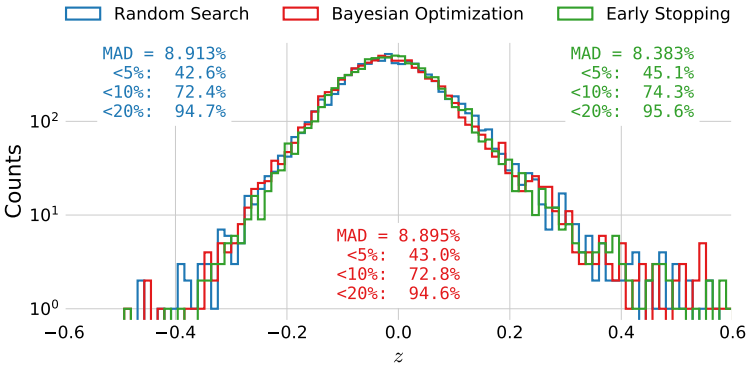


Figure A.27: Histogram of z-values of the XGB-model trained on apartments. The performance after hyperparameter optimization (HPO) using Random Search (RS) is shown in blue, for Bayesian Optimization (BO) in red. After finding the best model, BO in this case, the model is retrained using early stopping, the performance of which is shown in green.

	MAD (%)	$\leq 5\%(\%)$	$\leq 10\%(\%)$	$\leq 20\%(\%)$	μ
Train	6.35	56.22	83.41	97.08	0.00902 ± 0.00068
Test	8.38	45.06	74.32	95.58	-0.00820 ± 0.00115
2019	9.12	42.63	71.36	93.65	0.00297 ± 0.00235

Table A.13: XXX ejer tight

	MAD (%)	$\leq 5\%(\%)$	$\leq 10\%(\%)$	$\leq 20\%(\%)$	μ
Train	15.63	25.65	47.89	75.82	0.04543 ± 0.00080
Test	16.49	24.30	45.77	75.19	0.01686 ± 0.00194
2019	17.17	23.67	44.25	73.54	0.02056 ± 0.00279

Table A.14: XXX villa tight

B. Quarks vs. Gluons Appendix

List of Figures

List of Tables

Bibliography

[1] Advanced Topics in Machine Learning (ATML). URL https://kurser.ku.dk/course/ndak15014u.

[2] Allstate Claims Severity - Fair Loss. URL https://kaggle.com/c/allstate-claims-severity.

[3] Dmlc/xgboost. URL https://github.com/dmlc/xgboost.

[4] HEP meets ML award | The Higgs Machine Learning Challenge. URL https://higgsml.lal.in2p3.fr/prizes-and-award/award/.

[5] The Large Electron-Positron Collider | CERN. URL https://home.cern/science/accelerators/large-electron-positron-collider.

[6] Microsoft/LightGBM. URL https://github.com/microsoft/LightGBM/blob/b3975555d7023fd05f8e56326905fe7b185109de/src/treelearner/serial_tree_learner.cpp#L282.

[7] Scikit-hep/uproot. URL https://github.com/scikit-hep/uproot.

[8] Datashader: Revealing the Structure of Genuinely Big Data. URL https://github.com/holoviz/datashader.

[9] O. . Www.OIS.dk - Din genvej til ejendomsdata. URL https://www.ois.dk/.

[10] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems. URL http://tensorflow.org/.

[11] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning From Data*. AMLBook. ISBN 978-1-60049-006-4.

[12] D. Albanese, S. Riccadonna, C. Donati, and P. Franceschi. A practical tool for maximal information coefficient analysis. 7. ISSN 2047-217X. doi: 10.1093/gigascience/giy032. URL https://doi.org/10.1093/gigascience/giy032.

[13] H. Albrecht, H. Ehrlichmann, T. Hamacher, R. P. Hofmann, T. Kirchhoff, A. Nau, S. Nowak, H. Schröder, H. D. Schulz, M. Walter, R. Wurth, C. Hast, H. Kolanoski, A. Kosche,

A. Lange, A. Lindner, R. Mankel, M. Schieber, T. Siegmund, B. Spaan, H. Thurn, D. Töpfer, D. Wegener, M. Bittner, P. Eckstein, M. Paulini, K. Reim, H. Wegener, R. Eckmann, R. Mundt, T. Oest, R. Reiner, W. Schmidt-Parzefall, W. Funk, J. Stiewe, S. Werner, K. Ehret, W. Hofmann, A. Hüpper, S. Khan, K. T. Knöpfle, M. Seeger, J. Spengler, D. I. Britton, C. E. K. Charlesworth, K. W. Edwards, E. R. F. Hyatt, H. Kapitza, P. Krieger, D. B. MacFarlane, P. M. Patel, J. D. Prentice, P. R. B. Saull, K. Tzamariudaki, R. G. Van de Water, T. S. Yoon, D. Reßing, M. Schmidtler, M. Schneider, K. R. Schubert, K. Strahl, R. Waldi, S. Weseler, G. Kernel, P. Križnič, T. Podobnik, T. Živko, V. Balagura, I. Belyaev, S. Chechelnitsky, M. Danilov, A. Droutskoy, Y. Gershtein, A. Golutvin, G. Kostina, D. Litvintsev, V. Lubimov, P. Pakhlov, F. Ratnikov, S. Semenov, A. Snizhko, V. Soloshenko, I. Tichomirov, and Y. Zaitsev. A model-independent determination of the inclusive semileptonic decay fraction of B mesons. 318(2): 397–404. ISSN 0370-2693. doi: 10.1016/0370-2693(93)90146-9. URL http://www.sciencedirect.com/science/article/pii/0370269393901469.

[14] E. Anderson. The Species Problem in Iris. 23(3):457–509. ISSN 00266493. doi: 10.2307/2394164. URL www.jstor.org/stable/2394164.

[15] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand. Parton fragmentation and string dynamics. 97(2): 31–145. ISSN 0370-1573. doi: 10.1016/0370-1573(83)90080-7. URL http://www.sciencedirect.com/science/article/pii/0370157383900807.

[16] S. R. Armstrong. A Search for the standard model Higgs boson in four jet final states at center-of-mass energies near 183-GeV with the ALEPH detector at LEP. URL http://wwwlib.umi.com/dissertations/fullcit?p9910371.

[17] M. Awad and R. Khanna. Support Vector Regression. In M. Awad and R. Khanna, editors, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, pages 67–80. Apress. ISBN 978-1-4302-5990-9. doi: 10.1007/978-1-4302-5990-9_4. URL https://doi.org/10.1007/978-1-4302-5990-9_4.

[18] R. J. Barlow. *Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences (Manchester Physics Series)*. WileyBlackwell, reprint edition. ISBN 0-471-92295-1. URL http://www.amazon.co.uk/Statistics-Statistical-Physical-Sciences-Manchester/dp/0471922951%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0471922951.

[19] J. T. Barron. A General and Adaptive Robust Loss Function. URL http://arxiv.org/abs/1701.03077.

[20] W. Bartel et al. Experimental study of jets in electron-positron annihilation. 101(1):129–134. ISSN 0370-2693. doi: 10.1016/0370-2693(81)90505-0. URL http://www.sciencedirect.com/science/article/pii/0370269381905050.

[21] E. Becht, C.-A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell. Evaluation of UMAP as an alternative to t-SNE for single-cell data. page 298430, . doi: 10.1101/298430. URL https://www.biorxiv.org/content/10.1101/298430v1.

[22] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell. Dimensionality reduction for visualizing single-cell data using UMAP. 37 (1):38–44, . ISSN 1546-1696. doi: 10.1038/nbt.4314. URL https://www.nature.com/articles/nbt.4314.

[23] J. Bergstra and Y. Bengio. Random Search for Hyper-parameter Optimization. 13:281–305. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2188385.2188395.

[24] C. Bierlich. Rope hadronization, geometry and particle production in pp and pA Collisions. URL https://lup.lub.lu.se/search/ws/files/18474576/thesis.pdf.

[25] Bolighed. Bolighed - usikkerhed i data-vurderingen. URL https://bolighed.dk/om-bolighed/spoergsmaal-og-svar/#boligvaerdi.

[26] L. Breiman. Random Forests. 45(1):5–32. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

[27] R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall Series in Automatic Computation. Prentice-Hall. URL https://cds.cern.ch/record/113464.

[28] E. Brochu, V. M. Cora, and N. de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. URL http://arxiv.org/abs/1012.2599.

[29] R. Brun and F. Rademakers. ROOT — An object oriented data analysis framework. 389(1):81–86. ISSN 0168-9002. doi: 10.1016/S0168-9002(97)00048-X. URL http://www.sciencedirect.com/science/article/pii/S016890029700048X.

[30] A. Buckley, J. Butterworth, S. Gieseke, D. Grellscheid, S. Hoche, H. Hoeth, F. Krauss, L. Lonnblad, E. Nurse, P. Richardson, S. Schumann, M. H. Seymour, T. Sjostrand,

P. Skands, and B. Webber. General-purpose event generators for LHC physics. 504(5):145–233. ISSN 03701573. doi: 10.1016/j.physrep.2011.03.005. URL http://arxiv.org/abs/1101.2599.

[31] C. Burgard. Standard model of physics | TikZ example. URL http://www.texample.net/tikz/examples/model-physics/.

[32] D. Buskulic et al. An investigation of Bdo and Bso oscillation. 322(4):441–458. ISSN 0370-2693. doi: 10.1016/0370-2693(94)91177-0. URL http://www.sciencedirect.com/science/article/pii/0370269394911770.

[33] M. Cacciari, G. P. Salam, and G. Soyez. FastJet user manual. 72(3):1896. ISSN 1434-6044, 1434-6052. doi: 10.1140/epjc/s10052-012-1896-2. URL http://arxiv.org/abs/1111.6097.

[34] S. Catani, Y. L. Dokshitzer, M. H. Seymour, and B. R. Webber. Longitudinally-invariant kt-clustering algorithms for hadron-hadron collisions. 406(1):187–224, . ISSN 0550-3213. doi: 10.1016/0550-3213(93)90166-M. URL http://www.sciencedirect.com/science/article/pii/055032139390166M.

[35] S. Catani, G. Turnock, and B. R. Webber. Jet broadening measures in e+ e- annihilation. B295:269–276, . doi: 10.1016/0370-2693(92)91565-Q.

[36] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. pages 785–794. doi: 10.1145/2939672.2939785. URL http://arxiv.org/abs/1603.02754.

[37] A. Collaboration. Electron efficiency measurements with the ATLAS detector using 2012 LHC proton-proton collision data. 77(3):195, . ISSN 1434-6044, 1434-6052. doi: 10.1140/epjc/s10052-017-4756-2. URL http://arxiv.org/abs/1612.01456.

[38] C. Collaboration. Search for a Higgs boson in the decay channel H to ZZ(*) to q qbar l-l+ in pp collisions at sqrt(s) = 7 TeV. 2012(4):36, . ISSN 1029-8479. doi: 10.1007/JHEP04(2012)036. URL http://arxiv.org/abs/1202.1416.

[39] T. A. Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. 716(1):1–29, . ISSN 03702693. doi: 10.1016/j.physletb.2012.08.020. URL http://arxiv.org/abs/1207.7214.

[40] T. C. Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. 716(1):30–61, . ISSN 03702693. doi: 10.1016/j.physletb.2012.08.021. URL http://arxiv.org/abs/1207.7235.

[41] M. Dam. An upper limit for Br(Z->ggg) from symmetric 3–jet zo hadronic decays. 389(2):405–415. ISSN 0370-2693. doi: 10.1016/S0370-2693(96)01450-5.

[42] A. Diaz-Papkovich, L. Anderson-Trocmé, C. Ben-Eghan, and S. Gravel. UMAP reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts. 15(11). ISSN 1553-7390. doi: 10.1371/journal.pgen.1008432. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6853336/.

[43] Y. L. Dokshitzer, G. D. Leder, S. Moretti, and B. R. Webber. Better Jet Clustering Algorithms. 1997(08):001–001. ISSN 1029-8479. doi: 10.1088/1126-6708/1997/08/001. URL http://arxiv.org/abs/hep-ph/9707323.

[44] S. D. DST. Price Index (EJ14) - Statistics Denmark. URL https://www.dst.dk/en/Statistik/emner/priser-og-forbrug/ejendomme.

[45] S. D. Ellis and D. E. Soper. Successive combination jet algorithm for hadron collisions. 48(7):3160–3166. doi: 10.1103/PhysRevD.48.3160. URL https://link.aps.org/doi/10.1103/PhysRevD.48.3160.

[46] D. et al. Buskulic. A precise measurement of hadrons. 313(3): 535–548. ISSN 0370-2693. doi: 10.1016/0370-2693(93)90028-G. URL http://www.sciencedirect.com/science/article/pii/037026939390028G.

[47] F. Faye. Frederik Faye / deepcalo. URL https://gitlab.com/ffaye/deepcalo.

[48] R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. 7(2):179–188. ISSN 2050-1439. doi: 10.1111/j.1469-1809.1936.tb02137.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x.

[49] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In P. Vitányi, editor, *Computational Learning Theory*, pages 23–37. Springer Berlin Heidelberg. ISBN 978-3-540-49195-8. AdaBoost.

[50] S. L. Glashow. Partial-symmetries of weak interactions. 22(4): 579–588. ISSN 0029-5582. doi: 10.1016/0029-5582(61)90469-2. URL http://www.sciencedirect.com/science/article/pii/0029558261904692.

[51] P. Gras, S. Höche, D. Kar, A. Larkoski, L. Lönnblad, S. Plätzer, A. Siódmok, P. Skands, G. Soyez, and J. Thaler. Systematics of quark/gluon tagging. 2017(7):91. ISSN 1029-8479. doi: 10.1007/JHEP07(2017)091. URL http://arxiv.org/abs/1704.03878.

[52] N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. URL http://arxiv.org/abs/1612.04530.

[53] A. E. Harvey and S. Peters. Estimation Procedures for Structural Time Series Models. doi: 10.1002/for.3980090203.

[54] T. Hastie and R. Tibshirani. Generalized Additive Models: Some Applications. 82(398):371–386. ISSN 01621459. doi: 10.2307/2289439. URL www.jstor.org/stable/2289439.

[55] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer-Verlag, 2 edition. ISBN 978-0-387-84857-0. URL //www.springer.com/la/book/9780387848570.

[56] K. Hornik. Approximation capabilities of multilayer feedforward networks. 4(2):251–257. ISSN 0893-6080. doi: 10.1016/0893-6080(91)90009-T. URL http://www.sciencedirect.com/science/article/pii/089360809190009T.

[57] P. Huber and E. Ronchetti. *Robust Statistics*. Wiley Series in Probability and Statistics. Wiley. ISBN 978-1-118-21033-8. URL https://books.google.dk/books?id=j1OhquR_j88C.

[58] S. Hviid, Juul. Working Paper: A regional model of the Danish housing market. URL http://www.nationalbanken.dk/en/publications/Pages/2017/11/Working-Paper-A-regional-model-of-the-Danish-housing-market.aspx.

[59] F. James and M. Roos. Minuit – a system for function minimization and analysis of the parameter errors and correlations. 10:343–367. doi: 10.1016/0010-4655(75)90039-9.

[60] R. E. Kalman. A new approach to linear filtering and prediction problems. 82:35–45.

[61] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc. URL http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

[62] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. URL http://arxiv.org/abs/1412.6980.

[63] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-Normalizing Neural Networks. URL http://arxiv.org/abs/1706.02515.

[64] A. J. Larkoski, J. Thaler, and W. J. Waalewijn. Gaining (Mutual) Information about Quark/Gluon Discrimination. 2014 (11). ISSN 1029-8479. doi: 10.1007/JHEP11(2014)129. URL http://arxiv.org/abs/1408.3122.

[65] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. 49(4): 764–766. ISSN 0022-1031. doi: 10.1016/j.jesp.2013.03.013. URL http://www.sciencedirect.com/science/article/pii/S0022103113000668.

[66] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 4768–4777. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL http://dl.acm.org/citation.cfm?id=3295222.3295230.

[67] S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent Individualized Feature Attribution for Tree Ensembles - SHAP. . URL http://arxiv.org/abs/1802.03888.

[68] S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent Individualized Feature Attribution for Tree Ensembles. . URL http://arxiv.org/abs/1802.03888.

[69] A. L. Maas. Rectifier nonlinearities improve neural network acoustic models.

[70] L. McInnes and J. Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. URL http://arxiv.org/abs/1802.03426.

[71] T. C. Mills. *Time Series Techniques for Economists / Terence c. Mills*. Cambridge University Press Cambridge [England] ; New York. ISBN 0-521-34339-9 0-521-40574-2. URL http://www.loc.gov/catdir/toc/cam031/89007187.html.

[72] I. Mulalic, H. Rasmussen, J. Rouwendal, and H. H. Woltmann. The Financial Crisis and Diverging House Prices: Evidence from the Copenhagen Metropolitan Area. ISSN 1556-5068. doi: 10.2139/ssrn.3041272. URL https://www.ssrn.com/abstract=3041272.

[73] Particle Data Group et al. Review of Particle Physics. 98 (3):030001. doi: 10.1103/PhysRevD.98.030001. URL https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. 12:2825–2830.

[75] E. Polley and M. van der Laan. Super Learner In Prediction. URL https://biostats.bepress.com/ucbbiostat/paper266.

[76] J. Proriol, J. Jousset, C. Guicheney, A. Falvard, P. Henrard, D. Pallin, P. Perret, and B. Brandl. TAGGING B QUARK EVENTS IN ALEPH WITH NEURAL NETWORKS (comparison of different methods : Neural Networks and Discriminant Analysis). page 27.

[77] A. Purcell. Go on a particle quest at the first CERN webfest. URL https://cds.cern.ch/record/1473657.

[78] S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep Learning with Sets and Point Clouds. URL http://arxiv.org/abs/1611.04500.

[79] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting Novel Associations in Large Data Sets. 334(6062):1518–1524. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1205438. URL http://science.sciencemag.org/content/334/6062/1518.

[80] J. W. Rohlf. *Modern Physics from A to Z*. John Wiley and Sons. ISBN 978-0-471-57270-1.

[81] P. J. Rousseeuw and C. Croux. Alternatives to the Median Absolute Deviation. 88(424):1273–1283. ISSN 0162-1459. doi: 10.1080/01621459.1993.10476408. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1993.10476408.

[82] A. Salam. Weak and electromagnetic interactions. In *Selected Papers of Abdus Salam*, volume Volume 5 of *World Scientific Series in 20th Century Physics*, pages 244–254. WORLD SCIENTIFIC. ISBN 978-981-02-1662-7. doi: 10.1142/9789812795915_0034. URL https://www.worldscientific.com/doi/abs/10.1142/9789812795915_0034.

[83] L. Scodellaro. B tagging in ATLAS and CMS. URL http://arxiv.org/abs/1709.01290.

[84] L. Shapley. A value for n-person games. In *The Shapley Value*, volume 28 of *Annals of Math Studies*, pages 307–317. doi: 10.1017/CBO9780511528446.003.

[85] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands. An Introduction to PYTHIA 8.2. 191:159–177. ISSN 00104655. doi: 10.1016/j.cpc.2015.01.024. URL http://arxiv.org/abs/1410.3012.

[86] P. Skands. Peter Skands.

[87] S. J. Taylor and B. Letham. Forecasting at scale. doi: 10.7287/peerj.preprints.3190v2. URL https://peerj.com/preprints/3190.

[88] i. team. Iminuit – A python interface to minuit. URL https://github.com/scikit-hep/iminuit.

[89] J. Thaler. Report of the Les Houches Quark/Gluon Subgroup. (1):28.

[90] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. 58(1):267–288. ISSN 0035-9246. URL www.jstor.org/stable/2346178.

[91] A. Tikhonov. *On the Stability of Inverse Problems*, volume vol. 39 of *Doklady Akademii Nauk SSSR*.

[92] S. Toghi Eshghi, A. Au-Yeung, C. Takahashi, C. R. Bolen, M. N. Nyachienga, S. P. Lear, C. Green, W. R. Mathews, and W. E. O'Gorman. Quantitative Comparison of Conventional and t-SNE-guided Gating Analyses. 10. ISSN 1664-3224. doi: 10.3389/fimmu.2019.01194. URL https://www.frontiersin.org/articles/10.3389/fimmu.2019.01194/full.

[93] d. L. M. J. van, E. C. Polley, and A. E. Hubbard. Super Learner. 6(1). ISSN 1544-6115. doi: 10.2202/1544-6115.1309. URL https://www.degruyter.com/view/j/sagmb.2007.6.issue-1/sagmb.2007.6.1.1309/sagmb.2007.6.1.1309.xml.

[94] J. J. van der Bij and E. W. N. Glover. Z boson production and decay via gluons. 313(2):237–257. ISSN 0550-3213. doi: 10.1016/0550-3213(89)90317-9. URL http://www.sciencedirect.com/science/article/pii/0550321389903179.

[95] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. 9:2579–2605. ISSN ISSN 1533-7928. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[96] F. van Veen. The Neural Network Zoo. URL http://www.asimovinstitute.org/neural-network-zoo/.

[97] V. Vapnik. Principles of Risk Minimization for Learning Theory. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, pages 831–838. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-222-9. URL http://dl.acm.org/citation.cfm?id=2986916.2987018.

[98] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. page arXiv:1907.10121. URL https://ui.adsabs.harvard.edu/abs/2019arXiv190710121V/abstract.

[99] I. Wallach and R. Lilien. The protein–small-molecule database, a non-redundant structural resource for the analysis of protein-ligand binding. 25(5):615–620. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp035. URL https://academic.oup.com/bioinformatics/article/25/5/615/183421.

[100] S. Weinberg. A Model of Leptons. 19(21):1264–1266. doi: 10.1103/PhysRevLett.19.1264. URL https://link.aps.org/doi/10.1103/PhysRevLett.19.1264.

[101] H. Wickham. Tidy data. 59(10):1–23. ISSN 1548-7660. doi: 10.18637/jss.v059.i10. URL https://www.jstatsoft.org/v059/i10.

[102] M. Wobisch and T. Wengler. Hadronization Corrections to Jet Cross Sections in Deep-Inelastic Scattering. URL http://arxiv.org/abs/hep-ph/9907280.

[103] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep Sets. URL http://arxiv.org/abs/1703.06114.