UNIVERSITY OF COPENHAGEN

A Physicist's Approach To Machine Learning

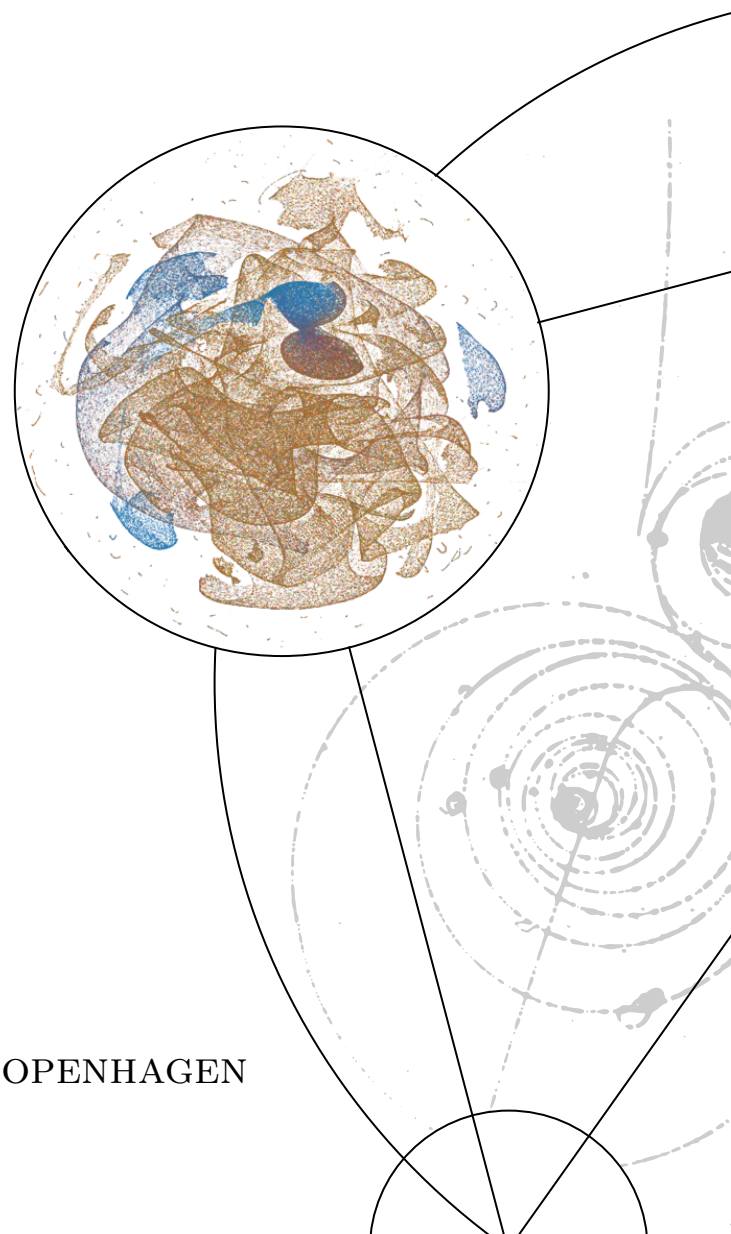
Understanding The Basic Bricks

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

Abstract iii

Table of Contents v

Foreword ix

1 *Introduction* 1

Part I 3

2 *Machine Learning Theory* 5

2.1 *Statistical Learning Theory* . . . 5

2.2 *Supervised Learning* . . . 6

2.3 *Generalization Bound* . . . 7

2.3.1 *Generalization Bound for infinite hypotheses* . . . 9

2.4 *Avoiding overfitting* . . . 10

2.4.1 *Model Regularization* . . . 10

2.4.2 *Cross Validation* . . . 12

2.4.3 *Early Stopping* . . . 13

2.5 *Loss functions* . . . 14

2.5.1 *Evaluation Function* . . . 16

2.6 *Decision Trees* . . . 16

2.6.1 *Ensembles of Decision Trees* . . . 17

2.7 *Hyperparamater Optimization* . . . 19

2.7.1 *Grid Search* . . . 20

2.7.2 *Random Search* . . . 20

2.7.3 *Bayesian Optimization* . . . 21

2.8 *Feature Importance* . . . 22

3 Danish Housing Prices 27
3.1 Data Preparation and Exploratory Data Analysis . . . 28
3.1.1 Correlations . . . 30
3.1.2 Validity of input variables . . . 31
3.1.3 Cuts . . . 33
3.2 Feature Augmentation . . . 33
3.2.1 Time-Dependent Price Index . . . 34
3.3 Evaluation Function . . . 35
3.4 Initial Hyperparameter Optimization . . . 36
3.5 Hyperparameter Optimization . . . 38
3.6 Results . . . 40
3.7 Model Inspection . . . 43
3.8 Multiple Models . . . 45
3.9 Discussion . . . 47
3.10 Conclusion . . . 49

Part II 51

4 Particle Physics and LEP 53
4.1 The Standard Model . . . 53
4.2 Quark Hadronization . . . 54
4.3 The ALEPH Detector and LEP . . . 56
4.4 Jet clustering . . . 58
4.5 The variables . . . 58

5 Quark Gluon Analysis 63
5.1 Data Preprocessing . . . 63
5.2 Exploratory Data Analysis . . . 64
5.2.1 Dimensionality Reduction . . . 66
5.2.2 Correlations . . . 67
5.3 Loss and Evaluation Function . . . 67
5.4 b-Tagging Analysis . . . 68
5.4.1 b-Tagging Hyperparameter Optimization . . . 68
5.4.2 b-Tagging Results . . . 70
5.4.3 b-Tagging Model Inspection . . . 71
5.5 b-Tagging Efficiency . . . 72
5.6 g-Tagging Analysis . . . 74
5.6.1 Permutation Invariance . . . 75
5.6.2 Truncated Uniform PDF . . . 75

5.6.3 *g-Tagging Hyperparameter Optimization* . . . 76
5.6.4 *PermNet* . . . 77
5.6.5 *1D Comparison of LGB and PermNet* . . . 78
5.6.6 *g-Tagging Results* . . . 78
5.7 *g-Tagging Efficiency* . . . 81
5.8 *Generalized Angularities in 3-jet events* . . . 82
5.9 *Gluon splitting* . . . 84
5.9.1 *Variables* . . . 84
5.9.2 *Efficiencies* . . . 86
5.9.3 *Closure Test* . . . 87
5.9.4 *4-jet results* . . . 89
5.10 *Discussion* . . . 90
5.11 *Conclusion* . . . 92

A *Housing Prices Appendix* 93

B *Quarks vs. Gluons Appendix* 121

List of Figures 151

List of Tables 154

Bibliography 155

Foreword

This masters's thesis is part of a 4+4 Ph.D. project (also known as an integrated Ph.D.). The Ph.D. dissertation is about the use of machine learning and deep learning in the field of ancient genomics. In this field ancient DNA is sampled and analysed with the hope of finding patterns and structure in the genome, patterns that were previously unknown. The overall goal is two-fold. On the big scale it is the better understand human history in the broadest sense of the word history. Where did we come from, where did we go? On a much smaller scale, the goal is to understand local history and migration patterns; how did we end up where we did?

It is with this background that this project should be seen: as an introduction to the general use of applied machine learning. Since the Ph.D. continues after this project, the focus here has been on learning and developing methods and tools which will be useful in the latter part of the Ph.D. The master's project originally started in early 2017 and became part of the Ph.D. project in the autumn of the same year.

I would like to thank my supervisor Troels Petersen for his help and time during the project, but most of all for his enthusiasm. I look forward to the continued collaboration during my Ph.D. I would also like to thank the office colleagues, both previous ones such as Stefan Hasselgren, Benjamin Henckel, and Frederik Faye, but also current ones as Helle Leerberg. A special thank goes to Daniel Nielsen for fruitful discussions and general technical help during the recent three years and for letting me lighten him up. I would also like to thank Boligsiden for all their help providing data and valuable insights during the first part of this project. Finally I would like to thank my friends and family for the continued support during this project and hopefully the rest of my Ph.D.

1. Introduction

> *"Begin at the beginning," the King said, gravely, "and go on till you come to an end; then stop."*
>
> — Lewis Carroll, *Alice in Wonderland*

Not only is the title of this project fairly broad, so are the subjects covered in this thesis. The overall goal of this project is to apply machine learning to different datasets and see how well these comparatively new tools might improve classical statistical methods. The project have dealt with two (seemingly) very different datasets: Danish housing prices and Quark-Gluon discrimination in particle physics. The aim of this section is to provide an initial overview of the scope and relationship of the two sub-projects.

The first part of the thesis deals with the problem of estimating housing prices as precisely and accurately as possible. This was the sub-project that was worked on in the beginning of the overall project and worked as an initial introduction to the application of machine learning to real-life datasets. The housing prices dataset thus became the playground in which the subtleties of these new modern tools were examined, where the difference between real life datasets with all its quirks, outliers and bad formatting, and curated toy datasets that works out of the box (such as the famous Iris dataset [14, 48]) were experienced first hand. Since the project started the dataset changed due to a new collaboration with the Danish housing agency Boligsiden where the agreement was, stated shortly, that we would get their data and they would get our results. Boligsiden is a natural collaborator since they are the biggest on the market[1] and have been very helpful in the continuos process of providing data. It should also be noted that they have had no say on the results presented in this thesis. During this initial stage, the author sparred with Simon Gudiksen[2] who also worked on the same dataset, however, both projects were done independently. Where Gudiksen focussed on the prediction of the time evolution of the housing prices using Recurrent Neural Networks (RNN), my work was mostly on the different levels and methods of hyperparameter optimization with some smaller detours into Natural Language Processing (NLP) as an example.

[1] Due to being owned by the "Dansk Ejendomsmæglerforening", The Danish Association of Chartered Estate Agents.

[2] Who afterwards went on to get a job at Boligsiden.

The second part, the Quark-Gluon discrimination in particle physics,

was the main part of the project. Not only was most of the time focussed on this sub-project, it was also the work that generated the highest academic output; an article based on this is in the making. This part dealt with data from the Large Electron Positron collider (LEP) which was an underground particle accelerator at CERN built in 1989 and was discontinued in 2000, where the first phase (LEP1), from 1989-1995, is the sole source of data. As the name suggests it collided electrons and positrons together in what is still the largest electron-positron accelerator ever built [5]. During LEP1 it was primarily the decay channels of the Z-boson that were probed where especially the $Z \to q\bar{q}g$ and $Z \to q\bar{q}gg$ were examined in this thesis. The distributions of these gluon jets and the difference between Data[3] and Monte-Carlo (MC) that are of interests to the theoreticians that develop the MC-models. At first an improved b-tagging algorithm was developed. Here methods and code developed in the hyperparameter optimization process from the housing prices part were used. After the improvement in the b-tagging model, an event-based g-tag model – in comparison to the jet-based b-tagging model – was implemented which allows one to extract useful events of interest. Having found these useful events, one can start looking at how the distributions in the relevant variables differ between Data and MC. Finally XXX **TODO!**.

[3] Where "Data" with capital D refers to the actual, measured data and "data" refers to any arbitrary selection of data.

The thesis is structured such that chapter 2 introduces the needed theoretical Machine Learning (ML) background needed for understanding the methods used throughout the thesis, chapter 3 describes the housing prices subproject as mentioned above, chapter 4 introduces the basic physics in the standard model and the Lund string model which is used throughout the rest of the theses, chapter 5 explains analysis of the main project in this thesis, i.e. the quark gluon analysis, and finally the two chapters **??** and **??** discusses the overall work in this thesis and concludes on it.

The work presented in this thesis is split up into two parts as presented above, however, it should be noted that during the analysis part of the project they were treated not as two different projects but rather as two complementary instances of same underlying problem: teaching computers how to find patterns automatically in high-dimensional data and should thus not be seen as two independent projects. This also highlights another key aspect of this project, that the author does not have any background in particle physics other than rudimentary knowledge stemming from an undergraduate education in general physics.

All of the work presented here is performed by the author unless otherwise noted.

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.

2. Machine Learning Theory

"People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world."

— Pedro Domingos

Machine Learning is the method of teaching computers how to automatically find patterns in (often high-dimensional) data. According to some sceptics machine learning (ML) is just glorified statistics, however, by the same logic physics is just glorified mathematics. In contrary, machine learning is a collection of different subjects located somewhere along the hypothetical line from simple, classical statistics to futuristic artificial intelligence. It includes methods ranging from the well-known statistical methods such as linear regression to the modern, advanced zoo of different neural networks [96] which has seen a plethora of use cases in recent years.

2.1 Statistical Learning Theory

This chapter deals with the theory of ML which Statistical Learning Theory is a subcategory of. Many books are written on the subject where this thesis especially follows the overall notation used in the very accessible introduction in the book Learning From Data [11] and the graduate course Advanced Topics in Machine Learning [1] at the computer science institute[1] at the faculty of Science, University of Copenhagen. Statistical learning theory is the analysis of how to not only find the function, or *hypothesis*, that matches the data best, but also bounding the difference in performance between this hypothesis and the hidden, underlying data generation distribution often only known by Nature.

[1] Datalogisk Institut Københavns Universitet, DIKU.

Overall there are two main branches within machine learning: *supervised* and *unsupervised*[2]. The difference depends on whether or not the data that is trained on is labelled or not. Classic linear regression is an example of the former and linear dimensionality reduction using PCA of the latter. Since unsupervised learning techniques are only used sparsely throughout this project, the main focus will be on supervised learning.

[2] Also known as "self-supervised" or "predictive" learning.

2.2 *Supervised Learning*

In supervised learning we are given a set of N different samples of which we for each one knows M different variables written as the column-vector $\mathbf{x}_i = [x_1, x_2, \ldots, x_M]^T \in \mathcal{X}$ for the ith observation and $\mathcal{X}$ denotes the sample space. All of these samples as a whole is written as the matrix $\mathbf{X}$ with the individual observations $\mathbf{x}_i$ transposed and stacked on top of each other $\mathbf{X} = [\mathbf{x}_0^T, \mathbf{x}_1^T, \ldots, \mathbf{x}_N^T]$ such that row i in $\mathbf{X}$ corresponds to observation i. In the case of supervised learning we also have the label y of each sample $y \in \mathcal{Y}$ where $\mathcal{Y}$ denotes the label space. In the case where y is a real number $\mathcal{Y} = \mathbb{R}$ the problem is said to a *regression* problem, e.g. predicting the price of a house. On the contrary, if $\mathcal{Y}$ is binary such that $\mathcal{Y} = \{0, 1\}$ then the problem is said to be a (binary) *classification* problem[3] e.g. predicting whether or not a particle is a quark or not.

Without any loss of generality let the focus for now be on classification. The goal is to find the underlying "true" function $f : \mathcal{X} \mapsto \mathcal{Y}$ that gives the correct label y for each observation $\mathbf{x}$. This function, however, is unknown and cannot perfectly be found. Although it is impossible to find f it is possible to learn an approximation of it, h, based on some training observations $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$. The optimal hypothesis, h^*, is chosen among a set of K candidate hypotheses $\mathcal{H} = \{h_1, h_2, \ldots, h_K\}$, and hopefully h^* will be a good approximation of f: $h^* \approx f$. A schematic overview of this process can be seen in Figure 2.1.

How can one make sure that h^* really is a good approximation of f? That is where statistical learning theory comes into play. From a statistical standpoint we are interested in modelling the unknown joint probability $P(\mathbf{x}, y)$ over $\mathcal{X}$ and $\mathcal{Y}$. We assume that $\mathcal{D}_{\text{train}}$ is independent and identically distributed (*iid*)[4] from $P(\mathbf{x}, y)$ and thus want to find the hypothesis whose predictions $h(\mathbf{x}) = \hat{y}$ matches the conditional probability distribution $P(y|\mathbf{x})$ as well as possible.

To quantify the statement "as well as possible" in the previous paragraph, we define the loss function ℓ which measures the loss for predicting $\hat{y}$ instead of y: $\ell(\hat{y}, y) = \ell(h(\mathbf{x}), y) \in \mathbb{R}^+$. Given ℓ we now introduce the method of (empirical) risk minimization [97] and the expected loss[5]:

$$L(h) = \mathbb{E}\left[\ell(h(\mathbf{x}), y)\right] = \int \ell(h(\mathbf{x}), y)\, dP(\mathbf{x}, y). \tag{2.1}$$

The optimal hypotheses h^* is the hypothesis which minimizes the expected loss $L(h)$. However, the joint probability distribution $P(\mathbf{x}, y)$ is unknown and we are thus left with the empirical loss[6] of h on $\mathcal{D}_{\text{train}}$:

$$\hat{L}(h, S) = \frac{1}{N} \sum_{i=1}^{N} \ell(h(\mathbf{x}_i), y_i), \tag{2.2}$$

which is an approximation of $L(h)$ based on the training data available. Now the optimal hypothesis h^* can defined:

$$h^* = \arg\min_{h \in \mathcal{H}} \hat{L}(h, S). \tag{2.3}$$

[3] If $\mathcal{Y} \subset \mathbb{Z}$ then it would be a multiclass classification problem.

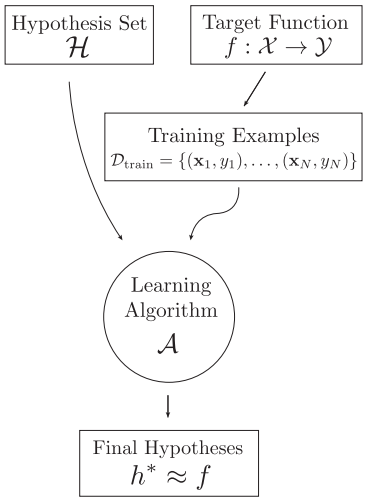


Figure 2.1: Schematic overview of the learning problem and how to find the optimal hypothesis h^* to approximate f given the training data $\mathcal{D}_{\text{train}}$.

[4] This is one of the two key assumptions of statistical learning theory, the other being that future events are coming from the same distrubution as the one that generated the past events. These assumptions are sometimes called the *PAC* assumptions where PAC is an abbreviation for Probably, Approximately Correct.

[5] Also called expected error or the out-of-sample error.

[6] Also called empirical error.

2.3 Generalization Bound

In section 2.2 the method of selecting the optimal hypotheses h^* out of the total set of candidate hypothesis $\mathcal{H}$ was sketched. However, there is still no guarantee that h^* will work well, that is to say that the *generalization error* $G(h)$ might be big:

$$G(h) = \hat{L}(h, S) - L(h). \tag{2.4}$$

The generalization error is thus the difference between the expected error $L(h)$ and the empirical error $\hat{L}(h, S)$. It describes the loss in performance of our chosen model compared to the optimal, yet hidden, model. Since $P(\mathbf{x}, y)$ is unknown, $G(h)$ cannot be computed, however, it is possible to bound this error using statistical learning theory. To do so, the union bound and Hoeffding's (one-sided) inequalities are introduced.

Lemma 1 (The Union Bound). *For any finite or countably infinite sequence of events $E_1, E_2, \ldots$ (not necessarily independent):*

$$\mathbb{P}\left\{\bigcup_{1\leq i} E_i\right\} \leq \sum_{1\leq i} \mathbb{P}\left\{E_i\right\}. \tag{2.5}$$

The union bound, in simple terms, states that the probability of any one of n events happening is less than or equal to the sum of the individual probabilities of the events happening. As an example, let $E_1 = \{2, 4, 6\}$ be the event that a die rolls an even number and $E_2 = \{4, 5, 6\}$ be the event that a die rolls a number larger than or equal to 4. Then $\mathbb{P}\{E_1 \cup E_2\} = \mathbb{P}\{2, 4, 5, 6\} \leq \mathbb{P}\{E_1\} + \mathbb{P}\{E_2\}$.

Lemma 2 (The one-sided Hoeffding's inequalities). *Let $Z_1, \ldots, Z_N$ be independent random variables each belonging to the $[0, 1]$ interval such that $\mathbb{P}\{Z_i \in [0, 1]\} = 1$ and $\mathbb{E}[Z_i] = \mu$ for all i, then for every $\epsilon > 0$:*

$$\mathbb{P}\left\{\frac{1}{N}\sum_{i=1}^{N} Z_i - \mu \geq \epsilon\right\} \leq e^{-2N\epsilon^2} \tag{2.6}$$

and

$$\mathbb{P}\left\{\mu - \frac{1}{N}\sum_{i=1}^{N} Z_i \geq \epsilon\right\} \leq e^{-2N\epsilon^2}. \tag{2.7}$$

When using the union bound on equation (2.6) and equation (2.7) we arrive at Hoeffding's (two) sided inequality:

Lemma 3 (The two-sided Hoeffding's inequality). *Let $Z_1, \ldots, Z_N$ be independent random variables each belonging to the $[0, 1]$ interval such that $\mathbb{P}\{Z_i \in [0, 1]\} = 1$ and $\mathbb{E}[Z_i] = \mu$ for all i, then for every $\epsilon > 0$:*

$$\mathbb{P}\left\{\left|\frac{1}{N}\sum_{i=1}^{n} Z_i - \mu\right| \geq \epsilon\right\} \equiv \mathbb{P}\left\{|\hat{\mu} - \mu| \geq \epsilon\right\} \leq 2e^{-2N\epsilon^2}, \tag{2.8}$$

where we have defined the empirical average of Z to be $\hat{\mu}$: $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{N} Z_i$

Assuming that the loss $\ell(\hat{y}, y)$ is bounded in the $[0,1]$ interval[7], $\ell(\hat{y}, y) \in [0,1]$ for all $(\hat{y}y)$, we can bound the generalization error $G(h)$ by letting $Z_i = \ell(\hat{y}_i, y_i) = \ell(h(\mathbf{x}_i), y_i)$ be the loss of h in sample $(\mathbf{x}_i, y_i)$. By comparing Lemma 3 and equation (2.2) we see that $\hat{\mu} = \hat{L}(h, S)$, and similar for equation (2.1): $\mu = L(h)$. We then see that the generalization error is bounded:

$$\mathbb{P}\left\{|G(h)| \geq \epsilon\right\} = \mathbb{P}\left\{\left|\hat{L}(h,S) - L(h)\right| \geq \epsilon\right\} \leq 2e^{-2N\epsilon^2}. \tag{2.9}$$

This equation provides a bound on the difference between the empirical loss and the expected loss. The generalization bound can be rewritten in terms of δ:

$$\delta \equiv 2e^{-2N\epsilon^2} \in (0,1) \quad \Rightarrow \quad \epsilon = \sqrt{\frac{\ln \frac{2}{\delta}}{2N}}. \tag{2.10}$$

Theorem 1 (Hoeffding's inequality for a single hypothesis)**.** *Assume that ℓ if bounded in the $[0,1]$ interval, then for a single hypothesis h and any $\delta \in (0,1)$ we have:*

$$\mathbb{P}\left\{\left|\hat{L}(h,S) - L(h)\right| \geq \sqrt{\frac{\ln \frac{2}{\delta}}{2N}}\right\} \leq \delta. \tag{2.11}$$

Equation (2.11) can be read as the probability of the generalization error being larger than $\sqrt{\frac{\ln \frac{2}{\delta}}{2N}}$ is δ or less, or, similarly, that with probability greater than $1-\delta$:

$$\left|\hat{L}(h,S) - L(h)\right| \leq \sqrt{\frac{\ln \frac{2}{\delta}}{2N}}. \tag{2.12}$$

This is a powerful result relating the performance for a (fixed) hypothesis h with the number of samples, N. We see that a higher N yields a tighter bound on the generalization error.

There is a big assumption of this derivation: that the hypothesis h cannot depend on the sample S and thus has to be chosen before seeing the data. We say that h has to be *fixed*. Of course the term machine learning indicates that some kind of learning is taking place: exactly as seen previously where we wanted to find the optimal hypotheses h^* out of all the possible ones $\mathcal{H}$. For now assume that $\mathcal{H}$ is finite and consists of K hypotheses: $|\mathcal{H}| = K$. We thus have $[h_1, h_2, \ldots, h_K]$ hypotheses which we test simultaneously and where Hoeffding's inequality is true for each of them leading to the following theorem:

Theorem 2 (Hoeffding's inequality for a finite set of hypotheses candidates)**.** *Assume that ℓ is bounded in the $[0,1]$ interval and that $|\mathcal{H}| = K$. Then for any $\delta \in (0,1)$ we have:*

$$\mathbb{P}\left\{\exists h \in \mathcal{H} : \left|\hat{L}(h,S) - L(h)\right| \geq \sqrt{\frac{\ln \frac{2K}{\delta}}{2N}}\right\} \leq \delta. \tag{2.13}$$

[7] Which it is for classification, however, it can be extended in a similar fashion for regression.

Proof. The proof begins by denoting h_i as the event where:

$$\left|\hat{L}(h_i, S) - L(h_i)\right| \geq \sqrt{\frac{\ln \frac{2}{\delta'}}{2N}},$$

and then taking the union bound (the first inequality below) followed by applying Hoeffding's inequality to each part in the sum (the second inequality):

$$\begin{aligned}
\mathbb{P}\left\{\exists h \in \mathcal{H} : \left|\hat{L}(h, S) - L(h)\right| \geq \sqrt{\frac{\ln \frac{2}{\delta'}}{2N}}\right\} & \\
&= \mathbb{P}\left\{\bigcup_{h\in\mathcal{H}} \left|\hat{L}(h, S) - L(h)\right| \geq \sqrt{\frac{\ln \frac{2}{\delta'}}{2N}}\right\} \\
&\leq \sum_{h\in\mathcal{H}} \mathbb{P}\left\{\left|\hat{L}(h, S) - L(h)\right| \geq \sqrt{\frac{\ln \frac{2}{\delta'}}{2N}}\right\} \\
&\leq \sum_{h\in\mathcal{H}} \delta' \\
&= K\delta'.
\end{aligned}$$

By making the substitution $\delta = K\delta'$ we arrive at equation (2.13). □

As we did in equation (2.12), equation (2.13) can also be read as with probability greater than $1 - \delta$ then for all $h \in \mathcal{H}$:

$$\left|\hat{L}(h, S) - L(h)\right| \leq \sqrt{\frac{\ln \frac{2K}{\delta}}{2N}}. \tag{2.14}$$

This bound is looser than the one for only a single hypothesis by a factor $\ln K$, however, this holds for the optimal hypothesis h^*.

2.3.1 *Generalization Bound for infinite hypotheses*

Section 2.3 dealt with the case of a single hypotheses h and a finite set of candidate hypotheses $h \in \mathcal{H}, |\mathcal{H}| = K$. When K goes towards infinity the generalization bound goes to infinity and the bound becomes useless. However, even simple models such as a linear classifier[8] that predicts $\hat{y} = 1$ when the dot product $\mathbf{w}^T\mathbf{x}$ is positive and $\hat{y} = -1$ when it is negative, has $|\mathcal{H}| = \infty$. Since an infinite number of hypotheses $h(\mathbf{w})$ exists, assuming we allow $\mathbf{w}$ to take any real values as is almost always the case, $\mathcal{H}$ is infinite.

To solve this obvious problem with the Hoeffding inequality, we introduce[9] the Vapnik-Chervonenkis (VC) generalization bound. The VC-bound is based on the so-called VC-dimension of the hypothesis space $\mathcal{H}$: $d_{\text{VC}}(\mathcal{H}) = d_{\text{VC}}$. The VC-dimension is a measure of the complexity of the hypothesis space, the degrees of freedom of the model so to say. For example the VC-dimension of the M-dimensional linear classifier defined above[10] is $d_{\text{VC}} = M$ for $\{\mathbf{x}, \mathbf{w}\} \in \mathbb{R}^M$.

Theorem 3 (VC Generalization Bound). *Let $\mathcal{H}$ be a hypotheses class with VC-dimension: $d_{\text{VC}}(\mathcal{H}) = d_{\text{VC}}$. Then with probability at least $1 - \delta$:*

[8] Also known as the perceptron. Often includes a constant offset b as well which is omitted here for brevity. The functional form of this function is $f(x) = \text{sign}(\mathbf{w}^T\mathbf{x})$.

[9] For proof, see: Abu-Mostafa et al. [11]

[10] In the general case when the offset b is included: $d_{\text{VC}} = M + 1$.

$$L(h) \leq \hat{L}(h,S) + \sqrt{\frac{8}{N}\ln\left(\frac{4}{\delta}\left((2N)^{d_{\text{VC}}}+1\right)\right)}. \tag{2.15}$$

Equation (2.15) states that the out of sample error $L(h)$ is bounded from above by the empirical error $\hat{L}(h,S)$ and the $\sqrt{\cdot}$ which is related to the complexity of the hypothesis space $\mathcal{H}$, the number of samples N and the certainty δ. We will call this model complexity penalty $\Omega(N,\mathcal{H},\delta)$:

$$\Omega(N,\mathcal{H},\delta) = \sqrt{\frac{8}{N}\ln\left(\frac{4}{\delta}\left((2N)^{d_{\text{VC}}(\mathcal{H})}+1\right)\right)}. \tag{2.16}$$

As the hypothesis space complexity d_{VC} grows, the model complexity penalty increases but it is more likely that $\mathcal{H}$ contains a strong hypothesis. This relationship is called the approximation-estimation or the *bias-variance* tradeoff. When the model is too simple to properly fit the complexity in the data it is called *underfitting*[11], when the model is so complex that it starts fitting the inherent noise in the data it is called *overfitting*[12]. The loss as a function of model complexity gives the characteristic curve illustrated in Figure 2.2. As the model complexity increases the training loss decreases. Initially also the validation loss decreases, but at some point the behavior of model on the validation set worsens and the loss increases; overfitting happens.

[11] Here the error from $\hat{L}(h,S)$ dominates.

[12] Here the error from $\Omega(N,\mathcal{H},\delta)$ dominates.

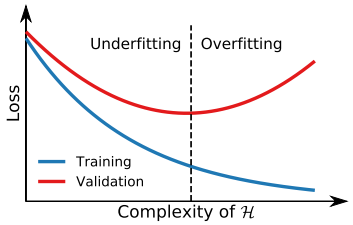


Figure 2.2: Illustration of the empirical loss as a function of model complexity. The training error is shown in blue and validation error in red.

2.4 *Avoiding overfitting*

Avoiding overfitting is one of the most important issues in machine learning. By now, most modern machine learning algorithms have the inherent model complexity needed for overfitting and thus it has to be managed. Due to the importance of the issue, a number of different methods preventing or reducing overfitting exists. Most of them are complementary of each other but can be taken advantage of in a combination. In this section model regularization will be introduced in subsection 2.4.1, cross validation in subsection 2.4.2, and early stopping in subsection 2.4.3.

2.4.1 *Model Regularization*

One of the earliest methods developed for preventing overfitting was model regularization. A. N. Tikhonov [91] was one of the first to describe this method in 1943. In particular, regularization was used to solve *ill posed* linear regression problems. Regular linear regression problems refer to minimizing the residual sum of squares written in matrix form as:

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{\text{LS}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y}-\mathbf{f}(\mathbf{X})\|_2^2 \\ \mathbf{f}(\mathbf{X}) = \mathbf{X}\boldsymbol{\beta},\end{aligned} \tag{2.17}$$

where $\mathbf{y}$ is the vector of values we are trying to predict[13], $\mathbf{X} \in \mathbb{R}^{N\times M}$

[13] E.g. the prices of a collection of houses.

the matrix of input variables, $\hat{\boldsymbol{\beta}}_{\text{LS}}$ the vector of unknown coefficients[14] of the linear least squares (LS) model $\mathbf{f}$, and $\|\cdot\|_2$ is the normal Euclidean norm. In general, the p-norm is defined as:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{N} |x_i|^p\right)^{1/p}. \tag{2.18}$$

[14] Here excluding the constant offset β_0 which can be included trivially.

Differentiating the objective $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$ with respect to (w.r.t.) $\boldsymbol{\beta}$ and setting the derivative equal to 0 to find the minimum[15] yields the solution for $\boldsymbol{\beta}$:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = 0 \Rightarrow \\ \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \Rightarrow \\ \hat{\boldsymbol{\beta}}_{\text{LS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \end{aligned} \tag{2.19}$$

[15] When checking the double derivative wrt. $\boldsymbol{\beta}$ it is seen that this really is a minimum and not a maximum (or saddle point).

However, this solution for $\hat{\boldsymbol{\beta}}_{\text{LS}}$ is only valid when $\mathbf{X}^T\mathbf{X}$ is invertible, i.e. $\mathbf{X}$ has to be full rank [55]. If this is not the case, the problem is said to be ill posed. Tikhonov solved this problem by adding an extra term to the minimization problem, which we will call Ω for simplicity. For a specific choice of Ω, one gets:

$$\hat{\boldsymbol{\beta}}_{\text{L}_2} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{y} - \mathbf{f}(\mathbf{X})\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \right\}, \tag{2.20}$$

where $\lambda \geq 0$ is the regularization strength. This is the so-called L_2-regularization, also known as ridge regression for linear problems. For ridge regression[16] the corresponding solution for $\boldsymbol{\beta}$ is:

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{2.21}$$

[16] Here $\hat{\boldsymbol{\beta}}_{\text{L}_2}$ is the solution for any general function $\mathbf{f}$ and $\hat{\boldsymbol{\beta}}_{\text{ridge}}$ is the specific solution for a linear function $\mathbf{f}$, where linear is w.r.t. to the model parameters $\boldsymbol{\beta}$.

where $\mathbf{I}$ is the identity matrix[17]. This extra term, $\Omega = \lambda\|\boldsymbol{\beta}\|_2^2$, acts as a shrinkage factor on the coefficients of $\boldsymbol{\beta}$. Looking at equation (2.20), we see that this is the Lagrangian form of the equivalent problem:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{\text{L}_2} &= \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{f}(\mathbf{X})\|_2^2 \\ \text{subject to :} \quad &\sum_{i=1}^{N} \beta_i^2 = \|\boldsymbol{\beta}\|_2^2 \leq t, \end{aligned} \tag{2.22}$$

[17] The $\lambda\mathbf{I}$ in equation (2.21) also acts as a conditioner on the problem in the sense that it reduces the condition number of the matrix to be inverted turning the ill-posed problem to a well-behaved one.

for some $t \geq 0$ with a one-to-one mapping between λ and t. We thus see that L_2 regularizes the coefficients of $\boldsymbol{\beta}$ to have some maximal norm. The effect of the regularization is controlled by λ, where $\hat{\boldsymbol{\beta}}_{\text{L}_2} \to \hat{\boldsymbol{\beta}}_{\text{LS}}$ for $\lambda \to 0$ and $\hat{\boldsymbol{\beta}}_{\text{L}_2} \to \mathbf{0}$ for $\lambda \to \infty$. An example of this can be seen in Figure 2.3.

Here $N = 9$ datapoints were randomly generated such that the x-values are evenly spaced from 0 to 2 and $y \sim \mathcal{N}(\mu = 0, \sigma = 10)$. They were then fit with a 9-order polynomial by minimizing equation (2.20) for different values of λ. Here we see the regularizing effect of λ, going from $\lambda = 0$ in blue which fits all points[18] with a high degree of variance and a wildly oscillatory pattern to $\lambda = 10$

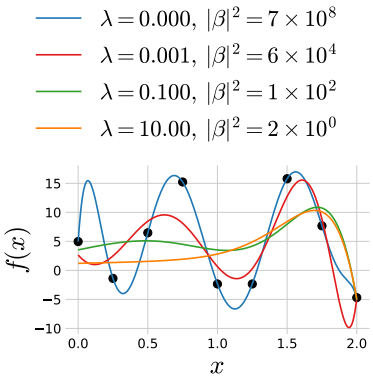


Figure 2.3: Effect of tuning the regularization strength λ in ridge regression.

[18] Since the order of the polynomial is the same as the number of datapoints.

in orange which is mostly flat in most of the interval. Also note in the legend how the norm of the fit parameters also decrease with λ as expected. This is a great example of the *bias-variance* tradeoff mentioned in subsection 2.3.1, where bias refers the error the model makes when it is not advanced enough to fit the overall trend in the data (underfitting) and variance refers to the error the model makes when it starts to fit spurious noisy fluctuations in the training set which are not present in the validation set (overfitting). In this example $\lambda = 0$ is clearly overfitting the data whereas $\lambda = 10$ is an example of underfitting.

Other values of the regularization function Ω exists, for example the L_1-penalty:

$$\Omega = \lambda \|\boldsymbol{\beta}\|_1, \tag{2.23}$$

where the 1-norm, also known as Manhattan norm, is used. In the case of linear problems the L_1-penalty leads to Lasso regression introduced by Tibshirani [90] in 1996. As with the L_2-penalty, the L_1-penalty also regularizes the coefficients of $\boldsymbol{\beta}$, however, this loss leads to sparse[19] solutions. An illustration of this can be seen in Figure 2.4 and Figure 2.5 where the constraint regions of $\boldsymbol{\beta}$ is shown in red and the grey ellipses are the contours of the non-constrained problem. Notice how the intersection of the contour lines and the constrain region leads to $\beta_1 \neq 0, \beta_2 \neq 0$ for the L_2-penalty whereas it leads to the sparse solution $\beta_1 = 0, \beta_2 \neq 0$ for the L_1-penalty. This is a general pattern seen for L_p-penalties for $p \leq 1$ [55].

Overall, model regularization is heavily used in modern machine learning algorithms. In general, the function or the so-called *objective function* $\mathcal{L}$ they are trying to minimize is:

$$\mathcal{L}(h) = \hat{L}(\ell, h, S) + \Omega(h) \tag{2.24}$$

where $\hat{L}$ is the empirical loss and Ω is the regularization penalty. As can be seen from the above discussion, choosing the right value for the regularization strength is fundamental problem in model regularization. How to choose a suitable value for λ via cross validation is discussed in subsection 2.4.2 and the choice of the training loss function ℓ in section 2.5.

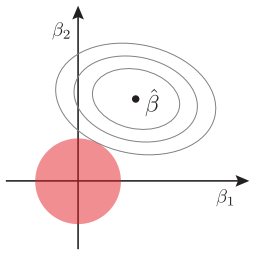


Figure 2.4: Sketch of the minimization problem defined in equation (2.22), i.e. for a L_2-penalty. The constrain region shown in red is defined as $\beta_1^2 + \beta_2^2 \leq t$ for L_2 in $2D$-space and the contours of the unconstrained solution is shown with grey, dashed lines.

[19] Meaning that a number of the β_i coefficients are 0, the number depending on λ.

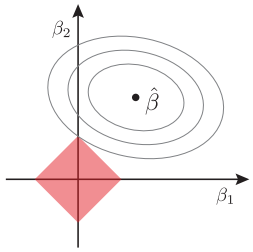


Figure 2.5: Sketch of the similar minimization problem defined in Figure 2.4 for the L_1-penalty. The constrain region shown in red is defined as $|\beta_1| + |\beta_2| \leq t$ for L_1 in $2D$-space and the contours of the unconstrained solution is shown with grey, dashed lines.

2.4.2 *Cross Validation*

In general we want to be able to estimate the performance[20] of the developed model. Since evaluating the model on the data it was already trained on would give a biased estimate of the performance, we need an unbiased method of doing so. The easiest way of doing so would be to set a fraction of the data aside, e.g. 20 %, train on the remaining part and then evaluate the performance on the data set aside. Splitting the data up like this would provide us with a training (data)set and a test set in a 80 : 20 ratio. This would then yield an unbiased performance estimate when evaluation the performance on the test set. However, if one then needed to compare two different models – e.g. with different values of regularization strength λ – and

[20] "Performance" is used here as the word for the general metric to be optimized for, whether or not this metric should be maximized or minimized.

choose the best one, as is often the case, this method would not work since we would choose the model with the best performance on the test set which can be seen as training on the test set and thus it has "tainted" the purity of the test set. To avoid this, an additional split is made such that we get a training set, a validation set, and a test set, where you often see a $80 : 10 : 10$ ratio. The two models can then be compared on the validation set and the performance of the chosen model can be estimated from the test set.

This way of splitting up the data has some clear benefits and is thus also often used. There is a drawback, however, and that is that we are not fully utilizing a lot of the data in this way. Basically 20 % of the data are only used to provide a single number of performance and does not necessarily allow an uncertainty or confidence interval of this measurement to be calculated. Thus other methods of estimating model performance are developed where one of the most used and well-known are the k-fold cross validation (CV). Here the entire dataset is split up into k chunks which are randomly drawn subsamples (without replacement). In the first iteration, the model is trained on the first $k-1$ subsamples and evaluated on the last k subsample. In the second iteration the evaluation subsample is a new one. This process is continued k times until all samples in the dataset have been trained and evaluated on [55]. For an illustration of this, see Figure 2.6. The process yields k estimates of the performance of the model which can then be averaged to form a single performance number and the variability of the performance can even be gauged[21]. The disadvantage of k-fold CV is that the performance estimate is now slightly biased, however, this effect is generally very small. The biggest disadvantage is the computational burden related to doing k-fold CV where $k \gg 1$. A compromise often used in applied machine learning is $k = 5$ which is also what is used in this project.

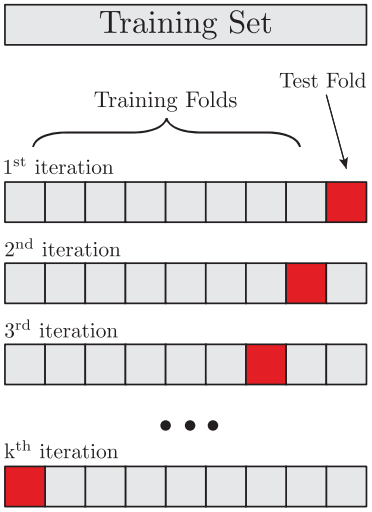


Figure 2.6: k-fold cross validation.

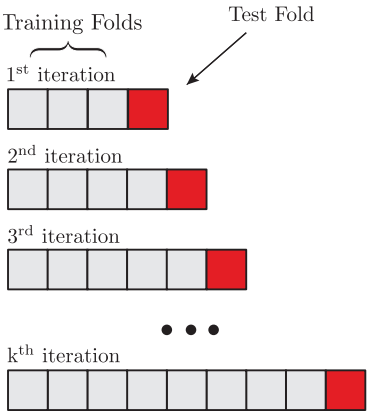


Figure 2.7: k-fold cross validation for time series data.

Special care has to be taken when dealing with time series data. Here the problem of "data leakage" is often introduced inadvertently. Data leakage is when the model is exposed to information from the test set that it was not supposed to be exposed to. In the case of time series data, if the data is split by the usual k-fold CV, then each subsample contains events from all times and the model does not learn how to predict future events. To circumvent this problem, a special type of k-fold CV for time series data has to be used. Here all samples up to a specific time, eg. all houses sold before 2018, is used for training and then the model is evaluated on the performance of samples after the event, e.g. houses sold in 2018. For an illustration of this, see Figure 2.7.

2.4.3 *Early Stopping*

Most modern machine learning models are trained iteratively. This is the case for both (boosted) decision trees and neural networks, both of which are used in this project. Iteratively here means that

[21] Special care has to be taken here since the k different performance values are not independent.

the model starts off with an initial guess of the parameters of the model and then by looking at the data "learns" a new and better set of values. The question then become: for how long should the model be allowed to continue training?

This is a another example of the bias-variance tradeoff. The model should be trained long enough to be able to capture the complexity inherent in the data but also should not train for so long that it starts to overfit the data. Even though Figure 2.2 was just an illustration of the bias variance tradeoff, it is also something that is seen in real data and can be taken advantage of through *early stopping*. Early stopping is the process of monitoring the loss for the training set $\mathcal{D}_{\text{train}}$ and validation set $\mathcal{D}_{\text{val}}$. As mentioned in subsection 2.4.2, the model is only fitted on $\mathcal{D}_{\text{train}}$ but the performance on $\mathcal{D}_{\text{val}}$ is also measured. Whenever the validation loss starts to increase, the training of the model should be terminated.

To avoid stopping due to a single noise-induced outlier which terminated the process, one often uses *patience* in the early stopping process: if the loss has not decreased since the last minimum after *patience* number of iterations, then terminate the process. Early stopping is thus an easy way of avoiding overfitting for iteratively trained models only requiring a validation set.

2.5 *Loss functions*

How we evaluate the performance of a model is of course very important since it defines the metric for the problem; what is good and what is bad? Obviously this depends on whether or not a classification problem or a regression problem is dealth with. Let us for know focus on the latter. Say that a house is estimated to cost 2 million DKK (M.kr.) but was sold for 4 M.kr.. Compare this to a house that was estimated to cost 8 M.kr. but was sold for 6 M.kr. In both cases the price was 2 M.kr. wrong, but does this mean that both predictions are equally good or bad? The first case was a factor of 2 off, whereas the prediction in the second case was only 33% off compared to the true value. The first case underestimated the price whereas the second overestimated; should this have any importance?

As it should be clear by now the choice of loss function ℓ is of utmost importance. It is also not a problem that can be solved by computers, it is problem-specific, and has to be defined manually. The choice of loss function is what is called a *hyperparameter*, the optimization of which is further discussed in section 2.7. The most common choice of loss function is by far the Squared Error (SE):

$$\ell_{\text{SE}}(y, \hat{y}) = (y - \hat{y})^2 , \tag{2.25}$$

where y is the true value and $\hat{y}$ is the predicted one. Squared Error has the advantage that it is differentiable everywhere, an effect that is both needed for many statistical derivations but also a requirement for some machine learning models. The disadvantage is that it gives too much weight to outliers since every deviation away from the

truth is squared. In contrast to this, there is the Absolute Error (AE) defined as:

$$\ell_{\text{AE}}(y, \hat{y}) = |y - \hat{y}|. \tag{2.26}$$

For AE, outliers have a lot smaller weight since it deals with the absolute value of the deviation and not the squared deviation. However, this comes at a price; AE is not differentiable at every point: at $y - \hat{y} = 0$ the derivative of the absolute value function is un-defined. Many functions have been invented trying to deal with these problems. For a more general discussion of loss functions, see e.g. Barron [19]. Six different loss functions (for regression problems) have been investigated in this project. In addition to SE and AE also the LogCosh, Cauchy [19], Welsch [19] and Fair [2] loss functions are used:

$$\begin{aligned}
\ell_{\text{LogCosh}}(y, \hat{y}) &= \log\left(\cosh\left(y - \hat{y}\right)\right) \\
\ell_{\text{Cauchy}}(y, \hat{y}) &= \log\left(\frac{1}{2}\left(\frac{y - \hat{y}}{c}\right)^2 + 1\right) \\
\ell_{\text{Welsch}}(y, \hat{y}) &= 1 - \exp\left(-\frac{1}{2}\left(\frac{y - \hat{y}}{c}\right)^2\right) \\
\ell_{\text{Fair}}(y, \hat{y}) &= c^2\left(\frac{|y - \hat{y}|}{c} - \log\left(\frac{|y - \hat{y}|}{c} + 1\right)\right).
\end{aligned} \tag{2.27}$$

The above loss functions share some similarities with AE, and in addition to this they are all (twice) differentiable functions. They are shown in Figure 2.9. They are shown for only positive values of $y - \hat{y}$ since they are symmetric in $y - \hat{y}$. Notice how SE quickly grows very large compared to the others. Absolute Error has a kink at $y - \hat{y} = 0$ as the only one of the functions. Welsch is bounded in the interval $[0, 1)$. The derivative of both LogCosh and Fair goes toward one when $y - \hat{y}$ goes towards infinity, whereas it goes to zero for the Cauchy loss. A priori it is almost impossible to know which one of these loss functions performs best for a specific data set, so they have to be treated as hyperparameters.

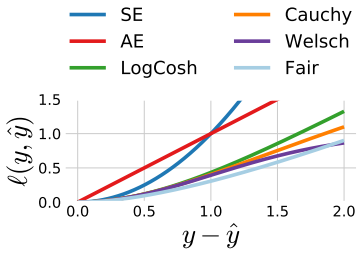


Figure 2.8: Zoom in of Figure 2.9.

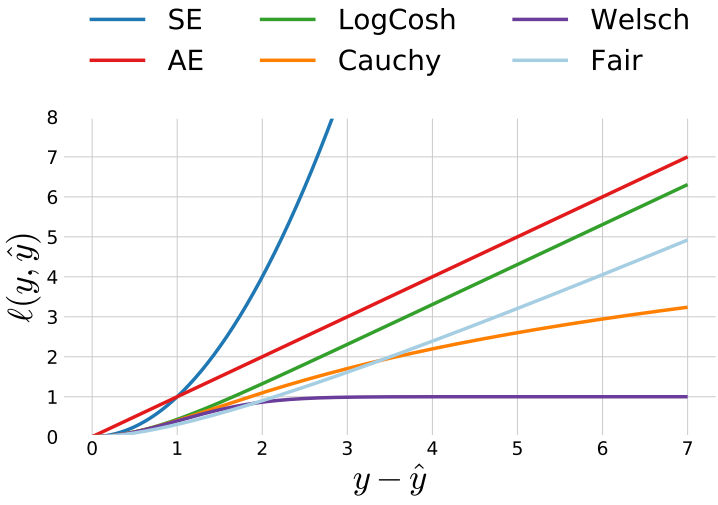


Figure 2.9: Comparison of the the six loss functions SE, AE, LogCosh, Cauchy, Welsch, and Fair as a function of $y - \hat{y}$, see equation (2.27). In the plot SE is shown in blue, AE in red, LogCosh in green, Cauchy in orange, Welsch in purple, and Fair in light blue. For the Cauchy, Welsch, and Fair functions c is set to 1. For a zoom in of the inner region where $y - \hat{y} < 2$ see Figure 2.8. All six graphs are symmetric in $y - \hat{y}$ which is why they are only shown for positive values of $y - \hat{y}$.

2.5.1 *Evaluation Function*

Since some machine learning models require an analytic expression for the derivative and second derivative, it not always possible to use a custom objective function if it is non-differentiable. The Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) is such a measure. Another example would be the width of the distribution of all $y_i - \hat{y}_i$. In this thesis this overall performance metric will be called the *evaluation function* f_{eval} compared to the differentiable proxy for this function, the objective function. To sum up: the loss function $\ell(y, \hat{y})$ measures the loss for an individual prediction and the objective function $\mathcal{L}$ is the aggregated version of the individual losses. The objective function is assumed to be a good proxy for the evaluation function f_{eval} which is the overall metric. For the loss functions defined in section 2.5 the objective function is based on the mean of the individual losses:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \Omega. \tag{2.28}$$

2.6 *Decision Trees*

Decision Trees are a simple machine learning method that works by partitioning the feature space into smaller subspaces, high-dimensional rectangles basically, and then fit each subspace with a constant for regression problems or a single label for classification problems [55]. A simple example of this can be seen in Figure 2.10 and Figure 2.11. In the first figure we see an illustration of how the signal and background distributions look in the 2D feature space. The dashed lines in the figure indicate the cuts made by the decision tree (DT), cuts which are shown on the second figure as a typical DT plot. Here

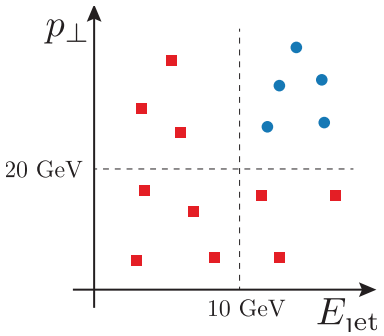


Figure 2.10: Illustration of the cuts a decision tree model make for signal in blue circles and background in red squares. This is an visualization in the feature space of the decision tree seen in Figure 2.11.

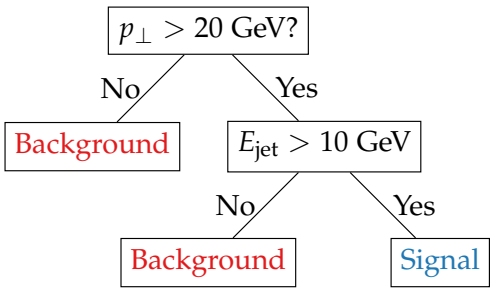


Figure 2.11: Illustration of a simple decision tree. Here the tree partitions the input feature space consisting of the two variables $p_\perp$ and E_{jet} into two categories; either signal or background. A visualization of the cuts in the feature space can be seen in Figure 2.10.

the top "box" is called the *root* of the tree, any subsequent boxes are *nodes* except the final ones that are not split any further which are *leaves*.

At first the decision tree partitions the space according to the value of the transverse momentum[22] $p_\perp$: if it is lower than (or equal to) 20 GeV then it classified as background. If the value is higher than 20 GeV then an extra split is made, this time on the energy of the jet E_{jet}: if it is higher than 10 GeV it is classified as signal and otherwise as background. Training a DT on this data allows us to predict a new

[22] All units in this project are in natural units such that both momentum and energy are in units of eV.

unseen event $(p_\perp, E_{\text{jet}}) = (24\,\text{GeV}, 11\,\text{GeV})$ to be a signal-like event. This DT is said to be a shallow tree since it only has a depth of 2. The maximum depth allowed for the model is an important hyperparameter since it clearly controls under- and overfitting by changing how many cuts and partitions in the feature space are allowed; the deeper the tree, the more complex the model becomes. Single DTs are very prone to overfitting , however, they are also extremely inspectable. They are even referred to as "white-box models" compared to black-box models such as neural networks. For a more thorough introduction to decision trees and how they are internally optimized (for finding the best cut values), see Hastie et al. [55].

2.6.1 *Ensembles of Decision Trees*

Single decision trees are prone to overfitting and generally suffer from high variance. Today especially two different methods exist to alleviate these problems: Random Forests (RFs) and Boosted Decision Trees (BDTs). Both methods are examples of so-called ensemble methods where a set of ML methods are combined into a single model. Typically ensemble methods are based on *weak learners*: simple, often fast, methods that individually show relatively poor generalization performance typically due to high variance.

Random Forests

Random Forests were first introduced in 2001 by Breiman [26]. Random Forests are a collection of B decision trees where each tree is trained on bootstrapped versions of the training data and then the individual trees' predictions T_b are averaged[23]:

[23] In the case of classification it is the majority vote which is taken.

$$f_{\text{RF}}(\mathbf{x}) = \hat{y}_{\text{RF}} = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x}). \tag{2.29}$$

The method of making artificial extra samples and training on them is in general called bootstrap aggregation or *bagging* [55]. It works by averaging out noisy estimates of the individual models hence reducing variance.

Theorem 4 (Variance of average of correlated i.d. variables)**.** *Given B identically distributed (but not necessarily independent) variables each with variance σ^2 and positive pairwise correlation ρ, the variance of the average is:*

$$\text{Var}(\bar{\mu}_B) = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2, \tag{2.30}$$

where $\bar{\mu}_B$ is the average of the i.d. variables [55].

Equation (2.30) is the main idea behind RFs. As the number of trees B in the forest increases, the first term goes towards zero. Thus, the more the correlation ρ between the individual trees is reduced, the more the variance is reduced (which is the main problem in the case of DTs to begin with).

In addition to training the trees on bagged samples, one more technique is used to further decrease the correlation between the individual trees: each bootstrapped sample of the dataset not only contains just a subset of the observations (rows), but also only a subset of the variables (columns)[24]. This is called column-subsampling (in contrast to row subsampling).

[24] Throughout this project all data is assumed to be *tidy data* unless otherwise explicitly mentioned. Tidy data was a concept formalized in 2003 by Wickham [101] which basically states that each variable forms a columns, each observation forms a row, and that each type of observation unit forms a table. This also means that the term variable and column will be used interchangeable (together with *feature*), and likewise with observation and row.

[25] Kaggle is an online platform where people compete with machine learning models.

Boosted Decision Trees

In the overall family of ensemble models, *boosting* might be the most successful of them all where especially the specific algorithm called XGBoost [36] revolutionized the ML world by winning numerous Kaggle[25] competitions [3] including the Higgs Machine Learning competition in 2014 [4].

Boosting is the process of sequentially applying weak learner models to repeatedly modified versions of data [55]. In an iterative fashion this combines many weak learners into a single strong learner. The final prediction is thus a weighted sum over the K different weak learners F_k:

$$f_{\text{BDT}}(\mathbf{x}) = F(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k F_k(\mathbf{x}) \tag{2.31}$$

Boosting thus works by reducing both bias and variance since it iteratively fits weak learners. The variance is not reduced as much as for RFs since the weak learners are more correlated, however, their bias is lower.

The term gradient boosting comes from the observation that repeatedly minimizing the residuals of the current model is similar to minimizing the gradient of the loss function for a specific choice of the loss function.

Imagine that we start off with an imperfect model $F_k(x) = \hat{y}$. In boosting we want the next iteration to be a better model than the previous one, so imagine that the perfect addition to the model that we needed to make was $h(x)$. For it to be perfect, the following would have to be true:

$$F_{k+1}(x) = F_k(x) + h(x) = y \Leftrightarrow h(x) = y - F_k(x). \tag{2.32}$$

The r.h.s. is the residual of the model, so at each stage the model is trying to fit the residuals of the current iteration of model. The "gradient" in "gradient boosting" comes from the following. Assume the loss function: $L(y, F_k(x)) = \frac{1}{2}(y - F_k(x))^2$. The gradient of the loss w.r.t. to $F_k(x)$ is:

$$\frac{\partial L(y, F_k(x))}{\partial F_k(x)} = F_k(x) - y. \tag{2.33}$$

This is exactly the negative of the r.h.s. of equation (2.32). Instead of looking at the model as trying to minimize the residual at each iteration, it can instead be generalized as trying to fit the negative gradients of the loss function:

$$h(x) = -\frac{\partial L}{\partial F_k}. \tag{2.34}$$

We thus end up with the following iterative model which is basically a *gradient descent* algorithm.

$$F_{k+1}(x) = F_k(x) + h(x) = F_k - \frac{\partial L}{\partial F_k}. \tag{2.35}$$

AdaBoost [49] was the first major algorithm to make use of boosting. It was seen as a way of iteratively giving wrongly predicted observations higher weight, however, this is just a result of a "lucky" choice of loss function[26] which was not realized until much later. AdaBoost could be used with many different weak learners, however, mostly DTs were used to form BDTs. XGBoost [36] is a fast, computationally efficient implementation of gradient boosting with DTs as base learners. It implements several model regularization techniques which makes it less prone to overfitting than other BDTs.

[26] Exponential loss for classification.

2.7 *Hyperparamater Optimization*

By now linear models with L_1 and L_2 regularization have been introduced along with decision trees (DTs), random forests (RFs) and gradient boosted trees (GBTs). All of these ML models tries to optimize their parameters according to some objective function. In addition to the parameters of the model, each one has a specific set of hyperparameters that cannot directly be optimized in the internal optimization process. This could be the amount of regularization λ for linear models, the maximum tree depth for DTs, the number of trees for RFs, or the column (or row) subsampling fraction for BTDs.

In general we say that we have the ML model $\mathcal{A}$ with K hyperparameters. Each of these hyperparameters have a domain Λ_k. The domain can be either real numbers $\Lambda_k \in \mathbb{R}$, integers $\Lambda_k \in \mathbb{Z}$, binary $\Lambda_k \in \{0, 1\}$, or categorical[27]. We define the hyperparameter configuration space as: $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_K$. Within this space we are searching for a vector of hyperparameters $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ which defines the optimal model $\mathcal{A}_{\boldsymbol{\lambda}^*}$. Here the optimal model is the defined as the model which gives the best generalization performance according to some evaluation function. The goal of finding the best hyperparameter $\boldsymbol{\lambda}^*$ is known as *hyperparameter optimization* (HPO.)

[27] Could e.g. be the choice of loss function.

The first naive approach would simply to manually[28] try out different combinations of $\boldsymbol{\lambda}$ and see performance on the validation set[29]. This is of course too cumbersome for advanced ML models, but it should be noted that it is a good place to start. In subsection 2.7.1 the HPO method called Grid Search is introduced which is further generalized and optimized in subsection 2.7.2 with Random Search. Both these methods are easily parallelizable since they do not have any inherent history in its guesses. This is in contrast to Bayesian Optimization introduced in subsection 2.7.3 which allows for "smart" guesses.

[28] Also known jokingly as "Grad Student Descent".

[29] Remember only to use the test set on the final model.

2.7.1 *Grid Search*

Grid search (GS) is a HPO method also known as full factorial design. It is called this because it tries out all possible combinations of the hyperparameter configuration space: the so-called cartesian product of $\mathbf{\Lambda}$. Imagine a 2D space where the two domains are respectively $x = \{1, 2, 3\}$ and $y = \{1, 2, 3, 4\}$. Then GS runs through all $3 \times 4 = 12$ combinations of these two sets:

$$(x_i, y_i) \in \{(1,1), (1,2), \ldots, (x_i, y_i), \ldots, (3,4)\}, \tag{2.36}$$

as visualized in Figure 2.12. The advantage of GS is that it is an exhaustive search over all combinations[30] of hyperparameters, however, the total number of combinations grows exponentially and GS as a method thus suffers the curse of dimensionality[31].

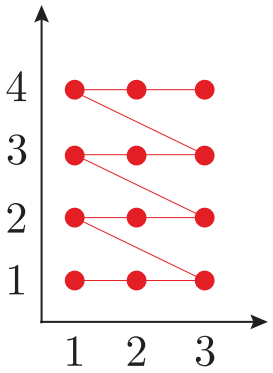


Figure 2.12: Visualization of grid search run on the two hyperparameters x and why y with the domains $x = \{1, 2, 3\}$ and $y = \{1, 2, 3, 4\}$.

[30] Note that the user has to provide the values for each hyperparamater to be tried out manually.

[31] Not the dimensionality of the input feature space, but of the hyperparameter configuration space.

2.7.2 *Random Search*

To circumvent the problems of grid search, Bergstra and Bengio [23] developed the Random Search (RS) algorithm in 2012. Regarding the effect of the curse of dimensionality on grid search they wrote: *"This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization"* [23]. Instead of searching through all possible values of $\boldsymbol{\lambda}$ like in GS, RS makes B runs where each $\boldsymbol{\lambda}_i$ is given by:

$$\boldsymbol{\lambda}_i \sim \sum_{j=1}^{K} \mathrm{PDF}_j(\Lambda_j) \cdot \hat{\boldsymbol{e}}_j. \tag{2.37}$$

Equation (2.37) should be understood in the following way. For each hyperparameter draw a random number from a user-defined Probability Density Function (PDF) and then let $\boldsymbol{\lambda}$ be the vector of those N random numbers. In a 2D-space $\boldsymbol{\lambda}_i$ could thus be:

$$\boldsymbol{\lambda}_i \sim = \begin{bmatrix} \mathcal{N}(100, 4) \\ \mathcal{U}(0, 1) \end{bmatrix}, \tag{2.38}$$

where $\mathcal{N}(100, 2)$ is normal (Gaussian) distribution with mean $\mu = 100$ and standard deviation $\sigma^2 = 4$ and $\mathcal{U}(0, 1)$ is the uniform distribution in the interval $[0, 1]$. Of course the PDF can be a PMF in the case of discrete hyperparameter domains.

The reason why random search is so powerful is not only because the number of function evaluations B is easily tunable[32], but also due to the fact that often some hyperparameter dimensions are more important than other. Even though the hyperparameter configuration space $\mathbf{\Lambda}$ might be high-dimensional, it often exhibits *low effective dimensionality* [23]. In the simplest 2D-case this can be written as the following example. Imagine that we want to maximize some evaluation function, e.g. accuracy of the predictions, and the model depends on the two independent hyperparameters x and y: $f(x, y)$. In this example assume that f is almost insensitive to y and thus has

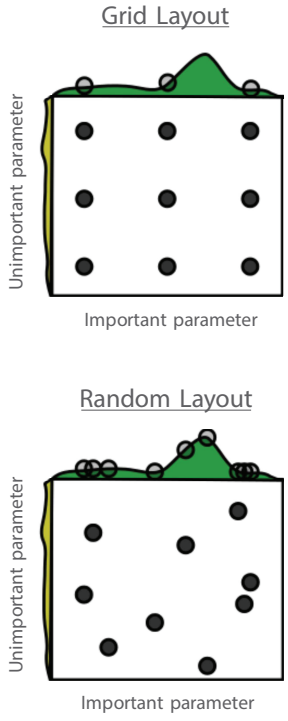


Figure 2.13: Visualization of the difference between grid search and random search. Adapted from Bergstra and Bengio [23].

[32] Compared to grid search which tries *all* possible combinations.

an effective dimensionality of 1. Then $f(x, y) = g(x) + h(y) \approx g(x)$. For a visualization of this example, see Figure 2.13.

Here GS is run with a grid of $3 \times 3 = 9$ points and RS is similarly run with 9 points drawn from uniform PDFs in the same interval. It easy to see that when the hyperparameter configuration space has a lower effective dimensionality than the actual dimensionality RS is far better at probing the space due to the projections into the sensitive dimensions cover more of these axes than for GS. In general in ML the hyperparameter configuration space has lower effective dimension than its actual dimension, but different hyperparameters matter in different datasets

In general only a fraction of all hyperparameters matter for any dataset but different hyperparameters matter in different datasets and thus generally RS performs as well as GS or better [23].

Note that RS can be seen as a generalization of GS, where GS is the specific example of RS if one uses a multidimensional binomial distribution as PDF where the PDF is reevaluated after each run.

2.7.3 *Bayesian Optimization*

When performing hyperparameter optimization it often takes a lot of time to evaluate the individual hyperparameters. Remember, that each evaluation consists of fitting $\mathcal{A}_\lambda$ to the training data and then measure the performance on the validation set. Fitting the model on the training set can often take minutes, if not hours. This process is even slower when using cross validation. The idea behind Bayesian Optimization is that when the ML model, or any other black box function, is expensive[33] to evaluate then "smart" guesses are worth spending a bit of time on developing. The hope is that the time taken to come up with smart guesses is negligible compared to the overall function evaluation time. This is contrast to both GS and RS where each new set of hyperparameters $\boldsymbol{\lambda}$ is independent of the value of the evaluation performance.

[33] With respect to time.

In Bayesian Optimization (BO) [28], the evaluation function as a function of hyperparameter is unknown. This unknown function is iteratively fitted with a probabilistic surrogate model, most often by Gaussian processes (GPs). Given the fitted surrogate model, an acquisition function is computed. This is a manually chosen function which is cheap to evaluate and is a measure of where in the hyper-dimensional hyperparameter space there is a highest chance of finding a new good value of $\boldsymbol{\lambda}$. The acquisition function has to be chosen manually and especially the tradeoff between *exploitation* versus *exploration* is particularly important. This value decides how "adventurous" or conservative the BO algorithm should be when exploring the evaluation space.

Bayesian Optimization is better explained by looking at Figure 2.14. First look at the top plot. This is a plot of the surrogate function in black with uncertainties shown in blue. This is a result of fitting GPs to the two previous points in black, $t = 2$. This surrogate function is

supposed to fit the unknown hyperparameter-dependent evaluation function (called objective in the figure) shown as a dashed black line. Below we see the acquisition function in green. This is a function of the blue curve and the position of its maximum decides where the next guess of λ should be. With the chosen acquisition function and exploration willingness, we see that the next guess should be slightly to the left of the right-most point. This is a simple 1D toy problem, but one should imagine this happening in a high-dimensional space. After making a new guess, $t = 3$ in the middle plot, the acquisition function changes since it learnt that this gave a worse evaluation value than the right-most point. Therefore, the next proposal for λ is slightly to the right of the right-most point. The process continues like this in an iterative fashion: first fitting GPs to the previous evaluation values and then choosing the next λ according the acquisition function given the GPs.

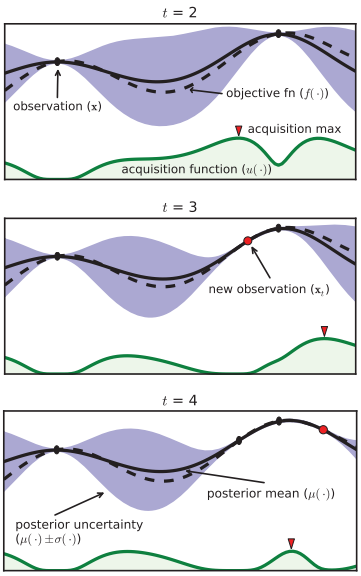


Figure 2.14: Illustration of the learning process of Bayesian optimization. The previous observations are shown as black dots and the true objective function is shown as a dashed black line. This line is fitted with Gaussian processes (GPs) which is shown as the solid line with its uncertainty in purple. The acquisition function is shown in green and its maximum decides what the next iteration of the hyperparameter value(s) should be. Adapted from Brochu et al. [28].

Gaussian Processes provide a posterior distribution given some prior distribution and the data-dependent likelihood. The process of BO is quite technical and mathematical, especially if GPs are new material. For a more in-depth explanation of the topic, see Brochu et al. [28]. The important thing to note is that GPs return not only a posterior mean $\mu(\mathbf{x})$ but also an uncertainty $\sigma(\mathbf{x})$, as seen in Figure 2.14 described above. The acquisition function used in this project is the Upper Confidence Bound (UCB):

$$\text{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \tag{2.39}$$

where $\kappa \geq 0$ is the parameter[34] controlling the exploration-exploitation tradeoff.

Bayesian Optimization has the great benefit of slowly learning the hyperparameter space and making smarter and more educated guesses over time. However, it also comes with the cost of being harder to numerically implement compared to GS and RS[35], and parallelization is non-trivial to implement since it by definition is a sequential process. The performance boost is also not guaranteed.

[34] Here κ can be regarded as a hyper-hyperparameter since it controlls how the other hyperparamaters are optimized.

[35] Which are basically plug-and-play with Scikit-Learn [74].

2.8 *Feature Importance*

Having first established in section 2.2 that machine learning algorithms are indeed able to not only learn from data but also to generalize well without overfitting (section 2.4), modern ML algorithms such as decision trees, random forests and boosted decision trees were introduced in section 2.6 and they were hyperparameter optimized in section 2.7, one would expect that one would have a well performing model by now.

Now comes one of the most important issues in ML today: model inspection. Actually trying to make sense of the learnt model. Why does it predict as it does? Which features or variables are most important according to the model? Model interpretation is still very much active research today with no universally accepted methods. Some methods are model dependent and accurate, others might be

model agnostic but slow or only approximations. In this project the focus will be on the so-called *SHAP* values.

In 2017 Lundberg and Lee [66] showed that six different previously used methods were all specific instances of a universal underlying method[36] and proposed SHaplay Additive exPlanation (SHAP) values as a unified measure of feature importance. In 2018 they developed a fast algorithm for computing SHAP values for tree ensemples and showed that previous measures of feature importance heavily used for trees, e.g. *gain*, were *inconsistent* [68]. That a measure for feature importance is inconsistent means that a model could rely more on feature A than B, however, the feature importance would indicate opposite. SHAP values and *permutation*-based feature importances are both consistent feature importance measure, however, only SHAP allows for individualized[37], or local, feature importances.

[36] The class of *additive feature attribution methods* [66].

[37] Meaning that you can get the feature importances for a single observation compared to only the global, overall feature importances as seen across the entire data set.

SHAP values are within the class of additive feature attribution methods, which are functions where the explanation model g is a linear combination of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i. \tag{2.40}$$

Here $\phi_i \in \mathbb{R}$ are the feature importances and z' is a binary variable such that $z'_i = 1$ if the feature is present and otherwise $z'_i = 0$. SHAP values are based on Shapley regression values known from cooperative game theory [84]. These values are based on the three axioms:

Axiom 1 (Local Accuracy)**.** *Local accuracy says that the sum of the feature importances should equal the total reward:*

$$f(x) = g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i. \tag{2.41}$$

Here f is the ML model, g is the explanation model, x is an observation in input feature space, z' is an observation in the binary space as described above, and ϕ_i is the feature importance.

Axiom 2 (Missingness)**.** *Missingness means that features missing in the original input feature space (such that $z'_i = 0$) should be attributed no importance:*

$$z'_i = 0 \Rightarrow \phi_i = 0. \tag{2.42}$$

Axiom 3 (Consistency)**.** *Consistency states if a model is changed such that it relies more on a certain feature, the feature importance of that feature should never decrease.*

Given these three axioms, Lundberg and Lee [66] show that the only solution to equation (2.40) is:

$$\phi_i = \sum_{S \subseteq \widetilde{M} \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} \left[f_x(S \cup \{i\}) - f_x(S) \right], \tag{2.43}$$

which one can simplify to:

$$\begin{aligned}\phi_i &= \sum_{S\subseteq\widetilde{M}\backslash\{i\}} w(S)\cdot\Delta_{f_x}(S)\\ w(S) &\equiv \frac{|S|!(M-|S|-1)!}{M!}\\ \Delta_{f_x}(S) &\equiv \left[f_x(S\cup\{i\})-f_x(S)\right].\end{aligned} \tag{2.44}$$

In equation (2.43) $\widetilde{M}$ is the *set* of all input features[38], $S\subseteq\widetilde{M}\backslash\{i\}$ means a subset S of $\widetilde{M}$ without feature i, $S\cup\{i\}$ means the set S with feature i and $f_x(S)=f(h_x(z'))$ where $h_x(z')$ is the mapping function from the binary z' space to the input feature space x. In equation (2.44) the function is simplified to its basic constituents: the difference in performance between including feature i and not including it, Δ_{f_x}, and its weight w.

[38] Compared to $M=|\widetilde{M}|$ which is the *number* of all input features.

To get a better understanding of the different sets in the summation, one could look at the decision tree shown in Figure 2.11. Here $\widetilde{M}$ would be $\widetilde{M}=\{p_\perp, E_{\text{jet}}\}$. For the feature $i=p_\perp$ one would thus have:

$$\begin{aligned}\phi_{p_\perp} &= \sum_{S\subseteq\widetilde{M}\backslash\{p_\perp\}} w(S)\cdot\Delta_{f_x}(S)\\ &= \sum_{S\in[\{\},\{E_{\text{jet}}\}]} w(S)\cdot\Delta_{f_x}(S)\\ &= \frac{0!(2-0-1)!}{2!}\cdot\left[f_x(\{p_\perp\})-f_x(\{\})\right]\\ &\quad+\frac{1!(2-1-1)!}{2!}\cdot\left[f_x(\{E_{\text{jet}},p_\perp\})-f_x(\{E_{\text{jet}}\})\right].\end{aligned} \tag{2.45}$$

Whereas $w(S)$ are easily calculated, Δ_{f_x} depends on the data. As the number of features grows, the number of terms in the sum grows exponentially. What Lundberg et al. [68] did was to develop an efficient algorithm that could solve this for trees in polynomial time[39].

[39] Specifically they managed to improve the time complexity from $\mathcal{O}(TL2^M)$ to $\mathcal{O}(TLD^2)$ where T is the number of trees, L is the maximum number of leaves in any tree, M is the number of features, and D is the maximum depth of any tree (where $D\approx\log L$ for balanced trees).

SHAP values allows one to explain for a single prediction why it got the prediction that it got. When applied to the entire data set $\mathbf{X}\in\mathbb{R}^{N\times M}$ with N observations each with M features, one gets the matrix $\mathbf{\Phi}$. When summing over the absolute value of each column, one gets the global impact ϕ_i^{tot} of the i^{th} feature [68]:

$$\phi_i^{\text{tot}}=\sum_{j=1}^{N}|\mathbf{\Phi}_{i,j}|. \tag{2.46}$$

The global feature importance ϕ_i^{tot} is thus a measure of the overall importance of feature i.

Note that if one introduces a new feature to the dataset correlated[40] to an already existing feature, the feature importance of the previous feature will decrease. This is due to the axiom of symmetry:

[40] Not necessarily linearly correlated.

Axiom 4 (Symmetry)**.** *If for all subsets S that do not contain i or j:*

$$f_x(S \cup \{i\}) = f_x(S \cup \{j\}) \tag{2.47}$$

then $\phi_i = \phi_j$.

It can be shown that axiom 4 is implied by axiom 3 [66, Supp. Material]. Two identical features, as seen by the model, will thus "share" the feature importance.

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].

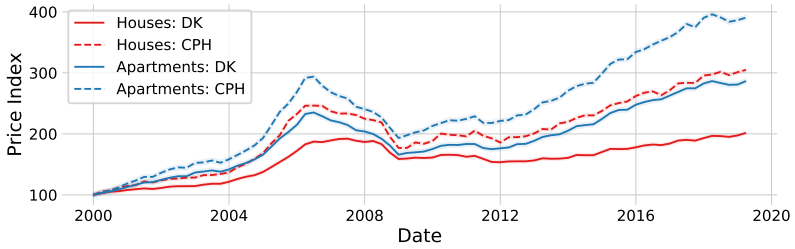


Figure 3.1: Price Index of the Danish housing market. Prince index of Danish one-family houses and owner-occupied apartments where houses are shown in red and apartments in blue, where full lines are for the entirety of Denmark and dashed lines are only for Copenhagen. Errorbars (scaled up with a factor 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).

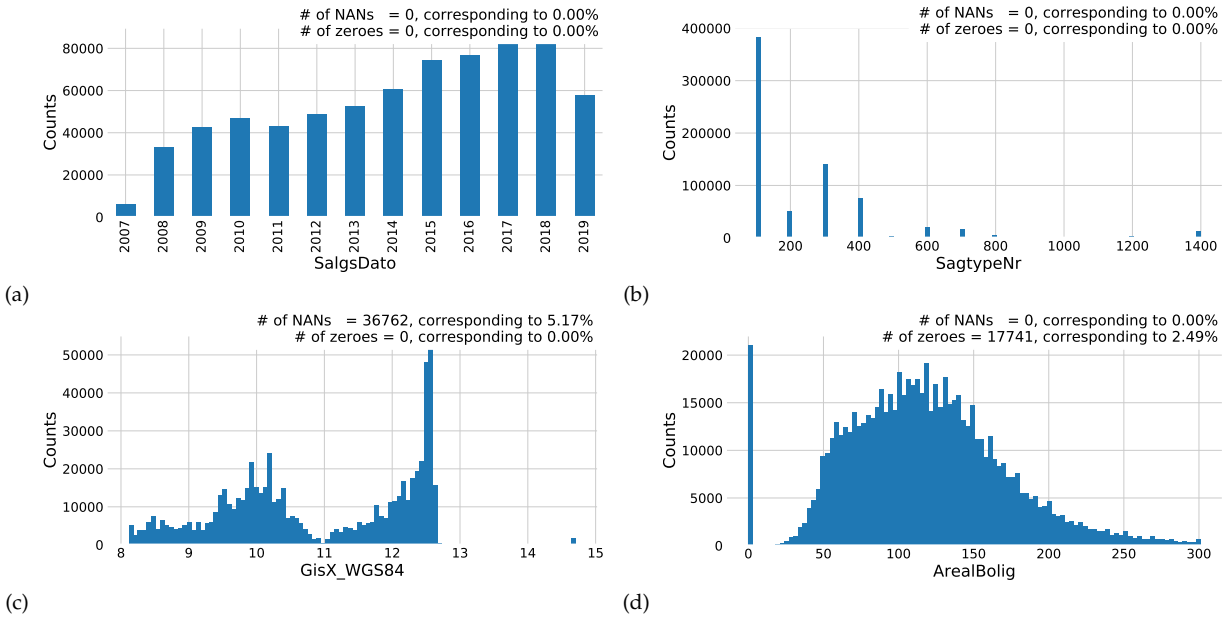


Figure 3.2: Distributions of four out of the 168 input variables. Subplot (a) shows the date of the sale, Subplot (b) shows the type of residence, Subplot (c) shows the longitude, Subplot (d) shows the area og the house.

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.

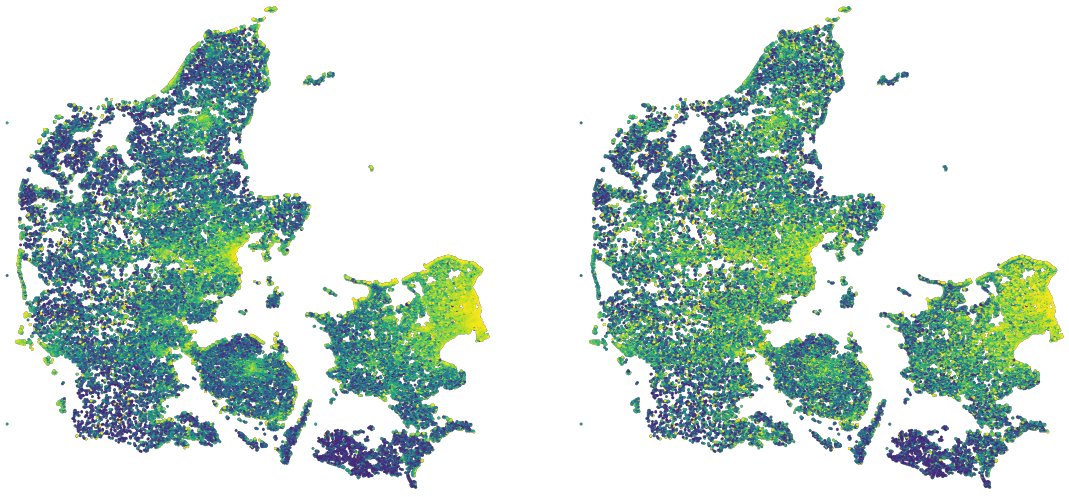
In this project only one-family houses – "Villas" in Danish – with code 100 and owner-occupied apartments – "Ejerlejlighed" in Danish – with code 300 are considered. As can be seen from Figure 3.2 (b) these two types of residences are also the most frequent sales with close to 400 000 and 150 000 sales 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.

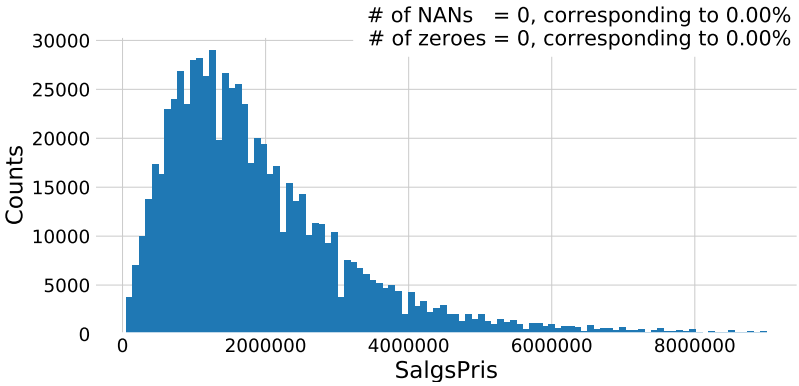


Figure 3.4: Histogram of prices of houses and apartments sold in Denmark.

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

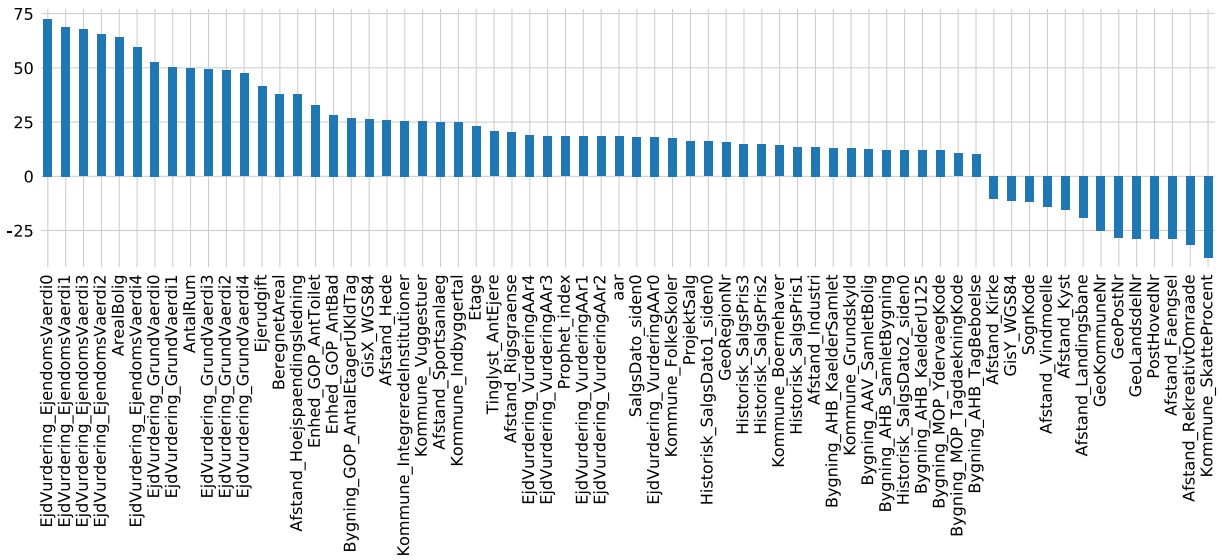


Figure 3.5: Linear correlation between variables and price for variables where the correlation coefficient ρ is $|\rho| > 10\,\%$.

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.

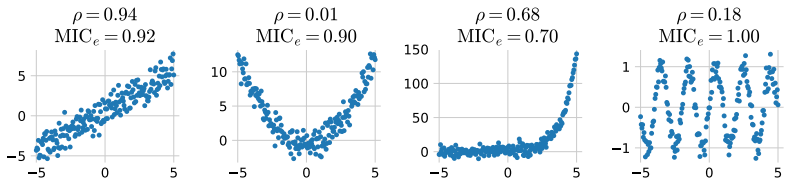


Figure 3.6: Comparison of the linear correlation ρ and the non-linear MIC for a straight line, a parabola, an exponential, and a sine wave, all with noise added. See also Figure A.17.

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

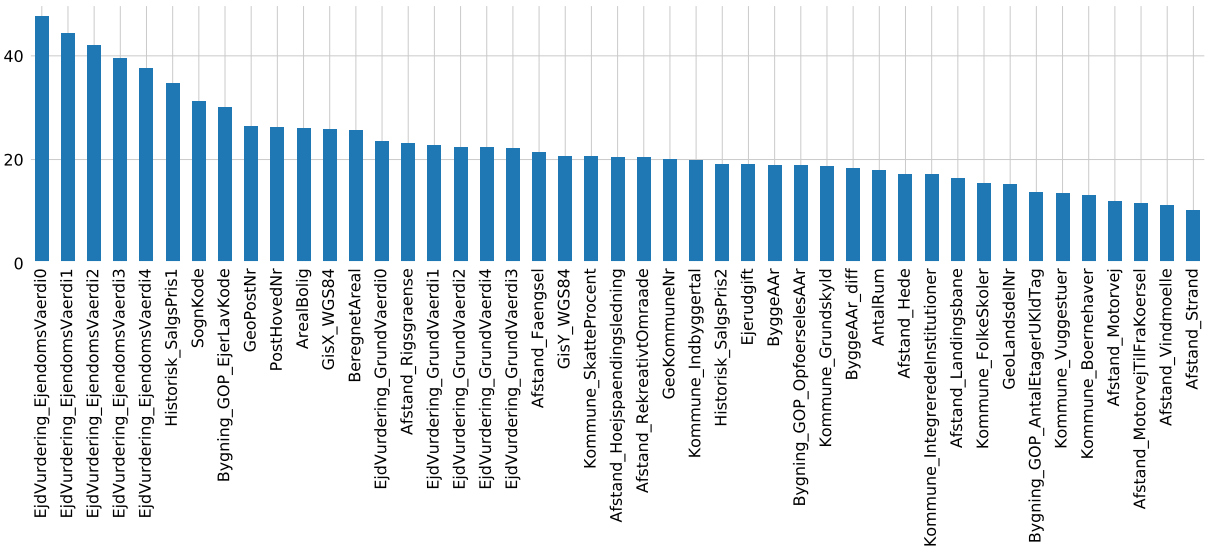


Figure 3.7: Non-linear correlation between variables and price using Maximal Information Coefficient (MIC) for variables where MIC $> 10\%$.

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.

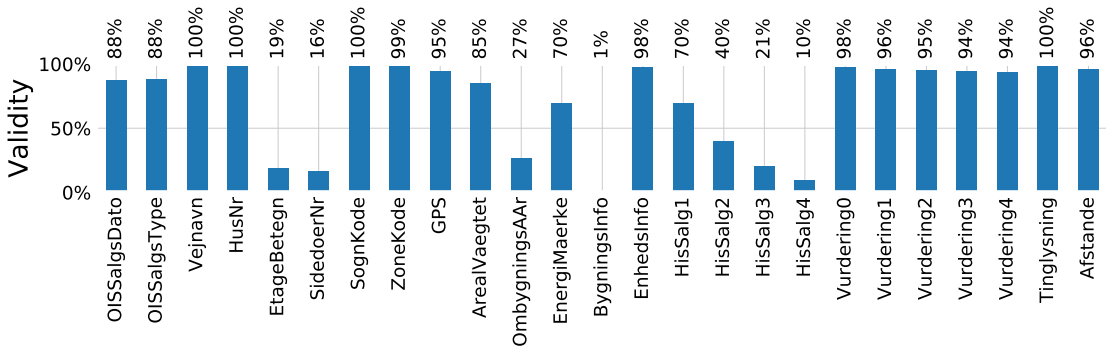


Figure 3.8: Percentage of valid counts for each variable grouped together in categories.

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.

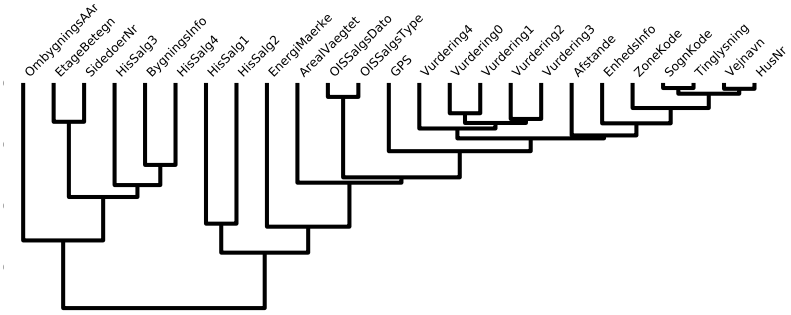


Figure 3.9: Validity Dendrogram based on hierarchical clustering of the linear correlation of validity for the housing price variables clustered together.

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.

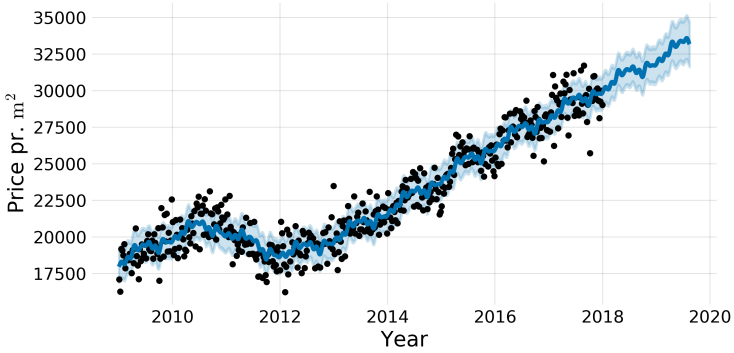


Figure 3.10: The predictions of the Facebook Prophet model trained on square meter prices for owner-occupied apartments sold before January 1st, 2018. The data is down-sampled to weekly bins where the median of each week is used as in input to the Prophet model. This can be seen as black dots in the 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

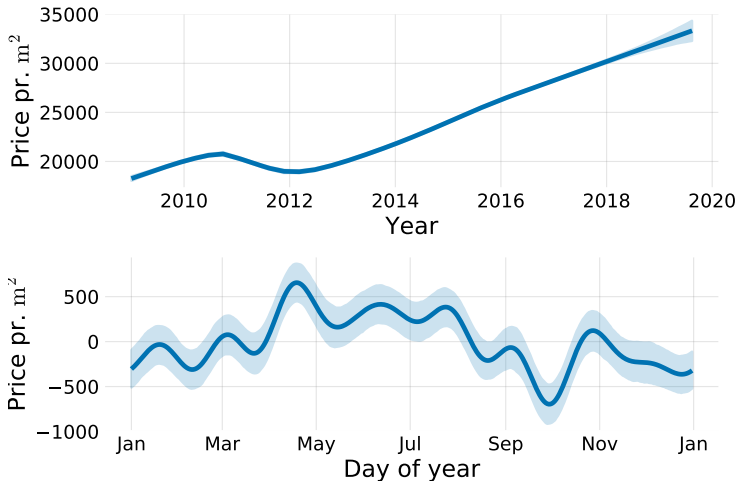


Figure 3.11: The trends of the Facebook Prophet model trained on square meter prices for owner-occupied apartments sold before January 1st, 2018. In the top plot is the overall trend as a function of year and in the bottom plot is the yearly variation as a function of day of year. It can be seen that 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.

	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".

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:

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

where $T_{\frac{1}{2}}$ is the half-life.

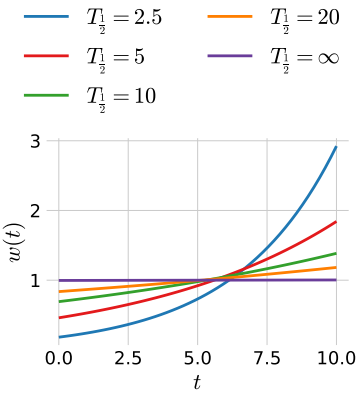


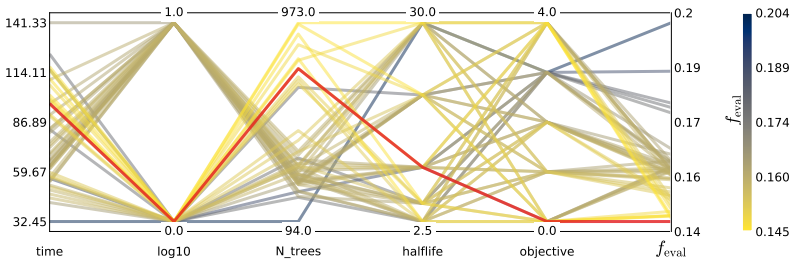
Figure 3.12: The sample weight $w(t)$ as a function of time T where the time is in years after January 1st, 2009, and the

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[17]. 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.

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) 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[18] generally performed better than the ones with fewer trees,

[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}.

[17] 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.

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.

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

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

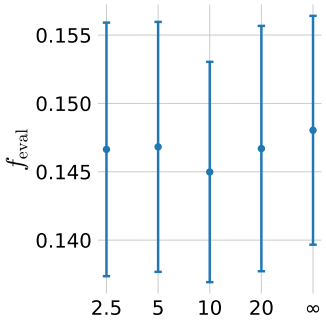


Figure 3.14: Evaluation score as a function of the weight half-life $T_{\frac{1}{2}}$ with the standard deviation over the 5 folds as errorbars for apartments.

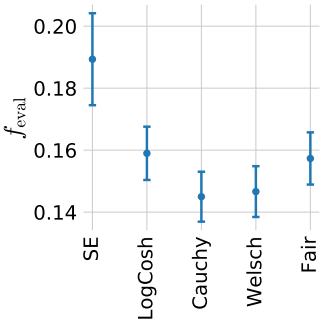


Figure 3.15: Evaluation score as a function of the loss function with the standard deviation over the 5 folds as errorbars for apartments.

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[19]. 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-downsampling 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.

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

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

[19] 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.

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 (in the appendix). The corresponding plots for houses can be seen in Figure A.23 and A.24.

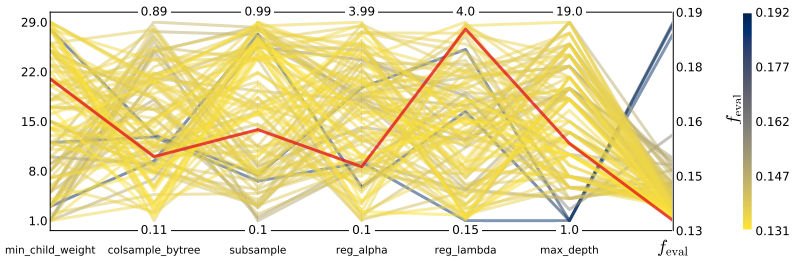


Figure 3.16: Hyperparameter optimization results of XGBoost parameters of the housing model for apartments shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

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 flat of a minimum this is: 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.

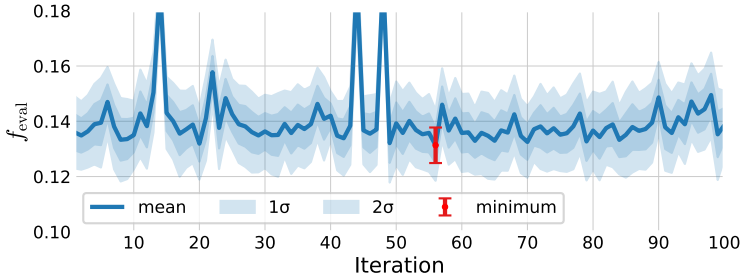


Figure 3.17: The results of running random search (RS) as hyperparameter optimization (HPO) on apartments using the XGB-model. The minimum (mean) loss along with its uncertainty is shown in red, the means for the different iterations of RS in blue, and as light blue bands are the one (and two) standard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

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 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.

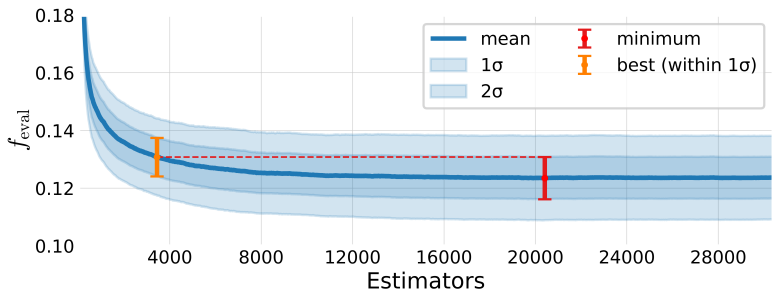


Figure 3.18: The results of early stopping on apartments using the XGB-model. The minimum (mean) loss along with its uncertainty is shown in red, the means for the different iterations of RS in blue, and as light blue bands are the one (and two) standard deviation(s), all as a function of number of estimators (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[20] 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 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 "easy"

[20] Written as `MAD`.

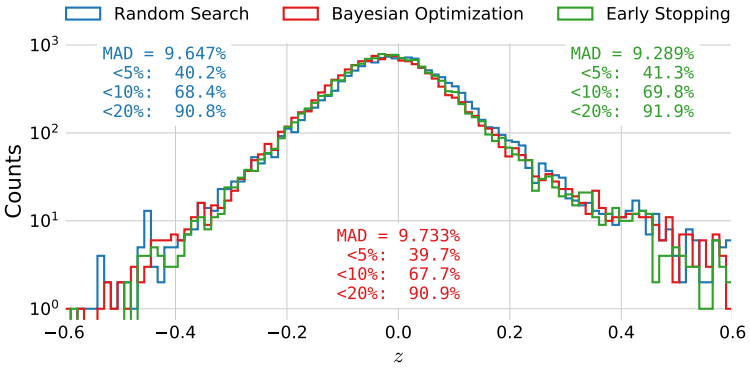


Figure 3.19: Histogram of z-values of the XGB-model trained on apartments. The performance after hyperparameter optimization using random search is shown in blue, for Bayesian optimization in red, and the learning rate optimized with early stopping in green.

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.

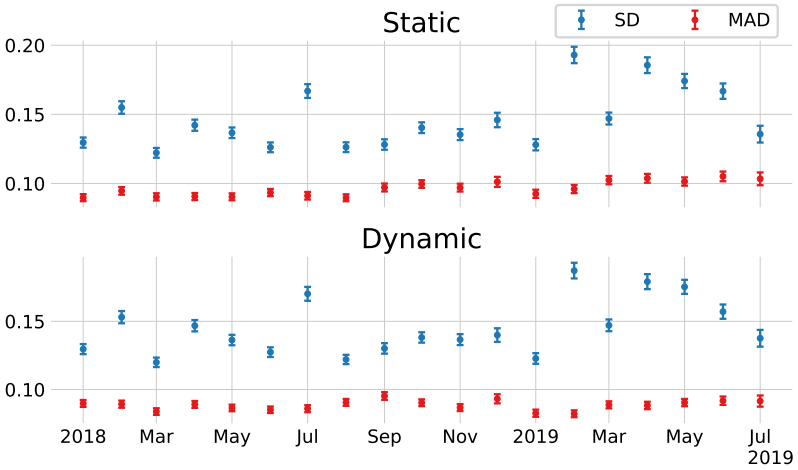


Figure 3.20: Performance of 1-month forecasts for apartments sold in 2018 and 2019. For both plots the XGB model is trained on data up to (but excluding) 2018. Top) The performance of the static model's prediction for both the standard deviation (SD) and MAD of the z-scores. Bottom) Same as above, however, 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[21] [18]. Notice the large fluctuations in the standard deviations over time compared to MADs which

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

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.

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 for the static model is shown where it is visible for $\text{MI}_{\text{median}}$ how it consistently overshoots. Compare this to the dynamic $\text{MI}_{\text{median}}$ which fluctuates around 0. BLABLA mere analyse her XXX. That the MI_{mean} is consistently lower than $\text{MI}_{\text{median}}$ is an indication of a low tail in z_{mi} (*negative skewness*) which means that some of the the predictions are much lower than the actual sales price. BLABLA, XXX. **TODO!**

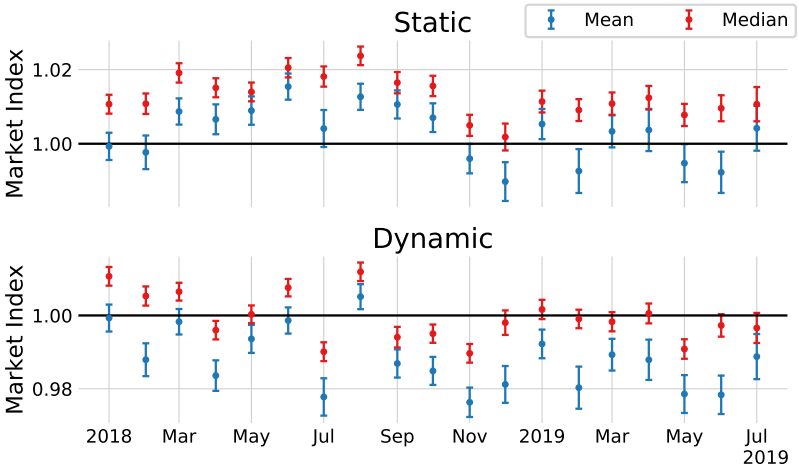


Figure 3.21: The Market Index as defined in equation (3.9) for the static and dynamic 1-month forecasts for 2018 and 2019. Plotted with using the mean in red and the median in blue.

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[22] 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 as

[22] Global here meaning for the entire dataset.

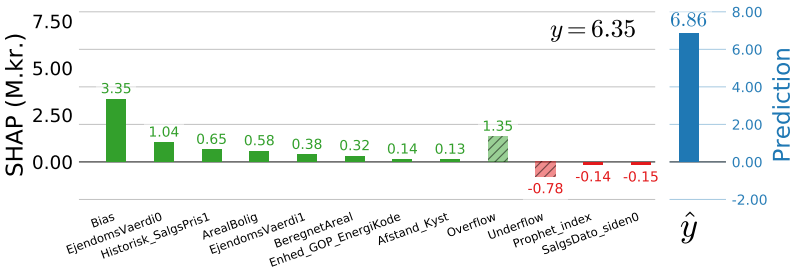


Figure 3.22: Model explanation for XGB model for a specific apartment. The bars are the variables in the dataset that the model found most important sorted after their importance for this particular apartment. The bias bar refers to the expected value of the model, which is simply the mean of the training 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.

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 (DOM) (`LiggetidSamlet`) is large it pushes the prediction in the negative direction.

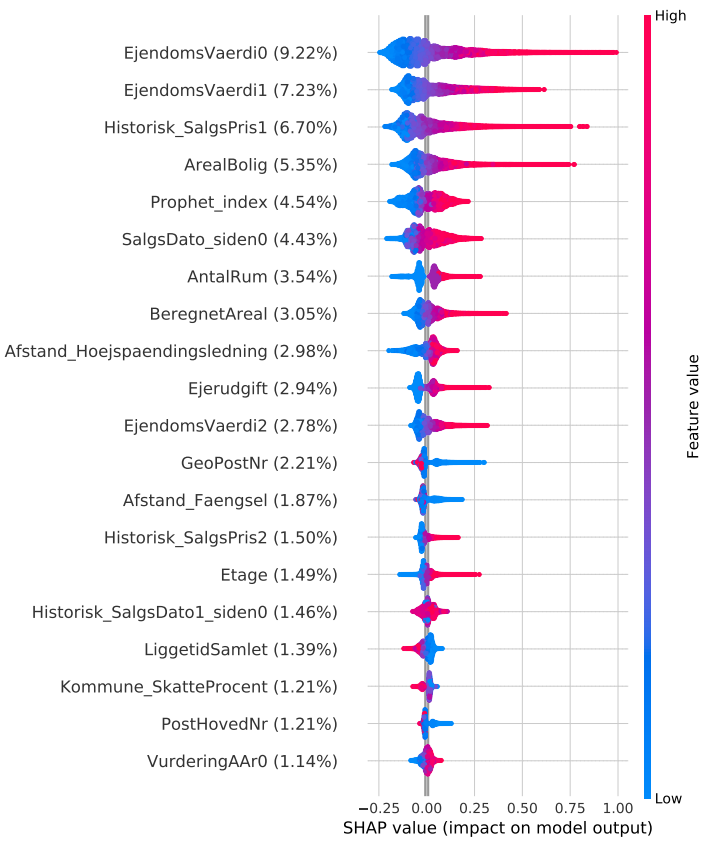


Figure 3.23: Feature importance of apartment prices using the XGB-model. The feature importance is measured using SHAP values. The variables are sorted top to bottom according to their overall feature importance, i.e. the previous public property valuation `EjendomdsVaerdi0` is the most important single feature. Along the x-axis is the impact on model output, in this example the price in XXX. 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 (DOM) 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 impor-

tant 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. 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.

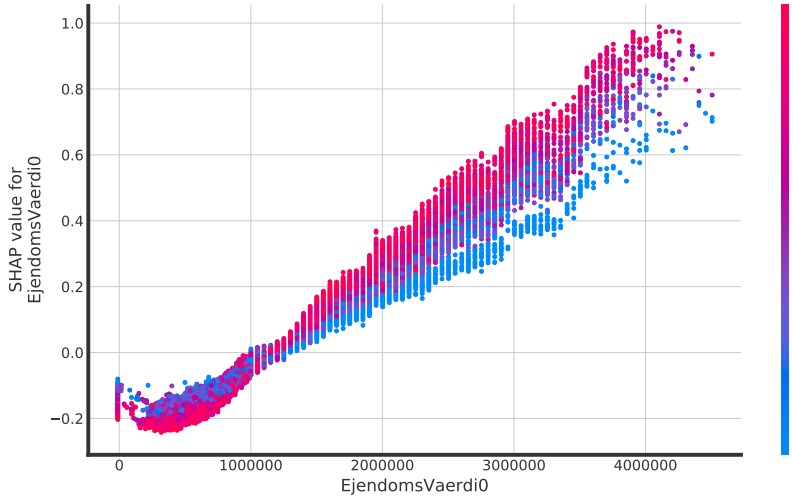


Figure 3.24: Feature importance of apartment prices using the XGB-model. XXX

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 where all of the NaNs were median-imputed[23] and the input features were scaled[24] with a robust scaler from Scipy [98] in the $(25, 75)$ quantiles range and

[23] Which means that all invalid values were replaced with the median along each column.

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

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[25] (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.

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

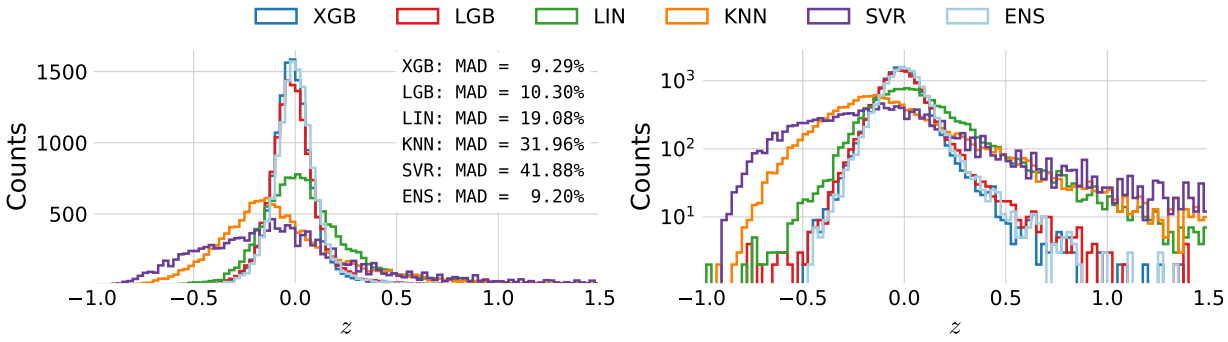


Figure 3.25: Distribution of the relative predictions $\mathbf{z}$ for the five different models and the ensemble model. The left plot shows the normal histogram (with a linear scale), whereas the right one has a log-scaled y-axis to better see the tails of the distributions. The MAD scores for the different 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[26] 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[27] 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

[26] See subsection 2.4.2.

[27] 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[28]. With the dataset at hand, decent results were made, however, they were nowhere near the performance of the realtors' predictions. 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.

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

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. They 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 `Ejendomsvaerdi0` (PPPV), `GeoPostNr` (postal code), `ByggeAAr` (year of construction), `Afstand_Kyst` (distance to shore), and `BeregnetAreal` (weighted area) and the text descriptions (encoded with TF-IDF). 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.

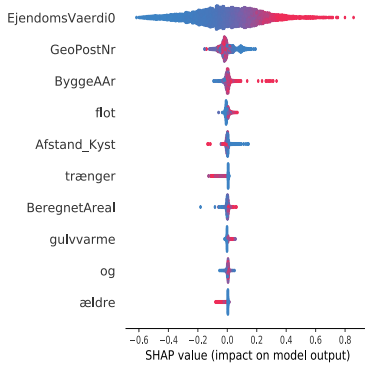


Figure 3.26: SHAP plot villa TFIDF XXX.

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,

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[29] 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 [30] 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.

[29] Basically advanced neural networks with many layers.

[30] 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.

4. Particle Physics and LEP

> *"Not only is the Universe stranger than we think, it is stranger than we can think."*
>
> — Werner Heisenberg

The aim of this chapter is to introduce the reader to the level of particle physics required for understanding the following chapter, in particular introducing the Standard Model in section 4.1, the theory behind quark hadronization in section 4.2, and the ALEPH detector at LEP in section 4.3. The goal is not to make a deep and thorough introduction to the field as this is not needed for the following analysis along with the fact that the author is no particle physicist himself.

4.1 The Standard Model

The *Standard Model* (SM) [50, 82, 100] of particle physics is the currently best known description of the elementary particles and thus describes the fundamental building blocks of our Universe. An overview of the particles explained by the Standard Model is shown in the typical tabular form seen in Figure 4.1. In general, particles comes in two categories: *bosons* and *fermions*.

The fermions, the left part of the figure, are particles with half-integer spin that obey Fermi-Dirac statistics and are further subdivided into *quarks* (upper left in figure) and *leptons* (lower left). The quarks interact with all of the four known forces[1], including the strong force. In contrary the leptons do not interact with the strong force. Quarks are never observed freely but are always combined into *hadrons* due to *color confinement* which is further explained in section 4.2. An example of this are protons which consists of two up-quarks and a down-quark. Leptons exist as either the charged leptons[2] or as neutral leptons, the so-called neutrinos[3]. The fermions come in three generations with increasing mass.

[1] Gravity, electromagnetism, and the strong and weak force.

[2] The electron e, the muon μ, and the tau τ.

[3] The electron neutrine ν_e, the muon neutrino ν_μ, and the tau neutrino ν_τ.

The bosons, the right part of the figure, are the force-carrying particles (with integer spin and which obey Bose-Einstein statistics) where the gluon g mediates the strong nuclear force (color charge), the photon γ mediates the electromagnetic force (charge), and the two $W^\pm$ and the Z bosons the weak nuclear force (weak isospin). The Higgs boson H, experimentally discovered in 2012 [39, 40], does not mediate any forces but interacts with all massive particles and explains why particles have mass.

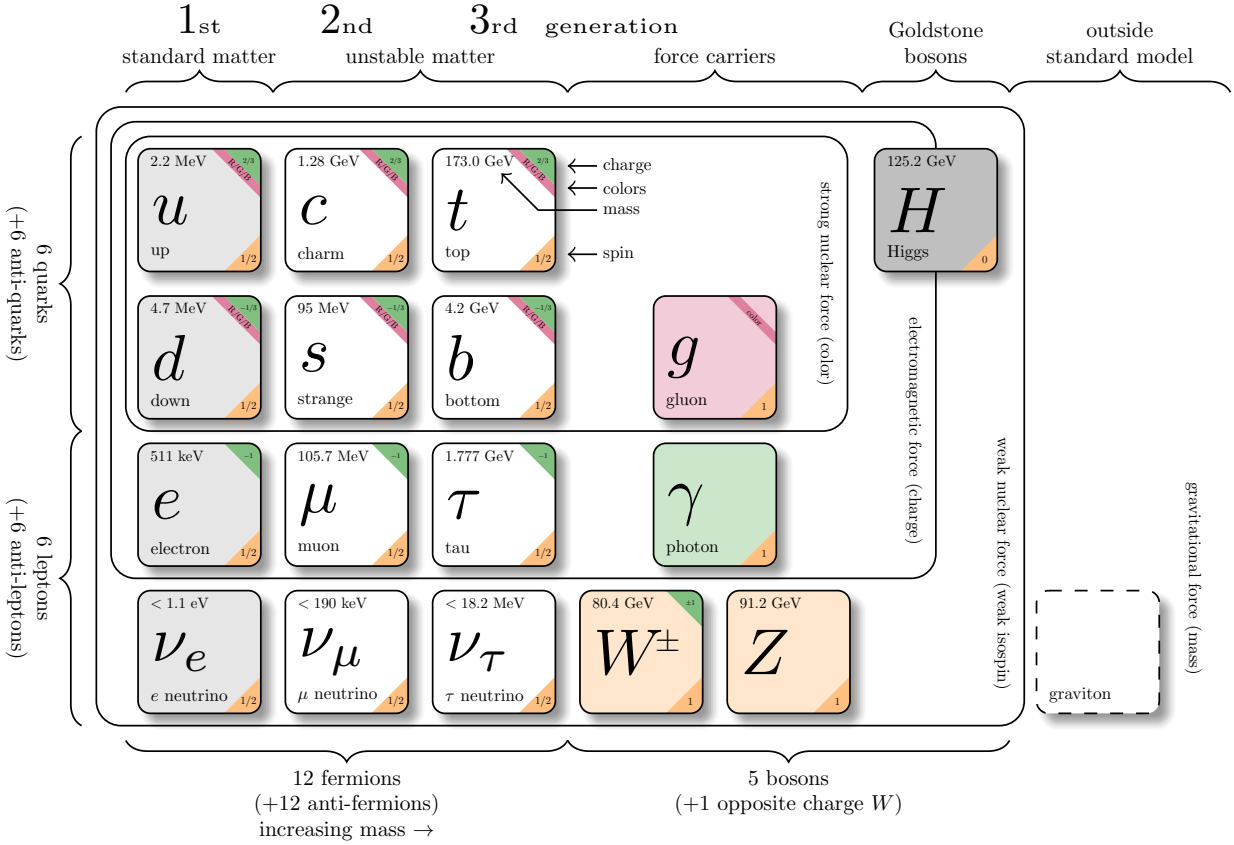


Figure 4.1: The Standard Model. Inspired by Purcell [77] using the template by Burgard [31] with manually updated masses according to Particle Data Group et al. [73].

All particles have antiparticles which are particles with opposite charge but the same mass. Some particles are their own antiparticles[4], such as the Z. At the Large Electron Positron collider (LEP), see section 4.3, electrons e^- and their antiparticles positrons e^+ were collided at an energy of around 91 GeV. This particular energy was chosen since this is at the resonance peak of the Z. Its mass distribution follows a Cauchy distribution (also known as Breit-Wigner) with mean[5] $m_Z = (91.1876 \pm 0.0021)\,\text{GeV}$ and a full width of $\Gamma_Z = (2.4952 \pm 0.0023)\,\text{GeV}$: LEP was as such a Z-factory. The Z, however, is only very short-lived with a half-life of $1/\Gamma_Z \sim 2.6 \times 10^{-25}\,\text{s}$. The decay mode for this unstable Z particle is primarily to hadrons ($(69.91 \pm 0.06)\,\%$) where the ratio (R) for b-quarks is $R_b = (Z \to b\bar{b}) = (15.12 \pm 0.05)\,\%$ and $R_g = (Z \to ggg) < (1.10 \pm 0.05)\,\%$ for gluons [73]. The fact that the Z is neutral and its own anti-particle means that it generally decays to a particle–anti-particle pair (due to charge-conservation). Antiparticles are written with a bar on top, e.g. the $\bar{b}$-quark is the antiparticle of the b-quark.

[4] The photon, the Z, and the Higgs.

[5] Calculated in natural units where $c = \hbar = 1$ which will also be used throughout this thesis.

4.2 *Quark Hadronization*

The electron-positron e^+e^- annihilations at LEP are complicated events that require advanced high-energy particle physics theory to be properly understood. Most of the aspects of the process is well-described by now, however, especially the hadronization process is still an area

of active research. To better get an overview of the different stages of the e^+e^- annihilations, see the Feynman diagram in Figure 4.2.

Reading from left to right, the electron and the positron annihilates to a Z. This interaction is well-described by quantum electrodynamics (QED), a theory that has been around for more than 60 years by now. As mentioned in the previous section, the Z has several decays modes, yet most of these are background processes of no interest in this project and the focus for now will be the decay mode $Z \to q\bar{q}$ (Z to quark–anti-quark) as seen in the Feynman diagram. The particles produced by the Z-decay are called primary *partons*. Since this process involves quarks, and thus color charge, QED is no longer an adequate theory: quantum chromodynamics (QCD) is needed [16]. The $q\bar{q}$ pair in this example acts as (color) dipoles from which a gluon can radiate. It can be shown with QCD that the gluon can only be radiated inside the cone that the $q\bar{q}$ pair spans [24]. As mentioned in the introduction, quarks cannot exist freely (due to *confinement*) and we therefore cannot observe the individual partons in a $q\bar{q}g$ event produced in the Feynman diagram. Confinement is basically the QCD principle saying that quarks are always confined or bound inside hadrons. The initial partons (carrying color charge) are converted to (color-neutral) hadrons by non-perturbative QCD processes in what is called *hadronization*, and these hadrons can be measured.

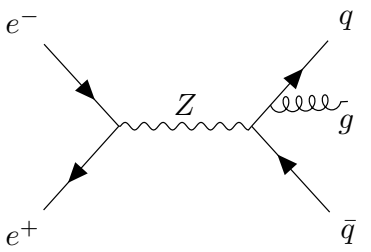


Figure 4.2: Feynman diagram showing the $e^+e^- \to Z^0$ production at LEP. The Z has several decay modes where the $Z \to q\bar{q}g$ is shown here.

The hadronization process is not yet fully modelled and currently two competing models for predicting the hadronization pattern exists: the Lund string model and the cluster model. In this project only the former of the models will be used. The Lund string model [15] is the theoretical framework underlying the widely used Monte Carlo event generator PYTHIA [85]. The string model is based on the observation that (color) field lines between quarks seem to compress into a tube-like region mediated by gluons, see the top part of Figure 4.3. The field can be described by a linearly rising potential $V(r) = \kappa r$ at large distances[6], where r is the distance and κ is the strength of the potential [30]. This field is similar to the (constant) force of a string: $V(r) = \kappa r \Rightarrow F(r) = -\kappa$ where κ is the to be regarded as the spring tension. As quarks move apart, the potential energy stored in the "string" increases until it is large enough to "snap" and convert its potential energy into mass. This mass energy is released with the production of a new $q\bar{q}$ pair as this energetically favorable, see the rest of Figure 4.3.

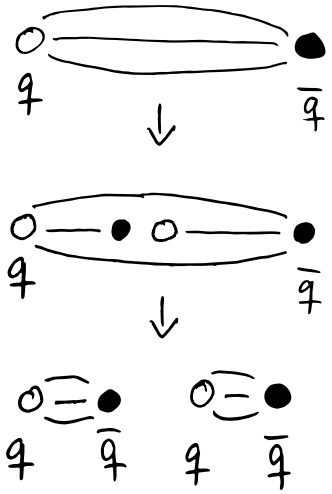


Figure 4.3: Illustration of the quarks splitting as explained by the Lund string model. For large charge separation the (color) field lines seem to be compressed to a tube-like region, where the strong interactions are mediated by the massless gluons (that couple to the color charge of quarks). When the two quarks are separated enough, the potential energy is released by the production of a new $q\bar{q}$ pair.

[6] At small distances a Coulumb term has to be included, however, this term is assumed to be negligible by the Lund string model.

An example of the hadronization process, or the transition from initial partons to final hadrons is sketched in Figure 4.4. Here the production of two kaons K^- and K^+, and two pions π^- and π^0 are shown. Since particles are created by "splits" in the "string", and the fact that there is energy-momentum conservation, they all have to share the total energy stored in the string. This is described by the fragmentation function:

$$f(z) \propto \frac{(1-z)^a}{z} \exp\left(-\frac{bm^2}{z}\right), \tag{4.1}$$

where $0 \leq z \leq 1$ is the remaining momentum that the new hadron takes, a and b are constants, and m is the mass[7] [24]. When the system runs out of available momentum, it will stop producing new hadrons and the fragmentation function thus explains the distribution of final state particles. The Lund string model can be extended from only $q\bar{q}$ events to $q\bar{q}g$ events where it predicts cones spanning the angular regions qg and $\bar{q}g$ should receive enhanced particle production compared to the $q\bar{q}$ region. This prediction by the Lund string model is also measured in e^+e^- collisions [30].

[7] Where $m \to m_\perp$ for particles with transverse momentum.

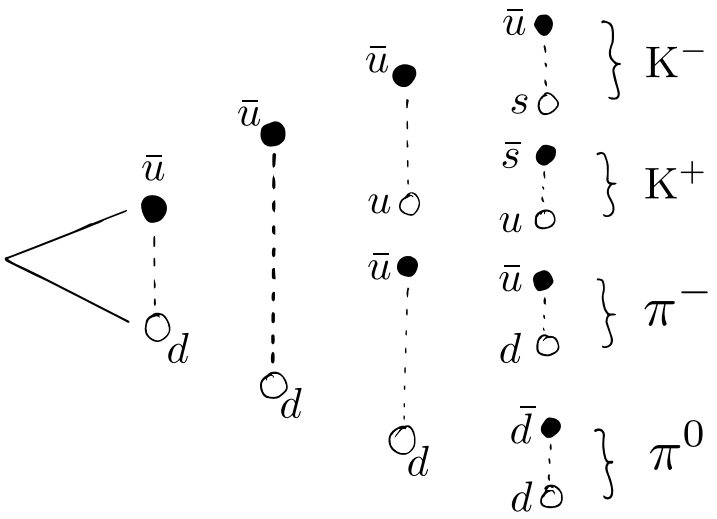


Figure 4.4: Illustration of the hadronization process by which $\bar{u}$- and d-quarks decay into four different mesons. The theoretical strings are shown as dashed lines and particles as circles, where filled circles are antiparticles.

The initial partons produced as Z decay therefore decay to final state hadrons[8] which create a whole "shower" in the direction of the initial parton: this is called a *parton shower* and it is this parton shower observed as particles, a *jet*, that is measured in the detector. The reverse computation from tracks measured in the detector is done with the use of *jet clustering* algorithms. The detector and the clustering algorithms are described in the following section.

[8] To either mesons which consist of two quarks (color–anti-color) or baryons (r-g-b) which consist of three quarks.

4.3 The ALEPH Detector and LEP

The Large Electron Positron collider (LEP) was a particle collider at CERN in Switzerland operating from 1989 to 2000. It collided counter-rotating bunches of electrons and positrons in a giant ring with a circumference of more than 26 km. The first phase, LEP1, ran from 1989 to 1995 at the Z resonance 91 GeV and the second phase, LEP2, continued afterwards closer to 200 GeV for W^+W^- pair production [16], however, it is only the data collected at the energy around $\sqrt{s} = 91.3$ GeV called the Z *peak data* that is used throughout the rest of this project. There were four independent detectors at the LEP experiment, one of them ALEPH[9].

[9] Together with DELPHI, L3, and OPAL.

The *apparatus for LEP physics* (ALEPH) was a particle detector at LEP with a wide coverage, almost 4π, consisting of cylindrical subdetectors, see Figure 4.5, with the coordinate system shown in the upper left corner[10]. The polar angle θ is illustrated in Figure 4.6 to-

[10] The z-axis pointing along the beam direction, the y-axis pointing upwards, and the x-axis pointing towards the center of LEP.

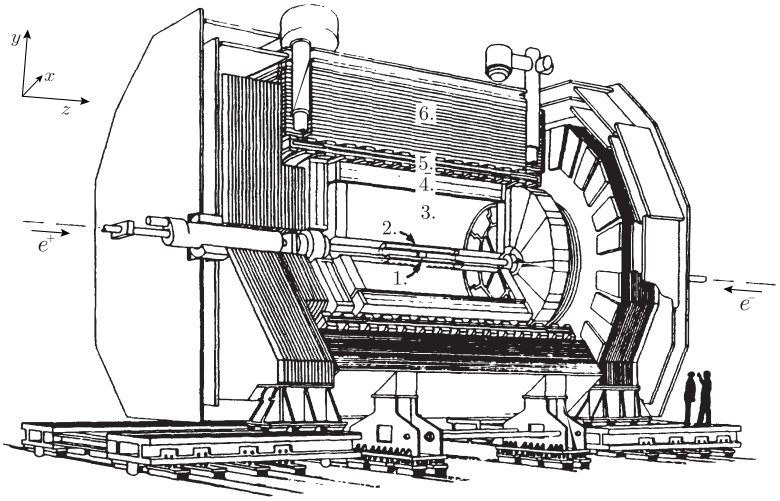


Figure 4.5: The ALEPH detector at LEP. 1) Vertex detector (VDET). 2) Drift chamber (ITC). 3) Time projection chamber (TPC). 4) Electromagnetic calorimeter (ECAL). 5) Superconducting magnet coil. 6) Hadron calorimeter (HCAL). Adapted from Buskulic et al. [32].

gether with the transverse (longitudinal) momentum $p_\perp$ (p_L) and the azimuthal angle ϕ in Figure 4.7. The ALEPH detector was designed to measure the energy deposited in the calorimeters by charged and neutral particles, measure the momenta of charged particles, measure the distance of travel of short-lived particles, and to identify the three lepton flavors (electron, muon, tau) [32]. As can be seen in Figure 4.5, ALEPH consisted of five subdetectors (the vertex detector (VDET), the drift chamber (ITC), and the time projection chamber (TPC)) and two calorimeters (the electromagnetic (ECAL) and the hadronic calorimeters (HCAL)).

The three innermost detectors allow for precise tracking of the charged particles produced in the parton shower and the two outer calorimeters of precise energy measurements for both charged and neutral particles going through the detector. The calorimeters measure the energy of particles by absorbing them.

When a particle interacts with the detector, a small electric charge is measured, referred to as hits. A hadronic event from a parton shower may leave a score of charged tracks resulting in hundreds of hits in the detectors (VDET, ITC, and TPC) which are fitted[11] with Kalman filters [60] to obtain global track fits, of which bad charged tracks are discarded for further analysis. The tracks are helical due to the presence of a 1.5 T magnetic field which curves the charged particles according to their transverse momentum, $p_\perp$.

The energy resolution σ of the calorimeters, or the *calorimeter performance*, is expected to increase with $\sqrt{E}$. In fact, it was found at ALEPH that the energy dependence of the resolution follows the parametrization [32]:

$$\sigma(E) = \Big((0.59 \pm 0.03) \cdot \sqrt{E/\text{GeV}} + (0.6 \pm 0.3)\Big)\,\text{GeV}. \tag{4.2}$$

Even though $\sigma(E)$ increases with E, the relative resolutions improves with higher energies. Since one never measures Nature directly, the results one obtains in a measurement are thus products of both model and experimental uncertainties folded together. To unfold the

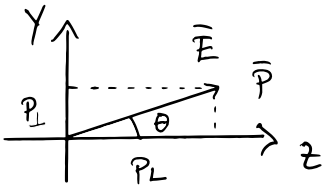


Figure 4.6: The polar angle θ defined in the zy coordinate system

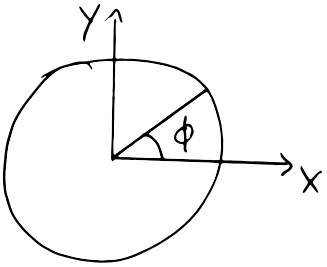


Figure 4.7: The azimuthal angle ϕ defined in the xy coordinate system.

[11] the process of fitting tracks is called *track reconstruction* in high energy particle physics.

measurements to obtain experiment-independent results, the uncertainties are important to understand. Of course there are dozens of other uncertainties in an advanced experiment like ALEPH, however, the energy dependence is the primary focus in this project.

4.4 *Jet clustering*

Since the initial partons created as decay products from the Z are unstable themselves, what is measured in the detector is a whole shower of hadrons seen as charged tracks in the detectors and energy deposits in the calorimeters. However, say that the Z decayed to a $b\bar{b}$ event. In this case the two b's would be back-to-back and the final hadrons would be observed approximately in the same direction as the b's were created. The interest of the experiment is not to measure the final hadrons, but rather to infer information about the initial quarks and gluons. This is done via the reverse-engineering process called *jet clustering*. Over the years many clustering algorithms have been developed, however, most of these are younger than LEP. In the ALEPH experiment the JADE algorithm was used [20]. JADE is a sequential recombination algorithm where final state particles are initially described as individual so-called pseudo-jets which are then recursively merged to larger jets according to their inter-jet distance d_{ij}^2. The distance measure for JADE is:

$$d_{ij}^2 = \frac{2E_iE_j(1-\cos\theta_{ij})}{E_{\text{vis}}^2}, \tag{4.3}$$

where E_{vis} is the visible energy[12] and θ_{ij} is the angle between jet i and j. The JADE algorithm computes d_{ij}^2 for all combinations of jets and merges the two jets with the lowest d_{ij}^2, continuing like that recursively until $\min(d_{ij}^2) > d_{\text{cut}}^2$ for some predefined value of d_{cut}^2. In the dataset at hand, only the final jets were available and not the jet constituents, unfortunately.

[12] The total sum of energies in the event.

4.5 *The variables*

The overall goal of the project is to be able to discriminate quarks and gluons using only vertex variables. The reason for the last condition is that the goal is to better understand the shape distributions of gluons in which there is still significant differences between Monte Carlo (MC) simulations and Data. Therefore only vertex variables will be used to avoid any biases introduced by using shape-related variables to detect differences in shape-distributions. The vertex variables are a subset of all variables which include the three variables `projet`, `bqvjet`, and `ptlrel`. These three particular variables have each shown discriminatory power in separating b-quarks from light quarks and gluons.

`projet`: PROBABILITY OF SIGNIFICANT LIFETIME. For each track in the jet an impact parameter δ is computed. This parameter is

the minimum distance between the estimated Z decay point and the track itself and its sign depends on whether or not the point of closest approach is in front of or behind the Z decay point (relative to the momentum). From δ the significance $\mathcal{S}$ – which is δ/σ_δ – is computed and is thus a measure of the certainty of a measured track being from primary vertex. High values of $\mathcal{S}$ is typically an indicator of b jets, since long-lived particles typically decay in front of the Z relative to the jet direction, while *uds*-jets generally have small significance and might as well have negative values of $\mathcal{S}$. An illustration of the difference in significance between *uds*-jets and b-jets can be seen in Figure 4.8.

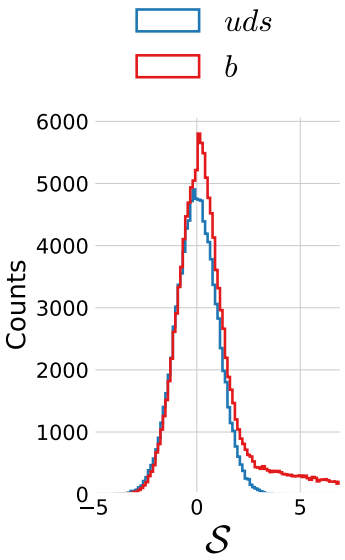


Figure 4.8: Distribution showing the difference in significance $\mathcal{S}$ between *uds*-jets and b-jets. Based on own, simulated data to illustrate this difference.

From $\mathcal{S}$ the track probability $\mathcal{P}_{\text{track}}$ of a track originating at the decay point of the Z can be computed, which can further be aggregated across all tracks within a jet to form the jet probability $\mathcal{P}_{\text{jet}}$ which `projet` is a function of [46]. Whether or not $\mathcal{P}_{\text{jet}}$ is strictly a probability can be discussed but it is related to the probability of all tracks within a jet to originate from long-lived particles, which itself is a good indicator of being a b- (or c-) jet. This variable further has the advantage of being independent of any vertex algorithm.

`bqvjet` : *b*-QUARK VERTEX OF JET. For any jet with well measured[13] charged tracks, a fit with a (hypothetical) secondary vertex is performed. The difference in χ^2 between the null hypothesis that all good tracks originate from the same primary vertex and the alternative hypothesis that a secondary vertex exists in addition to the primary one is calculated. For the long-lived massive b and c quarks this typically results in large differences in χ^2 compared to *uds*- and gluon jets which have much lower $\Delta\chi^2$-values [16]. The `bqvjet` is related to the $\Delta\chi^2$-value from the secondary vertex algorithm. This value is dependent of the vertex algorithm, but still explores other areas of phase space than `projet`, however, they are still very correlated. The linear correlations[14] ρ_{q_i} between `projet` and `bqvjet` for q_i jets are $\rho_b = 0.80, \rho_c = 0.65, \rho_{uds} = 0.23, \rho_g = 0.29$.

[13] Meaning that there are at least four TPC hits and the fit has a reduced χ^2 of less than four [16].

[14] Based on MC truth.

`ptlrel` : RELATIVE LEPTON MOMENTUM. If any leptons (in the case of $e^\pm$ or $\mu^\pm$) are measured in the jet by the detector, this is a good sign of the jet originating from a b-quark as $\sim 11\,\%(e) + \sim 11\,\%(\mu)$ decay semi-leptonically [13][15]. The high mass of the b quark leads to high $p_\perp$ for the leptons relative to the jet axis which is exactly measured by `ptlrel`.

[15] $\mathcal{B}(B \to lvX) = (10.5 \pm 0.3)\,\%$, $l = \{e, \mu\}$ [13].

The fact that the heavy b-quarks have much longer lifetimes than the lighter *uds*-quarks stems from their much lower coupling magni-

tudes written as the CKM matrix **V** [73]:

$$\mathbf{V} = \begin{array}{c} \\ u \\ c \\ t \end{array} \begin{array}{c} \begin{array}{ccc} d & s & b \end{array} \\ \begin{pmatrix} 0.97446 & 0.22452 & 0.00365 \\ 0.22438 & 0.97359 & 0.04214 \\ 0.00896 & 0.04133 & 0.99911 \end{pmatrix} \end{array}. \tag{4.4}$$

The matrix element $|V_{ij}|^2$ is proportional to the transition-probability of quark i transitioning to quark j. From the CKM matrix it can be seen that u and d quarks couples strongly together, likewise with c-s and b-t quark pairs. When a Z decays into a b-quark, this quark couples strongly with the top quark, however, due to the high mass of the top quark compared to the b-quark, the b-quark cannot decay into a t-quark but must (almost always) decay to a c, however, still with low probability, $V_{bc} \ll 1$. This, together with the fact that $V_{bu} \ll V_{bc}$ explains the long life-time of b quarks, $\tau_b \sim 1.3 \times 10^{-12}$ s [80]. This is also why the three variables above are very common variables for b-tagging algorithms. That c-quarks also have relative long life-times, $\tau_c \sim 1.1 \times 10^{-12}$ s [80], are not due to the CKM elements, as for b-quarks, but rather due to the c-decay being governed by the weak force through virtual W^* bosons, a force that is much weaker than the strong force (hence the name). The low phase space in a c-quark decay makes the c-quark longer-lived. This also happens for b-quarks which further explains why c-quarks share many similarities with b-quarks but also resembles light-quarks.

The rest of the non-vertex variables are:

`ejet` : The energy of the jet E_{jet}

`costheta` : The cosine of the θ angle defined in Figure 4.6.

`phijet` : The angle ϕ of defined in Figure 4.7: ϕ.

`sphjet` : The sphericity tensor **S** is defined as:

$$S^{(\alpha\beta)} = \frac{\sum_{i=1}^{N} p_i^{(\alpha)} p_i^{(\beta)}}{\sum_{i=1}^{N} |p_i|^2} \quad \alpha, \beta \in \{x, y, z\}, \tag{4.5}$$

and the sphericity is determined as $S = \frac{3}{2}(\lambda_2 + \lambda_3)$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3$, $\lambda_1 + \lambda_2 + \lambda_3 = 1$ are the three eigenvalues of the sphericity tensor. The sphericity $0 \leq S \leq 1$ is a measure of the angular distribution of the tracks and clusters in a jet. When $S = 0$ the jets form a perfect sphere, compared to $S = 1$ for a perfect line. The `sphjet` variable is the sphericity of the jet when calculated in its boosted rest frame, also known as *boosted sphericity*.

`pt2jet` : The sum of the square of transverse momentum w.r.t. the jet axis: $\sum_i p_{\perp,i}^2$.

`muljet` : The rescaled multiplicity of the jet.

For further details about the variables, see Armstrong [16].

The variables explained above are all used in the following analysis where the machine learning model is trained on only the vertex variables to probe differences in the shape-variables. The goal of this is to better understand the gluon hadronization process to minimize differences in MC simulations and ultimately get a better understanding of the rules governed by Nature.

5. Quark Gluon Analysis

"Research is what I am doing I don't know know what I'm doing."

— Wernher von Braun

The analysis of the quarks and gluons that were introduced in the previous chapter is described here. The overall goal is to be able to discriminate between quark and gluon jets to better be able to describe the gluon jets. The gluon jet distributions are measured in 3-jet events and how they split in 4-jet events. This chapter is organized as follows. In section 5.1 the data are presented and the initial cuts are described and the variables are visualized in section 5.2. The choices of loss and evaluation function are discussed in section 5.3. In section 5.4 the first of the two overall models developed in this chapter is presented, the b-tagging model. The efficiency of this model is measured in section 5.5. The second model, the g-tagging model is used for classification of entire events, compared to the b-tagging model which classifies individual jets, and is introduced in section 5.6 and its efficiency is measured in section 5.7. The jet distributions in 3-jet events are analyzed in section 5.8 and their the gluon splitting in 4-jet events in section 5.9. Finally the quark gluon project is discussed in section 5.10.

5.1 Data Preprocessing

The data files were acquired from Prof. Peter Hansen (NBI) who worked on the ALEPH experiment. The Data consists of 43 data files from between 1991 and 1995 totalling 3.5 GB (Data). Along with this comes 125 files based on Monte Carlo (MC) simulations (8.4 GB) and additional 42 MC-files with only b-quark events (MCb) (2.1 GB). The data files which are in the form of *Ntuples*, ROOT's data format [29], are converted to HDF5-files by using the Python package uproot [7]. While iterating over the Ntuples, some basic cuts are applied before exporting the data to HDF5-format. The first one being that the (center of mass) energy E in the event has to be within $90.8\,\text{GeV} \leq E \leq 91.6\,\text{GeV}$ to only use the Z peak data. The second one being that the sum of the momenta p_{sum} in each event is $32\,\text{GeV} \leq p_{\text{sum}}$ to remove $Z \to \tau^+\tau^-$ events. To ensure a primary vertex, at least two good tracks are required where a good track is

defined as having at least 7 TPC hits and 1 silicon hit or more. Finally it is required that the cosine of the thrust axis polar angle, which is the angle between the trust axis and the beam, is less than or equal to 0.8 to avoid any low angle events since the detector performance worsens significantly in that region. These cuts were standard requirements for the ALEPH experiment.

One last cut which was experimented with was the threshold value for *jet matching*. The jet matching is the process of matching the jet with one of the final state quarks. The jet is said to be matched if the dot product between the final quark momentum and the jet momentum is more than then threshold value. Higher thresholds means cleaner jets but at the expense of less statistics. A jet matching threshold of 0.90 was found to be a good compromise between purity and quantity where 97.8 % of all 2-jet events are matched and 96.7 % of all other jets were matched[1].

[1] Compare this to 98.5 % and 97.8 % for a threshold of 0.85 or 95.9 % and 93.9 % for a threshold of 0.95.

The data structure is quite differently structured in the Ntuples compared to normal structured data in the form of tidy data [101]. The data is organized such that one iterates over each event where the variables are variable-length depending on the number of jets in the events; this is also known as *jagged* arrays. The data is un-jagged[2] before exporting to HDF5-format and only the needed variables are kept. This reduces the total output file to a 2.9 GB HDF5-file including both Data, MC, and MCb.

[2] Such that e.g. a 3-jet event will figure as three rows in the dataset.

The number of events for each number of jets can be seen in Table 5.1 for the Data and in Figure 5.2 for the MC and MCb.

n	# jets	# events
2	2 359 738	1 179 869
3	3 619 290	1 206 430
4	854 336	213 584
5	52 775	10 555
6	510	85
Total	6 886 649	2 610 523

Table 5.1: The dimensions of the dataset for the actual Data for n-jet events. The numbers in the jet columns are the number of events multiplied with the number of jets; e.g. $85 \cdot 6 = 510$.

n	# jets	# events
2	7 293 594	3 646 797
3	10 780 890	3 593 630
4	2 241 908	560 477
5	103 820	20 764
6	588	98
Total	20 420 800	7 821 766

Table 5.2: The dimensions for the MC and MCb datasets for n-jet events.

5.2 *Exploratory Data Analysis*

Since the machine learning models are only trained on the three vertex variables `projet`, `bqvjet`, and `ptljet` – see chapter 4 for a deeper introduction to these variables – these variables will be the primary focus of this section. Given the fact that MC-simulated data exists, the truth of each simulated event is also known. This allows us visualize the difference between the different types of quarks. In the MC simulation each event are generated such that the type of quark, or *flavor*, is known and assigned the variable `flevt`. The mapping from flavor to `flevt` is:

Flavor:	*bb*	*cc*	*ss*	*dd*	*uu*
`flevt`:	5	4	3	2	1

In addition to knowing the correct flavor, we define that an event is *q-matched* if one, and only one, of the jets are assigned to one of the (final) quarks, if one, and only one, of the jets are assigned to the other (final) quark, and if no other jets are matched to any of the (final) quarks. We then define what constitutes a *b*-jet: if it has `flevt` $= 5$, the entire event is *q*-matched, and the jet is matched to one of the quarks. Similarly we define *c*-jets only with the change that `flevt` $= 4$, and *uds*-jets[3] with `flevt` $\in \{1, 2, 3\}$. A gluon

[3] Will also be called a *l*-jet.

jet is defined to be a jet in a (any-flavor) q-matched event where the jet itself is not assigned to any of the (final) quarks. Strictly speaking, this means that the gluon jet is not 100 % certain of being a gluon. We cannot know this, as not all of the input parameters in the MC simulation are known, only the final clustered jets. Due to the q-match criterion this also means that some jets are assigned the label "non-q-matched" which is regarded as background. The distribution of different types of jets can be seen in Table 5.3 and shown as relative numbers in Table B.1.

n	b	c	uds/l	g	non-q-matched
2	2 713 454	944 380	2 125 900	0	1 509 860
3	2 433 878	964 212	2 129 218	3 365 969	1 887 613
4	326 264	156 332	336 548	1 012 198	410 566
5	10 332	5960	12 668	54 525	20 335
6	42	26	52	320	148
Total	5 483 970	2 070 910	4 433 012	4 604 386	3 828 522

Table 5.3: Number of different types of jets for MC and MCb for n-jet events. See also Table B.1 for relative numbers.

With the criteria defined above for what constitutes a specific type of jet the 1D-distributions for the three vertex variables are plotted in Figure 5.1. For all three subplots the histograms are shown with logarithmic y-axes, all b-jets in blue, c-jets in red, g-jets in green, and all of the jets (no matter their type) are shown in orange. The distributions for 2-jet events are shown in fully opaque color, 3-jet events in dashed lines, and 4-jet events in lighter colors. In the left subplot the `projet` variable is plotted where it can be seen that high values of `projet` tend to indicate b-jets. In the middle subplot `bqvjet` is plotted which shares many similarities with the `projet` variables, including that high values indicate b-jets. In the right subplot the `ptljet` is plotted. This variable has many zeros in it which correlates with mostly with gluons[4] and large values are mostly due to b-jets. In general it is clear to see how the differences in distribution between the 2-, 3-, and 4-jet events are minor, with the one exception of 2-jet events which does not contain any gluons at all.

[4] Around 98 % of all g-jets are zeros for the variable `ptljet` compared to $\sim$ 82 % for c-jets and $\sim$ 70 % for b-jets.

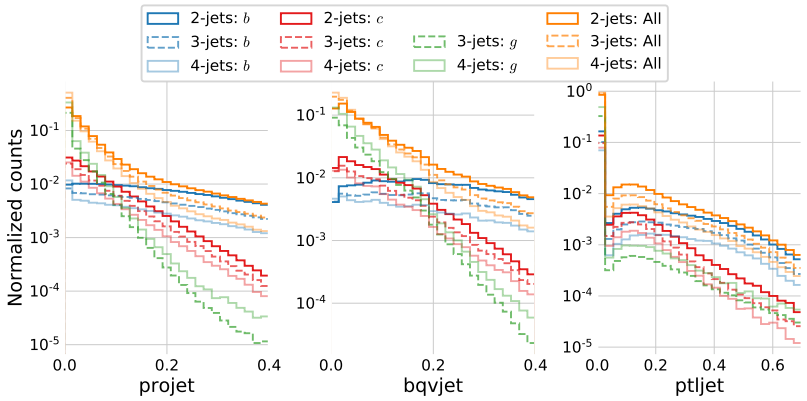


Figure 5.1: Normalized histograms of the three vertex variables: `projet`, `bqvjet`, and `ptljet`. In blue colors the variables are shown for true b-jets, in red for true c-jets, in green for true g-jets, and in orange for all of the jets (including non q-matched). In fully opaque color are shown the distributions for 2-jet events, in dashed (and lighter color) 3-jet events, and in semi-transparent 4-jet events. Notice the logarithmic y-axis, that there are no g-jets for 2-jet events (as expected), and that all of the distributions are very similar not matter how many jets.

5.2.1 *Dimensionality Reduction*

Even though there are only three vertex variables, it is difficult to properly get an intuition about how easily separated the different types of jets are. Since there are millions of points a single 3D scatter plot quickly becomes overcrowded if one plots all jets. We apply dimensionality reduction from the three dimensions down to two dimensions by using the UMAP algorithm [70]. Within recent years the field of dimensionality reduction algorithms has grown a lot from just the typical (linear) principal component analysis to also include nonlinear algorithms. The t-SNE algorithm [95] deserves an honorable mention since this algorithm revolutionized the usage of (nonlinear) dimensionality reduction algorithms in e.g. bioinformatics [92, 99] yet its mathematical foundation has strongly been improved with the newer, faster UMAP algorithm [70] which usage is also expanding [21, 22, 42].

The aim of UMAP, short for Uniform Manifold Approximation and Projection, is to correctly identify and preserve the structure, or topology, of the high-dimensional feature space in a lower-dimensional output space. It does so by trying to stitch together local manifolds in the high-dimensional feature space such that the difference between the high- and low-dimensional representations is minimized according to the cross-entropy such that both global structure and local structure is preserved [70]. Compared to t-SNE the approach in UMAP has a topological background compared to the more heuristic approach taken by t-SNE. Note that the UMAP algorithm is not provided any information about which jets are which types or any other truth information.

The UMAP algorithm has several hyperparameters, where two of the most important ones are the number of neighbors `n_neighbors` which controls the priority between correctly preserving the global versus the local structure, and the `min_dist` which defines how tightly together UMAP is allowed to cluster the points in the low-dimensional representation. To properly choose the best combination of `n_neighbors` and `min_dist` a grid search with `n_neighbors` $\in \{10, 50, 100, 250\}$ and `min_dist` $\in \{0, 0.2, 0.5\}$ is performed. This is shown for 4-jet events in Figure B.1.

The best combination of `n_neighbors` and `min_dist` is subjective at best, but I judged that `n_neighbors` $= 250$ and `min_dist` $= 0.2$ gave the best compromise between preserving local and global structure. The results of running UMAP on 4-jet events can be seen in Figure 5.2. Here the millions of points are plotted using Datashader [8] to avoid overplotting and colored according to the jet type. From the figure it is seen how there are some clear *b*-jet clusters, however, most of the data seem to be a mix of *g*-and *uds*-jets. The plots with the same UMAP parameters for 3-jet and 2-jet events are seen in Figure 5.3 and 5.4.

These figures suggest that it should be possible to discriminate the *b*-jets from the other jets somewhat, however, no clear separation is

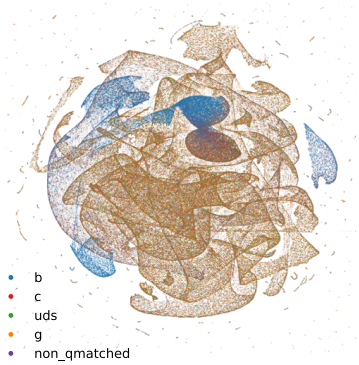


Figure 5.2: Visualization of the vertex variables in 4-jet events for the different categories: true b-jets in blue, true c-jets in red, true uds-jets in green, true g-jets in orange, and non q-matched events in purple. The clustering is performed with the UMAP algorithm which outputs a 2D-projection. This projection is then visualized using the Datashader which takes takes care of point size, avoids over and underplotting, and color intensity.

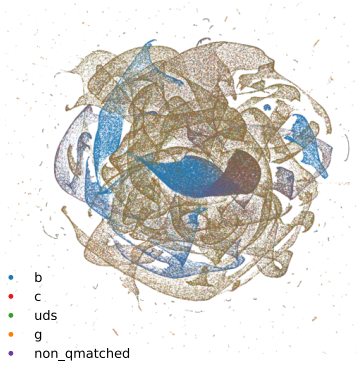


Figure 5.3: UMAP visualization of vertex variables for 3-jet events.

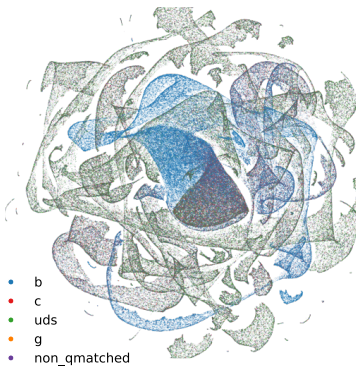
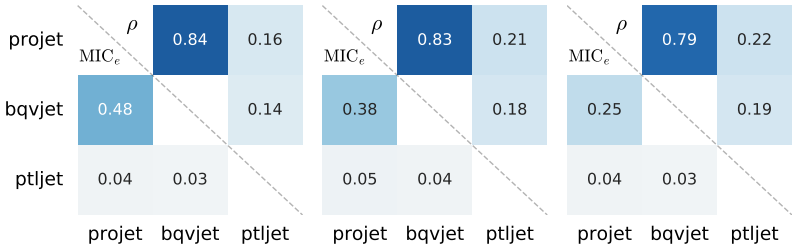


Figure 5.4: UMAP visualization of vertex variables for 2-jet events.

expected. The t-SNE algorithm was also tested but showed inferior performance compared to UMAP, see Figure B.2 for an example of this.

5.2.2 *Correlations*

The correlation between the vertex variables can be seen in Figure 5.5, where the upper diagonal shows the linear correlation ρ and the lower diagonal shows the nonlinear correlation MIC_e introduced in subsection 3.1.1. Here it can be seen that `projet` and `bqvjet` are the two variables that correlate the most whereas the other variables correlate a lot less. Had they all correlated a lot, it would be more difficult to extract any meaningful insights from the system as it would contain less information.



(a) 2-jet events (b) 3-jet events (c) 4-jet events

Figure 5.5: Correlation of the three vertex variables for 2-, 3- and 4-jet events. Here the upper diagonal shows the linear correlation ρ and the lower diagonal the nonlinear correlations MIC_e.

5.3 *Loss and Evaluation Function*

In contrary to the housing prices project, the goal in this project is to predict types of jets or events where the *signal* observations[5] are often assigned the label 1 and *background* observations 0. The combination of this being a *classification* problem (compared to a regression problem) along with the fact all the variables are actual measurements from a particle physics accelerator means that the issue of outliers is negligible. This also means that the problem of finding a robust loss function is less important since the in classification loss is already bounded in the $[0, 1]$-interval.

Accuracy, which is simply the fraction of correct predictions, is often used as the loss function in classification, however, accuracy as a metric suffers a lot when handling *imbalanced* data: when the ratio between the number of instances of each class is not approximately fifty-fifty. The problem is that if the sample contains 90 % background and only 10 % signal, then a simple model which simply predicts everything to be background will have a 90 % accuracy.

To circumvent this issue, the area under the *ROC*[6] curve (*AUC*) is used, where the ROC curve is the the *signal efficiency* ε_{sig} of the ML model plotted as a function of the *background efficiency* ε_{bkg}. The definition of these two measures are:

$$\varepsilon_{\text{sig}} = \frac{S_{\text{sel}}}{S_{\text{tot}}}, \qquad \varepsilon_{\text{bkg}} = \frac{B_{\text{sel}}}{B_{\text{tot}}}, \tag{5.1}$$

[5] Often called signal events, however, since a jet can also be signal, the term signal event is only used when the meaning is clear from the context.

[6] Receiver Operating Characteristic.

Strictly speaking it is not a function of the background efficiency, but rather ε_{sig} and ε_{bkg} plotted parametrically as functions of the threshold cut $\hat{y}_{\text{cut}}$.

where S_{sel} are signal events that were also selected (predicted) as signal by the ML model, $S_{\text{tot}} = S_{\text{sel}} + S_{\text{rej}}$ is the total number of signal events (the selected and rejected), and likewise for background events B. Within the machine learning community the signal efficiency is called the true positive rate (TPR) and the background efficiency the false positive rate (FPR). For the rest of this project, the AUC will be the evaluation function $f_{\text{eval}} = \text{AUC}$, however, since this metric does not work on single observations it cannot be used as the loss function. Instead we will use the *log-loss* as the loss function[7] which in comparison to the AUC is not only differentiable for single predictions but also takes the certainty of the prediction into account. When using tree-based algorithms or neural networks one can compute not only whether or not a single observation is classified as signal or background but also a prediction score. This is a number in the $[0, 1]$-interval and the closer to 1 the score is, the more certain the model is of the prediction being signal. Given the prediction score $\hat{y}$ and the true label y, the log-loss $\ell_{\log}$ is calculated as:

$$\ell_{\log}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log (1 - \hat{y}). \tag{5.2}$$

This is visualized in Figure 5.6. Here it can be seen how the loss changes as a function of the prediction score. Notice that when $y = 0$ the loss for $\hat{y} = 1$ diverges towards ∞ and likewise with $y = 1$ and $\hat{y} = 0$ (since $\log 0$ diverges to -∞).

[7] In the context of machine learning this is the same as the *cross entropy*.

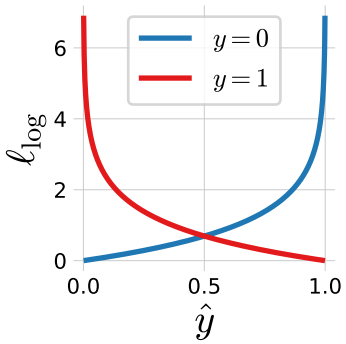


Figure 5.6: Plot of the log-loss $\ell_{\log}$ for background ($y = 0$) in blue and signal ($y = 1$) in red.

5.4 *b-Tagging Analysis*

The ability to discriminate between the different types of particles produced in a collision is obviously import to understand the results. Today much work go into tagging algorithms, e.g. b-tagging in ATLAS and CMS [83]. That b-quarks are tagged specifically is both due to b-quarks having more unique characteristics compared to e.g. c-quarks and are thus easier to tag, but also the fact that b-quarks are the second-heaviest of the quarks and are measured to better understand CP-violation[8] at LHC-b, contributes to the choice of tagging b-quarks. In ALEPH Proriol et al. [76] started the work of comparing different methods for b-tagging already in 1991. They concluded that a neural network had the best performance compared to e.g. a linear (Fisher) discriminant. The neural network used was a 3-layer neural network (NN) trained on nine variables and the outputted the prediction score `nnbjet`. From here on, this pre-trained network will be called NNB.

[8] Short for charge-parity.

I split the data into training and test sets in such a way that the individual jets in an event are not split. The events are split in a (80-20)% train-test ratio.

5.4.1 *b-Tagging Hyperparameter Optimization*

Compared to the housing prices dataset, the number of observations N is a lot larger ($3 \times 10^7 \gg 5 \times 10^5$), although the dimensionality

M is much smaller ($3 \ll 143$). Therefore both XGBoost (XGB) and LightGBM (LGB) were included as models initially since their performances in the housing dataset were very similar. LightGBM was expected to be faster on this dataset due to the large N. The models were hyperparameter optimized (HPO) using random search (RS) since the Bayesian optimization (BO) did not show any performance gains compared to RS in the housing project. They were both hypermeter optimized with 5-fold cross validation and early stopping with a patience of 100. The PDFs for the random search for the LightGBM model can be seen in Table 5.4, and the ones for XGBoost in Table B.2. The random search has been run with 100 iterations for LightGBM and only 10 iterations for XGBoost since XGBoost turned out to be very slow[9] at fitting datasets of this size. The results of the HPO for 3-jet and 4-jet events can be seen in Figure 5.7. For 3-jets it can be seen how most of the iterations share about the same performance (within 1σ), however, some iterations show a significant decrease in performance. The same clear pattern is not seen in the 4-jet events.

Hyperparameter	Range
`subsample`	$\mathcal{U}(0.4, 1)$
`colsample_bytree`	$\mathcal{U}_{\text{trunc}}(0.4, 1, 2)$
`max_depth`	$\mathcal{U}_{\text{int}}(-5, 63)$
`num_leaves`	$\mathcal{U}_{\text{int}}(7, 4095)$

Table 5.4: Probability Density Functions for the random search hyperparameter optimization process for the LightGBM model. For an explanation of $\mathcal{U}_{\text{trunc}}$, see subsection 5.6.2. All negative values of `max_depth` are interpreted as no max depth by both LGB and XGB.

[9] See page 70 for a discussion of the timings.

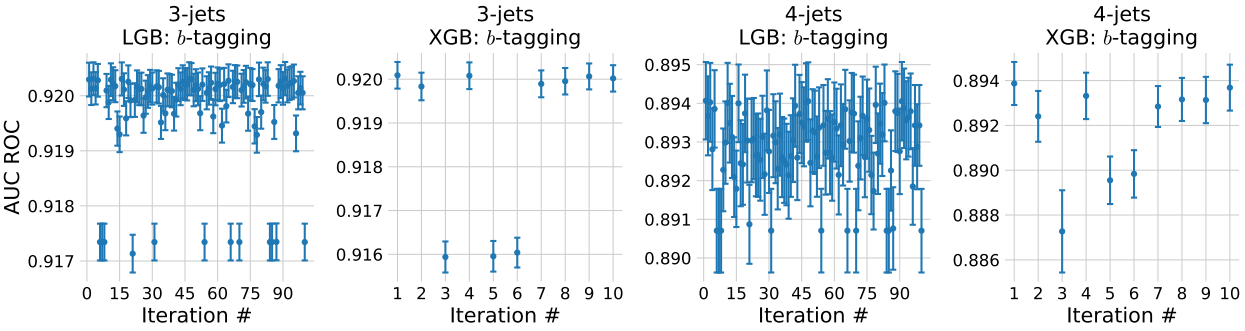


Figure 5.7: Hyperparameter Optimization results of b-tagging with random search. From left to right, we have A) 100 iterations of RS with LGB on 3-jets, B) 10 iterations of RS with XGB on 3-jets, C) 100 iterations of RS with LGB on 4-jets, D) 10 iterations of RS with XGB on 4-jets. Notice the different ranges on the y-axes.

The relationship between the different hyperparameters in 4-jet events can be seen in the parallel coordinate plot in Figure 5.8. First of all the importance of the column downsampling `colsample_bytree` variable is significant: all of the low-performing hyperparameter sets have a low value of this hyperparameter. Since $M = 3$ for the vertex variables this makes logical sense; using only $\text{int}(\sim 0.5 \cdot 3) = 1$ variable[10] the model cannot properly learn the structure in the data. Compared to the column downsampling, the other hyperparameters are less important. The same overall conclusion can be inferred in the 3-jet case, see Figure B.3.

[10] See subsection 5.6.2 for a deeper discussion about the `colsample_bytree` hyperparameter.

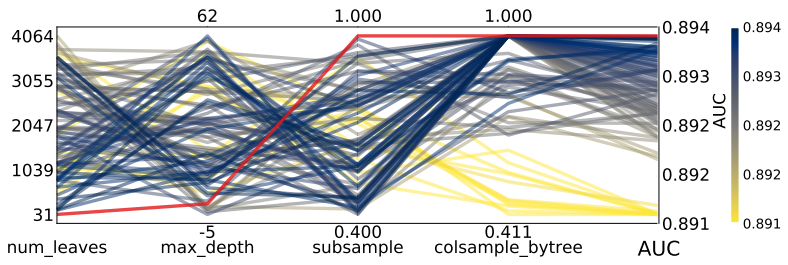


Figure 5.8: Hyperparameter optimization results of b-tagging for 4-jet events. The results are shown as parallel coordinates with each hyperparameter along the x-axis and the value of that parameter on the y-axis. Each line is an event in the 4-dimensional space colored according to the performance of that hyperparameter as measured by AUC. The single best hyperparameter is shown in red.

5.4.2 *b-Tagging Results*

The prediction score $\hat{y}$ is usually called the b-tag for b-tagging models and will be written as β_{tag}. The distribution of the b-tags for the two HPO-optimized models, LGB and XGB, together with the pre-trained neural network, NNB, can be seen in Figure 5.9 for 4-jet events and in B.4 for 3-jet events. Notice the strong match between the NNB and LGB models. The XGB model has almost no high b-tags ($\beta_{\text{tag}} > 0.8$), but a majority of b-tags in the very low end. This indicates that the XGBoost has focussed on the background events compared to the signal events, whereas the NNB and LGB models have focused more on the signal events.

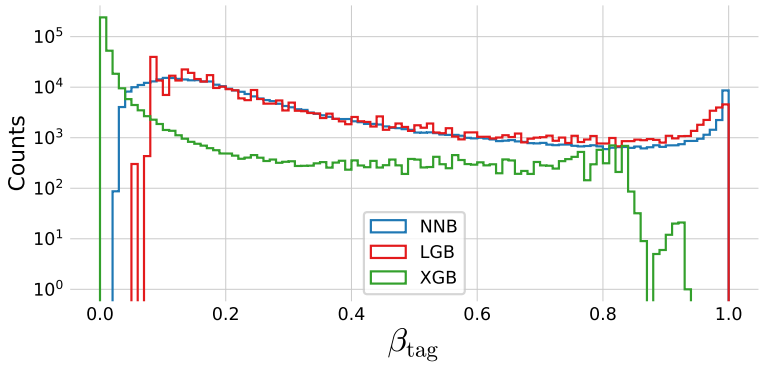


Figure 5.9: Histogram of b-tag scores β_{tag} in 4-jet events for NNB (the neural network pre-trained by ALEPH, also called `nnbjet`) in blue, LGB in red, and XGB in green.

Even though the distributions of b-tags are different between the three models, the real performance plot for classification is the ROC curve seen in Figure 5.10 for 4-jet events. Here the signal efficiency ε_{sig} is plotted as a function of the background efficiency ε_{sig} with the AUC shown in the bottom right corner. The LGB and XGB models performs similarly well with an AUC $= 0.896$ compared to the NNB with AUC $= 0.884$. The differences between the models are even smaller for 3-jet events seen in Figure B.5. In general the LGB and XGB models are so similar that they cannot be distinguished from another in any of the plots and their difference in AUC is on the fourth decimal point.

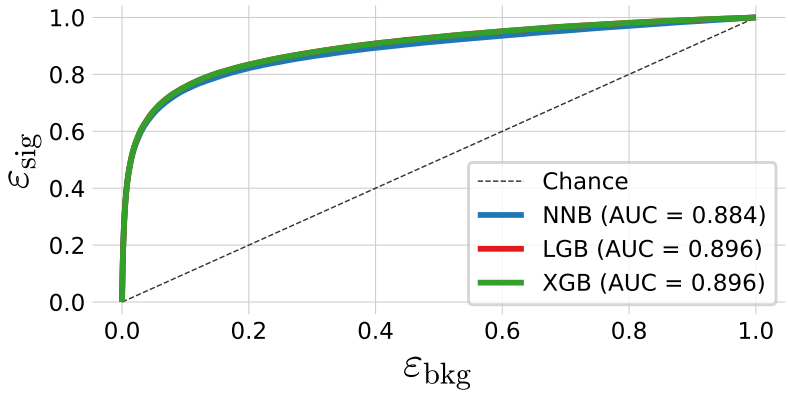


Figure 5.10: ROC curve of the three b-tag models in 4-jet events for NNB (the pre-trained neural network trained by ALEPH, also called `nnbjet`) in blue, LGB in red, and XGB in green. In the legend the area under curve (AUC) is also shown. Notice that the LGB and XGB models share performance and it is thus due to overplotting that only the green line for XGB can be seen.

The LGB model is, however, several times faster than the XGB model. In comparison, 10 iterations of HPO using RS on 3-jet events with XGB took more almost 34 hours on HEP[11] compared to just 23 hours for 100 iterations for LGB. The same performance difference

[11] The local computing cluster.

was seen in 4-jet events where the timings were 4 hours for XGB compared to 2.5 hours for LGB. Since their performance is similar, XGB is dropped in the subsequent analysis.

The distribution of the b-tag scores β_{tag} for signal and background in 4-jet events can be seen in Figure 5.11. The separation between the heavier quarks and light quarks (and gluons) is clear at high values of β_{tag}, however, a lot of c-quarks also get a high b-tag score. The same is seen for 3-jet events in Figure B.6.

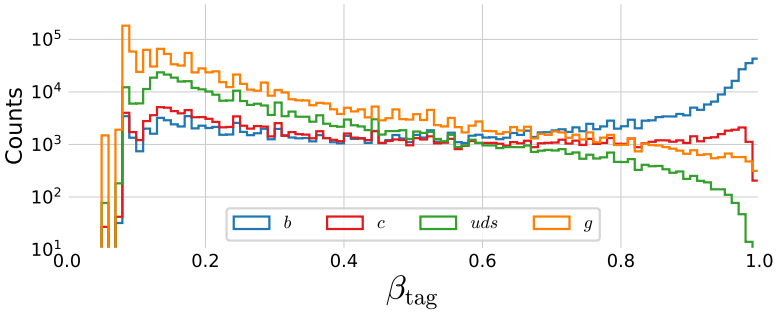


Figure 5.11: Distribution of b-tags in 4-jet events for b-jets in blue, c-jets in red, uds in green and g in orange.

5.4.3 *b-Tagging Model Inspection*

To get a better understanding of the trained LGB model, the global SHAP feature importances are shown in Figure 5.12 for 4-jet events. First of all it is noted that the `projet` has global feature importance of 57.32 %, `bqvjet` 29.16 %, and `ptljet` 13.52 %. For all three variables it is seen how most of the points have many small feature values which has a (small) negative impact on the model output. Especially the `ptljet` has many features with a low value (0 in fact) yet this does not pull the model too much towards background events. Compared this to a jet with a high value of `ptljet` which has a strong, positive impact on the output prediction.

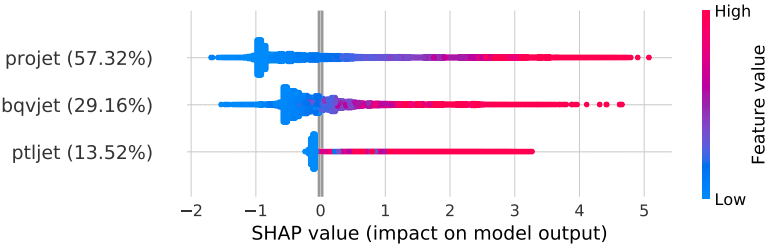


Figure 5.12: Global feature importances for the LGB b-tagging algorithm on 4-jet events. The normalized feature importance is shown in the parenthesis and the each dot is an observation showing the dependance between the SHAP value and the feature value.

In regression, the model output is a continuous prediction $\hat{y}_{\text{reg}} \in \mathbb{R}$. In classification what is actually happening under the hood is that the model predicts a value $\tilde{y} \in \mathbb{R}$ which is transformed to a number in the $[0, 1]$-interval via the *expit* function:

$$\text{expit}(\tilde{y}) = \frac{e^{\tilde{y}}}{1 + e^{\tilde{y}}} \equiv p, \tag{5.3}$$

where p is a number in the $[0, 1]$-interval. The expit function is also sometimes known as the logistic function and is visualized in Fig-

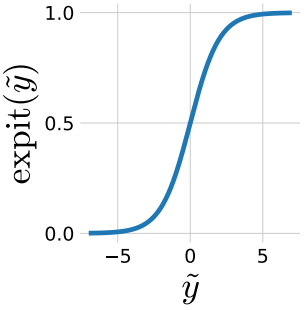


Figure 5.13: The expit function.

ure 5.13. Its inverse is the *logit* function:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \tilde{y}, \tag{5.4}$$

which is visualized in Figure 5.14. The fraction in equation (5.4) is called the *odds* and the logit-transformed value of p, $\text{logit}(p) = \tilde{y}$, is thus sometimes called the *log-odds*. Because LightGBM makes its predictions in this log-odds space, the SHAP values in Figure 5.12 are also in log-odds space[12].

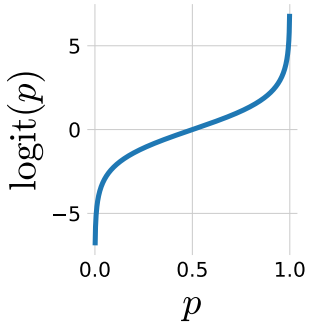


Figure 5.14: The logit function.

With this in mind, single predictions of the LGB b-tagging model can be understood with SHAP which Figure 5.15 is an example of. This plot, which is my own extension to Figure 3.22, shows the logic behind the model's prediction for a particular jet in a particular 3-jet event. That the bias is negative reflects that there is a majority of background jets compared to signal jets[13]. This particular event has `projet` $= 1.003$, `bqvjet` $= 0.529$, and `ptljet` $= 0$. In the plot it is seen how this high value of `projet` has the greatest impact on the model prediction, while the medium value of `bqvjet` also pushes the model prediction towards a signal-prediction. The four red and green bars in the left part of the plot are all in log-odds space and their sum is shown as the blue bar to right, where the right y-axis shows the value in probability space $p \in [0, 1]$. This jet was in fact a b-jet.

[12] The additivity property of SHAP, see section 2.8, is thus also in this log-odds space.

[13] Only around 22 % of all the jets are b-jets.

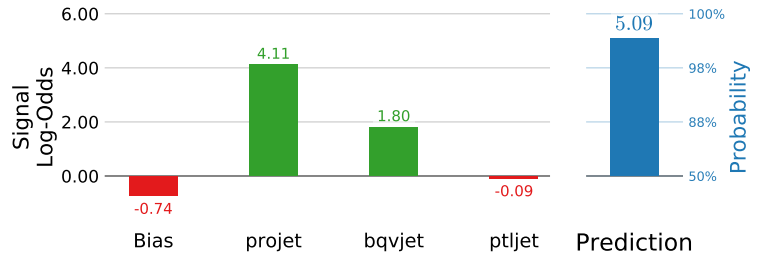


Figure 5.15: Model explanation for the 3-jet b-tagging LGB model for a b-like jet. The first column is the bias which can be seen as the naive prediction baseline, the rest are the input variables. On the right hand side of the plot is the model prediction shown. The left part of the plot is shown in log-odds space, the right part in probability space. The negative log-odd values are shown in red, positive ones in green, and the prediction value in blue.

5.5 *b-Tagging Efficiency*

Before any further analysis can be done, the efficiency of the b-tagging model has to be measured. The efficiency ε is defined as the number of particles, events, jets, or any other countable measure, N_{sel}, that are selected by the algorithm divided by the *true* number, N_{truth}:

$$\varepsilon = \frac{N_{\text{sel}}}{N_{\text{truth}}}. \tag{5.5}$$

Of course, the truth is never known in Nature, however, it is for simulated MC events. The efficiency is used to estimate how many particles (e.g.) that were generated even though only a subset of the particles were detected. Imagine a hypothetical experiment where 21 particles were observed and the efficiency of the experiment was $\varepsilon = 50\,\%$. This means that there were created $21/\varepsilon = 42$ particles in the experiment, yet only 21 of them were observed.

For measuring the b-tagging efficiency we apply a Tag-Tag-Probe (TTP) method based on the b-tags. In 3-jet events two of the jets will serve as tags and the last one as probe. The tags are jets where, if they are known, the probe is also known (with high probability). One can then apply the cut to the probe and see if it would have passed the cut or not. This method provides a clean and unbiased sample (the probes) and since (with high probability) the truth of the probe jet is known, the efficiency can be measured in this way [37]. Since the TTP method does not depend on real truth, it can be used on both MC and Data.

To measure the b-tagging efficiency we use that the Z decay has the form $Z \to q\bar{q}g$. Quantum field calculations based on the Standard Model predicts that the decay $Z \to ggg$ is extremely unlikely with a branching ratio of $\sim 10^{-6}$ [94]. This decay was actually tested at LEP which did not find any ggg events and concluded that the branching ratio was $< 1.6\,\%$ with an upper limit at 95 % confidence level [41]. This number is further reduced today to $< 1.1\,\%$ [73].

That the Z decays to $q\bar{q}g$ means that if one of the jets get a high b-tag (and is thus likely to be a b-jet), and another one of the jets gets a low b-tag (and is thus likely to be a g-jet), then it is highly probable that the remaining jet is a b-jet. To formalize this, sort the jets after their b-tags values from high to low such that $\beta_{\text{tag}_1} > \beta_{\text{tag}_2} > \beta_{\text{tag}_3}$ for the jets $\mathbf{J} = [J_1, J_2, J_3]$ where J_i has b-tag β_{tag_i}. We then define the two tags T_b and T_g as $\mathbf{J} = [T_b, P, T_g]$ where P is the probe. If the two tags T_b and T_g passes the cuts $\beta_{b\text{-cut}} < \beta_{\text{tag}_1}$ and $\beta_{\text{tag}_3} < \beta_{g\text{-cut}}$, then the probe is selected $P = J_2$. If the probe is selected, then the last cut $\beta_{b\text{-cut}} < \beta_{\text{tag}_2}$ is the one that the efficiency is based on.

Based on Figure 5.11 we define the threshold for the b-jet tag to be $\beta_{b\text{-cut}} = 0.9$ and for the g-jet to be $\beta_{g\text{-cut}} = 0.4$. The b-signal region is thus $0.9 < \beta$ and the g-signal region $\beta < 0.4$. With these cuts, the efficiency, denoted $\varepsilon_b^{b\text{-sig}}$, of the b-jets being tagged as b-signal is computed. The TTP method is applied to both Data (Data TTP) and MC (MC TTP), along with a measurement of the efficiency when measured using MC truth (MC Truth) and a measurement based on the probes that were actual b-jets according to truth (MC Truth TTP)[14]. The efficiency is measured as a function of jet energy E_{jet} to gauge the energy dependence of the efficiency, i.e. computed in a bin-by-bin basis split according to the jet energy (of the probe).

[14] So the probes are selected by the TTP method, however, only true b-jet probes are kept.

The efficiencies can be seen in Figure 5.16. The efficiency $\varepsilon_b^{b\text{-sig}}$ as a function of jet energy E_{jet} can be seen on the left y-axis, whereas the number of probes in each bin N_{truth} can be seen on the right y-axis. The efficiencies increase as a function of energy and reaches a plateau at $E_{\text{jet}} \sim 30\,\text{GeV}$: high-energy b-jets are easier to classify than low-energy ones. Even though the efficiencies of the MC TTP and Data TTP methods are lower than the MC Truth and MC Truth TTP, the important thing to notice is that they follow each other closely, an indicator of the trained b-tagging model working equally well on both MC and Data (as hoped).

The efficiency of g-jets in the g-jet signal region $\varepsilon_g^{g\text{-sig}}$ based on the

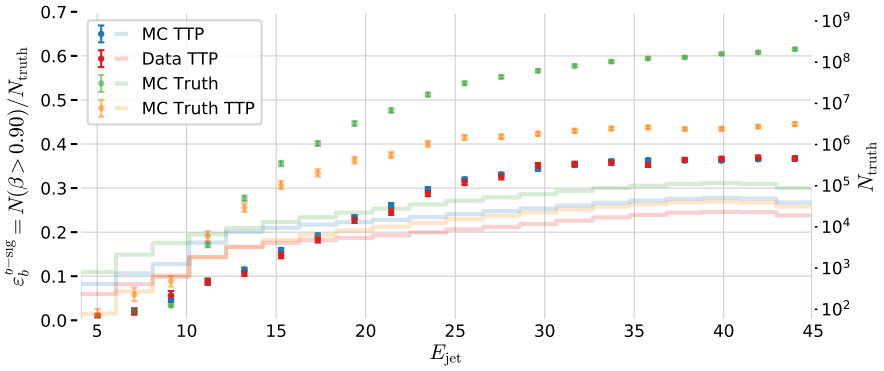


Figure 5.16: b-tag efficiency for b-jets in the b-signal region for 3-jet events, $\varepsilon_b^{b\text{-sig}}$, as a function of jet energy E_{jet}. In the plot the efficiencies are shown for MC TTP in blue, Data TTP in red, MC Truth TTP in green, and MC Truth TTP in orange. The efficiencies (the errorbars) can be read off on the left y-axis and the counts (histograms) on the right y-axis. Notice how both MC TTP and Data TTP follow each other closely.

b-tags can be measured in a similar manner. Again TTP method is used, however, now the two b-jets are the tags and the g-jet is the probe. The cuts are the same as before, however, now it is required that $0.9 < \beta_{\text{tag}_1}$ and $0.9 < \beta_{\text{tag}_2}$ before the probe is selected $P = J_3$. The efficiency is then based on $\beta_{\text{tag}_3} < 0.4 = \beta_{g\text{-cut}}$. The efficiency $\varepsilon_g^{g\text{-sig}}$ is plotted in Figure 5.17. Here the MC TTP and Data TTP also follow each other closely, this time to around $\sim 25\,\text{GeV}$.

Both of the efficiencies so far, $\varepsilon_b^{b\text{-sig}}$ and $\varepsilon_g^{g\text{-sig}}$, can be seen as signal efficiencies. Likewise, there are two background efficiencies, one for b-jets in the g-signal region $\varepsilon_b^{g\text{-sig}}$ seen in Figure B.20 and one for g-jets in the b-signal region $\varepsilon_g^{b\text{-sig}}$ seen in Figure B.21.

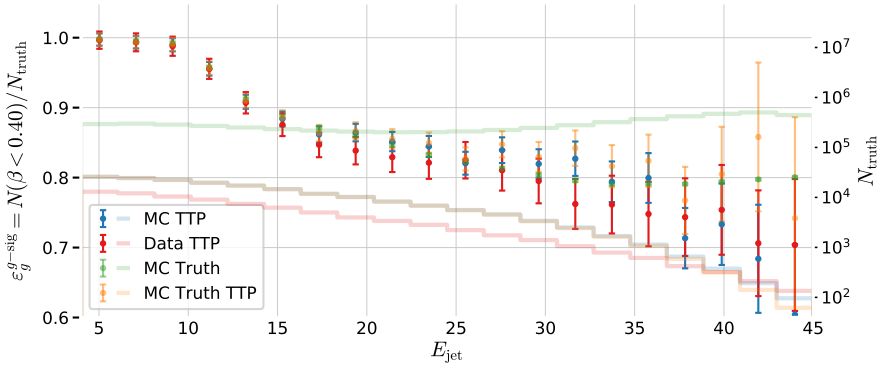


Figure 5.17: b-tag efficiency for g-jets in the g-signal region for 3-jet events, $\varepsilon_g^{g\text{-sig}}$, as a function of jet energy E_{jet}. In the plot the efficiencies are shown for MC TTP in blue, Data TTP in red, MC Truth TTP in green, and MC Truth TTP in orange. The efficiencies (the errorbars) can be read off on the left y-axis and the counts (histograms) on the right y-axis. Notice how both MC TTP and Data TTP follow each other closely until $\sim 25\,\text{GeV}$.

This section shows that the b-tagging efficiencies of the LGB b-tagging model shows comparable performance on both MC and Data which indicates that the model is un-biased (w.r.t. energy).

5.6 *g-Tagging Analysis*

The b-tagging model is a jet-based model which provides a b-tag score β_{tag} to a jet. This also means that each of the jets in e.g. a 4-jet event can get a b-tag: $\boldsymbol{\beta}_{\text{tag}} = [\beta_{\text{tag}_1}, \beta_{\text{tag}_2}, \beta_{\text{tag}_3}, \beta_{\text{tag}_4}]$. Treating $\boldsymbol{\beta}_{\text{tag}}$ as an individual observation, one can train a new model on the events based on the events compared to the individual jets. This event-based process will be called g-tagging and the trained model will return a g-tag score written as γ_{tag}.

For this model, signal events are defined to be events which are q-matched[15] and where the two jets with the highest b-tags are matched to the two (final) quark jets. Another way to say this, is that the non-q-matched jets are assigned the $n-2$ lowest b-tag scores for n-jet events. An example of a signal event is $\boldsymbol{\beta}_{\text{tag}} = [0.95, 0.89, 0.15, 0.07]$ for an event with the four jets $[b, \bar{b}, g, g]$. Compared to the b-tagging model, this model will allow one to extract entire events which contains a clear identification of gluons versus non-gluons.

[15] Remember that q-matched events are events with one, and only one, jet that is q-matched to one of the quark-jets, and one, and only one of the jets is q-matched to the other quark-jet.

5.6.1 *Permutation Invariance*

Since the b-tags are only based on the vertex variables, the goal of the g-tag is to also be constructed in an un-biased way with respect to the jet energy E_{jet}. However, even though β_{tag} is independent of E_{jet} and γ_{tag} is a function of β_{tag}, it turned out that γ_{tag} was not independent of E_{jet}. This was because the ordering of the jets within the event was energy-dependent: they are sorted according to their E_{jet}.

This meant that the different variables (b-tags) in $\boldsymbol{\beta}_{\text{tag}}$ had different feature importances when tested, even though they should be equally important. Instead of defining $\boldsymbol{\beta}_{\text{tag}}$ as a vector it should instead be seen as a set[16] $\boldsymbol{\beta}_{\text{tag}} \equiv \{\beta_{\text{tag}_1}, \ldots, \beta_{\text{tag}_n}\}$. The g-tagging model trained on the events should thus be *permutation invariant*[17] with regards to the input variables. The category of permutation invariant (and equivariant[18]) neural networks has seen an huge development within recent years in the deep learning community. The paper from Zaheer et al. [103] in 2017 was highly influential, however also other examples exists [78, 52]. Yet, the same development cannot be said to have happened within the more classic machine learning field.

[16] Since sets have no inherent order.

[17] $f(\mathbf{x}) = f(\tau(\mathbf{x}))$ for any permutation τ on an input vector $\mathbf{x}$.

[18] $\tau(f(\mathbf{x})) = f(\tau(\mathbf{x}))$ for any permutation τ on an input vector $\mathbf{x}$.

Although not being a novel software-technical solution, I circumvent the problem with two simple different approaches: 1) by simply shuffling the inputs variables independently for each observation (row) in the dataset, and 2) training on all possible permutations of the variables in the dataset. The second approach can be seen as a feature augmentation technique where the data is artificially increased with factor of n factorial: $N \to n! \cdot N$ where N is the number of events and $n \in \{3, 4\}$ is the number of jets. These two methods were tested along with keeping the original order of the dataset.

5.6.2 *Truncated Uniform PDF*

Initially when plotting the HPO performance as a function of iteration, it was seen how there were some very clear plateaus, where the highest plateau (i.e. the highest AUC value and thus the best score) was only seen in the very first iteration. It was quickly realized that this was due to the very first iteration was being run with the default values of the LGB in my HPO setup. However, what was not understood was why this value was performing so much better than the different sets of hyperparameters in the random search. Of

course LightGBM have chosen their default parameters wisely, however, one would not expect them to outperform other sets of hyperparameters that clearly. During the debugging process the column downsampling `colsample_bytree` was diagnosed to be the culprit. The default value is `colsample_bytree` $= 1$, however, the probability density function (PDF) used in random search for this parameter was $\mathcal{U}(0.4, 1)$ which was expected to give the same performance as the default value, at least for values of `colsample_bytree` close to 1. By inspecting the source code of LightGBM, I realized that if the column downsampling is less than 1 the model takes the integer of the column downsampling multiplied with the total number of columns (variables/features) [6]. This means that no matter how close to 1 the column downsampling get, the integer value of the total number of columns get floored to maximally 2 in 3-jet events, compared to when the column downsampling is exactly 1 (which it only is for the default values).

To deal with this problem I developed a new PDF[19] on top of the existing ones in Scipy: the truncated uniform PDF $\mathcal{U}_{\text{trunc}}(a, b, c)$. This PDF first generates a random number x from a uniform distribution between a and c. Then, if x is larger than b it is floored to b. In this way, it is possible to both get values of x in the interval $[a, b]$ but also values exactly equal to b. The value of c controls how often these "overflow" values of x are generated.

[19] Not strictly a PDF since it is not normalized, but otherwise behaves as one.

5.6.3 *g-Tagging Hyperparameter Optimization*

Four LightGBM models[20], two for 3-jet events and two for 4-jet events, were trained and hyperparameter optimized for both the the energy ordered and shuffled data sets with 100 iterations of random search. The same PDFs as for the b-tagging were used, see Table 5.4, and 5-fold cross validation and early stopping was applied with a patience of 100.

[20] The method with all permutations was trained using the same hyperparameters as the best ones founds in the HPO for the shuffled model to reduce time spent on HPO. The time performance is extra important for the method with all permutations as the dataset is 24 times larger the method with the shuffling for 4-jet events.

The results of the HPO can be seen in Figure 5.18. Here the two 3-jets models are seen in the two plots to the left, and the two 4-jets to the right. The very left plot shows the performance as a function of iteration number for the 3-jet energy-ordered method (no permutation or shuffling). This was where the issues mentioned in subsection 5.6.2 were first discovered. There are three very noticeable plateaus in this plot which corresponds to running column subsampling with zero, one, or two variables dropped. The three plateaus are also seen in the 3-jet events that were shuffled, however, with more variation in each plateau (along with a drop in performance). For the 4-jet events the plateaus are not as apparent but it can still be seen how some of the iteration show a significantly lower score than others. The parallel coordinate plots for the four plots can be seen in Figure B.8–B.11.

The global SHAP feature importances $\phi_{\beta_i}^{\text{tot}}$ are computed for each one of the three methods, energy-ordered, shuffled, or with all permutations, to verify their permutation invariance properties. The

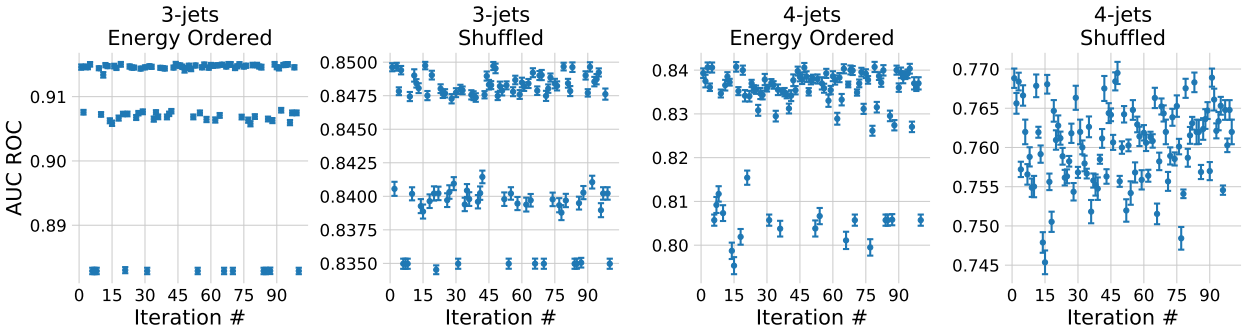


Figure 5.18: Hyperparameter Optimization results of g-tagging with 100 iterations of random search with LGB. From left to right, we have A) 3-jet events energy-ordered (no permutations), B) 3-jet events row-shuffled, C) 4-jet events energy-ordered, D) 4-jet events row-shuffled. Notice the different ranges on the y-axes.

results are seen in Table 5.5 for 4-jet events and in Table B.3 for 3-jet events. Here it can be seen that the model trained on the energy ordered data learned to attribute the highest weight to the first b-tag, second highest weight to the second b-tag, and so on. In contrary, the weights are uniformly distributed between the different b-tags in both the shuffled and all-permuted datasets (within a few sigma). The same overall pattern is seen for the 3-jet events. Based on the tables, it can be seen that both the shuffling method and all-permuting method are methods for training ML models with permutation invariant properties due to their approximately equal attribution of weight to the different variables (b-tags).

β_{tag_i}	Energy Ordered	Shuffled	All Permutations
$i = 1$	0.986 ± 0.008	0.474 ± 0.005	0.465 ± 0.005
$i = 2$	0.609 ± 0.006	0.467 ± 0.005	0.464 ± 0.005
$i = 3$	0.424 ± 0.004	0.461 ± 0.005	0.452 ± 0.005
$i = 4$	0.244 ± 0.002	0.481 ± 0.005	0.466 ± 0.005

Table 5.5: Global SHAP feature importances $\phi^{\text{tot}}_{\beta_i}$ for the three g-tagging models in 4-jet events. Each $\phi^{\text{tot}}_{\beta_i}$ is shown for the three methods in the columns and the four b-tags as variables in the rows.

5.6.4 *PermNet*

In addition to the LGB models, a permutation invariant neural network called PermNet based on the Deep Sets paper [103] implemented in Tensorflow [10] by Faye [47] was also tested. Zaheer et al. [103] showed that $f(X)$ is permutation invariant if and only if it can be decomposed in the following way:

$$f(X) = \rho\left(\sum_{x \in X} \phi(x)\right). \tag{5.6}$$

for suitable transformations ρ and ϕ (which the neural network learns[21]). The PermNet was trained using three layers[22] with leaky ReLU [69] as the activation function and ADAM [62] as the optimizer optimizing the log-loss. The network was trained with early stopping with a patience of 50 epochs and a batch size of 128. A visual overview of the PermNet architecture can be seen in Figure B.12. It took around 6 hours to fit the 3-jet events and 4.5 hours for the 4-jets (due to fewer events) for each of the models.

[21] This is possible since neural networks are universal function approximators [56].

[22] Where the two hidden layers have 128 and 64 neurons in each.

5.6.5 1D Comparison of LGB and PermNet

I made a small study to better understand the LGB and (especially) the PermNet models. This comparison was constructed by summing the b-tag scores in the n-jet event together $\sum_i^n \beta_{\text{tag}_i}$. The β_{tag_i} are summed together since this turns the problem into a 1D problem that is easy to visualize, the sum of numbers is a permutation invariant function. The sum also corresponds to the simplest functions of ρ and ϕ in equation (5.6): the identity function. Both 1D models are fit to the training events. After the fits, a linear scan from $\sum_i^n \beta_{\text{tag}_i} = 0.4$ to 3.1 is made to see how the predicted g-tags distribute. This is shown in Figure 5.19 for 4-jet events. Here the value of γ_{tag} is shown for the two models together with the fraction of signal to background in each bin. If the g-tag score should resemble a true probability it would be expected to follow the signal ratio, e.g. a model should predict $\gamma_{\text{tag}} = 0.9$ if there is 90 % signal in that bin. In the figure it is seen how the PermNet does a great job at fitting the signal fraction and the LGB model also does a decent job. Remember that none of these models were shown the signal fraction explicitly, only the b-tag sum and a truth label. The distribution of signal and background[23] together with the distribution of cuts made by the LGB model can be seen in Figure B.13. The similar plots for 3-jet events are plotted in Figure B.14 and B.15.

[23] Which the signal fraction is based on.

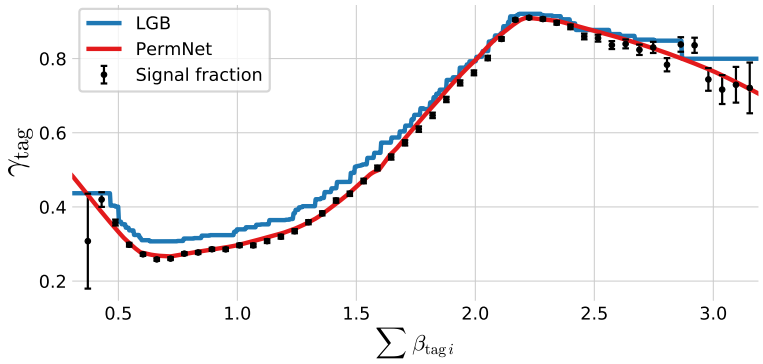


Figure 5.19: Plot of the (1D) g-tag scores for 4-jet events as a function of $\sum \beta_i$ for the LGB model in blue and the PermNet model in red. The signal fraction (based on the signal and background histograms in Figure B.13) is plotted as black error bars where the size of the error bars is based on the propagated uncertainties of the signal and background histogram assuming Poissonian statistics.

It can be concluded, at least in 1D, that both LGB and PermNet are able to capture the inherent structure in the 1D

5.6.6 g-Tagging Results

The distribution of g-tag scores in 4-jet (training) events the can be seen in Figure 5.20 for the eight combinations of the two models (LGB and PermNet) and the four data sorting methods (energy ordered, (row) shuffled, all permutations, and the 1D sum.). At first the increased number of events (a factor of 24 for 4-jet events) with the all-permutation scheme is seen separating the two light green curves from the rest. The energy ordered LGB model is the combination which utilizes most of the γ_{tag}-range, while the two 1D sum models have the most limited range, indicating that the models are more uncertain about their predictions. The energy ordered and

shuffled PermNet models can more or less only be distinguished because the latter is plotted with dashed lines. This makes sense, since they are also expected to make the same predictions were they really permutation invariant[24]. When plotted with normalized counts it is seen how the shuffled and all-permuted LGB models also follow each other very closely, which can still be partly seen in this plot by comparing the two distributions. The distribution of g-tags in 3-jet training events can be seen in Figure B.16.

[24] It is only because of the stochasticity in the optimization process of the two networks that they did not converge to exactly the same predictions.

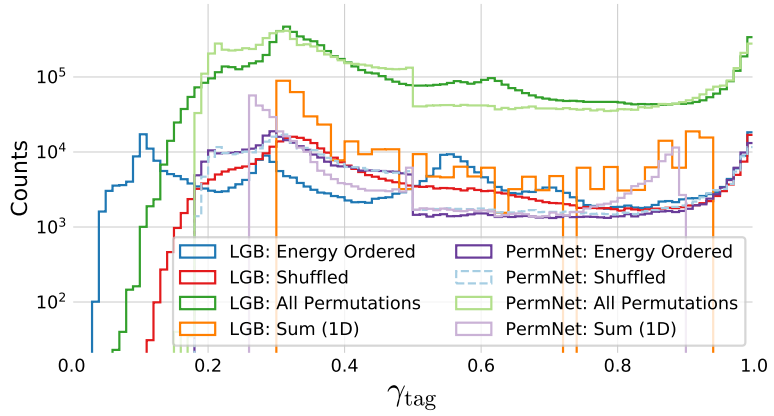


Figure 5.20: Distribution of g-tag scores in 4-jet events shown with a logarithmic y-scale for LGB: Energy Ordered in blue, LGB: Shuffled in red, LGB: All Permutations in green, LGB: Sum 1D in orange, PermNet: Energy Ordered in purple, PermNet: Shuffled in light-blue, PermNet: All Permutations in light-green, PermNet: Sum 1D in light-purple. Here LGB and PermNet are the two different type of models and "Energy Ordered", "Shuffled", "All Permutations", and "Sum 1D" are the different methods used for making the input data permutation invariant (except energy ordered).

The ROC curve in Figure 5.21 shows the performance of the different models on 4-jet events with the AUC shown in the legend. First of all it is easy to see that the energy ordered LGB model is significantly higher-performing than the rest of the models, however, this model is also energy-biased (not permutation invariant in the b-tags) and is only included to see how large a performance drop the permutation invariance criterion causes. The worst performing models are the two 1D sum models since they only have a single dimension to learn from, compared to the four dimensions that the other models have. Overall it can be seen that the rest of the models are performing almost identically, with the LGB model trained on all permutations to be the highest-performing of them all by a small margin. For 3-jet events a similar picture is seen, see Figure B.18, however, here the LGB model trained on the shuffled events performs the best, yet this performance improvement is so small compared to the all-permutations LGB model that it is expected to be due to statistical fluctuations and not a real performance difference.

Based on the AUC scores seen in the ROC curves in Figure 5.21 and B.18, the LGB-model trained on all permutations will be the g-tagging model choice. To see how this model's predictions of the g-tags distribute for signal and background events, see Figure 5.22. Here the distribution of γ_{tag} is shown for 4-jet signal events and background events. Remember that in g-tagging, the signal events are defined as events where the two jets with the highest b-tags are also the two q-matched jets (and the entire event is q-matched). In the figure it can be seen that at high values of γ_{tag} primarily $b\bar{b}gg$ events (signal b) are tagged where the jets are sorted according to their b-tags. Next after signal b events is $c\bar{c}gg$ (signal c) and $bg\bar{b}g$-events[25] (background

[25] Or any other permutation of b, $\bar{b}$, g, g which is not $b\bar{b}gg$.

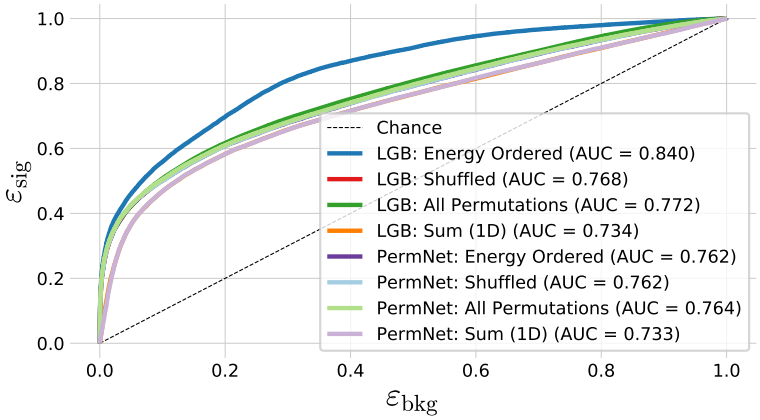


Figure 5.21: ROC curve of the eight g-tag models in 4-jet. First one in dashed black is the ROC curve that you get by random chance. The colors are the same as in Figure 5.20 and in the legend also the Area Under the ROC curve (AUC) is shown.

b). At low values of γ_{tag} light quarks (uds) dominate.

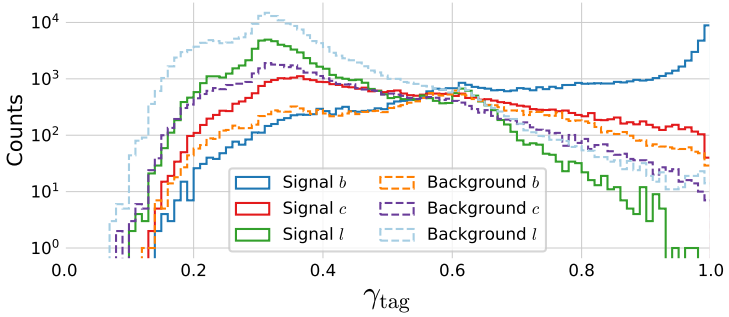


Figure 5.22: Histogram of g-tag scores from the LGB-model in 4-jet events for b signal in blue, c signal in red, l (uds) signal in green, b background in orange, c background in purple, l (uds) background in light-blue.

The similar plot for 3-jet events is seen in Figure B.19. This plot has some surprising bumps for mainly l-quark events. When comparing l-quark events in the high-γ_{tag} bump with the ones getting a low γ_{tag}-value, see Figure 5.23, one can see that l-quark events with high γ_{tag} has only two jets with high b-tags, compared to low-γ_{tag} l-quark events which more often has three jets with high b-tags.

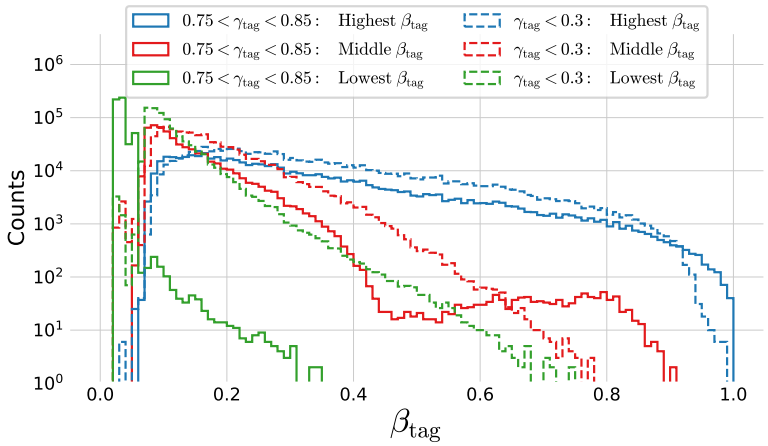


Figure 5.23: Distribution of b-Tag Scores in 3-Jet l-Quark Events for low and high g-tags values. Here l-quark events with $0.75 < \gamma_{\text{tag}} < 0.85$, so the high peak in Figure B.19, are plotted in fully connected lines, and events with $\gamma_{\text{tag}} < 0.3$ are plotted in dashed lines. For each of these two selection of events the value of the jet with the highest β_{tag} is shown in blue, the jet with the middle β_{tag} in red, and the jet with the lowest β_{tag} in green.

This is even more visible once seen in a 3D scatter plot with the lowest β_{tag} on the x-axis, the middle on the y-axis, and highest on the z-axis. Three small views from the 3D visualization can be seen in Figure 5.24. Here it is easily seen how the separating variable is the lowest b-tag: if an event where all three jets have high b-tags are

used as input to the g-tagging model it gives it a low g-tag compared to if only two of the three jets have high b-tags.

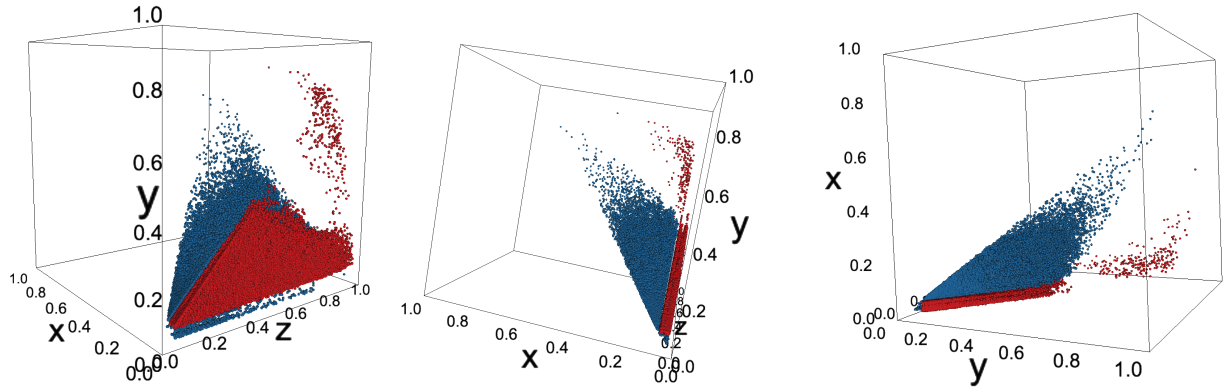


Figure 5.24: 3D scatter plot of β_{tag}-values for high and low γ_{tag} l-quark events. Here the x-axis is the lowest b-tag, the y-axis the middle, and the z-axis the highest. Here the high-γ_{tag} l-quark events are plotted in red and the low ones in blue.

5.7 *g-Tagging Efficiency*

These efficiencies are only possible to measure for MC-generated data as the truth labels are required. The Tag-Tag-Probe (TTP) method in section 5.5 is not possible for whole events as every event is completely independent of the other and thus one event cannot work as a tag for another event. We can, however, construct a pseudo g-tagging efficiency based on the b-tagging efficiencies. This efficiency will be computed by looking at 3-jet events with two jets with a high b-tag and one jet with a low b-tag, i.e. events where two jets has $0.9 < \beta_{\text{tag}}$ and one jet has $\beta_{\text{tag}} < 0.4$. This indicates a $b\bar{b}g$ event where all of the jets have been correctly identified by the b-tagging algorithm. The pseudo efficiency $\varepsilon_{b\bar{b}g}$ is then defined as:

$$\varepsilon_{b\bar{b}g} = \varepsilon_b^{b\text{-sig}}(b) \cdot \varepsilon_b^{b\text{-sig}}(\bar{b}) \cdot \varepsilon_g^{g\text{-sig}}(g)\,. \tag{5.7}$$

This is only a pseudo efficiency since this number is based on the jets in the event and not the event itself, however, by plotting it as a function of an event variable and comparing MC to Data, we can gauge the validity of the g-tagging algorithm. The first of the event variables used is the g-tag of the event γ_{tag}, see Figure 5.25. Here the pseudo efficiency is plotted as a function of γ_{tag} for Data and MC together with the counts in each bin and the ratio between $\varepsilon_{b\bar{b}g}$ for Data and MC is plotted below. At low values of γ_{tag} the uncertainties dominate due to low statistics, however, at higher γ_{tag} $\varepsilon_{b\bar{b}g}$ plateaus until very high values of γ_{tag} where it increases again. The important thing to note in this figure is the high agreement between Data and MC which converges to (almost) 1 at high γ_{tag}-values. This is an important indication of the 3-jet g-tagging model working.

Another event variable to look at is the mean of the two invariant masses[26] m_{bg} and $m_{\bar{b}g}$. The invariant mass between two quantities[27] is:

$$m_{12} = \sqrt{(E_1 + E_2)^2 - \|\mathbf{p}_1 + \mathbf{p}_2\|^2}, \tag{5.8}$$

[26] The invariant masses between all three jets $m_{b\bar{b}g}$, which otherwise might have been the first intuition to use as the event variable, is non-informative (in this context) since this is just the total event energy which is kept (approximately) constant at ~ 91 GeV.

[27] When measured in natural units.

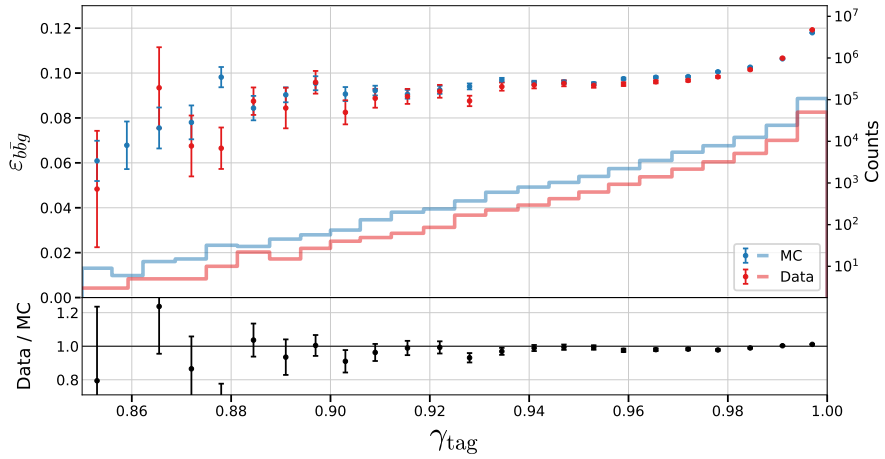


Figure 5.25: Pseudo efficiency of the g-tags for $b\bar{b}g$ 3-jet events as a function of the event's g-tag γ_{tag}. In the top plot the pseudo efficiency $\varepsilon_{b\bar{b}g}$ is shown for MC in blue and Data in red where the counts in each bin can be read on right y-axis. In the bottom plot the ratio between Data and MC is shown. The pseudo efficiency is measured by finding $b\bar{b}g$-events where $\beta_b > 0.9$, $\beta_{\bar{b}} > 0.9$, and $\beta_g < 0.4$. and then calculating $\varepsilon_{b\bar{b}g} = \varepsilon_b^{b\text{-sig}} \cdot \varepsilon_{\bar{b}}^{b\text{-sig}} \cdot \varepsilon_g^{g\text{-sig}}$.

where E_i is the energy of the i^{th} quantity and $\mathbf{p}_i$ its momentum. The pseudo efficiency is plotted as a function of the mean of m_{bg} and $m_{\bar{b}g}$ in Figure 5.26. Here the overall correspondence between Data and MC is lower than in Figure 5.25, especially for high values of the mean invariant mass. That the ratio is close to 1 for most of the data (from 20 GeV to 50 GeV) is a good sign, whereas the tails indicate the model is biased here which could be corrected for with correction factors, however, this would increase the systematic uncertainties.

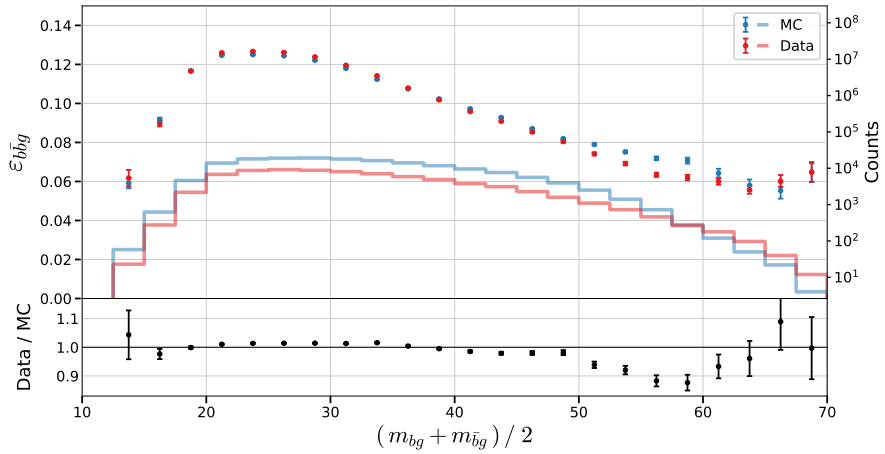


Figure 5.26: Pseudo efficiency of the g-tags for $b\bar{b}g$ 3-jet events as a function of the mean of the two invariant masses m_{bg} and $m_{\bar{b}g}$ in the event. In the top plot the pseudo efficiency $\varepsilon_{b\bar{b}g}$ is shown for MC in blue and Data in red where the counts in each bin can be read on right y-axis. In the bottom plot the ratio between Data and MC is shown.

These two figures strengthens the claim that the trained b-tagging and g-tagging models provide un-biased models that not only work in Data but also in MC.

5.8 *Generalized Angularities in 3-jet events*

To measure how gluon jet hadronizes, i.e. their jet distributions, number of tracks, etc., the *generalized angularities* provide an overall framework for doing so. The generalized angularities is a two-parameter family of variables depending on the angular weighting

$\beta \leq 0$ and an energy weighting factor $\kappa \leq 0$:

$$\lambda_\beta^\kappa = \sum_{i \in \text{jet}} z_i^\kappa \theta_i^\beta, \tag{5.9}$$

where $z_i \equiv E_i / E_{\text{jet}}$ is the momentum fraction, i.e. $0 \leq z_i \leq 1$, $\theta_i \equiv \Omega_i / R$ is the normalized angle with respect to the jet axis where R is the jet radius such that $0 \leq \theta_i \leq 1$, and i runs over all the jet constituents [51, 64]. Different values of (β, κ) probe different parts of the (gluon) jet fragmentation phase space. I will limit the analysis to the five sets of (β, κ)-values shown in Figure 5.27, where each of the sets of variables are related to the following aspects:

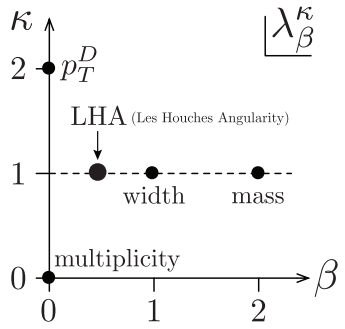


Figure 5.27: Generalized angularities. Adapted from Larkoski et al. [64].

(β, κ)

$(0, 0)$: Hadron Multiplicity.

$(0, 2)$: Transverse Momentum Distribution p_T^D: $\lambda_0^2 = \sum z_i^2 \equiv \left(p_T^D\right)^2$ [38].

$(\frac{1}{2}, 1)$: Les Houches Angularity (LHA) [89].

$(1, 1)$: Width or broadening [35].

$(2, 1)$: Mass.

We will look at the generalized angularity distributions for gluons in 3-jet events. We do so by using the g-tag from the g-tagging model to select events with a high g-tag and then select the jet in the event with the lowest of the b-tags since this jet is expected to be the gluon jet. From the plot in Figure B.19, the γ_{tag} cut off threshold is set to $\gamma_{\text{cutoff}} = 0.9$ for 3-jet events. This cut corresponds to selecting 340 476 events in MC as gluon events with a signal efficiency of $\varepsilon_g^{\text{3-jet}} = 19.68\,\%$ and a signal purity of $\rho_g^{\text{3-jet}} = 98.77\,\%$. Here a gluon event is defined as an event with a $0.9 < \gamma_{\text{tag}}$.

The distribution of λ_0^2, i.e. $(\beta, \kappa) = (0, 2)$, related to the transverse momentum distribution, in gluon events is seen in Figure 5.28. Here the distribution of λ_0^2 is shown for MC Truth (actual gluons jets using truth-label), MC selected (gluons jets selected using the g-tag) and Data (gluons jets selected using the g-tag). The generalized angularities are computed for both charged jets and neutral clusters, where this figure shows the distribution of λ_0^2 for charged jets. The MC Selected has been scaled to Data according to the fraction of the number of events in each: $w_{\text{MC}} = N_{\text{Data}} / N_{\text{MC}}$. The MC Truth has been scaled with the same weight multiplied with the gluon efficiency $w_{\text{MC-Truth}} = w_{\text{MC}} \cdot \varepsilon_g^{\text{3-jet}}$. The $\lambda_\beta^\kappa = 0$ values has been removed before plotting for all sets of values of (β, κ).

In Figure 5.28 it is seen how MC Truth and Data does not match perfectly well which indicates that this is not the perfect method, however, that MC Selected and Data distributions matches each other quite well indicates that the model is equally biased for both MC and Data.

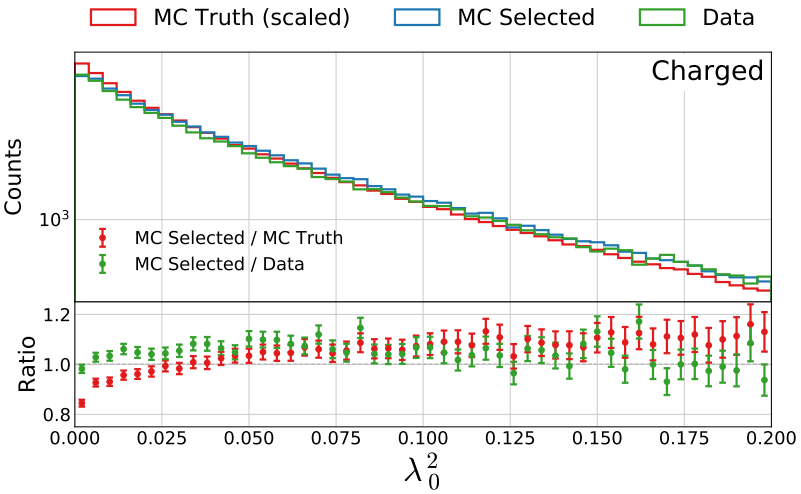


Figure 5.28: Distribution of the generalized angularity λ_0^2 for charged gluons jets in 3-jet events: λ_0^2. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

The rest of the plots can be seen in Figure B.22–B.31. For the charged tracks it can be seen that the first few bins generally has ratios just below 1 with a upwards trend until around the middle of the λ range and then it decreases again (and the uncertainties increases with lower and lower statistics). The discrepancy between MC selected and Data is larger for the neutral clusters, which are also less well-defined than the charged jets (at the energies used at LEP), and with the same trend as for the charged jets. In general the figures show a reasonable good correspondence between MC selected and Data for the generalized angularities.

5.9 *Gluon splitting*

In addition to measuring how the gluon jets hadronizes, we are also interested in measuring how they split: $g \to gg$. We do so by looking at 4-jet events with high g-tag values and then identify the two gluon jets (the ones with the lowest b-tag values). From Figure 5.22, the γ_{tag} cut off threshold is set to $\gamma_{\text{cutoff}} = 0.8$ for 4-jet events. This corresponds to selecting 41 117 events in MC as gluon events with a signal efficiency of $\varepsilon_g^{\text{4-jet}} = 23.30\,\%$ and a signal purity of $\rho_g^{\text{4-jet}} = 92.67\,\%$.

Note that the number of selected events in the 4-jet case is much lower than in the 3-jet case and that the signal purity is lower as well, $\rho_g^{\text{3-jet}} = 98.77\,\%$ vs. $\rho_g^{\text{4-jet}} = 92.67\,\%$, at the selected cut value.

5.9.1 *Variables*

The variables we use to measure the gluon splitting are each meant to probe different, smaller parts of the phase space. It is difficult to parametrize the basis of this phase space, but in discussion with one of the authors of the MC simulator Pythia [85], Peter Skands, the first dimension of the phase space will be the energy asymmetry. The energy asymmetry describes how the two gluons share their available energy between them; do they share it equally or does one of them get the majority of it? The second dimension is the resolution scale which probes the hardness of the gluons by looking at their invariant mass and the angle between the their jets. The third axis

is the azimuthal angle which measures the rotational distribution of the gluons compared to the quarks. In addition to these, some extra variables were proposed by Peter Skands [86]. These "Peter Skands"-variables are aimed at other probing other interesting areas of the phase space.

The gluon splitting variables are variables where current MC generators, such as Pythia, and Data show some discrepancies. Better measurements of these variables might lead to new theoretical insights that can reduce these discrepancies. The gluon splitting variables are:

Energy Asymmetry:

E_{diff}: The relative difference in energy between the gluon jet with the highest and lowest energy: $E_{\text{diff}} = \frac{E_{\text{max}} - E_{\text{min}}}{E_{\text{max}} + E_{\text{min}}}$.

$E_{\text{rel}_{\text{min}}}$: The relative energy of the gluon jet with the lowest energy and the sum: $E_{\text{rel}_{\text{min}}} = \frac{E_{\text{min}}}{E_{\text{max}} + E_{\text{min}}}$.

E_{rel}: The relative energy of the gluon jet with the lowest energy and the highest energy: $E_{\text{rel}} = \frac{E_{\text{min}}}{E_{\text{max}}}$.

Resolution Scale:

Δ_θ: The angle between the two gluon jets.

m_{gg}: The invariant mass of the two gluon jets.

Azimuthal Angle

$\phi_\parallel$: The angle between the plane spanned by the two b-jets and the plane spanned by the two gluon jets. This is the same angle as the angle between the b-jet cross product $\mathbf{p}_{b_1} \times \mathbf{p}_{b_2}$ and the gluon-jet cross product $\mathbf{p}_{g_1} \times \mathbf{p}_{g_2}$ where $\mathbf{p}$ is the jet momentum.

Peter Skands:

$\ln\left(k_t^2 / m_{\text{vis}}^2\right)$: Logarithm of the ratio between the k_t value, see equation (5.12), of the two gluon jets and the visible mass of the event.

$p_{\perp,\text{A}}^2$: The p_perp antenna defined as:

$$p_{\perp,\text{A}}^2 = \widetilde{m}_{12}^2 \cdot \min\left(\frac{\min\left(\widetilde{m}_{b1}^2, \widetilde{m}_{b2}^2\right) - m_b^2}{\widetilde{m}_{b12}^2 - m_b^2}, \frac{\min\left(\widetilde{m}_{\bar{b}1}^2, \widetilde{m}_{\bar{b}2}^2\right) - m_b^2}{\widetilde{m}_{\bar{b}12}^2 - m_b^2}\right), \tag{5.10}$$

where $\widetilde{m}^2 = m^2 cdot m_Z^2 / m_{\text{vis}}^2$ and m_Z is the mass of the Z boson.

The gluon splitting variables will be analyzed in distinct areas defined by the k_t [34, 45] and Cambridge/Aachen (CA) [43, 102] jet clustering algorithms for e^+e^- collisions which will first be described. The two algorithms uses the following[28] *jet distance measure*:

[28] When using $R = 1$ in eq. (9a) in Ref. [33].

$$d_{ij}^2(p) = \min\left(E_i^{2p}, E_j^{2p}\right)\left(\frac{1-\cos\theta_{ij}}{2}\right), \tag{5.11}$$

where E is the (pseudo)jet energy and θ_{ij} is the angle between (pseudo)jet i and j [33]. For $p = 1$ equation (5.11) is called the k_t algorithm and the Cambridge/Aachen for $p = 0$. Both the k_t and CA algorithms are newer jet clustering algorithms than JADE, see section 4.4, and their distance measures are also pretty similar to JADE's. The value $d_{ij}(p = 1)$ is also known as the k_t:

$$k_t = \sqrt{\min\left(E_i^2, E_j^2\right)\left(\frac{1-\cos\theta_{ij}}{2}\right)}. \tag{5.12}$$

Based on the two algorithms, we define the ratio R_{gg} between the two gluon jets d_{gg}^2 and the lowest value of d_{ij}^2 not including the two gluon jets:

$$R_{gg}(p) \equiv \frac{d_{gg}^2(p)}{\min_{(i,j)\neq(g,g)} d_{ij}^2(p)}. \tag{5.13}$$

We further define $R_{gg}^{k_t} \equiv R_{gg}(p = 1)$ and $R_{gg}^{CA} \equiv R_{gg}(p = 0)$.

Since the CA algorithm is energy-independent and the k_t algorithm is not, they describe different parts of the phase space. One example of this is the case of soft wide gluon jets illustrated in Figure 5.29. Here the two gluon jets are low-energy (soft) jets but with a high angle between them (wide). In this case the k_t algorithm would probably cluster the two gluon jets together but the CA algorithm would not. This means that $R_{gg}^{k_t}$ would be less than 1 since the distance measure between the two gluon jets would be the smallest of them all, whereas to R_{gg}^{CA} would be larger than 1 since $d_{\bar{b}g} < d_{gg}$. In addition to the soft, wide angle gluon jets, we have the case of soft, collinear gluon jets which the two algorithms agree to cluster together, see Figure 5.30 or the opposite case where the two algorithms both agree on not to cluster the gluon jets together, see Figure 5.31. These figures are just illustrations of how to better understand the R_{gg} phase space.

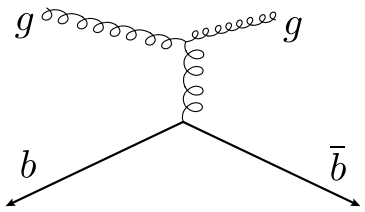


Figure 5.29: Soft, wide angle gluons in 4-jet events.

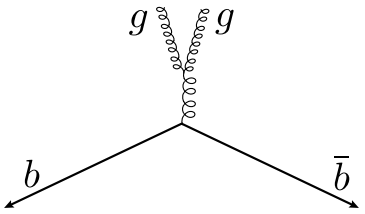


Figure 5.30: Soft, collinear gluons in 4-jet events.

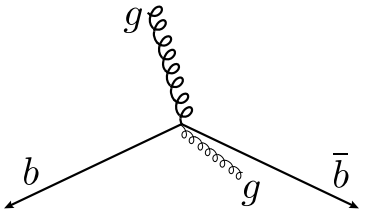


Figure 5.31: Hard, non $g \to gg$ gluons in 4-jet events.

The variables are computed for each event, and their relationship with γ_{tag} is shown in Figure B.32–B.39. From now on primarily events with with "good" g-tags, $\gamma_{\text{tag}} > 0.8$, are used. The efficiency of these signal events are measured in the following subsection.

5.9.2 *Efficiencies*

It is not possible to take advantage of the Tag-Tag-Probe method for whole events as it was for individual jets in the 3-jet case. As such, we are also unable to estimate the g-tagging efficiency in Data of the gluon splitting variables defined in the previous subsection. However, it is still possible to do so for MC using the truth labels. The efficiency $\varepsilon_{gg}^{\text{4-jet}}$ for the gluon splitting variable E_{diff} is shown in Figure 5.32 for signal and background based on MC Truth. Note that even though the efficiency here is quite low, it is close to constant (as a function of E_{diff}).

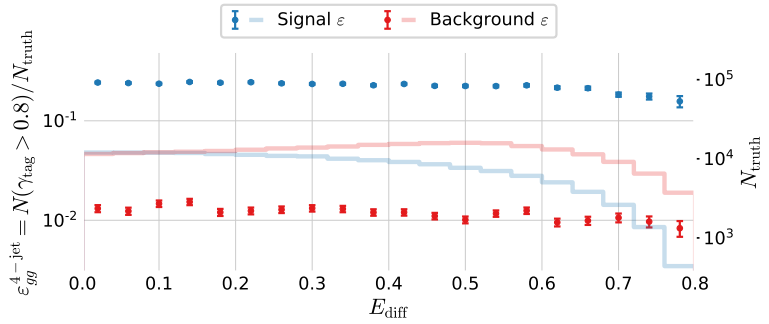


Figure 5.32: Efficiency of the g-tagging algorithm for 4-jet events as a function of normalized gluon-gluon jet energy difference (asymmetry) E_{diff} in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

The plot for the rest of the g-tagging efficiencies can be seen in Figure B.40–B.49.

5.9.3 *Closure Test*

I perform a closure test to validate the g-tagging model for the gluon variables in 4-jet events. Closure tests compare the developed method after corrections[29] to MC Truth. Any discrepancies can then be investigated and finally the closure test can gauge the systematic uncertainties of the analysis.

The closure test is based on the distribution of the gluon splitting variables, see subsection 5.9.1, for events in the signal region or the so-called *sideband region*. The sideband region[30] is a region close to the signal region where the background is expected to behave approximately similar to the background in the signal region (and likewise for the signal). From Figure 5.22, the signal region was defined to be $0.8 < \gamma_{\text{tag}}$ for 4-jet events. This region is expected to contain primarily signal events. For 4-jet events, we define the sideband region to be $0.6 < \gamma_{\text{tag}} < 0.8$.

The aim is to fully reconstruct the true distribution $\mathcal{P}_{gg}(x)$ of any gluon splitting variable x. To first order, this is given by $\mathcal{P}_{gg} \approx \mathcal{P}_{\text{sig}}/\varepsilon_{gg}^{\text{4-jet}}$, where $\mathcal{P}_{\text{sig}}(x)$ is the distribution of x for all events in the signal region. However, this expression completely ignores the background events that are also found in the signal region. To correct for the assumption of no background, we introduce $\mathcal{P}_{\text{bkg}}(x)$ which is the distribution of x for background events:

$$\mathcal{P}_{gg} = \frac{\mathcal{P}_{\text{sig}} - \alpha \cdot \mathcal{P}_{\text{bkg}}}{\varepsilon_{gg}^{\text{4-jet}}}. \tag{5.14}$$

Here α is the fraction of background events in the signal region $N_{\text{bkg}}^{\text{sig}}$ relative to the background events in the sideband $N_{\text{bkg}}^{\text{side}}$. Letting f_{gg}^{sig} denote the fraction of signal in the signal region and $f_{\text{bkg}}^{\text{side}}$ the fraction of background in the sideband region, α is defined as:

$$\alpha = \frac{N_{\text{bkg}}^{\text{sig}}}{N_{\text{bkg}}^{\text{side}}} = \frac{\left(1 - f_{gg}^{\text{sig}}\right) \cdot N_{\text{sig}}}{f_{\text{bkg}}^{\text{side}} \cdot N_{\text{side}}}, \tag{5.15}$$

where N_i is the number of events in region i (either signal or sideband). The background distribution $\mathcal{P}_{\text{bkg}}(x)$ itself can be approxi-

[29] E.g. Applying the efficiencies found in the previous subsection.

[30] Also sometimes known as the control region.

mated to be $\mathcal{P}_{\text{bkg}} \approx \mathcal{P}_{\text{side}}$ if assuming no signal events in the sideband region, yet this assumption is also not satisfied and is thus corrected for:

$$\mathcal{P}_{\text{bkg}} = \mathcal{P}_{\text{side}} - \beta \cdot \mathcal{P}_{gg} \cdot \varepsilon_{gg}^{\text{4-jet}}, \tag{5.16}$$

where β is the fraction of signal events in the sideband region relative to the signal events in the signal region and is defined as:

$$\beta = \frac{\left(1 - f_{\text{bkg}}^{\text{side}}\right) \cdot N_{\text{side}}}{f_{gg}^{\text{sig}} \cdot N_{\text{sig}}}. \tag{5.17}$$

Plugging equation (5.16) into (5.14) and solving for $\mathcal{P}_{gg}$ yields:

$$\mathcal{P}_{gg} = \frac{\mathcal{P}_{\text{sig}} - \alpha \cdot \mathcal{P}_{\text{side}}}{\varepsilon_{gg}^{\text{4-jet}} \cdot (1 + \alpha\beta)}. \tag{5.18}$$

The advantage of this equation is that only $\varepsilon_{gg}^{\text{4-jet}}$ and the two constants[31] f_{gg}^{sig} and $f_{\text{bkg}}^{\text{side}}$ depend on MC truth, the rest can be applied to data without any truth label. It should be made clear that the final expression, equation (5.18), is based on the assumption that the signal distribution of x is (approximately) similar in the signal and sideband regions $\mathcal{P}_{gg}^{\text{sig}} \approx \mathcal{P}_{gg}^{\text{side}}$ and likewise for the background distributions $\mathcal{P}_{\text{bkg}}^{\text{sig}} \approx \mathcal{P}_{\text{bkg}}^{\text{side}}$.

[31] Which are found to be: $f_{gg}^{\text{sig}} = \rho_g^{\text{4-jet}} = 92.67\,\%$ and $f_{\text{bkg}}^{\text{side}} = 62.5\,\%$.

The distribution of $\mathcal{P}_{gg}$ in MC is shown in Figure 5.33 together with $\mathcal{P}_{\text{sig}}$, $\mathcal{P}_{\text{side}}$, and the distribution for MC Truth $\mathcal{P}_{gg}^{\text{Truth}}$. In this figure the distributions are shown for the gluon splitting variable E_{diff} with histograms shown in the top plot and a ratio plot between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ in the bottom part. By just looking at the distributions, it can be seen that the distributions in the signal and sideband regions follow each quite closely even though they start to differ at large values of E_{diff}. Similarly, also $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ follow each other quite closely, however, by looking at the ratio plot it can be seen that $\mathcal{P}_{gg}$ generally has fewer counts in each bin than $\mathcal{P}_{gg}^{\text{Truth}}$.

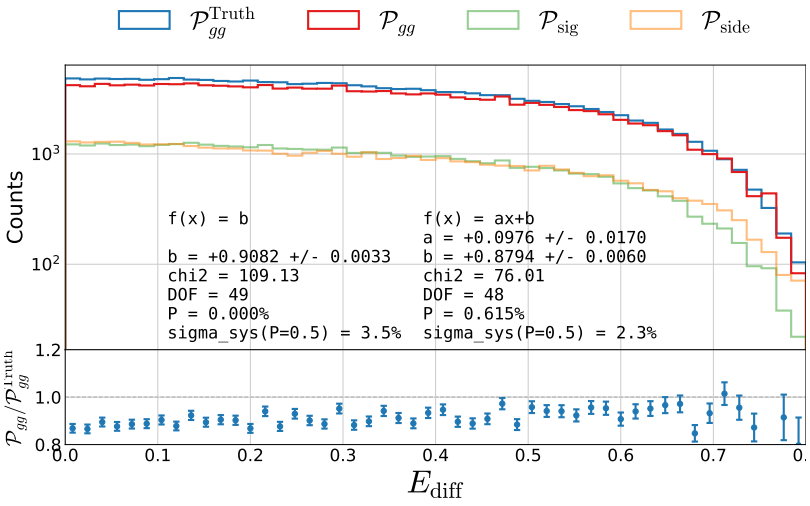


Figure 5.33: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for the normalized gluon gluon jet energy asymmetry E_{diff}. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

The errorbars in the ratio plot is fitted with both a constant $f(x) = b$ and a straight line $f(x) = ax + b$ with the fit results shown as

text in the plot. In the text box `DOF` is the number of degrees of freedom and `P` is the χ^2-probability[32]. I compute the systematic error σ_{sys} that would have to be added in quadrature to the standard deviation, $\sigma \to \sqrt{\sigma^2 + \sigma_{\text{sys}}^2}$, such that the χ^2-probability would be 50 %. I do this using Brent's method [27] and the systematic error is written as `sigma_sys` in the figure.

[32] $P(\chi^2; N_{\text{DOF}}) = \int_{\chi^2}^{\infty} f_{\chi^2}(x; N_{\text{DOF}})\, dx$, where $f_{\chi^2}(x; N_{\text{DOF}})$ is the χ^2 distribution with N_{DOF} degrees of freedom.

The closure plot for the rest of the gluon splitting variables in Figure B.50–B.57. In general it can be seen that the $\mathcal{P}_{gg}$ matches $\mathcal{P}_{gg}^{\text{Truth}}$ pretty well for all the gluon splitting variables, however, with a 10 % offset. This offset is easy to correct for with a constant correction factor. However, we see that for the energy asymmetry variables and Δ_θ there are clear, linear trends in the ratio plots. This is seen e.g. for E_{diff} in Figure 5.33 where the χ^2 value for the constant fit is 109.13 but only 76.01 for the linear fit. This difference in χ^2 value cannot be attributed to just the linear fit being a more advanced fit in which case the difference would be expected to be $\Delta\chi^2 \sim 1$ due to difference in the number of fit parameters being 1. For these four variables the systematic uncertainty will thus be based on the linear fit, whereas it will be based on the constant for the rest of the variables.

The g-tagging algorithm perform well in general where most of the systematic attributed are 2 % to 3 %. The highest systematic uncertainty is for the invariant mass of the two gluons: $\sigma_{\text{sys}}(m_{gg}) = 3.6\,\%$. A summary of all the systematic errors from the closure tests can be seen in Table 5.6 for all of the gluon splitting variables.

The closure test shows that the g-tagging algorithm can also be trusted in the 4-jet case, however, with high systematic uncertainties for the m_{gg} variable.

	σ_{sys}	$f(x)$
E_{diff}	2.3 %	$ax + b$
$E_{\text{rel}_{\text{min}}}$	2.3 %	$ax + b$
E_{rel}	2.5 %	$ax + b$
Δ_θ	2.0 %	$ax + b$
m_{gg}	3.6 %	b
$\phi_{\parallel}$	1.7 %	b
$\ln\left(k_t^2/m_{\text{vis}}^2\right)$	2.1 %	b
$p_{\perp,\text{A}}^2$	3.1 %	b

Table 5.6: Systematic errors for the gluon splitting variables based on the closure test, see subsection 5.9.3. The last column, $f(x)$, denotes which fit the systematic error is based on.

5.9.4 4-jet results

The comparison between the gluon splitting distributions will be done in four distinct areas of the $R_{gg}^{k_t}$-R_{gg}^{CA} phase space. These four areas, A, B, C, and D, are defined in Table 5.7. Three of these regions have already been described, that is area A which is the soft, collinear region with an example shown in Figure 5.30, area C which is the soft, wide angle region in Figure 5.29, and the hard non-$g \to gg$ area D in Figure 5.31. Area B is a region where neither the k_t algorithm nor the CA algorithm is totally sure whether or not the two gluons should be clustered together. The four areas are visualized in Figure 5.34 for both MC and Data. Here each dot shown in the scatter plot is an event with $0.8 < \gamma_{\text{tag}}$, with a total of 41 117 events in the MC sample and 22 473 events in the data sample. The number of events that fall into each of the four areas are shown in the figure. The events are split up into these four areas because they each concern different physical interactions. The number of events in each area were taken into account when creating the ranges in Table 5.7 to minimize the statistical uncertainties.

Region	$R_{gg}^{k_t}$	R_{gg}^{CA}
A	$[0, 1]$	$[0, 1]$
B	$[\frac{3}{5}, \frac{5}{3}]$	$[\frac{3}{5}, \frac{5}{3}]$
C	$[0, \frac{5}{3}]$	$[1, \infty]$
D	$[\frac{5}{3}, \infty]$	$[\frac{5}{3}, \infty]$

Table 5.7: Definitions of the four areas in the $R_{gg}^{k_t}$-R_{gg}^{CA} phase space.

The distributions for the different gluons splitting variables for area A, the soft, collinear region, is shown in Figure 5.35 for both

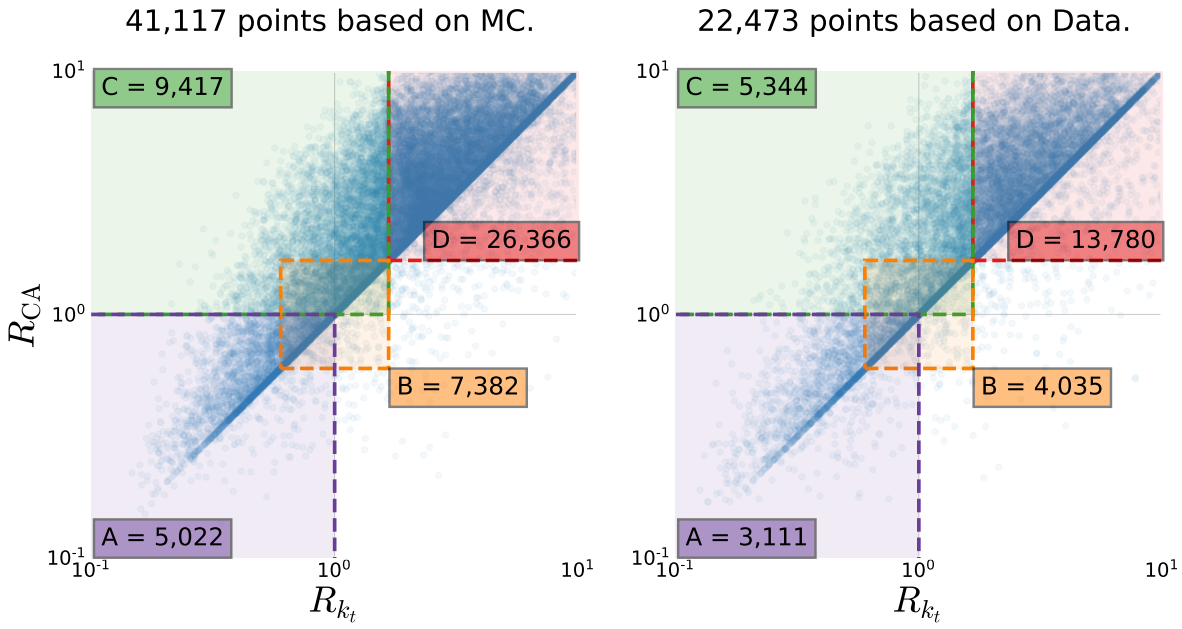


Figure 5.34: Overview of the four areas, A, B, C, and D, in the $R_{gg}^{k_t}$-R_{gg}^{CA} phase space. The areas are shown as colored rectangles together with a scatter plot showing the 2D-distribution for signal gluon events ($0.8 < \gamma_{\mathrm{tag}}$). The left plot shows is for MC and the right one for Data.

MC (scaled to Data) and Data. Area A contains 5022 events in the MC sample and 3111 in the Data sample. Generally the distributions match pretty well between the MC and Data, however, there seems to small discrepancies in the energy asymmetry variables, however, this might just be due to statistical fluctuations (notice the low bin count in each bin). When looking at the other areas, see Figure B.58–B.61, the energy discrepancies seem to disappear. In general there seems to be a reasonably good correspondence between the distributions for MC and Data.

5.10 Discussion

The distributions of both the generalized angularities and the gluon splitting variables are based on the *g*-tagging model which itself is based on the *b*-tagging model. At first this way of training two different ML models, where one is trained on the other[33], might seem a bit counterintuitive: why not just build a single, combined model that is able to extract gluons? The reason why the method used here is set up the way it is, is to be able to better understand the intermediate steps while also being able to compare the results to others. With the *b*-tagging model we are able to compare our model to the neural network trained by ALEPH (NNB) and measure the *b*-tagging efficiency for both gluons and *b*-quarks individually. Neither of these would not have been possible with a combined model which just outputs single gluons.

Regarding the NNB model, it is interesting to compare the *b*-tagging ROC curves in Figure 5.10 for 4-jets and B.5 for 3-jets. Even thought the performance for the old NNB model is comparable to our model[34], especially for 3-jet events, one has to take into account that the NNB model is trained on nine variables compared to only

[33] This is very different to the ensemble model used in section 3.8 which can basically be treated as a single, advanced ML model.

[34] Technically models, both the XGBoost model and the LightGBM model.

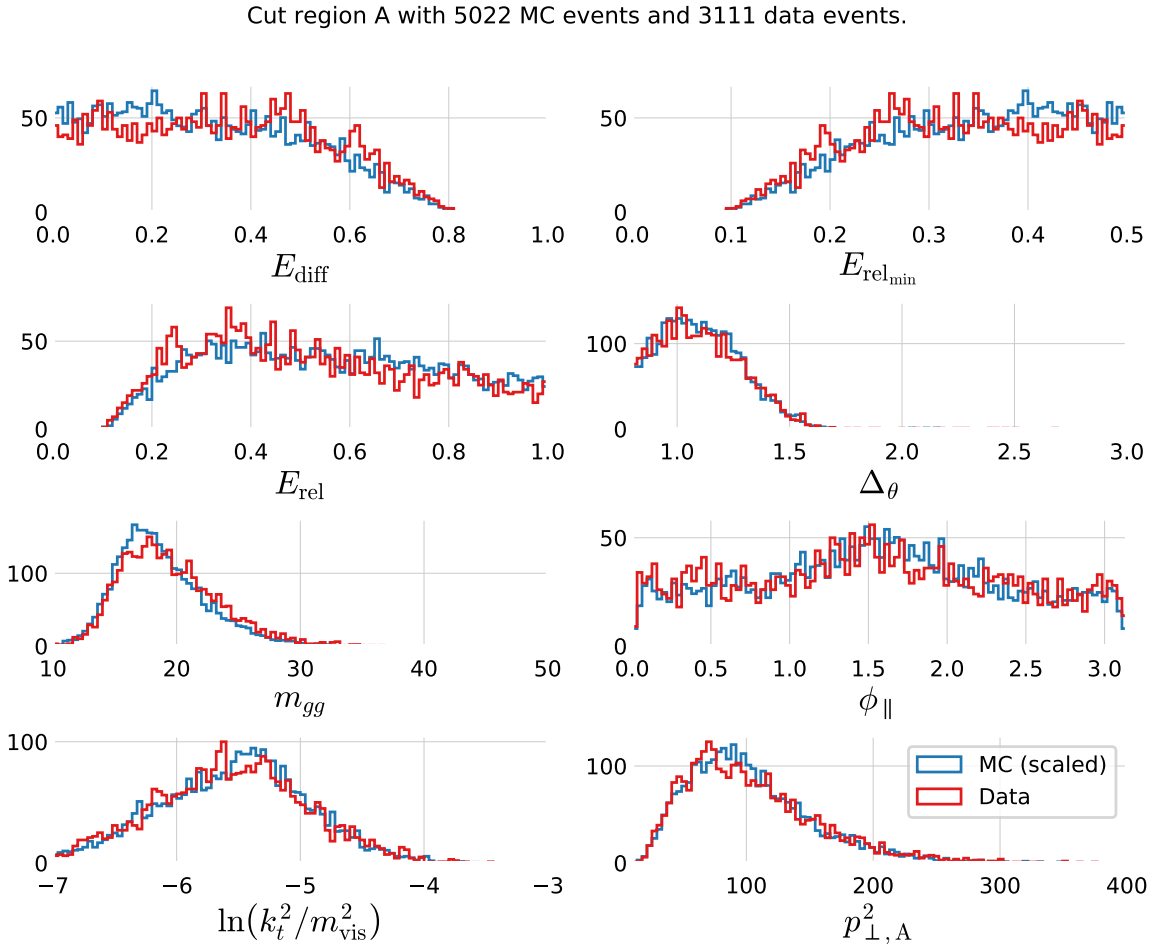


Figure 5.35: Comparison of the gluon splitting distributions in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area A, see Table 5.7. The distribution for MC (scaled to Data) is shown in blue and for Data in red. These eight distributions are for the $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area A which has 5022 events in the MC sample and 3111 in the Data sample.

three variables by our model. Yet, remember that this is a model that was trained more than 25 years ago by now. I doubt many other areas of science implemented machine models that long ago with a performance that is almost comparable to today's.

That we only train on the vertex variables is to reduce any potential bias that might be introduced when training on the shape variables (in addition to the vertex variables) and then afterwards used to compare the same shape variables between MC and Data afterwards. Another approach to this would be to train on all variables and then afterwards quantify the potential bias and correct for it. This would also allow one to compare the loss in performance by only using the vertex variables. This was not done in this project due to time limitations.

As mentioned above, with the modularized approach we apply in this project we are able to measure the *b*-tagging efficiencies not only in MC but also in Data. The efficiencies were measured using a tag-tag-probe method in 3-jet events. In 4-jet events this problem is a lot harder, however, it might have been possible to apply a tag-tag-tag-probe method, although the results of this are very likely to be limited by large uncertainties due to much lower statistics in 4-jet

events compared to 3-jet events.

5.11 Conclusion

blabla

A. Housing Prices Appendix

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

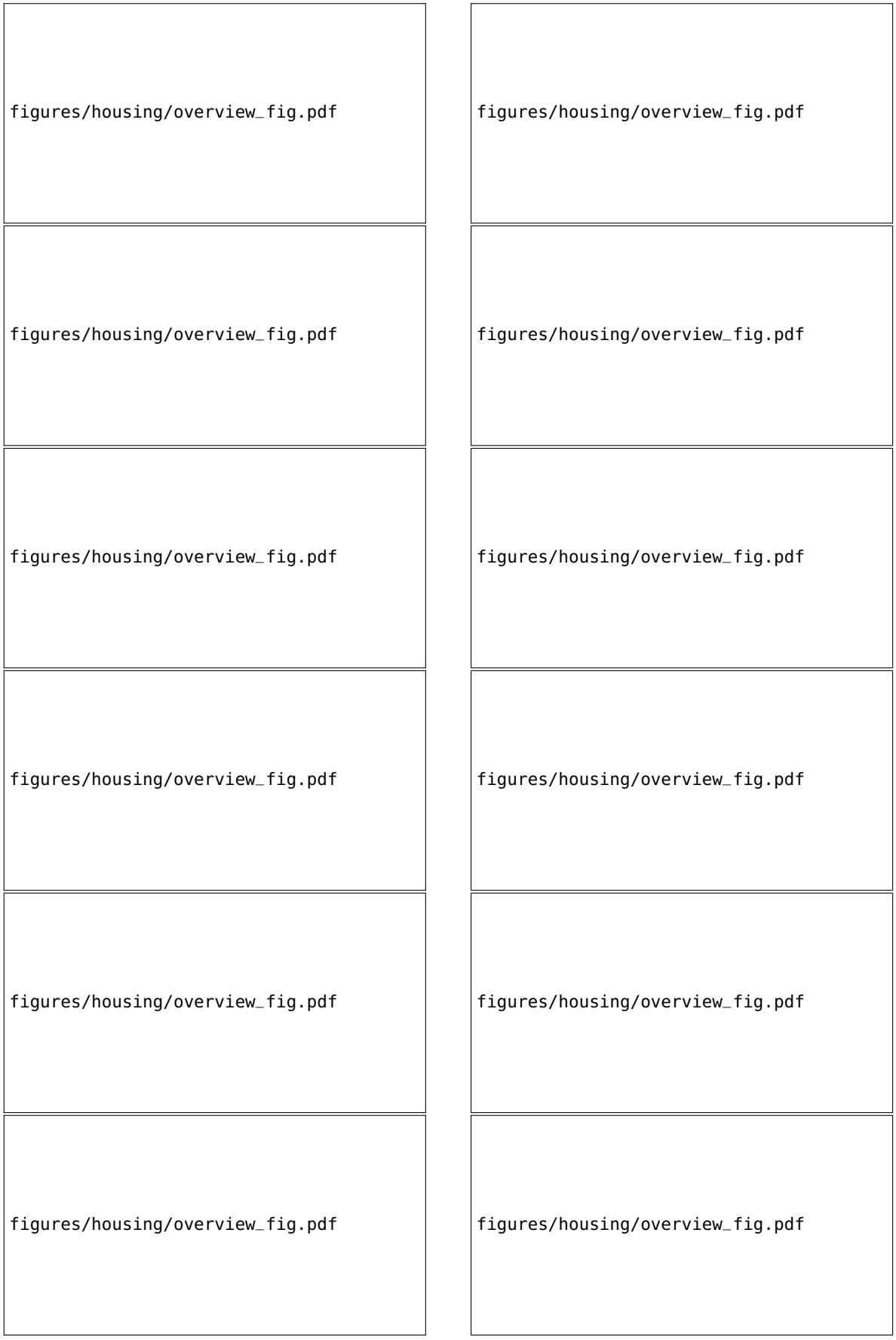


Figure A.2: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

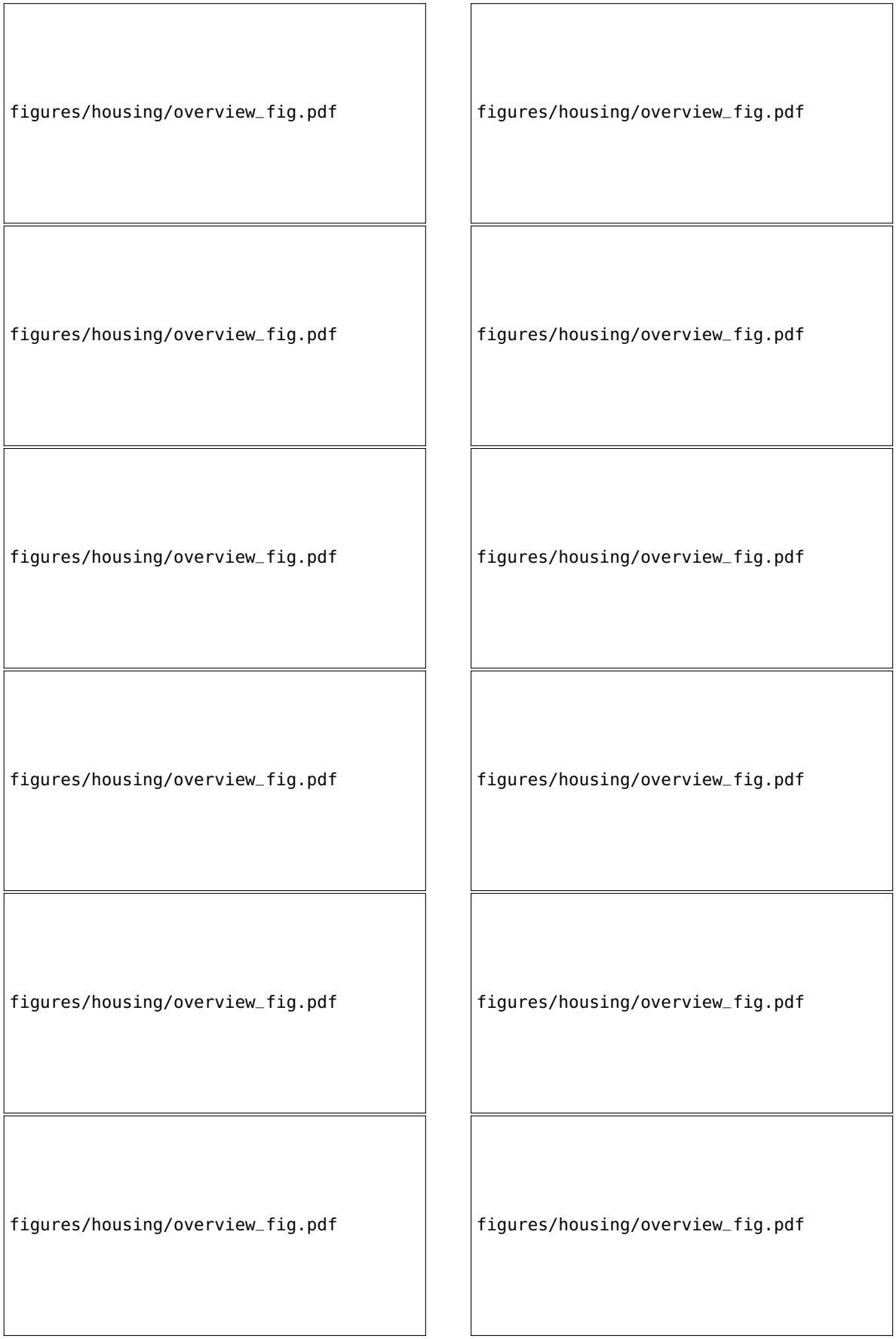


Figure A.3: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

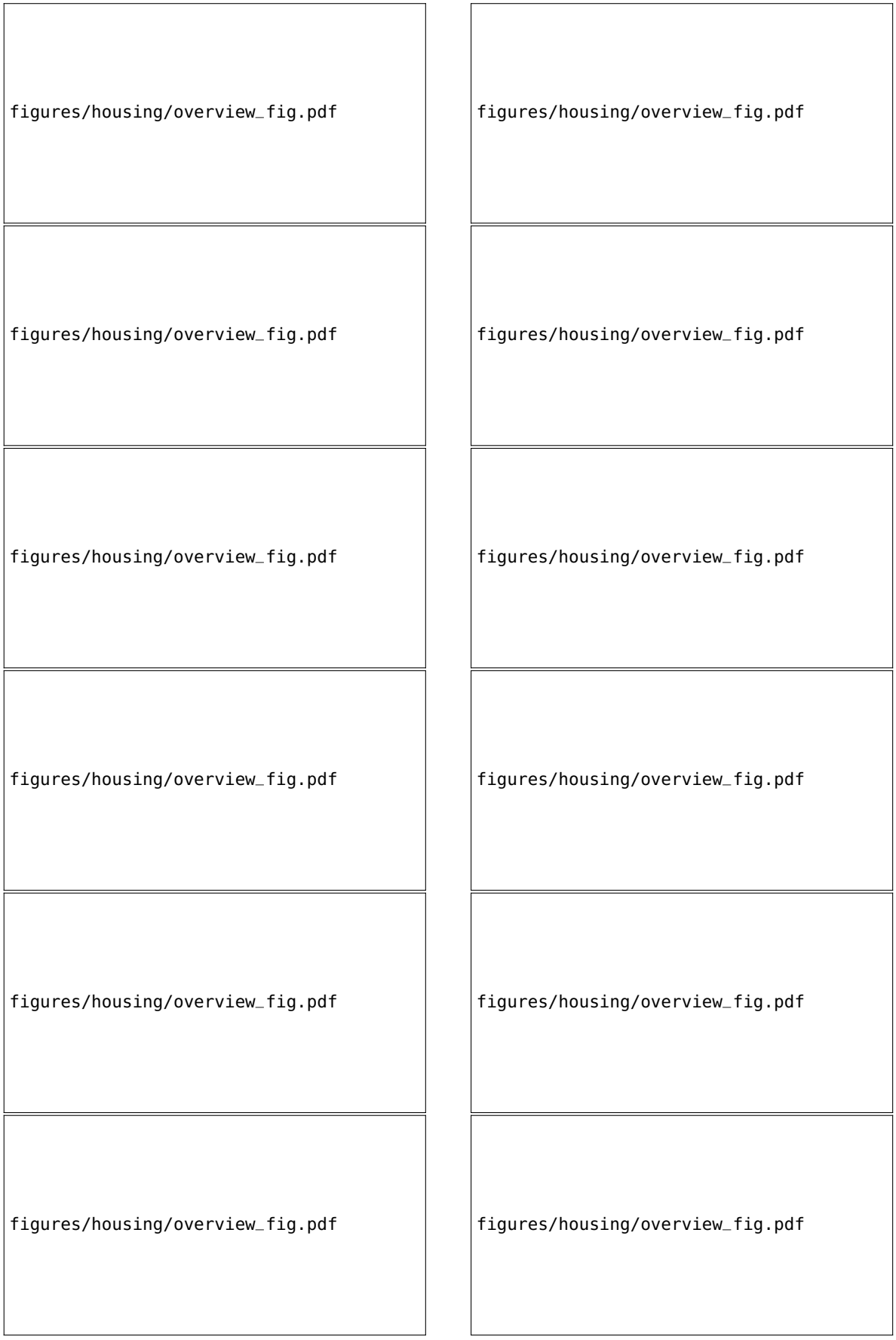


Figure A.4: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

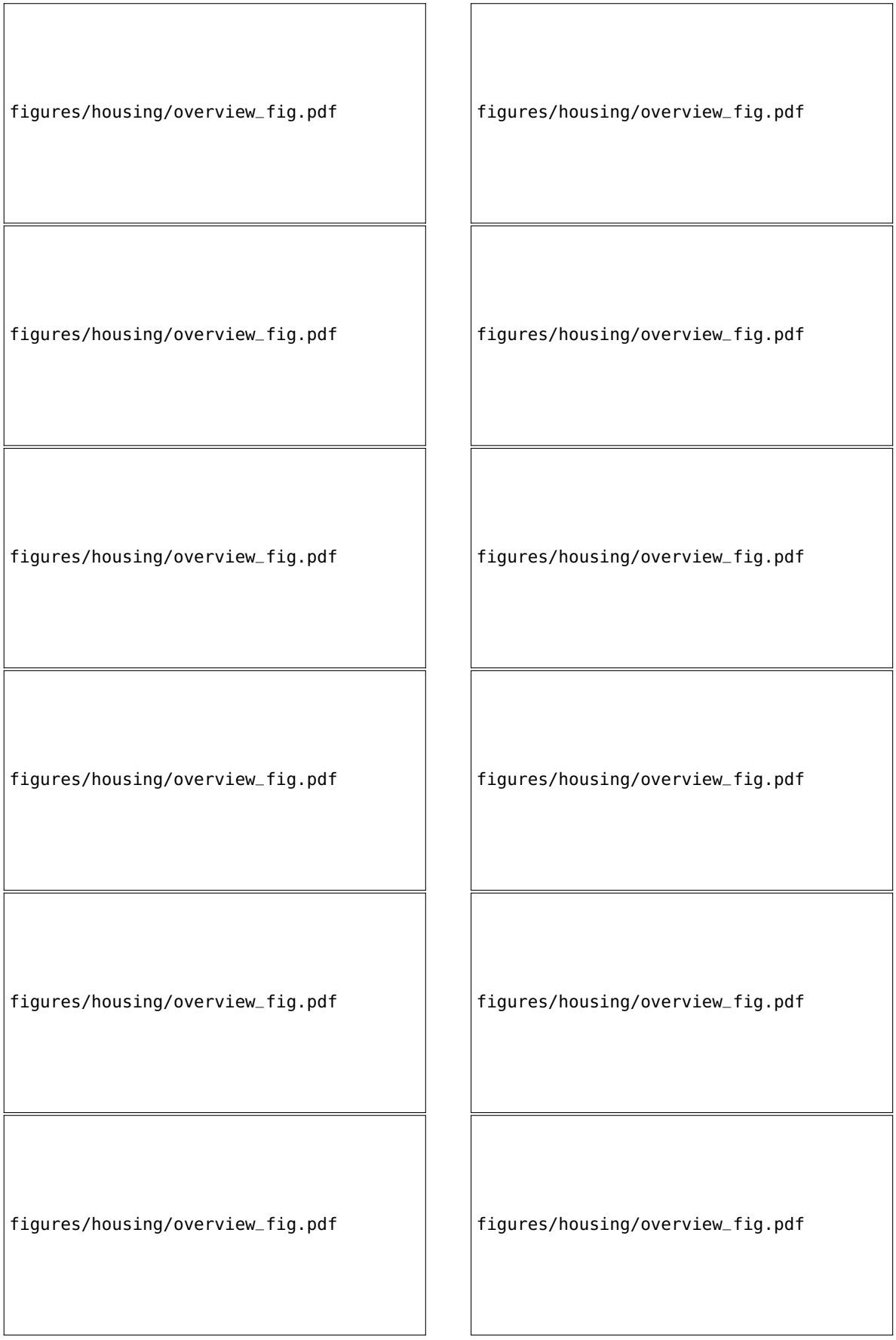


Figure A.5: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

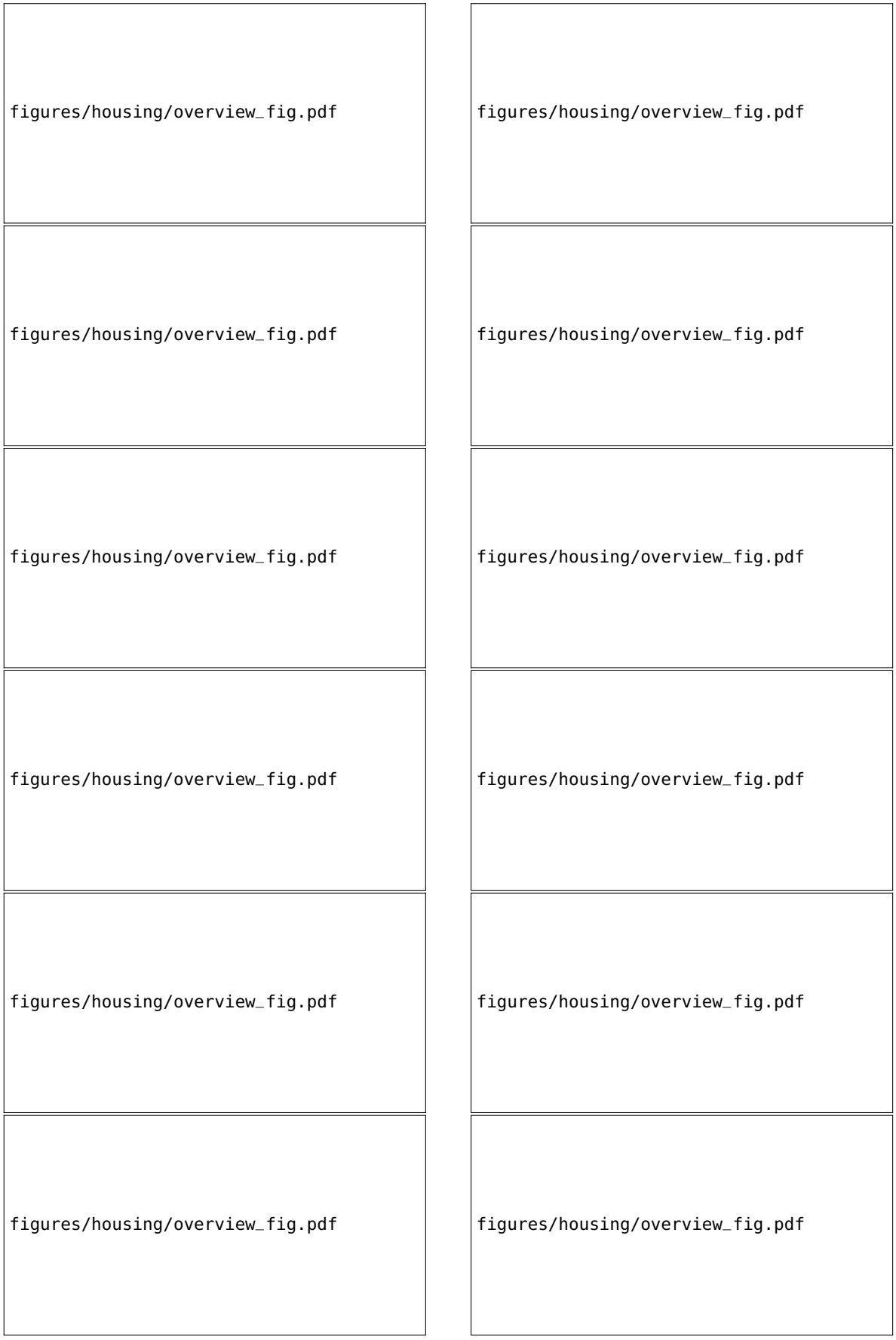


Figure A.6: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

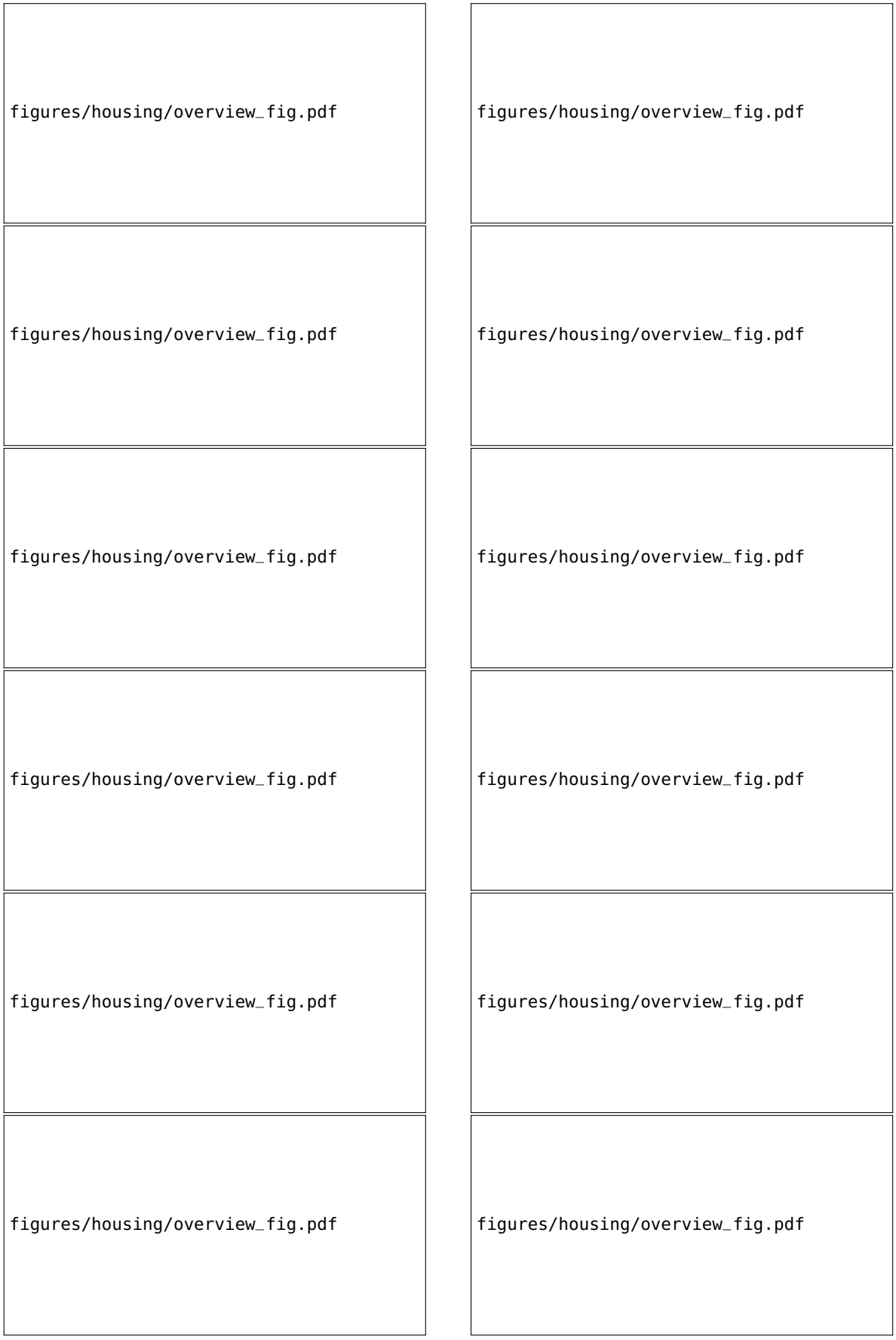


Figure A.7: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

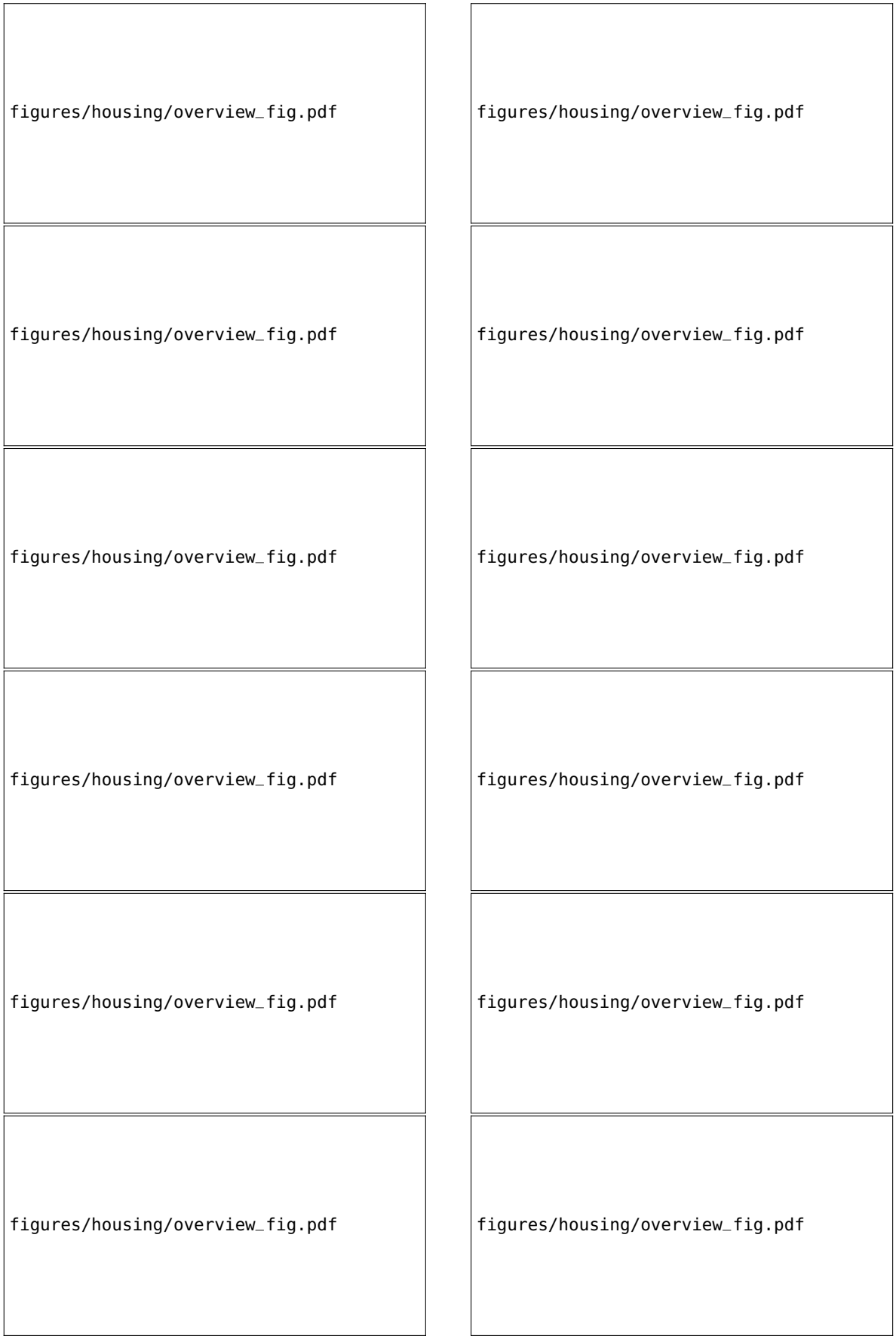


Figure A.8: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

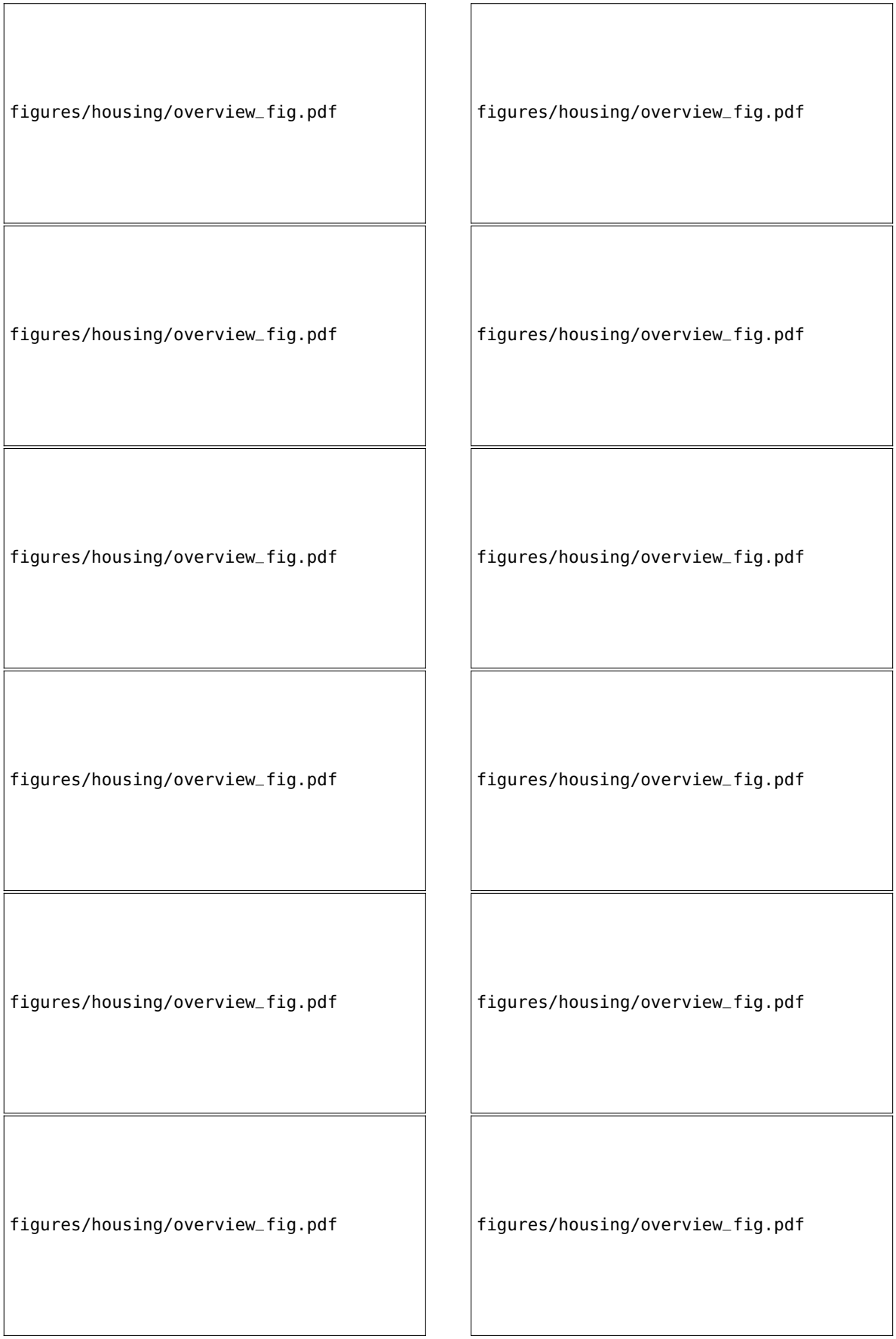


Figure A.9: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

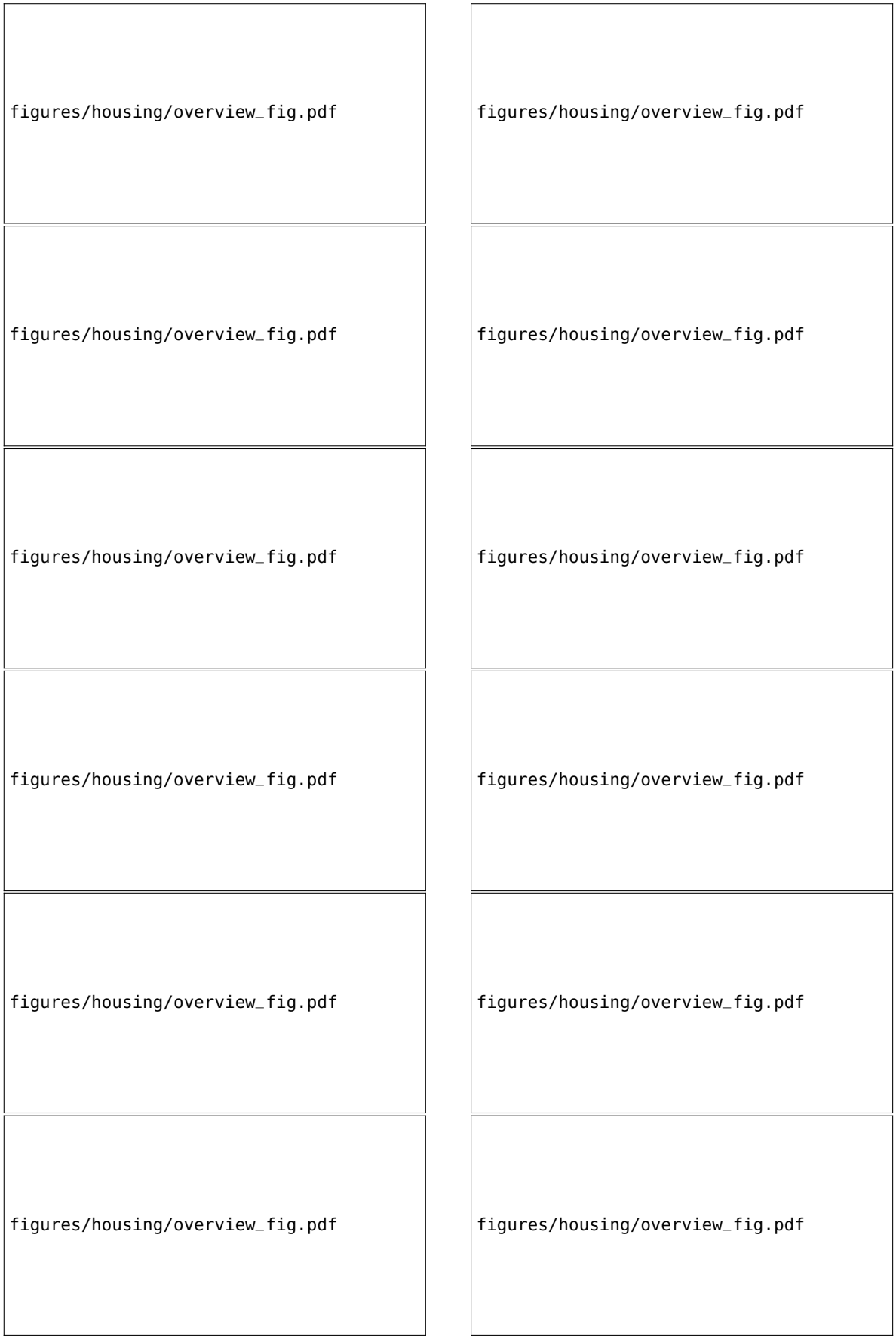


Figure A.10: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

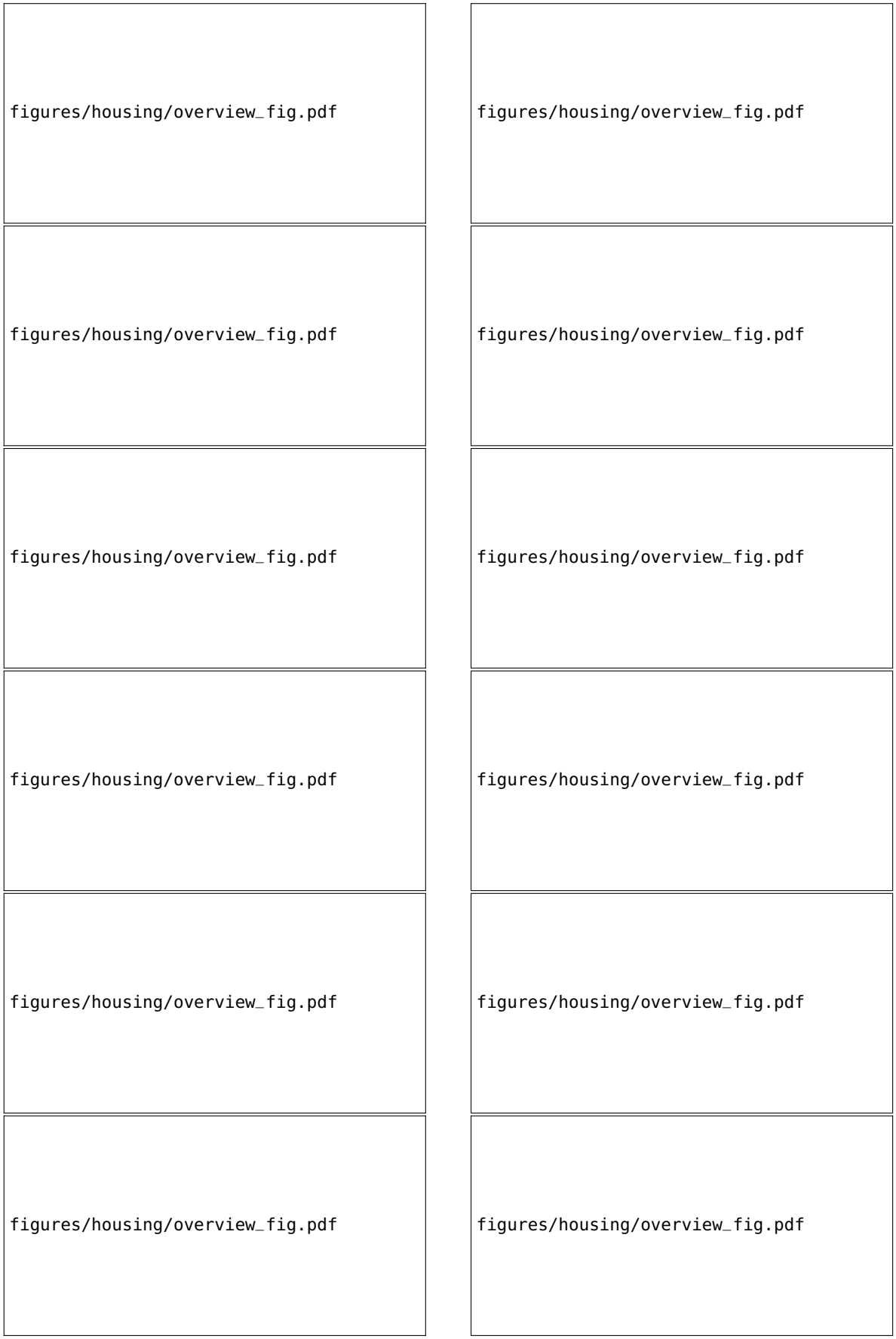


Figure A.11: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

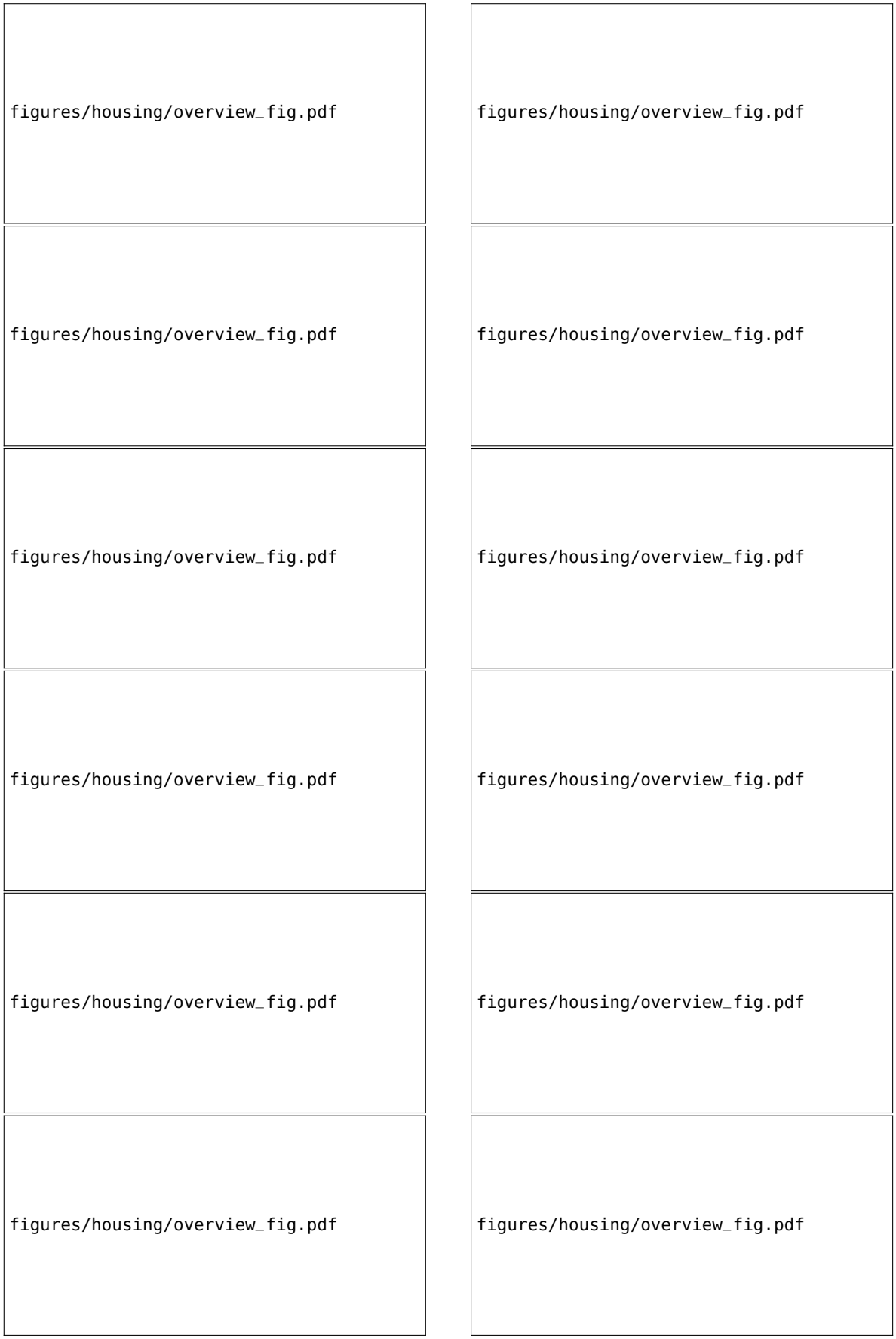


Figure A.12: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

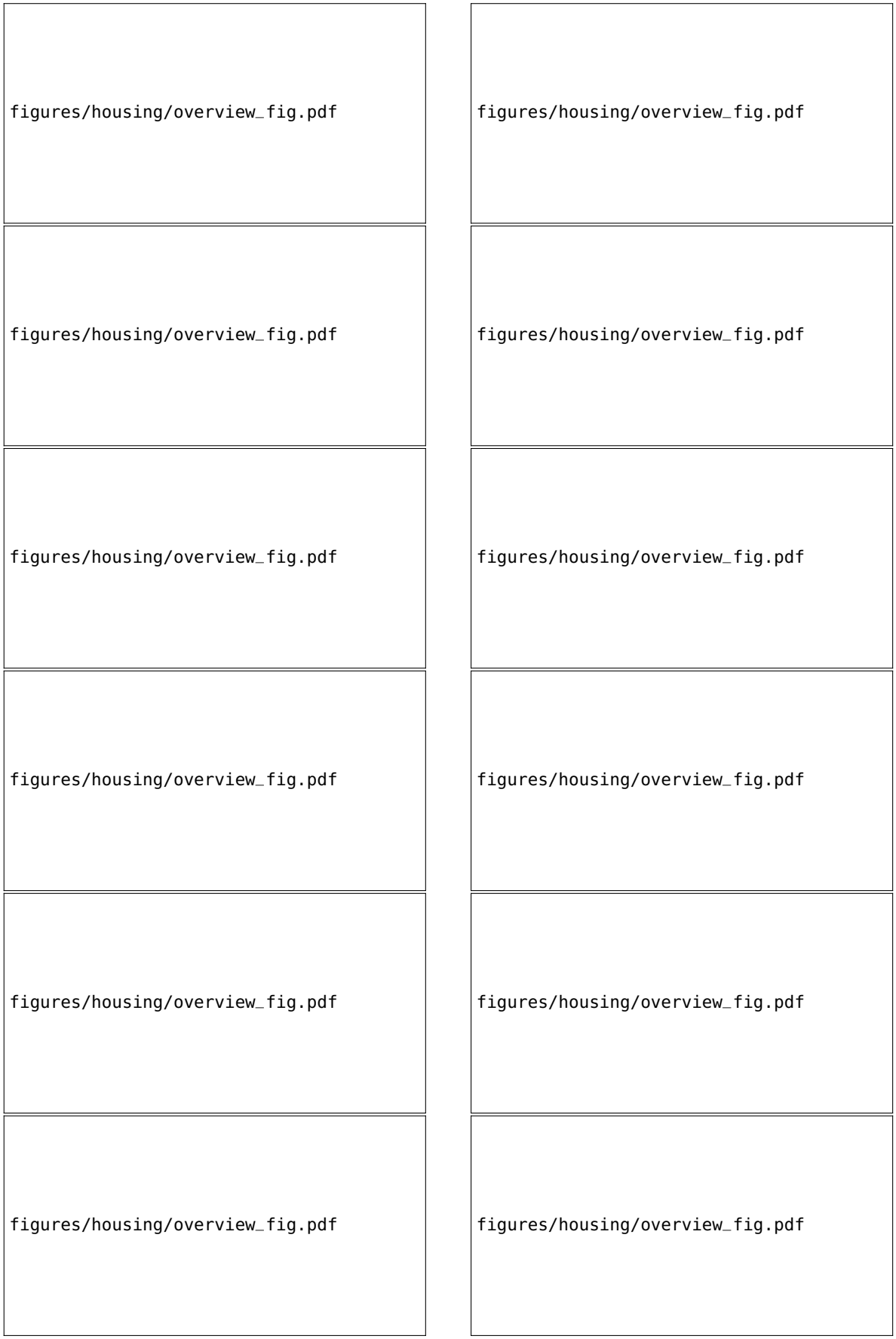


Figure A.13: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

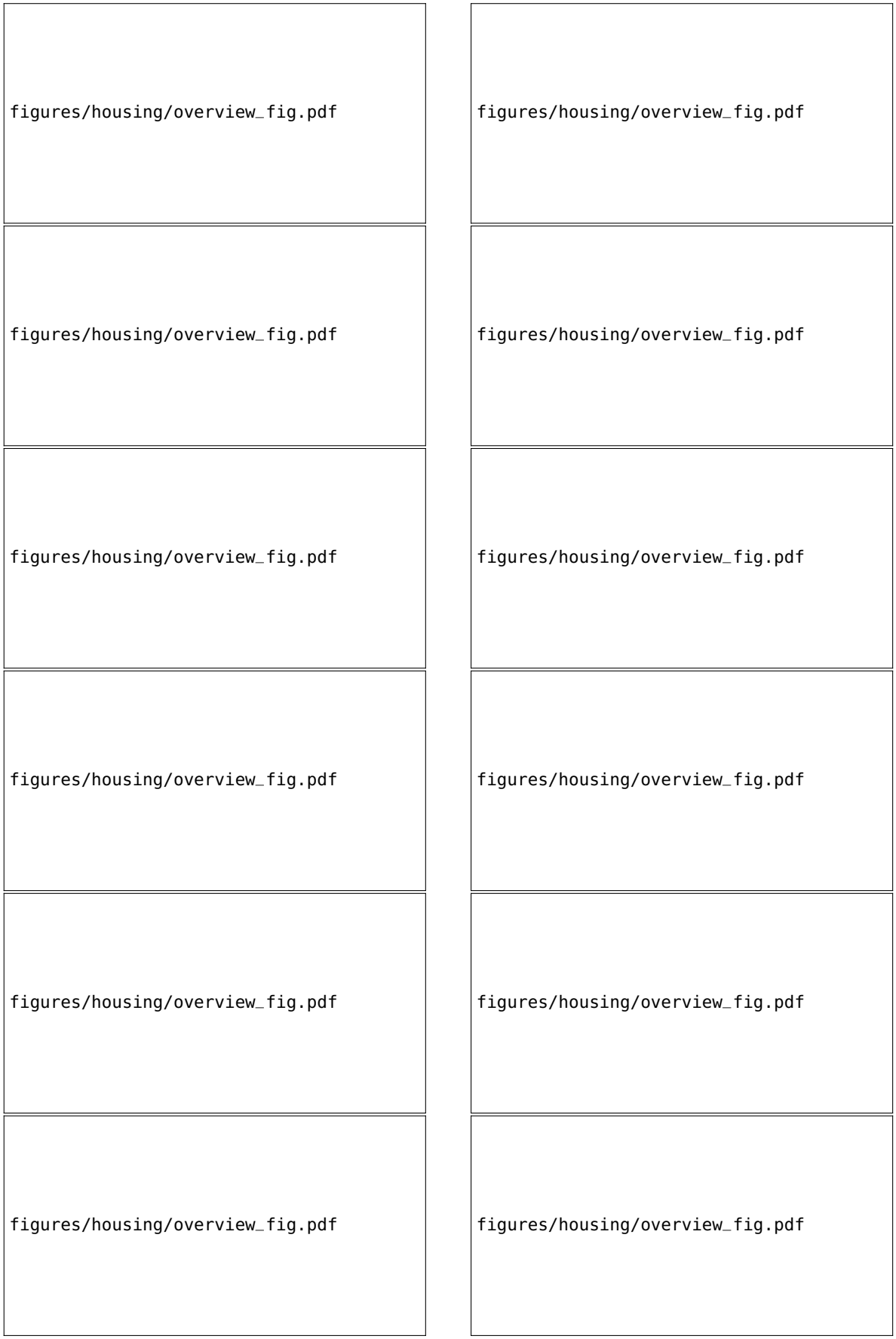


Figure A.14: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

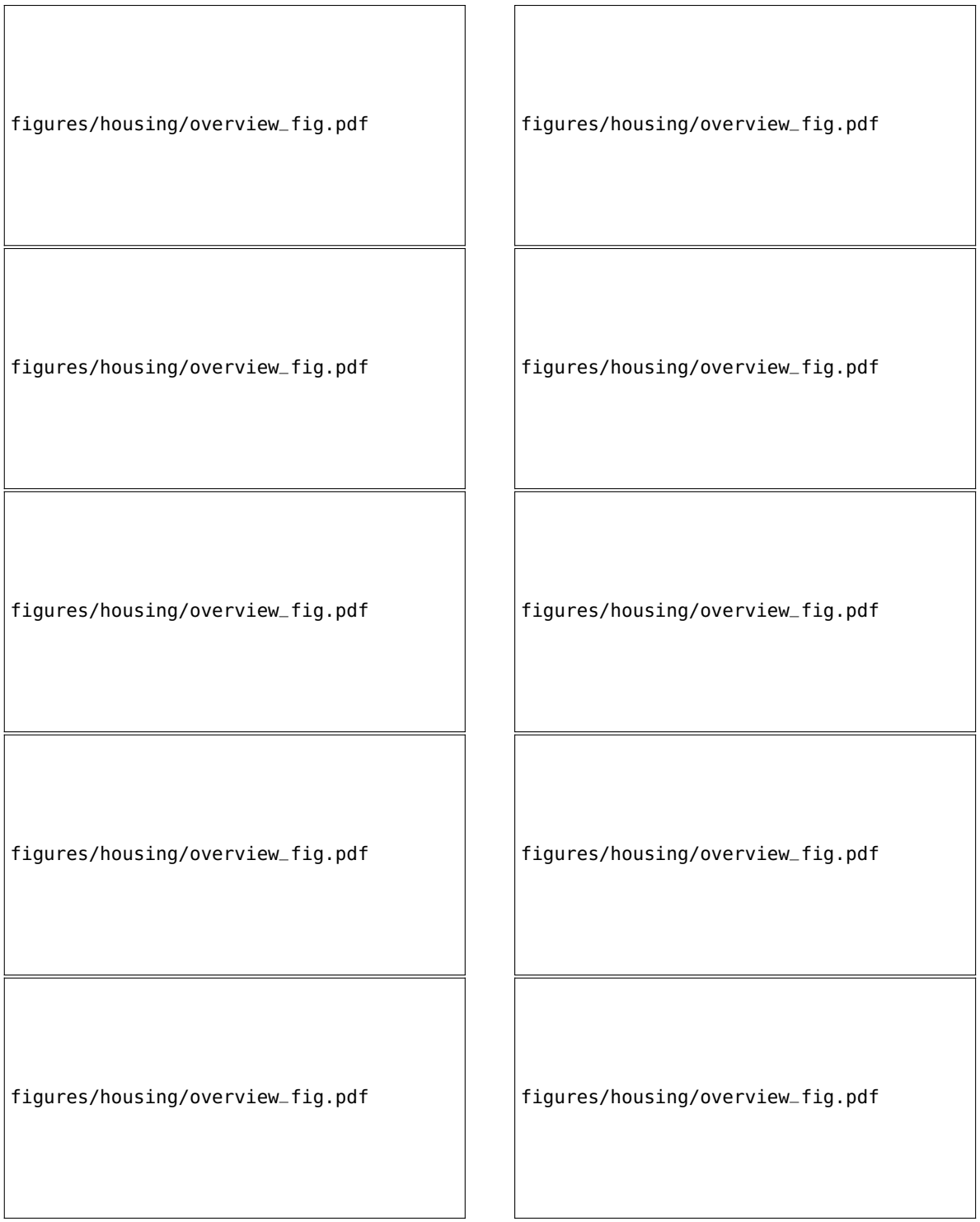


Figure A.15: Distributions the 168 input variables (excluding `ID` and `Vejnavn`).

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.

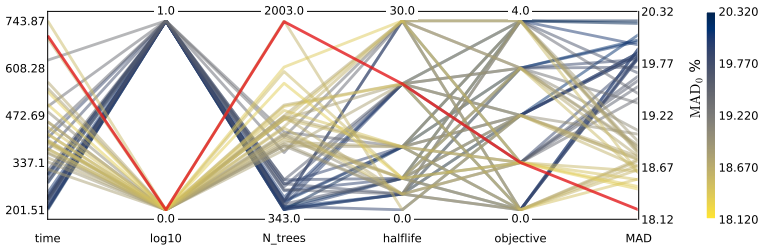


Figure A.20: Hyperparameter optimization results of the housing model for houses. The results are shown as parallel coordinates with each hyperparameter along the x-axis and the value of that parameter on the y-axis. Each line is an event in the 4-dimensional space colored according to the performance 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.

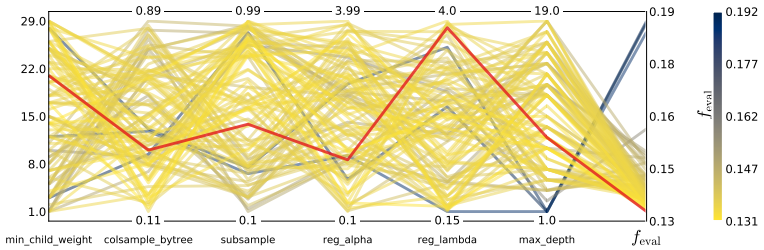


Figure A.21: Hyperparameter optimization results of XGBoost parameters of the housing model for apartments shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

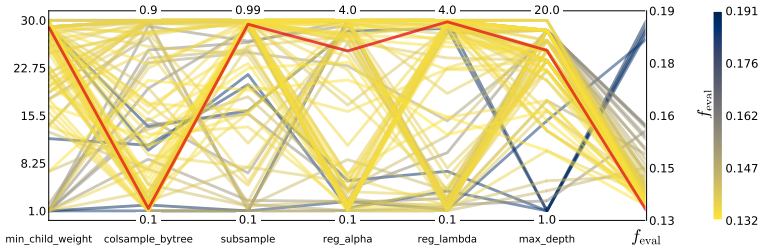


Figure A.22: Hyperparameter optimization results of XGBoost parameters of the housing model for apartments shown as parallel coordinates. Here shown for Bayesian optimization as hyperparameter optimization.

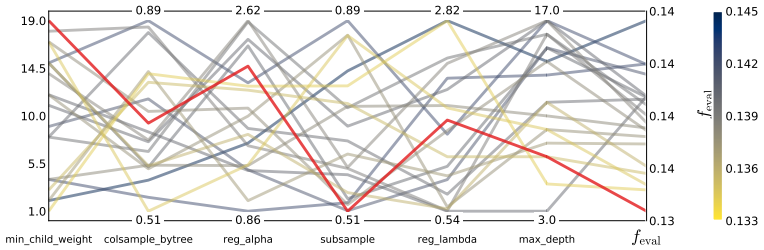


Figure A.23: Hyperparameter optimization results of XGBoost parameters of the housing model for houses shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

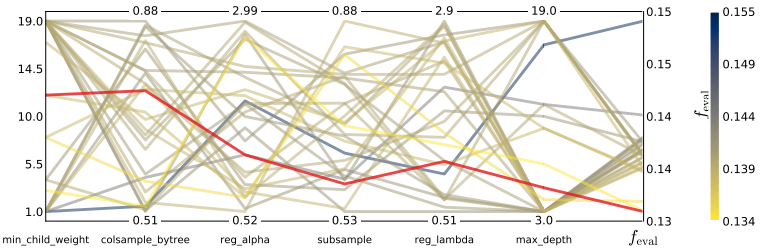


Figure A.24: Hyperparameter optimization results of XGBoost parameters of the housing model for houses shown as parallel coordinates. Here shown for Bayesian optimization as hyperparameter optimization.

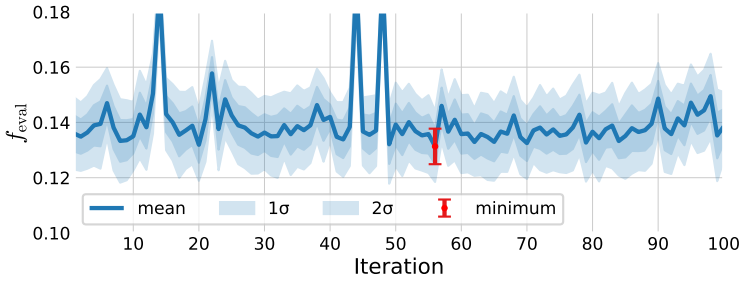


Figure A.25: XXX of the housing model for apartments shown as parallel coordinates. Here shown for random search as hyperparameter optimization.

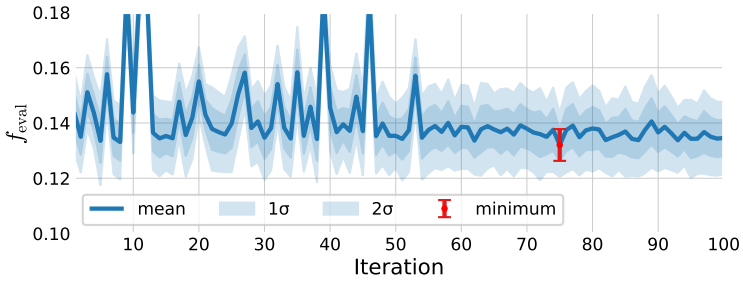


Figure A.26: XXX of the housing model for apartments shown as parallel coordinates. Here shown for Bayesian optimization as hyperparameter optimization.

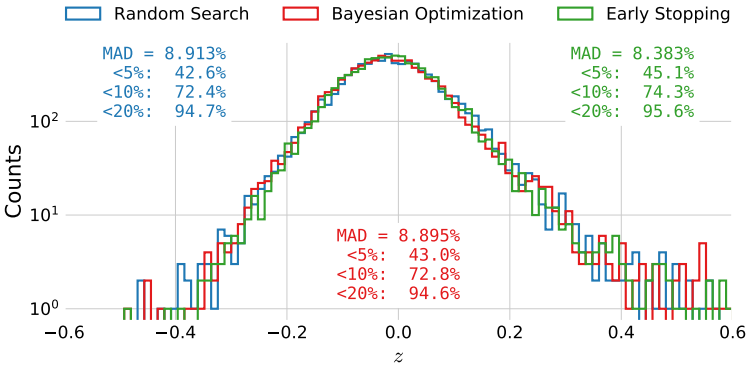


Figure A.27: Histogram of z-values of the XGB-model trained on apartments. The performance after hyperparameter optimization (HPO) using Random Search (RS) is shown in blue, for Bayesian Optimization (BO) in red. After finding the best model, BO in this case, the model is retrained using early stopping,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

	b	*c*	*uds*	*g*	non-*q*-matched
2	37.2 %	12.9 %	29.1 %	0.0 %	20.7 %
3	22.6 %	8.9 %	19.7 %	31.2 %	17.5 %
4	14.6 %	7.0 %	15.0 %	45.1 %	18.3 %
5	10.0 %	5.7 %	12.2 %	52.5 %	19.6 %
6	7.1 %	4.4 %	8.8 %	54.4 %	25.2 %

Table B.1: Number of different types of jets for MC and MCb written in relative numbers such that each row sum to 100 %. See also Table 5.3.

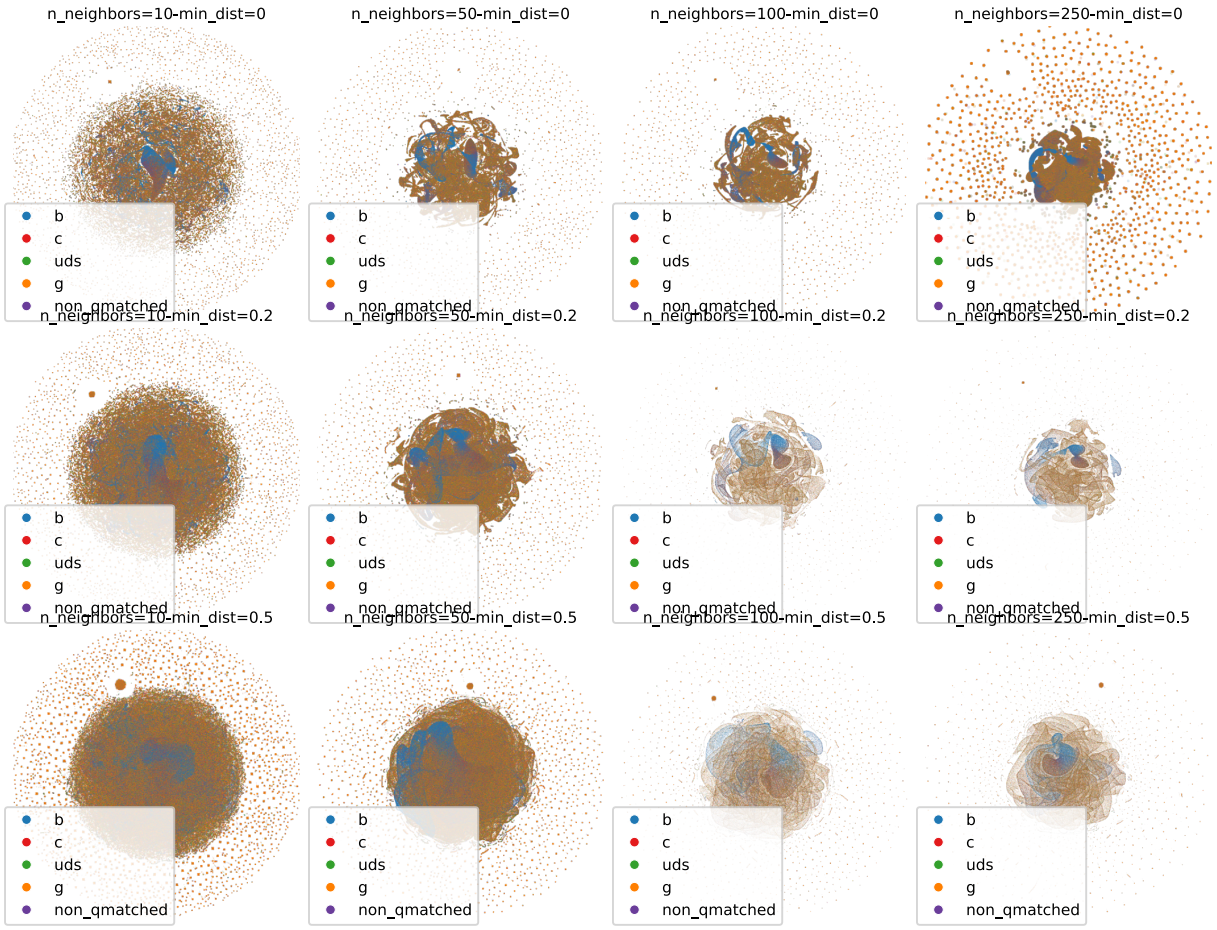


Figure B.1: Grid search of the two parameters `n_neighbors` and `min_dist` for the UMAP algorithm run on 4-jet events. For an explanation of these, see section 5.2.

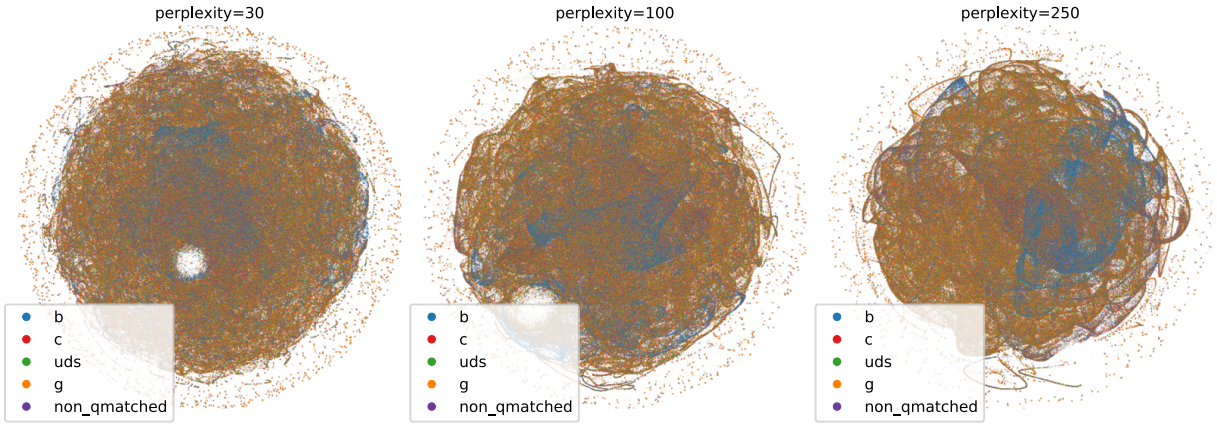


Figure B.2: Visualization of the t-SNE algorithm as a function of the `perplexity` parameters for 4-jet events.

Hyperparameter	Range
`subsample`	$\mathcal{U}(0.4, 1)$
`colsample_bytree`	$\mathcal{U}_{\text{trunc}}(0.4, 1, 2)$
`max_depth`	$\mathcal{U}_{\text{int}}(1, 20)$
`min_child_weight`	$\mathcal{U}_{\text{int}}(0, 10)$

Table B.2: Probability Density Functions for the random search hyperparameter optimization process for the XGBoost model.

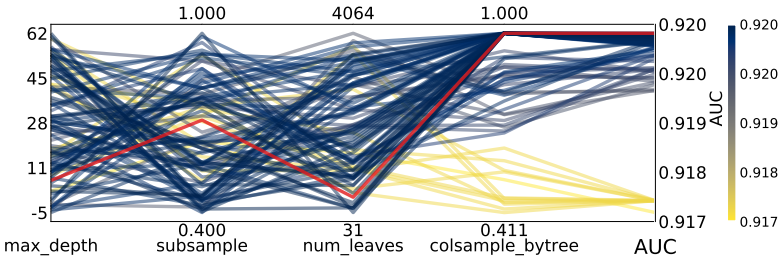


Figure B.3: Hyperparameter optimization results of b-tagging for 3-jet events. The results are shown as parallel coordinates with each hyperparameter along the x-axis and the value of that parameter on the y-axis. Each line is an event in the 4-dimensional space colored according to the performance of that hyperparameter as measured by AUC from highest AUC in dark blue to lowest AUC in yellow. The single best hyperparameter is shown in red.

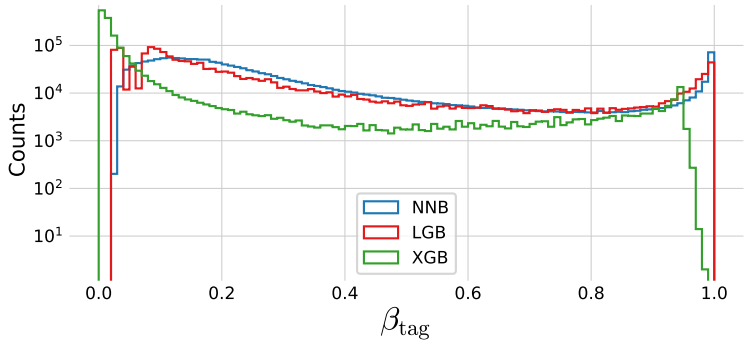


Figure B.4: Histogram of b-tag scores β_{tag} in 3-jet events for NNB (the neural network pre-trained by ALEPH, also called `nnbjet`) in blue, LGB in red, and XGB in green.

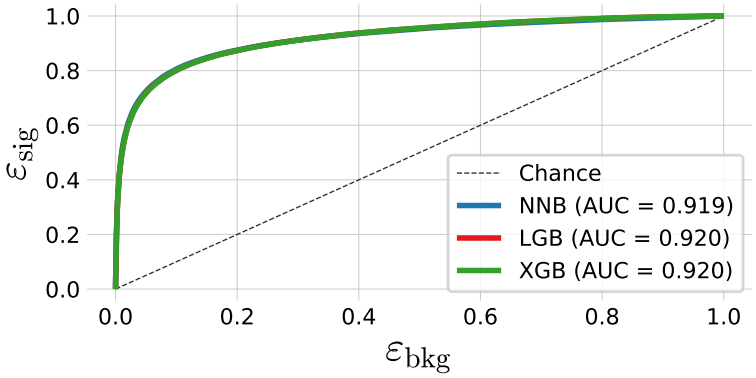


Figure B.5: ROC curve of the three b-tag models in 3-jet events for NNB (the pre-trained neural network trained by ALEPH, also called `nnbjet`) in blue, LGB in red, and XGB in green. In the legend the area under curve (AUC) is also shown. Notice that the LGB and XGB models share performance and it is thus due to overplotting that only the green line for XGB can be seen. In the machine learning community the background efficiency ε_{bkg} is sometimes know as the false positive rate (FPR) and the signal efficiency ε_{sig} as the true positive rate (TPR).

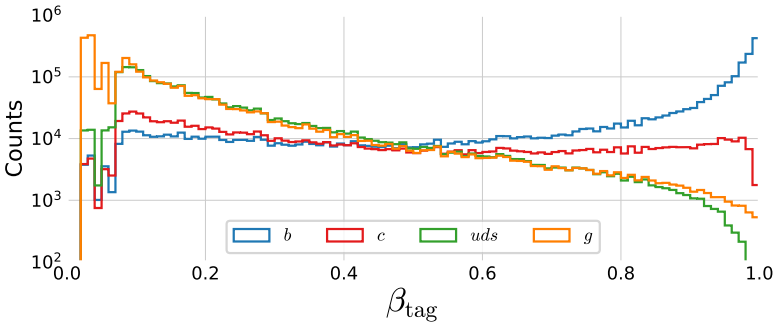


Figure B.6: Distribution of b-tags in 3-jet events for b-jets in blue, c-jets in red, uds in green and g in orange.

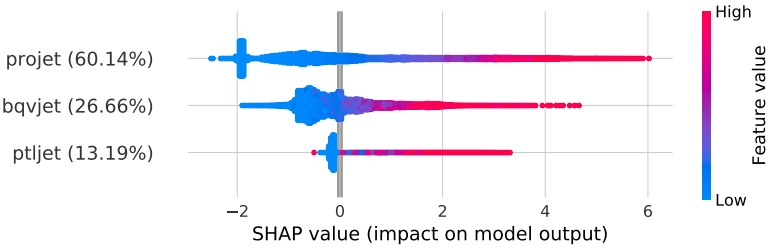


Figure B.7: Global feature importances for the LGB b-tagging algorithm on 3-jet events. The normalized feature importance is shown in the parenthesis and the each dot is an observation showing the dependance between the SHAP value and the feature's value.

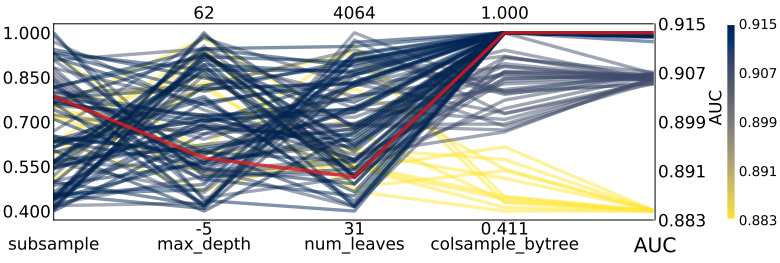


Figure B.8: Hyperparameter optimization results of g-tagging for 3-jet events for energy ordered jets.

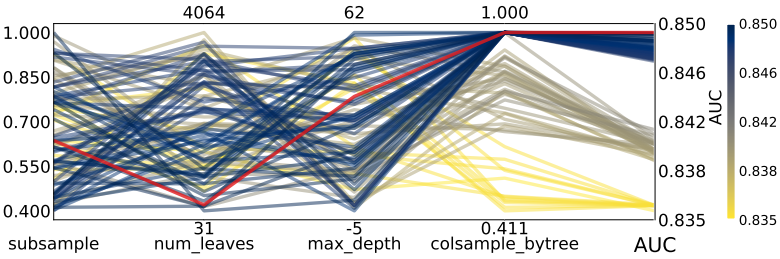


Figure B.9: Hyperparameter optimization results of g-tagging for 3-jet events for (row) shuffled jets.

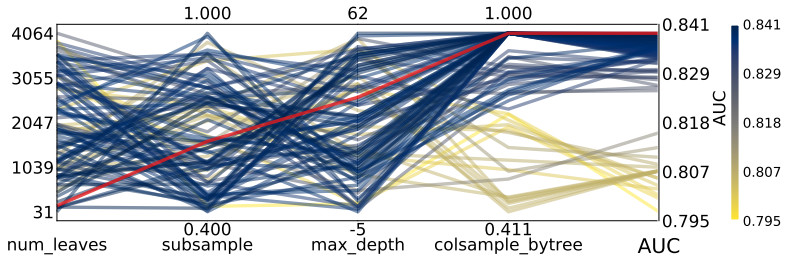


Figure B.10: Hyperparameter optimization results of g-tagging for 4-jet events for energy ordered jets.

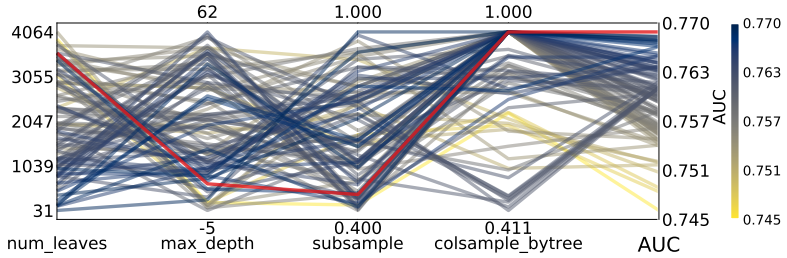


Figure B.11: Hyperparameter optimization results of g-tagging for 4-jet events for (row) shuffled jets.

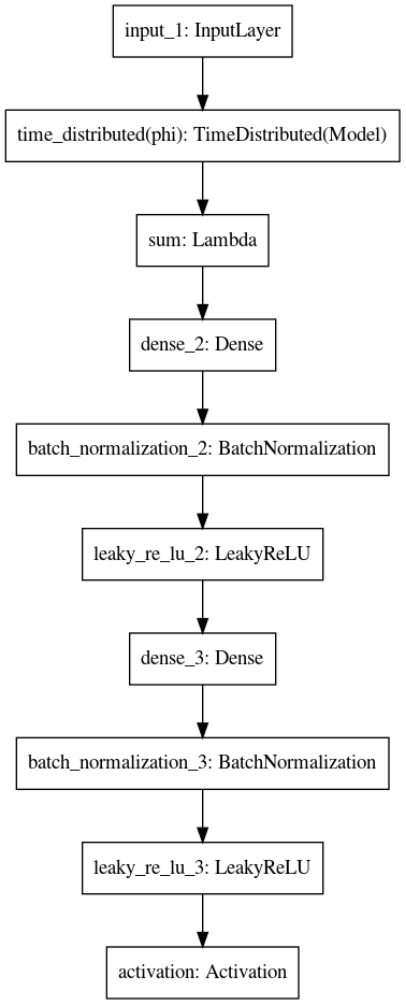


Figure B.12: Architecture of the PermNet neural network.

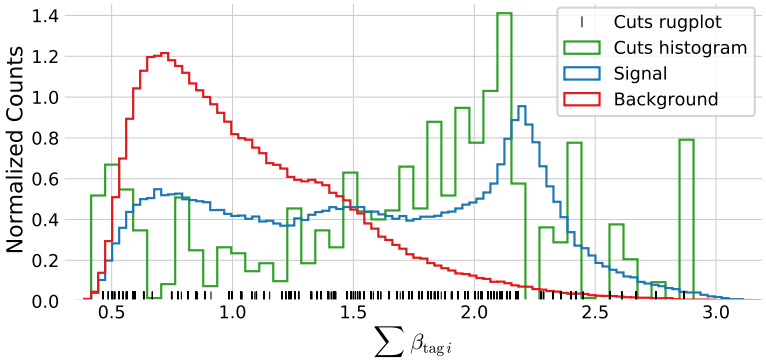


Figure B.13: Histogram of the distribution of signal in blue and background in red for the 1-dimensional sum of b-tags for 4-jet events. A histogram of the cut values from the LGB model trained on this data is shown in green together with a rug plot of the cut values in black. Notice how most of the cuts match up with the signal peak at around a $\sum \beta_i \sim 2.1$.

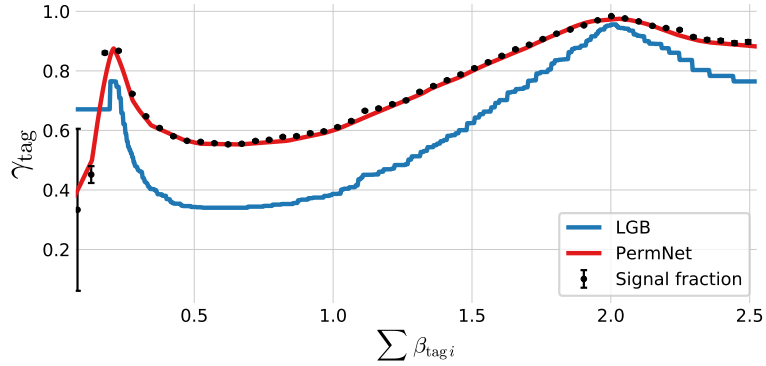


Figure B.14: Plot of the (1D) g-tag scores for 3-jet events as a function of $\sum \beta_i$ for the LGB model in blue and the PermNet model in red. The signal fraction (based on the signal and background histograms in Figure B.15) is plotted as black error bars where the size of the error bars is based on the propagated uncertainties of the signal and background histogram assuming Poissonian statistics.

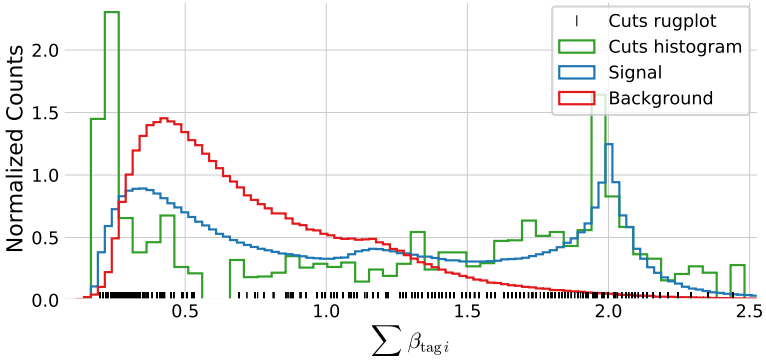


Figure B.15: Histogram of the distribution of signal in blue and background in red for the 1-dimensional sum of b-tags for 3-jet events. A histogram of the cut values from the LGB model trained on this data is shown in green together with a rug plot of the cut values in black. Notice how most of the cuts match up with the signal peak at around a $\sum \beta_i \sim 2.1$.

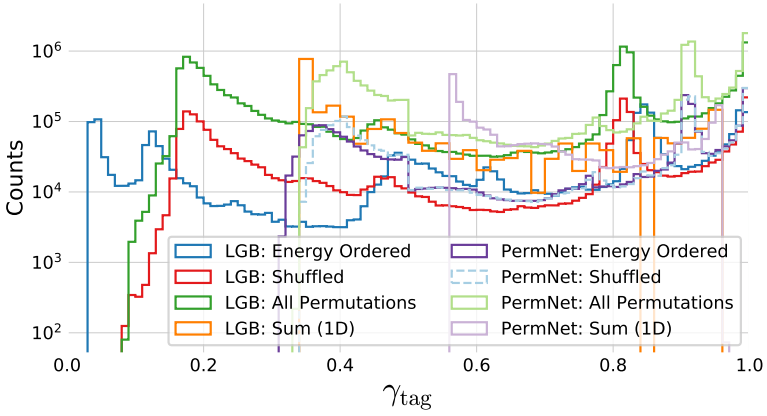


Figure B.16: Distribution of g-tag scores in 3-jet events shown with a logarithmic y-scale for LGB: Energy Ordered in blue, LGB: Shuffled in red, LGB: All Permutations in green, LGB: Sum 1D in orange, PermNet: Energy Ordered in purple, PermNet: Shuffled in light-blue, PermNet: All Permutations in light-green, PermNet: Sum 1D in light-purple. Here LGB and PermNet are the two different type of models and "Energy Ordered", "Shuffled", "All Permutations", and "Sum 1D" are the different methods used for making the input data permutation invariant (except energy ordered).

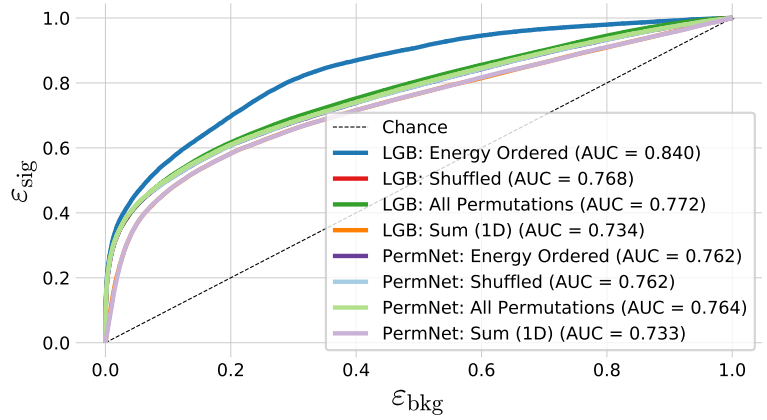


Figure B.17: ROC curve of the eight g-tag models in 4-jet events. First one in dashed black is the ROC curve that you get by random chance. The colors are the same as in Figure 5.20 and in the legend also the Area Under the ROC curve (AUC) is shown. Notice that the XGB model which uses the energy ordered data produced the best model, however, this model is not permutation invariant. Of the permutation invariant models (the rest), the XGB model trained on all permutations of the b-tags performs highest. The lowest performing models are the two models trained only on the 1-dimensional sum of b-tags, as expected, however, still with a better performance than expected by the author.

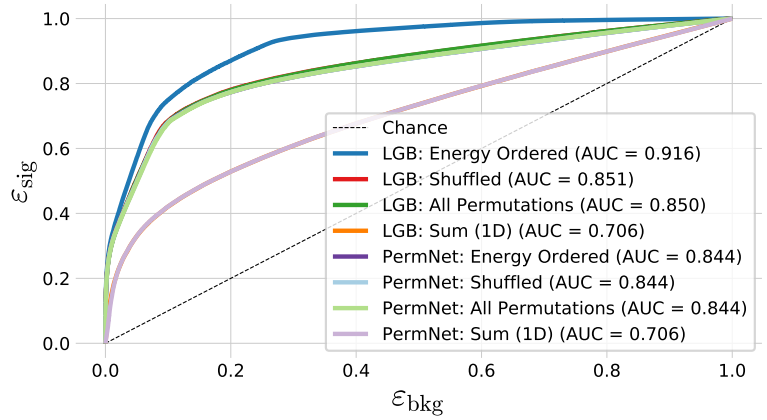


Figure B.18: ROC curve of the eight g-tag models in 3-jet. First one in dashed black is the ROC curve that you get by random chance. The colors are the same as in Figure 5.20 and in the legend also the Area Under the ROC curve (AUC) is shown.

β_{tag_i}	Energy Ordered	Shuffled	All Permutations
1	0.827 ± 0.006	0.924 ± 0.006	0.923 ± 0.006
2	0.749 ± 0.006	0.909 ± 0.006	0.918 ± 0.005
3	1.198 ± 0.006	0.878 ± 0.005	0.906 ± 0.005

Table B.3: Global SHAP feature importances $\phi^{\text{tot}}_{\beta_i}$ for the three g-Tagging Models in 3-Jet Events. Each $\phi^{\text{tot}}_{\beta_i}$ is shown for the three methods in the columns and the three b-tags as variables in the rows.

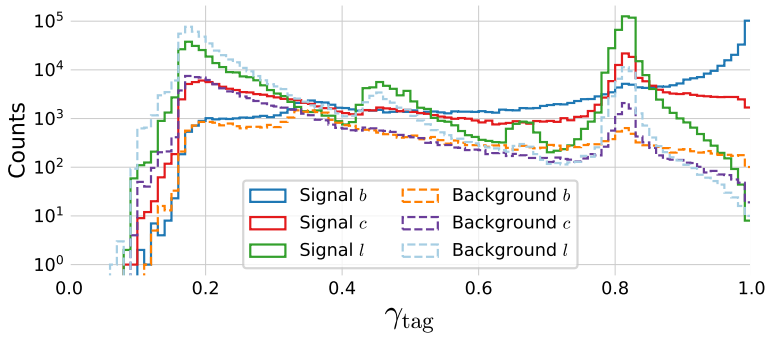


Figure B.19: Histogram of g-tag scores from the LGB-model in 3-jet events for b signal in blue, c signal in red, l (uds) signal in green, b background in orange, c background in purple, l (uds) background in light-blue.

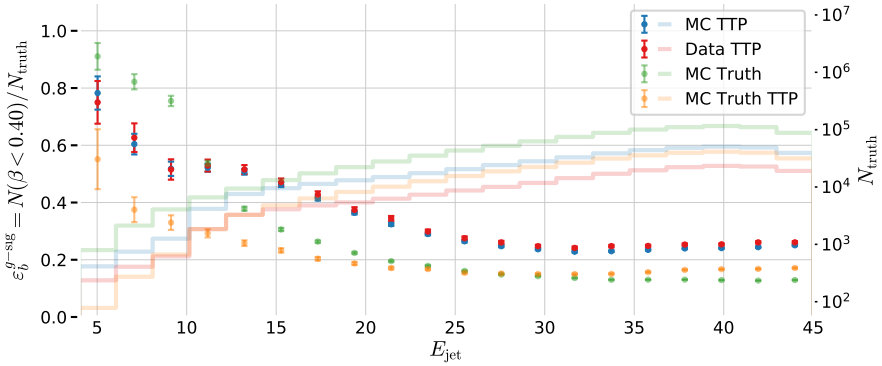


Figure B.20: b-tag efficiency for b-jets in the g-signal region for 3-jet events, $\varepsilon_b^{g\text{-sig}}$, as a function of jet energy E_{jet}. In the plot the efficiencies are shown for MC TTP in blue, Data TTP in red, MC Truth TTP in green, and MC Truth TTP in orange. The efficiencies (the errorbars) can be read off on the left y-axis and the counts (histograms) on the right y-axis.

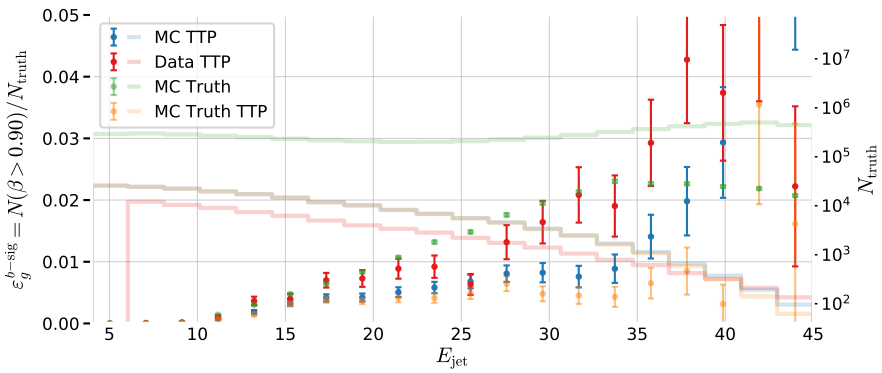


Figure B.21: b-tag efficiency for g-jets in the b-signal region for 3-jet events, $\varepsilon_g^{b\text{-sig}}$, as a function of jet energy E_{jet}. In the plot the efficiencies are shown for MC TTP in blue, Data TTP in red, MC Truth TTP in green, and MC Truth TTP in orange. The efficiencies (the errorbars) can be read off on the left y-axis and the counts (histograms) on the right y-axis.

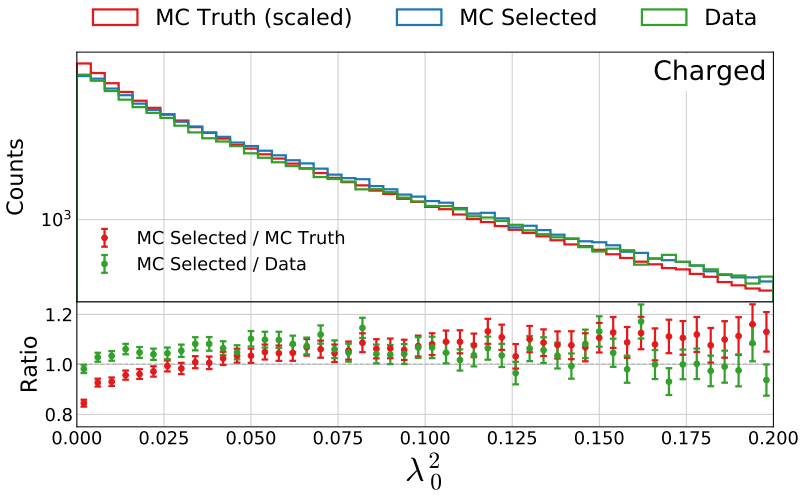


Figure B.22: Distribution of the generalized angularity λ_0^2 for charged gluons jets in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

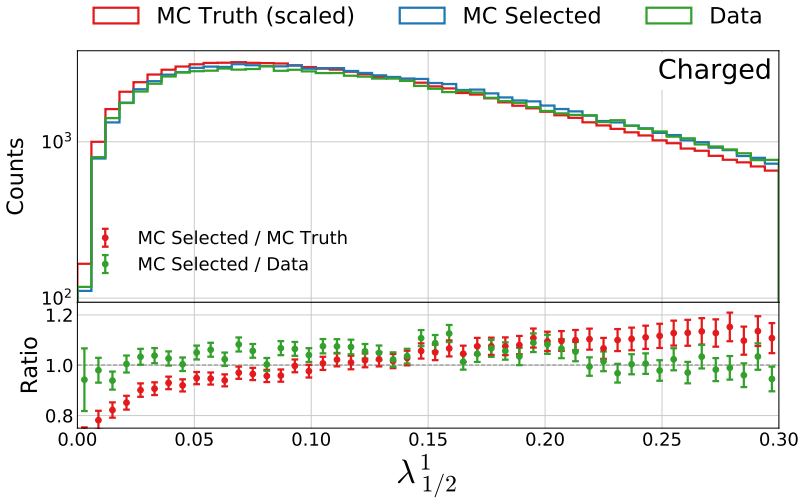


Figure B.23: Distribution of the generalized angularity $\lambda_{\frac{1}{2}}^1$ for charged gluons jets in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

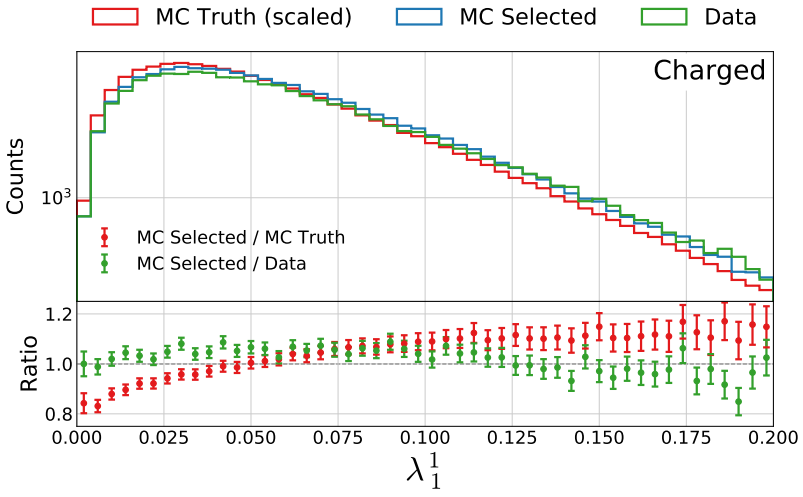


Figure B.24: Distribution of the generalized angularity λ_1^1 for charged gluons jets in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

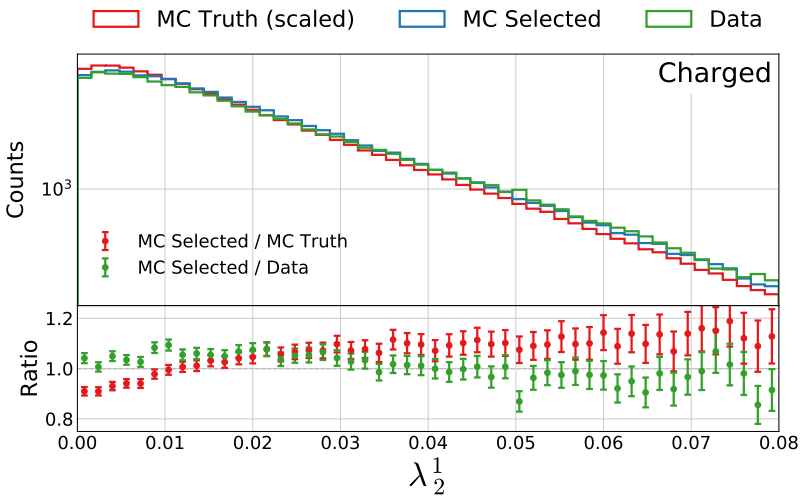


Figure B.25: Distribution of the generalized angularity λ_1^2 for charged gluons jets in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

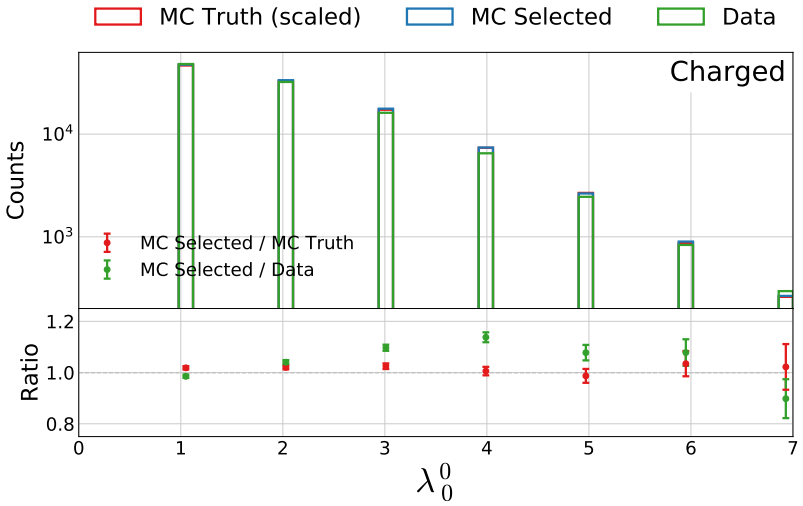


Figure B.26: Distribution of the generalized angularity λ_0^0 for charged gluons jets in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

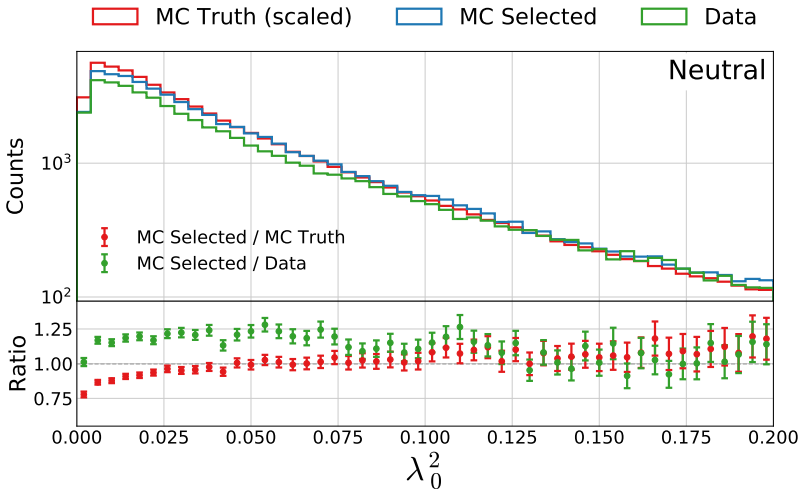


Figure B.27: Distribution of the generalized angularity λ_0^2 for charged gluons jets in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

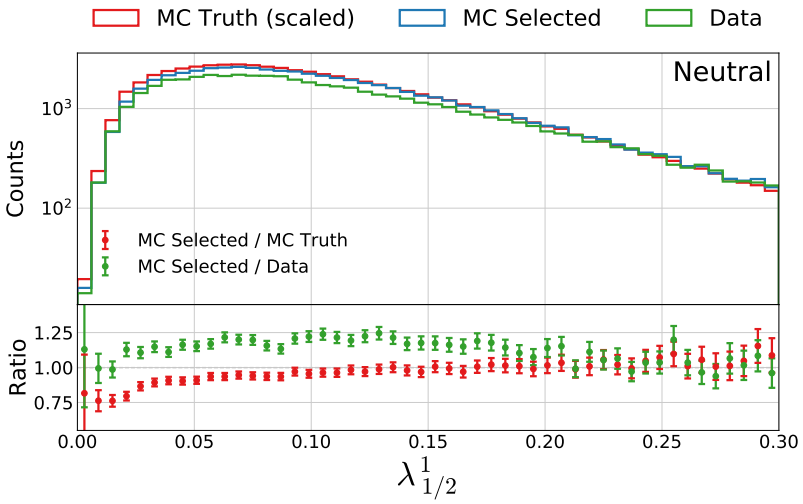


Figure B.28: Distribution of the generalized angularity $\lambda^1_{\frac{1}{2}}$ for neutral gluons clusters in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

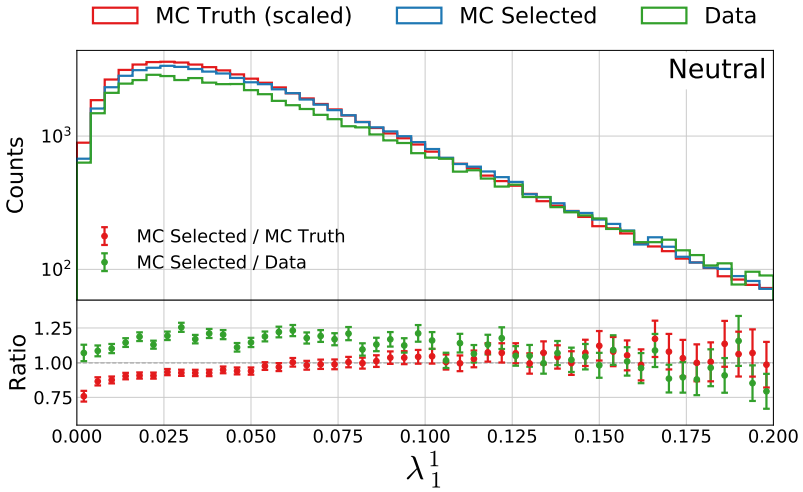


Figure B.29: Distribution of the generalized angularity λ^1_1 for neutral gluons clusters in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

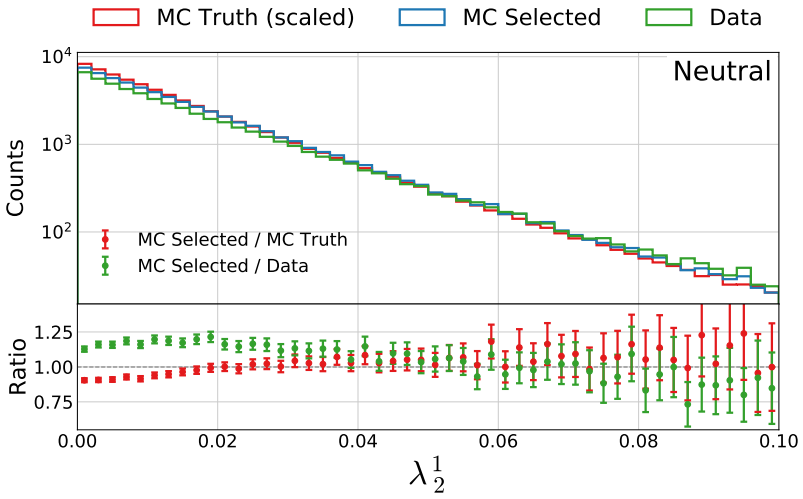


Figure B.30: Distribution of the generalized angularity λ^2_1 for neutral gluons clusters in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

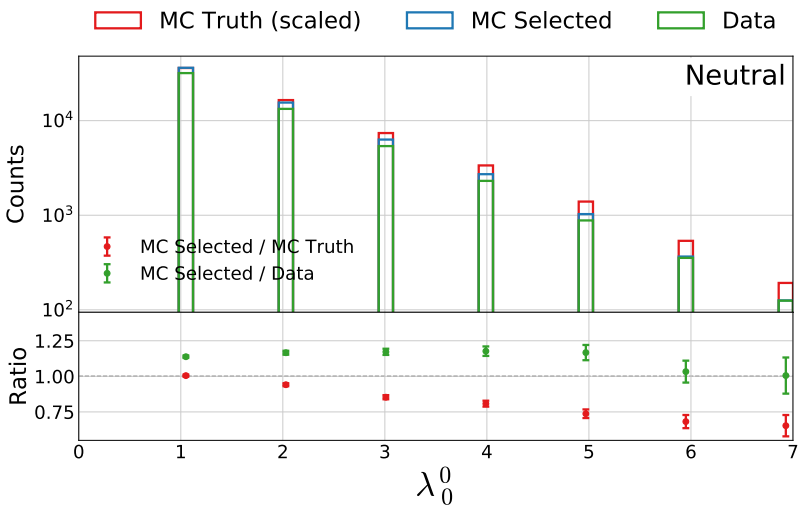


Figure B.31: Distribution of the generalized angularity λ_0^0 for neutral gluons clusters in 3-jet events. The distributions for MC Truth is shown in red, MC Truth in blue, and Data in green in the top plot and in the bottom plot the ratio between MC Selected and MC Truth is shown in red and between MC Selected and Data in green.

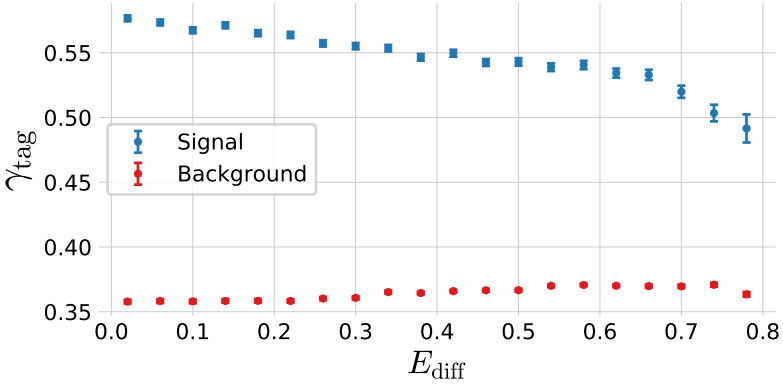


Figure B.32: Relationship between the g-tag value γ_{tag} and the gluon splitting variable E_{diff}. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

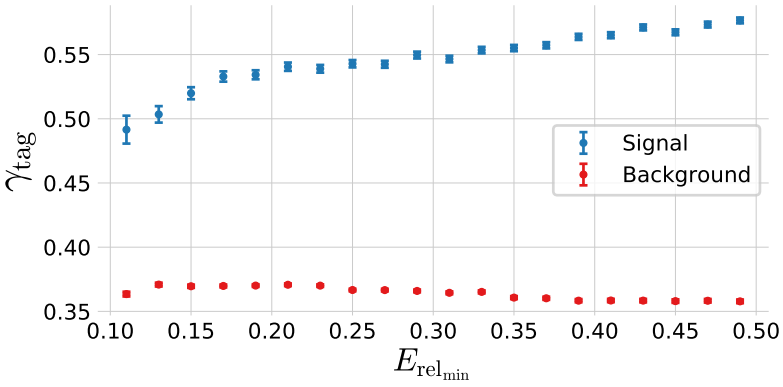


Figure B.33: Relationship between the g-tag value γ_{tag} and the gluon splitting variable $E_{rel_{min}}$. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

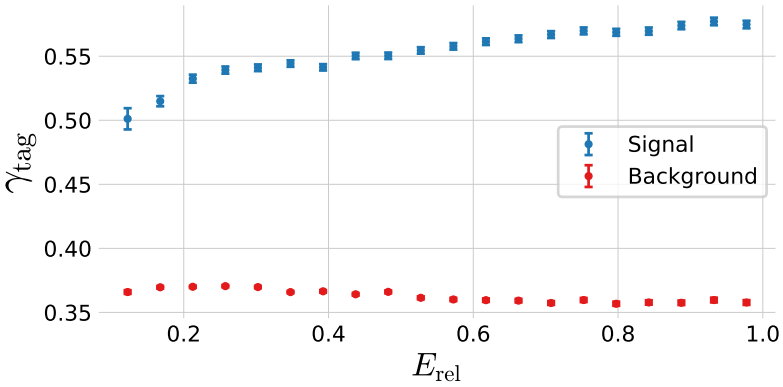


Figure B.34: Relationship between the g-tag value γ_{tag} and the gluon splitting variable E_{rel}. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

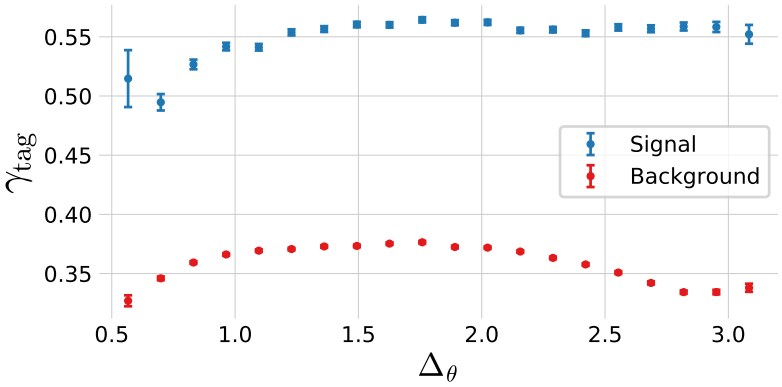


Figure B.35: Relationship between the g-tag value γ_{tag} and the gluon splitting variable Δ_θ. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

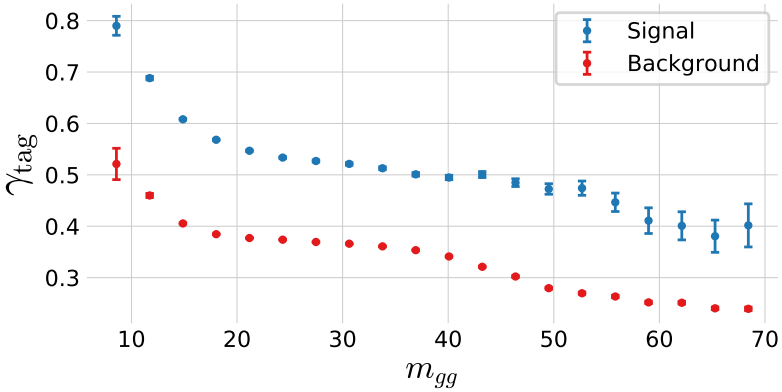


Figure B.36: Relationship between the g-tag value γ_{tag} and the gluon splitting variable m_{gg}. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

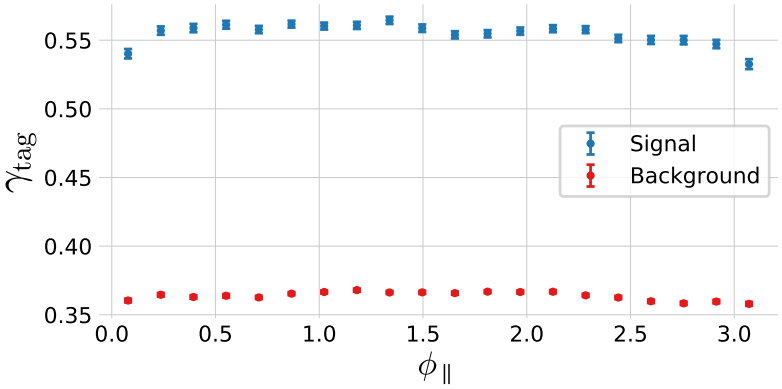


Figure B.37: Relationship between the g-tag value γ_{tag} and the gluon splitting variable $\phi_\parallel$. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

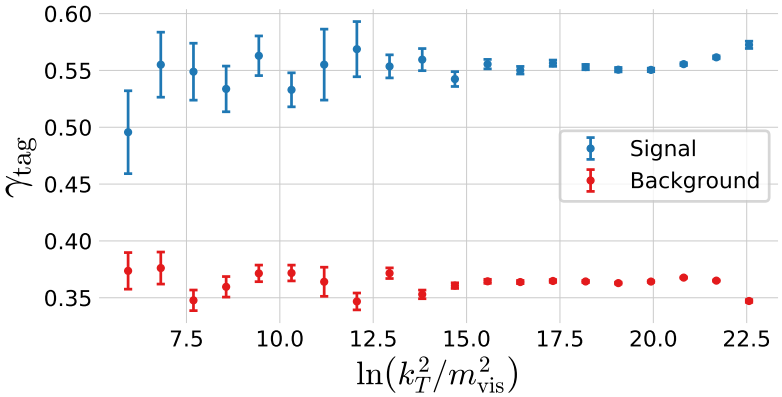


Figure B.38: Relationship between the g-tag value γ_{tag} and the gluon splitting variable $\ln\left(k_t^2/m_{\text{vis}}^2\right)$. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

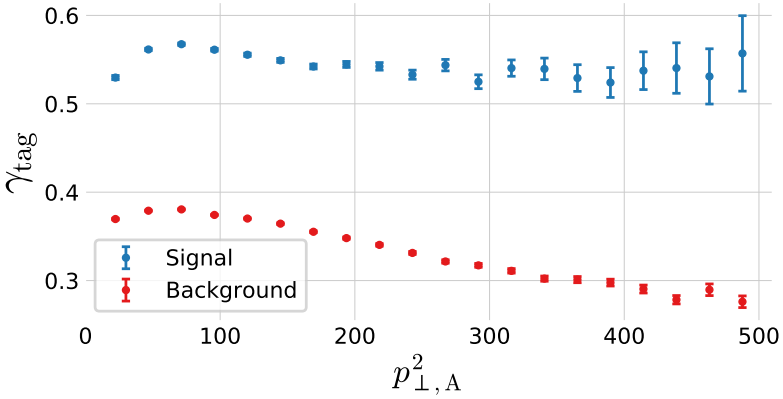


Figure B.39: Relationship between the g-tag value γ_{tag} and the gluon splitting variable $p^2_{\perp,\text{A}}$. The signal events (according to MC Truth) are plotted in blue and background events (according to MC Truth) in red.

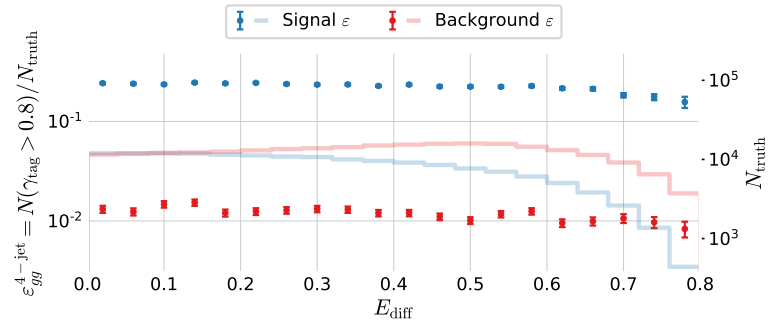


Figure B.40: Efficiency of the g-tagging algorithm for 4-jet events as a function of normalized gluon-gluon jet energy difference (asymmetry) E_{diff} in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

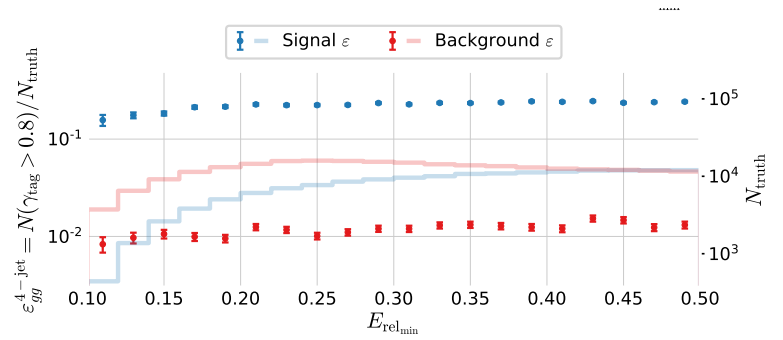


Figure B.41: Efficiency of the g-tagging algorithm for 4-jet events as a function of $E_{\text{rel}_{\text{min}}}$ in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

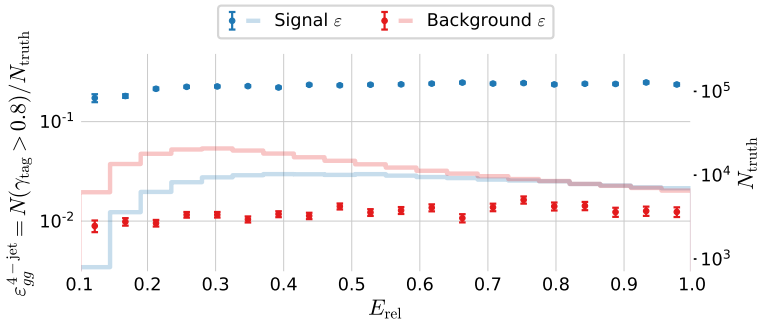


Figure B.42: Efficiency of the g-tagging algorithm for 4-jet events as a function of E_{rel} in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

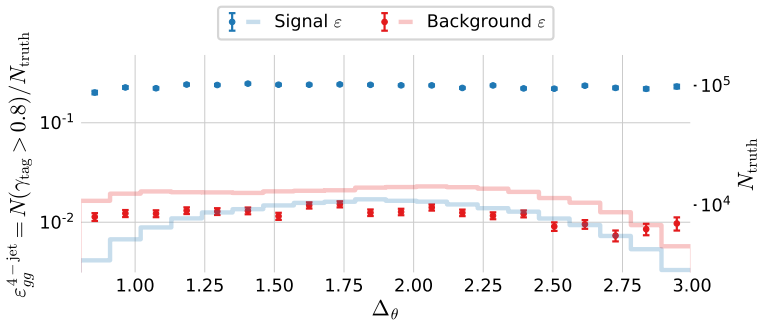


Figure B.43: Efficiency of the g-tagging algorithm for 4-jet events as a function of Δ_θ in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

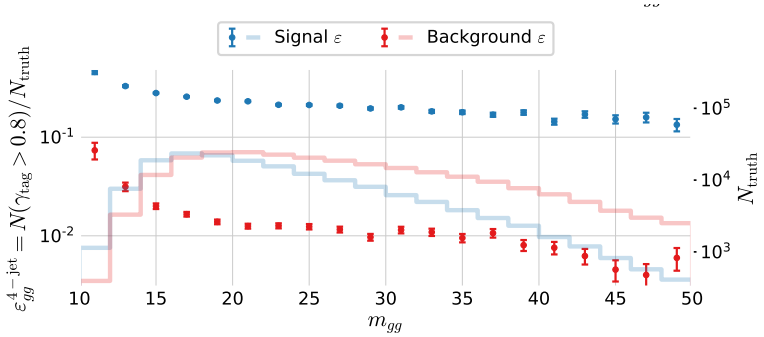


Figure B.44: Efficiency of the g-tagging algorithm for 4-jet events as a function of m_{gg} in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

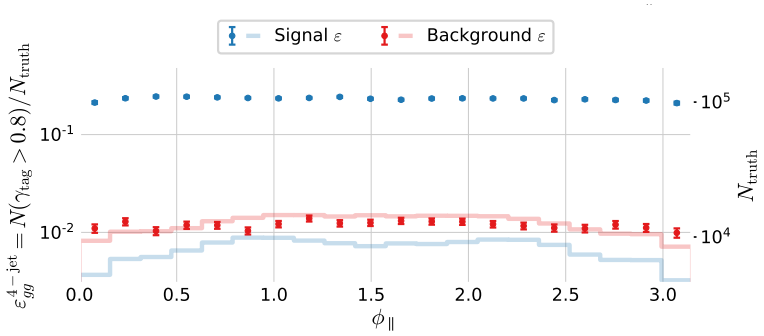


Figure B.45: Efficiency of the g-tagging algorithm for 4-jet events as a function of $\phi_{\parallel}$ in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

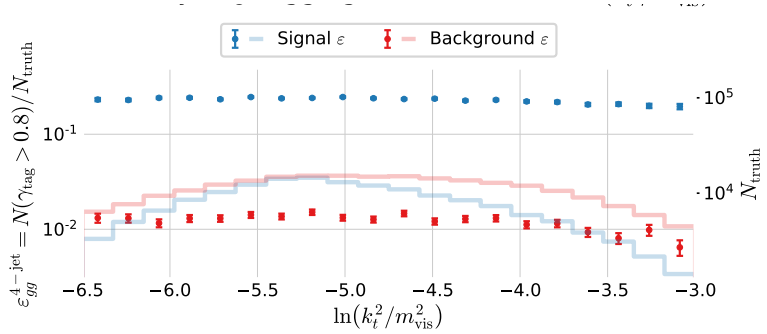


Figure B.46: Efficiency of the g-tagging algorithm for 4-jet events as a function of $\ln\left(k_t^2/m_{\text{vis}}^2\right)$ in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

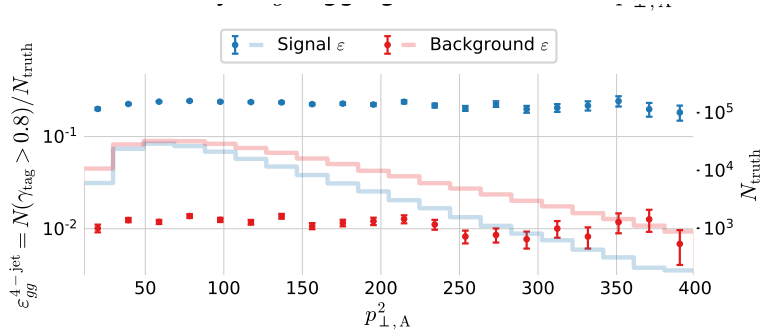


Figure B.47: Efficiency of the g-tagging algorithm for 4-jet events as a function of $p_{\perp,\text{A}}^2$ in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

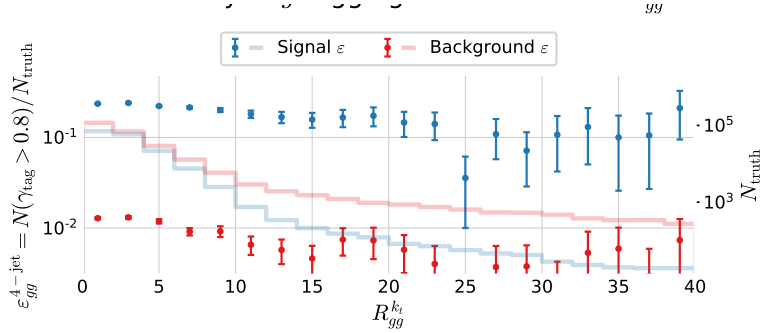


Figure B.48: Efficiency of the g-tagging algorithm for 4-jet events as a function of $R_{gg}^{k_t}$ in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

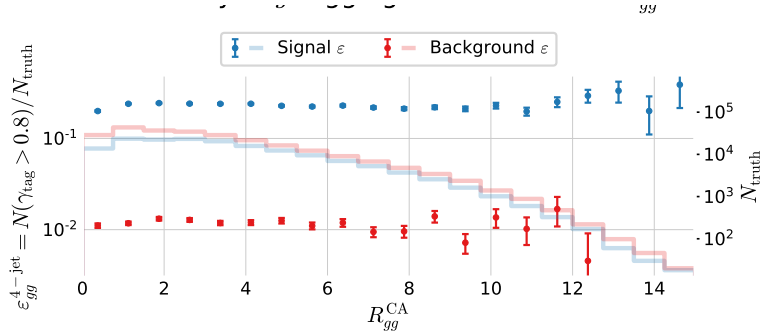


Figure B.49: Efficiency of the g-tagging algorithm for 4-jet events as a function of E in MC. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

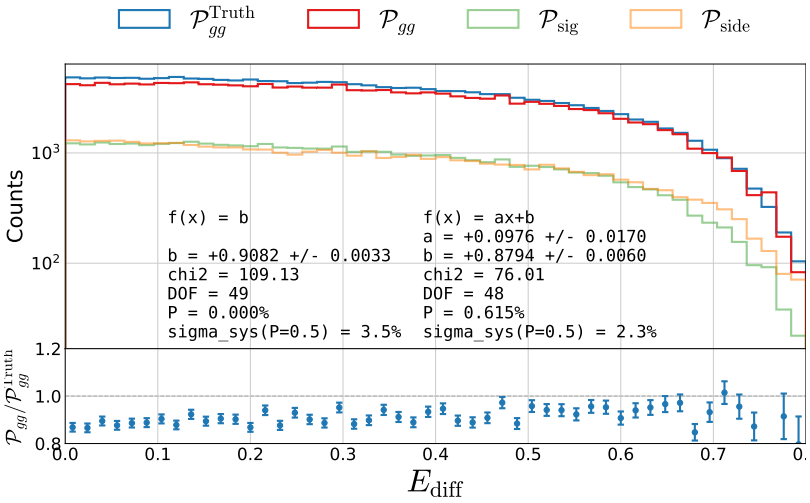


Figure B.50: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for E_{diff}. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

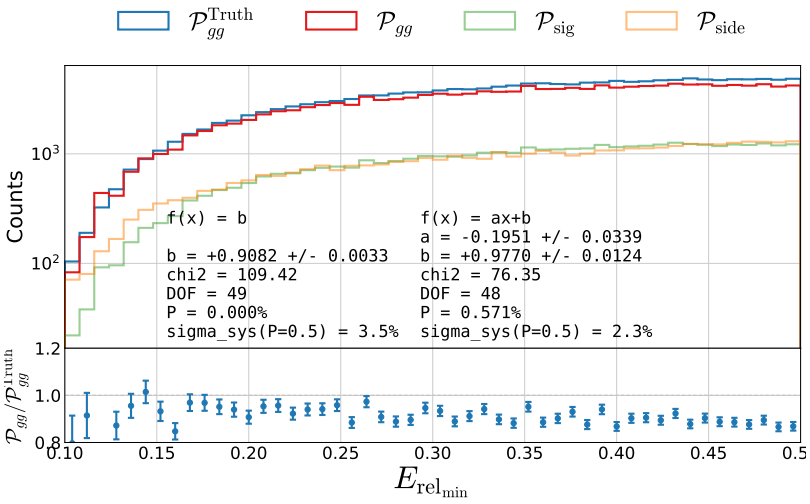


Figure B.51: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for $E_{\text{rel}_{\text{min}}}$. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

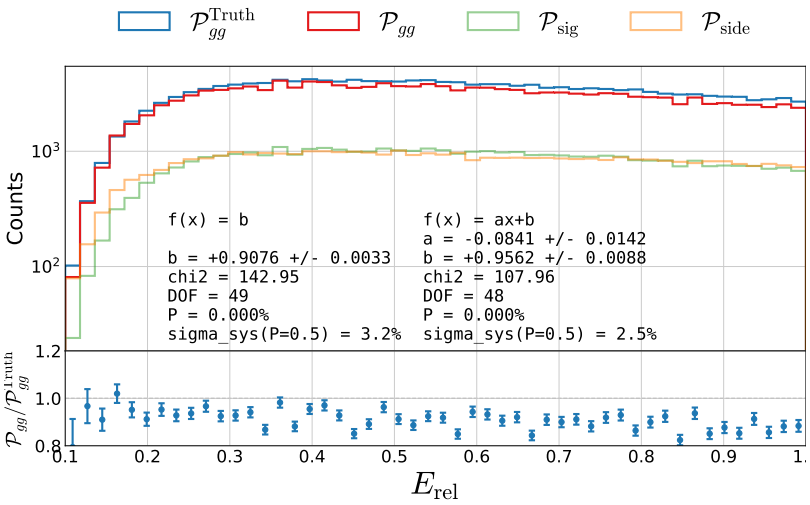


Figure B.52: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for E_{rel}. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

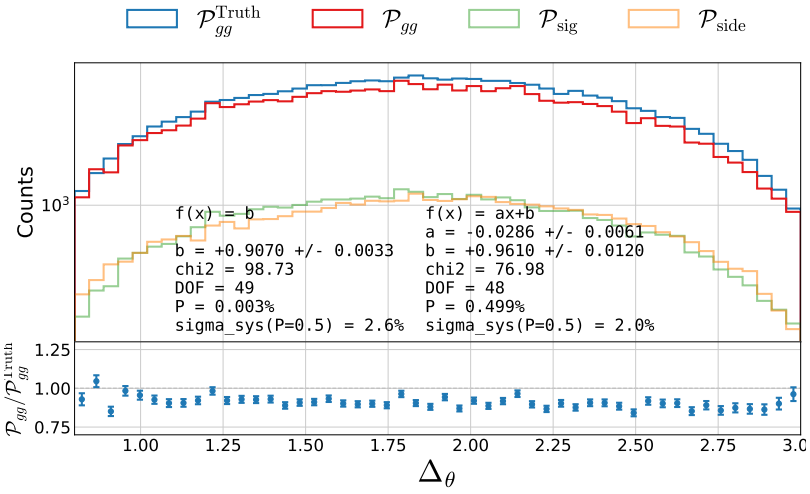


Figure B.53: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for Δ_θ. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

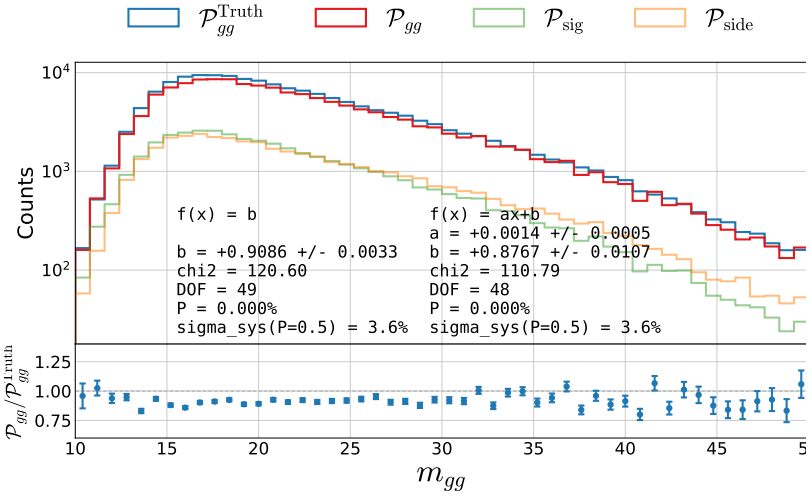


Figure B.54: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for m_{gg}. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

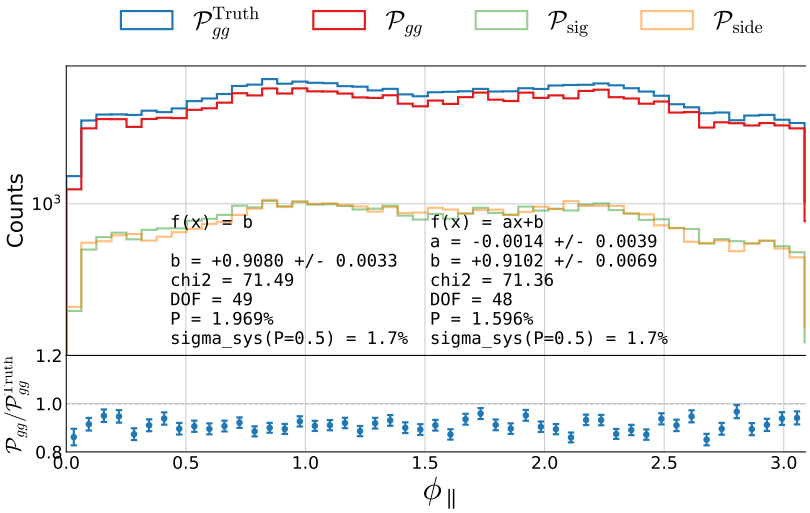


Figure B.55: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for $\phi_{\parallel}$. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

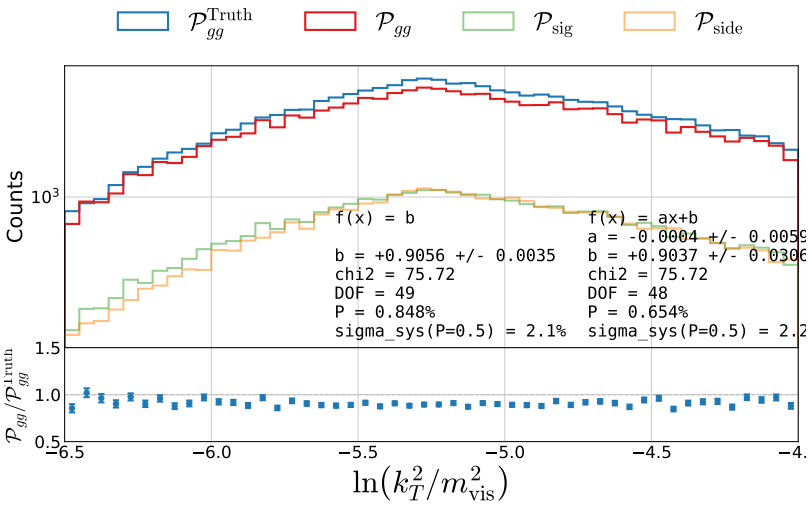


Figure B.56: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for $\ln\left(k_t^2/m_{\text{vis}}^2\right)$. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

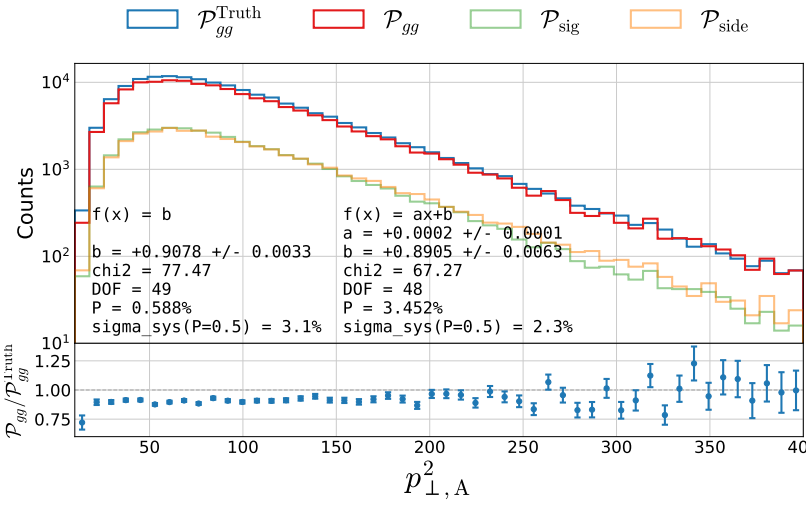


Figure B.57: Closure plot comparing MC Truth and the efficiency corrected g-tagging model in 4-jet events for $p_{\perp,\text{A}}^2$. In the top part of the plot $\mathcal{P}_{gg}^{\text{Truth}}$ based on MC Truth is shown in blue, the $\mathcal{P}_{gg}$ based on MC but without Truth in red, the distribution in the signal region $\mathcal{P}_{\text{sig}}$ in light green and the distribution in the sideband region $\mathcal{P}_{\text{side}}$ in light orange. In the bottom part of the plot the ratio between $\mathcal{P}_{gg}$ and $\mathcal{P}_{gg}^{\text{Truth}}$ is shown.

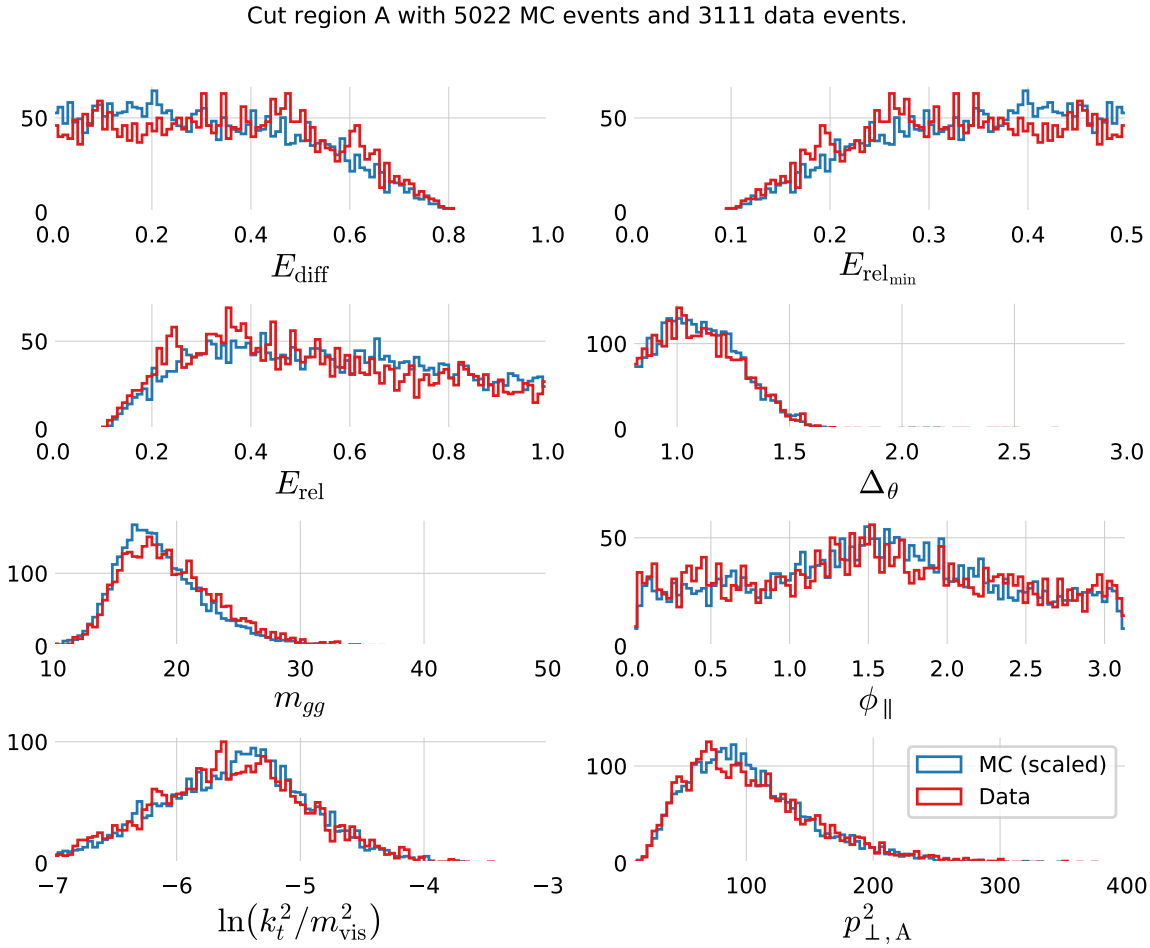


Figure B.58: Comparison of the gluon splitting distributions in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area A, see Table 5.7. The distribution for MC (scaled to Data) is shown in blue and for Data in red. These eight distributions are for the $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area A which has 5022 events in the MC sample and 3111 in the Data sample.

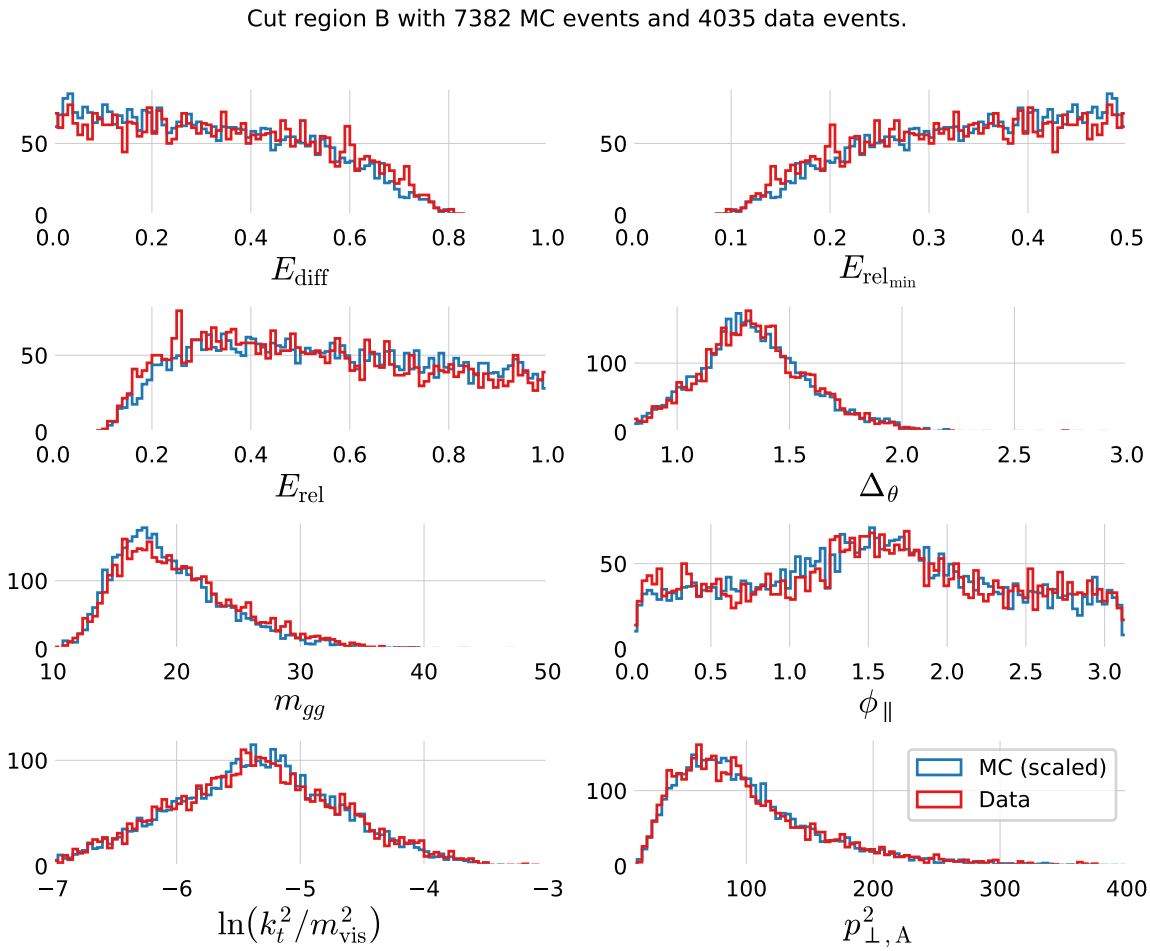


Figure B.59: Comparison of the gluon splitting distributions in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area B, see Table 5.7. The distribution for MC (scaled to Data) is shown in blue and for Data in red. These eight distributions are for the $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area B which has 7382 events in the MC sample and 4035 in the Data sample.

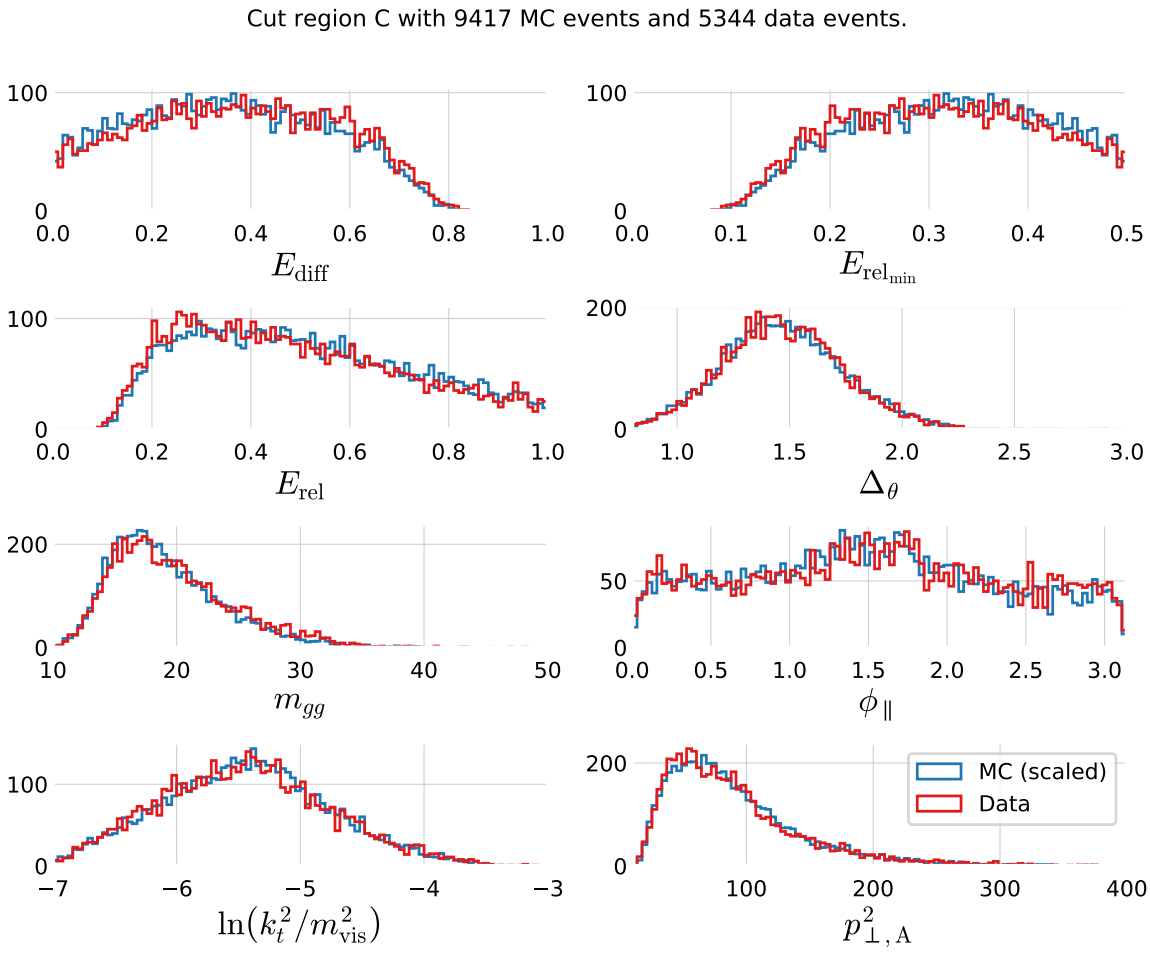


Figure B.60: Comparison of the gluon splitting distributions in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area C, see Table 5.7. The distribution for MC (scaled to Data) is shown in blue and for Data in red. These eight distributions are for the $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area C which has 9417 events in the MC sample and 5344 in the Data sample.

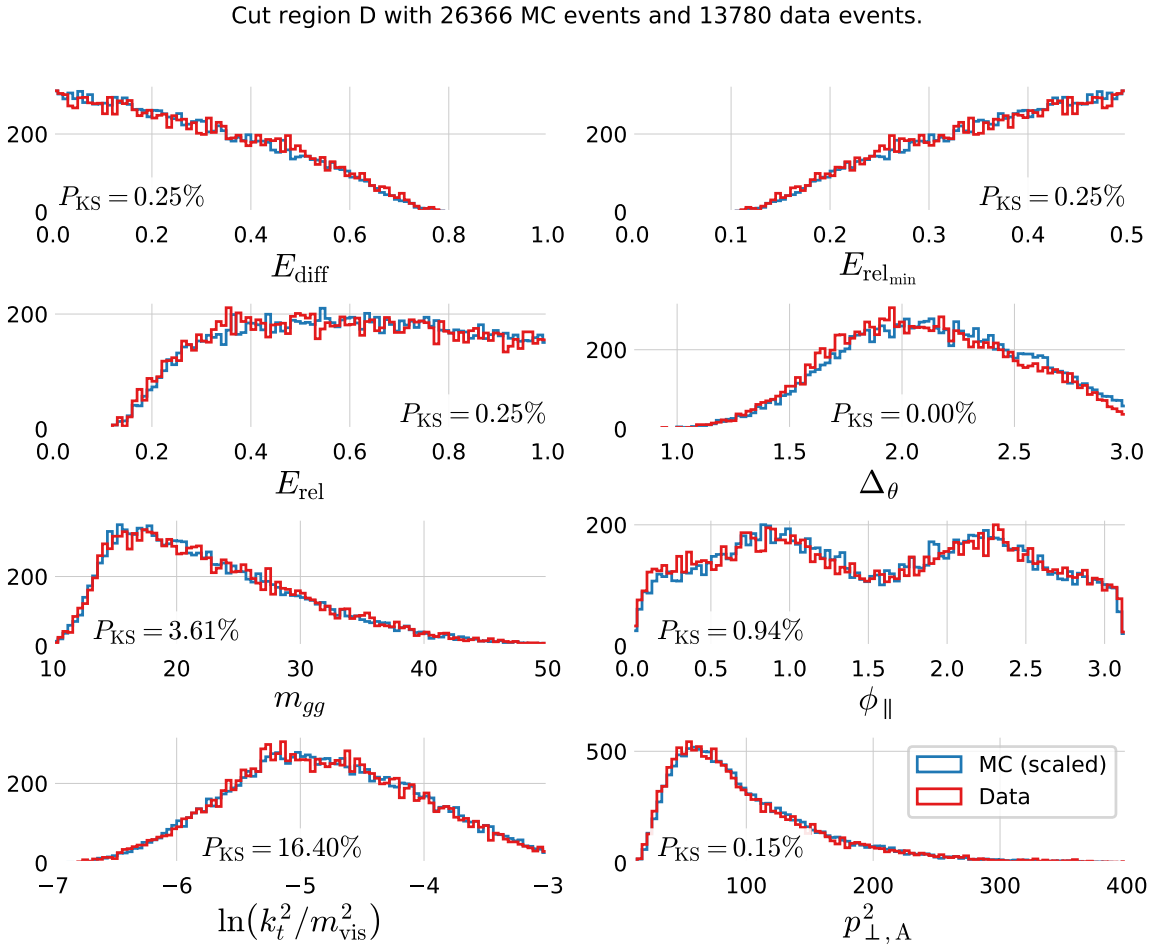


Figure B.61: Comparison of the gluon splitting distributions in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area D, see Table 5.7. The distribution for MC (scaled to Data) is shown in blue and for Data in red. These eight distributions are for the $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area D which has 26 366 events in the MC sample and 13 780 in the Data sample.

List of Figures

2.1 The learning problem. 6
2.2 Approximation-Estimation Tradeoff 10
2.3 Regularization Strength 11
2.4 Regularization Effect of L_2 12
2.5 Regularization Effect of L_1 12
2.6 k-Fold Cross Validation 13
2.7 k-Fold Cross Validation for Time Series Data 13
2.8 Objective Functions Zoom In. 15
2.9 Objective Functions. 15
2.10 Decision Tree Cuts In Feature Space 16
2.11 Decision Tree 16
2.12 Grid Search 20
2.13 Random Search 20
2.14 Bayesian Optimization 22

3.1 Danish Housing Price Index 27
3.2 Distributions for the housing price dataset 28
3.3 Distributions for the housing price dataset 29
3.4 Histogram of prices of houses and apartments sold in Denmark 30
3.5 Linear correlation between variables and price 31
3.6 Comparison of the Linear Correlation ρ and the Non-Linear MIC. 31
3.7 Non-linear correlation between variables and price 32
3.8 Validity of input features 32
3.9 Validity Dendrogram 33
3.10 Prophet Forecast for apartments 34
3.11 Prophet Trends 35
3.12 XXX 36
3.13 Parallel Coordinate Plot of the Initial Hyperparameter Optimization for Apartments 37
3.14 Initial HPO Results for the Weight Half-life $T_{\frac{1}{2}}$ 38
3.15 Initial HPO Results for the Loss Function 38
3.16 XXX 39
3.17 Hyperparameter optimization: random search results 39
3.18 Early Stopping results 40
3.19 Performance of XGB-model on apartment prices 41
3.20 Standard Deviation and MAD of the Static and Dynamic XGB Forecasts 41
3.21 Market Index based on the Static and Dynamic XGB Forecasts 42

3.22 SHAP Prediction Explanation for apartment 44
3.23 Feature importance of apartments prices using XGB 44
3.24 Feature importance of apartments prices using XGB XXX 45
3.25 Performance Comparison of Multiple Models 46
3.26 SHAP plot villa TFIDF XXX 48

4.1 The Standard Model 54
4.2 Feynman diagram for the jet production at LEP 55
4.3 Quark splitting 55
4.4 Hadronization process 56
4.5 The ALEPH detector 57
4.6 Polar angle 57
4.7 Azimuthal angle 57
4.8 Track Significance 59

5.1 Histograms of the vertex variables 65
5.2 UMAP visualization of vertex variables for 4-jet events 66
5.3 UMAP visualization of vertex variables for 3-jet events 66
5.4 UMAP visualization of vertex variables for 2-jet events 66
5.5 Correlation of Vertex Variables 67
5.6 Plot of the log-loss $\ell_{\log}$ 68
5.7 Hyperparameter Optimization of b-tagging 69
5.8 Parallel Plot of HPO Results for 4-Jet b-Tagging 69
5.9 b-Tag Scores in 4-Jet Events 70
5.10 ROC curve for 4-jet b-tagging 70
5.11 Distribution of b-Tags in 4-Jet Events 71
5.12 Global Feature Importances for the LGB b-Tagging Algorithm on 4-Jet Events 71
5.13 The expit Function 71
5.14 The logit Function 72
5.15 SHAP 3-Jet Model Explanation for b-like Jet 72
5.16 b-Tagging Efficiency $\varepsilon_b^{b\text{-sig}}$ as a Function of Jet Energy 74
5.17 b-Tagging Efficiency $\varepsilon_g^{g\text{-sig}}$ as a Function of Jet Energy 74
5.18 Hyperparameter Optimization of g-tagging 77
5.19 1D Sum Models Predictions and Signal Fraction for 4-jets events 78
5.20 g-Tag Scores in 4-Jet Events 79
5.21 ROC Curve for g-Tag in 4-Jet Events 80
5.22 Distribution of g-Tag Scores in 4-Jet Events for Signal and Background 80
5.23 Distribution of b-Tag Scores in 3-Jet l-Quark Events for Low and High g-Tag Values 80
5.24 3D Scatter Plot of β_{tag}-Values for High and Low γ_{tag} l-Quark Events 81
5.25 g-Tagging Pseudo Efficiency for $b\bar{b}g$-Events as a Function of g-Tag 82
5.26 g-Tagging Pseudo Efficiency for $b\bar{b}g$-Events as a Function of The Mean Invariant Mass 82
5.27 Generalized Angularities 83
5.28 Generalized Angularities for Charged Gluons Jets in 3-Jet Events: λ_0^2 84

5.29 Soft Wide Angle Gluons in 4-Jet Events 86
5.30 Soft Collinear Gluons in 4-Jet Events 86
5.31 Hard Non $g \to gg$ Gluons in 4-Jet Events 86
5.32 g-Tagging Efficiency for 4-Jet Events in MC as a Function of the Normalized Gluon-Gluon Jet Energy Difference Asymmetry E_{diff} 87
5.33 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for the Normalized Gluon Gluon Jet Energy Asymmetry 88
5.34 Overview of the Four Areas in the $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space 90
5.35 Gluon Splitting Distribution Comparison in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area A 91

A.1 Validity Heatmap 93
A.2 Distributions for the housing price dataset 94
A.3 Distributions for the housing price dataset 95
A.4 Distributions for the housing price dataset 96
A.5 Distributions for the housing price dataset 97
A.6 Distributions for the housing price dataset 98
A.7 Distributions for the housing price dataset 99
A.8 Distributions for the housing price dataset 100
A.9 Distributions for the housing price dataset 101
A.10Distributions for the housing price dataset 102
A.11Distributions for the housing price dataset 103
A.12Distributions for the housing price dataset 104
A.13Distributions for the housing price dataset 105
A.14Distributions for the housing price dataset 106
A.15Distributions for the housing price dataset 107
A.16Linear Correlations 109
A.17MIC non-linear correlation 110
A.18Prophet Forecast for apartments 111
A.19Prophet Trends 111
A.20Overview of initial hyperparameter optimization of the housing model for houses 115
A.21XXX 116
A.22XXX 116
A.23XXX 116
A.24XXX 117
A.25XXX 117
A.26XXX 117
A.27Performance of XGB-model on apartment prices 118

B.1 UMAP Parameter Grid Search 123
B.2 Visualization of the t-SNE algorithm 123
B.3 Parallel Plot of HPO results for 3-jet b-Tagging 124
B.4 b-tag scores in 3-jet events 124
B.5 ROC curve for 3-jet b-tagging 125
B.6 Distribution of b-Tags in 3-Jet Events 125

B.7 Global Feature Importances for the LGB b-Tagging Algorithm on 3-Jet Events 125
B.8 Parallel Plot of HPO Results for 3-Jet g-Tagging for Energy Ordered Jets 125
B.9 Parallel Plot of HPO Results for 3-Jet g-Tagging for Shuffled Jets 125
B.10 Parallel Plot of HPO Results for 4-Jet g-Tagging for Energy Ordered Jets 126
B.11 Parallel Plot of HPO Results for 4-Jet g-Tagging for Shuffled Jets 126
B.12 PermNet Architecture 126
B.13 1D LGB Model Cuts for 4-jets events 127
B.14 1D Sum Models Predictions and Signal Fraction for 3-jets events 127
B.15 1D LGB Model Cuts for 3-jets events 127
B.16 g-Tag Scores in 3-Jet Events 128
B.17 ROC curve for g-tag in 4-jet events 128
B.18 ROC Curve for g-Tag in 3-Jet Events 128
B.19 Distribution of g-Tag Scores in 3-Jet Events for Signal and Background 129
B.20 b-Tagging Efficiency $\varepsilon_b^{g\text{-sig}}$ as a Function of Jet Energy 129
B.21 b-Tagging Efficiency $\varepsilon_g^{b\text{-sig}}$ as a Function of Jet Energy 129
B.22 Generalized Angularities for Charged Gluons Jets: λ_0^2 130
B.23 Generalized Angularities for Charged Gluons Jets: $\lambda_{\frac{1}{2}}^1$ 130
B.24 Generalized Angularities for Charged Gluons Jets: λ_1^1 130
B.25 Generalized Angularities for Charged Gluons Jets: λ_1^2 131
B.26 Generalized Angularities for Charged Gluons Jets: λ_0^0 131
B.27 Generalized Angularities for Neutral Gluons Jets: λ_0^2 131
B.28 Generalized Angularities for Neutral Gluons Jets: $\lambda_{\frac{1}{2}}^1$ 132
B.29 Generalized Angularities for Neutral Gluons Jets: λ_1^1 132
B.30 Generalized Angularities for Neutral Gluons Jets: λ_1^2 132
B.31 Generalized Angularities for Neutral Gluons Jets: λ_0^0 133
B.32 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable E_{diff} 134
B.33 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable $E_{\text{rel}_{\text{min}}}$ 134
B.34 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable E_{rel} 134
B.35 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable Δ_θ 135
B.36 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable m_{gg} 135
B.37 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable $\phi_{\parallel}$ 135
B.38 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable $\ln\left(k_t^2/m_{\text{vis}}^2\right)$ 135
B.39 Relationship Between the g-Tag Value γ_{tag} and the Gluon Splitting Variable $p_{\perp,\text{A}}^2$ 136

B.40 g-Tagging Efficiency for 4-Jet Events in MC as a Function of E_{diff} 136
B.41 g-Tagging Efficiency for 4-Jet Events in MC as a Function of $E_{\text{rel}_{\text{min}}}$ 136
B.42 g-Tagging Efficiency for 4-Jet Events in MC as a Function of E_{rel} 137
B.43 g-Tagging Efficiency for 4-Jet Events in MC as a Function of Δ_θ 137
B.44 g-Tagging Efficiency for 4-Jet Events in MC as a Function of m_{gg} 137
B.45 g-Tagging Efficiency for 4-Jet Events in MC as a Function of $\phi_\parallel$ 137
B.46 g-Tagging Efficiency for 4-Jet Events in MC as a Function of $\ln\left(k_t^2/m_{\text{vis}}^2\right)$ 138
B.47 g-Tagging Efficiency for 4-Jet Events in MC as a Function of $p_{\perp,\text{A}}^2$ 138
B.48 g-Tagging Efficiency for 4-Jet Events in MC as a Function of $R_{gg}^{\text{k}_\text{t}}$ 138
B.49 g-Tagging Efficiency for 4-Jet Events in MC as a Function of R_{gg}^{CA} 138
B.50 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for E_{diff} 139
B.51 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for $E_{\text{rel}_{\text{min}}}$ 139
B.52 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for E_{rel} 140
B.53 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for Δ_θ 140
B.54 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for m_{gg} 140
B.55 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for $\phi_\parallel$ 141
B.56 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for $\ln\left(k_t^2/m_{\text{vis}}^2\right)$ 141
B.57 Closure Plot Comparing MC Truth and the Efficiency Corrected g-Tagging Model in 4-Jet Events for $p_{\perp,\text{A}}^2$ 141
B.58 Gluon Splitting Distribution Comparison in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area A 142
B.59 Gluon Splitting Distribution Comparison in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area B 143
B.60 Gluon Splitting Distribution Comparison in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area C 144
B.61 Gluon Splitting Distribution Comparison in MC and Data for $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space Area D 145

List of Tables

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). 29
3.2 Basic Cuts 33
3.3 Side Door Mapping. 33
3.4 Street Mapping 33
3.5 Number of Observations in the Housing Dataset 36
3.6 Number of Observations in the Housing Dataset for the Tight Selection 36
3.7 Results of the initial hyperparameter optimization for apartments for the best loss function ℓ_{Cauchy}. 37
3.8 Results of the initial hyperparameter optimization for houses for the best loss function ℓ_{Cauchy}. 37
3.9 PDFs Used in the Random Search 39
3.10 Realtors' MAD 41
3.11 Performance Metrics for the Housing Model on Apartments 43
3.12 Performance Metrics for the Housing Model on Houses 43

5.1 Dimensions of dataset for Data 64
5.2 Dimensions of dataset for MC and MCb 64
5.3 Number of different types of jets for MC and MCb for n-jet events. See also Table B.1 for relative numbers. 65
5.4 Random Search PDFs for LGB 69
5.5 Global SHAP Feature Importances for the g-Tagging Models in 4-Jet Events 77
5.6 Gluon Splitting Systemic Errors 89
5.7 Area Definition in the $R_{gg}^{k_t}$-R_{gg}^{CA} Phase Space 89

A.1 XXX **TODO!**. 108
A.2 Energy Rating Mapping 110
A.3 Rmse-ejerlejlighed-appendix. 112
A.4 Logcosh-ejerlejlighed-appendix. 112
A.5 Cauchy-ejerlejlighed-appendix. 112
A.6 Welsch-ejerlejlighed-appendix. 113
A.7 Fair-ejerlejlighed-appendix. 113
A.8 Rmse-villa-appendix. 113
A.9 Logcosh-villa-appendix. 113
A.10 Cauchy-villa-appendix. 114
A.11 Welsch-villa-appendix. 114

A.12Fair-villa-appendix. 114
A.13XXX ejer tight 119
A.14XXX villa tight 119

B.1 Number of different types of jets for MC and MCb written in relative numbers such that each row sum to 100 %. See also Table 5.3. 122
B.2 Random Search PDFs for XGB 124
B.3 Global SHAP Feature Importances for the g-Tagging Models in 3-Jet Events 128

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.