CHRISTIAN MICHELSEN
NIELS BOHR INSTITUTE
UNIVERSITY OF COPENHAGEN

A PHYSICIST'S APPROACH TO MACHINE LEARNING – UNDERSTANDING THE BASIC BRICKS

SUPERVISOR:
TROELS PETERSEN
NIELS BOHR INSTITUTE
UNIVERSITY OF COPENHAGEN

Christian Michelsen

https://github.com/christianmichelsen

First printing, November 2019

Contents

1 *Abstract* 1

2 *Introduction* 3

3 *Machine Learning Theory* 5

3.1 *Statistical Learning Theory* 5

3.2 *Supervised Learning* 6

3.3 *Generalization Bound* 7

3.3.1 *Generalization Bound for infinite hypotheses* 9

3.4 *Avoiding overfitting* 10

3.4.1 *Model Regularization* 10

3.4.2 *Cross Validation* 12

3.4.3 *Early Stopping* 14

3.5 *Loss functions* 14

3.5.1 *Evaluation Function* 15

3.6 *Decision Trees* 16

3.6.1 *Ensembles of Decision Trees* 17

3.7 *Hyperparamater Optimization* 19

3.7.1 *Grid Search* 20

3.7.2 *Random Search* 20

3.7.3 *Bayesian Optimization* 21

3.8 *Feature Importance* 22

4 *Housing Prices Analysis* 27

4.1 *Headings* 28

5 *Particle Physics and LEP* 37
5.1 *Page Layout* 37
5.1.1 *Headings* 37

6 *Quark Gluon Analysis* 39
6.1 *Sidenotes* 39
6.2 *References* 46
6.3 *Figures and Tables* 46
6.4 *Captions* 48
6.5 *Full-width text blocks* 49
6.6 *Typography* 49
6.6.1 *Typefaces* 49
6.6.2 *Letterspacing* 49
6.7 *Document Class Options* 50

7 *Discussion and Outlook* 53
7.1 *File Hooks* 53

8 *Conclusion* 55
8.1 *Tufte-LaTeX Website* 55
8.2 *Tufte-LaTeX Mailing Lists* 55
8.3 *Getting Help* 55
8.4 *Errors, Warnings, and Informational Messages* 55
8.5 *Package Dependencies* 56

Index 63

List of Figures

3.1 Schematic overview of the learning problem. 6
3.2 Approximation-Estimation tradeoff 10
3.3 Regularization Effect 11
3.4 Regularization Effect of L_2 12
3.5 Regularization Effect of L_1 12
3.6 k-Fold Cross Validation 13
3.7 k-Fold Cross Validation for Time Series Data 13
3.8 Comparison of different objective functions zoom in. 15
3.9 Comparison of different objective functions. 16
3.10 Decision Tree Cuts 16
3.11 Decision Tree Illustration 16
3.12 Grid Search 20
3.13 Random Search 21
3.14 Bayesian Optimization 22

4.1 Geographic overview of square meter price in Denmark 27
4.2 Histogram of prices of houses and apartments sold in Denmark 27
4.3 Input paramater distributions for the housing prices dataset 28
4.4 Percentage of valid counts for each variable 28
4.5 Linear correlation between variables and price 28
4.6 Non-linear correlation between variables and price 29
4.7 Registration of property 29
4.8 Prophet Forecast for apartments 29
4.9 Prophet Trends 29
4.10 Overview of initial hyperparamater optimization of the housing model for apartments 30
4.11 Performance of LGB-model on apartment prices 30
4.12 Feature importance of apartments prices using LGB 31
4.13 Total feature importance of apartment prices using LGB 32
4.14 SHAP Prediction Explanation for apartment 32
4.15 Hyperparameter optimization: random search results 32
4.16 Hyperparameter optimization: Bayesian optimization results 32
4.17 Early Stopping results 33
4.18 Comparison of different objective functions 33
4.19 Comparison of different half times weights 33
4.20 2018 LGB Forecast 34

5.1 Feynman diagram for the jet production at LEP 37

6.1 Histograms of the vertex variables 39
6.2 UMAP vizualisation of vertex variables 40
6.3 b-tag scores in 3-jet events 40
6.4 ROC curve for b-tag in 4-jet events 40
6.5 g-tag scores in 4-jet events 41
6.6 g-tag scores in 4-jet events for signal and background 41
6.7 ROC curve for g-tag in 4-jet events 41
6.8 1D Sum Model Cuts for 4-jets 42
6.9 1D Sum Models Predictions and Signal Fraction for 4-jets 42
6.10 Hyperparameter Optimization of b- and g-tagging 42
6.11 Overview of Hyperparamaters of g-tagging for 3-jet shuffled events 42
6.12 SHAP Prediction Explanation for b-like jet 43
6.13 Monte Carlo – Data bias for b-tags and jet energy 43
6.14 b-Tagging Efficiency $\varepsilon_b^{b-\text{sig}}$ as a function of jet energy 43
6.15 b-Tagging Efficiency $\varepsilon_b^{g-\text{sig}}$ as a function of jet energy 43
6.16 b-Tagging Efficiency $\varepsilon_g^{g-\text{sig}}$ as a function of jet energy 44
6.17 b-Tagging Efficiency $\varepsilon_g^{b-\text{sig}}$ as a function of jet energy 44
6.18 g-Tagging proxy efficiency for $b\bar{b}g$-events as function of the mean invariant mass 44
6.19 g-Tagging proxy efficiency for $b\bar{b}g$-events as function of g-tag 44
6.20 g-Tagging efficiency for 4-jet events in MC as a function of normalized gluon gluon jet energy difference 45
6.21 Closure plot between MC Truth and the corrected g-tagging model in 4-jet events for the normalized gluon gluon jet energy difference 45
6.22 R kt CA overview XXX **TODO!** 45
6.23 R kt CA cut region A XXX **TODO!** 45
6.24 This is a margin figure. The helix is defined by $x = \cos(2\pi z)$, $y = \sin(2\pi z)$, and $z = [0, 2.7]$. The figure was drawn using Asymptote (`http://asymptote.sf.net/`). 46
6.25 This graph shows $y = \sin x$ from about $x = [-10, 10]$. *Notice that this figure takes up the full page width.* 47
6.26 Hilbert curves of various degrees n. 47

List of Tables

4.1 RMSE. 30
4.2 LogCosh. 30
4.3 Heading styles used in *Beautiful Evidence*. 35

6.1 Here are the dimensions of the various margins used in the Tufte-handout class. 47

1. Abstract

This sample book discusses the design of Edward Tufte's books and the use of the `tufte-book` and `tufte-handout` document classes.

2. 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, and the aim of this section is to provide an initial overview of the scope and relationship of the two sub-projects; two sub-projects which are covered in each part of this book.

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 [7, 13]) 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 the continuos process of providing data but it also 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.

The second part, the Quark-Gluon discrimination in particle

The background for this masters's thesis, is that it is part of a so-called 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. Here ancient DNA is sampled and analysed with the hope of finding patterns, structure, in the genome which 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.

[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.

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. It is especially the distributions of these gluon jets and the difference between Data[3] and MC that are of interests to the theoreticians that develop the MC-models. At first an improved b-tagging algorithm was developed based on only the vertex variables since the primary variables of interest to the theoreticians are the shape variables. 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 (hopefully) 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 chapter 3 introduces the needed theoretical Machine Learning (ML) background needed for understanding the methods used throughout the thesis, chapter chapter 4 describes the housing prices part of the project as mentioned above, chapter chapter 5 introduces the basic physics in the standard model and the Lund string model which is used throughout the rest of the theses, chapter chapter 6 explains analysis of the main project in this thesis, i.e. the quark gluon analysis, and finally the two chapters chapter 7 and chapter 8 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 different instances of same underlying problem: teaching computers how to find patterns in high-dimensional data automatically and should thus not be seen as two independent projects. This also highlight 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.

3. 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 set of subject 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 [23] which has seen a plethora of use cases in recent years.

3.1 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 [6] 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 subfields 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

3.2 *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 $\mathbf{x}_i = [x_1, x_2, \ldots, x_M] \in \mathcal{X}$ 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 sample or 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 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 $y \in \mathcal{Y}$ for each observation $\mathbf{x} \in \mathcal{X}$. 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 candidate hypotheses $\mathcal{H} = \{h_1, h_2, \ldots, h_M\}$, and hopefully h^* will be a good approximation of f, $h^* \approx f$. A schematic overview of this process can be seen in Figure 3.1.

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 best as possible.

To quantify the statement "as best 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 [24] and the expected loss[5]:

$$L(h) = \mathbb{E}\left[\ell(h(\mathbf{x}), y)\right] = \int \ell(h(\mathbf{x}), y)\, dP(\mathbf{x}, y). \tag{3.1}$$

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 $\mathcal{D}$:

$$\hat{L}(h, S) = \frac{1}{N} \sum_{i=1}^{N} \ell(h(\mathbf{x}_i), y_i), \tag{3.2}$$

which is an approximation of $L(h)$ based on the training data avail-

[3] If $\mathcal{Y} \in \mathbb{Z}$ then it would be a multi-class classification problem.

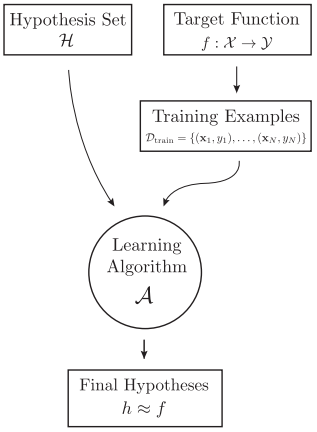

Figure 3.1: Schematic overview of the learning problem and how to find the optimal hypothesis h^* to approximate f given the training data $\mathcal{D}_{\text{train}}$.

[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 assumtions where PAC is an abbreviation for Probably, Approximatelly Correct.

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

[6] Also called empirical error.

able. Now the optimal hypothesis h^* can defined:

$$h^* = \arg\min_{h \in \mathcal{H}} \hat{L}(h, S). \tag{3.3}$$

3.3 *Generalization Bound*

In section 3.2 the method of selecting the optimal hypothesis 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:

$$G(h) = \hat{L}(h, S) - L(h). \tag{3.4}$$

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):*

$$\mathbb{P}\left\{\bigcup_{i \leq 1} E_i\right\} \leq \sum_{i \leq 1} \mathbb{P}\left\{E_i\right\}. \tag{3.5}$$

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 $\epsilon > 0$:*

$$\mathbb{P}\left\{\frac{1}{N}\sum_{i=1}^{N} Z_i - \mu \geq \epsilon\right\} \leq e^{-2n\epsilon^2} \tag{3.6}$$

and

$$\mathbb{P}\left\{\mu - \frac{1}{N}\sum_{i=1}^{N} Z_i \geq \epsilon\right\} \leq e^{-2n\epsilon^2}. \tag{3.7}$$

When using the union bound on equation (3.6) and equation (3.7) 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 $\epsilon > 0$:*

$$\mathbb{P}\left\{\left|\frac{1}{N}\sum_{i=1}^{n} Z_i - \mu\right| \geq \epsilon\right\} \equiv \mathbb{P}\left\{|\hat{\mu} - \mu| \geq \epsilon\right\} \leq 2e^{-2N\epsilon^2}, \tag{3.8}$$

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 (3.2) we see that $\hat{\mu} = \hat{L}(h, S)$, and similar for equation (3.1): $\mu = L(h)$. We then see that the generalization error is bounded:

$$\mathbb{P}\left\{|G(h)| \geq \epsilon\right\} = \mathbb{P}\left\{\left|\hat{L}(h,S) - L(h)\right| \geq \epsilon\right\} \leq 2e^{-2N\epsilon^2}. \tag{3.9}$$

This equation provides a bound on the difference between the empirical loss and the expected loss. Say the probability of the generalization error being larger than $\epsilon = 0.1$ is needed for $N = 100$ samples, we find that the this probability is $P = 27\%$. The generalization bound can be rewritten in terms of δ:

$$\delta = 2e^{-2N\epsilon^2} \in (0,1) \quad \Rightarrow \quad \epsilon = \sqrt{\frac{\ln \frac{2}{\delta}}{2N}} \quad \Rightarrow \tag{3.10}$$

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:*

$$\mathbb{P}\left\{\left|\hat{L}(h,S) - L(h)\right| \geq \sqrt{\frac{\ln \frac{2}{\delta}}{2N}}\right\} \leq \delta. \tag{3.11}$$

Equation (3.11) can be read as the probability of the generalization error being larger than $\sqrt{\frac{\ln \frac{2}{\delta}}{2N}}$ is δ or similarly that with probability greater than $1 - \delta$:

$$\left|\hat{L}(h,S) - L(h)\right| \leq \sqrt{\frac{\ln \frac{2}{\delta}}{2N}}. \tag{3.12}$$

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, however, inversely[8] related to on the certainty[9] δ of this bound.

There is a big assumption of this derivation although: 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 M hypotheses: $|\mathcal{H}| = M$. We thus have $[h_1, h_2, \ldots, h_M]$ 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 $|\mathcal{H}| = M$. Then for any $\delta \in (0,1)$ we have:*

$$\mathbb{P}\left\{\exists h \in \mathcal{H} : \left|\hat{L}(h,S) - L(h)\right| \geq \sqrt{\frac{\ln \frac{2M}{\delta}}{2N}}\right\} \leq \delta. \tag{3.13}$$

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

[8] Not to be understood as $1/x$ in this context.

[9] Technically the certainty of the model is $1 - \delta$.

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) followed by applying Hoeffding's inequality to each part in the sum (the second inequality):

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

By making the substitution $\delta = M\delta'$ we arrive at equation (3.13). □

As we did in equation (3.12), equation (3.13) can also be read as with probability greater than $1 - \delta$ then for all $h \in \mathcal{H}$:

$$\left|\hat{L}(h, S) - L(h)\right| \leq \sqrt{\frac{\ln \frac{2M}{\delta}}{2N}}. \tag{3.14}$$

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

3.3.1 *Generalization Bound for infinite hypotheses*

Section 3.3 dealt with the case of a single hypotheses h and a finite set of candidate hypotheses $h \in \mathcal{H}, |\mathcal{H}| = M$. When M goes towards ∞ the generalization bound goes to ∞ and the bound becomes useless. However, even simple models such as a linear classifier[10] that predicts $\hat{y} = 1$ when the dot product $\mathbf{w}^T\mathbf{x}$ is positive and $\hat{y} = 0$ when it is negative has $|\mathcal{H}| = \infty$. Since there an infinite number of hypotheses $h(\mathbf{w})$, 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[11] 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 d-dimensional linear classifier defined above[12] is $d_{\text{VC}} = d$ for $\{\mathbf{x}, \mathbf{w}\} \in \mathbb{R}^d$.

[10] Also known as the perceptron. Often includes a constant offset b as well which is omitted for brevity.

[11] For proof, see: [6]

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

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 $1-\delta$:*

$$L(h) \leq \hat{L}(h,S) + \sqrt{\frac{8}{N}\ln\left(\frac{4}{\delta}\left((2N)^{d_{\text{VC}}}+1\right)\right)}. \tag{3.15}$$

Equation (3.15) 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 $\Omega(N,\mathcal{H},\delta)$:

$$\Omega(N,\mathcal{H},\delta) = \sqrt{\frac{8}{N}\ln\left(\frac{4}{\delta}\left((2N)^{d_{\text{VC}}(\mathcal{H})}+1\right)\right)}. \tag{3.16}$$

As the hypothesis space complexity d_{VC} grows, the generalization bound loosens 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*[13], when the model is so complex that it starts fitting the inherent noise in the data it is called *overfitting*[14]. The loss as a function of model complexity gives the characteristic curve illustrated in Figure 3.2. 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.

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

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

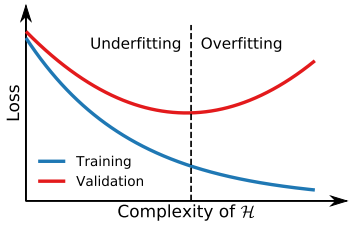

Figure 3.2: Illustration of the empirical loss as a function of model complexity. The training error is shown in blue and validation error in red.

3.4 *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 not complementary of each other but can be taken advantage of in a combination. In this section model regularization will be introduced in subsection 3.4.1, cross validation in subsection 3.4.2, and early stopping in subsection 3.4.3.

3.4.1 *Model Regularization*

One of the earliest methods developed for preventing overfitting was model regularization. A. N. Tikhonov [22] 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:

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{\text{LS}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y}-\mathbf{f}(\mathbf{X})\|_2^2 \\ \mathbf{f}(\mathbf{X}) = \mathbf{X}\boldsymbol{\beta},\end{aligned} \tag{3.17}$$

where $\mathbf{y}$ is vector of values we are trying to predict[15], $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the matrix of input parameters such that each observation $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional vector stacked in $\mathbf{X}$ as rows with N total observations, $\hat{\boldsymbol{\beta}}_{\text{LS}}$ is the vector of unknown coefficients[16] 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:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p}. \tag{3.18}$$

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[17] yields the solution for $\boldsymbol{\beta}$:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = 0 \Rightarrow \\ \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \Rightarrow \\ \hat{\boldsymbol{\beta}}_{\text{LS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{y}. \end{aligned} \tag{3.19}$$

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 [15]. 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:

$$\hat{\boldsymbol{\beta}}_{\text{L}_2} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{y} - \mathbf{f}(\mathbf{X})\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\}, \tag{3.20}$$

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[18] the corresponding solution for $\boldsymbol{\beta}$ is:

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y}, \tag{3.21}$$

where $\mathbf{I}$ is the identity matrix[19]. This extra term, $\Omega = \lambda\|\boldsymbol{\beta}\|_2^2$, acts as a shrinkage factor on the coefficients of $\boldsymbol{\beta}$. Looking at equation (3.20), we see that this is the Lagrangian form of the equivalent problem:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{\text{L}_2} &= \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{f}(\mathbf{X})\|_2^2 \\ \text{subject to :} \quad &\sum_{i=1}^{N} \beta_i^2 = \|\boldsymbol{\beta}\|_2^2 \leq t, \end{aligned} \tag{3.22}$$

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 3.3.

Here $N = 9$ datapoints were randomly generated such that the x-values are evenly spaced from 0 to 1 and $y \sim \mathcal{N}(\mu = 0, \sigma = 10)$.

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

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

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

[18] 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 $\boldsymbol{\beta}$

[19] The $\lambda\mathbf{I}$ in equation (3.21) 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

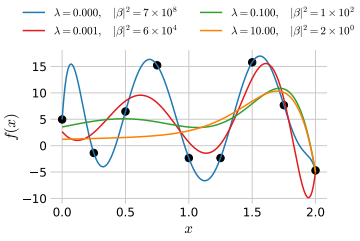

Figure 3.3: Effect of tuning the regularization strength λ in ridge regression.

They were then fit with a 9-order polynomial by minimizing equation (3.20) for different values of λ. Here we see the regularizing effect of λ, going from $\lambda = 0$ in blue which fits all points[20] with a high degree of variance and a wildly oscillatory pattern to $\lambda = 10$ 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 3.3.1, 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.

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

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

$$\Omega = \lambda \|\boldsymbol{\beta}\|_1, \tag{3.23}$$

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 [21] 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[21] solutions. An illustration of this can be seen in Figure 3.4 and Figure 3.5 where the constraint regions of $\boldsymbol{\beta}$ os 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 $p \leq 1$.

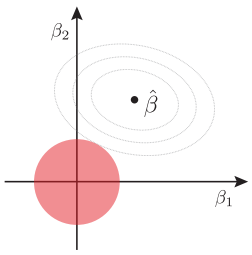

Figure 3.4: Sketch of the minimization problem defined in equation (3.22), i.e. for a L_2-penalty. The constrain region shown in red is defined as $\beta_1^2 + \beta_2^2 \leq t$ for L_2 in $2D$-space and the contours of the unconstrained solution is shown with grey, dashed lines.

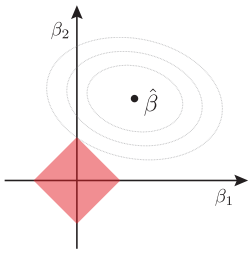

Figure 3.5: Sketch of the similar minimization problem defined in Figure 3.4 for the L_1-penalty. The constrain region shown in red is defined as $|\beta_1| + |\beta_2| \leq t$ for L_1 in $2D$-space and the contours of the unconstrained solution is shown with grey, dashed lines.

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

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:

$$\mathcal{L}(h) = \hat{L}(\ell, h, S) + \Omega(h) \tag{3.24}$$

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 is discussed in subsection 3.4.2 and the choice of the training loss function ℓ in section 3.5.

3.4.2 *Cross Validation*

In general we want to be able to estimate the performance[22] of the developed model. Since evaluation 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

[22] "Performance" is used here as the word for the general metric to be optimized for, no matter whether or not this metric should be maximized or minimized.

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 and 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 it that we are not fully utilizing a lot of the data in this way. Basically20 % 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 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 are 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 [15]. For an illustration of this, see Figure 3.6. 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[23]. 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 ML is $k = 5$ which is also what is used in this project.

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 it is evaluated on the performance of samples after the event, e.g. houses sold in 2018. For an illustration of this, see Figure 3.7.

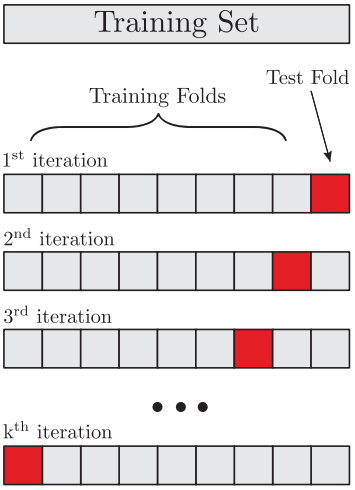

Figure 3.6: k-fold cross validation.

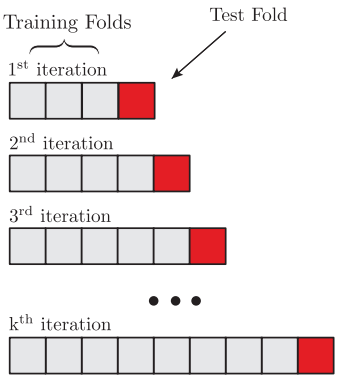

Figure 3.7: k-fold cross validation for time series data.

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

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

This is a prime 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 3.2 was just an illustration of the bias variance tradeoff, it is also something that is seen in real data and can taken advantage of through *early stopping*. Early stopping is the process of monitoring the loss for training set and validation set $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$. As mentioned in subsection 3.4.2, 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 just because of a single outlier due to noise that 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.

3.5 *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 we a dealing with a classification problem or a regression problem. 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 an 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? The first case was a factor of 2 off, whereas the second was only 25% off. The first case underestimated the price whereas the second overestimated.

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, is problem-specific, and has to be defined manually. The choice of loss function is what is called a *hyper parameter*, the optimization of which is further discussed in section 3.7. The most common choice of loss function is by far the Squared Error (SE):

$$\ell_{\text{SE}}(y, \hat{y}) = (y - \hat{y})^2 , \tag{3.25}$$

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:

$$\ell_{\text{AE}}(y, \hat{y}) = |y - \hat{y}|. \tag{3.26}$$

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 [8]. Six different loss functions (for regression problems) have been investigated in this project. In addition to SE and AE also the LogCosh, Cauchy [8], Welsch [8] and Fair [2] loss functions are used:

$$\begin{aligned}
\ell_{\text{LogCosh}}(y, \hat{y}) &= \log\left(\cosh\left(y - \hat{y}\right)\right) \\
\ell_{\text{Cauchy}}(y, \hat{y}) &= \log\left(\frac{1}{2}\left(\frac{y - \hat{y}}{c}\right)^2 + 1\right) \\
\ell_{\text{Welsch}}(y, \hat{y}) &= 1 - \exp\left(-\frac{1}{2}\left(\frac{y - \hat{y}}{c}\right)^2\right) \\
\ell_{\text{Fair}}(y, \hat{y}) &= c^2\left(\frac{|y - \hat{y}|}{c} - \log\left(\frac{|y - \hat{y}|}{c} + 1\right)\right).
\end{aligned} \tag{3.27}$$

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 3.9. 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 1 when $y - \hat{y}$ goes towards ∞, whereas it goes to 0 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 hyper parameters.

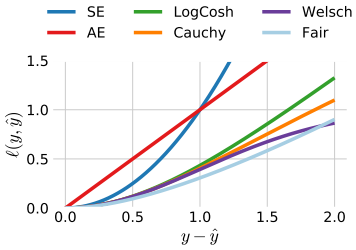

Figure 3.8: Zoom in of Figure 3.9.

3.5.1 *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

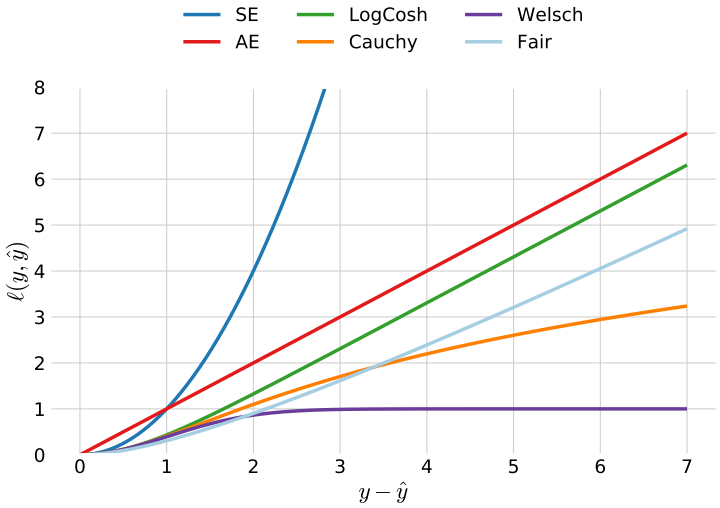

Figure 3.9: Comparison of the the six loss functions SE, AE, LogCosh, Cauchy, Welsch, and Fair as a function of $y - \hat{y}$. In the plot SE is shown in blue, AE in red, LogCosh in green, Cauchy in orange, Welsch in purple, and Fair in light blue. For the Cauchy, Welsch, and Fair functions c is set to 1. For a zoom in of the inner region where $y - \hat{y} < 2$ see Figure 3.8. All six graphs are symmetric in $y - \hat{y}$ which is why they are only shown for positive values of $y - \hat{y}$.

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.

For the loss functions defined in section 3.5 the objective function is based on the mean of the individual losses:

$$\mathcal{L} = \frac{1}{N} \sum^{N} \ell(y_i, \hat{y}_i) + \Omega. \tag{3.28}$$

3.6 *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 [15]. A simple example of this can be seen in Figure 3.10 and Figure 3.11. 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 the first "box" is called the *root* of the tree,

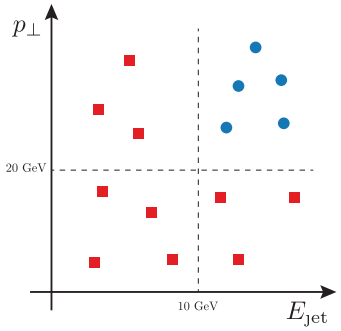

Figure 3.10: Illustration of the cuts a decision tree model make for signal in blue circles and background in red squares. This is an visualization in the feature space of the decision tree seen in Figure 3.11.

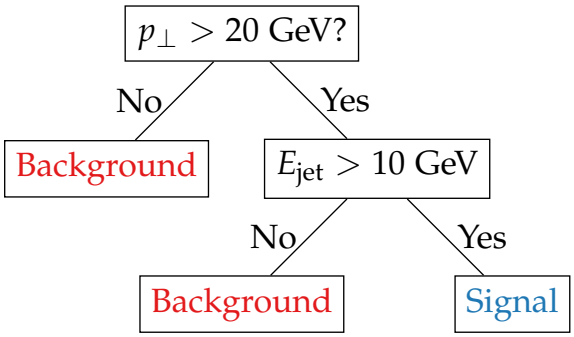

Figure 3.11: Illustration of a simple decision tree. Here the tree partitions the input feature space consisting of the two variables $p_\perp$ and E_{jet} into two categories; either signal or background. A visualization of the cuts in the feature space can be seen in Figure 3.10

any subsequent boxes are *nodes* except the final ones that are not split any further which are *leaves*.

At first the DT partitions the space according the the value of the transverse momentum[24] $p_\perp$: 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 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[25], 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. [15].

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

[25] With a solution to this problem given in subsection 3.6.1

3.6.1 *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 finite 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 variance.

Random Forests

Random Forests were first introduced in 2001 by Breiman [10]. Random Forests are a collection of B decision trees where each tree is trained on bootstrapped versions of the training data and the average[26] the individual trees' predictions T_b:

[26] In case of classification, it is the majority vote which is taken

$$f_{\text{RF}}(\mathbf{x}) = \hat{y}_{\text{RF}} = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x}). \tag{3.29}$$

The method of making artificial extra samples and training on them is in general called bootstrap aggregation or *bagging*[15]. 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:*

$$\text{Var}(\bar{\mu}_B) = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2, \tag{3.30}$$

where $\bar{\mu}_B$ is the average of the i.d. variables [15].

Equation (3.30) 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 a only subset of the observations (rows), but also only a subset of the variables (columns)[27]. This called columns subsampling (in contrast to row subsampling).

[27] 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 [25] 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.

[28] XXX **TODO!**

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 [12] revolutionized the ML world by winning numerous Kaggle[28] 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 [15]. In an iterative fashion this combines many weak learners into a single strong learner. The final prediction is thus a weighted sum over the M different weak learners F_m:

$$F(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m F_m(\mathbf{x}) \tag{3.31}$$

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 $F_m(x) = \hat{y}$. In boosting we want the next iteration to be a better model than the previous one, so image 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:

$$F_{m+1}(x) = F_m(x) + h(x) = y \Leftrightarrow h(x) = y - F_m(x). \tag{3.32}$$

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: $L(y, F_m(x)) = \frac{1}{2}(y - F_m(x))^2$. The gradient of the loss w.r.t. to $F_m(x)$ is:

$$\frac{\partial L(y, F_m(x))}{\partial F_m(x)} = F_m(x) - y. \tag{3.33}$$

This is exactly the negative of the r.h.s. of equation (3.32). 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:

$$h(x) = -\frac{\partial L}{\partial F_m}. \tag{3.34}$$

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

$$F_{m+1}(x) = F_m(x) + h(x) = F_m - \frac{\partial L}{\partial F_m}. \tag{3.35}$$

AdaBoost [14] 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[29] 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 [12] 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. In 2017 Microsoft released the competitor to XGBoost called LightGBM [16]. In comparison to XGBoost, LightGBM implements some extra binning and categorical assumptions that greatly speeds up the fitting process.

[29] Exponential loss for classification

3.7 *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 internal optimization process. This could be 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 N hyperparameters. Each of these hyperparameters have a domain Λ_n. The domain can be either real numbers $\Lambda_n \in \mathbb{R}$, integers $\Lambda_n \in \mathbb{Z}$, binary $\Lambda_n \in \{0, 1\}$, or categorical[30]. We define the hyperparameter configuration space as: $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_N$. 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.)

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

The first naive approach would simply to manually[31] try out different combinations of $\boldsymbol{\lambda}$ and see performance on the validation

[31] ALso known as jokingly as "Grad Student Descent"

set[32]. 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 3.7.1 the HPO method called Grid Search is introduced which is further generalized and optimized in subsection 3.7.2 with Random Search. Both these methods are easily parallelizable since they do not have any inherent history in its guesses. This is in contrary to Bayesian Optimization introduced in subsection 3.7.3 which allows for "smart" guesses.

[32] Remember only to use the test set on the final model!

3.7.1 *Grid Search*

Grid Seach (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 12 combinations of these two sets:

$$\{(1,1),(1,2),\ldots,(x_i,y_i),\ldots,(3,4)\}, \tag{3.36}$$

as visualized in Figure 3.12. The advantage of GS is that it is an exhaustive search over all combinations[33] of hyperparameters, however, the total number of combinations grows exponentially and GS as a method thus suffers the curse of dimensionality[34].

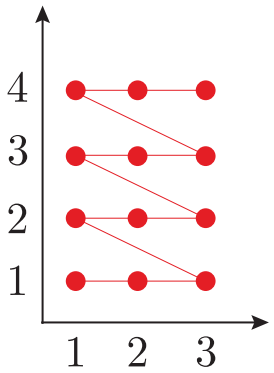

Figure 3.12: Visualization of grid search run on the two hyperparameters x and why y with the domains $x = \{1,2,3\}$ and $y = \{1,2,3,4\}$.

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

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

3.7.2 *Random Search*

To circumvent the problems of grid search, Bergstra and Bengio [9] 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."* [9]. 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:

$$\boldsymbol{\lambda}_i \sim \sum_{j=1}^{N} \mathrm{PDF}_j(\Lambda_j) \cdot \hat{\boldsymbol{e}}_j. \tag{3.37}$$

Equation (3.37) 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:

$$\boldsymbol{\lambda}_i \sim = \begin{bmatrix} \mathcal{N}(100,4) \\ \mathcal{U}(0,1) \end{bmatrix}, \tag{3.38}$$

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[35], but

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

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* [9]. 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 an effective dimensionality of 1. Then $f(x, y) = g(x) + h(y) \approx g(x)$. For a visualization of this example, see Figure 3.13.

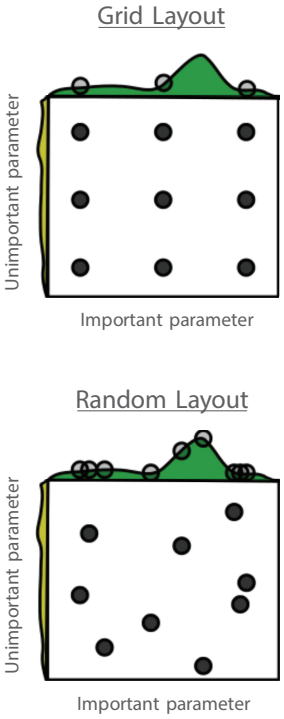

Figure 3.13: Visualization of the difference between grid search and random search. Taken from Bergstra and Bengio [9]

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 the 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 [9].

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

3.7.3 *Bayesian Optimization*

When optimizing the hyperparameters 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[36] 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 contrary to both GS and RS where each new set of hyperparameters $\boldsymbol{\lambda}$ is independent of the value of the evaluation performance.

[36] With respect to time

In Bayesian Optimization (BO) [11], the (unknown) evaluation 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 function 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 3.14. 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 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 $\boldsymbol{\lambda}$ 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 $\boldsymbol{\lambda}$ is slighty 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 $\boldsymbol{\lambda}$ according the acquisition function given the GPs.

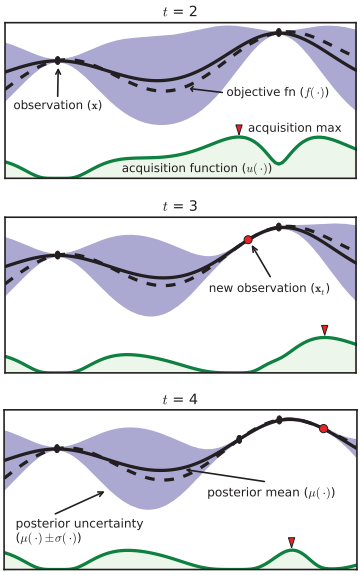

Figure 3.14: XXX **TODO!**. Taken from Brochu et al. [11]

Gaussian Processes provide a posterior distribution given some prior distribution and the datadependent ikelihood. 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. [11]. 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 3.14 described above. The acquisition function used in this project is the Upper Confidence Bound (UCB):

$$\text{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \tag{3.39}$$

where $\kappa \geq 0$ is the parameter[37] 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[38], and parallelization is non-trivial to implement since it by definition is a sequential process. The performance boost is also not guaranteed.

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

[38] Which are basically plug-and-play with Scikit-Learn [19].

3.8 *Feature Importance*

Having first established in section 3.2 that machine learning algorithms are indeed able to not only learn from data but also to generalize well without overfitting (section 3.4), modern ML algorithms such as decision trees, random forests and boosted decision

trees were introduced in section 3.6 and they were hyperparameter optimized in section 3.7, 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 the focus will be on the so-called *SHAP* values.

In 2017 Lundberg and Lee [17] showed that six different previously used methods were all specific instances of a universal underlying method[39] and propose SHaplay Additive exPlanation (SHAP) values as a unified measure of feature importance. In 2018-2019 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* [18]. 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* feature importance are both consistent feature importance measure, however, only SHAP allows for individualized[40], or local, feature importances.

[39] The class of *additive feature attribution methods* [17]

[40] 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:

$$g(z') = \phi_0 + \sum_{i=1}^{N} \phi_i z'_i. \tag{3.40}$$

Here the $\phi_i \in \mathbb{R}$'s 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 [20]. 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:*

$$f(x) = g(z') = \phi_0 + \sum_{i=1}^{N} \phi_i z'_i. \tag{3.41}$$

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:*

$$z'_i = 0 \Rightarrow \phi_i = 0. \tag{3.42}$$

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 [17] show that the only solution to equation (3.40) is:

$$\phi_i = \sum_{S \subseteq \tilde{M} \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} \left[f_x(S \cup \{i\}) - f_x(S) \right], \tag{3.43}$$

which can be simplified to:

$$\begin{aligned} \phi_i &= \sum_{S \subseteq \tilde{M} \setminus \{i\}} k(S) \cdot \Delta_{f_x}(S) \\ k(S) &\equiv \frac{|S|!(M - |S| - 1)!}{M!} \\ \Delta_{f_x}(S) &\equiv \left[f_x(S \cup \{i\}) - f_x(S) \right]. \end{aligned} \tag{3.44}$$

In equation (3.43) $\tilde{M}$ is the *set* of all input features[41], $S \subseteq \tilde{N} \setminus \{i\}$ means a subset S of $\tilde{N}$ 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 (3.44) 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 k.

[41] compared to $M = |\tilde{M}|$ 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 3.11. Here $\tilde{N}$ would be $\tilde{N} = \{p_\perp, E_{\text{jet}}\}$. For the feature $i = p_\perp$ one would thus have:

$$\begin{aligned} \phi_{p_|} &= \sum_{S \subseteq \tilde{M} \setminus \{p_\perp\}} k(S) \cdot \Delta_{f_x}(S) \\ &= \sum_{S \in [\{\}, \{E_{\text{jet}}\}]} k(S) \cdot \Delta_{f_x}(S) \\ &= \frac{0!(2 - 0 - 1)!}{2!} [f_x(\{p_\perp\}) - f_x(\{\})] \\ &\quad + \frac{1!(2 - 1 - 1)!}{2!} [f_x(\{E_{\text{jet}}, p_\perp\}) - f_x(\{E_{\text{jet}}\})]. \end{aligned} \tag{3.45}$$

Whereas $k(S)$ 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. [18] did was to develop an efficient algorithm that could solve this for trees in polynomial time[42].

[42] 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 te 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 of with M features, one gets the matrix $\mathbf{\Phi}$. When summing the over the absolute value of each column, one gets the global impact ϕ_i^{tot} of that feature [18]:

$$\phi_i^{\text{tot}} = \sum_{j=1}^{N} |\mathbf{\Phi}_{i,j}|. \tag{3.46}$$

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[43] to an already existing feature, the feature importance of the previous feature will decrease [44]

[43] Not necessarily linearly correlated

[44] This is due to the axiom of symmetry:

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

$$f_x(S \cup \{i\}) = f_x(S \cup \{j\}) \tag{3.47}$$

then $\phi_i = \phi_j$.

It can be shown that axiom 4 is implied by axiom 3 [17, Supp. Material]. Two identical features, as seen by the model, will thus "share" the feature importance.

4. Housing Prices Analysis

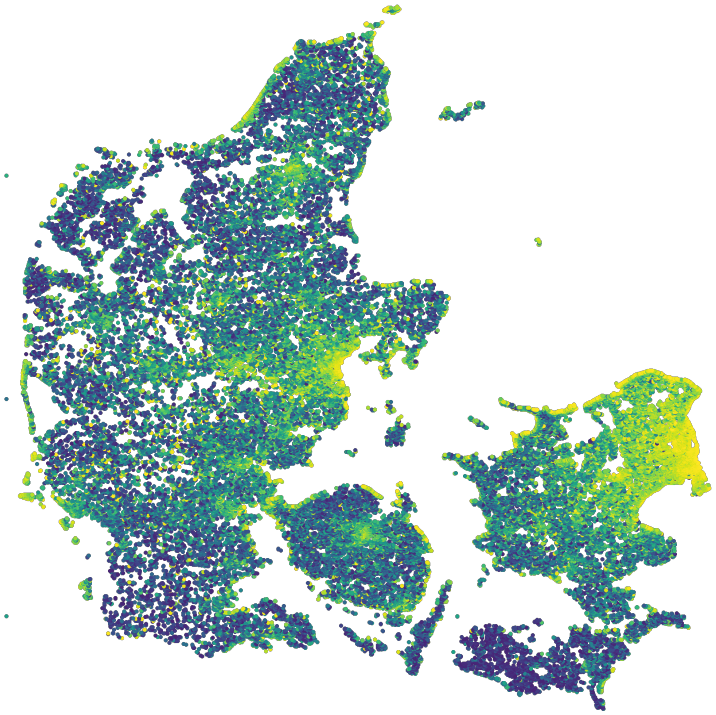

Figure 4.1: Geographic overview of square meter prices for houses and apartments in Denmark (excluding Bornholm for visual purposes). Notice the strong correlation with the major cities and the shore line. Also notice the three outliers west of Jutland.

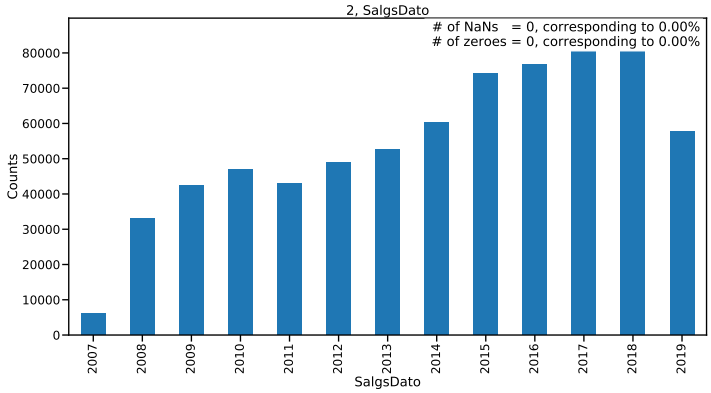

Figure 4.2: Histogram of prices of houses and apartments sold in Denmark.

Outside referencing isn't different: In fig. 4.3: Two images — fig. (a) shows an "A", fig. (b) shows a "B".

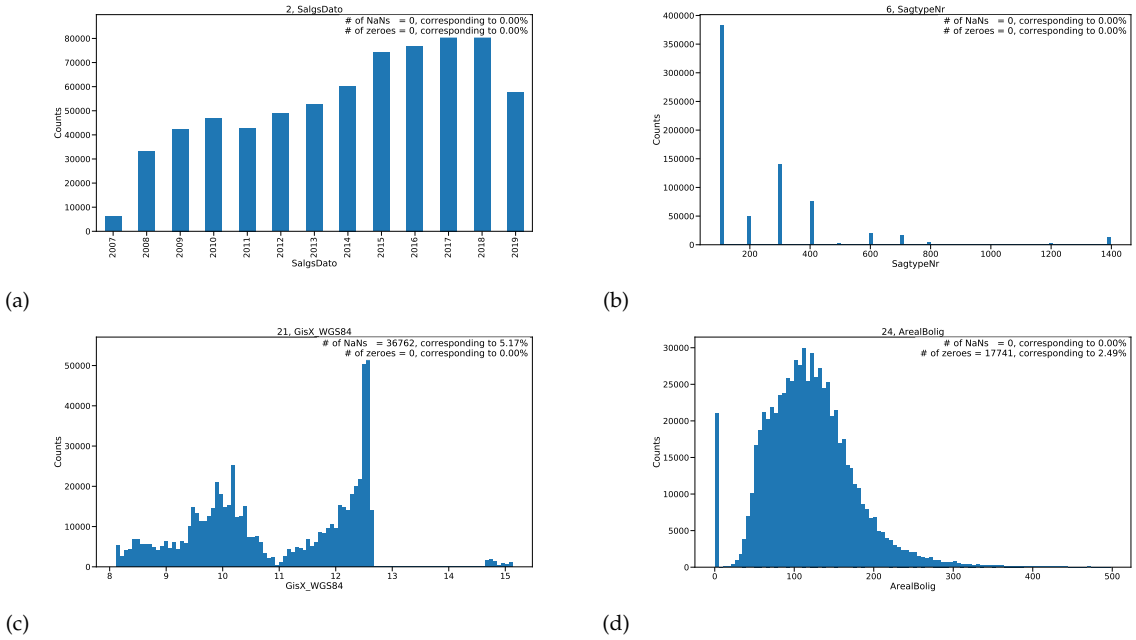

Figure 4.3: Distributions of the four out of the XXX input variables. Subplot (a) shows the date of the sale, Subplot (b) shows the type of sale, Subplot (c) shows the longitude, Subplot (d) shows the area,

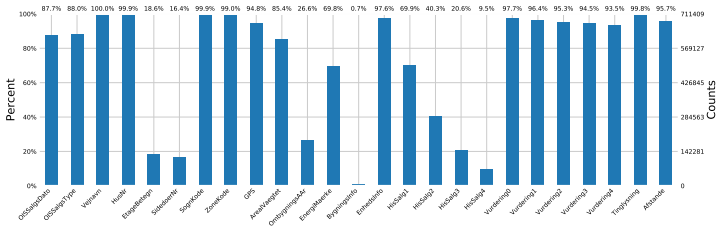

Figure 4.4: Percentage of valid counts for each variable grouped together in categories.

4.1 *Headings*

Tufte's books include the following heading levels: parts, chapters,[1] sections, subsections, and paragraphs. Not defined by default are: sub-subsections and subparagraphs.

Paragraph Paragraph headings (as shown here) are introduced by italicized text and separated from the main paragraph by a bit of space.

[1] Parts and chapters are defined for the `tufte-book` class only.

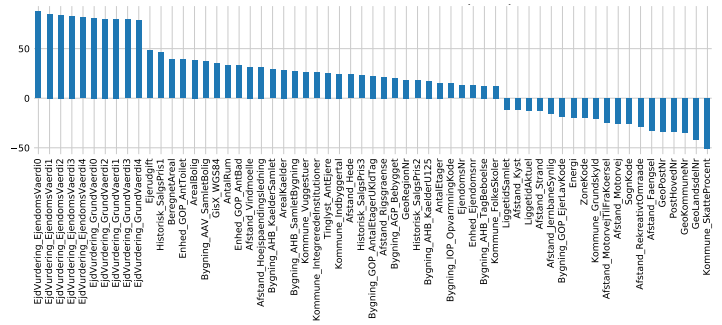

Figure 4.5: Linear correlation between variables and price for variables where the correlation coefficient ρ is $|\rho| > 10\%$.

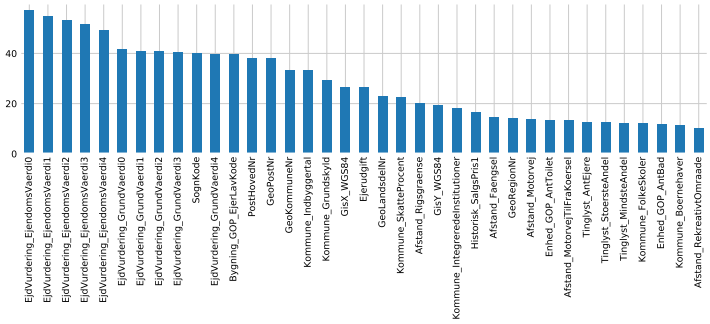

Figure 4.6: Non-linear correlation between variables and price using Maximal Information Coefficient (MIC) for variables where MIC $> 10\%$.

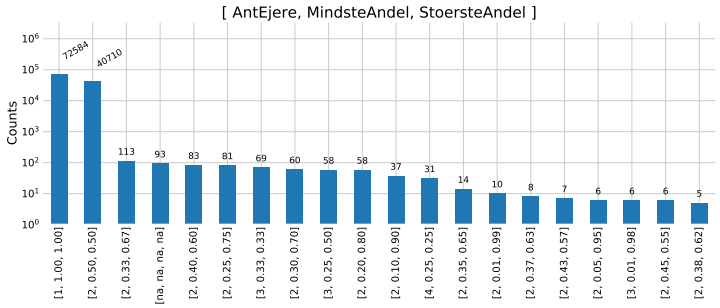

Figure 4.7: Overview of registration of property as a function of amount of owners (`AntEjere`), lowest share (`MindsteAndel`) and biggest share (`StoersteAndel`) written as `[AntEjere, MindsteAndel, StoersteAndel]` .

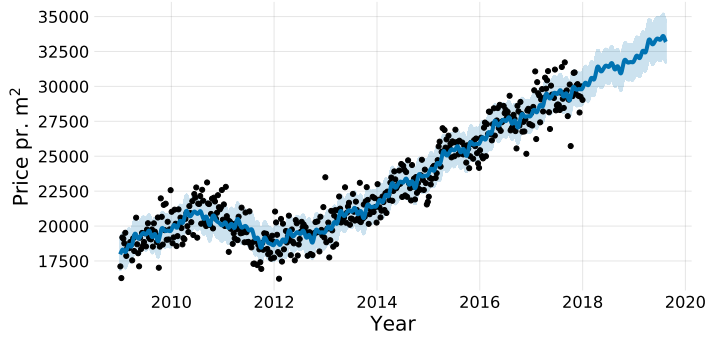

Figure 4.8: The predictions of the Facebook Prophet model trained on square meter prices for apartments sold before January 1st, 2018. The data is down-sampled to weekly bins where the median of each week is used as in input to the Prophet model. This can be seen as black dots in the figure. The model's forecasts for 2018 and 2019 are shown in blue with a light blue error band showing the 1-sigma confidence interval.

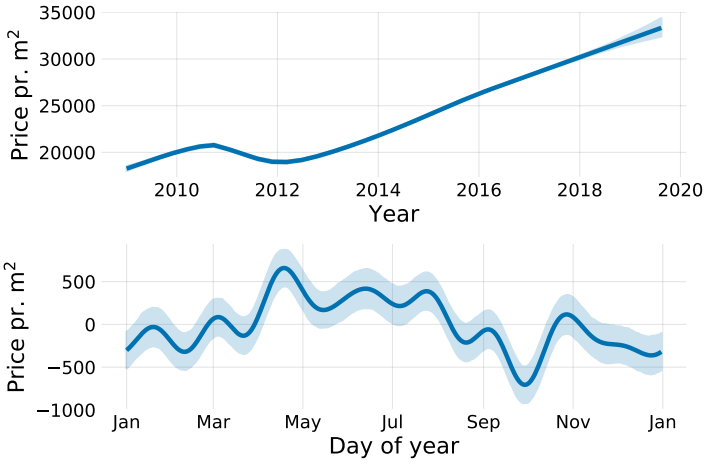

Figure 4.9: The trends of the Facebook Prophet model trained on square meter prices for apartments sold before January 1st, 2018. In the top plot is the overall trend as a function of year and in the bottom plot is the yearly variation as a function of day of year. It can be seen that the square meter price is higher during the Summer months compared to the Winter months, however, compared to the overall trend this effect is minor ($< 10\%$).

Halflife	$\log_{10}$	N_{trees}	Time [s]	MAD
2.5	True	210	81	0.1651
2.5	False	217	64	0.1791
5	True	209	78	0.1647
5	False	100	41	0.1950
10	**True**	**375**	**118**	**0.1563**
10	False	411	97	0.1739
20	True	333	106	0.1639
20	False	125	44	0.1925
∞	True	371	112	0.1596
∞	False	79	30	0.2062

Table 4.1: RMSE.

Halflife	$\log_{10}$	N_{trees}	Time [s]	MAD
2.5	True	180	57	0.1650
2.5	**False**	**682**	**111**	**0.1500**
5	True	198	62	0.1647
5	False	892	139	0.1514
10	True	244	68	0.1627
10	False	381	65	0.1608
20	True	396	95	0.1612
20	False	302	53	0.1626
∞	True	239	60	0.1639
∞	False	303	50	0.1620

Table 4.2: LogCosh.

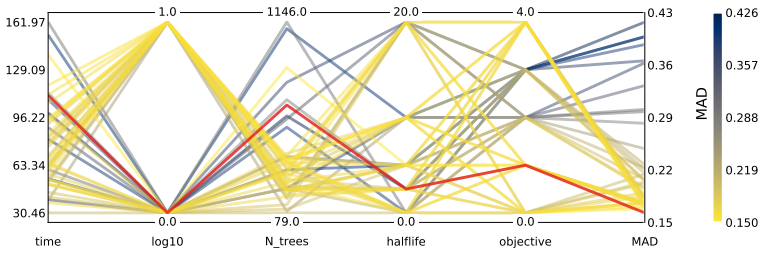

Figure 4.10: Hyperparameter optimization results of the housing model for apartments. The results are shown as parallel coordinates with each hyperparameter along the x-axis and the value of that parameter on the y-axis. Each line is an event in the 4-dimensional space colored according to the performance of that hyperparameter as measured by MAD from highest MAD in dark blue to lowest AUC in yellow. The single best hyperparameter is shown in red. For the hyperparamater `log10` `0` the LGBM 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.

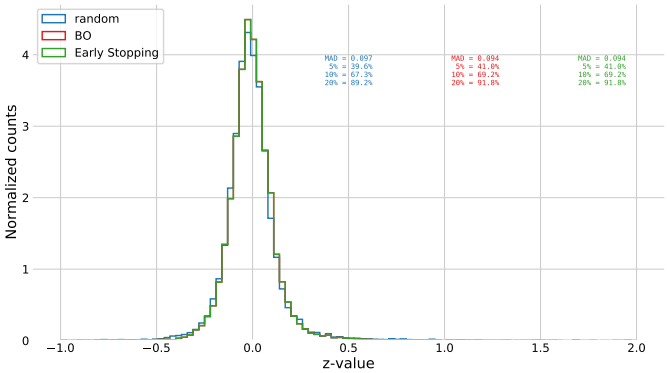

Figure 4.11: Histogram of z-values of the LGBM model trained on apartments. The performance after hyperparameter optimization (HPO) using Random Search (RS) is shown in blue for Bayesian Optimization (BO) in red. After finding the best model, BO in this case, the model is retrained using early stopping, the performance of which is shown in green.

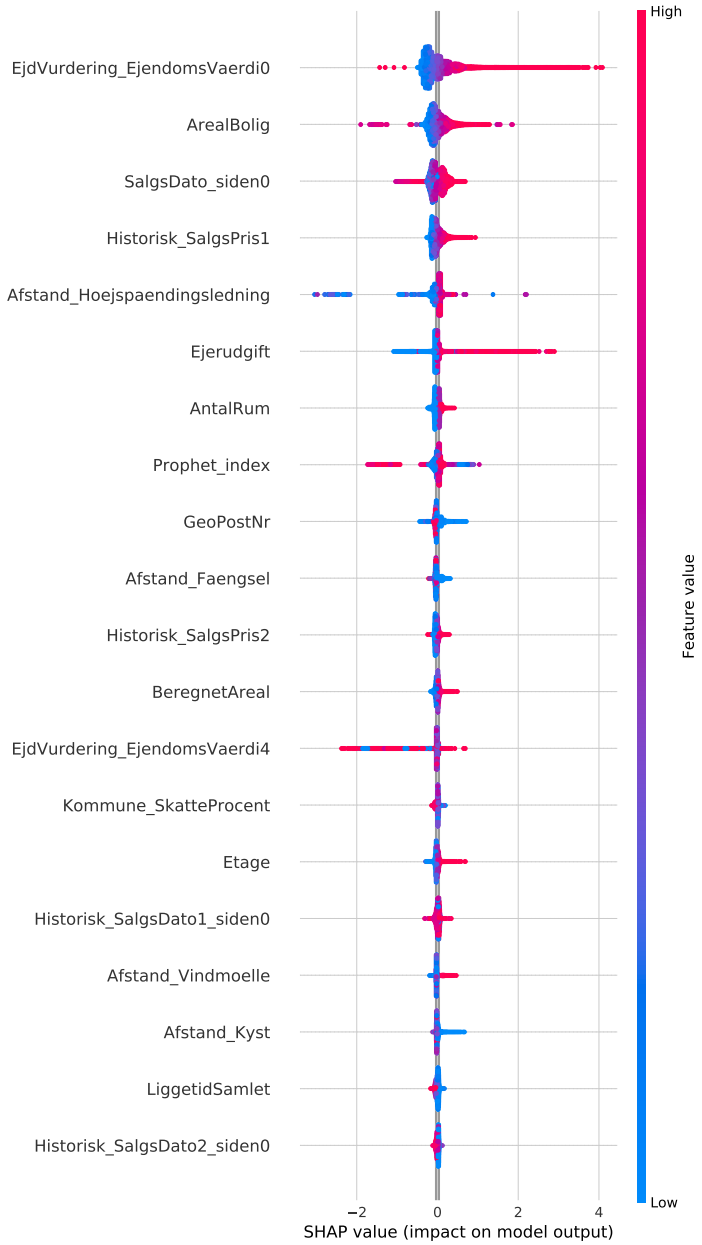

Figure 4.12: Feature importance of apartment prices using the LGB-model. The feature importance is measured using SHAP values. The variables are sorted top to bottom according to their overall feature importance, i.e. the previous public property valuation `EjdVurdering_EjendomdsVaerdi0` is the most 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.

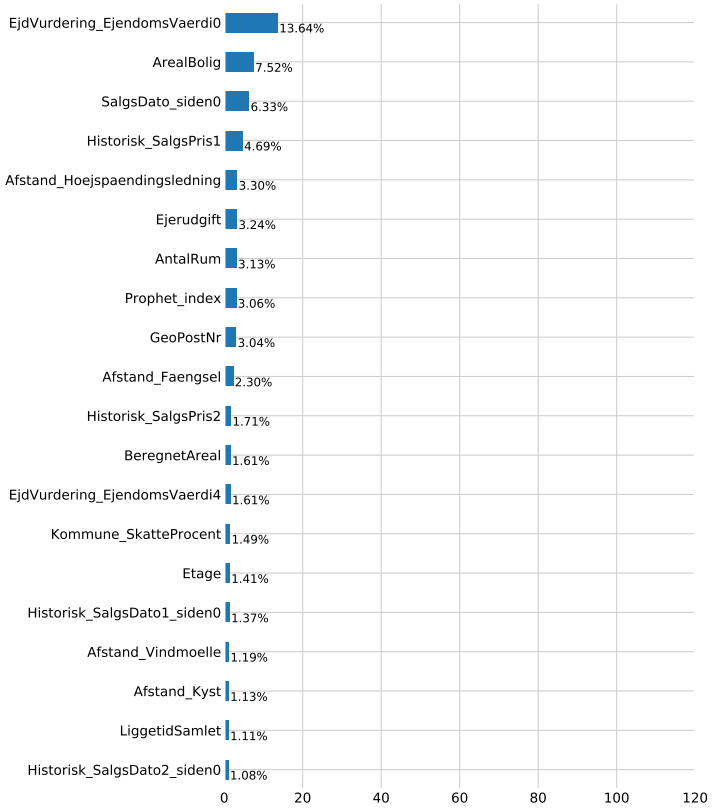

Figure 4.13: Total feature importance of apartment prices using the LGB-model. The total feature importance is measured using SHAP values, more specifically the mean of the absolute SHAP values for each variable. This allows a ranking of all of the features ordered after the total importance. Here the total feature importance is scaled such that they sum to 1 when summing over all variables. The absolute total feature importance is in itself not important, but the relative values indicate the difference in feature importance.

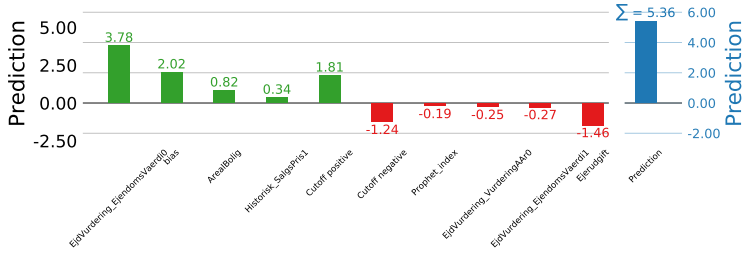

Figure 4.14: Model explanation for LGB model for a specific apartment. The bars are the variables in the dataset that the model found most important sorted after their importance for this particular apartment. The bias bar refers to the expected value of the model, which is simply the mean of the training 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 (5.36 in this example) calculated in 10^6 DKK. The negative values are shown in red, positive ones in green, and the prediction value in blue.

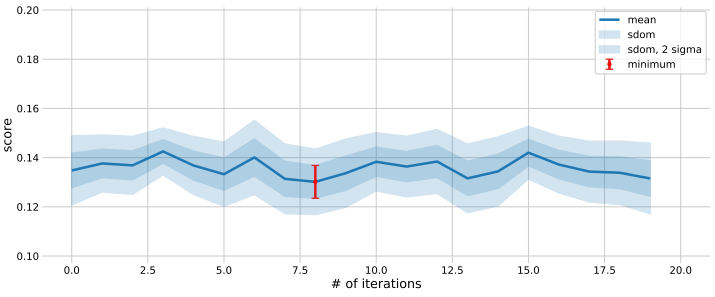

Figure 4.15: The results of running random search (RS) as hyperparameter optimization (HPO) on apartments using the LGB-model. The minimum (mean) loss along with its uncertainty is shown in red, the means for the different iterations of RS in blue, and as light blue bands are the one (and two) standard deviation(s) of the means (SDOM), all as a a function of iteration number.

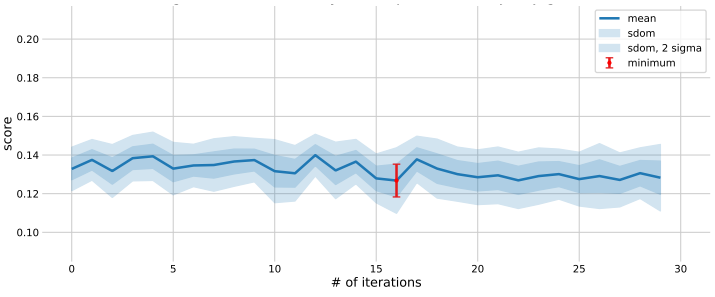

Figure 4.16: The results of running Bayesian optimization (BO) as hyperparameter optimization (HPO) on apartments using the LGB-model. The minimum (mean) loss along with its uncertainty is shown in red, the means for the different iterations of RS in blue, and as light blue bands are the one (and two) standard deviation(s) of the means (SDOM), all as a a function of iteration number. The first XXX iterations are run as RS for use as input to the internal optimization algorithm, the last XX iterations are thus the "smart" BO guesses.

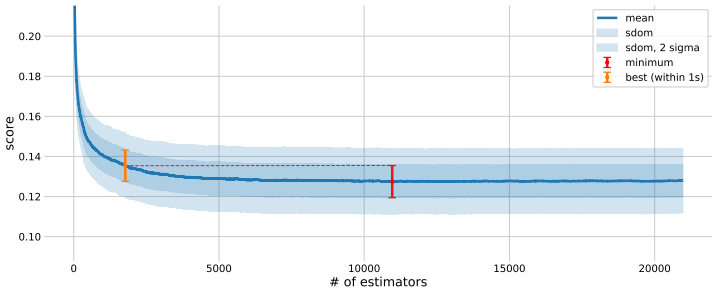

Figure 4.17: The results of early stopping on apartments using the LGB-model. The minimum (mean) loss along with its uncertainty is shown in red, the means for the different iterations of RS in blue, and as light blue bands are the one (and two) standard deviation(s) of the means (SDOM), all as a a function 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 1 SDOM of the minimum value. This leads to a model that has a performance that is within 1 SDOM of the best model, but a lot simpler and faster.

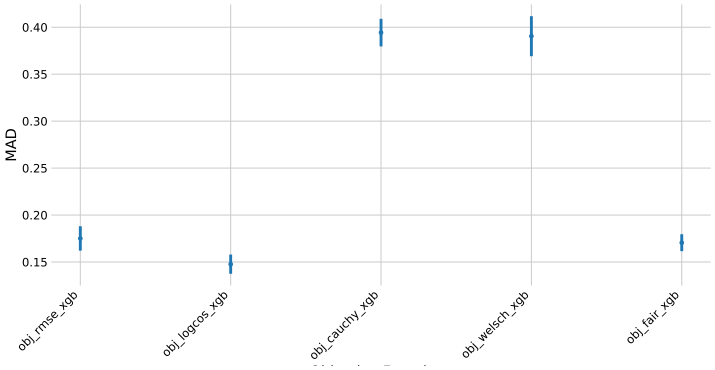

Figure 4.18: Comparison of performance (measured using the median of the absolute deviation, MAD, for apartments) of the five different objective functions: RMSE, LogCosh, Cauchy, Welsch, and Fair. We see that the default objective function, RMSE, does a reasonable job, however, the LogCosh function is a clear improvement.

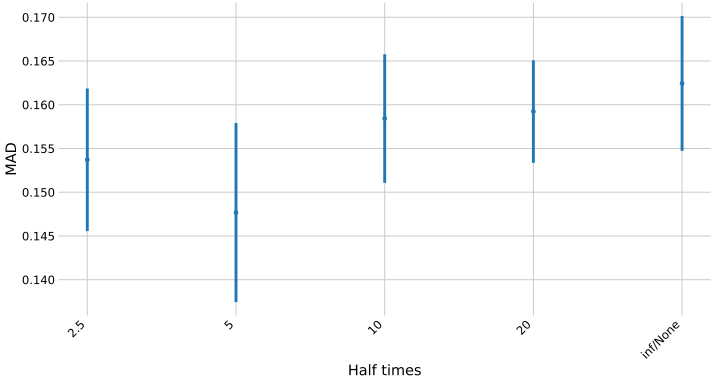

Figure 4.19: Comparison of performance (measured using the median of the absolute deviation, MAD, on apartments) of the five different half time weights: 2.5, 5, 10, 20, inf/None. We see that using no half time weights does has the worst performance, whereas a 5-year half time weight has the best performance (although with a very high uncertainty).

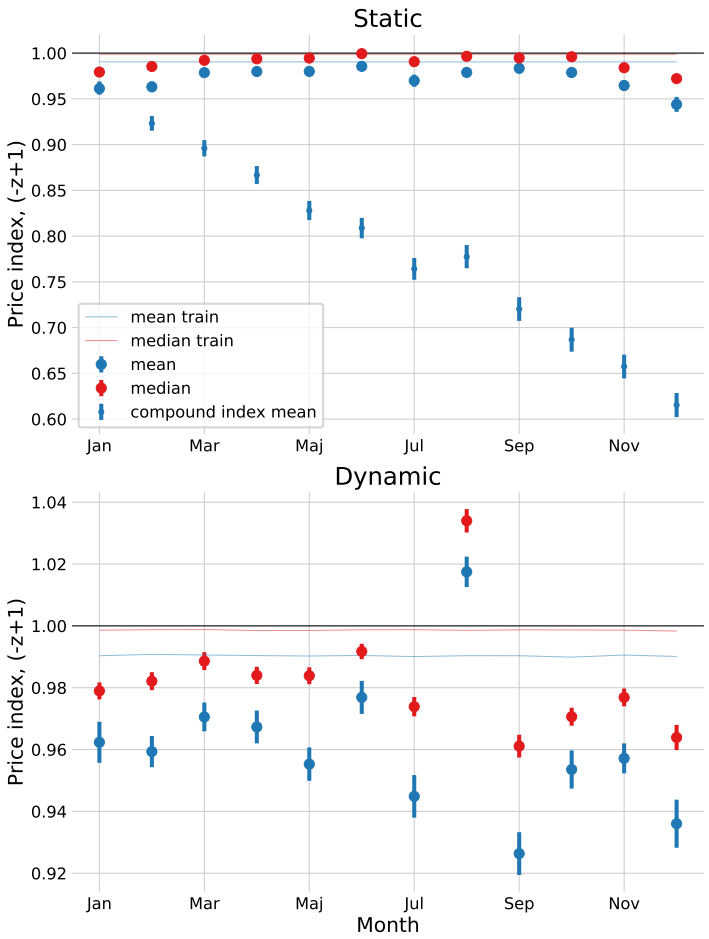

Figure 4.20: Performance of 1-month forecasts for 2018 for apartments. For both plots the LGM model is trained on data up to (but excluding) 2018. Top) The performance of the static model's prediction on sales in the individual months of 2018 is shown for both the mean and median of the z-scores. Bottom) Same as above, however this time based a dynamic model, i.e. a model which is retrained after every month to include the previous month's sales.

Heading	Style	Size
Part	roman	24/36×40 pc
Chapter	italic	20/30×40 pc
Section	italic	12/16×26 pc
Subsection	italic	11/15×26 pc
Paragraph	italic	10/14

Table 4.3: Heading styles used in *Beautiful Evidence*.

5. Particle Physics and LEP

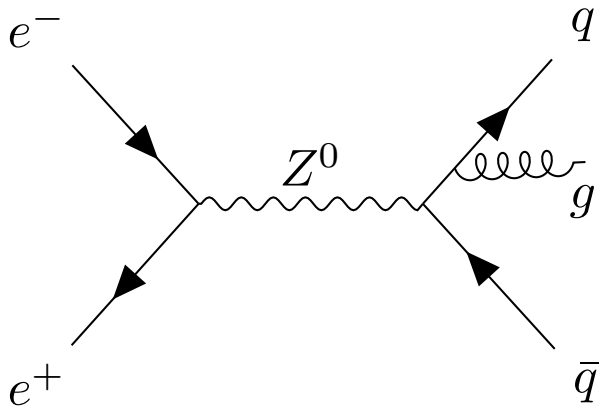

Figure 5.1: Feynman diagram showing the $e^+e^- \to Z^0$ production at LEP. The Z^0 has several decay modes where the $Z \to q\bar{q}g$ is shown here.

The Tufte-LaTeX document classes define a style similar to the style Edward Tufte uses in his books and handouts. Tufte's style is known for its extensive use of sidenotes, tight integration of graphics with text, and well-set typography. This document aims to be at once a demonstration of the features of the Tufte-LaTeX document classes and a style guide to their use.

5.1 Page Layout

5.1.1 Headings

This style provides A- and B-heads (that is, `\section` and `\subsection`), demonstrated above.

If you need more than two levels of section headings, you'll have to define them yourself at the moment; there are no pre-defined styles for anything below a `\subsection`. As Bringhurst points out in *The Elements of Typographic Style*, you should "use as many levels of headings as you need: no more, and no fewer."

The Tufte-LaTeX classes will emit an error if you try to use `\subsubsection` and smaller headings.

In his later books, Tufte starts each section with a bit of vertical space, a non-indented paragraph, and sets the first few words of the

sentence in SMALL CAPS. To accomplish this using this style, use the `\newthought` command:

```
\newthought{In his later books}, Tufte starts...
```

6. Quark Gluon Analysis

6.1 Sidenotes

One of the most prominent and distinctive features of this style is the extensive use of sidenotes. There is a wide margin to provide ample room for sidenotes and small figures. Any `\footnotes` will automatically be converted to sidenotes.[1] If you'd like to place ancillary information in the margin without the sidenote mark (the superscript number), you can use the `\marginnote` command.

[1] This is a sidenote that was entered using the `\footnote` command.

This is a margin note. Notice that there isn't a number preceding the note, and there is no number in the main text where this note was written.

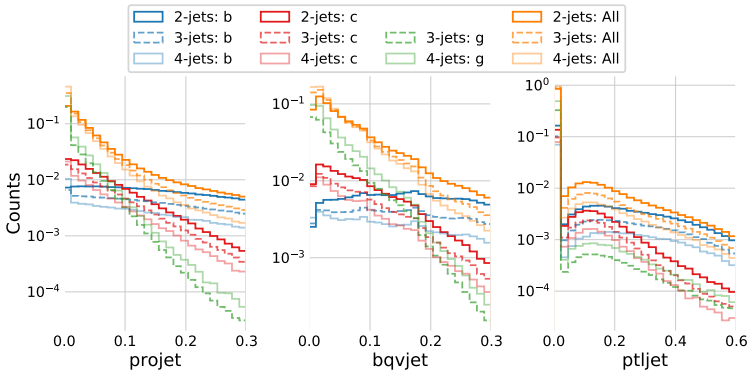

Figure 6.1: Histograms of the three vertex variables, `projet`, `bqvjet`, and `ptljet`, used as input variables in the b-tagging models. In blue colors the variables are shown for true b-jets, in red for true c-jets, in green for true g-jets, and in orange for all of the jets (including non q-matched). 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.

The specification of the `\sidenote` command is:

`\sidenote[`⟨*number*⟩`][`⟨*offset*⟩`]{`*Sidenote text.*`}`

Both the ⟨*number*⟩ and ⟨*offset*⟩ arguments are optional. If you provide a ⟨*number*⟩ argument, then that number will be used as the sidenote number. It will change of the number of the current sidenote only and will not affect the numbering sequence of subsequent sidenotes.

Sometimes a sidenote may run over the top of other text or graphics in the margin space. If this happens, you can adjust the vertical position of the sidenote by providing a dimension in the ⟨*offset*⟩ argument. Some examples of valid dimensions are:

```
1.0in    2.54cm    254mm    6\baselineskip
```

If the dimension is positive it will push the sidenote down the page; if the dimension is negative, it will move the sidenote up the page.

While both the ⟨*number*⟩ and ⟨*offset*⟩ arguments are optional, they must be provided in order. To adjust the vertical position of the sidenote while leaving the sidenote number alone, use the following syntax:

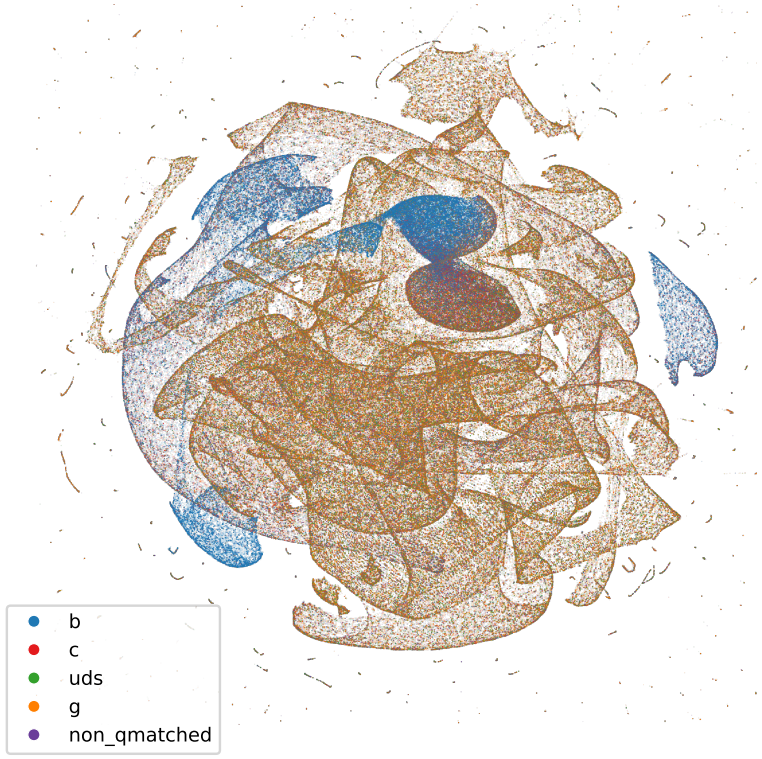

Figure 6.2: Vizualisation of the vertex variables for the different categories: true b-jets in blue, true c-jets in red, true uds-jets in green, true g-jets in orange, and non q-matched. The clustering is performed with the UMAP algorithm which outputs a 2D-projection. This projection is then visualized using the Datashader which takes takes care of point size, avoids over- and under-plotting, and color intensity.

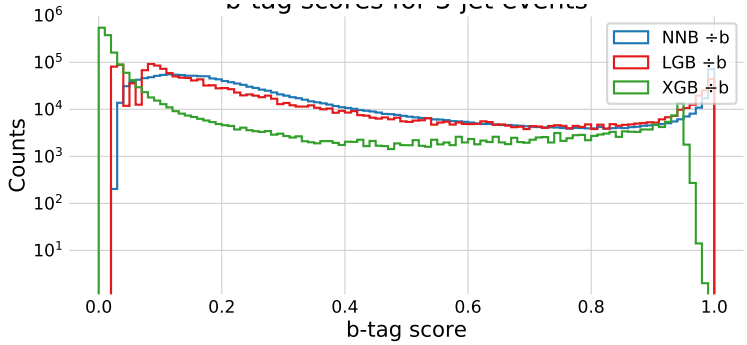

Figure 6.3: Histogram of b-tag scores (model prediction) in 3-jet events for NNB (the neural network trained by ATLAS, also called `nnbjet`) in blue, LGB in red, and XGB in green. We see that the LGB predictions closely match those of NNB which is a good confirmation of a successful fit.

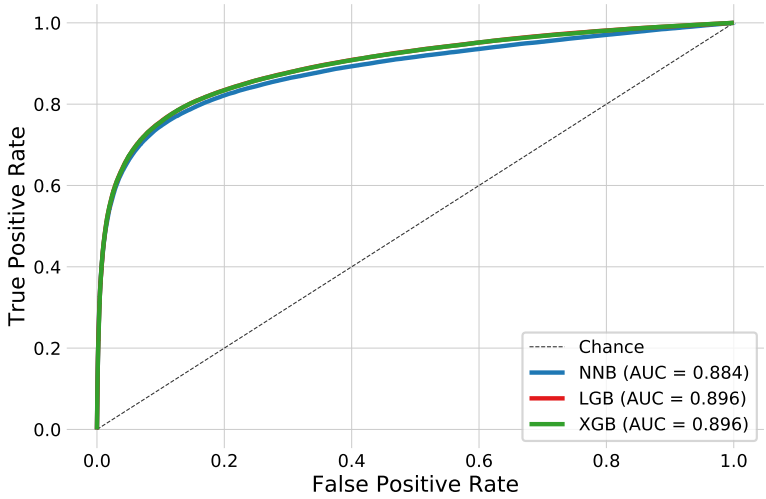

Figure 6.4: ROC curve of the three b-tag models in 3-jet events for NNB (the neural network trained by ATLAS, also called `nnbjet`) in blue, LGB in red, and XGB in green. In the legend the Area Under Curve (AUC) is also shown. Notice that the LGB and XGB models share performance and it is thus due to overplotting that only the green line for XGB can be seen. In the particle physics community False Positive Rate (FPR) is sometimes better known as background efficiency and True Positive Rate (TPR) as signal efficiency.

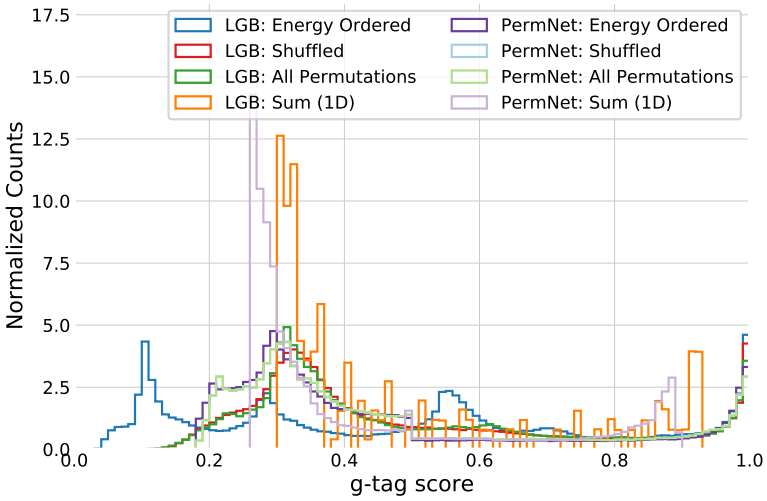

Figure 6.5: Histogram of g-tag scores (model prediction) in 4-jet events for LGB: Energy Ordered in blue, LGB: Shuffled in red, LGB: All Permutations in green, LGB: Sum 1D in orange, PermNet: Energy Ordered in purple, PermNet: Shuffled in light-blue, PermNet: All Permutations in light-green, PermNet: 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.

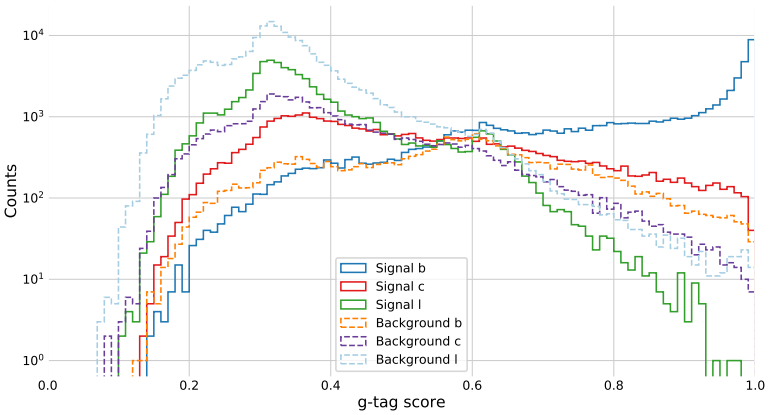

Figure 6.6: Histogram of g-tag scores (model prediction) from the LGB-model in 4-jet events for b signal in blue, c signal in red, l signal in green, b background in orange, c background in purple, l background in light-blue.

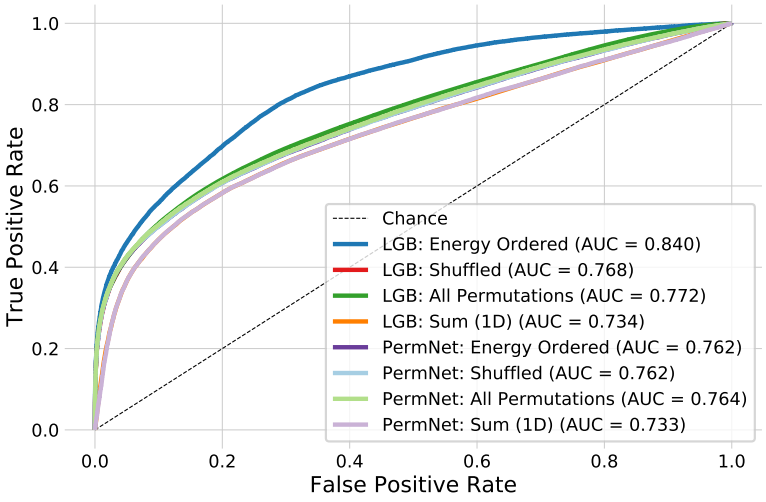

Figure 6.7: ROC curve of the eight g-tag models in 4-jet events. First one in dashed black is the ROC curve that you get by random chance. The colors are the same as in Figure 6.5 and in the legend also the Area Under the ROC curve (AUC) is shown. Notice that the LGB model which uses the energy ordered data produced the best model, however, this model is not permutation invariant. Of the permutation invariant models (the rest), the LGB 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.

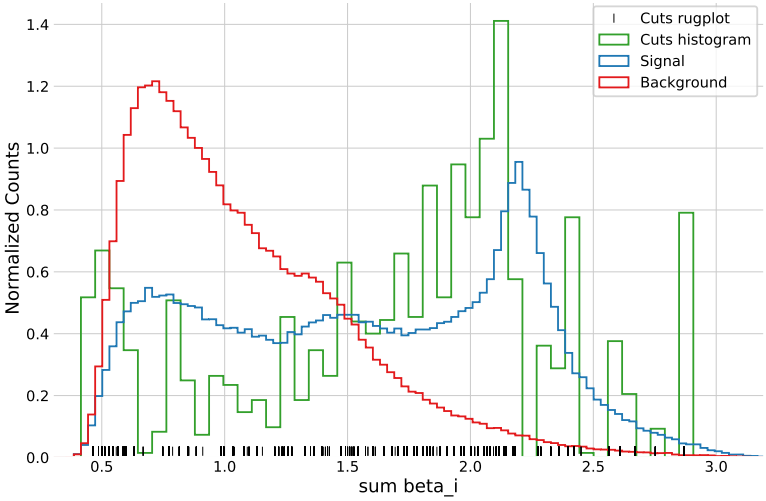

Figure 6.8: Histogram of the distribution of signal in blue and background in red for 1-dimensional sum of b-tags training data. A histogram of the cut values from the LGB model trained on this data is shown in green together with a rug plot of the cut values in black. Notice how most of the cuts match up with the signal peak at around a $\sum \beta_i \sim 2.1$, however, there are also quite a lot of cuts around $\sum \beta_i \sim 0.5$.

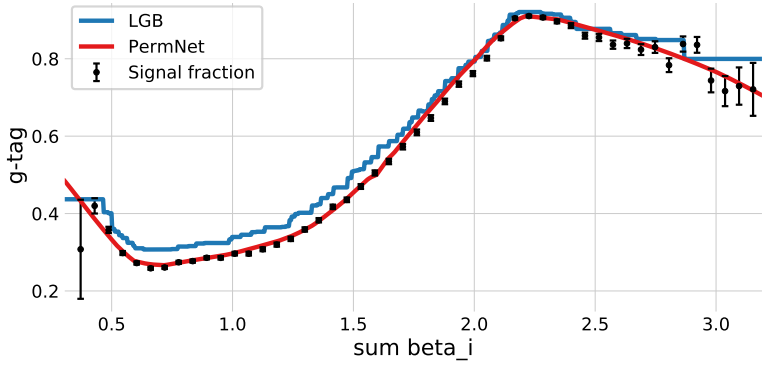

Figure 6.9: Plot of the (1D) g-tag scores as a function of $\sum \beta_i$ for the LGB model in blue and the PermNet model in red. Here the g-tag scores are just the models' output values when input a uniformly spaced grid of $\sum \beta_i$ values between 0 and 4. The signal fraction (based on the signal and background histograms in Figure 6.8) 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. Notice how both models capture the overall trend of the signal fraction with the PermNet being particularly close.

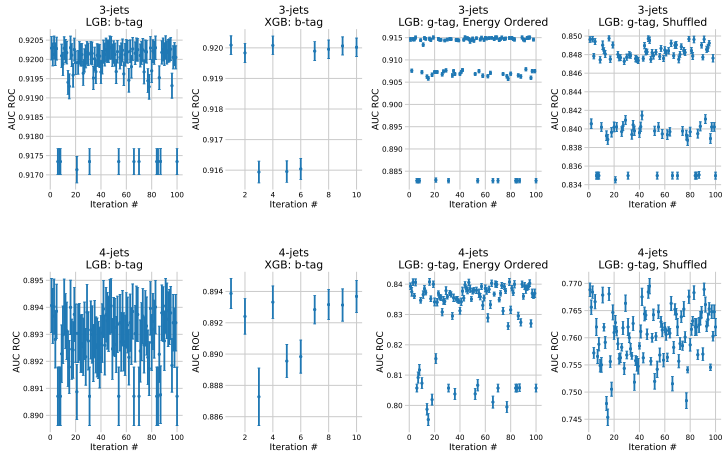

Figure 6.10: Hyperparameter Optimization (HPO) results after running 100 iterations of Random Search (only 10 for XGB). In the top row are the results of the 3-jet models and in the bottom row the results of the 4-jet models. From left to right, we have first) the b-tagging results of LGB, second) the b-tagging results of XGB using only 10 iterations of RS, third) the g-tagging results of LGB fit on the Energy Ordered b-tags, and forth) the g-tagging results of LGB fit on the shuffled b-tags. Notice the different ranges on the y-axes.

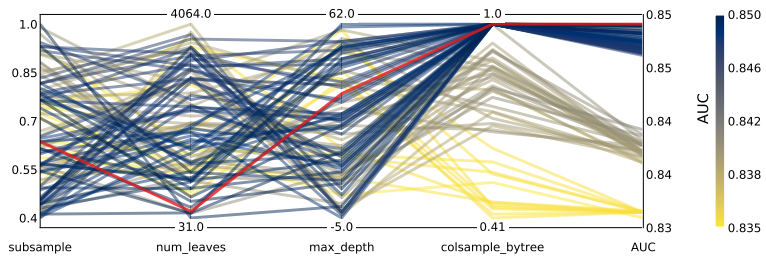

Figure 6.11: Hyperparameter optimization results of g-tagging for 3-jet shuffled events. The results are shown as parallel coordinates with each hyperparameter along the x-axis and the value of that parameter on the y-axis. Each line is an event in the 4-dimensional space colored according to the performance 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.

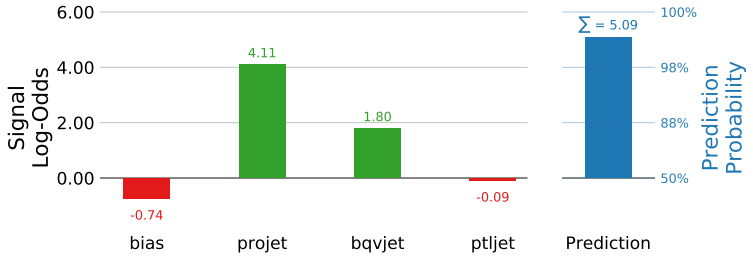

Figure 6.12: Model explanation for the 3-jet b-tagging model for a b-like jet. The first column is the bias of the training set which acts as the naive prediction baseline, the rest are the input data variables. On the right hand side of the plot is the model prediction shown. The left part of the plot is shown in log-odds space, the right part in probability space. The model prediction is the sum of the log-odds (5.09 in this example) transformed into probability space. The negative log-odd values are shown in red, positive ones in green, and the prediction value in blue.

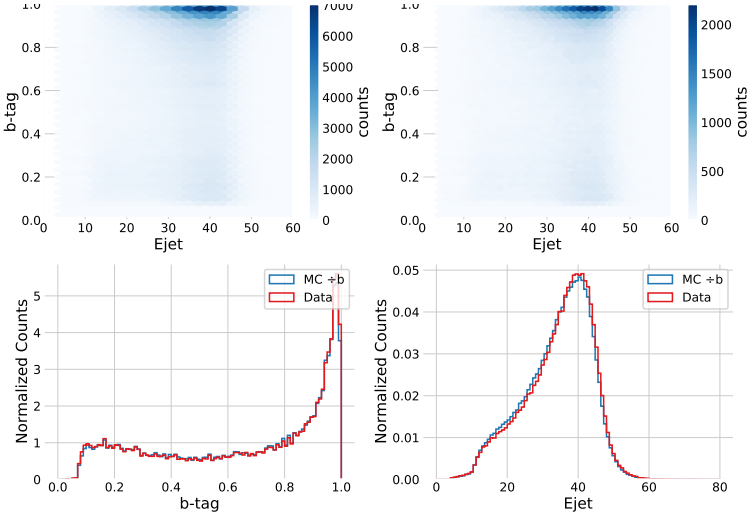

Figure 6.13: Comparison of the b-tag and jet energy (`Ejet`) distributions for Monte Carlo (MC) versus data. In the top row the 2D-distributions are shown for MC on the left (without the extra MCb samples) and data on the right. In the bottom row the 1D marginal distrubtions are shown for the b-tag and the jet energy with data in red and Monte Carlo ones in blue. Notice the the almost identical distributions in b-tag.

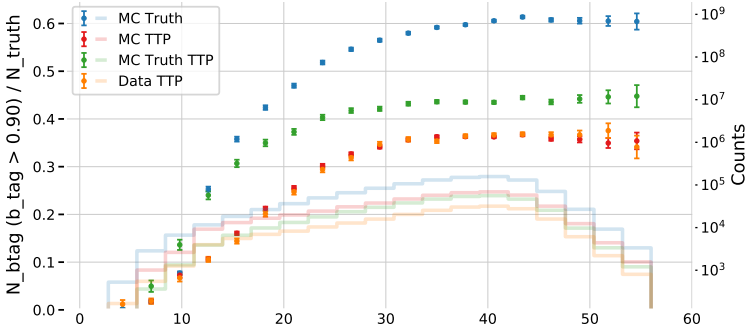

Figure 6.14: Efficiency of the b-tags for b-jets in the b-signal region for 3-jet events, $\varepsilon_b^{b-\text{sig}}$, as a function of jet energy `Ejet` . The b-signal region is defined as $\beta > 0.9$. In the plot the efficiencies are shown for MC Truth in blue, MC TTP in red, MC Truth TTP in green, and Data 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. The abbreviation TTP is short for "Tag, Tag, Probe" where two jets in a event are used as tags and the probe is then used for further analysis. Notice how both MC TTP and Data TTP follow each other closely.

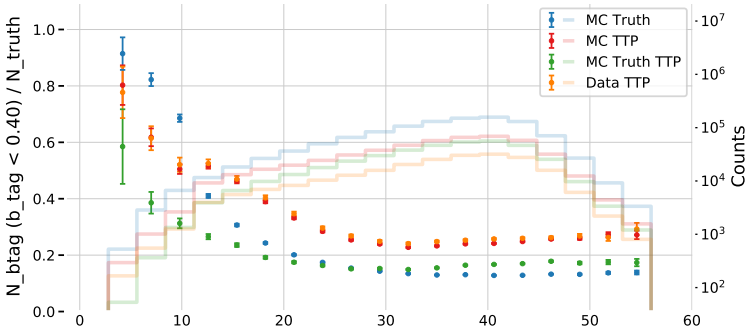

Figure 6.15: Efficiency of the b-tags for b-jets in the g-signal region for 3-jet events, $\varepsilon_b^{g-\text{sig}}$, as a function of jet energy `Ejet` . The g-signal region is defined as $\beta < 0.4$. In the plot the efficiencies are shown for MC Truth in blue, MC TTP in red, MC Truth TTP in green, and Data 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. The abbreviation TTP is short for "Tag, Tag, Probe" where two jets in a event are used as tags and the probe is then used for further analysis. Notice how both MC TTP and Data TTP follow each other closely.

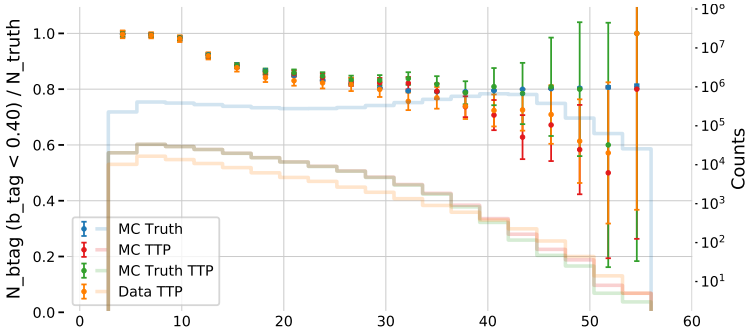

Figure 6.16: Efficiency of the b-tags for g-jets in the g-signal region for 3-jet events, $\varepsilon_g^{g-\text{sig}}$, as a function of jet energy `Ejet`. The g-signal region is defined as $\beta < 0.4$. In the plot the efficiencies are shown for MC Truth in blue, MC TTP in red, MC Truth TTP in green, and Data 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. The abbreviation TTP is short for "Tag, Tag, Probe" where two jets in a event are used as tags and the probe is then used for further analysis. Notice how both MC TTP and Data TTP follow each other closely.

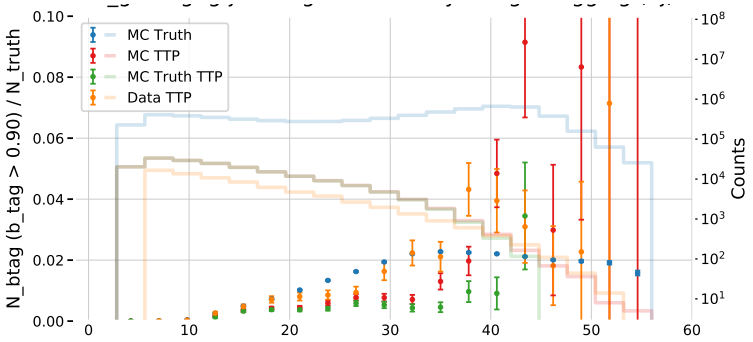

Figure 6.17: Efficiency of the b-tags for g-jets in the b-signal region for 3-jet events, $\varepsilon_g^{b-\text{sig}}$, as a function of jet energy `Ejet`. The b-signal region is defined as $\beta > 0.9$. In the plot the efficiencies are shown for MC Truth in blue, MC TTP in red, MC Truth TTP in green, and Data 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. The abbreviation TTP is short for "Tag, Tag, Probe" where two jets in a event are used as tags and the probe is then used for further analysis.

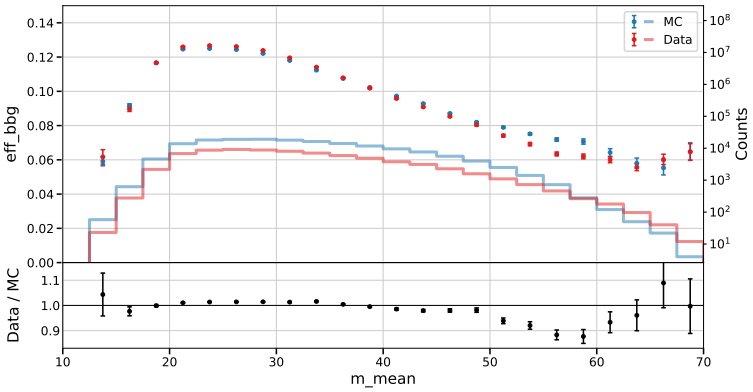

Figure 6.18: Proxy efficiency of the g-tags for $b\bar{b}g$ 3-jet events as a function of the mean of the two invariant masses m_{bg} and $m_{\bar{b}g}$. The proxy efficiency $\varepsilon_{b\bar{b}g}$ is measured by finding $b\bar{b}g$-events where $\beta_b > 0.9$, $\beta_{\bar{b}} > 0.9$, and $\beta_g < 0.4$. and then calculating $\varepsilon_{b\bar{b}g} = \varepsilon_b^{b-\text{sig}} \cdot \varepsilon_{\bar{b}}^{b-\text{sig}} \cdot \varepsilon_g^{g-\text{sig}}$. In the top plot $\varepsilon_{b\bar{b}g}$ 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.

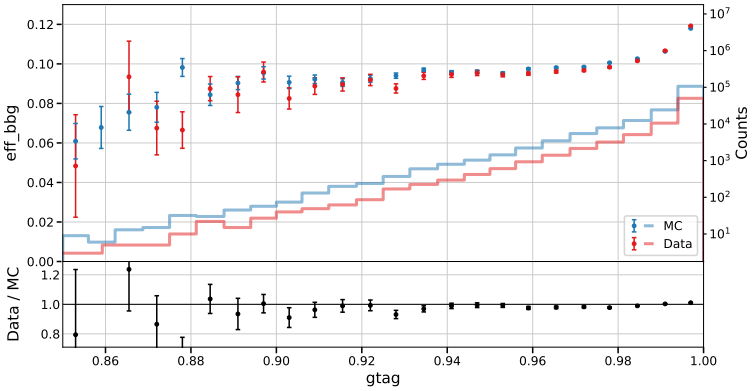

Figure 6.19: Proxy efficiency of the g-tags for $b\bar{b}g$ 3-jet events as a function of the event's g-tag. The proxy efficiency $\varepsilon_{b\bar{b}g}$ is measured by finding $b\bar{b}g$-events where $\beta_b > 0.9$, $\beta_{\bar{b}} > 0.9$, and $\beta_g < 0.4$. and then calculating $\varepsilon_{b\bar{b}g} = \varepsilon_b^{b-\text{sig}} \cdot \varepsilon_{\bar{b}}^{b-\text{sig}} \cdot \varepsilon_g^{g-\text{sig}}$. In the top plot $\varepsilon_{b\bar{b}g}$ 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.

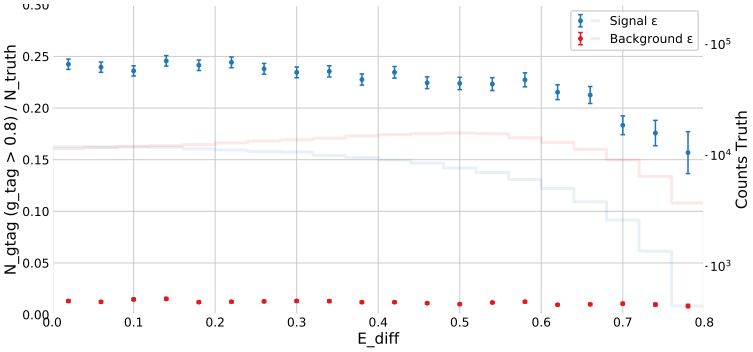

Figure 6.20: Efficiency of the g-tags for 4-jet events as a function of normalized gluon gluon jet energy difference in Monte Carlo. The efficiency is measured as the number of events with a g-tag higher than 0.8 ($\gamma > 0.8$) out of the total number and the normalized gluon gluon jet energy difference A is $A = \frac{E_{g_{\max}} - E_{g_{\min}}}{E_{g_{\max}} + E_{g_{\min}}}$ where $E_{g_{\max}}$ ($E_{g_{\min}}$) refers to the energy of the gluon with the highest (lowest) energy. The efficiency is plotted for signal events according to MC Truth in blue and background events according to MC Truth in red.

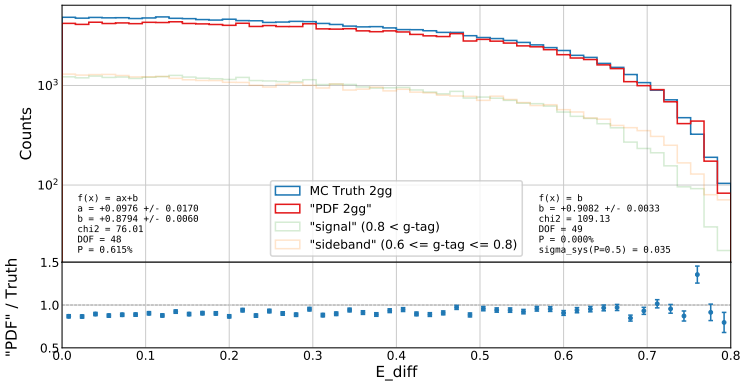

Figure 6.21: Closure plot between MC Truth and the corrected g-tagging model in 4-jet events for the normalized gluon gluon jet energy difference. The corrected g-taggingg model is described in further detail in section XXX **TODO!**. In the top part of the plot the MC Truth is shown in blue, the corrected g-tagging model `"PDF 2gg"` in red, the g-signal distribution in semi-transparent green and the g-sideband distribution in semi-transparent orange. In the bottom part of the plot the ratio between MC Truth and the output of the corrected g-tagging model is shown. The normalized gluon gluon jet energy difference A is $A = \frac{E_{g_{\max}} - E_{g_{\min}}}{E_{g_{\max}} + E_{g_{\min}}}$ where $E_{g_{\max}}$ ($E_{g_{\min}}$) refers to the energy of the gluon with the highest (lowest) energy.

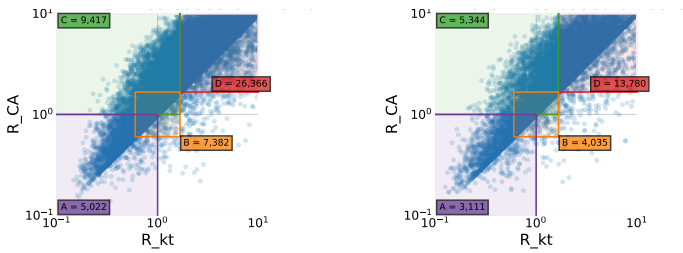

Figure 6.22: R kt CA overview XXX **TODO!**

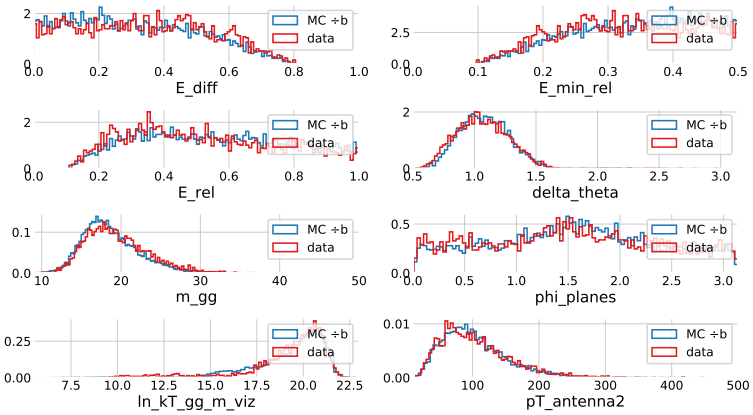

Figure 6.23: R kt CA cut region A XXX **TODO!**

`\sidenote[][`⟨*offset*⟩`]{`*Sidenote text.*`}`

The empty brackets tell the `\sidenote` command to use the default sidenote number.

If you *only* want to change the sidenote number, however, you may completely omit the ⟨*offset*⟩ argument:

`\sidenote[`⟨*number*⟩`]{`*Sidenote text.*`}`

The `\marginnote` command has a similar *offset* argument:

`\marginnote[`⟨*offset*⟩`]{`*Margin note text.*`}`

6.2 References

References are placed alongside their citations as sidenotes, as well. This can be accomplished using the normal `\citep` command.[2]

[2] The first paragraph of this document includes a citation.

The complete list of references may also be printed automatically by using the `\bibliography` command. (See the end of this document for an example.) If you do not want to print a bibliography at the end of your document, use the `\nobibliography` command in its place.

6.3 Figures and Tables

Images and graphics play an integral role in Tufte's work. In addition to the standard `figure` and `tabular` environments, this style provides special figure and table environments for full-width floats.

Full page–width figures and tables may be placed in `figure*` or `table*` environments. To place figures or tables in the margin, use the `marginfigure` or `margintable` environments as follows (see figure 6.24):

```
\begin{marginfigure}
    \includegraphics{helix}
    \caption{This is a margin figure.}
    \label{fig:marginfig}
\end{marginfigure}
```

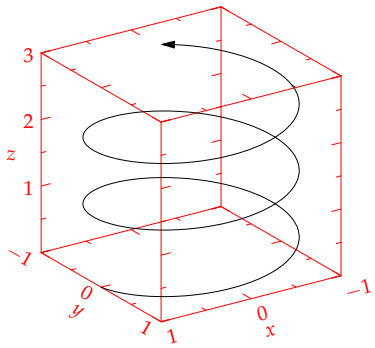

Figure 6.24: This is a margin figure. The helix is defined by $x = \cos(2\pi z)$, $y = \sin(2\pi z)$, and $z = [0, 2.7]$. The figure was drawn using Asymptote (`http://asymptote.sf.net/`).

The `marginfigure` and `margintable` environments accept an optional parameter ⟨*offset*⟩ that adjusts the vertical position of the figure or table. See the "Sidenotes" section above for examples. The specifications are:

```
\begin{marginfigure}[⟨offset⟩]
    ...
\end{marginfigure}

\begin{margintable}[⟨offset⟩]
    ...
\end{margintable}
```

Figure 6.25 is an example of the `figure*` environment and figure 6.26 is an example of the normal `figure` environment.

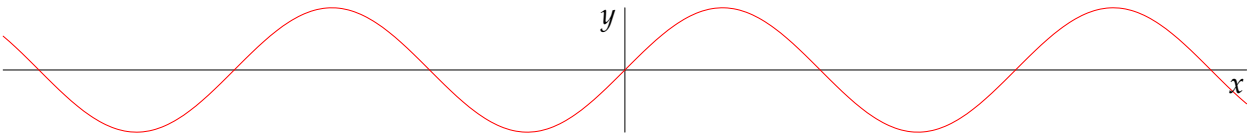

Figure 6.25: This graph shows $y = \sin x$ from about $x = [-10, 10]$. *Notice that this figure takes up the full page width.*

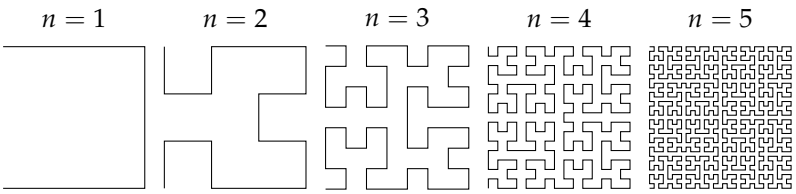

Figure 6.26: Hilbert curves of various degrees n. *Notice that this figure only takes up the main textblock width.*

As with sidenotes and marginnotes, a caption may sometimes require vertical adjustment. The `\caption` command now takes a second optional argument that enables you to do this by providing a dimension ⟨*offset*⟩. You may specify the caption in any one of the following forms:

```
\caption{long caption}
\caption[short caption]{long caption}
\caption[][⟨offset⟩]{long caption}
\caption[short caption][⟨offset⟩]{long caption}
```

A positive ⟨*offset*⟩ will push the caption down the page. The short caption, if provided, is what appears in the list of figures/tables, otherwise the "long" caption appears there. Note that although the arguments ⟨*short caption*⟩ and ⟨*offset*⟩ are both optional, they must be provided in order. Thus, to specify an ⟨*offset*⟩ without specifying a ⟨*short caption*⟩, you must include the first set of empty brackets `[]`, which tell `\caption` to use the default "long" caption. As an example, the caption to figure 6.26 above was given in the form

```
\caption[Hilbert curves...][6pt]{Hilbert curves...}
```

Table 6.1 shows table created with the `booktabs` package. Notice the lack of vertical rules—they serve only to clutter the table's data.

Margin	Length
Paper width	8 1/2 inches
Paper height	11 inches
Textblock width	6 1/2 inches
Textblock/sidenote gutter	3/8 inches
Sidenote width	2 inches

Table 6.1: Here are the dimensions of the various margins used in the Tufte-handout class.

Occasionally LaTeX will generate an error message:

```
Error:  Too many unprocessed floats
```

LaTeX tries to place floats in the best position on the page. Until it's finished composing the page, however, it won't know where those positions are. If you have a lot of floats on a page (including

sidenotes, margin notes, figures, tables, etc.), LaTeX may run out of "slots" to keep track of them and will generate the above error.

LaTeX initially allocates 18 slots for storing floats. To work around this limitation, the Tufte-LaTeX document classes provide a `\morefloats` command that will reserve more slots.

The first time `\morefloats` is called, it allocates an additional 34 slots. The second time `\morefloats` is called, it allocates another 26 slots.

The `\morefloats` command may only be used two times. Calling it a third time will generate an error message. (This is because we can't safely allocate many more floats or LaTeX will run out of memory.)

If, after using the `\morefloats` command twice, you continue to get the `Too many unprocessed floats` error, there are a couple things you can do.

The `\FloatBarrier` command will immediately process all the floats before typesetting more material. Since `\FloatBarrier` will start a new paragraph, you should place this command at the beginning or end of a paragraph.

The `\clearpage` command will also process the floats before continuing, but instead of starting a new paragraph, it will start a new page.

You can also try moving your floats around a bit: move a figure or table to the next page or reduce the number of sidenotes. (Each sidenote actually uses *two* slots.)

After the floats have placed, LaTeX will mark those slots as unused so they are available for the next page to be composed.

6.4 Captions

You may notice that the captions are sometimes misaligned. Due to the way LaTeX's float mechanism works, we can't know for sure where it decided to put a float. Therefore, the Tufte-LaTeX document classes provide commands to override the caption position.

Vertical alignment To override the vertical alignment, use the `\setfloatalignment` command inside the float environment. For example:

```
\begin{figure}[btp]
   \includegraphics{sinewave}
   \caption{This is an example of a sine wave.}
   \label{fig:sinewave}
   \setfloatalignment{b}% forces caption to be bottom-aligned
\end{figure}
```

The syntax of the `\setfloatalignment` command is:

```
\setfloatalignment{⟨pos⟩}
```

where ⟨*pos*⟩ can be either `b` for bottom-aligned captions, or `t` for top-aligned captions.

Horizontal alignment To override the horizontal alignment, use either the `\forceversofloat` or the `\forcerectofloat` command inside of the float environment. For example:

```
\begin{figure}[btp]
   \includegraphics{sinewave}
   \caption{This is an example of a sine wave.}
   \label{fig:sinewave}
   \forceversofloat% forces caption to be set to the left of the float
\end{figure}
```

The `\forceversofloat` command causes the algorithm to assume the float has been placed on a verso page—that is, a page on the left side of a two-page spread. Conversely, the `\forcerectofloat` command causes the algorithm to assume the float has been placed on a recto page—that is, a page on the right side of a two-page spread.

6.5 Full-width text blocks

In addition to the new float types, there is a `fullwidth` environment that stretches across the main text block and the sidenotes area.

```
\begin{fullwidth}
Lorem ipsum dolor sit amet...
\end{fullwidth}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.6 Typography

6.6.1 Typefaces

If the Palatino, Helvetica, and `Bera Mono` typefaces are installed, this style will use them automatically. Otherwise, we'll fall back on the Computer Modern typefaces.

6.6.2 Letterspacing

This document class includes two new commands and some improvements on existing commands for letterspacing.

When setting strings of A L L C A P S or SMALL CAPS, the letterspacing—that is, the spacing between the letters—should be strings of F U L L C A P I T A L L E T T E R S, and the `\smallcaps` command has letterspacing for SMALL CAPITAL LETTERS. These commands will also automatically convert the case of the text to upper- or lowercase, respectively.

The `\textsc` command has also been redefined to include letterspacing. The case of the `\textsc` argument is left as is, however. This allows one to use both uppercase and lowercase letters: The Initial Letters Of The Words In This Sentence Are Capitalized.

6.7 *Document Class Options*

The `tufte-book` class is based on the LaTeX `book` document class. Therefore, you can pass any of the typical book options. There are a few options that are specific to the `tufte-book` document class, however.

The `a4paper` option will set the paper size to A4 instead of the default US letter size.

The `sfsidenotes` option will set the sidenotes and title block in a sans serif typeface instead of the default roman.

The `twoside` option will modify the running heads so that the page number is printed on the outside edge (as opposed to always printing the page number on the right-side edge in `oneside` mode).

The `symmetric` option typesets the sidenotes on the outside edge of the page. This is how books are traditionally printed, but is contrary to Tufte's book design which sets the sidenotes on the right side of the page. This option implicitly sets the `twoside` option.

The `justified` option sets all the text fully justified (flush left and right). The default is to set the text ragged right. The body text of Tufte's books are set ragged right. This prevents needless hyphenation and makes it easier to read the text in the slightly narrower column.

The `bidi` option loads the `bidi` package which is used with XeLaTeX to typeset bi-directional text. Since the `bidi` package needs to be loaded before the sidenotes and citep commands are defined, it can't be loaded in the document preamble.

The `debug` option causes the Tufte-LaTeX classes to output debug information to the log file which is useful in troubleshooting bugs. It will also cause the graphics to be replaced by outlines.

The `nofonts` option prevents the Tufte-LaTeX classes from automatically loading the Palatino and Helvetica typefaces. You should use this option if you wish to load your own fonts. If you're using XeLaTeX, this option is implied (*i.e.*, the Palatino and Helvetica fonts aren't loaded if you use XeLaTeX).

The `nols` option inhibits the letterspacing code. The Tufte-LaTeX classes try to load the appropriate letterspacing package (either pdfTeX's `letterspace` package or the `soul` package). If you're using XeLaTeX with `fontenc`, however, you should configure your own letterspacing.

The `notitlepage` option causes `\maketitle` to generate a title block instead of a title page. The `book` class defaults to a title page and the `handout` class defaults to the title block. There is an analogous `titlepage` option that forces `\maketitle` to generate a full title

page instead of the title block.

The `notoc` option suppresses Tufte-LaTeX's custom table of contents (TOC) design. The current TOC design only shows unnumbered chapter titles; it doesn't show sections or subsections. The `notoc` option will revert to LaTeX's TOC design.

The `nohyper` option prevents the `hyperref` package from being loaded. The default is to load the `hyperref` package and use the `\title` and `\author` contents as metadata for the generated PDF.

7. Discussion and Outlook

The Tufte-LaTeX document classes are designed to closely emulate Tufte's book design by default. However, each document is different and you may encounter situations where the default settings are insufficient. This chapter explores many of the ways you can adjust the Tufte-LaTeX document classes to better fit your needs.

7.1 File Hooks

If you create many documents using the Tufte-LaTeX classes, it's easier to store your customizations in a separate file instead of copying them into the preamble of each document. The Tufte-LaTeX classes provide three file hooks: `tufte-common-local.tex`, `tufte-book-local.tex`, and `tufte-handout-local.tex`.

`tufte-common-local.tex` If this file exists, it will be loaded by all of the Tufte-LaTeX document classes just prior to any document-class-specific code. If your customizations or code should be included in both the book and handout classes, use this file hook.

`tufte-book-local.tex` If this file exists, it will be loaded after all of the common and book-specific code has been read. If your customizations apply only to the book class, use this file hook.

`tufte-common-handout.tex` If this file exists, it will be loaded after all of the common and handout-specific code has been read. If your customizations apply only to the handout class, use this file hook.

8. Conclusion

8.1 Tufte-LaTeX Website

The website for the Tufte-LaTeX packages is located at `http://code.google.com/p/tufte-latex/`. On our website, you'll find links to our SVN repository, mailing lists, bug tracker, and documentation.

8.2 Tufte-LaTeX Mailing Lists

There are two mailing lists for the Tufte-LaTeX project:

Discussion list The `tufte-latex` discussion list is for asking questions, getting assistance with problems, and help with troubleshooting. Release announcements are also posted to this list. You can subscribe to the `tufte-latex` discussion list at `http://groups.google.com/group/tufte-latex`.

Commits list The `tufte-latex-commits` list is a read-only mailing list. A message is sent to the list any time the Tufte-LaTeX code has been updated. If you'd like to keep up with the latest code developments, you may subscribe to this list. You can subscribe to the `tufte-latex-commits` mailing list at `http://groups.google.com/group/tufte-latex-commits`.

8.3 Getting Help

If you've encountered a problem with one of the Tufte-LaTeX document classes, have a question, or would like to report a bug, please send an email to our mailing list or visit our website.

To help us troubleshoot the problem more quickly, please try to compile your document using the `debug` class option and send the generated `.log` file to the mailing list with a brief description of the problem.

8.4 Errors, Warnings, and Informational Messages

The following is a list of all of the errors, warnings, and other messages generated by the Tufte-LaTeX classes and a brief description of their meanings.

```
Error:  \subparagraph is undefined by this class.
```

The `\subparagraph` command is not defined in the Tufte-LaTeX document classes. If you'd like to use the `\subparagraph` command, you'll need to redefine it yourself. See the "Headings" section on page 37 for a description of the heading styles available in the Tufte-LaTeX document classes.

```
Error:  \subsubsection is undefined by this class.
```

The `\subsubsection` command is not defined in the Tufte-LaTeX document classes. If you'd like to use the `\subsubsection` command, you'll need to redefine it yourself. See the "Headings" section on page 37 for a description of the heading styles available in the Tufte-LaTeX document classes.

```
Error:  You may only call \morefloats twice.  See the
        Tufte-LaTeX documentation for other workarounds.
```

LaTeX allocates 18 slots for storing floats. The first time `\morefloats` is called, it allocates an additional 34 slots. The second time `\morefloats` is called, it allocates another 26 slots.

The `\morefloats` command may only be called two times. Calling it a third time will generate this error message. See page 47 for more information.

```
Warning:  Option '⟨class option⟩' is not supported -- ignoring option.
```

This warning appears when you've tried to use ⟨*class option*⟩ with a Tufte-LaTeX document class, but ⟨*class option*⟩ isn't supported by the Tufte-LaTeX document class. In this situation, ⟨*class option*⟩ is ignored.

```
Info:  The 'symmetric' option implies 'twoside'
```

You specified the `symmetric` document class option. This option automatically forces the `twoside` option as well. See page 50 for more information on the `symmetric` class option.

8.5 Package Dependencies

The following is a list of packages that the Tufte-LaTeX document classes rely upon. Packages marked with an asterisk are optional.

- xifthen
- ifpdf*
- ifxetex*
- hyperref
- geometry
- ragged2e
- chngpage *or* changepage
- paralist
- textcase
- soul*

- letterspace*
- setspace
- natbib *and* bibentry
- optparams
- placeins
- mathpazo*
- helvet*
- fontenc
- beramono*
- fancyhdr
- xcolor
- textcomp
- titlesec
- titletoc

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

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

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

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

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

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

[24] 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.

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

Index

a4paper class option, 50
\author, 51

\bibliography, 46
bidi class option, 50
bidi package, 50
booktabs package, 47

\caption, 47
\citep, 46
class options, 50–51
  a4paper, 50
  bidi, 50
  debug, 50, 55
  justified, 50
  nofonts, 50
  nohyper, 51
  nols, 50
  notitlepage, 50
  notoc, 51
  oneside, 50
  sfsidenotes, 50
  symmetric, 50, 56
  titlepage, 50
  twoside, 50, 56
\clearpage, 48

debug class option, 50, 55
debug messages, 55

environments
  figure, 46
  figure*, 46
  fullwidth, 49
  marginfigure, 46
  margintable, 46
  table*, 46
  tabular, 46
error messages, 55

figure environment, 46
figure* environment, 46
file hooks, 53
  book, 53
  common, 53
  handout, 53
\FloatBarrier, 48
fontenc package, 50
\footnote, 39
\forcerectofloat, 49
\forceversofloat, 49
fullwidth environment, 49

headings, 28, 37
hyperref package, 51

justified class option, 50

letterspace package, 50
license, ii

\maketitle, 50
marginfigure environment, 46
\marginnote, 39, 46
margintable environment, 46
\morefloats, 48, 56

\newthought, 38
\nobibliography, 46
nofonts class option, 50
nohyper class option, 51
nols class option, 50
notitlepage class option, 50

notoc class option, 51

oneside class option, 50

packages
  bidi, 50
  booktabs, 47
  fontenc, 50
  hyperref, 51
  letterspace, 50
  soul, 50

\setfloatalignment, 48
sfsidenotes class option, 50
\sidenote, 39, 46
\smallcaps, 49
soul package, 50
\subparagraph, 56
\subsubsection, 56
symmetric class option, 50, 56

table* environment, 46
tabular environment, 46
\textsc, 50
\title, 51
titlepage class option, 50
tufte-book-local.tex, 53
tufte-common-handout.tex, 53
tufte-common-local.tex, 53
tufte-handout-local.tex, 53
twoside class option, 50, 56
typefaces, 49

warning messages, 55

XeLaTeX, 50