Customer Profitability Forecasting using Fair Boosting - an Application to the Insurance Industry

Mémoire

Alex St-Jean

Maîtrise en informatique - avec mémoire

Maître ès sciences (M. Sc.)

Québec, Canada

Customer Profitability Forecasting using Fair Boosting - an Application to the Insurance Industry

Mémoire

Alex St-Jean

Sous la direction de:

Mario Marchand, directeur de recherche

Résumé

La prévision de la profitabilité du client, ainsi que la tarification, sont des pièces centrales dans le monde des sciences actuarielles. En utilisant des données sur les historiques des clients et en optimisant des modèles statistiques, les actuaires peuvent prévoir, dans une certaine mesure, le montant qu'un client réclamera durant une certaine période. Cependant, ces modèles utilisent souvent des données sensibles reliées au client qui sont considérées comme étant des facteurs de risque très importants dans la prédiction de pertes futures. Ceci est considéré comme étant légal dans plusieurs jurisdictions tant que leur utilisation est supportée par des données actuarielles, car ces attributs permettent aux clients d'obtenir une prime plus précise. Toutefois, comme soulevé dans la littérature récente en apprentissage machine, ces modèles peuvent cacher des biais qui les rendent discriminants envers certains groupes. Dans ce mémoire, nous proposons un modèle de prévision de la profitabilité du client utilisant des avancées récentes provenant du domaine de l'apprentissage machine pour assurer que ces algorithmes ne discriminent pas disproportionnellement envers certains sous-groupes faisant partie de l'intersection de plusieurs attributs protégés, tel que l'âge, la race, la religion et l'état civil. En d'autres mots, nous prédisons équitablement la prime théorique de n'importe quel client en combinant l'état de l'art en prédiction de pertes en assurance et appliquant certaines contraintes d'équité sur des modèles de régression.

Suite à l'exécution de l'estimation de la profitabilité du client sur plusieurs jeux de données réels, les résultats obtenus de l'approche proposée sont plus précis que les modèles utilisés traditionnellement pour cette tâche, tout en satisfaisant des contraintes d'équité. Ceci montre que cette méthode est viable et peut être utilisée dans des scénarios concrets pour offrir des primes précises et équitables aux clients. Additionnellement, notre modèle, ainsi que notre application de contraintes d'équité, s'adapte facilement à l'utilisation d'un grand jeu de données qui contiennent plusieurs sous-groupes. Ceci peut être considérable dans le cas où un critère d'équité intersectionnel doit être respecté. Finalement, nous notons les différences entre l'équité actuarielle et les définitions d'équité provenant du monde de l'apprentissage machine, ainsi que les compromis reliés à ceux-ci.

Abstract

Customer profitability forecasting, along with ratemaking, are central pieces in the world of actuarial science. By using historical data and by optimising statistical models, actuaries can predict whether a client with certain liabilities will claim any loss and what amount will be claimed inside a defined policy period. However, these models often use sensitive attributes related to the customer that are considered to be crucial risk factors to consider in predicting future losses. This is considered legal in many jurisdictions, as long as their use is backed by actuarial data, as these attributes give a more accurate premium to clients. Nonetheless, as it has been noted in recent machine learning literature, models can hide biases that make them discriminate against certain groups. In this thesis, we propose a customer profitability forecasting model that uses recent advancements in the domain of machine learning to ensure that these algorithms do not discriminate disproportionately on a subgroup of any intersection of protected attributes, such as age, gender, race, religion and marital status. In other words, we fairly predict the theoretical premium of any client by combining state-of-the-art methods in insurance loss prediction and the application of fairness constraints on regression models.

After performing customer profitability estimation on multiple real world datasets, it is shown that the proposed approach outperforms traditional models usually used for this task, while also satisfying fairness constraints. This shows that this method is viable and can be used in real world scenarios to offer fair and accurate premiums to clients. Additionally, our model and our application of fairness constraints scale easily when using large datasets that contain many subgroups. This can be substantial in the case of satisfying an intersectional fairness criterion. Finally, we highlight the differences between actuarial fairness and fairness definitions in the world of machine learning, along with its related trade offs.

Contents

List of Tables

List of Figures

Whatever is my right as a man is also the right of another; and it becomes my duty to guarantee as well as to possess.

Thomas Paine

Remerciements

Tout d'abord, je tiens à remercier grandement mon directeur de recherche Mario Marchand pour m'avoir donné l'opportunité de collaborer avec lui sur ma maîtrise. Ses conseils, ses remarques et ses idées innovatrices m'ont permis d'approfondir mes connaissances, tout en allant toujours plus loin pour mieux vulgariser celles-ci. De plus, sa rigueur et son support ont été essentiels dans l'amélioration de ma démarche scientifique et de mon esprit critique.

Également, je tiens à remercier Mario de m'avoir permis de travailler dans un projet en partenariat avec une entreprise dans le but d'appliquer mes travaux à des cas d'utilisation concrets. La chance de participer à un partenariat avec SSQ, l'entreprise dans laquelle je travaille, m'a permis de développer un ensemble de compétences théoriques et pratiques sans égal.

Sur une note similaire, j'aimerais remercier Louise Bilodeau de m'avoir donné la chance de m'intégrer à l'équipe d'intelligence d'affaires de SSQ, sans laquelle cette maîtrise n'aurait pas été possible. D'ailleurs, sa coordination de ce partenariat du côté de SSQ a été faite avec une main de maître. J'aimerais également remercier Maxime Bélanger et Nicolas Dubuc pour nos discussions formatrices qui ont été essentielles dans ma compréhension du domaine de l'actuariat.

Je remercie Thierry Duchesne et Richard Khoury pour leurs commentaires sur mon mémoire. Leurs suggestions détaillées et leurs questions pertinentes m'ont permis d'améliorer davantage la qualité de ce mémoire.

Je remercie les membres du GRAAL pour leur aide et leur présence qui ont rendu le laboratoire une place agréable où travailler. Notamment, j'aimerais remercier Julien Laumonier, Jean-Thomas Baillargeon et David Beauchemin pour nos discussions sur les notions de tarification et d'équité en actuariat. Aussi, j'aimerais remercier Jean-Samuel Leboeuf, Gaël Letarte, Nicolas Garneau et Mathieu Alain pour leurs réponses à mes questions et leurs contributions à l'amélioration de mes compétences en LaTex.

À tous mes collègues faisant partie de ce partenariat : Fouad, Abdelali, Qi et Niffa, je vous souhaite du succès dans vos initiatives respectives. Ce fut un plaisir de vous côtoyer et j'espère pouvoir retravailler avec vous dans le futur.

Finalement, j'aimerais remercier mes parents, ma soeur et mes amis pour la motivation et le soutien qu'ils m'ont offert tout au long de mon parcours.

Introduction

Machine learning is a sub-domain of artificial intelligence that is often used for automated decision making. In businesses, its number of applications to accelerate existing processes using historical data is constantly increasing. By optimising a function properly to predict future outcomes using past outcomes, one can expect to reproduce a past behaviour. For example, these algorithms can be used for regression or classification tasks. Classic instances of classification tasks involve assigning the most probable label to a new item (e.g., image labelling, email categorisation and cancer detection). As for regression problems, these involve predicting a real number. This can include determining the potential grade of a student or predicting the price of a house.

Despite all the enthusiasm that surrounds this subject, many concerns have risen from the use of machine learning algorithms for automated decision making. As machine learning algorithms have made their way into many regulated domains, these algorithms must be evaluated as they have an impact on society. The externalities that come with the use of machine learning algorithms bring new concerns into decision making as a whole. The outcomes produced by these algorithms can sometimes discriminate towards parts of a population on the basis of gender or ethnicity. Of course, no ill or explicitly bad intention was considered to be the root cause of these negative biases. Often, the historical data that is used to train these algorithms can be biased. As such, the algorithm perpetuates and amplifies these unwanted implicit biases onto future results.

Common examples of regulated tasks that can use machine learning include risk recidivism prediction, loan approval, university acceptance and job hiring. These decisions need to be monitored for crucial reasons: they can negatively affect the livelihood of people. That includes people that are the most vulnerable. If an automated decision system discriminates against a certain group, it affects whole communities as much as it being done by a real person when it is based on unfounded biases. It perpetuates negative feedback loops on an even larger scale, as it can predict outcomes on millions of different cases in a matter of seconds. That is why government bodies must ensure fairness on these tasks. This is problematic, as these entities are not equipped with the proper tools to measure and enforce fairness on different outcomes.

Another known task supervised by a regulatory body is ratemaking for the insurance industry.

Estimating theoretical insurance premiums, along with customer profitability, is indispensable for providing a different set of protections to a client according to their specific needs and risks. As much as it is important for insurance providers to stay profitable, clients need to be insured without the presence of disloyal or abusive practices. An insurer giving unnecessarily high or inaccurate premiums to a subgroup effectively prevents them from being protected from catastrophic events, which increases their risk of financial hardship. Hence, if a biased algorithm is used for this task, then undue harm might disproportionately be put upon vulnerable communities.

In this thesis, we outline a framework for insurance companies and regulatory agencies alike to measure and ensure fairness when performing customer profitability estimation in the insurance industry. By intersecting many fields such as actuarial science, law, machine learning and philosophy, we propose guidelines that permit the existence of machine learning that comply with anti-discrimination laws, that minimises price changes to clients and that keep insurance companies profitable. These guidelines include the encoding of sensible variables, the selection of a metric that can be used to measure unfairness in this context and the efficient optimisation of a gradient boosted tree model that satisfies both performance and fairness constraints as much as possible. As these objectives often come at odds, we discuss and compromise on these subjects. We also apply those proposals by integrating them into state-of-the-art prediction models and by putting them in real use cases.

Chapter 1 presents preliminary notions that are essential in understanding the following chapters of this thesis. These prerequisites include an introduction to supervised learning, which covers classification and regression problems. Afterwards, we introduce common linear models that are used to solve these problems, as well as the different functions that can be optimised according to the target objective. Finally, we add an analysis of tree boosting models, which combine many weak learners to create a strong one.

Chapter 2 introduces and reviews the current state of fairness in machine learning. We begin by listing the many definitions that encompass fairness in this domain. Then, we show how different processes can be applied to automated decision making pipelines to satisfy these definitions. These processes differ in their placement in the decision making process and in their application with regards to the algorithm that is used. We finish with the definition of fairness that exists in actuarial science and the important distinctions that exist with machine learning in that regard.

Chapter 3 directly addresses the main problematic of this thesis, which is to fairly forecast customer profitability in the insurance industry using fair boosting. We formally present the task at hand and its relation to insurance pricing. We then go through all of the different concerns that should be taken into consideration when ensuring algorithmic fairness. This includes selecting the right fairness criterion, the proper method to satisfy this criterion, the intersectional encoding of sensible attributes and the appropriate algorithm to tackle this problem. Ultimately, we combine all of the selected methods to optimise fairness and performance objectives simultaneously.

Chapter 4 explores various practical results obtained from the method presented in the previous chapter. We begin by defining the methodology by which we train our model to ensure that it is optimal and fair. We later employ this procedure by using a real-life home insurance claim dataset. We show through empirical results that our proposal permits a more precise and fair risk assessment while using sensible attributes.

We conclude this thesis by reflecting on the qualities and shortcomings of this work, while also exploring future work that could be done to ensure fairness in the insurance industry.

Chapter 1

Background Notions

In its most basic form, algorithms derived from the domain of machine learning learn in a generic way how to perform a certain set of tasks from data. Using a set of previous observations of an unknown distribution, an algorithm tries to detect tendencies contained in its inputs in a way that can be applied on new ones. Each instance of the input data contain a collection of one or multiple quantified attributes that are proper to them. These features give out information that can be used to relate known instances to a real value, a label, a structure or a cluster of similar instances. Thus, we are able to link these trends on new items. Altogether, a machine learning algorithm is defined by the target task and the function that is being optimised to solve the problem at hand. This chapter focuses on these models in the context of regression problems.

1.1 Supervised Learning

The most common type of learning task in machine learning is supervised learning. In this case, the goal is to learn the function that relates a set of inputs to a set of labels. This relation is defined as a mapping $\mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{X}$ is the space of possible inputs and $\mathcal{Y}$ is the space of possible labels. It is assumed that this relation is generated from a distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$, where all samples are independent and identically distributed. Since we do not know the exact mapping between observations and labels, we hope to use a representative set of labeled samples randomly drawn from $\boldsymbol{S} \sim \mathcal{D}^n$ to learn the most probable one. $\boldsymbol{S}$ is composed of n tuples of known observations and labels $(\boldsymbol{x}_i, y_i)$, where x_{ij} is the j^{th} component of M real-valued attributes of the i^{th} row of known inputs $\boldsymbol{X}$ and y_i is the i^{th} row of known real-valued labels (i.e. outcomes) $\boldsymbol{Y}$. To represent standalone elements, x_j is the j^{th} attribute for any chosen instance $\boldsymbol{x}$ with outcome y and prediction $\hat{y}$. By using $\boldsymbol{S}$, a function $f(\boldsymbol{x}|\boldsymbol{\theta})$ is estimated. This function contains the parameters $\boldsymbol{\theta}$ that represent best the relation between $\boldsymbol{X}$ and $\boldsymbol{Y}$ for any observation $\boldsymbol{x}$. Afterwards, new outcomes are established by calculating the most likely label or value $\hat{y} = f(\boldsymbol{x}|\boldsymbol{\theta})$ of a new $\boldsymbol{x}$.

For regression problems specifically, $f(\boldsymbol{x}|\boldsymbol{\theta})$ models the relationship between observations and real-valued outcomes. For example, a specific regression task could be to predict the salary y of an individual $\boldsymbol{x}$ by using different attributes x_j related to it. These attributes include the person's age, sex, race and education level. The model would then be optimised by using historical data $\boldsymbol{X}$ containing a list of other people having these features and their own salaries $\boldsymbol{Y}$. A new individual could then have his/her salary $\hat{y}$ approximated by using $f(\boldsymbol{x}|\boldsymbol{\theta})$.

On the other hand, classification algorithms learn to categorise observations based on previously labeled data. If the task of predicting an individual's salary becomes the task of predicting if a person makes over or under 50,000 dollars per year, then there are only two possible outcomes. By attributing the label $y = 0$ to people making under 50,000 dollars and $y = 1$ to the ones making over 50,000 dollars, an algorithm can learn to apply the most probable label to subsequent individuals.

1.1.1 Objective functions

Multiple methods exist to approximate efficiently the best parameters describing the trends in $\mathcal{D}$. As defined previously, the goal of supervised learning is to find the best function that defines the relation between $\mathcal{X}$ and $\mathcal{Y}$. That objective corresponds to finding the function that minimises the expected loss $L_{\mathcal{D}}$, or **true risk**. The expected loss represents how far the learned function is from the true relation on the entire space of $\mathcal{X} \times \mathcal{Y}$. Hence, this objective could be written as Equation 1.1, where L is the selected loss function. The **loss function** L quantifies the discrepancy between a single label y and the predicted outcome $f(\boldsymbol{x})$.

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L_{\mathcal{D}}(f) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \underset{(\boldsymbol{x},y)\sim\mathcal{D}}{\mathbb{E}}[L(y, f(\boldsymbol{x}|\boldsymbol{\theta}))]. \tag{1.1}$$

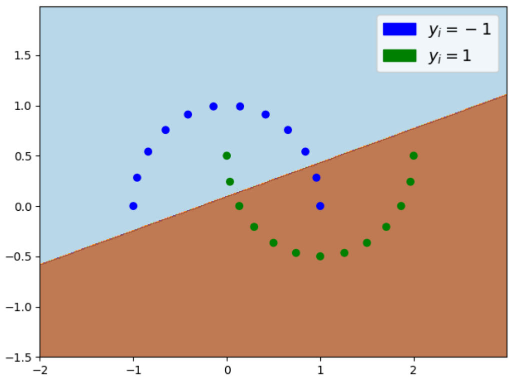
However, it is impossible to solve this objective directly, as we only have access to samples in set $\boldsymbol{S}$. That is why instead of optimising the expected risk, the objective focuses on empirical risk. The **empirical risk** L_S is equal to the average error that is between $\hat{y}$ and y on the training set $\boldsymbol{S}$. The resulting objective is equal to Equation 1.2.

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L_S(f) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i|\boldsymbol{\theta})). \tag{1.2}$$

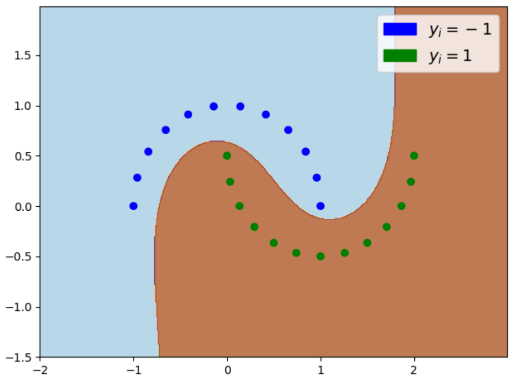
The choice for $L(y, \hat{y})$ depends on what is better adapted for the problem that needs to be solved. It also depends on what observations are the most critical in solving the target task. By minimising the empirical risk, we hope to find the best $\boldsymbol{\theta}$ that produces the model that has the smallest true risk possible.

1.1.1.1 Basic loss functions for classification

In order to understand the concept of loss functions in a more practical setting, it is essential to present loss functions in their most basic forms. Further examples of these objectives, along with different instances where their usage is required, are presented in this section.



(A) Simple linear classifier



(B) Complex classifier

FIGURE 1.1 – Example of different classifiers on a binary classification problem

The most basic loss function for classification problems is the zero-one loss:

$$L(y, \hat{y}) = I(\hat{y} \neq y), \text{where } I(a) = \begin{cases} 1 & \text{if } a = \text{True}, \\ 0 & \text{if } a = \text{False}. \end{cases} \tag{1.3}$$

Consider a dataset containing $n = 25$ observations $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_{25})$ that each have two attributes (x_{i1}, x_{i2}), where x_{i1} denotes the positioning of the *ith* observation with regards to the x-axis and x_{i2} denotes the position of the element with regards to the y-axis. As seen in Figure 1.1 elements are shaped in two moons that cannot be linearly separated, each are assigned a class $y_i \in \{-1, 1\}$ according to their respective moon. We train a simple linear and a more complex discriminant $f(\boldsymbol{x})$ to label future elements using our previous observations. To evaluate the zero-one loss on the training set as seen on Figure 1.1, we can simply count the number of elements that are on the wrong side of the classifiers' decision boundary and divide it by the total number of labeled elements. As elements inside the light blue zone are classified as $y_i = -1$ and elements inside the light brown zone are classified as $y_i = 1$, we can count that two of each group's elements are misclassified by the classifier. As such, the first figure has an expected loss of $\frac{4}{25} = 0.16$, while the second one perfectly classifies all the observations. Hence, we can assume that the model with the lowest loss will probably label more accurately future observations.

Another way to quantify the empirical risk of a classifier is to use hinge loss:

$$L(y, \hat{y}) = (1 - y\hat{y})_+ = \max(0, 1 - y \cdot f(\boldsymbol{x})), y \in \{-1, 1\}. \tag{1.4}$$

Instead of quantifying the rate at which an item is assigned to the wrong class, we estimate a distance between the predicted value and the true label. This distance is calculated by multiplying the true label y with the predicted score $\hat{y}$. This results in a loss that can only be decreased if both of the elements have the same sign. By using the model's initial output,

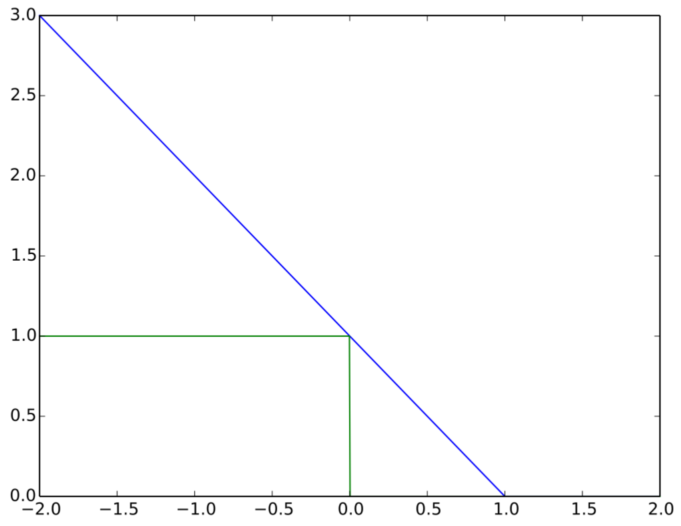


FIGURE 1.2 – Hinge loss (blue) compared to zero-one loss (green).

it is possible to determine how far off the classifier is to having the right result. By putting a lower bound to this loss to zero, we make sure that if the $f(\boldsymbol{x})$ predicts exceedingly well the correct label (over 1 or under -1), the model is not rewarded. On the other hand, exceedingly wrong outputs are not capped and are more severely penalised. This makes a model focus more on elements that are far from being correctly labeled, more than ones which are close to being correct.

If we come back to the previous example where a simple classifier makes four mistakes and another one perfectly classifies the elements based on their predicted labels, hinge loss gives a similar result but in a different manner. Instead of giving out the number of errors, it shows when some instances are closer to being errors and when they are errors, it shows how far they are to being correct. So for Figure 1.1, the predicted values in a) can cause predictions that are way off with a loss of 0.40, while b) has a loss of 0.05. Note that b) does not have a null loss even if all elements are correctly labeled, since some elements are too close to the decision boundary.

As shown in Figure 1.2, the hinge loss and the zero-one loss can be easily compared side to side in a single graphic. While zero-one loss incurs no cost to the classifier if the proper label is applied, the hinge loss continues to consider a loss until the exact label is obtained without using a threshold to remove any uncertainty near the decision boundary.

Depending on the use case, it is important to choose the proper loss function for the task at hand. If we were to consider one of the two losses previously presented, both of them quantify the capacity to correctly label elements. In a case where we want a classifier to make the least number of errors possible, we can use the zero-one loss. Otherwise, if an algorithm with the average minimal distance between each y and $\hat{y}$ is needed, the hinge loss would be a better candidate. There is no loss function that is the best for all problems. Nonetheless, the hinge loss is useful in cases where a convex and differentiable proxy to the zero-one loss is required.

1.1.1.2 Basic loss functions for regression

When applied to regression problems, loss functions such as the hinge loss are ill-suited. Since these objectives consider discrete labels and decision boundaries established in a $\mathcal{R}^n$ space, they would give out poor results where the predicted label is continuous. As such, other distance measures must be employed in order to create regressors.

Since the predicted label and the target number are now floating point numbers, an appropriate way to establish their discrepancies is to compute the distance between them. The main difference with the hinge loss is that there is no need for a minimal boundary if the algorithm is certain about its prediction. In a regression problem, a predicted outcome could be too low, too high or exactly equal to the target label. To know how close those two points are, we must not only determine their differences, but also the importance of that distance towards solving the objective. For example, in some cases, not all distances are treated in a directly proportionate manner to their size. Most of the distances contained in loss functions are computed differently, but contain desired properties, such as being always positive and convex.

One of the most simple ways to determine the distance between the real and predicted outcome is to calculate the absolute difference between them,

$$L(y, \hat{y}) = |\hat{y} - y|. \tag{1.5}$$

By taking the mean of the positive distance between predictions and outcomes, we obtain the **absolute error**, which can be used to attribute and compare the performance of regression models. This loss has the advantage of being the most simple and intuitive, as its distance is determined by subtracting y from $\hat{y}$. It is important to note that in this case, all instances have their errors considered as being of equal importance.

Another way to determine regressor performance is to calculate the squared error separating expected outcomes with results:

$$L(y, \hat{y}) = (\hat{y} - y)^2. \tag{1.6}$$

Instead of determining the loss according to the absolute distance, the squared distance is used. In return, higher errors are amplified and represented in bigger proportion in the **squared error**.

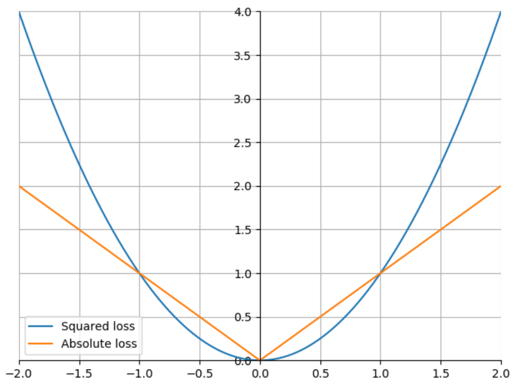
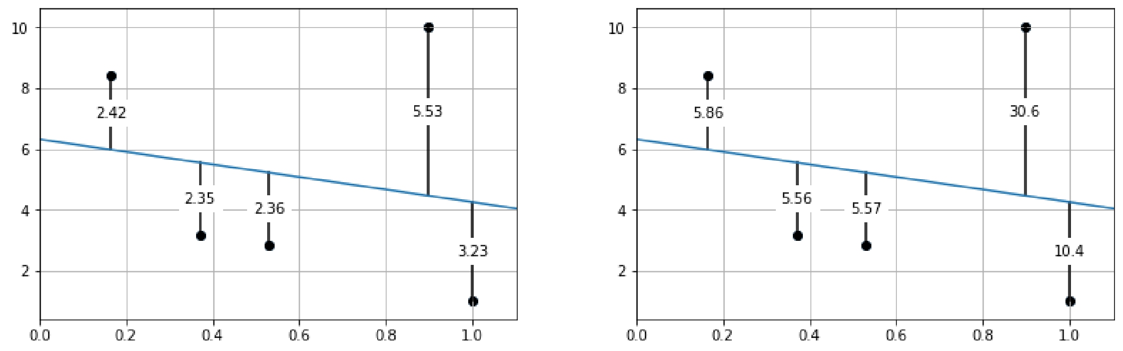


FIGURE 1.3 – Regression losses in relation to the distance between y_i and $\hat{y}_i$



(A) Absolute error (B) Squared error

FIGURE 1.4 – Example of a regressor and its errors

Figure 1.3 shows how fast each loss grows according to a residual (1.7).

$$r = y - \hat{y} \tag{1.7}$$

As squared loss increases in a quadratic way and absolute loss increases in a linear way, they obviously output different results and are useful in different cases. The absolute loss' penalties grow in a linear fashion, thus being higher than squared error if the absolute difference is under 1. Otherwise, if the absolute residual is over 1, the squared error's growth will result in giving it more importance.

In order to compare both of these objectives, a simple linear regression can be produced using the same random data points, while only comparing the effect each error has on the total mean error.

As it can be determined from Figure 1.4, using different loss functions allows to compare different models, while weighting each observation differently in the comparison. Because the squared error emphasises higher errors, it will also give greater importance to outliers. For example, the next to last data point has the highest error for both objectives, but does not represent the same proportion of the error. In Figure 1.4 a), the total sum of the error is 15.89, the mean error is 3.18 and the maximum error is of 5.53, which gives it 34.8% of the total error. On the other hand, in Figure 1.4 b), the error sums up to 58, the average loss is 11.6 and the maximum error is of 30.6. This same residual now accounts for 52.8% of the total discrepancies. Hence, comparing regressors using the mean squared error will reward being precise on extreme values, while mean absolute error will reward being precise equally on each instance. The trade off lies in the value of performance with outliers.

1.1.2 Model comparison and selection

Using losses to assess and compare model performance can be useful in cases where they are used on different models in the same context. They can easily give insight into model performance by knowing that a lower or higher value for a certain metric is better. However, these metrics must be taken with a grain of salt if they do not take into consideration the context in which they are used. In order to make qualitative assertions about the real world performance of a machine learning algorithm, one must find metrics that are proper to the task at hand. For regression problems, this means that error metrics must be scaled properly to give real insight into model behaviour if the errors are always abnormally high in a certain context. All relevant metrics, along with the context of application, must be considered as the whole portrait of an algorithm's performance and must not be taken individually. Altogether, they can be useful when translated into business terms that can transmit the value of using a specific model in a production setting.

1.1.2.1 Metrics for classification problems

In a simple binary classification problem, no single metric can be applied to any business problem to determine a satisfactory level performance. That is because the context in which an algorithm is applied matters.

A common way to represent most of the classification metrics is to place them into what is called a **confusion matrix**, as shown in Table 1.1. A confusion matrix simply lays out where each element was classified according to its true label. In other words, in the case where there are two possible outcomes, the goal would be to maximise the numbers in the green downwards diagonal, while minimising the numbers in the upwards red diagonal. In the case where all elements were correctly labeled by an algorithm, this would mean that the true elements would be counted in the upper left quadrant, that all the false elements would be counted in the bottom right quadrant and that the remaining two quadrants wouldn't have any elements placed in them.

To illustrate this, we can go back to the classification problem presented in Figure 1.1 a). In this case, twenty-five elements were used as training data. If we consider that elements with $y = 1$ are in the "positive" class and that instances with $y = -1$ are the "negative"

		True label	
		Positive	Negative
Predicted label	Positive	True positives (TP)	False positives (FP)
	Negative	False negatives (FN)	True negatives (TN)

TABLE 1.1 – A confusion matrix

ones, thirteen of them are positive, while twelve of them are negative. According to the decision boundaries of the simple classifier, of the thirteen positive elements, eleven of them are considered to be positive, while two of them are labeled as negative. On the other hand, ten of the negative elements are in the correct section, whereas two are incorrectly categorised. This would result in eleven true positives, ten true negatives, two false positives and two false negatives.

The confusion matrix is useful as it can be used in the future to calculate more complex and representative performance metrics. The most intuitive metric to extract out of this is the **accuracy** (1.12). Accuracy represents the rate at which instances of an overall distribution are placed in the right category by a classifier. For example, if we take into consideration the previous problem, out of 25 elements, 21 are properly labeled by the simple classifier. This would then mean that the algorithm has a rate of success of $\frac{21}{25} = 0.84$, or 84%.

$$\text{TPR/Recall/Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.8}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{1.9}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = 1 - \text{FPR} \tag{1.10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{1.11}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{1.12}$$

As discussed previously, no metric is perfect to assess performance in every context. If we take into consideration a problem where extremely rare events need to be detected, accuracy could not portray an accurate depiction of classifier performance to do a certain task. For instance, in order to find fraudulent transactions from millions of valid transactions, an algorithm must be capable of determining the criteria that can single out these odd occurrences. The problem here would lie in the fact that even if a model detects no anomalies, its accuracy would still be near perfect. Supposing that there are 4% of transactions that are fraudulent and that a model always returns "negative", it would still get a 96% accuracy score.

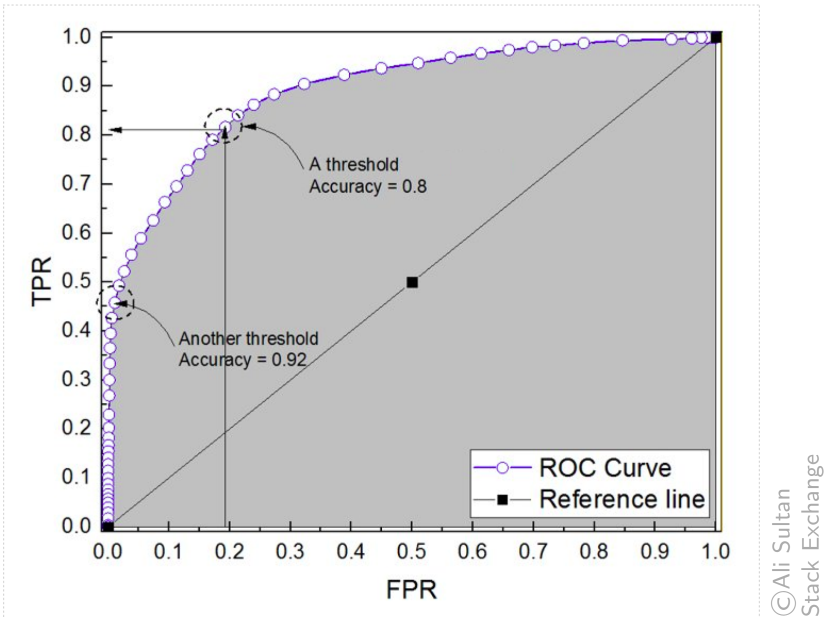


Figure 1.5 – Example of a ROC Curve [59]

To evaluate classifier performance on cases containing unbalanced data where an event occurs rarely, a metric such as the **recall** (1.8) can be of great use. Recall quantifies the rate at which an algorithm can detect the overall "positive" events. In the case of fraud detection, the true positive rate would determine the percentage of overall frauds that are detected. If 2 out of 4 total frauds are detected, then the true positive rate would be of 50%. On the other hand, recall is not useful in all cases. If an algorithm determines that all transactions are problematic, it would have a perfect recall, but most of the flagged transactions wouldn't be problematic. This would result in many clients being suspected of a crime they didn't commit. That is why the **false positive rate** (1.9) is a great metric to counterbalance recall. The false positive rate quantifies the rate at which flagged elements are truly negative. Given a situation that results in dire consequences when an instance is attributed the "positive" class, it is wise to ensure low FPR, as we can consider that flagged elements have a low chance of being "negative".

To represent the trade-off that occurs between the false positive rate and recall, a graphic commonly known as the ROC Curve (Receiver operating characteristic), as shown in Figure 1.5, is useful in showing how much recall increases when flagging more instances and how much the false positive rate decreases when flagging fewer instances. The higher the area under the curve, the better the classifier tends to perform. The ROC Curve can also be used to determine the optimal threshold to obtain a certain true or false positive rate, depending on the target objective.

1.1.2.2 Metrics for regression problems

With regards to problems requiring the prediction of continuous values, results cannot be put in a simple matrix. As outcomes vary in very different ways depending on their scale and their distribution, they cannot be put in discrete boxes. This makes the use of classification metrics difficult to apply. In return, the evaluation metrics for regression do not require the exact prediction, but accurate responses still result in minimal loss. Their goal is not only to address model performance, but also target complexity and explainability.

An interesting way to represent model capability relative to the task at hand is to use the total sum of squares of the outcomes (1.16). The total sum of squares (SST) is equal to the sum squares of each label in $\boldsymbol{Y}$, but can also be decomposed in a way that determines how much a model can explain the variation of results with its current parameterization and predictions:

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{1.13}$$

$$\bar{x}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n} \tag{1.14}$$

$$SST = SSE + SSR \tag{1.15}$$

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{1.16}$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1.17}$$

As shown in Equation 1.15, SST can be split into two components : the sum of squared estimates of error (1.17) and the sum of squares due to regression. The SSE of a regressor is the sum of the squared residuals between the prediction $\hat{y}$ and the observation's true value y. In other words, the SSE is the unaveraged mean squared error found in 1.6. As its counterpart, SSR represents the part of the variation of $\boldsymbol{Y}$ that is not a residual : it considers how much an algorithm performs better than simply using $\bar{y}$ (i.e. the mean of $\boldsymbol{Y}$) as a response. In the end, it shows how much a model can explain the variability in $\boldsymbol{Y}$.

As standalone absolute values, SST, SSE and SSR are not meaningful metrics because they increase naturally with the number of observations in a dataset. In order to scale these results **R-squared** (1.18) produces a measure of relative performance that gives a meaning to the previous absolute ones:

$$R^2 = 1 - \frac{SSE}{SST} = \frac{SSR}{SST}. \tag{1.18}$$

In short, the coefficient of determination, or R^2 metric, represents the proportion of the variability of $\boldsymbol{Y}$ that is explained by the predictions of the present model. By returning a proportion, it can be used on any regression problem. It also scales model improvement over the use of the mean, where the SST would be equal to the SSE. As an example, the problem presented in Figure 1.4 b) can be reused to illustrate its use. The calculated SST in this case is of 60.24, the SSE of 58.14 and the SSR of 2.1. By inserting these values in 1.18, a value of 0.035 is computed, which means that 3.5% of the variation of $\boldsymbol{Y}$ is explained by the regressor.

Another way to scale an error metric to more explainable terms for model comparison is to use the RMSE (Root mean squared error),

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}. \tag{1.19}$$

In itself, the mean squared error often gives high numbers that do not represent by how much a predictor is off in absolute terms. By applying a square root to this metric, Equation 1.19, it is easier to determine by how much the predictor will be off the mark when producing outcomes from new observations. It is still influenced by higher errors, but it is now presented in absolute units instead of squared units.

1.1.2.3 Model selection

Model selection is an essential part in the process of creating a real-world machine learning solution that not only facilitates the optimisation of the best $\boldsymbol{\theta}$ for $f(\boldsymbol{x})$, but also ensures that the selected function's type of model works well on an entire distribution, instead of only on a training set.

Furthermore, this step in a project's life cycle takes into consideration a model's **hyperparameters**. This type of parameter, which is not included in $\boldsymbol{\theta}$, is constant throughout its training. These hyperparameters are set before optimisation and alter algorithm complexity, convergence, loss function, etc. A common use of hyperparameters is to determine the rate at which $\boldsymbol{\theta}$ is adjusted with regards to the chosen loss function. Another one could be the number of relevant features to use in the algorithm. More practical applications of hyperparameters will be presented in the next sections.

The establishment of many metrics to evaluate models facilitates the comparison of models, which now gives way to an easier way to select good models. An important caveat to this is that these metrics have to be used properly to keep their credibility. For instance, a metric that is calculated on training observations does not guarantee that the model will perform similarly on new observations, as the training observations are in the known distribution of elements. That is why it is also crucial to have a proper training methodology to select the best model to use on new observations.

The most simple way to properly validate a certain model would be to randomly partition a dataset into three sections : a training set, a validation set and a test set. The splitting of $\boldsymbol{S}$ into these sections ensures that the algorithm with certain hyperparameters can generalise on other sets of data. Training data usually takes between 50% and 80% of the data, while the rest is divided into a validation and a test dataset. Obviously, the training data is used to train a model and to observe tendencies. On the other hand, the validation data is used to assess model performance on unknown entities and is used to compare models. Finally, the

	Split 1	Split 2	Split 3	Split 4	Split 5
Iteration 1	Validation	Training	Training	Training	Training
Iteration 2	Training	Validation	Training	Training	Training
Iteration 3	Training	Training	Validation	Training	Training
Iteration 4	Training	Training	Training	Validation	Training
Iteration 5	Training	Training	Training	Training	Validation

TABLE 1.2 – Example of a 5 fold cross-validation.

	Year 1	Year 2	Year 3	Year 4	Year 5
Iteration 1	Training	Validation			
Iteration 2	Training	Training	Validation		
Iteration 3	Training	Training	Training	Validation	
Iteration 4	Training	Training	Training	Training	Validation

TABLE 1.3 – Example of a 5 year time series cross-validation.

test set is used on the selected model to guarantee model performance on data that was never used for training or evaluation. The distinction here lies in the fact that using a validation set too often to evaluate model performance often creates a function that is optimised only for the validation set. The test set should preferably be only used once at the end of model selection to have an unbiased estimate of the chosen model.

The previous method presented is traditionally used to solve machine learning problems, but is still ineffective in most cases. To prevent the over usage of a validation set, $\boldsymbol{S}$ can be randomly split into k subsets of equal size. The **k-fold** cross-validation algorithm trains k times the same model using the same hyperparameters, but changes the validation and training set every time. As shown in table 1.2, for each combination of splits possible, we select $k-1$ sets as training data for our algorithm and we validate the resulting function with the validation set. In the end, by calculating the average of each metric and loss on each validation set, it is possible to get a proper assessment of the model and its hyperparameters over an entire dataset. Increasing k augments computation time, but also gives out a better insight into model performance when using subsets that resemble the entire dataset. A test set should still always be put aside to ensure that the selection of the final model is not only based on its capacity to perform on the cross-validation, but also on new observations that are not considered in the choice of model and hyperparameters. When the best combination of hyperparameters is found, the model can be refit with the entire training set to maximise performance.

K-fold cross-validation is a great method for validating models, but makes the crucial assumption that instances in $\boldsymbol{S}$ must be independent and uncorrelated to illustrate real-world classifier performance. As a counter-example, multiple observations that are taken on the same individual over time are not independent. In this case, k-fold cross-validation does not take into consideration that outcome distributions changes over time and that performance is essential for future observations more than past ones. As a result, **time series cross-validation** (TSCV) was created to take into consideration model performance over time. As shown in Table 1.3 and Algorithm 1.1, TSCV splits a dataset into k time periods, instead of random folds. In short, by ordering the splits by time period, this method sequentially adds data from a new period to the training set and computes validation metrics with elements from the next year. Hence, this methodology produces robust models and configurations that perform well over time on dependent observations.

Algorithm 1.1 *Time series cross-validation*

```
1. Set validation error $E = 0$.
2. For $i = 1$ to $k - 1$:
    a) Set training data $\mathcal{T}$ as $(\boldsymbol{x}, y) \in \boldsymbol{S}$ between periods 0 and $i$.
    b) Set validation data $\mathcal{V}$ as $(\boldsymbol{x}, y) \in \boldsymbol{S}$ at period $i + 1$.
    c) Fit a model $f(\boldsymbol{x})$ using training data $\mathcal{T}$.
    d) Validate the model using $\mathcal{V}$.
    e) Add the validation error to $E$.
3. Output $\frac{E}{(k-1)}$.
```

Grid-search is a methodology that attempts to find the best model and the optimal model configuration. For each specific type of function that needs to be assessed, we select all hyperparameters that may be of interest to it. For each hyperparameter, we then choose a list of values that may be close to the ideal one. These values can be established by domain experts, by rules of thumb for a specific algorithm or randomly if none of the above is applicable. This results in a grid of hyperparameters to be submitted to cross-validation. As such, each possible combination of hyperparameters is used on their target algorithm for cross-validation using the k-fold or time series window methods.

Choosing the proper cross-validation methodology is essential because it confirms two things: the selected algorithm is the best performing one overall according to the chosen metrics and the algorithm does not overfit on the training set. **Overfitting** is a problem that arises when an algorithm learns a training set too well, which leads to poor generalisation. For example, consider an algorithm that predicts the label of an observation by selecting the class of the training example which is the most similar that observation. It will have a perfect score on the original data, but will often have high validation loss, since it has learned most of the noise contained in the training set. Instead, models should be built to detect general tendencies that can be applied on new data. However, models that lack variance and complexity will also underperform on new data, as no particular tendency will have been detected. That is why cross-validation inherently aims to balance model complexity.

1.2 Linear Models

Linear models and their family of functions are some of the simplest types of models that can be easily understood as they use a unique vector of independent coefficients $\boldsymbol{\beta}$ as parameters. In the most basic cases, each coefficient is assigned an attribute that, when multiplied, shows the linear relation between the attribute and the average outcomes. Altogether, the multiplication of each coefficient with its attribute represents the independent impact of each attribute on a prediction. For the prediction of continuous values, one of the most straightforward linear models is the simple linear regression, where there is only one observed attribute:

$$f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) = f(\boldsymbol{x}|\hat{\boldsymbol{\beta}}) = \hat{\beta}_1 x_1 + \hat{\beta}_0. \tag{1.20}$$

For this particular model, the parameters $\hat{\boldsymbol{\beta}}$ represent the estimated coefficients $\hat{\beta}_1$ and $\hat{\beta}_0$. When making a prediction, $\hat{\beta}_0$ is the function's intercept, which is the function's value if $x_1 = 0$. $\hat{\beta}_1$ is the average variation of y in relation to the increase of x_1 by one unit. An attribute x_0 associated with $\hat{\beta}_0$ is implicitly present to add the intercept to the final prediction, but isn't indicated in the equation as it always has a value of 1.

To obtain the estimation of the coefficients that represents the relation between $\mathcal{X}$ and $\mathcal{Y}$, we must look to the model's loss function, which is traditionally the squared error. By putting Equation 1.6 and 1.20 into 1.2 the objective then becomes Equation 1.21:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L_S(f) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \underset{\hat{\boldsymbol{\beta}}}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^{n}[y_i - (\hat{\beta}_1 x_{i1} + \hat{\beta}_0)]^2. \tag{1.21}$$

In order to solve Equation 1.21, one must look to Figure 1.3 to gather intuition on how to minimise squared loss. Since we know how to calculate the loss, we can also know its gradient by obtaining its differentiation with respect to each parameter in the regressor. As long as we know that the loss function is convex, it is possible to know the parameters at which the residuals stops going down. When the gradient is equal to zero, the estimated parameters constitute the function which has reached the minimum possible loss. Equation 1.24 is an example of the calculations required to get the proper estimation of $\hat{\beta}_0$, and Equation 1.25 provides the result of the same process for $\hat{\beta}_1$. In these equations, $\bar{x}_j$ refers to the average value of the j^{th} attribute contained in set $\boldsymbol{X}$.

$$\begin{aligned}
\sum_{i=1}^{n}(y_i - \bar{y})(x_{i1} - \bar{x}_1) &= \sum_{i=1}^{n} y_i x_{i1} - \sum_{i=1}^{n} \bar{y} x_{i1} - \sum_{i=1}^{n} y_i \bar{x} + \sum_{i=1}^{n} \bar{x}\bar{y} \qquad (1.22)\\
&= \sum_{i=1}^{n} y_i x_{i1} - \sum_{i=1}^{n} \bar{y} x_{i1}\\
&= \sum_{i=1}^{n} (y_i - \bar{y}) x_{i1}
\end{aligned}$$

$$\sum_{i=1}^{n}(x_{i1}-\bar{x}_1)^2 = \sum_{i=1}^{n}(x_{i1}-\bar{x}_1)x_{i1} \tag{1.23}$$

$$\begin{aligned}
\frac{\partial L_S(f)}{\partial \hat{\beta}_0} &= \sum_{i=1}^{n} \frac{\partial}{\beta_0}(y_i-\hat{y}_i)^2 \\
0 &= \frac{\partial}{\partial \hat{\beta}_0} \sum_{i=1}^{n}(y_i-\hat{\beta}_1 x_{i1}-\hat{\beta}_0)^2 \\
0 &= -2\sum_{i=1}^{n}(y_i-\hat{\beta}_1 x_{i1}-\hat{\beta}_0) \\
\sum_{i=1}^{n}\hat{\beta}_0 &= \sum_{i=1}^{n}(y_i-\hat{\beta}_1 x_{i1}) \\
n\hat{\beta}_0 &= \sum_{i=1}^{n}(y_i-\hat{\beta}_1 x_{i1}) \\
\frac{n\hat{\beta}_0}{n} &= \frac{1}{n}\sum_{i=1}^{n} y_i - \frac{1}{n}\sum_{i=1}^{n}\hat{\beta}_1 x_{i1} \\
\hat{\beta}_0 &= \bar{y} - \hat{\beta}_1\bar{x}_1. \qquad (1.24)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L_S(f)}{\partial \hat{\beta}_1} &= -2\sum_{i=1}^{n} x_{i1}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_{i1}) \\
0 &= \sum_{i=1}^{n} x_{i1}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_{i1}) \\
&= \sum_{i=1}^{n} x_{i1}y_i - \hat{\beta}_0\sum_{i=1}^{n} x_{i1} - \hat{\beta}_1\sum_{i=1}^{n} x_{i1}^2
\end{aligned}$$

Substituting Equation 1.24 for $\hat{\beta}_0$

$$\begin{aligned}
0 &= \sum_{i=1}^{n} x_{i1}y_i - \left(\bar{y}-\hat{\beta}_1\bar{x}_1\right)\sum_{i=1}^{n} x_{i1} - \hat{\beta}_1\sum_{i=1}^{n} x_{i1}^2 \\
&= \sum_{i=1}^{n} x_{i1}y_i - \sum_{i=1}^{n} x_{i1}\bar{y} + \hat{\beta}_1\sum_{i=1}^{n} x_{i1}\bar{x}_1 - \hat{\beta}_1\sum_{i=1}^{n} x_{i1}^2 \\
&= \sum_{i=1}^{n}(y_i-\bar{y})x_{i1} - \hat{\beta}_1\sum_{i=1}^{n}(x_{i1}-\bar{x}_1)x_{i1} \\
\hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(y_i-\bar{y})x_{i1}}{\sum_{i=1}^{n}(x_{i1}-\bar{x}_1)x_{i1}}.
\end{aligned}$$

Using equations 1.22 and 1.23

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(y_i-\bar{y})(x_{i1}-\bar{x}_1)}{\sum_{i=1}^{n}(x_{i1}-\bar{x}_1)^2} \overset{\text{def}}{=} \frac{Cov(x_1,y)}{Var(x_1)} \tag{1.25}$$

In a multidimensional context, the same process to optimise a wider set of parameters is the same: differentiate the empirical risk of multidimensional least squares regression (1.28) with regards to each parameter and solve $\frac{\partial L_S(f)}{\partial \hat{\boldsymbol{\beta}}} = 0$. Solving Equation 1.29 provides the optimal solution for each parameter in the multivariate case, where $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0, ..., \hat{\beta}_j, ..., \hat{\beta}_M]$ are the $M + 1$ coefficients related to the intercept and M features:

$$\begin{aligned}
L_S(f) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}'\hat{\boldsymbol{\beta}})^2 && (1.26)\\
&= \frac{1}{n}\|\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\|^2 && (1.27)\\
&= \frac{1}{n}\left((\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})'(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})\right) && (1.28)\\
\frac{\partial L_S(f)}{\partial \hat{\boldsymbol{\beta}}} &= \frac{1}{n}\frac{\partial}{\partial \hat{\boldsymbol{\beta}}}\left((\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})'(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})\right) = 0\\
&= \frac{\partial}{\partial \hat{\boldsymbol{\beta}}}\left(\boldsymbol{Y}'\boldsymbol{Y} - 2\hat{\boldsymbol{\beta}}'\boldsymbol{X}'\boldsymbol{Y} + \hat{\boldsymbol{\beta}}'\boldsymbol{X}'\boldsymbol{X}\hat{\boldsymbol{\beta}}\right)\\
0 &= -2\boldsymbol{X}'\boldsymbol{Y} + 2\boldsymbol{X}'\boldsymbol{X}\hat{\boldsymbol{\beta}}\\
\hat{\boldsymbol{\beta}} &= \left(\boldsymbol{X}'\boldsymbol{X}\right)^{-1}\boldsymbol{X}'\boldsymbol{Y}. && (1.29)
\end{aligned}$$

1.2.1 Generalised Linear Models

As the name indicates, generalised linear models (GLMs) are a more generic form of the multivariate linear regression. While making some of the same assumptions, they mostly differ in the process in which the optimised objective is obtained and the variety of tasks that they can accomplish. GLMs assume that y can be split into two components: a systematic component and a random component. The systematic component is the variation of $\boldsymbol{Y}$ explained by the predictor $f(\boldsymbol{x})$, while the random component is the unexplained variance of $\boldsymbol{Y}$ due to missing information in $\boldsymbol{x}$ or random circumstances. As a result, GLMs aim to explain the variance of $\boldsymbol{Y}$ as much as possible.

1.2.1.1 The exponential family of distributions

Least squares linear regression assumes that $y \in \mathbb{R}$. When that is not the case, GLMs can work with the assumption that y is a random variable that follows a distribution from the exponential family of distributions, as expressed by Equation 1.30. The exponential family of distributions includes the Gaussian, Bernoulli, Poisson, Gamma and Tweedie distributions. It is important to note that in Equation 1.30, "*Exponential*" does not refer to a specific distribution and is meant to be substituted by any distribution that is part of the exponential family of distributions.

$$y \sim Exponential(\mu, \phi) \qquad (1.30)$$

An essential parameter of Equation 1.30 is the mean of the distribution $\mu(\boldsymbol{x})$. When each attribute in $\boldsymbol{x}$ has a linear relationship with the mean of the outcomes, $\mathbb{E}[\boldsymbol{Y}|\boldsymbol{x}] = \mu(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\beta}$ and $y = g(\mu(\boldsymbol{x})) + \epsilon$, where $\mu(\boldsymbol{x})$ is the systematic component and ϵ is model error that is due to random noise (i.e. the random component). As $\mu(\boldsymbol{x})$ needs to be linear with regards to $\boldsymbol{x}$, it must be transformed by using a link function. The purpose of a **link function** $g(\cdot)$ is to model the relationship between $\mu(\boldsymbol{x})$ and $\boldsymbol{x}$. When $\mu(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\beta}$, no link function is needed. However, if $\mu(\boldsymbol{x}) = \exp(\boldsymbol{x}'\boldsymbol{\beta})$ as in Equation 1.31, a link function is applied that transforms a model with a linear relationship into a model with an exponential relationship:

$$g(\mu(\boldsymbol{x})) = \log(\exp(\boldsymbol{x}'\boldsymbol{\beta})) = \boldsymbol{x}'\boldsymbol{\beta} \tag{1.31}$$

$$\Leftrightarrow g^{-1}(\boldsymbol{x}'\boldsymbol{\beta}) = \mu(\boldsymbol{x}) = \exp(\boldsymbol{x}'\boldsymbol{\beta}). \tag{1.32}$$

The other important parameter in 1.30 is the parameter ϕ, which is the dispersion parameter of the function. It is related to the variance, but it is not the actual variance. It is the constant component of variance which can be found in the distribution of $\boldsymbol{Y}$. Since a Gaussian distribution assumes that the variance does not vary with the mean $\mu(\boldsymbol{x})$ and thus that the variance is constant, σ^2 would be equal to ϕ. All the other distributions assume that the variance contains a constant component ϕ, but also that the variance depends on the mean of the distribution $\mu(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\beta}$, as seen in Table 1.4. As such, the mean and variance of the assumed distribution varies for each observation $\boldsymbol{x}$. For the Tweedie distribution, this mean-variance relationship can be set by using the *power* parameter ρ, which will be explored in Chapter 3.

1.2.1.2 Maximum likelihood estimation

In order to optimise a GLM, it is crucial to select the proper distribution with regards to the task to solve. As shown in Table 1.4, knowing the set of possible values of the response variable y can help in selecting a link function, but can also help in selecting the distribution used to compute the objective that needs to be optimised. As with linear regression, a loss minimisation function must be built to optimise each parameter with gradient descent. For least squares linear regression, $\hat{\boldsymbol{\beta}}$ can be directly obtained by using Equation 1.29, which is a closed-form expression. However, for GLMs in general, $\hat{\boldsymbol{\beta}}$ cannot be obtained by a closed-form expression. To overcome this, **maximum likelihood estimation** (MLE) can be used to get the estimate the best parameters when knowing the distribution of the model. When optimising a model, MLE tries to find the parameters that were the most likely to have generated $\boldsymbol{Y}$ using a known part of the total distribution. This likelihood is hard and inefficient to estimate and optimise, so extra assumptions must be made about the form of the objective and the original distribution. Firstly, the models using MLE assume that all the data that are used are independently and identically distributed. This lets us transform the likelihood of the distribution $P(\boldsymbol{Y}|\boldsymbol{\theta})$ into the product of independent likelihoods for all independent samples $\prod_{i=i}^{n} P(y_i|\boldsymbol{\theta})$. Secondly, since the derivative of the product of each likelihood in Equation 1.33 can get expensive to compute, the logarithmic transformation of the likelihood equation, or log-likelihood (1.34), can be used to transform Equation 1.33 into separate sums. Since the logarithm function is a monotonic transformation on the likelihood estimation, an increase of the log-likelihood is an increase of the likelihood. Afterwards, the derivative of the log-likelihood with respect to each parameter can be computed and the objective can be optimised by finding the parameters $\boldsymbol{\theta}$ that maximise the likelihood. Equations 1.34 to 1.37 show the process in which the log-likelihood function of the normal distribution is transformed into a simpler form:

$$P(\boldsymbol{Y}|\boldsymbol{\theta}) = \prod_{i=i}^{n} P(y_i|\boldsymbol{\theta}) \tag{1.33}$$

$$\ell(\boldsymbol{Y}|\boldsymbol{\theta}) \overset{\text{def}}{=} \log(P(\boldsymbol{Y}|\boldsymbol{\theta})) \tag{1.34}$$

$$= \sum_{i=i}^{n} \log(P(y_i|\mu(\boldsymbol{x}_i))) \tag{1.35}$$

Assuming that $y_i \sim N(\mu(\boldsymbol{x}_i), \sigma^2)$

$$= \sum_{i=i}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(y_i - \mu(\boldsymbol{x}_i))^2}{2\sigma^2}\right\}\right) \tag{1.36}$$

$$= n \cdot \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + \frac{1}{2\sigma^2}\sum_{i=i}^{n} -(y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2. \tag{1.37}$$

The final objective can now be easily differentiated. Given that a derivative with respect to any β_j removes any part not containing it, the objective now becomes the maximisation of $\frac{1}{2\sigma^2}\sum_{i=i}^{n} -(y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2$ with regards to $\boldsymbol{\beta}$. Also, since that the maximisation of an objective is equal to minimising its negation, the empirical risk function obtained from the differentiation of Equation 1.37 would be equivalent to the original empirical risk function of Equation 1.21, where the loss function $L = -\log(P(y|\mu(\boldsymbol{x})))$. Hence, this method is conceptually different, but estimates the same parameters as taking the empirical risk function that needs to be minimised. This process can be reproduced in the same manner for the Poisson, Gamma, Tweedie or Bernoulli distribution.

1.2.1.3 Data processing

After the log-likelihood related to the distribution of $\boldsymbol{Y}$ has been differentiated in a way that can be used to solve an objective, it is important to note that a dataset must be adapted before being used in a model's optimisation process for many reasons.

Firstly, the estimated coefficients represent the linear relation between j^{th} feature and a function of the mean of the response variable. This bodes well if the feature is continuous. On the other hand, categorical variables are not well supported out of the box, since the different

Distribution	**Set of Possible Outcomes**	**Most Frequently Used Link Function**	**Variance**
Normal	$\mathbb{R}$	$\mu(\boldsymbol{x}) = \boldsymbol{x}'\beta$	ϕ
Poisson	$\mathbb{Z}^+$	$\log(\mu(\boldsymbol{x})) = \boldsymbol{x}'\beta$	$\phi\mu(\boldsymbol{x})$
Gamma	$\mathbb{R}^+_*$	$\log(\mu(\boldsymbol{x})) = \boldsymbol{x}'\beta$	$\phi\mu(\boldsymbol{x})^2$
Tweedie	$\mathbb{R}^+$	$\log(\mu(\boldsymbol{x})) = \boldsymbol{x}'\beta$	$\phi\mu(\boldsymbol{x})^\rho$
Bernoulli	$\{0, 1\}$	$\log(\frac{\mu(\boldsymbol{x})}{1-\mu(\boldsymbol{x})}) = \boldsymbol{x}'\beta$	$\phi\mu(\boldsymbol{x})(1-\mu(\boldsymbol{x}))$

TABLE 1.4 – GLM distributions by set of possible outcomes and their characteristics

attribute values do not have any relation to each other. To overcome this, these features must be encoded in a way that is explainable and that allows a linear relationship to be formed between categorical values and a function of the mean of the response variable. This is done by encoding the categorical feature values into **dummy variables**. The encoding is done in two parts: the selection of a base level value and the transformation of the non-base level values into a vector of binary indicators. The **base level**, which is usually the most frequent value of the attribute, is the value of a categorical attribute which has its effect kept implicitly in the intercept of the model. Hence, it is not attributed a coefficient. On the other hand, all non-base level values are attributed a coefficient which quantify the marginal effect of having that value instead of the base value. When converting a non-base level categorical value, the coefficient attributed to this categorical value would then be multiplied by 1 and the other values would be multiplied by 0. A person's marital status can be used as an example of the creation of dummy variables, as it is a qualitative attribute that does not have an order of magnitude. Consider that the possible values of the attribute are "single", "married" or "divorced" and that "single" is the base level value. In this case, "single" is encoded as $[0, 0]$, "divorced" is encoded as $[1, 0]$ and "divorced" is encoded as $[0, 1]$.

Secondly, GLMs work best when all attributes $(x_1, ..., x_M)$ are not excessively correlated with all the other attributes. If some attributes are excessively correlated, a problem known as multicollinearity may occur. **Multicollinearity** happens when an attribute can be accurately predicted by a linear combination of the other variables. If an information can be found twice in a dataset, it is difficult to determine how much each representation of that information contributes to y, which adds much uncertainty to the estimates of $\hat{\boldsymbol{\beta}}$. As a result, it causes many problems such as inconsistent estimations of $\hat{\boldsymbol{\beta}}$, invalid prediction analysis, overfitting and high standard errors. If multicollinearity is detected, it can be eliminated by removing all but one of the correlated features from $\boldsymbol{X}$. It can be assumed that minimal information will be lost, as the information is redundant in the other attributes.

Algorithm 1.2 *Stochastic gradient descent*

```
1: procedure SGD($S, \lambda$)
2:     Initialise $\hat{\boldsymbol{\beta}}^{(0)} = \mathbf{0}$
3:     for $t = 1$ to $k$ do
4:         sample $\boldsymbol{x}_t, y_t$ from $\boldsymbol{S}$
5:         $\hat{\boldsymbol{\beta}}^{(t+1)} = \hat{\boldsymbol{\beta}}^{(t)} - \lambda \cdot \frac{\partial}{\partial \hat{\boldsymbol{\beta}}} L(f(\boldsymbol{x}_t|\hat{\boldsymbol{\beta}}^{(t)}), y_t)$
6:     end for
7:     return $\frac{1}{k}\sum_{t=1}^{k} \hat{\boldsymbol{\beta}}^{(t)}$
8: end procedure
```

1.2.1.4 Model optimisation

Since GLMs that assume a distribution of outcomes other than a normal distribution cannot use a closed-form expression to estimate $\hat{\boldsymbol{\beta}}$, **stochastic gradient descent** (SGD) can be used instead to obtain $\hat{\boldsymbol{\beta}}$. SGD (1.2) is an iterative process in which all parameters are updated using a single sampled observation's gradient at a time, instead of the whole dataset. The gradient is obtained by differentiating the loss function L, which is the log-likelihood of the selected distribution for GLMs. In the end, SGD optimises the parameters for k iterations, which is predefined as a hyperparameter. The rate at which the algorithm updates the parameters is set by the **learning rate** λ.

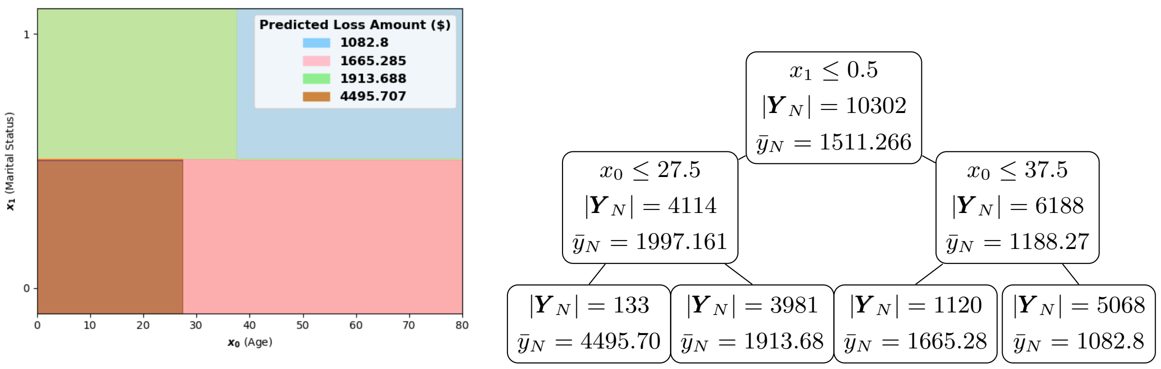
1.3 Tree Boosting

1.3.1 Classification and regression trees (CART)

A decision tree represents a partitioning of a $\mathbb{R}^n$ feature space into subspaces by defining the best boundaries that set apart different values of $\boldsymbol{x}$ with respect to y. Decision trees are composed of three types of components: internal nodes, leaf nodes and branches. An internal node is a representation of a decision rule, which is a boolean condition evaluated on a single attribute. A decision rule could be, for example, determining if a person is over or under the age of 40. Depending on the response to that question, the appropriate child branch is taken to the next node. A leaf node is found at the bottom of the tree and contains the final predicted value. Altogether, a decision tree is made with a single root node at the top, many levels of internal nodes, leaves at the bottom and branches that connect them all together.

To obtain a prediction from a decision tree, an observation is passed through the root node at the top of the tree. Then, it is guided by decision rules through the proper path of internal nodes and branches until a leaf node is reached. Hence, a prediction is obtained from a decision tree in a way that is similar to relating the conjunction of boolean conditions on attributes contained in $\boldsymbol{x}$ to its corresponding value.

The process in which a decision tree is created with training set $(\boldsymbol{X}, \boldsymbol{Y})$ is done in five steps.


Predicted Loss Amount ($)
1082.8
1665.285
1913.688
4495.707
x_1 (Marital Status)
x_0 (Age)
$x_1 \leq 0.5$, $|\boldsymbol{Y}_N| = 10302$, $\bar{y}_N = 1511.266$
$x_0 \leq 27.5$, $|\boldsymbol{Y}_N| = 4114$, $\bar{y}_N = 1997.161$
$x_0 \leq 37.5$, $|\boldsymbol{Y}_N| = 6188$, $\bar{y}_N = 1188.27$
$|\boldsymbol{Y}_N| = 133$, $\bar{y}_N = 4495.70$
$|\boldsymbol{Y}_N| = 3981$, $\bar{y}_N = 1913.68$
$|\boldsymbol{Y}_N| = 1120$, $\bar{y}_N = 1665.28$
$|\boldsymbol{Y}_N| = 5068$, $\bar{y}_N = 1082.8$

(A) Decision boundaries (B) Tree representation of decision rules

FIGURE 1.6 – Comparison of a decision tree and its resulting partitions

First, the node is assigned a subset data $(\boldsymbol{X}_N, \boldsymbol{Y}_N)$ provided from its parent node. As such, the root node is attributed the entire training set. It is also assigned the current depth $i = 0$, where the depth is the number of branches required to reach the node from the root. Second, for each attribute in $\boldsymbol{X}_N$, all possible decision rules are generated. Afterwards, for each x_j, all of the possible decision rules $x_j \leq \eta$ are applied to $(\boldsymbol{X}_N, \boldsymbol{Y}_N)$, where η can be any possible value of x_j. This creates a split sets of outcomes that would be attributed to left and right child nodes $\boldsymbol{Y}_l$ and $\boldsymbol{Y}_r$, depending on if the observation related to the outcome had x_j over or under the threshold η. These split sets are kept in a set Q. The split function, such as Equation 1.39, is then computed on all the split sets found in Q. The condition which minimises the most the empirical risk, is then selected and is assigned to the node. In return, the best gain of information about $\boldsymbol{Y}_N$ by using any attribute in $\boldsymbol{X}_N$ is obtained, as seen in Equation (1.40). Third, the sets $\boldsymbol{Y}_l$ and $\boldsymbol{Y}_r$ resulting from the objective 1.39 and their respective observations $\boldsymbol{X}_l$ and $\boldsymbol{X}_r$ are kept aside for the child nodes. Fourth, the current depth i is incremented by 1. Fifth, if the maximum depth D is not reached, a left child node N_l and right child node N_r are created and are assigned to the current node. The first three steps are then performed on the child nodes with their respective datasets $(\boldsymbol{X}_l, \boldsymbol{Y}_l)$ and $(\boldsymbol{X}_r, \boldsymbol{Y}_r)$. Otherwise, the parent node is assigned two leaves instead of nodes. The leaves are assigned values which are equal to the mean value of their respective outcomes $\boldsymbol{Y}_l$ and $\boldsymbol{Y}_r$, which are respectively $\bar{y}_l$ and $\bar{y}_r$.

As seen in the last paragraph, many hyperparameters affect the creation of the decision tree. Notably, the maximum depth D can be set to limit the number of decision levels inside of a tree. In addition, the splitting function can be adapted to other empirical risk functions than the mean square error to better minimise the target objective. The leaf value can also be computed using a different function. For example, the median can be used instead of the mean.

$$\bar{y}_N = \frac{1}{|\boldsymbol{Y}_N|} \sum_{y \in \boldsymbol{Y}_N} y \tag{1.38}$$

$$\boldsymbol{Y}_l, \boldsymbol{Y}_r = \underset{(\boldsymbol{Y}_l, \boldsymbol{Y}_r) \in Q}{\operatorname{argmin}} \left(\sum_{y \in \boldsymbol{Y}_l} (\bar{y}_l - y)^2 + \sum_{y \in \boldsymbol{Y}_r} (\bar{y}_r - y)^2 \right) \tag{1.39}$$

$$\boldsymbol{Y}_l, \boldsymbol{Y}_r = \underset{(\boldsymbol{Y}_l, \boldsymbol{Y}_r) \in Q}{\operatorname{argmax}} \left(\frac{\sum_{y \in \boldsymbol{Y}_N} (\bar{y}_N - y)^2}{|\boldsymbol{Y}_N|} - \frac{\sum_{y \in \boldsymbol{Y}_l} (\bar{y}_l - y)^2}{|\boldsymbol{Y}_l|} - \frac{\sum_{y \in \boldsymbol{Y}_r} (\bar{y}_r - y)^2}{|\boldsymbol{Y}_r|} \right) \tag{1.40}$$

An example of a decision tree and its resulting partitions can be seen in Figure 1.6. In that case, the task is to try to predict how much someone will claim losses to his/her insurer based on his/her age x_0 and his/her marital status x_1. The person is single if $x_1 = 0$, while the person is married if $x_1 = 1$. By associating Figure 1.6 a) to Figure 1.6 b), it can be seen that each internal node determines the boundary a partition, while the leaves are the values attributed to each partition. As taking either branch of an internal node leads to different leaves, the decision rule splits the feature space. Figure 1.6 also shows how decision trees can support either categorical or continuous values out of the box. In this case, the age is continuous, while the marital status is categorical. The distribution of the data is clearly not the same depending on the type of the variable, but the model manages that pretty well. Like GLMs, it is easy to explain why the model made a certain prediction, since the decision rules are explicit. However, decision trees do not have to assume that the effect of each feature is independent, as decision trees are built to make predictions based on a conjunction of rules. As such, complex decision trees can make the interaction between two variables stand out. Another quality that makes decision trees stand out from GLMs is that they do not assume that features have a continuous or monotonic relationship with regards to the objective. For instance, Figure 1.6 a) shows that the relationship between $\boldsymbol{x}$ and y depends on the effects of x_0 and x_1. Notably, the age of a single client decreases y at 27.5 years, while y decreases at 37.5 years if the client is married.

1.3.2 Boosting

Decision trees on their own produce explainable classifiers, but have low performance on new data. That is caused by many factors, including overfitting and high variance in the distribution of predicted outcomes. For example, training a decision tree with a high volume of complex data often leads to deep and large trees that do not contain many samples in each leaf. This creates terminal nodes (i.e. predictions) which are based on a small number of samples and which highly vary with small changes in data. Also, decision trees are unstable as they can change substantially with any slight difference in training data.

In order to obtain better generalisation, a set of decision trees can be combined by using ensembling methods such as boosting. Boosting starts with the idea that combining many weak learners into an ensemble of weak learners can create a strong ensemble model. To create this ensemble model, a weak learner is fit on the training data $\boldsymbol{S}$. Then, the errors of that weak learner are computed and alter the importance of each observation in $\boldsymbol{S}$, which are then fit on another weak learner. The worse an example was classified, the more weight it has in future models. Sequentially, a total of k weak learners are added to a committee of weak learners that continuously tries to label previously mislabeled observations accurately in what are called boosting rounds. This creates a ensemble model in which all types of observations have a higher chance of being labeled accurately.

1.3.3 Adaboost

AdaBoost, an additive boosting algorithm, is one of the most commonly known and simple form of boosting. To explain it properly, we will take into consideration its simplest use case. AdaBoost uses tree stumps as weak learners, which are decision trees with a maximum depth of 1, thus having two leaves. When used on a binary classification problem where $y \in \{\pm 1\}$, tree stumps will always try to find the best split that separates the positive from the negative class. In a vacuum, these stumps minimise a zero-one loss (1.4). However, as explained in Section 1.3.2, these weak learners do not act alone.

Algorithm 1.3 *AdaBoost*

```
1. Initialise $F = 0$.
2. Initialize $D_i^{(1)} = \frac{1}{n}$ for all $i = 1, ..., n$, where $n = |\boldsymbol{S}|$
3. For $t = 1$ to $k$:
    a) Train weak learner $h_t(\boldsymbol{x}|\boldsymbol{D}^{(t)})$ with observations $\boldsymbol{S}$, minimising the zero-one loss
    b) Compute error $\epsilon_t = \sum_{i=1}^{n} D_i^{(t)} I(h_t(\boldsymbol{x}_i) \neq y_i)$, where $I(a) = \begin{cases} 1 & \text{if } a = \text{True} \\ 0 & \text{if } a = \text{False} \end{cases}$
    c) Calculate classifier weight $\alpha_t = \frac{1}{2} \log \left( \frac{1}{\epsilon_t} - 1 \right)$
    d) $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{\sum_{j=1}^{n} D_j^{(t)} \exp(-\alpha_t y_j h_t(\boldsymbol{x}_j))}$ for all $i = 1, ..., n$
4. Return $F(\boldsymbol{x}) = sign\left( \sum_{t=1}^{k} \alpha_t h_t(\boldsymbol{x}) \right)$
```

In Algorithm 1.3, the boosting process starts by defining a distribution $\boldsymbol{D}^{(t)} \in \mathbb{R}_+^n$ over the training samples, with each sample starting with an equal probability mass $D_i^{(t)}$ of $\frac{1}{n}$ so that $\sum_{i=1}^{n} D_i^{(t)} = 1$. The first weak learner is fit with the training observations $\boldsymbol{S}$ and the distribution $\boldsymbol{D}$. As it makes mistakes, each $D_i^{(t)}$ is readjusted higher if $\boldsymbol{x}_i$ was wrongly classified, or lower if $\boldsymbol{x}_i$ was classified correctly. The next stump will then learn from these mistakes and will try to focus on the observations that were wrongly classified. However, since each stump will constantly need to work on observations that couldn't be classified properly by the previous weak learners, it will naturally increase the weak learner's error ϵ_t, as the observations that can't be properly classified will constantly gain weight in the total error. That is why AdaBoost builds functions that make predictions based on the majority vote of an ensemble of weak learners and the weighting α_t of each vote based on the weak learner's error at training time (1.41). The resulting ensemble of k weak learners is designated as $F(\boldsymbol{x})$:

$$F(\boldsymbol{x}) = sign\left(\sum_{t=1}^{k} \alpha_t h_t(\boldsymbol{x})\right). \tag{1.41}$$

As such, the principle of boosting is done through each iteration, where misclassified samples have their importance increased, while the well classified samples have their importance decreased. Hence, after enough iterations, a classifier able to perform predictions on simple and hard observations is provided.

1.3.4 Gradient Boosting

Gradient boosting is another method which uses weak learners, such as CART, and iteratively increases the importance of misclassified samples. Nevertheless, it is quite different from AdaBoost. For instance, it does not rely directly on the weight of each observation to optimise the next weak learner. Also, each weak learner is not attributed a weight with regards to its error rate. In short, the gradient boosting process improves the accuracy of the model on observations that aren't classified properly by setting the gradient of the errors of the previous weak learners as the target variable of the next weak learner.

Instead of being a stepwise algorithm where each h_t is independent, algorithm 1.4 is a stagewise algorithm, where $\hat{y}^{(t)}$ is in fact the result of the sum of the predictions of t previous base learners multiplied by the constant learning rate λ. At each stage of the algorithm, the gradient of the errors of the previous weak learners become the target variable of the next weak learner, while the predictions of the learner h_t are added to the total predictions $\hat{y}^{(t)}$ at a learning rate λ. The result is a function $F(\boldsymbol{x})$, which is the sum of all k base learners multiplied by the learning rate λ (1.42):

$$F(\boldsymbol{x}) = \sum_{t=1}^{k} \lambda \cdot h_t(\boldsymbol{x}). \tag{1.42}$$

All in all, this type of boosting is similar to gradient descent, in which $\hat{\boldsymbol{\theta}}$ is replaced with $\hat{\boldsymbol{Y}}$. By trying to predict how $\hat{y}$ must change to be equal to y at each iteration, the algorithm effectively performs gradient descent on the predicted outcomes to find their optimal value (i.e. y).

The biggest advantage that gradient boosting has over other boosting methods is that any differentiable and convex function can easily be used as a loss function. It can naturally use squared error and absolute error as objectives, but can also use any function that is obtained by maximum likelihood estimation. Therefore, any log-likelihood function that can be used on GLMs can also be used for Gradient Boosted Decision Trees (GBDT). Hastie et

Algorithm 1.4 *Gradient Boosting*

```
1. Initialise $F = 0$
2. Set $\hat{y}_i^{(0)}$ as zeroes, random values or the result of an initial predictor for all $i = 1, ..., n$
3. For $t = 1$ to $k$:
    a) $r_i^{(t)} = -\frac{\partial}{\partial \hat{y}_i^{(t-1)}} L(\hat{y}_i^{(t-1)}, y_i)$ for all $i = 1, ..., n$
    b) Fit a regression tree $h_t(\boldsymbol{x})$ with $\boldsymbol{X}$ as observations and residuals $\boldsymbol{r}^{(t)}$ as targets
    c) $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \lambda \cdot h_t(\boldsymbol{x}_i)$ for all $i = 1, ..., n$, where $\hat{y}_i^{(t-1)} = \sum_{j=1}^{t-1} \lambda \cdot h_j(\boldsymbol{x}_i)$
4. Return $F$, where $F(\boldsymbol{x}) = \sum_{t=1}^{k} \lambda \cdot h_t(\boldsymbol{x})$
```

al. [28] demonstrated that AdaBoost is equivalent to using GBDTs with an exponential loss $L(F(\boldsymbol{x}), y) = \exp(-y \cdot F(\boldsymbol{x}))$, but that a binary classifier could also be made with the assumption that outcomes follow a Bernoulli distribution, as there are only two possible outcomes:

$$-\ell(\boldsymbol{Y}|\theta) = \sum_{i=1}^{n} \log(P(y_i|F(\boldsymbol{x}_i)) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot F(\boldsymbol{x}_i)\right)\right), y_i \in \{-1, 1\} \tag{1.43}$$

$$-\frac{\partial}{\partial F(\boldsymbol{x}_i)} \log(P(y_i|F(\boldsymbol{x}_i))) = -y_i \cdot \frac{\exp\left(-y_i \cdot F(\boldsymbol{x}_i)\right)}{1 + \exp\left(-y_i \cdot F(\boldsymbol{x}_i)\right)} = \frac{-y_i}{1 + \exp\left(y_i \cdot F(\boldsymbol{x}_i)\right)}. \tag{1.44}$$

Equation 1.44 shows the flexibility in which GBDTs can use the distribution of outcomes as prior as an advantage in weighting the right elements to get the best possible distribution of predictions. GBDTs can be flexible to the point where custom loss functions can be used arbitrarily to fit a client's specific needs. The importance of outliers, the consideration of negative outcomes and the type of problem all play a role in the selection of the loss function, which not all algorithms can easily accomodate.

1.3.5 Advanced Gradient Boosting

In an era where managing huge amounts of data is necessary, using GBDTs with CART is often inefficient and does not generate the best results. The variety of data and the number of possible features are sometimes so high that finding the best split for each tree node is impossible on a large scale. Also, the number of hyperparameters can be so high that finding the best configuration becomes a time consuming task. Libraries such as XGBoost [7] and LightGBM [35] add multiple improvements to GBDTs that make them more accurate and efficient.

1.3.5.1 Optimisation method

Traditional GBDTs only use the gradient as an objective of the global stagewise algorithm. However, recent advancements in GBDTs have shown that using the second derivative of the loss function in addition to the gradient can obtain more accurate results. Advanced gradient boosting uses Taylor's second-order function approximation (1.45) on the gradient of the loss function. As adding additional terms to a Taylor Polynomial gives a better estimate of the curvature of a function around a certain point, it can be used to approximate the loss function more accurately. Hence, by estimating the first and second derivatives of the loss function at point $\hat{y}^{(t-1)}$, advanced gradient boosting methods obtain a more accurate estimation of how $\hat{y}$ should move between iterations $t-1$ and t to minimise the loss. This leads to models with higher accuracy and faster convergence.

$$L(y, \hat{y}^{(t)}) \approx L(y, \hat{y}^{(t-1)}) + (\hat{y}^{(t)} - \hat{y}^{(t-1)})L'(y, \hat{y}^{(t-1)}) + \frac{1}{2}(\hat{y}^{(t)} - \hat{y}^{(t-1)})^2 L''(y, \hat{y}^{(t-1)}) \quad (1.45)$$

In addition to the use of second derivatives, extreme gradient boosting enables the use of regularisation, which is implemented directly in its empirical risk and its objective. **Regularisation** is the process of constraining the complexity of models, thus preventing overfitting and increasing generalisation. This can be done, like in the case of XGBoost, by adding a regularisation term Ω to the empirical risk that includes the norm of the leaf values $||\boldsymbol{w}||$ and the number of leaves T. Hence, regularisation permits both the minimisation of the initial objective, while also ensuring that the model does not become too complex.

To understand how XGBoost works, Table 1.5 shows all the prior notation required to understand further exploration done in equations 1.46 to 1.50. As shown in these equations, the resulting empirical risk at iteration t is based on the value of each leaf and the number of leaves.

$$L_S(h_t) = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + h_t(\boldsymbol{x}_i)) + \Omega(\boldsymbol{w}^{(t)}) \quad (1.46)$$

Using Equation 1.45 with $w_i^{(t)} = (\hat{y}_i^{(t)} - \hat{y}_i^{(t-1)}) = h_t(\boldsymbol{x}_i)$, $g_i^{(t)} = L'(y_i, \hat{y}_i^{(t-1)})$ and $h_i^{(t)} = L''(y_i, \hat{y}_i^{(t-1)})$

$$L_S(h_t) = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)}) + g_i^{(t)} w_i^{(t)} + \frac{1}{2} h_i^{(t)} (w_i^{(t)})^2 + \Omega(\boldsymbol{w}^{(t)}) \quad (1.47)$$

Removing $L(y_i, \hat{y}_i^{(t-1)})$, since it is constant

$$L_S(h_t) \simeq \sum_{i=1}^{n} \left(g_i^{(t)} w_i^{(t)} + \frac{1}{2} h_i^{(t)} (w_i^{(t)})^2 \right) + \gamma T + \frac{1}{2} \sum_{m=1}^{T} \eta (w_m^{(t)})^2 \quad (1.48)$$

Symbol	**Meaning**	**Equation**
I_m	Set of observations contained in node m	
$I_m^{(t)}$	Set of observations contained in node m at iteration t	
g_i	Gradient for observation i	$\frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i}$
h_i	Second derivative for observation i	$\frac{\partial^2 L(y_i,\hat{y}_i)}{\partial \hat{y}_i^2}$
G_L	Sum of gradients in left side of split	$\sum_{g \in I_L} g$
G_R	Sum of gradients in right side of split	$\sum_{g \in I_R} g$
G_N	Sum of gradients in the parent node	$\sum_{g \in I_N} g$
$G_m^{(t)}$	Sum of gradients in node m at iteration t	$\sum_{g^{(t)} \in I_m^{(t)}} g^{(t)}$
H_L	Sum of second derivatives in left side of split	$\sum_{h \in I_L} h$
H_R	Sum of second derivatives in right side of split	$\sum_{h \in I_R} h$
H_N	Sum of second derivatives in the parent node	$\sum_{h \in I_N} h$
$H_m^{(t)}$	Sum of second derivatives in node m at iteration t	$\sum_{h^{(t)} \in I_m^{(t)}} h^{(t)}$
$w_i^{(t)}$	Value of leaf for example $\boldsymbol{x}_i$ at iteration t	$(\hat{y}_i^{(t)} - \hat{y}_i^{(t-1)})$
$w_m^{(t)}$	Value of leaf m at iteration t	$-\frac{\sum_{i \in I_m} g_i^{(t)}}{\sum_{i \in I_m^{(t)}} h_i + \lambda}$
$\boldsymbol{w}^{(t)}$	Vector containing all leaf values for tree at iteration t	
$\Omega(\boldsymbol{w}^{(t)})$	Regularisation function for iteration t	$\gamma T + \frac{1}{2}\lambda \lvert\lvert \boldsymbol{w}^{(t)} \rvert\rvert^2$
η	Constant leaf value regularisation factor	
γ	Constant leaf count regularisation factor	
T	Number of leaves in tree	

Table 1.5 – Definition of symbols used in the XGBoost algorithm

Grouping the loss per leaf m

$$L_S(h_t) = \sum_{m=1}^{T} \left(G_m^{(t)} + \frac{1}{2}(H_m^{(t)} + \eta)(w_m^{(t)})^2 \right) + \gamma T \tag{1.49}$$

Replacing $w_m^{(t)}$ with its definition from table 1.5

$$L_S(h_t) = -\frac{1}{2} \sum_{m=1}^{T} \frac{(G_m^{(t)})^2}{H_m^{(t)} + \eta} + \gamma T \tag{1.50}$$

Thus, the loss at iteration t can already be estimated from the loss at iteration $t-1$ and a next boosting iteration's predicted value $\hat{y}_i^{(t)}$. Nonetheless, the loss cannot be minimised directly as the loss is estimated on the entire tree structure. Since we cannot generate all possible tree structures to obtain the best one, the split function of the decision tree must be altered to adapt to this loss function.

1.3.5.2 Split function

Even if Equation 1.50 has interesting properties in estimating the empirical risk for the next iteration, it cannot be directly minimised in an objective. As Equation 1.50 computes the loss for the entire tree structure, it cannot be used to build a tree that minimises the loss. However, the equation can be transformed to compute the loss on a single split at a time, which can minimise the loss on the entire tree. Hence, as in Section 1.3.1, XGBoost computes the loss (1.51) for every split in set Q to determine the best decision function to apply to the current node. This contrasts with normal decision tree gradient boosting, which mostly uses the squared error (1.39) as an objective for splits. Instead, as the split function of each tree is altered to consider the current gradients and second derivatives, the splits which are selected in XGBoost trees are specific to the selected loss function.

$$\boldsymbol{Y}_l, \boldsymbol{Y}_r = \underset{(\boldsymbol{Y}_l, \boldsymbol{Y}_r) \in Q}{\operatorname{argmin}} \left(\frac{1}{2} \left(\frac{G_N^2}{H_N + \eta} - \frac{G_L^2}{H_L + \eta} - \frac{G_R^2}{H_R + \eta} \right) \right) \tag{1.51}$$

Equation 1.51 shows that XGBoost decision trees can be compared to CART decision trees, as their split functions both share a similar form. In fact, considering that the squared error has a constant second derivative of 2, it results in the same split function without regularisation.

1.3.5.3 Missing features handling

One shortcoming of CART is that it does not handle training sets $\boldsymbol{X}$ with missing values off the shelf. In order for decision trees to handle missing values, the data is normally preprocessed by value substitution, truncation or prediction. However, this adds unnecessary bias when training the model, because it may distort the data's true nature. Extreme gradient boosting improves on CART by selecting a default direction on each branch to handle missing values. This is done by calculating the optimal split on non-missing values twice : once with the default direction being on the right and once on the left. What this means is that all thresholds are tested, but the loss is also based on all instances with missing values either to the left or right child node. In the end, no information is lost, as all observations are still distributed to their respective children nodes for further splitting. At prediction time, the default direction can then be reused on new elements.

1.3.6 Model Interpretation

Model interpretability and transparency is a core concern when implementing machine learning models when it is crucial to understand why a prediction was made, instead of only having a precise prediction. Lending or job applications all need proper justifications when making a decision to ensure that the decision is based on valid factors. Healthcare experts cannot give a cancer diagnosis using a model that does not indicate which factors were relied on to determine the outcome. Models also often need to be debugged if the model doesn't generalise well or if experts don't agree with the predicted outcomes. Hence, models which are too complex to be explained are considered black-boxes, which limits their use for many tasks. On the other hand, some models can be used as their individual predictions, along with the global importance of each feature, are explained properly.

1.3.6.1 Linear Model Interpretation

As noted in the simple least squares regression equation (1.20), each parameter $\hat{\beta}_j$ represents the average increase of $\hat{y}$ when increasing the j^{th} attribute of $\boldsymbol{x}$ by a single unit. This makes for an easy explanation of each prediction. For example, if we try to predict the weight in kilograms y of a person $\boldsymbol{x}$ using its height in centimetres x_1 and age x_2, the corresponding model would be equivalent to Equation 1.52. In return, if $\hat{\boldsymbol{\beta}} = [5, 0.25, 1.25]$ and $\boldsymbol{x} = [1, 180, 20]$, then 1.52 would predict that the person's weight is 75 kilograms, with regards to the existing population.

$$\hat{y} = f(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{Y}|\boldsymbol{x}] = \boldsymbol{x}'\hat{\boldsymbol{\beta}} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{1.52}$$

The main advantage of this equation is that it is explicit when making a prediction. Each element of the sum shows the separate contribution of every attribute with regards to the final result. Also, each element of the sum represents the outstanding values that differentiate $\boldsymbol{x}$ from the average population.

A secondary property that contributes to model interpretability is the possibility to provide **confidence intervals** (1.53) on a model's parameters and on a model's predictions. One key property of linear regression is that it assumes that its errors are normally distributed. If that is the case, then it is possible to determine the range of possible results with regards to the existing residuals. For linear regression, the common way to do this is to compute confidence intervals that use a t-distribution, along with the variance of the prediction. The t-distribution has single parameter: $n - p$ degrees of freedom, where $n = |\boldsymbol{X}|$ is the number of rows in $\boldsymbol{X}$ and $p = |\hat{\boldsymbol{\beta}}|$. The variance of the prediction is estimated using the Gauss-Markov theorem (1.55), as stated in Hayashi et al. [29]. Then, by knowing the mean and the standard error of a prediction, a confidence interval provides the minimum and maximum value of the outcome at a confidence level α.

$$CI_\alpha(f(\boldsymbol{x})) = \boldsymbol{x}'\hat{\boldsymbol{\beta}} \pm t_{\alpha/2;n-p}\sqrt{Var(\boldsymbol{x}'\hat{\boldsymbol{\beta}})} \tag{1.53}$$

$$Var(\boldsymbol{x}'\hat{\boldsymbol{\beta}}) = \boldsymbol{x}'Var(\hat{\boldsymbol{\beta}})\boldsymbol{x} = s^2(1 + \boldsymbol{x}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}) \tag{1.54}$$

$$Var(\hat{\boldsymbol{\beta}}) = s^2(\boldsymbol{X}'\boldsymbol{X})^{-1} \tag{1.55}$$

$$s^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-p} \tag{1.56}$$

As an example, Equation 1.53 can be used on the previous weight prediction model used at the start of this section. Supposing a sample size $n = 100$, the number of coefficients $p = 3$ and a confidence level $\alpha = 1 - 0.95 = 0.05$, the t-value would be equal to $t_{0.05/2;100-3} \approx 1.98$. If the variance of the prediction $Var(\boldsymbol{x}'\hat{\boldsymbol{\beta}})$ is equal to 2 kilograms^2 and the predicted weight is 75 kilograms, then the confidence interval would consider that the real outcome would be between 71.04 and 78.96 kilograms at a confidence level of 95%. This shows that a model does not only produce outcomes, but that it also can determine a range of results at a certain confidence level.

1.3.6.2 Decision Tree Interpretation

The state-of-the-art precision of the GBDT makes it a solution of choice for many tasks using tabular data, such as fraud detection [68]. However, when implementing a GBDT in a production setting, it must be possible for flagged individuals to know why a certain decision was taken.

As discussed previously, GLMs can easily be interpreted, as each independent feature gets attributed a single coefficient. This additive contribution of features makes the explanation of a prediction a simple task. The comparison can even be done between two predictions on a feature by feature basis, by isolating the feature and its coefficient on each prediction. For a single CART decision trees, the decision rules taken are in themselves easy to interpret. However, as the prediction is the result of a conjunction of rules on multiple features, it is difficult to determine outright the contribution of each feature to the final output. This is exacerbated in GBDTs, as they are the result of an ensemble of trees.

Traditionally, two intuitive metrics have been used to determine global feature importance in CART and GBDTs: the sum of gains and the number of splits per feature. By knowing how much a feature was discriminant on a training set, or how many times an attribute was used in the decision rules, one may get an idea of which feature probably had more incidence on a prediction. However, these metrics do not give the exact contribution of each feature to an individual prediction. When a prediction is calculated, not all splits in a decision tree are used. Thus, each feature does not have the same impact on the result as if all splits were all used. Also, the gain and the number of splits vary greatly in importance depending on the level in which they are used. Notably, a feature which is mostly used in the top level of a tree will naturally have a lower number of splits than features that are used in lower levels.

Using notions from game theory, Lundberg et al. [41] introduced the use of SHAP (SHapley Additive exPlanation) values for model interpretation. In the context of model interpretation, Shapley values are the expected marginal contributions of each feature. In other words, they represent how much a prediction differentiated itself from the mean prediction when adding new information about $\boldsymbol{x}$ to the model.

Equation 1.57 shows how f is transformed in an additive form in which each $\phi_j(f, \boldsymbol{x})$ represents the marginal feature contribution of each feature j on the outcome provided by the model f with the observation $\boldsymbol{x}$ and $\phi_0(f) = \bar{y}$, where $\bar{y}$ is the average of the outcomes $\boldsymbol{Y}$ found in the training set $\boldsymbol{S}$. Thus, the prediction is decomposed as the sum of the effect of each feature added to the training set's expected outcome:

$$f(\boldsymbol{x}) = \phi_0(f) + \sum_{j=1}^{M} \phi_j(f, \boldsymbol{x}). \tag{1.57}$$

In order to obtain each ϕ_j, we must study how a prediction changes on average when a feature x_j is removed from the model. To simulate the removal of x_j, the model must consider that x_j is missing and that the model must do its prediction without x_j. The process in which this must be done varies depending on the type of model used. In a GLM, for example, this would mean that the coefficient β_j would be removed from the model. For other models, where two attributes are not necessarily independent, the effect of removing an attribute can vary depending on which remaining attributes are used in the altered model. As such, the function must be altered to estimate the average effect of adding x_j with any set of prior features, as shown in Equation 1.58. As there are M features, there are 2^M possible combinations of features that must be evaluated with and without x_j.

Consider $f_{\mathcal{G}}(\boldsymbol{x})$, an altered function which only considers the features in $\boldsymbol{x}$ whose indexes are in $\mathcal{G}$. By evaluating $f_{\mathcal{G}}(\boldsymbol{x})$ for each subset of feature indexes $\mathcal{G}$ that is in the set of all possible subsets of feature indexes $\mathcal{F}$, the expected contribution of each feature j can then be obtained.

$$\phi_j(f, \boldsymbol{x}) = \frac{1}{2^{M-1}} \sum_{\mathcal{G} \in \mathcal{F}} \left[f_{\mathcal{G} \cup j}(\boldsymbol{x}) - f_{\mathcal{G}}(\boldsymbol{x}) \right] \tag{1.58}$$

For decision trees, $f(\boldsymbol{x})$ is transformed into $f_{\mathcal{G}}(\boldsymbol{x})$ by using the TreeSHAP algorithm. The TreeSHAP algorithm computes $f_{\mathcal{G}}(\boldsymbol{x})$ by ignoring tree nodes if they involve a feature with an index not contained in $\mathcal{G}$. It starts exploring the decision tree at its root node and applies the decision rules on $\boldsymbol{x}$, as usual. If an encountered tree node uses a feature in $\mathcal{G}$, then the original decision path is used. However, if the feature is not in $\mathcal{G}$, we must obtain the average prediction if feature x_j were not to exist. As a result, the decision rule is ignored and both child nodes of the current node must be explored. Then, the outcomes of both child nodes are obtained by following their respective decision paths until a terminal node is reached, which generates two different leaf values (i.e. predictions). To combine both leaf values in a single one, both leaf values are weighted by the number of samples s that were passed to each child node at training time.

A simple example of the TreeSHAP algorithm can be done by using the tree in Figure 1.7, which uses the age x_1 and the marital status x_2 to determine how much an individual will

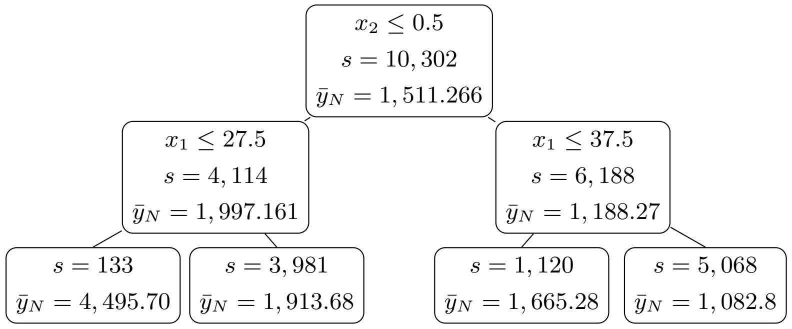


FIGURE 1.7 – Decision tree structure

claim in the next year. In this case, there are 4 altered functions $f_{\mathcal{G}}(\boldsymbol{x})$ to compute, as $\mathcal{F} = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$. To illustrate the prediction on the tree in Figure 1.7, consider a married individual who is 40 years old (i.e. $\boldsymbol{x} = [40, 1]$). If no feature is used, then the mean of the root node $\phi_0(f) = f_{\emptyset}(\boldsymbol{x}) = 1,511.266$ is returned. If $\mathcal{G} = \{2\}$, then the decision rule of the root node is applied, as $2 \in \mathcal{G}$. As $x_2 = 1$, the right branch is followed. The child node has a decision rule using x_1, which must be ignored. Thus, the weighted average value of the tree's leaves must be obtained. The left leaf has 1,120 corresponding observations and a value of $1,665.28$, while the right leaf has 5,068 corresponding observations and a value of $1,082.8$. The weighted average value of the child nodes is then $\frac{1{,}120 \cdot 1{,}665.28 + 5{,}068 \cdot 1{,}082.8}{1{,}120 + 5{,}068} = 1,188.27 = f_{\{2\}}(\boldsymbol{x})$. If $\mathcal{G} = \{1\}$, then the decision rule at the root node is ignored, as x_2 is not included, and the weighted mean of both of its child nodes must be returned. On both sides, the decision rule is based on x_1 and $1 \in \mathcal{G}$, so the rules are applied, which leads to their respective right leaves, as $x_1 = 40$. The right leaf of the left side of the tree has 3,981 corresponding observations and a value of $1,913.68$, while the right leaf of the right side of the tree has 5,068 corresponding observations and a value of $1,082.8$. The weighted average of the child nodes is then $\frac{3{,}981 \cdot 1{,}913.68 + 5{,}068 \cdot 1{,}082.8}{3{,}981 + 5{,}068} = 1,448.34 = f_{\{1\}}(\boldsymbol{x})$. Finally, if $\mathcal{G} = \{1, 2\}$, then the usual prediction of $1,082.8$ is returned.

To compute the marginal effect x_1, we must consider its effect when adding it to an empty set of features and its effect when adding it to x_2. This is equivalent to $f_{\{1\}}(\boldsymbol{x}) - f_{\emptyset}(\boldsymbol{x}) = 1,448.34 - 1,511.266 = -62.926$ and $f_{\{1,2\}}(\boldsymbol{x}) - f_{\{2\}}(\boldsymbol{x}) = 1,082.8 - 1,188.27 = -105.67$. The same can be done for x_2, where $f_{\{2\}}(\boldsymbol{x}) - f_{\emptyset}(\boldsymbol{x}) = 1,188.27 - 1,511.266 = -322.99$ and $f_{\{1,2\}}(\boldsymbol{x}) - f_{\{1\}}(\boldsymbol{x}) = 1,082.8 - 1,448.34 = -365.54$. Then, their average contributions can be computed, where $\phi_1(f, \boldsymbol{x}) = \frac{-62.926 - 105.67}{2} = -84.298$ and $\phi_2(f, \boldsymbol{x}) = \frac{-322.99 - 365.54}{2} = -344.27$. In other words, compared to the average predicted claim amount, being 40 years old reduces the predicted claim amount for the next year by 84.298\$ and being married reduces the predicted claim amount for the next year by 344.27\$. By putting these contributions and the average prediction into Equation 1.57, the predicted claim amount for the next year of this individual is $1,511.266 - 344.27 - 84.298 = 1,082.8 = f(\boldsymbol{x})$. As a result, the TreeSHAP algorithm allows the computation of the expected marginal effect of each feature to a prediction.

To explain individual predictions made by GBDTs, the TreeSHAP algorithm can be used to obtain the Shapley values of the ensemble of decision trees by summing up the mean marginal contributions of each feature for each decision tree.

In addition to the explanation of individual predictions, Shapley values have the advantage of providing a global interpretation for a model. By computing the Shapley values on an entire training set $\boldsymbol{S}$ and then aggregating them per feature, it is possible to obtain the absolute mean marginal contribution of each feature. Afterwards, these aggregations can give an approximation of the global behaviour of the model, and how each feature will make the prediction vary on average.

In the end, GBDT predictions can be interpreted accurately using the TreeSHAP algorithm by attributing exact marginal feature attributions to each prediction, while the GBDT can be explained by using the mean absolute Shapley values on any dataset. This enables the use of GBDTs in regulated settings where transparency, explanation, fairness is key and where black-box models cannot be used.

Chapter 2

Fairness in Machine Learning and Actuarial Science

Fairness, equity and justice have been pillars for modern societies to ensure that every single person is treated with impartiality, in that everyone is an equal as a human being. In itself, these principles have been codified into law for decades. However, it is important to note that in practice, most of these only stay in qualitative definitions that bear no standard in quantitative meaning. For example, the *Charter of human rights and freedoms* of Quebec [50] considers that the presence of discrimination is determined by the presence of a causality link between a sensitive attribute and negative outcomes, which is left open to interpretation [38]. As such, it does not indicate a specific threshold at which there is presence of discrimination. This shows the importance of defining fairness in a quantitative setting, as it has lead to an inconsistent application of the law.

Machine learning algorithms have long been considered able to produce fair outcomes that reflect the true nature of any dataset. The fact that no human takes part into making future predictions may tempt people to assume that no biases exist in these algorithms whatsoever. Nevertheless, with the increase of the use of these algorithms in regulated domains, legal bodies in different countries have began to scrutinise the impacts of automated decision making on society [48, 40, 62, 8, 19]. The externalities that come with their use bring new concerns into decision making as a whole. For example, many studies that have been made on some of these systems show that the outcomes produced by machine learning algorithms can sometimes discriminate towards parts of a population on the basis of gender [13, 45] or ethnicity [32]. In short, even if the process of making and using an algorithm is objective, the data and the experts that have been provided to train and validate the model may not be. That is to say that while negative biases may not be implied when fitting a model to predict future outcomes, the historical data that are used to train these algorithms can be biased. That was the case for Amazon's hiring algorithm, which selected preferred candidates for certain jobs. Its lack of historical data of good female candidates for IT jobs compared to their male counterparts inhibited the company from selecting new good female candidates [13]. As a result, future outcomes may be skewed by a model which contains unwanted implicit biases.

This chapter focuses on the different criteria that have been proposed to quantify biases in machine algorithms. At the same time, these criteria serve as a way to determine the appropriate criterion that would be needed to define quantitative fairness in different settings. We also focus on the methods that have been put forward to mitigate the chances of having a classification or regression model discriminate against a subgroup of people. Finally, we discuss the contradictions that exist between different definitions of fairness and the impact it has when used in actuarial science.

2.1 Fairness Criteria for Classification and Regression

The purpose of establishing different fairness criteria is to determine if disparate impact on a subgroup of individuals has taken place, which is the disproportionately negative effects that a policy has on a certain subgroup of individuals. Common examples of sensitive attributes are a person's gender, race, marital status, age, ethnic origin, religion or disability status. For example, in law, the four-fifth rule is a common convention that is often used to estimate that disparate impact has taken place [61]. For example, in a hiring process, that rule implies that no subgroup must be selected for a job more than 20% more frequently than any other subgroup. However, as it will be determined in the next sections, it is not clear what is considered preferential treatment in other contexts. The following proposals state possible solutions to quantify disparate impact in classification and regression problems.

Before getting into the different criteria where fairness is considered to be achieved, it is important to note that these criteria are not without conflict. First of all, fairness can be defined in different terms if dealing with individuals or groups of individuals.

Individual fairness is attained when the similarity of the outcomes of a model are proportional to the similarity of the individuals. Dwork et al. [15] formulated this definition as shown in 2.1, where both D and d are similarity functions:

$$D(f(\boldsymbol{x}), f(\boldsymbol{x}')) \leq d(\boldsymbol{x}, \boldsymbol{x}'). \tag{2.1}$$

A common example of this is when ensuring equal pay for equal work in the workplace. If a woman performs the same tasks as a man and that their curriculum are similar, then they should be paid approximately the same. If someone has more experience or takes on more tasks, then they should be paid in consequence. However, the difference in pay should be proportional to the dissimilarity between someone who has less experience and someone who has more. Hence, the similarity provided by function D should always be smaller than the similarity between two individuals provided by function d. This ensures that similar individuals are always provided similar treatment.

On the other hand, group fairness is presented in multiple forms that do not rely on similarities: it looks for the disparities in outcomes in different subgroups. As it will be described in the next sections, group fairness can be achieved in multiple ways. That includes having equal error rates per group or equal selection rates per group. As the concept of error and selection rate is inconsistent between classification and regression problems, they must be discussed in their own contexts.

2.1.1 Group Fairness Criteria for Classification Algorithms

In the most simple cases, classification models are used in a binary setting, where there is a positive outcome and a negative one. These outcomes, unlike continuous outcomes, are strictly bound by a probability combined with a threshold that binds the probability with the final label. For instance, a binary classification model normally outputs a probability between 0 and 1 of the positive outcome occurring. It then considers the positive outcome more likely than not if that probability is over 0.5. It is in this context that most fairness criteria for classification are applicable. They are often split into two types: selection rate parity or error parity. To simplify the notation in the next sections, $\boldsymbol{A}$ refers to the set of possible protected groups of persons, a is a specific subgroup in $\boldsymbol{A}$. Additionally, $\boldsymbol{X}_a$, $\boldsymbol{Y}_a$, $\hat{\boldsymbol{Y}}_a$ and $\boldsymbol{S}_a$ are the subsets of the observations $\boldsymbol{X}$, the labels $\boldsymbol{Y}$, the predicted labels $\hat{\boldsymbol{Y}}$ and the set of tuples of known observations and labels $\boldsymbol{S}$, where the observations in the subset are part of the subgroup a.

2.1.1.1 Group Unaware

Before any recent research was made into fairness in machine learning, it was assumed that a great way to ensure that an algorithm was fair was to exclude all information on subgroups from the training data. In return, an algorithm would not make predictions based on a sensitive attribute. However, it has been proven historically that models which do not take into consideration the presence of subgroups do not prohibit the model from discriminating against a subgroup. The presence of attributes which correlate with subgroups unknowingly encodes these subgroups in a dataset [15, 25]. This has been shown in multiple domains, such as in the insurance industry [47], where the outcomes were still highly correlated with gender even if gender was not present in the model. This was caused by the use of the employment of an individual, which is greatly predictive of gender. Hence, it is encouraged to include subgroup labels as features when available to quantify and mitigate bias in machine learning algorithms.

2.1.1.2 Demographic parity

Demographic parity, or statistical parity, is a fairness criterion in which the positive selection rate for all subgroups is equal [15]. For example, if a bank wants to ensure that the selection rate for a loan is equal for men and women, demographic parity would be the proper criterion. If the bank wants 50% of all groups to be accepted for a loan, half of all women and half of all men will be selected. In terms of the classifier, it would make predictions independently of gender. That is because you have the same chance of being selected, whether you are a man or a woman. In the simplest of cases, as the one pointed above, satisfying 2.2 would be equivalent to satisfying 2.3, where $\boldsymbol{A} = \{0, 1\}$ represents gender, $a \in \boldsymbol{A}$ is a specific value of either gender and $\hat{\boldsymbol{Y}} \in \{0, 1\}$ represents the outcome:

$$\hat{\boldsymbol{Y}} \perp\!\!\!\perp \boldsymbol{A} \implies \tag{2.2}$$

$$\begin{aligned} P(\hat{\boldsymbol{Y}} = 1|\boldsymbol{A} = 0) = P(\hat{\boldsymbol{Y}} = 1|\boldsymbol{A} = 1) \wedge \\ P(\hat{\boldsymbol{Y}} = 0|\boldsymbol{A} = 0) = P(\hat{\boldsymbol{Y}} = 0|\boldsymbol{A} = 1). \end{aligned} \tag{2.3}$$

One of the biggest critiques of demographic parity is that it does not take into consideration the distribution of outcomes with regards to the different subgroups. The major consequence of having an algorithm that gives equal selection rates is the disparity in the false positive rate of selected candidates. If one group is attributed a lower threshold of probability required for a positive outcome to increase its selection rate, then it would naturally select worse candidates as well. Higher false positive rates amongst groups with lower selection thresholds may even cause harm to the communities that were meant to be protected [39]. For one, it reinforces negative feedback loops about groups on which the algorithm performs badly on purpose: if all subgroups are selected at the same rate for loans, but one group tends to default more than others, experts will be weary of using the algorithm on that group. Also, by giving out too many loans to a disadvantaged group, it will be harmed in the long run as too many people default. That is why it is important to limit the balancing of selection rates to limit the harm of lowering the threshold for certain groups.

2.1.1.3 Equal Opportunity and Equal Odds

An important point about demographic parity is that it does not consider that there may be an actual existing correlation between $\boldsymbol{A}$ and $\boldsymbol{Y}$, even if it is not the causation of an outcome. The fact that a perfect classifier isn't always a solution for the demographic parity shows that it ignores the performance of an algorithm, along with the distribution of outcomes per subgroup.

Equal Opportunity [25] adds nuance to demographic parity by conditioning the independence between predicted outcomes and subgroups by true outcomes, in the sense that

$$\begin{aligned} \hat{\boldsymbol{Y}} \perp\!\!\!\perp \boldsymbol{A} \mid \boldsymbol{Y} = 1 \implies \quad & \quad (2.4) \\ P(\hat{\boldsymbol{Y}} = 1 \mid \boldsymbol{A} = 0, \boldsymbol{Y} = 1) = P(\hat{\boldsymbol{Y}} = 1 \mid \boldsymbol{A} = 1, \boldsymbol{Y} = 1). \quad & \quad (2.5) \end{aligned}$$

Instead of focusing on the selection rate of all individuals, equal opportunity takes aim at retrieving the same proportion of good candidates for each subgroup. In other words, it ensures that the recall, or true positive rate, is equal for each group. As a distinction, it gives the opportunity for anyone, regardless of subgroup, to get a positive result if they really are a good candidate. To put the previous scenario in the frame of equal opportunity, the goal would be to select an equal proportion of candidates who would repay the loan per subgroup instead of selecting an equal rate for loans among all men and women. This favours better quality results, because it considers that different factors to determine positive candidates may exist between subgroups. It also considers the distribution of positive candidates and prioritises recall as a metric.

Equal Opportunity has the advantage of being a simple single criterion between groups. Nevertheless, it is not applicable for all contexts. Notably, it only takes into consideration the quality of one type of outcome per subgroup. A classic case that is presented to counter it is a case where there are no particular positive or negative outcomes, such as the presentation of one of two possible job ads. As each company would like to have an equal proportion of appropriate candidates, having a single metric is not sufficient. Equal Odds is a stricter criterion that aims for equal selection rates for proper candidates of all outcomes:

$$\begin{aligned} \hat{\boldsymbol{Y}} \perp\!\!\!\perp \boldsymbol{A} \mid \boldsymbol{Y} \implies \quad & \quad (2.6) \\ P(\hat{\boldsymbol{Y}} = 1 \mid \boldsymbol{A} = 0, \boldsymbol{Y} = 1) = P(\hat{\boldsymbol{Y}} = 1 \mid \boldsymbol{A} = 1, \boldsymbol{Y} = 1) \land \quad & \quad (2.7) \\ P(\hat{\boldsymbol{Y}} = 1 \mid \boldsymbol{A} = 0, \boldsymbol{Y} = 0) = P(\hat{\boldsymbol{Y}} = 1 \mid \boldsymbol{A} = 1, \boldsymbol{Y} = 0). \quad & \end{aligned}$$

Afterwards, a classifier could be evaluated for bias if it either makes too many false positives or true positives in each group.

2.1.1.4 Error parity

In a more general sense, any other error metric can be used as a disparity measure if it is representative of the disparate impact that could result of the usage of the algorithm. Accuracy parity or loss parity could also be used as a fairness metric if it is appropriate for the context. They can be evaluated in two ways : as an absolute difference of losses or as a ratio of losses. These comparisons essentially come down to comparing the subgroup with the worst average error and the subgroup with the best average error:

$$\frac{\max\limits_{a \in \boldsymbol{A}} \left(\mathbb{E}_{(\boldsymbol{x}, y) \in \boldsymbol{S}_a} [L(y, f(\boldsymbol{x}))] \right)}{\min\limits_{a \in \boldsymbol{A}} \left(\mathbb{E}_{(\boldsymbol{x}, y) \in \boldsymbol{S}_a} [L(y, f(\boldsymbol{x}))] \right)} \leq \epsilon \tag{2.8}$$

$$\text{or } \left| \max_{a \in \boldsymbol{A}} \left(\mathbb{E}_{(\boldsymbol{x},y) \in \boldsymbol{S}_a} [L(y, f(\boldsymbol{x}))] \right) - \min_{a \in \boldsymbol{A}} \left(\mathbb{E}_{(\boldsymbol{x},y) \in \boldsymbol{S}_a} [L(y, f(\boldsymbol{x}))] \right) \right| \leq \epsilon. \tag{2.9}$$

Equations 2.8 and 2.9 are generic fairness measures but they differ from the previous ones in the fact that they are formulated as approximate fairness criteria, as opposed to absolute fairness formulations. Instead of ensuring equality of error rates, the error between groups lets a maximum discrepancy of ϵ occur. Most of the time, an exact solution may not be obtainable if the number of groups is too high. Hence, putting an upper bound on the maximum discrepancy between the errors of each group gives some leeway to classifiers and expands the number of possible solutions.

2.1.2 Group Fairness Criteria for Regression Algorithms

In comparison with binary classification problems, regression problems have an unlimited number of unique outcomes, which restricts the type of fairness criteria that can be applied on them. Equal odds, for example, cannot work in a setting that is not discrete, as it conditions error rates per label. As stated by Agarwal et al. [1], that leaves two group fairness criteria that are still applicable : demographic parity (2.10) and bounded group loss (2.11). Even if their formulation seems different, their goal are quite similar to fairness classification criteria:

$$\forall a \in \boldsymbol{A}, \forall z \in \mathbb{R} : |P(\hat{\boldsymbol{Y}}_a \geq z) - P(\hat{\boldsymbol{Y}} \geq z)| \leq \epsilon \tag{2.10}$$

$$\text{and } \forall a \in \boldsymbol{A} : \frac{1}{|\boldsymbol{S}_a|} \sum_{(\boldsymbol{x},y) \in \boldsymbol{S}_a} L(y, f(\boldsymbol{x})) \leq \epsilon. \tag{2.11}$$

As stated in its name, bounded group loss strides to limit the average loss of a group to a maximum of ϵ. This can be reformulated in the same form as error parity as a ratio (2.8) or a difference (2.9) between the biggest and smallest average error.

Equation 2.10 presents approximate demographic parity (2.2) in the case of continuous distributions. If each group has the same chance of having an outcome higher than z compared to the general population for any z, then outcomes would effectively be independent of the protected groups $\boldsymbol{A}$. Outcomes could be considered almost independent of subgroup if all discrepancies are lower than a small ϵ. Thus, even if it cannot be interpreted as equal selection rates, demographic parity ensures an equal distribution of outcomes when applied to regression problems.

2.1.3 Intersectional Fairness

The majority of work done on quantifying and mitigating bias on machine learning models has often focused on the simpler cases of discrimination, that is, contexts where there is a single binary sensitive attribute. While it is good to start with the basic cases to build upon them and generalise, it is crucial to understand the context in which fairness is needed and must be applied. Some work extends to multiple attributes [42] or multinomial attributes [66], but do not address the fact that discrimination does not only exist on independent attributes. The intersection of multiple sensitive attributes increases drastically the number of possible protected subgroups, and this highlights the direction in which the domain of fairness in machine learning should go.

The matter of the fact is that intersectionality is crucial in ensuring that all minority groups are considered when considering the fairness of an algorithm. One good case study on this is by Buolamwini et al. [5], in which they study the accuracy of modern commercial facial recognition algorithms. They noted that when considering independently gender and race, men or women, as well as black or white people, all were accurately classified in the same manner. However, by digging deeper, they noticed multiple discrepancies in the error rates that were compounded with the intersection of protected subgroups. For example, there was up to 20.6% difference in error rates between men and women faces and up to 19.2% difference in error rates between light and dark faces. When race was joined with gender, the contrast was starker, as darker women faced higher error rates of up to 34.7%. This increase of misclassification rate by a factor of 1.75 shows that minority groups can be left out of performance metrics if sensitive attributes are considered independently.

The undermining of minority groups in machine learning algorithms has taken place for a simple reason: there was no incentive for algorithms to perform well with all demographics. First of all, the datasets used to solve many problems were not representative of all subgroups. This created skewed distributions that lacked the proper information to process everyone properly. Secondly, the models and the loss functions used were not adapted to process skewed data accordingly. The models could be adjusted to consider imbalanced outcomes, but could not be adjusted if the outcomes where imbalanced for a specific subgroup. When all put together, this lead to algorithms that could ignore parts of the population by minimising the general loss on the general population, but that could not decipher the different factors that would produce accurate results. Intersectionality exacerbates these previous factors by making the point that subgroups are composed of other subgroups, and that they, in themselves, should not be forgotten. In return, many companies have now upgraded their facial recognition datasets and the transparency in their benchmarking so that they are now more inclusive of the entire population [44].

As noted by Foulds et al. [18], intersectional fairness should strive for fairness among independent sensitive attributes, among all intersections of sensitive attributes and that we understand the causes of disparate outcomes, whether they are structural, intentional or incidental. Those criteria can easily be dealt with, knowing that intersectional fairness leads to fairness with independant sensitive attributes. Since the intersection of attributes creates subsets of subgroups, we can bound an algorithm's fairness on groups by assessing its fairness on the intersection of all sensitive features. In any case, the subgroup treated the most unfairly will, at best, be considered better treated if it is summed with other groups which have an attribute in common with it [18]. For instance, taking into consideration that the highest mean error rate on a facial recognition algorithm is always on darker women, the mean error rate for women or darker people will always be lower. Thus, taking care of fairness under an intersectional lens will always consider equity at broader levels.

An interesting problem added by taking into consideration the intersection of attributes is the exponential number of constraints that result of that combination. The number of possible intersections is determined by the product of unique attribute values per sensitive attribute. $|\boldsymbol{A}|$ would then be the product of all bottom level groups, which is greater than their sum because they all contain at least a value. Considering that there are nine sensitive attributes and that there are at least two groups per attribute, there would be at least $2^9 = 512$ subgroups to consider, which is computationally expensive.

All points taken into consideration, intersectionality adds many challenges that have not been addressed by most works in the literature on fairness in machine learning, and must be in order to protect all individuals, no matter their ethnicity, gender, age, etc.

2.2 Supervised Learning with Fairness Constraints

While the previous section discussed how it can be determined that an algorithm is biased towards a subgroup, recognising a problem is only part of solving it. As such, the other part of ensuring fairness in statistical models is mitigating any discrimination that is detected by the metrics presented previously.

Bias can take place at any part of a machine learning pipeline. Hence, it can also be corrected at any time before a prediction is acted upon. On the other hand, not all bias mitigation methods are equal. Each has its costs and benefits and none are perfect. The existing methods alter data, the optimisation process or the predictions to make outcomes fair with regards to a certain fairness criterion.

2.2.1 Preprocessing

The idea of preprocessing the data in machine learning algorithms to obtain fairness stems from the fact that observed data can be skewed, can lack information or that sensitive attributes can be correlated with other attributes. Zemel et al. [69] introduced the idea of preprocessing the data by learning how to encode data into vectorial representations that do not include information about sensitive attributes. This can be done by creating a model which learns to map $\boldsymbol{x}$ to $\boldsymbol{x}'$, which is an encoded observation. In Zemel et al. [69], the encoder is built by creating a multiclass classifier that maps a label to a prototype (i.e. another existing $\boldsymbol{x}$).

The classifier outputs probabilities of $\boldsymbol{x}'$ being a good representation of $\boldsymbol{x}$. $\boldsymbol{x}'$ is then created as a weighted linear combination of the prototypes and the probabilities. The empirical risk function for the classifier has two essential parts : a fairness component and a reconstruction penalty. The fairness component ensures that each representation is as independent of subgroup as possible. The reconstruction penalty ensures that the resulting representation is as close as possible to the original observation. Altogether, the classifier creates a $\boldsymbol{x}'$ that is close to the original nature of $\boldsymbol{x}$, but is independent of its sensitive attributes.

Encoders have many advantages, as they can help reach demographic parity without knowing the eventual objective, are independent of model and don't require sensitive attributes when making future predictions, as they should not matter in the final representation. Nonetheless, because they do not have access to the objective, they lack the capacity of creating representations that are for other group fairness objectives than demographic parity.

Although there have been improvements on how to create fair representations that have been joined with in-processing methods [3, 4, 42], Zemel et al. limited their representations to a combination of existing prototypes $\boldsymbol{x} \in \boldsymbol{X}$. This limits the flexibility of these representations, as the relation between the original and the optimal transformed observations may not be necessarily linear. In fact, if new observations are out of sample, there may not be an accurate transformation possible. This loss of information makes classification unnecessarily inaccurate and may lead to disparate outcomes.

2.2.2 In-processing

Adding fairness to an optimisation process is intuitive, as it represents directly what needs to be accomplished. In other words, the general objective is still to be as accurate as possible, but the objective function is altered by the addition of a fairness constraint that limits its learning process.

This twofold objective can be subject to two different types of constraints : approximate constraints and absolute constraints. Absolute constraints can be thought of as all-or-nothing solutions : no unfairness whatsoever can take place at the expense of accuracy. This would create algorithms which respect exactly the criteria related to the constraints put upon them, such as demographic parity (2.3), equal odds (2.6) and equal opportunity (2.5) in their basic forms. In return, the predictions might not be accurate at all. Approximate fairness lets some unfairness take place to an extent, but does not forbid it with complete disregard to performance. Compatible metrics would be ones with relaxed requirements, such as error parity (2.8, 2.9), which can approximate any of the previous criteria.

2.2.2.1 Regularisation based fairness constraints

The first type of fairness constraint can be presented in a form similar to regularisation. The empirical risk in 2.12 is then split into two parts, a loss function L and a group fairness loss L_A,

$$L_S(f) = \frac{1}{|\boldsymbol{S}|} \sum_{(\boldsymbol{x},y)\in \boldsymbol{S}} L(y, f(\boldsymbol{x})) + \lambda L_A(f). \tag{2.12}$$

An example of a regression fairness penalty L_A was proposed in Berk et al. [2] where they try to minimise the distance between predictions produced by function d, weighted by the similarity between true outcomes among two groups a and a'. In order for this to work in practice for regression problems, all of the input attributes and outcomes are standardised so that their distributions have a mean of zero and a variance of one, while the function d is equal to Equation 2.13. For binary classification problems, all of the input attributes are standardised and the function d is equal to Equation 2.14.

$$d(y, y') = \exp(-(y - y')^2) \tag{2.13}$$

$$d(y, y') = I(y \neq y'), \text{ where } I(a) = \begin{cases} 1 & \text{if } a = \text{True} \\ 0 & \text{if } a = \text{False} \end{cases} \tag{2.14}$$

$$L_A(f) = \frac{1}{|\boldsymbol{S}_a| \cdot |\boldsymbol{S}_{a'}|} \left(\sum_{\substack{(\boldsymbol{x},y)\in \boldsymbol{S}_a \\ (\boldsymbol{x}',y')\in \boldsymbol{S}_{a'}}} d(y, y')(f(\boldsymbol{x}) - f(\boldsymbol{x}')) \right)^2, \tag{2.15}$$

where $a = 0$, $a' = 1$ and $\boldsymbol{A} = \{0, 1\}$.

To reduce the value of this objective as much as possible, the predictions must be as close to each other when the true outcomes are similar. The fact that the difference between predictions is not absolute allows compensation between positive or negative differences. As long as the average predictions across two groups even out, the group fairness penalty can be minimised. As such, this constraint aims for even outcomes between groups, as well as similar predictions for similar true outcomes.

As intuitive as it may be, there are many caveats to this constraint. Notably, it scales poorly if there are too many individuals as there would be at least $\frac{1}{2}|\boldsymbol{A}|(|\boldsymbol{A}| - 1)$ computations of Equation 2.15, where each computation contains $|\boldsymbol{S}_a| \cdot |\boldsymbol{S}'_a|$ comparisons of outcomes. Also, this constraint was only studied in the case where there are only two groups, which is rarely the case in practice.

For classification problems, most in-processing solutions were found using domain-adversarial neural networks (DANN) [20]. In short, these solutions [3, 4, 42] use DANN to create models that are unable to make a difference between two different subsets. These distributions can be, in this case, two different subset of individuals. In [3], Beutel et al. create a model that has two outputs : the first one is the normal prediction, the second predicts the subgroup of the individual. The goal here is to minimise the accuracy loss while maximising the loss with regards to predicting the subgroup. By inverting the gradient on the sensitive attribute prediction, the model learns to make predictions while not considering the individuals' subgroup. Hence, its predictions are independent of subgroup and satisfy the demographic parity criterion.

This was then adapted to work on other criteria, such as equal opportunity. Beutel et al. built on their previous work and replaced L_A with an absolute correlation loss [4]. In this case, the fairness penalty measures the correlation between the predictions $\hat{\boldsymbol{Y}}^+$ (2.17) made on observations with a positive true label $\boldsymbol{X}^+$ (2.16) and the respective subgroup of each observation with a positive true label $\boldsymbol{A}^+$ (2.18). The subgroups and outcomes must be binary for it to work.

$$\boldsymbol{X}^+ = \forall(\boldsymbol{x}, y) \in \boldsymbol{S} : \{\boldsymbol{x} | y = 1\} \tag{2.16}$$

$$\hat{\boldsymbol{Y}}^+ = \forall(\boldsymbol{x}, y) \in \boldsymbol{S} : \{f(\boldsymbol{x}) | y = 1\} \tag{2.17}$$

$$\boldsymbol{A}^+ = \forall a \in \boldsymbol{A} : \{a | \boldsymbol{x} \in \boldsymbol{X}^+ \wedge (\boldsymbol{x}, y) \in \boldsymbol{S}_a\}, \text{ where } \boldsymbol{A} = \{0, 1\} \tag{2.18}$$

$$\mu_{\hat{y}} = \frac{1}{|\hat{\boldsymbol{Y}}^+|} \sum_{\hat{y} \in \hat{\boldsymbol{Y}}^+} \hat{y} \tag{2.19}$$

$$\mu_a = \frac{1}{|\boldsymbol{A}^+|} \sum_{a \in \boldsymbol{A}^+} a \tag{2.20}$$

$$\sigma_{\hat{y}}^2 = \frac{1}{|\hat{\boldsymbol{Y}}^+|} \sum_{\hat{y} \in \hat{\boldsymbol{Y}}^+} (\hat{y} - \mu_{\hat{y}})^2 \tag{2.21}$$

$$\sigma_a^2 = \frac{1}{|\boldsymbol{A}^+|} \sum_{a \in \boldsymbol{A}^+} (a - \mu_a)^2 \tag{2.22}$$

$$L_A(f) = |Corr_{\boldsymbol{X}^+}| = \left| \frac{\left(\sum_{\hat{y} \in \hat{\boldsymbol{Y}}^+} \hat{y} - \mu_{\hat{y}}\right)\left(\sum_{a \in \boldsymbol{A}^+} a - \mu_a\right)}{\sigma_{\hat{y}} \cdot \sigma_a} \right| \tag{2.23}$$

If the absolute correlation is null, then the chance of dependence is minimised. However, since no correlation does not imply independence, it cannot guarantee equal opportunity. Using other measures of independence, such as mutual information [34], can ensure independence.

The approaches using DANN have the flexibility of being preprocessing approaches, as well as in-processing approaches. Since a fair neural network uses its last layer's outputs to make the final prediction, the entire network can be used as a fair feature extractor, thus being an encoder of fair representations. On top of that, these representations are much more flexible than the previously proposed preprocessing approaches, as they are not based on predefined prototypes. They can also yield an unlimited scope of results and can encode out-of-sample elements.

2.2.2.2 Risk based fairness constraints

The second form of fairness constraints is related to the domain of distributionally robust optimisation (DRO) and risk measures. These approximate constraints divert from normal objectives by focusing on observations with inaccurate predictions instead of aiming for the best average error possible:

$$\underset{\hat{\boldsymbol{\theta}}}{\operatorname{argmin}} \left\{ \max_{a \in \boldsymbol{A}} \left(\underset{(\boldsymbol{x},y)\in \boldsymbol{S}_a}{\mathbb{E}} [L(f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}), y)] \right) \right\}. \tag{2.24}$$

This can be adapted to a group fairness criterion by minimising the average subgroup error. This is done during the learning process by increasing the importance of the subgroups with the worst average errors. In an error parity setting, these restrictions allow models to minimise the maximum error over all groups, thus minimising the generalisation gap, or unfairness, between subgroups.

DRO is directly linked to risk measures that come from the domain of finance that were developed to indicate the probably worse loss performance possible for a certain portfolio of assets.

Two metrics, value at risk and conditional value at risk, are essential in order to understand how fairness can be achieved using DRO. Firstly, value at risk (VaR) 2.25 is the smallest value ξ such that the probability of obtaining a loss Z is lower than or equal to α. For example, let Z be the distribution of losses for a portfolio of stocks. Then its value at risk is defined as

$$VaR_\alpha(Z) = \inf \{\xi \in \mathbb{R} | P(Z \geq \xi) \leq 1 - \alpha\}. \tag{2.25}$$

All in all, VaR takes into consideration the potential risk of unforeseen events which would disproportionately increase losses for the owner of these risks. As it is possible to determine the VaR of a portfolio of risks based on the allocation its risks, it is possible to determine if the allocation of risks increases or decreases the VaR of the portfolio. The problem is that VaR is difficult to optimise, because it is not a convex objective [52]. To counter this, conditional value at risk (CVaR) (2.26) is a great alternative that can be optimised. CVaR, mean excess loss, expected shortfall, or tail value at risk, is the average loss over the VaR at percentile α. In other words, it is the average loss in a distribution from percentile $1 - \alpha$ to 1:

$$CVaR_\alpha(Z) = \mathbb{E}[Z | Z \geq VaR_\alpha(Z)]. \tag{2.26}$$

Both CVaR and VaR are very similar and can both be linked, as shown in equation 2.26 and figure 2.1. As CVaR is the mean value of the distribution Z above the VaR at a confidence level α, they are heavily correlated. If an actuary would like to ensure that his current reserves are sufficient to balance his portfolio of risks Z, he would need to know what is the exact upper bound of his possible losses at a confidence interval of 95%. This can be obtained by calculating the $VaR_{0.95}(Z)$. However, he would not be able to determine the optimal

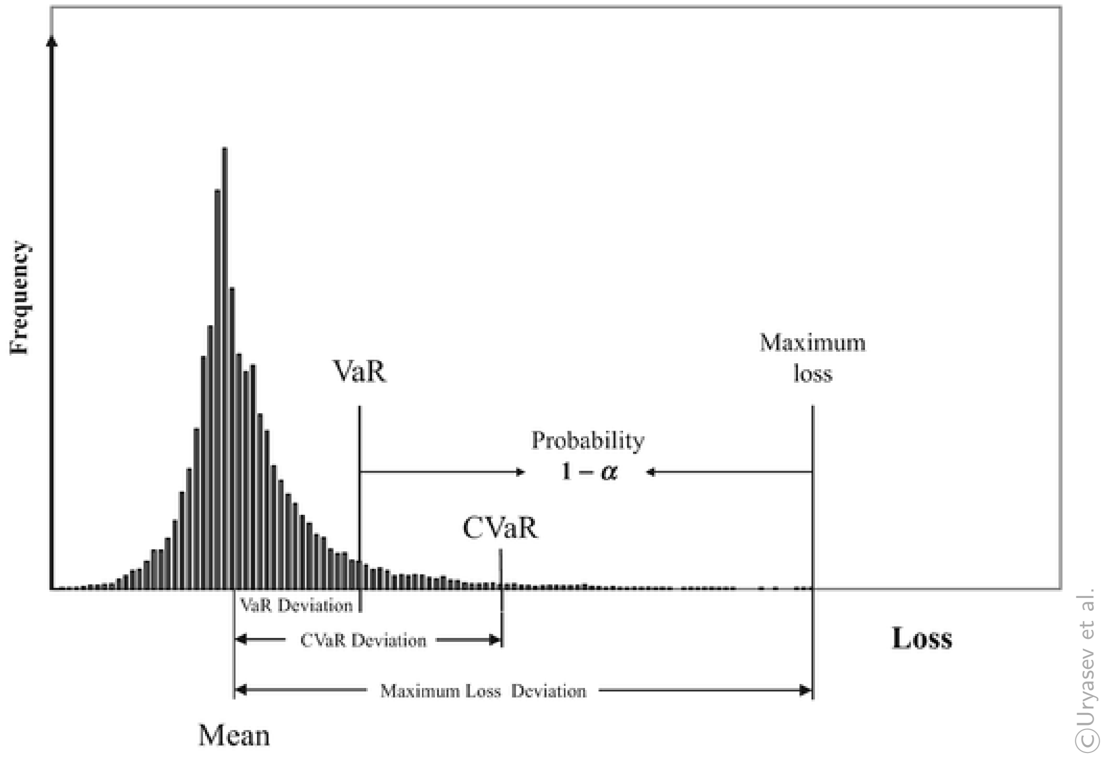


FIGURE 2.1 – Comparision of VaR and CVaR [56]

allocation of risks in order to keep the VaR as low as possible. The mean of Z above the 95th percentile, $CVaR_{0.95}(Z)$, does not mean much in itself as it is not a single point in the distribution. However, it has desirable properties that the VaR does not. It considers the distribution of losses exceeding the VaR, which cannot be done for VaR if the distribution is skewed. It can also be optimised as a smooth and convex objective [52]. Since the $CVaR$ is an upper bound to its VaR, minimising it naturally reduces the VaR. As such, only the $CVaR$ can be used as a proper objective for algorithms.

Since a loss distribution is usually not continuous and that risks are discrete, the discrete version of the $CVaR$ objective must be used. As presented by Williamson et al. [66], $CVaR$ can be used to reduce unfairness in any algorithm on a group-by-group basis, which is shown in Equation 2.27. In this case, m_a is the number of elements in set $\boldsymbol{S}_a$ and n is the number of protected groups in $\boldsymbol{A}$:

$$L_S(f,\xi) = \frac{1}{n\cdot(1-\alpha)}\sum_{a\in\boldsymbol{A}}\left[\frac{1}{m_a}\sum_{(\boldsymbol{x},y)\in\boldsymbol{S}_a} L(y,f(\boldsymbol{x}))-\xi\right]_+ + \xi, \tag{2.27}$$

$$\text{where } [v]_+ = \begin{cases} v & \text{if } v>0,\\ 0 & \text{if } v\le 0.\end{cases}$$

Compared to the original definition of CVaR (2.26), ξ and α play a different role in the formulation of the optimised empirical risk. Instead of being the true computed VaR, ξ is an additional parameter which must be optimised to minimise the empirical risk, but that also acts as a threshold for each group's mean loss. As such, the optimal value of ξ and the optimal function f that minimise the empirical risk L_S must be determined. The resulting objective is thus Equation 2.28:

$$\underset{f,\xi}{\operatorname{argmin}}\ L_S(f,\xi), \text{ where } L_S(f,\xi) \text{ is Equation 2.27.} \tag{2.28}$$

As for α, it is used as a tuning parameter which changes how much a model should focus on extreme values of the loss. As all losses over ξ are multiplied by $(\frac{1}{1-\alpha})$ and $\alpha \in (0,1)$, $1-\alpha$ is always between 0 and 1, all of the losses over ξ are increased by the same proportion. However, not all losses get the same absolute increase. As a result, the more α increases, the more the empirical risk adds weight to the largest subgroup risks. The group $CVaR$ objective (2.28) is then based on an empirical risk which ignores any group loss under ξ and that focuses only on the groups having the largest average losses. Thus, the optimised objective reduces the empirical risk over ξ instead of the mean of all the errors on $\boldsymbol{S}$.

The group $CVaR$ objective leads to algorithms with several desired characteristics. First of all, as discussed previously, the group $CVaR$ objective achieves error parity by reducing the generalisation gap, which bounds the empirical risks of all groups to similar values. Secondly, the group $CVaR$ objective is an approximate measure of fairness, which can be easily modulated and interpreted depending on the value of α. Thirdly, the group $CVaR$ objective increases the generalisation performance of a model on new data, since it aims to be robust on all groups.

The advantage of using in-processing methods is that the sensitive attributes are only needed at training time. Because protected group labels are in themselves very confidential, it is preferable that only a small subset of consenting individuals disclose their group labels, or that these labels be anonymized for everyone in order to avoid disclosing them [4]. As such, in-processing methods provide a way to keep the number of known labels to a minimum and doesn't require them when making predictions. In addition, these constraints always focus on the optimal compromise between performance and fairness. This can be adjusted easily by the importance of the approximate constraint, which can sacrifice more performance for more fairness in an incremental fashion. Finally, the outcomes are truly allocated in the fairest way possible, conditional to the task at hand. Notably, preprocessing approaches could come into situations where the representations are fair but that the outcomes are not, since the task may not be known when creating the representations.

However, there are many disadvantages to constraining only the training of the algorithm, and not the outcomes themselves. Even if error rates or selection rates are considered fair when training and validating the model, the fact that the data distribution changes over time might impact an algorithm's fairness. For example, if a fair algorithm leads to better

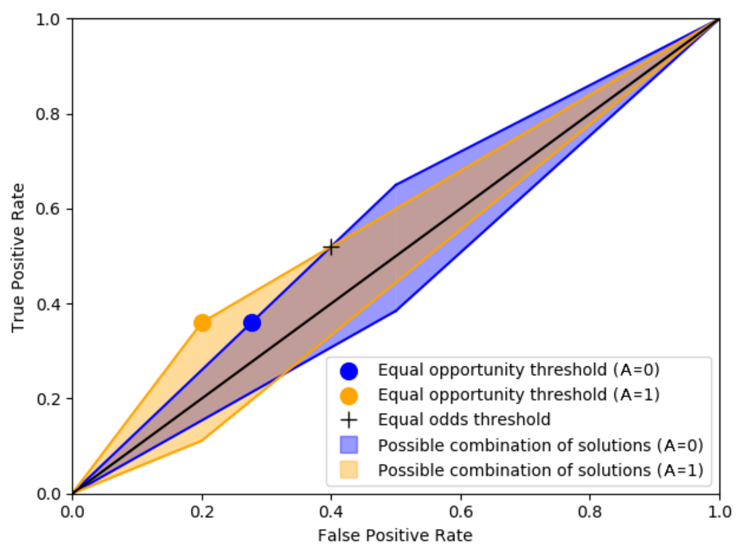


FIGURE 2.2 – Example of valid equal odds and equal opportunity thresholds.

outcomes for a certain group after many decades, the need for the fairness constraint will not be as critical as it once was. It could in fact lead to unfairness among individuals from the previously advantaged group. This is not a disqualifier, as long as the predictions and models are properly interpreted and monitored.

2.2.3 Post-processing

The final moment where predictions can be checked or altered to counter unfairness before making a decision is right before it is acted upon. In binary classification problems, post-processing methods rely on hard constraints or on altering the thresholds of each prediction with regards to their protected subgroups.

Hardt et al. [25] explained that selection rates can be adjusted for equal opportunity and equal odds by looking at a classifier's ROC by subgroup. Since these two criteria base themselves on the true positive rates and false positive rates, their effects can be visualised clearly by applying different thresholds to different groups on the ROC curves. Figure 2.2 illustrates two ROC curves made with the predictions of a toy model which was trained on a toy dataset. The dataset contains two subgroups of individuals: a blue group and an orange group. Each ROC curve assesses model performance for a separate group, which is designated by the colour of their ROC curve. The less opaque sections contain the set of possible TPRs and FPRs for each group, which can be done by altering the model's outcomes. As such, the ROC curves in Figure 2.2 provide the set of possible TPRs and FPRs per subgroup, which can help in obtaining either equalized odds or equal opportunity.

For equal opportunity, the thresholds need to be set so that all points end up on the same horizontal line. As discussed previously, even if these selected solutions lead to equal TPRs, the adverse impact is immediately shown on the ROC curve. Even if individuals from both groups have the same chance of being selected if they were a good candidate, the blue group will have a higher rate of false positives. If this disparity is too high, it may lead to active harm on the disadvantaged group.

For equal odds, the thresholds are not clearly defined. There can be a solution, as pointed out in the figure, that intersects both lines where both the TPR and FPR are equal. However, if these two lines never intersect, then a sub-optimal solution must be selected from the intersection of possible combination of solutions under each ROC curve. In essence, Hardt et al. use a *randomized threshold* to determine the final label of each element. In practice, the randomized threshold is applied after the subgroup labels, raw probabilities and true labels have been collected. When all data is collected, the explicit constraints, as laid out in equation 2.7, are placed in a convex optimisation problem solver. The solution provides two essential results per subgroup: how many elements must change from positive to negative and how many elements must change from negative to positive in order to achieve equal odds. Then, the amount of switches are performed on random individuals for all subgroups. This means that individuals with a positive prediction may randomly be selected to be changed to a negative prediction for the purpose of equal error rates and vice versa. This is why it is considered a randomized threshold.

Post-processing approaches have their purpose and have a proper context for them to be used. If absolute fairness is essential in the target setting or the target model is already in production, then post-processing approaches are clearly the methods of choice. The fact that they are model agnostic and that they don't constrain the model whatsoever adds a flexibility that affects minimally existing processes. Also, they are the most straightforward of all the approaches, as they transparently change outcomes to make them fair with regards to subgroups.

Some subtle but crucial points make it less appropriate to use absolute constraints in some contexts. If for example, a dataset contains a severely unbalanced group that has very few positive candidates, then the only solution for equal opportunity might be to return no positive candidates at all. After all, if there are no positive predictions, all groups would have an equal TPR of 0, which defeats the purpose of using a predictive algorithm altogether. The approach also does not set a target TPR, which is necessary in many settings. In cases like cancer diagnosis or job selection where false negatives are costly, only good positive candidates need to be selected. As such, higher thresholds will lead to lower recall, but higher precision. If the solver chooses thresholds which results in a high FPR, that cannot be easily adjusted.

Also, if there are too many groups and equal odds are required, the intersection of possible solutions may lead to poor classifier performance. This is because only fairness, and not performance, is considered. Other approaches, such as in-processing methods, can try to find a compromise between fairness and performance. A final critical point on this approach is the use of randomized thresholds to balance error rates which leads to inner group unfairness. If someone is in an advantaged group, this method may cause the person to be negatively affected, with only justification that the individual was part of a certain group, which can be unfair with regards to the other individuals in that group.

In hindsight, no method is perfect in its ability to ensure absolute fairness in all situations, as regression and classification problems have different varieties of outcomes. Also, the number of groups can lead to an exponential number of checks that may constrain algorithms too much if they are enforced too strongly. That is why it is critical to check the context where fairness is needed.

2.3 Fairness in Actuarial Science

The use of statistical models in actuarial science is not new. In fact, for over thirty years [22], the use of GLMs has enabled insurers to provide insurance premiums tailored to each client's risk. However, fairness was a concern in the insurance industry even before these models were put in use. Historically, insurance companies have been the target of many government regulators because of discriminatory policies that negatively affect minority groups. Hence, many laws are now in place to make sure that insurers are fair in the distribution of the pool of risk amongst their clients.

It is essential to review the previous and current policies, their effects and what can both the domain of actuarial science and the broader machine learning community can learn from each other in order to find balance in profitability, affordability, fairness and precision. As a case study, the history of insurance in the last century can provide valuable insights into what positive and adverse impacts fairness regulations can have on individuals and companies in other domains that use machine learning.

Actuarial fairness has the objective to give "equal treatment for equal risks" [19, 37]. In the instance of car insurance, this means that any two individuals with the same car and the same driving history should get the same premium. As they should not incur different losses for the same policy period, there is no reason for them to pay different rates. This also means that unequal risks should result in different premiums. In the long run, actuarial fairness implies that policyholders should only pay for their expected losses, as their premiums "must reflect the division of people according to actuarially accurate determination of their risks" [12]. If two individuals have identical situations and claims over time, but their initial premium was different, those risk estimations were unfair. Therefore, an accurate premium that appropriately estimates a client's expected losses is fair.

2.3.1 Impacts of Fairness Regulations in Actuarial Science

2.3.1.1 Fairness Regulations for Home Insurance

As much as it has been attributed to the financial industry for mortgage lending, redlining is a practice that has also affected minority groups with regards to insurance policies. Redlining generally refers to discrimination caused by racial profiling based on geographical location. This results in the loss of economic opportunities and the decay of certain urban areas [55]. By disproportionately raising premiums, offering worse policies, denying coverage and providing lower settlements, insurers in the United States [55] and Canada [26] have been subject to many out-of-court settlements due to unfair discrimination [24, 57].

Even if it is not directly related to prosperity amongst policyholders, property insurance plays a crucial role in home ownership. In Canada, home insurance is a key component into obtaining a mortgage that allows for home ownership. According to the Insurance Bureau of Canada,

> Unlike auto insurance, home insurance is not mandatory by law. However, most banks or mortgage holders will insist that you purchase home insurance – and show proof that you have home insurance – before they'll lend you money to buy a home. They will ask to be named as the mortgage holder on your policy. [6]

As such, even if it is not necessary for all cases, most people will need home insurance if they want to buy a home.

The impact of a premium increase in isolation might seem minor. Nevertheless, the requirement increases total costs for potential home owner who might not have the additional revenue for it. The Canadian Real Estate Association indicates that "increases in premiums can prohibit the marginal homebuyer from being able to afford homeownership and can severely strain the ability of existing homeowners on fixed incomes to continue to meet the ongoing costs of homeownership" [65]. Hence, affordability and availability of coverage is essential to give everyone the opportunity to get a mortgage and thus own a home.

It is normal that price increases reduce consumer buying power. The case for redlining is justified by the refusal of coverage and the higher premiums are hurting minority groups at a much higher rate than others. For example, Squires [55] studied the rare instances where insurance companies had to disclose publicly their coverage at a zip code level. He noted that in 1999, for the city of Milwaukee, 72.4% of homes in white majority neighbourhoods were covered while only 61.6% were covered in black majority neighbourhoods. This significant disparity puts communities that are already at higher risk of future harm by depriving them of insurance.

Coverage and price disparities are normally justified by economic, societal and property related factors. The geographical location of homes gives insight into a building's age, the quality of its construction and average revenue of the area [55]. Credit scores are often considered as a proxy between paying back loans and taking care of your property [36, 23] or the likelihood of fraudulent claims [55]. Even if those attributes are in themselves indicative of future losses, they have indirectly lead to statistical discrimination towards certain subgroups of individuals. In some cases, even controlling for income, black majority zip codes pay more dollars per loss than white majority zip codes [33]. Also, credit scores have been criticised for targeting disproportionately people with disabilities, women and racial minorities in ways that significantly inhibits their future opportunities [23]. This lead to Newfoundland and Labrador to ban its use for all insurance ratings [46]. The Federal Trade Commission also determined in 2013 that 5% of credit scores were based on erroneous historical data [9], which reduces their reliability. Hence, zip codes, socio-demographic data and credit scores should be taken with a grain of salt and should be monitored to ensure fairness.

As a response to these measures, individuals located in the United States have used the housing accessibility provisions from the Fair Housing Act to rein in insurance companies. Since home insurance is required to obtain a mortgage and that a mortgage is needed most of the time to own a property, discriminatory pricing reducing accessibility to housing falls under that law [33]. The same argument could be made under certain laws of many Canadian provinces, including Quebec [50] (except for gender and age, which will be discussed later).

Even as these laws exist to prohibit disparate impact for home insurance, they are rarely enforced. Regulatory agencies often fail to act, because they do not have the tools to do so. In order to determine that discrimination has taken place, historical data that are currently not disclosed are needed. Some have proposed that insurers should disclose annually the information on their clients, including their premiums, race, gender, income, coverage or location [55]. This is already done in the US for mortgages under the Home Mortgage Disclosure Act [10]. By having public access to anonymised outcomes containing all associated sensitive attributes, anyone can easily study if the allocation of risk is fair with regards to any protected subgroup.

2.3.1.2 Fairness Regulations for Auto Insurance

For auto insurance, the need for fairness has been discussed for more than thirty years [21]. States such as Michigan, Montana, Hawaii and others had unisex rating for auto insurance since the early 1980's [51]. In Canada, the provinces of British Columbia, Saskatchewan, Manitoba, Newfoundland and New Brunswick all prohibit the use of gender as a factor of risk [64].

As with property insurance, auto insurance has been regulated to ensure fairness for certain groups. The difference is that there have been many more studies on the effects of these regulations on groups of individuals. In the past, the auto insurance industry faced many problems with regards to providing coverage to everyone and abiding to fairness laws. Regulators know that their mandate is to mitigate these problems without making insurance inefficient or unprofitable as a whole. As a result, in 1974, the Federal Insurance Administration of the United States proposed several policies that tackled the use of sensitive attributes and that increased the availability of policies for all groups of individuals. Following suit, states having the same concerns such as Michigan, North Carolina and Massachusetts implemented certain measures to ensure accessibility and fairness in their insurance markets [21]. Their different policies and their effects were studied by the General Accounting Office (GAO) of the United States.

The states of North Carolina and Massachusetts banned the use of age and sex in 1977 and 1978 respectively. The policy had a positive effect, even if young drivers still paid more for the same coverage as older ones. Notably, the differential of premiums between the two groups was severely reduced and the number of insured young men increased. This was caused by the use of other important factors of risk, such as the years of driving experience, which is highly correlated with the driver's age. As a counterexample, in 1981, the state of Michigan implemented the Essential Insurance Act, which prohibited different insurance ratings based on an individual's sex or marital status [58]. Its impacts were then studied by the GAO, where they found that it resulted in a decrease in policy prices for young single men by as much as 20.4%, while increasing it by 28.9% for young single women [21]. While these cases do show that fairness regulations for auto insurance do have an effect on policy prices, the GAO could not establish a relationship between state regulations and the profitability of insurance companies.

Practices however, have changed over the years, and some jurisdictions have have put in place fairness laws for auto insurance ratings only recently. For example, the European Union enacted in 2004 a law demanding that insurers provide unisex auto insurance ratings [16]. Its impacts were not considered substantial, as it was analysed that the premiums use of profession, instead of gender, resulted in minor changes to premiums since they are both highly correlated [43].

These cases show that no location is alike, as they have different assets, risks and individuals. In the presence of the use of valid risk factors that are correlated with sensitive attributes, prices rarely differ after the implementation of the anti-discrimination laws. If there are no substitutes to the sensitive attributes, the resulting premiums may not reflect the true risk of each individual.

2.3.2 Comparison with Fairness in Machine Learning

By putting insurance anti-discrimination laws in the context of fairness criteria from the domain of machine learning, it is possible to determine many flaws in the approach taken by regulators and the insurance industry as a whole. The major flaw in insurance anti-discrimination laws is that they implement a "group unaware" approach to fair risk ratings, while demanding that premiums are fair with regards to sensitive subgroups.

All of the following consequences indicate that fairness cannot be ensured without knowing all of the sensitive attributes of each individual. First of all, it is impossible to confirm the presence of unfair risk assessments, as insurers do not know their risk distribution with regards to their clients' subgroups. Second, by being unable to determine the presence of unfairness, insurers are unable to correct bias if there is any, while regulators cannot assume that insurers have the capacity to be fair. Third, insurance anti-discrimination laws come directly in conflict with the definition of actuarial fairness presented previously. Giving equal treatment to equal risks is equivalent to the definition of individual fairness (2.1). However, anti-discrimination laws recognise discrimination on a group level basis, not on an individual one. If a majority group gets premiums that are more representative of their true risk, while minority groups do not, similar individuals who are part of different groups may not obtain the same premium. Giving inaccurate premiums to a minority group may give it unnecessarily higher average premiums for a similar true level of risk. Also, the fact that there is no distance metric that defines the similarity of individuals makes it impossible to obtain a certain form of individual fairness. Fourth, any regulation with regards to protected subgroups do not consider the intersectionality of sensitive attributes. As a result, any scrutiny that has been put onto insurance companies to give fair premiums in the past did not consider that the combination of some attributes may uncover discriminatory practices. This will continue under future audits as long as regulations do not cover the intersection of sensitive attributes. Finally, using a group unaware approach to risk assessment has historically lead to unfair premiums when there were no correlated attributes, or minor changes when there were other correlated attributes available. This means that their removal prohibits their explicit use as crutches for a lack of information. However, the presence of legal and relevant correlated risk factors show that it is impossible to remove unfairness by truncating sensitive attributes: the discrimination should be removed from the outcomes themselves.

2.3.3 Impact on Performance

In the insurance industry, giving precise ratings is not only beneficial to clients wanting competitive premiums, but also to insurers. As noted previously, abiding to a fairness constraint frequently requires sacrificing performance. For insurers, this means that giving premiums that are less precise for fairness purposes will lead to lower profitability. For clients, this will mean getting unjustifiably higher or lower premiums for some individuals. Clients with premiums truly higher than their risks will either take an unwarranted financial penalty or change insurer to obtain a competitive rate. On the other hand, clients with rates lower than their true risk will end up sticking with their current insurer while claiming more than they paid in premiums, leading to unprofitability for the insurer. This phenomenon, called **adverse selection** [22], exists when an insurer attracts high risk individuals with lower rates to a disproportionate degree, thus repelling low risk clients. By keeping too many high risk individuals erroneously considered as low risk, the insurance company must raise all of its clients premiums to keep afloat, as its entire pool is more risky on average than it once was. Since it increases everyone's paid premiums, risk averse individuals will easily find a better price at another company, while high risk individuals will stay as they cannot find a better premium anywhere else. As a result, the mean risk of the client pool increases once again and the insurers profitability spirals out of control until a more precise risk assessment can be done. Hence, performance is still an important factor when trying to produce fair premiums, as both the client and the insurer lose if the risk assessments are not precise.

Chapter 3

Customer Profitability Forecasting with Fairness Constraints

Estimating the frequency and the severity of an unplanned event occurring, in which a current or future client has negative financial consequences, is a central part of an insurance company's business. For home and auto insurance, the goal is to predict the chance of theft or damage occurring on a client's property, as well as the amount of financial loss related to the event.

By evaluating how much a client's property is prone to theft or damage, an insurer can estimate how much it should charge that client on average to cover the cost of repairing any damage that may occur. It also gives the general portrait of a client and indicates how much the insurer should reserve money for a client's liabilities. Knowing how much a client's property is prone to theft or damage helps an insurer keep a competitive edge over its competitors, while offering coverage to all individuals who need it.

Before jumping deeper into how this is done, it is important to define the underlying terms related to this task. Machine learning practitioners, along with actuaries, both use statistical models to describe, explain or predict outcomes by using historical data. Although the two of them use similar methods, they often use different terms to describe the same things, or use the same terms to mean different things. That is why, prior to explaining the task at hand in more specific terms, it is crucial that all of the terms used are explained in the context of actuarial science. As some words were defined differently in the previous chapters, it is essential to make the distinction between the many meanings a word may have depending on the context.

In the context of property insurance, the **risk** is the probability that damage will occur on a client's property. Thus, instead of being related to the error of a statistical model, the "risk" is mostly used to quantify how much a client's property is prone to damage. In the case that a client claims a financial compensation for damages done to its property, that compensation is considered a financial **loss** for the insurer.

In order to set the price to cover a client's assets, insurance companies must predict the amount of loss a client will claim due to damage done to his/her assets during a future period of time. As the amount of potential loss varies greatly depending on the number of assets covered and the duration of the coverage, the goal is to predict the amount of loss for a single asset over a period of an entire calendar year. This is called the **pure premium**. As insurance companies also have other overhead costs, such as operational, administrative and marketing costs, they must also add a part of these costs to each customer's dues, which results in the **rate** or **premium**. As a client may have a policy that covers a period other than a full calendar year, the final price paid by the client for the entire defined period is the **gross rate** or **gross premium**.

In exchange for paying the gross premium, a client is given an insurance **policy**, which is the contract in which the insurer promises that it will indemnify financial losses in the occurrence of a covered loss. Each policy is attributed a unique identifier, a period of enforcement (i.e. a policy period) and may cover one to many assets. As such, the main part of insurance pricing is to estimate, predict or assess a client's total claimed amount over a period of a full calendar year for each of his/her assets.

3.1 Customer Profitability Forecasting

A related task to pure premium estimation is the more generic task of estimating customer profitability. In its simplest form, customer profitability estimation is predicting the total cost incurred by a client and then determining if the income sources provided by that client cover the predicted total cost. For insurers, this can be translated by the insurance premium paid minus the incurred losses for each customer. Optionally, overhead costs, such as marketing costs, can be added to this equation. As this type of problem requires data that is laid out along many years, costs and premiums will have to be updated to current prices accordingly. Since the total premiums are already set by the insurer, the only remaining unknown variable in the equation is the incurred costs per policy, or the total claimed amount by a client for a certain policy period. As such, predicting a client's losses enables insurers to determine which clients are profitable.

Most importantly, customer profitability forecasting enables insurance companies to target the clients which are the most essential to its survival. Afterwards, the customer loyalty schemes can be performed through better quality services, efficiency and maximising customer satisfaction, but also through targeted advertising and lower total premiums. The goal is then not only to be profitable as a company, but to stay profitable throughout the years by keeping a base of loyal profitable clients. This minimisation of adverse selection is thus obtained through accurate risk estimations and targeted customer loyalty schemes.

In addition, the forecasting part of this problem helps actuaries determine the optimal amount of funds that must be put aside for the next year in order to compensate for all losses, and what should be the total amount of risk distributed amongst the pool of clients in premiums.

3.1.1 Problem Statement

As the costs related to a client's policy are an essential part in determining client profitability, it is crucial to study how a premium is estimated. The premium of a certain coverage can be decomposed into multiple components, as shown in Equation 3.1. Notably, it contains the amount of money required to cover the losses per unit of risk (i.e. the pure premium). The **units of risk** (3.3) represent the yearly duration of the policy. As such, a policy with a duration of two years will have two units of risk.

$$\text{Rate} = \text{Premium} = \text{Pure Premium} + \text{Overhead costs} \tag{3.1}$$

$$\text{Gross premium} = \text{Rate} \cdot \text{Units of risk} \tag{3.2}$$

$$\text{Units of risk} = \text{Duration of policy (in years)} \tag{3.3}$$

$$\text{Pure premium} = \text{Frequency of claim per unit of risk} \cdot \text{Average severity of claim} \tag{3.4}$$

$$\text{Yearly profitability} = \text{Paid premium} - \text{Total amount claimed during a full year} \tag{3.5}$$

Altogether, the frequency of claim per unit of risk and the average financial loss per claim compose the pure premium (3.4), which is equal to a client's expected future losses during the next policy period. That being said, the actual premium (3.1) is not equal to the pure premium, as premiums also include overhead costs.

For this work, customer profitability forecasting refers to determining the difference between a client's premium and losses for a future year (3.5). Thus, a policy holder is determined profitable if his/her pure premium is lower than his/her total premium. The goal is to predict the client's total claimed amount during a full year, which is equivalent to estimating the pure premium.

As with any supervised learning, the task of customer profitability forecasting relies on historical data to determine future outcomes. In this case, the training set $\boldsymbol{S}$ has three components instead of two: $\boldsymbol{X}$, $\boldsymbol{W}$ and $\boldsymbol{C}$.

Each observation $\boldsymbol{x}$ in $\boldsymbol{X}$ represents a single asset and the policy its assigned to during a year when it was covered. As it is rare that two policies start and end on the same day, all policies are split into multiple periods where each year a policy was enforced has an observation related to it. In return, each row is assigned units of risk $w \in \boldsymbol{W}$, which represent the span of a calendar year where the policy was enforced. For example, if a policy with a single asset has an effective date set as three quarters into the year 2019 and that the policy is effective for a full year, the policy will be found on two different occasions in the training set: once in 2019 with 0.25 unit of risk and 0.75 unit of risk for the year 2020. This must be considered when training a model, as those two observations do not bear the same level of importance. One represents three quarters of a year, while the other represents only a quarter. Also, the pure premium is directly proportional to the duration of the policy and the number of assets. Therefore, a model must consider that observations that encompass more time and assets carry a bigger weight, but naturally contain more losses.

All of the considered attributes x_j in $\boldsymbol{x}$ are called **risk factors**, as they help quantify the potential risk contained in the policy. For home insurance, this can be the age of a client, his/her credit score, the year the policy was in force, the maximum amount covered for flooding, the price of the house, the number of past claims in previous years, the proximity of a fire hydrant, the postal code of the residence, etc. Anything that is related to the behaviour of the individual or the risk level of an asset can be used as useful information for the task, and $\boldsymbol{x}$ can contain up to hundreds of attributes.

The outcomes $c \in \boldsymbol{C}$ for training observations $\boldsymbol{x} \in \boldsymbol{X}$ represent the claimed amount for a single asset during a certain calendar year. If a policy is enforced in both 2019 and 2020, and a claim was made in 2019, the claim will only appear once in the training set: on the observation for the year 2019. These observations even themselves out when put together to consider the whole duration of the policy. Since all observations must have their outcomes c normalised by their units of risk w during model training to consider the variable span of time that they cover, the outcomes to predict then become the claimed amount for an entire calendar year y, which are contained in the set $\boldsymbol{Y}$. Thus, a predictive model needs to predict the amount that a client will claim for a single asset during a full year (i.e. the amount claimed for a single unit of risk).

As shown in Equation 3.4, the pure premium can be decomposed into many components. The pure premium can be decomposed into the combination of two independent factors: the expected frequency of losses per unit of risk and the expected amount claimed per loss. The frequency itself is found only in whole numbers in observations. Nevertheless, the predicted frequency of claims are usually not whole numbers, as the expected predicted frequency for a certain $\boldsymbol{x}$ in an entire population might not result in a whole number. On the other hand, the severity of each claim presupposes that a claim was made, and thus only considers policies with non-zero claim amounts. As a result, someone can have a high premium based on the fact that the chance that claims will be made are high, or that the claim amount will be high, and may not be correlated. For example, in auto insurance, a driver with less driving experience may make more accidents on average, but have a car which doesn't cost much to repair. In return, a driver with more experience but a pricier car will have a lower claim frequency, but a higher severity of claim.

A way to formulate the profitability of a group of policies is to consider its loss ratio. A **loss ratio** (3.6) is the sum of the claimed amounts y divided by the sum of the predicted

Distribution	**Mean**	**Variance**
Poisson	$\mu = \lambda$	$\mu = \lambda$
Gamma	$\mu = \alpha\gamma$	$\phi\mu^2 = \alpha\gamma^2$
Tweedie	$\mu = \lambda\alpha\gamma$	$\phi\mu^\rho = \lambda\alpha\gamma^2(\alpha+1)$

TABLE 3.1 – Mean and variance parameters for Poisson, Gamma and Tweedie distributions

claim amounts $\hat{y}$. It represents the percentage of financial losses that would be covered by the predicted claim amounts if they were the premiums paid by clients. As long as a loss ratio is under 1 for a certain set of policies, they are considered profitable, as the income is greater than the claimed losses:

$$\text{Loss Ratio for population } \boldsymbol{S} = \frac{\sum\limits_{(\boldsymbol{x},y)\in\boldsymbol{S}} \text{Claimed amount}}{\sum\limits_{(\boldsymbol{x},y)\in\boldsymbol{S}} \text{Predicted claim amount}} = \frac{\sum\limits_{(\boldsymbol{x},y)\in\boldsymbol{S}} y}{\sum\limits_{(\boldsymbol{x},y)\in\boldsymbol{S}} f(\boldsymbol{x})}. \tag{3.6}$$

3.1.2 Generalised Linear Models for Pure Premium Estimation

The use of statistical models to tackle the task of pure premium estimation is not new. For decades, actuaries have used GLMs to estimate the pure premiums of their clients. Their simplicity, interpretability and statistical assumptions make them easy for debugging, auditing and confirming hypotheses concerning each risk factor. Also, their use of maximum likelihood estimation makes their optimisation intuitive with regards to justifying the loss function used.

3.1.2.1 Poisson and Gamma GLM Models

If we want to estimate pure premiums using GLMs, we can look at the distributions of the frequency and severity components. As the frequency of claims per unit of risk is a positive count of events, the Poisson distribution is an ideal representation of the outcomes. As for the severity component, its distribution is non-negative and has a long tail, as claims with huge amounts do occur, although they occur rarely. Therefore, the Gamma distribution is an appropriate representation of the amount claimed. As such, the task of pure premium prediction can be solved by using two separate models: a Poisson GLM $f_p(\boldsymbol{x})$ and a Gamma GLM $f_g(\boldsymbol{x})$. Both models can have their log-likelihood obtained from their density functions and they can then be optimised with stochastic gradient descent (1.2). By multiplying the outcomes of both of these models, a pure premium is estimated.

Splitting pure premium computation into multiple models can be useful when the analysis of risk factors x_j is of utmost importance. As each risk factor might not have the same importance towards the frequency as it has towards the severity, the coefficients of each model give a more granular explanation of the pure premium [22]. In addition, only the frequency model is dependent on the policy's units of risk. If the analysis of risk factors by each type of claim is required, this can be pushed even further by assigning a frequency and a severity model to each of them. It can be useful, considering that the risk factors of loss due to a fire are different from those due to flooding.

Poisson GLMs are also advantageous because they can naturally integrate the units of risk into the model with minimal required work. As Poisson GLMs usually use the log-link, they allow to easily add terms to the model that do not need any coefficient β. Considering that claim frequency is directly proportional to the duration of the policy, the duration of the policy doesn't require a coefficient. The duration of the policy can be added to predictions by adding the log-transformed value of the duration of the policy as an **offset**. For w units of risks, the offset is equal to $\log(w)$, as reflected in the Poisson GLM $f_p(\boldsymbol{x})$ represented in Equation 3.7:

$$f_p(\boldsymbol{x}) = \exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \log(w)) = e^{\hat{\beta}_0} \cdot e^{\hat{\beta}_1 x_1} \cdot ... \cdot w. \tag{3.7}$$

Thus, when fitting the model, the units of risk w related to the row $\boldsymbol{x}$ are used, along with its observed frequency of claims during that policy period. Afterwards, when making a prediction, the predicted claim frequency $f_p(\boldsymbol{x})$ can easily be obtained by setting $w = 1$, as we only need to estimate the claimed amount per unit of risk.

To increase model stability, other transformations can be performed on the training set to reduce the effect of outliers. Notably, outcomes can be capped to a certain amount to reduce the weight of certain observations on the entire population. If a house worth 1 million dollars is lost due to fire, then it would highly skew the mean and variance of the distribution of outcomes. This would result in inaccurate estimated pure premiums for the majority of the population. To minimise the effect of aberrations, outcomes in known observations can be capped up to a certain amount. As the capped outcomes are still high compared to the rest of the population, the capped outcomes will still be considered as high risk and have higher estimated pure premiums.

3.1.2.2 Tweedie GLM Models

Although the use of separate models for frequency and severity is most common in actuarial science for better interpretability, it is also possible to use a single robust model to predict pure premiums. Since GLMs can be used with any density function related to the exponential family of distributions, one must look for a distribution in that family that is similar to the distribution of pure premiums. A pure premium distribution notably has a high density at zero, considering that most individuals do not claim anything for the duration of their policies. However, as shown previously, the distribution of claim amounts is highly skewed, similarly to a Gamma distribution. The resulting distribution is what is known as a Tweedie distribution [60], as illustrated in Figure 3.1.

The parameters of the Tweedie distribution are similar to the ones other distributions in the exponential family have. It has a mean parameter μ, a scale parameter ϕ, but it also has a *power* parameter ρ. ρ is a shape parameter, which changes the mean-variance relation

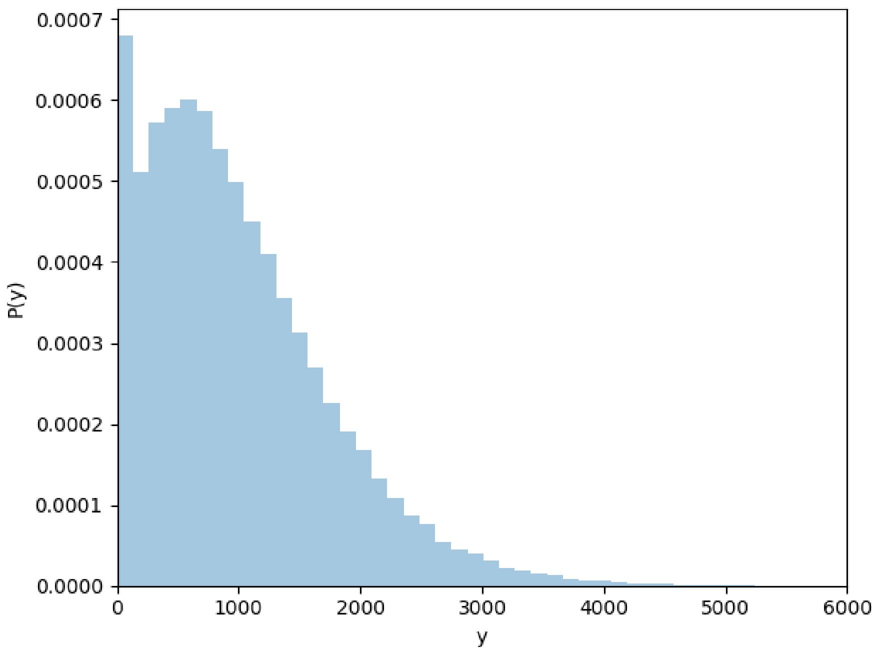


FIGURE 3.1 – A Tweedie distribution

depending on its value, as shown in Table 3.1. By changing this relation, it alters the distribution to represent other types of distributions. Jørgensen [31] noted that $\rho = 0$ represented a Gaussian distribution, $\rho = 1$ a Poisson distribution, $\rho = 2$ a Gamma distribution, etc. Most importantly, he showed that when $1 < \rho < 2$, the resulting distribution is a compound Poisson-Gamma distribution, which is a sum of a Poisson distributed number of samples from Gamma distributions.

A sample obtained from a compound Poisson-Gamma distribution can thus be decomposed as c in Equation 3.8:

$$c = \begin{cases} 0 & \text{if } N = 0 \\ \tilde{c}_1 + \tilde{c}_2 + ... + \tilde{c}_N & \text{if } N = 1, 2, \dots \ , \end{cases} \tag{3.8}$$

where N is Poisson distributed and $\tilde{c}_1 + \tilde{c}_2 + ...$ are iid Gamma random variables independent of N.

Yang et al. [67] noted that the probability density functions of the distributions found when $1 < \rho < 2$ can effectively be written as the sum of the products of Poisson and Gamma distributions with an additional mass at 0:

$$P(y|\lambda, \alpha, \gamma) = e^{-\lambda} d_0(y) + \sum_{j=1}^{\infty} \frac{\lambda^j e^{-\lambda}}{j!} \frac{\lambda^{j\alpha} y^{j\alpha-1} e^{-y/\gamma}}{\gamma^{j\alpha} \Gamma(j\alpha)}, \tag{3.9}$$

where d_0 is the Dirac delta function at zero, j is a value of $N \sim \text{Poisson}(\lambda)$ and that the conditional density of c on $N = j$ is $\text{Gamma}(j\alpha, \gamma)$.

As the compound Poisson-Gamma distribution is the most common use of the Tweedie distribution in statistical models, the latter usually refers to the former, as is the case in this work.

In the context of pure premium estimation, Equation 3.8 represents the total losses c under a specific year as the sum of N samples $\tilde{c}_d$ of N independent Gamma distributions. In this case, N is sampled from a Poisson distribution with a mean and variance λ, while $\tilde{c}_d$ is the amount sampled from the d^{th} Gamma distribution, which each have their own shape parameter α and scale parameter γ. This can also be shown by decomposing the Tweedie mean and variance into the Poisson and Gamma parameters. As shown in Table 3.1, the mean and variance of the Tweedie distribution is similar to the product of the mean and variance of the Poisson and Gamma distribution. As both separate components of the pure premium can be combined under the Tweedie distribution, the pure premium can represented under a single component, as shown in Equation 3.10.

$$y = \text{Pure premium} = \text{Average claimed amount per unit of risk} \tag{3.10}$$

As with Poisson GLMs, $\hat{y}$ can be obtained from the Tweedie GLM by using an offset for units of risk and setting $w = 1$ when the pure premium $\hat{y}$ needs to be estimated.

Having equivalent computed outcomes $\hat{y}$ obtained by both formulations in equations 3.4 and 3.10 that result in the pure premium for a certain policy means that either using one or two models should theoretically have the same results. Nonetheless, Tweedie models can be advantageous if frequency and severity are highly correlated, which Poisson and Gamma models assume not to be the case [22]. Also, in cases of where too few observations with claims are available or that a model must be built as quickly as possible, having a single model with pure premiums as a unique target variable can still efficiently produce accurate results.

3.1.3 Metrics for Customer Profitability Estimation

As discussed previously, it is essential for models to be assessed before being put into a production setting. The metrics presented in section 1.1.2, while being applicable to most classification and regression problems, do not paint the whole picture of the model's capacity to solve the specific task at hand. For the task of pure premium estimation, three types of criteria must be evaluated in order to validate and compare models: its accuracy on present and future data, its monotonicity, and its ability to differentiate low and high risk policies [22].

3.1.3.1 Accuracy Metrics

In order to validate that a model is accurate, two types of metrics must be computed. One ensures that the model is accurate on all individual policies, while the other ensures that the model behaves properly on the entire population.

Depending on the distribution of outcomes and the maximum capped value of losses, model accuracy on individual observations can be done using metrics such as the RMSE (1.19), the MAE (1.5) or the R-squared (1.18). The first two can give the average dollar amounts that separate the estimated pure premium and the claimed losses. While their weighting of the errors differs and can vary depending on the prioritised observations, the goal is to minimise the selected one. The latter metric must be maximised to explain the most variance in outcomes. These metrics do give an idea of the extent to which an algorithm is accurate. However, as the variance of claimed amounts is disproportionately high, high discrepancies bear a huge weight on the average loss. For example, consider a typical claim amount distribution with 100,000 clients and a mean of 1,000\$. In this case, 98,000 claim nothing and 2,000 clients have a single claim of 50,000\$. If the mean is always predicted and the RMSE is used, the 2,000 claims will contribute to 98% of the total RMSE of 7,000. As such, the RMSE is naturally high because of a low mean and a high variance, but is not necessarily indicative of a bad model if other metrics and comparative models are taken into consideration. MAE can be used to reduce the effect of outliers, but the scale of the errors still highly varies depending on the claim distribution.

As a result, other metrics and visualisations specific to insurance rating must be used. One of the most popular is the use of the loss ratio (3.6) and loss ratio charts, as will be defined later. As presented in Equation 3.6, the loss ratio for a population is the ratio between the sum of its claimed amounts and the sum of its estimated pure premiums. The resulting metric can be used as a validation tool to ensure company profitability. For example, if the sum of a model's estimated pure premiums is less than the sum of the claimed losses during a full calendar year, an insurance company would not be profitable and could go out of business. If the model overestimates the pure premiums on average, the insurer might charge too much to insure clients and thus lose profitable clients to competing companies. As a result, accuracy metrics must be accompanied by the overall loss ratio of the model to ensure that all the sum of the estimated pure premiums are closely related to the sum of the claimed losses for the full calendar year. Loss ratios can either be used on a case by case basis, on an entire population, or by bins of clients that are sorted by pure premium. This latter one allows an insurer to study the distribution of its rates and how the rates are over or underestimated on average depending on a certain level of risk. This can easily be visualised in what is called a **loss ratio chart**. Loss ratio charts, as illustrated in Figure 3.2, plot the average loss ratio of each decile of observations Q with similar estimated pure premiums contained in $\hat{\boldsymbol{Y}}$, which are the estimated pure premiums attributed by the model $f(\boldsymbol{x})$ [22]. By sorting the observations by their estimated pure premiums, putting equal numbers of observations with similar estimated pure premiums into buckets (i.e. deciles) and then computing Equation 3.11 for each bucket Q, it is possible to determine if all segments of a population are being given an actuarially

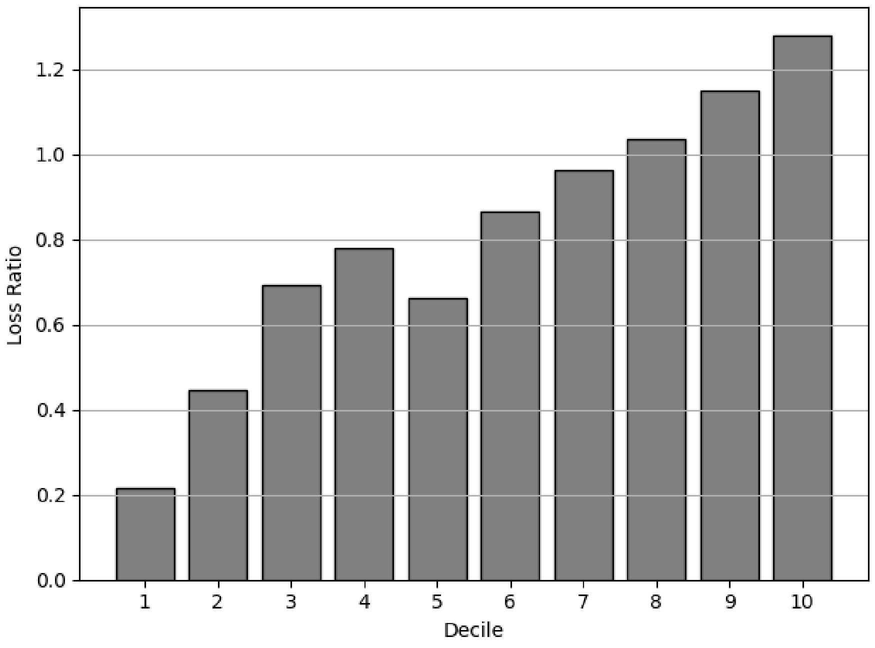


Figure 3.2 – Example of a loss ratio chart

fair estimated pure premium if they have loss ratios as close as possible to 1. It is important to note that Figure 3.2 is an example of a poorly performing pure premium estimation model, as the first seven deciles of estimated pure premiums are greatly overestimated, while the last two are underestimated.

$$\text{Loss Ratio for decile } Q \text{ of } \hat{\boldsymbol{Y}} = \frac{\sum\limits_{(\boldsymbol{x},y)\in Q} \text{Claimed amount}}{\sum\limits_{(\boldsymbol{x},y)\in Q} \text{Predicted claim amount}} = \frac{\sum\limits_{(\boldsymbol{x},y)\in Q} y}{\sum\limits_{(\boldsymbol{x},y)\in Q} f(\boldsymbol{x})} \tag{3.11}$$

3.1.3.2 Monotonicity and Lift Metrics

According to Goldburd [22], the monotonicity criterion is there to ensure that the estimated pure premiums increase monotonically as the true amount of pure premium increases. A model, albeit not being necessarily exact, must preferably always attribute higher pure premiums to observations with a true higher risk and lower pure premiums to observations with a true lower risk. As in the loss ratio chart, the set of observations is split into sorted decile of estimated pure premiums Q with similar estimated pure premiums. Each decile of observations is then compared by their average observed losses and average estimated pure premium, where the former should preferably always increase. Reversals, where the estimated pure premiums increase and the observed losses decrease, must be kept to a minimum.

An intuitive way to plot the relationship between the average claimed amount and the average estimated pure premium is to use **simple decile plots**. Simple decile plots graph two lines for

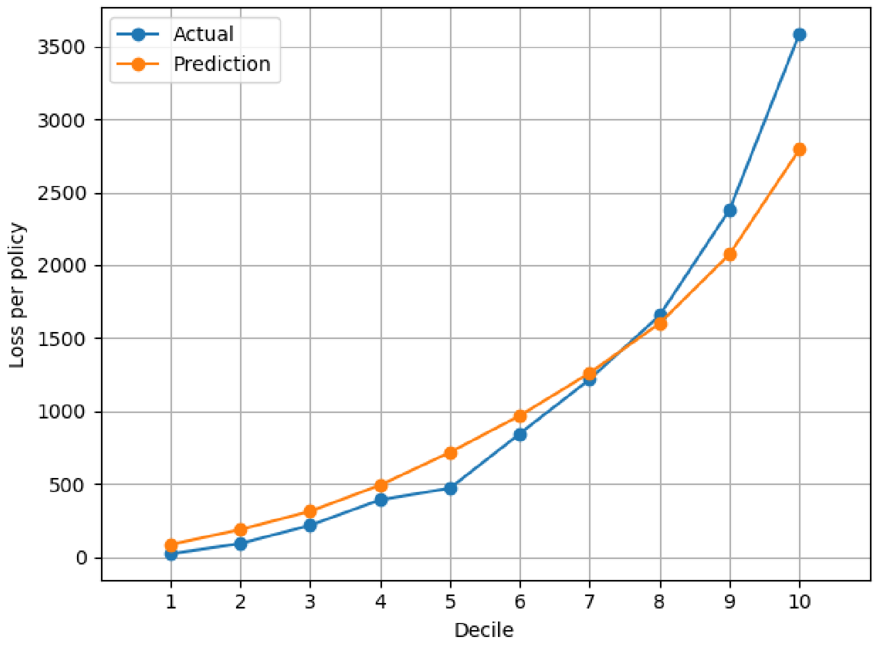


FIGURE 3.3 – Example of a decile plot

each decile of estimated pure premiums: the average claimed amount per observation and the average estimated pure premium per observation. As seen in Figure 3.3, the average claimed amount should always increase per decile and should never be lower than the previous one. Another use of simple decile plots is the visualisation of a model's capability of differentiating high risk observations and low risks observations (i.e. a model's **lift**). As discussed in Section 2.3, if similar risk levels get similar risk ratings and dissimilar risks get dissimilar risk ratings, then the overall rates are considered actuarially fair. Simple decile plots can be used to determine model lift by using the vertical distance between the actual average losses of the first and last deciles. For example, in Figure 3.3, the spread between the worst risks and the best risks is from 0$ to above 3,500$. Thus, when comparing pure premium estimation models on the same dataset, the model with the larger spread is better at differentiating observations with a true low risk level from observations with a true high risk level.

3.2 Achieving Fair Risk Assessment Models

The creation of a fair risk assessment model goes through the answering of many questions and concerns from all the concerned domains. Are there restrictions to the types of objectives that can be used? What types of models can be used? What fairness criterion attempts to satisfy both actuarial fairness and group fairness? How should the model be constrained to satisfy that criterion? How should sensitive attributes be used? The next sections proposes a framework in which all those concerns are addressed in order to put out a robust and efficient solution.

3.2.1 Selecting Model Form

A primary concern for solving the task of pure premium estimation using statistical models is to select how the final prediction is built. This can either be through the use of frequency and severity models (3.4) or pure premium estimation models (3.10). Although many concerns about efficiency and interpretability were raised previously, the addition of any fairness criterion constrains the forms that the pure premium estimation can take. All of the fairness criteria presented in Chapter 2 can only be used with the assumption that the final outcome is based on solving a single task. The issue that comes up when a prediction is based on many sub-tasks is the one of **compositional fairness**. Much research has been done on the fact that the conjunction of many fair binary classifiers with AND or OR conditions does not necessarily output fair final outcomes [14, 63]. Even though solutions were proposed to deal with multiple-task fairness for classifiers in previous research, none of them address how to adjust regression models to deal with compositionality. As such, to ensure that the rating model is fair, using a single pure premium estimation model is a simpler approach to the task.

3.2.1.1 TDBoost: Tweedie GBDT Models

Although most statistical models used for pure premium estimation are GLMs, it is important to know why more complex models are not often used for this task. As regulators ask that risk ratings must be fully explained by the coefficients and decision rules for each risk factor used [49], the use of interpretable models is of the utmost importance for compliance. Also, actuarial departments may use pure premium models for other applications where interpretability is of higher importance than model accuracy [22]. As a result, more complex models, such as GBDTs, have rarely been applied to this task.

However, even though they can be flexible and interpretable, GLMs have many disadvantages. Notably, as explained in Section 1.2, the target labels used on a GLM must be transformed to represent a linear relationship between $\boldsymbol{x}$ and $\mu(\boldsymbol{x})$. This relationship might not be linear at all. Even if some advanced transformations of attributes can be used to achieve linearity, the relation must be known and set beforehand. They are also sensitive to the selected features, as they try to model the relation throughout all attributes. Uninformative and correlated risk factors thus add noise, increase error rates and decrease model explainability.

As shown in Section 1.3.6.2, much research has been done recently to increase the explainability of GBDT models, which can model non-linear relationships between observations and outcomes. As a result, the use of more complex models in regulated domains may not be as out of reach as it once was. In fact, many new models specialised in the estimation of pure premiums could be used in cases where they could not beforehand, eliminating the dilemma of choosing between high accuracy and high interpretability. Amongst those recent models is TDBoost [67], which is the use of GBDT for Tweedie-distributed labels.

Like GLMs, gradient boosted decision trees have the flexibility of using loss functions that are differentiated from the log-likelihood of the label distribution. As a result, similar to Tweedie GLMs (3.1.2.2), TDBoost uses the log-likelihood of the Tweedie distribution (3.13) and its differentiation. The difference is found in the fitted parameters, where GBDTs are composed of boosting iterations and GLMs are composed of a single set of coefficients.

Also, as TDBoost can be used with advanced gradient boosting methods, the gradients g_i (3.15) and second derivatives h_i (3.16) of the log-likelihood (3.13) must be obtained at each boosting iteration t to fit each weak learner $h_t(\boldsymbol{x})$, which results in the GBDT model $\hat{y}_i = F(\boldsymbol{x}) = \sum_{t=1}^{k} \lambda \cdot h_t(\boldsymbol{x})$, where λ is the learning rate of the GBDT. To adapt GBDTs to use the Tweedie log-likelihood function, the likelihood function of a single observation is designated as $P(y_i|\boldsymbol{\theta}) = P(y_i|F(\boldsymbol{x}_i), \phi, \rho, w_i)$ and the log-likelihood for the set $\{\boldsymbol{X}, \boldsymbol{Y}\} = \boldsymbol{S}$ is designated as $\ell(\boldsymbol{Y}|\boldsymbol{\theta}) = \ell(\boldsymbol{Y}|F, \phi, \rho, \boldsymbol{S})$. The Tweedie log-likelihood function's parameters include the dispersion parameter ϕ and the power parameter ρ. The Tweedie log-likelihood function's second term is the normalizing function $a(\cdot)$, which returns a constant. Hence, the development and understanding of this function is not required for this work.

While equations 3.12, 3.13 and 3.15 can be found in the original paper by Yang et al. [67], Equation 3.16 can be obtained from the LightGBM implementation of TDBoost [11].

Many crucial elements must be taken from equations 3.12 to 3.16. First, the Tweedie likelihood function (3.9) is equivalent to the Tweedie likelihood function in Equation 3.12, as pointed out by Smyth [54]. Second, it is important to note the presence of the units of risk for the observation i (w_i) inside the computation of the likelihood, the log-likelihood, the gradient $g_i^{(t)}$ (3.15) and the second derivative $h_i^{(t)}$ (3.16) at iteration t. As y_i is the claimed amount for an entire calendar year, each original outcome c_i (i.e. the claimed amount during a certain calendar year) must be normalised by w_i, as shown in Equation 3.14. This is well managed in GLMs, which use an offset. However, to obtain the appropriate loss function without an offset, the gradient and second derivative of each observation must be multiplied by w_i to obtain a proportional loss for each observation. Third, the gradient and second derivative do not include the dispersion parameter ϕ, nor do they include the normalizing function $a(\cdot)$, which are not dependent on the pure premium estimation GBDT model $F(\boldsymbol{x})$.

$$P(y_i|F(\boldsymbol{x}_i), \phi, \rho, w_i) = \exp\left(\frac{w_i}{\phi}\left(y_i \frac{\exp\left(F(\boldsymbol{x}_i)(1-\rho)\right)}{1-\rho} - \frac{\exp\left(F(\boldsymbol{x}_i)(2-\rho)\right)}{2-\rho}\right)\right) \times a\left(y_i, \frac{\phi}{w_i}, \rho\right) \tag{3.12}$$

$$\begin{aligned}\ell(\boldsymbol{Y}|F,\phi,\rho,\boldsymbol{S}) &= \sum_{i=1}^{n} \log(P(y_i|F(\boldsymbol{x}_i),\phi,\rho,w_i)) \\ &= \sum_{i=1}^{n} \frac{w_i}{\phi}\left(y_i \frac{\exp\left(F(\boldsymbol{x}_i)(1-\rho)\right)}{1-\rho} - \frac{\exp\left(F(\boldsymbol{x}_i)(2-\rho)\right)}{2-\rho}\right) + \\ &\qquad \log\left(a\left(y_i, \frac{\phi}{w_i}, \rho\right)\right)\end{aligned} \tag{3.13}$$

Setting $y_i = \frac{c_i}{w_i}$

$$\begin{aligned} &= \sum_{i=1}^{n} \frac{w_i}{\phi}\left(\frac{c_i}{w_i} \frac{\exp\left(F(\boldsymbol{x}_i)(1-\rho)\right)}{1-\rho} - \frac{\exp\left(F(\boldsymbol{x}_i)(2-\rho)\right)}{2-\rho}\right) + \\ &\qquad \log\left(a\left(\frac{c_i}{w_i}, \frac{\phi}{w_i}, \rho\right)\right)\end{aligned} \tag{3.14}$$

$$g_i^{(t)} = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} \sum_{i=1}^{n} \log(P(y_i|\hat{y}_i^{(t-1)},\phi,\rho,w_i))$$

Using Equation 3.13

$$g_i^{(t)} = \frac{\partial}{\partial \hat{y}^{(t-1)}} \sum_{i=1}^{n} \frac{w_i}{\phi}\left(y_i \frac{\exp\left(\hat{y}_i^{(t-1)}(1-\rho)\right)}{1-\rho} - \frac{\exp\left(\hat{y}_i^{(t-1)}(2-\rho)\right)}{2-\rho}\right) + \log\left(a\left(y_i, \frac{\phi}{w_i}, \rho\right)\right)$$

Dropping the second term, which is not dependent on $\hat{y}_i^{(t-1)}$

$$g_i^{(t)} = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} \sum_{i=1}^{n} \frac{w_i}{\phi}\left(y_i \frac{\exp\left(\hat{y}_i^{(t-1)}(1-\rho)\right)}{1-\rho} - \frac{\exp\left(\hat{y}_i^{(t-1)}(2-\rho)\right)}{2-\rho}\right)$$

Removing ϕ, which is a constant not dependent on the observation i

$$g_i^{(t)} = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} \sum_{i=1}^{n} w_i\left(y_i \frac{\exp\left(\hat{y}_i^{(t-1)}(1-\rho)\right)}{1-\rho} - \frac{\exp\left(\hat{y}_i^{(t-1)}(2-\rho)\right)}{2-\rho}\right)$$

Differentiating the equation for $\hat{y}_i^{(t-1)}$

$$g_i^{(t)} = w_i\left\{y_i \exp\left[(1-\rho)\cdot \hat{y}_i^{(t-1)}\right] - \exp\left[(2-\rho)\cdot \hat{y}_i^{(t-1)}\right]\right\} \tag{3.15}$$

$$h_i^{(t)} = \frac{\partial^2}{\partial^2 \hat{y}_i^{(t-1)}} \log(P(y_i|\hat{y}_i^{(t-1)},\phi,\rho,w_i)) = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} \frac{\partial}{\partial \hat{y}_i^{(t-1)}} \log(P(y_i|\hat{y}_i^{(t-1)},\phi,\rho,w_i))$$

Using Equation 3.15

$$h_i^{(t)} = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} w_i\left\{y_i \exp\left[(1-\rho)\cdot \hat{y}_i^{(t-1)}\right] - \exp\left[(2-\rho)\cdot \hat{y}_i^{(t-1)}\right]\right\}$$

$$h_i^{(t)} = w_i\left\{y_i(1-\rho)\exp\left[(1-\rho)\cdot \hat{y}_i^{(t-1)}\right] - (2-\rho)\exp\left[(2-\rho)\cdot \hat{y}_i^{(t-1)}\right]\right\} \tag{3.16}$$

While GBDTs aren't usually used in practice for pure premium estimation models, using the TreeSHAP algorithm on TDBoost makes them a practical choice for increased accuracy while keeping an appropriate level of interpretability.

3.2.2 Selecting Fairness Criteria

Attempting to satisfy actuarial fairness and group fairness at the same time restricts the type of fairness criteria that can be used to quantify the fairness of estimated pure premiums. As actuarial fairness mostly relates to model accuracy in regression problems, any selected fairness criterion to quantify the fairness of estimated pure premium must relate to error parity, as defined in Section 2.1.1.4. As such, if actuarial fairness were to be related to a group fairness criterion, an algorithm would then be considered more fair if the provided estimated pure premiums have a similar level of accuracy across all subgroups of individuals. The resulting fairness criterion is expressed in Equation 3.17:

$$\frac{\max\limits_{a \in \boldsymbol{A}} (CVaR_\alpha(Z_a))}{\min\limits_{a \in \boldsymbol{A}} (CVaR_\alpha(Z_a))} \leq 1.2, \tag{3.17}$$

where $Z_a = \{L(y, F(\boldsymbol{x}) : (\boldsymbol{x}, y) \in \boldsymbol{S}_a)\}$.

Equation 3.17 considers an algorithm more fair if all distributions of loss Z_a related to a subgroup have similar conditional values-at-risk (2.26). The criterion considers that a model is fair if the subgroup with the worst mean error over the percentile α is at most 20% worse than the subgroup with the best mean error over the percentile α, as per the fourth-fifth rule. This means that the most inaccurate α% premiums for each group will all be similarly inaccurate, up to an error parity of 20%. Since it is difficult to reach exact group fairness on outcomes that can vary greatly, an approximate fairness criterion is better suited to quantify fairness for pure premium prediction models, as it allows a greater flexibility in the choice of the solution.

In practice, this criterion does not restrict the model when optimising the model, but it evaluates the disparity of $CVaR$ between subgroups so that models can be properly assessed for fairness afterwards. By using the ratio between the $CVaR$ of the group with the maximum $CVaR$ and the $CVaR$ of the group with the minimum $CVaR$, no group can be left behind when assessing the model. If the criterion is not respected, actuaries and machine learning practitioners are left with two choices: either try to increase the accuracy on the group with the worse average group error or decrease the accuracy on the best average group error, which can be done with the use of a fairness constraint. The use of the ratio also gives much more flexibility, as using the difference between the best and worst group error requires the use of an arbitrary upper bound depending on the problem on which it is used. It also enables the use of the criteria on more than two groups at a time, as any other group that has an average error in between both the upper and lower bound is treated more fairly than the group with the worst error.

Since most people do not claim anything during a year, the errors for rating models are often highly skewed. By using Equation 3.17, the model is allowed to ignore small errors in order to focus on the worst-case errors of each subgroup. As a result, the fairness aspect of the metric will mostly ensure that high risk individuals in each group, which usually have the biggest prediction errors, have similarly accurate ratings on average. All in all, this criterion allows for fair and accurate models by ensuring that all subgroups have pure premiums with similar worst-case errors.

As an example of the application of this criterion, consider a rating model with a Tweedie loss function that has the marital status as a sensitive variable. Consider that there are three possible subgroups: single, married and divorced, and that we want to be fair on 80% of the worst losses of each group (i.e. $1 - \alpha = 0.8$ and $\alpha = 0.2$). First, the loss is computed on all individuals in a target population $\boldsymbol{S}$. Second, the value-at-risk at $\alpha = 0.2$ is computed on the distribution of the losses Z_a for each group. In this case, suppose that the 20^{th} percentile of errors is 200 for single individuals, 220 for married individuals and 240 for divorced individuals. Third, the CVaR at $\alpha = 0.2$, or the average loss over the 20^{th} percentile, is computed. In this example, the average loss over 200 for single individuals would be 300 the average loss over 220 for married individuals would be 350 and the average loss over 240 for married individuals would be 360. As a result, the group of single individuals would be the most accurately assessed, while the divorced ones would have more inaccurate ratings. However, the error ratio of $\frac{360}{300} = 1.2$ separating them would not be considered disproportionate enough to consider it disparate treatment. As such, the criterion would be satisfied.

3.2.3 Selecting Fairness Constraint

As indicated in Chapter 2, Williamson et al. introduced the use of CVaR objective (2.27) to increase model fairness. However, it has many restrictions in the variety of its usages. First, it is usually used as an objective that must be minimised while always keeping its initial form. As noted in most papers optimising for CVaR [52], the objective is optimised using linear programming. As such, it cannot be integrated directly into GBDT models. Second, it does not take intersectionality into consideration: it can only be used on a single attribute at a time.

To deal with the first restriction, we can integrate the CVaR objective to a GBDT with a Tweedie loss function by obtaining its gradient and its second derivative with regards to $f(\boldsymbol{x})$, as shown in the steps provided in equations 3.18 through 3.22. Note that since the target model here is a GBDT, the ensemble of regression trees $F(\boldsymbol{x})$ is equal to the function $f(\boldsymbol{x})$, which is equal to the predicted outcome $\hat{y}$.

By knowing the gradient (3.15) and second derivative (3.16) of the Tweedie log-likelihood function, we can substitute them into the gradient (3.20) and second derivative (3.22) of the CVaR objective, making the objective usable with advanced gradient boosting methods. The result is simple and intuitive, while still keeping the essence of using the CVaR. As in Equation

2.27, the model optimises the CVaR at each boosting iteration by ignoring groups with a small average loss under ξ, while increasing the gradient and second derivative of subgroups with average losses over ξ by a factor of $\frac{1}{1-\alpha}$. As a result, the next boosting iteration will focus more on subgroups that are not yet properly assessed, as their importance will have been increased.

In addition, as noted in Equation 2.28, the parameter ξ must also be optimised to minimise the empirical risk. However, since GBDTs optimise the objective with regards to $f(\boldsymbol{x})$, this additional parameter is initialised at 0 and is updated separately via gradient descent after each boosting iteration. The gradient of $L_S(f, \xi)$ with regards to ξ, as determined by Rockafellar et al. [52], can be used to do so. To adapt this result to Williamson's formulation of CVaR for group fairness (2.27), the gradient is based on the average group probability of having a loss over ξ instead of the global probability of having a loss over ξ, as shown in Equation 3.23.

α controls how much should the model take into consideration the groups with higher average losses. In the end, this effectively means that a GBDT can optimise a CVaR using equations 3.20 and 3.22 by reducing the empirical risk of groups of observations with an average loss over ξ with an extreme value importance parameter α.

$$\frac{\partial}{\partial f(\boldsymbol{x})} L_S(f,\xi) = \frac{\partial}{\partial f(\boldsymbol{x})}\xi + \frac{1}{n\cdot(1-\alpha)}\frac{\partial}{\partial f(\boldsymbol{x})}\sum_{a\in\boldsymbol{A}}\left[\frac{1}{m_a}\sum_{(\boldsymbol{x},y)\in\boldsymbol{S}_a} L(y,f(\boldsymbol{x})) - \xi\right]_+ \tag{3.18}$$

Removing the first term since it is not dependent on $f(\boldsymbol{x})$.

$$\frac{\partial}{\partial f(\boldsymbol{x})} L_S(f,\xi) = \frac{1}{n\cdot(1-\alpha)}\frac{\partial}{\partial f(\boldsymbol{x})}\sum_{a\in\boldsymbol{A}}\left[\frac{1}{m_a}\sum_{(\boldsymbol{x},y)\in\boldsymbol{S}_a} L(y,f(\boldsymbol{x})) - \xi\right]_+ \tag{3.19}$$

Keeping the parts of the equation related to the group a of observation $\boldsymbol{x}$.

$$\frac{\partial}{\partial f(\boldsymbol{x})} L_S(f,\xi) = \begin{cases} \frac{\frac{\partial}{\partial f(\boldsymbol{x})}L(y,f(\boldsymbol{x}))}{m_a\cdot n\cdot(1-\alpha)} & \text{if } \left((\boldsymbol{x},y)\in\boldsymbol{S}_a : \frac{1}{m_a}\sum\limits_{(\boldsymbol{x},y)\in\boldsymbol{S}_a} L(y,f(\boldsymbol{x})) > \xi\right) \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

$$\frac{\partial^2}{\partial^2 f(\boldsymbol{x})} L_S(f,\xi) = \frac{\partial}{\partial f(\boldsymbol{x})}\frac{\partial}{\partial f(\boldsymbol{x})} L_S(f,\xi) \tag{3.21}$$

Using Equation 3.20

$$\frac{\partial^2}{\partial^2 f(\boldsymbol{x})} L_S(f,\xi) = \begin{cases} \frac{\frac{\partial^2}{\partial^2 f(\boldsymbol{x})}L(y,f(\boldsymbol{x}))}{m_a\cdot n\cdot(1-\alpha)} & \text{if } \left((\boldsymbol{x},y)\in\boldsymbol{S}_a : \frac{1}{m_a}\sum\limits_{(\boldsymbol{x},y)\in\boldsymbol{S}_a} L(y,f(\boldsymbol{x})) > \xi\right) \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

$$\begin{aligned} \frac{\partial}{\partial \xi} L_S(f,\xi) =& 1 + \frac{1}{n\cdot(1-\alpha)}\sum_{a\in\boldsymbol{A}} -1\cdot I\left(\frac{1}{m_a}\sum_{(\boldsymbol{x},y)\in\boldsymbol{S}_a} L(y,f(\boldsymbol{x})) > \xi\right), \\ &\text{where } I(q) = \begin{cases} 1 & \text{if } q = \text{True} \\ 0 & \text{if } q = \text{False} \end{cases} \end{aligned} \tag{3.23}$$

$\boldsymbol{A} =$

Gender	
Men	Women

Age Group	
Young	Old

$\Longrightarrow$

Gender×Age Group			
Young men	Old men	Young women	Old women

Table 3.2 – Example of sensitive attribute transformation

All in all, the $CVaR$ constraint can be adapted and combined to GBDTs to create fair pure premium prediction models.

3.2.4 Selecting Sensitive Attributes

The second restriction of the original CVaR fairness constraint is that it does not take into consideration the use of multiple sensitive attributes. Although it has been adapted to work on an unlimited number of groups, on categorical and continuous sensitive attributes, the case for quantifying and optimising for fairness amongst the intersection of multiple sensitive attributes using the CVaR has yet to be explored. It is also important that a model stays fair amongst the intersections of subgroups and the independent subgroups. For example, a model that is fair towards young men and women in addition to old men and women should also be fair towards men, women, young and old people independently.

In order to deal with intersectionality, the CVaR constraint and the sensitive attributes can be adapted to ensure fairness throughout all subgroups and all of their possible intersections. The adaptation mostly relies on transforming the intersections of attributes into a single categorical attribute, where each attribute value represents a combination of the protected features. As a result, the number of possible values for this single feature is the product of unique values for each separate sensitive feature. Table 3.2 is an example of how the two attributes "Age Group" and "Gender" are converted into a single attribute $\boldsymbol{A}$ which represents the combination of both features.

This transformation can work in the context of using the feature for ensuring fairness with the fairness criteria defined in Section 3.2.2. As noted by Foulds et al. [18], splitting a population into smaller subsets can help in detecting unfairness in what would seem to be a fair model when using a single sensitive attribute at a time. However, if these smaller subgroups are merged back together, the model will never become more unfair than it was on the smaller groups.

In the context of the proposed CVaR ratio criteria, the use of the maximum and minimum subgroup error amongst all the encoded intersections of individual subgroups ensures that the model is fair for all sensitive attribute intersection and for all individual sensitive attributes, as the gap between the maximum and minimum error can only be reduced when merging groups together.

The biggest caveat to the proposed method is that it only supports categorical attributes as a means to intersect different attributes into discrete subgroups, which limits the variety of attributes it can manage out of the box. To manage any sensitive attribute which is either too numerous or continuous, such as age, its values must be binned by putting similar values together. This can only be done if there is a relation of similarity between the different attribute values. For example, age can be binned by decade, while marital status cannot be binned at all.

After selecting the encoding methodology, the selection of the sensitive attributes can begin. It is important to define which attributes are selected and why they should be selected. Firstly, it should be important to determine which attributes are the most susceptible to be affected negatively for the specific task at hand. This can be from known historical biases. Second, the selected attributes should be relevant to anti-discrimination laws. Finally, the selected attributes must be available in the data set that is used on an individual basis, to ensure that fairness is applied properly on real subgroups. This last point is the most important, as using proxy attributes or trying to predict the sensitive attribute often leads to unfairness when errors are made. For example, Bayesian Improved Surname Geocoding (BISG) is commonly used to predict a person's race from its geolocation and surname. Since this statistical model always makes mistakes, it has been shown to overestimate pricing disparities between subgroups when auditing discrimination in lending applications and their outcomes [70]. As such, the use of true subgroup labels should always be preferable to ensure that the mitigating and auditing of bias is done properly.

Altogether, the laying out of the task of customer profitability estimation and the tools which have been established in this chapter allows the creation of intersectionnaly fair and accurate models that estimates each client's pure premium.

Chapter 4

Experimental Results

In this final chapter, we put together notions from the domains of machine learning, actuarial science with a focus on model fairness. Altogether, we provide experimental results showing that it is possible to create fair, accurate and interpretable models that are able to estimate customer profitability. First, we lay out the steps taken to obtain the final model, which applies the CVaR objective (2.27) to GBDTs. This includes selecting the proper risk factors, the appropriate training methodology and the right hyperparameter values. Then, we evaluate the accuracy and fairness of the model according to the metrics presented in sections 3.1.3 and 3.2.2. Finally, we compare these results to traditional GLMs to determine how GBDTs could be applied to provide more accurate and fair pure premiums while sacrificing minimal interpretability.

4.0.1 Experimental Data Set

In order to assess our fair Tweedie GBDT model, we analyse its behaviour on a large home insurance claim data set extracted and provided by a major Canadian insurer. The training set contains 1,935,876 rows collected between January 2012 and December 2016 that each represent a single home assigned to a policy during a specific calendar year, along with information about the policy holder, the previous claims made by the policy holder and the home itself, in a way that is similar to what is described in Section 3.1.1. Additional socio-demographic data related to the client's geolocation is also joined on each row. In total, 205 discrete or continuous risk factors were identified to influence a client's pure premium, as detailed in Table 4.1. However, only the factors that were included in the top 20 most important factors by one of the presented models in Figure 4.3 are shown here for brevity. Also, due to the confidential nature of the dataset, the other attributes of the dataset cannot be shown. Note that all data was anonymised prior to use so that no individual can be identified using the dataset. In order to reference the attributes of the dataset of Table 4.1 in the next sections and graphs, the abbreviated variable names indicated in italic between the parentheses next to the description of each variable will be used. For example, the payment method used by

Policy holder data	**Home attributes**	**Policy attributes**	**Socio-demographic data (by postal code)**
Age *(age)*	Basement finishing *(bas_ finish)*	Total value of covered assets (in 2016 dollars) *(cov_ amount_ idx)*	Adults with income (%) *(%_ ad_ income)*
Gender *(gender)*	Age of home *(home_ age)*	Total value of covered assets ($) *(cov_ amount)*	Adults with revenue over 125,000$ (%) *(%_ rev_ over_ 125)*
Credit score *(credit)*	Number of plumbing elements *(num_ plum_ elem)*	Policy type (tenant or homeowners insurance) *(pol_ type)*	Participation rate (%) *(%_ ad_ part_ rate)*
Payment method *(pay_ mode)*	Number of wood cords consumed per year for heating *(num_ wood_ cord)*	Flooding insurance coverage amount ($) *(cov_ amount_flo)*	Employment rate (%) *(%_ ad_ emp_ rate)*
Number of years where client had home insurance *(years_ insured)*	FSA of home *(fsa)*	Flooding insurance coverage amount (in 2016 dollars) *(cov_ amount_flo_ idx)*	Families with 20 to 24 children (%) *(%_fam_ 20_ 24_ ch)*
	Age of roof *(roof_ age)*	Presence of a lender *(has_ lender)*	Adults working at usual place (%) *(%_ work_ usu_ pl)*
	Roof covering material *(roof_ cov_ mat)*	Calendar year of policy *(year)*	Adults working in sales or service industry (%) *(%_ occ_ sale_ serv)*

Table 4.1 – Sample of risk factors used for pure premium estimation

the client will be described by the variable *pay_ mode*.

In total, 33,232 rows have claims, thus having a claim amount above zero. As these rows only make 1.71% of the total data set, they are a rare occurrence, but have an immense weight on the average claimed amount. As a result, a client claimed an amount 324$ per unit of risk on average, while a client that made at least a claim claimed 14,308$ on average.

To properly evaluate the performance of a model, an extra data set was extracted for test purposes. Thus, it can only be used as a final validation. It contains 184,388 rows that were collected between January 1st 2017 and December 31st 2017. In this case, clients claimed 479$ per unit of risk on average and the 4,116 rows with a non-zero claim amount contained an average amount of 21,460$. As assets often increase in price each year, the cost to repair or replace them also increases. Thus, claim amounts trend upwards over time [30]. This increase in the average claimed amount must then be taken into account when training a new model to estimate pure premiums, either by using the policy's calendar year as a risk factor, by trending the future pure premiums by a separately estimated coefficient that can be obtained with a simple linear regression or by expressing all amounts into deflated dollars (e.g. 2012 dollars). For this assessment, we used the former method to estimate that trend.

4.1 Training Methodology

4.1.1 Data Preprocessing and Data Engineering

Before using a dataset on our fair Tweedie GBDT model, the data needs to be transformed in a way that makes it usable and that makes it easier for the GBDT to detect the proper risk factors in a way that can generalise to future observations. Notably, the attributes contained in socio-demographic data at the postal code level are often bins in which the individuals in that postal code are put in if that bin corresponds to their label. For example, the attributes *%_ad_emp_rate* and *%_ad_part_rate* are originally bins containing the number of individuals who are employed or participating in the labour market in a certain postal code. As these attributes are in absolute terms, they must be normalised to compare each postal code relative to each other. As a result, *%_ad_income*, *%_ad_emp_rate* and *%_ad_part_rate* are the number of individuals in a certain bin divided by the total number of adults in the postal code and are the percentages of the total adult population in the postal code with a specific label.

As for the categorical attributes, LightGBM can manage them as a single column by finding the best split that can partition the categories into two subsets, as per Fisher [17]. When computing the optimal split for a categorical attribute, each observation is assigned to a bin based on their category, along with their current gradient and second derivative. These bins are then sorted based on the sum of the bin's gradients divided by the sum of its second derivatives. Then, as these unordered categories now have a numerical order of magnitude, the optimal split can be found by using the split function (1.51). As such, there is no need to create dummy variables.

4.1.2 Selected Sensitive Attributes

As mentioned in Section 3.2.4, the selected sensitive attributes that must be used to study and to create fair models must contain the true label of each individual. As such, only the attributes that were already collected at the individual level could be used for the purpose of this model. As a result, the two sensitive attributes that are selected were already available in the data set were the age and the gender of the client.

Since only categorical attributes can be encoded with the methodology proposed in Section 3.2.4, the age of each individual is accordingly put into 6 age bins. The age bins were set to 0 to 25 years old, 25 to 35 years old, 35 to 45 years old, 45 to 55 years old, 55 to 65 years old and 65 years old and older. Each of these bins were intersected with the sex of the policy holder. Altogether a total of 12 groups have to be considered when mitigating and auditing for bias. Table 4.2 shows the distribution of the rows for the test data set of 2017. As some policies have two policy holders (i.e. couples) and cannot be attributed to a single individual, they were filtered from this fairness assessment.

Sex	Age Group	# of rows in group
M	0-25	722
F	0-25	176
M	25-35	3,786
F	25-35	1,941
M	35-45	5,154
F	35-45	4,696
M	45-55	6,225
F	45-55	6,463
M	55-65	5,836
F	55-65	6,354
M	65+	4,009
F	65+	4,209

Table 4.2 – Number of rows per group

Hyperparameter	Values
Power parameter ρ	$\{1.3;1.4;1.5;1.7;1.8\}$
Subsampling	$\{0.25;0.5;0.75;1.0\}$
Column subsampling	$\{0.25;0.5;0.75;1.0\}$
Learning rate	$\{0.001;0.01;0.1;1.0\}$
Number of boosting rounds	$\{50;100;500;1{,}000;5{,}000;10{,}000\}$
Maximum depth	$\{1;2;3;4;5\}$

Table 4.3 – Grid search hyperparameter values

4.1.3 Hyperparameter Selection

As discussed in Section 1.1.2.3, using techniques such as grid search can help in searching the space of the possible hyperparameters to find the best combination for the task at hand. In this case, the subset of selected hyperparameters to tweak were the percentage of the training set that was sampled at the start of each boosting round, the percentage of risk factors that were sampled for each boosted round, the learning rate, the number of boosting rounds and the maximum depth of each decision tree. In addition, the power parameter ρ of the Tweedie log-likelihood (3.13) must be selected, as the negative Tweedie log-likelihood (i.e. the Tweedie log loss) is minimised by the Tweedie models. Each hyperparameter has a set of values, as can be seen in Table 4.3.

Then, each combination of hyperparameter is used for a 5-fold cross validation using the training dataset, to ensure that it works best on average for any set of observations. In the end, the selected hyperparameters were a power parameter of 1.5, a subsampling of 0.5, a subsampling of 50% of the features, a learning rate of 0.001, 10,000 boosting rounds and a maximum depth of 3. As for the tuning parameter α of the CVaR constraint, it was kept at a constant value of 0.90 as to focus on each group's error over the 90^{th} percentile of errors.

4.2 Cost and Effects of Fairness Constraint

After training the model, it can be tested on the test data set of 2017. For the purpose of reviewing the effects of adding a fairness constraint, we trained a model with the same hyperparameters without the CVaR constraint and evaluate the cost on performance and gains made on fairness.

Additionally, we compare the GBDTs with an equivalent GLM with a Tweedie objective provided by the partner insurance company, along with the set of current pure premium estimation models provided by the partner insurance company. The former model is optimised with gradient descent and uses the same power parameter as the GBDTs of 1.5. The latter models estimate a pure premium by computing the product of the estimations of a Gamma and a Poisson GLM for each claim type, which are then summed up to make up the entire pure premium. All of these models were optimised using the same training set defined in Section 4.0.1. The only additional transformations made to the data are made to the categorical attributes. As GLMs do not support the encoding of categorical attributes in a single column, they must all be transformed into dummy variables. As such, we compare the performance of the Tweedie GBDT models with different configurations of linear models to assess the gains that could be made with regards to pure premium estimation accuracy.

To compare the fairness of the Tweedie GBDT with and without constraint, the CVaR ratio (3.17) of the subgroup errors with $\alpha = 0.90$ is used. As for the comparison of the performance of the models, the R-squared, the MAE, the loss ratio along with loss ratio charts and simple decile charts are used. Additionally, the Tweedie GBDTs and the Tweedie GLM will be compared based on their mean Tweedie log loss as a comparative measure of pure premium estimation accuracy, with a power parameter of 1.5. The lower the Tweedie log loss is, the more the model is accurate.

4.2.1 Constraint effect on model fairness

As can be seen in Table 4.5, the addition of a fairness constraint decreases the ratio between the worst and best average group loss over the 90^{th} percentile of losses by 7%. It is important to note that most of the conditional value-at-risks of the grouped Tweedie log loss distributions in the comparison made in Table 4.4 varied greatly for most groups. For 10 out of 12 subgroups, the addition of the CVaR fairness constraint resulted in less accurate estimated pure premiums for the 10% most inaccurate estimated pure premiums of each of these groups. This notably included the group which had the most accurate 10% most inaccurate estimated pure premiums. On the other hand, the other 2 subgroups had more accurate estimated pure premiums for the 10% most inaccurate estimated pure premiums of each of these groups. One of these two subgroups had the highest subgroup CVaR. This shows that the constraint increases the best subgroup CVaRs and reduces the CVaR of the worst subgroup CVaR. Thus,

Sex	**Age**	$CVaR_{0.90}(Z_a)$ **w/o constraint**	$CVaR_{0.90}(Z_a)$ **w/ constraint**	**Diff.**
M	0-25	337.17	334.01	-3.16
F	0-25	290.08	302.88	12.8
M	25-35	305.49	318.51	13.02
F	25-35	326.06	336.23	10.17
M	35-45	335.73	336.61	0.88
F	35-45	333.84	348.88	15.04
M	45-55	299.56	303.52	3.96
F	45-55	270.43	286.95	16.52
M	55-65	**226.67 (most accurate)**	229.07	2.40
F	55-65	328.46	353.17	24.71
M	65+	**434.13 (most inaccurate)**	424.53	-9.60
F	65+	307.52	316.04	8.52

TABLE 4.4 – Average Tweedie log loss over the 90^{th} percentile of Tweedie log losses per group

Model Type	**CVaR Ratio** ($\alpha = 0.90$)
GBDT without constraint	1.92
GBDT with constraint	**1.85**

TABLE 4.5 – Fairness constraint effect on fairness metric

this reduces the ratio between the worst and the best group CVaR of the losses. In other words, the model sacrificed the worst-case accuracy of the most accurately estimated groups in order to improve the worst-case accuracy of the less accurately estimated groups. In the end, the addition of the fairness constraint did not manage to keep the CVaR ratio (3.17) at or under 1.20. While the model is not fair enough, the addition of the constraint did manage to slightly work towards the goal of creating a fair pure premium estimation model. As a result, additional measures should be taken to increase model worst-case accuracy on the least accurately estimated groups. This includes exploring other values of α for the fairness constraint and the fairness criterion, performing further hyperparameter optimisation and adding additional attributes that contain information about the least accurately estimated groups. Future work could be done to study the application of other constraints based on risk measures to ensure fair pure premium estimations (e.g. [27, 53]).

4.2.2 Constraint effect on model performance

As for model performance, Table 4.6 shows that the increase of overall model fairness by using the CVaR constraint comes at the cost of model performance. In this case, the constraint increased the average Tweedie log loss by .08 and decreased the R-squared by 0.003%, which is not much considering the slight gains made towards fairness. Due to the fact that a form of error parity is used as a metric in this fairness assessment, model accuracy can still be prioritised as long as all subgroups have similar errors. Even with this decrease in overall

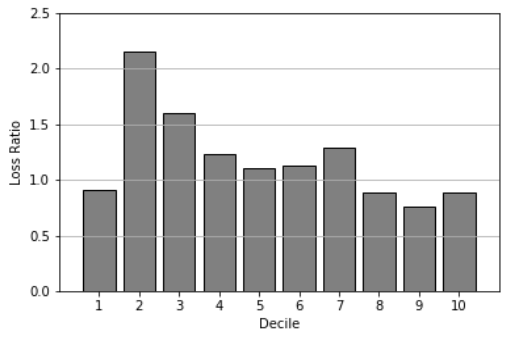
Model	**Mean Tweedie log loss**	**MAE**
Tweedie GBDT w/o constraint	**65.90**	**616.48**
Tweedie GBDT w/ constraint	65.98	624.81
Tweedie GLM	87.55	1,035.20
Poisson and Gamma GLMs	-	934.03

Model	**R-squared**	**Loss ratio**
Tweedie GBDT w/o constraint	**0.00075**	**0.97**
Tweedie GBDT w/ constraint	0.00072	1.09
Tweedie GLM	-0.00028	0.79
Poisson and Gamma GLMs	-0.00010	1.21

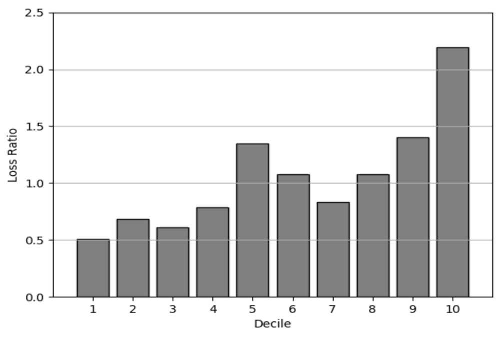
Table 4.6 – Model assessment - performance metrics

accuracy, the GBDT with fairness constraint is much more accurate than Tweedie GLM and the sum of the products of Poisson and Gamma GLMs per claim type. This shows that with proper frameworks and regulations with regards to model fairness and transparency in pure premium estimation, more accurate and fairer premiums could be provided to clients, all while ensuring company profitability. Regarding the loss ratios of each model, both of the GBDTs have overall loss ratios that are close to 1, which shows that the overall sum of estimation pure premiums over large populations will always stay close to the true sum of pure premiums. In contrast, the Tweedie GLM greatly overestimates the average pure premium, while the pure premium estimation model based on the sum of the products of Poisson and Gamma GLMs per claim type underestimates more the average premium for future years.

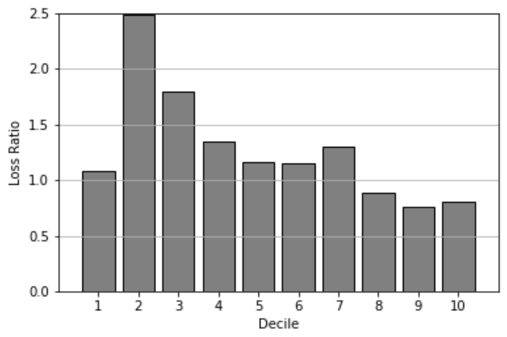
Figure 4.1 contrasts with the previous accuracy metrics, as the loss ratio of the entire test set is split by decile of estimated pure premiums. As the loss ratio of the GBDT without constraint is lower than the one with constraint, the LR charts depict a similar picture. The most noticeable difference can be found in the higher and lower deciles of the LR chart for the GBDT with constraint. Since the constrained model focuses on the worst errors of each subgroup, it overestimates the pure premium of low risks observations and underestimated the pure premium of high risk observations. The Tweedie GLM estimated much higher pure premiums for the majority of estimated pure premium deciles, but was the most accurate on observations with higher true risk. As for the pure premium model based on the sum of the products of Poisson and Gamma GLMs per claim type, Figure 4.1 c) shows that observations with low true risk get much higher premiums than with the Tweedie GBDT without constraint, while estimating similar pure premiums on average from the 6^{th} decile onwards. Compared to the LR charts, the decile plots in Figure 4.2 show less contrast between the two Tweedie GBDT models. Both Tweedie GBDT models can properly detect high risk observations to a similar extent, with 1,702.17$ average claimed amount for the last decile of estimated pure premiums of the Tweedie GBDT without constraint and 1,632.27$ for the Tweedie GBDT with



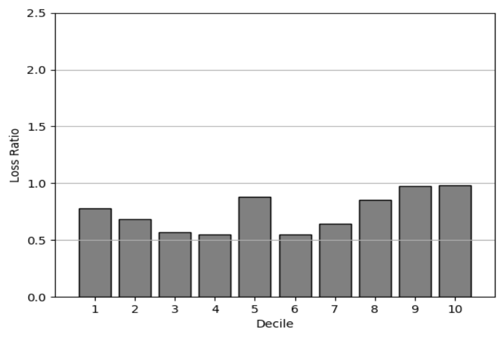
(A) LR chart - Tweedie GBDT w/o constraint



(B) LR chart - Tweedie GBDT w/ constraint



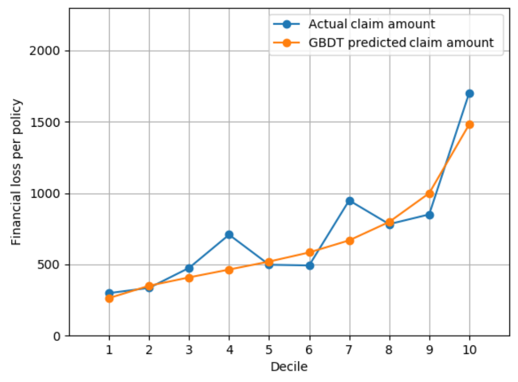
(C) LR chart - Sum of the products of Poisson and Gamma GLMs per claim type



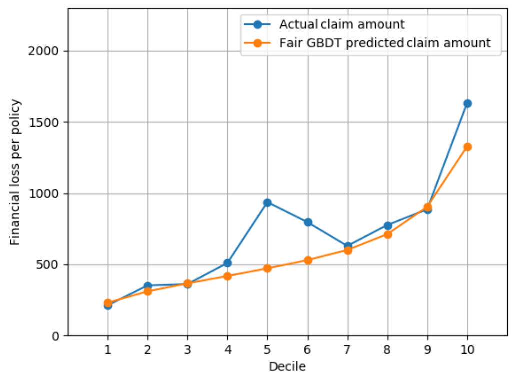
(D) LR chart - Tweedie GLM type

FIGURE 4.1 – Model assessment - ratio of financial loss per decile of estimated pure premium

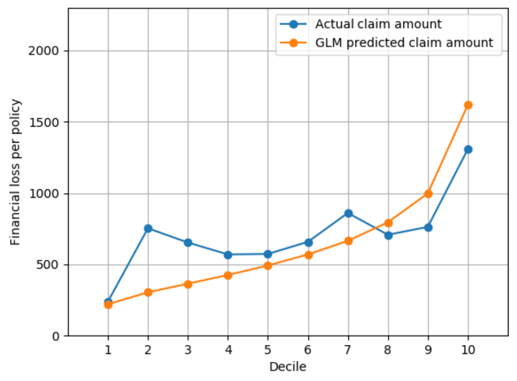
constraint. In addition, the Tweedie GBDT with constraint had a similar capacity in detecting low risk observations, with 298.13$ average claimed amount for the last decile of estimated pure premiums of the Tweedie GBDT without constraint and 211.37$ for the Tweedie GBDT with constraint. Apart from a few differences in certain deciles, both models closely match each estimated pure premium decile with the average claimed amount of the observations contained inside that decile. As for the sum of the product of Poisson and Gamma GLMs per claim type, the deciles of estimated pure premiums are much lower than the actual pure premiums contained, since the estimated pure premiums are lower than the claimed amounts overall. Also, the first decile of pure premiums estimated by the sum of the product of Poisson and Gamma GLMs per claim type contained an average actual pure premium of 238.39$, while the last decile of pure premiums estimated contained an average actual pure premium of 1,307.06$. On the other hand, the Tweedie GLM estimated much higher premiums than the claimed amounts overall and had an average actual pure premium of 283.51$ in its first decile, while its last decile contained an average actual pure premium of 931.39$. As the vertical distances between the first and last deciles in both configurations of GLMs are much smaller, they cannot differentiate low risk observations from high risk observations as well as



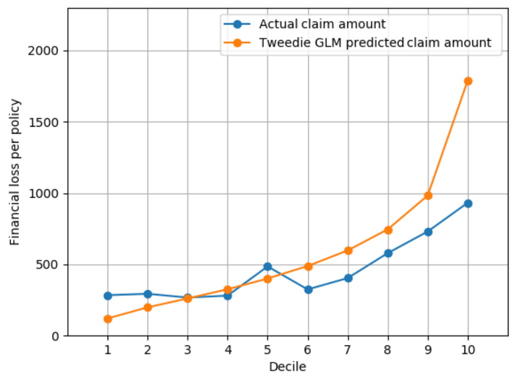
(A) Decile plot - Tweedie GBDT w/o constraint



(B) Decile plot - Tweedie GBDT w/ constraint



(C) Decile plot - Sum of the product of Poisson and Gamma GLMs per claim type



(D) Decile plot - Tweedie GLM

FIGURE 4.2 – Model assessment - ability to differentiate low from high risk assets

the GBDTs.

With all the metrics and charts previously gathered, we can see that using GBDTs for estimating pure premiums, and thus for estimating customer profitability, can provide more accurate individual pure premiums, which corresponds to actuarially fairer pure premiums. Also, with the addition of a fairness constraint such as the one added to our GBDT, strides can be made towards achieving group-fair pure premiums with minimal impact on overall accuracy. In the end, GBDTs are better at providing more competitive pure premiums by being better at differentiating low risk observations from high risk observations.

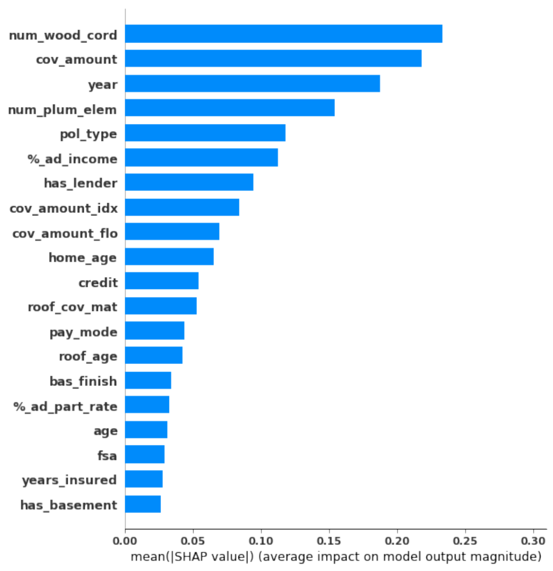
4.2.3 Model interpretation

The current biggest barriers to using models other than GLMs for estimating pure premiums is their lack of model interpretability and prediction explicability. However, as shown in Section 1.3.6.2, GBDT predictions can be explained using SHAP values, while whole models can be interpreted using the mean absolute SHAP values over an entire population. For a single prediction, adding each of its computed SHAP value to mean prediction ϕ_0 results in the final prediction. As a result, the SHAP values of different models can also be used to compare two different GBDTs, such as the model with and without a fairness constraint. The interpretation of the GLMs provided by the partner insurance company could not be compared to the GBDTs due to the confidential nature of these models.

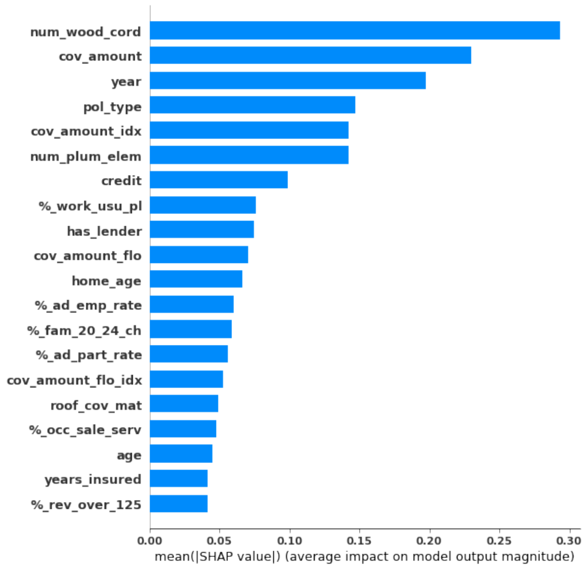
An important point with using SHAP values on a GBDT with a Tweedie objective is that the core model of the GBDT uses log-transformed outcomes, as with Poisson, Gamma or Tweedie GLMs, before using the inverse link function to obtain the true outcomes. As such, the SHAP values are still additive, but they must be transformed with the inverse link function to be properly interpreted. When transformed, each independent SHAP value then represents the multiplicative factor of impact an attribute has with regards to the expected prediction. For example, if a model with a single attribute makes a prediction on a single $\boldsymbol{x}$, where $\phi_1 = 0.25$ and the expected value of the model is $\phi_0 = 5.0$, then the predicted value was $e^{(\phi_0+\phi_1)} = e^{\phi_0} \cdot e^{\phi_1} = e^{5.0} \cdot e^{0.25} = 190.57$. In that case, the expected value was $e^{\phi_0} = e^{5.0} = 148.41$ and the attribute x_1 had the impact of increasing the prediction with regards to the average prediction by a factor of $e^{0.25} = 1.284$, or 28.4%. For the global explainability of the model, the mean absolute SHAP values over overall populations can be transformed using the inverse link function to obtain the average absolute multiplicative factor of impact that each attribute has. Using the average absolute SHAP values, we can then explain the GBDTs which were previously assessed for accuracy and fairness in Section 4.2.1. As both models are similar, we can also study the direct effect that the constraint had on the importance of each attribute.

As can be seen in Figure 4.3, the use of a fairness constraint did not only have an impact on estimating the pure premium for home insurance policies: it also changed the average impact of each variable on the final estimations.

Before underlining the differences, some attributes stay important, constraint notwithstanding. Notably, the three most important factors (the value of the covered assets, the number of wood cords consumed per year and the year of the policy) stay in the same order of importance for both models. Also, some important attributes show the usefulness of being able to manage missing values out of the box. Notably, the attribute *bas_finish*, which indicates the basement finishing, can be null. If the attribute is null, then the house has implicitly no basement. This can also be said for the number of wood cords used per year *(num_wood_coord)*, which implies that the house is heated using a fireplace or a wood stove if that attribute is not null. This is managed out-of-the-box by LightGBM, which is one of its many advantages over traditional GBDTs, as it does not need the presence of an attribute to make a decision based on it, as described in Section 1.3.5.3. As fires are the cause of the biggest claims, it is expected that the number of wood cords consumed per year *(num_wood_coord)* carry the second most importance, with an average multiplicative factor of importance of $e^{0.2316} = 1.26$ in the GBDT



(A) SHAP summary plot - Tweedie GBDT without constraint



(B) SHAP summary plot - Tweedie GBDT with constraint

FIGURE 4.3 – Model assessment - Attribute impact on pure premium

without constraint and $e^{0.2933} = 1.341$ in the GBDT with constraint.

To explore the differences between both models, we can simply use the ratio between their transformed mean absolute SHAP values for each attribute in both graphs in Figure 4.3 to draw conclusions on the effect of the constraint. As the two models have similar expected values ($\phi_0 = e^{5.1393} = 170.60$ for the GBDT without constraint and $\phi_0 = e^{5.1517} = 172.72$ for the GBDT with constraint), their SHAP values can be compared as is. One of the main differences is found at the level of importance of the most important attribute: the number of wooden cords consumed per year *(num_ wood_ coord)*. The initial model had an average importance factor of $e^{0.2316} = 1.261$, while the addition of the constraint increased its importance factor to $e^{0.293} = 1.341$. An increase of 6.3% of importance per prediction. Other notable changes are the increase in the importance of the indexed value of the covered assets *(cov_ amount_ idx)* (8%) and the credit score *(credit)* (4.1%). The increased reliance on these already important risk factors shows that they provide a similar level of information throughout all subgroups. By giving more importance to these attributes, the estimated pure premiums will be more accurate for groups where these attributes were a good indicator of risk. The inverse can be said for other attributes which became less important, such as the percentage of adults in the postal code with income *(%_ ad_ income)* (-7.2%).

Altogether, the change in the interpretation of the model shows that adding the fairness constraint increased the use of information that was useful for groups that were inaccurately assessed and decreased the use of information that was useful for groups that were already well assessed, reducing the overall error parity of the model and thus giving group-fair premiums.

With all the previous findings taken into account, we can conclude that it is possible to use models other than GLMs for pure premium estimation in order to gain accuracy, while also losing minimal model interpretability.

Conclusion

In conclusion, we have shown in this thesis that it is possible to create fairer and more accurate machine learning models which predict future customer profitability in the insurance industry. We also suggest a method to quantify group unfairness in pure premium estimation models, which balances actuarial and group fairness by targeting the worst case inaccuracies of each subgroup. To reduce this unfairness, we propose a fairness constraint that can be applied to gradient boosted decision trees at training time.

The proposed criterion has the advantage of quantifying fairness in a multidisciplinary way by considering the legal, actuarial and machine learning aspects of fairness. However, it still leaves questions unanswered as to the possibility of determining a strict threshold at which discrimination has taken place. The fairness constraint has shown in practice that it was able to reduce unfairness, but it was not able to reduce unfairness to a point where the criterion was satisfied. As such, research should be done to further increase the fairness of pure premium estimation models based on the proposed criterion.

The impacts of this work also raises concerns on current regulations that oversee insurance pricing models with regards to fairness requirements and model transparency. As a more complex model such as a gradient boosted decision tree can have its pure premium estimations explained by knowing the impact of each of the individual attributes related to the insurance policy of a client, other models could become mainstream. As a result, insurance companies could give more competitive pure premiums to clients while still being profitable.

As for fairness requirements, it is crucial that a consensus is reached on what is considered a fair pure premium by considering legal and actuarial fairness definitions, as we attempted to do in this thesis. Since there are currently no precise fairness regulations that require the need to quantify fairness for insurance pricing models, insurance companies have no incentive to quantify unfairness or to adjust models resulting of that quantified unfairness.

Therefore, regulatory bodies should look into providing insurance companies with the tools to ensure fairness, accountability and transparency in pure premium estimation models. These tools include the definition of a proper fairness criterion, of an explicit methodology to audit more complex models and of a method to provide individual subgroup labels, while ensuring their proper use. They should be based on the successes and shortcomings of previously made policies and consider the input of each domain involved to ensure fairness, competitiveness and profitability.

In order to expand the possibilities of what these regulations could and should look like, future research should be done on each level of the model creation process. This includes looking into additional types of models, fairness constraints and insurance branches that could benefit even more from ensured fairness, such as car and life insurance.

Altogether, insurance companies and regulatory agencies need to continue to find a balance between profitability and fairness, in order to ensure that no population is left behind and that everyone pays a fair share of its potential liabilities.

Bibliography

[1] Alekh Agarwal, Miroslav Dudik, and Zhiwei Steven Wu. "Fair Regression: Quantitative Definitions and Reduction-Based Algorithms". *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 120–129.

[2] Richard Berk et al. *A Convex Framework for Fair Regression*. 2017. arXiv: 1706.02409.

[3] Alex Beutel et al. *Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations*. 2017. arXiv: 1707.00075.

[4] Alex Beutel et al. "Putting Fairness Principles into Practice: Challenges, Metrics, and Improvements". *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 2019, pp. 453–459.

[5] Joy Buolamwini and Timnit Gebru. "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification". *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. 2018, pp. 77–91.

[6] "Buying Home Insurance". *Insurance Bureau of Canada* (2020). URL: http://www.ibc.ca/on/home/buying-home-insurance.

[7] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794.

[8] European Commission. *Artificial intelligence: Commission takes forward its work on ethics guidelines*. 2019. URL: http://europa.eu/rapid/press-release_IP-19-1893_en.htm.

[9] Federal Trade Commission. "In FTC Study, Five Percent of Consumers Had Errors on Their Credit Reports That Could Result in Less Favorable Terms for Loans" (2013). URL: https://www.ftc.gov/news-events/press-releases/2013/02/ftc-study-five-percent-consumers-had-errors-their-credit-reports.

[10] United States Congress. *Home Mortgage Disclosure Act*. 1975. URL: https://www.fdic.gov/regulations/laws/rules/6500-3030.html#6500hmda1975.

[11] Microsoft Corporation. *LightGBM*. https://github.com/microsoft/LightGBM. 2017.

[12] Norman Daniels. Genetics and Life Insurance: Medical Underwriting and Social Policy. Ed. by Mark A. Rothstein. 2004. Chap. The Functions of Insurance and the Fairness of Genetic Underwriting, p. 124.

[13] Jeffrey Dastin. *Amazon scraps secret AI recruiting tool that showed bias against women.* 2018. URL: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G.

[14] Cynthia Dwork and Christina Ilvento. "Fairness Under Composition" (2018), pp. 1–72. arXiv: 1806.06122.

[15] Cynthia Dwork et al. "Fairness through Awareness". *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference.* 2012, pp. 214–226.

[16] The Council of the European Union. "Equal Treatment in Goods and Services Directive 2004". *Official Journal of the European Union* (2004).

[17] Walter D. Fisher. "On Grouping for Maximum Homogeneity". *Journal of the American Statistical Association* 53.284 (1958), pp. 789–798.

[18] J. R. Foulds et al. "An Intersectional Definition of Fairness". *2020 IEEE 36th International Conference on Data Engineering (ICDE).* 2020, pp. 1918–1921.

[19] Human Technology Foundation. *Artificial intelligence, solidarity and insurance in Europe and Canada.* 2020.

[20] Yaroslav Ganin et al. "Domain-Adversarial Training of Neural Networks". *J. Mach. Learn. Res.* 17.1 (2016), pp. 2096–2030.

[21] U.S. Government Accountability Office (GAO). "AUTO INSURANCE: State Regulation Affects Cost and Availability" (1986).

[22] Mark Goldburd. *Cas Monograph Series Number 5: Generalized Linear Models for Insurance Rating.* 2016. URL: https://www.casact.org/pubs/monographs/papers/05-Goldburd-Khare-Tevet.pdf.

[23] Christopher P. Guzelian, Michael Ashley Stein, and Hagop S. Akiskal. *Credit scores, lending, and psychosocial disability.* 2015.

[24] Maryann Haggerty. "Allstate Settles Allegations of Insurance Bias". *The Washington Post* (1997). URL: https://www.washingtonpost.com/archive/realestate/1997/02/22/allstate-settles-allegations-of-insurance-bias/eca469f8-ce06-467f-84ac-a03b333a1725/.

[25] Moritz Hardt, Eric Price, and Nathan Srebro. "Equality of Opportunity in Supervised Learning". *Proceedings of the 30th International Conference on Neural Information Processing Systems.* 2016, pp. 3323–3331.

[26] Richard Harris and Doris Forrester. “The Suburban origins of redlining: A Canadian case study, 1935-54”. *Urban Studies* 40.13 (2003), pp. 2661–2686.

[27] Tatsunori Hashimoto et al. “Fairness Without Demographics in Repeated Loss Minimization”. *Proceedings of the 35th International Conference on Machine Learning*. Proceedings of Machine Learning Research. 2018, pp. 1929–1938.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. “The Elements of Statistical Learning Data Mining, Inference, and Prediction” (2009).

[29] Fumio Hayashi. *Econometrics*. 2000.

[30] Burt D. Jones. “An Introduction to Premium Trend”. *CAS Study Note* June (2002).

[31] Bent Jorgensen. “Exponential Dispersion Models”. *Journal of the Royal Statistical Society* 49.2 (1987), pp. 127–162.

[32] Surya Mattu Julia Angwin Jeff Larson and Lauren Kirchner. *Machine Bias*. 2016. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[33] Dana L. Kaersvang. “The fair housing act and disparate impact in homeowners insurance”. *Michigan Law Review* 104.8 (2006), pp. 1993–2018.

[34] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. “Fairness-aware learning through regularization approach”. *Proceedings - IEEE International Conference on Data Mining* (2011), pp. 643–650.

[35] Guolin Ke et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 3149–3157.

[36] Bruce Kellison. “Check the Score - Credit Scoring and Insurance Losses: Is There a Connection?” *Texas Business Review* (2003).

[37] Xavier Landes. “How Fair Is Actuarial Fairness?” *Journal of Business Ethics* 128.3 (2015), pp. 519–533.

[38] Louise Langevin. “Réflexion sur le lien de causalité en matière de discrimination: une difficle intégration”. *Queen’s Law Journal* 22 (1996), pp. 51–80.

[39] Lydia T. Liu et al. “Delayed Impact of Fair Machine Learning”. *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 3150–3158.

[40] Great Britain. Parliament. House of Lords. Select Committee on Artificial Intelligence. *AI in the UK: Ready, Willing and Able?* HL paper v. 1. 2018.

[41] Scott M. Lundberg et al. “From local explanations to global understanding with explainable AI for trees”. *Nature Machine Intelligence* 2.1 (2020), pp. 56–67.

[42] David Madras et al. “Learning Adversarially Fair and Transferable Representations”. *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 3384–3393.

[43] Stephen Mcdonald. “Indirect Gender Discrimination and the ‘ Test-Achats Ruling ’: An Examination of the UK Motor Insurance Market”. *Presentation at the Royal Economic Society Conference* 44.0 (2015).

[44] Michele Merler et al. “Diversity in Faces” (2019), pp. 1–29. arXiv: 1901.10436.

[45] Claire Cain Miller. *When Algorithms Discriminate*. 2015. URL: https://www.nytimes.com/2015/07/10/upshot/when-algorithms-discriminate.html.

[46] Government of Newfoundland and Labrador. *Regulation 46/11*. 2011. URL: https://www.assembly.nl.ca/legislation/sr/regulations/rc110046.htm.

[47] Kate Palmer. “Men are still charged more than women for car insurance, despite EU rule change”. *The Telegraph* (2015). URL: https://www.telegraph.co.uk/finance/personalfinance/insurance/motorinsurance/11521781/Men-are-still-charged-more-than-women-for-car-insurance-despite-EU-rule-change.html.

[48] Executive Office of the President. “Big Data: Seizing Opportunities, Preserving Values”. *Big Data: An Exploration of Opportunities, Values, and Privacy Issues* 1.May (2014), pp. 1–92.

[49] Government of Quebec. *Automobile Insurance Act*. 1977. URL: http://legisquebec.gouv.qc.ca/fr/ShowDoc/cs/A-25?langCont=en#se:180.

[50] Government of Quebec. *Charter of Human Rights and Freedoms*. 1975. URL: http://legisquebec.gouv.qc.ca/en/ShowDoc/cs/C-12.

[51] T.R. Reid. “Montana Implements Policy Of ’Unisex’ Insurance”. *The Washington Post* (1985). URL: https://www.washingtonpost.com/archive/politics/1985/10/01/montana-implements-policy-of-unisex-insurance/c8897a22-a667-4d6e-acb2-b695702e1b31/.

[52] R. Tyrrell Rockafellar and Stanislav Uryasev. “Optimization of conditional value-at-risk”. *The Journal of Risk* 2.3 (2000), pp. 21–41.

[53] Shiori Sagawa et al. “Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization” (2019), pp. 1–19.

[54] Gordon K Smyth. “Regression analysis of quantity data with exact zeroes.” *Proceedings of the Second Australia-Japan Workshop on Stochastic Models in Engineering, Technology and Management.* (1996), pp. 572–580.

[55] Gregory D. Squires. “Racial profiling, insurance style: Insurance redlining and the uneven development of metropolitan areas”. *Journal of Urban Affairs* 25.4 (2003), pp. 391–410.

[56] Uryasev Stan et al. “VaR vs CVaR in Risk Management and Optimization”. *CARISMA conference* (2010).

[57] “State Farm Unit Seeks Bias Settlement With HUD”. *The Washington Post* (1995). URL: https://www.washingtonpost.com/archive/realestate/1995/04/08/state-farm-unit-seeks-bias-settlement-with-hud/b14d0d33-2e2b-4e31-a659-6c75ff146290/.

[58] State of Michigan. *THE INSURANCE CODE OF 1956*. 1956.

[59] Ali Sultan. *Accuracy vs. area under the ROC curve*. Cross Validated - Stack Exchange. URL: https://stats.stackexchange.com/q/225210.

[60] M. C. K. Tweedie. “An index which distinguishes between some important exponential families”. *Statistics: applications and new directions*. 1984, pp. 579–604.

[61] “Uniform Guidelines on Employee Selection Procedures”. *Federal Register* 43 (1978), pp. 38290–38309.

[62] Cédric Villani et al. *Donner un sens à l’intelligence artificielle : Pour une stratégie nationale et européenne*. 2018.

[63] Xuezhi Wang et al. “Practical Compositional Fairness: Understanding Fairness in Multi-Task ML Systems” (2019). arXiv: 1911.01916.

[64] Kent West. “Gender in Automobile Insurance Underwriting: Some Insureds Are More Equal Than Others”. *Alberta Law Review* 50 (2018).

[65] “What Realtors should know about property insurance”. *The Canadian Real Estate Association* (2005), pp. 1–24.

[66] Robert Williamson and Aditya Menon. “Fairness risk measures”. *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 6786–6797.

[67] Yi Yang, Wei Qian, and Hui Zou. “Insurance Premium Prediction via Gradient Tree-Boosted Tweedie Compound Poisson Models”. *Journal of Business and Economic Statistics* 36.3 (2018), pp. 456–470.

[68] J. Yao, J. Zhang, and L. Wang. “A financial statement fraud detection model based on hybrid data mining methods”. *2018 International Conference on Artificial Intelligence and Big Data*. 2018, pp. 57–61.

[69] Rich Zemel et al. “Learning Fair Representations”. *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. 325–333.

[70] Yan Zhang. “Assessing fair lending risks using race/ethnicity proxies”. *Management Science* 64.1 (2018), pp. 178–197.