Lisbon School of Economics & Management
Universidade de Lisboa

Master in Actuarial Science

Master's Final Work

Internship Report

Interpreting Third-Party Liability Model with SHAP

Cheng Chen

Faculty Advisor: João Afonso Bastos
Industry Advisor: Malachy Toner
October 2024

Acknowledgments

I would like to thank the following members for their help:

Malachy Toner, my industry advisor from Liberty Insurance's risk modeling team, is an experienced pricing actuary who guided me through the entire project during my internship. He also provided me with training sessions covering the basics of pricing, various topics related to the private car insurance industry in Ireland, and advice on the Institute and Faculty of Actuaries (IFoA) exams.

Prof. Doutor João Afonso Bastos, my faculty advisor at Instituto Superior de Economia e Gestão (ISEG), provided me with guidance and feedback on this report. He is also the professor of my Actuarial Data Science course at ISEG, and the techniques I learned from that course greatly helped me in this project.

Ciara Logue, my line manager at the risk modeling team, provided me with help during my internship and feedback on several of my presentations, including this project.

Gaston Presman, from the risk modeling team, is a knowledgeable actuary who greatly inspired me whenever I got stuck.

Zhifeng Xu, from the risk modeling team, and also an alumnus of ISEG's actuarial master's program, provided me with advice about the project and shared many valuable experiences with me.

Rodrigo Serra, from the street pricing team, walked me through the SHAP process at Liberty and clarified some questions about the dataset.

Filipe Santos, from the demand modeling team, provided me with support on Liberty's data science platform "Cortex" and both the theoretical and practical aspects of SHAP.

Abstract

Applying data science techniques in the insurance industry has become increasingly popular in recent years, especially in the pricing of non-life insurance. Machine Learning (ML) models have demonstrated better accuracy compared to traditional Generalized Linear Model (GLM). However, in terms of interpretability, GLM is inherently easier to explain than ML models. The beta coefficients of GLM provide straightforward indicators of how important a factor is and how it influences the predictions. As a result, finding ways to better interpret ML models has become essential for companies using solely ML models or ensemble models that combine ML and GLM results. SHapley Additive exPlanations (SHAP), a model-agnostic method and an application of Shapley Value in machine learning, can help us better understand the mechanisms behind ML's "black box". We expect SHAP to play a similar role in ML as the beta coefficients do in GLM. This project primarily focuses on using SHAP to analyze the ML model, rather than the model itself, aiming to make companies feel more confident when using models that are less interpretable. In addition to using SHAP, we also applied another method called Partial Dependence Plot (PDP) to validate the conclusions drawn from SHAP.

Keywords: Shapley value, generalized linear model, machine learning, insurance pricing, partial dependence plot

Resumo

A aplicação de técnicas de ciência de dados na indústria de seguros tem se tornado cada vez mais popular nos últimos anos, especialmente na precificação de seguros de não-vida. Os modelos de ML (aprendizado de máquina) demonstraram melhor precisão em comparação aos GLM (modelos lineares generalizados) tradicionais. No entanto, em termos de interpretabilidade, os GLM são inerentemente mais fáceis de explicar do que os modelos de ML. Os coeficientes beta dos GLM fornecem indicadores claros sobre a importância de um fator e como ele influencia as previsões. Como resultado, encontrar maneiras de interpretar melhor os modelos de ML tornou-se essencial para empresas que utilizam exclusivamente modelos de ML ou modelos em conjunto que combinam resultados de ML e GLM. O SHAP, um método independente de modelo e uma aplicação do Valor de Shapley no aprendizado de máquina, pode nos ajudar a entender melhor os mecanismos por trás da "caixa-preta" dos modelos de ML. Esperamos que o SHAP desempenhe um papel semelhante nos modelos de ML ao que os coeficientes beta fazem nos GLM. Este projeto foca principalmente no uso do SHAP para analisar o modelo de ML, em vez do próprio modelo, com o objetivo de fazer as empresas se sentirem mais seguras ao utilizar modelos que são menos interpretáveis. Além de utilizar o SHAP, também aplicamos outro método chamado PDP para validar as conclusões obtidas a partir do SHAP.

Palavras-chave: valor de Shapley, modelo linear generalizado, aprendizado de máquina, precificação de seguros, gráfico de dependência parcial

Contents

Chapter 1

Introduction

1.1 Background

This is an internship report conducted within the risk modeling team of Liberty Seguros, Compañía de Seguros y Reaseguros, S.A. – Sucursal em Portugal.

Liberty Seguros, Compañía de Seguros y Reaseguros, S.A. (Liberty Seguros), as part of Liberty Mutual Insurance Group (LMIG), was an insurance company operating in Spain, Portugal, Ireland, and Northern Ireland. On January 31st, 2024 [1], Assicurazioni Generali S.p.A. (Generali Group) completed the acquisition of Liberty Seguros from LMIG, and its Portuguese branch is now named Generali Seguros y Reaseguros, S.A. - Sucursal em Portugal.

Old Name	**Name After Acquisition**
Liberty Seguros, Compania de Seguros y Reaseguros, S.A. (Liberty Seguros)	Generali Seguros y Reaseguros, S.A.U. (Generali Seguros)
Portugal: Liberty Seguros, Compania de Seguros y Reaseguros, S.A. – Sucursal em Portugal.	**Portugal:** Generali Seguros y Reaseguros, S.A. - Sucursal em Portugal
Ireland: Liberty Insurance	**Ireland:** RedClick

Table 1.1: Name changes after acquisition

The goal of Liberty's risk modeling team is to build the best-practice pricing strategy for Liberty's property insurance products in Spain, Ireland, and Portugal. This project studied the Third-Party Liability Model of the personal car insurance product in Ireland.

1.2 Objectives

The importance of getting an accurate pricing result in a property and casualty insurance company is similar to avoiding the "Adverse Selection" in a health insurance company.

For health insurance products, the company does not have as much information as the customers, so some customers would buy health insurance once they notice that their health has started to deteriorate.

In car insurance, the technical pricing result is based on how much we expect to have to pay out in claims for that customer.

If we cannot obtain enough information from the data we receive, the premium would deviate from its true value, which would attract “unqualified” customers to this level of premium and subsequently increase the risk to the company.

Furthermore, according to the rules of the Central Bank of Ireland released in 2021 [2, p. 5], insurance companies’ pricing policies and processes should be reviewed annually to:

- Focus on the impact these practices may have on customers.
- Ensure fair treatment of consumers.

An ML model is used in the pricing process. It outperforms GLM in both accuracy and efficiency; however, it operates like a “black box” and is more difficult to explain compared to GLM. Therefore, to better understand its impact on customers, methods should be applied to interpret the ML model.

The main objective of this project is to use SHAP and PDP to interpret the Gradient Boosting Machine (GBM) model used in the Third Party Liability: Personal Injury (TPI) coverage of private car insurance in Ireland. Gaining a better understanding of the ML model would not only improve the pricing process but also help meet regulatory requirements.

1.3 Structure of the Report

The first chapter of this report introduces the background and objectives.

The second chapter provides details about the insurance product, the dataset and variables used, and the GBM model.

The third chapter explains the basic concept of Shapley Value and its application in machine learning, specifically SHAP, and uses SHAP to analyze the pricing model.

The fourth chapter uses PDP as an alternative method for interpreting the pricing model and compares the results and conclusions with those from the previous chapter.

The fifth and final chapter presents the project’s conclusions and raises several topics for potential further investigation.

Chapter 2

Product, Dataset and Model

2.1 Product Information

2.1.1 Coverage

This project uses the pricing model for Liberty Ireland (RedClick)'s private car insurance [3]. For this type of product, there are seven types of coverage [4, p. 4]:

No.	**Coverage**
1	Liability to others
2	Own damage
3	Fire cover
4	Theft cover
5	Windscreen cover
6	Breakdown assistance
7	Driving other cars

Table 2.1: Coverages for Liberty Ireland's private car insurance

For the above coverages, there are three combinations [5, p. 1]:

- **Third Party Only**: with coverage 1
  Provides the minimum cover required by law, including cover for third-party property damage resulting from an insured loss up to €30,000,000, and unlimited cover for third-party personal injury.
- **Third Party, Fire and Theft**: with coverage 1, 3, 4
  Covers damage or loss to the insured vehicle resulting from attempted theft, theft, or fire. It also includes cover for third-party property damage up to €30,000,000 due to an insured loss, and provides unlimited cover for third-party personal injury.
- **Comprehensive**: with coverage 1, 2, 3, 4, 5
  This provides the coverage outlined in the two coverages mentioned above, along with cover for accidental damage to the insured vehicle.

Optional coverages 6 and 7 are applicable if listed in the insurance schedule.

To model this product, we use four types of model:

Accidental Damage (AD)	related to coverage 2.
Fire and Theft (FT)	related to coverage 3, 4.
Third Party Liability: Personal Injury (TPI)	part of coverage 1.
Third Party Liability: Property Damage (TPD)	part of coverage 1.

Table 2.2: Main combinations of coverage

Liability to others, commonly referred to as Third Party Liability, provides coverage under which RedClick insures the policyholder against legal liability for damages (including associated costs and expenses) related to [4, pp. 15–16]

- death or bodily injury to any person
- damage to property

arising from an accident involving the insured vehicle. Additionally, RedClick will not pay more than €30,000,000 for property damage resulting from any single claim or multiple claims arising from the same cause.

Among these four perils, we focused on the TPI model in this project, as it is the most complex model when considering the number of factors involved.

2.2 Dataset and Variables

2.2.1 Dataset

The dataset used in this study is the Direct - New Business - 2022 dataset.

- Direct: Indicates that the data were sourced from Liberty's direct channel, rather than through brokers.
- New Business (NB): Refers to data from new clients, excluding those from renewals.
- 2022: Represents the exposure year.

The dataset comprises 479,506 records across 36 factors, along with 479,506 pricing results generated by the TPI model.

2.2.2 Variable List

Due to confidentiality, the names of all factors have been anonymized. The list of variables, along with their types and categories, is as follows:

No.	Type	Category	No.	Type	Category
F-1	Continuous	Policy Information	F-19	Discrete	Vehicle Information
F-2	Discrete	Policy Information	F-20	Categorical	Geographic Information
F-3	Continuous	Policy Information	F-21	Discrete	Driver Information
F-4	Categorical	Geographic Information	F-22	Discrete	Policy Information
F-5	Discrete	Vehicle Information	F-23	Discrete	Policy Information
F-6	Categorical	Vehicle Information	F-24	Discrete	Vehicle Information
F-7	Discrete	Vehicle Information	F-25	Discrete	Vehicle Information
F-8	Categorical	Vehicle Information	F-26	Discrete	Driver Information
F-9	Discrete	Vehicle Information	F-27	Discrete	Driver Information
F-10	Continuous	Policy Information	F-28	Discrete	Vehicle Information
F-11	Discrete	Driver Information	F-29	Discrete	Vehicle Information
F-12	Discrete	Claim Information	F-30	Discrete	Vehicle Information
F-13	Discrete	Time Information	F-31	Categorical	Policy Information
F-14	Continuous	Policy Information	F-32	Categorical	Vehicle Information
F-15	Discrete	Vehicle Information	F-33	Categorical	Driver Information
F-16	Discrete	Policy Information	F-34	Categorical	Driver Information
F-17	Discrete	Policy Information	F-35	Categorical	Policy Information
F-18	Discrete	Vehicle Information	F-36	Categorical	Policy Information

Table 2.3: Variable list

The reason of choosing direct channel data rather than broker channel data is because of its quality, for some factors like F-35, not all brokers would give the details of these factors. Use Label Encoding for ordinal categorical variables and One-Hot Encoding for nominal categorical variables. For continuous variables, their values are turned into several ranges when the GBM model is applied.

As observed, the dataset we received is structured. Among various machine learning algorithms, GBM performs particularly well with structured data. It consistently excels in competitions on Kaggle, a prominent data science platform, when applied to this type of dataset. [6, p. 1]

2.3 General Information about the Model

Although the ML modeling process is not the primary focus of this report, and developing such a model was not part of my internship tasks, it's helpful to provide a brief overview of the model we use and compare it to the traditional GLM. This comparison will provide context for the advantages and limitations of the machine learning approach in relation to more conventional methods.

2.3.1 Introduction: tree model

The following subsection (2.3.1) is based on the work of James et al. [7].

The model we used for the TPI is a GBM, a method based on decision trees. The mechanism involves segmenting the predictor space into several subspaces, and the tree model can function as either: [7, pp. 327–335]

- Regression Tree: Predicts by using the mean value within each subspace.
- Classification Tree: Predicts by using the mode value within each subspace.

The process of training a regression tree begins with the construction of a recursive binary tree. After the tree is built, it often results in overfitting because of its complexity. To address this, pruning techniques are applied to reduce the depth of the model, simplifying it by removing unnecessary splits. This reduces the model's variance and improves its generalization performance on new data.

After pruning, if the tree model results in M partitions, its structure would take the form: [7, p. 338]

$$f(X) = \sum_{m=1}^{M} c_m \cdot \mathbb{1}_{(X \in R_m)} \tag{2.1}$$

Where M is the number of partitions after pruning, and c_m represents the mean of the y-values of the observations within partition R_m.

2.3.2 Ensemble and Gradient Boosting

Ensemble

This ensemble part of subsection (2.3.2) is based on the work of James et al. [7].

Although a single decision tree may not perform well, combining multiple trees can result in a highly effective model. The ensemble method is a technique that integrates many simple "building block" models to form a more robust and potentially powerful model [7, p. 340].

For tree models, several ensemble methods can be applied, including:

1. Bagging
2. Random Forest

3. Boosting

In this report, since only the boosting method is employed, we will focus on discussing it in detail below.

Boosting

This boosting part of subsection (2.3.2) is informed by the work of Hastie et al. [8] and Li [9].

In 1997, Freund and Schapire introduced a boosting algorithm called "AdaBoost.M1". The key idea behind this algorithm is to fit an additive model [9, pp. 4–7]

$$H(x) = \sum_t \rho_t h_t(x) \tag{2.2}$$

using a forward stage-wise approach.

At each stage of the boosting process, a new weak learner is introduced to address the shortcomings of the existing weak learners. These shortcomings are identified by assigning higher weights to data points that were misclassified in previous stages.

If the model fails to predict certain data points accurately, the weights of these data points are increased in the next stage. The updated dataset is then used to train the next weak learner, $h(x)$. Furthermore, the weight ρ_t is assigned to ensure that more accurate classifiers have a greater influence on the final model [8, pp. 337–341].

Gradient Boosting

This gradient boosting part of subsection (2.3.2) follows the explanation by Li[9].

In 2001, Friedman developed the explicit regression gradient boosting algorithm. The key idea behind gradient boosting is to combine the principles of boosting with gradient descent optimization [9, pp. 7–9].

For a regression problem using the squared loss function:

$$L(y, F(x)) = \frac{[y - F(x)]^2}{2} \tag{2.3}$$

Starting with an initial model $F(x)$ and observations $(x_i, y_i), i = 1, \cdots, n$, the goal is to minimize the total loss $J = \sum_i L(y_i, F(x_i))$. To achieve this, we treat $F(x_1), F(x_2), \cdots, F(x_n)$ as parameters and compute the derivatives accordingly: [9, pp. 11–47]

$$\frac{\partial J}{\partial F(x_j)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_j)} = \frac{\partial L(y_j, F(x_j))}{\partial F(x_j)} = F(x_j) - y_j \tag{2.4}$$

The negative of the gradients is expressed as:

$$y_j - F(x_j) = -\frac{\partial J}{\partial F(x_j)} \tag{2.5}$$

This equals to the residuals that the current model $F(x)$ is unable to predict accurately.

Next, fit a regression tree h to the negative gradients (i.e., the residuals): $(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \cdots, (x_n, y_n - F(x_n))$. Then update the model as $F = F + h$. If the model performance is still insufficient, another regression tree can be added to further improve accuracy. This iterative process continues until the model fits the data adequately.

Compared with AdaBoost: [9, p. 7]

1. In Gradient Boosting: the model's shortcomings are identified through the gradients of the loss function.
2. In AdaBoost: the shortcomings are identified by assigning higher weights to misclassified data points.

2.3.3 Comparison between GBM and GLM

Review of GLM

At Liberty Ireland, the old pricing model was built using GLM.

For the TPI Loss Cost (LC) GLM model, we divided it into three sub-models, along with a parameter called **TPIL propensity**, which represents the predicted probability of a large claim.

1. **TPI Freq**: frequency model for the claim.
2. **TPIS Sev**: severity model for small claim amount.
3. **TPIL Sev**: severity model for large claim amount.

The loss cost is calculated as follows:

$$\text{TPIS LC} = \text{TPI Freq.} \times (1 - \text{TPIL Prop.}) \times \text{TPIS Sev.} \tag{2.6}$$

$$\text{TPIL LC} = \text{TPI Freq.} \times \text{TPIL Prop.} \times \text{TPIL Sev.} \tag{2.7}$$

The following part of subsection (2.3.3) is based on the material presented by Dunn et al. [10].

The GLM is an extension of the linear regression model and is expressed in the following form: [10, p. 229]

$$g\left(\mathbb{E}\left[\mathbf{Y}\right]\right)=\mathbf{X}\boldsymbol{\beta}=\beta_0+\sum_{j=1}^{n}\beta_j x_j \tag{2.8}$$

Here, $g(z)$ represents the link function, which can be $\ln(z)$, referred to as the log link. The link function can also take other forms, depending on the problem setting.

For the log link function, the equation takes the following form: [10, pp. 233–234]

$$\mathbb{E}\left(Y_i\right)=\exp\left(\beta_0\right)\cdot\exp\left(\beta_1 X_{i1}\right)\cdot\exp\left(\beta_2 X_{i2}\right)\cdot\ldots\cdot\exp\left(\beta_n X_{in}\right) \tag{2.9}$$

Aside from choosing the link function, we also need to determine which exponential family the response variable $\mathbf{Y}$ belongs to. An exponential family is a group of probability distributions that can be fully characterized by their mean and variance, where the variance is a function of the mean.

By selecting a distribution for our model, we implicitly assume a specific relationship between the mean and variance of $\mathbf{Y}$.

By selecting the following:

1. **Link Function** $g(z)$.
2. **Exponential Family** for the response variable $\mathbf{Y}$.

we can improve linear regression in the following three aspects:

1. Setting constraints on $\mathbf{Y}$:
   In the insurance industry, we do not expect claim amounts to be negative. Therefore, by selecting an appropriate distribution for $\mathbf{Y}$, we can impose this non-negativity constraint on the response variable [10, pp. 430–431].
2. Relaxing the constraint on ε:
   In linear regression, it is assumed that the error terms ε_{i} are normally distributed with a mean of 0 and constant variance σ^2. However, this assumption is often too strict for practical applications. In a GLM, this requirement is relaxed, allowing for greater flexibility in the error structure [10, pp. 233–234].
3. Transforming from additive to multiplicative structure:
   In linear regression, the model follows an additive structure:

   $$\mathbf{Y}=\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\epsilon} \tag{2.10}$$

   with the expected value of $\mathbf{Y}_i$ given by:

   $$\mathbb{E}\left(Y_i\right)=\beta_0+\beta_1 X_{i1}+\beta_2 X_{i2}+\ldots+\beta_n X_{in} \tag{2.11}$$

   However, when using a log link function in a GLM, the additive relationship is transformed into a multiplicative one, as shown in formula 2.9. This transformation is particularly important in the insurance industry, where features often have multiplicative effects on the response variable, making GLMs more suitable for modeling such interactions.

In summary, the GLM relaxes some of the strict assumptions of linear regression and introduces new ones.

For the TPI Freq model, the Poisson distribution is selected for $\mathbf{Y}$, along with a log-link function. In contrast, for the TPIS Sev and TPIL Sev models, the Gamma distribution is used for $\mathbf{Y}$, with a log-link function.

Comparison

In comparison, GLM involves more human influence, requires less coding, and offers greater transparency. On the other hand, GBM are capable of uncovering deeper patterns within the data, require more coding, but offer less transparency.

When it comes to the pricing process, GLM and GBM perform differently across several key aspects:

1. **Accuracy:**
   GBM generally outperforms GLM and other machine learning algorithms in most cases when applied to structured data.
2. **Monotonicity Constraints:**
   In the pricing process, we may want to impose monotonicity constraints on certain features. For instance, when modeling frequency, if the past claim history is worse, the response variable should also be predicted as worse, reflecting a logical progression.
   In GLM, when a log link function is chosen, the coefficients (betas) provide insight into the relationships between the features and the response variable.
   In contrast, for GBM models such as LightGBM, monotonicity constraints must be explicitly defined. This is done by specifying a Python list, such as $[0, -1, 1]$, where 0 indicates no restriction, 1 enforces a monotonic decreasing relationship, and 1 enforces a monotonic increasing relationship. [11]
3. **Interpretability and Explainability**
   Interpreting GLM is relatively straightforward because the model can be expressed using a simple formula. However, interpreting GBM is more challenging. In this project, the primary focus is on interpreting GBM models using SHAP. SHAP are analogous to the betas in GLM, but

   (a) GLM's betas predict the response variable:

$$\hat{y}_i = g^{-1}\left[\beta_0 + \sum_j \beta_j X_{j,i}\right] \tag{2.12}$$

   While GBM's SHAP values predict the difference from the mean prediction, attributing each feature's contribution to the deviation from the average prediction:

$$\hat{y}_i - \mathbb{E}[\hat{y}] = \sum_j \text{SHAP}_{j,i} \tag{2.13}$$

(b) GLM's beta coefficients are directly calculated as part of the GLM model fitting process, while SHAP values for GBM are computed separately after the model has been developed.

(c) The number of betas and SHAP values differs after encoding categorical variables into dummy variables. In GLM, each column of the dataset corresponds to a single beta coefficient. However, for SHAP values, each entry in the $m \times n$ dataset has its own SHAP value. For any given row, SHAP generates n SHAP values, where n is the number of columns.

2.3.4 Platform, Software and Programming Language

The traditional GLM model was developed using Emblem and Radar, software provided by Willis Towers Watson (WTW), a consulting firm. Emblem is utilized for model development, while Radar is used to manage and evaluate the resulting models.

In contrast, the GBM model was developed on the Cortex platform using Python as the programming language. Cortex [12] is a platform that allows Liberty's data scientists to set up modeling workstations on AWS, equipped with the necessary data and tools, within an appropriately scaled GPU environment. This is achieved simply by specifying the technical and dataset requirements. The open-source Python packages used for GBM development include LightGBM and Optuna. The code is executed in JupyterLab, with model implementation carried out in Earnix, a software developed by an Israeli company.

The SHAP process in this project is also built on Cortex, using the open-source Python package SHAP.

When comparing the development of GLM and GBM from a practical perspective, key factors include IT infrastructure limitations, process maturity, and the speed of model development.

Over the past decade, advancements in GPUs and other hardware have significantly eased the application of ML models. Prior to these advancements, IT infrastructure limitations were a major obstacle for ML, in contrast to GLM.

Regarding process maturity, the GLM model has been developed and maintained at Liberty Ireland for over twenty years, whereas the application of ML models began less than a decade ago.

In terms of development speed, fitting a GLM model is generally faster than fitting a GBM model for the following reasons:

1. When setting up the initial model, GBM requires considerable time for tuning, whereas GLM does not.
2. When modifying features in a GLM, the variables are additive, making it relatively easy to adjust a single variable without significantly affecting the rest of the model. In contrast, GBM models are more integrated, meaning that changing the model often necessitates retraining the entire model from scratch.

2.3.5 Restricted and Unrestricted Models

In the pricing process, two versions of the model are used: the restricted model and the unrestricted model. The unrestricted model, also called the technical model, provides the most accurate results. However, there are several limitations in practical applications. The most common reasons include:

1. **Legal and Compliance:** For instance, certain variables, such as the driver's gender, cannot be included in the model due to regulatory restrictions.
2. **IT Limitation:** Some legacy systems and platforms are not able to handle the large volume of data required for the model.

The restricted model is derived from the unrestricted model by applying various limitations to meet regulatory requirements, and its results are used to determine the final rates. Although the rates of the unrestricted model are not used directly for commercial purposes, its results are crucial for assessing the profitability of the company. As discussed in Section 2.3.4, modifying the GLM model is relatively straightforward, while altering the GBM model is more complex. This makes it particularly challenging to apply restrictions to the unrestricted GBM model.

Chapter 3

Interpreting using SHAP

3.1 Introduction about Shapley Value: Theoretical Part

This section (3.1, 3.1.1 and 3.1.2) follows Molnar's explanation of Shapley Value in his textbook [13]. The key point about Shapley Value is that it is a model-agnostic method, meaning that it can be applied to any model without requiring knowledge of the model's internal mechanisms. Whether it is a simple model like linear regression or a more complex model like GBM, Shapley Value can be calculated in both cases.

To introduce the Shapley Value in one sentence, it calculates a value for each input feature, indicating how that feature contributes to the prediction for a specific input record.

Take the figure on the homepage of the Python package shap as an example [14]. The model uses four features: age, sex, BP, and BMI. The model output is 0.4, while the mean output of the dataset is 0.1. The difference of 0.3 is explained by the sum of the SHAP values for each feature. In this case, being 65 years old contributes positively by 0.4 to the result, while being female contributes negatively by 0.3.

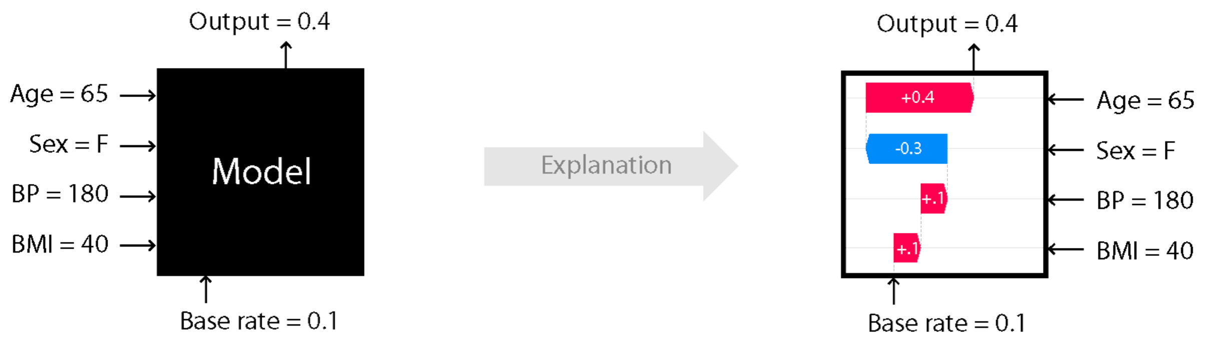

Figure 3.1: Example in Python Package **shap**'s Homepage [14]

3.1.1 Original Concept in Game Theory

Shapley Value was first introduced by Lloyd Shapley, a Nobel laureate in economics, in 1953 [15] to address problems in game theory. It is used to fairly allocate profits or costs within a group.

Consider a coalition of three players with a total payout of 51. The question of how to allocate this payout in a mathematically fair manner is solved by the Shapley Value. A key concept in this context is the **marginal contribution**, which is defined as follows: [13, p. 22]

The marginal contribution of a player to a coalition is the difference between the value of the coalition with the player and the value of the coalition without the player.

The concept used in the following formula is defined as follows: [13, pp. 25–26]

- **Players:** $P_1, P_2, \cdots, P_N$
- **Coalition of all players:** U
- **Coalition of no players:** $\emptyset$
- **Coalition:** S, a set of several players.
- **Size of a Coalition:** $|S|$, for example $|P_2, P_7, P_{11}| = 3$, $|U| = N$
- **Value function:** $v()$, maps from all possible coalition of N players to the payout of that coalition, a real number.
- **Total Payout:** $v(U)$, the cost with all players
- **Shapley Value:** ϕ_j for player j

The Shapley Value is then calculated using the following formula:

$$\phi_j = \sum_{S \subseteq U \setminus \{j\}} \frac{|S|!(N - |S| - 1)!}{N!} [v(S \cup j) - v(S)] \tag{3.1}$$

To explain the formula step by step:

- $v(S \cup j) - v(S)$: This represents the marginal contribution of the player j, which is the change in the total payout after adding the player j to the coalition set S.
- $\sum_{S \subseteq U \setminus \{j\}}$: This sums over all possible coalitions that do not include player j, considering the contribution of j to each of these coalitions.
- $\frac{|S|!(N-|S|-1)!}{N!}$: This is the weighting factor for the marginal contribution, calculated based on the number of possible coalitions involving S, ranging from the empty set $\emptyset$ to $U \setminus \{j\}$. The sum of all weights equals 1, ensuring a fair distribution across all coalitions.

The Shapley Value, defined in this way, possesses a unique property:

Efficiency
The sum of the Shapley Values for all players equals the total payout: [13, p. 27]

$$\sum_{j \in U} \phi_j = v(U) \tag{3.2}$$

This property is crucial because it ensures that the total payout is distributed fairly among all players. If the sum of the payouts allocated to each player is not equal to the total payout, the method would fail to achieve its objective of fair distribution.

3.1.2 Application in ML: SHAP

Although Lloyd Shapley introduced the Shapley Value in the mid-20th century, it was not until 2017 that Lundberg and Lee published a paper [16] that applied SHapley Additive exPlanations (SHAP) to interpret machine learning models. One reason for this delay is that, when the Shapley Value was first introduced, machine learning had not yet developed to its current state due to the computational limitations of that era.

Let the machine learning model be denoted as f, and the data input as $x^i \in X$, where the model prediction is given by $f(x^i) \in \mathbb{R}$. As introduced in Section 3.1.1, the concepts used in the following formula are defined as follows: [13, pp. 30–34]

Game Theory Concept	**ML Concept**	**Term**
Player index	Feature index	j, F_j is a feature
Coalition of all players	Set of all feature indexes	$U = \{1, 2, \cdots, p\}$
Coalition of no players	Set of no features	$\emptyset$
Coalition	Set of feature indexes	$S \subseteq U$
Players not in Coalition	Indexes of features outside set S	$C := S^c$
Coalition Size	Number of features in S	$\|S\|$
Total number of players	Total number of features	p
Value function	Prediction for feature value in S minus expected	$v_{x^i}(S)$
Total Payout	Prediction for feature value in U (which is x^i) minus average prediction	$f(x^i) - \mathbb{E}(f(X))$
SHAP values	Contribution of F_j towards total payout	ϕ_j^i

Table 3.1: Concepts used for SHAP formula for ML

The value function, given f and x^i, is defined as follows:

$$v_{x^i}(S) = \int_{x_C \in C} f(x_S^i \cup x_C) dx_C - \mathbb{E}(f(X)) \tag{3.3}$$

In this formula, the data input $x_S^i \cup x_C$ represents a combined data point, where values for features in set S come from x^i, and values for features in set C are not specified.

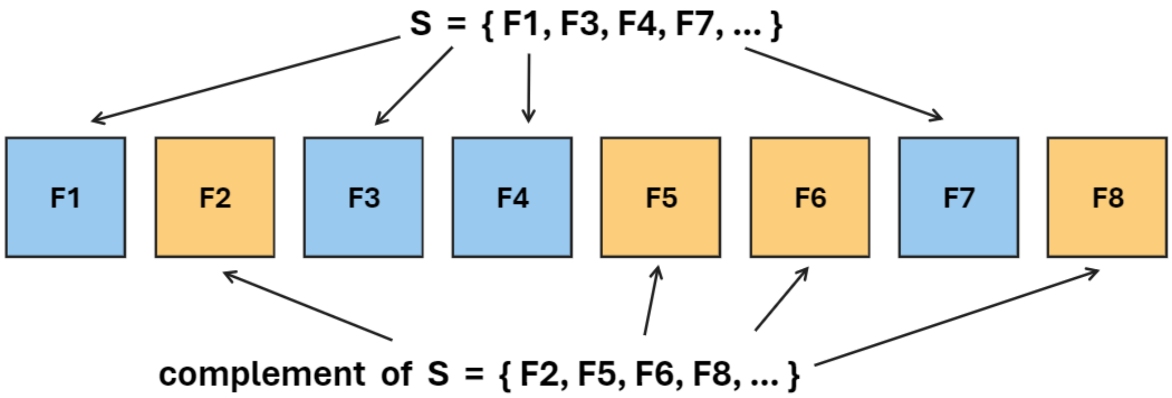
S = { F1, F3, F4, F7, ... }
F1
F2
F3
F4
F5
F6
F7
F8
complement of S = { F2, F5, F6, F8, ... }

Figure 3.2: Combined Data Point from S and S's Complement

The term $\mathbb{E}(f(X))$ is included in the formula to ensure that $v_{x^i}(\emptyset) = 0$. If none of the features are selected for analysis, the integral reduces to the average prediction over all data points in X. The feature values in set C are treated as random variables, with their distribution based on $X \backslash \{x^i\}$.

The marginal contribution is then expressed as:

$$v_{x^i}(S \cup \{j\}) - v_{x^i}(S) = \int f(x^i_{S \cup \{j\}} \cup x_{C \backslash \{j\}}) dx_{C \backslash \{j\}} - \int f(x^i_S \cup x_C) dx_C \tag{3.4}$$

Thus, according to the Shapley Value formula in game theory, SHAP is expressed as:

$$\phi_j^i = \sum_{S \subseteq U \backslash \{j\}} \frac{|S|!(p - |S| - 1)!}{p!} \left[\int f(x^{(i)}_{S \cup \{j\}} \cup x_{C \backslash \{j\}}) dx_{C \backslash \{j\}} - \int f(x^{(i)}_S \cup x_C) dx_C \right] \tag{3.5}$$

It represents the average marginal contribution of a feature value $x_j^{(i)}$ to all possible coalitions of features. Additionally, it continues to satisfy the efficiency property: [13, p. 35]

$$\sum_{j=1}^{p} \phi_j^i = f(x^i) - \mathbb{E}[f(X)] \tag{3.6}$$

The sum of the Shapley Values for data point $x^{(i)}$ equals the difference between the prediction for $x^{(i)}$ and the average prediction for all data points in X.

However, in practice, we do not have the exact distribution of x_C. Additionally, if there are p features, the number of possible coalitions is 2^p, making direct computation intractable.

Therefore, ϕ_j^i is typically estimated using Monte Carlo simulation. The value function is then expressed as follows: [13, pp. 38–43]

$$\hat{v}_{x^i}(S) = \frac{1}{n} \sum_{k=1}^{n} \left[f\left(x_S^i \cup x_C^k\right) - f\left(x^k\right) \right] \tag{3.7}$$

Where n represents the number of samples drawn from the dataset. The marginal contribution is then calculated as:

$$\hat{v}_{x^i}(S \cup \{j\}) - \hat{v}_{x^i}(S) = \frac{1}{n} \sum_{k=1}^{n} \left[f\left(x^i_{S \cup \{j\}} \cup x^k_{C \setminus \{j\}}\right) - f\left(x^i_S \cup x^k_C\right) \right] \tag{3.8}$$

Thus, the SHAP value is given by the following formula:

$$\hat{\phi}^i_j = \sum_{S \subseteq U \setminus \{j\}} \frac{|S|!(p - |S| - 1)!}{p!} \left[\frac{1}{n} \sum_{k=1}^{n} \left[f\left(x^i_{S \cup \{j\}} \cup x^k_{C \setminus \{j\}}\right) - f\left(x^i_S \cup x^k_C\right) \right] \right] \tag{3.9}$$

3.2 Model Analysis: Practical Part

3.2.1 SHAP Process and Transformation

From the theoretical foundation of SHAP, we know that for each data record, the sum of the SHAP values for all features equals the difference between the prediction for that record and the mean prediction across the population. However, in this SHAP project, an exponentially transformed Shapley value is used. This transformation is necessary because a log-link function is employed in the GLM. By applying a similar transformation, it becomes possible to make direct comparisons between the SHAP values and the beta coefficients in the GLM.

$$\text{SHAP}^i_j = e^{\phi^i_j} \tag{3.10}$$

Instead of linking the sum of the Shapley values with the prediction, this transformed Shapley value links the product of the transformed Shapley values to the prediction.

$$\text{For each data point } x^i : \prod_j \text{SHAP}^i_j = e^{\sum_j \phi^i_j} = e^{f(x^i) - \mathbb{E}[f(X)]} \tag{3.11}$$

The conclusion we draw is as follows: if we multiply all transformed SHAP values, we obtain $\exp(\text{final prediction} - \text{base})$, where the base represents the average prediction across all data inputs. This is referred to as the "base" because, similar to the base level in a GLM, everything else is an adjustment relative to it. In this SHAP project, the base is calculated using 479,506 data inputs, with a value of 0.5781 (slightly altered for confidentiality purposes).

$$\text{base} = \mathbb{E}[f(X)] = \frac{1}{479,506} \sum_{i=1}^{479,506} f(x^i) = 0.5781 \tag{3.12}$$

3.2.2 Mechanism of the Analysis

Based on the previous formula, the prediction can be expressed as:

$$f(x^i) = e^{\text{base}} \prod_j \text{SHAP}_j^i \tag{3.13}$$

The input data has dimensions of $479,506 \times 36$, and the SHAP process will compute a corresponding matrix of Shapley values with the same size, assigning one value for each position in the input matrix. For each row in the matrix, the relational formula described above holds true.

The process can be illustrated by the following graph:

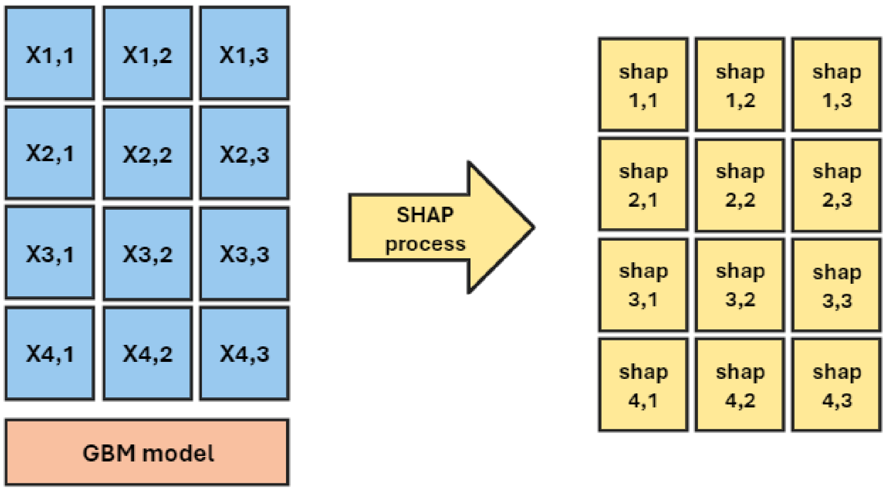

Figure 3.3: SHAP process

The most important assumption of this project is:

- **The SHAP values for different levels within each feature are same:**

$$\phi_j^i = \phi_j \quad \text{for each i.} \tag{3.14}$$

This implies that the SHAP values for features should not depend on specific data inputs; the SHAP values generated by different data inputs are simply different estimates of the same underlying true SHAP value. However, the true SHAP value itself should remain constant. For example, the effect of being a 46-year-old driver named Peter on the premium should be the same as the effect of being a 46-year-old driver named Mike on the premium when considering the effect of age.

Based on this assumption, several lookup tables are constructed. For continuous features, such as the market value of the car, their ranges are divided into multiple intervals. These intervals are aligned with the divisions used in the tree model during training. For discrete features, each distinct value represents a level. For each level in the discrete features and each interval in the continuous features, the mean, variance, and count of the SHAP values within those levels are calculated. As a result, each feature is associated with a lookup table containing four columns: the first column represents the levels or intervals, while the second, third, and fourth columns contain the count, mean, and variance of the SHAP values for each respective category.

For each data input row, a new prediction is calculated as follows:

$$\hat{f}(x^i)_{\text{SHAP}} = e^{\text{base}} \prod_j \text{SHAP}^{\text{lookup}}(x_{i,j}) \tag{3.15}$$

Where $\text{SHAP}^{\text{lookup}}(x_{i,j})$ represents the Shapley value found using the lookup table, based on the value of feature j for data input x_i. Next, we compare $f(x^i)$ and $\hat{f}(x^i)_{\text{SHAP}}$, and calculate their relative difference:

$$\textbf{Relative Difference:} \quad \frac{f(x^i) - \hat{f}(x^i)_{\text{SHAP}}}{f(x^i)} \tag{3.16}$$

3.2.3 Results of Using SHAP

Overall, the results of this experiment are quite satisfactory.

- **The sum of all ML prediction:** 840486.97
- **The sum of all SHAP prediction:** 837462.46
- **Relative difference of the total sum:** 0.36%
- **The average of the absolute value of the relative difference:** 6.9285%
- **The average of the relative difference:** 0.5155%
- **The standard deviation of the of the relative difference:** 0.0935

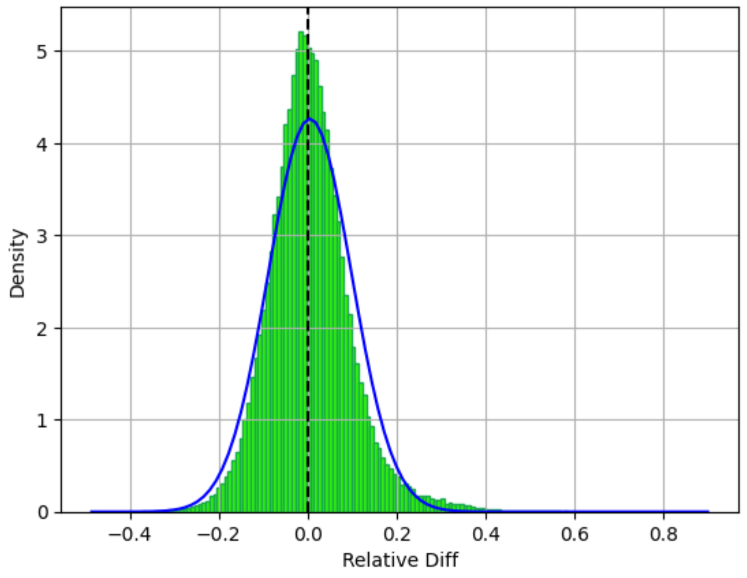

Figure 3.4: Histogram of Relative Differences

The blue line represents a normal distribution with a mean of 0.5155% and a standard deviation of 0.0935 . As shown in the histogram, the relative difference exhibits a bell-shaped pattern.

Due to confidentiality considerations, only a subset of the SHAP plots for certain features is presented below, and the x-axis labels have been removed in some plots.

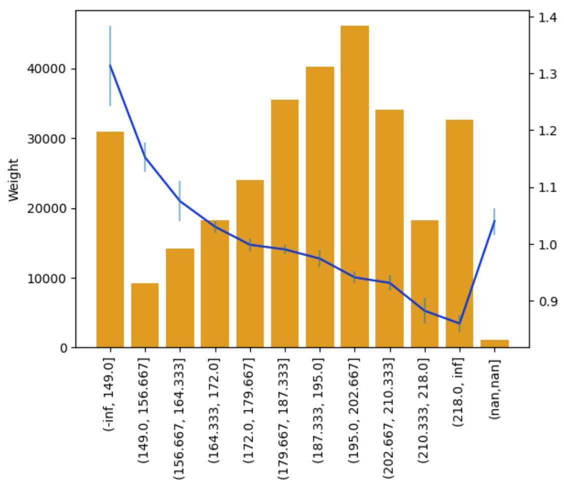

Figure 3.5: SHAP of F-3

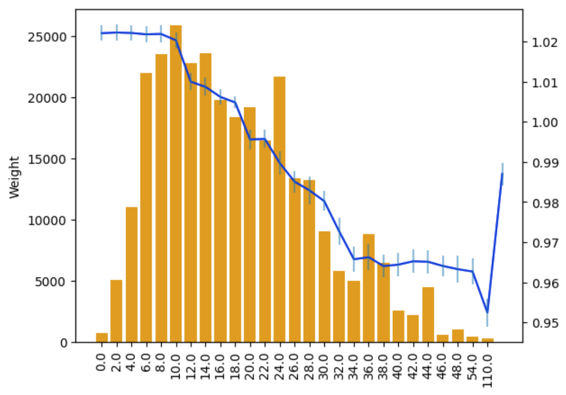

Figure 3.6: SHAP of F-5

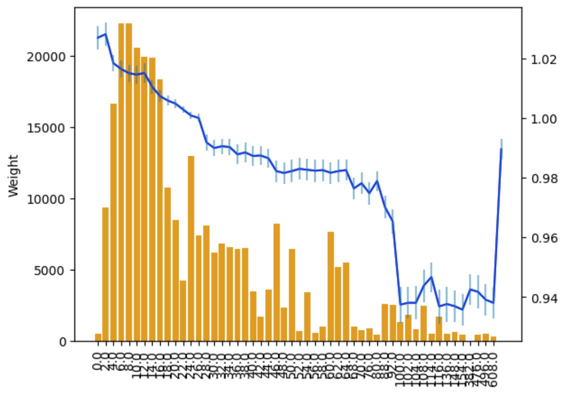

Figure 3.7: SHAP of F-9

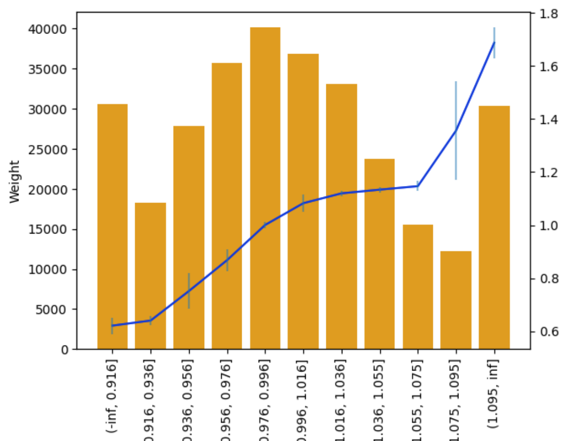

Figure 3.8: SHAP of F-10

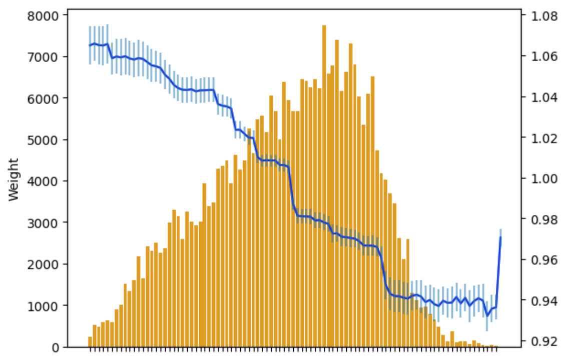

Figure 3.9: SHAP of F-11

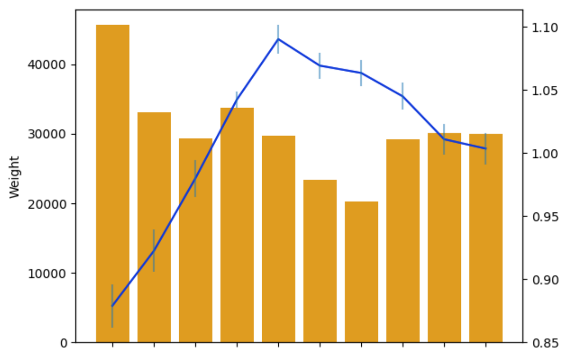

Figure 3.10: SHAP of F-13

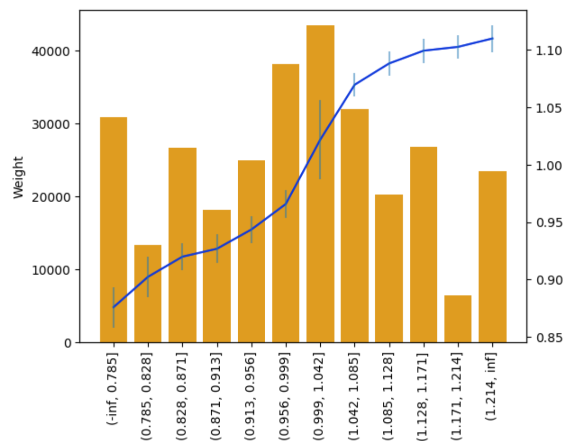

Figure 3.11: SHAP of F-14

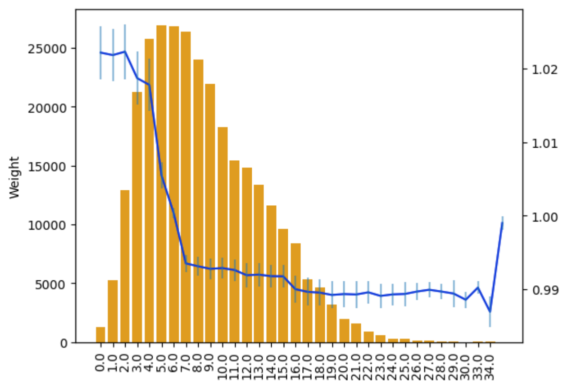

Figure 3.12: SHAP of F-15

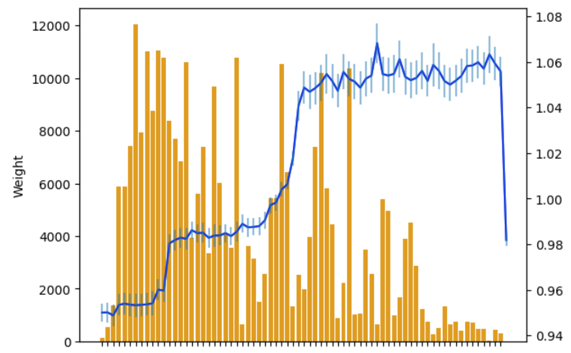

Figure 3.13: SHAP of F-16

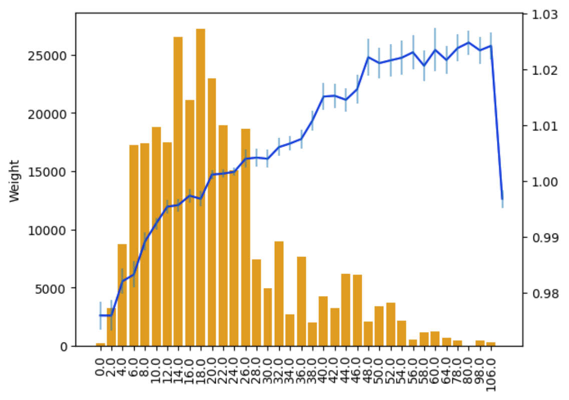

Figure 3.14: SHAP of F-17

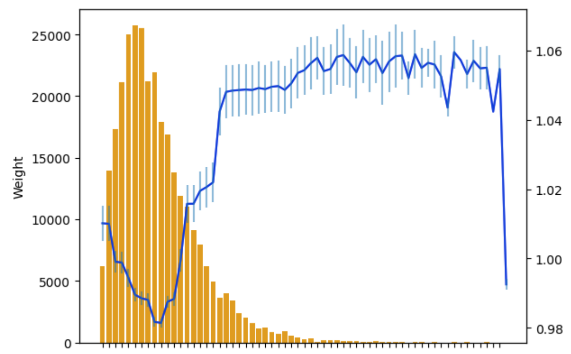

Figure 3.15: SHAP of F-24

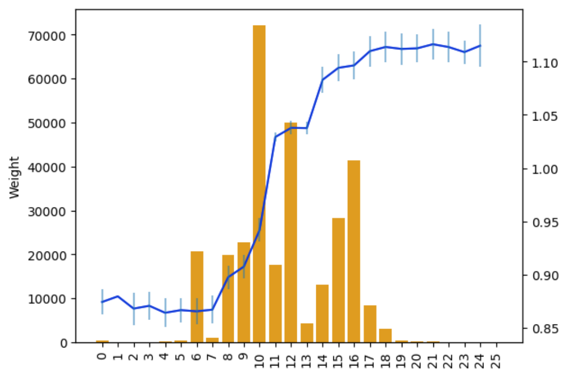

Figure 3.16: SHAP of F-25

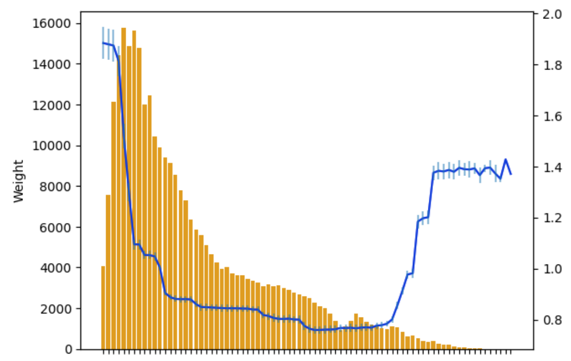

Figure 3.17: SHAP of F-26

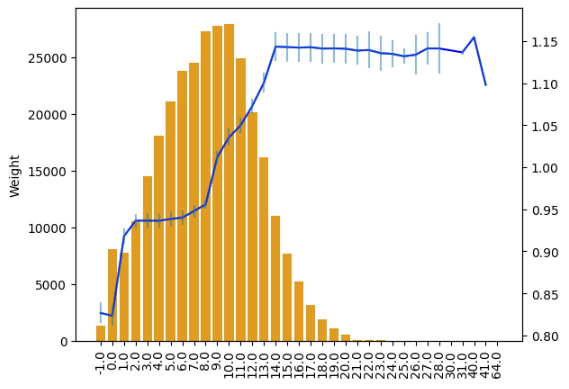

Figure 3.18: SHAP of F-28

The orange bars represent the exposure (the count) of data inputs at each level, while the blue line shows the mean SHAP value for each level. The light blue vertical lines illustrate the range of SHAP values, from the minimum to the maximum. In some plots, a turning point appears on the right side due to the placement of the "unknown" level in that position.

Since the SHAP values here are exponentially transformed, a SHAP greater than 1 indicates that this level within the feature has a positive effect on the prediction, while a SHAP less than 1 indicates a negative effect. From the plots above, we can observe a clear relationship between the SHAP values and the levels within several features. For example, the SHAP value consistently increases as the value of F-10 increases, and it continuously decreases as the value of F-3 increases. These insights help us to gain a deeper understanding of the GBM model.

In addition to examining the relationships between SHAP values and levels, the average absolute relative differences were also analyzed:

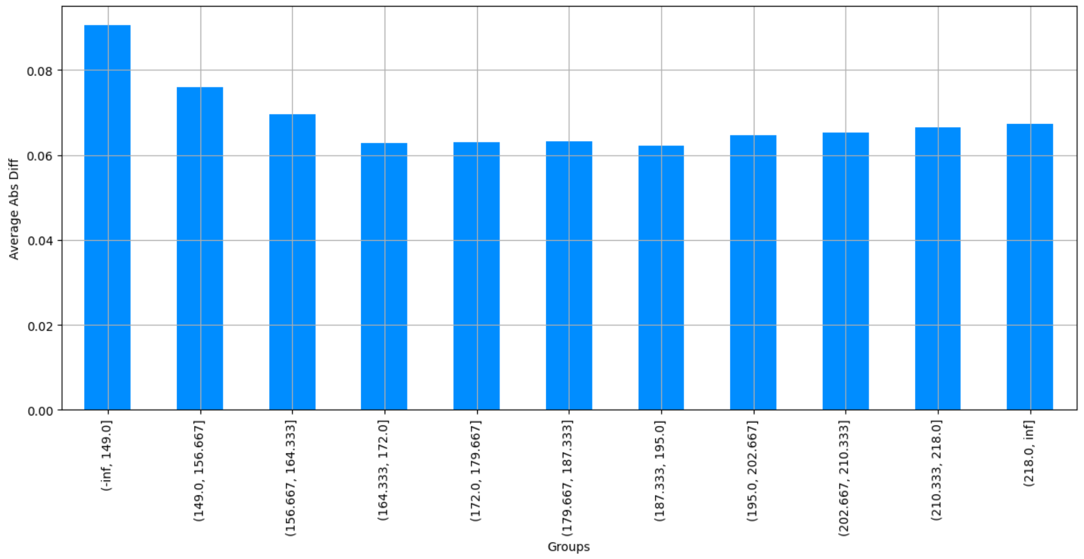

Figure 3.19: Average Absolute Relative Difference for F-3

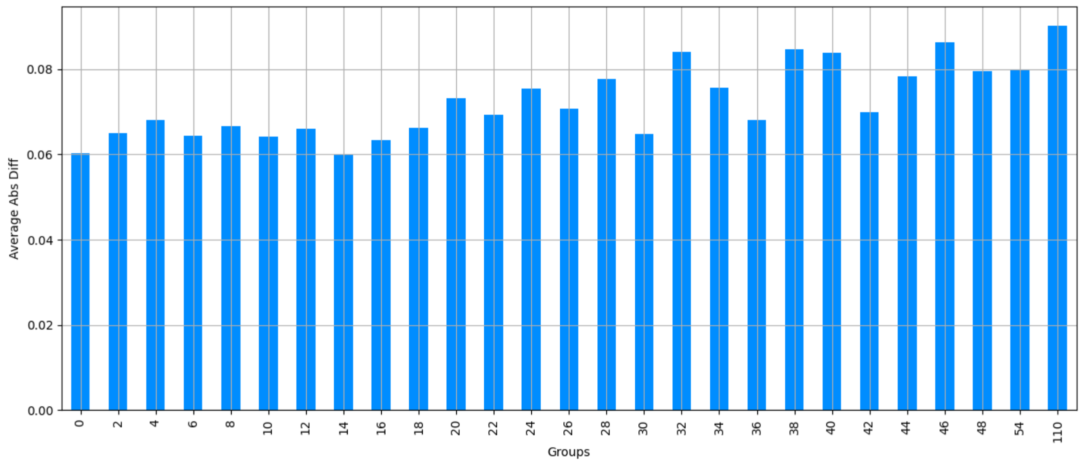

Figure 3.20: Average Absolute Relative Difference for F-5

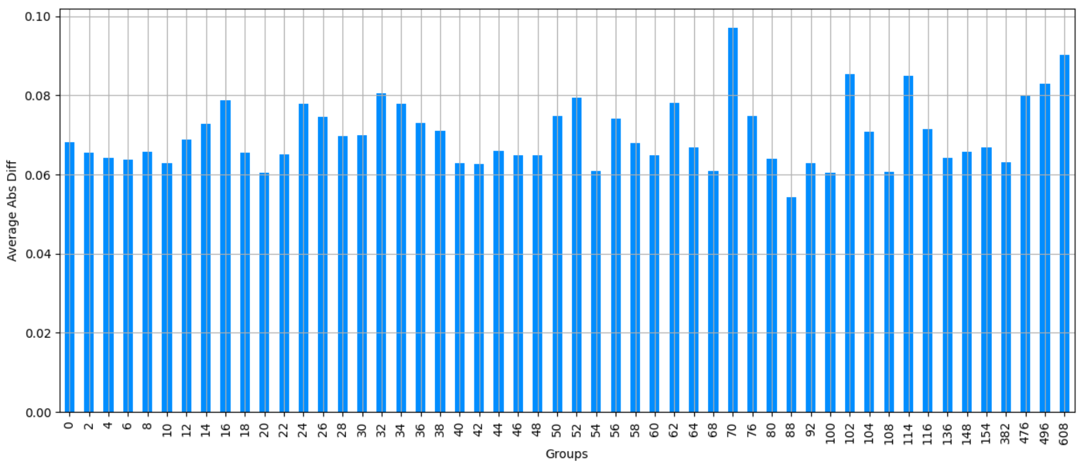

Figure 3.21: Average Absolute Relative Difference for F-9

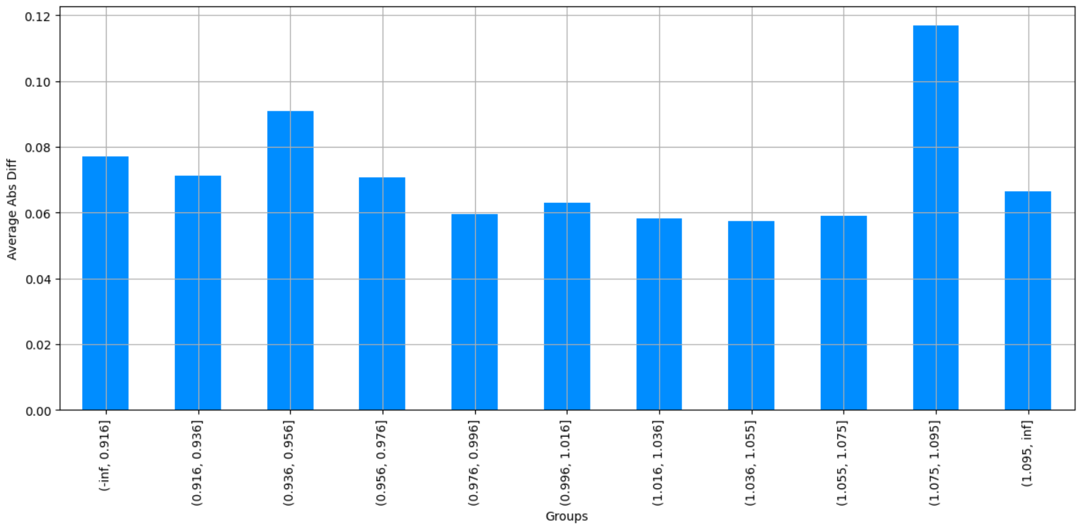

Figure 3.22: Average Absolute Relative Difference for F-10

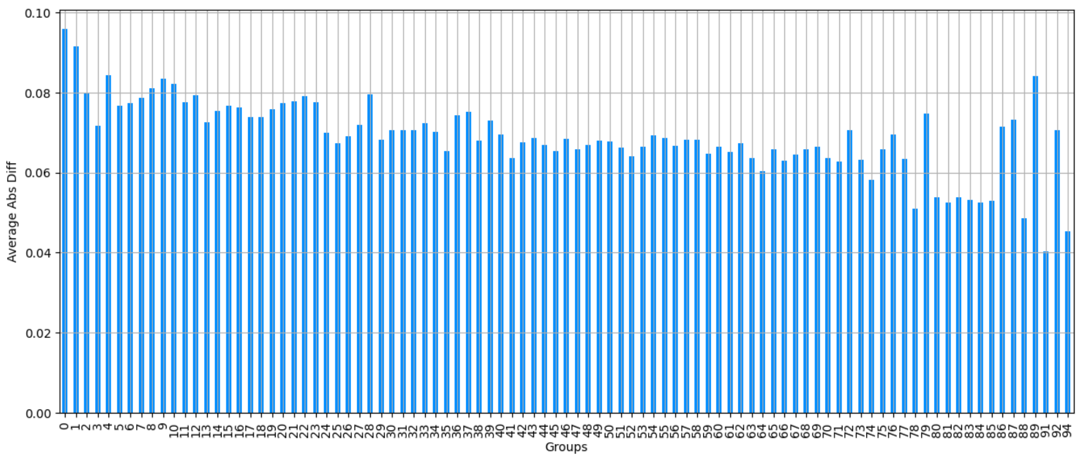

Figure 3.23: Average Absolute Relative Difference for F-11

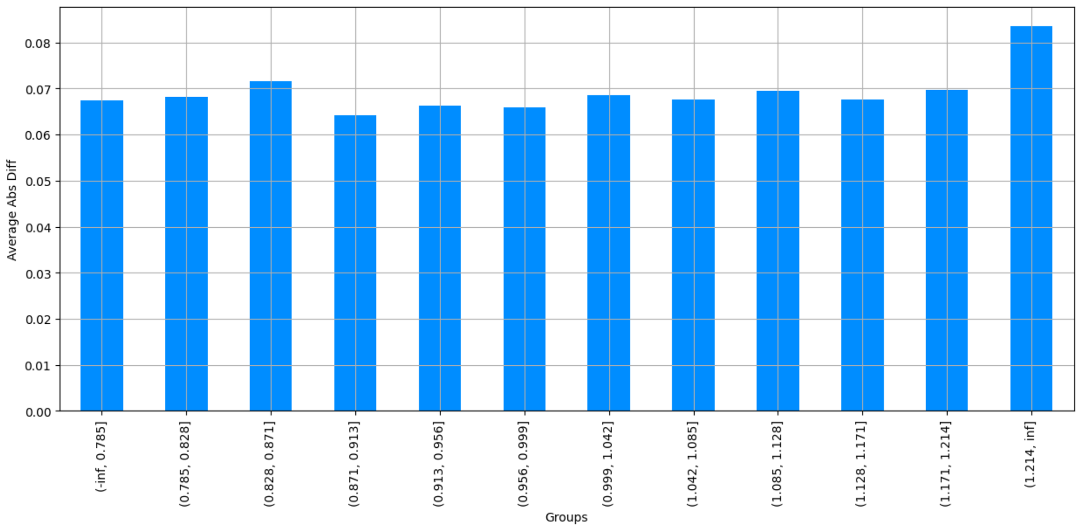

Figure 3.24: Average Absolute Relative Difference for F-14

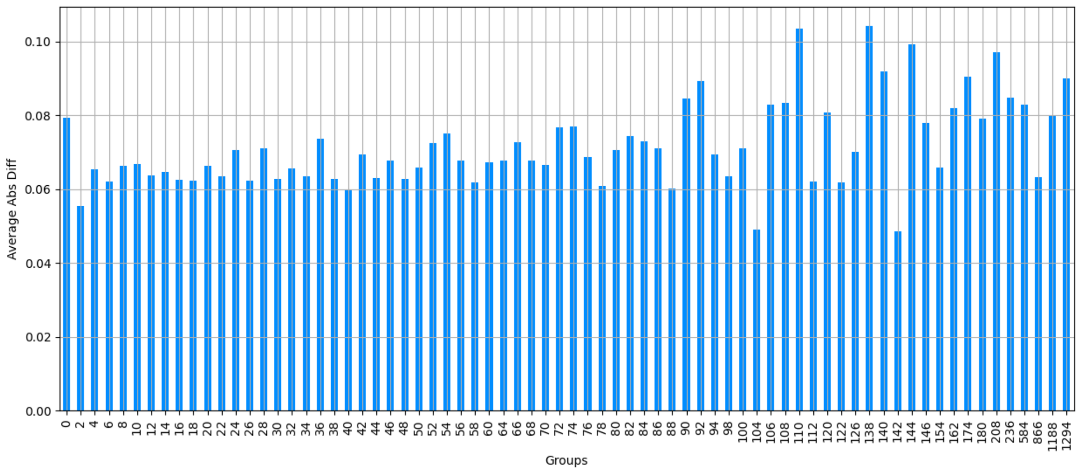

Figure 3.25: Average Absolute Relative Difference for F-16

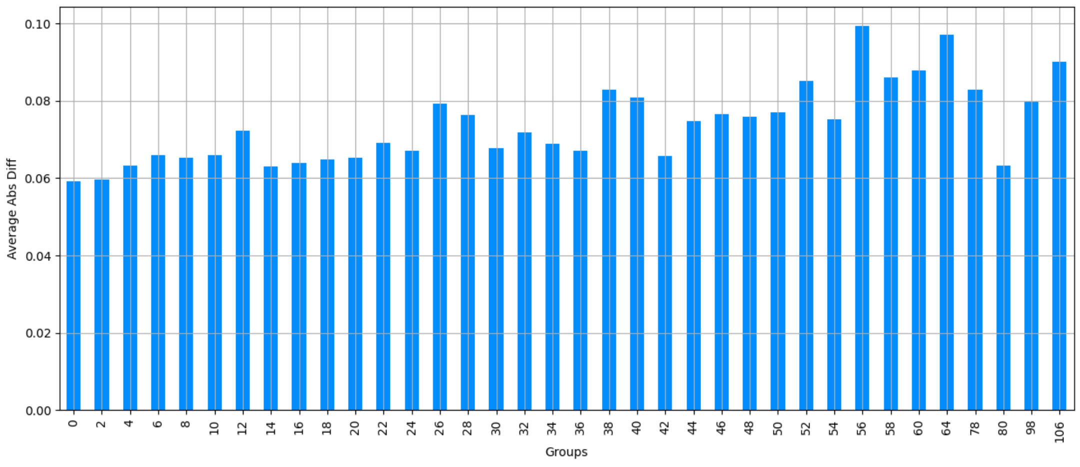

Figure 3.26: Average Absolute Relative Difference for F-17

As observed from the plots, the average absolute relative differences are nearly the same across all levels within each feature, except for level 56 in F-17. However, when examining its exposure, we find that level 56 in F-17 has significantly lower exposure compared to other levels within the same feature. Based on this, we can conclude that:

- When the exposures of the levels are large enough, the average absolute relative differences are nearly the same across all levels.
- If we consider the original prediction as the "true value" and the value generated by the SHAP method as the "predicted value", the accuracy is relatively low. Almost every level exhibits an average absolute relative difference greater than 6% but less than 12%.

Based on the results above, the purpose of this project has been achieved and our initial assumption $\phi_j^i = \phi_j$ is difficult to reject because:

1. For each level within every feature, the corresponding SHAP values fall within a narrow range.
2. When comparing the original prediction with the prediction generated by the SHAP lookup table, the relative difference in the total sum is notably low.

Therefore, the methodology used to interpret GLM models with betas can be effectively applied to interpret the GBM model using SHAP values. This approach allows the company to have greater confidence in using the model and explaining its results to the board, regulators, and customers. Although the GBM tree is complex due to the division of high-dimensional spaces, SHAP provides a straightforward way to uncover the inner workings of this "black box".

Chapter 4

Interpreting using PDP

4.1 Introduction of PDP

This theoretical section on PDP follows the explanation of the partial dependence plot in Molnar's book [17].

In addition to SHAP, PDP is another model-agnostic method used to interpret machine learning models. It illustrates the marginal effect of features on the predictions of a model. The purpose of this chapter is to explore an alternative method to validate the conclusions drawn in the previous chapter. The partial dependence function is defined as: [17, p. 113]

$$\hat{f}_S(x_S) = \mathbb{E}\left[f(x_S \cup x_C)\right] = \int f(x_S \cup x_C)dx_C \tag{4.1}$$

Let S be the feature of interest, represented on the x-axis in the PDP, and let C represent the other features in the model. The data point $x_S \cup x_C$ is constructed by assigning the value x_S to the position of the feature S and assigning the values from x_C, a $(n-1) \times 1$ vector, to the positions of features in C. The integration is performed over all possible values of x_C.

Similar to SHAP, in practice, the Monte Carlo method is used to calculate PDP. The formula is as follows:

$$\hat{f}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n} f(x_S \cup x_C^{(i)}) \tag{4.2}$$

Here, n represents the number of instances in the dataset, which in this case is 479,506, and $x_C^{(i)}$ are the actual feature values for the feature set C from the dataset. In this project, the Python package scikit-learn is used to calculate the PDP and generate the corresponding plots. Although Monte Carlo simulation is used, the computational load remains significant due to the large size of the data set, making it possible to obtain results only on the Cortex platform.

The process of calculating the partial dependence function is illustrated in the following figure. To calculate the partial dependence function for the “pear” level of feature one, the value of feature one is replaced by “pear” for all other data inputs. The average of the generated predictions is then used as the partial dependence value. By averaging over the entire population, the process effectively approximates the integral.

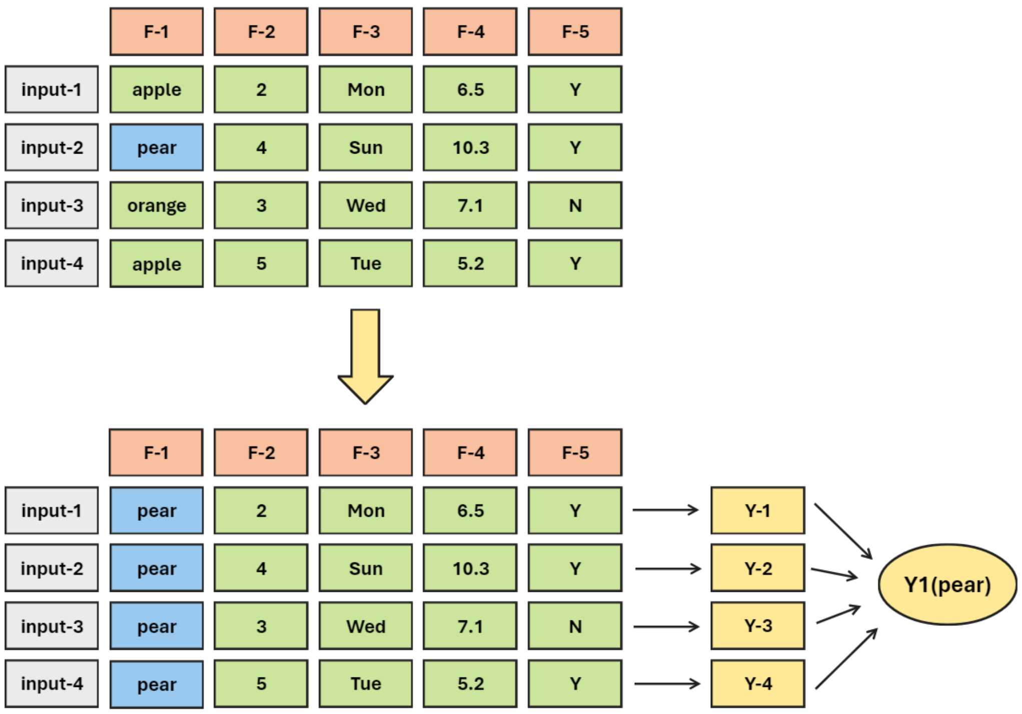

Figure 4.1: Calculation Process of the PDP

4.2 PDP Results

Some comparisons between SHAP and PDP are listed below:

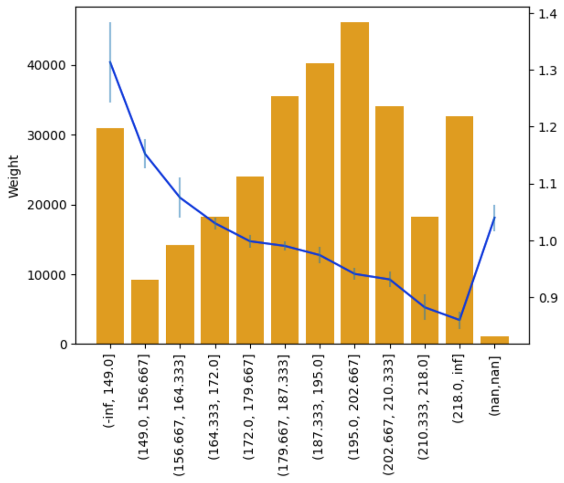

Figure 4.2: SHAP of F-3

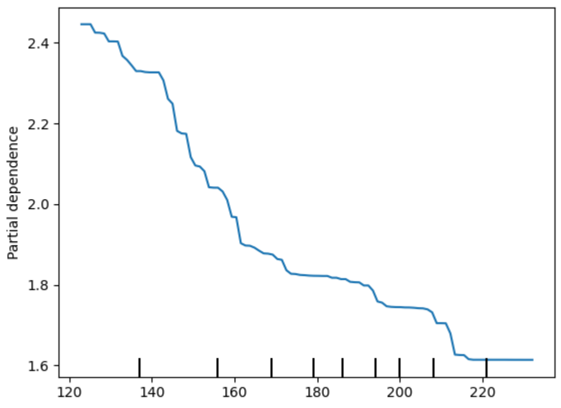

Figure 4.3: PDP of F-3

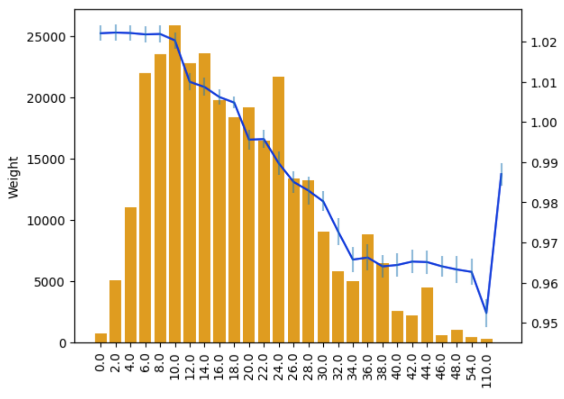

Figure 4.4: SHAP of F-5

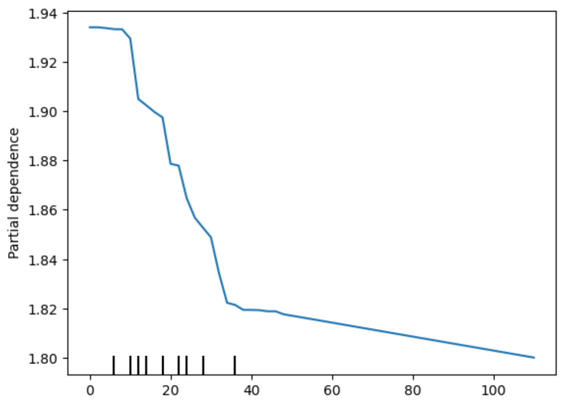

Figure 4.5: PDP of F-5

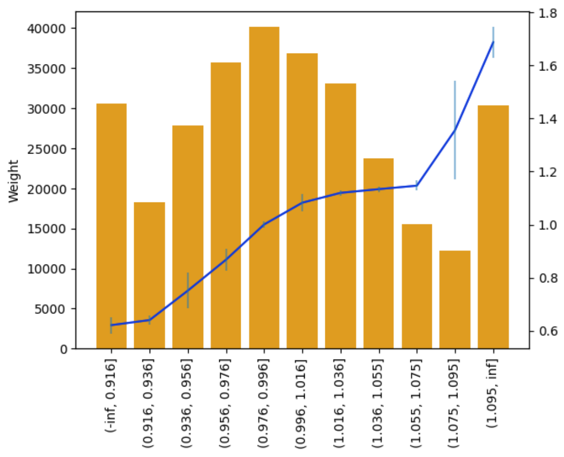

Figure 4.6: SHAP of F-10

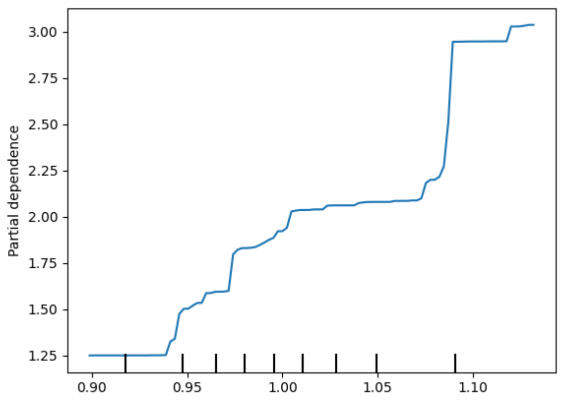

Figure 4.7: PDP of F-10

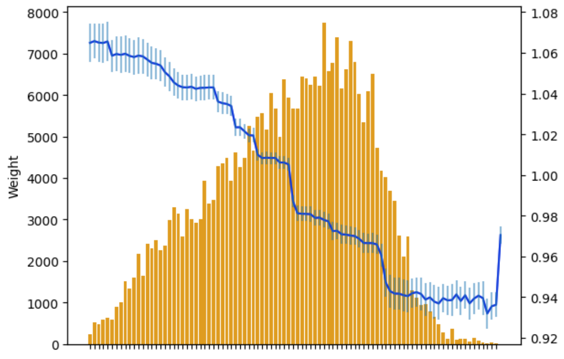

Figure 4.8: SHAP of F-11

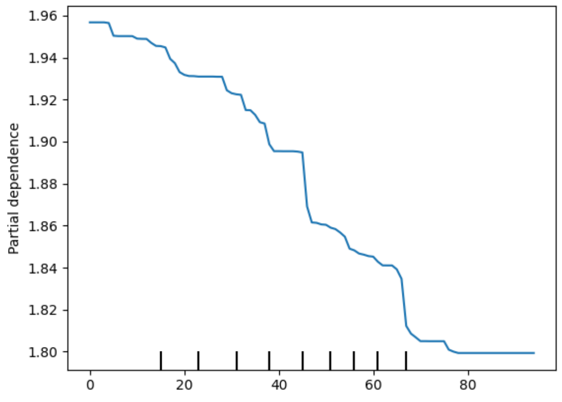

Figure 4.9: PDP of F-11

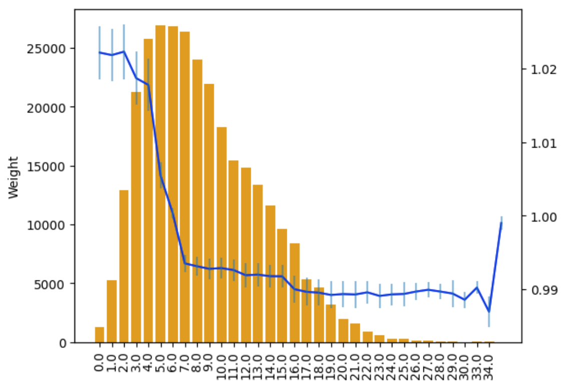

Figure 4.10: SHAP of F-15

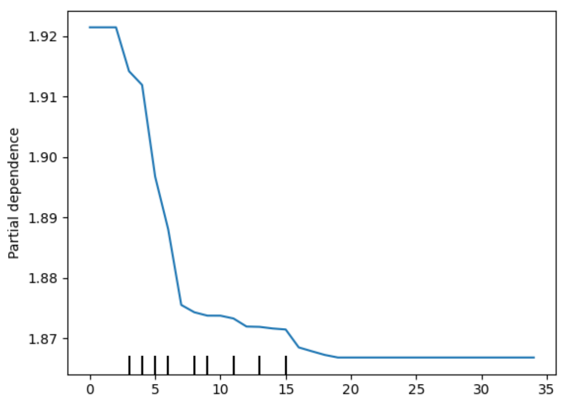

Figure 4.11: PDP of F-15

We observe that in these plots, the trends of SHAP values versus the feature and PDP values versus the feature are very similar. Since PDP offers an alternative way to interpret machine learning models using a completely different mechanism than SHAP, this similarity supports the conclusions drawn in the previous chapter. Both SHAP and PDP reveal the underlying relationship between the output and the features.

However, unlike SHAP, PDP does not provide information specific to individual data points and lacks the efficiency property that directly links the prediction to the value of the partial dependence function. As a result, PDP cannot generate new predictions in the way SHAP can.

Chapter 5

Conclusion

5.1 Result of the Analysis

The results of the predictions generated using the lookup table's SHAP values are reasonable, though not as accurate as expected. The average relative difference is close to zero and the standard deviation is small. However, the average relative differences remain around 6%. This suggests that the assumption we tested, $\phi_j^i = \phi_j$, holds to some extent. From Figures 3.5 to 3.18, the minimum and maximum SHAP values for each level within each feature are relatively close, making it reasonable to assume the existence of a true SHAP value for each level of each feature in the model. The SHAP values we obtain can be viewed as estimates of this true value, similar to how we estimate population means in traditional statistics.

There is potential for further improvement in this project, but due to the time constraints of this internship, these enhancements can only be proposed and not implemented at this time.

- For example, when dividing the range of continuous features into small intervals, we used the same partitioning method as the tree model. However, dividing the range into smaller intervals might yield better results.
- In this project, the arithmetic mean was used to calculate the average SHAP values. However, using the geometric mean might produce more accurate results, given that an exponential transformation was applied. This proposal seems reasonable and worth exploring.

5.2 Potential Future Investigation

In addition to the potential improvements, there are other topics that could be explored further.

- If there is only one level for certain features, would the SHAP process assign the same SHAP value to every row for these features? And would this value be equal to 1? Since having only one level means that this feature does not influence the prediction (as all data inputs are the same for that feature), this situation resembles the "Dummy" property of SHAP described in Molnar's book [13]. The "Dummy" property states that adding an irrelevant feature to the SHAP process results in a SHAP value of 0 for every level within that feature. However, what happens if the model was trained on feature A with five levels, but during the SHAP process the input data for feature A contains only one level?

- The ensemble methods combine the predictions of different machine learning algorithms, and the SHAP process can generate SHAP values for the final ensemble model. Additionally, the SHAP process can produce SHAP values for each sub-model within the ensemble. The question arises: is the SHAP value of the ensemble model equal to the weighted combination of the SHAP values of the sub-models, using the same weights as in the ensemble method? If not, which of these two SHAP approaches would perform better based on the analysis in this project?

- Not only should the difference between the original prediction and the SHAP-based prediction be calculated, but also the difference between the SHAP-based prediction and the true value should be calculated and analyzed. This approach could lead to further insights and additional research avenues.

- Could the process of generating SHAP-based predictions be considered a new machine learning method? Additionally, is there a possibility that these SHAP-based predictions could even outperform the original predictions compared to the true observations?

- If it is not possible, it may be because the information fed into the machine learning process is more "original" than the information used in the SHAP process. In the permutation step of the SHAP process, if combinations $x_S^i \cup x_C$ exist in the input data or training data, we could use the true values rather than the predicted values to calculate an alternative version of SHAP. Would this approach yield better results than the analysis in Section 3.2.3?

- It is possible to plot the betas and SHAP values on a single graph for comparison, provided the betas are also exponentially transformed and the SHAP baseline is properly adjusted. In such plots, the transformed betas and SHAP values should exhibit a similar trend. The process for doing this is straightforward: instead of using the GBM, we can incorporate a GLM into the analysis in Section 3.2.2. After adjusting the baselines in both SHAP and GLM, a direct comparison becomes feasible.

List of Figures

List of Abbreviations

AD Accidental Damage. 4

FT Fire and Theft. 4

GBM Gradient Boosting Machine. 2

Generali Group Assicurazioni Generali S.p.A.. 1

GLM Generalized Linear Model. ii

IFoA Institute and Faculty of Actuaries. i

ISEG Instituto Superior de Economia e Gestão. i

LC Loss Cost. 8

Liberty Seguros Liberty Seguros, Compañía de Seguros y Reaseguros, S.A.. 1

LMIG Liberty Mutual Insurance Group. 1

ML Machine Learning. ii

NB New Business. 4

PDP Partial Dependence Plot. ii

RedClick Liberty Ireland. 3

SHAP SHapley Additive exPlanations. ii

TPD Third Party Liability: Property Damage. 4

TPI Third Party Liability: Personal Injury. 2

WTW Willis Towers Watson. 11

Bibliography

[1] Assicurazioni Generali S.p.A. *Generali completes the acquisition of Liberty Seguros from Liberty Mutual*. URL: https://www.generali.com/media/press-releases/all/2024/Generali-completes-the-acquisition-of-Liberty-Seguros.

[2] Central Bank of Ireland. *Review of Differential Pricing in the Private Car and Home Insurance Markets - Final Report and Public Consultation*. 2021. URL: https://www.centralbank.ie/docs/default-source/publications/consultation-papers/cp143/differential-pricing-review---final-report-and-public-consultation.pdf.

[3] RedClick. *Car Insurance*. URL: https://www.redclick.ie/car-insurance.

[4] RedClick. *Car Insurance Policy Booklet*. URL: https://www.redclick.ie/sites/redclick/files/documents/Car_Policy-Booklet_GDE_RCECOQMP0524.pdf.

[5] RedClick. *Insurance Product Information Document*. URL: https://www.redclick.ie/sites/redclick/files/documents/RedClick_Motor_IPID_GDE_0524.pdf.

[6] Andrea Treviño Gavito, Diego Klabjan, and Jean Utke. *Gradient-Boosted Based Structured and Unstructured Learning*. 2023. DOI: https://doi.org/10.48550/arXiv.2302.14299.

[7] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R (2nd ed.)* Springer Texts in Statistics, 2021.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.)* Springer Series in Statistics, 2009.

[9] Cheng Li. *A Gentle Introduction to Gradient Boosting*. URL: https://www.chengli.io/tutorials/gradient_boosting.pdf.

[10] Peter K. Dunn and Gordon K. Smyth. *Generalized linear models with examples in R*. Springer Texts in Statistics, 2018.

[11] Microsoft Corporation. *Parameters — LightGBM 4.5.0.99 documentation*. 2024. URL: https://lightgbm.readthedocs.io/en/latest/Parameters.html.

[12] Paula Rooney. *AI in the cloud pays dividends for Liberty Mutual*. 2022. URL: https://www.cio.com/article/350368/ai-in-the-cloud-pays-dividends-for-liberty-mutual.html.

[13] Christoph Molnar. *Interpreting Machine Learning Models With SHAP: A Guide With Python Examples And Theory On Shapley Values*. Independently published, 2023.

[14] Scott Lundberg. *Welcome to the SHAP documentation*. 2018. URL: https://shap.readthedocs.io/en/latest/_images/shap_header.png.

[15] Lloyd Stowell Shapley. “17. A Value for n-Person Games”. In: *Contributions to the Theory of Games* II (1953), pp. 307–317. DOI: 10.1515/9781400881970-018.

[16] Scott Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *31st Conference on Neural Information Processing Systems* (2017). DOI: 10.48550/arXiv.1705.07874.

[17] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Independently published, 2019.