UNIVERSITAT ROVIRA i VIRGILI

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

UNIVERSITAT DE BARCELONA

Artificial Intelligence Seminar

SHAP & Shapley Values

Explainable AI

Hajar Lachheb & Clea Han

Table of contents

Introduction

Artificial Intelligence (AI) has revolutionized numerous industries by enabling machines to perform complex tasks that were once exclusive to human intelligence. As AI systems become more prevalent, there is an increasing need for transparency and interpretability to understand the reasoning behind their decisions. The interpretability of AI models is crucial for building trust, identifying biases, and ensuring accountability. In this report, we delve into the world of SHAP (SHapley Additive exPlanations) values and Shapley values, two powerful techniques that contribute to the interpretability of AI models.

The interpretability of AI models has traditionally been a challenge due to their complexity and the "black-box" nature of many machine learning algorithms. SHAP values and Shapley values offer a solution to this challenge by providing insights into the importance and contributions of individual features in the decision-making process. These techniques allow us to understand the impact of each feature on the model's output, unravel complex interactions between features, and explain the reasoning behind the model's predictions. The concept of Shapley values originates from cooperative game theory and has been adapted to the context of machine learning. It quantifies the marginal contribution of each feature by considering all possible feature subsets and their corresponding predictions. By calculating Shapley values, we can assign a fair and interpretable measure of importance to each feature, providing a comprehensive understanding of their influence on the model's output.

SHAP values, on the other hand, build upon the foundation of Shapley values and offer a unified framework for explaining the output of AI models. SHAP values take into account both the main effects and interaction effects between features, resulting in a more comprehensive understanding of feature importance. These values provide a holistic view of feature contributions, enabling us to not only understand the relative importance of each feature but also how features interact with each other to shape the model's predictions.

In this report, we will explore the underlying concepts and methodologies behind SHAP values and Shapley values. We will examine their applications in various domains, including healthcare, finance, and image recognition, highlighting their contributions to interpretability, fairness, and bias detection. Additionally, we will discuss the challenges and limitations associated with these techniques, as well as the ongoing research and advancements in the field. By delving into the world of SHAP values and Shapley values, we aim to provide a comprehensive understanding of these techniques and their potential for enhancing the interpretability of AI models. Through their application, we can gain valuable insights into the decision-making process of AI systems, leading to increased transparency, trust, and responsible deployment of AI in real-world applications.

I. Literature Review and Analysis of bibliographical resources

1. Literature Review

The area of explainable AI has attracted a lot of interest recently with the goal of addressing the "black box" aspect of machine learning models and revealing how they make decisions. In this context, Shapley values and SHAP (SHapley Additive exPlanations) have become well-known methods for deciphering and elucidating the predictions provided by AI models. In this overview of the literature, we look at a number of studies that explore the uses, approaches, and developments of SHAP and Shapley values.

The main publications that have helped people comprehend and develop SHAP and Shapley values are summarized in the comparative table that follows. Each work provides insightful information on a variety of topics, including interpretability, feature attribution, model explanation, and their use in certain fields including credit scoring, natural language processing (NLP), and compound potency prediction.

Title	Authors	Year	Main Domain	Other aspects
A Unified Approach to Interpreting Model Predictions	Lundberg, S., & Lee, S.	**2017**	Interpretability	SHAP Values
Consistent Individualized Feature Attribution for Tree Ensembles	ındberg, S. M., Erion, G., Chen, H., DeGrave, A., ʻutkin, J. M., Nair, B., ... & Lee, S.	**2019**	Interpretability	Feature Attribution, Tree Ensembles
\n introduction to explainable AI with Shapley values	Lundberg, S.	**2018**	Explainable AI	Shapley Values
PSD2 explainable AI model for credit scoring	Llop et al.	**2020**	Explainable AI, Credit Scoring	Model Explanation, PSD2
Interpretation of machine learning odels using Shapley values: applicatior o compound potency and multi-target activity predictions	Rodríguez, Bajorath	**2020**	Interpretability	Shapley Values, Compound Potency
Ensembles of random SHAPs	Utkin, Konstantinov	**2021**	Interpretability	SHAP Values, Ensembles

AcME-Accelerated model-agnostic explanations: fast whitening of the Machine-Learning black box	Dandolo et al.	**2021**	Model Explanation	cME, Model-Agnostic Explanations
hapley variable importance clouds for interpretable machine learning	Ning et al.	**2021**	nterpretable Machine Learning	Shapley Values, /ariable Importance
SHAP-Based Explanation Methods: A Review for NLP Interpretability	Mosca et al.	**2022**	Natural Language Processing	SHAP-Based Explanation, NLP Interpretability

We hope to obtain a thorough grasp of the body of research that has already been done on SHAP and Shapley values by studying these works. The table offers a quick summary of the key areas, authors, years of publication, and additional pertinent sections included in each work. Researchers and practitioners interested in investigating the uses and developments of interpretable AI employing SHAP and Shapley values will find it to be a useful resource.

2. Analysis of bibliographical resources

After mentioning the articles we did work on in our literature review and mentioning the important sections included in the articles. We can do an analysis of the bibliographical resources which is in our case the articles provided to us.

"A Unified Approach to Interpreting Model Predictions" (Lundberg, S., & Lee, S., 2017): This seminal paper introduces the concept of SHAP values, providing a unified framework for interpreting model predictions. It lays the foundation for subsequent research in the field.

"Consistent Individualized Feature Attribution for Tree Ensembles" (Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... & Lee, S., 2019): This paper focuses on the interpretability of tree ensembles and proposes consistent individualized feature attribution techniques. It addresses the challenge of feature attribution in complex ensemble models.

"An Introduction to Explainable AI with Shapley Values" (Lundberg, S., 2018): This resource provides an overview of Shapley values and their applications in explainable AI. It offers a clear introduction to the concept and its relevance in understanding AI model decisions.

"PSD2 Explainable AI Model for Credit Scoring" (Llop et al., 2020): This paper explores the application of explainable AI using SHAP values in the context of credit scoring. It highlights how SHAP values can contribute to the transparency and interpretability of credit scoring models.

"Interpretation of Machine Learning Models using Shapley Values: Application to Compound Potency and Multi-Target Activity Predictions" (Rodríguez, Bajorath, 2020): This resource focuses on applying Shapley values to interpret machine learning models in the context of compound potency and multi-target activity predictions. It demonstrates the practical use of Shapley values in pharmaceutical research.

"Ensembles of Random SHAPs" (Utkin, Konstantinov, 2021): This paper explores the concept of ensembles of random SHAP values and their applications in interpretable AI. It investigates the benefits of using multiple SHAP explanations for model interpretation.

"AcME-Accelerated Model-Agnostic Explanations: Fast Whitening of the Machine-Learning Black Box" (Dandolo et al., 2021): This resource focuses on fast and efficient explanations for black-box models using the AcME framework. It presents a method for model-agnostic explanations, addressing the need for real-time interpretability.

"Shapley Variable Importance Clouds for Interpretable Machine Learning" (Ning et al., 2021): This paper introduces the concept of Shapley variable importance clouds, a novel approach for interpreting machine learning models. It provides a visual representation of variable importance, enhancing interpretability.

"SHAP-Based Explanation Methods: A Review for NLP Interpretability" (Mosca et al., 2022): This resource presents a comprehensive review of SHAP-based explanation methods for Natural Language Processing (NLP) models. It explores the application of SHAP values in NLP interpretability.

II. Teamwork Management

The preparation time for this seminar about XAI gives for each group is about 2 months, from the beginning of **march** to the end of **april**. For the organisation aspect, we had to use Notion and we divided the tasks and put in place a timeline as well.

We, in fact, divided our work on SHAP and Shapley values in **3 main steps** : Brainstorming for the first month of preparation, then in the meantime preparing the presentation from mid-march to mid-april and finalizing our presentation and doing implementations of SHAP values until the end of april.

Seminar Organisation

Filtrer Trier Nouvelle page

↗ Subject Timeline

Aa Nom	Dates	Type	Sujet
📌 Brainstorming #1	09/03/2023	Brainstorming	SHAP
✏️ Noting what was brainstormed & Reading articles	23/03/2023	Brainstorming	SHAP
📌 Brainstorming #2	26/03/2023	Brainstorming	SHAP
✏️ Organising the key points and categories in our presentation	04/04/2023	Working on the presentation	SHAP
💯 Starting our presentation	05/04/2023	Working on the presentation	SHAP
Implementation #1	10/04/2023 → 13/04/2023	Implementation	SHAP
Implementation #2	20/04/2023 → 25/04/2023	Implementation	SHAP
Implementation #3 (if needed, to correct and test)	24/04/2023 → 30/04/2023	Implementation	SHAP

Team management using Notion

Three independent phases made up the study we did on SHAP and Shapley values. During the first month of planning, a thorough brainstorming process was part of the initial phase. Then, from the middle of March until the middle of April, we concentrated all of our efforts on the careful planning of the presentation. This stage included acquiring pertinent material, putting essential ideas together, and developing a coherent story. Finally, to ensure the correctness and efficacy of our findings, we concentrated on the presentation's improvement and the application of SHAP values over the remaining weeks of April. We were able to properly investigate the nuances of SHAP and Shapley values and present our study in a thorough and cogent manner by adhering to this systematic strategy.

We have chosen six main elements for our presentation that will serve as the foundation of our debate. These points cover the significance of AI and the significance of interpretability, give an overview of SHAP and Shapley values, delve into the connection between SHAP values and feature importance, examine the function of SHAP values in model explanation, look at the difficulties and restrictions related to SHAP values, and conclude with future directions and perspectives.
We've divided the work up among us to ensure thorough coverage. One of us will concentrate on AI, the significance of interpretability, the elements connected to SHAP values and feature importance, as well as the difficulties and constraints associated with SHAP values. The other team member will examine the specifics of SHAP and Shapley values, investigate how SHAP values relate to model explanation, and wrap off the presentation by talking about the potential future applications and perspectives for SHAP values.

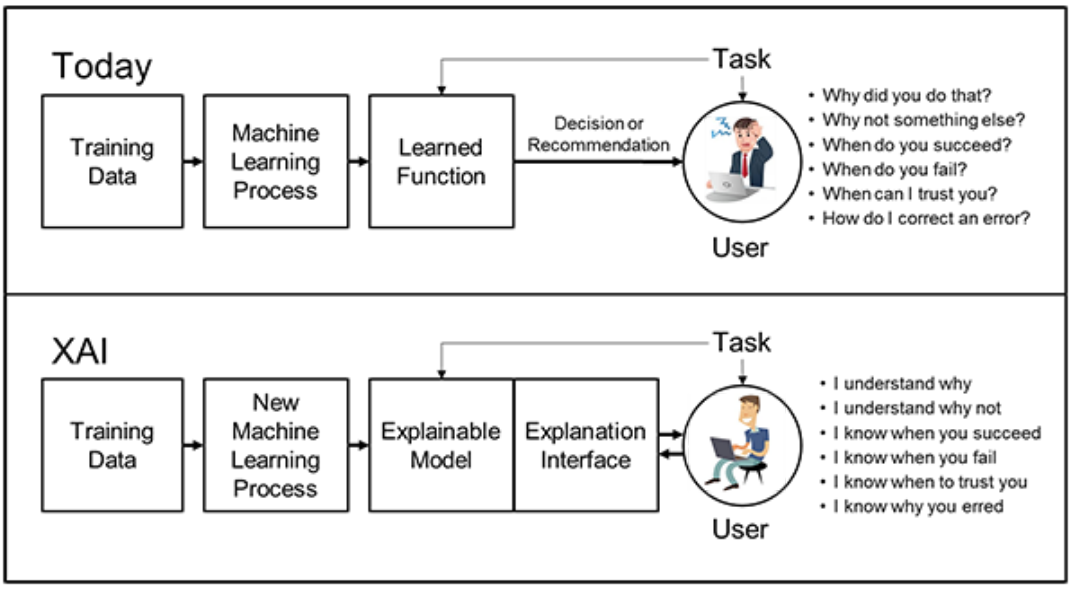
Each team member can learn more about SHAP and Shapley values while studying various aspects of its application and ramifications thanks to the division of labor. This strategy allows us to analyze SHAP and Shapley values comprehensively and coherently, highlighting their significance, promise, and the areas that require additional research.

III.AI & Importance of Interpretability

1. The notion of explainable AI

The study and practice of explainable AI (XAI) aims to improve the transparency and interpretability of artificial intelligence models and systems. The purpose of XAI is to give rational justifications for the choices and actions of AI systems so that people can understand and believe in the results that these systems create.

It can be difficult for users and stakeholders to understand how traditional AI models—such as deep neural networks—arrive at their predictions or conclusions because these models frequently function as "black boxes." The adoption and acceptability of AI systems may be significantly hampered by this lack of interpretability, particularly in crucial fields where explanations are necessary, like healthcare, finance, and legal applications. Explainable AI in the other hand, aims to overcome this difficulty by creating methods and procedures that make it possible for people to comprehend and analyze the workings and logic of AI models. It seeks to shed light on the characteristics, parameters, or elements that affect the model's results and their interrelationships. Users are able to understand the decision-making process, spot biases or errors, and verify the fairness, dependability, and accountability of AI systems thanks to XAI's clear explanations.



Explainable AI explanation

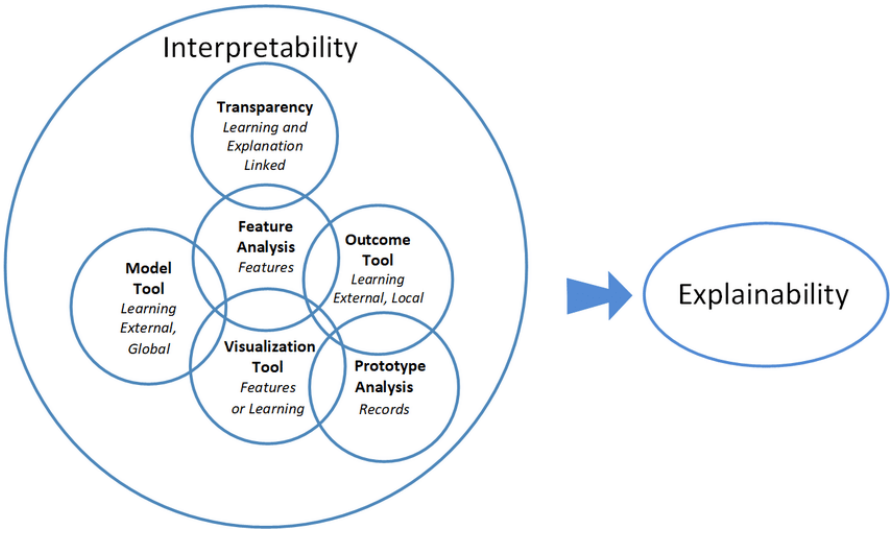
Rule-based systems, feature importance techniques, model-agnostic methodologies, and post-hoc interpretability are just a few of the approaches and techniques used in XAI. These methods try to draw out important information from AI models and present it in a way that anyone can understand. Examples of common XAI methods that provide feature attributions and explanations for specific predictions include SHAP (SHapley Additive exPlanations) and Shapley values.

More, applications and advantages of explainable AI are numerous. As users and stakeholders can verify and comprehend the logic behind the system's outputs, it can increase trust in AI systems. With the use of XAI, errors can be found, models may be improved, and any biases or flaws can be found and fixed. Additionally, XAI ensures adherence to ethical and regulatory

criteria in sensitive fields like healthcare or judicial systems, promoting accountability and justice.

2. Interpretability, Explainability and AI

AI, explainability, and interpretability are related ideas that seek to improve openness and comprehension in artificial intelligence systems. **Interpretability** is the capacity to understand and interpret the internal logic and decision-making of an AI model. Gaining knowledge of the model's data processing and decision-making process is necessary.
The goal of **explainability** in the other hand is to convey human-understandable justifications for the results and judgments of AI systems. It fills the gap between users' comprehension and trust of the model's outputs and the intricate inner workings of AI models.



Interpretability and explainability

The creation of intelligent systems capable of carrying out activities that traditionally require human intelligence is known as artificial intelligence, or AI. But as AI models become more complicated, there is frequently a lack of transparency that makes it difficult for people to comprehend how the models come to their predictions or choices. This problem is solved through interpretability and explainability, which make AI models more clear and comprehensible. They offer explanations in a way that humans can understand and insights into the variables impacting the model's outputs. To achieve interpretability and explainability, strategies including feature importance analysis, rule-based systems, and post-hoc interpretability techniques like SHAP and Shapley values are used. Finally, users can acquire confidence in AI systems, spot biases or errors, assure ethical considerations, and make wise judgments based on the model's outputs through enhancing interpretability and explainability.

IV.SHAP and Shapley Values

1. Motivation of SHAP values

One data scientist usually has to choose between a simple model that can be interpretable but not accurate enough and a complex model that is accurate but not that interpretable. Indeed data scientists want a model with a good enough accuracy to build user trust, while being able

to interpret them. However, people tend to prefer simple models for their ease of interpretation even if their accuracy is worse than the complex ones. Taking complex models gives enough accuracy for data scientists, but they are too complex to interpret or even explain their predictions. One way to solve this issue is to take a simple model that can be interpretable but still accurate enough for the behavior observed. Another way to solve this problem is to use SHAP values which allow the data scientist to work on complex models that are optimally accurate and make them interpretable. This need for interpretability for complex models is essential in fields such as finance, banking, health or medicine.

2. From Shapley values to SHAP values

Introduced by Scott Lundberg, a Senior Researcher from Microsoft Research AI team, SHAP stands for SHapley Additive exPlanations. SHAP values come from game theory, based on Shapley values, to explain machine learning models. It is inspired from the game theory of Lloyd Shapley who won a Nobel Prize in 2012 for it. He created Shapley values as he studied the problem of how to make a fair payout between all players of a game that interact with each other. In this situation, one's outcome depends on others actions, this is an interactive decision making process. In order to make a complex game fair, we need to know the contribution of each player so they could be rewarded fairly according to their investments. So Shapley values are a concept from cooperative game theory that attribute a value to each player in a cooperative game. In the same way, SHAP values aim to give us insights on our models by quantifying the contributions of each feature relative to a prediction of machine learning models. Said differently, SHAP values assign importance for the features in machine learning models. So SHAP values give a fair trade-off between accuracy and interpretability, as using them allows scientists to use complex models with good accuracy and get some interpretations from them. SHAP values give interpretability and explainability, for feature importance and model explanation. So this concept gives essential information for informed feature engineering, debugging, identifying potential biases in models, informing human decision-making, and building trust for example.

For complex models, the model itself cannot explain itself, so we must use a simple explanation model to interpret the complex model. An explanation model is any interpretable approximation of the original model. For that, one can take an explanation model from the additive feature attribution methods which Shapley values come from. These methods are the foundation of many several explanation models used in XAI.

Additive feature attribution methods have an explanation model that is a linear function of binary variables which can activate or not the effect of each feature of a machine learning model. Summing the effects of all feature attribution approximates the output of the original model. These methods use classic equations from cooperative game theory to compute the explanation of model predictions : Shapley regression values, Shapley gambling values and quantitative input influence. Those concepts have their equation explained in papers proposed in bibliography, but one of them is worth mentioning : Shapley regression values.

Shapley regression values are feature importances for linear models in the presence of multicollinearity. This method gives an intuition about the logical reasoning behind SHAP and Shapley values. To get Shapley regression values, it requires to train one model on all feature subsets $S \subseteq F$, where F is the set of all features. In that way, we have the computation of prediction of all different subsets of features, taking into account the effects of each feature on each other. The Shapley values are then computed and used as feature attributions as they are a

weighted average of all possible differences of having one feature or not in the different subsets. Therefore, it assigns an importance value to each feature that impacted one prediction of the model.
Shapley regression values already give an insight on the way SHAP values are going to work. Indeed, these additive feature attribution methods help us to understand the contribution of individual features to the output of a machine learning model. In these methods, the model's output is decomposed into contributions from each input feature. This decomposition can help identify which features are most influential in the model's decision-making process and can be used to interpret the model's behavior. Known additive feature attribution methods are LIME (local approximation of a model prediction for interpretation of individual model predictions), DeepLIFT (recursive prediction explanation method for deep learning), Layer-Wise Relevance Propagation (gives interpretability of predictions of deep networks) for example.

To get to SHAP values, there is one theorem from Scott Lindberg paper that combines cooperative game theory results and states there is only one possible explanation model that follows the definition of additive feature attribution methods and respects 3 properties : Local accuracy, Missingness, Consistency. Previously mentioned additive feature attribution methods only respect missingness property. SHAP values method as presented by Lundberg is proposed as a unified approach that improves previous methods while respecting all the properties. So, according to the theorem It is a unified measure of feature importance and SHAP values, as an explanation model, give results that match the output of the original model we are trying to explain, that's local accuracy. Then it respects missingness as the other additive feature attributions method, which means that features missing in the original input of the original model have no impact, then the SHAP values for missing features should be 0. Finally, SHAP values are consistent as the importance of the features is preserved even when the original model changes, to allow meaningful feature comparison and selection across different models and datasets. In other words, SHAP values are consistent if for any two data points with similar feature values except for one feature, the absolute difference in their SHAP values for this one feature should be less than or equal to the absolute difference in their explanation model for those data points.

3. SHAP values in practical

Indeed, a fair distribution could be done using SHAP values. We treat each feature like a player in the game and calculate SHAP values to get their contribution to the game. Each feature is a player and the payout is the model prediction. So we have to consider how the cooperation of players perform with or without a specific player, taking into consideration their interactions between them, which is to take into account all the subset possible of the coalition.

Therefore, SHAP values measure the contributions to one final outcome from each player separately among the coalition, while preserving the sum of contributions being equal to the final outcome. As for Shapley values, to know the true contribution of a feature/player, SHAP values aim to compute the marginal contribution of a feature by averaging all its contributions in every kind of situation (different subsets of groups, so therefore different kinds of interactions possible...).

4. Computing SHAP values

Computing SHAP values is computationally extremely expensive, as they are usually a NP-hard problem. Therefore, one computes SHAP values using approximations. It can be done through

different methods of approximation. One can use approximations applicable for a wide range of models : model-agonistic approximation, such as Kernel SHAP which is a mix between Linear LIME and Shapley values.

One can also use approximation tailored to a specific machine learning model, those are model-specific approximations, such as :

- **Linear SHAP :** it is fitted to compute SHAP values for linear models by assuming the linearity of the model considered, then considering that the relationship between the features and the output is linear too. It is widely used for linear regression models and other linear models.
- **Low-order SHAP :** it is for low-order models that consider negligible interactions between features, then computing SHAP values for each feature independently. It removes the need to consider all possible feature interactions.
- **Max SHAP :** it is a sampling-based approximation that involves selecting a subset of the features that have the highest absolute SHAP values and computing the SHAP values only for those features.
- **Deep SHAP :** it is fitted for deep neural networks, it is a variant of the backpropagation algorithm to compute the SHAP values for each feature in the input.
- **Tree SHAP :** it is fitted for tree-based models, it is the most commonly used in applications for finance, healthcare and marketing. However, it calculates the SHAP values very fast because it takes advantage of the conditional distribution from the tree structure of the model, but the use of conditional distribution instead of marginal distribution can introduce the problem of causality.

SHAP values can be applied on different machine learning models as we saw, but even in Natural Language Processing, where SHAP values can explain how each word composing a text leads to a specific signification of one model's prediction.

5. The advantageous properties of SHAP values

In the context of providing information about the relationship between features and model predictions, SHAP values take into consideration the interactions between input features unlike other methods such as :

- **Permutation feature importance :** it is a method to break the relationship between a feature and the target to quantify the model dependency on the feature.
- **Partial dependence plots :** it is a method for visualizing the relationship between a single feature and the model predictions while marginalizing over the other features.

Also the value of SHAP values are directly in the unit of the feature that one is trying to predict, so comparisons are relevant and direct. SHAP values explains locally the individual prediction of a black box model, by explaining how the prediction has been fairly distributed among the individual input features, showing how each feature has contributed to a specific prediction. However, we can still get valid global explanations, by aggregating individual explanations (summarizing with the average of the absolute SHAP value over all background values for example). Therefore it can explain a model locally but also globally unlike other methods such as LIME (Local Interpretable Model-agnostic Explanations).

Also, SHAP values not only show feature importance but also show whether the feature has a positive or negative impact on predictions, so it gives global interpretability, whereas other methods such as Anchors for example. Anchors are generated by identifying the minimum set of conditions that must be true for the model to make a specific prediction for an instance. Anchors can provide a simple, interpretable explanation of the model's behavior, but do not

necessarily capture the complexity of the model like SHAP values. SHAP values can be used to generate counterfactual explanations by identifying the input features that, if changed, would result in a different model output. This can help to understand how the model would behave under different scenarios and provide insights into how it could be improved. However, one has to keep in mind that SHAP values depend on the model itself, so if the model is overfitting or underfitting, the SHAP values will be of poor quality to interpret.

Moreover, as we saw, SHAP values give global interpretability and can generate counterfactual explanations. So, they can provide a clear and intuitive explanation of how the model is making its decisions, but also how it fails sometimes as they allow us to compare the relative importance of input features across different models. Indeed, it provides insights into the underlying relationships between the input and output variables so SHAP values can be used to detect and mitigate bias in machine learning models.
By analyzing the contribution of different features to the prediction, SHAP values can help identify features that may be unfairly influencing the model and allow for more equitable decision-making. SHAP values are useful for model monitoring. Indeed, instead of explaining the output of a model, we can decompose the loss for example among each of the input features, to know which features are driving the lack of accuracy. Therefore we can use SHAP values to explain indirectly the behavior of a metric, in order to identify the features that are driving the lack of performance of a machine learning model. However this functionality of SHAP values is quite limited to some applications cases.

6. Perspectives about SHAP values

As the use of AI is growing exponentially in nowadays technologies, it is essential to understand SHAP values for making informed decisions in data science and machine learning. It is even more important for fields like finance or healthcare where decisions making should be explainable and understandable as economical stability and human lives are at stake. In this context, many perspectives exist for SHAP values.

One perspective is developing new algorithms for calculating SHAP values in order to handle large datasets and models with high dimensional feature spaces. Although existing algorithms for calculating SHAP values are effective for many applications, such as the Monte Carlo or kernel-based methods, they can be computationally expensive for large models with many input features. New algorithms that are more computationally efficient and scalable could expand the use of SHAP values to a wider range of applications.

Developing new applications for SHAP values could be considered. While SHAP values have been mainly used for model explanation and interpretability, they have potential applications in other areas of AI, such as model selection, feature engineering, and fairness and bias detection that are not generalized enough. Further research is needed to explore these potential applications and to develop new methods that leverage the power of SHAP values, for fairer and more transparent models.

Another perspective could be combining SHAP values with other model explanation techniques: SHAP values can be used in combination with other model explanation techniques, to provide more accurate and comprehensive explanations of model behavior. Further research is needed to investigate the strengths and weaknesses of different combinations of model explanation techniques and to develop methods that optimize the use of these techniques.

Overall, the future of SHAP values in AI looks promising, and further research and development could expand their use and potential applications in model explanation and interpretability in essential fields as transparency is more considered in model development and deployment.

V.SHAP Values and Feature Importance

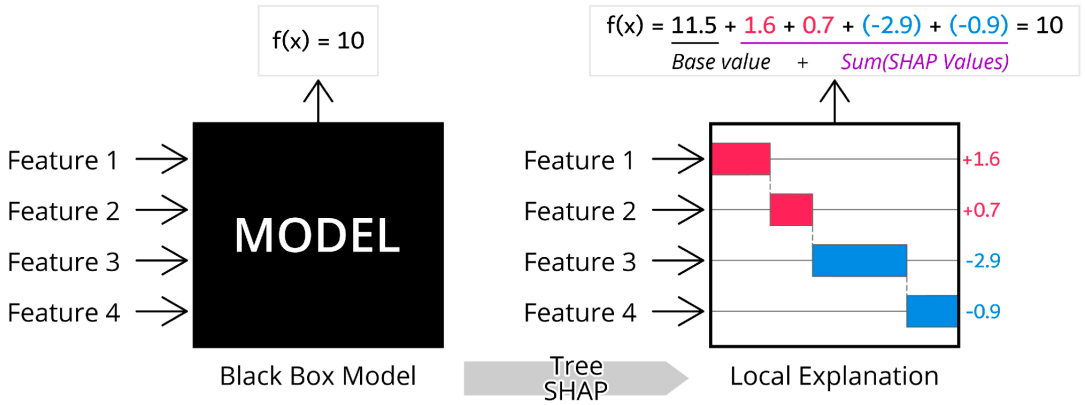
1. Feature Importance

The term "feature importance" describes how much an individual input feature contributes to the prediction or decision-making process in a machine learning model. It seeks to determine which characteristics or factors have the most influence on the model's output. Understanding feature importance is essential because it offers useful insights into the applicability and usefulness of various input features in the decision-making process of the model. We may better grasp the underlying patterns and relationships that underlie the model's predictions by determining the key features.
In machine learning models, determining the relevance of a feature can be done in a variety of ways. Several typical strategies include:

- **Statistical Methods:** These techniques determine the correlation, p-values, or coefficients of each attribute to determine its statistical significance. The size of the coefficients, for instance, in a linear regression, shows the significance of the feature.
- **Techniques tailored to a particular model:** Some machine learning algorithms come with built-in estimates of feature relevance. For example, feature relevance scores are given by Random Forests and Gradient Boosting models based on how frequently a feature is chosen for splitting or how much it lowers the loss function.
- **Importance of Permutation:** This technique includes randomly permuting the values of a feature and measuring how much the model's performance suffers as a result. Features are deemed more crucial if they result in a considerable decrease in performance when permuted.
- **Shapley Values:** Shapley values are used to quantify each feature by calculating how much it contributes to the prediction for a specific instance. They are originated from cooperative game theory. Shapley values offer a just method for allocating relative weight to various attributes.

2. The use of SHAP Values for feature importance

The application of SHAP (SHapley Additive exPlanations) values for feature importance offers a thorough and effective method for comprehending how specific attributes affect predictions or judgments made by machine learning models. The cooperative game theory-derived SHAP values provide a consistent framework for calculating the contribution of each feature to the output of the model.



Explainable AI for SHAPE

By analyzing the impact of each feature on the model's output over all feasible feature combinations, SHAP values assign each feature a relevance score. Greater relevance is indicated by higher absolute SHAP levels. According to the "additivity" principle, the difference between the model's forecast for a given instance and the average prediction for all instances in the dataset will be equal to the sum of SHAP values for all features.
Interpreting SHAP values provides insights into how each feature contributes to the prediction for a given instance. They can be visualized using various techniques, allowing users to easily grasp the relative importance of features. SHAP values offer both global and local feature importance, enabling a nuanced understanding of feature relevance in different contexts.

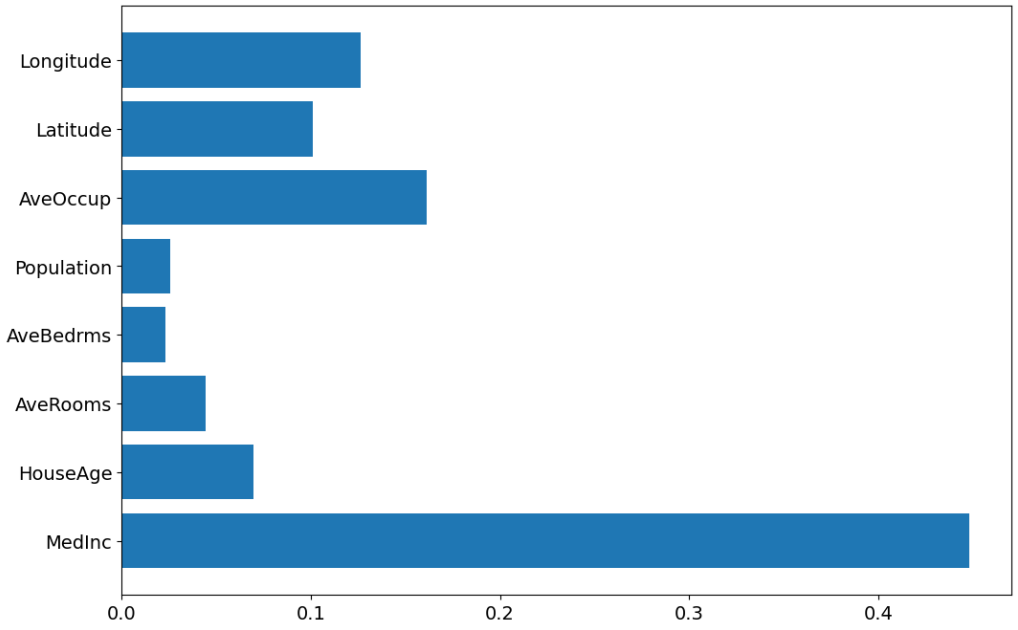
There are many advantages to using SHAP values to determine feature importance. They are model agnostic since they may be used with any machine learning model. A constant and fair attribution of feature importance is made possible by SHAP values, guaranteeing that features are assessed according to their real contributions. They simplify the understanding of complex models by providing intuitive explanations by quantifying the effect of each feature on the model's predictions. Additionally, SHAP values capture feature interactions, allowing for the identification of synergistic or redundant feature combinations.

VI. Implementation and code testing

1. Feature Importance

For this part of the implementation, it wont be complex. In fact, what we tried to do is implement three different methods to plot the feature importance figures. The three method were mentioned in the feature importance part, and it includes the SHAP method as well.
To start off, we used the first method which is the Built in Feature Importance. We are actually using the xgb.feature_importances_, which is an attribute in the XGBoost library that provides the feature importance scores for each feature in a trained XGBoost model. XGBoost (eXtreme Gradient Boosting) is actually a popular machine learning algorithm known for its efficiency and effectiveness in handling structured data.

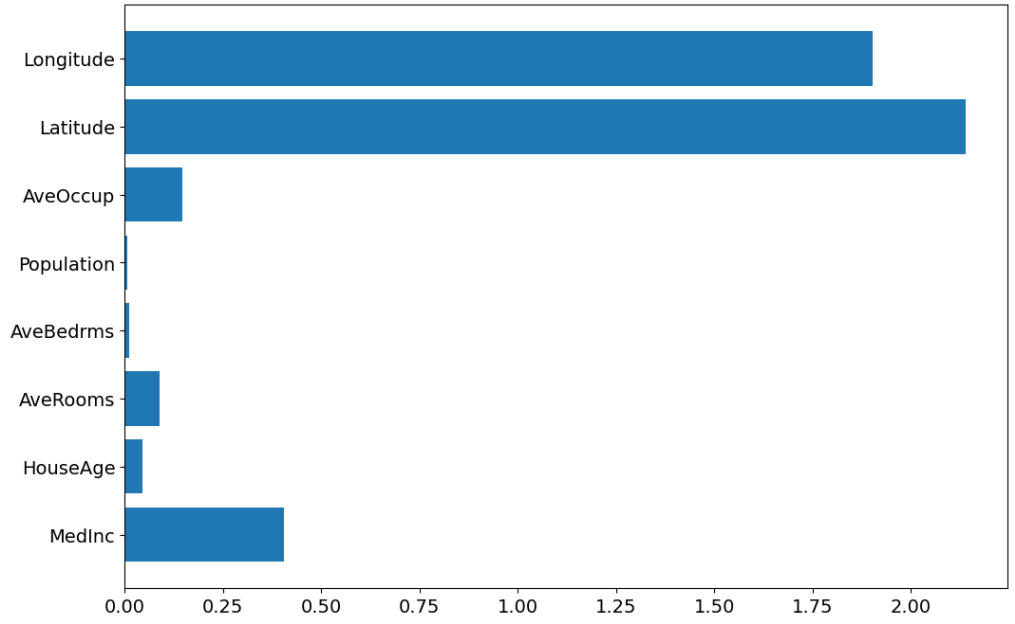
As a result, we got the figure representing the features and their degree of importance using the Built in Feature Importance.



Bar chart of Feature Importance using XGBoost

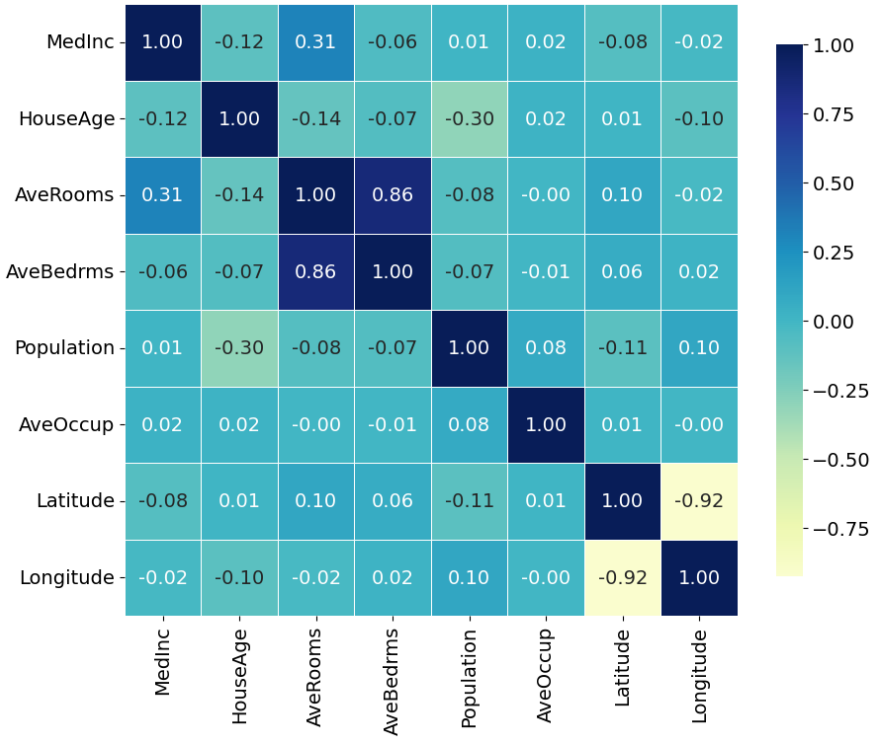
To follow up, we used the second method which is the Permutation-based importance. In fact, it is a technique that is used to quantify the importance of features in a machine learning model by measuring the impact of shuffling or randomly permuting the values of a feature on the model's performance. The basic idea was to assess the decrease in model performance when the values of a particular feature are randomly shuffled or permuted. If the feature is important, shuffling its values will disrupt the underlying patterns or relationships in the data, resulting in a significant drop in model performance. Conversely, if the feature is not important, shuffling its values will have little effect on the model's performance.

As a result, we got the figure representing the features and their degree of importance using the Permutation Based Importance.



Bar chart of Feature Importance using Permutation Based Importance

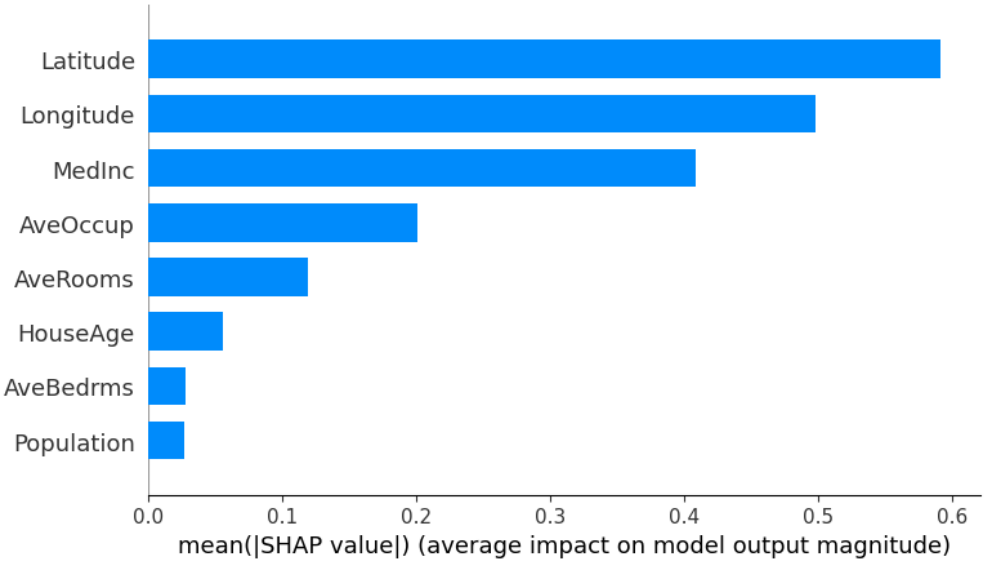
And we can also plot the correlation heatmap to have an idea about the interaction and degree of correlation corresponding to each respective feature.



Correlation heatmap

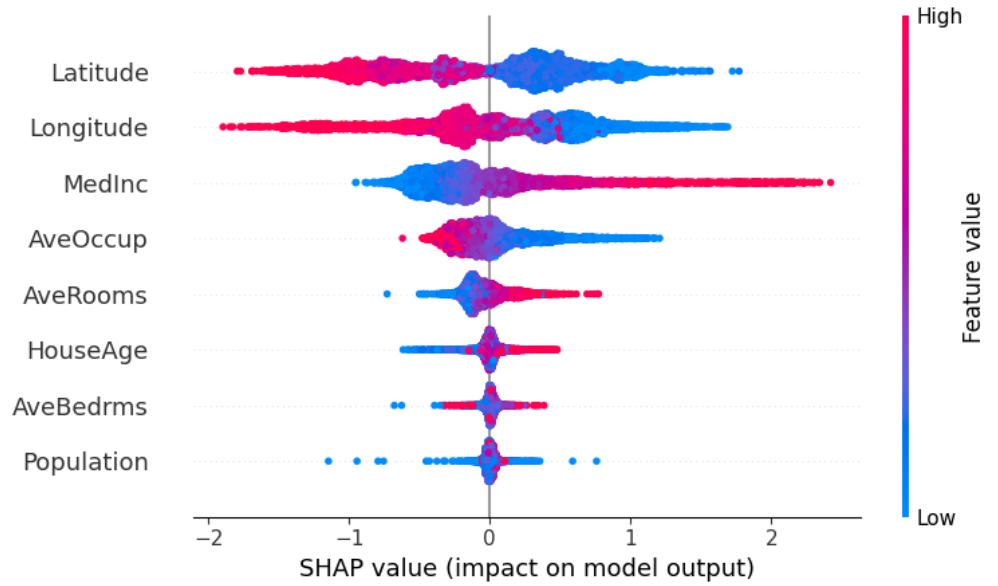
Finally, we used the SHAP Method to get the feature importance. In fact, feature importance is obtained by calculating the SHAP values for each feature. SHAP values then provide a unified measure of feature importance by assigning a value to each feature for every instance in the dataset.

As a result, we got the figure representing the features and their degree of importance using the SHAP Values method.



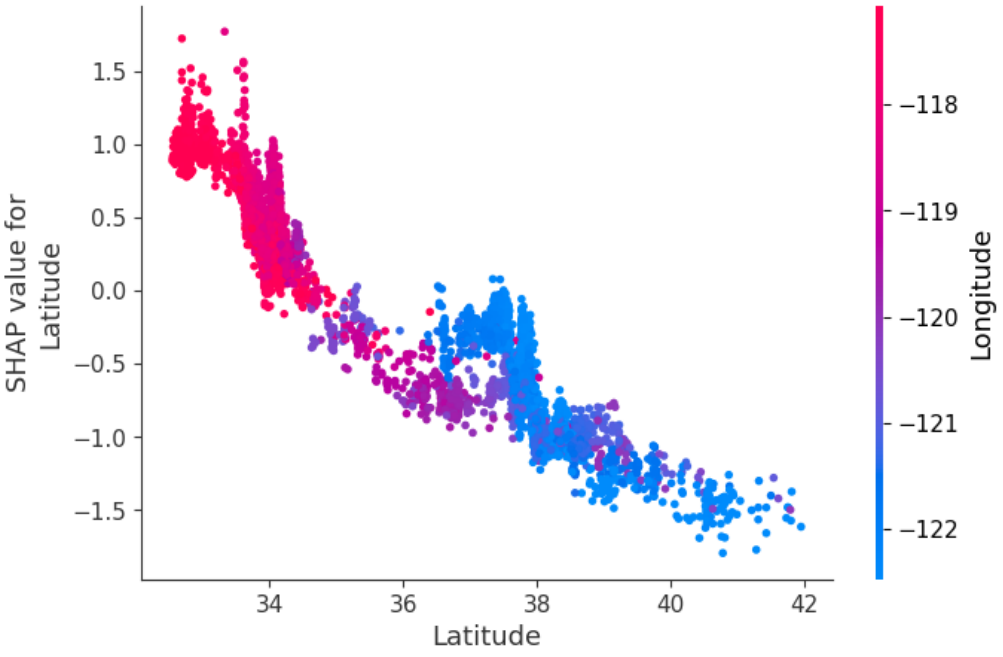
Bar chart of mean importance of SHAP values

In SHAP, we can also plot the degree of impact corresponding to each feature. And we can see the results ranging from High to Low.



SHAP Summary plot

We can plot a dependence plot to observe the degree of dependence between the very first important feature and the second important feature.



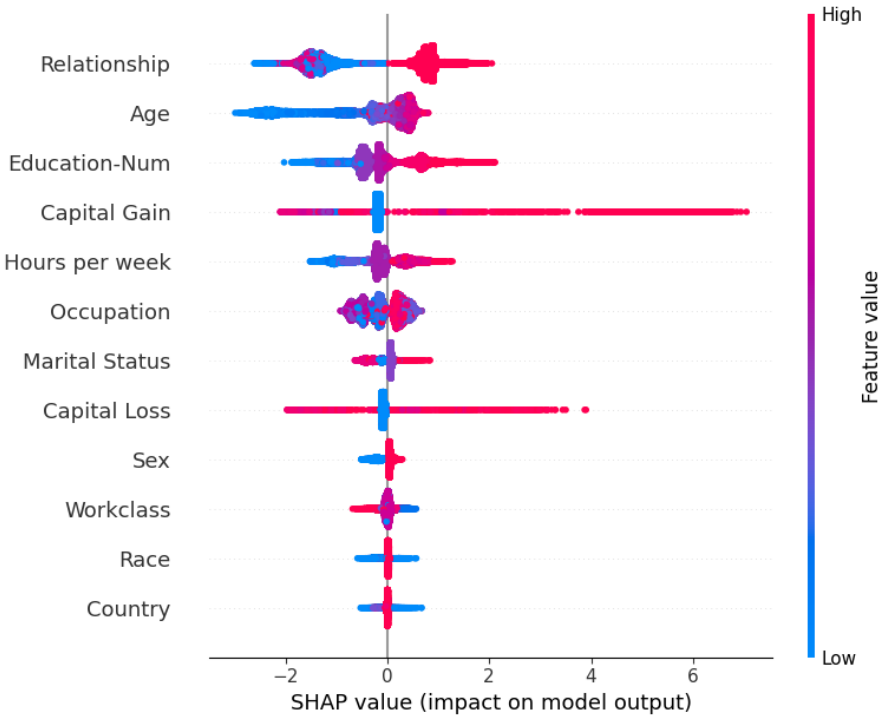
Dependence plot between SHAP values of Latitude and Longitude

2. Model explanation : SHAP values for model debugging

To illustrate the utility of SHAP values for model explanation, we are going to take SHAP values for an example of model debugging. The example is a classifier model (using an XGBoost Classifier) that predict if a person makes over 50K a year,, from the standard UCI Adult income

dataset that uses 12 features : 'Age', 'Workclass', 'Education-Num', 'Marital Status', 'Occupation', 'Relationship', 'Race', 'Sex', 'Capital Gain', 'Capital Loss', 'Hours per week', 'Country'. Here we are summarizing the main parts of interest of the code which is inspired from a resource cited in bibliography. The code used is in the annexes.

Here, XGBoost can also provide feature importance scores for each feature in the classification task. However, in this context, the feature importance scores indicated by XGBoost are contradicting each other, as they are not consistent. So, it motivates the use of SHAP values that come with consistency guarantees. We are then going to apply SHAP values to understand the results given by the classifier model :



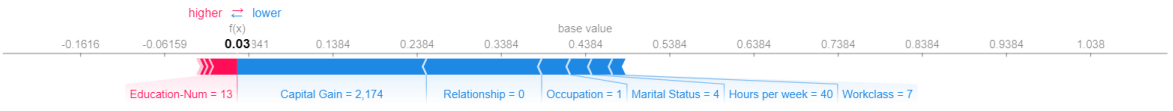
SHAP Summary plot of the XGBoost Classifier

There we can interpret that the relationship feature has more total model impact than the capital gain feature. People who are highly affected by capital gain features, then the capital gain feature matters more than age or relationship features. Capital gain affects a few predictions by a large amount, while age affects all predictions by a smaller amount.

SHAP values can now explain the loss of the model with TreeExplainer from SHAP values, as XGBoost is a decision tree-based model. We are computing now the SHAP loss values. Here the loss considered is the log-loss as we have a classification problem.

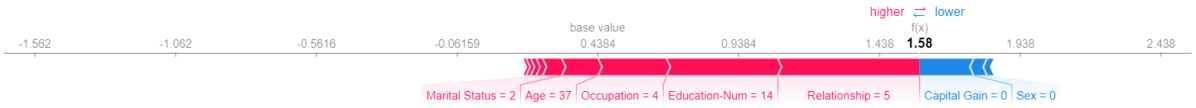
We can check the SHAP loss values for a correct classification and one wrongly classified prediction :

For right :



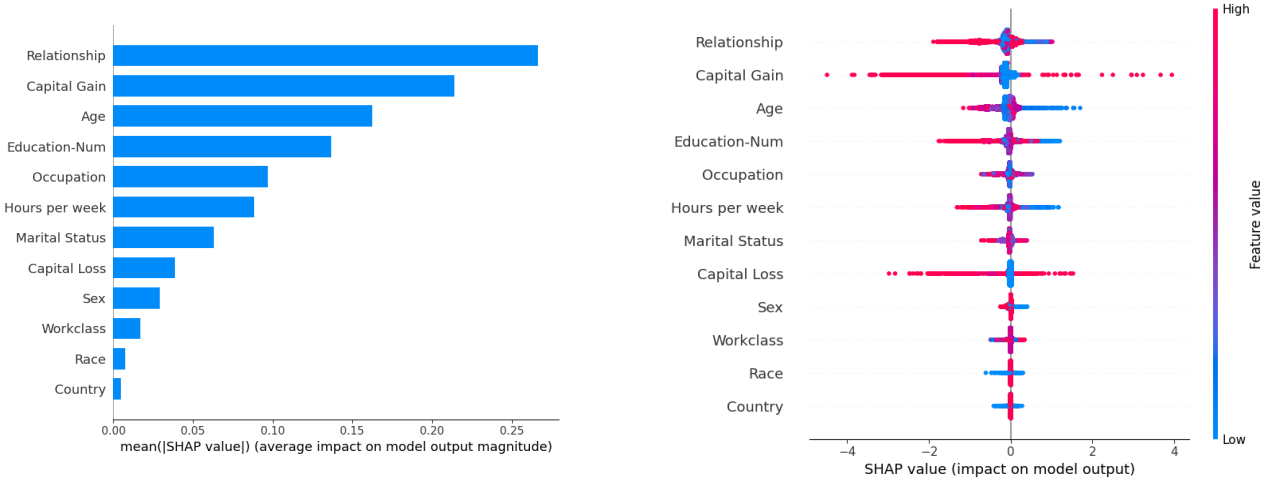
SHAP Force plot for a right classification prediction

For wrong :



SHAP Force plot for a wrong classification prediction

For rightly classified (A and B), the SHAP loss values show the interpretations of their small loss, and that's mostly the capital gain feature that helps to lower the loss. While, for the wrongly classified prediction (for C), the Education Num mostly contributed, with a relationship to increase the loss highly and indicate that the sample was wrongly classified. Using a summary plot, we can visualize the features that mostly contributed to the loss determination :

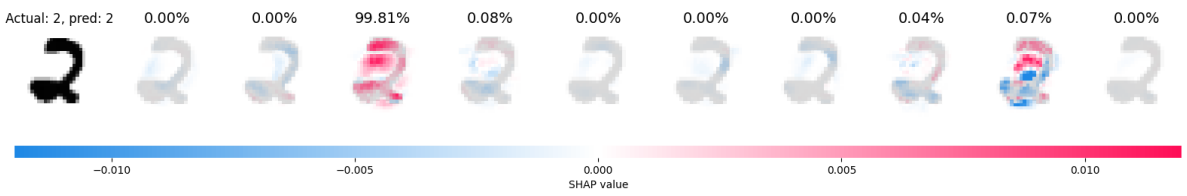


We have the *Bar chart of mean importance of SHAP values* (left) and the *SHAP Summary plot* (right).

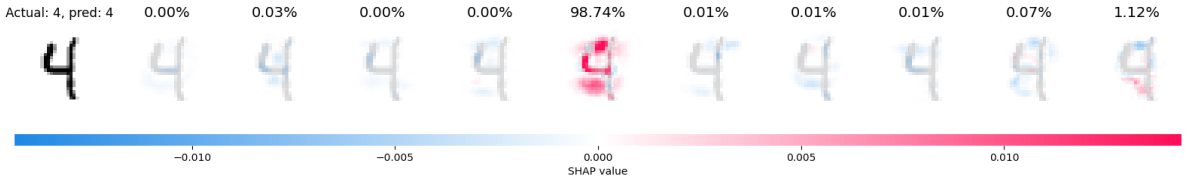
Those graphs confirmed the previous interpretations made.

To illustrate the second part of the model explanation which is mainly related to the image classification, we had to use the dataset MNIST and the model CNN. We did perform two experiments, first when the number of epochs is 8 and therefore the accuracy is > 90%. And the second experiment where the number of epochs is 1 and the accuracy is 80%.

The first experiment will indeed utilizes a reasonably precise network to investigate how SHAP attributes these predictions. Therefore this is the results we got from it.
The provided visualization illustrates the explanation of ten output values representing digits 0 to 9, corresponding to four different images. In this plot, red pixels contribute to increasing the model's output, while blue pixels have the opposite effect. The input images are displayed on the left, accompanied by nearly transparent grayscale backgrounds that serve as contextual information for each explanation.

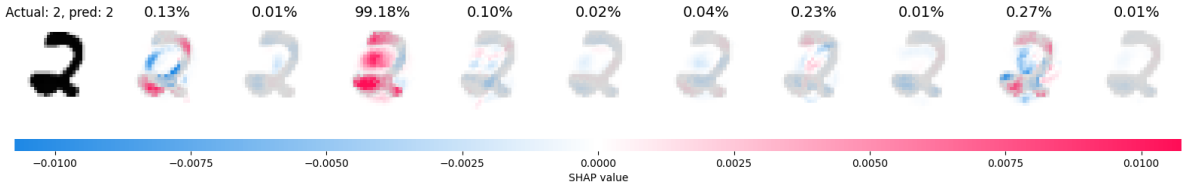


Prediction for the number 2

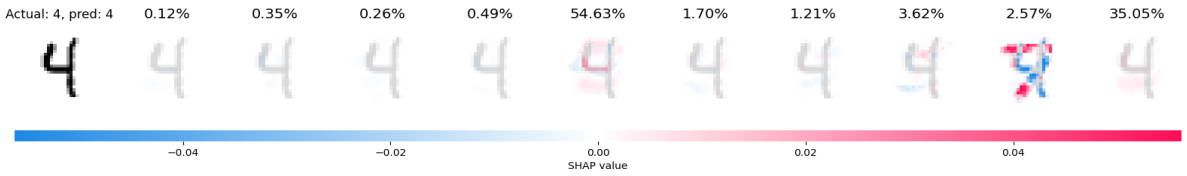


Prediction for the number 4

In the second experiment, we are investigating the behavior of SHAP in the context of inaccurate predictions. This experiment will indeed employs a network with reduced accuracy and prediction probabilities that exhibit less reliability (greater variability among the classes) to comprehend how SHAP functions when the predictions lack reliability. And this is the results we got from it.

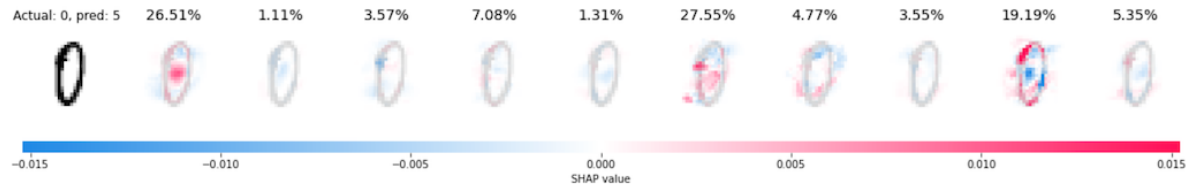


Prediction for the number 2



Prediction for the number 4

What we can conclude from the two experiments is that SHAP only explains the behaviour of the model and it predictions. It doesnt give us a clear explanations of how we could get the true predictions or if the predictions are correct.



Prediction for the number 0

For this experiment, it was really exciting to understand why the model did predict the number 5 as if it was 0 even if there is a lot of differences when it comes to the number's detail.

Conclusion

SHAP values are a method for interpreting and explaining the input features in a model to explain their contribution to one model output. It is based on cooperative game theory and calculates the average marginal contribution of each feature across all possible subsets of features. SHAP values assign then a value for each feature based on its contribution to the overall prediction. It provides local, global and interpretable explanations of the model's behavior and can quantify the importance of features of the model. Shapley values have several advantages over other model explanation techniques, especially their ability to capture feature interactions and their possible applications to a wide range of machine learning models, even though it needs approximation algorithms to compute them.
So, this method has many useful applications such as in model explanation, model selection, feature engineering, and fairness and bias detection in growing fields such as healthcare or finance.

This presentation about SHAP and Shapley values is a way for us to work on a long term project during the year, while doing teamwork. Indeed, this presentation is an investment of time and an opportunity for us to be fully immersed in one particular subject of XAI. As a project, it taught us to be organized to be invested in the learning process. Indeed, even though we had two months of preparation, it was nonetheless too short in time to fully explore all the possibilities in SHAP and Shapley values. However we now understand the concept and the way to implement it in its application cases. As XAI is more and more needed in nowadays technologies, this work is an enriching and rewarding experience.

Bibliography

Lundberg, S. (2017, May 22). *A Unified Approach to Interpreting Model Predictions*. arXiv.org. https://arxiv.org/abs/1705.07874

Microsoft Developer. (2020, May 16). *The Science Behind InterpretML: SHAP* [Video]. YouTube. https://www.youtube.com/watch?v=-taOhqkiuIo

DeepFindr. (2021, March 9). *Explainable AI explained! | #4 SHAP* [Video]. YouTube. https://www.youtube.com/watch?v=9haIOplEIGM

H2O.ai. (2019, October 23). *Scott Lundberg, Microsoft Research - Explainable Machine Learning with Shapley Values - #H2OWorld* [Video]. YouTube. https://www.youtube.com/watch?v=ngOBhhINWb8

GeostatsGuy Lectures. (2021, February 16). *05e Machine Learning: Shapley Value* [Video]. YouTube. https://www.youtube.com/watch?v=oCIybnawLdg

Mage. (2021). How to interpret machine learning models with SHAP values. *DEV Community*. https://dev.to/mage_ai/how-to-interpret-machine-learning-models-with-shap-values-54jf

Lin, C. (2022, March 30). Use SHAP loss values to debug/monitor your model - Towards Data Science. *Medium*. https://towardsdatascience.com/use-shap-loss-values-to-debug-monitor-your-model-83f7808af40f

Luvsandorj, Z. (2022, January 4). Explaining Scikit-learn models with SHAP - Towards Data Science. *Medium*. https://towardsdatascience.com/explaining-scikit-learn-models-with-shap-61daff21b12a

Annexes

Code - Feature Importance using XGBoost

Python

```
!pip install shap

# Importations
import numpy as np
import pandas as pd
import shap
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.inspection import permutation_importance
from matplotlib import pyplot as plt

from xgboost import XGBRegressor

# Generating the data

data_c= fetch_california_housing()
X = pd.DataFrame(data_c.data, columns=data_c.feature_names)
y = data_c.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=12)

xgb = XGBRegressor(n_estimators=100)
xgb.fit(X_train, y_train)

"""## Built-in Feature Importance"""

xgb.feature_importances_

xgb.importance_type

plt.barh(data_c.feature_names, xgb.feature_importances_)

"""## Permutation based Importance"""

perm_importance = permutation_importance(xgb, X_test, y_test)

plt.barh(data_c.feature_names, perm_importance.importances_mean)

import matplotlib.pyplot as plt
import seaborn as sns
```

```
def correlation_heatmap(train):
    correlations = train.corr()

    fig, ax = plt.subplots(figsize=(10,10))
    sns.heatmap(correlations, vmax=1.0, center=0, fmt='.2f',
cmap="YlGnBu",
                square=True, linewidths=.5, annot=True,
cbar_kws={"shrink": .70}
                )
    plt.show();

correlation_heatmap(X_train[data_c.feature_names])

"""## SHAP importance"""

explainer = shap.TreeExplainer(xgb)
shap_values = explainer.shap_values(X_test)

shap.summary_plot(shap_values, X_test, plot_type="bar")

shap.summary_plot(shap_values, X_test)

shap.dependence_plot("Latitude", shap_values, X_test)
```

Code - Model explanation : SHAP values for model debugging

Python

```
# Census income classification with XGBoost

This notebook demonstrates how to use XGBoost to predict the
probability of an individual making over $50K a year in annual
income. It uses the standard UCI Adult income dataset.

Here we show how to use SHAP values to understand XGBoost model
predictions and use SHAP loss values to demonstrate model
debugging.

# Importations

from sklearn.model_selection import train_test_split
import xgboost
import shap
import numpy as np
```

```
import pandas as pd
import matplotlib.pylab as pl
import scipy

# print the JS visualization code to the notebook
shap.initjs()
%load_ext autoreload
%autoreload 2

# Regular SHAP values: feature contribution to prediction

## Load dataset
X,y = shap.datasets.adult()
X_display,y_display = shap.datasets.adult(display=True)

# create a train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=7)
d_train = xgboost.DMatrix(X_train, label=y_train)
d_test = xgboost.DMatrix(X_test, label=y_test)
print(X.columns.tolist())
X.shape
**check some basic information**
print(X.shape)

print("imbalance ratio: entire dataset")
print(y.sum()/len(y))

print("imbalance ratio: training dataset")
print(y_train.sum()/len(y_train))

print("imbalance ratio: test dataset")
print(y_test.sum()/len(y_test))
X.describe()

## Train the model

params = {
    "eta": 0.01,
    "objective": "binary:logistic",
    "subsample": 0.5,
    "base_score": np.mean(y_train),
    "eval_metric": "logloss"
}
```

```
model = xgboost.train(params, d_train, 5000, evals = [(d_test,
"test")], verbose_eval=100, early_stopping_rounds=20)
## Feature attributions from XGBoost

We notice that feature attributions all contradict each other,
which motivates the use of SHAP values since they come with
consistency gaurentees (meaning they will order the features
correctly)
xgboost.plot_importance(model)
pl.title("xgboost.plot_importance(model)")
pl.show()
xgboost.plot_importance(model, importance_type="cover")
pl.title('xgboost.plot_importance(model,
importance_type="cover")')
pl.show()
xgboost.plot_importance(model, importance_type="gain")
pl.title('xgboost.plot_importance(model,
importance_type="gain")')
pl.show()
# Explain predictions

Here we use the Tree SHAP implementation integrated into XGBoost
to explain the entire dataset.
# this takes a minute or two since we are explaining over 30
thousand samples in a model with over a thousand trees
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)
## Visualizing a single prediction

shap.force_plot(explainer.expected_value, shap_values[0,:],
X_display.iloc[0,:])
## Visualizing many predictions

We only visualize 1000 individuals
shap.force_plot(explainer.expected_value, shap_values[:1000,:],
X_display.iloc[:1000,:])
## Bar chart of mean importance

This takes the average of the SHAP value magnitudes across the
dataset and plots it as a simple bar chart.
shap.summary_plot(shap_values, X_display, plot_type="bar")
# SHAP Summary Plot
```

```
We use a density scatter plot of SHAP values for each feature to
identify how much impact each feature has on the model output for
individuals in the validation dataset.
shap.summary_plot(shap_values, X)
# Explain the Log-Loss of the Model with TreeExplainer


## Calculate the shap loss values on the entire dataset
# subsample to provide the background data (stratified by the
target variable)

X_subsample = subsample_data(X, y, n_sample=100, seed_temp=1234)
explainer_bg_100 = shap.TreeExplainer(model, X_subsample,

feature_perturbation="interventional",
                                      model_output="log_loss")

shap_values_logloss_all = explainer_bg_100.shap_values(X, y)
pd.DataFrame(shap_values_logloss_all,
columns=X.columns).to_pickle("./shap_values/shap_logloss_total_bg
_100.pkl")
**Verify the additivity of construction to the model outcome
(here the logloss)**
model_loss = -np.log(model.predict(xgboost.DMatrix(X)))*y \
                +
-np.log(1-model.predict(xgboost.DMatrix(X)))*(1-y)
**check the consistency of the first 10 instances**
model_loss[:10]
explainer_bg_100.shap_values(X.iloc[:10,:], y[:10]).sum(1) + \
    np.array([explainer_bg_100.expected_value(v) for v in
y[:10]])
**check the calculation of expected values**
print(explainer_bg_100.expected_value(True))

print(explainer_bg_100.expected_value(False))
# the calculation of expected values is based on the background
data

X_subsample_index = X_subsample.index
y_subsample = y[X_subsample_index]
### conditional on label

# for instance with True label
y_temp = np.array([True]*len(y_subsample)) # switch all the label
to True
```

```
model_loss_condition_true =
-np.log(model.predict(xgboost.DMatrix(X_subsample)))*y_temp \
                                  +
-np.log(1-model.predict(xgboost.DMatrix(X_subsample)))*(1-y_temp)

# for instance with False label
y_temp = np.array([False]*len(y_subsample)) # switch all the
label to False

model_loss_condition_false =
-np.log(model.predict(xgboost.DMatrix(X_subsample)))*y_temp \
                                  +
-np.log(1-model.predict(xgboost.DMatrix(X_subsample)))*(1-y_temp)


print(len(model_loss_condition_true))
print(len(model_loss_condition_false))

print("Expected model logloss (based on 100 samples), condition
on True: ", model_loss_condition_true.mean())
print("Expected model logloss (based on 100 samples), condition
on False: ", model_loss_condition_false.mean())
## Examples: force plot to understand the contribution to loss
# model prediction: probability/score of being True
y_predict = model.predict(xgboost.DMatrix(X))
**case 1: ground truth 0 and predict 0**
i = 0
print("data label:", y[i])
print("model prediction:", y_predict[i])
print("log loss: ", model_loss[i])

shap.force_plot(explainer_bg_100.expected_value(y[i]),
shap_values_logloss_all[i], X.iloc[i,:], matplotlib = TRUE)
**case 2: ground truth 1 and predict 1**
i = 8
print("data label:", y[i])
print("model prediction:", y_predict[i])
print("log loss: ", model_loss[i])

shap.force_plot(explainer_bg_100.expected_value(y[i]),
shap_values_logloss_all[i], X_display.iloc[i,:])
**case 3: ground truth 0 and predict 1**
i = 5
print(y[i])
```

```
print(y_predict[i])
print(model_loss[i])

shap.force_plot(explainer_bg_100.expected_value(y[i]),
shap_values_logloss_all[i], X.iloc[i,:])
**case 4: ground truth 1 and predict 0**
i = 7
print("data label:", y[i])
print("model prediction:", y_predict[i])
print("log loss: ", model_loss[i])

shap.force_plot(explainer_bg_100.expected_value(y[i]),
shap_values_logloss_all[i], X_display.iloc[i,:])
## 1.2 SHAP loss value Summary plots
shap.summary_plot(shap_values_logloss_all, X)
shap.summary_plot(shap_values_logloss_all, X, plot_type="bar")
**summary plot (decompose to negative and positive)**
shap_values_logloss_all_pd =
pd.DataFrame(shap_values_logloss_all, columns=X.columns)

# positive
shap_values_logloss_all_sum_pos_pd =
shap_values_logloss_all_pd.apply(lambda x: x[x>=0.0].sum(),
axis=0)

# negative
shap_values_logloss_all_sum_neg_pd =
shap_values_logloss_all_pd.apply(lambda x: x[x<0.0].sum(),
axis=0)
print("Shap loss values sum (negative)")
shap_values_logloss_all_sum_neg_pd.plot.bar(fontsize=14)
print("Shap loss values sum (positive)")
shap_values_logloss_all_sum_pos_pd.plot.bar(fontsize=14)
```

Code - Model explanation : SHAP values for image classification

Python

```
"""Image_Classification_AISeminar_SHAP.ipynb

# Exploring SHAP feature attributions in image classification
```

```
"""

import torch
torch.manual_seed(42)

import random
random.seed(42)

import numpy as np
np.random.seed(42)

"""Import the modules we will use in the folllowing cells."""

!pip install models

!pip install shap

"""##Model Definition """

"""Models and training/test fnuctions."""

import torch
from torchvision import datasets, transforms
from torch import nn, optim
from torch.nn import functional as F

DEVICE = torch.device('cpu')
BATCH_SIZE = 128


class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.conv_layers = nn.Sequential(
            nn.Conv2d(1, 10, kernel_size=5),
            nn.MaxPool2d(2),
            nn.ReLU(),
```

```
            nn.Conv2d(10, 20, kernel_size=5),
            nn.Dropout(),
            nn.MaxPool2d(2),
            nn.ReLU(),
        )
        self.fc_layers = nn.Sequential(
            nn.Linear(320, 50),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(50, 10),
            nn.Softmax(dim=1)
        )

    def forward(self, x):
        x = self.conv_layers(x)
        x = x.view(-1, 320)
        x = self.fc_layers(x)
        return x


def optimizer(model):
    return optim.SGD(model.parameters(), lr=0.01,
momentum=0.5)


def train(model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output.log(), target)
        loss.backward()
        optimizer.step()
        if batch_idx % 200 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss:
{:.6f}'.format(
                epoch, batch_idx * len(data),
len(train_loader.dataset),
```

```
                100. * batch_idx / len(train_loader),
loss.item()))


def test(model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            # sum up batch loss
            test_loss += F.nll_loss(output.log(),
target).item()
            # get the index of the max log-probability
            pred = output.max(1, keepdim=True)[1]
            correct +=
pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)
    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{}
({:.0f}%)\n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))


train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('mnist_data', train=True, download=True,
                   transform=transforms.Compose([
                       transforms.ToTensor()
                   ])),
    batch_size=BATCH_SIZE, shuffle=True)

test_loader = torch.utils.data.DataLoader(
    datasets.MNIST('mnist_data', train=False,
transform=transforms.Compose([
        transforms.ToTensor()
    ])),
```

```
    batch_size=BATCH_SIZE, shuffle=True)


def train_model(model, num_epochs):
    opt = optimizer(model)
    for epoch in range(1, num_epochs + 1):
        train(model, DEVICE, train_loader, opt, epoch)
        test(model, DEVICE, test_loader)

import shap
import numpy as np
import matplotlib.pyplot as plt

"""## Experiment 1 - accurate network"""

model_accurate = Net().to(DEVICE)
train_model(model_accurate, 8)

"""### Prepare a batch of images for feature attribution

Extract a batch of images and their labels from the test set.
"""

images, targets = next(iter(test_loader))

"""Select a set of background examples to take an expectation
over" (from the [SHAP
README](https://github.com/slundberg/shap#deep-learning-exampl
e-with-deepexplainer-tensorflowkeras-models))"""

BACKGROUND_SIZE = 100
background_images = images[:BACKGROUND_SIZE]
background_targets = targets[:BACKGROUND_SIZE].numpy()

"""What is left from the test batch will be used to show
feature attribution."""

test_images = images[BACKGROUND_SIZE:]
test_targets = targets[BACKGROUND_SIZE:].numpy()
```

```
print('Digit:      ', end='')
[print('{:6}'.format(x), end='') for x in range(0, 10)];
print('\nBackground: ', end='')
[print('{:6.1%}'.format(x), end='')
    for x in
np.bincount(background_targets)/background_targets.shape[0]];
print('\nTest:       ', end='')
[print('{:6.1%}'.format(x), end='')
     for x in
np.bincount(test_targets)/test_targets.shape[0]];
```

```
def show_attributions(model):
    # Predict the probabilities of the digits using the test
images
    output = model(test_images.to(DEVICE))
    # Get the index of the max log-probability
    pred = output.max(1, keepdim=True)[1]
    # Convert to numpy only once to save time
    pred_np = pred.numpy()

    expl = shap.DeepExplainer(model, background_images)

    for i in range(0, len(test_images)):
        ti = test_images[[i]]
        sv = expl.shap_values(ti)
        sn = [np.swapaxes(np.swapaxes(s, 1, -1), 1, 2) for s
in sv]
        tn = np.swapaxes(np.swapaxes(ti.numpy(), 1, -1), 1, 2)

        # Prepare the attribution plot, but do not draw it yet
        # We will add more info to the plots later in the code
        shap.image_plot(sn, -tn, show=False)

        # Prepare to augment the plot
        fig = plt.gcf()
```

```
        allaxes = fig.get_axes()

        # Show the actual/predicted class
        allaxes[0].set_title('Actual: {}, pred: {}'.format(
            test_targets[i], pred_np[i][0]))

        # Show the probability of each class
        # There are 11 axes for each picture: 1 for the digit
+ 10 for each SHAP
        # There is a last axis for the scale - we don't want
to apply a label for that one
        prob = output[i].detach().numpy()
        for x in range(1, len(allaxes)-1):
            allaxes[x].set_title('{:.2%}'.format(prob[x-1]),
fontsize=14)
        plt.show()

show_attributions(model_accurate)

"""## Experiment 2 - inaccurate network """

model_inaccurate = Net().to(DEVICE)
train_model(model_inaccurate, 1)

"""### Show SHAP feature attributions

"""

show_attributions(model_inaccurate)

"""------

# Test code

"""

# Turn the test code on when needed to not pollute the
notebook
TEST_CODE = True
```

```
# Model to test (uncomment one of them)
modelTest = model_accurate
#modelTest = model_inaccurate

if TEST_CODE:
    e = shap.DeepExplainer(modelTest, background_images)
    shap_values = e.shap_values(test_images)

    shap_numpy = [np.swapaxes(np.swapaxes(s, 1, -1), 1, 2) for
s in shap_values]
    test_numpy = np.swapaxes(np.swapaxes(test_images.numpy(),
1, -1), 1, 2)

    shap.image_plot(shap_numpy, -test_numpy)
```