Improving SHAP Explanations in Sparse Data Contexts: Applications with XGBoost and LSTM

Niyang Bai
niyang.bai@fau.de

October 13, 2024

Abstract

This thesis enhances the robustness of SHAP (SHapley Additive exPlanations) by integrating techniques for sparse data, focusing on one-hot encoded features. It also extends SHAP implementation to XGBoost and LSTM models. For XGBoost, an efficient tree traversal method is employed, while a gradient-based approximation is used for LSTM to explain the influence of sequential data. By leveraging preprocessing, stochastic groupings, PCA, and feature group setups, this research provides scalable and interpretable model explanations, contributing to Explainable AI (XAI) in high-dimensional, sparse, and complex settings.

1 Introduction

Shapley Additive Explanation (SHAP) values are a crucial tool in Explainable Artificial Intelligence (XAI), offering local explanations for predictions made by supervised learning models. However, KernelSHAP, a popular method for approximating SHAP values, encounters significant challenges in high-dimensional and sparse data environments due to its computational demands. This thesis extends the scope of SHAP by not only addressing these challenges through stochastic feature grouping, Principal Component Analysis (PCA), and strategic feature setups but also by implementing SHAP for complex models like XGBoost and LSTM. These efforts aim to enhance the robustness, scalability, and interpretability of SHAP explanations, particularly in scenarios involving sparse datasets, such as those with predominantly one-hot encoded features, and complex sequential or tree-based models.

2 Literature Review

The concept of SHAP values, introduced by Lundberg and Lee (2017), utilizes cooperative game theory to provide a unified framework for interpreting machine

learning predictions, ensuring consistency and local accuracy. KernelSHAP, designed for complex models, faces computational challenges in high-dimensional and sparse data settings.

Lundberg et al. (2020) further advanced SHAP with TreeSHAP, specifically tailored for tree-based models like XGBoost, significantly improving scalability by leveraging the tree structure to efficiently compute SHAP values. This work demonstrated how SHAP could be effectively applied to ensemble methods, addressing the computational challenges associated with traditional SHAP approaches in models like XGBoost.

For deep learning models, such as LSTM, Sundararajan et al. (2017) and Ancona et al. (2017) explored gradient-based methods, which inspired the development of DeepSHAP. These methods utilize backpropagation to approximate SHAP values, making it feasible to interpret complex sequential models. The adaptation of SHAP to LSTM models allows for the explanation of predictions based on temporal data, which is critical for understanding the contributions of individual time steps and features.

Ribeiro et al. (2016) emphasized local interpretability in their work on LIME, which inspired enhancements in SHAP methodologies, including those for XGBoost and LSTM. Dimensionality reduction techniques, as discussed by Jolliffe (2002) through Principal Component Analysis (PCA), reduce computational complexity while retaining essential data characteristics. Hastie et al. (2009) explored stochastic processes, and Goodfellow et al. (2016) discussed deep learning optimization, providing foundational methods to enhance SHAP computations.

Murphy (2012) and Ng (2004) address challenges in handling sparse data, emphasizing the importance of feature selection for computational efficiency. These works collectively underscore the necessity of efficient SHAP computation methods, guiding this thesis to develop innovative approaches for robust and interpretable SHAP explanations in both sparse datasets and complex models like XGBoost and LSTM.

3 Problem Statement

KernelSHAP, while widely accepted and utilized in XAI, suffers from degraded performance in sparse settings due to the heavy computational burden of evaluating numerous feature subsets. This limitation not only hampers the scalability of KernelSHAP but also affects the accuracy and reliability of the explanations it generates, thereby undermining model interpretability and user trust. Additionally, when applying SHAP to complex models such as XGBoost and LSTM, the challenges are further compounded by the need to efficiently compute SHAP values in the context of tree-based and sequential data structures. The core challenge addressed in this thesis is to develop methods that can efficiently handle the computational complexity of SHAP value computation in sparse data, while also extending SHAP's applicability to XGBoost and LSTM models. This involves preserving essential data characteristics and improving the interpretability of explanations across diverse and complex model architectures.

4 Objectives

The primary objectives of this thesis are to:

- **Develop Algorithms**: Create innovative algorithms that apply stochastic feature groupings, PCA, and grouped feature setups for sparse data to compute SHAP values. Extend these approaches to efficiently compute SHAP values for complex models such as XGBoost and LSTM, leveraging tree traversal techniques and gradient-based methods, respectively. These algorithms aim to preprocess sparse datasets and complex model outputs to retain essential information necessary for accurate and interpretable SHAP explanations, while significantly reducing computational complexity.
- **Evaluate Effectiveness**: Assess the effectiveness of these approaches in enhancing the robustness and interpretability of SHAP explanations across various model types and datasets, including XGBoost and LSTM. This includes focusing on:
  - Maintaining or improving the accuracy of SHAP values.
  - Reducing computational time and resources.
  - Enhancing the overall interpretability of the explanations, particularly in tree-based and sequential models.
- **Comparative Analysis**: Conduct a comparative analysis with traditional KernelSHAP strategies and other SHAP implementations, such as TreeSHAP for XGBoost and DeepSHAP for LSTM. This analysis will highlight the advantages and potential limitations of the proposed methods, providing a comprehensive understanding of how stochastic dimensionality reduction, feature grouping, and model-specific SHAP adaptations can be optimally utilized in XAI.

5 Methodology and Comparison

5.1 Naive SHAP (Benchmark)

The naive SHAP method provides a comprehensive way to explain a machine learning model's predictions by distributing the prediction among the features based on their contribution. This is done using Shapley values from cooperative game theory, which considers all possible combinations of feature subsets to ensure that each feature's contribution is fairly evaluated. This method is computationally intensive because it requires evaluating the model on every possible subset of features.

Mathematically, the SHAP value for feature i is given by:

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

where:

- S is a subset of all features N excluding feature i,
- $f(S)$ is the model's prediction using only the features in subset S,
- $f(S \cup \{i\})$ is the prediction when feature i is added to the subset S,
- $|S|$ is the number of features in subset S,
- $|N|$ is the total number of features.

The calculation step can be show as follows:

Algorithm 1 Naive SHAP

Input: Set of all features $X = \{x_1, x_2, \ldots, x_n\}$
for each feature x_i **do**
    for each subset $S \subseteq N \setminus \{i\}$ **do**
        Compute the marginal contribution $v(S \cup \{i\}) - v(S)$
        Weight the marginal contribution by $\frac{|S|!(|N|-|S|-1)!}{|N|!}$
    end for
    Aggregate the weighted marginal contributions to obtain ϕ_i
end for
Output: SHAP values ϕ_i for each feature x_i

Complexity: The computational complexity of naive SHAP is $O(2^n)$ since it requires evaluating the model on every possible subset of n features.

5.2 Stochastic Group and KernelSHAP - Selecting Only Non-Zeros

This algorithm aims to simplify the SHAP value computation for sparse data by focusing only on the non-zero features. Since zero entries in one-hot encoded data do not contribute to feature interactions, excluding them from the SHAP computation reduces computational complexity without affecting the results.

Algorithm 2 Stochastic Group and KernelSHAP (Selecting Non-Zeros)

Input: Set of features $X = \{x_1, x_2, \ldots, x_n\}$
Identify non-zero one-hot encoded features $Z \subseteq X$
for each feature $x_i \in Z$ **do**
    for each subset $S \subseteq Z \setminus \{i\}$ **do**
        Compute the marginal contribution $v(S \cup \{i\}) - v(S)$
        Weight the marginal contribution by $\frac{|S|!(|Z|-|S|-1)!}{|Z|!}$
    end for
    Aggregate the weighted marginal contributions to obtain ϕ_i
end for
Aggregate the SHAP values over multiple stochastic selections
Output: SHAP values ϕ_i for each feature x_i

Mathematical Proof and Justification:

Consider the set of features $X = \{x_1, x_2, \ldots, x_n\}$ where some features are one-hot encoded and hence sparse. Let $Z \subseteq X$ be the set of non-zero features, and let Z_0 be the set of features that are zero in the instance being explained.

For KernelSHAP, we select a subset S of Z and compute the SHAP value as shown in Equation 1.

Since zeros in one-hot encoded features indicate the absence of categorical attributes, their contribution to the value function $v(S)$ is effectively neutral. Specifically, for any $S \subseteq Z$:

$$v(S \cup Z_0) = v(S)$$

This implies that the value of the function does not change with the inclusion of zero-valued features, meaning their marginal contribution is zero:

$$v(S \cup \{i\}) - v(S) = 0 \quad \text{if} \quad i \in Z_0$$

Therefore, the SHAP value for a zero feature $i \in Z_0$ is:

$$\begin{aligned}
\phi_i &= \sum_{S \subseteq Z \setminus \{i\}} \frac{|S|!(|Z| - |S| - 1)!}{|Z|!} \cdot (v(S \cup \{i\}) - v(S)) \\
&= \sum_{S \subseteq Z \setminus \{i\}} \frac{|S|!(|Z| - |S| - 1)!}{|Z|!} \cdot 0 \\
&= 0
\end{aligned}$$

Thus, zero-valued features do not contribute to the SHAP value, confirming that they are not relevant in the computation. The expected value of the SHAP value remains the same whether zero features are included or excluded:

$$\begin{aligned}
\mathbb{E}[\phi_i^{\text{Stochastic}}] &= \mathbb{E}\left[\sum_{S \subseteq Z \setminus \{i\}} \frac{|S|!(|Z| - |S| - 1)!}{|Z|!}(v(S \cup \{i\}) - v(S))\right] \\
&= \phi_i
\end{aligned}$$

Hence, the expected value of the SHAP value using stochastic selection of non-zero features is equivalent to the naive SHAP value.

Complexity: Let m be the number of non-zero features in a particular instance. The computational complexity of this method is $O(2^m)$, which is significantly lower than $O(2^n)$ if $m \ll n$.

5.3 PCA, Stochastic Grouping, and KernelSHAP

This algorithm integrates Principal Component Analysis (PCA) to reduce the dimensionality of the original features while retaining the most significant components. The reduced feature set, combined with the non-zero one-hot encoded features, allows for efficient SHAP value computation. The results are then mapped back to the original feature space.

Algorithm 3 PCA, Stochastic Grouping, and KernelSHAP

Input: Set of original features X
Apply PCA to the original features and select the top k principal components
Combine the selected principal components with non-zero one-hot encoded features Z
for each feature $x_i \in X' = \{\text{top } k \text{ principal components}\} \cup Z$ **do**
    for each subset $S' \subseteq X' \setminus \{i\}$ **do**
        Compute the marginal contribution $v(S' \cup \{i\}) - v(S')$
        Weight the marginal contribution by $\frac{|S'|!(|X'|-|S'|-1)!}{|X'|!}$
    end for
    Aggregate the weighted marginal contributions to obtain ϕ'_i
end for
Map the SHAP values back to the original feature space
Output: SHAP values ϕ_i for each feature x_i

Mathematical Proof and Justification:

Let X be the original feature set and X' be the transformed feature set after applying PCA, where X' consists of the top k principal components.

The SHAP value is computed on X':

$$\phi'_i = \sum_{S' \subseteq X' \setminus \{i'\}} \frac{|S'|!(|X'| - |S'| - 1)!}{|X'|!} (v(S' \cup \{i'\}) - v(S'))$$

Mapping back to the original space, the SHAP values ϕ_i can be approximated by:

$$\phi_i \approx \sum_{j=1}^{k} \alpha_{ij} \phi'_j$$

where α_{ij} are the coefficients that map the principal components back to the original features.

The expected value of the SHAP value using PCA and stochastic grouping is:

$$\mathbb{E}[\phi_i^{\text{PCA}}] = \mathbb{E}\left[\sum_{j=1}^{k} \alpha_{ij} \sum_{S' \subseteq X' \setminus \{j\}} \frac{|S'|!(|X'| - |S'| - 1)!}{|X'|!}(v(S' \cup \{j\}) - v(S'))\right]$$

Given the linearity of expectation and the properties of PCA preserving variance and interactions:

$$\mathbb{E}[\phi_i^{\text{PCA}}] = \phi_i$$

Thus, the expected value of the SHAP value using PCA and stochastic grouping is equivalent to the naive SHAP value.

Complexity:

- PCA transformation has a complexity of $O(n^3)$ for computing the principal components.
- After PCA, the complexity of KernelSHAP on k components and m nonzero features is $O(2^k \cdot 2^m)$. If k and m are significantly smaller than n, this complexity is much lower than $O(2^n)$.

5.4 Stochastic Grouping and KernelSHAP with Grouped Feature Setup

This algorithm groups one-hot encoded features based on their original categorical attributes, allowing for a more structured approach to feature selection. By applying stochastic grouping within these groups, the algorithm maintains the integrity of feature interactions and reduces dimensionality.

Algorithm 4 Stochastic Grouping and KernelSHAP with Grouped Feature Setup

Input: Set of one-hot encoded features grouped by original categorical attributes $G = \{G_1, G_2, \ldots, G_m\}$
for each group $G_j \in G$ **do**
  for each feature $x_i \in G_j$ **do**
    for each subset $S_j \subseteq G_j \setminus \{i\}$ **do**
      Compute the marginal contribution $v(S_j \cup \{i\}) - v(S_j)$
      Weight the marginal contribution by $\frac{|S_j|!(|G_j|-|S_j|-1)!}{|G_j|!}$
    end for
    Aggregate the weighted marginal contributions to obtain ϕ_{ij}
  end for
end for
Aggregate the SHAP values across all groups to obtain ϕ_i
Output: SHAP values ϕ_i for each feature x_i

Mathematical Proof and Justification:

Let $G = \{G_1, G_2, \ldots, G_m\}$ be the groups of features, where each group G_j contains related one-hot encoded features.

For KernelSHAP with grouped features, we compute the SHAP value within each group G_j:

$$\phi_{ij} = \sum_{S_j \subseteq G_j \setminus \{i\}} \frac{|S_j|!(|G_j| - |S_j| - 1)!}{|G_j|!}(v(S_j \cup \{i\}) - v(S_j))$$

The overall SHAP value for feature x_i considering the grouped setup is:

$$\phi_i = \sum_{j=1}^{m} \phi_{ij}$$

Since the grouping preserves the interaction patterns and contributions of features, the expected value is:

$$\mathbb{E}[\phi_i^{\text{Grouped}}] = \mathbb{E}\left[\sum_{j=1}^{m} \sum_{S_j \subseteq G_j \setminus \{i\}} \frac{|S_j|!(|G_j| - |S_j| - 1)!}{|G_j|!}(v(S_j \cup \{i\}) - v(S_j))\right]$$

Again, by the linearity of expectation and the grouping preserving feature interactions:

$$\mathbb{E}[\phi_i^{\text{Grouped}}] = \phi_i$$

Thus, the expected value of the SHAP value using the grouped feature setup is equivalent to the naive SHAP value.

Complexity:

- Let g be the number of groups and h be the average number of features per group.
- The complexity of KernelSHAP for each group is $O(2^h)$.
- Since there are g groups, the total complexity is $O(g \cdot 2^h)$. If h is significantly smaller than n, this complexity is much lower than $O(2^n)$.

5.5 Comparison of Complexities

Name	Complexity	Comments
Naive SHAP	$O(2^n)$	
Stochastic Group and KernelSHAP	$O(2^m)$	m is the number of non-zero features
PCA, Stochastic Grouping, and KernelSHAP	$O(n^3 + 2^k \cdot 2^m)$	k is the number of principal components, m is the number of non-zero features
Stochastic Grouping and KernelSHAP with Grouped Feature Setup	$O(g \cdot 2^h)$	g is the number of groups, h is the average number of features per group

Table 1: Comparison of Complexities

As shown in Table 1, each method significantly reduces the complexity compared to the naive SHAP approach, making SHAP computations feasible for high-dimensional and sparse data.

6 Implementation on XGBoost

XGBoost (Extreme Gradient Boosting) is an ensemble learning method that constructs multiple decision trees in a sequential manner. Each tree is trained to correct the errors of the previous trees. The final prediction for an input instance is obtained by summing the predictions of all individual trees.

Mathematically, for an input feature vector $x = [x_1, x_2, \ldots, x_n]$, the prediction $\hat{y}$ made by an XGBoost model with M trees is given by:

$$\hat{y} = \sum_{m=1}^{M} T_m(x),$$

where:

- $T_m(x)$ is the prediction of the m-th tree for the input x,
- M is the total number of trees in the ensemble.

Each tree in XGBoost is a binary decision tree, where internal nodes represent conditions on input features, and leaf nodes represent the predicted output.

6.1 SHAP Value Calculation for XGBoost

The SHAP value ϕ_i for feature i in the context of XGBoost is derived from Shapley values in cooperative game theory. The Shapley value quantifies the contribution of a particular feature to the model's prediction by averaging its marginal contributions across all possible subsets of features, as shown in Equation 1.

In the case of XGBoost, $f(S)$ can be interpreted as the sum of the contributions of the trees to the prediction using only the features in S. The challenge in calculating SHAP values for tree-based models lies in efficiently computing these contributions for all possible subsets.

6.2 Tree SHAP Algorithm

XGBoost's tree structure allows for an efficient computation of SHAP values by exploiting the hierarchical nature of decision trees. The key idea is to traverse the trees and compute the contribution of each feature by considering the paths taken in the trees.

For each tree T_m, we calculate the contribution of the feature i by evaluating the change in the expected value of the output as we cross from the root to the leaves.

1. **Define the Prediction Function** $f(S)$: The prediction function $f(S)$ for a subset S of features is defined as the sum of the predictions of the trees, considering only the features in S:

$$f(S) = \sum_{m=1}^{M} T_m(S),$$

where $T_m(S)$ is the prediction of tree m using only the features in subset S.

2. **Compute Marginal Contributions**: For each feature i and each subset S, the marginal contribution is computed as:

$$\Delta_i(S) = f(S \cup \{i\}) - f(S).$$

3. **Calculate SHAP Value**: The SHAP value for feature i is then:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Delta_i(S).$$

This involves calculating the weighted sum of marginal contributions over all subsets S.

6.3 Efficient Calculation in Trees

To efficiently compute SHAP values for each feature in a tree ensemble, XGBoost leverages the structure of the decision trees:

- **Tree Traversal**: Each tree is traversed from the root to the leaves. At each internal node, a decision is made based on the value of a feature, determining which branch of the tree to follow.
- **Expected Value Change**: At each node, the expected change in the model output is calculated by considering the contribution of the feature that the node splits on.
- **Path Contribution**: For each path in the tree, the contribution of each feature along that path is accumulated.

For a feature i, the SHAP value ϕ_i can be computed by summing the contributions from all paths in all trees:

$$\phi_i = \sum_{m=1}^{M} \sum_{\text{path}} \text{Contribution of } i \text{ in path of tree } m.$$

This allows for an efficient computation of SHAP values without explicitly considering all possible feature subsets.

6.4 Implementation

Below is the pseudocode that implements the calculation of SHAP values for an XGBoost model:

Algorithm 5 Compute SHAP Values for XGBoost

Input: Trained XGBoost model f, input features x
Output: SHAP values ϕ_i for each feature i
Initialize SHAP values $\phi_i = 0$ for all features i
for each tree T_m in the model **do**
    Initialize a stack to keep track of node contributions
    Start at the root node of T_m
    while not at a leaf node **do**
        Traverse the tree according to the value of the feature at the current node
        Calculate the expected change in prediction due to this feature
        Accumulate the contribution of this feature to ϕ_i
    end while
end for
Return SHAP values ϕ_i for each feature i

In this pseudocode:

- The function traverses each tree in the XGBoost model, calculating the contribution of each feature to the final prediction.
- The stack is used to keep track of contributions as the algorithm traverses down the paths of each tree.
- The SHAP values ϕ_i are accumulated over all trees in the model.

This algorithm provides a method to compute SHAP values for XGBoost models by efficiently traversing each decision tree and aggregating the contributions of each feature.

7 Implementation on LSTM

An LSTM network processes sequences of data, typically represented as a 3D tensor X with dimensions (N, T, D), where:

- N is the batch size,
- T is the sequence length (the number of time steps),
- D is the dimensionality of each input at each time step.

The LSTM network is composed of units (or cells) that manage the information through gates, namely the forget gate f_t, input gate i_t, cell candidate

$\tilde{C}_t$, cell state C_t, and output gate o_t. The forward pass through an LSTM unit at time step t is governed by the following equations:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t, \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\
h_t &= o_t \cdot \tanh(C_t),
\end{aligned}
$$

where:

- x_t is the input vector at time step t,
- h_{t-1} and C_{t-1} are the hidden state and cell state from the previous time step,
- W_f, W_i, W_C, W_o are the weight matrices associated with the forget gate, input gate, cell state, and output gate, respectively,
- b_f, b_i, b_C, b_o are the corresponding bias terms,
- $\sigma(\cdot)$ represents the sigmoid activation function, and $\tanh(\cdot)$ represents the hyperbolic tangent activation function.

The output h_t at each time step is used in subsequent computations or passed to the next LSTM cell in the sequence.

7.1 SHAP Value Calculation for LSTM

The goal of SHAP is to assign an importance value (SHAP value) to each feature that contributes to a particular prediction made by the model. The SHAP value is mathematically derived from Shapley values, a concept in cooperative game theory. For a model f and a feature i, the SHAP value ϕ_i is defined above as Equation 1.

In the context of an LSTM, the input data X is a sequence represented as a 3D tensor $X = [x_1, x_2, \ldots, x_T]$, where each x_t is a D-dimensional vector corresponding to the input at time step t.

For each feature (t, d), where t represents the time step and d represents the dimension within that time step, the SHAP value $\phi_{t,d}$ can be computed by considering all possible subsets S of the input features and measuring the impact of including the feature (t, d) in each subset.

Mathematically, the marginal contribution of the feature (t, d) is:

$$\Delta_{t,d}(S) = f(X_{S \cup \{(t,d)\}}) - f(X_S),$$

where X_S is the input tensor with only the features in subset S active, and $X_{S \cup \{(t,d)\}}$ is the input tensor with feature (t, d) added to the subset S.

The SHAP value for feature (t, d) is then:

$$\phi_{t,d} = \sum_{S \subseteq N \setminus \{(t,d)\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Delta_{t,d}(S).$$

This formulation involves evaluating the model output for all possible combinations of feature subsets, which is computationally expensive.

7.2 Gradient-Based Approximation

Given the high computational complexity of calculating exact SHAP values, especially for models like LSTMs with a large number of features, we can use a gradient-based approximation method similar to the one used in DeepExplainer.

For an input sequence X and a reference sequence X_{ref}, which typically consists of baseline values (e.g., zeros or mean values), the gradient of the model's output with respect to the input can be used to approximate the SHAP values.

The gradient of the output $\hat{y} = f(X)$ with respect to the input X is denoted as:

$$\nabla_X f(X) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_T} \right].$$

The SHAP value for each feature (t, d) can be approximated by:

$$\phi_{t,d} \approx (X_{t,d} - X_{\text{ref},t,d}) \cdot \nabla_{X_{t,d}} f(X),$$

where $X_{t,d}$ is the value of the input feature at time step t and dimension d, and $\nabla_{X_{t,d}} f(X)$ is the corresponding gradient.

This approximation leverages the intuition that the contribution of a feature to the model's output can be estimated by how much the output changes with respect to that feature, scaled by the difference between the actual input and the reference input.

7.3 Implementation

Below is the pseudocode that implements the calculation of SHAP values for an LSTM model using the gradient-based approximation:

In this pseudocode:

- The function `compute_gradients` computes the gradient of the model's output with respect to its input.
- The variable `diff_input_reference` represents the difference between the actual input and the reference input.
- The nested loops iterate over all time steps T and dimensions D of the input sequence to calculate the SHAP values for each feature.

Algorithm 6 Compute SHAP Values for LSTM
H

Input: Trained LSTM model f, input sequence X, reference sequence X_{ref}
Output: SHAP values $\phi_{t,d}$ for each feature (t, d)
Compute the output for the full input sequence: $\hat{y} = f(X)$
Compute the gradient of the output with respect to the input sequence: $\nabla_X f(X)$
Compute the difference between the input and reference: $\Delta X = X - X_{\text{ref}}$
for each time step t in T **do**
    for each dimension d in D **do**
        Approximate the SHAP value: $\phi_{t,d} = \Delta X_{t,d} \cdot \nabla_{X_{t,d}} f(X)$
    end for
end for
Return SHAP values $\phi_{t,d}$

This algorithm provides a method to compute SHAP values for LSTM models by approximating the contribution of each feature using gradient information. The gradient-based approach significantly reduces the computational complexity compared to the exact calculation of SHAP values, making it feasible for deep learning models with large input spaces.

8 Evaluation Criteria

The evaluation of the proposed methodology will be conducted based on the following criteria:

- **Accuracy**: Measure the fidelity of the SHAP values in representing the true contribution of features to the model's predictions, particularly in complex models like XGBoost and LSTM. This will involve validating the SHAP values against known benchmarks or expected feature contributions.
- **Computational Efficiency**: Assess the reduction in computational time and resources compared to conventional SHAP value computation methods, including KernelSHAP, TreeSHAP, and gradient-based SHAP methods. This criterion will evaluate the scalability of the proposed approaches in high-dimensional, sparse, and complex model settings.
- **Interpretability**: Evaluate the clarity and usefulness of the generated explanations in providing insights into the model's decision-making process, with a focus on how well the explanations facilitate understanding of feature contributions in both tree-based models like XGBoost and sequential models like LSTM.

By addressing these criteria, the research will provide a comprehensive understanding of the proposed method's effectiveness in enhancing the robustness,

scalability, and interpretability of SHAP explanations across diverse model architectures, including XGBoost and LSTM.

9 Expected Outcomes

The anticipated outcome of this research is to develop more robust and computationally viable methods for generating SHAP explanations across various data settings, including sparse data and complex model architectures like XGBoost and LSTM. These methods are expected to significantly enhance the interpretability of such models by providing scalable, accurate, and model-specific explanations that can be easily understood and trusted by users. The innovative approaches proposed in this thesis aim to contribute substantially to the field of XAI by overcoming current limitations in KernelSHAP and expanding the applicability of SHAP to a broader range of models, thereby opening new avenues for further exploration and development.

10 Novelty of the Thesis

This thesis introduces a novel approach to explainability in AI by integrating stochastic dimensionality reduction, strategic feature setups, and model-specific adaptations into the computation of SHAP values for both sparse data and complex models like XGBoost and LSTM. Unlike previous methods, this approach emphasizes providing scalable, robust, and model-tailored mechanisms for generating explanations in high-dimensional, sparse, and sequential or tree-based feature spaces. The use of repeated stochastic preprocessing, PCA, feature grouping, and efficient tree traversal or gradient-based techniques represents an innovative step towards more accurate and reliable model explanations, offering significant advancements in the field of XAI.

11 Conclusion

In conclusion, this thesis proposes groundbreaking methodologies to enhance the robustness, computational efficiency, and interpretability of SHAP explanations across a diverse range of models and data scenarios, including sparse datasets and complex architectures like XGBoost and LSTM. By innovatively applying stochastic dimensionality reduction, PCA, feature grouping, and model-specific SHAP implementations, this research addresses existing limitations in XAI and provides scalable, adaptable solutions for interpreting complex models. The expected outcomes underscore the potential of these approaches to significantly advance the interpretability of AI systems, contributing to the creation of more transparent, understandable, and trustworthy models.

References

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. (2017). Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Jolliffe, I. T. (2002). Principal component analysis and factor analysis. *Principal component analysis*, pages 150–166.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.