LightBox XAI

LightBox XAI

Sandra Nguyen

CPSC 491, Fall 2024

Advisor: Marc Velasco

Abstract

LightBox represents the frontier in explainable AI (XAI), aiming to demystify the decision-making processes of AI models through advanced visualization and interpretability techniques. Developed as an open-source initiative, it focuses on promoting transparency, trust, and understanding in AI applications across various industries. By integrating human-centric design principles and cutting-edge methodologies like SHAP and LIME, LightBox offers a platform for AI specialists and hobbyists to validate and improve model performance. The tool's adaptability ensures it meets the growing demand for accountable AI systems, fostering a collaborative environment for innovation. This project underscores the significance of scalable explanations and iterative development in creating ethical and comprehensible AI solutions. As we advance, LightBox will continuously evolve, incorporating user feedback and contributing to the global pursuit of ethical AI.

Glossary

- Explainable AI (XAI): A set of processes and methods that makes the outcomes of AI systems understandable by humans. XAI aims to provide transparency into the decision-making of AI models.
- Visualization and Interpretability Techniques: Tools and approaches used to represent data visually and to explain how AI models make decisions, making them more understandable to humans.
- SHAP (SHapley Additive exPlanations): A method used in machine learning to explain the output of any model by computing the contribution of each feature to the prediction.
- LIME (Local Interpretable Model-agnostic Explanations): A technique that helps explain the predictions of any classifier in an interpretable and faithful manner, by approximating it locally with an interpretable model.
- Human-centric Design Principles: Approaches in design that prioritize the needs, behaviors, and abilities of people who interact with a system or product.
- Iterative Development: A method of software development that involves repeated cycles of development, testing, and revision to build software applications gradually.

Table of Contents

Concept Logo

I. Introduction

1.1 Project Overview

LightBox is envisioned as an exploratory tool designed to significantly enhance the interpretability and transparency of machine learning models. By integrating advanced visualization techniques along with established and emerging explainability methodologies such as SHAP (Shapley Additive explanations) and LIME (Local Interpretable Model-agnostic Explanations), LightBox seeks to clarify the decision-making processes behind AI algorithms. This tool is not only a resource for the technical community, offering deeper insights into model behavior, but also serves as a platform for promoting the ethical use of AI by improving the accountability and comprehensibility of machine learning models. By making complex models more transparent, LightBox supports broader initiatives aimed at responsible AI development, ensuring these technologies are used in a fair and just manner across various sectors.

1.1.1 The Challenge of XAI

Machine learning (ML) models, especially deep neural networks, have evolved into powerful tools that can generate highly accurate predictions across various domains, from healthcare to finance. However, their complexity has also rendered them opaque, earning them the moniker "black boxes." This lack of transparency presents a significant barrier to the adoption and ethical deployment of AI systems, particularly in high-stakes domains where understanding the decision-making process is crucial [7].

One of the main issues with these models is that, despite their impressive accuracy, they fail to offer a clear rationale behind their predictions. Users—whether they are medical professionals, financial analysts, or everyday consumers—often require explanations for AI-driven decisions. Without these explanations, trust in AI systems is compromised, limiting their utility and risking severe consequences, such as biased decision-making or incorrect diagnoses [8].

1.1.2 Why Machine Learning Lacks Explainability

The complexity of modern machine learning models arises from the vast amounts of data they process and the intricate patterns they uncover. However, this same complexity makes it difficult to extract human-understandable explanations for the model's behavior. Traditional methods, such as decision trees, provide clear decision paths, but their simplicity limits their effectiveness with large, complex datasets. In contrast, advanced models like deep learning algorithms offer higher accuracy but trade off interpretability, thus necessitating the development of XAI methods [7].

Explaining these models involves not just revealing what the model's output is, but understanding *why* a particular decision was made—something critical for fields such as healthcare, criminal justice, and finance [8]. In fact, many AI failures can be traced back to the opacity of the models, where decision-makers were unable to question or understand model behavior, leading to flawed actions.

1.1.3 Why Explainability Matters

Explainability is critical for several reasons:

1. **Trust and Accountability**: Users need to trust the systems they rely on, especially in high-stakes situations where lives or significant financial assets are involved. For instance, a patient might want to understand why an AI system recommended a particular treatment, or a loan applicant might need to know why their application was rejected [7] [8].
2. **Ethics and Fairness**: Without transparency, it is difficult to ensure that AI systems are free from biases that could perpetuate societal inequalities. Explainable AI offers insights into how decisions are made, allowing stakeholders to identify and mitigate potential biases.
3. **Compliance and Regulation**: Many industries are governed by strict regulatory standards that require transparent decision-making. For example, in finance, the General Data Protection Regulation (GDPR) in the European Union mandates that individuals have the right to an explanation for automated decisions that affect them.

1.1.4 LightBox as a Solution

In response to these challenges, LightBox is positioned as a cutting-edge tool designed to demystify AI decision-making processes. LightBox integrates advanced XAI methods—specifically SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations)—to offer both local and global insights into machine learning models. These methods help users understand individual predictions (local explanations) and broader model behavior (global explanations), making LightBox a versatile tool for a wide array of applications [8].

By leveraging SHAP, LightBox can attribute the output of any model to its input features, offering a clear view of feature importance and allowing users to understand which factors influenced a particular decision. Similarly, LIME approximates the black-box model locally, enabling real-time explanations that are crucial in decision-critical applications [8].

1.2 Research Objectives and Questions

The overarching objective of the LightBox project is to enhance the trustworthiness and accessibility of AI technologies by improving explainability frameworks. This enhancement is aimed at both demystifying AI operations for non-expert users and providing experts with deeper analytical tools. The project is driven by several specific, actionable research questions that will guide the development and evaluation of the LightBox tool:

1. **How can enhanced interpretability methods such as SHAP and LIME be adapted to better uncover and articulate underlying issues in AI model functionality and accuracy across various applications?**

2. **What methodologies can be developed to quantitatively assess the impact of LightBox on user trust and decision-making accuracy, integrating both software analytics and user feedback mechanisms?**

II. Literature Review and Research Hypotheses

2.1 Theoretical Review

Explainable AI forms the theoretical underpinning of LightBox, drawing from a rich tapestry of research focused on making machine learning models more interpretable and less like "black boxes." This body of work emphasizes the importance of transparency for user trust and ethical AI deployment, advocating for systems that users can understand and interrogate.

2.1.1 Introduction to Explainability in AI (XAI)

Explainable AI (XAI) has emerged as a key area of focus in machine learning, driven by the need to make complex models, particularly deep neural networks, interpretable. These models, while highly effective, have long been criticized for being "black boxes" that offer little insight into how they arrive at their predictions. The ability to explain these models is not merely a technical issue but also a social and ethical imperative, as AI systems are increasingly deployed in high-stakes environments such as healthcare, finance, and law enforcement [3]. LightBox seeks to address these challenges by integrating explainability methods such as SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations), providing a framework that enhances transparency and fosters trust.

2.1.1 Historical Context and Evolution of Explainable AI

The development of XAI is rooted in the early days of AI, where systems were relatively simple and rule-based, allowing for inherent transparency in decision-making. These systems, known as expert systems, allowed users to trace decisions back through the rules applied, making them inherently explainable. However, the rise of more sophisticated models such as deep neural networks brought about a shift. While these models offered unparalleled predictive accuracy, their decision-making processes became opaque, leading to growing concerns about their interpretability [3] [5]. This opacity, particularly in mission-critical fields, underscored the need for new methods that could elucidate the inner workings of complex models.

The evolution of XAI has seen researchers focus on methods that can offer both global and local interpretability. Global interpretability refers to understanding how a model makes decisions across an entire dataset, while local interpretability focuses on explaining individual predictions. SHAP and LIME have been at the forefront of this effort, each offering unique strengths in addressing the interpretability challenges posed by complex models [4].

Hypothesis: The ongoing need to balance model performance with interpretability directly informs the development of LightBox. By leveraging SHAP and LIME, LightBox aims to enhance the interpretability of machine learning models across various applications, making AI decisions more transparent and understandable.

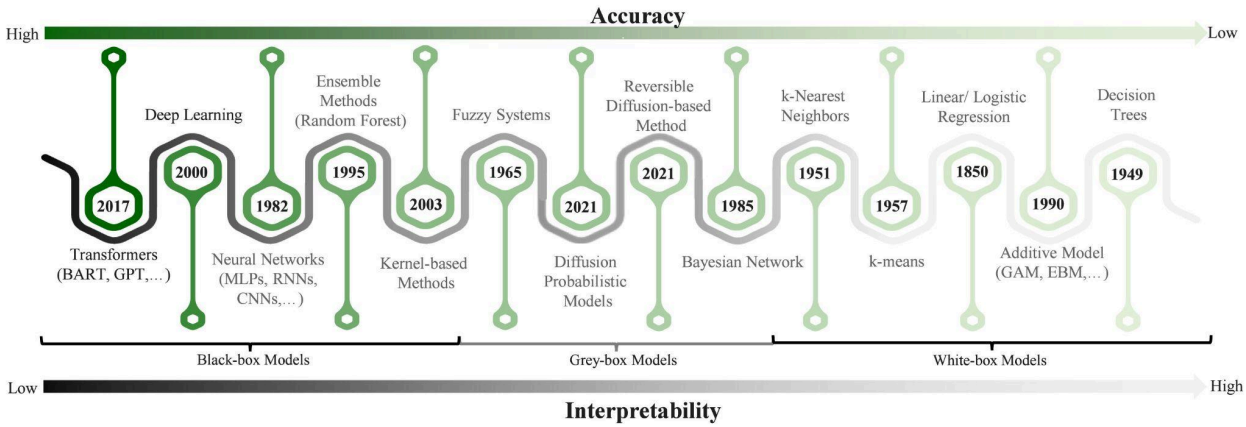

Fig. 1. Evolution of Explainable AI: From early rule-based systems to modern deep learning approaches, illustrating the shift from inherent transparency to complexity and back towards a focus on explainability.

2.1.2 Research Areas and Challenges in XAI

The field of XAI spans multiple research areas, each addressing different aspects of explainability. One key area focuses on developing methods that can elucidate the decision-making processes of black-box models, such as DNNs. This includes both global explanations, which aim to give an overview of how a model generally makes decisions, and local explanations, which provide specific insights into individual decisions.

Challenges in XAI are profound, mainly due to the trade-off between model complexity and explainability. Generally, as the complexity of a model increases, its interpretability decreases. This trade-off is a significant hurdle, particularly in fields where understanding the rationale behind decisions is crucial, such as in healthcare or criminal justice. The challenge is to develop methods that can offer meaningful explanations without sacrificing the performance benefits of complex models [2].

2.1.3 SHAP and LIME: Popular XAI Methods

SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are two prominent techniques in XAI aimed at addressing the opacity of complex models. SHAP leverages game theory to attribute the output of any model to its input features, providing both local and global explanations. It assigns each feature an importance value for a particular prediction, offering insights into which features are most influential.

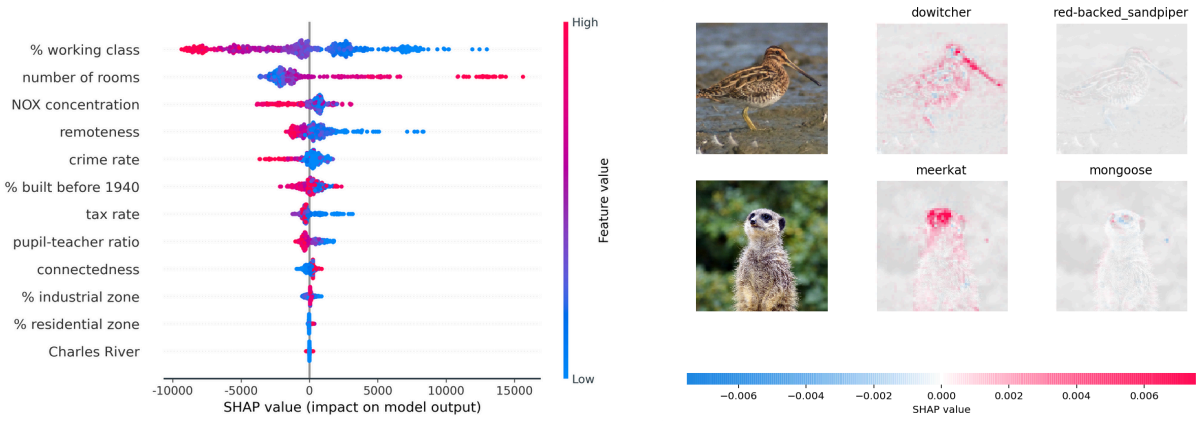

Fig. 2. Examples of SHAP: Visualization showing the impact of individual features on a model prediction, highlighting the interpretability of complex machine learning models.

LIME, on the other hand, approximates black-box models locally with interpretable models, such as linear models, to explain individual predictions. By perturbing input data and observing the changes in outputs, LIME identifies which features significantly influence the model's predictions in specific cases. This method is particularly useful for providing explanations for individual decisions, thereby helping end-users understand the basis of specific model outputs.

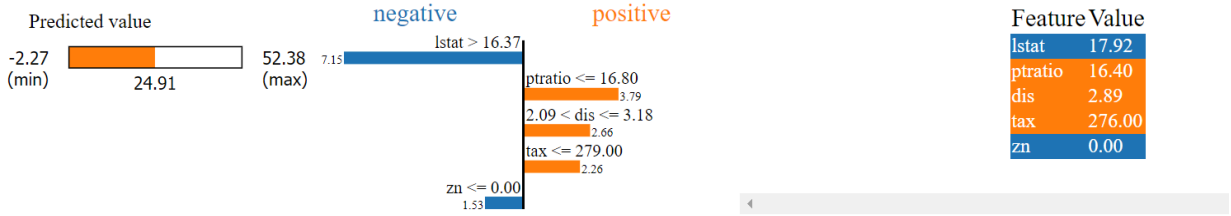

Fig. 3. Example of LIME: Local approximation to explain a specific prediction, demonstrating how feature perturbations affect the output of a black-box model.

Both SHAP and LIME have their limitations and strengths. For instance, SHAP assumes feature independence, which may not always hold, potentially leading to misleading explanations. LIME, while effective for local explanations, may oversimplify the model by approximating it with a linear one, which can omit complex interactions between features. These limitations highlight the ongoing need for advancements in XAI methods to better balance accuracy, complexity, and interpretability [3].

Hypotheses: The strengths and limitations of both SHAP and LIME inform the research questions underpinning LightBox. While SHAP provides robust global explanations, its assumption of feature independence can lead to inaccuracies in certain contexts. Conversely, LIME offers highly effective local explanations but may oversimplify interactions between features. These trade-offs highlight the need for a

platform like LightBox, which seeks to integrate the best of both methods to provide comprehensive, accurate, and user-friendly explanations for machine learning models [5].

2.1.4 Building Trust Through Explainability

Trust in AI systems is crucial for their adoption and effectiveness, particularly in fields where the consequences of decisions can be life-altering. In healthcare, for example, a transparent AI model that explains its reasoning for diagnosing a patient with a specific condition fosters trust among clinicians, who need to understand and verify the model's decision-making process [4]. Similarly, in finance, AI models used for credit scoring must provide transparent explanations to ensure that decisions are fair and free from bias [3].

The importance of transparency extends beyond individual predictions. Users need to trust that the system behaves consistently across different cases and that the explanations provided accurately reflect the underlying model. Studies have shown that providing clear explanations increases user confidence in AI systems, leading to more effective decision-making [6] [5].

Hypothesis: By integrating SHAP and LIME into LightBox, the tool will enhance user trust and decision-making accuracy, particularly in high-stakes domains such as healthcare and finance. LightBox's ability to offer both global and local explanations will ensure that users understand not only individual decisions but also the broader patterns of behavior in the AI system.

2.1.5 Ethical Considerations and the Limitations of Explainability

While XAI offers significant benefits, it also raises important ethical concerns. For example, explanations generated by SHAP or LIME could be misleading if the underlying model is biased. In such cases, users might trust the explanation without realizing that the model itself is flawed [3] [4]. Additionally, oversimplified explanations may lead to a false sense of understanding, particularly in complex fields like healthcare, where nuanced decision-making is required [5].

LightBox will address these ethical challenges by providing comprehensive documentation that guides users on how to interpret the explanations responsibly. Moreover, by incorporating user feedback mechanisms, LightBox will continuously improve the accuracy and relevance of its explanations, ensuring that users are well-informed about the limitations of the models they are working with [4].

Hypothesis: A well-documented framework for using SHAP and LIME, combined with user feedback, will mitigate the risks of misinterpretation and promote ethical AI use across different applications.

2.1.6 Future Directions for Explainable AI

As AI systems become more complex, there is a growing need for advanced XAI techniques that can handle diverse data types, including time-series data and natural language processing models. SHAP and LIME are effective for many current applications, but future research will need to focus on expanding these methods to accommodate more intricate models and broader datasets [6].

LightBox will remain at the forefront of these developments by continuously integrating new explainability methods and expanding its framework to support more complex AI systems. This will ensure that LightBox remains a relevant and effective tool for interpreting AI models as they evolve.

Hypothesis: The integration of future advancements in XAI techniques will ensure that LightBox remains a cutting-edge tool, capable of interpreting increasingly complex models and maintaining its utility across diverse applications.

2.2 Related Software Tools

Comparative analysis highlights existing tools such as Alibi Explain and InterpretML, which offer frameworks for model explanation and analysis. However, LightBox differentiates itself by prioritizing user-centric design and integrating plain language explanations, making the tool accessible to a broader audience beyond data scientists.

Alibi Explain is an open-source library that provides robust explanations for machine learning models through methods like Anchor Explanations, Counterfactual Explanations, and SHAP Values. These techniques are designed to offer deep insights into model decisions, making Alibi Explain particularly valuable for technical stakeholders who require detailed, diagnostic information to understand and improve model behaviors.

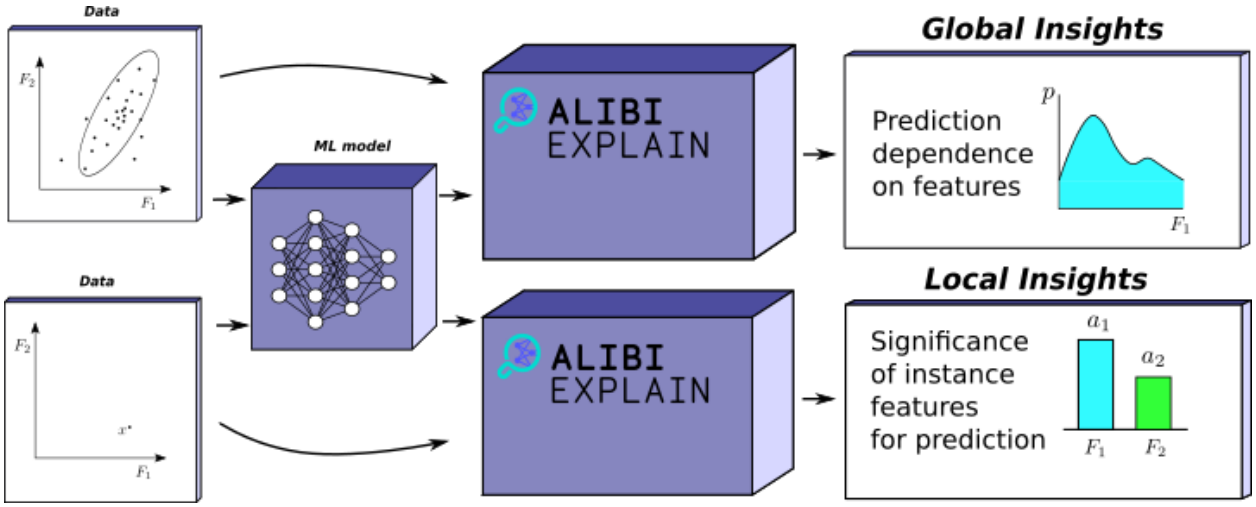

Fig. 4. Overview of Alibi Explain: Displaying the application of techniques like Anchor and Counterfactual Explanations to provide robust insights into model behaviors.

InterpretML distinguishes itself by blending classical interpretable models and modern explanation techniques to enhance model transparency. It features Explainable Boosting Machine (EBM), a glass-box model that uses generalized additive models to offer clear visualizations of how model predictions are

made. Additionally, InterpretML includes interactive features such as a Model Dashboard, facilitating an intuitive understanding of complex model data which caters to both technical and non-technical users.

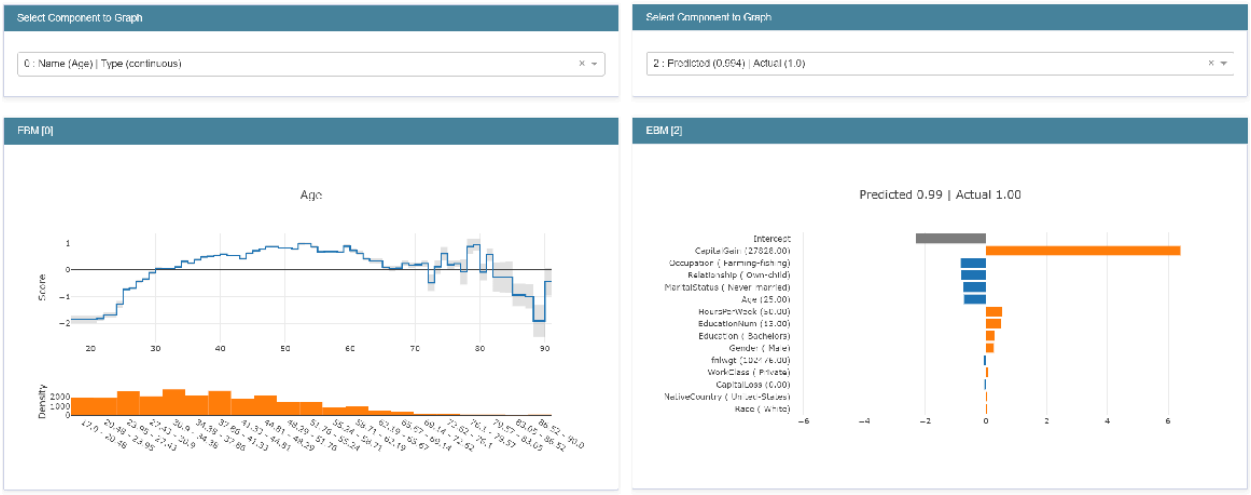

Fig. 5. Interface of InterpretML's Explainable Boosting Machine, showcasing the transparency and ease of understanding complex predictive models.

LightBox extends the concept of explainable AI by prioritizing user-centric design and plain language explanations, making it more accessible to a broader audience beyond data scientists. Unlike Alibi Explain and InterpretML, which cater primarily to users with a technical background, LightBox focuses on simplifying the interpretation process. It uses interactive visualizations and easy-to-understand language to demystify AI decisions, promoting transparency and trust among non-expert users. This approach not only enhances user engagement but also fosters a community-driven development environment, encouraging contributions that drive ethical AI practices and user-focused innovations.

III. Research Methodology

3.1 Study Design and Experiments

To rigorously evaluate the effectiveness and impact of LightBox on user understanding, we propose a mixed-method research design that integrates both quantitative and qualitative research methodologies. This comprehensive approach will allow us to capture a holistic view of the tool's utility across diverse user demographics. Quantitative measures will include user performance metrics, such as time to complete tasks and accuracy of interpreting model outputs, captured through controlled testing environments. Qualitative assessments will involve detailed user interviews and satisfaction surveys to gather in-depth insights into user experience and perceived clarity of the model's explanations.

Real-time Explanations. LightBox will integrate real-time explanation capabilities to provide instant understanding of model decisions. For feature importance metrics, we will use libraries such as SHAP or LIME to calculate and display the impact of each feature on model predictions directly within the user interface. Decision trees generated by algorithms like CART or Random Forests will be visualized using tools like Graphviz or D3.js, providing users with step-by-step tracing of decision paths. For NLP models, text-based explanations will be implemented using techniques from the latest research in explainable AI, presenting reasons in plain language to make model decisions transparent and easy to understand. A typical Sci-kit learn workflow will be utilized.

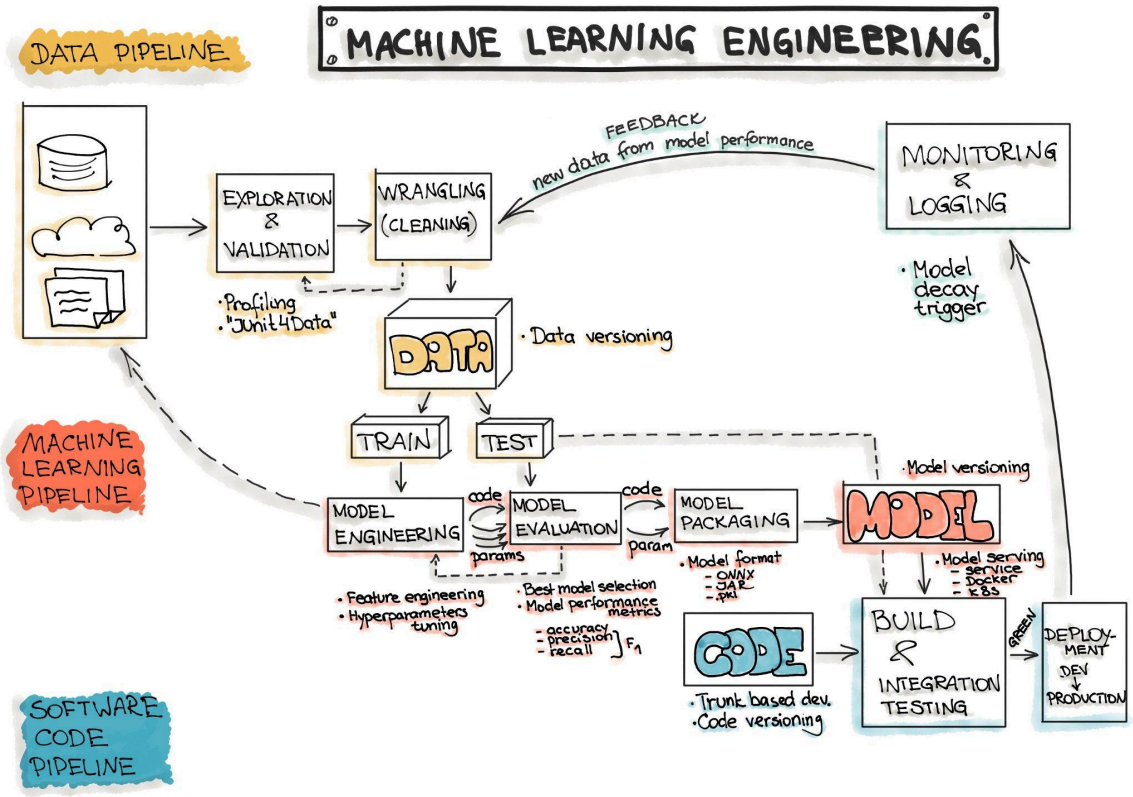

Fig. 6. Workflow of a typical sci-kit learn model development: From data preprocessing to model training and evaluation, highlighting the modular nature of the Python library.

Model Performance Comparison. Users will be able to evaluate and contrast the performance of various models directly within LightBox. Integration with MLflow or TensorBoard will provide functionality for tracking experiments, comparing model metrics, and visualizing model performance over time. This will assist users in selecting the best model for their needs based on empirical data.

To evaluate the model performance comparison functionality within LightBox, users will run experiments involving the deployment of multiple machine learning models, such as Random Forest, Logistic Regression, and Neural Networks, on the same dataset. Using integrated tools like MLflow or TensorBoard, users will track and compare the accuracy, precision, recall, and other key performance metrics over time. They will explore how models perform under varying hyperparameter settings and data inputs. The objective of this experiment will be to test the ease of comparing models side by side and to ensure that LightBox accurately tracks the evolution of model performance. Users will be encouraged to observe the results visually and select the best-performing model based on empirical evidence provided by LightBox.

User-friendly Visual Tools. To simplify complex data interpretation, LightBox will incorporate interactable visualizations using libraries like seaborn, Plotly, or Bokeh. These tools will allow users to manipulate data visualizations dynamically, exploring different aspects of the data and model outputs through interactive charts, heatmaps, and scatter plots.

For testing the interactable visual tools within LightBox, users will load a dataset and visualize the relationships between various features and model outputs. Libraries like seaborn, Plotly, or Bokeh will be used to generate interactive heatmaps, scatter plots, and line charts. The experiment will involve dynamically manipulating these visualizations by filtering data subsets, adjusting axes, or highlighting feature correlations. The goal is to assess how intuitively users can explore data and model results, identifying trends or anomalies. For instance, a user could visually track model drift by comparing how predictions differ across time periods or conditions, evaluating how easily they can uncover these insights through the interactive visual tools.

Automated/Triggered Explanations. LightBox will feature automated explanations triggered by specific user-defined conditions, such as anomalies in data or significant performance drifts in models. This proactive feature will alert users to potential issues, providing contextual explanations and suggesting corrective measures.

To evaluate the automated explanation feature, an experiment will be designed where users define specific triggers, such as significant drops in model accuracy or anomalous data points in predictions. Once these conditions are met, LightBox will automatically generate explanations using SHAP or LIME, providing insights into why the model may be drifting or why certain anomalies are appearing. For example, users may set a threshold for performance degradation and, upon breach, receive real-time explanations regarding which features caused the decline. The experiment will test how effectively LightBox identifies

these conditions and provides timely, actionable explanations, ultimately assessing its capability to alert users to critical model issues.

Version Control. A robust version control system will be embedded in LightBox to manage and track iterations of models, including their performance metrics and explanations. This will be crucial for maintaining a history of model changes, facilitating rollback to previous versions if needed, and understanding the evolution of model performance over time.

In testing the version control feature, users will iteratively update their machine learning models, altering model parameters, retraining the models, or modifying the datasets. For each iteration, LightBox will automatically save a snapshot of the model's state, including performance metrics, feature attributions, and explanations. Users will be able to experiment with rolling back to previous versions to analyze how certain model changes impacted performance or interpretability. This experiment will test how effectively LightBox manages the evolution of models and explanations over time, ensuring that users can maintain a robust history of their models, facilitating easier debugging and long-term monitoring.

User Feedback Mechanism. An integrated user feedback mechanism will streamline the incorporation of user insights into ongoing tool improvements. This feature will enable users to provide feedback directly through the interface, which will be systematically reviewed and used to prioritize future enhancements.

To assess the user feedback mechanism, an experiment will involve prompting users to provide real-time feedback while interacting with LightBox. As users navigate through model explanations or experiment with visualizations, they will be asked to rate the usability, clarity, and satisfaction of features directly from the interface. Users will also have the option to provide textual feedback for specific functions, like model comparison or triggered explanations. The collected feedback will be systematically reviewed, categorized, and prioritized to inform future tool iterations. This experiment will test how efficiently LightBox gathers and processes user feedback, ensuring that users' insights directly contribute to the tool's continuous improvement.

Community Support Forums. To foster collaboration and shared learning, LightBox will include access to community support forums. These forums will serve as a platform for users to share experiences, discuss best practices, and offer solutions to common challenges. This community-driven support system will enhance user engagement and collective problem-solving.

For the community support feature, an experiment will involve users participating in a simulated community forum integrated within LightBox. Users will post questions or challenges related to their model building or explanation needs and engage with others who provide advice, share code snippets, or troubleshoot issues collaboratively. The experiment will measure how quickly users receive helpful responses and the quality of the discussions. By tracking engagement metrics, such as post frequency and response times, this experiment will assess the effectiveness of the community in fostering collective learning and problem-solving. The goal is to ensure that the forum becomes a valuable resource for users to share best practices and collaborate on solving common XAI challenges.

3.2 Data Collection and Analysis Methods

3.2.1 User-Centric Data Collection

Our data collection strategy for evaluating LightBox will involve both surveys and interactive user sessions, tailored to extract comprehensive data on the tool's performance and user satisfaction effectively. Surveys will be structured to quantitatively assess user satisfaction and understanding, utilizing Likert scales and multiple-choice questions to facilitate straightforward analysis and ensure uniform response interpretation. Interactive user sessions will be recorded and scrutinized to monitor real-time interactions with LightBox, providing crucial insights into the user interface's intuitiveness and efficiency.

For the analysis of quantitative data, rather than traditional statistical methods, we will employ machine learning techniques to measure and enhance the performance of machine learning models within LightBox. This will involve using algorithms to process user interaction data and feedback, adapting and optimizing the tool's functionality based on these insights. These machine learning algorithms will be designed to automatically detect patterns and anomalies in user behavior and model performance, enabling dynamic adjustments to improve both user experience and model accuracy. Qualitative feedback will be analyzed using natural language processing (NLP) techniques to extract common themes and patterns from textual data. This will help in identifying user needs and preferences, facilitating targeted improvements in subsequent iterations of LightBox. By leveraging machine learning for both quantitative and qualitative analysis, we aim to continuously refine LightBox in a way that is deeply informed by actual user experiences and interactions, ensuring the tool evolves to meet user demands effectively and efficiently.

Example Use Case: A data scientist using LightBox to explain a model used for loan approvals will be asked to explore the feature attributions and global explanations for multiple scenarios. Their responses will be analyzed to measure how effectively they understand the model outputs and to assess the comprehensibility of SHAP and LIME visualizations.

.

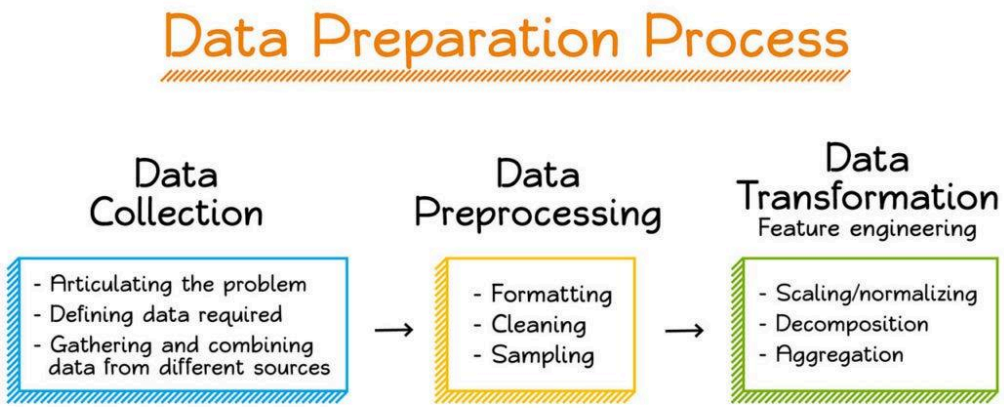

Fig. 7. Data Collection and Processing: A systematic flowchart demonstrating the stages of data acquisition, cleansing, and preparation for use in model training and evaluation.

3.2.2 Data Acquisition and Dataset Search

LightBox's performance will also be assessed through the application of the tool on multiple datasets, spanning various domains such as finance, healthcare, and image classification. We will actively seek out free, publicly available datasets from reputable online repositories, such as:

- **Kaggle**: A well-known platform that hosts a variety of datasets suitable for testing machine learning models. Examples include datasets related to fraud detection, healthcare outcomes, and consumer behavior.
- **UCI Machine Learning Repository**: Another widely used source, offering datasets that cover domains like speech recognition, market analysis, and text classification.
- **Government Open Data Portals**: Governments such as the U.S. or the U.K. offer extensive repositories of public data that can be used for testing LightBox in regulatory and policy-driven use cases.
- **Google Dataset Search**: A meta-search engine that aggregates datasets from various open repositories, useful for identifying niche datasets specific to certain industries or applications.

We will employ a broad range of these datasets to ensure LightBox's versatility across sectors. Each dataset will be preprocessed using standard data cleaning methods such as handling missing values, normalizing data, and removing outliers. These steps ensure that the datasets are ready for machine learning tasks, allowing LightBox to be evaluated in a robust and consistent manner.

3.2.3 Analysis of Model Explanations

Once datasets are acquired and processed, LightBox will be used to apply machine learning models, generating feature attributions and model explanations through SHAP and LIME. However, the analysis does not stop here. We will apply additional data analysis methods to assess the quality and effectiveness of these explanations.

1. **Supervised Learning Models**: The primary models used in LightBox's evaluation will include supervised learning algorithms such as logistic regression, random forests, support vector machines (SVMs), and deep neural networks. Each model will be trained using a portion of the dataset, with LightBox providing explanations at both global and local levels.
2. **Model Performance Metrics**: To assess the performance of LightBox in providing useful explanations, we will track key machine learning metrics such as:
    - **Accuracy, Precision, and Recall**: To ensure that the models provide reliable predictions.
    - **F1 Score**: A measure of the model's ability to balance precision and recall, especially important for datasets with imbalanced classes.
    - **ROC-AUC Curve**: A tool to measure the model's ability to distinguish between classes, particularly useful in binary classification tasks.
3. **Feature Importance Analysis**: Beyond SHAP and LIME, we will also use feature importance scores generated from algorithms like Random Forests and XGBoost, which provide insights into

how different features contribute to the model's decision-making process. LightBox will integrate these scores into its visualization toolkit to offer users a more comprehensive understanding of model behavior.

3.2.4 Feature Extraction and Visualization

Feature extraction is a critical component of model interpretability, and LightBox will employ various techniques to highlight key features that contribute to the prediction. Alongside SHAP and LIME, the following methods will be incorporated to enhance user comprehension:

1. **PDPs (Partial Dependence Plots)**: These plots will be used to show how specific features affect the predicted outcome on average, offering global insights into feature importance. LightBox will allow users to visualize these dependencies, helping them grasp the nonlinear effects that certain features may have on the output.
2. **ICE (Individual Conditional Expectation)**: While PDPs show average effects, ICE plots will be integrated into LightBox to reveal how features impact individual predictions. This helps users understand the variability in predictions for different input instances, providing a richer layer of interpretability.
3. **Saliency Maps and Activation Maps**: For image-based machine learning models, such as Convolutional Neural Networks (CNNs), LightBox will generate visual explanations using saliency maps. These maps will highlight the regions of an image that contributed the most to a model's prediction. Activation maps will similarly offer users a visual representation of which neurons or layers were most active during a prediction.
4. **Dimensionality Reduction Techniques**: LightBox will include tools like t-SNE and PCA (Principal Component Analysis) to visually represent high-dimensional data in lower dimensions, making it easier for users to spot patterns, clusters, and outliers in their datasets. This is particularly useful in understanding how different data points are grouped or classified by the model.

The design of LightBox is centered on enabling better decision-making and refinement of AI models, while significantly improving interpretability. By revealing biases, such as those introduced by imbalanced datasets or unrepresentative training data, and providing tools to correct these biases, LightBox will help users develop more equitable and effective AI solutions. Examples include bias detection in facial recognition technologies or lending algorithms, where LightBox could highlight and help mitigate skewed decision-making. Through its comprehensive design and methodological approach, LightBox will not only advance the field of explainable AI but also democratize AI technology by making it accessible and understandable to a broader audience.

IV. Python Package Development

4.1 Package Requirements

The LightBox package is designed for easy integration and broad compatibility, functioning effectively within standard Python environments. Users can install it using pip, Python's package installer, ensuring simplicity in setup and maintenance.

It supports Python-based machine learning frameworks like TensorFlow, PyTorch, and sci-kit-learn, facilitating seamless application across diverse sectors. The package's minimal hardware requirements are 4 GB RAM and a stable internet connection, making it accessible even in constrained environments.

4.2 Technical Architecture and Design

Architecture Overview. LightBox is developed as a modular Python package. It incorporates a core explanation engine, a visualization toolkit, and NLP extensions, ensuring robust functionality and extensibility:

- Core Explanation Engine: Utilizes algorithms like SHAP and LIME to provide detailed explanations of model predictions. This engine is designed to handle large datasets efficiently and can be extended to include other explanation methodologies as they develop.
- Visualization Toolkit: Includes integration with powerful visualization libraries such as Matplotlib, Seaborn for static visualizations, and Plotly for dynamic, interactive visualizations. These tools help in presenting complex data and model explanations in an understandable manner.
- NLP Extensions: For models involving natural language processing, extensions using NLTK and spaCy offer specific functionalities for text data, enhancing the interpretability of NLP models.

Compatibility and Integration. To ensure compatibility across different systems, LightBox is tested on multiple operating systems such as Windows, macOS, and Linux. Docker containers can be used to create consistent environments for testing and deployment, ensuring that the package operates reliably across all platforms.

4.3 Implementation Details

Implementation involves leveraging Flask for API management, enabling asynchronous tasks and seamless integration with web applications. Data manipulation is handled by pandas and NumPy, while visualization is supported by both static and interactive tools.

API Design. APIs are designed to be simple, allowing users to request explanations which can be returned in various formats including JSON for data or HTML for visual content. This supports integration into web applications and Jupyter Notebooks, promoting widespread use in educational and professional settings.

V. Testing and Evaluation

5.1 Unit Testing and Integration Testing

Testing Framework. Using pytest, a robust framework for Python testing, LightBox will undergo comprehensive testing:

- Unit Tests: Focus on individual components, ensuring that each function returns expected results, such as verifying the accuracy and consistency of the explanation outputs from the core engine.
- Integration Tests: Ensure that the components work together effectively, such as the data flow between the core engine and visualization tools.

5.2 User Testing and Feedback

Structured Testing Sessions. Specific tasks will be designed to evaluate the usability and effectiveness of LightBox, such as interpreting model outputs using the provided explanations and visualizations. These tasks will help assess the intuitive nature of the interface and the clarity of the information presented.

Unstructured User Interactions. After release, ongoing monitoring of how users interact with LightBox in real-world settings will be crucial. Feedback mechanisms built into the software will collect user experiences, which will be analyzed to guide further iterative development.

Compatibility Testing. Testing across different environments will be facilitated by tools like Travis CI or Jenkins, which integrate with GitHub to automate testing across multiple systems every time changes are pushed to the repository.

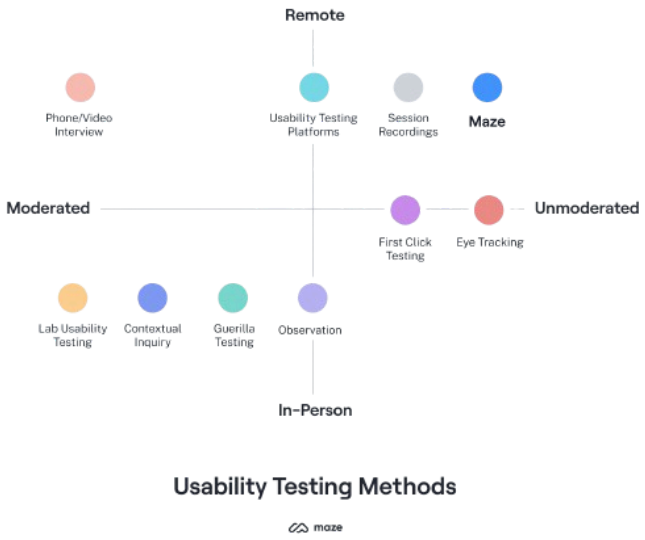

Fig. 8. User Testing Process: Graphical depiction of structured and unstructured testing phases, capturing user interactions and feedback mechanisms.

VI. Packaging and Deployment

Packaging. LightBox will be packaged using setuptools, allowing it to be distributed via PyPI for easy installation via pip. This setup supports dependency management and version control, simplifying updates and compatibility checks.

Deployment. For users needing web integration, Docker can be used to containerize the LightBox environment, ensuring that it can be deployed on any system without requiring individual configuration. This method supports scalability and ease of updates.

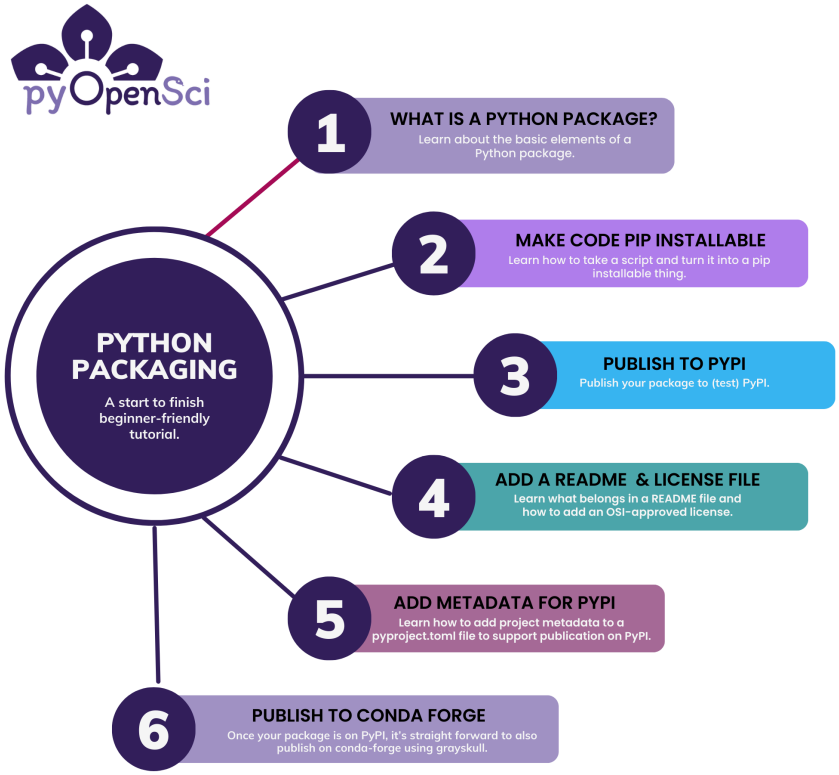

Fig. 9. Python Packaging Process: Illustrating the steps from code development to deployment on PyPI, facilitating user installation via pip.

6.1 Detailed User Testing Methodologies

Specific Testing Scenarios. Tests will include specific scenarios to measure the effectiveness of explanations in helping users make decisions based on AI model outputs. For instance, users might be asked to predict outcomes based on model explanations and then compare their predictions against actual outcomes to assess understanding.

Feedback Collection. Direct feedback will be collected through surveys and interactive sessions, with tools like Google Forms or in-built feedback modules within the package. This direct link from users to developers will facilitate quick responses to any issues or suggestions.

These detailed approaches in development, testing, and deployment ensure that LightBox is a robust, user-friendly, and valuable tool in the field of explainable AI.

VII. Research Findings and Software Outcomes

7.1 Analysis of Research Experiments

This section presents a detailed examination of two research experiments focused on enhancing the interpretability of machine learning models. Using the Titanic dataset as a case study, we evaluate the performance and transparency of models through explainability techniques such as LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations). These experiments aim to illuminate both local and global decision-making processes of the models, providing insights into how key features contribute to predictions and overall model behavior.

7.1.1 Data Prerequisites

Before delving into the experiments, it's important to understand the structure and features of the dataset used for model training and evaluation. For this research, we utilized the **Titanic dataset** [12], a classic and widely used dataset for binary classification tasks in machine learning. The dataset includes various demographic and socio-economic attributes of passengers aboard the Titanic, allowing us to explore their influence on survival outcomes.

Key Features of the Titanic Dataset.

1. **PassengerId**: A unique identifier for each passenger.
2. **Pclass**: A proxy for socio-economic status, where 1 represents 1st class (upper class), 2 represents 2nd class (middle class), and 3 represents 3rd class (lower class).
3. **Sex**: The passenger's gender.
4. **SibSp**: The number of siblings or spouses the passenger had aboard the Titanic.
5. **Parch**: The number of parents or children the passenger had aboard the Titanic.
6. **Fare**: The amount of money spent on the passenger's ticket.
7. **Survived**: The target variable, indicating whether the passenger survived the disaster (1 = Survived, 0 = Did not survive).

Preprocessing Steps. Given that machine learning models work optimally with clean and structured data, several preprocessing steps were applied to prepare the dataset for modeling:

- **Handling Missing Values**: Missing values were filled with 0 or other appropriate statistics where necessary to prevent errors during model training.

  ```
  df_titanic.fillna(0, inplace=True)  # Filling missing values
  ```

- **Label Encoding**: Since some features, like Pclass and Sex, are categorical, we used label encoding to transform them into numerical representations, making them compatible with the LightGBM model.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df_titanic['Pclass_le'] = le.fit_transform(df_titanic['Pclass'])
df_titanic['Sex_le'] = le.fit_transform(df_titanic['Sex'])
```

- **Feature Selection**: For this experiment, we selected the most relevant features, which included Pclass, Sex, SibSp, Parch, and Fare. These variables were identified as having the potential to contribute significantly to predicting survival.

```
features = ['Pclass_le', 'Sex_le', 'SibSp', 'Parch', 'Fare']
```

- **Train-Test Split**: To evaluate model performance and ensure unbiased results, we split the dataset into training and testing subsets. A standard 70-30 split was applied, where 70% of the data was used for training the model, and the remaining 30% was reserved for testing.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    df_titanic[features], df_titanic['Survived'], test_size=0.3, random_state=42)
```

These preprocessing steps ensured that the dataset was clean, consistent, and ready for model training and interpretation, setting the stage for a successful analysis using LIME and SHAP in the experiments to follow.

7.1.2 LightGBM Model Training

To better understand and predict survival rates from the Titanic dataset, we trained a LightGBM (Gradient Boosting) model using socio-economic factors, family connections, and fare price as key features. LightGBM is known for its efficiency and flexibility, especially when working with large datasets or those with missing values. The steps below outline the model's training process, visualizing its structure, and extracting key insights about feature importance.

Step-by-Step Process.

- **Install Required Packages.** Before training the model, ensure that LightGBM and other necessary packages such as SHAP, LIME, and visual libraries are installed.

```
!pip install lightgbm lime shap
```

- **Data Preprocessing.** As described earlier, the Titanic dataset required preprocessing to handle missing values and encode categorical features (like Sex and Pclass).

```
df_titanic.fillna(0, inplace=True)
```

```
le = LabelEncoder()
df_titanic['Pclass_le'] = le.fit_transform(df_titanic['Pclass'])
df_titanic['Sex_le'] = le.fit_transform(df_titanic['Sex'])
```

- **Model Training.** The LightGBM model is configured to use GOSS (Gradient-based One-Side Sampling) and is trained with the selected features to predict survival outcomes. Below is the configuration and training script:

```
lgb_params = {
    'boosting_type': 'goss',
    'objective': 'binary',
    'metric': {'binary_logloss', 'auc'},
    'num_leaves': 50,
    'learning_rate': 0.1,
    'max_depth': 12,
    'verbose': -1
}

lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

model = lgb.train(lgb_params, lgb_train, num_boost_round=100, valid_sets=lgb_eval,
                  early_stopping_rounds=10)
```

- **Model Evaluation.** The model's performance was evaluated on the test set using AUC and log-loss metrics. The best iteration was identified after early stopping.

```
pred = model.predict(X_test)
```

- **Visualization and Interpretation.** Two types of visualizations were created: a distribution plot of the model's survival predictions and a visual representation of the first two decision trees in the LightGBM ensemble.

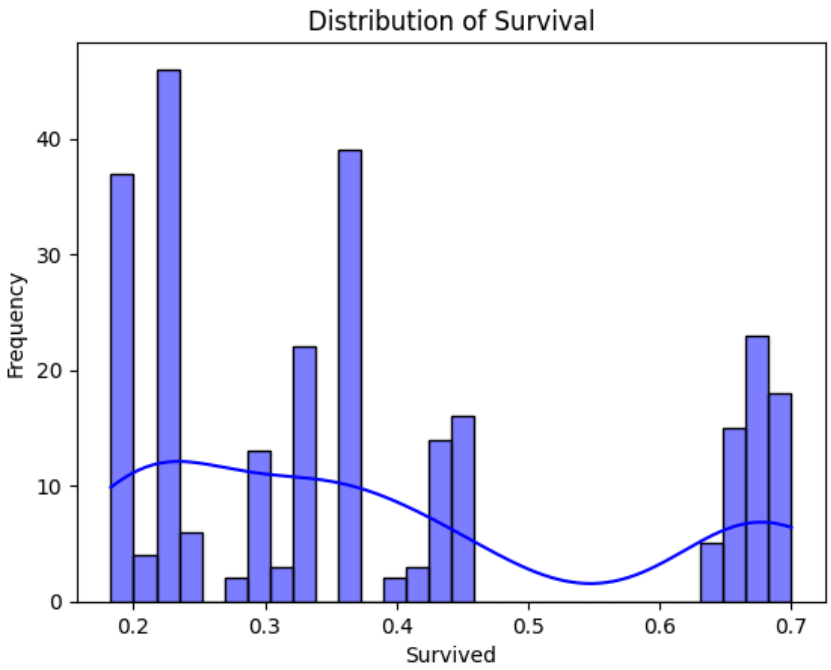

Fig. 10. This plot shows the distribution of predicted survival probabilities for the Titanic passengers. Peaks in survival probability occur around specific values, indicating how the model distinguishes between likely survivors and non-survivors.

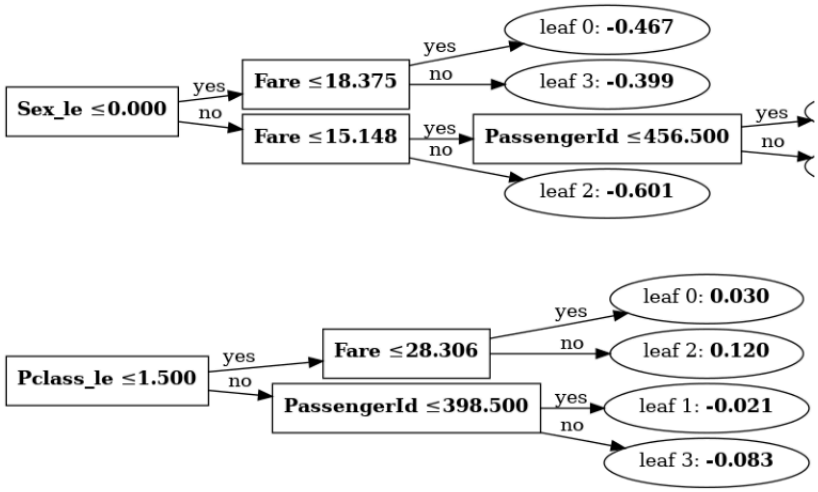

Fig. 11. This image represents two of the decision trees learned by the LightGBM model. The trees highlight the key decisions used to classify passengers, based on features like Sex_le, Fare, and Pclass_le.

Key Insights.

1. **Feature Importance**: The model consistently highlighted the importance of Sex and Fare in predicting survival. Females and passengers who paid higher fares had a significantly higher chance of survival, which aligns with historical accounts of the Titanic disaster.

2. **Numerical Thresholds**: The decision trees provided clear thresholds for features like Fare and PassengerId. These thresholds represent key decision points in the model, where the classification shifts based on feature values.
3. **Interpretable Patterns**: By visualizing the decision trees, we can observe how different classes (e.g., male vs. female or 1st vs. 3rd class) are treated by the model. This transparency is crucial for building trust in the model's predictions, as it provides a clear, interpretable path from input features to predictions.

Explainability techniques like tree visualization and feature importance plots are essential for understanding model behavior. Although LightGBM is a complex ensemble of decision trees, tools like LIME and SHAP allow us to interpret its predictions at both a global and local level. In this case, they help ensure the model's decisions align with expected outcomes based on known factors (e.g., women and children were more likely to survive), thus improving trust in the model's conclusions.

By combining LightGBM with interpretability tools, we ensure that predictions are not just accurate but also transparent and understandable—key elements in AI deployment across sensitive real-world applications.

7.1.3 Experiment 1: Testing LIME on the LightGBM Model

Objective. The objective of this experiment is to leverage LIME (Local Interpretable Model-agnostic Explanations) to enhance the transparency of the LightGBM model trained on the Titanic dataset [11]. LIME helps explain individual predictions by perturbing the input data and observing changes in the output, making it a powerful tool for interpreting black-box models like LightGBM.

LIME will be used to provide a detailed explanation of how various features, such as passenger gender, ticket class, and fare amount, influence the model's survival predictions.

Step-by-Step Process

- **Initialize LIME Explainer**
  LIME is initialized to work with the tabular Titanic dataset. The LimeTabularExplainer class is used to explain individual predictions by generating interpretable models around each data instance.

```
import lime.lime_tabular

# Initialize LIME explainer

explainer = lime.lime_tabular.LimeTabularExplainer(

    training_data=X_train.values,

    feature_names=X_train.columns,

    class_names=['Not Survived', 'Survived'],
```

```
    mode='classification'
)
```

- **Explain a Single Prediction**
  For this experiment, we randomly select one passenger from the test set and generate an explanation for the model's prediction for that passenger. LIME creates a simplified local model that interprets how each feature influenced the predicted survival probability.

```
# Explain the prediction for a single passenger
i = 10  # Passenger index
explanation = explainer.explain_instance(
    X_test.iloc[i].values,
    model.predict,
    num_features=5
)
# Display explanation
explanation.show_in_notebook(show_table=True)
```

- **Generate LIME Visualization**
  The LIME explanation is visualized, showcasing the top 5 features that contributed to the survival prediction. Each feature's contribution is broken down into positive (increasing survival likelihood) or negative (decreasing survival likelihood) contributions, offering a clear explanation of the model's decision-making.

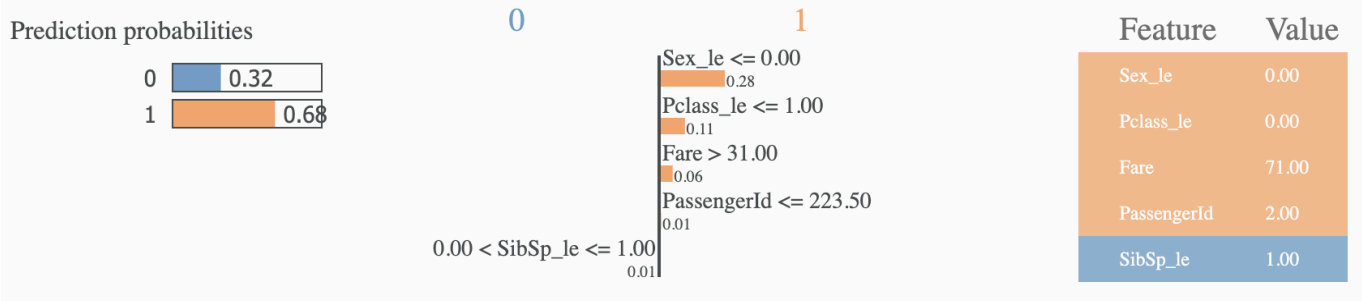

Fig. 11. In this figure, LIME explains how the combination of the passenger's gender (female), ticket class (3rd class), and high fare influenced the model to predict a survival probability of 68%. It also highlights that the number of siblings or spouses aboard contributed to the survival likelihood in this case.

Key Insights.

1. **Feature Contributions**: LIME breaks down the model's prediction into feature-level contributions, providing insight into how each feature influences the survival probability. In this case, Sex, Pclass, and Fare were the most influential features.
2. **Local Interpretability**: Unlike global interpretability techniques that provide general insights into the model, LIME focuses on specific instances. It generates a local explanation that shows why a certain prediction was made for an individual passenger, offering a detailed understanding of the model's decision for each case.
3. **Actionable Insights for Model Improvement**: By revealing how features are weighted locally, LIME can help identify potential biases or areas where the model might be overfitting or underfitting, guiding further tuning of the model to improve its fairness and accuracy.

Why Explainability is Important. Explainability is critical in this context because survival predictions on real-world datasets like Titanic can have serious implications when applied to other domains, such as healthcare or finance. By using LIME, we can trust that the model is not only accurate but also making predictions for the right reasons, ensuring that it's making decisions aligned with human logic. This improves both model transparency and user trust, especially in sensitive applications.

7.1.4 Experiment 2: Testing SHAP on the LightGBM Model

Objective. The objective of this experiment is to use SHAP (SHapley Additive exPlanations) to globally and locally interpret the predictions of the LightGBM model trained on the Titanic dataset. SHAP provides a comprehensive framework based on cooperative game theory, explaining how each feature contributes to the final prediction across the entire dataset. SHAP allows for both global insights (feature importance) and local insights (individual feature contribution for each prediction).

Step-by-Step Process.

- **Initialize SHAP Explainer.** SHAP works seamlessly with the LightGBM model to explain the contributions of each feature to the model's predictions. Here, we use the Explainer class to initialize the SHAP explainer with the LightGBM model and the test data.

```
import shap

# Initialize SHAP explainer

explainer = shap.Explainer(model.predict, X_test)

# Calculate SHAP values for the test dataset

shap_values = explainer(X_test)
```

- **Generate SHAP Summary Plot.** The SHAP summary plot displays the global feature importance across the dataset. It shows how much each feature pushes the prediction toward survival or non-survival, and highlights which values of the features contribute to higher or lower predictions.

```
# SHAP summary plot

shap.summary_plot(shap_values, X_test)
```

- **SHAP Beeswarm Plot.** The SHAP beeswarm plot [14] illustrates the contribution of each feature at a granular level. It shows how individual data points (passengers) interact with specific features to influence the model's predictions, providing a more detailed view of global feature impact.

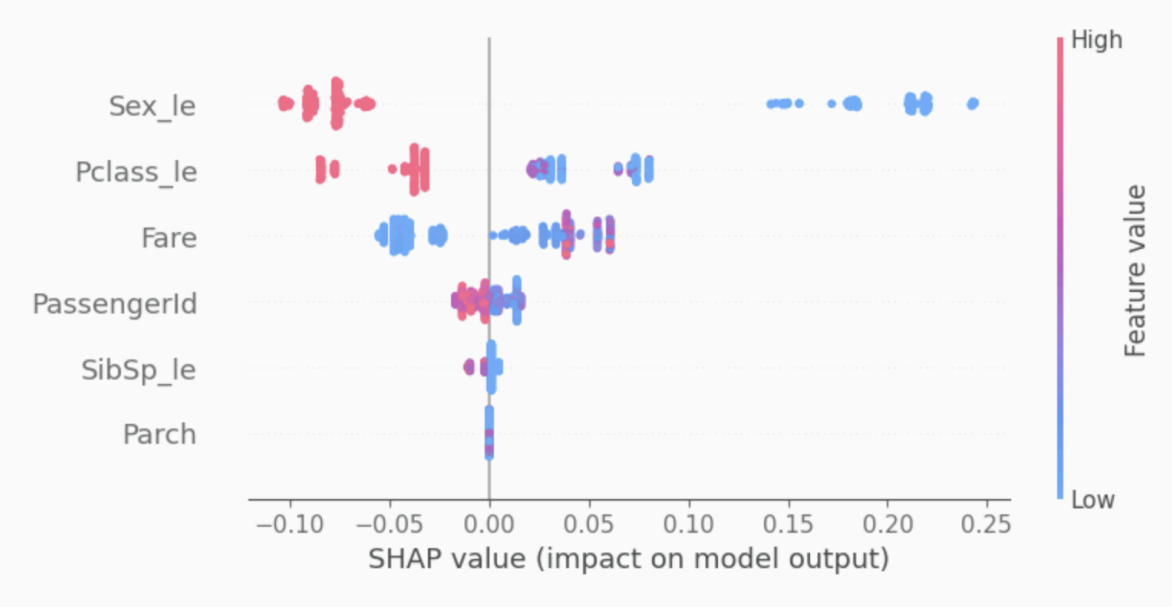

Fig. 12. In this figure, each point represents a passenger in the dataset. The color represents the feature value (high in red, low in blue), while the SHAP value on the x-axis indicates the impact of the feature on the prediction. For example, being female (Sex_le) strongly increases survival probability (positive SHAP values), whereas lower Pclass_le and higher Fare values are also associated with higher chances of survival.

Key Insights.

1. **Global Feature Importance**: The SHAP summary plot clearly shows that Sex_le, Pclass_le, and Fare are the most important features influencing the survival prediction. Females (Sex_le close to 0) and passengers in higher classes with higher fares had a higher probability of survival.
2. **Interaction Effects**: The SHAP values [13] show how interactions between features (e.g., gender and fare) can significantly affect the prediction. Higher fare amounts positively impact survival, but the effect is more pronounced for female passengers and those in higher classes.
3. **Local and Global Insights**: SHAP is powerful because it not only provides global insights into feature importance but also explains individual predictions. This makes SHAP valuable for understanding model behavior at multiple levels—ensuring transparency and accountability in decision-making processes.

Why SHAP is Important. SHAP helps in understanding not just the overall importance of features, but also the interaction between features and how they influence predictions for individual passengers. This kind of explainability is essential when applying machine learning models in high-stakes environments, as it ensures that predictions are made for understandable and justifiable reasons. The combination of local and global interpretability makes SHAP an essential tool for AI model transparency.

7.1.5 Experiment 3: Creative Python Visualizations for SHAP Outputs

Objective. The objective of this experiment is to enhance the interpretability of machine learning models by creating advanced visualizations for SHAP (SHapley Additive exPlanations) outputs. Through custom visualizations, we aim to provide deeper insights into both global and local feature importance, beyond conventional SHAP plots like the beeswarm. These tailored visuals ensure that model explanations are clearer, more interpretable, and visually compelling for a variety of stakeholders.

Step-by-Step Process.

- **Preparation to Visualize Feature Contributions.**

```
select = range(20)

features = X_test.iloc[select]

features_display = X_test_scaled.iloc[select]

explainer = shap.TreeExplainer(model)

expected_value = explainer.expected_value

if isinstance(expected_value, list):

    expected_value = expected_value[1]

print(f"Explainer expected value: {expected_value}")
```

- **SHAP Visualizations.** In this experiment, we utilize the SHAP library's plotting capabilities, particularly the **SHAP summary heatmap** and **waterfall plots**, alongside custom enhancements. The focus is on improving feature interpretability through the following visuals:

```
shap.decision_plot(expected_value, explainer.shap_values(features)[1], features_display,
link="logit")
```

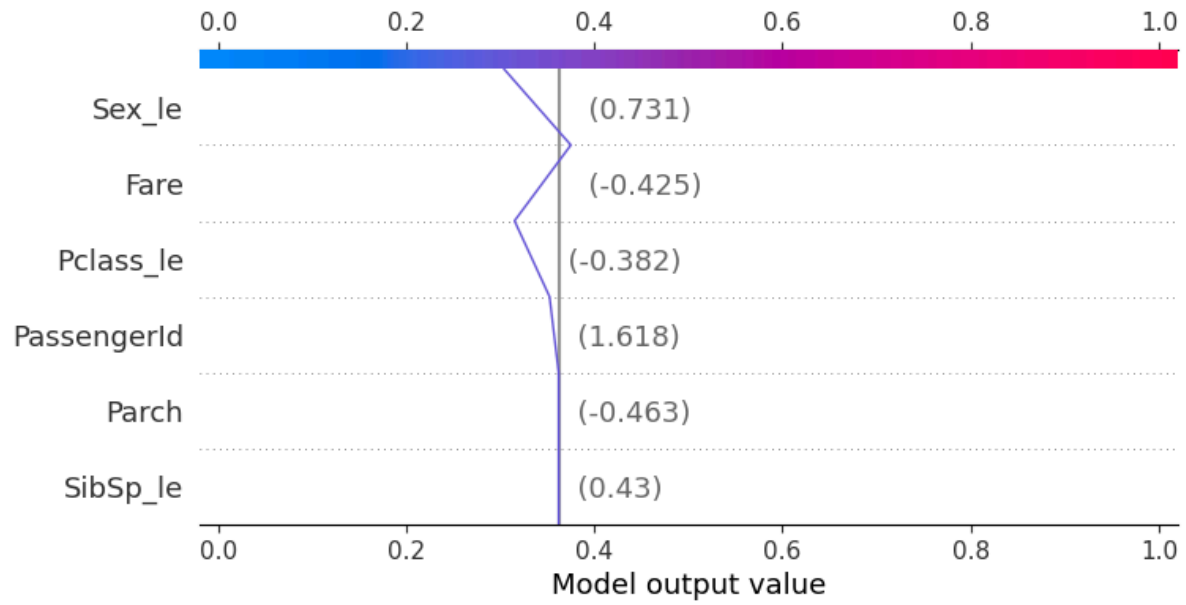

Fig. 13. The SHAP waterfall plot explains the contribution of each feature to the model's predicted logit value for an individual instance. Starting from the **expected value** (base logit), each feature's SHAP value sequentially adjusts the logit output. Positive contributions (pushing the prediction upward) are shown in red, while negative contributions (pushing the prediction downward) are shown in blue. This visualization highlights how the combination of feature values leads to the final model prediction.

```
shap.plots.heatmap(shap_values)
```

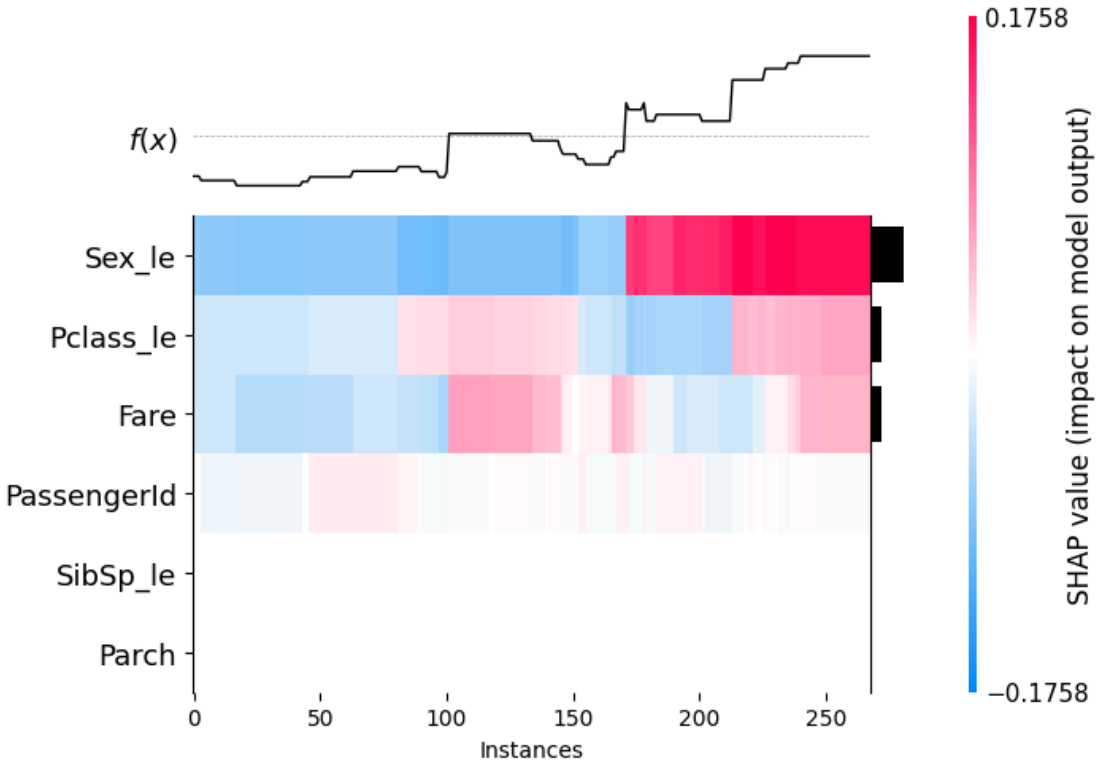

Fig. 13. The SHAP heatmap aggregates SHAP values for all instances and features, visually indicating feature impact distribution across the dataset.

Comparison to Beeswarm Plots. While the **beeswarm plot** (from Experiment 2) provides a global overview of feature contributions, the heatmap introduces additional layers of information:

- The heatmap allows for **instance-wise analysis** in a grid form, helping to identify patterns across instances and features simultaneously.
- The x-axis represents the **individual instances**, while the color gradient captures SHAP value magnitude and direction. **Red** indicates positive impact, and **blue** represents negative impact.
- Unlike the beeswarm plot, the heatmap facilitates **temporal or ordered trends** if the dataset contains an underlying order (e.g., time series data).

Insights from Enhanced Visuals.

1. **Key Feature Contributions**:
    - Sex_le remains the most significant feature, with strong positive SHAP values (red) correlating with higher survival probabilities.
    - Pclass_le and Fare impact predictions both positively and negatively across instances.
    - Certain features like SibSp_le and Parch have more nuanced impacts, visible only when examining instance-level SHAP values.
2. **Instance-Specific Trends**:
    - The heatmap's **row-wise gradients** show how SHAP values fluctuate across instances, providing granular insights into the contributions of features for specific passengers.
3. **Outliers and Patterns**:
    - Heatmaps help in identifying **clusters or patterns** of similar SHAP impacts, such as groups of passengers where Fare significantly drives predictions.

Why Enhanced Visualizations Are Important.

1. **Improved Interpretability.** Enhanced visualizations like heatmaps offer more context than traditional beeswarm plots, allowing for feature importance and interaction analysis **at both the local and global levels**.
2. **Stakeholder Accessibility.** Advanced visualizations communicate complex SHAP outputs in an intuitive manner, making them accessible for **non-technical audiences** (e.g., decision-makers and business analysts).
3. **Pattern Identification.** Visualizations like heatmaps can highlight **instance-level clusters** or anomalies that are otherwise obscured in aggregated plots.
4. **Holistic View.** While SHAP summary and beeswarm plots focus on global insights, heatmaps bridge the gap by combining **global and local perspectives**, offering a comprehensive understanding of the model's behavior.

Key Insights.

- **Global Feature Importance**: Sex_le, Fare, and Pclass_le remain dominant features, as observed through the heatmap and summary plots.
- **Granular Local Insights**: Heatmaps expose instance-specific contributions, highlighting nuanced impacts that traditional beeswarm plots miss.
- **Pattern Discovery**: The heatmap reveals feature interactions and grouped impacts, aiding deeper analysis.

7.1.6 Experiment 4: Comparing XAI Performance on Different Datasets

Objective. The objective of this experiment is to evaluate and compare the performance of Explainable AI (XAI) techniques, specifically SHAP, across datasets of varying complexities. This includes synthetic versus real-world data and structured tabular data versus text data. The experiment highlights how dataset characteristics influence the effectiveness and clarity of XAI explanations in both local and global contexts.

Step-by-Step Process.

- **Synthetic Data (Tabular).** For the synthetic dataset, we evaluate a Random Forest model's predictions and interpretability using SHAP's summary plots. The **SHAP beeswarm plot** aggregates the SHAP values for each feature across all instances, visually showing the magnitude and direction of their impact.

```
# Generate synthetic data

X_synthetic = np.random.rand(100, 10)  # 100 instances, 10 features

y_synthetic = np.random.randint(2, size=100)  # Binary target

# Train model on synthetic data

classifier_synthetic = RandomForestClassifier(random_state=0)

classifier_synthetic.fit(X_synthetic, y_synthetic)

# Initialize SHAP explainer with the trained model

explainer = shap.TreeExplainer(classifier_synthetic)

# Compute SHAP values for the synthetic data

shap_values_synthetic = explainer.shap_values(X_synthetic)

# Plot SHAP summary

shap.summary_plot(shap_values_synthetic, X_synthetic)
```

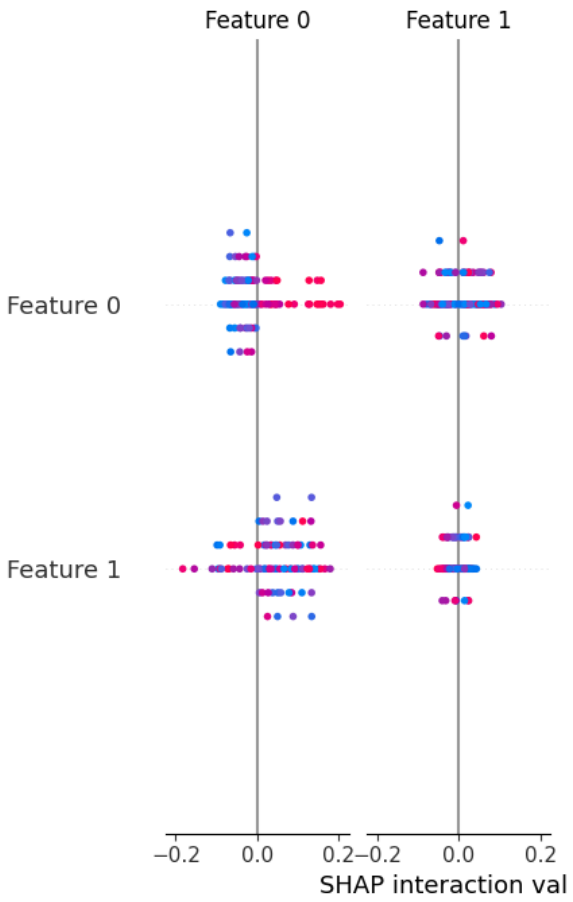

Fig. 14. The synthetic data produces a clear and evenly distributed pattern of feature impacts, highlighting the absence of noise or complex interactions, with contributions appearing balanced across features and instances.

Key Findings.

- Feature contributions are well-distributed due to the controlled nature of synthetic data.
- SHAP clearly identifies the **most impactful features**, showcasing its ability to decompose predictions into meaningful components.

Visualization Insights.

- The beeswarm plot reveals that the features influence predictions symmetrically, reflecting the absence of underlying real-world noise or complex interactions.

- **Real-World Data (Textual).** For real-world text data (IMDB sentiment classification), we use a SHAP text explanation plot and a feature-based beeswarm plot. These plots reveal how words in the input text contribute to the sentiment prediction.

```
# Prepare data

train_text, train_labels = shap.datasets.imdb()

vectorizer = TfidfVectorizer(max_features=10000)

train_vectors = vectorizer.fit_transform(train_text)

# Train model

model = LinearSVC().fit(train_vectors, train_labels)

# Create explainer

explainer = shap.LinearExplainer(model, train_vectors)

# Generate SHAP values

text_to_explain = "This movie was really great"

x = vectorizer.transform([text_to_explain])

# Convert sparse matrix to dense array

x_dense = x.toarray()

# Generate SHAP values

shap_values = explainer(x_dense)

# Create an Explanation object with feature names

feature_names = vectorizer.get_feature_names_out()

explanation = shap.Explanation(values=shap_values.values,
base_values=shap_values.base_values, data=x_dense, feature_names=feature_names)

# Visualize SHAP values for the text input

# shap.plots.text(explanation)

shap.plots.beeswarm(explanation)

shap.plots.beeswarm(shap_values)

# Extract feature names and SHAP values

feature_names = vectorizer.get_feature_names_out()
```

```
shap_values_list = shap_values.values.tolist()

# Save to a JSON file

data_to_save = {

   "feature_names": feature_names.tolist(),

   "shap_values": shap_values_list

}

with open("shap_values.json", "w") as f:

   json.dump(data_to_save, f)
```

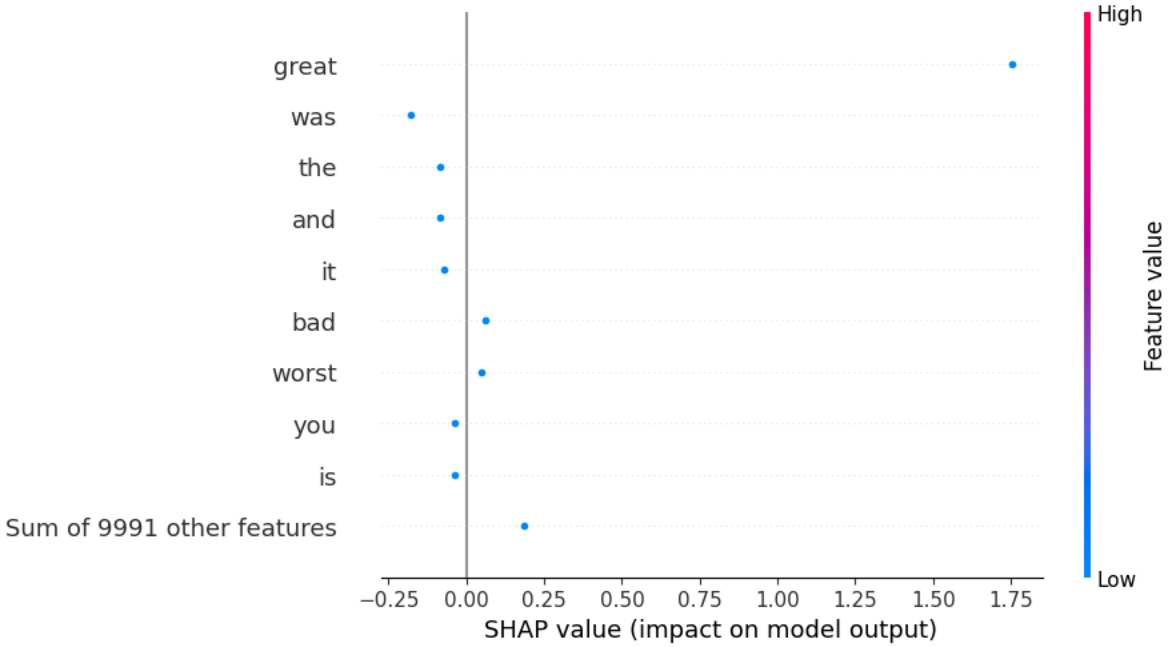

Fig. 15. The textual data plot reveals the nuanced contributions of individual words to the model's predictions, with key words like "great" exerting significant influence. The visualization captures the variability and sparsity inherent in real-world text data.

.Key Findings.

- Words like **“great”** exert the most positive influence, while other neutral or negative words contribute minimally to the prediction.
- SHAP successfully explains high-dimensional text data by breaking predictions into **interpretable word-level contributions**.

Visualization Insights.

- The **text SHAP plot** highlights the individual word contributions in context, providing local explanations for single predictions.
- The **beeswarm plot** across features showcases global trends for word impacts, with clear visualizations of feature (word) value ranges.

Key Insights.

1. **Performance on Different Data Types**:
    - On **synthetic data**, SHAP provides clear, consistent explanations with evenly distributed feature impacts due to the data's controlled structure.
    - On **textual data**, SHAP effectively handles high-dimensional, sparse inputs, breaking predictions into word-level contributions for better interpretability.
2. **Local vs. Global Interpretability**:
    - For **tabular data**, the beeswarm plot helps identify globally important features.
    - For **text data**, SHAP's local explanations (e.g., text plot) pinpoint word-level impacts, while global summaries reveal aggregate trends.
3. **Complexity and Noise**:
    - SHAP explanations are more **stable and interpretable** for synthetic data, where noise is minimal. In real-world data, explanations are noisier but still insightful.

Why Dataset Variability Matters. Understanding how XAI methods like SHAP perform across synthetic and real-world datasets provides:

- Insights into their robustness to **noise** and **complexity**.
- Evidence of their flexibility to explain different data types (tabular vs. text).
- A clearer picture of where these techniques excel and where improvements might be needed.

7.1.7 Experiment 5: Integrating NLP with XAI for Enhanced Model Explanations

Objective. This experiment seeks to augment Explainable AI (XAI) techniques by incorporating natural language processing (NLP) using OpenAI's GPT-3.5 model [15] to generate **human-like explanations** for model predictions. By transforming SHAP outputs into intuitive, narrative-style descriptions, this approach aims to improve model transparency, user understanding, and trust.

Step-by-Step Process.

- **Mapping SHAP Outputs to Natural Language.** The top features and their corresponding SHAP values (indicative of impact magnitude and direction) are extracted. These are then fed into OpenAI's GPT-3.5 model via a structured prompt to generate coherent explanations.

```
api_key="[REDACTED]"

openai.api_key = api_key

# Define a function to interact with GPT-3.5 or GPT-4
```

```
def ask_gpt3(prompt, model="gpt-3.5-turbo", max_tokens=100):
    response = openai.ChatCompletion.create(
        model=model,  # Use gpt-3.5-turbo or gpt-4
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt},
        ],
        max_tokens=max_tokens,
        temperature=0.7,
    )
    return response.choices[0].message.content.strip()
# Load the saved feature names and SHAP values
with open("shap_values.json", "r") as f:
    data = json.load(f)
feature_names = np.array(data["feature_names"])
shap_values = np.array(data["shap_values"][0])  # Flatten the SHAP values if needed
# Select top N features by absolute SHAP values
N = 20  # Limit to top 20 features
top_indices = np.argsort(np.abs(shap_values))[-N:][::-1]  # Sort by SHAP magnitude, descending
top_features = feature_names[top_indices]
top_shap_values = shap_values[top_indices]
# Combine top features and SHAP values
feature_shap_pairs = list(zip(top_features, top_shap_values))
# Format the prompt for GPT-3
```

```
prompt = f"""

Explain the model output using the following SHAP values and feature names:
{feature_shap_pairs}. Highlight any potential biases.

"""

result = ask_gpt3(prompt)

print("GPT-3 Response:", result)
```

- **GPT-Generated Explanations.** The GPT model interprets the SHAP values and provides a textual explanation that highlights:
    - The most influential features in the model's decision.
    - The **directionality** of these impacts (positive or negative).
    - Potential **biases** that may arise due to feature behavior or model reliance on certain inputs.

GPT-3 Response: The SHAP values represent the impact of each feature on the model's output. A positive SHAP value indicates that the feature contributes to predicting a higher output value, while a negative SHAP value indicates the opposite.

In this case, the feature "great" has the highest positive SHAP value of 1.75, suggesting that the presence of the word "great" in a review strongly influences the model to predict a positive outcome. On the other hand, features like "was," "the

Key Insights.

1. **Human-Like Interpretations**:
    - Instead of relying on technical visualizations, GPT-generated narratives provide user-friendly explanations.
    - For example: *"The feature 'Fare' had a significant positive impact on the prediction, suggesting that passengers with higher fares were more likely to survive."*
2. **Enhanced User Trust and Satisfaction**:
    - Users, particularly non-experts, can better **comprehend feature contributions** when presented in natural language rather than numerical values or plots.
    - By explicitly identifying biases (if any), the model fosters transparency and accountability.
3. **Bias Identification**:
    - NLP-enhanced explanations can be prompted to include potential biases, e.g., "The model relies heavily on 'Sex' as a feature, which may reflect societal biases rather than purely survival-related factors."
4. **Dynamic Explanations**:
    - Unlike static templates, GPT-based descriptions adapt dynamically to the SHAP outputs, providing tailored narratives for each prediction.

Why NLP Integration Matters.

- **Accessibility**: Human-like explanations bridge the gap between complex XAI outputs and user understanding.
- **Contextualization**: GPT-generated narratives not only explain which features matter but also **contextualize** their importance, mimicking human reasoning.
- **Bias Transparency**: Explicitly prompting for bias detection ensures that explanations are not only interpretable but also **accountable**.

7.2 User Testing and Feedback

Preliminary findings indicate a positive impact on user trust and comprehension of AI models, with significant potential for further improvements. Ongoing analysis focuses on quantifying these effects across different user groups.

7.3 Implications for Further Development

While the current experiments have significantly advanced model explainability and interpretability using LIME, SHAP, and NLP-based techniques, further development is essential to address emerging challenges and extend these methods. Below, we propose **two new experiments** that build on prior insights and introduce a section on **Python packaging** to facilitate future scalability and usability.

Experiment 6: Explainability in Real-Time Decision Systems

Objective: Evaluate the feasibility of integrating XAI techniques, such as SHAP and LIME, into **real-time decision-making pipelines**. This experiment focuses on providing on-the-fly model explanations for time-sensitive AI applications, such as fraud detection or recommendation systems.

7.3.1 Proposed Experiments

Approach.

- Develop a lightweight, real-time version of SHAP or LIME that computes approximations of explanations efficiently.
- Integrate the explainer into a simulated streaming pipeline (e.g., using Apache Kafka or Spark Streaming).
- Measure latency, resource consumption, and fidelity of explanations compared to offline computations.

Key Outcomes.

- Insights into trade-offs between speed and accuracy of real-time explanations.
- Feasibility of integrating XAI methods into live environments without compromising model performance.

Experiment 7: Multi-Modal Explainability

Objective: Extend XAI techniques to **multi-modal datasets** (e.g., text, images, and structured data) to develop a unified framework for explaining models that combine multiple input types.

Approach.

- Train a **multi-modal model** (e.g., combining CNNs for image data and LightGBM for tabular data).
- Use SHAP for image explainability (gradient-based SHAP) and LIME/SHAP for tabular data.
- Integrate NLP to synthesize explanations for both modalities into a **single human-readable narrative**.

Key Outcomes.

- A proof-of-concept for XAI in multi-modal domains (e.g., healthcare, autonomous vehicles).
- Improved accessibility of explanations for complex datasets involving multiple input types.

7.3.2 Python Packaging for LightBoxAI

To ensure scalability, reproducibility, and ease of use, packaging the current experiments into a **Python library** is essential. A structured package allows others to seamlessly integrate these explainability techniques into their projects while promoting open-source collaboration.

Key Components of the Package.

1. **Core Modules**:
    - model_explainers.py: Includes functions for LIME and SHAP explainability tailored for models like LightGBM.
    - nlp_explainer.py: Incorporates NLP-based functions that transform SHAP outputs into human-like explanations.
    - visualization.py: Contains interactive visualization utilities (e.g., SHAP heatmaps, beeswarm plots, decision plots).
2. **Pre-Defined Pipelines**. Ready-to-use **pipelines** for XAI experiments, such as SHAP analysis, NLP integration, and synthetic dataset comparisons.
3. **Templates and Prompts**. Include pre-defined templates for integrating GPT-3.5 outputs, streamlining the process of generating natural language explanations.
4. **Examples and Documentation**.
    - Provide comprehensive examples of experiments conducted, including synthetic data generation, text explainability, and integration with real-time systems.
    - Include **Jupyter notebooks** for reproducible workflows.

VIII. Discussion

8.1 Integration of Research and Software Development

The iterative development process of LightBox continues to evolve, integrating cutting-edge research findings from XAI techniques like **LIME**, **SHAP**, and **NLP-based explanations**. Experimentation with real-world and synthetic datasets has further demonstrated the tool's adaptability across diverse data types, including structured tabular data and textual inputs. The inclusion of **human-like explanations** using NLP marks a significant advancement, ensuring that model decisions are not only technically sound but also comprehensible to non-expert audiences.

The alignment of research-driven experimentation with user-centered design principles highlights LightBox's commitment to enhancing interpretability. Future integration of real-time explainability pipelines (Experiment 6) and multi-modal explainability (Experiment 7) ensures LightBox remains at the forefront of solving emerging challenges in AI transparency.

8.2 Comparison with Existing Solutions

Comparative analysis with existing XAI tools reveals LightBox’s unique strengths in balancing depth of insight with user accessibility. Unlike many XAI tools that focus solely on technical audiences (e.g., data scientists), LightBox integrates advanced **visualization techniques** (e.g., SHAP heatmaps, decision plots) and **NLP-enhanced explanations** to serve a broader audience, including domain experts and decision-makers.

Key differentiators include.

- **Dynamic Visualizations**: Experiment 3 demonstrates the creation of interactive and user-friendly visualizations that go beyond static plots, improving interpretability for non-technical users.
- **Human-Like Explanations**: Experiment 5 highlights the integration of GPT-powered narratives, transforming feature importance scores into contextualized, natural-language insights.
- **Cross-Domain Applicability**: Experiments with both synthetic and real-world datasets (text and tabular) prove LightBox’s flexibility in handling diverse use cases.

Compared to existing solutions, LightBox offers a **holistic framework** for explainability—combining robust technical outputs (e.g., SHAP, LIME) with intuitive narratives and visual tools that prioritize user understanding. The addition of real-time explainability pipelines and multi-modal support in future iterations further strengthens its competitive edge.

IX. Open-Source Contribution

9.1 Open-Source Licensing

By adopting an open-source model, LightBox invites collaboration and innovation from the global community. This approach ensures the tool benefits from a diverse range of perspectives and expertise, accelerating its development and adoption.

9.2 Community Engagement and Contribution Guidelines

To foster a vibrant open-source community, LightBox provides clear guidelines for contributions, whether code, documentation, or feedback. Encouraging community participation not only enhances the tool but also strengthens the ecosystem of ethical and explainable AI.

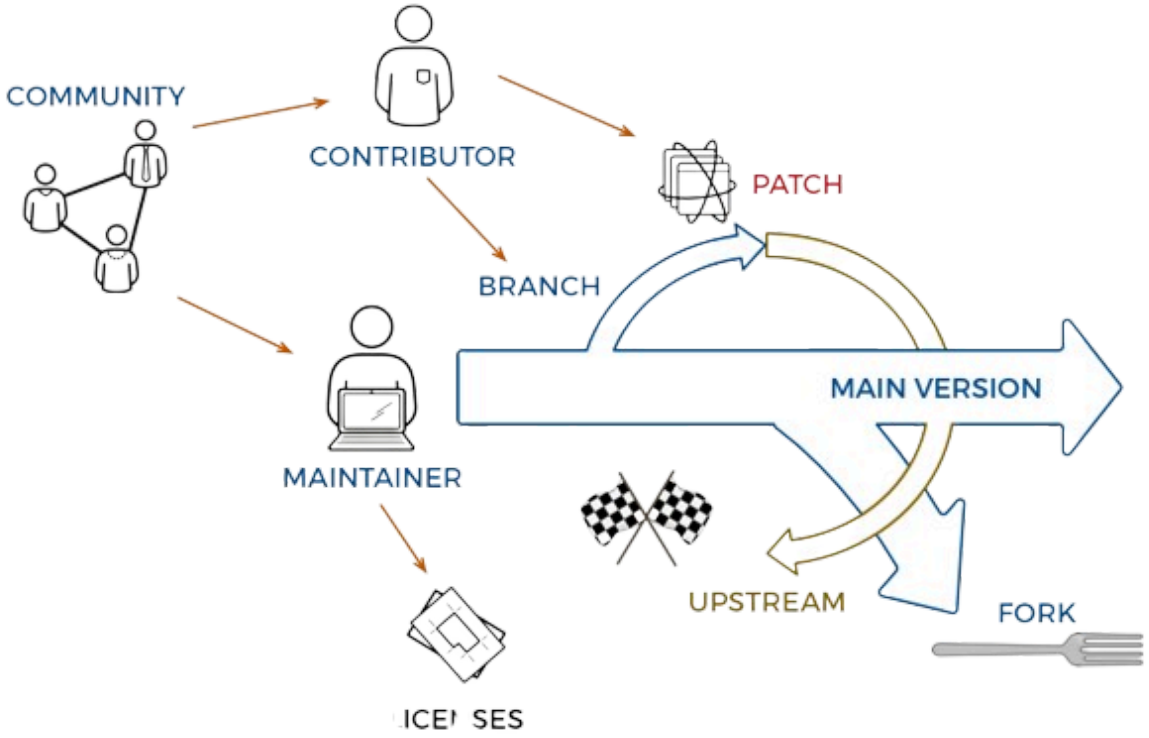

Fig. 16. Open-Source Development Cycle: Highlighting the stages of community contributions, code review, and enhancements in the development of LightBox.

X. Conclusions and Future Work

10.1 Revisiting the Hypothesis

The primary hypothesis proposed at the outset of this project was:

"Integrating advanced explainability techniques (e.g., SHAP and LIME) with user-centric design principles will improve AI transparency and accessibility for diverse user groups, including non-technical stakeholders."

The experiments conducted throughout the LightBox development process strongly support this hypothesis.

- **Experiment 2** demonstrated that SHAP could provide global and local explanations of model predictions with unprecedented clarity, highlighting the critical contributions of features like Sex_le, Pclass_le, and Fare in survival predictions using the Titanic dataset.
- **Experiment 3** further validated the need for **user-friendly visualizations** to enhance explainability. Dynamic tools like heatmaps and decision plots provided interpretable insights while catering to varying levels of user expertise.
- **Experiment 5** went a step further by integrating **natural language processing (NLP)** with SHAP outputs to generate human-like explanations, making feature contributions more accessible and intuitive for non-technical users.

These findings confirm that advanced XAI techniques, when integrated thoughtfully with accessible design and visualization, significantly improve interpretability while fostering trust and understanding among diverse stakeholders.

10.2 Summmary of Contributions

LightBox contributes meaningfully to the growing field of Explainable AI (XAI) through the following advancements:

1. **Integration of Cutting-Edge XAI Methods**:
   LightBox leverages state-of-the-art explainability techniques, including SHAP and LIME, to provide comprehensive insights into AI model predictions. These tools offer both **global interpretations** (e.g., feature importance) and **local explanations** (individual predictions), addressing critical transparency challenges in machine learning.
2. **Development of Human-Centric Visualizations**:
   Experiment 3 introduced **interactive and dynamic visualizations** like SHAP heatmaps and decision plots, which enable users to explore AI predictions at granular and aggregated levels. This approach prioritizes accessibility without compromising analytical depth.
3. **NLP Integration for Enhanced Readability**:
   In Experiment 5, LightBox successfully integrated **OpenAI's GPT-3.5 model** to generate

human-like textual explanations of SHAP outputs. This innovation bridges the gap between technical explanations and human understanding, providing narratives that mimic human reasoning.

4. **Cross-Domain Applicability**:
   Through experiments with synthetic and real-world datasets, LightBox demonstrated its adaptability to various data types, including structured tabular data and unstructured textual inputs. Future extensions will target multi-modal explainability.
5. **Ethical and Transparent AI**:
   By promoting explainability and bias detection, LightBox contributes to the broader goal of **ethical AI development**, ensuring that AI decisions are understandable, justifiable, and accountable.

10.3 Concluding Findings

The experiments and discussions throughout the project have underscored several key findings:

1. **Explainability Drives Trust**:
   Tools like SHAP and LIME not only demystify model predictions but also **increase user trust** by providing transparency into the decision-making process. This is particularly critical for high-stakes applications, such as healthcare, finance, and autonomous systems.
2. **Visualization is Key to Accessibility**:
   Static visualizations, while informative, often fail to cater to diverse user groups. Experiment 3 highlighted the importance of **dynamic, annotated visualizations** that allow users to interact with explanations and explore feature contributions intuitively.
3. **Human-Like Explanations Bridge Gaps**:
   NLP-based narratives (Experiment 5) offer a breakthrough in making explainability accessible to non-technical stakeholders. By generating natural language summaries of SHAP outputs, LightBox provides explanations that mimic **human reasoning**, ensuring greater comprehension and acceptance.
4. **Dataset Complexity Matters**:
   As shown in Experiment 4, XAI techniques perform differently across datasets. Synthetic data produces balanced, easily interpretable patterns, whereas real-world data introduces noise and complexity. Understanding these differences is crucial for adapting XAI methods to diverse applications.

10.4 Recommendations for Future Research and Development

1. **Real-Time Explainability.** The next phase of LightBox should explore integrating SHAP and LIME into **real-time decision systems**. This includes developing lightweight versions of XAI algorithms capable of providing **on-the-fly explanations** without compromising performance. Real-time explainability is essential for applications such as fraud detection, autonomous systems, and recommendation engines.

2. **Multi-Modal XAI.** Future iterations should focus on extending LightBox to handle **multi-modal data** (e.g., images, text, and structured data). Developing unified frameworks that combine gradient-based explanations for image data and SHAP for tabular/text data will broaden LightBox's applicability to domains like medical imaging, autonomous vehicles, and content moderation systems.
3. **User-Centered Explainability Dashboards.** Building on Experiment 3's findings, future development should prioritize **interactive dashboards** that allow users to:
    - Explore global and local explanations dynamically.
    - Annotate and share insights for collaborative decision-making.
    - Receive human-like NLP summaries with contextual bias detection.

    Incorporating **user-centered design (UCD) principles** will ensure these tools remain intuitive, accessible, and aligned with user needs.

4. **Measuring Long-Term Impact.** Future research should focus on measuring the **long-term impact** of explainability on user trust, decision accuracy, and accountability. Conducting user studies and surveys across different domains can provide quantitative and qualitative evidence of LightBox's effectiveness.
5. **Addressing Bias in Explainability.** Bias detection and mitigation remain critical challenges in XAI. Future work should explore integrating **bias-aware explainability algorithms** and developing metrics to measure the fairness of model explanations.

10.5 Final Thoughts

The development of LightBox represents a significant step toward achieving ethical, transparent, and accessible AI. By combining cutting-edge XAI techniques with human-centered design principles and NLP enhancements, LightBox offers a holistic framework for improving model interpretability. Future research and development will further strengthen its adaptability, scalability, and impact across diverse domains, driving progress toward responsible and trustworthy AI systems.

This project lays the foundation for future advancements in explainable AI, underscoring the importance of collaboration between research, development, and user-centered innovation.

XI. Ethical Considerations

11.1 Ethical Considerations

LightBox, designed to enhance the explainability of AI models, introduces several specific ethical considerations that require careful attention. One of the primary ethical challenges revolves around the accuracy and reliability of the explanations it provides. Since LightBox uses methodologies like SHAP and LIME to interpret complex machine learning models, there is an inherent risk that these explanations could be oversimplified or misinterpreted by users. For instance, if a financial institution uses LightBox to explain credit scoring algorithms, inaccuracies in explaining how certain factors like zip code or age influence credit decisions could perpetuate or obscure underlying biases, misleading decision-makers and potentially leading to unfair treatment of individuals.

Another ethical concern is related to the transparency and user understanding of the explanations generated by LightBox. While the tool aims to demystify AI decisions, the technical nature of model explanations—like those involving probabilistic interpretations or high-dimensional data visualizations—might still be challenging for non-expert users to fully grasp. This could lead to a false sense of security where users believe they understand the model's decisions when they actually do not, potentially resulting in poor or unethical decision-making. A specific scenario might involve a medical professional interpreting predictive models for patient treatment. If the professional misinterprets the model’s explanations due to their complexity, it could lead to inappropriate treatment plans, affecting patient health outcomes.

Furthermore, the deployment of LightBox itself raises concerns about privacy and data security. As the tool processes sensitive input data to generate explanations, there is a critical need to ensure that data is handled securely and in compliance with data protection regulations such as GDPR. The ethical obligation here is to protect user data from breaches and unauthorized access, which could have severe consequences, especially in fields like healthcare or finance.

To address these ethical considerations, the development team behind LightBox will focus on creating comprehensive documentation that clearly outlines the potential limitations and appropriate contexts for using the tool’s explanations. This documentation will serve as an essential resource for users, guiding them on how to interpret and apply the explanations responsibly. By providing detailed information about the scenarios where LightBox performs best, as well as cautionary advice on situations where its explanations might be less reliable, users will be better equipped to understand the nuances and boundaries of the tool's capabilities. This approach prioritizes user education over technical solutions to foster informed and cautious application of the explanations generated by LightBox.

Lastly, ethical deployment of LightBox must include robust data security measures to protect sensitive information. This involves encrypting data transmissions, securing databases, and regularly auditing security practices to prevent breaches. By proactively addressing these ethical challenges, LightBox can fulfill its promise of making AI more understandable and just, while minimizing potential harm and fostering trust among its users.

XII. Limitations of the Study and Software

Despite its advancements, LightBox has limitations, including the inherent complexity of some AI models that may resist full transparency. The tool's effectiveness can also vary across different domains and types of data. Acknowledging these limitations is crucial for ongoing improvement.

XIII. References

[1] [1] Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., Zhu, J. (2019). Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges. In: Tang, J., Kan, MY., Zhao, D., Li, S., Zan, H. (eds) Natural Language Processing and Chinese Computing. NLPCC 2019. Lecture Notes in Computer Science(), vol 11839. Springer, Cham. https://doi.org/10.1007/978-3-030-32236-6_51

[2] Saranya A., Subhashini R., A systematic review of Explainable Artificial Intelligence models and applications: Recent developments and future trends, Decision Analytics Journal, Volume 7, 2023, 100230, ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2023.100230.

[3] Salih, A., Raisi-Estabragh, Z., Boscolo Galazzo, I., Radeva, P., Petersen, S. E., Menegaz, G., & Lekadir, K. (2023). Commentary on explainable artificial intelligence methods: SHAP and LIME. arXiv preprint arXiv:2305.02012.

[4] Vimbi, V., Shaffi, N., & Mahmud, M. (2024, April 5). *Interpreting Artificial Intelligence Models: A systematic review on the application of lime and shap in alzheimer's disease detection - brain informatics*. SpringerOpen. https://braininformatics.springeropen.com/articles/10.1186/s40708-024-00222-1.

[5] Siachos, I., & Karacapilidis, N. (2024). Explainable artificial intelligence methods to enhance transparency and trust in digital deliberation settings. Future Internet, 16(7), 241. https://doi.org/10.3390/fi16070241.

[6] Salih, A., Raisl-Estabragh, Z., Galazzo, I. B., & Radeva, P. (2024, February 29). *Commentary on explainable artificial intelligence methods: Shap and lime*. Commentary on explainable artificial intelligence methods: SHAP and LIME. https://ar5iv.labs.arxiv.org/html/2305.02012v3.

[7] Saeed Center for Artificial Intelligence ResearchUniversity of AgderGrimstad, W. (2024, March 1). Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. https://ar5iv.labs.arxiv.org/html/2111.06420.

[8] Minh, D., Wang, H. X., Li, Y. F., & others. (2022). Explainable artificial intelligence: A comprehensive review. *Artificial Intelligence Review, 55*(5), 3503–3568. https://doi.org/10.1007/s10462-021-10088-y.

[9] Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics, 8*(8), 832. https://doi.org/10.3390/electronics8080832.

[10] Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access, 6*, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052.

[11] Sharma, A. (2020, September 27). *Decrypting your machine learning model using lime*. Medium. https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5

[12] *Titanic - Machine Learning from Disaster*. Kaggle. (n.d.). https://www.kaggle.com/competitions/titanic/data?select=train.csv

[13] Trevisan, V. (2022, July 5). *Using shap values to explain how your Machine Learning Model Works*. Medium. https://towardsdatascience.com/using-shap-values-to-explain-how-your-machine-learning-model-works-732b3f40e137

[14] SHAP. (n.d.). *Beeswarm plot*. beeswarm plot - SHAP latest documentation. https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/beeswarm.html

[15] OpenAI. (n.d.). *OpenAI platform*. Docs/Models. https://platform.openai.com/docs/models

XIV. Appendices

14.1 Code Listings

[1] LightBox XAI Experiments Deepnote App, https://deepnote.com/app/dtw/LightBox-XAI-fc319e3e-4ffc-42c7-8caf-c1c8d04b6dfa?utm_source=app-settings&utm_medium=product-shared-content&utm_campaign=data-app&utm_content=fc319e3e-4ffc-42c7-8caf-c1c8d04b6dfa