Black Box to Glass Box: Hands- On Explainable AI

(2024-12)

Dr. Abhishek Singh

AI Engineering

Overview

Topics We'll Be Covering

Hour 1: Foundations of Explainable AI

- Machine Learning Basics
- K- Nearest Neighbor
- Deep Learning Basics

Hour 2: Introduction to Explainable AI

- Explainable AI: Unveiling the Black Box
- What is Explainable AI?
- Brief History of XAI
- XAI Applications
- Challenges in XAI

Hour 3: Model-agnostic Explainability Techniques

- Tree Explainer
- LIME (Local Interpretable Model-agnostic Explanations)
- SHAP (SHapley Additive exPlanations)

Hour 4: Model-specific Explainability Techniques

- DeepLIFT (Deep Learning Important FeaTures)
- CAM
- Grad-CAM
- Case 1: Image Classification
- Case 2: Natural Language Processing

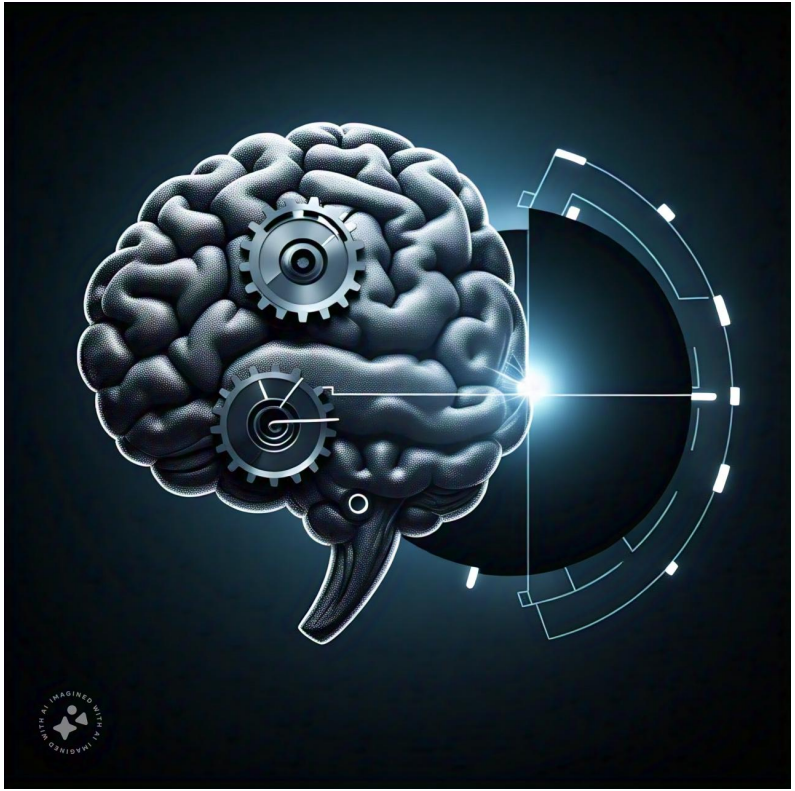
Hour 5: XAI in Practice - Challenges and Opportunities

- Challenges in Deploying XAI
- Opportunities for XAI
- Case 3: Healthcare
- Case 4: Finance
- Future Directions for XAI

Hour 6: Conclusion and Next Steps

- Summary: Key takeaways
- Final Thoughts
- Additional Resources
- Conclusion

Foundation: Machine Learning



What is Machine Learning?

- Machine Learning (ML) is a field of artificial intelligence that empowers computers to learn from data without explicit programming. Instead of being explicitly programmed with rules, ML algorithms learn patterns from data and make predictions or decisions based on those patterns.

Core of Machine Learning?

- Data - Any unprocessed fact, value, text, sound or picture.
- Model - Learning general trend from data.

Features of Machine Learning?

- ML techniques are able to find and highlight various patterns in the data very quickly in comparison to human beings, who may miss these patterns due to the size of the data.
- ML techniques make informed inferences on a wide range of problems, from helping diagnose diseases to coming up with solutions for critical business deals.

Foundation: Classification of Machine Learning

- **Supervised Learning:** A supervised learning model is a type of machine learning model that is trained on a labeled dataset. This means that the model is provided with both input data and the corresponding correct output, or label. The model's goal is to learn the mapping between the input data and the output labels, so that it can make accurate predictions on new, unseen data.
- **Unsupervised Learning:** Unsupervised learning is a type of machine learning algorithm that allows a model to learn patterns and relationships within data without the need for labeled training data. In this approach, the model is exposed to raw data and left to discover underlying structures and generate insights on its own. This makes it a powerful tool for exploring large, complex datasets and uncovering hidden patterns that might not be apparent to human analysts.
- **Reinforcement Learning:** Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent learns to take actions that maximize a reward signal, similar to how a child learns through trial and error.

Foundation: Supervised Machine Learning

Key characteristics:
- Labeled data: The training data is labeled, meaning that each data point is associated with a correct output.
- Prediction: The model learns to predict the output for new, unseen input data.
- Accuracy: The model's performance is evaluated based on its accuracy in making correct predictions.

Common types of models:
- Regression: Models that predict a continuous numerical value.
- Classification: Models that predict a categorical label.

Example:
- Let's say we have a dataset of house prices, including features like square footage, number of bedrooms, and location. We want to create a model that can predict the price of a new house based on its features.
  In this case, we would use a regression model. We would train the model on the labeled dataset of house prices, and then use the trained model to predict the price of a new house.

Advantages:
- Can achieve high accuracy on well-defined problems.
- Can be used for both regression and classification tasks.

Disadvantages:
- Requires a large amount of labeled data.
- Can be sensitive to the quality of the training data.

Foundation: Unsupervised Machine Learning

Key characteristics:

- Unlabeled Data: Unsupervised learning algorithms work with data that has not been pre-classified or categorized. The model must identify patterns and structures within the data itself.
- Self-Organization: The model learns to organize the data into meaningful groups or clusters based on similarities and differences.
- Feature Learning: Unsupervised learning algorithms can learn to extract relevant features from the data, reducing the need for manual feature engineering.

Common types of models:

- Clustering: Clustering algorithms group similar data points together, creating clusters. Common clustering algorithms include K-means, hierarchical clustering, and DBSCAN.
- Dimensionality Reduction: Dimensionality reduction techniques reduce the number of features in a dataset while preserving the most important information. Principal Component Analysis (PCA) and t-SNE are popular dimensionality reduction algorithms.

Example:

- Imagine a dataset containing customer purchase history. An unsupervised learning algorithm could analyze this data and identify groups of customers with similar buying habits. This information could be used to create targeted marketing campaigns or personalize product recommendations.

Advantages:

- Discovery of Hidden Patterns: Unsupervised learning can uncover patterns and relationships that might not be apparent through human analysis.
- Reduced Reliance on Labeled Data: Unsupervised learning can be applied to large datasets that are difficult or expensive to label.
- Feature Learning: Unsupervised learning algorithms can learn to extract relevant features from the data, reducing the need for manual feature engineering.

Disadvantages:

- Lack of Ground Truth: Without labeled data, it can be difficult to evaluate the accuracy of unsupervised learning models.
- Sensitivity to Initialization: Some unsupervised learning algorithms, such as K-means, are sensitive to the initial choice of cluster centers.
- Computational Cost: Some unsupervised learning algorithms can be computationally expensive, especially for large datasets.

Foundation: Reinforcement Machine Learning

Key characteristics:

- Agent-Environment Interaction: The agent interacts with an environment, taking actions and receiving feedback in the form of rewards or penalties.
- Trial and Error Learning: The agent learns through experience, trying different actions and adjusting its behavior based on the outcomes.
- Reward Maximization: The goal of the agent is to maximize the cumulative reward over time.

Common types of models:

- Q-Learning: A popular algorithm that learns the optimal action to take in a given state by estimating the expected future reward.
- Deep Q-Networks (DQN): A combination of Q-learning and deep neural networks that allows for learning complex tasks.
- Policy Gradient Methods: These methods directly optimize the policy function, which maps states to actions.
- Actor-Critic Methods: These methods combine the strengths of value-based and policy-based methods, using both a value function and a policy function.

Example:

- A classic example of reinforcement learning is training an agent to play a game like chess or Go. The agent learns by playing numerous games, receiving rewards for winning and penalties for losing. Over time, the agent learns to make better moves and eventually becomes a skilled player.

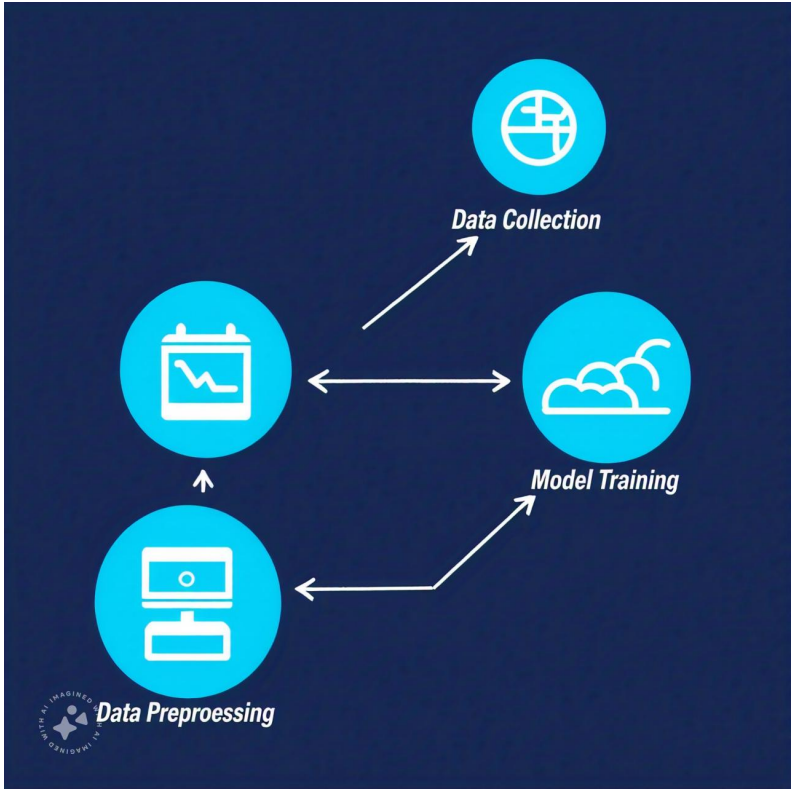
Advantages:

- Adaptability: Reinforcement learning agents can adapt to changing environments and learn new strategies.
- Autonomy: Agents can learn to make decisions independently, without explicit human intervention.
- Potential for Complex Tasks: Reinforcement learning can be applied to complex tasks that are difficult to model explicitly.

Disadvantages:

- Sample Inefficiency: Reinforcement learning can require a large number of trials and errors to learn optimal behavior.
- Exploration-Exploitation Trade-off: The agent must balance exploring new actions with exploiting known good actions.
- Reward Engineering: Designing appropriate reward functions can be challenging and can significantly impact the learning process.

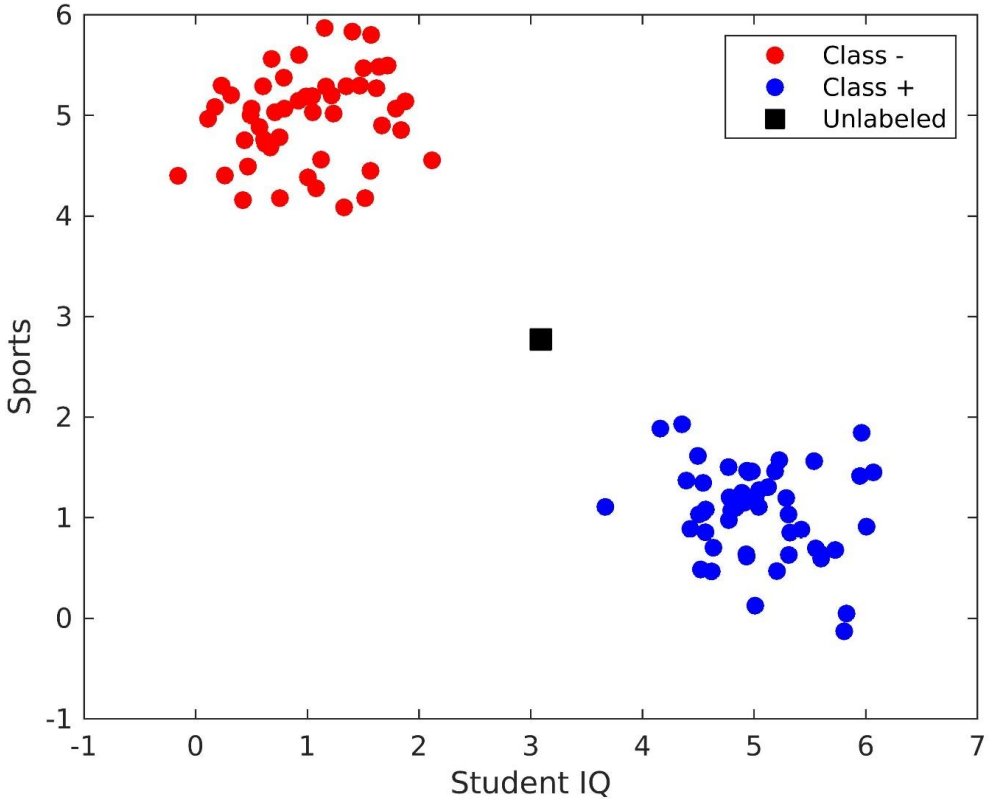
Foundation: Machine Learning Workflow



The machine learning workflow involves:

- Data Collection: Gathering relevant data for training and testing.
- Data Preprocessing: Cleaning, transforming, and preparing the data for training.
- Model Training: Training the algorithm using the prepared data.
- Model Evaluation: Evaluating the performance of the trained model.
- Deployment: Deploying the trained model in a production-ready environment.

Foundation: k- Nearest Neighbors

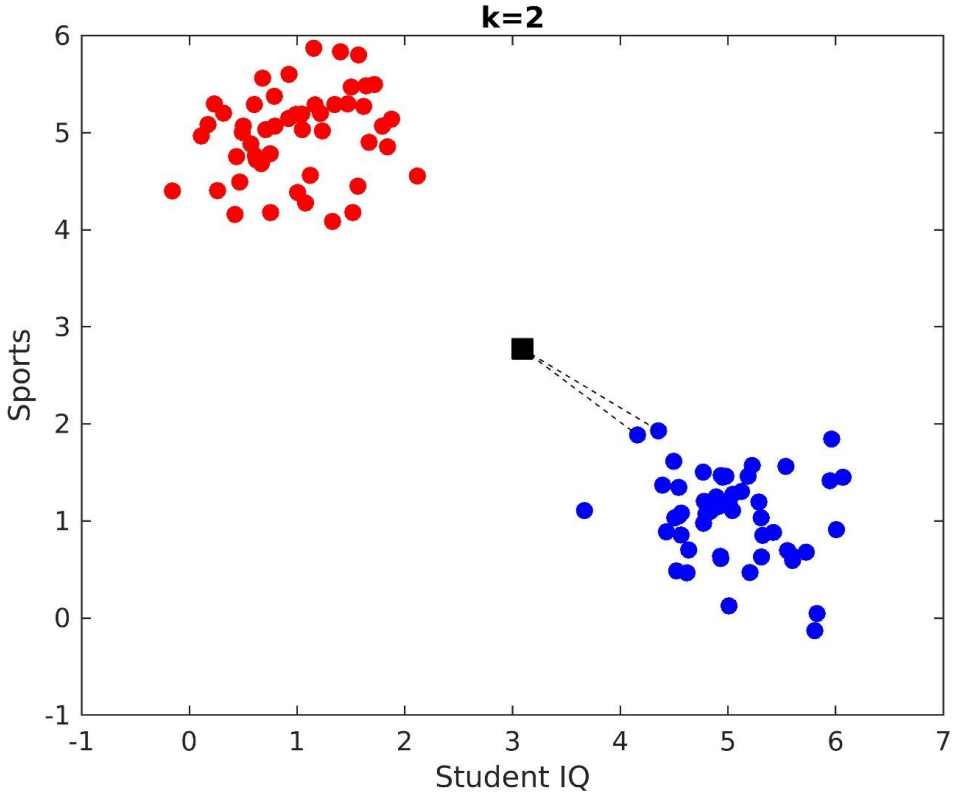


Activity:

- Form groups of close friends among yourselves except one student.
- Ask everybody to rate themself on a scale of 1-10 in studies and sports including the student left in previous step.
- With the help of unknown student ratings, choose the appropriate group he/she should belong to.

Solution: "You tell me about your interests and I will guess the group you are likely to belong"

Foundation: k- Nearest Neighbors



Solution:

- Select the unlabeled data point.
- Measure the distance from unlabeled data point to all other data points.
- Sort the distance and take top “k” labeled data points.
- Count the number of members from each group across top “k” points.
- Assign the label whose members have higher count.
- In case of tie, choose randomly.

Foundation: k- Nearest Neighbors

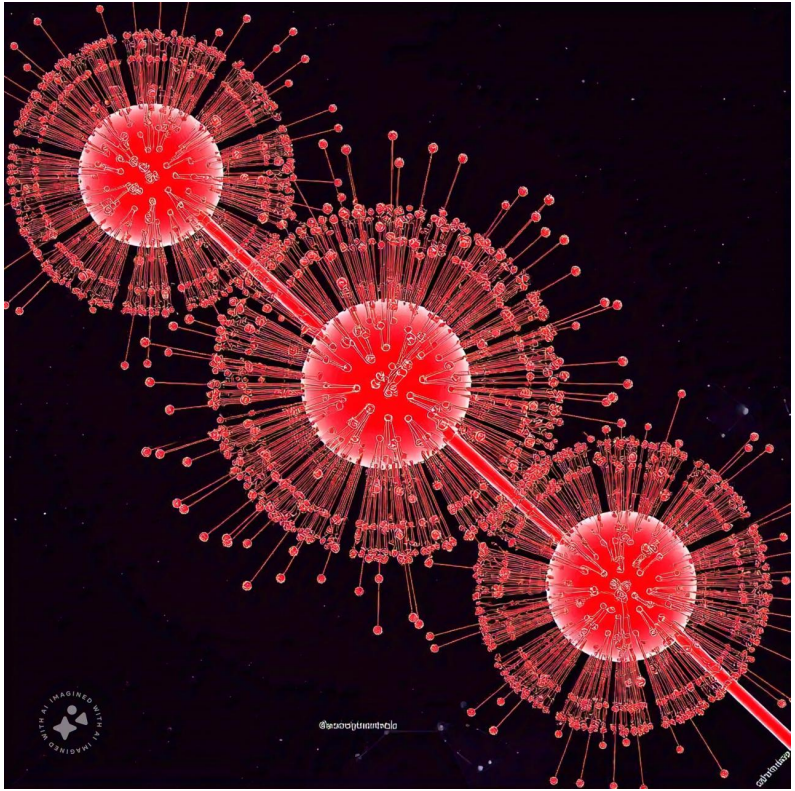
Advantages:

- Natural method.
- Simplest among all.
- Requires no explicit training or model.
- Can be extended to other models (graph based methods).

Limitations:

- Computationally intensive.
- Difficult to decide value of “k”.
- Susceptible to noise and curse of dimensionality.

Foundation: Deep Learning Basics



Deep Learning:

- It is a subset of Machine Learning, involves training artificial neural networks with multiple layers to learn complex patterns in data. The very complexity that makes deep learning so powerful.
- At its core, deep learning mimics the structure and function of the human brain. Neural networks, the building blocks of deep learning, consist of interconnected nodes called neurons. These neurons process information in layers, with each layer learning increasingly complex features from the input data.

Artificial Neural Networks (ANNs):

- ANNs are composed of layers of interconnected nodes (neurons) that process and transform inputs into meaningful outputs. Each node applies a non-linear transformation to the input data, allowing the network to learn complex patterns and relationships.

Foundation: Deep Learning Basics

Key Concepts in Deep Learning

- **Artificial Neural Networks:** Inspired by biological neurons, artificial neurons process information and transmit signals to other neurons.
- **Activation Functions:** These functions introduce non-linearity into the network, enabling it to learn complex patterns.
    - e.g.: linear, sigmoid, tanh, relu, leaky relu, softmax, etc.
- **Backpropagation:** An algorithm used to adjust the weights of the network to minimize the error between the predicted output and the actual output.
- **Deep Neural Networks:** Networks with multiple hidden layers, allowing them to learn intricate representations of data.
    - e.g.: Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Autoencoders, etc.

Foundation: Deep Learning Basics

Applications of Deep Learning

- **Computer Vision:** Image Classification, Object Detection, Image Segmentation, Image Generation.
- **Natural Language Processing (NLP):** Language Translation, Sentiment Analysis, Text Summarization, Chatbots.
- **Speech Recognition:** Voice Assistants, Speech-to-Text Systems, Speech Recognition Systems.
- **Healthcare:** Medical Imaging Analysis, Predictive Medicine, Drug Discover.
- **Finance:** Risk Analysis, Predictive Modeling, Fraud Detection.
- **Autonomous Vehicles:** Self-Driving Cars, Object Detection, Motion Forecasting.
- **Robotics:** Robot Perception, Robot Control, Human-Robot Interaction.

Foundation: Deep Learning Basics

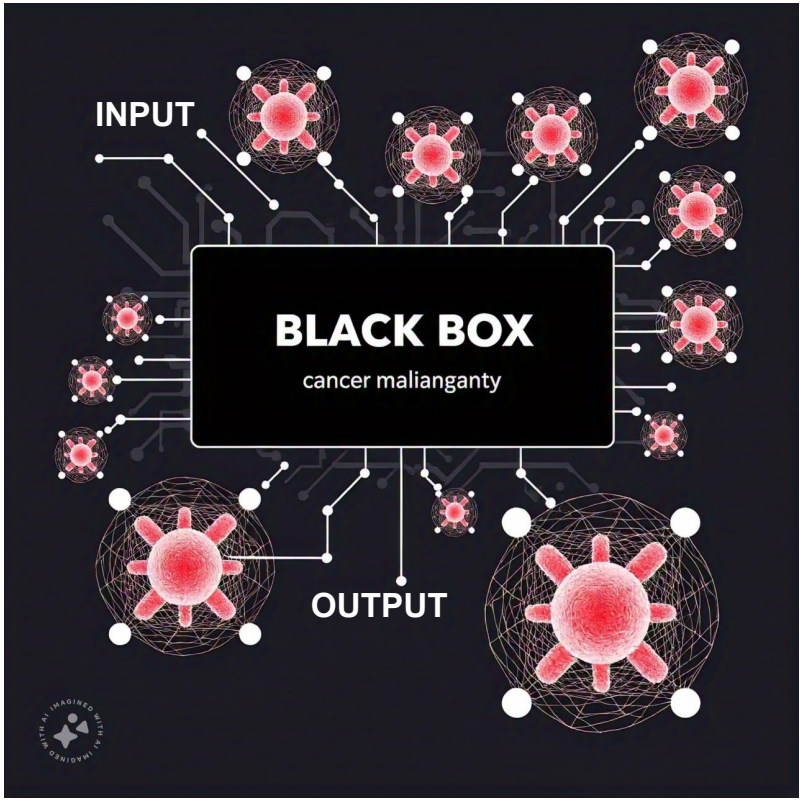
Why Deep Learning Models are not Interpretable?

- **Complexity of Deep Neural Networks:** Deep neural networks have millions of parameters, making it challenging to understand how they arrive at their predictions. The complexity of these models leads to a lack of transparency, making it difficult to interpret their results.
- **Black-Box Nature:** The multiple layers of the network and the non-linear transformations involved create a complex system that is challenging to dissect.
- **Lack of Transparency:** Deep learning models are often treated as "black boxes," providing little insight into their decision-making processes. This lack of transparency makes it challenging to understand why a particular prediction was made.
- **Non-Linearity and Interactions:** Deep learning models involve non-linear transformations and complex interactions between features, making it difficult to interpret their results. These non-linear interactions lead to a lack of explainability, making it challenging to understand how the model arrived at its predictions.
- **Feature Importance and Partial Dependence Plots:** While techniques like feature importance and partial dependence plots can provide some insights, they are limited and do not fully capture the complexity of deep learning models. These techniques can provide a general understanding of which features are important, but they do not provide a detailed understanding of how the model is using those features.
- **Data-Driven Decisions:** The model's decisions are based on statistical patterns in the data, which may not always align with human intuition.

Foundation: Deep Learning Example

- **Task:** Imagine a computer tasked with identifying objects in images. This is a complex task that humans do effortlessly, but for a machine, it requires a sophisticated approach.
- **Solution:** Convolutional Neural Networks (CNNs)
- Simplified breakdown of how a CNN works for image recognition:
    - Convolutional Layers: These layers apply filters to the input image, extracting features like edges, corners, and textures.
    - Pooling Layers: These layers reduce the spatial dimensions of the feature maps, making the network more efficient and reducing overfitting.
    - Fully Connected Layers: These layers classify the extracted features into different categories, such as "cat," "dog," or "car.”

Introduction: The Black Box Problem



- Traditional AI models, particularly deep learning models, are often referred to as **"black boxes"**. They can make highly accurate predictions, but their decision-making processes are opaque. This lack of transparency poses significant challenges:
    - **Trust and Accountability:** How can we trust a system if we don't understand how it arrives at its conclusions?
    - **Ethical Considerations:** If an AI system makes a biased or harmful decision, how can we identify and rectify the issue?
    - **Regulatory Compliance:** Many industries are subject to regulations that require transparency in decision-making processes.

Introduction: What is Explainable AI?

Explainable AI or XAI is a field of study that aims to make AI models more interpretable. It involves developing techniques and methods to understand how AI systems work, why they make certain decisions, and the factors that influence their outputs.

Introduction: Explainable AI (XAI)

“Interpretability is the ability to provide explanations in understandable terms to a human”

- Performance of supervised learning models generally depends upon their predictive accuracy.
- In case of sensitive application domains accuracy is not enough.
- Einstein said,
  ”If you can’t explain it simply, you don’t understand it well enough.”
- It is not sufficient to understand ”how” does a model work, rather ”why” the model is true becomes an important issue to validate the model.
- For humans, explanations can be intuitive but must be unambiguous and transformable to logical forms.
- We define interpretability as
  “The ability to explain or to present in understandable terms to a human with the requisite domain knowledge”.
- The definition emphasizes context specific explanations.

Foundation: Explainable vs Interpretable

- Most of the researchers interchangeably use explainability and interpretability.
- **Interpretability:** the ability to explain or to present in understandable terms to a human.
    - interpretability is mostly connected with the intuition behind the outputs of a model with the idea being that the more interpretable a machine learning system is, the easier it is to identify cause-and-effect relationships within the system's inputs and outputs.
    - For example, in image recognition tasks, part of the reason that led a system to decide that a specific object is part of an image (output) could be certain dominant patterns in the image (input).
- **Explainability:** associated with the internal logic and mechanics that are inside a machine learning system.
    - Explainability, on the other hand, is associated with the internal logic and mechanics that are inside a machine learning system. The more explainable a model, the deeper the understanding that humans achieve in terms of the internal procedures that take place while the model is training or making decisions.

"An interpretable model does not necessarily translate to one that humans are able to understand the internal logic of or its underlying processes."

Introduction: Significance of Interpretability

- To transfer the learning into a broader knowledge base.
- To make informed decisions about how to improve the model.
- Guarding against embedded bias or debugging an algorithm.
- Concepts similar to Interpretability:
    - **Explainability:** If a model's inner working can be conveyed to the stakeholders having less technical skills than the experts, then the model is explainable.
    - **Trust:** If a model is robust, reliable and can be generalized to handle various real-life scenarios, it can be considered as trustworthy.
    - **Fairness:** A fair model should be un-biased and should not discriminate among samples.
    - **Accountability:** If a model can identify and evaluate the conduct of an entity responsibile for a decision, the model is called accountable.

Introduction: Recognized Interpretable Models

- **Decision Tree:** DT are an interpretable and understandable models that are suitable for both global and local explanations.
- **Decision Rules:** Decision rules can explain a model outcome and can be used for designing of transparent techniques.
- **Features Importance:** In Feature Importance approaches the explanations are expressed in terms of features weights used by the black box model.
- **Activation Maximization:** By opening up and inspecting the deep learning model, the features, neurons that get maximally activated for some specific inputs can be attributed for determining the output.

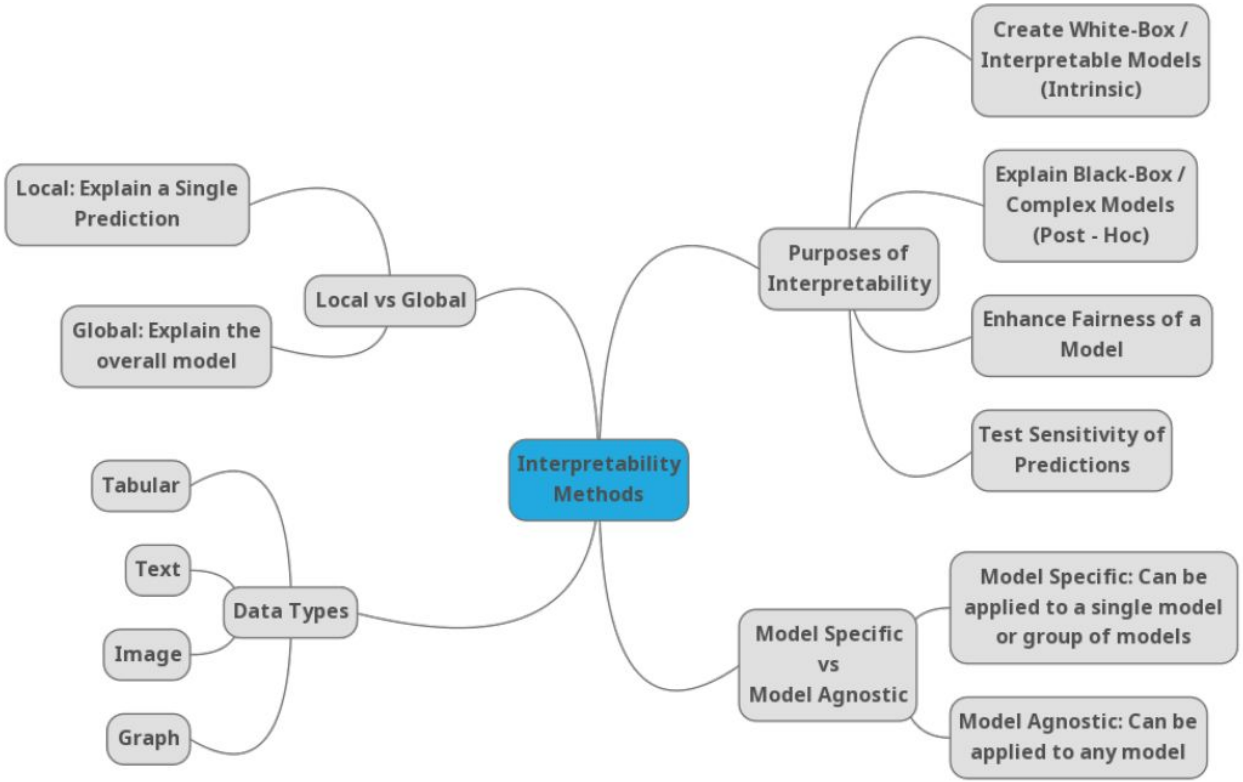
Introduction: Brief History of XAI

- Artificial Intelligence (AI) has rapidly evolved, transforming industries and our daily lives. However, with this advancement, a significant challenge has emerged: the "black box" problem. Traditional AI models, especially deep neural networks, are often complex and opaque, making it difficult to understand how they arrive at their decisions. This lack of transparency has raised concerns about trust, accountability, and ethical implications.
- XAI aims to make AI systems more transparent, understandable, and accountable.
- **Early Roots:** The early roots of XAI can be traced back to the development of knowledge-based systems and expert systems in the 1970s and 1980s. These systems were designed to mimic human reasoning and decision-making processes, making them inherently explainable.
- **The Rise of Machine Learning:** With the rise of machine learning in the 1990s and 2000s, particularly with the advent of deep learning, the black box problem became more pronounced. These models, while powerful, often lacked interpretability.

Introduction: Benefits of XAI

- **Improved Trust and Transparency:** By understanding the reasoning behind AI decisions, users can build trust in the system.
- **Enhanced Model Performance:** XAI can help identify biases, errors, and areas for improvement in models.
- **Ethical AI:** XAI can help ensure that AI systems are fair, unbiased, and aligned with human values.
- **Regulatory Compliance:** XAI can help organizations meet regulatory requirements for transparency and accountability.

Introduction: Classification of ML interpretability methods



Introduction: Applications of XAI

Healthcare

- Medical Diagnosis: XAI can help doctors understand the reasoning behind AI-powered medical diagnoses, leading to more informed decisions and potentially saving lives.
- Drug Discovery: XAI can be used to analyze vast amounts of biological data, accelerating the discovery of new drugs.
- Personalized Medicine: XAI can help tailor treatments to individual patients based on their unique genetic makeup and medical history.

Finance

- Credit Scoring: XAI can help explain the factors that influence credit decisions, making the process more transparent and fair.
- Fraud Detection: XAI can help identify patterns of fraudulent behavior, leading to more effective fraud prevention strategies.
- Algorithmic Trading: XAI can help explain the reasoning behind complex trading algorithms, reducing risks and increasing confidence in investment decisions.

Introduction: Applications of XAI

Autonomous Vehicles

- Decision-Making: XAI can help explain the decisions made by self-driving cars, increasing trust and safety.
- Accident Investigation: XAI can be used to analyze accidents involving autonomous vehicles, helping to identify the root causes and prevent future accidents.

Criminal Justice

- Risk Assessment: XAI can help explain the factors that influence risk assessment tools used in the criminal justice system, reducing the potential for bias.
- Recidivism Prediction: XAI can help explain the factors that influence recidivism prediction models, leading to more accurate and fair sentencing decisions.

Introduction: Applications of XAI

Environmental Science

- Climate Modeling: XAI can help explain the complex models used to predict climate change, improving our understanding of the risks and potential solutions.
- Natural Disaster Prediction: XAI can help explain the factors that influence natural disaster prediction models, leading to more accurate and timely warnings.

Other Applications

- Education: XAI can help explain the reasoning behind AI-powered tutoring systems, making the learning process more effective.
- Human Resources: XAI can help explain the factors that influence hiring decisions, reducing the potential for bias.
- Marketing: XAI can help explain the factors that influence consumer behavior, leading to more effective marketing campaigns.

Introduction: Challenges in XAI

Technical Challenges

- Model Complexity: As AI models become increasingly complex, especially deep neural networks, it becomes more difficult to extract meaningful explanations.
- Interpretability-Accuracy Trade-off: Often, making a model more interpretable can lead to a decrease in its accuracy. This is a delicate balance that XAI researchers strive to optimize.
- Scalability: Applying XAI techniques to large-scale models can be computationally expensive, making it challenging to deploy in real-world applications.

Human Factors

- User Understanding: Even with XAI techniques, it can be difficult for non-experts to understand complex explanations.
- Trust and Bias: Users may still be skeptical of AI systems, even with explanations. Additionally, biases in the data or algorithms can be reflected in the explanations, leading to misleading interpretations.

Introduction: Challenges in XAI

Ethical Considerations

- Fairness and Bias: XAI can help identify and mitigate biases in AI systems, but it's essential to ensure that explanations themselves are fair and unbiased.
- Privacy: Explaining how a model works may reveal sensitive information about the data used to train it, raising privacy concerns.

Model agnostic explainability techniques: EDA

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	\
0	14.23	1.71	2.43	15.6	127	2.80	
1	13.20	1.78	2.14	11.2	100	2.65	
2	13.16	2.36	2.67	18.6	101	2.80	
3	14.37	1.95	2.50	16.8	113	3.85	
4	13.24	2.59	2.87	21.0	118	2.80	

	Flavanoids	Nonflavanoid_phenols	Proanthocyanins	Color_intensity	Hue	\
0	3.06	0.28	2.29	5.64	1.04	
1	2.76	0.26	1.28	4.38	1.05	
2	3.24	0.30	2.81	5.68	1.03	
3	3.49	0.24	2.18	7.80	0.86	
4	2.69	0.39	1.82	4.32	1.04	

	0D280_0D315_of_diluted_wines	Proline
0	3.92	1065
1	3.40	1050
2	3.17	1185
3	3.45	1480
4	2.93	735

Model agnostic explainability techniques: EDA

Role of EDA in Model Interpretability

- EDA helps identify relationships, trends, and anomalies in data.
- Key tasks include:
- Understanding feature distributions.
- Detecting outliers and anomalies.
- Analyzing feature importance and relationships.

Key Techniques in EDA for Interpretability

- Descriptive Statistics
  - Summary Statistics: Mean, median, standard deviation, etc.
  - Helps understand feature scales and variability.
- Correlations:
  - Detect linear relationships between features.
  - Use heatmaps to visualize correlations between variables.

Model agnostic explainability techniques: EDA

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium
count	178.000000	178.000000	178.000000	178.000000	178.000000
mean	13.000618	2.336348	2.366517	19.494944	99.741573
std	0.811827	1.117146	0.274344	3.339564	14.282484
min	11.030000	0.740000	1.360000	10.600000	70.000000
25%	12.362500	1.602500	2.210000	17.200000	88.000000
50%	13.050000	1.865000	2.360000	19.500000	98.000000
75%	13.677500	3.082500	2.557500	21.500000	107.000000
max	14.830000	5.800000	3.230000	30.000000	162.000000

	Total_phenols	Flavanoids	Nonflavanoid_phenols	Proanthocyanins
count	178.000000	178.000000	178.000000	178.000000
mean	2.295112	2.029270	0.361854	1.590899
std	0.625851	0.998859	0.124453	0.572359
min	0.980000	0.340000	0.130000	0.410000
25%	1.742500	1.205000	0.270000	1.250000
50%	2.355000	2.135000	0.340000	1.555000
75%	2.800000	2.875000	0.437500	1.950000
max	3.880000	5.080000	0.660000	3.580000

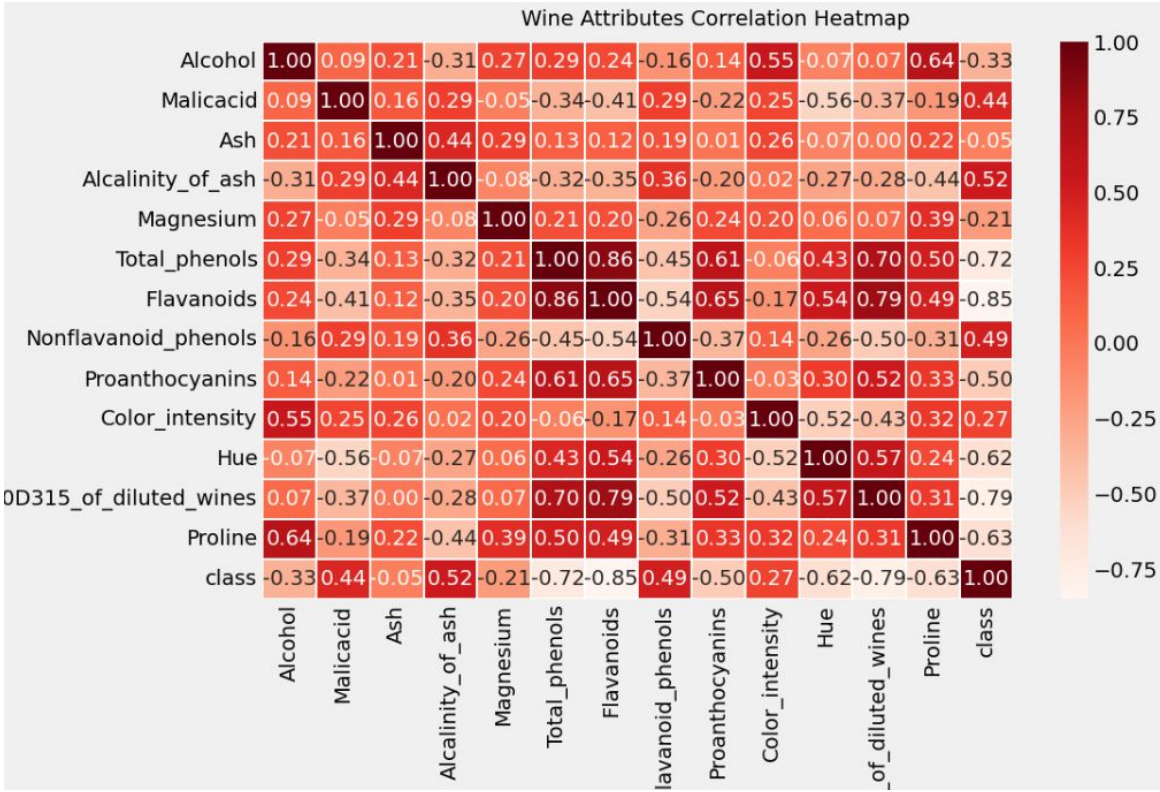
	Color_intensity	Hue	0D280_0D315_of_diluted_wines	Proline
count	178.000000	178.000000	178.000000	178.000000
mean	5.058090	0.957449	2.611685	746.893258
std	2.318286	0.228572	0.709990	314.907474
min	1.280000	0.480000	1.270000	278.000000
25%	3.220000	0.782500	1.937500	500.500000
50%	4.690000	0.965000	2.780000	673.500000
75%	6.200000	1.120000	3.170000	985.000000
max	13.000000	1.710000	4.000000	1680.000000

Model agnostic explainability techniques: EDA

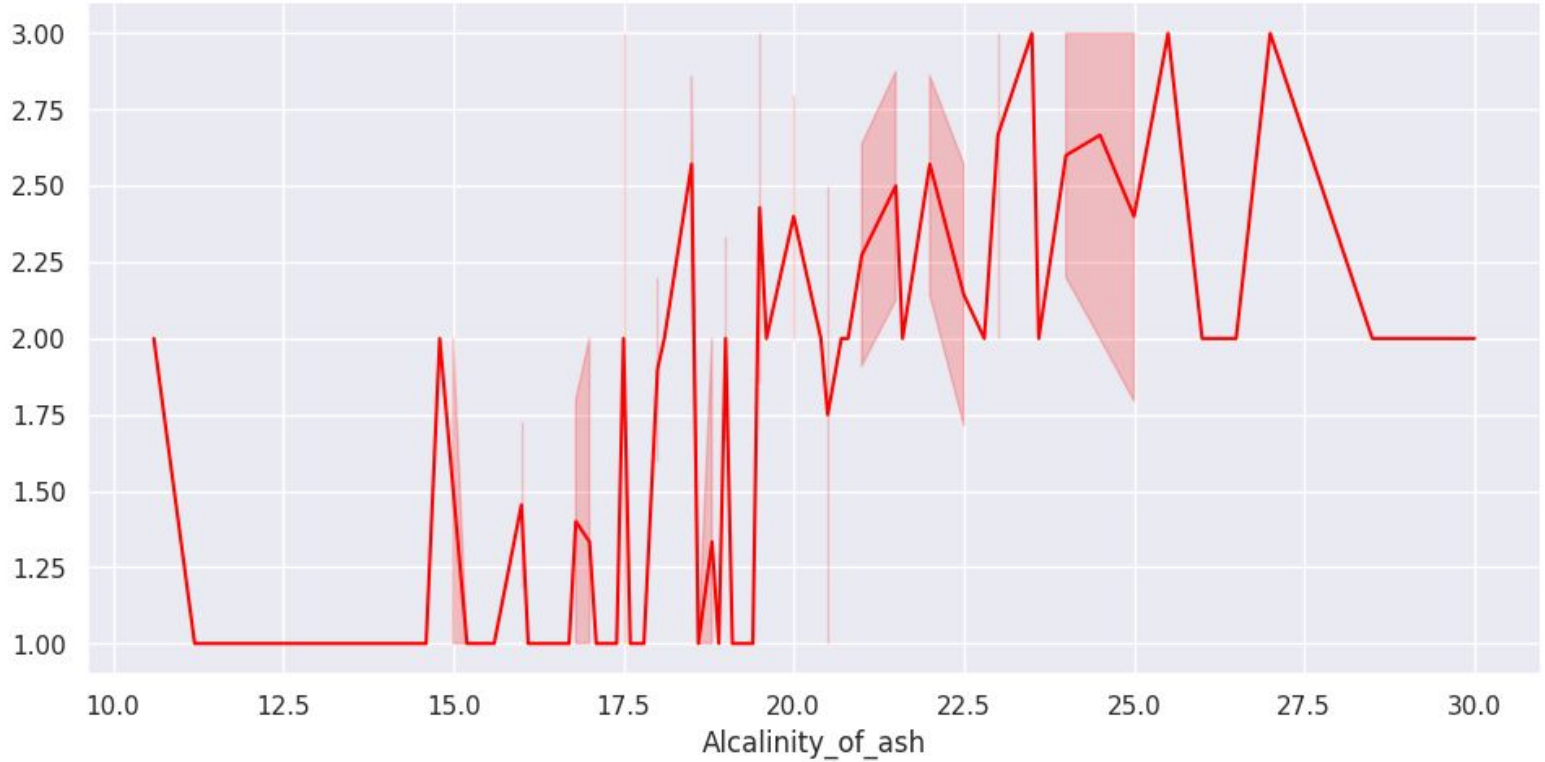
Visualization

- **Univariate Analysis:**
  - Histograms, box plots, and KDE plots to study feature distributions.
  - Example: Age distribution for a loan dataset can indicate the age group most prevalent.
- **Bivariate Analysis:**
  - Scatter plots and pair plots to identify relationships.
  - Example: Plot income vs. spending to assess linearity.
- **Multivariate Analysis:**
  - PCA for dimensionality reduction and visualizing relationships between features.
  - Example: Reduce high-dimensional data for visualization and understanding latent patterns.

Model agnostic explainability techniques: EDA



Model agnostic explainability techniques: EDA



Model agnostic explainability techniques: EDA

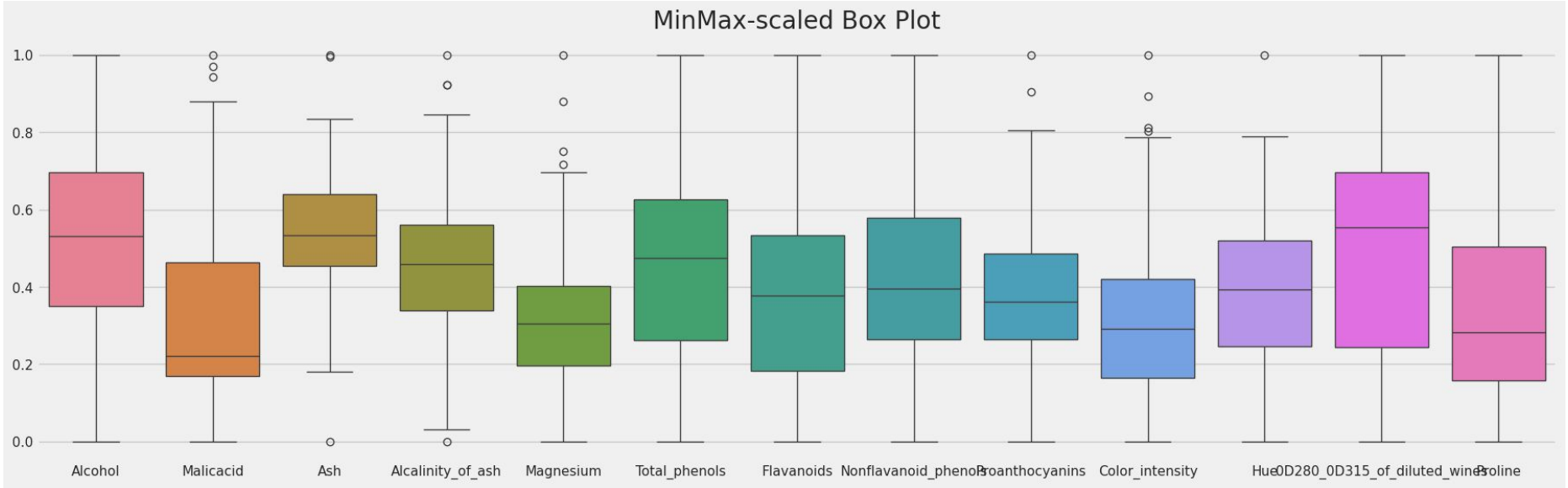
Outlier Detection

- Box plots, IQR method, and z-scores highlight extreme values.
- Removes noise that might distort the model.

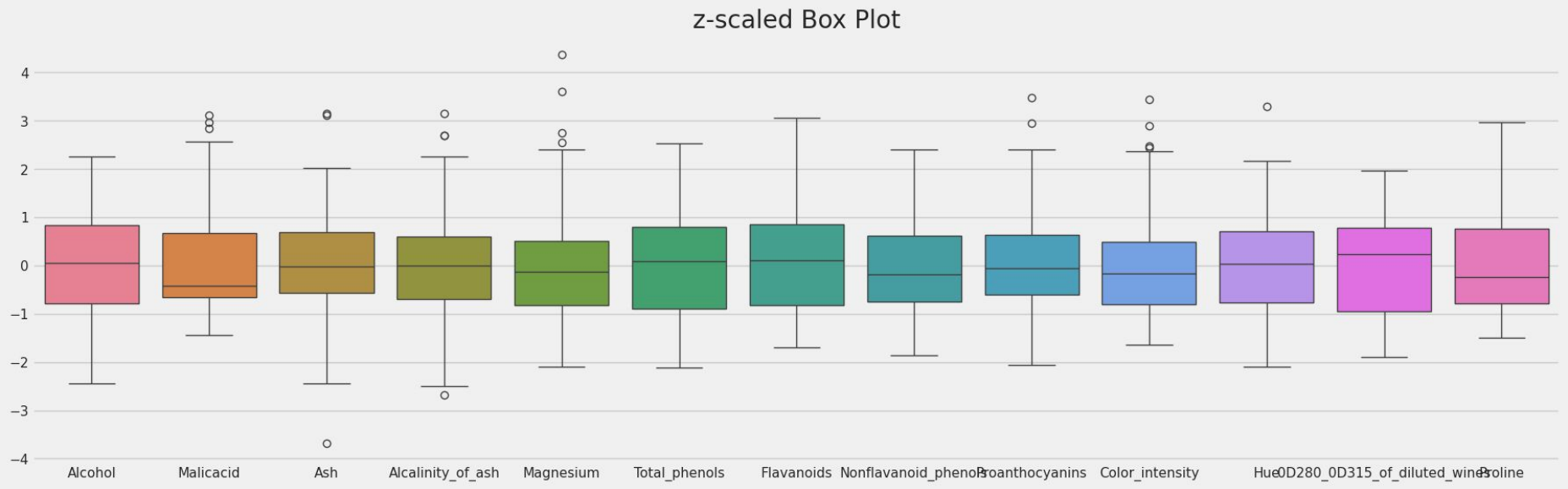
Missing Value Analysis

- Identifies the amount and pattern of missing data.
- Example: Heatmaps show clusters of missing values.

Model agnostic explainability techniques: EDA



Model agnostic explainability techniques: EDA

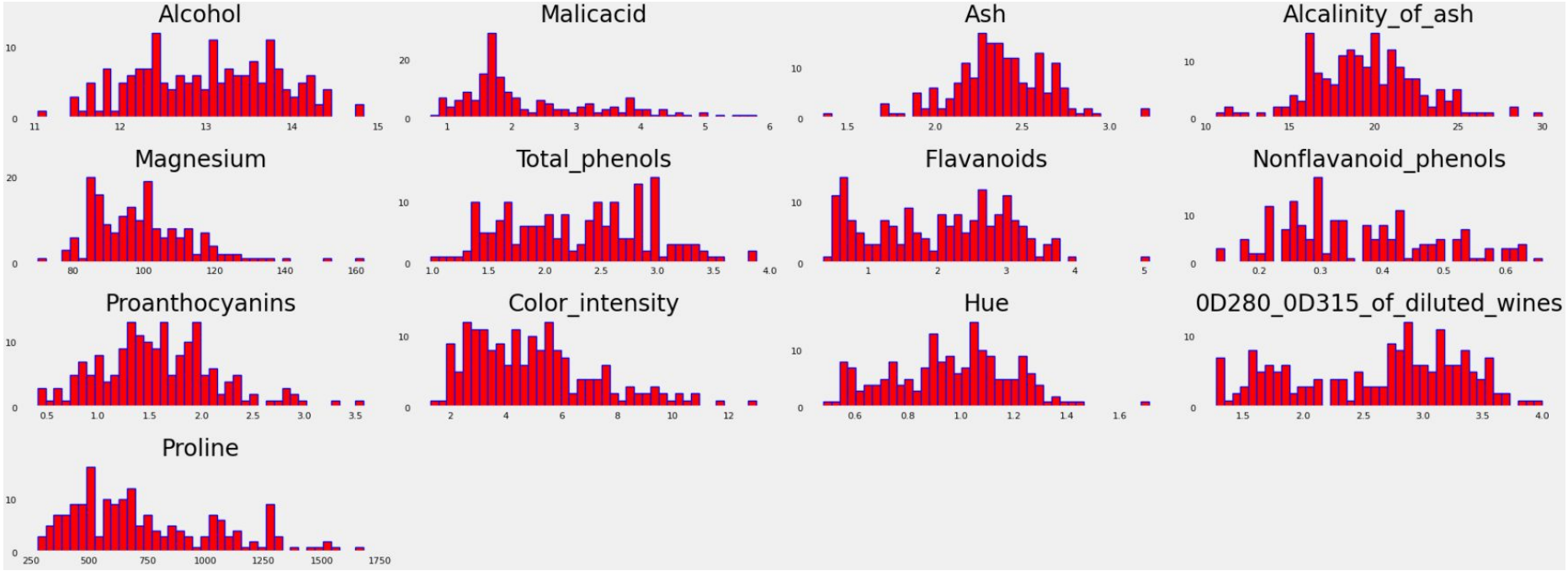


Model agnostic explainability techniques: EDA

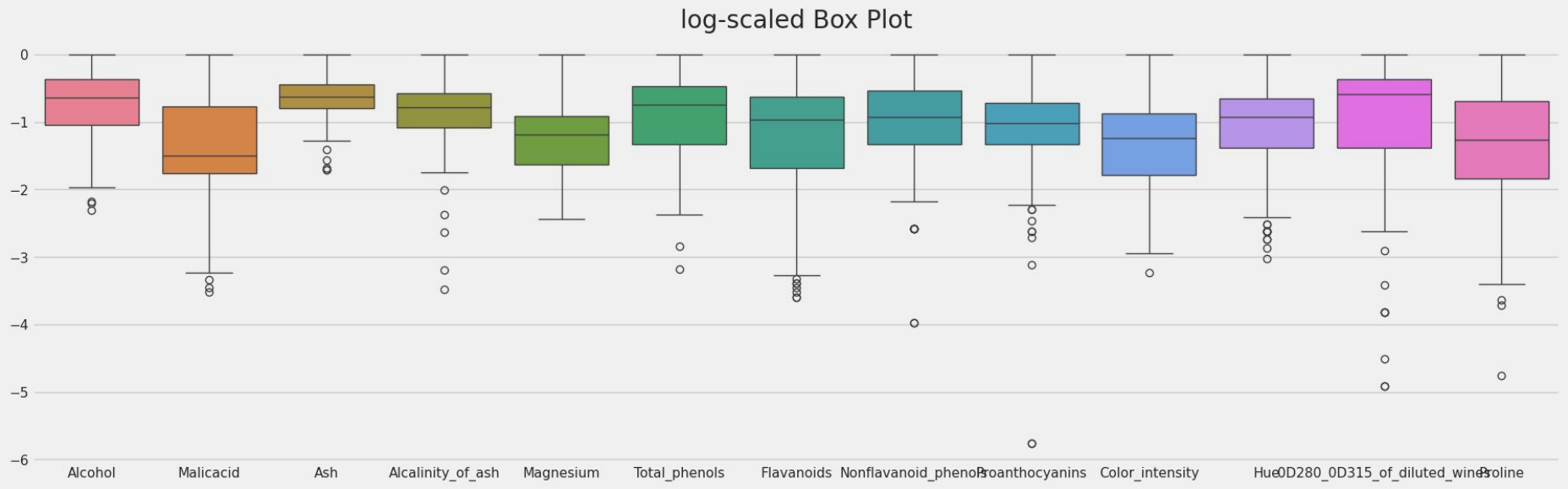
EDA Insights for ML Model Selection

- **Feature Relationships:**
  - Understand which features are relevant and their impact on the target variable.
  - Example: Use scatter plots to assess non-linear relationships; this can suggest tree-based models over linear ones.
- **Class Imbalance:**
  - Use bar plots to identify imbalances in categorical features.
  - Guides strategies like SMOTE or class-weighted models.
- **Data Transformations:**
  - Log transforms or scaling based on distribution observed during EDA.

Model agnostic explainability techniques: EDA



Model agnostic explainability techniques: EDA



Model agnostic explainability techniques: EDA

Hands-on Assignment

- **Dataset: Titanic Survival Prediction**
- Step 1: Load Data
  - Study the survival rates based on gender and class using bar charts.
- Step 2: Univariate Analysis
  - Age: Check the distribution to handle missing values.
- Step 3: Bivariate Analysis
  - Fare vs. Survival: Box plot to understand the influence of ticket price on survival.
- Step 4: Correlation Analysis
  - Check multicollinearity to eliminate redundant features.
- Step 5: Feature Importance
  - Use random forests with SHAP or permutation importance after initial EDA.

Model agnostic explainability techniques: Tree

Basic Tree Models:

- **Decision Trees:**
  - Each decision is a split based on a feature threshold.
  - The path from root to leaf is the explanation for a prediction.
  - Example: Deciding whether to approve a loan based on income and credit score.
  - Visual Explanation:
    - Nodes represent decisions; edges represent outcomes.

Model agnostic explainability techniques: Tree

Advanced Tree Models:

- **Random Forests:**
  - Ensemble of multiple decision trees.
  - Harder to interpret directly but can use feature importance scores to understand overall contributions.
- **Gradient Boosting Machines (GBMs):**
  - Combine weak learners iteratively for better performance.
  - Tools like SHAP (SHapley Additive exPlanations) help explain predictions.

Model agnostic explainability techniques: Tree

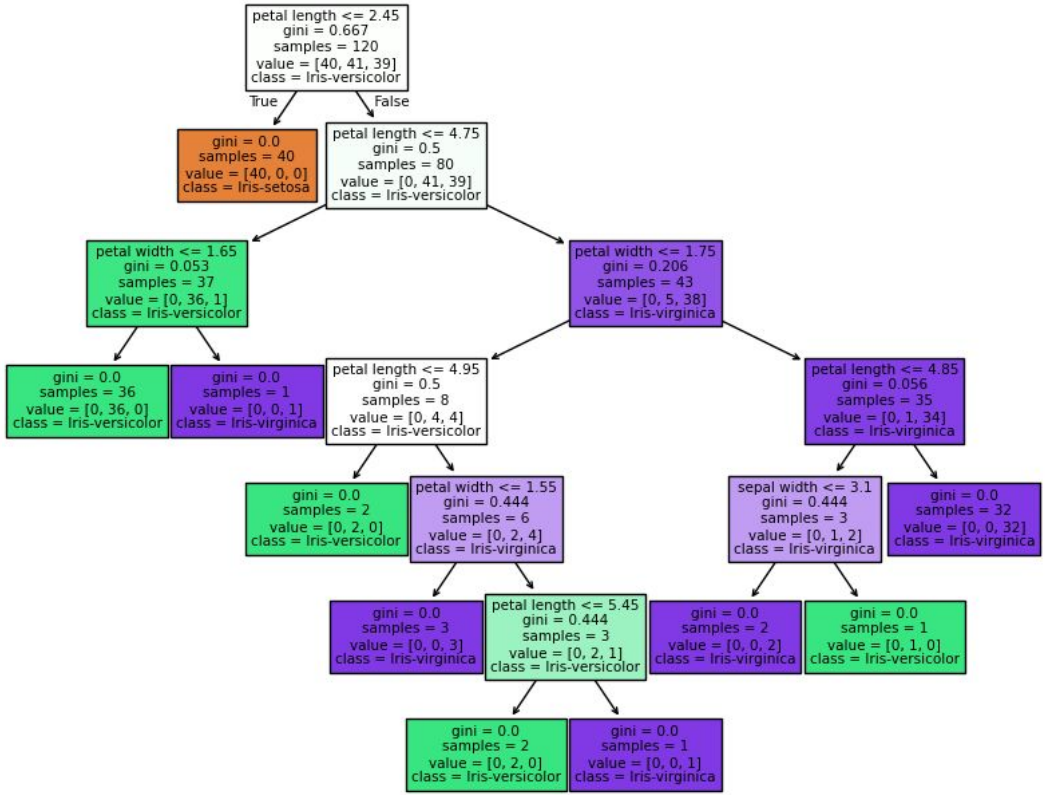
Key Metrics in Tree Explainers:

- **Feature Importance:**
  - Identifies which features contribute most to the predictions.
- **Partial Dependence Plots:**
  - Shows the relationship between a feature and the target outcome.

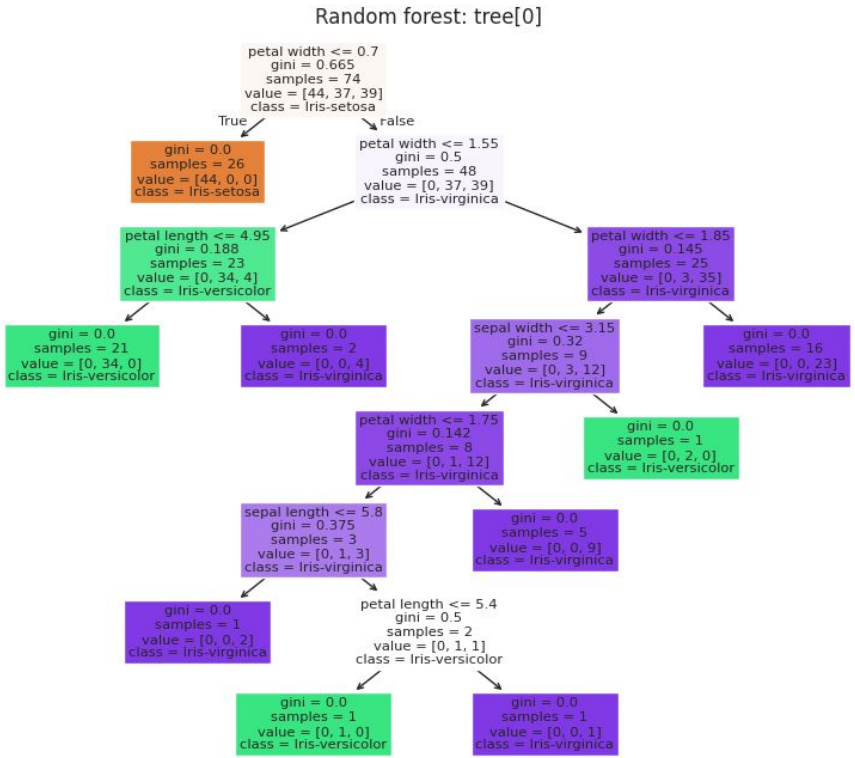
Model agnostic explainability techniques: Tree


```
{'uci_id': 53,
 'name': 'Iris',
 'repository_url': 'https://archive.ics.uci.edu/dataset/53/iris',
 'data_url': 'https://archive.ics.uci.edu/static/public/53/data.csv',
 'abstract': 'A small classic dataset from Fisher, 1936. One of the earliest known datasets used for evaluating classification methods.\n',
 'area': 'Biology',
 'tasks': ['Classification'],
 'characteristics': ['Tabular'],
 'num_instances': 150,
 'num_features': 4,
 'feature_types': ['Real'],
 'demographics': [],
 'target_col': ['class'],
 'index_col': None,
 'has_missing_values': 'no',
 'missing_values_symbol': None,
 'year_of_dataset_creation': 1936,
 'last_updated': 'Tue Sep 12 2023',
 'dataset_doi': '10.24432/C56C76',
 'creators': ['R. A. Fisher'],
 'intro_paper': {'ID': 191,
  'type': 'NATIVE',
  'title': 'The Iris data set: In search of the source of virginica',
  'authors': 'A. Unwin, K. Kleinman',
  'venue': 'Significance, 2021',
  'year': 2021,
  'journal': 'Significance, 2021',
  'DOI': '1740-9713.01589',
  'URL': 'https://www.semanticscholar.org/paper/4599862ea877863669a6a8e63a3c707a787d5d7e',
  'sha': None,
  'corpus': None,
  'arxiv': None,
  'mag': None,
  'acl': None,
  'pmid': None,
  'pmcid': None},
 'additional_info': {'summary': 'This is one of the earliest datasets used in the literature on classification methods and widely used in statistics and machine learning.  The data set contains 3 classes of 50 instances each, where each class refers to a type
of iris plant.  One class is linearly separable from the other 2; the latter are not linearly separable from each other.\n\nPredicted attribute: class of iris plant.\n\nThis is an exceedingly simple domain.\n\nThis data differs from the data presented in
Fishers article (identified by Steve Chadwick,  spchadwick@espeedaz.net ).  The 35th sample should be: 4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in the fourth feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa" where the errors are in the second
and third features.  ',
  'purpose': 'N/A',
  'funded_by': None,
  'instances_represent': 'Each instance is a plant',
  'recommended_data_splits': None,
  'sensitive_data': None,
  'preprocessing_description': None,
  'variable_info': None,
  'citation': None}}
```

Model agnostic explainability techniques: Tree



Model agnostic explainability techniques: Random Forest



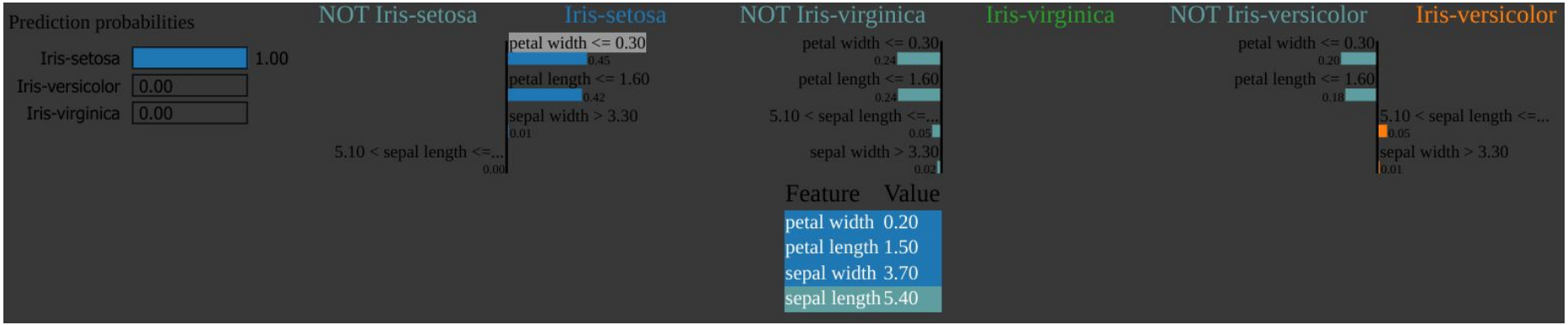
LIME (Local Interpretable Model-agnostic Explanations)

- Explains individual predictions effectively.
- Model-agnostic: Works with any black-box model.
- Flexible: Handles tabular, text, and image data.
- How LIME works?
  - Problem Setup:
    - Consider a trained ML model f that outputs predictions.
    - We want to explain why the model predicted a particular output $f(x)$ for an input x.
  - Core Idea:
    - LIME creates a simple, interpretable model (e.g., linear regression) to approximate f locally around x.
    - Uses perturbations (slight changes) to generate a new dataset, helping to understand how the model responds near x.

LIME (Local Interpretable Model-agnostic Explanations)

- Steps of LIME:
  - Perturb the Input:
    - For tabular data: Modify feature values (e.g., add noise, remove features).
    - For text: Mask words or replace them with synonyms.
    - For images: Modify patches of pixels (e.g., blurring).
  - Predict Using the Original Model:
    - Pass these perturbed inputs to the original model f to get predictions.
  - Weight Perturbations by Proximity:
    - Assign higher weights to perturbed samples that are closer to the original input x.
  - Train a Simple Model:
    - Fit a surrogate model (e.g., linear regression) to approximate f locally.
  - Extract Explanations:
    - Interpret the surrogate model's coefficients or rules to understand $f(x)$.

LIME (Local Interpretable Model-agnostic Explanations)



LIME (Local Interpretable Model-agnostic Explanations)

Limitations:

- Approximation Errors:
  - LIME's surrogate model is a simplified representation of the original model and may not always reflect its behavior accurately.
- Choice of Perturbations:
  - The method used to perturb data can affect explanations.
  - Example: Removing words in text may not reflect real-world scenarios.
- Computational Cost:
  - Generating perturbations and training a surrogate model for each explanation can be slow, especially for large datasets or complex models.

SHAP (SHapley Additive Explanations)

- SHAP (SHapley Additive Explanations) is a popular XAI technique for understanding predictions of ML models:
  - Developed using concepts from cooperative game theory.
  - Provides both local and global interpretability.
  - Based on Shapley values, ensuring fair attribution of contributions to features.
- Key Idea:
  - Features are players in a game.
  - The "payout" is the model's prediction.
  - The Shapley value quantifies each feature’s contribution to the prediction by averaging its marginal contribution across all possible subsets of features.
- Desirable Properties:
  - Additivity: The sum of feature contributions equals the total prediction.
  - Symmetry: Features contributing equally get the same value.
  - Null Player: Features with no impact have a contribution of zero.

SHAP (SHapley Additive Explanations)

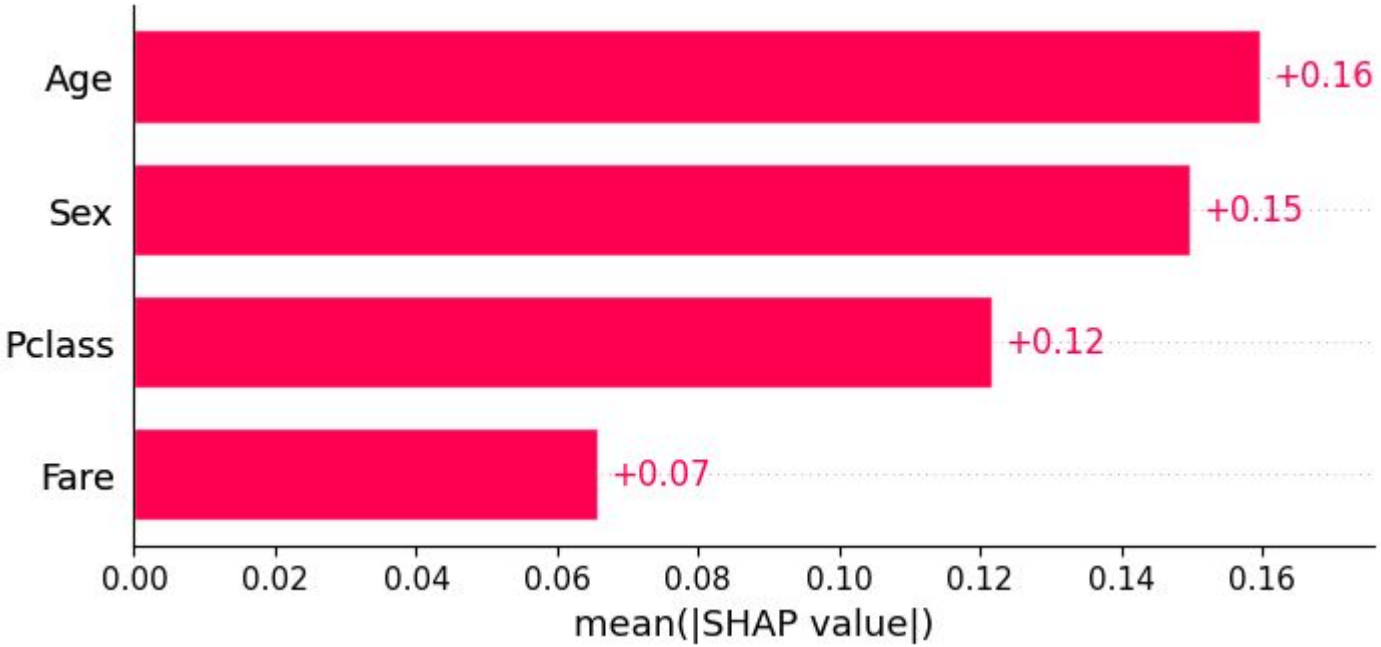
Why Use SHAP?

- Supports: TreeExplainer, DeepExplainer, GradientExplainer, KernelExplainer
- Local Explanations:
  - Explains individual predictions by showing feature contributions.
  - Example: For a loan approval system, SHAP can explain why a specific applicant's loan was approved.
- Global Explanations:
  - Summarizes feature importance across the entire dataset.
  - Identifies the most influential features.
- Visualization Power:
  - Force plots for local explanations.
  - Summary plots for global feature importance.
  - Dependence plots for feature interaction analysis.

SHAP (SHapley Additive Explanations)

- Input and Output:
  - Input: Trained model, data, and target prediction.
  - Output: Shapley values for each feature per prediction.
- Workflow:
  - Sampling Feature Subsets:
    - SHAP computes feature contributions using subsets of features.
  - Model Evaluation:
    - Evaluates the model with and without subsets of features to calculate marginal contributions.
  - Weighting Subsets:
    - Weight subsets based on the number of features, ensuring fair distribution.
  - Output Explanations:
    - Combines weighted contributions to calculate Shapley values.

SHAP (SHapley Additive Explanations)



SHAP (SHapley Additive Explanations)

Limitations:

- Computational Complexity:
  - Calculating Shapley values involves evaluating all subsets of features, which can be computationally expensive.
  - Approximation methods help but might lose some accuracy.
- Model Dependency:
  - Different models may produce different explanations for the same dataset.
- Interpretation Challenges:
  - Visualizations might be overwhelming for non-technical audiences.

SHAP vs LIME

- LIME provides simpler local explanations, but SHAP is more consistent and mathematically grounded.
- SHAP vs. Feature Importance:
  - Feature importance is global, while SHAP provides both global and local explanations.
- SHAP is a robust, mathematically rigorous method for interpreting ML models.
- It provides actionable insights at both local and global levels.
- While computationally intensive, its ability to explain complex models makes it invaluable in high-stakes domains.

ELI5 (Explain Like I’m 5)

- ELI5 (Explain Like I’m 5) is a Python library designed to demystify machine learning (ML) models.
- It provides simple yet effective tools to explain predictions from ML models and understand feature importance.
- Suitable for both traditional ML models (like Scikit-learn and XGBoost) and some deep learning models.

Key Features:

- Feature Importance: Shows which features contribute the most to predictions.
- Text Analysis: Explains predictions for text classification models.
- Permutation Importance: Measures feature importance by shuffling values and checking the impact on performance.
- Sklearn Pipeline Inspection: Visualizes preprocessing pipelines and intermediate results.
- Support for Debugging Models Explains model behavior, including weights, biases, and predictions.

How ELI5 Works

- Model-Agnostic: Works with any model, regardless of the algorithm used.
- White-Box Models: Offers insights into linear models like logistic regression by exposing weights and coefficients.
- Black-Box Models: Uses tools like permutation importance or decision paths for opaque models like random forests.

Advantages:

- User-friendly with clear, readable outputs.
- Works across a variety of models and tasks.
- Supports debugging and feature importance analysis.

Limitations:

- Limited support for deep learning models.
- Less visualization-heavy compared to tools like SHAP.
- Relies on tabular, interpretable datasets for meaningful outputs.

How ELI5 Works

```
Explained as: feature importances

Random forest feature importances; values are numbers 0 <= x <= 1;
all values sum to 1.

0.4260 ± 0.5043  petal length
0.3992 ± 0.5875  petal width
0.1491 ± 0.3210  sepal length
0.0258 ± 0.0520  sepal width
```

How ELI5 Works

```
Explained as: decision path

Features with largest coefficients per class.

Feature weights are calculated by following decision paths in trees
of an ensemble (or a single tree for DecisionTreeClassifier).
Each node of the tree has an output score, and contribution of a feature
on the decision path is how much the score changes from parent to child.
Weights of all features sum to the output score or proba of the estimator.

Caveats:
1. Feature weights just show if the feature contributed positively or
   negatively to the final score, and does not show how increasing or
   decreasing the feature value will change the prediction.
2. In some cases, feature weight can be close to zero for an important feature.
   For example, in a single tree that computes XOR function, the feature at the
   top of the tree will have zero weight because expected scores for both
   branches are equal, so decision at the top feature does not change the
   expected score. For an ensemble predicting XOR functions it might not be
   a problem, but it is not reliable if most trees happen to choose the same
   feature at the top.
```

Comparison

Feature	Eli5	SHAP	LIME
Ease of use	Very simple	Moderate	Moderate
Visualization	Limited	Extensive	Moderate
Model support	Broad (ML-focused)	Broad (ML-focused)	Broad (ML-focused)
Output	Text	Plot	Text/Plot

DeepLIFT

What is DeepLIFT?

- DeepLIFT (Deep Learning Important FeaTures) is a method for interpreting neural network predictions. It attributes the output of a model to its input features based on their contribution to the result.

Why is it important?

- Neural networks are often considered black boxes. DeepLIFT helps make these models interpretable by analyzing how much each input feature contributes to a prediction.

How it differs from other methods?

- DeepLIFT compares activations to a reference baseline and uses this comparison to assign importance scores. This can handle issues like vanishing gradients better.

DeepLIFT

Key Concepts

- DeepLIFT requires a reference baseline, which represents the "neutral" or "default" state of the input features.
- Example: For an image classifier, the baseline might be a completely black image or an average pixel value.

Contribution Scores

- DeepLIFT calculates contribution scores by comparing the activation of neurons in the input to their activation at the baseline.
- It assigns these scores to each feature, indicating its impact on the model's output.

Difference from Gradients

- While gradients measure the sensitivity of the output to infinitesimal changes in input, DeepLIFT looks at changes from the baseline, ensuring stable and interpretable attributions.

DeepLIFT

Step 1: Define the Baseline

- Choose a meaningful baseline input that represents "neutral" data. This varies based on the problem:
- Images: A black or average pixel-value image.
- Text: An empty or neutral text.

Step 2: Forward Pass Comparison

- Perform a forward pass on both the input data and the baseline. Calculate the difference between the activations of each neuron.

Step 3: Backpropagation of Contributions

- Propagate contributions backward from the output layer to the input layer to attribute importance to each feature.

DeepLIFT

Advantages

- Handles Vanishing Gradients
  - By looking at differences from a baseline, DeepLIFT avoids the problems of small gradients that occur in saturated activation functions.
- Computational Efficiency
  - DeepLIFT is faster than methods like SHAP for deep learning models, especially in large datasets.
- Interpretability
  - Provides clear and intuitive attributions for both positive and negative contributions of features.

Limitations

- Baseline Sensitivity
  - The choice of baseline significantly affects attributions, and selecting a meaningful baseline can be challenging.
- Non-additivity
  - Contributions from correlated features may not sum up intuitively.
- Limited to Specific Frameworks
  - Currently, supports Keras, TensorFlow, and PyTorch.

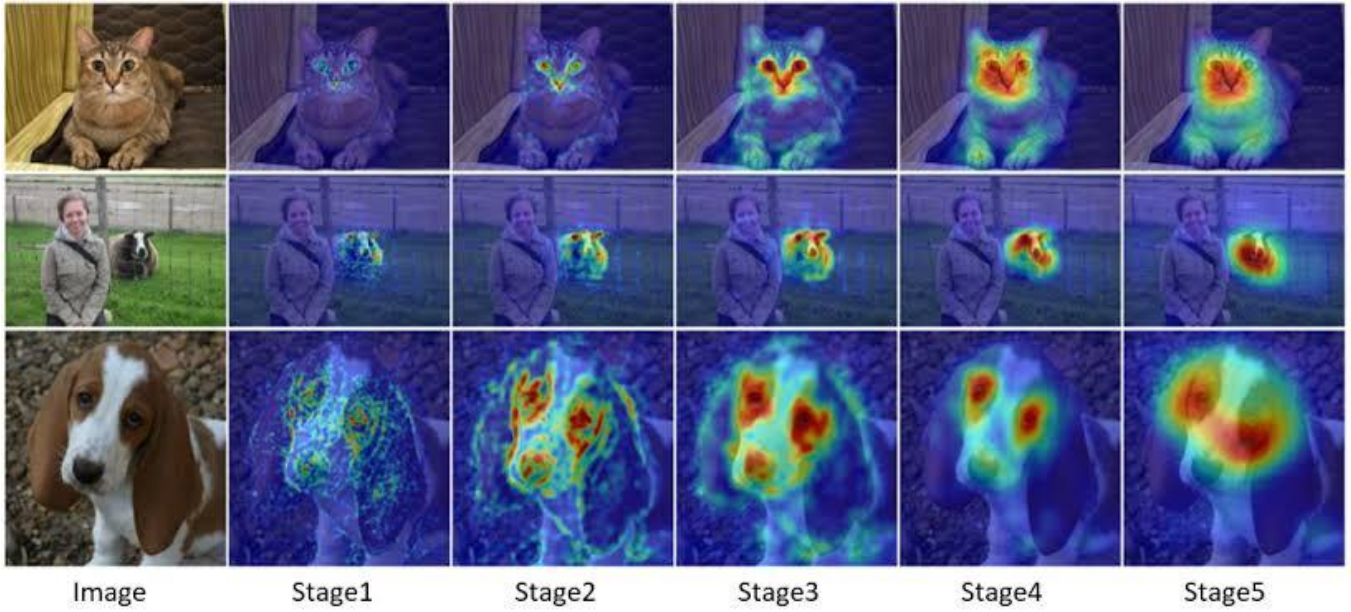
DeepLIFT

Applications

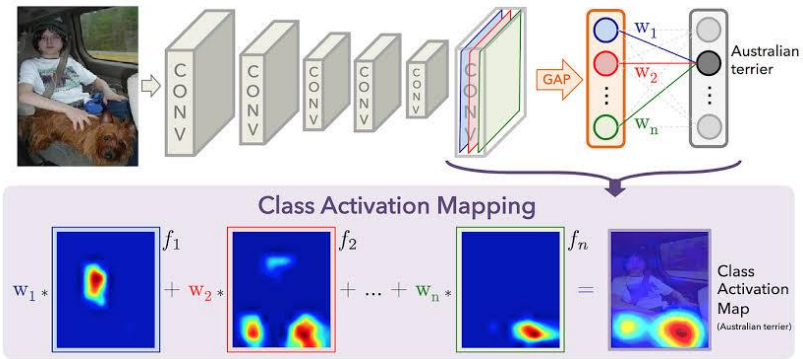
- Healthcare: Identifying which features (e.g., symptoms, biomarkers) influenced a diagnosis prediction.
- Image Recognition: Highlighting regions in an image that contribute to classification.
- Finance: Understanding model predictions for credit scoring or fraud detection.
- Natural Language Processing: Determining which words or phrases contribute most to a sentiment or classification result.

Class Activation Map (CAM) and GradCAM

CAM (Class Activation Mapping) and Grad-CAM (Gradient-weighted Class Activation Mapping) are techniques used for visualizing the regions of an input image that contribute to a model's decision. While they have similar goals, they differ significantly in methodology, usability, and flexibility.



Class Activation Map (CAM)



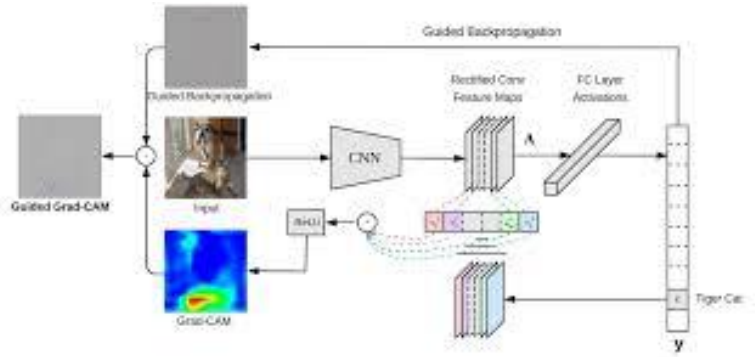
Key Characteristics:

- CAM is a simpler approach that works with Global Average Pooling (GAP) layers.
- It requires modifications to the architecture of the neural network. Specifically:
- The final fully connected layers are replaced with a GAP layer.
- This is followed by a classification layer.
- Uses the weights of the fully connected layer to compute the activation map.

How It Works:

- Take the output of the last convolutional layer (before GAP).
- Multiply each feature map by the corresponding weight from the classification layer.
- Sum these weighted feature maps to create a class-specific activation map.

Grad-Class Activation Map (CAM)



Key Characteristics:

- Grad-CAM does not require architectural modifications.
- It is architecture-agnostic and can work with any pre-trained CNN model.
- Uses gradients flowing back to the last convolutional layer to compute the importance of each neuron.

How It Works:

- Compute the gradient of the output (score for a specific class) with respect to the feature maps of the last convolutional layer.
- Average these gradients spatially to get the importance weights.
- Multiply the feature maps by these weights and sum them to produce the class activation map.
- Normalize the map to make it visually interpretable.

CAM vs Grad-CAM

Aspect	CAM	Grad-CAM
Dependency	Requires GAP layer (architecture-specific)	Works with any CNN model (architecture-agnostic)
Flexibility	Limited to models with modified architecture	Highly flexible; no architectural constraints
Computation	Directly uses feature map weights	Uses gradients to compute feature importance
Model Modification	Requires modifications	No modifications needed
Resolution	Lower resolution	Can achieve high resolution with Guided Grad-CAM
Use Cases	Simple and customized networks	Pre-trained models and complex networks

Scenarios

Key Differences Between CAM and Grad-CAM:

- CAM (Class Activation Mapping) requires the network to have a Global Average Pooling (GAP) layer followed by a fully connected layer. It directly uses the learned weights of the final layer to compute class-specific activation maps.
- Grad-CAM (Gradient-weighted CAM) works with any pre-trained network without modification. It computes class-specific gradients with respect to the feature maps of a chosen convolutional layer to generate the activation map.

When to Use CAM vs. Grad-CAM?

- Use CAM when:
  - You are designing a custom architecture with GAP layers.
  - Simplicity and speed are preferred.
- Use Grad-CAM when:
  - You are working with pre-trained models.
  - Flexibility and adaptability are critical.
  - You need to combine with other visualization methods.

Any Questions?