EXPLAINABLE AI (XAI)

Explaining the Predictions of any Classifier

Niharika Dobanaboina
Graduate Student, MSBA
Kent State University

ABSTRACT

This paper focuses on explaining the need of Explainable AI (XAI) in increasing the trust and transparency of Machine Learning models among decision-makers and customers. When making critical decisions using the predictions of ML models in fields like Healthcare, Finance, Investments, Fraud detection etc. it becomes crucial to know the reasons as to why a model has predicted a certain outcome. Two techniques in specific, namely SHAP and LRP have been discussed in this paper. Both techniques are commonly used XAI techniques which can be used to interpret the results of different Machine Learning models like Random Forest, GBM and Deep Neural Networks. The paper describes the concept behind SHAP and the calculation of the contribution of each input characteristic to the model's final output. It further focuses on implementation of SHAP technique using *shap* python library. Layer-wise Relevance Propagation (LRP) is another technique which operates by propagating the predictions made by a neural network in the backward direction by using *Conservation* property. At the end, this paper discusses the future scope of XAI in various areas of AI and ML.

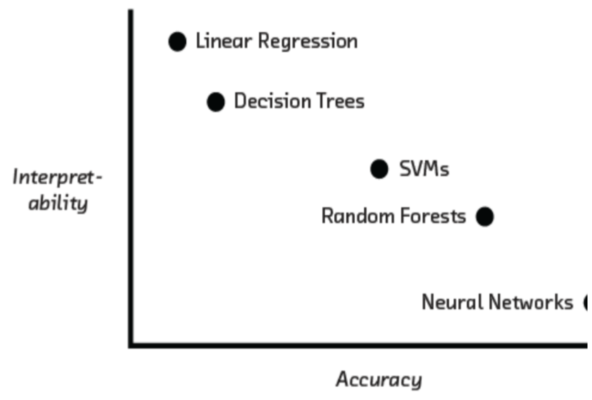
INTRODUCTION

Machine Learning models are built in such a way that the model learns the patterns and trends from the data and use these patterns to predict on the unseen data. While using advanced and complicated techniques like deep neural networks, it’s difficult for humans to understand as to on what basis certain predictions were made by the model. Even the data scientists who developed those models cannot explain the reasoning behind the decision-making process of algorithms. Since machine learning is applied in a wide variety of fields like healthcare, criminal justice, finance etc., the lack of explainability of a model becomes an issue.

Explainable AI(XAI) refers to the set of methods that allows humans to understand and trust the results created by machine learning algorithms.

XAI can be used to provide transparency and increase trust among users, while helping in detecting and mitigating bias of a model. One of the main advantages of XAI is that it seeks to address the concept of INTERPRETEABILITY of a model. Considering the real-world problem, we're aiming to solve, interpretability enables us to comprehend what a model is learning, the other information it has to offer, and the reasons behind its decisions. Interpretability is necessary when model metrics are inadequate. By comparing a model to its training environment, model interpretability helps us to forecast how a model will perform under various test scenarios. In circumstances involving responsibility, such as with autonomous vehicles, explainable AI systems may be useful since if something goes wrong, a human is still responsible for their actions. The explainability techniques that use textual descriptions that can be understood by humans to explain the thinking behind a model's prediction are used to train the explainable AI models. Natural language processing (NLP), computer vision, medical imaging, health informatics, and many more fields of artificial intelligence use explainability techniques today.

Lack of INTERPRETABILITY and PERFORMANCE according to the type of models:



Neural networks are powerful models, but harder to interpret than simpler and more traditional mc

SHAP is a prime example of explainable AI, which refers to methods and techniques that help people in interpreting the choices made by AI systems. By considering all potential feature combinations and their contributions, SHAP calculates the Shapley values for each feature. This method makes sure that each feature's importance is calculated in connection to the other features rather than by itself. The SHAP values that emerge give a thorough description of how the input features influence the model's output.

1. SHAP

SHAP (SHapley Additive exPlanations) employs the idea of Shapley values from cooperative game theory to explain the results of any machine learning model. The fundamental idea behind SHAP is to use the Shapley values to determine the relative contributions of each input feature to the final output after attributing the prediction of a model to various input features.

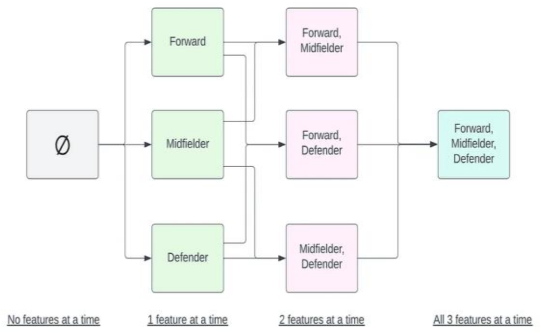
The 2017 publication SHAP (Shapley Additive Explanations) tries to break down the prediction and illustrate the influence of each attribute. The idea of shapely values, which Lloyd Shapley first introduced in the realm of game theory in 1952, serves as the foundation for SHAP.

According to game theory: Imagine a soccer match. A group of participants works together and benefits in some way from it overall. What benefits can each participant expect to receive and how significant a role does he or she play in the overall teamwork? One answer to this question is given by the Shapley value. SHAP is concerned with the predictability of a model locally. Players are its features, and a game is an observation. We are looking for each player's (or feature's) role in the result for a specific game.

SHAP supports a wide range of data types, such as numerical, categorical, and text data, and it may be used with a variety of machine learning models, such as decision trees, neural networks, and random forests.

Understanding The Logic Behind SHAP:

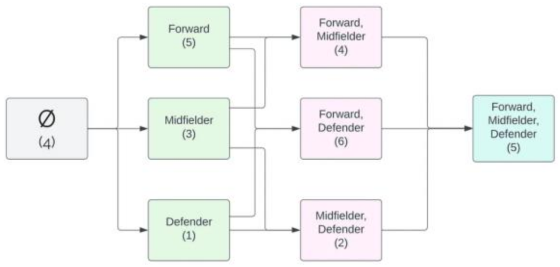
Let's say we wish to forecast how many goals (perhaps in decimal form) a team will score during a soccer match, and we understand three positions: striker, midfielder, and defense (goalkeeper is rarely involved in goal scoring:).



Different combination of features (i.e. players info)

The first column in the image above represents 0 features, meaning that we don't know anything about any of the features (players). A node in the second column denotes the precise knowledge we have on a given feature. Like how a node in the third column represents, those 2 attributes are precisely what we are aware of. We have information about all the features, as shown by the last column.

Shapley values are calculated based on the idea that every potential combination's outcome is considered. We'll assume we have 8 regression models trained on the same training data but with various feature combinations because there are 3 features (forward, midfielder, and defender). We wish to anticipate the result of the new game we have now. Below are the forecasts from 8 different regression models:

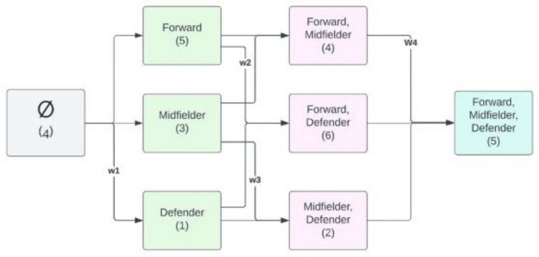


It is important to note that the model with no features, represented by column 1, will only forecast the average number of goals across all training observations (4).

The "Forward" node in the first column indicates that if we only knew the information on the team's "forward" players, we would predict the team would score five goals.

Understanding columns sets a little more thoroughly A node with "forward, defender" information predicted 4 goals, but a node with "forward" information only predicted 5. It is known as the marginal contribution (MC) brought by that extra feature and the gap of one goal is because of the additional information about the additional feature defender. Consequently, the MC of the defender is one goal for this connection between the "forward" and "forward, defender" nodes. By examining prediction differences between the nodes in successive columns, we will similarly identify MC for all features.

A feature's SHAP value will be determined by weighing the MC for that feature. In this instance, the defender's SHAP value will be determined. Please refer to the graphic below for the weights assigned to each defensive edge.



SHAP value for defender =

w1 (1–4) + w2 (6–5) + w3(2–3) + w4(5–4) where *w1 + w2 + w3 + w4 = 1*

Marginal weight calculation properties.

1. The total weights should equal 1 because it is a weighted average.

That is, w1 + w2... w4 = 1

2. The weights of all MC to 1-feature-models added together equal the weights of all MC to 2-feature-models added together, and so forth.

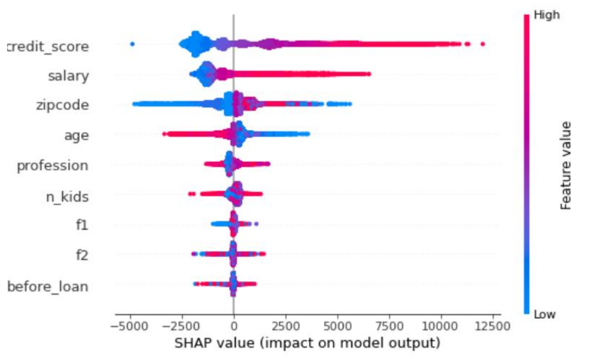
That is, w1 = w2 + w3 = w4.

3. For each f, all the MC weights to f-feature models should be equal. i.e., w2 = w3

SHapley values helps to achieve:

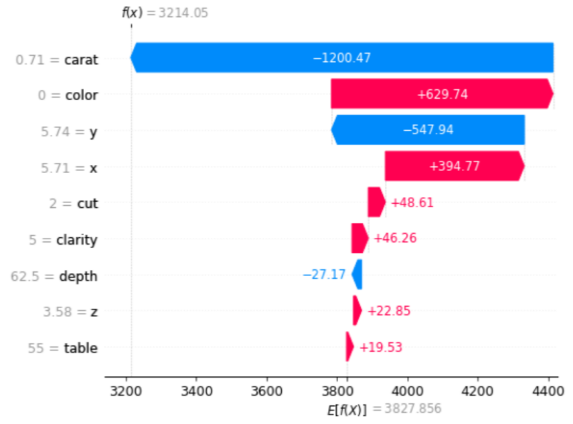
Global Model Interpretability:

Imagine developing a classification model for loan applications while working at a bank. Your management wants you to describe what (and how) certain factors affect the conclusions drawn by your model. With the use of SHAP values, you may provide a specific response that includes information on which features result in more loans, and which features result in more rejections. Your boss is pleased with you since now he can create fundamental rules for potential bank clients to improve their chances of obtaining a loan.



Local Interpretability:

Let's say that one of the applications that the bank received a few days ago is rejected by your model. The client says that by adhering to all the requirements, he was guaranteed to be approved for a loan from your bank. Now that the candidate was rejected by your model, you are required by law to give a justification for your decision. All cases can be independently examined using Shapley values without having to consider how they relate to other samples in the data. In other words, your interpretability is local. To help the customer who is complaining, you extract the Shapley values and explain which elements of their application were the reason it was rejected. You use a plot like this to show them wrong:



Example to explain SHAP using Linear Regression:

Standard linear regression is one of the most straightforward model types, therefore in the section below, we train a linear regression model on the California housing dataset. Our objective is to estimate the natural log of the median home price from 8 different variables using this dataset, which includes 20,640 blocks of houses in California in 1990.

MedInc- block group median income

HouseAge- Age of the average houses in the block group

AveRooms- Average number of rooms per home.

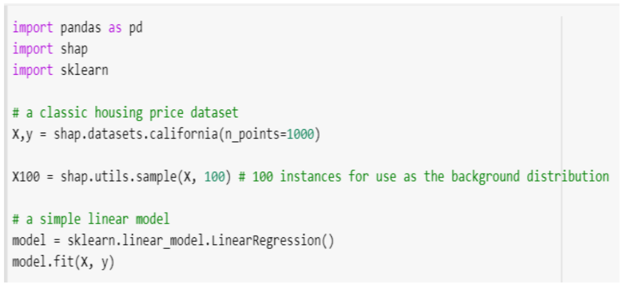
AveBedrms- Average bedrooms per household

Population - population by block

AveOccup- Average number of household members

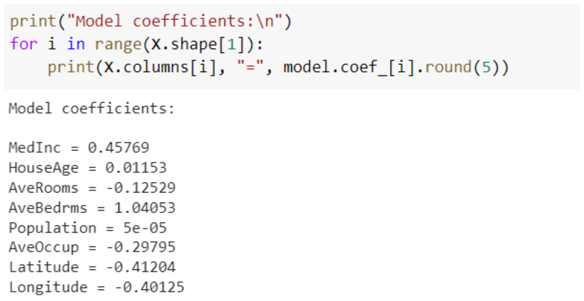
latitude -Block group for latitude

longitude-Block group longitude


```
import pandas as pd
import shap
import sklearn

# a classic housing price dataset
X,y = shap.datasets.california(n_points=1000)

X100 = shap.utils.sample(X, 100) # 100 instances for use as the background distribution

# a simple linear model
model = sklearn.linear_model.LinearRegression()
model.fit(X, y)
```

Evaluating the model coefficients:

The most typical method for understanding a linear model is to look at the coefficients generated for each feature. These coefficients indicate the degree to which changing each of the input features alters the model's output:


```
print("Model coefficients:\n")
for i in range(X.shape[1]):
    print(X.columns[i], "=", model.coef_[i].round(5))
```

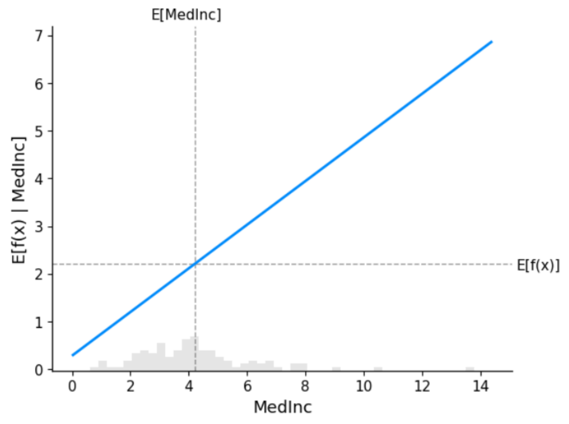
```
Model coefficients:

MedInc = 0.45769
HouseAge = 0.01153
AveRooms = -0.12529
AveBedrms = 1.04053
Population = 5e-05
AveOccup = -0.29795
Latitude = -0.41204
Longitude = -0.40125
```

Coefficients are a fantastic way to predict what will happen when we alter the value of an input feature, but they are not a very good way to assess the overall significance of a feature on their own. This is so because each coefficient's value is based on the size of the input features. The coefficients for the HouseAge feature would be 0.0115 / (3652460) = 2.18e-8 if, for instance, we were to assess a home's age in minutes rather than years. Despite having a much higher coefficient value, it is obvious that the number of years since a house was built is not more significant than the number of minutes. This indicates that, in a linear model, the size of a coefficient is not always a reliable indicator of a feature's significance.

A more thorough depiction using partial dependence charts.

Understanding how changing a feature affects the model's output as well as the distribution of that feature's values is required to comprehend a feature's significance in a model. We can create a traditional partial dependence plot and display the distribution of feature values as a histogram on the x-axis to see this for a linear model:

```
shap.partial_dependence_plot(
    "MedInc", model.predict, X100, ice=False,
    model_expected_value=True, feature_expected_value=True
)
```



The expected value of the model when used with the California housing dataset is represented by the gray horizontal line in the plot above. The median income feature's average value is shown by the vertical gray line. Keep in mind that the intersection of the two gray expected value lines is always crossed by the blue partial dependence plot line, which represents the average value of the model output when the median income feature is fixed to a certain value. This intersection point serves as the partial dependence plot's "center" in relation to the data distribution. When we move on to Shapley values, the effect of this centering will become apparent.

using partial dependency graphs to read SHAP values.

To assign credit for a model's output among its input features, Shapley value-based explanations of machine learning models leverage fair allocation findings from cooperative game theory. Matching the input features of a model with the players in a game and the model function with the game rules are both necessary to combine game theory with machine learning models. We require a method for a feature to "join" or "not join" a model as in game theory a player can choose whether to play. The most typical definition of what it means for a feature to "join" a model is to say that the feature has "joined a model" when its value is known and that it has not joined a model when its value is unknown. We integrate out the other features using a conditional expected value formulation to evaluate an existing model f when just a subset S of features are included in the model. There are two possible versions of this formula:

$$E[f(X) \mid X_s = x_s]$$

(Or)

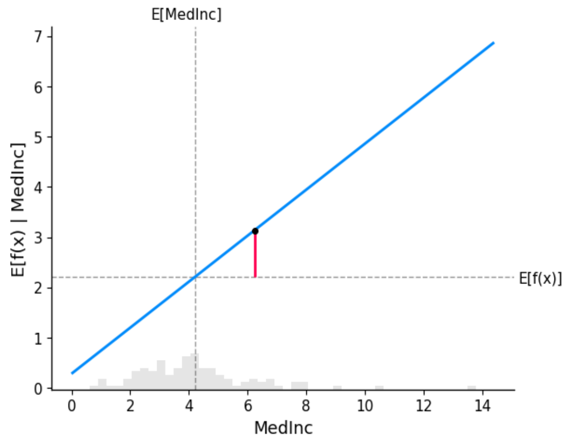
$$E[f(X) \mid do\ (X_S = x_s)]$$

Because we observe them in the first form, we are aware of the values of the features in S. Because we set the features in S in the second form, we are aware of their values. Both because it is significantly simpler to compute and because it tells us how the model would behave if we were to interfere and change its inputs, the second form is typically preferred in most cases. The second formulation is the only one we'll cover in detail in this session. Shapley values applied to a conditional expectation function of a machine learning model will also be referred to by the more precise name SHAP values in this paper.

SHAP values can be extremely difficult to calculate, however because linear models are so straightforward, we can read the SHAP values directly from a partial dependence plot. The SHAP value for a certain feature i is just the difference between the expected model output and the partial dependence plot at the feature's value x_i when we are explaining a prediction f(x) :

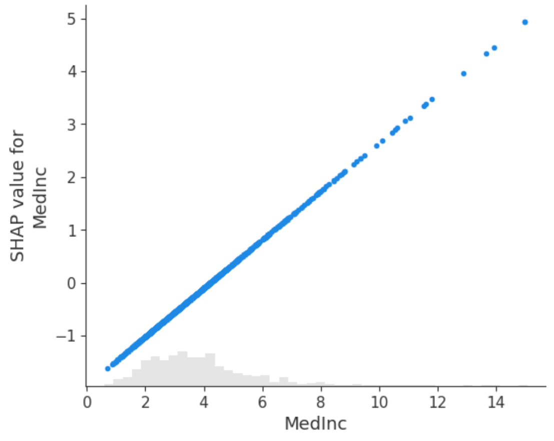
```
# compute the SHAP values for the linear model
explainer = shap.Explainer(model.predict, X100)
shap_values = explainer(X)

# make a standard partial dependence plot
sample_ind = 20
shap.partial_dependence_plot(
    "MedInc", model.predict, X100, model_expected_value=True,
    feature_expected_value=True, ice=False,
    shap_values=shap_values[sample_ind:sample_ind+1,:]
)
```

```
Exact explainer: 1001it [00:15, 43.24it/s]
```



The conventional partial dependence plot and SHAP values have a strong connection, so if we plot the SHAP value for a particular feature throughout the entire dataset, we will precisely trace out a mean-centered version of the partial dependence plot for that feature:

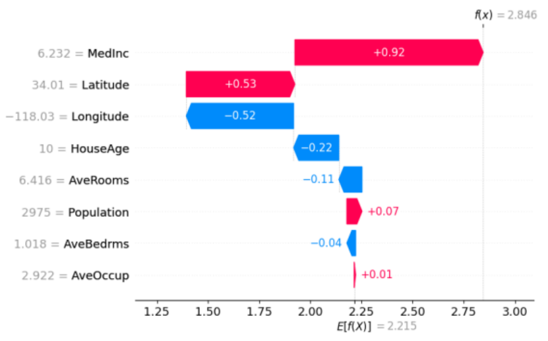
```
shap.plots.scatter(shap_values[:,"MedInc"])
```



The Additive Nature of Shapley Values:

The fact that Shapley values always add up to the difference between the result of the game when all players are present and the result of the game when no players are present is one of their fundamental characteristics. The difference between the baseline (anticipated) model output and the present model output for the prediction being explained will always equal the SHAP values of all the input features for machine learning models. The simplest method to do this is to visualize this using a waterfall plot that begins at our initial prior assumption for the price of a property, f(x).

```
# the waterfall_plot shows how we get from shap_values.base_values to model.predict(X)[sample_ind]
shap.plots.waterfall(shap_values[sample_ind], max_display=14)
```



2. LAYER-WISE PROPAGATION LAYER (LRP)

The goal of LRP is to explain any neural network's output in the context of its input. The explanation provided by LRP might, for instance, be a map of which pixels in the original image contribute to the diagnosis and to what amount if your network predicts a cancer diagnosis from a mammogram (an image of breast tissue). You may easily use this strategy to classifiers that have already undergone training because it does not interfere with the network's training.

The conservation property, which requires that what has been received by a neuron be distributed in an equal amount to the lower layer, governs the propagation method used by LRP. Let j and k represent neurons in the neural network's second and third layers, respectively. Applying the following rule allows relevance scores (Rk) k at a given layer to propagate onto neurons of a lower layer:

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k.$$

The quantity zjk models the contribution made by neuron j to the relevance of neuron k. The conservation property is enforced by the denominator. Once the input features are reached, the propagation process is over. It is simple to confirm the layer-wise conservation property jRj=kRk and, by extension, the global conservation property iRi=f(xx) if the rule is applied to each neuron in the network.

LRP was used to find biases in datasets and commonly used ML models. It was also used to glean fresh insights from efficient ML models, for instance, in the field of face expression detection. LRP has been used to locate pertinent features for localizing audio sources, to locate interesting locations inside channel traces, and to locate EEG patterns that can be utilized to deduce judgments in brain-computer interfaces. LRP has been applied in the biomedical field to explain therapy predictions, detect subject-specific traits in gait patterns, and emphasize pertinent cell structure in microscopy. The application of an extension known as CLRP to highlight pertinent molecular regions in the context of protein-ligand scoring was the last step.

LRP Rules for Deep Rectifier Networks

We explore the application of LRP to rectifier (ReLU) nonlinear deep neural networks, undoubtedly the most popular option in current applications. It contains popular image recognition designs like VGG-16 and Inception v3, as well as neural networks utilized in reinforcement learning. The neurons in deep rectifier networks are of the following type:

$$a_k = \max\left(0, \textstyle\sum_{0,j} a_j w_{jk}\right).$$

The sum 0, j includes one additional neuron to reflect the bias in addition to all lower-layer activations (aj)j. To be more specific, we assign a0=1 and say that w0k is the neuron bias. We outline these networks' features and provide three propagation rules for them.

Basic Rule:

This rule redistributes according to how much each input contributed to the activation of the neurons in Eq.

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$$

Basic features of this rule, like (aj=0) (wj: =0) Rj=0, make concepts like zero weight, deactivation, and absence of connection coincide. Even though this rule appears to be obvious, it has been demonstrated that when applied uniformly to the entire neural network, it results in an explanation that is equal to gradient input (cf.

[47]). Since a deep neural network's gradient is frequently noisy, as we mentioned in the introduction, one needs to design more durable propagation rules.

Epsilon Rule (LRP-ϵ)

The basic LRP-0 rule can first be improved by including a modest positive term in the denominator:

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$$

When the contributions to the activation of neuron k are insufficient or conflicting, the function of is to absorb some importance. Only the most important explanation components survive the absorption as grows larger. Usually, reduced noise and explanations with fewer input features result from this.

Gamma Rule (LRP-γ)

The effect of preferring the effect of positive contributions over negative contributions is another improvement that we introduce here:

$$R_j = \sum_k \frac{a_j \cdot (w_{jk} + \gamma w_{jk}^{+})}{\sum_{0,j} a_j \cdot (w_{jk} + \gamma w_{jk}^{+})} R_k$$

The parameter determines how much beneficial contributions are valued. Negative contributions begin to disappear as rise. The extent to which positive and negative relevance might increase throughout the transmission phase is constrained by the prevalence of positive contributions. This makes for more solid justifications. With the LRP-rule, the concept of considering positive and negative contributions unequally was first put forth in (see Appendix 10.A). Additionally, selecting makes LRP-equal to LRP-10, the z+ -rule, and 'excitation-backprop'.

Future Scope of Explainable AI

Scientists expect that explainable AI will advance and improve as they develop new techniques and algorithms for understanding and explaining AI systems. Explainable AI will likely become a standard requirement for AI systems used in critical applications as AI technology advances, making the capacity to explain its decision-making processes increasingly more important. The future potential of explainable AI is vast and offers great promise for enhancing the design and implementation of reliable AI systems.

In future, XAI can help with upholding moral and legal requirements, such as GDPR and AIA, and guarantee regulatory compliance. Second, by encouraging openness, minimizing bias, and enhancing accountability, it can improve trust and acceptance of AI systems. Thirdly, it can encourage cooperation between people and AI systems, making interactions and decision-making more effective. Finally, it may open new applications in industries where transparency and interpretability are essential, like healthcare, banking, and transportation.

CONCLUSION

Due to the demand for more accountability and transparency in AI systems, explainable AI has become a crucial topic of study in recent years. With growing demand for AI systems that are both powerful and reliable, explainable AI is predicted to become increasingly important as the field of AI develops and expands. Even if there are still many issues to be resolved, the development of explainable AI approaches and algorithms has made significant progress thus far. This development has a lot of potential to advance the creation and use of reliable AI systems in the future. Explainable AI approaches and algorithms will keep developing, enabling interaction between people and AI systems, expanding the applications and use cases for AI, and assuring adherence to moral and legal standards.

References:

https://www.ibm.com/watson/explainable-ai

https://insights.sei.cmu.edu/blog/what-is-explainable-ai/

https://medium.com/@gauravagarwal_14599/explainable-ai-understanding-the-shap-logic-586fcf54c1b9

https://www.kaggle.com/code/bextuychiev/model-explainability-with-shap-only-guide-u-need

https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html

https://link.springer.com/chapter/10.1007/978-3-030-28954-6_10