Explainable Artificial Intelligence & Machine Learning
University of Tulsa
Fall 2022
Bryan Lavender
bal7708@utulsa.edu

XAI Literature Review with Reproduction and Examples

Abstract

Explainable Artificial Intelligence (XAI) is becoming a widely adopted field of research within the scientific community. With the impact of deep learning models' accuracy on complex applications, it is important to have assurance and trust for both machine learning (ML) developers and model users. This literature review intends to summarize the methodologies and technologies currently behind explanations as well as present a brief understanding to the realm of XAI and its nomenclature.

Data Description

The data being used in the examples is a data set consisting of over 1300 stocks. The variables of this data set are the stock name, the Earnings Per Share (EPS), the Beta (Defined by the multiple increase compared to the SP 500) and the prior 5-year percent increase. For regressive data sets, the outcome variable is the percent increase over 1 year investment given the year (i.e. the data set for 2018 uses performance over past 5 years prior to but not including 2018 and the performance at 2018). For classification, I optimized this set of stocks using the GMM algorithm with a set of 4 differing classes. The classes average performance for the year 2018 are as follows:

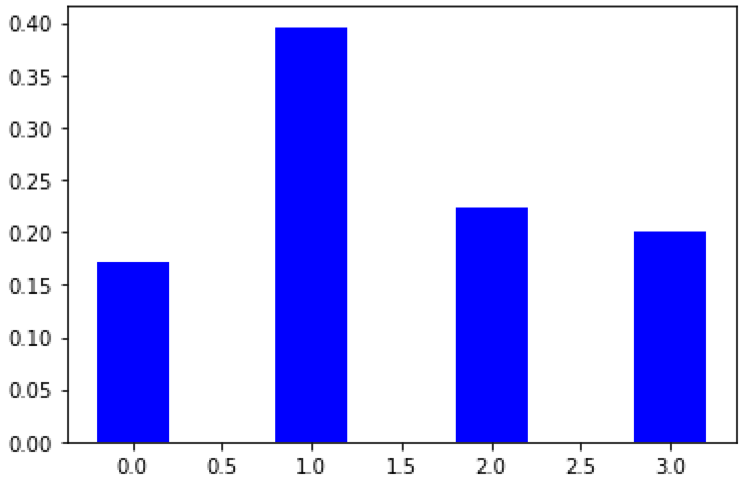


Figure 1: 4 classes separated by the GMM algorithm of the 1300+ stocks in the data set. As shown, class 1 has a 40% increase for the year. This portfolio will be our "best" ans used for classification

There are three different models made with this data using ensemble methods. The first is a binary decision on rather or not a stock exists in the optimal portfolio. The next is a multi-class model that provides the specific portfolio of which the model is classified as. The final is a regressive ensemble tree that maps the three features, EPS, Beta, and Performance, to the percent increase for the year.

0.1 Paper Layout Description

Nomenclature

Here we define the current nomenclature used within XAI and this paper. Many of these definitions were obtained by [1], a great paper for a general overview of technologies in XAI. We start with the first five most important definitions: *Explainability, Understandability/ Intelligibility, Comprehensibility, Interpretability, Transparency.* It is important to define these directly for their further use within this literature review.

- *Explainability*: the notion of an explanation generated to a human that is an accurate proxy of a decision and is comprehensible to the human.
- *Understandability/ Intelligibility*: the characteristics of a model to enable a human to understand its functionality without the need for explaining the internal functions of the model
- *Comprehensibility*: The ability for a model to represent its learned knowledge in a human understandable fashion.
- *Interpretability*: The ability to provide meaning in understandable terms to a human.
- *Transparency*: The ability for a model to be understandable on its own.

General Ideas

So far, in the literature, there have been a surprisingly discrete amount of concepts in XAI. Explanations are relatively general, as the audience it is trying to reach is based on the role the interacted takes. With our definition of Explainability, I have found three general outcomes of explanation technologies:

- Explanations For Model Confidence: These explanations are for a model developer or allow for an understanding of how a model acts under bias' or differing cases. These are more global methods than local or surrogate methods.
- Explanations of Model Reasoning: These explanations are directed more towards local cases and justification among decisions. Many of these are local and surrogate model explanations that are contrastive in nature, pulling from local points in the hypothesis space.
- Explanations of Model Actions: These are typically high level, learned and textual/visual explanations that generalize the actions of a model. Although they are

produced by learned or even deep surrogate models, these are more intended for users, identifying particular effects within a space instead of justifying them.

Having these three general ideas of explanations, there are many different rigid-forms of technologies to provide one of these three. We will go over a listing of technologies later, but the paper [1] presents 6 different technical forms of explanations:

- Textual Explanations: learned, generated text for decisions made by a model
- Visual Explanations: Visualization of a models behavior/objects of weight in decision making
- Local Explanations: Explanations for a singular solution given by a model
- Explanations by Example: Explanations given by representative outcomes
- Explanations by Simplification: Also known as surrogates, it is a simpler or more transparent model explaining a more complex model
- Feature Relevance: Explanations on the importance or effect of features within a model.

Interpretable Models

Interpretable Models are models with high transparency. The most common of these are low level, statistical models that are based on regression or decision trees. These models are typically used as surrogate models within explanation generation for larger or more complex models. They can be used to define hypothesis space or show feature relevance. Eight methodologies are mentioned within [2] and their pros and cons are listed below:

- Gaussian Classifiers:
  Gaussian Classifiers, such as Naive Bayes, use statistical probabilities and variances in data to make a prediction of a classification. These are very low-level conditional probabilities and show basic information over the data. Pros: Classifications made are easily explained by past data. The rules are readable in the data. Cons: Large data sets can obscure trends in the explanation. Smaller-rule chunks may ease this. Most predictions are statistically based.
- K- Nearest Neighbors:
  K-Nearest Neighbors is a classification problem where the determination of a point is made by its surrounding neighbors. This is very clear, as the hypothesis space can be explained by other classifications, although it is very low-level. Pros: Easy reading and explanation on classifications. Easily mappable and justifiable when generating an explanation. Cons: Larger-dimensional data can obscure actual relationships. The low-level evaluation of this model becomes less reliable as dimensionality is scaled.
- Linear Regression:
  Linear regression is mapping a function to input variables as closely as possible to the output or prediction. This is an incredibly transparent model and interpretable in the fact that there is a 1 to 1 relationship between the predictors and the outcome.

Pros: This model has a 1 to 1 relationship with features and variables. It is easy to determine the accuracy of a model with the sum of square error, and the variability of a model with the R-squared value. Importance of variables might be deceiving in the coefficients of the function given, but the T-value allows for us to note the variability of as well as effect of certain coefficients, therefore making this great for feature-importance explanations. Cons: On top of the instability of non-linear models, no feature is shown as dependent on another, in fact if a feature is dependent, it could skew the model's accuracy. Also, there is an assumed constant variance and normal distribution of error when calculating SSE and R-Squared value. Finally, the coefficient can be unintuitive due to its dependence on all other variables.

- Logistic Regression:
  Logistic Regression is almost the same as linear regression but attempts to turn the output into a probability. This statistical model is good for classification among linearly separable data sets. Pros: This model, again, has all the advantages as Linear Regression has, although with the probabilistic output, it is slightly better in interpreting performance accuracy and in-variance. Cons: This model suffers from the same as linear regression, but the weights might be less interpretable due to this being a classification problem. Also, complete separation is a problem in that if a feature perfectly separates two classes, it will never converge.

- GLM using GAM's:
  Generalized Linear Models (GLM's) are like linear regression without the expected normal distribution of features. This means keeping the coefficients but allowing non-Gaussian outcomes to connect the expected mean through a non-linear function. To capture non-linearity, we use Generalized Additive Models (GAM's) that might bound or apply better feature interactions for the predictors of a model. This is like, instead of having a sum of coefficients, you have a sum of equations for each predictor that allow for non-linearity. Pros: There is still a 1 to 1 relationship between the outcome and the predictors. On top of this, there is a better understanding of the distribution a model might follow when it is combined with other features as well as the distribution of predictors. Cons: Due to the immense number of ways a GLM can be made, a linear model is less interpretable and rely on assumptions of data. This means explanations generated may be biased or incorrect.

- Decision Trees:
  Decision trees are statistical models that are great for classification and alright for generalized regression. They consist of features being 'split' until the leaf of a split is determined, being a classification. Pros: These create a great way to capture interactions between features as well as a good visualization for how features are divided for outcomes. It is easily interpretable when it comes to which features effect the model, as the point of which a feature has an effect is a node. This creates a naturally transparent model. Cons: Trees do not show linear relationships among features. This, and the statistical nature of trees being generated may make it unstable in explanations (as some features may have more importance when generating a tree). Finally, the depth of a tree also creates less interpretability, as it is farther trees that overwhelm data understandability.

- Decision rules:

Decision rules are a more generalized version of decision trees. These allow different algorithms to focus on different feature values such as discrete features, and produce verbal descriptions following an IF x AND y THEN z format. With this said, it also produces a listing of relationships and a rather speedy classification time. Confidence can also be given when calculating if-then rules upon output. Pros: Decision rules are great for linguistic explanations of data already visible and can be as accurate as decision trees. Decision rules are more robust, as they can capture more than one feature in a rule. Cons: Decision rules sometimes must be categorical unless time is extended in training. Decision rules also completely neglect regression. It seems decision rules become unstable when dealing with continuous values.

- RuleFit:
  Rule fit, at least to me, seems like a conjunction of rules that are set aside from form a tree but still developed in the form or fashion of decision trees or decision rules. The main difference is the feature interaction is automatically calculated, as well as feature importance, when generating these rules. Pros: These models automatically have literary explanations as well as feature interaction calculations and feature importance calculations within the model generation steps. These work for both classification and regression problems such as decision trees. Since they are proposed as rules, deep tree formations turn into a string of rules met, making them less difficult to understand. Cons: Many rules may be useless in rule generation. Although Rules may capture feature interactions, the entirety of the model is linear.

Global Models

Global models aim to interpret a model as a whole. This includes generalizing the patterns of a model and determining the relative effects features have on a model. Seven different models are exampled for global methods.

Partial Dependents Plot (PDP's)

Partial Dependence Plots (PDP's) is a statistical plot that aims to answer the question "at what values of a feature are more important to the model" This is done by calculating the probability effect on the model of a particular value of the feature's value space. We can do this for the entire value space to graph the PDP. This allows us to see which values of a particular feature are important.
For Binary class values, it is just the feature values that are calculated. For multi-class models, we need to focus on which class the feature values become important. For regressive models, this calculates which values provide the most variance in the output of the model.
An example for our stock data set is given below with Binary, Multi-Class, and regression in order.

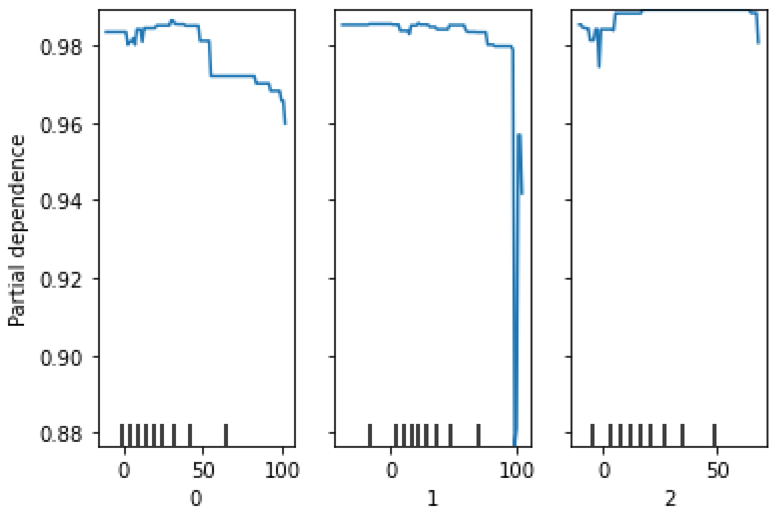


Figure 2: The Generalized PDP plot for the binary classification data set. This shows all values as incredibly important. Speaking that the values are relatively unique and the proportion of items in portfolio 1 to the rest is incredibly small

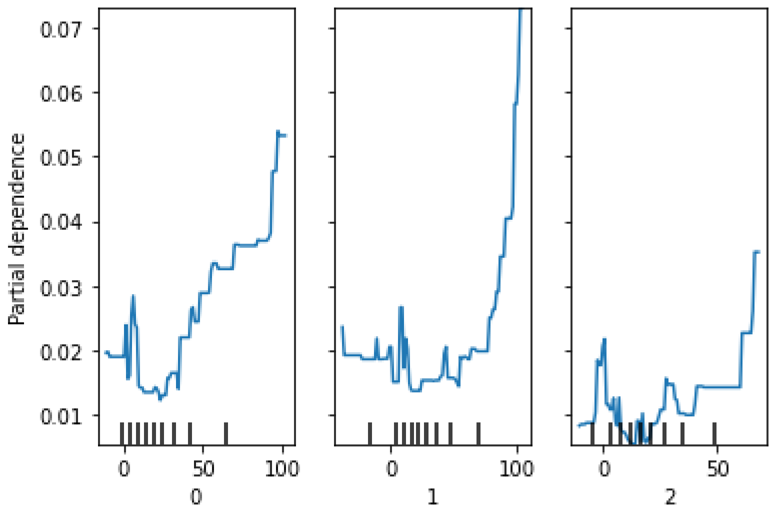


Figure 3: The Generalized PDP plot for the Multi-class classification set. This shows a better relationship in the classification of portfolio 1. As the values get incredibly large they are more likely to have an effect on portfolio 1.

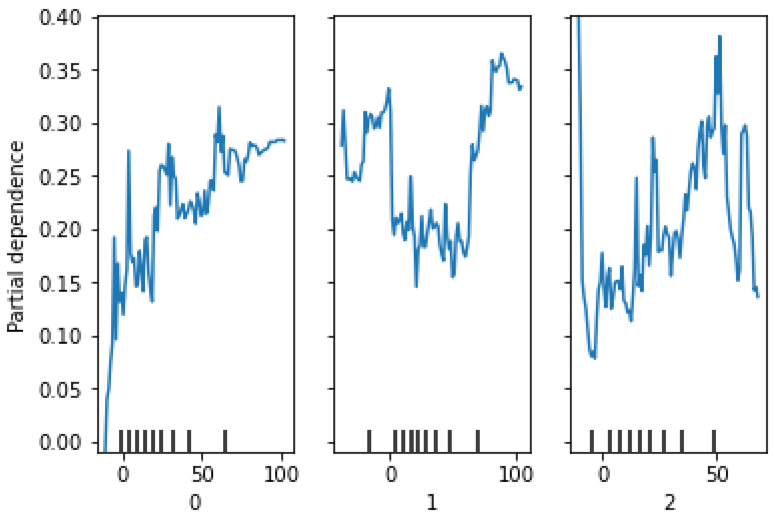


Figure 4: The Generalized PDP plot for the Regression data set. This shows that, again, larger values tend to have a larger impact on the model This might be due to the linear nature of random forest tree. Also, it is important to show that the EPS data has the most stable trend, due to EPS being an amazing determinant for stock performance.

PDP plot can also generate relationships between (up to) 2 different features in the hypothesis space. This is done by heat mapping. The following is the 2d relationship PDP map for EPS and Performance for the regressive data set.

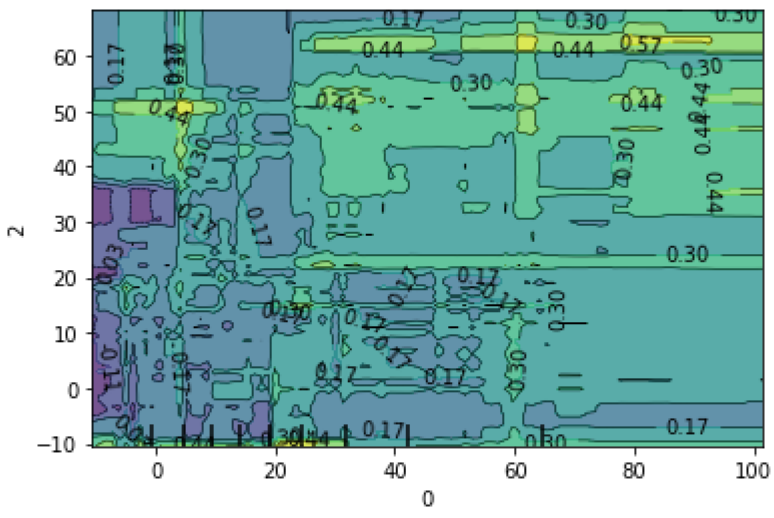


Figure 5: The 2D PDP plot for EPS [0] and Performance [2] features. Lighter values have higher effect on the mapping. For example, higher EPS and higher Performance is at .57 (bright yellow)

Permutation Importance

Permutation Importance does almost the opposite of PDP, as it asks the question "which feature values decrease the accuracy the most?" This then makes candle plots of which values vary the most in the decrease in accuracy, as well as the mean, minimum and maximum. I performed this with the stock data set to determine which features decreased the data set the most. The following was produced for the Binary, Multi-class, and regressive data sets respectively:

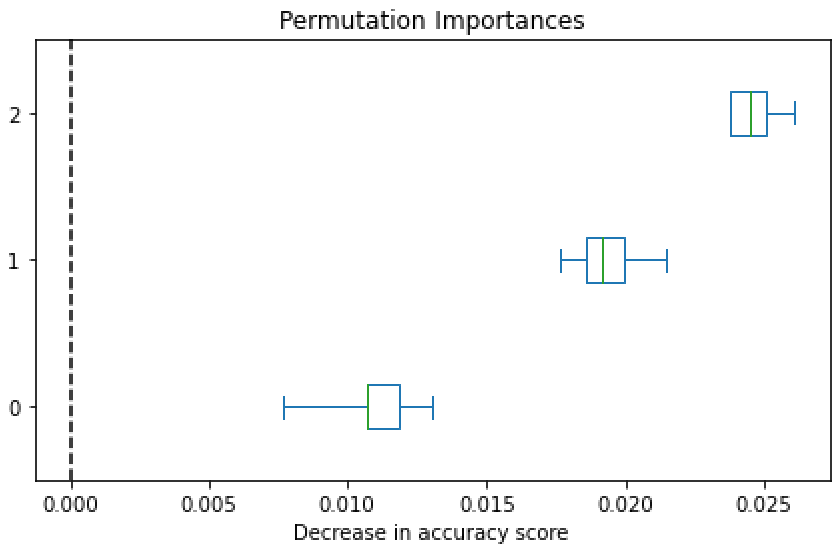


Figure 6: This is the box plot of the binary stock data set. This shows that the EPS feature had the least decrease in accuracy, meaning it is the best feature for accuracy. This also shows high variance in the BETA and Performance features in their effect on the model.

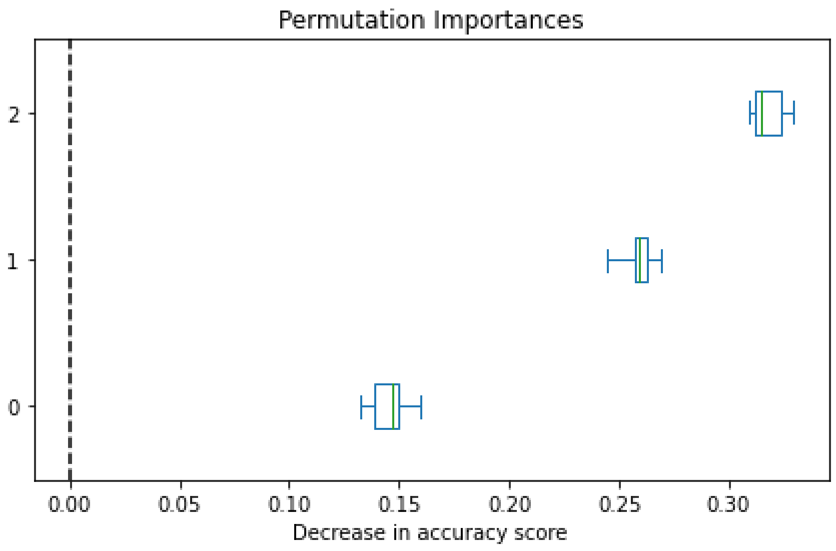


Figure 7: This is the box plot for the multi-class stock data set. This is a little worse than the binary data set, as the binary data set might be over-fit. Although, EPS is still the best feature value.

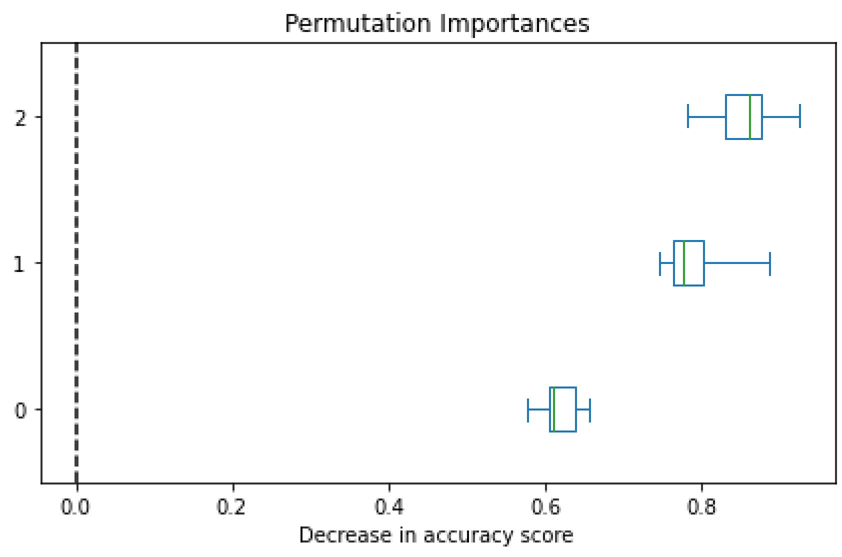


Figure 8: This is the box plot for the Regressive stock data set. This is by far the worst in decreasing accuracy due to the nature of regression being the sum of square errors. Still, the best feature is EPS

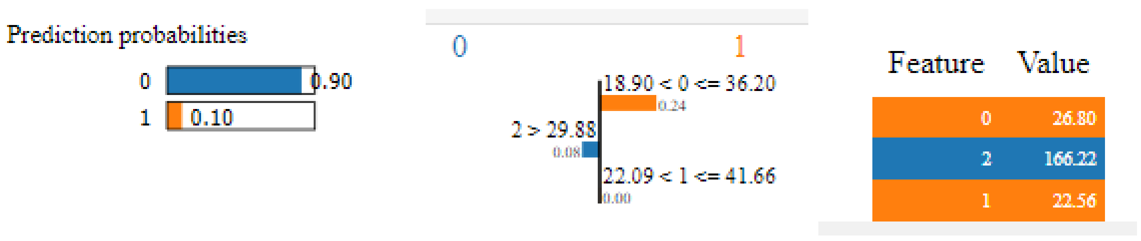
Local Models

Local models are models that take in a singular prediction and explain it by analyzing local areas in the hypothesis space. Many local models implement surrogate models for mapping features in later hypothesis spaces.

LIME

Locally Interpretable Model-agnostic Explanations [LIME] [3] is a methodology of which a prediction point (z) is explained by sampling uniformly around z in the hypothesis space by skewing variables of z, weighted by the likelihood these combinations would exist (π). The idea is the explanation is a local model of variable fidelity plus the complexity of an explanation, and to produce a good explanation, we must minimize the fidelity of a model and the complexity of an explanation. Most commonly, LIME's local model is Lasso-Regression, as this tends to be tunable to different weights on variables. So by sampling locally from a point in the hypothesis space we can produce trends in the model by some regression where the sampled points are X and the models prediction is Y.

For example, we ran this on the three stock data models. The binary prediction one displayed the following:



Feature	Value
0	26.80
2	166.22
1	22.56

Figure 9: LIME Output: prediction probabilities from sampled set of local regressive model (left), Weight of features on predictions (middle), feature value and individual classification (right).

As shown for this example, dilebrately chose due to its '0' classification and its high EPS, the feature value that effects a stock being classified as '0' is only 2: EPS. The other values are low enough that they weight more towards the do not buy classification, or '1'. We see that all local samples take, of course weighted by how close they are to the original in this case, are the same classification. I believe this to be due to the domain of high EPS. Next, we see that all feature values around '0' only effected the opposite result. For the regressive value we see the same, except it bounds by highest to lowest possible output of the lasso-regression surrogate model.

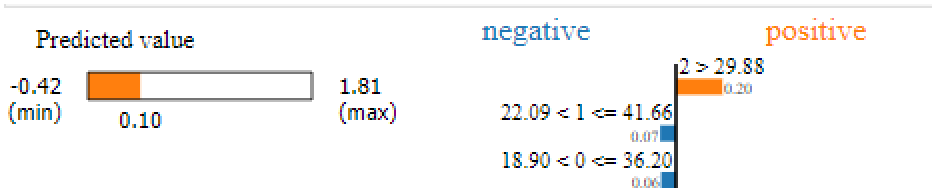


Figure 10: LIME Output: Regression predicted value

We see that the minimum is -0.42 and max is 1.81 (2018, again, was a good year for the market, as this is percent increase). We see that this model was positive at .1, and that the highest positive effect was the EPS while the rest actually supported a lower percent.

SHAP

Shapley values (SHAP) [4] of features state how much effect a feature has on the probability of an outcome. This is done by acting as if the predictions is a weighted sum of the features, and the weights are learned by local samples taken from the actual sample space and not randomly modified. This gives us a unique view of the data set's affect on the feature space. This is measured as a probability weight, but note that it is a log probability weight due to the python implementation. Applying SHAP to the Stock data, we get a slightly different story.

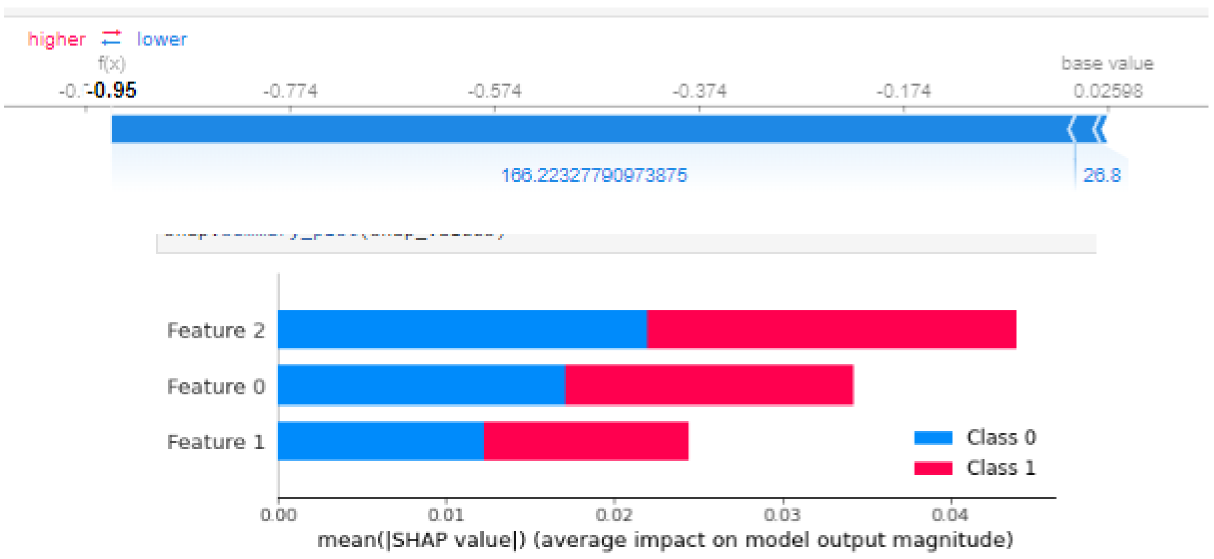


Figure 11: SHAP Output: Weight each feature's value has on the class '1' (top), class effect of each feature (bottom)

As we can see, since SHAP uses actual datapoints from the dataset instead of randomly skewing different features, we see that all values of the same datapoint (classified as buy in all models, or 2 in the multi-class model) has features that all point to class '0' or negatively effect class 1. We also see that all features are predominantly class 1, and the average impact is higher for feature 2. This does not mean higher feature 2 (EPS) values are more likely class 1, but rather have a higher effect on class 1 and 0. For the multi-class model, we see almost the same, except for when we see each values effect on class 2 (even though this is defined as class 1).

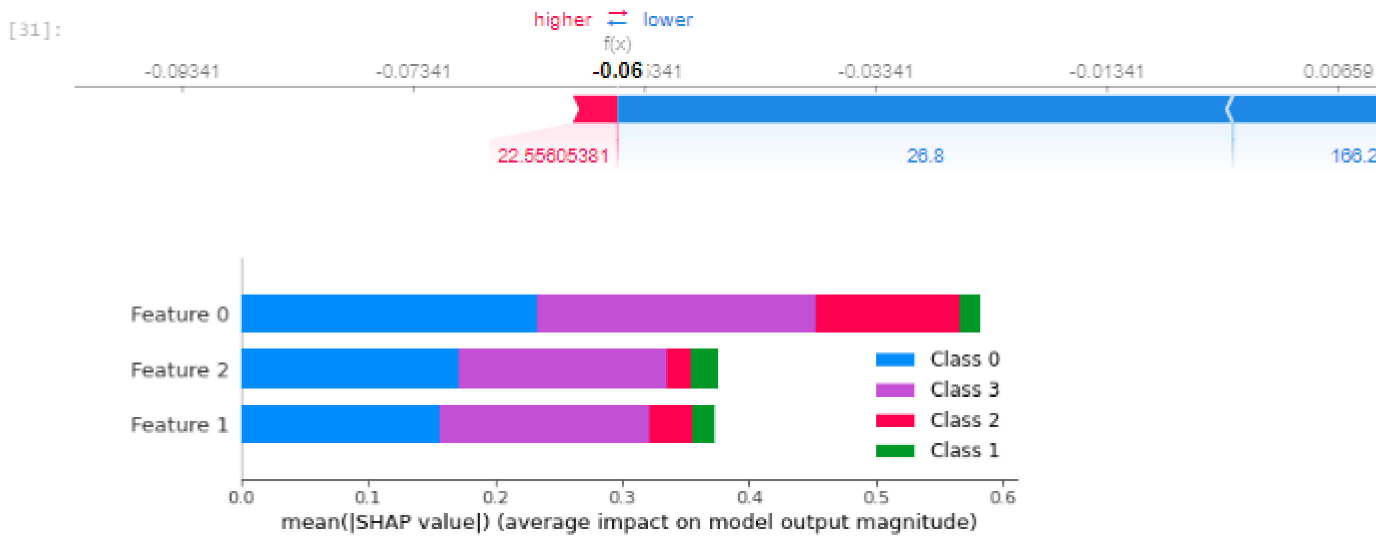


Figure 12: SHAP Output: Weight each feature's value has on the class '3' (top), class effect of each feature (bottom)

We can see that all have a negative effect on class 3, except 5 year percent increase. This gives us a pattern in the data set that maybe class 3 focuses more around higher percent increase while class 1 focuses more on eps. Regressive SHAP graphs are a little bit different, as there are two things to focus on: the feature value and the SHAP value.

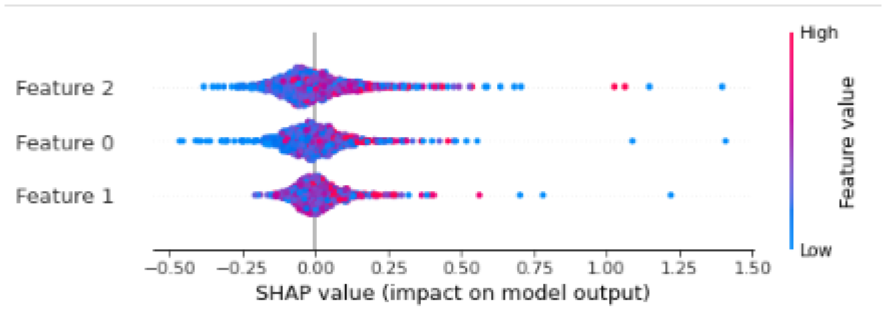


Figure 13: SHAP Output: SHAP value (x-axis) associated with feature value (scaled between max and min values, heatmap on right)

From this we see that on average, feature 2 had the largest (least centralized) effect, but surprisingly, the highest output is not defined by higher Feature 2 (EPS) outputs, based on the data set. This gives us a good perspective that on average higher EPS values are have higher affect, but when it comes to incredibly SHAP values, high feature values might not be the main indicator.

Model-Specific Explanations

Model Specific Explanations are explanation methodologies used on specific architectures (such as neural networks, time-series networks, CNN's, or reinforcement learning models). The reason these are important to study is that they provide explanations that take into account criterion of modelsfor the explanation.

Neural Networks

Here we talk about approaches for general neural networks. Examples may include specific implementations, but over-all these are used for basic NN's.

TCAV

Deep Neural Networks (DNN) are tricky to explain as they cover large domains and the hidden layers may not be able to be dissected as easily as shallow networks. For neural networks, a common approach is Testing Concept Activation Vectors (TCAV) [5]. TCAV is interesting because it compares concepts with predictions. This is commonly used with basic neural networks but a great thought experiment is visual networks such as CNN's. lets take a network that classifies cars vs. animals. "wheels" might be a concept within these images (a car or a person on a bike). The process of tcav is to train a surrogate model to associate these concepts and then see the association applied to an actual network. For example, wheels is a big part of a car, so it must have a higher feature value, therefore wheels must have a high association rate. We then can form concepts to global explanations for neural networks.

Adversarial Examples

Adversarial Examples [2] are simply skewing a prediction until an opposing result is formed. For example, if there is a regressive neural network, skew the neural network until the output either lowers or increases for each attribute. This allows a further understanding of the domain space and produces a local explanation.

Influential Instances/Deletion Diagnostic

This methodology plays on the training data for neural networks. This is a way of finding influential instances in the training data by removing an instance from training. This is obviously very heavy to calculate, but shows which instances changed the model the most.

FOL Logic

The FOL Logic method [6] is a methodology of training a surrogate model to produce first order logic mapping to the output of a trained, deep learning model. This paper talks about two ways, one is mapping each node in a neural network to an FOL logic statement and the next is mapping the whole network to an FOL statement. This is interesting as it is the only method directly requesting weights to calculate explanations that I have read.

Time Series

Here we talk about explanations used for Time Series or Temporal data. Many methods might be able to be reused, such as SHAP or LIME, mentioned in [7], but they must be altered. For LIME, it is not correct to alter data based on uniform information; we must first alter based on evidence given prior, that is, around the same mean as the change in a time step shown in the data. For SHAP, we have to not bias based on time period, but it is noted we can assign time-points importance as well as features for each time point. There are other methodologies, though. First is DeepLIFT [8]: a methodology for propagating differences in activation of a network to find explanations. This is done by viewing the differences in activation's when a sample is propagated through. The next is using attention mechanisms to map where attention quantifies particular inputs and outputs. This is useful for language and transformers and, although may not give stable explanations as a models outcome can be further defined by different networks, can be used for a variety of explanations [9].

Visual Explanations

Large data, such as images or videos, are where machine learning techniques thrive. Mainly, in computer vision. LIME can be used to produce visual explanations by skewing regions of an image and using that to map local feature importance. This is time consuming, as the image needs to be propagated and skewed, and not to mention a high-resolution image skewing factor must be large. Grad-Cam [10] is a methodology presented that provides a very elegant solution. This works as follows:

- *Forward Propagate*: Forward propagate any image

- *Back Propagate*: Back Propagate the gradient of all classes set to zero except the class in question to the last convolutions layer
- *Average*: for each pixel, average the propagated connections.
- *Apply ReLU*: Take only positive activates, set negative ones to zero.
- *Scale*: Scale between 0 and 1 and apply a heat map.

There are many variations of this, such as smoothing by skewing by a normal distribution of noise or by focusing or amplifying certain pixels, but the idea stays the same.

Attention Mapping

A final explanation methodology is mapping attention to a source of medium. This could be visual [11] or textual [12]. Visual explanations produced by Convolutional Block Attention Models (CBAM) produce explanations simply weighing the attention per pixel to produce an explanation. However, there is no reason to think attention cannot be further mapped to textual explanations, such as the "SEDA" paper [12], which maps attention produced by a propagation to some directional language vector. This can be added even further by using NLP word vectors and generating a domain-custom explanation.

References

[1] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253519308103

[2] C. Molnar, *Interpretable Machine Learning.* BookDown. [Online]. Available: https://christophm.github.io/interpretable-ml-book/index.html

[3] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016. [Online]. Available: http://arxiv.org/abs/1602.04938

[4] M. Sundararajan and A. Najmi, "The many shapley values for model explanation," *CoRR*, vol. abs/1908.08474, 2019. [Online]. Available: http://arxiv.org/abs/1908.08474

[5] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 2668–2677. [Online]. Available: https://proceedings.mlr.press/v80/kim18d.html

[6] G. Ciravegna, F. Giannini, M. Gori, M. Maggini, and S. Melacci, “Human-driven fol explanations of deep learning,” in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2234–2240, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/309

[7] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim, “Towards a rigorous evaluation of xai methods on time series,” in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 4197–4201.

[8] A. Shrikumar, P. Greenside, and A. Kundaje, “Learning important features through propagating activation differences,” *CoRR*, vol. abs/1704.02685, 2017. [Online]. Available: http://arxiv.org/abs/1704.02685

[9] I. Simic, V. Sabol, and E. E. Veas, “XAI methods for neural time series classification: A brief review,” *CoRR*, vol. abs/2108.08009, 2021. [Online]. Available: https://arxiv.org/abs/2108.08009

[10] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, “Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization,” *CoRR*, vol. abs/1610.02391, 2016. [Online]. Available: http://arxiv.org/abs/1610.02391

[11] S. Woo, J. Park, J. Lee, and I. S. Kweon, “CBAM: convolutional block attention module,” *CoRR*, vol. abs/1807.06521, 2018. [Online]. Available: http://arxiv.org/abs/1807.06521

[12] A. Stringer, B. Sun, Z. Hoyt, L. Schley, D. Hougen, and J. K. Antonio, “Seda: A self-explaining decision architecture implemented using deep learning for on-board command and control,” in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, pp. 1–10.