Explaining Predictions using Neural Additive Models

Sai Pradeep Peri
sap187@pitt.edu
University of Pittsburgh School of Computing and Information
Pittsburgh, PA, USA

Kinori Rosnow
ksr43@pitt.edu
University of Pittsburgh School of Computing and Information
Pittsburgh, PA, USA

ABSTRACT

As powerful function approximators, Deep Neural Networks are widely used in many real-world applications. But due to their complex structure, they are considered Black-box models which cannot be explained or interpreted on how they make predictions. While many existing methods use surrogate models to explain and interpret DNN predictions, few methods exist to explain predictions at a global view. As a result linear and other simpler models are employed due to their higher transparency, but often with a reduction in performance.

In an attempt to combine the best attributes of linear models and DNNs we explore Neural Additive models (NAMs). NAMs are a sub-family of General Additive Models suggested by Agarwal et. al. which leverage much of the performance of DNNs while maintaining inherent interpretability[1]. NAM's learn linear combinations of networks, each attending to a single feature. Interpreting NAMs is easy as the impact of a feature on the prediction does not rely on the other features and can be understood by visualizing its corresponding shape function (e.g., plotting fi(xi) vs. xi). These graphs learned by NAMs are not just an explanation but an exact description of how NAMs compute a prediction. This could help harness the expressivity of neural nets on high-stakes domains with intelligibility requirements.

Our primary focus for the project is to reproduce results of the NAM paper as well as we can with the provided information. The paper does not fully explain every hyperparameter and experiment so there will be some experimental decisions made that may deviate slightly from the paper. This motive for this project is to highlight the power of NAMs as explainable models with comparable performance to DNNs. Given enough time we also aim to address feature interactions, one of the major flaws of NAMs. This will be done by inputting combinations of features into neural networks which will allow the identification of the specific feature combinations which have high importance to the output.

KEYWORDS

neural additive models, general additive models, neural networks, explainable models

Fairness in ML Spring '21, April 27, 2021, Pittsburgh, PA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Sai Pradeep Peri and Kinori Rosnow. 2021. Explaining Predictions using Neural Additive Models. In *Proceedings of Fairness in ML Spring '21.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As society adopts automated decision making, the importance of understanding the models behind the decisions grows. Transparency and fairness in automation is more than just a public demand, but a necessary consideration when designing machine learning models. The ability to interpret a model can prevent problematic models from harming people in sensitive applications in addition to improve understanding of the problem. This required interpretability drives the societal need for models whose predictions can be explained.

While model performance has progressed by increased complexity, their added complexity has reduced explainability in many cases. Many recent advances in machine learning performance have come from deep neural networks (DNNs), which are highly expressive and flexible models. While being known for their ability to solve a variety of previously intractable problems, they have also been labeled "black box" models due to their uninterpretable nature. Their depth and complex network of connections account to their high-performance, but it shrouds our understanding of model decision base. Methods like Locally Interpretable Model-agnostic Explanations (LIME)[8] and Shapley additive explanations (SHAP)[6] provide only a local view at feature importance which doesn't explain how the feature affected the prediction outcome. Using surrogate explainable models trained on the predictions of a more opaque model is another attempt to improve interpretability, but these are also approximations of the model and do not necessarily capture the truth of the black-box model.

Designing models that are interpretable is currently the best way of understanding model predictions with respect to the input features. Generalized additive models, or GAMs, are class of models with inherent interpretability due to their linear combination of functions each taking in a single input feature. Understanding the output of the individual feature functions provides direct and complete information of how the feature impacted the prediction outcome. Neural additive models (NAMs) proposed by Agarwal et. al. are GAMs with DNNs for feature functions[1]. NAMs combine much of the performance benefits of DNNs with the inherent interpretability of GAMs. Agarwal et. al. performance of NAMs was comparable to DNNs for many problems and in doing so, established them as an interpretable alternative to DNNs.

For this work, we reproduce some of the experiments done by Agarwal et. al. and in doing so provide evidence that NAMs are an interpretable alternative to DNNs. We aim to highlight that NAMs should be considered when designing and experimenting with solutions to a given machine learning problem. This is especially true when the model will be affecting sensitive outcomes and may need to be explained. We also discuss the feature independence characteristic as one of the weaknesses of NAMs despite it being what provides interpretability. Further, we propose that involving feature interaction terms into the linear combination producing the model output can help address some of this weakness while not compromising interpretability. This proposed solution does begin to reduce interpretability if extended to more than 2 features as inputs in the feature functions.

In Section 2 we discuss NAMs in detail. Experimental set up is described in Section 3 and the results of those experiments are described in Section 4. We describe a limitation of NAMs and propose a way to address it in Section 5. Finally we conclude with final remarks in Section 6.

2 NAMS

NAMs are a sub family of Generalized Additive Models (GAMs)[4]. General additive models are a linear combination of functions that are dependent on individual input features. The mathematical form of GAMs is given as[4]:

$$g[E(y)] = \beta + f_1(x_1) + f_2(x_2) + ... + f_k(x_k) \tag{1}$$

Generally, low-order smoothing splines are used to fit the predictor functions analytically. In recent years the state-of-the-art GAMs were fitted using boosted trees to learn more "jumpy" and complex feature trends called Explainable boosting Machines [2]. In this project, we explored Neural Additive Models (NAMs), which use Deep Neural Networks (DNNs) to fit the predictor functions of GAMs. Each feature dependent function in the linear combination will be an independent DNN with an individual feature as input. The architecture is visualized in figure 1.

In figure 1, the $x = [x_1, x_2, \ldots, x_k]$ is the input with k features, and y is the target variable. NAMs are linear combinations of DNNs where each network attends to individual input features. The feature network outputs are then summed and can be passed through a sigmoid or other squash function for classification. The entire network together is trained using backpropagation and can be easily scaled as they can be trained on the GPUs. Modern deep learning libraries such as Pytorch make implementation easy and handle the backpropagation training. The architecture of NAMs is designed around interpretability while attempting to get comparable performance to DNNs. The introductory work of NAMs has shown their performance to be comparable to that of DNNs.

2.1 Interpretability

Neural Additive Models are inherently interpretable and intelligible due to their linear nature. In NAMs, Neural Networks each attend to individual predictor attributes, which makes each features independent of the each other, resulting in a highly interpretable models. The linear combination of feature independent models allows for inspection of how each input feature affects the linear combination and therefore the output. Each prediction outcome can be checked with respect to the input features by inspecting the output of the network that attended to the feature.

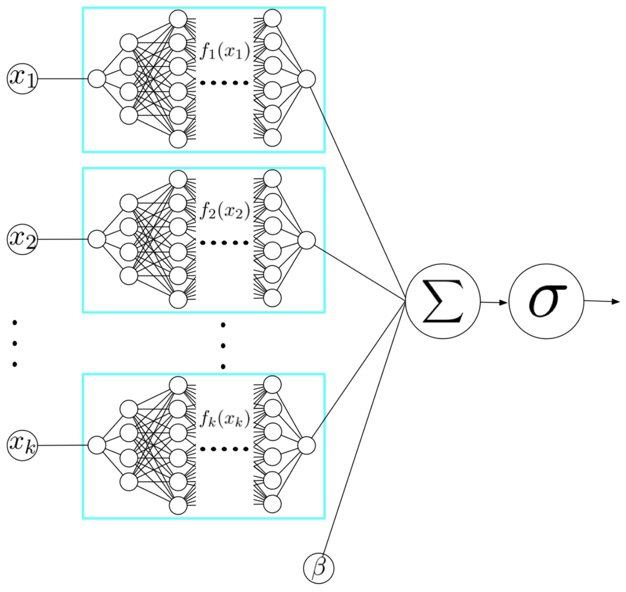

Figure 1: Architecture of Neural Additive Model for Binary Classification. via Agarwal et. al [1].

Taking the above inspection further, completely interpreting NAMs is easy as the predictions can be easily understood by visualizing the corresponding feature shape function i.e., plotting $f(x_i)$ vs. x_i. These graphs not only provide the explanations of the predictions but also the exact description of how each feature network affects the predictions. The combination of all of the visualizations is complete information of every possible prediction within the space of the plots (dependent on the domain of each x_i).

2.2 Implementation Nuance

There is an implementation nuance that we found in the paper's implementation of NAMs. One was the expected form from equation (1). This involves a direct sum of the feature function outputs. The other form was passing the outputs as an input vector to a linear combination layer. In Pytorch, this would result in the mathematical form:

$$g[E(y)] = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + ... + \beta_k f_k(x_k) \tag{2}$$

The potential mistakes when interpreting this model comes from assuming the β_i or $f_i(x_i)$ individually carries information. This is not true because without knowledge of $f_i(x_i)$ we cannot know how the β_i affect the prediction and vice versa. In this case $\beta_i f_i(x_i)$ must be treated as $f_i(x_i)$ in equation (1) to get the actual affect of the feature function on the outcome.

2.3 ExU Activations:

Additive models require fitting jagged or highly jumpy functions as seen in real-world datasets. As each feature is parameterized individually to fit the target variable through a univariate function, we don't want to miss genuine details in the real data.

Even though Neural networks with ReLU activations have achieved high-performance results they are not always beneficial to use in the case of NAMs. The ReLU activations struggle to fit to the data when trained with mini-batch training and model smooth functions [7]. The NAMs paper of Agarwal et. al [1] showcases this issue by fitting an empirical toy example based on Bernoulli random variables as shown in Figure 2a.

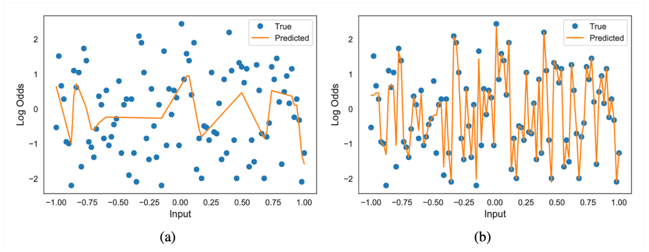

Figure 2: Predictions learned using Neural Networks with (a) ReLU activations, (b) ReLU with Exp-Centered Units . via Agarwal et. al [1].

To overcome the problem of NN's failing to model highly jumpy functions, the authors propose Exp-centered Units (ExU's). For a given scalar input x, the activation function is given by

$$h(x) = f(e^{w} * (x - b)) \tag{3}$$

where w represents weights and b represents bias terms. The ExU units are logarithmic in nature; a small change in the inputs results in a steep value of jumpy output values. Figure 2b shows how NN's with ExU units capped with ReLU-n fits better than plain ReLU activations [5].

Due to their highly flexible nature, overfitting can be a concern when using ExUs. We avoided overfitting our models containing ExU units using the following techniques:

- Dropout
- Weight Decay
- Output L2 Penalty
- Feature Dropout

2.4 Training

The training of Neural Additive models is done based on the loss function below

$$L(\theta) = E_{x,y\ D}[l(x, y; \theta) + \lambda_1 \eta(x; \theta)] + \lambda_2 \gamma(\theta) \tag{4}$$

where $D = (x^{(i)}, y^{(y)})_{i=1}^{N}$ is the training set of size N,
$\eta(x;\theta) = \frac{1}{K}\sum_x \sum_k (f_k^{\theta}(x_k))$ is the output penalty,
$\gamma(\theta)$ is the weight decay, f_k^{θ} is the feature network for the k^{th} feature

$l(x, y; \theta)$ is the loss function specific to task. For binary classification it represents the cross-entropy loss given as,

$$l(x, y; \theta) = -y log(p_{\theta}(x)) - (1 - y) log(1 - p_{\theta}(x)) \tag{5}$$

In this project we use the binary cross-entropy loss for a classification problem on the COMPAS dataset [3, 9].

For our regression task we used the CA housing dataset. For our regression task the loss function represents Mean-squared error,

$$l(x, y; \theta) = (\beta^{\theta} + \sum_{k=1}^{K} f_k^{\theta}(x_k) - y)^2 \tag{6}$$

Datasets and experimental set up are discussed further in the following Section 3.

3 EXPERIMENTAL SET UP

3.1 Datasets

Five different datasets are used by Agarwal et al. to demonstrate the functional applications of NAMs. But due to the constraints in time and computational resources, we showcase our reproduced analysis of NAMs on two datasets, one classification task and one regression task.

- **COMPAS:** Risk Prediction of the Criminal Justice.
- **California Housing:** Predicting Housing Prices.

The focus for the COMPAS dataset is to predict the risk of recidivism for court reappearances. Deeming a person high risk will mean the person will be held in jail. Low risk classification means the person is free to go home until their following court appearance. This decision has massive repercussions on a person's life. Many activist demand fairness and explainability of automated risk classification of defendants due to the high impact potential of such a decision.

The California housing prices dataset problem is to guess the actual median value of a house. This sort of problem has large financial implications because if a company can properly identify a house's value using an automated system, it may target currently undervalued houses as an investment strategy. Understanding such an algorithm can lead to improved investment strategies and possible alternative business models.

3.2 Baseline Models

To showcase the performance of NAMs, we compare the NAMs with the following baseline models.

- Logistic/Linear Regression.
- Decision Trees.
- XGBoost.
- Explainable Boosting Machines.
- Deep Neural Nets.

Logistic/Linear Regression and Decision Trees are known basic models that can easily be explained, but often not the top performing models. XGBoost and Deep Neural Networks (DNNs) are higher performing models, but compromise some explainability. Explainable Boosting Machines (EBMs) are the current state-of-the-art GAMs. Comparing to all of these models spans the potential combinations of model performance and interpretability.

4 RESULTS

4.1 COMPAS: Risk Prediction (Classification)

The COMPAS is used to predict the recidivism risk and used for sentencing and early parole decision making. This is a very high-stake task and has been subject to scrutiny for being biased racially[3, 9].

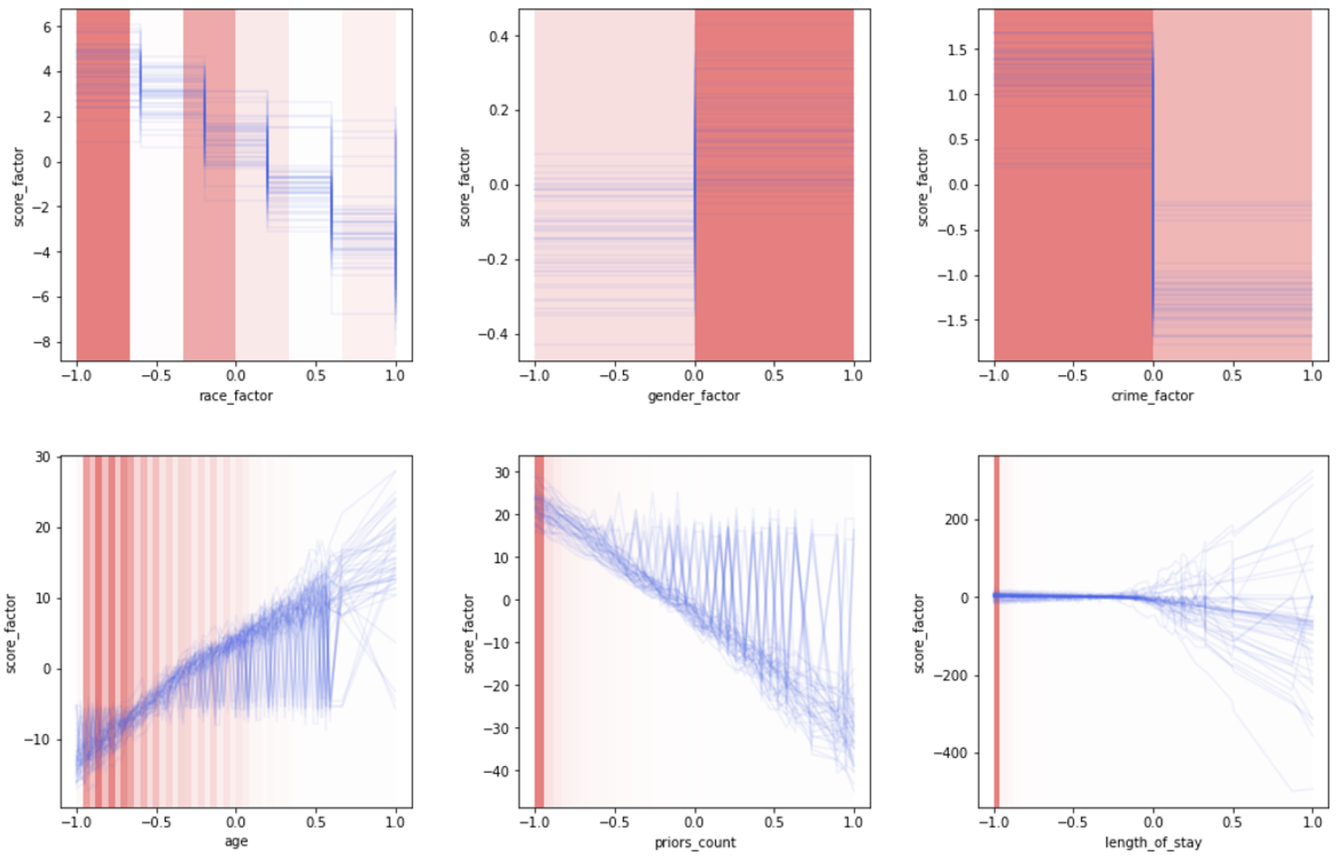

Figure 3: COMPAS graph explanations: Plots showing the feature shape functions learned by ensemble of NAMs along with the density of the data.

We want to use the NAMs model for showcasing its explanatory power for explaining the model prediction and expose any biases involved in the model.

We are using the recidivism data released by ProPublica on defendants in Broward County, Florida. We are using a total of 6 features, including race, gender, etc., to predict the risk score.

Table 1: COMPAS model comparison

Model Name	AUC Score
Logistic Regression	0.75
Decision Trees	0.73
XGBoost	0.75
Explainable Boositng Machines(EBMs)	0.76
Deep Neural Nets	0.74
NAMs	0.72

Results Comparison: We have trained the NAM feature networks jointly using the Adam optimizer with a learning rate of 0.0208 and a batch size of 1024. An ensemble of 50 models are trained each for 35 epochs with a step_lr factor of 0.1 and Cross-Entropy loss. We haven't used the 5fold cross-validation method as done by Agarwal et al. but validated with a 1fold validation set. The AUC score reported is the average across the 50 ensemble models.

From Table 1, we can see that even though the NAMs haven't performed as better as other interpretable Models, we can see NAMs are closer in performance to Deep Neural Network models. As computing resources limit us, we haven't tried out the grid of hyper-parameter tuning and larger training epochs, which might have caused a drop in performance compared to Agarwal et al.

Explaining Graphs: Figure 3 shows the individual shape function learned by an ensemble of 50 NAM models. The redness indicates the data density at that location the darker the bar the more data is present at that location and the blue lines show the trend of the learned feature shape functions. The main factor of using this dataset is to showcase the biases involved in the dataset through the transparent nature of NAMs.

Figure 4 shows the zoomed version of just race and gender features. Here we can observe that the risk score is higher for African American defendants compared to Asian or white defendants. This shows that NAMs trained on the COMPAS dataset are racially biased. The shape feature function of gender suggests that Male defendants are predicted to have higher risk scores than female defendants suggesting a gender bias. The flexible feature of the Neural

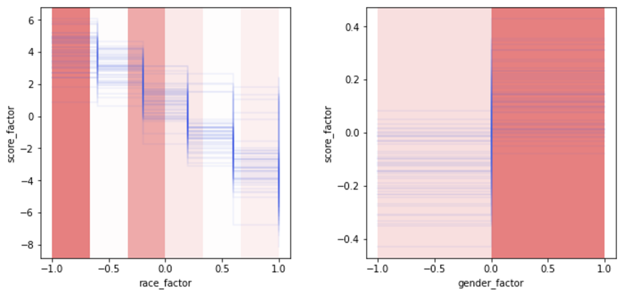

Figure 4: COMPAS graph explanations Zoomed: Plots showing the Race and Gender feature shape functions. The mapping of the labels to bars positions are Race: 'African-American': 0, 'Asian': 1, 'Caucasian': 2, 'Hispanic': 3,'Native American': 4, 'Other': 5 and Gender: 'Female': 0, 'Male': 1

Additive Model is once we identify the biased feature, we can easily correct the model by removing the discriminating feature.

4.2 California Housing Price Prediction (Regression)

We are using the California housing price dataset for the regression task to predict the median housing prices of house on the scale of millions.

Results: The NAM feature networks are trained using Adam Optimizer with a default learning rate and a batch size of 1024. We have used ExU activation units with hidden sizes of [64,64,32] and a low output penalty of 0.001. A total of 5 ensemble models are trained each for 50 epochs with Mean Square Error loss. The average MSE score based on the optimized parameters is 0.43, with the median house price being on the scale of millions.

Table 2: California Housing Price Prediction model comparison

Model Name	Mean Square Error
Linear Regression	0.53
Decision Trees	0.52
XGBoost	0.29
Explainable Boositng Machines(EBMs)	0.25
Deep Neural Nets	0.46
NAMs	0.43

Table 2 shows the comparison of the results between NAMs and the baseline models based on the Mean Square Error metric. For the task of housing price prediction, we can see that NAMs outperform the linear regression, Decision trees, and Neural Networks with an MSE value of 0.43. The best performance is shown by Explainable Boosting Machines (EBM's), a GAM-based model with predictor functions modeled using boosted trees. The results showcase that NAMs can perform as better as Deep Neural Networks and are as highly interpretable as Linear Models.

Visualization: Figure 5 explains the NAMs shape functions learned on the California Housing price prediction dataset leaned on an ensemble of 5 NAM models. The results show that the housing price increases as the median income increases. The median house price and the latitude and longitude seem like important features based on the score factor.

5 LIMITATIONS

Although the linear combination of independent feature DNNs results in an easily interpretable model, the single feature DNNs compromise on one of the major strengths of DNNs, feature interaction. For many use-cases, feature interactions boost DNN performance, but these interactions are not interpretable when the network is sufficiently complex. We propose feature interaction DNNs that take in multiple inputs. There is no loss of interpretability for two feature interaction terms as we can still visualize the output of the feature network with a 3-dimensional surface. We can adapt equation (1) to accommodate 2 feature interaction terms as shown in (7) and more.

$$g[E(y)] = \beta + f_1(x_1) + f_2(x_2) + ... + f_k(x_k) + f_k + 1(x_1, x_2) + ... \quad (7)$$

However, if we consider more than 2 features in our interaction terms, we will no longer be able to easily visualize the feature networks with respect to all of the input features. Visualizations will need to be level sets, cross-sections, subspaces, or any other method of visualizing $\mathbb{R}^N$ spaces where N > 3. The other method would be localized, instance-based explanations where we take data instances and see the particular output of the feature network. As more features are considered in the feature interaction networks, the interpretability decreases.

Another consideration must be made to the computational complexity of the model when including feature interaction terms. For no interaction, the number of terms in the linear combination is linear with the number of features $O(k)$ where k is the number of features. If 2 features are considered in interaction terms, then the number of terms will grow on the order of $O(k + k\mathrm{C}2)$ where C is the combinations. Finally, to generalize this to n features for interaction, if interaction terms are not omitted, the number of terms will be $O(\sum_i^n k\mathrm{C}i)$. This can increase computational complexity to exceed certain equally performing DNNs, but this depends on many variables such as feature network architecture.

Feature interaction terms are a possible solution to improving model performance while maintaining much the interpretability. However, the larger the number of features used in the interaction, the less interpretable the model. In addition, the NAM model may grow to be larger than an equally performant DNN. A 2 feature interaction NAM seems like the logical next iteration of improvement as it also does not reduce interpretability. We are in the process of implementing 2 feature interaction NAMs; however, due to time constraints did not have them ready by the end of the semester.

6 CONCLUSION

For this project, we aimed to establish NAMs as an interpretable alternative to DNNs.

This work has reproduced comparable results to the original work from Agarwal et al. [1]. The tests spanned one classification (COMPAS) and one regression (CA housing prices) problem, and 6 model types (Linear/Logistic Regression, Decision Trees, XGBoost,

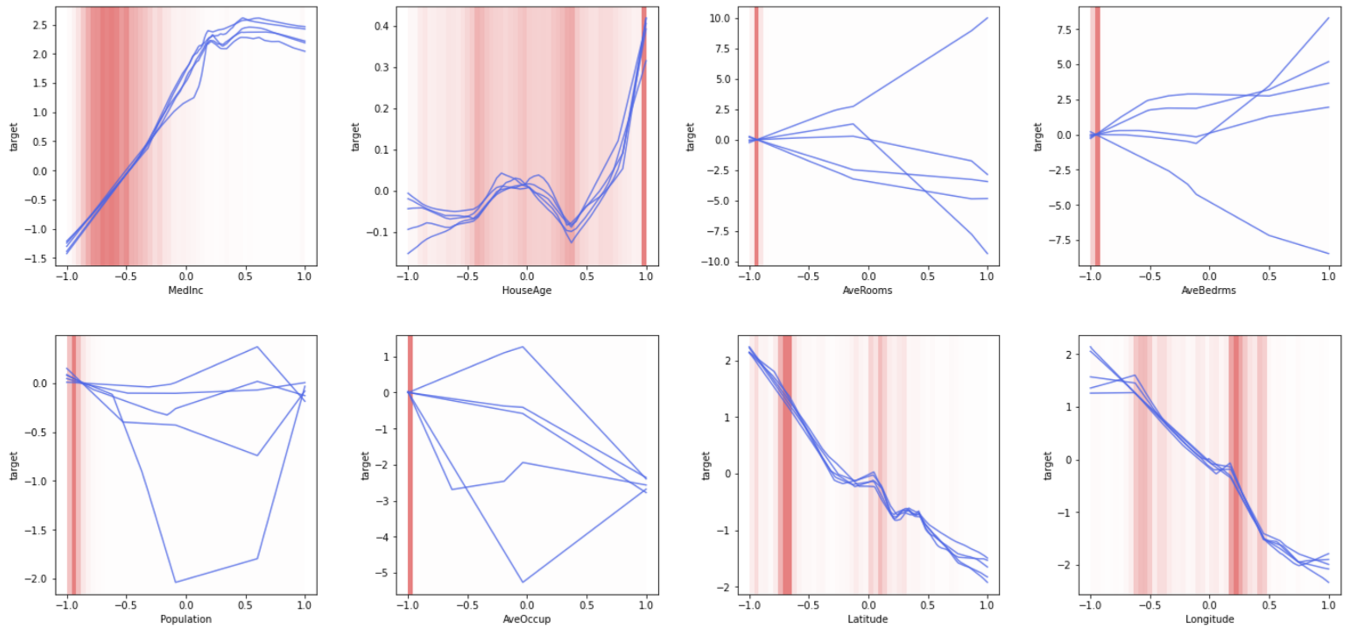

Figure 5: California Housing Price prediction NAM model feature explanation plots. Each feature shape function is plotted based on ensemble of NAM models. The blue lines indicate the function trend of individual features and the redness indicate the data density at that location.

EBMs, DNNs, and NAMs) for comparison. We established the relative performance between NAMs and DNNs but did not reproduce exact results due to time and resource constraints. Proper tuning and experimentation with NAMs and DNNs similar to the original paper would have required thousands of epochs. We handled this by tuning the NAMs and DNNs similarly across both datasets.

Due to their linear nature, NAMs have global interpretability on the outcome of their predictions with respect to the inputs. Feature network outputs can be easily calculated, and in turn, simple visualizations of feature network output with respect to features can be created. NAMs' design allows for explainability without surrogate models or other local, approximate techniques.

Although independent feature networks serve as a point of strength for interpretability, this is probably where DNNs may outperform NAMs for certain problems. DNNs are not only performant due to depth but also feature interactions. In order to address this, we propose feature interaction terms (Section 5: Limitations) that may improve NAM performance. Interpretability is only reduced if more than two features are considered for interaction. This means two feature interaction networks can be used with no loss of interpretability. Our implementation of 2 feature interaction NAMs will not be complete by the close of the academic year. This project can be taken further to explore the potential improvements of feature interaction NAMs.

Due to their comparable performance to DNNs, NAMs are a natural alternative to DNNs. As the field of AI progresses, and automated decision-making becomes more ubiquitous, the need for explainable decision outcomes grows. A performant AI solution that cannot be completely audited has the potential to cause harm in inexplicable ways. As a result, the cause of the harm may not be able to be addressed and may be perpetuated. Finding interpretable alternatives to high-performance black-box models is necessary for fairness and the future of AI technology.

ACKNOWLEDGMENTS

Thank you Professor Yu-Ru Lin for your teaching and guidance throughout the semester. Thank you to our supportive classmates for your stimulating discussion, through which we have learned so much.

REFERENCES

[1] Rishabh Agarwal, Nicholas Frosst, Xuezhou Zhang, Rich Caruana, and Geoffrey E. Hinton. 2020. Neural Additive Models: Interpretable Machine Learning with Neural Nets. 1 (April 2020). https://arxiv.org/pdf/2004.13912.pdf
[2] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noémie Elhadad. [n.d.]. Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission.
[3] Julia Dressel and Hany Farid. 2021. The Dangers of Risk Prediction in the Criminal Justice System. *MIT Case Studies in Social and Ethical Responsibilities of Computing* (5 2 2021). https://doi.org/10.21428/2c646de5.f5896f9f https://mit-serc.pubpub.org/pub/risk-prediction-in-cj.
[4] T.J. Hastie and R.J. Tibshirani. 2017. *Generalized Additive Models.* 1–335 pages. https://doi.org/10.1201/9780203753781
[5] Alex Krizhevsky. 2010. Convolutional deep belief networks on cifar-10.
[6] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
[7] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the Spectral Bias of Neural Networks. arXiv:1806.08734 [stat.ML]
[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*

(San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[9] S. Tan, R. Caruana, G. Hooker, and Yin Lou. 2018. Distill-and-Compare: Auditing Black-Box Models Using Transparent Model Distillation. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (2018).