Explanable Machine Learning

D S Abhishek (160218)
dsabhi@iitk.ac.in
Harshit Nyati (160282)
hnyati@iitk.ac.in
Ritesh Kumar (160575)
riteshk@iitk.ac.in
Saumya Shah (160633)
sshah@iitk.ac.in
Suryateja B.V. (160729)
suryab@iitk.ac.in
Vipul Bajaj (160794)
vipulbjj@iitk.ac.in

1 Disclaimer

The work carried out in this project has not been re-used from any another course project at IIT Kanpur or elsewhere, or any other project done elsewhere.

2 Motivation

> "If AI cant do better than us at explaining what its doing, then dont trust it."
> - Daniel Dennett, Renowned Philosopher

Machine Learning is widely used today in fields ranging from classification of dog breeds, to predicting important policy decisions in health care. However, most models are seen as *black boxes* - we don't know why the models are behaving in a particular way [2] [5]. Google, seen as the forerunner in AI, had to remove *gorilla* label as their Photos app tagged two black people as gorillas. [13] Hence, it is of paramount importance that we *open* these black boxes, and make the models interpretable and accountable. Via explaining machine learning, we can help humans improve their models and make more informed decisions instead of relying solely on a model's predictions. Every model learns certain parameters like weights and biases, and it extracts useful features out of data. But these features and weights are just long vectors which provide no intuitive understanding to humans. Also, deep networks have millions of weights at play. So, just presenting a large number of learned weights would not *explain* the model. Thus, our task is to derive simple explainable models from a complex model, which humans can *interpret* and use to differentiate the predictions. [11]

3 Literature Review

3.1 LIME - Local Interpretable Model-agnostic Explanations

Deciphering it's name, its aim is to give explanations that are Model-agnostic(independent of the model) which means it considers model as a black-box. The first two words of the full form of the acronym LIME stand for Interpretable - It should provide a reasonable interpretation to classifier's prediction and not bombard the user with hundreds of features(if present) and Local which means that the model should at least be locally faithful.

If the users do not trust a model or a prediction, they will not use it. This paper tries to explain the predictions of any classifier in an interpretable and faithful manner. It does so by learning an interpretable model locally around a prediction.

In certain applications of machine learning, predictions cannot be acted upon blindly, such as medical diagnosis and terrorism detection, as the consequences may be fatal. The conventional evaluation metrics like accuracy on a hand-held validation dataset are significantly concealing about the true strength of the classifier. As real-world data is often quite different, and also the evaluation metric may not suit the personal needs of the product's goal. What does "explaining a prediction" mean? It means to present textual or visual artifacts that help us comprehend the

relationship between the instance's components and the model's prediction. Practically, a person may wish to deploy a less accurate model for certain tasks like content recommendation that has a reasonable accuracy but does not give high importance to "click bait" articles [8] as doing that might increase the accuracy in cross-validation but is not good for user retention. Most often, for explaining models, we need to give a modified form of input(interpretable data representations) to our explaner model as raw features may not be very interpretable for common public.

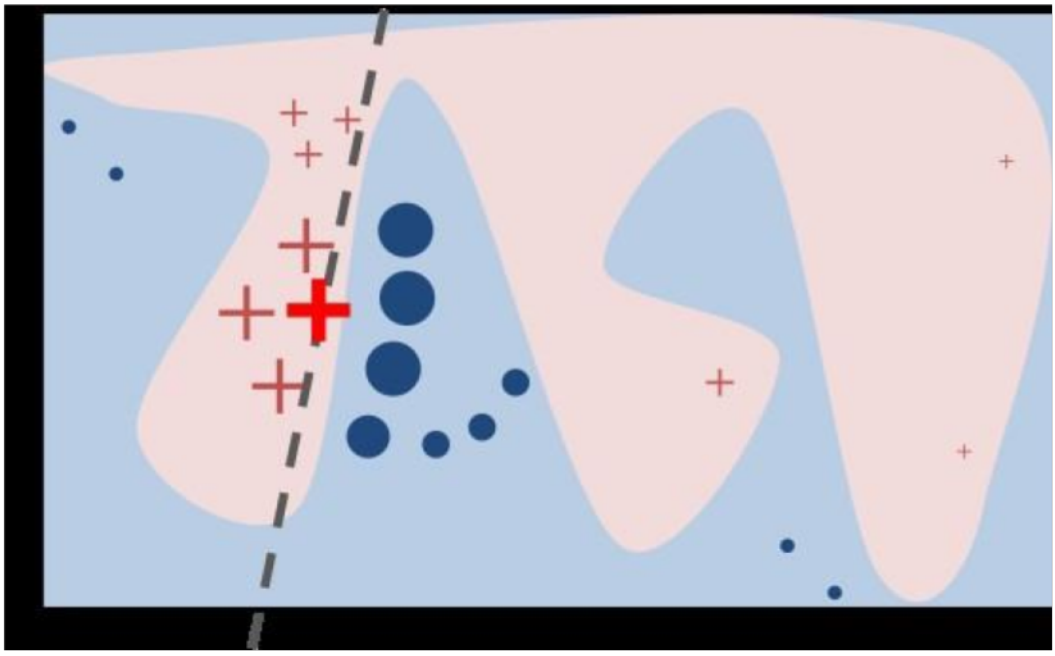


Figure 1: The black-box models complex decision function f (unknown to LIME) is represented by the blue/pink background. The bright bold red cross is the instance being explained.LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained(represented here by size).The dashed line is the learned explanation that is locally (but not globally) faithful.

3.2 Maths behind LIME

LIME can be formulated into an optimization problem. Let us define the notations first. $\Omega(g)$ is defined as the a measure of complexity. (f, g, π_x) is the measure of how unfaithful(not faithful enough) g is in estimating f in the neighbourhood π_x. [8]

$$\xi(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

$z' \in \{0,1\}^{d'}$ is a perturbed example (a fraction of elements of x' that are positive are present in z'). f(z) is used as the label to the model gotten from the retrieved example in original representation $z \in R^d$ [8]

$$L(f, g, \pi_x) = \sum_{z, z' \in Z} \pi_x(z)(f(z) - g(z'))^2 \text{ where } \pi_x(z) = exp(\frac{D(x, z)^2}{\sigma^2})$$

Linear models G are such that $g(z') = w_g z'$. The loss function weighted in the neighbourhood(locally) L is defined by the equation above. Also in the above equation π_x is an exponential kernel.

Summarizing LIME - LIME can approximate any black-box model locally in the neighborhood of the prediction we are interested. [8]. Suppose we want to know why an input x gave a prediction f(x), where f is a complex model (black box). First, we build interpretable data representations of our dataset. LIME sees what happens to the complex models predictions when we send in perturbed samples of our new data. It creates a dataset of our new samples and their

corresponding predictions, on which it learns a simple model (decision trees, linear model) weighted by the proximity of the new samples to x. Notably, the learned model only explains local phenomena in vicinity of x.

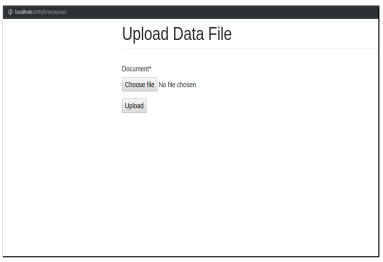
3.3 aLIME

Anchor Local Interpretable Model-Agnostic Explanations, also known as aLIME, is an improvement/extension of LIME model explained above. It is a system that explains individual predictions with if-then rules in a model-agnostic manner. Such rules are intuitive to humans, and usually require low effort to comprehend and apply. In particular, an aLIME explanation (or an anchor) is a rule that sufficiently anchors a prediction such that changes to the rest of the instance do not matter (with high probability). This technique is more intuitive, has high precision, and very clear coverage boundaries. [9]

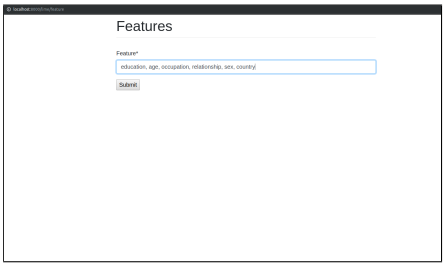
4 Web-App

We built a web application through which people can explain the predictions of various classifiers on different datasets using LIME as well as aLIME. They needn't have any expertise in programming or machine learning. We've made the entire cumbersome process end to end. The user just needs to upload the dataset in a specified standard format. Then he needs to type in various features and class labels that the dataset comprises of. After that he can select various parameters like the explainer, classifier, number of top features, etc. Some sample screenshots of the process are shown in the figure below. The outputs have been shown in the Results section below. The link to project page is this

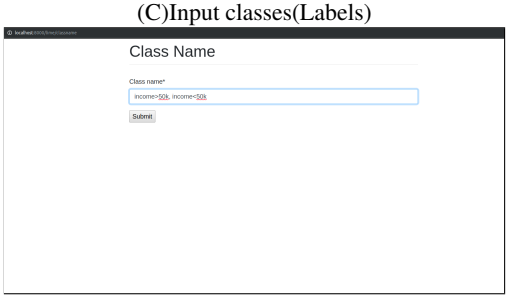
(a)Dataset



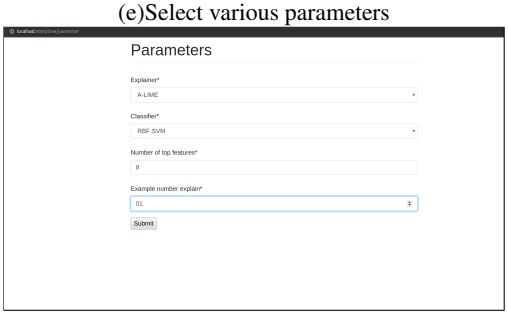
(b) Input Features



(C)Input classes(Labels)



(e)Select various parameters



(f) Different classifiers available

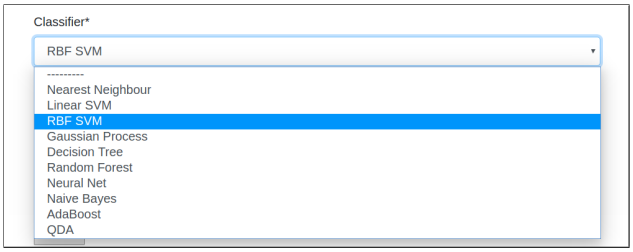


Figure 2: The flow of the web-app for generating explanation

5 Interpretation and Implementation of LIME and ALIME

We implemented LIME and ALIME on multiple datasets which gave some interesting insights on the application of various classifiers. We mainly analyzed the following datasets for LIME and ALIME. We chose these datasets specifically because the explanations on these are intuitively easy to understand and to check whether the classifier chooses the right features to classify.

- Zoo Dataset: This dataset contains 17 features for each of the 101 animals. The dataset can be found here. There are 7 classes of animals namely - Land Animals, Birds, Reptiles, Fishes, Amphibians, Insects, Sea Creatures.
- Heart Disease Dataset: This dataset contains 76 features of which we have used 14 features for each of the 303 patients. The dataset can be found here(Cleveland dataset). The label specifies the level of the heart disease. It is an integer value ranging from 0 (indicating no presence) to 4.
- Adult Dataset: This dataset contains the general details like sex, education, relationship and 11 other features of 48842 persons. the dataset can be found here. here. We predict whether a person earns more than 50K per year.
- Lending Loan Prediction task is to determine whether a person makes over 50K a year. Dataset: This dataset contains 75 features for 890,000 instances of issuance of loan from 2007-2015, containing status of the various loans - Current, Late, Fully Paid. The dataset can be found here.

Examples of LIME on Zoo dataset: We classify the data and explain the classifier using LIME and check how good the classifier is! We have used a Random forest classifier from sklearn library to classify these animals.

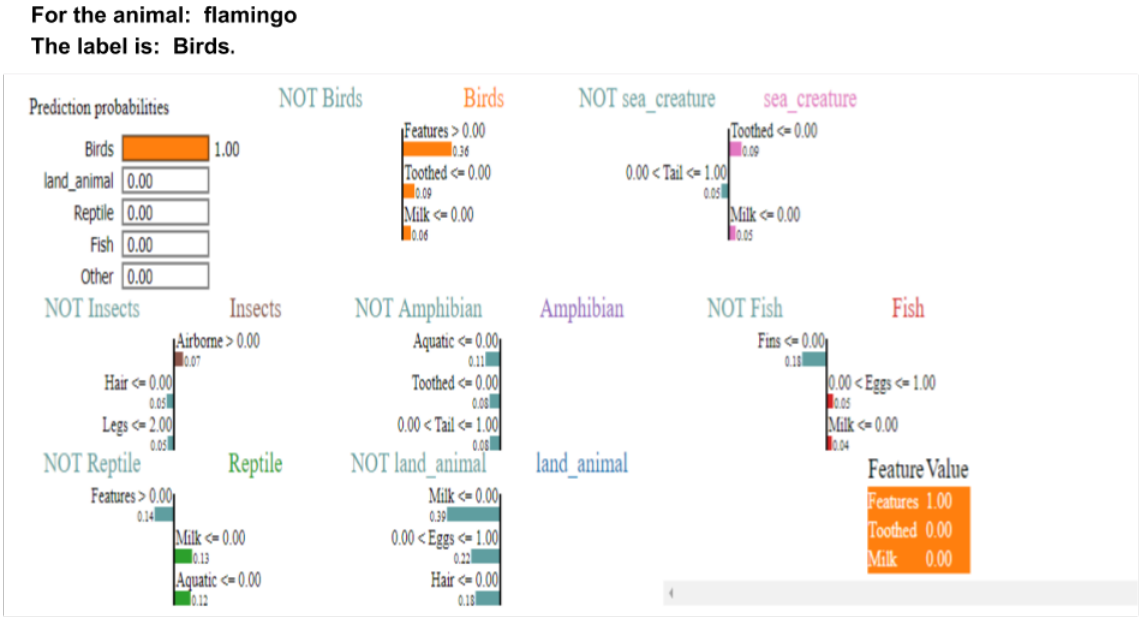


Figure 3: Explanation for the classification of flamingo as a bird

The above figure 3 shows that flamingo is classified as a bird with a probability of 1. The figure also explains what features make flamingo belong or not belong to a particular class. The bottom right indicates the three main features for which it is classification namely feathers, toothed and milk. We can observe that the classifier is doing a good job of predicting the class of flamingo and it also uses the correct feature(feathers) for this.

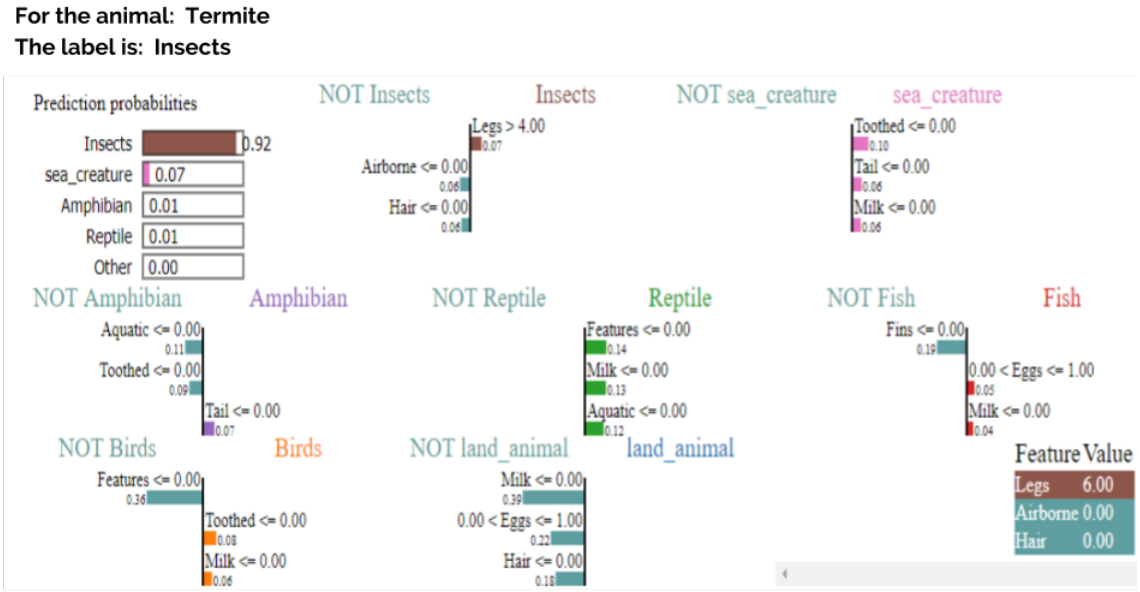


Figure 4: Explanation for the classification of termite as a insect

The above figure 4 shows that termite is classified as a insect with a probability of 0.92. The figure also explains what features make termite belong or not belong to a particular class. The bottom right indicates the three main features for which it is classification namely legs, airborne and hair.We observe that the probability of it being a sea creature is 0.07 because of it being toothed, not having a tail and not giving milk. We also can observe that the classifier is doing a good job of predicting the class of termite and it also uses the correct feature(legs = 6) for this.

Examples of LIME on Heart disease dataset: We classify the data and explain the classifier using LIME and check how good the classifier is! We have used a Random forest classifier from sklearn library to classify these animals.

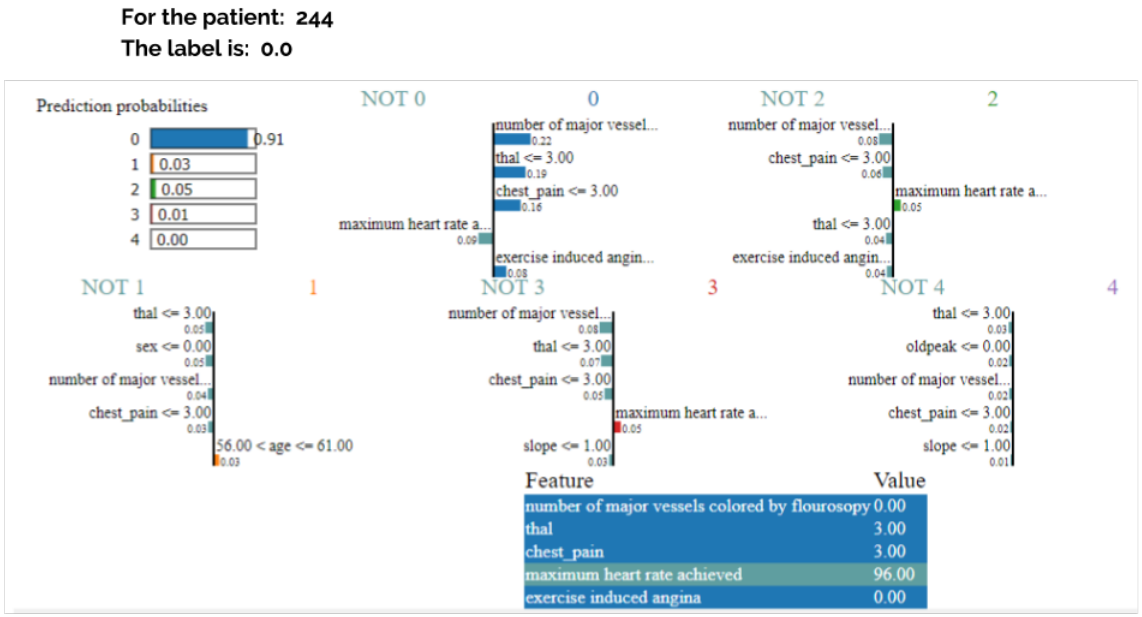


Figure 5: Explanation for the prediction of heart disease intensity for a patient

The above figure 5 shows that the classifier has correctly predicted a heart disease intensity as 0 indicating no heart disease. According to LIME, the main features for this classification are mentioned in the bottom right part of the figure. Also, it explains which features are used to accept or reject a particular class as the output.

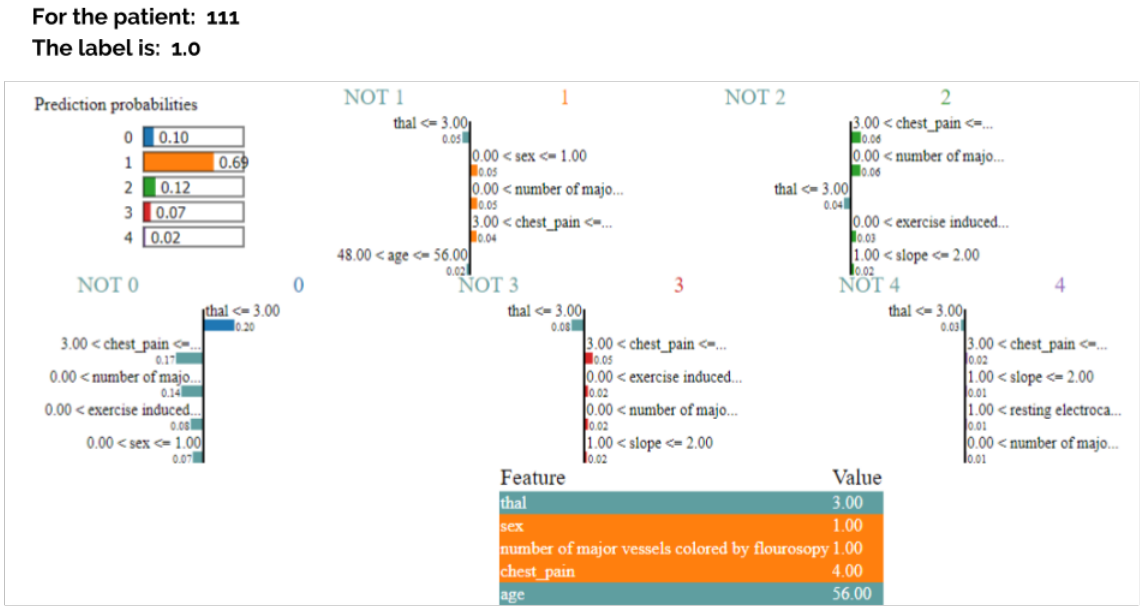


Figure 6: Explanation for the prediction of heart disease intensity for another patient

This figure 6 shows the explanation of LIME for the classification of another patient. Here, the classifier has correctly predicted a heart disease intensity as 1 indicating mild heart disease. It also shows that there is some chance of the intensity being 0 because of the low age of the patient.

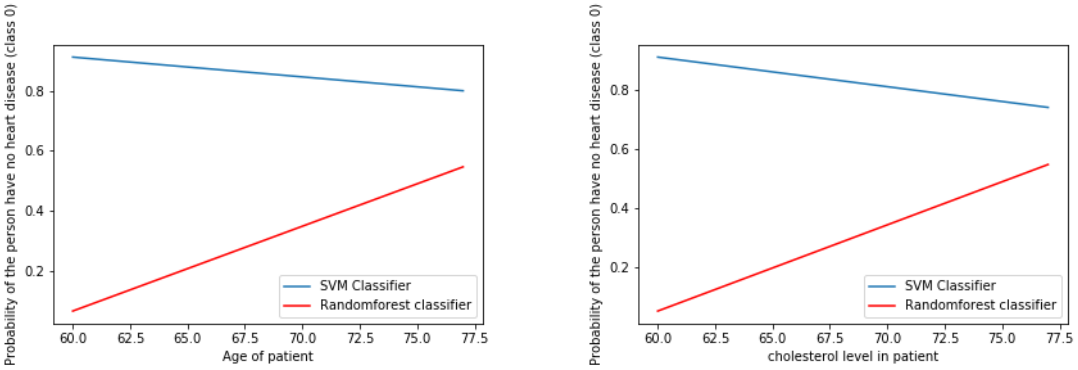


Figure 7: Left: Plot of prob. of No Heart disease v/s Age for patient 244, Right: Plot of prob. of No Heart disease v/s Cholesterol for patient 244

The Heart dataset was chosen specifically to compare different classifiers. For example, for a given patient, if we increase the age, we can intuitively say that the probability of no heart disease i.e. label 0 must decrease. This could be one criteria to judge how good a classifier is! In the above figure 7, we have compared two classifiers SVM and Random Forest Classifier for the same patient. For the SVM classifier, we see that the chances of having no heart disease increases with increase in age and the opposite happens in the Random Forest Classifier.
Below is another example in which we have chosen cholesterol instead of age. Again, the chances of having no heart disease increase with increase in the cholesterol for the SVM case. For the Random Forest Classifier, the reverse is true.

We conclude that the SVM classifier is not a good classifier for this dataset while the Random Forest Classifier gives good predictions that are both accurate and intuitively explanable.

Examples of aLIME: We demonstrate how aLIME makes it much more sleek and easy to comprehend than LIME's explanations. Anchors are **precise counter-factuals**. An anchor is a sufficient condition as they provide probabilistic guarantee. They have a clear (and adaptive) coverage. Using Anchor LIME, the idea is to create local explanations where they can be tied to if-then rules for more precise explanations. We saw many features being accounted with different weights in the LIME process. However, in case of aLIME process, these are distilled down to exemplar rules, or anchors, which are used to understand the key features which are driving the prediction. In Figure 8, aLIME finds certain conditions on features that largely result in income(b') being greater than 50K$. Similarly, in 9, aLIME gives certain anchors with a probabilistic guarantee regarding a prediction. We can observe that these figures have become extremely self explanatory(easy to understand).

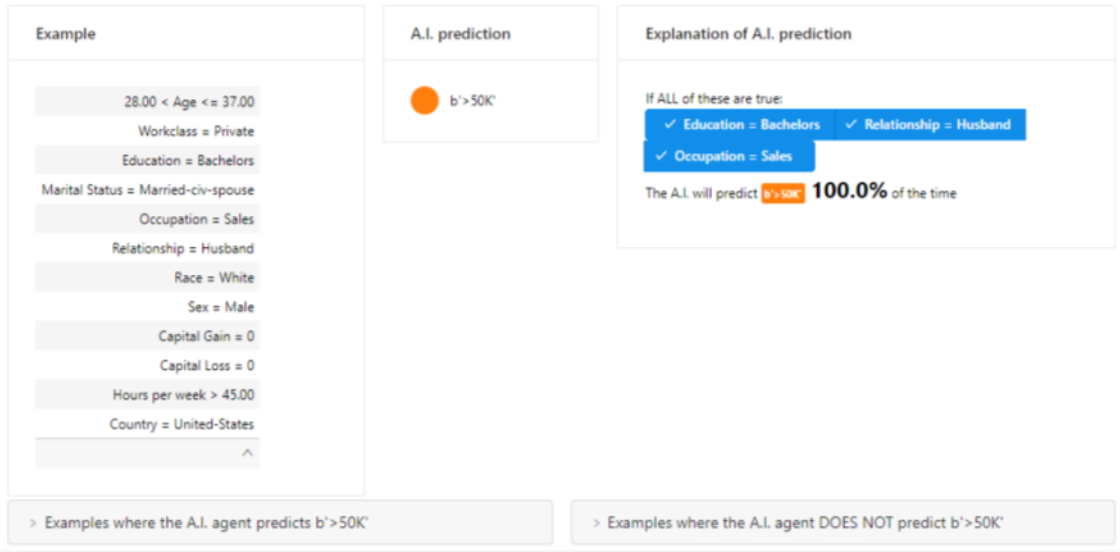


Figure 8: Explaining the prediction of a person's income by aLIME

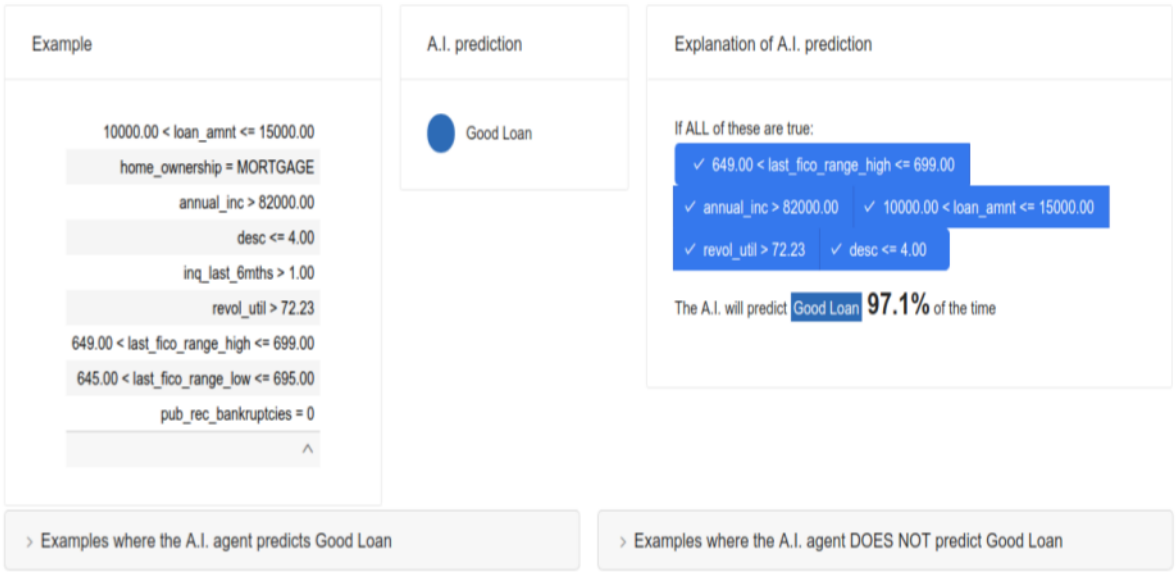


Figure 9: Explaining the factors that lead to the prediction of a Good Loan through anchors(aLIME)

6 Interpreting CNNs

Deep neural networks involve a large number of parameters and it is very difficult to visualize the parameters on a human-scale. [14] To make visualizations that are human-scale, we turn to the method of matrix factorization on the activation and gradient matrices.

6.1 Feature Visualization

Visualizing what a neuron (or a channel) sees is termed as feature visualization. To achieve this, we have to maximize the activation of the particular neuron [1]. Stating formally, let the input image be x, and the activation of the i^{th} neuron of the j^{th} layer of a trained CNN be to $h_{ij}(\theta, x)$, where θ denotes the network parameters (weights and biases). Then the optimization problem is

$$x^* = max_x h_{ij}(\theta, x)$$

We input a random image x, and follow an iterative gradient ascent procedure until we reach the image that maximally activates the selected neuron. While the procedure is useful, we yet don't know whether we have to visualize what a neuron sees, or what a channel sees. There is no particular reason to believe that the neuron is the fundamental unit to be visualized. Moreover, given the large number of neurons in today's networks, it is not possible to look at them all.

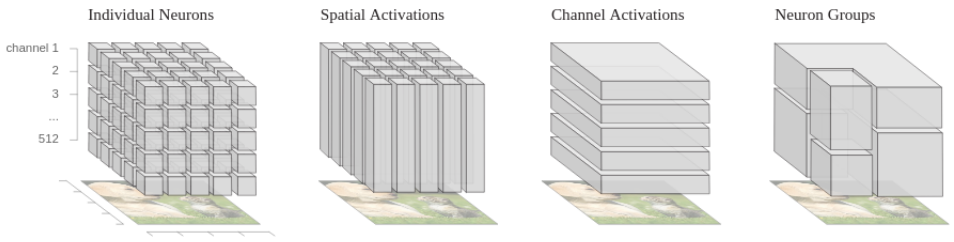


Figure 10: Different ways to choose groupings [4]

6.2 Matrix Factorization

In order to find the best groupings, we employ the matrix factorization method as proposed by Olah et al [4]. Figure 10 above shows the view of a single layer with the channels along the z-axis, and the image on x-y axes. Each cube can be thought of as a neuron, while the horizontal stripes are the channels. The vertical stripes above each pixel correspond to the spatial activations. We can group the neurons shown above - containing either the activation values or gradient values, and factorize on the resultant matrix (tensor). This gives us optimal groupings, and we can expect more semantically meaningful neurons. In order to do matrix factorization, we use scikit-learn's implementation with minor changes. The results of applying this method on the mixed4d layer of InceptionV2 model are shown in Figure 11. The six figures corresponding to each group are the feature visualizations of the corresponding group. We notice that the neuron groups are capturing semantically meaningful parts of the figure. For example, the red-coded group seems to capture a dog's ear, the green-coded group sees the dog's nose etc.

6.3 Adversarial Example

One way this method can be used is in recognizing adversarial images in the data, and explaining how the attack is impacting the predictions. So, we generate an adversarial example for targeted misclassification. We used a simple algorithm called L_∞ bound Bound iterative fast gradient sign (BFGS) method proposed by Kurakin et al [3]. Let x be the original image and y be the constructed image, then the L_∞ distance is given by

$$|x - y|_\infty = argmax_x(|x_1 - y_1|, \cdots, |x_n - y_n|)$$

The algorithm is as follows -

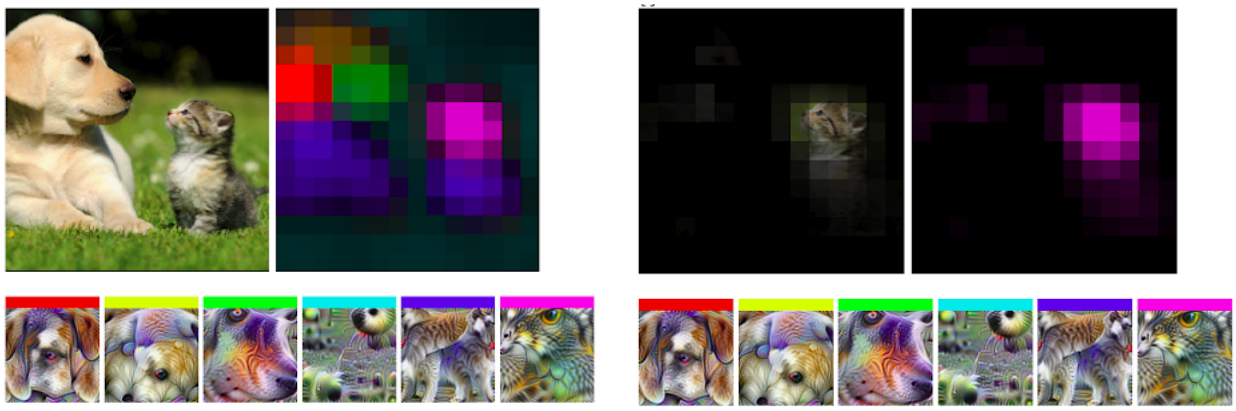


Figure 11: Left: Matrix Factorization on Activation Matrix of mixed4d layer, Right: Displaying the neuron group corresponding to the pink-coded matrix factor

1. Let adversarial image be $y^{(0)} = 0$. Set i = 1
2. Compute cross-entropy loss $loss_t\left(y^{(i-1)}\right)$
3. Compute the gradient of the loss wrt $y^{(i-1)}$
4. $y^{(i)} = y^{(i-1)} - \alpha * \text{sign}\left(grad\right)$
5. $y^{(i)} = \text{clip}\left(y^{(i)}, x - \epsilon, x + \epsilon\right)$
6. If $t \neq \text{pred}\left(y^{(i)}\right)$, go to Step-2.

The method essentially tells us in which direction the pixels intensity should be changed to minimize the loss function. We are clipping so that we don't move too far away from the original image. We ran the loop for 100 iterations only, as it is very expensive, checking if the target has been met every 10 iterations. We have used the cross-entropy loss function, and set $\alpha = 0.05$ and $\epsilon = 0.05$. The results of running the algorithm with target label as "ostrich" of the Imagenet dataset are shown in Figure 12. Clearly, the second image looks almost the same as the first image except that it is a bit grainy.

Figure 12: Left: Original Image, Right: Adversarial Image generated by L_∞ BFGS method

Now we employ matrix factorization on the same layer mixed4d as used before. As Figure 13 shows, the results show a striking difference with the previous ones. The dog's eye seems to be captured as before, but the blue and cyan coded

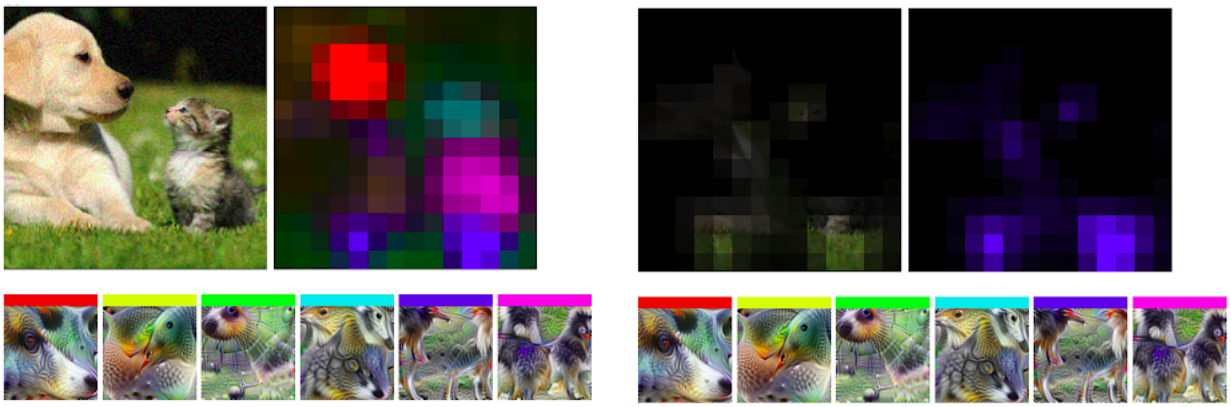


Figure 13: Left: Matrix Factorization on Activation Matrix of adversarial image, Right: Displaying the neuron group corresponding to the blue-coded matrix factor

groups clearly contain some beak-ish features. If we look at what the blue-coded group is capturing, we notice that it covers rather insignificant parts of the original image, like the grass under dog, some part of the dark background etc. This indicates that there must have been an adversarial attack on this image with high probability. In fact, we can also guess that the target to misclassify would've been a bird (We have used the target as ostrich).

The adversarial example is used as a proxy for any attack. The advantages of this method are two-fold. One, we have brought down the explanations to a human-readable form. Instead of searching in the entire neuron space for the neuron that is capturing the malice in our image, we have reduced it to a search in the layers. Looking at just around 6 groups per layer, we'll be able to catch the error. Secondly, we can now explain to others why our model has derailed, establishing a sense of trust and a sign of scope for improvement.

7 Tools/softwares and codes

7.1 Python Libraries

Our code for implementing LIME on Zoo dataset and heart disease dataset can be found here - . The results can be checked after downloading the datasets linked above. We worked with the Python libraries numpy, pandas and sklearn. We extensively used Jupyter notebook for implementing our code and used its interactive environment to efficiently debug the code. The LIME and aLIME libraries can be found in [7], [10] respectively.

7.2 Tools for interpreting CNNs

Our code for interpreting CNNs can be found in this link - https://tinyurl.com/y7msr2cp. Most of the code involving the svelte components is written by Olah et al [4]. The library used is TensorFlow's Lucid, which is excellent for visualizations of neural networks. Scikit-learn's factorization tools were used for matrix factorization. The entire code is in an interactive Google Colaboratory environment - which syncs well with the svelte components.

8 Things we learned

8.1 Importance of Explainable ML

Through this project we explored the emerging field of Explanable ML. We got insights as to why a certain model give its predictions. Through these insights we realized that the most accurate model may not always be the best one. For instance we implemented LIME Heart-disease dataset and found that according to a few classifiers the chances of having a heart disease decreases with increase in cholesterol levels; this is clearly untrue and could have severe

consequences if used in real life. Thus, if ML is to be applied on a large scale basis then explaining these models is of vital importance, especially in the fields of Health Care and Defence.

8.2 Data Preprocessing

We implemented LIME on different datasets comprising of tabular data. This task required preprocessing of datasets to the format specified to use the LIME library.

8.3 Human-scale interpretation of Deep Learning

We tried to attack our trained neural networks using targeted misclassification, for which we have learnt the BFGS method. We also got familiarized with state-of-the-art techniques of visualizing CNNs via Lucid, neuron group methods etc. As we had seen in the lectures, we saw how neural networks learn features and tried to represent those features in the form of human-interpretable groups. We did visual representation of different hidden layers and saw how neural networks become more and more complex with increasing depth.

8.4 Web App Development

During the development of the web-app we learnt the working of lime library and studied the implementation of lime using modular paradigm. This task also required us to know GitHub, Django, Databases and a working knowledge of web development. While working on the app, we also learned about collaborative software development.

9 Conclusion

We can conclude that explainability of a classifier's predictions is of utmost importance for its deployment in the wild. Hence, we shouldn't madly rush after increasing accuracy of classifier. Instead, aim for a holistic progress of machine learning models that takes accuracy as well as explainability into account.

10 Future Work

The field of explainability is still quite raw and a rich space exists for various new algorithms for machine learning models and better interactions with neural networks. Some directions that we would like to work on in the future are:

1. Note that our feature visualizations look a bit messy and incoherent which can be improved upon using filtering techniques. This would help us in giving a clearer picture of what neurons detect. Also, Random input sampling methods for CNNs [6]
2. Extending LIME to clustering tasks and other unsupervised methods. Specifically, trying to explain Unsupervised learning models like Latent Variable Models(LVMs).
3. Rectifying A-LIMEs drawbacks like failure in case of highly nonlinear decision boundary and thereby improving upon A-LIME .
4. Explanations based on Causal Inference (rather than correlation).
5. Explanations through Visual Question Answering (VQA).
6. Extending our website to newer classifiers as they are developed and for various other explanation techniques like Grad-CAM [12].

11 Acknowledgements

We thank our advisor and instructor, Dr. Piyush Rai, for his support and encouragement throughout the academic semester, and for introducing and guiding us to the field of explanable machine learning.

References

[1] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. 2009.

[2] W. Knight. Intelligent machines the dark secret at the heart of ai. *MIT Technology Review*, 2018.

[3] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[4] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.

[5] P. Olson. The algorithm that helped google translate become sexist. *Forbes*, 2018.

[6] V. Petsiuk, A. Das, and K. Saenko. Rise: Randomized input sampling for explanation of black-box models. In *BMVC*, 2018.

[7] M. T. Ribeiro, S. Singh, and C. Guestrin. https://github.com/marcotcr/lime. 2016.

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

[9] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin. https://github.com/marcotcr/lime. *AAAI*, 2018.

[11] W. Samek, T. Wiegand, and K. Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *CoRR*, abs/1708.08296, 2017.

[12] 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*, abs/1610.02391, 2016.

[13] J. Vincent. Google fixed its racist algorithm by removing gorillas from its image-labeling tech. *Verge*, 2018.

[14] Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. *CoRR*, abs/1710.00935, 2017.