**Week 4: Model Explainability**

Model Explainability is a technique to understand any given machine learning model. When a programmer deploys any trained machine learning model, the model needs to be explained.

This technique is very much essential because

1. When we interpret any model, the trust of machine learning model increases.
2. After understanding the model, we find any bias present in that model.
3. It is also important when we perform model debugging during the development.

There are 2 ways to interpret a model:

1. Global Interpretation:

* It will help to understand how a particular model takes a decision for the entire structure.
* It will explain the complete behavior of the model and help in understanding which model will be suitable for deployment.

1. Local Interpretation:

* It will help to understand how the model will takes a decision for single instance.
* We can explain the individual predictions.
* It will help to understand the behavior of the model in the local neighborhood.

ELI5 (Explain like I am 5): ELI5 is a python library that allows visualize and debugs various machine learning models using unified API.

Using global interpretation, the model parameters are inspected, and we try to find out how the model works globally. ELI5 will provide **show\_weights()** function for this interpretation.

Using local interpretation, it will check for a single prediction of the model and figures out why a particular model takes a certain decision. ELI5 will provide **show\_prediction()** function for this interpretation.

**Note**:

* explain\_weights and show\_weights methods are present in ELI5 library which are used to explain the weights of a machine learning model.
* explain\_weights returns an explanation of feature containing model's weights, importance of each weight. This information can be used to understand how the model works globally.
* show\_weights returns an HTML object that displays the model's weights in a table. But it does not provide much information.
* explain\_prediction and show\_prediction methods are used to explain a single prediction made by a machine learning model.
* explain\_prediction is helping us to explain the information about the prediction like the importance of each feature, the impact of each feature on the prediction and visualization of the explanation. This can be used to understand the reason why a particular model made a particular prediction.
* show\_prediction will display the explanation of the prediction in a table. But it is not a detailed explanation.

**Steps used in eli5**:

1. In our example, we explain the top 5 features of the Decision Tree Model and Random Forest model using explain\_weights. It is done using eli5.explain\_weights (model\_dt, feature\_names = X\_features\_dt, top=5) or eli5.explain\_weights (model\_rf, feature\_names = X\_features\_rf, top=5).
2. We also consider explaining different estimator parameters both the classifier models. It is done using eli5.explain\_weights(model\_dt) or eli5.explain\_weights(model\_rf).
3. eli5 shows us the contribution of each feature in predicting the output. In this Decision Tree Model, bias is having the highest contribution. It is also same for Random Forest Classifier Model. It is done using eli5.explain\_prediction (model\_dt, np.array(X\_test)[0]) or eli5.explain\_prediction(model\_rf , np.array(X\_test)[0]).
4. Also, different combination of features was being compared for different predictions in both the classifier models for different row labels. E.g. 1) eli5.show\_prediction (model\_dt, X\_test. iloc[0], feature\_names=X\_features\_dt, show\_feature\_values=True) or 2) eli5.show\_prediction(model\_rf, X\_test.iloc[0], feature\_names=X\_features\_rf, show\_feature\_values=True).
5. Next, different weights are shown for both the classifiers using show\_weights. E.g. eli5.show\_weights(model\_dt, feature\_names=X\_features\_dt, top=35,show\_feature\_values=True) or eli5.show\_weights(model\_rf, feature\_names=X\_features\_rf, top=35,show\_feature\_values=True).

**ELI5 Permutation Model**: eli5's **show\_weights** method is good, but for more complex models, such as trees the information provided starts to be less helpful.

Since show\_weights is accessing the internal weights of a model, it does not work with all algorithms, making it harder to compare different models.

eli5 implements another technique called Permutation Importance that is model agnostic. By shuffling at the values of a feature randomly, we can observe how that affects the predictions and quantify how important that feature is.

If we repeat on all features, we can get the overall importance of each feature for comparison.

It works only for global interpretation. As output, it gives weight values like feature importance that you get with algorithms.

For each feature:

1. Shuffle the values in the data.

2. Generate predictions using model.

3. Compute the decrease in accuracy.

It will compare the impact on accuracy of shuffling each feature individually. It will shuffle numbers of times and give as output average importance & standard deviation. In ELI5, a prediction is basically the sum of positive features inclusive of bias.

This technique is important because:

1. ELI5 Permutation model is used for model explainability because it can be used to measure the importance of each feature in a model by randomly shuffling the feature values of a single feature and checking for the impact on the model's prediction.
2. It is relatively easy to understand and interpret, effective in identifying the most important features in the model and computationally efficient. e.g. In the test dataset, if we have randomly shuffled the values of logged\_in and model's accuracy decreases, then we come to know that the logged\_in is an important feature of the model.

**Different Global Interpretation methods**

### **Understanding the model using Partial Dependence Plots (PDPs):**

* It will show the dependence between the target response and a set of input features of interest.
* The partial dependence can be interpreted as the expected target response as the function of the input features of interest.
* The size of the input features must be small, so the input features of interest are chosen amongst the most important features.
* This technique is helpful because it provides a global view of how a feature will affect the predictions of a model.
* **Pros**:

1. They are easy to understand.
2. They can be used to identify the important features and interaction effects.

* **Cons**:

1. They can be time-consuming.
2. They can be sensitive to the choice of hyperparameters.

We use PartialDependenceDisplay.from\_estimator () method to generate plots for each feature.

### **Understanding the model using Individual Conditional Expectations (ICE):**

* ICE is an extension of PDP. We can explain heterogeneous relationships.
* While PDP supports two feature explanations, using ICE, we can explain only one feature at a time.
* PDP will show the average effect of a feature while ICE plot shows the effect for a single instance.
* ICE is required in global interpretation because they provide a more detailed understanding of how a model makes a prediction.

We use PartialDependenceDisplay.from\_estimator() method with kind=’individual’ or kind=’both’ to generate ICE plots.

**Different Local Interpretation methods**

* These methods are typically used with machine learning models whose predictions are difficult to explain.
* LIME and SHAP determine feature importance in complex models where direct interpretation of model predictions is not feasible.
* Understanding these methods can help data scientists approach model explainability for a variable of machine learning models whether they are simple or complex.

1. **Local Interpretation of Model Explanation (LIME)**:

* LIME treats the model as a black box.
* It doesn’t distinguish between random forest, a decision tree, or neural networks.
* It uses linear models to provide a local explanation.
* LIME provides a local interpretation by modifying feature values of a single data sample and observing its impact on the output.
* It builds a model from the input and model predictions.
* An interpretable model can be used as a surrogate model.
* Because LIME is a model agnostic technique, therefore it can be used on any model.
* It aims to identify an interpretable model over the interpretable representation that is locally faithful to the classifier.

**Properties of LIME:**

1. Interpretable: It should provide a qualitative understanding between the input variables and the response.
2. Local Fidelity: It might not be possible for an explanation to be completely faithful unless it is the complete description of the model itself. It should be at least locally faithful. i.e. it must replicate the model’s behavior in the vicinity of the instance being predicted.
3. Model Agnostic: The explainer should be able to explain any model and should not make any assumptions about the model while providing explanations.
4. Global perspective: The explainer should explain a representative set to the user so that the user has a global intuition of the model.

**Steps involved in LIME:**

1. It creates a permutation (fake) of the given data.
2. It calculates the distance between permutations and the original observations. Also, we can specify the distance measured.
3. It makes predictions on the new data.
4. It picks “m” features that describe the complex model. It is an outcome from the permuted data in the best possible way through the maximum likelihood approach. Here, we can decide the number of features i.e. the value of “m” we want to use.
5. It picks the “m” features and fits a simple model to the permuted data with the similarity score as weights.
6. The weights from the simple model are used to provide explanations for the complex model’s local behavior.

**Pros**:

1. It will work well on images, text, and tabular data.
2. It is model agnostic.

**Cons**:

1. It suffers from labels and data shift, explanations dependent on the choice of hyperparameters.
2. Sometimes similar datapoints may have different explanations.

LIME is typically faster to compute than SHAP.

1. In LIME, lime. lime\_tabular.LimeTabularExplainer() method is used to describe the explainer.
2. Next, using explainer.explain\_instance() method, we find the intercept, prediction\_local and right prediction values are calculated for both the classifiers.
3. In the next step, lime\_explainer.show\_in\_notebook(show\_table=True, predict\_proba=True, show\_predicted\_value=True, show\_all=True), we get the prediction probabilities along with the feature, value table for all the features used in the model.

#### **Lime explainer for class 0 and class 1**

* An intercept means how likely is the positive class when all the features happen to have zero value.
* A large intercept value means a large chance of the positive class Formula for prediction\_local: exp.local\_exp = exp.intercept[1] + sum([weight[1] for weight in exp.local\_exp[1]])
* Right: Actual prediction of the model

### **SHapley Additive exPlanations (SHAP)**:

* SHAP is framework which explains the output of any model using Shapley values. It follows Game theory approach.
* It can compute efficiently on the specific model classes.
* It can be used globally and locally.

Though, SHAP will be more accurate with feature explanation than LIME but it is mathematically rigorous. For this reason, SHAP is more computationally intensive and is a good option.

The **predict\_proba** function will return the probabilities of a classification label. This function analyses a row of independent data in a dataset and returns the probability of a result. In our results, we use summary plots like dot, bar to get the following results:

* src\_bytes has the highest mean SHAP value in the Decision Tree Classifier.
* Flag has the highest mean SHAP value in Random Forest Classifier.

**References**

1. <https://www.analyticsvidhya.com/blog/2021/11/model-explainability/#:~:text=Model%20explainability%20refers%20to%20the,a%20particular%20disease%20or%20not>.
2. <https://www.analyticsvidhya.com/blog/2020/11/demystifying-model-interpretation-using-eli5/>
3. <https://www.analyticsvidhya.com/blog/2022/01/build-a-trustworthy-model-with-explainable-ai/>
4. <https://coderzcolumn.com/tutorials/machine-learning/shap-explain-machine-learning-model-predictions-using-game-theoretic-approach>
5. <https://towardsdatascience.com/a-look-into-global-cohort-and-local-model-explainability-973bd449969f>
6. <https://shap.readthedocs.io/en/latest/generated/shap.plots.force.html>
7. <https://www.twosigma.com/articles/interpretability-methods-in-machine-learning-a-brief-survey/>
8. <https://medium.com/mlearning-ai/explainable-ai-a-complete-summary-of-the-main-methods-a28f9ab132f7>
9. <https://www.analyticsvidhya.com/blog/2022/07/everything-you-need-to-know-about-lime/>
10. <https://christophm.github.io/interpretable-ml-book/pdp.html>
11. <https://coderzcolumn.com/tutorials/machine-learning/how-to-use-eli5-to-understand-sklearn-models-their-performance-and-their-predictions>
12. <https://github.com/mayur29/Machine-Learning-Model-Interpretation/blob/master/ELI5%20Model%20Interpretation.ipynb>
13. <https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5>