**How to Explain Unexplainable Models Using Lime**

****

**Author:** *Uman Sheikh*

**Introduction**

Today in this document we are going to learn how to explain our AI model results or more simply how our model get that predicted value. For this we will use a python library called *lime.* It basically explains these predictions locally and it supports all models.

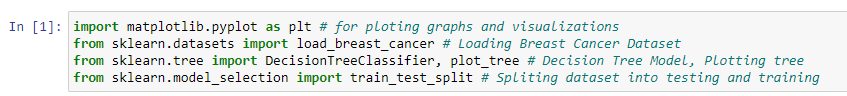
Now let’s start.

Install 3 python libraries:

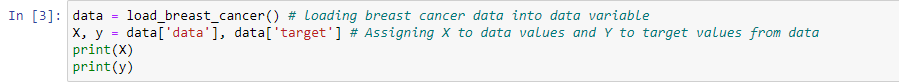
* Scikit-learn (for Machine Learning)
* Matplotlib (for visualizations)
* Lime (for Explanation)

Before diving into unexplainable models, let’s start with an explainable model *i.e.* Decision Tree.

Let’s do our imports. I will use breast cancer dataset provided by scikit-learn library. I want to keep it as simple as possible.



*DecisionTreeClassifier* is one of the explainable models that explains how it concluded that result. Now let’s start how we can check what steps *DecisionTreeClassifier* took to conclude that result. But before that let’s train our model.



The output of this can be seen as:

A screenshot of a computer code

Description automatically generated

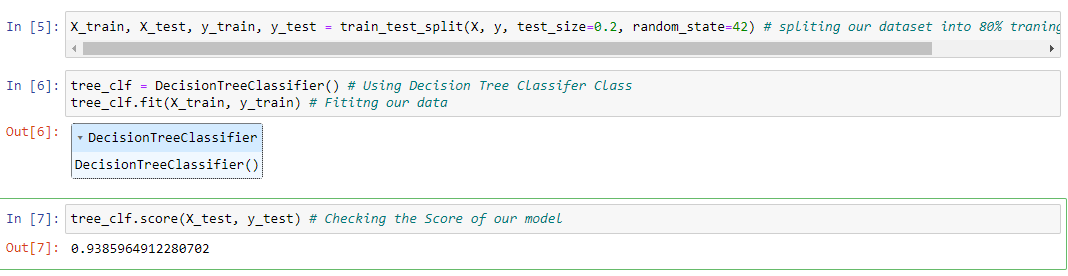
Whereas the array contains different values and 0, 1 means malignant and benign respectively.

We can confirm that by using:

A close-up of a computer screen

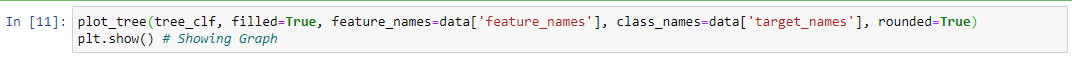
Description automatically generated

Now training our model,



We can see our model is 93% accurate. Of course, we don’t want to explain a model which is not accurate.

Now let’s explain the model by plotting the tree graph, since it is already explainable by default and scikit-learn algorithm provides us with *plot\_tree* function.



Output is:

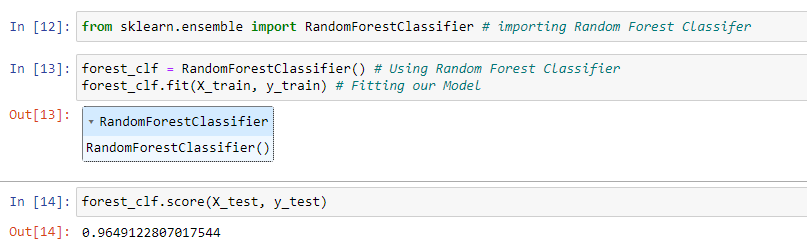
A diagram of a structure

Description automatically generated

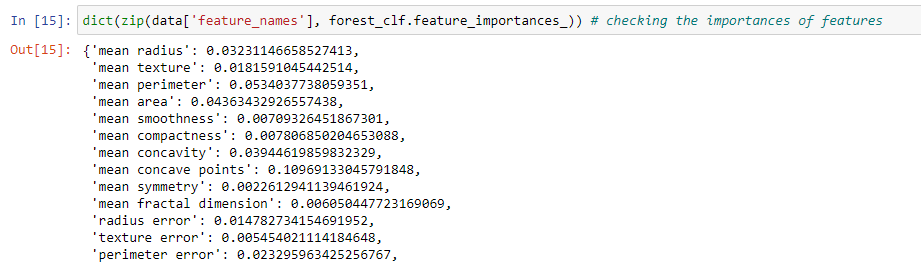
* **tree\_clf**: This is the decision tree classifier that I've already trained and now wish to visualize.
* **filled=True**: This parameter, when set to **True**, colours the nodes to indicate the majority class in classification or the mean value in regression.
* **feature\_names=data['feature\_names']**: This specifies the names of each feature in the dataset for which the decision tree was trained.
* **class\_names=data['target\_names']**: Like **feature\_names**, this parameter improves interpretability by labeling the leaves with the names of the target classes.
* **rounded=True**: This parameter rounds the corners of the nodes in the visual representation of the tree, making the diagram aesthetically pleasing and easier to read.

We can see what happens before our model predicted the result.

Now let’s do with an unexplainable model *i.e. RandomForestClassifer.*

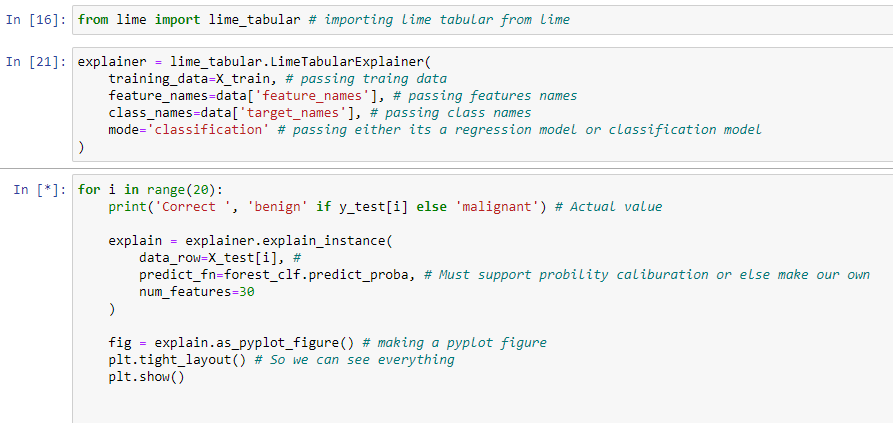


In this case our model accuracy is 96%. Let’s explain how it concludes results.



This will give us the importances of each feature in our model/dataset.

But the thing is we cannot explain exactly how the individual decision forest classifier made. That is where lime comes in.



The code seems a bit tricky, but it is easy to understand.

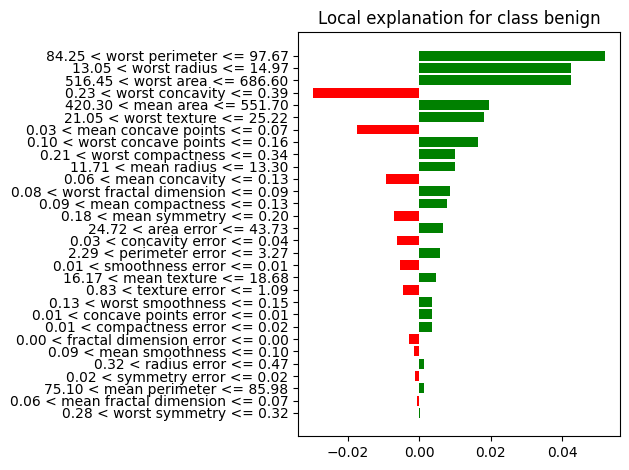
First, we import lime tabular from lime. Now what is lime tabular? **lime\_tabular** is a specific implementation of LIME for tabular (or structured) data, which is the most common form of data used in machine learning. Tabular data refers to data that is organized into rows and columns, like in a spreadsheet or a SQL table.

Then we make an explainer object of Lime Tabular Explainer and pass a few arguments to it. Arguments are straight forward except predict\_fn. It requires probability calibration.

Why do we use for loop? Just to get first 20 results from our lime explainer.

Results are as follows:







A graph with text overlay

Description automatically generated

And up to so on.

Now how do we read it? I am reading the last one. Worst concave points are greater then.

0.16. The worst radius is greater than 18.41 and so on. On these bases our model predicted the results.

**Code available at following link:**