# Model interpretability

Before we begin to explore the interpretability of the model, we first need to figure out why we should solve this problem. The following three aspects will explain the significance of model interpretability.

- First of all, it helps to tune the model. The general real business scenario will have a lot of untrustworthy, unorganized dirty data. Engineers may add potential errors when preprocessing data, or accidentally make a data leak. Considering various potential catastrophic consequences, the idea of the tuned model is especially important. When you encounter data that cannot be explained with existing business knowledge, in-depth understanding of the model prediction pattern can help you quickly locate the problem. Anyway, this is of great significance to algorithm engineers.
- Secondly, it can give us instructions in feature engineering. Feature engineering is usually the most effective way to improve the accuracy of the model. Usually involves using various algorithms to manipulate the original data (or the previous simple features) to get new features. Sometimes you only use your intuition or business knowledge to complete the feature engineering process. In fact, this is not enough. When you have hundreds of original features, or when you lack detailed business background knowledge, you will need more guidance.
- Finally, it guides the direction of future data collection. Many companies and institutions will use data science to guide them to collect data from more aspects. In general, collecting new data is likely to be expensive or difficult, so everyone wants to know which data is worth collecting. Model-based insight analysis can teach you a good understanding of existing features, which will help you infer what new features are useful.

Many people think that machine learning or deep learning models are black boxes, and they think that models can make good predictions. But people cannot understand the logic behind these predictions. Indeed, many data scientists do not know how to use models to explain the actual meaning of data. Here I will introduce two method called LIME (Local Interpretable Model-Agnostic Explanations)and SHAP (SHapley Additive exPlanations) which could provide the ways of explaining models.

# LIME

In order to understand the behavior inside the model, all we can do is to interfere with the input and then observe how the predicted results will change. Experiments show that this approach is useful for interpretability, because we can change the input by changing components that humans can understand (such as words or parts of images), even if the model uses more complex components (such as words Vector) as input features.

The key intuition behind LIME is that it is much easier to approximate a black box model locally (near the prediction we want to explain) with a simple model than to approach the model globally. This idea can be realized by setting the weight of the changed input image, and the value of the weight is the value of the similarity between the changed graph and the instance we want to explain.

Let's take an image of tree frog as an example to illustrate the LIME:

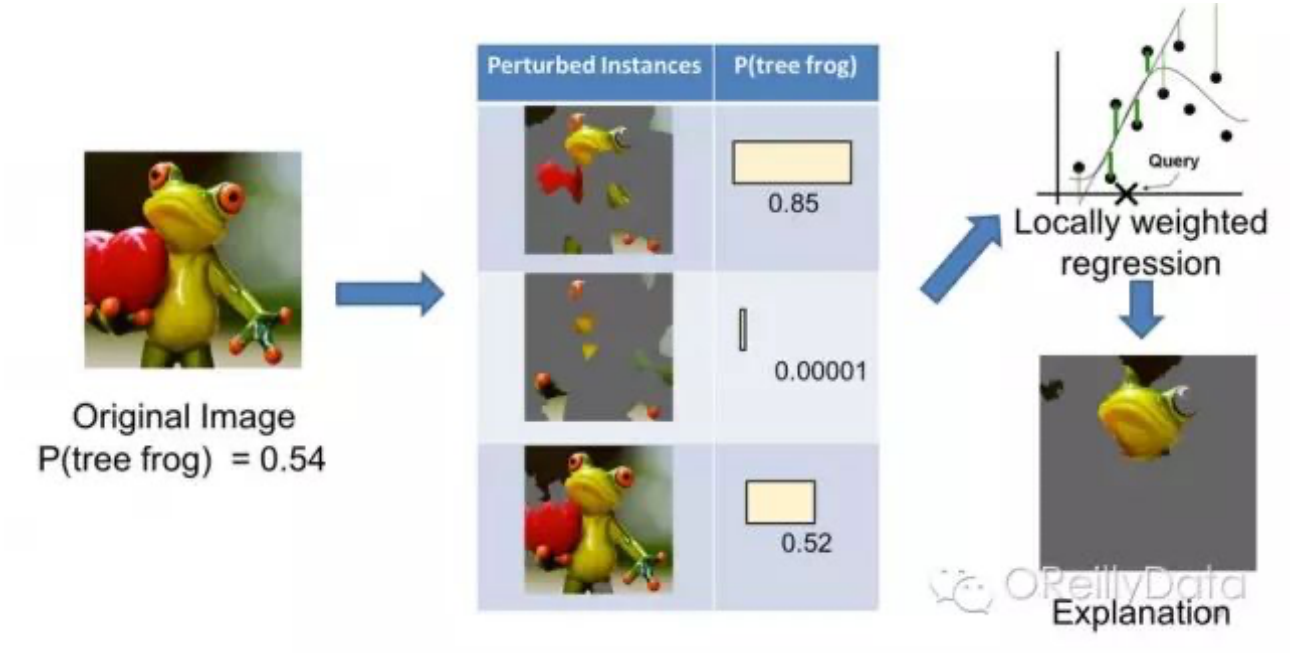
Original Image

Interpretable Components

Here we divide the picture into many interpretable components (superpixels).

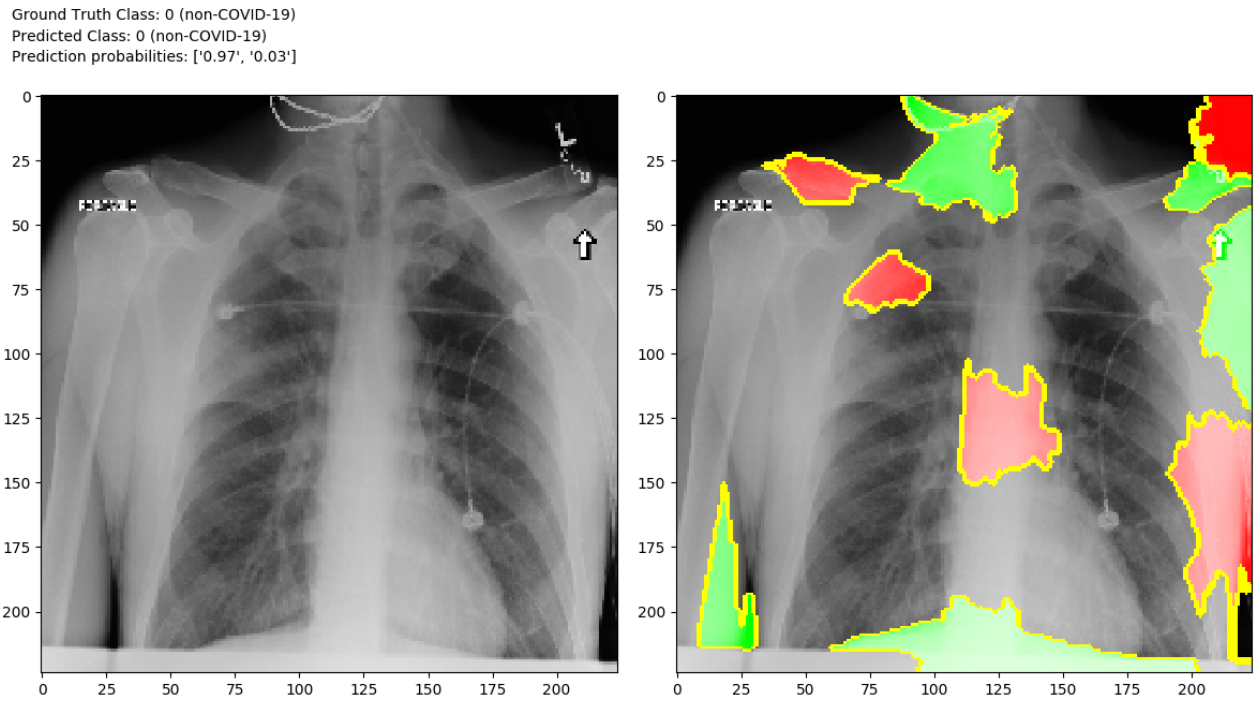
We generate a modified image data by "hiding" some interpretable components (in this example, all hidden components are set to gray color in below figure). For each modified instance, the model will determine whether the image instance contains a tree frog with a certain probability. Then we got a simple (linear) regression model on this locally weighted image, and we are more concerned about errors that occur on modified images that are closer to the original image.

Finally, we give the superpixel with the highest positive weight as an explanation, and change all other parts to gray color.

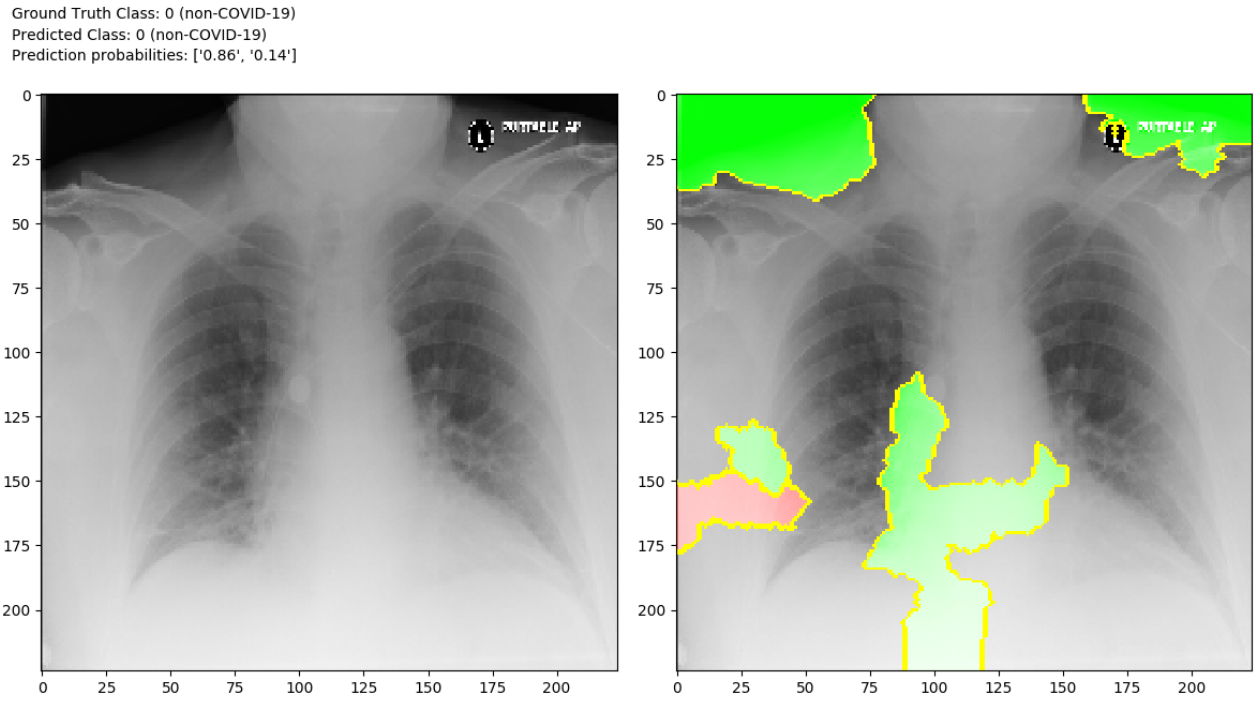



This is how LIME works. In our COVID data set, the CT image of the lungs of the patient is also divided into many superpixels in this way. After LIME interpretation, superpixels in CT images that are helpful for diagnosis as COVID-19 will be marked in green. Conversely, superpixels that are helpful in diagnosis as non-COVID-19 will be marked in red. Here are a few examples of LIME explanations.

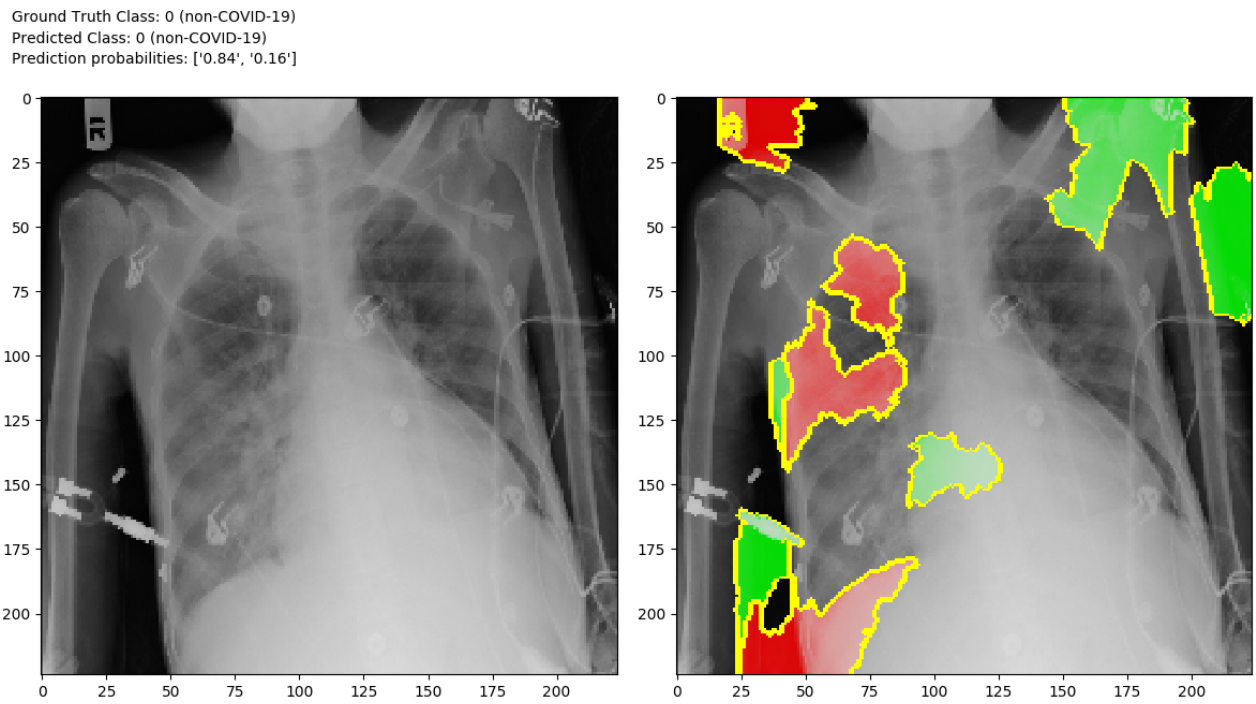
IE Explanation for image /home/paperspace/covid-cxr/data/RAW_DATA/unknown_rsna/rsna0a8efdc1-1dff-4d65-8948-c56a5d74a906.j



E Explanation for image /home/paperspace/covid-cxr/data/RAW_DATA/unknown_rsna/rsna014b7b58-f641-4477-8bbc-ae6f337745d6.




E Explanation for image /home/paperspace/covid-cxr/data/RAW_DATA/unknown_rsna/rsna0ae738d0-5bdd-4ebf-9c1f-6b10a25be92a.

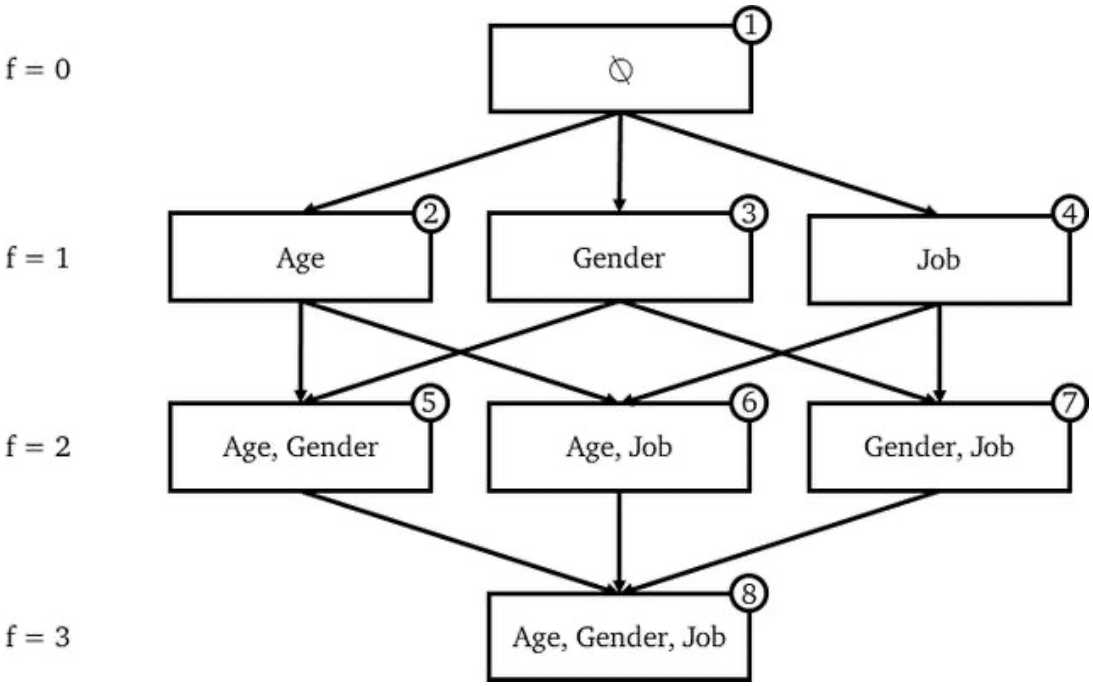


# Shapley Value

The SHAP is based on the Shapley value, which is a concept in game theory. But game theory requires at least two things: games and participants. How does this apply to the interpretability of machine learning? Participants in SHAP means the features. The Shapley value is based on the idea that each possible combination of features should be considered to determine the importance of a single feature.

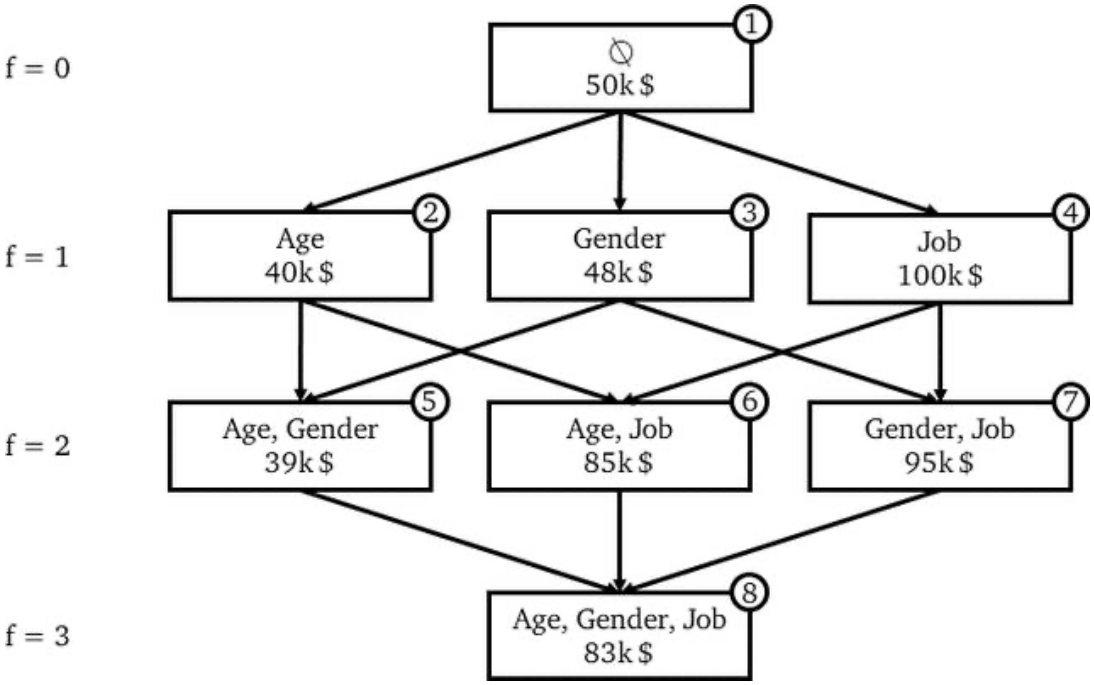
Suppose we have a predictive model (Assuming linear regression, but it can also be any other machine learning algorithm), the model can predict a person's income based on a person's age, gender, and work. In our case, this corresponds to every possible combination of f features (f from 0 to F, where F is the number of all available features in our case). The figure below shows the idea of it.

Each node represents a combination of features. Each edge represents a feature that did not exist in the previous combination. We know mathematically that when we have F features, the total number of possible feature combinations is 2 ^ F. Here we have 2^3 = 8 types of combinations which is already indexed in the figure.




Now, Shapley value needs to train a different prediction models for each different feature combination, that is, 2 ^ F models (in our case it's 8 models). Of course, these models are completely identical in terms of their hyper-parameters and training data. The only thing that has changed is the features included in the model.

Suppose we have trained 8 linear regression models on the same training data. And these 8 linear regression models have made corresponding predictions on our wages. The predicted results are shown in the figure below.




In the figure above, take index 2 as example, the linear model predicts the salary with only "age" feature is 40k$. Here, each node represents a model and each edge represents the marginal contribution (MC) of the features to the model.

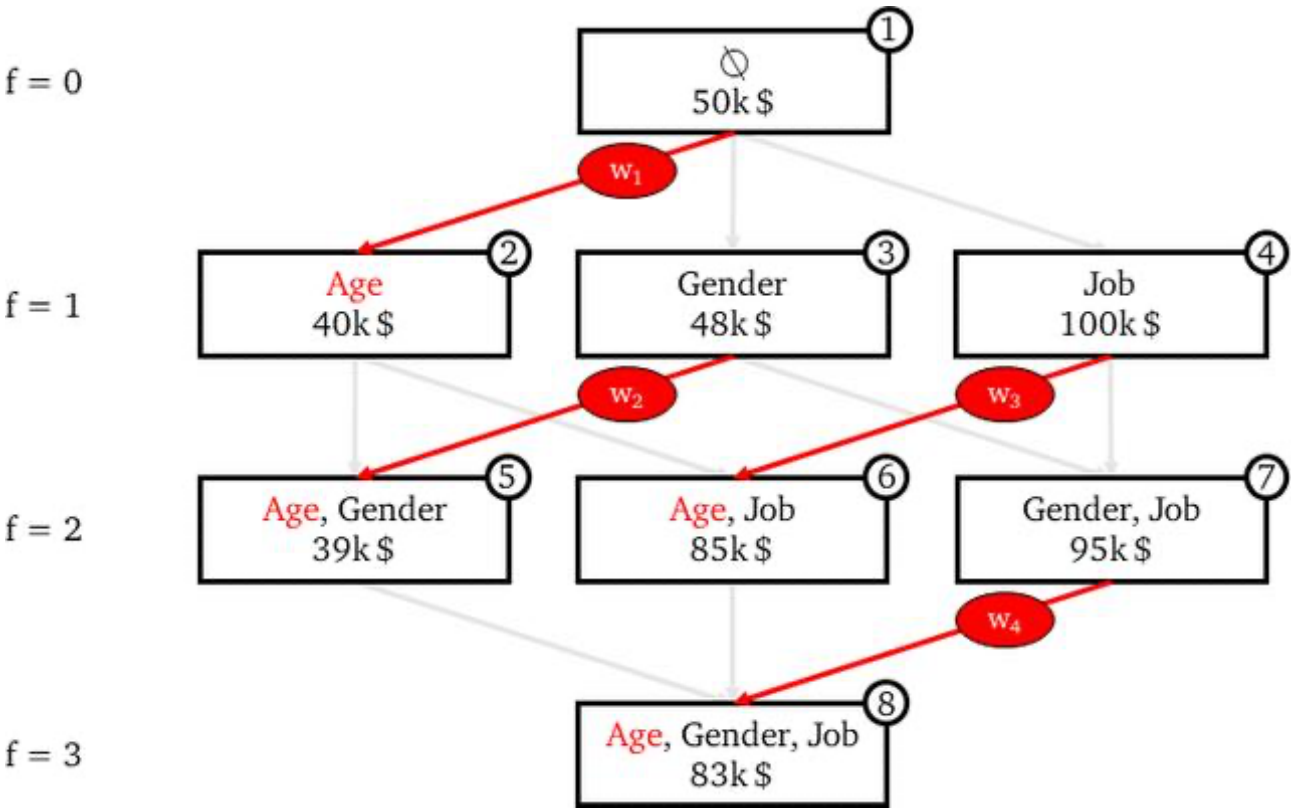
So here we can calculate the marginal contribution of edge between node 1 and node 2:

Marginal Contribution(1,2) = Prediction(2) - Prediction(1) = -10k$

But here MC indicates that the marginal contribution of "age" to models containing only "age" as a feature. So the Function needs a little modification:

Marginal Contribution_{age}(1,2) = Prediction_{age}(2) - Prediction{None}(1) = -10k$

Then we need to find the total MC of "age" which requires to consider the MC of Age in the model. The red line in the below figure indicates all models containing "age" feature.




Finally we can calculate the total Shapley value of feature “age” with weighted sum of them:

Shapley_{age} = w1*MC_{age}(1,2) + w2*MC_{age}(3,5) + w3*MC_{age}(4,6) + w4*MC_{age}(7,8)

Here:

w1 + w2 + w3 + w4 = 1

But here raises another question, that is, how to assign weights of different model? We consider this issue from the perspective of each line (f = 0, f = 1, f = 2, f = 3). Each row is used for the arrangement of the combination of three features, so the marginal contribution of a feature in each row is certain. In other words, taking “age” as an example, the sum of the “age” marginal contribution weights of the single feature model should be equal to the sum of the “age” marginal contribution weights of the dual feature model and the sum of the “age” marginal contribution weights of the three feature model.

Besides, the marginal contribution weight of each model containing “age” features in

$$SHAP_{feature}(x) = \sum_{set:feature \in set} [|set| \times \binom{F}{|set|}]^{-1} [Predict_{set}(x) - Predict_{set \setminus feature}(x)]$$

each row should be the same. This is because in the process of permutation and combination, the probability of generating these two feature combinations is identical.

From a mathematical point of view, the relationship between weights is as below:

① w1 = w2 + w3 = w4

② w2 = w3

So here we can calculate the total SHAP value of “age” by solving the above equations.

w1 = 1/3, w2 = 1/6, w3 = 1/6, w4 = 1/3

SHAP_{age} = (1/3) * (-10k$) + (1/6) * (-9k$) + (1/6) * (-15k$) + (1/3) * (-12k$)

= -11.33k$

SHAP_{gender} = (1/3) * (-2k$) + (1/6) * (-1k$) + (1/6) * (-5k$) + (1/3) * (-2k$)

= -2.33k$

SHAP_{job} = (1/3) * (50k$) + (1/6) * (45k$) + (1/6) * (47k$) + (1/3) * (44k$)

= 46.67k$

Generalized to any feature and any set, we get the general formula of Shapley value:

By calculating the Shapley values of different features, we can clearly calculate the marginal contribution of each feature to the overall model. Combined with our professional knowledge for analysis, we can give corresponding explanations to the black box machine learning model and deep learning model.

## SHAP

SHAP considers the Shapley value interpretation as an additive feature, and SHAP interprets the model's predicted value as the sum of the Shapely values of each input feature (this is also the definition of the Shapely values). The formula is shown below:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j$$

Where g indicates the interpreted model, z’ Indicates whether the corresponding feature can be observed. Φj is the Shapely value of each feature and Φ0 is Is a constant in explaining the model.

SHAP has three ideal properties:

Local accuracy: The sum of the Shapely value is equal to the output of the model we want to explain, that is to say, for each sample, the sum of the Shapely value of each feature and the constant value is equal to the output value f (x) of the model:

$$f(x) = g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j$$

Missingness: It means that if a feature do not appear (z' = 0), it will have no SHAP contributes.

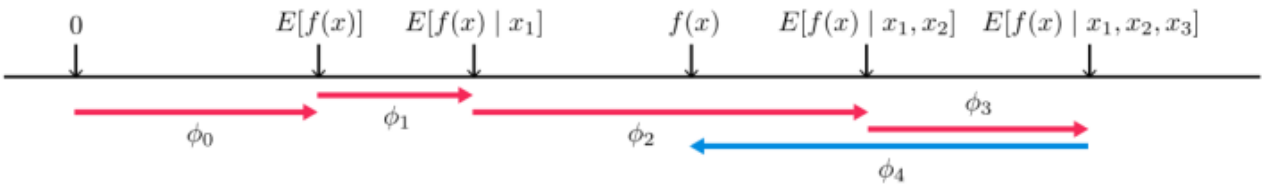
$$x'_j = 0 \Rightarrow \phi_j = 0$$

Consistency: The consistency attribute indicates that if the model changes so that the marginal contribution of the feature value increases or stays the same (not related to other features), the Shapely value will also increase or stay the same.

In order to calculate SHAP, we defined:

$$f_x(S) = E[f(x)|x_S]$$

The figure below explains how we get the predicted value：




First, when S is an empty set, it can be approximated by the average value of the model predictions of the training samples.

$$\phi_0 = f_x(\varnothing) = E[f(x)]$$

Then after we initiated it, we can calculate the value using the following formula.

$$\phi_i = f_x(\{x_1, x_2, \ldots x_i\}) - f_x(\{x_1, x_2, \ldots x_{i\text{-}1}\})$$

However, in practical situations, when the model is nonlinear or the input features are not independent, the SHAP should calculate the weighted average of all possible feature rankings. It is calculated according to the following formula.

$$\phi_j = \sum_{S \subseteq \{x_1, \cdots, x_p\} \setminus \{x_j\}} \frac{|S|!(p - |S| - 1)!}{p!} (f_x(S \cup \{x_j\}) - f_x(S))$$

That is how we calculate SHAP.

But in implementing the SHAP algorithm in COVID-19 dataset. There is a problem prevent me from experimenting it:

In tensorflow 2.0+, the layers in a model can not be hashed so the feed dict of a Session can not be established correctly. It suggest to use experimental_ref ( ) method instead. But here raises another error:

```
Traceback (most recent call last):
  File "shap_inf.py", line 75, in <module>
    shap_values, indexes = shap_explain(shap_dict)
  File "shap_inf.py", line 68, in shap_explain
    map2layer(preprocess_input.copy(), 0, model))
  File "shap_inf.py", line 52, in map2layer
    ret = sess.run(model.layers[layer], feed_dict)
  File "/home/paperspace/.local/lib/python3.5/site-packages/tensorflow_core/python/client/session.py", line 960, in run
    run_metadata_ptr)
  File "/home/paperspace/.local/lib/python3.5/site-packages/tensorflow_core/python/client/session.py", line 1126, in _run
    e.args[0])
TypeError: Cannot interpret feed_dict key as Tensor: Can not convert a Reference into a Tensor.
```

After referring to Tensorflow website, I found that if I use experimental_ref( ) as hash key it will generate a Reference object which is not able to be convert to a Tensor which shut down the process. I have tried several methods including downgrade Tensorflow and Keras version, reimplement the specific part code but it still exists. So what I can do is to understand SHAP calculation process and try my best to explain it more clearly and efficiently!

# Reference

https://arxiv.org/abs/1705.07874

https://arxiv.org/pdf/1602.04938.pdf

https://blog.csdn.net/zkh880loLh3h21AJTH/article/details/78100487?utm_medium=distribute.pc_relevant.none-task-blog-OPENSEARCH-3&depth_1-utm_source=distribute.pc_relevant.none-task-blog-OPENSEARCH-3

https://zhuanlan.zhihu.com/p/83412330

https://www.kaggle.com/dansbecker/shap-values

https://christophm.github.io/interpretable-ml-book/shap.html