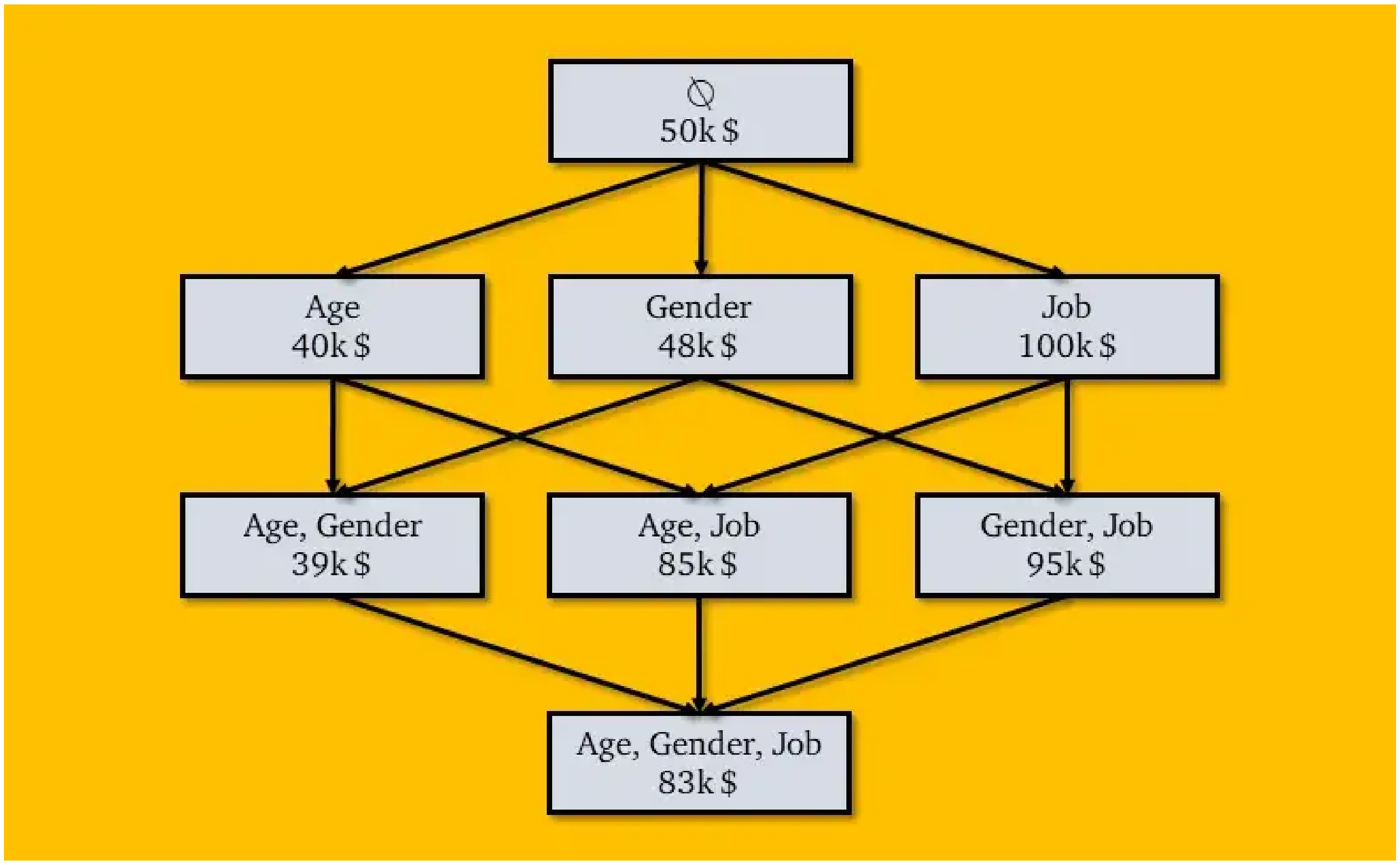
SHAP Values Explained Exactly How You Wished Someone Explained to You

Making sense of the formula used for computing SHAP values



Demystifying the demystifier

SHAP — which stands for SHapley Additive exPlanations — is probably the state of the art in Machine Learning explainability. This algorithm was first published in 2017 by Lundberg and Lee (here is the original paper) and it is a brilliant way to reverse-engineer the output of any predictive algorithm.

In a nutshell, SHAP values are used whenever you have a complex model (could be a gradient boosting, a neural network, or anything that takes some features as input and produces some predictions as output) and you want to understand what decisions the model is making.

> Predictive models answer the "how much". SHAP answers the "why".

In a previous post (Black-Box models are actually more explainable than a Logistic Regression) we used SHAP to understand why a gradient boosting model was suggesting that a Titanic passenger was more or less likely to survive. In other words, we used SHAP to demystify a black-box model. But, so far, we exploited the SHAP library for Python without worrying too much about how it works.

> Ironically enough, we used SHAP as a black-box itself!

However, understanding the idea behind the calculation of SHAP values is crucial to make sense of their outcome. This is why, in this post, we will go through the theoretical foundation of SHapley Additive exPlanations described in the article by Slundberg and Lee, and see why SHAP values are computed the way they are computed.

Game theory and machine learning

SHAP values are based on Shapley values, a concept coming from game theory. But game theory needs at least two things: a game and some players. How does this apply to machine learning explainability? Imagine that we have a predictive model, then:

- **the "game" is reproducing the outcome of the model**,
- **the "players" are the features included in the model**.

What Shapley does is quantifying the contribution that each player brings to the game. What SHAP does is quantifying the contribution that each feature brings to the prediction made by the model.

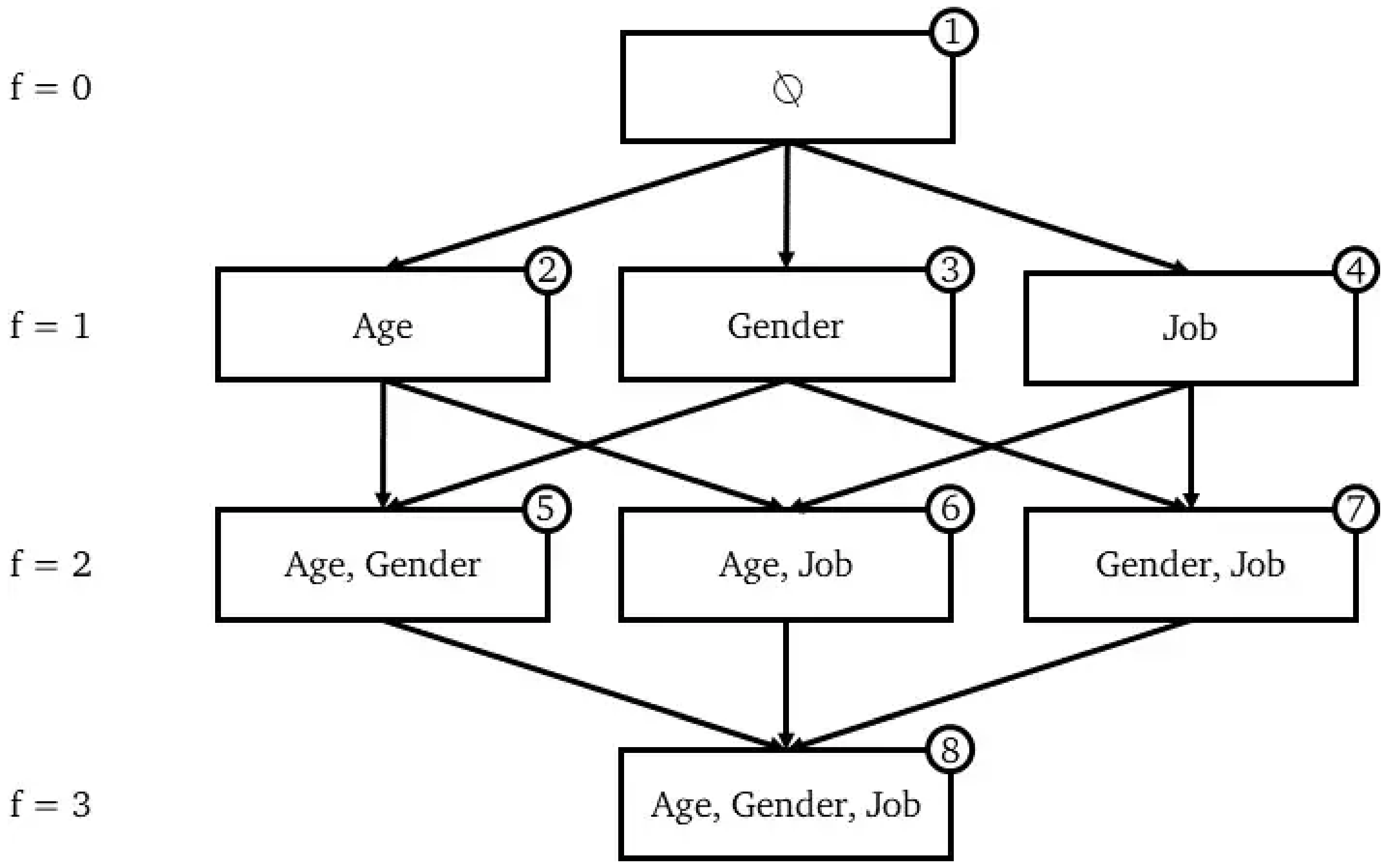
It is important to stress that what we called a "game" concerns a single observation. **One game: one observation**. Indeed, SHAP is about local interpretability of a predictive model.

A power set of features

By way of example, we will imagine a machine learning model (let's say a linear regression, but it could be any other machine learning algorithm) that predicts the income of a person knowing age, gender and job of the person.

Shapley values are based on the idea that the outcome of **each possible combination (or coalition) of players should be considered to determine the importance of a single player**. In our case, this corresponds to each possible combination of *f* features (*f* going from 0 to *F*, *F* being the number of all features available, in our example 3).

In math, this is called a "power set" and can be represented as a tree.



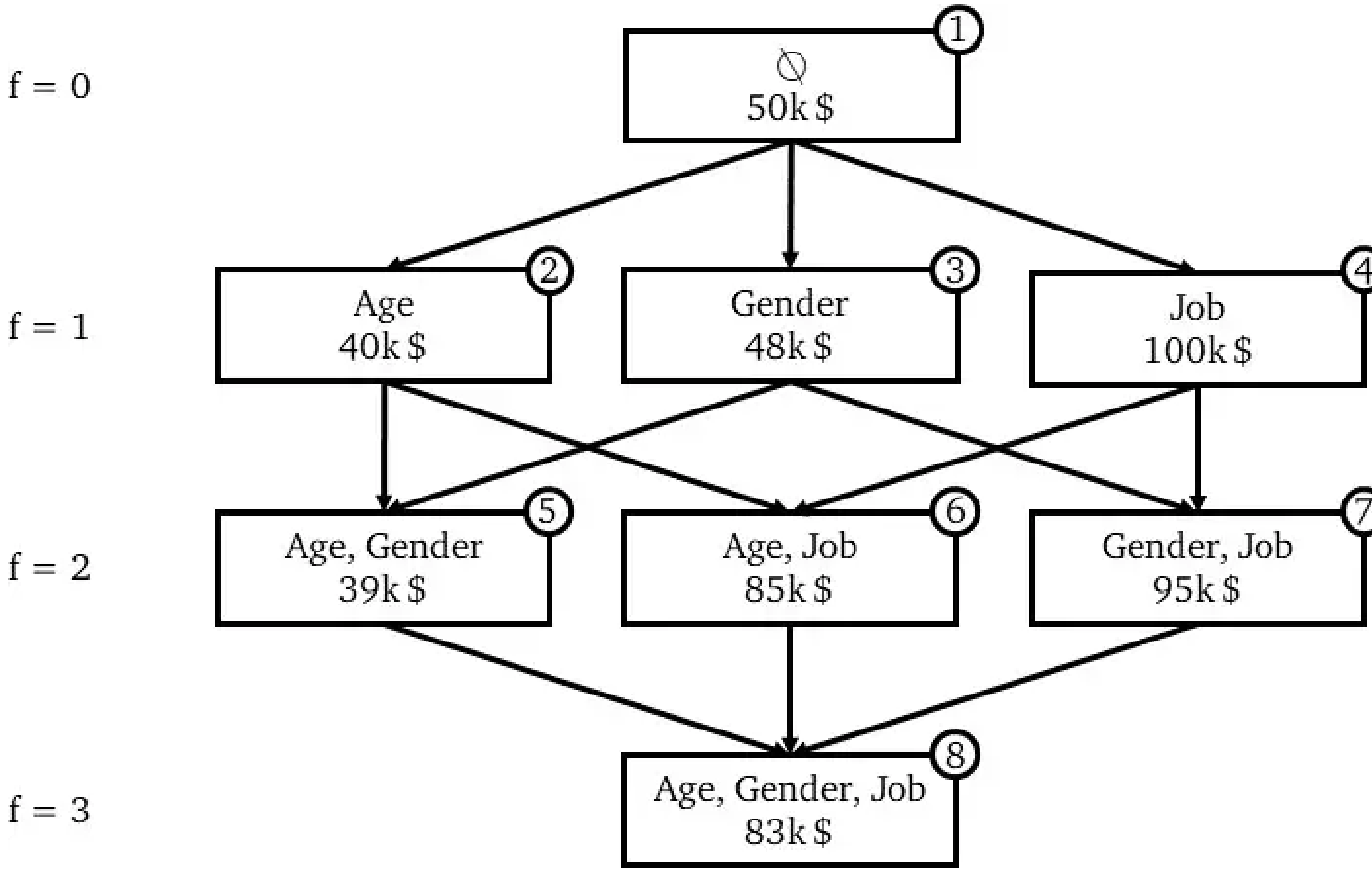
Power set of features

Each node represents a coalition of features. Each edge represents the inclusion of a feature not present in the previous coalition.

We know from math that the cardinality of a power set is 2 ^ *n*, where *n* is the number of elements of the original set. Indeed, in our case, we have 2 ^ *F* = 2 ^ 3 = 8 possible coalitions of features.

Now, **SHAP requires to train a distinct predictive model for each distinct coalition in the power set, meaning 2 ^ *F* models**. Of course, **these models are completely equivalent to each other for what concerns their hyperparameters and their training data (which is the full dataset). The only thing that changes is the set of features included in the model.**

Let us imagine that we have already trained our 8 linear regression models on the same training data. We can then take a new observation (let us call it *xo*) and see what the 8 different models predict for the same observation *xo*.



Predictions made by different models for x_0. In each node, the first row reports the coalition of features included in the model, the second row reports the income predicted for x_0 by that model.

Here, **each node represents a model**. But what do edges represent?

Building SHAP formula (1/2) — Marginal contributions of a feature

As seen above, two nodes connected by an edge differ for just one feature, in the sense that the bottom one has exactly the same features of the upper one plus an additional feature that the upper one did not have. Therefore, **the gap between the predictions of two connected nodes can be imputed to the effect of that additional feature. This is called "marginal contribution" of a feature**.

Therefore, **each edge represents the marginal contribution brought by a feature to a model**.

Imagine that we are in node 1, which is the model with no features. This model will simply predict the average income of all the training observations (50k $). If we move to node 2, which is a model with just one feature (Age), the prediction for *xo* is now 40k $. This means that knowing the age of *xo* has lowered our prediction by 10k $.

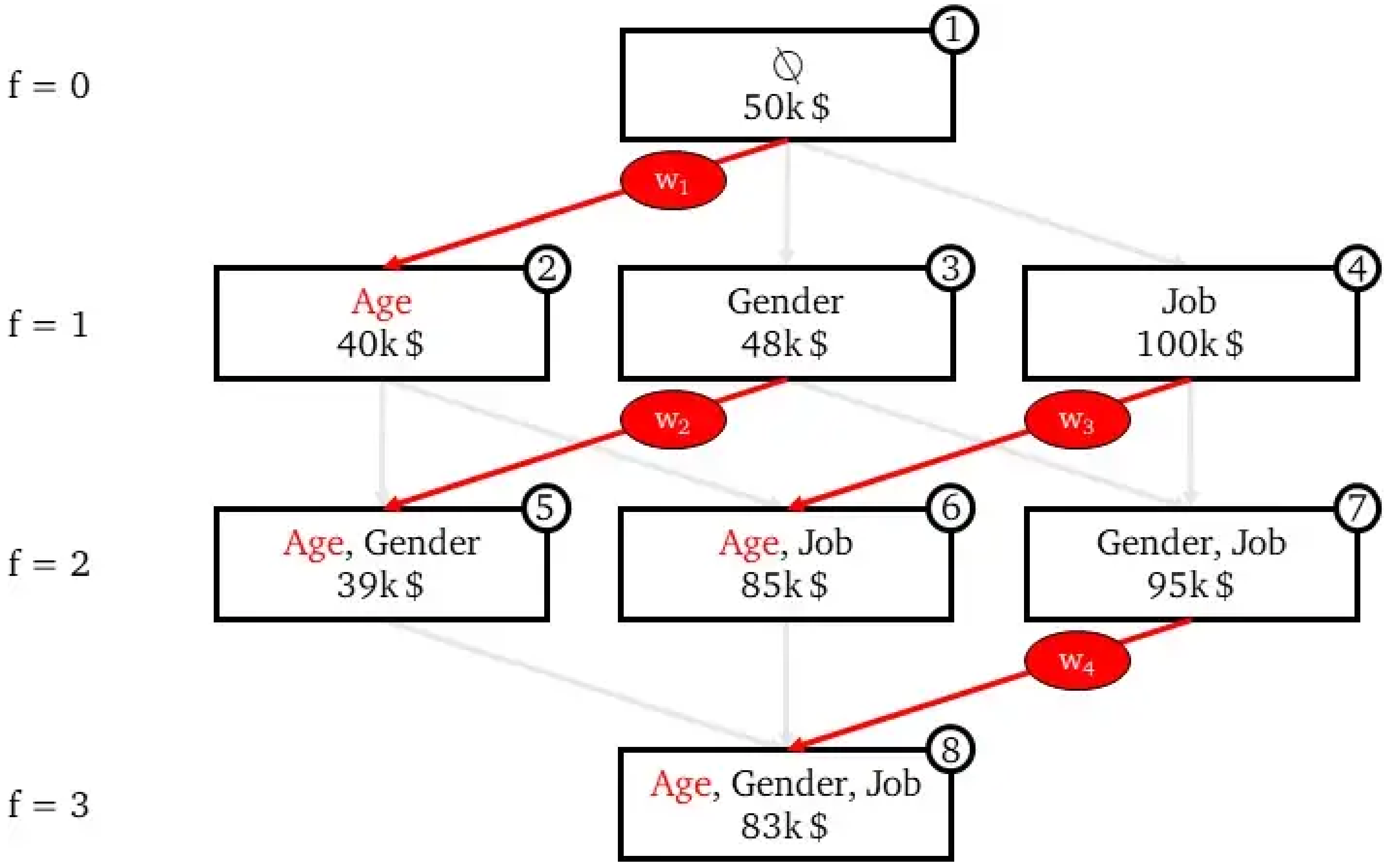
Thus, the marginal contribution brought by Age to the model containing only Age as a feature is -10k $. In formula:

$$MC_{Age,\{Age\}}(x_0) = Predict_{\{Age\}}(x_0) - Predict_{\varnothing}(x_0) = 40k\$ - 50k\$ = -10k\$$$

Of course, to obtain the overall effect of Age on the final model (i.e. the SHAP value of Age for *xo*) it is necessary to consider **the marginal contribution of Age in all the models where Age is present**. In our tree representation, this means to consider all the edges connecting two nodes such that:

- the upper one does not contain Age, and
- the bottom one contains Age.

In the following figure, such edges have been highlighted in red.



Marginal contributions of Age

All these marginal contributions are then aggregated through a weighted average. In formula:

$$
\begin{aligned}
SHAP_{Age}(x_0) = {} & w_1 \times MC_{Age,\{Age\}}(x_0) + \\
& w_2 \times MC_{Age,\{Age,Gender\}}(x_0) + \\
& w_3 \times MC_{Age,\{Age,Job\}}(x_0) + \\
& w_4 \times MC_{Age,\{Age,Gender,Job\}}(x_0)
\end{aligned}
$$

where $w_1 + w_2 + w_3 + w_4 = 1$.

Building SHAP formula (2/2) — Weighing the marginal contributions

But how do we determine the weights of the edges (i.e. of the marginal contiributions of Age in the 4 models)?

The idea is that:

- **the sum of the weights of all the marginal contributions to 1-feature-models should equal the sum of the weights of all the marginal contributions to 2-feature-models and so on…** In other words, the sum of all the weights on the same "row" should equal the sum of all the weights on any other "row".
  In our example, this implies: $w_1 = w_2 + w_3 = w_4$.
- **All the weights of marginal contributions to *f*-feature-models should equal to each other, for each *f*.** In other words, all the edges on the same "row" should equal to each other.
  In our example, this means: $w_2 = w_3$.

Therefore, (keeping in mind that they should sum to 1) the solution is:

- $w_1 = 1/3$
- $w_2 = 1/6$
- $w_3 = 1/6$
- $w_4 = 1/3$

Looking at the figure above, can you guess the pattern for determining weights in a general framework?

Spoiler: the weight of an edge is the reciprocal of the total number of edges in the same "row". Or, equivalently, **the weight of a marginal contribution to a *f*-feature-model is the reciprocal of the number of possible marginal contributions to all the *f*-feature-models**.

Is there a formula to calculate this? It is straightforward, actually.

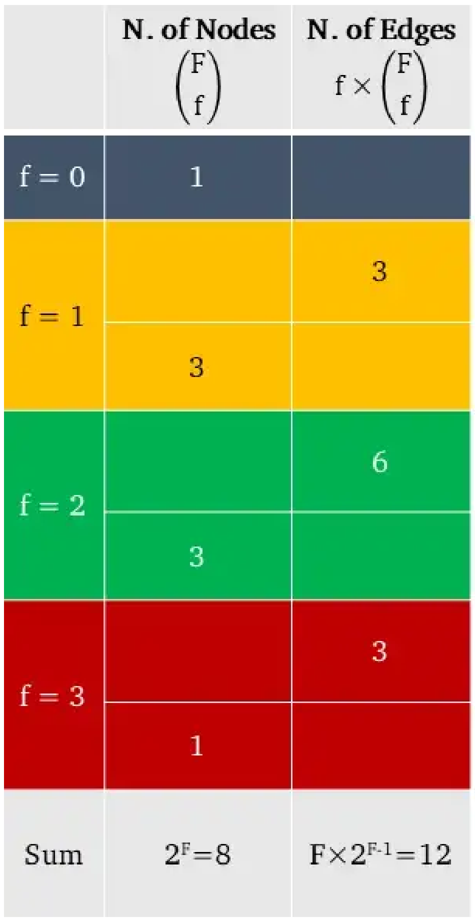
Each *f*-feature-model has *f* marginal contributions (one per feature), so it is enough to count the number of possible *f*-feature-models and to multiply it by *f*. Thus, the problem boils down to counting the number of possible *f*-feature-models, given *f* and knowing that the total number of feature is *F*. This is simply the definition of binomial coefficient.

Putting things together, we have that the number of all the marginal contributions of all the *f*-feature-models — in other words, the number of edges in each "row" — is:

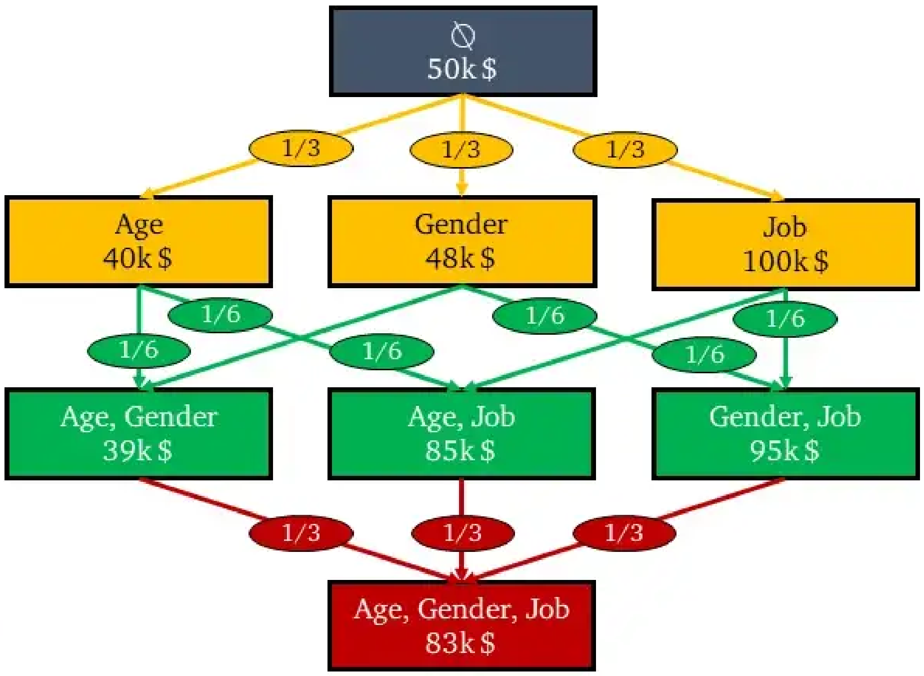
$$f \times \binom{F}{f}$$

It is enough to take the reciprocal of this and we have the weight of a marginal contribution to a *f*-feature-model.

This is exemplified in the figure below:



	N. of Nodes $\binom{F}{f}$	N. of Edges $f \times \binom{F}{f}$
f = 0	1	
f = 1		3
	3	
f = 2		6
	3	
f = 3		3
	1	
Sum	$2^F=8$	$F\times2^{F-1}=12$



How weigths are obtained from the number of edges

Now, we have all the elements required for calculating the SHAP value of Age for *xo*:

$$\begin{aligned} SHAP_{Age}(x_0) = & [(1\times\binom{3}{1})]^{-1}\times MC_{Age,\{Age\}}(x_0)\ + \\ & [(2\times\binom{3}{2})]^{-1}\times MC_{Age,\{Age,Gender\}}(x_0)\ + \\ & [(2\times\binom{3}{2})]^{-1}\times MC_{Age,\{Age,Job\}}(x_0)\ + \\ & [(3\times\binom{3}{3})]^{-1}\times MC_{Age,\{Age,Gender,Job\}}(x_0)\ + \\ = & \frac{1}{3}\times(-10k\$)+\frac{1}{6}\times(-9k\$)+\frac{1}{6}\times(-15k\$)+\frac{1}{3}\times(-12k\$) \\ = & -11.33k\$ \end{aligned}$$

Wrapping it up

We have built the formula for calculating the SHAP value of Age in a 3-feature-model. Generalizing to any feature and any *F*, we obtain the formula reported in the article by Slundberg and Lee:

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

Applied on our example, the formula yields:

- SHAP_Age(*xo*) = -11.33k $
- SHAP_Gender(*xo*) = -2.33k $
- SHAP_Job(*xo*) = +46.66k $

Summing them up gives +33k $, which is exactly the difference between the output of the full model (83k $) and the output of the dummy model with no features (50k $).

This is a fundamental characteristic of SHAP values: **summing the SHAP values of each feature of a given observation yields the difference between the prediction of the model and the null model** (or its logistic function, as we have seen here). This is actually the reason for their name: SHapley Additive exPlanations.

Great! Now I can calculate SHAP values by myself

Err… No!

As seen above, the original SHAP formula requires to train 2 ^ *F* models. For a model with just 50 features, this would mean to train 1e15 models! Indeed, as *F* increases, the formula seen above becomes inapplicable soon.

However, libraries such as the one by Slundberg employ some brilliant approximations and samplings (I will treat the topic in a follow-up post) that make the job feasible.

Machine Learning Shap Explainable Ai Explainability Deep Learning