Predictive Modeling SHAP and CatBoost

DS Development Presentations

Peter Nicholas S. Onglao | MNL

Contents

1

SHAP: Interpretable ML

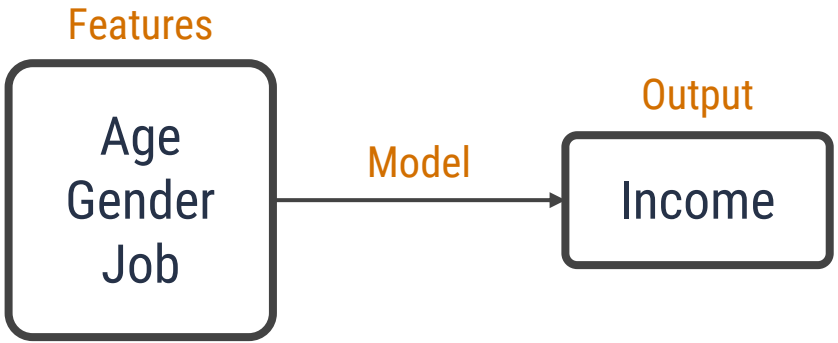
SHapley Additive exPlanations

Shapley Values

- Introduced by Lloyd Shapley in 1951
- Part of cooperative game theory
  - For each **game**, we fairly distribute the **payout** among **players** depending on their contributions.
    - **game** = **prediction instance**
    - **payout** = **model output**
    - **players** = **features**
  - For each **prediction instance**, we fairly distribute the **model output** among **features** depending on their contributions.

Shapley Values: Computation

- Consider the following:

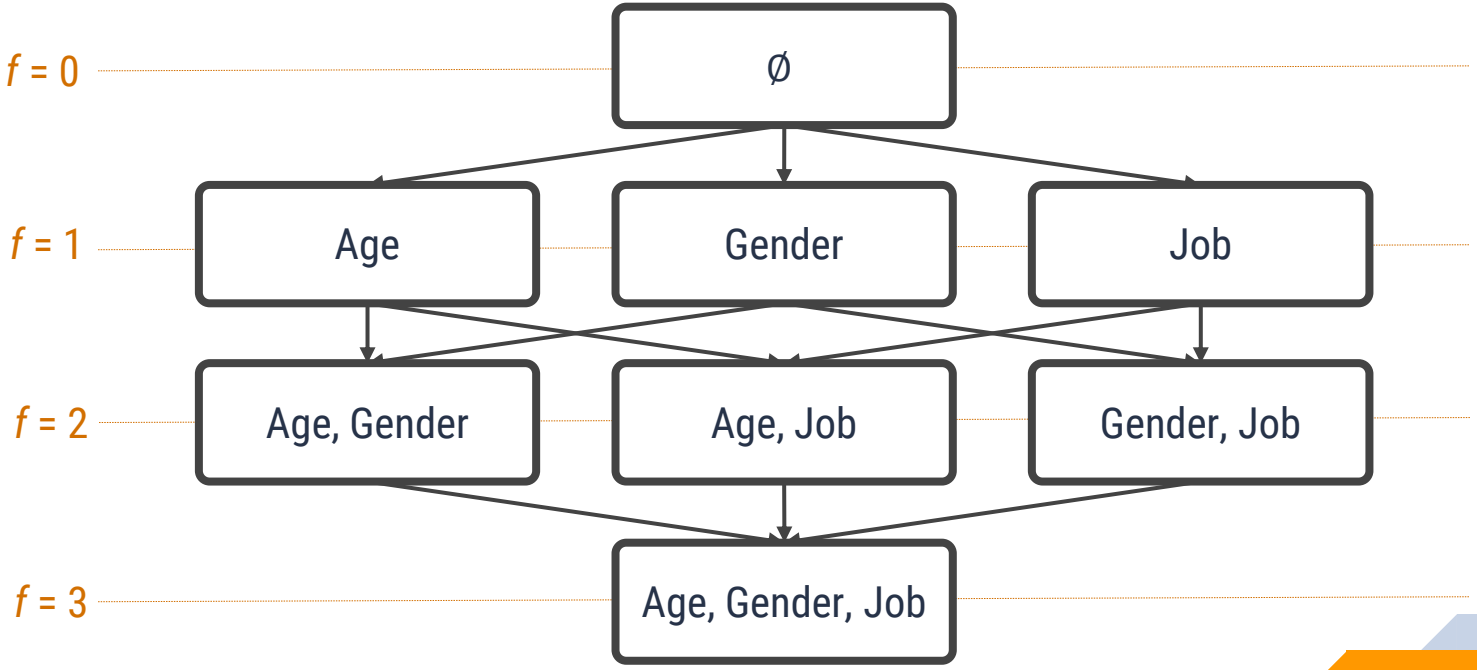


Shapley Values: Computation

- The weighted sum of all marginal contributions of a certain feature.
- We need to consider all possible combinations ('coalitions') of features to determine marginal contributions.
  - Power set: a possible combinations of *f* features
  - If we have F total features, there are 2^F total coalitions

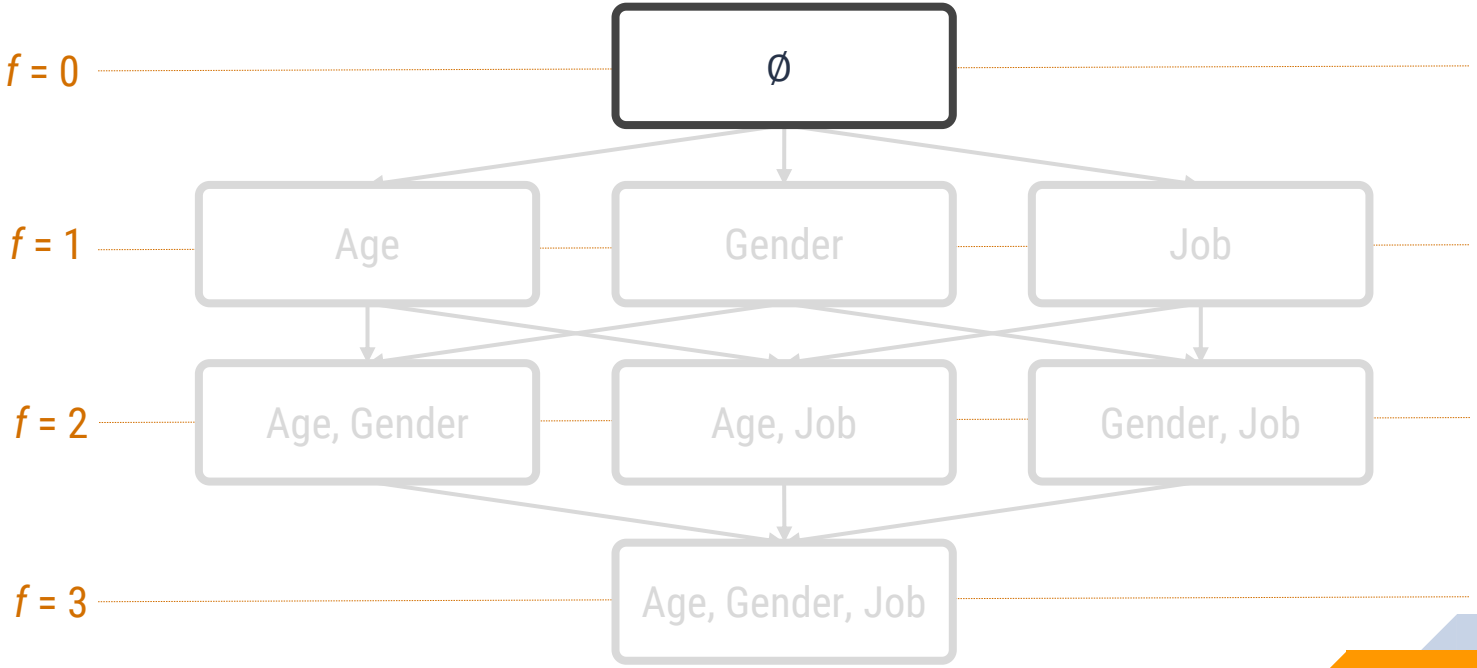
Shapley Values: Computation

Try out each possible combination of *f* features (power set). $2^F = 2^3 = 8$



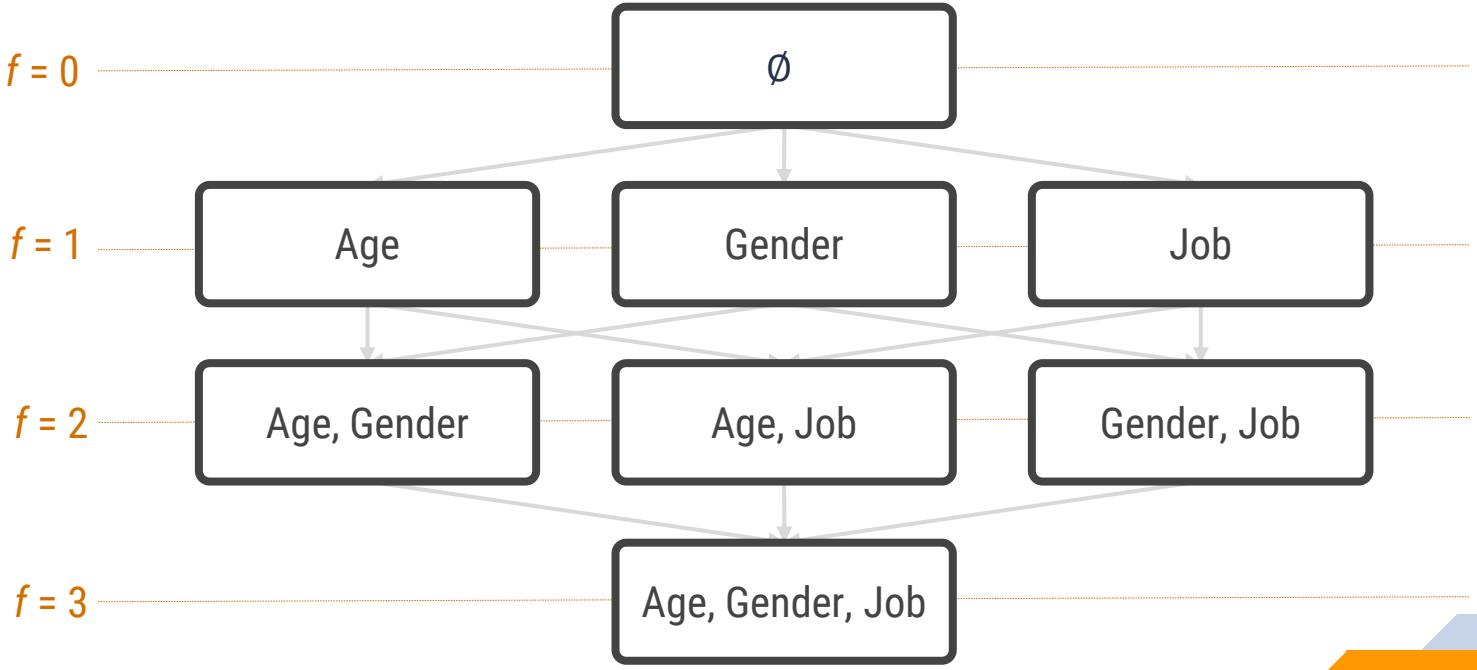
Shapley Values: Computation

Each node is a model



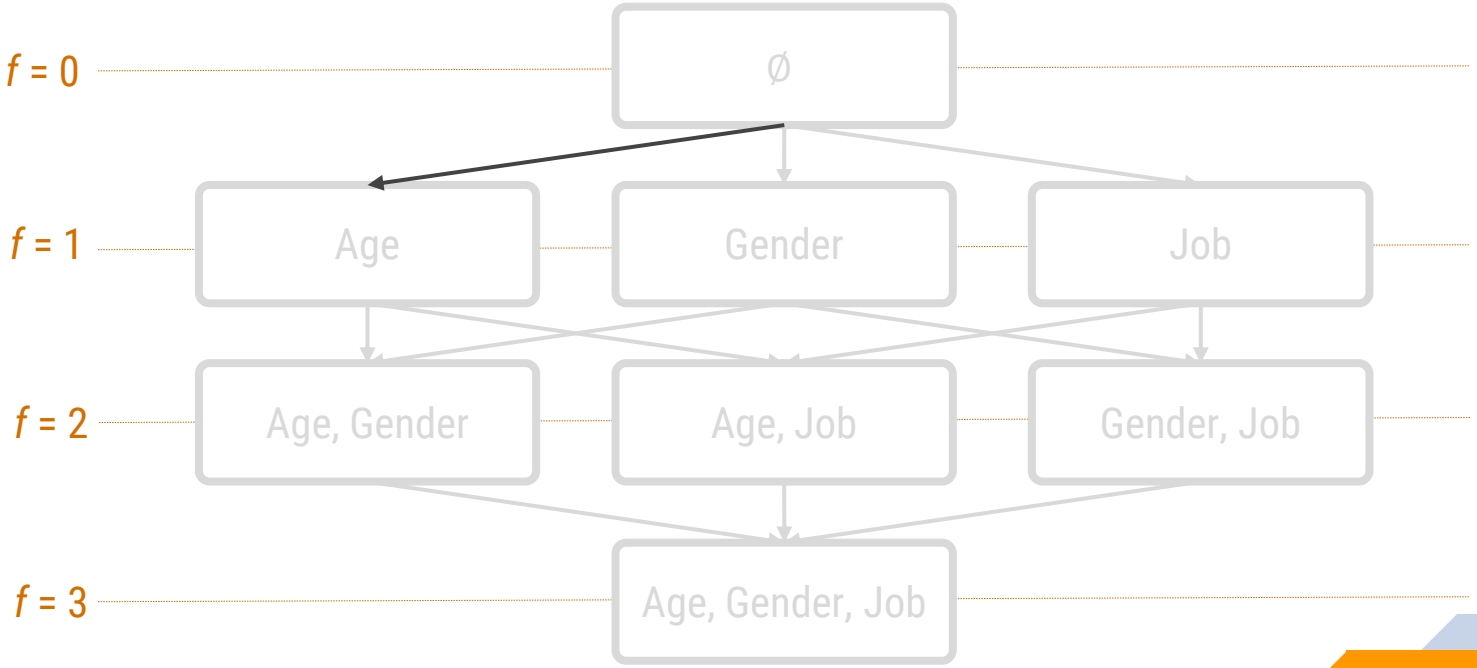
Shapley Values: Computation

Each model uses the same hyperparameters and training data, only the feature set changes



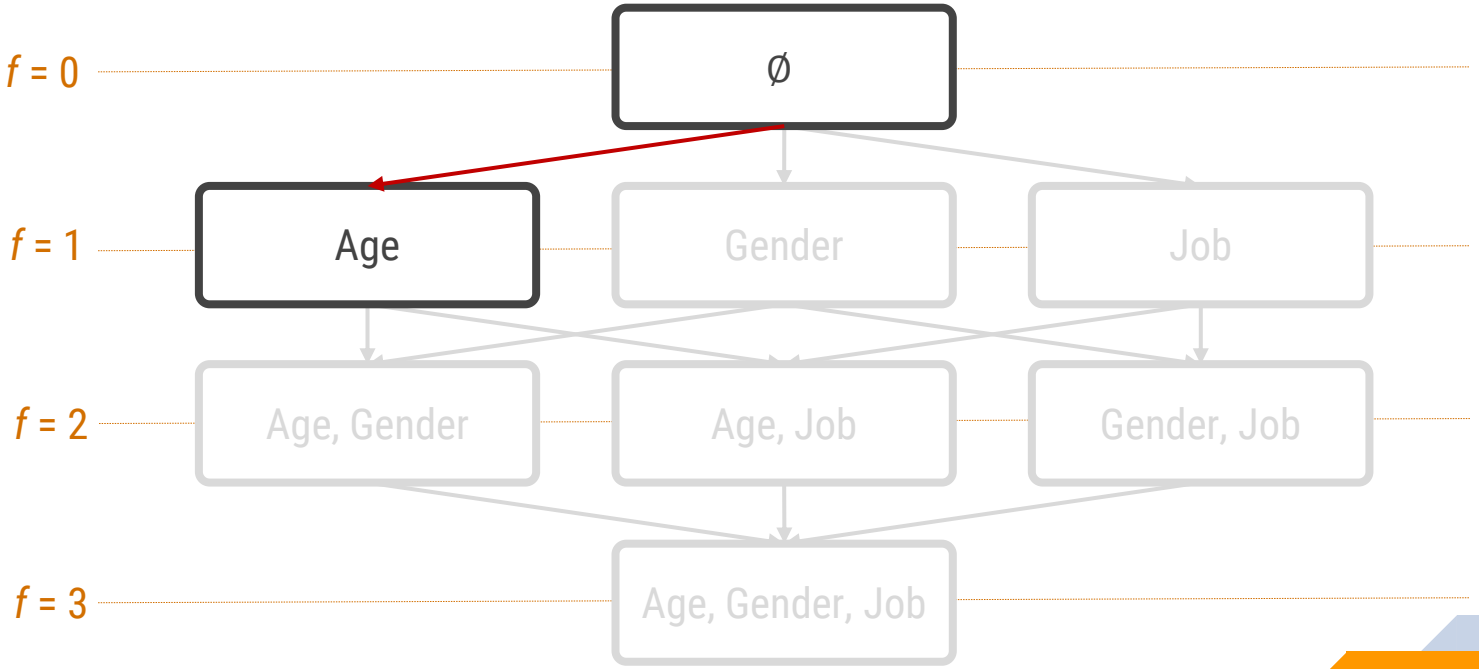
Shapley Values: Computation

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



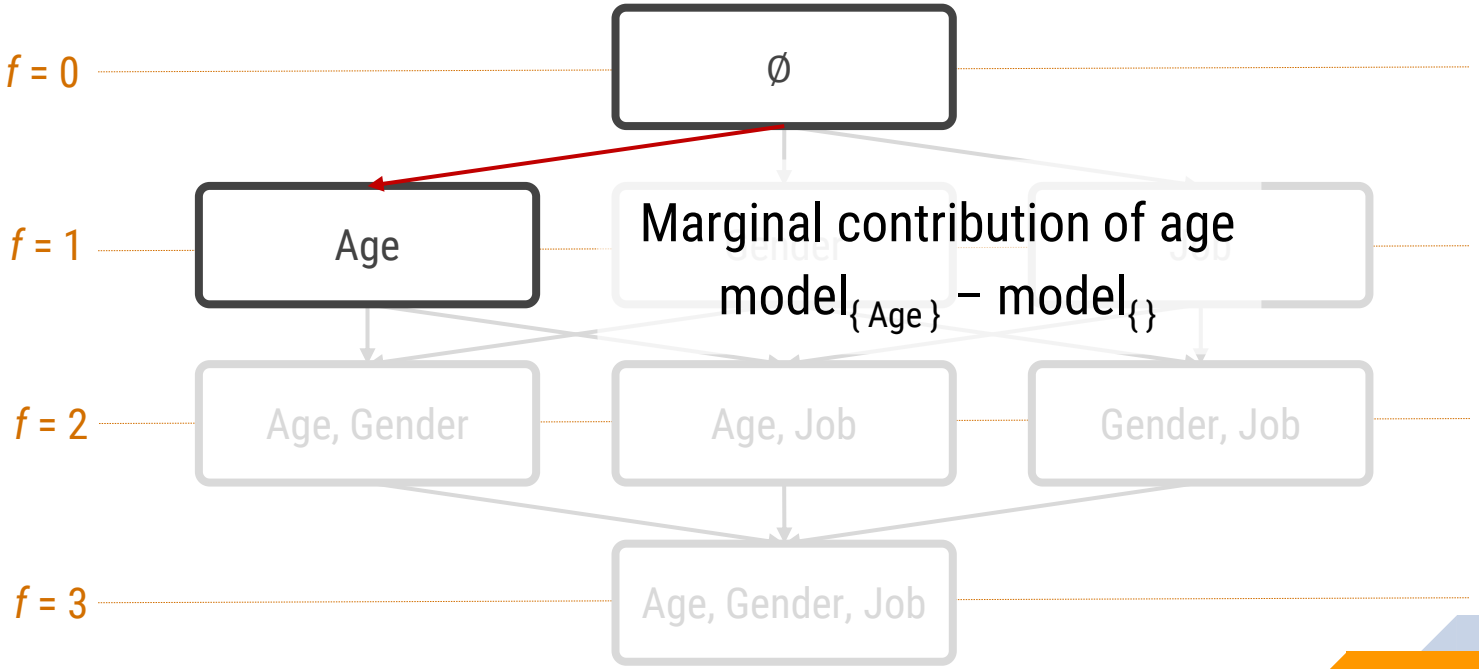
Shapley Values: Computation

The difference between two connected nodes is a marginal contribution



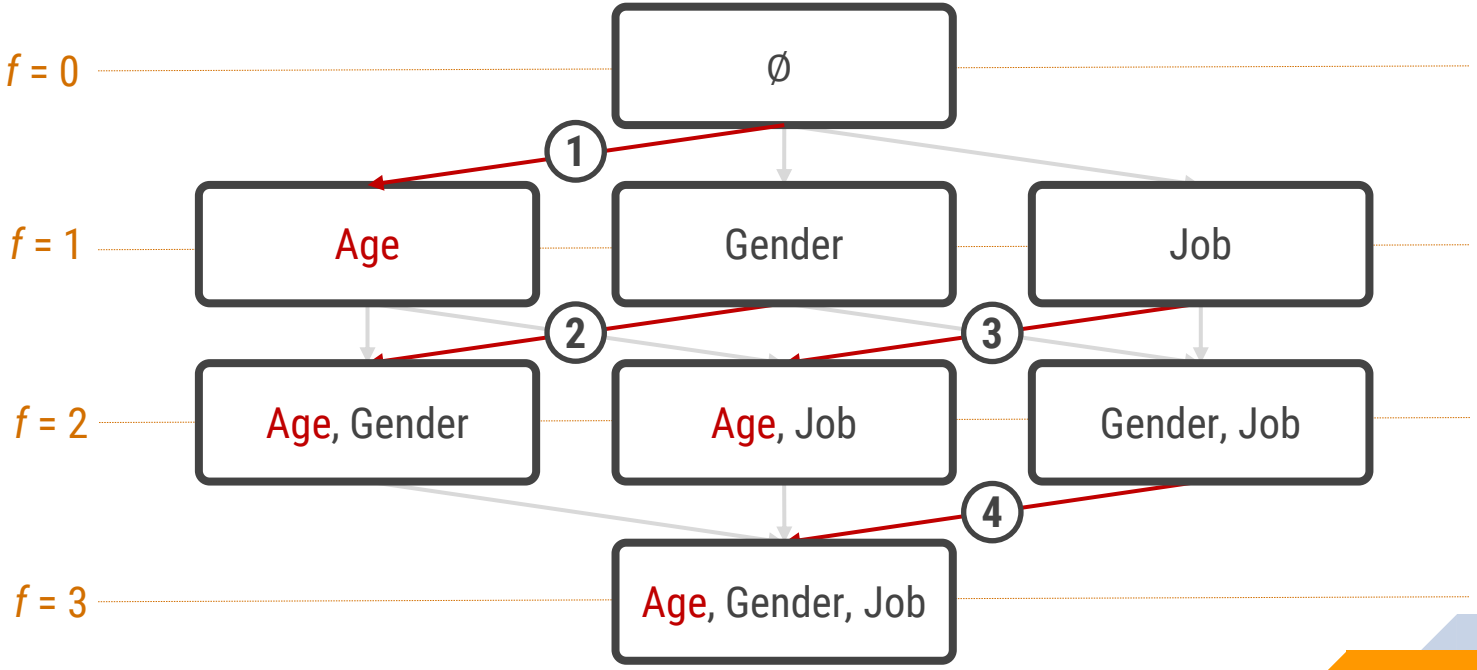
Shapley Values: Computation

The difference between two connected nodes is a marginal contribution



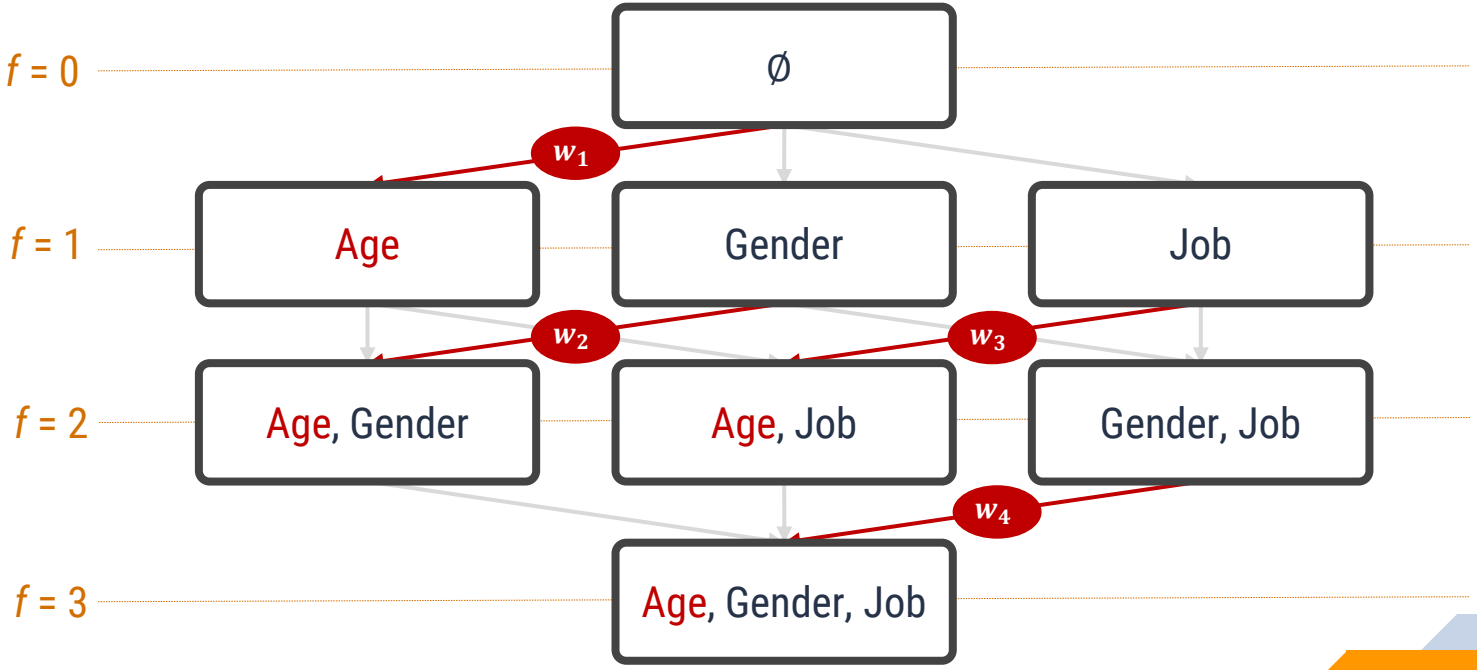
Shapley Values: Computation

We have to get the weighted sum of all marginal contributions



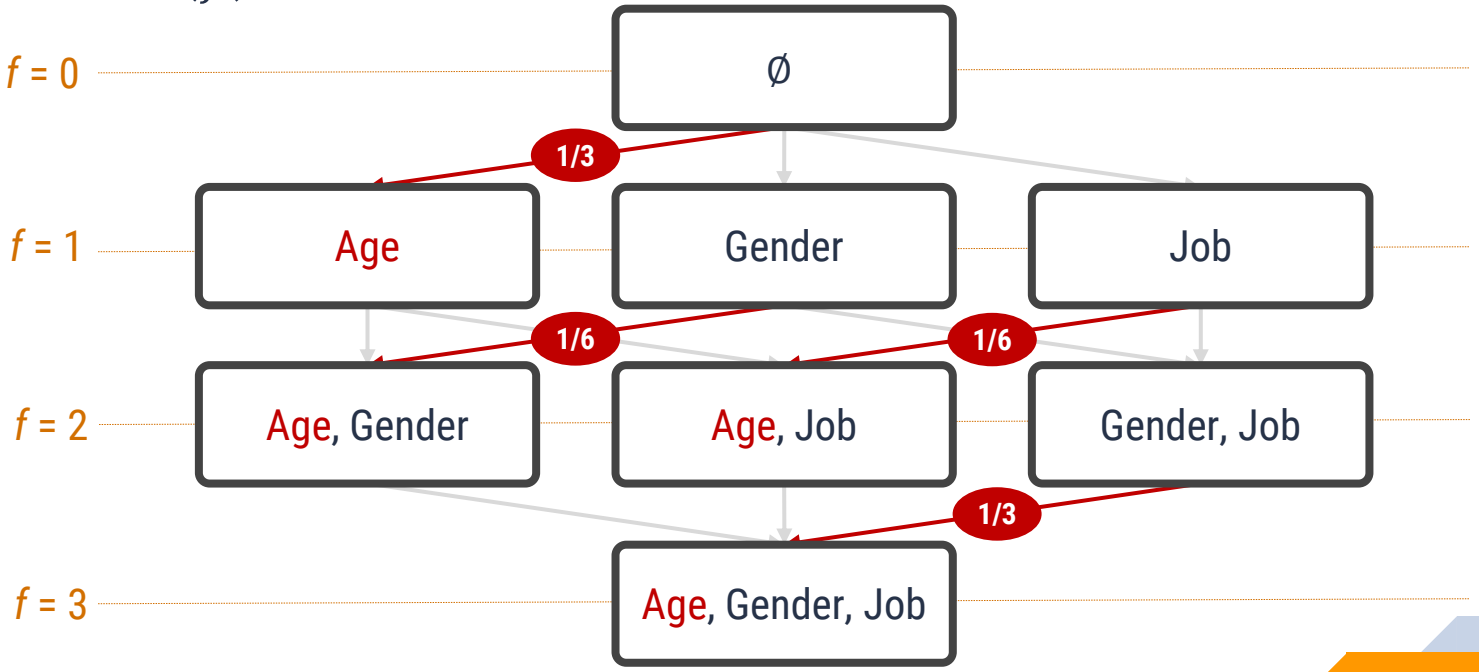
Shapley Values: Computation

What weights should we use?



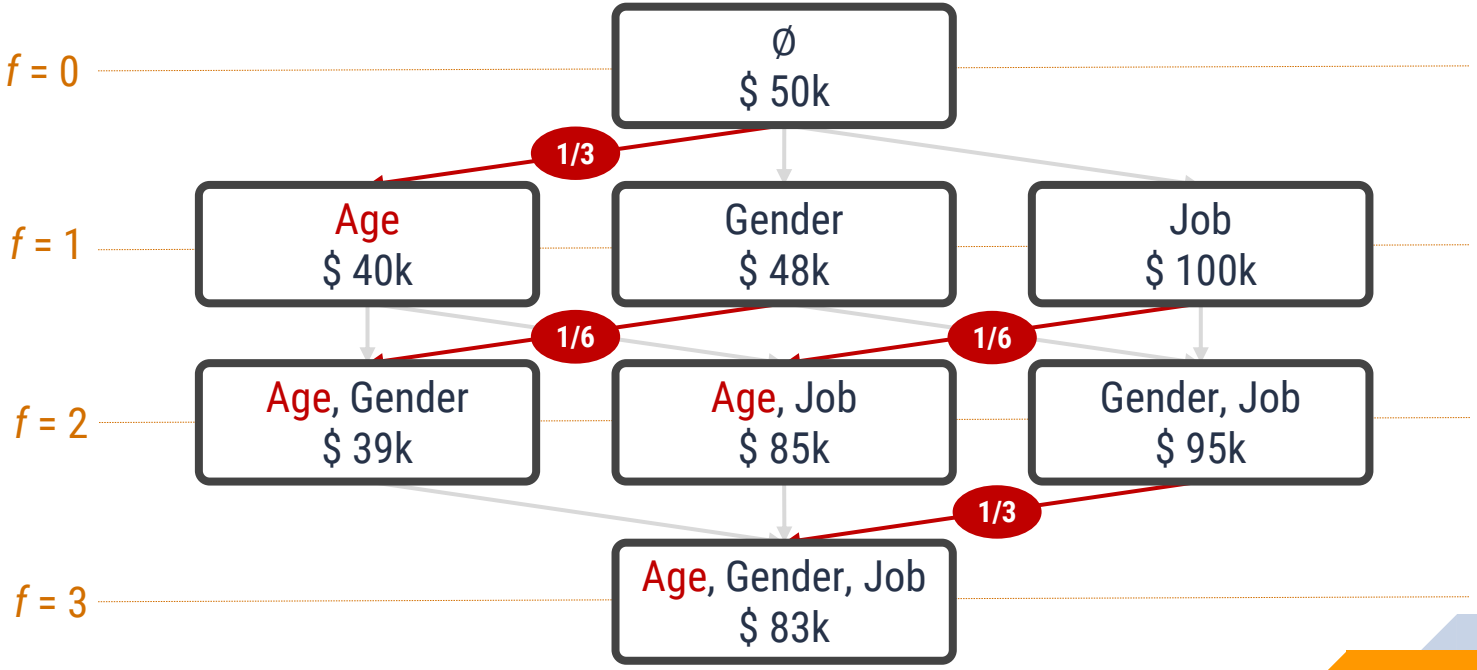
Shapley Values: Computation

$$f \times \binom{F}{f} = \text{the reciprocal of how many edges are going to that level}$$



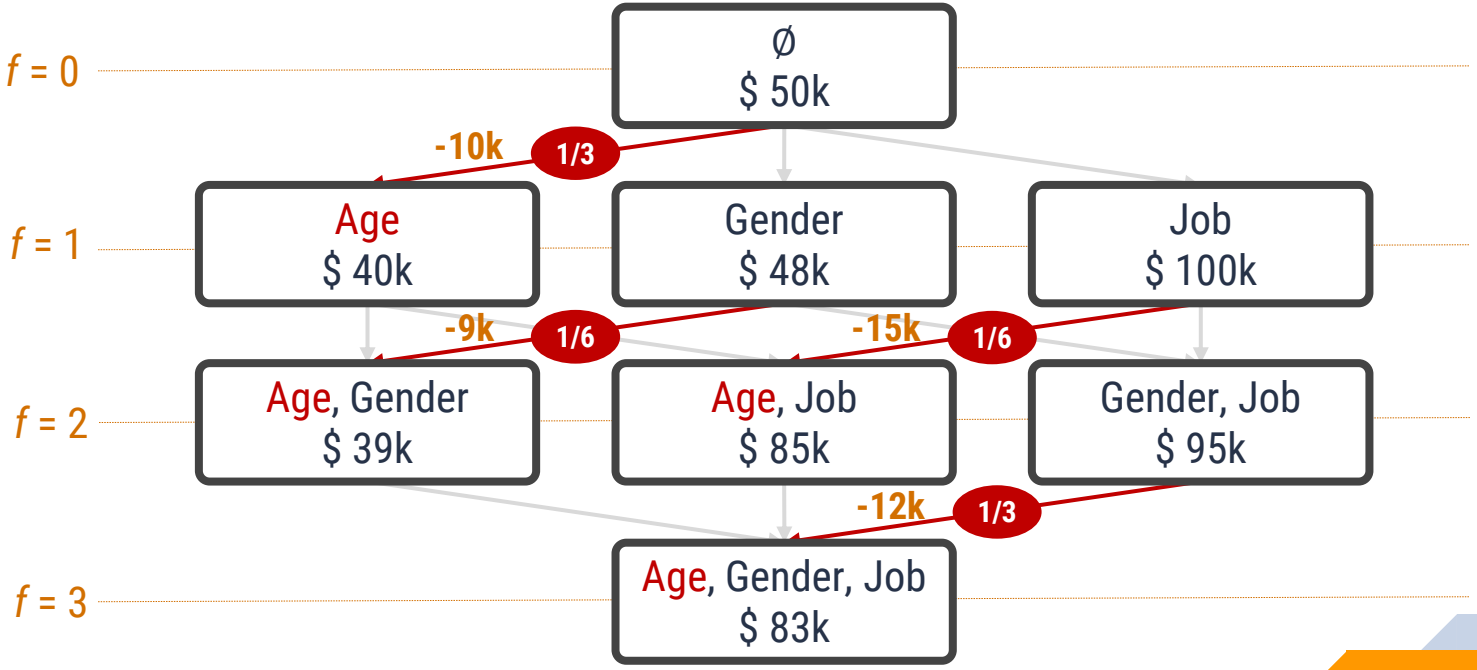
Shapley Values: Computation

Let's add sample values. Note that this is for <u>one prediction instance</u>, say x_0.



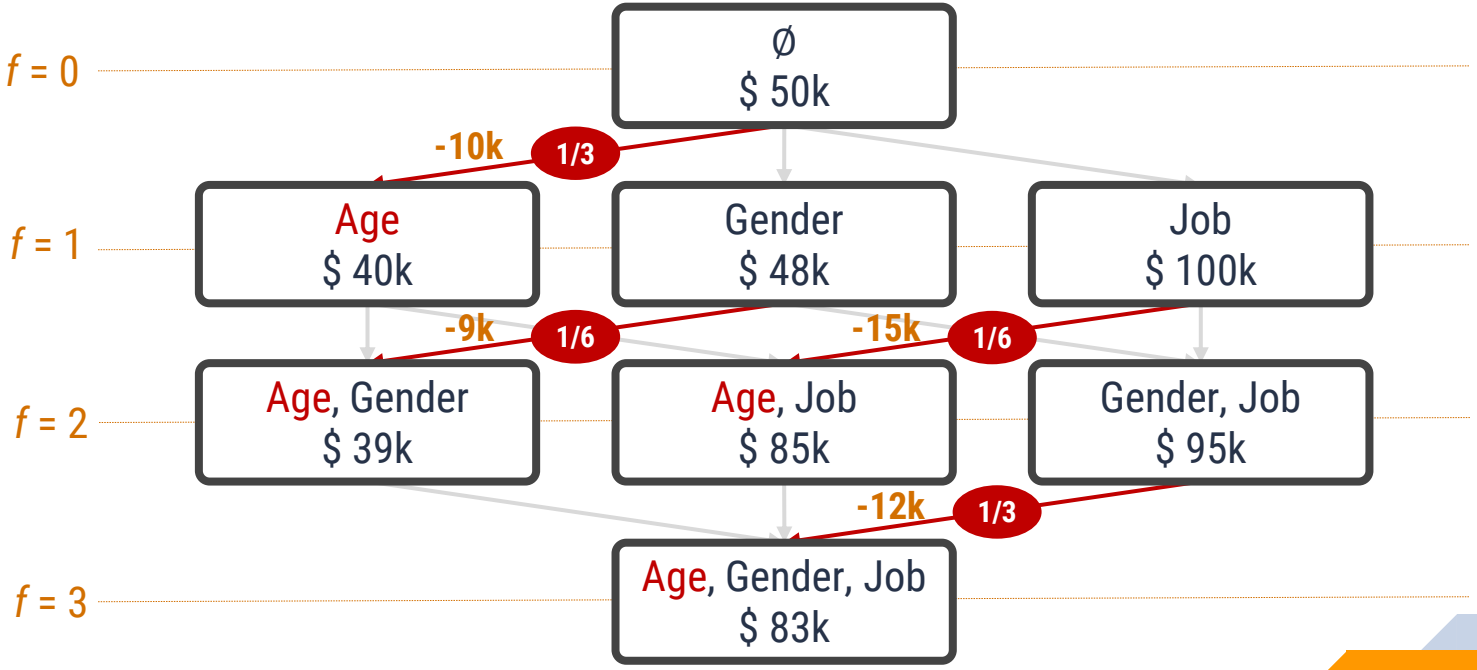
Shapley Values: Computation

Let's calculate marginal contributions



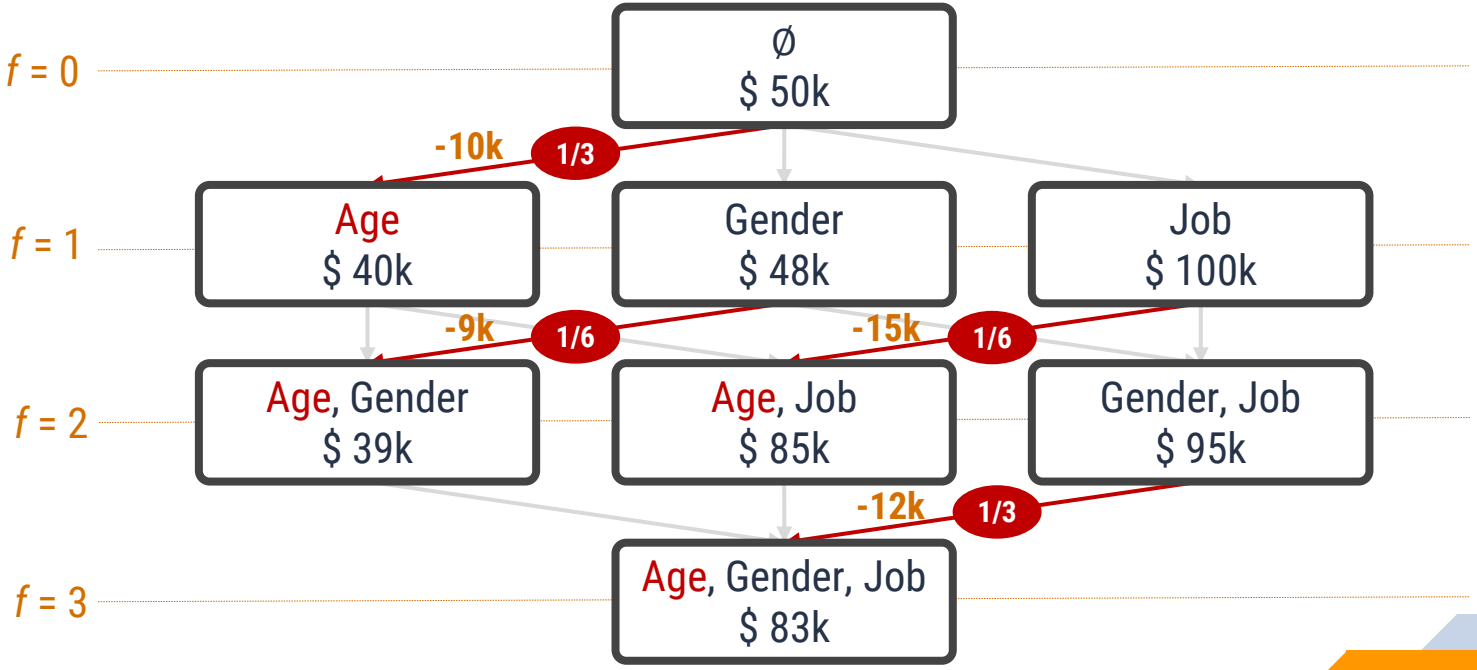
Shapley Values: Computation

$$\mathrm{SHAP}_{Age}(x_0) = {}^{1}/_{3} \cdot (-10\mathrm{k}) + {}^{1}/_{6} \cdot (-9\mathrm{k}) + {}^{1}/_{6} \cdot (-15\mathrm{k}) + {}^{1}/_{3} \cdot (-12\mathrm{k})$$



Shapley Values: Computation

$$\mathrm{SHAP}_{Age}(x_0) = -11.33\mathrm{k}$$



Shapley Values: Computation

- The formula:

$$\mathrm{SHAP}_{feat}(x) = \sum \left[|S| \times \binom{F}{|S|} \right]^{-1} \left[predict_S(x) - predict_{S \setminus feat}(x) \right]$$

where S: $feat \in S$

Shapley Values: Properties

- Efficiency
  - Feature contributions should add up for every prediction
- Symmetry
  - Features that contribute equally have the same Shapley values
- Dummy
  - Features that have no contribution have a Shapley value of 0
- Additivity
  - If you add models together, you can also add their Shapley values
  - e.g. for a random forest, the prediction is an average of many trees, and the Shapley values of the random forest are averages of the Shapley values for each tree

Shapley Values: Advantages

- Model agnostic
  - Will work for any blackbox model that takes in an input and gives an output
- Fair distribution
  - Guarantees the prediction is fairly distributed among features
- Grounded on theory
  - Several properties (efficiency, symmetry, dummy, additivity) give a reasonable foundation, unlike other methods such as LIME

LIME:
Local surrogate model. Approximates feature contributions based on local approximations of your model.

Shapley Values: Disadvantages

- Computationally expensive
  - Exact computation needs 2^F coalitions of features
  - In almost all problems, only the approximate solution is feasible
- Can be misinterpreted
  - Wrong: The Shapley value is the difference of the predicted value after removing the feature from the model training
  - Correct: Given the current set of feature values, the contribution of a feature value to the difference between the actual prediction and the mean prediction is the estimated Shapley value
- Always uses all features
  - Does not offer a sparse explanation

Shapley Values: Takeaways

- Usually, as model complexity increases, model interpretability decreases (e.g. neural networks vs. logistic regression)
- Shapley values help us better interpret black box models that were considered before as “non interpretable”
  - We can balance accuracy and interpretability
  - Yes, we can calculate them for neural networks, boosted tree methods, or any input-output model
- Shapley values are only for model interpretability; as of now we can’t make statistical inferences
- SHAP = local feature importance (on a per instance basis)

2

CatBoost Continuation

Tuning + Model Interpretation

Highlights

- Encoding Categorical Features
- Pool
- Parameter Tuning
- Feature Importances

Sample Code

Data: Heart Disease

1. `age` (int)
2. `sex` (cat)
   0 = femaile, 1 = male
3. `cp` (cat) - chest pain type
   0 = asymptomatic, 1 = atypical angina, 2 = non-anginal pain, 3 = typical angina
4. `trestbps` (int) - resting blood pressure
5. `chol` (int) - blood serum cholesterol
6. `fbs` (cat) - if fasting blood sugar is > 120 mg/dl
   0 = False, 1 = True
7. `restecg` (cat) - resting electrocardiographic results
   0 = probable/definite left ventricular hypertrophy, 1 = normal, 2 = having ST-T wave abnormality
8. `thalach` (int) - maximum heart rate achieved
9. `exang` (cat) - exercise induced angina
   0 = no, 1 = yes
10. `oldpeak` (float) - ST depression (a finding on an electrocardiogram) induced by exercise relative to rest
11. `slope` (cat) - slope of the peak exercise ST segment
    0 = downsloping, 1 = flat, 2 = upsloping
12. `ca` (int) - number of major vessels colored by fluoroscopy
13. `thal` (cat) - blood flow measurement using thallium
    0 = null, 1 = fixed defect, 2 = normal, 3 = reversible defect
14. `target` (cat) - **indicator of NO heart disease**
    0 = disease, 1 = no disease

	Count	Percent
1	165	54.46%
0	138	45.54%

Sample Code

Get numeric indices of the categorical features

In [10]:
```
cat_features = ['sex','cp','fbs','restecg','exang','slope', 'thal'] #list down t
cat_features = [i for i,item in enumerate(X.columns) if item in cat_features] #g
cat_features
```

Out[10]:
```
[1, 2, 5, 6, 8, 10, 12]
```

Sample Code

- The Pool function
  - Groups together X, y, cat feature indices, and other details into one object

```
for var in ['train','test']:
    globals()[var + '_pool'] = Pool(data = globals()['X_'+var],
                                    label = globals()['y_'+var],
                                    cat_features = cat_features)
```

Sample Code

CatBoostClassifier

```
In [12]: model = CatBoostClassifier(random_state=1010)
```

Built-in randomized search function (and grid search)

```
In [23]: model = CatBoostClassifier(random_state = 1010) #we need a model that has not been fitted
         np.random.seed(seed=1010)
         randomized_search_result = model.randomized_search(param_distributions = grid,
                                                            X=train_pool,
                                                            cv = 5,
                                                            n_iter = 100,
                                                            partition_random_seed = 1010,
                                                            search_by_train_test_split = True,
                                                            plot = True)
```

Sample Code

Parameter Tuning

- `iterations`
- `learning_rate`
- `depth` - tree depth
- `one_hot_max_size` - perform one hot encoding if categores ≤ `one_hot_max_size`.
- `l2_leaf_reg` - coefficient of L2 regularization term, positive number
- `random_strength` - use to avoid overfitting, the amount of randomness in scoring splits
- `bagging_temperature` - [0,inf). For bayesian bootstrap.

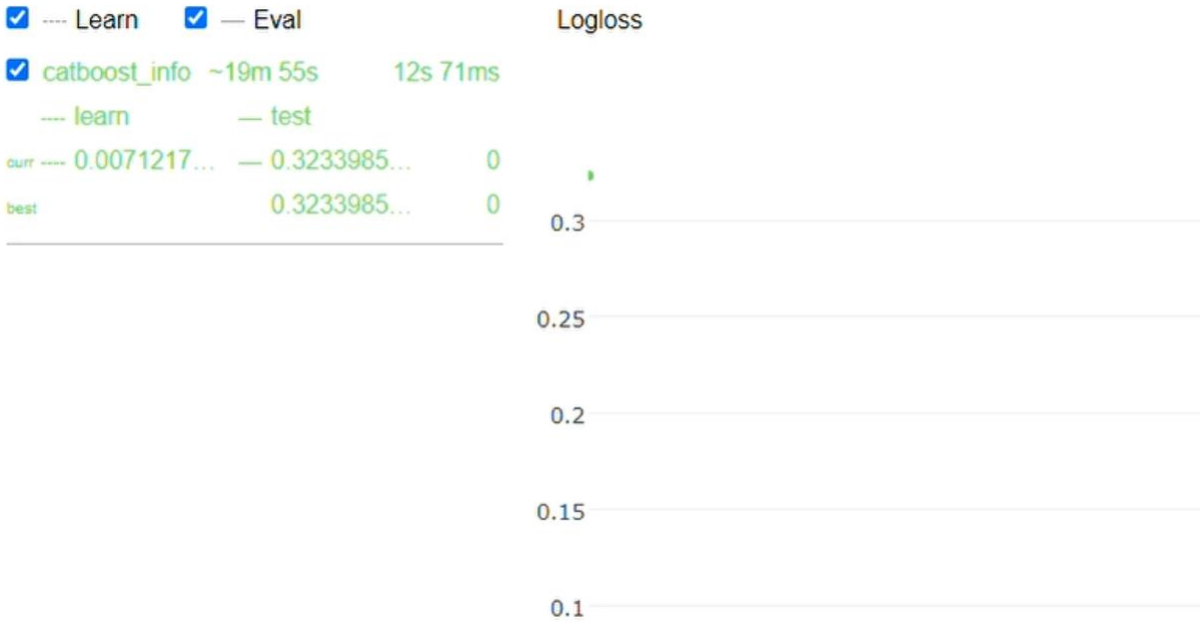
```
In [21]: grid = {'learning_rate': stats.uniform(0.03,0.47),
                 'depth': stats.randint(4,10),
                 'l2_leaf_reg': stats.uniform(1,9),
                 'one_hot_max_size': stats.randint(2,6),
                 'random_strength': stats.uniform(0,10),
                 'bagging_temperature': stats.uniform(0,10)}
```

```
In [134]: model.get_params()

Out[134]: {'random_state': 1010,
           'cat_features': [1, 2, 5, 6, 8, 10, 12],
           'bagging_temperature': 1.7901291717351375,
           'random_strength': 2.2923825925049677,
           'depth': 5.0,
           'one_hot_max_size': 2.0,
           'learning_rate': 0.35204250884248744,
           'l2_leaf_reg': 1.0542193946184995}
```

Sample Code

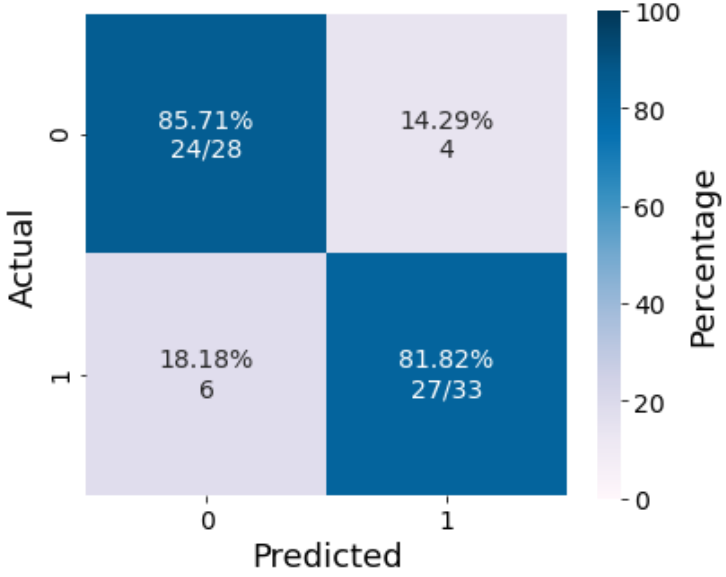
- Plot = True: time estimate + interactive real-time plot



Sample Code

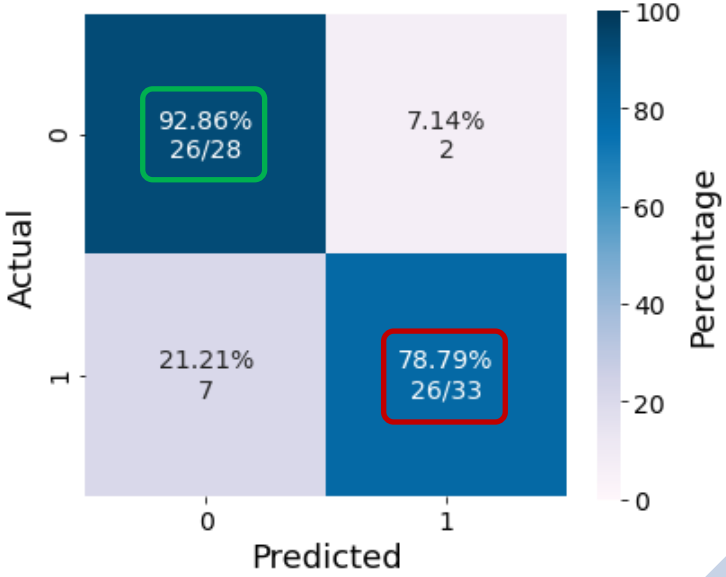
Results of parameter tuning

Default



Bal. acc = 0.838

Randomized Search



Bal. acc = 0.858

Sample Code

- Feature Importances
  - Only one function: model.get_feature_importance(type = _)

1. `'PredictionValuesChange'`

   > Default for non-ranking metrics. Shows how much on average the prediction changes if the feature value changes. The bigger the value of the importance the bigger on average is the change to the prediction value, if this feature is changed.

2. `'LossFunctionChange'`

   > Default for ranking metrics. For each feature the value represents the difference between the loss value of the model with this feature and without it. Approximate only (values are dataset-dependent)

3. `'FeatureImportance'`

   > PredictionValuesChange for non-ranking metrics and LossFunctionChange for ranking metrics

Sample Code

- Feature Importances
  - Only one function: model.get_feature_importance(type = _)

Advanced - more advanced feature analysis (recommended)

4. `'ShapValues'`

> Get SHAP Values for every feature. Generates a vector with contributions of each feature to the prediction for every input object and the expected value of the model prediction for the object (average prediction given no knowledge about the object).

5. `'Interaction'`

> Get pairwise interaction strengths between features.

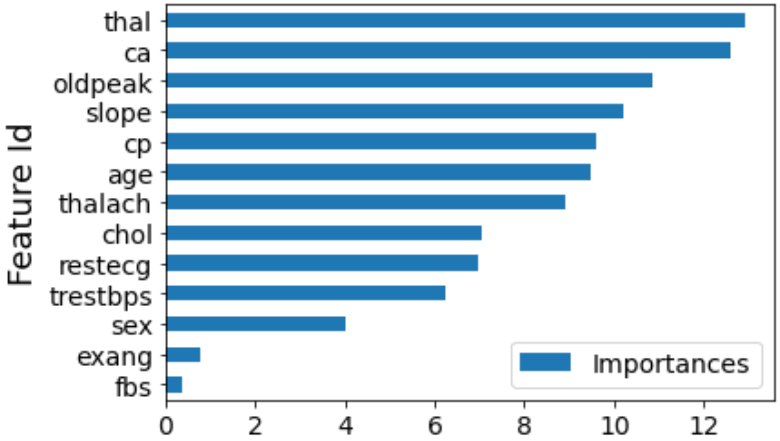
Sample Code

Vanilla feature importances

In [27]:
```
fi_values = model.get_feature_importance(train_pool, type = "FeatureImportance",
fi_values
```

Out[27]:

	Feature Id	Importances
0	thal	12.906382
1	ca	12.610759
2	oldpeak	10.874527
3	slope	10.193008
4	cp	9.590038
5	age	9.460423
6	thalach	8.914464
7	chol	7.047519
8	restecg	6.956936
9	trestbps	6.260252
10	sex	4.009231
11	exang	0.785320
12	fbs	0.391141



Sample Code

SHAP values

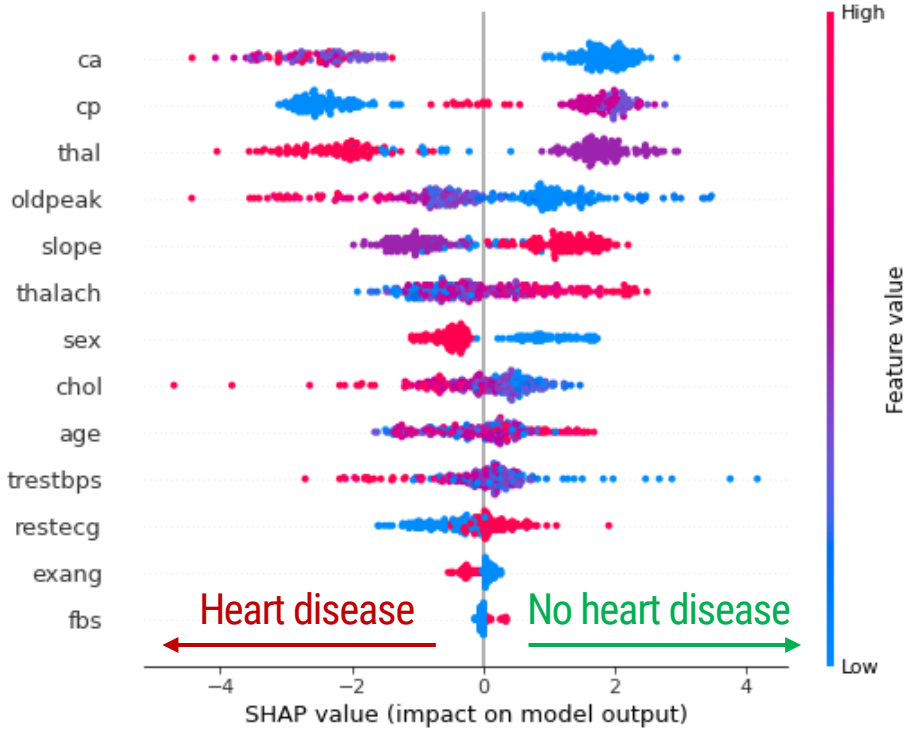
```
In [29]: shap_values = model.get_feature_importance(train_pool, type = "ShapValues")
```

We must separate the actual SHAP values from the baseline value (which is just one value for all rows in your data). This is sort of like the base_score in XGBoost. That's why we take `shap_values[0,-1]` for the base_value, but you could specify any other row (e.g. `shap_values[1,-1]`) since all values in column -1 are the same.

```
In [30]: base_value = shap_values[0,-1]
         shap_values = shap_values[:,:-1]
```

Sample Code

```
In [34]: shap.summary_plot(shap_values, X_train)
```

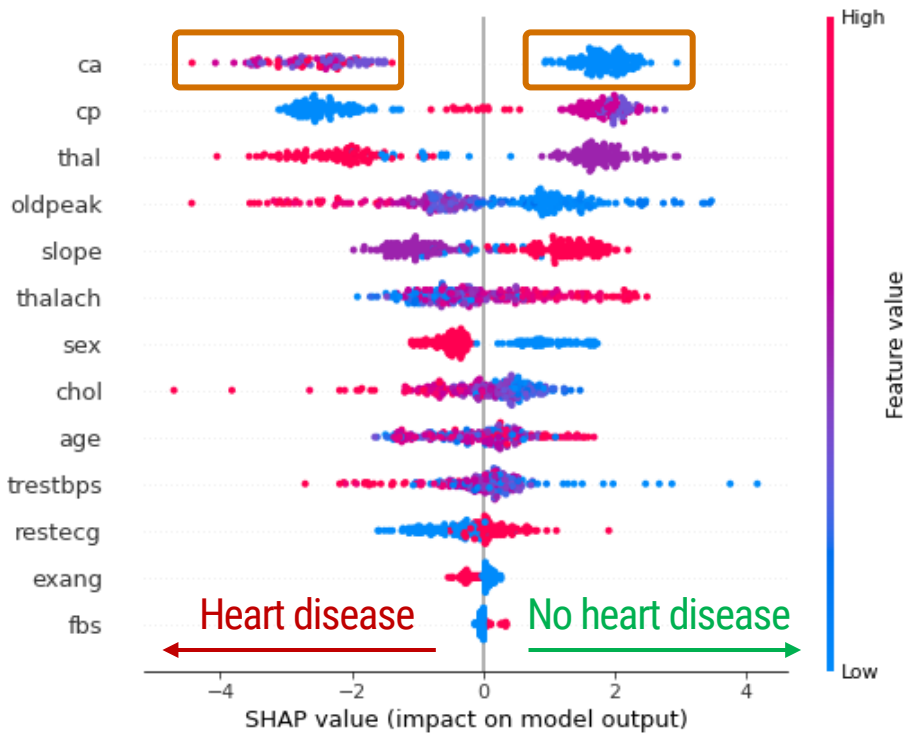


Summary Plot

Describes how high/low valued inputs affect the model output

Sample Code

```
In [34]: shap.summary_plot(shap_values, X_train)
```



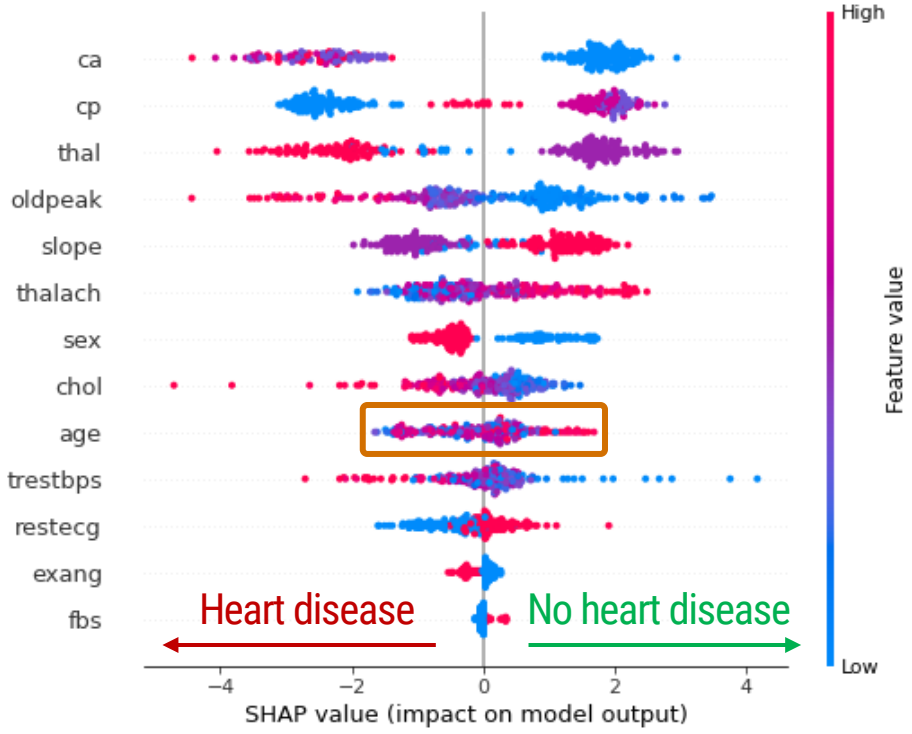
Interpretation

We want good separation between highs & lows.

A high value of ca (# of colored arteries via fluoroscopy) tends to predict heart disease.

Sample Code

```
In [34]: shap.summary_plot(shap_values, X_train)
```



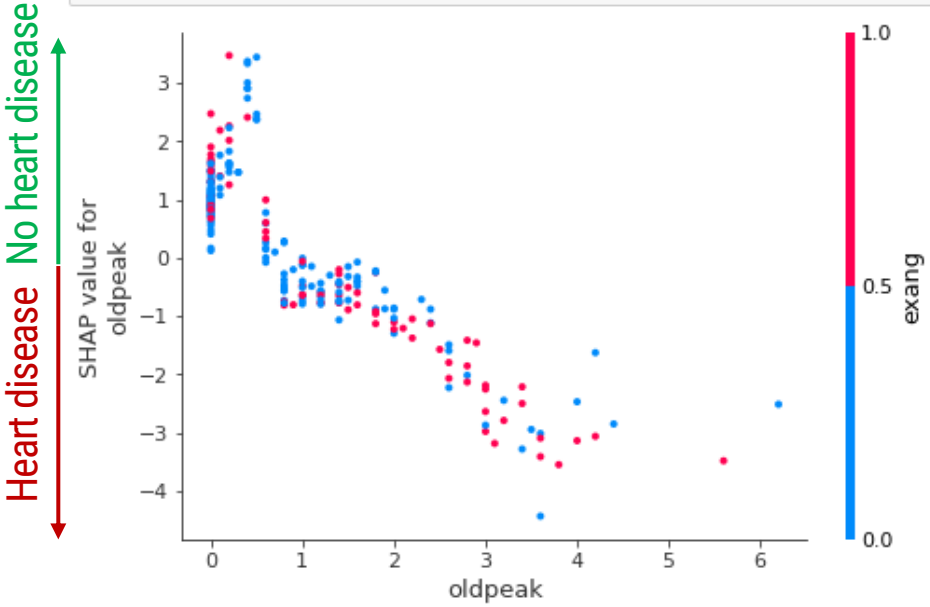
Interpretation

We want good separation between highs & lows.

Age is not a good predictor of heart disease.

Sample Code

```
In [43]: shap.dependence_plot("oldpeak",shap_values,X_train)
```



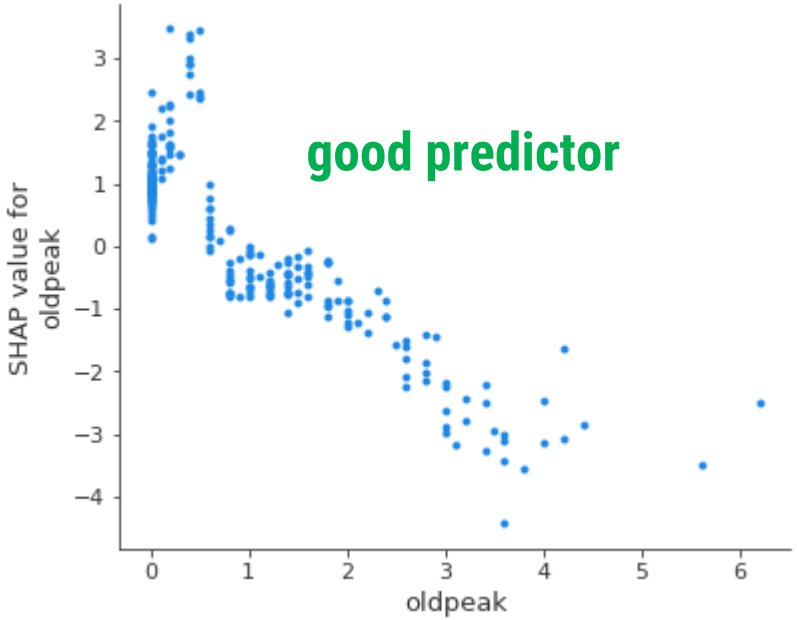
Dependence Plot

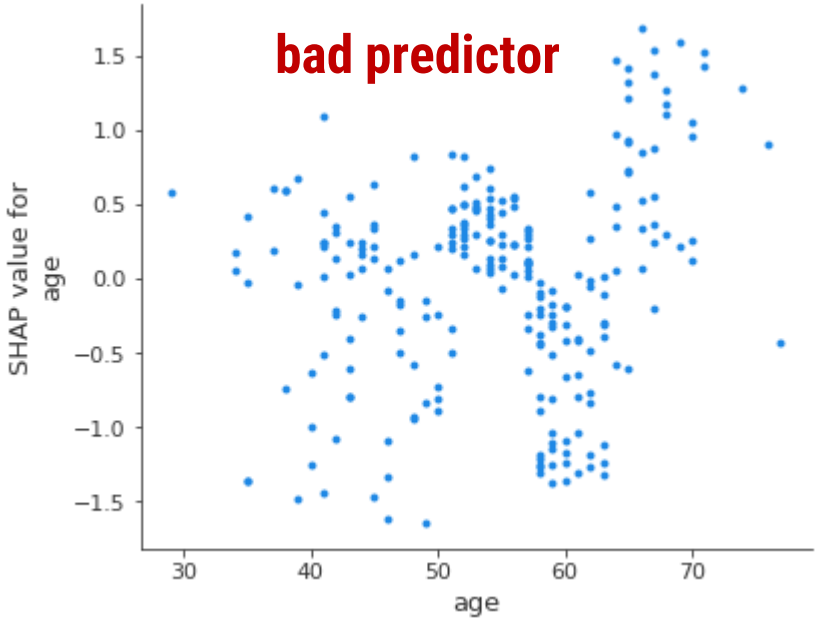
Visualize distribution of a feature and its SHAP values.

Also visualize the interaction with another feature.

By default, the strongest interaction is plotted.

Sample Code





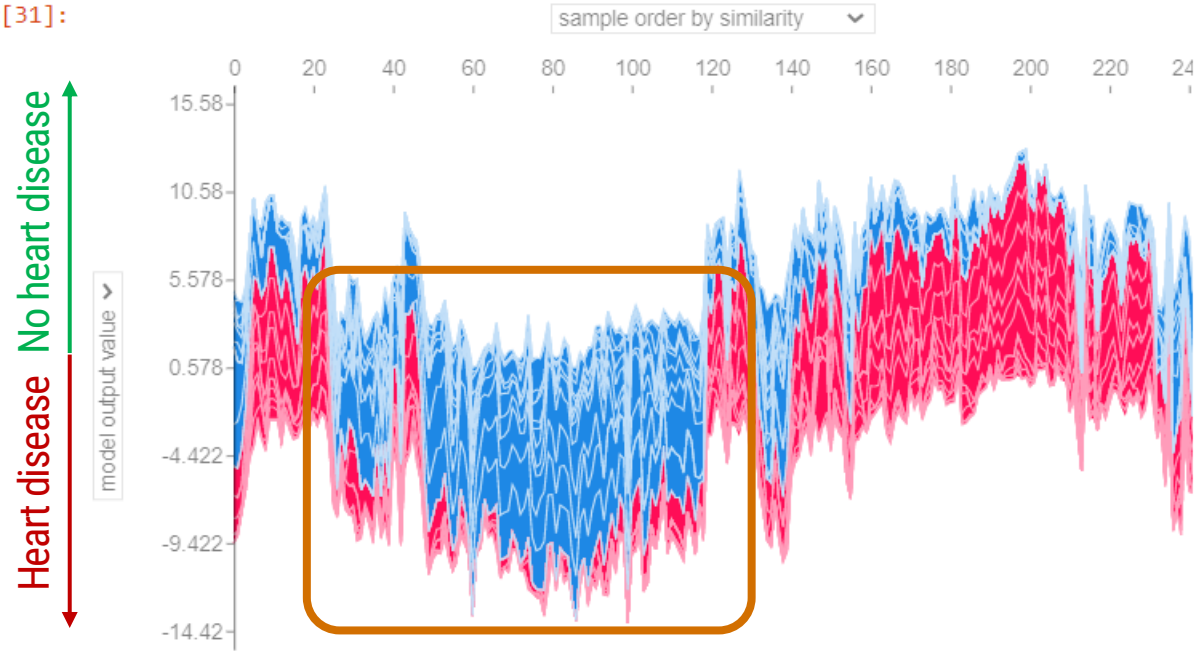
Interpretation

For the scatter plot, a random looking plot is not a good predictor

Sample Code

```
In [31]: shap.force_plot(base_value, shap_values, features = X_train)
```

Out[31]:



This group has more predictions of heart disease

Force Plot

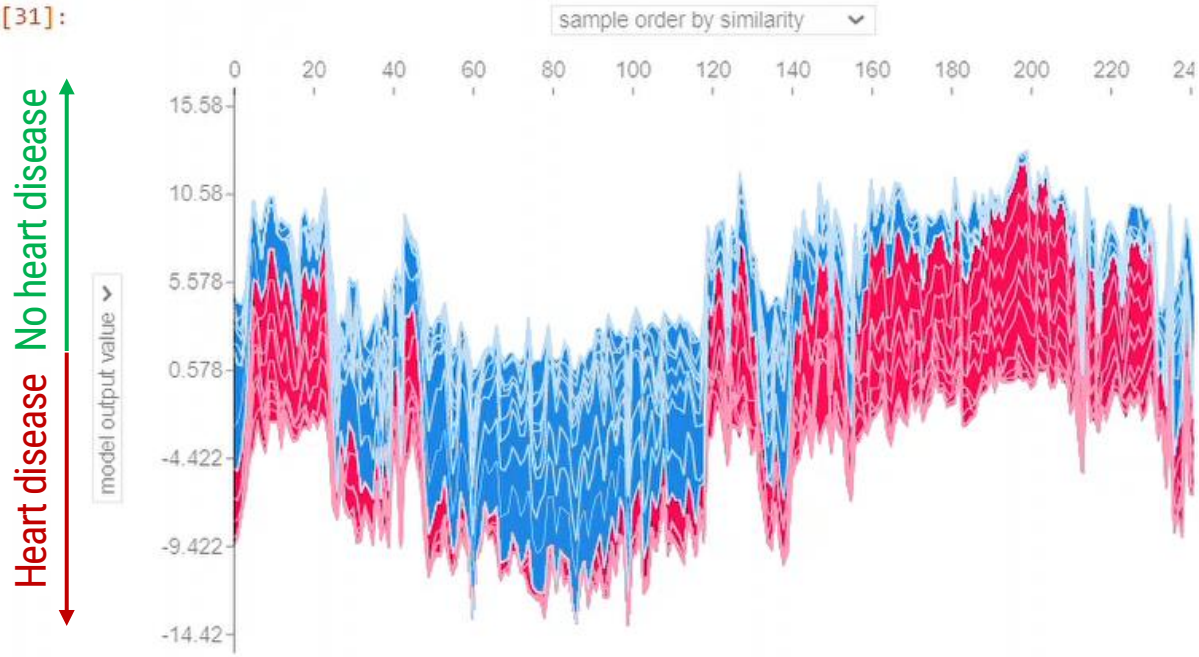
Uses hierarchical clustering on SHAP output values.

You can visualize 'clusters'.

Sample Code

```
In [31]: shap.force_plot(base_value, shap_values, features = X_train)
```

Out[31]:



Interactive

You can view feature values and change plot views.

📖 SHAP in terms of Probability

- We use shap.TreeExplainer
  - Currently, CatBoost does not support outputting SHAP values in terms of probability. SHAP values are in terms of log odds for binary classification.
  - Conversion from log odds to probability is not as simple as using the logistic function.
  - SHAP values in terms of probability add up to the probability output of the model (i.e. if you did .predict_proba)

SHAP in terms of Probability

In [263]:
```
explainer = shap.TreeExplainer(mod,
                               model_output = "probability",
                               data = X_train,
                               feature_perturbation = "interventional",
                               feature_dependence = "independent"
                              )
shap_probs = explainer.shap_values(X_train)
```

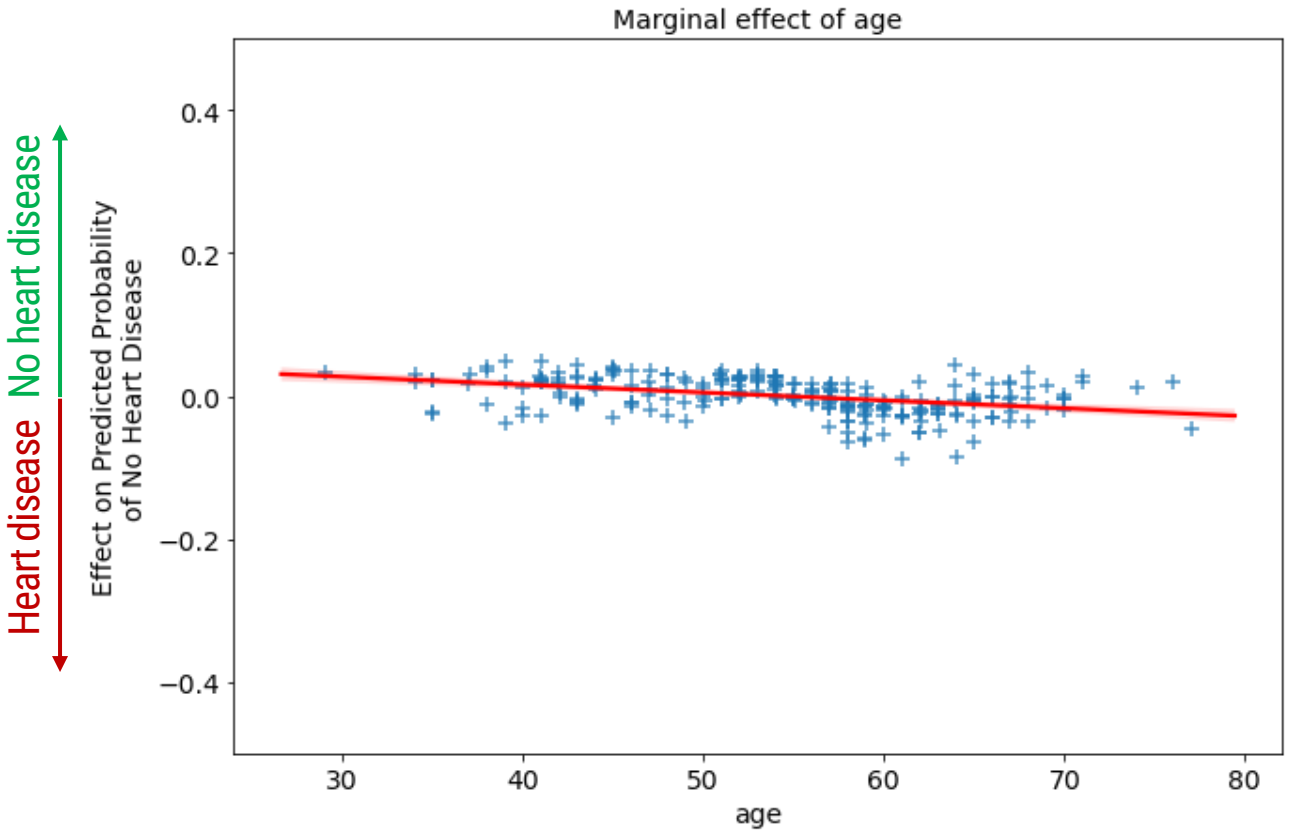
```
97%|================== | 235/242 [00:35<00:01]
```

In [240]:
```
df_shap = pd.DataFrame(shap_probs)
df_shap.columns = X.columns
df_shap.head()
```

Out[240]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	
0	0.017115	-0.017922	0.059139	0.015947	0.041898	-0.001643	-0.015254	0.042317	0.012165	0
1	0.015692	0.074102	0.052651	0.007392	0.011492	0.000832	-0.000727	0.031893	0.026902	-0
2	0.022999	-0.015172	0.115383	0.011865	0.024167	-0.001086	0.025795	0.048111	0.014104	0
3	0.008344	-0.011798	-0.039794	0.002689	0.037178	0.001060	0.021934	0.001189	0.027361	0
4	0.013837	0.078729	-0.035023	0.069530	0.000808	0.001552	0.009542	-0.017483	0.024573	0

SHAP in terms of Probability

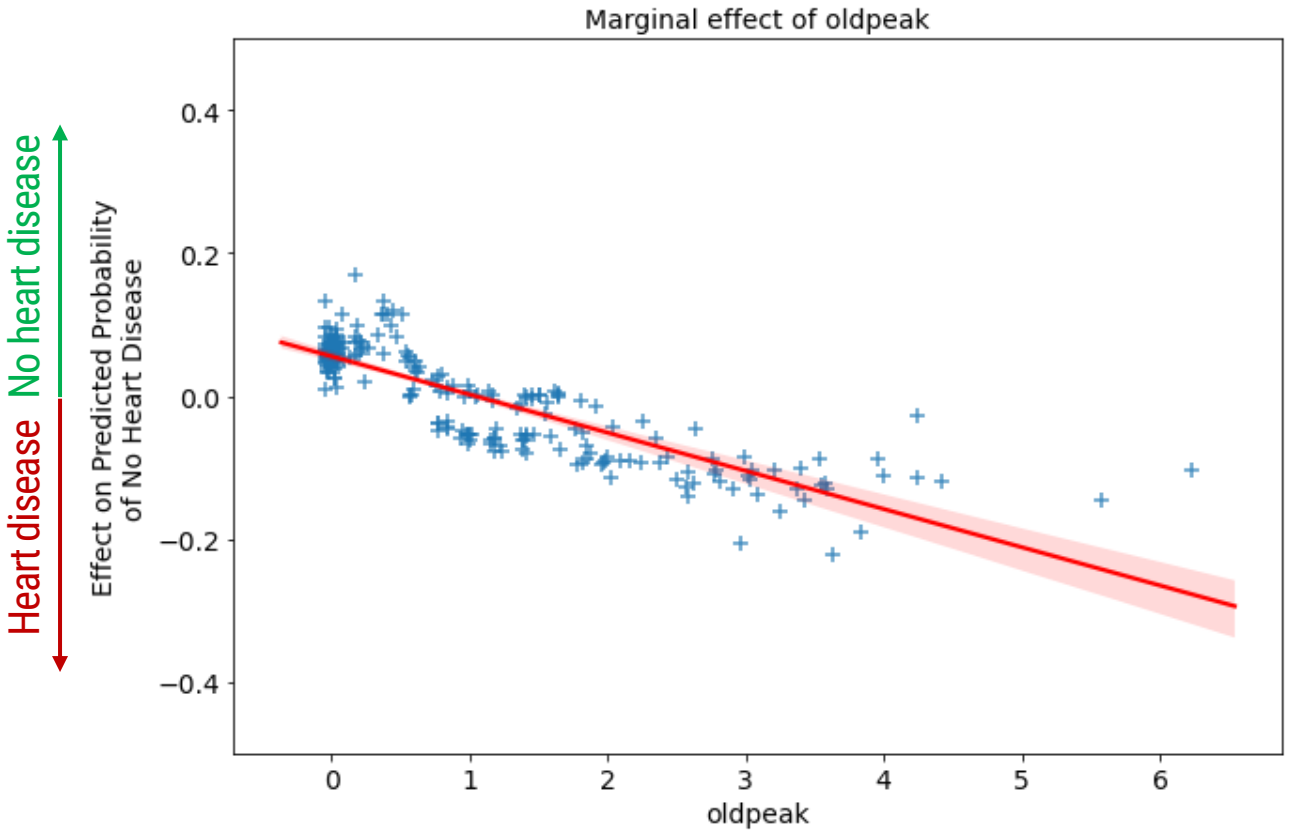


Age

No clear pattern. The effect seems to be random.

The line is almost horizontal.

📖 SHAP in terms of Probability

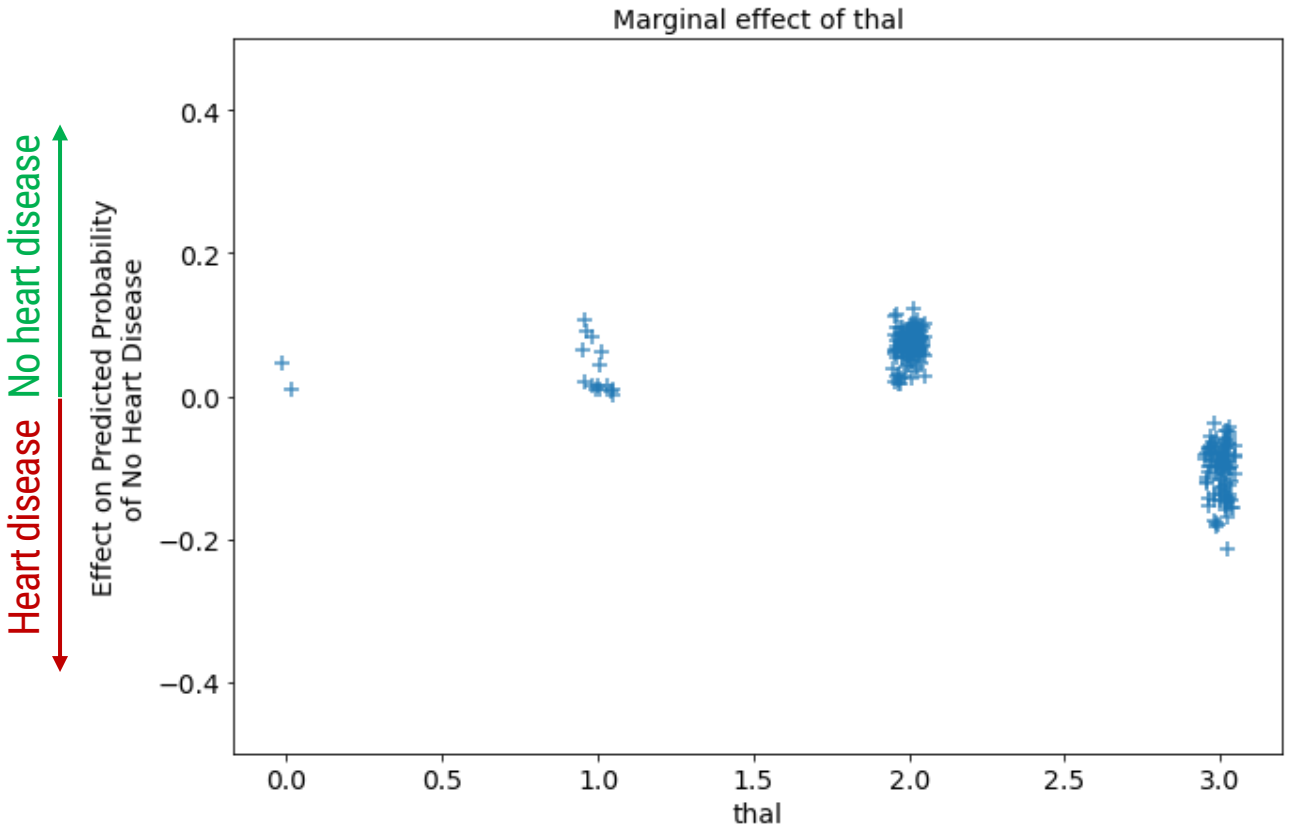


Oldpeak
ST Depression (ECG)

There is a pattern.

Low value = no heart disease.
High value = heart disease.

SHAP in terms of Probability



Thal

Blood Flow

2 – normal

1 & 3 – defect

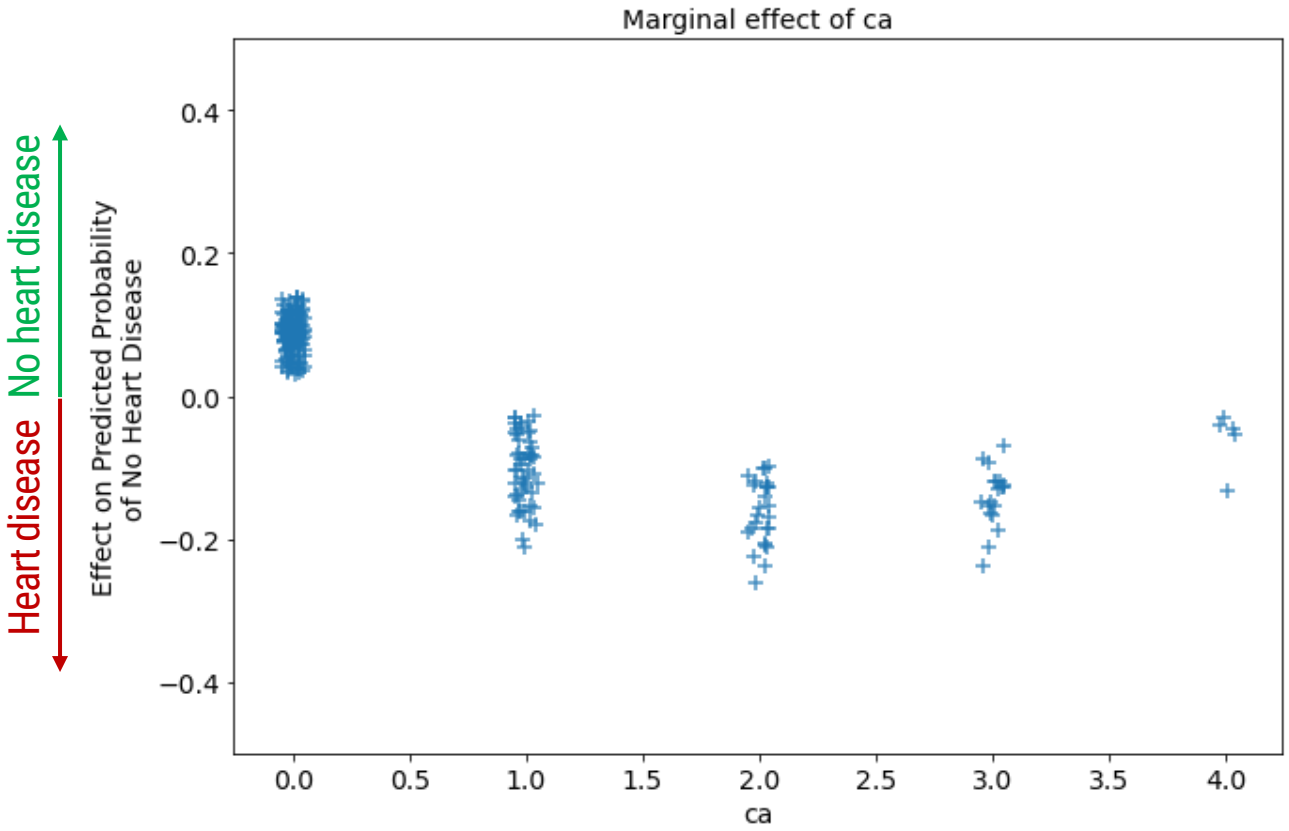
0 – null

Effects are consistent.

Normal = no HD

Defect = HD

SHAP in terms of Probability



Ca

No. of colored arteries

Effects are consistent.
No ca = no HD
Ca > 0 = HD

Thank you!

References

- Cristoph Molnar, Interpretable Machine Learning
- Lundberg & Lee, A Unified Approach to Interpreting Model Predictions
- TDS: SHAP values explained exactly how you wished someone explained it to you
- CatBoost Docs

CREDITS

Special thanks to:

- Presentation template by SlidesCarnival
- Photographs by Startup Stock Photos