Model Interpretability with Visual Cues

Introducing the SHAP python package

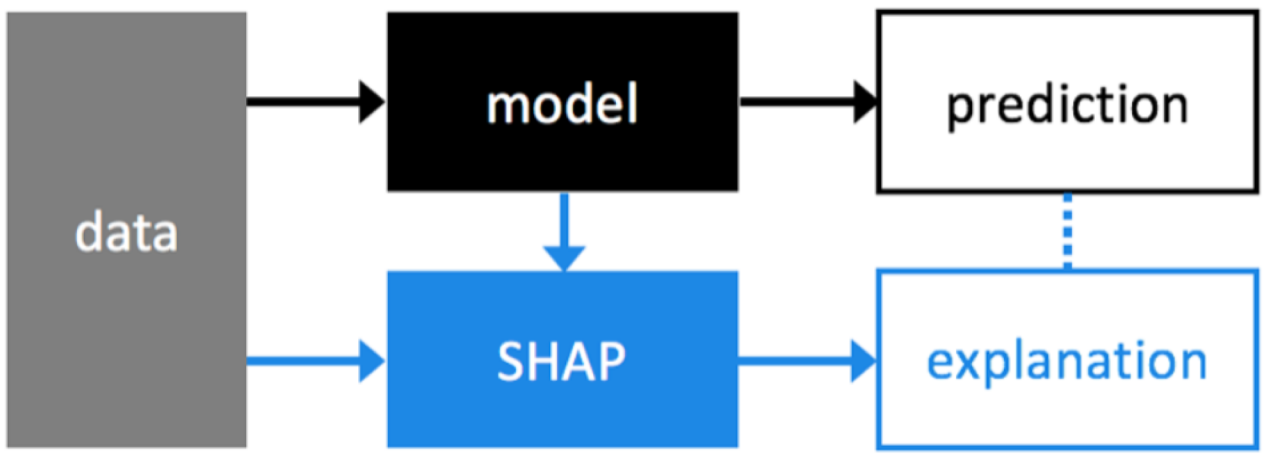
Accuracy - Interpretability trade-off

	Interpretable	Accurate
Complex model	✗	✔
Simple model	✔	✗

What we would like to do

SHAP (SHapley Additive exPlanation)

A unified approach to explain the output of any machine learning model (*Scott Lundberg 2017*)



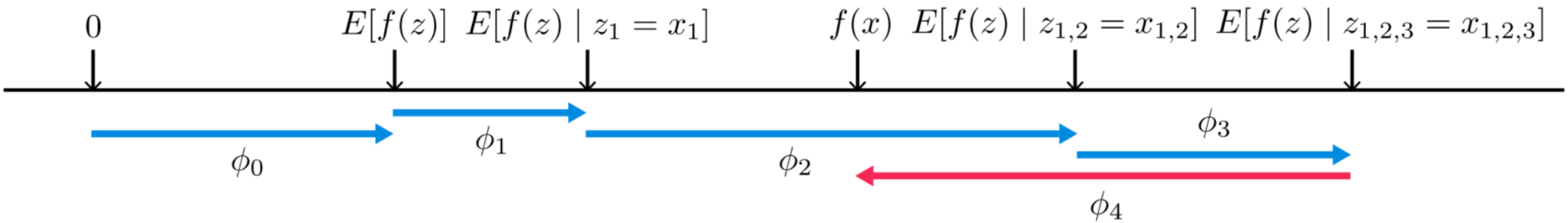
How does it works?

Fits an *explanation model* on the prediction of your model.

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

1. The parameters of the *explanation model* are called the *Shapley values*; they represent the role of a variable on the prediction of a particular input.
2. The *explanation model* is an additive model, making it trivial to interpret.
3. Works for any function that takes an input and return an output

In a nutshell



If features are non-linear or not independent, then the ordering matters. The `SHAP values` is the average of all computed `shapley values` over different ordering.

For an additive model, the python package `shap` implements the only solution with theoretical guarantees on the following 3 assumptions:

1. Local accuracy: The prediction of the *explanation model* matches the original prediction.

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

For an additive model, the python package `shap` implements the only solution with theoretical guarantees on the following 3 assumptions:

1. Local accuracy: The prediction of the *explanation model* matches the original prediction.

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

2. Missingness: A missing in the data has no effect on the explanation

$$x'_i = 0 \implies \phi_i = 0$$

3. Consistency: If toggling off a feature in a model always has a greater impact than in another model. Then its SHAP-value is greater in the former than in the latter.

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i)$$

$$\textit{for all inputs } z' \in \{0,1\}^M, \textit{ then } \phi_i(f', x) \geq \phi_i(f, x)$$

Function that returns the prediction from a model

Function that returns the prediction from a model

In [5]:
```
def f(X, model):
    try:    # Keras
        return model.predict(X)[:,1]
    except: # sklearn
        return model.predict_proba(X)[:,1]
```

Project intuition:

To identify drivers who's driving score varies a lot

Project intuition:

To identify drivers who's driving score varies a lot

In [7]:
```
import joblib, pandas as pd
from functools import partial
train = pd.read_csv('../inputs/S2D/train.csv', usecols=keep_cols)
test = pd.read_csv('../inputs/S2D/test.csv', usecols=keep_cols)
gb = joblib.load('../inputs/S2D/gb.pkl') # Tree-based model
```

Project intuition:

To identify drivers who's driving score varies a lot

In [7]:
```
import joblib, pandas as pd
from functools import partial
train = pd.read_csv('../inputs/S2D/train.csv', usecols=keep_cols)
test = pd.read_csv('../inputs/S2D/test.csv', usecols=keep_cols)
gb = joblib.load('../inputs/S2D/gb.pkl') # Tree-based model
```

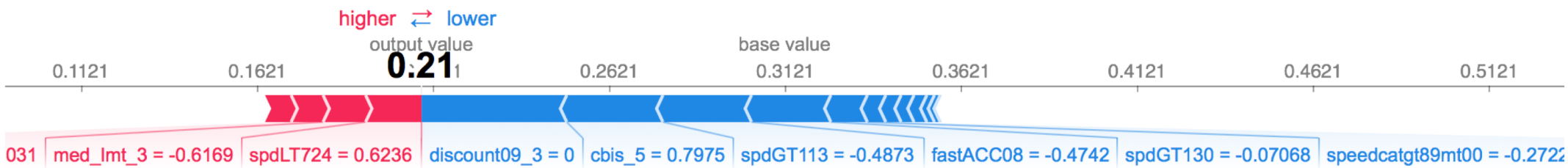
In [8]:
```
import shap
shap.initjs()
```

Using a gradient-boosting tree ensemble

In [9]:
```
row_id = 31
ff = partial(f, model=gb)

explainer = shap.KernelExplainer(ff, train.sample(300))
shap_values = explainer.shap_values(test.iloc[row_id,:], nsamples=500)
shap.force_plot(shap_values, test.iloc[[row_id]])
```

Out[9]:



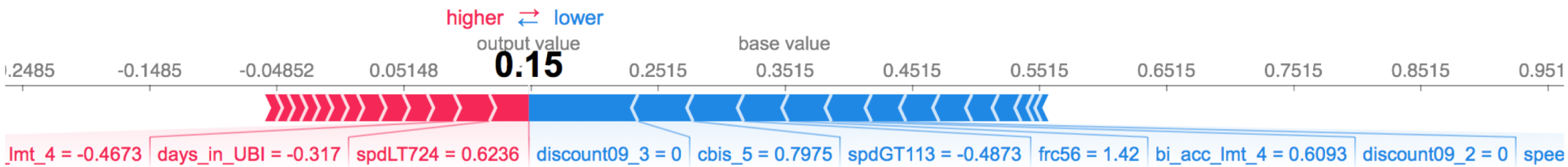
Using a neural network

In [10]:
```
%%capture
from keras.models import load_model
snn = load_model('../inputs/S2D/snn.h5')
```

In [11]:
```
row_id = 31
ff = partial(f, model=snn)

explainer = shap.KernelExplainer(ff, train.sample(300))
shap_values = explainer.shap_values(test.iloc[row_id,:], nsamples=500)
shap.force_plot(shap_values, test.iloc[[row_id]])
```

Out[11]:

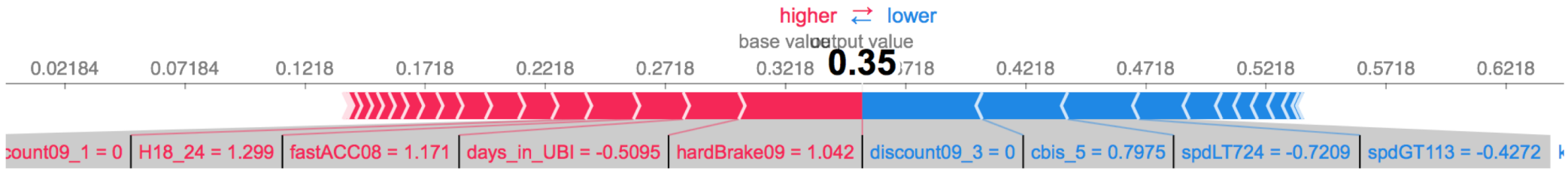


Let’s take a look at a couple more examples

In [28]:
```
#row_id = 91 # 91: Man + days_in_UBI>3, 559: Woman + days_in_UBI>3
#row_id = 96 # 96: Man + days_in_UBI<3, 568: Woman + days_in_UBI<3
row_id = 144 # 144: high HB
```

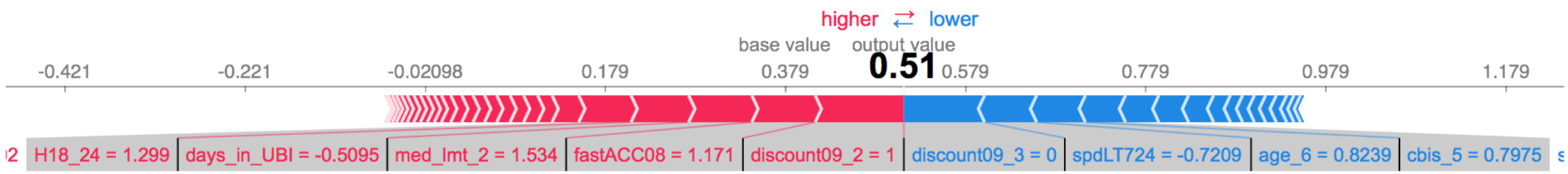
In [29]:
```
ff = partial(f, model=gb)
explainer = shap.KernelExplainer(ff, train.sample(400))
shap_values = explainer.shap_values(test.iloc[row_id,:], nsamples=300)
shap.force_plot(shap_values, test.iloc[[row_id]])
```

Out[29]:



In [30]:
```
ff = partial(f, model=snn)
explainer = shap.KernelExplainer(ff, train.sample(300))
shap_values = explainer.shap_values(test.iloc[row_id,:], nsamples=500)
shap.force_plot(shap_values, test.iloc[[row_id]])
```

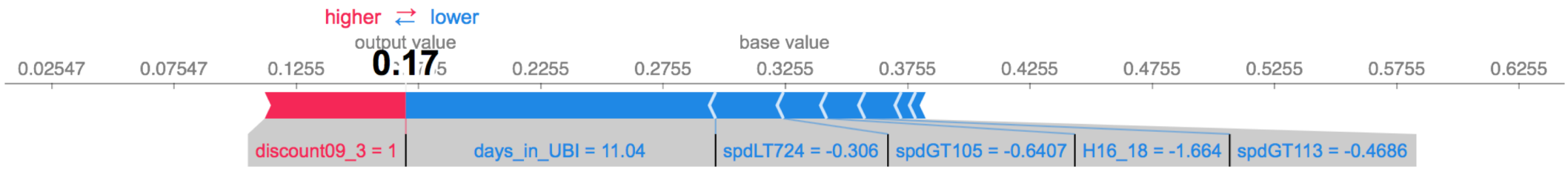
Out[30]:



In [31]:
```
row_id = 91 # 91: Man + days_in_UBI>3, 559: Woman + days_in_UBI>3
#row_id = 96 # 96: Man + days_in_UBI<3, 568: Woman + days_in_UBI<3
#row_id = 144 # 144: high HB
```

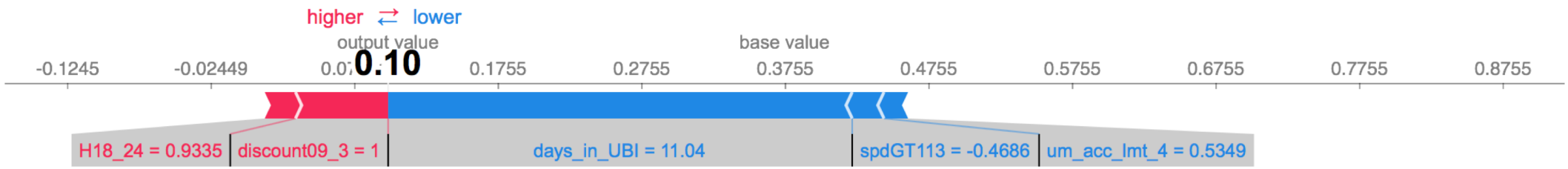
In [32]:
```
ff = partial(f, model=gb)
explainer = shap.KernelExplainer(ff, train.sample(400))
shap_values = explainer.shap_values(test.iloc[row_id,:], nsamples=300)
shap.force_plot(shap_values, test.iloc[[row_id]])
```

Out[32]:



In [33]:
```
ff = partial(f, model=snn)
explainer = shap.KernelExplainer(ff, train.sample(300))
shap_values = explainer.shap_values(test.iloc[row_id,:], nsamples=500)
shap.force_plot(shap_values, test.iloc[[row_id]])
```

Out[33]:



Importance features

For any type of models?!

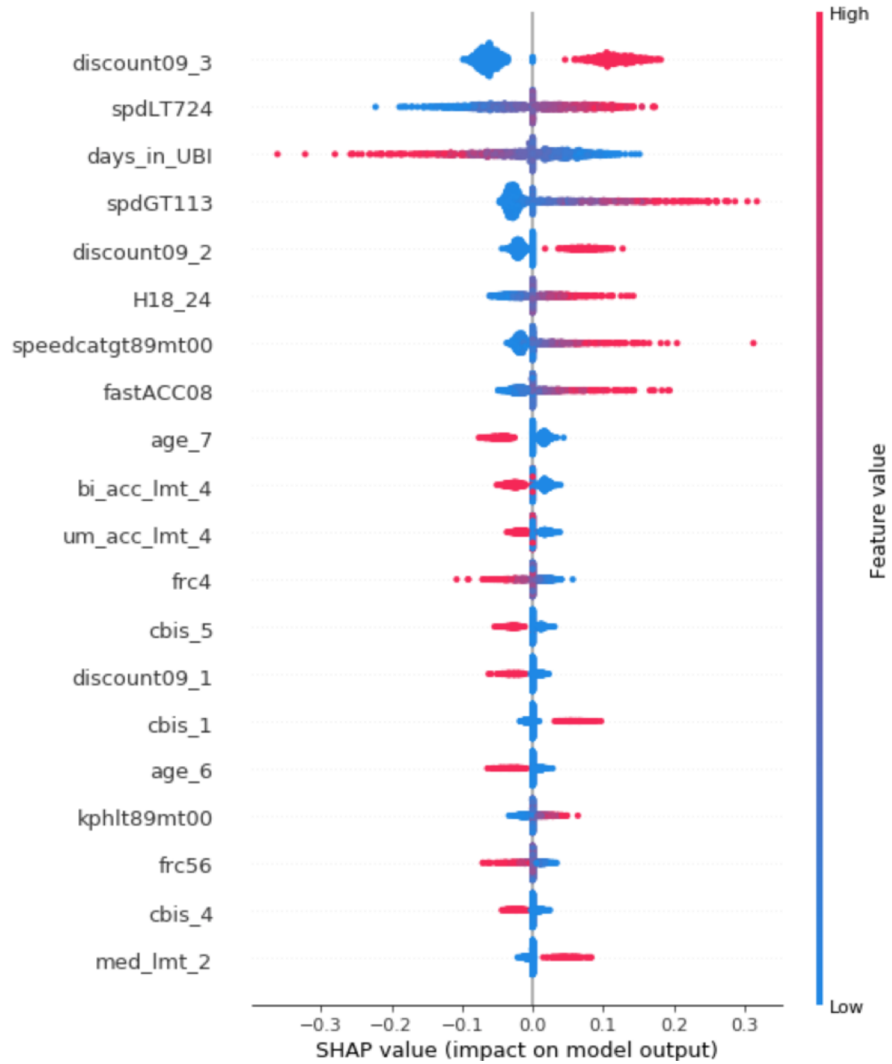
In [12]:
```
# Put importances features in a dataset
imp_ft = pd.DataFrame({'colnames': train.columns.values, 'importances':gb.feature_impor
# Print
imp_ft.sort_values(by='importances', ascending=False).head(5).reset_index(drop=True)
```

Out[12]:

	colnames	importances
0	H18_24	0.078420
1	spdLT724	0.071804
2	fastACC08	0.061273
3	days_in_UBI	0.059452
4	spdGT113	0.052176

In [13]:
```
ff = partial(f, model=snn)
explainer = shap.KernelExplainer(ff, train.sample(300))
shap_values = explainer.shap_values(test, nsamples=400)
shap.summary_plot(shap_values, test)
```

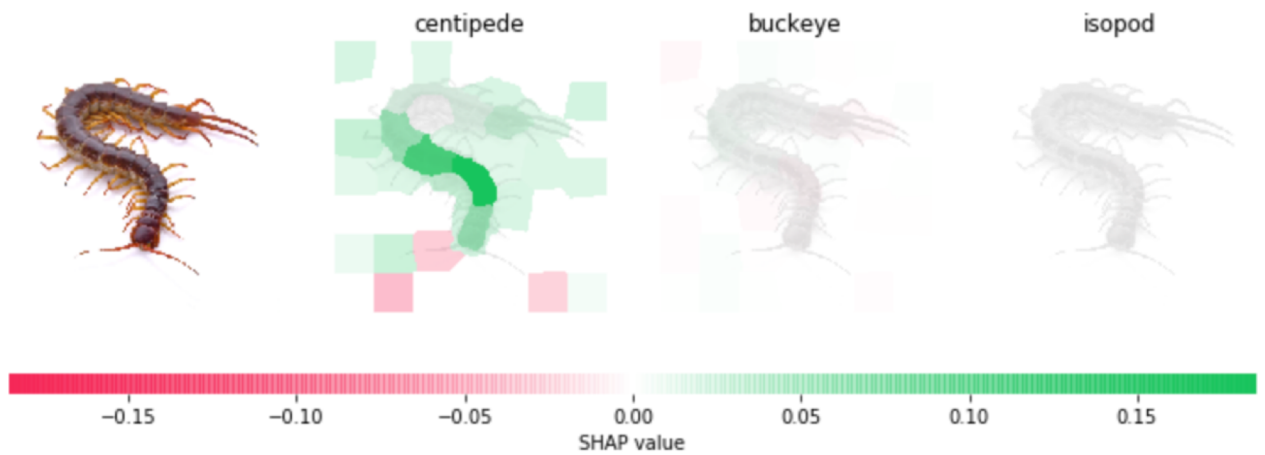
```
100%|██████████| 995/995 [4:09:14<00:00, 15.03s/it]
```



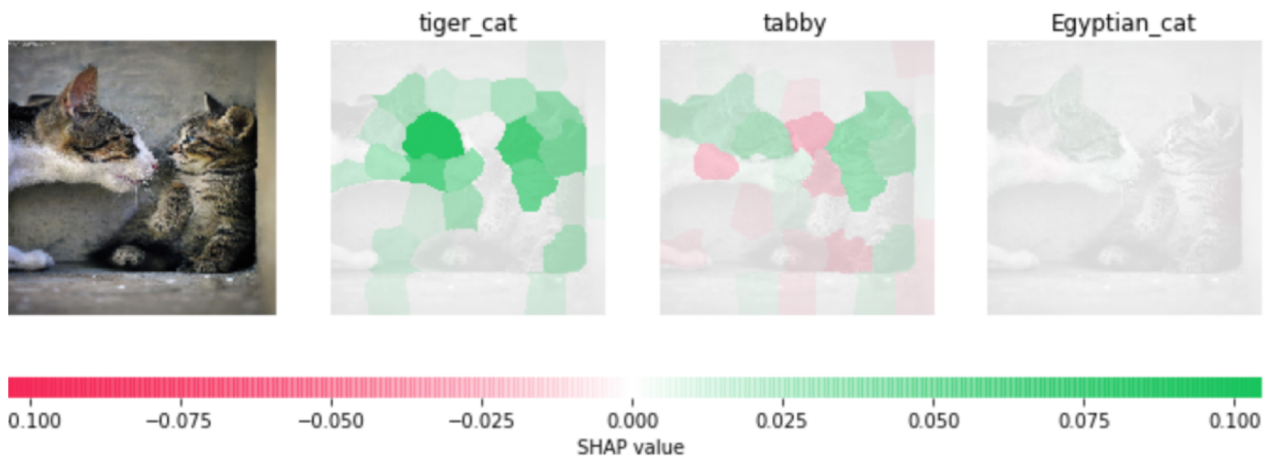
Model explanation with images

```
# K-mean the image
# Each centroid becomes a feature
# Each patch is a feature for our explanation model
from skimage.segmentation import slic

segments_slic = slic(img, n_segments=50, compactness=30,
                     sigma=3)
```


centipede
buckeye
isopod
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
SHAP value


sock
Christmas_stocking
digital_watch
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
SHAP value


tiger_cat
tabby
Egyptian_cat
0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100
SHAP value