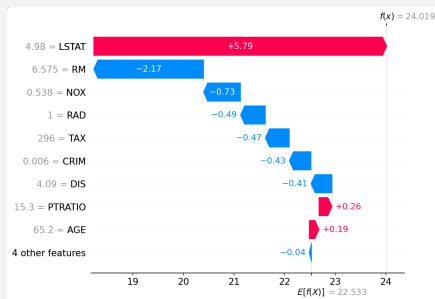


### Before you start

```
import xgboost
import shap
import plotly.express as px
import dalex as dx
X, y = shap.data.set.suboston()
model = xgboost.XGBRegressor().fit(X, y)
explainer = shap.Explainer(model)
boston_rf_exp = dx.Explainer(model, X, y,
                             label="Boston
houses RF Pipeline")
shap_values = explainer(X)
```

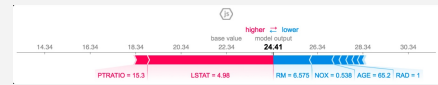
### Waterfall chart



Used to see contributions of different attributes for the prediction. These SHAP values are valid for this observation only. With other data points the SHAP values will change.

```
shap.plot.force(shap_values[0])
```

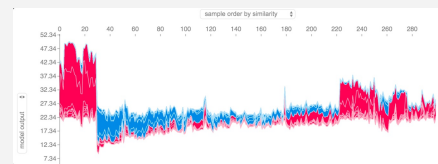
### Force plot



Exactly the same purpose as the waterfall chart but much more compact

```
shap.plot.force(shap_values[0])
```

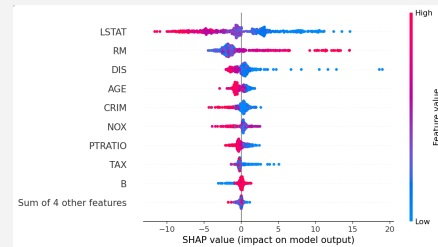
### SHAP Summaries



If you take force plots for all observations, rotate them by 90 degrees and then put next to each other you obtain a SHAP summary plot. This is very useful if you want to see explanations for the entire dataset.

```
shap.plot.force(shap_values)
```

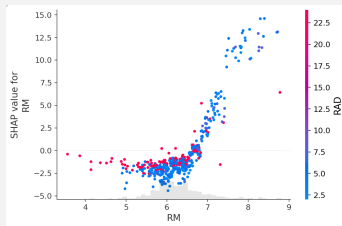
### SHAP Beeswarm



Useful to see which attributes are the most important. For every feature and every sample we plot a dot. We denote value of the feature with color: big (red) or small (blue). On the X-axis we see the importance. From this plot we see that LSTAT is probably the most important attribute. Also, high value of RM increases the model prediction

```
shap.plot.beeswarm(shap_values)
```

### Feature interaction



This one is helpful to capture feature interaction and how they influence SHAP value for given feature. On X and Y axis we have information about attribute we are interested in. Color represents value of another feature that is interacting with considered. From here we see that if RAD is small then RM have quite big impact on the prediction whereas when RAD is big then this impact is much smaller.

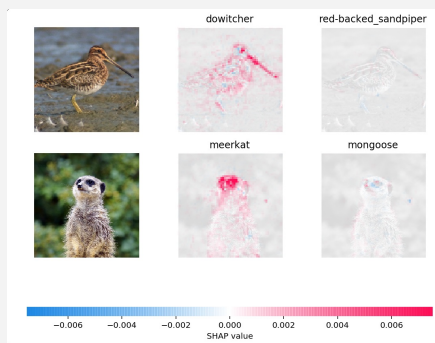
```
shap.plot_s.catter(shap_values[:, "RM"], color=shap_values)
```

### SHAP for text



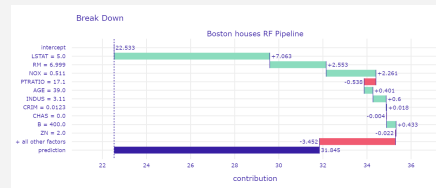
We can extend this idea to text and see how particular words influence the prediction.

### SHAP for images



This can be also used for images to see the influence of individual pixels.

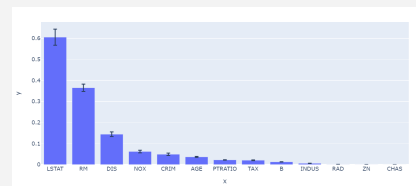
### Breakdown plot



This plot shows the decomposition of the model's prediction into contributions of different attributes

```
bd = boston_rf_ex.p.predict_parts -
    (house, type='break_down')
bd.plot()
```

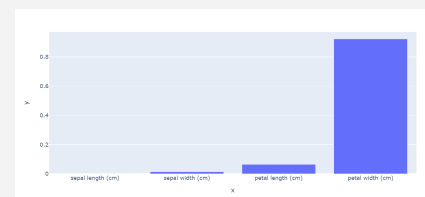
### Permutation importance



Every attribute is scrambled and then based on some evaluation metric (MSE, ACC) we give them scores. Can be visualized on bar chart.

```
from sklearn.inspection import permutation_
importance
r = permutation_importance(model,
X, y)
```

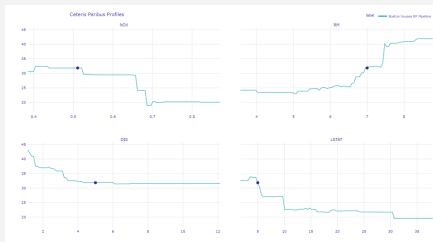
### Tree models feature importance



Tree algorithms offer importance scores based on the reduction in the evaluation criterion, like Gini or entropy. Can be used either in regression or classification problems in decision trees, random forests or boosting methods.

```
px.bar(x=X.columns, y=model.feature_
importance_)
```

## Ceteris paribus profiles (partial dependence plot)



This figure shows how different attributes in a new instance can change a prediction of the model.

In a nutshell, we held all explanatory variables but one (can increase this but computational cost increases by much) constant. Then we change the values of one selected and see how the response changes.

```
cp = boston_rf_exp.predict_profile(
    house)
cp.plot(variables = ['NOX', 'RM',
                    'DIS', 'LSTAT'])
```

## Linear model feature importance



After scaling features we can measure how each attribute is important for the model

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import scale

linearModel = LinearRegression().fit(scale(X), y)
px.bar(y = linearModel.coef_, x = X.columns)
```