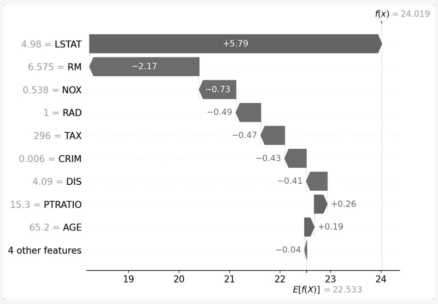
Explainable AI Cheat Sheet

by SebChw via cheatography.com/149723/cs/32502/

Before you start - SHAP

```
import xgboost
import shap
X, y = shap.d ata set s.b oston()
model = xgboos t.X GBR egr ess or( ).f it(X, y)
explainer = shap.E xpl ain er( model)
shap_v alues = explai ner(X)
```

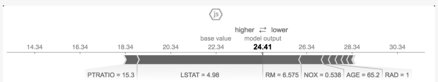
Waterfall Chart



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

```
shap.p lot s.f orc e(s hap _va lue s[0])
```

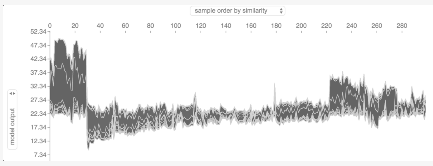
Force Plot



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

```
shap.p lot s.f orc e(s hap _va lue s[0])
```

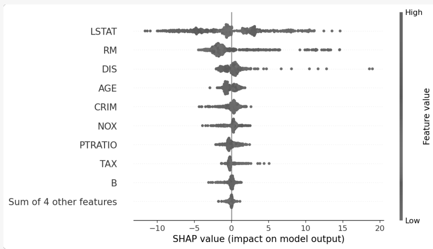
SHAP Summaries



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 te see explanations for the entire dataset.

```
shap.p lot s.f orc e(s hap _va lues)
```

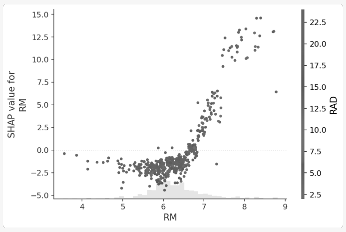
SHAP Beeswarm



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.p lot s.b ees war m(s hap _va lues)
```

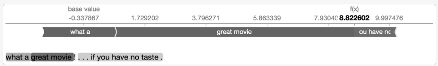
Feature Interaction



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.

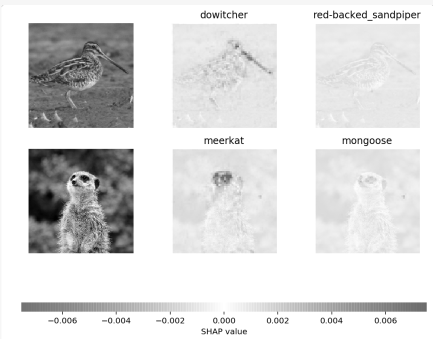
```
shap.p lot s.s cat ter (sh ap_ val ues [:, " RM"],
color= sha p_v alues)
```

SHAP for text



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

SHAP for images

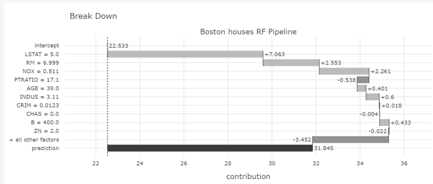


This can be also used for images to see the influence of individual pixels.

Before you start - OTHER XAI PLOTS

```
from sklearn.linear_model import LinearRegression
import plotly.ex press as px
import dalex as dx
linarModel = Linear Reg res sio n().fi t(s cal -
e(X), y)
boston _rf_exp = dx.Exp lai ner (model, X, y,
label= " Boston houses RF Pipeli ne")
```

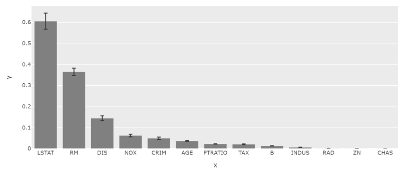
Break down plot



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

```
bd = boston _rf _ex p.p red ict _pa rts (house, ty
pe=' bre ak_ down')
bd.plot()
```

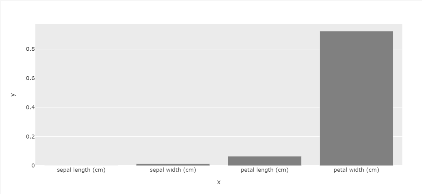
Permutation importance



This functions function calculates the feature importance of estimators for a given dataset for given evaluation metrics. Can be visualized on bar chart.

```
r = permut ati on_ imp ort anc e(m odel, X, y)
```

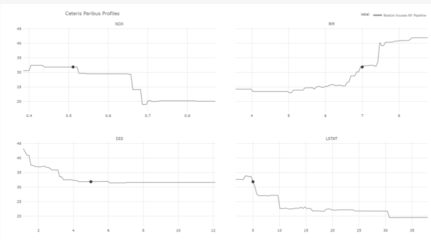
Tree models feature importance



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.c olumns, y=mode l.f eat ure _im por t
an ces_)
```

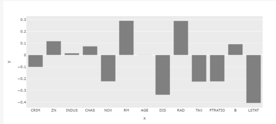
Ceteris paribus profiles (partial dependence plot)



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 const increases by much) constant. Then we change the values of one selected and see how the response changes.

```
cp = titani  c_ r f _e  xp.p  r e d ic  t_p  ro f i
 le  (house)
cp.plo  t( v a ri  abl  es = [ 'NOX', 'RM', 'DIS',
'LSTAT'])
```

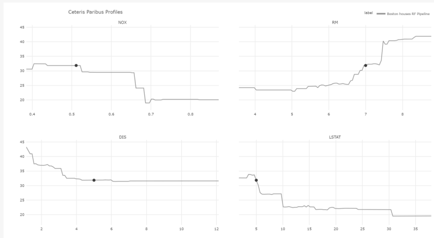
Linear model feature importance



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

```
px.bar  (y = a bs  (l  ina  rM o d el.c o  ef_), x
=X.co  lumns)
```

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 const increases by much) constant. Then we change the values of one selected and see how the response changes.

```
cp = titani   c _ r  f  _e   xp.p   r  e d  ic   t
 _p   ro  f  i l e   (house)
cp.plo   t ( v  a  ri   ab l   es  = [  'NOX', 'RM
', 'DIS', 'LSTAT'])
```