ML Interpretation

What Types of Insights Are Possible

- What features in the data did the model think are most important?
- For any single prediction from a model, how did each feature in the data affect that particular prediction?
- How does each feature affect the model's predictions in a big-picture sense (what is its typical effect when considered over a large number of possible predictions)?

Why Are These Insights Valuable

These insights have many uses, including

- Debugging
- Informing feature engineering
- Directing future data collection
- Informing human decision-making
- Building Trust

One of the most basic questions we might ask of a model is *What features have the biggest impact on predictions?* This concept is called *feature importance*

Permutation importance

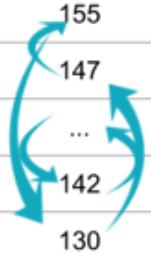
Consider data with the following format:

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24

We want to predict a person's height when they become 20 years old, using data that is available at age 10.

Permutation importance is calculated after a model has been fitted. So we won't change the model or change what predictions we'd get for a given value of height, sock-count, etc.

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24



Randomly re-ordering a single column should cause less accurate predictions,

Code Example

Our example will use a model that predicts whether a soccer/football team will have the "Man of the Game" winner based on the team's statistics.

```
data = pd.read_csv('../input/fifa-2018-match-statistics/FIFA 2018 Statistics.csv')
y = (data['Man of the Match'] == "Yes")  # Convert from string "Yes"/"No" to binary
feature_names = [i for i in data.columns if data[i].dtype in [np.int64]]
X = data[feature_names]
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)
my_model = RandomForestClassifier(n_estimators=100,
                                  random_state=0).fit(train_X, train_y)
```

```
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(my_model, random_state=1).fit(val_X, val_y)
eli5.show_weights(perm, feature_names = val_X.columns.tolist())
```

Weight	Feature
0.1750 ± 0.0848	Goal Scored
0.0500 ± 0.0637	Distance Covered (Kms)
0.0437 ± 0.0637	Yellow Card
0.0187 ± 0.0500	Off-Target
0.0187 ± 0.0637	Free Kicks
0.0187 ± 0.0637	Fouls Committed
0.0125 ± 0.0637	Pass Accuracy %
0.0125 ± 0.0306	Blocked
0.0063 ± 0.0612	Saves
0.0063 ± 0.0250	Ball Possession %
0 ± 0.0000	Red
0 ± 0.0000	Yellow & Red
0.0000 ± 0.0559	On-Target
-0.0063 ± 0.0729	Offsides
-0.0063 ± 0.0919	Corners
-0.0063 ± 0.0250	Goals in PSO
-0.0187 ± 0.0306	Attempts
-0.0500 ± 0.0637	Passes

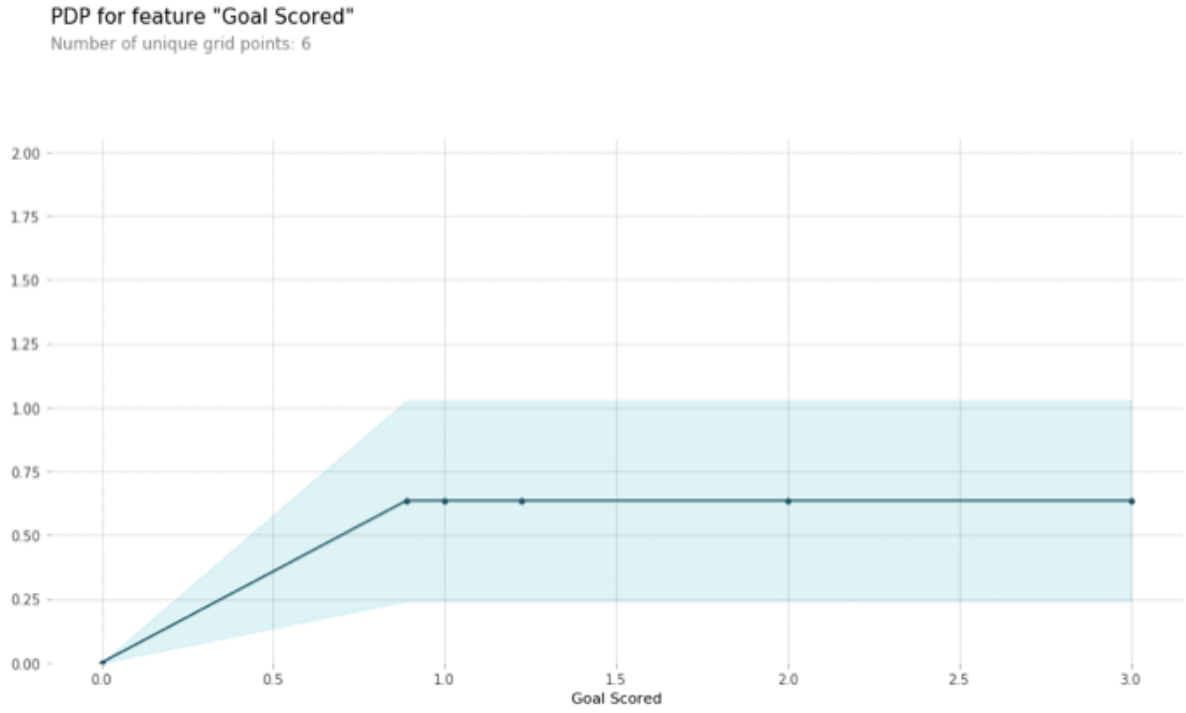
Partial Dependence Plots

While feature importance shows **what** variables most affect predictions, partial dependence plots show ***how*** a feature affects predictions.

This is useful to answer questions like:

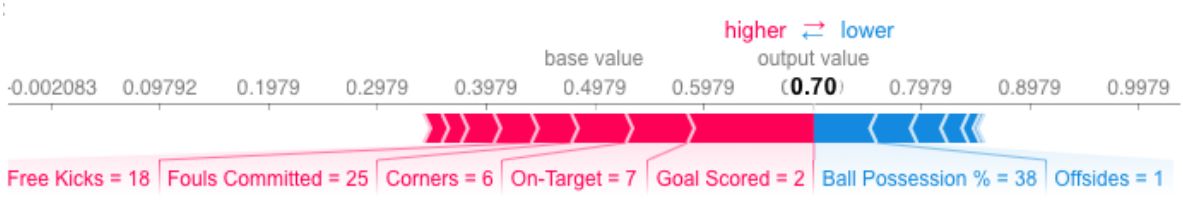
- Controlling for all other house features, what impact do longitude and latitude have on home prices? To restate this, how would similarly sized houses be priced in different areas?
- Are predicted health differences between two groups due to differences in their diets, or due to some other factor?

Like permutation importance, **partial dependence plots are calculated after a model has been fit.**The model is fit on real data that has not been artificially manipulated in any way.



SHAP Values (an acronym from SHapley Additive exPlanations) break down a prediction to show the impact of each feature. Where could you use this?

- A model says a bank shouldn't loan someone money, and the bank is legally required to explain the basis for each loan rejection
- A healthcare provider wants to identify what factors are driving each patient's risk of some disease so they can directly address those risk factors with targeted health interventions



```
import shap  # package used to calculate Shap values

# Create object that can calculate shap values
explainer = shap.TreeExplainer(my_model)

# Calculate Shap values
shap_values = explainer.shap_values(data_for_prediction)
```

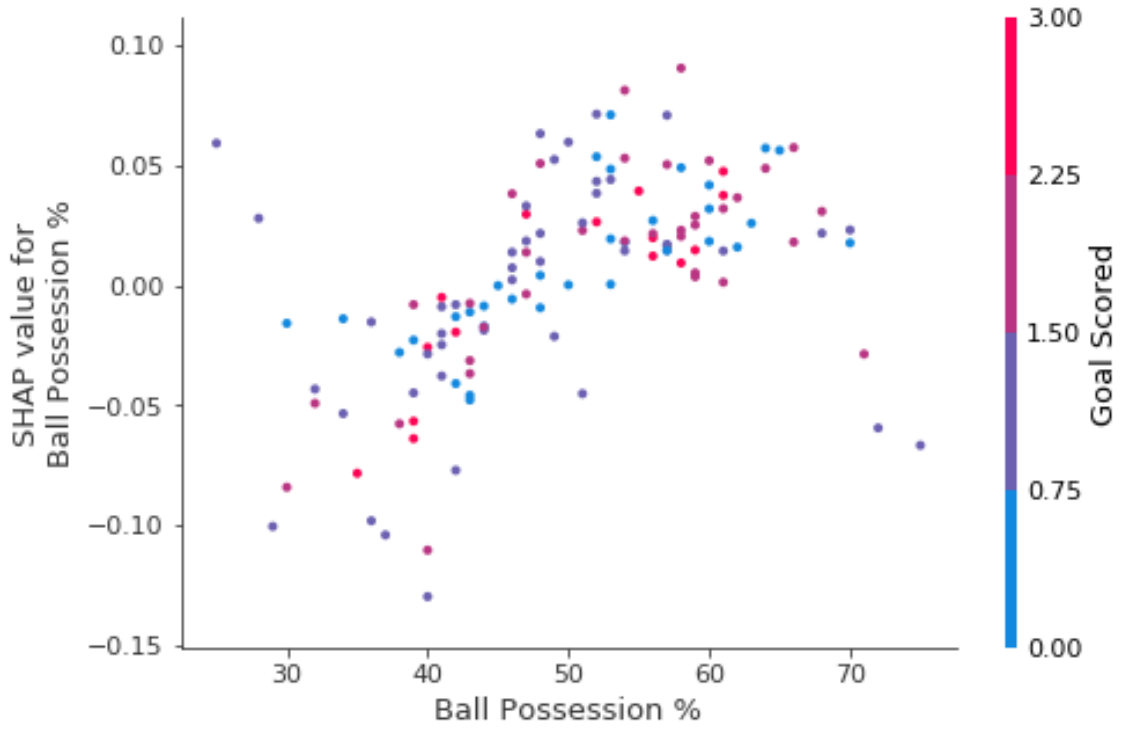
```
shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1], data_for_predicti
on)
```

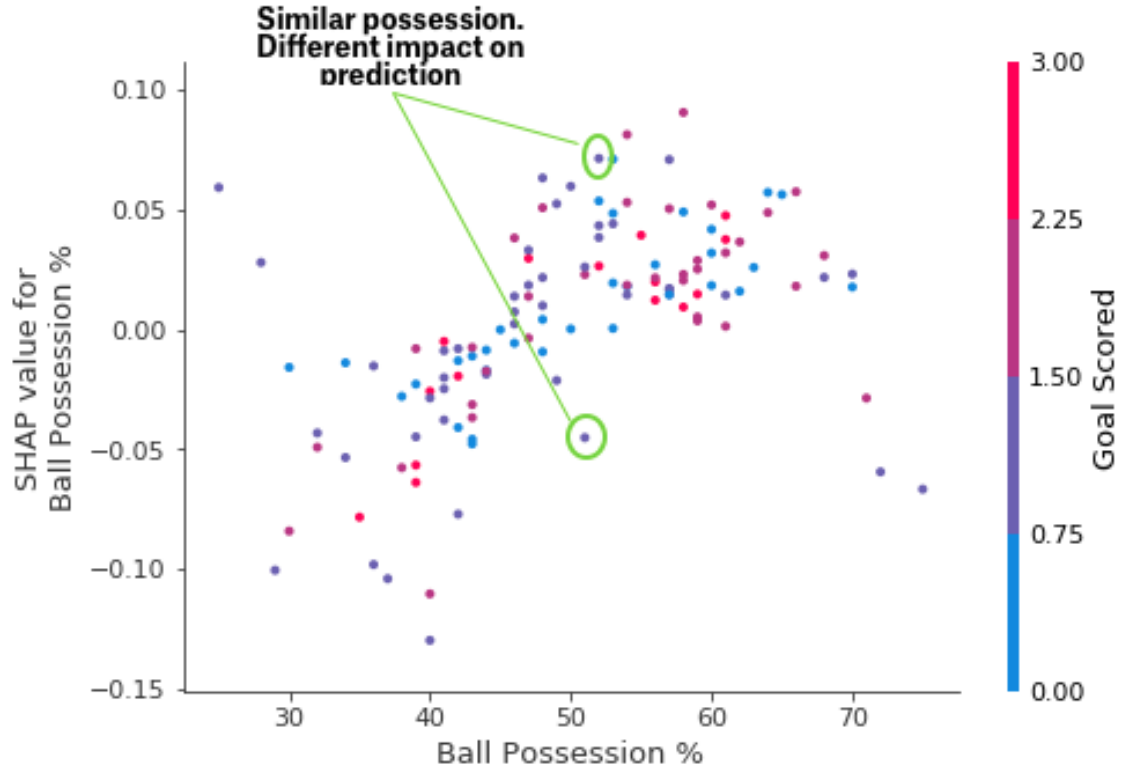
Advance use of Shap Values

Some things you should be able to easily pick out:

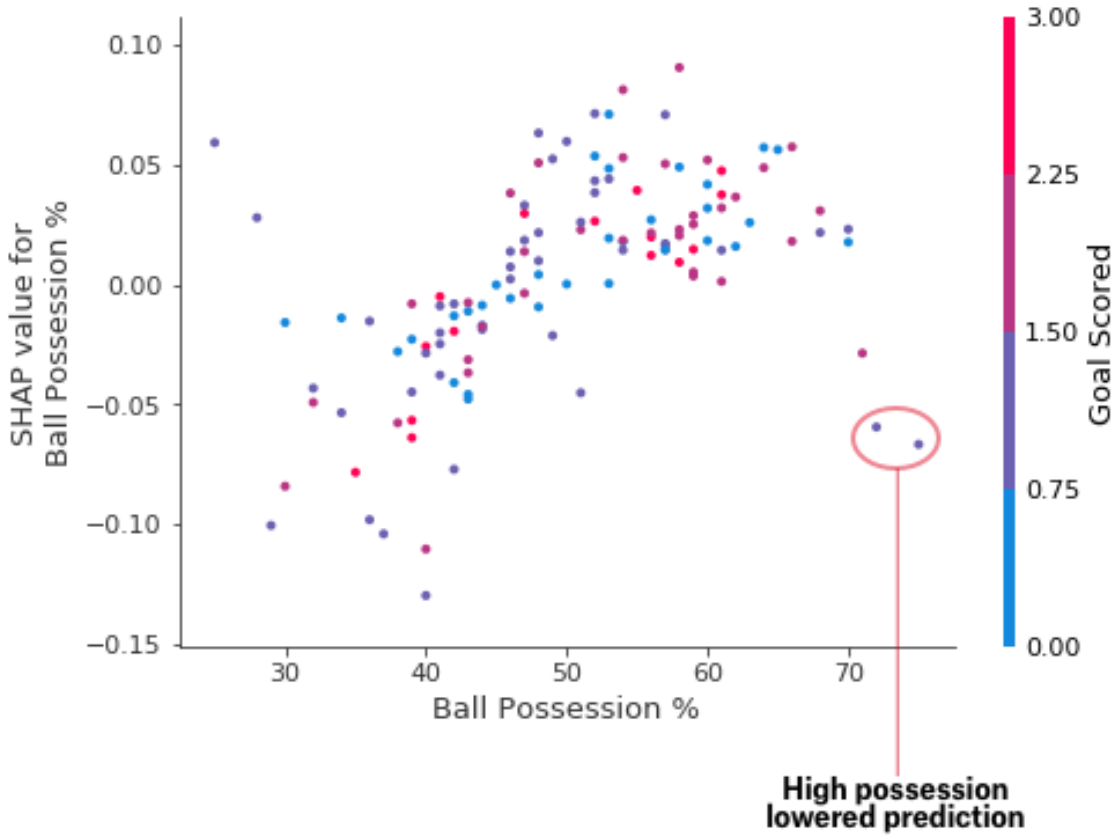
- The model ignored the `Red` and `Yellow & Red` features.
- Usually `Yellow Card` doesn't affect the prediction, but there is an extreme case where a high value caused a much lower prediction.
- High values of Goal scored caused higher predictions, and low values caused low predictions

SHAP Dependence Contribution Plots





The spread suggests that other features must interact with Ball Possession %.



Resource: ***https://www.kaggle.com/learn/machine-learning-explainability***

By Dan Becker