Feature Importance

Learn about feature importance in making model predictions.

Chapter Goals:

- Understand how to measure each dataset feature's importance in making model predictions
- Use the matplotlib pyplot API to save a feature importance plot to a file

A. Determining important features

Not every feature in a dataset is used equally for helping a boosted decision tree make predictions. Certain features are more important than others, and it is useful to figure out which features are the most important.

After training an XGBoost model, we can view the relative (proportional) importance of each dataset feature using the `feature_importances_` property of the model.

The code below prints out the relative feature importances of a model trained on a dataset of 4 features.

```
model = xgb.XGBClassifier()
# predefined data and labels
model.fit(data, labels)

# Array of feature importances
print('Feature importances:\n{}'.format(
  repr(model.feature_importances_)))
```

RUN SAVE RESET

Close

Output 2.237s

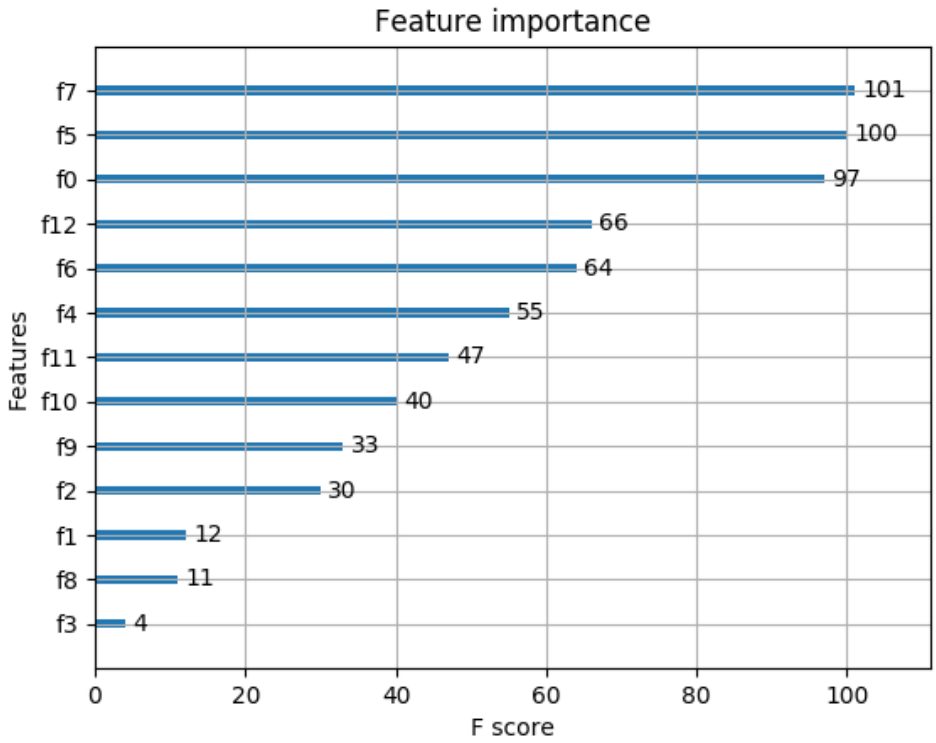
```
Feature importances:
array([0.17941953, 0.11345647, 0.41556728, 0.29155672], dtype=float32)
```

We can plot the feature importances for a model using the `plot_importance` function.

```
model = xgb.XGBRegressor()
# predefined data and labels (for regression)
model.fit(data, labels)

xgb.plot_importance(model)
plt.show() # matplotlib plot
```

RUN SAVE RESET



Plotting the feature importances and showing the plot using matplotlib.pyplot (plt).

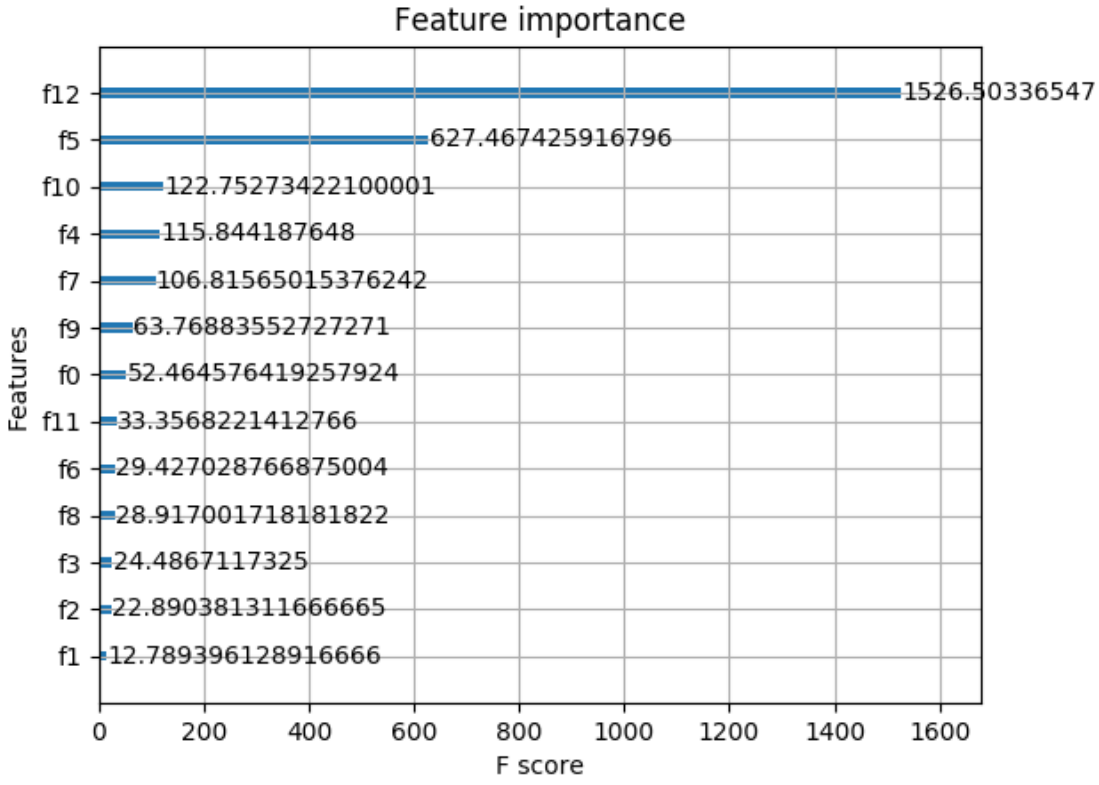
The resulting plot is a bar graph of the F-scores (F_1-scores) for each feature (the number next to each bar is the exact F-score). Note that the features are labeled as "f*N*", where *N* is the index of the column in the dataset. The F-score is a standardized measurement of a feature's importance, based on the specified importance metric.

By default, the `plot_importance` function uses feature *weight* as the importance metric, i.e. how often the feature appears in the boosted decision tree. We can manually choose a different importance metric with the `importance_type` keyword argument.

```
model = xgb.XGBRegressor()
# predefined data and labels (for regression)
model.fit(data, labels)

xgb.plot_importance(model, importance_type='gain')
plt.show() # matplotlib plot
```

RUN SAVE RESET



Plotting the feature importances with information gain as the importance metric

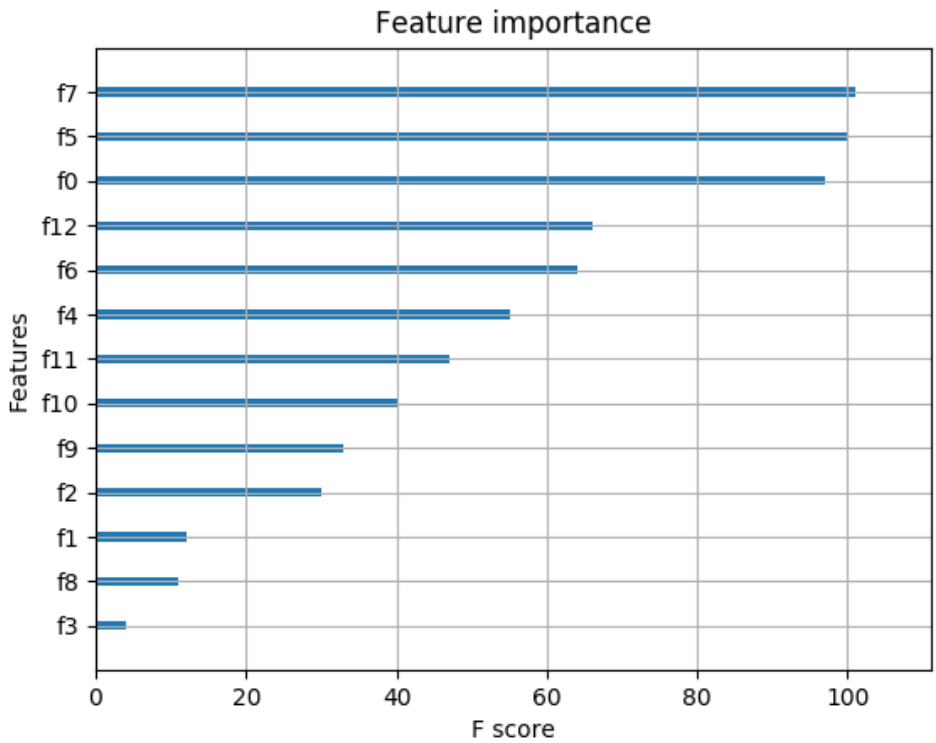
In the code above, we set `importance_type` equal to `'gain'`, which means that we use information gain as the importance metric. Information gain is a commonly used metric for determining how good a feature is at differentiating the dataset, which is important in making predictions with a decision tree.

Finally, if we don't want to show the exact F-score next to each bar, we can set the `show_values` keyword argument to `False`.

```
model = xgb.XGBRegressor()
# predefined data and labels (for regression)
model.fit(data, labels)

xgb.plot_importance(model, show_values=False)
plt.savefig('importances.png') # save to PNG file
```

RUN SAVE RESET



Plotting the feature importances without the exact F-score.