XGBoost Classifier

Create an XGBoost classifier object.

Chapter Goals:

- Learn how to create a scikit-learn style classifier in XGBoost

A. Following the scikit-learn API

While XGBoost provides a more efficient model than scikit-learn, using the model can be a bit convoluted. For people who are used to scikit-learn, XGBoost provides wrapper APIs around its model for classification and regression. These wrapper APIs allow us to use XGBoost's efficient model in the same style as scikit-learn.

For classification, the XGBoost wrapper model is called `XGBClassifier`. Like regular scikit-learn models, it can be trained with a simple call to `fit` with NumPy arrays as input arguments.

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

# new_data contains 2 new data observations
predictions = model.predict(new_data)
print('Predictions:\n{}'.format(repr(predictions)))
```

RUN SAVE RESET

Close

Output 4.079s

```
Predictions:
array([0, 1])
```

Note that the `predict` function for `XGBClassifier` returns actual predictions (not probabilities).

All the parameters for the original `Booster` object are now keyword arguments for the `XGBClassifier`. For instance, we can specify the type of classification, i.e. the `'objective'` parameter for `Booster` objects, with the `objective` keyword argument (the default is binary classification).

```python
model = xgb.XGBClassifier(objective='multi:softmax')
# predefined data and labels (multiclass dataset)
model.fit(data, labels)

# new_data contains 2 new data observations
predictions = model.predict(new_data)
print('Predictions:\n{}'.format(repr(predictions)))
```

RUN SAVE RESET

Close

Output 4.255s

```
Predictions:
array([2, 0])
```