Decision Trees in sklearn

In this section, you'll use decision trees to fit a given sample dataset.

Before you do that, let's go over the tools required to build this model.

For your decision tree model, you'll be using scikit-learn's `Decision Tree Classifier` class. This class provides the functions to define and fit the model to your data.

```
>>> from sklearn.tree import DecisionTreeClassifier
>>> model = DecisionTreeClassifier()
>>> model.fit(x_values, y_values)
```

In the example above, the `model` variable is a decision tree model that has been fitted to the data `x_values` and `y_values`. Fitting the model means finding the best line that fits the training data. Let's make two predictions using the model's `predict()` function.

```
>>> print(model.predict([ [0.2, 0.8], [0.5, 0.4] ]))
[[ 0., 1.]]
```

The model returned an array of predictions, one prediction for each input array. The first input, `[0.2, 0.8]`, got a prediction of `0.`. The second input, `[0.5, 0.4]`, got a prediction of `1.`.

Hyperparameters

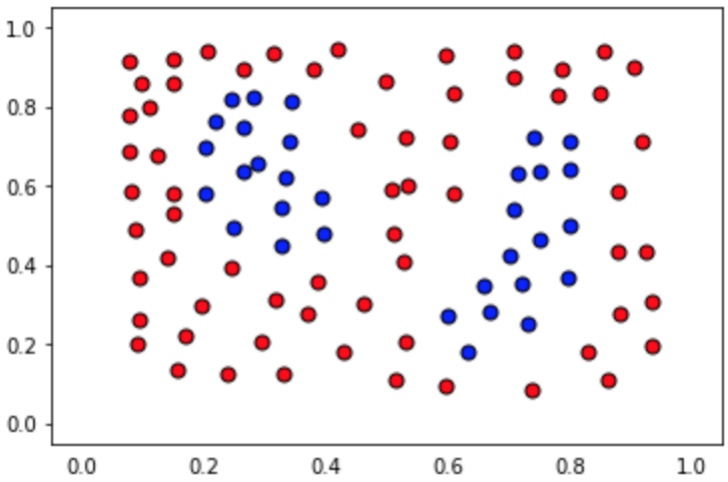
When we define the model, we can specify the hyperparameters. In practice, the most common ones are

- `max_depth`: The maximum number of levels in the tree.
- `min_samples_leaf`: The minimum number of samples allowed in a leaf.
- `min_samples_split`: The minimum number of samples required to split an internal node.
- `max_features`: The number of features to consider when looking for the best split.

```
>>> model = DecisionTreeClassifier(max_depth = 7, min_samples_leaf = 10)
```

Decision Tree Quiz

In this quiz, you'll be given the following sample dataset, and your goal is to define a model that gives 100% accuracy on it.



The data file can be found under the "data.csv" tab in the quiz below. It includes three columns, the first 2 comprising of the coordinates of the points, and the third one of the label.

The data will be loaded for you, and split into features `X` and labels `y`.

You'll need to complete each of the following steps:

1. Build a decision tree model

- Create a decision tree classification model using scikit-learn's `DecisionTree` and assign it to the variable `model`.

2. Fit the model to the data

- You won't need to specify any of the hyperparameters, since the default ones will fit the data with an accuracy of 100% in the dataset. However, we encourage you to play with hyperparameters such as `max_depth` and `min_samples_leaf`, and try to find

Decision Trees in sklearn

- Predict the labels for the training set, and assign this list to the variable `y_pred`.

4. Calculate the accuracy of the model

- For this, use the function sklearn function `accuracy_score`.

When you hit **Test Run**, you'll be able to see the boundary region of your model, which will help you tune the correct parameters, in case you need them.

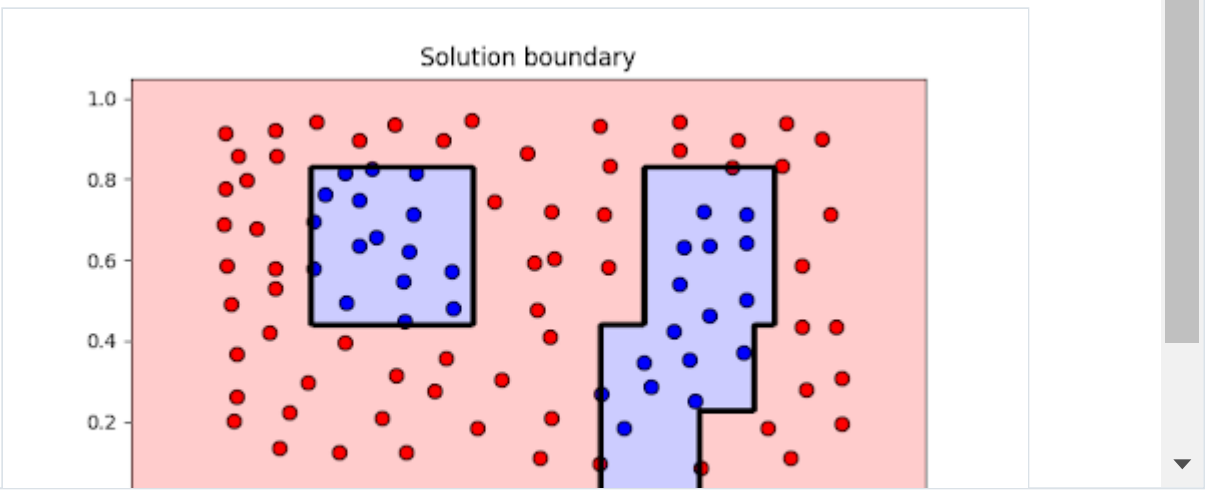
Note: This quiz requires you to find an accuracy of 100% on the training set. Of course, this screams overfitting! If you pick very large values for your parameters, you will fit the training set very well, but it may not be the best model. Try to find the smallest possible parameters that do the job, which has less chance of overfitting, although this part won't be graded.

quiz.py | data.csv | solution.py

```
# Import statements
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

# Read the data.
data = np.asarray(pd.read_csv('data.csv', header=None))
# Assign the features to the variable X, and the labels to the variable y.
X = data[:,0:2]
y = data[:,2]

# TODO: Create the decision tree model and assign it to the variable model.
# You won't need to, but if you'd like, play with hyperparameters such
# as max_depth and min_samples_leaf and see what they do to the decision
# boundary.
model = DecisionTreeClassifier()

# TODO: Fit the model.

model = model.fit(X, y)

# TODO: Make predictions. Store them in the variable y_pred.
y_pred = model.predict(X)

# TODO: Calculate the accuracy and assign it to the variable acc.
acc = accuracy_score(y, y_pred)
```

Decision Trees in sklearn

```
Great job!
```



RESET QUIZ TEST RUN SUBMIT ANSWER

NEXT