6. Random Forest in Scikit-learn

You now have the ability to make a random forest using your own decision trees. However, `scikit-learn` has a `RandomForestClassifier` class that will do all of this work for you! `RandomForestClassifier` is in the `sklearn.ensemble` module.

`RandomForestClassifier` works almost identically to `DecisionTreeClassifier` — the `.fit()`, `.predict()`, and `.score()` methods work in the exact same way.

When creating a `RandomForestClassifier`, you can choose how many trees to include in the random forest by using the `n_estimators` parameter like this:

```
classifier = RandomForestClassifier(n_estimators = 100)
```

We now have a very powerful machine learning model that is fairly resistant to overfitting!

Instructions

1.
Create a `RandomForestClassifier` named `classifier`. When you create it, pass two parameters to the constructor:

- `n_estimators` should be `2000`. Our forest will be pretty big!
- `random_state` should be `0`. There's an element of randomness when creating random forests thanks to bagging. Setting the `random_state` to `0` will help us test your code.

2.
Train the forest using the training data by calling the `.fit()` method. `.fit()` takes two parameters — `training_points` and `training_labels`.

3.
Test the random forest on the testing set and print the results. How accurate was the model?

```
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
from cars import training_points, training_labels, testing_points, testing_labels
import warnings
from sklearn.ensemble import RandomForestClassifier


# 1
classifier = RandomForestClassifier(n_estimators = 2000, random_state = 0)

# 2
classifier.fit(training_points, training_labels)
```

```python
# 3
print(classifier.predict(testing_points))

print(classifier.score(testing_points, testing_labels))
```

```
 'unacc' 'unacc' 'unacc' 'unacc' 'acc' 'vgood' 'vgood'
'vgood' 'unacc'
 'unacc' 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
'unacc' 'acc' 'acc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc'
'vgood' 'unacc'
 'good' 'unacc' 'acc' 'acc' 'unacc' 'acc' 'acc' 'acc'
'acc' 'good' 'unacc'
 'acc' 'unacc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'acc'
'unacc' 'unacc'
 'acc' 'acc' 'good' 'acc' 'acc' 'vgood' 'unacc' 'unacc'
'unacc' 'acc'
 'unacc' 'acc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
'unacc' 'acc' 'acc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'acc'
'acc' 'unacc'
 'vgood' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
'unacc' 'unacc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
'unacc' 'good'
 'unacc' 'good' 'acc' 'unacc' 'unacc' 'unacc' 'unacc'
'unacc' 'acc'
 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc' 'unacc'
'good' 'unacc'
 'acc' 'acc' 'unacc' 'acc' 'unacc' 'acc' 'unacc' 'unacc'
'good' 'good'
 'unacc' 'unacc']
0.9826589595375722
```

Review

Nice work! Here are some of the major takeaways about random forests:

- A random forest is an ensemble machine learning model. It makes a classification by aggregating the classifications of many decision trees.

- Random forests are used to avoid overfitting. By aggregating the classification of multiple trees, having overfitted trees in a random forest is less impactful.

- Every decision tree in a random forest is created by using a different subset of data points from the training set. Those data points are chosen at random *with replacement*, which means a single data point can be chosen more than once. This process is known as *bagging*.

- When creating a tree in a random forest, a randomly selected subset of features are considered as candidates for the best splitting feature. If your dataset has `n` features, it is common practice to randomly select the square root of `n` features.