Cross Validation and Ensemble Techniques for Decision Trees: Takeaways

Syntax

CROSS-VALIDATION

- Using K-Fold Cross-Validation

```
from sklearn.model_selection import cross_val_score
tree = DecisionTree()
print(cross_val_score(tree, X, y))
```

- Getting the mean value of cross-validated scores

```
print(cross_val_score(tree, X, y).mean())
```

- Getting all available predefined scoring methods

```
from sklearn.metrics import get_scorer_names
print(get_scorer_names())
```

- Using alternate scoring methods

```
from sklearn.model_selection import cross_val_score
tree = DecisionTree()
print(cross_val_score(tree, X, y, scoring = < your alternate scorer from get_scorer_names, as
a string >))
```

- Using more than one scoring method with `cross_validate()`

```
from sklearn.model_selection import cross_validate
tree = DecisionTree()
multiple_cross_scores = cross_validate(tree, X, y, scoring= ("accuracy", "recall_macro") )
print(multiple_cross_scores["test_accuracy"]) # Always add a "test_" prefix
print(multiple_cross_scores["test_recall_macro"])
```

- Creating a scoring method with `make_scorer()`

```
from sklearn.metrics import make_scorer, mean_squared_error
rmse = make_scorer(mean_squared_error, squared = False, greater_is_better = False)
regression_tree = DecisionTreeRegressor()
print(cross_val_score(regression_tree, X, y, scoring = rmse))
```

GRID SEARCH AND RANDOMIZED SEARCH

- Using Grid Search

```
from sklearn.model_selection import GridSearchCV
params = {
        "parameter1": < list of attributes to iterate >,
```

```
        "parameter2": < list of attributes to iterate >,
        "parameter3": < list of attributes to iterate >
         }
tree = DecisionTree()
grid_search = GridSearchCV(tree, param_grid = params)
grid_search.fit(X, y)
```

- Using Randomized Search

```
from sklearn.model_selection import RandomizedSearchCV
params = {
        "parameter1": < list of attributes to iterate >,
        "parameter2": < list of attributes to iterate >,
        "parameter3": < list of attributes to iterate >
         }
tree = DecisionTree()
random_search = RandomizedSearchCV(tree, n_iter = 10, param_distributions = params) #
GridSearchCV uses "param_grid" instead
random_search.fit(X, y)
```

- Getting the best parameters, estimator, and score

```
searchcv.best_params_
searchcv.best_estimator_
searchcv.best_score_
```

    - **NOTE:** This applies to both Grid Search and Randomized Search

RANDOM FOREST

- Using random forest

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier # Or Regressor
X_train, X_test, y_train, y_test = train_test_split(X, y)
random_forest_clf = RandomForestClassifier()
random_forest_clf.fit(X_train, y_train)
random_forest_clf.score(X_test, y_test)
random_forest_clf.predict(X_test)
```

- Getting random forest OOB score (out of bag score)

```
from sklearn.ensemble import RandomForestClassifier # Or Regressor
random_forest_clf = RandomForestClassifier(bootstrap = True, oob_score = True)
random_forest_clf.fit(X, y) # We don't need train_test_split here
random_forest_clf.oob_score_
```

- Getting RandomForestRegressor predictions when using OOB score

```
from sklearn.ensemble import RandomForestRegressor
random_forest_reg = RandomForestRegressor(bootstrap = True, oob_score = True)
random_forest_reg.fit(X, y) # We don't need train_test_split here
random_forest_reg.oob_prediction_
```

- Getting RandomForestClassifier predictions when using OOB score

```
from sklearn.ensemble import RandomForestClassifier
random_forest_clf = RandomForestClassifier(bootstrap = True, oob_score = True)
random_forest_clf.fit(X, y) # We don't need train_test_split here
random_forest_clf.oob_decision_function_
```

EXTRA TREES

- Using Extra Trees

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesClassifier # Or Regressor
X_train, X_test, y_train, y_test = train_test_split(X, y)
extra_clf = ExtraTreesClassifier()
extra_clf.fit(X_train, y_train)
extra_clf.score(X_test, y_test)
extra_clf.predict(X_test)
```

Concepts

- To perform CV, `scikit-learn` uses the `cross_val_score` tool. As default scores, it uses R^2 for Regression Trees, and Accuracy for Classification Trees. When using CV, it isn't necessary to fit the Decision Tree or use `train_test_split`.
- It's possible to use alternative scores in `cross_val_score`, via the `scoring` parameter, which accepts a string from the following list of predefined scorers or from `get_scorer_names`.
- The predefined scorers assume that higher values indicate better quality than lower ones, so when dealing with error metrics (like the Mean Squared Error) these are automatically transformed into negative values so that higher values indicate a smaller error and lower values indicate a larger error.
- `cross_val_score` can only accept one metric, therefore to use multiple metrics at the same time, use the complementary `cross_validate` tool.
- If the score you want to use doesn't exist on the predefined list, or if you want to apply one of the predefined scores but with some specific parameters, use the `make_scorer` tool. If you're dealing with error metrics, set the parameter `greater_is_better = False` to negate the metrics, so that the rule "higher is better" is respected.
- If we have enough computational power we can apply `GridSearchCV` to our Decision Trees, which allows us to exhaustively try all the possible combinations from a list of selected values for every parameter. In this way, we avoid a lot of manual tweaking when looking for optimal parameters at the expense of waiting several hours (or even more than a day) until we get an output.
- If we don't have that much time or computational power, we can use `RandomizedSearchCV`, which only tries a selected number from all the possible combinations.
- **Random Forest** is part of the bigger family of **ensemble methods**, where several individual Machine Learning models are combined to get more robust scores and predictions. However, unlike the rest of ensemble methods, Random Forest is exclusively for Decision Trees.

- `scikit-learn` includes two Random Forest instantiators: `RandomForestClassifier` and `RandomForestRegressor`. They share the same parameters as Decision Trees, but also they have some additional ones:
    - `n_estimators`: The number of Trees that our Random Forest will generate. The default value is `100` estimators.
    - `bootstrap`: If we deactivate bootstrapping, the whole dataset will be used to generate each Tree.
    - `n_jobs`: The number of computational processing jobs that run in parallel when performing all the calculations. If we want to use all the available processors, we use `-1` as an argument.
    - `verbose`: Prints logs that describe the operations that the algorithm performs in each step. Although `scikit-learn` doesn't specify what each number prints, the higher the integer value, the more detailed the logs will be. The default is `0` (no logs are printed).
    - `warm_start`: If we're using the same dataset, and we want to find the optimal number of estimators (for instance, as part of a Grid Search), this boolean parameter allow us to keep some learned aspects from the previous estimator so that we can use them on the next one, saving computation time.
    - `max_samples`: Specifies the number of observations to train each Tree. It requires `bootstrap` to be set to `True`.
- **Extra Trees** (which stand for "Extremely Randomized Trees") work exactly the same as Random Forest, except for two fundamental differences to increase their level of randomness: instead of using bootstrap aggregating ("Bagging") every Tree uses the whole dataset; and secondly, the thresholds are decided at random, and only the most optimal threshold of this random selection will be chosen as a split.
- Just like Random Forest, Extra Trees have two versions: an `ExtraTreesRegressor` and an `ExtraTreesClassifier`. They have exactly the same parameters as Random Forest.
- Among the main advantages of Decision Trees, they're easier to understand compared to other Machine Learning algorithms, we can visualize their insights and even show them to non-technical audiences, and they are suitable for both Regression and Classification problems. Also, unlike other models, like Linear Regression, they don't make assumptions about the data.
- The main disadvantage of Decision Trees is their tendency to overfit the data, something that will force us to spend additional time tweaking their parameters when we build them with the `scikit-learn` library. In addition, aside from the considerable computational time required when using a large dataset, they are also extremely sensitive to small changes in the dataset — although this weakness is mitigated when we use either Random Forest or Extra Trees, as these models take advantage of this oversensitivity.

Resources

CROSS-VALIDATION

- Cross-Validation - Wikipedia
- Cross-Validation - scikit-learn

- cross_val_score
    - cv
- Alternate scoring options:
    - scoring
    - Using simple metric evaluation
    - Using multiple metric evaluation
    - List of predefined scoring methods
    - get_scorer_names
    - cross_validate
        - The cross_validate function
    - make_scorer
        - scorer
        - Defining your scoring strategy from metric functions
        - Implementing your own scorers
- Alternate Cross-Validation options:
    - All available options
    - StratifiedKFold
    - RepeatedKFold
    - LeavePOut
    - cross_val_predict
        - Obtaining predictions with cross_val_predict
        - Using cross_val_predict

GRID SEARCH & RANDOMIZED SEARCH

- Hyperparameter Optimization - Wikipedia:
    - Grid Search
    - Random Search
- scikit-learn - Tuning the hyperparameters of an estimator
- GridSearchCV
    - Exhaustive Grid Search
- RandomizedSearchCV
    - RandomizedParameterOptimization
- Specifying an objective metric
- Specifying multiple metrics for evaluation

RANDOM FOREST

- Wikipedia:
    - Ensemble Learning

        - Random Forest
        - Bootstrapping
        - Bootstrap Aggregating / Bagging
    - Ensemble Methods - scikit-learn
    - Forest of Randomized Trees:
        - Random Forest
    - RandomForestRegressor
    - RandomForestClassifier
        - decision_function
    - RandomForest Parameters:
        - Parallelization
            - n_jobs
        - verbose
        - warm_start

EXTRA TREES

- Extra Trees - Wikipedia
- Extremely Randomized Trees - scikit-learn:
    - ExtraTreesRegressor
    - ExtraTreesClassifier
- Avoid confusions:
    - ExtraTreeRegressor
    - ExtraTreeClassifier

MISCELLANEOUS

- Decision Trees - Tips on Practical Use
- Alternate Decision Tree algorithms
- Decision Trees - Mathematical formulations:
    - Mathematical formulation for Classification Trees
    - Mathematical formulation for Regression Trees
- Other ensemble methods # NOTE: They aren't exclusive to Decision Trees