Introduction to hyperparameter tuning

MODEL VALIDATION IN PYTHON

Kasey Jones
Data Scientist

Model parameters

Parameters are:

- Learned or estimated from the data
- The result of fitting a model
- Used when making future predictions
- Not manually set

Linear regression parameters

Parameters are created by fitting a model:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X, y)
print(lr.coef_, lr.intercept_)
```

```
[[0.798, 0.452]] [1.786]
```

Linear regression parameters

Parameters do not exist before the model is fit:

```
lr = LinearRegression()
print(lr.coef_, lr.intercept_)
```

```
AttributeError: 'LinearRegression' object has no attribute 'coef_'
```

Model hyperparameters

Hyperparameters:

- Manually set *before* the training occurs
- Specify how the training is supposed to happen

Random forest hyperparameters

Hyperparameter	Description	Possible Values (default)
n_estimators	Number of decision trees in the forest	2+ (10)
max_depth	Maximum depth of the decision trees	2+ (None)
max_features	Number of features to consider when making a split	See documentation
min_samples_split	The minimum number of samples required to make a split	2+ (2)

What is hyperparameter tuning?

Hyperparameter tuning:

- Select hyperparameters
- Run a single model type at different value sets
- Create ranges of possible values to select from
- Specify a single accuracy metric

Specifying ranges

```
depth = [4, 6, 8, 10, 12]
samples = [2, 4, 6, 8]
features = [2, 4, 6, 8, 10]
# Specify hyperparameters
rfc = RandomForestRegressor(
    n_estimators=100, max_depth=depth[0],
    min_samples_split=samples[3], max_features=features[1])
rfr.get_params()
```

```
{'bootstrap': True,
 'criterion': 'mse'
 ...
}
```

Too many hyperparameters!

```
rfr.get_params()
```

```
{'bootstrap': True,
 'criterion': 'mse',
 'max_depth': 4,
 'max_features': 4,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 8,
 ...
 }
```

General guidelines

- Start with the basics
- Read through the documentation
- Test practical ranges

Let's practice!

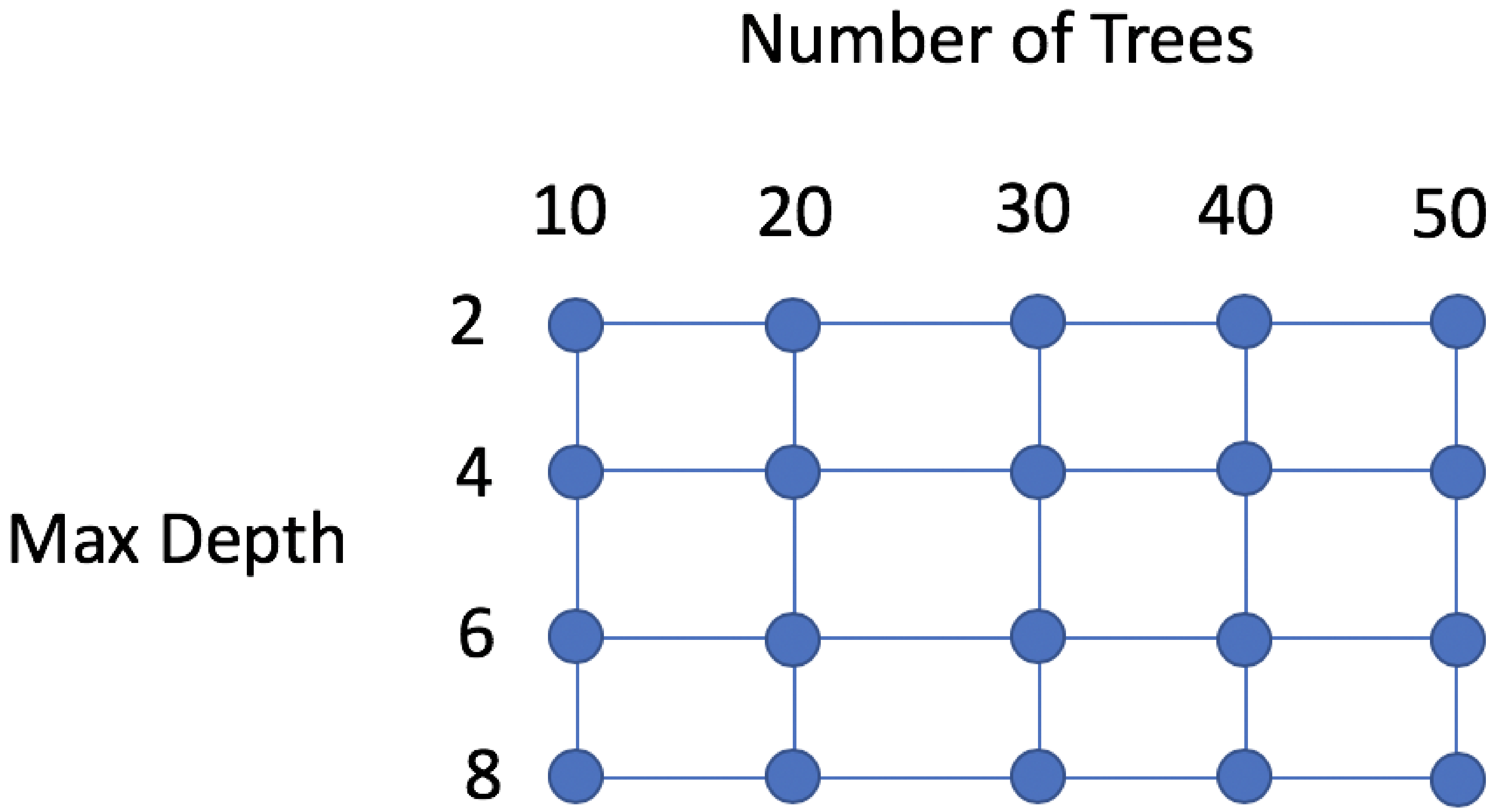
MODEL VALIDATION IN PYTHON

RandomizedSearchCV

MODEL VALIDATION IN PYTHON

Kasey Jones
Data Scientist

Grid searching hyperparameters



Grid searching continued

Benefits:

- Tests every possible combination

Drawbacks:

- Additional hyperparameters increase training time exponentially

Better methods

- Random searching
- Bayesian optimization

Random search

```
from sklearn.model_selection import RandomizedSearchCV


random_search = RandomizedSearchCV()
```

Parameter Distribution:

```
param_dist = {"max_depth": [4, 6, 8, None],
              "max_features": range(2, 11),
              "min_samples_split": range(2, 11)}
```

Random search parameters

Parameters:

- `estimator` : the model to use
- `param_distributions` : dictionary containing hyperparameters and possible values
- `n_iter` : number of iterations
- `scoring` : scoring method to use

Setting RandomizedSearchCV parameters

```
param_dist = {"max_depth": [4, 6, 8, None],
              "max_features": range(2, 11),
              "min_samples_split": range(2, 11)}
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import make_scorer, mean_absolute_error

rfr = RandomForestRegressor(n_estimators=20, random_state=1111)
scorer = make_scorer(mean_absolute_error)
```

RandomizedSearchCV implemented

Setting up the random search:

```
random_search =\
    RandomizedSearchCV(estimator=rfr,
                       param_distributions=param_dist,
                       n_iter=40,
                       cv=5)
```

- We cannot do hyperparameter tuning without understanding model validation
- Model validation allows us to compare multiple models and parameter sets

RandomizedSearchCV implemented

Setting up the random search:

```
random_search =\
    RandomizedSearchCV(estimator=rfr,
                       param_distributions=param_dist,
                       n_iter=40,
                       cv=5)
```

Complete the random search:

```
random_search.fit(X, y)
```

Let's explore some examples!

MODEL VALIDATION IN PYTHON

Selecting your final model

MODEL VALIDATION IN PYTHON

Kasey Jones
Data Scientist

```
# Best Score
rs.best_score_
```

```
5.45
```

```
# Best Parameters
rs.best_params_
```

```
{'max_depth': 4, 'max_features': 8, 'min_samples_split': 4}
```

```
# Best Estimator
rs.best_estimator_
```

Other attributes

```
rs.cv_results_
rs.cv_results_['mean_test_score']
```

```
array([5.45, 6.23, 5.87, 5,91, 5,67])
```

```
# Selected Parameters:
rs.cv_results_['params']
```

```
[{'max_depth': 10, 'min_samples_split': 8, 'n_estimators': 25},
 {'max_depth': 4, 'min_samples_split': 8, 'n_estimators': 50},
 ...]
```

Using .cv_results_

Group the max depths:

```
max_depth = [item['max_depth'] for item in rs.cv_results_['params']]
scores = list(rs.cv_results_['mean_test_score'])
d = pd.DataFrame([max_depth, scores]).T
d.columns = ['Max Depth', 'Score']
d.groupby(['Max Depth']).mean()
```

```
Max Depth  Score
2.0        0.677928
4.0        0.753021
6.0        0.817219
8.0        0.879136
10.0       0.896821
```

Other attributes continued

Uses of the output:

- Visualize the effect of each parameter
- Make inferences on which parameters have big impacts on the results

```
Max Depth   Score
2.0         0.677928
4.0         0.753021
6.0         0.817219
8.0         0.879136
10.0        0.896821
```

Selecting the best model

`rs.best_estimator_` contains the information of the best model

```
rs.best_estimator_
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=8,
           max_features=8, max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=12, min_weight_fraction_leaf=0.0,
           n_estimators=20, n_jobs=1, oob_score=False, random_state=1111,
           verbose=0, warm_start=False)
```

Comparing types of models

Random forest:

```
rfr.score(X_test, y_test)
```

```
6.39
```

Gradient Boosting:

```
gb.score(X_test, y_test)
```

```
6.23
```

Predict new data:

```
rs.best_estimator_.predict(<new_data>)
```

Check the parameters:

```
random_search.best_estimator_.get_params()
```

Save model for use later:

```
from sklearn.externals import joblib

joblib.dump(rfr, 'rfr_best_<date>.pkl')
```

Let's practice!

MODEL VALIDATION IN PYTHON

Course completed!

MODEL VALIDATION IN PYTHON

Kasey Jones
Data Scientist

Course recap

Some topics covered:

- Accuracy/evaluation metrics
- Splitting data into train, validation, and test sets
- Cross-validation and LOOCV
- Hyperparameter tuning

Next steps

Check out **kaggle**

Next steps

- Hyperparameter Tuning in Python
- Introduction to Deep Learning in Python

Thank you!

MODEL VALIDATION IN PYTHON