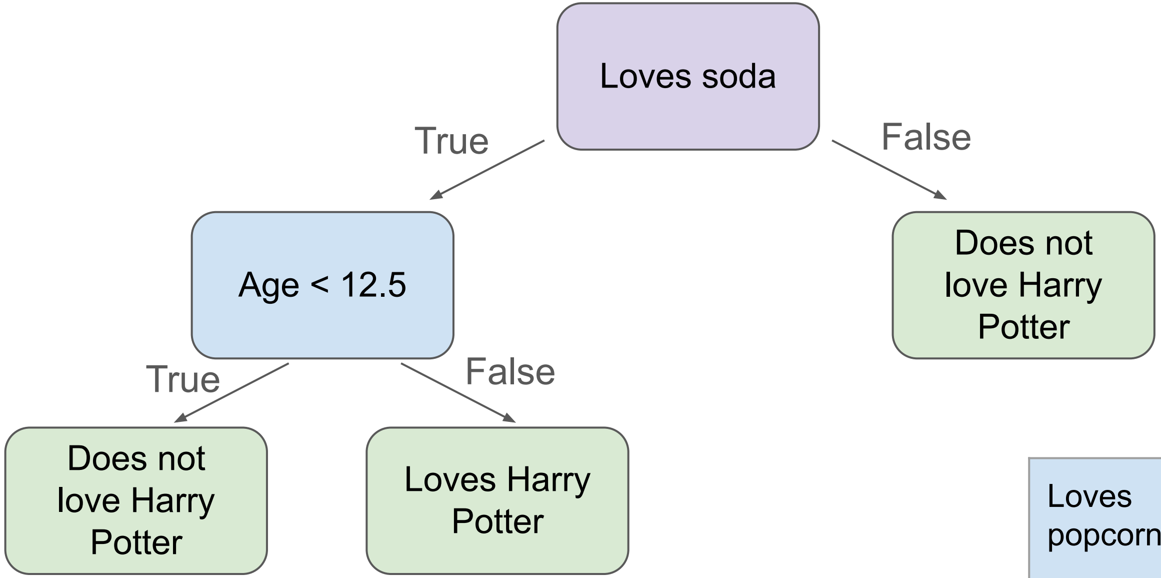
Random Forest

Presentation by Sarah Vastani

Decision Trees Recap



Loves soda

True

False

Age < 12.5

Does not love Harry Potter

True

False

Does not love Harry Potter

Loves Harry Potter

Loves popcorn	Loves soda	Age	Loves Harry Potter
Yes	Yes	15	???

Yes

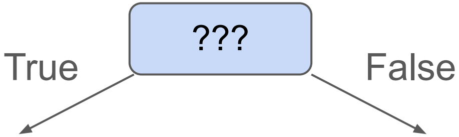
Step 1: Create a Bootstrap Dataset

Original Dataset

	Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
1	No	No	No	125	No
2	Yes	Yes	Yes	180	Yes
3	Yes	Yes	No	210	No
4	Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes



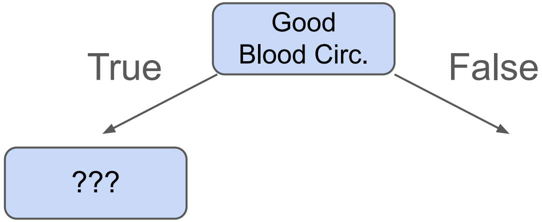
???

True

False

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes



Good Blood Circ.

True

False

???

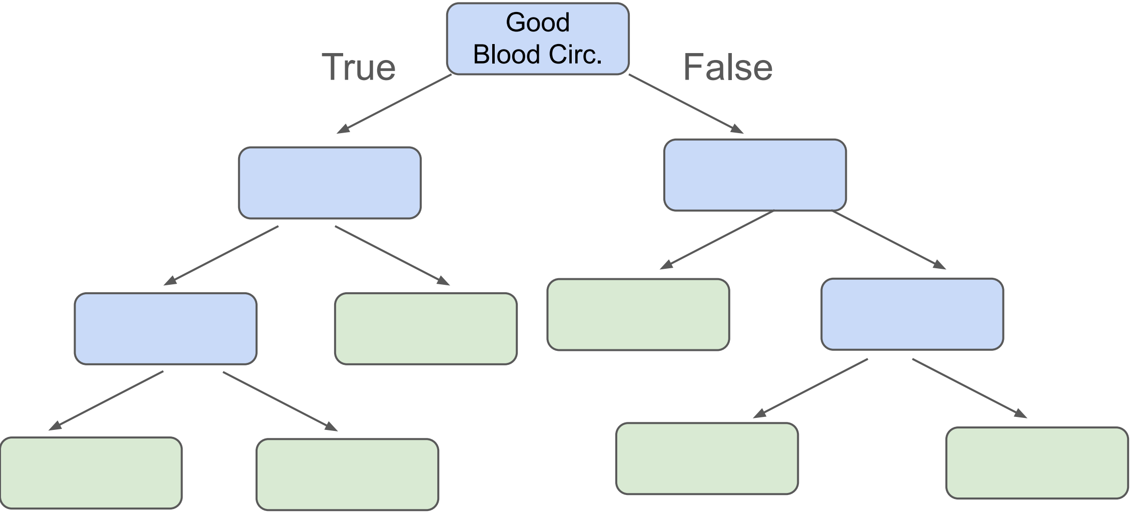
Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

→

Bootstrapped Dataset

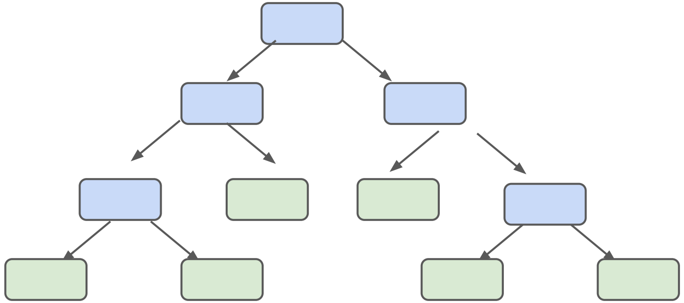
Chest Pain	Blocked Arteries	Weight	Heart Disease
Yes	Yes	180	Yes
No	No	125	No
Yes	Yes	167	Yes
Yes	Yes	167	Yes

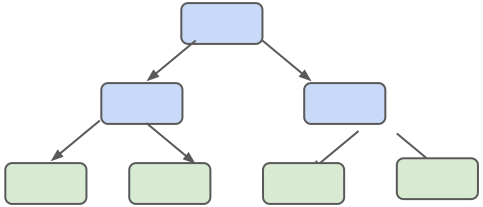


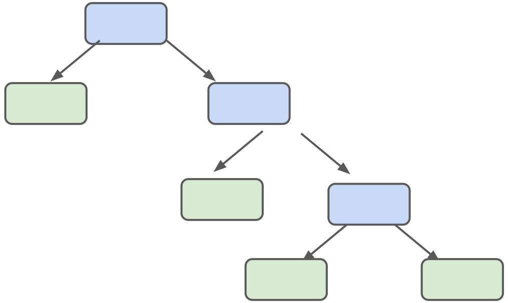
Bootstrapped Dataset

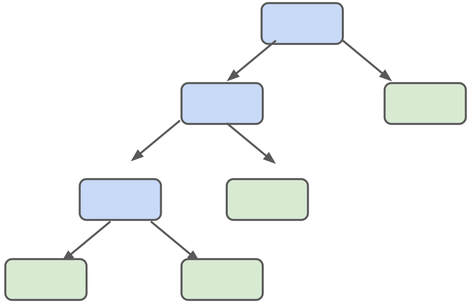
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

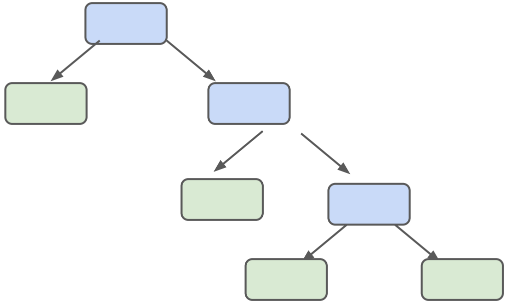
Now, make a new bootstrapped dataset
Randomly pick another subset of columns

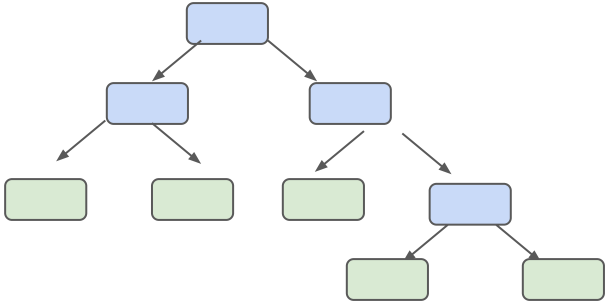






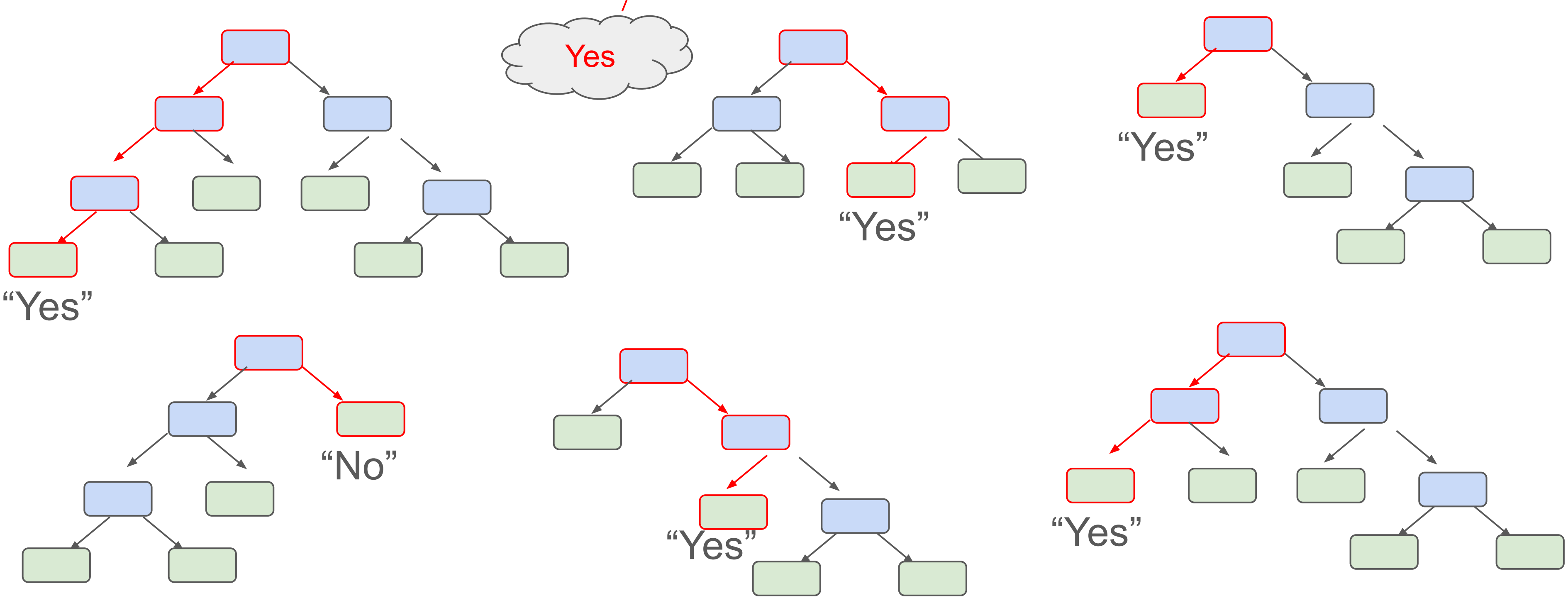






Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	???

Heart Disease	
Yes	No
5	1



Original Dataset

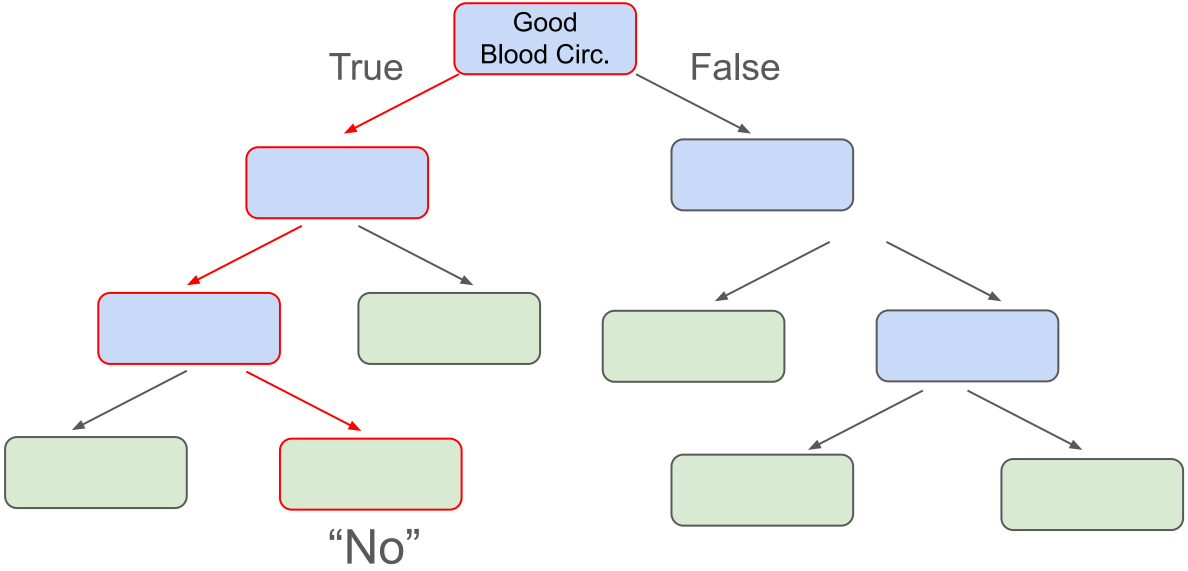
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

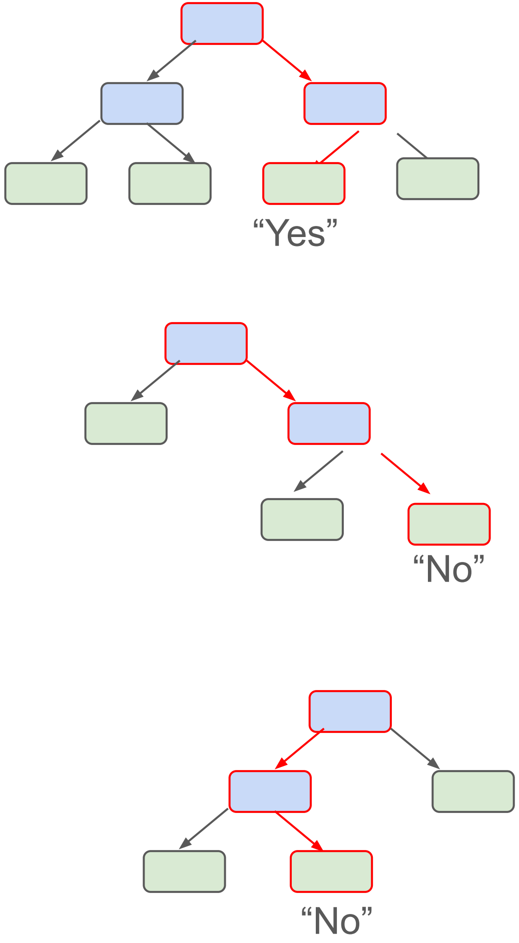
“Out-Of-Bag Dataset”

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No



“Out-Of-Bag Dataset”

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No



“Out-Of-Bag Dataset”

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

Heart Disease

Yes	No
1	3

Heart Disease	
Yes	No
1	3

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

Heart Disease	
Yes	No
4	0

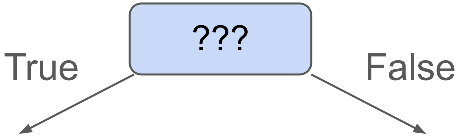
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes

Heart Disease	
Yes	No
4	0

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No

⋮

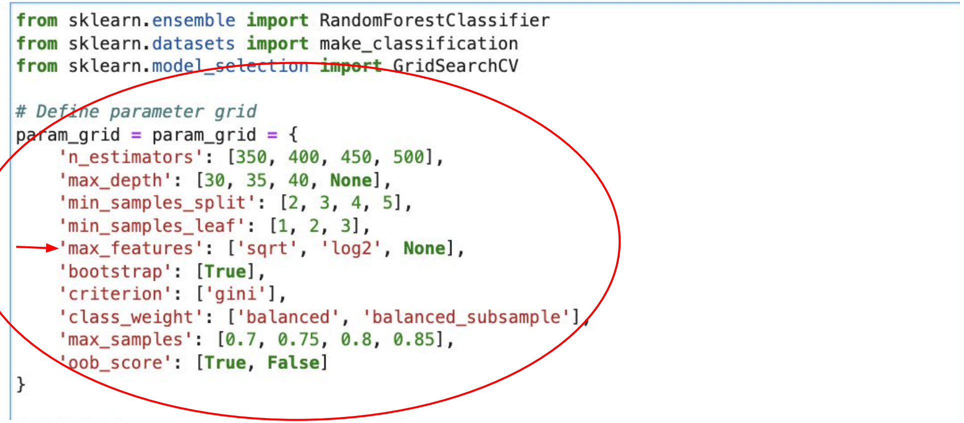
And so on …



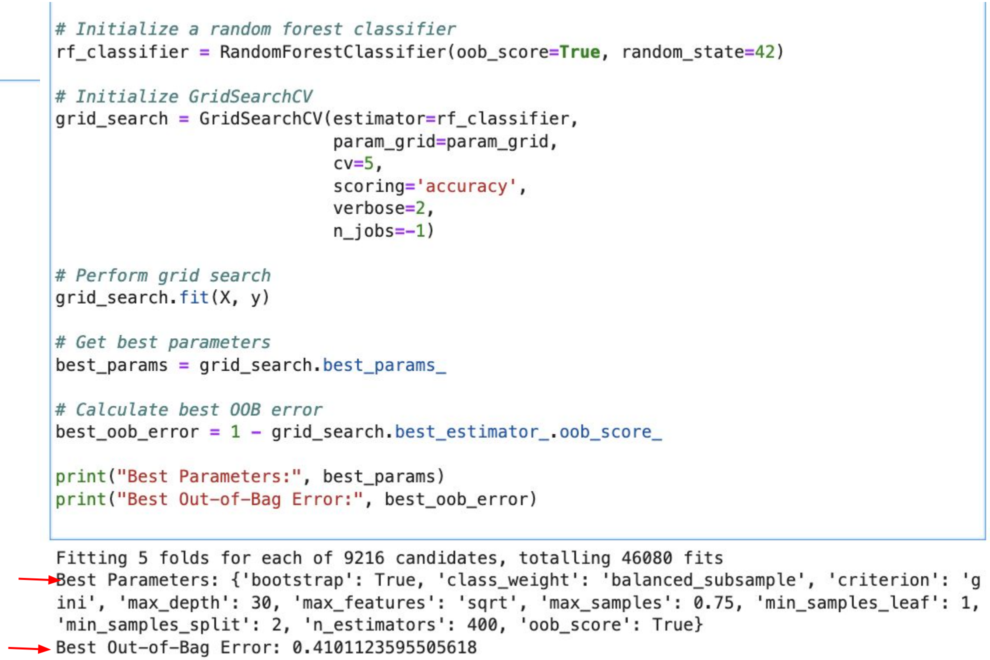
Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Parameter Tuning



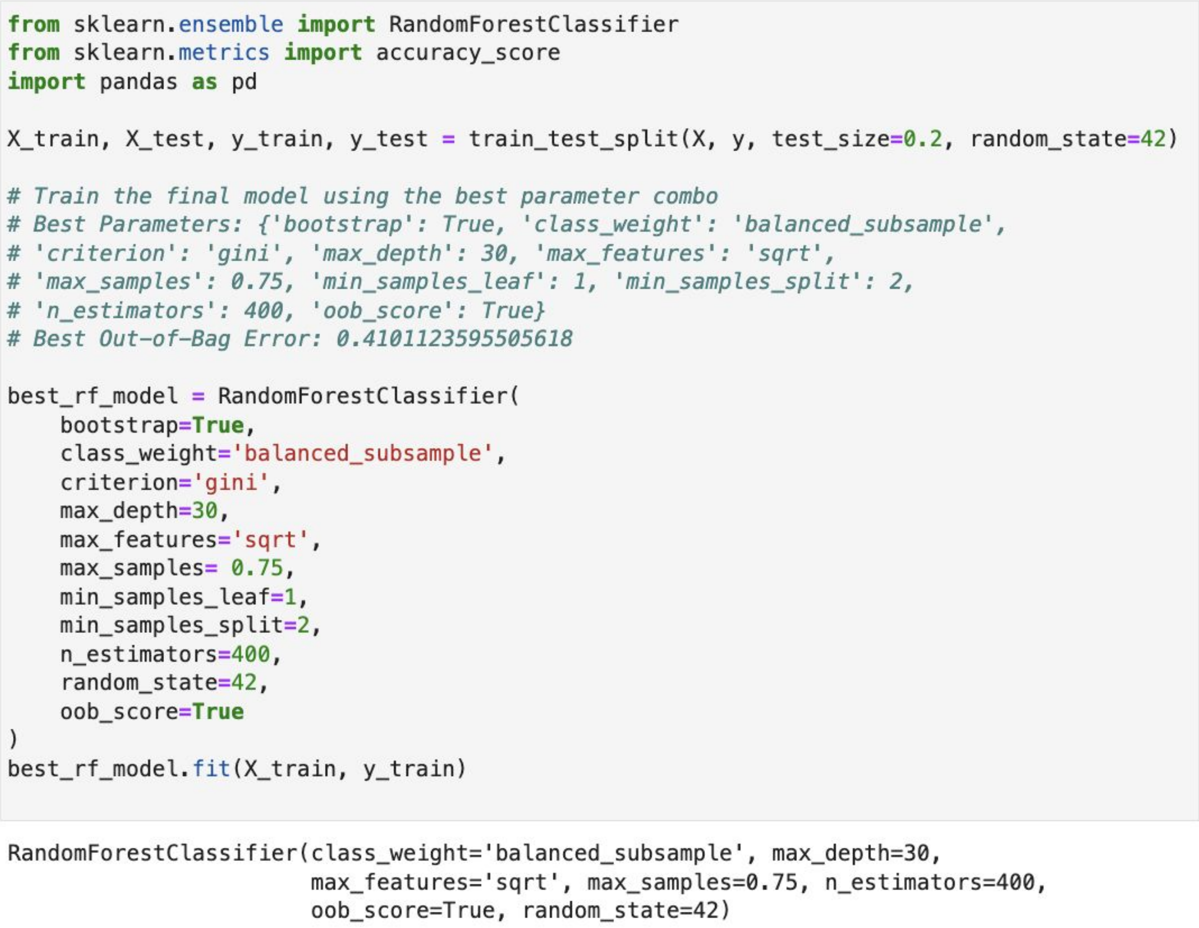
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV

# Define parameter grid
param_grid = param_grid = {
    'n_estimators': [350, 400, 450, 500],
    'max_depth': [30, 35, 40, None],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3],
    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True],
    'criterion': ['gini'],
    'class_weight': ['balanced', 'balanced_subsample'],
    'max_samples': [0.7, 0.75, 0.8, 0.85],
    'oob_score': [True, False]
}
```



```
# Initialize a random forest classifier
rf_classifier = RandomForestClassifier(oob_score=True, random_state=42)

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=rf_classifier,
                           param_grid=param_grid,
                           cv=5,
                           scoring='accuracy',
                           verbose=2,
                           n_jobs=-1)

# Perform grid search
grid_search.fit(X, y)

# Get best parameters
best_params = grid_search.best_params_

# Calculate best OOB error
best_oob_error = 1 - grid_search.best_estimator_.oob_score_

print("Best Parameters:", best_params)
print("Best Out-of-Bag Error:", best_oob_error)
```

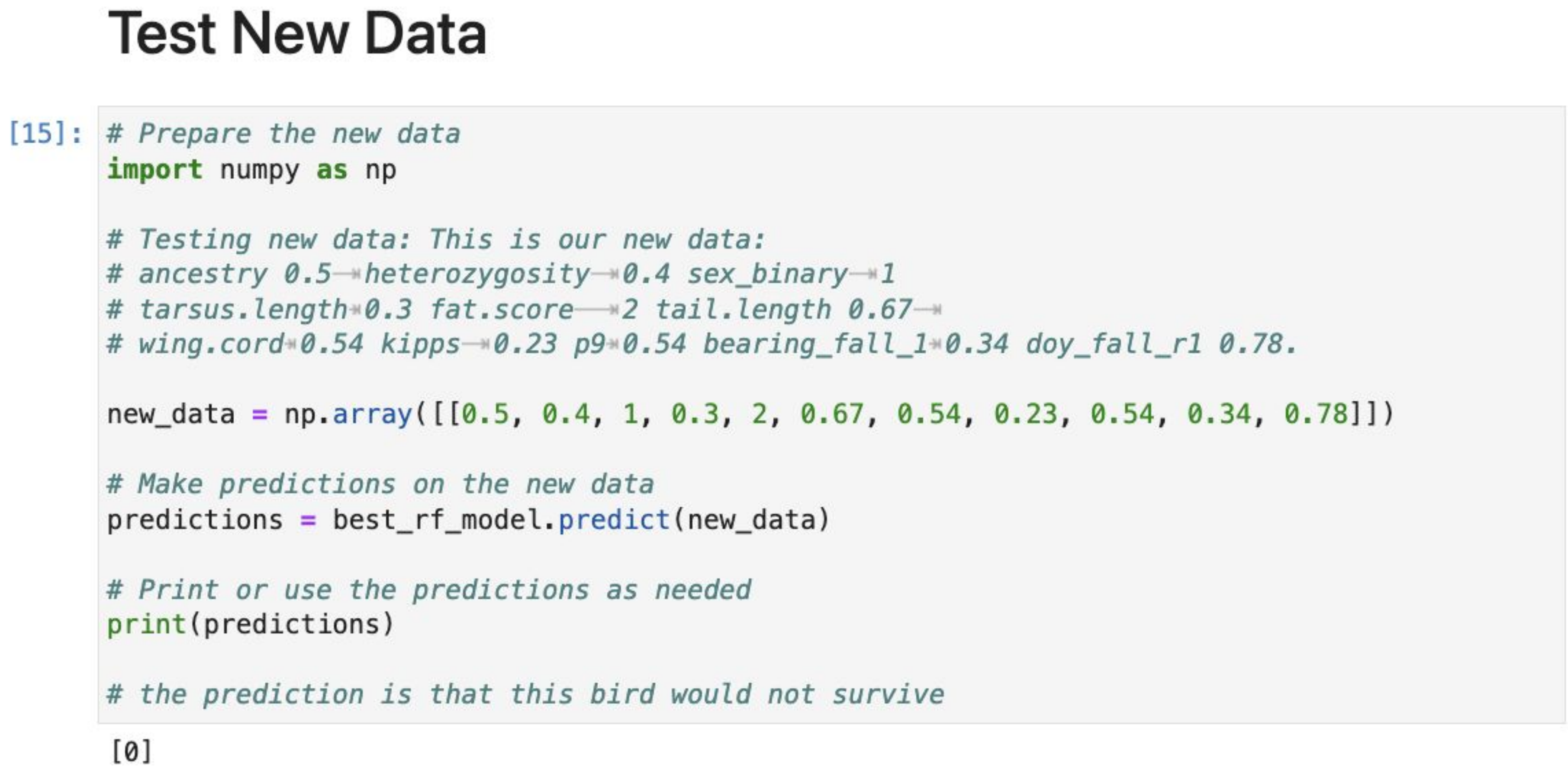
```
Fitting 5 folds for each of 9216 candidates, totalling 46080 fits
Best Parameters: {'bootstrap': True, 'class_weight': 'balanced_subsample', 'criterion': 'g
ini', 'max_depth': 30, 'max_features': 'sqrt', 'max_samples': 0.75, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 400, 'oob_score': True}
Best Out-of-Bag Error: 0.4101123595505618
```

Training the Model



```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the final model using the best parameter combo
# Best Parameters: {'bootstrap': True, 'class_weight': 'balanced_subsample',
# 'criterion': 'gini', 'max_depth': 30, 'max_features': 'sqrt',
# 'max_samples': 0.75, 'min_samples_leaf': 1, 'min_samples_split': 2,
# 'n_estimators': 400, 'oob_score': True}
# Best Out-of-Bag Error: 0.4101123595505618

best_rf_model = RandomForestClassifier(
    bootstrap=True,
    class_weight='balanced_subsample',
    criterion='gini',
    max_depth=30,
    max_features='sqrt',
    max_samples= 0.75,
    min_samples_leaf=1,
    min_samples_split=2,
    n_estimators=400,
    random_state=42,
    oob_score=True
)
best_rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier(class_weight='balanced_subsample', max_depth=30,
                       max_features='sqrt', max_samples=0.75, n_estimators=400,
                       oob_score=True, random_state=42)
```



Test New Data

[15]:

```
# Prepare the new data
import numpy as np

# Testing new data: This is our new data:
# ancestry 0.5	heterozygosity	0.4 sex_binary	1
# tarsus.length	0.3 fat.score	2 tail.length 0.67	
# wing.cord	0.54 kipps	0.23 p9	0.54 bearing_fall_1	0.34 doy_fall_r1 0.78.

new_data = np.array([[0.5, 0.4, 1, 0.3, 2, 0.67, 0.54, 0.23, 0.54, 0.34, 0.78]])

# Make predictions on the new data
predictions = best_rf_model.predict(new_data)

# Print or use the predictions as needed
print(predictions)

# the prediction is that this bird would not survive
```

```
[0]
```