MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Bagging

Elie Kawerk
Data Scientist

Ensemble Methods

Voting Classifier

- same training set,
- $\neq$ algorithms.

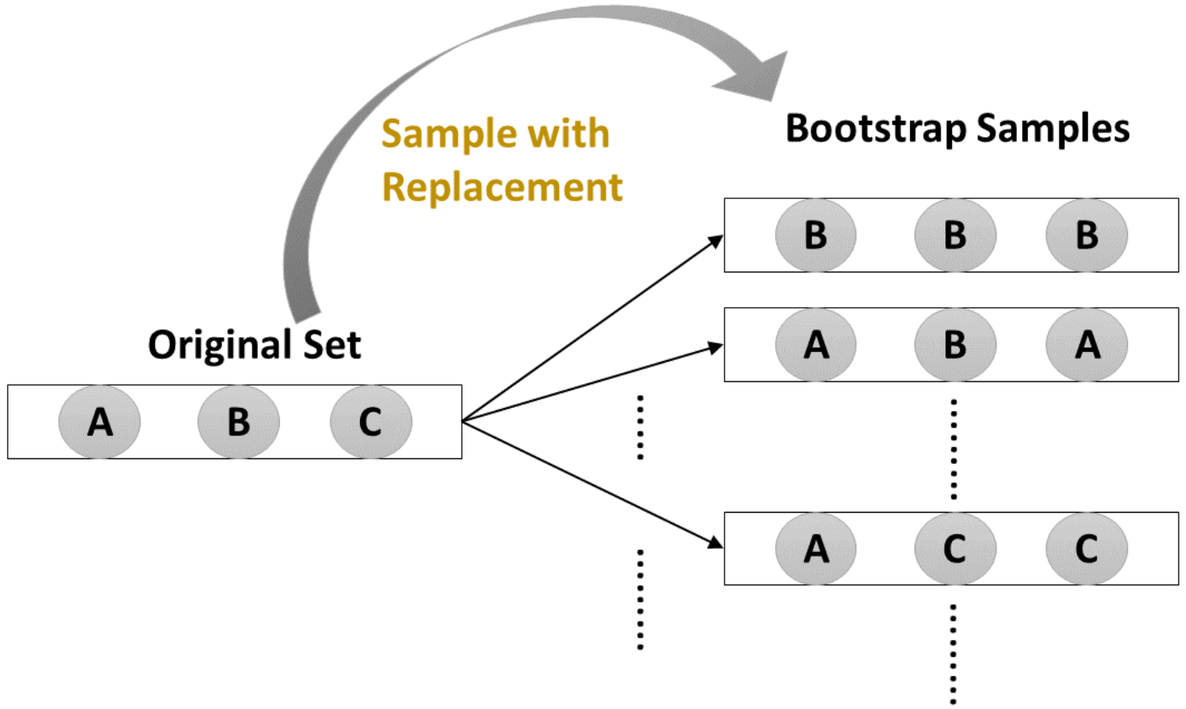
Bagging

- one algorithm,
- $\neq$ subsets of the training set.

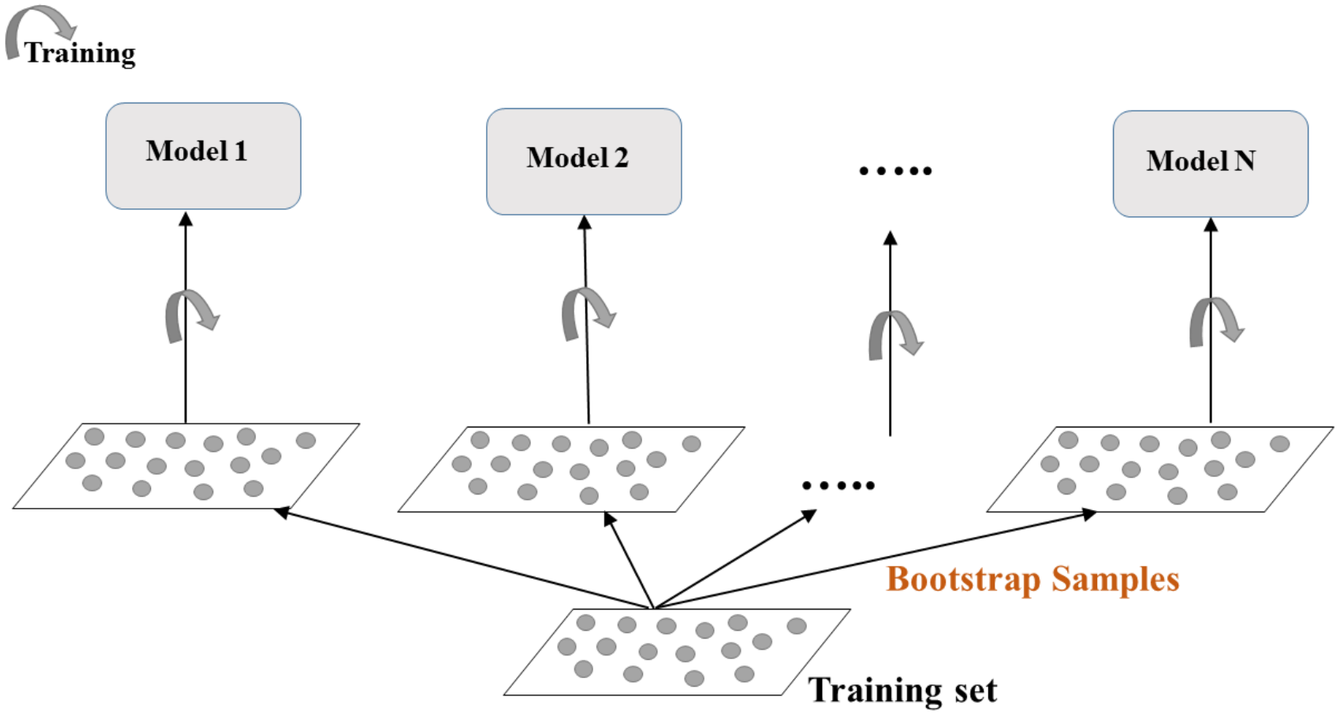
Bagging

- Bagging: Bootstrap Aggregation.
- Uses a technique known as the bootsrap.
- Reduces variance of individual models in the ensemble.

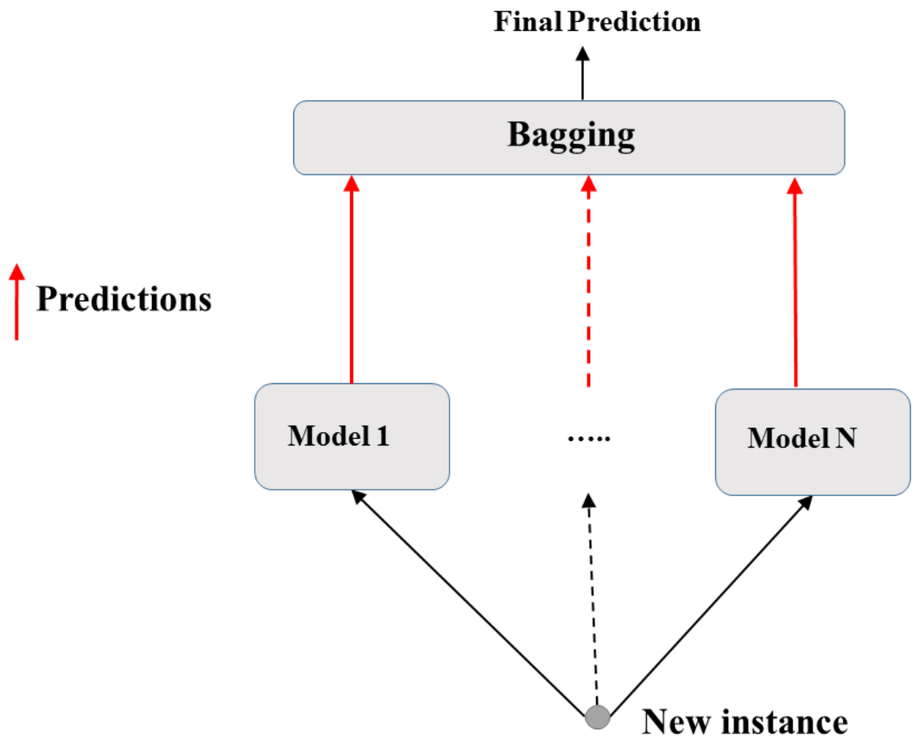
Bootstrap


Sample with Replacement
Bootstrap Samples
Original Set
A B C
B B B
A B A
A C C

Bagging: Training



Bagging: Prediction



Bagging: Classification & Regression

Classification:

- Aggregates predictions by majority voting.
- `BaggingClassifier` in scikit-learn.

Regression:

- Aggregates predictions through averaging.
- `BaggingRegressor` in scikit-learn.

Bagging Classifier in sklearn (Breast-Cancer dataset)

```
# Import models and utility functions
In [1]: from sklearn.ensemble import BaggingClassifier
In [2]: from sklearn.tree import DecisionTreeClassifier
In [3]: from sklearn.metrics import accuracy_score
In [4]: from sklearn.model_selection import train_test_split

# Set seed for reproducibility
In [5]: SEED = 1

# Split data into 70% train and 30% test
In [6]: X_train, X_test, y_train, y_test = \
                                        train_test_split(X, y,
                                                         test_size=0.3,
                                                         stratify=y,
                                                         random_state=SEED)
```

Bagging Classifier in sklearn (Breast-Cancer dataset)

```
# Instantiate a classification-tree 'dt'
In [7]: dt = DecisionTreeClassifier(max_depth=4, min_samples_leaf=0.16,
                                    random_state=SEED)

# Instantiate a BaggingClassifier 'bc'
In [8]: bc = BaggingClassifier(base_estimator=dt, n_estimators=300,
                               n_jobs=-1)

# Fit 'bc' to the training set
In [9]: bc.fit(X_train, y_train)

# Predict test set labels
In [10]: y_pred = bc.predict(X_test)

# Evaluate and print test-set accuracy
In [11]: accuracy = accuracy_score(y_test, y_pred)

In [12]: print('Accuracy of Bagging Classifier: {:.3f}'.format(accuracy))

Out[12]: Accuracy of Bagging Classifier: 0.936
```

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Out Of Bag Evaluation

Elie Kawerk
Data Scientist

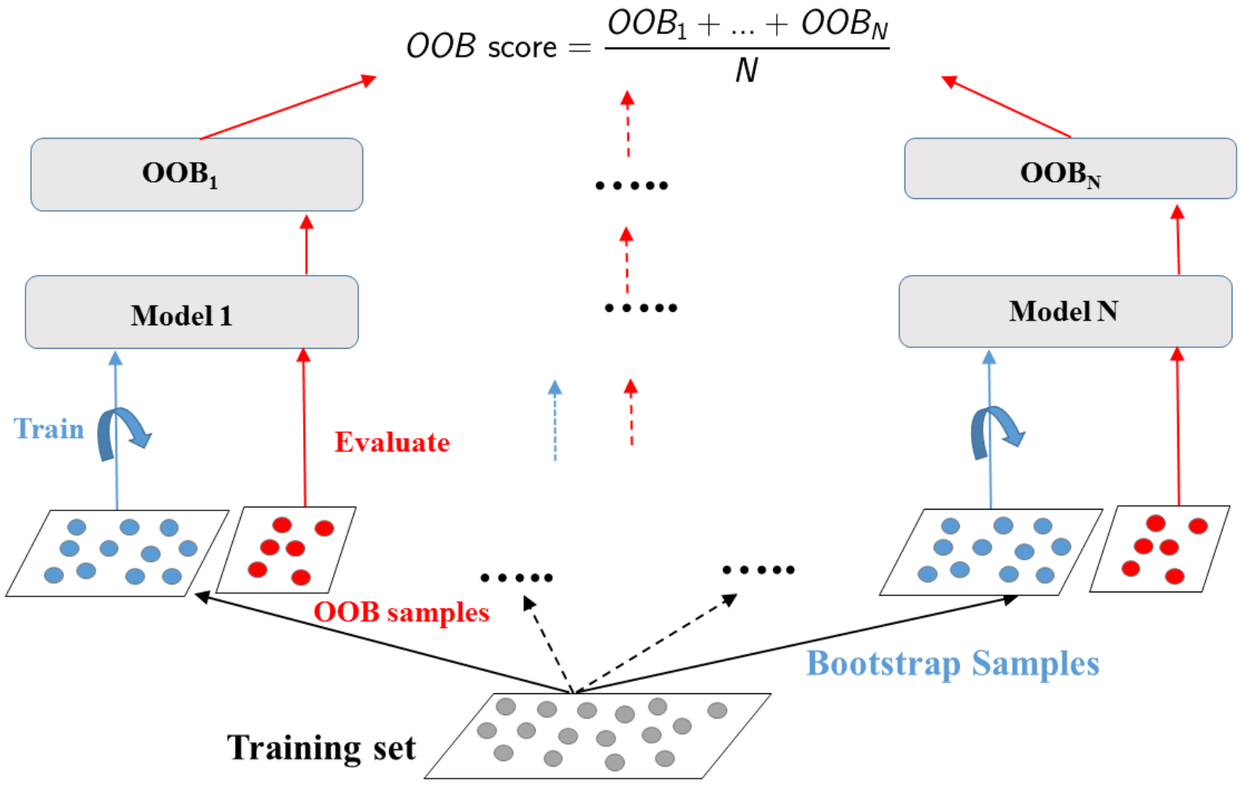
Bagging

- some instances may be sampled several times for one model,
- other instances may not be sampled at all.

Out Of Bag (OOB) instances

- On average, for each model, 63% of the training instances are sampled.
- The remaining 37% constitute the OOB instances.

OOB Evaluation



OOB Evaluation in sklearn (Breast Cancer Dataset)

```
# Import models and split utility function
In [1]: from sklearn.ensemble import BaggingClassifier
In [2]: from sklearn.tree import DecisionTreeClassifier
In [3]: from sklearn.metrics import accuracy_score
In [4]: from sklearn.model_selection import train_test_split

# Set seed for reproducibility
In [5]: SEED = 1

# Split data into 70% train and 30% test
In [6]: X_train, X_test, y_train, y_test =  \
                                    train_test_split(X, y,
                                                     test_size= 0.3,
                                                     stratify= y,
                                                     random_state=SEED)
```

OOB Evaluation in sklearn (Breast Cancer Dataset)

```
# Instantiate a classification-tree 'dt'
In [7]: dt = DecisionTreeClassifier(max_depth=4,
                                    min_samples_leaf=0.16,
                                    random_state=SEED)

# Instantiate a BaggingClassifier 'bc'; set oob_score= True
In [8]: bc = BaggingClassifier(base_estimator=dt, n_estimators=300,
                               oob_score=True, n_jobs=-1)

# Fit 'bc' to the traing set
In [9]: bc.fit(X_train, y_train)

# Predict the test set labels
In [10]: y_pred = bc.predict(X_test)
```

OOB Evaluation in sklearn (Breast Cancer Dataset)

```
# Evaluate test set accuracy
In [11]: test_accuracy = accuracy_score(y_test, y_pred)

# Extract the OOB accuracy from 'bc'
In [12]: oob_accuracy = bc.oob_score_

# Print test set accuracy
In [13]: print('Test set accuracy: {:.3f}'.format(test_accuracy))
Out[13]: Test set accuracy: 0.936

# Print OOB accuracy
In [14]: print('OOB accuracy: {:.3f}'.format(oob_accuracy))
Out[14]: OOB accuracy: 0.925
```

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Random Forests

Elie Kawerk
Data Scientist

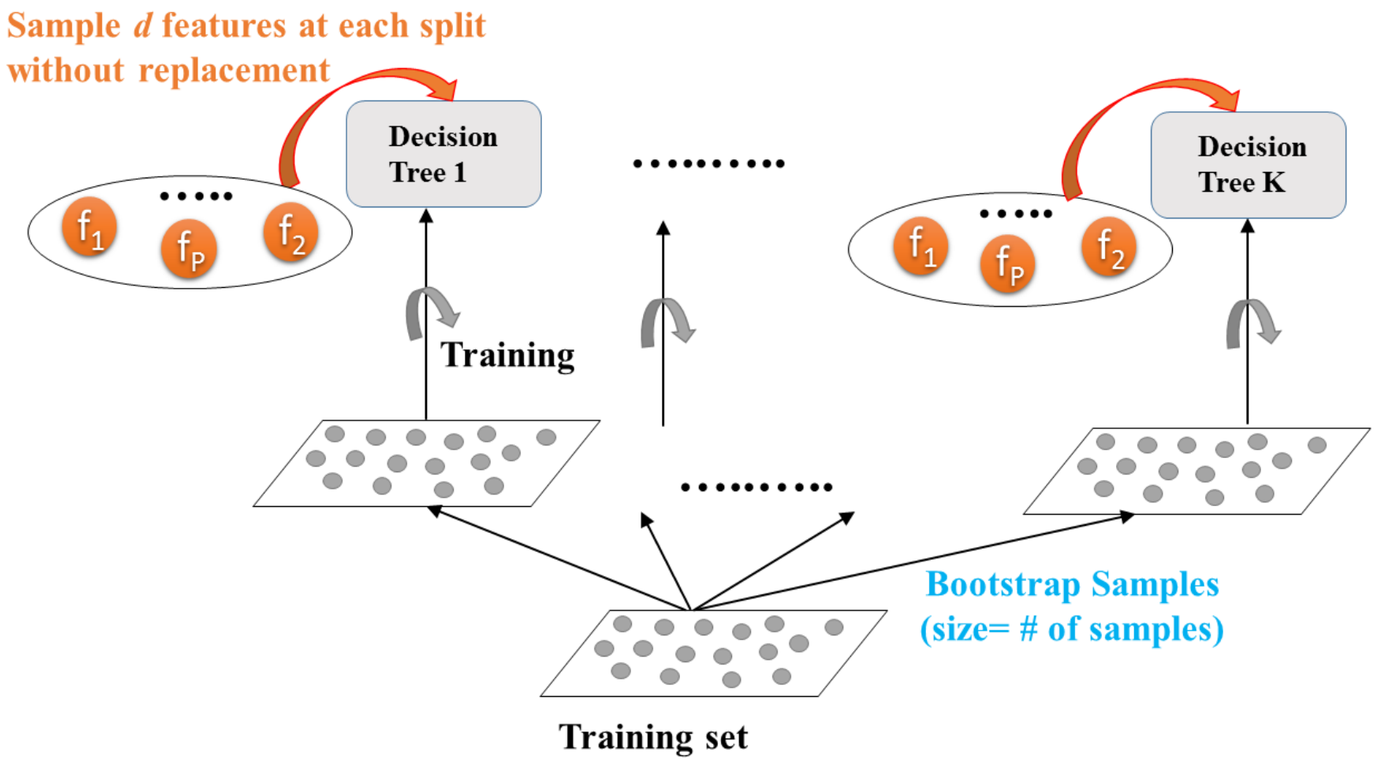
Bagging

- Base estimator: Decision Tree, Logistic Regression, Neural Net, ...
- Each estimator is trained on a distinct bootstrap sample of the training set
- Estimators use all features for training and prediction

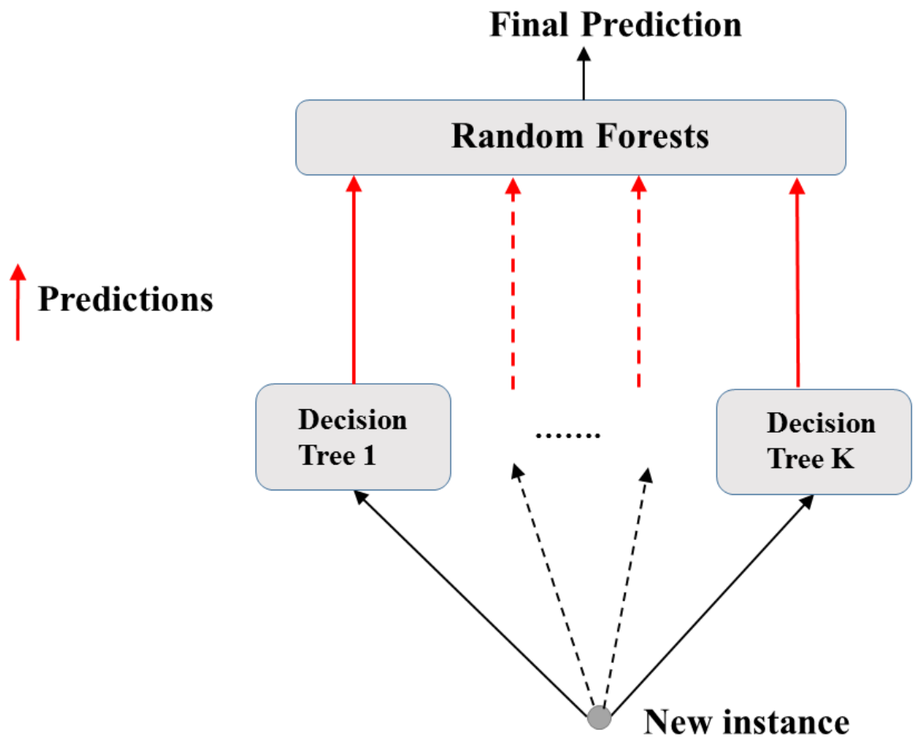
Further Diversity with Random Forests

- Base estimator: Decision Tree
- Each estimator is trained on a different bootstrap sample having the same size as the training set
- RF introduces further randomization in the training of individual trees
- d features are sampled at each node without replacement
  ($d < \text{total number of features}$)

Random Forests: Training



Random Forests: Prediction



Random Forests: Classification & Regression

Classification:

- Aggregates predictions by majority voting
- `RandomForestClassifier` in scikit-learn

Regression:

- Aggregates predictions through averaging
- `RandomForestRegressor` in scikit-learn

Random Forests Regressor in sklearn (auto dataset)

```
# Basic imports
In [1]: from sklearn.ensemble import RandomForestRegressor
In [2]: from sklearn.model_selection import train_test_split
In [3]: from sklearn.metrics import mean_squared_error as MSE

# Set seed for reproducibility
In [4]: SEED = 1

# Split dataset into 70% train and 30% test
In [5]: X_train, X_test, y_train, y_test = \
                                        train_test_split(X, y,
                                                         test_size=0.3,
                                                         random_state=SEED)
```

Random Forests Regressor in sklearn (auto dataset)

```
# Instantiate a random forests regressor 'rf' 400 estimators
In [6]: rf = RandomForestRegressor(n_estimators=400,
                                   min_samples_leaf=0.12,
                                   random_state=SEED)

# Fit 'rf' to the training set
In [7]: rf.fit(X_train, y_train)

# Predict the test set labels 'y_pred'
In [8]: y_pred = rf.predict(X_test)

# Evaluate the test set RMSE
In [9]: rmse_test = MSE(y_test, y_pred)**(1/2)

# Print the test set RMSE
In [10]: print('Test set RMSE of rf: {:.2f}'.format(rmse_test))

Out[10]: Test set RMSE of rf: 3.98
```

Feature Importance

Tree-based methods: enable measuring the importance of each feature in prediction.

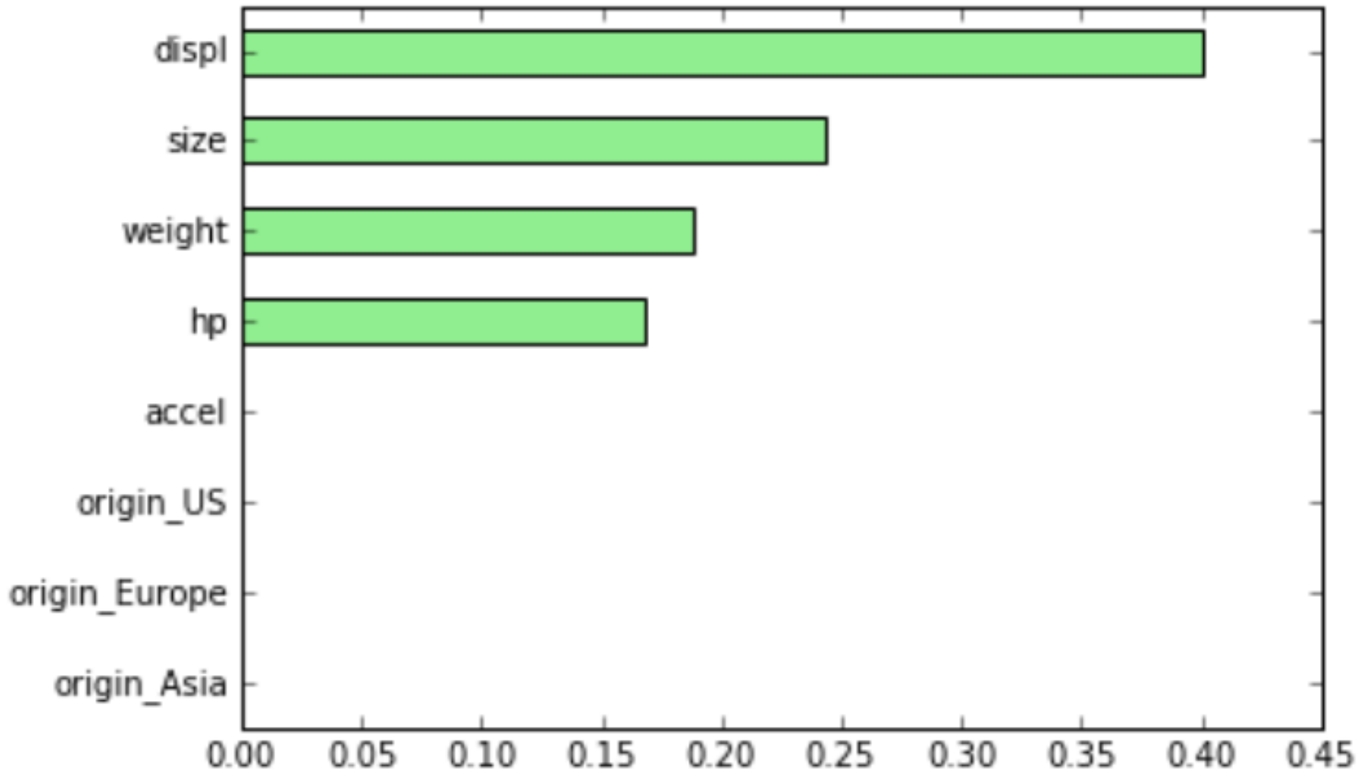
In `sklearn`:

- how much the tree nodes use a particular feature (weighted average) to reduce impurity
- accessed using the attribute `feature_importance_`

Feature Importance in sklearn

```
In [11]: import pandas as pd

In [12]: import matplotlib.pyplot as plt

# Create a pd.Series of features importances
In [13]: importances_rf = pd.Series(rf.feature_importances_,
                                    index = X.columns)

# Sort importances_rf
In [14]: sorted_importances_rf = importances_rf.sort_values()


# Make a horizontal bar plot
In [15]: sorted_importances_rf.plot(kind='barh',
                                    color='lightgreen'); plt.show()
```

Feature Importance in sklearn



MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Let's practice!