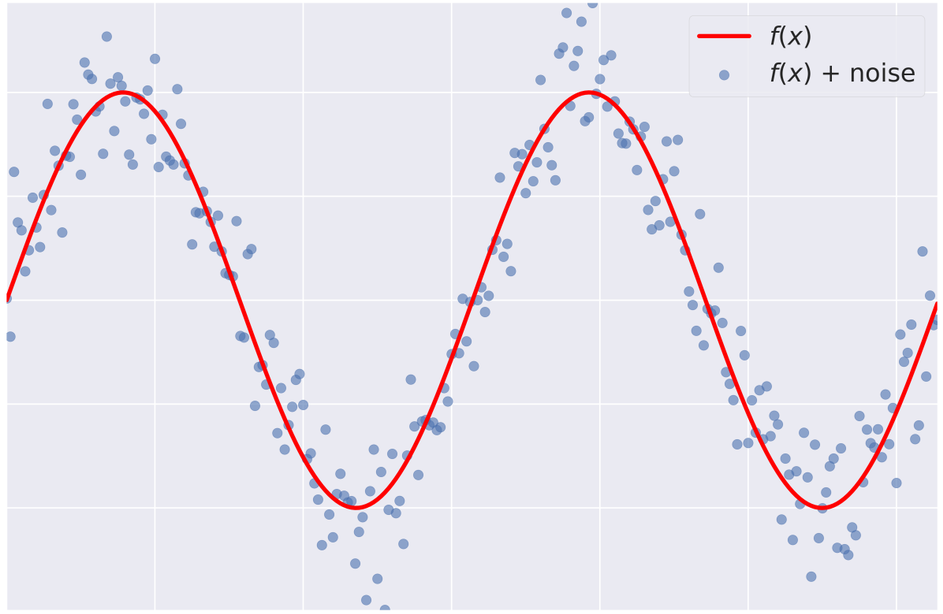
MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Generalization Error

Elie Kawerk
Data Scientist

Supervised Learning - Under the Hood

- Supervised Learning: $y = f(x)$, f is unknown.



Goals of Supervised Learning

- Find a model $\hat{f}$ that best approximates f: $\hat{f} \approx f$
- $\hat{f}$ can be Logistic Regression, Decision Tree, Neural Network ...
- Discard noise as much as possible.
- **End goal**: $\hat{f}$ should acheive a low predictive error on unseen datasets.

Difficulties in Approximating f

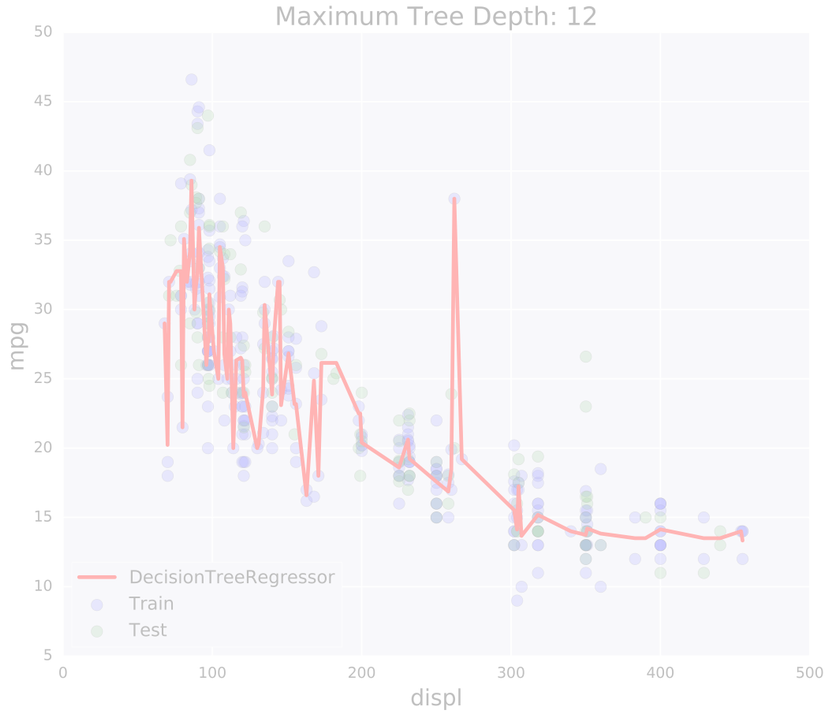
- **Overfitting**:

  $\hat{f}(x)$ fits the training set noise.

- **Underfitting**:

  $\hat{f}$ is not flexible enough to approximate f.

Overfitting



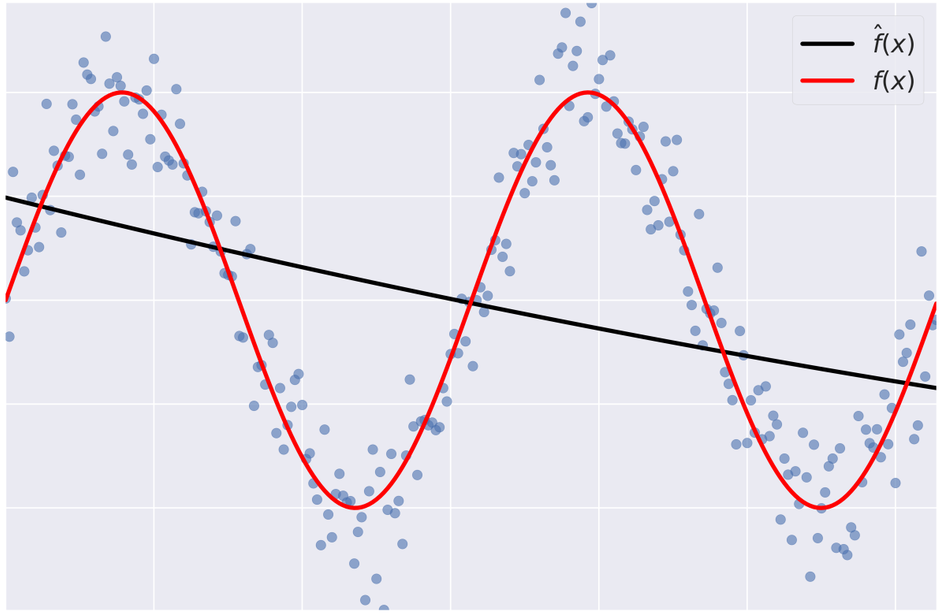
Generalization Error

- **Generalization Error of** $\hat{f}$: Does $\hat{f}$ generalize well on unseen data?
- It can be decomposed as follows:

  Generalization Error of $\hat{f} = bias^2 + variance + \text{irreducible error}$

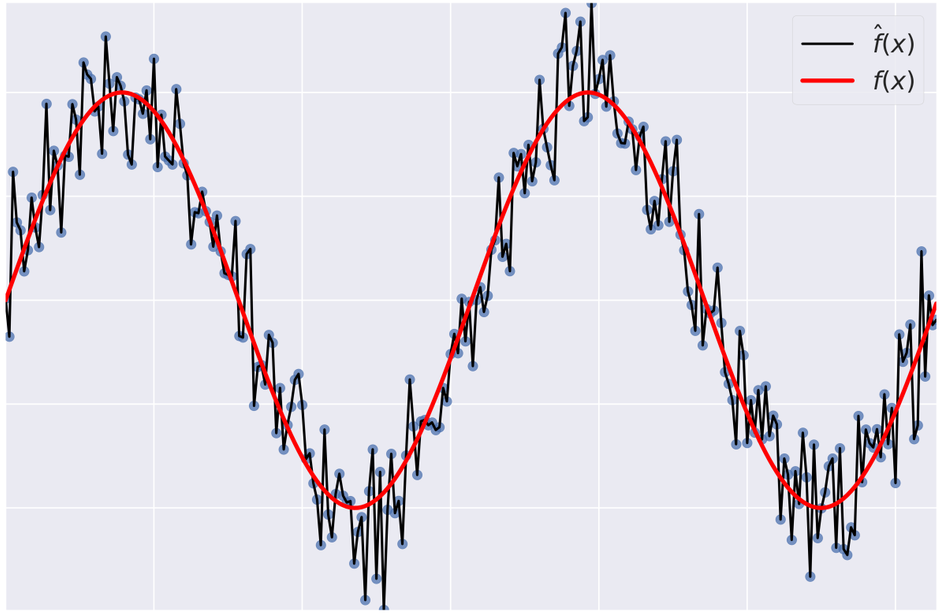
Bias

- **Bias**: error term that tells you, on average, how much $\hat{f} \neq f$.



Variance

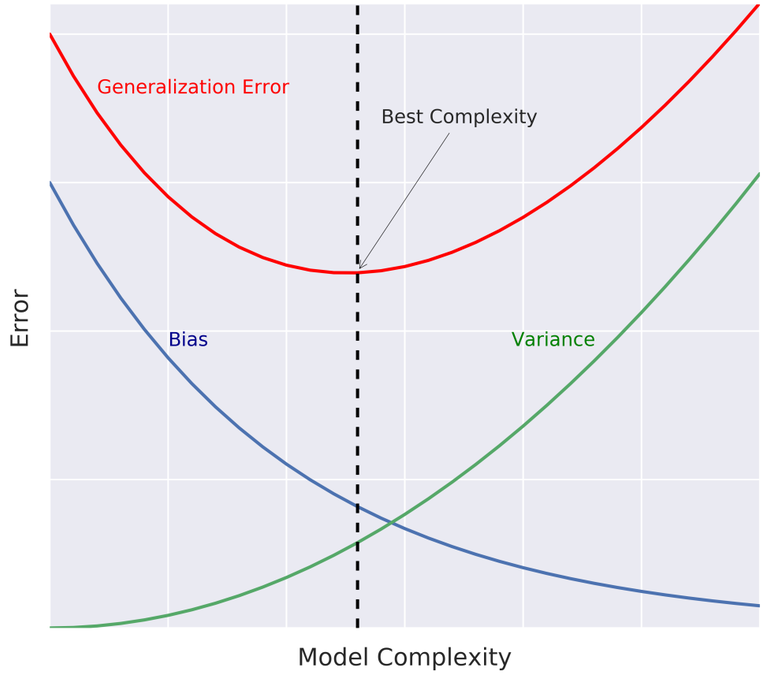
- **Variance**: tells you how much $\hat{f}$ is inconsistent over different training sets.



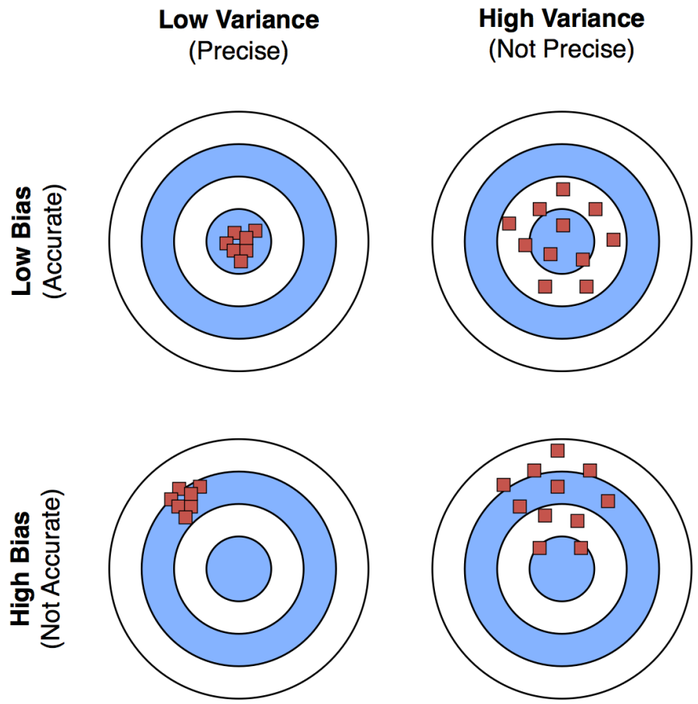
Model Complexity

- **Model Complexity**: sets the flexibility of $\hat{f}$.
- Example: Maximum tree depth, Minimum samples per leaf, ...

Bias-Variance Tradeoff



Bias-Variance Tradeoff: A Visual Explanation



MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Diagnosing Bias and Variance Problems

Elie Kawerk
Data Scientist

Estimating the Generalization Error

- How do we estimate the generalization error of a model?
- Cannot be done directly because:
  - f is unknown,
  - usually you only have one dataset,
  - noise is unpredictable.

Estimating the Generalization Error

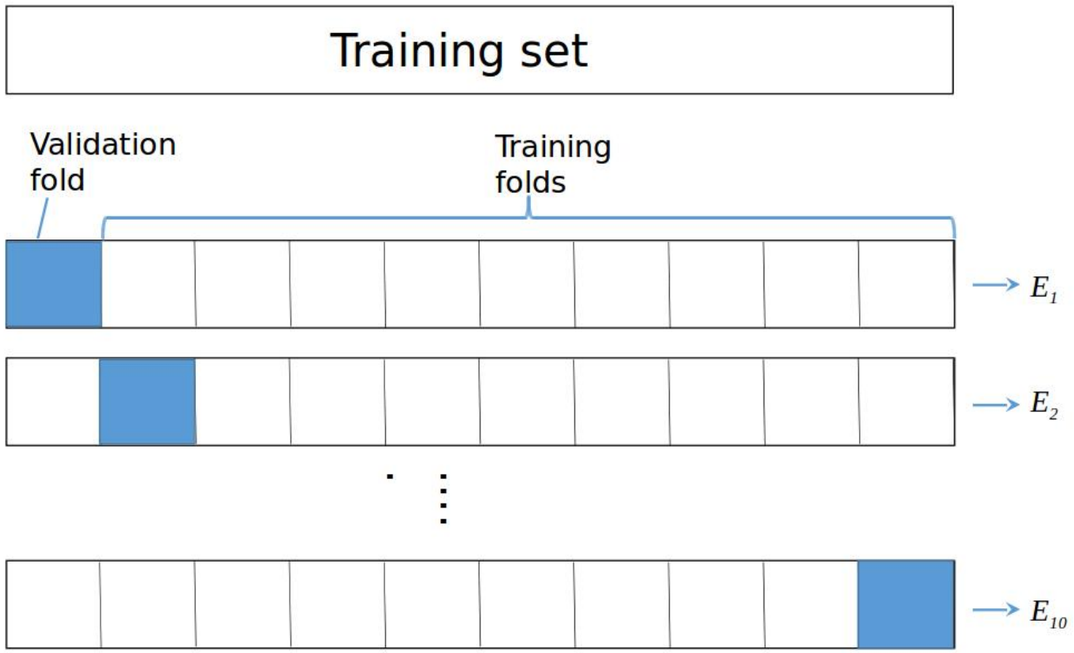
Solution:

- split the data to training and test sets,
- fit $\hat{f}$ to the training set,
- evaluate the error of $\hat{f}$ on the **unseen** test set.
- generalization error of $\hat{f} \approx$ test set error of $\hat{f}$.

Better Model Evaluation with Cross-Validation

- Test set should not be touched until we are confident about $\hat{f}$'s performance.
- Evaluating $\hat{f}$ on training set: biased estimate, $\hat{f}$ has already seen all training points.
- Solution $\rightarrow$ Cross-Validation (CV):
  - K-Fold CV,
  - Hold-Out CV.

K-Fold CV



K-Fold CV

$$\text{CV error} = \frac{E_1 + \ldots + E_{10}}{10}$$

Diagnose Variance Problems

- If $\hat{f}$ suffers from **high variance**:

  CV error of $\hat{f}$ > training set error of $\hat{f}$.

- $\hat{f}$ is said to overfit the training set. To remedy overfitting:
  - decrease model complexity,
  - for ex: decrease max depth, increase min samples per leaf, ...
  - gather more data, ..

Diagnose Bias Problems

- if $\hat{f}$ suffers from high bias:

  CV error of $\hat{f} \approx$ training set error of $\hat{f} >>$ desired error.

- $\hat{f}$ is said to underfit the training set. To remedy underfitting:

  - increase model complexity
  - for ex: increase max depth, decrease min samples per leaf, ...
  - gather more relevant features

K-Fold CV in sklearn on the Auto Dataset

```
In [1]: from sklearn.tree import DecisionTreeRegressor
In [2]: from sklearn.model_selection import train_test_split
In [3]: from sklearn.metrics import mean_squared_error as MSE
In [4]: from sklearn.model_selection import cross_val_score

# Set seed for reproducibility
In [5]: SEED = 123

# 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,
                                                         random_state=SEED)

# Instantiate decision tree regressor and assign it to 'dt'
In [7]: dt = DecisionTreeRegressor(max_depth=4,
                                   min_samples_leaf=0.14,
                                   random_state=SEED)
```

K-Fold CV in sklearn on the Auto Dataset

```
# Evaluate the list of MSE ontained by 10-fold CV
# Set n_jobs to -1 in order to exploit all CPU cores in computation
In [8]: MSE_CV = - cross_val_score(dt, X_train, y_train, cv= 10,
                                   scoring='neg_mean_squared_error',
                                   n_jobs = -1)

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

# Predict the labels of training set
In [10]: y_predict_train = dt.predict(X_train)

# Predict the labels of test set
In [11]: y_predict_test = dt.predict(X_test)
```

K-Fold CV in sklearn on the Auto Dataset

```
# CV MSE
In [12]: print('CV MSE: {:.2f}'.format(MSE_CV.mean()))
Out[12] CV MSE: 20.51

# Training set MSE
In [13]: print('Train MSE: {:.2f}'.format(MSE(y_train, y_predict_train)))
Out[13]: Train MSE: 15.30

# Test set MSE
In [14]: print('Test MSE: {:.2f}'.format(MSE(y_test, y_predict_test)))
Out[14]: Test MSE: 20.92
```

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Ensemble Learning

Elie Kawerk
Data Scientist

Advantages of CARTs

- Simple to understand.
- Simple to interpret.
- Easy to use.
- Flexibility: ability to describe non-linear dependencies.
- Preprocessing: no need to standardize or normalize features, ...

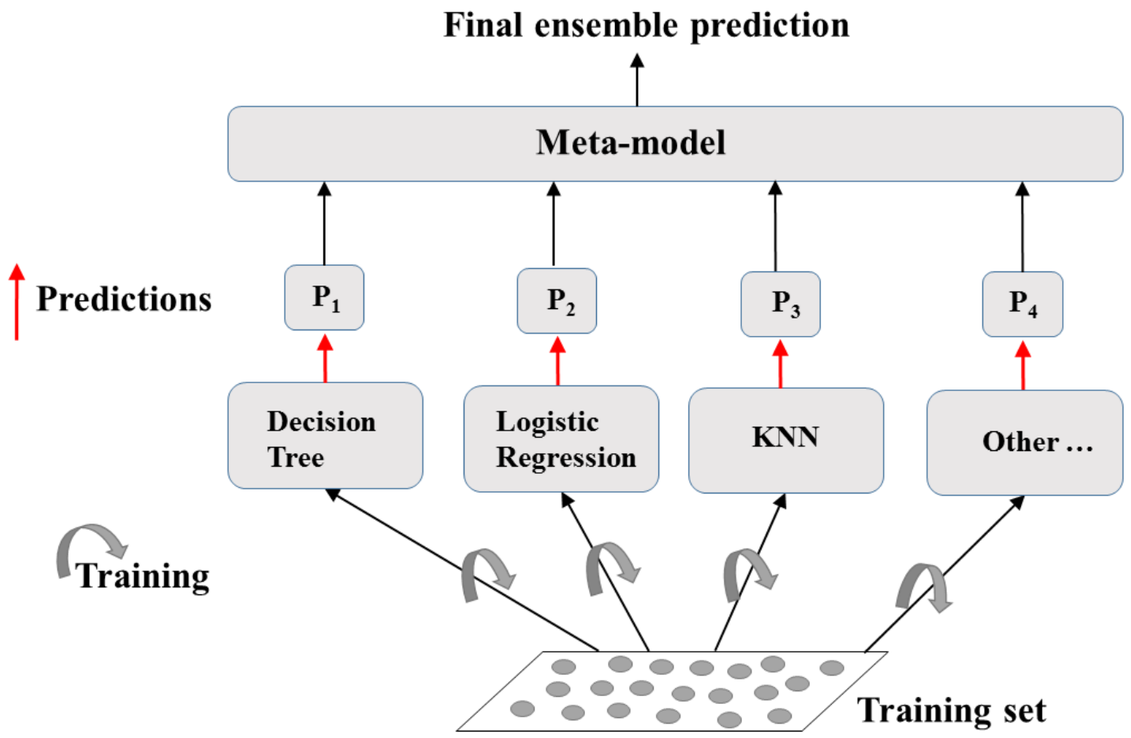
Limitations of CARTs

- Classification: can only produce orthogonal decision boundaries.
- Sensitive to small variations in the training set.
- High variance: unconstrained CARTs may overfit the training set.
- Solution: ensemble learning.

Ensemble Learning

- Train different models on the same dataset.
- Let each model make its predictions.
- Meta-model: aggregates predictions of individual models.
- Final prediction: more robust and less prone to errors.
- Best results: models are skillful in different ways.

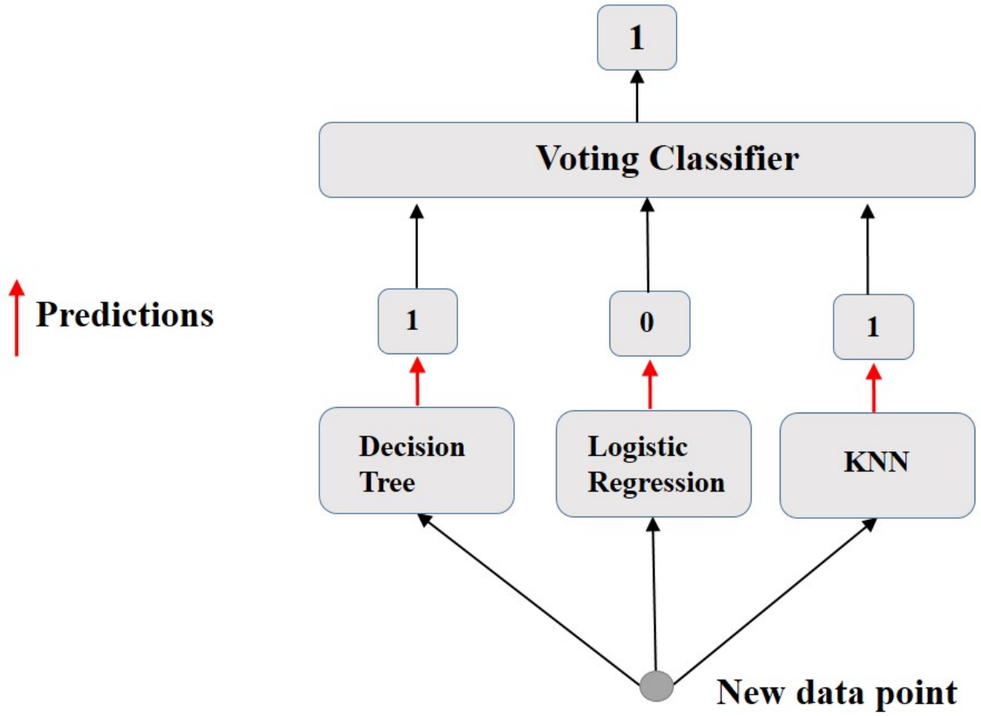
Ensemble Learning: A Visual Explanation



Ensemble Learning in Practice: Voting Classifier

- Binary classification task.
- N classifiers make predictions: P_1, P_2, ..., P_N with P_i = 0 or 1.
- Meta-model prediction: hard voting.

Hard Voting


1
Voting Classifier
Predictions
1
0
1
Decision Tree
Logistic Regression
KNN
New data point

Voting Classifier in sklearn (Breast-Cancer dataset)

```
# Import function to compute accuracy
In [1]: from sklearn.metrics import accuracy_score

# Import function to split data
In [2]: from sklearn.model_selection import train_test_split

# Import models
In [3]: from sklearn.linear_model import LogisticRegression
In [4]: from sklearn.tree import DecisionTreeClassifier
In [5]: from sklearn.neighbors import KNeighborsClassifier as KNN

# Import the VotingClassifier meta-model
In [6]: from sklearn.ensemble import VotingClassifier

# Set seed for reproducibility
In [7]: SEED = 1
```

Voting Classifier in sklearn (Breast-Cancer dataset)

```
# Split data into 70% train and 30% test
In [8]: X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                          test_size= 0.3,
                                                          random_state= SEED)
# Instantiate individual classifiers
In [9]: lr = LogisticRegression(random_state=SEED)

In [10]: knn = KNN()

In [11]: dt = DecisionTreeClassifier(random_state=SEED)

# Define a list called classifier that contains
# the tuples (classifier_name, classifier)
In [12]: classifiers = [('Logistic Regression', lr),
                        ('K Nearest Neighbours', knn),
                        ('Classification Tree', dt)]
```

Voting Classifier in sklearn (Breast-Cancer dataset)

```
# Iterate over the defined list of tuples containing the classifiers
In [13]: for clf_name, clf in classifiers:

                #fit clf to the training set
                clf.fit(X_train, y_train)

                # Predict the labels of the test set
                y_pred = clf.predict(X_test)

                # Evaluate the accuracy of clf on the test set
                print('{:s} : {:.3f}'.format(clf_name,
                                             accuracy_score(y_test, y_pred)))

Out[13]: Logistic Regression: 0.947
         K Nearest Neighbours: 0.930
         Classification Tree: 0.930
```

Voting Classifier in sklearn (Breast-Cancer dataset)

```
# Instantiate a VotingClassifier 'vc'
In [14]: vc = VotingClassifier(estimators=classifiers)

# Fit 'vc' to the traing set
In [15]: vc.fit(X_train, y_train)

# Predict test set labels
In [16]: y_pred = vc.predict(X_test)

# Evaluate the test-set accuracy of 'vc'
In [17]: print('Voting Classifier: {.3f}'.format(
                                        accuracy_score(y_test, y_pred)))

Out[17]: Voting Classifier: 0.953
```

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Let's practice!