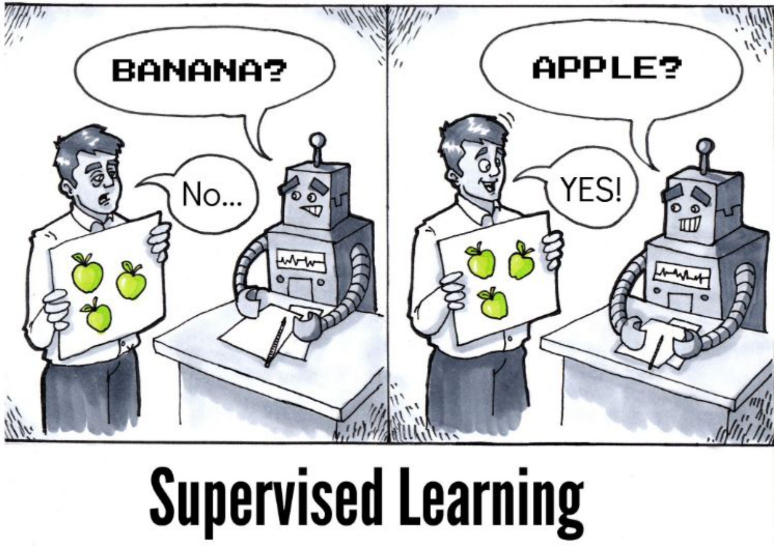
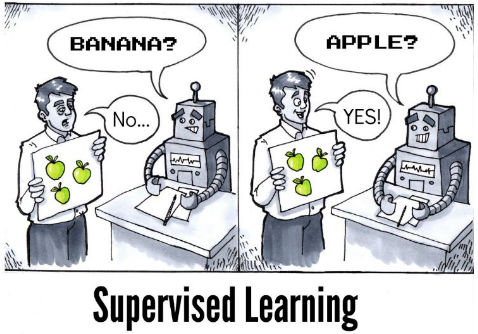
Supervised Learning



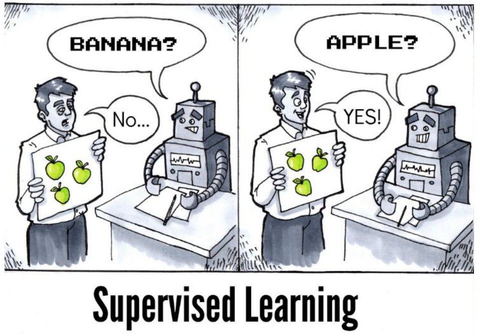
Supervised Learning

DATA

TRAIN



TEST

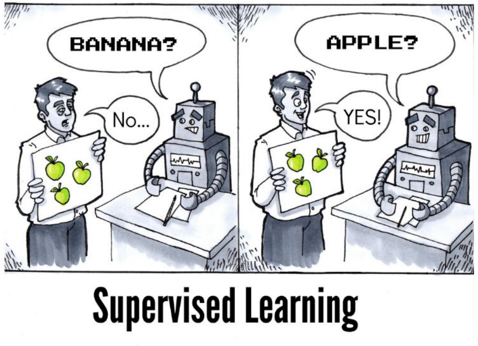


Supervised Learning

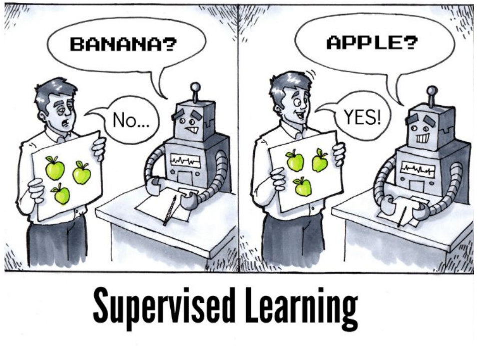
DATA

TRAIN

TEST



Used to train the model

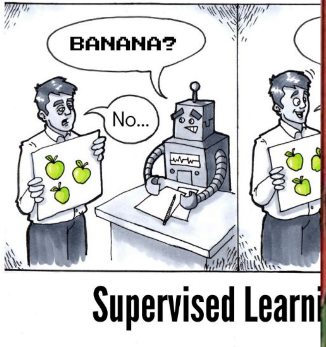


Used to test the model

Supervised Learning

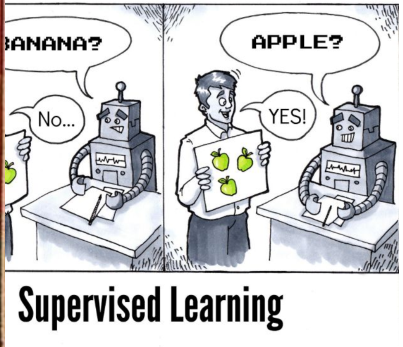
DATA

TRAIN



TRAIN AND TEST

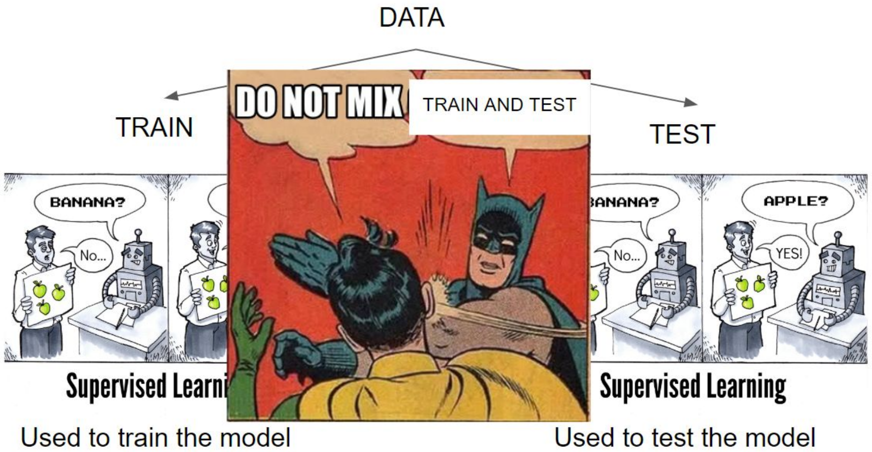
TEST

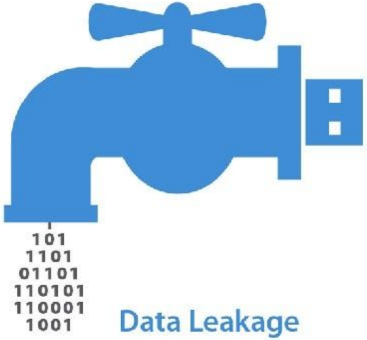


Used to train the model

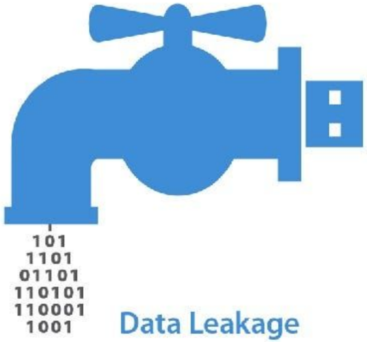
Used to test the model

Supervised Learning



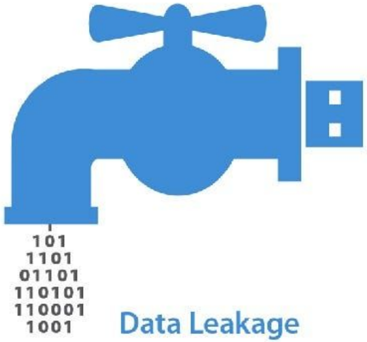


Supervised Learning



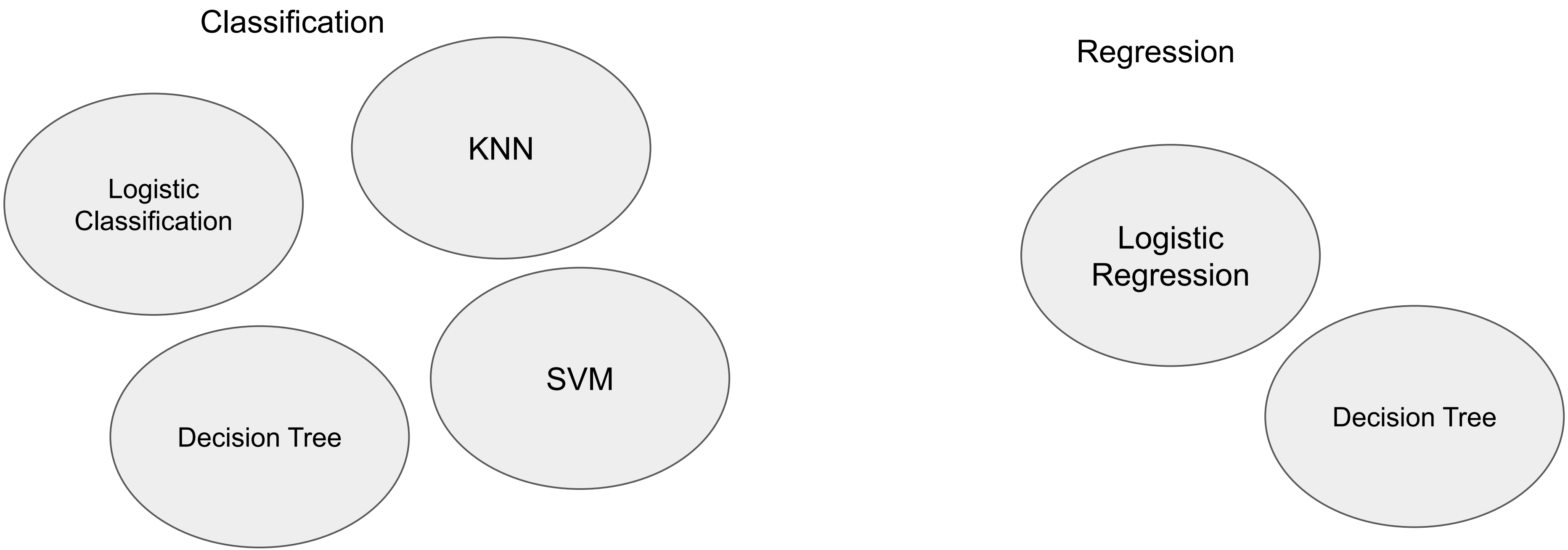
- When information outside the training dataset is used to create the model.
- When training data leaks into test data
- When test data leaks into training data

Supervised Learning



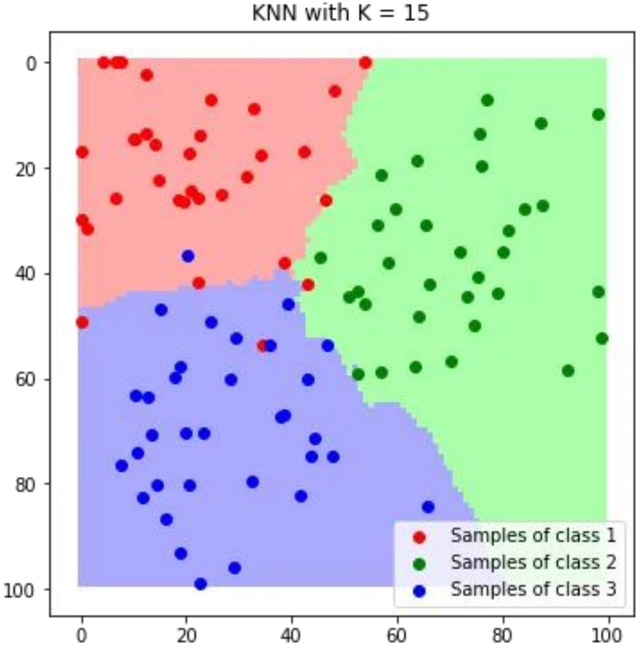
- Thread carefully when applying transformations on the entire dataset e.g,
  - Handle missing values per dataset split (mean, std, normalizations,...)
  - Feature extractions

Supervised Learning



Supervised Learning

K-Nearest Neighbors (KNN)

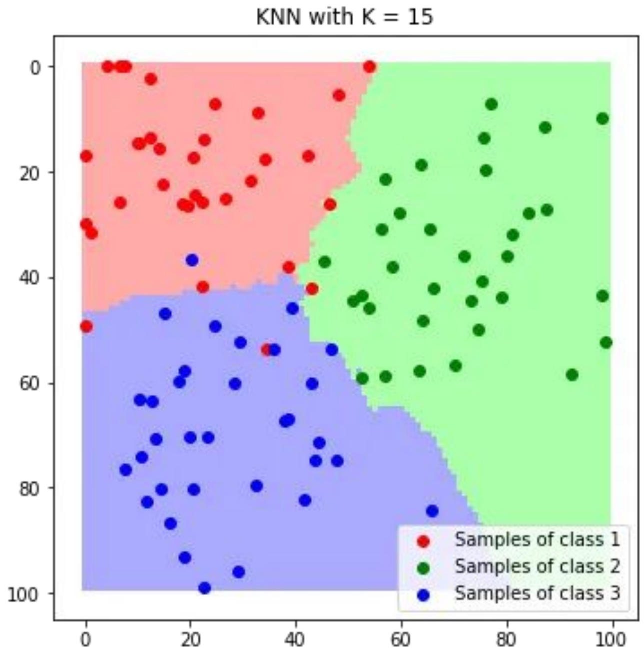


Supervised Learning

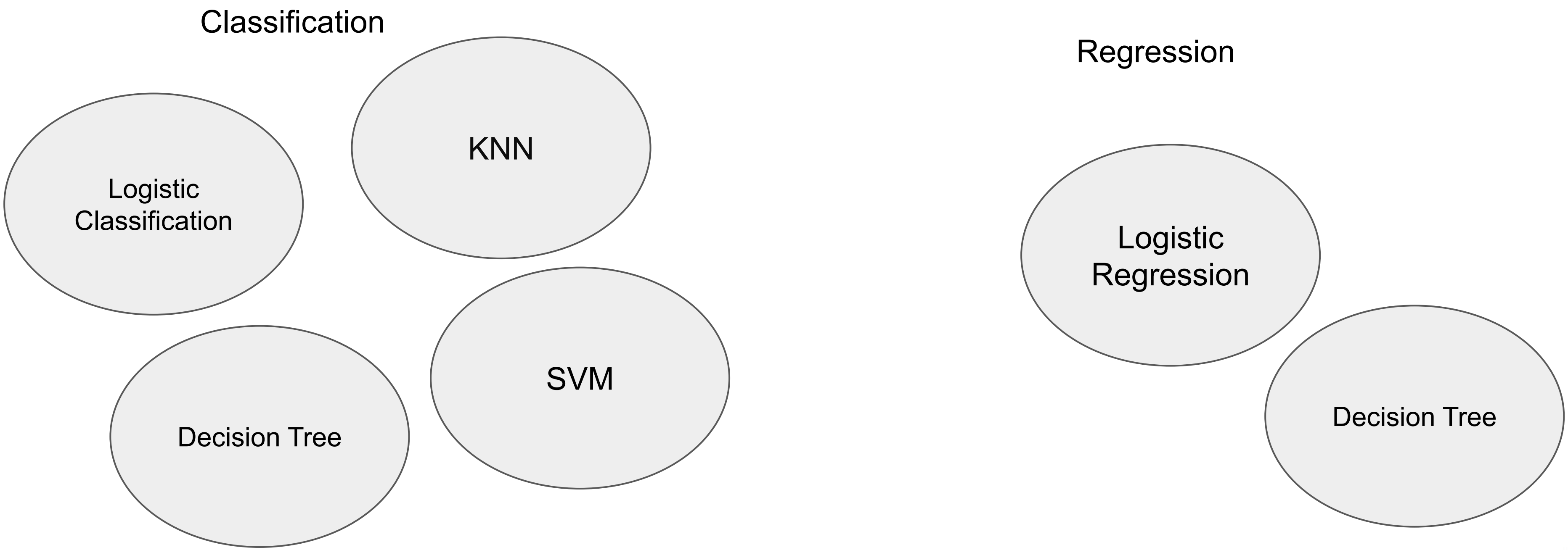
K-Nearest Neighbors (KNN)
(https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)

Training:
- It’s not “trained”
- Load training data

Evaluate:
- Select K
- New data point
- K closest belong to which class ?
- Select that class



Supervised Learning



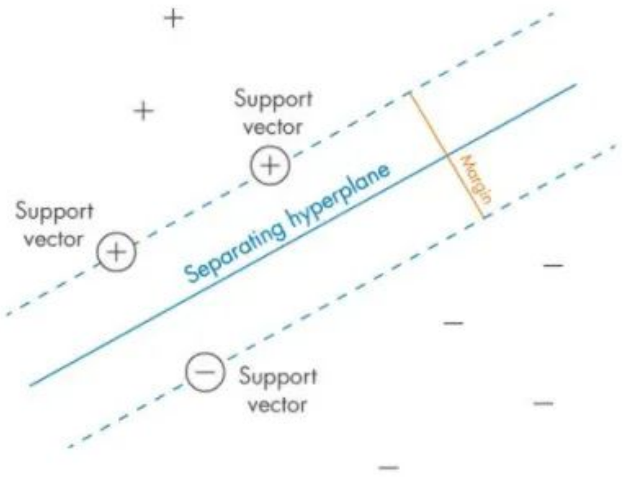
Supervised Learning

Support Vector Machine
(sklearn.svm.SVC)

Support Vectors − Data points that are closest to the hyperplane are called support vectors. A separating line will be defined with the help of these data points.

Hyperplane − It is a decision plane or space which is divided between a set of objects having different classes.

Margin − Gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. A large margin is considered as a good margin and a small margin is considered as a bad margin.

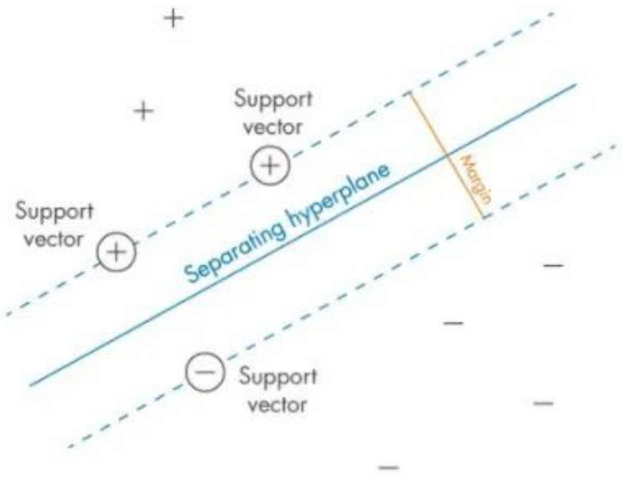


Supervised Learning

Support Vector Machine
(sklearn.svm.SVC)

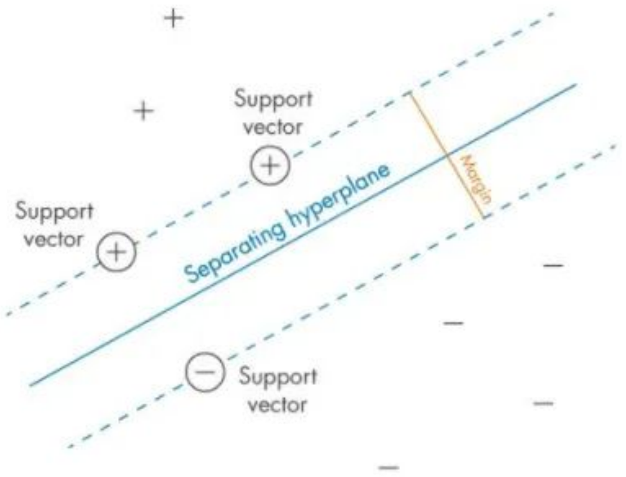
Training

- Generate hyperplanes iteratively
- Find hyperplane that separates the classes as best as possible
    - Transforms data using the kernel trick (i.e adds more dimensions to the data)
    - Tries to separate the data



Supervised Learning

Support Vector Machine
(sklearn.svm.SVC)



Pros

- SVM works relatively well when there is a clear margin of separation between classes.
- SVM is more effective in high dimensional spaces.

Cons

- SVM algorithm is not suitable for large data sets.
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.

Supervised Learning

Classification

KNN

Logistic Classification

SVM

Decision Tree

Regression

Logistic Regression

Decision Tree

Supervised Learning

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.DecisionTreeClassifier`(**, criterion='gini', splitter='best', max_depth=None, min_sam*
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes
min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini i
and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to c
and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are p
contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_sampl`
number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any dep
considered if it leaves at least `min_samples_leaf` training samples in each of the left and
may have the effect of smoothing the model, especially in regression.

sklearn.svm.SVC

class `sklearn.svm.SVC`(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `LinearSVC` or `SGDClassifier` instead, possibly after a `Nystroem` transformer or other Kernel Approximation.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how `gamma`, `coef0` and `degree` affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Parameters:

C : *float, default=1.0*
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*
Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`. For an intuitive visualization of different kernel types see Plot classification boundaries with different SVM Kernels.

degree : *int, default=3*
Degree of the polynomial kernel function ('poly'). Must be non-negative. Ignored by all other kernels.

gamma : *{'scale', 'auto'} or float, default='scale'*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Supervised Learning

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassi**
min_samples_leaf=1, min_weight_fraction_lea
min_impurity_decrease=0.0, class_weight=No

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entro*
The function to measur
and "entropy" both for

splitter : *{"best", "rando*
The strategy used to ch
and "random" to choos

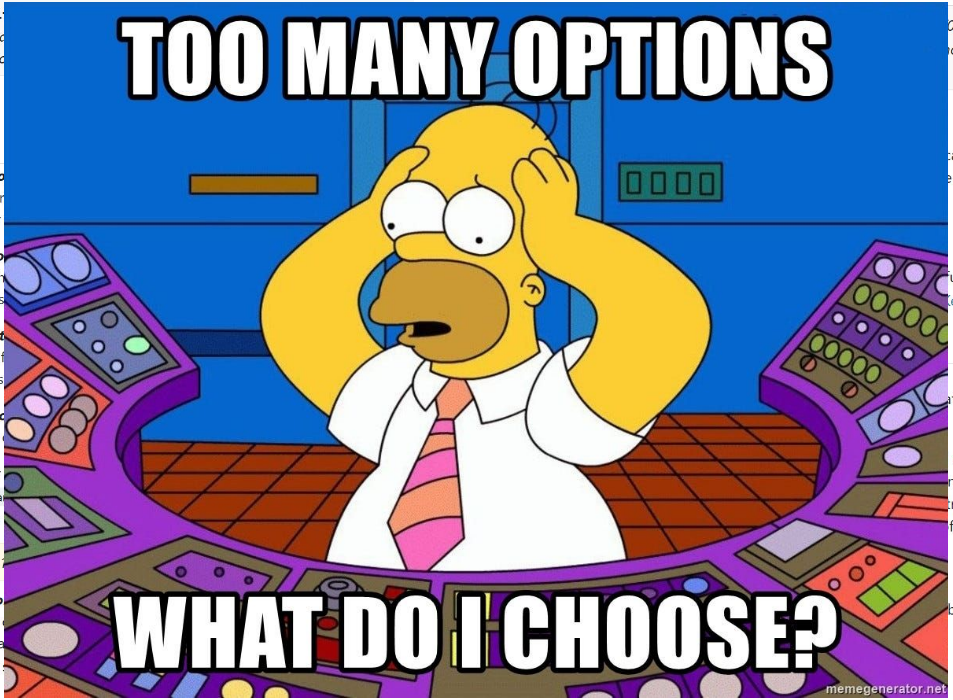
max_depth : *int, default*
The maximum depth of
contain less than min_s

min_samples_split : *int c*
The minimum number

- If int, then consider
- If float, then `min_sa`
  number of samples

Changed in version 0.1

min_samples_leaf : *int o*
The minimum number
considered if it leaves a
may have the effect of



sklearn.svm.SVC

0=0.0, shrinking=True, probability=False, tol=0.001,
ction_shape='ovr', break_ties=False,
[source]

ally with the number of samples and may be
r using `LinearSVC` or `SGDClassifier` instead, possibly

unctions and how `gamma`, `coef0` and `degree` affect
ernel functions.

ation is inversely proportional to C. Must be strictly

or callable, default='rbf'
none is given, 'rbf' will be used. If a callable is given it
rices; that matrix should be an array of shape
f different kernel types see Plot classification

be non-negative. Ignored by all other kernels.

Supervised Learning

Task

Supervised Learning

Task

- Classification
- Regression

Supervised Learning

Task → Classification → Models

Task → Regression → Models

Supervised Learning

Task → Classification → Models → Hyperparameters

Task → Regression → Models → Hyperparameters

Supervised Learning

Parameters vs Hyperparameters

Supervised Learning

Parameters vs Hyperparameters

Are learned by the model, i.e they change during training

Supervised Learning

Parameters vs Hyperparameters

Parameters

Are learned by the model, i.e they change during training

Hyperparameters

Are decided before and fixed during training, i.e they are not learnt by the model and impact learning.

Supervised Learning

Hyperparameter Search
Find the best hyperparameters for the target task, model, and existing data.

Supervised Learning

Hyperparameter Search
Find the best hyperparameters for the target task, model and existing data.

Search Space
 - given a model, all hyperparameters combinations possible

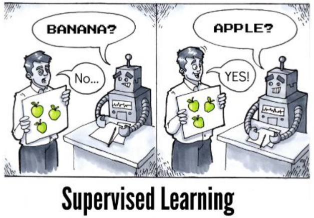
Supervised Learning

Hyperparameter Search

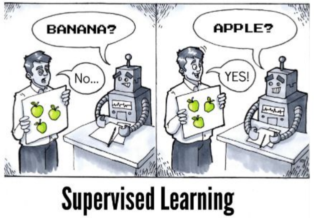
Find the best hyperparameters for the target task, model and existing data.

DATA

TRAIN



TEST



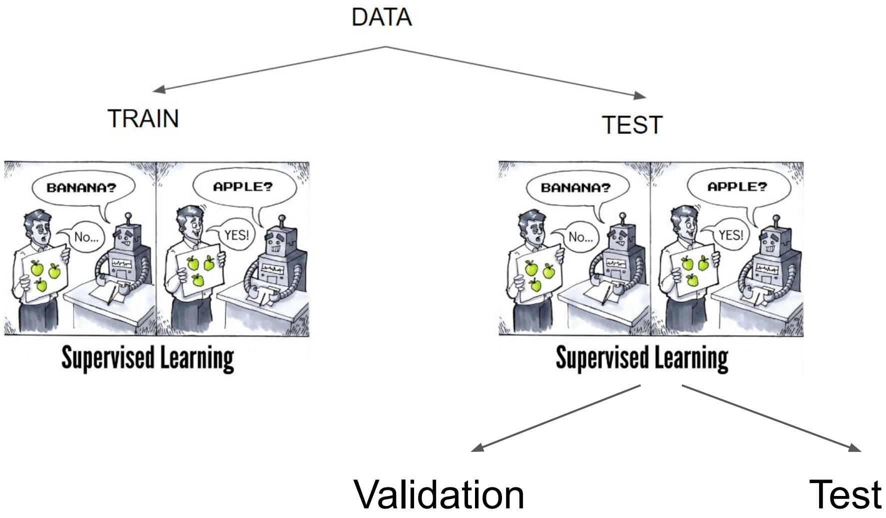
Validation

Test

Supervised Learning

Hyperparameter search consists of:

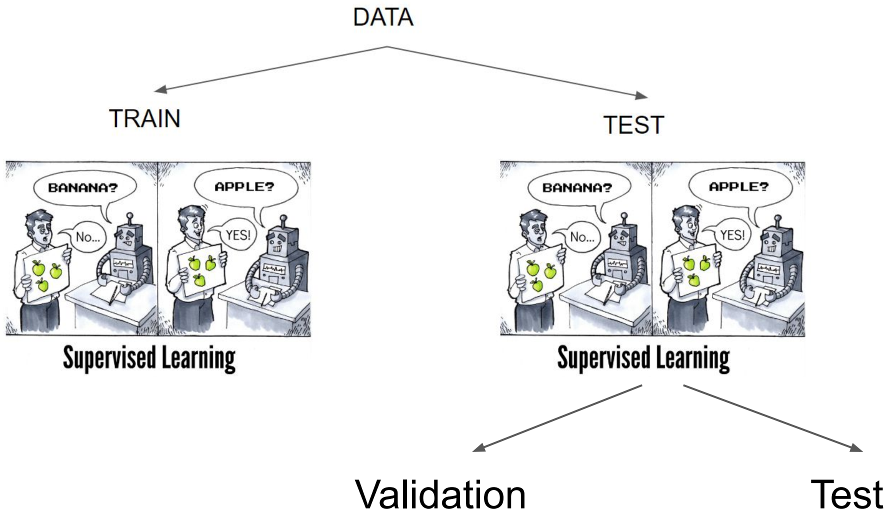
- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- a score function.



Supervised Learning

Hyperparameter search consists of:

- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- ***a method for searching or sampling candidates;***
- a cross-validation scheme
- a score function.



Supervised Learning

Hyperparameter search

- Grid Search
- Random Search
- Bayesian Search

Supervised Learning

Hyperparameter search

- Grid Search
  (systematically evaluates a predefined set of hyperparameters combinations to find the best set of hyperparameter for a given model)
  (sklearn.model_selection.GridSearchCV)

Supervised Learning

Hyperparameter search

- Grid Search (sklearn.model_selection.GridSearchCV)

Supervised Learning

Hyperparameter search

- Grid Search
  (sklearn.model_selection.GridSearchCV)
- Random Search
  (randomly *samples* hyperparameter values from predefined distributions to efficiently explore the search space and find optimal hyperparameter configurations for machine learning models)
  (sklearn.model_selection.RandomizedSearchCV)

Supervised Learning

Hyperparameter search

- Grid Search (sklearn.model_selection.GridSearchCV)
- Random Search (sklearn.model_selection.RandomizedSearchCV)
- Bayesian Search

Supervised Learning

Hyperparameter search

- Grid Search
(sklearn.model_selection.GridSearchCV)
- Random Search
(sklearn.model_selection.RandomizedSearchCV)
- Bayesian Search
(is a probabilistic optimization technique that uses surrogate models to efficiently explore and optimize complex hyperparameter spaces in machine learning by iteratively selecting hyperparameters to evaluate based on a balance of exploration and exploitation)
(skopt.BayesSearchCV)

Supervised Learning

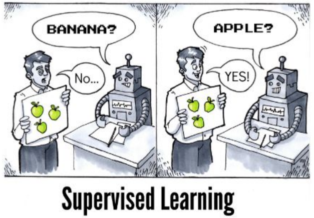
Hyperparameter search consists of:

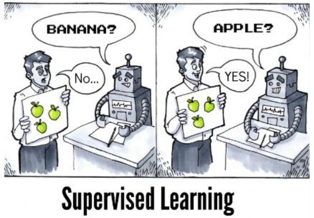
- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- a method for searching or sampling candidates;
- **a cross-validation scheme**
- a score function.

DATA

TRAIN

TEST





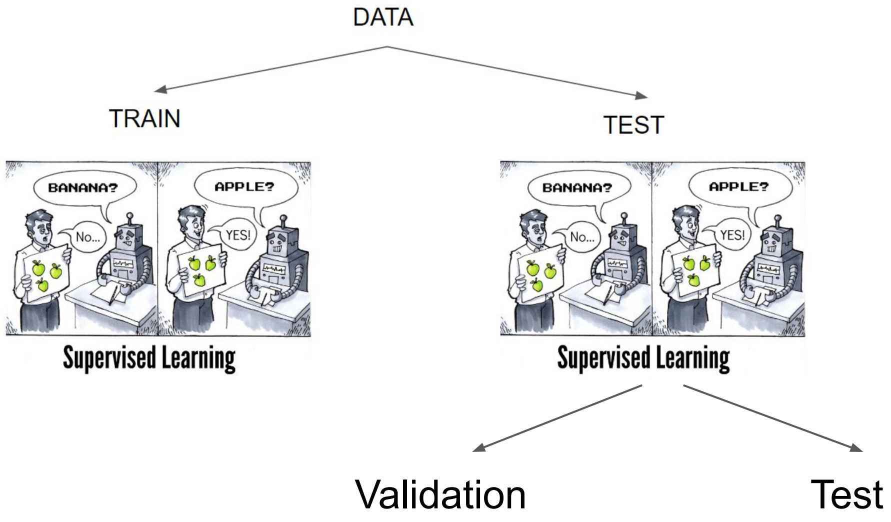
Validation

Test

Supervised Learning

Dataset splits

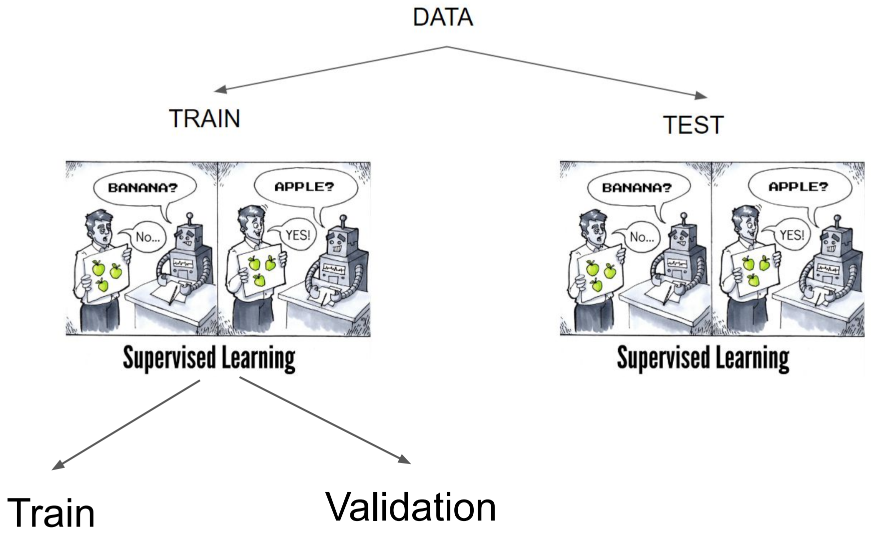
- **Train** is used to train the selected model with the selected hyperparameters
- **Validation** is used to validate the trained models
- **Test** is used to test the validated model with the higher selected metric



Supervised Learning

Dataset splits

- **Train** is used to train the selected model with the selected hyperparameters
- **Validation** is used to validate the trained models
- **Test** is used to test the validated model with the higher selected metric



Supervised Learning

Hyperparameter search

Cross Validation Schemes

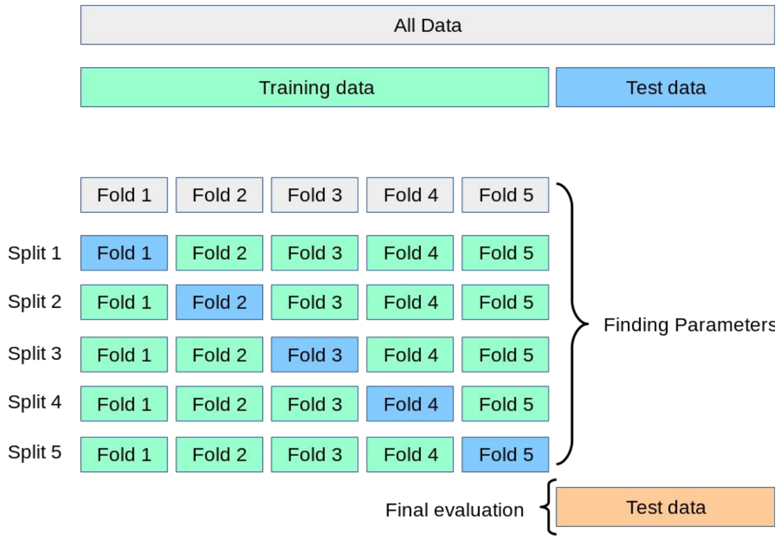
- K-Fold Cross Validation
- Leave-One-Out Cross Validation
- Stratified Cross Validation
- Time-Series Cross Validation
- Repeated Cross-Validation

Supervised Learning

Hyperparameter search

Cross Validation Schemes

- K-Fold Cross Validation



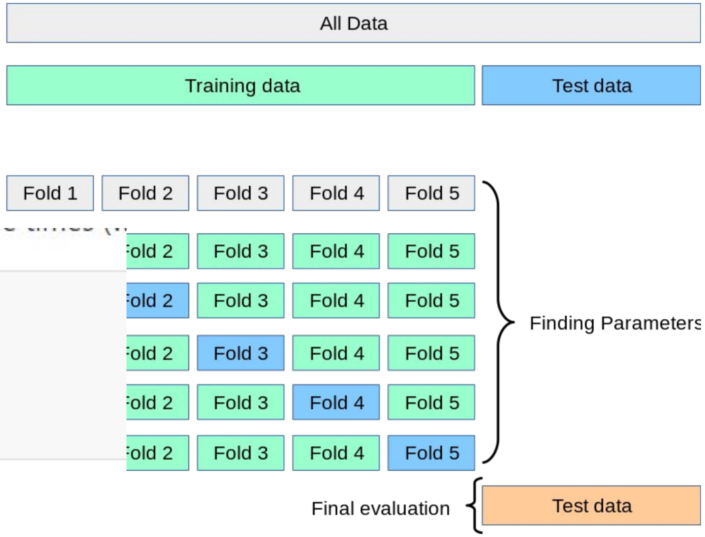
Supervised Learning

Hyperparameter search

Cross Validation Schemes

- K-Fold Cross Validation (sklearn.model_selection.cross_val_score)

```
>>> from sklearn.model_selection import cross_val_score
>>> clf = svm.SVC(kernel='linear', C=1, random_state=42)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores
array([0.96..., 1. , 0.96..., 0.96..., 1. ])
```



Supervised Learning

Hyperparameter search

Cross Validation Schemes

- Leave-One-Out Cross Validation

Leave-One-Out

Supervised Learning

Hyperparameter search

Cross Validation Schemes

- Leave-One-Out Cross Validation (sklearn.model_selection.LeaveOneOut)
  I.e leave one out and repeat process N times, where N is the time number of observations

Leave-One-Out

Supervised Learning

Hyperparameter search

Cross Validation Schemes

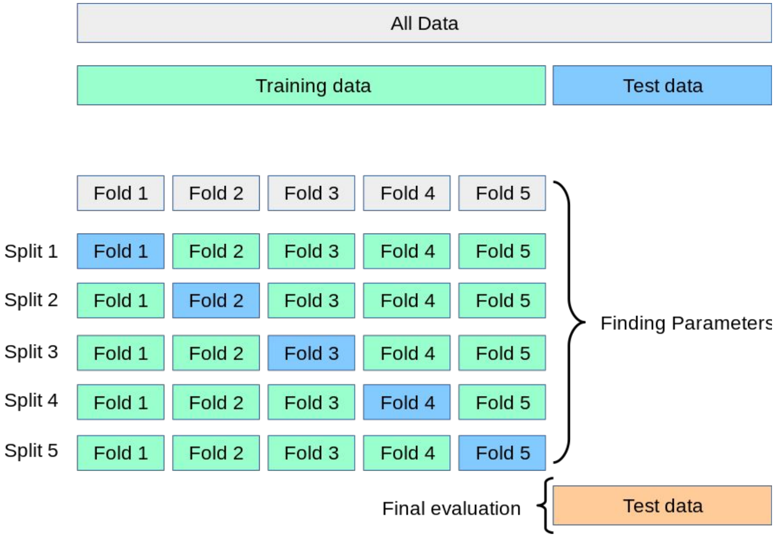
- Stratified Cross Validation

Supervised Learning

Hyperparameter search

Cross Validation Schemes

- Stratified Cross Validation
  (sklearn.model_selection.StratifiedKFold)
  is a variation of K-Fold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.



Supervised Learning

Hyperparameter search
Cross Validation Schemes

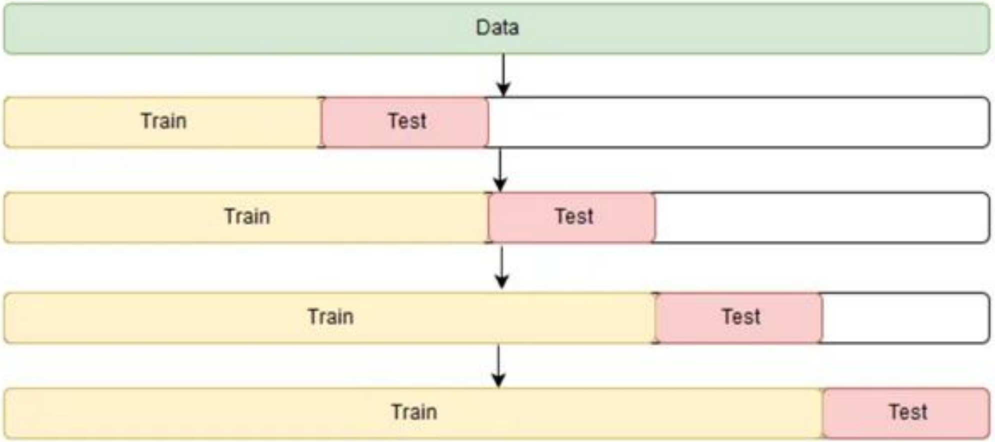
- Time-Series Cross Validation

Supervised Learning

Hyperparameter search

Cross Validation Schemes

- Time-Series Cross Validation
  (sklearn.model_selection.TimeSeriesSplit)
  i.e folds are sequential



Supervised Learning

Hyperparameter search

Cross Validation Schemes

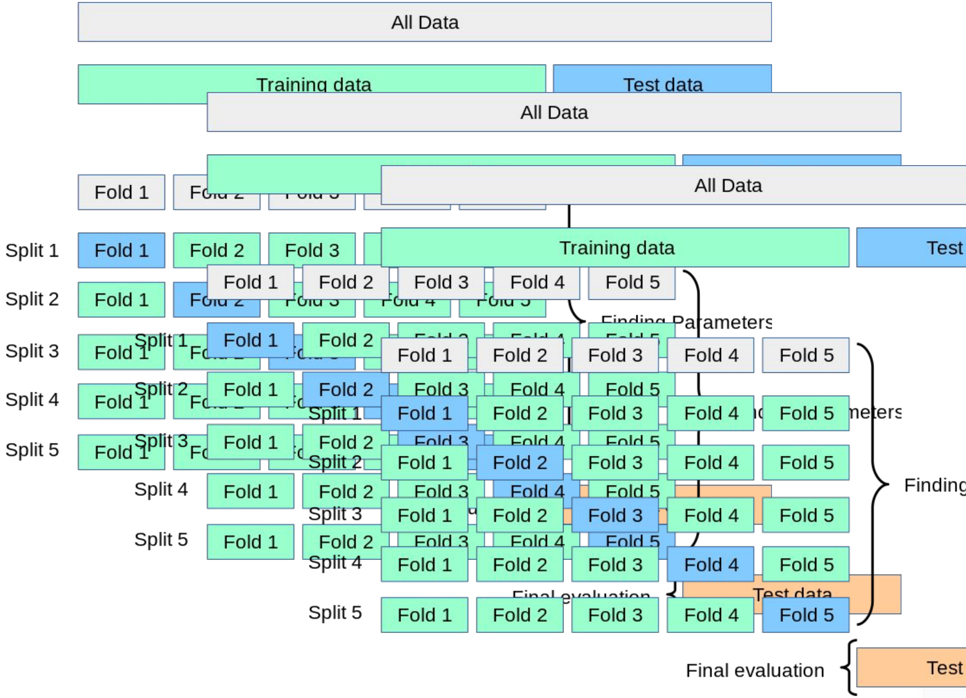
- Repeated Cross Validation

Supervised Learning

Hyperparameter search

Cross Validation Schemes

- Repeated Cross Validation
  i.e repeat k-fold validation multiple times
  (with different splits)



Supervised Learning

Hyperparameter search
consists of:

- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- **a score function**.

F1 Score
Precision
Recall
Accuracy

Supervised Learning

Hyperparameter search consists of:

- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- a score function.

In practice sklearn (and other libs) does most of it for us

Supervised Learning

Hyperparameter search consists of:

- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- a score function.

Supervised Learning

Supervised Learning

Supervised Learning

What if there are more than two sentiments ?

Supervised Learning

What if there are more than two sentiments ?
i.e Multi-class

Supervised Learning

Multi-class problems

Supervised Learning problems that have >= 3 possible classes
i.e positive, neutral, negative
dog, cat, donkey, etc…

Supervised Learning

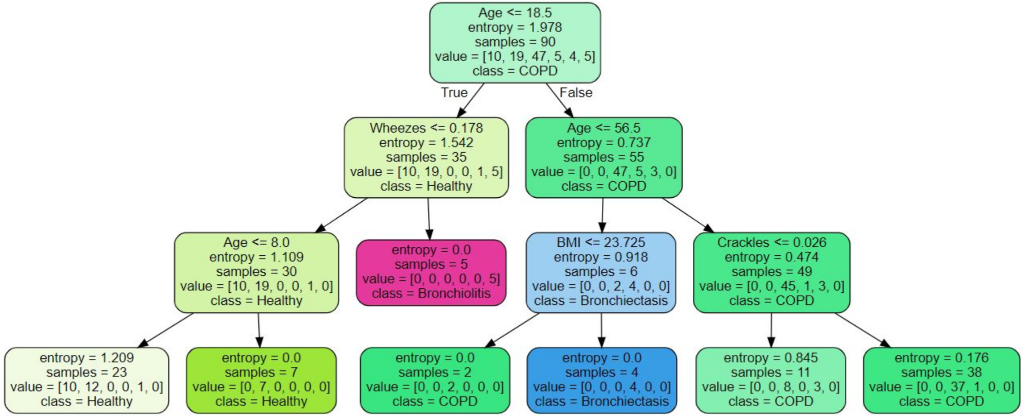
Multi-class problems

- Use models that allow multiple classes i.e decision tree

Supervised Learning

Multi-class problems

- Use models that allow multiple classes i.e decision tree



Supervised Learning

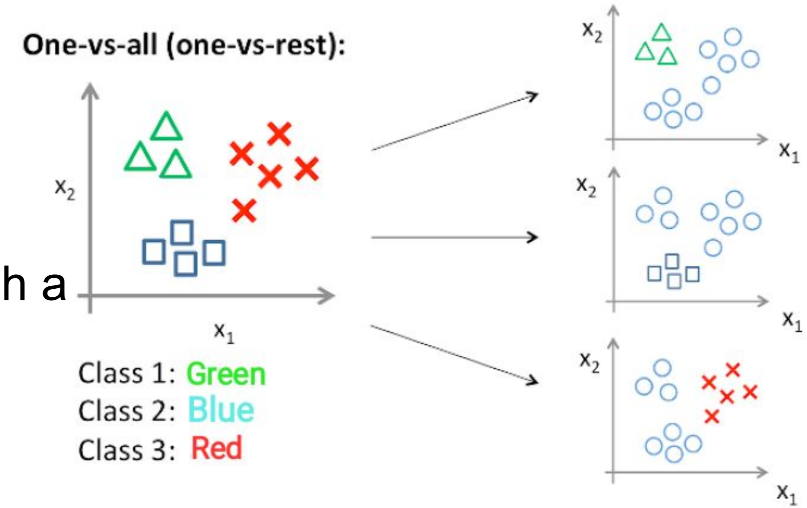
Multi-class problems

- Use models that allow multiple classes i.e decision tree
- Use multiple binary models
  - one for each class i.e one vs all
    (each classifier is trained to distinguish a class from all the others together)

Supervised Learning

Multi-class problems

- Use models that allow multiple classes i.e decision tree
- Use multiple binary models
  - one for each class i.e one vs all
    (each classifier is trained to distinguish a class from all the others together)



Supervised Learning

Multi-class problems

- Use models that allow multiple classes i.e decision tree
- Use multiple binary models
  - one for each class i.e one vs all
    (each classifier is trained to distinguish a class from all the others together)
  - one for each pair of classes i.e one vs one
    (each classifier is trained to distinguish two classes)

Supervised Learning

Multi-class problems

Evaluation

- Accuracy
- Precision (per class)
- Recall (per class)
- …

		Predicted		
		Greyhound	Mastiff	Samoyed
Actual	Greyhound	P_{GG}	P_{MG}	P_{SG}
	Mastiff	P_{GM}	P_{MM}	P_{SM}
	Samoyed	P_{GS}	P_{MS}	P_{SS}

Sentiment Analysis

Supervised Learning

What if there are more than two sentiments ?

Put it to the test, enter a competition.

kaggle

Supervised Learning

Supervised Learning

Classification

- Logistic Classification
- KNN
- …
- Decision Tree
- SVM
- …

Regression

- Logistic Regression
- Decision Tree

Supervised Learning

Naive Bayes

Supervised Learning

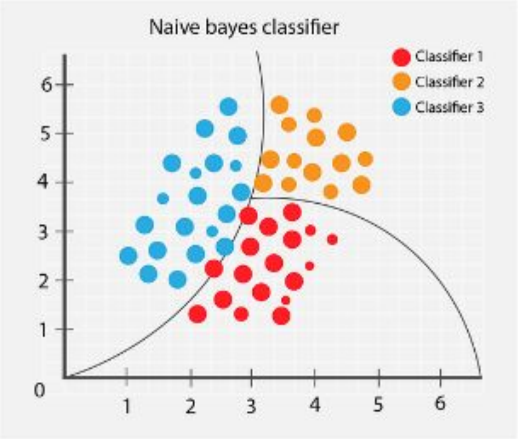
Naive Bayes

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

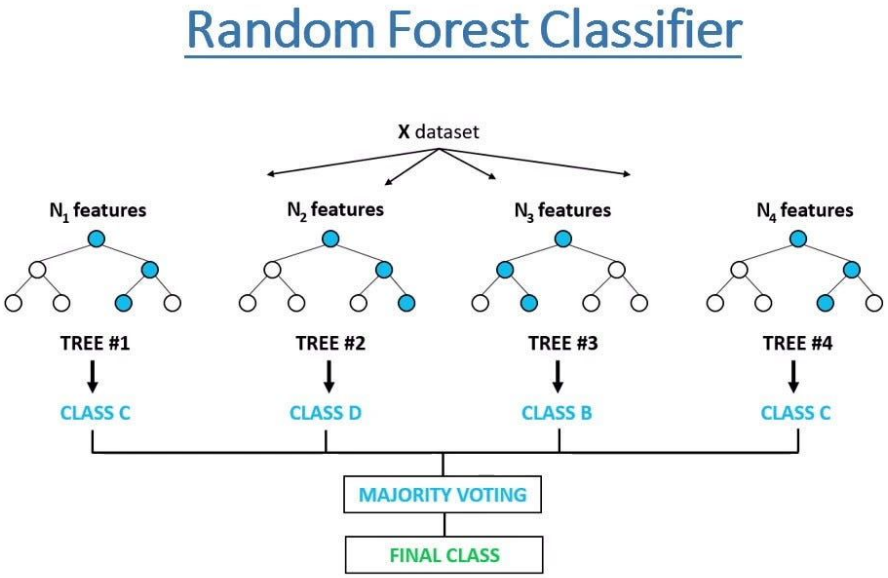
$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$



Supervised Learning

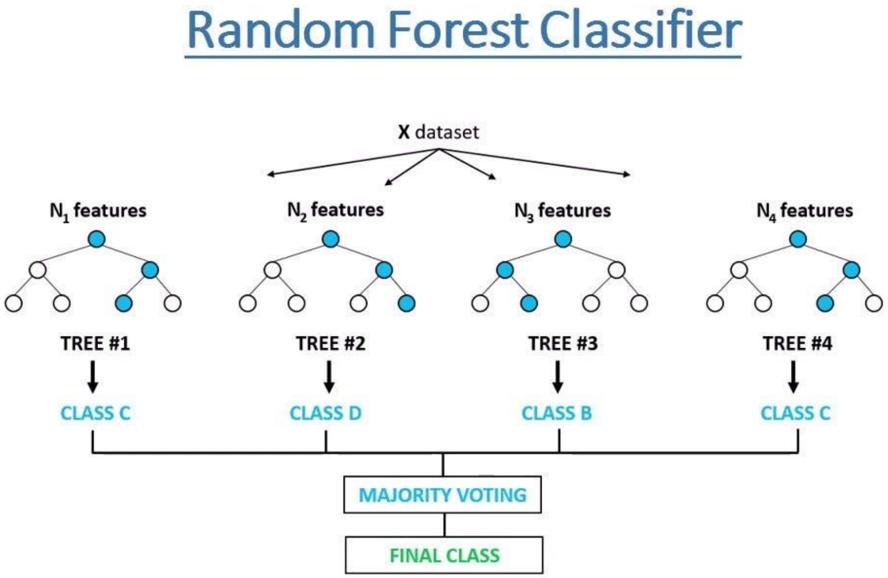
Random Forest

Random Forest



Supervised Learning

Random Forest
Ensemble Method



Supervised Learning

Random Forest
Ensemble Method

- Selects random subsets of data (bagging)
- Selects random features
- Create decision tree for each combination
- Voting

Supervised Learning

(some) Ensemble types

- Voting
- Bagging
- Boosting
- Stacking

Supervised Learning

(some) Ensemble types

- Voting
- Bagging
- Boosting
- Stacking

Supervised Learning

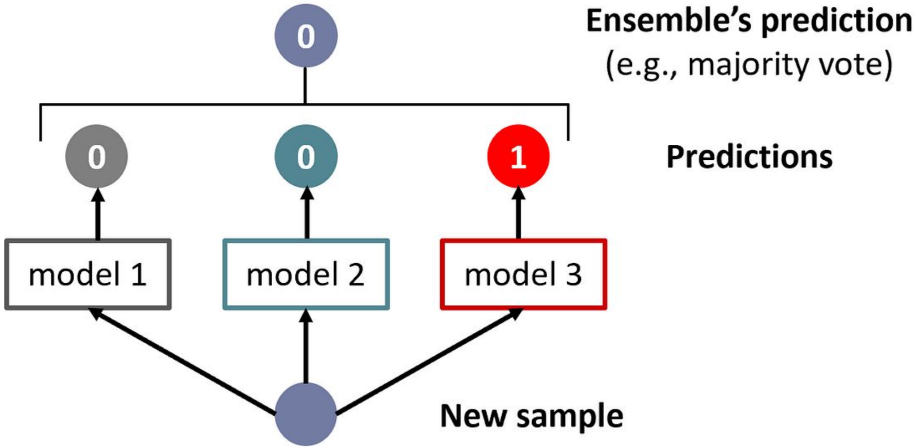
(Ensemble types

- Voting

Supervised Learning

Ensemble types

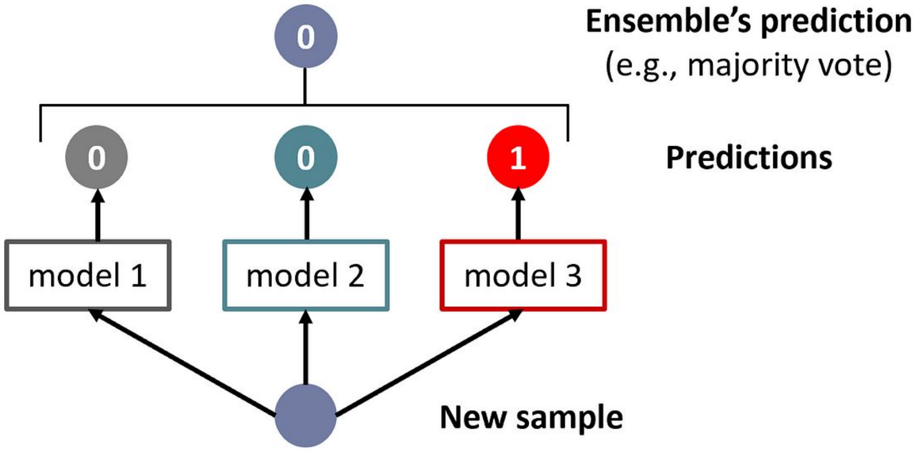
- Voting



Supervised Learning

Ensemble types

- Voting
  - Arithmetic Average
  - Weighted Average



Supervised Learning

(some) Ensemble types

- Bagging (Bootstrap Aggregating)
  - Creates multiple subsets of the data i.e bags
  - Train each model separately
  - Aggregate models through voting for example

Supervised Learning

(some) Ensemble types

- Bagging (Bootstrap Aggregating)
  - Creates multiple subsets of the data i.e bags
  - Train each model separately
  - Aggregate models through voting for example

Supervised Learning

Ensemble types

- Boosting
  - Creates an ensemble by giving more weight to misclassified data

Supervised Learning

Ensemble types

- Boosting
  - Train a base (weak) model on the entire dataset.
  - Assign weights to data points; misclassified points get higher weights.
  - Train the next base model, giving more attention to misclassified points.
  - Repeat steps 2 and 3 iteratively.
  - Combine the base models with different weights to create a final strong model.

Supervised Learning

Ensemble types

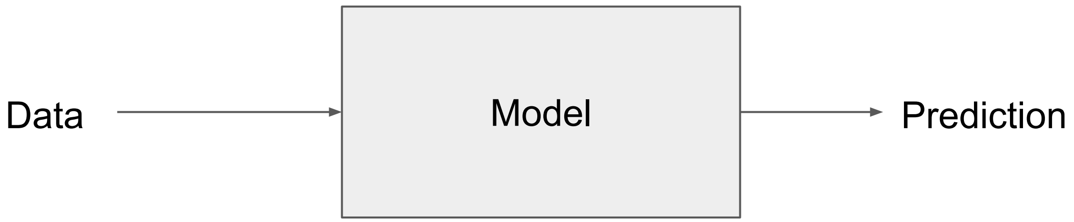
- Boosting
  - Train a base (weak) model on the entire dataset.
  - Assign **weights to data points**; misclassified points get higher weights.
  - Train the next base model, giving more attention to misclassified points.
  - Repeat steps 2 and 3 iteratively.
  - Combine the base models with different weights to create a final strong model.

Examples are XGBoost, AdaBoost

Supervised Learning

Ensemble types

- Boosting



1. Find misclassified points, increase their weight
2. Train again but using the new weights i.e the loss is bigger if the model fails to predict it right

Supervised Learning

Ensemble types

- Stacking
  - Combines multiple models and use their output to train another model
  - Example: Voting but the weights are also learned.

Supervised Learning

Why ensembles?

- Improved Predictive Performance
- Reduces Overfitting
- Robustness to Outliers
- Increased Model Diversity
- Versatility for Various Problem Types

Supervised Learning

Why ensembles?

- Improved Predictive Performance
- ***Reduces Overfitting (variance)***
- ***Robustness to Outliers***
- Increased Model Diversity
- Versatility for Various Problem Types

Supervised Learning

What is variance?

Variance is the ***variability*** of model prediction for a **given data point** or a **value** which *tells us spread of our data*. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

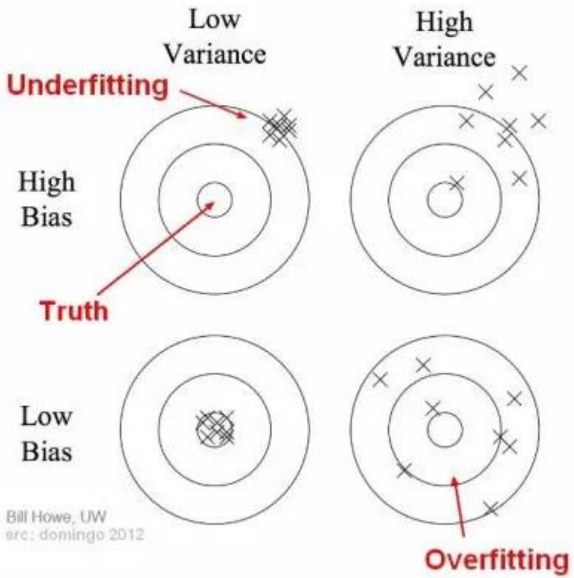
Supervised Learning

What is bias?

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

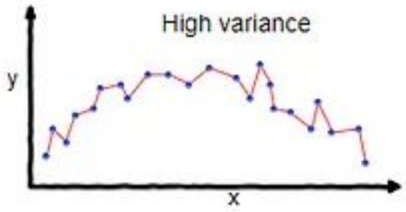
Supervised Learning

Bias Variance Tradeoff

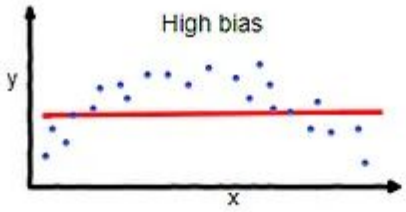


Supervised Learning

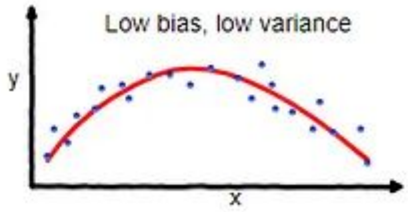
Bias Variance Tradeoff



overfitting



underfitting



Good balance

Supervised Learning

Bias Variance Tradeoff

- If the model is too simple and has very few parameters then it may have high bias and low variance.
- If the model has large number of parameters then it’s going to have high variance and low bias.
- Thus we need to find the right/good balance without overfitting and underfitting the data.

Supervised Learning

Bias Variance Tradeoff

Total Error = Bias^2 + Variance + Irreducible Error