CLASSIFICATION

Carlos J. Costa

Learning Goals

- Know concept of classification
- Distinguish between main algorithms
- Apply algorithms by using python libraries

Summary

- Concept of classification
- Algorithms
  - K -Near Neighbour (KNN)
  - Support Vector Machines (SVM)
  - Naive Bayes
  - Logistic Regression
  - Decision Trees
  - Ensemble

Classification

Categorizing some unknown items into discrete set of categories or “classes”

Classification

age	address	income	ed	employ	equip	callcard	wireless	churn
33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	Yes
33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	Yes
30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	No
35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	No

age	address	income	ed	employ	equip	callcard	wireless	churn
35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	No
35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	?

Classification

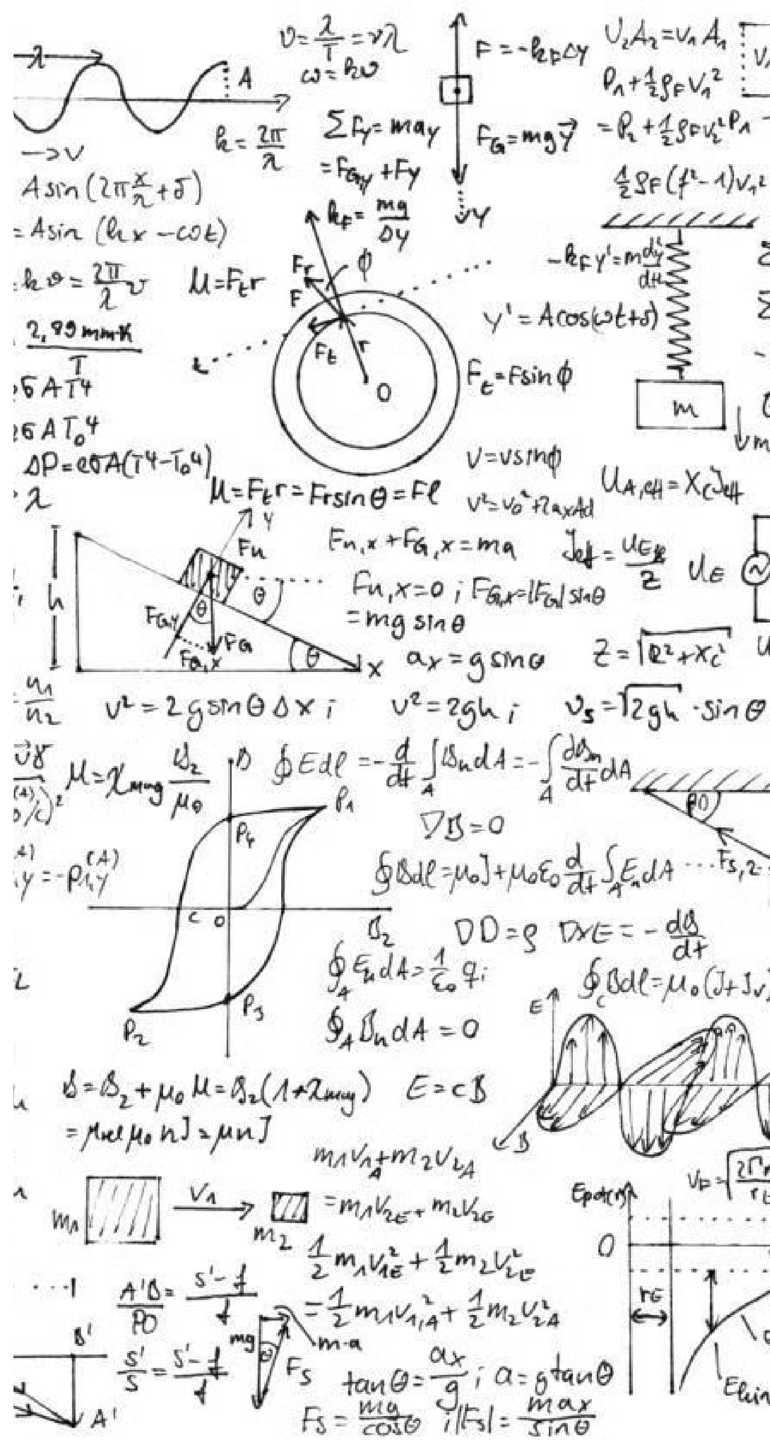
Age	Sex	BP	Cholesterol	Na	K	Drug
23	F	HIGH	HIGH	0.793	0.031	drugY
47	M	LOW	HIGH	0.739	0.056	drugC
47	M	LOW	HIGH	0.697	0.069	drugC
28	F	NORMAL	HIGH	0.564	0.072	drugX
61	F	LOW	HIGH	0.559	0.031	drugY
22	F	NORMAL	HIGH	0.677	0.079	drugX
49	F	NORMAL	HIGH	0.79	0.049	drugY
41	M	LOW	HIGH	0.767	0.069	drugC
60	M	NORMAL	HIGH	0.777	0.051	drugY
43	M	LOW	NORMAL	0.526	0.027	drugY

Categorical Variable

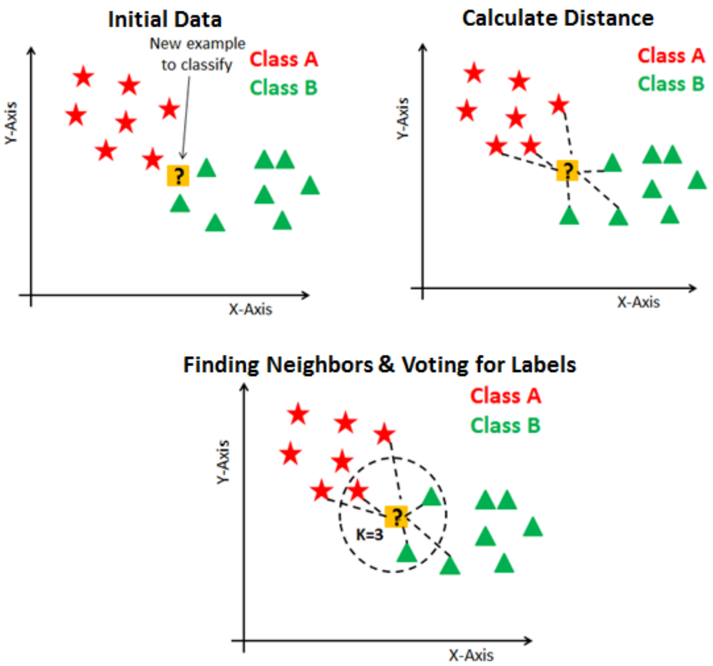
Age	Sex	BP	Cholesterol	Na	K	Drug
36	F	LOW	HIGH	0.697	0.069	

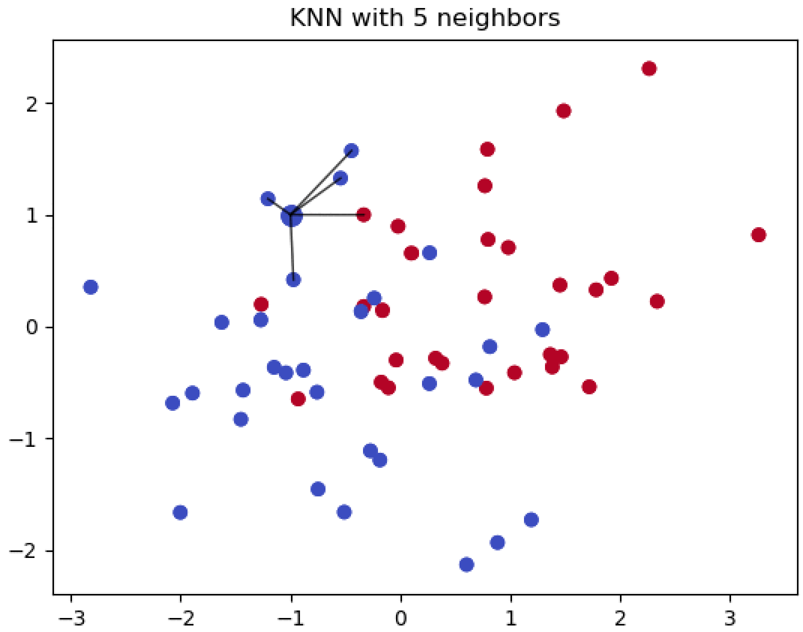
Algoritms

- K -Near Neighbour (KNN)
- Support Vector Machines (SVM)
- Naive Bayes
- Logistic Regression
- Decision Trees



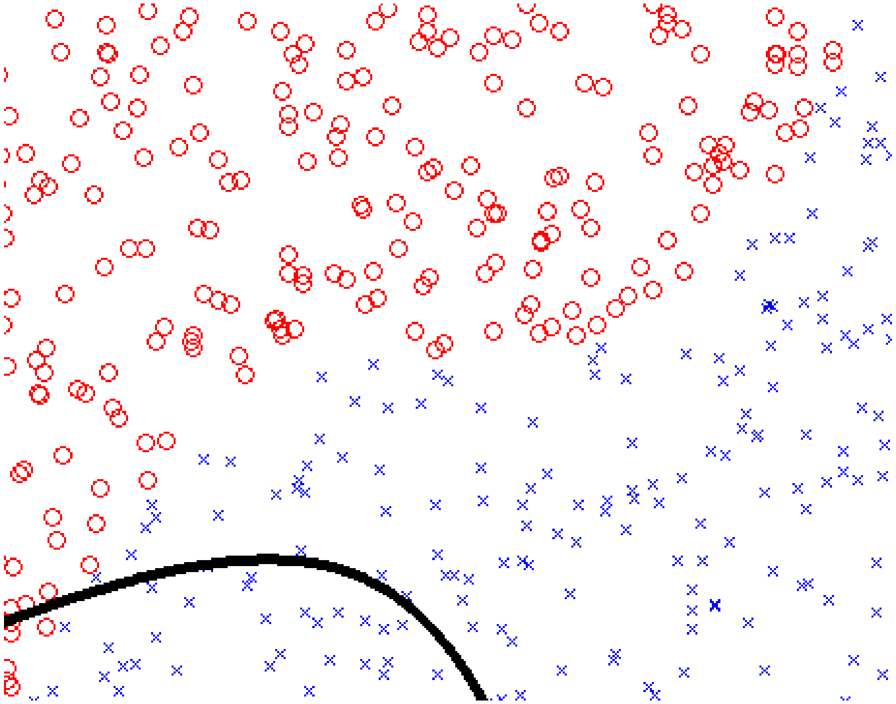
KNN


Initial Data
New example to classify
Class A
Class B
Y-Axis
X-Axis
Calculate Distance
Class A
Class B
Y-Axis
X-Axis
Finding Neighbors & Voting for Labels
Class A
Class B
K=3
Y-Axis
X-Axis


KNN with 5 neighbors
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

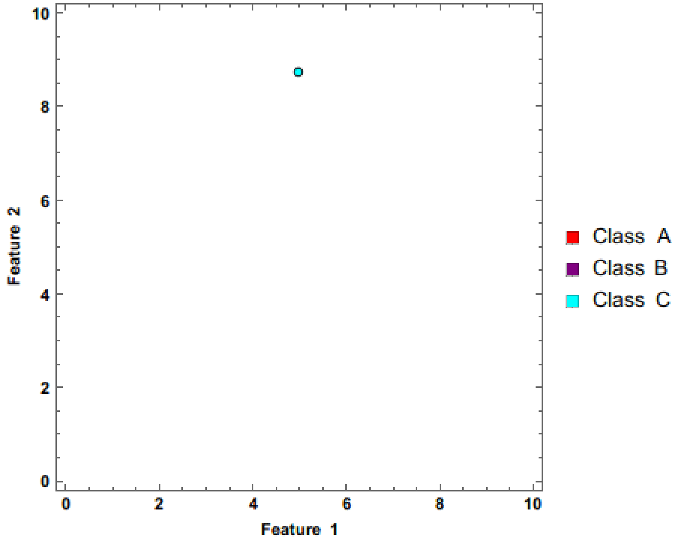
```python
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris

# Load a sample dataset (Iris dataset)
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
  test_size=0.3, random_state=42)

# Create a KNN classifier with k=3
knn = KNeighborsClassifier(n_neighbors=3)

# Train the classifier
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Evaluate the accuracy
accuracy = accuracy_score(y_test, y_pred)
• print(accuracy)
```

SVM (support vector machine)



```python
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris

# Load a sample dataset (Iris dataset)
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
  random_state=42)

# Create an SVM classifier with a linear kernel
svm = SVC(kernel='linear')

# Train the classifier
svm.fit(X_train, y_train)

# Make predictions
y_pred = svm.predict(X_test)

# Evaluate the accuracy
accuracy = accuracy_score(y_test, y_pred)
• print(f"Accuracy: {accuracy:.2f}")
```

Naive Bayes



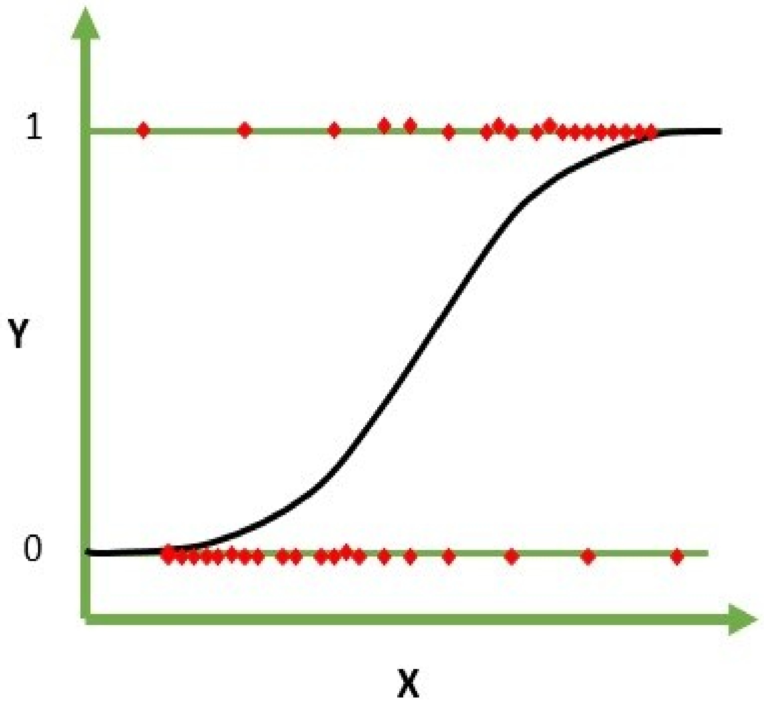
$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

$$X = (x_1, x_2, x_3, ..., x_n)$$

It assumes that all the features in a class are unrelated to each other.

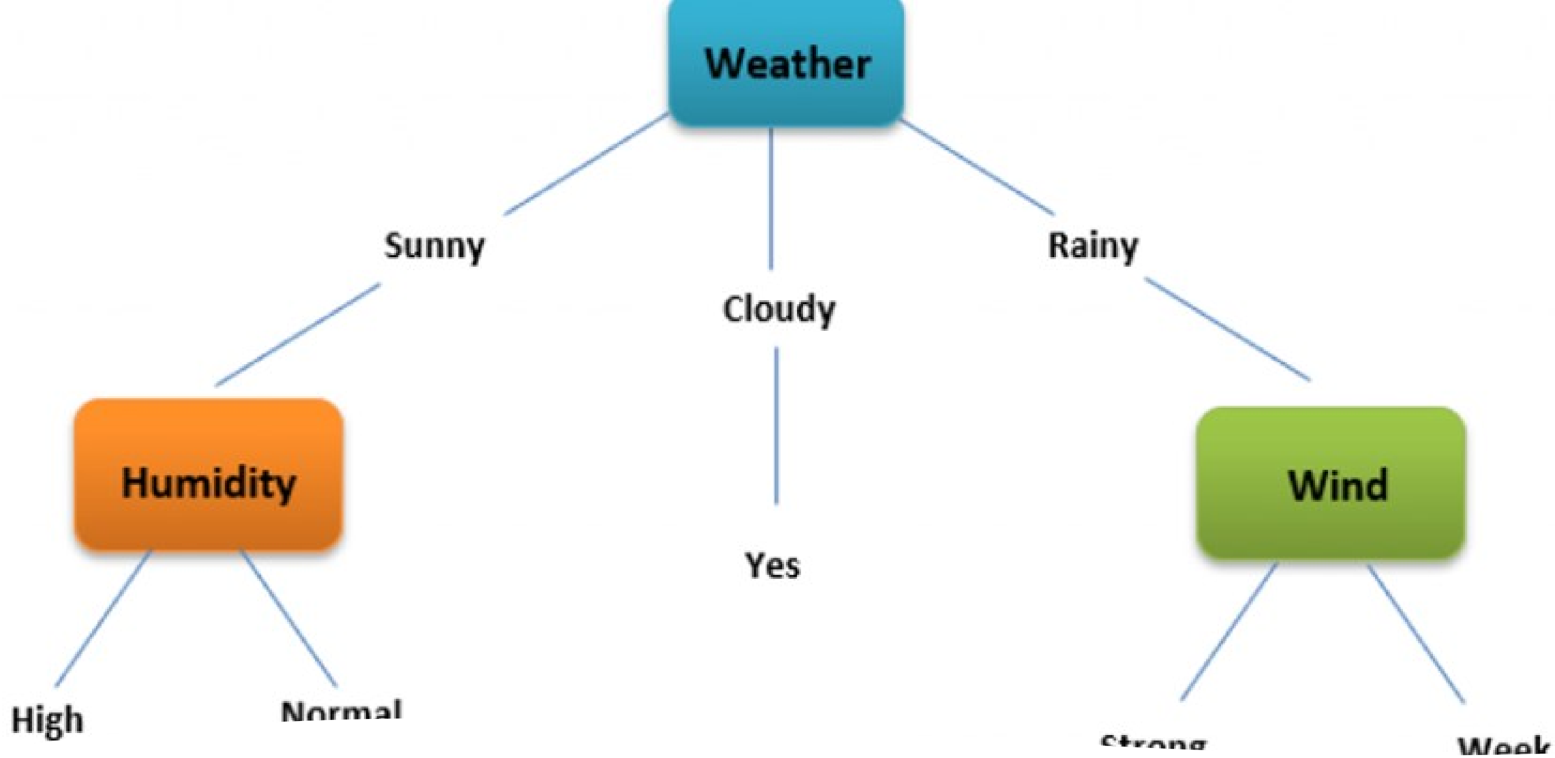
```python
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Create a Gaussian Naive Bayes classifier
gnb = GaussianNB()

# Train the classifier
gnb.fit(X_train, y_train)

# Make predictions on the test set
y_pred = gnb.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

Logistics Regression



```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
# Load a sample dataset (Iris dataset)
iris = load_iris()
X = iris.data
y = iris.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
  random_state=42)
# Create a Logistic Regression classifier
logistic_regression = LogisticRegression(max_iter=1000) # Increased max_iter
# Train the classifier
logistic_regression.fit(X_train, y_train)
# Make predictions
y_pred = logistic_regression.predict(X_test)
# Evaluate the accuracy
accuracy = accuracy_score(y_test, y_pred)
```

- `print(accuracy)`



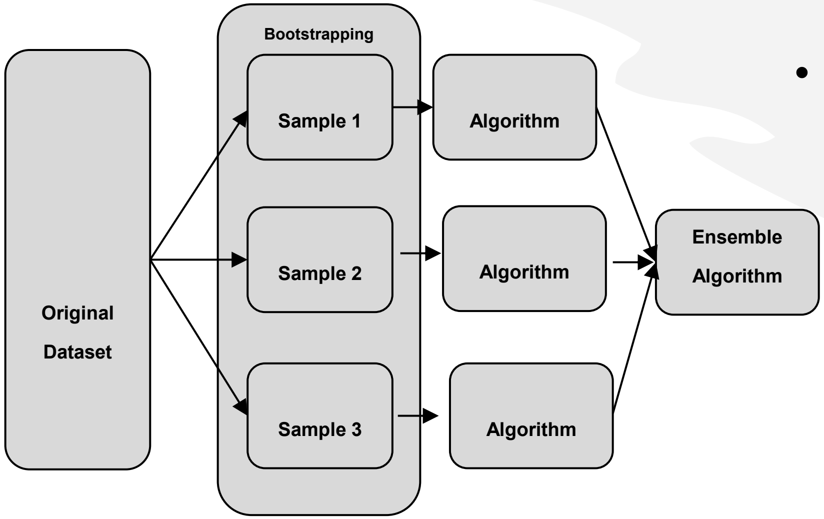
Decision Tree

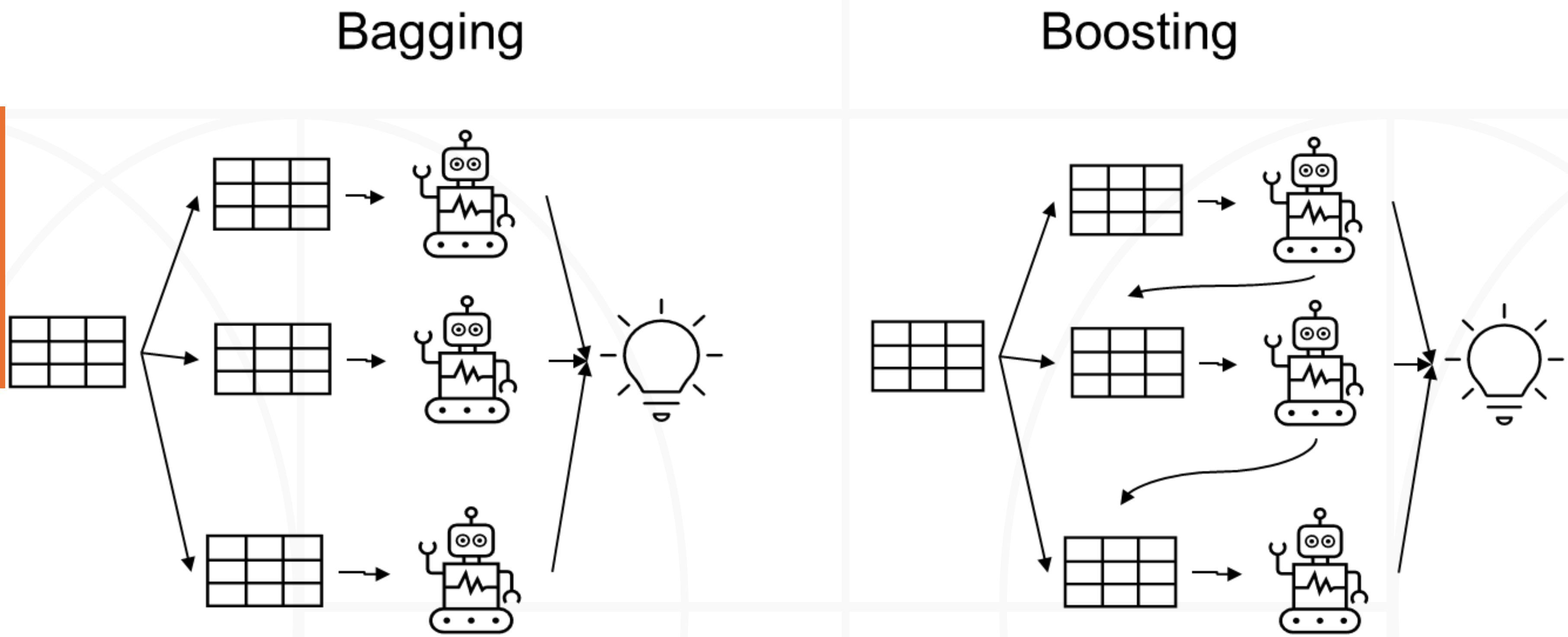
- is a non-parametric supervised learning algorithm
- Is used for classification and regression tasks.
- has a hierarchical tree structure consisting of:
  - root,
  - branches,
  - leaf.
- easy-to-understand models.

```python
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_wine

# Load the wine dataset
wine = load_wine()
X = wine.data
y = wine.target

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
  random_state=42)

# Initialize a Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Train the classifier
dt_classifier.fit(X_train, y_train)

# Make predictions
y_pred = dt_classifier.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

Ensemble

- is a Machine Learning concept
- the idea is to train multiple models using the same learning algorithm.
- used for classification, regression
- multitude of decision trees at training time
- outputting the class that is the mode of the classes (classification)

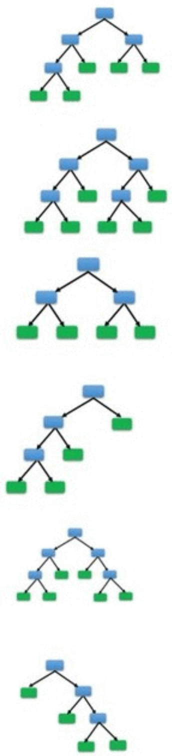




Bagging vs. Boosting

- classification, regression and other tasks
- multitude of decision trees at training time
- outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Random Forest



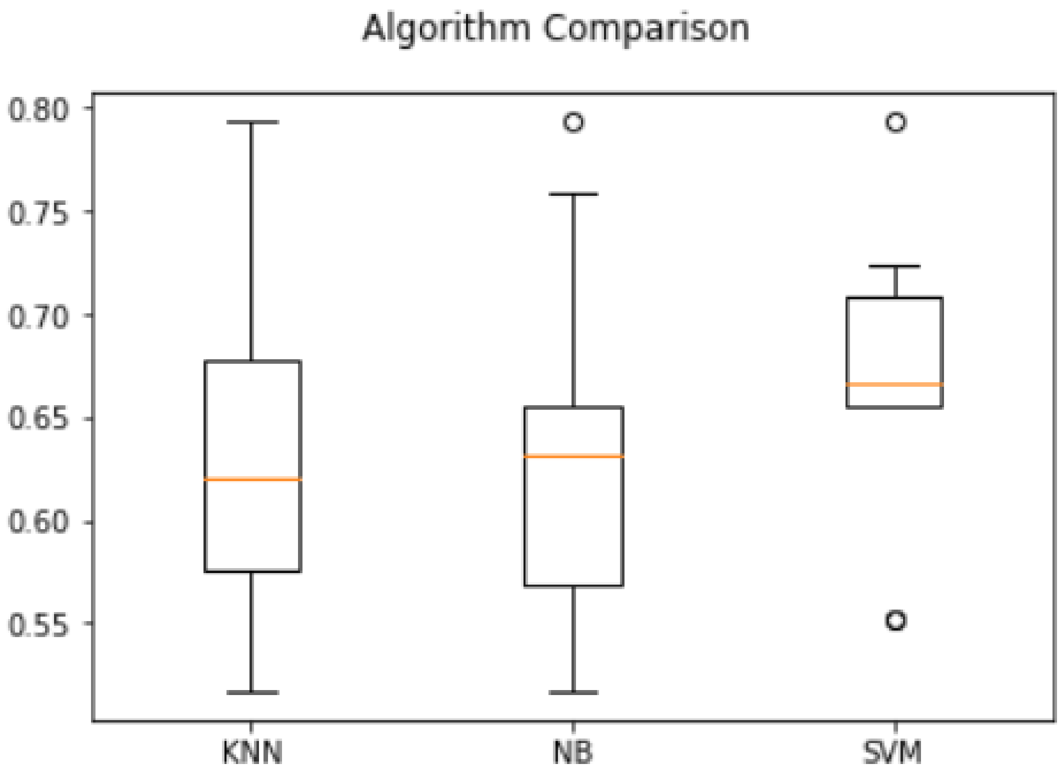
Random Forest in Action!!!

```python
from sklearn.preprocessing import StandardScaler
standardizer=StandardScaler()
X=standardizer.fit_transform(Xfeatures)
```

```python
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

results = []
names = []
scoring = 'accuracy'

seed = 7

for name, model in models:
    #, random_state=seed
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
KNN: 0.635222 (0.084238)
NB: 0.635099 (0.084984)
SVM: 0.666872 (0.070033)
```

```python
import matplotlib.pyplot as plt

fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



Conclusions

- Classification
- K -Near Neighbour (KNN)
- Support Vector Machines (SVM)
- Naive Bayes
- Logistic Regression
- Decision Trees
- Ensemble

References

- Albon, Ch. (2018) *Machine Learning with Python Cookbook*. O'Reilly
- Domingos, P. (2015) *The Master Algorithm*, Penguin Books
- Hinton, J.; Sejnowski, T.(1999). *Unsupervised Learning: Foundations of Neural Computation*. MIT Press
- Morgan; P. (2019) *Data Science from Scratch with Python*, AI Science
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective* (1 edition). Cambridge, MA: The MIT Press.
- Otte, E.; Rousseau, R. (2002). "Social network analysis: a powerful strategy, also for the information sciences". *Journal of Information Science*. 28 (6): 441–453. doi:10.1177/016555150202800601.
- Stuart J. R., Norvig, P. (2010) *Artificial Intelligence: A Modern Approach*, Third Edition, Prentice Hall ISBN 9780136042594.