Machine Learning

Bahram Mobasher

Definitions

Machine Learning (ML) can be defined as computational methods using experience to improve performance or to make accurate predictions.
Here experience means the past information available to the learner. This takes form of electronic data collected and made available for analysis. The quality and size of the data are crucial for the success of the predictions.
ML lies at the intersection between statistics and computer science. It is applied on many disciplines from history and literature to physics, engineering and biology.

Further Definitions

ML consists of designing efficient and accurate prediction algorithms. It involves the algorithms used to extract useful information from data. ML looks for patterns in the data and turns data into information.

ML is a set of methods that can automatically detect patterns in data then perform decision making or predict future developments.

Machine learning is a branch of artificial intelligence that allows computer systems to learn directly from examples, data, and experience. Through enabling computers to perform specific tasks intelligently, machine learning systems can carry out complex processes by learning from data, rather than following pre-programmed rules.

Royal Society Report on Machine Learning

Traditional Programming

Machine Learning

Slide from Pedro Domingos

Machine Learning: A History

This all started in 1959 when Arthur Samuel first defined ML as “a field of study that gives computers the ability to learn without being explicitly programmed”.

Tom Mitchell provided an analytic definition: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”.

As a scientific field ML grew out of Artificial Intelligence (AI). Some researchers were interested in having machines learn from the data. This was termed “neural networks”. These were mainly perceptron that were found to be generalized linear models of statistics. Between 1980s and 1990s ML based on statistics was out of favor with the researchers invented back propagation.

ML flourished during 1990s by changing its goals from AI to problems of practical nature. It started to use methods from statistics and probability theory. It benefited from availability of digitized data.

ML vs. Data Mining:

ML and data mining often use the same methods. They can be distinguished as follows:

ML focuses on prediction based on known properties learned from training data. Data mining focuses on discovery of unknown properties in the data. Data mining uses many ML techniques but for a different goal. ML uses data mining methods as unsupervised learning or as a step to improve

Applications of ML

ML techniques have been successfully used in a diverse range of disciplines Including:

- Text or document classification
- Language processing (e.g. morphological analysis, statistical parsing)
- Optical character recognition
- Computational; biology (e.g. protein function or structure prediction, computer vision)
- Fraud detection and network intrusion.
- Medical diagnosis
- Search engines, information extraction systems
- Quantitative Finance

Classes of Learning Problems

For different applications of ML, different techniques are used that include:

Classification: This assigns a category to each item. For example, document classification may assign items with categories such as politics, business, sports or weather. Image classification may assign items like landscape, portrait or animal. The number of categories is often small but they could be large in case of, for example, text classification or speech recognition.

Regression: Predict a real value for each item. Examples include prediction of stock values or variation of economic variables. The penalty for an incorrect prediction depends on the magnitude of the difference between the predicted and the true values This is in contrast with the classification where there is no notion of closeness between various categories.

Ranking: Orders items according to some criterion. The examples include returning web pages relevant to a search query.

Clustering: Partitions items into homogeneous regions. This is often used to analyze very large data sets. Examples include identification of communities within large groups of people (in the context of social network analysis).

Dimensionality Reduction: This transforms an initial representation of items into a lower dimensional representation of these items, while preserving some properties of the initial representation. Examples include processing digital images in computer vision tasks.

Types of ML

ML is divided into two types:

Supervised ML: Learn a mapping from inputs x_i to outputs y_i, given a labeled set of input-output pairs $D = \{(x_i , y_i)\}^N_{i=1}$ Here D is called the training set and N is the number of training examples.

Each training input x_i is a D-dimensional vector of numbers representing for example, the weight and height of someone. These are called *features* or *attributes*. These are stored in an N x D matrix, X. In general x_i could be anything – an image, a sentence a shape etc.

Similarly, the form of the output, y_i, can in principle be anything. In most methods y_i is assumed to be a nominal variable from some finite set $y_i = \{1,\ldots,C\}$. When y_i is categorical, the problem is known as classification or pattern recognition. And when y_i is real-valued, the problem is known as regression.

Unsupervised ML: Here we are only given inputs $D=\{x_i\}^N_{i=1}$ and the goal is to find patterns in the data. Here we are not told what kind of patterns to look for and there is no obvious error metric to use (unlike supervised ML where we can compare our prediction *y* for a given *x* to the expected value).

Unsupervised Learning

Clustering:

In these methods of learning we usually have a data set without any label and we are trying to find an underlying patterns. In the language of probability, we are trying to find the probability density of the inputs. We want to come up with a **priori** probability distribution.

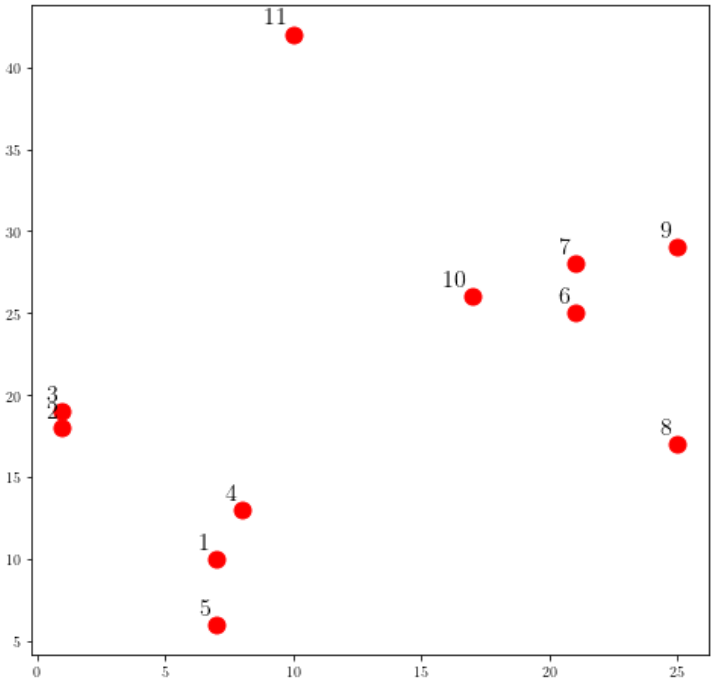
1. Hierarchical Clustering
2. K-means Clustering

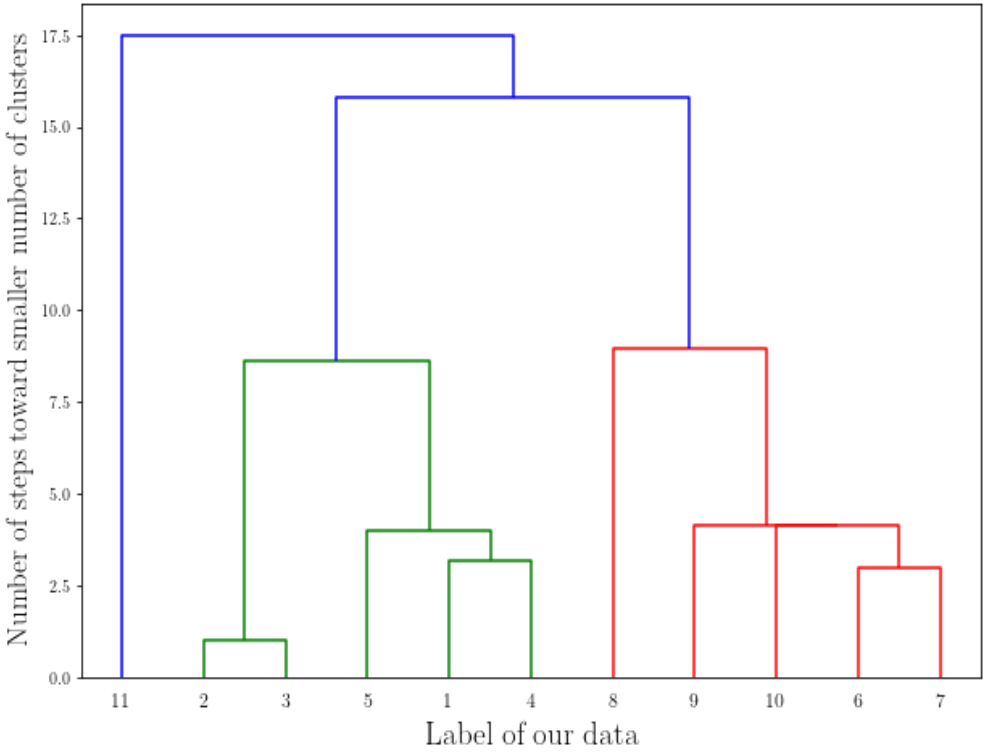
Hierarchical Clustering

Naive algorithm: Agglomerative hierarchical clustering

1. Assign a cluster to each point.
2. Find the most similar clusters and merge them together.
3. We do the second part until we get to a cluster that contains all of the points.

Example:


11
40
35
30
25
20
15
10
5
0
5
10
15
20
25
9
7
10
6
3
2
8
4
1
5


Number of steps toward smaller number of clusters
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
11
2
3
5
1
4
8
9
10
6
7
Label of our data

K-means Clustering:

Algorithm:

1. Start by randomly choosing k examples as our initial centroid.

2. Create k clusters by assigning examples to closest centroid.

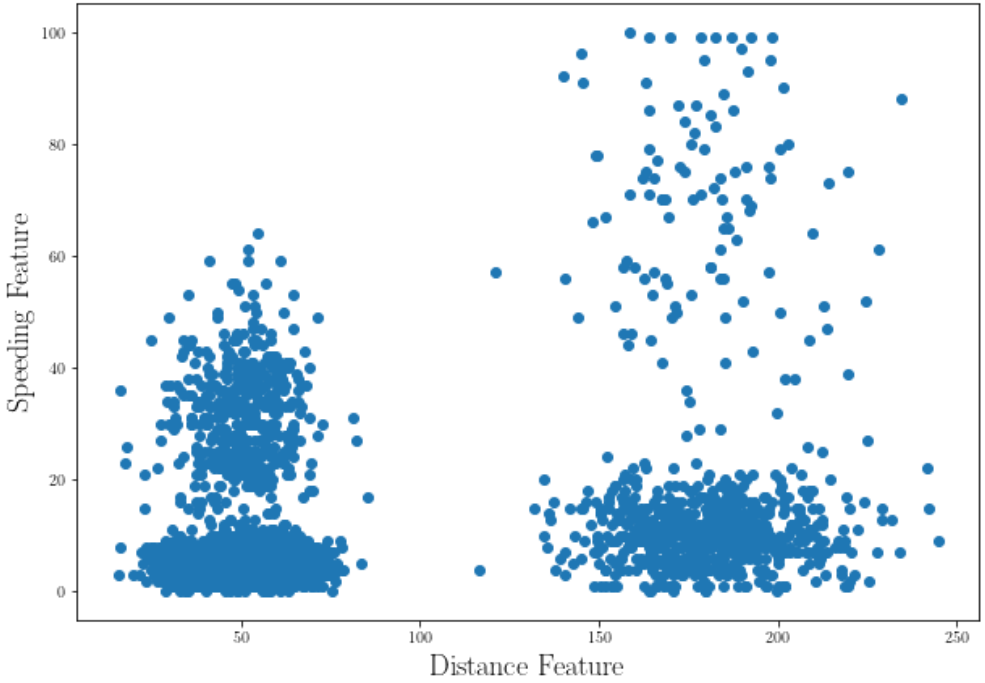
3. Assign a new values for the centroid: The average of previous cluster configuration.

4. Go to (2) while centroids are changing, else break. We have our final configuration.

Example: Delivery Fleet Data Set

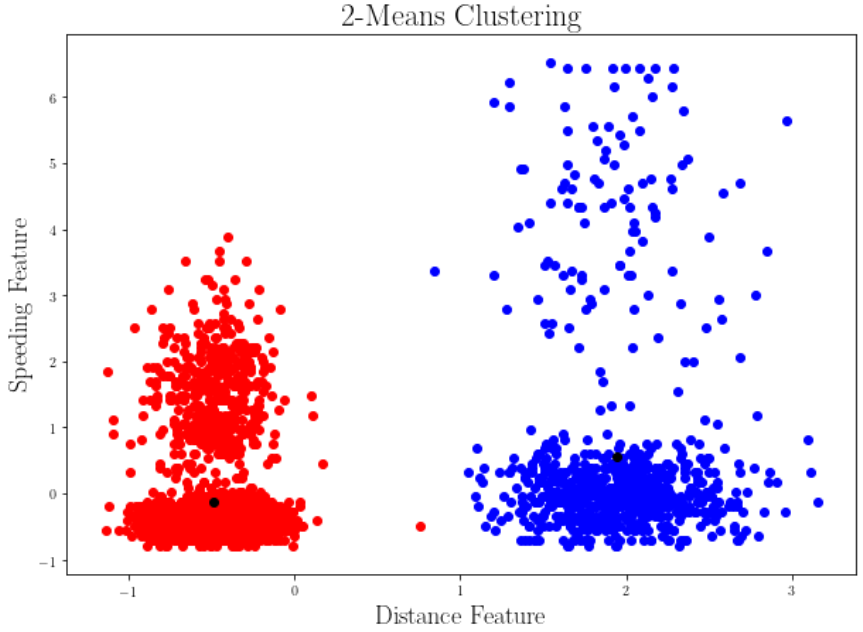
Two Features are important:

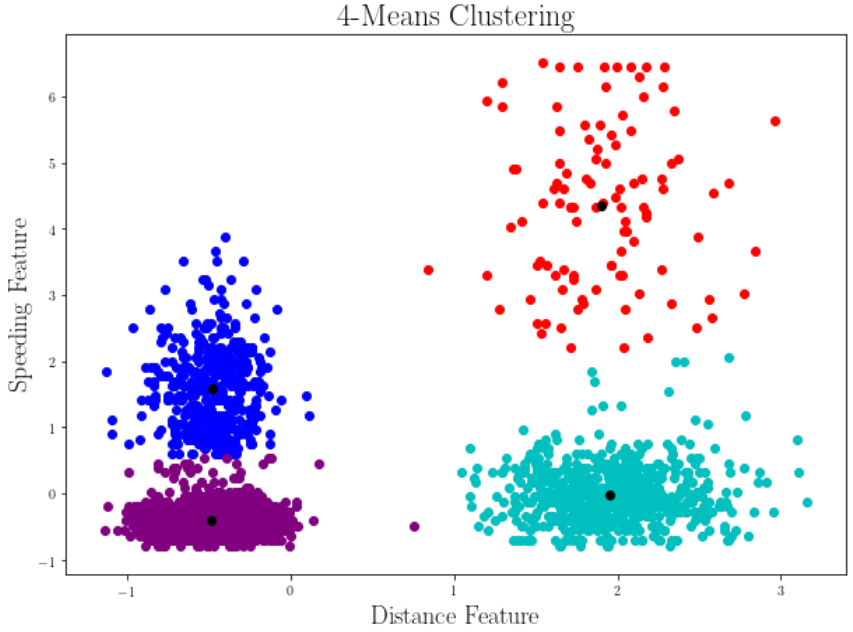
1. Speed
2. Distance



After Scaling the Features:

Note: Feature scaling is necessary when dealing with features with different dynamic range.



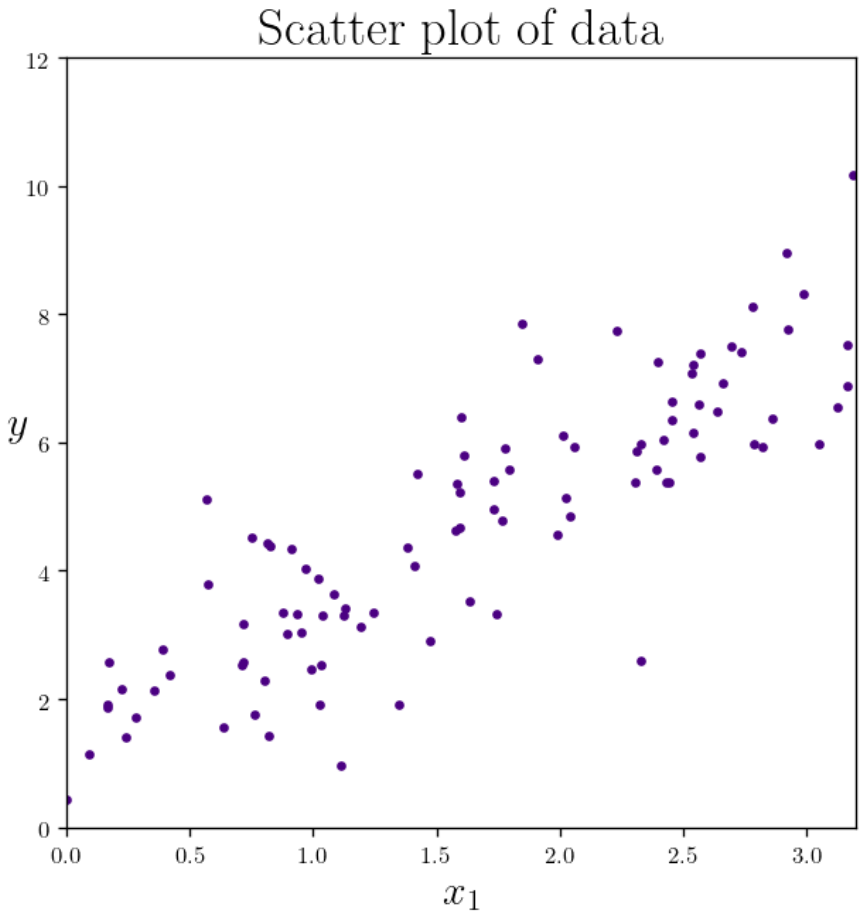


Supervised Learning:

- **Regression**:
  - Fitting curve to data
  - Finding a Real valued measure for data vectors (features)
- **Classification**:
  - Associate each data vector to a label. (N-dimensional vector of features to a discrete value which are the labels.)

Regression:

Fitting a Curve (Model)



Regression:

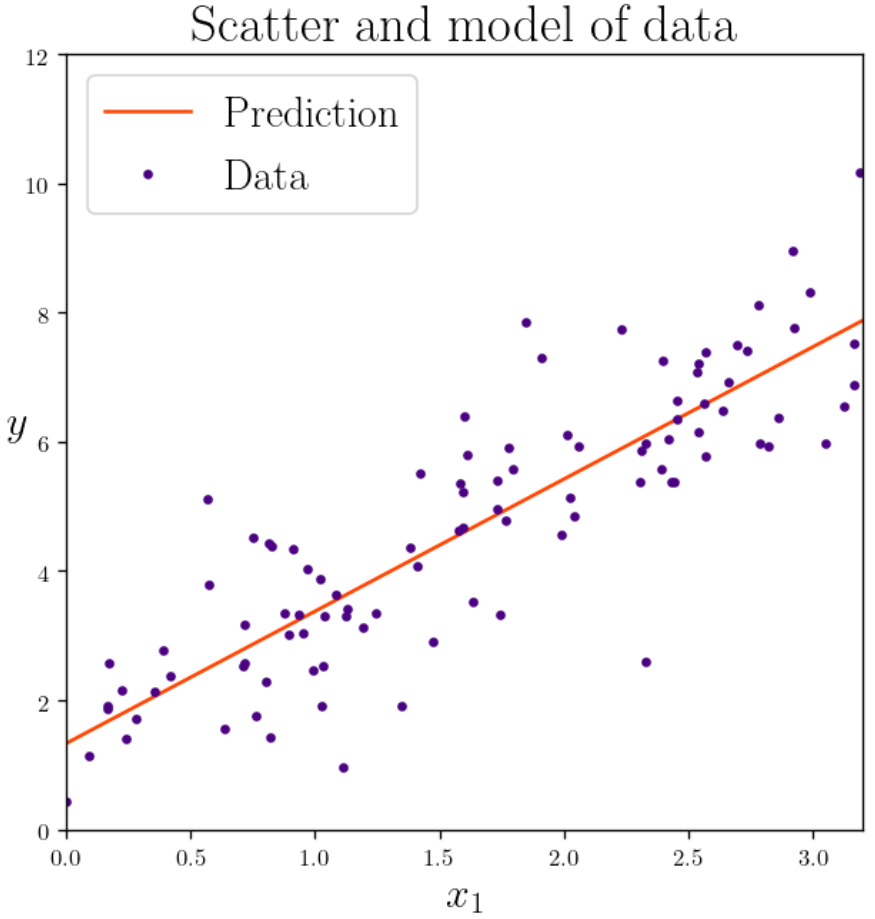
Finding the best model parameters t

minimize the Loss function.

Ordinary least squares:

$$r_i = y_i - f(x_i, \vec{\alpha})$$

$$L_2 = \sum_i r_i^2$$



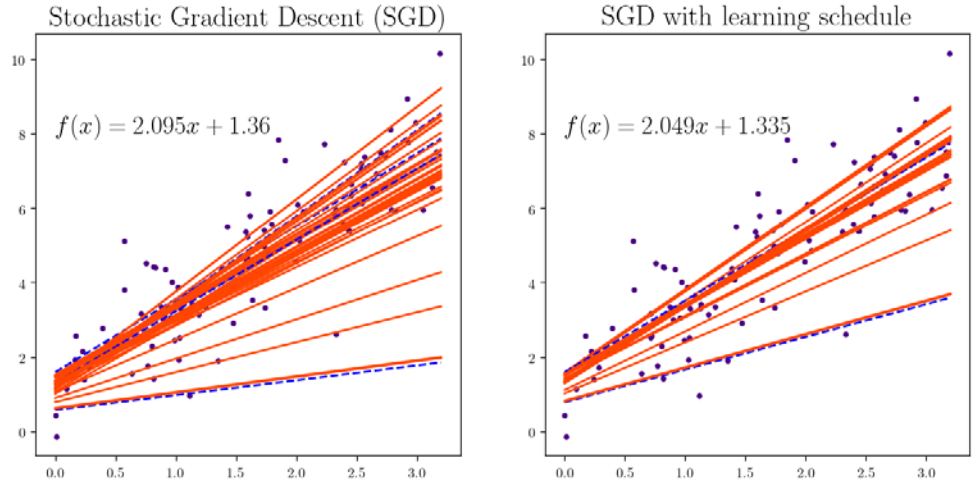
Finding The Best Model Parameters:

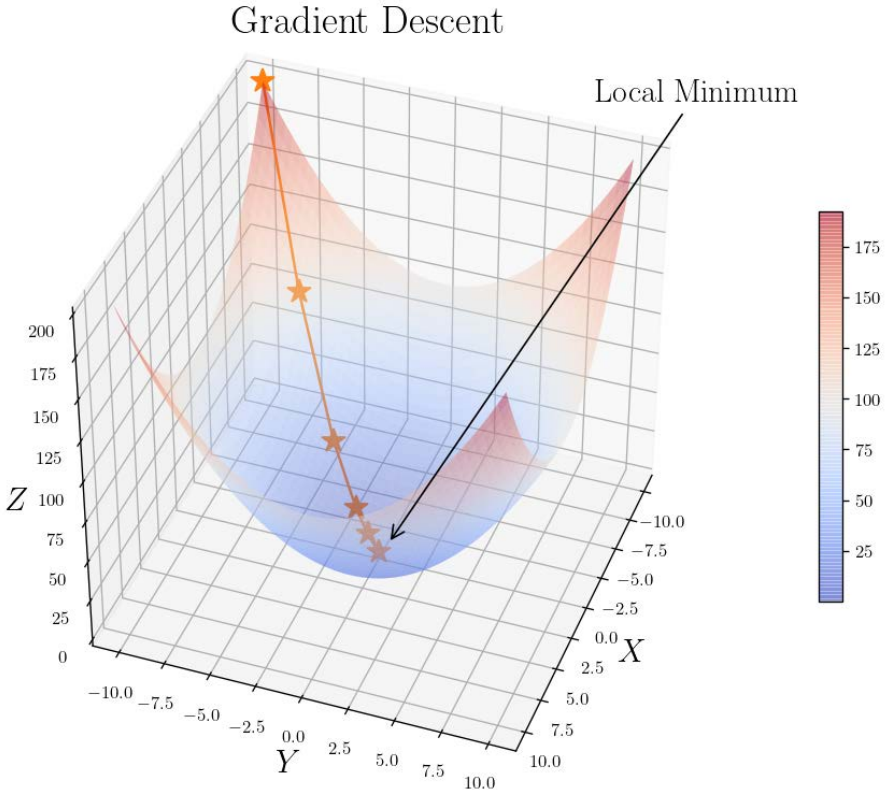
- Sometimes an **analytical solution** exists.
- When the **analytical solution** does not exist or is computationally expensive:

We can use some form of **Gradient descent** on the parameter space:

This is basically finding the parameters that minimize/maximize the loss/gain function, by moving in the direction of the gradient at each point in parameter space.

Gradient Descent in the space of parameters:





Classification:

1. K-Nearest Neighbors
2. Decision Tree
3. Random Forest of Decision Trees
4. Support Vector Machines

Nearest Neighbor:

One of the most commonly used algorithm for classification is **nearest neighbour**. In which you just remember the data you trained on and you will not perform any fancy manipulation on the data.

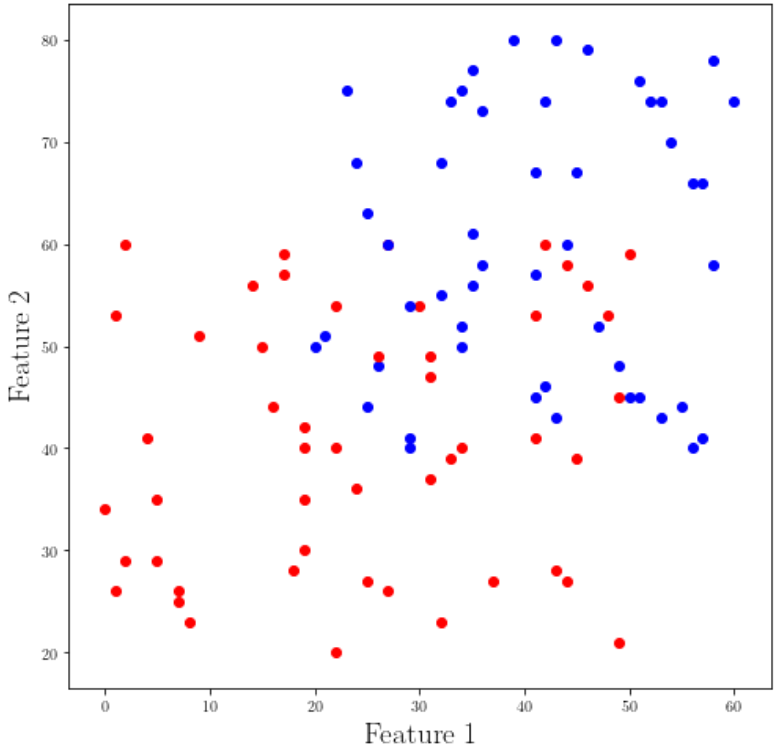
Algorithm:

- Find the nearest example from the training data to the new data.
- Assign the known label of that example to your new data.

Nearest Neighbor

Data set with labels :

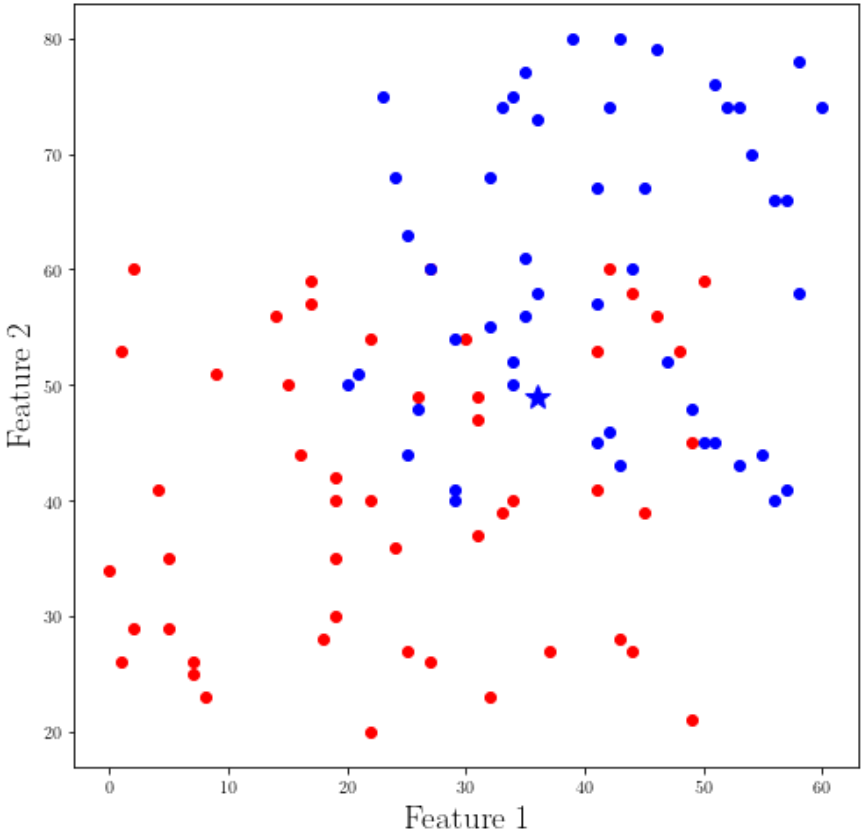
1. Blue
2. Red



Nearest Neighbor

Now we classify the new point (*)

with finding the nearest neighbor

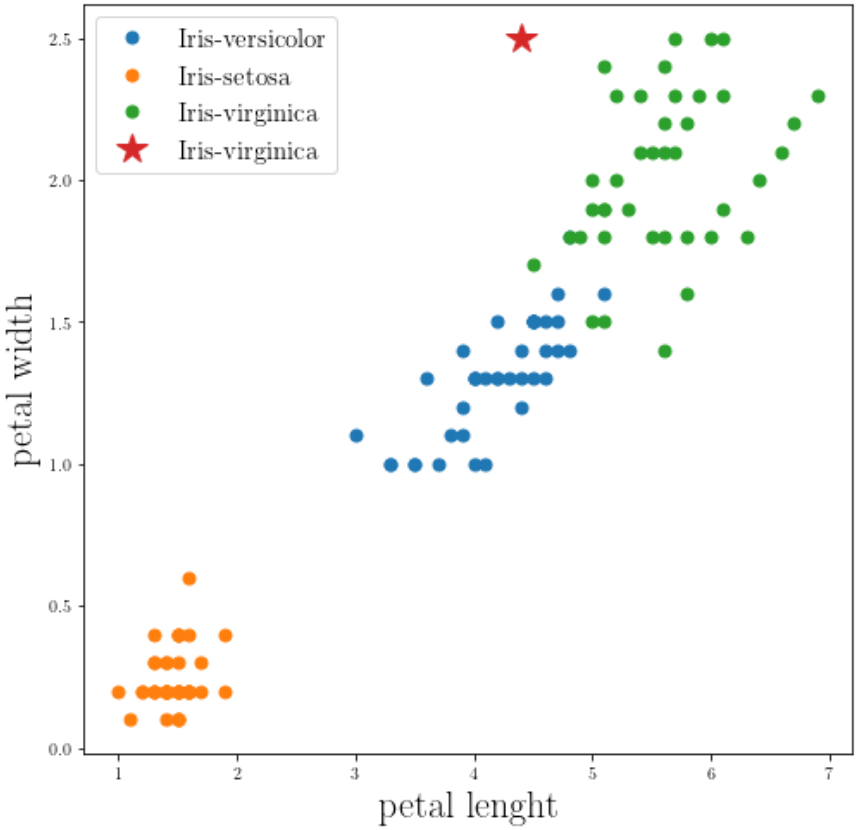


K-Nearest Neighbors:

This is very similar to nearest neighbour algorithm but this time we are looking for **K** nearest neighbours and to take **the most occured label** of those K neighbours as your prediction for new data.

Example: iris flowers data set

The closest 5-Neighbors to star (*) is **virginica**.

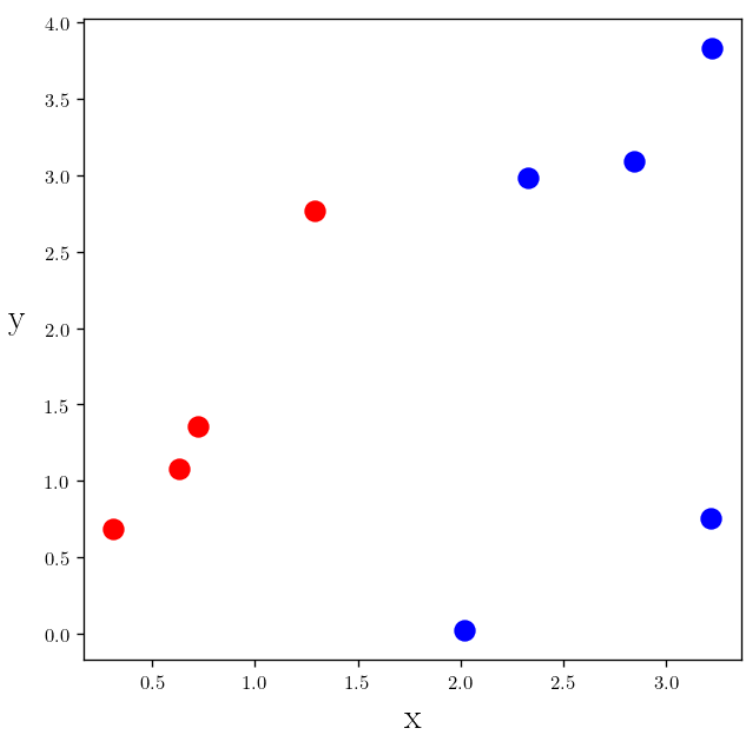


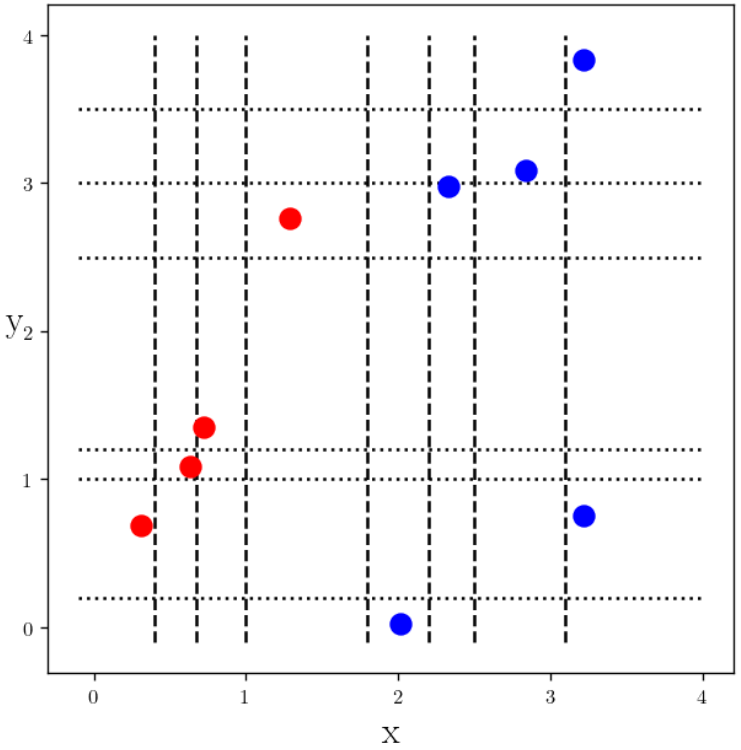
Decision Tree:

1. Find the most important features that make the classification more pure by finding the **purity measure** (Gini Index). Or finding the feature that maximize the **information gain** (Entropy measure).
2. Do the same thing with the rest of the features until reaching to a maximum depth of the tree.

Decision Tree:

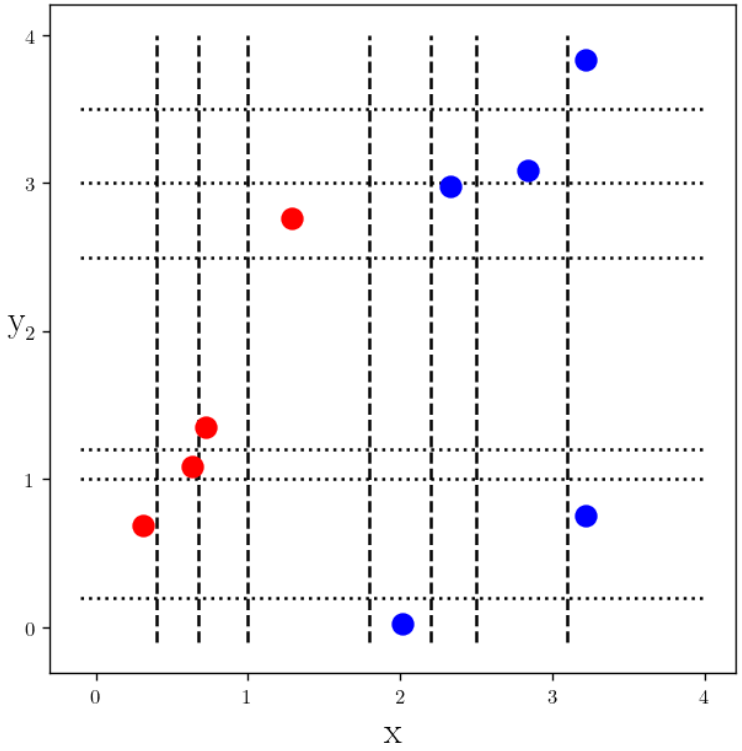
Possible Division (classification) based on **each** feature

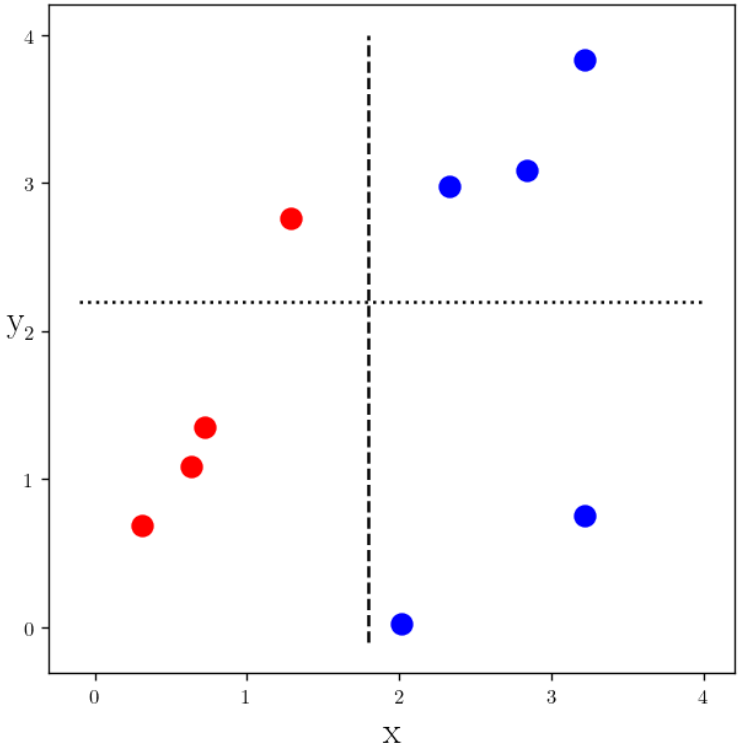




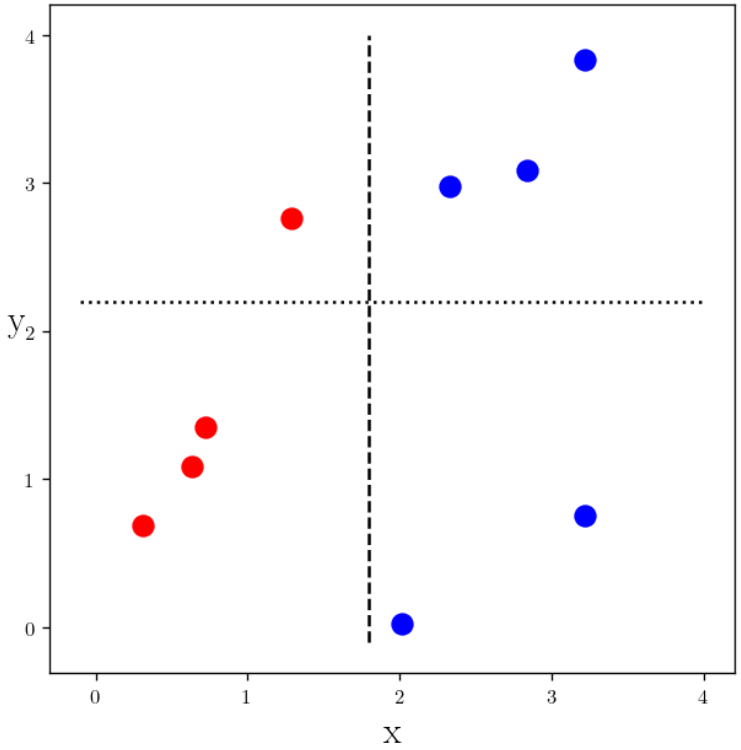
Decision Tree:

Best division based on each feature

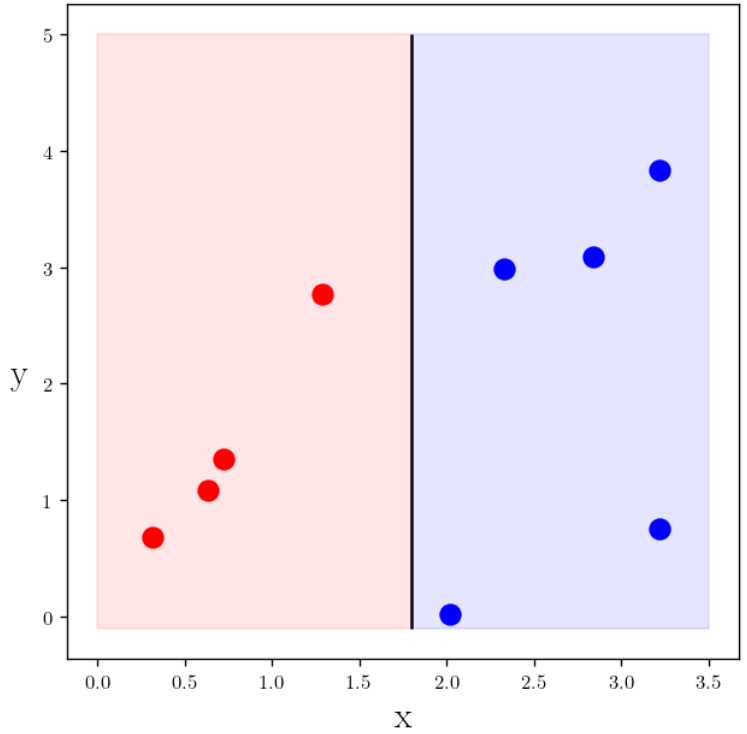




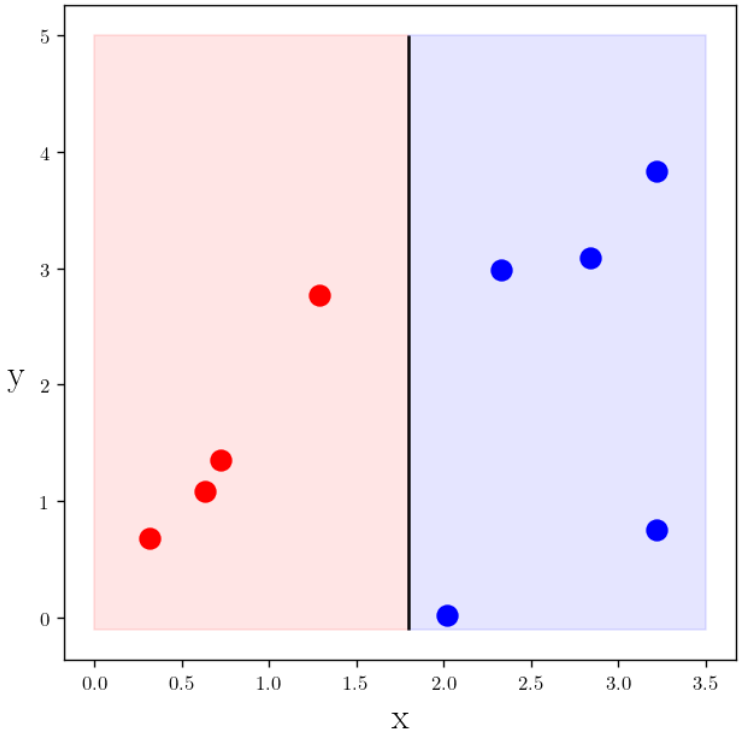
Decision Tree:



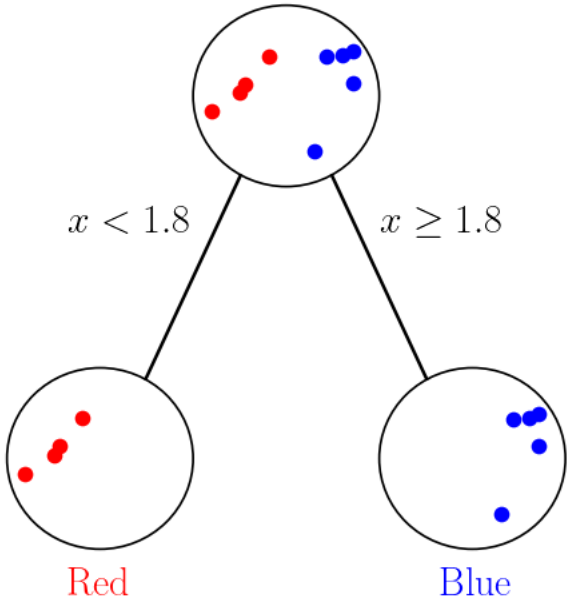
Best division based on the **most important feature**



Decision Tree:

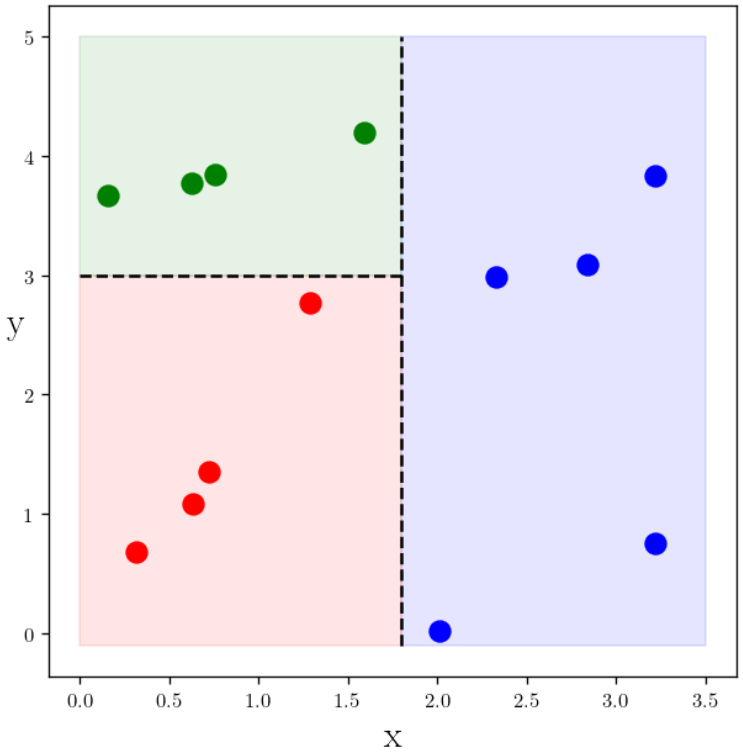


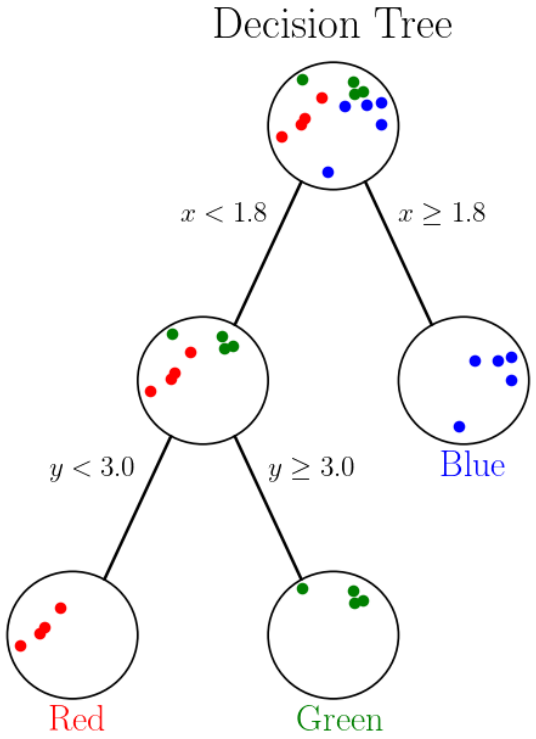
Tree Structure



Decision Tree:

Three Classes instead of two



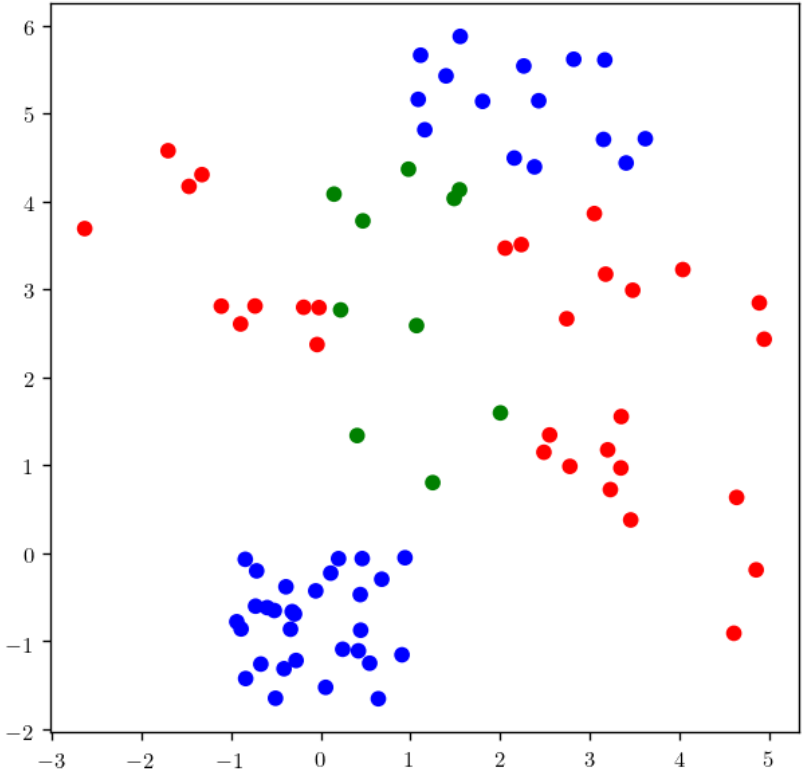


Example:

Task:
Classifying using:

1. Gini Index
2. Information Gain (entropy)

With different depths for the tree.



Measuring Information

The measure of information of a set is known as entropy

Entropy is defined as the expected value of the information. First we need to define information. If you are classifying something that could take multiple values, the information for symbol x_i is defined as

$I(x_i)=\log_2 p(x_i)$

where $p(x_i)$ is the probability of choosing this class.

To calculate entropy, you need the expected value of all the information of all possible values of our class.

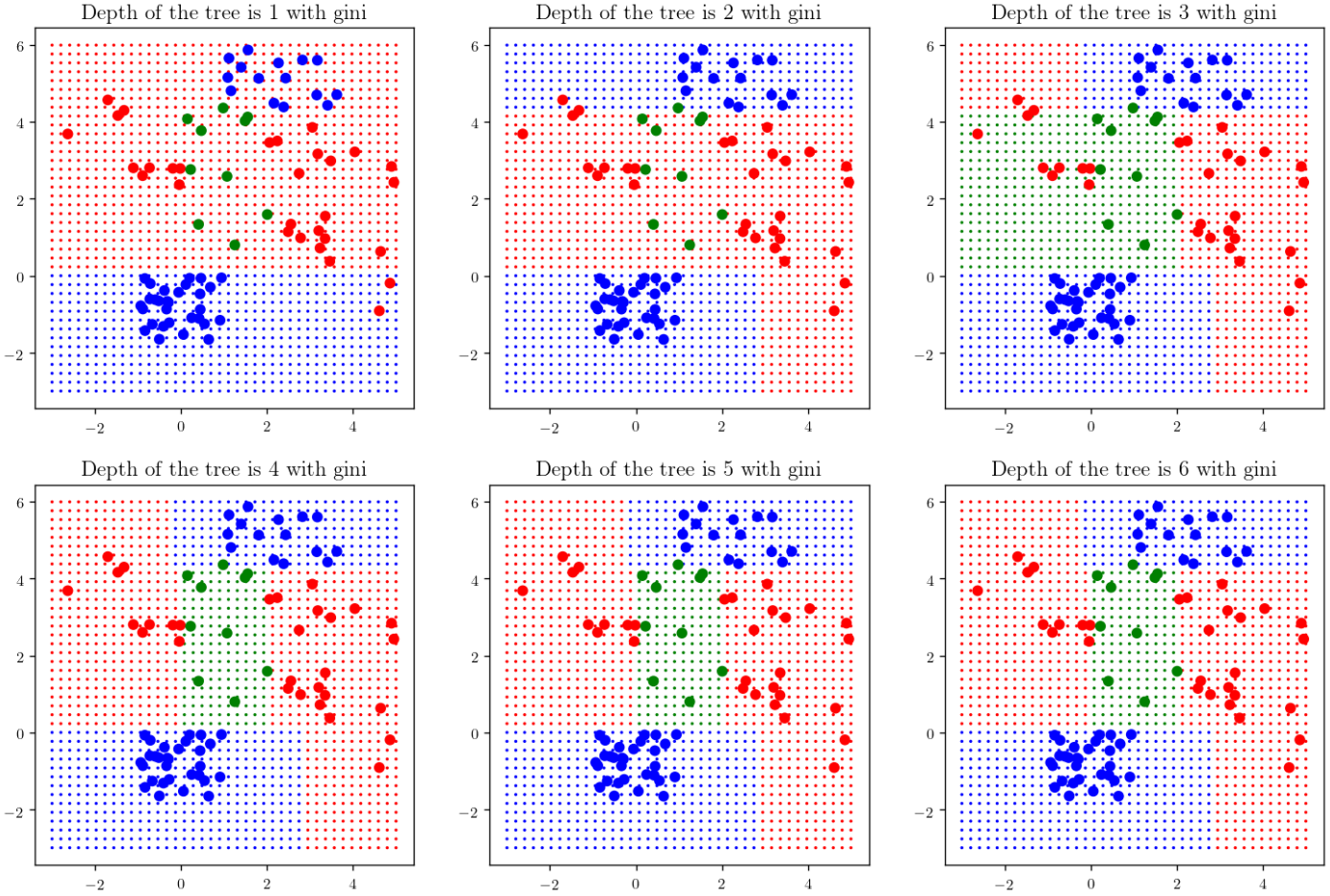
Entropy

The entropy is given by

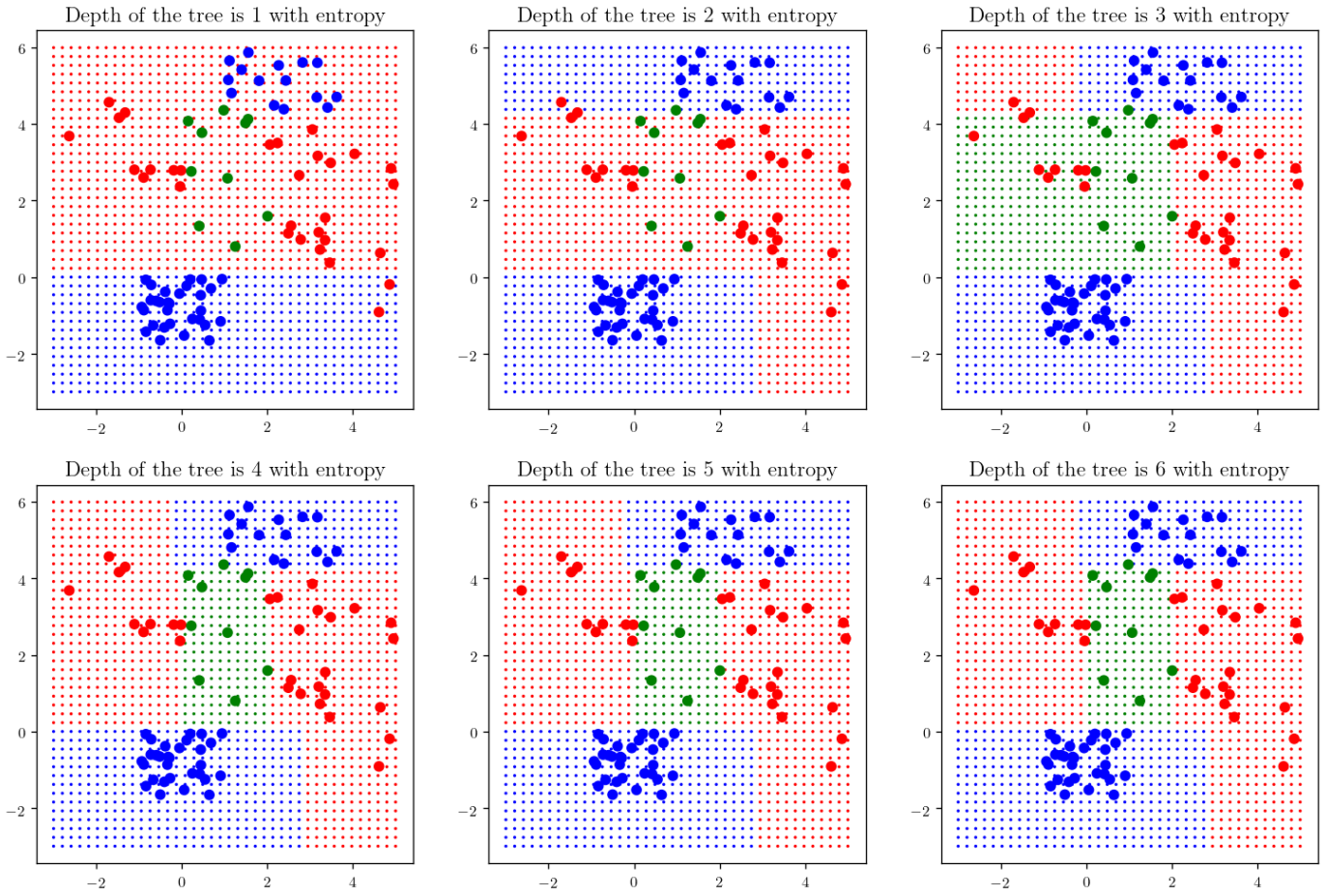
$$H = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

where n is the number of classes. This measures the amount of disorder in a dataset. The higher the entropy, the more mixed up the data is.

Gini Index


Depth of the tree is 1 with gini
Depth of the tree is 2 with gini
Depth of the tree is 3 with gini
Depth of the tree is 4 with gini
Depth of the tree is 5 with gini
Depth of the tree is 6 with gini

Information Gain



Random Forest of Decision Trees:

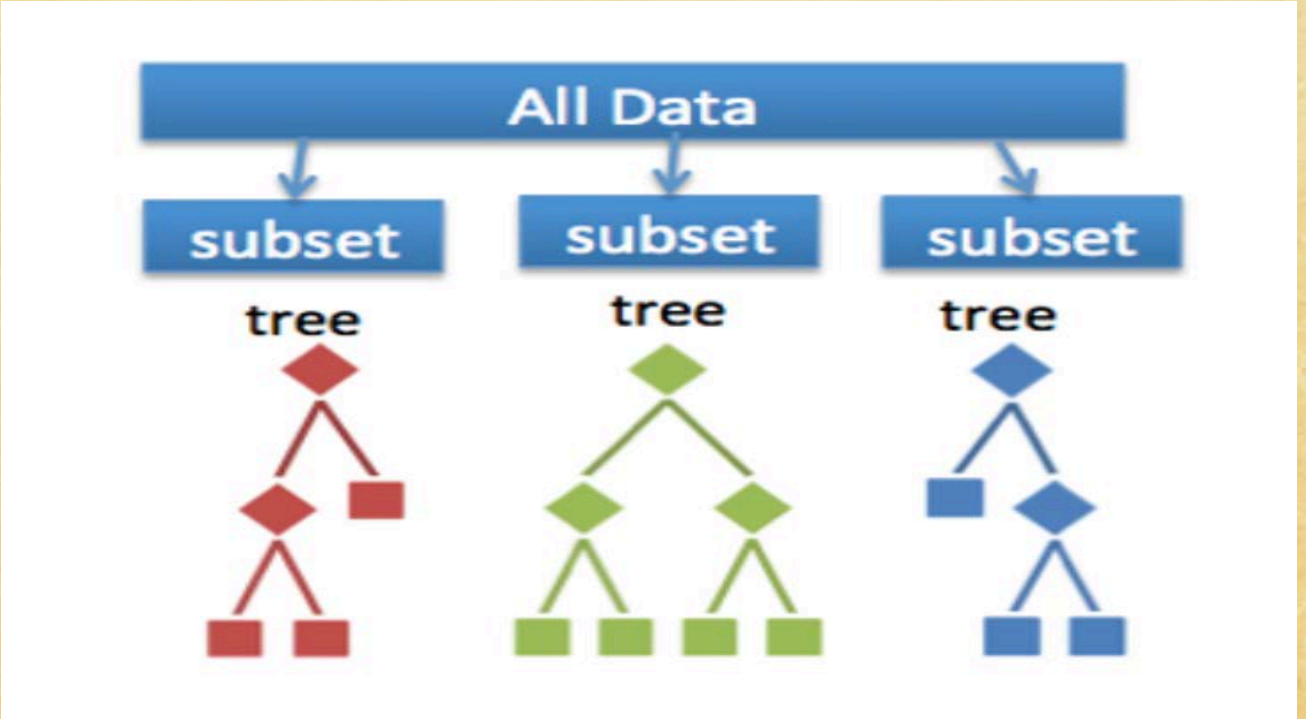
In order to reduce the possible overfitting of a decision tree, we build several decision trees and then take a vote. (An Ensemble Learning method)

We can use any or both of these methods:

1. Randomly select a subset from the features and build a decision tree.
2. Randomly select **N** examples from the **data set with N examples** with replacement and build a decision tree.

In simple words, Random forest builds multiple decision trees (called the forest) and glues them together to get a more accurate and stable prediction. The forest it builds is a collection of Decision Trees, trained with the bagging method.

Random Forest



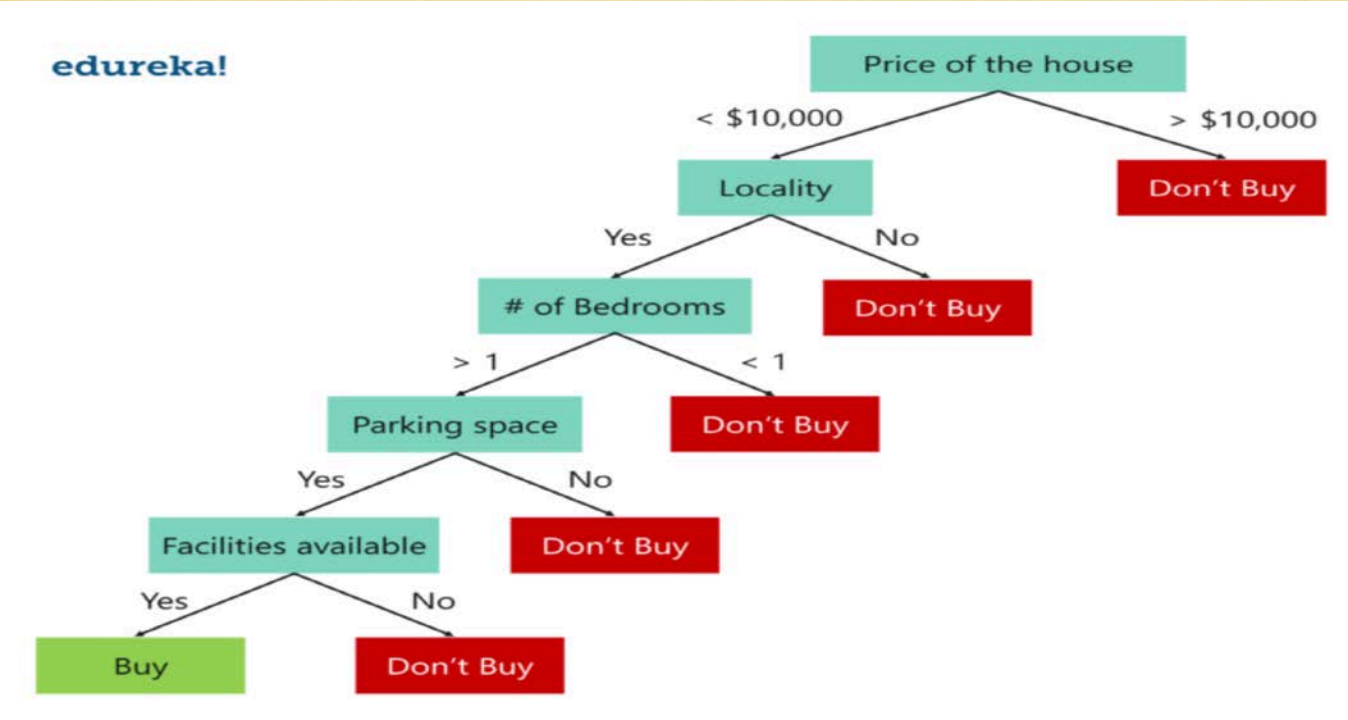
The Difference between Decision Tree and Random Forest

Example: Suppose you are planing to buy a house. The range of important parameters you consider for buying the house are:
Price of the house
Locality
Number of bedrooms,
Parking space
Available facilities
These parameters are called predictor variables, which are used to find the response variables. Using these parameters, we could build a decision tree.
Decision trees are built on the entire parameters space.
Random forest is an ensemble of decision trees. It randomly selects a set of parameters and creates a decision tree for each set of chosen parameters

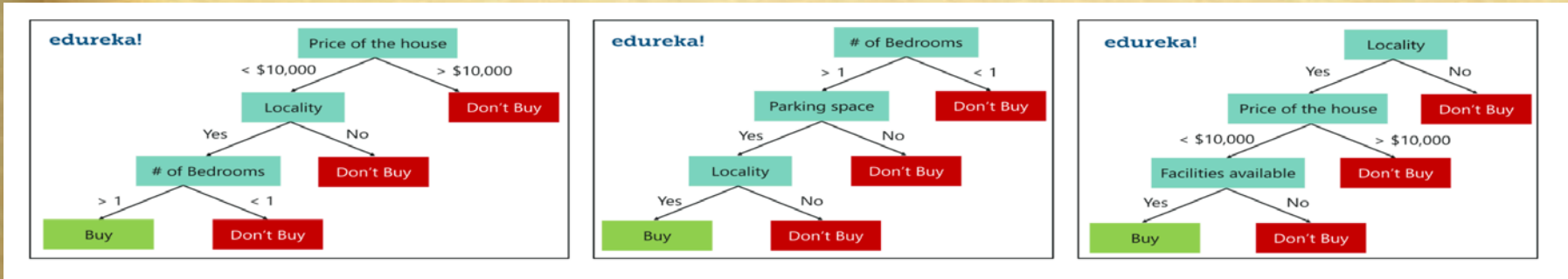
Example of a Decision Tree when using all the available Data
(From edureka webpage)

Random forest is an ensemble of decision trees. It randomly selects a set of parameters and creates a decision tree of each set of chosen parameters.
Next figure shows a set of three decision trees with each tree taking only three parameters from the entire dataset. Each tree predicts the outcome based on the respective predictor variables used in that tree and takes the average of the results from all the decision trees in the random forest.
After creating multiple Decision trees using this method, each tree selects or votes the class (in this case the decision trees will choose whether or not a house is bought), and the class receiving the most votes by a simple majority is termed as the predicted class.

Decision tree using all the parameters

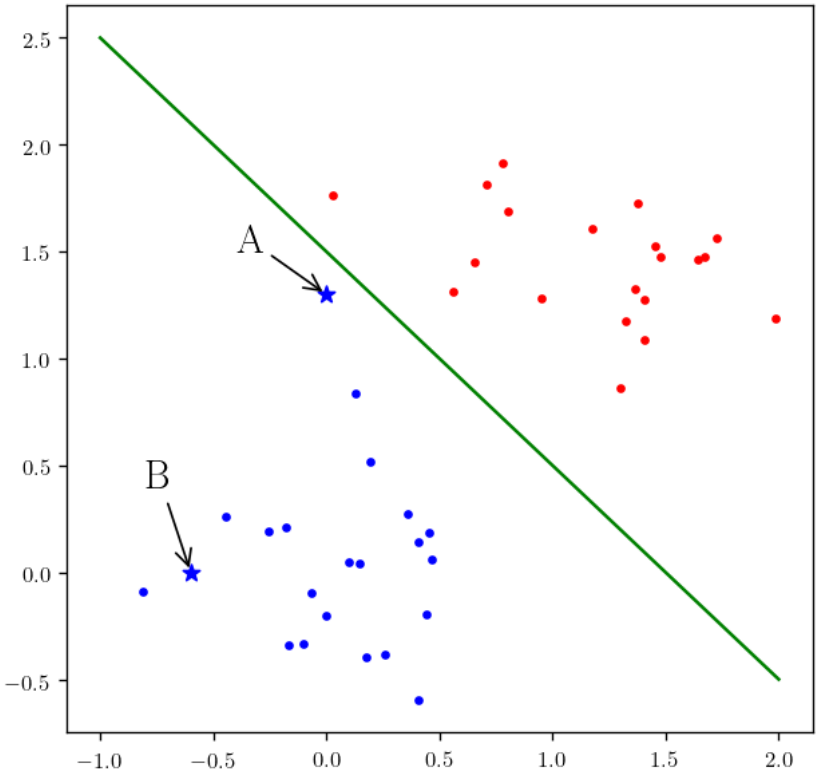


Three Decision Trees based on sub-sets of the parameters



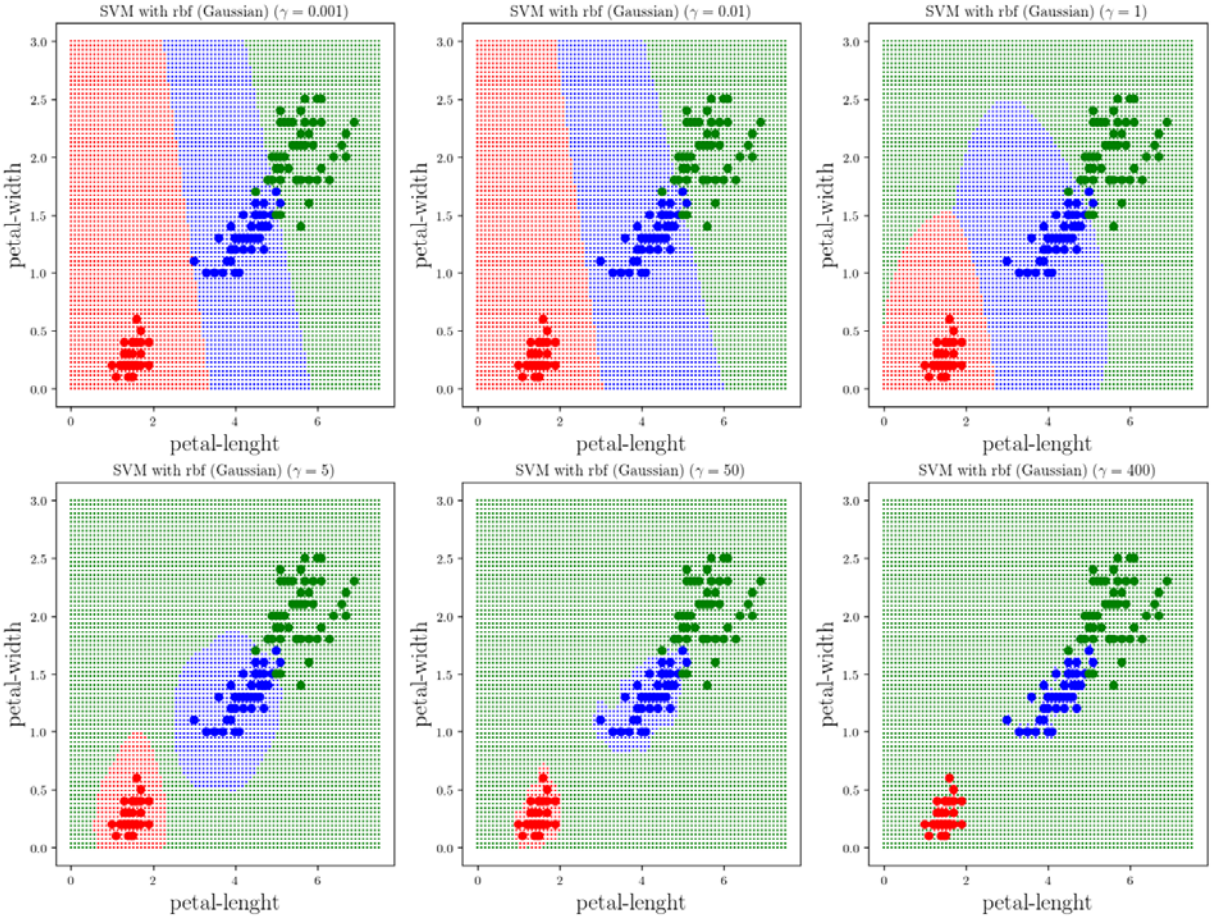
Support Vector Machines:

For the point **A** you can see we are very close to the decision boundary, which means that by changing the boundary we change the classification of point **A** as blue to red. But at the same time if you look at the point **B,** we are pretty certain in classifying the point as blue since it is not in the vicinity of the decision boundary or in a more general case of multi-dimension our **separating hyperplane**. So the task is to find a decision boundary that makes our confidence in our classification better.
(We can change this linear classifier to a non-linear classifier using polynomial, Gaussian and other **kernels**.)



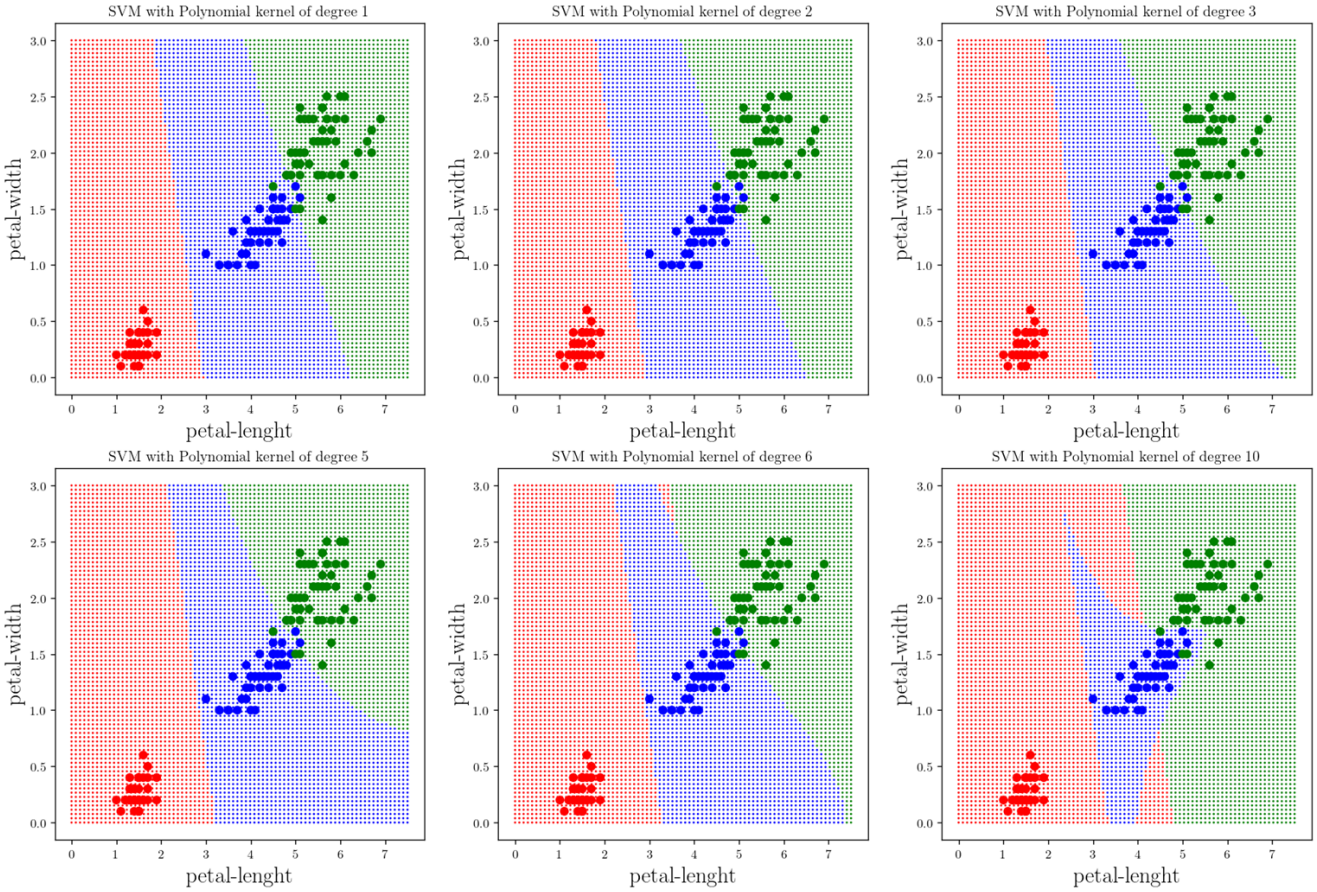
Example: Iris flowers again

Gaussian Kernel with Different variances



Example: Iris flowers again

Polynomial Kernel with Different degrees



Bayes Classifier

Introduction

The KNN and decision tree classifiers provide definite answers as to whether the data belongs to certain class or not. Their classification could be right or wrong. There are some classifiers that provide a best guess or assign a probability to a dataset to be in a class. Indeed the probability theory forms the basis of many machine learning algorithms. Here we look at the ways the probability theory could be used to classify things. Naïve Bayes classifier is such a technique. It is called “naïve” because its formulation makes some naïve assumptions.

The Baysian learning method calculates explicit probabilities for hypothesis and selects the hypothesis with higher probability. Figure 1 shows datasets with two classes of data. We have a measure of the probability of a new data point (x,y) belonging to class 1, which we call it $p_1(x,y)$, and a probability for the data point belonging to class 2, which we call it $p_2(x,y)$. To classify the data point we use the following rules:

If $p_1(x,y) > p_2(x,y)$ the class is 1

If $p_1(x'y) < p_2(x'y)$ the class is 2.

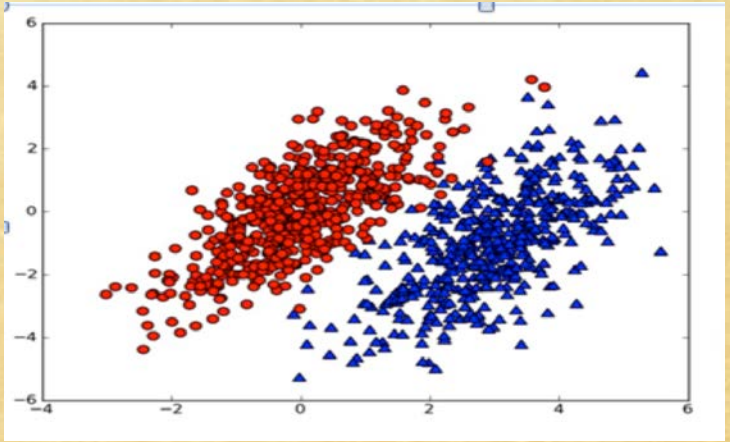
Simply, we choose the class with higher probability to be the class of the data point. This is Bayesian decision theory- choosing the decision with highest probability.

Classification

Probability Distributions

Figure 1 shows two probability distributions with known parameters describing the distributions

Figure 1



The Bayesian Method

The premise of the Bayesian method is that probability statements are not limited to data but can be made for models themselves. Inferences are made by producing Probability Density Functions (PDFs). Model parameters are treated as random variables. Bayesian methods give optimal results given all the available information.

Difference between Classical and Bayesian Approaches

The classical and Bayesian techniques are both concerned with the data likelihood function. In classical statistics the data likelihood function is used to find model parameters that yield the highest data likelihood. The data likelihood cannot be interpreted as a probability density function for model parameters. However, the Bayesian method extends the concept of data likelihood function by adding extra prior information to the analysis and assigning PDFs to all model parameters and models themselves.

The Bayesian method is able to provide a full probabilistic framework for data analysis.

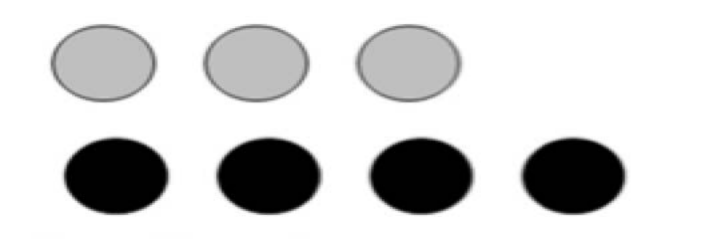
Example # 1: Conditional Probability

Let's assume we have a jar containing seven stones. Three of these stones are grey and four are black. The chance of randomly selecting a grey stone is *p(grey)=3/7* while selecting a black stone is *p(black)= 4/7*. Now, we divide these into two buckets. What is the probability of drawing a grey stone from bucket B? This is known as conditional probability. We are calculating the probability of a grey stone, given that the unknown stone is coming from bucket B. We can write this as *P(grey |bucket B)*, which is read as "the probability of grey given bucket B". It is easy to find *p(grey | bucket A)= 2/4* and *p(grey | bucket B)= 1/3.* In other words, we can say

$$p(grey|bucket\ B) = \frac{p(grey\ and\ bucket\ B)}{p(bucket\ B)}$$

Example

Collection of seven stones that are grey and black. If we randomly select a stone from this set, the probability that it will be grey or black stone is 3/7 and 4/7 respectively



Seven stones divided in two buckets

Expression of Bayes Probability

Which is calculated as

$$p(grey\ and\ bucket\ B) = \frac{1}{7}\ ;\ p(bucket\ B) = \frac{3}{7}$$

Therefore

$$p(grey|bucket\ B) = \frac{1/7}{3/7} = \frac{1}{3}$$

Bayes rule tells us how to swap the symbols in a conditional probability statement. If we have *p(x|c)* but want to have *p(c|x)*, we use the following formula

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)}$$

which is just another way of interpreting the relation we already came to.

Principle Component Analysis

Definition

Principle Component Analysis (PCA) is a technique to study the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualised as a set of coordinates in a high dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection of this object when viewed from its most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

PCA can be thought of as fitting an n-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a small amount of information (Figure 1).

Example of the principle component in the direction of maximum variance and the second component orthogonal to it (Fig 1). (from *Machine Learning in Action" by Peter Harrington)*

Fig 1: Finding the component with maximum variance

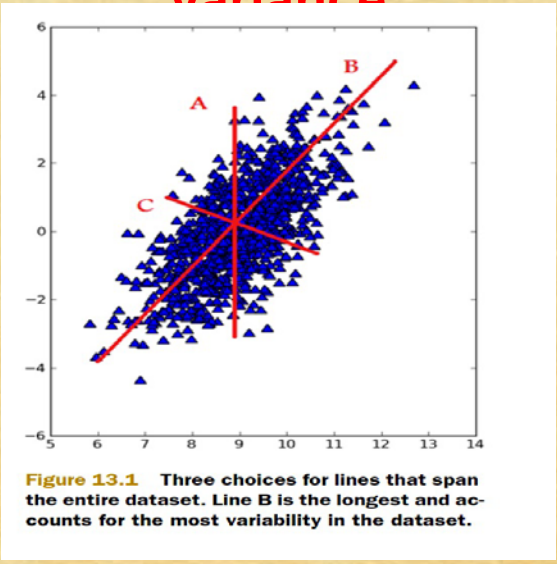


Figure 13.1 Three choices for lines that span the entire dataset. Line B is the longest and accounts for the most variability in the dataset.

Example of results after applying the PCA and reducing dimensionality (Fig 2) (from *Machine Learning in Action" by Peter Harrington)*

Fig 2: Reduction from 2 to one dimension

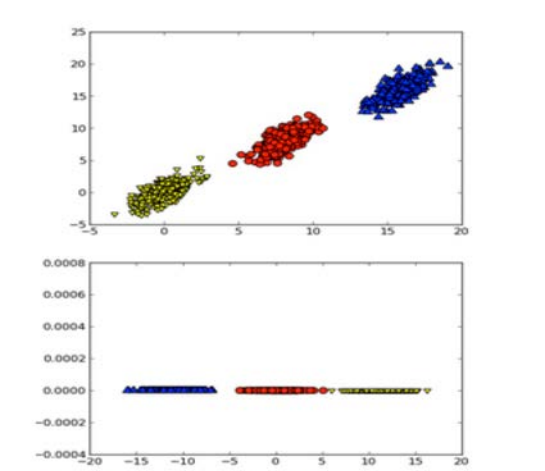


Figure 13.2 Three classes in two dimensions. When the PCA is applied to this dataset, we can throw out one dimension, and the classification problem becomes easier.

To find the axes of the ellipsoid, we must first subtract the mean of each variable from the dataset to center the data around the origin. Then, we compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix. Then we must normalize each of the orthogonal eigenvectors to become unit vectors. Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data. This choice of basis will transform our covariance matrix into a diagonal form with the diagonal elements representing the variance of each axis. The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues (Figure 2).

PCA is defined as an orthogonal linear transformation that transforms data to a