Crafted by :

Rizki Fajar Nugroho

Machine Learning II

Support Vector Machine Algorithm
Nearest Neighbours Algorithm
Random Forest Algorithm

Updated

Q2.2020

Hi, I'm Rizki Fajar Nugroho

Experiences:

- **Data Tech Specialist at IYKRA**
- **Machine Learning Engineer at Omdena, Singapore**
- **Graduated from Electrical and Electronic Engineering**

Objectives

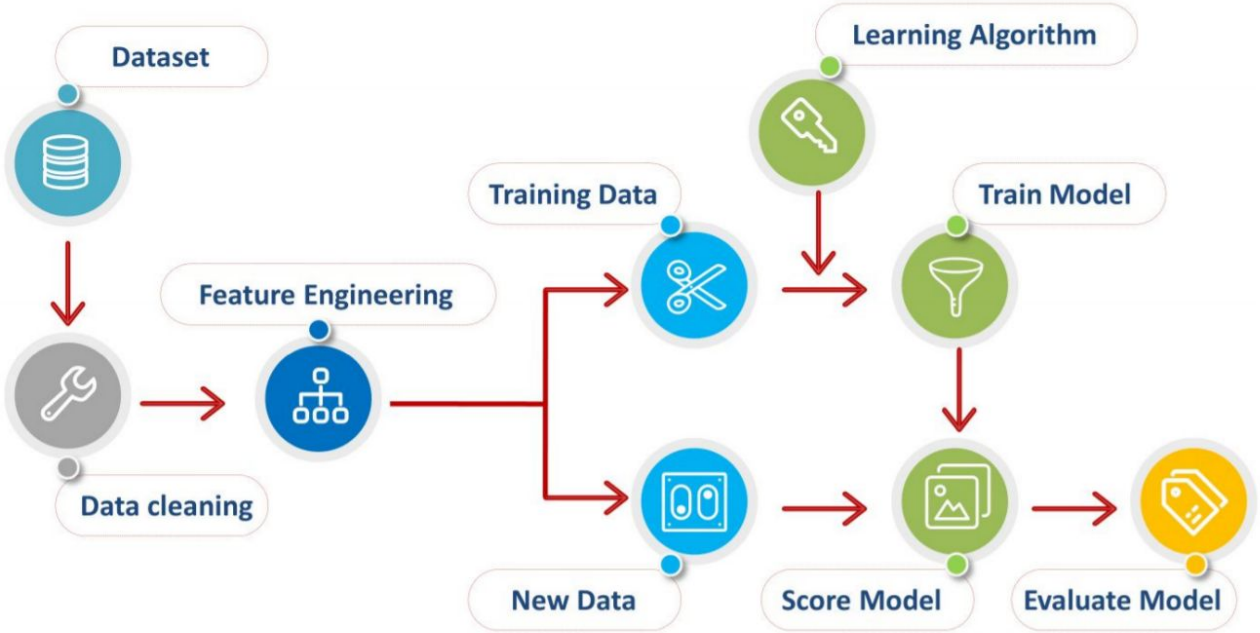
- To getting to know about general information of machine learning pipeline
- To learn the intuition and some of the concepts in the K-Nearest Neighbour, Support Vector Machine, and Random Forest

Outline

1. Machine Learning Introduction
2. K-Nearest Neighbours (Classification)
3. Support Vector Machines (Classification)
4. Random Forest (Classification)

Machine Learning Introduction

General Steps



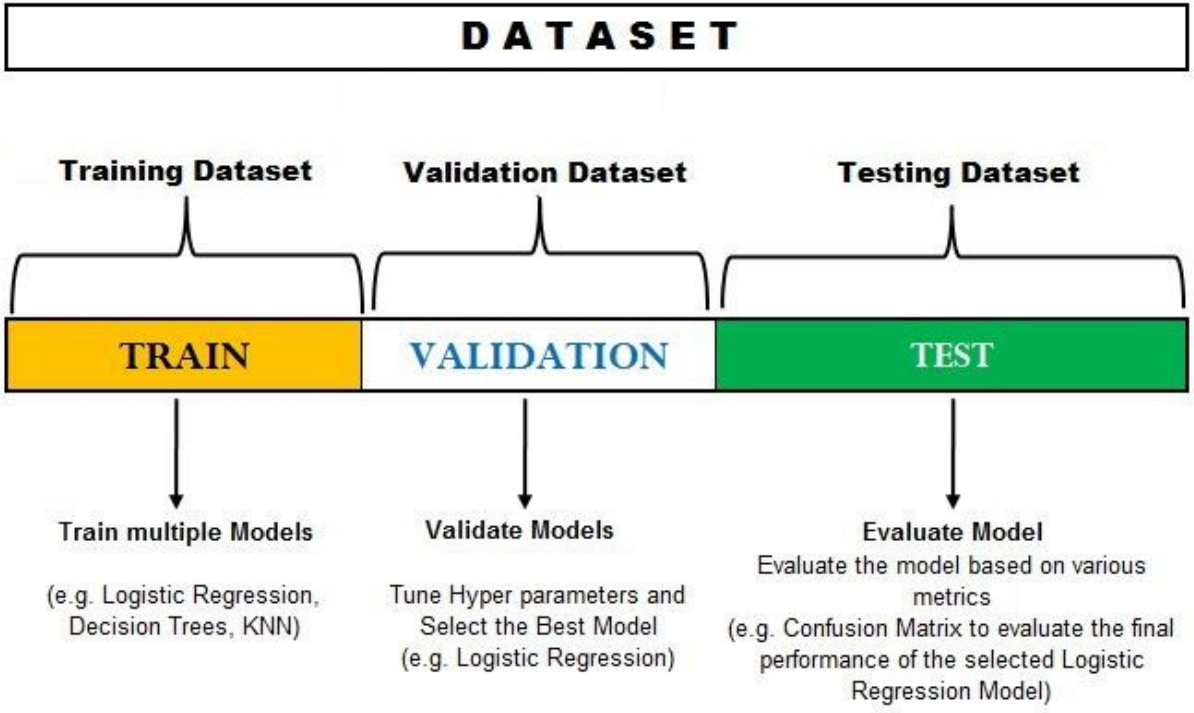
Data Cleansing

1. Duplicate Dataset
2. Missing Data
3. Outliers
4. Data Type

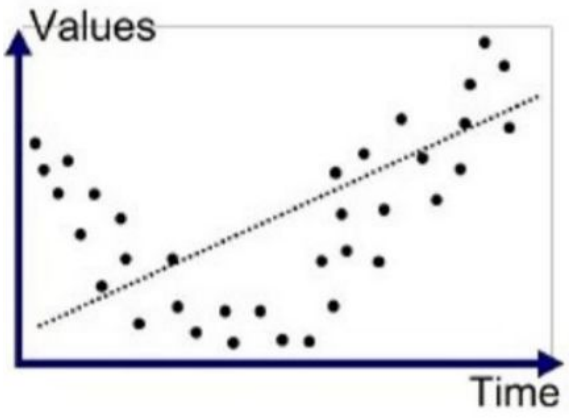
Feature Engineering

1. Create a New Variable
2. Change all variable to numerical
3. Scaling variable (Depends on the algorithms)
4. Variable reduction
5. Inbalance target class

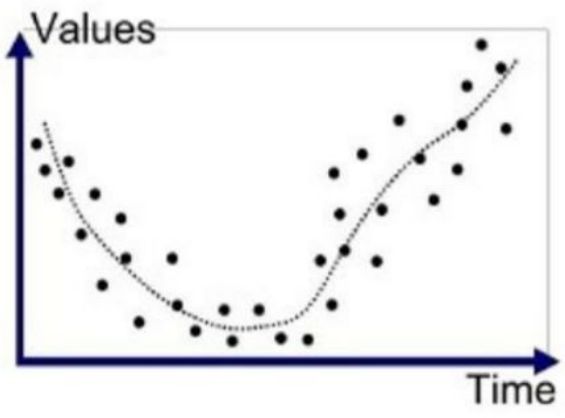
Training dan Testing Dataset



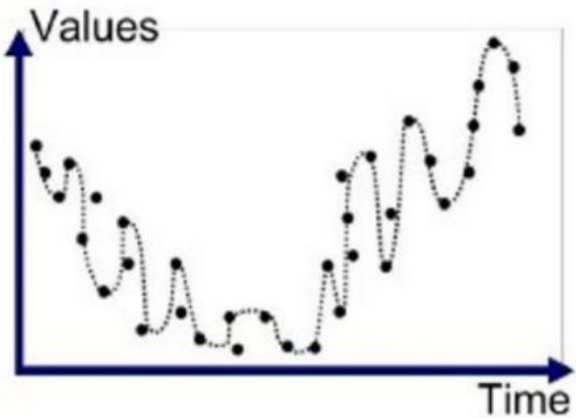
General Problems in Machine Learning Model



Underfitted

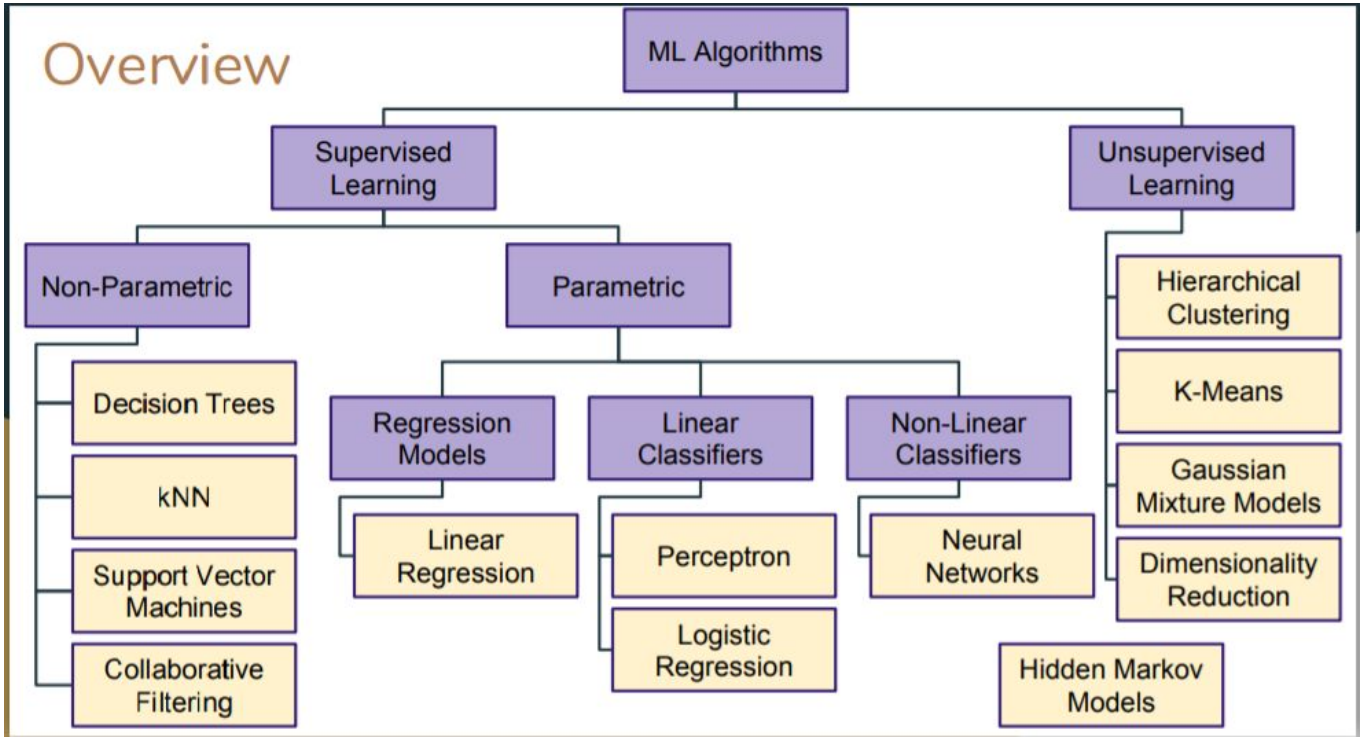


Good Fit/Robust



Overfitted

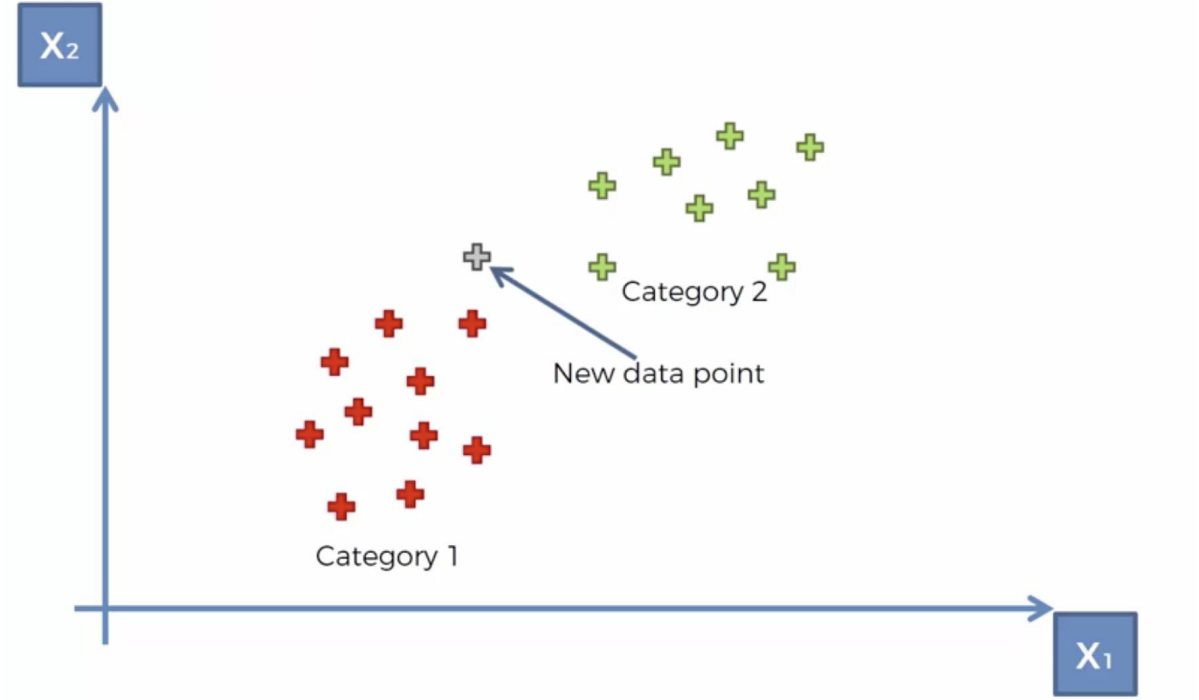
Machine Learning Algorithms



K-Nearest Neighbours (Classification)

Proprietary Document of IYKRA, 2020

K-Nearest Neighbour Introduction

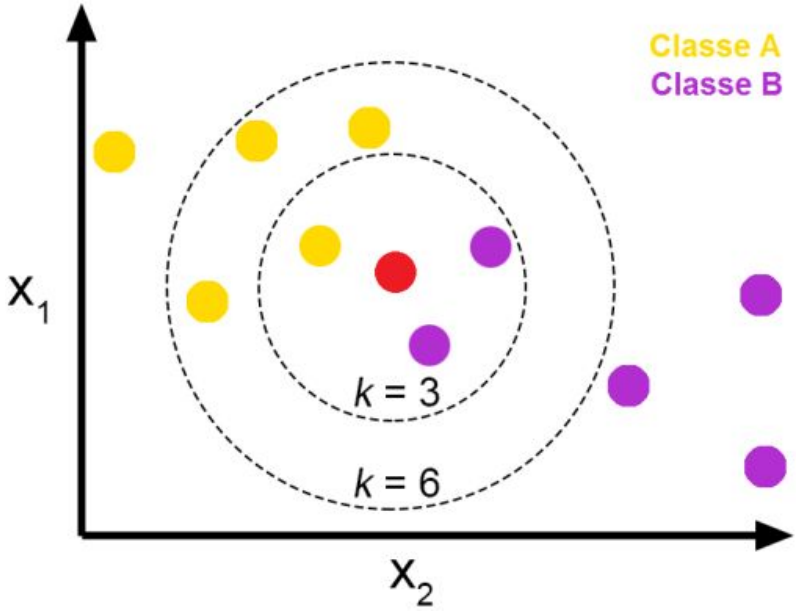


K-Nearest Neighbour Introduction

K-NN is a **non-parametric** and **lazy learning algorithm**.

- Non-parametric means there is no assumption for underlying data distribution i.e. the model structure determined from the dataset.
- It is called Lazy algorithm because it does not need any training data points for model generation.
- All training data is used in the testing phase which makes training faster and testing phase slower and costlier.
- K-Nearest Neighbor (K-NN) is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure.

K-Nearest Neighbour Introduction



Distance Measure in K-NN

- To determine which of the K instances in the training dataset are most similar to a new input, **a distance measure is used**. For real-valued input variables, the most popular distance measure is the Euclidean distance.
- While Euclidean distance is useful in low dimensions, it **doesn't work well in high dimensions and for categorical variables**. The drawback of Euclidean distance is that it **ignores the similarity between attributes**. Each attribute is treated as totally different from all of the attributes.

Distance Measure in K-NN

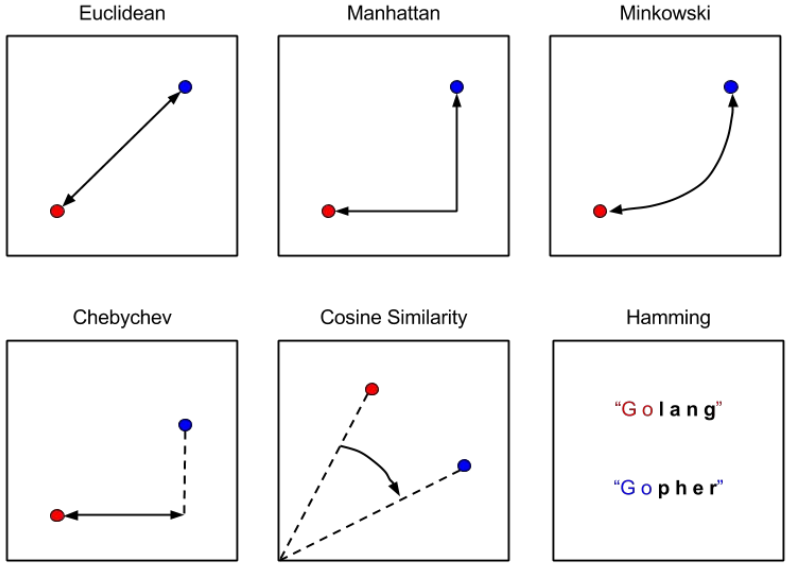
Jarak Eucledian

$$d_E(a,b) = \sqrt{\sum_{i=1}^{D}(a_i - b_i)^2}$$

Jarak Manhattan

$$d_M(a,b) = \sum_{i=1}^{D} |a_i - b_i|^2$$

Distance Measure in K-NN



Due to the squaring in Equation, the Euclidean distance is more sensitive to outliers. **Furthermore, most distance measures are sensitive to the scale of the features**. Data with features that have different scales will bias the distance measures as those predictors with the largest values will contribute most to the distance between two samples

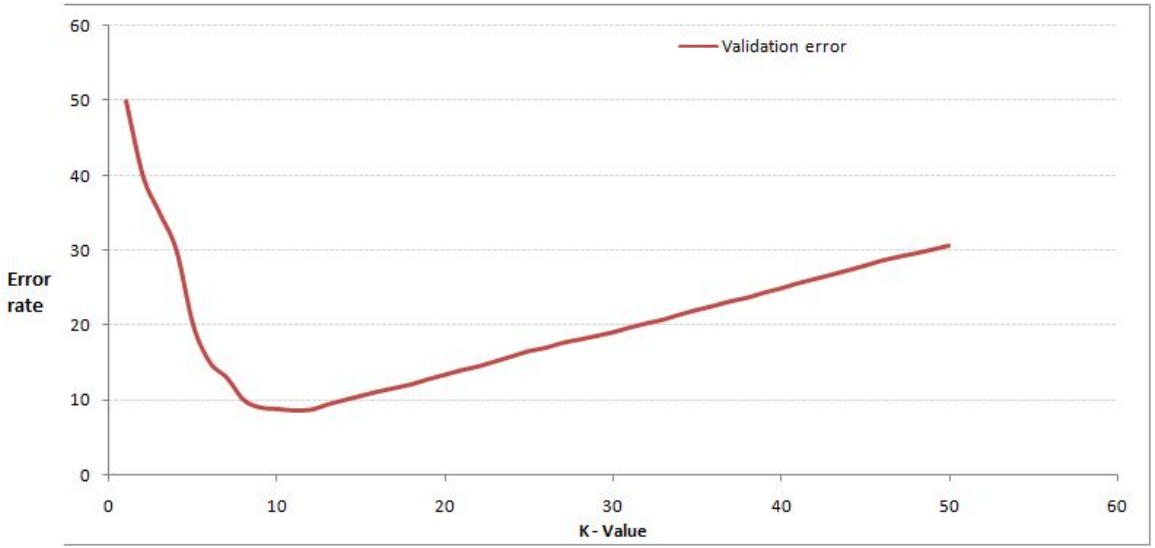
Proprietary Document of IYKRA, 2020

How to choose the best K?

There is no physical way to determine the best ‘K’, so you may have to try out a few values before settling on one value. Do this by pretending the part of training data is “unknown”

- Low values of for K (like 1 or 2) can be noisy and subjected to the effect of outlier
- Large value of K smooth over things, but you don’t want K to be so large that a category with only a few

How to choose the best K?



You can do this process during the cross validation process

Advantages of K-NN

- Simple and intuitive
- It’s good for non linear separable dataset
- Few parameters (K and distance metrics)

Disadvantages of K-NN

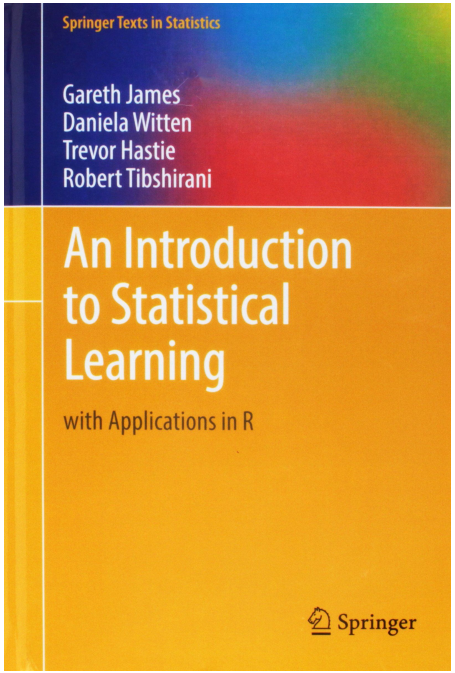
- High prediction cost (worse for large datasets)
- Not good with high dimensional data
- Categorical features doesn't work well
- Prediction time is slow

Pseudocode of K-NN

1. Load the data
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
    2. Sort the calculated distances in ascending order based on distance values
    3. Get top k rows from the sorted array
    4. Get the most frequent class of these rows
    5. Return the predicted class

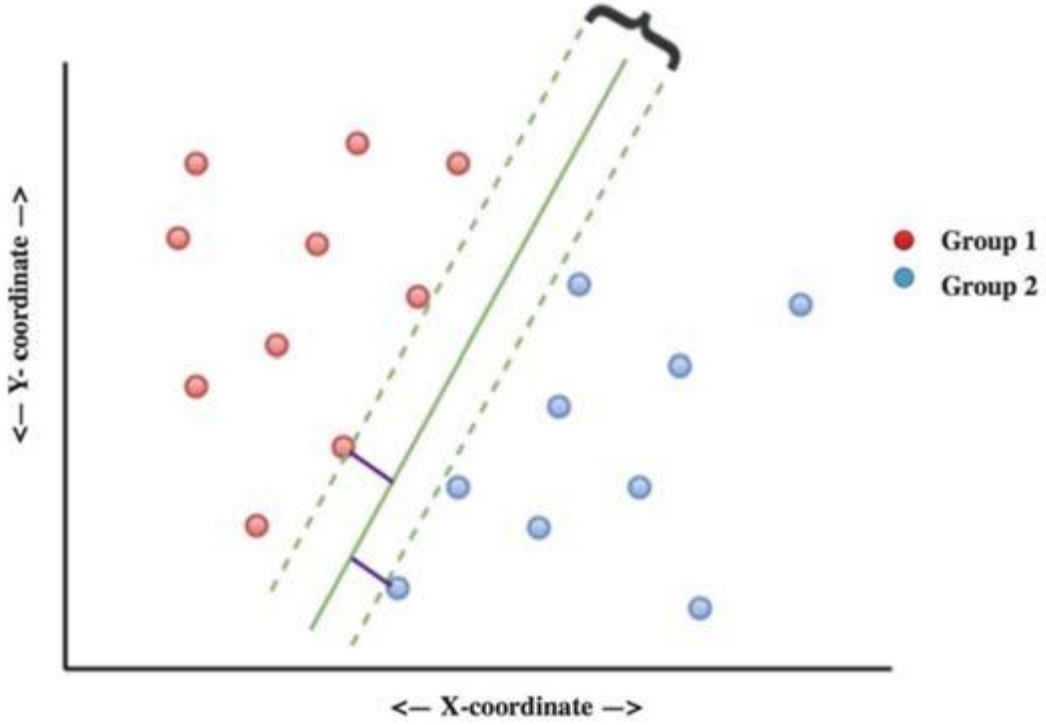
Support Vector Machines (Classification)

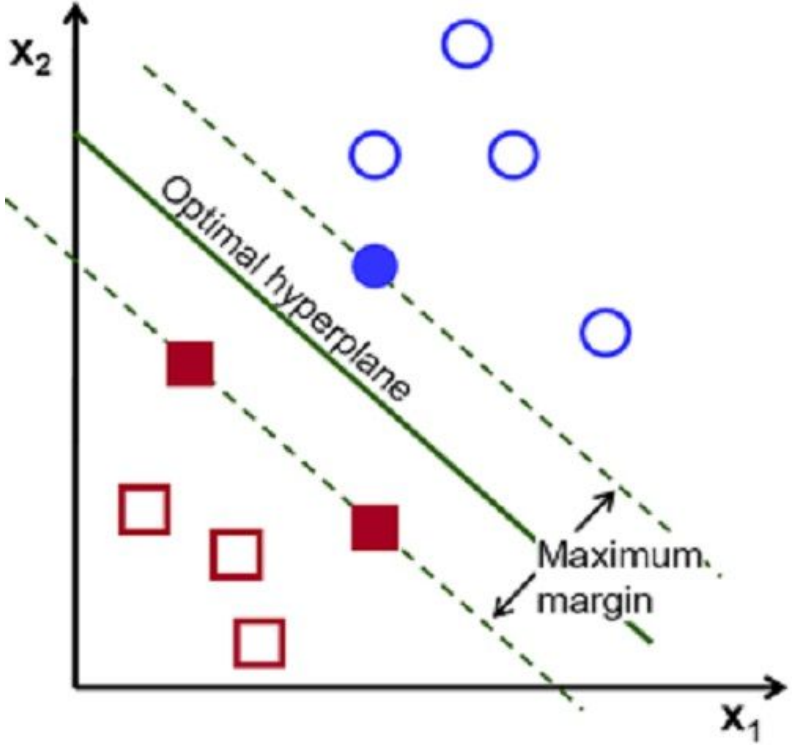
For the greater mathematical explanation

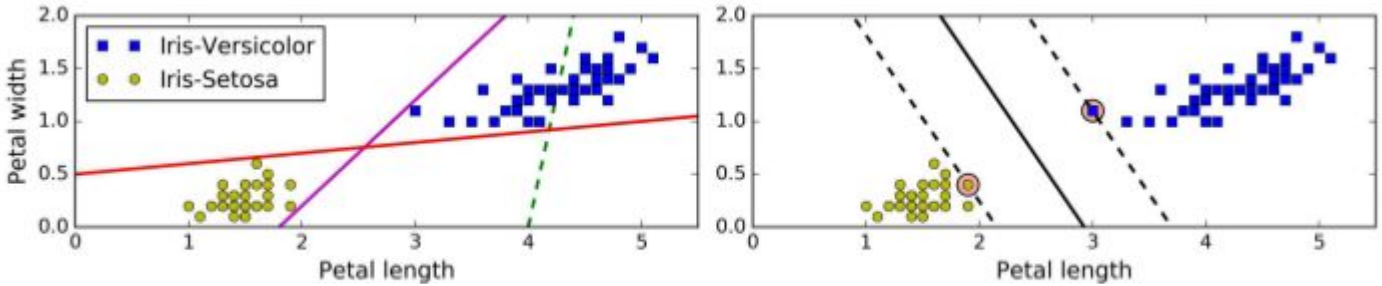


http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf. Introduction to Statistical Learning Page 337 - 356

Hard Margin Classifier

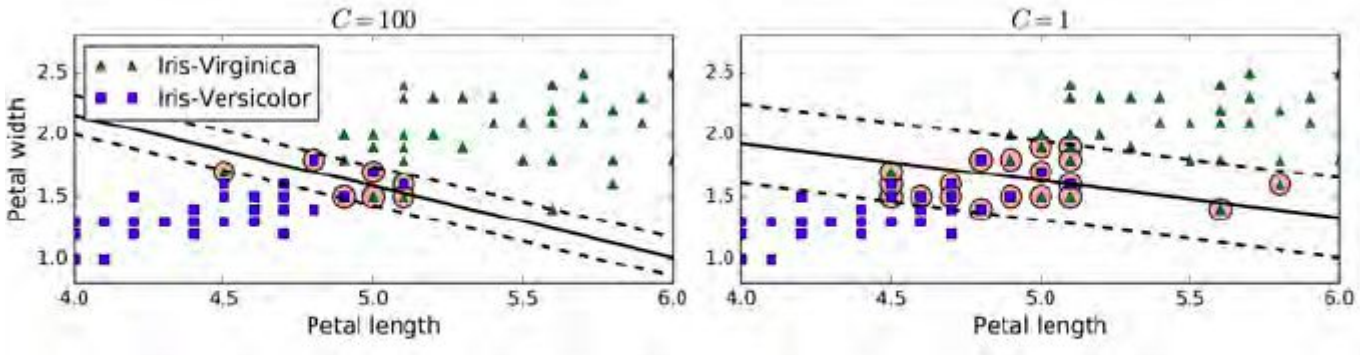






Notice that adding more training instances “off the street” will not affect the decision boundary at all: it is fully determined (or “supported”) by the instances located on the edge of the street. These instances are called the support vectors (they are circled in)

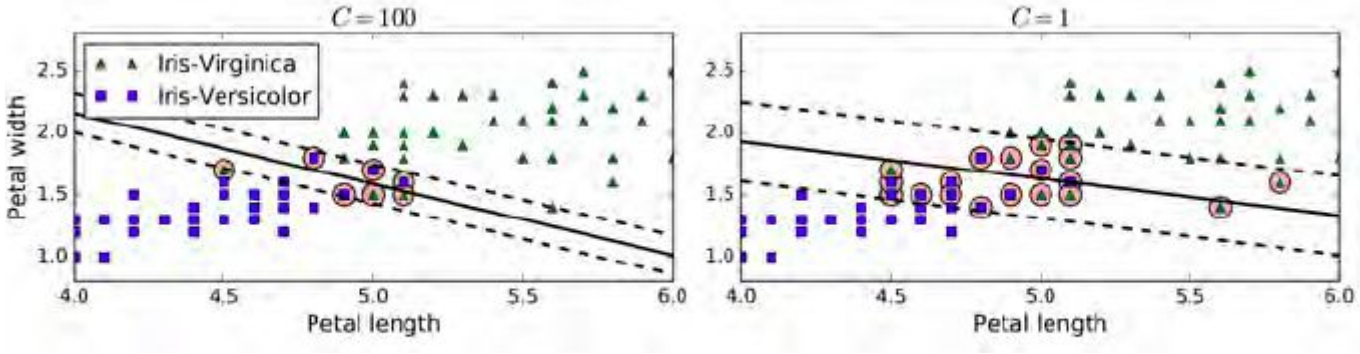
Soft Margin Classification



the **C hyperparameter**

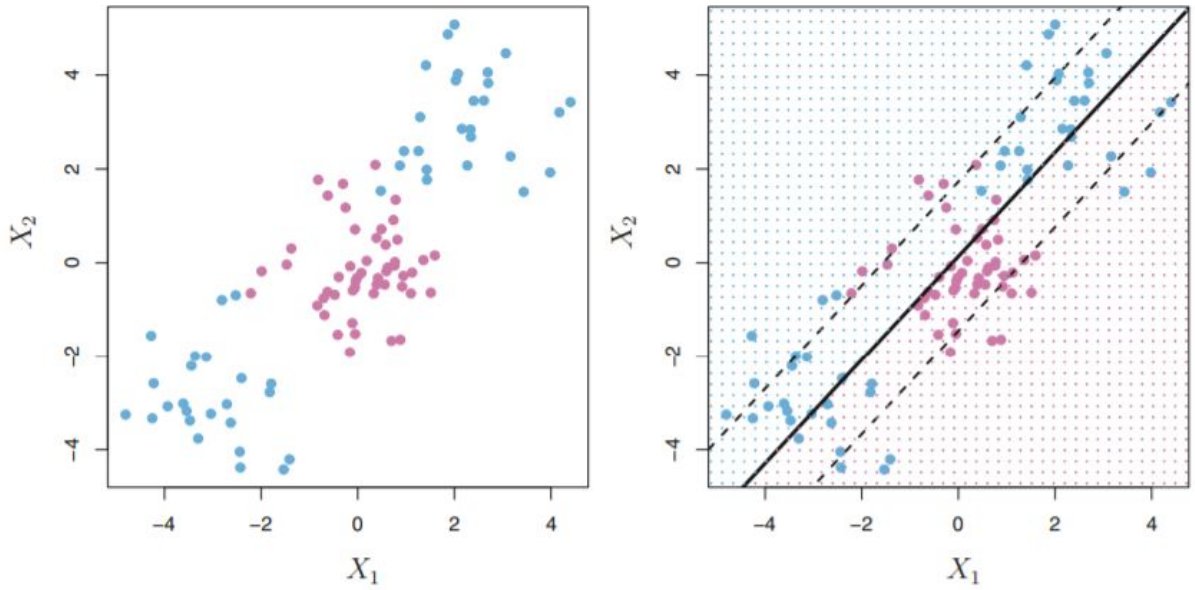
- A smaller C value leads to a wider street but more margin violations
- **Using a high C value** the classifier **makes fewer margin violations** but ends up **with a smaller margin.**
- **Using a low C value** the **margin is much larger**, but **many instances end up on the street**.
- However, **it seems likely that the second classifier will generalize better**: in fact even on this training set it makes fewer prediction errors, since most of the margin violations are actually on the correct side of the decision boundary

Soft Margin Classification



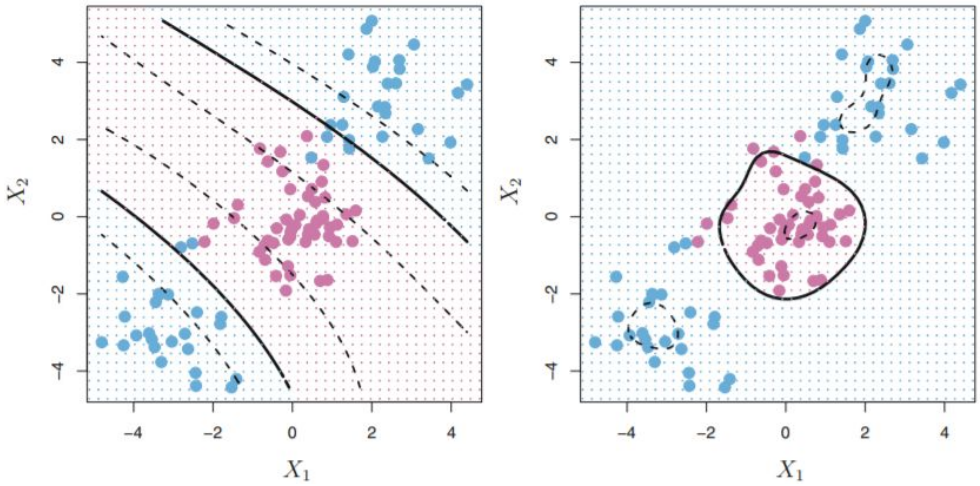
The objective is to **find a good balance between keeping the marginal distance as large as possible and limiting the margin violations** (i.e., instances that end up in the middle of the street or even on the wrong side). This is called **soft margin classification**

Support Vector Machines



Is this a good decision boundary?

Support Vector Machines



The support vector machine (SVM) is an extension of the support vector classifier (soft margin classification) that results from enlarging the feature space in a specific way, using kernels. we may want to enlarge our feature space to accommodate a non-linear boundary between the classes. This is called the **SVM Kernel Trick**

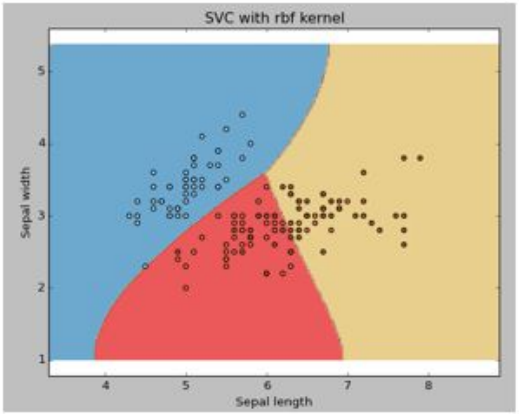
1. Linear
2. Nonlinear
3. polynomial
4. radial basis function (RBF)
5. sigmoid

Check out the details explanation of these kernel tricks at:

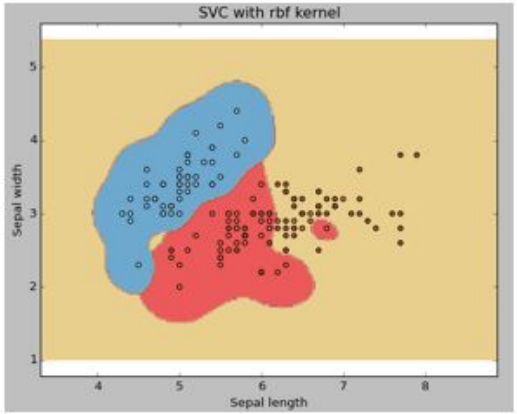
- **The Element of Statistical Learning** (Source)
- **Pattern Recognition and Machine Learning** (Source)

Gamma in the SVM

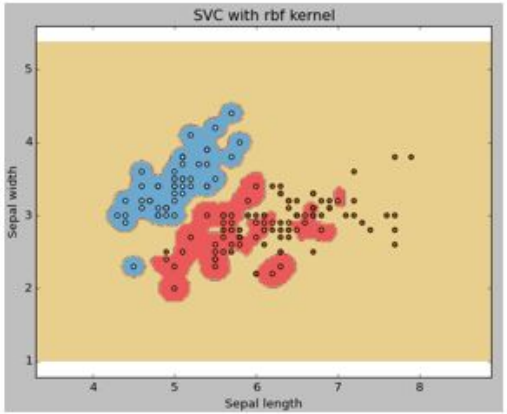
gamma =0



gamma =10



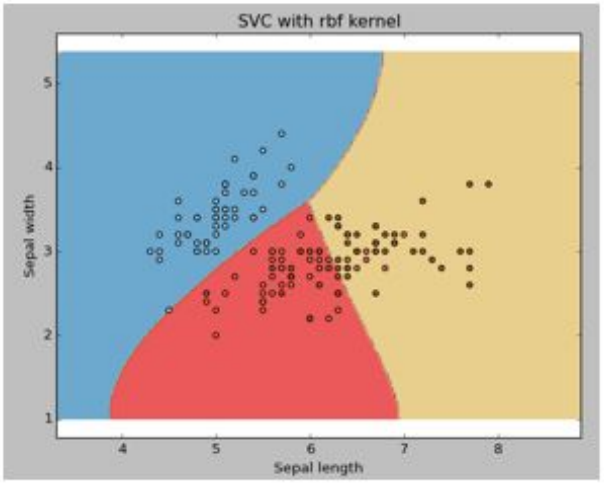
gamma =100



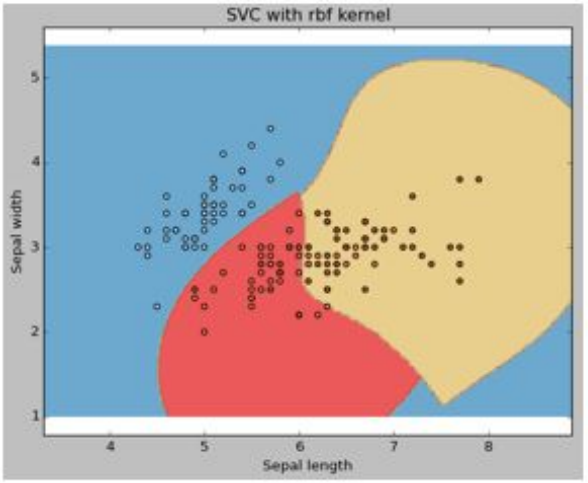
Higher the value of gamma, will try to exact fit the as per training data set i.e. generalization error and cause over-fitting problem

C in the SVM

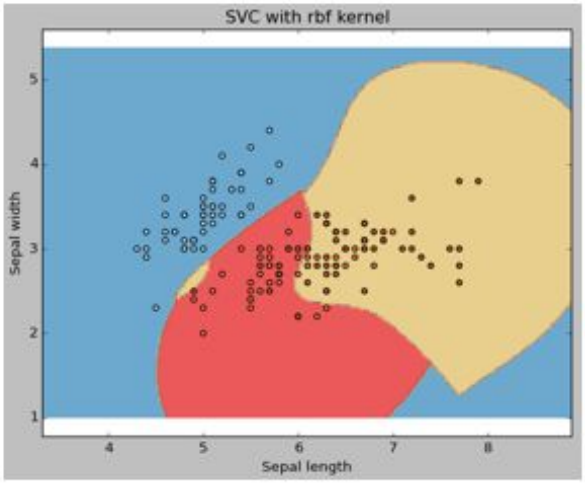
c =1



C =100

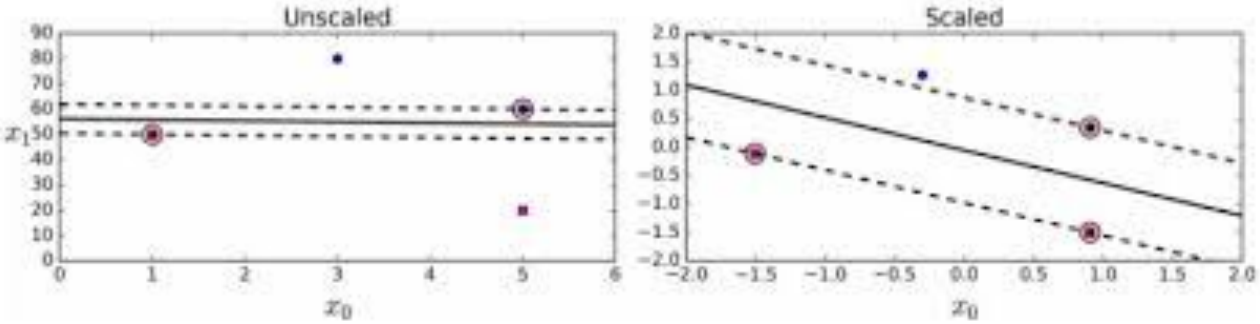


c =1000



Penalty parameter C of the error term. It also controls the trade-off between smooth decision boundaries and classifying the training points correctly

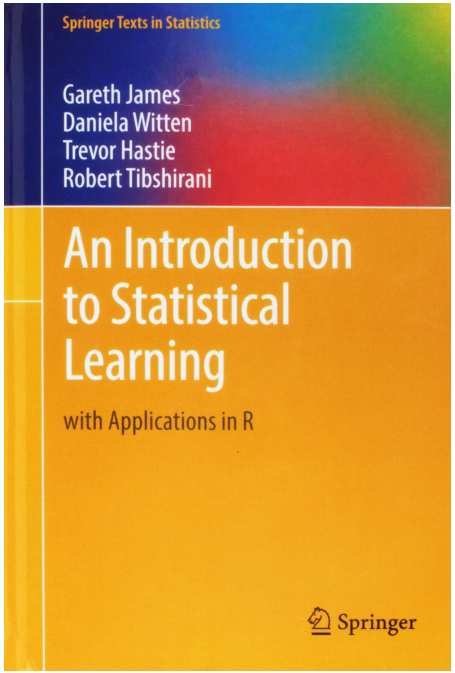
Feature Scaling in the SVM



SVMs are sensitive to the feature scales. The vertical scale is much larger than the horizontal scale, so the widest possible street is close to horizontal. After feature scaling, the decision boundary looks much better (on the right plot).

Random Forest (Classification)

For the greater mathematical explanation

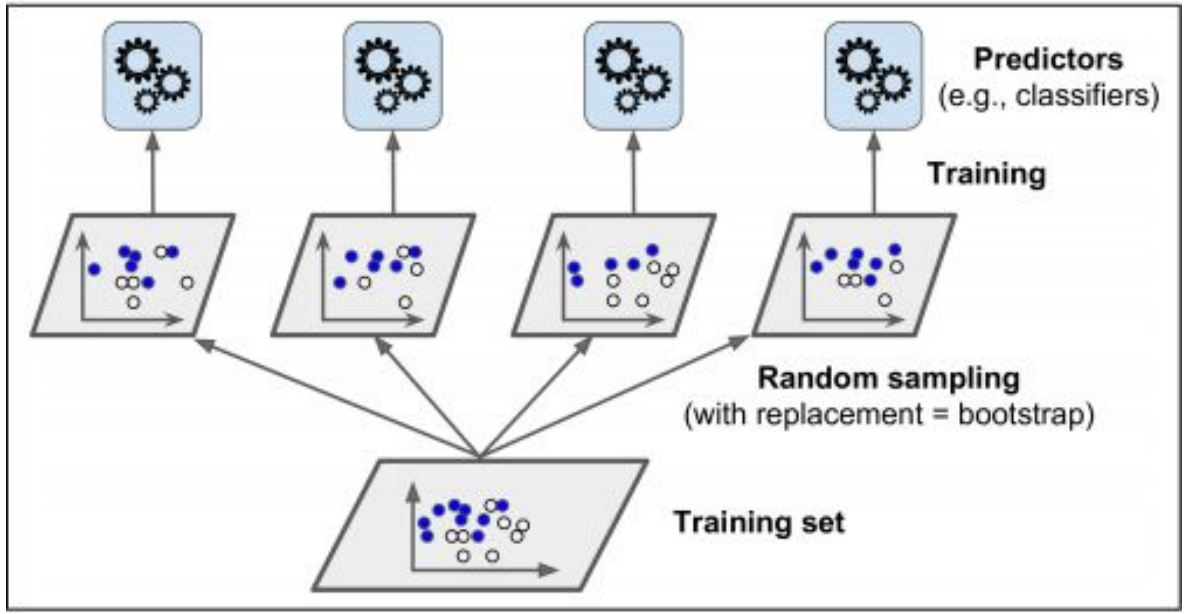


http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf. Introduction to Statistical Learning Page 303 - 321

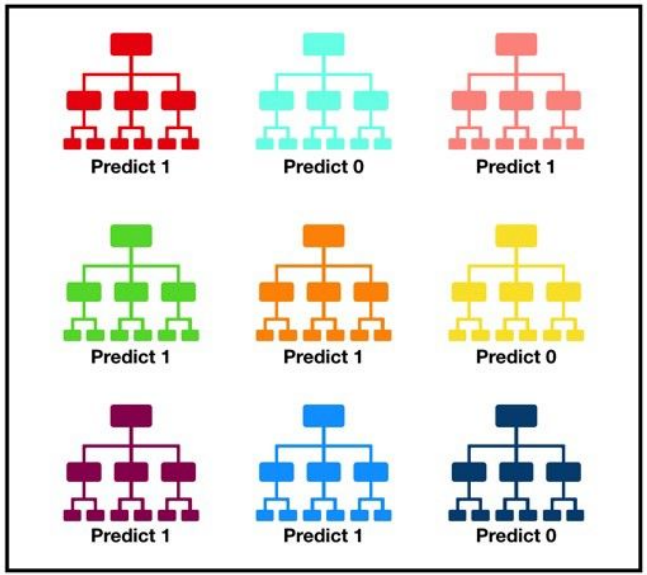
What is Bagging?

Bagging is an approach is to use the same training algorithm for every predictor, but to train them on **different random subsets of the training set**. When sampling is performed with replacement, this method is called **bagging** (bootstrap aggregating).
Once all predictors are trained, the ensemble can make a prediction for a new instance by simply aggregating the predictions of all predictors. The aggregation function is typically the statistical mode for classification

What is Bagging?



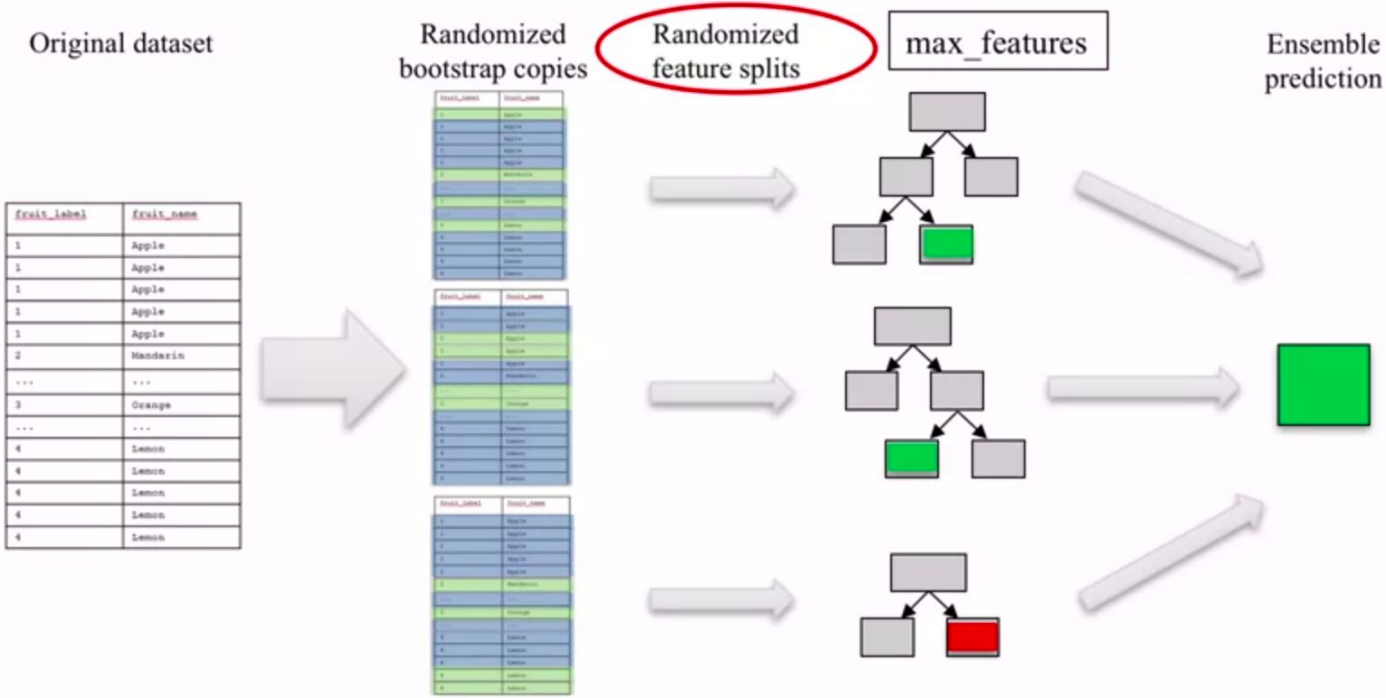
Random Forest Introduction



Tally: Six 1s and Three 0s
Prediction: 1

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. **Each individual tree in the random forest spits out a class prediction** and **the class with the most votes becomes our model's prediction**

Random Forest Process



Few Parameters in Random Forest

- **max_features:**
  These are the maximum number of features Random Forest is allowed to try in individual tree
- **n_estimators :**
  This is the number of trees you want to build before taking the maximum voting or averages of predictions
- **min_sample_leaf :**
  Leaf is the end node of a decision tree. A smaller leaf makes the model more prone to capturing noise in train data.

Few Parameters in Random Forest

- **n_jobs :**
  This parameter tells the engine how many processors is it allowed to use
- **Random_state:**
  A definite value of random_state will always produce same results if given with same parameters and training data

For greater explanation of each parameter, go to here

The Advantages of RF

1. It **reduces overfitting** problem in decision trees and also **reduces the variance** and therefore **improves the accuracy**.
2. Random Forest can automatically **handle missing values**.
3. **No feature scaling required:** No feature scaling (standardization and normalization) required in case of Random Forest as it uses rule based approach instead of distance calculation.
4. Random Forest can automatically **handle missing values**.
5. Random Forest is usually **robust to outliers** and can handle them automatically.

The Disadvantages of RF

1. **Complexity:** Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. On the other hand decision tree is simple and does not require so much computational resources.
2. **Longer Training Period:** Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

Tips and References

1. Introduction to Statistical Learning:
   Source
2. The Element of Stastical Learning:
   Source
3. Pattern Recognition and Machine Learning:
   Source
4. Interpretable ML Book:
   Source

Group Practice

Based on this dataset:

1. Build a churn prediction model (any technique is fine) and show the performance of the model in forms of confusion metrics / ROC / Precision vs Recall curve with **the proper ML workflow**(slide 6) that we have discussed in the previous classes.
2. Why do you choose this model over other techniques?
3. If you have time to improve the model, how would you do so?

Thank you!