# Classification

# Classification:

is a process in which you can train a model based on existing data and predict the classes or values of new samples based on the trained model.

- The objective of classification is to accurately predict the target class for each record in the data.

# The Most Popular Classification Algorithm:

Decision Tree

K Nearest Neighbor

Naïve Bayes Classifier

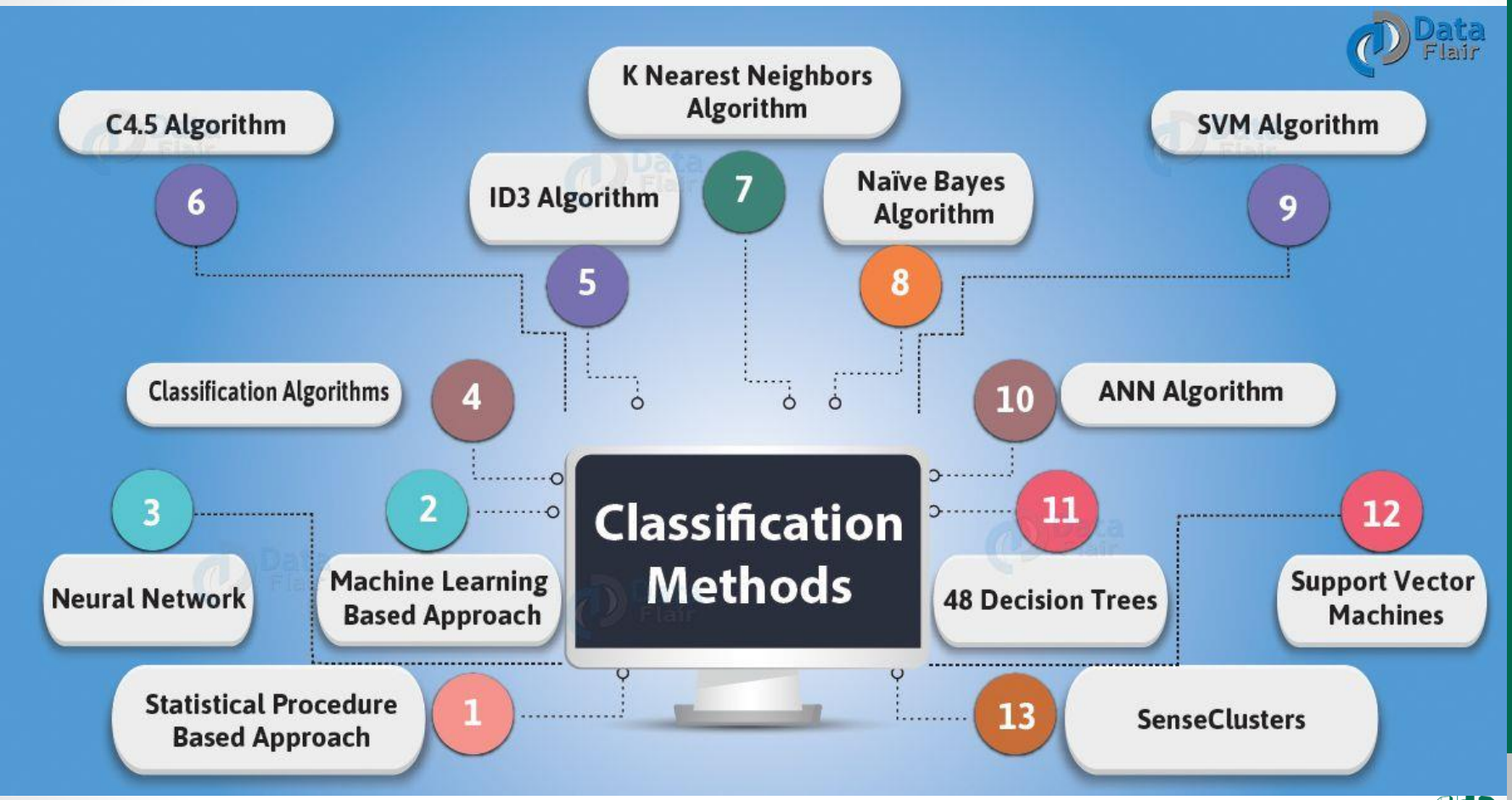
Support Vector Machine

Linear Regression

Logistic Regression

Artificial Neural Networks

Random Forests


Data Flair
Classification Methods
1
Statistical Procedure Based Approach
2
Machine Learning Based Approach
3
Neural Network
4
Classification Algorithms
5
ID3 Algorithm
6
C4.5 Algorithm
7
K Nearest Neighbors Algorithm
8
Naïve Bayes Algorithm
9
SVM Algorithm
10
ANN Algorithm
11
48 Decision Trees
12
Support Vector Machines
13
SenseClusters

# 1. Decision Trees:

A decision tree is a flowchart-like tree structure. The structure represents the “leaves” and “branches”, where each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label.

- It works for both categorical and continuous input and output attributes.
- Decision trees have a natural “if … then … else …” construction

# Example 1:

For a bank to consider whether or not to offer someone a loan they often go through a sequential list of questions to figure out if it is safe to give said loan to an individual.

- What kind of income does the person have?

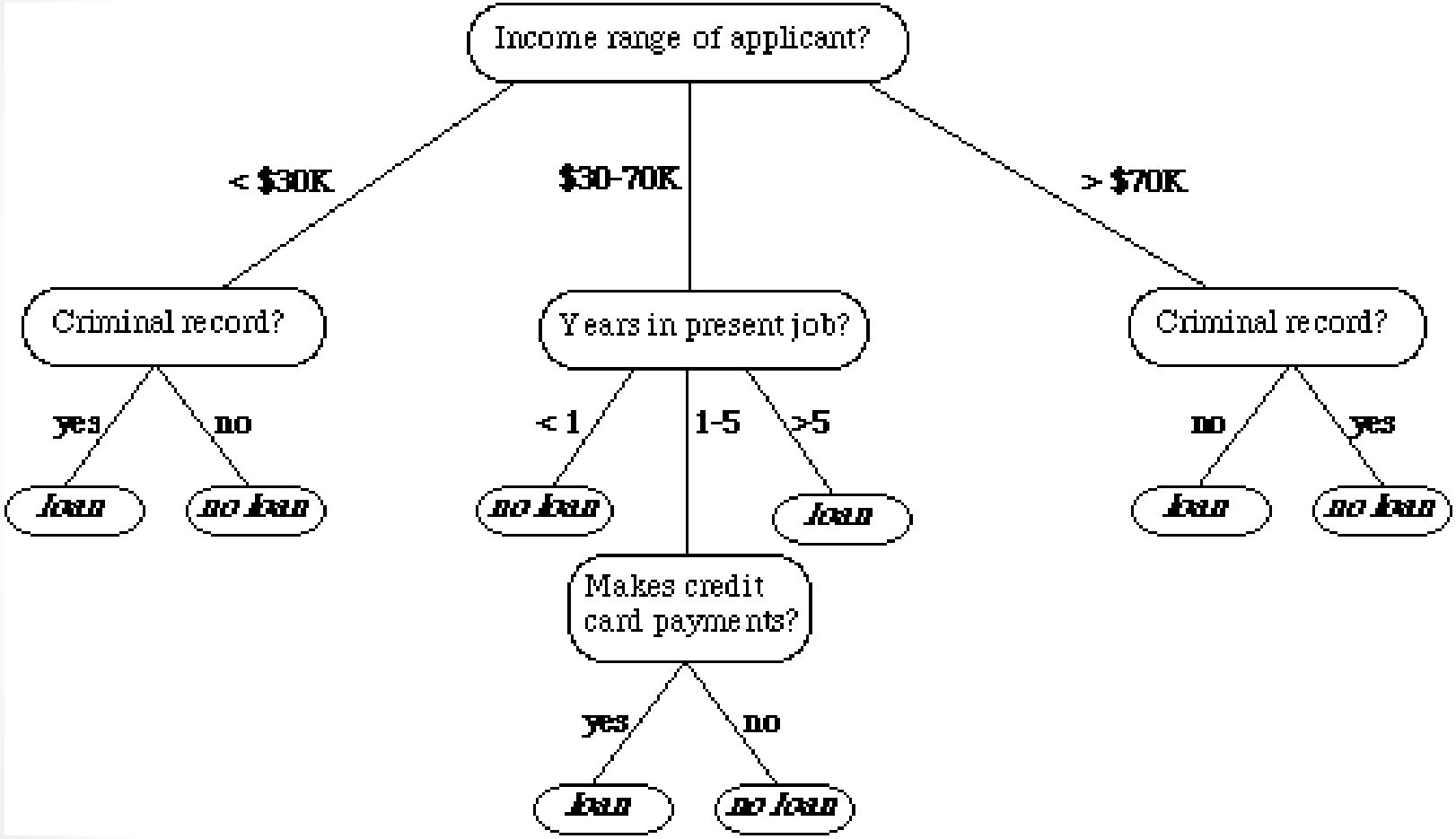
If it is between $30–70k they move on to the next question.

- How long have they held their current job?

If 1–5 years it leads to their next question.

- Do they make their credit card payments?

If yes then they offer the Loan and if no they do not.

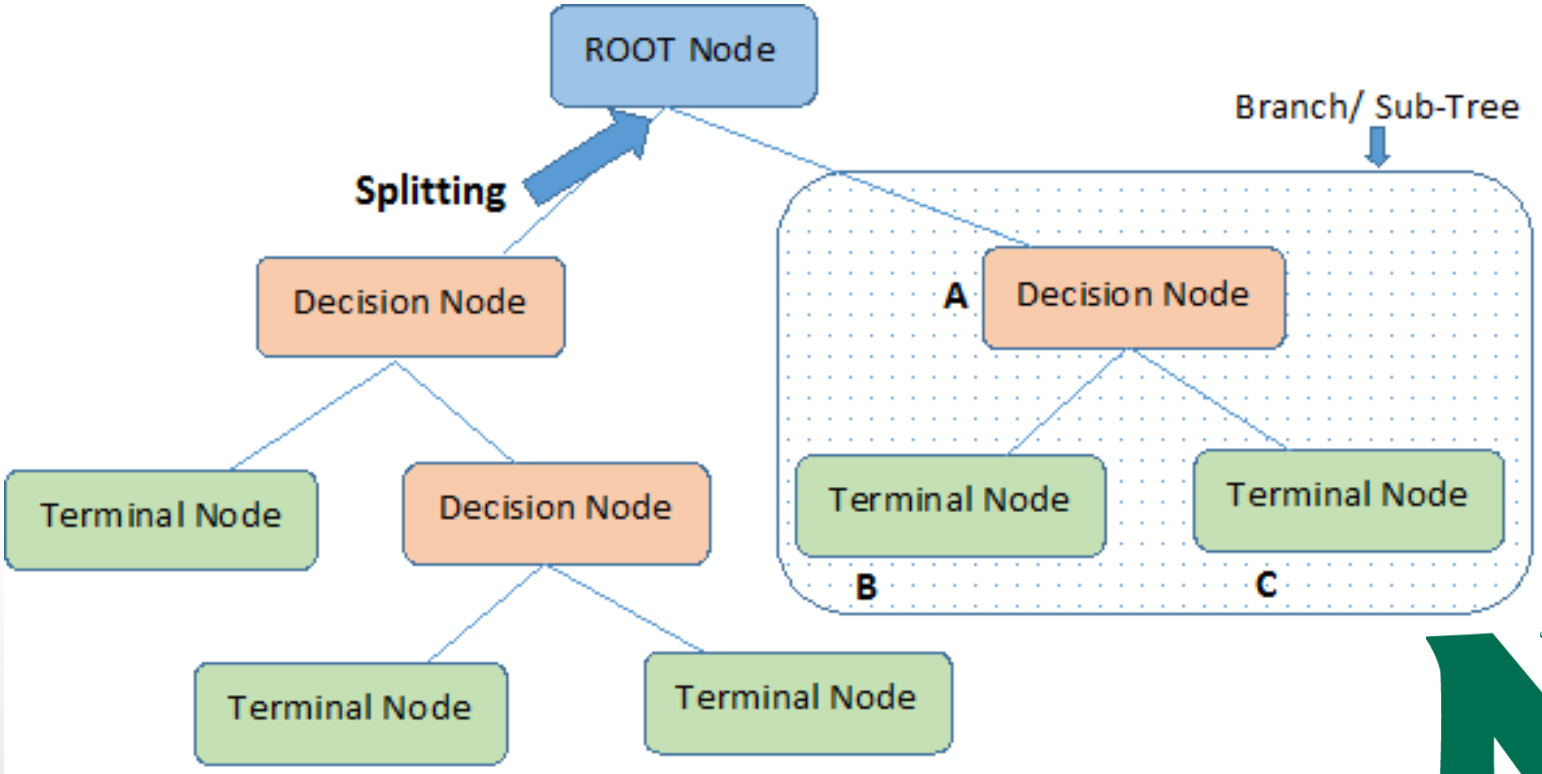

Income range of applicant?
< $30K
$30-70K
> $70K
Criminal record?
Years in present job?
Criminal record?
yes
no
< 1
1-5
>5
no
yes
loan
no loan
no loan
loan
loan
no loan
Makes credit card payments?
yes
no
loan
no loan

# Important Terminology Related to Decision Tree:

1. Root Node: It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. Splitting: It is a process of dividing a node into two or more sub-nodes.
3. Decision Node: When a sub-node splits into further sub-nodes, then it is called decision node.
4. Leaf / Terminal Node: Nodes do not split is called Leaf or Terminal node.
5. Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.

# Important Terminology Related to Decision Trees:

1. Branch / Sub-Tree: A sub section of entire tree is called branch or sub-tree.
2. Parent and Child Node: A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.



# Example 2: (Decision Tree - Titanic)

On April 15, 1912, the largest passenger liner ever made collided with an iceberg during her maiden voyage. When the Titanic sank it killed 1502 out of 2224 passengers and crew. "*Titanic.xls*" includes a subset of the original dataset.

1. Import the dataset to R.

2. Find the dataset dimension. (dim())

```
> dim(Titanic)
[1] 1312    5
```

# Example 2: (Decision Tree - Titanic)

3. Show the dataset structure. (str())

4. Show the dataset head. (head())

```
> str(Titanic)
Classes 'tbl_df', 'tbl' and 'data.frame':       1312 obs. of  5 variables:
 $ Name    : chr  "Abbing, Mr Anthony" "Abbott, Master Eugene Joseph" "Abbott,
ton (Rosa)" ...
 $ Class   : chr  "3rd" "3rd" "3rd" "3rd" ...
 $ Age     : num  42 13 16 35 16 25 30 28 18 20 ...
 $ Sex     : chr  "male" "male" "male" "female" ...
 $ Survived: chr  "dead" "dead" "dead" "survived" ...
```

```
> head(Titanic)
# A tibble: 6 x 5
  Name                        Class   Age Sex    Survived
  <chr>                       <chr> <dbl> <chr>  <chr>
1 Abbing, Mr Anthony          3rd      42 male   dead
2 Abbott, Master Eugene Joseph 3rd     13 male   dead
3 Abbott, Mr Rossmore Edward  3rd      16 male   dead
4 Abbott, Mrs Stanton (Rosa)  3rd      35 female survived
5 Abelseth, Miss Anna Karen   3rd      16 female survived
6 Abelseth, Mr Olaus          3rd      25 male   survived
```

# Example 2: (Decision Tree - Titanic)

5. Check if there are missing values. (summary())

6. Create a new dataset (Titanic1) and select only Class, Sex, and Survived. (You need to install dplyr package)

```
> summary(Titanic)
     Name               Class                Age              Sex                Survived
 Length:1312        Length:1312        Min.    : 0.17   Length:1312        Length:1312
 Class :character   Class :character   1st Qu.:21.00   Class :character   Class :character
 Mode  :character   Mode  :character   Median :28.00   Mode  :character   Mode  :character
                                       Mean   :30.40
                                       3rd Qu.:39.00
                                       Max.   :71.00
                                       NA's   :556
```

```
> Titanic1 = select(Titanic, -Name, -Age)
> head(Titanic1)
# A tibble: 6 x 3
  Class Sex    Survived
  <chr> <chr>  <chr>
1 3rd   male   dead
2 3rd   male   dead
3 3rd   male   dead
4 3rd   female survived
5 3rd   female survived
6 3rd   male   survived
```

# Example 2: (Decision Tree - Titanic)

5. Install the "rpart" package and library.
6. Use the "rpart" function to create a decision tree.

```
install.packages("rpart")
library("rpart")
```

```
> Titanic_Tree = rpart(Survived~., data=Titanic1, method="class", model=TRUE)
> Titanic_Tree
n= 1312

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 1312 450 dead (0.6570122 0.3429878)
  2) Sex=male 850 142 dead (0.8329412 0.1670588) *
  3) Sex=female 462 154 survived (0.3333333 0.6666667)
    6) Class=3rd 212  80 dead (0.6226415 0.3773585) *
    7) Class=1st,2nd 250  22 survived (0.0880000 0.9120000) *
```

# Example 2: (Decision Tree - Titanic)

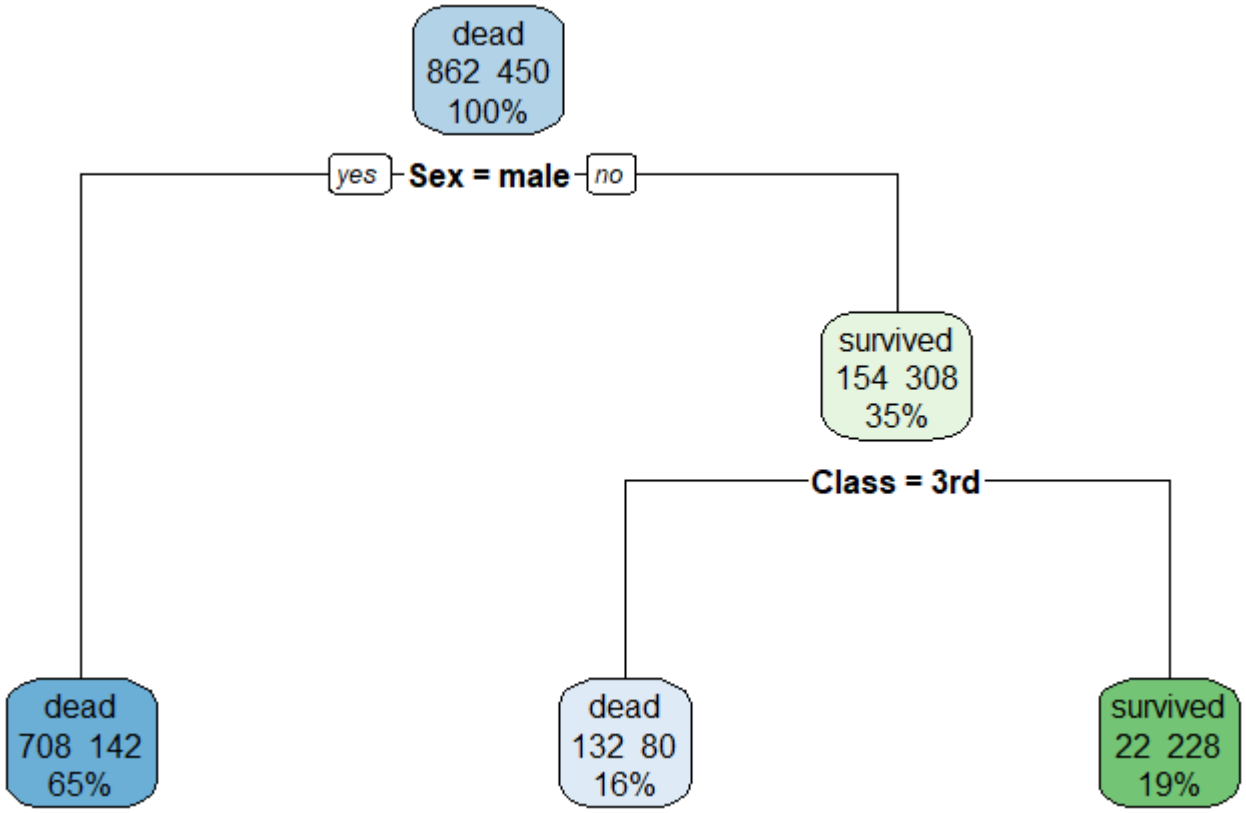
5. Install the “rpart.plot” package and library.

6. Use the “rpart.plot” function to graph the decision tree.

```
install.packages("rpart.plot")
library("rpart.plot")
```

```
> rpart.plot(Titanic_Tree, extra = 101)
```

# Example 2: (Decision Tree - Titanic)




- There were 1,312 observations (862 dead, 450 survived).
- Out of 850 male, 708 dead (65%).
- Out of 462 female, 154 dead (35%).
- Out of 154 survived female 3rd class holders, 132 dead (16%).
- 22 (19%) dead of the female 1st and 2nd class holders.

# Example 2: (Decision Tree - Titanic)

7. Use the count() function to explain the tree.

```
> Sex_count = count(Titanic1, Sex)
> Sex_count
# A tibble: 2 x 2
  Sex        n
  <chr>  <int>
1 female   462
2 male     850
```

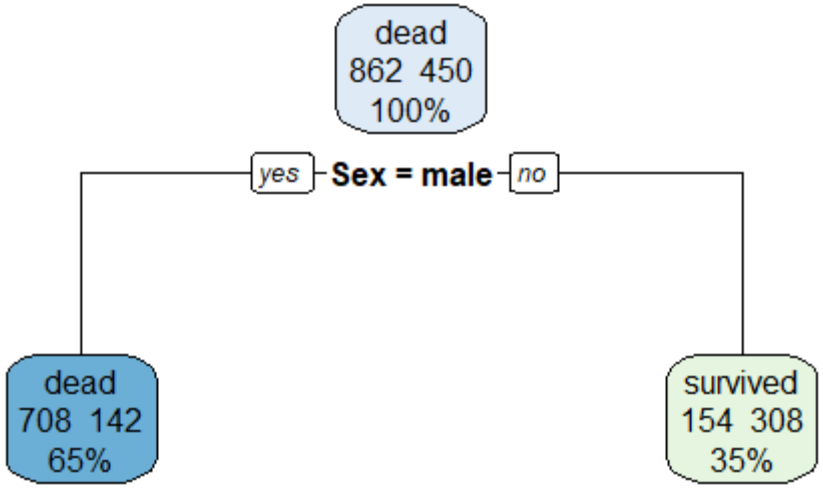
```
> Sex_Survived_count = count(Titanic1, Survived, class=Sex)
> Sex_Survived_count
# A tibble: 4 x 3
  Survived class      n
  <chr>    <chr>  <int>
1 dead     female   154
2 dead     male     708
3 survived female   308
4 survived male     142
```

```
> All_count = count(Titanic1, Sex, class=Survived, Class)
> All_count
# A tibble: 12 x 4
   Sex    class    Class     n
   <chr>  <chr>    <chr> <int>
 1 female dead     1st       9
 2 female dead     2nd      13
 3 female dead     3rd     132
 4 female survived 1st     134
 5 female survived 2nd      94
 6 female survived 3rd      80
 7 male   dead     1st     120
 8 male   dead     2nd     147
 9 male   dead     3rd     441
10 male   survived 1st      59
11 male   survived 2nd      25
12 male   survived 3rd      58
```

# Example 2: (Decision Tree - Titanic)

8. Create a decision tree for the survivals by sex.

```
> Titanic_Tree = rpart(Survived~Sex, data=Titanic1, method="class", model=TRUE)
> rpart.plot(Titanic_Tree, extra = 101)
```




- There were 1,312 observations.
- Out of 850 male, 708 dead (65%).
- Out of 462 female, 154 dead (35%).

# Advantages of Decision Tree:

1. Easy to Understand.
2. Useful in Data exploration.
3. Less data cleaning required.
4. Non Parametric Method.
5. Data type is not a constraint (numerical or categorical).

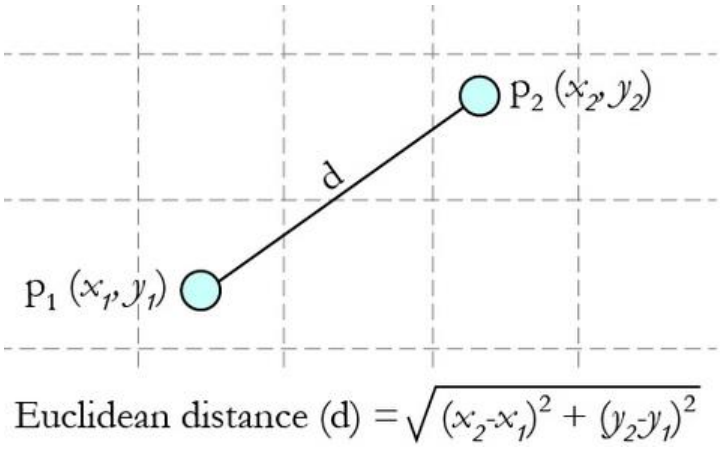
# Disadvantages of Decision Tree:

1. Over fitting.
2. Not fit for continuous variables.
3. It can become excessively complex.

# 2. K-Nearest Neighbor (KNN):

The KNN classifier is commonly based on the Euclidean distance between a test sample and the specified training samples.

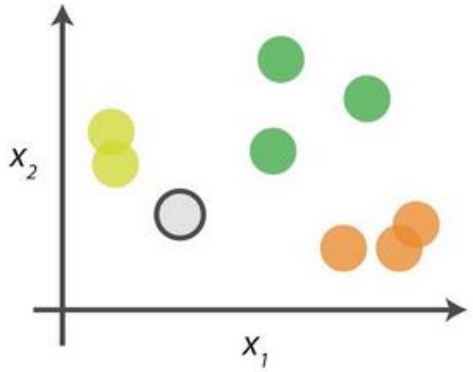
- It should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution of the data.
- KNN is used in the variety of applications.



# How Does KNN Work?

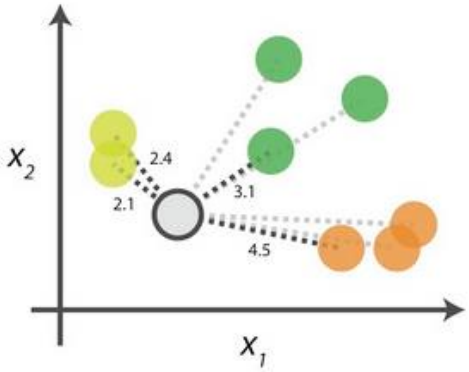
- Select the number K of the neighbors.
- Calculate the Euclidean distance of K number of neighbors.
- Take the K nearest neighbors as per the calculated Euclidean distance.
- Among these k neighbors, count the number of the data points in each category.
- Assign the new data points to that category for which the number of the neighbor is maximum.
- Our model is ready.

## 0. Look at the data



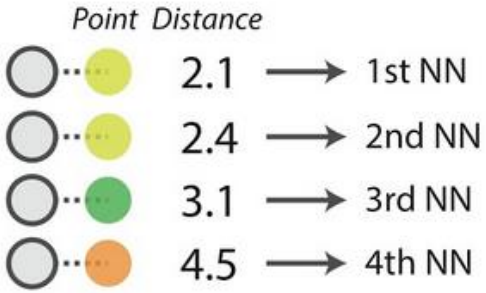

Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

## 1. Calculate distances



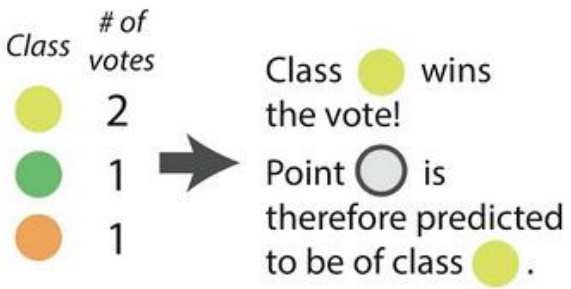

Start by calculating the distances between the grey point and all other points.

## 2. Find neighbours

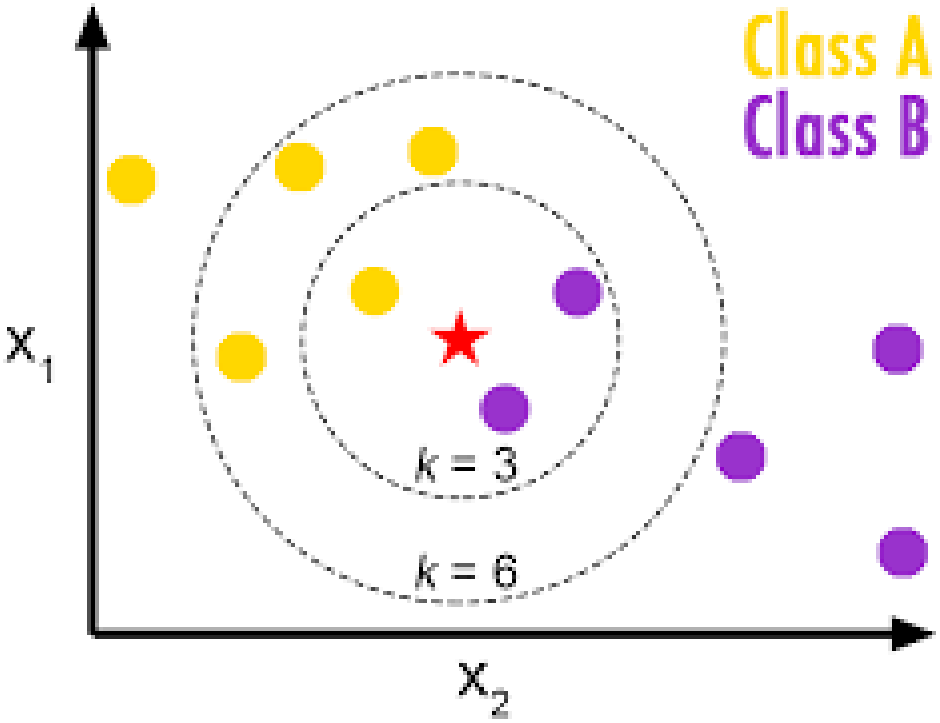



Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

## 3. Vote on labels




Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.


Class A
Class B
$x_1$
$x_2$
$k = 3$
$k = 6$

# Example:

The *ChickWeight* data frame has 578 rows and 4 columns from an experiment on the effect of diet on early growth of chicks. The body weights of the chicks were measured at birth and every second day thereafter until day 20. They were also measured on day 21. There were four groups on chicks on different protein diets.

1. Check the data dimension.

```
> dim(ChickWeight)
[1] 578   4
```

# Example:

2. Check the data head (top).
3. Check the data tail (bottom).

```
> head(ChickWeight)
  weight Time Chick Diet
1     42    0     1    1
2     51    2     1    1
3     59    4     1    1
4     64    6     1    1
5     76    8     1    1
6     93   10     1    1
```

```
> tail(ChickWeight, 5)
    weight Time Chick Diet
574    175   14    50    4
575    205   16    50    4
576    234   18    50    4
577    264   20    50    4
578    264   21    50    4
```

# Example:

4. Split the data into to groups training data (90%) and test data (10%).
5. Check the dimension of the two groups.

```
> smp_size = floor(0.90 * nrow(ChickWeight))
> smp_size
[1] 520
> index = sample(seq_len(nrow(ChickWeight)), size = smp_size)
```

```
> train = ChickWeight[index, ]
> dim(train)
[1] 520   4
```

```
> test = ChickWeight[-index, ]
> dim(test)
[1] 58  4
```

# Example:

6. To work with the KNN classification, we need to install the Class package and library.

```
install.packages("class")
library("class")
```

Note: type "?" before the function in R console and hit enter, then the help document will appear in plots and files.

```
?knn
```

# Example:

7. Use the function KNN() with 1 neighbor (K).
8. Evaluate the model using a cross-tabulation "contingency table".
9. Use the function mean() to check how much of the observations are correctly predicted

```
> knn_pred1 = knn(train, test, train[, 4], k = 1)
```

```
> table(knn_pred1, test[, 4])

knn_pred1  1  2  3  4
        1 27  1  0  0
        2  0 14  2  0
        3  0  1  4  2
        4  0  0  0  7
```

```
> mean(knn_pred1 == test[, 4])
[1] 0.8965517
```

- Among 28 chicks, 27 or 96%, is success rate in the first group.

# Example:

10. Find the probability contingency table. Compare the results with the results of the function mean().

```
> prop.table = prop.table(table(knn_pred1, test[, 4]))
> prop.table

knn_pred1          1          2          3          4
        1 0.46551724 0.01724138 0.00000000 0.00000000
        2 0.00000000 0.24137931 0.03448276 0.00000000
        3 0.00000000 0.01724138 0.06896552 0.03448276
        4 0.00000000 0.00000000 0.00000000 0.12068966
```

**Note:** the total of the diagonal probability elements is

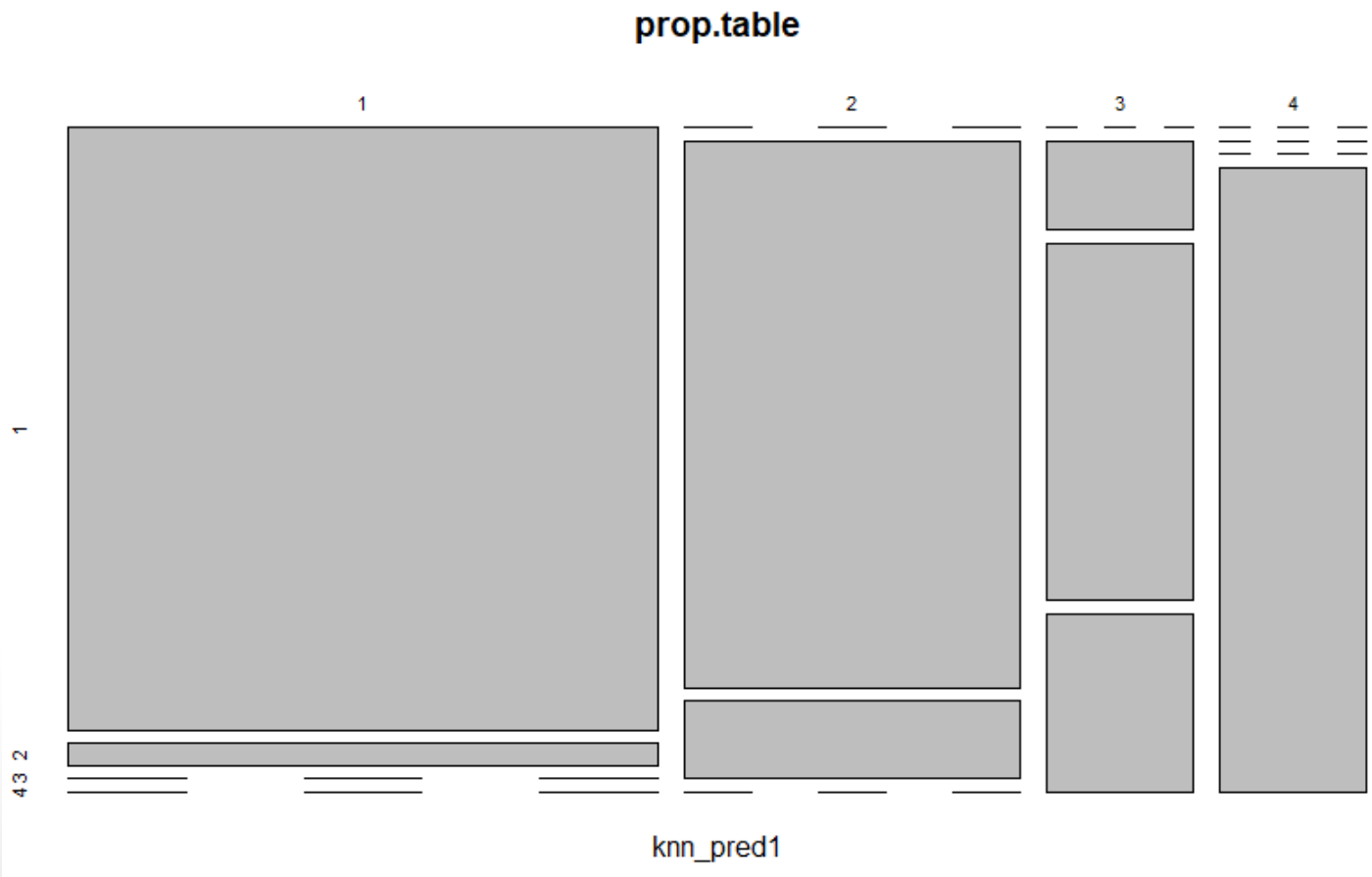
$$0.466 + 0.241 + 0.069 + 0.121 = 0.987$$

Which is the same value of the function mean.

# Example:

11. Graph the probability contingency table.

```
> plot(prop.table)
```



# Example:

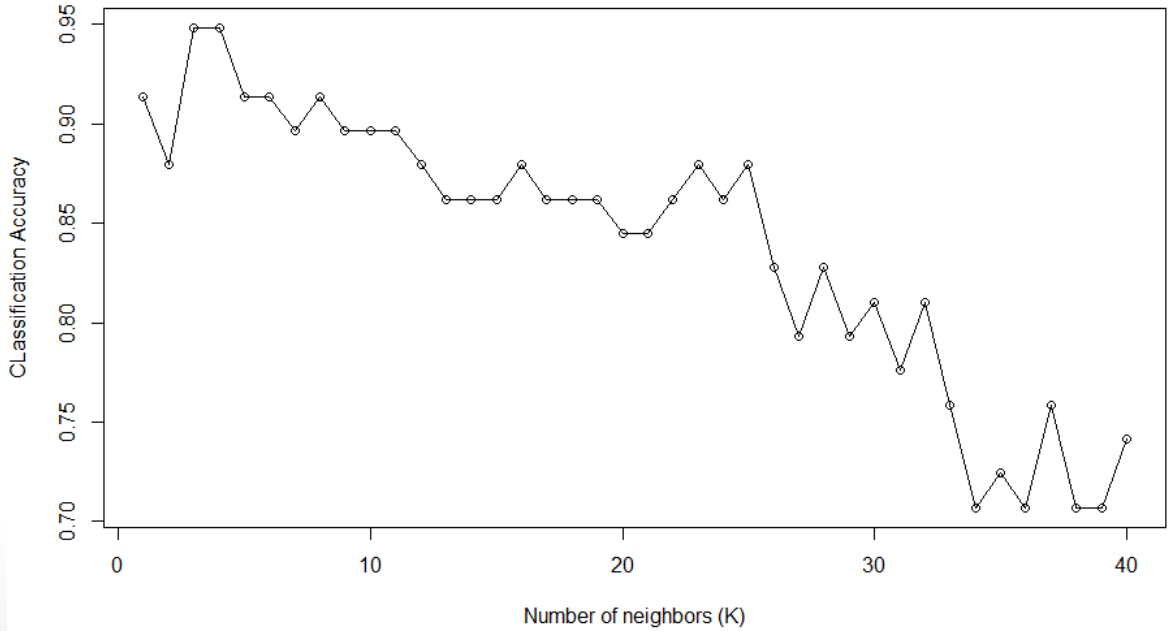
12. How many neighbors should we use?

Use loop to find that.

13. Graph the means.

```
> mean1 = 0
> for (i in 1:40) {
+   knn_pred = knn(train, test, train[, 4], k = i)
+   mean1[i] = mean(knn_pred == test[, 4])
+ }
> mean1
 [1] 0.9137931 0.8793103 0.9482759 0.9482759 0.9137931 0.9137931 0.8965517 0.9137931 0.8965517 0.8965517
[11] 0.8965517 0.8793103 0.8620690 0.8620690 0.8620690 0.8793103 0.8620690 0.8620690 0.8620690 0.8448276
[21] 0.8448276 0.8620690 0.8793103 0.8620690 0.8793103 0.8275862 0.7931034 0.8275862 0.7931034 0.8103448
[31] 0.7758621 0.8103448 0.7586207 0.7068966 0.7241379 0.7068966 0.7586207 0.7068966 0.7068966 0.7413793
```

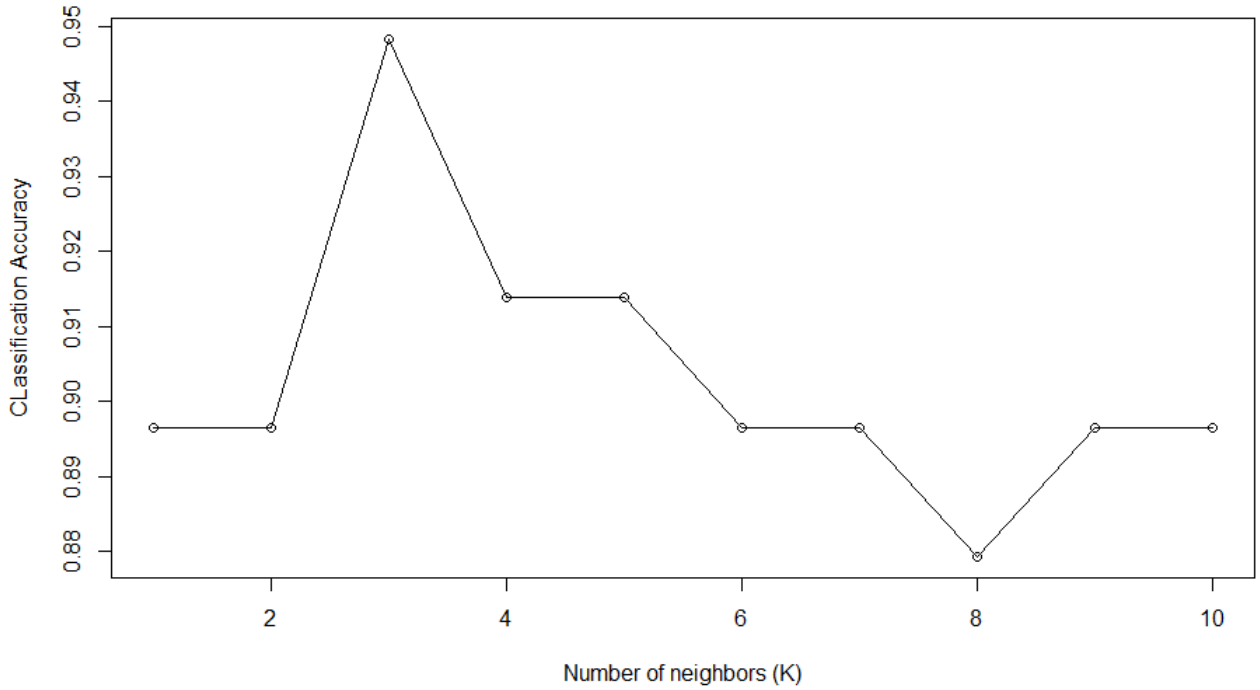
```
> plot(mean1, xlab = "Number of neighbors (K)", ylab = "CLassification Accuracy")
> lines(mean1)
```



# Example:

14. Reduce the loop to 10 turns.

```
> mean1 = 0
> for (i in 1:10) {
+   knn_pred = knn(train, test, train[, 4], k = i)
+   mean1[i] = mean(knn_pred == test[, 4])
+ }
> mean1
 [1] 0.8965517 0.8965517 0.9482759 0.9137931 0.9137931 0.8965517 0.8965517 0.8793103 0.8965517 0.8965517
```




**Note:** There is no big difference between the classification accuracy for the first 10 neighbors.

## Advantages of the KNN:

1. Easy to use and understand.
2. Quick calculation time.
3. Does not make assumptions about the data.

## Disadvantages of the KNN:

1. High memory requirement.
2. Accuracy depends on the quality of the data.
3. Must find an optimal k value (number of nearest neighbors).
4. Poor at classifying data points in a boundary where they can be classified one way or another.