# k-NN Classification

Predicting using nearest neighbors

## k Nearest Neighbours

- k is the number of nearest neighbours the classifier will use to make it’s prediction
- Instance based/Memory based learning which means that it chooses to memorize the training instances
- Non parametric which means it make no explicit assumptions of the relationship between the dependent variable and the independent variable

# k-NN Algorithm

The k-NN algorithm gets its name from the fact that it uses information about the test's k-nearest neighbours to classify it.

- k-NN algorithm treats the features as coordinates in a multidimensional feature space.
- After choosing k, for each unlabeled record in the test dataset, k-NN identifies k records in the training data that are nearest in similarity.
- The unlabeled test instance is assigned the class of the majority of the k-nearest neighbors.
- The most common measure of similarity used is Euclidean distance, but there are other measures that can be used including the Manhattan, Chebyshev and Hamming distance.

## k Nearest Neighbour

Euclidean distance which is given by

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

which is the straight line distance between any two points in space x and x' having the coordinates ($x_1$, $x_2$, …,$x_n$) and ($x'_1$, $x'_2$, …,$x'_n$) in a n-dimensional hyperplane.

# k Nearest Neighbour

Features (j) →

↓ Training Data (i)

| | $x_{ij}$ | … | $x_{ip}$ |
|---|---|---|---|
| $x_{ij}$ | 1 | | |
| | 2 | | |
| | 3 | | |
| | … | | |
| | | | |
| | | | |
| | | | |
| $x_{nj}$ | n | | |

| y |
|---|
| 1 |
| 2 |
| 3 |
| … |
| |
| |
| |
| n |

| distance |
|---|
| $d(z,x_{1j})$ |
| $d(z,x_{2j})$ |
| $d(z,x_{3j})$ |
| … |
| |
| |
| |
| $d(z,x_{nj})$ |

| result |
|---|
| r1 |
| r2 |
| r3 |
| … |
| |
| |
| |
| rn |

# k Nearest Neighbour

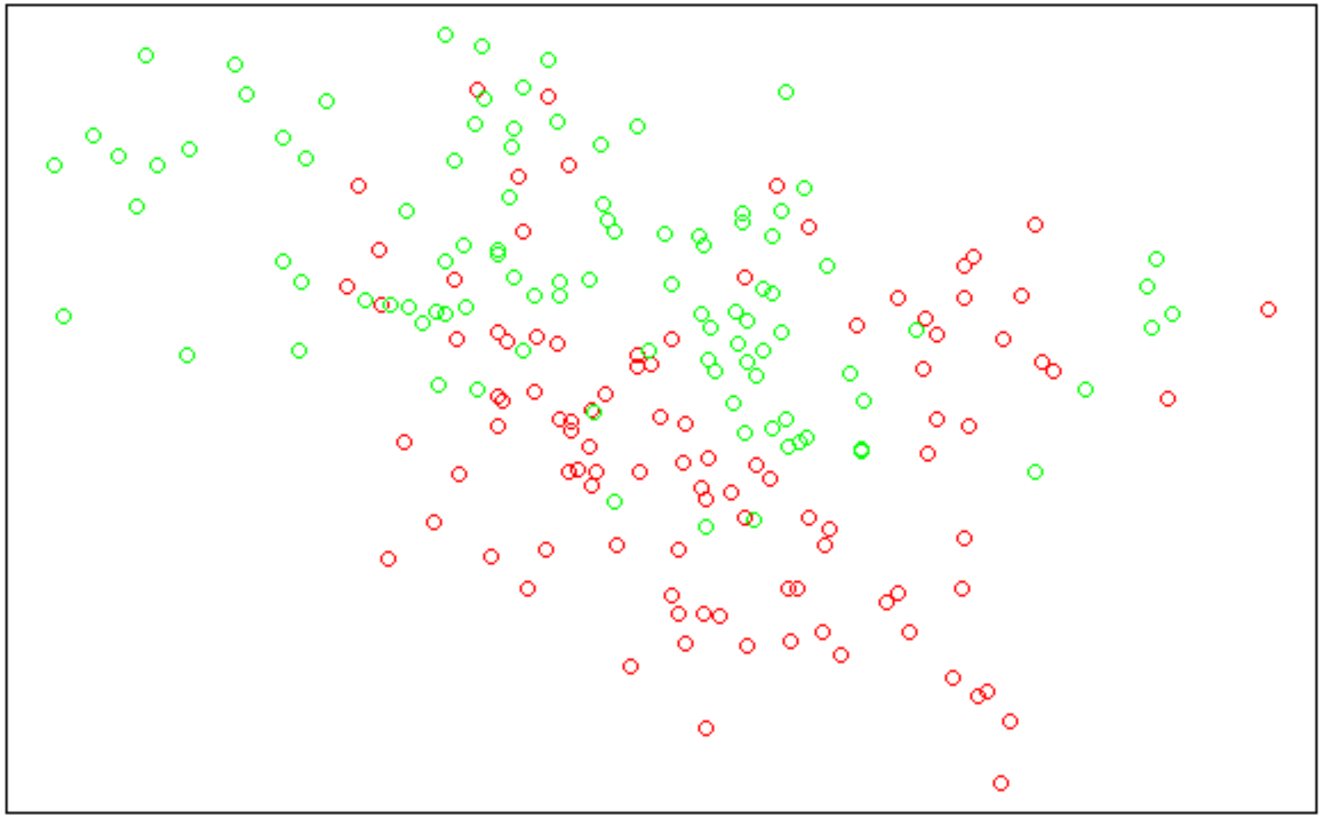
From the final output in the result vector, we choose the k values based on it’s distance in the distance vector.

The final output is calculated as an arithmetic mean across the result values of the k nearest neighbours.

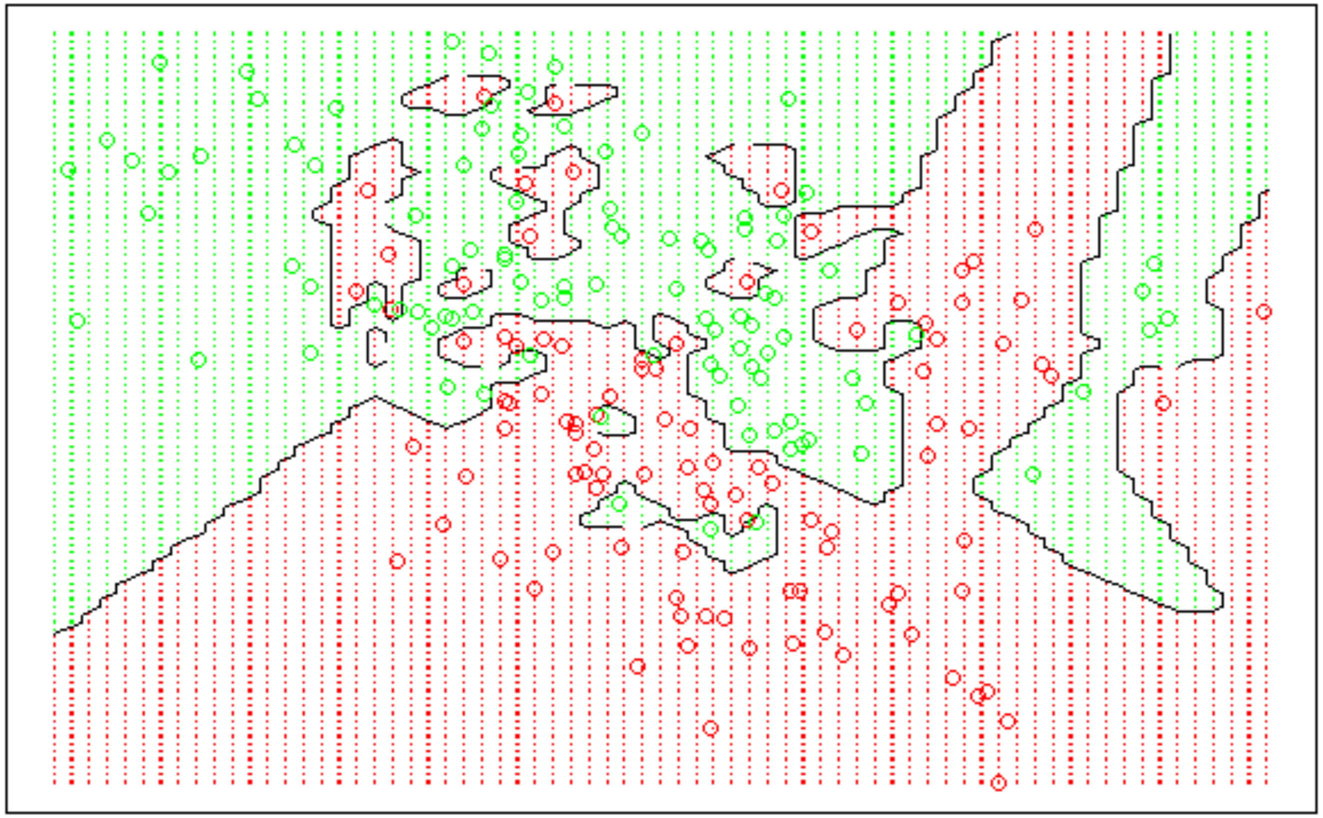
$$\bar{r} = \frac{1}{k} \sum_{i=1}^{k} r_i$$

Choosing a appropriate k determines how well the model will generalize to the test data. The balance between overfitting and underfitting the training data has to be kept in mind.

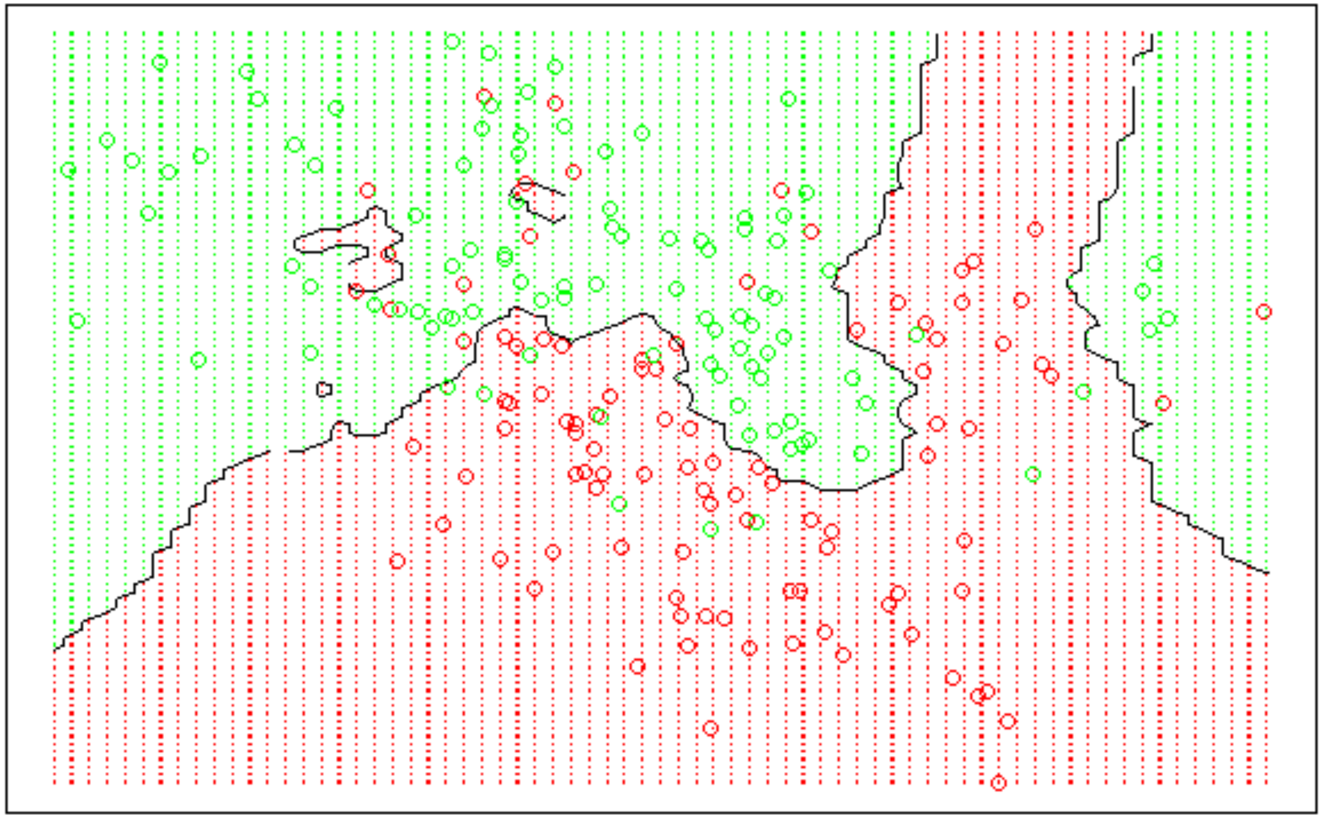
# Distribution of the 2 classes



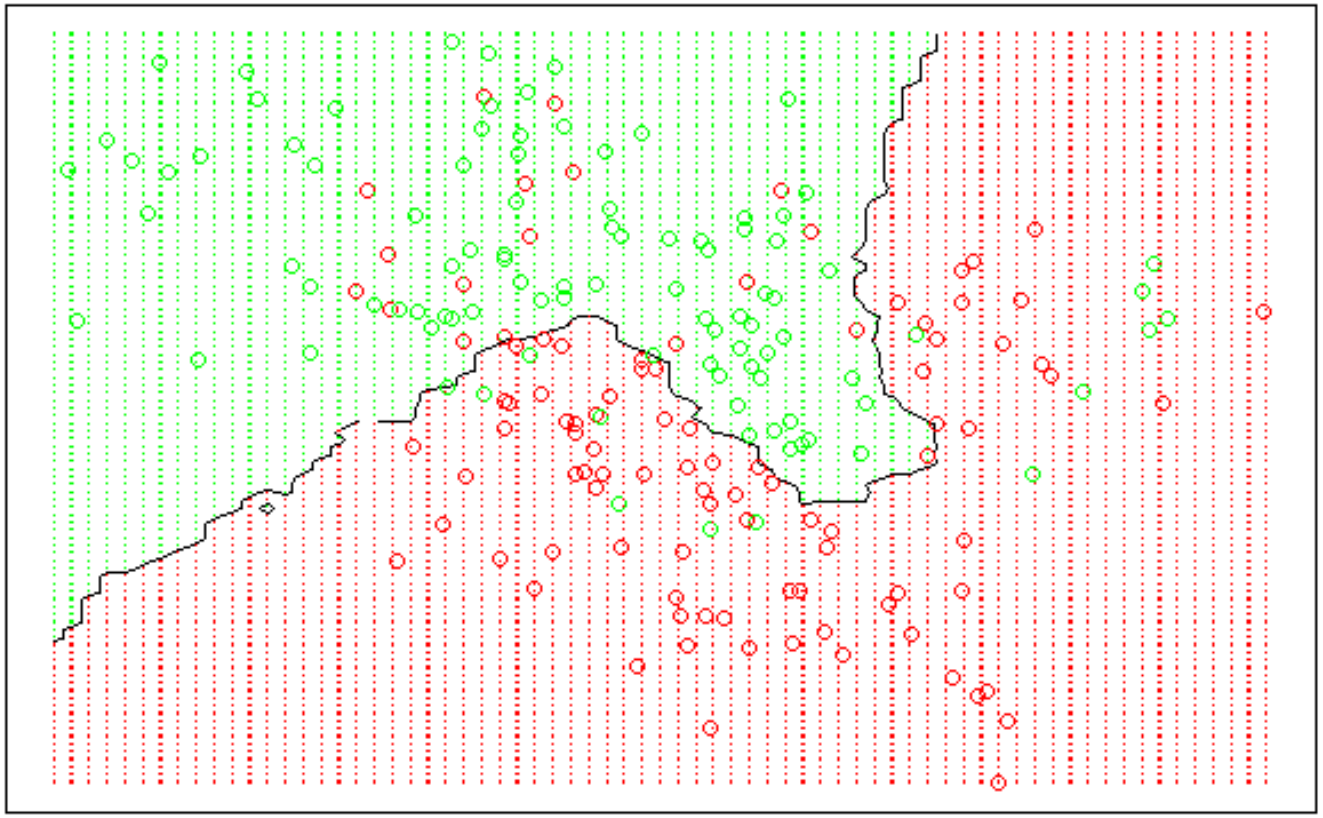
# 1-Nearest Neighbour



# 5-Nearest Neighbours



# 15-Nearest Neighbours



# Curse of Dimensionality

When the number of p is large, there tends to be deterioration in the performance of kNN and other local approaches that perform prediction using only observations that are near the test observations for which a prediction must be made.

To understand this, let's assume we have a set of observations X that is uniformly distributed on [0,1]. We have only one feature, i.e.

p = 1 feature, X

We want to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation, for instance X = 0.4, the range will be [0.35, 0.45]

which is

$$= \frac{1}{10} \times 100\%$$

10% of the space or available observations.

# Curse of Dimensionality

Now let’s take a similar example, but with 2 dimensions

p = 2 dimensions, $X_1$ and $X_2$

- $(X_1, X_2)$ are uniformly distributed on [0,1] x [0,1]
- 10% of the range of $X_1$, for example $X_1 = 0.2$, therefore [0.15, 0.25]
- 10% of the range of $X_2$, for example $X_2 = 0.7$, therefore [0.65, 0.75]

The cube which results from the ranges,

$$= \frac{1}{10} \times \frac{1}{10} \times 100\%$$

1% of the space or available observations.

# Curse of Dimensionality

Let's try to compute the percentage of the available observations for p = 100

- $(X_1, X_2, \ldots, X_{100})$ are uniformly distributed on [0,1] x [0,1] x …. x [0,1]
- 10% of the range of each X, for that test observation

The hypercube which results from the ranges, make up

$$= \frac{1}{10} \times \frac{1}{10} \times \cdots \times \frac{1}{10} \times 100\% \; = 0.1^{100} \times 100\%$$

$1^{-98}$% of the space or available observations.

**Therefore as p increase linearly, observations that are geometrically near decrease exponentially, which means that there are very few training observations NEAR any given test observation!!**

# Summary

**Advantages**

- Non parametric which means it makes no explicit assumptions of the relationship between the dependent variable and the independent variable.
- **No training phase since it keeps the training data.**
- Simple and effective.

**Disadvantages**

- It is a lazy algorithm and is computationally inefficient, since it stores all the training data.
- **Cannot work with high dimensional data, which means p has to be a small number.**
- **Slow classification phase.**