K Nearest Neighbours (K-NN)

Machine Learning Techniques

Dr. Ashish Tendulkar
IIT Madras

K-nearest neighbour or KNN is a supervised learning algorithm that can be used for both regression and classification tasks.

Overview

- It is an instance based learning technique.

- There is no explicit model, but KNN compares a new example with existing training examples, obtains k nearest neighbours and assigns an output label based on the labels of k nearest neighbors.

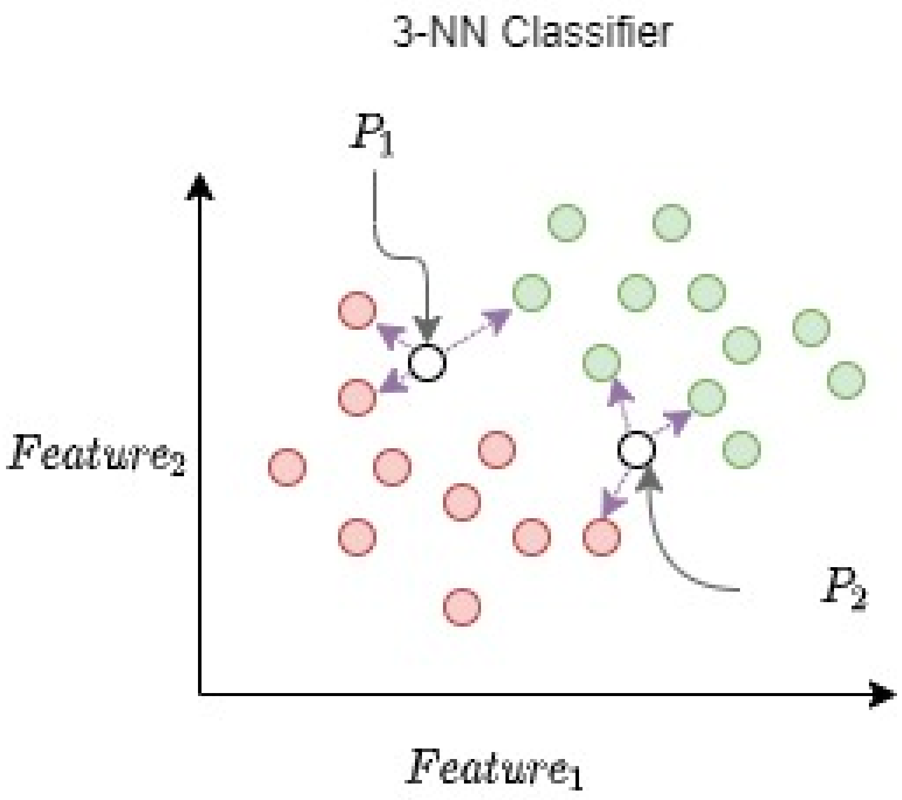
- The examples are compared with a variety of distance metrics such as Euclidean and Manhattan distance.

- There are two important questions or hyper parameters:
  - How many neighbours (k) to choose?
  - Which distance metric should be used for comparing examples?

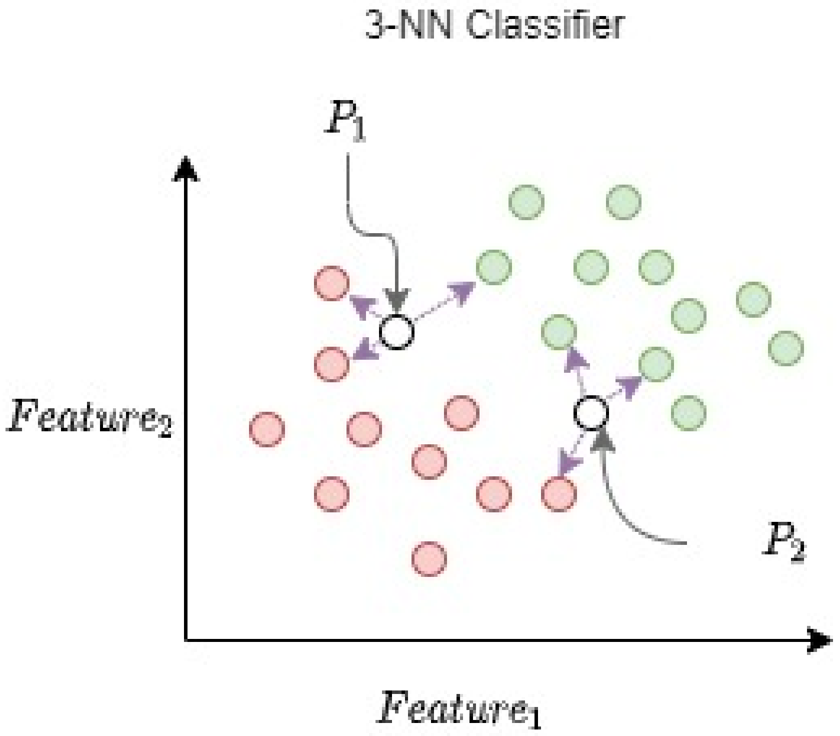
Key insight

An example is labeled by the company it keeps.

An example of 3-NN classifier



- This is an example of binary classification problem with 3-NN classifier. In this example, nearest neighbours are calculated based on *Euclidean distance.*
- It has examples from two classes: Red and Green.
- Now there are two new points P_1 and P_2, for which labels are unknown or yet to be predicted.
- Each point looks at its 3 neighbours and computes the class that is represented by 2 or 3 neighbours. Hence, the point is labeled with the majority class in its neighbourhood.



- In the figure, for P_1, 2 out of 3 neighbours are red, therefore, it is predicted to be in class Red.

- For P_2, 2 out of 3 neighbours are green, therefore, it is predicted to be in class Green.

Distance Metric

Following two metrics are used quite often:

- Euclidean distance

- Manhatten distance

Distance between two points $\mathbf{x}_1$ and $\mathbf{x}_2$ represented with m features is calculated as follows:

Euclidean distance :

$$\delta(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sqrt{(x_1^{(1)} - x_1^{(2)})^2 + (x_2^{(1)} - x_2^{(2)})^2 + \cdots + (x_m^{(1)} - x_m^{(2)})^2}$$

Manhatten Distance:

$$\delta(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \mid x_1^{(1)} - x_1^{(2)} \mid + \mid x_2^{(1)} - x_2^{(2)} \mid + \cdots + \mid x_m^{(1)} - x_m^{(2)} \mid$$

Euclidean distance

$$\delta(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sqrt{(x_1^{(1)} - x_1^{(2)})^2 + (x_2^{(1)} - x_2^{(2)})^2 + \cdots + (x_m^{(1)} - x_m^{(2)})^2}$$

Writing this compactly

$$= \left(\sum_{j=1}^{m} \left(x_j^{(1)} - x_j^{(2)} \right)^2 \right)^{\frac{1}{2}}$$

This can be rewritten in vectorized format as follows

$$= \left(\left(\mathbf{x}^{(1)} - \mathbf{x}^{(2)} \right)^T \left(\mathbf{x}^{(1)} - \mathbf{x}^{(2)} \right) \right)^{\frac{1}{2}}$$

Manhattan distance

$$\delta(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \mid x_1^{(1)} - x_1^{(2)} \mid + \mid x_2^{(1)} - x_2^{(2)} \mid + \cdots + \mid x_m^{(1)} - x_m^{(2)} \mid$$

Writing this compactly as follows

$$= \sum_{j=1}^{m} \mid x_j^{(1)} - x_j^{(2)} \mid$$

The vectorized form is as follows:

$$= \mathbf{1}_{1\times m}^T \mid \mathbf{x}^{(1)} - \mathbf{x}^{(2)} \mid_{m\times 1}$$

Model

Classification

For classification task, the k neighbours take part in voting. The class that receives highest number of votes is the predicted class.

Regression

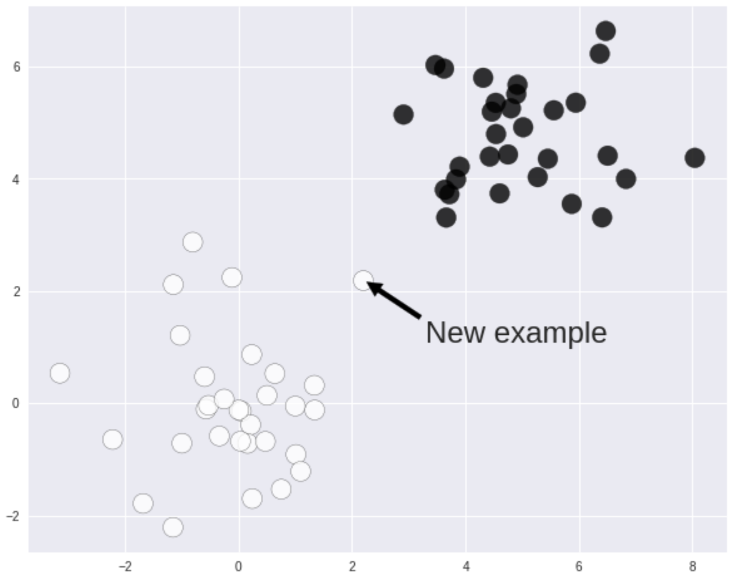
For regression task, the output/prediction is calculated as average of the outputs/labels of k neighbours.

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i$$

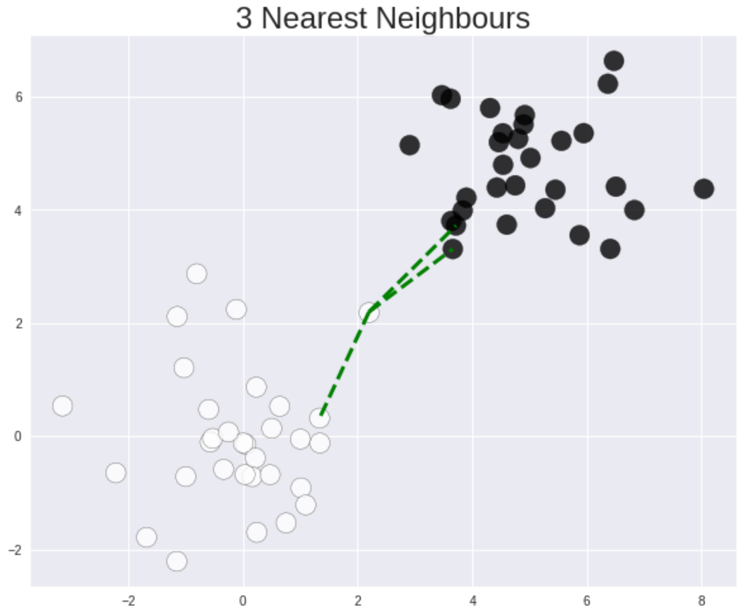
Visualization

Let us apply KNN technique and visualise how a new example is assigned a label. For this example value of k is set to be 3.

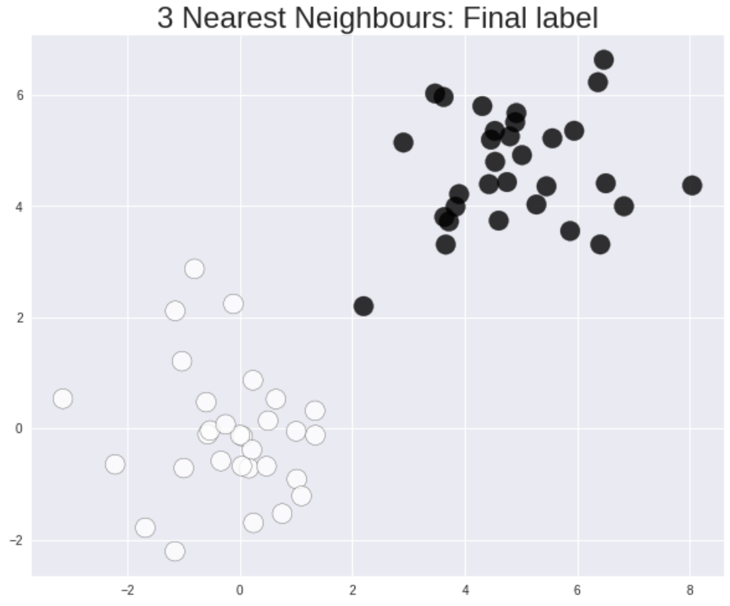
Visualize data and the new example



Find nearest 3 neighbours

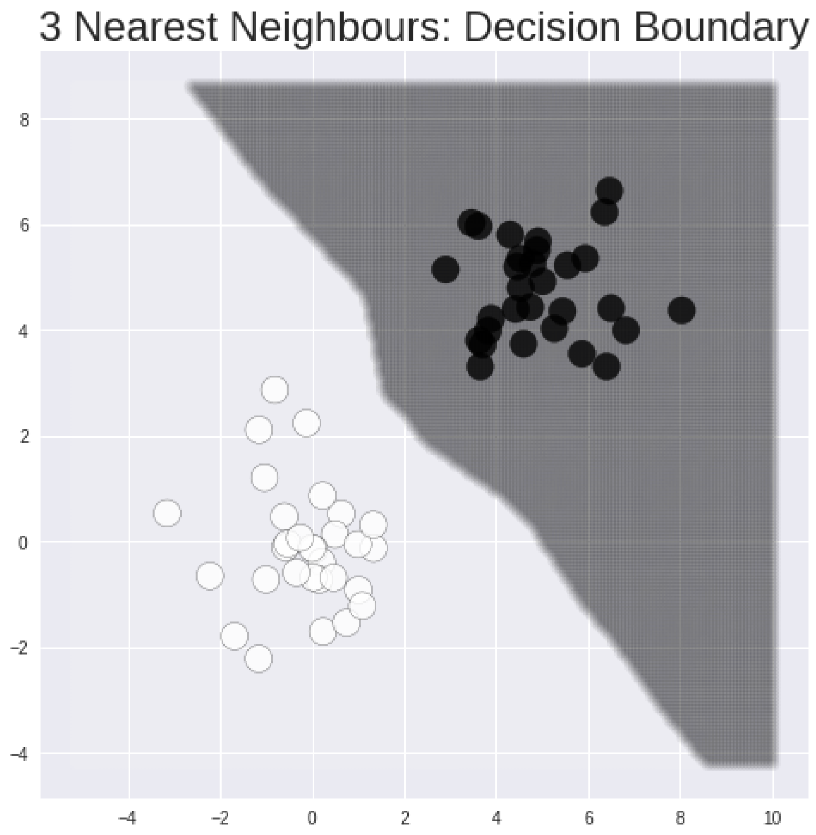


Label the new data point with majority class out of 3.



Visualization

Decision Boundary



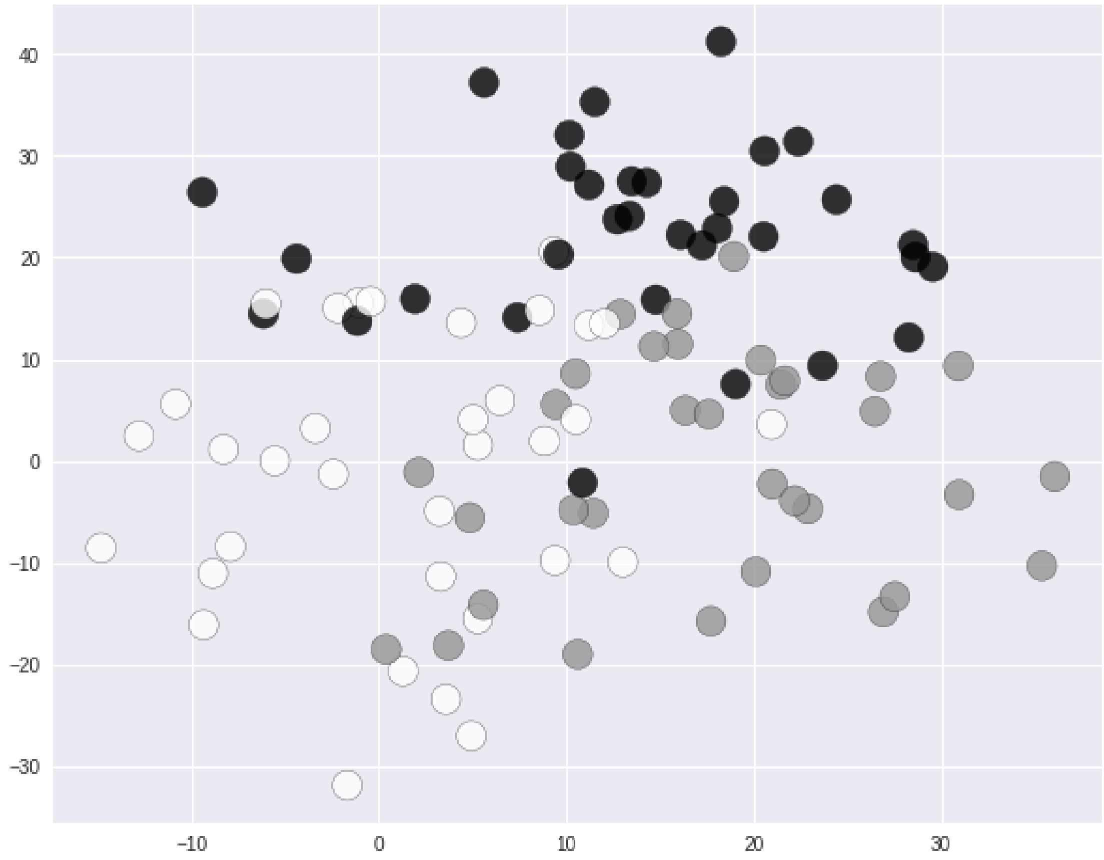
Finding best value of k

- If k is too small e.g. 1 or 2, then our model is sensitive to noise. The model will try to adjust to small changes in variance. In this case, the model will *overfit*. The decision boundary will be very jagged.

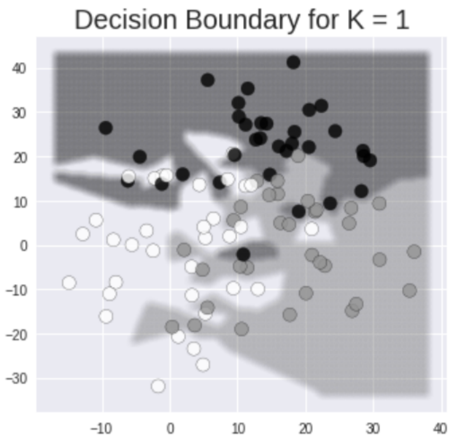
- On the other hand, if k is too large, then our model will be biased. The model will tend to ignore the underlying trend. In this case, the model will *underfit*.

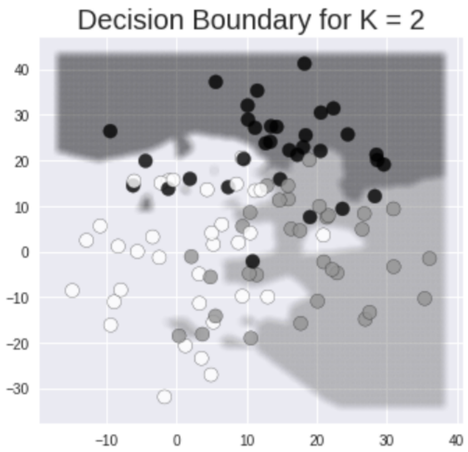
- As value of k comes close to total number of points in the dataset, the model will predict label of majority class for every possible example.

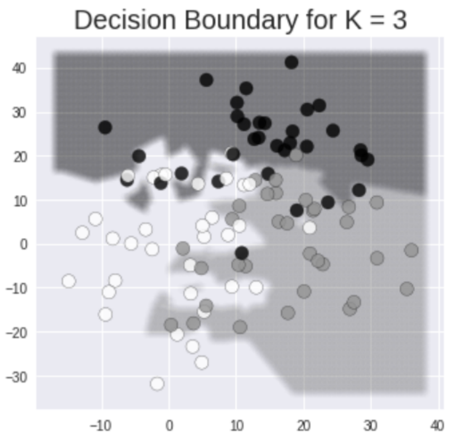
Generate another dataset and observe decision boundary for different values of k.

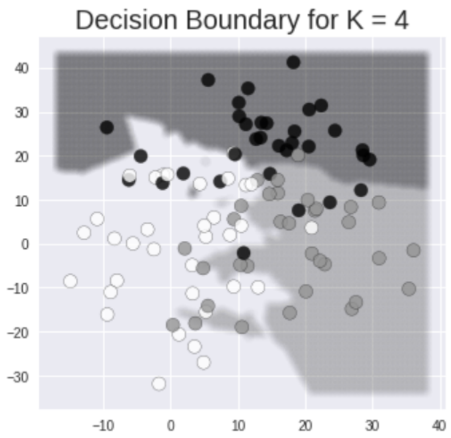


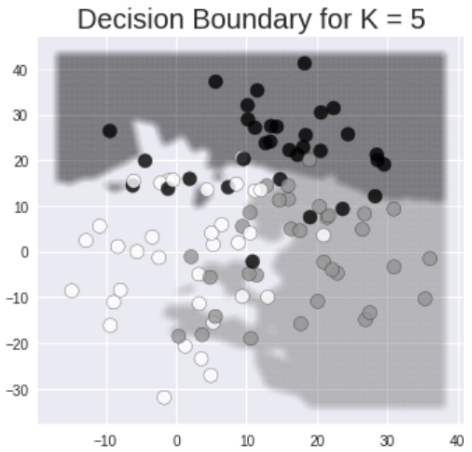
Decision boundary for different values of k.

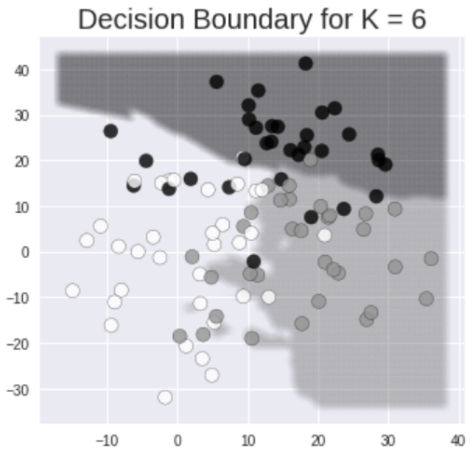


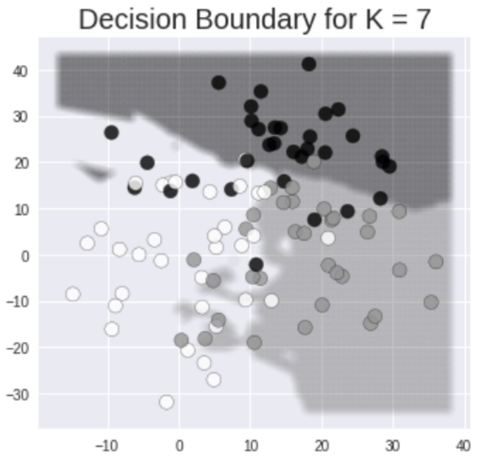


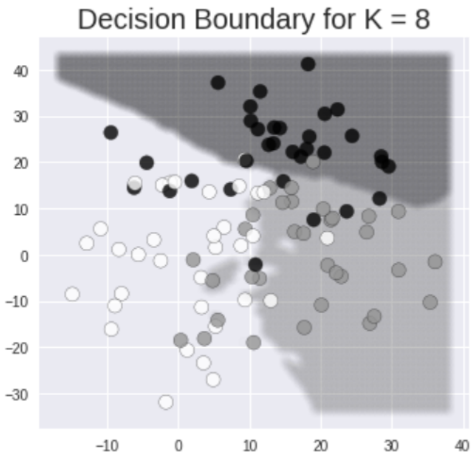


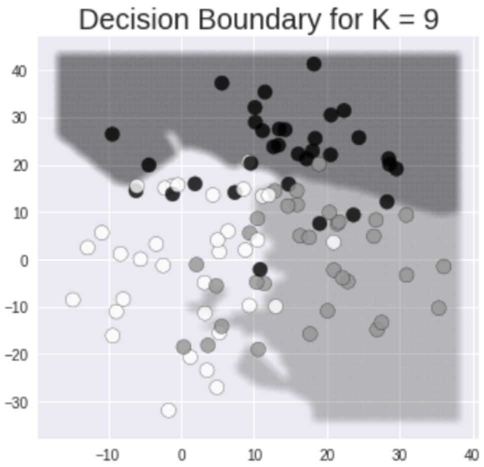


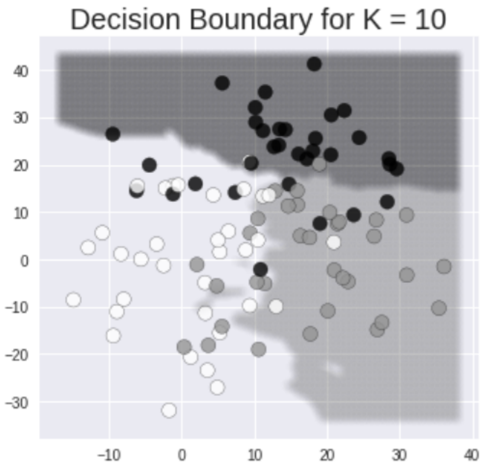


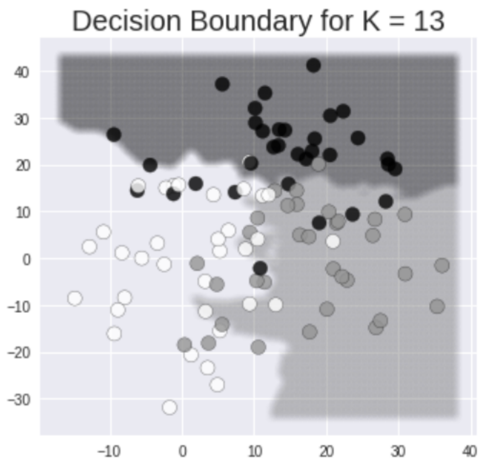


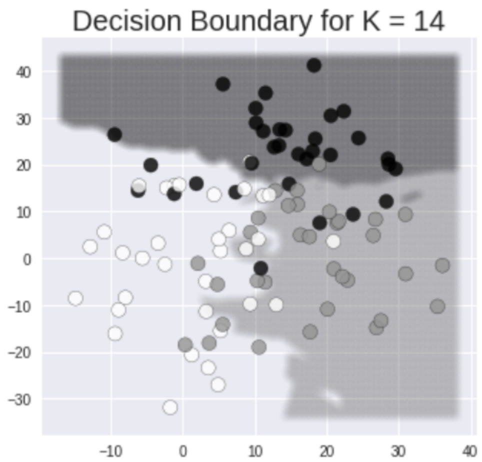


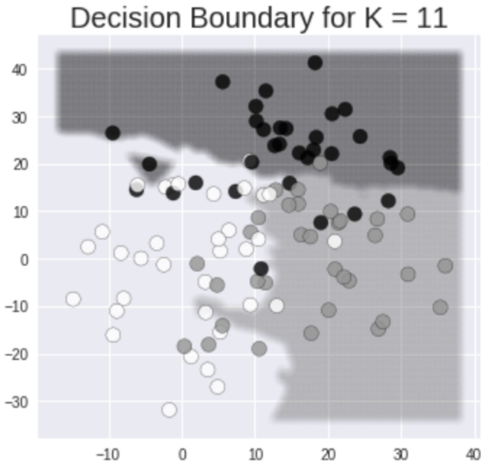


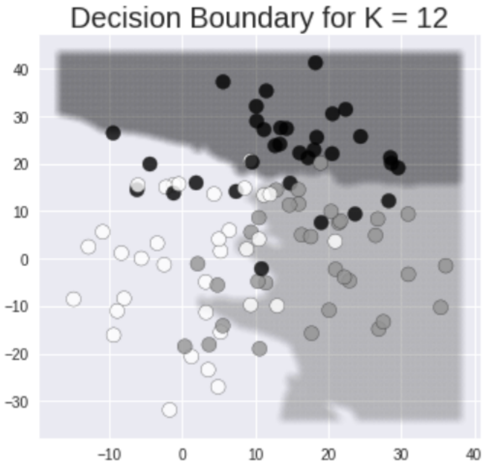

Decision Boundary for K = 9

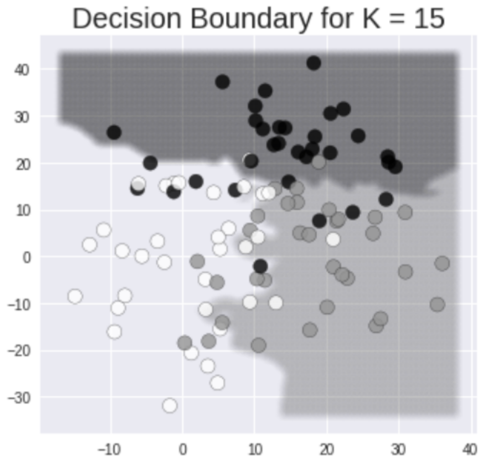

Decision Boundary for K = 10

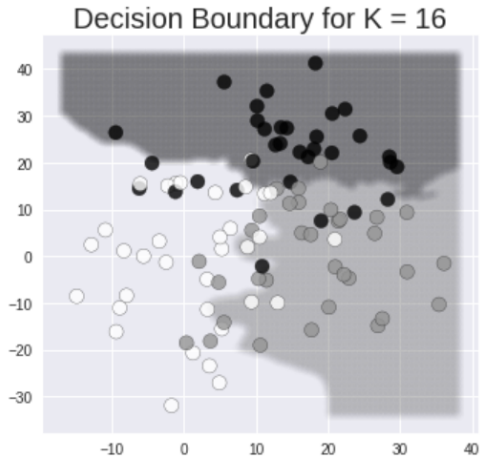

Decision Boundary for K = 13


Decision Boundary for K = 14

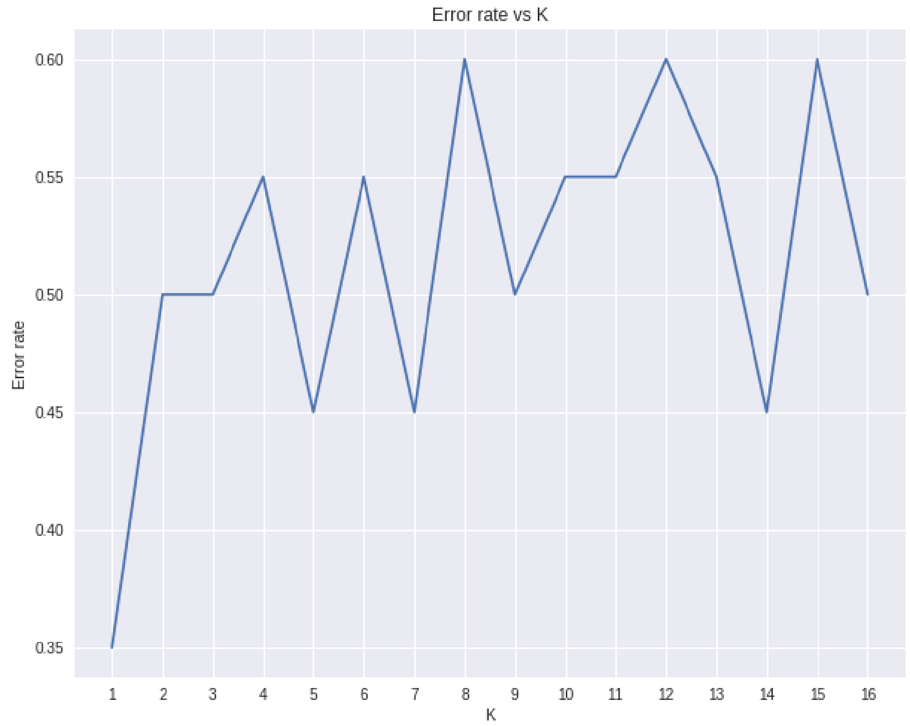

Decision Boundary for K = 11


Decision Boundary for K = 12


Decision Boundary for K = 15


Decision Boundary for K = 16

Error vs k chart.



The value k that yields in minimum test error is most suitable.

Advantages

- Quite easy to understand and implement the algorithm.
- The output of a prediction can be explained based on its neighbours. This adds to interpretability of the K-NN model.

Limitations

- For large training set, K-NN can be time consuming, since all computations are performed at runtime.
- K-NN is sensitive to redundant or irrelevant features since all features are used to compute distance between two points.
- On significantly difficult tasks, it can be out performed by other techniques such as SVM, Neural Networks.