CSC411: Assignment #1 (bonus)

Due on Monday, February 5, 2018

Chawla Dhruv

February 4, 2018

Investigating K Nearest Neightbours: Increasing training set performance on increasing K

K Nearest Neighbours works on the assumption that data points with same labels are going to be bunched together (are going to be close together in terms of how we are measuring distance in out model)

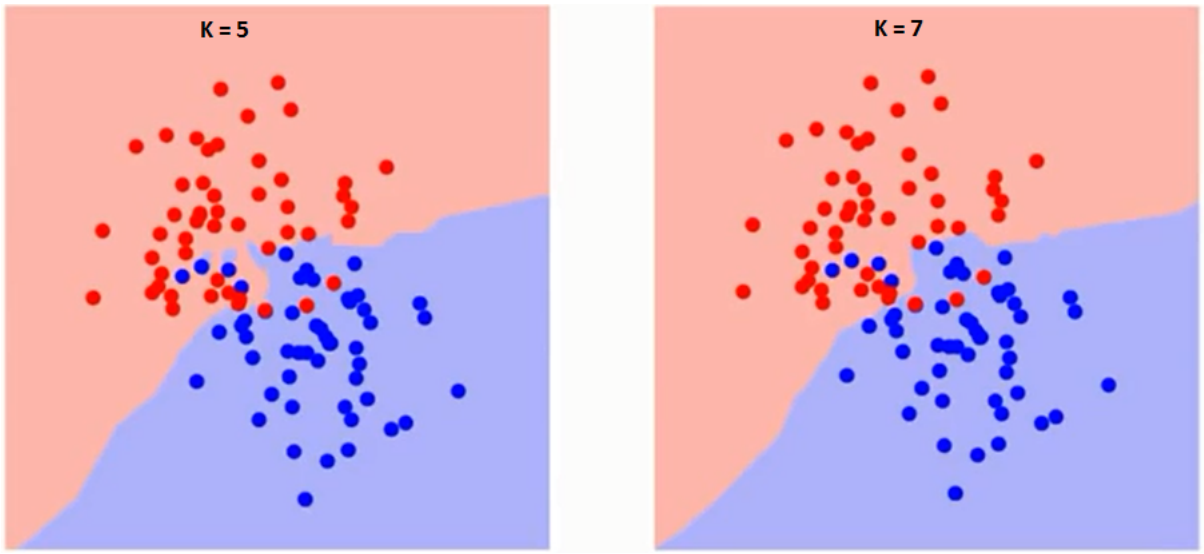

Figure 1: K Nearest Neighbours on a dataset (from analyticsvidhya.com)

As can be observed, the algorithm seems to be working pretty well as the red and blue dot seem to be clustering in specific areas.

Also, observe that in $K = 7$ case, the performance on the training set seems to be lower than on the $K = 5$ case (more blue and red dots are being classified as the other way around). This can also be seen on the error vs K chart produced (more error means worse performance on the training set).

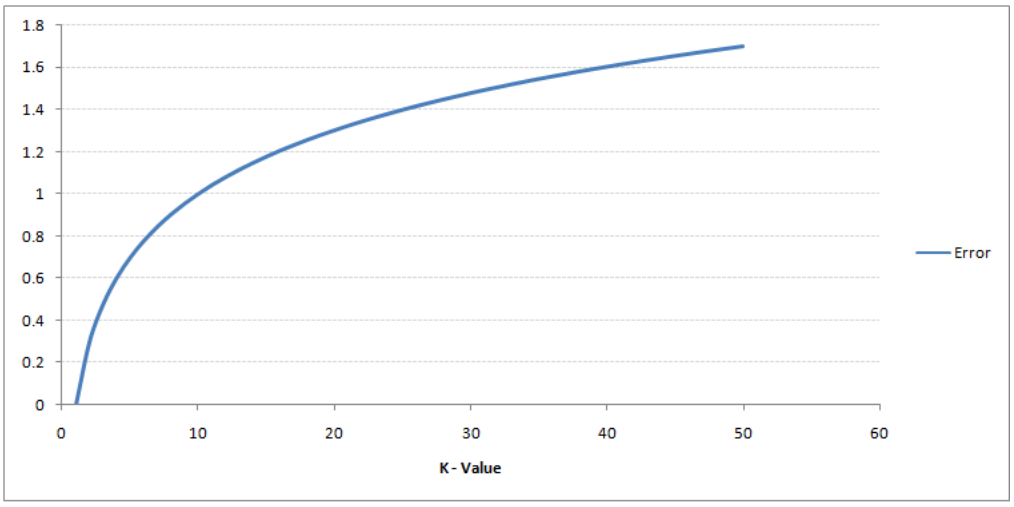

Figure 2: Error vs K (from analyticsvidhya.com)

This can be generally understood as the more you increase, the more influence surrounding neighbours of

different labels have and they can swing the classifier to their favour (especially on boundary data points). As an extension, performance on $K = 1$ is always a 100% as the only neighbour we are considering is the data point itself.

In this investigation, I have tried to produce a data set which does not have data points cluster together in terms of Euclidian/Cartesian distance (although they would in terms of Polar/Radial distances). My dataset can be described as concentric circular rings with alternating labels.

This data set is produced in `generate_data.py`.

```
def generate_data():
    """Generate random data resembling alternating concentric rings

    Returns
    X: list(list(x, y)) : coordinates of the data point
    y: list(either 0 or 1) : label of the data point
    """
    X = []
    y = []

    label = 1

    np.random.seed(5)

    for i in range(3):
        for j in range(8):
            for k in range(40):
                x_coord = (i - 0.5 + np.random.random_sample()) * np.cos(j/8 * 2*np.pi)
                y_coord = (i - 0.5 + np.random.random_sample()) * np.sin(j/8 * 2*np.pi)

                X += [[x_coord, y_coord]]
                y += [label]

        label = 1 - label

    return X, y
```

This data set challenges the underlying assumption of K Nearest Neighbours and produces a dip in training set performance at around $K = 120$. Performance picks up soon after as the outermost ring starts to influence the inner circle classification. The performance finally settles at roughly 67%, which is due to the fact that there are twice as many data points labelled as 1 than labelled as 0.

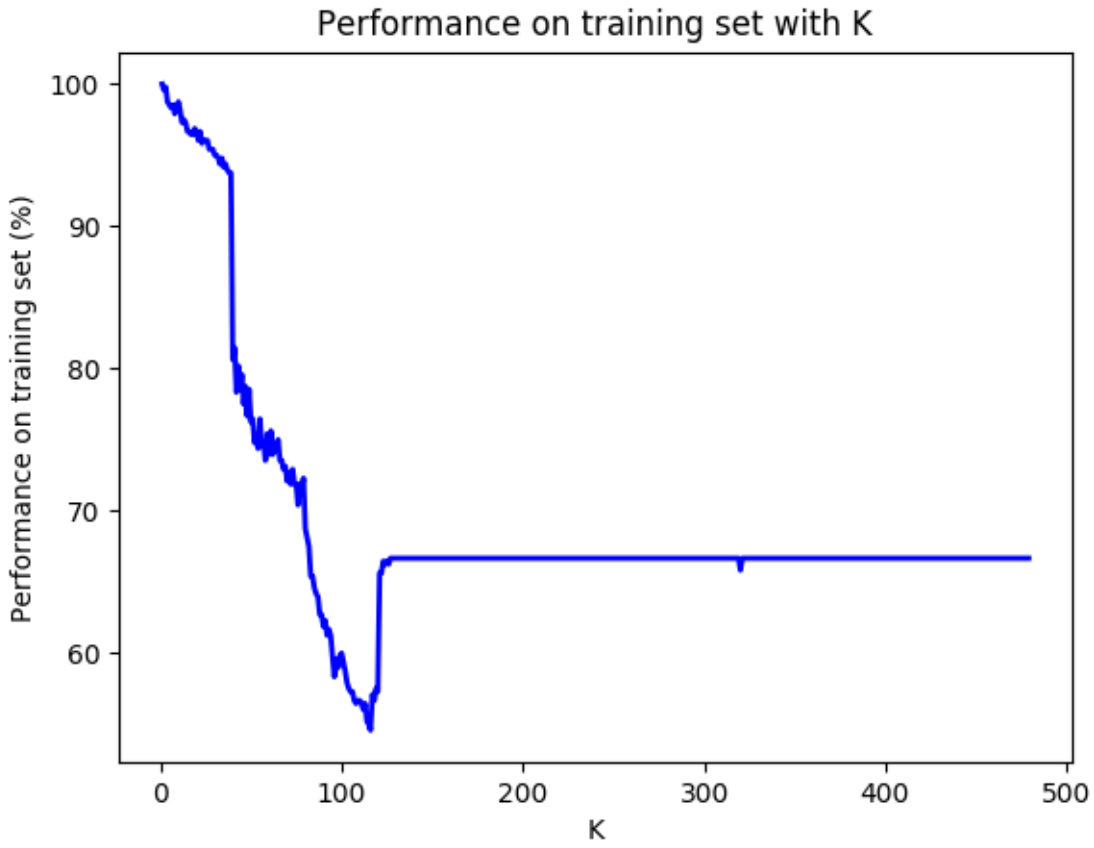

Figure 3: Performance vs K on training set