k-NN Classifier

**Table of contents**

1. Introduction
2. Need of a k-NN Algorithm
3. Working of k-NN
4. Curse of Dimensionality
5. Deciding k-Value
6. Model Training & Predicting Results
7. Pros & Cons of k-NN

# Introduction

* k-Nearest Neighbour is based on Supervised Learning technique.
* k-NN algorithm assumes the similarity between the new sample and available data then put the new sample into the category that is most similar to the available categories.
* k-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using k-NN algorithm. And that’s why it is called **Lazy Learner Algorithm.**
* k-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

# Need of a k-NN Algorithm

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a k-NN algorithm. With the help of k-NN, we can easily identify the category or class of a particular dataset.

# Working of k-NN

k-NN follows below mentioned steps for generating model:

**Step-1:** Select the number k of the nearest neighbor.

**Step-2:** Calculate the Euclidean distance of k number of neighbors.



**Step-3:** Take the k nearest neighbors as per the calculated Euclidean distance.

**Step-4:** Among these k neighbors, count the number of the data points in each category.

**Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

**Step-6:** Model is ready.

# Curse of Dimensionality

The curse of dimensionality means that k-NN predicts better results with a minimum no. of features. When the number of features increases, then it requires more data. When there’s more data, it creates an overfitting problem because no one knows which piece of noise will contribute to the model.

# Deciding k-Value

# There is no particular way to determine the best value for "k", so we need to try some values to find the best out of them. The most preferred value for k is 5.

# A very low value for k such as k=1 or k=2, can be noisy and lead to the effects of outliers in the model.

# Large values for k are good, but it may find some difficulties.

* Generally, we choose as an odd number if the number of classes is even. we can also check by generating the model on different values of k and by checking their performance.

# Pros & Cons of k-NN

|  |  |
| --- | --- |
| Pros | Cons |
| The training phase of K-nearest neighbor classification is much faster compared to other classification algorithms | The testing phase of K-nearest neighbor classification is slower and costlier in terms of time and memory. It requires large memory for storing the entire training dataset for prediction. |
| There is no need to train a model for generalization, that’s why KNN is known as the simple and instance-based learning algorithm. | KNN requires scaling of data because it uses the Euclidean distance.  Euclidean distance is sensitive to magnitudes. |
| KNN can be useful in case of nonlinear data. It can be used with the regression problem. | The features with high magnitudes will weight more than features with low magnitudes. |
| Output value for the object is computed by the average of k closest neighbors value. | KNN also not suitable for large dimensional data. |

# Model Training & Predicting Results

### For **k = 3**

1. **Classification Report:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | support |
| 0 [<=50K] | 0.82 | 0.87 | 0.84 | 6821 |
| 1 [>50K] | 0.51 | 0.42 | 0.46 | 2228 |
| Accuracy |  |  | 0.76 | 9049 |
| Macro avg. | 0.67 | 0.64 | 0.65 | 9049 |
| Weighted avg. | 0.74 | 0.76 | 0.75 | 9049 |

1. **Confusion Matrix:**

[[5923 898]

[1290 938]]

True Positives(TP) = 5923

True Negatives(TN) = 938

False Positives(FP) = 898

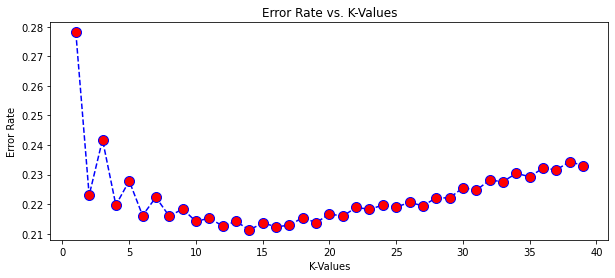
False Negatives(FN) = 1290

The confusion matrix shows 5923 + 938 = 6861 correct predictions and 898 + 1290 = 1088 incorrect predictions.

1. **Accuracy:**

Accuracy is: 0.75

Visualizing Error-rate while k-values is in range of (1, 40)



Here we can see that for around k = 15, error rate is almost minimal, so we will predict results again,

For **k = 15**

1. **Classification Report:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | support |
| 0 [<=50K] | 0.79 | 0.97 | 0.87 | 6821 |
| 1 [>50K] | 0.71 | 0.22 | 0.34 | 2228 |
| Accuracy |  |  | 0.79 | 9049 |
| Macro avg. | 0.75 | 0.60 | 0.61 | 9049 |
| Weighted avg. | 0.77 | 0.79 | 0.74 | 9049 |

1. **Confusion Matrix:**

[[6619 202]

[1731 497]]

True Positives(TP) = 6619

True Negatives(TN) = 497

False Positives(FP) = 202

False Negatives(FN) = 1731

The confusion matrix shows 6619 + 497 = 7116 correct predictions and 898 + 1290 = 2188 incorrect predictions.

1. **Accuracy:**

Accuracy is: 0.78

# Conclusion

While we considering k-value 3, accuracy is 75% while for k=15, accuracy is 78%