**2019911278**

**Berkay Can Özçelik**

**K-Nearest Neighbor Algorithm**

**Breast Cancer Diagnosis using K-Nearest Neighbor Algorithm**

**1.Introduction**

In this research we have a dataset of Breast Cancer patients with Malignant and Benign tumor.  
The main aim is to predict with the K-nearest neighbor algorithm whether is patient is having cancer (Malignant tumor) or not (Benign tumor). We have knowledge of 30 features which refers that the patient has cancer or not.

**2.Material and Method**

**A )**

In this research i used a dataset which has 33 Features. First of all i dropped the 2 columns which will affect our algorithm in a bad way cause they dont have any relation with the main data. The columns which are dropped are the “*id*” and “**Unnamed: 32”.**

**The classifier is the column named “**diagnosis**” which was defined as String M (** Malignant ) and B ( Benign ) to classify it in digit we convert the diagnosis value of M and B to a numerical value as

M (Malignant) = 1  
B (Benign) = 0

**We will discuss the affect of the other 30 features in the third step.**

**B)**

K-NN is a **non-parametric** and **lazy learning algorithm**. Non-parametric means there is no assumption for underlying data distribution i.e. the model structure determined from the dataset.

It is called Lazy algorithm because it does not need any training data points for model generation. All training data is used in the testing phase which makes training faster and testing phase slower and costlier.

K-Nearest Neighbor (K-NN) is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure.

In K-NN classification, the output is a **class membership**. An object is classified by a plurality vote of its neighbors, with the object being assigned to the **class most common among its k nearest neighbors** (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

To determine which of the K instances in the training dataset are most similar to a new input, **a distance measure is used**. For real-valued input variables, the most popular distance measure is the Euclidean distance.

# ****Advantages of K-NN :****

1. The K-NN algorithm is very easy to implement.
2. Nearly optimal in the large sample limit.
3. Uses local information, which can yield highly adaptive behavior.
4. Lends itself very easily to parallel implementation.

# ****Disadvantages of K-NN :****

1. Large storage requirements.
2. Computationally intensive recall.
3. Highly susceptible to the curse of dimensionality.

**C)**

**# In[77]:**

**import numpy as np**

**import pandas as pd**

**# visualisation**

**import matplotlib.pyplot as plt**

**#reading the excel file**

**data = pd.read\_csv(r"C:\Users\user\Desktop\data.csv")**

**data.head()**

**# In[78]:**

**#dropping columns which we dont need to classify**

**df = data.drop(['Unnamed: 32','id'],axis=1)**

**# In[79]:**

**#convert the diagnosis value of M and B to a numerical value**

**def diagnosis\_value(diagnosis):**

**if diagnosis == 'M':**

**return 1**

**else:**

**return 0**

**df['diagnosis'] = df['diagnosis'].apply(diagnosis\_value)**

**df**

**# In[93]:**

**#plotting the data**

**import seaborn as sns**

**sns.lmplot(x = 'radius\_mean', y = 'texture\_mean', hue = 'diagnosis', data = df)**

**# In[99]:**

**#plotting the data**

**sns.lmplot(x ='smoothness\_mean', y = 'compactness\_mean',**

**data = df, hue = 'diagnosis')**

**# In[103]:**

**#Splitting the data as train and test (test size = 33% of the whole data)**

**X = np.array(df.iloc[:, 1:])**

**y = np.array(df['diagnosis'])**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X, y, test\_size = 0.33, random\_state = 42)**

**# In[104]:**

**#appyling KNN algorithm**

**from sklearn.neighbors import KNeighborsClassifier**

**knn = KNeighborsClassifier(n\_neighbors = 13)**

**knn.fit(X\_train, y\_train)**

**# In[105]:**

**#calculating KNN score**

**knn.score(X\_test, y\_test)**

1. **Results and Discussion**

First of all we calculated the prediction score with then K value as 13. But i want to see the affect of

the neigbors so i re-compile the code with some more different numbers of neighbors. Results are

shown below :

|  |  |
| --- | --- |
| *Number of Neighbors* | *Prediction Score* |
| 3 | 0.9414893617021277 |
| 5 | 0.9521276595744681 |
| 13 | 0.9627659574468085 |
| 15 | 0.9680851063829787 |
| 30 | 0.9468085106382979 |
| 150 | 0.898936170212766 |

From this table we can easily see that the highest score is fort he k value = 15.

I decided to take care more about the affect of the other values so I started to change the test size which was 33% of the whole dataset.

|  |  |  |
| --- | --- | --- |
| Test Size ( as % ) | Number of Neighbors | *Prediction Score* |
| 20 | 3 | 0.9298245614035088 |
| 33 | 3 | 0.9414893617021277 |
| 50 | 3 | 0.9368421052631579 |
| 20 | 5 | 0.956140350877193 |
| 33 | 5 | 0.9521276595744681 |
| 50 | 5 | 0.9578947368421052 |
| 20 | 13 | 0.9736842105263158 |
| 33 | 13 | 0.9627659574468085 |
| 50 | 13 | 0.9473684210526315 |
| 20 | 15 | 0.9649122807017544 |
| 33 | 15 | 0.9680851063829787 |
| 50 | 15 | 0.9543859649122807 |

It was clearly to see how important the test size was. For a test size of 33% the best number of the K value was 15 but the K value changed to 13 when we took the test size to 20%.

After all these i started to drop some features from the dataset to see the effect of it in this algorithm here are some changes of the prediction score ( test size = 20% , K = 13 )

|  |  |  |
| --- | --- | --- |
| Dropped Feature | Prediction Score ( Before ) | Prediction Score ( After ) |
| radius\_mean | 0.9736842105263158 | 0.9736842105263158 |
| smoothness\_mean | 0.9736842105263158 | 0.9736842105263158 |
| symmetry\_worst | 0.9736842105263158 | 0.9736842105263158 |
| area\_worst | 0.9736842105263158 | 0.956140350877193 |

To be honest this part of the research was shocking cause just one parameter alone was changing the Prediction score and it was the parameter named “area\_worst”.

After i found it out i tried to take out more features at the same time and i could change some changes on the prediction score.

|  |  |  |
| --- | --- | --- |
| Dropped Features | Prediction Score ( Before ) | Prediction Score ( After ) |
| First 5 | 0.9736842105263158 | 0.9210526315789473 |
| Last 10 | 0.9736842105263158 | 0.9385964912280702 |

So i found out that these prediction depence on more than just one feature.

1. **References**

# K-Nearest Neighbors (KNN) Algorithm

[**https://www.geeksforgeeks.org/ml-kaggle-breast-cancer-wisconsin-diagnosis-using-knn/**](https://www.geeksforgeeks.org/ml-kaggle-breast-cancer-wisconsin-diagnosis-using-knn/)

# ML | Kaggle Breast Cancer Wisconsin Diagnosis using KNN

[**https://towardsdatascience.com/k-nearest-neighbors-knn-algorithm-bd375d14eec7**](https://towardsdatascience.com/k-nearest-neighbors-knn-algorithm-bd375d14eec7)

# Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python & R)

[**https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-**](https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-)**clustering/**