ရင်သား ကင်ဆာ ဖြစ် မဖြစ် k-NN model တည်ဆောက်၍ ခန့်မှန်းခြင်း

k-NN machine learning algorithm ဥပမာ-၁

k-NN သဘောတရားတွေကို ပြီးခဲ့တဲ့ post မှာ တင်ပြီးခဲ့ပါပြီ။ အခု k-NN သဘောတရားတွေကို Python code တွေ function တွေနဲ့ ပဲ ရေးထားတဲ့ k-NN algorithm ကို Python code တစ်ကြောင်းစီကို အဓိပ္ပာယ်နှင့် တကွ ရှင်းပြထားပါတယ်။

Code တွေနဲ့ programming အကြောင်းကို နားလည် သဘောပေါက်ဘို့ machine learning algorithm ရဲ့ သဘောနဲ့ ဘယ်လို မျိုးပြဿာကို ဖြေရှင်းဘို့ ကြိုးစားနေတယ်ဆိုတာကို အရင် ကြိုသိထားသင့် ပါတယ်။

Classification ပြဿနာ ကို ဖြေရှင်းဘို့ k-NN ကို သုံးမှာပါ။ supervised machine learning algorithm ဖြစ်သည့် k Nearest Neighbor (k-NN) algorithm ကို သုံးမှာပါ။

ရင်သား ကင်ဆာ ဖြစ်နေတယ် မဖြစ်ဘူး။ ဖြစ်နိုင်ခြေများတယ်။ မများဘူးဆိုတဲ့ ခန့်မှန်းချက်ကို ထုတ်ပေးမဲ့ k-NN algorithm ဖြစ်ပါတယ်။ အသုံးပြုမည့် ဒေတာတွေကို '/kaggle/input/breast-cancer-wisconsin-data/data.csv' ရနိုင်ပါတယ်။ အခု ဒေါင်းလုပ် လုပ်ထားတဲ့ ဖိုင်ထဲမှာ data.csv ကို ထည့်ထားပြီ ဖြစ်လို့ သီးခြား ဒေါင်းလုပ် လုပ်စရာမလိုပါဘူး။

ရင်သားကင်ဆာဖြစ်နိုင်တဲ့ အကြိတ်အဖုရဲ့ ပျမ်းပျှအချင်း (radius_mean), ပျမ်းပျှ ပတ်လည်အနား (perimeter_mean) ပျမ်းပျှဧရိယာ(area_mean) texture_mean, smoothness_mean, compactness_mean concavity_mean, concave points_mean စတဲ့ အချက်တွေ ပါဝင်ပြီး ကင်ဆာဖြစ်နေသူ ကင်ဆာ မဖြစ်သေးသူ ဆိုတဲ့ အချက်တွေလည်းပါတယ်။

ရင်သားကင်ဆာဖြစ်နိုင်တဲ့ အကြိတ်အဖုရဲ့ ပျမ်းပျှအချင်း (radius_mean), ပျမ်းပျှ ပတ်လည်အနား (perimeter_mean) ပျမ်းပျှဧရိယာ(area_mean)စတာတွေကို machine learning ဝေါဟာရနဲ့ feature လို့ ခေါ်ပါတယ်။ အဲဒါတွေက input (X) တွေပါ။

ကင်ဆာဖြစ်နေသူ ၊ ကင်ဆာ မဖြစ်သေးသူ ဆိုတဲ့ အချက်တွေက output (y)တွေပါ။ labeled output လုပ်လည်း ခေါ်ပါတယ်။

k-NN machine learning algorithm နှင့် ခန့်မှန်းဘို့ k-NN model တစ်ခုတည်ဆောက်ရမှာပါ။ k-NN model ကို train လုပ်ဘို့ နှင့် train လုပ်ပြီးတဲ့ k-NN model ကို test လုပ်ဘို့ အတွက်ဒေတာတွေကို နှစ်စုခွဲပါမယ်။

k-NN model ကို train လုပ်နေတဲ့အချိန်မှာ ထည့်ပေးတဲ့ (input)ဒေတာတွေကို Xtrain လို့ခေါ်ပြီးတော့ သူနဲ့ သက်ဆိုင်တဲ့ output ဒေတာတွေက ytrain ပါ။

k-NN model ဟာ ထည့်ပေးလိုက်တဲ့ Xtrain ဒေတာနဲ့ ytrain တွေမှ တစ်ဆင့် အကြိတ်အဖုဘယ်လောက်ကြီးရင် ရင်သားကင်ဆာ ဖြစ်ပြီ၊ ဖြစ်နိုင်တယ်ဆိုတဲ့ ခန့်မှန်းချက်တွေကို learn လုပ်ပါတယ်။ အဲဒါမျိုး ကွန်ပျူတာက သူဖာသာ သင်ယူလို့ learn လုပ်လို့ machine learning algorithm လို့ခေါ်တာပါ။

train လုပ်ပြီးတဲ့ k-NN model ကို test လုပ်ဘို့ အတွက် Xtest ကို ထည့်ပေးရင် k-NN model က ပြန်ပြီးတော့ သူရဲ့ ခန့်မှန်းချက်တွေကို ထုတ်ပေးပါတယ်။ အဲဒီ ခန့်မှန်းချက်တွေ ဘယ်လောက်မှန်သလဲဆိုတာကို ရှုပြီး အဖြေ ytest ဒေတာတွေနဲ့ နှိုင်းယှဉ်ကြည့်ပြီး k-NN model ရဲ့ accuracy ကို တွက်ယူနိုင်ပါတယ်။

Algorithm ကို coding စလုပ်ကြပါစို့။ ပထမဦးစွာ pandas နှင့် numpy ကို import လုပ်ပါတယ်။

In [1]:
```
import pandas as pd
import numpy as np
```

ရှိပြီးသား ဒေတာများထဲမှ train လုပ်ရန် ဒေတာ နှင့် test လုပ်ရန် ဒေတာများ အဖြစ် နှစ်စုခွဲသည်။ Input (X) နှင့် output (y) နှစ်မျိုးစလုံးကို နှစ်စုခွဲသည်။ collections package ထဲမှ Counter ကို import လုပ်ပါတယ်။

In [2]:
```
#Using to split data randomly
from sklearn.model_selection import train_test_split
#Using to get most common element from list
from collections import Counter
```

ဒေတာဖိုင်မှ ဒေတာတွေကို ဖတ်ဘို့ `def readData()` function တစ်ခုတည်ဆောက်ပါတယ်။ ။ (reading in the data) ID Column နှင့် Unnamed ကော်လံတွေကို ခဏ ဖယ်ထုတ်ထားပါတယ်။ dropping လုပ်ပါတယ်။ null data ကို ဖယ်ထုတ်ထားပါတယ်။

In [3]:
```
def readData():
    data = pd.read_csv('data.csv')
    # Original Data file '/kaggle/input/breast-cancer-wisconsin-data
/data.csv'
    #Drop Id as it has no influence on diagnosis, and Unnamed for Na
n Vals
    data = data.drop(['id', 'Unnamed: 32'], axis = 1)
    data = data.dropna()
    return data
```

train set နှင့် test set ခွဲရန် အတွက် `def splitData()` function တစ်ခုတည်ဆောက်သည်။ data များ အားလုံးထည့်ပေးသည်။ ခွဲမည့် test set ၏ အရွယ်အစား(testSize) ကို ထည့်ပေးသည်။ ခွဲပြီးသား ဒေတာများ Xtrain, Xtest, ytrain, ytest ကို numpy arrays အဖြစ်သို့ ပြောင်းပြီး function မှ ပြန်ပေးသည်။

Split data into a train and a test set. Converting them to numpy arrays, and returning them.

In [4]:
```
def splitData(data, testSize):
    y = data['diagnosis'].to_numpy()
    data = data.drop(['diagnosis'], axis = 1)
    X = data.to_numpy()
    Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=
testSize)
    return Xtrain, Xtest, ytrain, ytest
```

ကင်ဆာ ဖြစ်နေသူ၏ အကြိတ်အဖု အရွယ်အစား၊ ဇရိယာ စသည်တို့နှင့် အနီးစပ်ဆုံးတူသူ၊ တူညီမှု(similarity) အများဆုံးရှိသူ ကို ကင်ဆာ ဖြစ်နေသူ ဖြစ်နိုင်သူဟု model က ဆုံးဖြတ်ခန့်မှန်းသည်။

တူညီမှု(similarity)ကို ဂရပ်ဖော်တွင် ဖော်ပြ ပြောဆိုလျှင်၊ သခ်ၤာနည်းဖြင့် ဖော်ပြလိုလျှင် အကွာအဝေး (distance)ကို အသုံးပြု ရသည်။ အနီးစပ်ဆုံး (distance အနည်းဆုံး) ဖြစ်လျှင် အတူညီဆုံး ဖြစ်သည်။

ပွိုင့် နှစ်ပွိုင့် အကြားအကွာဝေး (distance between two points)ကို တိုင်းယူနိုင်သည့် နည်းများအားလုံးကို သုံးနိုင်သည်။

Euclidean distance ကို သုံး၍ အကွာအဝေးဖြင့် ဆုံးဖြတ်သောကြောင့် feature များအားလုံးသည် အညီအမျှ အရေးပါသည်။ (Euclidean distance treats each feature as equally important) ။ တစ်နည်းအားဖြင့် feature များအားလုံးသည် အညီအမျှ အရေးပါမှသာ Euclidean distance ကို အသုံးပြုနိုင်သည်။

(၁) Equal weight to all attribute , only if scale attribute are similar.

(၂) Scale attribute to equal range.

(၃) Classes are spherical in shape. စတဲ့အချက်တွေနဲ့ ပြည့်စုံပါမှ simple Euclidean distance equation ကို အသုံးပြုနိုင်ပါသည်။

မည်မျှ တူညီမှု(similarity)ရှိသည်ကို ဆုံးဖြတ်ရန် `def euclidean()` တစ်ခုတည်ဆောက်သည်။ Xtrain, Xtestval တို့ ထည့်ပေးသည်။

xtestval နှင့် Xtrain value တစ်ခုစီအကြား euclidean distance ကို တွက်ယူပြီး list (distances) တစ်ခုပြုလုပ်သည်။ ထို list ကို အနီးစပ်ဆုံး lowest (best) မှ အဝေးဆုံး (highest distances) ကို အစဉ်အတိုင်း စဉ်သည်။ (to make sure we keep track of the index of the Xtrain value to look up the corresponding ytrain value (the actual class of the Xtrain value).)

In [5]:
```
def euclidean(Xtrain, Xtestval):
    distances = []
    for i in range(len(Xtrain)):
        #euclidean equation
        distance = np.sqrt(np.sum(np.square(Xtestval-Xtrain[i])))
        distances.append([distance, i])
    return sorted(distances)
```

ခန့်မှန်းသည့် predictions function တစ်ခုတည်ဆောက်သည်။ Xtrain, ytrain, Xtestval, k တို့ function သို့ ထည့်ပေးရသည်။ k အရေအတွက် မည်မျှသုံးမည်ကို သတ်မှတ်ပေးရသည်။ Xtrain, Xtestval တို့၏ euclidean distance ကိုတွက်သည်။ predict list တစ်ခုဆောက်သည်။

It does this by taking the index of the k smallest distances (which is already sorted in the distances list so really the first k values of the list), and finding the value of the corresponding index of the y train values (the actual class of those xtrain features). We will then return the most common value in the predict list, which is our predicted class of the Xtest features.

In [6]:
```
def predict(Xtrain, ytrain, Xtestval, k):
    distances = euclidean(Xtrain, Xtestval)
    predict = []
    for i in range(k):
        predict.append(ytrain[distances[i][1]])

    return Counter(predict).most_common(1)[0][0]
```

`def KNN()` function တည်ဆောက်သည်။ Xtest ဒေတာများအားလုံးအတွက် ကင်ဆာ ဖြစ်မဖြစ် ခန့်မှန်းပေးသည်။ (predicted class) ဤ ဥပမာအတွက် k=3 ဖြစ်သည်။

In [7]:
```
def KNN(Xtrain, Xtest, ytrain):
    predictions = []
    for i in range(len(Xtest)):
        predictions.append(predict(Xtrain, ytrain, Xtest[i], 3))

    return predictions
```

Model မှ ထုတ်ပေးသည့် ခန့်မှန်းချက်များ ဘယ်လောက်မှန်သလဲဆိုတာကို ရှိပြီး အဖြေ ytest ဒေတာတွေနဲ့ နှိုင်းယှဉ်ကြည့်ပြီး k-NN model ရဲ့ accuracy ကို တွက်ယူသည်။

In [8]:
```
def accuracy(ytest, predictions):
    correct = 0
    for i in range(len(predictions)):
        if(predictions[i]==ytest[i]):
            correct += 1

    score = (correct/len(ytest))*100
    return score
```

main

Function များ တည်ဆောက်ပြီး၍ algorithm စတင်ရန် ဒေတာဖတ်သည်။ ဖတ်ပြီး ဒေတာများ စစ်ရန် ပထမဆုံး (၅)ခုကို ထုတ်ယူကြည့်သည်။

In [9]:
```
data = readData()
data.head(5)
```

Out[9]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	M	17.99	10.38	122.80	1001.0	0.11840
1	M	20.57	17.77	132.90	1326.0	0.08474
2	M	19.69	21.25	130.00	1203.0	0.10960
3	M	11.42	20.38	77.58	386.1	0.14250
4	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 31 columns

ကော်လံ (၃၁)ခု ရှိသည်။ ဒေတာများကို Xtrain, Xtest, ytrain, ytest စသည်ဖြင့် ခွဲသည်။ ခွဲပြီး ဒေတာများ၏ row အရေအတွက် နှင့် ကော်လံ အရေအတွက်ကို ပြန်ကြည့်သည်။

In [10]:
```
Xtrain, Xtest, ytrain, ytest = splitData(data, 0.2)
Xtrain.shape ,
print('Xtrain : ',Xtrain.shape)
print('Xtest : ',Xtest.shape)
print('ytrain : ',ytrain.shape)
print('ytest : ',ytest.shape)
```

```
Xtrain :  (455, 30)
Xtest :  (114, 30)
ytrain :  (455,)
ytest :  (114,)
```

Xtrain, Xtest, ytrain စသည်တို့ကို k-NN model ထဲသို့ ထည့်၍ ခန့်မှန်းချက် ကို ထုတ်ယူသည်။ ytest သည် အဖြေ ဖြစ်သောကြောင့် ထည့်ပေးရန် မလိုပါ။

`predictions[:10]` ဖြင့် ပထမဆုံး ခန့်မှန်းချက် (၁၀)ကို ကြည့်သည်။

In [11]:
```
predictions = KNN(Xtrain, Xtest, ytrain)
predictions[:10]
```

Out[11]:
```
['B', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'M', 'M']
```

ရှိပြီးသား အဖြေများနှင့် နှိုင်းယှဉ်ရန် ytest မှ ပထမဆုံး (၁၀)ခုကို ကြည့်ရန် `ytest[:10]` ကို သုံးသည်။

In [12]:
```
ytest[:10]
```

Out[12]:
```
array(['M', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'M', 'M'], dtype=ob
ject)
```

Model မှ ထုတ်ပေးသည့် ခန့်မှန်းချက် ပထမဆုံး (၁၀) ခုနှင့် ရှိပြီးသားအဖြေ ytest မှ ပထမဆုံး (၁၀)ခုတို့ လုံးဝ တူညီသည်။ ထို့နောက် model ၏ accuracy ကိုတွက်သည်

In [13]:
```
print(accuracy(ytest, predictions))
```

```
96.49122807017544
```

k=3 နှင့် testSize = 0.2 အတွက် accuracy မှ 90-95% ခန့် ဖြစ်သည်။