k-NN Example-3

k-Nearest Neighbors Classifier Example

ပန်းသီး လိမ္မော်သီး ရွေးချယ်ခြင်း

ပန်းသီး နှင့် လိမ္မော်သီး နှစ်မျိုးတို့၏ အလေးချိန်(weight)နှင့် အချင်း(diameter)တို့ကို တိုင်းတာထားသည့် ဒေတာကို အသုံးပြု၍ k-NN classifier model တစ်ခု တည်ဆောက်ပြီး ဒေတာအသစ်များ ထည့်ကာ ပန်းသီးဖြစ်မည် လိမ္မော်သီး ဖြစ်မည်ဟု ခန့်မှန်းချက် ထုပ်ပေးသည့် k-NN ဥပမာ တစ်ခု ဖြစ်သည်။

ပထမ ဦးစွာ ဒေတာဖတ်ရန် `pandas package` မှ `read_csv function` ကို import လုပ်သည်။

```
In [1]: from pandas import read_csv
```

ဒေတာများကို code ထဲတွင် variable တစ်ခု အဖြစ် သိမ်းဆည်းထားသည်။ numpy arrays အနေဖြင့် သိမ်းဆည်းရန်အတွက် numpy pakage ကို import လုပ်သည်။ numpy pakage တစ်ခုလုံးကို import လုပ်ရန်လိုအပ်သည်။

```
In [2]: import numpy as np
```

`read_csv function` ဖြင့် apple-orange-dataset.csv ဖိုင်ထဲမှ ဒေတာများကို ပြီး တစ်ခု pandas DataFrame တည်ဆောက်သည်။ DataFrame ၏ နာမည်မှာ apple_orange ဖြစ်သည်။

```
In [3]: apple_orange = read_csv('apple-orange-dataset.csv')
```

ဒေတာများကို စစ်ဆေးခြင်း

row အရေအတွက်နှင့် column အရေအတွက်ကို ကြည့်ရန်

```
In [4]: apple_orange.shape
```

```
Out[4]: (419, 3)
```

```
In [5]: apple_orange.size
```

```
Out[5]: 1257
```

ဒေတာများထဲမှ ပထမဆုံး row (၅)ခုကို ကြည့်ရှုရန် `.head(5)` ကို သုံးသည်။ label ၊ weight နှင့် diameter ဟူ၍ ကော်လံ(၅)ခု ပါရှိသည်။ label ကော်လံထဲတွင် 0 နှင့် 1 တန်ဖိုးများသာပါဝင်သည်။ 1 သည် ပန်းသီး ဖြစ်ပြီး 0 သည် လိမ္မော်သီးများ ဖြစ်ကြသည်။

In [6]:
```
apple_orange.head(5)
```

Out[6]:

	label	weight	diameter
0	1	136	10.557432
1	1	192	11.292246
2	1	112	10.120368
3	1	185	8.786955
4	1	129	10.070343

ဒေတာများထဲမှ နောက်ဆုံး row (၅)ခုကို ကြည့်ရှုရန် `.head(5)` ကို သုံးသည်။

In [7]:
```
apple_orange.tail(5)
```

Out[7]:

	label	weight	diameter
414	1	150	18.262581
415	1	122	9.105190
416	1	58	8.688009
417	1	50	6.881814
418	1	60	5.987806

labels ထုတ်ယူခြင်း(extracting)

ဒေတာ ကော်လံတွင် label ဟု နာမည်ပေးပြီးသားဖြစ်သည်။ label ကော်လံအောက်တွင် ရှိသည် 0 နှင့် 1 သည် label value များ ဖြစ်ကြသည်။ label များသည် output များ ဖြစ်သောကြောင့် y အမည် ဖြင့် ဒေတာများကို သိမ်းဆည်းသည်။

In [8]:
```
y = apple_orange['label']
```

In [9]:
```
y.head()
```

Out[9]:
```
0    1
1    1
2    1
3    1
4    1
Name: label, dtype: int64
```

Extracting Features

ပန်းသီး နှင့် လိမ္မော်သီး နှစ်မျိုးတို့၏ အလေးချိန်(weight)နှင့် အချင်း(diameter)တို့သည် feature များ ဖြစ်သည်။ ၎င်းတို့သည် interdependent input variable များ ဖြစ်ကြသည်။ feature များသည်။ ၎င်းတို့ကို 2D array ဖြင့် သိမ်းဆည်းထားမည်။ Input ဖြစ်သောကြောင့် X ဟု အမည်ပေးသည်။

In [10]:
```
features = ['weight','diameter']
X = apple_orange[features]
```

In [11]:
```
X.head()
```

Out[11]:

	weight	diameter
0	136	10.557432
1	192	11.292246
2	112	10.120368
3	185	8.786955
4	129	10.070343

ဒေတာများကို ဂရပ်ဖြင့် ပုံဖော်ကြည့်ခြင်း:(Visualizing the data)

apple-orange-dataset.csv ဖိုင်ထဲမှ ဒေတာများကို 2D ဆွဲမည်။ `apple_orange[apple_orange['label'] == 1` တွင် apple_orange DataFrame ထဲ ['label'] ကော်လံထဲမှာ တန်ဖိုးများဖြစ်သည်။

ထိုတန်ဖိုးများသည် 1 ဖြစ်လျှင် `== 1` ပန်းသီးများဖြစ်ကြသောကြောင့် apples ဟုသည် ပန်းသီးများသာ ပါသည့် သီးခြား DataFrame တစ်ခု တည်ဆောက်သည်။ `apples.shape` သည် 220 rows ဖြစ်သောကြောင့် မူလဒေတာ ထဲတွင် ပန်းသီး ၂၂၀ လုံး ပါဝင်သည်။

In [12]:
```
apples = apple_orange[apple_orange['label'] == 1 ]
apples.shape
```

Out[12]:
```
(220, 3)
```

`oranges.shape` သည် 199 rows ဖြစ်သောကြောင့် မူလ ဒေတာ ထဲတွင် လိမ္မော်သီး ၁၁၉ လုံးပါဝင်သည်။

In [13]:
```
oranges = apple_orange[apple_orange['label'] == 0 ]
oranges.shape
```

Out[13]:
```
(199, 3)
```

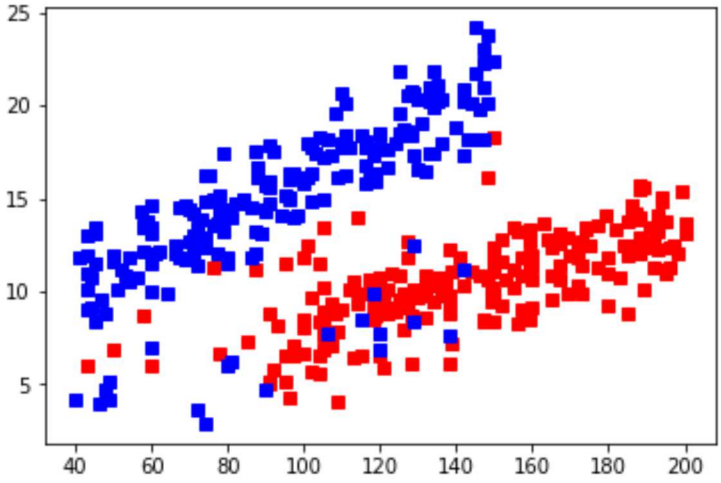
pattern ကို ရှာဖွေရန် အကောင်းဆုံးနည်းမှာ ဂရပ်ပုံများ ဆွဲကြည့်ခြင်း ဖြစ်သည်။ matplotlib.pyplot ကို import လုပ်သည်။ matplotlib မှ subpackage ဖြစ်သည်သည့် pyplot မှ plot(x,y)function ဖြင့် ဂရပ်ပုံ ဆွဲသည်။

In [14]:
```
import matplotlib.pyplot as plt
```

အလေးချိန်(weight)ကို X ဝင်ရိုး(X-axis)တွင် ထား၍ အချင်း(diamter) ကို Y ဝင်ရိုး(Y-axis)တွင် ထားကာ ဂရပ်ဆွဲသည်။

In [15]:
```
plt.plot(apples['weight'],apples['diameter'], 'rs')
plt.plot(oranges['weight'],oranges['diameter'],'bs')
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x93d0198>]
```



Classifier ရွေးချယ်ခြင်း

ပန်းသီး နှင့် လိမ္မော်သီး များကို ခွဲခြားရန်အတွက် classifier တစ်ခုလိုအပ်သည်။ K-Nearest Neighbour classifier ကို အသုံးပြုမည်။ K-Nearest Neighbour ၏ သဘောတရားများကို [http://www.acmv.org/MachineLearning/kNN/Introduction_to_K_Nearest_Neighbors.pdf (http://www.acmv.org/MachineLearning/kNN/Introduction_to_K_Nearest_Neighbors.pdf)](http://www.acmv.org/MachineLearning/kNN/Introduction_to_K_Nearest_Neighbors.pdf) တွင် ရှင်းပြခဲ့ပြီး ဖြစ်သည်။

`sklearn.neighbors` မှ `KNeighborsClassifier` ကို အသုံးပြုမည်။

In [16]:
```
from sklearn.neighbors import KNeighborsClassifier
```

K-Nearest Neighbour machine learning model တည်ဆောက်သည်။

In [17]:
```
model = KNeighborsClassifier(n_neighbors=3)
```

Train Test Split

Train လုပ်ရန် ဒေတာ နှင့် Test လုပ်ရန် ဒေတာ ခွဲသည်။

- အရေးကြီးဆုံးအချက်မှာ machine learning model တည်ဆောက်ရန်အတွက် ဒေတာများသည် လုံလောက်အောင် များပြားရမည်။

training data နှင့် testing data အဖြစ် (၂)စု ခွဲမည်။ training data များတွင်လည်း train input ဒေတာ X_train နှင့် output ဒေတာ y_train ဟူ၍ (၂)မျိုးရှိသည်။ testing data များတွင်လည်း input ဒေတာ X_test နှင့် output ဒေတာ y_test ဟူ၍ (၂)မျိုးရှိသည်။

sklearn မှ cross_validation subpackageတွင် ရှိပြီးသား function ဖြစ်သည်သည့် train_test_split() ကို အသုံးပြုသည်။

In [18]:
```
from sklearn.model_selection import train_test_split
```

train_test_split သည် array များစွာကို return အဖြစ် ပြန်ပေးသည့် စပါယ်ရှယ် function တစ်ခုဖြစ်သည်။
train_test_split function argument သည် test_size ဖြစ်သည်။
`test_size=0.4` ဖြစ်သောကြောင့် X_test နှင့် y_test သည် X နှင့် y ၏ ၄၀% ဖြစ်သည်။
X_train နှင့် y_train သည် X နှင့် y ၏ ၆၀ % ဖြစ်သည်။

In [19]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, y,
                                                    test_size=0.4,
                                                    random_state=2)
```

`fit() function` ဖြင့် model ထဲသို့ X_train ဒေတာများ နှင့် Y_train များ ထည့်ပေးသည်။

In [20]:
```
model.fit(X_train, Y_train)
```

Out[20]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minko
wski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

Fit လုပ်ပြီးသား model ထဲသို့ X_test ဒေတာများ နှင့် Y_test များ ထည့်ပေးကာ ခန်းမှန်းချက်များ ထုတ်ယူသည်။
`predict()` function ကို သုံးသည်။

In [21]:
```
y_pred = model.predict(X_test)
```

In [22]:
```
y_pred
```

Out[22]:
```
array([1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0,
1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0, 0,
       0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1,
1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0,
1, 1,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1], dtype=int64)
```

In [23]:
```
Y_test.head()
```

Out[23]:
```
146    1
223    0
193    1
305    0
188    1
Name: label, dtype: int64
```

တိကျမှု(Accuracy)

Model မှ ထုတ်ပေးသည့် ခန့်မှန်းချက်များ၏ တိကျမှန်ကန်မှု(accuracy of predictions)ကို စစ်ဆေးရန် sklearn မှ metrics ဆိုသည့် subpackage မှ accuracy_score function ကို အသုံးပြုသည်။

```
In [24]: from sklearn.metrics import accuracy_score
```

၉၄% တိကျမှန်ကန်မှု ရှိသည်။

```
In [25]: 100*accuracy_score(y_pred, Y_test)
Out[25]: 94.04761904761905
```

တိကျမှန်ကန်မှု ပိုကောင်းအောင် ဘယ်လို လုပ်မလဲ

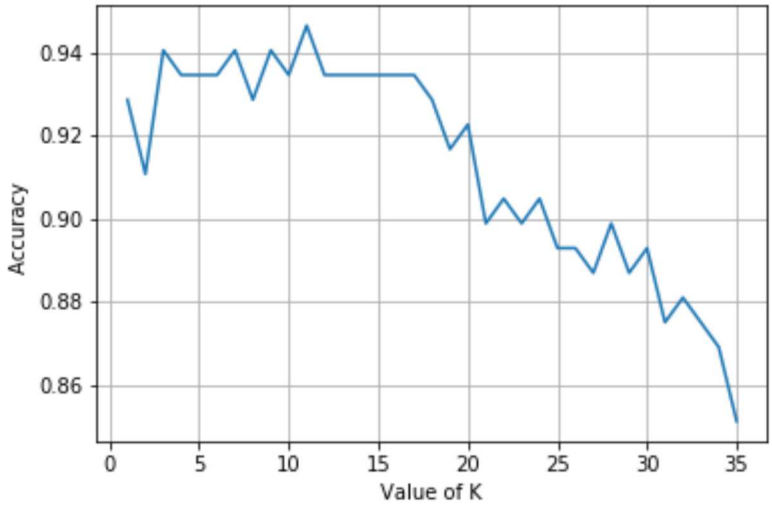
k-Nearest Neighbor တွင် k အရေအတွက် (value of k) သည် တိကျမှန်ကန်မှု အတွက် အရေးကြီးသည့် အချက်ဖြစ်သည်။ k အရေအတွက် (value of k) ပြောင်းလဲခြင်းဖြင့် တိကျမှန်ကန်မှု (accuracy)% ပြောင်းလဲသည်။ k အရေအတွက် (value of k)ကို (၁၀) မှ (၁၅) အတွင်း အမျိုးမျိုး ပြောင်းလဲ၍ higher accuracy ရအောင် စမ်းသပ်သည်။

တိကျမှု အကောင်းဆုံးပေးနိုင်သည့် best_k ကို ရှာဖွေရန်

k အရေအတွက် 1 မှ စ၍ 36 အထိ for loop ဖြင့် ပတ်ပြီး best_accuracy တန်ဖိုးကို ရှာသည်။ k အရေအတွက် နှင့် ၎င်းနှင့်သက်ဆိုင်သည့် accuracy တန်ဖိုး တို့ကို ပုံဆွဲသည်။

In [26]:
```
print("To test for best value of K and plotting the graph ")
acc = []
best_k = 1
best_accuracy = 0
for k in range(1,36):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,Y_train)
    acc.append(accuracy_score(knn.predict(X_test), Y_test))
    if(acc[k-1] > best_accuracy):
        best_accuracy = acc[k-1]
        best_k = k
plt.plot(range(1,36), acc)
plt.xlabel("Value of K")
plt.ylabel("Accuracy ")
plt.grid()
plt.show()
print("Best value for k is "+str(best_k))
print("Highest accuracy is "+str(best_accuracy))
```

```
To test for best value of K and plotting the graph
```




```
Best value for k is 11
Highest accuracy is 0.9464285714285714
```

k အရေအတွက် (၁၁) သည် တိကျမှန်ကန်မှု အမြင့်ဆုံး(highest accuracy) ဖြစ်သည်။

In [ ]:
```
```