In [ ]:
```
#how to find best values for hyper-parameters
>GridSearch
>RandomizedSearch

Note:both techniques are based on cross validation
```

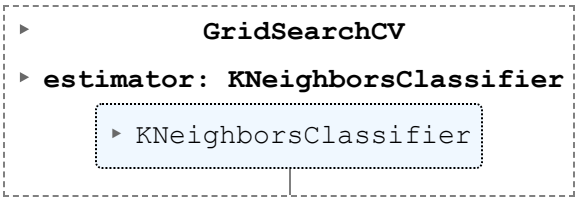
In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv("f:/dataset/classification/fruits.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [4]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
```

In [5]:
```
model=KNeighborsClassifier()
gs=GridSearchCV(model,param_grid={'n_neighbors':[2,3,4,5,6,7,8]},cv=5)
gs.fit(X,y)
```

Out[5]:



In [6]:
```
gs.best_score_
```

Out[6]:
```
0.86
```

In [7]:
```
gs.best_params_
```

Out[7]:
```
{'n_neighbors': 2}
```

In [8]:
```
model=KNeighborsClassifier()
gs=GridSearchCV(model,param_grid={'n_neighbors':[2,3,4,5,6,7,8],'metric':['manhattan','e
gs.fit(X,y)
```

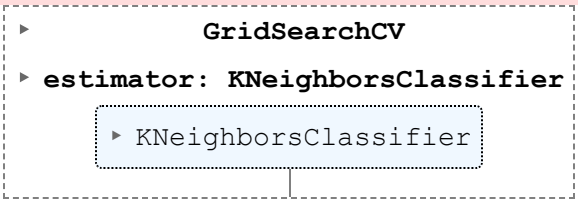
```
C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:824: U
serWarning: Scoring failed. The score on this train-test partition for these parameters
will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.p
y", line 813, in _score
    scores = scorer(estimator, X_test, y_test)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\metrics\_scorer.py", line 52
7, in __call__
    return estimator.score(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\base.py", line 705, in score
    return accuracy_score(y, self.predict(X), sample_weight=sample_weight)
                             ^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.p
y", line 249, in predict
    probabilities = self.predict_proba(X)
                    ^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.p
y", line 327, in predict_proba
    probabilities = ArgKminClassMode.compute(
```

```
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\metrics\_pairwise_distances_r
eduction\_dispatcher.py", line 590, in compute
    unique_labels=np.array(unique_labels, dtype=np.intp),
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'Apple'

  warnings.warn(
C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:824: U
serWarning: Scoring failed. The score on this train-test partition for these parameters
will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.p
y", line 813, in _score
    scores = scorer(estimator, X_test, y_test)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\metrics\_scorer.py", line 52
7, in __call__
    return estimator.score(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\base.py", line 705, in score
    return accuracy_score(y, self.predict(X), sample_weight=sample_weight)
                             ^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.p
y", line 249, in predict
    probabilities = self.predict_proba(X)
                    ^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.p
y", line 327, in predict_proba
    probabilities = ArgKminClassMode.compute(
                    ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\metrics\_pairwise_distances_r
eduction\_dispatcher.py", line 590, in compute
    unique_labels=np.array(unique_labels, dtype=np.intp),
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'Apple'

  warnings.warn(
C:\Users\Ducat\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:976: UserW
arning: One or more of the test scores are non-finite: [0.86 0.86 0.86 0.86 0.86 0.86  n
an 0.86 0.86 0.86 0.86 0.86 0.86 0.86]
  warnings.warn(
```

Out[8]:

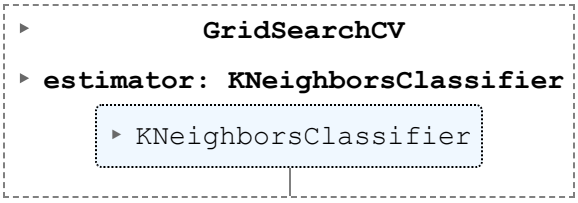


In [9]:
```
df.FruitName=df.FruitName.map({'Apple':0,'Banana':1})
```

In [12]:
```
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

model=KNeighborsClassifier()
gs=GridSearchCV(model,param_grid={'n_neighbors':[3,4,5,6,7,8],'metric':['manhattan','euc
gs.fit(X,y)
```

Out[12]:



In [13]:
```
gs.best_params_
```

Out[13]:
```
{'metric': 'manhattan', 'n_neighbors': 3}
```

In [17]:
```
from sklearn.model_selection import RandomizedSearchCV
model=KNeighborsClassifier()
rs=RandomizedSearchCV(model,param_distributions={'n_neighbors':[3,4,5,6,7,8],'metric':['
rs.fit(X,y)
```

Out[17]:

▸ RandomizedSearchCV

▸ estimator: KNeighborsClassifier

▸ KNeighborsClassifier

In [18]:
```
rs.best_params_
```

Out[18]:
```
{'n_neighbors': 7, 'metric': 'manhattan'}
```

In [16]:
```
rs.best_score_
```

Out[16]:
```
0.86
```

In [ ]:
```
#LogisticRegression
>Note:it is classification algorithm but internally uses linear model hence
name include Regression term.

>it also gives you probabilty regarding prediction.
```

In [19]:
```
from sklearn.linear_model import LogisticRegression
```

In [20]:
```
model=LogisticRegression()
model.fit(X,y)
```

Out[20]:

▾ LogisticRegression

LogisticRegression()

In [21]:
```
model.predict([[3.1,50]])
```

Out[21]:
```
array([1], dtype=int64)
```

In [22]:
```
model.predict_proba([[3.1,50]])
```

Out[22]:
```
array([[0.14627037, 0.85372963]])
```

In [31]:
```
model.predict_proba([[4.2,76]])
```

Out[31]:
```
array([[0.47294084, 0.52705916]])
```

In [32]:
```
model.predict([[4.2,76]])
```

Out[32]:
```
array([1], dtype=int64)
```

In [33]:
```
from sklearn.datasets import load_iris
```

```
iris=load_iris()
```

In [36]:
```
X=iris.data
y=iris.target

model=LogisticRegression(max_iter=200)
model.fit(X,y)
print(model.predict_proba([[.7,.1,2.5,1.1]]))
print(model.predict([[.7,.1,2.5,1.1]]))
```

```
[[0.75585864 0.24060393 0.00353743]]
[0]
```

maths behind LogisticRegression

- first it builds a linear model to find coefs and intercept
- then convert this linear model to non linear sothat it can perform classification.
- logit fun is used to convert linear part to non linear
- sigmoid & softmax are used as logit funs.
- with binary classification,sigmoid is used and with multiclass softmax is used.

In [37]:
```
df=pd.read_csv("f:/dataset/classification/fruits.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
model=LogisticRegression()
model.fit(X,y)
```

Out[37]:
```
▾ LogisticRegression
LogisticRegression()
```

In [38]:
```
model.coef_
```

Out[38]:
```
array([[-0.98526751, -0.02200081]])
```

In [39]:
```
model.intercept_
```

Out[39]:
```
array([5.91852746])
```

In [40]:
```
sample=[3,50]
z=-0.98526751*3+-0.02200081*50+5.91852746
print(z)
```

```
1.8626844299999998
```

In [41]:
```
model.decision_function([sample]) #returns linear part of sample
```

Out[41]:
```
array([1.86268452])
```

In [42]:
```
import numpy as np
```

In [48]:
```
p_1=1/(1+np.exp(-z))
print(p_1)
if p_1<=.5:
    print("Apple")
else:
    print("banana")
```

```
0.8656095335016416
banana
```

In [47]:
```
model.predict_proba([sample])
```

Out[47]:
```
array([[0.13439046, 0.86560954]])
```

In [49]:
```
model.predict([sample])
```

Out[49]:
```
array(['Banana'], dtype=object)
```

In [50]:
```
p_0=1-p_1
print(p_0)
```

```
0.1343904664983584
```

In [51]:
```
1/(1+np.exp(z))
```

Out[51]:
```
0.13439046649835837
```

In [ ]: