Model Validation Iris Data using K-Neighbor

In [1]:

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
```

In [2]:

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=1) #point 1
```

In [3]:

```
model.fit(X, y)
y_model = model.predict(X) #point 2
```

In [4]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y, y_model) #point 3
```

Out[4]:

```
1.0
```

โมเดลเราดีขนาดที่สามารถทำนายถูกต้อง 100% เลยหรือ หรือ เราทำอะไรผิดไปหรือเปล่า

1) Holdout

In [5]:

```
from sklearn.model_selection import train_test_split
# Holdout => split the data with 50% in each set1 แบ่งว่าจะเป็น Training และ Testing data เท่าไ
X1, X2, y1, y2 = train_test_split(X, y, random_state=0, train_size=0.5)
#X1 => X_train
#X2 => X_test
#y1 => y_train
#y2 => y_test


# fit the model on one set of data
model.fit(X1, y1) #still using n_neighbors=1

# evaluate the model on the second set of data
y2_model = model.predict(X2)
accuracy_score(y2, y2_model)
```

Out[5]:

```
0.9066666666666666
```

In [6]:

```
y1_model = model.fit(X2, y2).predict(X1)

y2_model = model.fit(X1, y1).predict(X2)

accuracy_score(y1, y1_model), accuracy_score(y2, y2_model)
```

Out[6]:

```
(0.96, 0.9066666666666666)
```

เมื่อใช้ Holdout technique แล้ว ความถูกต้องอยู่ที่ 96 หรือ 90.6 ขึ้นอยู่กับว่าเราเลือกอะไรเป็น Training และ Testing

2) Cross-validation

In [7]:

```
from sklearn.model_selection import cross_val_score
cross_val_score(model, X, y, cv=5) # cv=5 : 5-fold cross validation
```

Out[7]:

```
array([0.96666667, 0.96666667, 0.93333333, 0.93333333, 1.        ])
```

In [8]:

```
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=LeaveOneOut())
#cv=LeaveOneOut : Leave-One-Out cross-validator
#Each sample is used once as a test set (singleton) while the remaining samples form the tr
scores
```

Out[8]:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [9]:

```
scores.shape
```

Out[9]:

```
(150,)
```

In [10]:

```
scores.mean()
```

Out[10]:

```
0.96
```