In [8]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_iris
dataset = load_iris()
X = dataset.data
y = dataset.target
```

In [2]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rand
```

In [3]:
```
print("X_train dim :",X_train.shape)
print("y_train dim :",y_train.shape)
print("X_test dim :",X_test.shape)
print("y_test dim :",y_test.shape)
```

```
X_train dim : (105, 4)
y_train dim : (105,)
X_test dim : (45, 4)
y_test dim : (45,)
```

In [17]:
```
knn1 = KNeighborsClassifier(n_neighbors = 11,p = 1)
knn2 = KNeighborsClassifier(n_neighbors = 6,p = 2)
knn3 = KNeighborsClassifier(n_neighbors = 5,p = float('inf'))
```

In [14]:
```
#L1
knn1.fit(X_train,y_train)
knn1.predict(X_test)
knn1.score(X_test, y_test)
```

Out[14]:
```
0.9777777777777777
```

In [19]:
```
#Euclidean Distance
knn2.fit(X_train,y_train)
knn2.predict(X_test)
knn2.score(X_test, y_test)
```

Out[19]:
```
0.9777777777777777
```

In [20]:
```
#Chebyshev
knn3.fit(X_train,y_train)
knn3.predict(X_test)
knn3.score(X_test, y_test)
```

Out[20]: 0.9555555555555556

In [11]:
```
from sklearn.model_selection import GridSearchCV
#create a dictionary of all values we want to test for n_neighbors
param_grid = {"n_neighbors": np.arange(1, 25)}
#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn1, param_grid, cv=5)
#fit model to data
knn_gscv.fit(X, y)
knn_gscv.best_params_
```

Out[11]: {'n_neighbors': 11}

In [15]:
```
from sklearn.model_selection import GridSearchCV
#create a dictionary of all values we want to test for n_neighbors
param_grid = {"n_neighbors": np.arange(1, 25)}
#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5)
#fit model to data
knn_gscv.fit(X, y)
knn_gscv.best_params_
```

Out[15]: {'n_neighbors': 6}

In [16]:
```
from sklearn.model_selection import GridSearchCV
#create a dictionary of all values we want to test for n_neighbors
param_grid = {"n_neighbors": np.arange(1, 25)}
#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn3, param_grid, cv=5)
#fit model to data
knn_gscv.fit(X, y)
knn_gscv.best_params_
```

Out[16]: {'n_neighbors': 5}

In [ ]:
```
# The socres of them are a bit overfitted due to small size of dataset
```