In [5]:
```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
from sklearn.metrics import confusion_matrix,precision_score,recall_score,classification
```

In [6]:
```
iris=load_iris()
X=iris.data
y=iris.target
```

In [20]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1)
model=KNeighborsClassifier()
model.fit(X_train,y_train)
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.9553571428571429
1.0
```

In [8]:
```
pred_train=model.predict(X_train)
confusion_matrix(y_train,pred_train) #recommended
```

Out[8]:
```
array([[37,  0,  0],
       [ 0, 31,  3],
       [ 0,  2, 39]], dtype=int64)
```

In [9]:
```
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [11]:
```
31/34
```

Out[11]:
```
0.9117647058823529
```

In [12]:
```
37+31+39
```

Out[12]:
```
107
```

In [13]:
```
107/112
```

Out[13]:
```
0.9553571428571429
```

In [15]:
```
print(classification_report(y_train,pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       0.94      0.91      0.93        34
           2       0.93      0.95      0.94        41

    accuracy                           0.96       112
   macro avg       0.96      0.95      0.96       112
weighted avg       0.96      0.96      0.96       112
```

In [ ]:
```
#Model deployment
--->saving model to a file
--->hosting model to a server(installing to server) sothat it can be used for unseen pre
```

In [ ]:
```
#Picling & Unpickling
#Pickling------->prcocess to converting python object to
```

```
                #binary stream and then saving it to a file.
#Unpickling----->reverse process of pickling

#joblib module
```

In [16]:
```
import joblib
```

In [22]:
```
joblib.dump(model,"f:/iris_model.pkl")
```

Out[22]:
```
['f:/iris_model.pkl']
```

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

In [21]:
```
model.predict([[2.5,50]])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[21], line 1
----> 1 model.predict([[2.5,50]])

File ~\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.py:234, in KNeighbo
rsClassifier.predict(self, X)
    218 """Predict the class labels for the provided data.
    219
    220 Parameters
   (...)
    229     Class labels for each data sample.
    230 """
    231 if self.weights == "uniform":
    232     # In that case, we do not need the distances to perform
    233     # the weighting so we do not compute them.
--> 234     neigh_ind = self.kneighbors(X, return_distance=False)
    235     neigh_dist = None
    236 else:

File ~\anaconda3\Lib\site-packages\sklearn\neighbors\_base.py:806, in KNeighborsMixin.kn
eighbors(self, X, n_neighbors, return_distance)
    804         X = _check_precomputed(X)
    805     else:
--> 806         X = self._validate_data(X, accept_sparse="csr", reset=False, order="C")
    808 n_samples_fit = self.n_samples_fit_
    809 if n_neighbors > n_samples_fit:

File ~\anaconda3\Lib\site-packages\sklearn\base.py:588, in BaseEstimator._validate_data
(self, X, y, reset, validate_separately, **check_params)
    585     out = X, y
    587 if not no_val_X and check_params.get("ensure_2d", True):
--> 588     self._check_n_features(X, reset=reset)
    590 return out

File ~\anaconda3\Lib\site-packages\sklearn\base.py:389, in BaseEstimator._check_n_featur
es(self, X, reset)
    386     return
    388 if n_features != self.n_features_in_:
--> 389     raise ValueError(
    390         f"X has {n_features} features, but {self.__class__.__name__} "
    391         f"is expecting {self.n_features_in_} features as input."
    392     )

ValueError: X has 2 features, but KNeighborsClassifier is expecting 4 features as input.
```

In [ ]: