```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import KFold, cross_val_score, train_test_split
```

```
iris_data = pd.read_csv('IRIS.csv')
iris_data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
iris_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
X = iris_data[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].values
X[0:5]
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

```
y = iris_data['species'].values
y[0:5]
```

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa'], dtype=object)
```

```
X = preprocessing.StandardScaler().fit(X).transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

```
from sklearn import svm
svm_model = svm.SVC(kernel = 'linear')
svm_model.fit(X_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

```
yhat_svm = svm_model.predict(X_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
k = 5
knn_model = KNeighborsClassifier(n_neighbors = 5).fit(X_train, y_train)
```

```
yhat_knn = knn_model.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test, yhat_svm)
```

```
array([[19,  0,  0],
       [ 0, 12,  1],
       [ 0,  0, 13]])
```

```
from sklearn.metrics import f1_score
f1_score(y_test, yhat_svm, average='weighted')
```

```
0.9777448559670783
```

```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, knn_model.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat_knn))
```

```
Train set Accuracy:  0.9523809523809523
Test set Accuracy:  1.0
```

```
from sklearn.model_selection import cross_val_score
# For SVM
svm_model_linear = svm.SVC(kernel = 'linear')
score_linear = cross_val_score(svm_model_linear, X, y, cv=5)
print(score_linear.mean())

svm_model_poly = svm.SVC(kernel = 'poly')
score_poly = cross_val_score(svm_model_poly, X, y, cv=5)
print(score_poly.mean())

svm_model_rbf = svm.SVC(kernel = 'rbf')
score_rbf = cross_val_score(svm_model_rbf, X, y, cv=5)
print(score_rbf.mean())

svm_model_sigmoid = svm.SVC(kernel = 'sigmoid')
score_sigmoid = cross_val_score(svm_model_sigmoid, X, y, cv=5)
print(score_sigmoid.mean())
```

```
0.9666666666666668
0.9266666666666665
0.9666666666666666
0.9
```

```
#For KNN
knn_model_4 = KNeighborsClassifier(n_neighbors = 4)
score_4 = cross_val_score(knn_model_4, X, y, cv = 4)
print(score_4.mean())

knn_model_5 = KNeighborsClassifier(n_neighbors = 5)
score_5 = cross_val_score(knn_model_5, X, y, cv = 5)
print(score_5.mean())

knn_model_6 = KNeighborsClassifier(n_neighbors = 6)
score_6 = cross_val_score(knn_model_6, X, y, cv = 6)
print(score_6.mean())

knn_model_7 = KNeighborsClassifier(n_neighbors = 7)
score_7 = cross_val_score(knn_model_7, X, y, cv = 7)
print(score_7.mean())
```

```
0.9464793741109531
0.96
0.9533333333333333
0.9529993815708101
```