In [1]:
```
import sklearn
print(sklearn.__version__)
```

```
1.3.2
```

In [2]:
```
from sklearn import datasets
print(datasets.__all__)
```

```
['clear_data_home', 'dump_svmlight_file', 'fetch_20newsgroups', 'fetch_20newsgroups_vectorized', 'fetch_lfw_pairs',
'fetch_lfw_people', 'fetch_olivetti_faces', 'fetch_species_distributions', 'fetch_california_housing', 'fetch_covtyp
e', 'fetch_rcv1', 'fetch_kddcup99', 'fetch_openml', 'get_data_home', 'load_diabetes', 'load_digits', 'load_files', '
load_iris', 'load_breast_cancer', 'load_linnerud', 'load_sample_image', 'load_sample_images', 'load_svmlight_file',
'load_svmlight_files', 'load_wine', 'make_biclusters', 'make_blobs', 'make_circles', 'make_classification', 'make_ch
eckerboard', 'make_friedman1', 'make_friedman2', 'make_friedman3', 'make_gaussian_quantiles', 'make_hastie_10_2', 'm
ake_low_rank_matrix', 'make_moons', 'make_multilabel_classification', 'make_regression', 'make_s_curve', 'make_spars
e_coded_signal', 'make_sparse_spd_matrix', 'make_sparse_uncorrelated', 'make_spd_matrix', 'make_swiss_roll']
```

In [3]:
```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn import svm                # its used for regression or classification
```

In [4]:
```
# Loading the dataset
x, y = load_wine(return_X_y=True)     # return X is always capital

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0)
```

Train size:

- 0.8 mean 80% of data means ration 80 , 20

In [5]:
```
x_train , x_test, y_train, y_test = train_test_split(x,y , random_state=0 ,train_size=0.8)
```

In [6]:
```
x.shape
```

Out[6]:
```
(178, 13)
```

In [7]:
```
y.shape
```

Out[7]:
```
(178,)
```

In [8]:
```
print(x_train.shape)
```

```
(142, 13)
```

In [9]:
```
print(y_train.shape)
```

```
(142,)
```

SVC for classification >>> initilize the classifier >>> algorithm

In [10]:
```
# Creating an SVM classifier
classifier_svm = svm.SVC()
```

Training

In [11]:
```
# Training the classifier >>> first step train then testing
classifier_svm.fit(x_train, y_train)
```

Out[11]:
```
SVC
SVC()
```

Testing

In [12]:
```
svm_accuracy= classifier_svm.score(x_test, y_test)
```

In [13]:
```
print(svm_accuracy)
```

```
0.7777777777777778
```

In [14]:
```
print(svm_accuracy * 100)
```

```
77.77777777777779
```

Testing using svm

In [15]:
```
svm_accuracy_test = classifier_svm.score(x_test, y_test)
```

In [16]:
```
print(svm_accuracy_test*100)
```

```
77.77777777777779
```

Training using svm

In [17]:
```
svm_accuracy_train = classifier_svm.score(x_train, y_train)
```

In [18]:
```
print(svm_accuracy_train*100)
```

```
71.83098591549296
```

In [19]:
```
print(round(svm_accuracy_train,2)*100)
```

```
72.0
```

Cross validation

In [20]:
```
 # for cross validation >> no of instances according to class
from sklearn.model_selection import cross_val_score
```

In [21]:
```
# 5 ka cross validation show hoga >>> only for testing accuracies
cv_svm = cross_val_score(classifier_svm, x,y, cv=5)
print (cv_svm)
```

```
[0.63888889 0.61111111 0.63888889 0.68571429 0.74285714]
```

In [22]:
```
print(cv_svm.mean())
```

```
0.6634920634920635
```

In [23]:
```
print(cv_svm.std())
```

```
0.04636170738133653
```

In [24]:
```
# Create a DataFrame for the results
import pandas as pd
results_df = pd.DataFrame({
    'Metric': ['Testing Accuracy', 'Training Accuracy', 'Mean CV Accuracy', 'Std CV Accuracy',"cv_svm"],
    'Value': [svm_accuracy_test, svm_accuracy_train, cv_svm.mean(), cv_svm.std(),cv_svm]
})

# Display the results table
results_df
```

Out[24]:

	Metric	Value
0	Testing Accuracy	0.777778
1	Training Accuracy	0.71831
2	Mean CV Accuracy	0.663492
3	Std CV Accuracy	0.046362
4	cv_svm	[0.6388888888888888, 0.6111111111111112, 0.638...

a

In [ ]: