INDIAN INSTITUTE OF TECHNOLOGY MADRAS

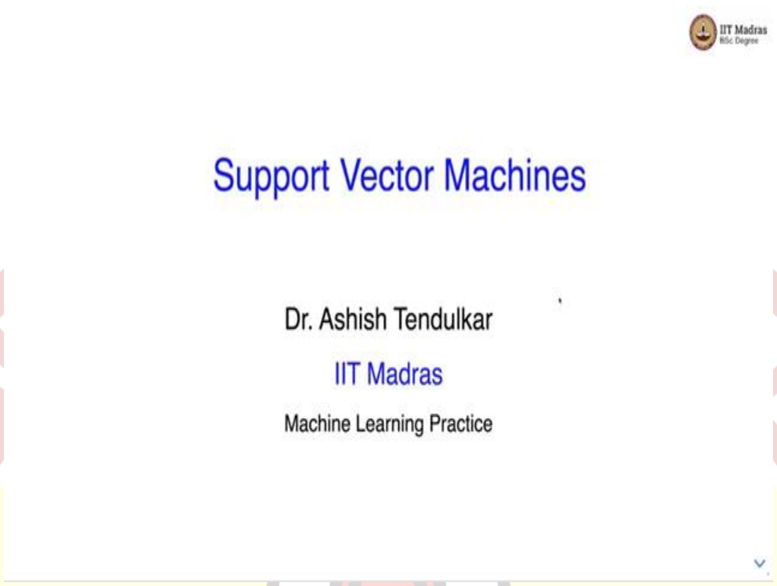
सिद्धिर्भवति कर्मजा

IIT Madras

ONLINE DEGREE

Machine Learning Practice
Professor Doctor Ashish Tendulkar
Indian Institute of Technology, Madras
SVM in scikit-learn

(Refer Slide Time: 0:12)



Support Vector Machines

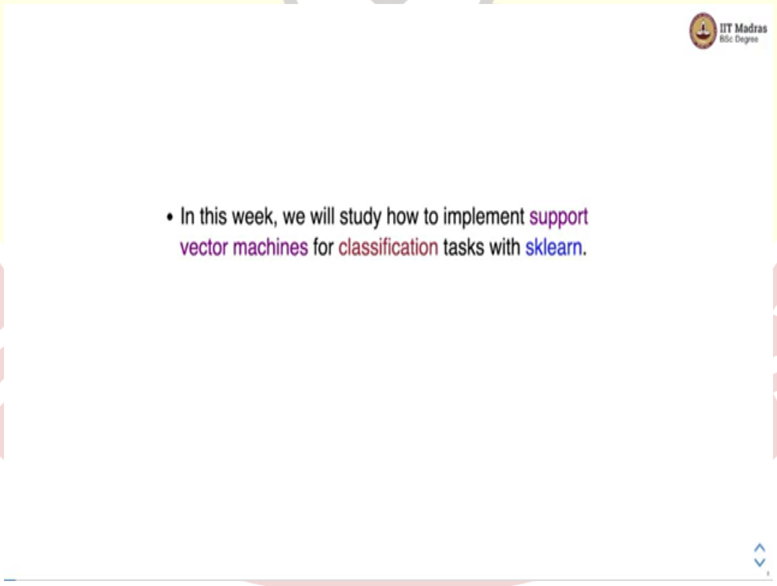
Dr. Ashish Tendulkar

IIT Madras

Machine Learning Practice

Namaste, welcome to the next video of Machine Learning Practice course.

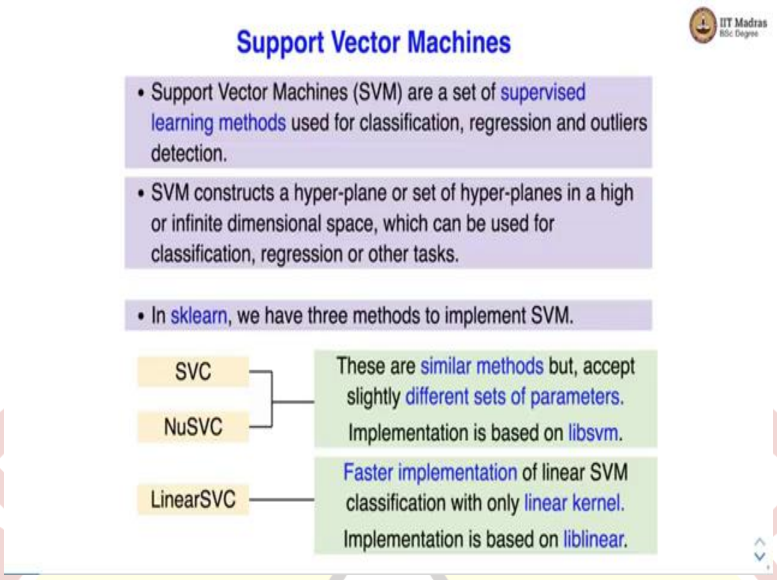
(Refer Slide Time: 0:18)



- In this week, we will study how to implement support vector machines for classification tasks with sklearn.

In this video we will study how to implement support vector machines for classification task with sklearn.

(Refer Slide Time: 0:25)



Support Vector Machines

- Support Vector Machines (SVM) are a set of supervised learning methods used for classification, regression and outliers detection.
- SVM constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks.
- In sklearn, we have three methods to implement SVM.

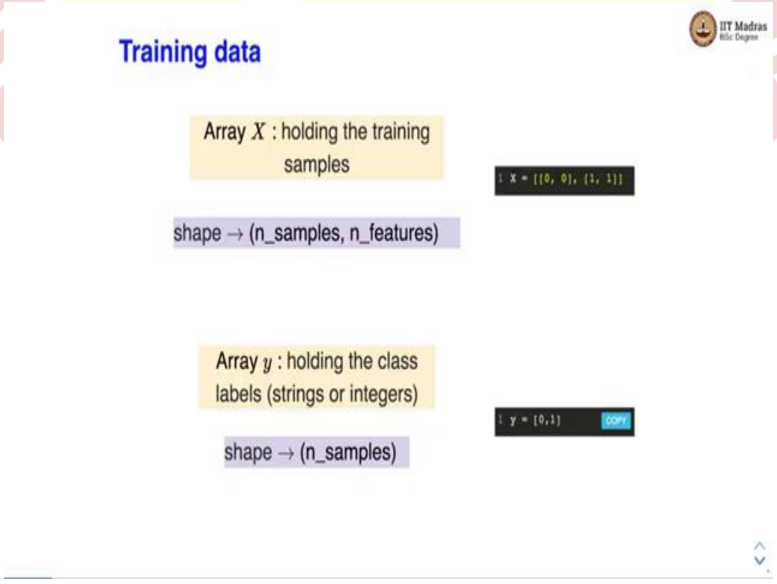
SVC, NuSVC: These are similar methods but, accept slightly different sets of parameters. Implementation is based on libsvm.

LinearSVC: Faster implementation of linear SVM classification with only linear kernel. Implementation is based on liblinear.

Support Vector Machines are a set of supervised learning methods used for classification, regression and outliers detection. SVM constructs a hyperplane or a set of hyperplanes in high dimensional or infinite dimensional space which can be used for classification regression and other tasks.

In sklearn SVM is implemented with three methods, one is SVC, second is no SVC and third is linear SVC. The first two methods are similar but they accept slightly different set of parameters, they are implemented based only based libsvm. Linear SVC on other hand is a faster implementation of linear SVM, it uses only linear kernel and the implementation is based on liblinear library.

(Refer Slide Time: 1:20)



Training data

Array X : holding the training samples

shape → (n_samples, n_features)

```
X = [[0, 0], [1, 1]]
```

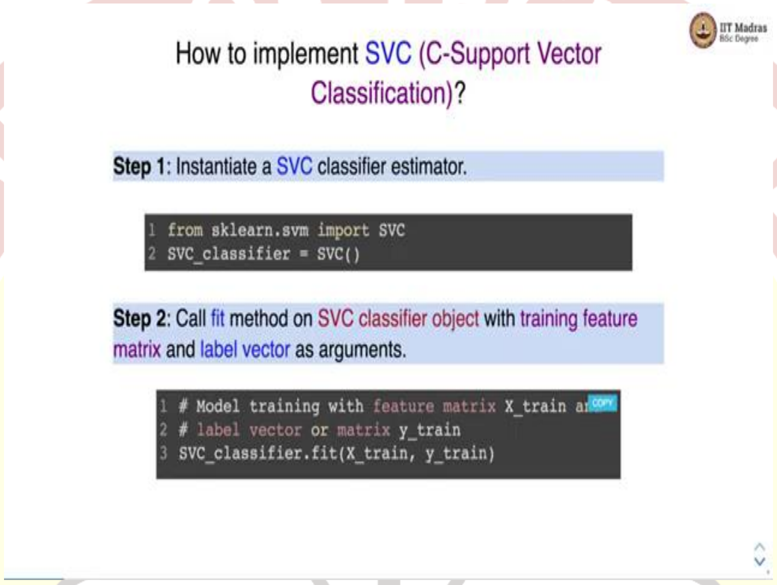
Array y : holding the class labels (strings or integers)

shape → (n_samples)

```
y = [0,1]
```

The training data has two components, the training samples or the feature matrix X which has got the shape of number of samples by number of features. So, this is an example of a feature matrix. Here the first sample has got two features which are set to 0, 0 and second one has got two features that are set to 1. We have y which holds the class labels and y has shape of number of samples it is a vector. So, y in this case is 0 and 1, the label for the first example is 0 and for the second example the label is 1.

(Refer Slide Time: 2:00)



How to implement SVC (C-Support Vector Classification)?

Step 1: Instantiate a SVC classifier estimator.

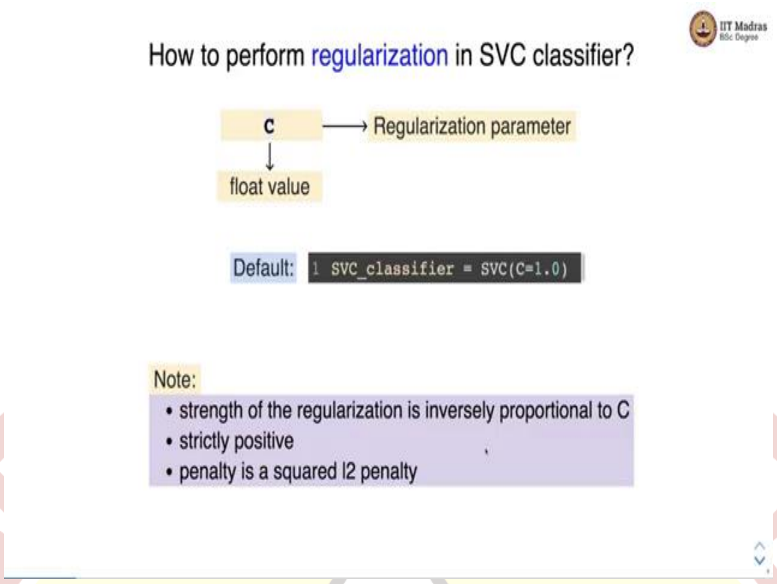
```python
from sklearn.svm import SVC
SVC_classifier = SVC()
```

Step 2: Call fit method on SVC classifier object with training feature matrix and label vector as arguments.

```python
# Model training with feature matrix X_train ar
# label vector or matrix y_train
SVC_classifier.fit(X_train, y_train)
```

Let us see how to implement SVC or C support vector classification. We first instantiate a SVC classifier estimator, we import SVC estimator from sklearn.svm module, we instantiate SVC object. In the second step we call fit method on SVC classifier object with training feature matrix and label vectors as argument. So, X_train is the feature matrix and Y_train is a label vector.

(Refer Slide Time: 2:36)



How to perform regularization in SVC classifier?

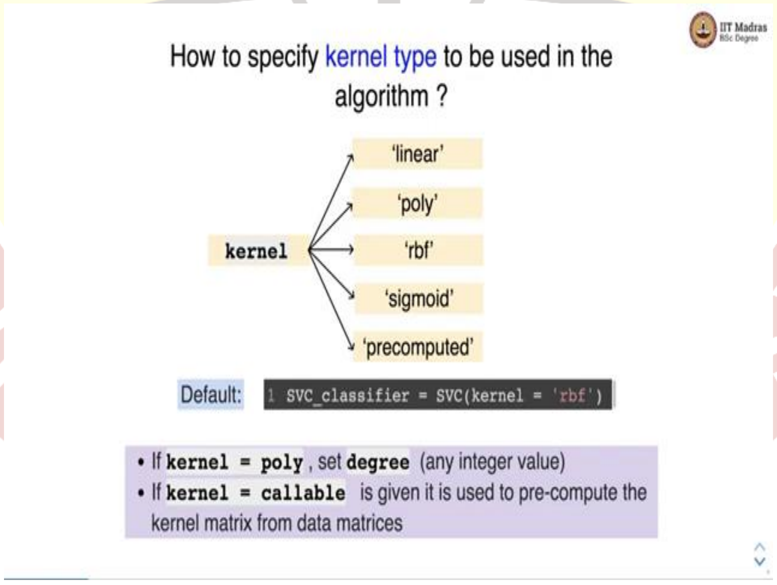
C
Regularization parameter
float value

Default:
```
SVC_classifier = SVC(C=1.0)
```

Note:
- strength of the regularization is inversely proportional to C
- strictly positive
- penalty is a squared l2 penalty

Let us see how to perform regularization in SVC classifier. So, there is a parameter c which is a regularization parameter which takes a float value. The default value of c is 1. The strength of regularization is inversely proportional to C, C is strictly positive and the penalty that is used is squared l2 penalty.

(Refer Slide Time: 3:02)



How to specify kernel type to be used in the algorithm ?

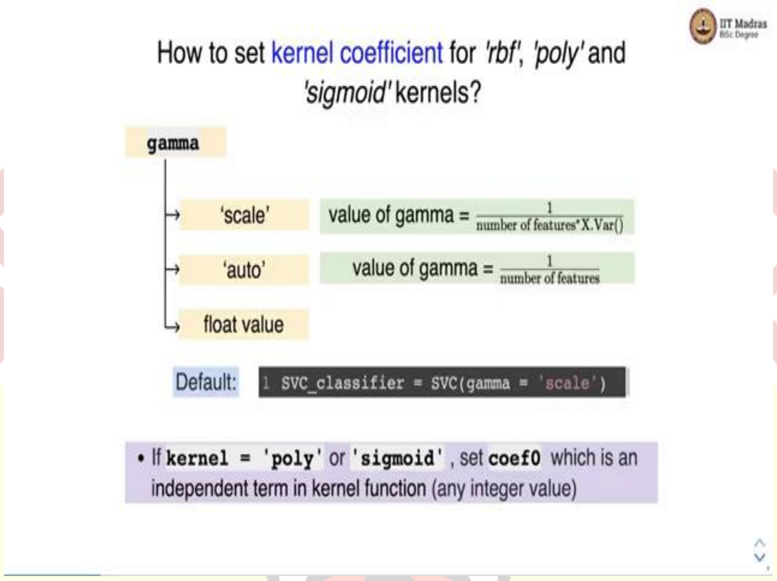
Default:
```
SVC_classifier = SVC(kernel = 'rbf')
```

- If `kernel = poly`, set `degree` (any integer value)
- If `kernel = callable` is given it is used to pre-compute the kernel matrix from data matrices

Let us see how to specify kernel type to be used in the algorithm. So, we can set the kernel to linear, polynomial, rbf which is radial basis function and sigmoid. And we can in addition to these we can also have precomputed kernels. We can set the kernel in the kernel argument of SVC object. Here we have set the kernel to rbf which is a default choice.

If kernel is = poly then we need to set an additional parameter which is degree of the polynomial and it takes any kind of any integer value. If kernel is = callable then it is used to precompute the kernel matrix from the data matrices. So, here with kernel = callable you can define your own kernel matrix based on the data that you have.

(Refer Slide Time: 4:00)



How to set kernel coefficient for *'rbf'*, *'poly'* and *'sigmoid'* kernels?

gamma

- 'scale' — value of gamma = $\frac{1}{\text{number of features} * X.Var()}$
- 'auto' — value of gamma = $\frac{1}{\text{number of features}}$
- float value

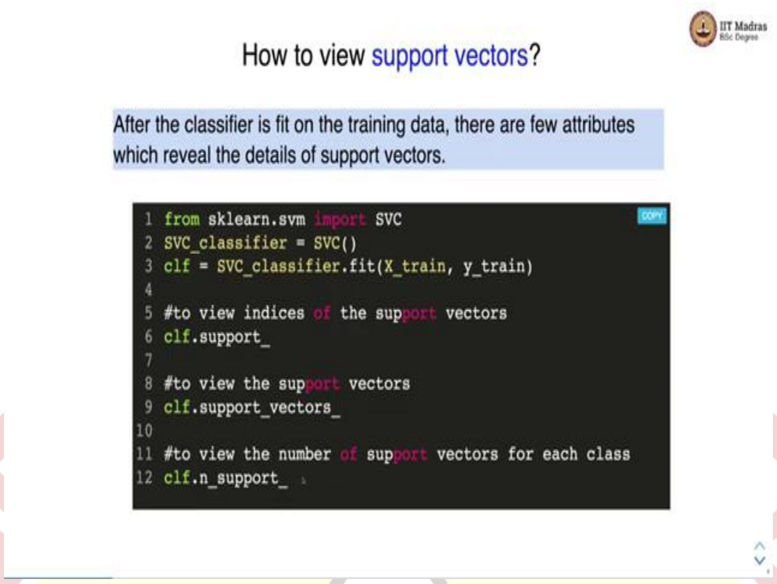
Default:

```
1 SVC_classifier = SVC(gamma = 'scale')
```

- If `kernel = 'poly'` or `'sigmoid'`, set `coef0` which is an independent term in kernel function (any integer value)

Let us see how to set the kernel coefficients for rbf, polynomial and sigmoid kernels. So, there is a parameter gamma that is used to set these coefficients. Gamma defines scale, auto value and it takes a float value the value of gamma in case of scale is 1 by number of features * variance of X which is the feature matrix. The default value of gamma is scale. If kernel is = poly or sigmoid then set coefficient 0 which is an independent term in the kernel function to any integer value.

(Refer Slide Time: 4:40)



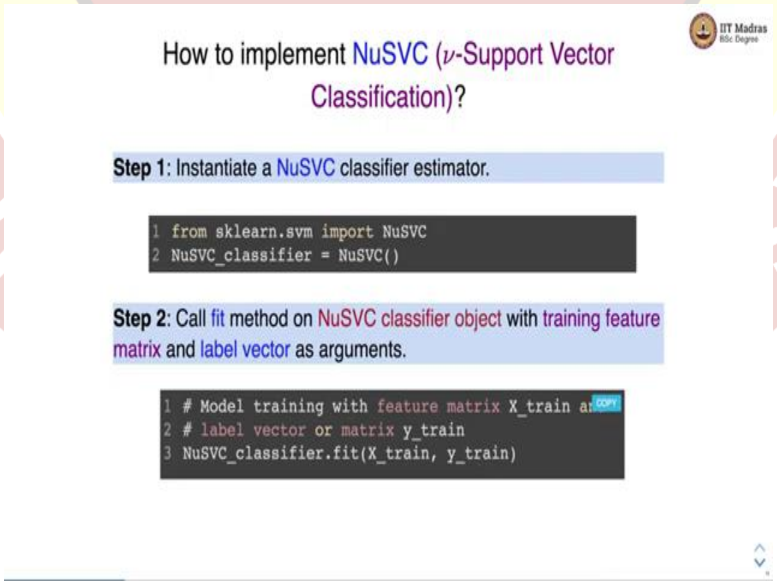
How to view support vectors?

After the classifier is fit on the training data, there are few attributes which reveal the details of support vectors.

```
from sklearn.svm import SVC
SVC_classifier = SVC()
clf = SVC_classifier.fit(X_train, y_train)

#to view indices of the support vectors
clf.support_

#to view the support vectors
clf.support_vectors_

#to view the number of support vectors for each class
clf.n_support_
```

Let us see how to view the support vectors. After the classifier is fit on the training data there are few attributes which reveal the details of the support vectors. So, we can take the support vectors using the object of SVC and we can access them using support_vectors_member variable. The indices of the support vectors can be accessed through support_member variable and to view the number of support vectors for each class we can access that with n_support_member variable.

(Refer Slide Time: 5:20)



How to implement NuSVC (ν-Support Vector Classification)?

Step 1: Instantiate a NuSVC classifier estimator.

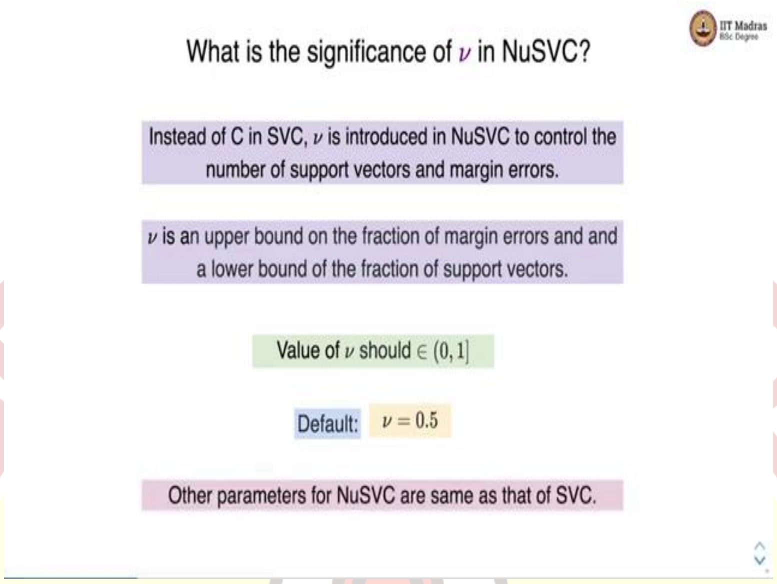
```
from sklearn.svm import NuSVC
NuSVC_classifier = NuSVC()
```

Step 2: Call fit method on NuSVC classifier object with training feature matrix and label vector as arguments.

```
# Model training with feature matrix X_train ar
# label vector or matrix y_train
NuSVC_classifier.fit(X_train, y_train)
```

Let us see how to implement NuSVC or v support vector classification. Now, to first instantiate a NuSVC classifier estimator. NuSVC is implemented again in sklearn.svm module. We instantiate NuSVC object and then call a fit method with the training feature matrix and label vectors as arguments.

(Refer Slide Time: 5:45)



What is the significance of ν in NuSVC?

Instead of C in SVC, ν is introduced in NuSVC to control the number of support vectors and margin errors.

ν is an upper bound on the fraction of margin errors and and a lower bound of the fraction of support vectors.

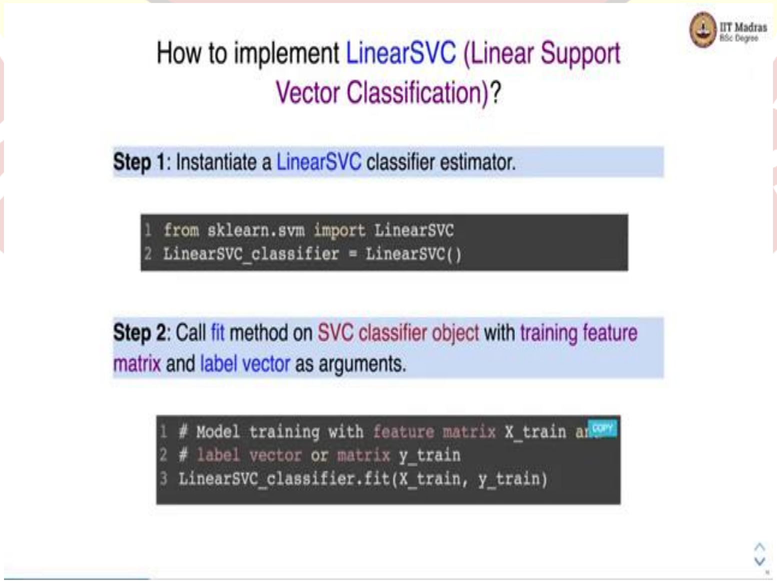
Value of ν should $\in (0,1]$

Default: $\nu = 0.5$

Other parameters for NuSVC are same as that of SVC.

Let us see the significance of v in NuSVC instead of C in SVC v is introduced in NuSVC to control the number of support vectors and margin errors, v is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors. The value of v should be between 0 and 1. The default value of v is 0.5, other parameters for NuSVC are same as that of SVC.

(Refer Slide Time: 6:21)



How to implement LinearSVC (Linear Support Vector Classification)?

Step 1: Instantiate a LinearSVC classifier estimator.

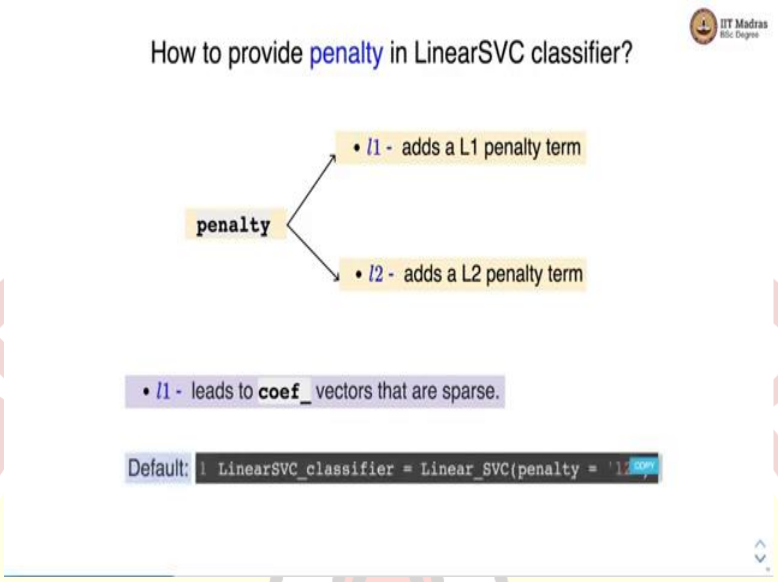
```
from sklearn.svm import LinearSVC
LinearSVC_classifier = LinearSVC()
```

Step 2: Call fit method on SVC classifier object with training feature matrix and label vector as arguments.

```
# Model training with feature matrix X_train an
# label vector or matrix y_train
LinearSVC_classifier.fit(X_train, y_train)
```

Let us see how to implement linear SVC which is linear support vector classification. So, we instantiate linear SVC classifier estimator which is imported from sklearn.SVM module, we instantiate the object of linear SVC and then call the fit method with training feature vectors and label vector as argument just like other SVM classifiers.

(Refer Slide Time: 6:47)



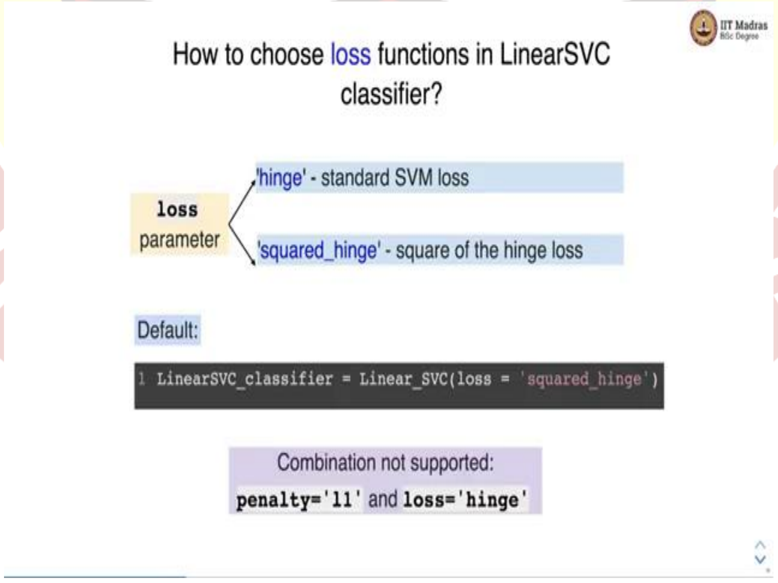
How to provide penalty in LinearSVC classifier?

• l1 - leads to **coef_** vectors that are sparse.

Default:
```
1 LinearSVC_classifier = Linear_SVC(penalty = 'l2')
```

Let us see how to provide penalty in linear SVC classifier. So, we can add l1 penalty which is l1 norm or l2 penalty and l1 leads to coefficient vectors being sparse and by default linear SVC uses l2 penalty.

(Refer Slide Time: 7:05)



How to choose loss functions in LinearSVC classifier?

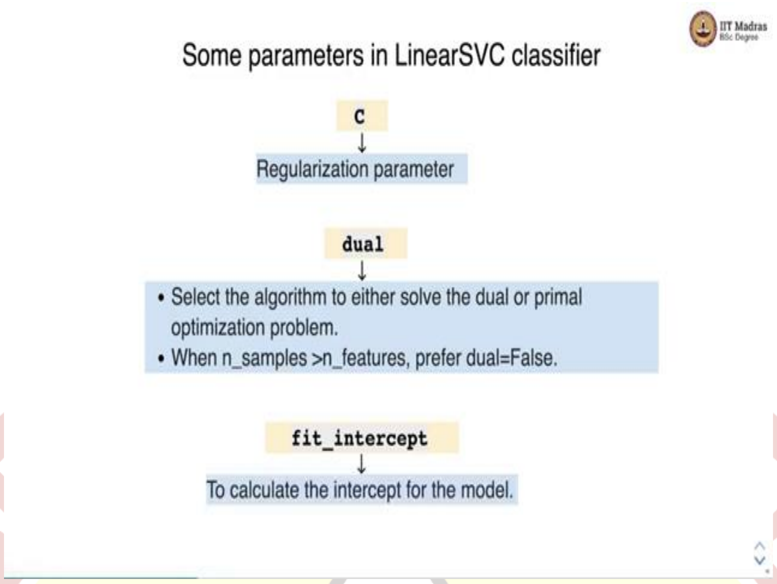
loss parameter
'hinge' - standard SVM loss
'squared_hinge' - square of the hinge loss

Default:
```
1 LinearSVC_classifier = Linear_SVC(loss = 'squared_hinge')
```

Combination not supported:
penalty='l1' and **loss='hinge'**

Let us see how to choose loss function in linear SVC classifier. So, we have a loss parameter that can be set to hinge loss which results into standard SVM. Then we can also use squared_hinge loss which is square of the hinge loss by default linear SVC uses squared hinge loss, cannot use penalty l1 and hinge loss that is not supported in linear SVC.

(Refer Slide Time: 7:33)



Some parameters in LinearSVC classifier

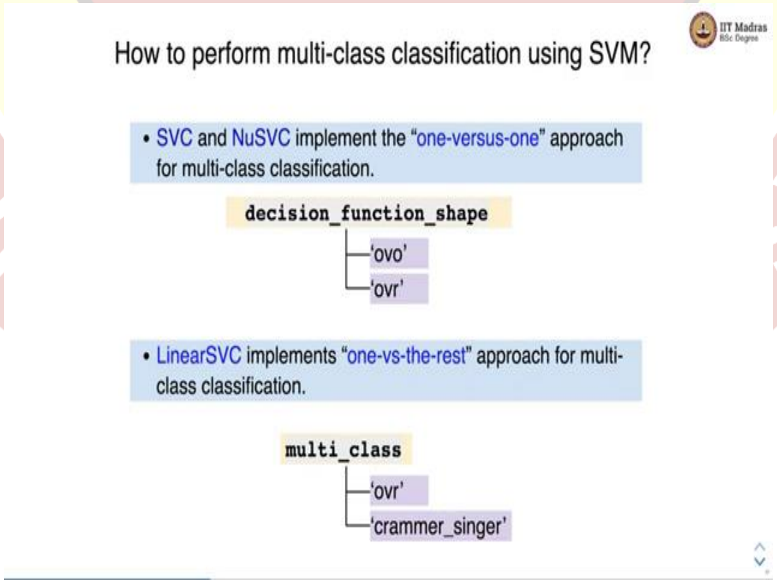
C → Regularization parameter

dual →
- Select the algorithm to either solve the dual or primal optimization problem.
- When n_samples >n_features, prefer dual=False.

fit_intercept → To calculate the intercept for the model.

Let us look at some of the parameters in linear SVC classifier. There is a parameter C which is a regularization parameter. Then there is a parameter dual which selects the algorithm to solve, it could be either a dual or primal optimization problem. When number of samples are greater than number of features prefer to use the primal problem by setting dual to false. We also have another parameter which is fit_intersect parameter that instructs model to calculate the intercept.

(Refer Slide Time: 8:08)



How to perform multi-class classification using SVM?

- SVC and NuSVC implement the "one-versus-one" approach for multi-class classification.

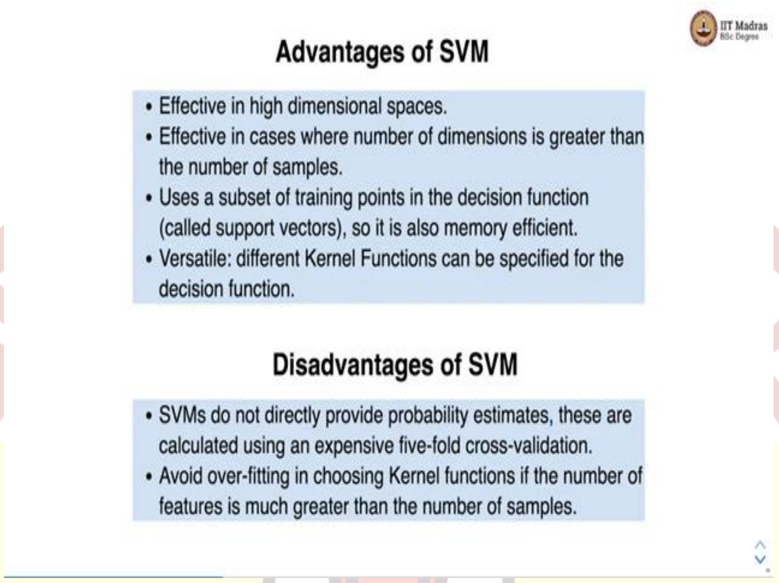
decision_function_shape
- 'ovo'
- 'ovr'

- LinearSVC implements "one-vs-the-rest" approach for multi-class classification.

multi_class
- 'ovr'
- 'crammer_singer'

Let us see how to perform multi-class classification using SVM. SVC and NuSVC implement "one versus one" approach for multi-class classification. You have to specify the decision function shape which can either be one versus one or one versus rest. Linear SVC implements "one versus the rest" approach for multi-class classification. So, multi-class is either ovr or grammar singer which is more optimized way for doing multi-class classification, but crammer singer is rarely used in practice because it is slower.

(Refer Slide Time: 8:51)



Advantages of SVM

- Effective in high dimensional spaces.
- Effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel Functions can be specified for the decision function.

Disadvantages of SVM

- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.
- Avoid over-fitting in choosing Kernel functions if the number of features is much greater than the number of samples.

Let us look at some of the advantages and disadvantages of SVM. So, SVM is effective in high dimensional spaces. It is effective in cases where the number of dimensions is greater than the number of samples. It uses a subset of turning points in the decision function and this subset of training points is called support vectors, so it is very memory efficient.

It is versatile, different kernel functions can be specified for the decision function. So, the disadvantages of SVM includes SVM do not directly provide probability estimates, these are calculated using expensive five-fold cross-validation. It avoids overfitting in choosing kernel functions if the number of features is much greater than the number of samples. So, these are some advantages and disadvantages of SVM. So, in this video we studied how to implement SVM with sklearn library.