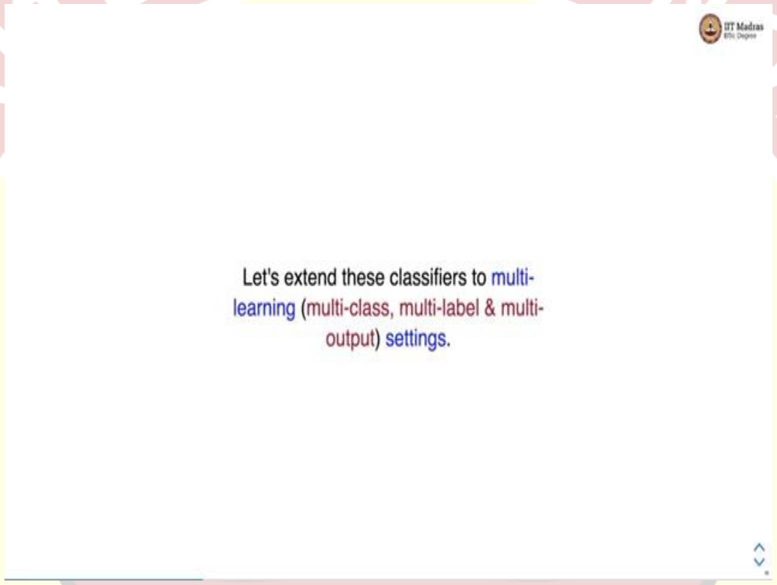
INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

IIT Madras
ONLINE DEGREE

Machine Learning Practice
B. Sc in Programming and Data Science
Diploma Level
Indian Institute of Technology, Madras
Multi-learning classification set up

Namaste welcome to the next video of machine learning practice course. In this video we will study how sklearn supports multi-learning classification set up.

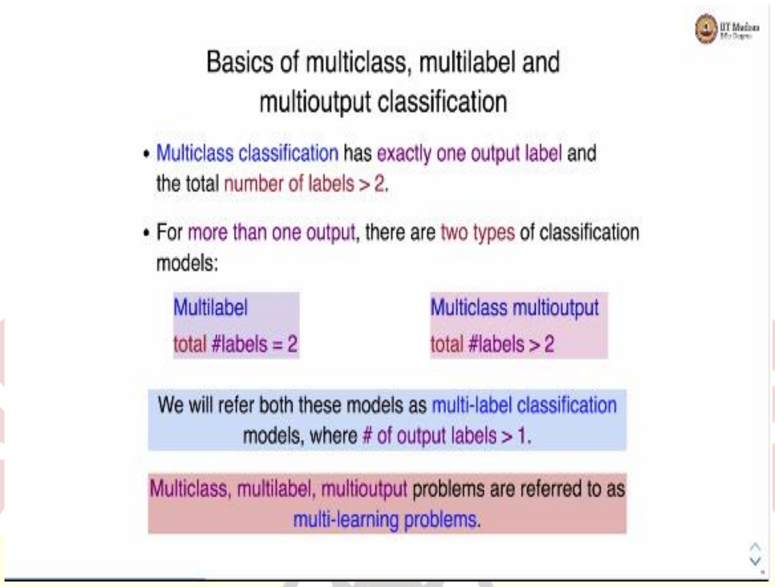
(Refer Slide Time: 0:22)



Let's extend these classifiers to multi-learning (multi-class, multi-label & multi-output) settings.

In the last video, we studied different classifiers that are supported by sklearn. In this video we will extend these classifiers to multi-learning settings.

(Refer Slide Time: 0:33)



Basics of multiclass, multilabel and multioutput classification

- Multiclass classification has exactly one output label and the total number of labels > 2.
- For more than one output, there are two types of classification models:

Multilabel	Multiclass multioutput
total #labels = 2	total #labels > 2

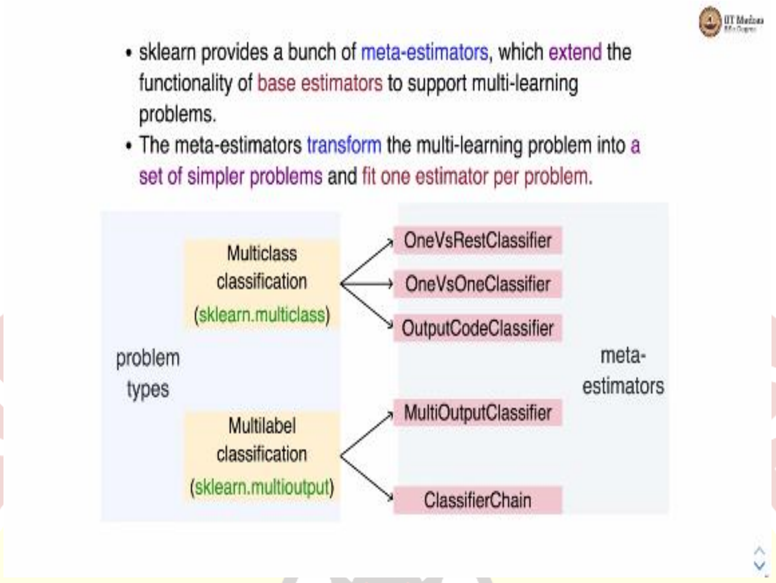
We will refer both these models as multi-label classification models, where # of output labels > 1.

Multiclass, multilabel, multioutput problems are referred to as multi-learning problems.

There are three different type of setups in multi-learning settings. The first one is multiclass classification, it has exactly one output label and the total number of labels > 2. For more than one output, there are two types of classification model, multilabel and multioutput. In multilabel the total number of labels = 2. And there are more than one label for example.

In multiclass multi output, we have more than one output label for example and the total number of labels > 2. So, only difference here is the number of labels, here it is 2, here it > 2. So, we refer to both these models as multilabel classification. Here remember that number of output labels per samples > 1. So, multiclass, multilabel and multioutput problems are referred to as multi-learning problems.

(Refer Slide Time: 1:44)

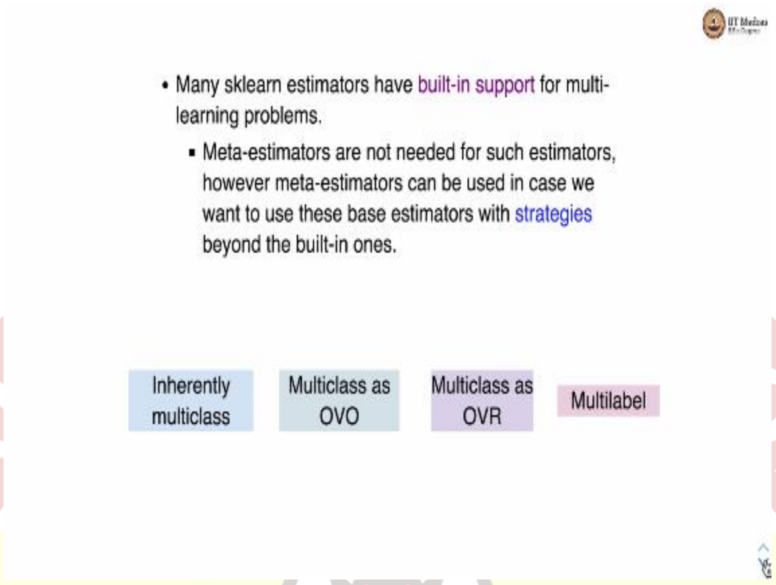


- sklearn provides a bunch of meta-estimators, which extend the functionality of base estimators to support multi-learning problems.
- The meta-estimators transform the multi-learning problem into a set of simpler problems and fit one estimator per problem.

Sklearn provides a bunch of meta-estimators, which extend the functionalities of base estimator to support multi-learning problems. The meta-estimators transform the multi-learning problem into a set of simple problems and fit one estimator per problem. So, we have primarily two type of problems, multiclass classification and multilabel classification. And there are meta-estimators for each of these problem types.

So, in case of multiclass classification we have three meta-estimator, OneVsRestClassifier, OneVsOneClassifier and OutputCodeClassifier. We will focus on first two meta-estimators in this, in this video, which is OneVsRestClassifier and OneVsOneClassifier. In multilabel classification we have two meta-estimator, MultiOutputClassifier and classification, or and classifier chain.

(Refer Slide Time: 2:53)

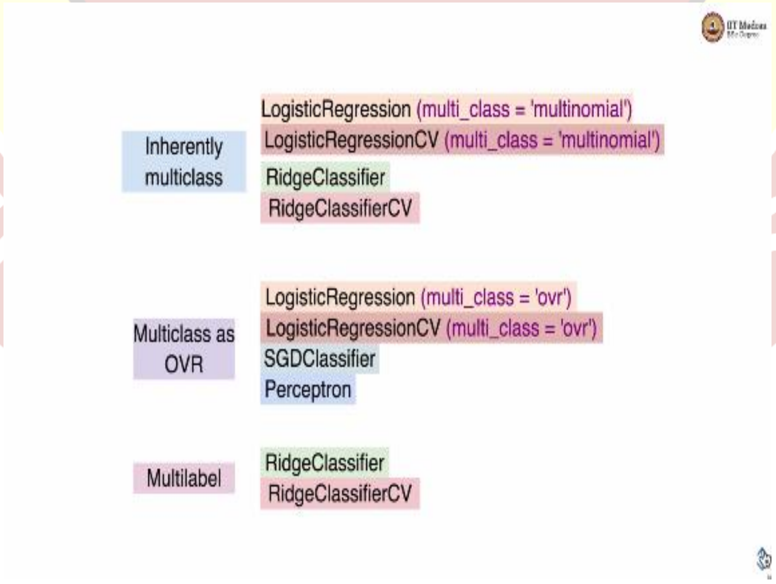


- Many sklearn estimators have built-in support for multi-learning problems.
  - Meta-estimators are not needed for such estimators, however meta-estimators can be used in case we want to use these base estimators with strategies beyond the built-in ones.

Inherently multiclass
Multiclass as OVO
Multiclass as OVR
Multilabel

So, many sklearn estimators have built-in support for multi-learning problems. Meta-estimators are not needed for such estimators, however meta-estimators still can be used in case we want to use this base estimator with strategies beyond the ones that are already built in. Some sklearn estimators are inherently multiclass, the other ones implement one Vs one, or one Vs straight strategies. And there are also multilabel sklearn estimators.

(Refer Slide Time: 3:32)

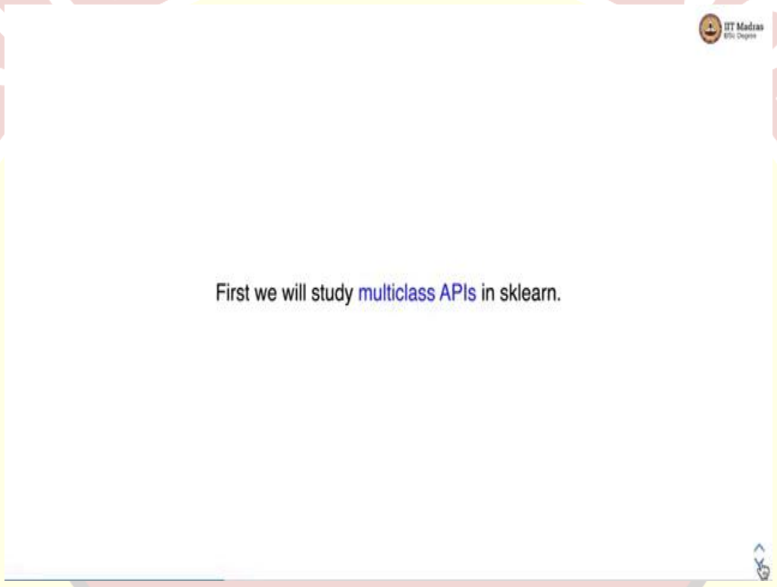


Inherently multiclass
LogisticRegression (multi_class = 'multinomial')
LogisticRegressionCV (multi_class = 'multinomial')
RidgeClassifier
RidgeClassifierCV
Multiclass as OVR
LogisticRegression (multi_class = 'ovr')
LogisticRegressionCV (multi_class = 'ovr')
SGDClassifier
Perceptron
Multilabel
RidgeClassifier
RidgeClassifierCV

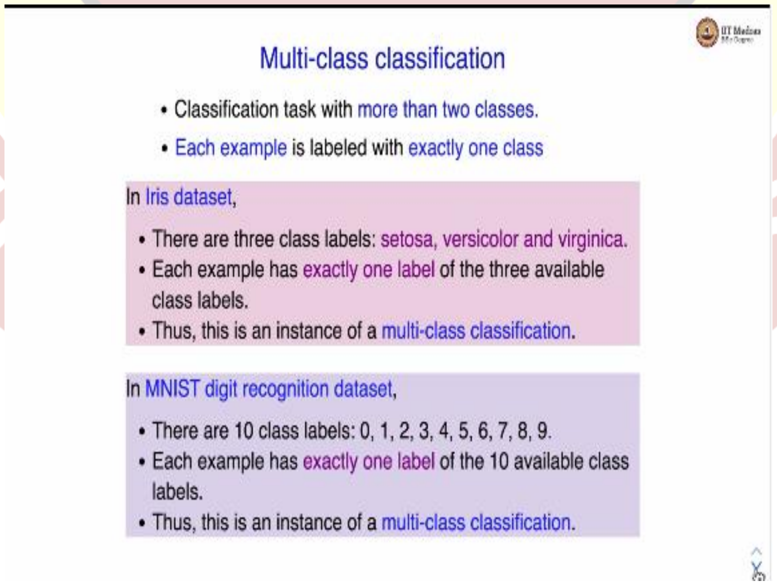
So, logistic regression with multi_class =multinomial is an inherently multiclass estimator. In the same manner logistic regression CV also supports multiclass by setting the multiclass parameter to multi multinomial. Ridge classifier and its counterpart ridge classifier CV are also inherently multiclass.

So, logistic regression can also implement multiclass using one Vs the rest strategy. So, for that we set multi-underscore class argument to OVR. SGD classifier and perceptron also uses one Vs rest strategy to implement multiclass classification. Ridge classifier on the other hand is inherently multilabel estimator.

(Refer Slide Time: 4:42)



First we will study multiclass APIs in sklearn.



Multi-class classification

- Classification task with more than two classes.
- Each example is labeled with exactly one class

In Iris dataset,

- There are three class labels: setosa, versicolor and virginica.
- Each example has exactly one label of the three available class labels.
- Thus, this is an instance of a multi-class classification.

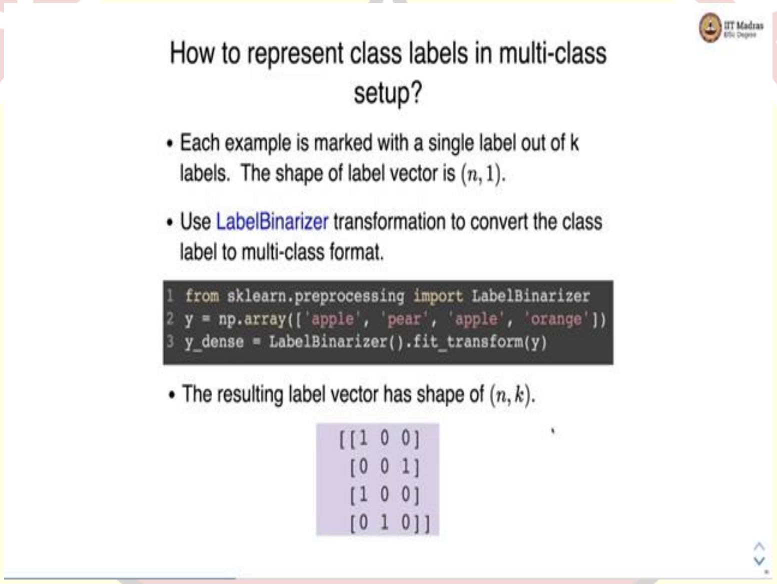
In MNIST digit recognition dataset,

- There are 10 class labels: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Each example has exactly one label of the 10 available class labels.
- Thus, this is an instance of a multi-class classification.

First we will study multiclass APIs in sklearn. So, as you know multiclass classification is the classification task with more than two classes, each example is labeled with exactly one class. For example in Iris data set, there are three labels, setosa, versicolor and virginica, each example has exactly one label out of these three available class label. Hence, Iris data set is an instance of a multiclass classification problem.

In the same manner in MNIST digit recognition data set, there are 10 class labels, 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. So, here the idea is to look at the image of a digit and recognize what is the digit present in the image. So, each example has exactly one label out of 10 available class labels. Hence, MNIST digit recognition is also an instance of a multiclass classification problem.

(Refer Slide Time: 5:52)



How to represent class labels in multi-class setup?

- Each example is marked with a single label out of k labels. The shape of label vector is $(n, 1)$.
- Use LabelBinarizer transformation to convert the class label to multi-class format.

```
from sklearn.preprocessing import LabelBinarizer
y = np.array(['apple', 'pear', 'apple', 'orange'])
y_dense = LabelBinarizer().fit_transform(y)
```

- The resulting label vector has shape of (n, k).

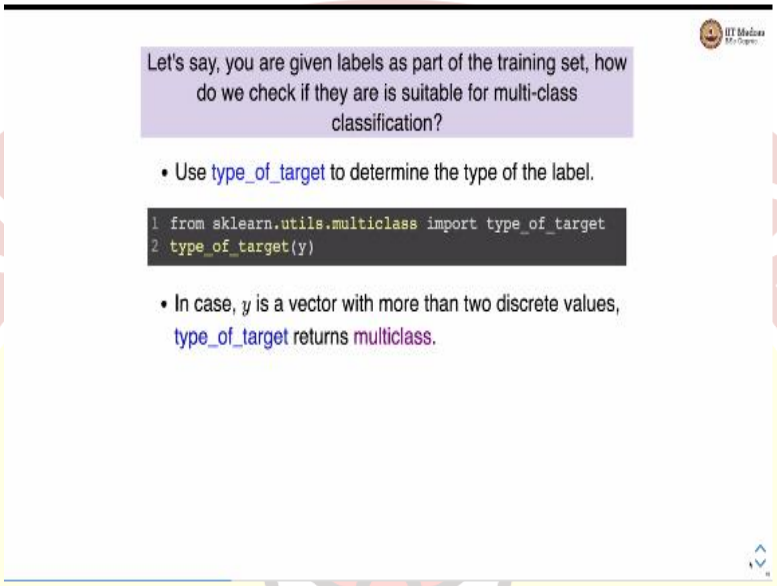
```
[[1 0 0]
 [0 0 1]
 [1 0 0]
 [0 1 0]]
```

Let us look at how to represent class labels in multiclass setup, each example is marked with a single label out of k labels. The shape of the label vector is n. So, what we do is we use label binarizer transformation to convert the class label into the multiclass format. So, we simply use a label binarizer transformation from sklearn. preprocessing and we call fit_transform with by supplying the available label vector.

And labele binarizer returns a label vector with shape n, k, where n is the number of examples and k is the number of classes. So, in this case since there are three, three distinct labels, apple, pear, and orange. So, you can see that there are three different labels that are created. So, the first label corresponds to apple, second one corresponds to orange, and the third one corresponds to pear.

So, you can see that apple is present in the first and third example, that is why the encoding here is 1, 0, 0, and 1, 0, 0. Orange is present in the last example, so that is why 0, 1, 0, and pear is present in the, in the second example hence it is 0, 0, 1.

(Refer Slide Time: 7:38)



Let's say, you are given labels as part of the training set, how do we check if they are is suitable for multi-class classification?

- Use type_of_target to determine the type of the label.

```
from sklearn.utils.multiclass import type_of_target
type_of_target(y)
```

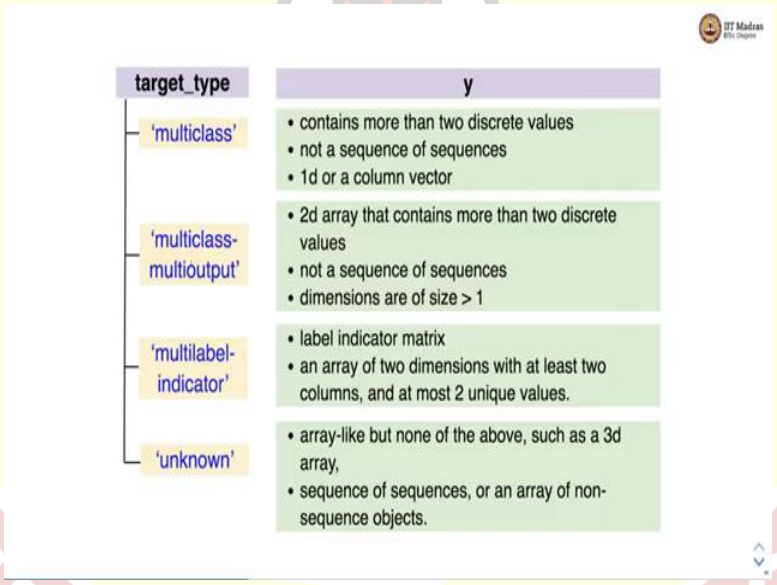
- In case, y is a vector with more than two discrete values, type_of_target returns multiclass.

So, let us say you are given labels as part of the training set. How do we check if they are suitable for multiclass classification? So, for that, we use type_of_target to determine the type of the label. So, type of target is present in sklearn. utils. multiclass library.

So, we simply supply the label matrix, or the label input to type of target. And it tells us what exactly is the type of target. So, in this case, where the vector has more than two discrete values, type of target returns multiclass.

(Refer Slide Time: 8:28)

type_of_target can determine different types of multi-learning targets.

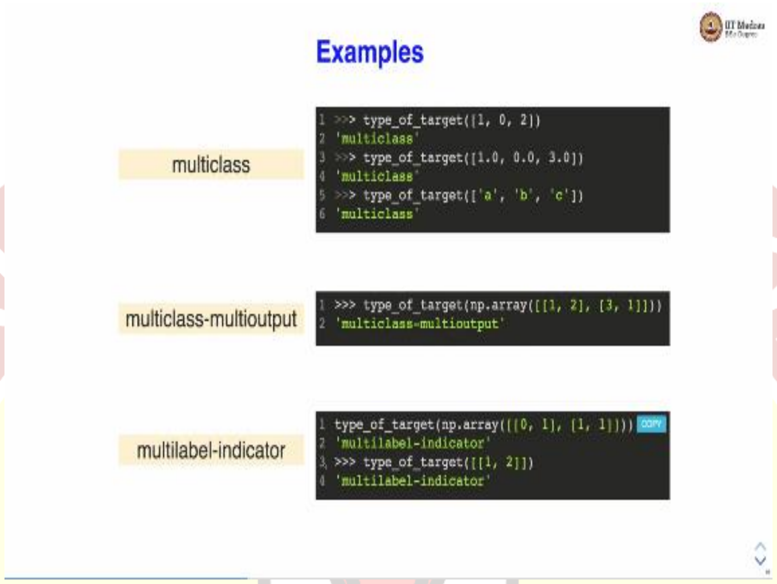


target_type	y
'multiclass'	• contains more than two discrete values • not a sequence of sequences • 1d or a column vector
'multiclass-multioutput'	• 2d array that contains more than two discrete values • not a sequence of sequences • dimensions are of size > 1
'multilabel-indicator'	• label indicator matrix • an array of two dimensions with at least two columns, and at most 2 unique values.
'unknown'	• array-like but none of the above, such as a 3d array, • sequence of sequences, or an array of non-sequence objects.

So, type of target can determine different types of multi-learning targets. Let us look at different type of targets that are recognized by type of target. So, multiclass, which has got more than two discrete values, then there is multiclass, multi output, where we have more than two discrete values. And we also have more than one label for example.

So, that becomes multiclass multioutput, then we have multilabel indicator, where we have exactly two unique values in the label column and each example gets more than one label. So, we have label indicator matrix as the input that is multilabel indicator. And finally there is unknown, where it is array-like but none of the above cases hold, it could also be sequence of sequences, or an array of non-sequence objects. So, type of targets there are multiclass, multiclass multioutput, and multilabel indicator.

(Refer Slide Time: 9:49)



Examples

multiclass

```
>>> type_of_target([1, 0, 2])
'multiclass'
>>> type_of_target([1.0, 0.0, 3.0])
'multiclass'
>>> type_of_target(['a', 'b', 'c'])
'multiclass'
```

multiclass-multioutput

```
>>> type_of_target(np.array([[1, 2], [3, 1]]))
'multiclass-multioutput'
```

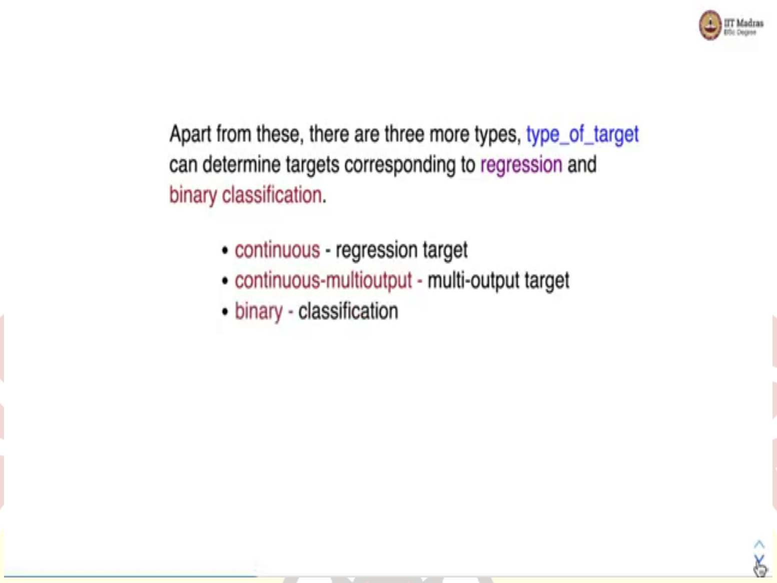
multilabel-indicator

```
type_of_target(np.array([[0, 1], [1, 1]]))
'multilabel-indicator'
>>> type_of_target([[1, 2]])
'multilabel-indicator'
```

So, let us look at the multiclass targets. So, you can see that here there are more than two labels, that is 0, 1, and 2, hence this is multiclass and each example is labeled exactly with one of the labels, this is also multiclass, there are three distinct values. So, these are three examples of multiclass targets.

Then we have multiclass multioutput, you can see that each example has got more than one label, so that is why this is multioutput and there are more than two distinct labels, so that is why this is multiclass multioutput. Then here we have again more than one, one label for sample and there are exactly two labels here, that is 0, and 1, and this is multilabel indicator.

(Refer Slide Time: 10:56)

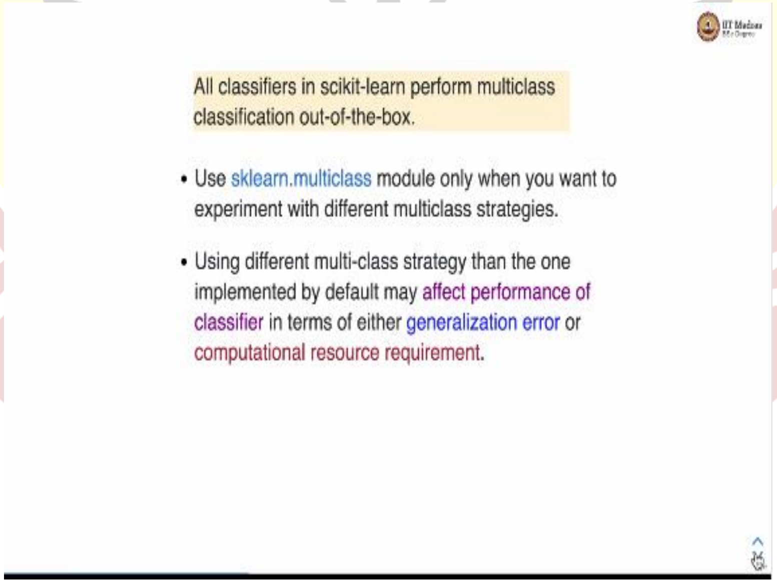


Apart from these, there are three more types, type_of_target can determine targets corresponding to regression and binary classification.

- continuous - regression target
- continuous-multioutput - multi-output target
- binary - classification

So, apart from this there are three more types of target that can determine whether the target corresponds to regression, or binary classification. Continuous, which denotes the regression target, continuous multi output, which is multi output regression. And then there is binary, which is for binary classification.

(Refer Slide Time: 11:24)



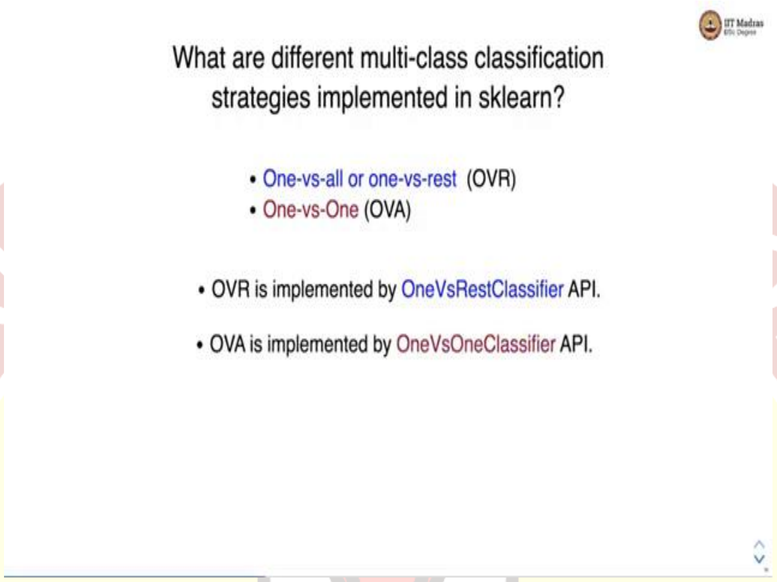
All classifiers in scikit-learn perform multiclass classification out-of-the-box.

- Use sklearn.multiclass module only when you want to experiment with different multiclass strategies.
- Using different multi-class strategy than the one implemented by default may affect performance of classifier in terms of either generalization error or computational resource requirement.

So, all classifiers in scikit-learn perform multiclass classification out of the box. So, we use sklearn. multiclass module only when we want to experiment with different multiclass strategy than the one that is implemented in the in the state of the in the, in the state of the art, sklearn estimators.

So, using different multiclass strategy, then the one implemented may affect the performance of the classifier in terms of either the generalization error, or computational resource requirement.

(Refer Slide Time: 11:59)



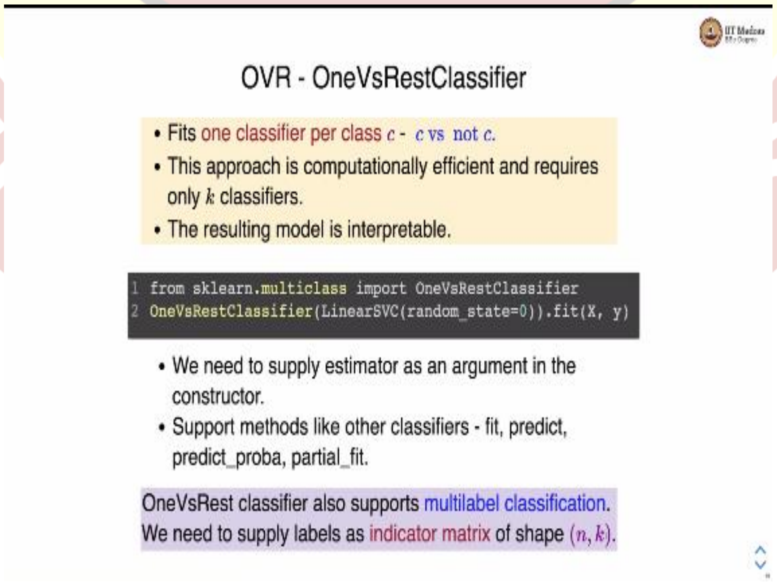
What are different multi-class classification strategies implemented in sklearn?

- One-vs-all or one-vs-rest (OVR)
- One-vs-One (OVA)

- OVR is implemented by OneVsRestClassifier API.
- OVA is implemented by OneVsOneClassifier API.

So, what are different multiclass classification strategies implemented in sklearn? That is One-vs-all, or One-vs-rest abbreviated as OVR, or and One-vs-One which is abbreviated as OVA. OVR is implemented by OneVsRestClassifier API, whereas OVA is implemented by OneVsOneClassifier API.

(Refer Slide Time: 12:30)



OVR - OneVsRestClassifier

- Fits one classifier per class c - c vs not c.
- This approach is computationally efficient and requires only k classifiers.
- The resulting model is interpretable.

```
from sklearn.multiclass import OneVsRestClassifier
OneVsRestClassifier(LinearSVC(random_state=0)).fit(X, y)
```

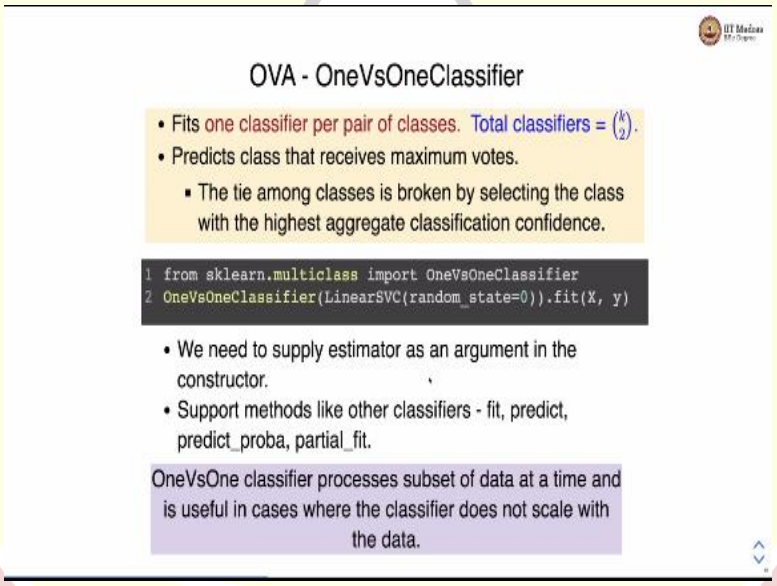
- We need to supply estimator as an argument in the constructor.
- Support methods like other classifiers - fit, predict, predict_proba, partial_fit.

OneVsRest classifier also supports multilabel classification. We need to supply labels as indicator matrix of shape (n, k).

So, OVR, fits one classifier per class. So, there is a class c vs not c, everything which is not part of class c gets the negative label and that is how the classification problem is formed. So, this approach is computationally efficient and requires only k classifier. So, there is one classifier required per label and resulting model is also interpretable.

We first import OneVsRestClassifierfrom sklearn. multiclass library. In OneVsRestClassifierwe supply estimator as an argument. So, here we are using linear SVC as an estimator for one Vs rest classifier. The OneVsRestClassifier supports methods like fit, predict, predict probability and partial fit just like any other classifier. So, OneVsRestClassifier also supports multilabel classification. We need to supply labels as indicator matrix of shape n, k.

(Refer Slide Time: 13:48)



OVA - OneVsOneClassifier

- Fits one classifier per pair of classes. Total classifiers = $\binom{k}{2}$.
- Predicts class that receives maximum votes.
  - The tie among classes is broken by selecting the class with the highest aggregate classification confidence.

```
from sklearn.multiclass import OneVsOneClassifier
OneVsOneClassifier(LinearSVC(random_state=0)).fit(X, y)
```

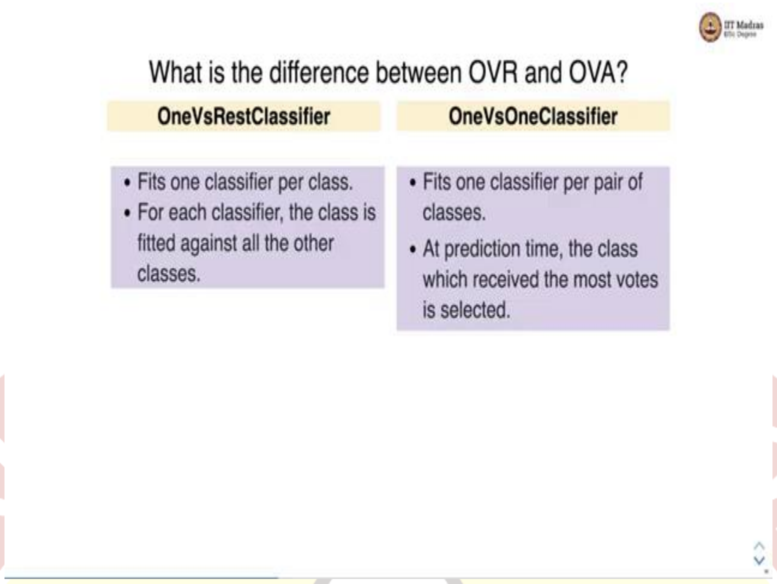
- We need to supply estimator as an argument in the constructor.
- Support methods like other classifiers - fit, predict, predict_proba, partial_fit.

OneVsOne classifier processes subset of data at a time and is useful in cases where the classifier does not scale with the data.

So, OneVsOneClassifier is another strategy, that is used in order to fit the multiclass classification problems, it fits one classifier per pair of classes. So, total number of classifiers $=C_2^k$, it predicts class that receives maximum volts, the tie among classes is broken by selecting the class with the highest aggregate classification confidence.

We import OneVsOneClassifier from sklearn. multiclass library. We supply the estimator as an argument in the constructor of OneVsOneClassifier, it supports methods like fit, predict, predict probability, and partial fit just like any other classifier. OneVsOneClassifier processes subset of data at a time and hence it is useful in cases, where the classifier does not scale with the data.

(Refer Slide Time: 14:53)

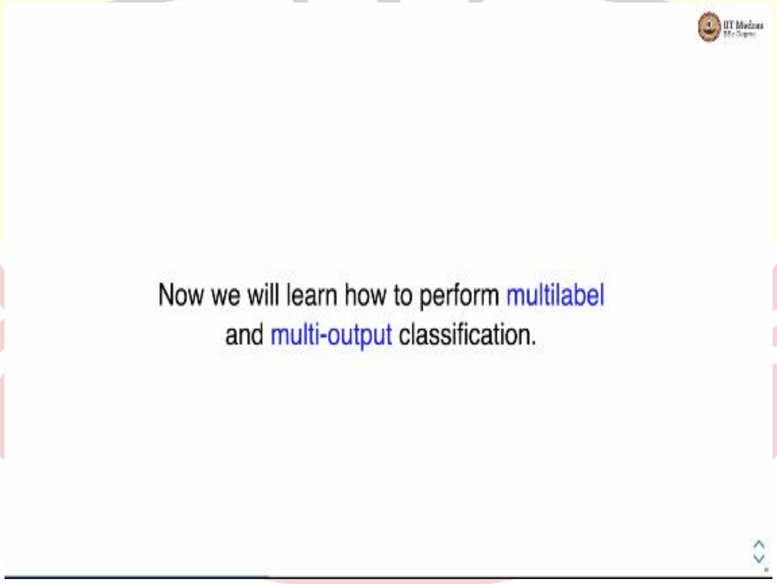


What is the difference between OVR and OVA?

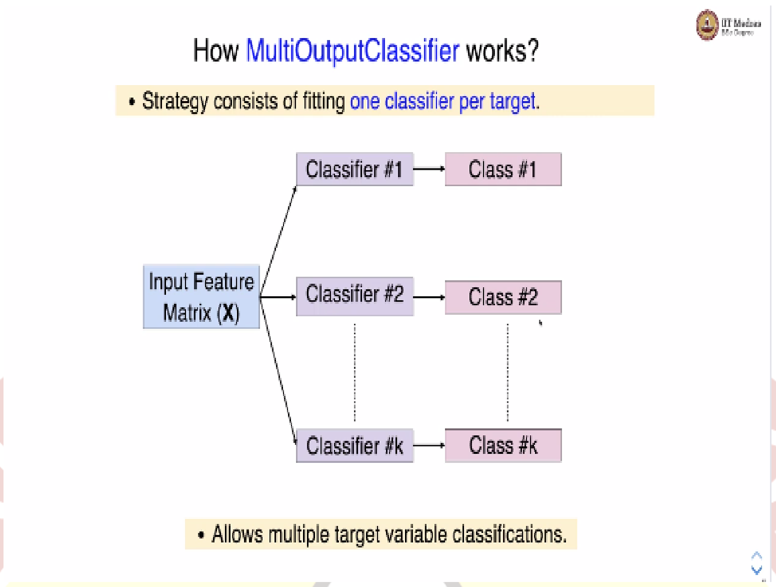
OneVsRestClassifier	OneVsOneClassifier
• Fits one classifier per class. • For each classifier, the class is fitted against all the other classes.	• Fits one classifier per pair of classes. • At prediction time, the class which received the most votes is selected.

So, what is the difference between OVR and OVA strategies? OVR, or OneVsRestClassifier, fits one classifier per class, for each classifier the class is fitted against all other, all other classes. OneVsOneClassifier, on the other hand fits one classifier per pair of classes, at prediction time the class, which receives the most votes is selected.

(Refer Slide Time: 15:22)



Now we will learn how to perform multilabel and multi-output classification.



How MultiOutputClassifier works?

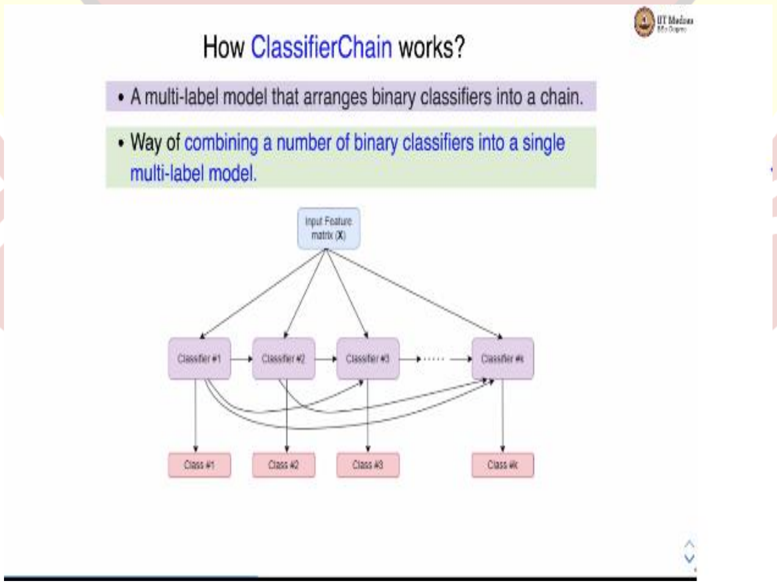
- Strategy consists of fitting one classifier per target.

- Allows multiple target variable classifications.

Now, we will learn how to perform multilabel and multioutput classification. So, let us look at how multioutput classifiers work? So, strategy here is to fit one classifier per target. So, here X is the input feature matrix, and we fit one classifier per target and that classifier is responsible for predicting that particular target.

So, if there are k different, if there are k different classes, then we will be effectively training a different classifiers.

(Refer Slide Time: 16:06)



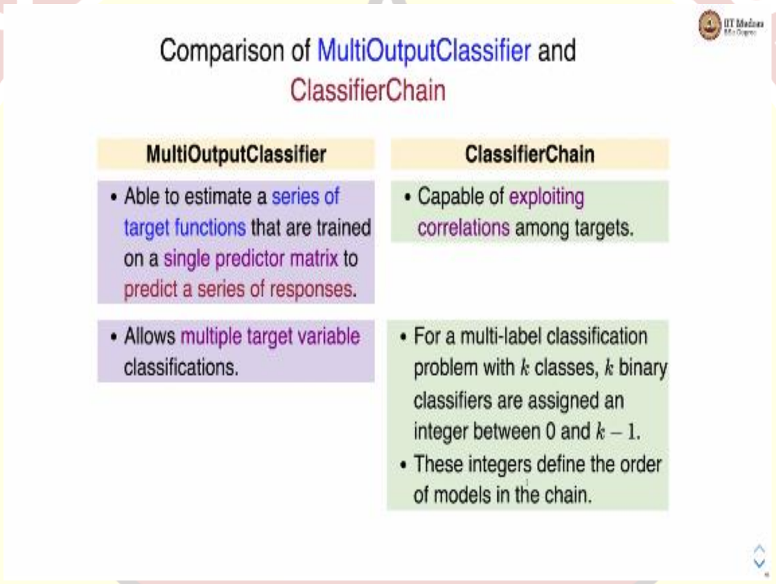
How ClassifierChain works?

- A multi-label model that arranges binary classifiers into a chain.
- Way of combining a number of binary classifiers into a single multi-label model.

The second strategy is to use classifier chain. So, multilabel model arranges binary classifiers into a chain. It is a way of combining a number of binary classifiers into a single multilabel model. So, the input feature matrix X is first sent to classifier 1, and we get the output of this particular classifier and dense output is also fed into the subsequent classifiers.

Then this for the second classifier, we get the input feature matrix as well as the output of the previous classifier. And this classifier basically learns to predict the class label based on the input feature matrix and input from the previous class. So, this is how the classification, or the classifier chain is set up. So, there is some configuration that is required in order to decide what is the optimal sequence of class labels.

(Refer Slide Time: 17:22)

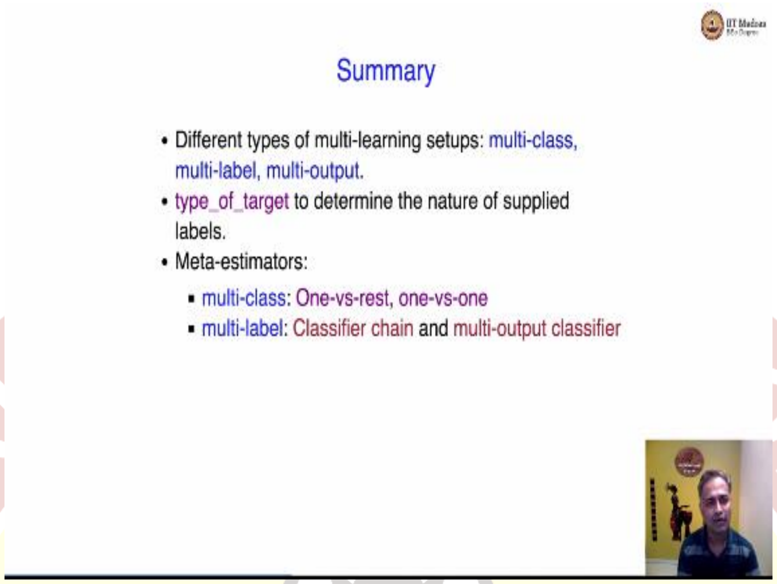


Comparison of MultiOutputClassifier and ClassifierChain

MultiOutputClassifier	ClassifierChain
• Able to estimate a series of target functions that are trained on a single predictor matrix to predict a series of responses.	• Capable of exploiting correlations among targets.
• Allows multiple target variable classifications.	• For a multi-label classification problem with k classes, k binary classifiers are assigned an integer between 0 and $k-1$. • These integers define the order of models in the chain.

So, in multioutput classifier, we are able to estimate a series of target functions, that are trained on a single predictor matrix to predict a series of responses. Whereas, classifier chain is capable of exploiting correlation among targets. Multioutput classifier allows multiple target variable classification. For multilabel classification problem with key classes k binary classifiers are assigned an integer value between 0 and k - 1. These integers define the order of models in the chain.

(Refer Slide Time: 18:00)



Summary

- Different types of multi-learning setups: multi-class, multi-label, multi-output.
- type_of_target to determine the nature of supplied labels.
- Meta-estimators:
  - multi-class: One-vs-rest, one-vs-one
  - multi-label: Classifier chain and multi-output classifier

So, in this video, we studied different types of multi-learning setups, multi-class, multi-label and multi-output. We studied type of target, that determines the nature of the supplied label and whether that label is adequate for the given multi-learning setup.

We studied meta estimator for multi-class and multi-label classification problem. In multi-class classification problem, we used One-vs-rest and one-vs-one meta-estimators. Whereas, in multi-label, we use classifier chain and multi-output meta-estimators.