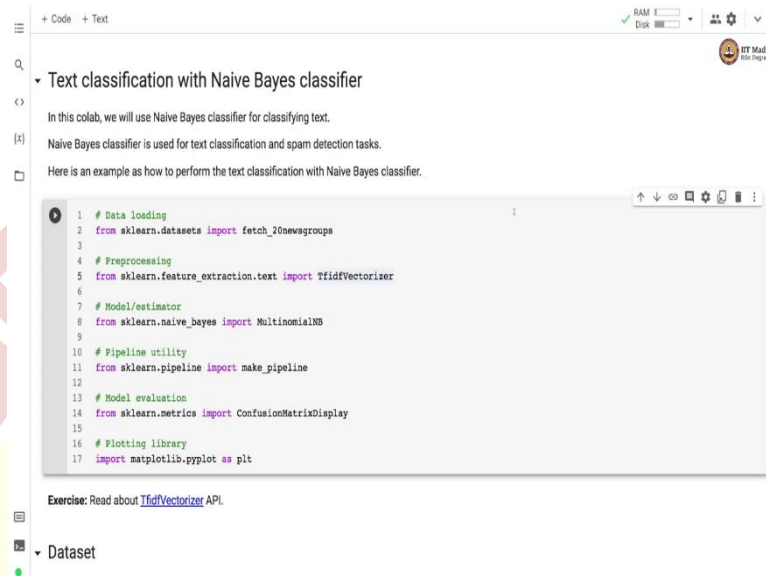


IIT Madras

ONLINE DEGREE

Machine Learning Techniques
Professor Doctor Ashish Tendulkar
Indian Institute of Technology Madras
Demonstration: Naive Bayes Classifier

(Refer Slide Time: 00:10)



```
1 # Data loading
2 from sklearn.datasets import fetch_20newsgroups
3
4 # Preprocessing
5 from sklearn.feature_extraction.text import TfidfVectorizer
6
7 # Model/estimator
8 from sklearn.naive_bayes import MultinomialNB
9
10 # Pipeline utility
11 from sklearn.pipeline import make_pipeline
12
13 # Model evaluation
14 from sklearn.metrics import ConfusionMatrixDisplay
15
16 # Plotting library
17 import matplotlib.pyplot as plt
```

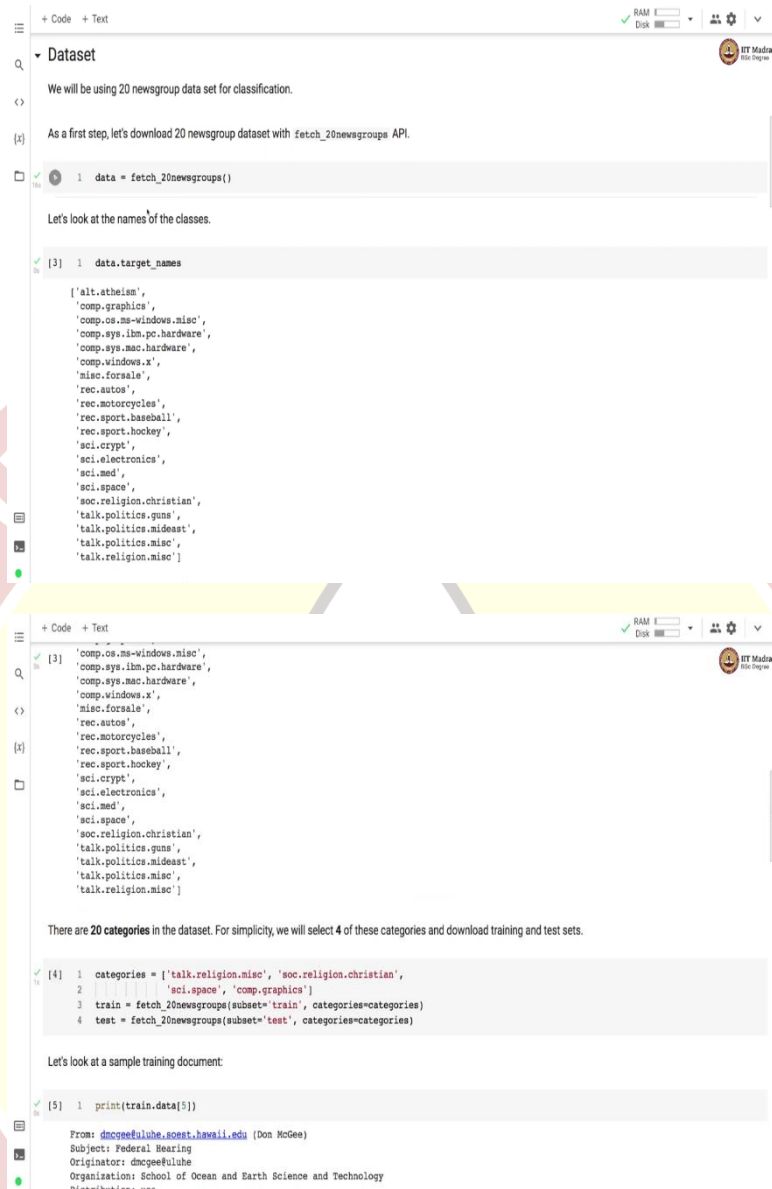
Exercise: Read about [TfidfVectorizer](#) API.

Namaste! Welcome to the next video of Machine Learning Practice Course. In this video, we will use Naive Bayes classifier for classifying text. As you know, Naive Bayes classifier is used for text classification, and spam detection tasks, here is an example as how to perform the text classification with Naive Bayes classifier.

Let us begin by importing necessary libraries. In this case, we will be using 20 newsgroup data, and we will fetch it with fetch _20newsgroup api from sklearn,datasets library. Then we will use TfidfVectorizer in order to convert the text document into a set of numbers, and TfidfVectorizer is part of sklearn.feature _extraction.text library.

Then we will be using multinomial Naive Bayes classifier that we that we load from sklearn.naive _bayes library. Then we have pipeline utility and confusion matrix display for model evaluation. We will be using matplotlib.pyplot for plotting the confusion matrix. So, I would urge all of you to read a bit more about TfidfVectorizer API in the sklearn user guide.

(Refer Slide Time: 01:51)



The image shows two screenshots of a Jupyter Notebook interface. The top screenshot shows the initial steps: importing the `fetch_20newsgroups` API and checking the target names of the classes. The bottom screenshot shows the selection of four categories for training and testing, and a sample training document.

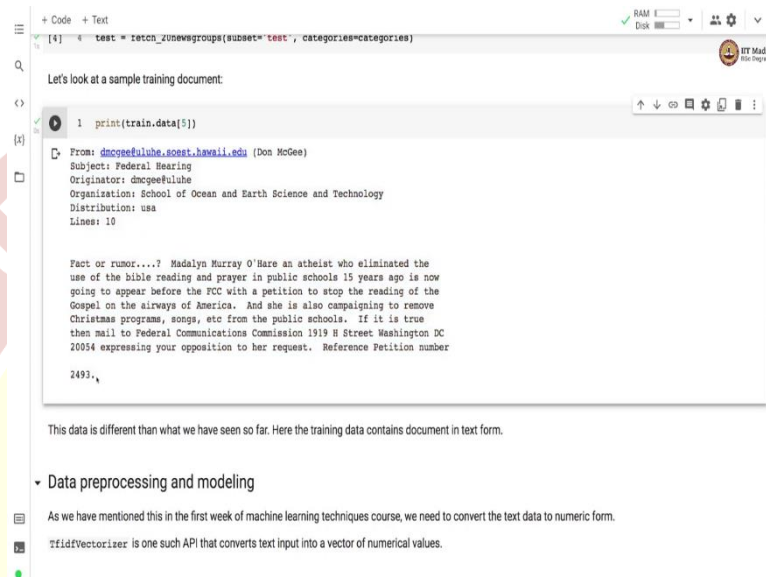
```
+ Code + Text
Dataset
We will be using 20 newsgroup data set for classification.
As a first step, let's download 20 newsgroup dataset with fetch_20newsgroups API.
1 data = fetch_20newsgroups()
Let's look at the names of the classes.
[3] 1 data.target_names
['alt.atheism',
'comp.graphics',
'comp.os.ms-windows.misc',
'comp.sys.ibm.pc.hardware',
'comp.sys.mac.hardware',
'comp.windows.x',
'misc.forsale',
'rec.autos',
'rec.motorcycles',
'rec.sport.baseball',
'rec.sport.hockey',
'sci.crypt',
'sci.electronics',
'sci.med',
'sci.space',
'soc.religion.christian',
'talk.politics.guns',
'talk.politics.mideast',
'talk.politics.misc',
'talk.religion.misc']
There are 20 categories in the dataset. For simplicity, we will select 4 of these categories and download training and test sets.
[4] 1 categories = ['talk.religion.misc', 'soc.religion.christian',
2 'sci.space', 'comp.graphics']
3 train = fetch_20newsgroups(subset='train', categories=categories)
4 test = fetch_20newsgroups(subset='test', categories=categories)
Let's look at a sample training document:
[5] 1 print(train.data[5])
From: dncgee@ulbue.socst.hawaii.edu (Don McGee)
Subject: Federal Hearing
Originator: dncgee@ulbue
Organization: School of Ocean and Earth Science and Technology
HighResolution.mca
```

Here we will be using 20 newsgroup dataset for classification. As a first step, let us download the 20 newsgroup dataset with fetch 20 newsgroup API. So, we simply call `fetch_20newsgroup` and we get the data. Now, let us look at the names of the classes. So, `data.target_names` will give you names of different classes. So as name suggested, 20 newsgroup there are 20 categories in this dataset.

For simplicity, what we will do is we will select four of these categories at random and download training and test dataset corresponding to those four categories. So, we have selected four categories which are `talk.religion.misc`, `soc.religion.Christian`, `sci.space`, and `comp.graphics`.

These four categories we are selecting, and what we will do is we will fetch training data corresponding to these four categories using fetch _20newsgroup API and by specifying the categories. So, we are specifying to subset train and test. So, we get dataset for training, as well as for testing for these four categories.

(Refer Slide Time: 03:17)



```
[4]: test = fetch_20newsgroups(subset='test', categories=categories)
```

Let's look at a sample training document:

```
1 print(train.data[5])
```

From: dncpee@ulbhe.acast.hawaii.edu (Don McGee)
Subject: Federal Hearing
Originator: dncpee@ulbhe
Organization: School of Ocean and Earth Science and Technology
Distribution: usa
Lines: 10

Fact or rumor....? Madalyn Murray O'Hare an atheist who eliminated the use of the bible reading and prayer in public schools 15 years ago is now going to appear before the FCC with a petition to stop the reading of the Gospel on the airways of America. And she is also campaigning to remove Christmas programs, songs, etc from the public schools. If it is true then mail to Federal Communications Commission 1919 H Street Washington DC 20054 expressing your opposition to her request. Reference Petition number 2493.

This data is different than what we have seen so far. Here the training data contains document in text form.

▼ Data preprocessing and modeling

As we have mentioned this in the first week of machine learning techniques course, we need to convert the text data to numeric form.

TfidfVectorizer is one such API that converts text input into a vector of numerical values.

Let us look at a sample training document. So, we look at the sixth example in the training set. And you can see that it is in the form of a text. And this data is different what we have seen so far or what we are used to. Here the training data contains the text. So, this is one training example and it has got text. Then there is some kind of a header, and then there is some number in present in the document.

Now how do we handle it? So, in the first week of machine learning techniques course, we talked about different kinds of datasets like text and images. When we are presented with any non-numeric dataset, we need to perform some kind of pre-processing to get that data into numeric form. So here we will be using TfidfVectorizer to convert text input into a vector of numerical values.

(Refer Slide Time: 04:32)

+ Code + Text

This data is different than what we have seen so far. Here the training data contains document in text form.

Data preprocessing and modeling

As we have mentioned this in the first week of machine learning techniques course, we need to convert the text data to numeric form.

`TfidfVectorizer` is one such API that converts text input into a vector of numerical values.

We will use `TfidfVectorizer` as a preprocessing step to obtain feature vector corresponding to the text document.

We will be using multinomial naive Bayes classifier for categorizing documents from 20newsgroup corpus.

```
1 model = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

Let's train the model.

```
[7] 1 model.fit(train.data, train.target)
```

```
Pipeline(steps=[('tfidfvectorizer', TfidfVectorizer()),
                  ('multinomialnb', MultinomialNB())])
```

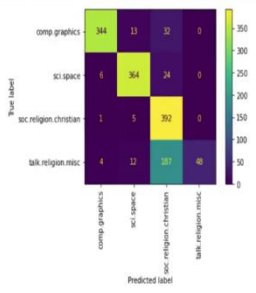
Model evaluation

Let's first predict the labels for the test set and then calculate the confusion matrix for the test set.

```
[8] 1
```

```
2
```

```
[8] 5 display_labels=test.target_names,
6     xticks_rotation='vertical')
7 plt.show()
```



	comp.graphics	sci.space	soc.religion.christian	talk.religion.misc
comp.graphics	344	13	32	0
sci.space	6	364	24	0
soc.religion.christian	1	5	362	0
talk.religion.misc	4	12	107	48

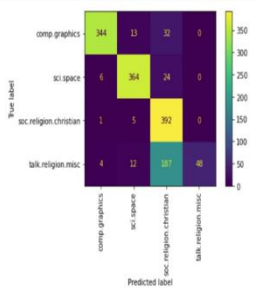
Observe that:

- There is a confusion between documents of class `soc.religion.christian` and `talk.religion.misc`, which is along the expected lines.
- The classes `comp.graphics` and `sci.space` are well separated by such a simple classifier.

Now we have a tool to classify statements into one of these four classes.

Make use of `predict` function on pipeline for predicting category of a test string.

```
[8] 5 display_labels=test.target_names,
6     xticks_rotation='vertical')
7 plt.show()
```



	comp.graphics	sci.space	soc.religion.christian	talk.religion.misc
comp.graphics	344	13	32	0
sci.space	6	364	24	0
soc.religion.christian	1	5	362	0
talk.religion.misc	4	12	107	48

Observe that:

- There is a confusion between documents of class `soc.religion.christian` and `talk.religion.misc`, which is along the expected lines.
- The classes `comp.graphics` and `sci.space` are well separated by such a simple classifier.

Now we have a tool to classify statements into one of these four classes.

Make use of `predict` function on pipeline for predicting category of a test string.

So, what we do is, we define a pipeline object with two steps. The first step is `TfidfVectorizer` that obtains feature vector for a given text document, and then, we pass that feature vector through a multinomial Naive Bayes classifier for learning the classification model for these four categories. In order to train the model, we call the `fit` function on the pipeline object by passing the training data and the labels.

And once the model is learned, we use that to predict the label from test set and calculate the confusion matrix for the test set. So here we make use of confusion matrix display API and we use from `_estimator` method of this object of this class. So, now here, we pass the name, the object corresponding to the model.

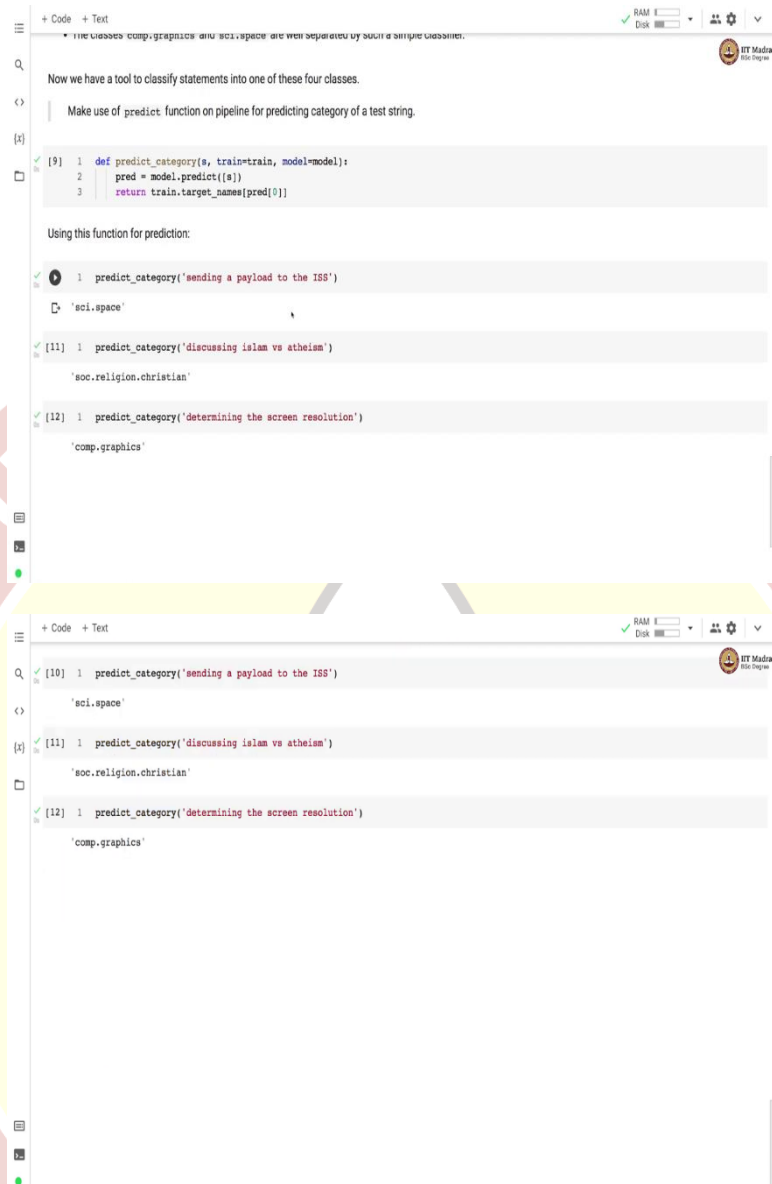
The test feature matrix test feature, test labels and then the names of the classes. And we also specify that on the x axis, the class name should be rotated so that they are in the vertical format. So, here on your screen, what you see is a confusion matrix between four different categories that we have selected out of 20 different categories in 20 newsgroup dataset.

So, what we see is, the `com.graphics` is classified quite well, because there is much less confusion with other classes, and so is the case with `sci.space` class. But if you observe the `soc.religion.Christian` and `talk.religion.misc`, there is some amount of confusion right. So, especially, for `talk.religion.misc`, it is getting confused with the members or with the example strong `soc.religion.Christian`.

So, `soc.religion.Christian` as such is again classified fairly well because the examples from this class are rarely misclassified. Only six examples are misclassified into the other classes. Whereas, in `talk.religion.misc` 187 examples or majority of examples are misclassified into `soc.religion.Christian`.

So, there is a confusion between these two classes, which is along the expected lines because they are talking about some religion-related content. The classes which are `com.graphics` and `sci.space` are well separated by such a simple classifier.

(Refer Slide Time: 08:00)



```
+ Code + Text
• The classes 'comp.graphics' and 'sci.space' are well separated by such a simple classifier.

Now we have a tool to classify statements into one of these four classes.

Make use of predict function on pipeline for predicting category of a test string.

[9] 1 def predict_category(s, train=train, model=model):
    2     pred = model.predict([s])
    3     return train.target_names[pred[0]]

Using this function for prediction:

1 predict_category('sending a payload to the ISS')
'sci.space'

[11] 1 predict_category('discussing islam vs atheism')
'soc.religion.christian'

[12] 1 predict_category('determining the screen resolution')
'comp.graphics'

+ Code + Text
[10] 1 predict_category('sending a payload to the ISS')
'sci.space'

[11] 1 predict_category('discussing islam vs atheism')
'soc.religion.christian'

[12] 1 predict_category('determining the screen resolution')
'comp.graphics'
```

Now, we have a tool to classify statements into one of these four classes. So, for that we make use of predict function on the pipeline object for predicting category of the test string. So, here we define a small function called predict_category that take a statement, the training set and the name of the model.

And remember, we have set these two parameters to their default values. So, we first called the predict function on the model by using the statement as an argument, we get a prediction. And for this prediction, we find out what is the name of the class through train.target_names. So, let us use this function for prediction.

So, you want to predict category of this statement which is sending a payload to ISS which is International Space Station, and this is clearly you know, a statement from sci.space class. Then, we use another statement discussing Islam versus atheism. This is clearly related to religion, so that is why it is classified into religious content.

And the third statement, which is determining the screen resolution is part of com graphics. So, this is how the statements are classified into respective classes. So, in this video, we implemented Naive Bayes classifier in order to classify text into different categories. And we can use it in different tests classification task.

