CS 696 Intro to Big Data: Tools and Methods
Spring Semester, 2019
Doc 11 Scikit Learn, Bayes
Feb 28, 2019

Scikit Learn

https://scikit-learn.org/stable/index.html

Common Python Machine Learning library

Included in anaconda jupyter notebook install

Basic Operation

Choose which model to use

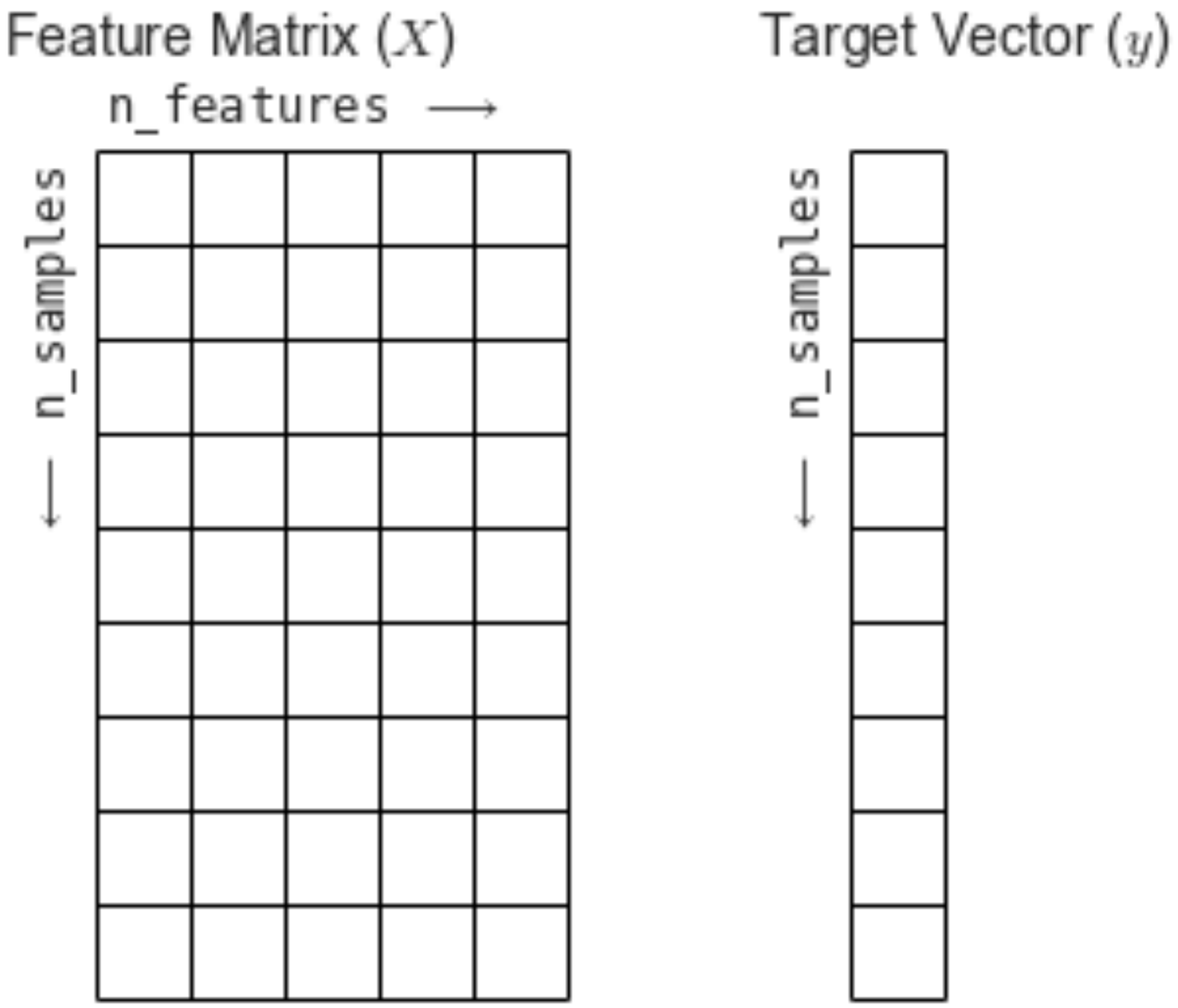
Choose model hyperparameters

Arrange data into a features matrix and target vector

Fit the model to data using fit()

Apply the model to new data

Feature Matrix & Target Vector



Feature Matrix - independent variables

Target Vector - dependent variable

```
import numpy as np
from sklearn.linear_model import LinearRegression

rng = np.random.RandomState(42)
x = 10 * rng.rand(50)
y = 2 * x - 1 + rng.randn(50)   # y = Target Vector

X = x[:,np.newaxis]             # X = Feature matrix

model = LinearRegression(fit_intercept=True)
model.fit(X, y)

xfit = np.linspace(-1,11, num=50)
Xfit = xfit[:, np.newaxis]                  # New Data
yfit = model.predict(Xfit)
```

Bayes Theorem

P(A) = probability of A
P(B) = probability of B
P(A | B) = probability of A given B is true
P(B | A) = probability of B given A is true

P(A | B) = P(B | A) * P(A)/ P(B)

	Female	Male	Total
Teacher	8	12	20
Student	32	48	80
Total	40	60	100

P(Teacher | Male) ?

P(Male | Teacher) = 12/20
P(Teacher) = 20/100
P(Male) = 60/100

P(Teacher | Male) = P(Male | Teacher)*P(Teacher)/P(Male)
= 12/20 * 20/100 / (60/100)
= 12/20 * 20/100 * 100/60
= 12/20 * 20 / 60
= 12/20 * 2/6
= 12/20 * 1/3
= 12/60

Classification

Given a datum which group does it belong

Give each group a label (identifier)

Given a datum label should it have

Bayesian Classification

P(A | B) = P(B | A) * P(A)/ P(B)

P(Label | features) = P(features | Label) * P(Label)/ P(features)

If we have two labels

P(L1 | features)/P(L2 | features) = P(features | L1) * P(L1)/P(features | L2) * P(L2)

P(features | Li)

Need a model of how the data is generated to compute this

Gaussian Naive Bayes

Data from each label is drawn from a simple Gaussian distribution

Naive

Because we are making assumption about the distribution

Some Data

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

X, y = make_blobs(100, 2, centers=2, random_state=2, cluster_std=1.5)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='coolwarm');
```


```
y[0:5]          array([0, 0, 0, 1, 1])
```

```
X[0:5]
```

```
array([[ 0.92141506, -9.98499137],
       [-5.26927614, -9.6186543 ],
       [-0.45292089, -6.04316334],
       [-0.0856312 , -2.16867404],
       [ 1.53194956, -0.36022153]])
```

```
sklearn.datasets.make_blobs(
    n_samples=100,
    n_features=2,
    centers=None,               #The number of centers to generate
    cluster_std=1.0,
    center_box=(-10.0, 10.0),
    shuffle=True,
    random_state=None)      #Determines random number generation
```

Training the Model

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X, y);
```

Test Data

```
rng = np.random.RandomState(0)
Xnew = [-6, -14] + [14, 18] * rng.rand(2000, 2)
```

X values [-6, 8]

Y values [-14, 4]

```
pairs = rng.rand(3, 2)
[-10, 50]* pairs
```

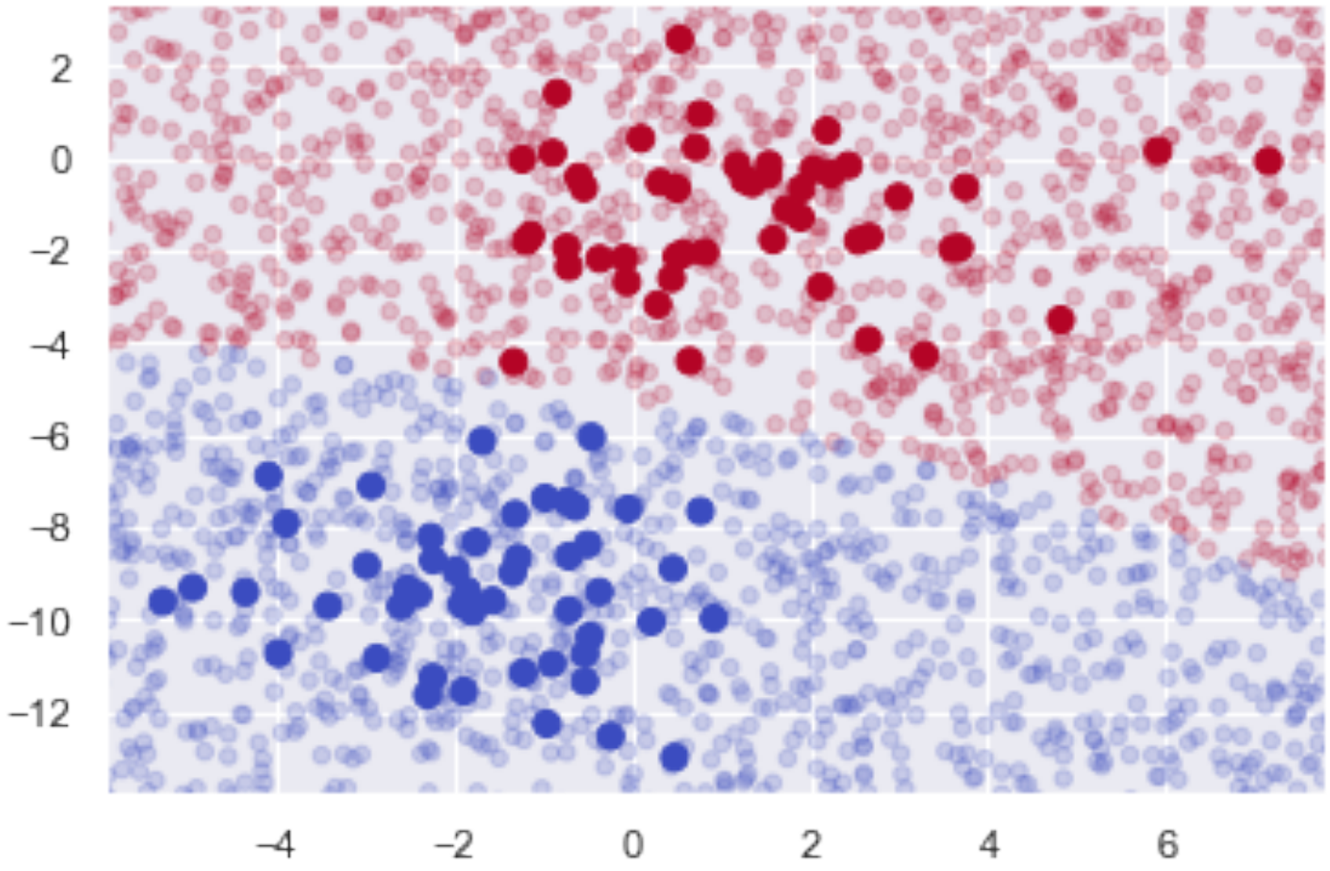
```
array([[-9.47419519, 43.23204192],
       [-6.16054049, 42.07036184],
       [-2.09692145, 43.70144114]])
```

```
Xnew[0:4]
```

```
array([[ 1.68338905, -1.12659141],
       [ 2.43868727, -4.19210271],
       [-0.06883281, -2.37390596],
       [ 0.12622096,  2.05191401]])
```

Predict

```
ynew = model.predict(Xnew)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='coolwarm')
lim = plt.axis()
plt.scatter(Xnew[:, 0], Xnew[:, 1], c=ynew, s=20, cmap='coolwarm', alpha=0.2)
plt.axis(lim);
```




```
ynew[0:10]
```

```
array([1, 1, 1, 1, 1, 1, 1, 0, 1, 1])
```

When to Use Naive Bayes

Generally not as good as more complex model

Advantages

They are extremely fast for both training and prediction

They provide straightforward probabilistic prediction

They are often very easily interpretable

They have very few (if any) tunable parameters

When to Use Naive Bayes

When the naive assumptions actually match the data (very rare in practice)

For very well-separated categories, when model complexity is less important

For very high-dimensional data, when model complexity is less important

Text Analysis & Bayes

How to use Bayes to analysis text?

Assume English has 5 words - at, bat, cat, dog, enough

Convert sentences into vectors

bat at dog	[0, 1, 0, 1, 0]
enough dog enough	[0, 0, 0, 1, 2]
sad dog sad	[0, 0, 0, 1, 0]

Text feature extraction

Tokenizing strings

Give an integer id for each possible token

Counting the occurrences of tokens in each document.

Normalizing and Weighting

Give less weight to tokens in majority of samples / documents

Reviews

Group the reviews

- Positive
- Neutral
- Negative

Collect all words in all reviews

- Drop unimportant words - a, the, at, ... — Stop words
- Convert words to their root
  - laughing, laughed -> laugh
  - ran, run -> run
- Drop punctuation
- ...

Order the remaining words

Convert each review into vector

Review Groups become set of vectors

Use Bayes to predict which group a new review is in

Issue - sparse vectors

All the text may have 100,000's of distinct words

A single message may only have 40 words

message converts to a vector of size 100,000
But only has 40' ones in it

Scikit represents the vector using a sparse representation

How to Convert to Words to Vectors

CountVectorizer

Text documents -> matrix of token counts

HashingVectorizer

TfidfTransformer

Count matrix -> normalized tf or tf-idf representation

TfidfVectorizer

Text documents -> normalized tf or tf-idf representation

Same as CountVectorizer + TfidfTransformer

Scikit learn functions

Tokenize

Remove stop words

Drop punctuation

~~Convert words to roots~~

Vectorize

Tf - Term frequency

tf(term, document) or tf(t,d)

Raw count f(t,d)

Boolean

tf(term, document) = 1 if term in document, 0 otherwise

Length adjusted

f(t,d)/ (number of words in d)

Logarithmically scaled

tf(t, d) = log(1 + f(t,d)

idf - Inverse Document Frequency

idf - Inverse Document Frequency

Measure of how much information a word provides

If word is in every document - provides no information

Multiple ways to compute idf(t,D)

D = set of documents

tf–idf

tfidf(t,d,D) = tf(t,d) * idf(t,D)

High value is reached when

- Term occurs a lot in a document
- The term does not occur in many documents in the corpus

As a term appears in more documents in the corpus

- tf-idf get closer to zero

CountVectorizer

Counts the words

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()

corpus = [ 'This is the first document.',
           'This is the second second document.',
           'And the third one.',
           'Is this the first document?']
```

```
X = vectorizer.fit_transform(corpus)
vectorizer.get_feature_names()   #['and', 'document', 'first', 'is', 'one','second', 'the', 'third', 'this']


  X.toarray()
  array([[0, 1, 1, 1, 0, 0, 1, 0, 1],
         [0, 1, 0, 1, 0, 2, 1, 0, 1],
         [1, 0, 0, 0, 1, 0, 1, 1, 0],
         [0, 1, 1, 1, 0, 0, 1, 0, 1]]...)

 vectorizer.vocabulary_.get('document')
       1
```

```
vectorizer.transform(['Something completely new.']).toarray()
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0]
```

CountVectorizer Methods

build_analyzer()	Return a callable that handles preprocessing and tokenization
build_preprocessor()	Return a function to preprocess the text before tokenization
build_tokenizer()	Return a function that splits a string into a sequence of tokens
decode(doc)	Decode the input into a string of unicode symbols
fit(raw_documents[, y])	Learn a vocabulary dictionary of all tokens in the raw documents.
fit_transform(raw_documents[, y])	Learn the vocabulary dictionary and return term-document matrix.
get_feature_names()	Array mapping from feature integer indices to feature name
get_params([deep])	Get parameters for this estimator.
get_stop_words()	Build or fetch the effective stop words list
inverse_transform(X)	Return terms per document with nonzero entries in X.
set_params(**params)	Set the parameters of this estimator.
transform(raw_documents)	Transform documents to document-term matrix.

TfidfVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()

corpus = [ 'This is the first document.',
        'This is the second document.',
        'And the third one.',
        'Is this the first document?']
X = vectorizer.fit_transform(corpus)
vectorizer.get_feature_names()
```

```
import pandas as pd
pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names())
```

	and	document	first	is	one	second	the	third	this
0	0.000000	0.438777	0.541977	0.438777	0.000000	0.000000	0.358729	0.000000	0.438777
1	0.000000	0.404129	0.000000	0.404129	0.000000	0.633146	0.330402	0.000000	0.404129
2	0.552805	0.000000	0.000000	0.000000	0.552805	0.000000	0.288477	0.552805	0.000000
3	0.000000	0.438777	0.541977	0.438777	0.000000	0.000000	0.358729	0.000000	0.438777

Ngrams

Word pairs may be informative

```
from sklearn.feature_extraction.text import CountVectorizer
bigram_vectorizer = CountVectorizer(ngram_range=(1, 3), token_pattern=r'\b\w+\b', min_df=1)
analyze = bigram_vectorizer.build_analyzer()
analyze('Bi-grams are cool!')
```

```
['bi',
'grams',
'are',
'cool',
'bi grams',
'grams are',
'are cool',
'bi grams are',
'grams are cool']
```

Text Example

```
from sklearn.datasets import fetch_20newsgroups

data = fetch_20newsgroups()
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',
```

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

From: guykuo@carson.u.washington.edu (Guy Kuo)
Subject: SI Clock Poll - Final Call
Summary: Final call for SI clock reports
Keywords: SI,acceleration,clock,upgrade
Article-I.D.: shelley.1qvfo9INNc3s
Organization: University of Washington
Lines: 11
NNTP-Posting-Host: carson.u.washington.edu

A fair number of brave souls who upgraded their SI clock oscillator have
shared their experiences for this poll. Please send a brief message detailing
your experiences with the procedure. Top speed attained, CPU rated speed,
add on cards and adapters, heat sinks, hour of usage per day, floppy disk
functionality with 800 and 1.4 m floppies are especially requested.

I will be summarizing in the next two days, so please add to the network
knowledge base if you have done the clock upgrade and haven't answered this
poll. Thanks.

Guy Kuo <guykuo@u.washington.edu>
```

Fetch Small Train and Test data

```
categories = ['talk.religion.misc', 'soc.religion.christian', 'sci.space',
              'comp.graphics']
train = fetch_20newsgroups(subset='train', categories=categories)
test = fetch_20newsgroups(subset='test', categories=categories
```

Train and Test Data Structure

data

DESCR

print(train.DESCR) - print description of data and how to process it

filenames

Where data is stored

target

target_names

array([2, 0, 1, ..., 1, 2, 1])

['comp.graphics', 'sci.space', 'soc.religion.christian', 'talk.religion.misc']

train.target.shape

(2153,)

train.target[0:5]

array([2, 2, 1])

Index in target_names

Train and Test Data Structure

pd.DataFrame({"data":train.data, "Category": train.target}).head()

	data	Category
0	From: jono@mac-ak-24.rtsg.mot.com (Jon Ogden)\...	2
1	From: MANDTBACKA@finabo.abo.fi (Mats Andtbacka...	2
2	From: mccall@mksol.dseg.ti.com (fred j mccall ...	1
3	From: revdak@netcom.com (D. Andrew Kille)\nSub...	2
4	From: hl7204@eehp22 (H L)\nSubject: Re: Graphi...	0

The Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

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

Pipeline

- Series of transforms
- Ended by estimator - needs to implement fit function

Training the Model

```
model.fit(train.data, train.target)
```

Classifying the Test Data

labels = model.predict(test.data)

What is labels?

But was it good at classifying?

What is labels?

labels array([2, 0, 1, ..., 1, 2, 1])

Array of predicted labels/Categories of each element in the test data

Need to use test.target_names to see actual label

test.target_names[labels[1]]

'comp.graphics'

What is labels?

pd.DataFrame({"data":test.data, "Predicted": labels.data}).head()

	data	Predicted
0	From: livesey@solntze.wpd.sgi.com (Jon Livesey...	2
1	From: ddennis@nyx.cs.du.edu (Dave Dennis)\nSub...	0
2	From: isaackuo@jell-o.berkeley.edu (Isaac Kuo)...	1
3	From: fitz@cse.ogi.edu (Bob Fitzsimmons)\nSubj...	0
4	From: vis@world.std.com (Tom R Courtney)\nSubj...	1

print(test.data[1])

From: ddennis@nyx.cs.du.edu (Dave Dennis)
Subject: Re: Adobe Type Manager - what good is it??
Organization: University of Denver, Dept. of Math & Comp. Sci.
Lines: 29

menchett@dws015.unr.edu (Peter J Menchetti) writes:

>The subject says it all. I bought Adobe Type Manager and find it completely
>useless. I ftped some atm fonts and couldn't install them. What's the use?
>Are you supposed to be able to convert ATM fonts to Truetype?

>If there's anyone out there who has this program and actually finds it
>useful, enlighten me!

>Pete

There are some tricks to installing ATM to windows... install them first
to dos, then run the ATM control panel to get them into windows.

The best reason for ATM is that Adobe IS the standard. Truetype is a
failed MS venture to undercut Adobe when Adobe was being nasty about

The Prediction

train.target_names[labels[1]]

'comp.graphics'

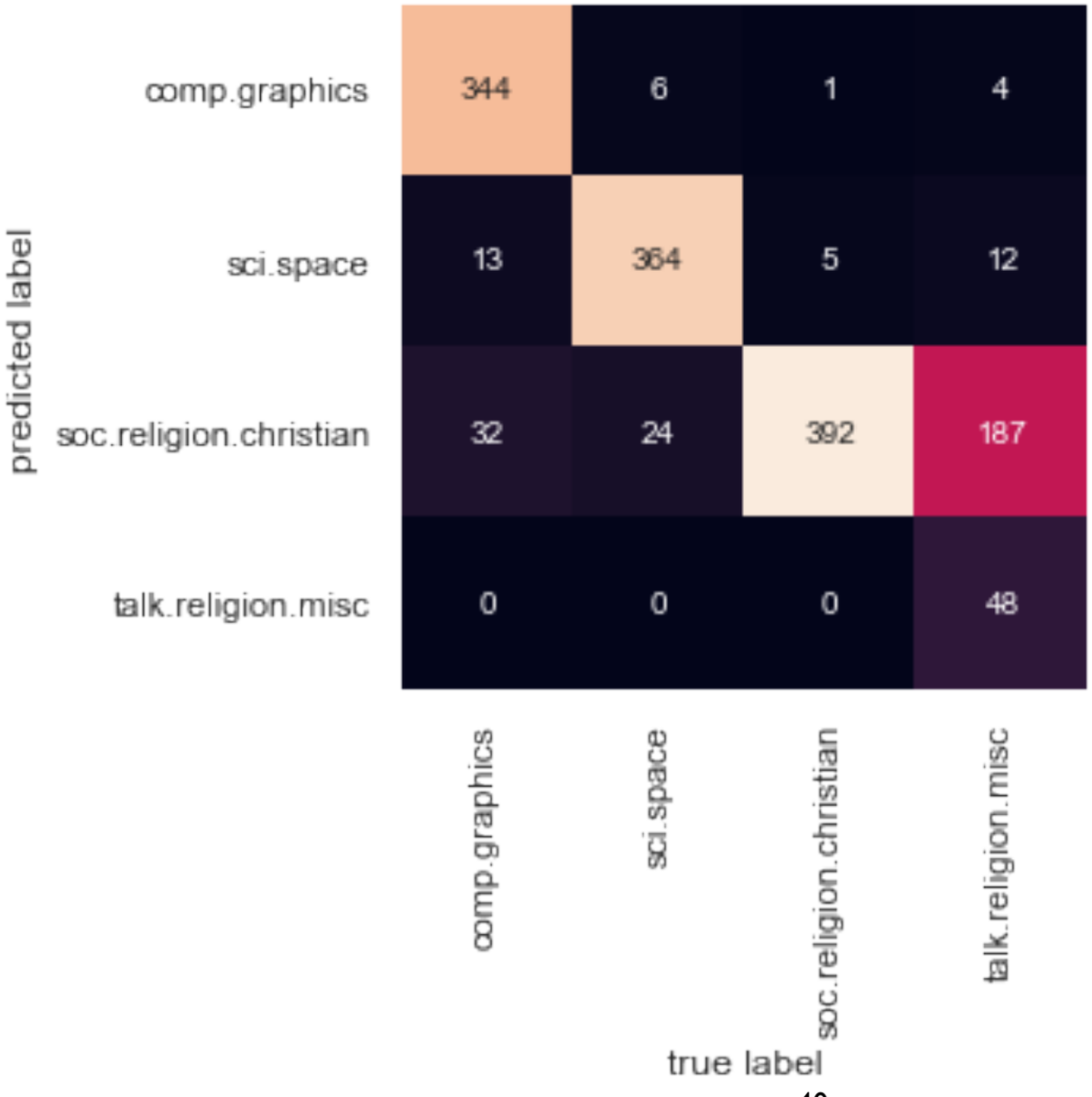
So it got location 1 correct

But how many are there in the test data?

test.target.shape

(1432,)

```
from sklearn.metrics import confusion_matrix
error_matrix = confusion_matrix(test.target, labels)
sns.heatmap(error_matrix.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=train.target_names, yticklabels=train.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label');
```



Examining Where Classification Failed

Or how do you write confusion_matrix

test.data

Actual data

test.target

Actual classification

labels.data

Predicted classification

```
pd.DataFrame({"data":test.data, "Actual": test.target, "Predicted": labels.data}).head()
```

	data	Actual	Predicted
0	From: livesey@solntze.wpd.sgi.com (Jon Livesey...	3	2
1	From: ddennis@nyx.cs.du.edu (Dave Dennis)\nSub...	0	0
2	From: isaackuo@jell-o.berkeley.edu (Isaac Kuo)...	1	1
3	From: fitz@cse.ogi.edu (Bob Fitzsimmons)\nSubj...	0	0
4	From: vis@world.std.com (Tom R Courtney)\nSubj...	1	1

```
results = pd.DataFrame({"data":test.data, "Actual": test.target, "Predicted": labels.data})
graphics_as_christian =results[ (results.Actual == 0) & (results.Predicted == 2)]
graphics_as_christian.head()
```

	data	Actual	Predicted
27	From: val@fcom.cc.utah.edu (Val Kartchner)\nSu...	0	2
54	From: jk87377@lehtori.cc.tut.fi (Kouhia Juhana...	0	2
57	From: grady@netcom.com (1016/2EF221)\nSubject:...	0	2
77	From: grady@netcom.com (1016/2EF221)\nSubject:...	0	2
169	From: cstxqbe@dcs.warwick.ac.uk (Kate Kingman)...	0	2

```
test.target_names
```

```
['comp.graphics', 'sci.space', 'soc.religion.christian', 'talk.religion.misc']
```

```
print(graphics_as_christian['data'][54])
```

From: jk87377@lehtori.cc.tut.fi (Kouhia Juhana)
Subject: Re: Oh make up your mind!! (was: Re: XV problems)
Organization: Tampere University of Technology
Lines: 40
Distribution: world
NNTP-Posting-Host: cc.tut.fi

In article <1993Apr30.182605.5999@nessie.mcc.ac.uk>
C.C.Lilley@mcc.ac.uk writes:
>
>>XV allows this feature, but I don't recommend to use it with the
>>mentioned type images.
>
>Ah! now we see thew problem! First you want to extend xv to allow
>editing of 8 bit previews of 24 bit images. Then I point out problems
>with this. Now you are saying there is no problem because you,
>personally, happen not to use those parts of the program that cause
>the problem!!
[..see previous article on this debate for the rests..]

I can see XV-3.00 agree with my view in cases you don't -- even I say
my personal opinion (as above), it doesn't mean that it is not most