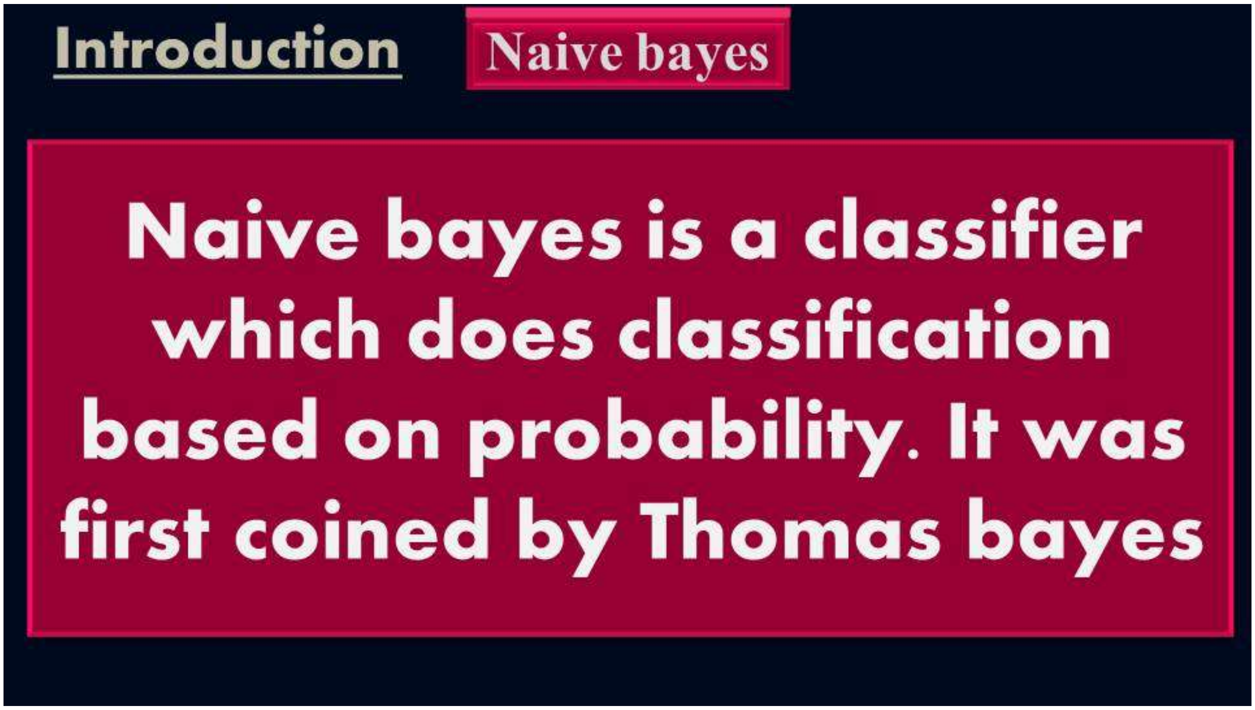
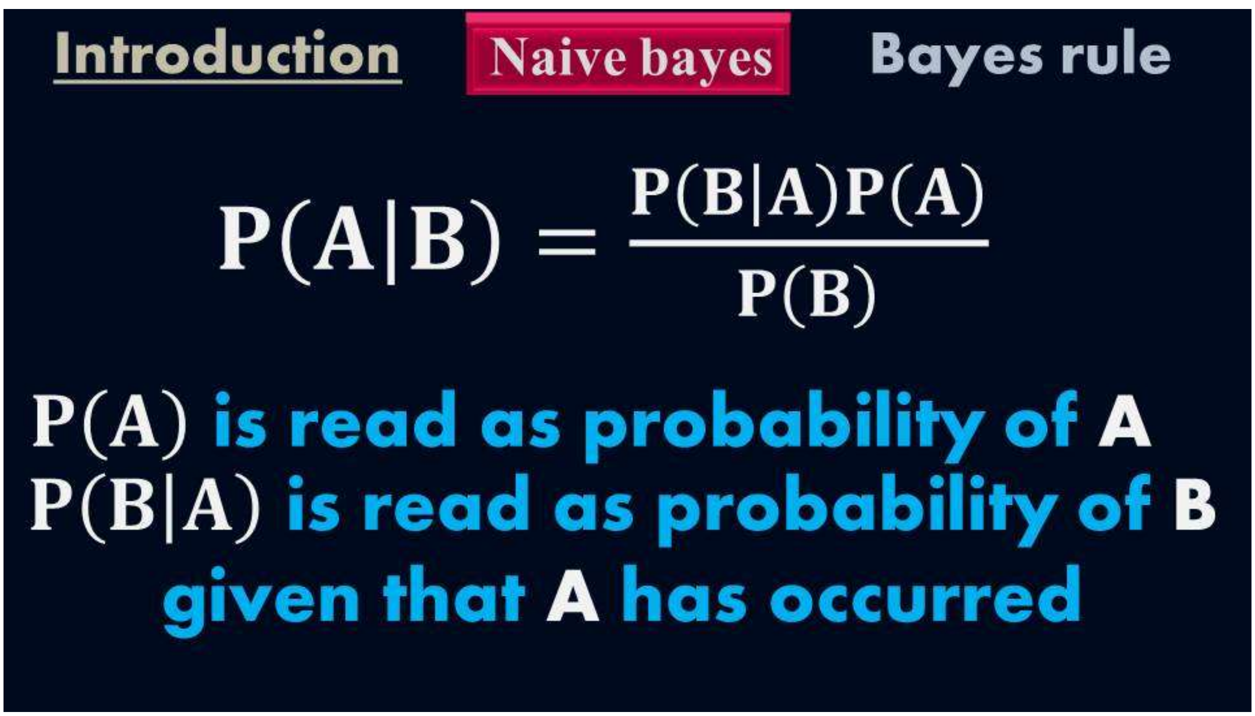

Introduction
Naive bayes
Naive bayes is a classifier which does classification based on probability. It was first coined by Thomas bayes


Introduction
Naive bayes
Bayes rule
$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
P(A) is read as probability of A
P(B|A) is read as probability of B given that A has occurred

Introduction | **Naive bayes** | Bayes rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(\text{Outcome given that we know some Evidence}) = \frac{P(\text{Evidence given that we know some Outcome})P(\text{Outcome})}{P(\text{Evidence})}$$

$$P(\text{Covid given that Test positive}) = \frac{P(\text{Test positive given Covid})P(\text{Covid})}{P(\text{Testing positive})}$$

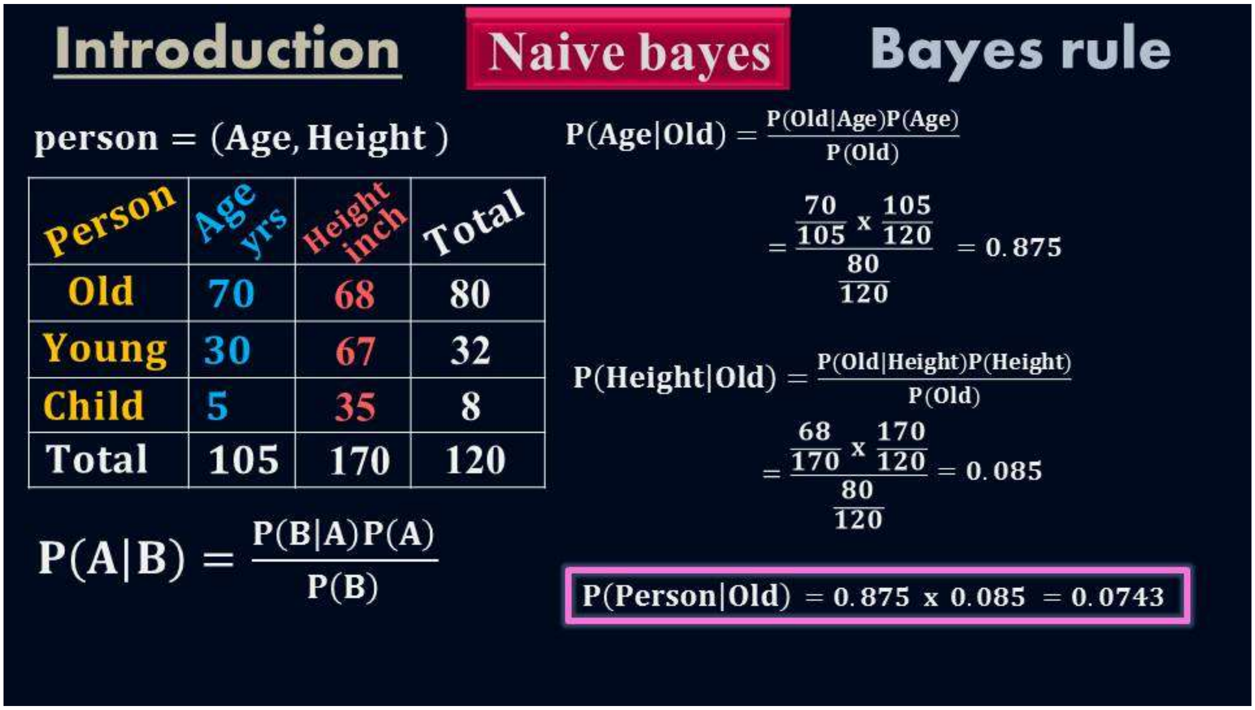
Introduction | **Naive bayes** | Bayes rule

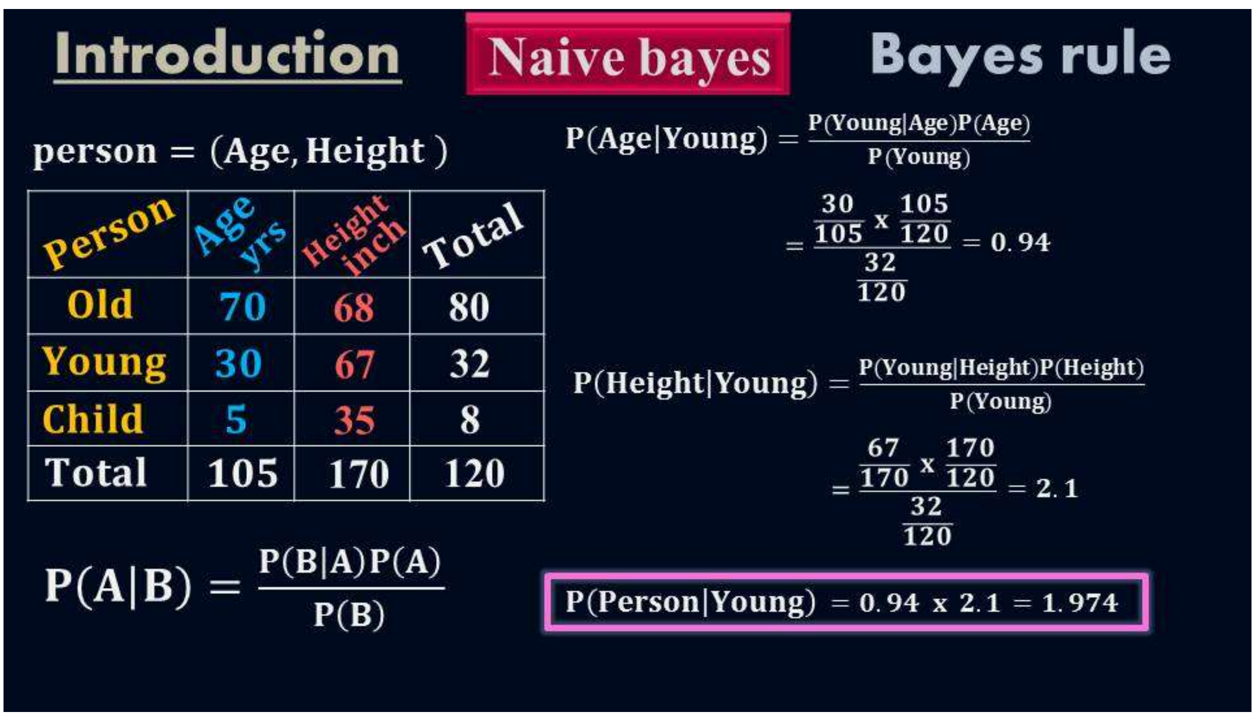
There's 70% chance that you might suffer covid if you don't follow SOPs. There are 90% of people who suffered covid as they kept less than three feet distance. It's 100% sure that you will suffer covid if you don't keep distance of people. But what will be chance that you might not suffer covid even if you keep distance less than three feet.

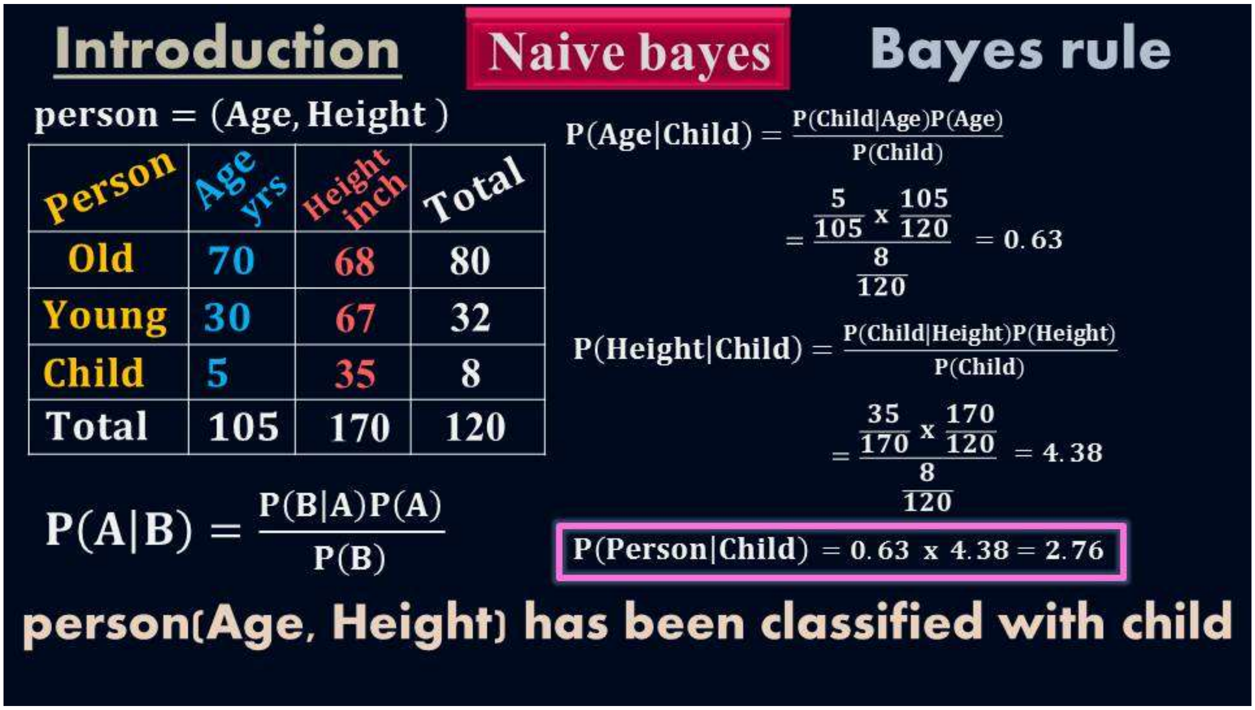
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \Rightarrow P(\text{covid}|\text{distance}) = \frac{P(\text{distance}|\text{covid})P(\text{covid})}{P(\text{distance})}$$

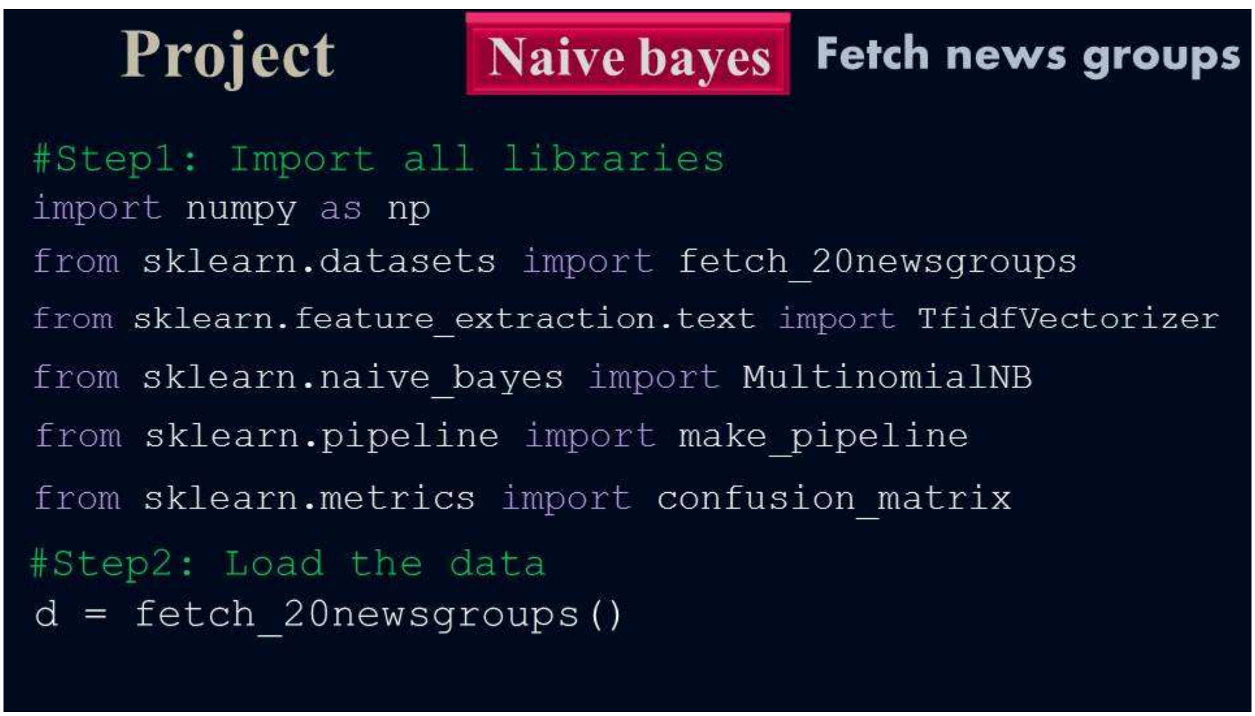
$$= \frac{1.0 \times 0.7}{0.9} = 0.63$$

There's 63% chance that you may not suffer covid even if you don't keep three feet distance

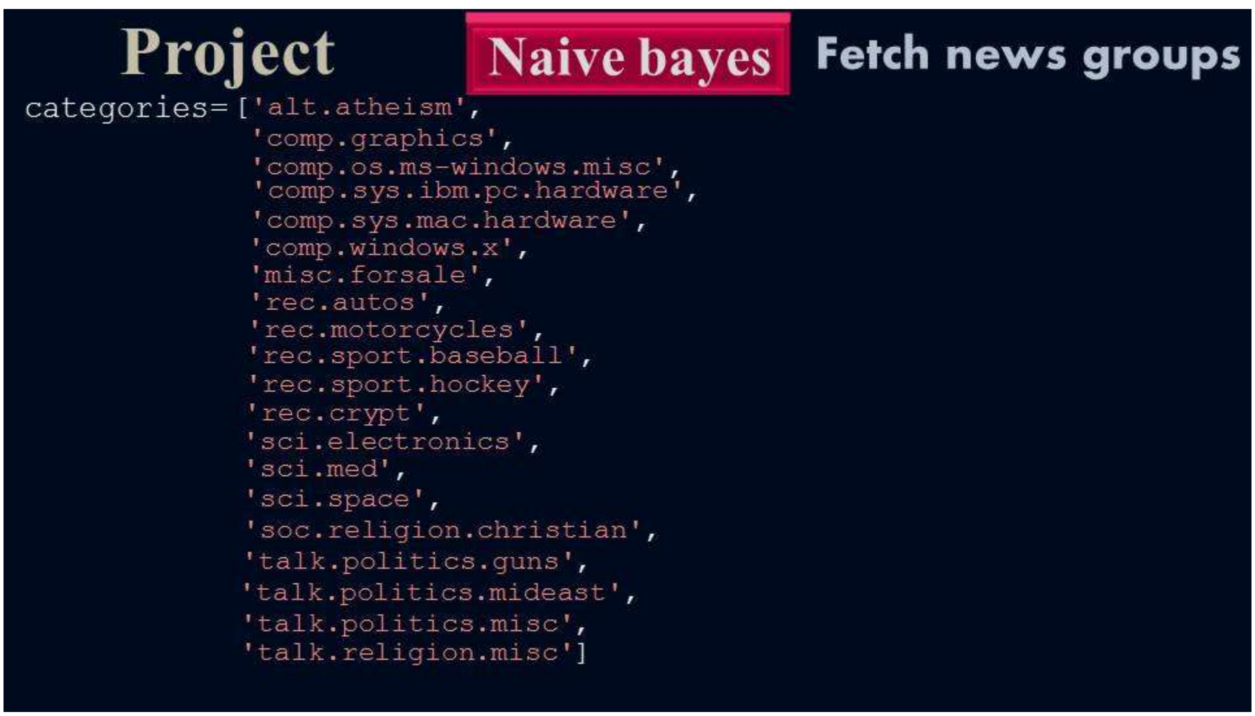

Introduction
Naive bayes
Bayes rule
person = (Age, Height)
P(Age|Old) = P(Old|Age)P(Age) / P(Old)
person | Age yrs | Height inch | Total
Old | 70 | 68 | 80
Young | 30 | 67 | 32
Child | 5 | 35 | 8
Total | 105 | 170 | 120
= (70/105 x 105/120) / (80/120) = 0.875
P(Height|Old) = P(Old|Height)P(Height) / P(Old)
= (68/170 x 170/120) / (80/120) = 0.085
P(A|B) = P(B|A)P(A) / P(B)
P(Person|Old) = 0.875 x 0.085 = 0.0743


Introduction
Naive bayes
Bayes rule
person = (Age, Height)
P(Age|Young) = P(Young|Age)P(Age) / P(Young)
person | Age yrs | Height inch | Total
Old | 70 | 68 | 80
Young | 30 | 67 | 32
Child | 5 | 35 | 8
Total | 105 | 170 | 120
= (30/105 x 105/120) / (32/120) = 0.94
P(Height|Young) = P(Young|Height)P(Height) / P(Young)
= (67/170 x 170/120) / (32/120) = 2.1
P(A|B) = P(B|A)P(A) / P(B)
P(Person|Young) = 0.94 x 2.1 = 1.974


Introduction
Naive bayes
Bayes rule
person = (Age, Height)
person
Age yrs
Height inch
Total
Old 70 68 80
Young 30 67 32
Child 5 35 8
Total 105 170 120
P(Age|Child) = P(Child|Age)P(Age) / P(Child)
= (5/105 x 105/120) / (8/120) = 0.63
P(Height|Child) = P(Child|Height)P(Height) / P(Child)
= (35/170 x 170/120) / (8/120) = 4.38
P(A|B) = P(B|A)P(A) / P(B)
P(Person|Child) = 0.63 x 4.38 = 2.76
person(Age, Height) has been classified with child


Project
Naive bayes
Fetch news groups
#Step1: Import all libraries
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
#Step2: Load the data
d = fetch_20newsgroups()


Project
Naive bayes
Fetch news groups
#Explore the data
d.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']


Project
Naive bayes
Fetch news groups
categories=['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',
'rec.crypt',
'sci.electronics',
'sci.med',
'sci.space',
'soc.religion.christian',
'talk.politics.guns',
'talk.politics.mideast',
'talk.politics.misc',
'talk.religion.misc']

Project **Naive bayes** **Fetch news groups**

```
#Step3: Clean the data: data is already clean
#Step4: Split the data in train and test
train = fetch_20newsgroups(subset='train',categories=categories)
test = fetch_20newsgroups(subset='test',categories=categories)
print(len(train.data))
print(len(test.data))
>> 11314
   7532
#Step5: Create the model based on Multinomial Naive bayes
model = make_pipeline(TfidfVectorizer(),MultinomialNB())
#Step6: Train the model
model.fit(train.data,train.target)
#Step7: Prediction
prediction = model.predict(test.data)
```

Project **Naive bayes** **Fetch news groups**

```
#Step8: Final evaluation
def pred_category(s, train=train, model=model):
  prediction = model.predict([s])
  return train.target_names[prediction[0]]
pred_category('Jesus Christ')
>> 'soc.religion.christian'
pred_category('International space station')
>> 'sci.space'
pred_category('lamborghini is better than ferrari')
>> 'rec.autos'
pred_category('President of America')
>> 'talk.politics.misc'
```

COMPLETE CODES ON ONE PAGE

Project **Naive bayes** **Fetch news groups**

```
#All codes on one page
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
d = fetch_20newsgroups()
train = fetch_20newsgroups(subset='train',categories=categories)
test = fetch_20newsgroups(subset='test',categories=categories)
model = make_pipeline(TfidfVectorizer(),MultinomialNB())
model.fit(train.data,train.target)
prediction = model.predict(test.data)
```