Text Classification

Learning Objectives

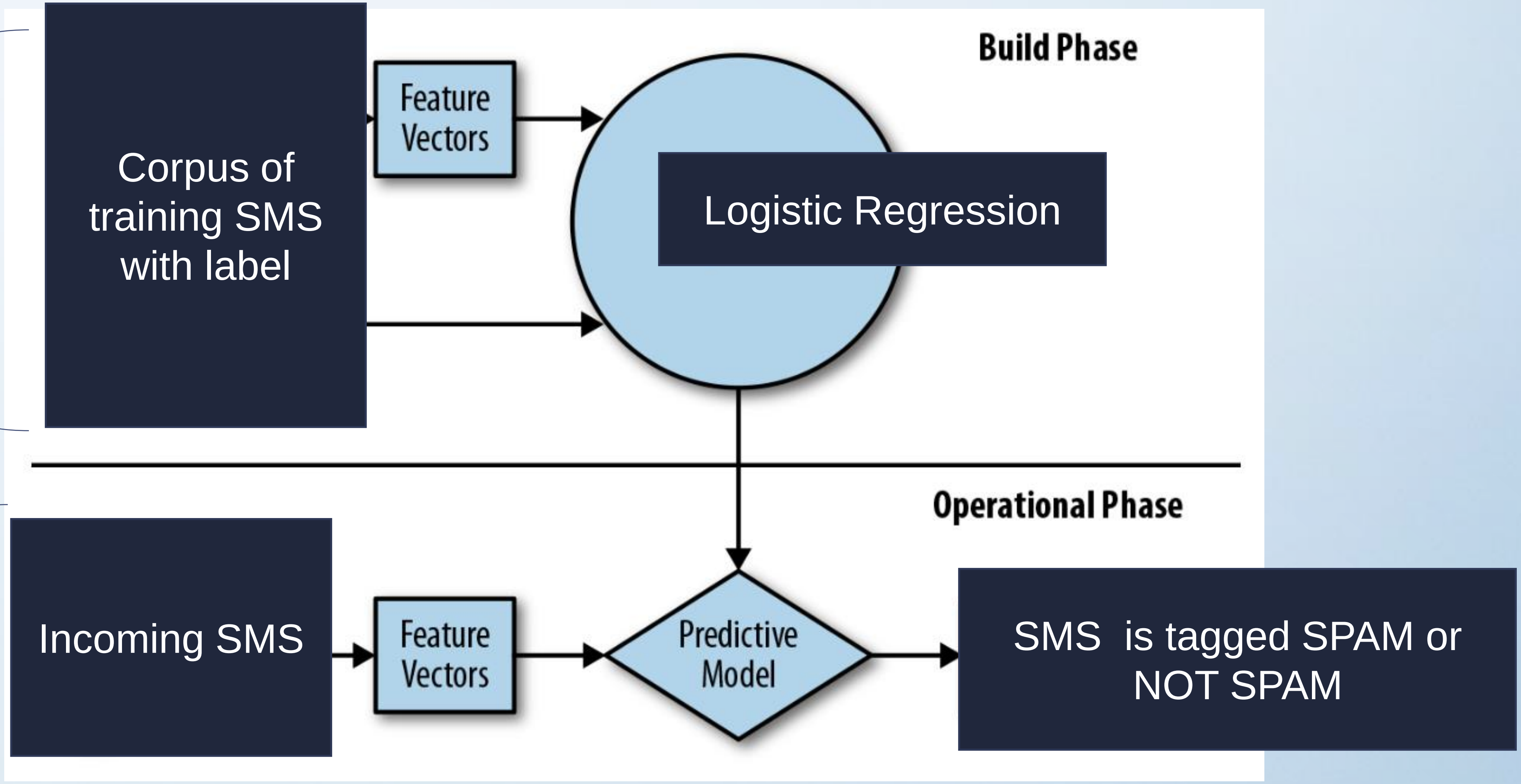
- Describe each component in Text Classification System
- Calculate classification evaluation metric
- Build and use Logistic Regression model in Text Classification System
- Build and use Naive Bayes model in Text Classification System

Logistic Regression

Example

Data with Label

Data without Label



Exercise 1: Build Logistic Regression model to predict ham or spam for SMS message

Step 1:

Read the SMSSpamCollection.txt and pre-process the data

- Remove all the words with numbers and pure numbers. e.g. 21^{st}, 2005
- Remove all the punctuation. e.g. !, ?
- Convert capital letter to small letter

	label	new_text
0	ham	ok lar joking wif u oni
1	spam	free entry in a wkly comp to win fa cup final tkts may text fa to to receive entry questionstd txt ratetcs apply s
2	ham	u dun say so early hor u c already then say
3	ham	nah i dont think he goes to usf he lives around here though
4	spam	freemsg hey there darling its been weeks now and no word back id like some fun you up for it still tb ok xxx std chgs to send £ to rcv

Exercise 1: Build Logistic Regression model to predict ham or spam for SMS message

Step 2:

Split the dataset into training set and test set

- Test dataset = 30% of observation and Training dataset = 70% of observation
- Random state =42, so we all get the same random train and test split

```
The size of original dataset: (5571,)
The size of training dataset: (3899,)
The size of test dataset: (1672,)
```

Exercise 1: Build Logistic Regression model to predict ham or spam for SMS message

Step 3:

Convert the text to vectors using count vectorizer

```
The dimensions of the training set: (3899, 6663)
The dimensions of the test set: (1672, 6663)
The features:
 ['aa' 'aah' 'aaooooright' ... 'zoom' 'zouk' 'üll']
```

Exercise 1: Build Logistic Regression model to predict ham or spam for SMS message

Step 4:

Fit Logistic Regression model on training data and apply the fitted model to test data. Predict the test data.

```
print(list(y_test[:10]))
print(list(y_pred_cv[:10]))
```

```
['ham', 'spam', 'ham', 'spam', 'ham', 'ham', 'ham', 'ham', 'ham', 'ham']
['ham', 'spam', 'ham', 'spam', 'ham', 'ham', 'ham', 'ham', 'ham', 'ham']
```

Exercise 1: Build Logistic Regression model to predict ham or spam for SMS message

Step 5:

Evaluate the mode. Decide how good the model is by calculating various metrics

- Confusion Metrix
- Precision
- Recall
- F1-score
- Accuracy

Exercise 1: Build Logistic Regression model to predict ham or spam for SMS message

Step 5 (cont.):

Evaluate the mode. Decide how good the model is by calculating various metrics

```
array([[1451,    2],
       [  32,  187]])
```

```
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99      1453
        spam       0.99      0.85      0.92       219

    accuracy                           0.98      1672
   macro avg       0.98      0.93      0.95      1672
weighted avg       0.98      0.98      0.98      1672
```

Exercise 1: Build Logistic Regression model to predict ham or spam for SMS message

Step 6:

Save the model and counter vectorizer

- Save count vectorizer so that we can retain the vocabulary list and other setting to get the features. New text will have to be transformed through the count vectorizer
- Save the model for predict the new text
- The model name is: `lr-2022-<MM>-<DD>.pkl`
- The vectorizer name: `countvectoriser-2022-<MM>-<DD>.pkl`

Exercise 1 Answers:

Step 1: Read the SMSSpamCollection.txt and pre-process the data

```
import pandas as pd
import os
from google.colab import drive
drive.mount('/content/drive')

data_path = ['drive', 'MyDrive', 'ITB241', 'data']
```

```
Mounted at /content/drive
```

```
pd.set_option('display.max_colwidth', None) # Set display options so that the text aren't cut off

filepath = os.sep.join(data_path + ['SMSSpamCollection.txt'])
df=pd.read_csv(filepath,sep='\t')
df.columns=['label','text']
df.head()
```

```
import re
import string

pattern_alphanumeric="\w*\d\w*"
pattern_punctuation="["+re.escape(string.punctuation)+"]"

df['new_text']="empty"

for ind in df.index:
  temp=re.sub(pattern_alphanumeric, '',df['text'][ind])
  df['new_text'][ind]=re.sub(pattern_punctuation,'',temp).lower()
df_cleaned=df[['label', 'new_text']].copy()
df_cleaned
```

Exercise 1 Answers

Step 2: Split the dataset into training set and test set

```
# split the data into inputs and outputs
X = df_cleaned['new_text']
Y = df_cleaned['label']
```

```
# split the data into a training and test set
from sklearn.model_selection import train_test_split

# test size = 30% of observations, which means training size = 70% of observations
# random state = 42, so we all get the same random train / test split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

```
print("The size of original dataset:", X.shape)
print("The size of training dataset:",  X_train.shape)
print("The size of test dataset:",  X_test.shape)
```

Exercise 1 Answers

Step 3: Convert the text to vectors using count vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words='english', ngram_range=(1,1))

X_train_cv = cv.fit_transform(X_train)
X_test_cv  = cv.transform(X_test)

# print the dimensions of the training set (text messages, terms)
print("The dimensions of the training set:",X_train_cv.toarray().shape)
print("The dimensions of the test set:", X_test_cv.toarray().shape)

# print the features that has been create as a result of fit_transform()
print("The features:\n", cv.get_feature_names_out())
```

Exercise 1 Answers

Step 4:

Fit Logistic Regression model on training data and apply the fitted model to t est data. Predict the test data.

```
# Use a logistic regression model
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver='lbfgs')

# Train the model
lr.fit(X_train_cv, y_train)

# Take the model that was trained on the X_train_cv data and apply it to the X_test_cv data
y_pred_cv = lr.predict(X_test_cv)
# The output is all of the predictions
y_pred_cv
```

```
array(['ham', 'spam', 'ham', ..., 'ham', 'ham', 'spam'], dtype=object)
```

```
print(list(y_test[:10]))
print(list(y_pred_cv[:10]))
```

```
['ham', 'spam', 'ham', 'spam', 'ham', 'ham', 'ham', 'ham', 'ham', 'ham']
['ham', 'spam', 'ham', 'spam', 'ham', 'ham', 'ham', 'ham', 'ham', 'ham']
```

Exercise 1 Answers

Step 5: Evaluate the mode. Decide how good the model is by calculating various metrics

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_cv)
cm
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_cv, target_names=['ham','spam']))
```

Exercise 1 Answers

Step 6: Save the model and counter vectorizer

```
# There are two objects that we need to save.

# First, the Count Vectorizer so that we can retained the vocabulary list and
# other settings to get the features. New text will have to be transformed through the
# Count Vectorizer.

# Second, the LR model. This will be used for prediction.

import pickle
from datetime import datetime

model_path = ['drive', 'MyDrive', 'ITB241', 'models']

time = datetime.now().strftime("%Y-%m-%d")
filename = 'lr-{}.pkl'.format(time)
tempList=[]
tempList.append(filename)
path1 = os.sep.join(model_path + tempList)

filename = 'countvectoriser-{}.pkl'.format(time)
tempList=[]
tempList.append(filename)
path2 = os.sep.join(model_path + tempList)

with open(path1, 'wb') as f1:
    pickle.dump(lr, f1)

with open(path2, 'wb') as f2:
    pickle.dump(cv, f2)
```

Exercise 2: Use Logistic Regression model to predict ham or spam for SMS message

Step 1:

Load in the regression model that was saved during the modelling stage

Step 2:

Load in the vectorizer that was used to encode the training set

Exercise 2: Use Logistic Regression model to predict ham or spam for SMS message

Step 3:

Pre-process the new text: Clean the input data in the SAME way as it was done during modelling

- Create function preprocess(text) to clean the data

```
new_text="SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ \
TsandCs apply Reply HL 4 info"
new_text=preprocess(new_text)
new_text
```

```
'six chances to win cash from  to  pounds txt  and send to  cost day   tsandcs apply reply hl  info'
```

Exercise 2: Use Logistic Regression model to predict ham or spam for SMS message

Step 4:

Numerically encode the input : Convert the text to vectors using the previous Vectorizer

- Create function encode_text_to_vector(cv, text) to encode the new text

```
new_text_vector=encode_text_to_vector(trained_cv,new_text)
print(new_text_vector)
```

```
(0, 248)	1
(0, 859)	1
(0, 900)	1
(0, 1187)	1
(0, 1346)	1
(0, 2521)	1
(0, 2746)	1
(0, 4335)	1
(0, 4685)	1
(0, 4972)	1
(0, 5971)	1
(0, 5997)	1
(0, 6400)	1
```

Exercise 2: Use Logistic Regression model to predict ham or spam for SMS message

Step 5:

Predict the label

```python
predicted_label = (model.predict(new_text_vector))[0]
print ("The new text:\n",new_text)
print("Predicted label is:\n", predicted_label)
```

```
The new text:
 SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
Predicted label is:
 spam
```

Exercise 2 Answers

Step 1:

Load in the regression model that was saved during the modelling stage

```python
import os
import pickle
from google.colab import drive

drive.mount('/content/drive')

model_path = ['drive', 'MyDrive', 'ITB241', 'models']
filename=['lr-2022-10-10.pkl']

path1 = os.sep.join(model_path + filename)
with open(path1, 'rb') as f:
    model = pickle.load(f)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```python
model
```

```
LogisticRegression()
```

Exercise 2 Answers

Step 2:

Load in the vectorizer that was used to encode the training set

```
model_path = ['drive', 'MyDrive', 'ITB241', 'models']
filename=['countvectoriser-2022-10-10.pkl']

path2 = os.sep.join(model_path + filename)
with open(path2, 'rb') as f:
    trained_cv= pickle.load(f)
```

```
trained_cv
```

```
CountVectorizer(stop_words='english')
```

Exercise 2 Answers

Step 3: Pre-process the new text

```python
import re
import string

def preprocess(text):
  pattern_alphanumeric="\w*\d\w*"
  pattern_punctuation="["+re.escape(string.punctuation)+"]"

  text=re.sub(pattern_alphanumeric, '',text)
  text=re.sub(pattern_punctuation,'',text).lower()

  return text
```

```python
new_text="SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ \
TsandCs apply Reply HL 4 info"
new_text_processed=preprocess(new_text)
new_text_processed
```

```
'six chances to win cash from  to  pounds txt  and send to  cost day   tsandcs apply reply hl  info'
```

Exercise 2 Answer

Step 4:

Numerically encode the input : Convert the text to vectors using the previous Vectorizer

```
def encode_text_to_vector(cv, text):
        text_vector = cv.transform( [text ] )
        return text_vector
```

```
new_text_vector=encode_text_to_vector(trained_cv,new_text_processed)
print(new_text_vector)
```

Exercise 2 Answers

Step 5: Predict the label

```python
predicted_label = (model.predict(new_text_vector))[0]
print ("The new text:\n",new_text)
print("Predicted label is:\n", predicted_label)
```

```
The new text:
 SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
Predicted label is:
 spam
```

Test other SMS messages

Message	Predicted Label
I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.	ham
I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times.	ham
Oh k...i'm watching here:)	ham
Eh u remember how 2 spell his name... Yes i did. He v naughty	ham
Fine if thatś the way u feel. Thatś the way its gota b	ham
Is that seriously how you spell his name?	ham

Test other SMS messages

Message	Predicted Label
SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info	spam
URGENT! You have won a 1 week FREE membership in our £100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18	spam
XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL	spam
England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/ú1.20 POBOXox36504W45WQ 16+	spam

Naïve Bayes

Exercise 3: Training dataset

Text	Label
A great game	sports
The election was over	not sports
Very clean match	sports
It was a close election	not sports
A clean but forgettable game	Sports
A very close game	??

What is the label for a new text “A very close game”?
Would it be “Sports” or “Not Sports”?

Naïve Bayes Approach

Text	Label
a great game	sports
the election was over	not sports
very clean match	sports
it was a close election	not sports
a clean but forgettable game	sports

Find:

P(sports | “a very close game")

P(not sports | "a very close game”)

- If P(sports | “a very close game") is the larger value, then the label is sports
- If P(not sports | "a very close game”) is the larger value, then the label is not sports

count / total count of ALL WORDS

Laplace Smoothing

Text	Label
a great game	sports
The election was over	not sports
very clean match	sports
it was a close election	not sports
a clean but forgettable game	sports

P(sports | "A very close game") $\xrightarrow{\text{Bayes Theorem}}$ $\dfrac{P(\text{"a very close game"} \mid \text{sports}) \times P(\text{sports})}{P(\text{"a very close game"})}$

Naïve Bayes $\longrightarrow$ $\dfrac{P(\text{"a"} \mid \text{sports}) \times P(\text{"very"} \mid \text{sports}) \times \boxed{P(\text{"close"} \mid \text{sports})} \times P(\text{"game"} \mid \text{sports}) \times P(\text{sports})}{c}$

0

$\frac{0+1}{11+14}$

easier to calculate by breaking up

The word "close" isn't present in any *sports* text. This means P("close" | sports) is 0 and the end result will be 0 !

- The above issue can be eliminated with Laplace smoothing, where every count is added by 1
- The possible number of words is also added to the divisor, and the division will not be more than 1

In this case, the set of possible words are:
['a', 'great', 'very', 'over', 'it', 'but', 'game', 'match', 'clean', 'election', 'close', 'the', 'was', 'forgettable'].

always add 1

The possible number of words is 14. Therefore:
P("close" | sports) = (0+1)/ (11+14)=0.04 P("game" | sports) = (2+1)/ (11+14)= 0.12, and etc

total features for entire dataset (NON-REPEATED) --> 14

Naïve Bayes

P(sports | "A very close game") $\xrightarrow{\text{Naïve Bayes}}$

$$\frac{\boxed{P(\text{"a"} \mid sports) \times P(\text{"very"} \mid sports) \times P(\text{"close"} \mid sports) \times P(\text{"game"} \mid sports) \times P(sports)}}{P(\text{"a"}) \times P(\text{"very"}) \times P(\text{"close"}) \times P(\text{"game"})}$$

can ignore denominator

P(not sports | "A very close game") $\xrightarrow{\text{Naïve Bayes}}$

$$\frac{\boxed{P(\text{"A"} \mid not\ sports) \times P(\text{"very"} \mid not\ sports) \times P(\text{"close"} \mid not\ sports) \times P(\text{"game"} \mid not\ sports) \times P(not\ sports)}}{P(\text{"A"}) \times P(\text{"very"}) \times P(\text{"close"}) \times P(\text{"game"})}$$

can ignore denominator

As the dominator are the same, we can compare only the numerators to decide which probability is higher

P("A" | sports) = (2+1) / (11+14) = 0.12
P("very" | sports) = (1+1) / (11+14) = 0.08
P("close" | sports) = (0+1) / (11+14) = 0.04
P("game" | sports) = (2+1) / (11+14) = 0.12
P(sports) = 3/5

words for not sports

P("A" | not sports) = (1+1) / (9+14) = 0.087
P("very" | not sports) = (0+1) / (9+14) = 0.043
P("close" | not sports) = (1+1) / (9+14) = 0.087
P("game" | not sports) = (0+1) / (9+14) = 0.043
P(not sports) = 2/5

calculate probability P based on the table

Text		Label
a great game	3	sports
The election was over	4	not sports
very clean match	3	sports
it was a close election	3	not sports
a clean but forgettable game	2	sports

Exercise 3 answer

P (“a" | sports) x P (“very" | sports) x P (“close" | sports) x P ("game" | sports) x P(sports)

$= 0.12 \times 0.08 \times 0.04 \times 0.12 \times 0.6 = 2.76 \times 10^{-5}$

P (“a" | not sports) x P (“very" | not sports) x P (“close" | not sports) x P ("game" | not sports) x P(not sports)

$= 0.087 \times 0.043 \times 0.087 \times 0.043 \times 0.4 = 0.56 \times 10^{-5}$

Hence, P (sports | “a very close game”) gives a higher probability, suggesting that the sentence belongs to the sports category.

Exercise 4:

Can you implement the **Build Phase and Operational Phase** using Naïve Bayes to predict ham or spam for SMS messages

Hint: Modify the code from "Logistic Regression". A very small changes only!

Exercise 4 Answers:

Step 4: Fit Naive Bayes model on training data and apply the fitted model to test data

```
[ ]
# Use a Naive Bayes model
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()

# Train the model
nb.fit(X_train_cv, y_train)

# Take the model that was trained on the X_train_cv data and apply it to the X_test_cv data
y_pred_cv = nb.predict(X_test_cv)
# The output is all of the predictions
y_pred_cv
```

- Only shows the difference from the code Exercise 1 and Exercise 2
- Change all the variable **lr** to **nb** in the subsequent code