Computational Communication Science 2

Week 7

Lab session

Roeland Dubèl
r.dubel@uva.nl

2025-04-08

Digital Society Minor
University of Amsterdam

Today

1. Good coding practices
2. Weekly MC-questions
3. Q&A
4. Classification metrics
5. Break
6. Weekly exercises

Good coding practices

Good coding practices

- Try to read in libraries once at the top of your script or logically clustered

```
[1]: import numpy as np
     import pandas as pd

     import matplotlib.pyplot as plt
     import seaborn as sns

     from sklearn.model_selection import train_test_split
```

```
[2]: # code xxx
```

Good coding practices

- Use markdown to structure your code

Read in libaries

[1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
```

Read in data

[2]:
```
df = pd.read_csv("data.csv")
```

[ ]:

Good coding practices

- Cluster code in blocks

```
[4]: number = 1

[5]: number = number * 2

[6]: number = number ** 4

[7]: number = number - 10

[8]: number = round(number)

[9]: print(number)

     6

•[10]: number = 1
       number = number * 2
       number = number ** 4
       number = number - 10
       number = round(number)
       print(number)

       6
```

Good coding practices

- Try to explain code line-by-line

[11]:
```
number = 1 # We define number
number = number * 2 # We multiply number by two
number = number ** 4 # We take the power of 4 of number
number = number - 10 # We subtract 10 from number
number = round(number) # We round number to an integer
print(number) # We print number
```

Good coding practices

- Make use of functions → less repetitive

[14]:
```
def get_new_number(number):
    number = number * 2 # We multiply number by two
    number = number ** 4 # We take the power of 4 of number
    number = number - 10 # We subtract 10 from number
    number = round(number) # We round number to an integer
    print(number) # We print number
```

[15]:
```
get_new_number(1)
```

```
6
```

Good coding practices

- Be aware of what must be inside or outside the function → more speed

```
[28]:
def do_something_with_text(text):
    nlp = spacy.load("en_core_web_sm") # We load in a language model
    doc = nlp(text) # We apply natural language processing to the text
    # Some code ...


nlp = spacy.load("en_core_web_sm")

def do_something_with_text(text, nlp=nlp):
    doc = nlp(text) # We apply natural language processing to the text
    # Some code ...
```

Good coding practices

- Built escapes in your code → more robustness

```python
def get_new_number(number):
    type_of_number = type(number) #We store the type of number
    if type_of_number == int: # We check if number is an integer
        number = number * 2 # We multiply number by two
        print(number) # We print number
    else:
        print(f"Number is not an integer but an {type_of_number}.")

get_new_number("1")
```

```
Number is not an integer but an <class 'str'>.
```

```python
def get_new_number(number):
    try:
        number = number * 2 # We multiply number by two
        print(number) # We print number
    except Exception as e: # We catch the error
        print(e) # We print the error

get_new_number(None)
```

```
unsupported operand type(s) for *: 'NoneType' and 'int'
```

```python
number = 1 # We define number
while True: # We loop over number conditionally
    number *= 2 # We multiply and store over number
    if number > 100: # We stop the loop if 100 is passed
        break
print(number)
```

```
128
```

Good coding practices

- Type 'object' is rather uninformative

```
df = pd.DataFrame({"integers": [1, 2, 3],
                   "strings": ["a", "b", "c"],
                   "empty_dictionaries": [{}, {}, {}],
                   "empty_tuples": [(), (), ()],
                   "mixed_objects": [1, "a", {}],
                   "nonetype_objects": [None, None, None],
                   "lists": [[], [], []],
                   "dataframes": [pd.DataFrame(), pd.DataFrame(), pd.DataFrame()]
                  })
```

```
df
```

	integers	strings	empty_dictionaries	empty_tuples	mixed_objects	nonetype_objects	lists	dataframes
0	1	a	{}	()	1	None	[]	Empty DataFrame Columns: [] Index: []
1	2	b	{}	()	a	None	[]	Empty DataFrame Columns: [] Index: []
2	3	c	{}	()	{}	None	[]	Empty DataFrame Columns: [] Index: []

```
df.dtypes
```

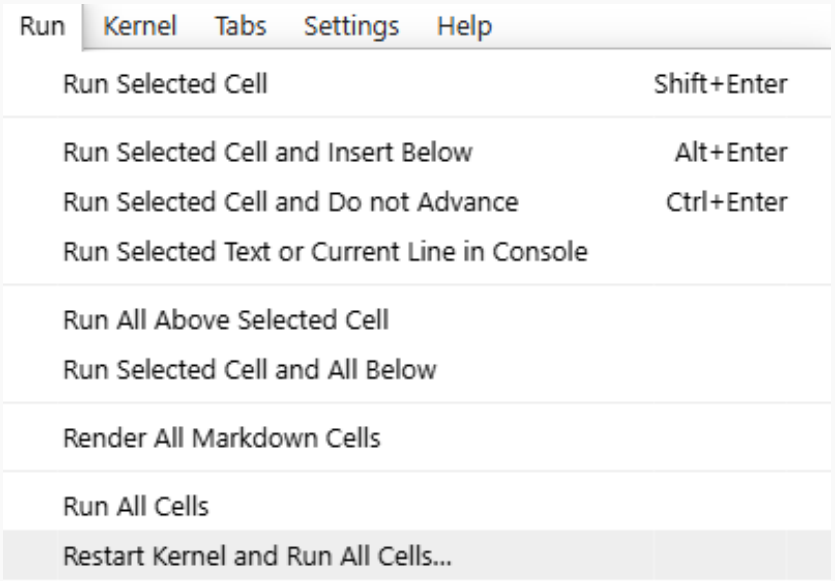
```
integers              int64
strings              object
empty_dictionaries   object
empty_tuples         object
mixed_objects        object
nonetype_objects     object
lists                object
dataframes           object
dtype: object
```

```
for column in df.columns: #We loop over all columns
    object_type = type(df[column].values[0]) #We take the type of the first item of each column
    print(object_type)
```

```
<class 'numpy.int64'>
<class 'str'>
<class 'dict'>
<class 'tuple'>
<class 'int'>
<class 'NoneType'>
<class 'list'>
<class 'pandas.core.frame.DataFrame'>
```

Good coding practices

- Make sure that your code runs ...



- Note. We will not penalize for it in the in-class exam

Weekly MC-questions

MC-questions Week 7

- Canvas $\rightarrow$ Modules $\rightarrow$ Week 7 $\rightarrow$ MC-questions
- 4 questions, 8 minutes (in silence)
- Afterwards we will discuss the questions

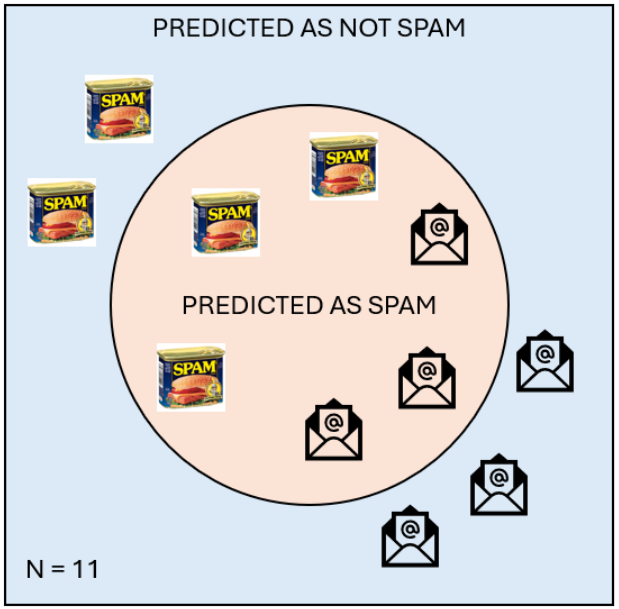
Q&A

Q&A

- Remaining questions about this or last week?

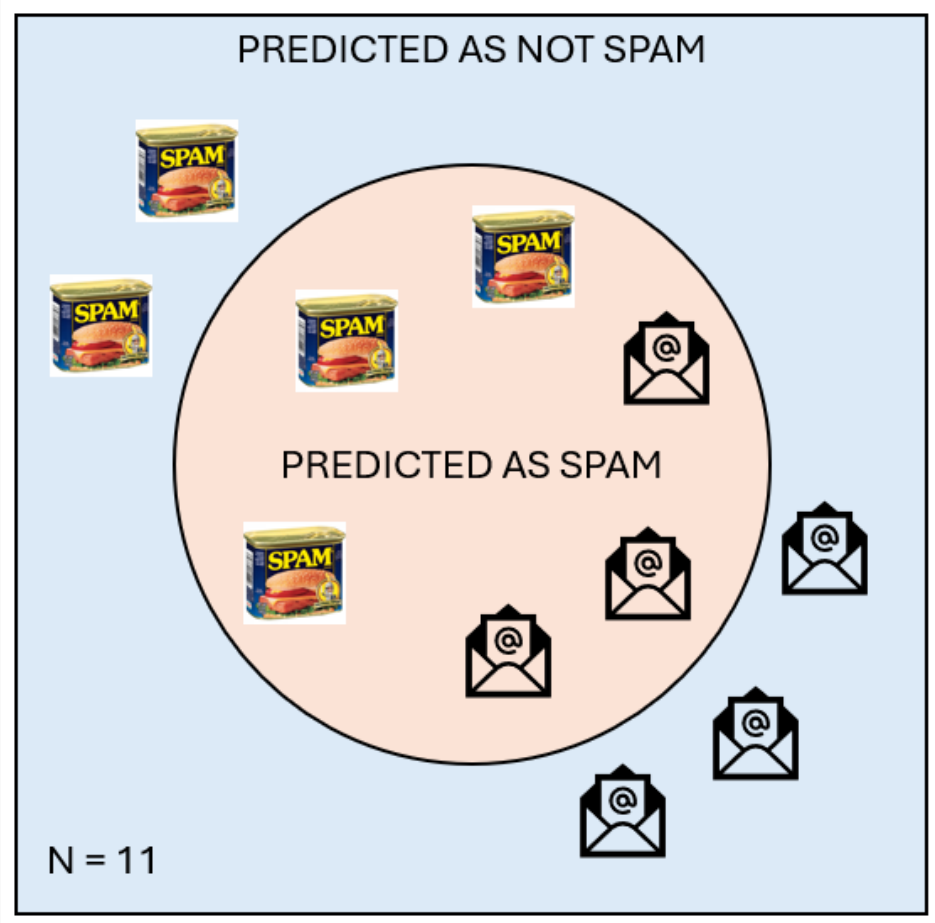
Classification metrics

Classification metrics



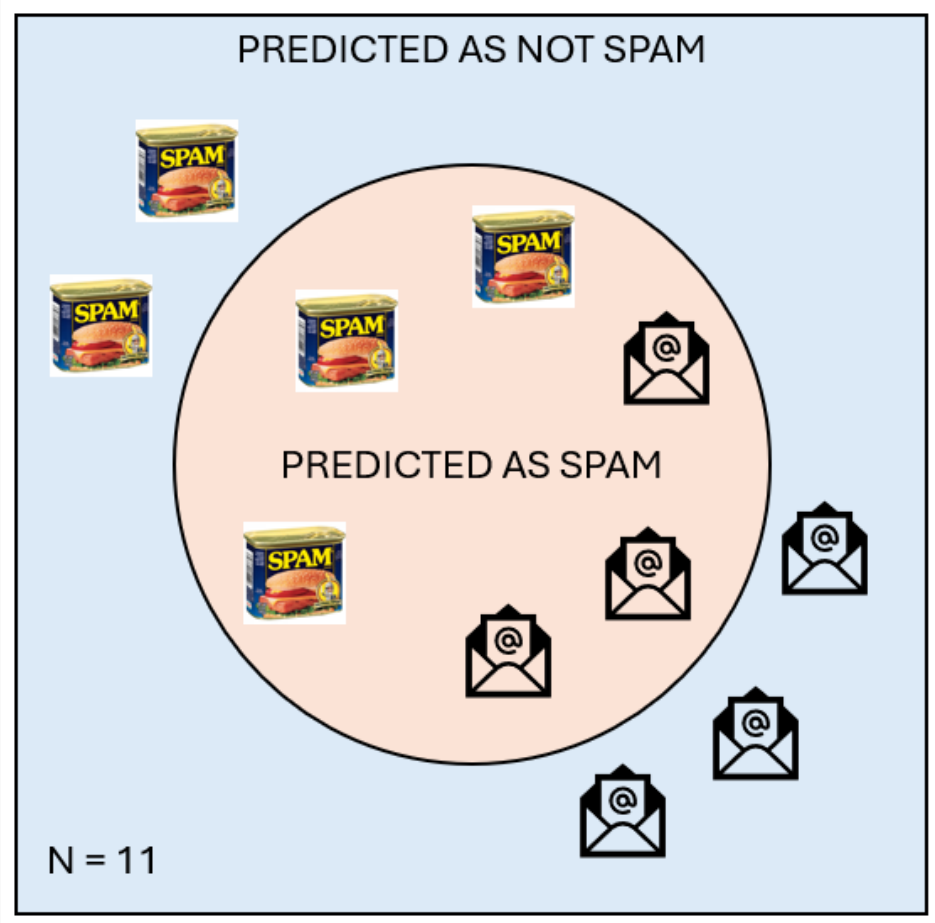
- True positives (TP) → spam correctly predicted as spam
- False positives (FP) → not spam incorrectly predicted as spam
- True negatives (TN) → not spam correctly predicted as not spam
- False negatives (FN) → spam incorrectly predicted as not spam

Classification metrics



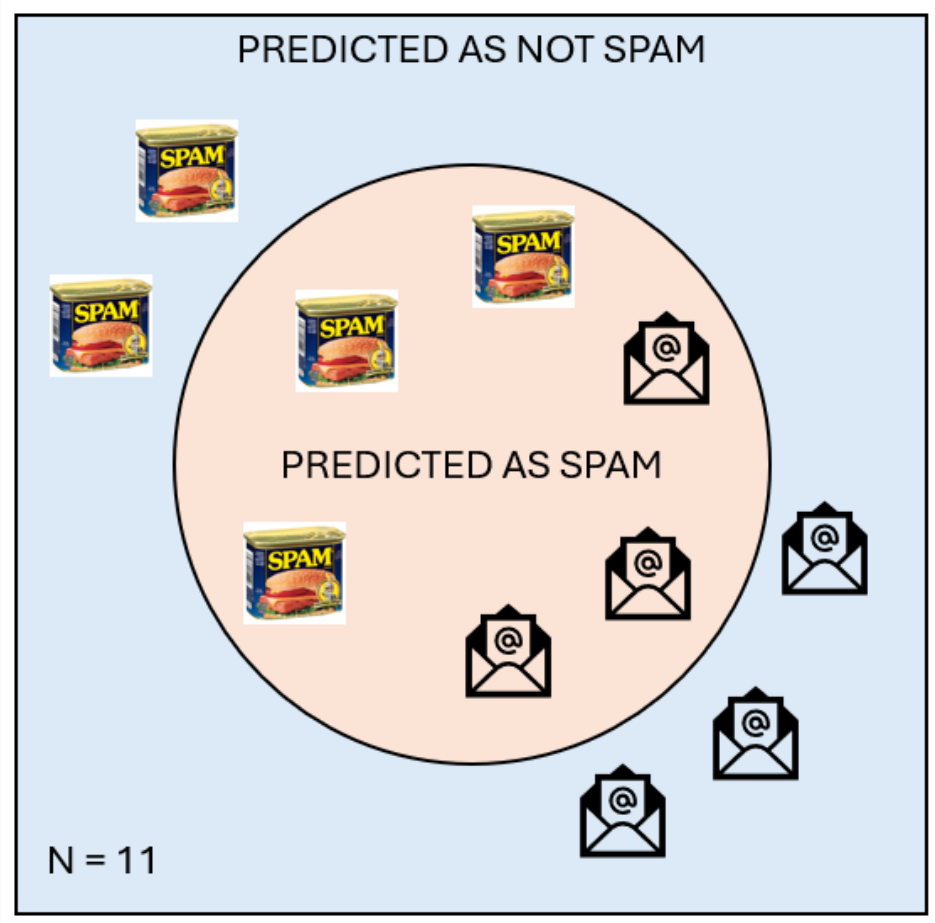
$$\text{Accuracy} = \frac{3\ \text{TP} + 3\ \text{TN}}{3\ \text{TP} + 3\ \text{FP} + 2\ \text{FN} + 3\ \text{TN}} \approx 0.55$$

Classification metrics



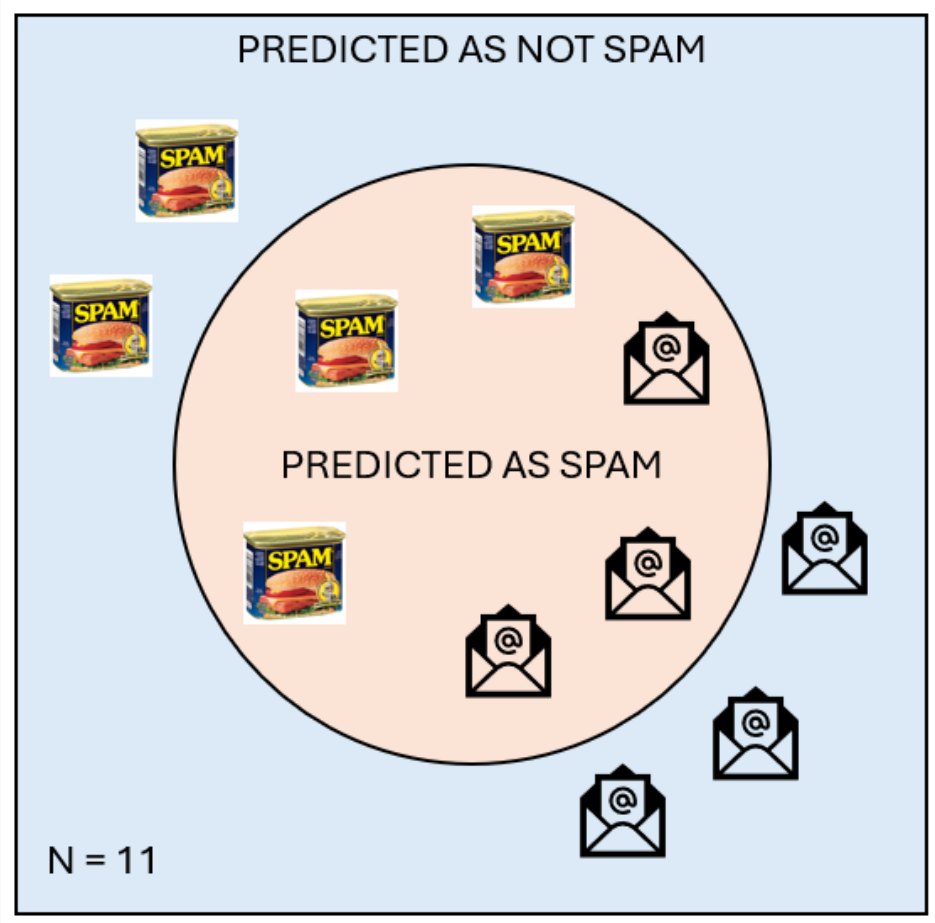
$$\text{Precision} = \frac{3\ \text{TP}}{3\ \text{TP} + 3\ \text{FP}} = 0.50$$

Classification metrics



$$\text{Recall} = \frac{3\ \text{TP}}{3\ \text{TP} + 2\ \text{FN}} = 0.60$$

Classification metrics



$$\text{F1-score} = \frac{2 \times 0.50 \text{ Precision} \times 0.60 \text{ Recall}}{0.50 \text{ Precision} + 0.60 \text{ Recall}} \approx 0.55$$

Break

15 minute break

Weekly exercises

Weekly exercises: Week 7

- Form groups of 3
- Go through the weekly exercises (GitHub → week07 → exercise-tutorial → week7-exercises.ipynb)
- `hatespeech_text_label_vote_RESTRICTED_100K.csv`

Weekly exercises: Week 7

1. Opening the data (Q1a, in class)
2. Training and testing a classifier (Q2 – Q5, 30 minutes)
3. Extra questions in the notebook with answers (Q6 – Q14)

Next week

Next week

- Any questions left?
  - Pose questions for next week via Google Docs
  - Sign up for the consultation hours
- See you next week!