UNIVERSITY OF AMSTERDAM

Python for Teachers:

# Machine Learning

Day 5 – Afternoon Session

Dr. A. Marthe Möller

UNIVERSITY OF AMSTERDAM

## What are we talking about?!

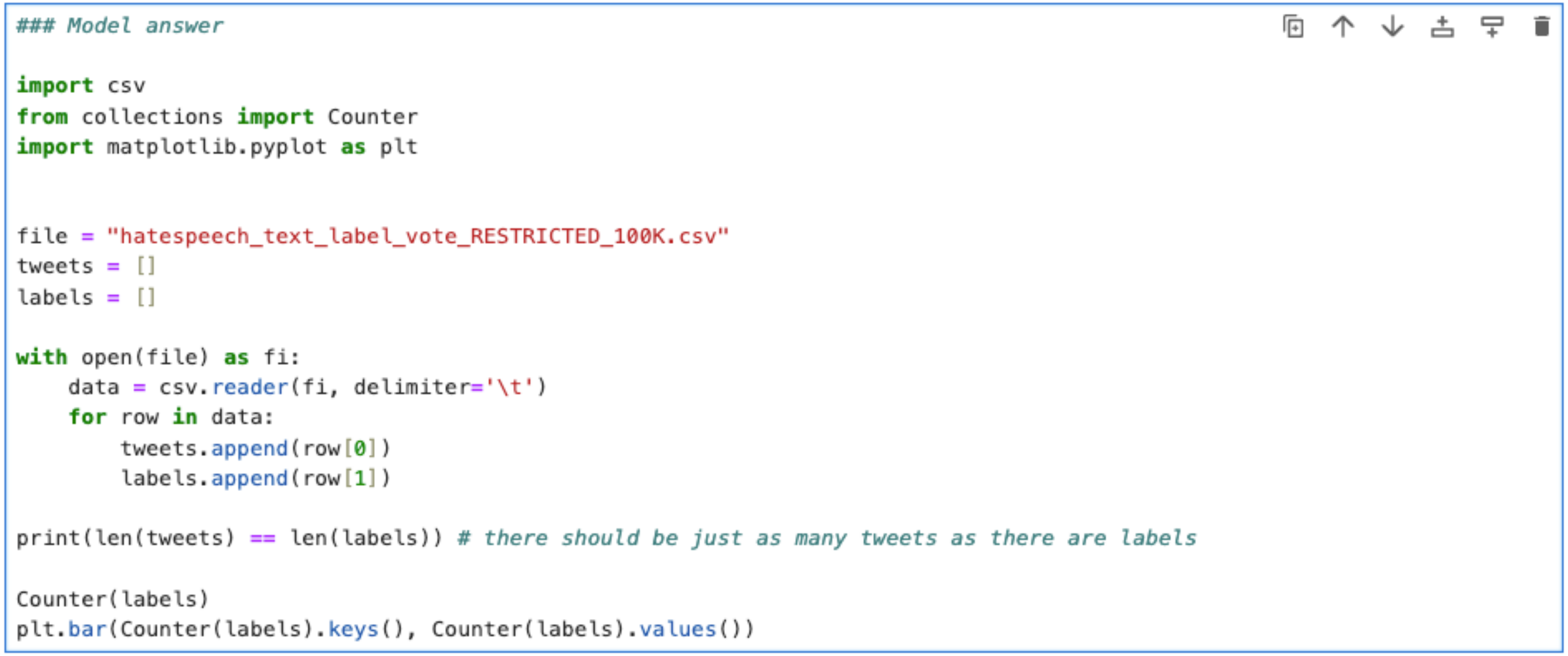
- Exercise 1: Recap
- Model validation
- Validation metrics
- About input for SML

## Exercise 1: Recap

- What did you think? Hard? Easy?

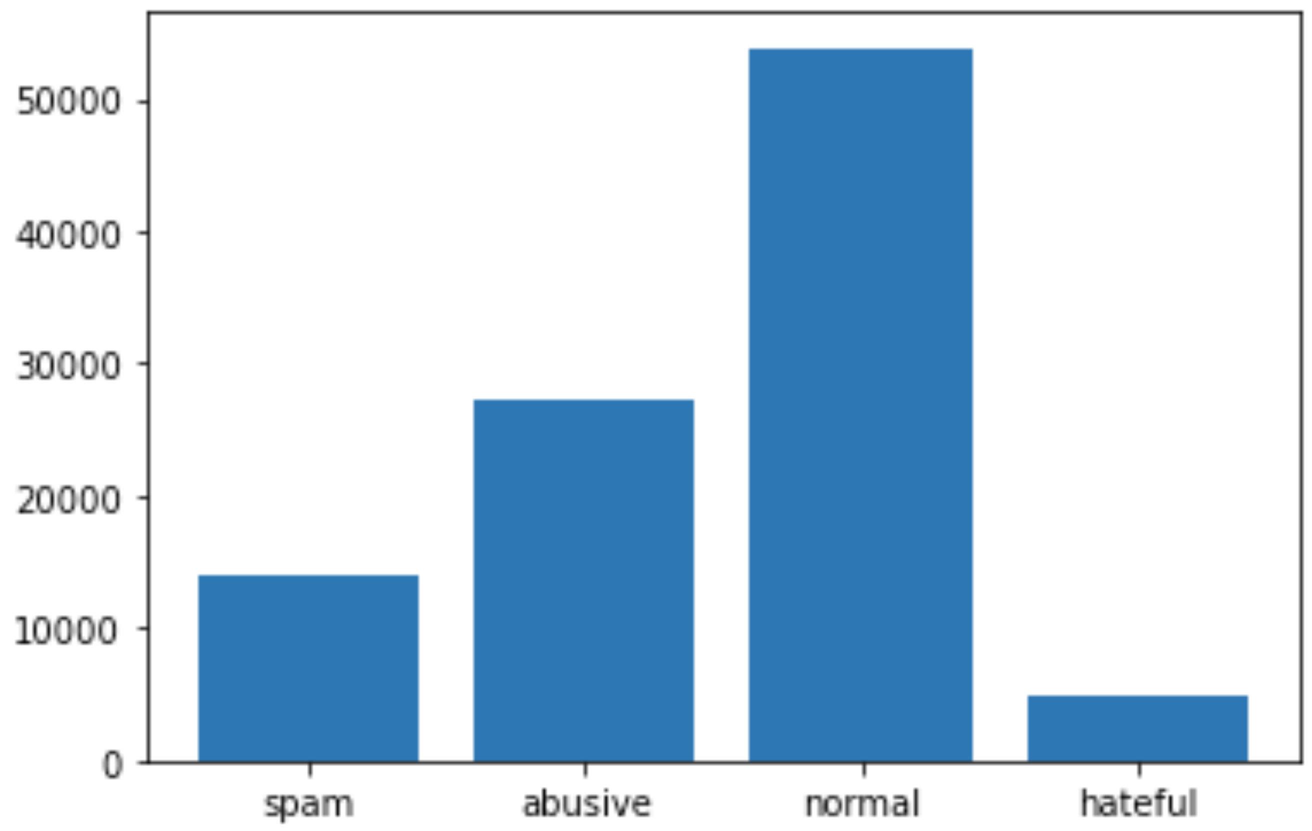
UNIVERSITY OF AMSTERDAM

## Exercise 1: Question 1



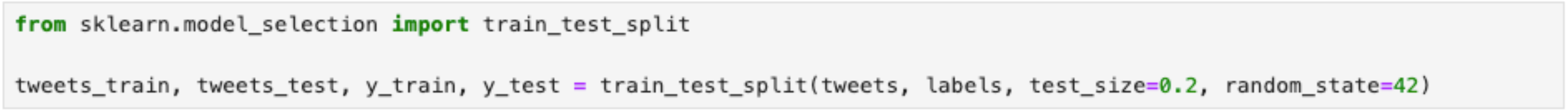
```
### Model answer

import csv
from collections import Counter
import matplotlib.pyplot as plt


file = "hatespeech_text_label_vote_RESTRICTED_100K.csv"
tweets = []
labels = []

with open(file) as fi:
    data = csv.reader(fi, delimiter='\t')
    for row in data:
        tweets.append(row[0])
        labels.append(row[1])

print(len(tweets) == len(labels)) # there should be just as many tweets as there are labels

Counter(labels)
plt.bar(Counter(labels).keys(), Counter(labels).values())
```

## Exercise 1: Question 1



## Exercise 1: Question 2

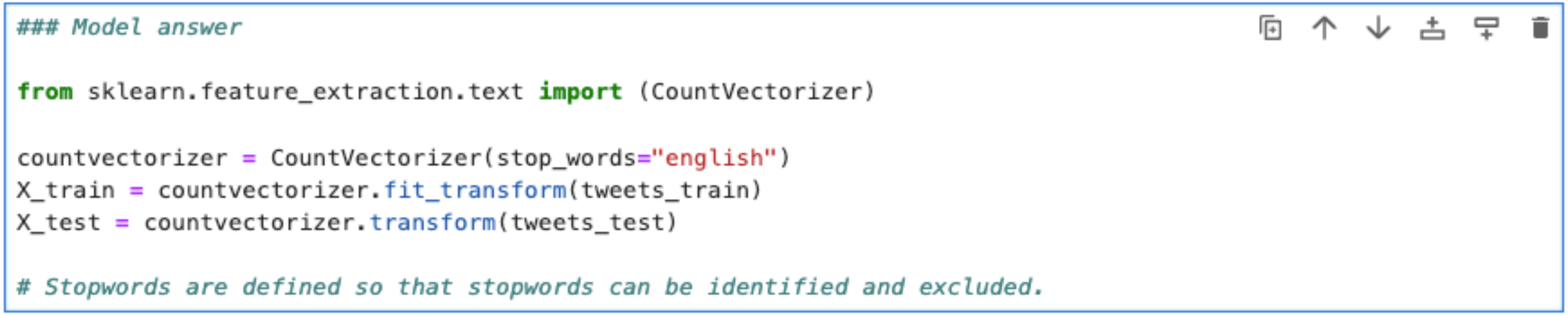
- What do these lines of code do?
- Do you know what the random_state part refers to? Why is this useful?



```python
from sklearn.model_selection import train_test_split

tweets_train, tweets_test, y_train, y_test = train_test_split(tweets, labels, test_size=0.2, random_state=42)
```

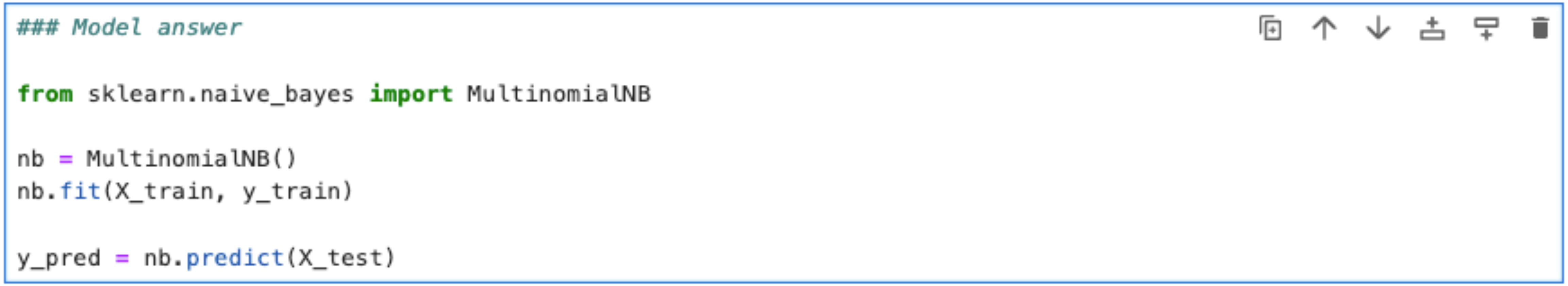
UNIVERSITY OF AMSTERDAM

## Exercise 1: Question 3



```
### Model answer

from sklearn.feature_extraction.text import (CountVectorizer)

countvectorizer = CountVectorizer(stop_words="english")
X_train = countvectorizer.fit_transform(tweets_train)
X_test = countvectorizer.transform(tweets_test)

# Stopwords are defined so that stopwords can be identified and excluded.
```

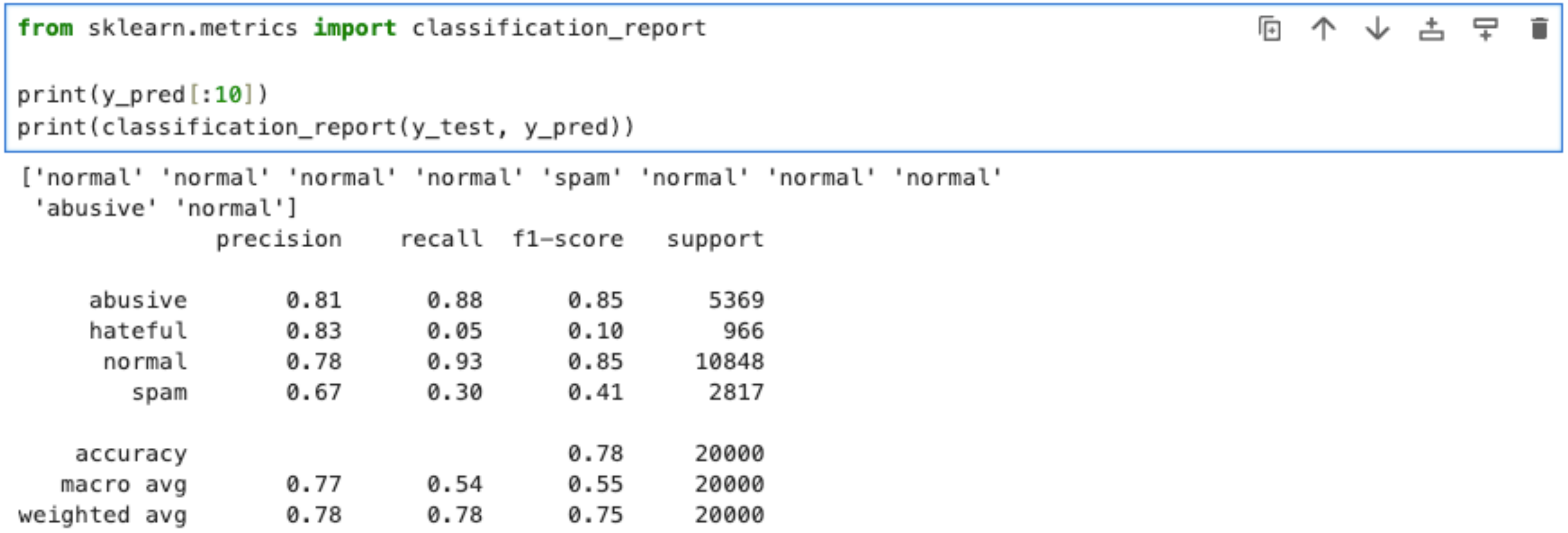
## Exercise 1: Question 4



```
### Model answer

from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB()
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
```

UNIVERSITY OF AMSTERDAM

## Exercise 1: Question 5



```python
from sklearn.metrics import classification_report

print(y_pred[:10])
print(classification_report(y_test, y_pred))
```

```
['normal' 'normal' 'normal' 'normal' 'spam' 'normal' 'normal' 'normal'
 'abusive' 'normal']
              precision    recall  f1-score   support

     abusive       0.81      0.88      0.85      5369
     hateful       0.83      0.05      0.10       966
      normal       0.78      0.93      0.85     10848
        spam       0.67      0.30      0.41      2817

    accuracy                           0.78     20000
   macro avg       0.77      0.54      0.55     20000
weighted avg       0.78      0.78      0.75     20000
```

## Model validation

Validation: When we assess the performance of a classifier

Or when we try to answer the question: How well does this classifier work?

We can use this information to learn what model works best for us!

## Model validation

What criteria should we use to decide on this?

Entirely context specific!

## Compare different goals for using SML

- To automatically decide what Instagram users should see a specific advertisement
- To automatically remove spam from a Twitter feed

Would you use the same criterion in both cases to determine what classifier to use? Why (not)?

## Compare different goals for using SML

There are various evaluation metrics available for machine learning.

In Scikit-learn, they are presented by ways of a classification report.

## Validation metrics: Precision

Precision quantifies the number of positive class prediction that actually belong to the positive cases.

How much of what we found is actually correct?

## Compare different goals for using SML

- To automatically decide what Instagram users should see a specific advertisement
- To automatically remove spam from a Twitter feed

Is precision a useful metric in these cases?

(Remember: How much of what we found is actually correct?)

## Validation metrics: Recall

Recall quantifies the number of positive class predictions made out of all positive examples in the dataset.

How many of the cases that we wanted to find did we actually find?

## Compare different goals for using SML

- To automatically decide what Instagram users should see a specific advertisement
- To automatically remove spam from a Twitter feed

Is recall a useful metric in these cases?

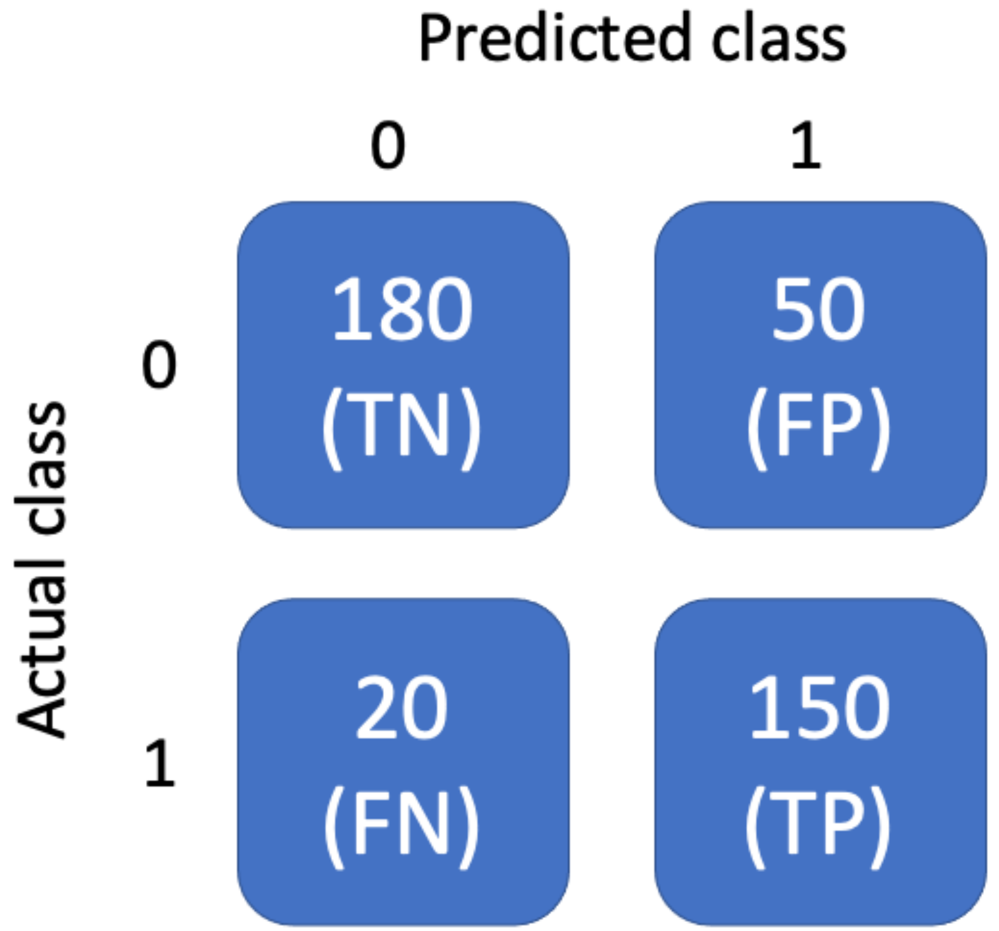
(Remember: How many of the cases that we wanted to find did we actually find?)

UNIVERSITY OF AMSTERDAM

## Precision and Recall

|  |  | Predicted class |  |
|---|---|---|---|
|  |  | 0 | 1 |
| Actual class | 0 | True Neg | False Pos |
|  | 1 | False Neg | True Pos |

## Precision and Recall



|  |  | Predicted class |  |
|---|---|---|---|
|  |  | 0 | 1 |
| Actual class | 0 | 180 (TN) | 50 (FP) |
|  | 1 | 20 (FN) | 150 (TP) |

## Precision and Recall

Precision is calculated as: $\frac{TP}{TP+FP}$

In this example: $\frac{150}{150+50}$ which is 0.75

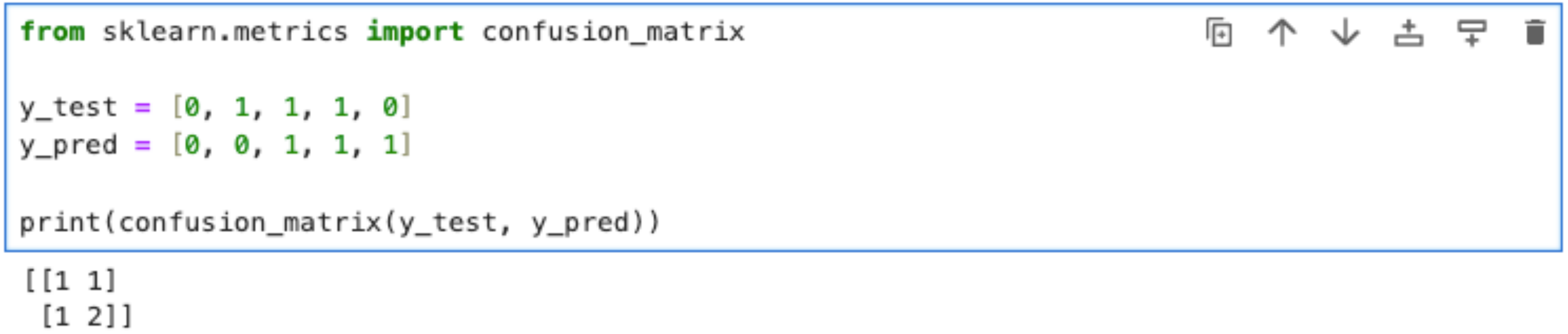
Recall is calculated as: $\frac{TP}{TP+FN}$

In this example: $\frac{150}{150+20}$ which is 0.88

|  |  | Predicted class |  |
|---|---|---|---|
|  |  | 0 | 1 |
| Actual class | 0 | 180 (TN) | 50 (FP) |
|  | 1 | 20 (FN) | 150 (TP) |

## What does this look like in code?

Let’s ask for a confusion matrix:

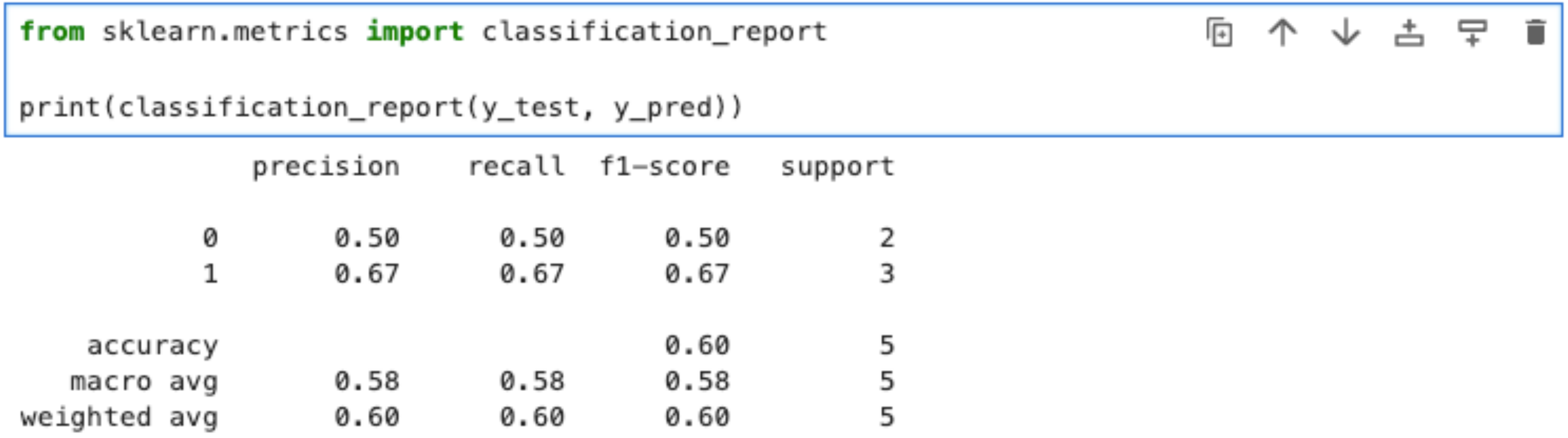


```python
from sklearn.metrics import confusion_matrix

y_test = [0, 1, 1, 1, 0]
y_pred = [0, 0, 1, 1, 1]

print(confusion_matrix(y_test, y_pred))
```

```
[[1 1]
 [1 2]]
```

## What does this look like in code?

Let’s get some metrics for validation:



```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       0.67      0.67      0.67         3

    accuracy                           0.60         5
   macro avg       0.58      0.58      0.58         5
weighted avg       0.60      0.60      0.60         5
```

## But wait…

Compare different goals for using SML:

- To automatically decide what Instagram users should see a specific advertisement
- To automatically remove spam from a Twitter feed

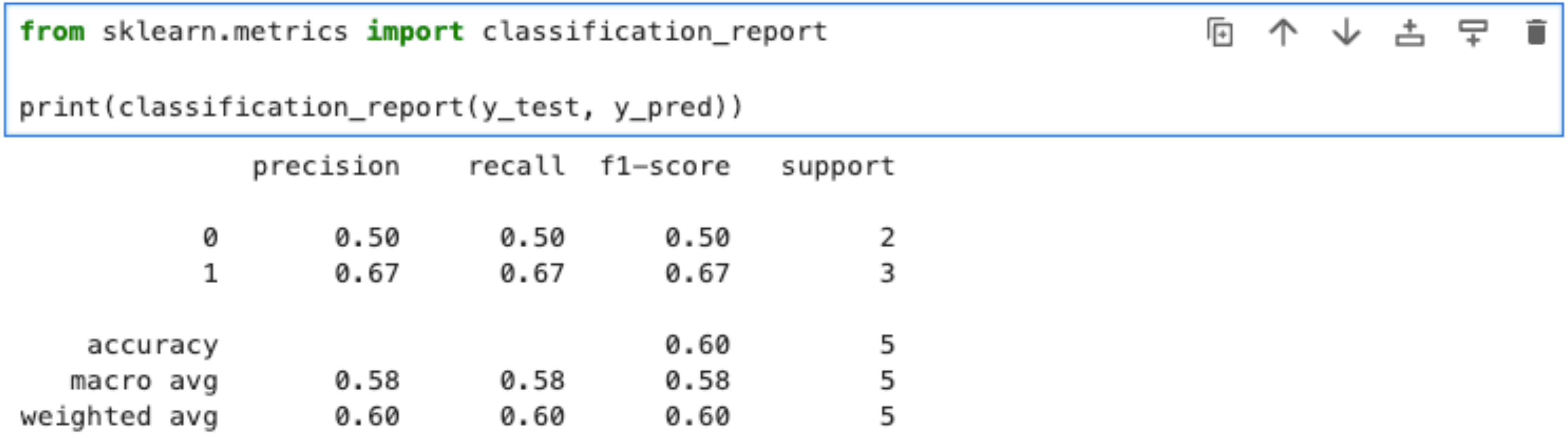
Such information was not at all available in the exercise!

## $F_1$-score

$F_1$-score: The harmonic mean of precision and recall (weighted average of precision and recall)

$$F_1\text{-score} = 2 \times \frac{precision \times recall}{precision + recall}$$

## Accuracy



```python
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       0.67      0.67      0.67         3

    accuracy                           0.60         5
   macro avg       0.58      0.58      0.58         5
weighted avg       0.60      0.60      0.60         5
```

## Accuracy

Accuracy: In which percentage of all cases was our classifier right?

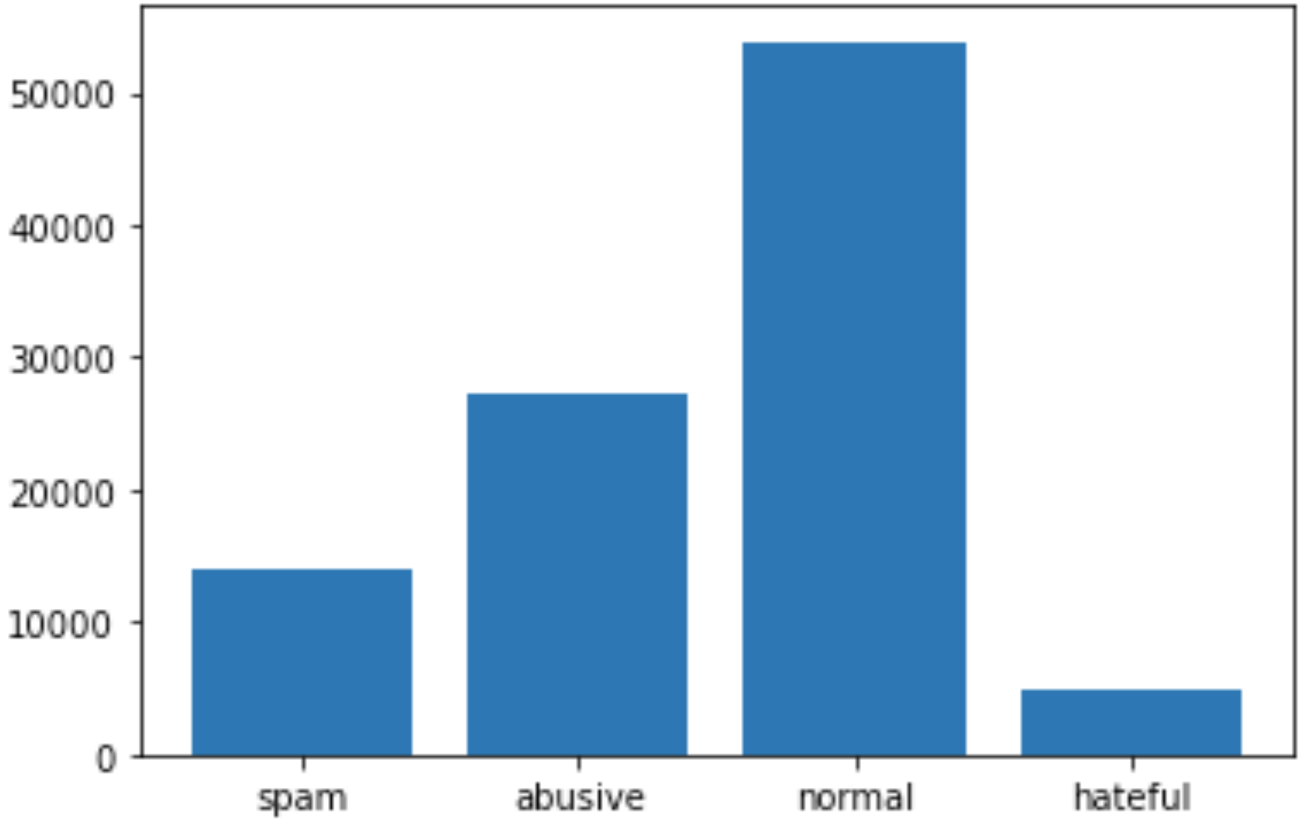
Always consider class distribution

## Accuracy

Class distribution: The number of examples that belong to each class

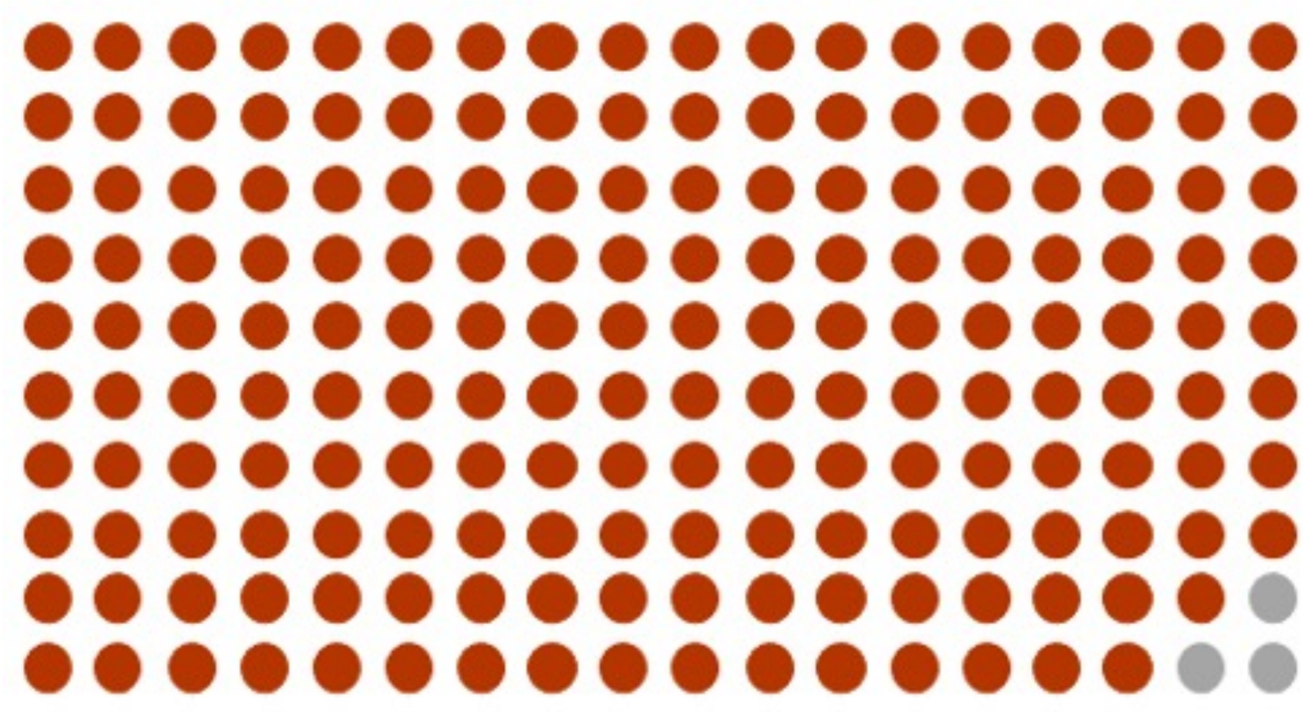
Imbalanced classification: A predictive modeling problem where the distribution of examples across the classes within a training dataset is not equal.

## Accuracy


50000
40000
30000
20000
10000
0
spam
abusive
normal
hateful

## Accuracy

Always check how your cases are distributed across the labels.



## I recommend

Always select what validation metric(s) you will use to pick your best performing classifier *beforehand* (i.e., you can pre-register it).

Think carefully about which metric is the best for your RQ.

## What are we talking about?!

- ~~Exercise 1: Recap~~
- ~~Model validation~~
- ~~Validation metrics~~
- About input for SML

## Validating models

Amongst others, classifiers can differ based on:

- Preprocessing steps (e.g., stopwords removed or not)
- The vectorizer that is used on the data (i.e., count or tf-idf)
- The underlying model (e.g., Logistic Regression, Decision Trees)

## Validating models

You can compare these models/check their performance based on (amongst others):

- Different validation metrics (e.g., accuracy, recall)
- Comparing its results to the results of the manual classifications

## About input

Pour data quality can cause:

- Your classifier not being able to identify specific categories
- Your classifier to perform badly overall
- Your classifier to pick up implicit bias

## What makes input bad input?

- Class imbalance
- Unreliable manual coding
- Unclear codebook
- Bias in your coders
- Input data from a different population than the data you want to study

## What can you do to create better input?

- Avoid class imbalance: increase your sample (add rare cases), or random selection
- Improve reliability of manual coding: check Krippendorf’s Alpha, add coder training, spend more time/resources developing better codebooks
- Learn more about the causes and cures of coder bias
- Don’t use classifiers on data from populations other than those that produced the training data

## The limits of your output: User comments

- Different populations, different characteristics
- Machine is trained on these characteristics
- For example: comments written in response to music videos vs. comments written in response to videos of political debates

## To SML or not to SML?

SML suitability depends on:

- How hard/easy it is to translate decision process into rules
- How much (annotated) data is available or can be made available
- How many cases need to be classified
- What room you have for errors
- What the annotated data looks like (quality, class imbalance)

## Up next

| **Time:** | **Topic:** |
|---|---|
| ~~09:30 – 11:00~~ | ~~Basics of (supervised) machine learning~~ |
| ~~11:00 – 12:00~~ | ~~Exercise 1~~ |
| ~~12:00 – 13:00~~ | ~~Break~~ |
| ~~13:00 – 13:30~~ | ~~Recap on Exercise 1~~ |
| ~~13:30 – 14:30~~ | ~~Evaluating models~~ |
| 14:30 – 15:30 | Exercise 2 |
| 15:30 – 16:00 | Recap on Exercise 2 |
| 16:00 – 16:30 | Closure |

Image: https://pixabay.com/nl/vectors/koffie-kop-koffie-beker-cafe%c3%afne-5009730/

## Enough talking – let’s get cracking!

Find Exercise 2 on Github (Jupyter Notebook)

The exercise takes you through the steps to train and validate two machines

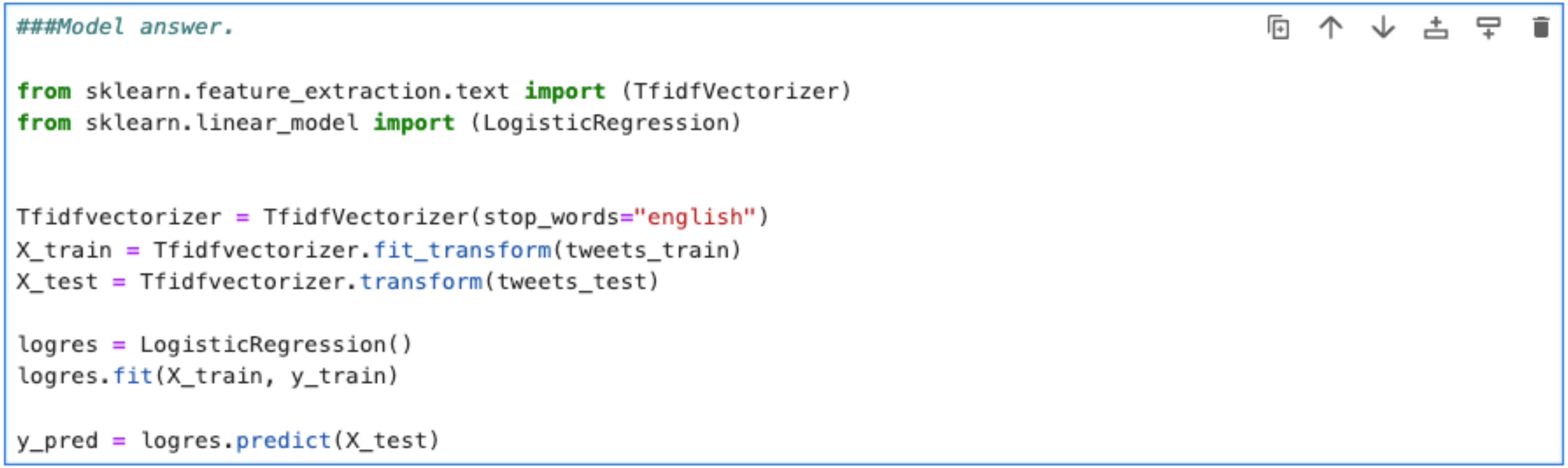
We will decide on a best machine together!

## The cracking stops at: … hrs

## Exercise 2: Recap

- What did you think? Hard? Easy?

## Exercise 2: Question 1



```
###Model answer.

from sklearn.feature_extraction.text import (TfidfVectorizer)
from sklearn.linear_model import (LogisticRegression)


Tfidfvectorizer = TfidfVectorizer(stop_words="english")
X_train = Tfidfvectorizer.fit_transform(tweets_train)
X_test = Tfidfvectorizer.transform(tweets_test)

logres = LogisticRegression()
logres.fit(X_train, y_train)

y_pred = logres.predict(X_test)
```

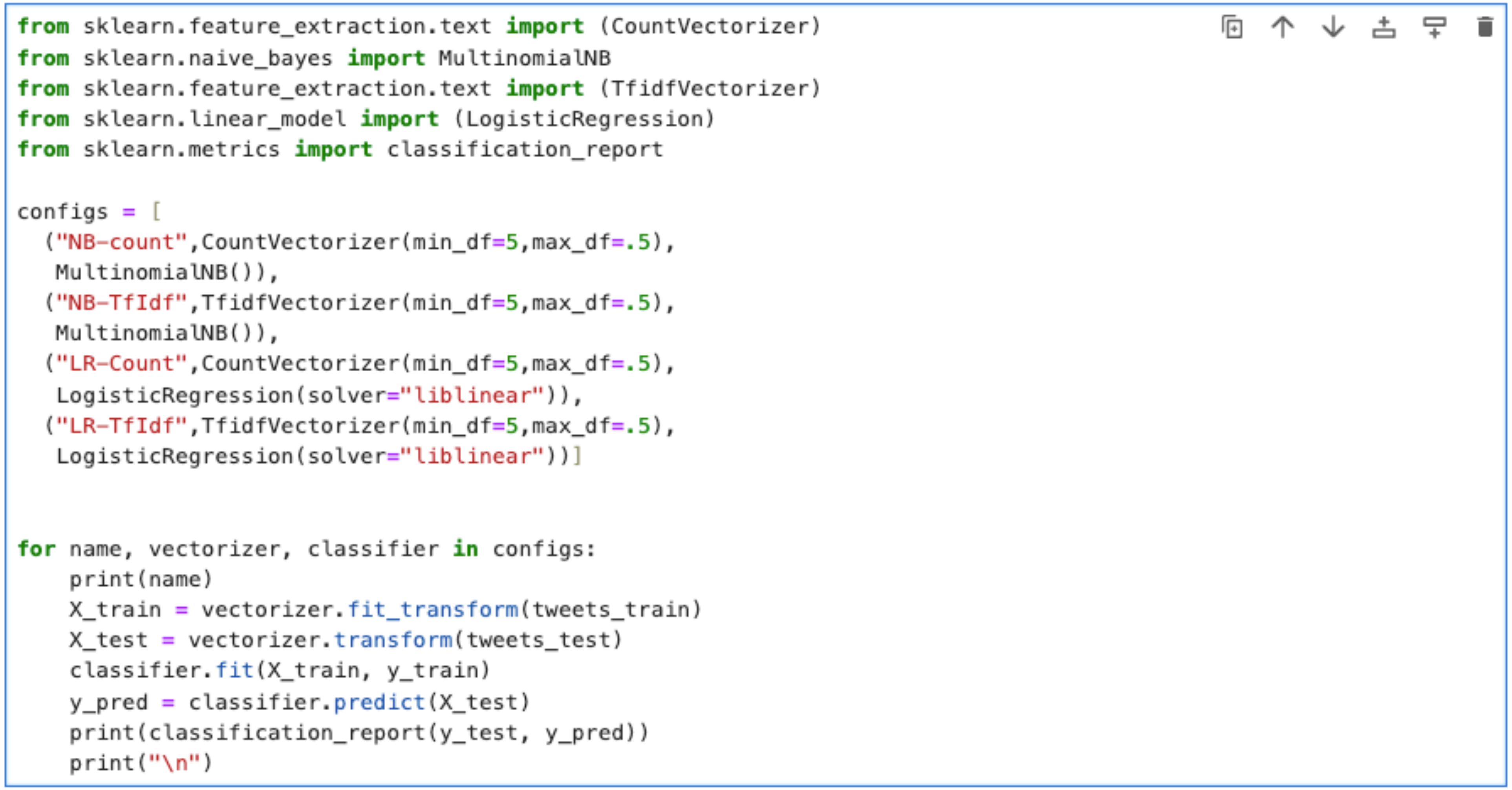
## Exercise 2: Question 2

As discussed earlier, we can try different combinations of these models (Naïve Bayes and Logistic Regression) and vectorizers (count and tf-idf). If you want to use Naïve Bayes and Logistic Regression as the models for a classifier, and a count vectorizer and a tf-idf vectorizer, how many classifiers could you then train?

You could then train 2 (count vectorzier vs. tf-idf vectorizer) * 2 (Naïve Bayes vs. Logistic Regression) = 4 classifiers.

UNIVERSITY OF AMSTERDAM

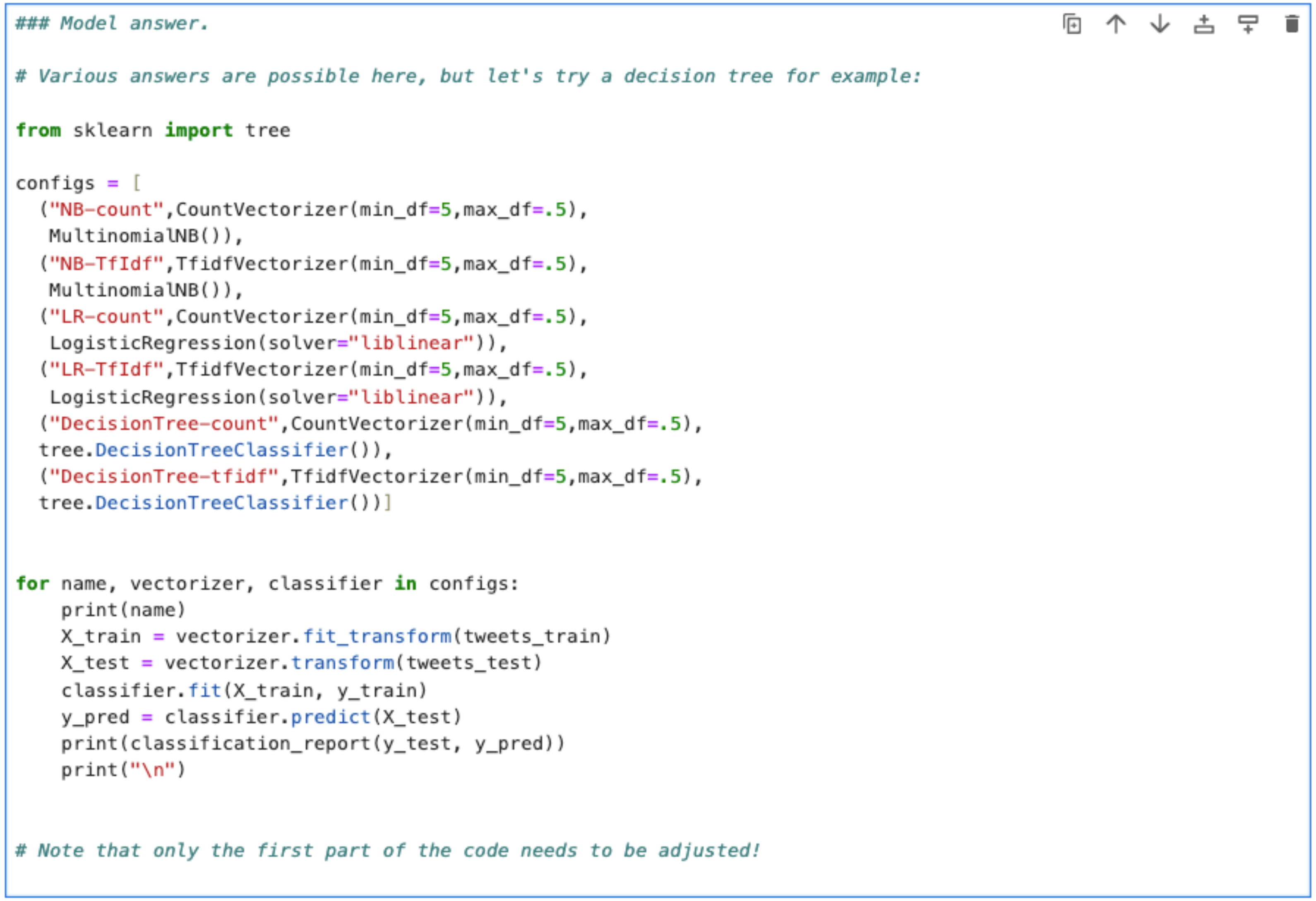
## Exercise 2: Question 3



```python
from sklearn.feature_extraction.text import (CountVectorizer)
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import (TfidfVectorizer)
from sklearn.linear_model import (LogisticRegression)
from sklearn.metrics import classification_report

configs = [
  ("NB-count",CountVectorizer(min_df=5,max_df=.5),
   MultinomialNB()),
  ("NB-TfIdf",TfidfVectorizer(min_df=5,max_df=.5),
   MultinomialNB()),
  ("LR-Count",CountVectorizer(min_df=5,max_df=.5),
   LogisticRegression(solver="liblinear")),
  ("LR-TfIdf",TfidfVectorizer(min_df=5,max_df=.5),
   LogisticRegression(solver="liblinear"))]


for name, vectorizer, classifier in configs:
    print(name)
    X_train = vectorizer.fit_transform(tweets_train)
    X_test = vectorizer.transform(tweets_test)
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)
    print(classification_report(y_test, y_pred))
    print("\n")
```

UNIVERSITY OF AMSTERDAM

## Exercise 2: Question 4



```
### Model answer.

# Various answers are possible here, but let's try a decision tree for example:

from sklearn import tree

configs = [
  ("NB-count",CountVectorizer(min_df=5,max_df=.5),
   MultinomialNB()),
  ("NB-TfIdf",TfidfVectorizer(min_df=5,max_df=.5),
   MultinomialNB()),
  ("LR-count",CountVectorizer(min_df=5,max_df=.5),
   LogisticRegression(solver="liblinear")),
  ("LR-TfIdf",TfidfVectorizer(min_df=5,max_df=.5),
   LogisticRegression(solver="liblinear")),
  ("DecisionTree-count",CountVectorizer(min_df=5,max_df=.5),
  tree.DecisionTreeClassifier()),
  ("DecisionTree-tfidf",TfidfVectorizer(min_df=5,max_df=.5),
  tree.DecisionTreeClassifier())]


for name, vectorizer, classifier in configs:
    print(name)
    X_train = vectorizer.fit_transform(tweets_train)
    X_test = vectorizer.transform(tweets_test)
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)
    print(classification_report(y_test, y_pred))
    print("\n")


# Note that only the first part of the code needs to be adjusted!
```

## Exercise 2: Question 5

Based on the output that the classifier prints, what classifier performs the best? In your answer, consider:

- What information you need to identify the best classifier
- What metric you base your conclusion (i.e., precision, recall, accurcay, or F1-score) on and why

Depending on the metric that you base your evaluation on, a different classifier could be considered best. The best metric in turn, differs depending on what a classifier is used for.

In this case, no specific information about the usage of the classifier is provided. Therefore, the F1-score may be the best metric to use.

## Exercise 2: Question 5

Based on the output that the classifier prints, what classifier performs the best? In your answer, consider:

- What information you need to identify the best classifier
- What metric you base your conclusion (i.e., precision, recall, accurcay, or F1-score) on and why

Depending on the metric that you base your evaluation on, a different classifier could be considered best. The best metric in turn, differs depending on what a classifier is used for.

In this case, no specific information about the usage of the classifier is provided. Therefore, the F1-score may be the best metric to use.

## Exercise 2: Question 6

Let's say that you base your evaluation on the F1-score of the classifier. You can choose between the macro average and the weighted average of the F1-score. Check out the scikit learn documentation (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html#sklearn.metrics.precision_recall_fscore_support). What F1-value (macro average or weighted average) would you select?

From the documentation:

'macro': Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

'weighted': Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

It seems that there are relaitvely many normal tweets present in the dataset. To account for this, the weighted average may be a good choide.

## Exercise 2: Question 7

When looking at the classification report, you will see another column indicating values for something labelled 'support'. Can you do some searching online and find out what 'support' is?

Support refers to the number of true instances for each label (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)

## Exercise 2: Question 8

Two researchers want to use the classifier to distinguish between tweets that are spam or hateful and tweets that are not (either because they are normal or abusive). They are, however, not happy with the performance of the classifier when looking at the accuracy, precision, and recall for the spam category or the hateful category. One of the researchers suggests to first recode the labels, so that all tweets that were annotated as spam receive a label 'spam' or 'hateful' are grouped together and all other tweets are grouped together as well.

What would the consequences be of doing so? What can you do to check your own answer? Try to recode the labels and see what happens!

It could increase the performance of the machine because the groups are no made larger and more easy to dinstinguish. You can recode the labels and train/validate the machines again to see if this is indeed what happens!

## Exercise 2: Question 9

For now, let's say that the classifier based on a count vectorizer and Logistic Regression is the one we prefer. We now want to use this model to predict the label for new data that we have not annotated (remember, this was the whole goal of SML)!

To do this, let's save our classifier and our vectorizer to a file. If we don't do this, we would need to re-train our model every time we want to use it. This is not so convenient, for example, we would always need to have our training data at hand. The code below shows you how to make a vectorizer and train a classifier (a repetition of what we did before to show you the whole process) and store them into a file.

In the code, you will see that both the classifier and the vectorizer are stored into a file. Why do you need to store both (why not just store the classifier only)?

The vectorizer needs to be stored as well. Once a vectorizer is fit on certain data, fitting it again on new data changes the set up of the vectorizer!

## Overfitting

- When your model fits to the training data very well, but is focused too much on (random) peculiarities of that data and can’t perform well on new data anymore
- We split the data into a training set and a test set

## Hyperparameter tuning

- Hyperparameters: parameters that explicitly specified
- We run the risk of overfitting on our test data
- We split our data into three groups instead:
  - Training dataset → train the model
  - Validation dataset → tune hyperparameters
  - Test dataset → evaluate model performance

## Up next

| **Time:** | **Topic:** |
|---|---|
| ~~09:30 – 11:00~~ | ~~Basics of (supervised) machine learning~~ |
| ~~11:00 – 12:00~~ | ~~Exercise 1~~ |
| ~~12:00 – 13:00~~ | ~~Break~~ |
| ~~13:00 – 13:30~~ | ~~Recap on Exercise 1~~ |
| ~~13:30 – 14:30~~ | ~~Evaluating models~~ |
| ~~14:30 – 15:30~~ | ~~Exercise 2~~ |
| ~~15:30 – 16:00~~ | ~~Recap on Exercise 2~~ |
| 16:00 – 16:30 | Closure |

Image: https://pixabay.com/nl/vectors/koffie-kop-koffie-beker-cafe%c3%afne-5009730/