HOME

a HOPEFULLY-SMART NEWS AGGREGATOR

Data Mining Project

Filippo Scotto

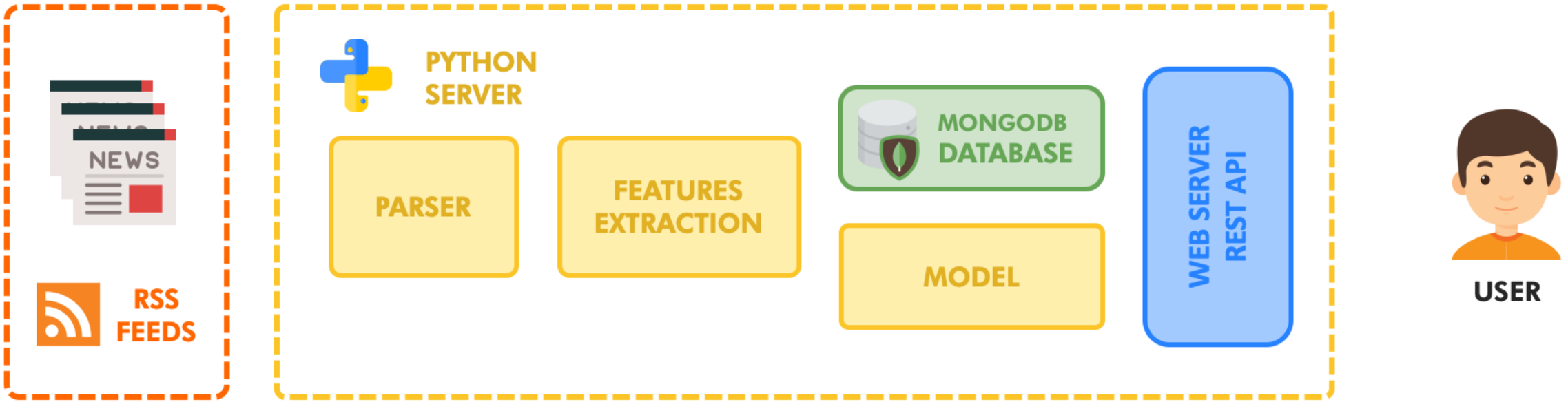
INTRODUCTION

There are thousands of news sites, but most of their articles are *not for everybody*. Can we build a *machine learning based* system capable to **filter out** what we are not interesting in?

We need a system capable to classify the **category** of the articles and correctly predict their **likability**.

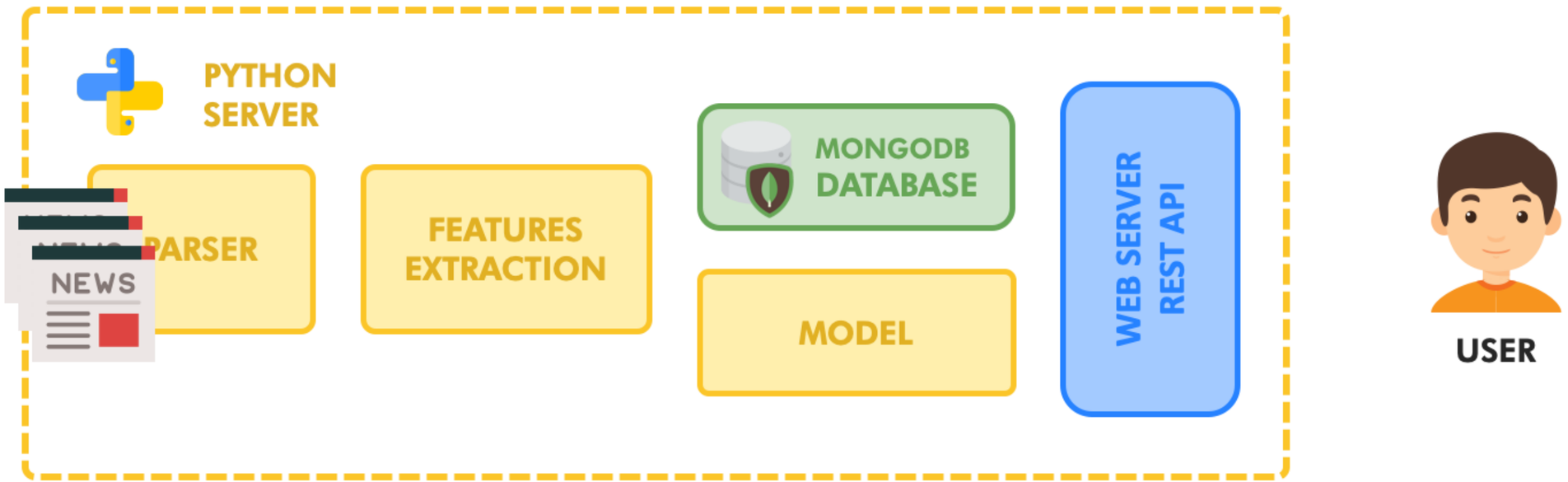
- The designed system will parse the **RSS feeds** coming from the most popular italian news sites;
- It will **preprocess** the data and **extract some features;**
- It will **classify** the news and predict wheter the user may like it or not.

THE SYSTEM



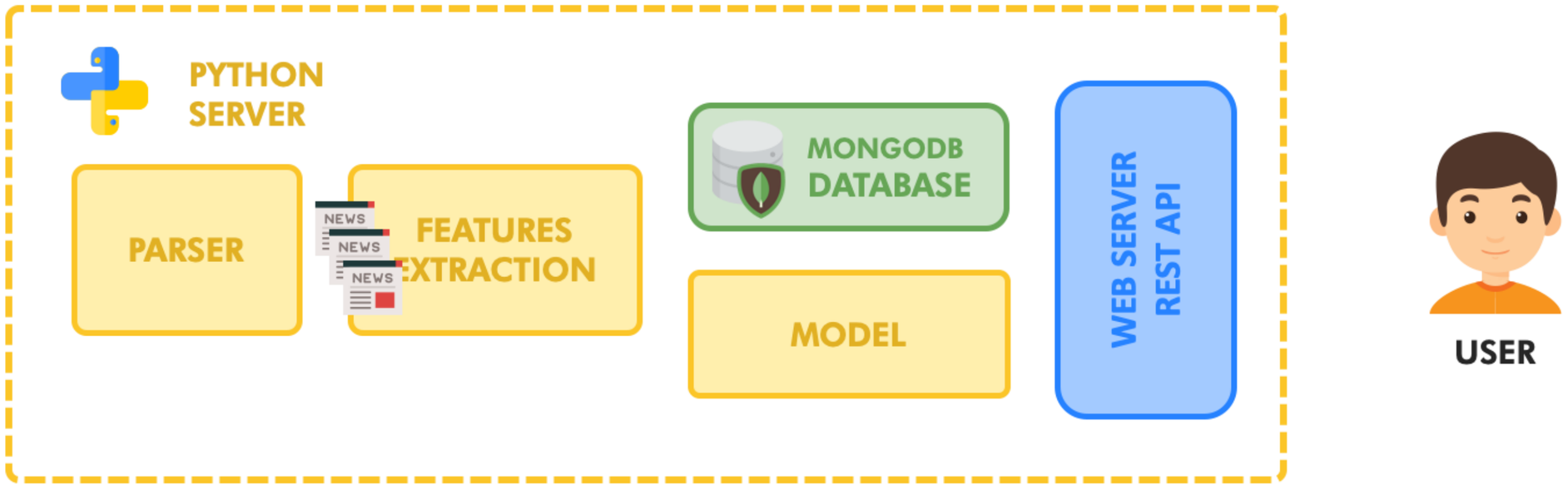
1 The articles are downloaded from the RSS Feeds

THE SYSTEM



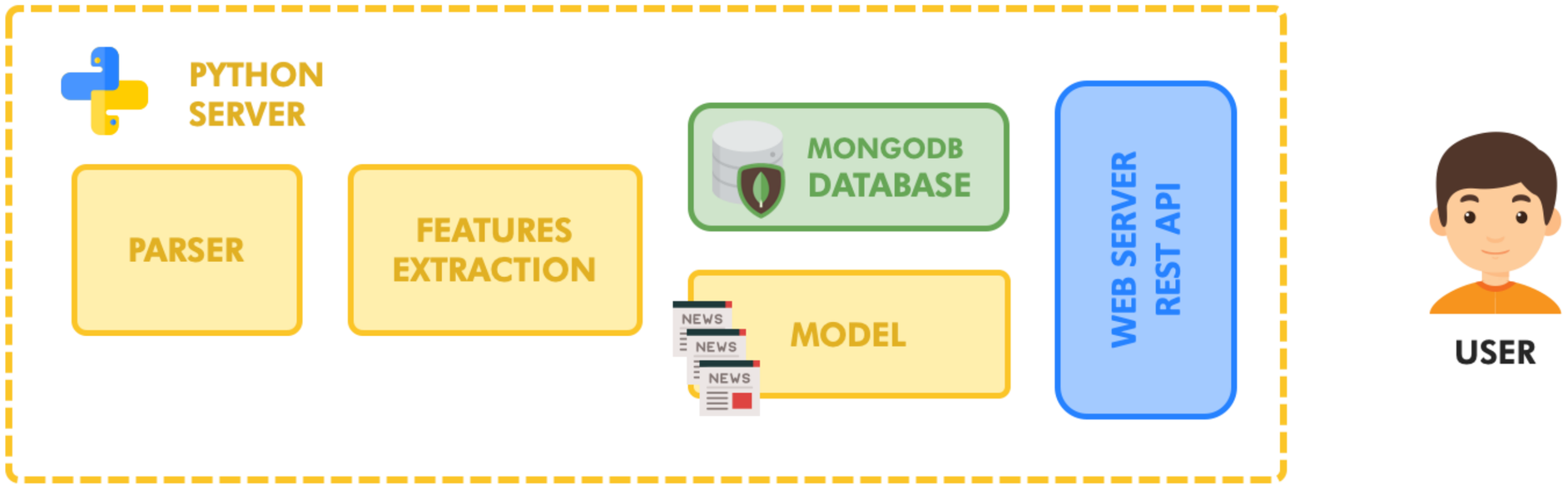
2 The articles are parsed by a Python Script

THE SYSTEM



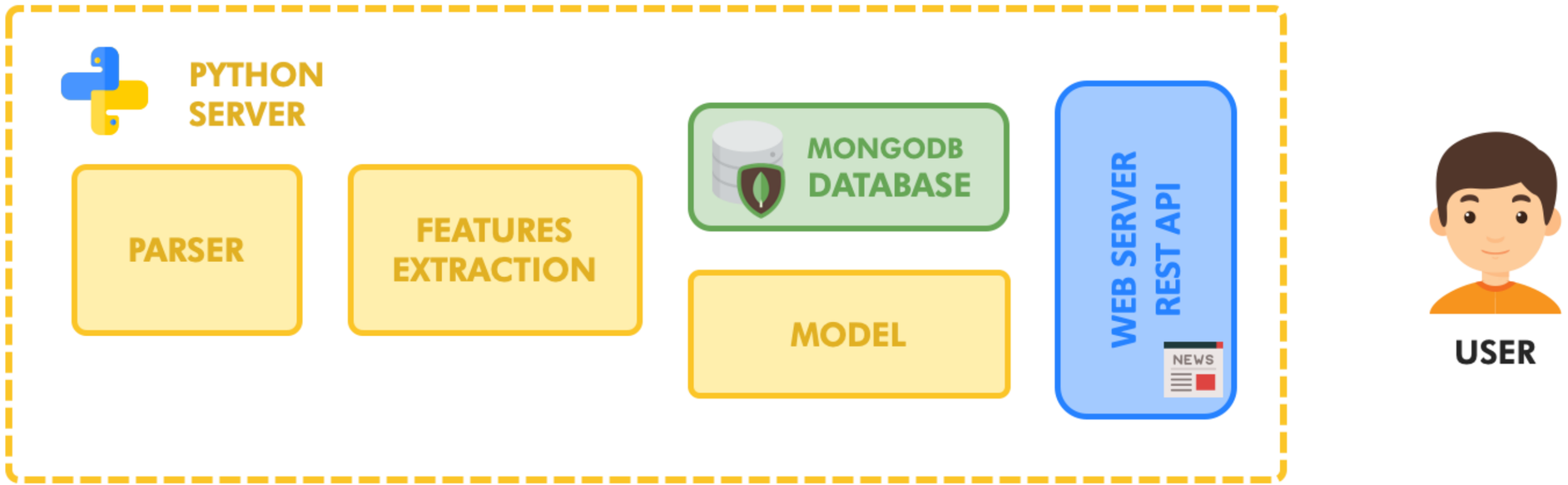
3 **The parsed articles are processed to extract some featuers**

THE SYSTEM



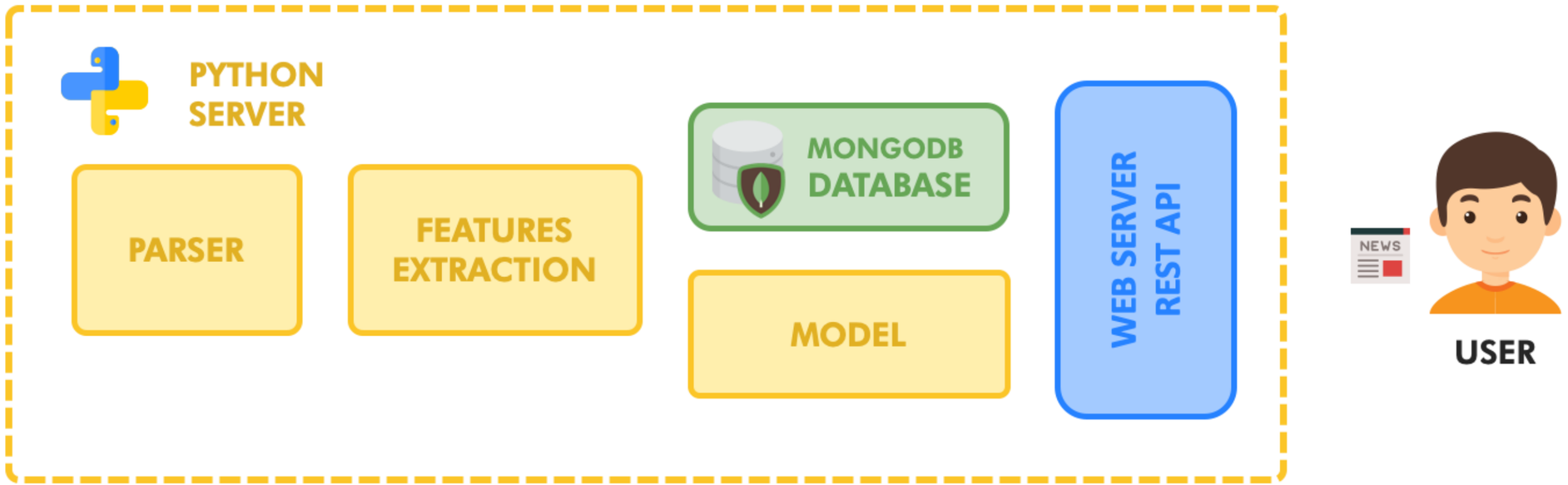
4 The features are given as input to a Machine Learning Model

THE SYSTEM



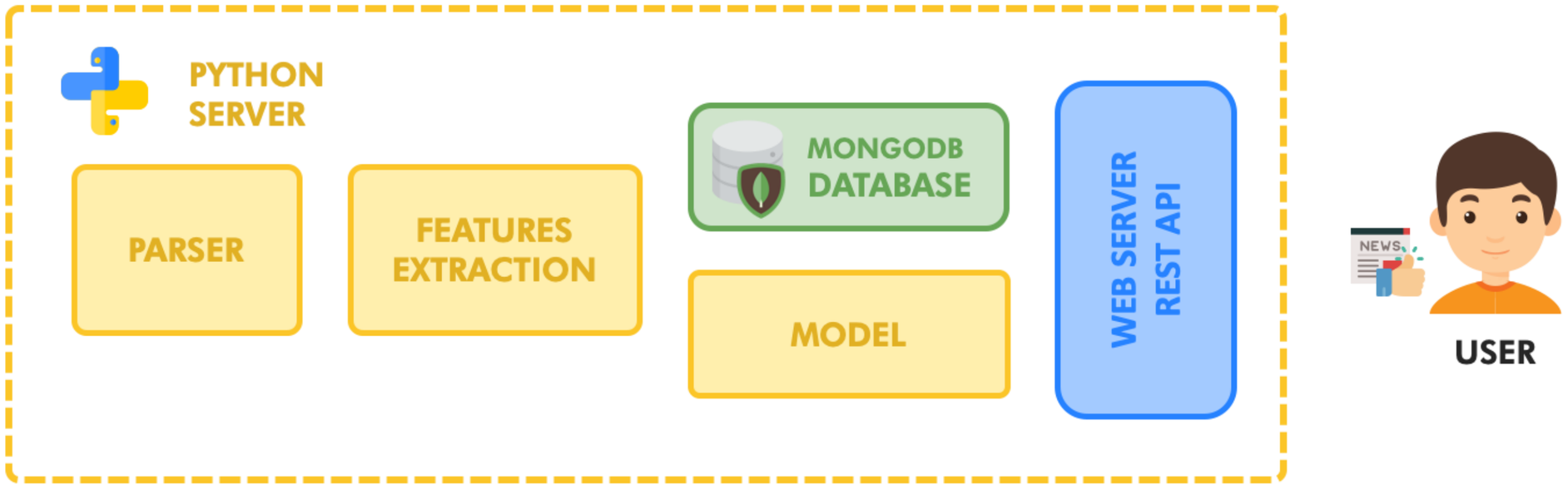
5 **The filtered news are passed to the webserver**

THE SYSTEM



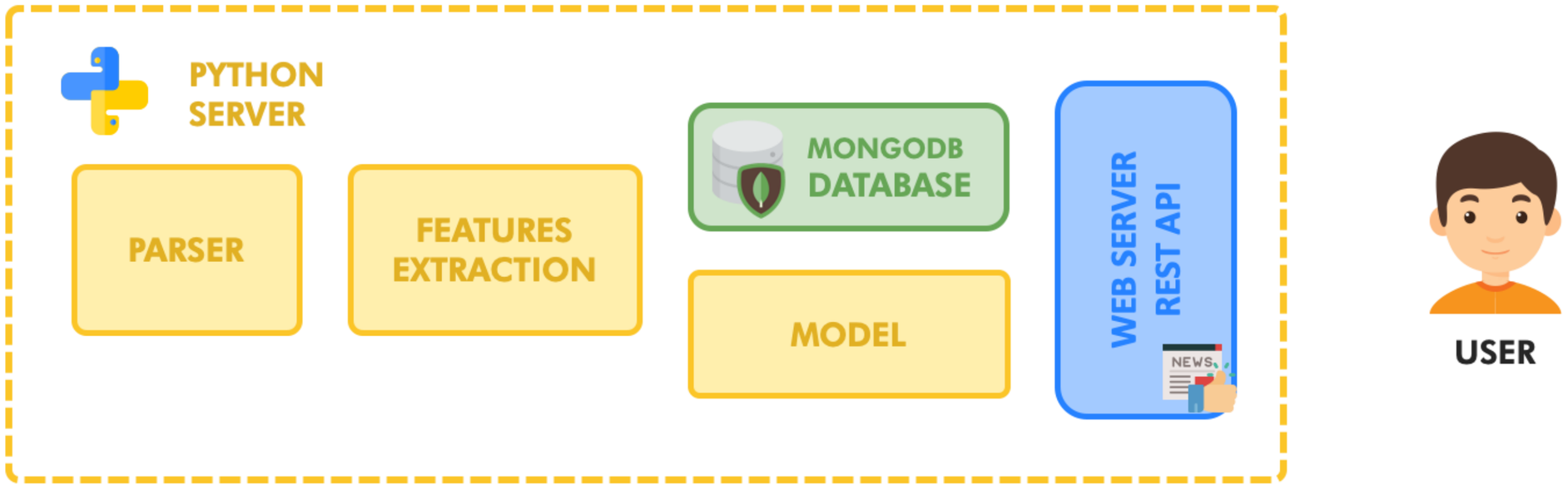
6 **The user can request its personalized feed using a REST API**

THE SYSTEM



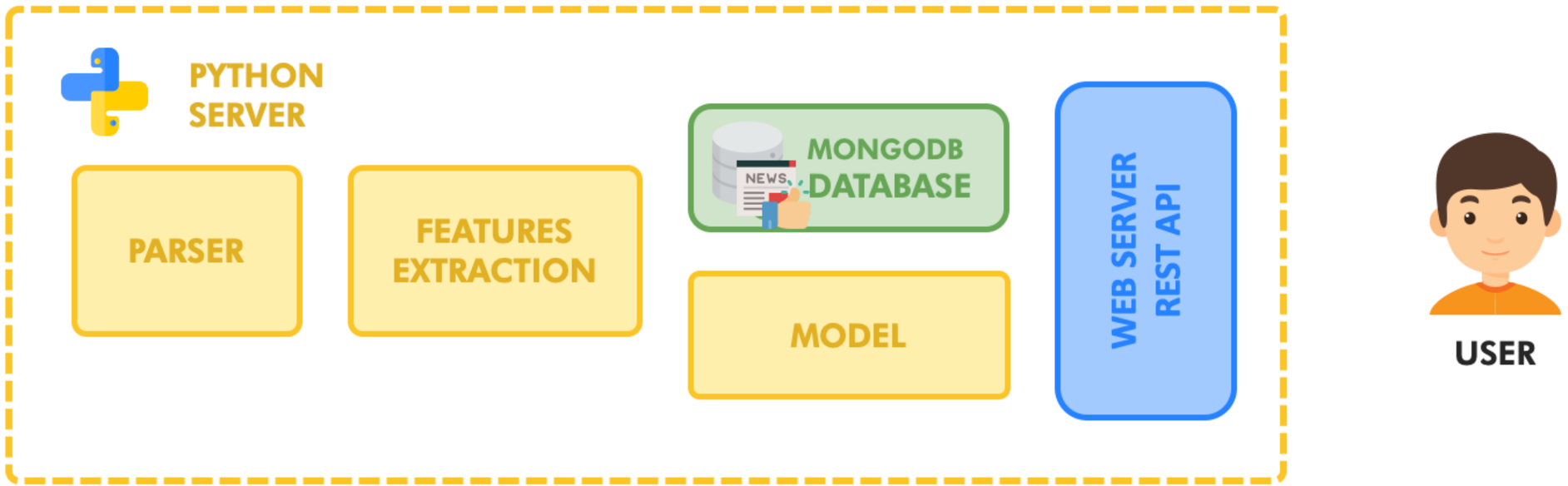
7 **The user can “like” the article using the WebApp**

THE SYSTEM



8 **The like request is processed by the WebServer**

THE SYSTEM



9 Finally the database entry for the liked article is updated

THE PARSER MODULE

News are collected from the RSS feeds from the most popular italian news sites. Unfortunately RSS feeds are not as structured as they were ment to be…

- **Missing values** and **different tag names**
- Article descriptions containing **images** or **raw HTML**
- We don’t need **everything**!

NEWS FEED

PYTHON SCRIPT

ADJUSTED FEED

THE PARSER MODULE: AN EXAMPLE

In the end what we obtain as output from the parser is a list of articles with the following structure:

Title
The title of the article

Author
The author of the article

Description
A *text-only snippet* of the content

Link
Hyperlink to the source

Timestamp
Published date-time

Image
Header image of the article

Missing *and* ***not valid values*** *will be replaced by an empty string!*

BUILDING THE DATASET

Once the feeds are parsed, we have to build a dataset for our model. In particular what we need to do is:

1. **Manually tag the articles**: each will be labelled with a category (nine different categories available)
2. **Manually Like/Dislike the articles**

To do this a **WebApp** was built: the app will show all the articles coming from the feed and then:

1. The article must be liked/disliked

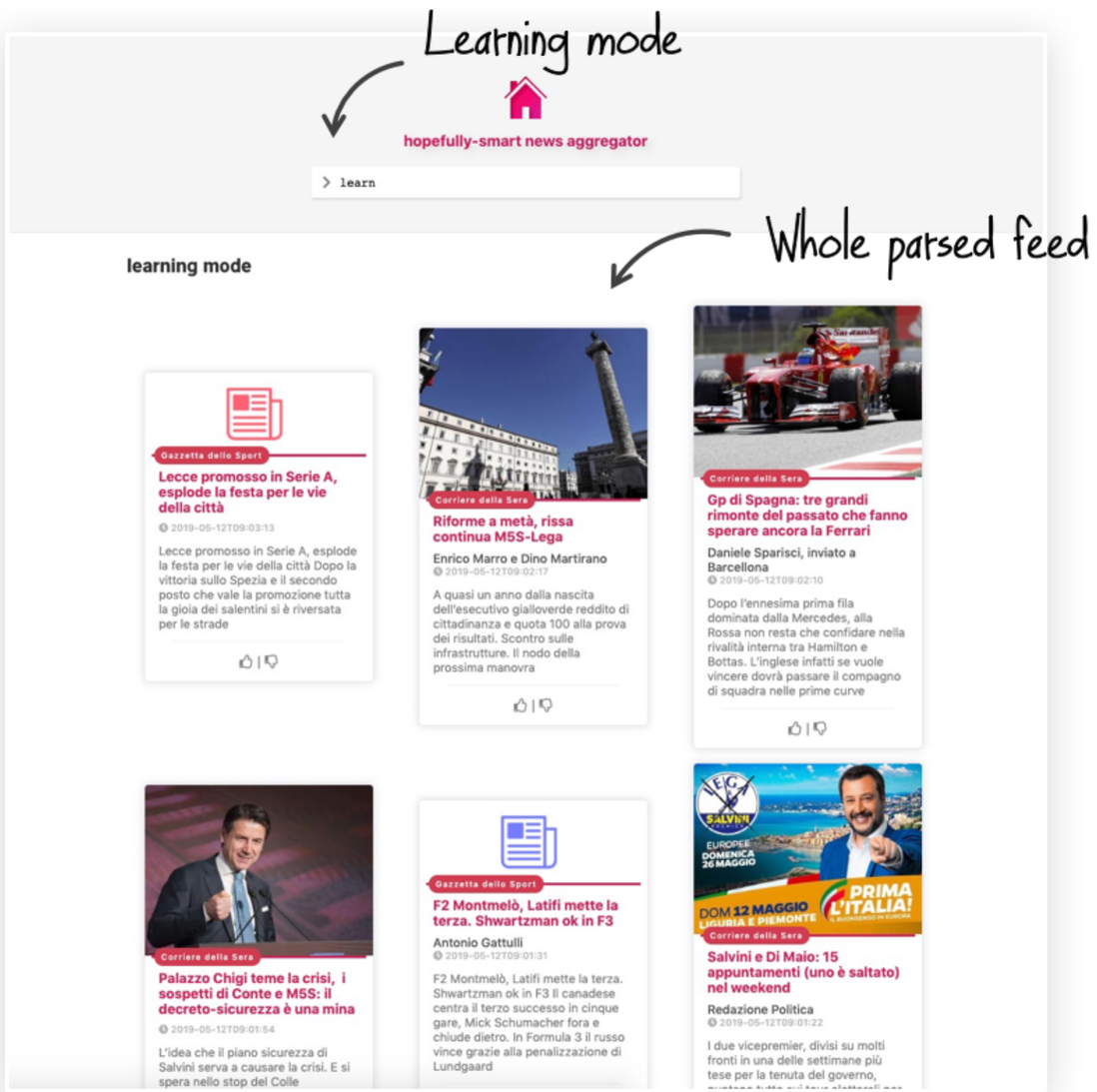
Like/Dislike buttons

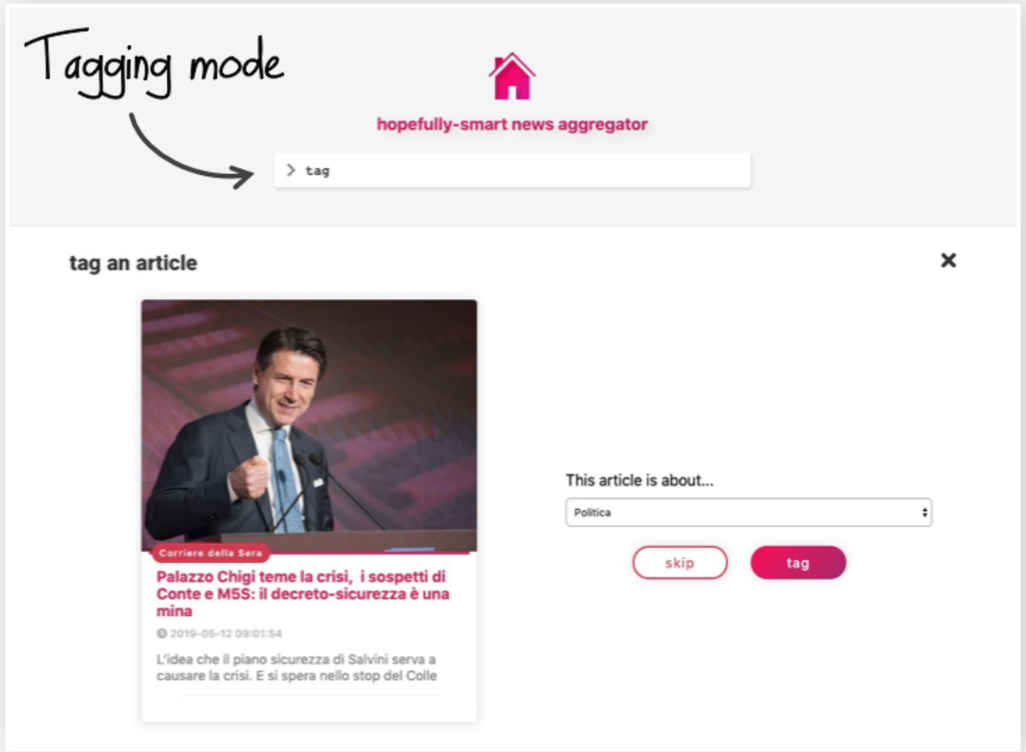
2. The liked/disliked article will be inserted in the database and can be tagged

Form to tag an article

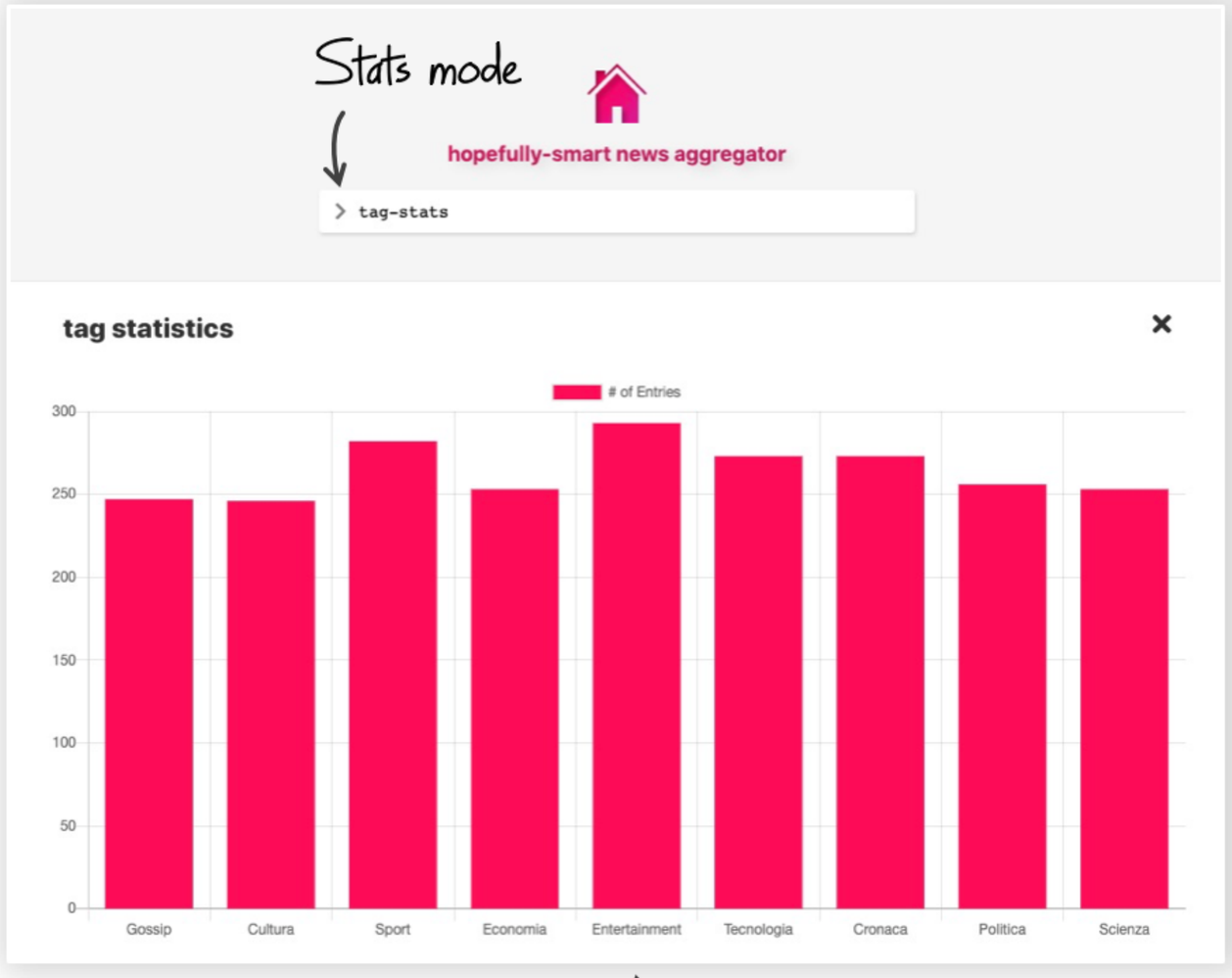
Button to skip current article and tag the next one

BUILDING THE DATASET: THE WEBAPP





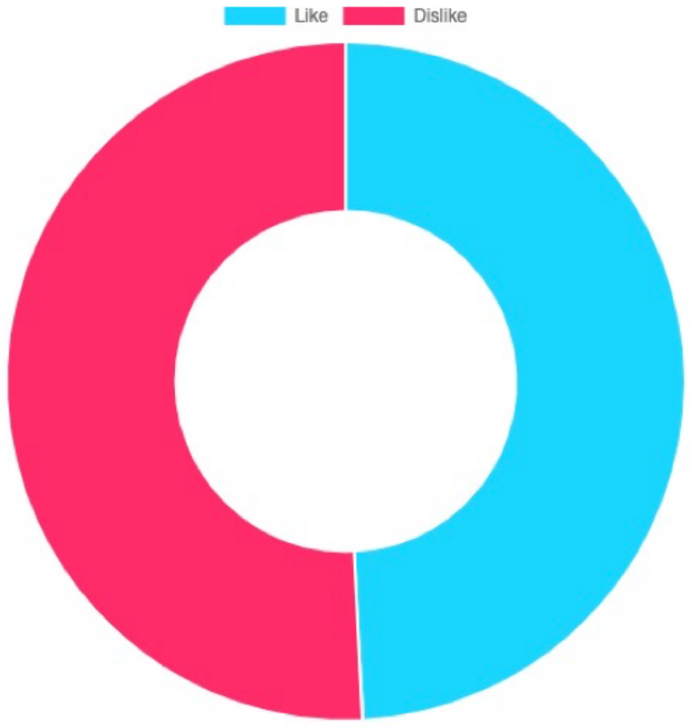
BUILDING THE DATASET: THE NEWS CATEGORIES



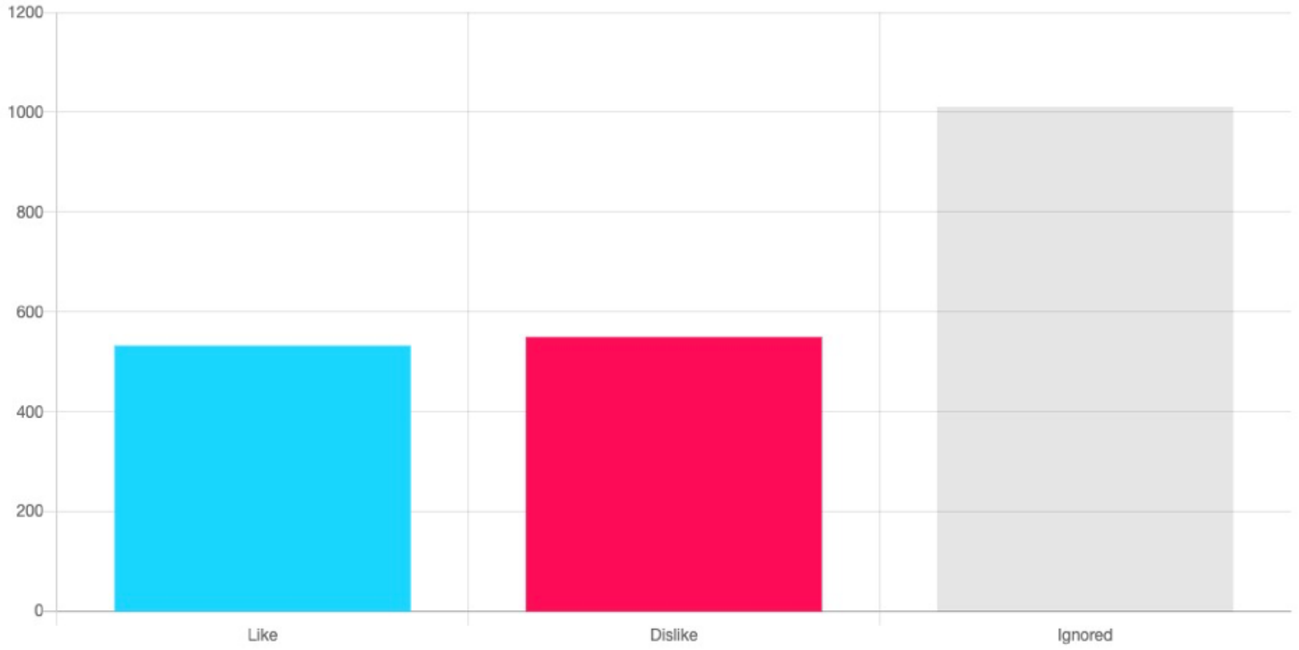
Tag distribution

- **9 different categories**
- **3 months** news (since **March 2019**)
- Over **30 sources**
- **~ 250** articles per category

BUILDING THE DATASET: THE LIKE/DISLIKE DISTRIBUTION



Like/Dislike distribution



Extra articles, useful to improve the category classification

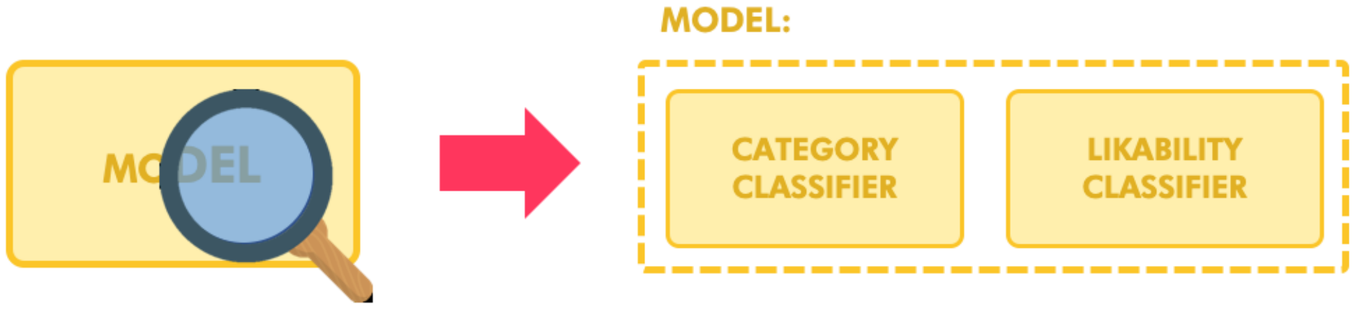
EXTRACTING THE FEATURES

Now we need to mine the news in order to extract some useful features.

1. **Natural Language Processing:**
    a. **Normalization**
    b. **Tokenization**
    c. **Stemming**
    d. **Ignoring Stopwords**
2. **Vectorizer**

Input: Article title + article description
Output: Sparse matrix containing the TF-IDF scores for the article's terms

BUILDING THE MODEL



The machine learning model that we are going to use it's actually composed by **two classifiers**:

CATEGORY CLASSIFIER

Predict the category for the article knowing the matrix of TF-IDF scores.

LIKABILITY CLASSIFIER

Predict wheter the user will like the article or not knowing the TF-IDF matrix and the category.

THE CATEGORY CLASSIFIER: BASE ESTIMATORS

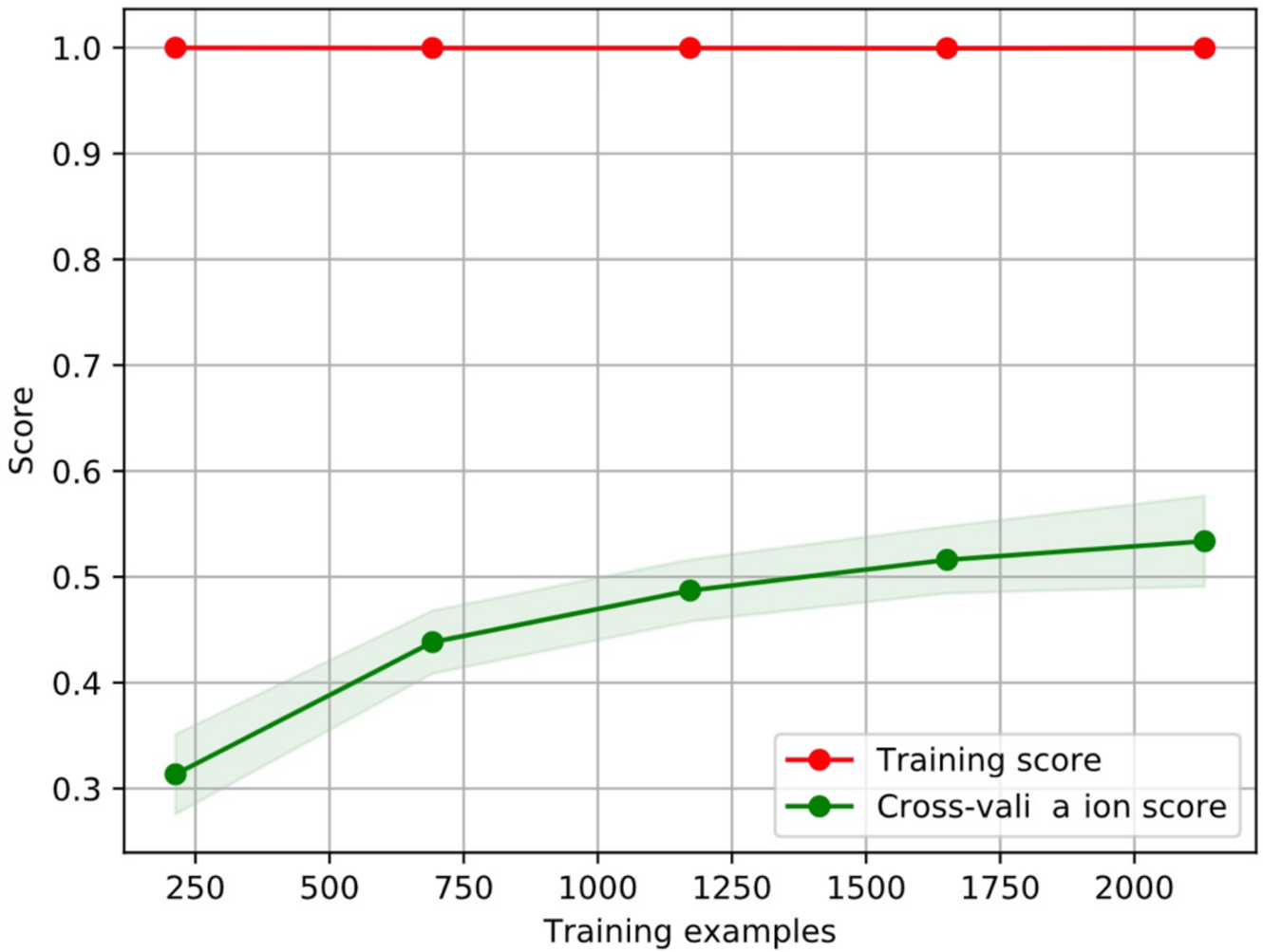
Here are reported the **scores** and **performance** of some **simple classifiers** trying to predict the **category** of an article:

CLASSIFIER	F1-SCORE		ACCURACY		PRECISION		RECALL	
	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV
Decision Tree	0.5276	0.0327	0.5267	0.0333	0.5447	0.0333	0.5267	0.0333
MN-Bayes	0.7704	0.0142	0.7729	0.0152	0.7899	0.0169	0.7729	0.0152
LinearSVC	0.8002	0.0203	0.8015	0.0195	0.8055	0.0211	0.8015	0.0195
LogisticRegression	0.7862	0.0209	0.7878	0.0209	0.7932	0.0223	0.7878	0.0209

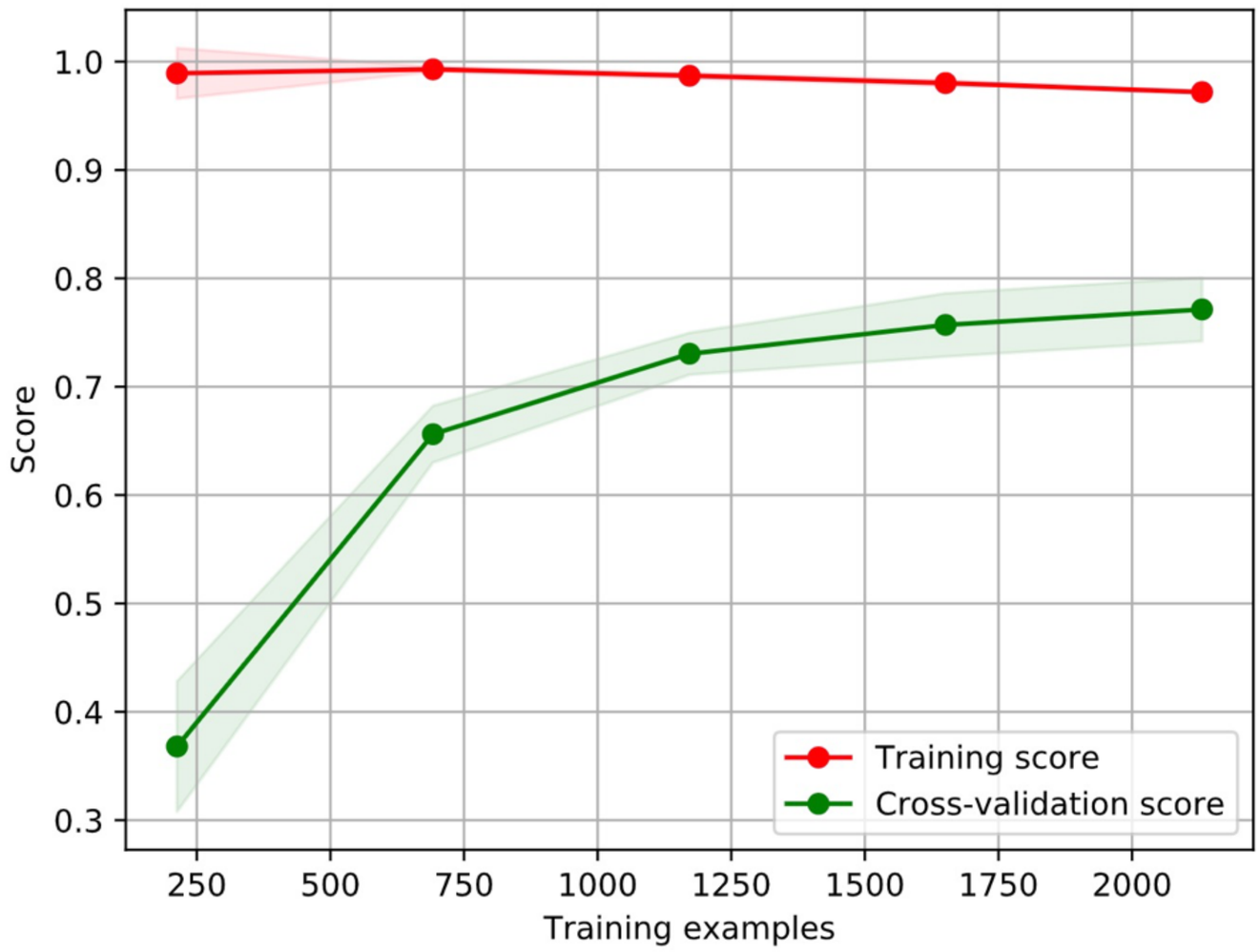
Best score!

The following classifiers were **tuned** using a GridSearch algorithm! The testing and validation were performed using **10 Folds Cross Validation**

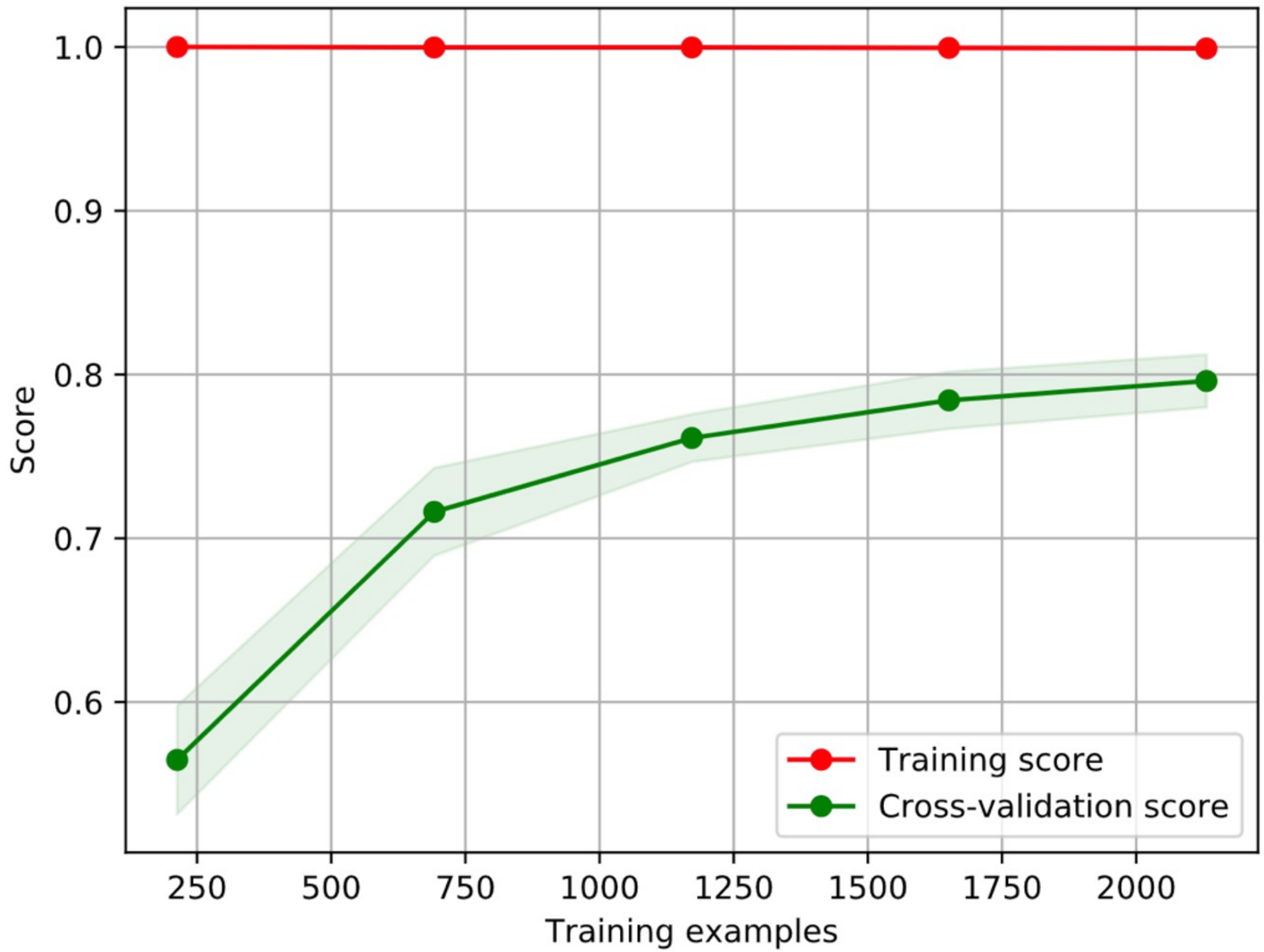
THE CATEGORY CLASSIFIER: DECISION TREE LEARNING CURVE



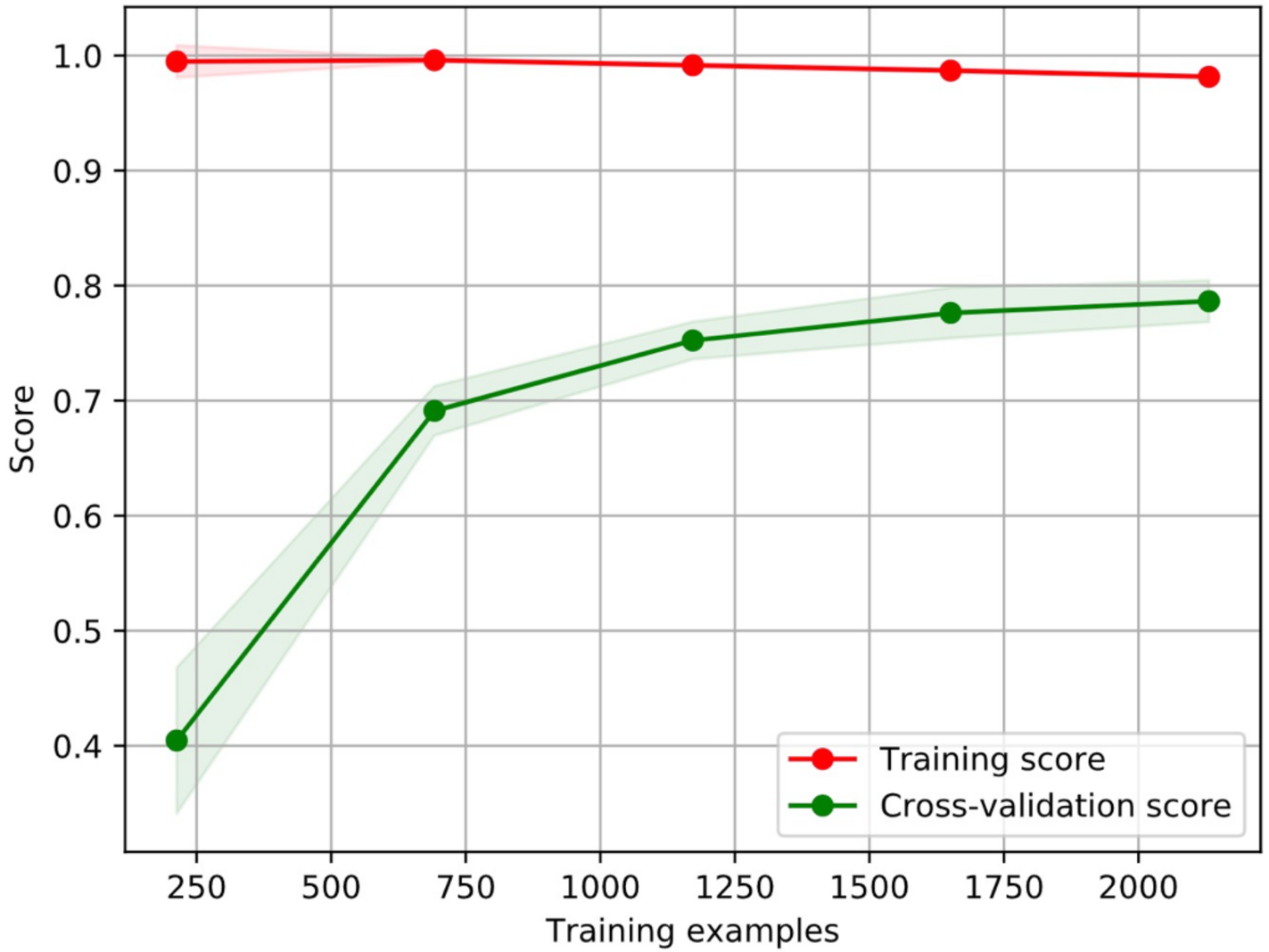
THE CATEGORY CLASSIFIER: MULTINOMIAL BAYES LEARNING CURVE



THE CATEGORY CLASSIFIER: LINEAR SVC LEARNING CURVE



THE CATEGORY CLASSIFIER: LOGISTIC REGRESSION LEARNING CURVE



THE CATEGORY CLASSIFIER: ENSEMBLES AND METACLASSIFIERS

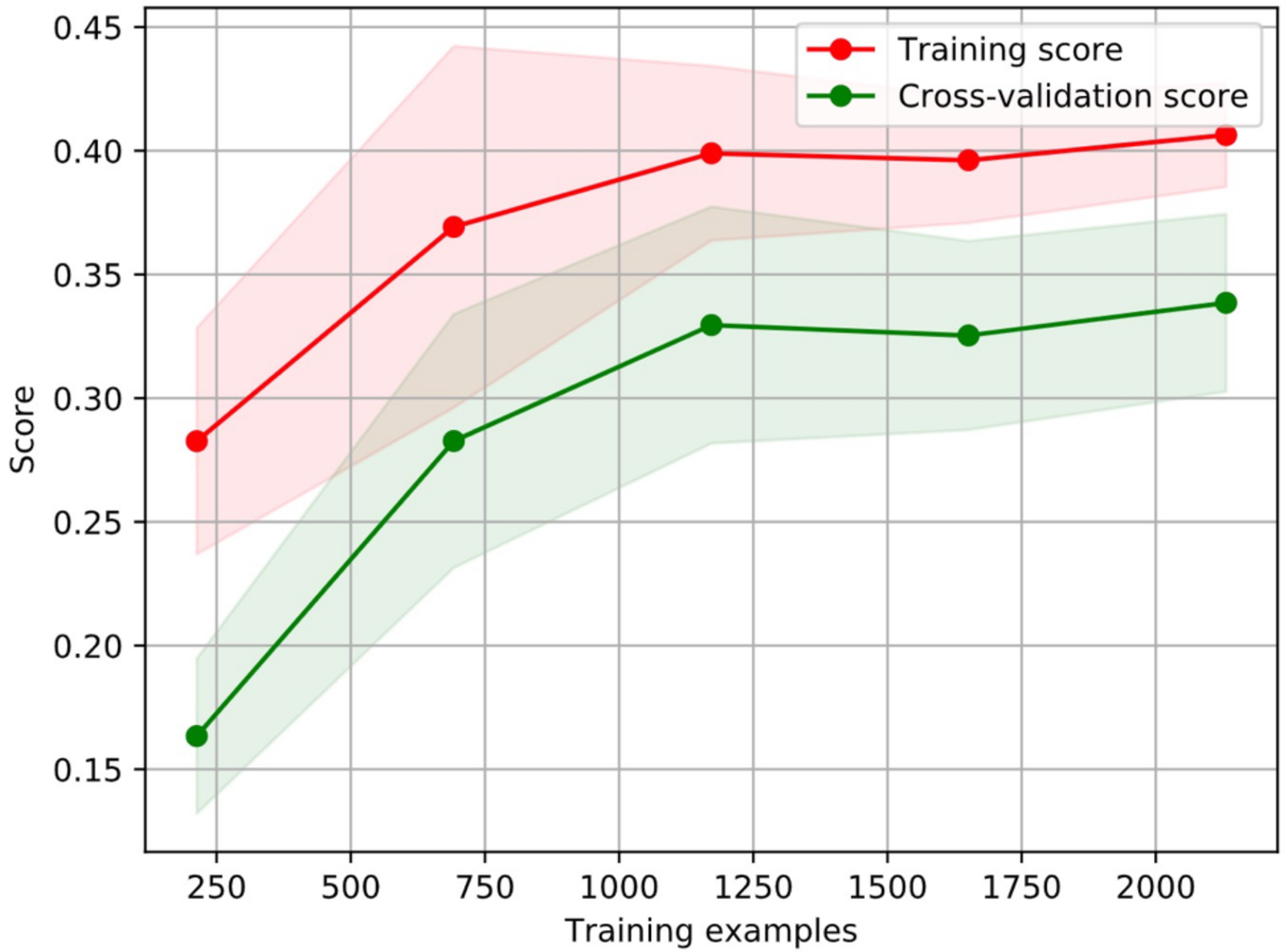
Here are reported the **scores** and **performance** of some **ensemble classifiers** trying to predict the **category** of an article:

CLASSIFIER	F1-SCORE		ACCURACY		PRECISION		RECALL	
	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV
D. Tree + ADABoost	0.3617	0.0347	0.3597	0.0248	0.5272	0.0493	0.3597	0.0248
Random Forest	0.6852	0.0286	0.6869	0.0274	0.6949	0.0285	0.6869	0.0274
Voting	0.7911	0.0208	0.7924	0.0207	0.7989	0.0228	0.7924	0.0207
Bagging	0.7960	0.0203	0.7970	0.0195	0.8016	0.0211	0.7970	0.0195
XGBClassifier	0.6649	0.0224	0.6631	0.0228	0.6753	0.0209	0.6631	0.0228
Stacking	0.7541	0.0147	0.7557	0.0147	0.7686	0.0127	0.7557	0.0147

Best scor
but worse
than SVC

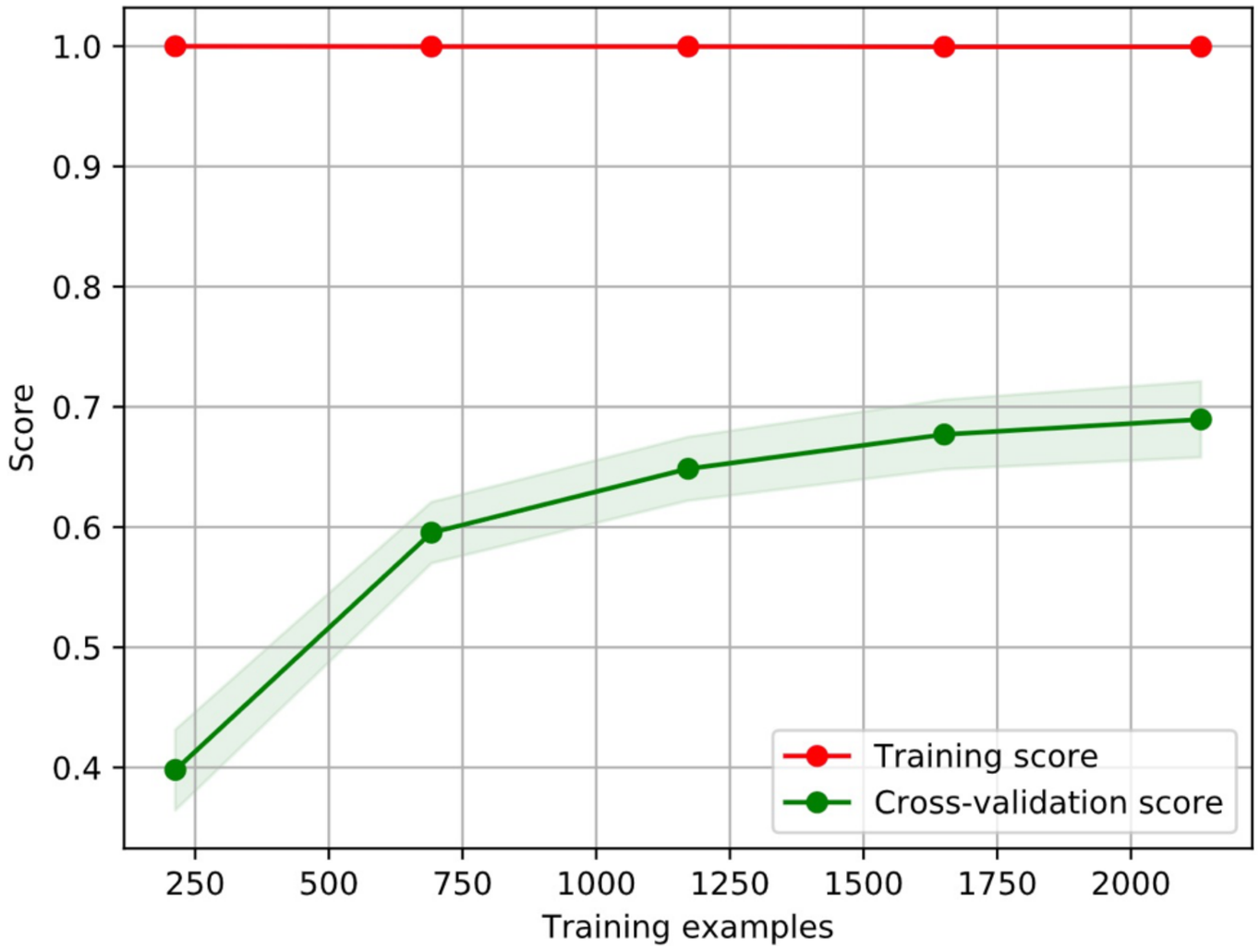
The following classifiers were **tuned** using a GridSearch algorithm! The testing and validation were performed using **10 Folds Cross Validation**

THE CATEGORY CLASSIFIER: ADABOOST LEARNING CURVE



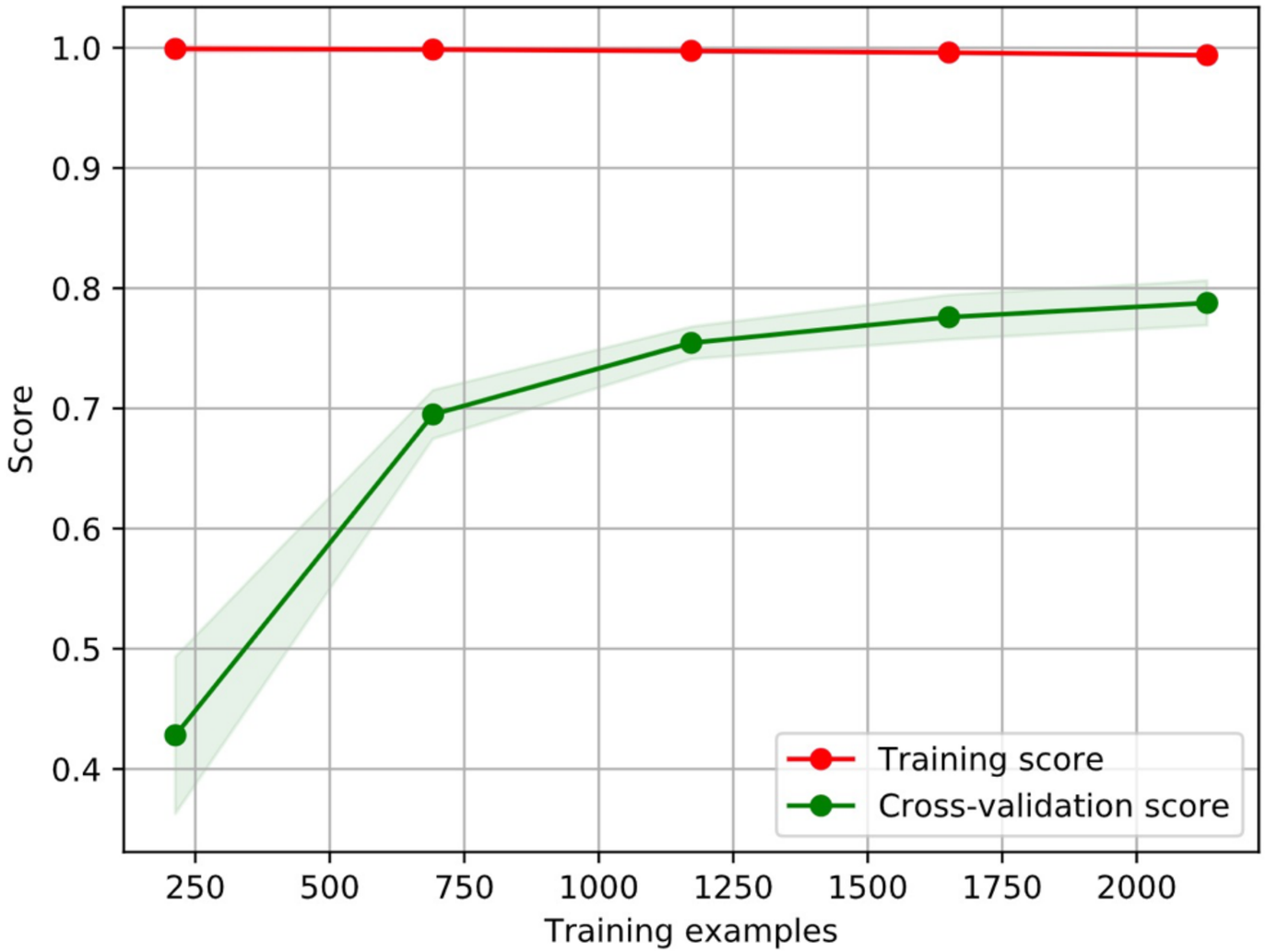

```
parameters {
  'base': DecisionTree(),
  'n_estimators': 100
}
```

THE CATEGORY CLASSIFIER: RANDOM FOREST LEARNING CURVE

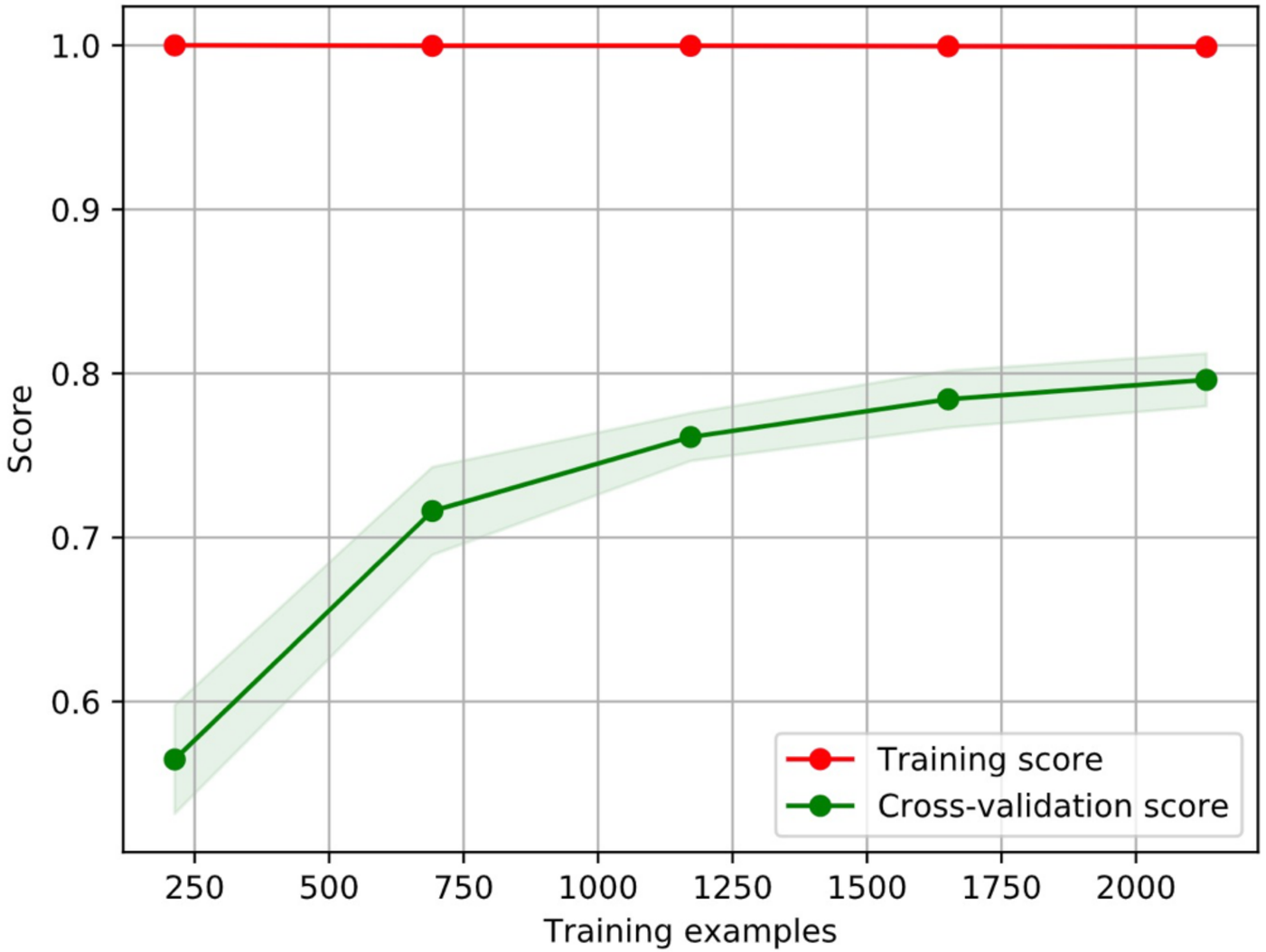



```
parameters {
  'n_estimators': 100
}
```

THE CATEGORY CLASSIFIER: VOTING LEARNING CURVE

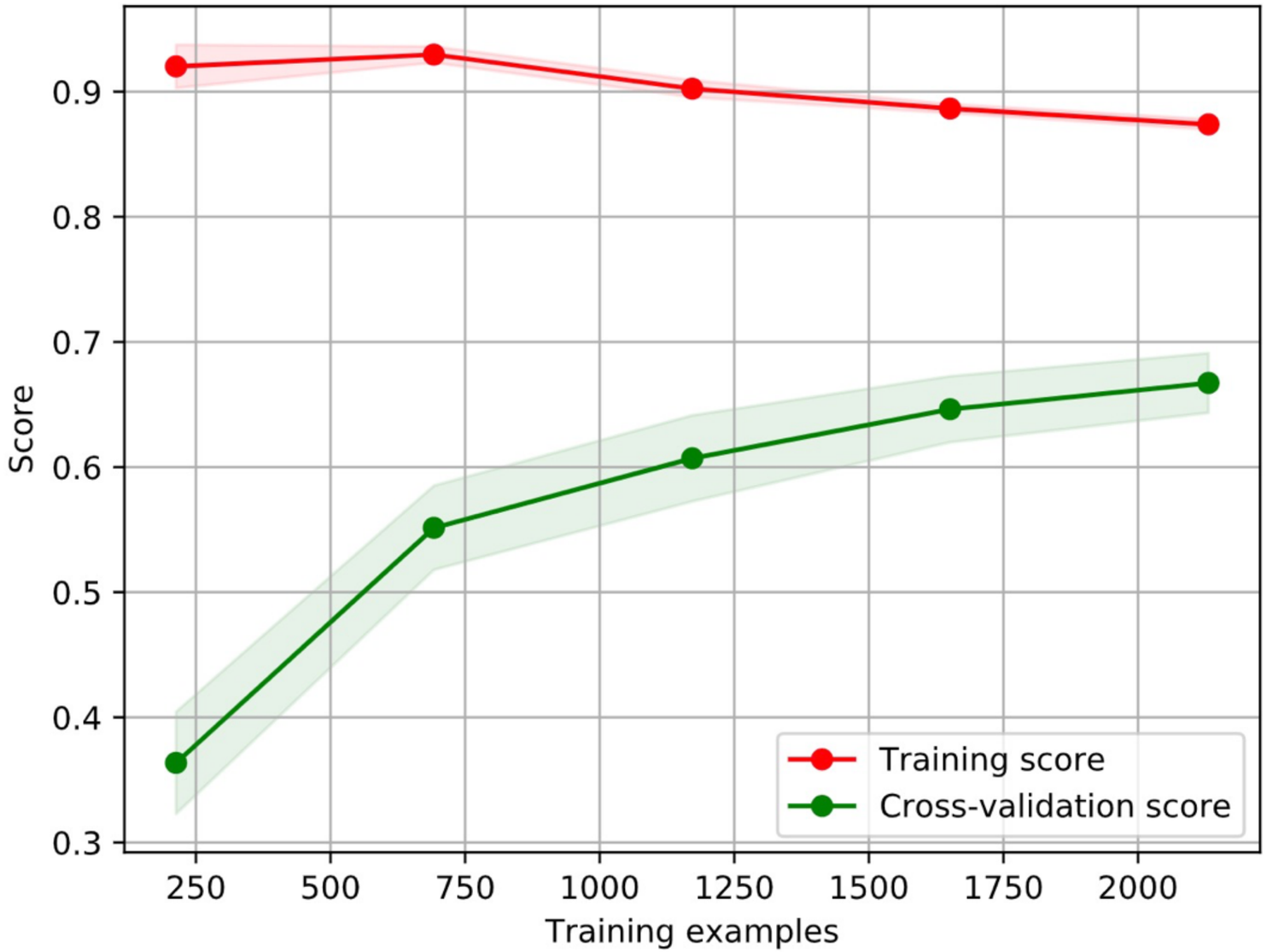


THE CATEGORY CLASSIFIER: BAGGING LEARNING CURVE



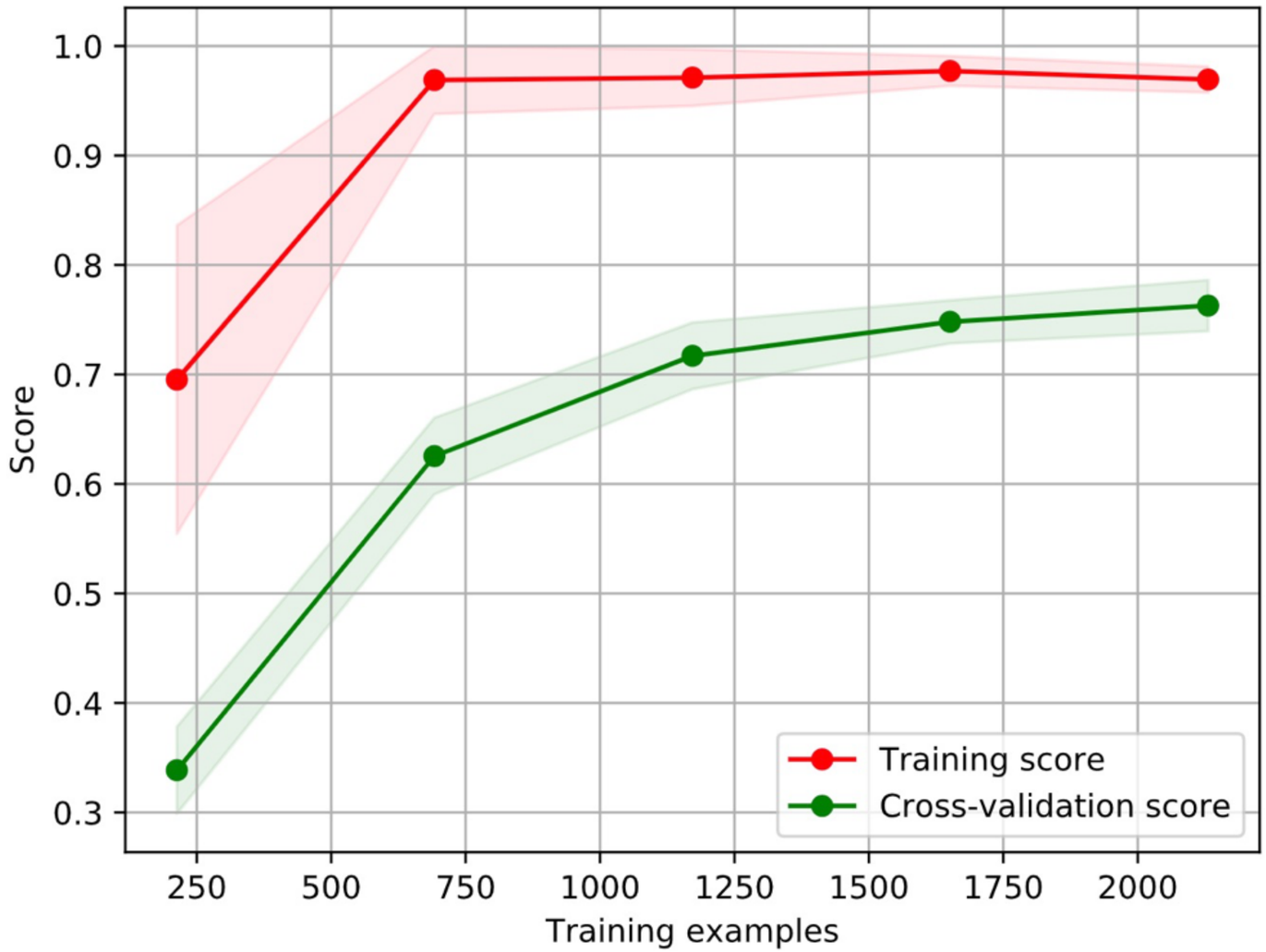

```
parameters {
  'base': LinearSVC(),
  'n_estimators': 100
}
```

THE CATEGORY CLASSIFIER: XGB LEARNING CURVE



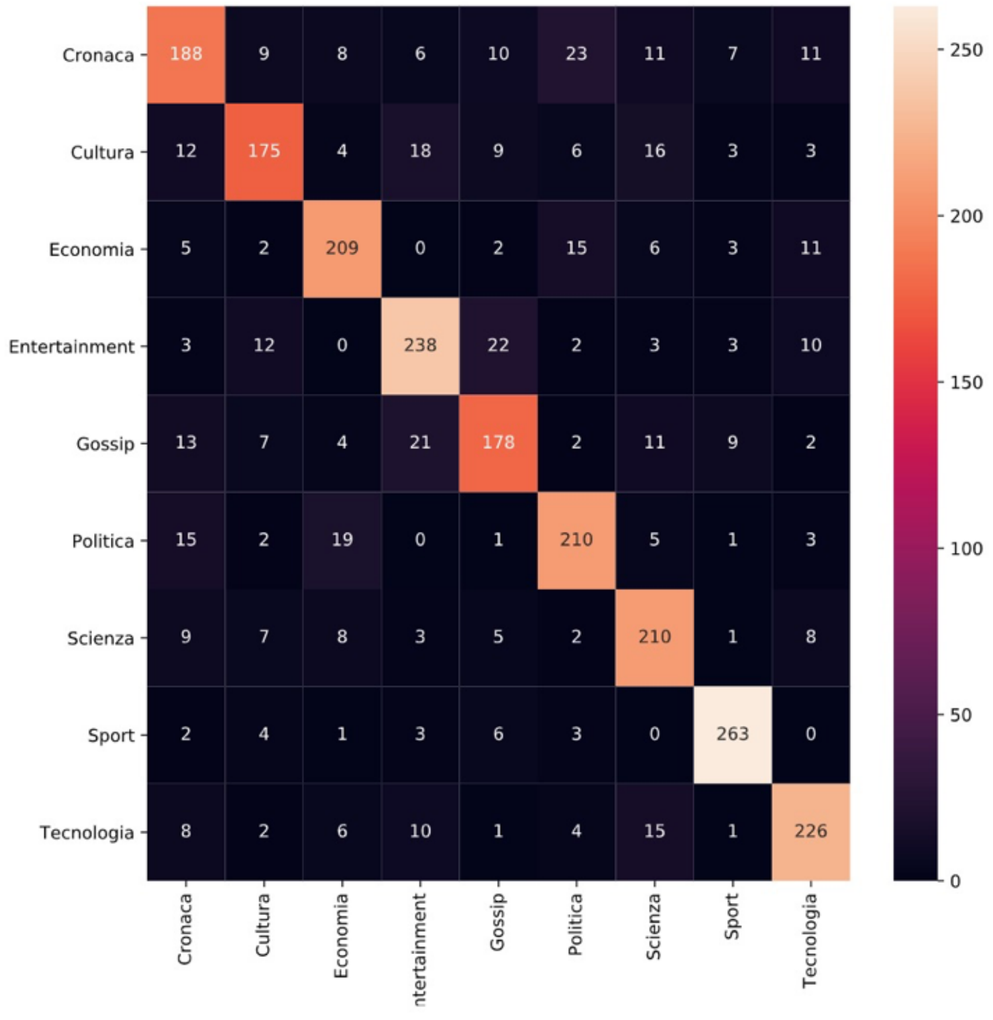

```
parameters {
  'n_estimators': 100,
  'max_depth': 4,
  'learning_rate': 0.9
}
```

THE CATEGORY CLASSIFIER: STACKING LEARNING CURVE



THE CATEGORY CLASSIFIER

In the end, no classifier did better than the base **LinearSVC**, so it was chosen as the classifier that should predict the article category. Here is shown its **confusion matrix**:



The chosen classifier has an 80% accuracy, is this good?
*Indeed it could be better, however predicting the category of an article is not an easy task (not even for a human), so in the end we can say that it is **fair enough.***

THE LIKABILITY CLASSIFIER: BASE ESTIMATORS

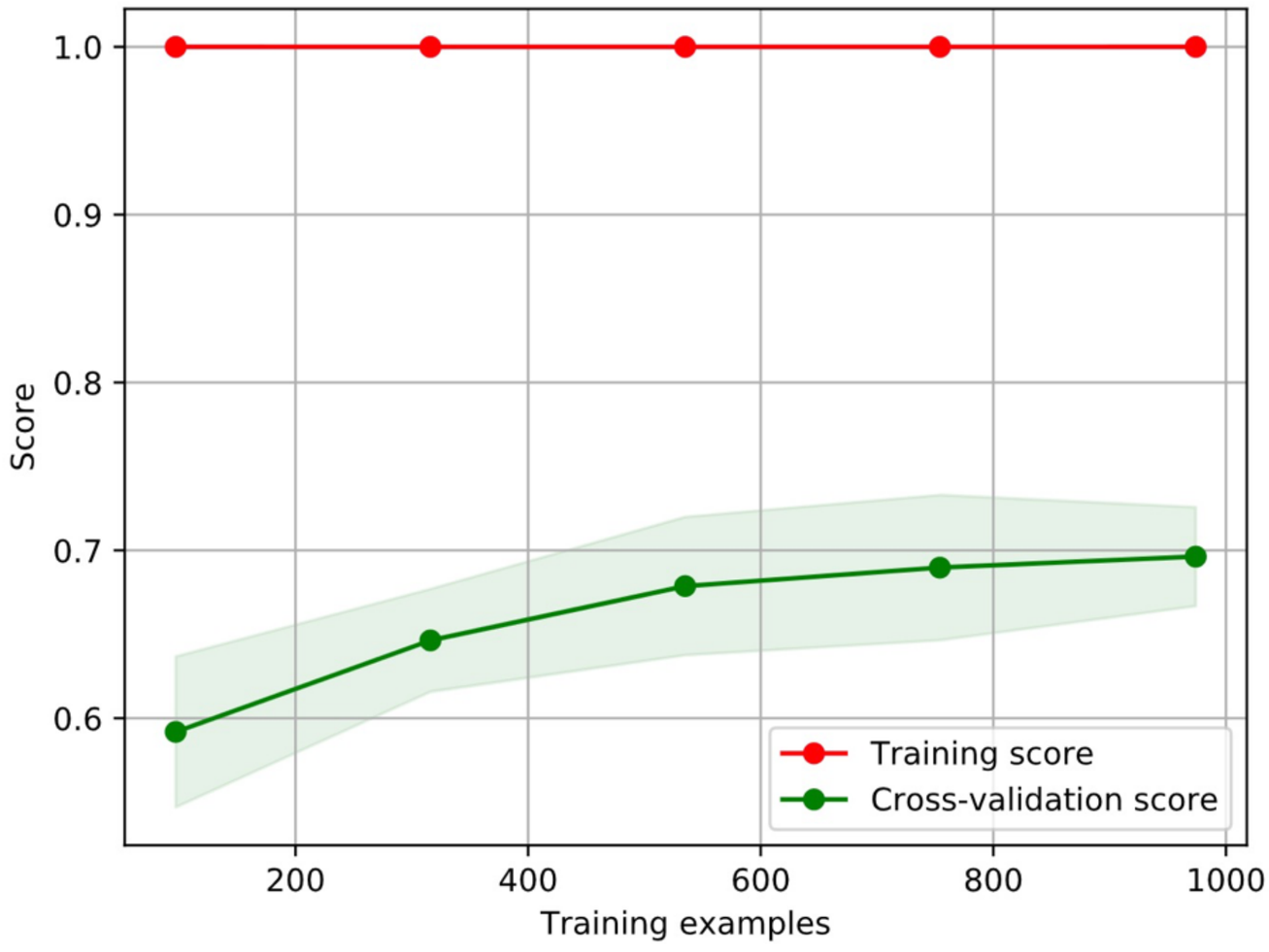
Here are reported the **scores** and **performance** of some **simple classifiers** trying to predict the **likability** of an article:

CLASSIFIER	F1-SCORE		ACCURACY		PRECISION		RECALL	
	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV
Decision Tree	0.6967	0.0275	0.6972	0.0275	0.6983	0.0276	0.6972	0.0275
MN-Bayes	0.8607	0.0244	0.8615	0.0241	0.8680	0.0242	0.8615	0.0241
LinearSVC	0.8669	0.0286	0.8670	0.0286	0.8690	0.0287	0.8670	0.0286
LogisticRegression	0.8725	0.0312	0.8726	0.0312	0.8739	0.0316	0.8726	0.0312

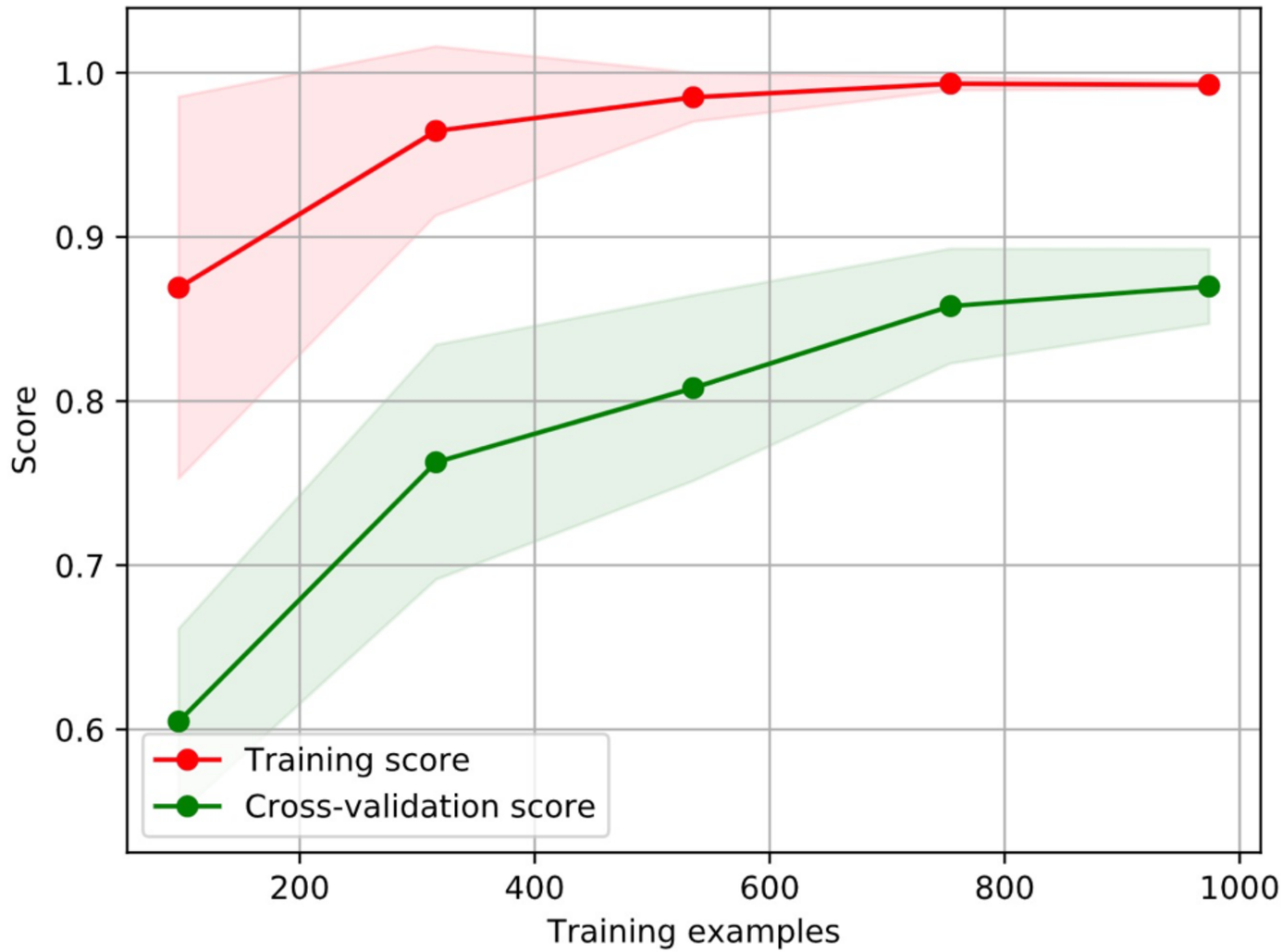
Best score!

The following classifiers were **tuned** using a GridSearch algorithm! The testing and validation were performed using **10 Folds Cross Validation**

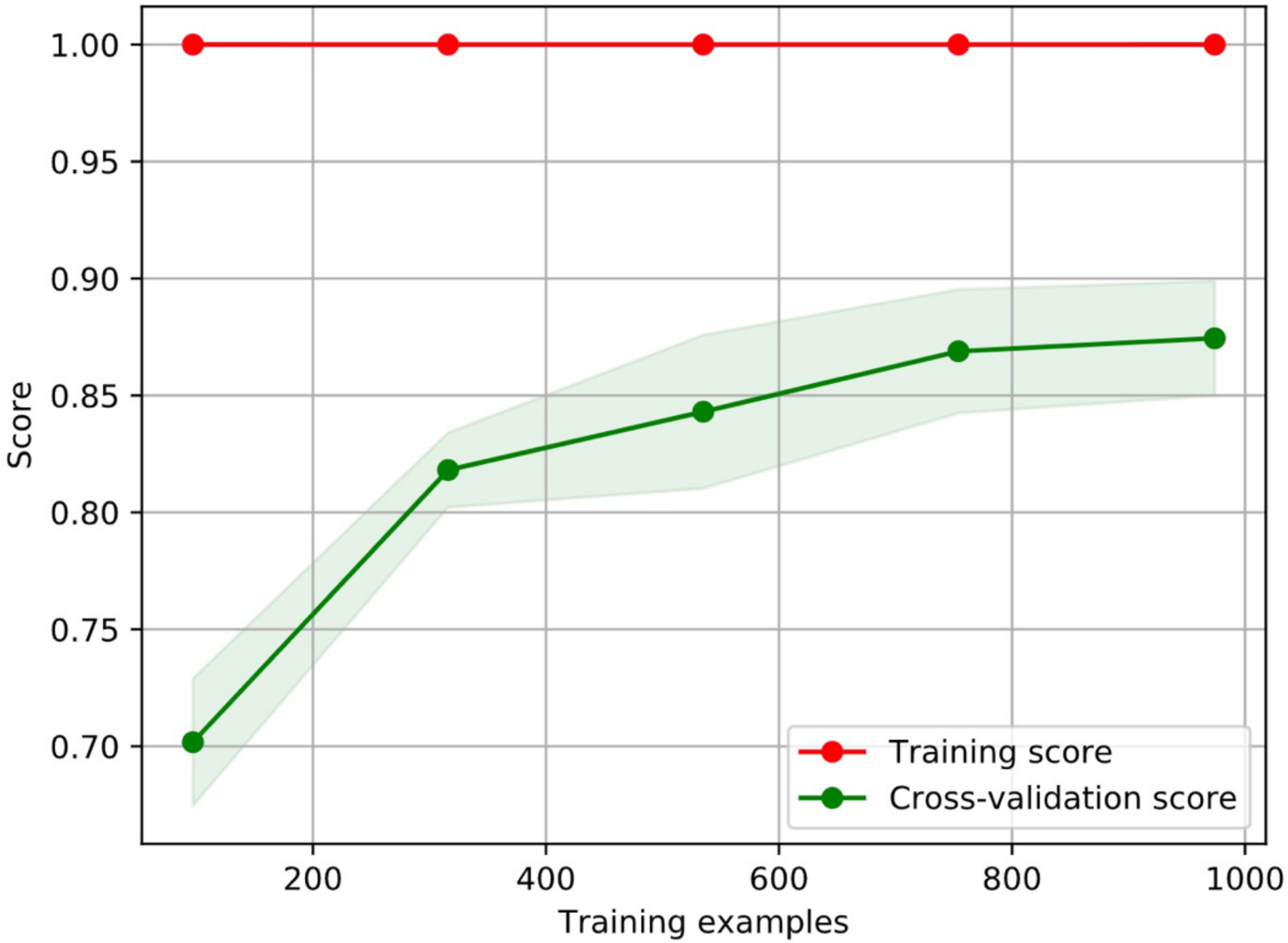
THE LIKABILITY CLASSIFIER: DECISION TREE LEARNING CURVE



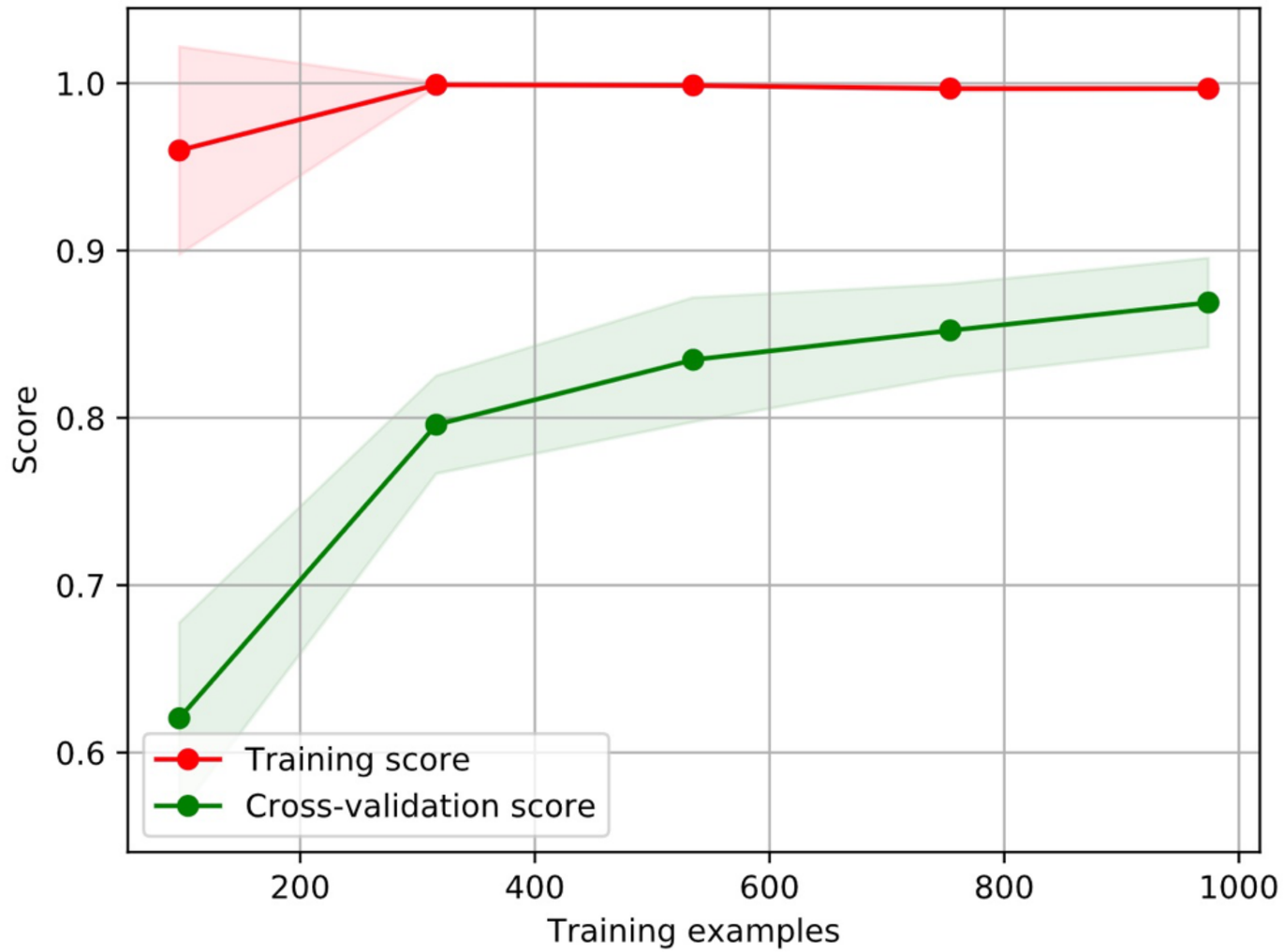
THE LIKABILITY CLASSIFIER: MULTINOMIAL BAYES LEARNING CURVE



THE LIKABILITY CLASSIFIER: LINEAR SVC LEARNING CURVE



THE LIKABILITY CLASSIFIER: LOGISTIC REGRESSION LEARNING CURVE



THE LIKABILITY CLASSIFIER: ENSEMBLES AND METACLASSIFIERS

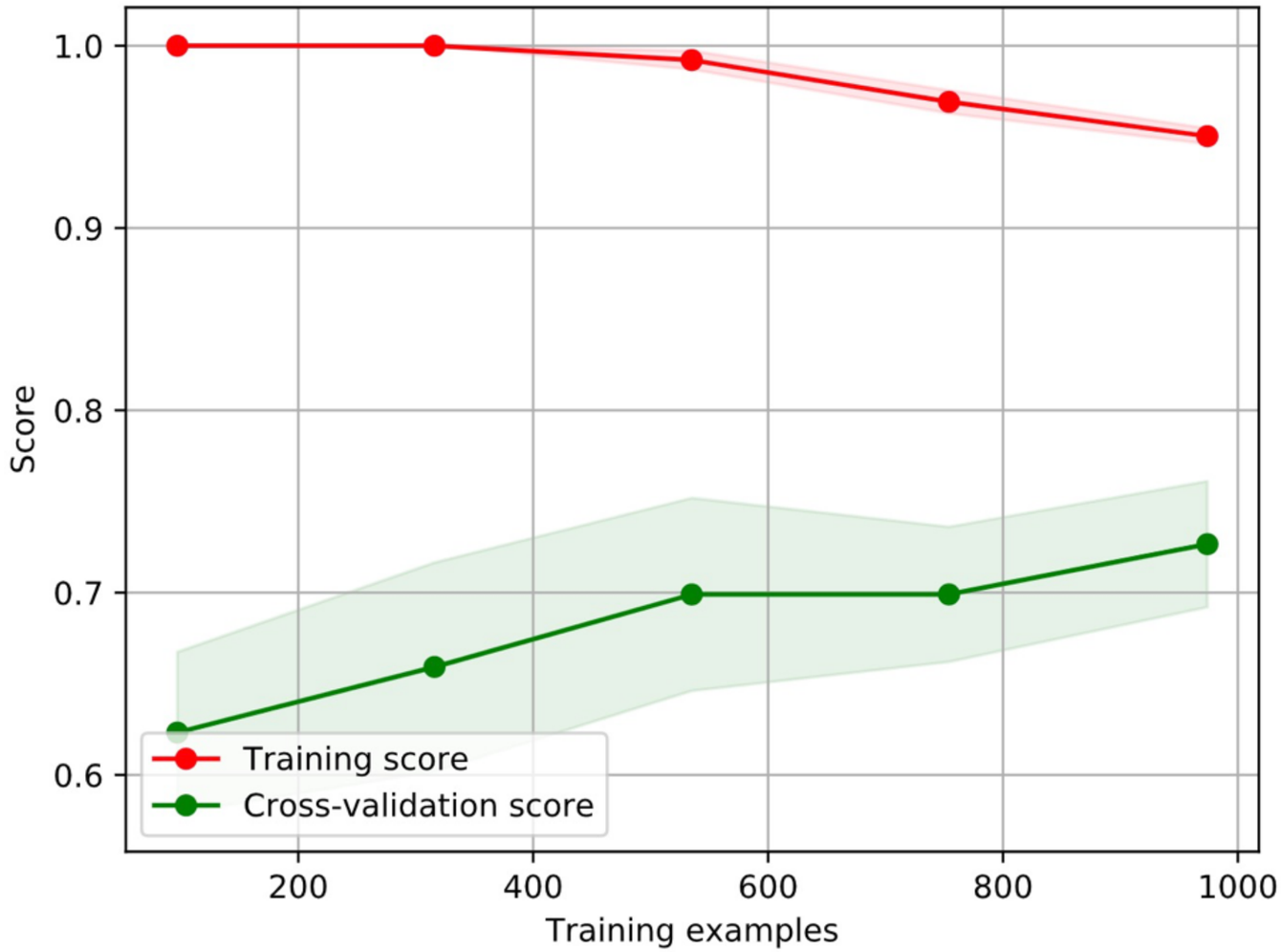
Here are reported the **scores** and **performance** of some **ensemble classifiers** trying to predict the **category** of an article:

CLASSIFIER	F1-SCORE		ACCURACY		PRECISION		RECALL	
	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV
D. Tree + ADABoost	0.7358	0.0324	0.7369	0.0322	0.7397	0.0326	0.7369	0.0322
Random Forest	0.7976	0.0312	0.7996	0.0302	0.8087	0.0271	0.7996	0.0302
Voting	0.8714	0.0339	0.8717	0.0338	0.8739	0.0341	0.8717	0.0338
Bagging	0.8669	0.0286	0.8670	0.0286	0.8690	0.0287	0.8670	0.0286
XGBClassifier	0.7234	0.0410	0.7267	0.0402	0.7356	0.423	0.7267	0.0402

Best scor
but not
better than
LogReg

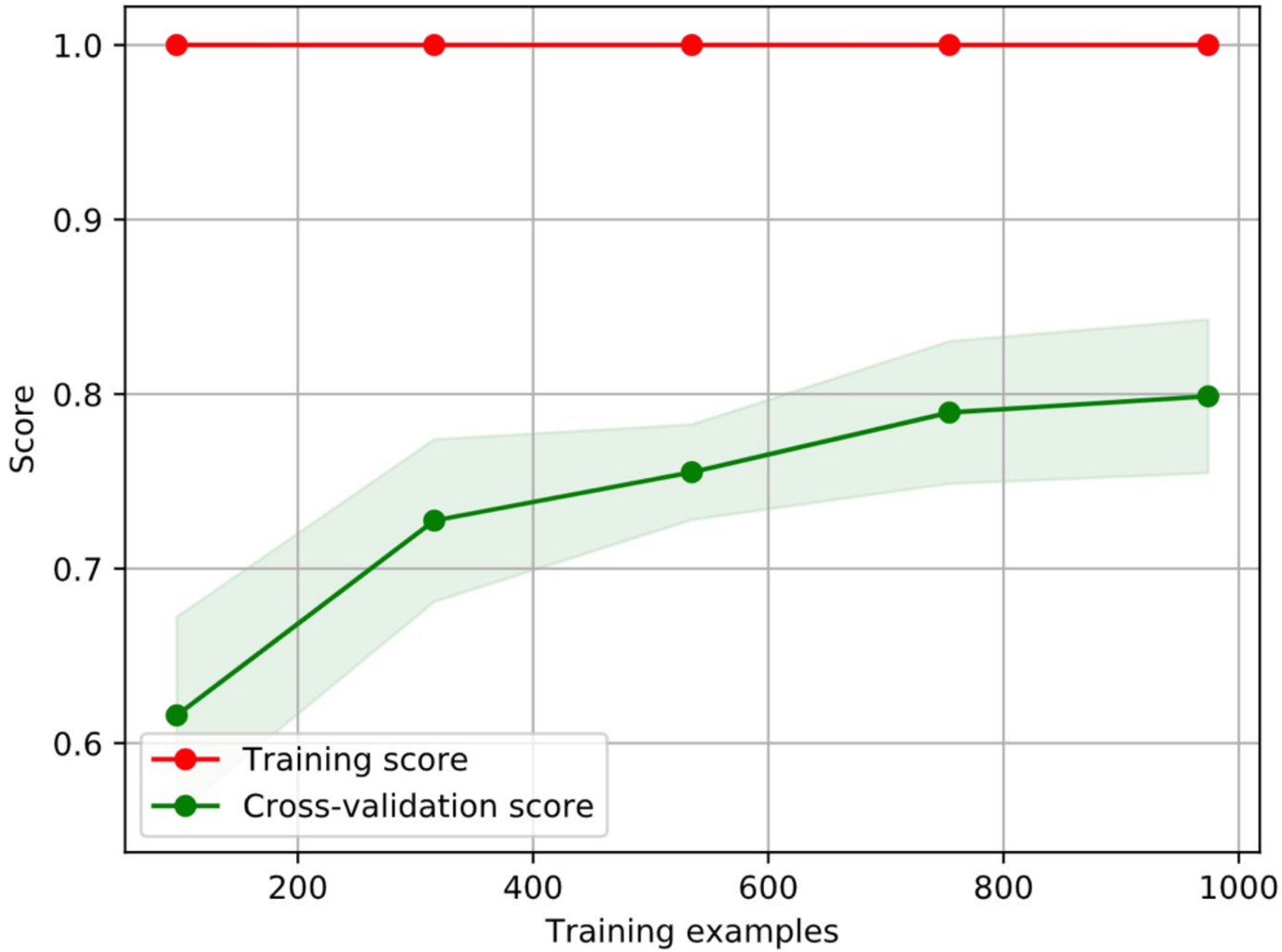
The following classifiers were **tuned** using a GridSearch algorithm! The testing and validation were performed using **10 Folds Cross Validation**

THE LIKABILITY CLASSIFIER: ADABOOST LEARNING CURVE



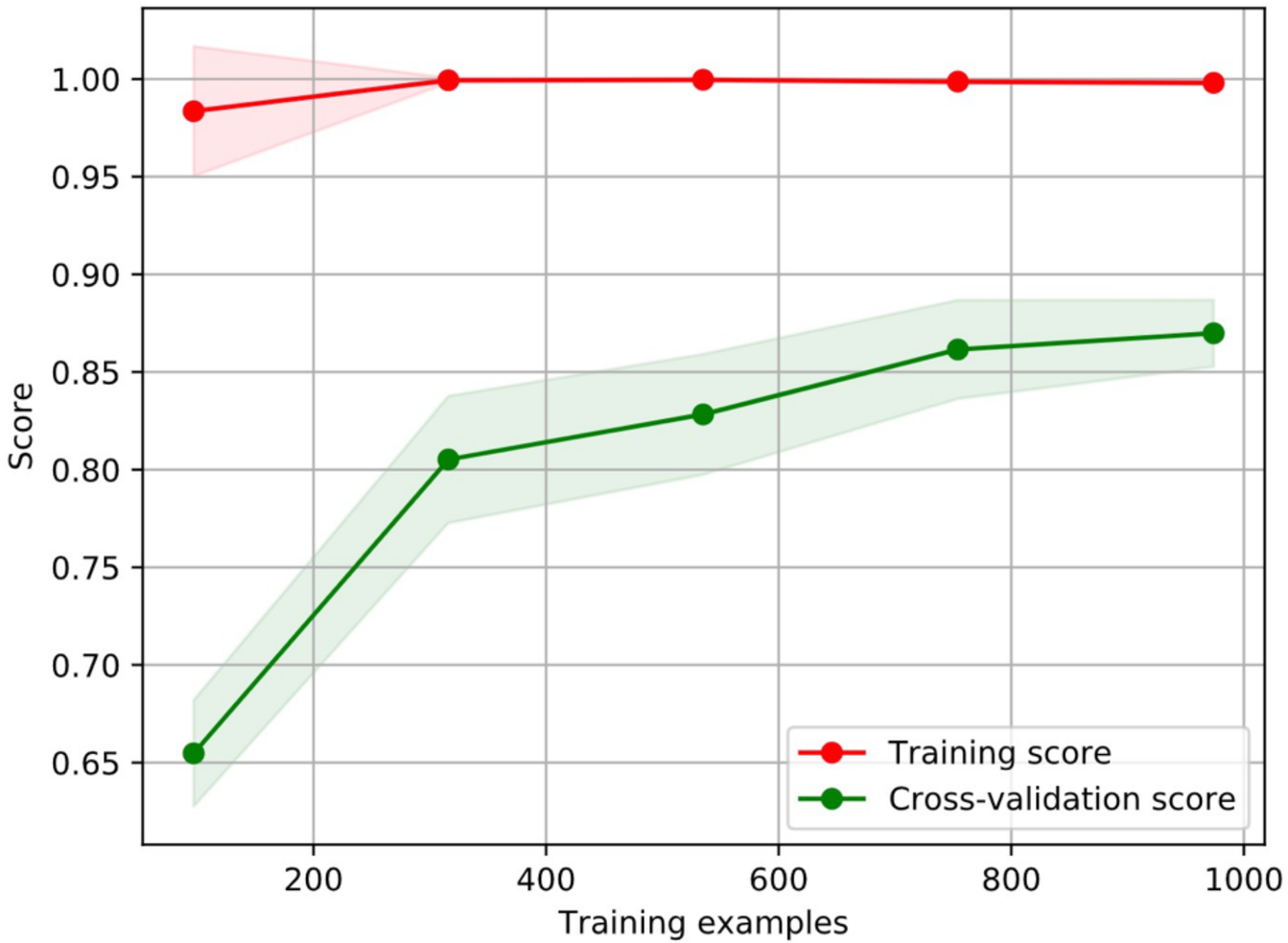

```
parameters {
  'base': DecisionTree(),
  'n_estimators': 100
}
```

THE LIKABILITY CLASSIFIER: RANDOM FOREST LEARNING CURVE

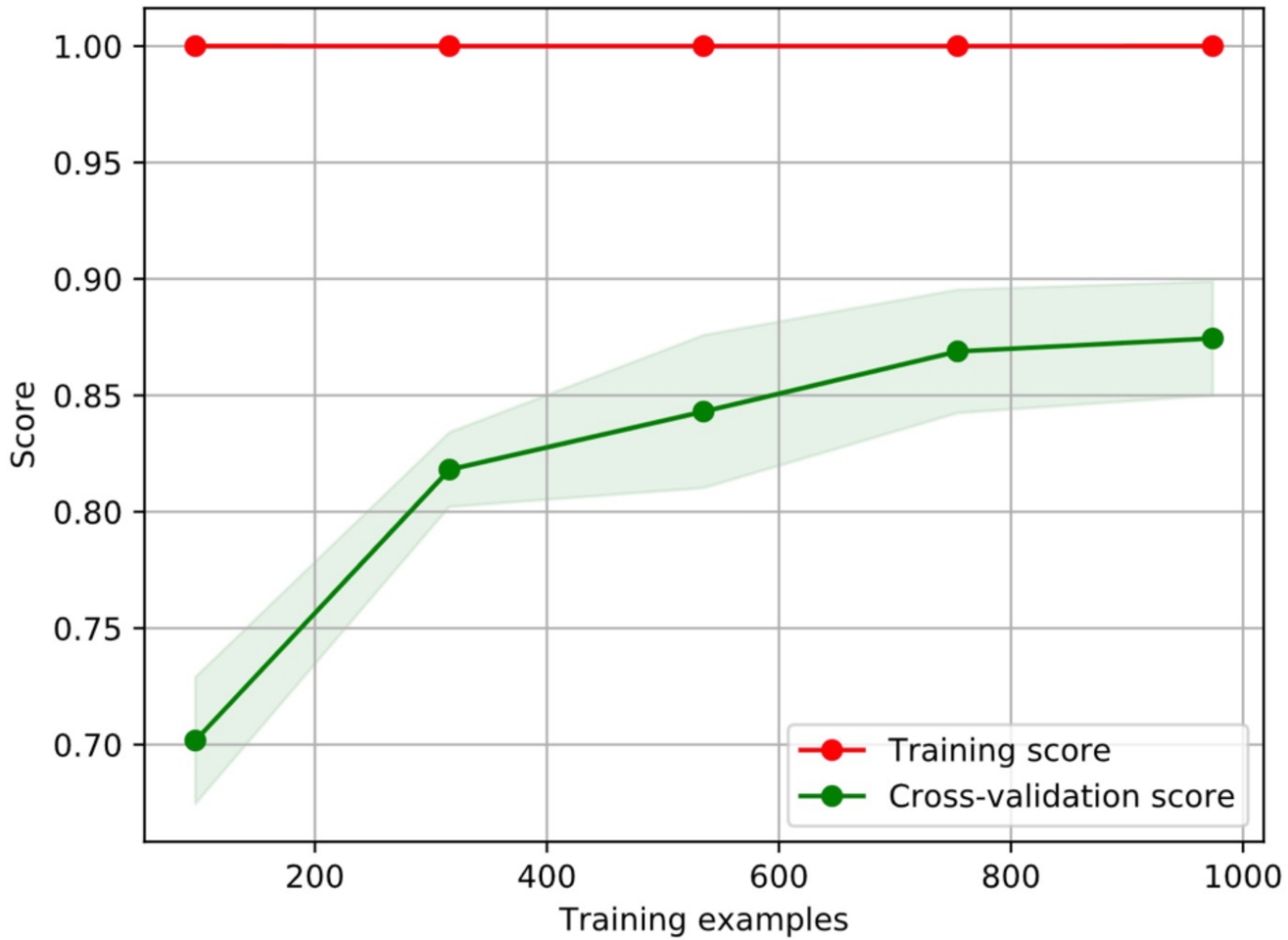



```
parameters {
  'n_estimators': 100
}
```

THE LIKABILITY CLASSIFIER: VOTING LEARNING CURVE

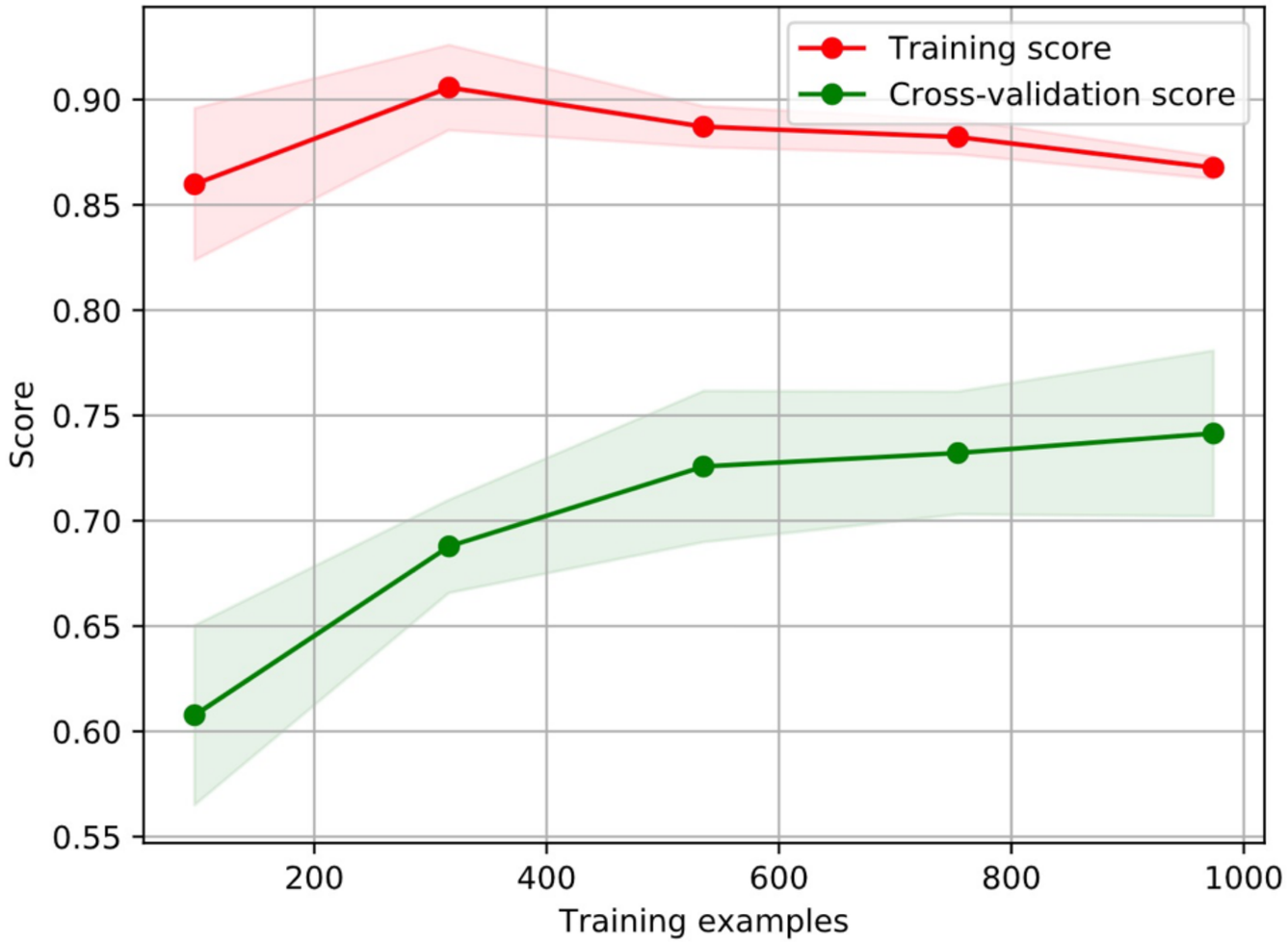


THE LIKABILITY CLASSIFIER: BAGGING LEARNING CURVE




```
parameters {
  'base': LinearSVC(),
  'n_estimators': 100
}
```

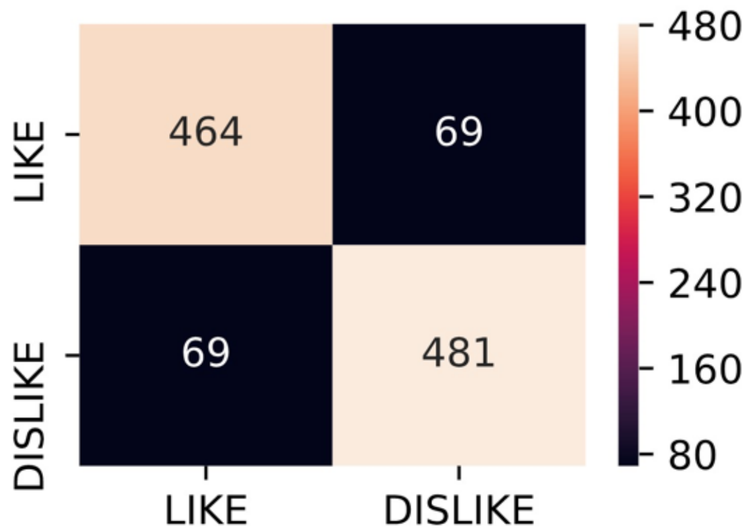
THE LIKABILITY CLASSIFIER: XGB LEARNING CURVE




```
parameters {
  'n_estimators': 100,
  'max_depth': 3,
  'learning_rate': 1
}
```

THE CATEGORY CLASSIFIER

In the end, no classifier did better than the base **LogisticRegression**, so it was chosen as the classifier that should predict the article likability. Here is shown its **confusion matrix**:



Please notice that this result was achieved under the assumption that the category of the article was correctly predicted !

THE WEB APPLICATION: A SCREENSHOT OF THE NEWSFEED