SMS Spam Detection

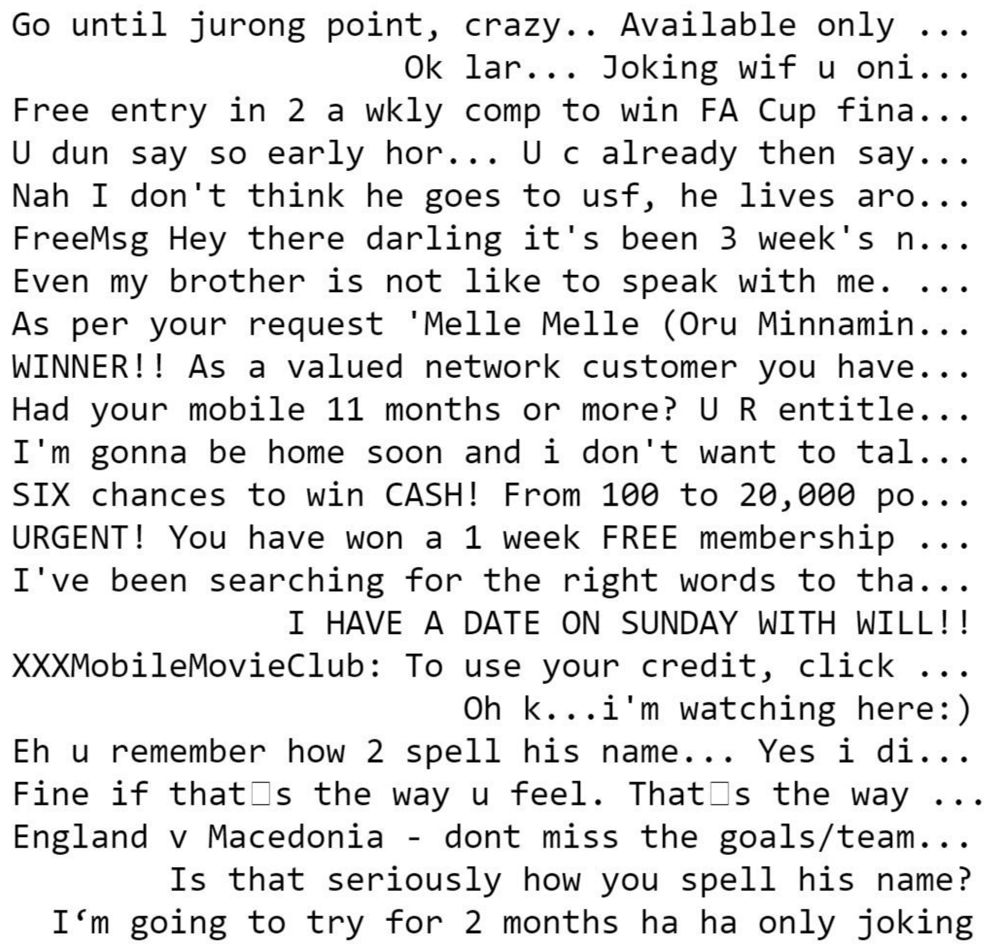
Karolina Bzdusek

Capstone Project for Intensive Data Science Career Track, May 20th 2019

Springboard

Introduction

Source of data: http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/



```
Go until jurong point, crazy.. Available only ...
                    Ok lar... Joking wif u oni...
Free entry in 2 a wkly comp to win FA Cup fina...
U dun say so early hor... U c already then say...
Nah I don't think he goes to usf, he lives aro...
FreeMsg Hey there darling it's been 3 week's n...
Even my brother is not like to speak with me. ...
As per your request 'Melle Melle (Oru Minnamin...
WINNER!! As a valued network customer you have...
Had your mobile 11 months or more? U R entitle...
I'm gonna be home soon and i don't want to tal...
SIX chances to win CASH! From 100 to 20,000 po...
URGENT! You have won a 1 week FREE membership ...
I've been searching for the right words to tha...
              I HAVE A DATE ON SUNDAY WITH WILL!!
XXXMobileMovieClub: To use your credit, click ...
                       Oh k...i'm watching here:)
Eh u remember how 2 spell his name... Yes i di...
Fine if that□s the way u feel. That□s the way ...
England v Macedonia - dont miss the goals/team...
        Is that seriously how you spell his name?
  I‘m going to try for 2 months ha ha only joking
```

5574 SMS Messages

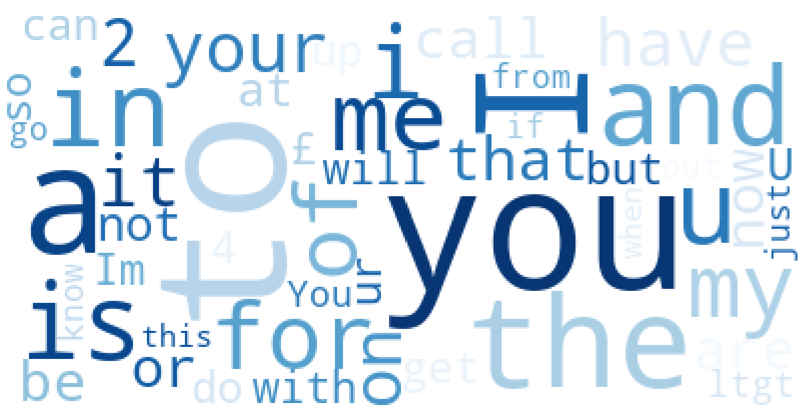
# Ham
4827

# Spam
747

Binary classification problem with highly imbalanced dataset

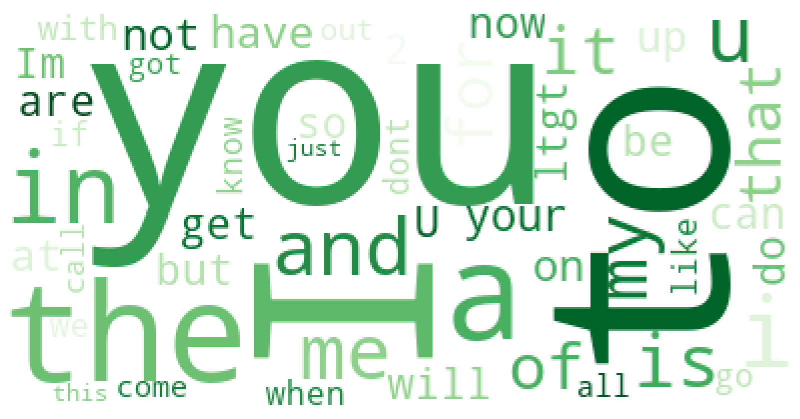
Karolina Bzdusek,
20th May 2019

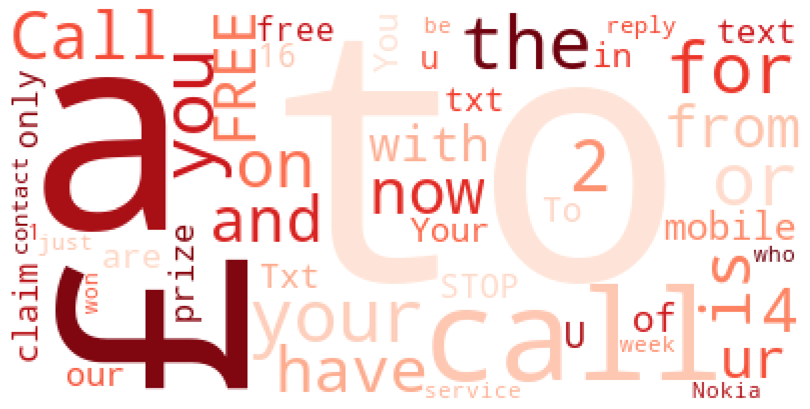
How to recognize spam messages?



Ham

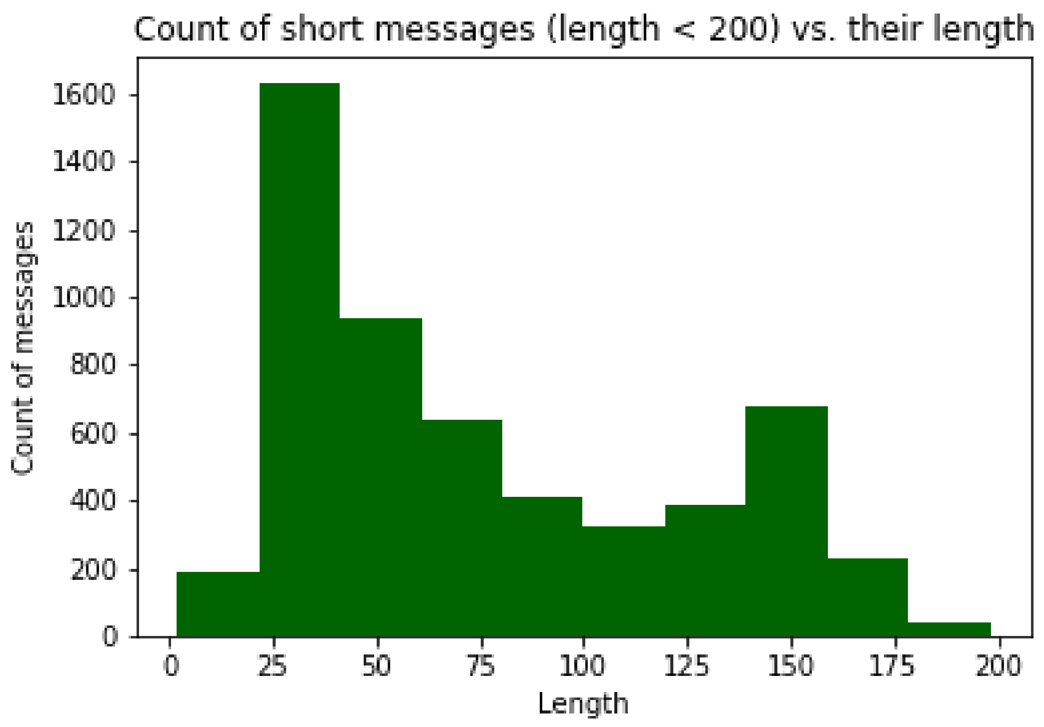
Spam

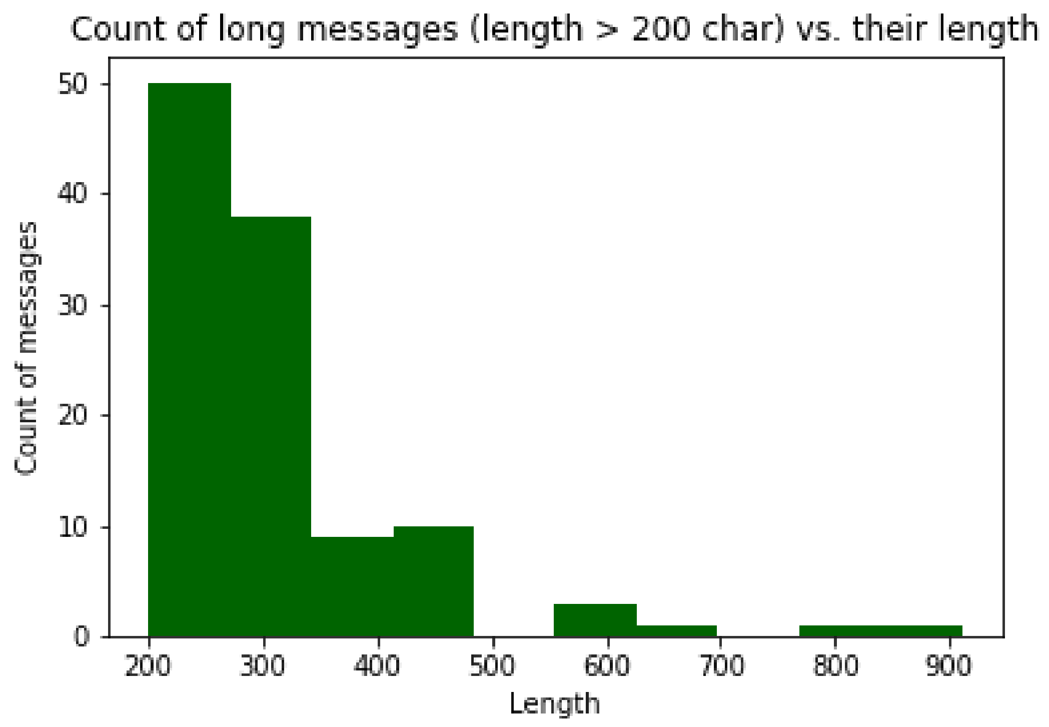




Karolina Bzdusek,
20th May 2019

How long spam/ham messages are?





‘length’ = 122 … 75%
Spam ratio … 19%
Ham ratio …. 84%

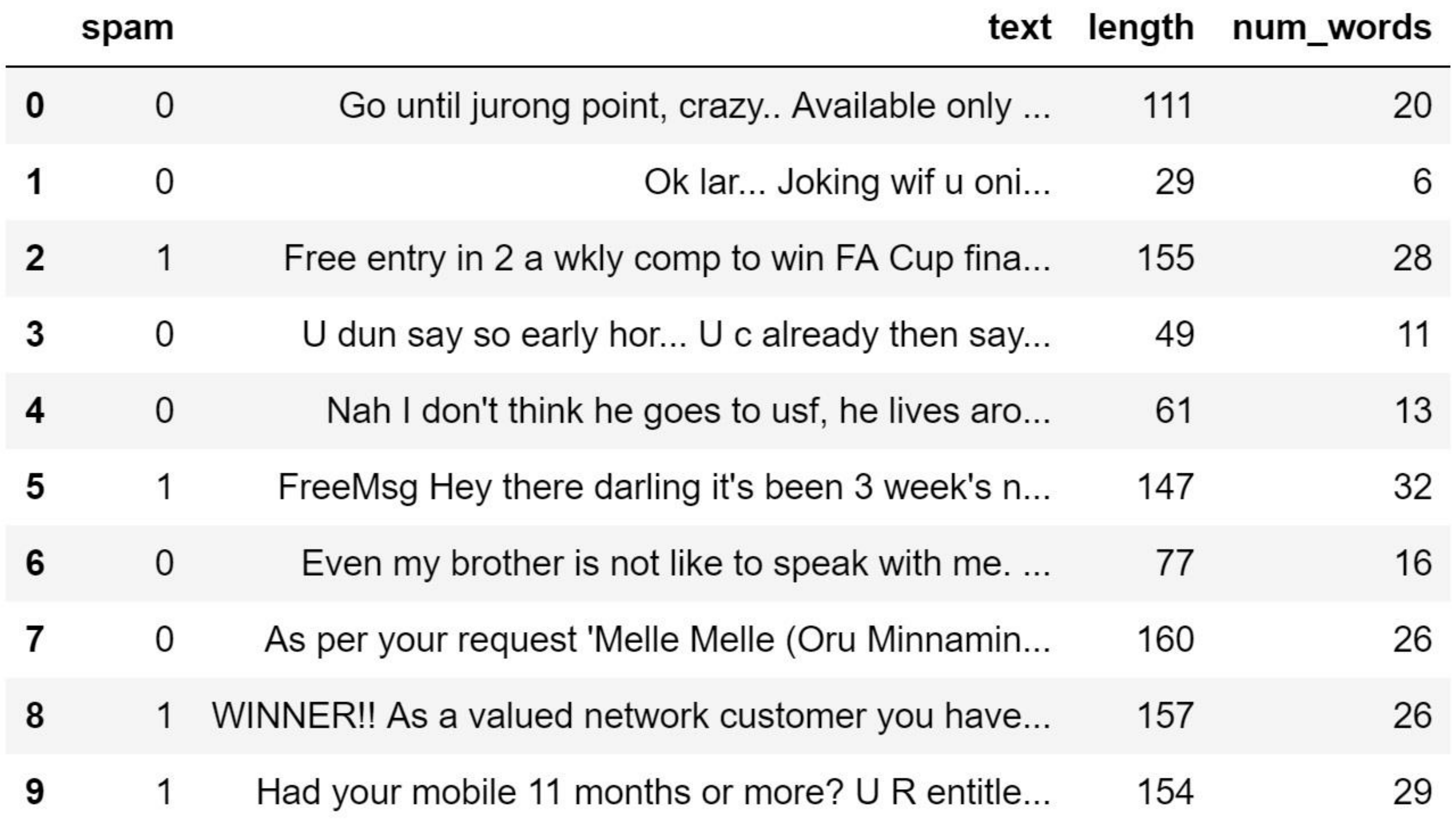
'Ok', 'Yup', '645', 'Ok.', ':) ', 'Ok..', 'Okie’, 'U 2.', 'Ok...', 'G.W.R', 'Y lei?', 'Yup...’, 'ALRITE', 'Okie...', 'Where @', 'Oh ok..', 'Ok lor.', 'Nite...', 'Havent.', ':-) :-)', 'Thanx...', 'Thank u!', 'Beerage?', 'U too...', 'My phone', "I'm home.", 'Yup ok...', 'How come?'

‘length’ = 300
→ only 41 messages, all of them are ham

Springboard

Karolina Bzdusek,
20th May 2019

Text and Numerical Features + Preprocessing



	spam	text	length	num_words
0	0	Go until jurong point, crazy.. Available only ...	111	20
1	0	Ok lar... Joking wif u oni...	29	6
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	28
3	0	U dun say so early hor... U c already then say...	49	11
4	0	Nah I don't think he goes to usf, he lives aro...	61	13
5	1	FreeMsg Hey there darling it's been 3 week's n...	147	32
6	0	Even my brother is not like to speak with me. ...	77	16
7	0	As per your request 'Melle Melle (Oru Minnamin...	160	26
8	1	WINNER!! As a valued network customer you have...	157	26
9	1	Had your mobile 11 months or more? U R entitle...	154	29

n-grams

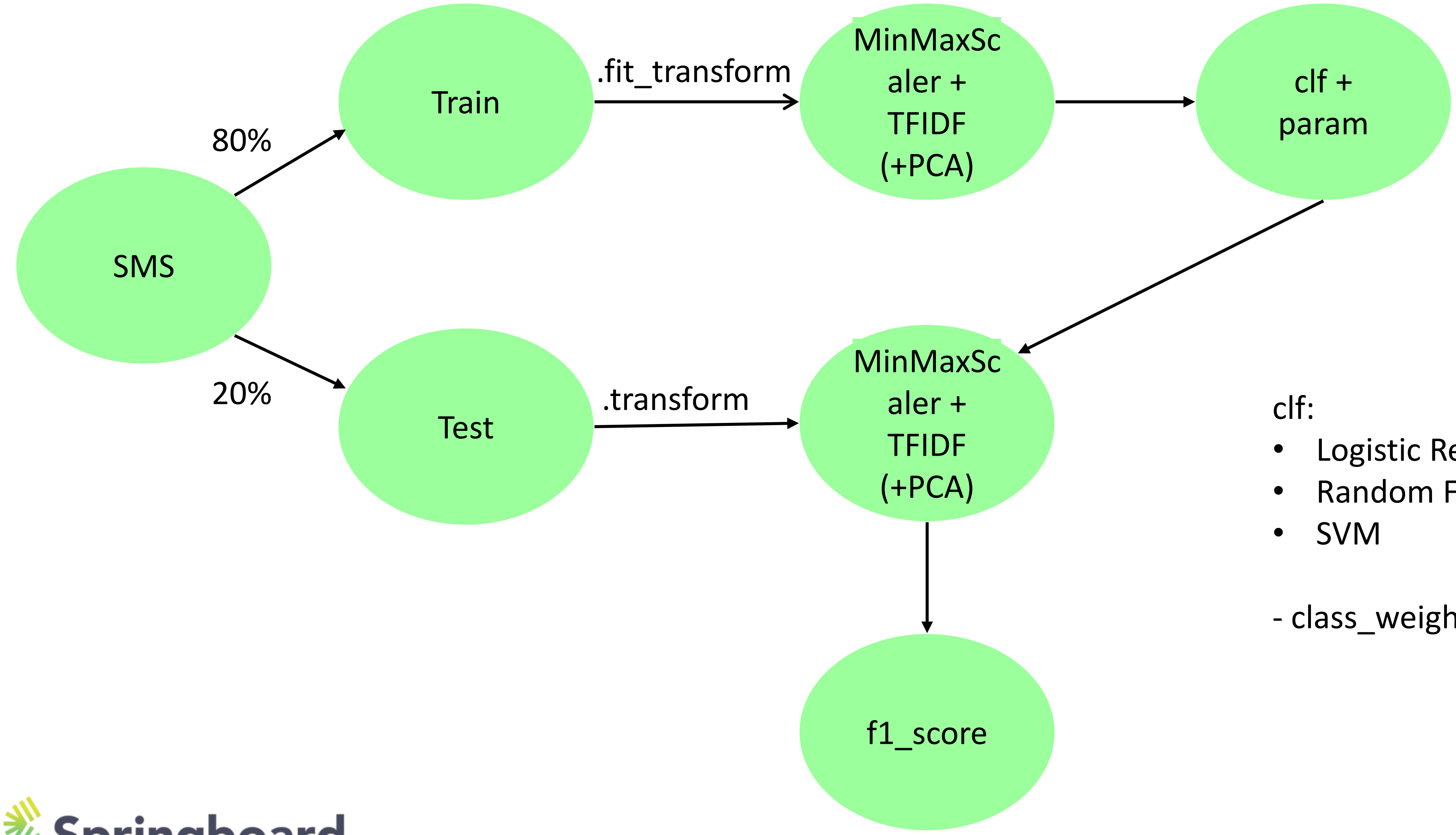
Stop words, lemmatization, …

Token = words
11662 tokens – typical task in NLP, more features than data
TweetTokenizer()
TFIDF

Springboard

Karolina Bzdusek,
20th May 2019

Modelling



clf:

- Logistic Regression
- Random Forest
- SVM

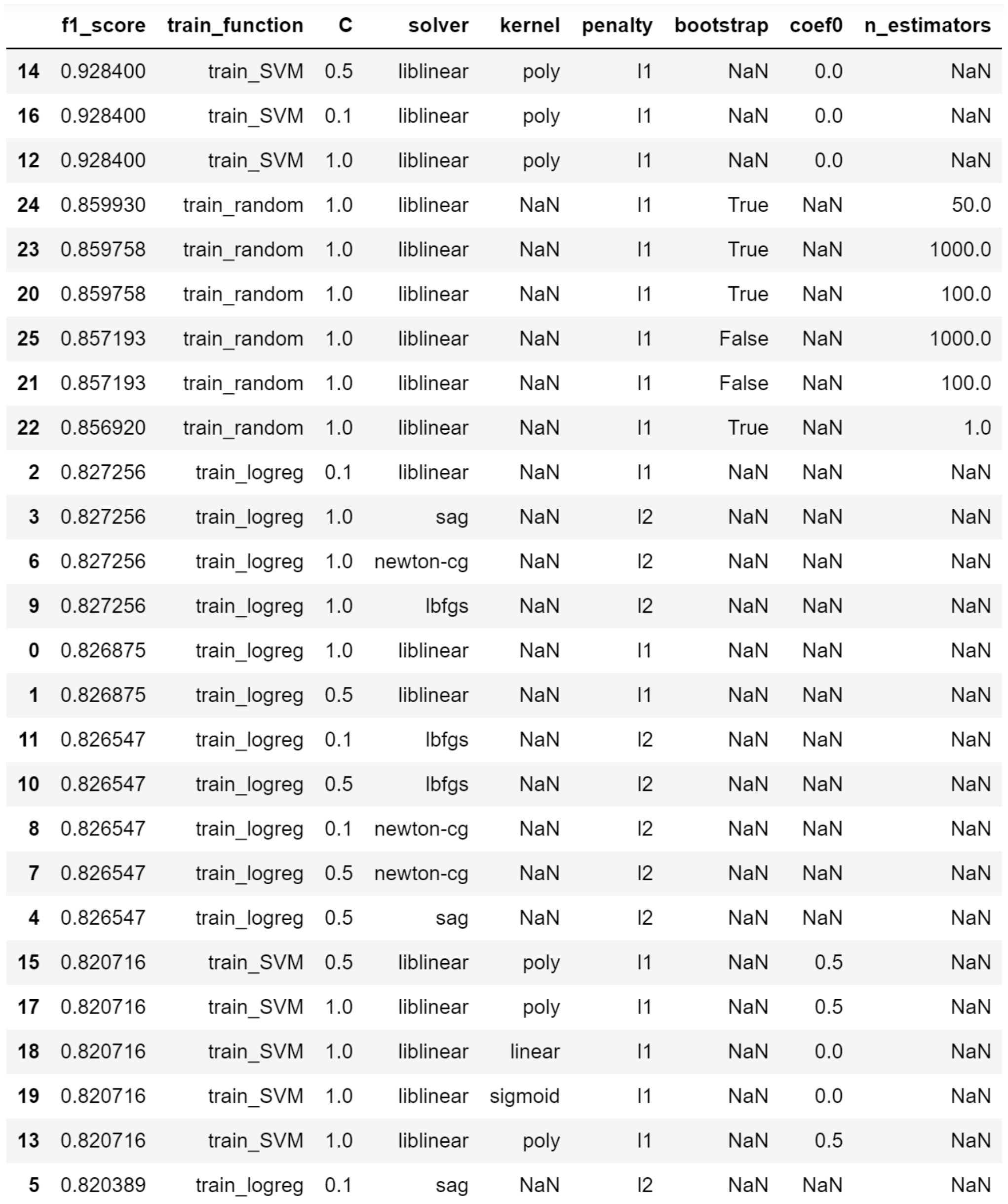
- class_weight = ‘imbalanced’

Springboard

Karolina Bzdusek,
20th May 2019

Results I.

'length' model



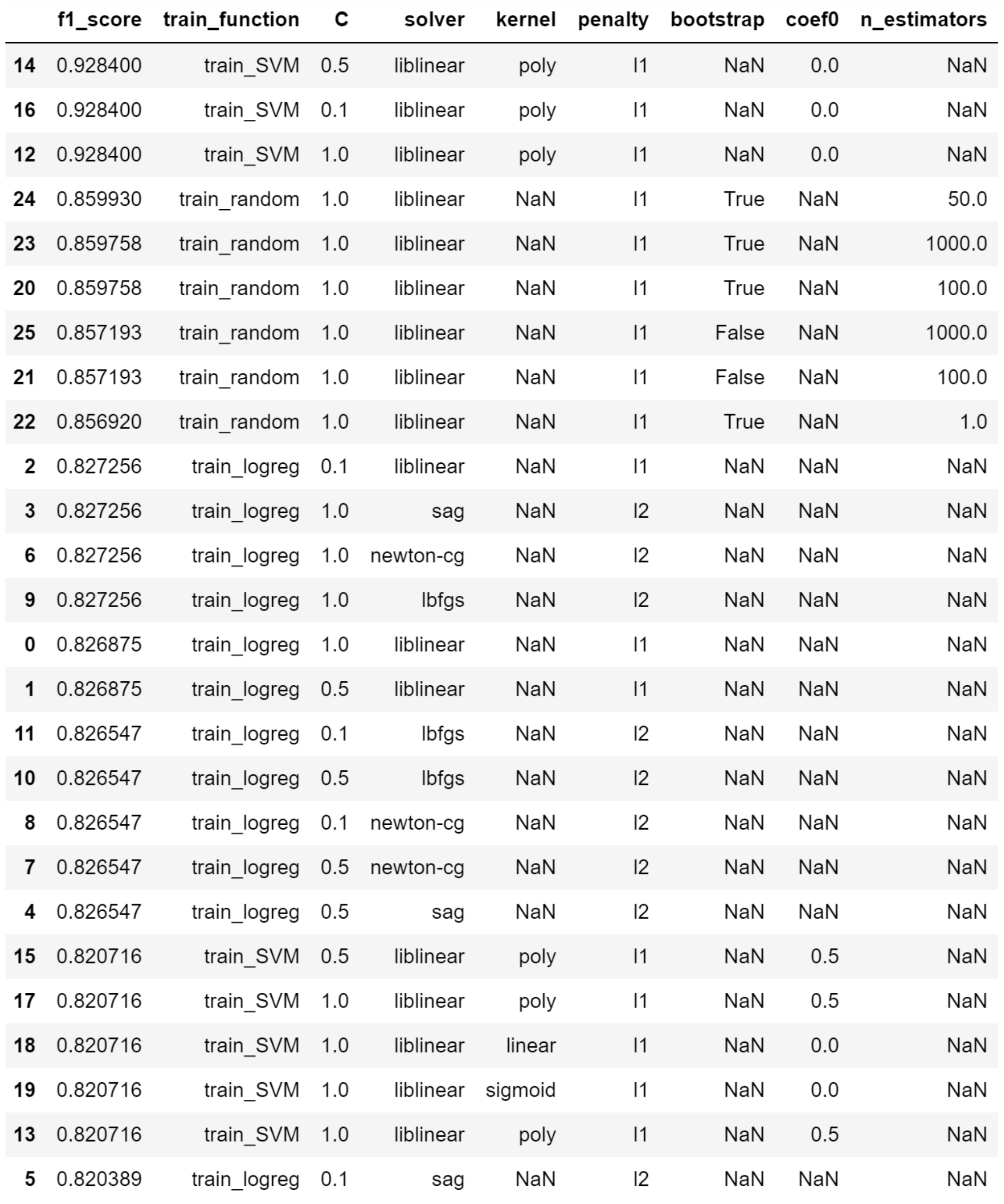
	f1_score	train_function	C	solver	kernel	penalty	bootstrap	coef0	n_estimators
14	0.928400	train_SVM	0.5	liblinear	poly	l1	NaN	0.0	NaN
16	0.928400	train_SVM	0.1	liblinear	poly	l1	NaN	0.0	NaN
12	0.928400	train_SVM	1.0	liblinear	poly	l1	NaN	0.0	NaN
24	0.859930	train_random	1.0	liblinear	NaN	l1	True	NaN	50.0
23	0.859758	train_random	1.0	liblinear	NaN	l1	True	NaN	1000.0
20	0.859758	train_random	1.0	liblinear	NaN	l1	True	NaN	100.0
25	0.857193	train_random	1.0	liblinear	NaN	l1	False	NaN	1000.0
21	0.857193	train_random	1.0	liblinear	NaN	l1	False	NaN	100.0
22	0.856920	train_random	1.0	liblinear	NaN	l1	True	NaN	1.0
2	0.827256	train_logreg	0.1	liblinear	NaN	l1	NaN	NaN	NaN
3	0.827256	train_logreg	1.0	sag	NaN	l2	NaN	NaN	NaN
6	0.827256	train_logreg	1.0	newton-cg	NaN	l2	NaN	NaN	NaN
9	0.827256	train_logreg	1.0	lbfgs	NaN	l2	NaN	NaN	NaN
0	0.826875	train_logreg	1.0	liblinear	NaN	l1	NaN	NaN	NaN
1	0.826875	train_logreg	0.5	liblinear	NaN	l1	NaN	NaN	NaN
11	0.826547	train_logreg	0.1	lbfgs	NaN	l2	NaN	NaN	NaN
10	0.826547	train_logreg	0.5	lbfgs	NaN	l2	NaN	NaN	NaN
8	0.826547	train_logreg	0.1	newton-cg	NaN	l2	NaN	NaN	NaN
7	0.826547	train_logreg	0.5	newton-cg	NaN	l2	NaN	NaN	NaN
4	0.826547	train_logreg	0.5	sag	NaN	l2	NaN	NaN	NaN
15	0.820716	train_SVM	0.5	liblinear	poly	l1	NaN	0.5	NaN
17	0.820716	train_SVM	1.0	liblinear	poly	l1	NaN	0.5	NaN
18	0.820716	train_SVM	1.0	liblinear	linear	l1	NaN	0.0	NaN
19	0.820716	train_SVM	1.0	liblinear	sigmoid	l1	NaN	0.0	NaN
13	0.820716	train_SVM	1.0	liblinear	poly	l1	NaN	0.5	NaN
5	0.820389	train_logreg	0.1	sag	NaN	l2	NaN	NaN	NaN

Springboard

Karolina Bzdusek,
20th May 2019

Results II.

'length' + 'num_words' model



	f1_score	train_function	C	solver	kernel	penalty	bootstrap	coef0	n_estimators
14	0.928400	train_SVM	0.5	liblinear	poly	l1	NaN	0.0	NaN
16	0.928400	train_SVM	0.1	liblinear	poly	l1	NaN	0.0	NaN
12	0.928400	train_SVM	1.0	liblinear	poly	l1	NaN	0.0	NaN
24	0.859930	train_random	1.0	liblinear	NaN	l1	True	NaN	50.0
23	0.859758	train_random	1.0	liblinear	NaN	l1	True	NaN	1000.0
20	0.859758	train_random	1.0	liblinear	NaN	l1	True	NaN	100.0
25	0.857193	train_random	1.0	liblinear	NaN	l1	False	NaN	1000.0
21	0.857193	train_random	1.0	liblinear	NaN	l1	False	NaN	100.0
22	0.856920	train_random	1.0	liblinear	NaN	l1	True	NaN	1.0
2	0.827256	train_logreg	0.1	liblinear	NaN	l1	NaN	NaN	NaN
3	0.827256	train_logreg	1.0	sag	NaN	l2	NaN	NaN	NaN
6	0.827256	train_logreg	1.0	newton-cg	NaN	l2	NaN	NaN	NaN
9	0.827256	train_logreg	1.0	lbfgs	NaN	l2	NaN	NaN	NaN
0	0.826875	train_logreg	1.0	liblinear	NaN	l1	NaN	NaN	NaN
1	0.826875	train_logreg	0.5	liblinear	NaN	l1	NaN	NaN	NaN
11	0.826547	train_logreg	0.1	lbfgs	NaN	l2	NaN	NaN	NaN
10	0.826547	train_logreg	0.5	lbfgs	NaN	l2	NaN	NaN	NaN
8	0.826547	train_logreg	0.1	newton-cg	NaN	l2	NaN	NaN	NaN
7	0.826547	train_logreg	0.5	newton-cg	NaN	l2	NaN	NaN	NaN
4	0.826547	train_logreg	0.5	sag	NaN	l2	NaN	NaN	NaN
15	0.820716	train_SVM	0.5	liblinear	poly	l1	NaN	0.5	NaN
17	0.820716	train_SVM	1.0	liblinear	poly	l1	NaN	0.5	NaN
18	0.820716	train_SVM	1.0	liblinear	linear	l1	NaN	0.0	NaN
19	0.820716	train_SVM	1.0	liblinear	sigmoid	l1	NaN	0.0	NaN
13	0.820716	train_SVM	1.0	liblinear	poly	l1	NaN	0.5	NaN
5	0.820389	train_logreg	0.1	sag	NaN	l2	NaN	NaN	NaN

Springboard

Karolina Bzdusek,
20th May 2019

‘length’ + ‘num_words’ + ‘text’ model

Results III.

	f1_score	train_function	C	solver	penalty
1	0.878984	train_logreg	0.5	liblinear	l1
0	0.874926	train_logreg	1.0	liblinear	l1
13	0.858637	train_random	1.0	liblinear	l1
21	0.850227	train_random	1.0	liblinear	l1
14	0.847529	train_random	1.0	liblinear	l1
18	0.842359	train_random	1.0	liblinear	l1
12	0.842359	train_random	1.0	liblinear	l1
20	0.841724	train_random	1.0	liblinear	l1
15	0.841724	train_random	1.0	liblinear	l1
19	0.841340	train_random	1.0	liblinear	l1
17	0.840661	train_random	1.0	liblinear	l1
16	0.840296	train_random	1.0	liblinear	l1
2	0.827256	train_logreg	0.1	liblinear	l1
6	0.816717	train_logreg	1.0	newton-cg	l2
9	0.816717	train_logreg	1.0	lbfgs	l2
3	0.816717	train_logreg	1.0	sag	l2
4	0.816010	train_logreg	0.5	sag	l2
10	0.816010	train_logreg	0.5	lbfgs	l2
7	0.816010	train_logreg	0.5	newton-cg	l2
5	0.815667	train_logreg	0.1	sag	l2
8	0.815667	train_logreg	0.1	newton-cg	l2
11	0.815667	train_logreg	0.1	lbfgs	l2

	f1_score	train_function	C	solver	penalty
1	0.879706	train_logreg	0.5	liblinear	l1
0	0.879064	train_logreg	1.0	liblinear	l1
17	0.838605	train_random	1.0	liblinear	l1
9	0.831677	train_logreg	1.0	lbfgs	l2
3	0.831677	train_logreg	1.0	sag	l2
6	0.831677	train_logreg	1.0	newton-cg	l2
2	0.827256	train_logreg	0.1	liblinear	l1
21	0.825825	train_random	1.0	liblinear	l1
7	0.822808	train_logreg	0.5	newton-cg	l2
10	0.822808	train_logreg	0.5	lbfgs	l2
4	0.822808	train_logreg	0.5	sag	l2
13	0.821970	train_random	1.0	liblinear	l1
14	0.818401	train_random	1.0	liblinear	l1
16	0.818198	train_random	1.0	liblinear	l1
19	0.818198	train_random	1.0	liblinear	l1
15	0.817342	train_random	1.0	liblinear	l1
20	0.817342	train_random	1.0	liblinear	l1
8	0.814011	train_logreg	0.1	newton-cg	l2
5	0.814011	train_logreg	0.1	sag	l2
11	0.814011	train_logreg	0.1	lbfgs	l2
12	0.806900	train_random	1.0	liblinear	l1
18	0.806900	train_random	1.0	liblinear	l1

	f1_score	train_function	C	solver	penalty
0	0.881229	train_logreg	1.0	liblinear	l1
1	0.879706	train_logreg	0.5	liblinear	l1
21	0.856935	train_random	1.0	liblinear	l1
14	0.844948	train_random	1.0	liblinear	l1
13	0.841871	train_random	1.0	liblinear	l1
20	0.835431	train_random	1.0	liblinear	l1
15	0.835431	train_random	1.0	liblinear	l1
3	0.834395	train_logreg	1.0	sag	l2
6	0.834395	train_logreg	1.0	newton-cg	l2
9	0.834395	train_logreg	1.0	lbfgs	l2
16	0.828060	train_random	1.0	liblinear	l1
7	0.828009	train_logreg	0.5	newton-cg	l2
10	0.828009	train_logreg	0.5	lbfgs	l2
4	0.828009	train_logreg	0.5	sag	l2
2	0.827256	train_logreg	0.1	liblinear	l1
19	0.824058	train_random	1.0	liblinear	l1
11	0.821087	train_logreg	0.1	lbfgs	l2
8	0.821087	train_logreg	0.1	newton-cg	l2
5	0.821087	train_logreg	0.1	sag	l2
18	0.806262	train_random	1.0	liblinear	l1
12	0.806262	train_random	1.0	liblinear	l1
17	0.797686	train_random	1.0	liblinear	l1

PCA = {100, 500, 1000}

Karolina Bzdusek,
20th May 2019

‘length’ + ‘num_words’ + ‘text’ model

Results III.

	train_function	f1_score	precision	recall	kernel	C	coef0
0	train_SVM	0.798019	0.785805	0.812556	poly	1.0	0.0
1	train_SVM	0.798019	0.785805	0.812556	poly	1.0	0.5
2	train_SVM	0.800303	0.749651	0.858296	poly	0.5	0.0
3	train_SVM	0.800303	0.749651	0.858296	poly	0.5	0.5
4	train_SVM	0.928400	0.866368	1.000000	poly	0.1	0.0
5	train_SVM	0.798019	0.785805	0.812556	poly	1.0	0.5
6	train_SVM	0.818254	0.899546	0.787444	linear	1.0	0.0
7	train_SVM	0.799853	0.749545	0.857399	sigmoid	1.0	0.0

	train_function	f1_score	precision	recall	C	coef0
0	train_SVM	0.798019	0.785805	0.812556	1.0	0.0
1	train_SVM	0.798019	0.785805	0.812556	1.0	0.5
2	train_SVM	0.800303	0.749651	0.858296	0.5	0.0
3	train_SVM	0.800303	0.749651	0.858296	0.5	0.5
4	train_SVM	0.928400	0.866368	1.000000	0.1	0.0
5	train_SVM	0.798019	0.785805	0.812556	1.0	0.5
6	train_SVM	0.818254	0.899546	0.787444	1.0	0.0
7	train_SVM	0.799853	0.749545	0.857399	1.0	0.0

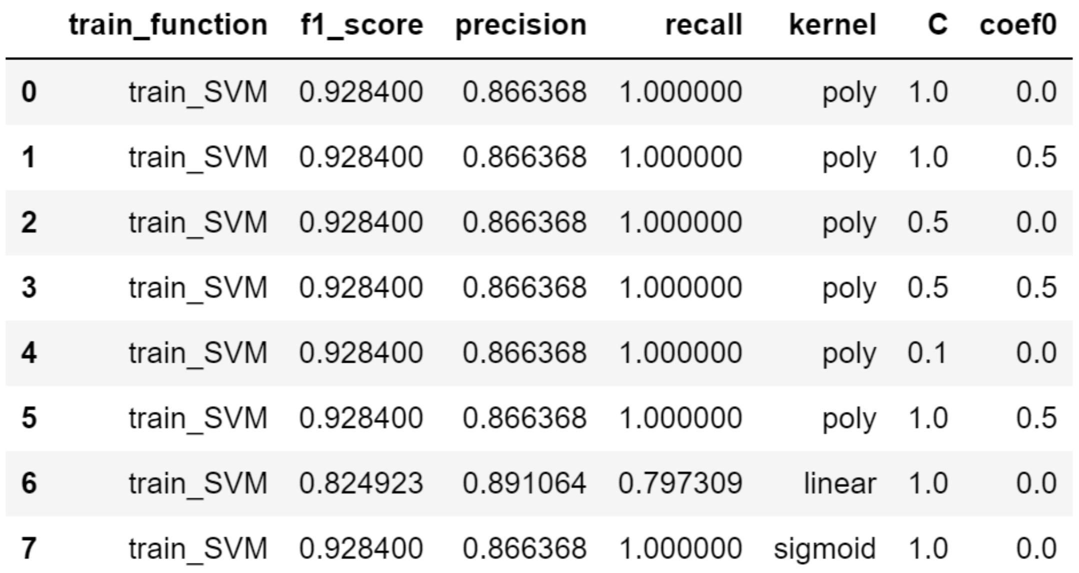
Springboard

PCA = {0, 100}

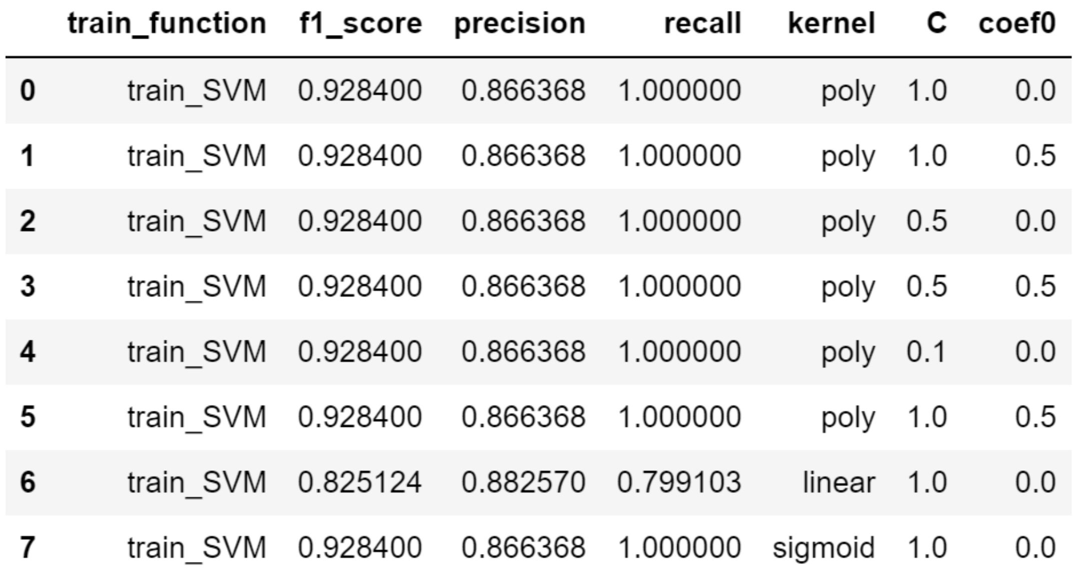
Karolina Bzdusek,
20th May 2019

‘length’ + ‘num_words’ + ‘text’ model

Results III.



	train_function	f1_score	precision	recall	kernel	C	coef0
0	train_SVM	0.928400	0.866368	1.000000	poly	1.0	0.0
1	train_SVM	0.928400	0.866368	1.000000	poly	1.0	0.5
2	train_SVM	0.928400	0.866368	1.000000	poly	0.5	0.0
3	train_SVM	0.928400	0.866368	1.000000	poly	0.5	0.5
4	train_SVM	0.928400	0.866368	1.000000	poly	0.1	0.0
5	train_SVM	0.928400	0.866368	1.000000	poly	1.0	0.5
6	train_SVM	0.824923	0.891064	0.797309	linear	1.0	0.0
7	train_SVM	0.928400	0.866368	1.000000	sigmoid	1.0	0.0



	train_function	f1_score	precision	recall	kernel	C	coef0
0	train_SVM	0.928400	0.866368	1.000000	poly	1.0	0.0
1	train_SVM	0.928400	0.866368	1.000000	poly	1.0	0.5
2	train_SVM	0.928400	0.866368	1.000000	poly	0.5	0.0
3	train_SVM	0.928400	0.866368	1.000000	poly	0.5	0.5
4	train_SVM	0.928400	0.866368	1.000000	poly	0.1	0.0
5	train_SVM	0.928400	0.866368	1.000000	poly	1.0	0.5
6	train_SVM	0.825124	0.882570	0.799103	linear	1.0	0.0
7	train_SVM	0.928400	0.866368	1.000000	sigmoid	1.0	0.0

PCA = {500, 1000}

Springboard

Conclusion

- Best model

  - SVM classifier, PCA = 500 (or 1000), kernel = ‘poly’, coef0 and hyperparameter C didn’t play recognizable role, therefore we keep the defaults from sci-kit learn, with f1_score = 0.9284.

- Assumption and limitations
- Future work
- What we have learnt?

Springboard

Karolina Bzdusek,
20th May 2019

Thank you!

(Q&A)

Springboard

Karolina Bzdusek,
20th May 2019