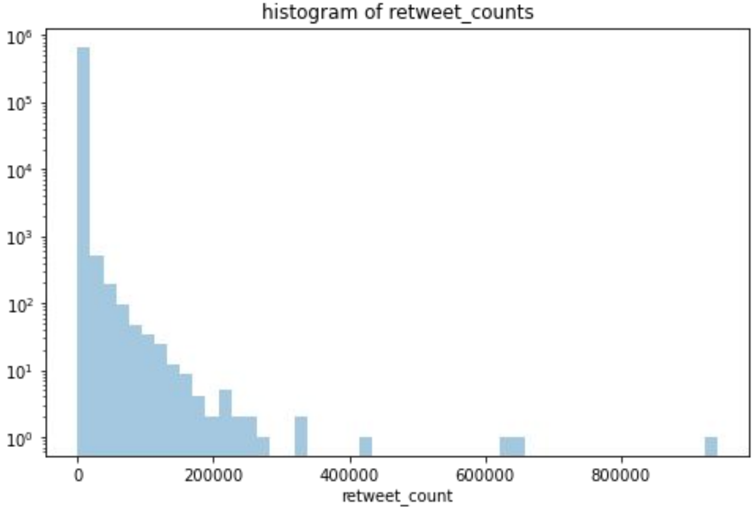
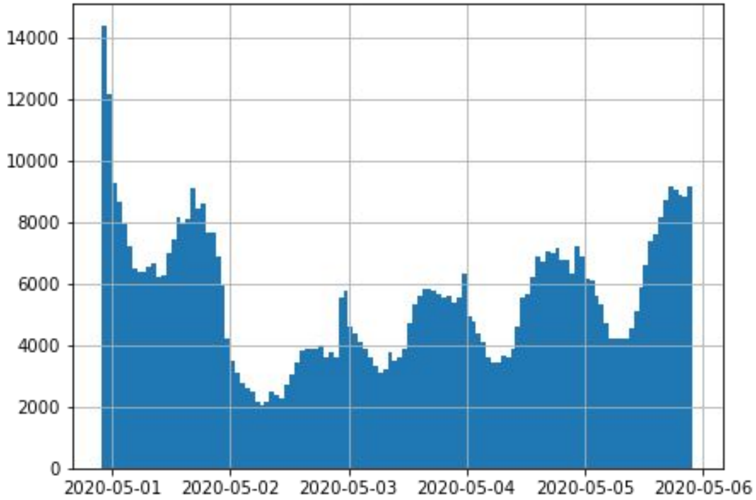
Team Batiki

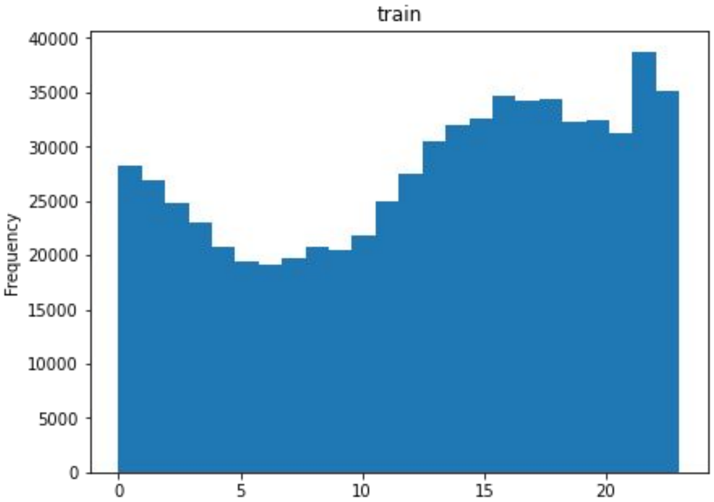
Members: ALBUQUERQUE SILVA Igor
GALVÃO LOPES Aloysio
MAIA MORAIS Lucas

Exploratory Data Analysis



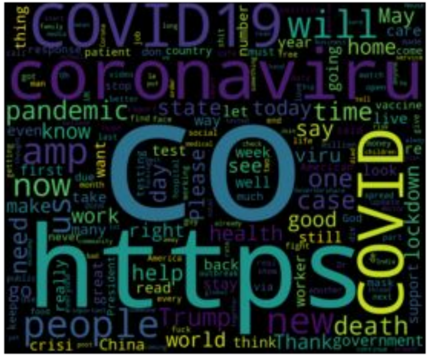
Exploratory Data Analysis





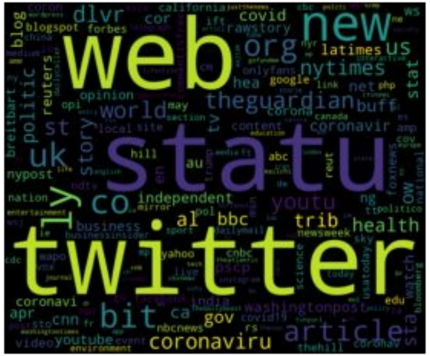
Exploratory Data Analysis

Tweets in Train DS



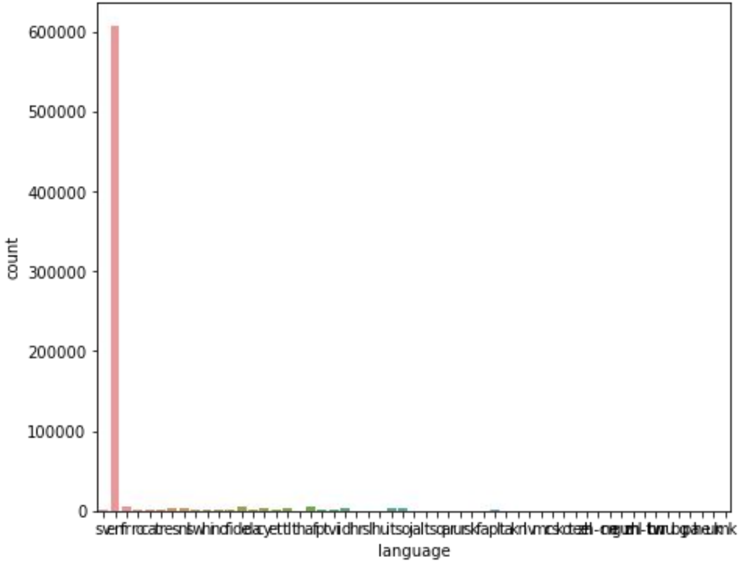
Hashtags in Train DS

Urls in Train DS



Mentions in Train DS

Exploratory Data Analysis



Exploratory Data Analysis

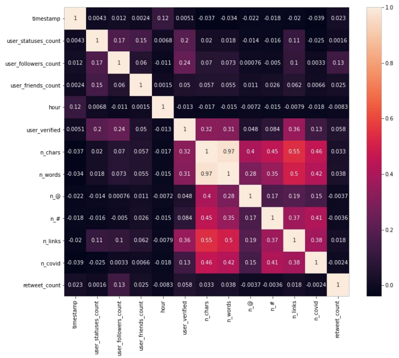
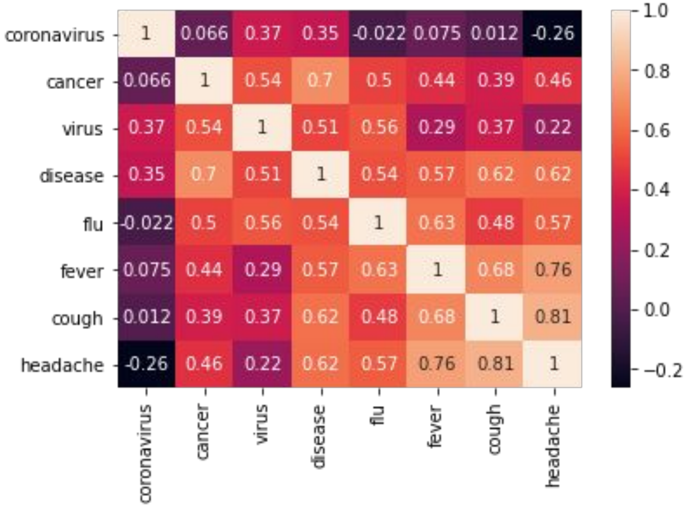


Table 1: Feature importance measured using correlations and a random forest model.

	n_followers	verified	n_words	n_chars	n_friends	timestamp	n_urls	n_statuses	n_covid	n_#	n_@	hour
Correlation	0.13	0.06	0.04	0.03	0.02	0.02	0.02	0.00	-0.00	-0.00	-0.00	-0.01
Random Forest	0.37	0.00	0.05	0.07	0.09	0.12	0.01	0.16	0.01	0.01	0.00	0.04

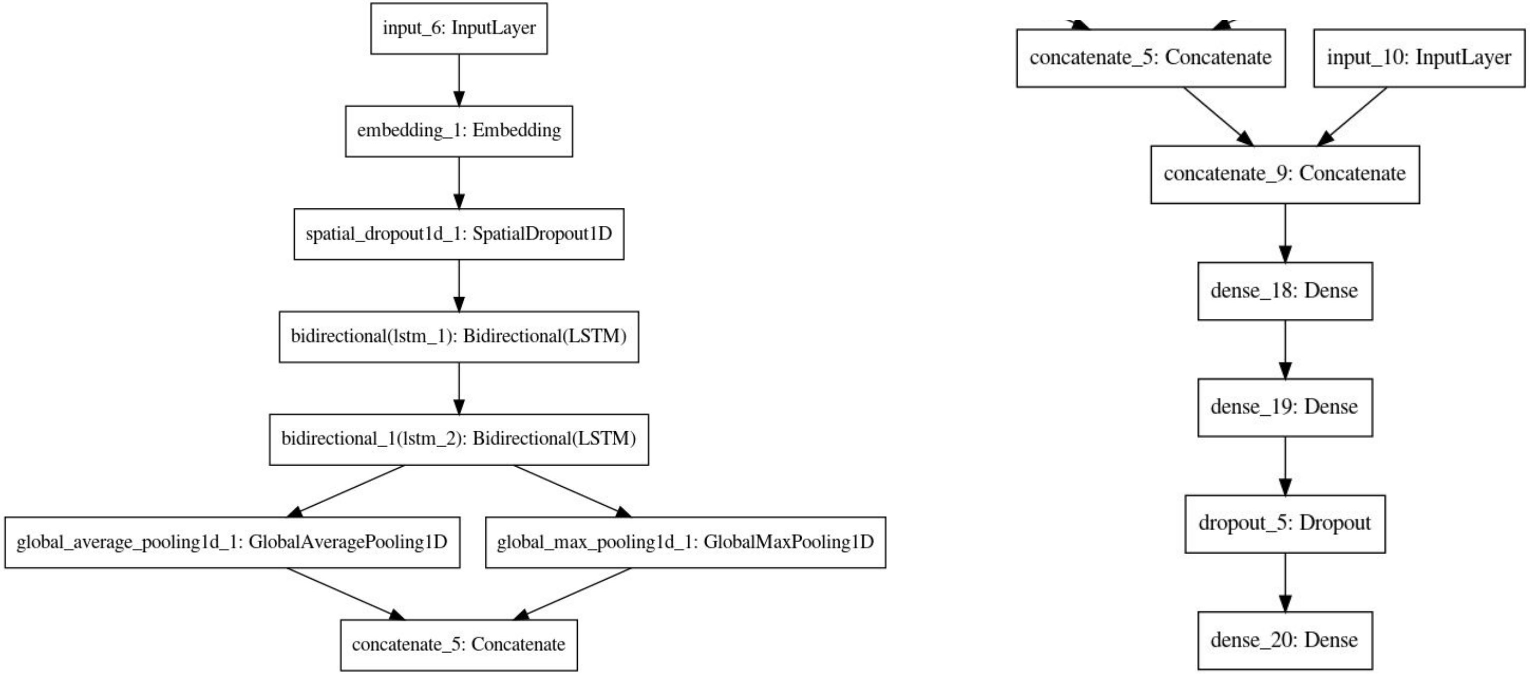
First model: GloVe

	input	preprocessed
0	https://t.co/hfuhufds	<url>
1	@kaggle	<user>
2	:')	<smile>
3	8`p	<lolface>
4	)-=	<sadface>
5	;/	<neutralface>
6	<3	<heart>
7	123.56	<number>
8	#kaggle	<hashtag> kaggle
9	#KAGGLE	<hashtag> kaggle <allcaps>
10	#KaggleCompetition	<hashtag> kaggle competition
11	kaggle!!!!!	kaggle! <repeat>
12	kaggleeee	kaggle <elong>
13	KAGGLE	kaggle <allcaps>
14	cat/dog	cat / dog




```
Found embeddings for 57.38% of vocab
Found embeddings for  98.92% of all text
```

First model: GloVe + LSTM



First approach

- The correlations indicated very small linear correlation between features and retweet count
- Feed forward neural network should behave well with lack of linearity
- Feed forward neural network with numerical features
- Equivalent to constant zero after training

After thinking a little bit more

- Should be able to able to capture nonlinear relations from the numerical data
- A spline regression would be a possibility
- Approximate by ranges could also be a good idea
- These ideas are similar to estimate the point based on a neighborhood

KNN!

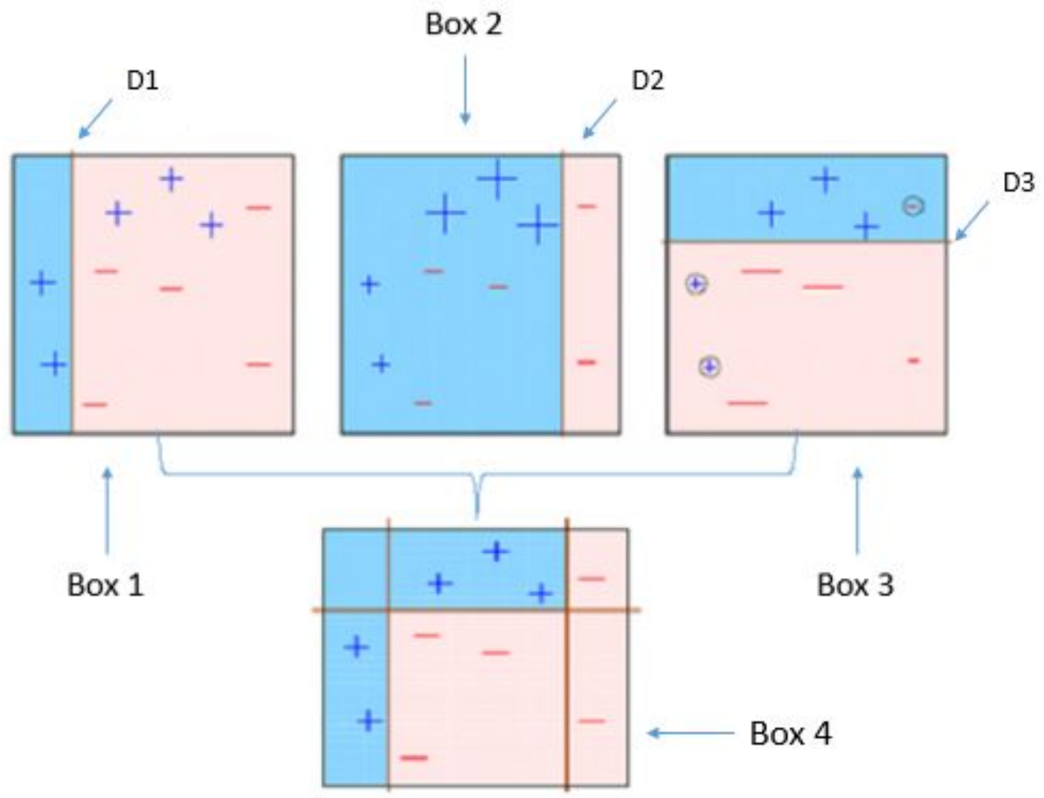
- We decided to use KNN classifier just with the numerical features and it yielded awesome results
- We selected the features that were working best and also the best number of neighbors, 3.
- Very hard to generalize to lots of features because of the metric

Tentative of improvement of the KNN using neural networks

- First use NN to learn metric, but new metric on KNN was too slow and NN wasn't learning anything
- Then try to predict inputting to a NN the predictions of the nearest neighbors
- Results still worse than KNN we believe that because there was not enough data and because of the imbalanced dataset
- Upsampling/downsampling didn't work

New ideas...

- We could divide the space in different zones based on different criteria
- For example if user verified or not, if n_followers > 100 or not etc
- For each zone we could predict the the data point to the best prediction in a given zone


Box 2
D1
D2
D3
Box 1
Box 3
Box 4

But what's the best prediction for a given zone

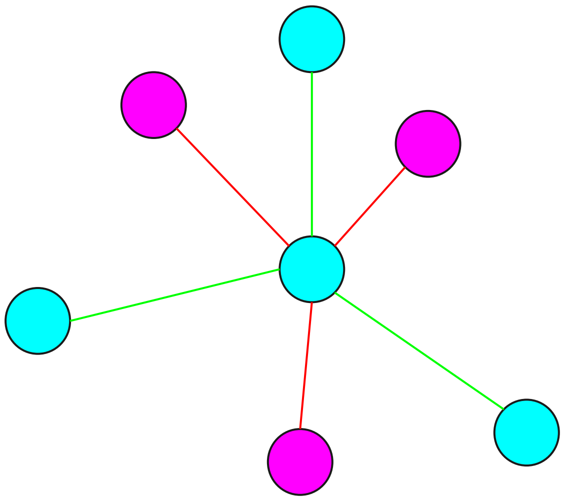
- We realized that given a set of points with different values, if we introduce a new value, its MAE in relation to the others is minimized if the value is the median, not the average!
- This way if we assume that all of the point in a given region are equally likely the median of the region is the best prediction

A new inspiration for the KNN

- We can apply the same principle for the KNN, we can select the median among the neighbors
- This gave us an incredibly big performance boost (the two main ones were KNN and KNN+median)
- We could make it a little bit better by selecting the median-1 as not all neighbors are equal (this is equivalent to be more conservative)
- Now from 3 we are considering 10 neighbors

An extra boost to the KNN

- The feature user verified is boolean, thus easy to consider
- Do KNN in groups didn't work
- We decided to bring closer the verified ones and that worked!

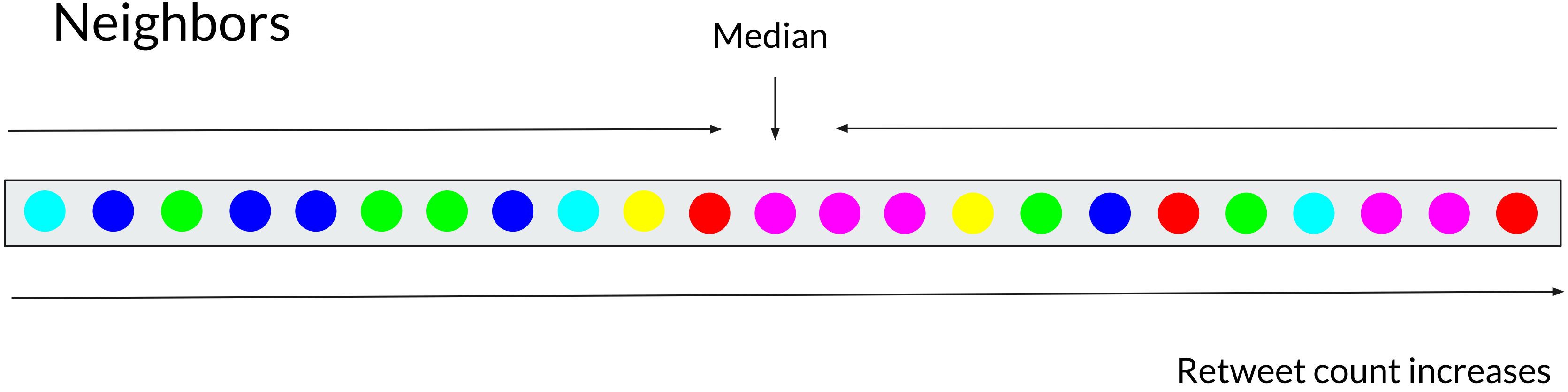


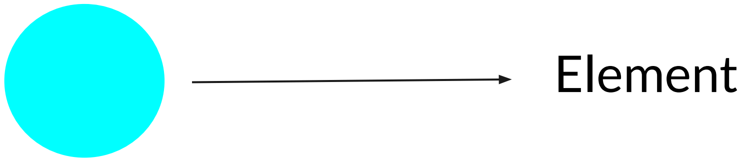
New possibilities

- Even just considering two neighbors the best neighbor prediction is better than anything we did
- A key possibility would be to choose better the neighbor
- We decided to study the text to better decide which neighbor to pick

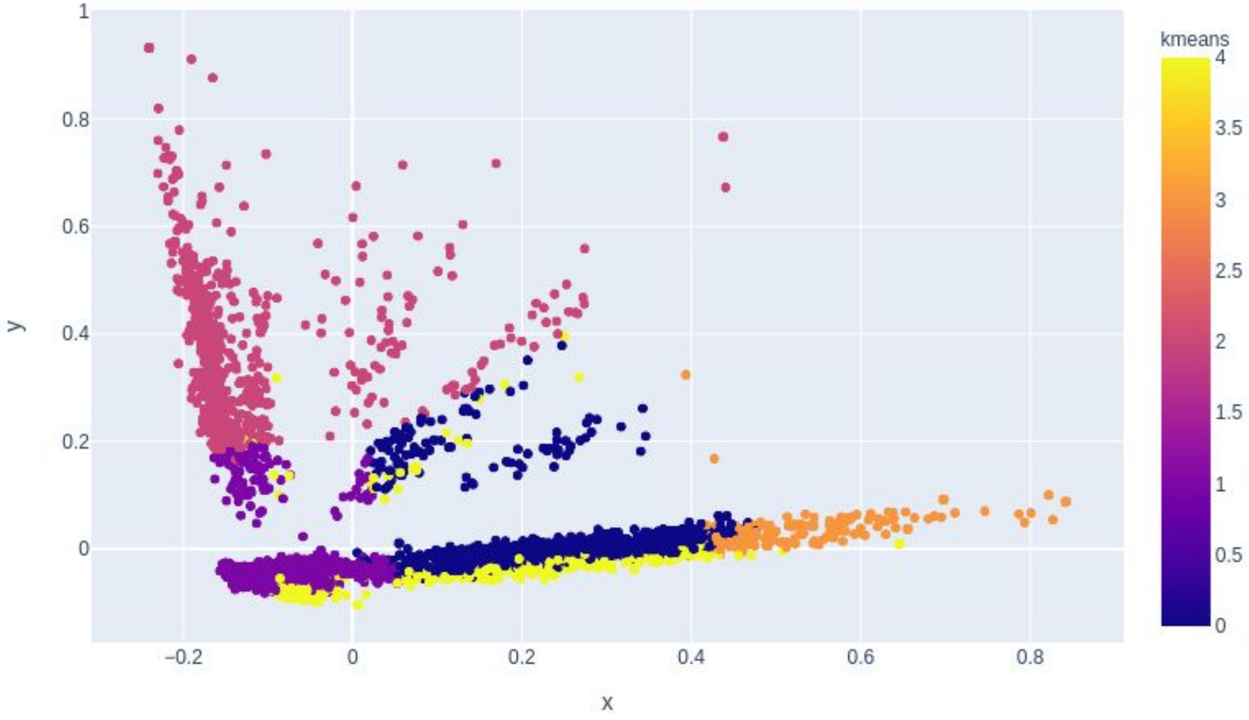
All-zero	143
Median-KNN	135
Best KNN n_neighbors= inf	1.7
Best KNN n_neighbors=1000	32
Best KNN n_neighbors=100	65
Best KNN n_neighbors=50	73
Best KNN n_neighbors=10	92
Best KNN n_neighbors=5	103
Best KNN n_neighbors=3	111
Best KNN n_neighbors=2	127
Best KNN n_neighbors=1	231

Neighbors

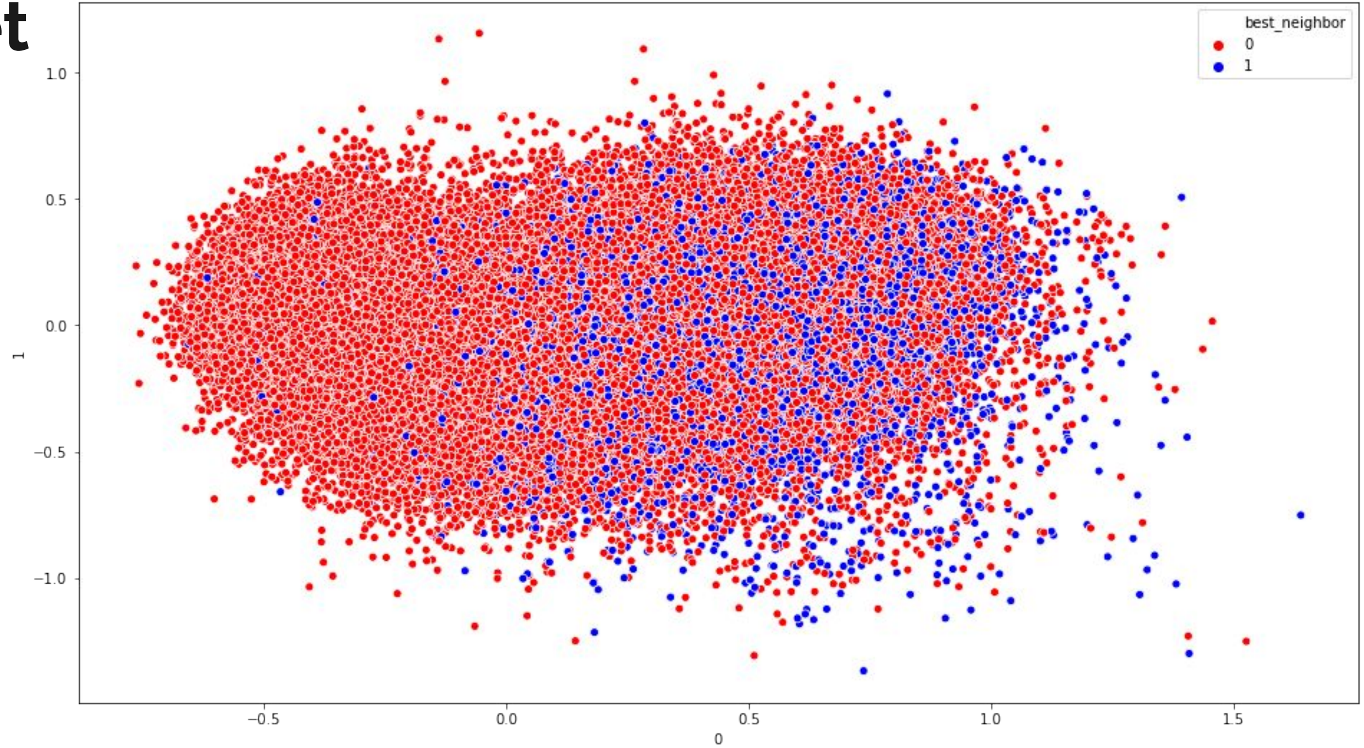




TF-IDF

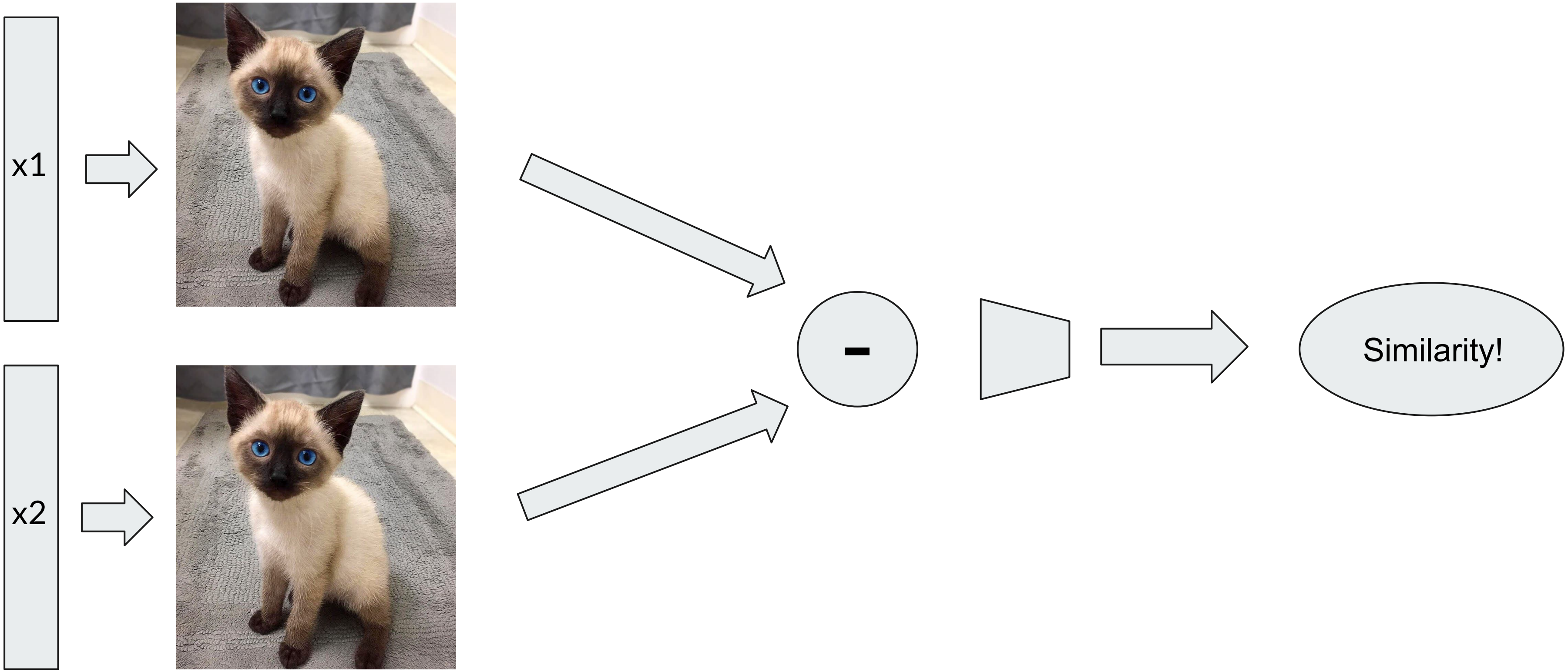


BERTweet



How to integrate the text embedding in our KNN

- Two approaches: learn a similarity measure or use the embedding directly on a classifier or regressor
- We tried to train siamese networks to learn a similarity measure between tweets but that didn't work


x1
x2
-
Similarity!

Gradient Boosting

- Predict best neighbor index, using numeric and textual features.
- Calculate training data neighbors on KNN and exclude the first one.
- Using neighbors features didn't provide noticeable improvement.

PCA 1	followers_count	statuses_count	PCA 2	friends_count	verified
0.59	0.18	0.07	0.06	0.05	0.04

Final results

Table 2: Models MAE's for a Monte-Carlo cross validation.

	0	1	2	3	4	Average	Public	Private
XGBoost-uv-KNN	145.50632	139.22579	137.72118	140.47844	136.37302	139.86095	149.2915	130.8230
XGBoost-KNN	145.74780	138.72360	137.92012	140.62056	136.35116	139.87265	149.5399	131.2015
Grouped-KNN	145.93770	139.96711	137.74569	140.93580	136.61703	140.24066	150.2692	131.2121
Corrected-Median-UV	146.38611	140.08023	138.62018	141.46235	136.73156	140.65608	149.6434	131.7287
Corrected-10-Median	146.57085	140.09473	138.60681	141.52308	136.69178	140.69745	-	-
KNN	145.86524	139.18615	140.24764	143.34072	138.09455	141.34686	152.7042	131.0499
LogGradientBoosting	150.29024	143.07951	142.19235	145.13001	139.56261	144.05095	-	-
GradientBoosting	151.26142	144.10285	143.01257	145.65797	140.39442	144.88585	-	-
All-zero	155.43425	149.21205	147.99024	150.39015	145.28295	149.66193	161.0395	141.3318
RandomForest	235.29432	225.99619	231.22650	234.94854	231.32672	231.75845	-	-
LinearRegression	268.36285	260.95524	260.24058	265.35576	258.78022	262.73893	-	-