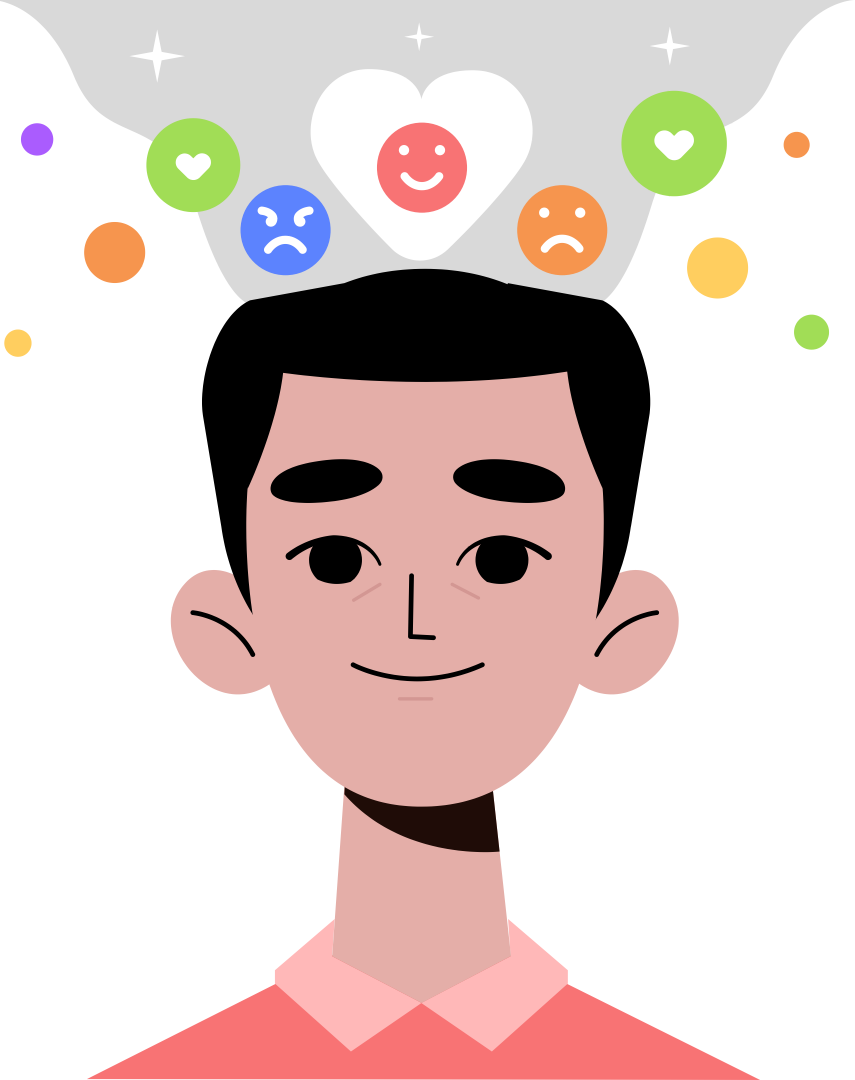
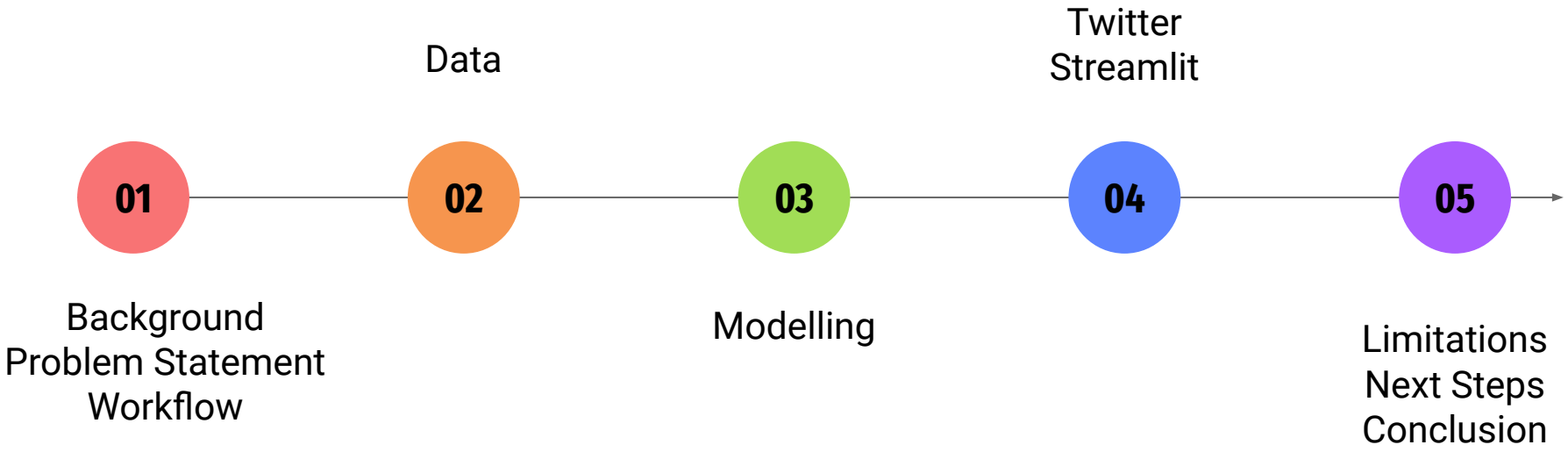
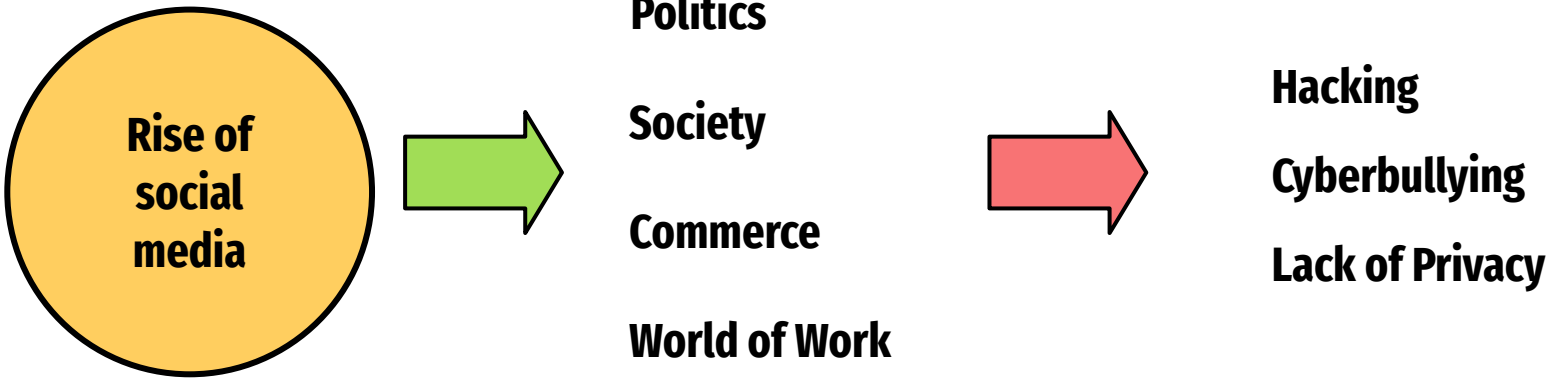


Tweet Sentiment and Toxic Tweets

contents


01
02
03
04
05
Data
Twitter
Streamlit
Background
Problem Statement
Workflow
Modelling
Limitations
Next Steps
Conclusion

01 background



01 problem statement

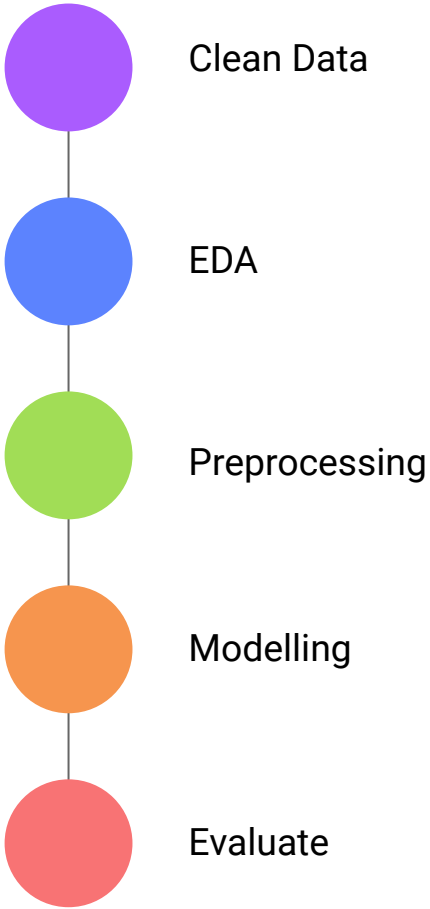
To identify sentiments:

- Sentiment analysis model
- >90% accuracy
- Identify between different sentiments
- Tweets

To identify toxic comments:

- Multi-label classification model
- >90% accuracy
- Rate level of toxicity in comments
- Tweets

01 workflow



02 data

Sentiment analysis

- Kaggle
- Twitter sentiment analysis dataset

Toxic comment classifier

- Kaggle
- Toxic comment classification

02 data: sentiment analysis: overview

74, 682 rows

4 columns
- ID
- Topic
- Tweet
- Sentiment

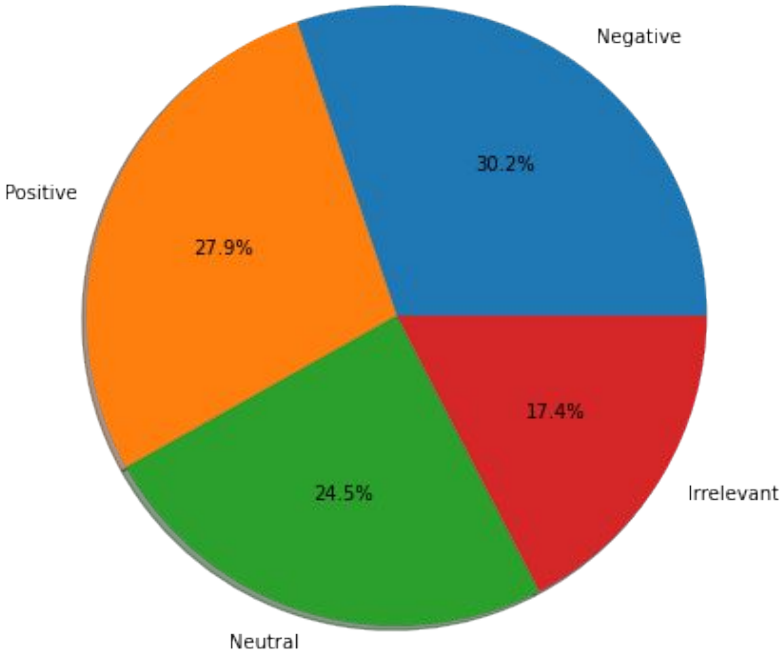
Topic

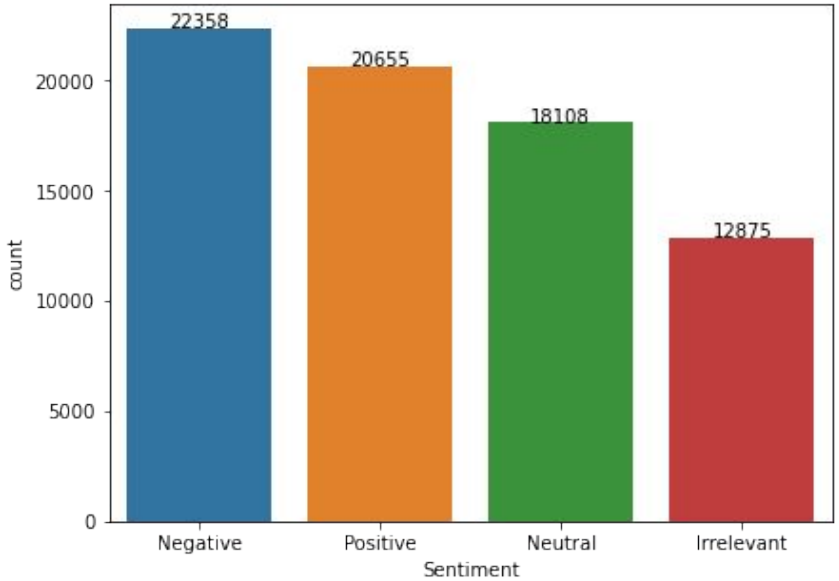
MaddenNFL	2377
LeagueOfLegends	2377
CallOfDuty	2376
Verizon	2365
TomClancysRainbowSix	2364
Facebook	2362
Microsoft	2361
Dota2	2359
WorldOfCraft	2357
ApexLegends	2353
NBA2K	2343
CallOfDutyBlackopsColdWar	2343
FIFA	2324
johnson&johnson	2324
TomClancysGhostRecon	2321
Battlefield	2316
Overwatch	2316
GrandTheftAuto(GTA)	2293
HomeDepot	2292
PlayStation5(PS5)	2291
Hearthstone	2286
CS-GO	2284
Xbox(Xseries)	2283
Borderlands	2280
Amazon	2276
Google	2274
Nvidia	2271
Cyberpunk2077	2262
RedDeadRedemption(RDR)	2249
Fortnite	2249
PlayerUnknownsBattlegrounds(PUBG)	2234
AssassinsCreed	2234

Sentiment

Negative	22358
Positive	20655
Neutral	18108
Irrelevant	12875

02 data: sentiment analysis: EDA





02 data: toxic comments classifier: overview

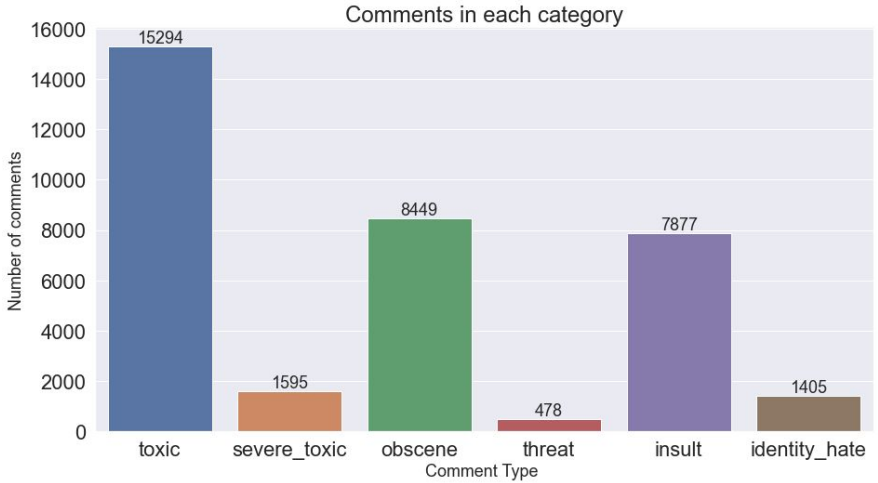
159, 571 rows

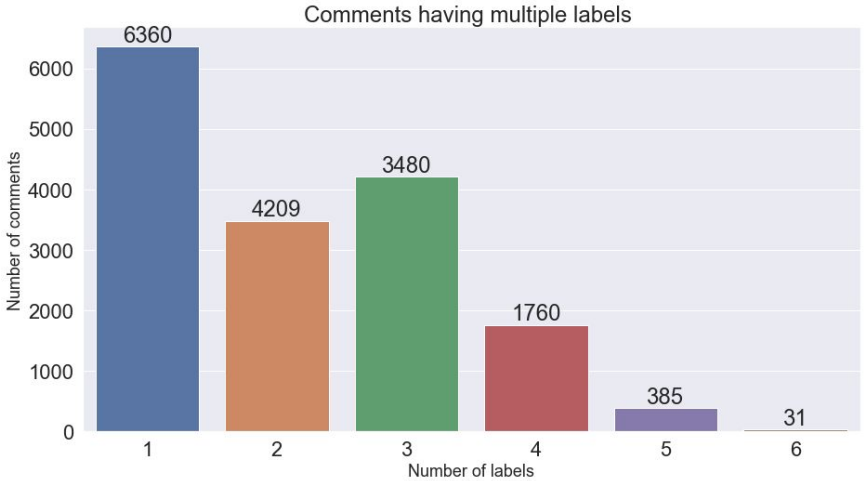
8 columns
- id
- comment_text
- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

```
Total number of comments =  159571
Number of clean comments =  143346
Number of comments with labels = 16225
```

	category	number of comments
0	toxic	15294
1	severe_toxic	1595
2	obscene	8449
3	threat	478
4	insult	7877
5	identity_hate	1405

02 data: toxic comments classifier: EDA





03 Preprocessing

Function was created to:

- Convert text to lower case
- Remove user's handle
- Remove http links
- Remove digits and special characters
- Remove contractions
- Tokenized the text
- Remove stopwords
- Lemmetize the text
- Remove hashtags
- Remove 'RT' characters
- Remove additional spaces

Same function was used in both sentiment analysis and toxic comments data

03 Model: Sentiment Analysis

Vectorizers:
- Count Vectorizer
- TF-IDF Vectorizer

*TF-IDF is an abbreviation for **Term Frequency Inverse Document Frequency**

Classifiers:
- Bernoulli Naive Bayes
- Multinomial Naive Bayes
- Random Forest

03 Model: Sentiment Analysis

	Vectorizer	Classification Model	Model Score
BASELINE	TF-IDF	Logistic Regression	0.748
	Count Vectorizer	Bernoulli NB	0.905
	Count Vectorizer	Multinomial NB	0.914
	Count Vectorizer	Random Forest	0.618
	TF-IDF	Bernoulli NB	0.777
	TF-IDF	Multinomial NB	0.792
	TF-IDF	Random Forest	0.554

03 Model: Toxic Comments

Vectorizer:
- TF-IDF Vectorizer

Classifiers:
- Multiple Binary Classifications
  - One vs Rest Classifier
  - Binary Relevance
- Classifier Chains
- Label Powerset

03 Model: Toxic Comments

Vectorizer	Classification Model	Model Score
TF-IDF	One vs Rest (Logistic Regression)	0.964
TF-IDF	Binary Relevance	0.513
TF-IDF	Classifier Chains	0.904
TF-IDF	Label Powerset	0.905

03 Model: Count Vectorizer

Machines cannot understand characters and words. So when dealing with text data we need to represent it in numbers to be understood by the machine. Countvectorizer is a method to convert text to numerical data.

```
text = ['Hello my name is james, this is my python
notebook']
```

	hello	is	james	my	name	notebook	python	this
0	1	2	1	2	1	1	1	1

Countvectorizer makes it easy for text data to be used directly in machine learning and deep learning models such as text classification.

03 Model: TF-IDF Vectorizer

TF-IDF is an abbreviation for **Term Frequency Inverse Document Frequency**. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction.

Train Document Set:
d1: The sky is blue.
d2: The sun is bright.
Test Document Set:
d3: The sun **in** the sky is bright.
d4: We can see the shining sun, the bright sun.

Count Vectorizer

	blue	bright	sky	sun
Doc1	1	0	1	0
Doc2	0	1	0	1

TD-IDF Vectorizer

	blue	bright	sky	sun
Doc1	0.707107	0.000000	0.707107	0.000000
Doc2	0.000000	0.707107	0.000000	0.707107

Count Vectorizer give number of frequency with respect to index of vocabulary where as *tf-idf* consider overall documents of weight of words.
In TfidfVectorizer it is the **overall document weightage** of a word. It helps us in dealing with most frequent words. Using it we can penalize them. TfidfVectorizer weights the word counts by a measure of how often they appear in the documents.

03 Model: Bernoulli Naive Bayes

Bernoulli Naive Bayes is a part of the family of Naive Bayes. It only takes binary values. The most general example is where we check if each value will be whether or not a word that appears in a document. That is a very simplified model. In cases where counting the word frequency is less important, Bernoulli may give better results. In simple words, we have to count every value binary term occurrence features i.e. a word occurs in a document or not. These features are used rather than finding the frequency of a word in the document.

To understand it in layman's terms,Bernoulli distribution has two mutually exclusive outcomes:P(X=1)=p or P(X=0)=1-p.In BernoulliNB theorem, we can have multiple features but each one is assumed to be binary valued variable i.e. boolean. Therefore, this class requires samples to be represented as binary-valued feature vectors. In case, any other kind of data, is provided, then a BernoulliNB instance may binarize its input.

Naive Bayes is considerably best suitable for large datasets problem. BernoulliNB work only for Binary values and produces results that are computationally better than other traditional algorithms. Further, probabilistic nature make it more stable in relation to results. BeronoulliNB demands higher independence in feature in dataset. Thus, co relation in feature reduced the performance of the algorithm. This is the major drawback of this algorithm.

03 Model: Multinomial Naive Bayes

The multinomial naïve Bayes is widely used for assigning documents to classes based on the statistical analysis of their contents. It provides an alternative to the "heavy" AI-based semantic analysis and drastically simplifies textual data classification.

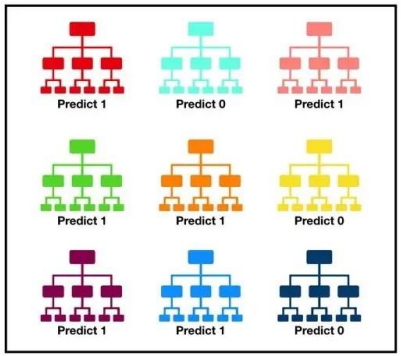
The classification aims to assign fragments of text (i.e. documents) to classes by determining the probability that a document belongs to the class of other documents, having the same subject.
Each document consists of multiple words (i.e. terms), that contribute to an understanding of a document's contents. A class is a tag of one or multiple documents, referring to the same subject.

The multinomial Bayesian model is an efficient alternative to the known K-means clustering and decision trees algorithms, capable of classifying various data that are normally not easy to be quantified. For example, it can be used as part of ANN-based text classification models that learn and predict texts to classes, based on the summary of their contents, inferred by the multinomial Bayesian classifier.

Unlike similar AI and machine learning (ML), used for content-based texts classification, the multinomial Bayesian classifiers are entirely a data mining approach, that allows predicting classes for texts, introduced to the model, without its continuous training. However, to prevent the early convergence and the cold start issues, encountered in the multinomial models, it's recommended to use the semi-supervised learning algorithm to train the model for a better prediction of texts. This is normally done by training the model and simultaneously using it for prediction.

03 Model: Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.



Tally: Six 1s and Three 0s
Prediction: 1

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

03 Model: One vs Rest

One-vs-rest (OvR for short, also referred to as One-vs-All or OvA) is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

A possible downside of this approach is that it requires one model to be created for each class. For example, three classes requires three models. This could be an issue for large datasets (e.g. millions of rows), slow models (e.g. neural networks), or very large numbers of classes (e.g. hundreds of classes). This approach requires that each model predicts a class membership probability or a probability-like score. The argmax of these scores (class index with the largest score) is then used to predict a class.

This approach is commonly used for algorithms that naturally predict numerical class membership probability or score, such as:

- Logistic Regression
- Perceptron

03 Model: Binary Relevance

In this case an ensemble of single-label binary classifiers is trained, one for each class. Each classifier predicts either the membership or the non-membership of one class. The union of all classes that were predicted is taken as the multi-label output. This approach is popular because it is easy to implement, however it also ignores the possible correlations between class labels.

In other words, if there's *q* labels, the binary relevance method create *q* new data sets from the images, one for each label and train single-label classifiers on each new data set. One classifier may answer yes/no to the question "does it contain trees?", thus the "binary" in "binary relevance". This is a simple approach but does not work well when there's dependencies between the labels.

OneVsRest & Binary Relevance seem very much alike. If multiple classifiers in OneVsRest answer *"yes"* then you are back to the binary relevance scenario.

X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	0
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	0
$x^{(4)}$	1	0	0	1
$x^{(5)}$	0	0	0	1

X	Y_1
$x^{(1)}$	0
$x^{(2)}$	1
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	0

X	Y_2
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	1
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_3
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_4
$x^{(1)}$	0
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	1

03 Model: Classifier Chains

A chain of binary classifiers C0, C1, . . . , Cn is constructed, where a classifier Ci uses the predictions of all the classifier Cj , where j < i. This way the method, also called classifier chains (CC), can take into account label correlations.

The total number of classifiers needed for this approach is equal to the number of classes, but the training of the classifiers is more involved.

Following is an illustrated example with a classification problem of three categories {C1, C2, C3} chained in that order.

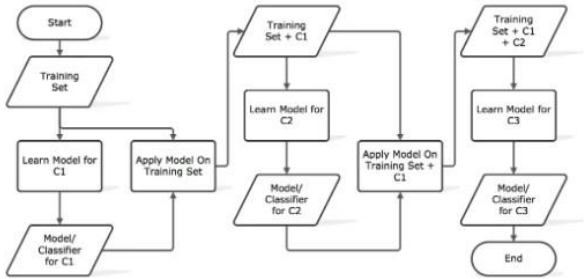


Fig-13: Classifier Chains

03 Model: Label Powerset

This approach does take possible correlations between class labels into account. More commonly this approach is called the label-powerset method, because it considers each member of the power set of labels in the training set as a single label.

This method needs worst case (2^|C|) classifiers, and has a high computational complexity.

However when the number of classes increases the number of distinct label combinations can grow exponentially. This easily leads to combinatorial explosion and thus computational infeasibility. Furthermore, some label combinations will have very few positive examples.

04 twitter & streamlit

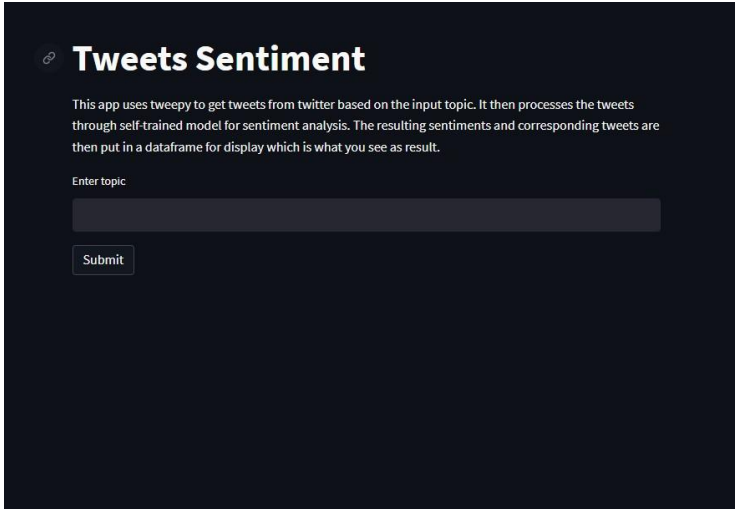
Twitter API v2

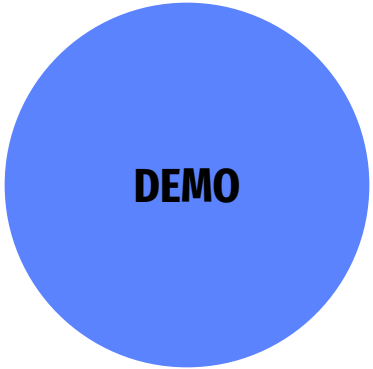
Elevated Access
- 3 environments per project
- 2M Tweets per month/ Project
- Free

Search all recent tweets

Rate limit	Tweet cap	Special attributes
180 requests / 15 mins **PER USER** 450 requests / 15 mins **PER APP**	yes	• 10 default results per response • 100 results per response • 512 query length • core operators

Streamlit




DEMO

05 limitations & moving forward

Sentiment Model

- Sentiment 140 Dataset
  - 1.6 million rows of data
- Identifying abbreviations
- Identifying images/ gif/ videos

Toxic Model

- More toxic data
- Hyperparameter tuning of models
- Developing a deep learning model

Twitter

- Filter only for English
  or
- Include all languages

Dashboard Development

- Real-time live dashboard
  - Generate sentiments
  - Detecting toxic comments and removing them

05 conclusion

Problem Statement:

Sentiment analysis model:

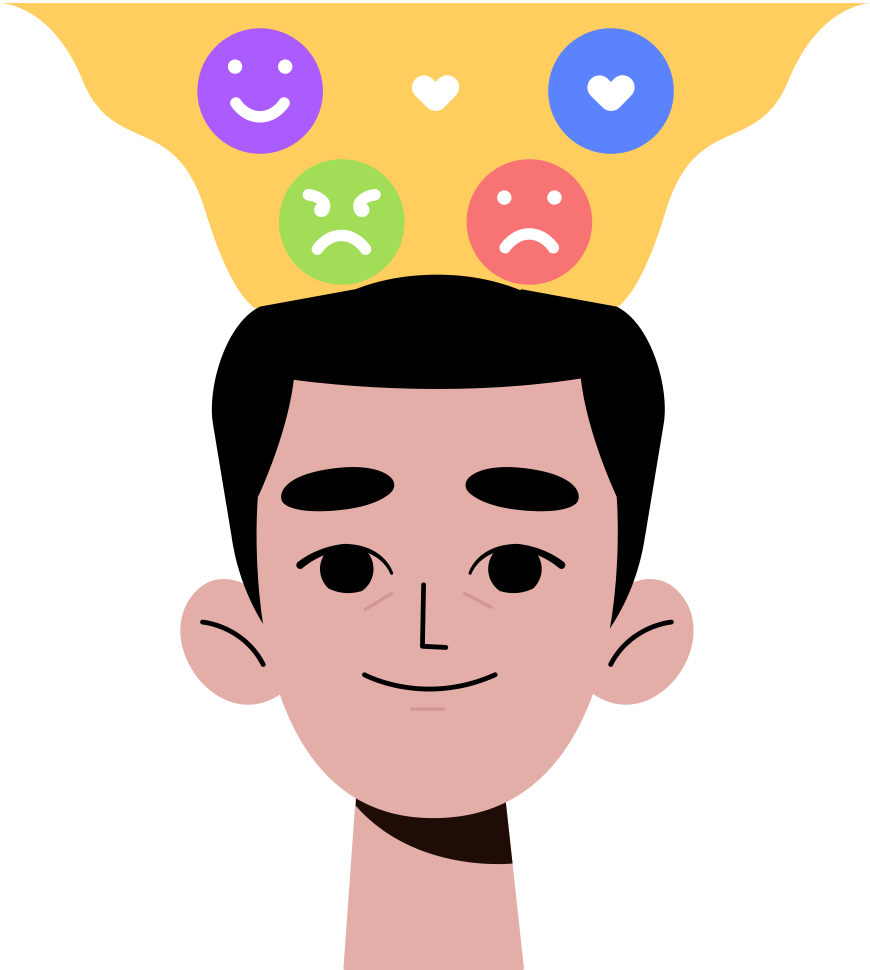
91% accuracy

Toxic comment model:

96% accuracy

Additionally:

- Further development of models
- Accessibility to languages and mediums
- Creating a real-time live dashboard



Thank you questions?