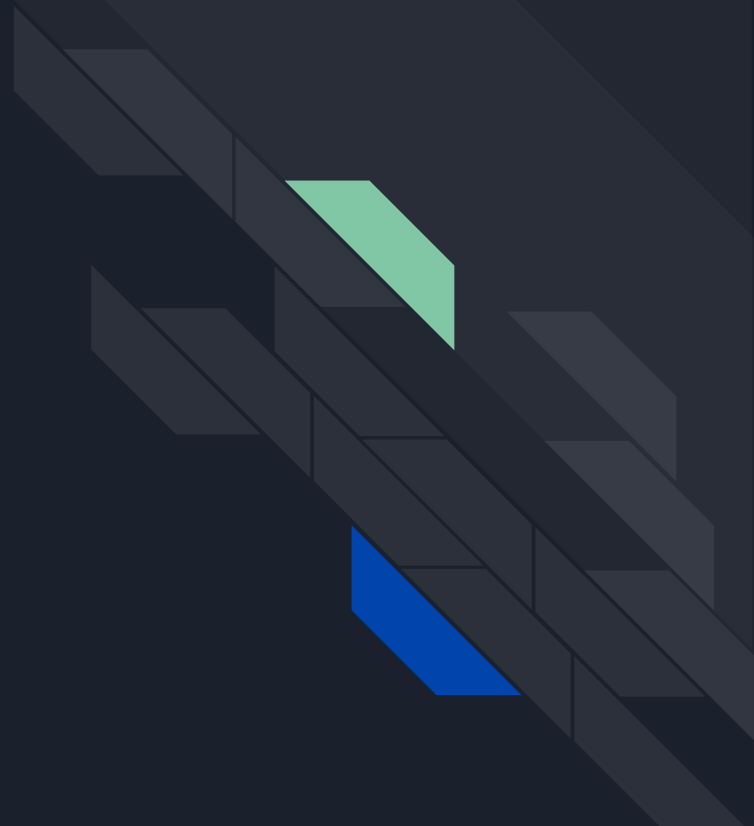


IMDB Movie Reviews Sentiment Classification

Aleksa Milenovic





Goal of this Project



Hey, this movie is amazing!



**POSITIVE
COMMENT**



Wow, such a terrible movie!



**NEGATIVE
COMMENT**

~90% Metric

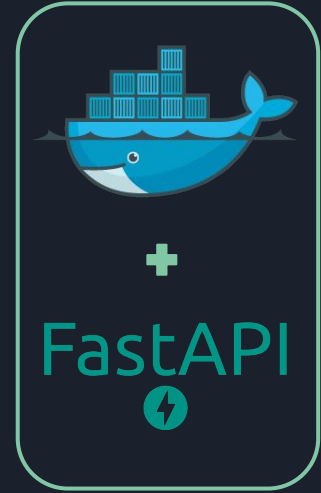
Technologies overview



Preprocessing
Classifier training

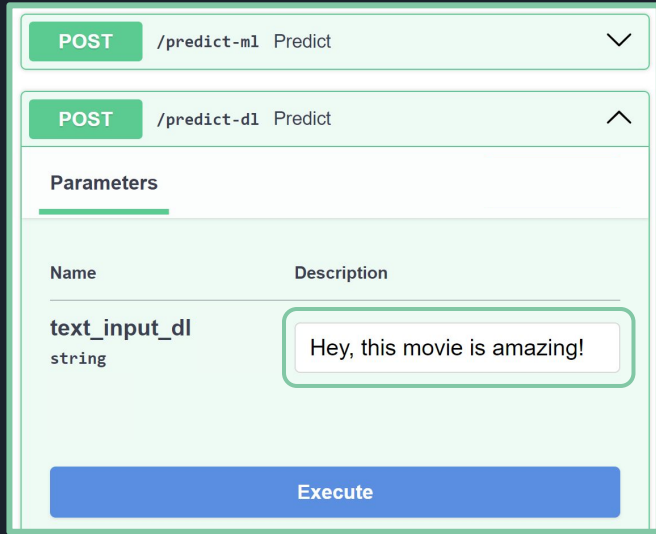


Model Logging



Deployment

Result product



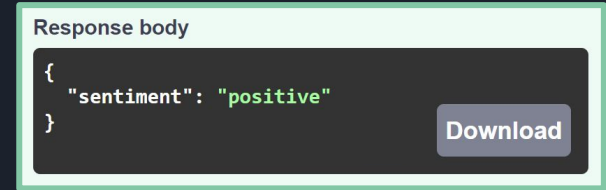
The image shows a FastAPI endpoint interface. At the top, there are two tabs: "POST /predict-m1 Predict" (selected) and "POST /predict-dl Predict". Below the tabs is a "Parameters" section. It contains a table with two columns: "Name" and "Description". The first row shows "text_input_dl" with the type "string". To the right of the table is a text input field containing the sentence "Hey, this movie is amazing!". At the bottom of the interface is a blue "Execute" button.

Name	Description
text_input_dl	string

Hey, this movie is amazing!

Execute

Example sentence on Fast API endpoint



The image shows a "Response body" section. It contains a JSON object: {"sentiment": "positive"}. To the right of the JSON is a grey "Download" button.

```
{  
  "sentiment": "positive"  
}
```

Download

Output Sentiment



Table of Contents

- Introduction
- Machine Learning approach
- Deep Learning approach
- Result overview
- Conclusion



Introduction



NLP



Sentiment
Classification

Dataset

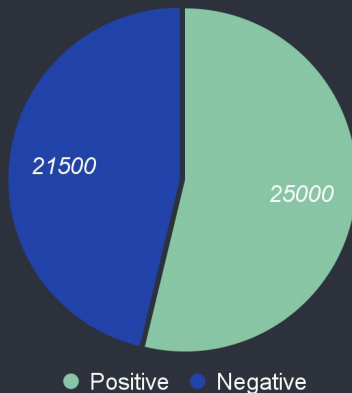


This film made John Glover a star. Alan Raimy is one of the most compelling character that I have ever seen on film., positive

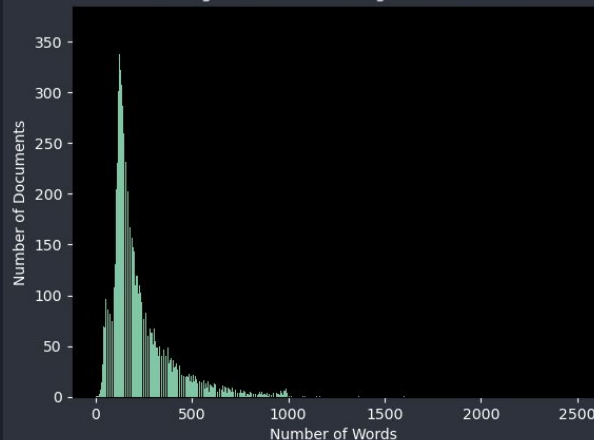


- 46,500 reviews
- 419,475 unique words

Class Distribution



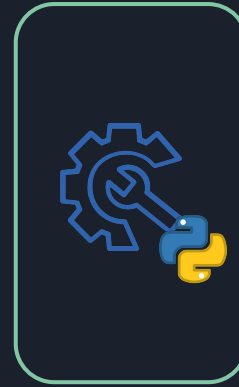
Histogram of Review Length Distribution



Machine Learning approach



Preprocessing



Training models
(Hyperparameter tuning)

Preprocessing Steps 1/7

- Lowercase text



Hey, this movie is A amazing!
You should REALLY watch it! I'm
watching it again! :)

<http://google.com>.



hey, this movie is a amazing!
you should really watch it! i'm
watching it again! :)

<http://google.com>.



Preprocessing Steps 2/7

- Split contractions (*can't* -> *can not*)



hey, this movie is amazing!
you should really watch it! i'm
watching it again! :)

<http://google.com>.



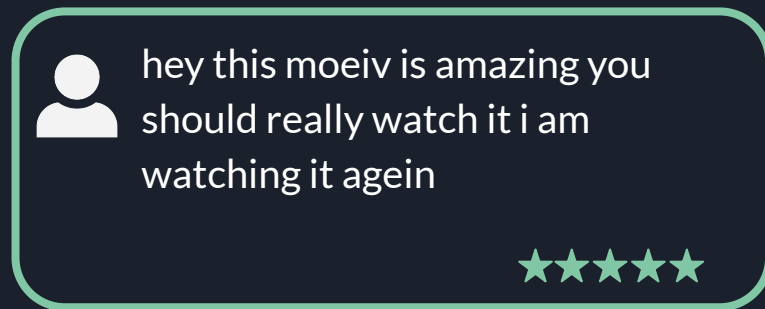
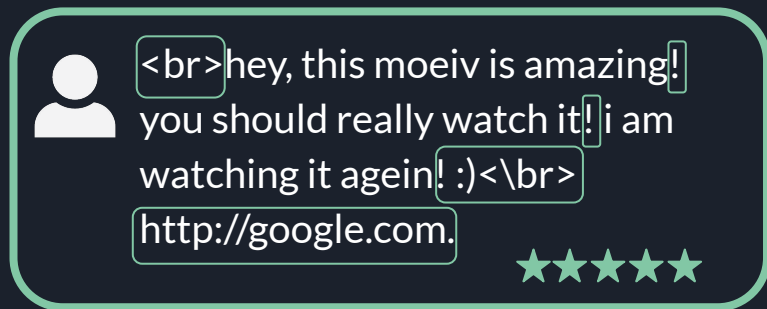
hey, this movie is amazing!
you should really watch it! i am
watching it again! :)

<http://google.com>.



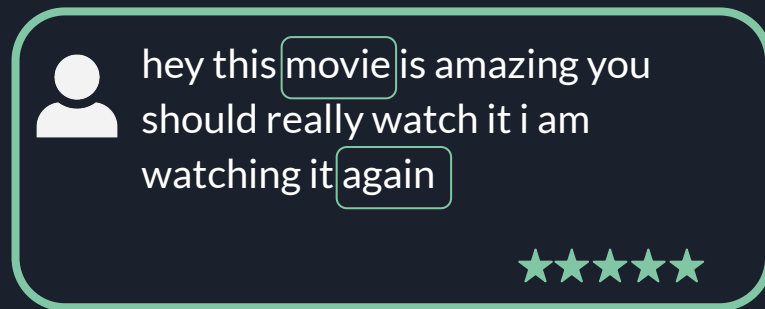
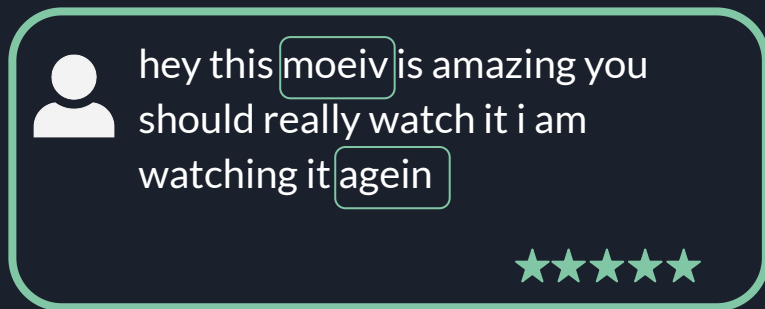
Preprocessing Steps 3/7

- Clean text (*remove urls, html tags, special characters...*)



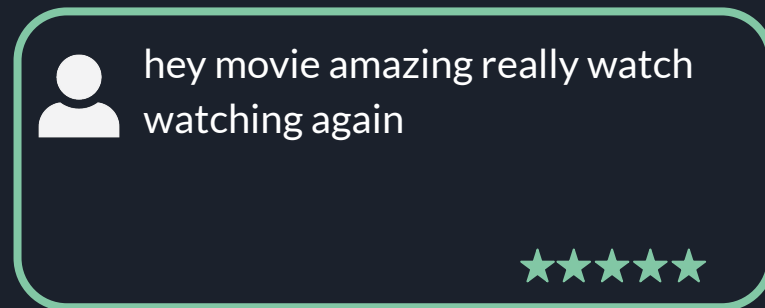
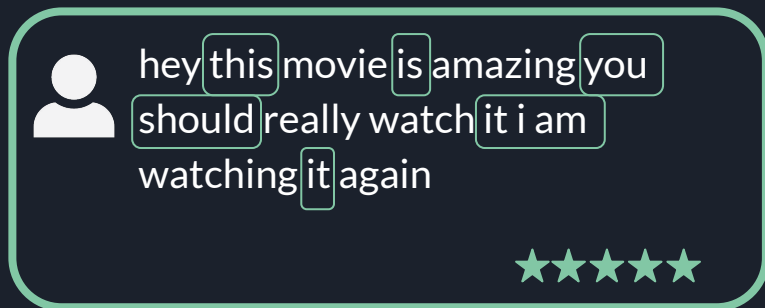
Preprocessing Steps 4/7

- Correct spelling mistakes



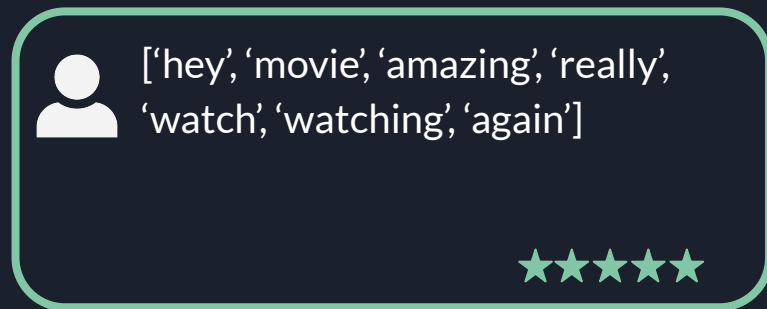
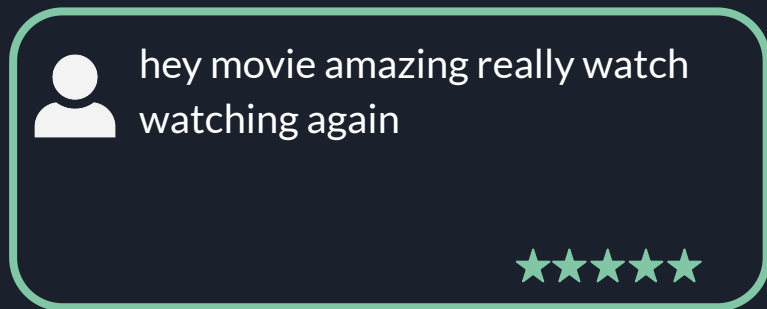
Preprocessing Steps 5/7

- Remove stop words (*the, a, is, what,...*)



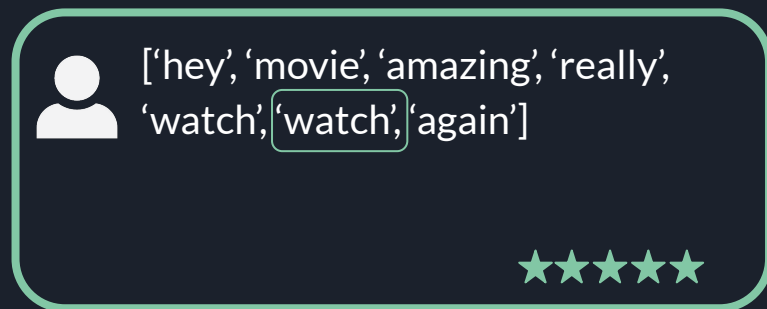
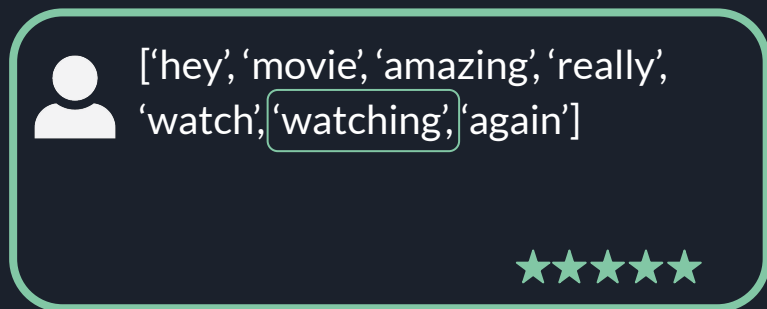
Preprocessing Steps 6/7

- Tokenization



Preprocessing Steps 7/7

- Lemmatization (*meeting* -> *meet*, *was* -> *be*, *mice* -> *mouse*,...)





Train-test Split

IMDB Dataset

Training data

Test data

80%

20%



TF-IDF Vectorization

$$\text{TFIDF}(t,d) = \boxed{\text{TF}(t,d)} \times \boxed{\text{IDF}(t)}$$

Term Frequency

=

$$\frac{\text{Word occurrences in a sentence}}{\text{No of words in a sentence}}$$

Inverse Document
Frequency

= log (

$$\frac{\text{Number of sentences}}{\text{Number of sentences containing word}})$$

TF-IDF Vectorization

1. good movie

2. good book

3. movie book good

TF	1	2	3
G	1/2	1/2	1/3
M	1/2	0	1/3
B	0	1/2	1/3

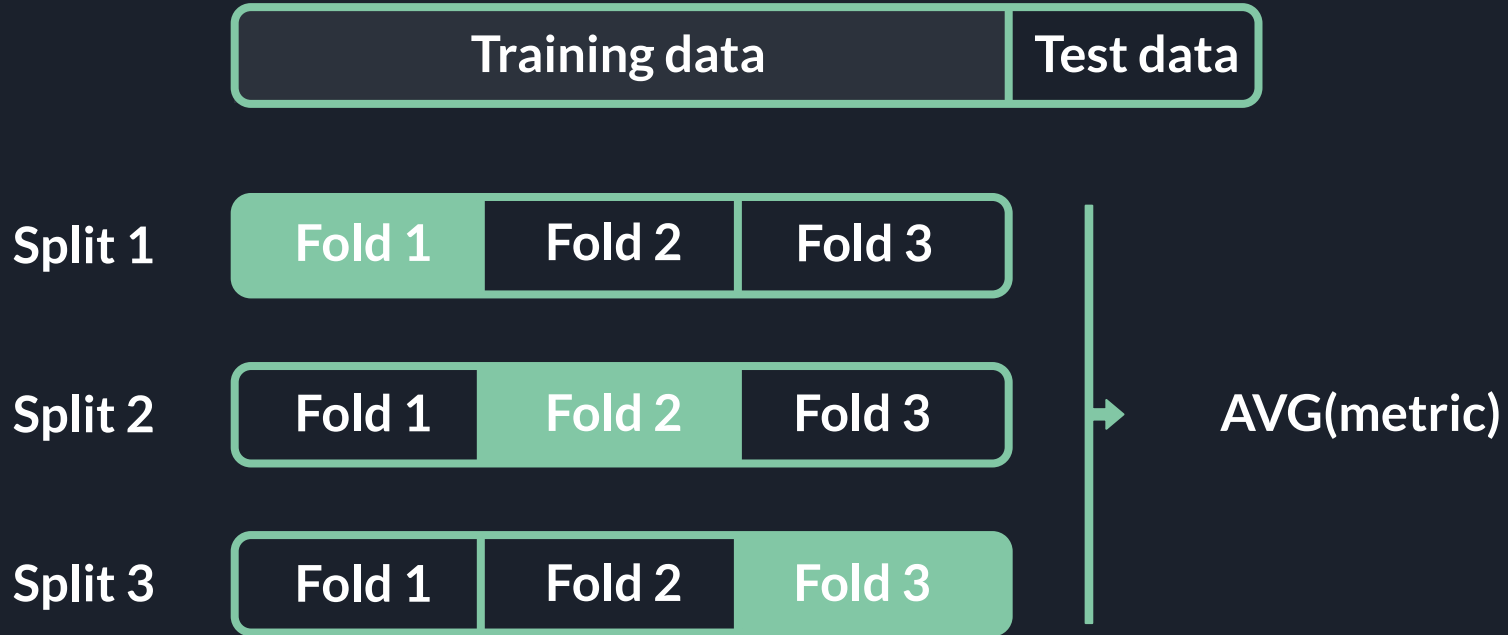
×

	IDF
G	$\log(3/3)$
M	$\log(3/2)$
B	$\log(3/2)$

=

TF	G	M	B
1	0	$\frac{1}{2}^* \log(3/2)$	0
2	0	0	$\frac{1}{2}^* \log(3/2)$
3	0	$\frac{1}{3}^* \log(3/2)$	$\frac{1}{3}^* \log(3/2)$

K-split cross validation





Performance Metrics

		ACTUAL VALUE	
		+	-
PREDICTED VALUE	+	TP	FN
	-	FP	TN

Confusion Matrix

Accuracy:
$$\frac{TP + TN}{TP + FN + FP + TN}$$

Performance Metrics - Balanced

		ACTUAL VALUE	
		+	-
PREDICTED VALUE	+	80	10
	-	20	90

Confusion Matrix

$$\frac{TP + TN}{TP + FN + FP + TN}$$

$$\frac{80 + 90}{80 + 10 + 20 + 90} = 85\%$$

Performance Metrics - Imbalanced

		ACTUAL VALUE	
		+	-
PREDICTED VALUE	+	980	90
	-	20	10

Confusion Matrix

$$\frac{TP + TN}{TP + FN + FP + TN}$$

$$\frac{980 + 10}{980 + 20 + 90 + 10} = 90\%$$

Performance Metrics - Imbalanced

		ACTUAL VALUE	
		+	-
PREDICTED VALUE	+	TP	FN
	-	FP	TN

Confusion Matrix

Precision:

$$\frac{TP}{TP + FP}$$

Recall:

$$\frac{TP}{TP + FN}$$

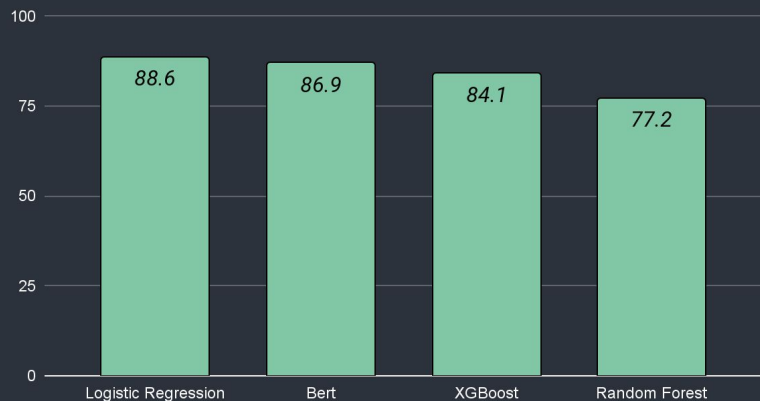
F1 Score:

$$2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

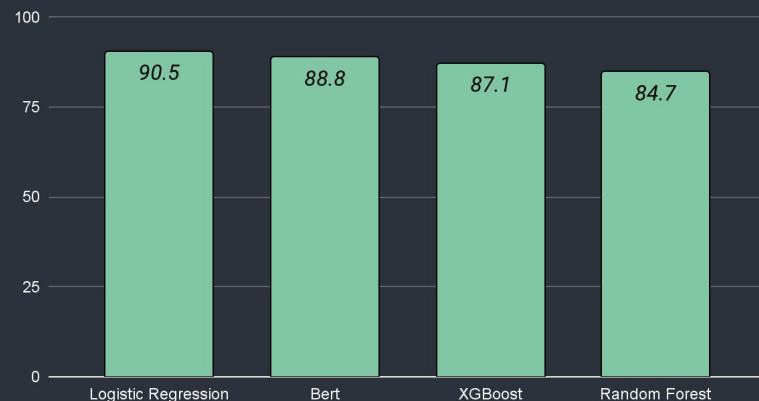
Results

- Best results for each model (*F1 Score*)

F1 Score for Negative class



F1 Score for Positive class





Results

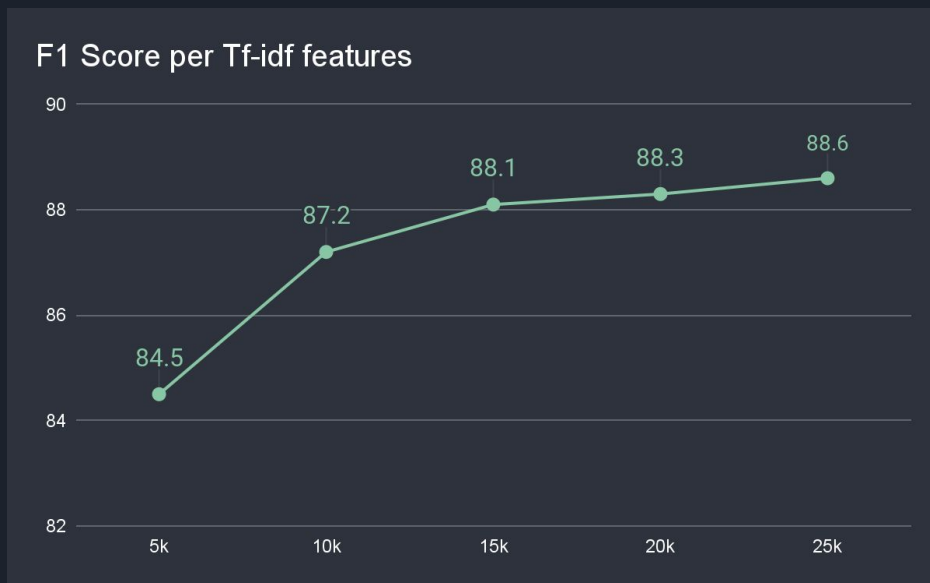
- Best performing dataset:
- Split Contraction, Lemmatization, Without Spelling, 25k features

Contractions expanded	150,492
-----------------------	---------

URL's	121
HTML tags	187,484
Other characters	1,822,452
Digits	150,344

Results

- Best TF-IDF features - 25000



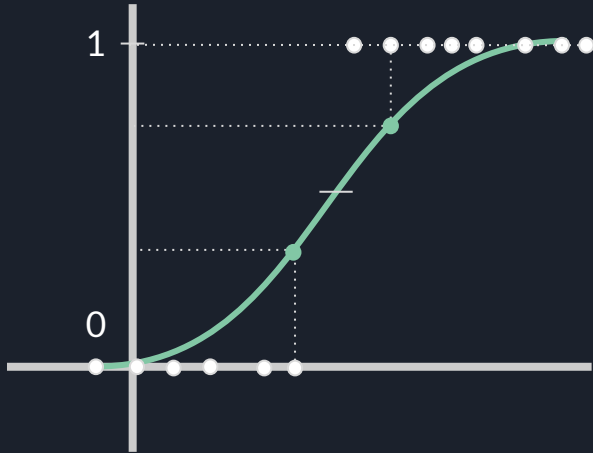


Machine learning models

- Logistic Regression
- Random Forest Tree
- XGBoost

Machine learning models

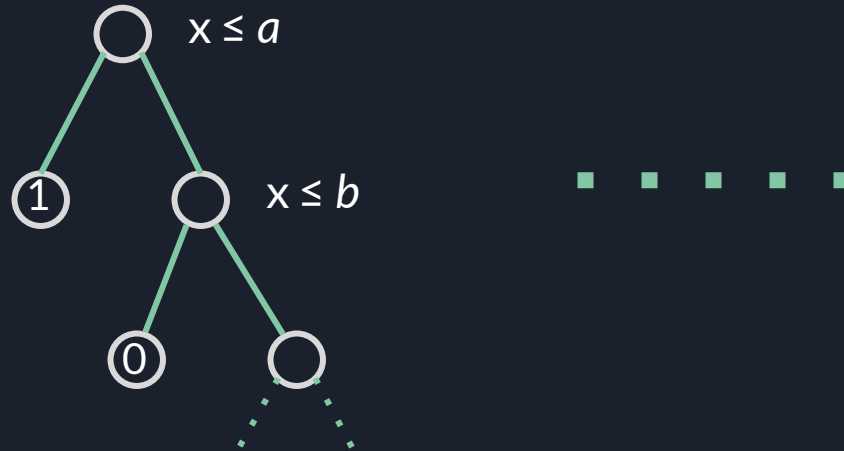
- Logistic Regression



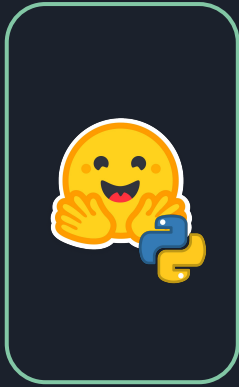
$$f(z) = \frac{1}{1 + e^{-z}}$$

Machine learning models

- Random Forest



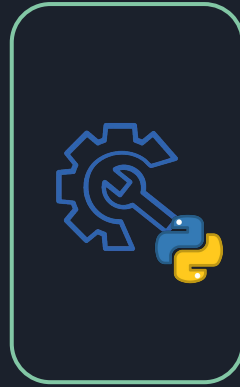
Deep Learning approach



Load pre-trained
Bert from Hugging Face

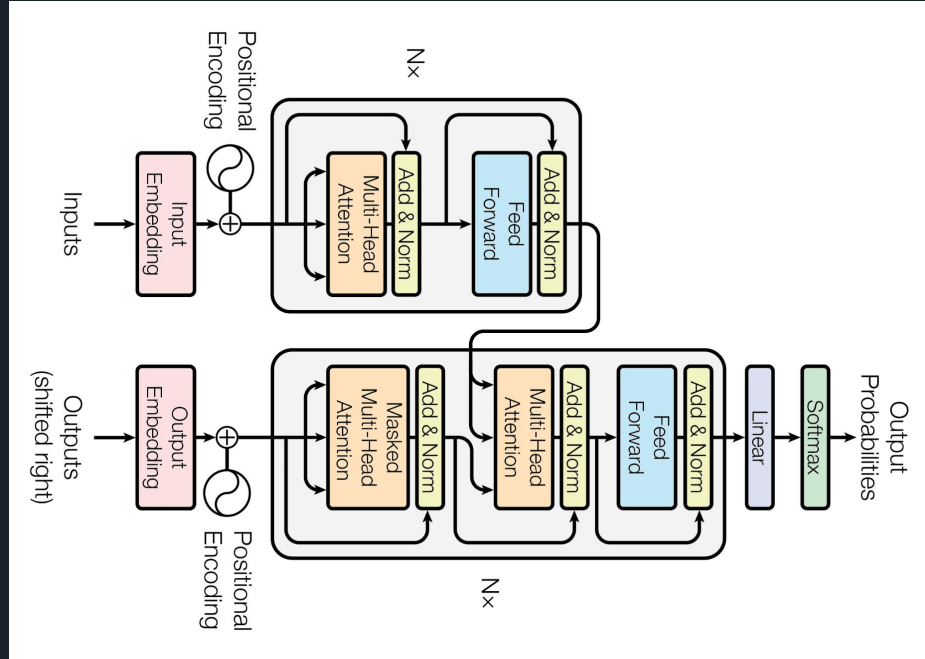


Tokenize data using
BertTokenizer



Fine tune model
on tokenized dataset

Deep Learning model - Transformer





Deep Learning model - BERT

- Bidirectional Encoder Representation from Transformers
- Pretraining - Masked language modeling
 - Next sentence prediction



Results

- Falsely classified reviews



I don't think this movie is good!



**NEGATIVE
COMMENT**



Further improvements

- More data preprocessing (:) -> 'good', '9/10' -> 'good', *balancing?*)
- Different ML Classifier
- Different Transformer

Thank you for your
attention!

