Master SIF - Module SML

Movie Review classification

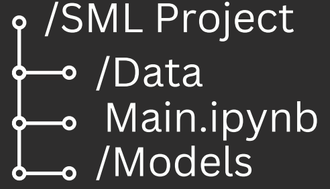
Le Marre Thibaut - Ravon Kilian - Spatha Mirto

Summary:

1. Preprocessing

Multiples questions ?

- Setting up the working environment:
  - We have chosen to work on GoogleCollab platform with this environment organisation.



- Where is the data coming from ?
  - [The] data contains 1000 positive and 1000 negative reviews all written before 2002, with a cap of 20 reviews per author (312 authors total) per category.
  - The correct label [has been] ex-tracted automatically from rating information (e.g., number of stars).

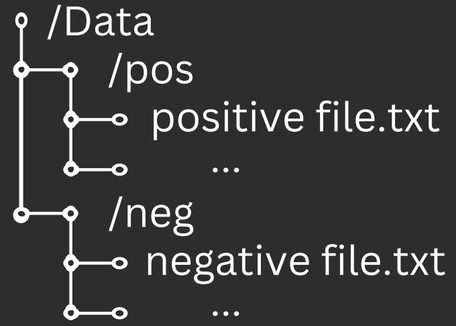
  [Bo Pang and Lillian Lee, ACL 2004]

We already are losing information:

- Started from [0-5] Stars --> [pos/neg] reviews
  - Make it harder to predict a 3 star score because it's neither positive / negative.

Multiples questions ?

- What is it format of the data ?
  - Arrange the data for easier future manipulation.



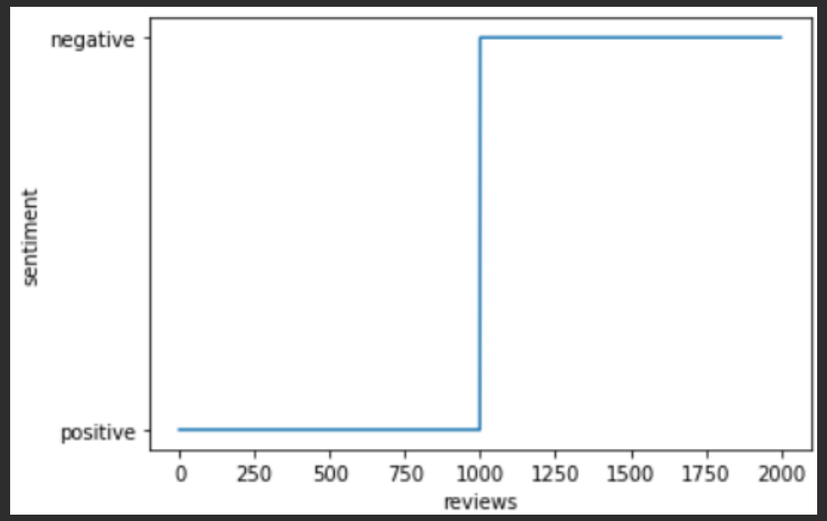
	review	sentiment
0	the american action film has been slowly drown...	positive
1	the ultimate match up between good and evil , ...	positive

Cleaning the dataset :

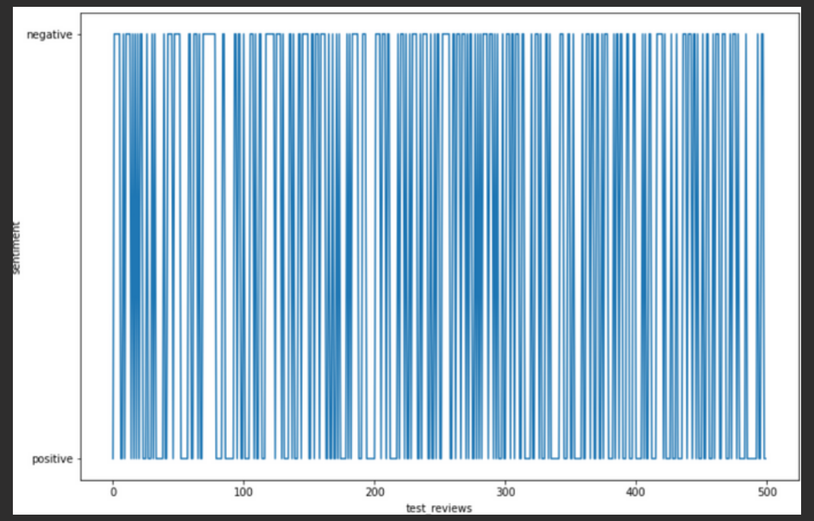
- Cleaning (html, non alphabet, stop words, urls, lower case).
- Word Spelling Correction
- Word Normalization (Stemming and Lemmatization)
- Encode Sentiments

Spliting the Datased

- We decided to split the data into 75% of Training and 25% of Testing set using train_test_split method from Scikit-Learn.



It's randomly shuffled



Preparing the text Data

Our computer can't do classification based on our raw data. It needs to be able to understand it and so we need to vectorize it.

Vectorizing Methods

- Words Counts with CountVectorizer (scikit-learn)
  - We will tokenize the documents and form a vocabulary with it. Then we will use the vocabulary to encode new documents but we remember the number of occurence of each word.
  - Word counts are pretty basic. It'll prioritize words that have no meaning : stops words ...

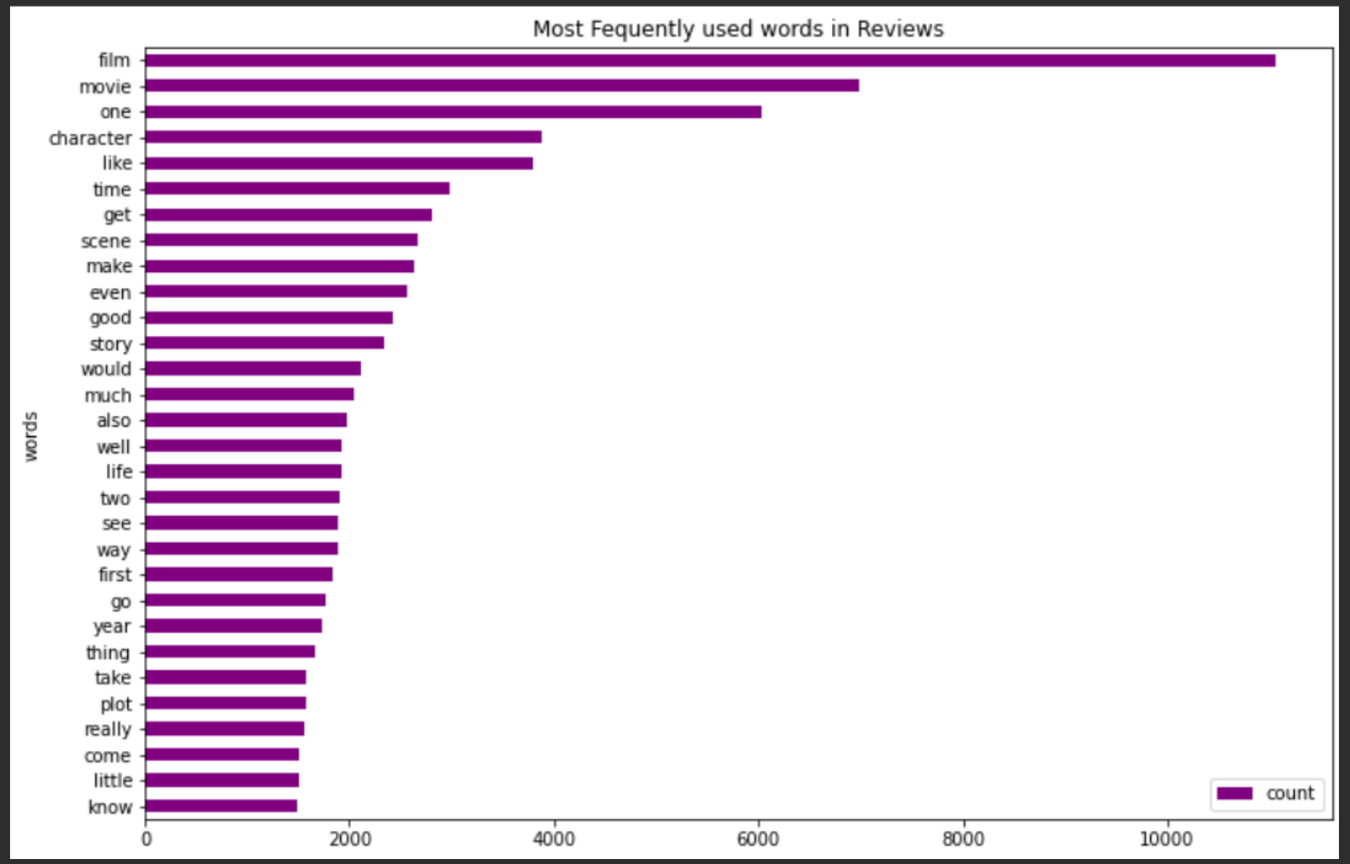
- Word Frequencies with TfidfVectorizer (scikit-learn)
  - It's words frequency scores that try to highlight words that are more interesting, e.g frequent in a review but not across reviews.

We implemented our model and tested them with both vectorized representation to see if the tfidf is really better or not.

2. Data Analysis

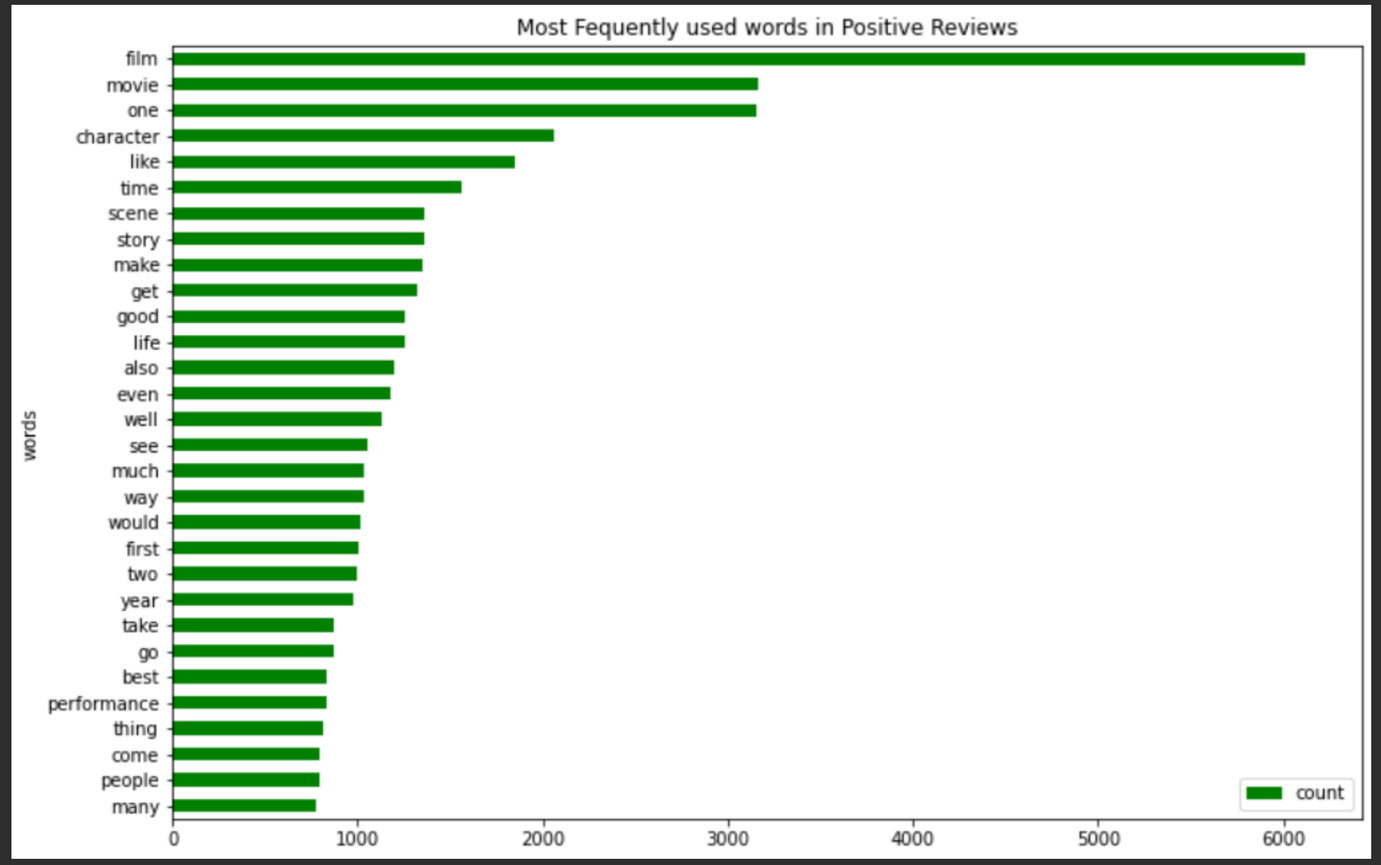
Frequent words

- We observe that the biggest amount of those do not have a sentimental meaning (24 neutral words, 6 sentimental words).



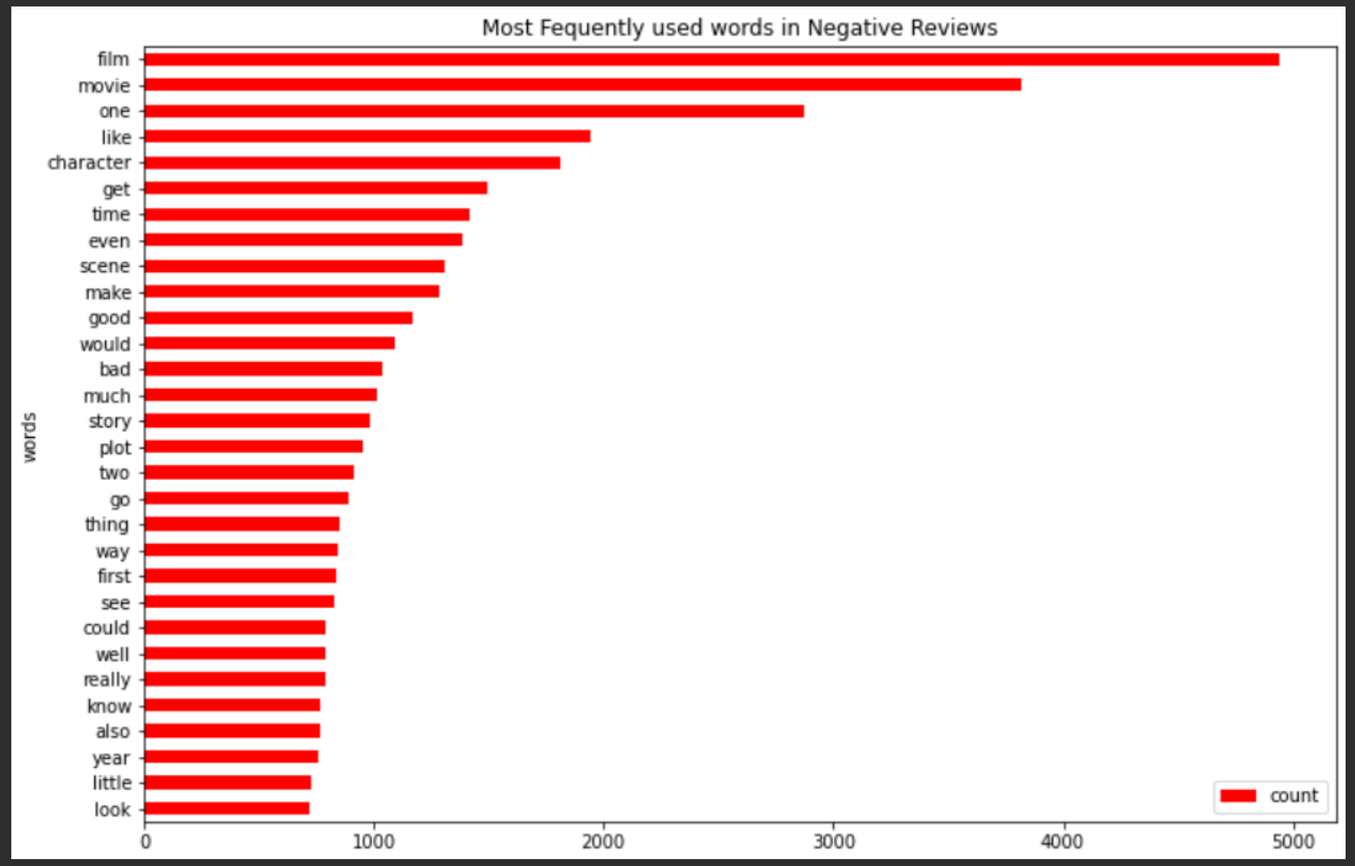
Frequent words

- Positive Reviews
  - We observe that the biggest amount of those do not have a sentimental meaning (24 neutral words, 6 sentimental words).
  - We also observe that the word 'best' for example, which has a positive meaning, appears frequently only in positive reviews, as expected.



Frequent words

- Negative Reviews
  - We observe that the biggest amount of those do not have a sentimental meaning (22 neutral words, 8 sentimental words).
  - We also observe that the word 'good' for example, which has a positive meaning, appears almost 1000 times in negative reviews, some times less that in positive reviews.
  - Also, the word 'little', which has a negative meaning, appears only in negative reviews, as expected.



3.Statistical models

Multinomial Naive Bayes

- Without Smoothing
  - With Count Vectorizer
  - With Tf-Idf Vectorizer
- With Smoothing
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

Logistic Regression

- Linear logistic Regression
  - With Count Vectorizer
  - With Tf-Idf Vectorizer
- Polynomial logistic Regression
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

SVM

- Poly Kernel
  - With Count Vectorizer
  - With Tf-Idf Vectorizer
- Linear Kernel
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

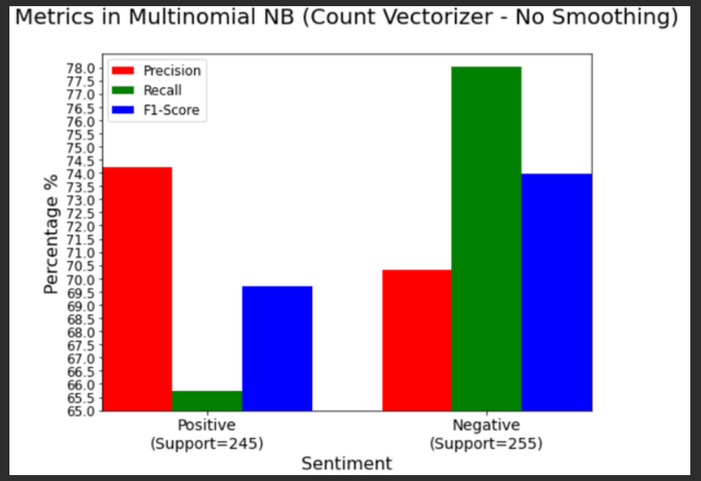
Multinomial Naive Bayes

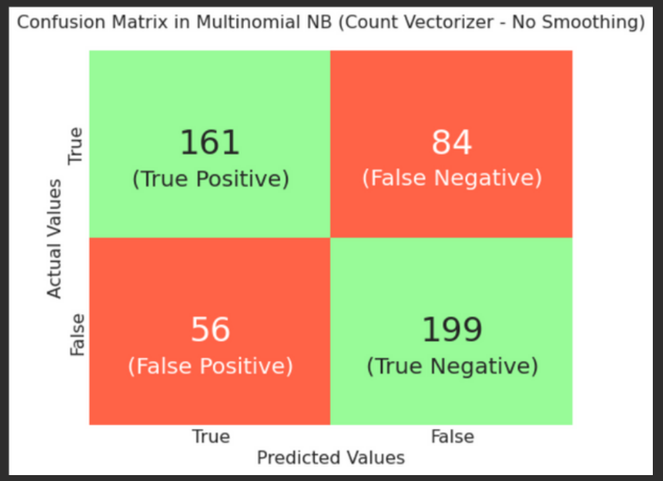
- Multinomial Naive Bayes Without Smoothing
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

Smoothing

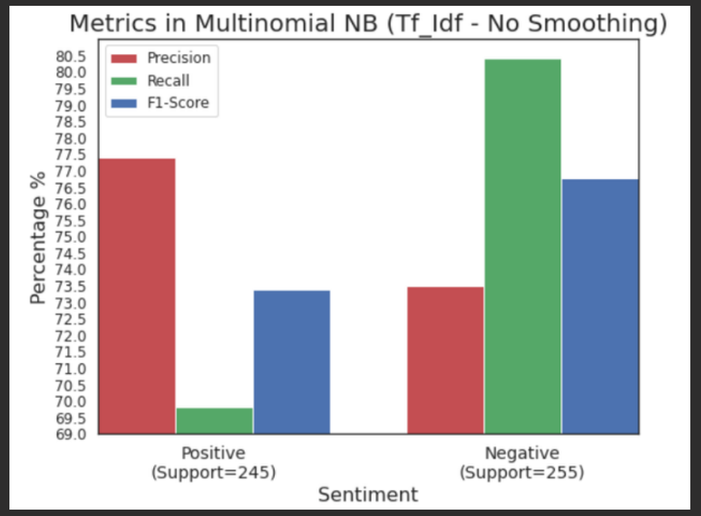
We used Laplacian Smoothing to eliminate the Zero Frequency Error (Occurs when the model tries to classify an unrecognizable word, resulting in a 0% probability score for this word), by increasing the word count by 1 for both classes.

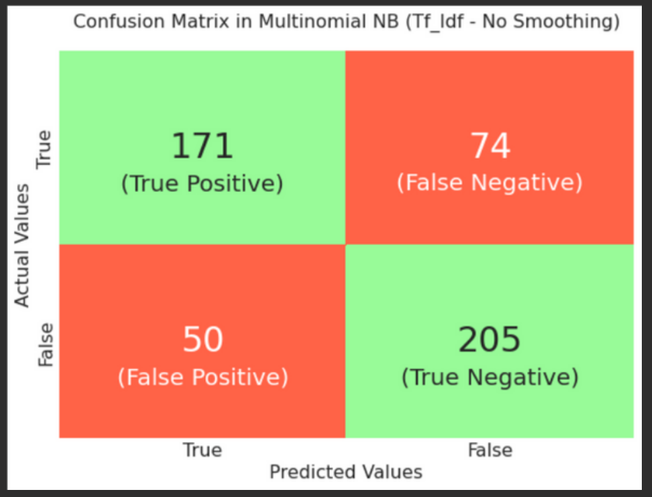
accuracy: 0.72





accuracy: 0.752

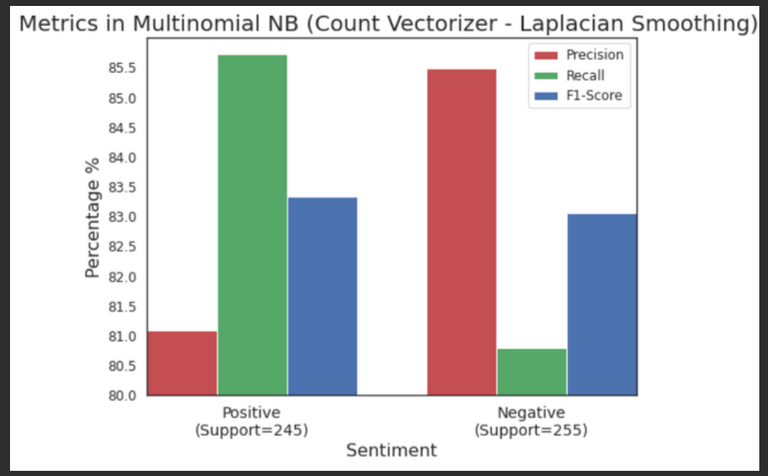


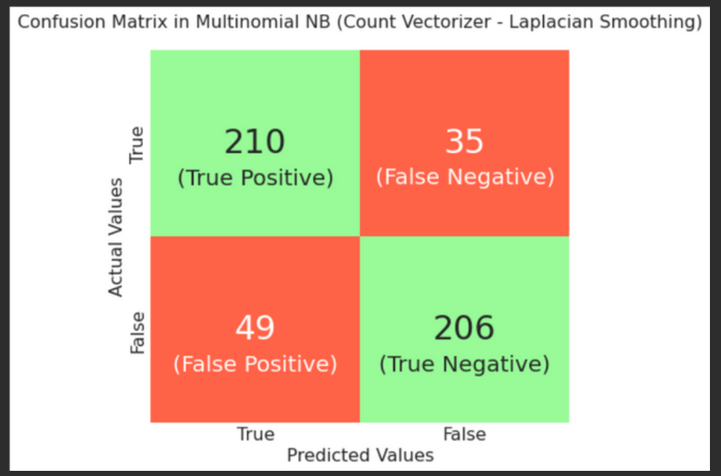


Multinomial Naive Bayes

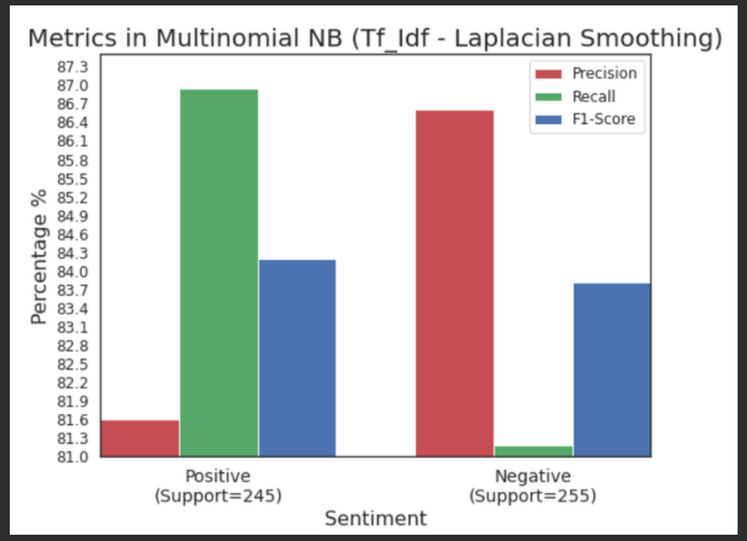
- Multinomial Naive Bayes With Smoothing
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

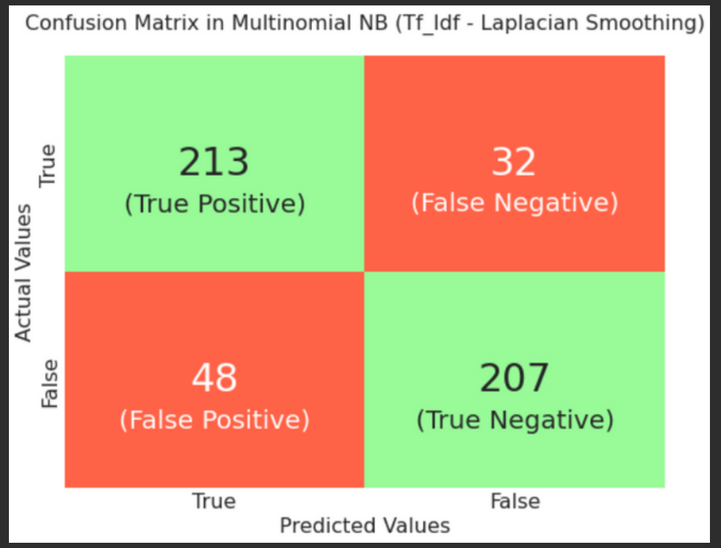
accuracy: 0.832





accuracy: 0.84





We observe that the number of the false negative reviews is reduced by more than a half.

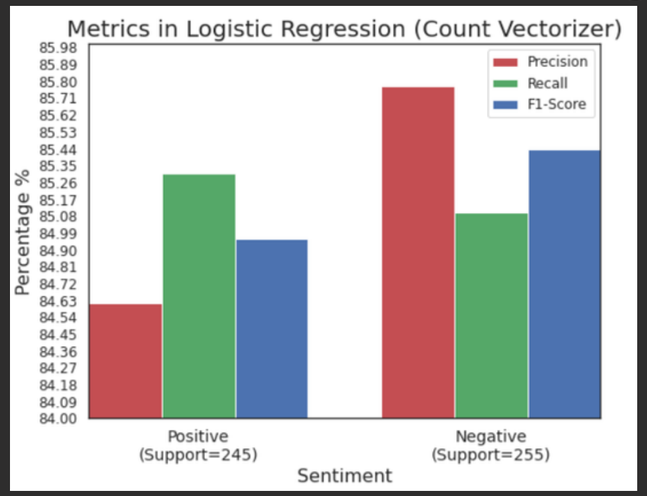
LR

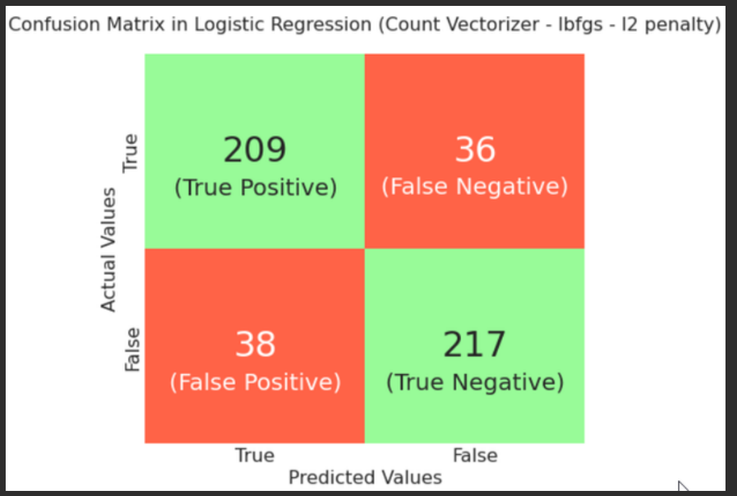
- LR with l2 penalty
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

l2 penalty

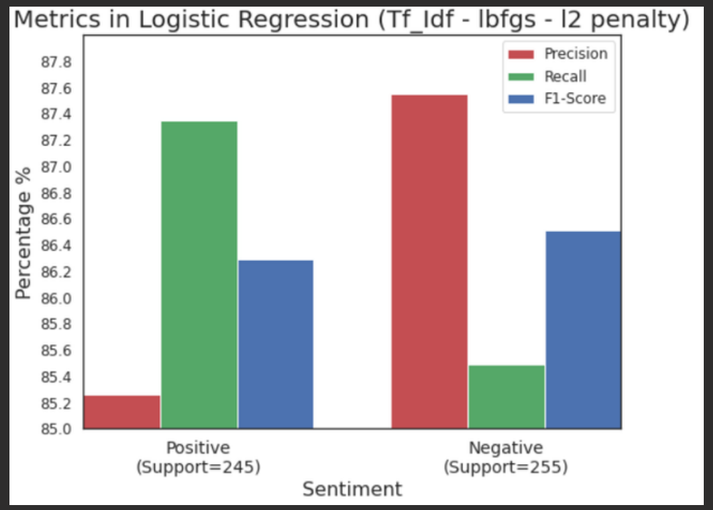
Penalty given to more complex models in order to avoid overfitting, thus reducing the classification score. The penalty equals the square of the magnitude of regression coefficients

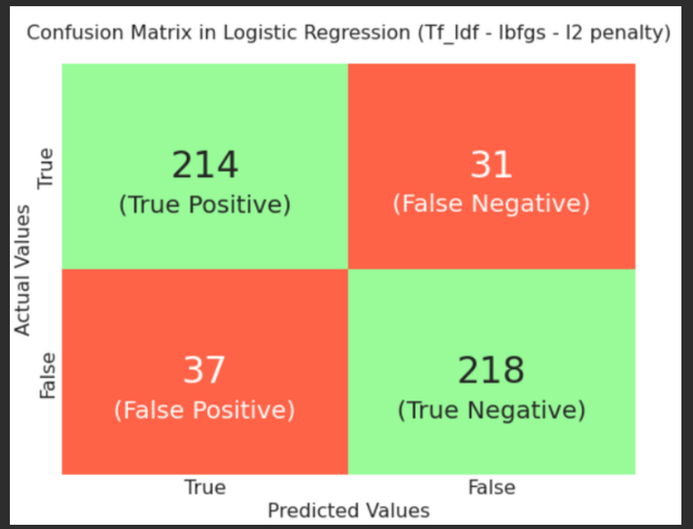
accuracy: 0.852





accuracy: 0.864





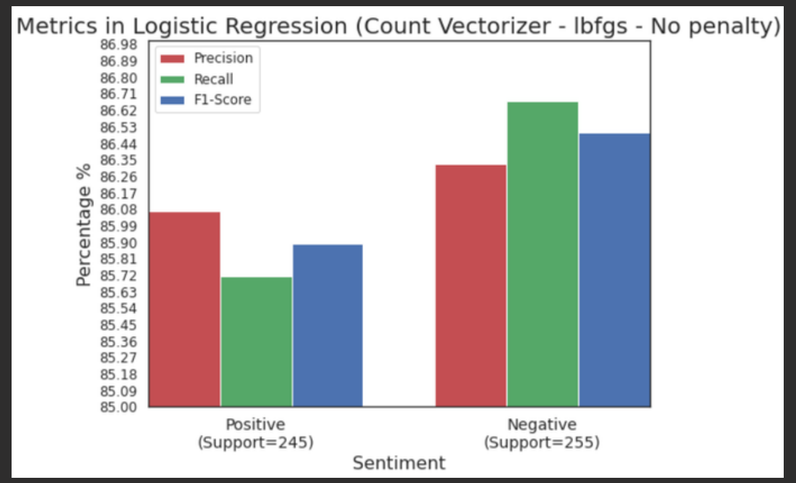
LR

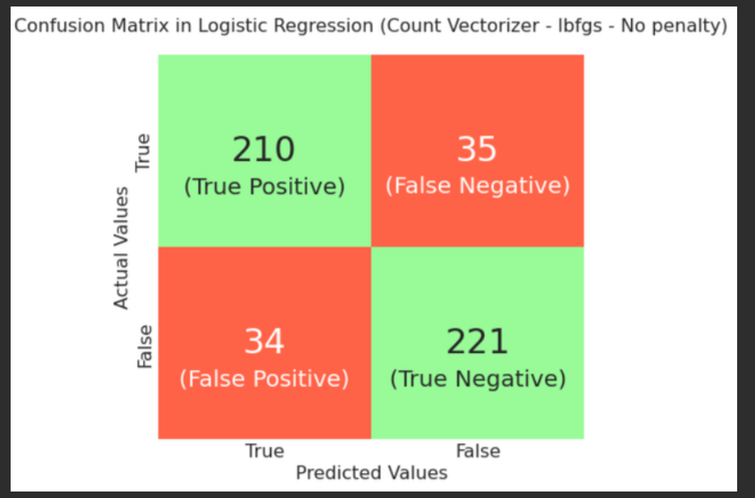
- LR without penalty
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

What's no penalty

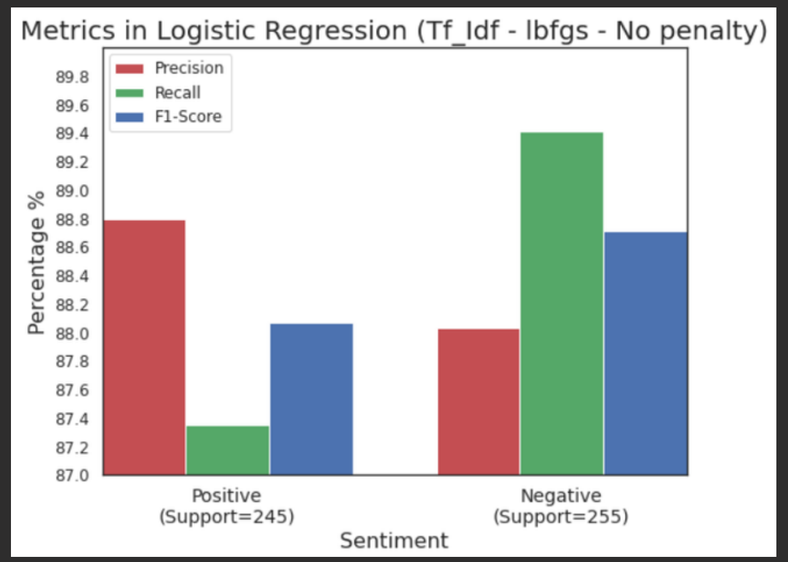
There is no regularization to avoid overfitting.
The predicted model is the one which gives the best clssification score.

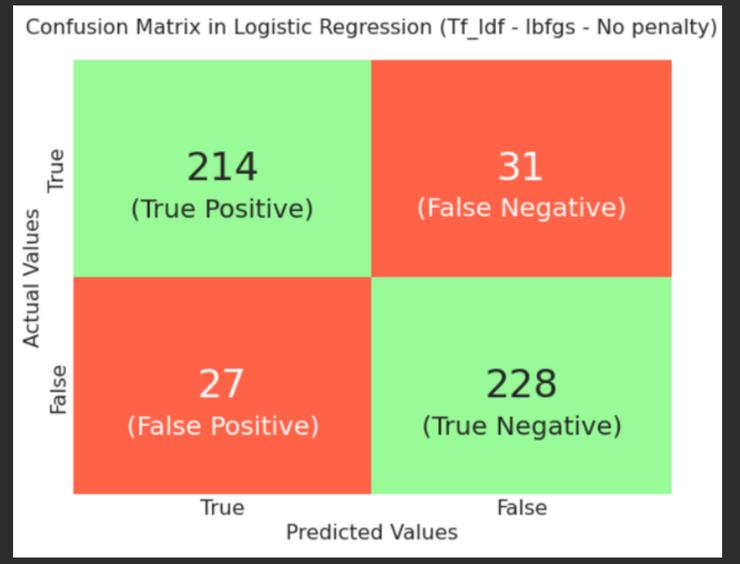
accuracy: 0.862





accuracy: 0.884





We observe that the classes that have the best recall and precision percentage are inverted.

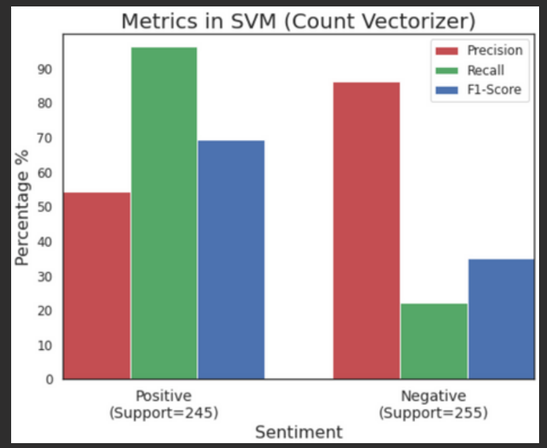
SVM

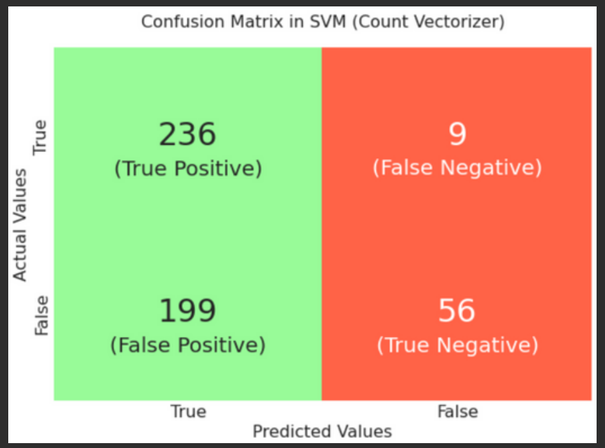
- SMV with poly kernel
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

poly kernel

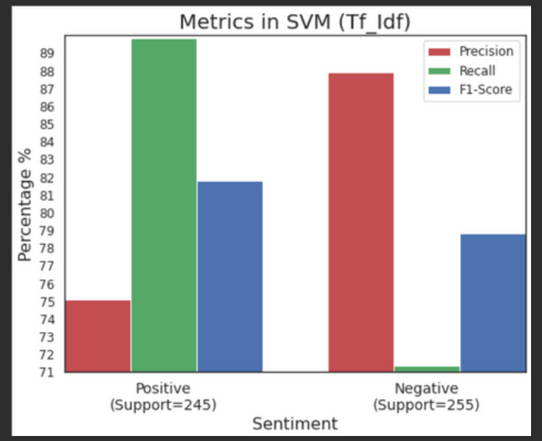
It is used for learning of non-linear models. It finds similarities between the input features and their combinations.

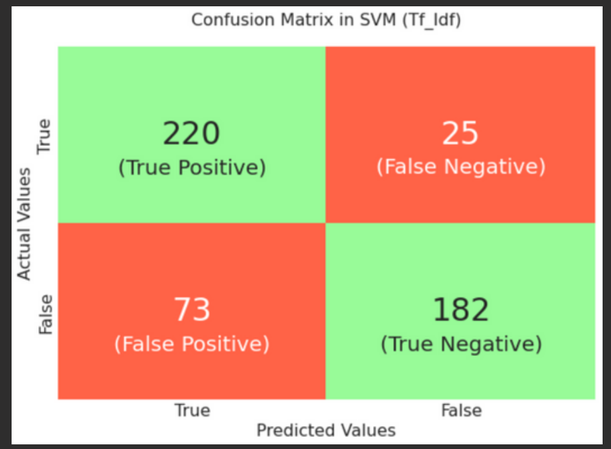
accuracy: 0.584





accuracy: 0.804





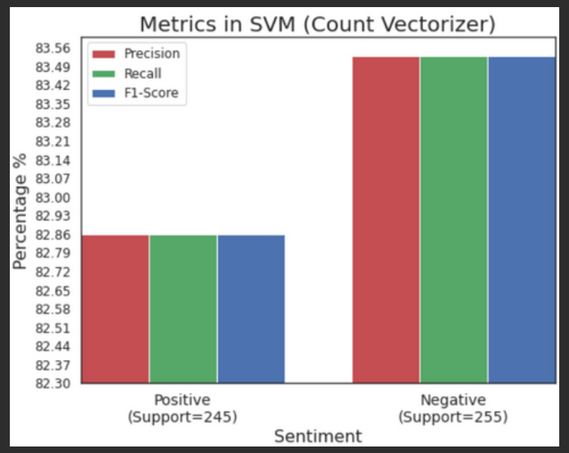
SVM

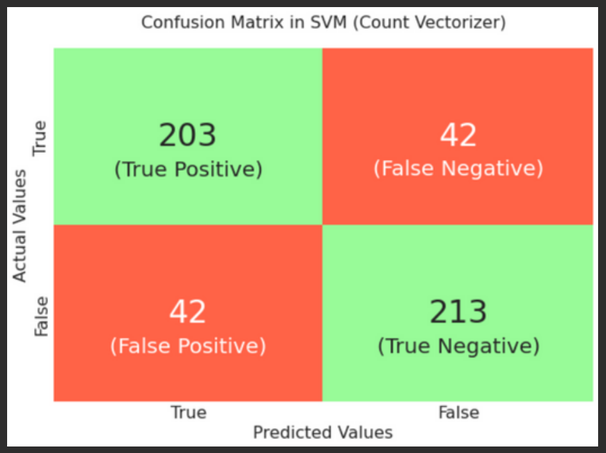
- SMV with linear kernel
  - With Count Vectorizer
  - With Tf-Idf Vectorizer

linear kernel

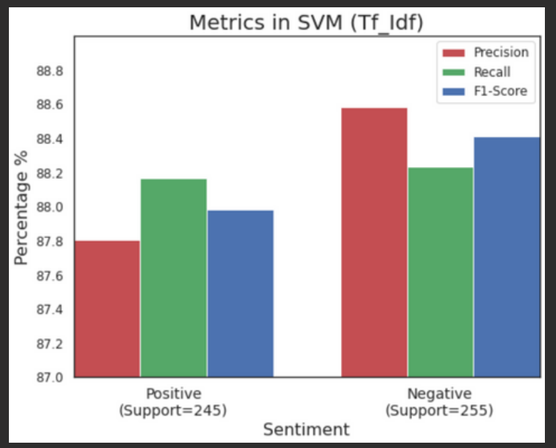
It is used when data is linearly separable.

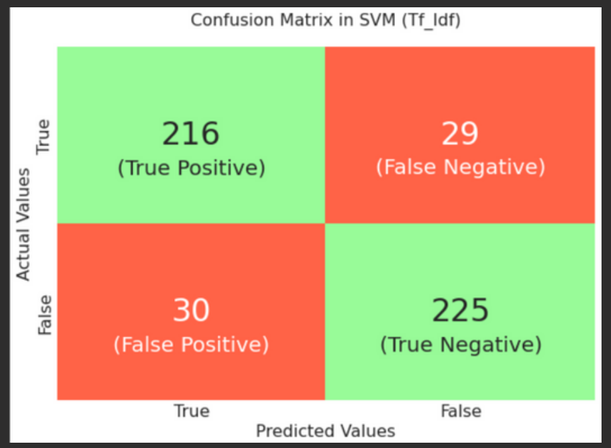
accuracy: 0.832





accuracy: 0.882





We observe that that the metrics are counterbalanced.

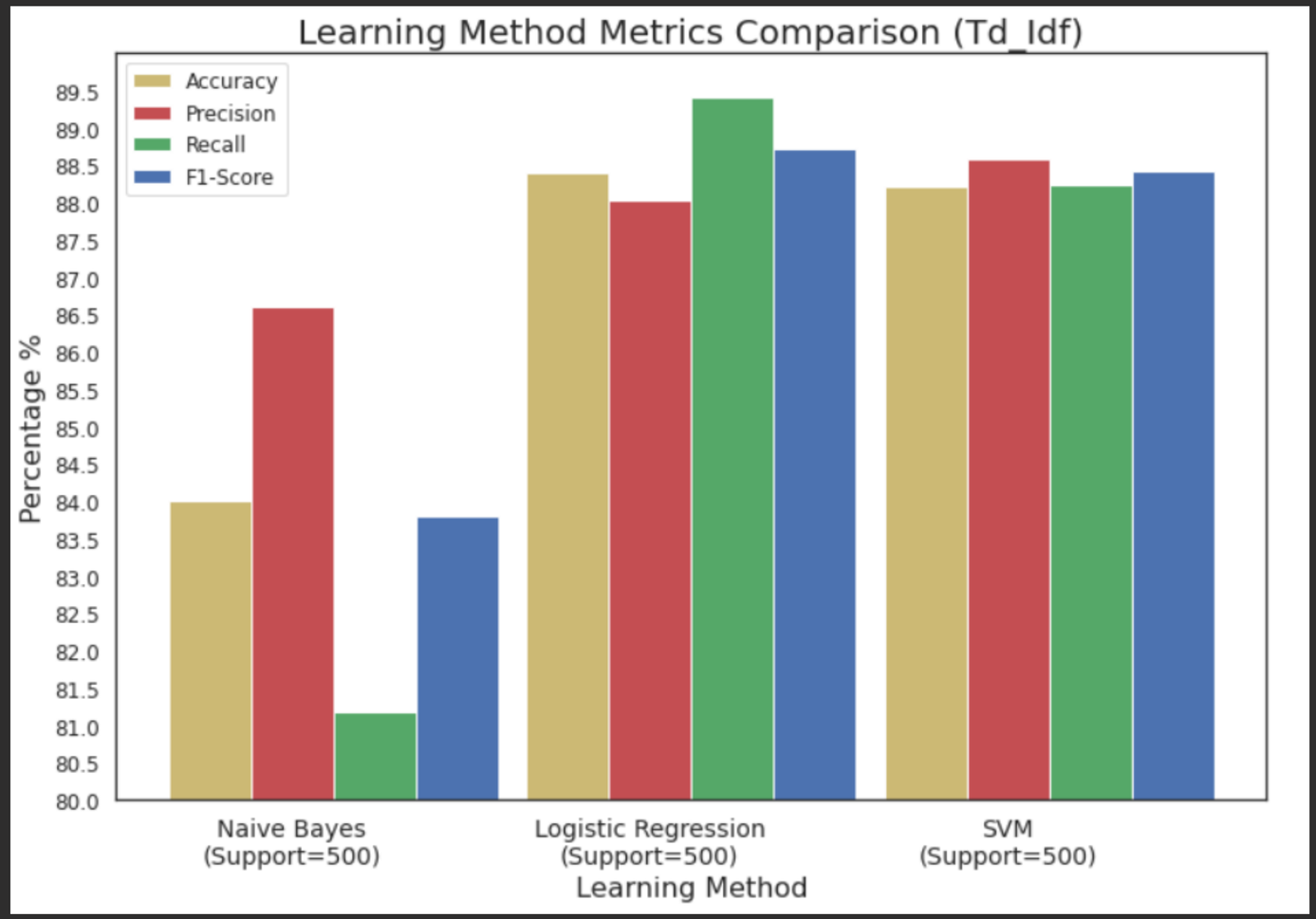
4. Results Comparison

Best Implementation:

- Naive Bayes: Tf_Idf Vectorizer
  (Laplacian Smoothing)
- Logistic Regression: Td_Idf Vectorizer
  (lbfgs solver, no penalty)
- SVM: Td_Idf Vectorizer
  (linear kernel)

Comments:

The precision and recall metrics which are presented here, are equivalent to the precision and recall of the positive classification.



5. Deep Learning models

Long short-term Memory & Word Embedding

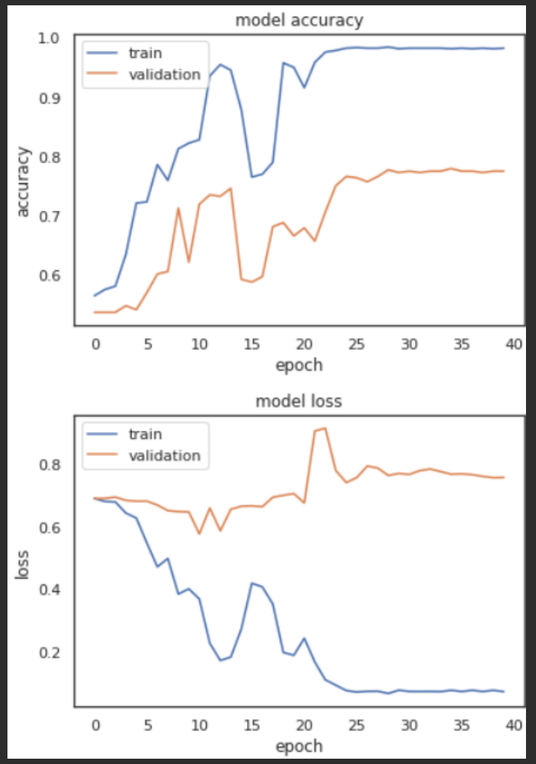
- We tried to implement a simple LSTM model to make those film review sentiements, based on a work done on IMBD Datasets that has high results values.

```
# ARCHITECTURE
EMBED_DIM = 32
LSTM_OUT = 64

model = Sequential()
model.add(Embedding(total_words, EMBED_DIM, input_length = max_length))
model.add(LSTM(LSTM_OUT, dropout=0.1))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

print(model.summary())
```



The best model we produced was done with 100 batch_size, 40 epochs and validation_split = 0.3 and has:

- 79% accuracy with 399 correct prediction and 101 wrong predictions

5. Conclusion

Our classification scores were significantly high (88% accuracy), after using some pre-processing techniques, while the best score from the techniques used by the proposed papers was 87% (SVM), even though their implementations and the training and test data were different than ours.

Further Research:

Our research can be used as a baseline for further investigation in multiple axes, such as:

- Data Analysis of falsely classified reviews (in terms of spelling, unrecognised words by the models and other data patterns).
- Testing with more review or general datasets (combine results).
- Omitting non sentimental words in the pre-processing step.

Thank you for your attention.

Questions ?