Sentiment Analysis on 

movie reviews

Emma Corbett, Nuneke Kwetey, Alan Wong, Xiqian Yuan, Anna Micros

The dataset used consists of 50,000 film reviews that are labelled as either positive or negative. The goal is to develop an optimal model that accurately predicts sentiment and identifies patterns in review content.

Dataset: 50,000 rows, 2 columns

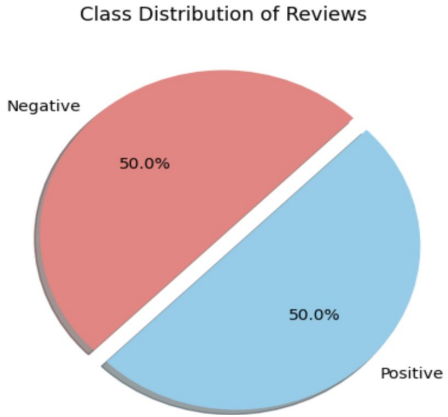
- Column 1: Text review
- Column 2: Sentiment label (positive or negative)

(50000, 2)

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

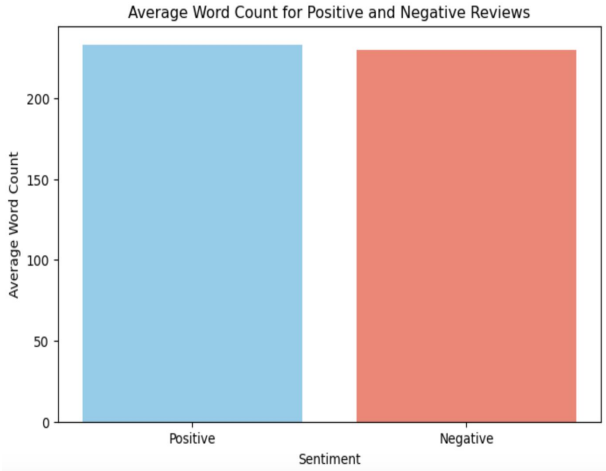
Class Distribution: Evenly distributed

- Positive reviews: 25,000 (50%)
- Negative reviews: 25,000 (50%)



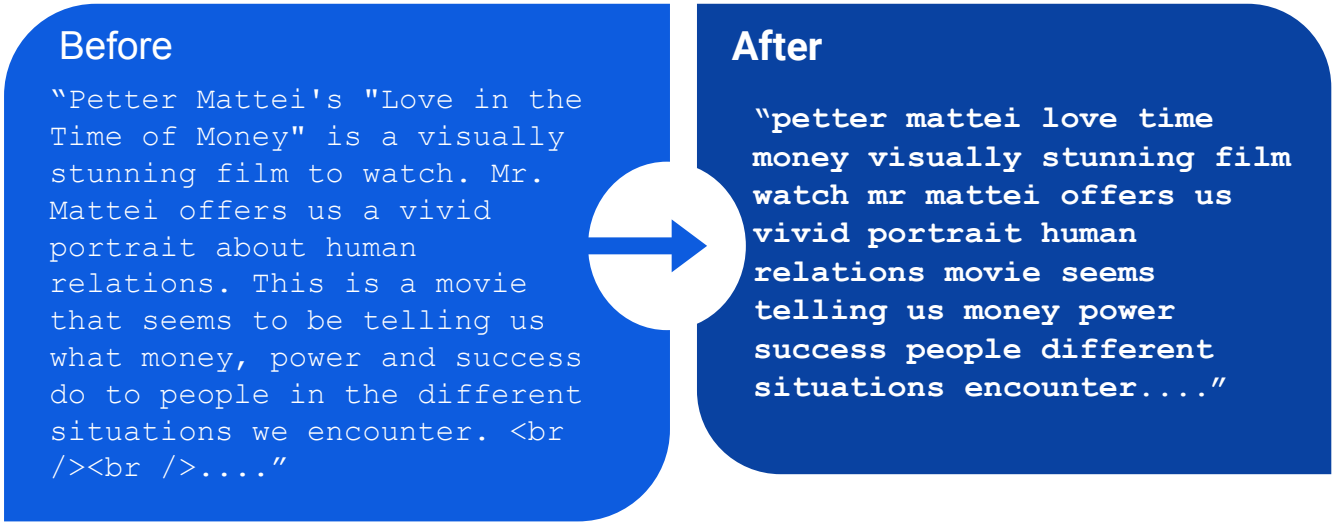
Average Word Count:

- Positive reviews: 233 words (approx.)
- Negative reviews: 230 words (approx.)



Data Cleaning:

- **Regex**: Removed special characters (punctuation marks, mathematical symbols, etc), URLs, and HTML tags.
- **Removal of Filler Words**: Used NLTK stopwords to eliminate common filler words (e.g., "the", "and", "is").
- **Text Transformation**: Converted all text to lowercase for uniformity.
- **Additional Scrubbing**: Removed extra white spaces and non-alphanumeric characters.



Tokenization is the process of separating text into smaller units called **tokens** after cleaning/scrubbing of the text. For our purposes, we chose each **word** to be a token. To mark the beginning or end of a review, we pad each review with 'START' and 'STOP' tokens. Below illustrates an example of how we tokenize each review:

Original Sentence	"pure gothic vampire cheese nothing compare subspecies vampire cheese films highly recommend every one"
Unigram	'pure' 'gothic' 'vampire' 'cheese' 'nothing' 'compare' 'subspecies' 'vampire' 'cheese' 'films' 'highly' 'recommend' 'every' 'one' 'STOP'
Bigram	'START' 'pure' 'gothic' 'vampire' 'cheese' 'nothing' 'compare' 'subspecies' 'vampire' 'cheese' 'films' 'highly' 'recommend' 'every' 'one' 'STOP'
Trigram	'START' 'START' 'pure' 'gothic' 'vampire' 'cheese' 'nothing' 'compare' 'subspecies' 'vampire' 'cheese' 'films' 'highly' 'recommend' 'every' 'one' 'STOP'

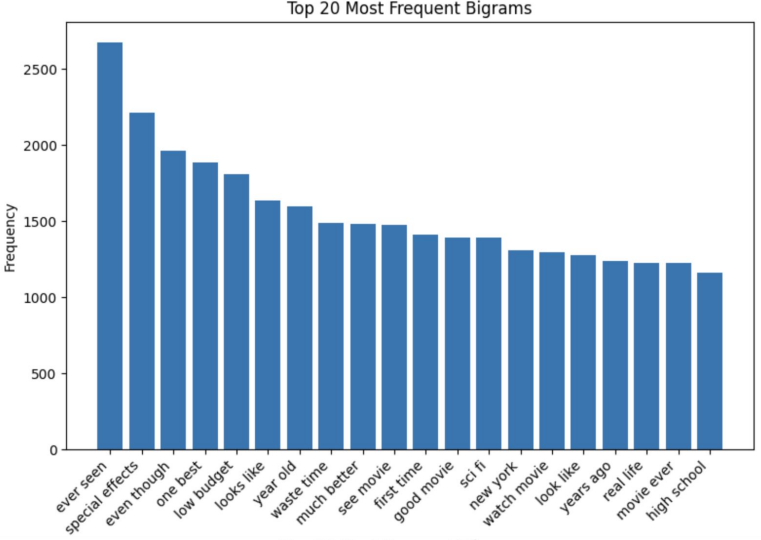
Unigrams, count:
{ 'pure': 1,
'gothic': 1,
'vampire': 2,
'cheese': 2,
….
'one': 1 }

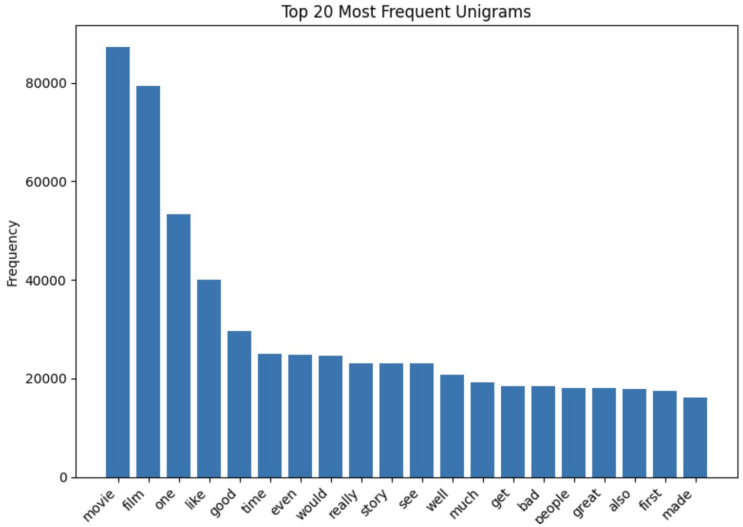
Bigrams, count:
{ 'START pure': 1,
'pure gothic': 1,
'gothic vampire': 1,
'vampire cheese': 2,
…
'every one': 1 }

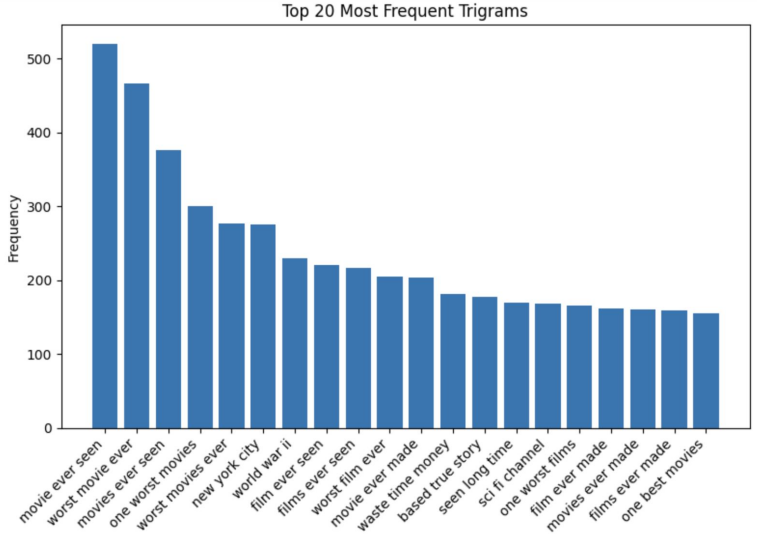
Trigrams, count:
{ 'START START pure': 1,
'START pure gothic': 1,
'pure gothic vampire': 1,
'gothic vampire cheese': 1,
…
'recommend every one': 1}

Unigram, bigram, and trigram distribution:

- Figures show the top 20 unigrams, bigrams, and trigrams in the IMDB reviews corpus
- Follow Zipf's law
- For bigrams/trigrams, a lot of common '**low entropy**' phrases
  - In the context of LLMs, '**low entropy**' phrases are when the next word is highly predictable based on the preceding context. For example, that 'city' would follow 'new york'.
- Many of these phrases also look like they would be common in movie reviews (ex: 'worst movie ever')







Word Clouds

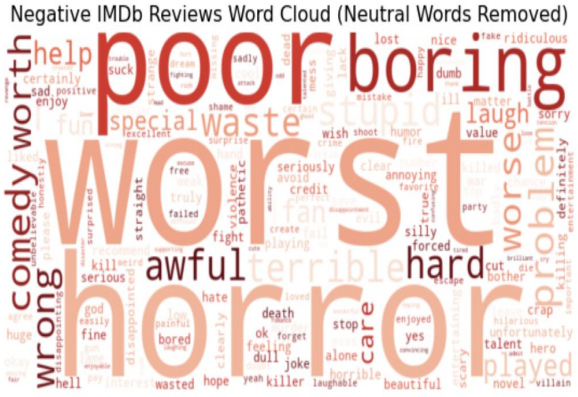
- Once data cleaning is complete, words like "**film**", "**one**", and "**movie**" dominate the word cloud due to their high frequency in movie reviews.
- They are neutral and do not reflect sentiment, so filtering out neutral words can highlight more meaningful terms.

Positive Reviews:

- After removing all neutral words from the positive reviews, we see that that word cloud emphasizes enjoyment and personal connections through words like “**fun**” and “**excellent**.”
- This highlights the emotional and experiential aspects as well as the overall satisfaction of the films.

Negative Reviews:

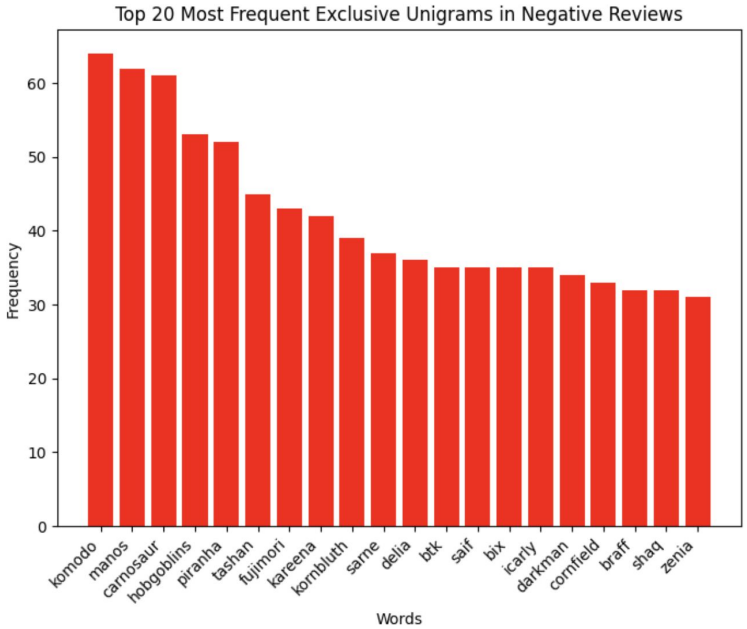
- After removing all neutral words from the negative reviews, we see that that word cloud emphasizes dissatisfaction and frustration, with words like "**worst** and" "**poor**.”
- This highlights the disappointment, poor quality, and overall dissatisfaction with the films.

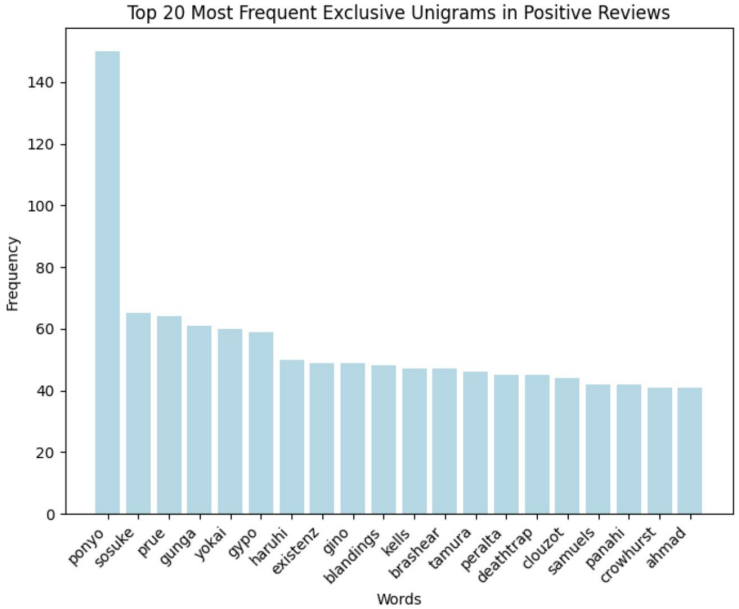


Additional Insights

We also took a look at the exclusive set of words with positive and negative reviews.

Instead of filtering out neutral words as before, we just took a look at all words found in only positive reviews, and only negative reviews.



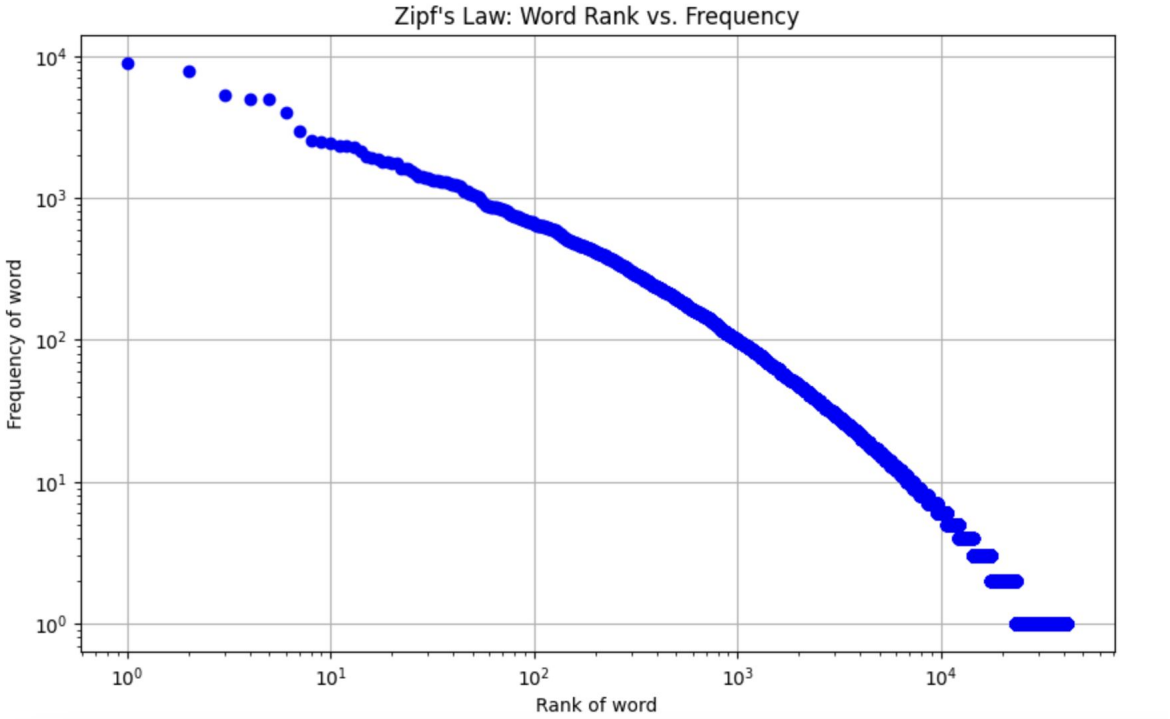


Interestingly, the most frequent words for both positive and negative reviews were movie names (“ponyo” for positive & “komodo” for negative) and characters from movies with good and bad ratings respectively.

Also we see that even here, the word frequencies follow Zipf’s Law.

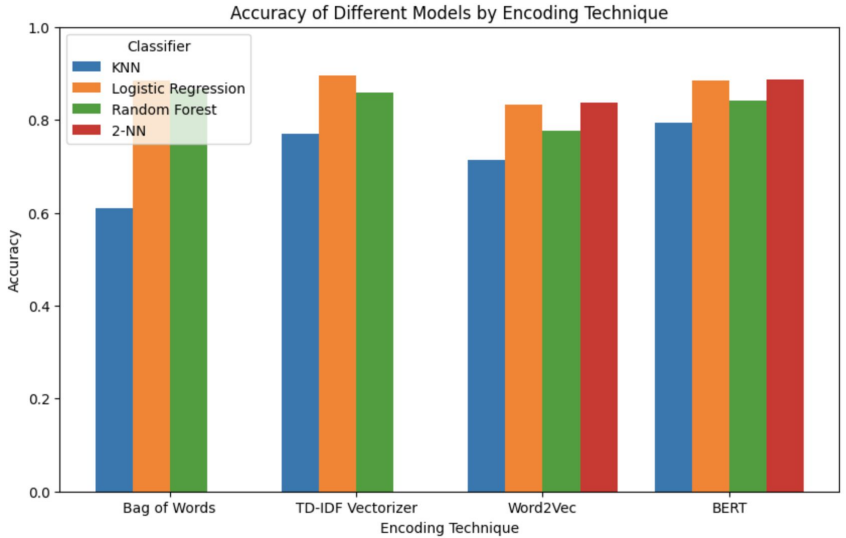
Further Insights from data exploration: Zipf's Law

- In NLP, word frequency distribution typically follows **Zipf's law**
- Meaning the **frequency** of a word is inversely proportional to its **rank** in a frequency table
- We verified this in the IMDB reviews dataset by plotting on a loglog scale the frequency of a unigram vs its rank
- The graph **confirms** that the words in our dataset follow a typical language distribution

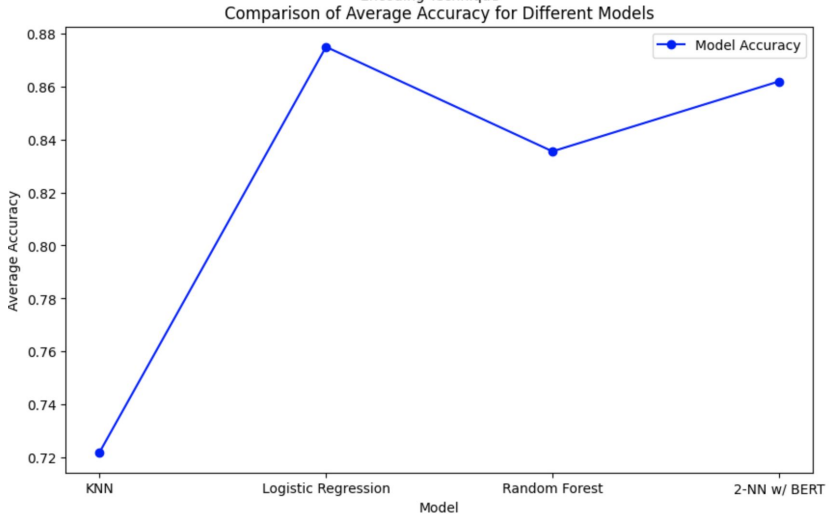


Encoding Text Data

- In order to train our ML models, we need a encode our text data.
- We tried out 4 methods:
  - Bag of words (Count Vectorization)
  - TD-IDF Vectorization
  - Word2Vec Embeddings
  - BERT Embeddings



- Preliminary Exploration on (KNN, Logistic Regression, Random Forest and 2-layer Neural Network) shows that **TD-IDF Vectorization** on **Logistic Regression** has highest accuracy (**89.54%**)
- It also shows that **Logistic Regression** is the model with the highest average accuracy over 4 embedding methods. (**87.5%**) and **KNN** has the lowest average accuracy (**72.0%**)



Machine Learning Techniques

Logistic Regression

Preliminary Results

We run a default logistic regression with text represented using TD-IDF Vectorization. We used an 80:20 train-test split. As a baseline (ie, pre hyperparameter tuning) the accuracy on the test set is 0.8965.

	precision	recall	f1-score
negative	0.91	0.88	0.89
positive	0.89	0.91	0.90
accuracy	0.90	0.90	0.90
macro avg	0.90	0.90	0.90
weighted avg	0.90	0.90	0.90

Hyperparameters to tune

- Ridge (L2 Regularization)
- Lasso (L1 Regularization)

K-Nearest Neighbor (KNN)

Preliminary Results

We run a KNN with k=3 with text represented using BERT Encoding. We used an 80:20 train-test split. As a baseline, the accuracy on the test set is 0.7942.

	precision	recall	f1-score
0.0	0.75	0.87	0.81
1.0	0.85	0.72	0.78
accuracy	0.79	0.79	0.79
macro avg	0.80	0.79	0.79
weighted avg	0.80	0.79	0.79

Hyperparameters to tune

- Number of Neighbors (k)
- Distance Metric (Euclidean/ Manhattan/ cosine)

More Machine Learning Techniques

Tree Based Model [Random Forests/ Gradient Boosted Trees]

Preliminary Results

We run a default random forest classifier with text represented using Bag of words. We used an 80:20 train-test split. As a baseline, the accuracy on the test set is 0.8657.

	precision	recall	f1-score
negative	0.85	0.86	0.86
positive	0.86	0.85	0.86
accuracy	0.86	0.86	0.86
macro avg	0.86	0.86	0.86
weighted avg	0.86	0.86	0.86

Hyperparameters to tune

- Number of Estimators
- Max depths
- Learning Rate (for Gradient Boosted Trees)

Deep Neural Network Model with BERT

Preliminary Results

As a baseline, we run a 2-layer Neural Network. The first layer with 128 nodes, second layer with 64 nodes and output layer with 1 node. The activation function is relu, sigmoid for the final layer, and optimizer is ADAM. Also used BERT Encoding and trained model for 20 epochs. We used an 80:20 train-test split. The accuracy on the test set is 0.8876.

	precision	recall	f1-score
0.0	0.87	0.90	0.89
1.0	0.90	0.87	0.89
accuracy	0.89	0.89	0.89
macro avg	0.89	0.89	0.89
weighted avg	0.89	0.89	0.89

Hyperparameters to tune

- Neural Network Architecture
- Number of Epochs
- Learning Rate
- Activation function
- Optimization function