Sentiment Analysis with IMDB dataset

Introduction:

1. Problem Statement

IMDB wants to deeply anaylze the movie review and figure out whether the review contains negative or positive sentiment.

2. Company Background

According to wikipedia, “IMDb is an online database of information related to films, television series, home videos, video games, and streaming content online – including cast, production crew and personal biographies, plot summaries, trivia, ratings, and fan and critical reviews.”

3. Goal

With a given dataset, create a robust machine learning model which classifies positive or negative movie review.

More reference:
https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Dataset

The ‘IMDB_Dataset.csv’ contains 50,000 observations of ‘review’ and ‘sentiment’ variables. ‘sentiment’ variable has been equally distributed. Each sentiment (positive and negative) has 25,000 observations.

Data Wranggling

1. Data Information

- Original dataset (IMDB_Dataset.csv) has 50,000 observations with 2 columns composed of 2 objects.
- Data doesn't have any missing value.

2. Dependent Variable (Target Variable)

A variable 'sentiment' is our dependent variable because we are going to classify positive or negative by analyzing each review text.

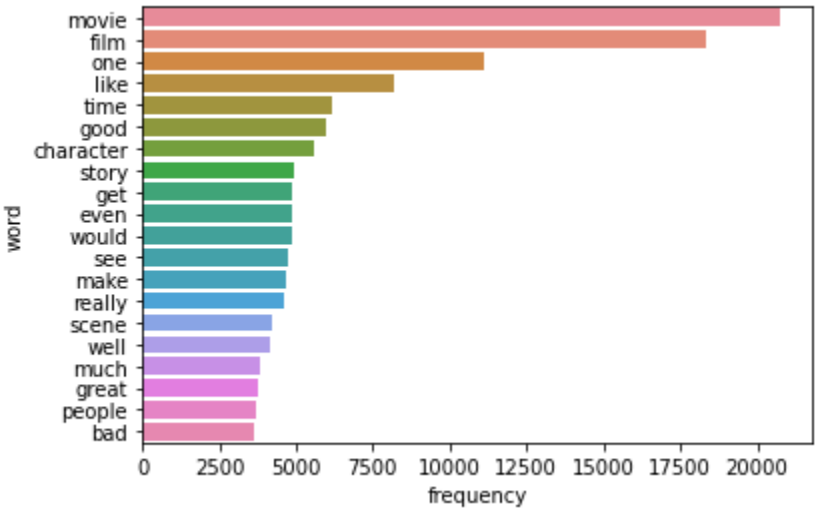
3. Independent Variables

A variable 'review' will be our independent variable.

Exploratory Data Analysis

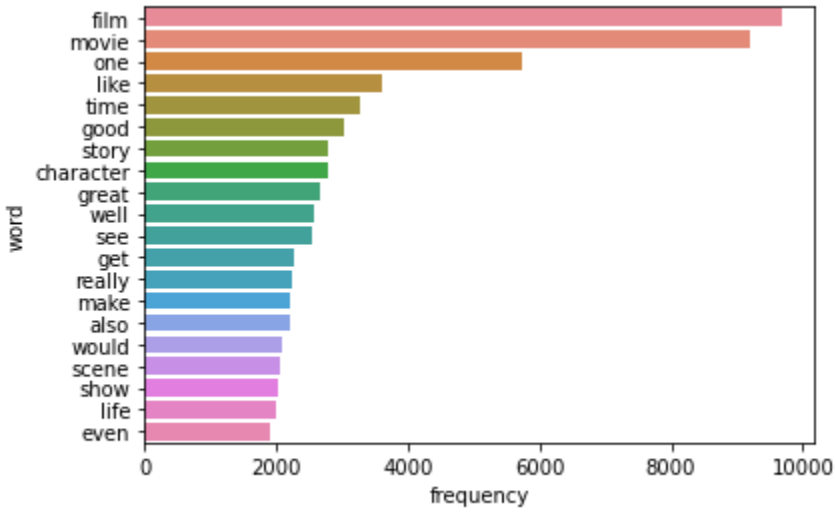
Below chart will show frequency analysis of top 20 words in descending order. I am going to present word frequency with both positive and negative reviews, only negative reviews, and only positive reviews.

1. Both positive and negative



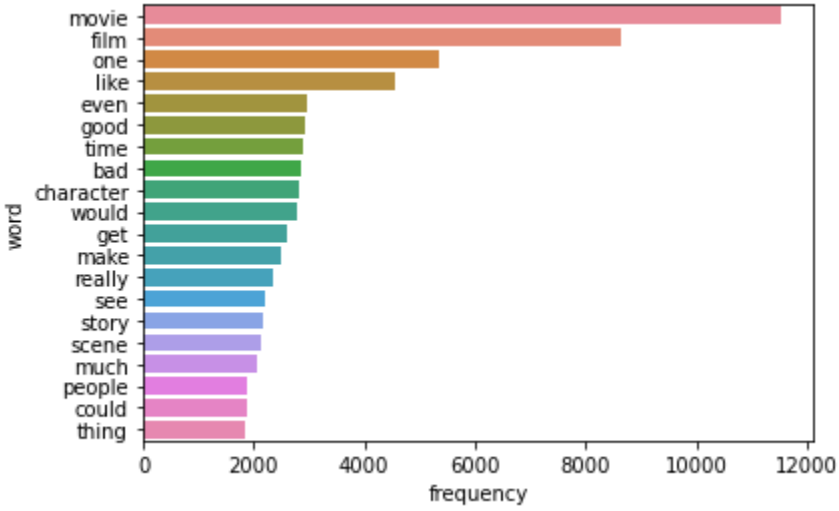
Top 2 mentioned words are 'movie' and 'film'. Since it includes both positive and negative reviews, chart presents both words 'good' and 'bad'.

2. Positive



Word ‘bad’ disappeared.

3. Negative



On the other hand, the frequency chart for only negative review still has word ‘good’. This means that even though sentiment of review is negative, many reviewers still use the word ‘good’.

Preprocessing

The very first preprocessing step I did was subsetting dataset. The whole observation is 50,000. Since it took so long to train the dataset, I initially subset 10,000 observation samples to train and test the model.

Secondly, I cleaned up the column ‘review’ before running models to classify sentiment of review. Below list presents what I handled to clean dataset.

1. Lower case
2. Delete punctuation except ‘.’,’?’,’!’
3. Remove URLs
4. Remove text ‘br’
5. Remove stopwords
6. Lemmatize word

To verify data cleaning step, I also ran the models without cleaning the dataset.

Model Evaluation

Below chart is the outcome of model evaluation.

1. Before Cleaning Dataset (Sample 10,000)
    a. CountVectorizer

	Cleaned	Model	F1 Score	Log Loss	Time
0	No	KNeighborsClassifier(n_neighbors=3)	0.628550	4.177094	3.111396
1	No	RandomForestClassifier()	0.840853	0.504325	8.261566
2	No	XGBClassifier(base_score=0.5, booster='gbtree'...	0.843028	0.361784	13.079273
3	No	AdaBoostClassifier()	0.798737	0.676570	3.318392
4	No	GradientBoostingClassifier()	0.806923	0.457975	11.578111
5	No	LogisticRegression()	0.871440	0.399793	0.855010
6	No	MultinomialNB()	0.831081	1.220445	0.018177

    b. TFIDF

	Cleaned	Model	F1 Score	Log Loss	Time
0	No	KNeighborsClassifier(n_neighbors=3)	0.699700	3.141768	3.095500
1	No	RandomForestClassifier()	0.824010	0.516154	7.220086
2	No	XGBClassifier(base_score=0.5, booster='gbtree'...	0.829249	0.368336	16.705167
3	No	AdaBoostClassifier()	0.793185	0.675091	6.342808
4	No	GradientBoostingClassifier()	0.802611	0.456218	27.769987
5	No	LogisticRegression()	0.884677	0.377822	0.520117
6	No	MultinomialNB()	0.860455	0.445545	0.015754

TFIDF with Logistic Regression shows the best outcome. F1 score is 0.884677.

2. After Cleaning Dataset (Sample 10,000)
    a. CountVecotizer

	Cleaned	Model	F1 Score	Log Loss	Time
0	Yes	KNeighborsClassifier(n_neighbors=3)	0.541550	5.235694	2.191641
1	Yes	RandomForestClassifier()	0.837475	0.481931	7.832799
2	Yes	XGBClassifier(base_score=0.5, booster='gbtree'...	0.835869	0.364079	5.006551
3	Yes	AdaBoostClassifier()	0.781693	0.676675	1.881088
4	Yes	GradientBoostingClassifier()	0.807058	0.460172	7.720532
5	Yes	LogisticRegression()	0.863876	0.400531	0.678967
6	Yes	MultinomialNB()	0.848939	1.021409	0.015253

    b. TFIDF

	Cleaned	Model	F1 Score	Log Loss	Time
0	Yes	KNeighborsClassifier(n_neighbors=3)	0.715905	3.004626	2.256408
1	Yes	RandomForestClassifier()	0.832787	0.491764	6.973346
2	Yes	XGBClassifier(base_score=0.5, booster='gbtree'...	0.831398	0.371258	11.456856
3	Yes	AdaBoostClassifier()	0.786305	0.675571	4.107554
4	Yes	GradientBoostingClassifier()	0.805799	0.458508	19.750268
5	Yes	LogisticRegression()	0.876404	0.378960	0.460435
6	Yes	MultinomialNB()	0.867104	0.433591	0.013661

Cleaning dataset doesn't have improving F1 score metric. It somehow reduct the time processing, but it's not critical. I would recommend skipping data cleaning step.

I even doubled the sample size to increase f1 score, but it didn't help.

Model Optimization

I tried random CV search and bayesian optimization to hypertune the classifier model. I couldn't find any improvements. Therefore, I decieded to run BERT pre-trained model to increase the f1 metric value. I expect BERT will have better metric score because its bidirectional capability interprets contexts and ambiguity very well by using pre-trained text from wikipedia and fine-tuning the question and answer dataset.

Below is the results of each metric value (accuracy, recall, and f1 score).

1. Accuracy

```
Epoch 1/6
39/39 [==============================] - 3s 74ms/step - loss: 0.3042 - accuracy: 0.8732
Epoch 2/6
39/39 [==============================] - 28s 74ms/step - loss: 0.2386 - accuracy: 0.8990
Epoch 3/6
39/39 [==============================] - 3s 74ms/step - loss: 0.1912 - accuracy: 0.9232
Epoch 4/6
39/39 [==============================] - 3s 73ms/step - loss: 0.1437 - accuracy: 0.9433
Epoch 5/6
39/39 [==============================] - 3s 74ms/step - loss: 0.1140 - accuracy: 0.9565
Epoch 6/6
39/39 [==============================] - 3s 74ms/step - loss: 0.0851 - accuracy: 0.9692
```

The accuracy rate with validation data reaches above 96%.

2. Recall

```
Epoch 1/6
39/39 [==============================] - 3s 74ms/step - loss: 0.3103 - recall_m: 0.8782
Epoch 2/6
39/39 [==============================] - 29s 75ms/step - loss: 0.2370 - recall_m: 0.8906
Epoch 3/6
39/39 [==============================] - 3s 73ms/step - loss: 0.1939 - recall_m: 0.9034
Epoch 4/6
39/39 [==============================] - 3s 73ms/step - loss: 0.1604 - recall_m: 0.9166
Epoch 5/6
39/39 [==============================] - 3s 73ms/step - loss: 0.1416 - recall_m: 0.9271
Epoch 6/6
39/39 [==============================] - 3s 74ms/step - loss: 0.1591 - recall_m: 0.9198
```

The recall metric is about 91%.

3. F1 Score

```
Epoch 1/6
39/39 [==============================] - 3s 75ms/step - loss: 0.3068 - f1_m: 0.8680
Epoch 2/6
39/39 [==============================] - 29s 75ms/step - loss: 0.2343 - f1_m: 0.8792
Epoch 3/6
39/39 [==============================] - 3s 74ms/step - loss: 0.1894 - f1_m: 0.9044
Epoch 4/6
39/39 [==============================] - 3s 74ms/step - loss: 0.1537 - f1_m: 0.9154
Epoch 5/6
39/39 [==============================] - 3s 74ms/step - loss: 0.1147 - f1_m: 0.9345
Epoch 6/6
39/39 [==============================] - 3s 73ms/step - loss: 0.0871 - f1_m: 0.9495
```

The F1 score exceeds 94%.

These scores are better than classifier Logistic Regression.

Ideas for Further Research

I would try more fundamental word embedding skills and see the difference with BERT model example. BERT might work better than using LSTM and RNN with word embedding, but I still want to try and see the difference. Throughout this step, I could learn more about how to build strong neural network in Natural Language Processing.