Support Vector Machine Vs Multilayer Perceptron Over Sentiment Analysis.

Antonio Jose Lopez Roldan

Antonio.Jose-Lopez.Roldan@city.ac.uk

Abstract

This work compares the performance of Support Vector Machine (SVM) and Multilayer Perceptron (MLP) models for sentiment analysis on the "IMDb 50k Reviews Dataset." The work follows a unique pipeline for both models hence the data input is the same in both. Basic statistics calculations are made on the processed text. Various configurations are set for both models and results are recorded in a table providing enough information for comparison. Confusion matrices and ROC-AU curves are used to display the best and worst performance of the models.

1. Introduction

The explosion of social media, platform reviews, online communications, etc has created a massive amount of data containing opinions and emotions. Customer preferences drive the success or failure of any product or company. While we can't fully access every customer's mind, sentiment analysis brings us closer to understanding their likes and dislikes, significantly improving the chances of success for companies' products. Applied to this data set, sentiment analysis could add insights into creating high-grossing films. For our sentiment analysis, we will apply text analysis through Natural Language Processing (NLP) to determine whether the feeling is negative or positive.

The purpose of this paper is to compare two machine learning models' performance, a Support Vector Machine (SVM) Vs a Multilayer Perceptron (MLP) to see which one would be more efficient and effective for the sentiment analysis on the IMDb 50k reviews dataset.

The work is structured into four extra sections: Dataset, where we will overlook some basic statistics and initial analysis. Methods, where we will discuss the architecture and parameters for each model. A section dedicated to results, findings, and evaluation to analyse the performance of both models and finally, a section dedicated to the conclusions of the work.

1.1. Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) is a feedforward artificial neural network, consisting of multiple layers of nodes (neurons). MLP use non-linear activation functions which allow them to learn and model complex relationships between inputs and outputs. MLP has at least three layers, an input layer, one or more hidden layers, and an output layer. The layers are fully connected by weighted connections. MLP learn by updating these weights using the backpropagation algorithm [1].

1.2. Support Vector Machines (SVM)

Support vector machines (SVMs) find the optimal decision boundary or hyperplane that best separates different classes in the training data. For linearly separable data, SVMs locate the maximum-margin hyperplane that has the largest distance to the nearest data points of each class. These nearest points defining the margins are called the support vectors. Maximizing this margin distance between classes allows SVMs to generalize well. The key principle is locating the maximum-margin decision boundary using only critical support vector data points near the boundary. [2]

2. Dataset

The original dataset was compiled and classified by A. L. Maas et al.[1] using many methods such as BiLSTM and LSVM for example. The dataset is publicly available on Kaggle and consists of 50.000 reviews from the IMDb website. Is perfectly balanced with 50% negative reviews, 50% positive reviews, and no neutral reviews.

For this project, the original dataset was causing a heavy computational cost. The dataset was randomly reduced to only 5000 reviews, maintaining the original balance and some "trivial" preprocessing was skipped for the same reason e.g. addressing punctuations.

2.1. Initial Data Analysis

Although the original dataset was reduced the following analysis is made considering the original 50.000 reviews. A histogram has been ignored since wouldn't add relevant visual information.

From the following report, we can deduce many insights: for instance, although the average review length is slightly higher for positive reviews than for negative reviews, is not a strong indicator of the sentiment, however, if we look at the extremes, this is, the Max and Min for each sentiment the difference is substantially higher in favour of positive reviews. This might imply that movie watchers are more keen to share detailed positive experiences rather than the negatives.

Looking at the variance, we also observe that the positive reviews have a wider range of sentiment "intensities" in opposition to the negative sentiment, with lower variance suggesting that negative reviews have mostly the same length.

Table 1. Statistics

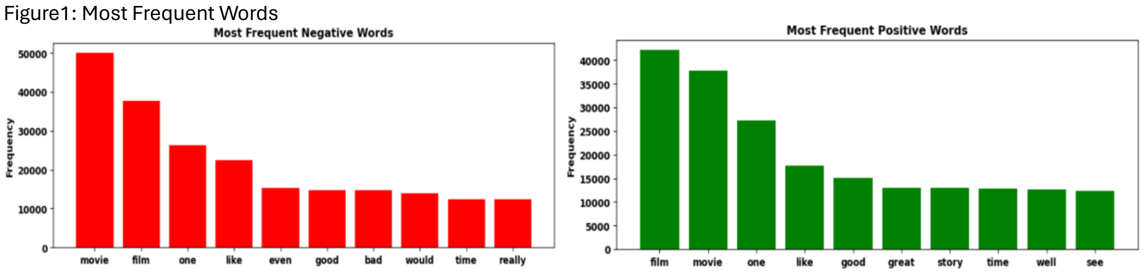
Sentiment	Min	Max	Mean	Variance	Standard Deviation
Negative	3	824	118	7386	86
Positive	6	1429	121	8905	94

Average Words: 120

Unique Words: 103774

The following chart represents the most used words in each sentiment, 4 of them are top on both sides. We could say that only bad, well, and great, are distinctly representative of their respective categories.

Figure1: Most Frequent Words



3. Methods

In this section, we detail the methodology employed for data preprocessing, model training, validation, and testing. We also describe the architectures and hyperparameters used for the Multilayer Perceptron (MLP) and Support Vector Machine (SVM) models.

[1] For further information: https://ai.stanford.edu/~amaas/data/sentiment/

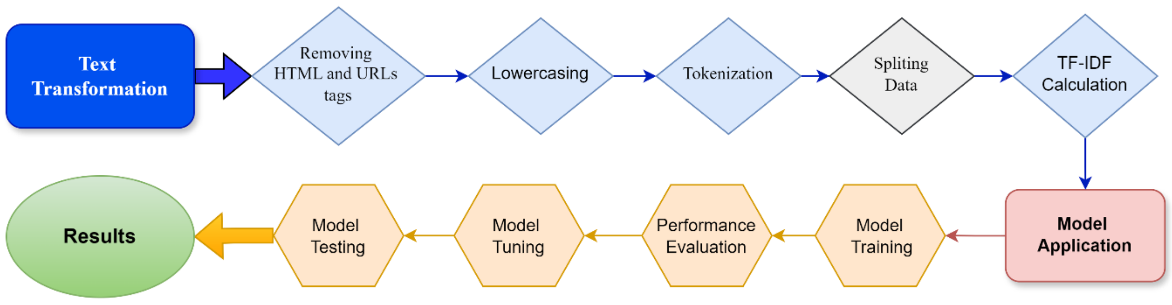
3.1. Methodology

The 5000 reviews were split into training (70%), validation (15%), and testing (15%) sets using stratified random sampling to preserve the class distribution in splits [3].

The sentiment analysis we are doing is a binary classification, Linear Support Sector is ideal for dealing with this kind of problem since uses a hyperplane to separate the two classes [4], in the same way, Multilayer Perceptron also performs well with binary classification. Regularization and hyperparameter tuning have been tested and tailored in both models.

Both models follow similar data preprocessing, training, evaluation, and testing. The flow diagram below shows the generic approach implemented in both models.

Figure 2: Pipeline



*Note that after splitting data, we continue with text transformation only for the training set.

3.2. Architecture and parameters used for the MLP.

The implemented MLP, consist of four fully connected layers. The input layer has 33154 neurons matching the dimension of the TF-IDF features. The output layer defines the classification results.

The mentioned configuration of hidden layers and neurons was based on experimentation setting and results obtained. Weights in the networks were initialized randomly and adjusted during the training. Started with a high set of neurons in the second layer, then adjusted to the number producing the best outputs. The decreasing number of neurons in the hidden layers allows the model to learn and capture the patterns from the input features.

In the hidden layers, we apply ReLU (Rectified Linear Unit) activation function. ReLU introduces non-linearity, which means that the model gains margin to learn complex decision boundaries to segregate data points defining the learning outcomes. This activation function is computationally efficient and has proved good performing in deep learning tasks [5].

In the output layer, we apply the Sigmoid function, next use the Binary Cross-Entropy (BCE) loss function which takes as input the outputs of the Sigmoid functions. During the training, the optimizer minimizes the BCE by adjusting the model parameters.

For weight optimization, we use Adam optimizer with a learning rate of 0.001 and no weight decay. After some experimentation, the Adam optimizer was chosen for being the best in terms of computational cost as well as performance. This optimizer adapts the learning rates for each parameter based on their historical gradients [6]. Early stopping was applied to prevent overfitting and improve the generalization of the model as well as a limit of 5000 epochs.

3.3. Architecture and parameters used for SVM.

In the Support Vector Machine model, we apply linear and radial basis functions (RBF) kernel. The model input is represented by the output of the TF-IDF which as we mentioned earlier is a space of 33.154. Linear kernels perform well in text classification tasks and have been widely used [7].

For the "C" hyperparameter we apply different combinations of fixed values and "np.logspace()" to search evenly over a log scale[8]. "C" controls the trade-off between achieving a low training error and a low testing error. We used 2 sets of values for the grid search space for "C", and then based on the results of the "best model" narrowed the space for 'C' in the next combinations. To limit the long search we set tolerance 'tol' hyperparameter limiting the improvement (minimization) on the loss function.

4. Results, Findings, Evaluation

4.1. Model Selection

The first step was to set a baseline model for each of our models. We configured our model following some standards for some parts whereas for others there wasn't a clear consensus about the "best" starting point hence experimentation took place. Table 2 shows the different configurations applied to both models and their results. Green colour shows the best results, yellow colour intermediates results, and red colour is for the worst performance.

In this line, for our MLP model, we applied 4 layers finding out that the neuron numbers applied on the penultimate layer affected the most on the results. Hidden layers were kept constant since during experimentation didn't show significant variations. Selected Adam optimizer. Epochs are limited to 5000 which is why there are 3 interrupted scenarios. Combinations of low learning rate (lr) and low weight decay (wd) were the steadiest as well as the most computationally costly obtaining the best results when lr > wd.

On the other hand, the SVM model shows consistent performance across the different configurations. Despite we used both Linear and RBF, the best and worst performances are produced by a linear kernel using a logarithmic grid search space for the hyperparameter "C". Even though the linear kernel produces the best performance, the majority of the worst misclassification rates are produced by the linear kernel too, in other words, RBF misclassification rates are indicative of model robustness which is generally more desirable.

Table 2:

Multilayer Perceptron							
Hidden Layers	Learning rate	Weight Decay	Stopped at Epoch	Validation accuracy	Test accuracy	Computational Cost (Time;Min)	Misclassified (%)
[610, 377, 23]	0.00001	0.00001	2545	0.81	0.81	105	18.8
[610, 377, 23]	0.0001	0.00001	200	0.83	0.81	9	18.66
[610, 377, 23]	0.001	0.00001	37	0.81	0.82	2	18.13
[610, 377, 23]	0.01	0.00001	17	0.84	0.82	1	17.86
[610, 377, 23]	0.1	0.00001	29	0.82	0.82	2	18.26
[610, 377, 23]	1	0.00001	12	0.5	0.47	1	52.66
[610, 377, 23]	0.00001	0.0001	3039	0.84	0.83	129	17.06
[610, 377, 23]	0.00001	0.001	3994	0.84	0.84	165	15.86
[610, 377, 23]	0.00001	0.01	Interrrupted; 5000<	0.5	0.53	231	47.33
[610, 377, 23]	0.00001	0.1	Interrrupted; 5000<	0.5	0.47	220	52.66
[610, 377, 23]	0.00001	1	Interrrupted; 5000<	0.5	0.47	218	52.66
[610, 377, 23]	1	0	12	0.5	0.53	1	52.66
[610, 377, 23]	0.1	0	24	0.81	0.8	1	20.4
[610, 377, 23]	0.01	0	15	0.76	0.77	1	22.9
[610, 377, 23]	0.001	0	44	0.81	0.81	2	19.06
[610, 377, 23]	0.0001	0	241	0.81	0.81	10	18.66
[610, 377, 23]	0.00001	0	676	0.83	0.81	199	19.06
			Average	0.72	0.72	76	28.39
			Max	0.84	0.84	231	52.66
			Min	0.5	0.47	1	15.86

Support Vector Machine							
Kernel function	Cross Validation	GridSearch Space for "C"	Box Constant	Validation accuracy	Test accuracy	Computational Cost (Time;Min)	Misclasified (%)
Linear	3	np.logspace(0.1, 5, 3)	1.2589	0.85	0.85	27	14.93
Linear	3	[0.1, 3, 5]	3	0.84	0.85	28	16
Linear	3	np.logspace(1, 30, 3)	10	0.84	0.84	46	16.14
Linear	3	[10, 20, 30]	10	0.84	0.84	46	16.14
Linear	3	np.logspace(0.75,1.75,3)	5.6234	0.84	0.84	27	16.4
Linear	3	[2.5, 3, 3.5]	3.5	0.83	0.84	29	16.26
Linear	3	np.logspace(-50,50,3)	1	0.86	0.85	27	15.2
RBF	3	np.logspace(0.1, 5, 3)	354.8134	0.86	0.85	36	15.06
RBF	3	[0.1, 3, 5]	3	0.86	0.85	36	15.06
RBF	3	np.logspace(1, 30, 3)	10	0.86	0.85	38	15.06
RBF	3	[10, 20, 30]	10	0.86	0.85	41	15.06
RBF	3	np.logspace(9, 11, 3)	1.00E+09	0.86	0.85	43	15.06
RBF	3	[2.5, 3, 3.5]	2.5	0.86	0.85	40	15.06
RBF	3	np.logspace(-50,50,3)	1.00E+50	0.86	0.85	40	15.06
			Average	0.85	0.85	36	15.46
			Max	0.86	0.85	46	16.4
			Min	0.83	0.84	27	14.93

4.2. Algorithm Comparison

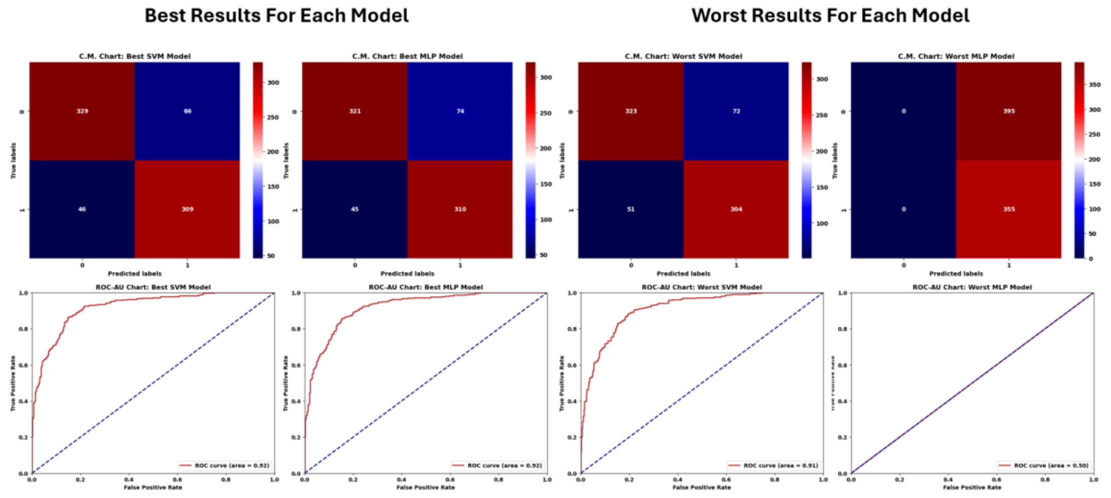
Figure 3 displays the best and the worst performance for each model using two kinds of charts, a confusion matrix, and an AUC-ROC curve. For each category (best, worst results) we plot models one next to the other for better comparison. Those results are from the test set, we trained and validated both models before. On average, the MLP misclassification rate is twice the SVM (see Table 2).

During the selection process, we appreciate a high variability in the MLP model in every point measured, and this is reflected also in its worst vs best models. Looking at the best results for the MLP we see an accuracy of 84% which is lower than the best SVM model with 85% and a misclassification of 15.86% Vs 14.93 respectively. On the other hand, the worst scenario offers an accuracy of 52.66% for the MLP model and 16.4% for the SVM model. Notice we omitted the "interrupted" models (due to reaching the 5000 limit epochs) for two reasons, first of all, it didn't finish the training, secondly, if we pay close attention to the previous two configurations, we see that the models improve as we decrease the weight decay keeping the learning rate constant (See Table 2), hence it might kept improving if more time/computing resources were available, this appreciation is supported by the steady decrease on loss function calculations during the validation loss.

At first glance the ROC-AU curve doesn't provide any decisive information, however, apart from the obvious poor performance for the worst scenario of the MLP model, looking closer seems that the best model of the MLP has the most regular curve, whereas the other two charts for SVM has some more noticeable spikes or irregularities in comparison with MLP. This could indicate that the model's performance is more consistent across different classification thresholds, yet when we check the absolute difference between false positives and false negatives, overall, the best classifier is without any hesitation the SVM model: SVM; |66-46| = 20, MLP; |45-74| = 29. 20<29.

The computational cost is another factor to consider, as training time does not always correlate directly with performance. In the case of MLP, during our selection process, we noticed that certain configurations which involved higher times, tend to achieve better performance. However, the SVM models exhibit a different behaviour since during the selection process the computational cost remains relatively stable. Interestingly, this stability is accompanied by a consistent accuracy performance. In other words, SVM is less sensitive to variations in the configuration of hyperparameters.

Figure 3: Confusion Matrix (C.M.) and ROC-AU graphical representation for the worst a best performance on both models.



5. **Conclusion**

ROC Curve is a tool we can use to contrast the results shown by the confusion matrix and support our findings, to build more solid conclusions.

In the real world, many industries demand an accuracy higher than 90% for model deployment hence to achieve a meaningful piece of work with higher accuracy we should keep investigating and experimenting further in both models. In this line, we have a wide range of options, from setting larger epochs (more time) for the MLP model to choosing different routes in the pipeline, in particular for the case of SVM due to its insensitiveness, such as punctuation removal, application of n-grams or use of word embedding e.g. "Word2Vec".

Regarding robustness, SVM is the clear winner as showed to be less sensitive to the variation on the hyperparameters, this can be beneficial as we mentioned before stability is one of the things we usually seek in a model. However, this is also a point against the model, why? Showing little variation when fine-tuning the hyperparameters could mean that our possibilities for improving this model are limited by the preprocessing text data we followed, this is, if we want to increase our chances of improving the model's performance, we would need to start over a different path for the preprocessing following a different pipeline. Taking a different perspective, the instability of the MLP model can be its strength that we could use for the sake of improving the accuracy. Even more, considering that the mentioned instability appears only with specific combinations of values, this is, when the learning rate is greater than the weight decay. As the learning rate increases over the weight decay results start to improve, therefore would be interesting to experiment in this direction. In summary, if we have time for experimentation MLP has the potential for improvement although we would still have the problem of the computational cost. On the other side, if our priority is stability at a manageable computational cost, then SVM should be the choice.

6. **References**

[1] "Multilayer Perceptron," Wikipedia, https://en.wikipedia.org/wiki/Multilayer_perceptron (accessed Mar. 29, 2024).
[2] "Support Vector Machine," Wikipedia, https://en.wikipedia.org/wiki/Support_vector_machine (accessed Mar. 29, 2024).
[3] S. Raschka, 'Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning'. arXiv, Nov. 10, 2020. Accessed: Apr. 01, 2024. [Online]. Available: http://arxiv.org/abs/1811.12808
[4] D. O. Ratmana, G. Fajar Shidik, A. Z. Fanani, Muljono, and R. A. Pramunendar, 'Evaluation of Feature Selections on Movie Reviews Sentiment', in 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), Semarang, Indonesia: IEEE, Sep. 2020, pp. 567–571. doi: 10.1109/iSemantic50169.2020.9234287.
[5] X. Glorot, A. Bordes, and Y. Bengio, 'Deep Sparse Rectifier Neural Networks'.
[6] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization'. arXiv, Jan. 29, 2017. Accessed: Apr. 02, 2024. [Online]. Available: http://arxiv.org/abs/1412.6980
[7] T. Joachims, 'Text categorization with Support Vector Machines: Learning with many relevant features', in Machine Learning: ECML-98, vol. 1398, C. Nédellec and C. Rouveirol, Eds., in Lecture Notes in Computer Science, vol. 1398. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142. doi: 10.1007/BFb0026683.
[8] J. Stalfort, "Hyperparameter tuning using grid search and Random Search," Medium, https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35 (accessed Apr. 14, 2024).
[9] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, 'A Practical Guide to Support Vector Classification'.
[10] J. C. Platt, 'Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines'.
[11] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, 'Learning Word Vectors for Sentiment Analysis'

Glossary and Other Results

Data Preprocessing:

- BiLSTM (Stands for Bidirectional Long Short-Term Memory): Is a type of recurrent neural network (RNN). It consists of two Long Short-Term Memory layers, they allow the model to capture the information from both past and future states [1].
- LSVM (Linear Support Vector Machine): Is a binary classification algorithm that use an optimal linear hyperplane to separate two classes [2].
- "Word2Vec" Uses vectors, which are representations of words to capture the semantic relationships between those words [3].

MLP model:

- Computational cost: We used the time of running the model and printing results as the computational cost.
- Activation function: The activation function introduces non-linearity in our MLP model enabling to learn complex patterns from the input data [4].
- Binary Cross-Entrophy: is a loss function used to train binary classifiers. It compares the predicted probabilities of the positive class against the true labels and penalizes the model for incorrect predictions [5].
- Adam (Adaptative Moment Estimation) optimizer: This algorithm adapts the learning rate for each parameter based on the estimations for the first and second moments of the gradients [6].
- Learning Rate: is about how quickly the model adapts to the training data. It controls the weight updates during the training. Small values lead to slow and precise updates, large values fast but potentially more investable.
- Weight Decay: used to prevent overfitting penalizing the loss function. It encourages the model to learn smaller weights.

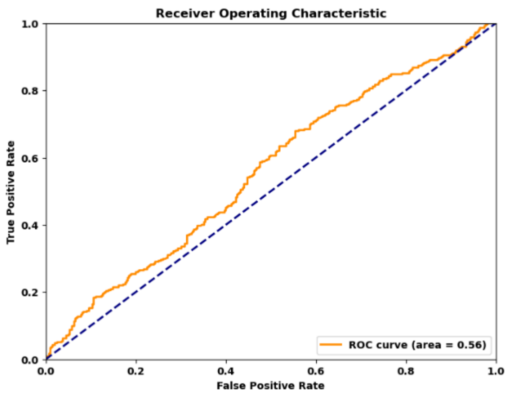
SVM model:

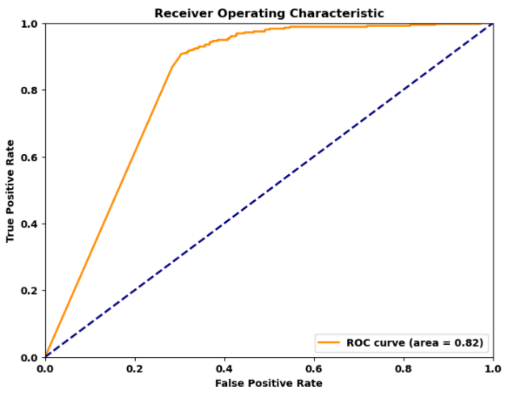
- Kernels, are functions that transform the input data allowing to learn non-linear decision boundaries. In our work used Linear and RBF (Radial Basis Function) [7].
- C parameter, is used to regularize the trade-off between achieving a low training error and a low testing error. Small values tend to produce larger margins and more misclassifications, larger values small margins, fewer misclassifications.
- Tolerance "tol", is a threshold parameter to set when to stop the optimization process, we used it to avoid excessive running times [8].
- Logarithmic grid search np.logspace() is supposed to be more efficient compared to linear spacing. It searches hyperparameters over a logarithmic space (defined) [9].

About experimentation and the selection process.

During the construction of the MLP, initially used the Stochastic Gradient Descendent (SDGs), however, the results were poor and changed for Adam optimizer which adjusts the initial learning rate, set in the code as "lr", it also "internally" adjusts momentum.

As we mentioned in the course work, Multilayer Perceptron was the most unstable model hence it consumed more time in the configuration process, in addition provided a wider room for experimentation (and learning) about the influence on the model of the different configurations. Also, apart from the already displayed results, I also got some interesting ROC-AUC curves, the second curve for instance shows an anomaly slope at the beginning meaning that initially is not that good at classifying the labels and misclassify many True Positive rates as False Positive rates:





The initial pipeline was as follows: 1) Removing HTML tags and URLs, Punctuation, Replacing emoticons. 2) Tokenization, 3) remove stop words, 4) lemmatization, 5) split the data, 6) build vocabulary, 7) TF-IDF Calculation, 8) padding. However, got stuck in the last two points getting some errors, after investigating, initiated the current path as shown in the diagram workflow.

References:

[1] E. Zvornicanin, "Differences between bidirectional and Unidirectional LSTM," Baeldung on Computer Science, https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm (accessed Apr. 16, 2024).
[2] A. Saini, "Guide on Support Vector Machine (SVM) algorithm," Analytics Vidhya, https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/ (accessed Apr. 16, 2024).
[3] "Word2vec : text : tensorflow," TensorFlow, https://www.tensorflow.org/text/tutorials/word2vec (accessed Apr. 16, 2024).
[4] "Activation function," Wikipedia, https://en.wikipedia.org/wiki/Activation_function (accessed Apr. 16, 2024).
[5] D. Godoy, "Understanding binary cross-entropy / log loss: A visual explanation," Medium, https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a (accessed Apr. 16, 2024).
[6] N. Vishwakarma, "What is Adam Optimizer?," Analytics Vidhya, https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/#:~:text=The%20Adam%20optimizer%2C%20short%20for,Stochastic%20Gradient%20Descent%20with%20momentum. (accessed Apr. 16, 2024).
[7] J. Stalfort, "Hyperparameter tuning using grid search and Random Search," Medium, https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35 (accessed Apr. 14, 2024).
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[8] "Sklearn.svm.SVC," Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (accessed Apr. 16, 2024).