Daniel Cersosimo

1. <u>Introduction</u>

Contextualization:

Every minute, people across the world are constantly assessing and evaluating various industries and sectors of society. Oftentimes, these assessments are penned into reviews, effectively providing organizations with the means of gauging the sentiment which reviewers express for their products, operations, etc.. These reviews are quite useful as they provide insight into the state of mind of those involved in a company or institution which when analyzed, can allow these entities to assess their business practices to optimize their consumers views of the organization which in turn, renders a more optimal operation overall. As a result of this clear business incentive, review data has skyrocketed in recent years in volume, variation, and complexity as more organizations gather review data in an effort to contribute to optimal practices. While one can note the sound reasoning which gives way for this "review data explosion", this rapid increase has also rendered human evaluation on these vast datasets virtually impossible. As a result, the application of Machine Learning to streamline these assessments in various capacities has expanded tremendously. The construction of these models has rendered transformative results in allowing organizations to evaluate the intricacies of their review data at such a large scale. As this realm of unstructured review data and modeling continues to grow, I have conducted a project to process and model large scale IMDB movie review data to build models which can effectively predict the sentiment of these reviews.

Overview of the Project:

The IMDB data houses a two dimensional dataset of reviews and their sentiments. The reviews are unstructured text data as strings containing these paragraph long reviews. The sentiment feature contains two options, positive or negative and is associated as a label in a sense to each review. Given the nature of the already stated goal of this project, I will be employing the sentiment labels as the target variable with the reviews converted to vectors via word embedding as inputs for the ML and DL models to be trained on with respect to the target labels. In order reach this point, extensive preprocessing and exploration of the data was required which is detailed in the subsequent sections of this report

2. Data Preparation and Exploration

Data Loading:

The data was imported via a csv file which contained the IMDB review data described above. This was loaded via the pandas read_csv and saved to a dataframe.

Initial Exploration of the Data:

In order to properly understand how to clean the data, I conducted some preliminary exploration to nuance myself with the intricacies of the dataset. This consisted of checking for null values, which there were none of, as well as searching for duplicate instances. There were a small number of repeated reviews, relative to the 50,000 total, which I removed from the dataset. I also noticed the presence of HTML tags within the text which I would need to handle in cleaning. After the duplicate adjustment, the data was in good condition and ready for progression into preprocessing.

Data Preprocessing:

As I embarked on this procedure, my initial focus was on processing the unstructured text data into a format which can be utilized by ML and DL models to conduct sentiment classification. In order to achieve this, I began by executing a code block which executed the multiple processes necessary to achieve a cleaned text output. I coded a twice nested loop which effectively iterates through the reviews, terms of said review, and characters of said term in order to execute the cleaning and formatting of the text. The intricacies of this code logic are detailed in the comments within the code itself. Following execution, every review text has html tags removed, stop words taken out, lowercase terms, special characters.punctuation removed, and the terms lemmatized. Given I will be utilizing word embedding to generate vectors for the reviews, I opted for lemmatization since this practice of term conversion simplifies them into terms which are more likely to be linguistically aligned and not stems which are not actual terms. This is important since I will be utilizing the word2vec model to generate the vectors which may have difficulty working with stems that are not real terms and therefore may not be in the corpus of the training. In sum, this code effectively processes the text and it is primed for word embedding. I downloaded the word2vec pretrained model which computes 300 dimensional vectors for each term in every review. I opted for the 300 dimensional vectors to provide enhanced complexity in an effort to optimize performance despite the potential high computation. Following this, I wrote another code block which iterates through each review and the terms within each via a nested loop in order to compute a 300 dimensional vector for each term within a review. Upon this, these vectors for every term are aggregated to an averaged 300 dimension vector for each review, effectively representing the terms within. This is done for every review and the contents saved and subsequently loaded to a dataframe after this block. Upon completion of this, I pivoted to handling the target feature by converting the sentiments of positive and negative toi 1 and 0 respectively to guarantee the data can be interpreted by the upcoming models. This binary sentiment feature was then added to the review vectors data frame followed by a train test split executed via scikit learn, resulting in the finalized data ready for modeling

3. <u>Modeling for Sentiment Classification</u>

Logistic Regression:

For the first modeling technique, I utilize logistic regression as a means to conduct the binary classification for the sentiments of the reviews. I import the necessary libraries and functions for logistic regression to train and predict with the model. Upon successful completion of this training and evaluation, this model scores a 85.38 percent accuracy. This appears rather good however comparison against the other models will be needed to truly be able to make this assessment. Accuracy is focused on for this and will be for all the models since the dataset has a roughly even distribution of positive and negative labels making accuracy an excellent metric in this context. The additional metrics such as f1-score, precision and recall are displayed in the classification report for each model in the notebook alongside support.

LinearSVC:

For the second modeling technique, I leverage a linear classifier using support vectors and import the necessary libraries and functions for LinearSVC to train and predict with the model. After training and evaluation, this model scores a 85.74 percent accuracy, marginally better than logistic regression. A similar performance as logistic regression allows a benchmark to begin to form as a means of assessing model performance. This will be further refined based on further model accuracies.

k-NN Modeling:

As my third technique, I use k-NN classification and load the necessary functions to do so. Following training and testing, this model performs with an accuracy of 79.36 percent. This is noticeably lower than the previous two models, resulting in the logistic regression and LinearSVC being better performing models based on this data and the context of the project.

Feedforward Neural Network:

For the fourth model, I designed a neural network of multiple fully connected layers, each providing some unique functionality to the collective model. I sought to design the architecture for a neural network which abides by the criteria of the 4th model for the project. In order to execute this, I employed a feedforward neural network composed of dense layers and a dropout layer. I start by defining my model using keras sequential model as it is a versatile model which I can harness to construct the intricacies feedforward neural network. I use relu activation in the non output dense layers below as a means to allow for nonlinear patterns to be captured. Subsequently, the first input layer is added with the shape of 300 for the vector dimensions and 128 neurons. Another dense hidden layer is connected beneath for enhanced complexity for the model to pick up on complex underlying patterns in the data. This is followed with a dropout layer to combat overfitting by dropping 50 percent of neurons at each iteration via random selection. After this structure is developed, I add an output dense layer of one neuron for the single binary classification output which uses sigmoid activation here as this provides the means of the either 0 or 1 output value. The model is then compiled ready to be used with the data. This is done by training the FNN on the data below with 12 epochs and a batch size of 35 following some experimentation. This is then tested on as it is quite intensive. The input shape is 300 for the dimensions of the vectors. I incorporate a flatten layer to effectively connect the convolutional layers output after pooling to reformat the multi dimensional output to one which is fed into the dense layer with relu activation and a dropout for the same reasons as with the FNN. The single dense layer for output is configured similar to the FNN as well with sigmoid activation function to allow for the 0 or 1 binary output value. Further interactions into this are detailed in the inline comments in the notebook. This is then tested on the test data with an ultimate accuracy of 86.01 percent displayed, the best performing model thus far by a slight amount.

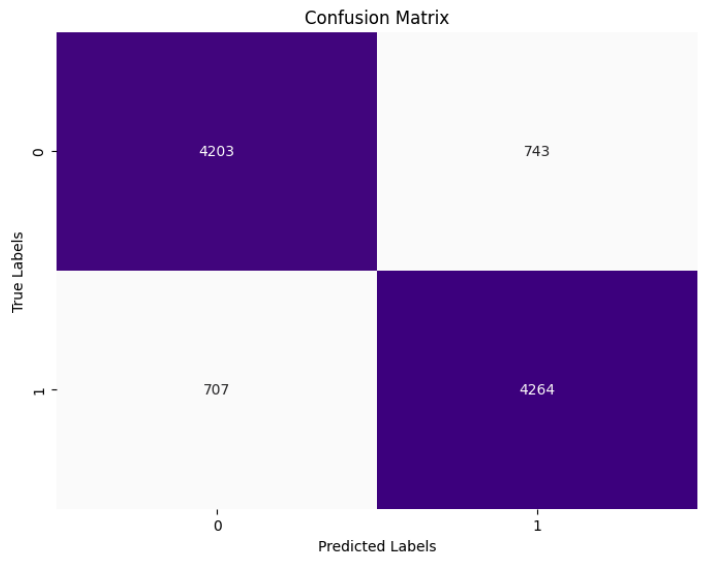
CNN:

For the fifth model, I again leveraged keras to construct a CNN architecture. I utilize two convolutional layers 200 and 100 filters each after experimentation with a max pooling for each and is then compiled ready to be used with the data. The CNN is trained on the data below with 12 epochs and a batch size of 35 as done with the FNN to primarily improve computation as this is with an ultimate accuracy of 85.22 percent displayed, falling just behind logistic regression by a slight margin.

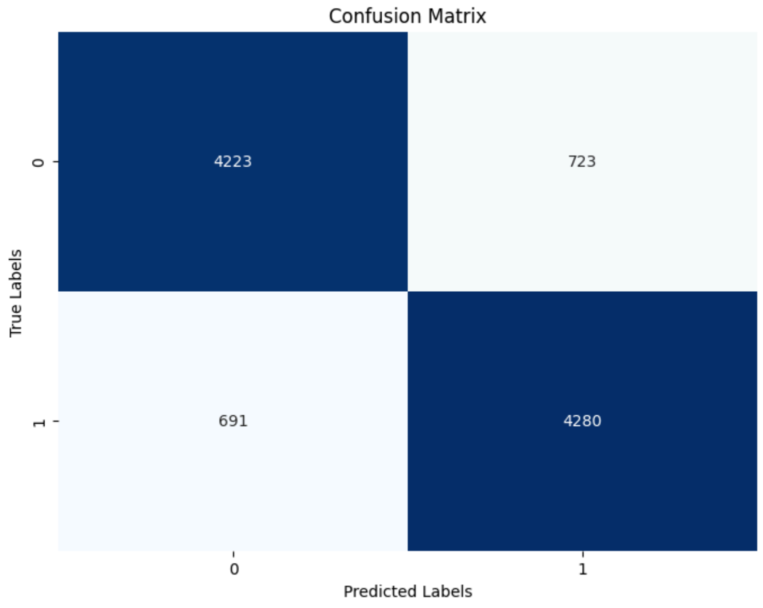
<u>4. Results, Conclusions, and Future Work</u>

Visualizations:

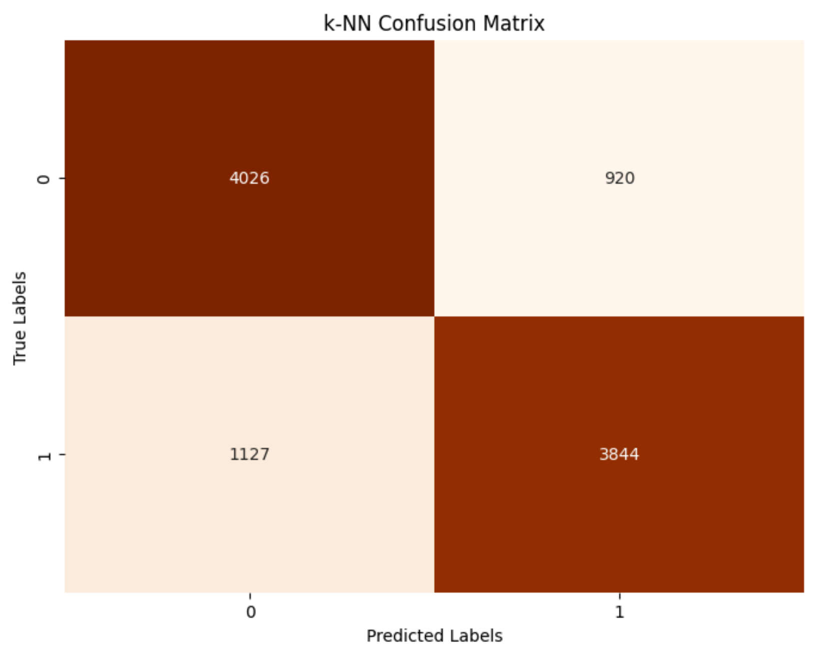
Below is the confusion matrix for logistic regression



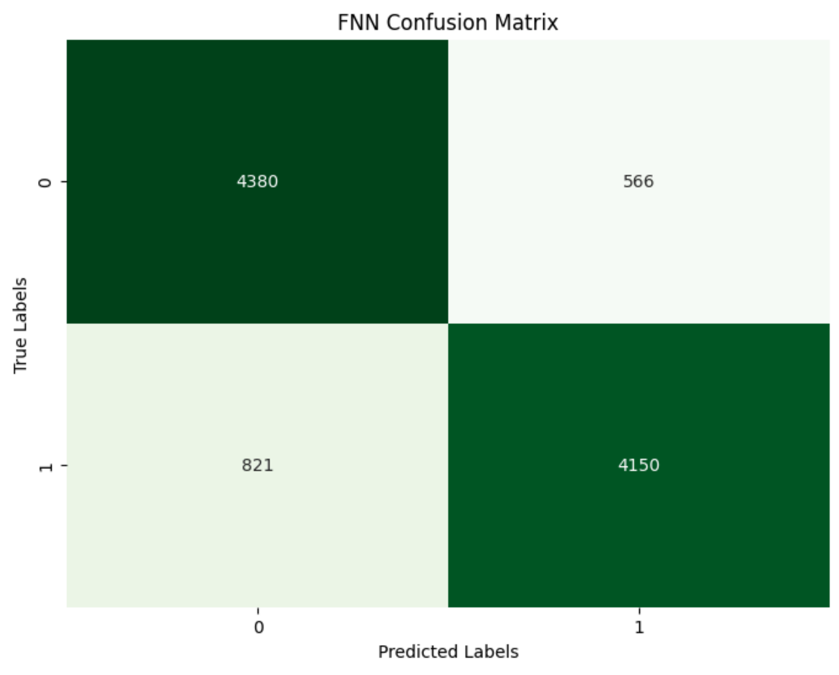
Below is the confusion matrix for LinearSVC



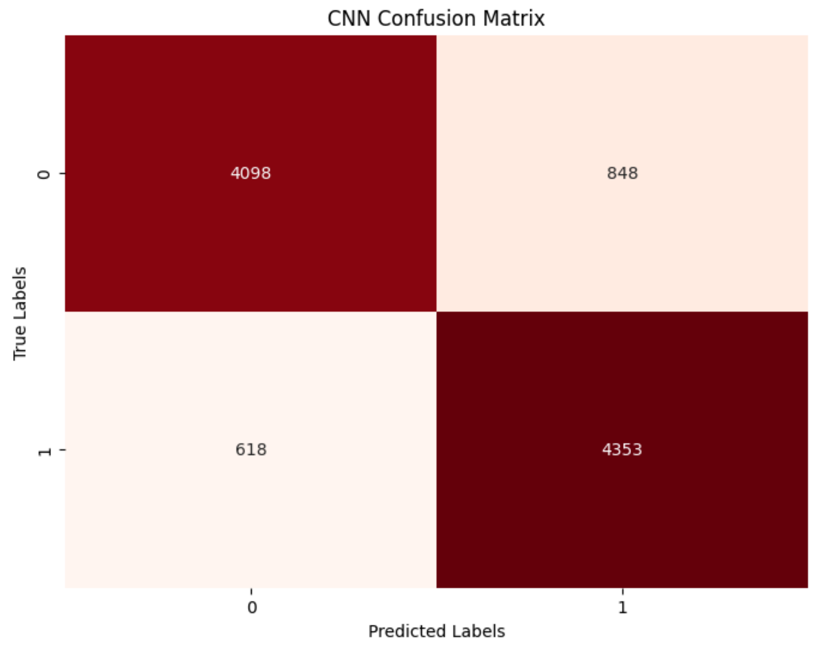
Below is the confusion matrix for k-NN classification



Below is the confusion matrix for the Feedforward neural network



Below is the confusion matrix for the CNN



Conclusion and Future Work:

Overall, this project unveiled some notable insight into this dataset while resulting in the successful training and testing of five models. As for the data itself, the presence of duplicates was quite intriguing, especially given it was such a minute amount relative to the total. In addition, the even distribution of the sentiment labels was insightful and spearheaded the usage as accuracy to evaluate the models. For the models, they look to be able to classify the reviews quite well with almost all of them hovering around 85 to 86 percent accuracy, k-NN (79.36%) being the exception. As stated above, I focused on accuracy due to the nature of the even class distribution to evaluate these models. While the FNN had the best performance, I would not be inclined to pursue this model in hypothetical deployment scenarios based on this single test as LinearSVC performs almost just as well with much more rapid computation. In order to truly assess which would be the most viable for a deployment, several factors would need to be considered with some examples being what tool it will be deployed for, if the data will be adequately represented by this training, the computation capability. One route of future work would be considering potential deployments for the models, one potential idea being a tool to uptake user reviews of movies to classify them in real time. In addition, additional methods of word embedding this review data could be utilized such as BERT word embedding (this was attempted but too computationally intensive) or even simpier bag of words methods which could impact the performances of these models. In sum, given the techniques employed by this project to clean the data, and the usage of word2vec to compute vectors for the terms, the training of these various models was able to be successfully done with the Feedforward neural network as the best performer and the LinearSVC having a strong balance of performance and efficiency.