Rohan Athalye
CS 171
12/12/2022

Final Exam Report

Dataset Description

For this project, I used the IMDB Movie Reviews dataset from TensorFlow Datasets. The IMDB Movie Reviews dataset consists of 50,000 movie reviews, with 25,000 reviews labeled as positive and 25,000 reviews labeled as negative. This is a binary classification problem where the goal is to correctly classify a movie review as either positive or negative.

Data Preprocessing

Because the input data, which are the movie reviews, are text data, I had to preprocess it in order for it to be used with ML models. To perform some of the preprocessing, I used the Natural Language Toolkit (NLTK), a Python package containing a multitude of libraries and programs for natural language processing.

First, I converted all of the reviews to lowercase characters and removed any HTML tags from them. Then, I removed any square brackets and special characters from the reviews. Next, I tokenized the reviews, meaning I split each review into a list of words. I then removed any irrelevant words from the reviews such as “a”, “the”, “and”, etc. These words are known as stop words. I also applied stemming to each word in every review, meaning cutting down the word to its root form. For example, the words “likes”, “liked”, and “likely” would be stemmed to the root word “like”. Next, I applied lemmatization to the verbs in each review where verbs with similar meanings are linked to 1 verb. For example, the verbs “am”, “are”, and “is” would be linked to the verb “be”. After lemmatizing the verbs, I similarly lemmatized the nouns in each review. Finally, I converted the reviews to numeric vectors. Specifically, I used TF-IDF vectorization, which involves calculating the TF-IDF score for every word in the collection of reviews relative to that review. The TF-IDF score is a measure of how important/relevant a word is. A higher TF-IDF score means that a word is more important/relevant as opposed to a lower TF-IDF score. So, each review corresponds to a vector that has a TF-IDF score for every word in the collection of reviews. This concludes the preprocessing phase of the input data, which is now ready to be used with ML models.

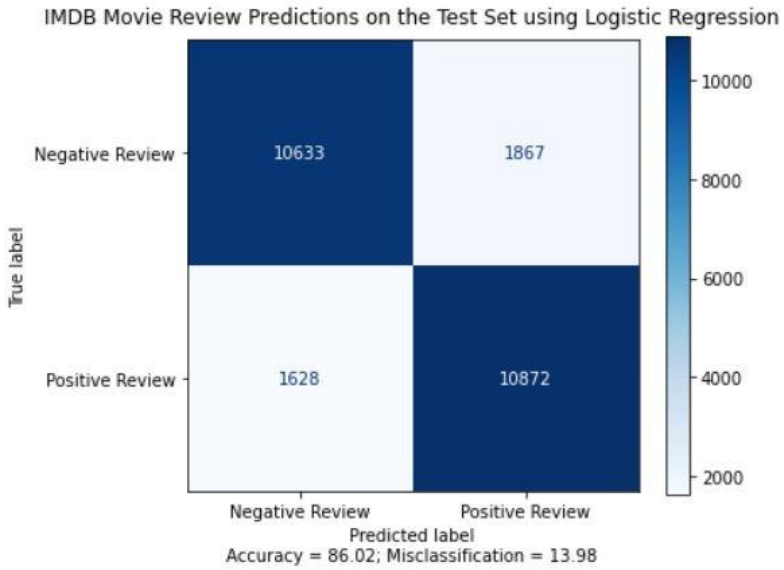
Results

I trained 5 ML models with the default hyperparameters to classify movie reviews as positive or negative. I summarized the results of my models' performance in the table below.

Model	Training Accuracy (%)	Mean Cross-Validation Accuracy (%)	Test Accuracy (%)
Logistic Regression	87.47	85.92	86.02
KNN	83.44	73.61	70.44
MLP	100	84.7	84.95
Random Forest	100	82.65	83.07
Ensemble Learning	95.86	86.3	86.28

Logistic Regression

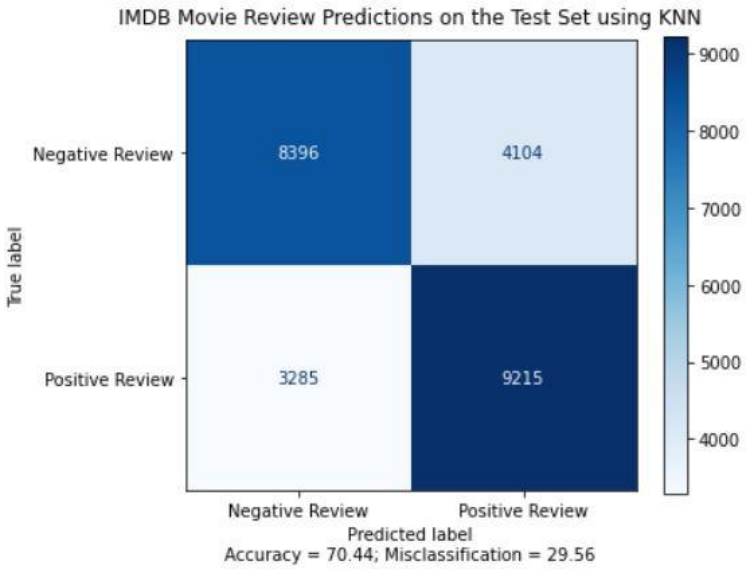
Using Logistic Regression to classify movie reviews as positive or negative yielded a test accuracy of 86.02%. There does not seem to be any overfitting as the training, mean-cross validation, and testing accuracies are fairly close to each other. I plotted a confusion matrix below to visualize the performance of Logistic Regression on the test set.



From the confusion matrix, we can see that 1628 positive reviews were incorrectly classified as negative reviews, and 1867 negative reviews were incorrectly classified as positive reviews. Thus, the resulting test accuracy was 86.02%.

KNN

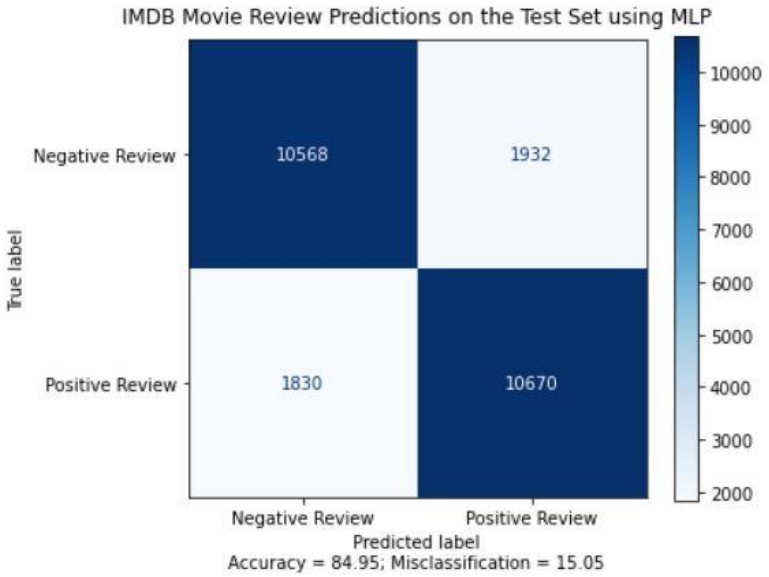
Using KNN to classify movie reviews as positive or negative yielded a test accuracy of 70.44%. Overall, KNN performed poorly, producing a mediocre training accuracy and low mean cross-validation and test accuracies. I plotted a confusion matrix below to visualize the performance of KNN on the test set.



From the confusion matrix, we can see that 3285 positive reviews were incorrectly classified as negative reviews, and 4104 negative reviews were incorrectly classified as positive reviews. Thus, the resulting test accuracy was 70.44% with many more misclassifications compared with using Logistic Regression.

MLP

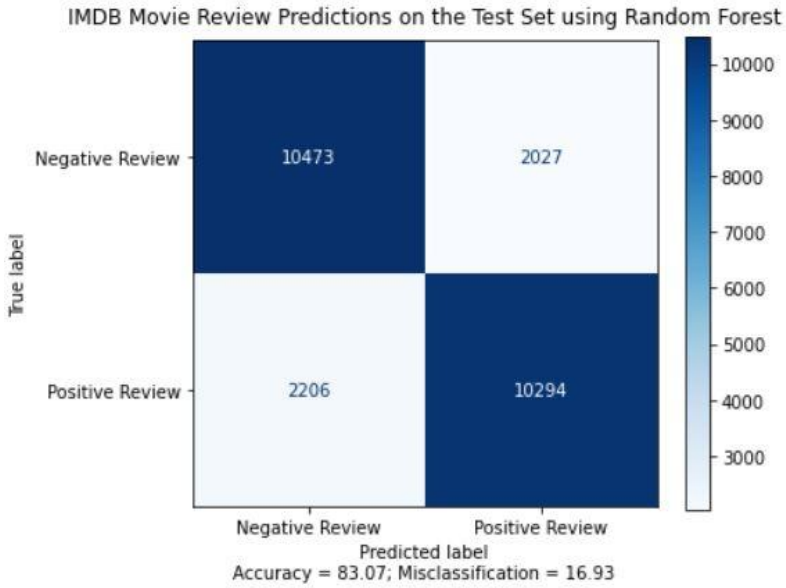
Using MLP to classify movie reviews as positive or negative yielded a test accuracy of 84.95%. There is clear overfitting since the model performs perfectly on the training set with an accuracy of 100% but significantly worse on the test set with an accuracy of 84.95%. I plotted a confusion matrix on the next page to visualize the performance of MLP on the test set.



From the confusion matrix, we can see that 1830 positive reviews were incorrectly classified as negative reviews, and 1932 negative reviews were incorrectly classified as positive reviews. Thus, the resulting test accuracy was 84.95%.

Random Forest

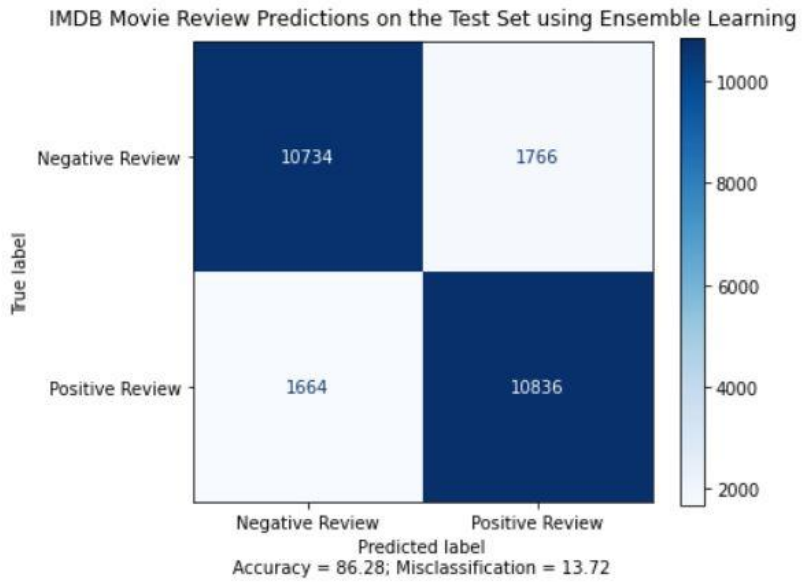
Using Random Forest to classify movie reviews as positive or negative yielded a test accuracy of 83.07%. Just like with MLP, there is clear overfitting since the model performs perfectly on the training set with an accuracy of 100% but significantly worse on the test set with an accuracy of 83.07%. I plotted a confusion matrix below to visualize the performance of Random Forest on the test set.



From the confusion matrix, we can see that 2206 positive reviews were incorrectly classified as negative reviews, and 2027 negative reviews were incorrectly classified as positive reviews. Thus, the resulting test accuracy was 83.07%.

Ensemble Learning

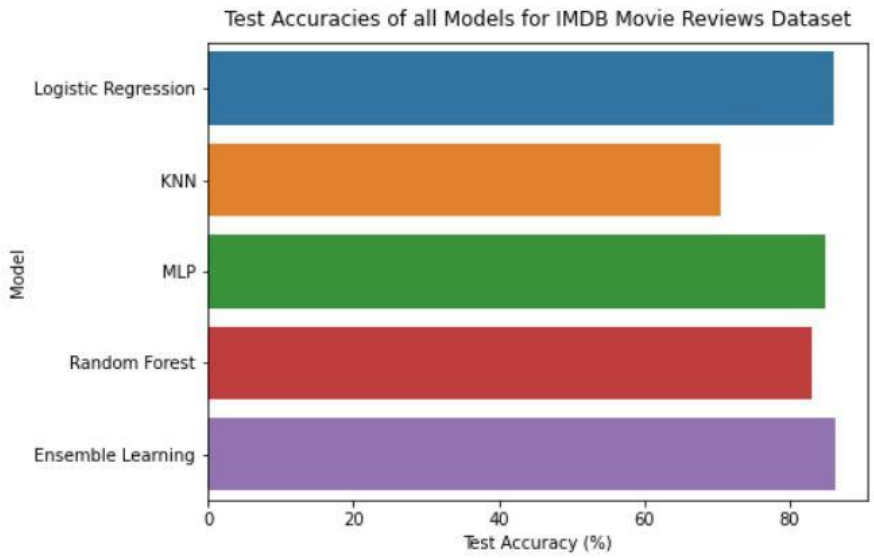
Using Ensemble Learning to classify movie reviews as positive or negative yielded a test accuracy of 86.28%. For Ensemble Learning, I used a Stacking Classifier that takes my individual models and combines them into 1 model. The model performs well on the training set with an accuracy of 95.86% but fairly worse on the test set with an accuracy of 86.28%. So, there is some overfitting but not as much compared with MLP and Random Forest. I plotted a confusion matrix below to visualize the performance of Ensemble Learning on the test set.



From the confusion matrix, we can see that 1664 positive reviews were incorrectly classified as negative reviews, and 1766 negative reviews were incorrectly classified as positive reviews. Thus, the resulting test accuracy was 86.28%.

Conclusions

For this project, the goal was to correctly classify movie reviews as either positive or negative. Because the movie reviews were text data, I first had to preprocess it in order for it to be used with ML models. After preprocessing the movie reviews, I trained 5 ML models using the default hyperparameters. The models that I trained were Logistic Regression, KNN, MLP, Random Forest, and Ensemble Learning using a Stacking Classifier. On the next page, I plotted the resulting test accuracies for each model in the form of a bar graph.



From the bar graph, we can see that KNN is the worst performing model as it has the lowest test accuracy. The best model is the model that produces the highest test accuracy without overfitting the training set. The bar graph shows that Logistic Regression and Ensemble Learning have the highest test accuracies, but according to the table that summarizes the results of my models' performance, Ensemble Learning has a slightly higher test accuracy by just 0.26%. However, I analyzed that Ensemble Learning overfits the training set, so it cannot be the best model. Logistic Regression has the 2nd highest test accuracy, and I analyzed that it does not overfit the training set. Thus, I believe that the best model is Logistic Regression.

I would like to point out some weaknesses in my project that may have prevented me from achieving the best possible results. First, I had to limit the number of words from the collection of movie reviews to 1,000 words for calculating the TF-IDF scores. When I tried to use more words, my runtime session would crash because I had run out of available RAM. Perhaps if I could use more words, my results could have been different. Second, I wanted to use SVM as one of my models, but it was taking too long to train. So, due to the time constraints of this project, I had to halt its training and look to use other models. Third, I wanted to perform hyperparameter tuning on each of my models, but it was taking a long time, and my runtime session would eventually crash because I had run out of available RAM. As a result, I decided to use the default hyperparameters for my models. Hyperparameter tuning would have produced higher test accuracies for my models, and I would have made sure not to fiddle with the hyperparameters too much so as to prevent overfitting.

In the future, I would like to use RNN, specifically LSTMs, and see if it produces a higher test accuracy than my other models and if it also tends to overfit the training set.

Sources

- https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/
- https://www.kaggle.com/code/sohamdas27/imdb-movie-review-eda-sentiment-analysis/notebook
- https://www.kaggle.com/code/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews
- https://colab.research.google.com/drive/1u00O59LZBtYjFS7Bb9dhDXpbCH4Bkob9?usp=sharing
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
- https://stackoverflow.com/questions/61325314/how-to-change-plot-confusion-matrix-default-figure-size-in-sklearn-metrics-packa