Analysis of Classification Methods for IMDB Movie Reviews

University of California, Irvine
CS 178 | Spring 2023
Professor Gavin Kerrigan

Project Members:
Safeeullah Saifuddin [84845365] | Nathan Huey [41269786] | Aman Grewal [10568426]

I. Summary

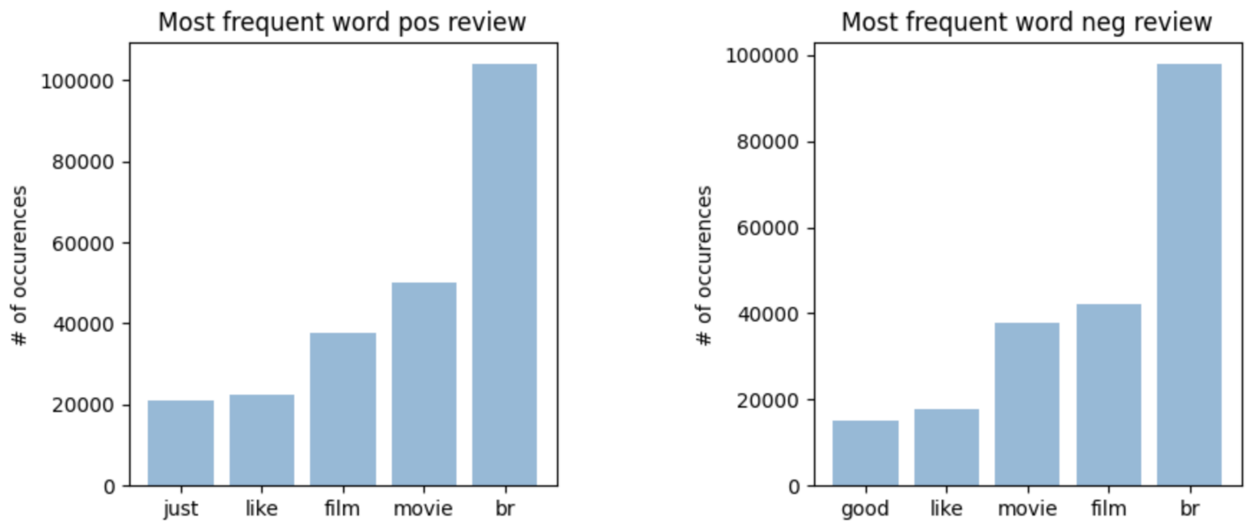
The purpose of this project was to analyze the effect increasing the amount of training data has on testing accuracy. We chose to analyze this relationship by use of the IMDB Movie Review dataset. The classifiers that we analyzed were k Nearest Neighbors, Decision Tree, Feed Forward Neural Network, and Logistic Regression. Through this project, we were able to observe that the increase of testing accuracy in relation to the amount of training data added plateaus for some classifiers. Thus in some classifiers, the benefit of adding more training data becomes negligible once a certain threshold is passed.

II. Data Description

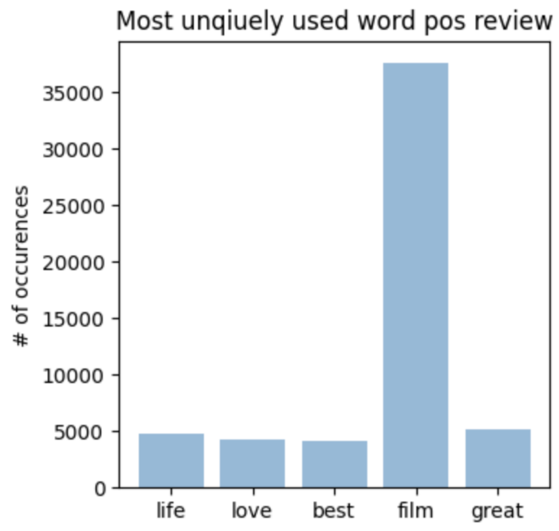
Data Exploration

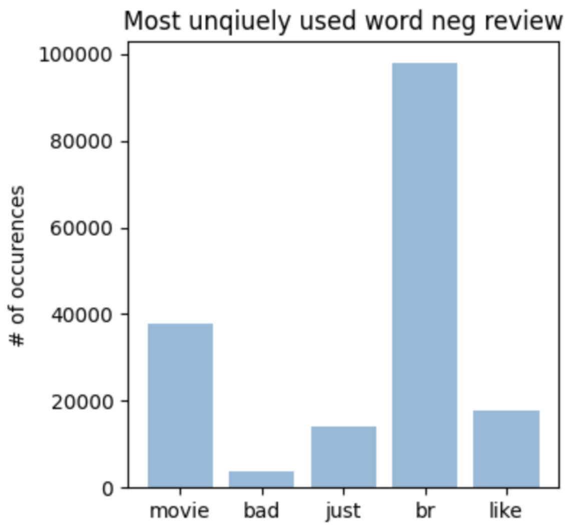
In our project, we used the IMDb movie review dataset. This dataset consists of 50,000 opinionated movie reviews evenly split between positive and negative reviews.

To get a better understanding of the dataset, we examined the most frequently used words in both positive and negative reviews.



This figure did not provide much insight into the differences between positive and negative reviews. Next, we examined the words that were most frequently used in positive reviews but not negative reviews and vice versa.





From these figures, we can make observations such as words like “great” and “love” being indicative of positive reviews.

Additional Note: we observed that “br”, which represents the html <br> tag, is a part of our bag of words. We decided to not remove this tag as it represents additional data which may determine whether a review is negative or positive.

Previous Research

The IMDb dataset is commonly used in machine learning studies. One such study is "Fast and accurate sentiment classification using an enhanced Naive Bayes model" by Vivek Narayanan, Ishan Arora and Arjun Bhatia. The authors used this dataset to train a Naive Bayes classifier with some modifications and achieved an accuracy of 88.8%.

III. Classifiers

The following classifiers were analyzed in this project:

K Nearest Neighbors Classifier

The K-Nearest Neighbors (kNN) classifier is a supervised machine learning algorithm used for both classification and regression tasks. It does not make any assumptions about the data itself and does not build a model during the training phase. kNN rather stores all data points and utilizes them during the prediction phase.

The classifier works by looking at the nearest k neighbors and then classifying the current data point as the majority class label of those k neighbors. k is also known as the hyper parameter of this classifier since it is a parameter for the classifier that can vary according to various factors such as dataset size, data type, etc.

Logistic Regression Classifier

The Logistic Regression Classifier is oftentimes used for binary or categorical classification problems. At the base, the classifier transforms the input into a probability and decides on a certain threshold. This threshold is used to determine the classification of the label.

Decision Tree Classifier

The Decision Tree Classifier works by building a flowchart-like tree where every node splits a feature on a certain threshold. This threshold is determined by finding the best gini score from all possible thresholds.

Once the tree is built based on the training data, it can then be used to predict the class label of any testing data point by starting from the root node, and falling down the branches of the tree.

Feed Forward Neural Network Classifier

A Neural Network has multiple layers of interconnected nodes. The first layer is the input layer, which is followed by one or more hidden layers that finally lead to an output layer.

The input layer's nodes take in the input and calculate an output to the node it is connected to in the next layer based on some biases and weights. This process continues until reaching the output layer which then makes a prediction on the class label of the input. The biases and weights of the intermediate layers are determined by the training phase, where each of them are tuned.

IV. Experimental Setup

The project was conducted in Google Colab, using the Interactive Python Notebook. To start, pip was used to install the datasets library (!pip install datasets), which gives access to the Hugging Face Data Sets. The imdb dataset was imported using the function "load_dataset".

Our goal was to generate and analyze the learning curves for all the classifiers listed above. A learning curve is a graph of Accuracy versus dataset size. To generate a learning curve for a certain classifier, we calculate the training and testing accuracy for that classifier at a certain dataset size. Additionally, we randomly sample for training and testing data at each dataset size step. Fortunately, this is all done for us using the *learning_curve* function in sklearn.

For each classifier, the sklearn *learning_curve* function tracked the learning and validation accuracy while increasing the amount of training data used. For the K Nearest Neighbors Classifier, the hyperparameter, k, was chosen to be 30. For the Feed Forward Neural Network Classifier, the hyperparameter is the amount of hidden layers and was chosen to be three. The hyperparameter for the Decision Tree Classifier is max depth and was chosen to be five. The hyperparameters for the Logistical Regression Classifier are solver, fit_intercept, and penalty which were chosen to be liblinear, true, and 11 respectively.

The parameters passed to the *learning_curve* function were

```
learning_curve(classifier, X, y, train_sizes=np.linspace(10000, 40000, 5, dtype=int), cv=5)
```

where `classifier` is the instance of the classifier we are testing.

Finally, a separate plotting function, LearningCurveDisplay, was also imported from sklearn to display the learning curve graphically with the help of matplotlib.

V. Experimental Results

K Nearest Neighbors Classifier

To tune the hyper parameter, k, we first plotted k vs the accuracy of the model. The results can be seen in Figure 1. Based on the Figure 1, we decided to use k as 30 because of the diminishing returns in accuracy as we increased k further.

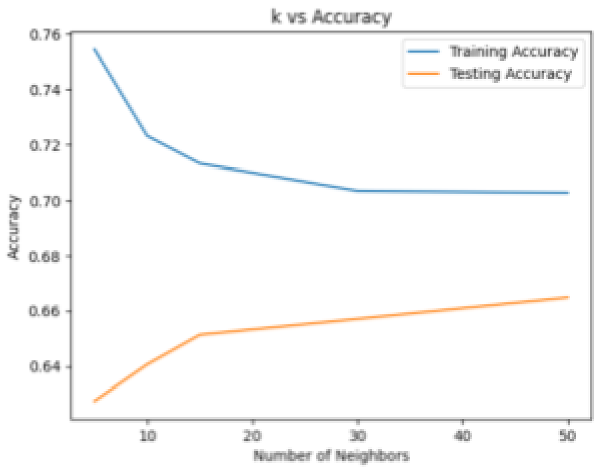


Figure 1: Plotting k versus Accuracy

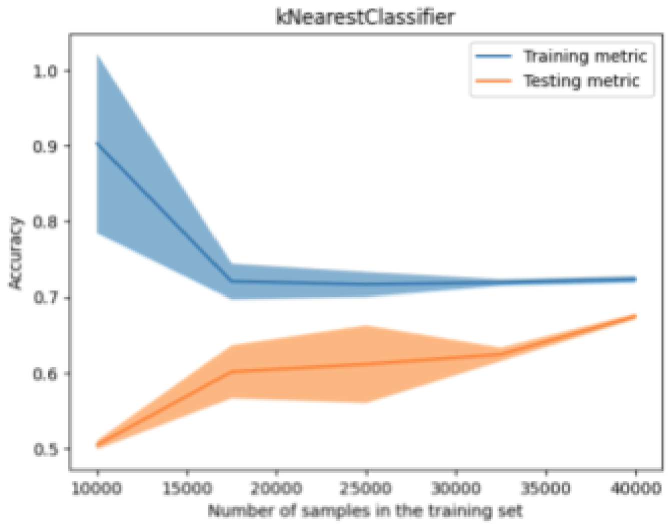


Figure 2: kNN Learning Curve with k = 30

The kNN classifier generally had a low average testing accuracy. kNN models can have lower testing accuracy compared to other models because they make decisions based on a small number of nearest neighbors. If the dataset is noisy or the classes are not well separated, the decision of kNN can be easily influenced by noise, leading to lower accuracy. Increasing the number of training samples used continued to have a positive effect on the testing accuracy. This is because as the number of training samples increases, the 30 closest neighbors are now closer and more specialized to the sample being predicted.

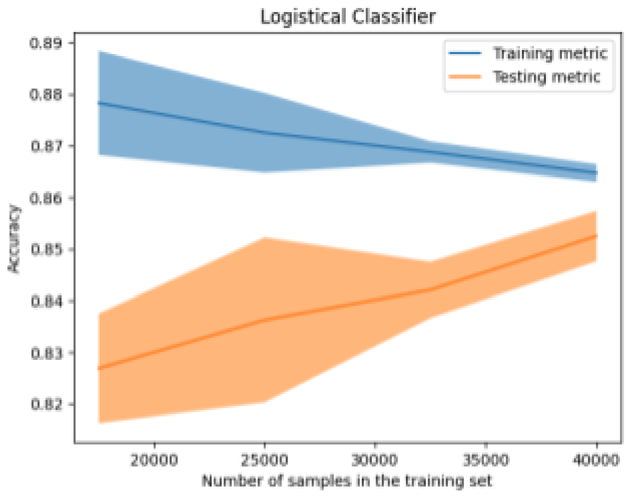


Figure 3: Learning Curve of Logistical Regression Classifier

Logistical Regression

For the Logistical Regression Classifier, we noticed that the testing accuracy had a steady increase from datasets sized 10000 through 32500 and then increased at a slightly higher rate afterwards. In terms of training accuracy, we noticed a steady increase across the entire range of dataset sizes. This could indicate that the model is less likely to overfit data with larger datasets as the training data becomes more generalized with increasing dataset sizes.

Decision Tree Classifier

The results for the Decision Tree Classifier surprised us as the testing accuracy started to have a rate of decrease in testing accuracy for medium sized datasets, but was in tune with the Logistical Regression Classifier and kNN for large sized datasets. Decision Trees can easily overfit to the training data, especially when the tree is allowed to grow without restrictions. For medium-sized datasets, if the decision tree is too complex (deep), it can start overfitting, leading to a decrease in testing accuracy. This is likely why you observed a dip in accuracy for medium-sized datasets. The overall accuracy was generally lower since Decision Trees can also be easily influenced by noise.

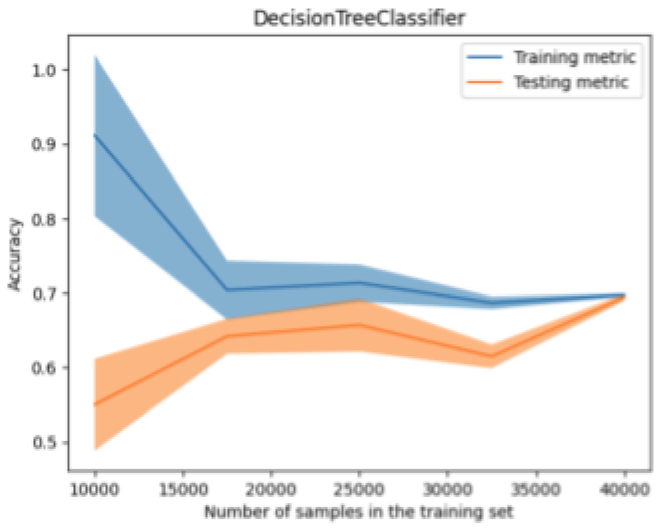


Figure 4: Learning Curve of Decision Tree Classifier

Feed Forward Neural Network Classifier

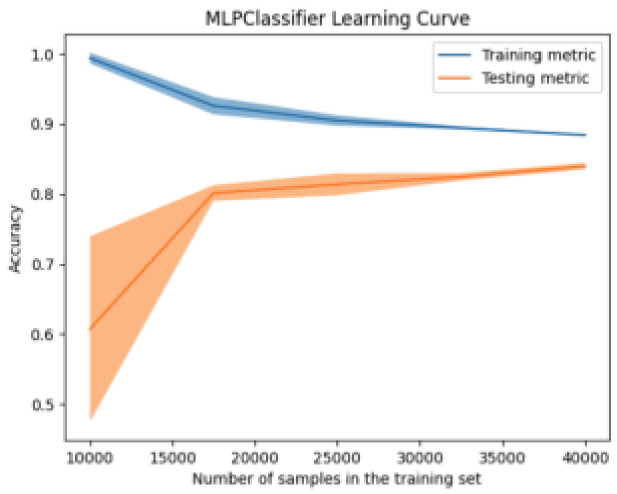


Figure 5: Learning Curve for Feed Forward Neural Network

For the Feed Forward Neural Network, the rate of increase starts to plateau for medium and large datasets. As the dataset size increases, the model has more examples to learn from, reducing the chance of overfitting. However, this can also lead to a plateau in testing accuracy as the model begins to generalize better and overfit less. Generally, the testing accuracy was high for this model.

VI. Insights

The purpose of this project was to analyze the accuracy of various classifiers using varying dataset sizes.

The learning curve of all the models tended to follow a somewhat logarithmic path. When there were few training samples, adding additional samples greatly increased the testing accuracy. But once there was a large amount of samples being trained on, increasing the number of samples used minimally affected the testing accuracy. Moreover, thresholds exist such that adding more training data affected the testing error negligibly.

The kNN classifier continued to benefit from increasing the amount of training data. This is because additional training data allowed the kNN classifier to better fit the data. The Logistical Classifier benefitted the most from using the whole dataset. This is because the additional training data allows the logistical classifier to more closely fit the data. The Decision Tree Classifier did not benefit from increasing the amount of training data. This is because the rigid boundaries of the Decision Tree Classifier struggled to fit the data enough to create accurate predictions. Lastly, the benefit of adding additional training data when training the Neural Network. We believe that we could have added additional complexity when training the model to make the Neural Network better fit the data.

VII. Contributions

Safeeullah Saifuddin:

- Generating the learning curve for kNN and Decision Tree Classifiers
- Tuning the hyper parameter for kNN
- Report Formatting and Editing

Nathan Huey

- Creating a bow (bag of words) representation for imbd movie reviews
- Creating a method to visualize learning curves using sklearn
- Data Description

Aman Grewal

- Generating learning curve for feed forward neural network.
- Tuning hyper parameter for feed forward neural network
- Implemented learning curve function
- Insights
- Experimental Setup