## Framing the Problem

The objective of this project is to discover meaningful insights in the IMDB datasets, and train a machine learning model that can reliably predict whether a review has a positive or negative sentiment. Since movie reviews and their corresponding labels are provided, this is an example of a supervised learning problem. Unsupervised machine learning algorithms can still be used in the exploratory data analysis section to discover interesting information (e.g., clusters of labels). The problem involves a binary target, positive or negative, so it is a classification problem.

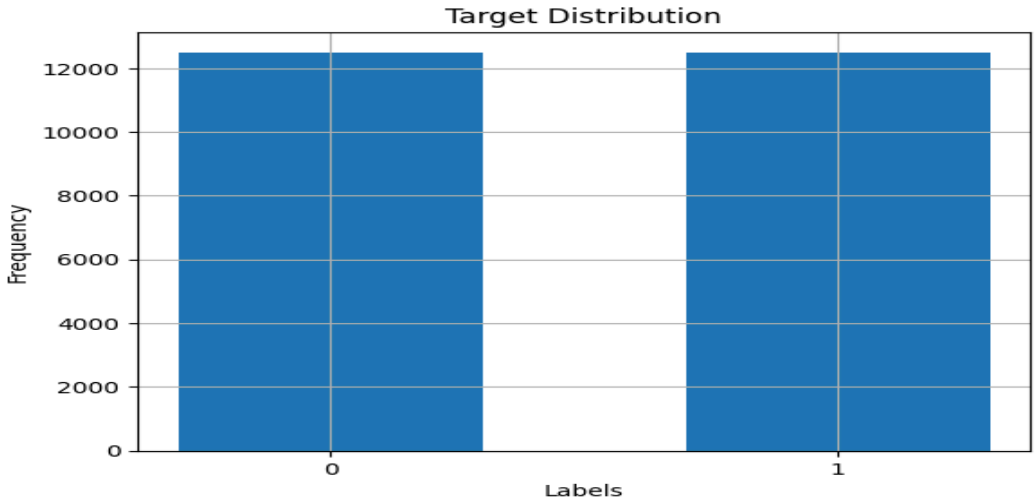
Performance will be measured using the model's accuracy, which is a common metric to use in binary classification. Specifically, accuracy on a validation set (a subset of the original training set) will be used to compare candidate models, and accuracy on a test set will be used to get a final measure of the model's reliability for predicting a review's sentiment.

## Getting the Data

The IMDB dataset is available in the tensorflow.keras.datasets.imdb module. There is a load_data() function in this module that returns this dataset as an iterable of 2-tuples with the format (X_train, y_train), (X_test, y_test). The Keras documentation states that the data is sequential, implying that the reviews are ordered from left to right. The documentation also states that each word in a review is replaced with an integer, starting with integer 2. If a word is replaced with a 2 it means the word is the most common word in the training data set, a 3 means the word is the second most common word, and so on. A 1 represents the start of a review, so each list in the array starts with a 1. A 0 represents a pad token.

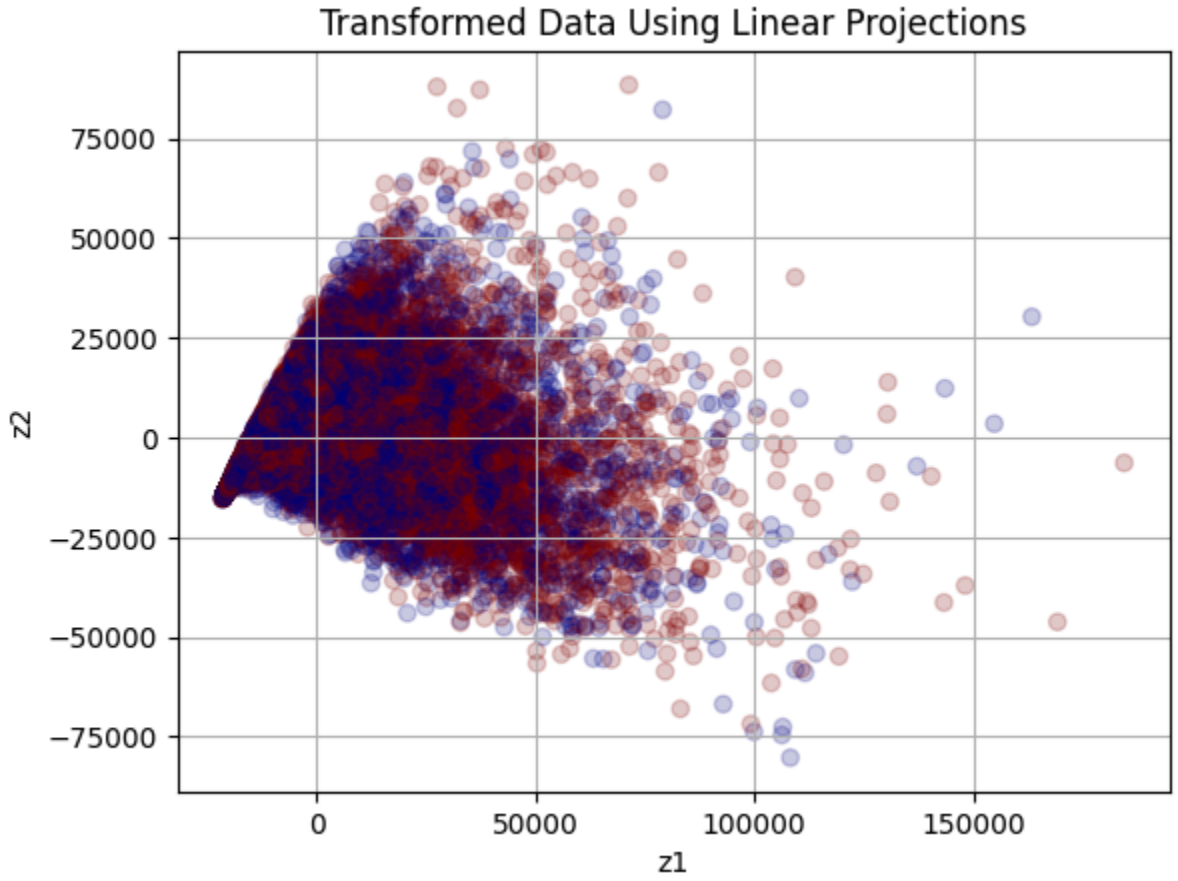
## Exploratory Data Analysis

The reviews have varied lengths, so they were padded with zeros to match the length of the longest review. There are 50,000 reviews in total (25,000 for training and another 25,000 for testing). The class labels are balanced, meaning that the count of positive and negative reviews are equivalent.



The longest review had a total of 2,493 words! The start of sequence token was removed since it does not provide any meaningful information to a model's training algorithm and can slow down training. The sentiment for a review is encoded in y_train and y_test with either a 0 (negative) or a 1 (positive).

Dimensionality reduction was used to visualize the data in 2 dimensions to identify potential patterns in the data set. Identifying patterns in a reduced dataset with 2 or 3 features is important because it can help give you a better understanding of the data (e.g., by identifying clusters in the data). Unfortunately, there were no meaningful patterns in the visualization of the reduced dataset I used.



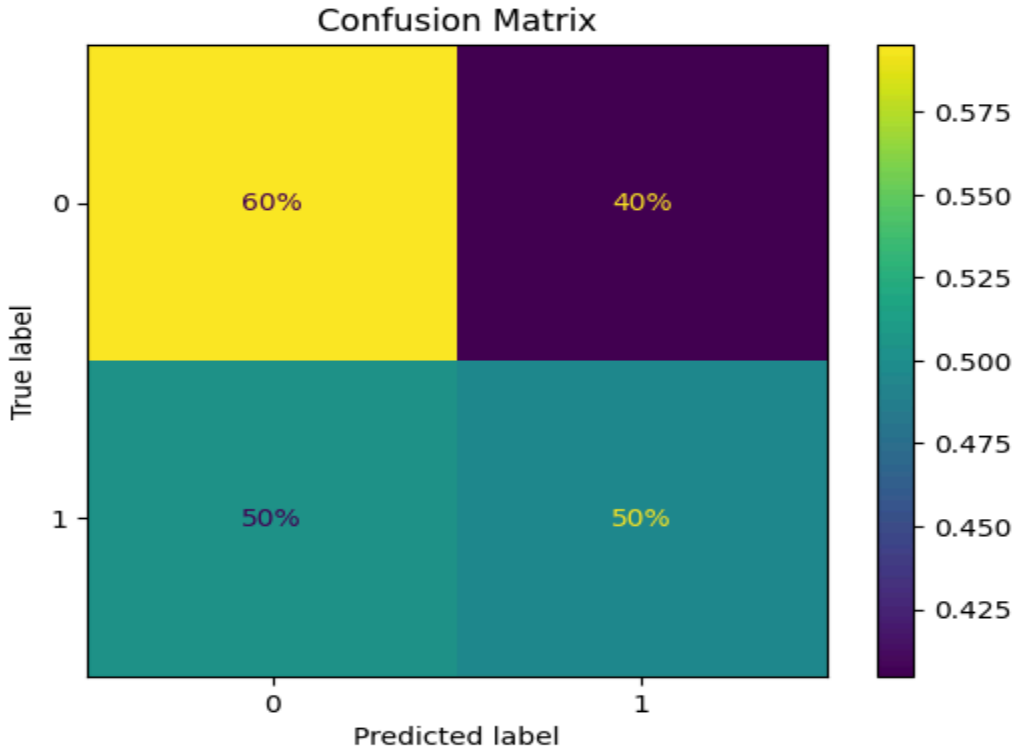

Machine Learning

Decision trees in scikit-learn are binary trees. Predictions are made by traversing the binary tree vertically from the root node down to a leaf node. At each interior node (also called a split node) a comparison operation is done between a threshold feature value and an instance's feature value. If the result is true then the interior node's left child is visited during traversal, otherwise the right child is visited. Once a leaf node is reached, a prediction is made by looking at proportions of class labels corresponding to the subset of the training data that the leaf node received. The class with the highest proportion is the predicted class.

Scikit-learn uses a recursive greedy algorithm called CART to grow/train a decision tree. The main aspect of the algorithm is in how it selects an optimal feature value for the comparison operation at split nodes. The algorithm goes through all feature values of all input training instances at each interior node. For each feature value a loss is computed whose result depends on a weighted sum of the gini impurity of the left and right subsets of input training instances. The subsets are formed by using the same comparison operation as during prediction. The feature value that results in the minimum loss is selected as an interior node's threshold feature value.

Random forests are ensemble methods, which are a class of machine learning algorithms that make predictions by aggregating predictions of the ensemble's base predictors. Within the context of random forests these base predictors are all decision trees that are trained and make predictions in the same exact way as described above. A key difference between random forests and decision trees in scikit-learn is in how training instances are used. With random forests only a subset of features and instances are used to train each base decision tree by default. The subset of features and instances to search through is determined by using bootstrap random sampling.

Default decision trees and random forests were trained and evaluated. Their performance was only slightly better than random guessing (which corresponds to 50% accuracy). A 2% accuracy boost was achieved after tuning the hyperparameters of the random forest classifier. The best model had an accuracy of about 55% on the test data. Here is the confusion matrix that corresponds to its predictions on the test data:



Of all instances in the test set that had a negative label 60% were correctly classified as negative and 40% were mistakenly classified as positive. Of all positive labels in the test set 50% were correctly classified as positive and 50% were mistakenly classified as negative.

One great aspect of decision trees and random forests is that they can be used to provide a measure of each feature's importance or usefulness. The most interesting discovery in this project was that words more towards the beginning of a review were more important than words that were towards the end of reviews for predicting a review's sentiment.