Supervised Learning

## Introduction

In this paper different supervised learning algorithms are applied to two different datasets and the results of the algorithms are evaluated. The learning algorithms used include decision tree, neural network, boosted decision tree, support vector machine, and K-nearest neighbor. The performance of each model is compared and analyzed. Additionally, techniques to improve the testing accuracy of each model are discussed along with alternative algorithms.

## Datasets

The two datasets used for evaluation are fall detection and sentiment analysis. The fall detection dataset was obtained from Kaggle and contains 164,000 sample data points which are bodily metrics of patients along with the labels of their physical state. There are five labels in total which include 0- Standing 1- Walking 2- Sitting 3- Falling 4- Cramps 5- Running. The data was obtained from Chinese hospitals and is based on old age patients. There are six features in the dataset which include the monitoring time, sugar level, EEG monitoring rate, blood pressure, heart rate, and blood circulation. All the features are recorded as numbers.

This dataset was interesting because it used numeric measurements of different biological activities to indicate whether a patient could be at risk of falling down. More importantly, using a model to predict a patient's risk of falling can easily save doctors and medical professionals a lot of time from analyzing the results themselves. If the data was collected on different measurements such as sugar level, BP, and heart rate, a doctor could run those features against the model and retrieve a prediction on their patient's state. Therefore, the patient can be handled with more care or caution if they are in a more critical or handicapped state. Additionally, the availability of different features in the model gives more scope to explore which features have a higher influence on a patient's state than others.

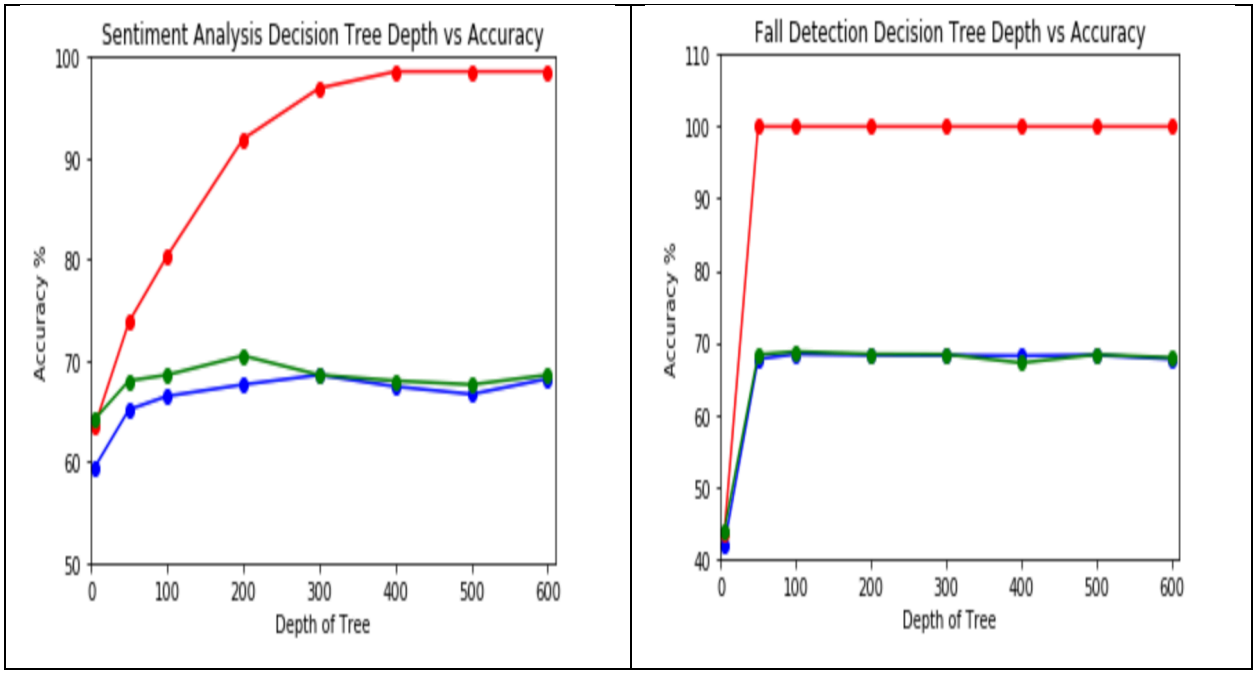
The second dataset was sentiment analysis and it was retrieved from the UCI machine learning repository. This dataset was a collection of text data containing reviews from IMDB, Yelp, and Amazon, and each text was either labeled with a 1 indicating that it was positive or 0 meaning that it was negative. There were 3000 reviews in the dataset and the reviews ranged from a few words to a couple of sentences. Some of the reviews consisted of incomplete sentences and short remarks while others were more fleshed out thoughts. The reviews were equally distributed over the topics of movies, local businesses, and online products. This dataset used a bag of words model for its feature vector and had 3269 features which are all the words appearing in the text.

This dataset was interesting because it used human language in the form of reviews to decipher an emotion or sentiment of a person towards a particular experience. While it may be easier for humans to understand sentiments, a machine has to learn to differentiate positive emotions from negative. The way it can do so is by using the words in the reviews as features and determining how important each word is towards predicting the sentiment of a human review. This bag of words model is unique as it contains thousands of features and helps generalize a model's prediction capability by giving it more data to look up. Additionally, it can also make the model susceptible to overfitting due to the abundance of features available. If a feature or word does not contribute to sentiment analysis, it may have no use in the model and can lead to incorrect predictions.

Each dataset was split into training, validation, and test sets using the following ratio 70:15:15. After fitting each classifier on the training set, the classifier's hyperparameters were tuned based on the validation score and then the classifier was evaluated on the test set. For all the graphs below, red, blue, and green represent the training, testing, and validation accuracy.

**Decision Tree**

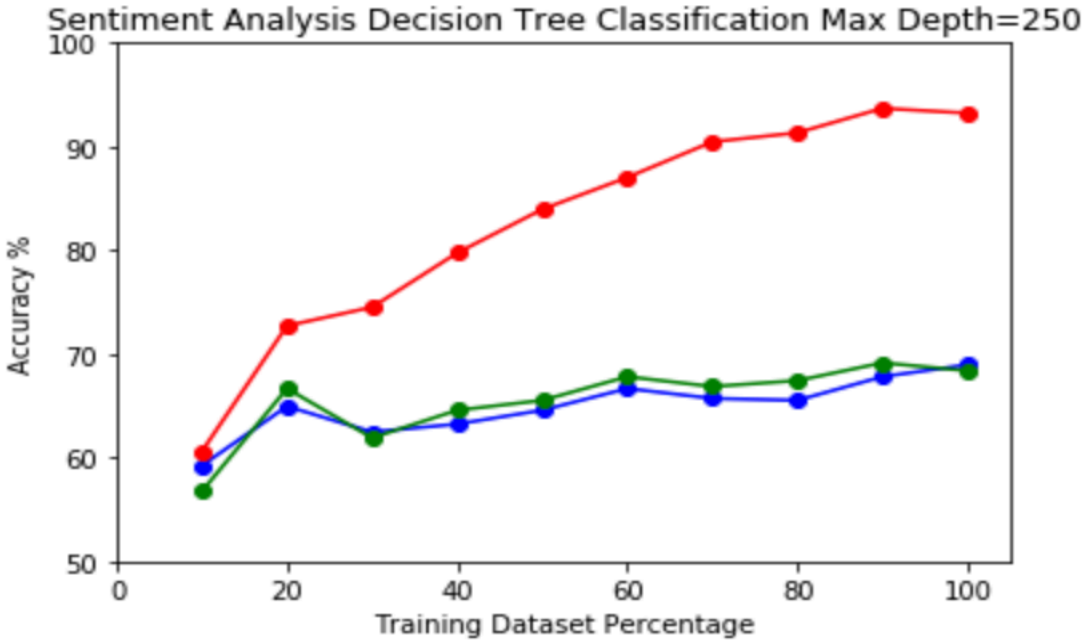
For the decision tree classifier, information gain was used to measure the quality of the split. Information gain tells us which attribute in a feature vector is most important for discriminating between the classes to be learned. Pre Pruning was also implemented by varying the depth of the tree. Pre Pruning is used to avoid overfitting in decision trees by limiting the max number of nodes or the depth of the tree which thereby limits the tree from creating nodes that split on bad features and does not generalize while predicting classes. Consequently, the performance of the tree was tuned based on its maximum depth.

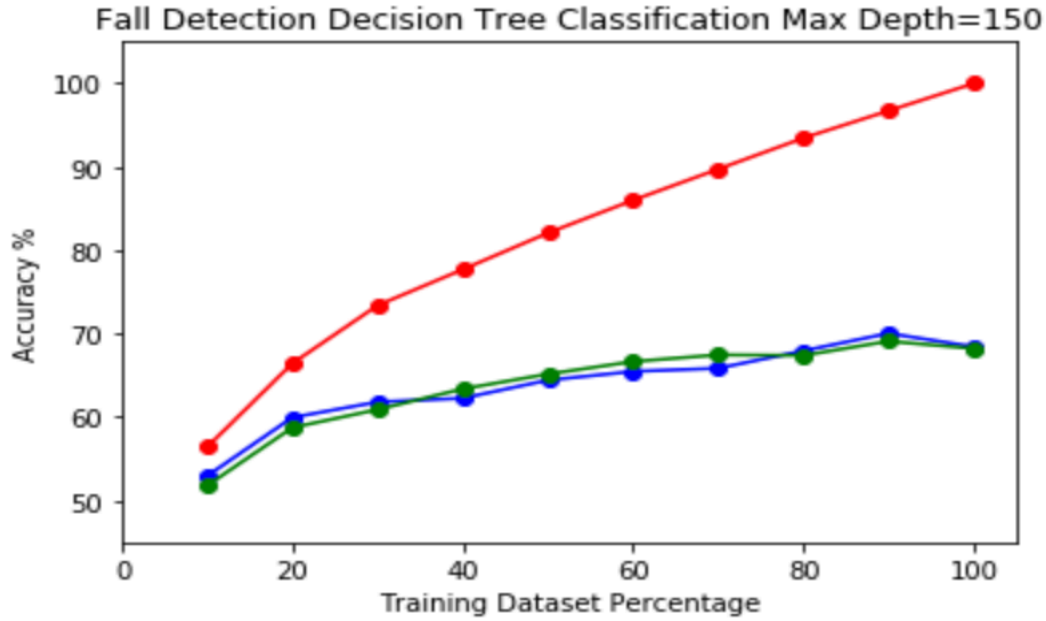



The results of varying the depth of the tree show that overfitting becomes more prevalent as the depth of the tree increases. This result concurs with the ID3 algorithm's preference bias which states that the smallest tree that correctly classifies all the training examples is the best. This is shown in the graphs above where the testing accuracy remains pretty much the same after the depth of the tree increases beyond 100. The best tree depth for sentiment analysis and fall detection was 250 and 150 respectively. A smaller tree depth is needed for the fall detection dataset because the set has significantly fewer features which means that it requires fewer nodes to split on and therefore needs a smaller tree depth.

Using the optimal depths, the decision tree classifier was then applied to different sizes of the two training sets. Overfitting becomes more apparent as a larger percentage of the training set is used for training the classifier. With a greater amount of training data, the classifier begins to look at new features and creates more nodes and branches in order to label them correctly and therefore becomes less generalizable. Consequently, the testing accuracy grows slowly compared to the training accuracy. The overfitting in sentiment analysis is due to a large number of features in the dataset. Since the features represent all the words found in the dataset vocabulary, the

feature space has a large number of attributes which causes the classifier to find meaningless regularity in the data. The overfitting in fall detection occurs due to its large number of training instances which may contain noisy or bad data and make the classifier more biased to the training set. In order to overcome overfitting, KFold validation can be used to average predictions across different trees that split on different features and use different data.

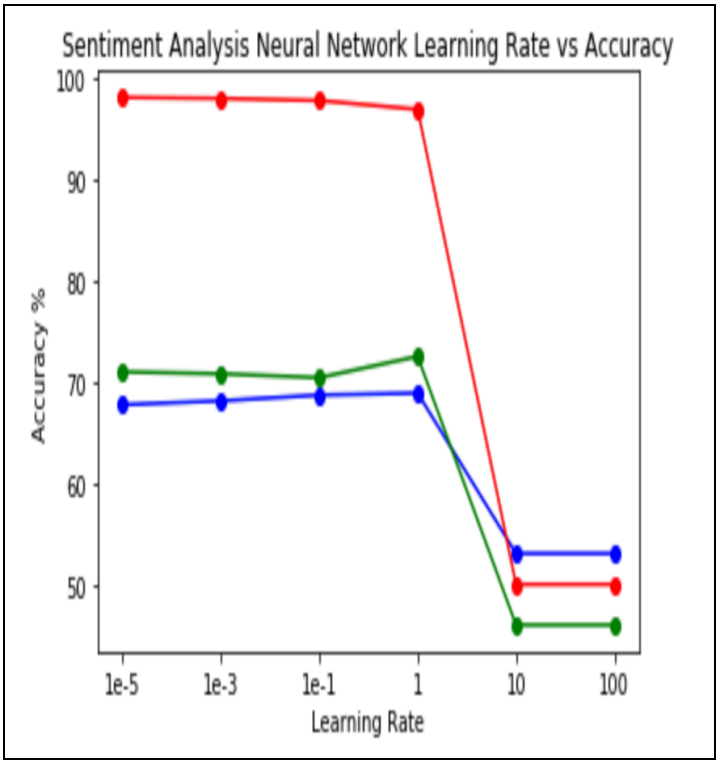


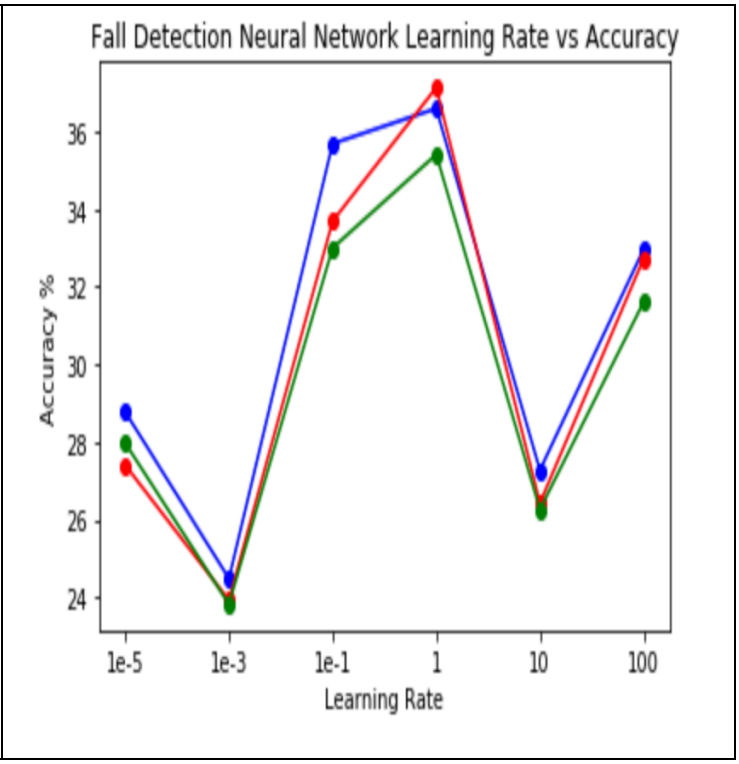




| Set | Max Depth | Training Size Percentage | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|
| Sentiment Analysis | 250 | 100 | 93.23 | 68.95 |
| Fall Detection | 150 | 90 | 100 | 70.04 |

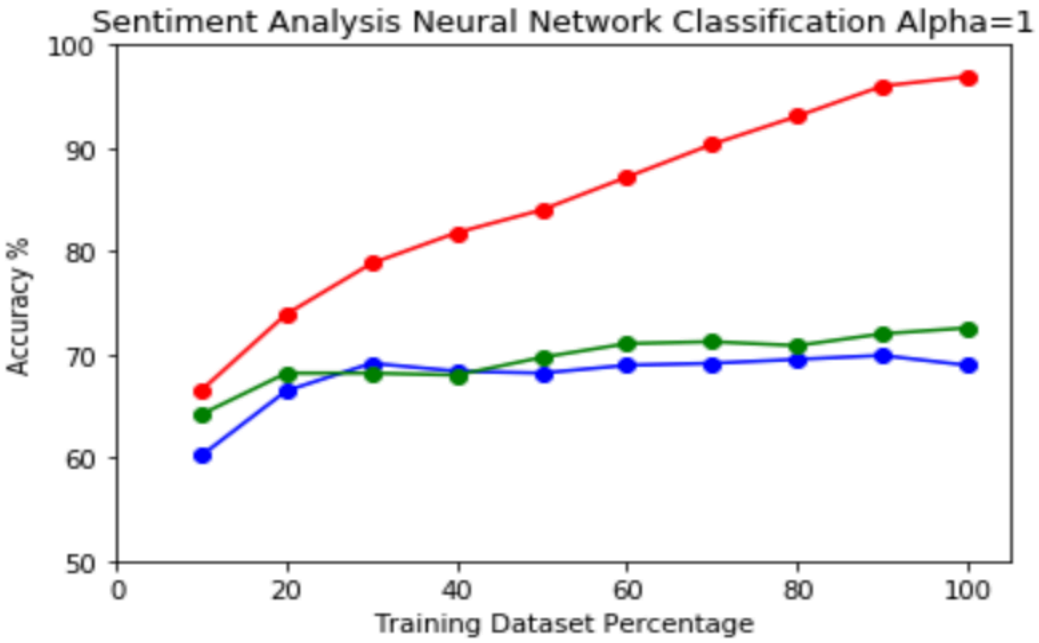
**Neural Network**

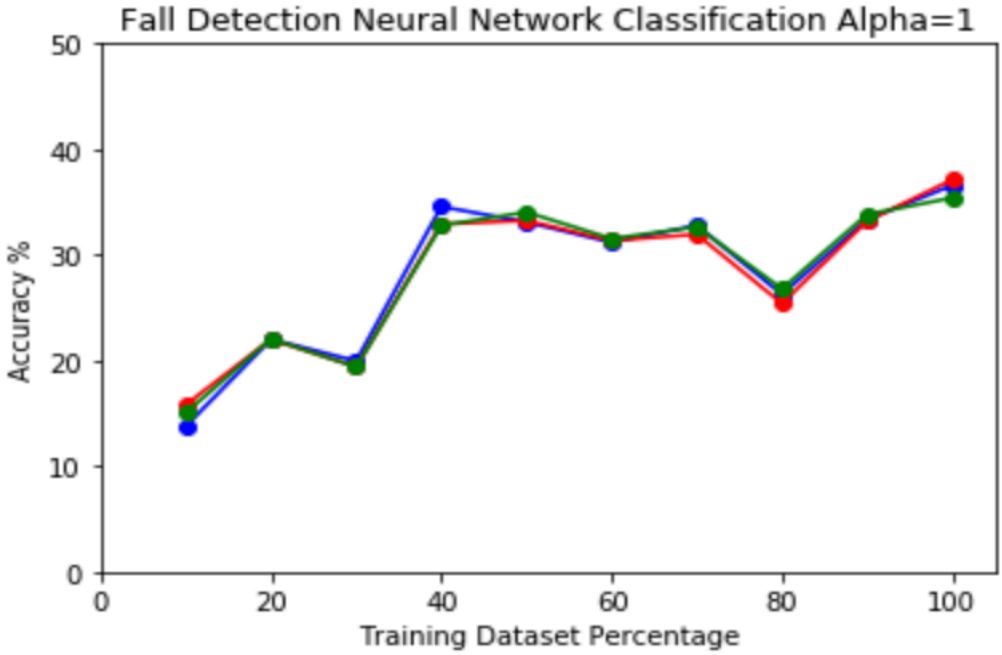
A multilayer perceptron (MLP) classifier was fitted on the two datasets. The advantage of using a neural network is that it has the capability to learn and classify nonlinear models such as fall detection which has more than two classes. For the MLP classifier, the solver adam was used for weight optimization, relu was used as the activation function, and two hidden layers were used for each classification problem. Additionally, the learning rate of the neural network was varied to see the effect on prediction accuracy.




From the results above, the best learning rate for sentiment analysis and fall detection was found to be 1. It is also evident that overfitting vanishes in sentiment analysis as the learning rate goes above ten, and overall there is less sign of overfitting in the fall detection model. Since neural networks create complex models to solve classification problems, they require thousands of training points to overcome overfitting. Since the sentiment analysis dataset is only composed of a few thousand training samples, the neural network isn't able to generalize for all types of data points. On the other hand, the neural network is able to avoid overfitting in larger datasets such as fall detection which have more than a hundred thousand training samples.



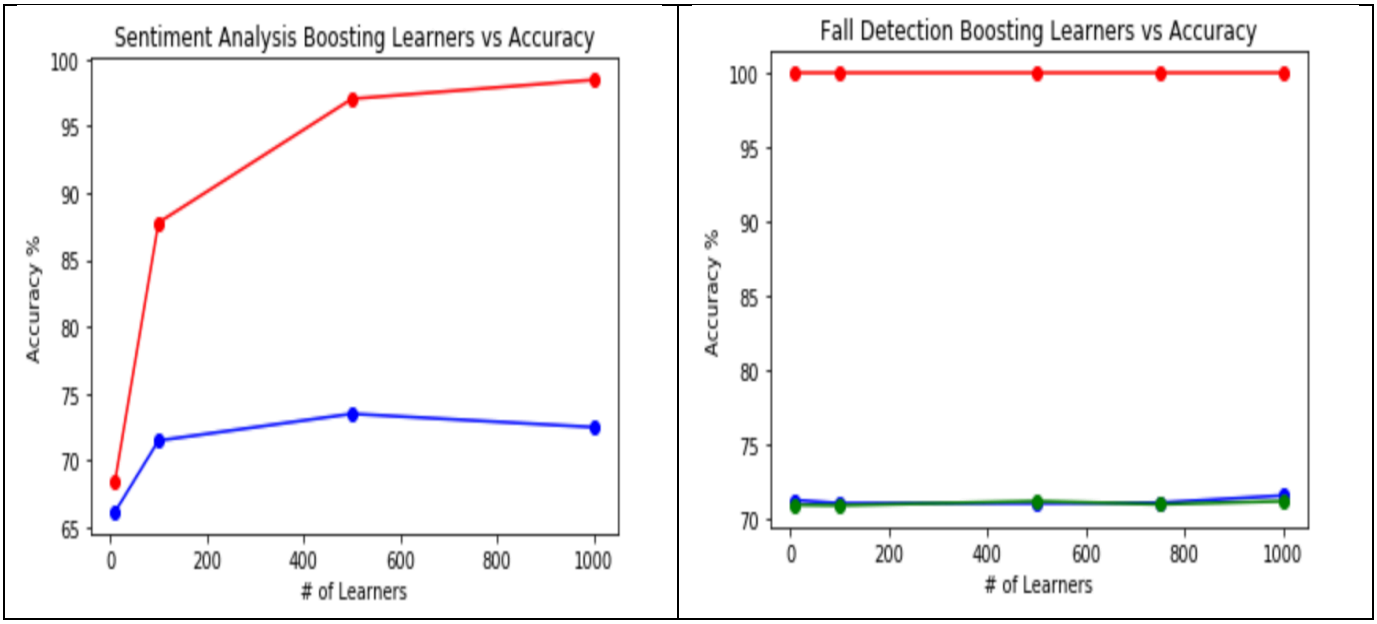

| Set | Learning Rate-Alpha | Training Size Percentage | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|
| Sentiment Analysis | 1 | 90 | 96.00 | 69.9 |
| Fall Detection | 1 | 100 | 37.12 | 36.59 |

Despite preventing overfitting in the fall detection problem, the neural network performs best on sentiment analysis. Since sentiment analysis is a linear model, it requires a simple classifier with fewer layers and fewer neurons in each layer. Increasing the number of layers will make the classifier more complex and lead to overfitting. On the other hand, fall detection is not linearly

separable and requires a more complex model with a greater number of layers and neurons in each layer to have an improvement in testing performance. Due to time constraints of running the neural network on a large number of layers, only two layers were used with 10 and 8 neurons in the first and second layers for sentiment analysis and fall detection.

**Boosting**

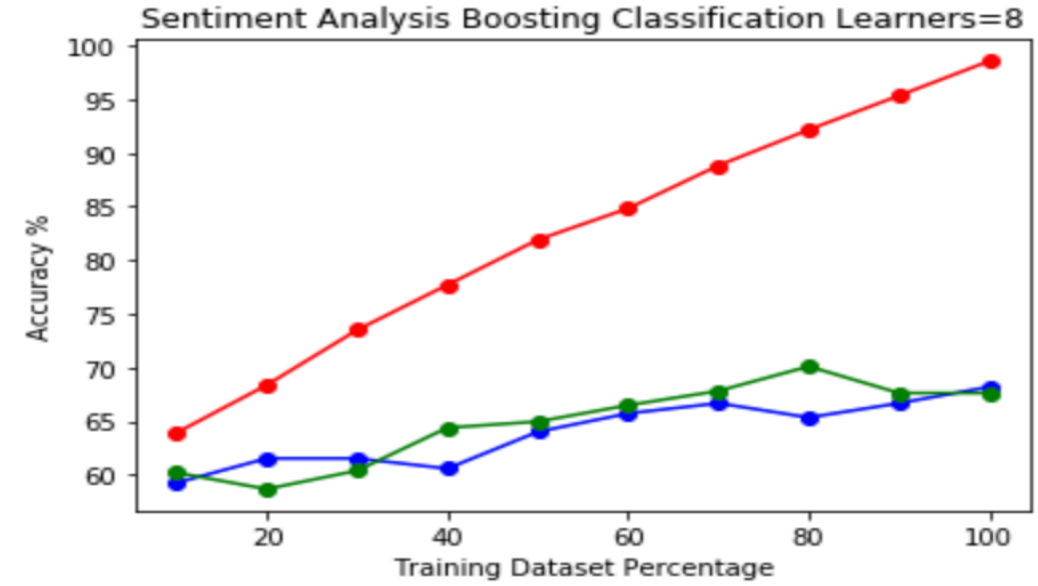
A gradient boosted tree classifier was used to train the two models and the number of learners in the model parameter was varied to see the differences in prediction accuracy. Boosting uses non-complex weak learners to create a high performance classifier and resists overfitting by driving the test error to zero even after the training error has reached zero. Boosted decision trees require very less parameter tuning and work well when the complexity of the classifier is low which can be done by limiting the tree depth. In order to prune the trees, the maximum tree depth used for sentiment analysis and fall detection was 250 and 150 respectively.

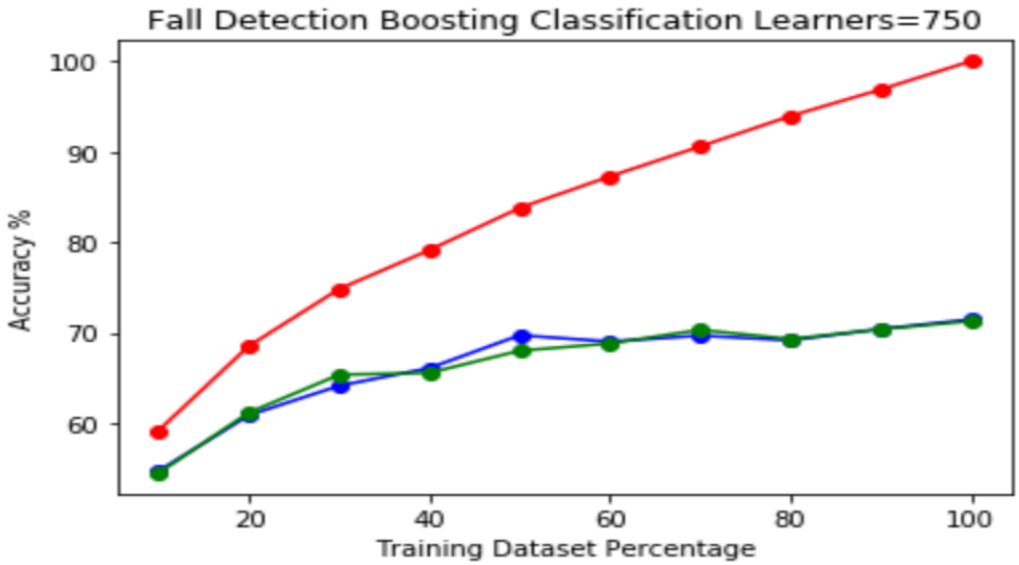



The graphs above show that sentiment analysis has the best validation accuracy when the number of learners is equal to 500 and for fall detection the optimal number of weak learners is close to 10. More learners are required for sentiment analysis due to its large number of feature attributes which can create more outliers in the dataset, thereby requiring more weak learners to correct the labels of the outliers. Using these number of weak learners for the two datasets, the boosting classifier was applied to different training dataset sizes.

It is evident from the results below that as the training accuracy increases so does the testing accuracy. This shows that the boosting classifier continues to drive the testing error to zero even after optimizing the training error. The results also show that increasing the boosting iterations, which in this case is the training set size, improves the testing performance. The testing accuracy increases more quickly for sentiment analysis from 50% to 100% of the training set size while the rise in accuracy is much slower for fall detection. This can be due to the fact that the fall detection dataset has more training samples and therefore more noise in its data. Since boosting is prone to noise and the decision tree classifier already fit the dataset quite well, the boosting classifier could risk overfitting the data as the number of training samples increases. Therefore, cross validation could be used to improve the fall detection testing performance by averaging the

estimates across different boosted tree classifiers. In order to improve the sentiment analysis performance, more training samples can be added to increase the number of boosting iterations and continuously drive up the testing accuracy.

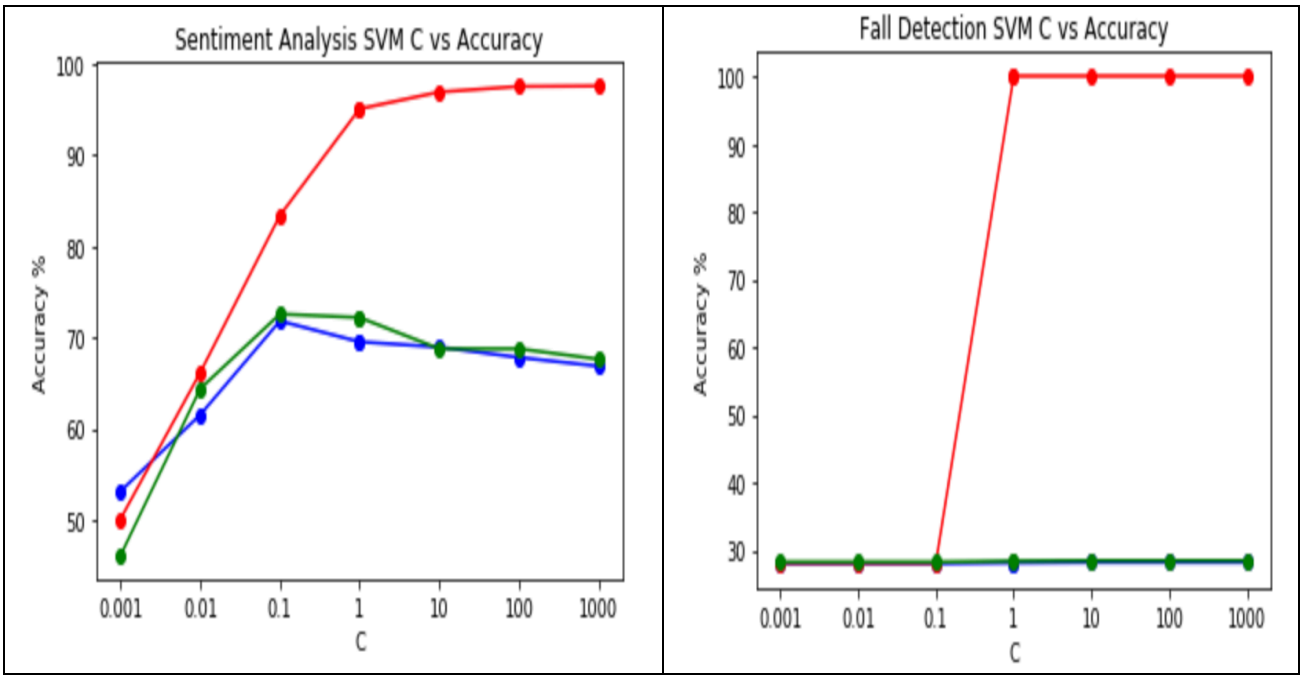


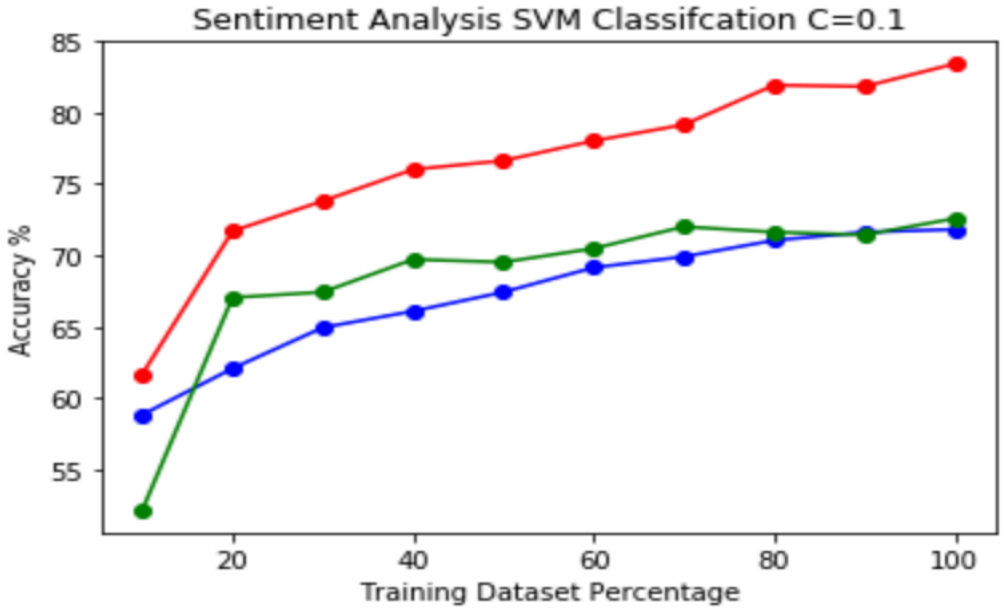

| Set | # of Weak Learners | Training Size Percentage | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|
| Sentiment Analysis | 8 | 100 | 98.56 | 69.70 |
| Fall Detection | 750 | 100 | 100 | 71.46 |

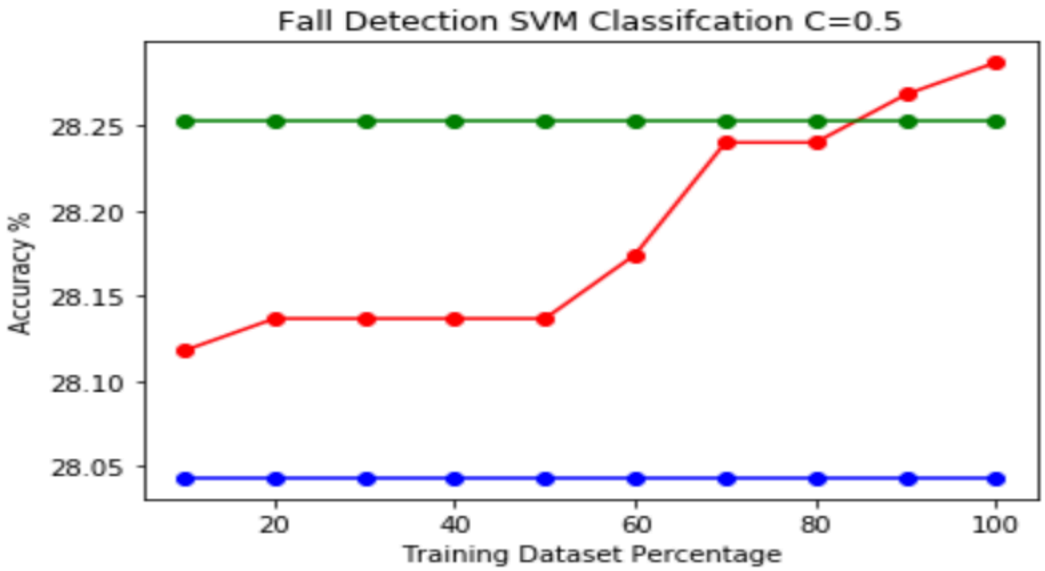
**Support Vector Machines**

SVMs have great generalization capabilities as they strive to find the maximum margin between the decision lines of the classes they are trying to predict. Although SVMs are linear separators, they can use a kernel function in order to fit nonlinear decision boundaries. Therefore, a linear kernel was used to train the sentiment analysis model and an rbf kernel was used to train the fall detection model. The value of the hyperparameter C was varied in the SVM to see its effect on the prediction accuracy.




The C parameter tells the SVM classifier how much you want to avoid misclassifying each training example. For both the models, overfitting begins to occur as C becomes greater than 0.1. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Therefore, the model will find a decision boundary to predict most of the training points correctly but not the testing points. Conversely, a very small value of C will cause the classifier to look for a larger-margin separating hyperplane which will lead to more points being misclassified if they lie on the wrong side of the decision margin. Therefore, with smaller values of C, incorrect predictions become more common and the prediction accuracy starts to decrease.



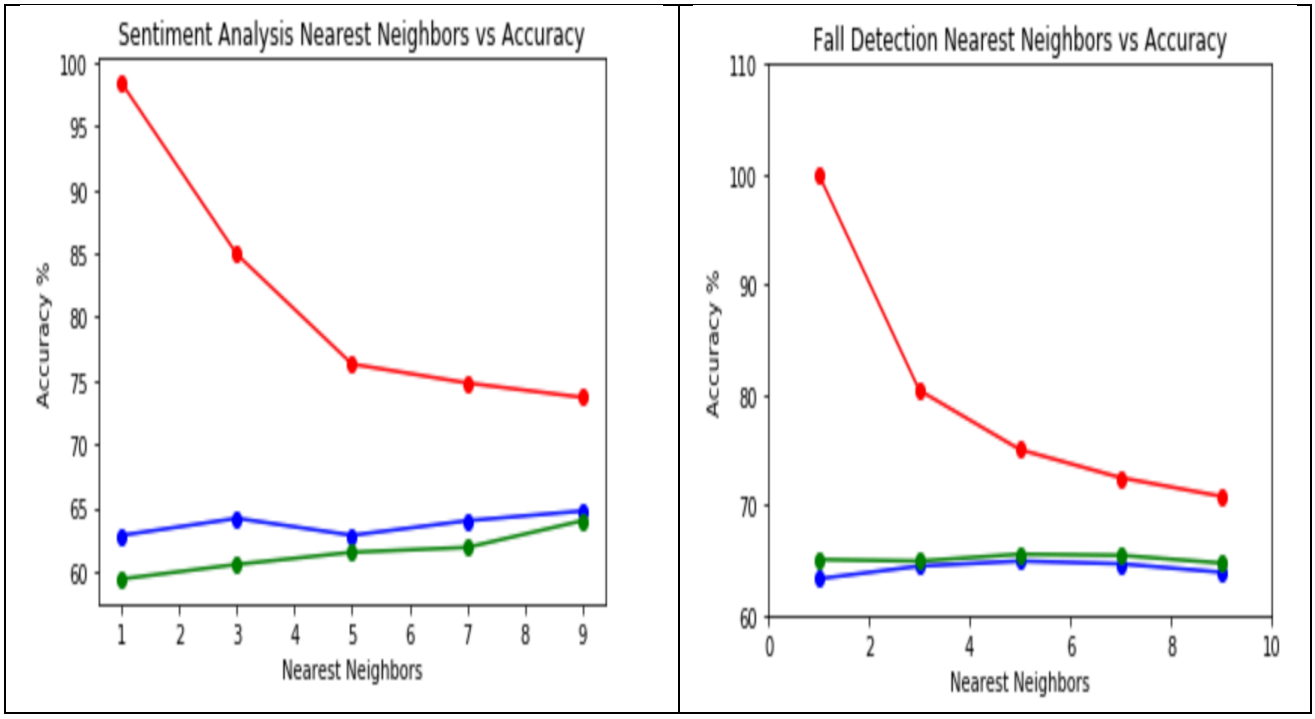



The optimal value for C was equal to 0.1 and 0.5 for sentiment analysis and fall detection respectively. Since the sentiment analysis data is linearly separable, the linear kernel performs well in increasing the testing accuracy as the training dataset becomes larger. SVMs usually tend to perform better on datasets with a small number of training instances and a large number of feature attributes. Since the fall detection dataset has a significantly fewer number of features and more training instances that that of sentiment analysis, the SVM's performance doesn't improve while trying to fit the fall detection model. One way to improve the performance could be to use boosting with SVM as the SVM can be used as a weak learner for training the fall detection model.

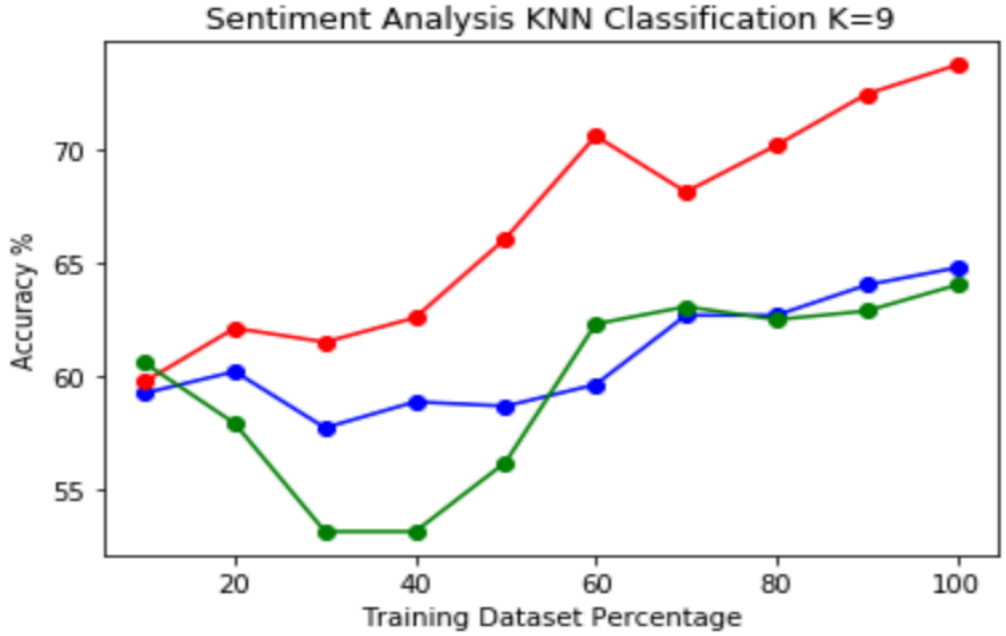
| Set | C | Training Size Percentage | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|
| Sentiment Analysis | 0.1 | 100 | 83.43 | 72.00 |
| Fall Detection | 0.5 | 100 | 28.28 | 28.04 |

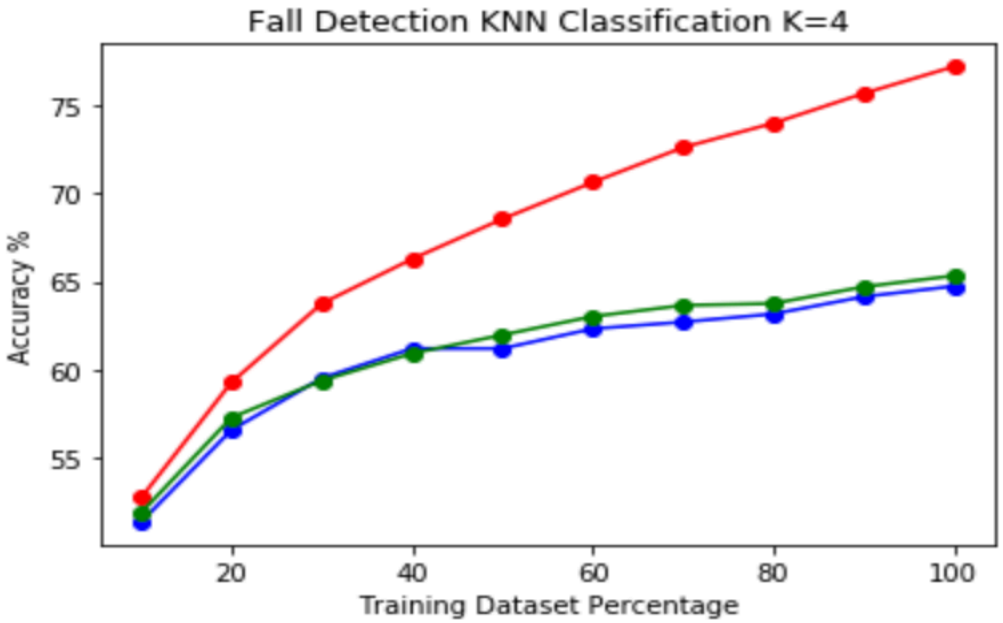
**K-Nearest Neighbor**

K-nearest neighbor is an instance based learning algorithm that uses the simple heuristic that points closer to each other have similar labels to make predictions. A new unlabeled point is assigned the most common label of its k nearest neighbors, and the value of k can be varied to make the most accurate prediction on a particular dataset. For the two classification problems, the nearest neighbor parameter was varied in the classifier to observe differences in prediction accuracy.




The best nearest neighbor parameter (k) was determined to be 9 and 5 for sentiment analysis and fall detection respectively. Since the sentiment analysis model has thousands of features, the nearest neighbor of one feature vector may be quite different from the feature vector itself. Also if a neighbor of a point has similar looking features but a different label, then the point can be misclassified. Therefore, using more neighbors allows a feature vector to find different similarities with its neighbors and allows it to have a more comprehensive view of its neighborhood. Fall detection uses a smaller number of nearest numbers because it only has 6 features as opposed to a couple thousand. Therefore, it requires a much smaller neighborhood to predict its label as the number of variations in feature attributes is significantly less.






The KNN classifier performs better when the number of training instances increases as shown above. As the training set size increases, so does the number of neighbors around a data point which gives a more accurate view of where similar features lie in relation to the data point. Improving the performance of the KNN classifier would require adding more training instances to the dataset.

| Set | Neighbors | Training Size Percentage | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|
| Sentiment Analysis | 9 | 100 | 73.69 | 64.76 |
| Fall Detection | 4 | 100 | 77.16 | 64.74 |

**Conclusion**

In addition to tuning the hyperparameters of each classifier to obtain an improvement in performance, the dataset itself was also cleaned up before training. For the sentiment analysis dataset, stop words were removed from the text and stemming was used to normalize all the vocabulary in the text. This was done to remove words and symbols that could be considered as bad features and also to reduce the feature size by converting words with similar roots to the same root word.

K-Fold cross validation was also performed on each dataset for each classifier. The value for k was chosen such that each train/test group of data samples was large enough to be statistically representative of the broader dataset. Therefore, a smaller K of 3 was used for the smaller dataset which was sentiment analysis and a large K of 5 was chosen for fall detection. K-Fold validation usually results in a less biased and less optimistic estimate of the model skill than other methods, such as a simple train/test split.

| Classifier | K-Fold Testing Accuracy for Sentiment Analysis K=3 | Best Testing Accuracy for Sentiment Analysis | K-Fold Testing Accuracy for Fall Detection K=5 | Best Testing Accuracy for Fall Detection |
|---|---|---|---|---|
| Decision Tree | 75.77 | 68.95 | 87.59 | 70.04 |
| Neural Network | 76.04 | 69.90 | 31.5 | 36.59 |
| Boosted Decision Tree | 75.77 | 69.71 | 88.9 | 71.46 |
| SVM | 71.23 | 72.00 | 28.23 | 28.04 |
| KNN | 65.43 | 64.76 | 73.4 | 64.74 |

Using K-Fold cross validation helped improve the accuracy of decision trees and boosted decision trees the most. Since each cross validation fold has random data, trees fit on each data set differently and have differing depths. Averaging the predictions across different trees makes the model less prone to overfitting and therefore the testing accuracy improves with the training accuracy.

| Classifier | Best Testing Accuracy for Sentiment Analysis | Best Testing Accuracy for Fall Detection | Average Predictive Accuracy | Training Speed | Prediction Speed |
|---|---|---|---|---|---|
| Decision Tree | 68.95 | 70.04 | Higher | Increases with maximum depth | Fast |
| Neural Network | 69.90 | 36.59 | Lower | Increases with more layers | Fast |
| Boosted Decision Tree | 69.71 | 71.46 | Higher | Increases with more learners | Fast |
| SVM | 72.00 | 28.04 | Lower | Increases with C | Fast |
| KNN | 64.76 | 64.74 | Lower | Fast | Increases with K |

The SVM classifier performed the best for the sentiment analysis model and the boosted decision tree performed the best for the fall detection model. Overall the boosted decision tree was a high performing classifier due to its ability to drive up the training and validation accuracy despite reaching zero training error. SVMs, when used with a linear kernel, are great for finding the most generalizable linear decision boundaries and reducing the bias typically provided by linear classifiers. The neural network underperformed for the fall detection model because it needed more layers and neurons, which required more time and computing power. The boosted decision tree did improve the performance of regular decision trees by combining weaker decision tree classifiers into a strong learner. KNN performed the same for both of the datasets and the results show that its performance keeps improving with a large training set size which means that increasing the number of training instances will continue to improve the KNN performance.

**References**

Sentiment Analysis Dataset
https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences
Fall Detection Data Set
https://www.kaggle.com/pitasr/falldata
A Gentle Introduction to k-fold Cross-Validation
https://machinelearningmastery.com/k-fold-cross-validation/