A Comparison of Machine Learning Models Applied to Binary Classification Problems

Author	**Caelean Barnes**
Degree	B.S. Computer Science/Minor Cognitive Science
Institution	UC San Diego, La Jolla, CA
Contact	cgbarnes@ucsd.edu

Abstract

Machine learning is easier than ever to access. With the widespread availability of free and well documented implementations, anyone with an interest in machine learning can create classifiers and train them on the data set of their choice. The following paper describes the methodology and results of running three standard machine learning libraries on four unique classification problems. The three classifiers in question are standard implementations of Support Vector Machines, K Nearest Neighbors, and Artificial Neural Networks. The problems are binary classification problems derived from various data sets taken from the UCI Machine Learning Repository [3]. Across all problems, K Nearest Neighbors performed the best, with Support Vector Machines coming second and Artificial Neural Networks a close third.

1 Introduction

The comparison of the different models for classification is intended to provide some insight into how successful each classifier is at accurately predicting binary labels across large data sets. The four classification problems are all binary classification problems, utilizing data sets from the UCI Machine Learning Repository [3]. The results of in this paper are intended to closely follow "An Empirical Comparison of Supervised Learning Algorithms" [1]. As such, the methodology used to implement each classifier mirrors the techniques used in Rich Cuarana's paper.

1.1 Support Vector Machines

Support Vector Machines convert each data entry into a vector, and use a calculated hyperplane to classify each vector within the label space. The hyperplane is calculated such that the margin is maximized; that is the distance from the hyperplane to the nearest point in the training set at any point is greatest [2]. The implementation used in this paper utilizes a linear kernel, and is available through scikit-learn's public library [4].

1.2 K Nearest Neighbors

K Nearest Neighbors is an algorithm that classifies data based on the nearest data points when the data is represented in multidimensional space. For a given k, a data point is classified based on the k nearest points in the training set which have known labels. The algorithm finds the mode of the labels of the k nearest points in the space, and classifies the data based on this result [2]. The implementation used in this paper is available through scikit-learn's public library [4].

1.3 Artificial Neural Networks

Artificial Neural Networks are a method of learning that can be applied to binary classification. The network is composed of several layers, each consisting of nodes. The input layer receives the data, and passes it to a number of hidden layers, which, based on Bayesian calculations and internal weights, produces a result which is passed to the output layer. This outputs the classification result. Each node learns with each pass of the training data, and becomes better over time. The implementation used in this paper is available through Google's tensorflow public library [5].

1.4 Data sets

Three data sets are used in this paper. All are from the UCI Machine Learning Repository [3].

Adult is a data set with 48,842 entries that shows whether yearly income exceeds $50,000 based on several factors, including age, sex, race, work status, marital status, and more. In total, there are 14 attributes. The data set contains missing values.

Cover Type is a data set with 581,012 entries that predicts the type of forest cover based on cartographic data of the region. In total, there are 15 attributes. The data set does not contain missing values.

Letter is a data set with 20,000 entries that predicts what letter is shown in a picture. Attributes of the picture such as width of box are used as input. In total, there are 16 attributes. The data set does not contain missing values.

2 Method

Each dataset is processed to be in standard numpy format, with the classification labels in the last column. Categorical values are converted to binary columns based on one hot encoding. Each classifier is first tested with varying hyperparameters, and the best are selected based on validation accuracy. Following the aforementioned paper, the data sets are randomly shuffled, then broken into training sets of 5,000 rows with the rest composing the test set [1]. Methods for displaying heatmaps are borrowed from COGS 118A [2]. All code is written in Jupyter Notebooks. There is a separate notebook to parse and clean each data set, and separate notebook for each classifier.

2.1 Classifiers

Using scikit-learn's LinearSVC, the **Support Vector Machine** is first trained and validated using six possible C values [4]. C* is decided based on validation accuracy. A new SVM is then created with C*, and classifies the test data.

For **K Nearest Neighbors**, methods for calculating cross validation and grid search are borrowed from COGS 118A [2]. Caruana uses k values from 1 to |training set| with an interval of |training set| / 26 [1]. Validation accuracy is calculated for each k value, and the best is selected to classify the test data.

The **Artificial Neural Network** implementation is tensorflow's DNNClassifier (DNN stands for Deep Neural Net). The number of hidden units is varied according to the paper, as is the momentum used to optimize [1]. Training and testing is run for 1,000 steps in batches of 100, as is standard in the tensorflow documentation [5]. The best momentum and hidden unit # is then used to classify the test data.

2.2 Data sets

Adult was the hardest to convert to a binary classification problem. Although the label space is

already binary, eight of the attributes are text strings. Each of these columns are cast as categories in pandas, and then converted to one hot encoding utilizing pandas' get_dummies() function. The results are then stacked in numpy into an array that can be saved and loaded by the classifiers.

As done in Caruana's paper, the most common label for **Cover Type** is used as the positive label [1]. The rest are classified as negative. The data is then converted from csv format into a numpy array that can be saved and loaded by the classifiers.

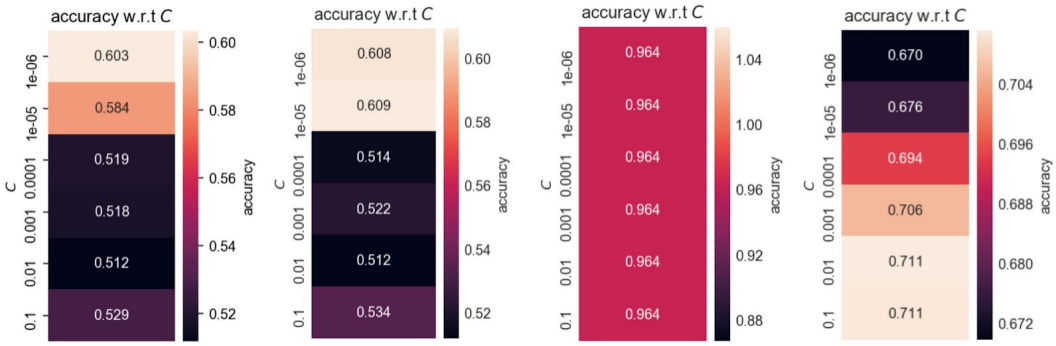
Letter is converted into two data sets representing different binary classification problems. In one case, only the letter 'O' is considered a positive label, and the rest are considered negative. In the second case, half of the letters are considered positive and the other half negative. This is done by defining converter functions that numpy utilizes when loading in the csv data. The datasets are referred to as Letter P1 and Letter P2 respectively. After being loaded and converted, the data is saved so that the classifiers can use it.

3 Experiment

The main variable in question is the test accuracy. That is, how accurate is the classifier at predicting the correct label for each entry in the test set. Before calculating test accuracy, each model is tuned based on specifications defined in Caruana's paper [1].

3.1 Support Vector Machines

The hyperparameter tuned for the SVM is C. The values tested are [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]. Below are the validation accuracies calculated for each C for problems Adult, Cover Type, Letter P1, and Letter P2 respectively.

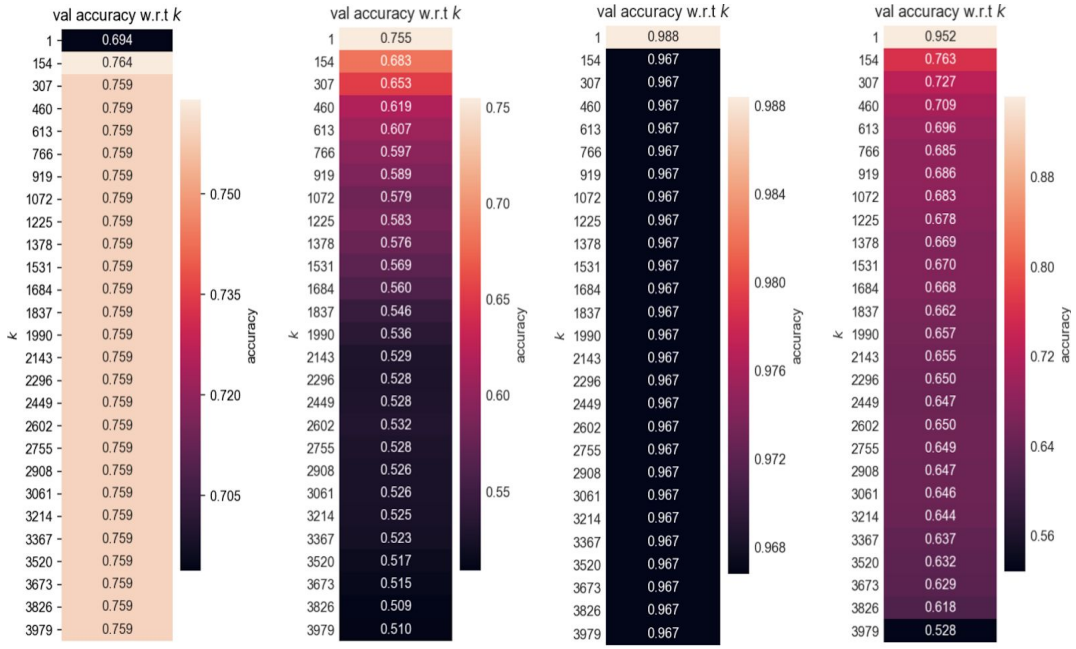


The chosen C values and final test accuracies are shown below.

Dataset	**C***	**Test Accuracy**
Adult	10^{-6}	0.61327
Cover Type	10^{-5}	0.61139
Letter P1	No best choice	0.96193
Letter P2	10^{-1}	0.79192

3.2 K Nearest Neighbors

The parameter that is optimized for the KNN classifier is K. 26 different values were used, evenly spaced across the total size of the data set. For each K, the validation accuracy is calculated and reported. The results for problems Adult, Cover Type, Letter P1, and Letter P2 respectively are shown below.

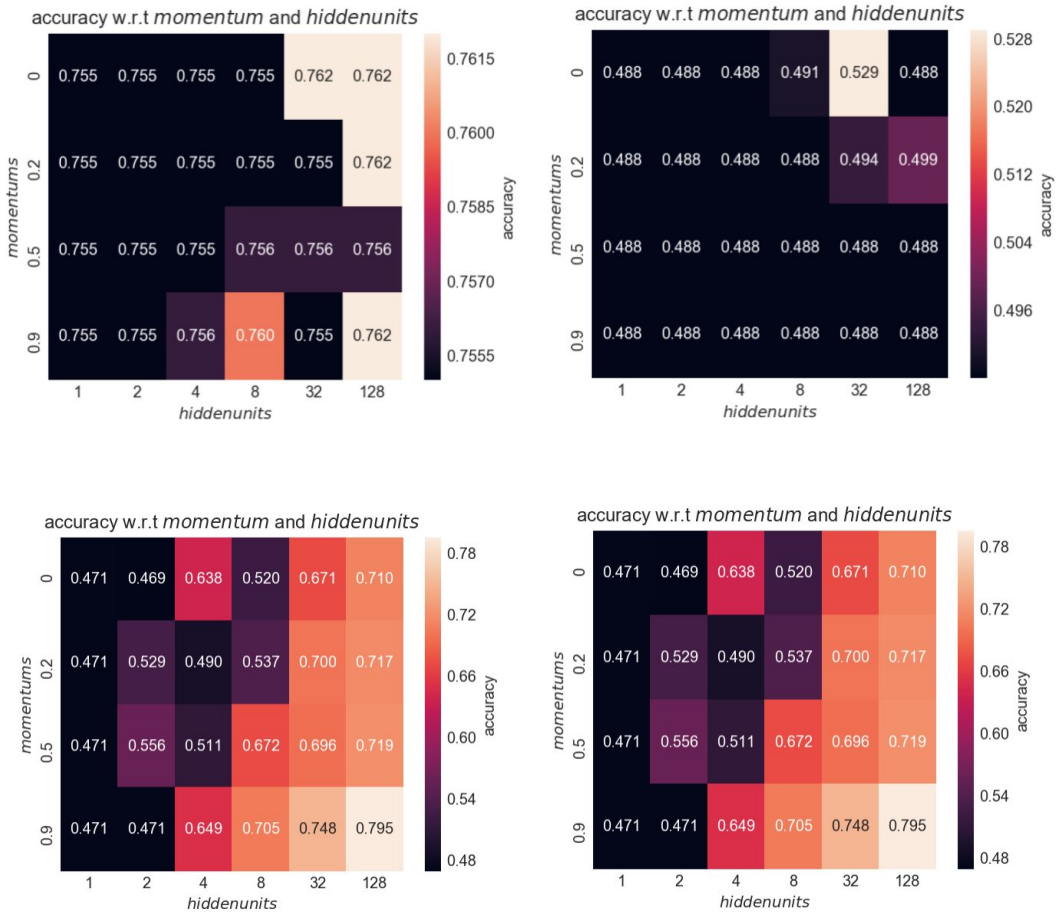


The chosen K values and final test accuracies are shown below.

Dataset	**K**	**Test Accuracy**
Adult	154	0.76220
Cover Type	1	0.78388
Letter P1	1	0.99080
Letter P2	1	0.95640

3.3 Artificial Neural Networks

Two hyperparameters are tuned for the Artificial Neural Networks. The first is the number of hidden units. The values tested are [1, 2, 4, 8, 32, 128]. Momentum is also used to optimize the network. The values tested are [0, 0.2, 0.5, 0.9]. A Neural Network is created with each combination of hidden units and momentum, and after training the validation accuracy is calculated. The results are displayed below. Clockwise, from the top left: Adult, Cover Type, Letter P1, and Letter P2.



The chosen momentum, hidden unit number, and resulting test accuracies are shown below.

Dataset	**Momentum**	**Hidden Units**	**Test Accuracy**
Adult	0	128	0.76341
Cover Type	0	32	0.48563
Letter P1	0.9	128	0.79056
Letter P2	0.9	128	0.80684

4 Conclusion

Below is a comparison of test accuracies across all problems and classifiers.

Classifier	Adult	Cover Type	Letter P1	Letter P2	Average
SVM	0.61327	0.61139	0.96193	0.79192	0.74463
KNN	0.76220	0.78388	0.99080	0.95640	0.87332
ANN	0.76341	0.48563	0.79056	0.80684	0.71161

As we can see, K Nearest Neighbors outperforms Support Vector Machines and Artificial Neural Networks by a significant margin. In fact, K Nearest Neighbors outperforms the others at every problem. This is consistent with the Caruana paper with the exception of Artificial Neural Networks, which outperforms K Nearest Neighbors [1]. Because of the limited access to resources and training time, it is understandable that Artificial Neural Networks perform worse in this case. With more serious training, Artificial Neural Networks will continue to improve and report better test accuracies. It is important to note that every classifier on average achieved a positive classification accuracy. The only exception for problem by problem classification was the Artificial Neural Network, most likely for the same reason as the disparity from Caruana's paper. It is astounding how (relatively) simple it is to implement and run these machine learning models. The number of applications are virtually limitless, and machine learning will likely continue to expand into a brand new industry in and of itself.

References

[1] Rich Caruana & Alexandru Niculescu-Mizil. (2006) An Empirical Comparison of Supervised Learning Algorithms. Ithaca, NY: Cornell University.

[2] Zhuowen Tu (2018) COGS 118A: Machine Learning. La Jolla, CA: University of California, San Diego.

[3] UCI Machine Learning Data Repository. *archive.ics.uci.edu/ml.* Irvine, CA. University of California, Irvine.

[4] scikit-learn: Machine Learning in Python. *scikit-learn.org.*

[5] tensorflow:An open-source machine learning framework for everyone. *tensorflow.org.*