Supervised Learning

Elijah Rockers
CS-7641

Abstract

This submission will explore several different machine learning classification models in Python, using the **scikit-learn** (sklearn)[1] package. The goal is to explore how different hyper parameters affect scoring in terms of accuracy and the ROC AUC metric.

The classification problem is explored using two datasets, **Beans**[2] and **Census**[3]. The Beans dataset is a multiclass problem involving continuous attributes, while the Census dataset is a binary classification problem involving both continuous and categorical attributes. The goal is to explore how these two different datasets can be classified by the five different models, K-Nearest_Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, Boosting, and Neural Network (NN). These two datasets are intermediary in size (Beans N = 13,611, Census N = 48,842).

Beans

The Beans dataset is a multivariate classification problem consisting of 13,611 samples of images of 7 different registered dry beans taken with a high-resolution camera. Each image yields 16 features, 12 of which are dimension based (perimeter, area, etc), and 4 of which are shape factors. The goal for each bean is to be classified into 1 of 7 named classes.

Census

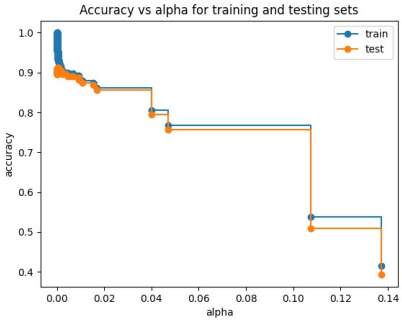
The Census dataset is a multivariate binary classification problem consisting of 48,842 samples of tabular data describing features from the 1994 Census database. Each sample contains a mix of 14 categorical and real features. The features include categorical attributes such as education, occupation, race, and sex, and continuous attributes such as age, hours per week, and capital gain. The goal for each sample is to determine if the person described makes more than $50,000 per year.

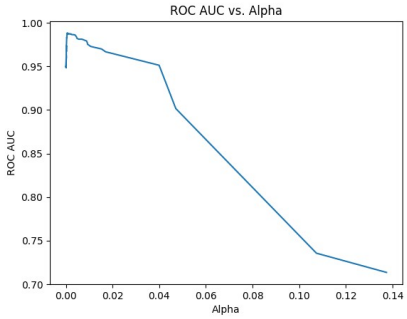
Models

Decision Trees

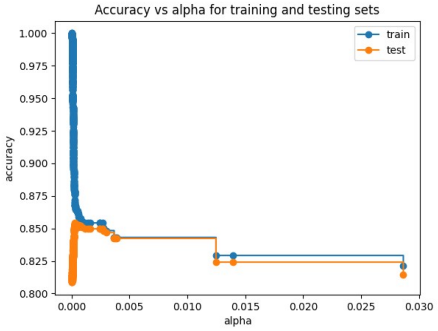
The decision tree model uses a tree-like model of decisions to classify samples based on attributes at a given node. Trees were defined using different parameters for cost-complexity pruning[4], and trees were compared for accuracy against both the training and test set. As expected, with no pruning the modeled decision tree performs with 100% accuracy. When alpha pruning is introduced, the model becomes for generalizable and performance on the training and test set begins to converge indicating a generalizable model. As alpha increases further, however, performance on both data subsets decreases, indicating the decision trees have become “over-pruned” and are no longer sufficient to accurately classify samples. Split criterion was given by the “gini” measure.

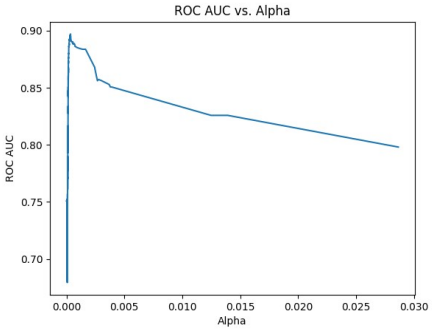
Beans accuracy and ROC AUC graphs against alpha parameter:





Census accuracy and ROC AUC graphs against alpha parameter:





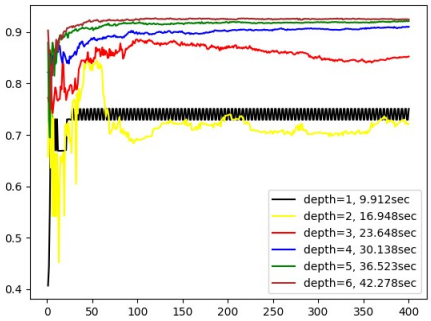
For the Beans dataset, the alpha corresponding to the maximum ROC AUC was 0.0005715, with an accuracy of 0.916.

For the Census dataset, the alpha corresponding to the maximum ROC AUC was 0.0003393, with an accuracy of 0.848.

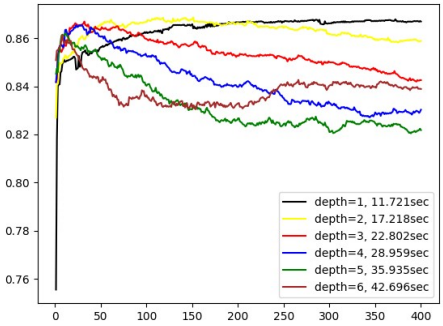
Boosting

For boosting, a decision tree estimator was used. The goal is to use an ensemble of simpler decision trees using the sklearn AdaBoostClassifier[5]. Each iteration attempts to target incorrectly classified data to improve the final model. For this evaluation, decision tree depths from 1 – 6 were tried, across 400 estimators. Interestingly, the two datasets were classified differently according to depth. For the multiclass Beans problem, increased depth appeared to lead to better accuracy. This was not the case for the binary Census classification problem though, with a depth of 1 ultimately yielding the best accuracy on a test dataset.

Beans dataset Boosting accuracy graphed against the number of estiamtors:



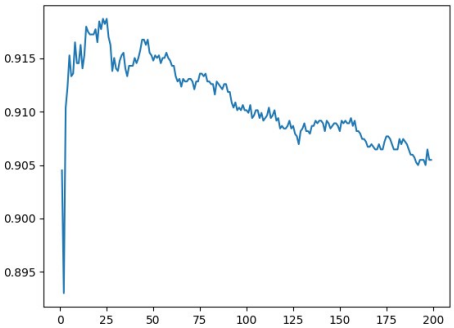
Census dataset Boosting accuracy graphed against the number of estiamtors:



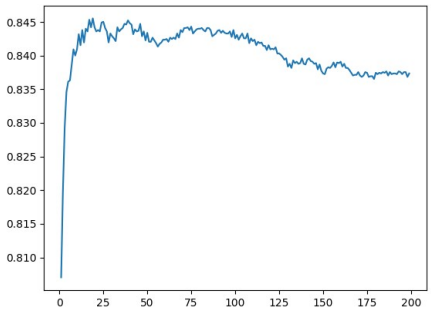
K-Nearest-Neighbors

For KNN, the model is trained on varying values for *K*, where *K* is the number of sample neighbors used to decide on a classification for a given sample. The model is then used to train values from an unseen test dataset. As expected, the optimal *K* peaks and then begins to decrease model performance as more neighbors are taken into account. At the maximum limit, *K* would encompass all samples in the domain, yielding no differentiating information for the model to accurately classify samples.

Beans dataset KNN Accuracy graphed against K:



Census dataset KNN Accuracy graphed against K:

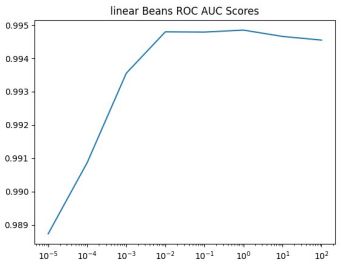


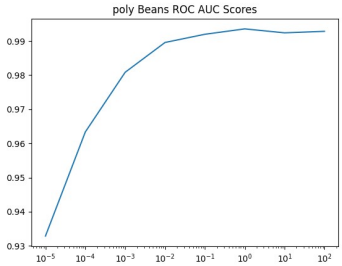
For both the Beans and Census dataset, a *K* of about 22 yielded the highest accuracy.

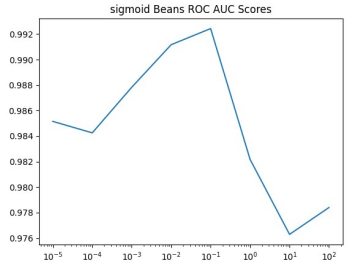
SVM

Support vector machines maps training examples to points in space so as to maximize the width of the gap between categories. Different kernels can yield different results, as well as different values for *C*, the L2 regularization term. Each of these relationships with ROC AUC is explored in the following graphs using “linear”, “poly”, and “sigmoid” kernels, and a regularization term varying from 0.00001 to 100, with *C* graphed on a log scale on the x-axis.

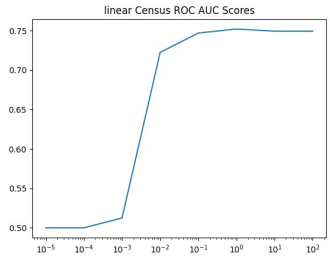
Beans:

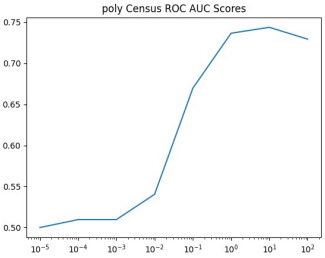


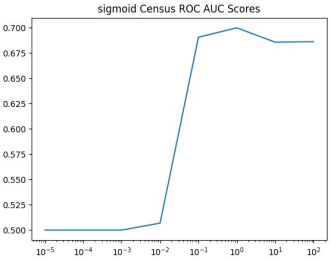




Census:





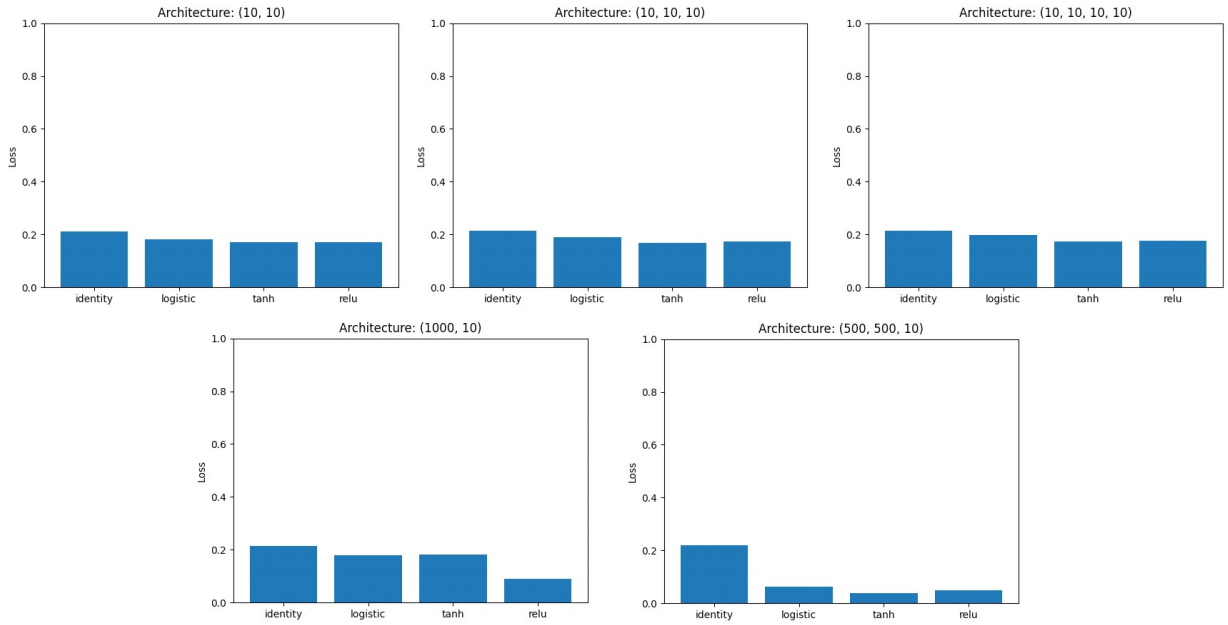


As shown, the optimal C regularization term and kernel varies between the two datasets. For the Beans dataset, a C = 0.01 appears to generalize across all evaluated kernels, while for Census the C = 1 seems to work best with all kernels.

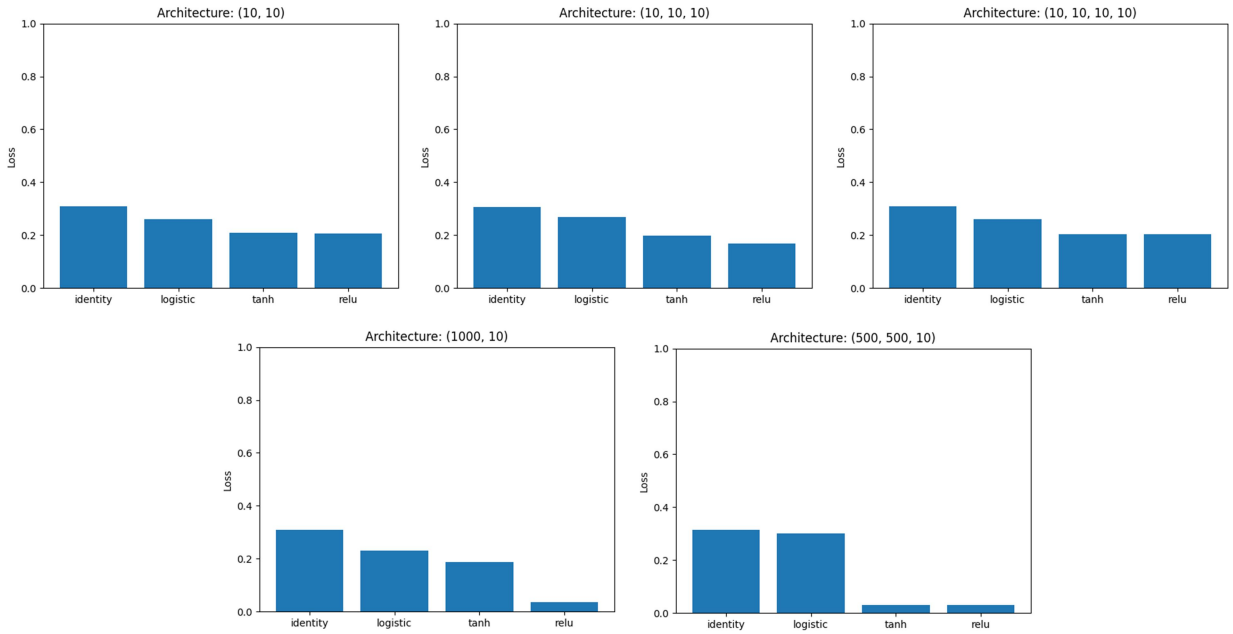
Neural Network

For the neural network model, several different architectures and activation units were explored. The default iteration maximum was set at 500 to decrease the probability that a model would fail to converge during gradient descent. Five different architectures were explored, with hidden layers defined by number of nodes as follows, where the *ith* element refers to the number of nodes in the *ith* hidden layer: (10, 10), (10, 10, 10), (10, 10, 10, 10), (1000, 10), (500, 500, 10), with the loss being described for each below.

For the Beans neural network:



For the Census neural network:



For the Beans dataset, minimum loss of 0.039 was achieved with the (500, 500, 10) hidden layer architecture, and the “tanh” activation unit. For the Census dataset, minimum loss of 0.029 was achieved with the (500, 500, 10) hidden layer architecture and the “tanh” activation unit.

Conclusion

The model preferred depends on the dataset and various hyperparameters, but it seems that the neural network (multi-layer perceptron) with an architecture of (500, 500, 10) and activation function of “tanh” produces the least amount of loss.

It’s difficult to predict which model might perform better for a given data modeling task, and the idea of performance also entails compute time, total number of parameters, in addition to obvious metrics like accuracy and ROC AUC. Tested models might perform differently against varying other types of datasets.

References

[1] https://scikit-learn.org/stable/

[2] https://archive.ics.uci.edu/ml/datasets/dry+bean+dataset

[3] https://archive.ics.uci.edu/ml/datasets/census+income

[4] https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html