Machine Learning vs. US Census Data

Christopher Paterno and Michael White

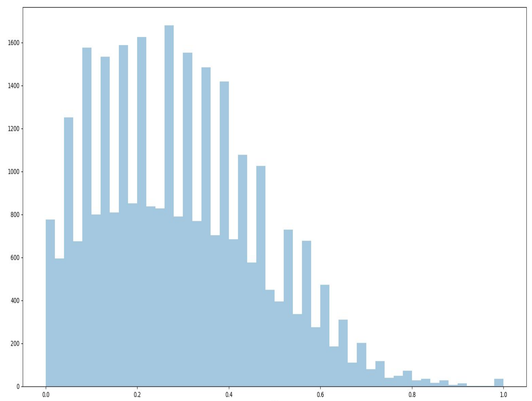
https://github.com/mikeyw1227/csc_461_final_project

Introduction

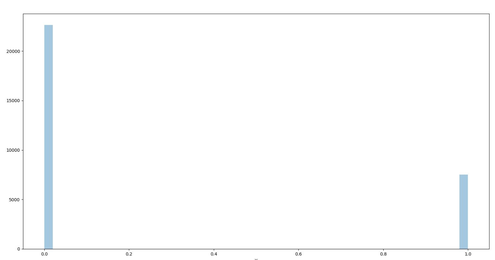
In machine learning there is an abundance of different models that you can attempt to fit to your data. We have attempted to compare ten different types of machine learning models on the UCI Census Income Dataset in order to see which model will perform best for our given problem.

Problem Description

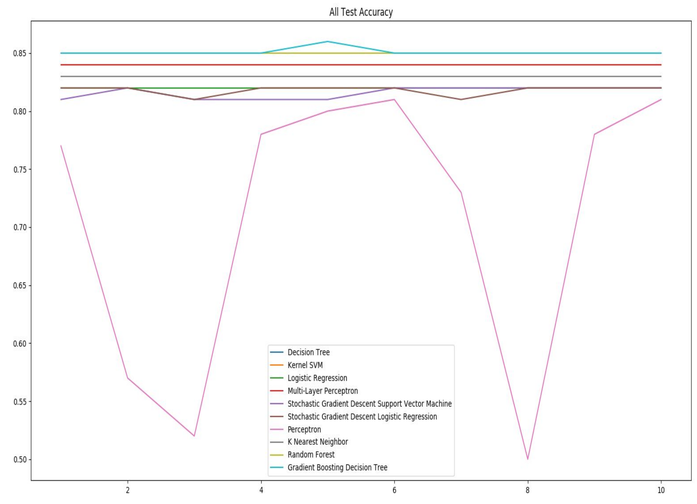
Our problem consists of fitting ten different models to the US Census data set in an attempt to predict if adults will earn greater than or less than/equal to fifty thousand US dollars per year. This is a classical machine learning binary classification problem. The data, taken from the 1994 US census database, consists of 48,842 instances of 14 different features including age, workclass, education, marital status, occupation, relationship, race, sex, capital gain, capital loss, and native country.



Age Distribution



Class Distribution



Data Transformation

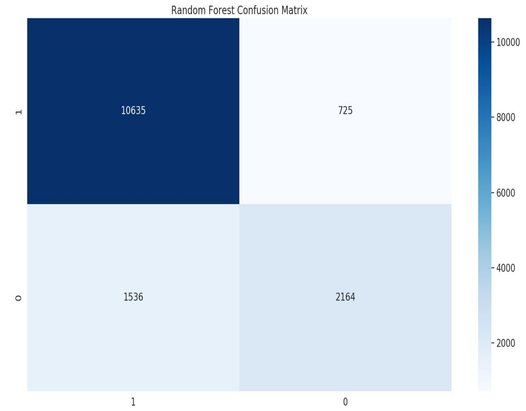
The first thing to note about this data set was that it did have missing values. Due to this our first step was to drop all instances that had missing values. We then had to transform the categorical data into numerical identifiers in order for the data to work with our selected models. Afterwards we had to normalize our data, which we did using sklearn's MinMaxScaler. This transformed our features into a range between zero and one. Our last step was to put twenty-five percent of the training data aside for validation purposes. By doing this we help our model generalize and not overfit the training data.

Approach

To find the optimal model we also needed to find the optimal hyper-parameters for each of the models that we tested. Otherwise our research on which model is best would not have been fully exhaustive. In order to optimize each model we used sklearn's GridSearchCV. This class allowed us to experiment on the various permutations of hyperparameters that we wanted to test, in order to find the best model for our data. Then we compared each of these best models through accuracy reports and confusion matrices.

Models Used

- Decision Tree
- Kernelized SVM
- Logistic Regression
- Multilayer Perceptron
- Linear Perceptron
- Stochastic Gradient Descent SVM
- Stochastic Gradient Descent Logistic Regression
- K-Nearest Neighbors
- Random Forest
- Gradient Boosted Decision Tree



Conclusion

After running the test data on all ten of our selected models we found that the models produced mostly the same accuracy. That being said, Random Forest model had the highest accuracy with 86%. We were satisfied with these results as it was consistent with previous research done on this data set. We found that the best hyper-parameters for the Random Forest was to use entropy for splitting, a max depth of 15 and 100 different trees in our forest. We were slightly surprised that the Multilayer perceptron was not the most accurate as we thought a neural network would have a better chance at classifying the data. However, with that being the case we did notice that the data did seem to set itself up very nicely to work with different types of decision tree models. Our model with the lowest accuracy was the linear perceptron model with its best run producing 81% accuracy.