Using Supervised Learning Algorithms to Solve Several Problems

Sai Kiran Komatineni **Victor Manuel Miranda**

Abstract

In this paper, several supervised learning algorithms are used to solve regression and classification problems. The algorithms varied in complexity while the datasets varied in size. New algorithms are also implemented that were not discussed in class to expand the machine learning skillset. General trends observed in the results are explained; some of which are expected and others are new. The project can be expanded further to explore results obtained by newer classifiers and more number of datasets and that results in a more firm conclusion that is proposed here.

1. Introduction

For the final project, four datasets were explored in order to solve problems related to the dataset using supervised learning algorithms. The datasets were obtained from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/index.php) and Kaggle (https://www.kaggle.com/datasets). The four datasets explored were Exam Scores for Students at a Public School Dataset (Kimmons, 2012), Car Evaluation Dataset (Dua & Karra, 2017), Rain in Australia Dataset (Young, 2018), and Turkey Ankara Ayrancı Anadolu High School's Sign Language Digits Dataset (Dikle & Mavi, 2017). These datasets are later referenced as EXAM, CAR, RAIN, and DIGITS respectively.

The datasets were loaded and processed using the Python programming language (https://www.python.org/) through interactive Jupyter notebooks (https://jupyter.org/), along with the useful and necessary Python libraries: Numpy (http://www.numpy.org/), Pandas (https://pandas.pydata.org/), and Scipy (https://www.scipy.org/). The supervised learning algorithms were implemented using the Python machine learning library Sklearn (https://scikit-learn.org/). Data visualizations were made possible with the use of the Matplotlib (https://matplotlib.org/) Python library.

The supervised learning algorithms used on the EXAM, CAR, and RAIN datasets include Linear Regression (LinReg), Logistic Regression (LogReg), Support Vector Machines (SVM), Decision Trees (DT), Random Forests (RF) and K-Nearest Neighbors (KNN). Additionally, Adaptive Boosting (AdaBoost) was applied to this list of learning algorithms in order to improve their performance. For the DIGITS dataset, the learning algorithms used were the One-vs-One (OVO) and One-vs-All (OVA) classifiers. These algorithms were implemented from scratch and compared to their Scikit Learn implementation.

All of the datasets and Jupyter notebooks used for this project can be found on the Github repository:

https://github.com/vmm2297/COGS-118A-Final-Project

2. Methodology

2.1. General Approach

The objective was to evaluate the performance of each learning algorithm based on dataset. At least six learning algorithms were performed on the first 3 datasets and the testing accuracy was used to evaluate each algorithm's performance. The types of problems in this study include: regression where the data was used to estimate the numerical value that best approximates the true output; binary classification, where the class was predicted based on the given feature vectors; and multi-class classification, where the most probable class was predicted based on the given feature vectors.

Different encoding schemes were performed on the EXAM, CAR, and RAIN datasets feature vectors. The EXAM and CAR datasets were converted into binary classification problems in order to maximize the overall performance of all the classifiers. To include variation among the data while training and testing, differences in partition of the data were such that the training dataset was 80%, 50% and 20% of

the total dataset for each execution. Cross validation was implemented to compute testing accuracies and the best hyper-parameters for each classifier.

A new approach not introduced in COGS 118A was used to predict the output of the DIGITS dataset. For this multi-class classification problem, the least squares solution was solved within the classification algorithm. These algorithms were implemented in Python from scratch and then compared to the Sklearn's implementation of the algorithm.

2.2. Exam Performance (EXAM)

The purpose of using the EXAM dataset is to provide an estimate for a student's exam performance based on the 8 features that describe the student's background. The dataset contains 1000 data points of these 8 features. The first 5 features are categorical and include gender (male, female), race/ethnicity (Group A, B, C, D, E), parental level of education (high school to master's degree), lunch (standard, free/reduced), and test preparation (none, completed). The last 3 features are 3 different test scores (math, reading, writing) for a given student, which are integer values from 0 to 100.

To solve this problem as a regression problem, a label was created as the average of the 3 test scores for a given data point, called "average score". The other 3 test scores were deleted. The categorical features were then needed to be encoded in order to use the LinReg model.

The first encoding approach was to use ordinal encoding, normalized from 0 to 1. The race/ethnicity column was deleted for this approach to avoid having to "rank" race/ethnicity, which is a sensitive topic on its own. The rest of the features were encoded accordingly, with gender having the arbitrary decision of 1 for female and 0 for male. Once encoded, the entire dataset was fit with the LinReg model and received a **21.5% accuracy** on the entire dataset. Splitting the dataset into separate training and testing sets was not done for this regression problem since the accuracy was very low for the entire set to being with.

The second encoding approach was to use one-hot encoding. The race/ethnicity column was re-added since one-hot encoding would not spark controversy for that feature. Once encoded, the entire dataset was fit with the LinReg model and received a **24.2% accuracy** on the entire dataset. This is a higher accuracy than from ordinal encoding, but still very low. Regression variations of SVM, DT, RF, and KNN classifiers were also used to fit the data to obtain **accuracies of ~23-37%** on the entire dataset. DT obtained the highest accuracy with 37.2%. Again, the dataset was not split into seperate training and testing sets due to low accuracies.

Based on the poor accuracies obtain as a regression problem, the dataset was converted into a binary classification problem. The classification labels were split among the mean of the "average score" feature. Thus, a positive label was created for a given data point if "average score" was greater than (or equal to) the mean and a negative label was created if "average score" was less than the mean. The dataset was kept as one-hot encoded because of the slightly higher LinReg accuracy for one-hot encoding. Our general iterative approach of cross validation was then performed for the 3 different partitions types over 3 different random trials for each of the classifiers: LogReg, SVM, DT, RF, and KNN.

2.3. Car Evaluation (CAR)

The purpose of the CAR dataset is to evaluate how reasonable the price of a car is given the following 6 features: Buying Price, Maintenance, Number of Doors, Passenger Capacity, Trunk Size and Safety Considerations. All of the feature vectors were categorical, thus ordinal encoding was used to convert the data into a format such that the data is easier to process. The default labels had four categories: "very good", "good", "acceptable", and "unacceptable." To convert this into a binary classification problem, the first three categories were labeled as positive (integer 1) and the last category as negative (integer 0).

The classifiers used on this dataset were: SVM, KNN, DT, RF, and Adaboost. As an attempt to simplify the readability of the code, all the classifiers were implemented as functions such that each function prints the results at the end. For each partition type of the dataset, three random trials of training and testing were run to obtain the testing accuracies along with the best hyper-parameters for the respective classifier. The average of the nine training attempts was calculated and returned at then end for comparison.

2.4. Rain Prediction (RAIN)

The purpose of the RAIN dataset is to predict whether it will rain the next day based on 19 weather conditions (feature vectors) on a given date and location in Australia. When a classifier was executed on all the given features, the program's run time was too large. To combat this, the number of features was reduced to 13, including: "Min Temp", "Max Temp", "Rainfall", "Wind Gust Speed", "Wind Speed", "Humidity", "Pressure" and "Temperature." This decision was made based on the fact that the reduced features contained categorical information and the kept features contained continuous data.

A lot of the categorical features contained NAN/NULL data, sometimes up to ~40-50%. Additionally, some of the other data points contained NAN/NULL values, which could be case of the lack of data or of accidental loss. To solve this issue, those data points with NAN/NULL values were removed from the dataset and not included in the experiment.

The main benefit of the RAIN dataset is its size. There are approximately 142,000 instances which is by far the largest of our datasets used in the project. Since fitting the classifiers on the entire dataset would take a longer period of time, only a random sample of 10,000 instances were used. The classifiers and methods used to make predictions and report the testing accuracies were the same as explained for the CAR dataset.

2.5. Digit Recognition (DIGITS)

The DIGITS dataset is reminiscent of the MNIST dataset (LeCun, Bottou, Bengio, & Haffner, 1998). Like MNIST, the DIGITS dataset proposes a multi-class classification problem of recognizing digits 0 through 9. The difference is that MNIST is a dataset of handwritten images (28x28 pixel grayscale images) while DIGITS is a dataset of sign-language images (64x64 pixel grayscale images).

Unlike the other datasets, the purpose of using the DIGITS dataset is to implement algorithms not discussed COGS 118A. These algorithms include the OVO and OVA classifiers. The inspiration to use these algorithms for the DIGITS dataset was drawn from ECE 174: Introduction to Linear and Nonlinear Optimization with Applications. In ECE 174, the OVO and OVA algorithms were implemented from scratch in MATLAB and tested with the MNIST dataset. The algorithms were implemented with the objective of minimizing the least squares solution. For this final project, a goal was to implement OVO and OVA in Python from scratch with a similar dataset as MNIST, hence the DIGITS dataset.

The dataset was already given as X and Y '.npy' files which is easily loaded with numpy. The file X.npy was given as 3D numpy array of size Nx64x64 where N is the number of samples and 64x64 is the pixel dimension of the image. The X dimensions were flattened to be a 2D numpy array of size Nx4096. The file Y.npy was given as a 2D numpy array of size Nx10 where N is the number of samples and 10 is a vector for the binary label of the 10 classes (digits 0-9). A 1 is placed where the digit is classified and 0 everywhere else. The Y dimensions were flattened to be of size Nx1 by assigning the label to be a single digit from 0 to 9 for that sample class. For DIGITS, N=2062, so there is not as much data points as in MNIST where N=70,000.

After implementing the OVO and OVA algorithms, they were compared against each other on 80/20 dataset split and were reported by their respective testing accuracies. The algorithms implemented from scratch were also compared against their respective Sklearn implementation.

3. Results

For the EXAM dataset, we see that converting the problem from a regression problem to a classification problem resulted in **much higher accuracies into the ~60-70% ranges**, however, these accuracies are still poor. The best results were gained from fitting the LogReg and SVM models across all partition types.

For the EXAM, CAR, and RAIN datasets, the performance of each classifier was displayed under three different tables in **Appendix A.1**. **Table 1** shows, for each dataset/classifier pair, the best training accuracy along with its respective optimal hyper-parameter. **Table 2** shows, for each dataset/classifier pair, the average testing accuracy for each partition type (80/20, 50/50, 20/80) across the 3 trials. **Table 3** shows, for each dataset/classifier pair, the average test accuracy across all the partition types.

For each of these datasets, heatmaps were plotted for each individual trial. **Appendix A.2**. shows the

resulting heatmaps for the CAR dataset for one trial of the 80/20 partition type. The rest of the heatmaps for each dataset/problem can be found under their respective Jupyter notebook in the Github repo.

Looking at **Table 1**, the general trend that the 80/20 partition type resulted in the best testing accuracies for most classifiers was noticed. Looking at **Table 2**, it was seen that SVM is the strongest classifier for the EXAM dataset, but is weakest for the CAR dataset. It is also seen that KNN is the strongest classifier for the CAR dataset, but is weakest for the EXAM dataset. For the RAIN dataset, all classifiers seem to fall within the ~83-85% accuracy range. Looking at **Table 3**, the DT, RF, and their boosted variants all obtained their optimal hyper-parameter at D=5. The optimal hyper-parameter for KNN varied between datasets, while the optimal hyper-parameter for SVM, and its boosted variant, was C=1.0 for the EXAM and RAIN datasets, while it was C=0.1 for the CAR dataset.

Lasty, for the DIGITS dataset, the OVA and OVO implementations were compared against each other on their testing accuracy on a single trial of a 80/20 partition type. The OVA implementation received a training accuracy of **37.05%** while the OVO implementation received a training accuracy of **76.03%**. Thus, the implementation of OVO was superior to OVA. When comparing these implementations to those of Sklearn, the OVA implementation exactly matches up with Sklearn's OneVsRestClassifier function using Linear Regression as the objective function with a 37.05% accuracy. The OVO implementation is better than Sklearn's OneVsOneClassifier function using Linear Regression as the objective function with only a 37.78% accuracy. However, when using SVM as the objective function for Sklearn, the OneVsOneClassifier function obtains a 78.93% accuracy, which is a little higher than the OVO implementation.

4. Discussion

For the majority of the results obtained, they were within expectations and some new and interesting conclusions could be made. Some of the observations that will be discussed are the effect of having more data, effect of boosting on each classifier, comparison between performing regression on a dataset versus converting it into a binary classification problem, comparison between different types of encoding, comparison of performance for each type of partition and the comparison of the best hyper parameters for each dataset.

The comparison between number of instances was done between the CAR and EXAM datasets. Car had twice as many instances and one less feature as EXAM; as it can be seen in the table, CAR's performance was around 1.6 times better on average among all classifiers, excluding SVM. This is a simple example proving the fact that the more data is available, the better the performance of all classifiers. One anomaly that was noticed however is that when classification was performed on the RAIN dataset, the overall accuracy seems to be worse by around ~10% compared to the CAR dataset. The conclusion that was made based on this is that the data may be experiencing overfitting as there were too many features and data points that could have contradicted each other.

A second observation that was made is that boosted SVM did not perform any much better than unboosted SVM; this is true across all datasets and partitions. The test accuracy of the boosted SVM was within ~1-2% of that of unboosted SVM. This is an expected observation as it has been shown by Wickramaratna, Holden, and Buxton (2001) in their paper "Performance degradation in boosting". The reason for this is that SVM is already a strong classifier and having it reclassify may incorrectly classify some points which does not change the result.

From the results it can be concluded that boosting has the most impact on DT and RF. The most significant improvement can be seen for the CAR dataset with the 80% training data partition: boosting improved the test accuracy of DT by 6.35%. This shows that the weights for each data point can significantly improve the classifier's decision making capability allowing it to reach to the right conclusion. Following up from this for the same dataset and the same partition, it can also be seen that both boosted RF and boosted DT have exactly the same test accuracy. This may be due to the fact that the boosted RF algorithm found the same "tree" which is implemented as the boosted DT.

Another final observation that can be made from the table is that the best hyper parameters are generally the same across all the data. Which means that all the classifiers reach the same conclusion. Where they are

not the same, there is a difference in the testing accuracy obtained.

5. Bonus Points

Bonus points are requested for implementing the new OVO and OVA algorithms not discussed in COGS 118A. These algorithms were also a direct application used from another class, ECE 174, which shows the use of material from outside classes. Bonus points are also requested for using a large number of classifiers, including 6 classifiers and 3 boosted variants.

6. Conclusion

Unexpected results were obtained from the RAIN dataset, so it was disregarded in the conclusions that were made. Attempts were made to “adjust” the data to get the results to match those of the expected, however these attempts were not successful. The adjustments that were made include: removing all columns with categorical values, reshuffling the data and using only a small portion of the data (5000 instances).

The overall performance of a classifier was heavily determined by it hyper parameters, number of features in the dataset, number of instances in the dataset and also on the quality of the data as not all datasets show a clear pattern between the feature vectors and labels.

References

Dikle, Z. & Mavi, A. (2017). Turkey Ankara Ayrancı Anadolu High School's Sign Language Digits Dataset, Version 2. Retrieved December 8, 2018 from https://www.kaggle.com/ardamavi/sign-language-digits-dataset.

Dua, D. & Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Kimmons, R. (2012). Exam Scores for Students at a Public School, Version 1. Retrieved December 8, 2018 from https://www.kaggle.com/spscientist/students-performance-in-exams.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner., P. (1998). "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324.

Wickramaratna, J., Holden, S., & Buxton, B. (2001) Performance degradation in boosting. In J. Kittler & F. Roli (Eds.), *Proceedings of the 2nd International Workshop on Multiple Classifier Systems* (pp. 11-21). London, UK: Springer.

Young, J. (2018). Rain in Australia, Version 2. Retrieved December 8, 2018 from https://www.kaggle.com/jsphyg/weather-dataset-rattle-package.

A. Appendix

A.1. Table Results

Table 1: Best Training Accuracy with Optimal Hyper-parameter

		Classifier						
		SVM	KNN	DT	RF	Boosted SVM	Boosted DT	Boosted RF
Dataset	EXAM	68.20% C = 1.0	72.20% K = 3	68.90% D = 5	69.60% D = 5			
	CAR	73.32% C = 0.1	97.58% K = 5	93.70% D = 5	93.47% D = 5	72.61% C = 0.1	93.70% D = 5	93.32% D = 5
	RAIN	85.73% C = 1.0	86.78% K = 6	86.74% D = 5	87.58% D = 5	85.73% C = 1.0	86.74% D = 5	87.58% D = 5

Table 2: Average Test Accuracy for Each Partition (80/20, 50/50, 20/80)

		Classifier						
		SVM(%)	KNN(%)	DT(%)	RF(%)	Boosted SVM(%)	Boosted DT(%)	Boosted RF(%)
Dataset	EXAM	66.66 61.87 63.29	56.83 58.6 59.29	61.50 59.2 59.96	64.33 61.33 62.04			
	CAR	73.32 73.50 71.88	97.58 96.00 95.00	93.70 94.36 95.36	93.47 93.81 95.58	72.60 73.48 68.12	93.70 94.38 95.36	93.32 93.55 95.15
	RAIN	84.75 85.28 84.46	84.50 84.00 83.35	83.80 82.78 82.28	83.45 83.42 82.96	84.80 85.48 84.43	83.85 84.42 82.96	83.45 83.42 82.96

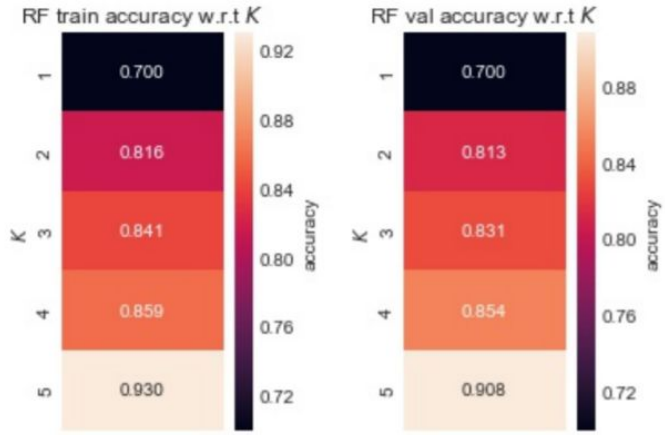
Table 3: Average Test Accuracy Across All Partitions

		Classifier						
		SVM(%)	KNN(%)	DT(%)	RF(%)	Boosted SVM(%)	Boosted DT(%)	Boosted RF(%)
Dataset	EXAM	63.94	58.24	60.22	62.57			
	CAR	72.9	96.19	94.47	94.29	71.4	94.48	94.01
	RAIN	84.83	83.95	82.95	83.28	84.9	83.74	83.28

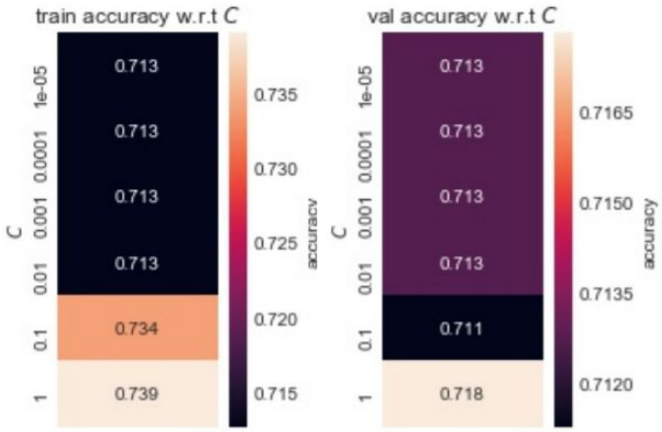
A.2. Sample Heatmaps*

CAR Heat Maps: heat maps for each classifier for the 80/20 partition

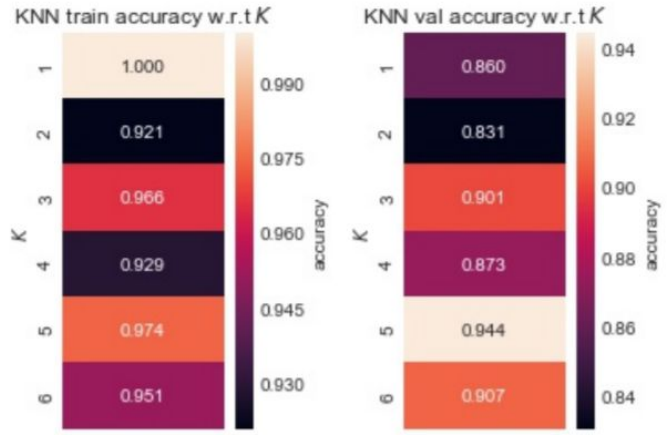
Random Forest

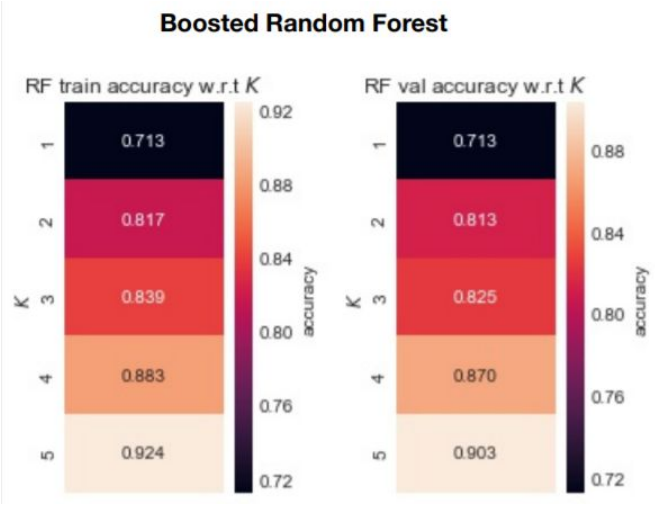


Boosted SVM



Boosted Decision Tree





***The rest of the heatmaps for each problem/dataset can be found in their respective Jupyter notebook in the Github repository.**