COGS 118A Final Project

Stanley Hahm sthahm@ucsd.edu
Cognitive Science, Machine Learning
University of California, San Diego
San Diego, CA 92092-0100, USA

Editor:

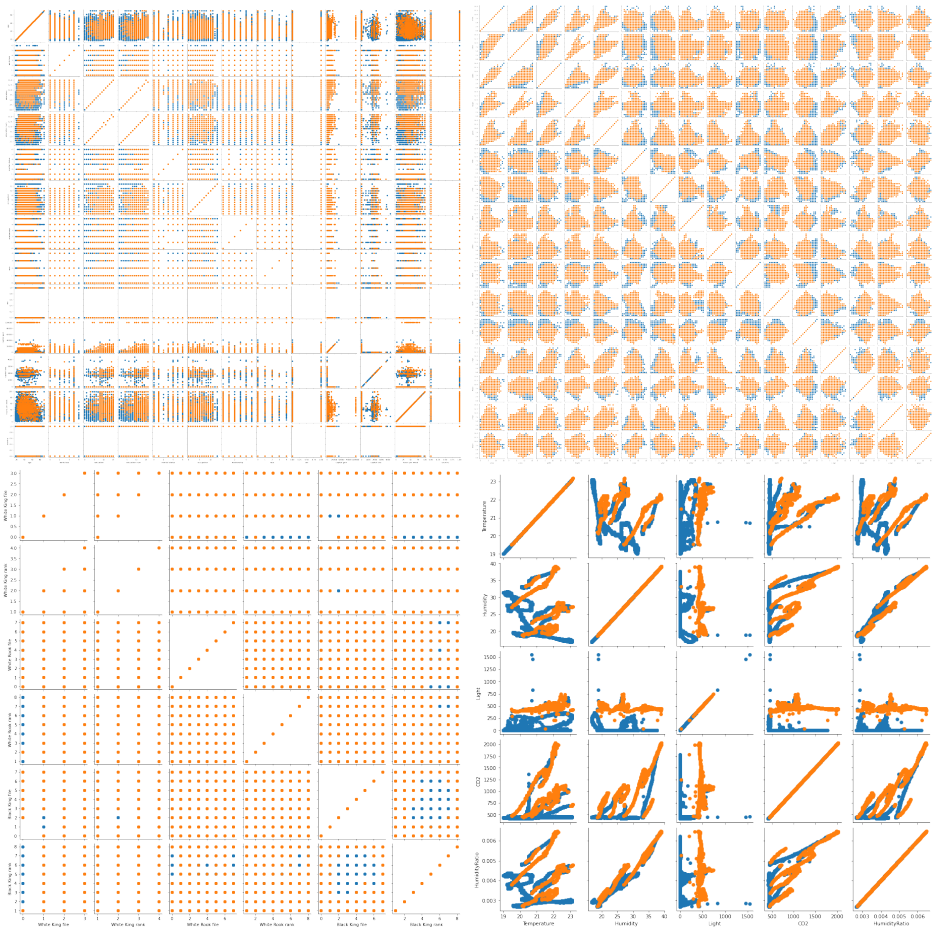
Abstract

This paper is attempting to replicate the CNM06 paper. The paper includes three different algorithms (with different hyper-parameters than that of the paper), three different scoring metrics, and four datasets. The training algorithsm are SVM, Logistic Regression, and Decision Trees and the scoring metrics are accuracy, AUC of ROC, and F1 score.

Keywords: Binary Classification, One-Hot Encoding, Decision Trees, CNM06 Paper, SVMs, Logistic Regression

1. Introduction

Caruana and Niculescu-Mizil in 2006 published a paper referred to as CNM06 that utilized various algorithms like SVMs, neural nets, logistic regression, and others to compare and measure their efficiency. This was significant as it also included different metrics like accuracy, F-score, LIFT, ROC Area, and more to examine the various capabilities and accuracy of these algorithms. In addition, they did this across 11 problem datasets.
For this project, I will be making a small-scale replica of this paper. First, I will be only using three algorithms; Decision Trees, Logistic Regression, and SVMs. Second, I will only restrict this project to three metrics; accuracy, ROC-AUC, and F1. And finally, I have only used four datasets instead of the eleven used in CNM06. This is due to simply time constraint for this project and the lack of expansive knowledge for machine learning algorithms.
For support vector machines, the C parameter checks for any outliers or potentially misclassified data and makes the SVM optimization avoid any misclassification, based on how big the C value is. For my SVMs, I avoided larger C values ($C < 10^2$) due to the binary classified data not exactly being separable. So if I had a high C value, it would overfit and have a really hard time avoiding any misclassification. Below are the graphs demonstrating how each of my four datasets have binary data that is not very well separated.



As for calibration, I omitted them from this project and instead simply used the algorithms mentioned prior. This can lead to some results not being compared fairly across algorithms, but I considered those into the costs for this project and still came out content with leaving the calibration out.

As for the results, there was a mixture of amazement and satisfaction as I received similar results to the CNM06 paper, with some exceptions. Both Decision Trees and Logistic Regression actually performed much better than expected. They garnered higher accuracy and metric scores than what they received in the CNM06 paper. Granted, we didn't have any calibration in this project so this may have been to the cause of a better performance from those two algorithms. And although both of our SVMs were of similar success, my SVM had a surprisingly higher accuracy rate.

However, what was consistent with the paper is that SVM performed better than both Decision Trees and Logistic Regression and was the best algorithm for learning these datasets. In fact, SVM outperformed every algorithm except for one dataset (OCCUPANCY) in which Decision Trees had a higher performance by a slim margin. In addition, an interesting finding was that across the metrics, both Decision Trees and Logistic Regression had very similar performance to one another with their values being within just a few hundredths of a difference from each other. This is also consistent with CNM06 as the non-calibrated Decision Trees and Logistic Regressions had values also within a few hundredths of a decimal

of each other.

2. Methods

2.1 Learning Algorithms

I used various parameters and variations across each learning algorithm and ran a 5 cross validation for each of them. This section will go over the specific parameters of each algorithm used:

SVMs: the kernels used in SVC were linear and rbf. For the gamma, it was ran by factors of ten from 10^-6 to 10^-2. And for our C value, I tried to mimic the CNM06 parameters but like mentioned before, I chose to not go past 10^2, so the C values started from 10^-7 (like in the CNM06 paper) and ended it earlier at 10^1.

Logistic Regression (LOGREG): The solvers that I used for LogisticRegression were saga and lbfgs. For the penalties, I had L1 and L2 penalties for saga but only L2 for lbfgs. And finally, for the C values, it was kept from 10^-4 to 10^4. These were the parameters set in place by Professor Jason Fleischer in his Lecture 19 Notebook.

Decision Trees (DT): I used gini and entropy for my criterions and only dealt with altering the max depth and min samples split. For max depth, I kept the depth ranging between 1 to 4. And for min samples split, I had two variations going from 2 to 5, but had one variation just having 2 to 3 for simplicity.

2.2 Performance Metrics

The performance metrics for this project were Accuracy, ROC, and F1.
Accuracy, since there is a lack of calibration, may not be totally accurate. There is a tendency, especially amongst Decision Trees and Logistic Regression to overfit the classification especially if there is a class imbalance and based on Table 1, there are clear imbalances between the positive and negative class for every dataset. ROC helps account for this by helping decide what classifies as positive cases before the negative cases. F1 score uses recall and precision to also demonstrate another way to calculate accuracy in a learning algorithm.

2.3 Datasets

We utilize four datasets, all of which have above 5000 instances and range from 5 attributes to as much as 16. The more specific information is available in Table 1 in the Appendix.
For the ADULT dataset, I dropped one column 'fnlwgt' as I was not familiar with what that represented and the website that provided this dataset didn't provide an explanation for that variable so I took it out. And I hot one encoding all the non-continuous vairables. And I split the binary classification based on if the person made equal to or less than $50k (mapped as 0) or if they made more than $50k (mapped as 1).
For the LETTER dataset, I simply split the alphabet in half and anything from A-M was mapped to 0 and anything past that was mapped to 1.
For the CHESS dataset, I also had to hot one encode all of the letter grid chess board

names to numbers. In addition, I mapped any Optimal Depth of Win that between zero and eight as 1 and anything bigger than that from nine to sixteen including the draw option was mapped as 0. This was my binary variable.
For the occupancy dataset, it was already categorized for me and the binary variable, occupancy, was already mapped as 0's or 1's.

3. Experiment and Results

There was 5000 randomly selected data samples for each trial of a data set and algorithm combination. This was done within a 5-fold cross validation. After scaling the respective algorithm, I utilized a grid search to run this experiment on each dataset for each hyperparameter. After that, I was able to fit the model to dataset and find the most optimal parameters for each dataset for each metric.
After receiving the most optimal parameters, I fit them back into both the training and testing variables and made a classification report on how well each algorithm did.
Then more specifically, after receiving all that data, I ran accuracy_score, roc_auc_score, and f1_score for each test and training set and found the relevant accuracy, roc auc, and f1 scores for each dataset. This data was able to be averaged and displayed on both Table 2 and 3.

4. Performance

In Table 2, each column with the exception of the last one (mean) represents the average metrics over each of the four datasets. The row represents the algorithm used in the calculation. This makes 9 scoring performances and each of these are the averaged scores. And mean is simply the average of all the values in that particular algorithm.
When evaluating which algorithm had the highest performance per metric, I boldfaced that value so you can see that SVM performed the best in all of them. And if you look closer, you'll notice as there is an asterisk on Decision Tree's performance on ROC AUC and F1 MICRO as it had a p-value > 0.5 when compared to SVM's performance over those metrics.
In Table 3, the columns (except for the last one, mean) are the datasets used and the rows again list the algorithms used over the three metrics. The values listed are the average of each datasets' accuracy, roc auc, and F1 score of each algorithm. The mean column in the end signifies the average of all the values in that particular algorithm.
And again, I boldfaced the best performing algorithms in each dataset, and although SVMs performed the best in most daetaset, Decision Trees managed to perform the best in one of them.

5. Conclusion

Overall, the best performing algorithm was the SVM, with having the highest performance on all four metrics across every single dataset. In addition, it had the highest performance across each dataset except for OCCUPANCY in which Decision Tree had the highest per-

formance. In fact, Decision Trees had an almost significant performance as SVM when it came to metrics ROC AUC and F1 MICRO because its p value was over 0.05 with the performance of the SVMs under those metrics.
However, despite those exceptions, SVM far outperformed the other algorithms especially when it came to the three datasets ADULT, LETTER, and CHESS. The lowest performance was that of Logistic Regression which never seemed to have a significant p-value to the point where it can compare to SVMs' performance. However, it did not perform completely horribly as it usually garnered similar results to Decision Trees.
When I look back onto the CNM06 paper, I was able to definitely get similar responses. Decision Trees and Logistic Regression seemed to perform similar to each other and definitely performed at a decent level, just like in the CNM06 paper. But by no means was it remotely close to SVMs' performance which is also a characteristic shown in the CNM06 paper.

6. Bonus

As for the differences between each algorithms' training and test set performance, it is really remarkable to see the values are not as close as I expected. For instance, the SVM accuracy of the training set is 0.8661 which has a significant difference to our test set performance of 0.9311. Another example is if we take Decision Tree's F1 score across the training set, then it is 0.9886, a huge increase in performance compared to the test set performance of 0.7584. As for Logistic Regression's ROC AUC score in the training set, it is 0.9763, another drastic increase than the test set value of 0.8336.
Why would this be the case? I think for the SVMs, since we avoided higher C values, we managed to reach a lot of misclassification which led to our training set to have a lesser performance accuracy. And as for our Logistic Regression and Decision Trees it is the other way in which we allowed the algorithms, with the parameters it had, to overfit the data and gather a really high performance rate while it trained.

Acknowledgments

I would like to acknowledge Professor Jason Fleischer and the entire COGS 118A staff. They were all extremely helpful not only towards this paper, but helping me understand the material of this course that was definitely relevant to this project. In fact, Professor Fleischer's Week 19 notebook helped me extremely which is why I included it in my references. In addition, I really am appreciate of all the classmates in Piazza but especially Discord as there was definitely a family formed there all online.

Appendix A.

	#ATTR	TRAIN SIZE	TEST SIZE	%POS
ADULT	13/104	5000	25161	24.9%
LETTER	16	5000	14999	50.3%
CHESS	6	5000	23055	13.6%
OCCUPANCY	5	5000	3143	21.2%

Table 1: Description of Problems

	ACC	ROC AUC	F1 MICRO	MEAN
DT	0.8669	0.8439∗	0.7584∗	0.8231
LOGREG	0.8629	0.8336	0.7257	0.8074
SVM	**0.9311**	**0.9149**	**0.8354**	0.8938

Table 2: Performance of algorithms over metrics

	ADULT	LETTER	CHESS	OCCUPANCY	MEAN
DT	0.7471	0.7280	0.8180	**0.9892**	0.8206
LOGREG	0.7357	0.7229	0.7946	0.9773	0.8076
SVM	**0.7686**	**0.9405**	**0.8894**	0.9766	0.8938

Table 3: Performance of algorithms over problems

PROB/ALGO	ACC	ROC AUC	F1 MICRO
ADULT/DT	0.8396	0.806613	0.582726
LETTER/DT	0.7410	0.723571	0.715572
CHESS/DT	0.9156	0.875365	0.666667
OCCUPANCY/DT	0.9952	0.984676	0.988462
ADULT/LOGREG	0.8196	0.786888	0.481013
LETTER/LOGREG	0.7080	0.734560	0.694049
CHESS/LOGREG	0.9064	0.847433	0.464531
OCCUPANCY/LOGREG	0.9886	0.973237	0.974079
ADULT/SVM	0.8558	0.833054	0.644302
LETTER/SVM	0.9238	0.923908	0.923355
CHESS/SVM	0.9496	0.923144	0.808801
OCCUPANCY/SVM	0.9896	0.977734	0.976619

Table 4: Appendix 1 (Training Set Data)

The difference between the algorithms' training and test set performance is discussed in the bonus section.

	AD 1	AD 2	AD 3	AD 4	AD 5	LET 1	LET 2	LET 3	LET 4	LET 5	CH 1	CH 2	CH 3	CH 4	CH5	OCC 1	OCC 2	OCC 3	OCC 4	OCC 5
SVM ACC	0.866241	0.866750	0.864870	0.866852	0.865806	0.866241	0.866750	0.864870	0.866852	0.865806	0.866241	0.866750	0.864870	0.866852	0.865806	0.866241	0.866750	0.864870	0.866852	0.865806
SVM ROC AUC	0.994381	0.990451	0.991452	0.995215	0.991923	0.994381	0.990451	0.991452	0.995215	0.991923	0.994381	0.990451	0.991452	0.995215	0.991923	0.994381	0.990451	0.991452	0.995215	0.991923
SVM F1	0.866241	0.866750	0.864870	0.866852	0.865806	0.866241	0.866750	0.864870	0.866852	0.865806	0.866241	0.866750	0.864870	0.866852	0.865806	0.866241	0.866750	0.864870	0.866852	0.865806
LOGREG ACC	0.958483	0.952828	0.952241	0.956448	0.955138	0.958483	0.952828	0.952241	0.956448	0.955138	0.958483	0.952828	0.952241	0.956448	0.955138	0.958483	0.952828	0.952241	0.956448	0.955138
LOGREG ROC AUC	0.977764	0.978407	0.973820	0.976461	0.974833	0.977764	0.978407	0.973820	0.976461	0.974833	0.977764	0.978407	0.973820	0.976461	0.974833	0.977764	0.978407	0.973820	0.976461	0.974833
LOGREG F1	0.958483	0.952828	0.952241	0.956448	0.955138	0.958483	0.952828	0.952241	0.956448	0.955138	0.958483	0.952828	0.952241	0.956448	0.955138	0.958483	0.952828	0.952241	0.956448	0.955138
DT ACC	0.989444	0.987778	0.989111	0.986806	0.989694	0.989444	0.987778	0.989111	0.986806	0.989694	0.989444	0.987778	0.989111	0.986806	0.989694	0.989444	0.987778	0.989111	0.986806	0.989694
DT ROC AUC	0.993807	0.993435	0.992511	0.990452	0.994675	0.993807	0.993435	0.992511	0.990452	0.994675	0.993807	0.993435	0.992511	0.990452	0.994675	0.993807	0.993435	0.992511	0.990452	0.994675
DT F1	0.989444	0.987778	0.989111	0.986806	0.989694	0.989444	0.987778	0.989111	0.986806	0.989694	0.989444	0.987778	0.989111	0.986806	0.989694	0.989444	0.987778	0.989111	0.986806	0.989694

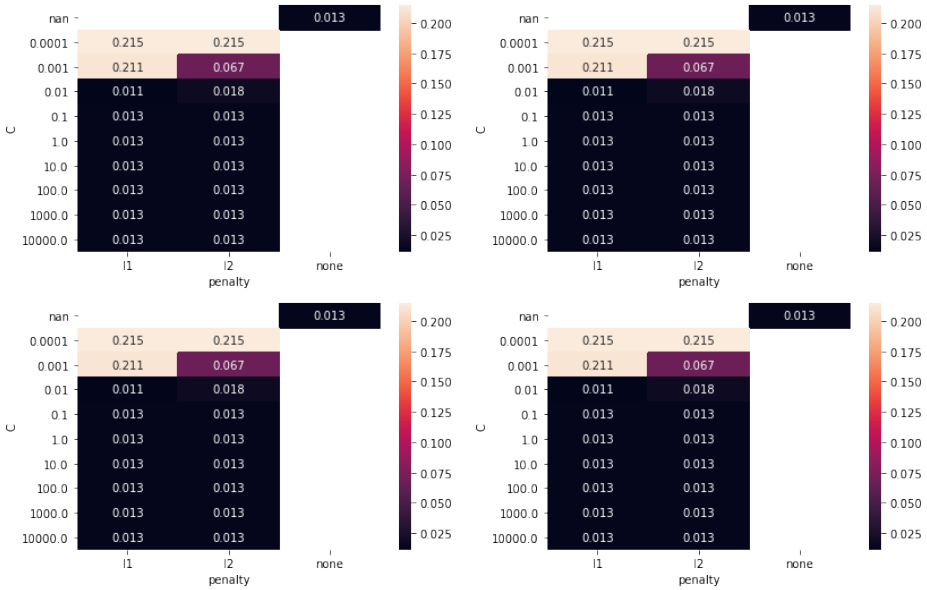
Table 5: Appendix 2 (Raw Test Set Scores)

'AD' = ADULT, 'LET' = LETTER, 'CH' = CHESS, 'OC' = OCCUPANCY

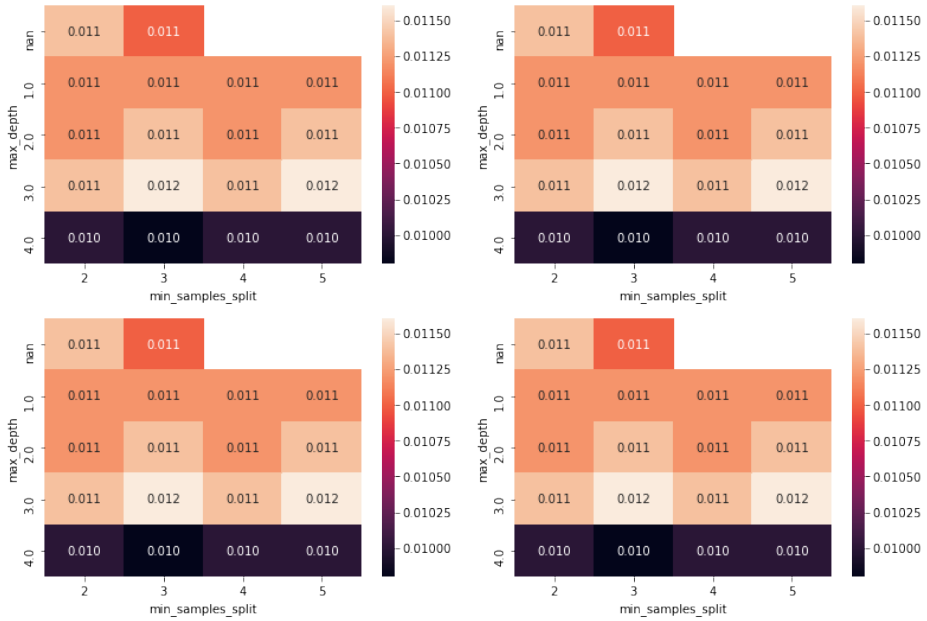
Appendix A. Heatmaps

Below are heatmaps I have plotted to check the validation performance to the specific metric used. For each four graphs, which are all under the same learning algorithm, I will be listing out the algorithm, hyperparameter, and metric used in label.

SVM Heatmaps: Kernel=rbf, Accuracy



LOGREG Heatmaps: Solver=saga, Accuracy



DT Heatmaps: Criterion=gini, Accuracy

References

[1] R. Caruana and A. Niculescu-Mizil. "An emperical comparison of supervised learning algorithms." *In Proceedings of the 23rd international conference on Machine learning, 161-168, 2006.*

[2] J. Fleischer. (2021, March 08). *Lecture_19_model_selection.ipynb* GitHub. `https://github.com/jasongfleischer/UCSD_COGS118A/blob/main/Notebooks/`

Lecture_19_model_selection.ipynb

[3] *Exhaustive search over specified parameter values for an estimator* scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[4] *Decision Tree Classifier* scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[SVC] *C Support Vector Classification* scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html