COGS 118A: Algorithm Comparison

Jacob Schenberg
A1554104

Abstract

This paper serves as a replication of the work done in the Caruana and Niculescu-Milzi (CNM06) paper written in 2006 by comparing three different supervised machine learning algorithms across four different datasets. The algorithms utilized are logistic regression, SVM, and KNN, and the datasets were collected from the UC Irvine repository of machine learning datasets. These three algorithms were repeated per each dataset in order to find which one performed better in comparison to the others. Each dataset underwent one-hot-encoding in order to clean up the data and were put through hyperparameter tuning per each algorithm. The results seem to mimic the findings of CNM06 as SVM led the way with the best performance followed by KNN and Logistic Regression.

1. Introduction

For this project, the purpose is to compare three different supervised machine learning algorithms and determine which performs better. The three used were Logistic Regression, SVM, and KNN.

The project attempted to follow the steps that were taken in CNM06 as the algorithms provided try to show the work done in the paper. Not only are the steps simulated, but the accuracy of the algorithms is seen to be similar as well.

To make the results that much similar to CNM06, the datasets that were used were taken from the same repository that the paper used which were located at the UC Irvine repository of machine learning datasets.

2. Methodology

2.1 Learning Algorithms

For this project, three algorithms were utilized: Logistic Regression, SVM, and KNN. This section talks about the algorithms and how they were implemented.

Logistic Regression (LOGREG): Uses a pipeline with the LogisticRegression() classifier to plug into a GridSearch with max iterations of 5000. The penalties of “l1”, “l2”, and “none” are analyzed with either the solver “saga” or “lbfgs”. We use this classifier to find best results for each dataset.

SVM: Uses the kernel “rbf” with parameters of “C” and “gamma” for the SVM classifier. This classifier is plugged into a GridSearch to output the best model for the data.

KNN: Finds 26 different values of k (i.e. 1,5,9,13,....,105) between a range of 1 and 100. Euclidean distance is used as the metric for the KNN GridSearch and cross-validation helps find results for each dataset.

2.2 Data Sets

Four datasets were used for this project and were taken from the UC Irvine repository of machine learning datasets. They are as follows: “AI4I 2020 Predictive Maintenance Dataset” (PRED MAINT), “Nursery Dataset” (NURSERY), “Electrical Grid Stability Simulated Data Dataset” (ELEC GRID), and “Chess (King-Rook vs. King) Dataset” (CHESS) (see Table 1).

Since maintenance datasets are typically hard to obtain, the PRED MAINT dataset was established to provide a synthetic dataset that reflects real predictive maintenance. It consists of 14 attributes in columns and 10,000 features that were tested. The NURSERY dataset was made from a decision model whose purpose was to rank applications for nursery schools. It consists of 8 attributes in columns and 12,960 features that were tested. Next, the ELEC GRID data set was established as a local stability analysis of the 4-node star system that was utilizing the Decentral Smart Grid Control concept. It consists of 14 attributes in columns and 10,000 features that were tested. Lastly, the CHESS data set was made to establish a chess database for the White King and Rook against the Black King. It consists of 6 attributes in columns and 28,056 features that were tested.

At the time of download, all datasets contained strings which were utilized for one-hot-encoding and then dropped in order to be put through a grid search. Not only that, but each dataset was randomized with a sample of train size 5000.

Table 1. Dataset Summary

	# ATTR	TRAIN SIZE	TEST SIZE
PRED MAINT	14	5000	10000
NURSERY	8	5000	12960
ELEC GRID	14	5000	10000
CHESS	6	5000	28056

3. Experiment and Results

Each algorithm was used on each dataset which means three algorithms were used four separate times. However, to distribute the results more effectively, each algorithm per dataset used five different classifiers with random samples of data. This means that sixty tests were conducted over the course of this experiment. For each combination, 5000 data samples were randomly selected to conduct the grid search. With this, the most efficient hyperparameters were utilized to use the random sample in GridSearch.

Prior to being placed into a GridSearch, however, the datasets were cleaned by using one-hot-encoding and then unnecessary data and unusable strings were dropped so that the code

could run smoothly in the GridSearch. It took me a couple of hours to realize that the reason my one-hot-encoding wasn't working is because of this unnecessary data.

The GridSearch for all algorithms helped find "accuracy", "roc auc", and "f1 micro". Unfortunately, I found myself really low on time, so when I was running my code for SVM, I opted for a confusion matrix and a classification report in order to quickly print out precision, recall, f1-score, and accuracy scores. From this, the best results were found out of the three thresholds. A heat map for Logistic Regression was used to convey accuracy, a heat map for SVM to display the training error, and a dotted line plot for KNN to show the mean error in terms of k values. The Logistic Regression heat map was used from Professor Fleischer's GitHub, the SVM heat map was used from a past homework assignment, and the KNN dotted line plot was used from a scikit-learn helper.

As stated before the heat map in Logistic Regression takes the data from a table that prioritizes the accuracy of the score that was found and simulates a heat map accordingly. On the other hand, the heat map in SVM prioritizes the training error. The dotted line plot for KNN that finds mean error is included to visualize the mean error change in inputs as the range of k values increases. The trend for the KNN dotted line plot is that the first couple of inputs start off far away from the rest of the inputs as they steeply drop off and exponentially continue leaving the predecessor input as an outlier. The range of k values in the KNN dotted line plot is between 1 and 100. The results show that at around 100, the inputs become the same so it isn't worth running code for three hours for a plot with 5000 inputs that looks identical to the one with 100.

After using the GridSearch in KNN and Logistic Regression, the results are printed and the values of the best parameters for accuracy, f1-score, and AUC are made available. They are then stored in different models and then the data is modeled.

For the results, the average of the algorithms per the five classifiers was found by adding up the results that appeared in the outputs in the code and dividing for the mean. This is how the information in Figure 2 came to be. Unfortunately, the code was incomplete as it took too long to run through all iterations, so it's missing data for the f1-score and precision in the logistic regression algorithm datasets. The same thing occurred for the later portion of KNN, so the data seen in Table 2 for KNN was inputted from the classification report shown in the code.

Table 2. Mean Test Set Performance Across all Algorithm/Dataset Data

MODEL	DATASET	ACCURACY	F1-SCORE	PRECISION	MEAN
Logistic Regression	PRED MAINT	2.217	N/A	N/A	**N/A**
Logistic Regression	NURSERY	28.246	N/A	N/A	**N/A**
Logistic Regression	ELEC GRID	5.9144	N/A	N/A	**N/A**

Logistic Regression	CHESS	1.849333	N/A	N/A	**N/A**
SVM	PRED MAINT	99.364	97	90	**95.455**
SVM	NURSERY	75.5873854	70.5	73	**73.029**
SVM	ELEC GRID	100	100	100	**100**
SVM	CHESS	100	100	100	**100**
KNN	PRED MAINT	100	99	99.5	**99.5**
KNN	NURSERY	69	64.5	65	**66.17**
KNN	ELEC GRID	100	100	100	**100**
KNN	CHESS	100	100	100	**100**

4. Conclusion

As a whole, this project emulates the CNM06 findings in the sense that SVM performed better overall than Logistic Regression and KNN. I previously stated that my code was lacking much needed information as I ran out of time, so by opting to use a confusion matrix and a classification report, my numbers are very different than that of CNM06. A take away from the project is that the datasets of ELEC GRID and CHESS have a much higher accuracy score across the board than the other datasets which is in part to my utilization of one-hot-encoding as the data was cleaned in a fashion that made it easier to run the algorithm successfully through it.

All three algorithms seemed to perform consistently throughout their iterations, but Logistic Regression had significantly worse accuracy than KNN or SVM. SVM seemed to have the most successful accuracy scores as the training error (as seen in the heat maps) is quite low.

5. Bonus

I'd say the only thing that constitutes for "bonus points" would be my usage of a confusion matrix and a classification report, or my inclusion of a dotted line plot in KNN to convey mean error in terms of k.

Acknowledgements

I would like to first acknowledge Professor Fleischer for being so generous as to extend the due date on this project. I was personally struggling to understand and execute the project, and it was quite nice to have an extension on something I knew I wasn't going to do very well on. Not only that, but he provided a GitHub for "model selection" as to which I modeled my Logistic Regression classifier after and used the code to find my accuracy, roc auc, and f1 results

included in my results. I also took advantage of the past homework assignments for my SVM code as I used the SVM classifiers that we have previously worked on to shape my SVM work and develop my heat maps, so I would like to acknowledge that. Lastly, I'd like to acknowledge scikit-learn for providing useful imports that my code would not work without.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

J. Fleischer. "Lecture_19_model_selection". https://github.com/jasongfleischer/UCSD_COGS118A/blob/main/Notebooks/Lecture_19_model_selection.ipynb.

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

S. Robinson. "K-Nearest Neighbors Algorithm in Python and Scikit-Learn" https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/