A Comparison of Supervised Machine Learning Algorithms at Binary Classification Tasks

Ashna Sood **ASHNASOOD@UCSD.EDU**
Department of Cognitive Science
University of California, San Diego

Abstract
Adapting the framework from the Caruana and Niculescu-Mizil 2006 paper (CNM06), this report is a replication of the analysis performed to investigate various Machine Learning (ML) algorithms and compare their respective performances on multiple binary classification problems. This is a modified replication of the CNM06 paper as I will only be comparing four different Supervised Machine Learning algorithms -- Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forests, and utilizing the Area Under the Receiver Operating Characteristic Curve (ROC AUC), Accuracy, and F1 score as the metrics to evaluate model performance across four different datasets. After finding the optimal hyperparameters using grid search, each trained model was evaluated with the three performance metrics and ultimately the Random Forests models outperformed the other algorithms, similar to the results in the CNM06 paper.
Keywords: Supervised Learning Algorithms, Binary Classification, Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forests, CNM06

1. Introduction

The CNM06 paper focuses on performing a thorough comparison of the most common and newly emerged Machine Learning algorithms. Prior studies such as the STATLOG (King et al., 1995) paper was relevant and comprehensive at its time, however it is not an accurate source for comparison today since newer algorithms such as Boosted trees and SVMs have developed and gained popularity since the study was performed. The CNM06 paper highlights that certain algorithms may perform better on certain metrics compared to other metrics, and thus the paper makes sure to evaluate the models on a variety of metrics. Thus, the CMN06 paper set out to provide a detailed comparison and review of ten supervised machine learning algorithms that are evaluated on eight performance metrics, all across eleven datasets that are binary classification problems.

Specifically, Caruana and Niculescu-Mizil assess the performance of SVMs, Artificial Neural Networks, Logistic Regression, Naive Bayes, KNN, Random Forests, Decision Trees, Bagged Trees, Boosted Trees, and Boosted stumps. Each of the model predictions are both calibrated, using both Platt Scaling and Isotonic Regression, and also not calibrated as means of comparison of model performance. The performance metrics used are divided into three subcategories: threshold metrics, ordering/rank metrics, and probability metrics. The threshold metrics used are Accuracy, F1 score, and Lift Curves. The main focus for threshold metrics is whether the prediction is above or below a threshold, rather than the actual proximity of the prediction to the threshold. The ordering metrics are the Area Under the Receiver Operating Characteristic Curve, average precision, and the precision/recall break even point. These ordering metrics focus on the ordering of the cases and the amount of positive cases ahead of negatives. The probability metrics used are squared error and cross-entropy, which are used to calculate the likelihood of that prediction being in the positive class. In order to compare performance metrics, all metrics were scaled between 0 and 1. The comprehensive use of metrics in the CNM06 paper sets this paper as a standard to compare replications of similar training with. Ultimately, the CNM06 paper discovered that the calibrated boosted trees performed the best overall, with the Random Forests algorithm performing second best. However, it is important to note that sometimes even the best algorithms perform poorly and likewise sometimes the poorly performing algorithms outshone in certain trials.

In this paper, I attempt to replicate a similar approach and compare a smaller selection of supervised machine learning algorithms using fewer metrics across a smaller subset of datasets. Specifically, I will be comparing the Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forests algorithms and evaluating their performance on the Area Under the Receiver Operating Characteristic Curve (ROC AUC), Accuracy, and F1 score. Three of the datasets I have selected are from the UC Irvine Machine Learning Repository and the last dataset is from the StatLib repository and was used in (Perlich et al., 2003), with three of these datasets (Letter, Covertype, Calhousing) being utilized in the CNM06 analysis. Following a similar procedure as the paper, for each Machine Learning algorithm, I perform 5 folds stratified cross validation using grid search to select and tune the model with the best hyperparameters to yield the best results for the three selected metrics. This smaller scaled replication is performed in attempts of verifying that similar results are attainable to the CNM06 paper when a smaller subset of variables are used, and ultimately to determine which algorithms perform better at binary classification problems.

2. Methodology

2.1. Learning Algorithms

The following descriptions explain the four different Supervised Machine Learning Algorithms compared in the analysis and which hyperparameters are explored and tuned during the grid search. The hyperparameters listed are mainly replicated from the procedure done in the CNM06 paper for those algorithms, with a couple hyperparameter adjustments that are justified below.

Logistic Regression: The Logistic Regression models were trained as both regularized by using the L2 ridge regularization, and unregularized by setting the penalty parameter to None. I varied the C regularization parameter from 10^{-8} to 10^{4} by factors of 10, as the CNM06 paper had performed. After exploring multiple solvers (liblinear, saga, lbfgs), I ultimately decided to keep the default solver lbfgs as it handles both L2 regularization and no regularization. Finally, I also increased the max_iter parameter to 2000 to account for a classifier__C error appearing during the training, thus allowing for more iterations before the lbfgs solver converges.

SVM: For the SVM models, I chose to use the standard svm.SVC() models instead of SVMLight like the CNM06 paper used as this was the type of SVM model that was taught in the COGS 118A curriculum. The SVM models were trained using the linear, polynomial, and Radial Basis Function (RBF) kernels. The C regularization parameter is ranged from 10^{-7} to 10^{3} by factors of 10, as the CNM06 paper had performed. To account for the varying degrees of the polynomial kernels, the degree parameter was set to both 2 and 3. Similarly, for the RBF kernel, the gamma parameter (or the inverse of the width of the Gaussian curve), is set to {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2}.

KNN: The K-Nearest Neighbors models were trained on a range of 26 different evenly spaced K values that were incremented by 4, starting at K = 1 and ending at K = 101. Although the CNM06 paper ranged it's last value up to K = the size of the training set (n), I chose to use a smaller subset of values as suggested by the professor, as I felt that the results would not improve drastically if I had increased my K values up to 500 (n/training size). The distance metric used in the KNN models is the Euclidean distance, however during prediction the uniform and the inverse of the distance weights are incorporated. Additionally, I decided to vary the algorithm used to compute the nearest neighbors -- BallTree, KDTree, and Brute-force search, to see which algorithm performs the best.

Random Forests: Based on the results from the CNM06 paper, I adopted the Brieman-Cutler implementation and used 1024 trees in the forest. The Random Forests models also have the max features considered when looking for the best tree split parameter set to [1, 2, 4, 6, 8, 12, 16, 20].

For all four algorithms, I standardized the values using StandardScalar() to have a mean 0 and standard deviation of 1. Although the CNM06 paper chose to not standardize the data for the Random Forests model as it is not necessary, I still chose to (as an exploration if it would impact the results), even though I was aware that the algorithm is not sensitive to the variance and magnitude of the data points.

2.2. Performance Metrics

As described before, the three performance metrics evaluated on the four Machine Learning algorithms are the Area Under the Receiver Operating Characteristic Curve (ROC AUC), Accuracy, and F1 score.

The ROC AUC score measures the area under the ROC curve (plots the True positive rate vs the False positive rate) and is a way to measure the performance of the model in its ability to classify between the positive and negative class, whose value ranges from 0 to 1. The ROC AUC score displays the probability that a model will rank a positive sample before a negative sample, thus measuring how well the predictions are ranked. The higher the ROC AUC score is and is closer to 1, the better the model is able to perform in the binary classification problem. When the value is 1, the model perfectly classifies the positive instances from the negative instances. When the value is 0, the model incorrectly predicts the positive cases as negative and vice versa. A ROC AUC score of 0.5 indicates that the model cannot distinguish between the positive and negative cases, indicating random chance.

Accuracy measures the amount of correctly predicted samples out of all the predictions and is a value ranging from 0 to 1. Although accuracy is a standard metric used for model comparison, it is not the best at capturing the actual performance of a model trained on an imbalanced dataset. Because the dataset is biased towards a certain outcome, even if the model is poorly trained, it will still return a higher accuracy since it has a high probability of correctly predicting the value to belong to the imbalanced class. Thus, metrics such as ROC AUC and F1 Score are also used to get a more accurate assessment of the performance of the models trained on imbalanced datasets. The equation to calculate accuracy is displayed here.

$$Accuracy = \frac{Correct\ Predictions}{All\ Predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

The F1 Score is a metric that balances both precision and recall through its harmonic mean and measures the accuracy of a model on a dataset. Precision measures how many of the model's positive predictions are actually positive (minimizing false positives). Recall, also known as Sensitivity, measures how well the model correctly classifies the actual positive samples. (minimizing false negatives). To examine the performance of the model, it is not enough to measure just precision or recall, but rather it is ideal to measure both combined since there is a tradeoff between precision and recall. The F1 Score balances out both precision and recall and is a good metric for imbalanced datasets. The score ranges from 0 to 1, where 1 indicates both a perfect precision and recall, and 0 is when either precision or recall is 0. The equation to calculate the F1 score is displayed here.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives} \qquad precision = \frac{true\ positives}{true\ positives + false\ positives}$$

2.3 Datasets

The four datasets that are trained on the various machine learning algorithms are all binary classification problems. Three of the datasets used in this analysis are from the UC Irvine Machine Learning Repository and the last dataset is from the StatLib repository and was used in (Perlich et al., 2003). In order to compare the performance and see if I could replicate the CNM06 results, I chose three datasets that were utilized in the paper -- Letter, Covertype, Calhousing. The last dataset I chose to train the algorithms on is also from the UCI repository, and is the Dry Beans dataset. All four datasets have been converted to continuous values by one-hot encoding any categorical variables. In my initial analysis, I had chosen the UCI Adult dataset, but later realized how the high number of variables (108 columns) due to the one hot encoding of the categorical variables immensely slowed down the training of the models. Regardless, I include the data cleaning performed on that dataset as a reference. *Table 1* (displayed below) describes each dataset's characteristics.

The BEANS dataset initially was used as a multi-class classification problem as there are 7 dry beans classes. In order to convert it into a binary classification problem, I chose the two highest frequency classes and classified those data points as positive (1) and the rest as negative (0) in order to create a more balanced positive/negative samples ratio.

The LETTER dataset was similarly a multi-class classification problem that categorized different images of the 26 letters. Similar to the CNM06 LETTER.p2 procedure, I binarized the problem by assigning the letters A-M to the positive class (1) and the rest as negatives (0) to create two balanced classes.

The COVTYPE dataset predicts the forest cover type with seven class categories. Similar to the CNM06 paper, I set the highest frequency class as the positive class (1) and the other 6 classes as the negative class (0).

The CALHOUS dataset was originally a dataset used to predict the median value of a house. This dataset was converted into a binary classification problem by binarizing the median house value variable. Similar to the (Perlich et al., 2003) paper, if the median house value was above \$130,000, then the data sample was classified as positive (1) and if less than or equal to \$130,000, the sample was in the negative class.

Lastly, within the ADULT dataset (which eventually was not used in the final analysis), I first binarized the sex column, making female 1 and male 0, and also binarized the income variable. If the income was greater than or equal to \$50K, then the sample was positive (1) and otherwise negative (0). Lastly, in order to perform binary classification, I one hot encoded seven nominal variables, which created 100 new columns. Ultimately, the adult dataset was too computationally expensive as the training time on the SVM algorithm in particular took over a day to complete. Thus, it was removed from the final analysis.

Table 1. Description of problems

PROBLEM	# ATTRIBUTES	TRAIN SIZE	TEST SIZE	% POZ
BEANS	17	5000	8,611	45%
LETTER	17	5000	15,000	49%
COVTYPE	13/54	5000	576,012	50%
CALHOUS	9	5000	15,640	24%

3. Experiment

Each of the four algorithms contains its own training loop with the same general structure, each with different hyperparameters pertaining to that algorithm. For each algorithm and dataset combination, there are five trials where within each trial a random subset of 5000 samples are randomly selected as the training set, and the rest are allocated as the test set. The particular hyperparameters for that algorithm and performance metrics are then defined and assigned to the current algorithm through a pipeline. In order to create multiple models that each have a unique subset of hyperparameters (for example, in a SVM that uses a polynomial kernel, a degree parameter must be specified, and for a SVM with a rbf kernel, a gamma parameter is required), a search space is created with different learning algorithms and their required hyperparameters. After creating the grid search and training the models using 5 folds Stratified Cross Validation, where roughly each fold has an even distribution of the two classes, I evaluated which hyperparameters yielded the best results for the three metrics using the ranked hyperparameters array. Within each K-fold, 4000 of the samples were allocated for training the models, and the remaining 1000 samples were the validation set to select the best hyperparameters. I then trained 3 separate models with the specific optimal hyperparameters for that metric using the 5000 samples training set, and finally evaluated the performance using the remaining test set samples. After storing the AUC, Accuracy, and F1 score from the model performance for the current trial, this process is repeated five times. The AUC, Accuracy, and F1 score metrics are then averaged across the 5 trials and stored for that particular dataset. This process was repeated for all four datasets, and ultimately the averaged performance across the datasets (averaged across columns) and across metrics (averaged across rows) are saved (*Tables 2 and 3*). These values are averaged in order to compare performance across algorithms at the end of the experiments, evaluating which algorithms performed better in the different binary classification problems.

4. Results

4.1. Performances by Metric

The results displayed below in *Table 2* are the normalized average metrics for each Machine Learning algorithm, and the respective means across metrics. Each metric was averaged across the columns, meaning across all four datasets and five trials per dataset, to result in three averaged metrics for each algorithm. Within the table, the best performing algorithm's value for each metric is **bolded**, and the values within that metric's column that are asterisked (*) if there is not a significant difference between that value and the bolded value. The statistically significant difference between the highest performing algorithm and every other algorithm within each metric column was determined using an Independent t-test, specifically between the raw values for those specific averages in the column. An Independent t-test was used as the random seed was not set for each trial to ensure that the same draw is taken out for each algorithm, thus the samples were independent. If the p-value was less than or equal to 0.05, then it is statistically significant and we can reject the null hypothesis, indicating that there was a significant difference between the highest performing algorithm's value and the current value. If the p-value was greater than the threshold p = 0.05, then it is not statistically significant and you fail to reject the null hypothesis, indicating that there was not a significant enough difference between the highest performing algorithm's value and the current value. Thus, for the values in *Table 2* with a p-value greater than 0.05, there is an asterisk next to the number. The p-values corresponding to *Table 2* are in *Appendix A Table 6.1*. However, it is important to note that there could be a significant difference between two close values and no statistical difference between two different close values, due to a higher amount of variance. The final column in this table is the mean of the normalized metrics, four datasets, and five trials for each algorithm (taken across the rows).

Table 2. Normalized scores for each learning algorithm by metric (average over four problems)

MODEL	AUC	ACCURACY	F1 SCORE	MEAN
LR	0.866*	0.799*	0.812*	0.826
SVM	0.840*	0.768	0.796*	0.801
KNN	0.858*	0.815*	0.843*	0.839
RF	**0.961**	**0.911**	**0.917**	**0.930**

Based on the performance values in *Table 2*, it is evident that overall the Random Forests algorithm performed the best across all metrics, with a ROC AUC score of 0.961, an Accuracy of 0.911, and an F1 Score of 0.917. It is important to note that most of the values in the table also have asterisks, indicating that there was not too significant of a difference between their value and the Random Forest's best values, indicating that overall the algorithms all performed fairly well, with KNN performing second best followed by SVM and Logistic Regression being fairly similar. For example, within the AUC metric, each algorithm had values that were not significantly different from the Random Forests value. These results mainly correlate with *Table 2* in the CNM06 paper, where Random Forests is one of their strongest performing algorithms across metrics. This was expected as Random Forests is one of the better performing algorithms due to the group consensus of numerous decision trees, the splitting of a smaller subset of features for each tree, and it's strength in handling imbalanced classes.

Additionally, *Appendix A Table 4* displays the mean training set performance averaged across the four algorithms, and was calculated as a means of comparison with Table 2's test set values. In the training set performance, the Random Forests model similarly performed the best across the metrics, with the KNN algorithm trailing behind with slightly lower values. This trend between Random Forests and KNN was not as close in the testing set performance, as the KNN values were a bit lower, but still there was not a statistically significant difference between the KNN and Random Forests values. Overall, all of the training set normalized values are slightly higher than the test set performance, mainly attributing to the fact that the models were trained on the training set, and then evaluated on the same data, whereas the testing set was brand new to the models. However, the training values are not significantly higher than the test set values, indicating that the models were not overfitted with the training data. Otherwise, the testing performance would have been quite low if the model was not generalized well.

4.2. Performances by Learning Algorithm

The results displayed in *Table 3* are the normalized average metrics scores across datasets for each Machine Learning algorithm, and the respective means across the four datasets. Each metric was averaged across the rows, meaning across all three metrics and five trials per dataset, to result in four averaged dataset scores for each algorithm. A similar procedure from Table 2 is followed where the highest performing algorithm per dataset is bolded, with the non-significant difference values within that column asterisked (*). These values were similarly determined by comparing the p value to the threshold p = 0.05, where a value lower than 0.05 indicated a significant difference and a value higher did not indicate a significant difference. The p-values corresponding to *Table 3* are in *Appendix A Table 6.2*.

Table 3. Normalized scores of each learning algorithm by problem (averaged over three metrics)

MODEL	BEANS	LETTER	COVTYPE	CALHOUS	MEAN
LR	0.930*	0.754	0.749	0.869	0.826*
SVM	0.813	0.903	0.711	0.779	0.801
KNN	0.859	**0.968**	0.806	0.721	0.839*
RF	**0.980**	0.962	**0.847**	**0.930**	**0.930**

Based on the performance values in *Table 3*, the Random Forests algorithm similarly performed the best overall across the datasets, with an exception to the LETTER dataset where the KNN algorithm slightly outperformed. For the BEANS dataset, the data was fairly split between the positive (45%) and negative (55%) classes, so all of the algorithms are well-equipped to handle balanced datasets. From the table, it is evident that the Logistic Regression model also performed fairly well with the BEANS dataset as there is an asterisk next to it’s value, indicating that there was not a statistically significant difference between the Random Forests and Logistic Regression scores. It is interesting to note that in general the Logistic Regression model performed poorly amongst the four models, but for some reason on the BEANS dataset it had a higher performance compared to both SVM and KNN. This variation in the Logistic Regression performance could be due to random chance, and perhaps with more trials, the results could have been different, as on average the Logistic Regression model is not as good of a classifier as for example SVMs which have the advantage of using different kernels. For the LETTER dataset, the data was almost exactly split between the positive (49%) and negative (51%) classes, it was a little surprising that the KNN model performed better only for this dataset, even though the classes were fairly balanced and the Random Forests model generally performed the best in all other datasets with similar evenly split data. Additionally, there was not even an asterisk next to the Random Forests score, despite the values being 0.968 (KNN) and 0.962 (RF), which are numerically very close. Similarly, the SVM performance was also around 0.903, but it's independent t-test metric indicated that the difference was statistically significant. However, this significant difference between the two values could be attributed to a higher variance for the Random Forests score. The variation in the LETTER dataset could also have been due to chance and could have had different results with perhaps more trials, if time permitted. For the COVTYPE dataset, the data is exactly split evenly between the positive and negative classes (50%/50%), and the Random Forests model performed the best with a fairly significant difference in values between the other models. Lastly, for the CALHOUS dataset, the data was quite imbalanced as there were only 24% positive samples and 76% negative samples. Regardless, the Random Forests model handled the imbalanced data and performed the best by a wider margin compared to the other models (0.93). Ultimately, there were only some non-significant differences between the mean values across the models, and most values in the dataset columns were significantly lower than the highest performing algorithm value. Regardless, the algorithms all performed fairly well on the different binary classification problems. If time was further permitted, I would increase the number of trials to see if the similar patterns hold.

5. Discussion & Conclusion

Because the data in this experiment was not calibrated, the values in Tables 2 and 3 are the raw model predictions, where higher values denote better performance. Thus, these values can be compared with the CNM06 Table 2 where the calibration column has a "–", indicating that there was no calibration for that model. As observed from the CNM06 results, their calibrated boosted trees model overall performed the best, with Random Forests being the second best learning algorithm. The results conducted in this paper similarly follow the trends from the CNM06 paper as the Random Forests models here performed the best overall both across metrics and across the various binary classification datasets. The model that relatively did not perform the best in the experiment was the Logistic Regression model, similar to the CNM06 results where Logistic Regression was one of their lowest performing algorithms. Similarly, the KNN model here was overall the second best model and consistently performed well, whereas in the CNM06 paper it was not one of their best performing models as they had trained and tested other complex models such as boosted trees and bagging. The SVM models here performed fairly well and in similar range as the CNM06 analysis, but was not one of the best performing models, as in CNM06. However, in the CNM06 paper, their calibrated SVM models performed better. When comparing the exact values of the model performances between this paper and the CNM06 study, some of the values slightly differ, but are relatively in the same range, indicating a fairly consistent and replicable performance of the CNM06 structure.

As a means of ensuring that the results I obtained were consistent within the 5 trials, I repeated all experiments and trials a second time and received very similar results that were minutely different. Ultimately, the last run of all the models are the final results displayed in the paper. In the future, in order to have maximum statistical power, I would have liked to perform a Paired t-test instead of an Independent t-test between the different algorithm's scores by setting the random seed so that for each trial the same random sampling is happening for the different algorithms. Additionally, when comparing the p values at a threshold of $p = 0.5$ from the independent t tests, it can be problematic as some of the differences between scores that are calculated to be significantly different may not actually be truly significant. Thus, if a lower p value of $p = 0.1$ was used as the threshold, then more values would be labeled as not statistically different in value (with the *). Thus, there would be fewer false positives in the table, whose values are relatively close and truly not significantly different but were labeled as significantly different.

If further time was permitted, I would have liked to explore more Machine Learning algorithms, specifically boosted trees since their performance was the best overall in the CNM06 paper. Similarly, I would have liked to have used all five datasets that I had explored, including the Adult dataset, and compared performance across the bigger datasets. In order to get more comprehensive results, I would have liked to also evaluate the models on more metrics as the CNM06 paper performed in order to complete a closer replication of the paper and even further validate the performance of the models.

The main objective of this research project was to replicate a smaller scaled version of the Caruana and Niculescu-Mizil 2006 paper (CNM06) and determine which Supervised Machine Learning algorithms generally perform better on various binary classification tasks. Ultimately, the results gathered from the experiments performed on the Logistic Regression, SVM, KNN, and Random Forests models all evaluated on the ROC AUC score, Accuracy, and F1 score metrics using four generally balanced datasets are fairly consistent with the CNM06 paper, where the Random Forests model overall outperformed the other models. Therefore, these results further validate the findings from the CNM06 paper.

6. Bonus

As per the original instructions for the project to explore at least three different Machine Learning algorithms in the project, this project additionally explores the Random Forest algorithm. This algorithm was briefly taught during lecture and based on prior experience, I knew the algorithm generally yields good results due to the accumulation of individual decision trees and also its strong performance on imbalanced datasets. Thus, I wanted to see how well the algorithm would perform with the given binary classification problems. Additionally, the extended data cleaning and training I had performed on the Adult dataset had consumed multiple days of training time, as the combination of the large dataset and the extensive hyperparameters was computationally expensive. I chose to initially explore the Adult dataset as I wanted to gain experience converting the categorical variables into continuous values, and further explore how the one-hot encoded variables aided in the classification of the data. In order to best replicate the results of the CNM06 paper, I attempted to use all the C values for the SVM model with the Adult dataset and would even create smaller subsets of C values to see which range was best, but these experiments ended up taking multiple days to run. Additionally, to determine the consistency in the results, I would run the four models with all the datasets including Adult multiple times. Ultimately, I believed that the Dry Beans dataset would be easier to replicate in future studies due to its smaller size, so I discarded my Adult dataset results in the final notebook. However, I am proud of the extra work and dedication I devoted to understanding the nuances of the Adult dataset and its algorithmic performances. The initial data cleaning performed on the Adult dataset is featured in the code pdf attached below, and the overall performances from the Adult dataset are featured in Appendix B.

Acknowledgements

I would like to thank the COGS 118A Professor Jason Fleischer and the teaching staff, specifically TA Abdullah Al-Battal for their continuous support and encouragement throughout the quarter. Their extended support through lectures, discussions, over email, and in office hours helped clarify my confusions and furthered my insight on how to complete the project. The assignments, both written and coding, throughout the quarter and the guided lessons strengthened my Machine Learning knowledge and confidence to be able to train such complex models with numerous hyperparameters, while also understanding the purpose behind each algorithm and its strengths and weaknesses. Lastly, I am thankful for my fellow classmates and peers who were always helpful during discussions, office hours, and on Piazza to clarify my questions about the project.

References

California Housing Dataset. StatLib Repository, www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html.

Caruana, R., and Niculescu-Mizil, A. (2006). An Empirical Comparison of Supervised Learning Algorithms. *Proceedings of the 23rd International Conference on Machine Learning*

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Fleischer, J. Lecture_19_model_selection.Ipynb. *GitHub*, 8 Mar. 2021, github.com/jasongfleischer/UCSD_COGS118A/blob/main/Notebooks/Lecture_19_model_selection.ipynb.

"Metrics and Scoring: Quantifying the Quality of Predictions". *Scikit-Learn: Machine Learning in Python*, scikit-learn.org/stable/modules/model_evaluation.html.

Perlich, C., Provost, F., & Simonoff, J. S. (2003). Tree Induction vs. Logistic Regression: A Learning-Curve Analysis. J. Mach. Learn. Res., 4, 211–255.

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

"Scipy.stats.ttest_ind - SciPy v1.6.1 Reference Guide." *SciPy.org*, 2021, docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html.

Appendix A

Table 4. Mean training set performance for the optimal hyperparameters on each of the dataset/algorithm combos

MODEL	AUC	ACCURACY	F1 SCORE	MEAN
LR	0.871	0.803	0.817	0.830
SVM	0.851	0.773	0.801	0.809
KNN	0.999	0.967	0.970	0.979
RF	0.999	0.999	0.999	0.999

Table 5. Raw test set scores

MODEL	DATASET	METRIC	TRIAL 1	TRIAL 2	TRIAL 3	TRIAL 4	TRIAL 5
		AUC	0.987	0.986	0.987	0.897	0.986

Logistic Regression (LR)	**BEANS**	**ACC**	0.947	0.944	0.950	0.784	0.948
		F1 SCORE	0.942	0.940	0.946	0.766	0.944
	LETTER	**AUC**	0.812	0.809	0.813	0.811	0.810
		ACC	0.728	0.724	0.728	0.723	0.722
		F1 SCORE	0.729	0.727	0.732	0.723	0.724
	COVTYPE	**AUC**	0.799	0.780	0.791	0.793	0.799
		ACC	0.731	0.726	0.717	0.716	0.731
		F1 SCORE	0.725	0.727	0.724	0.725	0.735
	CALHOUS	**AUC**	0.891	0.892	0.888	0.890	0.889
		ACC	0.831	0.834	0.829	0.831	0.830
		F1 SCORE	0.885	0.887	0.883	0.885	0.885
Support Vector Machines (SVM)	**BEANS**	**AUC**	0.860	0.864	0.866	0.861	0.864
		ACC	0.782	0.781	0.785	0.780	0.786
		F1 SCORE	0.792	0.789	0.792	0.789	0.795
	LETTER	**AUC**	0.952	0.953	0.951	0.952	0.951
		ACC	0.880	0.879	0.877	0.880	0.881
		F1 SCORE	0.879	0.879	0.876	0.880	0.879
	COVTYPE	**AUC**	0.756	0.758	0.752	0.752	0.755
		ACC	0.694	0.698	0.692	0.695	0.696
		F1 SCORE	0.677	0.694	0.676	0.684	0.681
	CALHOUS	**AUC**	0.825	0.751	0.806	0.784	0.787
		ACC	0.712	0.713	0.712	0.714	0.716
		F1 SCORE	0.832	0.832	0.832	0.833	0.835
	BEANS	**AUC**	0.919	0.911	0.912	0.923	0.916
		ACC	0.842	0.799	0.841	0.832	0.849
		F1 SCORE	0.835	0.799	0.838	0.828	0.842

K- Nearest Neighbors (KNN)	**LETTER**	**AUC**	0.991	0.991	0.990	0.990	0.991
		ACC	0.959	0.957	0.957	0.958	0.956
		F1 SCORE	0.959	0.957	0.957	0.958	0.956
	COVTYPE	**AUC**	0.859	0.859	0.861	0.861	0.61
		ACC	0.777	0.780	0.782	0.779	0.785
		F1 SCORE	0.779	0.777	0.776	0.775	0.784
	CALHOUS	**AUC**	0.670	0.666	0.668	0.665	0.654
		ACC	0.697	0.691	0.679	0.693	0.696
		F1 SCORE	0.813	0.802	0.812	0.804	0.811
Random Forests (RF)	**BEANS**	**AUC**	0.996	0.996	0.995	0.995	0.996
		ACC	0.973	0.974	0.973	0.974	0.974
		F1 SCORE	0.971	0.972	0.970	0.972	0.971
	LETTER	**AUC**	0.991	0.991	0.991	0.991	0.991
		ACC	0.946	0.948	0.946	0.946	0.949
		F1 SCORE	0.946	0.948	0.947	0.946	0.948
	COVTYPE	**AUC**	0.902	0.902	0.899	0.901	0.899
		ACC	0.823	0.820	0.819	0.820	9.821
		F1 SCORE	0.822	0.819	0.181	0.817	0.819
	CALHOUS	**AUC**	0.958	0.959	0.956	0.958	0.961
		ACC	0.902	0.904	0.900	0.897	0.905
		F1 SCORE	0.932	0.933	0.931	0.928	0.934

Table 6.1. The p-values of the comparisons across algorithms in Table 2

MODEL	AUC	ACCURACY	F1 SCORE	MEAN
LR	0.095	0.102	0.133	0.019
SVM	0.062	0.039	0.068	0.010

KNN	0.237	0.204	0.205	0.012
RF	1	1	1	1

Table 6.2. The p-values of the comparisons across algorithms in Table 3

MODEL	BEANS	LETTER	COVTYPE	CALHOUS	MEAN
LR	0.157	1.388e-10	1.203e-07	6.021e-10	0.106
SVM	4.390e-10	2.785e-12	2.222e-09	8.698e-07	0.045
KNN	3.121e-05	1	2.919e-09	3.758e-14	0.190
RF	1	3.083e-06	1	1	1

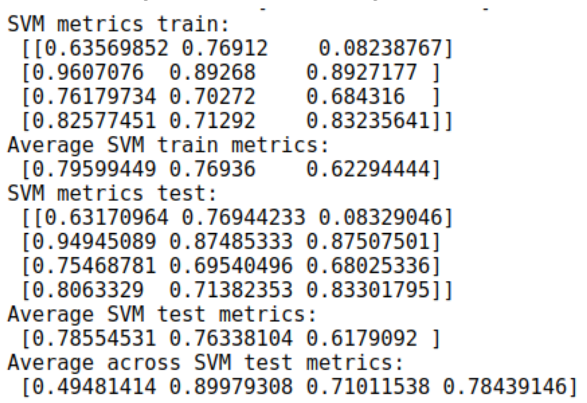
Appendix B

Results using the **ADULT Dataset**

Table 7. Description of problems

PROBLEM	# ATTRIBUTES	TRAIN SIZE	TEST SIZE	% POZ
ADULT	14/108	5000	27,561	71%

Table 8. SVM Algorithm Metrics Using the Adult Dataset



```
SVM metrics train:
 [[0.63569852 0.76912    0.08238767]
 [0.9607076  0.89268    0.8927177 ]
 [0.76179734 0.70272    0.684316  ]
 [0.82577451 0.71292    0.83235641]]
Average SVM train metrics:
 [0.79599449 0.76936    0.62294444]
SVM metrics test:
 [[0.63170964 0.76944233 0.08329046]
 [0.94945089 0.87485333 0.87507501]
 [0.75468781 0.69540496 0.68025336]
 [0.8063329  0.71382353 0.83301795]]
Average SVM test metrics:
 [0.78554531 0.76338104 0.6179092 ]
Average across SVM test metrics:
 [0.49481414 0.89979308 0.71011538 0.78439146]
```

Table 9. Normalized scores for each learning algorithm by metric (average over four problems)

Algorithm	AUC	Accuracy	F1 Score	Mean
LR	0.731	0.736	0.641*	0.703
SVM	0.786*	0.763*	0.618*	0.722*

KNN	0.781*	0.797*	0.713*	0.764
RF	**0.938**	**0.881**	**0.842**	**0.887**

Table 10. Normalized scores of each learning algorithm by problem (averaged over three metrics)

Algorithm	**ADULT**	**LETTER**	**COVTYPE**	**CALHOUS**	**mean**
LR	0.586	0.754	0.619	0.852	0.703*
SVM	0.495	0.900	0.710	0.784	0.722*
KNN	0.564	**0.967**	0.806	0.718	0.764*
RF	**0.808**	0.961	**0.848**	**0.931**	**0.887**

Table 11.1. The p-values of the comparisons across algorithms in Table 9

Algorithm	**AUC**	**Accuracy**	**F1 Score**	**Mean**
LR	0.0463	0.042	0.165	0.012
SVM	0.099	0.058	0.316	0.068
KNN	0.172	0.2450	0.445	0.032
RF	1	1	1	1

Table 11.2. The p-values of the comparisons across algorithms in Table 10

Algorithm	**ADULT**	**LETTER**	**COVTYPE**	**CALHOUS**	**mean**
LR	3.061e-09	3.211e-15	4.074e-11	1.138e-09	0.051
SVM	5.126e-08	7.826e-09	1.729e-09	2.176e-06	0.149
KNN	7.648e-09	1	7.703e-09	1.274e-08	0.248
RF	1	0.003	1	1	1