Empirical Comparison of Supervised Binary Classification Models

Siddhant Joshi[1]

Abstract

This paper presents a comprehensive comparative analysis of some common solutions to the binary classification problem–logistic regression, support vector machine, K-nearest neighbors, and perceptron classifier–across four diverse datasets. Through extensive experimentation, our survey of these models aims to produce analysis comparable to the results derived in "An Empirical Comparison of Supervised Learning Algorithms."[1]

1. Introduction

In the realm of machine learning, selecting the right algorithm for binary classification is a perennial challenge. This paper takes a pragmatic approach by conducting a thorough examination of four commonly used solutions–logistic regression, support vector machines, K-nearest neighbors, and perceptron classifiers. Our focus is on practicality, analyzing the performance of these models across four diverse datasets to provide a more informed understanding of their real-world applicability. Drawing inspiration from the study "An Empirical Comparison of Supervised Learning Algorithms,"[1] we aim to provide an evaluation grounded in extensive experimentation.

2. Methodology

We explore the performance within and across classifiers using the following procedure. We use four different classifiers, experimented on four different data sets of varying size and feature complexity. Additionally, each model is trained and tested under the following splits for each dataset: 20/80, 50/50, 80/20. Before testing, hyper-parameters for each classifier are fine-tuned using 5-fold cross validation on the training set. Finally, we use the entire training set and tuned hyper-parameters to train each classifier and report the error metrics on the test set. Each training, validation, and testing trial will be conducted three times and the average performance metrics for each experiment will be reported.

[1]Department of Cognitive Science and Halıcıoğlu Data Science Institute, University of California, San Diego, La Jolla.

2.1. Learning Algorithms

We explore common parameters for each classification model. This section summarizes the specific parameters and respective ranges that were tested for each algorithm.

2.1.1. Logistic Regression (LOGREG)

We train models varying in decision thresholds with probabilities between the interval $[0, 1]$ at increments of 0.01. We assessed each model at a learning rate of 0.001 with 5,000 iterations of gradient descent when fitting to training data.

2.1.2. Perceptron Classifier (PTC)

Models are assessed at various learning rates from the range $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$ The stopping criterion for each model was either 5,000 iterations of gradient descent or a loss equal zero, whichever occurred first.

2.1.3. K-Nearest Neighbors (KNN)

We use 20 values of k, ranging from 1 to 20 for each data split. We also leverage Euclidean distance as our similarity metric between samples.

2.1.4. Support Vector Machines (SVM)

For this analysis, we only used the linear kernel and tested various regularization parameters (C) from the range $\{0.001, 0.01, 0.1, 1, 10\}$.

2.2. Data Sets

We compare the algorithms on four carefully selected datasets that vary in both sample size and feature complexity. This way, we would be able to generate more meaningful and generalized results for each classifier. For interpretability, each dataset was also selected to be relatively balanced between classes. All datasets underwent meticulous preprocessing and feature engineering to ensure quality and integrity for downstream analyses (Table 1).

2.2.1. Fruits

This dataset is from Kaggle[5] and uses diameter, weight, and RGB color information on 10,000 citruses to determine whether it is a grapefruit or an orange. No features were one-

Table 1. Descriptions of each data set after pre-processing and feature engineering.

Data set	Features	Samples	Positive	Negative
Fruits	4	10000	5000	5000
Chess	373	20058	10951	9107
Music	28	200	100	100
Lepiota	46	8124	3916	4208

hot encoded as all values were continuous. Additionally, during pre-processing, diameter and weight were found to be extremely correlated. Therefore, we dropped one of the features to avoid redundancy. We opted to drop weight with the intention of making modelling less computationally intensive, as its values were larger in magnitude.

2.2.2. CHESS

This dataset is the largest of the four with slightly over 20,000 observations. Also from Kaggle[4], it uses features such as turns per game, opening move, and player rating to predict which player–black or white–wins a game of chess. Some features, most notably the codes for opening moves, had to be one-hot encoded.

2.2.3. MUSIC

This dataset is the smallest one for this analysis from Kaggle[3] and uses features such as tempo, spectral metrics, and feature tags extracted using lib-ROSA, to predict the genre of a song to be either classical or pop. Similar to the fruits dataset, no features were one-hot encoded as all values were continuous. Aside from song IDs, no features were dropped either.

2.2.4. LEPIOTA

This dataset was from the UCI Machine Learning Repository [2] and contained 22 categorical features on 23 species of gilled mushrooms in the Agaricus and Lepiota family. Each species is identified as edible, poisonous, or unknown. To make this a binary classification problem, the latter class was combined with the poisonous one. Many features were combined such that one metric from either sizes, colors, odors of gills, caps, or stalks were preserved. Every feature was one-hot encoded.

2.3. Performance Metrics

For the sake of simplicity and comparability across classifiers, we only report standard classification accuracy for each of training, validation, and testing trials. Additionally, given that the datasets are quite balanced between positive and negative classes, we believed standard accuracy to be a suitable metric for these models. Accuracy will be calculated as the average proportion of correct predictions made by the classifier on a set of three trials for a given dataset and split.

3. Experimental Results

3.1. Training Results

The following tables summarize training results for each classifier across each split for each dataset. There are no training metrics for KNN as the algorithm does not require training.

20/80 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.927	0.680	1.000	0.989
PTC	0.866	0.543	0.950	0.998
KNN	–	–	–	–
SVM	0.936	0.718	1.000	0.996

50/50 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.928	0.675	1.000	0.995
PTC	0.896	0.601	0.946	0.997
KNN	–	–	–	–
SVM	0.935	0.695	1.000	0.997

80/20 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.930	0.670	1.000	0.995
PTC	0.908	0.586	0.920	0.997
KNN	–	–	–	–
SVM	0.928	0.683	1.000	0.998

3.2. Cross-Validation Results

The following tables summarize the cross-validation results for each classifier across each split for each dataset. Each model was tuned using the average of three 5-fold cross-validation procedures.

20/80 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.938	0.678	1.000	0.995
PTC	0.848	0.604	0.866	0.990
KNN	0.869	0.641	0.975	0.996
SVM	0.935	0.667	0.933	0.994

50/50 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.932	0.667	1.000	0.996
PTC	0.851	0.579	0.923	0.997
KNN	0.888	0.627	0.977	0.997
SVM	0.933	0.657	0.986	0.997

80/20 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.933	0.669	1.000	0.997
PTC	0.855	0.574	0.950	0.995
KNN	0.895	0.635	0.983	0.998
SVM	0.929	0.681	0.989	0.997

Each model's respective tuned hyper-parameter is also displayed (k, regularization, learning rate, etc.) in the table below.

Hyper-parameters

Classifier	Split	Fruits	Chess	Music	Lepiota
LOGREG	20/80	0.52	0.48	0.01	0.31
	50/50	0.55	0.48	0.01	0.27
	80/20	0.57	0.57	0.02	0.30
PTC	20/80	0.01	0.001	0.001	1.0
	50/50	0.01	0.0001	0.1	1.0
	80/20	0.001	0.0001	0.1	1.0
KNN	20/80	3	16	11	3
	50/50	8	16	2	12
	80/20	5	19	4	7
SVM	20/80	0.01	1.0	0.1	1.0
	50/50	0.1	10	0.1	10
	80/20	0.1	10	0.1	10

3.3. Testing Results

The following tables display the testing results for each classifier across each split for each dataset. Each classifier was trained on the entire training set under the optimal hyper-parameter selected from above and tested on the held out testing set.

20/80 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.928	0.664	0.933	0.991
PTC	0.821	0.597	0.887	0.995
KNN	0.869	0.636	0.962	0.994
SVM	0.928	0.652	0.960	0.998

50/50 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.929	0.659	0.973	0.995
PTC	0.774	0.470	0.800	0.995
KNN	0.887	0.632	0.973	0.997
SVM	0.932	0.660	0.975	0.996

80/20 Split

Classifier	Fruits	Chess	Music	Lepiota
LOGREG	0.927	0.659	0.975	0.997
PTC	0.868	0.555	0.900	0.992
KNN	0.893	0.631	0.983	0.998
SVM	0.935	0.665	0.981	0.998

4. Discussion

The logistic regression performance was quite stable across datasets. With increasing training set sizes, we observed slight increases in accuracy. Hyper-parameter tuning was also as we expected, since balanced classes across datasets would be indicative of a decision threshold at around 50%. In the music dataset, however, we see a tuned threshold of 0.01, highlighting the high separability of those data afforded to us by the sigmoid transformation involved in the logistic regression algorithm.

The perceptron classifier was one of the more unstable models, with accuracy being lower than normal for FRUITS and CHESS, but matching the other classifiers when tested on the remaining datasets. The model did not converge during any training splits on any dataset, consistently stopping only when 5,000 iterations were reached. This may be attributed to a combination of feature complexity as well as the robust criteria to achieve an error of zero in order to stop fitting parameters. This model had some variable accuracies, as shown by the decrease in training accuracy of 50/50 splits for the datasets. Furthermore, PTC struggled the most with CHESS, achieving an average testing accuracy lower than bonehead accuracy.

Tuning the hyper-parameters for the KNN truly highlighted the difficulty models faced with CHESS. For the other three datasets, optimal k values were selected to be between 1 and

10, however when fitting to CHESS, that number shot up to above 15 neighbors. This shows that the features in CHESS might have been of low quality in terms of the classification it was attempting to perform. Beside this, KNN performance was in-line with other classifiers, consistently producing accuracies slightly below those of SVM or LOGREG.

Lastly, SVM was one of the best-performing algorithms we implemented, achieving higher average test accuracies than the other classifiers approximately 75% of the time. We also witnessed the characteristic increase in training and testing performance as the training split increased for most datasets. From this, we conclude that SVM had the best adaptability of the four classifiers across datasets. However, it was still unable to surpass performance on CHESS by a large margin, exceeding other classifiers in accuracy by 0.001.

Considering the varying complexities and sizes of the data, these results are more or less what would be expected and align quite closely to the trends identified by Caruana et al. (2006). With the exception of some splits, model performances increased with more training data and more relevant features contributed greatly to excellent testing performance. For example, MUSIC contained metrics on the spectrum readings of song clips, providing meaningful characteristics for models to leverage when classifying. LEPIOTA also contained this richness in features, having useful descriptions about most physical properties of mushrooms at a relatively high level of granularity. This is demonstrated in perfect training performance of LOGREG and SVM. On the other hand, CHESS features included opening moves and turn counts, data that would be more likely to only afford indirect predictive power.

5. Conclusion

Our comprehensive exploration of four widely-used classifiers—logistic regression, perceptron classifier, K-nearest neighbors, and support vector machines—has provided valuable insights into their performances across diverse binary classification tasks.

Logistic regression and SVM exhibited remarkable stability, showcasing incremental accuracy improvements with larger training sets. In contrast, the perceptron classifier displayed variability and struggled to converge on all datasets, especially facing challenges with CHESS. This highlights the algorithm's sensitivity to feature complexity and could also be a result of our stringent convergence criteria. KNN performance, although generally comparable to other classifiers, faced notable difficulty with the CHESS dataset during hyper-parameter tuning, indicating challenges with lower-quality features. SVM emerged as a robust performer, consistently achieving higher average test accuracies and demonstrating adaptability across datasets. However, it faced a marginal challenge in surpassing performance on the CHESS dataset compared to other classifiers.

These findings align closely with the trends reported by Caruana et al. (2006) and underscore the importance of dataset characteristics in algorithm selection. We also observed that larger, more relevant training data contributed to superior performance, as seen in datasets like MUSIC and LEPIOTA with rich feature sets. Conversely, challenges emerged with datasets like CHESS, where indirect predictive power constrained algorithm effectiveness.

Our empirical investigation illuminates the strengths and limitations of these common binary classifiers. The nuanced performance variations across datasets demonstrate the need for a pragmatic approach to algorithm selection, considering the specifics of each dataset and tuning capabilities of each classifier. The significance of empirical evaluations in guiding the choice of machine learning models for binary classification tasks is well-underscored by the experiments performed here as well as in our model paper[1]. Understanding these intricacies makes for effective deployment of supervised machine learning models in diverse real-world scenarios.

Software and References

Software

All data were pre-processed in Python using NumPy, Pandas, Matplotlib, and Seaborn. Logistic Regression, Perceptron, and K-Nearest Neighbors classifiers were built, trained, and tested from scratch using base Python, NumPy, and KFold capabilities of Sci-Kit Learn. The Support Vector Machines were trained and tested entirely in Sci-Kit Learn. All code and data can be found here: https://github.com/sidjo/Supervised-Machine-Learning-Comparison

References

[1] Caruana, R. (n.d.). An Empirical Comparison of Supervised Learning Algorithms. https://www.cs.cornell.edu/caruana/ctp/ct.papers/caruana.icml06.pdf

[2] Mushroom. (1987). UCI Machine Learning Repository. https://doi.org/10.24432/C5959T.

[3] Music features. (n.d.). www.kaggle.com. https://www.kaggle.com/datasets/insiyeah/musicfeatures/data

[4] Online Chess Games. (n.d.). www.kaggle.com. https://www.kaggle.com/datasets/ulrikthygepedersen/online-chess-games/data

[5] Oranges vs. Grapefruit. (n.d.).www.kaggle.com. https:// www.kaggle.com/datasets/joshmcadams/oranges-vs-grapefruit