An Analysis of Six Supervised Machine Learning Approaches on Four Real-World Data Sets

Josh Dawson, Mario Stinson-Maas

May 14th, 2024

Abstract

We discuss how a range of supervised machine learning models perform on four different data sets. We consider both regression and classification tasks and investigate how the data set, model, and task all determine the resulting performance. We employed linear regression, support vector machines, k-nearest neighbors, decision trees, random forests, and neural networks. We find that neural networks statistically outperform other supervised learning approaches on all data sets, while linear regression underperforms other supervised learning approaches in all cases we came across. Most of the other approaches perform similarly across data sets, although support vector machines rival neural networks in regression tasks.

1 Introduction

In the past several decades, the world we live in has become increasingly dependent on automation. The ability of machines to automate tasks has allowed for less error, and has greatly reduced the amount of necessary human labor. Machine learning is the next rung in this automation ladder, where models are able to learn and improve their performance on a task over time. With machine learning, the amount of tasks that can be automated has never been higher, but before we fully integrate such a new tool, there are many important questions to answer.

In the context of this paper, we are interested in understanding how different supervised machine learning approaches compare through the use of real-world data sets. We are interested in this comparison across both classification and regression tasks. By learning how these approaches compare across real-world data sets, we are able to identify when one solution is better than another.

To do this, we decided to choose four different data sets and apply six popular supervised machine learning approaches to them. We selected data sets with both classification and regression tasks and approaches that handle both of these cases. Then, we compare the results to identify any trends or statistically significant performance differences of certain models and data sets.

We ran four models on classification tasks, and three models on regression tasks. We calculated the performances and generated plots comparing them, and found that many models performed similarly on the classification data sets we used; however, neural networks did better than most models, while linear regression models did worse than most.

2 Identified Problem

Throughout this semester, we focused largely on supervised machine learning and encountered several approaches to supervised machine learning problems. As mentioned above, we are interested in comparing these approaches through the use of real-world data sets. In this case, we have four different real-world data sets, and we want to learn how to automatically make predictions for new instances of our real-world data. To understand how the performances compare, and ensure that our predictions are logical, it is important to understand what data these data sets contain and how it is organized. This information will also inform how our results are interpreted. These four data sets are: MNIST1000, Penguins, Energy, and Seoulbike.

MNIST1000 is a data set made available by the Modified National Institute of Standards and Technology database. It contains optical character recognition of numeric digits; each instance is a handwritten digit, and each attribute is intensity values for each pixel. This data set contains 1,000 instances of each written digit sampled randomly from the original training data, resulting in 10,000 total rows. The label we are trying to predict is the written digit (0-9), so the task is classification. Labels in this dataset are evenly distributed, meaning there are exactly 1000 instances of each possible label.

Penguins is a data set that comes from the University of California Santa Barbara. It contains several body measurements of different penguins belonging to three different species. Here, each row is the measurements and label for an individual penguin. The label we are trying to predict (the species) is categorical – Adelie, Gentoo, or Chinstrap – and so this data set represents a classification task as well. Seen in 1, Adelie is the most frequent label in this dataset, followed closely by Gentoo with Chinstrap far behind.

Label	Count
Adelie	151
Gentoo	123
Chinstrap	68

Table 1: Label Distribution for Penguins

Energy is a data set that comes from the UCI Machine Learning Repository, detailing the energy consumption in 10-minute increments by appliances in a Belgian residence. Each row is an appliance, and the attributes are continuous numeric measurements. The label we are trying to predict is also continuous, and it represents the total amount of energy consumed by a given appliance. This makes it a regression task. As seen in 1, there is a heavy right skew in the distribution of labels, with 30-60 watt-hours being the most frequent label, by far.

Seoulbike is another data set from the UCI Machine Learning Repository, and it describes bike rentals in Seoul. The attributes are a mix of categorical and numeric, and they include information such as season and weather data. Each row contains information for the attributes over one hour, and the label we are trying to predict is the number of bikes that will be rented that hour. This is a continuous label, and so this is also a regression task. As seen in 2, it was common to have a several hundred bike rentals an hour, and becomes much less common the more hourly bike rentals there are.

A table of summary statistics for the aforementioned data sets can be seen below [2]. It should be noted how largely the data sets vary, and how regression and classification data sets differ. This

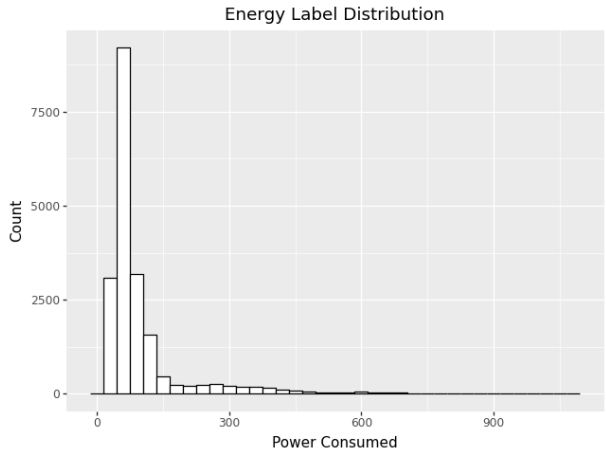

Figure 1: Distribution of observed power usage, in watt-hours, among ≈20,000 observed appliances in the Energy data set. Each bin has width 30 watt-hours.

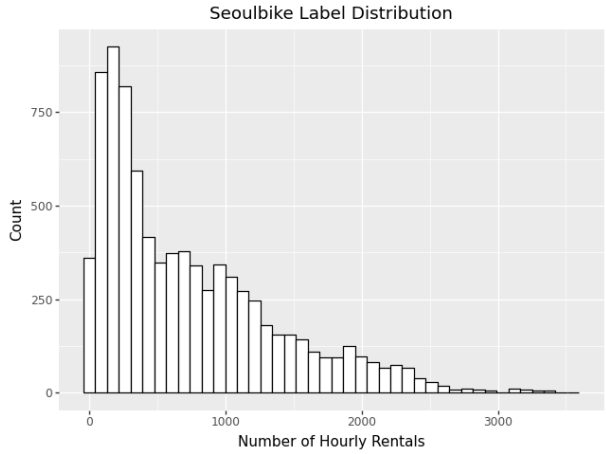

Figure 2: Distribution of observed bike rentals per hour, among ≈8,500 observed hours in the Seoulbike data set. Bins have width 95.

variance in data sets suggests that the machine learning approaches employed on them need to deal with a large range of labels and both small and large amounts of data.

Data Set	Attributes	Rows	(Range of) Possible Labels
MNIST	785	10000	10
Penguins	5	342	3
Energy	28	19735	10–1080
Seoulbike	12	8465	2–3556

Table 2: Summary Statistics for each Data Set. Since Energy and Seoulbike are regression task data sets, they have a range of possible labels, from the minimum to the maximum label encountered.

3 Proposed Solution

To make successful predictions about the labels of new instances of our data sets, we apply six supervised machine learning approaches. From these six approaches, we have six solutions. These approaches are k-nearest neighbors, decision trees, random forests, linear regression, support vector machines, and neural networks. Not all models are able to be used on all data sets, and in fact, most models only work on either classification or regression, since they are such different tasks. Some models also require the data to be preprocessed, so that they most optimally predict a label.

To start, let us cover k-nearest neighbors (k-NN). In k-nearest neighbors, k is a hyperparameter that can be selected, dictating how many nearest neighbors we consider. The underlying model is quite simple. We have some metric that allows us to consider the distance between attributes. We have a Euclidean distance metric for ordinal attributes and a Hammond metric for nominal ones, but the algorithm requires all attributes in the data set be either nominal or ordinal. From this distance metric, the k nearest neighbors are determined, and the prediction for the label is the majority label of the k nearest neighbors. To increase performance, we use min-max normalization on the attributes, this offsets scale differences between them, making it less likely for a large scale attribute to throw off the distance. Here, we only k-nearest neighbors for classification tasks.

Next, we will discuss decision trees. Decision trees are exactly what they sound like, trees created from branching decisions. These trees work by partitioning attributes within certain bounds. The algorithm is able to determine the best attribute to start with, by finding the attribute that is most impactful in determining the label. New branches are made until only one label is left, and the conditions based on this label can be read by following the branches leaf upto the root. Decision trees work quickly, since there are not a lot of calculations that need to be made. Here, we only use decision trees for classification tasks.

Forests exist as a generalization of decision trees, where we have many trees in the same model that work to determine a label. When creating multiple trees, we manipulate the training sets, and sometimes consider fewer attributes. This means there will be less correlation between the trees, which reduces the variance of our predictions. In a forest we have a specified amount of trees, and the output of the forest is the majority label of the trees, similar to how the label is selected in k-NN. Forests can be used for regression and classification tasks, but here we only use them for classification.

Like the name implies, linear regression is a regression model that considers a linear fit. Quite simply, this approach determines how to draw a straight line to best fit the data based on the attributes provided. The equation will be of the form

$$\hat{y} = \omega_0 + \sum_{i=1}^{d} \omega_i x_i,$$

where our predicted label, $\hat{y}$, is equal to an intercept, ω_0 added to the sum of all weights and attributes. Linear regression determines the best set of weights to most accurately predict the label. To use linear regression, we have to perform one-hot encoding, where all categorical attributes are transformed into binary variables. Min-max normalization can also be used to optimize performance.

Support vector machines (SVMs) are another regression model. A big limit of linear regression is that it can only learn linear trends. There is an inherent assumption when using linear regression that there are no higher order relationships between attributes and labels. SVMs allow us to

overcome this by transforming the data to a new space using a kernel function. This kernel specifies how to replace the attributes with new combinations so that more complex relationships between attributes and labels can be learned. The kernel function is a hyperparameter specified by the user. Like linear regressors, SVMs need one-hot encoding and benefit from min-max normalization.

Neural nets (NN) are a bunch of logistic regressors (similar to linear regressors but with an activation function) attached together that undergo an algorithm called backpropagation. Each logistic regressor is called a neuron, and groups of neurons are organized in layers. The layers feed into each other to propagate information, and since each neuron will learn a trend/pattern, following the layers left to right represents a level of abstraction in learning. This allows neural nets to learn complex trends. Although NNs are fantastic models, they require more preprocessing to be used. Just like other regression models, they need one-hot encoding and benefit from min-max normalization. Unlike other regression models, the labels need to be converted to numbers to predict classification tasks. The number of neurons per layer and the rate of learning (used for backpropagation) are both hyperparameters chosen by the user. Neural nets are the only model we used for both classification and regression tasks.

Since our solutions have been outlined, we may now discuss how they were implemented.

4 Experimental Design

For each data set, we began by preprocessing the data; this consists of performing min-max normalization, one-hot encoding, and splitting the data into testing and training sets (along with converting labels into integers for our classification sets). We performed this using a random seed of 12345 and training percentage of 75.

We used scikit-learn to implement all of our solutions except for neural networks, for which we used tensorflow. Using our performance charts from HW4, we chose our optimal parameters (see [3]), with $k = 1$ and standard Euclidean distance for each k-NN. Each random forest had 100 trees, and we used a kernel of degree 4 for our SVM models.

Data Set	Neurons	Learning Rate
Penguins	16	0.1
MNIST1000	64	0.01
Energy	64	0.01
Seoulbike	64	0.1

Table 3: Hyperparameters for Neural Networks

Once we trained all of these machines, we used them to make predictions, and then calculate performance. Making predictions on our testing data set is straightforward, as we can create predictions using scikit-learn, from one call to our models. However, we need to find a way to calculate performance in order to classify these models and answer our research question. For our classification data sets (Penguins and MNIST1000), we used the following formula, where y is the testing set labels, $\hat{y}$ is the predicted labels, and $|y| = n$:

$$\text{accuracy} = \frac{1}{n}\sum_{i=0}^{n}(1 \text{ if } y_i = \hat{y}_i, \text{ else } 0) \ .$$

For our regression data sets (Energy and Seoulbike), we instead calculated mean average error (MAE). To do this, we used scikit-learn's `metrics` module, which has a built-in function that returns the MAE of a given set of predictions.

After training our models, making predictions, and measuring performances, we still had to calculate confidence intervals and present the information in a cohesive manner. Since we are comparing more than two models at a time, we also need to implement Bonferroni correction. For comparing the 4 classification models, this results in the following formula for an accuracy confidence interval, where p is accuracy and n is the number of testing instances:

$$p \pm 2.39 \cdot \sqrt{\frac{p \cdot (1-p)}{n}} \ .$$

For comparing our 3 regression models, we get the following formula for MAE confidence intervals:

$$MAE \pm 2.24 \sqrt{\frac{V_{MAE}}{n}} \ ,$$

where MAE is mean average error, $V_{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left(|y_i - \hat{f}_i| - MAE \right)^2$ is the variance in the absolute error of predictions made on test set y.

5 Discussion of Results

5.1 Penguins

(See 3). All the models here performed exceptionally well – each model's confidence interval fell entirely above 90% accuracy. Compared to the null model (predicting Adelie every time), which has accuracy of 44.1%, we see a remarkably accurate set of models. Although the decision tree appears to perform the worst, we note that there is no statistically significant difference in performance between any of these models. One reason this might be is that there are only 5 attributes, so the correlations between the attributes and species of penguin are relatively simple; this means that the underlying patterns can be modeled by several machine learning solutions with high accuracy. Moreover, if the underlying relations are simplistic, then the way that the relations are modeled might be similar between models.

5.2 MNIST1000

(See 4). When testing on MNIST1000, we saw nearly identical performance within K-nearest neighbors, neural networks, and random forests. However, decision trees performed statistically significantly worse. Since random forests perform so much better here than a single tree, we believe that the tree is overfitting; since there are nearly 800 attributes to be considered, the decision tree likely struggled to find an optimal order of attributes to look at. On the other hand, the random forest was able to solve this feature selection problem by learning from the results of 100 trees.

We note that the upper bound of our confidence interval for calculated accuracies falls just below 95%. It makes sense that this data set has a lower accuracy than the Penguins data set; not only are there many more attributes to consider here, but there is also a lot more natural variance within classes of labels. For example, many handwritten 1s may appear to be 7s through natural differences in handwriting styles – however, there are genetic limitations that result in limited variance between penguins of the same species.

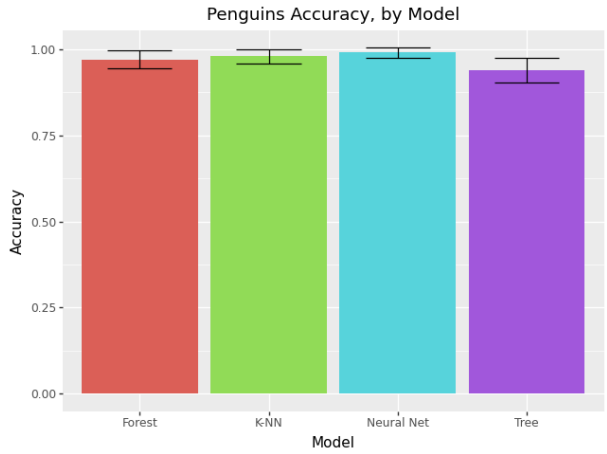

Figure 3: Accuracies and confidence intervals for models trained on the Penguins data set. There are no statistically significant differences in model performance.

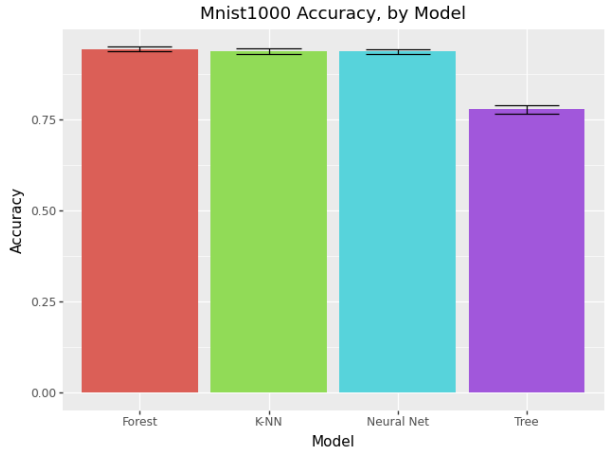

Figure 4: Accuracies and confidence intervals for models trained on the MNIST1000 data set.

5.3 Energy

(See 5). When looking at the mean average errors over the Energy data set, we notice that each model performs statistically differently from the others. The linear model performs the worst, with the neural net seeing slightly better MAE. However, the SVM model outperforms both of these significantly. It makes sense that the linear model performs the worst because it is the most simple way of modeling our data, and we don't believe that the data's underlying relations are all linear. However, we aren't sure why the SVM performs so much better than the neural on this dataset; further research would be needed, as outlined in our concluding section.

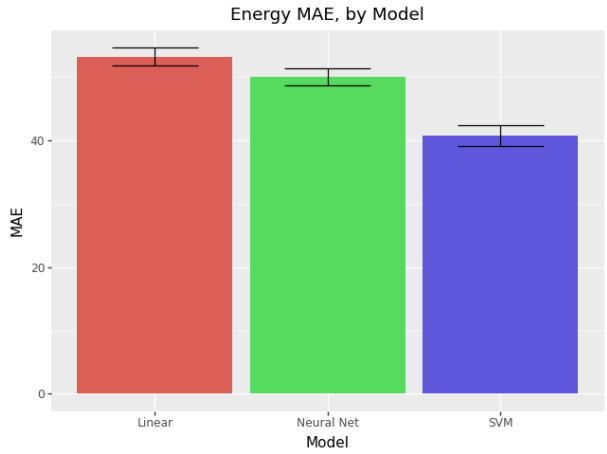

Figure 5: Accuracies and confidence intervals for models trained on the Energy data set.

5.4 Seoulbike

(See 6). Here again we see a statistically significant difference between each model. Linear performs the worst, while the SVM does slightly better and the neural net performs drastically better than both of the others. This makes sense, because neural nets are capable of abstracting trends in the data with high precision. This data set seems to confirm our main takeaway, that linear models perform much worse than other models, while neural nets perform much better.

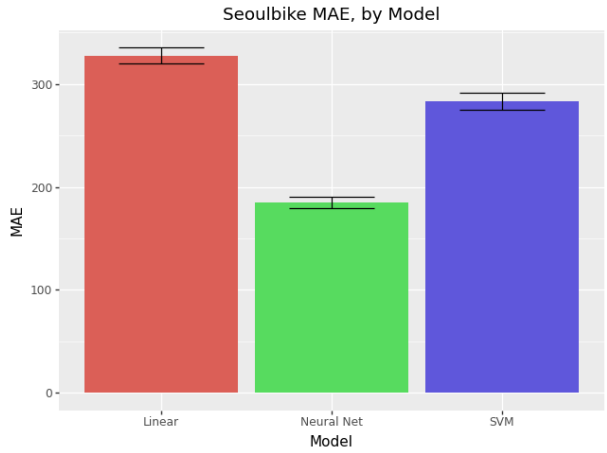

Figure 6: Accuracies and confidence intervals for models trained on the Seoulbike data set.

5.5 Summary

(See 4). Overall, we observed the best performance when using neural networks; they performed better or as well as all models except for SVM on the Energy data set. Since they are applicable to both classification and regression tasks, we believe that they are the best choice for solving supervised learning tasks. However, we take care to note that there cannot be a "best model"

because every data set is structured differently, and contains different underlying relations. Since the SVM model with kernel of degree 4 performed similarly well against other models and was only outperformed once, we believe that it also should be considered as an optimal solution to many regression tasks.

We also observed that linear regression statistically underperformed every model it was tested against; this seems to be one of the worst supervised learning approaches, although it could potentially be useful as a baseline for understanding other models' performance.

Model	Outperformed	Underperformed	Neither
Linear	0	4	0
SVM	3	1	0
K-NN	1	0	5
Tree	0	3	3
Forest	3	0	3
NN	4	1	5

Table 4: Model performance. Describes how many times a model statistically significantly outperformed, underperformed, or performed similarly to another model.

6 Conclusions and Future Work

In order to find out which supervised machine learning approaches compare across real-world data sets, we devised 6 solutions to test on 4 data sets. After training these 6 types of models, generating predictions, and calculating performances, we discovered that neural networks statistically *outperformed* other models the most frequently, while linear regression models statistically *underperformed* other models the most frequently. Moreover, SVM models performed almost as well as neural networks on regression tasks. Thus, when faced with a supervised learning task, we conclude that neural networks are the best approach to take; when the task is regression, SVM models are also worth exploring, as they are capable of outperforming neural networks in some cases.

In order to expand on this research, we would repeat our experiment with many more datasets; specifically, we are interested in examining regression tasks. Since we recorded an instance of neural networks underperforming SVMs, we would test out data sets of different dimensions with the hopes of drawing conclusions about when one might opt to deploy one model over the other. We also understand that neural nets have many more hyperparameters that we didn't alter, such as number of hidden neurons/layers; a follow-up experiment might test several variations of these hyperparameters in order to determine the most optimal network, and then test this against an SVM.