CS 7641 - Supervised Learning Report

Yijun Xie (yxie400)

February 12, 2023

1 Datasets

1.1 MAGIC Gamma Telescope Dataset

1.1.1 Description

MAGIC dataset contains synthetic registrations of high energy gamma particles in a ground-based telescope. When photons passes the telescope, some of them will leave a mark in the detector. We can collect these marks and conduct a principle component analysis to collect features of the distribution of these particles. The goal is determine whether these marks come from a primary gamma (signal) or hadron (background).

1.1.2 Why I find it interesting?

One of my childhood dream was becoming a physicist, and I majored in physics in my freshman year. Although later I switch to another major, physics still means a lot to me. I cannot recall anything I learned in my Modern Physics class, but I would still love to work on something related to physics.

The dataset itself is also interesting. To start, it was generated by Monte Carlo, and all features are continuous numerical. It contains near 200K observations but only 10 features. It almost synthesize a perfect situation for data analysis. On the other hand, by nature of the underlying problem, the cost of false positive outweighs the cost of false negative, which poses challenges when designing loss functions.

1.2 Statlog (German Credit Data) Dataset

1.2.1 Description

The Statlog data contains information of 1000 credit card clients and the goal is to decide whether the client is of good or back credit risk. This is a classic machine learning application and has been widely adopted in industry. I believe it serves well as a complimentary of the MAGIC dataset.

1.2.2 Why I find it interesting?

I will be working on a very similar project where we need to assign labels to users based on collected features. This dataset can be used as a rehearsal.

More importantly, this dataset is the opposite of MAGIC dataset. It is significantly smaller (1K observations) but has more features. A lot of its features are ordinal or categorical, and some might be irrelevant, which makes the data processing more challenging.

2 Methodology

In order to facilitate a fair comparison across different machine learning models, we adopt the following methodology.

For the choice of metrics, while in the documentation of MAGIC data the owner suggests using AUC as a measure of model performance, such metric is not feasible for models that cannot directly generate probabilistic predictions, such as KNN or SVM. We could, in principle, find a route to convert the class prediction outcome to probability. However, that would be out of the scope of this report. Therefore, we stick with an F-score to handle class imbalance. For example, in the MAGIC dataset, the cost of classifying a non-signal (0) as a signal (1) is smaller than classifying a signal (1) as non-signal (0). Similarly, in German Credit data, the cost of classifying a bad credit (0) as good (1) is greater than classifying good (1) as bad (0). We therefore use F-10 for MAGIC dataset, and F-0.2 for German Credit data.

For all models but Neural Network, we split the data into training and testing set. The fraction of training set varies from 0.1 to 0.9. We apply model specific hyper-parameter tuning on the training set via a 5-fold cross-validation, and report the model performance on testing set.

For Neural Network, we using 90% of data as training set and the remaining as testing set. We iterate the number of epochs from 10 to 200.

3 Decision Tree

For MAGIC dataset, we tune the maximum depth allowed for the tree, and the learning curves for the best available are shown in Fig.1. We can tell from the plot that The training score is horizontal against different training data size. Furthermore, the testing score, while fluctuated a little bit more, also show this horizontal pattern. The testing score is also very close to the training score. These are signs of a "saturated" training, i.e. the model has already done a great job even with a very small sample of data, and adding more to the training set does not make a big difference. This is likely due to the fact that the data came from Monte Carlo simulation, and hence it is too clean compared with real world dataset.

Figure 1: Learning Curve of Decision Tree on MAGIC Dataset.

For German Credit dataset, we start with only tuning maximum depth, and the learning curve is shown in Fig.2. We notice that the training score reached the perfect score several times, and there is a large discrepancy between training and testing score. It suggest the model may suffer from over fitting, and we need to prune the tree more. We further prune the tree by tuning minimum samples for a split, and minimum samples in each leaf node along with maximum depth. The learning curve for the best available

Figure 2: Learning Curve of Decision Tree on German Credit Dataset. Only maximum depth is tuned.

model is presented in Fig.3.

4 KNN

For K Nearest Neighbor model, we tuned two parameters: The number of neighbors K varies from 3 to 10, and three different tree algorithms, namely *kdtree*, *brutetree*, and *balltree*.

In Fig.4 I present the learning curve of the best selected model on MAGIC dataset. The performance is evaluated using Area Under Curve (AUC). Compared with decision tree, we see much smoother learning curves, and the model performance increases when the fraction of training set increases.

Also compared with decision tree, we notice that the model performance is consistently lower. One possible reason is that we use Euclidean distance on raw data, however, the scale or impact of each feature might not be the same. For example, it is indifferent for decision tree to make a split on a feature scale from 0 to 1 and another feature scale from 0 to 100. But

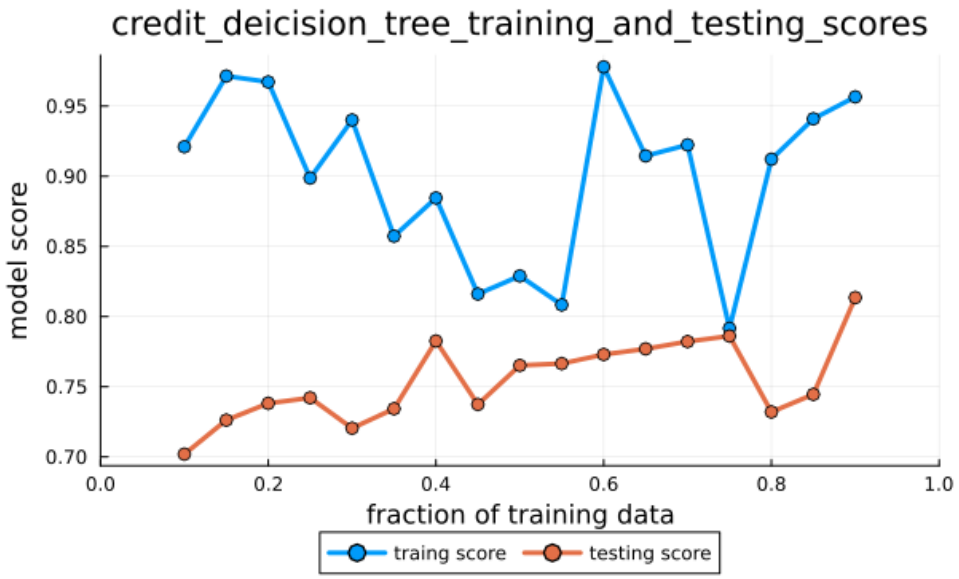

Figure 3: Learning Curve of Decision Tree on German Credit Dataset. Maximum depth, minimum samples for a split and in each leaf are tuned.

for KNN, the latter is always preferred. Another possible reason is that the data might contain parts where both positive and negative samples are well mixed. For KNN, it is quite sensitive to local noises, and thus the model performance are bad in these part, which further affects the overall performance. On the other hand, decision tree can be more robust and have better tolerance on local noises.

KNN also performs worse on German Credit dataset. This is less surprising because of the two reasons state above. Additionally, the dimension of German Credit data is significantly higher than MAGIC data, especially consider the feature to sample size ratio. In a high dimension space, every point is far away from each other, and for distance based algorithm such as KNN, it makes less sense to compute the vanila Euclidean distance.

5 Boosting

We trained XGBoost classifiers to tackle these two problems. More specifically, we tuned the maximum depth allowed (3 to 10) and the minimum

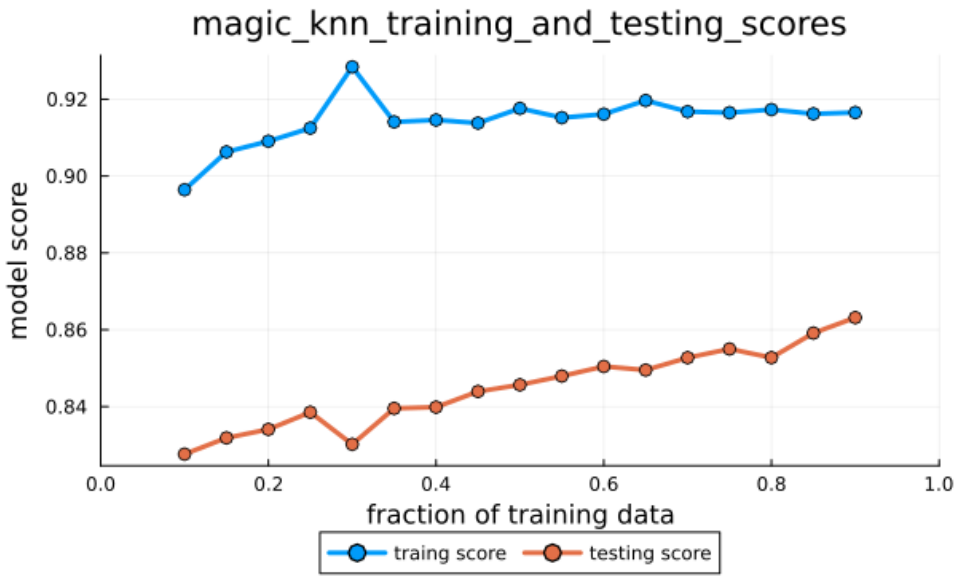

Figure 4: Learning Curve of KNN on MAGIC Dataset.

child weight (0 to 5).

XGBoost provides us the best performance among all models, and that partially explains its prevalence in industrial applications. For both MAGIC (Fig.6) and German Credit (Fig.7) data, the training scores hover around 1, which means it can be fully trained even with a very small sample size. However, that dose not necessarily translate to overfitting. If we focus on the testing score, we can tell that it outperforms other models as well by a significant amount. The increasing pattern in testing score suggests that XGBoost benefits from a large sample size, and it does not suffer from high dimensionality like KNN.

6 Neural Network

From the above models we learn that in principle the model performance is proportional to the fraction of training dataset. Therefore, for neural networks, we fix the training fraction to be 0.9 and evaluate the model performance against the number of epochs. We tuned the batch sizes from 1 (stochastic gradient descent) to 500, and various strength of regulaty.

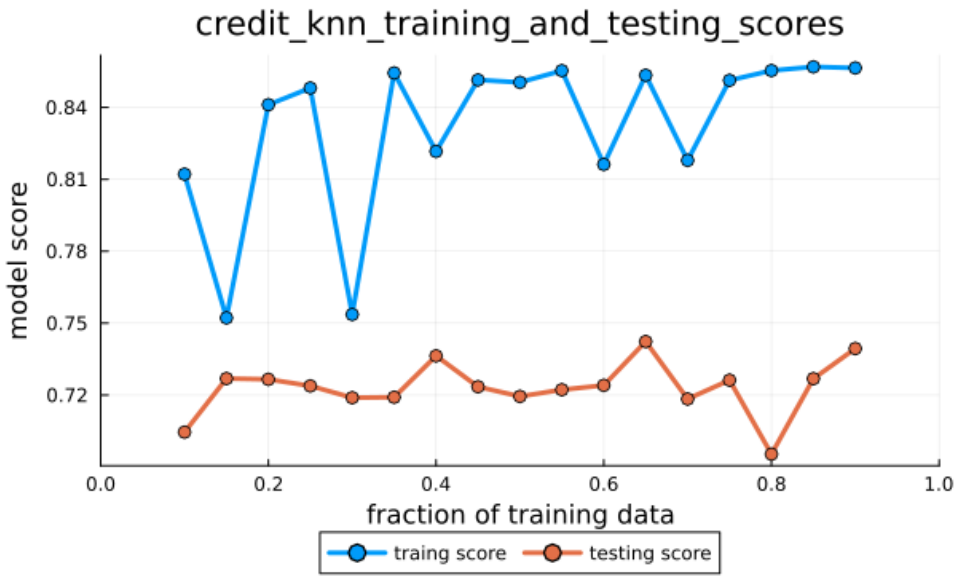

Figure 5: Learning Curve of KNN on German Credit Dataset.

From Fig.8 we can tell that the neural network for MAGIC data is fully trained after 50 epochs, and the eventual performance is still worse than XGBoost. Similarly in Fig.9 we observe the exact same pattern for German Credit data.

The possible explanation could be that we use a relatively simple architecture, i.e. 2 hidden layers, with 16 and 8 nodes respectively. This neural network is not deep and hence cannot fully reveal the potential of neural network. However, for a relatively simple application such as our 2 datasets, building a too complicated model is not always cost-efficient.

7 Support Vector Machine

Unlike models discussed above, SVM does not provide probabilistic inference. That is, the output is either 0 or 1, instead of a probability of being 1. While there are papers proposed methods for estimating the probability, they are not well adopted and hence not considered in this report. The consequence is that for MAGIC data we cannot evaluate model performance using AUC. In these scenarios, I choose F-10 score as alternative to AUC to

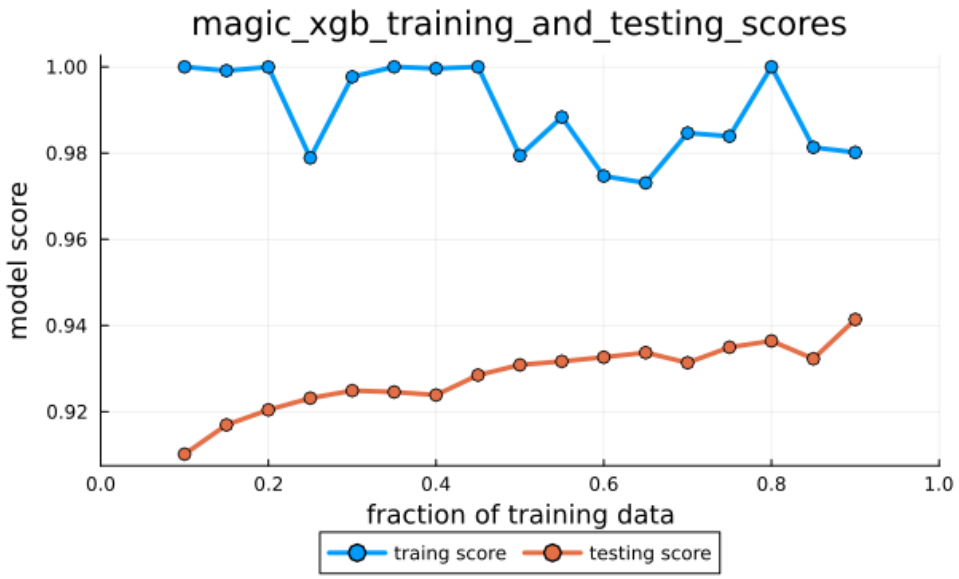

Figure 6: Learning Curve of XGBoost on MAGIC Dataset.

compensate for imbalanced classes.

As shown in Fig.10 and Fig.11, the SVM does not work particularly well for our 2 examples. The model is clearly over-fitted since the score is close to 1, possibly because of the imbalanced data. And the large discrepancy between training and testing scores suggest that the autual model performance is not satisfying.

My conjecture is that:

1. The sample size is too large for SVM. Even though it's still relatively small compared to modern applications, the sample size is still too large for SVM to handle.

2. The dimension is too high, and some overlapping between different labels is inevitable. This leads to the underperformance of SVM.

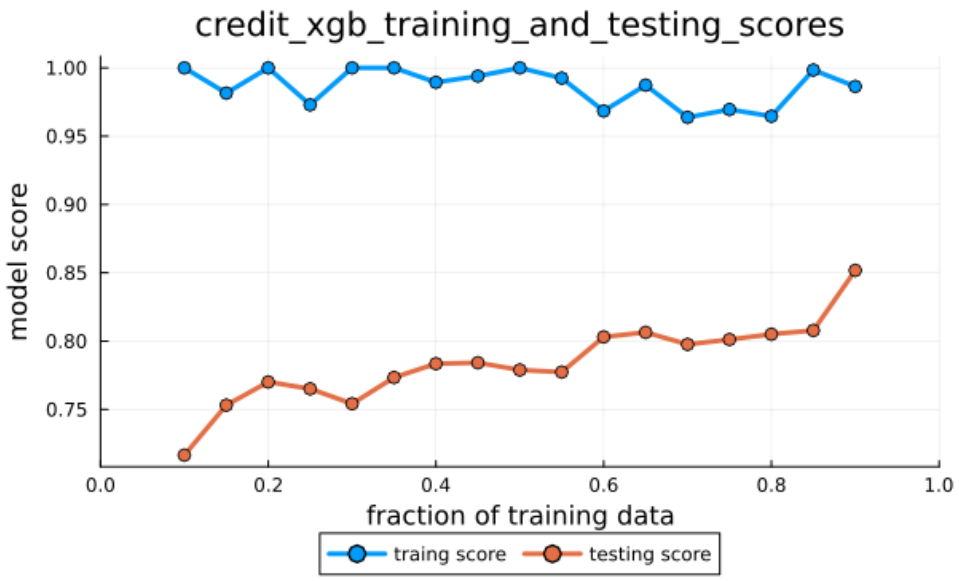

Figure 7: Learning Curve of XGBoost on German Credit Dataset.

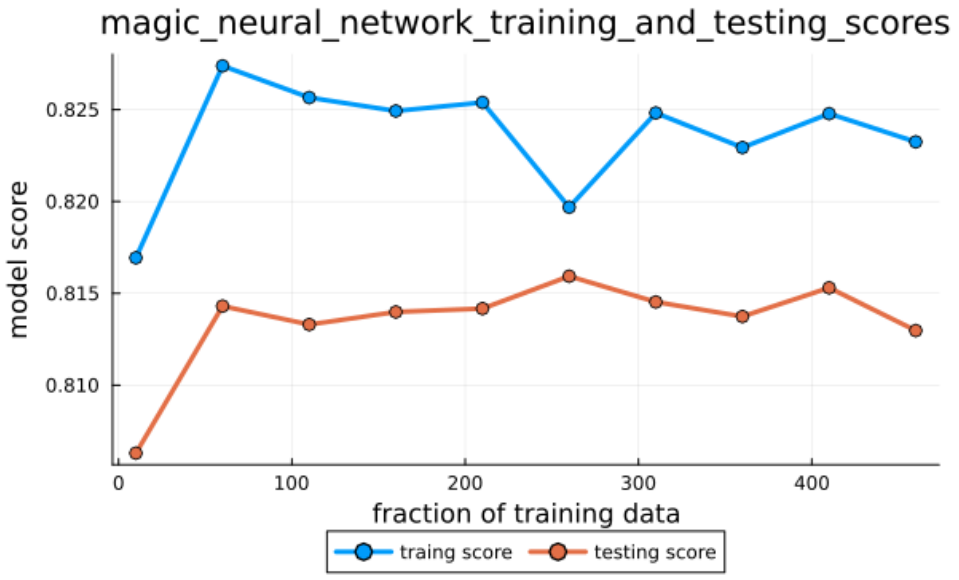

Figure 8: Learning Curve of Neural Network on MAGIC Dataset.

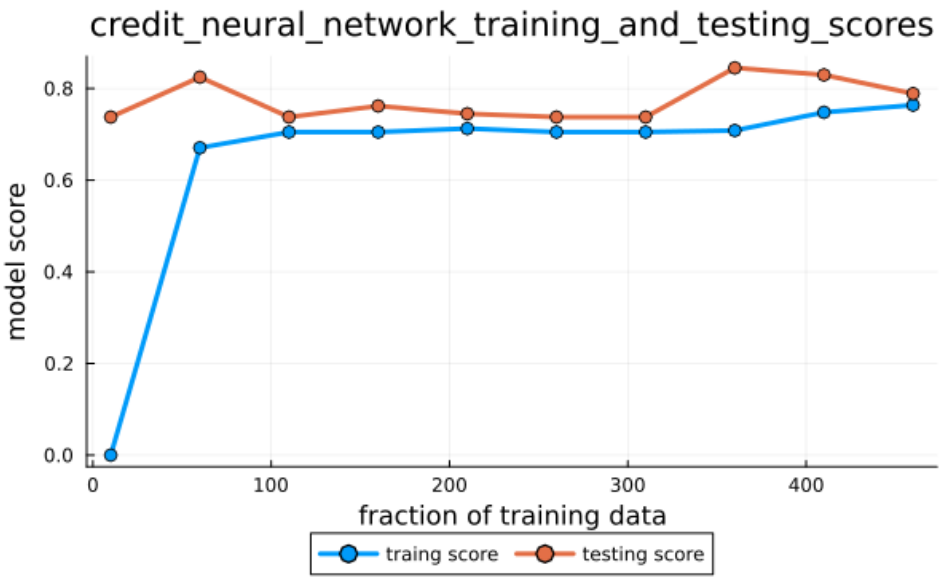

Figure 9: Learning Curve of Neural Network on German Credit Dataset.

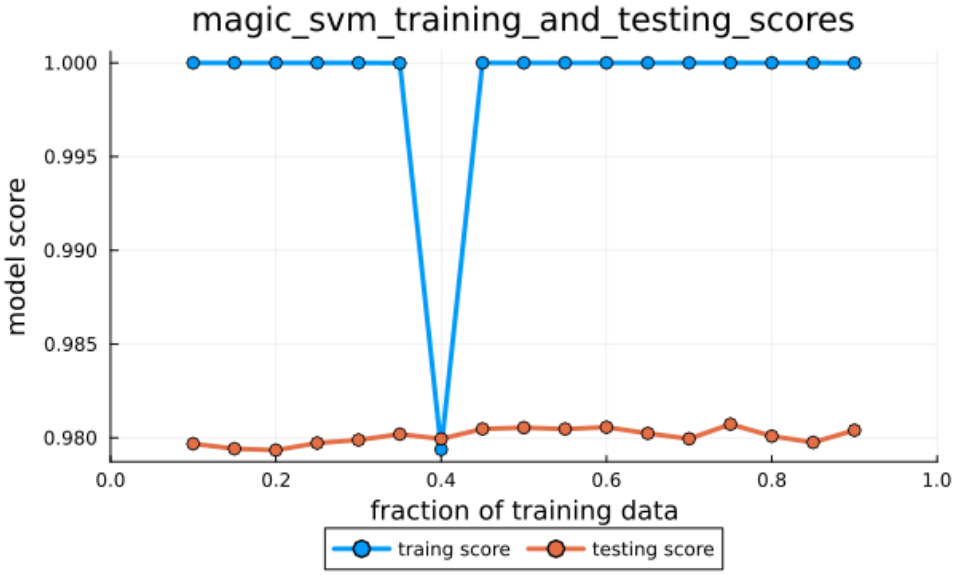

Figure 10: Learning Curve of SVM on MAGIC Dataset.

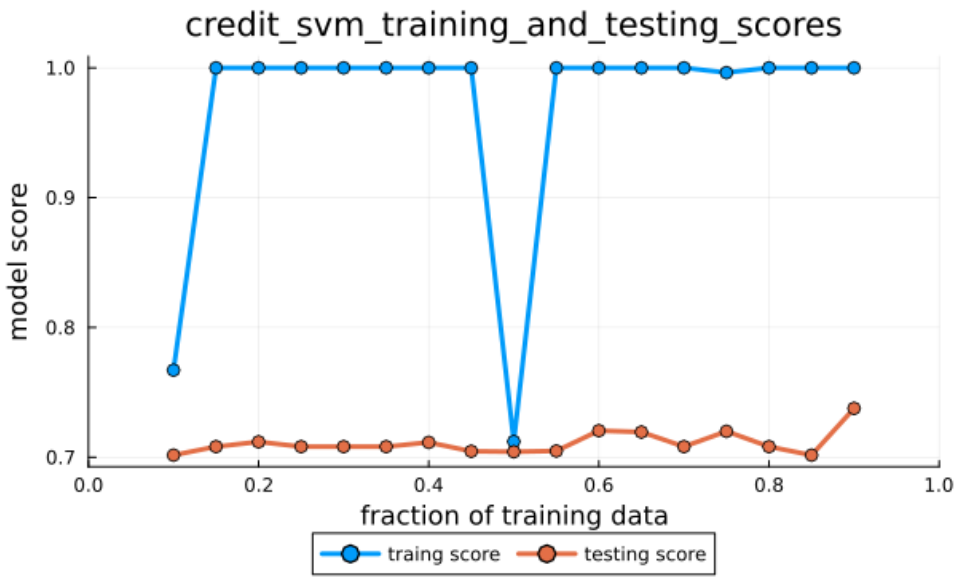

Figure 11: Learning Curve of SVM on German Credit Dataset.