CS 7641 Assignment 1

Yijun Xie

I. Introduction

In this report, we apply several common classification methods on two datasets, namely the `German Credit Data` [1] and `Bank Marketing` [2] from UCI data repository. We choose these 2 datasets because they pose challenges in handling small sample sizes and non-accuracy loss functions.

The primary objective of this study is to empirically evaluate the performance of different classification algorithms on these datasets. More specifically, we focus on the impact of encoding methods, training size, and hyper-parameter tuning. The model performance is measured by both the in-sample cross-validation and the out-of-sample testing data.

We hypothesize that encoding methods significantly affect model performance and that XGBoost will outperform other models due to its robustness and flexibility. The results from our experiments provide insights into the strengths and weaknesses of each algorithm, guiding future applications in similar contexts.

Our experiments verify the above hypotheses, suggesting that a carefully chosen encoder paired with a more flexible model that can handle non-linearity will perform better in such classification tasks.

II. Datasets

We consider the following 2 datasets: `German Credit Data` [1] and `Bank Marketing` [2] from UCI data repository.

These 2 datasets share many similarities:

- Their targets are both binary.
- Their covariates include both numerical and categorical variables.

However, these 2 datasets are challenging in different aspects:

A. Class Balance

- In the German Credit dataset, the target variable is relatively balanced (30% `good` and 70% `bad`).
- The Bank Marketing dataset suffers more from an imbalanced target variable. Only 11.7% of the targets are classified as 2 while 88.3% of the targets are classified as 1. The imbalanced target will lead to more challenges in choosing metrics as discussed below.

B. Loss Functions

- The German Credit dataset has a specified loss function, i.e., the cost of classifying an object with `bad` label into `good` is 5 times as high as classifying an object with `good` label into `bad`. This means we need a customized loss function when training and evaluating the model performance.
- The Bank Marketing data, on the other hand, does not have a specified loss function. However, due to the class imbalance discussed above, we need to carefully consider what each metric implies in practice. From a marketing perspective, the cost of targeting a customer is relatively low with modern techniques (e.g., emails, in-app pop-ups, website banners). Hence, we should focus more on trying to reach out to as many potential customers as possible. With that assumption, we choose **Recall** as our metric, which measures how many positives are identified among all true positives.

C. Feature-to-sample size ratio

- For the German Credit data, there are 12 features from 1000 observations. This makes the feature-to-sample size ratio less than 10, which might pose some challenges in model fitting.
- On the other hand, in the Bank Marketing dataset, we have 17 features for 45K observations, and hence suffer less from the high dimensionality issue.

III. Experiment Setup

For each of the datasets, we consider the following 4 possible models: K Nearest Neighbors (KNN), Support Vector Machine (SVM), Neural Network (NN), and XGBoost. For each of the candidate models, we conduct the following experiments to mimic stages of exploring and evaluating models.

A. Hypotheses

In our report, we make the following hypotheses:

- Encoding methods will significantly affect the model performance, because in both datasets there are a lot of categorical variables, and different encoding methods will significantly affect how the model intakes these categorical variables.
- XGBoost model will be the best performing model, because it is flexible due to the tree-based weak learners, fast to train, and is robust against different sets of hyper-parameters.

B. Choice of Metrics

For German Credit data, we use the **unnormalized** weighted misclassification loss as our metric. That is, a false positive will receive a loss of 5 while a false negative will receive a loss of 1. The lower the misclassification loss is, the better the model performance is.

For Bank Marketing data, we use recall as our metric. We measure the proportion of qualified clients correctly labeled by the classification model. The higher the recall rate is, the better the model performance is.

C. Encoding Categorical Variables

In both datasets, we have categorical variables that need to be pre-processed before feeding them to the models. Here we consider 2 commonly used encoders: one-hot encoder and label encoder. The former creates k-1 binary variables for a categorical variable with k classes. The latter encodes the k classes with a numerical value from 1 to k.

D. Training Size Test

In the first step, we use the off-the-shelf setting of these models to get a baseline performance based on cross-validation. We measure the baseline performance against different training sizes and wish to understand how sensitive the model performance is to training size. We also want to evaluate whether the performance is saturated with the available data size, i.e., whether the available data is large enough for the underlying model to fit well. The result of this step is presented in the learning curves shown below.

E. Hyper-parameter Tuning

In the second step, we start evaluating the model performance based on cross-validation with different hyper-parameters. We want to understand how the hyper-parameters affect the model performance, and more importantly, how sensitive the candidate model is to different hyper-parameters. This step will help us to plan on searching for the best available model within each model family.

In this report, we present the validation curves of following hyper-parameters for each model:

- KNN: Number of neighbors ranging from 1 to 30.
- NN: Maximum number of iterations from the following values: 50, 100, 200, 300, 400, 500.
- SVM: Strength of regularization ranging from 10^{-4} to 10.
- XGBoost: The maximum depth of the fitted tree in each weak learner ranging from 1 to 10.

A more thorough hyper-parameter tuning results are presented in the next section.

F. Grid Search and Validation

The above steps help us to have an initial understanding of the model performance based on cross-validation on the training data with a single hyper-parameter. In the third step, we would like to: a) explore the model's best possible performance in a larger hyper-parameter space, and b) validate model performance on reserved test data.

More specifically, we consider the following hyper-parameter space for each model:

1) KNN:

- Number of neighbors: odd integers from 1 to 30.
- Weight function for neighbors: uniform or inverse of distance.
- Metrics for measuring the distance: Euclidean, Manhattan, Chebyshev, Minkowski.

2) Neural Network:

- Hidden layer sizes: One layer(128), deep neural network (32, 32, 32, 32, 32, 32, 32), wide neural network (1024, 64), deep and wide neural network (256, 128, 64, 32, 8, 8).
- Activation functions: ReLU, Tanh.
- Solvers: Adam, SGD.
- Alpha (L2 penalty): 0.0001, 0.001, 0.01.
- Learning rate: constant, adaptive.
- Maximum number of iterations: 50, 100, 200, 300, 400, 500.

3) Support Vector Machine:

- Regularization parameter C: from 10^{-4} to 10^{1} on a logarithmic scale.
- Kernel types: linear, RBF, polynomial, sigmoid.
- Degree (for polynomial kernel): from 1 to 4.
- Gamma (for RBF, polynomial, and sigmoid kernels): scale, auto.

4) XGBoost:

- Number of estimators: from 50 to 200.
- Maximum depth of trees: from 1 to 10.
- Learning rate: from 0.01 to 0.21.
- Subsample ratio of the training instances: from 0.5 to 1.
- Subsample ratio of columns when constructing each tree (colsample_bytree): from 0.5 to 1.
- Minimum loss reduction required to make a further partition on a leaf node of the tree (gamma): from 0 to 0.5.

IV. Results

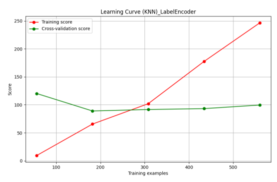
A. German Credit Data

1) Training Size: Below are the performance comparisons against different models and training sizes for the German Credit data. From the above plots, we can tell that:

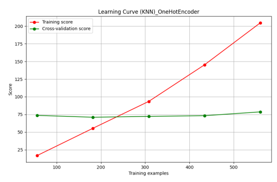
- The overall best model is the support vector machine with one-hot encoder.
- The uplift in model performance with a larger training size is marginal when the training size is greater than 200. It might even slightly hurt the model performance.
- The XGBoost model suffers more from overfitting. Similarly, the Neural Network model with one-hot encoder also shows overfitting, as seen by the training losses being close to 0 while the cross-validation losses are much higher.

2) Hyper-parameter Tuning: Below are the performance comparisons against different models and different values of selected hyper-parameters for the German Credit data. From the above plots, we find that:

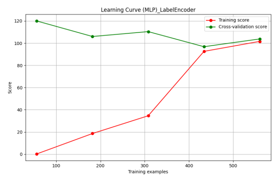
- The Neural Network and XGBoost are less sensitive to the hyperparameters.
- KNN benefits from a larger number of nearest neighbors, but the uplift is marginal after the hyper-parameter reaches 10 or more.
- SVM does not benefit from stronger regularization.



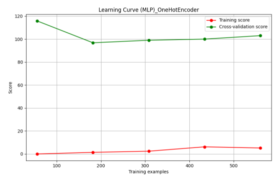
(a) KNN-Label Encoder



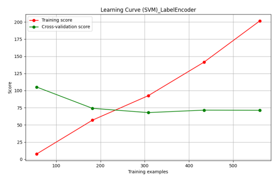
(b) KNN-One-hot Encoder



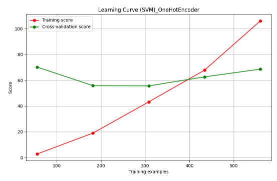
(c) NN-Label Encoder



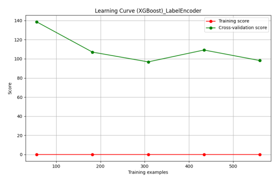
(d) NN-One-hot Encoder



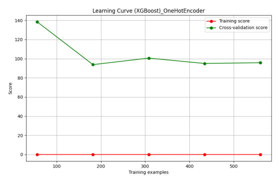
(e) SVM-Label Encoder



(f) SVM-One-hot Encoder



(g) XGBoost-Label Encoder



(h) XGBoost-One-hot Encoder

Fig. 1. The unnormalized weighted misclassification loss from 5-fold cross-validation against different training sizes for the German Credit Data. On the left panel are results with label encoder for categorical variables. On the right panel are results with one-hot encoder.

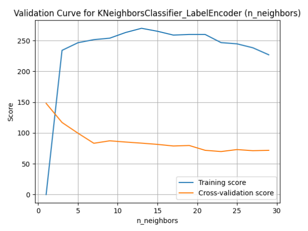
3) Grid Search and Validation: We applied a random grid search in the hyper-parameter space detailed above and report the performance of the best model we found on testing data below.

Model	Encoding Method	Test Score	Train Score
KNN	Label Encoder	281	148.0
	One-hot Encoder	372	175.6
SVM	Label Encoder	499	204.0
	One-hot Encoder	299	148.6
NN	Label Encoder	311	150.0
	One-hot Encoder	201	130.0
XGBoost	Label Encoder	197	99.4
	One-hot Encoder	171	100.8

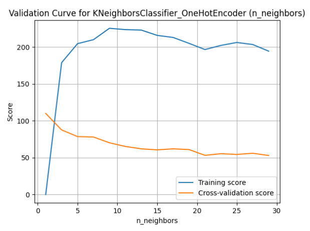
TABLE I
MODEL PERFORMANCE FROM RANDOM GRID SEARCH WITH DIFFERENT ENCODING METHODS ON THE GERMAN CREDIT DATA.

With a thorough search in the full parameter space, we find that XGBoost with One-hot encoder performs the best across all models and encoding methods on the test data of German Credit. The wining model has the following hyperparameters:

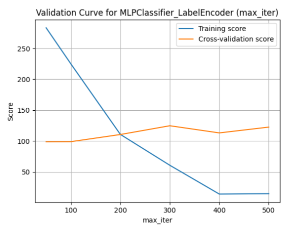
- Number of estimators: 145



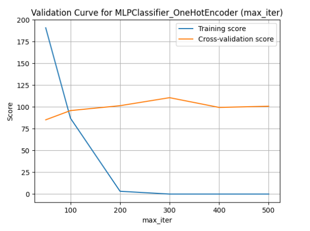
(a) KNN-Label Encoder



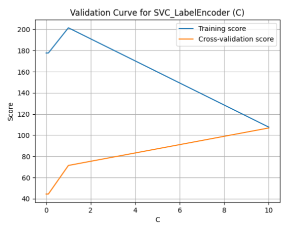
(b) KNN-One-hot Encoder



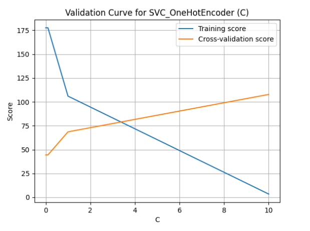
(c) NN-Label Encoder



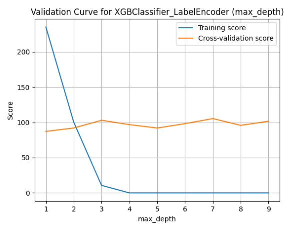
(d) NN-One-hot Encoder



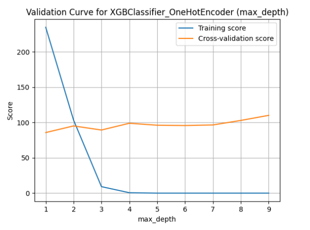
(e) SVM-Label Encoder



(f) SVM-One-hot Encoder



(g) XGBoost-Label Encoder



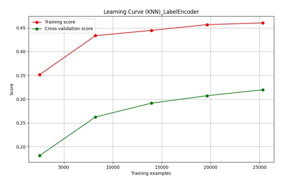
(h) XGBoost-One-hot Encoder

Fig. 2. The unnormalized weighted misclassification loss from 5-fold cross-validation against different training sizes for the German Credit Data. On the left panel are results with label encoder for categorical variables. On the right panel are results with one-hot encoder. For KNN, we fine-tuned the number of nearest neighbors. For NN, we fine-tuned the maximum number of iterations. For SVM, we fine-tuned the strength of regularization. For XGBoost, we fine-tuned the maximum depth of the tree.

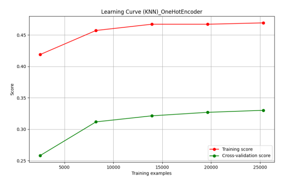
- Maximum depth of trees: 4
- Learning rate: 0.184
- Subsample ratio of the training instances: 0.96
- Subsample ratio of columns when constructing each tree (colsample_bytree): 0.61
- Minimum loss reduction required to make a further partition on a leaf node of the tree (gamma): 0.21

B. Bank Marketing Data

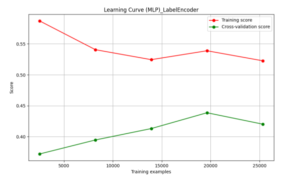
1) Training Size: Below are the performance comparisons against different models and training sizes for the Bank Marketing data. From the above plots, we can tell that:



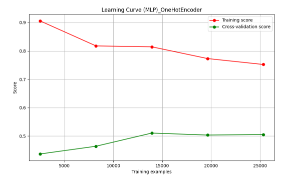
(a) KNN-Label Encoder



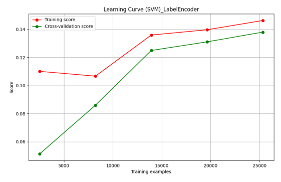
(b) KNN-One-hot Encoder



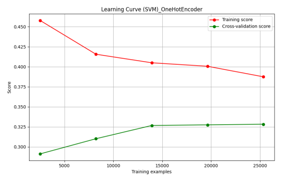
(c) NN-Label Encoder



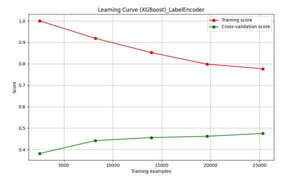
(d) NN-One-hot Encoder



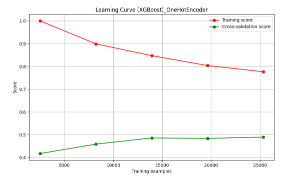
(e) SVM-Label Encoder



(f) SVM-One-hot Encoder

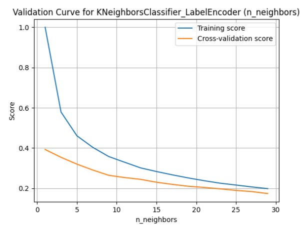


(g) XGBoost-Label Encoder

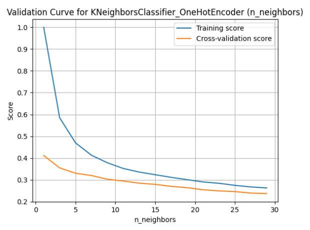


(h) XGBoost-One-hot Encoder

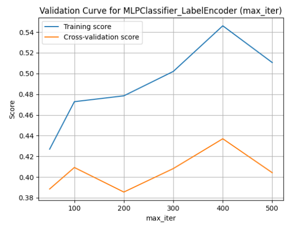
Fig. 3. The recall rate from 5-fold cross-validation against different training sizes for the Bank Marketing Data. On the left panel are results with label encoder for categorical variables. On the right panel are results with one-hot encoder.



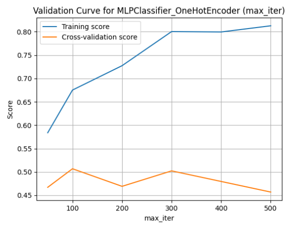
(a) KNN-Label Encoder



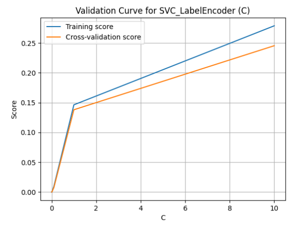
(b) KNN-One-hot Encoder



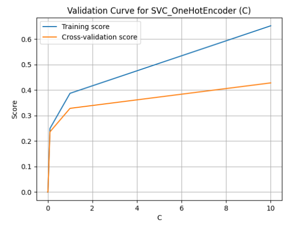
(c) NN-Label Encoder



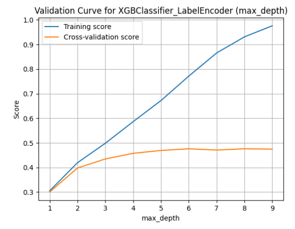
(d) NN-One-hot Encoder



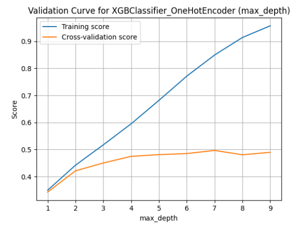
(e) SVM-Label Encoder



(f) SVM-One-hot Encoder



(g) XGBoost-Label Encoder



(h) XGBoost-One-hot Encoder

Fig. 4. The recall rate from 5-fold cross-validation against different training sizes for the Bank Marketing Data. On the left panel are results with label encoder for categorical variables. On the right panel are results with one-hot encoder. For KNN, we fine-tuned the number of nearest neighbors. For NN, we fine-tuned the maximum number of iterations. For SVM, we fine-tuned the strength of regularization. For XGBoost, we fine-tuned the maximum depth of the tree.

- The overall best model is the neural network with one-hot encoder. XGBoost's performance is very close.
- The larger the sample size, the higher the recall rate. However, the effect will be saturated at a certain point.
- When applying one-hot encoder with small sample sizes, the neural network and XGBoost both suffer from over-fitting (the training score is close to 1 but the cross-validation score is much lower).

2) Hyper-parameter Tuning: Below are the performance comparisons against different models and different values of selected hyper-parameters for the Bank Marketing data. From the above plots, we find that:

- Similar to the German Credit data, the Neural Network and XGBoost are less sensitive to the hyper-parameters. Notably, when the maximum depth of trees increases, the model might be overfitted but the cross-validation score remains constant.
- Unlike the German Credit data, when the number of neighbors increases, the KNN tends to perform worse.

3) Grid Search and Validation: We applied a random grid search in the hyper-parameter space detailed above and report the performance of the best model we found on testing data below.

With a thorough search in the full parameter space, we find that the neural network with One-hot encoder performs the best across all models and encoding methods on the test data for Bank Marketing data. The wining model has the following hyperparameters:

- Hidden layer sizes: deep neural network (32, 32, 32, 32, 32, 32, 32)
- Activation functions: ReLU

Model	Encoding Method	Test Score	Train Score
KNN	Label Encoder	0.399	0.408
	One-hot Encoder	0.424	0.412
SVM	Label Encoder	0.381	0.364
	One-hot Encoder	0.434	0.430
NN	Label Encoder	0.383	0.558
	One-hot Encoder	0.632	0.549
XGBoost	Label Encoder	0.491	0.487
	One-hot Encoder	0.504	0.489

TABLE II
MODEL PERFORMANCE FROM RANDOM GRID SEARCH WITH DIFFERENT ENCODING METHODS ON THE BANK MARKETING DATA.

- Solvers: SGD
- Alpha (L2 penalty): 0.0001
- Learning rate: 0.001
- Maximum number of iterations: 500

V. DISCUSSION

The above results verify our hypotheses. See detailed discussion below.

A. *Encoding method matters*

We notice that the one-hot encoder generally outperforms or is at least as good as the label encoder across all scenarios. One possible explanation is that when applying the label encoder we assign an ordinal numerical value to class memberships. This approach makes sense in certain situations, e.g., we can translate the highest education level from (high school, college, graduate school) to (1, 2, 3) as there is indeed a natural order for different educational levels which reflects the number of years in education. However, for certain situations such a natural order does not exist. An obvious example is racial identity which does not reflect any logical order. In this case, representing the categorical feature with an ordinal numerical value will be misleading or even hurt the model performance.

Another issue we need to consider when applying the label encoder is that even though there could be an order in the categories, that might not be faithfully reflected in the metric space. For example, if we use Euclidean distance when applying KNN, then the difference between someone with a high school diploma and another one with a graduate school degree is 4 times as much as that between someone with a college degree and another one with a graduate school degree. However, such a difference cannot be easily justified.

On the other hand, if we only apply the one-hot encoder to categorical variables with many categories, we will face the challenge of inflated dimensions. For example, in the German Credit data, we only have 1000 observations in total and the challenge of high dimensionality is even more severe. That partially explains why the advantage of the one-hot encoder is less revealed in the German Credit experiment, but for the Bank Marketing data, it consistently outperforms the label encoder.

In practice, we should pick the encoding approach based on the nature of each categorical variable instead of applying one encoder for all.

B. *Handling non-linearity helps*

The XGBoost and neural network consistently show superior performance compared to the other two models under different settings. It could be explained by their capability of handling non-linearity. KNN and SVM are essentially distance-based methods. Although we can apply different distance metrics or kernels, they cannot effectively capture the interactions between features as well as other types of non-linearity.

In our experiments, both datasets have a fairly large number of features, especially when we apply label encoder or one-hot encoders to categorical variables. Hence, it is not surprising to see that XGBoost or neural network consistently outperform the distance-based methods.

C. *XGBoost is justified as the go-to method*

While both XGBoost and neural network are shoulder to shoulder in our experiments, we notice that:

- XGBoost has a slight advantage with out-of-the-box settings, while the optimal neural networks are usually both wide and deep, which aligns with the observation in [3].
- In terms of hyper-parameter tuning, XGBoost's performance is less reliant on fine-tuned hyper-parameters, while neural network's performance varies with different sets of hyper-parameters.
- In terms of the training time, XGBoost is significantly faster (1s versus 35s for neural network).

The above observations justify why XGBoost is the industry standard model for tabular data. While neural network-empowered deep learning shines in many frontier applications such as language models or image recognition, in most basic machine learning tasks practitioners would prefer to implement XGBoost which works well enough out-of-the-box, so that we can spend more time on data cleaning and scaling up.

D. *Future study*

Our experiments are limited in a few ways. First, as discussed above, each categorical variable should be treated separately. Second, the grid search is limited by the relatively small parameter space. In our next step, we could focus on how to choose the optimal encoding method and expand the hyper-parameter space to conduct a more thorough comparison.

REFERENCES

[1] Hofmann, Hans. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. https://doi.org/10.24432/C5NC77.
[2] Moro, S., Rita, P., and Cortez, P.. (2012). Bank Marketing. UCI Machine Learning Repository. https://doi.org/10.24432/C5K306.
[3] Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Shah, H. (2016, September). Wide & deep learning for recommender systems. In Proceedings of the 1st workshop on deep learning for recommender systems (pp. 7-10).