Business School and Department of Computer Science

Learning from Data Individual Report

Report Title: Learning from Data Report

Anonymous ID: Z0150297

Word Count: 1252

I declare that;
The attached work is my own.
I have read and understood the Durham University policy on academic misconduct.
Text quoted from journals, books, web or other source is clearly identified (e.g. quotation marks) and referenced at the point of use.
Figures, diagrams, tables or other non-textual information not of my own construction has been referenced at the point of use.
I have neither sought nor used the services of any professional agencies to produce this work.
This work has not been submitted previously for any other assessed unit for this or other degree courses.

Learning from Data Report

1. Data Set Explanation and Data Cleaning Process

The data used in the assignment are described below [1]:

"This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like ``leaflets three, let it be'' for Poisonous Oak and Ivy."

The original dataset contains 22 attributes (see appendix 1 for details) and two classes, edible (e) and poisonous (p). Additionally, all data in the dataset are categorical. This assignment aimed to train two machine learning classifiers, random forest and neural network classifiers, to classify these two classes.

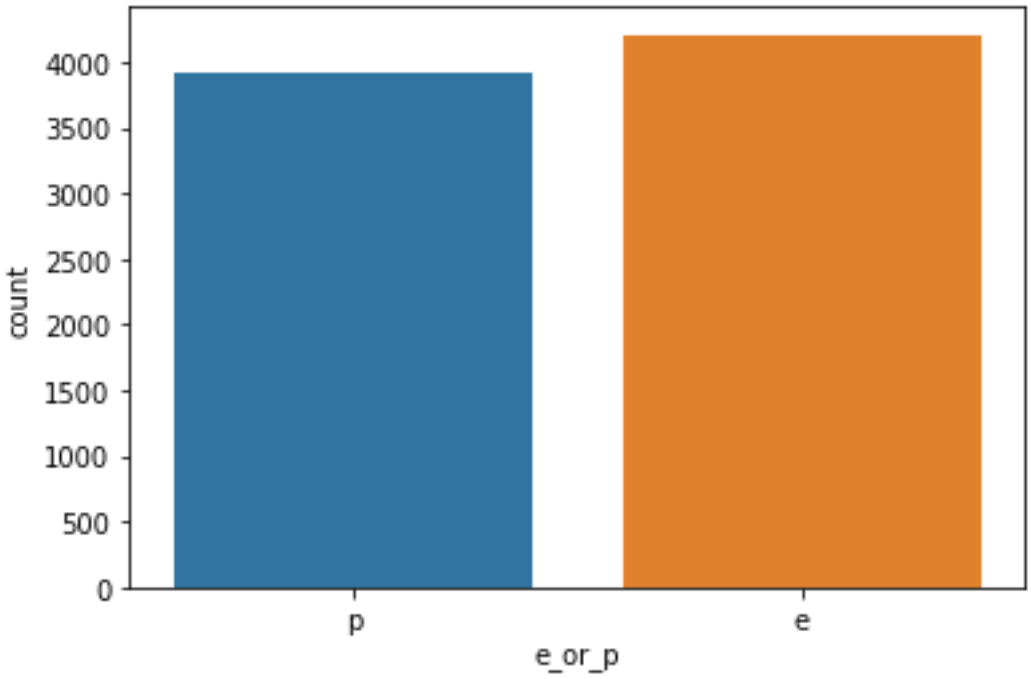

Figure 1. Counts of two classes of mushroom, edible (e, in orange) and poisonous (p, in blue).

As seen in figure 1, the dataset contains similar numbers of edible and poisonous classes. Hence, there was no need to balance the dataset.

Moreover, the attribute 'veil-type' only contains one value 'p', which was useless for model training. Therefore, this attribute was dropped. To carry out further data cleaning, a histogram was plotted for every attribute remained, and the results are shown in figure 2.

Figure 2. Histograms of classes and remaining attributes.

It is noted from figure 2 that for the attribute 'gill-attachment', the height-ratios between columns of the same class are similar. Therefore, the posterior probabilities $P(feature|class)$ are approximately the same for the two classes. Additionally, P (e) is roughly equal to P (p) for this dataset. According to the formula of Bayes theorem shown below,

$$P(class|feature) = \frac{P(feature|class) * P(class)}{P(feature)}$$

$P(class|feature)$ are almost identical for the two classes, which means that this attribute is inefficient for Naïve Bayes classifier training. Hence, this attribute was dropped.

Furthermore, it is also noted that nearly 1/3 of values are missing value for attribute 'stalk-root'. By considering that different plans of making up these missing values may lead to different training results and that there is no evidence available for evaluating these plans, the attribute was also dropped.

2. Explanation of random forest

The idea of random forest classifier is based on the concept of decision tree. "Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression" [2]. Decision trees can be represented by flow-chart structure, with each internal node as an attribute test, each branch corresponding to attribute value and each leaf node assigning a classification. Figure 3 demonstrates an example of decision tree structure.

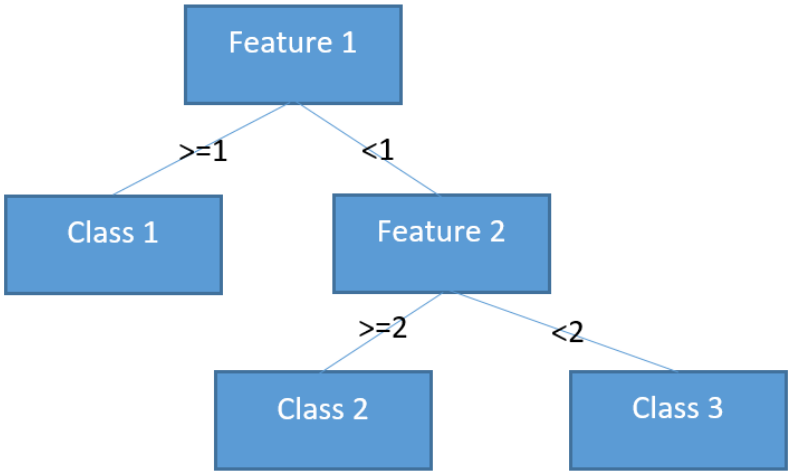

Figure 3. A simple demonstration of decision tree structure.

A decision tree is constructed from top to bottom based on the node splits that maximise the reduction in the entropy (how many data in different classes are mixed) in each resulting sub-branch of the tree [3]. Although decision tree is extremely powerful for classification on training data, it often creates over-complex trees that may not be general for testing data, known as overfitting [2]. However, random forest machine learning technique helps to solve this issue. In short, random forest classifier builds on a group of decision trees, and the classification decision is made based on the mode (most frequent output) of the prediction made by each decision tree. In this assignment, Bagging Classifier (samples are drawn with replacement) was

used. For Bagging Classifier, increasing the number of estimators often increases the accuracy of the model but with a cost of increasing the training time. Therefore, grid search was performed to find the highest possible accuracy with the lowest training time (or lowest number of estimators).

3. Neural network model optimization

In this section, neural network machine training was performed. The optimization for the neural network model was done in the following five steps:

I. Train the model with a large number of epochs, look for the point of overfitting and fix that epoch number for the rest of the steps.
II. Try different activation functions (relu or sigmoid), select and fix the one that gives higher accuracy.
III. Try adding more layers with different number of output units in each layer, select and fix the one that gives higher accuracy.
IV. Try different batch size, select and fix the one that gives higher accuracy.
V. Try different learning rate, select and fix the one that gives higher accuracy.

The results of the optimization steps are indicated in Table 1 below.

Optimization test	**Best structure or hyperparameter**	**Accuracy**
Activation function: Sigmoid or relu	relu	0.99
Layer: Adding one layer or not Different number of output units (5, 10, 20, 30, 40, 50, 60, 70, 80)	1. No hidden layer 2. One hidden layer with 5, 10, 30, 40, 70 or 80 output units	0.99
Batch size: Different batch size (5, 10, 20, 32, 40 ,50, 100, 200)	Batch size equal to 5 or 32	0.99
Learning rate: Different learning rate (from 1, 0.5, 0.1, 0.01, 0.001)	Learning rate equal to 1, 0.5, or 0.01	0.99

Table 1. Best structure or hyperparameter results of different optimization test according to accuracy.

It is noted from Table 1 that for layer optimization test, the accuracies are the same for the case of no hidden layer and adding one hidden layer with 5, 10, 30, 40, 70 or 80 output units. However, adding one hidden layer is equivalent to increasing the complexity of the model, which may make the model less generous for new datasets. Therefore, no hidden layer was chosen for the rest of the tests.

Furthermore, for batch size test, although two of the batch sizes both result in the highest accuracy, the commonly used batch size, 32, was selected by considering the following two factors:

a) Training time, having too small batch size will lead to slower training time as more iterations per epoch are performed.
b) Generalization ability, some papers show that too large batch size tends to converge to sharp minimizers of training and testing functions, leading to poor generalization [4].

Finally, for the learning rate test, all five learning rates result in the same accuracy, but learning rate 0.01 was selected for the final model. Again, there are two considerations for this decision. First, a large learning rate can lead to the oscillation of learning curve (see example in figure 4), causing overshooting the minima and ineffective model training [5].

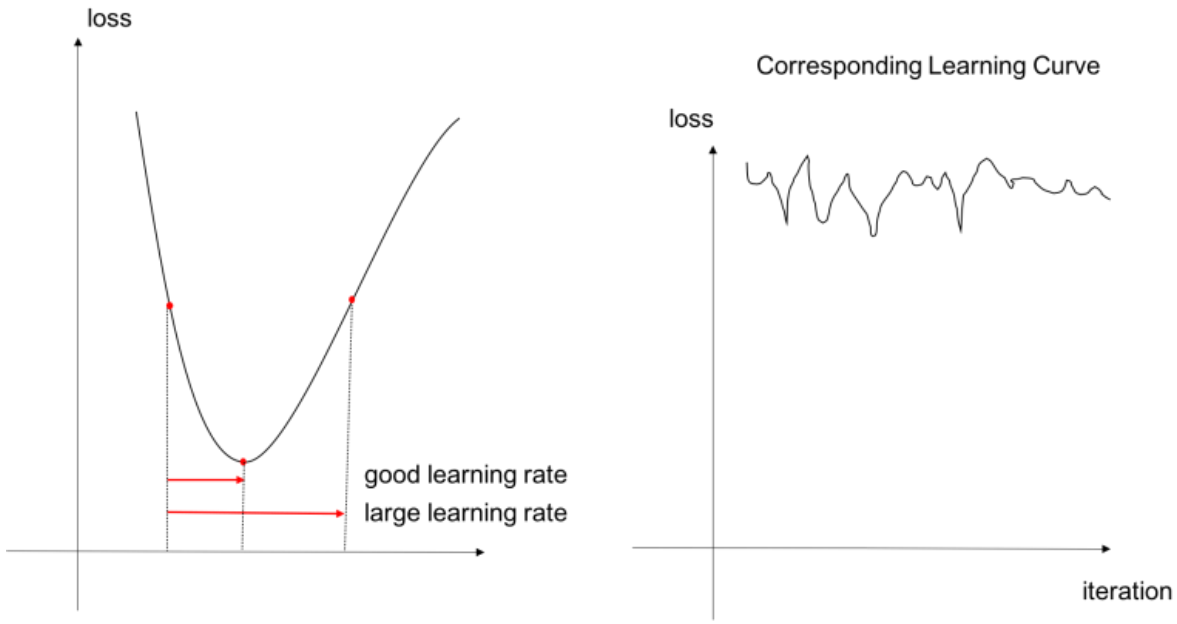

Figure 4 [5]. A demonstration of the effects on loss function (left diagram) and learning curve, by having a large learning rate.

Second, a small learning rate leads to a slow model optimization process as changes made to the weights for each update are smaller and more epochs are required. Besides, a small learning rate can also lead to local minima, but this can be improved by applying Nesterov momentum.

In summary, the hyperparameter and structure setups for the final neural network model are: 'relu' for activation function, no hidden layers, batch size 32 and learning rate 0.01.

4. Comparison and analysis

In the first part of this section, two comparisons were made for the random forest model, which were datasets using label encoder and one-hot encoder, and datasets with different size. The results of the comparisons are shown in table 2.

Dataset	Training time	Cross validation mean score (cv=10)	Number of training data points
Original dataset with label encoder	807 ms ± 11 ms	0.959	154356
Original dataset with one-hot encoder	1.74 s ± 164 ms	0.968	731160
Smaller size dataset with one-hot encoder	211 ms ± 3.82 ms	0.998	86686

Table 2. Training times, cross validation mean scores (cv=10) and numbers of training data points for three different datasets.

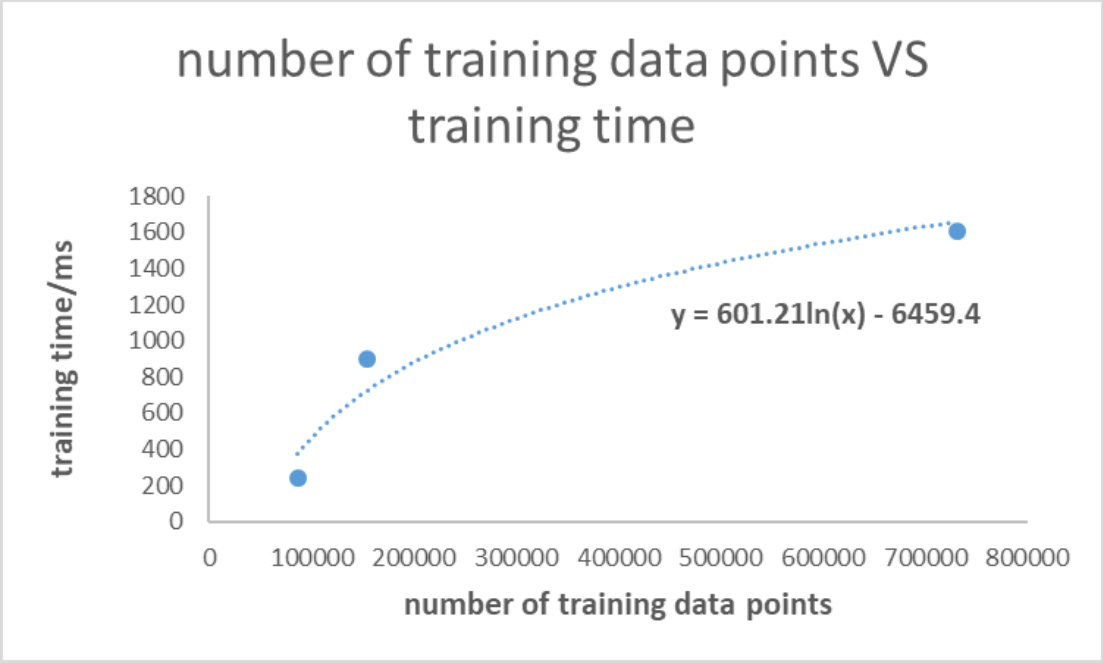

Figure 5. Number of training data points against training time.

First, from table 2 and figure 5, it can be seen that training time is proportional to the number of training data points and that they are in a natural logarithmic relation. Second, after using one-hot encoder, the accuracy of the random forest model increases; this is expected as one-hot encoding method removes unintended pattern generated by label encoder, which can distract the training process. Finally, interestingly, as the size of the dataset decreases, the accuracy increases. Nevertheless, due to having limited time for this assignment, no sensible

reasons could be found to explain this phenomenon. However, it is worth for someone to perform a further investigation on this 'problem' in the future.

In the second part of this section, a comparison was made between the random forest model and neural network model on the same dataset (smaller size dataset with one-hot encoder). The result of this comparison is shown in Table 3.

Model name	**Accuracy/ Cross validation mean score (cv=10)**	**Training time**
Random forest	1	211 ms ± 3.82 ms
Neural network	0.99	41 ms ± 1.15 ms

Table 3. Accuracies/cross validation mean scores (cv=10) and training times for random forest and neural network models on the smaller size dataset using one-hot encoder.

Summarising from table 3, although the random forest model has slightly higher accuracy than the neural network model, this difference is negligible as compared to the difference between the training times of the two models. Hence, for mushroom data, neural network classifier is more suitable than random forest classifier.

Reference list:

[1] Schlimmer, J. (1987). UCI Machine Learning Repository: Mushroom Data Set. [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/mushroom [Accessed 19 Dec. 2019].

[2] Scikit-learn.org. (n.d.). 1.10. Decision Trees — scikit-learn 0.22 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/tree.html [Accessed 19 Dec. 2019].

[3] J.R. Quinlan. (1993). C4. 5: programs for machine learning. Morgan Kaufmann.

[4] Keskar, N., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. ICLR.

[5] Kurita, K. (2018). Learning Rate Tuning in Deep Learning: A Practical Guide. [online] mlexplained.com. Available at: https://mlexplained.com/2018/01/29/learning-rate-tuning-in-deep-learning-a-practical-guide/ [Accessed 19 Dec. 2019].

Appendix 1

Attribute Information:

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,
pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,
green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,
pink=p,red=e,white=w,yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,
pink=p,red=e,white=w,yellow=y
16. veil-type: partial=p,universal=u
17. veil-color: brown=n,orange=o,white=w,yellow=y
18. ring-number: none=n,one=o,two=t
19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,
none=n,pendant=p,sheathing=s,zone=z
20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,
orange=o,purple=u,white=w,yellow=y
21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d