Supervised Learning Assignment

# 1.1 Requirements

Python 3.6, Scikit-learn 0.21.3, Pandas 0.23.0, Numpy 1.14.3, Keras 2.2.4 with Tensorflow backend

# 1.2 Dataset

## Dataset 1

The first dataset is a modification of the “Otto Group Product Classification” from Kaggle (<https://www.kaggle.com/c/otto-group-product-classification-challenge>). For the purpose of this assignment, this dataset is modified to balance to number of each label.

## Dataset 2

The second dataset is the EEG eye state dataset from UCI machine learning repository (<https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State>), where y is the eye state of 1 (open) or 0 (closed). The dataset is used as it is without any modifications.

## ***Why I Find These Datasets Interesting***

First, both problems have practical applications. The first dataset is from the retail industry. The inputs are 91 features and the output are item labels, which is very useful to build recommender system. The second problem is very useful to develop model for face identification, where eye blinking is important to differentiate real face vs. fake ones such as photos.

Second, the first dataset formulates a multiclass classification problem while the second dataset is a binary classification problem. Therefore working on these datasets offers a wider scope of classification problems.

## Dataset Pre-processing

For Otto dataset, the original dataset has the following statistics:

Class\_2 16122

Class\_6 14135

Class\_8 8464

Class\_3 8004

Class\_9 4955

Class\_7 2839

Class\_5 2739

Class\_4 2691

Class\_1 1929

The class with highest counts (Class\_2) is about 8 times as much as the class with lowest counts (Class\_1). If the original dataset is applied for model training, classes with lower counts are prone to be treated as noise, since machine learning models are better to identify similarity rather than abnormity. The address this issue, I reduce the dataset to include only the 4 classes with highest counts (Class\_2, \_6, \_8, \_3).

Otto dataset also contains a large number of features (93), which leads to a long model training time. To finish this assignment in a reasonable time frame, a new Otto dataset is created based on 40 randomly selected features out of the original 93 features.

Some key attributes for both datasets are tabulated below.

|  |  |  |
| --- | --- | --- |
|  | Otto Dataset | EGG dataset |
| Size | 46725 x 40 | 14980 x 15 |
| Features | All numeric | All numeric |
| Classification Type | Multiclass (4 classes) | Binary (2 classes) |

## Exploratory Data Analysis

For dataset 1, heatmap on the original dataset (Fig 1, left) suggests all features except ‘amount’ has no collinearity, which according to the data description on Kaggle is due to PCA, and also is desirable for a machine learning. However after the data pre-processing, heatmap on the new dataset suggests some collinearity, especially amount V1~V18 (Fig 1, right). This is the result of synthetic data generated to balance the imbalanced original dataset.

For dataset 2, heatmap suggests collinearity exists among multiple feature pairs.

|  |  |
| --- | --- |
|  |  |

Figure 1. heatmap on the original dataset (left) vs. heatmap on the new dataset (right)

A picture containing object, toiletry

Description automatically generated

Figure 2. heatmap on dataset 2

In practice, more efforts should be spent on feature engineering to reduce collinearity. For the purpose of this assignment, however, I only focus on model comparison and feature engineering is out of the scope of this assignment.

Results

5-fold cross-validation is applied when available.

Training and testing accuracy, training and testing loss function are presented when available.

Final accuracy/precision/recall results are presented based on the best set of parameters from cross validation.

# 2.1 Decision Trees

|  |  |
| --- | --- |
| Item | Description |
| Package | sklearn.tree.DecisionTreeClassifier |
| Base hyperparameter | Criterion: GINI  Min\_samples\_split: 3  Min\_samples\_leaf: 2  Max\_features: ‘auto’  Scoring: ‘accuracy’ \*  The rest are the default params |

To test the effect of pruning on model results, an array of different maximal branch depth is selected as [ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75]. Results for Otto dataset are shown in Figure 2.1, and for EGG dataset is shown in Figure 2.2.

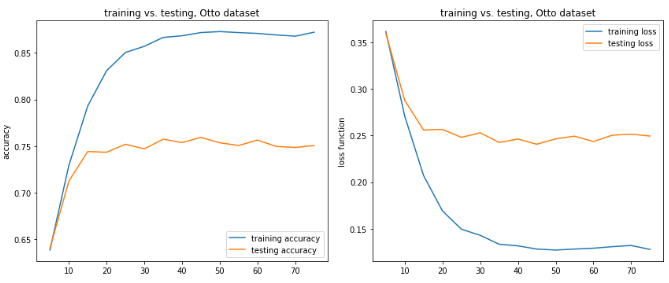


Figure 2.1 accuracy (left) and loss (right) for Otto dataset

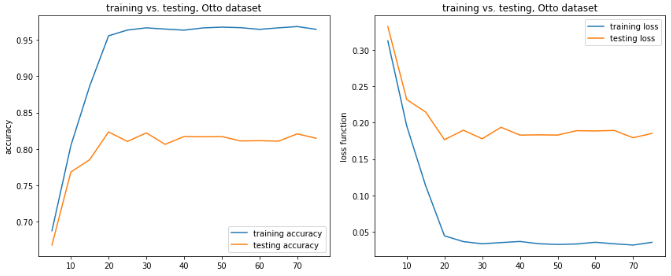


Figure 2.2 accuracy (left) and loss (right) for EGG dataset

Note that the x axis is the maximal depth of branches.

Results from both datasets show similar pattern: initially as max branch depth grows, both training and testing accuracy improves. For Otto dataset when max branch depth grows beyond 20, even though training accuracy continues to improve, testing accuracy reaches its plateau. This is a clear indication of overfitting. It suggests that even new information can still be learned from the training dataset (more branch growth for information gain), the new knowledge cannot be generalized when tested against the testing set, where deeper depth does not improve the testing accuracy score at all. For the Otto case, maximal branch depth should be set at around 20 for the best testing accuracy, as well as maintaining model training efficiency.

Same behavior can be observed for EGG dataset result as well around max branch depth= 15, but in a less prominent fashion.

# 2.2 Neural Networks

|  |  |
| --- | --- |
| Item | Description |
| Package | Keras with Tensorflow backend |
| Base hyperparameter | Hidden layer: 4  Number of neurons for each hidden layer: 1024, 512, 128, 32  Dropout between each layer: 0.2  Activation function: ‘relu’ for hidden layer, ‘sigmoid’ for output layer  Loss: ‘sparse\_categorical\_crossentropy’  Epoch: 300 for Otto dataset, and 1200 for EGG dataset  Early stopping: when loss does not improve for 40 epochs (EGG dataset) |

Figure 2.3 shows the results (accuracy and loss) for Otto dataset case. 80% of Otto dataset is randomly selected for model training and the remaining 20% is used for testing (validation).

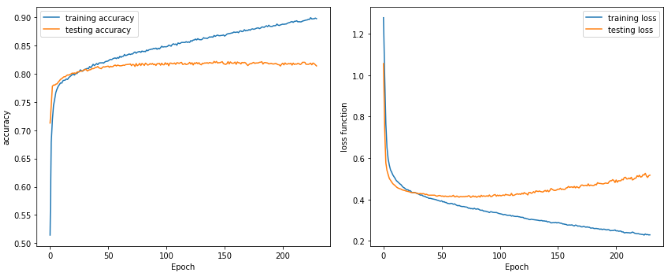


Figure 2.3 accuracy (left) and loss (right) for training and testing, Otto dataset

Accuracy scores (Figure 2.3, left) for both training and testing keep improving until epoch 50, after which further iterations only improve training score but not testing score. This observation echoes with the results seen in Decision Trees, where new knowledge learned from the training dataset cannot be generalized.

Loss plot (Figure 2.3, right) shows that for testing, minimal loss is reached at around epoch 50, which agrees with the observation in accuracy score plot. After epoch 50, the decreasing training loss does not reduce testing loss. Quite to the contrary, testing loss starts to increase after epoch 50. For the loss function used in this case (categorical cross entropy), loss is calculated based on probability which varies between 0 and 1. When loss increases but accuracy does not change, it means the model is getting less confident in predicting the labels, even though the labels are predicted correctly. For instance, a label is predicted as 1 when the probability of being 1 is 0.8. When the probability of being 1 decreases (for example, 0.7), the loss increases, but the predicted label does not change given 0.7 is still greater than 0.5.

Figure 2.4 shows the results for EGG dataset. Model training stops at epoch 898 when it does not see any improvement in training loss.

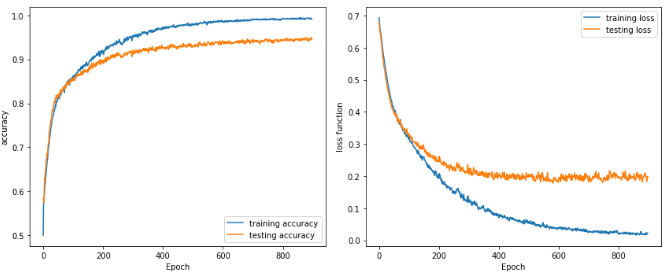


Figure 2.4 accuracy (left) and loss (right) for training and testing, EGG dataset

Unlike Figure 2.3, loss function for testing does not deteriorate after it reaches the minimum in Figure 2.4. Instead it remains at the minimal value beyond epoch 400, meanwhile training loss keeps decreasing. After epoch 400, training accuracy improve by 3% while during the same period testing accuracy only improves by 1%.

# 2.3 Boosting

There are many boosting algorithms and packages available. In this assignment, I only use the Gradient Boost package from Scikit-learn.

|  |  |
| --- | --- |
| Item | Description |
| Package | Sklearn.ensemble.GradientBoostingClassifier |
| Base hyperparameter | N\_ estimators: 300 (Otto), 1200 (EGG)  max\_depth: 3  The rest are the default params |

One interesting, and also attractive, behavior of boosting algorithm is that it usually does not overfit, if the weak learner does not overfit. In this case, the weak learner is a tree model with maximal 3 layers. Recall the results obtained in 2.1 Decision Trees where overfitting happens at max\_depth=20 (Otto) and max\_depth=15 (EGG), a tree model with maximal 3 layers is still underfitting the data. Therefore overfitting for both datasets is unlikely here.

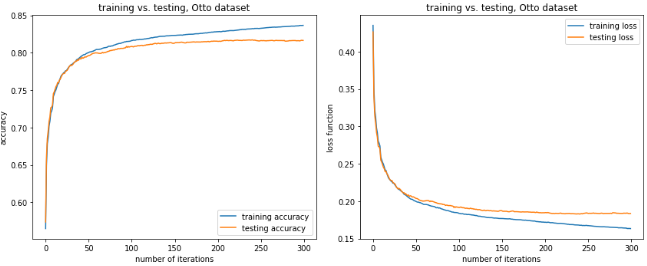


Figure 2.5 accuracy (left) and loss (right) for training and testing, Otto dataset

Both results in Figures 2.5 (Otto dataset) and 2.6 (EGG dataset) agree with the argument. Both training and testing accuracies are improving as number of iterations increases, while training and testing losses are decreasing. Especially for the EGG dataset, after 1200 iterations there are still noticeable improvement in accuracy and loss for both training and testing.

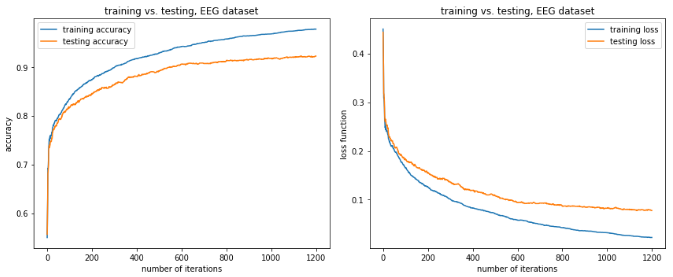


Figure 2.6 accuracy (left) and loss (right) for training and testing, EGG dataset

# 2.4 Support Vector Machines

For Support Vector Machines, two kernel functions are tested: “rbf” and “linear”. For each kernel function, an array of penalty C is selected.

|  |  |
| --- | --- |
| Item | Description |
| Package | Sklearn.svm.SVC |
| Base hyperparameter | Use default params |

## RBF Kernel

Support Vector Machine with RBF kernel results are shown below. The x axis is the penalty value which ranges from 0 to 15000. Note that results are only for EGG dataset, because SVM is too slow to complete model training and testing for Otto dataset.

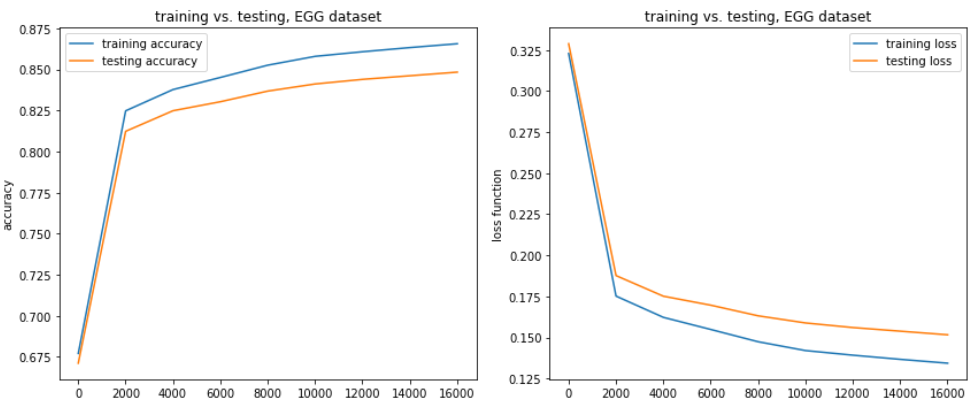


Figure 2.7 accuracy (left) and loss (right) for training and testing, EGG dataset

Even with C value up to 15000, both training and testing scores are still increasing. However attempt to run model training on even larger C value results in very long convergence time (no limitation on iterations) and is there not complete for this assignment.

I have never encountered such large C value in my work. It is interesting to dive deeper into the dataset and try to understand why it happens, but it is beyond the scope of this assignment.

## Linear Kernel

Support Vector Machine with RBF kernel results are shown below. The x axis is the penalty value which ranges from 0 to 15000. Similar to the RBF kernel, modeling training on the Otto dataset is very resource demanding and is therefore not complete.

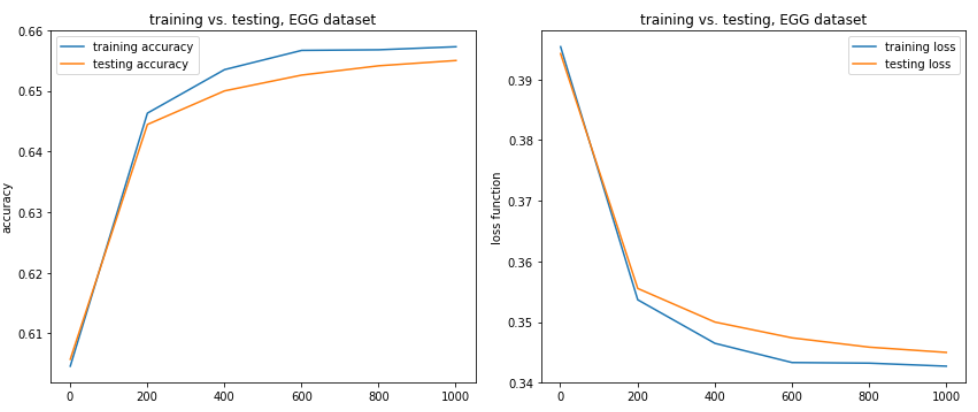


Figure 2.8 accuracy (left) and loss (right) for training and testing, EGG dataset

Compared

# 2.5 k-Nearest Neighbors

KNN is a instance-based model, and as a result, there is no training accuracy or loss. Only testing accuracy and loss is computed.

Discussion

# 3.1 Model Ranking

# 3.2 One-dimensional Grid Search

To address the issue of overfitting, in this assignment usually only one hyperparameter is allowed to change while the remaining are kept constant (for example, only penalty C is allowed to change in Support Vector Machine). In real world, hyperparameters form a hyperspace with much higher dimensions and to search for the best combination of hyperparameters usually requires significant computational resources. A practical approach for grid search is always a balanced effort between model accuracy (or any other metrics that is appropriate) and available resources.

# 3.3 Dataset Comparison

Even though EGG dataset has less than half data points (~20,000 vs. ~ 47,000) than Otto dataset, testing accuracy for all models tested in the assignment all shows a better accuracy of EGG than Otto. Among the many factors that could contribute, Curse-of-Dimensionality is the one that deserves attention.

Curse-of-Dimensionality suggests when dataset dimensionality increases, the number of data points needs to increase exponentially in order to keep the predictive power of a machine learning model. EGG dataset has 14 features and