Name: Zahiduzzaman Biswas
GTID: zbiswas3

First Classification Case:

1. Problem Definition

Using some numeric and categorical data of a person such as age, education, race, job title etc., predict if a person's annual income is more than $50,000 or not.

2. Data

To build the corresponding classifier, we used "adult" data set (Blake and Merz 1998) from the UCI repository. (source: http://archive.ics.uci.edu/ml/datasets/Adult).

3. Project Purpose

Observe the behavior of a few machine learning algorithms such as Decision trees, Neural networks, Boosting, Support Vector Machines, and k-nearest neighbors by building solution models for the above-mentioned problem.

4. EDA & Interesting Points

Provided training and test datasets have 32561 and 16281 instances respectively. After removing the unknown and duplicate data, we have 30162 training instances and 15060 test instances. Both sets are properly labeled and have 14 predictor variables of categorical and numeric types. One of the interesting features of the dataset is that it is not well balanced. In both training and test dataset about 25% instances belong to the class of people with higher income (>50k) and about 75% are in the other class. However, the datasets are already shuffled for experiments.

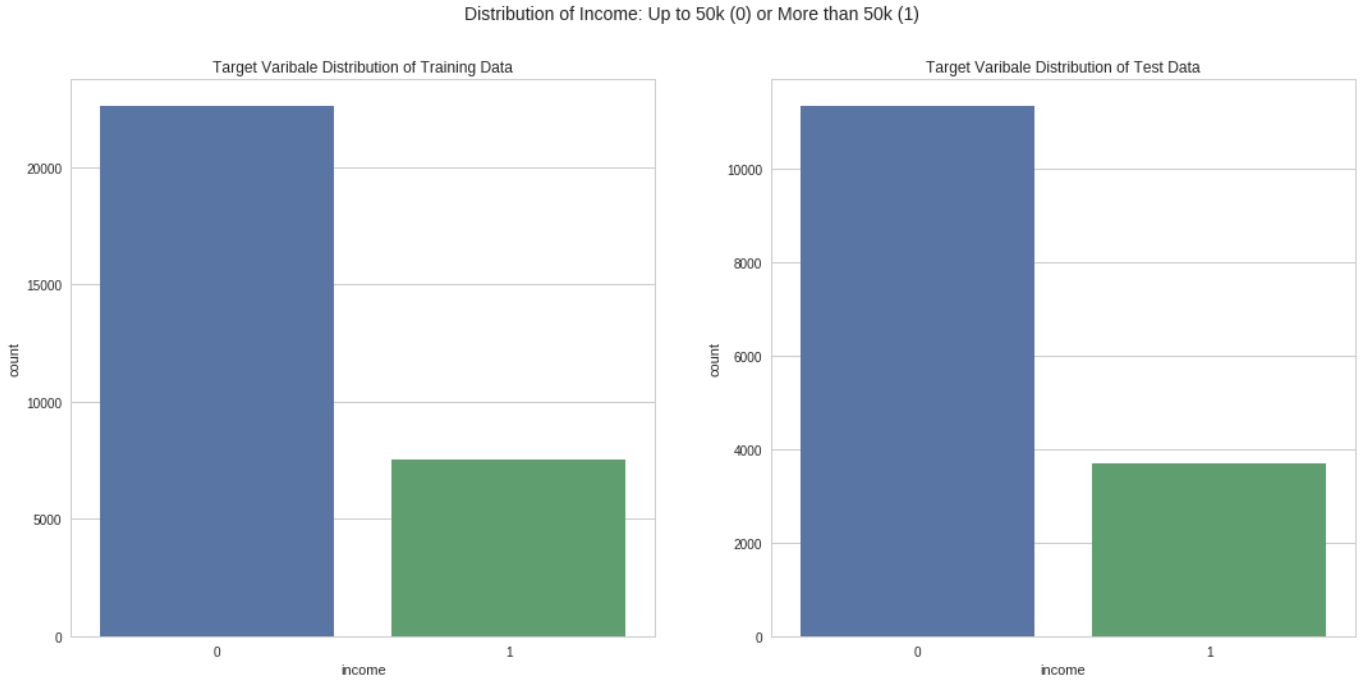


Figure 1. The distribution of the instances in the datasets by target classes

In the training data, the predictor variables do not show any kind of strong correlation with the target variable. Some of the variables show almost zero correlation. **Some of the machine learning algorithms use distances between data points for classification, and some use the categorical features. This dataset would, therefore, be an interesting example to show the differences in performance of these algorithms**.

Name: Zahiduzzaman Biswas
GTID: zbiswas3

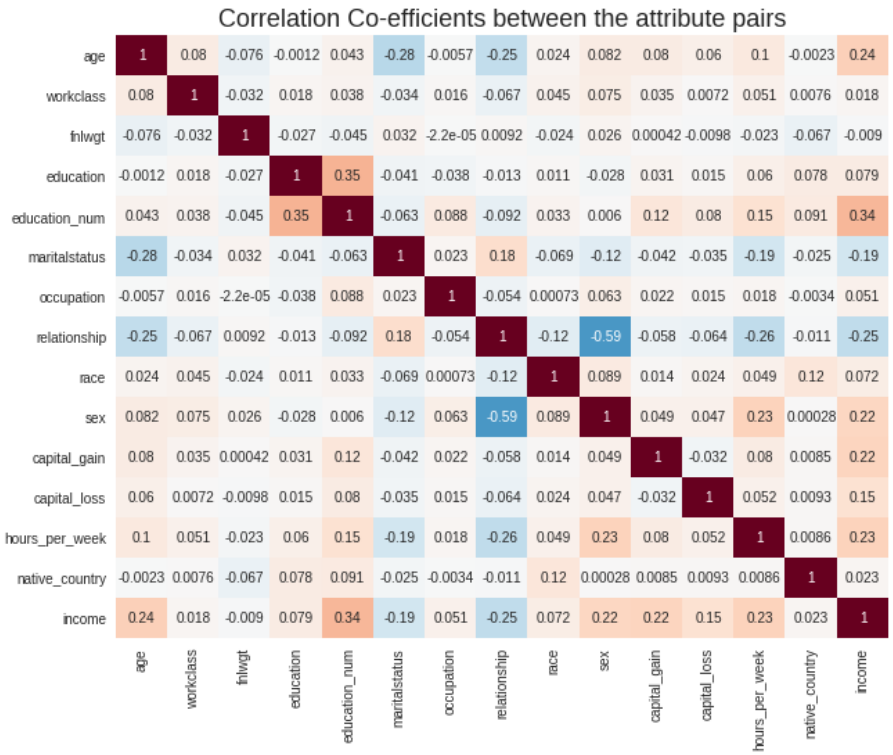


Figure 2: Correlation heat-map of the variable pairs of training dataset

5. Experiments and Approaches

5.1. Decision Trees:

Decision trees classification model is a non-parametric supervised learning method that predicts the value of a target variable by learning simple decision rules inferred from the data features. There are various algorithms to implement this method. We used an optimized version of CART (classification and regression trees) algorithm, which constructs binary trees using the numeric features and threshold that yield the largest information gain at each node. CART uses GINI index as a measure of sanity.

The Gini index:

$$\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

where $\hat{p}_{m,k}$ is the proportion of class k within R_m, and q_m is the proportion of samples in R_m.

Figure 3: Mathematical interpretation of GINI index

At first, we implanted a vanilla version of the classifier without any restriction or pruning using scikit-learn python library. Later we used a grid search and cross-validation approach to find out the optimal depth of the tree and minimum number of data samples at a node before it is split to grow the tree further.

5.3. Boosted Decision Trees:

To observe the effect of boosting, we implemented Adaboost classifier from scikit learn library. We experimented with 10, 50, 250, and 500 estimators with a learning rate of 0.01. A grid search k-fold cross validation was performed to tune the hyper parameters for the best performance.

Name: Zahiduzzaman Biswas
GTID: zbiswas3

5.2. KNN:

We also implemented KNeighborsClassifier from the scikit-learn library. This classification method is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. A query point is classified based on which class has the most representatives within the **k** number of nearest neighbors of the point. We have experimented both with and without pruning the classifier. We also performed cross validation with different hyper-parameters to find the optimal K-value.

5.3. Support Vector Machine

SVM uses a subset of training points in the decision function (called support vectors) near the decision boundary for the classification. We implemented SVC classifier from scikit-learn library with a linear kernel and a non-linear kernel('rbf'). Initially, we fed the dataset without scaling, and the server crashed each time regardless of reducing cross validation folding, hyper-parameter tweaking, and using powerful hardware.

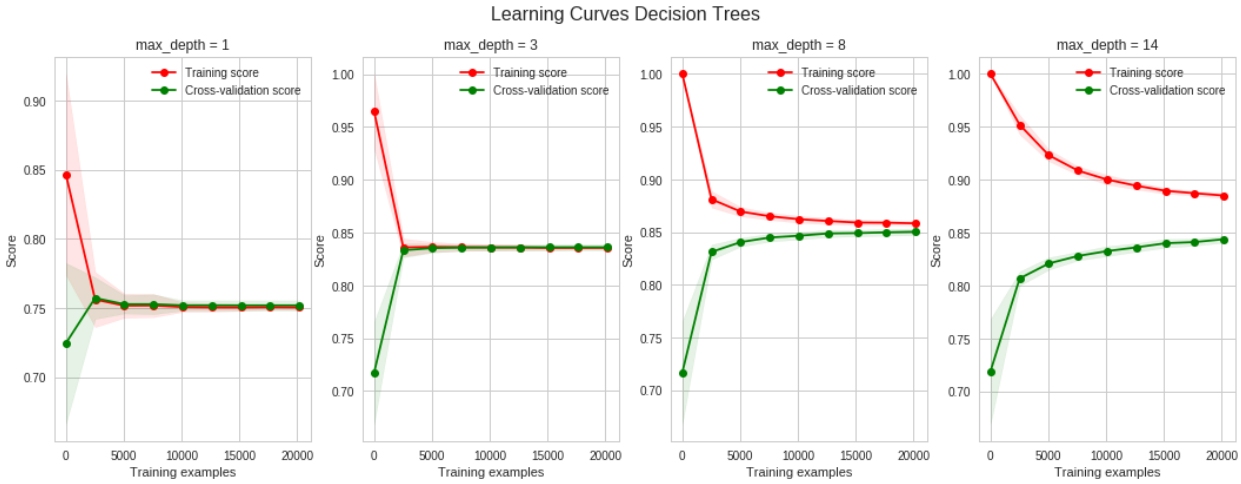
5.4 Neural Network

Finally we implemented deep neural network using python's Tensoflow based Keras framework. We experimented with 3 to 5 densely connected hidden layers with '**relu'** activation function and **adam** optimizer with different learning rate. Each hidden layer contain 200 nodes. 14 scaled feature variables are used for input and one-hot-coded output was produced by the output layer using 'sigmoid' activation function. We compared the binary cross entropy loss of the training and validation data to determine the optimal network architecture and hyper-parameters.

6. Observations and Results Analysis:

6.1 Decision Trees:

I. **Default version with no depth limitation (high complexity)**
   - **a.** Result : training accuracy 99.99% and test accuracy 81.44% (**Overfitting)**
   - **b.** Due to the high complexity and no restriction on branching, trees don't generalize well.
   - **c.** We tune max_depth and min_samples_split for improved performance



II. Pruning: max_depth = 1 and max_depth = 3 , (min_sample_split = 0.001)
   a. At both depth, as we can see from the learning curve, the models are stagnant most of the time. It does not learn from the data.
   b. Overall accuracy for both training and validation set are also very low. These models suffer from Underfitting due to the High bias.

Name: Zahiduzzaman Biswas
GTID: zbiswas3

III. Pruning: max_depth = 14
   a. This models remain unchanged from the untuned model. It suffers from the Overfitting due to the high complexity of the trees.

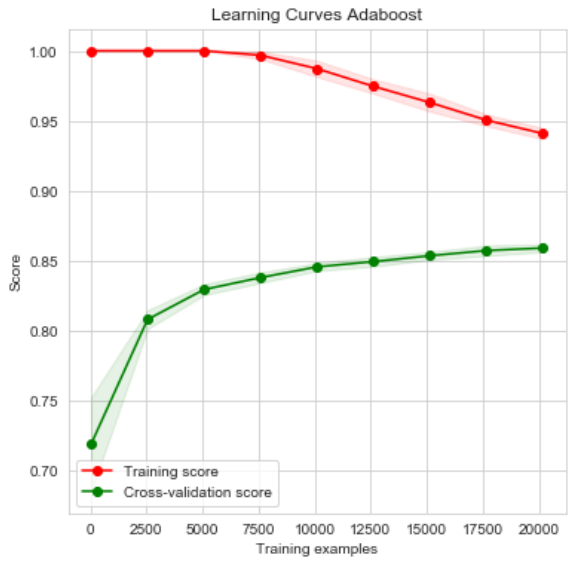
IV. Pruning: max_depth = 8
   a. At this depth the training accuracy drops close to 85% and then plateaus. Similarly, the validation data accuracy increases over the data size. Eventually it reaches close to the convergence. Therefore, the optimal depth would be around 8. We performed k-fold cross validation to with parameter grids to find the optimal hyper-parameters.

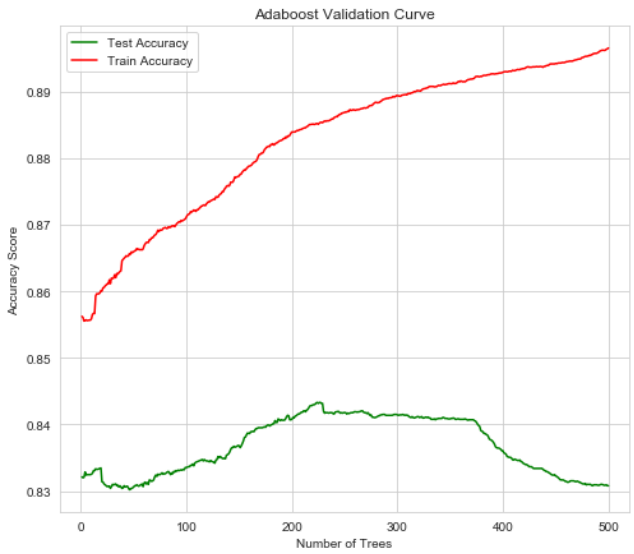
V. Best Model with max_depth = 9 and min_samples_split = 0.001
   a. K-fold cross validation was performed over the grid of parameters: min_samples_split of 0.001, 0.01, 0.03, and 0.05 (since more than 5% is not recommended) and max_depth ranging from 3 to 10.
   b. With optimal max_depth = 9 and min_samples_split=0.001 training accuracy dropped to 85.86% and test accuracy improved to 82.83%. The overfitting is reduced significantly, because of the controlled branching and complexity.

6.3. Boosting with Adaboost classifier:

I. Default Adaboost classifier(n_estimator=50, lr=1) on the pruned Decisiton Trees
   a. Train accuracy is 0. 986 and test accuracy is 0.813 (Highly overfitted model)
   b. Potential Cause of Overfitting: Adaboost classifier (**SAMME.R** algorithm) uses probability estimate to update the additive model. Since the data is imbalanced, the higher probability of the '0' class always has a greater effect on the process.

II. Hyper-parameter tuned Adaboost (n_estimator=300, learning rate=0.01)
   a. From validation curve it is apparent that the optimal number of iterations would be between 150 to 350. In grid search cross validation we found it to be 300 and learning rate 0.01.
   b. Training Score lowered to 0.916 but the test score improved to 0.837**.** Yet the overfitting remains. From validation curve we see that it increases with the iteration. Even after tuning, it can not avoid the effect of the imbalanced data. Handling data imbalance would increase the accuracy which requires more time and powerful hardware.



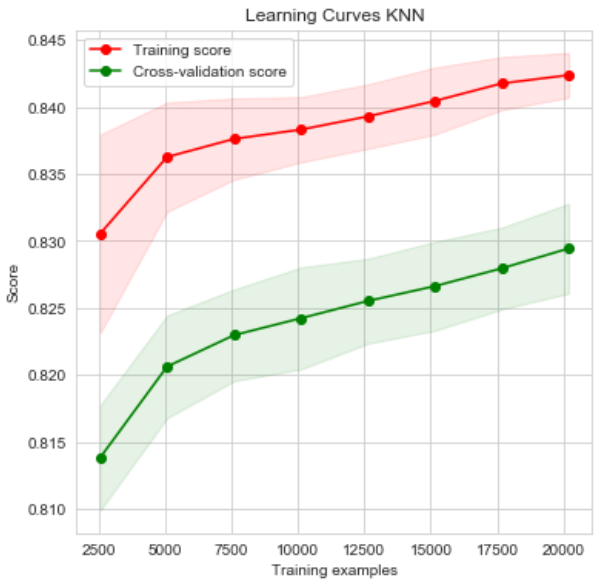


6.4. KNN classifier:

Name: Zahiduzzaman Biswas
GTID: zbiswas3

I. Defualt KNN classifier with k-value =1:
   - **a.** Due to high bias it overfits. Training score is 0.871 and test score 0.817
   - **b.** Cross validation with different Neighbor size would be helpful to improve the situation

II. Hyper-parameter Tuned KNN (Number of Neighbors =23)
   - **a.** From grid search cross validation over the range of 5 to 30 neighbors, the best performing n_neigbors =23
   - **b.** Train score is 0.844 and test score is 0.829. The overfitting problem is solved but the model lacks generalization in learning from data. As we can see in the learning curve thetraining curve and testing curve do not converge.



6.4. SVM classifier:

I. SVM model with Linear kernel (gamma=0.001, C=1)
   - **a.** Through grid search cross validation on Linear kernel we found optimal gamma =0.001 and C=1
   - **b.** Train score 81.25 and test score 81.05 which are close. The model generalized the learning well.
   - **c.** Since it takes a subset of data for classification, effect of the imbalanced data is apparent in the learning curve. Training and testing data is stagnant over data size. The accuracy is score is not as it is expected to be close to 100% on training data set.
   - **d.** Learning curve with gamma = 0.01 and 0.1 is almost same as the optimal one(not shown here), which also shows that gamma has very less influence on Linear SVM.

II. SVM model with Linear kernel (gamma=0.1, C=10)
   - **a.** For rbf kernel too, we performed the grid search cross validation. Interestingly, the hyperparameters are much different from linear kernel. gamma is 0.1 (>0.001) and C=10(>1).
   - **b.** Model performance is also better. Training score 84.29 and test score 84.19.
   - **c.** The penalty factors of this non-linear kernel did improve the accuracy
   - **d.** We at first tried to use SVM without scaling data. It crashed the server. It is potentially because of the distance metrics that it uses for classification. So the support vector space was dispersed to high which caused the extra delay. Scaling reduced the distance and improved the performance. Still SVM models took longest (~4min) wall time to classify the data.

Name: Zahiduzzaman Biswas
GTID: zbiswas3

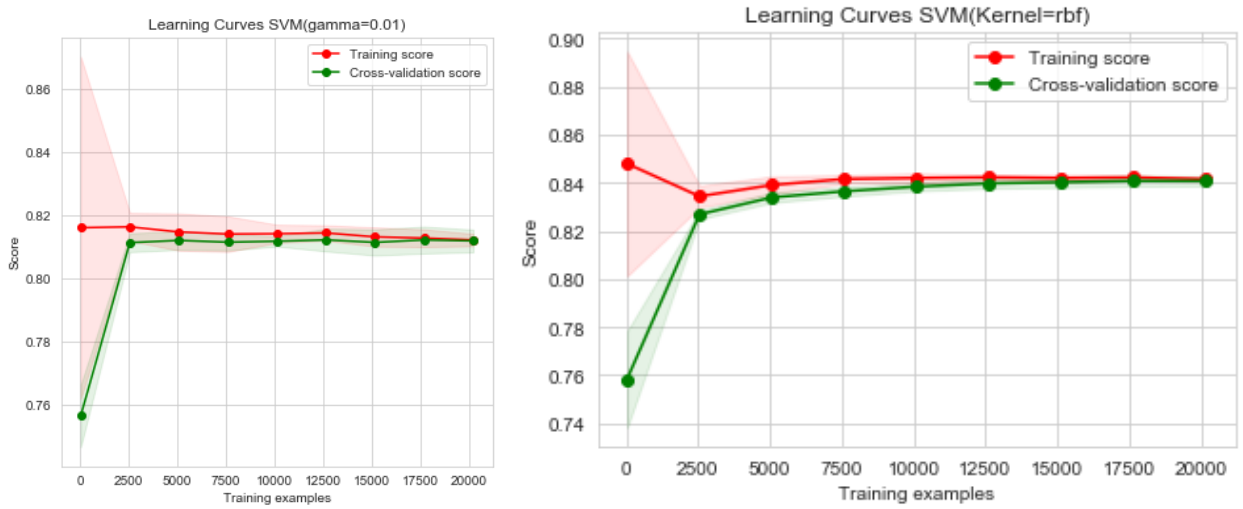


Figure : Linear SVM learning curve (left) and 'rbf' SVM learning curve (right)

6.4. Deep Neural Network:

I. We focused on the effect of the number of layers and the learning rate of gradient descent. With four hidden layer, as we can see from the learning curves, learning rate 0.01 has the lowest log loss as well as optimal generalization level. With too small learning rate, optimizer failed to optimize the learning. Potentially it failed to find the universal minima. Similarly, too big step also caused overshoot in the optimization process and the loss is too high. It also failed to optimize learning.

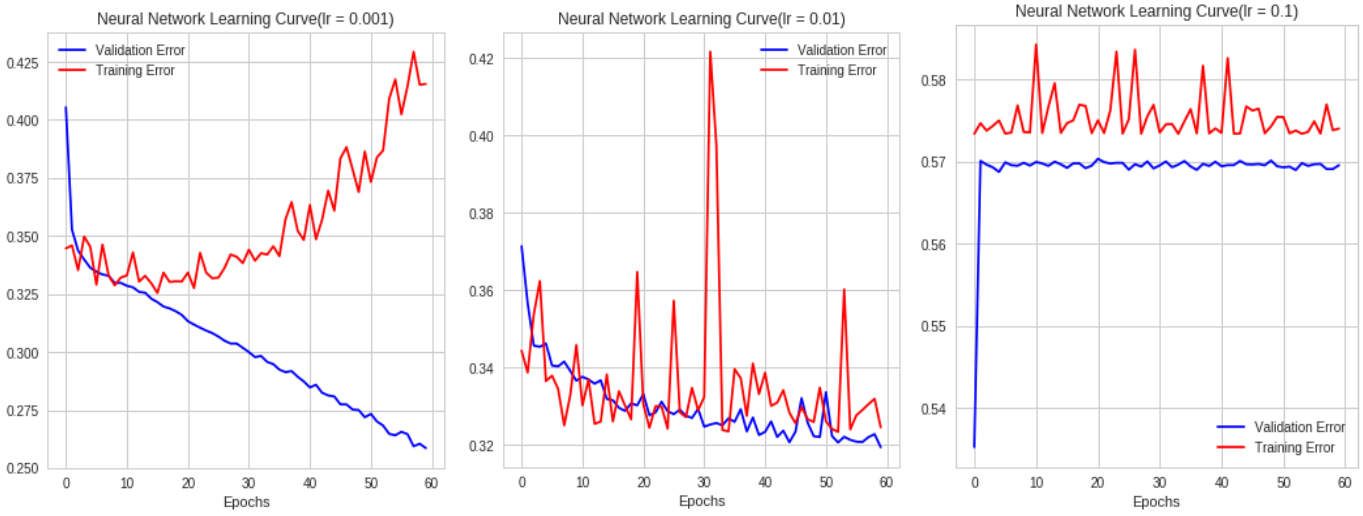


Figure: Log loss vs iteration curve of Deep neural network for different learning rate

II. Effect of number of layers: Log loss score for 2, 3, 4, and 5 hidden layers respectively 0.334, 0.339, 0.329, 0.339 on the validation data. Therefore, we choose to use 4 hidden layers for final modeling. Final train

accuracy about 84.8% and test accuracy 84.3 %, which is better than all other models. It is also very time efficient (37s training time).

Model Description	Train Accuracy (%)	Test Accuracy (%)	Wall time
Default Decision trees	99.99	81.44	0.1s
Tuned Decision trees (max_depth =9, min_samples_split=0.001)	85.86	82.83	5.7s
Adaboost (Tuned Decision trees, estimators=50, learning_rate =1)	98.60	81.30	32.9s
Adaboost(Tuned Decision Trees, estimators=300, learning_rate=0.01)	91.60	83.71	~3min
Default KNN classifier	87.17	81.74	0.3s
Tuned KNN classifier (n_neighbors =23)	84.43	82.93	~4min
SVM Linear Kernel (gamma = 0.001, C=1)(cross-validated)	81.25	81.05	~4mins
SVM rbf Kernel (gamma=0.1, C=10)(cross-validated)	84.29	84.19	~4mins
Deep Neural Network(3 hidden layers with 100 nodes at each)	84.75	84.30	37s

Table: Brief overview of different experiments and corresponding results

---

Second Case Study :

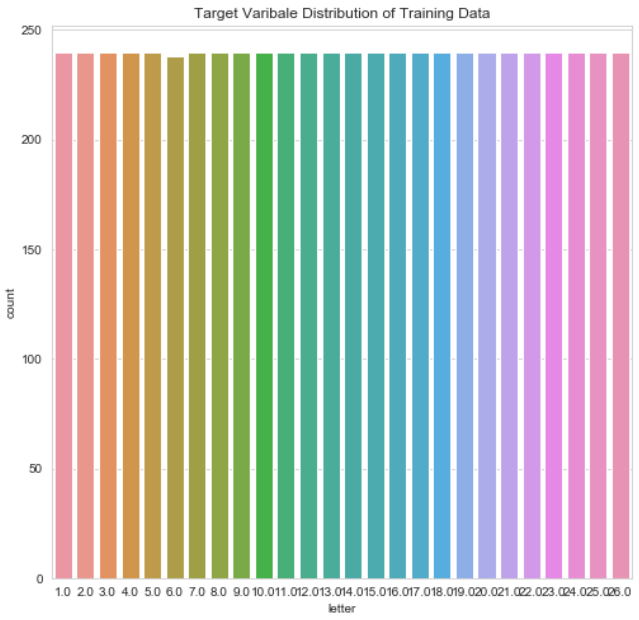
Problem Definition: From spectral coefficients; contour features, sonorant features, pre-sonorant features, and post-sonorant features predict which letter-name was spoken

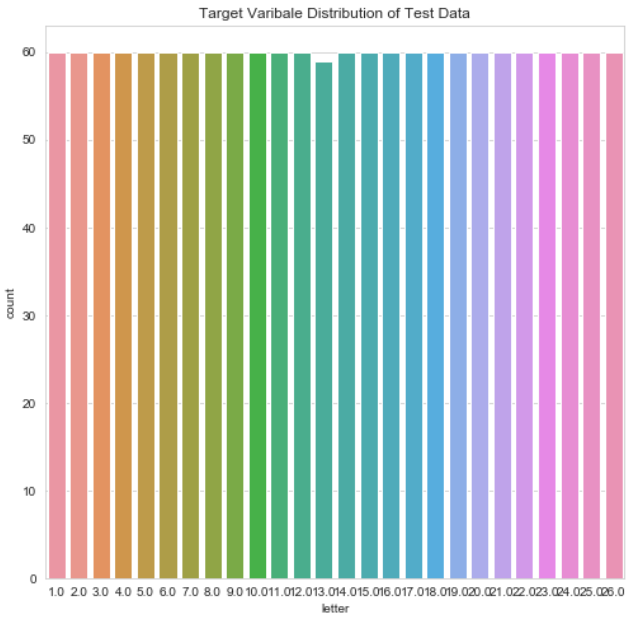
Data: For this classification task the ISOLET dataset is used from UCI repository. (source: http://archive.ics.uci.edu/ml/datasets/ISOLET)

EDA and Interesting Points:

Overall data is well balanced as indicated in the data description at the source. It is well balanced. It has 6238 training instances with 26 classes. Each class has about 240 instances except only 1 class with 238. Testing data, like-wise, well balanced and free from noises. It has 1559 instances with same 26 classes and each class has about 60 instances except one with 59. **Since this data set requires multi-class classifier, it would be interesting to observe different algorithm how they perform. Another interesting feature is that all the feature-variables are continuous, unlike the first dataset. It will also show how these algorithms behaves at differently when the dataset is balanced.**

Name: Zahiduzzaman Biswas
GTID: zbiswas3





Observations & Analysis:

Decision Trees:

1. **Default version with no depth limitation (high complexity)**
    - **c.** Result : training accuracy 100% and test accuracy80.03% (**Overfitting)**
    - **d.** Due to the high complexity and no restriction on branching, trees don't generalize well. Just like the first case.
    - **e.** We cross validated with grid search on max_depth and min_samples_split to improve.

2. Pruning: max_depth = 1 (min_sample_split = 0.001)
    - f. Overall accuracy for both training and validation set are also very low. These models suffer from Underfitting due to the High bias.
3. Pruning: max_depth = 14
    - a. This model suffers from the Overfitting due to the high complexity of the trees.

Name: Zahiduzzaman Biswas
GTID: zbiswas3

4. Pruning: max_depth = 9
    a. At this depth the training accuracy drops close to 90% and then plateaus. Similarly, the validation data accuracy increases over the data size. Eventually it reaches close to the convergence. We performed k-fold cross validation with parameter grids and found it to be the optimal choice.
5. **Best Model with max_depth = 9 and min_samples_split = 0.001**
    a. With optimal max_depth = 9 and min_samples_split=0.001 training accuracy dropped to 85.02% and test accuracy improved to 79.5%. The overfitting is reduced significantly, because of the controlled branching and less complexity.

6.3. Boosting with Adaboost classifier:

III. Default Adaboost classifier(n_estimator=50, lr=1) on the pruned Decisiton Trees

- **a.** Train accuracy is 0. 883 and test accuracy is 0.793 (Highly overfitted model)
- **b.** Potential Cause of Overfitting: **High dimensionality and continuous variables. None of the variables are categorical. This issues can be handled using SMOTE type algorithm**

IV. Hyper-parameter tuned Adaboost (n_estimator=250, learning rate=0.01)

- **a.** Training score 99% and testing improved to 94% because of proper learning rate and estimator size which used probability estimator to improve the result.

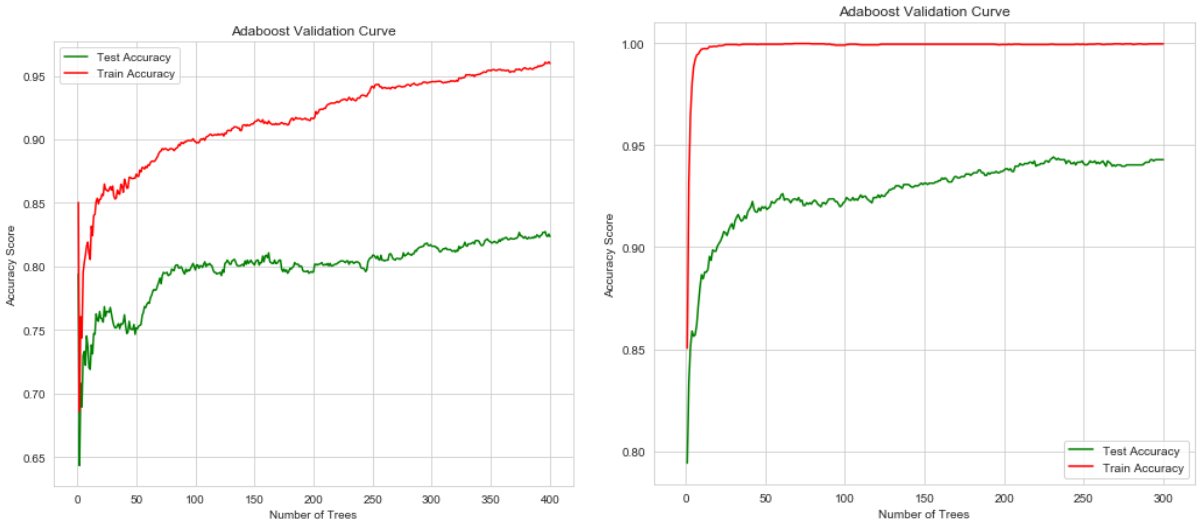


Figure: Learning curve for Adaboost with learning rate 1 and 0.1

6.4. KNN classifier:

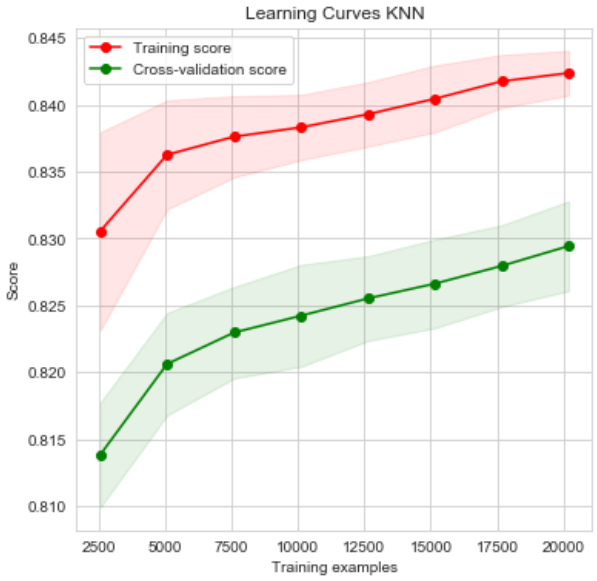
III. Default KNN classifier with k-value =1:

- **a.** Training score is 0.933 and test score 0.913. Which is a good accuracy score.
- **b.** Unlike the first case, this data set is well-balanced. Despite having high dimensions, KNN performs great. Even after grid search cross validation the score stayed same.

IV. Hyper-parameter Tuned KNN (Number of Neighbors =5)

- **a.** Accuracy score remained unchanged, since the distance metrics is not affected by the number of neighbors in this case. Chances of bias is bias is very low. Therefore, both model perform the same. As we can see in the learning curve the both curves are parallel.

Name: Zahiduzzaman Biswas
GTID: zbiswas3



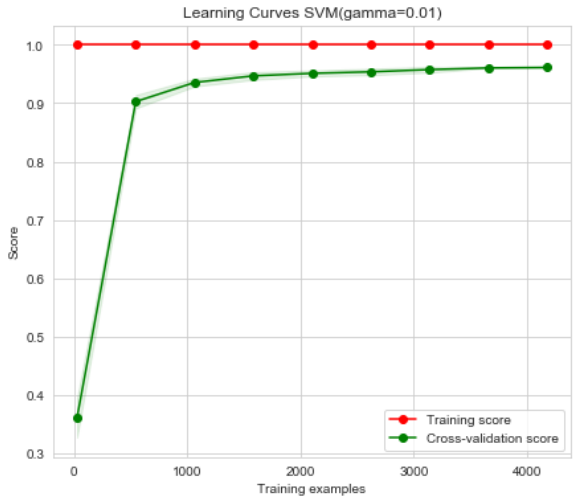
6.4. SVM classifier:

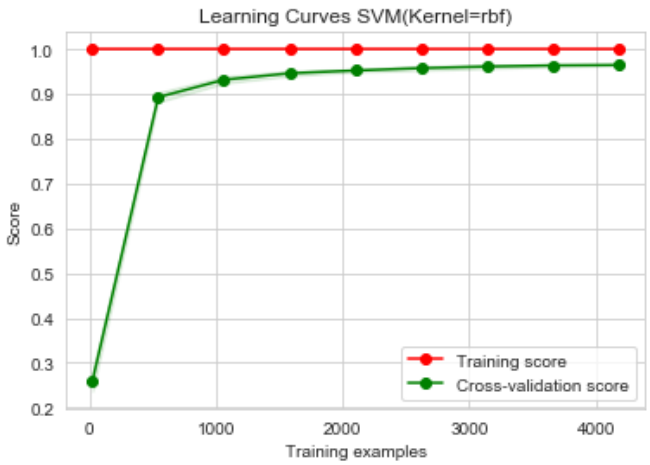
III. SVM model with Linear kernel (gamma=0.001, C=1)

- **a.** Through grid search cross validation on Linear kernel we found optimal gamma =0.001 and C=1 just like the first case.
- **b.** However, this time Train score 100% and test score about 96% which are excellent. The model generalized the learning well. It does not overfit the data.
- **c.** Since it takes a subset of data for classification, with balanced data it performed pretty well.

IV. SVM model with Linear kernel (gamma=0.1, C=10)

- **a.** For rbf kernel too, we performed the grid search cross validation. Interestingly, the hyperparameters are much different from linear kernel. gamma is 0.1 (>0.001) and C=10(>1) just like the first case.
- **b.** Model performance is also better. Training score 100% and test score about 97% which is even better.
- **c.** Since the data is balanced and scaled, the decision boundaries are formed well. The support vector distance metrics worked efficiently and faster than the first case. (Result table includes the time comparison).





Name: Zahiduzzaman Biswas
GTID: zbiswas3

Figure : Linear SVM learning curve (left) and 'rbf' SVM learning curve (right)

6.4. Deep Neural Network:

III. We focused on the effect of the number of layers and the learning rate of gradient descent this time too. With four hidden layer, as we can see from the learning curves, learning rate 0.01 has the lowest log loss as well as optimal generalization level. With too small learning rate, optimizer failed to optimize the learning. Potentially it failed to find the universal minima. Similarly, too big step also caused overshoot in the optimization process and the loss is too high. It also failed to optimize learning.

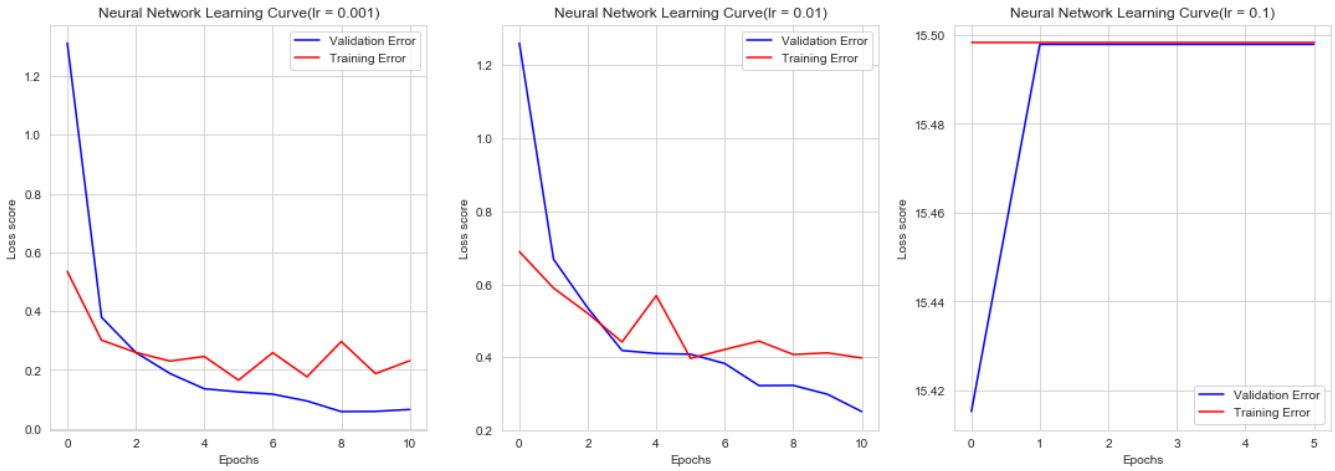


Figure: Log loss vs iteration curve of Deep neural network for different learning rate

IV. Due to time constraint we could not find the best result. According the data site it should be close to 96% on the test data. However we found it close to 70% as it is seen in the learning curve with four layer and 0.01 learning rate.

Model Description	Train Accuracy (%)	Test Accuracy (%)	Wall time
Default Decision trees	100	80.3	3s
Tuned Decision trees (max_depth =9, min_samples_split=0.01)	85.02	79.5	45.7s
Adaboost (Tuned Decision trees, estimators=50, learning_rate =1)	88.3	79.3	~2mins
Adaboost(Tuned Decision Trees, estimators=250, learning_rate=0.01)	99	94	~16min
Default KNN classifier(n_neighbors =1)	93.3	91.3	0.3s
Tuned KNN classifier (n_neighbors =5)(no change in overall result)	93.3	91.3	~2min
SVM Linear Kernel (gamma = 0.001, C=1)(cross-validated)	100	96.02	~1mins
SVM rbf Kernel (gamma=0.1, C=10)(cross-validated)	100	96.9	~4mins
Deep Neural Network(3 hidden layers with 100 nodes at each)	(100)	(96)	

Table: Brief overview of different experiments and corresponding results

Name: Zahiduzzaman Biswas
GTID: zbiswas3

References

[1] Blake, C. L. and C. J. Merz (1998). UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA. Available at http://www.ics.uci.edu/~mlearn/MLRepository.html

[2] M. Fanty and R. Cole (1998). UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA. Available at http://archive.ics.uci.edu/ml/datasets/ISOLET

[3] Sci-kit Learn Library Website, https://scikit-learn.org/stable/index.html