Assignment 1: CS7641 - Machine Learning Yogesh Edekar(GTUID – yedekar3)

February 20, 2021

1. Introduction

As part of assignment 1 I am trying to apply the Supervised learning approach to two datasets in order to prepare a model to solve the classification problem. The first classification problems include segregating the bank clients who open a term deposit account at the bank from the ones who do not open a term deposit account at the bank (sharanmk, 2020). The second classification problem includes segregating the working professionals into a group that earns <= $50,000 as salary from the one that earn > $50,000 salary (lodetomasi, 2020). The algorithms implemented for this analysis are decision trees, artificial neural networks, boosting (ADA boosting), support vector machines and K nearest neighbors.

2 Data Sets

The data sets used for analyzing the 5 classification algorithms are a) Term deposit dataset with 17 features identifying a successful term deposit at   downloaded from Kaggle and b) income evaluation dataset with 14 features that are impacting the income of a person also downloaded from Kaggle.

2.1 Term deposit dataset

The term deposit data set is pretty much balanced dataset with 5873 people of the selected population are not opening the term deposit at the bank and 5289 people of the selected population are opening the term deposit at the bank. As mentioned above Term deposit dataset features are divided as 9 features having categorical data and 6 features having numerical data. After analyzing the data due to presence of categorical values data cleansing was performed to prepare the data to be used for modelling.

2.2 Income evaluation dataset

The income evaluation dataset is not as balanced as term deposit dataset where 76% of the samples fall in the <= $50,000 category and the 24% samples fall in the > $50,000 category. However the dataset is not severely imbalanced so no oversampling and under sampling methods are applied in order to tune the data set. Label encoding is used over this dataset to transform the String data into corresponding numeric categories.

3 Implementation

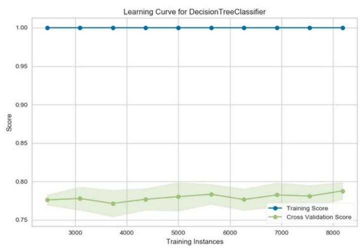
All the 5 machine learning algorithms mentioned above are implemented using scikit-learn module available in python. In order to make classification easier and more intuitive the output class labels are changes from <=50k and >50k to 0 and 1 for the income evaluation dataset and yes and no labels of deposit column in the term deposit dataset are changed as well to 1 and 0 respectively. In all the cases cross fold validations of 10 folds are used to determine the fit of the model and based on the learning curve for baselined model corresponding hyper parameters were tuned to see where the model performance improves towards achieving a good bias- variance tradeoff.

3.1 Term Deposit Data set

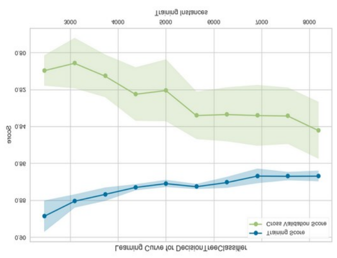
3.1.1 Decision Tree with pruning

DecisionTreeClassifier class of scikit learn is used for analyzing and implementing the decision tree algorithm for analysis of term deposit prediction problem. Baseline accuracy is first obtained for the given data and then grid search is performed multiple times to identify the correct hyperparameter to be used for tuning.

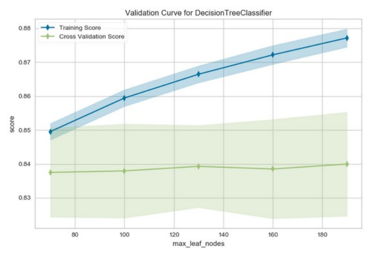
The learning curves and validation curves obtained for baseline and tuned version of the decision tree are as follows. The blue line represents training data whereas green line represents cross validation data.



Baseline Learning curve



Tuning learning curve



Validation curve

As seen from the learning curves the untuned decision tree is completely overfitting the training data and performs poorly on the unknown data indicating the model is not generalized to fit to any data. This indicates high variance in the model. Using the max_leaf_nodes(tuned value of 130) hyperparameter the tree is pruned back so that the model becomes generalized enough to not learn the training data along with noise of it. This is shown by the converging scores after tuning in second graph. We are looking at a model which has is maintaining a good offset of variance and bias since the scores are not too high as well as not too low. The accuracy changes post tuning can be shown in the below tables.

The validation curve also emphasizes the fact that at max_leaf_nodes = 130 the cross validation score matches the most with the training score and after that the curves start moving away from each other causing high variance indicating adding more data does not improve any learning capping the accuracy at 0.84.

Baseline Scores

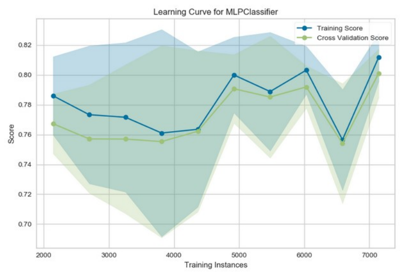
Label	Precision	Recall	F1 score	accuracy
0	0.81	0.80	0.81	0.79
1	0.78	0.79	0.78	

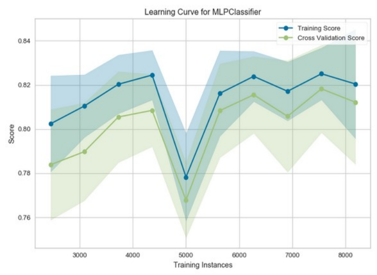
Tuned Scores

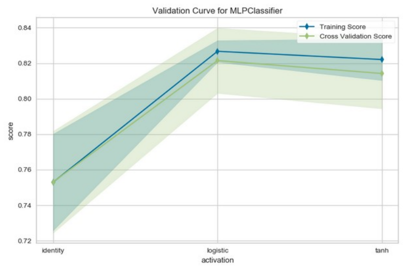
Label	Precision	Recall	F1 score	accuracy
0	0.87	0.81	0.84	0.84
1	0.80	0.87	0.84	

3.1.2 Neural Networks

The MLPClassifier class of scikit learn library is used for analyzing and implementing the multi layer neural network algorithm for analysis of term deposit prediction problem. Baseline accuracy is first obtained for the given data and then grid search is performed multiple times to identify the correct hyperparameter to be used for tuning. *The learning curves and validation curves obtained for baseline and tuned version of the multi layer neural network are as follows. The blue line represents training data whereas green line represents cross validation data.*







Baseline Tuned Validation Curve

As we can see from the graphs above the baseline accuracy score of the neural networks is 0.78. After tuning the score of the model comes out to be around 0.82 which is a 4% gain. By performing grid search for the parameters I have selected the activation function as the hyperparameter to be used for tuning. The logistic or sigmoid function is the hyperparameter chosen that is used for firing a Perceptron to next layer with default 100 layers. Due to the non linear nature of the function small changes in input are captured as well and very clear predictions are made. This becomes evident from the graph of the function that the accuracy is increased from 0.78 to 0.82. One very interesting fact to be observed here is upon increasing the training instances the validation scores converges with the training score indicating a good fit and then it starts to maintain a fixed distance. That indicates maximum learning accomplished and nothing more can be learned from the data. This indicates that the model has generalized to sufficient degree and it is neither overfitting severely not is it underfitting. Below are the classification reports for the baselined and tuned version of the neural network.

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.85	0.69	0.76	0.78
1	0.72	0.87	0.78	

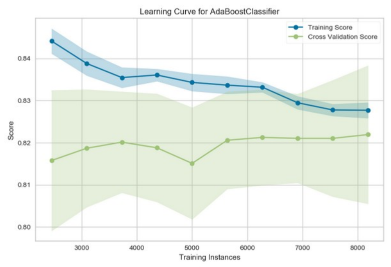
Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.85	0.81	0.83	0.82
1	0.80	0.84	0.82	

Confusion Matrix for tuned data

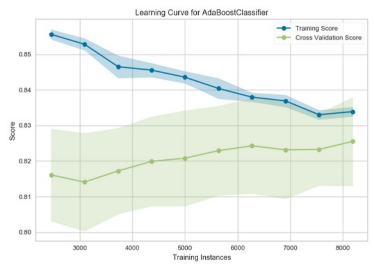
```
[[940 227]

 [167 899]]
```

3.1.3 ADA (Adaptive) Boosting

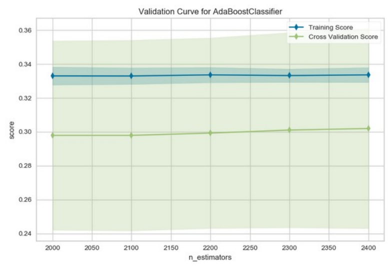
The *AdaBoostClassifier* of scikit learn is used for analyzing and implementing the boosting algorithms for analysis of the term deposit prediction problem. Like all other problems a baseline score is found first and then later grid searching is used to identify correct hyper parameters for tuning. In case of ADA boosting we had to resort to 2 most important hyper parameters n_estimators and learning rate. The learning curve and validation curves for ADA boosting are as follows. *The blue line represents training data whereas green line represents cross validation data.*



Baseline learning curve



Tuned learning curve



Validation curve

Due to the fact that ADA boosting uses stumps with extremely low height of a single level decision tree we can see that the baselined score of the ADA boosting algorithm itself is around 0.82 which is pretty much closer to the tuned Decision Tree score. However unlike the decision tree baselined score ADS boosting avoid overfitting due to the fact that it is an ensemble of bad learners and each learner picks up from the wrongly classified records of the previous learner and improves slightly upon the error. This fact clearly indicates that we can increase the number of learners in the ensemble to improve the performance of the algorithm. With the grid searching the optimum number of learners that can be used for boosting come out to be 2200 learners. With 2200 learners each improving upon the mistakes from the previous learner we can see and improvement of 2% from the baselined score. Also upon looking at the learning curve we can see that the validation dataset does not converge smoothly for the baselined case, however for the tuned learning curve the validation dataset converges smoothly indicating a good fit and appropriate generalization. Since both the training and validation curves seem to converge very well with any signs of overfitting or underfitting we can say the ADA boost model neither overfits nor underfits the data. This indicates that the data used for training the model contains features that dominate the output which we will discuss below in correspondence with this dataset. The classification reports for this algorithm is shown as follows.

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.81	0.86	0.83	0.82
1	0.84	0.77	0.80	

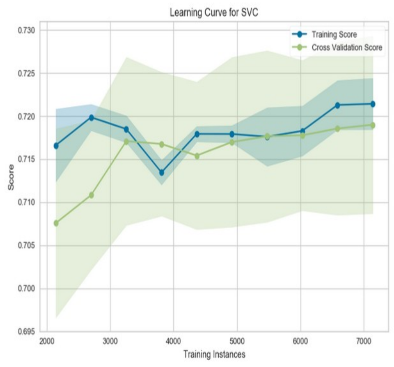
Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.83	0.85	0.84	0.84
1	0.84	0.82	0.83	

Confusion matrix for tuned data

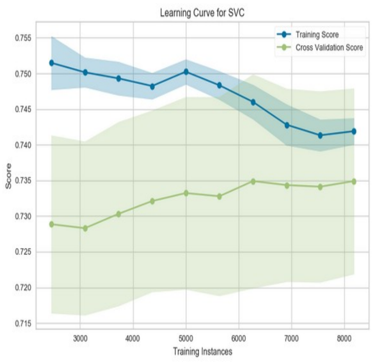
```
[[984 168]

 [199 882]]
```

3.1.4 Support Vector Machines (SVM)

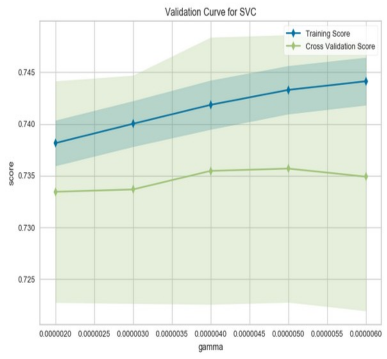
The *svm* class of scikit learn is used for analyzing and implementing the support vector machines algorithm for analysis of the term deposit prediction problem. Like all other problems a baseline score is found first and then later grid searching is used to identify correct hyper parameters for tuning. In case of support vector machines I resort onto the hyper parameter gamma for tunning the kernel in order to obtain the optimal result since RBF kernel is used in the model. Gamma works approximately like k in the KNN model.



Baseline



Tuned



Validation curve

As the above learning curves depict the Support Vector machines are underfitting the data with the baseline score of around 0.73 which upon tuning changes to 0.74. This indicates high bias in the model and suggests to basically tweak the model to be more complicated nature so that the underfitting can be handled successfully. As we can see that in spite of tuning the mode the difference between training and cross validation dataset does not reduce completely and at no point they are converging with each other. Even after increasing the training instance the cross validation curve plateaus and can not learn any more from the data and hence the model underfits the data. We have tried to tune the model using a low value of gamma thus opting for high curvature. As indicated by the validation curve the optimum score is obtained at gamma = 0.000004. The classification report is given as follows

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.70	0.84	0.76	0.73
1	0.78	0.60	0.68	

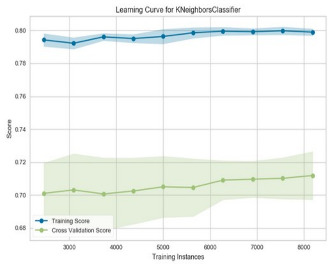
Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.74	0.78	0.76	0.74
1	0.74	0.69	0.72	

Confusion Matrix

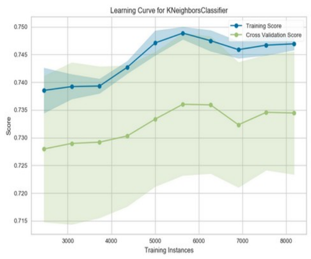
```
[[916 260]
 [323 734]]
```

3.1.5 K Nearest Neighbors (KNN)

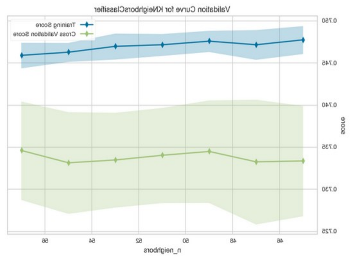
The *KNeighborsClassifier* class of scikit is used for analyzing and implementing the K nearest neighbors algorithm for analysis of the term deposit prediction problem. To being with we do not configure the class at all with any values and take the baseline performance of the KNN algorithm with all the default values. We resort to grid to find out the optimum value of K to figure our exactly how many neighboring points are needed to correctly predict the value of the output of the test dataset. The learning and validation curves for K nearest neighbors are specified as follows.



Baseline Learning curve



Tuned Learning Curve


Validation Curve

As the baseline curve indicates we can see that there is a huge difference in the training score and the cross validation score of the model. This indicates a case of very high variance. Since the model seems to be fitting to the training data we can say that this is the case of overfitting. Also the cross validation score is not converging to the training dataset and it simply plateaus. This indicates that even adding more data will not help with the training and thus the data used for term deposit can not be analyzed effectively with the KNN algorithm. After grid searching with the optimum value of k =31 this situation does not change thus confirming the fact that the data is not suitable to be analyzed by the KNN algorithm. A wide difference in the training and cross validation scores for the validation curve also indicates the similar fact. The classification report for the model is given a follows.

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.72	0.74	0.73	0.71
1	0.70	0.68	0.69	

Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.74	0.79	0.76	0.75
1	0.75	0.69	0.72	

Confusion Matrix

[[925 240]

 [329 739]]

3.1.6 Conclusion

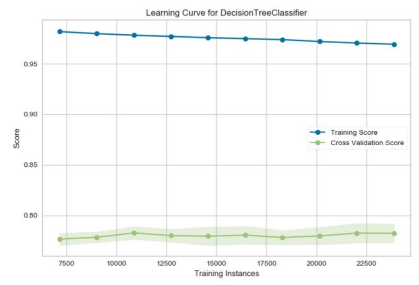
Since the term deposit dataset is a balanced dataset we can use accuracy as a metric for evaluating the algorithms for the dataset. Looking at the accuracy metric for the dataset we can say that the term deposit dataset can be trained and predicted well using the ADA Boost algorithm as well as the Decision Tree algorithm. The Ada boost and Decision Tree algorithms both converge in 15 seconds while the other algorithms take much longer time to converge. This clarifies the fact that the dataset fits the decision tree algorithms and hence it must be having certain features that are important than other to predict the output. The failure of fitting the data to KNN algorithm and SVM algorithm stress this fact since KNN is biased towards the uniform weight of all the points.

Algorithm	Decision Tree	ADA boost	Neural networks	KNN	SVM
Time (seconds)	15	650	244	431	557
Accuracy	0.84	0.84	0.82	0.75	0.74

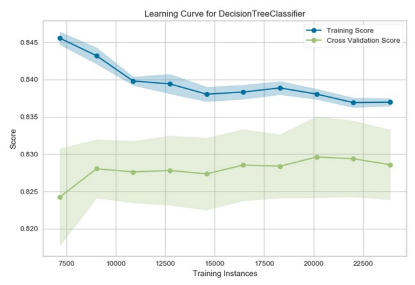
3.2 Income Evaluation Dataset

3.2.1 Decision tree with pruning

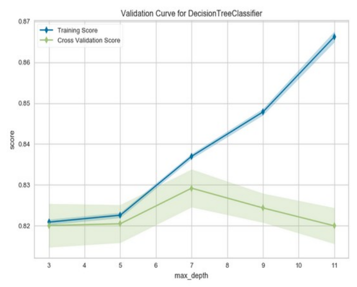
DecisionTreeClassifier class of scikit learn is used for analyzing and implementing the decision tree algorithm for analysis of income evaluation problem. Baseline accuracy is first obtained for income evaluation dataset and then grid search is performed to obtain optimum value for the hyper parameter max_tree_depth in order to prune the tree. Restricting the tree depth allows us to reduce the bias obtained by creating deep tree with smaller and smaller number of leaves with the minimum number of leaves to be 1. This pruning thus results in more generalization of the decision tree and hence it can fit to unseen data as well in order to avoid overfitting to the unseen data. The learning cure and validation curve for the Decision Tree are as follows.



Baseline learning curve



Tuned learning curve



Validation curve

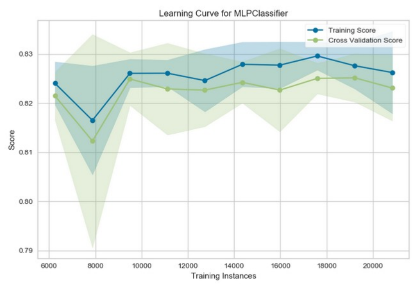
In the baseline dataset it is clearly visible that the tree is overfitting to the training data. The validation data simply flattens and there is no learning in spite of increasing the training instances. This indicates the decision tree can not handle unseen data very well and can not be generalized. After tunning the max_tree_depth parameter we can see that the variance between the training and validation data has reduced however the training and validation scores do not converge and again start to move away form each other indicating a high variance and less learning. The validation curve gives us an optimal point of maximum tree depth of 7 since at that point the training and validation curves are very close to each other thus marking the accuracy of the tree to be 0.83. The classification report for the decision tree is given as follows

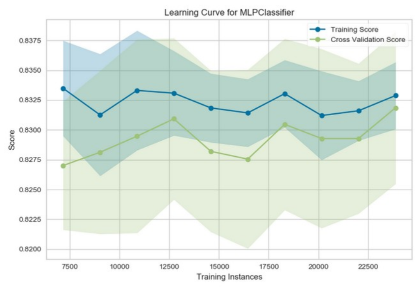
Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.85	0.86	0.85	0.78
1	0.53	0.52	0.53	

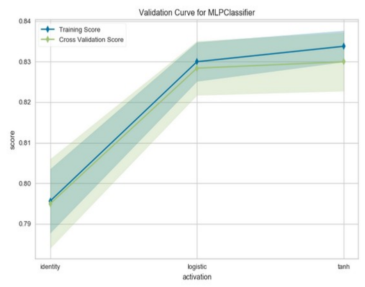
Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.87	0.92	0.89	0.83
1	0.68	0.55	0.61	

3.2.2 Neural networks

The MLPClassifier class of scikit learn library is used for analyzing and implementing the multi layer neural network algorithm for analysis of income evaluation problem. Gris search is performed to identify the hyper parameter to be used for tuning the model for optimum performance. The parameters considered for tuning are momentum, activation and alpha. However we obtained efficient results with activation function again and the sigmoid activation function was chosen for tuning the model. The learning curve, validation curve and classification report for the same are given as follows.







Baseline learning curve　　　Tuned learning curve　　　validation curve

The baseline learning curve shows that the validation score follows the training score parallelly and when the data is increased they almost converge but again diverge and this keeps on happening until we get to see the curves to be parallel to each other. Due to this lack of a certain trend to fit the curve we can say that the data is not fit properly with the baselined version of the neural network. Upon tuning with the sigmoid function we can see that a specific pattern emerges and after increasing the training instances the validation score begins converge with the training score. This curve does not indicate high variance or high bias and we can say the bias variance tradeoff seems to be maintained. This behavior is supported by the trends displayed in the validation curve as well where the scores are near to each other at the sigmoid(logistic) function and then they begin to diverge indicating that adding further training instance would not help the model.

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.84	0.95	0.89	0.83
1	0.73	0.45	0.56	

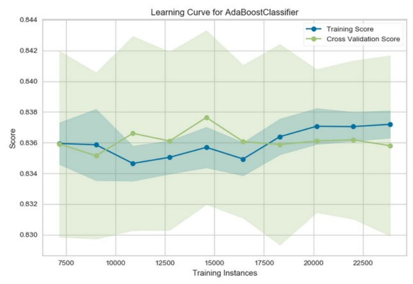
Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.86	0.94	0.90	0.84
1	0.71	0.51	0.60	

Tuned Data Confusion Matrix

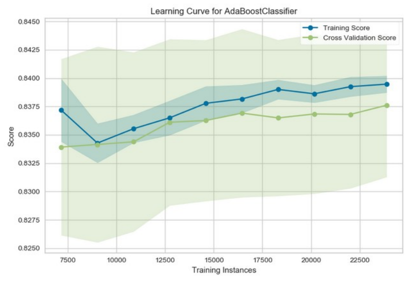
```
[[4661  314]

 [ 751  787]]
```

3.2.3 ADA (Adaptive) Boosting

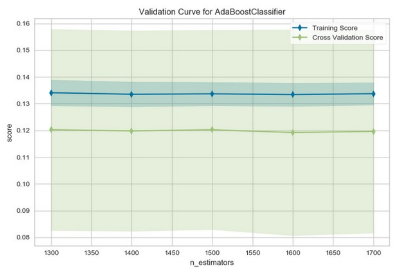
The *AdaBoostClassifier* of scikit learn is used for analyzing and implementing the boosting algorithms for analysis of the income evaluation problem. We perform baselined fitting first with 10 fold cross validation and then perform hyper parameter tuning to enhance the efficiency of the model. We use the random state to ensure that the randomization is done similarly every time for predictable performance. The learning curve for baseline and tuned model as well as the validation curve for the ADA boosting algorithm are shown as follows.



Baselined performance



Tuned performance



Validation Curve

The baseline performance indicates that the training and testing scores are converging with each other before flattening. The model predicts very well and surpassed the training score for new data when the training instances are low. This continues until more and more data is added and finally we can see the testing performance going down and it starts moving away from the training score. This is a really interesting observation throughout the whole experiment where the model fits to the new data even better than the training data. This indicates the power of bagging where several weak learners counter each other’s weaknesses causing the model to perform so well. This can further be improved by adding more weak learners and hence increasing the count of estimators to be 1500. However adding more estimators even though improves the performance it makes the model slower. After tuning we can see that the model fits the data very well .The validation curve converges very well with the training curve but instead of sudden spikes it improves slowly and keeps on improving as the size of the data increases thus improving the generalization of the model when more and more data is provided, this indicates low variance for the model. Same observation is stressed upon by the validation curve where at the optimal sample count of 1500 we can see the maximum validation score which starts to go down later on indicating model has learnt maximum from the data. Classification report and confusion matrix can be given as follows

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.87	0.92	0.89	0.83
1	0.69	0.67	0.62	

Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.87	0.92	0.90	0.84
1	0.70	0.58	0.63	

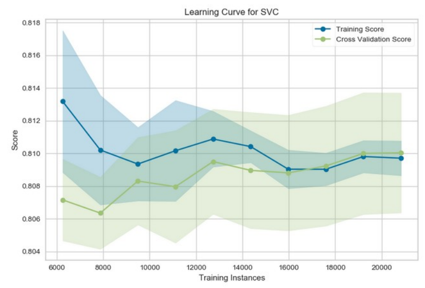
Tuned data confusion matrix

[[4528 413]

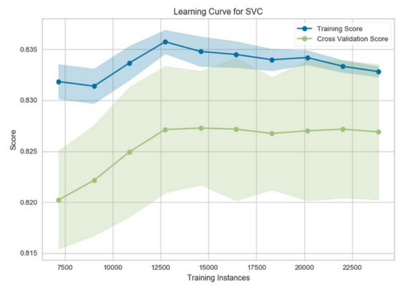
[654 918]]

3.2.4 Support Vector Machines (SVM)

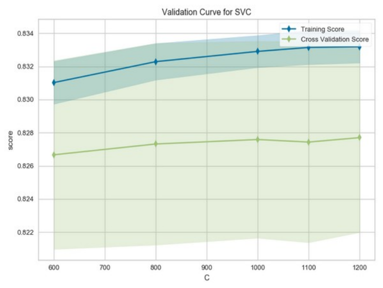
The *svm* class of scikit learn is used for analyzing and implementing the support vector machines algorithm for analysis of the term income evaluation problem. During the hyper parameter tunning process for the income dataset a different hyper parameter C the error control parameter is used for tuning the model. After grid search the value of C is found out to be 1000 which essentially indicates that the data has lot of error points. What it means in terms of planes the data is very mixed data where it is very difficult to find out the a clear distinction between the features resulting into 2 classes. With that in mind let us analyze the learning curve and validation curve of the model to get more insights about the model.



Baseline



Tuned



Validation Curve

The baseline performance of the SVM model indicates a very good fit for the data. When we look at the validation curve and training curve we can see that the gap between validation curve and training curve is very small this this indicates that the model generalizes the unseen data very well and in not only bound to the training data. This indicates that the model is not overfitting. Also there scores for training and cross validation are not overly low and they are not very different from each other which indicates that the model is neither overfitting nor underfitting. Upon adding more data the validation score eventually passes the training score indicating model’s capability to predict the unseen data very well. However all this changes upon tuning the model on the error parameter C. When model is tuned the accuracy is increased but now model has high bias which can be seen by the large distance between the cross validation and the training scores. This indicates that a different hyper parameter needs to be used for tuning the model to ensure that the model is tuned in optimum capacity. Same fact is illustrated by the validation curve as well which simply keeps going flat even with increase training instances indicating that the parameter C should not be used for tuning the model. The classification report can be given as follows.

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.82	0.95	0.88	0.80
1	0.69	0.35	0.46	

Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.89	0. 87	0.90	0.83
1	0.70	0.54	0.63	

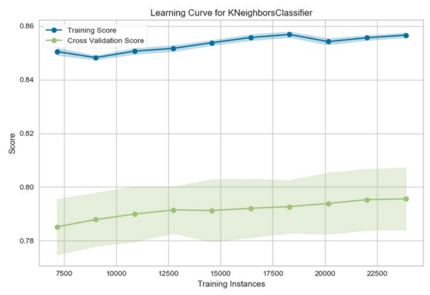
Tuned data confusion matrix

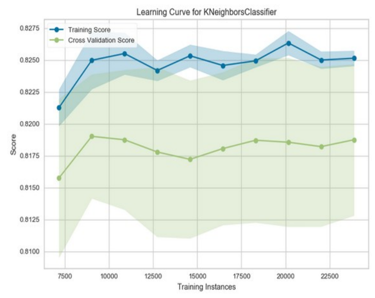
[[4660 250]

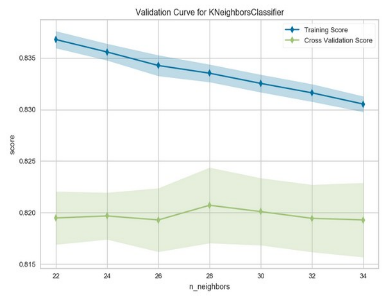
 [1048 555]]

3.2.5 K Nearest neighbors (KNN)

The *KNeighborsClassifier* class of scikit is used for analyzing and implementing the K nearest neighbors algorithm for analysis of the income evaluation problem. We begin with the default values of hyper parameters to obtain a baseline performance and later on we tune the number of neighbors K with the distance metric of Manhattan distance. This specific matrix is chosen since the we have different dimensions available in the input and in general the KNN algorithm suffers from the **curse of dimension**. The learning curve and validation curves can be shown as follows.







The baseline curve indicates huge difference between the training and the validation scores amounting to very high variance and overfitting. After choosing optimum k = 72 the score has gone up but the overfitting of the model has not changed. There is a constant difference between the validation and the training scores and the gap does not seems to go away no matter however training instances are added. This indicates that the model can not learn any more from the data and will be severely overfitting with a very high variance. This model can not be generalized when trained with the income evaluation data. The classification report and the confusion matrix can be given as follows.

Baseline				
Label	Precision	Recall	F1 score	accuracy
0	0.86	0.87	0.87	0.79
1	0.58	0.54	0.56	

Tuned				
Label	Precision	Recall	F1 score	accuracy
0	0.86	0.91	0.89	0.82
1	0.66	0.54	0.60	

Tuned Data Confusion Matrix

[[4512 435]

 [713 853]]

3.2.6 Conclusion

Upon analyzing the income evaluation dataset with the above 5 algorithms we can see that the data fits appropriately using Neural networks and ADA boosting. However when we take clock time as well in consideration neural network become a clear winner as the large number of stumps for ADA boosting take a lot more time to converge. On the closing notes we can say that while handling balanced datasets

Decision tree and Boosting approaches win in terms of accuracy where as we can say the oversimplicity and aggressive pruning of ADA boosting results in more iterations of the model making it time consuming to predict.

Time and accuracy chart for the income evaluation dataset

Algorithm	Decision Tree	ADA boost	Neural networks	KNN	SVM
Time (seconds)	14	1957	731	715	734
Accuracy	0.83	0.84	0.84	0.83	0.82

When we look at the accuracy and clock times for the 2 chosen datasets we can say that decision trees perform equally well irrespective of the size of the dataset. However the iterative models like Neural networks and KNN take more time for larger datasets. Balanced datasets are handled well by decision trees. However neural networks do not seem to be biased about whether a dataset is balanced or not.

3.3 References

https://www.scikit-yb.org/en/latest/api/model_selection/learning_curve.html

https://www.scikit-yb.org/en/latest/api/model_selection/validation_curve.html

https://www.vebuso.com/2020/03/svm-hyperparameter-tuning-using-gridsearchcv/

https://www.ritchieng.com/machinelearning-learning-curve/

https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680