# CS7641: Machine Learning
# Assignment 1, Supervised Learning

Martina Nacheva, GT: 903659095

**26-Sept-2021**

## Summary

This analysis applies 5 algorithms (Decision Trees, KNN, AdaBoost, SVM and Neural Networks) to a classification problem of socioeconomic data. The dataset comprises of 14 categorical and continuous features, as well as a binary target variable of income category, which is heavily imbalanced. Results show that the best performing algorithm was found to be Adaboost by a significant margin, reflecting lower bias, albeit , as high variance was also observed. In comparison, relatively poor performance was observed in KNN, likely reflecting the equal weight placed to all features and relatively low correlations observed among some with the target. Fit times show that KNN and SVC take the longest time, while DT scored the fastest runtime.

## 1.0 Dataset

The 'Census Income dataset provides socioeconomic data sourced from the 1994 Census Database. It includes 48,842 observations, a total of 14 categorical and continuous features. The target variable is binary and describes income category ( >50k or ≤50k). The dataset is interesting for several reasons:

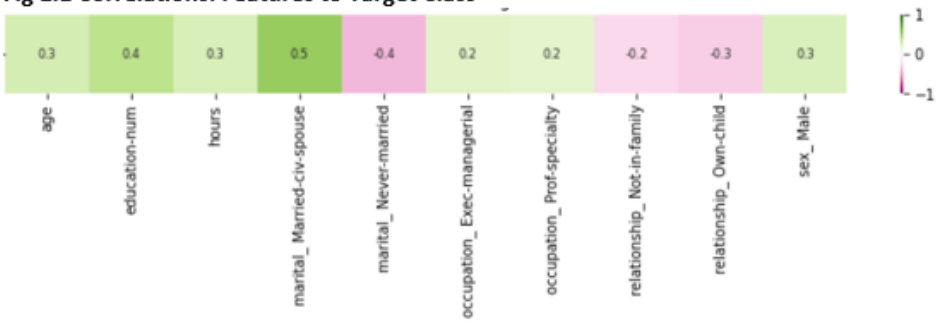
- The target is imbalanced, which will require measures to prevent the algorithms from developing a bias in favour of the majority class.

Table 1.1: Target Class Imbalance

| Income>50K | 1 | 0 |
|---|---|---|
| Share | 0.24 | 0.76 |

- Following one-hot encoding the 14 variables grow to 85. This high number of dimension can illuminate computational efficiency differences between the algorithms.
- The 85 resultant variables are correlated with the target to varying degrees. Fig 1.1 shows 10 features for which the absolute correlation exceeds 0.2. This presents an opportunity to compare the ability of algorithms to identify causal factors and classify optimally.

**Fig 1.1 Correlations: Features to Target Class**



To pre-process the dataset, train and test datasets were created using an 80%-20% ratio. The training set was then undersampled to balance the data, resulting in 18,664 training and 9,769 testing samples. K-fold cross validation was used in training with 5 folds to ensure sufficiently large validation set. The dataset is standardized.

***Performance Metric***

In light of the class imbalance in both datasets, performance will be measured using F1. Since the training sets are balanced, accuracy could be used, but would be biased in favour of the majority class in testing. F1 is the harmonic mean between precision and recall, making it suited to imbalanced data.

# 2.0 Analysis : Census Data

## 2.1 KNN

***Validation Curves***

We test the impact of varying:

- K (Fig 2.1a): At k=1, the model overfits. In training, the model considers the closest point, which is itself, resulting in F1=1. Therefore, the decision boundary is most flexible. The validation score is low because the model isn't generalizing. Any outliers in the data would have greatest distortive impact at k=1. As K increases, the decision boundary is smoothed, overfitting is reduced and validation score improves. At the optimal K=23, the model is still slightly overfitting.
- Weights (Fig 2.2b): uniform and distance (where the points are weighted by the inverse of their distance such that closer points have greater weight)

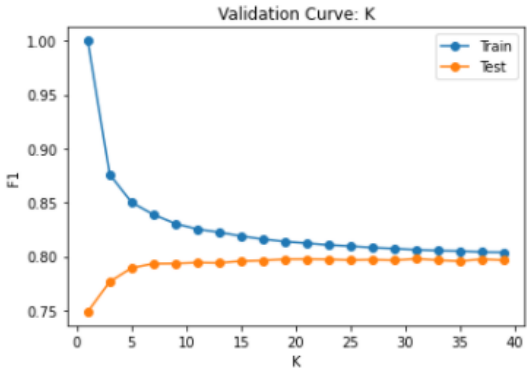
***Learning Curve***

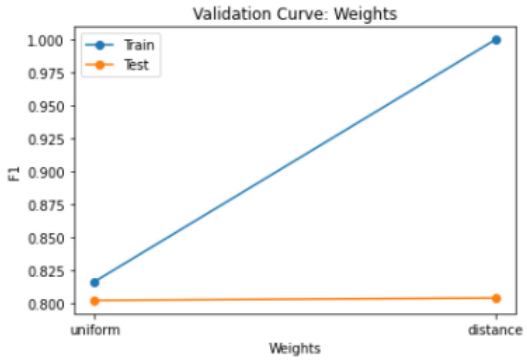
Fig 2.2c shows that the model indicates some variance error, since the model is overfitting. As the sample size increases to 6,000 this is slightly reduced. There is also indication of bias error, since the training and validation scores are not too high. This may be due to the simplicity of the algorithm, or due to lacking features. A greater sample size is unlikely to improve performance, since it would not correct the bias of the model, and signs of the variance error appear to have been corrected as the sample size grew to 6,000.

***Final Model***

The tuned model reports 0.82 F1 on the train set and 0.63 on test set. Test performance is significantly lower than validation due to differing distributions of the target between the sets. Confusion matrix shows relatively good generalization for positive class, but many misclassifications for negative class.

**Figures 2.1**



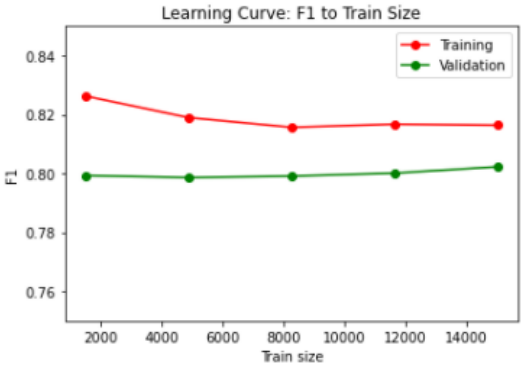




Table 2.1, Confusion Matrix

| True/ Pred. | 0 | 1 |
|---|---|---|
| 0 | 5494 | 1978 |
| 1 | 323 | 1974 |

## 2.2 Decision Tree

***Validation Curves***

Parameters tested pertain to pre-and post-pruning approaches. Additionally, the criterion was optimized to Gini.

- Max depth: Up to depth of 6, the model is underfitting, since both training and validation curves as improving (with the exception of depth=3, likely due to randomness). As the depth increases above 6, the model quickly overfits the training set until all leaves are pure. This leads the validation performance to deteriorate, as the model places greater weight on randomness and noise in the data.
- Minimum samples in a leaf: When the parameter is 1, the depth of and width of the tree increase until all leaves are pure. This process places weight on unimportant features, outliers and noise. Therefore, the validation set performance is relatively low. As the parameter increases, entropy in the leaves increases and training performance declines. Generalization is improved and validation score increases. The two sets converge at a value of approximately 80 samples, at which the model is fitted well. Lack of divergence thereafter implies that important leaves contain at least 200 samples, so changes in value have no effect.

***Learning Curve***

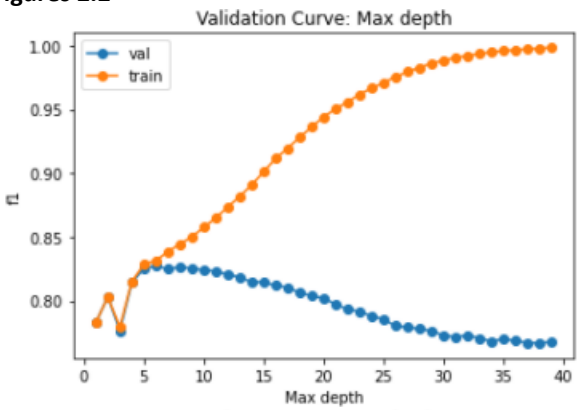
There is no indication of high variance, since the training and validation scores are very similar. Further, there is no indication of high bias since performance is improving with additional data. This indicates the model is well pruned and is neither under nor overfitted. Additional data is likely to improve generalization further based on trend.

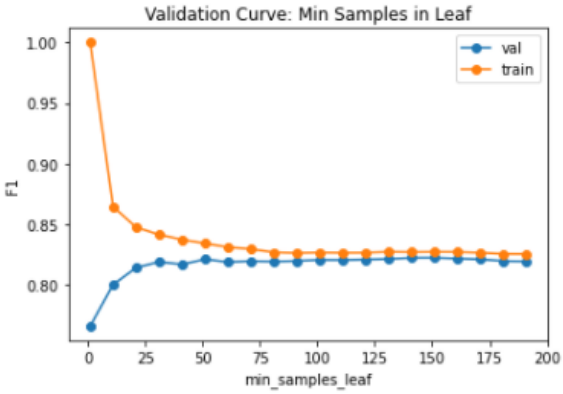
***Final Model***

Tuned model scores 0.83 on training set and 0.66 on test set. Again, this is due to differences in class distribution in test set. Among the most important features were 'marital status – Married', 'education-num' and 'capital=gain', the former two of which show the strongest correlation with the target variable (0.5 and 0.4 respectively). The

model predicts both classes more accurately than KNN, indicating the DT has generalized determinant factors better, which is to be expected (since KNN doesn't measure feature importance). However, the negative majority class is still misclassified frequently.

**Figures 2.2**



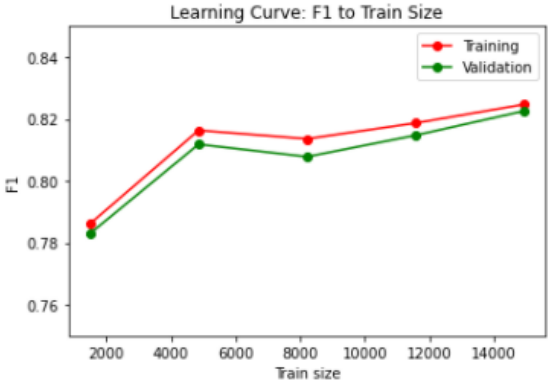




Table 2.2, Confusion Matrix

| True/ Pred. | 0 | 1 |
|---|---|---|
| 0 | 5548 | 1866 |
| 1 | 283 | 2072 |

## 2.3 SVC

***Validation Curves***

We tune the SVC by considering the regularization term and testing three kernels: linear, polymomial and RBF. For poly, we tune the degree and gamma (and the intercept, not shown). For RBF, we tune gamma (not shown).

- C : As the penalty for misclassification is increased, the model finds smaller margin hyperplanes which better classify both the training and validation sets. As C increases, there is some indication of increase in variance error, as shown by growing gap between the training and validation scores.
- Polynomial, Degree: As the order is increased from second to third, the model generalizes better. Therefore, bias error is reduced with increase in model complexity. As the order increases beyond 3, surprisingly, the training performance deteriorates. Overfitting could have been expected considering the high number of features. This could be indicative that our data is quite linear.
- Polynomial, Gamma: The model experiences quite high sensitivity to this parameter, as demonstrated by its impact on F1. Under a value of approximately 0.005, it underfits the data. This is because model complexity is low, since the region of influence of support vectors is large. As gamma tends to 0, the region of influence of any given support vector would increase to the entire training set and the kernel converges to a linear one. Vice versa, as the parameter is increased, the margins are influenced by closer points, and the model complexity is increased. Progressively, this leads to overfitting, introducing variance error.

Following tuning, the linear kernel is found to perform best. Non-linear kernels are found to overfit the training data and to perform poorer in validation.

***Learning Curve***

The model exhibits some variance error, which is corrected with the increase of the sample size. As the data increases, overfitting is reduced, and generalization improves as demonstrated by increasing validation score. With the full sample size, training and testing score have almost converged. The model appears to have completed learning, as shown by the relatively flat training and validation curves at the end of the training fit times. Most of the learning is completed in the initial iterations.

***Final Model***

The tuned model scores 0.83 on the training set and 0.62 on the test set. While training performance is very similar to that of DT, test performance is inferior to the DT and similar to KNN. Confusion matrix shows the majority negative class was misclassified most of the time, and more frequently than with either DT or KNN. Due to this class having a greater weighting in the test set (0.75), the overall performance of SVM ranks poorer. This indicates the possibility of non-linear causal factors for this class, which a non-linear model such as DT captures better.

## Figures 2.3

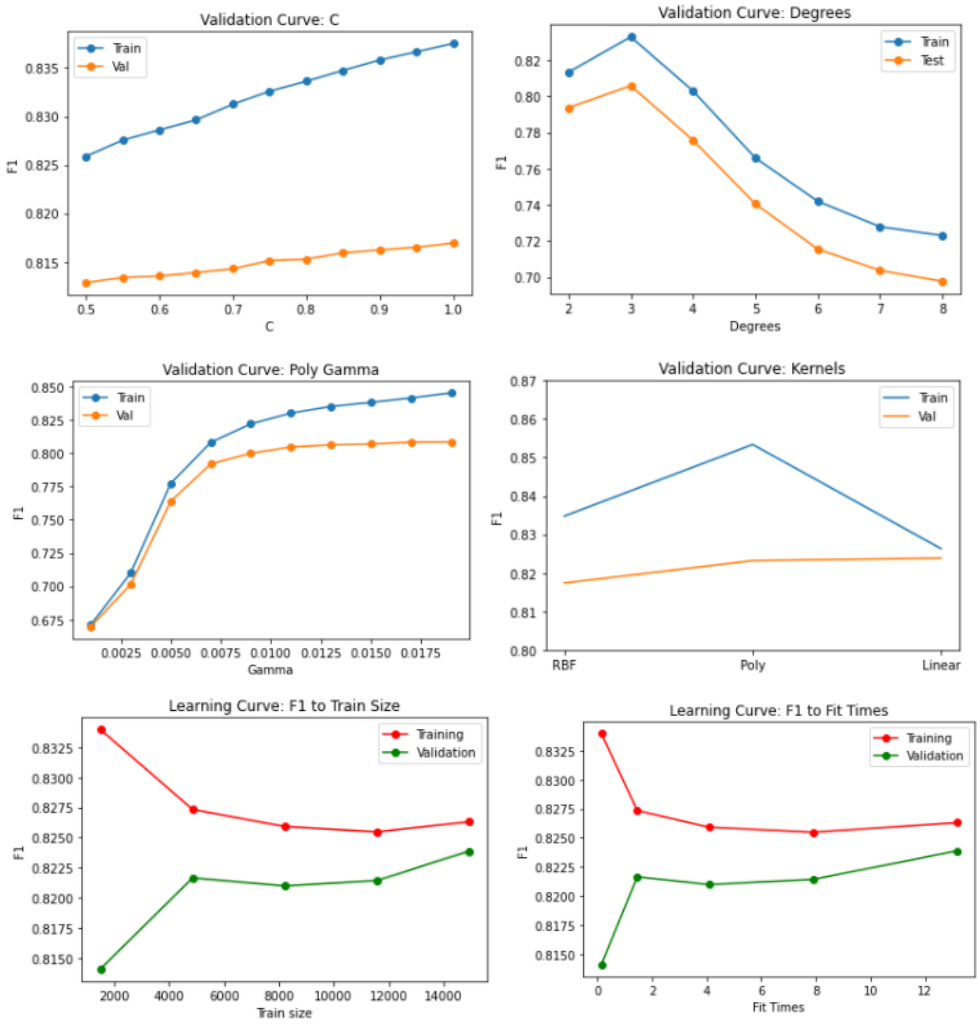



Table 2.2, Confusion Matrix

| True/ Pred. | 0 | 1 |
|---|---|---|
| 0 | 5548 | 1866 |
| 1 | 283 | 2072 |

## 2.4 Neural Network

***Validation Curve***

To tune the Neural Network the following hyperparameters are tuned:

- Hidden layer shape: We consider all permutations of up to 3 layers and up to 4 nodes (where all layers have the same number of nodes). The architectures include a final sigmoid layer. We find that expanding the architecture beyond a single layer with two nodes does not significantly improve model performance. The model fits the data well, neither underfitting nor overfitting. The fact that the training set does not begin to overfit as the network grows could reflect the fact that the chosen Sigmoid activation function has an output range of [0, 1], which mutes the effect of outliers, and therefore hindering overfitting.
- Activation function: We consider Relu, Tanh and Sigmoid. Performances are comparable. Sigmoid is found to perform marginally better. This is somewhat surprising, since arguments exist why both Tanh and Relu ought to perform better than Sigmoid; Tanh due to its wider activation range, and Relu because it doesn't suffer from the vanishing gradient problem. There are two factors which could explain Sigmoid's better performance. First, the shallow depth of the network mitigates the vanishing gradient problem. In a deeper problem, the product of *n* small derivatives (output of Sigmoid is between 0 and 1) would tend to 0 as n is increased, stopping the model's learning, but in a shallow network, vanishing gradients do yet cause issues. Second, although the input data has been normalized, there is no normalization between the hidden layers, which may make Relu susceptible to blow up.
- First moments of decay for Adam solver: Increases improve train and validation performance. This could be due to the curvature of the loss curve being sufficiently steep such that a larger learning rate makes greater progress towards the minimum.

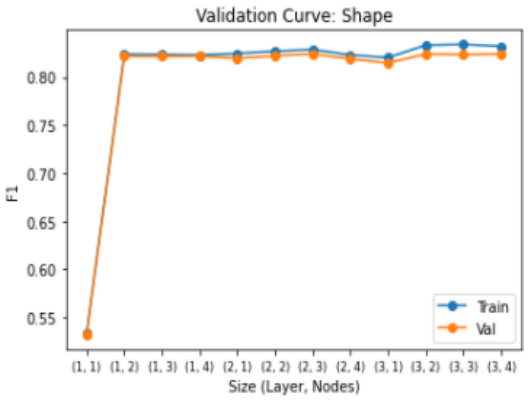
***Learning Curve***

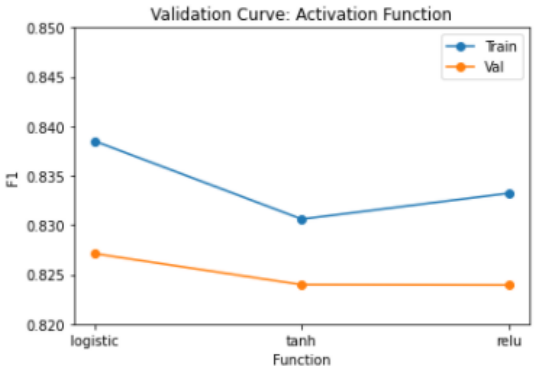
With few data samples the model exhibits high variance. As data is increased, overfitting is reduced, and the model begins to generalize better the validation performance improves. Considering the ability of the activation to mute outliers, this may be driven by better representativeness of the sample with regards to the distributions of X and y and implied correlations. Towards the end of the sample, training performance is declining, which indicates that more data could further reduce overfitting and improve generalization. The curve shows that learning is at completion at the end of the iterations, as demonstrated by the relatively flat curves towards the end of the training. The majority of learning is completed by approximately 6.5 seconds into the run.

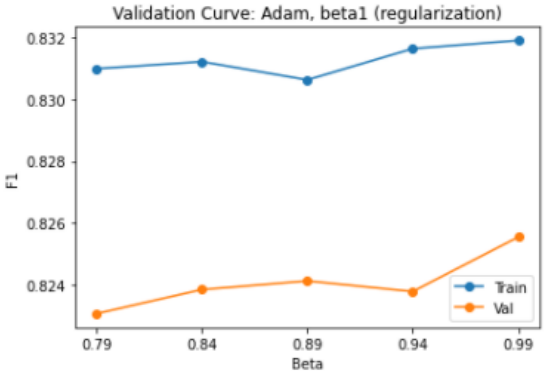
***Final Model***

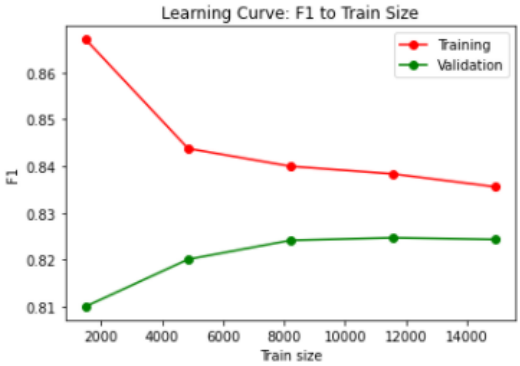
The final results show a training score of 0.83, and a testing score of 0.61, which is the lowest score yet. The confusion matrix shows the misclassifications are concentrated in the negative class. As with previous models, its greater weight in the test sample explains the deterioration from validation performance. The variance error is among the highest seen in the models thus far (0.02 score difference between training and validation), so it is possible that the addition of more samples will reduce overfitting and improve test score.

**Figures 2.4**



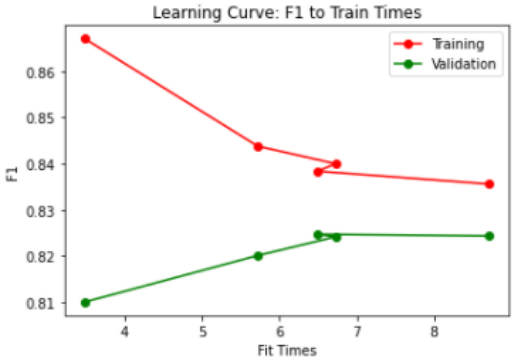




Table 2.4 Confusion Matrix

| True/ Pred. | 0 | 1 |
|---|---|---|
| 0 | 4648 | 2766 |
| 1 | 123 | 2232 |

## 2.5 AdaBoost

***Validation Curves***

To optimize Adaboost we consider:

- Number of weak learners: When the number of weak learners is below approximately 20, the model is underfitting, as demonstrated by the relatively low training and validation scores. Increasing the complexity through increasing the number of weak learners improves performance, and the model gradually begins to overfit when approximately 50 learners are exceeded. Nevertheless, validation performance continues to improve until approximately 260 learners. This indicates that as the tree grows, although some tree paths may be placing undue weight on noise, others continue to improve the model's ability to generalize.
- Learning rate: The model underfits until the rate exceeds 0.2. The learning rate is optimized at a relatively low level of approximately 0.5, reflecting the high number of estimators in the model. Thereafter, further

increases do not significantly impact either training or validation performance. Overall, the model remains overall well fitted.

- Depth of the weak learners as a pruning measure: Beyond a depth of two we see that the weak learners overfit the model, and would increase the variance error.

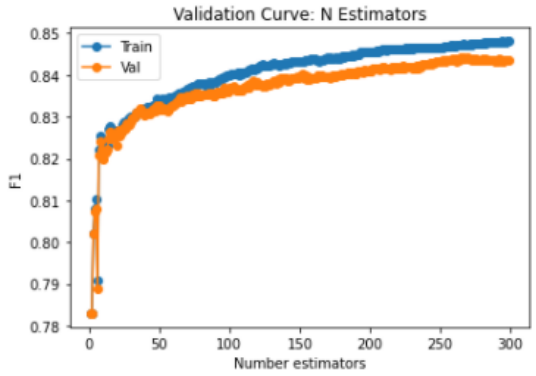
***Learning Curve***

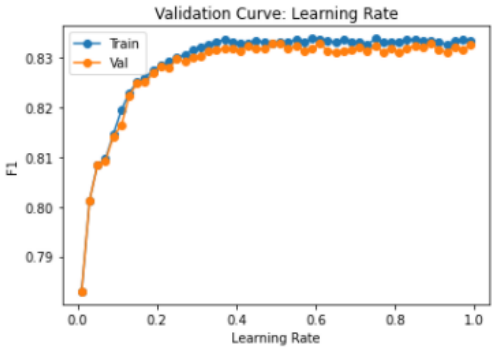
The learning curve shows high variance. With a small sample, the model overfits in training so that the leaves are almost completely pure and performance on the validation set is at its lowest due to failure to generalize. As the sample size increases the overfitting is reduced, but the high variance error remains towards the end of the sample, as demonstrated by the gap between training and validation scores. The addition of further data could help to continue to reduce overfitting.

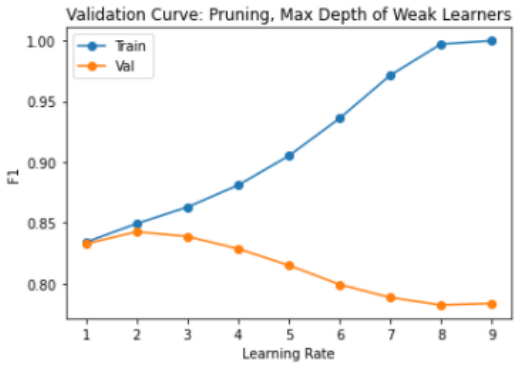
***Final Model***

Train set score stands at 0.86 and test score at 0.71. This makes Adaboost the best performing algorithm thus far. The confusion matrix shows much better classification on the negative class (majority class) in comparison to the previous algortithms.

**Figures 2.5**



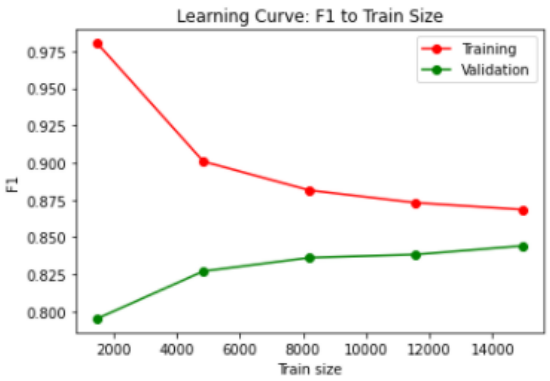




Table 2.5 Confusion Matrix

| True/ Pred | 0 | 1 |
|---|---|---|
| 0 | 6038 | 1376 |
| 1 | 323 | 2032 |

## 3.0 Conclusion

The best performing algorithm was found to be Adaboost by a significant margin. The reason for this appears to be lower bias, likely stemming from the following factors:

- Unlike KNN, feature importance is considered.
- A more complex model in comparison to DT, reflecting the working of the algorithm, whereby each sequential weak learner is added to reduce the error of the previous combination of learners. This adds new weak rules, which cumulate to form a strong learner.
- SVC fitted a linear kernel, which may not be appropriate to the data. Due to high variance, however, the non-linear kernels were found to be suboptimal.
- The lower bias in comparison to NN could indicate a deeper network could rival Adaboost, since the variance error between the two algorithms is comparable (based on validation set).

Fit times show that KNN and SVC take the longest time. In the case of KNN, this reflects the high dimensionality of the data, and the requirement for the dataset to be scanned for each new prediction. In the case of SVC, this is likely contributed to by the high C parameter, which calls for a more intricate boundary. The DT was the fastest, reflecting relatively shallow tree depth of 6 layers. Nevertheless, it is the second best performer, clearly having identified the deterministic features. Neural Networks rank in the middle, reflecting the relatively shallow network, while Adaboost requires significantly more time than DT, due to the number of weak learners and DT's shallow depth.

Table 2.6: Comparison of Train and Test Scores and Fit Times

| | Train Score | Test Score | Fit Time (seconds) |
|---|---|---|---|
| KNN | 0.83 | 0.63 | 32.7 |
| DT | 0.83 | 0.66 | 0.08 |
| SVC | 0.83 | 0.62 | 27.4 |
| NN | 0.83 | 0.61 | 7.8 |
| AB | 0.86 | 0.71 | 8.4 |