CS 7641: Machine Learning
Assignment – 1: Supervised Learning
Ravi Prakash Singh
February 3rd, 2019

Introduction

In this assignment we will explore some techniques in supervised learning. Specifically, we will implement 5 different techniques namely,

- *Decision trees with some form of pruning*
- *Neural networks*
- *Boosting*
- *Support Vector Machines*
- *k-nearest neighbors*

We will use two different classification datasets: Default of Credit Card Clients and Bank Marketing, to compare and contrast results from above mentioned techniques. The idea is to gather understanding on each of these algorithms.

Datasets Description

Default of Credit Card Clients – Dataset has 30,000 instances of credit card customers. Demographics information and credit card limit for each customer is provided. In addition, we also have information on repayment status, billing amount and payment amount for last 6 months. Each customer is labelled with a flag indicating if they default i.e. miss their credit card payment for next month. The problem wants us to predict if the customer is going to default in next month given this information.

Bank Marketing Data - The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. The data has approximately 45,000 instances of different customers with information on their demographics and also campaign related information such as number of contacts, duration of call etc. The classification goal is to predict if the client will subscribe a term deposit.

Why are the problems interesting?

Two problem sets are identifying default on credit cards and predicting if the customer will subscribe bank term deposit. Both problems are related to banking domain. The first problem can help the bank to identify if the customer is going to default in future based on their demographics and spending behavior in last 6 months. This information can be further used by the bank to develop their marketing strategies. For example, the bank can increase credit limit for the customers with very low probability of default in future. This might help them to increase revenue while keeping losses at the minimum and improving customer loyalty.

The second problem can help identify a bank if the client will subscribe to their deposit and what could be the potential factors that drive clients to subscribe it. Using this information, bank can improve on their direct marketing campaign strategies and increase their customer base of term deposit product.

In addition, both the datasets are classification problems having more than 30k instances with more than 15+ features. It provides an opportunity to apply different machine learning algorithms as mentioned above.

Methodology

Splitting Datasets –

Each dataset is split in 80:20 ratio. 80% of original dataset is used for training. The remaining 20% dataset is used for testing. All the different machine learning algorithms learns on the training set and the test set in used only for model evaluation purposes. Hyperparameters for most of the algorithms are tuned using 3-fold cross-validation. For some algorithms, such as Neural Networks, cross validation has been avoided due to time constraints. In such a case,

we have used a part of training set (validation set) to tune our hyperparameters. Hence, for such models the train data is further split such that overall, we use 60% data for training, 20% data for validation and 20% data for testing.

Scaling/Normalization –

Features for both the datasets have been standardized by removing the mean and scaling to unit variance. We first standardize training set and then use it's mean and standard deviation to standardize test set.

Choosing Evaluation Metrics –

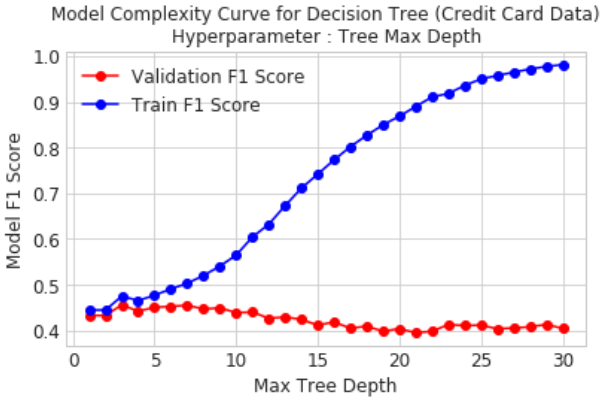
Both the datasets have imbalanced classes. The response rate of Credit Card dataset is around 11% whereas the response rate for Bank Marketing dataset is approximately 22%. In such as case, evaluation metrics like accuracy could be misleading. We would evaluate our model using F1-score since it accounts for both Precision and Recall.

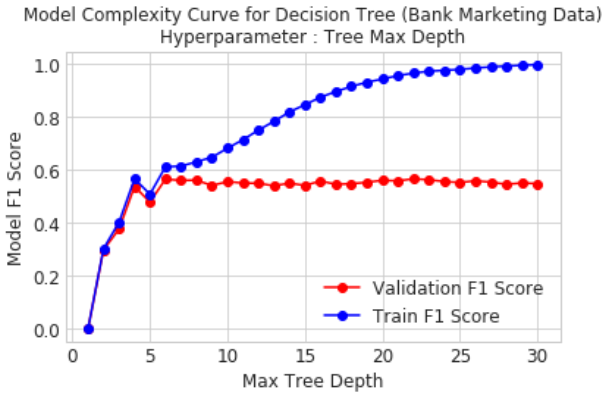
Model Selection –

For most of the models, we use Grid Search to identify best set of hyperparameters. Grid search is done using cross validation and the best set of hyperparameters is chosen based on best cross-validation F1-score. We also look at learning curve for each algorithm and try to work on bias and variance of the model.

Decision Trees

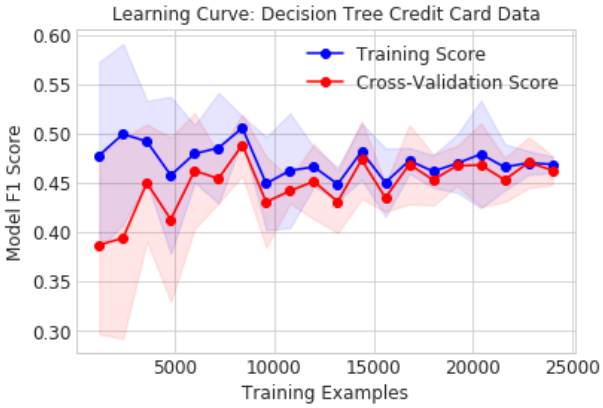
We have built Decision Tree Classifier using information gain (based on entropy) to determine the best feature split per the ID3 algorithm. The model will be pre-pruned by limiting tree depth using the hyperparameter 'max_depth' and by ensuring that each leaf (a terminal node on the tree) has at least 'min_samples_leaf'.

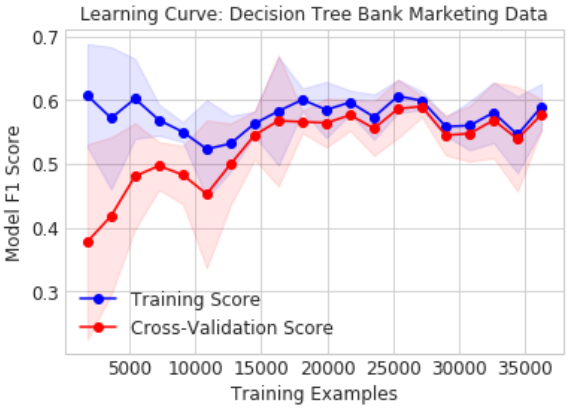




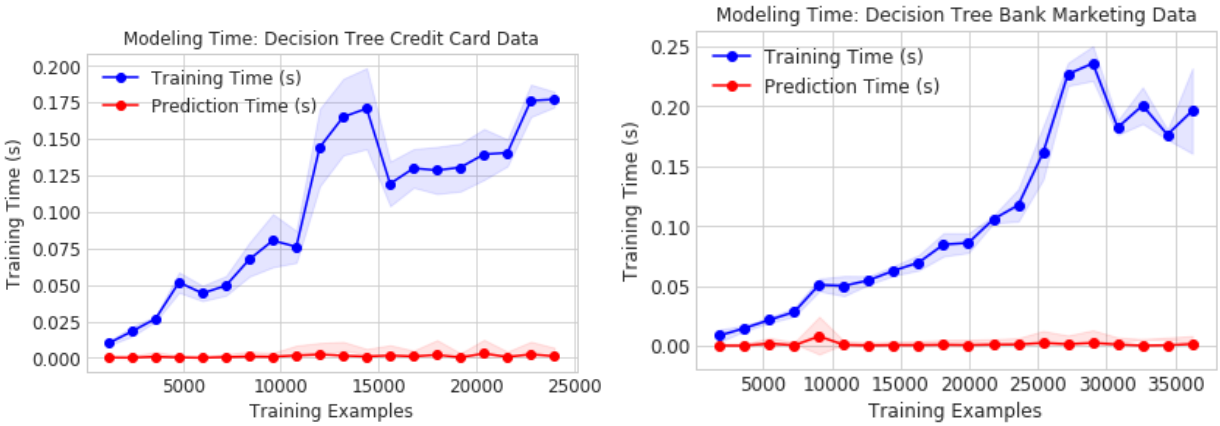
As expected, it is easy to observe that with deeper trees we get very high F1-score on the training set. However, the validation (not test) set does not show any improvement after certain tree depth is reached. This implies that tree has started overfitting beyond that depth. Hence, we get an idea of optimum tree depth for both the datasets. Along with max_depth, we have then used Grid Search to vary following parameters –

- min_samples_leaf - The minimum fraction of samples required to be at a leaf node.
- min_samples_split - The minimum fraction of samples required to split an internal node.
- max_features - The number of features to consider when looking for the best split





The learning curves are obtained by using optimal values from Grid Search. As expected, as training size increases, we see decrease in training F1-score and increase in cross-validation F1-score. The learning curve for both the dataset seems to have converged and the gap between Training and Cross Validation scores with sufficiently large amount of data is minimal. This represents that our model is well tuned and there are no traces of overfitting.



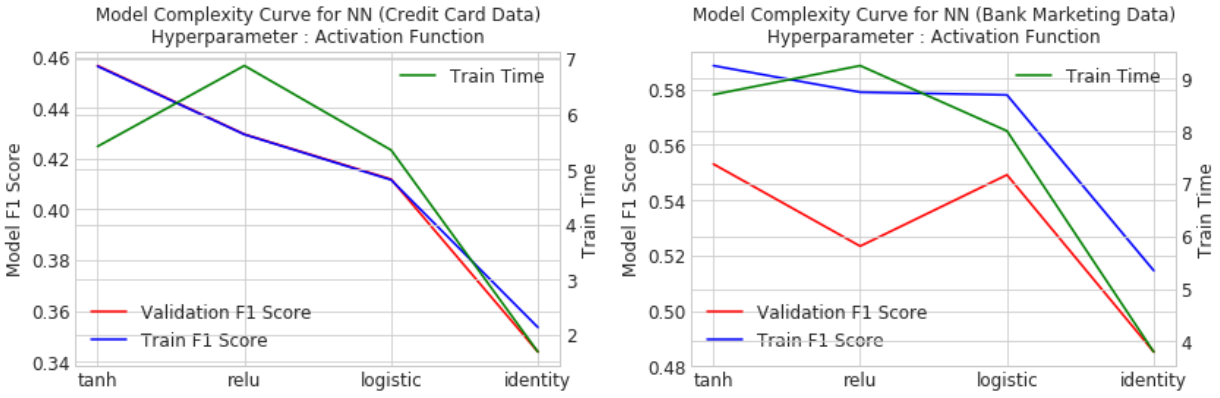
The graph above suggests that training time increases with number of training examples. However, the prediction time is constant. Below is the summary of model evaluation metrics for both the datasets.

Model Evaluation Metrics	
Credit Card Default	**Bank Marketing**
Train Dataset ********** Model Training Time (s): 0.23478 F1 Score: 0.46 Accuracy: 0.82 AUC: 0.65 Precision: 0.68 Recall: 0.35 Untouched Test Dataset ********** Model Prediction Time (s): 0.00069 F1 Score: 0.48 Accuracy: 0.83 AUC: 0.66 Precision: 0.69 Recall: 0.37 **********	Train Dataset ********** Model Training Time (s): 0.24922 F1 Score: 0.59 Accuracy: 0.91 AUC: 0.75 Precision: 0.64 Recall: 0.55 Untouched Test Dataset ********** Model Prediction Time (s): 0.00131 F1 Score: 0.58 Accuracy: 0.91 AUC: 0.75 Precision: 0.63 Recall: 0.54 **********

Neural Networks

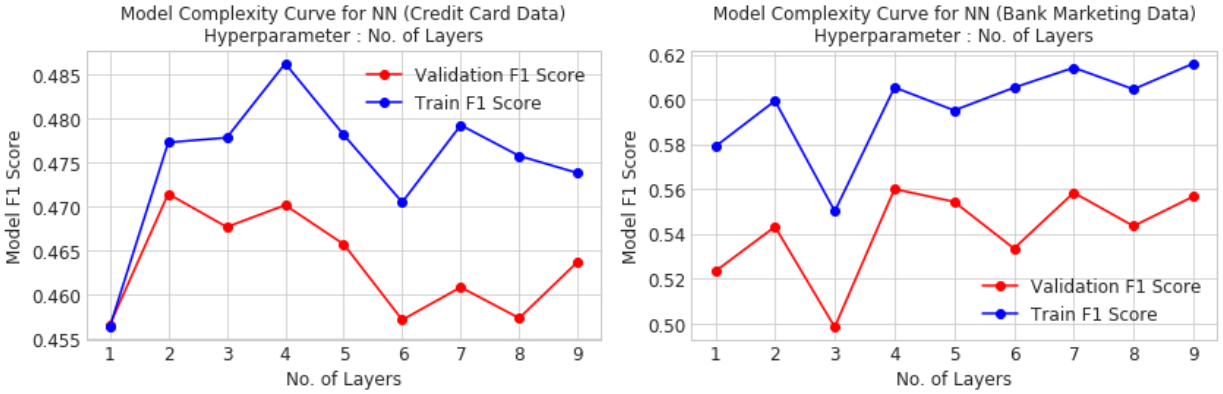
This section will build a forward-feed neural network which computes weights via backpropagation (a multilayer perceptron). Since we have sufficiently large datasets and there were multiple hyperparameters to be tuned, we did not do cross validation due to obvious time constraints. Instead, we have tuned the model using one validation set. While tuning, one hyperparameter we have kept other hyperparameters constant. We see some interesting results.

Activation Function



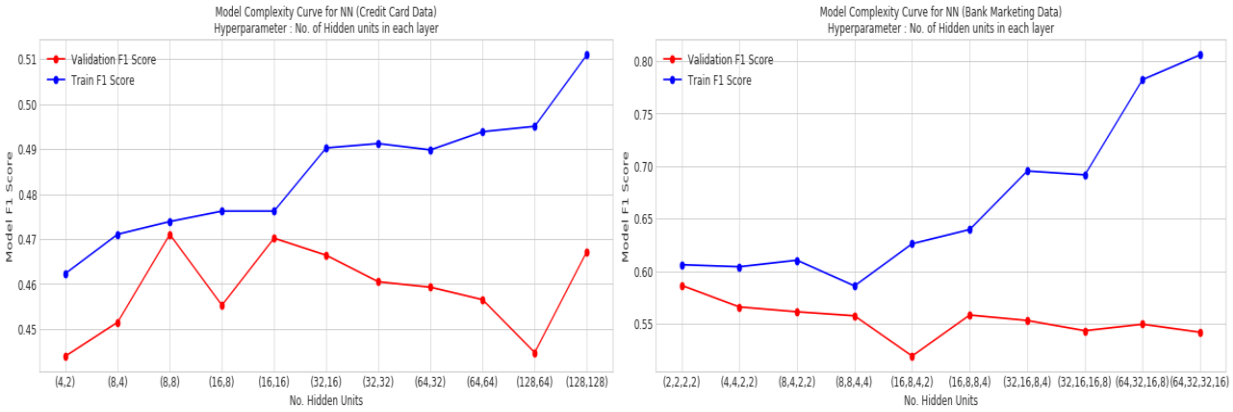
We identify that training time for both the datasets is interestingly low for tanh activation function than ReLU. This might be because our network is not deep enough to produce difference. Identity function have lowest training time, but it has least F1 score. This is because identity function is linear with constant function's derivative i.e. all the weights get same update during backpropagation and hence our network doesn't learn well. Therefore, we choose tanh activation function for our problem because it has lowest training time and highest Validation F1 -score.

Number of Layers



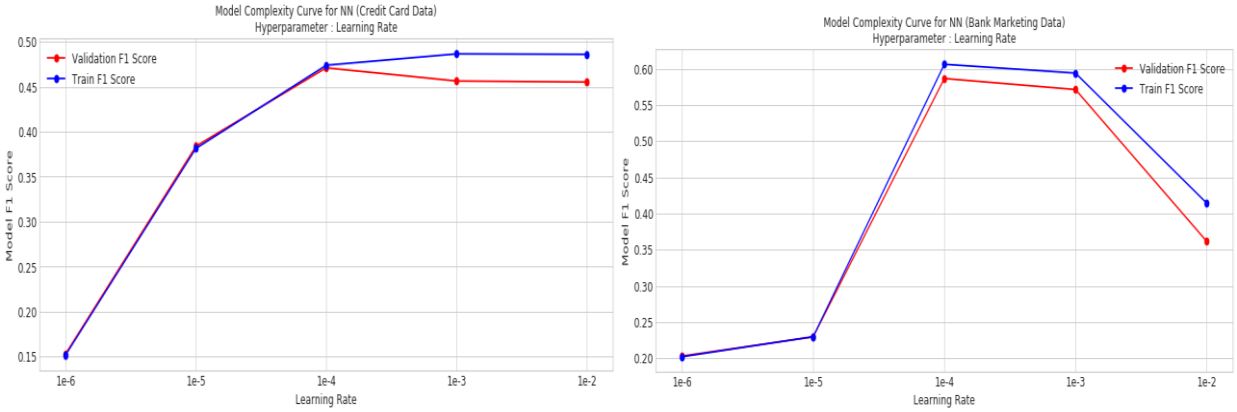
As number of layers increases, we observe that training F1-score increases whereas Validation F1-score increases for few layers and then stops. This is due to overfitting. As number of layers increases, model complexity increases, and model starts to learn co-incidences in the data instead of real information. So, we choose optimal number of layers as 2 & 4 for Credit Card & Bank Marketing data respectively because we get highest validation accuracy in that case.

Number of Neurons



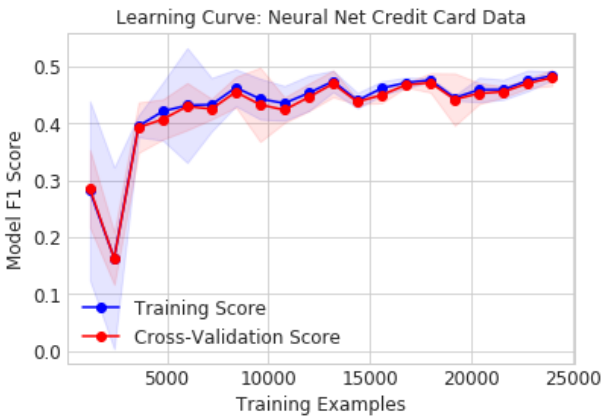
Keeping number of layers fixed as obtained previously, we have varied number of neurons in each layer and found that as number of neurons increases in each layer, model complexity increases, and our model starts overfitting. So, we choose the optimal combination of neurons which gives highest validation F1-score i.e. (8,8) and (2,2,2,2) for credit card and Bank Marketing data respectively.

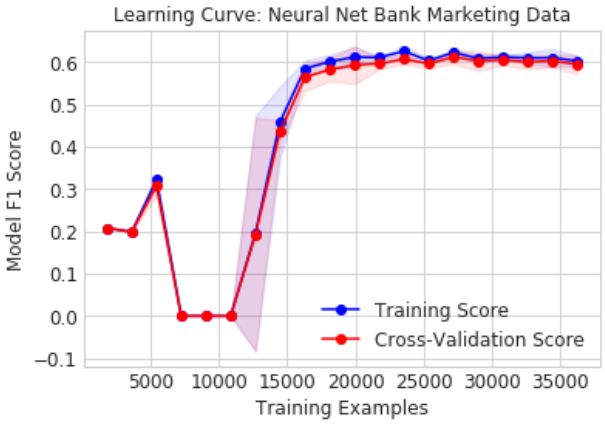
Learning Rate



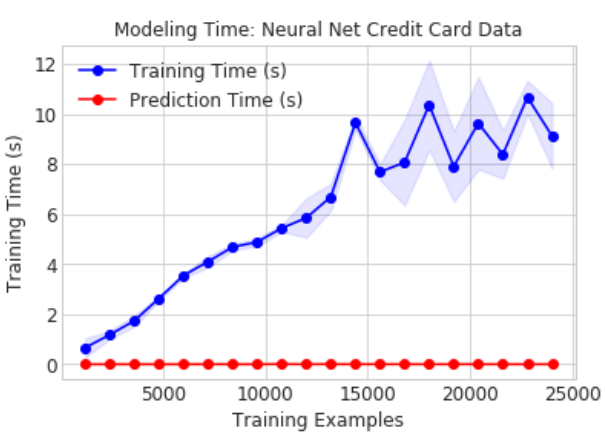
The learning rate curves show that our model performs well when learning rate increases whereas the performance drops when learning rate is too high. This is reasonable because when learning rate is too low, network barely learns whereas when it is too high, we might skip global minima or even diverge. Co-incidentally, 1e-4 is optimal learning rate for both the datasets.

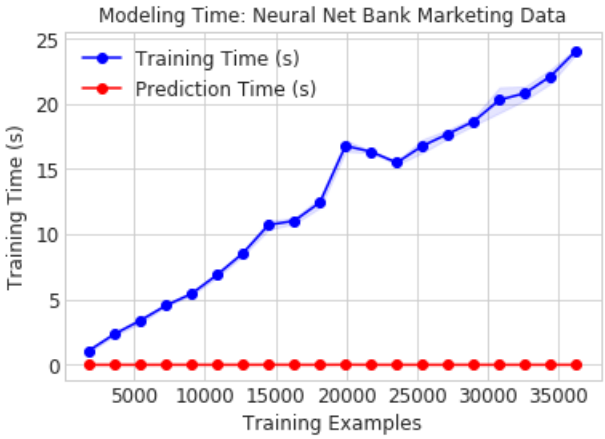
Using these set of hyperparameters, we build our final Neural Network on both the datasets.





Learning curve for both the datasets is well converged and saturated. This shows no sign of overfitting and underfitting implying that our search for best set of hyperparameter has been almost successful.



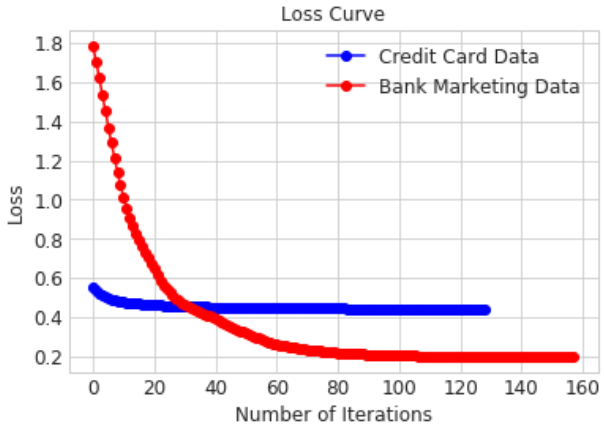


Training time increases almost linearly as number of training example increases. This is obvious as training more examples makes it hard for Neural Network to learn.

We have identified best set of hyperparameters by varying one hyperparameter and keeping others constant. This is not the best way to find the hyperparameters since it is biased by the fixed set. Ideally, we should have done cross-validation using Grid Search. In future, we would like to do Grid Search to obtain optimal set of hyperparameters.

Model Evaluation Metrics	
Credit Card Default	**Bank Marketing**
Train Dataset ********** Model Training Time (s): 7.55154 F1 Score: 0.47 Accuracy: 0.82 AUC: 0.66 Precision: 0.67 Recall: 0.37 Untouched Test Dataset ********** Model Prediction Time (s): 0.00616 F1 Score: 0.48 Accuracy: 0.83 AUC: 0.66 Precision: 0.67 Recall: 0.38 **********	Train Dataset ********** Model Training Time (s): 22.07539 F1 Score: 0.60 Accuracy: 0.91 AUC: 0.77 Precision: 0.61 Recall: 0.60 Untouched Test Dataset ********** Model Prediction Time (s): 0.00921 F1 Score: 0.60 Accuracy: 0.91 AUC: 0.77 Precision: 0.62 Recall: 0.59 **********

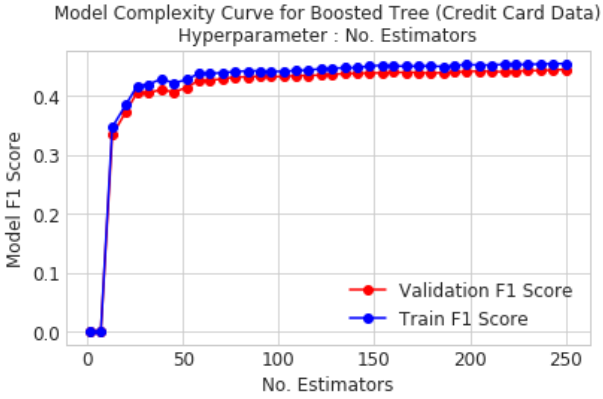
We also look at loss curve for each dataset over the iterations.

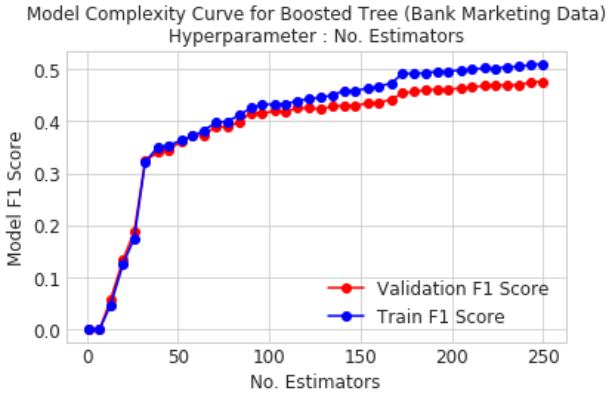


The loss curve for both the datasets stabilizes as number of iterations increases. Hence, it reflects that our Neural Network model is well tuned on both the datasets.

Boosting

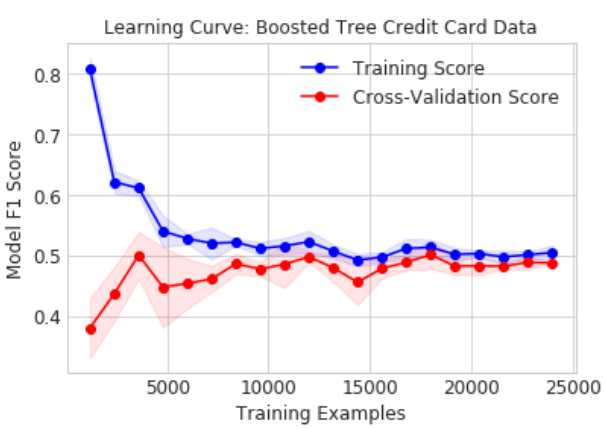
This section will implement a boosted version of the earlier decision tree. We will still keep the pruning based on max_depth, but the cutoff thresholds will be more aggressive (lower) since the power of boosting is to combine multiple weak learners. We also test the hyperparameter of n_estimators and learning rate.

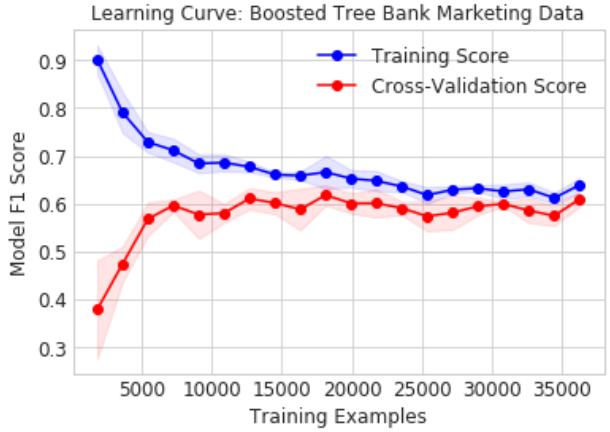




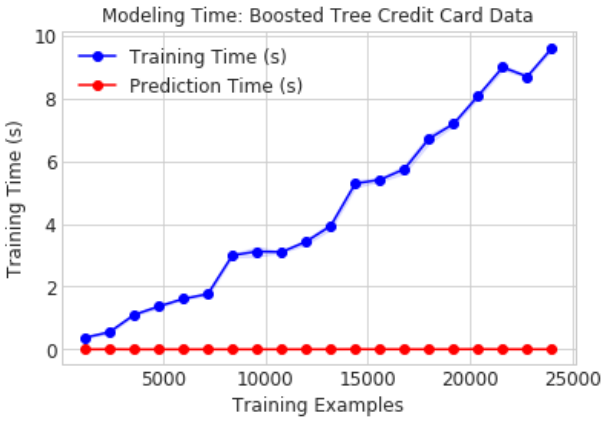
We observe that as number of estimators i.e. number of gradient boosted trees increases, the training F1-score increases. However, it saturates for Credit Card data, but it keeps on increasing for Bank Marketing data. In contrast, validation F1-score increases slowly compared to training. This represents that we might start overfitting with large number of trees. So, we need to control on number of trees to be built.

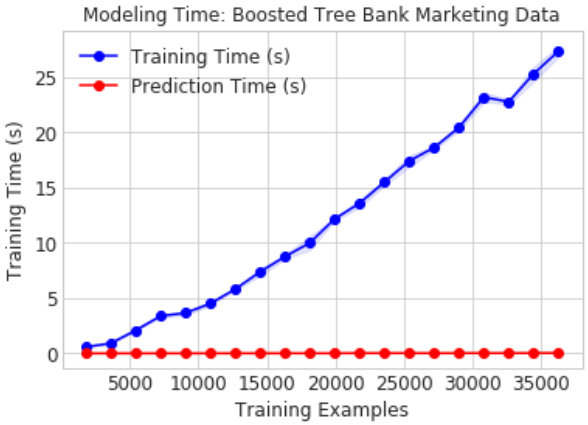
We have used Early Stopping criteria if F1-score on Validation set does not improve after 10 iterations to identify optimal number of trees. We do Grid Search using cross validation to identify optimal set of hyperparameters.





Apart from min_samples_leaf, max_depth and min_samples_split as described in decision trees, we also search for optimal learning rate. The learning curves for both the datasets indicated that we don't have sign of overfitting or underfitting. The cross validation has been successful in identifying optimal set of hyperparameters. Using those hyperparameters, model has converged well and generalizes excellently on untouched test data set.



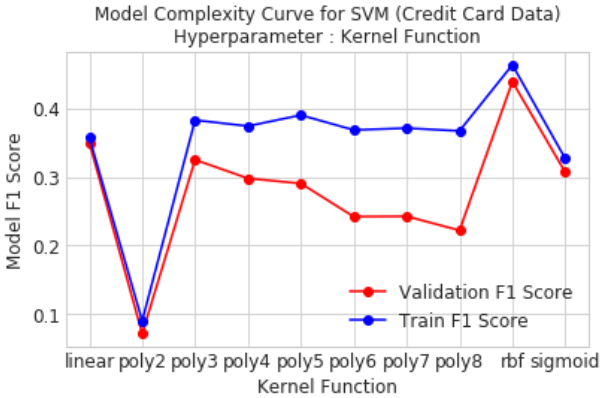


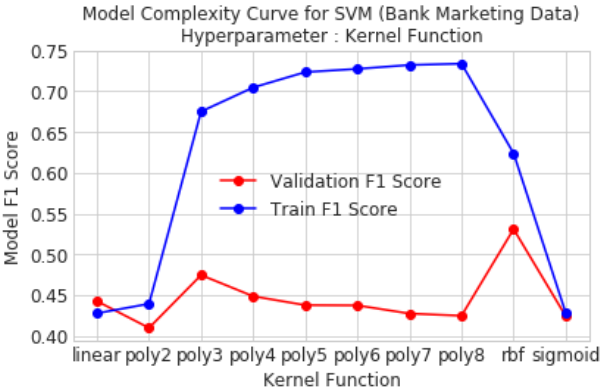
Training time increases linearly with number of training examples. In future, to improve performance of the boosted trees, we can further prune our tree based on min_impurity_decrease i.e. a node will be split if this split induces a decrease of the impurity greater than or equal to this value or maybe we can try using Adaptive boosting.

Model Evaluation Metrics	
Credit Card Default	**Bank Marketing**
Train Dataset ********** Model Training Time (s): 13.59756 F1 Score: 0.49 Accuracy: 0.82 AUC: 0.66 Precision: 0.67 Recall: 0.38 Untouched Test Dataset ********** Model Prediction Time (s): 0.04580 F1 Score: 0.50 Accuracy: 0.83 AUC: 0.67 Precision: 0.67 Recall: 0.39 **********	Train Dataset ********** Model Training Time (s): 32.37411 F1 Score: 0.60 Accuracy: 0.92 AUC: 0.76 Precision: 0.68 Recall: 0.55 Untouched Test Dataset ********** Model Prediction Time (s): 0.03469 F1 Score: 0.59 Accuracy: 0.92 AUC: 0.75 Precision: 0.66 Recall: 0.53 **********

Support Vector Machines

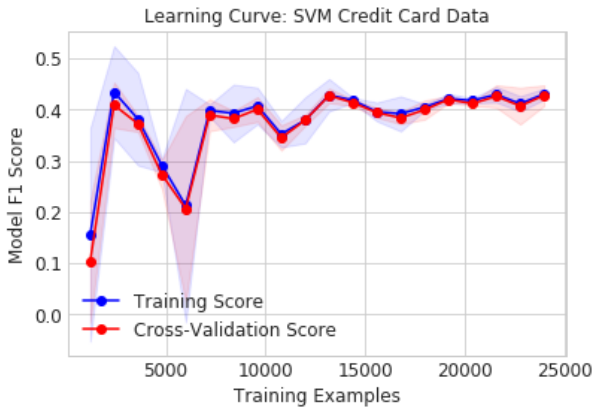
This section will build a support vector machine classifier. The primary hyperparameter will be kernel function: linear, polynomial, rbf (radial basis function), and sigmoid. We have tried polynomials up to 8 degrees. We will also explore the penalty term 'C' and the kernel coefficient 'gamma'.

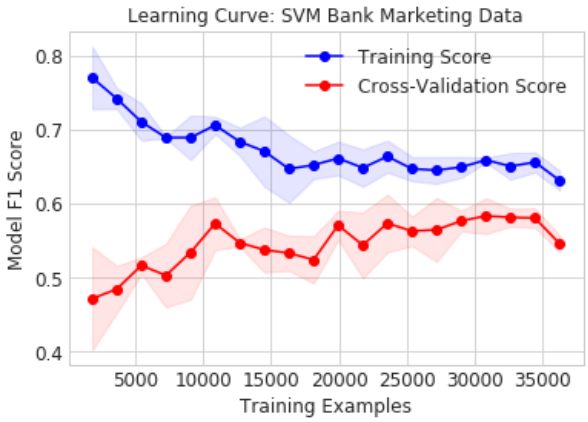




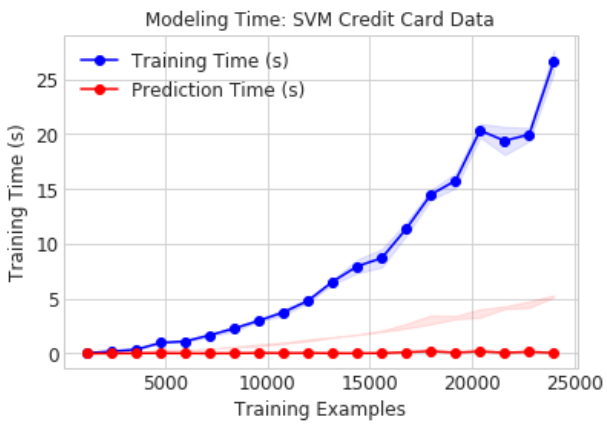
The training F1-score increases as model complexity increases as is evident by increasing nature of curve with

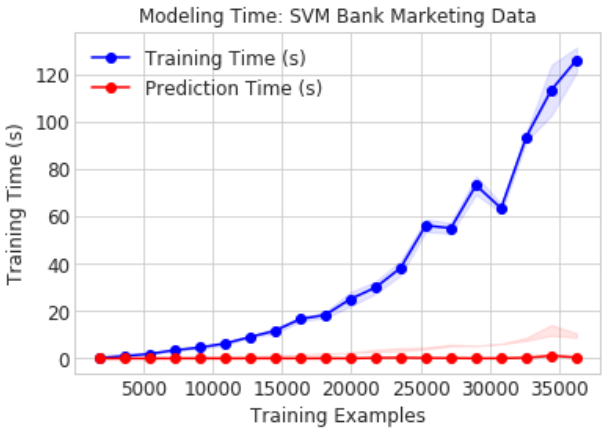
increasing degree of polynomial. However, Validation F1-score keeps on decreasing with increasing model complexity which represents strong overfitting. In addition, for Bank Marketing data we observe that there is huge gap in training and Validation F1-score for almost all the kernels except linear and sigmoid. This might represent that these kernels are much complex, and we might need more data to reduce the gap. For both the datasets we observe that 'rbf' kernel gives highest F1-score in Validation set. Hence, we have chosen 'rbf' kernel for further exploration.





We have done Grid Search using Cross-Validation to find optimal combination of Hyperparameters of penalty parameter 'C' and 'rbf' kernel coefficient 'gamma'. The learning curve for Credit Card dataset for both Training and Validation has converged and they have minimal gap, indicating our model is well tuned. However, for Bank Marketing data, we observe some gap in Training and Validation F1-score which shows there is some overfitting. In future, we can improve this model by adding more training instances.



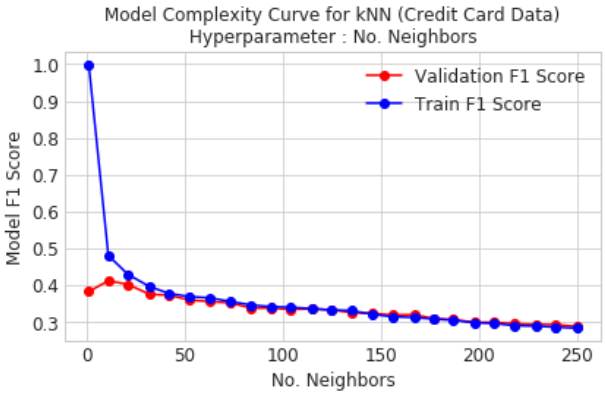


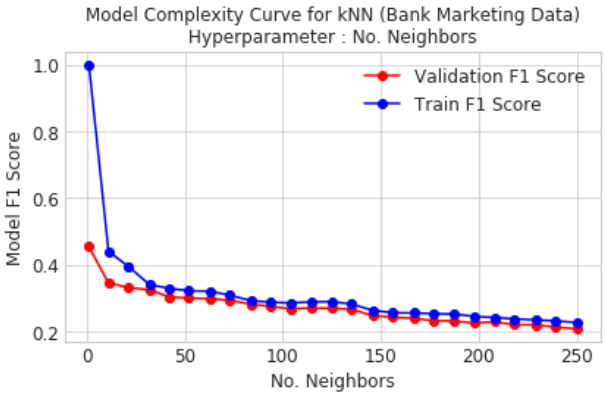
Training time increases linearly with number of training examples whereas prediction time is constant and negligible.

Model Evaluation Metrics	
Credit Card Default	**Bank Marketing**
Train Dataset ********** Model Training Time (s): 39.31730 F1 Score: 0.43 Accuracy: 0.81 AUC: 0.64 Precision: 0.69 Recall: 0.31 Untouched Test Dataset ********** Model Prediction Time (s): 3.41286 F1 Score: 0.43 Accuracy: 0.83 AUC: 0.64 Precision: 0.68 Recall: 0.32 **********	Train Dataset ********** Model Training Time (s): 348.53473 F1 Score: 0.60 Accuracy: 0.92 AUC: 0.74 Precision: 0.74 Recall: 0.50 Untouched Test Dataset ********** Model Prediction Time (s): 8.52250 F1 Score: 0.51 Accuracy: 0.91 AUC: 0.70 Precision: 0.64 Recall: 0.42 **********

K-nearest neighbors

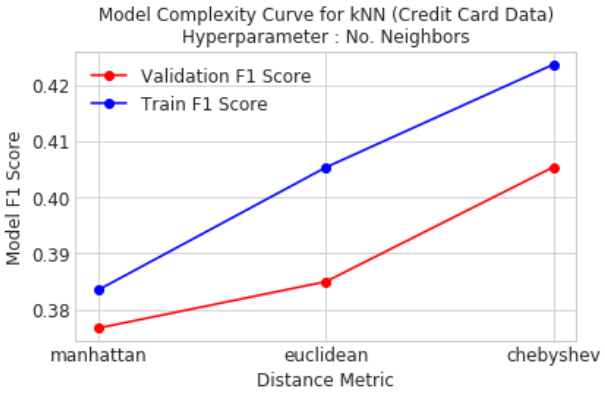
This section will build a classifier using K-nearest neighbors. The hyperparameter will be n_neighbors and the distance metric. The Model Complexity curve will show F1 score as a function of number of neighbors.

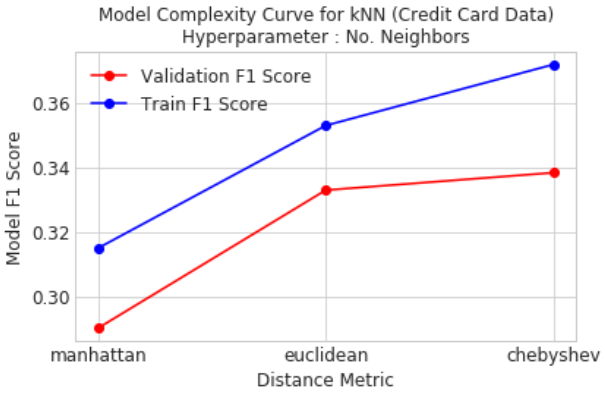




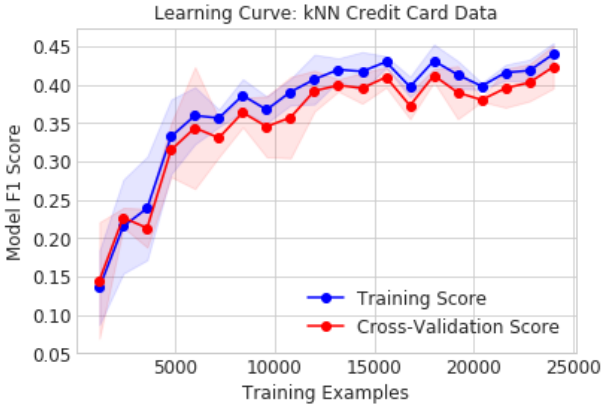
Both the graph represents that as number of neighbors increases, training as well as validation F1-score goes down very slowly or remain constant. This might tell us that increasing value of k after certain point does not help model in learning. So, we can choose k to be close to the point from where the performance graph starts flattening i.e. 33.

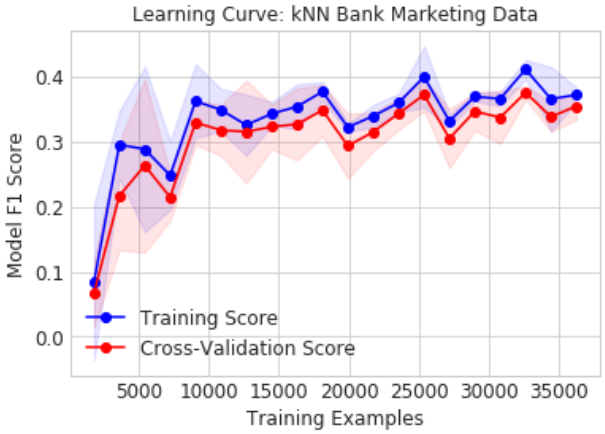
Further, we have also explored three different distance metrics for the classification namely, Manhattan, Euclidean and Chebyshev.



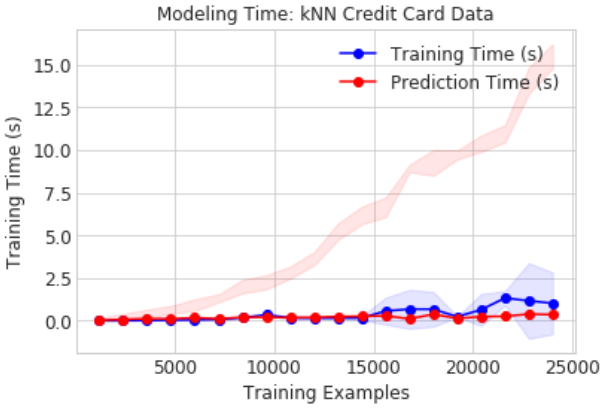


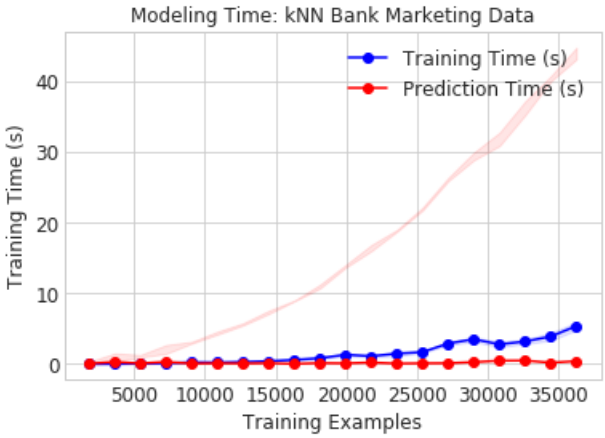
The above two graphs suggest that Chebyshev distance metric gives highest Validation F1-score for both the datasets. However, we see that there is significant different in train and validation F1-scores.





The learning curve for both the datasets is not very good. It seems like the model has not completely learnt yet. This indicate that we might need more data to learn or create more/better features for KNN to learn readily. In addition, to improve model performance we could have performed Grid Search using cross validation to find optimal set of hyperparameters k and distance metric.





In contrast to all other algorithms, KNN has found to take constant amount of time to train for both the datasets.

Model Evaluation Metrics	
Credit Card Default	**Bank Marketing**
Train Dataset ********** Model Training Time (s): 0.51985 F1 Score: 0.43 Accuracy: 0.81 AUC: 0.64 Precision: 0.68 Recall: 0.31	Train Dataset ********** Model Training Time (s): 6.50791 F1 Score: 0.37 Accuracy: 0.90 AUC: 0.62 Precision: 0.65 Recall: 0.26
Untouched Test Dataset ********** Model Prediction Time (s): 6.36513 F1 Score: 0.40 Accuracy: 0.82 AUC: 0.62 Precision: 0.64 Recall: 0.29 **********	Untouched Test Dataset ********** Model Prediction Time (s): 18.23783 F1 Score: 0.33 Accuracy: 0.90 AUC: 0.61 Precision: 0.62 Recall: 0.23 **********

Summary

This brings us to the conclusion of this report. We will now compare 5 different classifiers on built on two datasets.

Model Performance

We compare model evaluation statistics on test set across different classifiers for both the datasets.

Credit Card Default Dataset

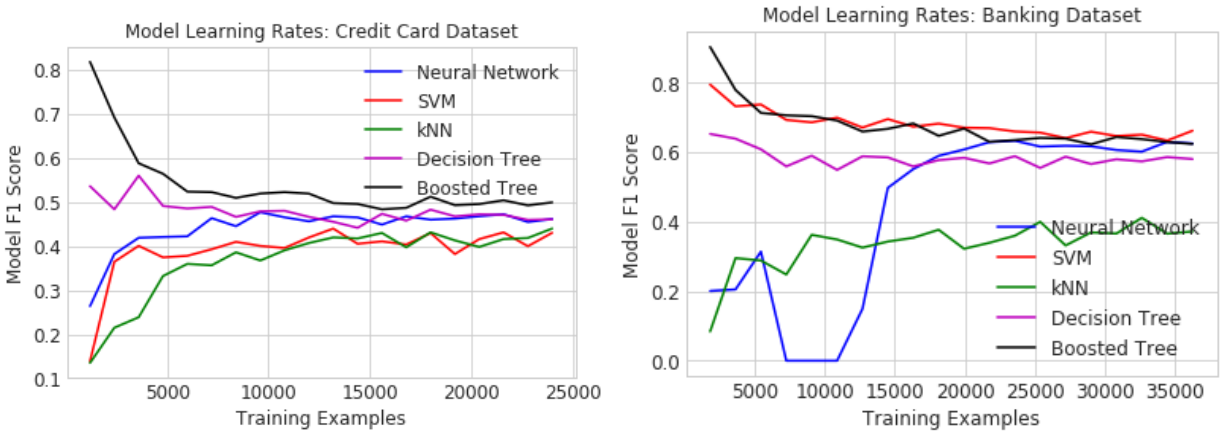
Classifiers	F1-Score	Accuracy	AUC	Precision	Recall
Decision Trees	0.48	0.83	0.66	0.69	0.37
Neural Networks	0.48	0.83	0.66	0.67	0.38
Boosted Trees	0.50	0.83	0.67	0.67	0.39
SVM	0.43	0.83	0.64	0.68	0.32
KNN	0.40	0.82	0.62	0.64	0.29

Bank Marketing Dataset

Classifiers	F1-Score	Accuracy	AUC	Precision	Recall
Decision Trees	0.58	0.91	0.75	0.63	0.54
Neural Networks	0.60	0.91	0.77	0.62	0.59
Boosted Trees	0.59	0.92	0.75	0.66	0.53
SVM	0.51	0.91	0.70	0.64	0.42
KNN	0.33	0.90	0.61	0.62	0.23

For both the datasets we observe that Accuracy is almost the same. However, F1-score fluctuates a lot. This is because of unbalanced response in both the datasets. All the classifiers are able to get good Precision but it's hard

for them to get a good Recall. Specially, classification algorithms like SVM and KNN, find it very difficult to get good recall.



Above graph compares learning rates for all the different classifiers. We observe that for both the datasets learning curve converges to approximately similar values. However, KNN is exceptionally bad for Bank Marketing data. Whereas, we observe that complex algorithms like Neural Network and Boosted trees learn pretty quickly.

Training & Prediction Time

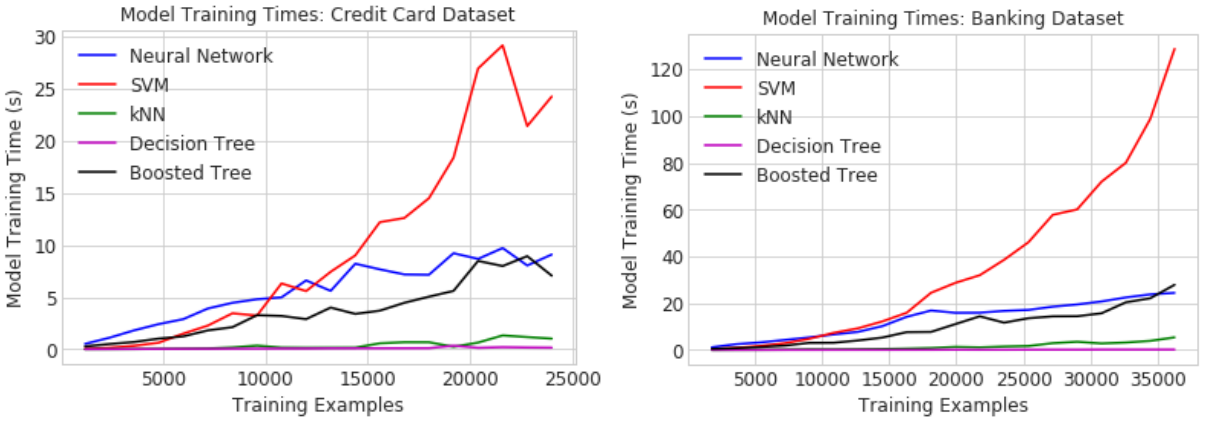
Credit Card Default Dataset

Classifiers	Training Time (s)	Prediction Time (S)
Decision Trees	0.23478	0.00069
Neural Networks	7.55154	0.00616
Boosted Trees	13.59756	0.04580
SVM	39.31730	3.41286
KNN	0.51985	6.36513

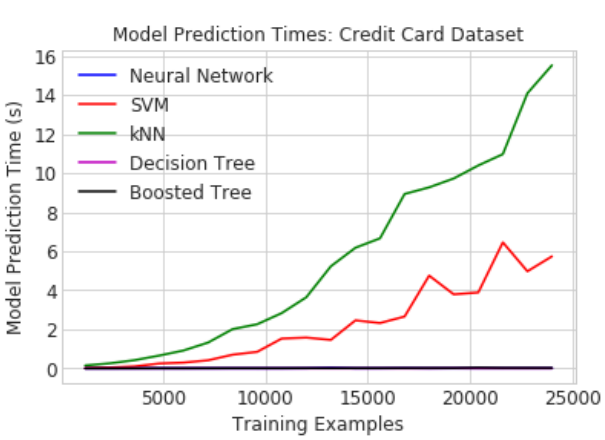
Bank Marketing Dataset

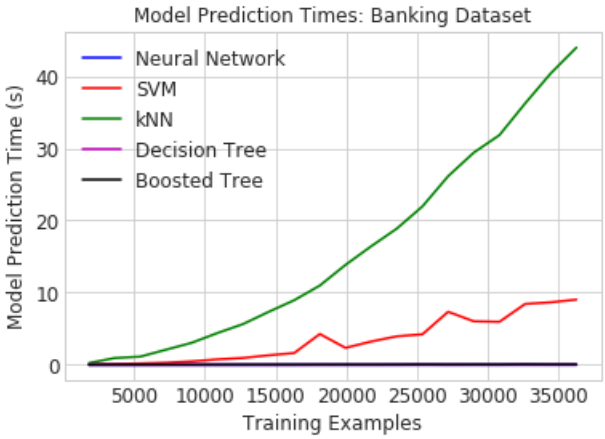
Classifiers	Training Time (s)	Prediction Time (S)
Decision Trees	0.24922	0.00131
Neural Networks	22.07539	0.00921
Boosted Trees	32.37411	0.03469
SVM	348.53473	8.52250
KNN	6.50791	18.23783

In terms of training time, we see that SVM takes most amount of time for training. And Decision Trees are best when it comes to training time. Neural Networks are second best in terms of both training and prediction times.



Even in terms of prediction times, we find that Decision Trees are the best. KNN takes largest amount of time when it comes to prediction.





Based on model evaluation statistics and training and prediction times, we can conclude that Neural Network has been found to be best model for both the datasets. It is because Neural Networks are complex enough to learn the structure of both the datasets and their training and prediction times are also reasonable.