Bank Marketing and Phishing Websites Using Supervised Learning

CS 7641 – Machine Learning

Manish Mehta

Introduction

The purpose of this project is to explore and apply some of the supervised learning techniques to interesting classification problems. For the purpose of this assignment, we have selected the data on "Phishing Websites" and "Bank Marketing", as these are some of the most attractive areas of interest and have a large-scale impact for any corporation and organization. Further details about each of the datasets and what makes them so interesting for the purpose of our analysis will be provided in the next section. Furthermore, we are applying the Decision Tree, Neural Network, Boosting, Support Vector Machines and k-Nearest Neighbors algorithms to each of the datasets we selected. The aim is to understand, implement and analyze each of these algorithms across each of the datasets and against each other in terms of performance, accuracy, speed and efficiency. We have mostly used F1 score as a metric for estimating the accuracy/performance of the model. Along with that, we also report the training, cross-validation and testing accuracies, and fit times for each of the algorithms for the sake of completion.

What makes it interesting?

One of the major questions before we begin our analysis is what makes these problems particularly interesting, and why should we care about looking at them. Before answering that, let us see in detail what each of the datasets are (along with a few features of the dataset):

- Bank Marketing: This dataset is related to direct marketing campaigns of a Portuguese banking institution. These marketing campaigns were based on phone calls. Often, the same client needs to be contacted more than once to assess if the product being offered by the bank (bank term deposit in this case) would be subscribed to by the client or not. The classification goal for the problem is to identify whether the client with subscribe a term deposit or not. This is a binary classification problem based on certain features including but not limited to – Age, Client Job, Education, Credit in Default, Last Contacted, etc. In total there are 45,307 rows of data with 20 predicting features and 1 response variable.

- Phishing Website: This dataset arises from the need of several researchers who have issues obtaining a reliable dataset in the field of Phishing and Website Frauds. This dataset aims to bring to light the features that have proven to be successful in identifying and predicting the cases of phishing websites. With a collection of 11,055 rows and 30 predicting variables and 1 response variable, this dataset contains features including but not limited to URL Features (fake IP address in URL, @ sign in the URL, overtly Long URL, shorted URL redirecting to a phishing website, etc.), HTML and JavaScript based features, Domain based features, etc.

These datasets are particularly interesting because of the scale of impact and the relevance they have with a lot of markets across the globe. Being a Bank Marketing data does not restrict the analysis and conclusions associated with it to not be extended to other marketing activities. We can always obtain similar features as in this dataset, pertaining to different businesses to draw similar conclusions and predict whether the customer/client will, for example, subscribe to premium membership on an e-commerce company, or will the riders be willing to say yes to subscription schemes offered by ride-sharing agencies like Uber, Lyft, etc. Such data helps understand the metrics which the clients respond to and can help understand and establish a stronger hold on market and lead to a marketplace balance. More direct applications of the same can be found here[1],[2].

One of the biggest threats to online security is the phishing websites. The phishing websites usually appear as legit websites, but it is important for the anti-malware or firewall system and the browser to detect the phishing websites that are embedded sometimes in the webpages or in spam emails. Identifying features and classifying websites as phishing has practical implications. Several researches have been going on around the world, such as here[3],[4],[5]. With over 10k+ rows of data and 15+ features (more information on data type of features, we can always refer to the websites we took the data from, as mentioned in the References section), these serve as some non-trivial problems to which classification algorithms can be applied to obtain interesting insights.

Analysis and Supervised Learning Algorithms

In this section we go through the results of applying each of the 5 Supervised Learning algorithms to each of the two classification problems at hand and look for any interesting insights and possible scope of improvement, that could have been implanted given more time and resources. In general, for each case, 80% of the data was used for training and validation and 20% was used for testing the model. Hyperparameter tuning was performed using a 10-fold cross validation to arrive at the best set of parameters to use for final testing of the model. Until then, the test dataset was untouched. Validation curves and understanding of the algorithm were used to select the final range of hyperparameters to run a cross-validated Grid Search on, to report optimal values for final model. We report the Training, Validation and Testing accuracies also, along with the F1 score for each case, and towards the end compare and contrast each of the algorithms that were implemented. All the analysis has been done in Python3.

Neural Networks

A feed-forward neural network was used (using the MLPClassifier routine in scikit-learn). It computes the weights by back-propagation and as hyperparameters, a couple of things were tried – changing the number of neurons in a single hidden layer structure, changing the number of layers and number of neurons in each layer with the number of neurons increasing, decreasing and staying the same as the number of layers increased and finally keeping the number of layers and the neurons in the layer constant, the learning rate was varied. For this exercise, the activation function used was 'logistic'. This helped plot a series of validation curves as shown below:

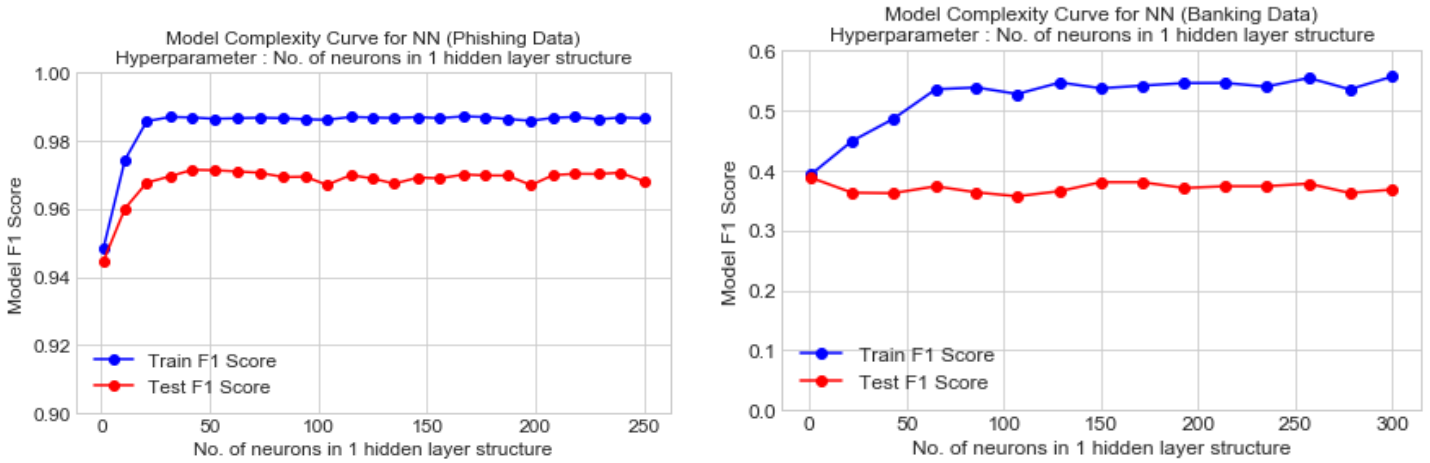


Fig.1 Validation Curve for 1-layer structure with multiple neurons – Phishing and Banking data

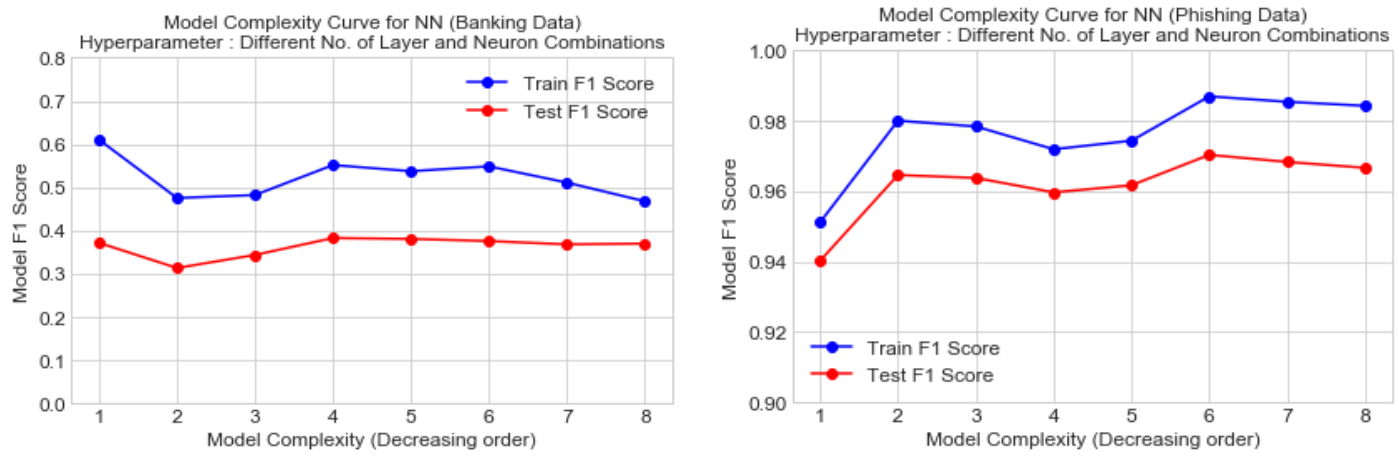


Fig.2 Validation Curve for different layer structures and neurons – Phishing and Banking data

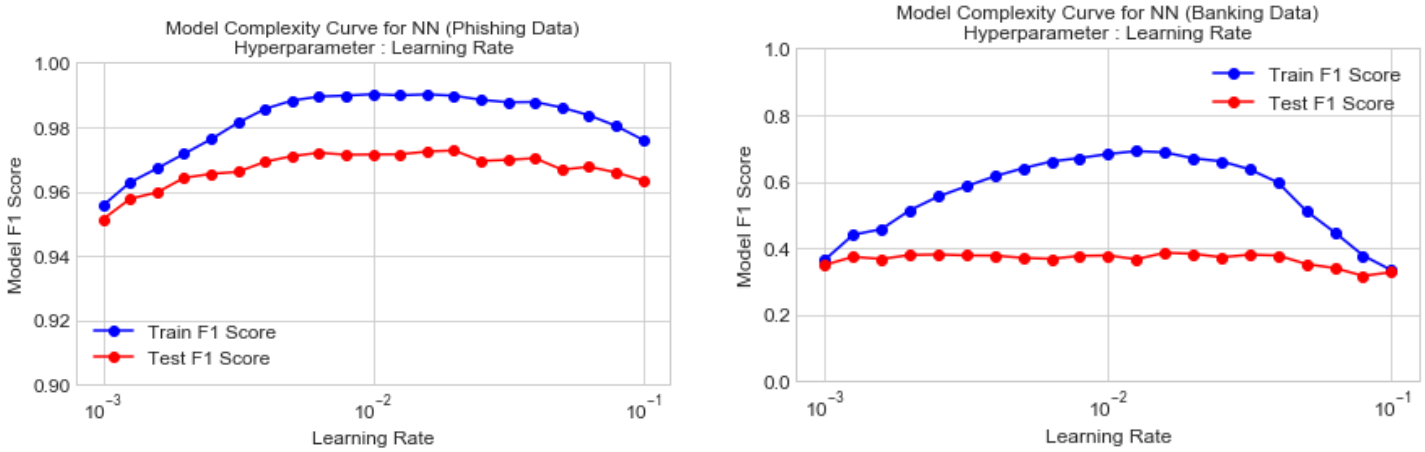


Fig.3 Validation Curve for different learning rates – Phishing and Banking data

We can observe from above that as the number of neurons (and hence the model complexity) increases, the model accuracy *(we refer to F1 score for each algorithm as the accuracy of the model, unless otherwise stated)* saturates for both training and validation sets. Also, the layer structures being used for Phishing and Banking data are different, because of the difference in the size of dataset. Phishing has more data, so a larger network performs better as compared to Banking data which is smaller in size and needs smaller network. Similarly, from the learning curve we can observe that when learning rate is too high or too low, then performance goes down because the network cannot converge. Below are the Learning curves for the two datasets, along with the fit times and loss curve.

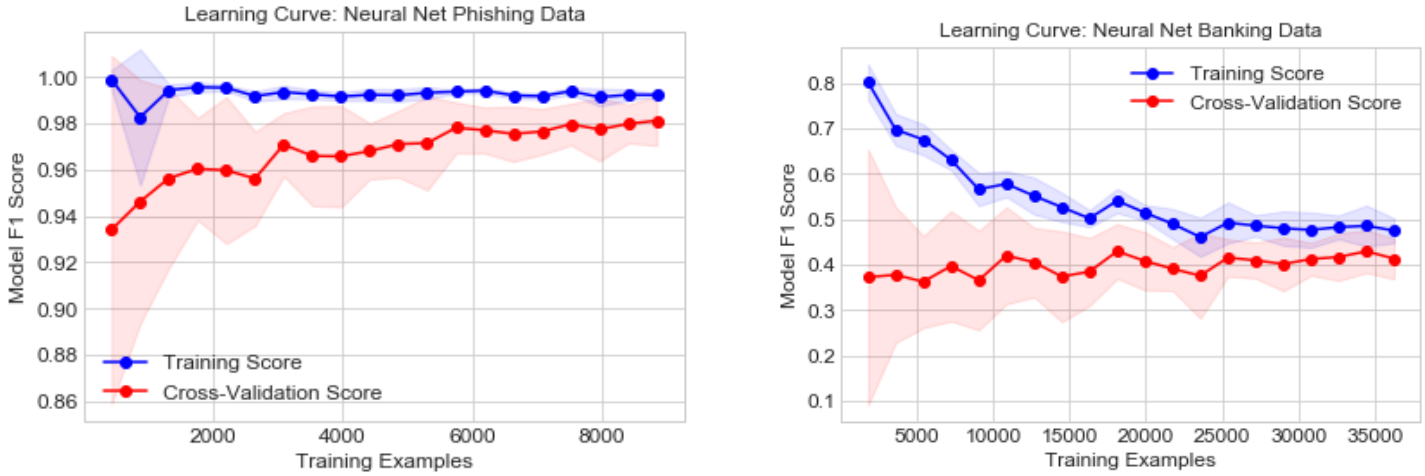
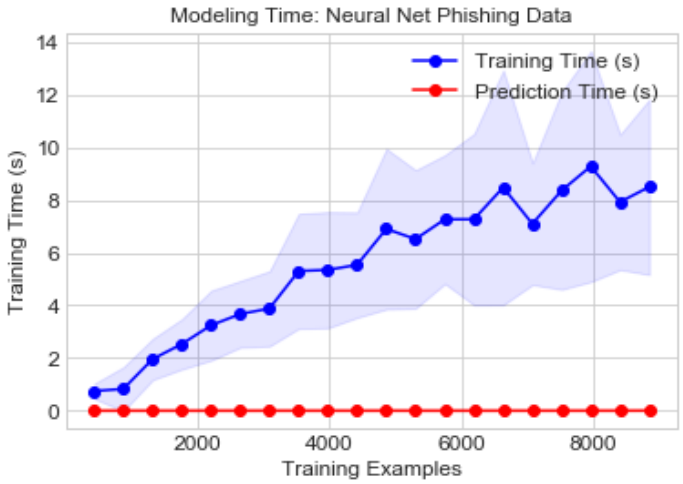


Fig.4 Learning Curves – Phishing and Banking data



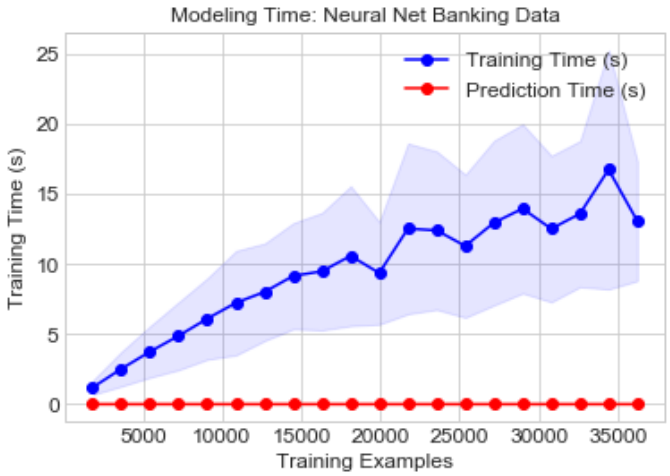


Fig.5 Modeling/Fit Times – Phishing and Banking data

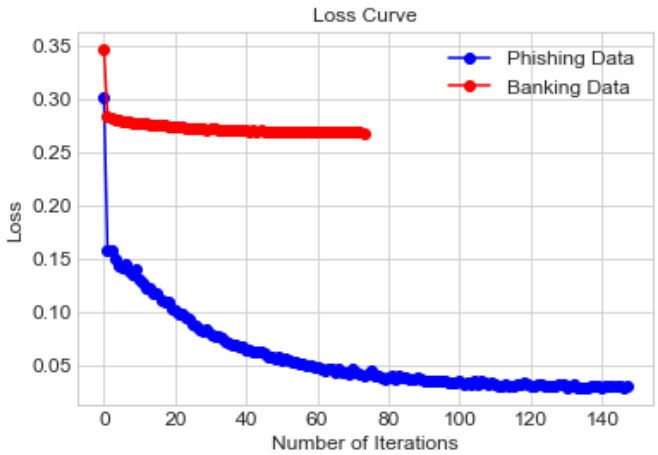


Fig6 Loss Curve

We can observe from the plots of Learning curve that for the Phishing data, the train and test plots converge at a high value, indicating that there is neither bias nor variance in the model and the classifier is performing great. In case of Banking data, they converge at a lower value indicating high bias (underfitting). This can also be seen from the loss curve, where the training loss is lower for Phishing model than Banking model, indicating that Banking data is suffering from high bias. This can be improved by addition of more layers or increasing no. of neurons in the layers so that the network can learn better. The plot of the modeling time just indicates that once the model gets trained, the predictions are done pretty quick as compared to the time spent in training.

Another quick summary of the results:

Metric	Phishing Data	Banking Data
Model Training Time (s)	6.075	5.6
Model Prediction Time (s)	0.004	0.005
Train F1 Score	0.99	0.48
Validation F1 Score	0.98	0.40
Test Accuracy	0.97	0.89
Test F1 Score	0.98	0.31
Optimal Layer Structure	(100,)	(10,)
Optimal Learning Rate	0.015	0.0158

In further analysis of other algorithms, we will not show the modeling time plots, since for all the algorithms, training takes time while prediction happens pretty quickly. Also, we will not show the validation curves for each of the

algorithms, for the sake of compactness of the report, but throughout we will assume that the intervals for Grid Search in each of the algorithms was obtained by first obtaining a validation curve.

Support Vector Machine (SVM)

While building the SVM, the major hyperparameters checked were the kernel function (among 'rbf', 'linear', 'polynomial' of degrees 2 through 8, and 'sigmoid'), penalty term 'C' and kernel coefficient 'gamma'. For deciding the kernel function, below was the plot obtained for F1 score VS kernel:

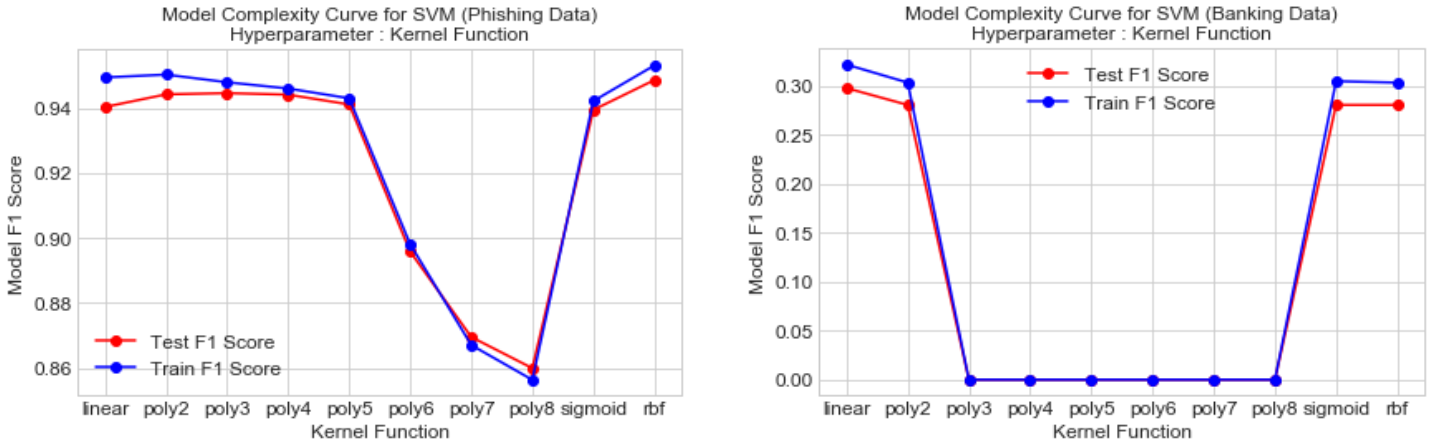


Fig 7 Validation curve for kernel function – Phishing and Banking Data

So, we can see that for Phishing data 'rbf' was used and for Banking data 'linear' function was used (though, we could use 'rbf' too since F1 score for that is slightly lesser than 'linear'). Now, Grid Search was performed to obtain the optimal value of 'C' and 'gamma' and then the learning curve was plotted as shown below (it was observed that overfitting occured for larger values of 'C'):

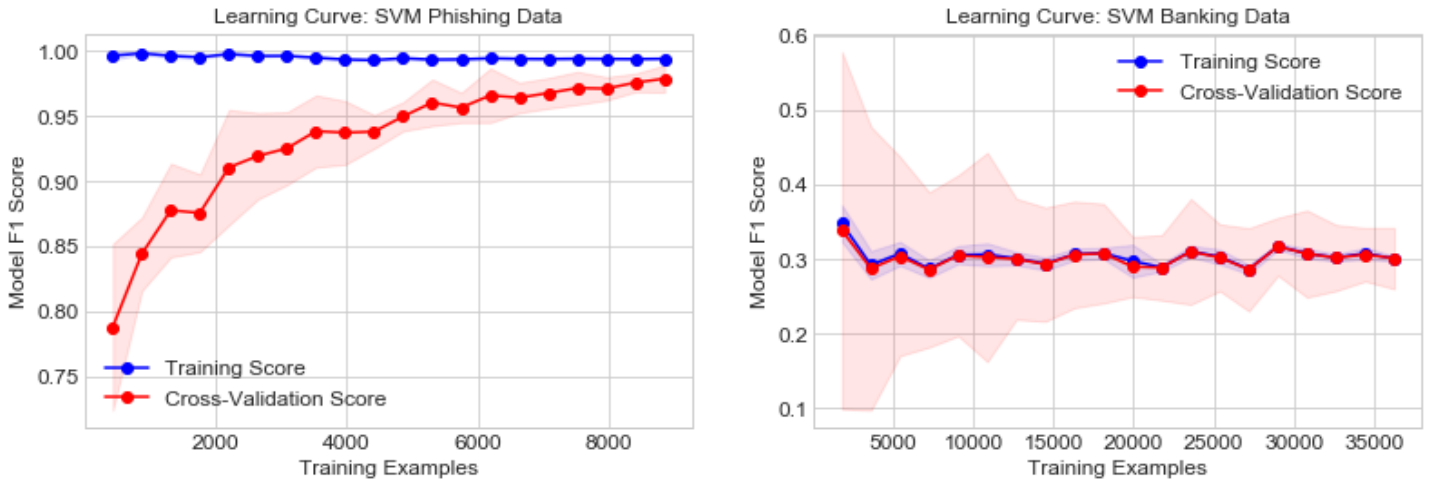


Fig 8 Learning Curve – Phishing and Banking Data

As we can observed, for the Phishing data the model performed well, with the training accuracy (again, we mean F1 score) around 1 and the validation accuracy increased, approaching the train accuracy for higher number of training examples. Thus the model did not suffer from overfitting or underfitting. For the Banking data, the plot converges as a lower value, indicating underfitting. This could be improved if better features are taken into consideration. It also indicates that the data might not be linearly separable, and that class imbalance exists (since accuracy is 90%, as shown in the below table, which can happen only if lots of 0s are being predicted correctly). After hyperparameter tuning, the following summary table was obtained:

Metric	Phishing Data	Banking Data
Model Training Time (s)	5.09	107.47
Model Prediction Time (s)	0.68	7.7
Train F1 Score	1	0.3
Validation F1 Score	0.98	0.3
Test Accuracy	0.97	0.9
Test F1 Score	0.97	0.29
Optimal C	10	1
Optimal gamma	1	1

Although, the F1 score is low (0.29) for the Banking data, we can always say that the model accuracy is higher, and that the model still predicts the chance of response being 0 with a higher confidence (since the F1 score for that will be more than 80%). The class imbalance can be seen from the confusion matrix below:

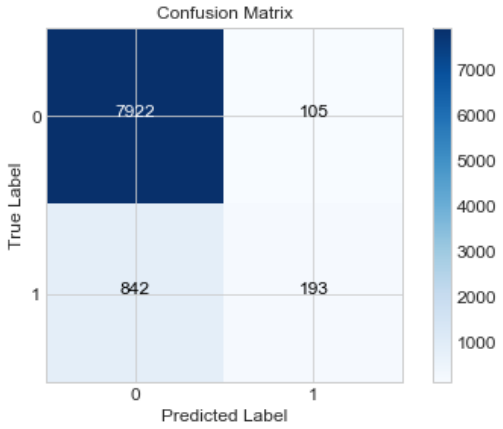


Fig. 9 Confusion matrix for SVM on Banking Data

k-Nearest Neighbors

For the sake of simplicity the hyperparameter we chose was the number of neighbors (distance metric used can also be chosen as a hyperparameter but for faster execution of code, it has been avoided for this analysis). The validation curve for variation of the F1 score with the number of nearest neighbors is shown below:

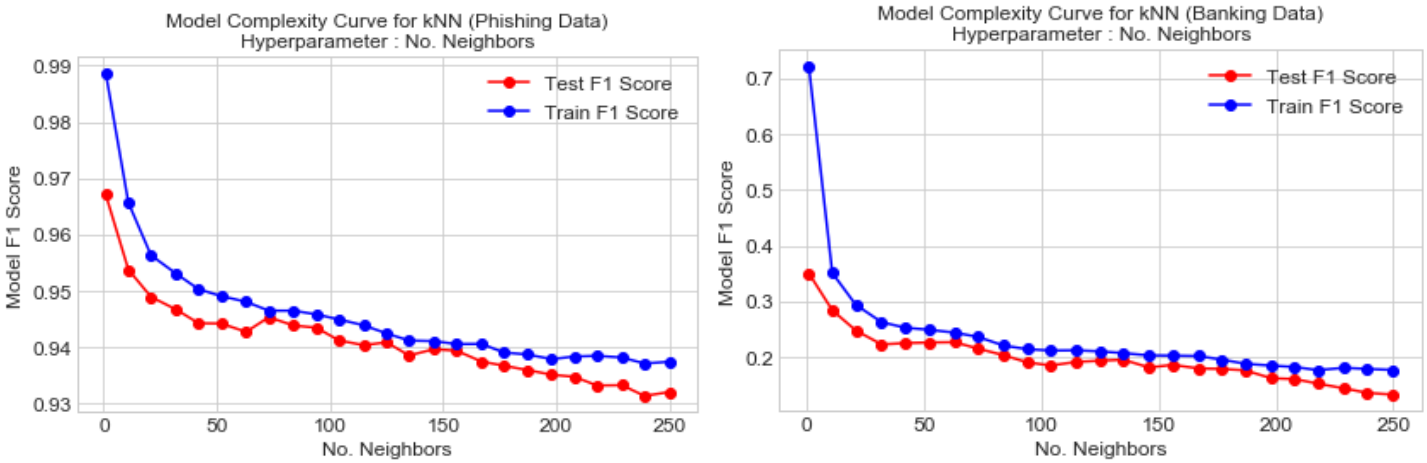


Fig 10 Validation curve for number of neighbors – Phishing and Banking Data

From above, we can see that with for small values of k, training accuracy was high and validation accuracy was low, thus indicating an overfitting/ high variance. As we move towards right, with higher k, the accuracies became low and converged but started diverging again at lower values itself, indicating that the model became prone to high bias/underfitting. Thus, the optimal number of neighbors chosen for Phishing data was 40 and for Banking data it was 10 (elbow method). Learning curve is shown below:

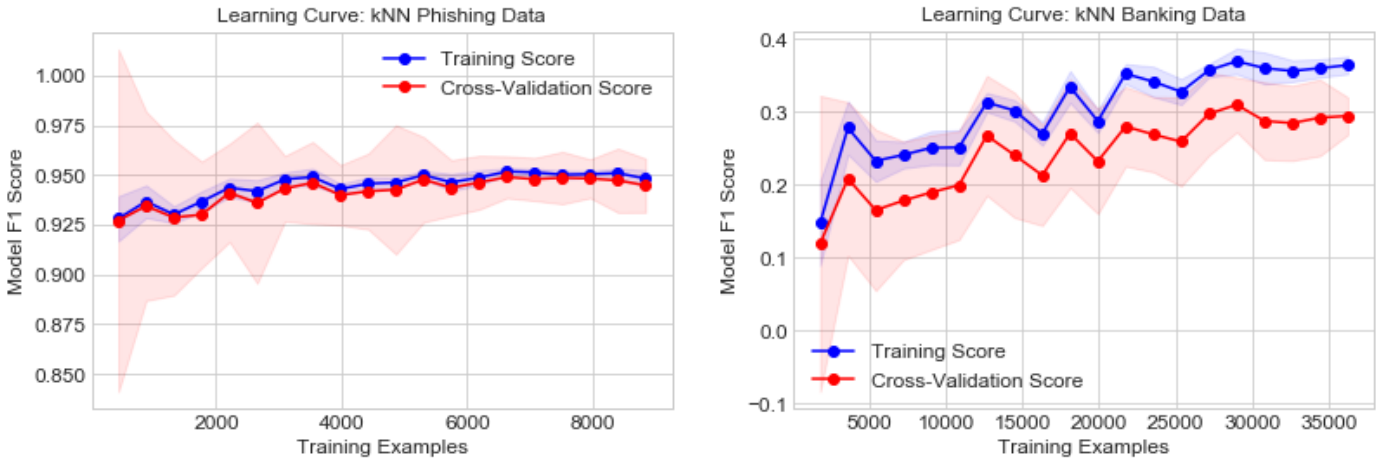


Fig 11 Learning Curve – Phishing and Banking Data

From the learning curve it is clear that model for Phishing data performed better, and the training and validation accuracies converged at higher value. For Banking data, the accuracies were low and still increased for both train and validation with increase in training examples, indicating that the model was prone to high bias/underfitting and we need to come up with better set of predictive features or maybe reduce the number of neighbors selected. Another interesting observation is the modeling time for each of the two cases.

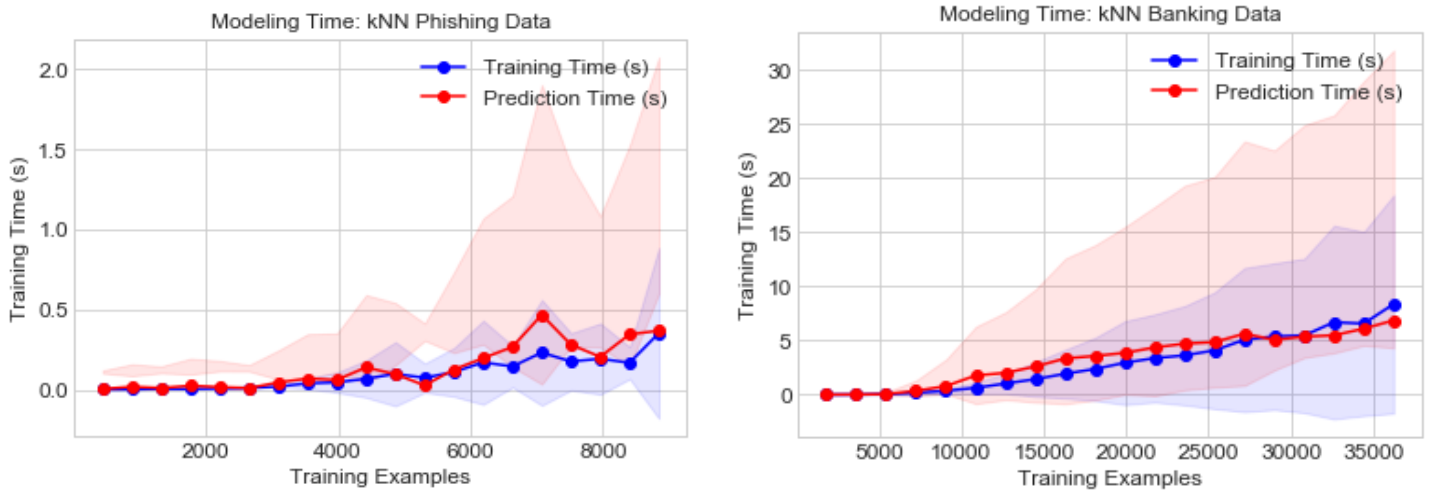


Fig.12 Modeling time plots – Phishing and Banking Data

The interesting bit here is that the Training time for kNN is lesser than the prediction time.

The summary table is shown below:

Metric	Phishing Data	Banking Data
Model Training Time (s)	0.075	1.54
Model Prediction Time (s)	0.55	7.93
Train F1 Score	0.95	0.36
Validation F1 Score	0.949	0.3
Test Accuracy	0.94	0.9

Test F1 Score	0.95	0.26
Optimal k	40	10

Again, due to class imbalance kNN for the Banking Data couldn't perform well, but we can interpret the results in the same was as SVM before, to be more confident of our predictions. There are always ways of improving that by using a different distance metric or by using weighted values from the neighbors for making the classification.

Decision Tree (with pruning)

For building the decision tree, we used the information gain or entropy as a measure to determine the split attribute (ID3 algorithm, implemented personally so more comfortable with the concept behind it than GINI index). The pruning of the tree was done by use of hyperparameters in the model, such as 'max_depth' (the maximum depth of the tree, which if not specified causes the split to happen until the leaves are pure, or some other hyperparameter restricts it) and 'min_samples_leaf' (minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least 'min_samples_leaf' training samples in each of the left and right branches). For Grid Search on these hyperparameters, we vary the size of 'min_samples_leaf' from 0.5% to 5% of the training data and depth of tree was varied between 1 and 20 (from validation curves we know that for low values of 'max_depth', the tree is excessively pruned and hence led to underfitting while at higher values, we see the plots diverging, indicating that at higher tree depths, overfitting occurs). The validation curve is shown below (for 'max_depth'):

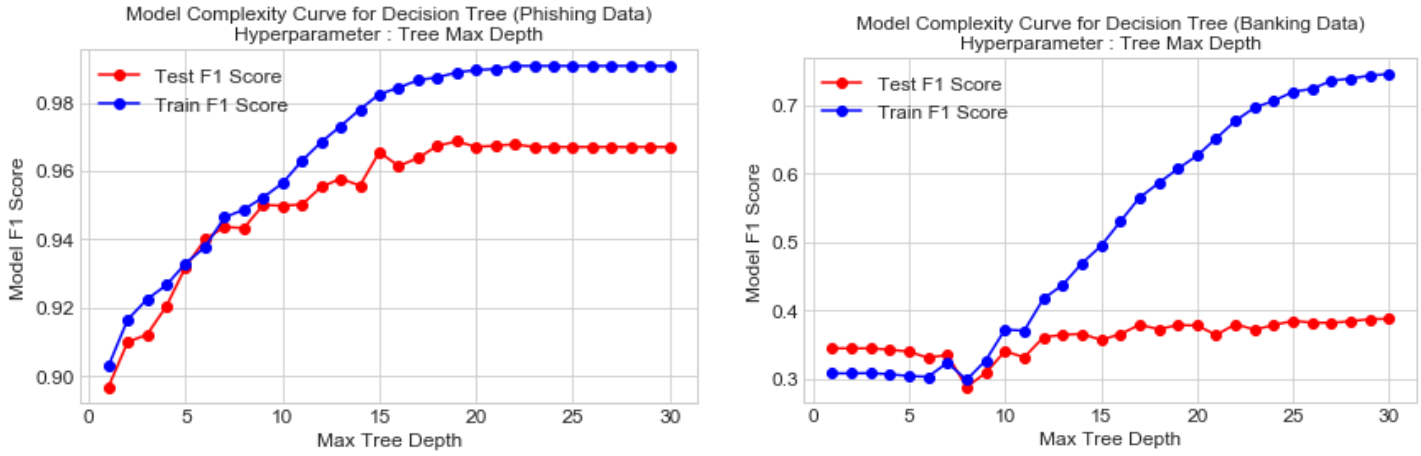


Fig.13 Validation curve for 'max_depth' – Phishing and Banking data

As we can see, for lower values of tree depth, underfitting is predominant and for higher values of tree depth, there is a huge gap between the two plots with higher values of training accuracy indicating the presence of overfitting. The learning curves (using optimal values of 'max_depth' and 'min_samples_leaf') are shown below:

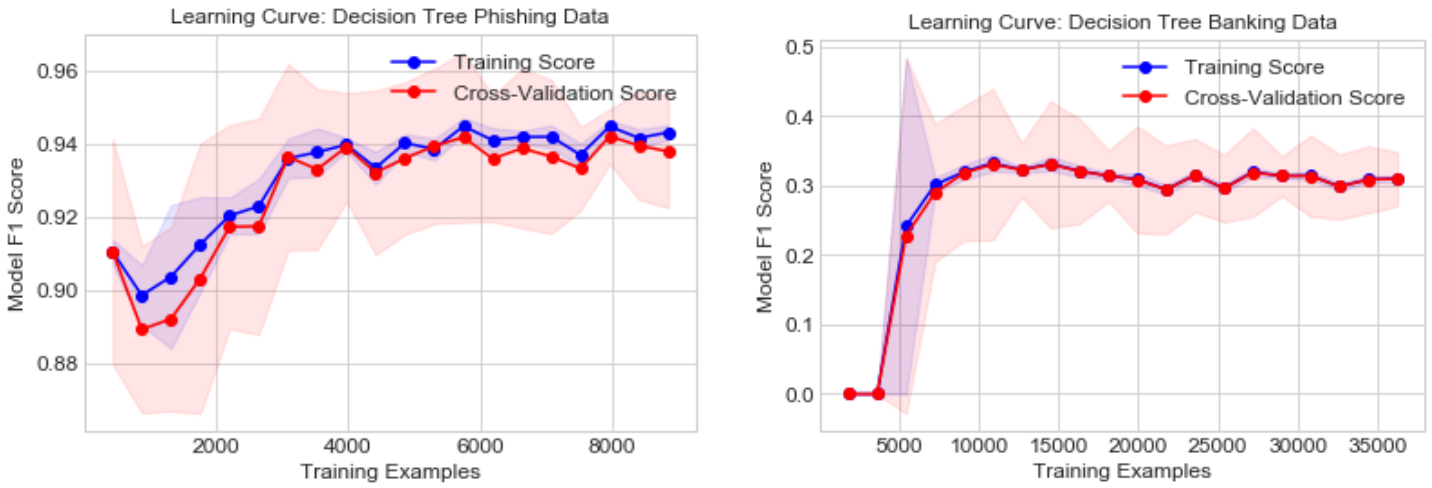


Fig.14 Learning curves – Phishing and Banking data

The summary table is shown below:

Metric	Phishing Data	Banking Data
Model Training Time (s)	0.016	0.03
Model Prediction Time (s)	0.0005	0.002
Train F1 Score	0.99	0.3
Validation F1 Score	0.965	0.3
Test Accuracy	0.93	0.9
Test F1 Score	0.94	0.34
Optimal max_depth	9	1
Optimal min_samples_leaf	42	180

From the learning curve and the table above, we can conclude that the model performs pretty well for Phishing dataset, since the plots converge at a higher accuracy (though if one observes closely they can see that towards the right plot started to diverge indicating chances of overfitting). Again for the Banking data, we see that since it's just 1 depth the tree is allowed to go, the training and validation plots overlap perfectly. This again means that we could use better predictors which can help split the data better.

Boosting

Here, we still use the pruning and the hyperparameters from before, but we could be more aggressive about pruning threshold (since it combines multiple weak learner). To indicate the contribution of each tree (weak learner), other thresholds such as number of estimators and learning rate were used. Since Gradient boosting is fairly robust to over-fitting so a large n_estimators usually results in better performance. Let us check that using validation curve:

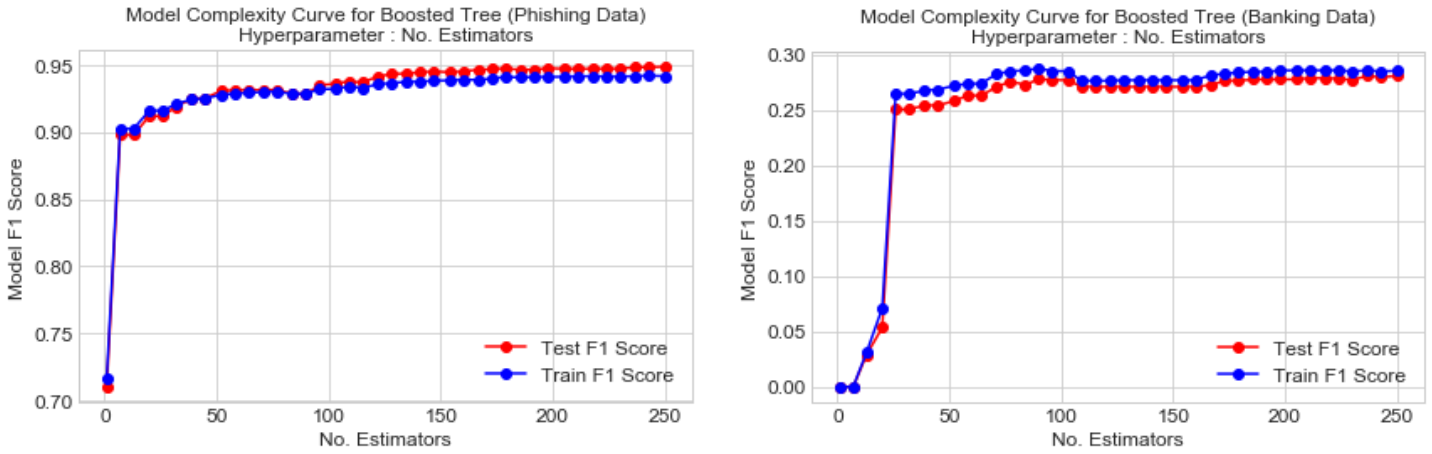


Fig.15 Validation Curve for n_estimators – Phishing and Banking data

Clearly, our knowledge of n_estimators checks out (though Banking data might be experiencing underfit/noise due to class imbalance). Let us look at the Learning curves and summary table:

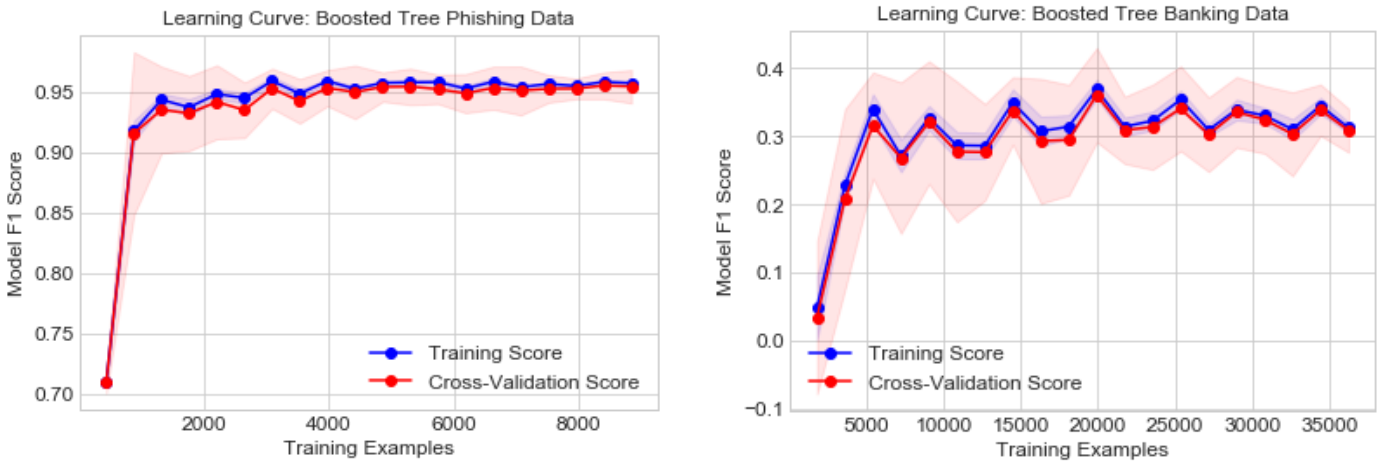


Fig.16 Learning curves – Phishing and Banking data

Summary table:

Metric	Phishing Data	Banking Data
Model Training Time (s)	1	7.1
Model Prediction Time (s)	0.004	0.01
Train F1 Score	0.96	0.3
Validation F1 Score	0.96	0.3
Test Accuracy	0.96	0.89
Test F1 Score	0.96	0.29
Optimal max_depth	3	3
Optimal min_samples_leaf	241	180
Optimal n_estimators	100	100
Optimal learning_rate	0.1	0.1

As we can observe from the Phishing data, we were able to be more aggressive with reducing out depth even further, and have more samples in the leaf node, increasing the accuracy in the process. Problems with Banking data persist, due to the class imbalance and lack of better feature set.

Conclusion and Future Scope

Conclusive summary is provided below:

- For Phishing Dataset

	Model Training Time (s)	Model Prediction Time (s)	F1 Score
Decision Trees	**0.016**	**0.0005**	**0.94**
Neural Network	**6.075**	0.004	**0.98**
Boosting	1	0.004	0.96
SVM	5.09	**0.68**	0.97
kNN	0.075	0.55	0.95

Since Neural Networks is the most powerful model, it also has the best accuracy. For the same reason that its powerful, it is also complex which causes highest model training and prediction times. Decision tree results in smaller accuracy since it is the cheapest and quickest, and hence least model training and prediction times too. Next best predictors include SVM and Boosting algorithms, which are expected to be better than kNN since Phishing Websites dataset contains mostly categorical binary variables.

- For Banking Dataset

	Model Training Time (s)	Model Prediction Time	F1 Score
Decision Trees	**0.03**	**0.002**	**0.34**
Neural Network	5.6	0.005	0.31
Boosting	7.1	0.01	0.29
SVM	**107.47**	**7.93**	**0.29**
kNN	1.54	7.7	0.26

Here, we see that overall F1 score is low, but the best F1 score is obtained by Decision Trees, since the data consists of both numerical and categorical data, decision tree could easily split and do it quicker than the worst algorithm for this dataset, which is SVM (chances are that the data might not be linearly separable) which is the slowest. Also, Neural network is still powerful to handle such dataset and has the next best accuracy after Decision tree, at decent training and prediction times. We focused on F1 score as an accuracy metric because for classification algorithms, accuracy ((True positives + True Negatives)/(Total Data/Rows)) doesn't give a very good idea of the classification in case of class imbalances. Hence, it was best to use F1 score.

Future scope of this analysis is that the feature set for the Banking data could be better created and picked. A more balanced data should be collected or if one has to work with this dataset, we have to come up with strategies of better handling the class imbalance (maybe we can only model for portion which are mostly 1s, or over-sample in a way that we get more of 1s than 0s while splitting into training, validation and testing datasets).

References

[1] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

[2] S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimaraes, Portugal, October 2011. EUROSIS

[3] Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi (2012) An Assessment of Features Related to Phishing Websites using an Automated Technique. In: International Conferece For Internet Technology And Secured Transactions. ICITST 2012 . IEEE, London, UK, pp. 492-497. ISBN 978-1-4673-5325-0

[4] Mohammad, Rami, Thabtah, Fadi Abdeljaber and McCluskey, T.L. (2014) Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 25 (2). pp. 443-458. ISSN 0941-0643

[5] Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi Abdeljaber (2014) Intelligent Rule based Phishing Websites Classification. IET Information Security, 8 (3). pp. 153-160. ISSN 1751-8709

[6] Datasets obtained from UCI and OpenML Repository:

- Bank Marketing Data – UCI Repository, OpenML
- Phishing Websites Data – UCI Repository, OpenML