SUPERVISED LEARNING WITH SKLEARN

Bosun Anifowoshe
bosunani@gmail.com

Introduction

This paper presents an analysis on the performance of 5 algorithms in supervised learning - Decision Trees, Decision Trees with Adaptive Boosting, k Nearest Neighbors, Artificial Neural Networks, and Support Vector Machines. These learning algorithms were tested on two unique classification problems – Magic Telescope[1] and Phishing Websites[2].

Data Overview

Magic Telescope: This data set was generated by a Monte Carlo program to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. The binary classification task is to discriminate between patterns caused by primary gammas from those caused by hadronic showers using the properties of the images captured by the telescope.

This data consists of 19,000 observations with 10 *continuous* valued attributes and a binary target attribute. One thing to note with this data is that the target attribute is somewhat unbalanced i.e. 65% of the observation belong to the primary gamma class. This could be attributed to the fact that the data set was simulated by a Monte Carlo program causing some of the hadronic showers class to be underestimated. It will be interesting to see how the various machine learning algorithm handle this artifact in the data set. To compensate for the unbalanced nature, we may need to use different model evaluation metrics beyond accuracy such as a ROC curve and possibly different hyperparameters to train the models. In addition, this data set is also interesting because we are trying to use some learning algorithm to model the results obtained by another algorithm.

Phishing Websites: This data was collected by a group of researchers to shed light on the important features that have proved to be sound and effective in predicting phishing websites. The binary classification task is to discriminate if a website is legitimate or not using the properties of the website such as URL length, appearance, etc.

This data consists of 11,000 observations with 30 *discrete* valued attributes and a binary target attribute. In contrast to the Magic Telescope data set, this data is relatively balanced i.e. 56% of the observations belong to illegitimate webpage class. One interesting thing to note with this data is that in addition to the researchers gathering the data, they also added some new features which they think will be important in improving the classification of the phishing websites. The upshots of this data gathering approach are (1) since humans are prone to errors, there is a potential that this data will be prone to noise which might lead to overfitting (2) the added features might improve the predictive power of the model. It will be interesting to see how the various machine learning algorithm handle this artifact in the data set. Furthermore, several of the attributes in this data set are categorical with the levels {-1,0,1}. The data is preprocessed to using one-hot encoding to create dummy features with level {0,1}. This further increased the number of discrete attributes that will be used by the learning algorithm to 46 resulting in this data set having quadruple number of attributes compared to the Magic Telescope data set and it will be interesting to see how the learning algorithms handle this data in relation to the curse of dimensionality.

Methodology

All implementation of the 5 learning algorithms is performed using Python's scikit-learn module. Each data is split into two sets – 70% used for training, 30% for testing. The training set is used to tune each algorithm, and the test set to evaluate the final performance of the model. In addition, the 70% training set, is further split into 70-30 sets for hyperparameter tuning and testing respectively using GridSearchCV. Hyperparameters are tuned through 10-fold cross validation to remove the bias of selecting a model that performs well on training data, but does not generalize well. Furthermore, the 70% training data is further split in portions (5% - 100%) to generate learning curves by training and cross validating on each portion of the data. A detailed analysis and learning curves will be given for each of the learning algorithms. The performance of each learning algorithm will be based on F1 score, accuracy, AUC, precision, and recall. F1 score is chosen as the primary performance metric because F1 is usually more useful than accuracy, especially when we have an uneven class distribution as observed in the Magic Telescope data.

Decision Tree

In this analysis, the decision tree classifier is built using information gain based on Gini impurity measure to split the nodes at the most informative attributes. The Gini impurity can be understood as a criterion to minimize the probability of misclassification and typically yield very similar results to the Entropy impurity measure. Although scikit-learn module does not implement post-pruning, we will pre-prune the tree by setting a limit for the maximal depth of the tree to prevent the decision tree from growing too large and possibly overfitting the noise in the data.

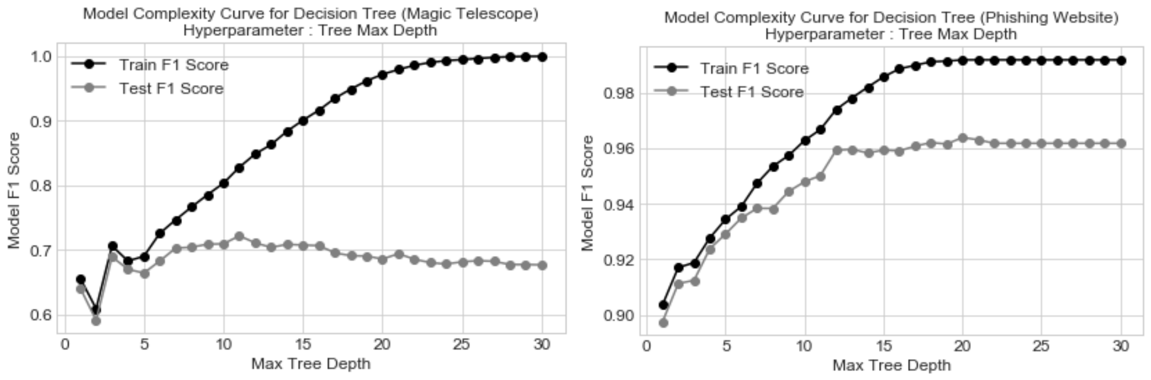


Figure 1: Model Complexity Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

Figure 1 illustrates the impact of overfitting on the Magic Telescope (MT) and Phishing Website (PW) data set. The black curve shows the F1 score of the decision tree over the training set, whereas the grey curve shows F1 score measured over an independent set of test set. As observed, the F1 score of the tree over the training set increases monotonically and then flattens out as the tree is grown. However, the F1 score measured over the independent test set first increases, then decreases before flattening out. Once the tree size exceeds approximately max tree depth of 11 for the MT data and 20 for the PW data, further elaboration of the tree decreases the F1 score over the test set despite increasing its F1 score on the training set. This phenomenon occurs due to the fact that the training data contain random errors or noise that the decision tree is trying to fit thereby causing overfitting.

Next, using the max tree depth for both data set as shown above, learning curve were generated by training and cross validating on portions of the training set as shown in Figure 2 below. The black curve shows the F1 score of the decision tree over the training set, whereas the grey curve shows F1 score measured by performing cross validation on the training set.

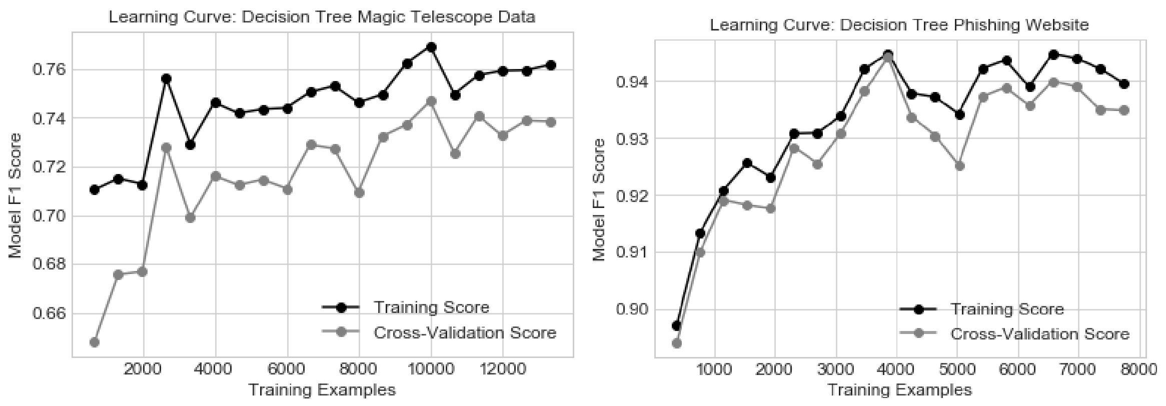
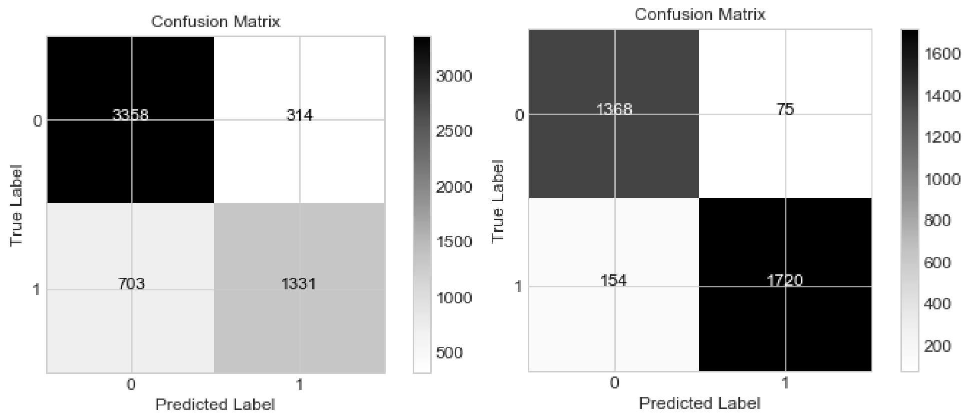


Figure 2: Model Learning Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

The learning curve tells us how much we benefit from adding more training data and whether the estimator suffers more from a variance error or a bias error. Ideally, with smaller training sets, we expect the training error will be low because it will be easier to fit to less data. So as training set size grows, the average training set error is expected to grow. Conversely, we expect the average validation error to decrease as the training set size increases. For MT data set shown in Figure 2 above, we observe both the training and validation score increasing with training examples, however the training error is much lower than the validation error. This is an indication of high variance in the model which can possibly be remedied by adding more training data if available, reducing model complexity, or by utilizing bagging or boosting model which we will try in subsequent section. For the PW data set, we observe both the training and validation score increasing with training examples and then flattens out around 4000 samples before decreasing towards the end of the chart. Both the training and validation errors approximately matches each other which is indicative of a low variance and high bias model, hence adding more training examples will not help much and we might have to use a more complex model or boosting to improve model performance.

Finally, the model performance is examined for both data sets by building the model using the 70% training data and evaluating on the untouched 30% test set. Figure 3 below shows the confusion matrix and summary of the model evaluation metrics.



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.02155
Model Prediction Time (s): 0.00056

F1 Score:  0.72
Accuracy:  0.82     AUC:       0.78
Precision: 0.81     Recall:    0.65
*****************************************************
```

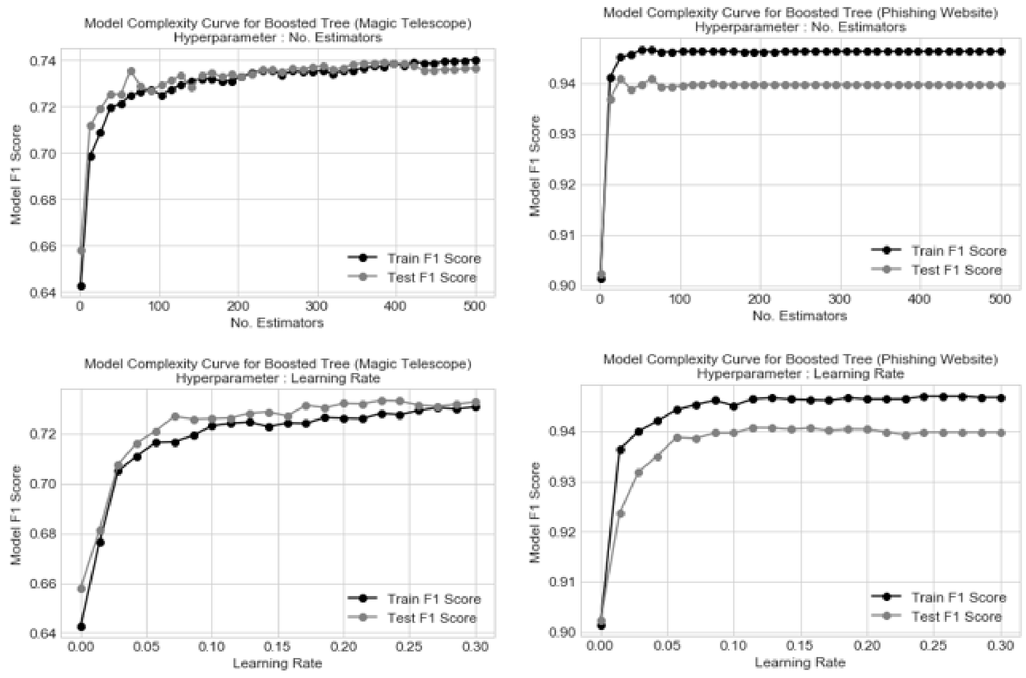
```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.01313
Model Prediction Time (s): 0.00050

F1 Score:  0.94
Accuracy:  0.93     AUC:       0.93
Precision: 0.96     Recall:    0.92
*****************************************************
```

Figure 3: Model Evaluation Metric for – Left: Magic Telescope Data, Right: Phishing Website Data

Comparing both models, the decision tree model performed better on the PW data set compared to the MT data set on all metric – F1 score, accuracy, AUC, precision, and recall even though the PW model is more complex (higher tree depth). This is potentially due to the fact the PW model uses more discrete attributes and the target attribute is more balanced than the MT data set. In addition, for both data set, the training time is much higher than the time it takes for testing which makes sense because the ID3 algorithm used is an eager learner.

Boosting

The AdaBoost algorithm with decision tree stumps as weak learners to improve the performance for the classification problem. Generally speaking, boosting can lead to a decrease in bias as well as variance however, some boosting algorithms such as AdaBoost are also known for their high variance, that is, the tendency to overfit the training data, hence some form of pre-pruning was performed by limiting the number of sequential trees to be modeled and the learning rate which determines the impact of each tree on the final outcome. If the learning rate is too large, the algorithm will overshoot the global cost minimum. If the learning rate is too small, the algorithm requires more epochs until convergence, which can make the learning slow—especially for large datasets.



```
Per Hyperparameter tuning, best parameters are:
{'learning_rate': 0.2857, 'n_estimators': 446}
```

```
Per Hyperparameter tuning, best parameters are:
{'learning_rate': 0.2143, 'n_estimators': 282}
```

Figure 4: Model Complexity Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

Figure 4 shows the model complexity curves for both the MT and PW data sets. The top plots show the hyperparameter tuning for the number of estimators while the bottom plot show the learning rate. As observed, for the MT data, the F1 score of the tree over both the training and test set increases and flattens out as the number of estimators increases. At number of estimators of about 400, we see that the F1 score of the training set continue to increase slightly while the score of the test set flattens out. This implies that model is starting to overfit beyond the that point. Similarly, for the PW data, the F1 score of the tree over both the training and test set increases and flattens out as the number of estimators increases. At number of estimators of about 50, we see that the F1 score of the both the training and test sets flattens out. In addition, we see that the learning rate for the MT and PW data approaches their asymptotes around 0.2 and 0.1 respectively.

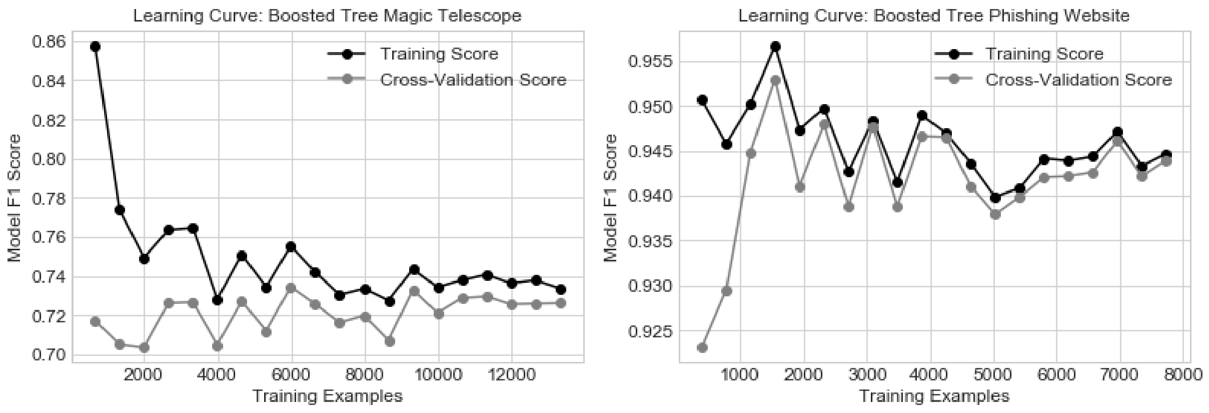
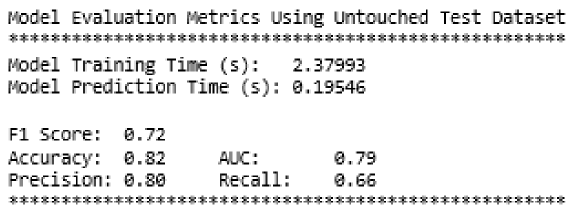
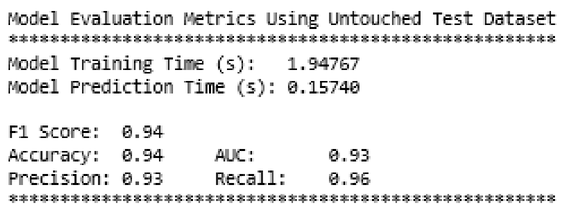


Figure 5: Model Learning Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

Next, we performed hyperparameter tuning to select the best combination of number of estimators and learning rate which resulted in 446 and 0.29 respectively for the MT data and 282 and 0.21 for the PW data. These parameters were used in generating learning curve by training and cross validating on portions of the training set as shown in Figure 5 above.



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   2.37993
Model Prediction Time (s): 0.19546

F1 Score:  0.72
Accuracy:  0.82     AUC:       0.79
Precision: 0.80     Recall:    0.66
*****************************************************
```



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   1.94767
Model Prediction Time (s): 0.15740

F1 Score:  0.94
Accuracy:  0.94     AUC:       0.93
Precision: 0.93     Recall:    0.96
*****************************************************
```

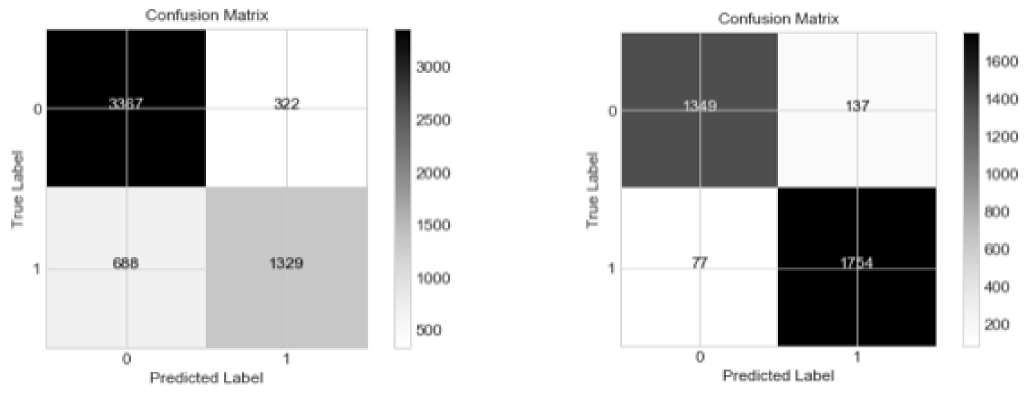


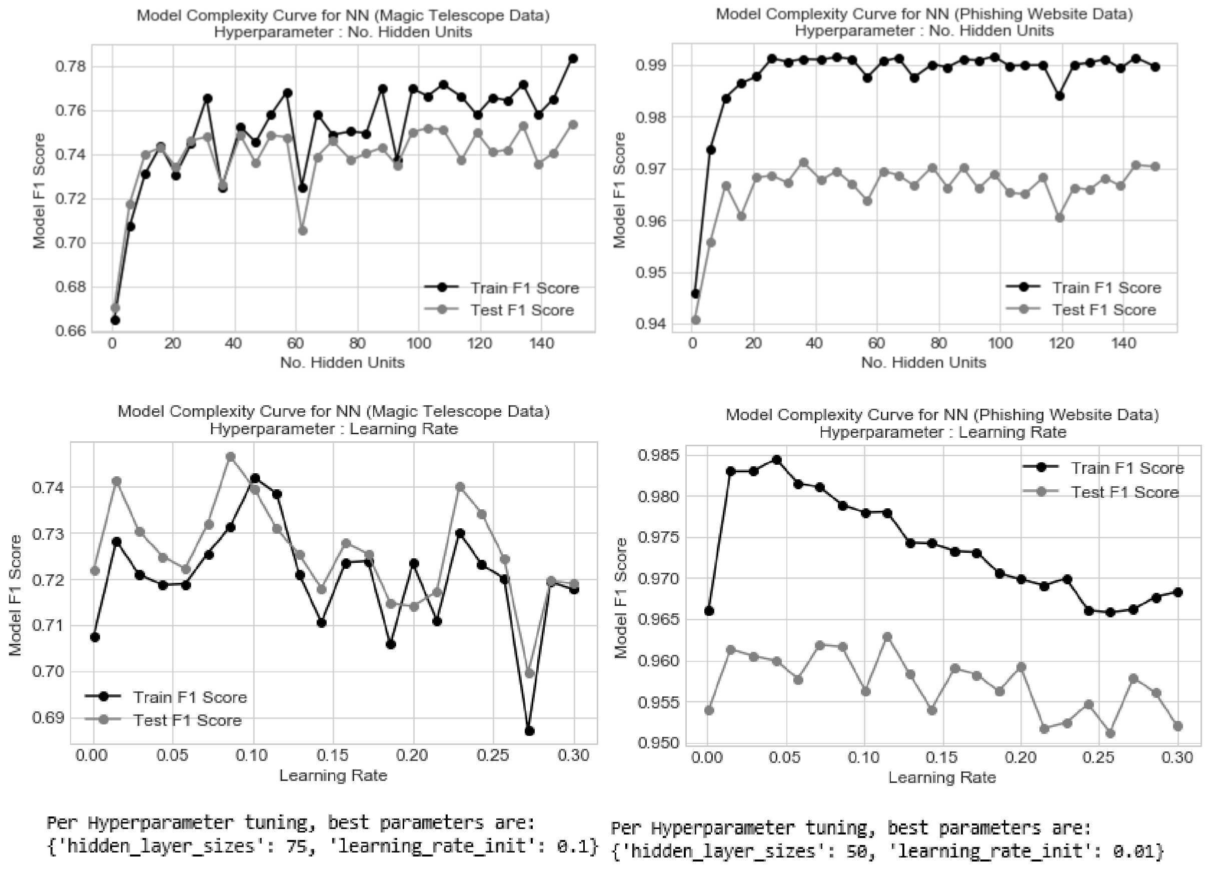
Figure 6: Model Evaluation Metric for – Left: Magic Telescope Data, Right: Phishing Website Data

For MT data set shown in Figure 5 above, we observe the training score decreasing, the validation score increasing until about 6000 samples before both curves somewhat converges and flattens out. This is an indication of high bias in the model which is expected because this model only uses 10 features and this effect can possibly be remedied by adding more features to the model or using a more complex model. For the PW data set, we observe similar phenomena until about 4000 samples where both the training and validation errors approximately matches each other which is indicative of a low variance and high bias model, hence adding more training examples will not help much and we might have to use a more complex model to improve model performance.

Finally, the model performance is examined for both data sets by building the model using the 70% training data and evaluating on the untouched 30% test set. Figure 6 below shows the confusion matrix and summary of the model evaluation metrics. Comparing both models, the boosted decision tree model performed better on the PW data set compared to the MT data set on all metric – F1 score, accuracy, AUC, precision, and recall. This is potentially due to the fact the PW model uses more discrete attributes and the target attribute is more balanced than the MT data set. In addition, for both data set, the training time is much higher than the time it takes for testing which comes from the fact that the algorithm is building multiple trees during training and then combining them into a single classifier for testing. The final models built with boosted trees perform similar to the final decision tree models. This suggests that for these two datasets the key benefit of the boosted version is that it can be trained on fewer examples and possibly have less overfitting however, it is computationally more expensive than the decision tree model.

Neural Networks

The multi-layer perceptron back-propagation algorithm with a sigmoid activation function was used to generate the forward feed neural network models for the classification problems. For this analysis, a network structure with one hidden layer was used while the number of hidden units and learning rate were the main hyperparameters tuned to optimize the model. The attributes were standardized also to help the algorithm run faster. Figure 7 below shows the model complexity curves for both the MT and PW data sets. The top plots show the hyperparameter tuning for the number of hidden units while the bottom plot show the learning rate.



```
Per Hyperparameter tuning, best parameters are:
{'hidden_layer_sizes': 75, 'learning_rate_init': 0.1}
```

```
Per Hyperparameter tuning, best parameters are:
{'hidden_layer_sizes': 50, 'learning_rate_init': 0.01}
```

Figure 7: Model Complexity Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

As observed, for the MT data, the F1 score of the tree over both the training and test set increases and flattens out as the number of hidden unit increases. At number of hidden units of about 90, we see that the F1 score of the training set continue to increase slightly while the score of the test set flattens out. This implies that model is starting to overfit beyond the that point. Similarly, for the PW data, the F1 score of the tree over both the training and test set increases and flattens out as the number of hidden units increases. At number of hidden units of about 20, we see that the F1 score of the both the training and test sets flattens out. In addition, we see that the learning rate for the MT and PW data take a nose dive at around around 0.1 and 0.05 respectively for both the training and test set. This is because the error surface computed by gradient descent contains only a single global minimum, thus, this algorithm will converge to a weight vector with minimum error. As the learning rate continues to increase, the gradient descent search runs the risk of overstepping the minimum in the error surface rather than settling into it which is what we observe in the plots above.

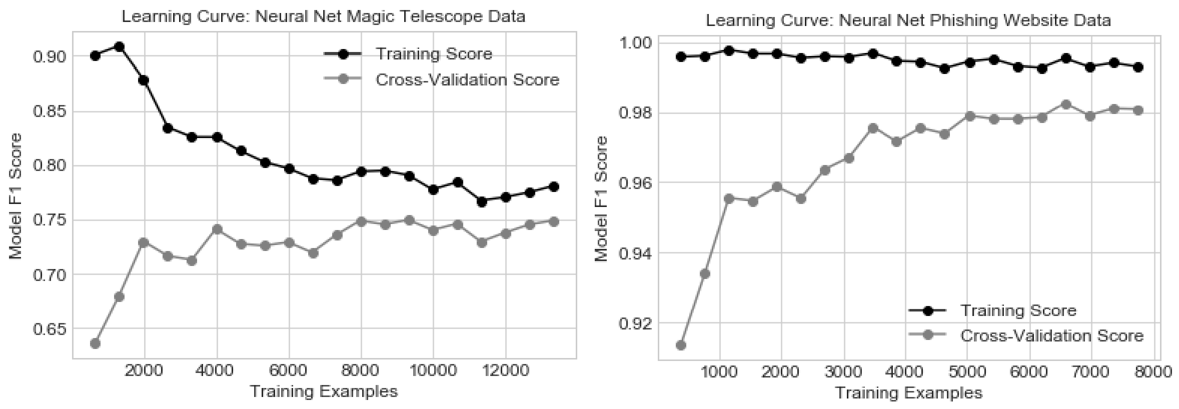
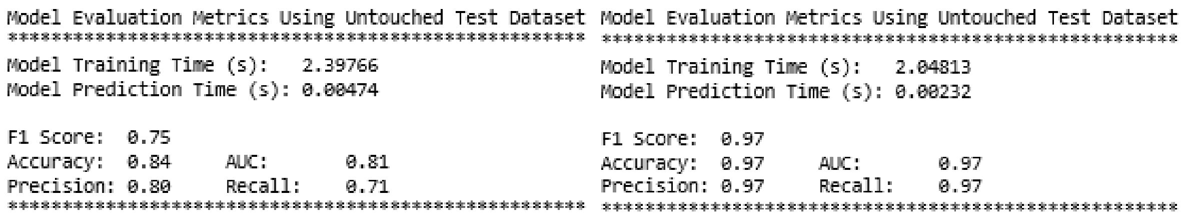


Figure 8: Model Learning Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

Next, we performed hyperparameter tuning to select the best combination of number of hidden units and learning rate which resulted in 75 and 0.1 respectively for the MT data and 50 and 0.01 for the PW data. These parameters were used in generating learning curve by training and cross validating on portions of the training set as shown in Figure 8 above. For both datasets, we observe the training score decreasing and the validation score increasing until about 8000 samples for the MT data set and 5000 samples for the PW data set before both curves somewhat tries to converge and flatten out. This is an indication of low variance, high bias model as we add more training examples, hence adding more training examples will not help much and we might have to use a more complex model to improve model performance.



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   2.39766
Model Prediction Time (s): 0.00474

F1 Score:  0.75
Accuracy:  0.84     AUC:       0.81
Precision: 0.80     Recall:    0.71
*****************************************************
```

```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   2.04813
Model Prediction Time (s): 0.00232

F1 Score:  0.97
Accuracy:  0.97     AUC:       0.97
Precision: 0.97     Recall:    0.97
*****************************************************
```

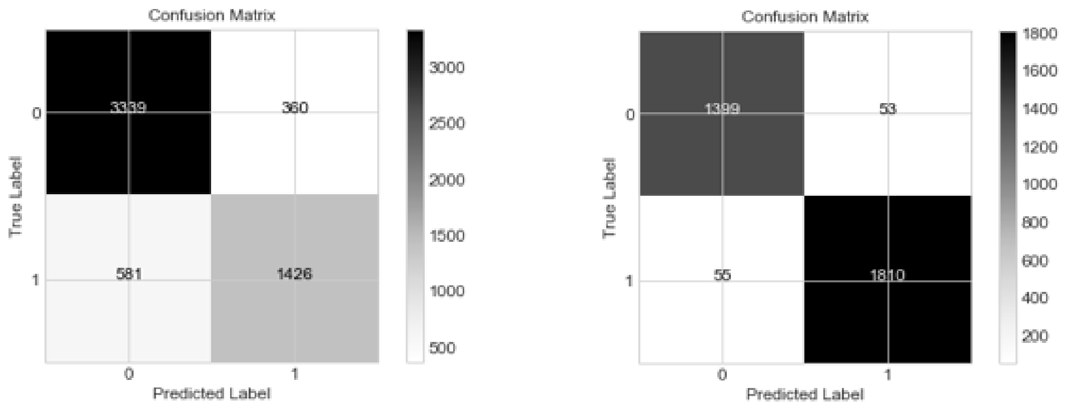


Figure 9: Model Evaluation Metric for – Left: Magic Telescope Data, Right: Phishing Website Data

Furthermore, the model performance is examined for both data sets by building the model using the 70% training data and evaluating on the untouched 30% test set. Figure 9 above shows the confusion matrix and summary of the model evaluation metrics. Comparing both models, the neural network model performed better on the PW data set compared to the MT data set on all metric – F1 score, accuracy, AUC, precision, and recall. This is potentially due to the fact the PW model uses more discrete attributes and the target attribute is more balanced than the MT data set. In addition, for both data set, the training time is much higher than the time it takes for testing which comes from the fact that the back-propagation algorithm is an eager learner.

Finally, Figure 10 below shows the loss curve for both data set using the full training set and a learning rate of 0.1. As observed, we see that the MT model converges faster after about 15 iterations compared with the PW model which converges at about 40 iterations. This is due to the fact that the PW is a more complex model (more attributes) compared to the MT model.

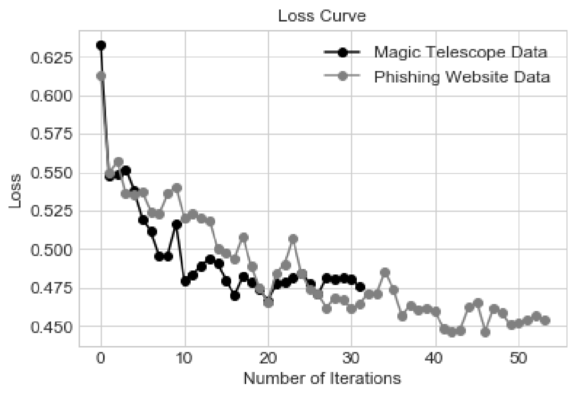
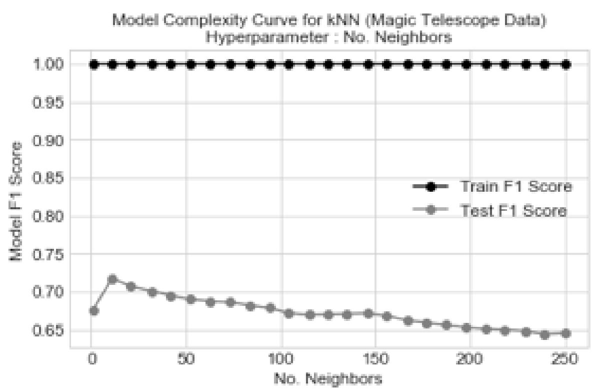


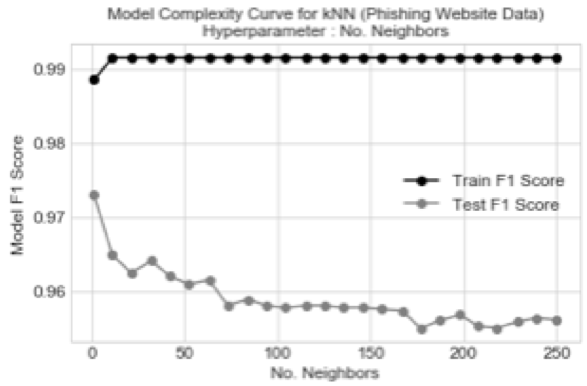
Figure 10: Loss Curve for Both Datasets

k-Nearest Neighbors

The instance-based method k-nearest neighbor (kNN) algorithm was used to generate classifier for both datasets. The number of nearest neighbors, k, as well as two distance functions - Euclidean and Manhattan were explored in this model. Also, the features were standardized so that each feature contributes equally to the distance.



```
Per Hyperparameter tuning, best parameters are:
{'n_neighbors': 21, 'p': 1}
```



```
Per Hyperparameter tuning, best parameters are:
{'n_neighbors': 11, 'p': 1}
```

Figure 11: Model Complexity Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

Figure 11 below shows the model complexity curves for both the MT and PW data sets. For the MT model, we observe F1 score to be relatively constant for the training set, however, for the test set, an initial increase in the F1 score was observed before decreasing at k>10. This is due to the fact that the model starts to fit as the number of neighbors increases. For the PW model, we observe an initial increase in F1 score before it flattens out for the training set, however, for the test set, the F1 score decreases as k increases. This is potentially due to the fact that the PW data has a lot more attributes making the model suffers from curse of dimensionality.

Furthermore, we performed hyperparameter tuning to select the best combination of number of neighbors and distance function which resulted in 21 neighbors and Manhattan distance for the MT data and 11 neighbors and Manhattan distance for the PW data as the optimal hyperparameters. These parameters were used in generating learning curve by training and cross validating on portions of the training set as shown in Figure 12 below.

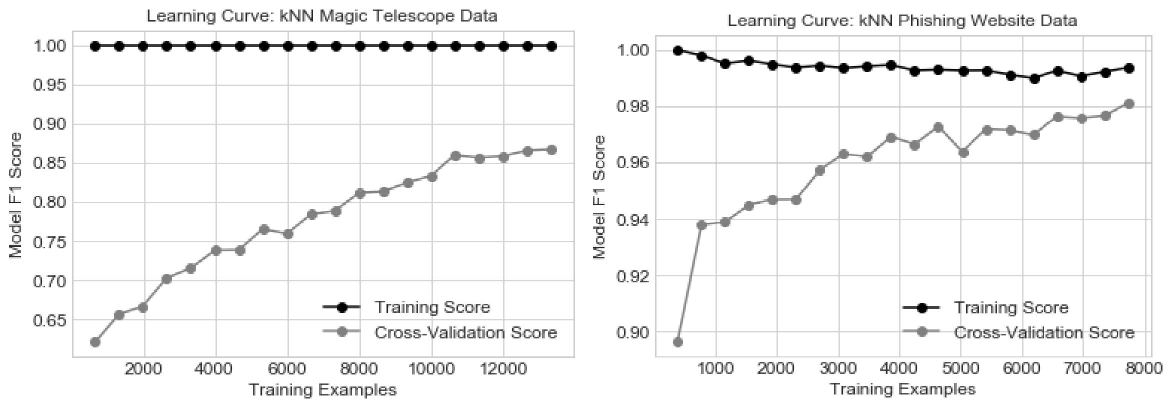
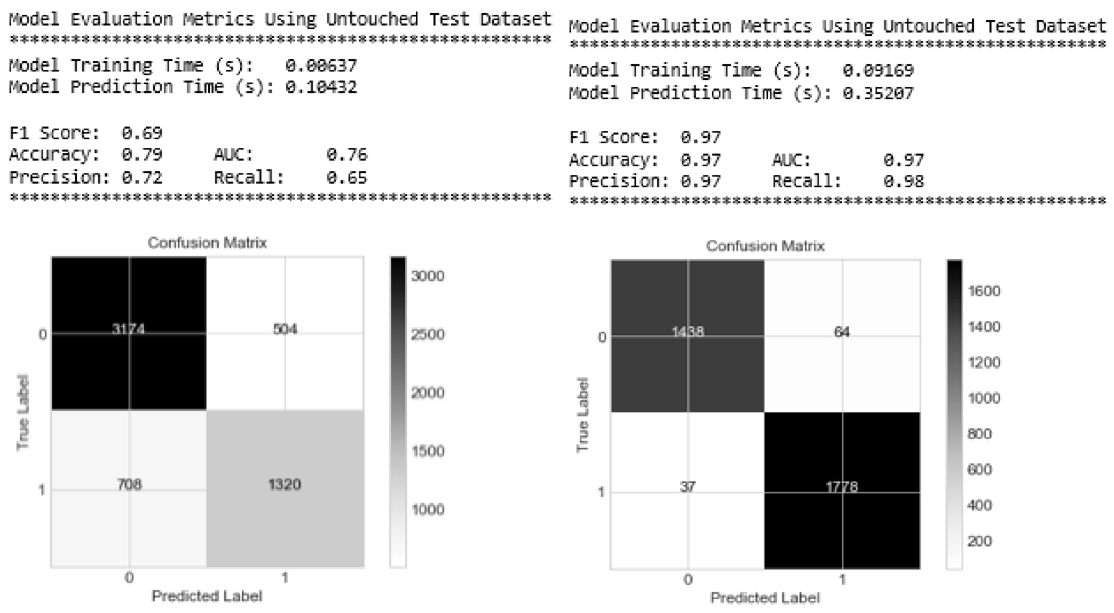


Figure 12: Model Learning Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

For the MT data set, we observe the validation score increasing while the training score remained flat. We also observe a large gap between both curves which is indicative of a high variance model although it improves as we increasr the number of training examples. This model might benefit from adding more data to improve its performance. For the PW data set, we observe that the training score decreases while the validation score increases with increase in the training example and both curves somewhat tries to converge. This is an indication of low variance, high bias model as we add more training examples, hence adding more training examples might not help much and we will have to use a more complex model to improve model performance.



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.08637
Model Prediction Time (s): 0.10432

F1 Score:  0.69
Accuracy:  0.79     AUC:       0.76
Precision: 0.72     Recall:    0.65
*****************************************************
```

```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.09169
Model Prediction Time (s): 0.35207

F1 Score:  0.97
Accuracy:  0.97     AUC:       0.97
Precision: 0.97     Recall:    0.98
*****************************************************
```

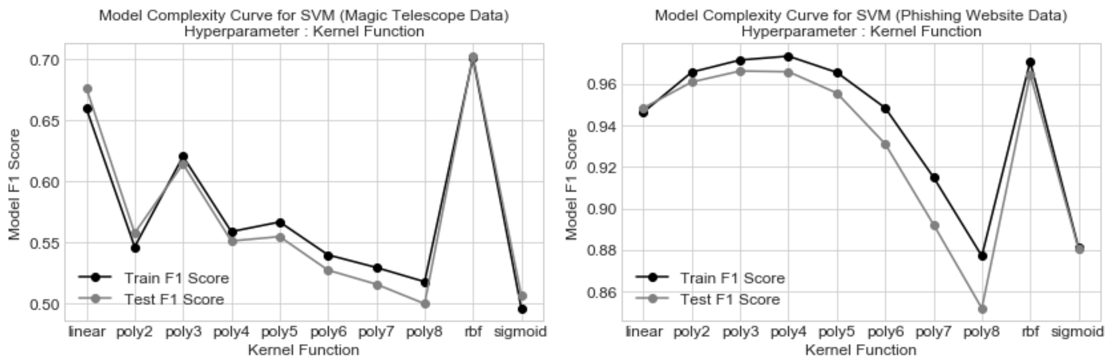
Figure 13: Model Evaluation Metric for – Left: Magic Telescope Data, Right: Phishing Website Data

Furthermore, the model performance is examined for both data sets by building the model using the 70% training data and evaluating on the untouched 30% test set. Figure 13 below shows the confusion matrix and summary of the model evaluation metrics. Comparing both models, the neural network model performed better on the PW data set compared to the MT data set on all metric – F1 score, accuracy, AUC, precision, and recall. This is potentially due to the fact the PW model uses more discrete attributes and the target attribute is more balanced than the MT data set. In addition, for both data set, the testing time is much higher than the time it takes for training which comes from the fact that the kNN algorithm is a lazy learner.

Support Vector Machines

The SVM implementation is based on libsvm package for the classification problems. The misclassification penalty, similarity or kernel function, and the gamma coefficient were explored for both data set. The attributes were standardized to ensure the learner converges more quickly.

Figure 14 below shows the model complexity curves for both the MT and PW data sets. For both models, we observe F1 score initially increases before decreasing as we go to a higher order polynomial. This implies that moving towards a complex decision boundary is doing more harm than good for both models. The RBF or Gaussian kernel performed the best for both models. Using the Gaussian kernel, hyperparameter tuning was performed on the penalty term ‘C’ and kernel coefficient ‘gamma’ via cross-validation. The gamma coefficient can be understood as a cut-off parameter for the Gaussian sphere. If we increase the value for gamma, we increase the influence or reach of the training samples, which leads to a tighter and bumpier decision boundary. The penalty term C is used to balance the margin-error tradeoff of the decision boundary. Large values of C correspond to large error penalties, whereas we are less strict about misclassification errors if we choose smaller values for C. The combination of C and gamma that yielded the best F1 score coincidentally are 10 and 0.1 respectively for both datasets. These parameters were used in generating learning curve by training and cross validating on portions of the training set as shown in Figure 15 below.



```
Per Hyperparameter tuning, best parameters are:
{'C': 10.0, 'gamma': 0.1}
```

```
Per Hyperparameter tuning, best parameters are:
{'C': 10.0, 'gamma': 0.1}
```

Figure 14: Model Complexity Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

For the both data sets, we observe that the training score decreases while the validation score increases with increase in the training example and both curves somewhat tries to converge. This is an indication of low variance, high bias model as we add more training examples, hence adding more training examples might not help much and we will have to use a more complex model to improve model performance. Furthermore, the model performance is examined for both data sets by building the model using the 70% training data and evaluating on the untouched 30% test set.

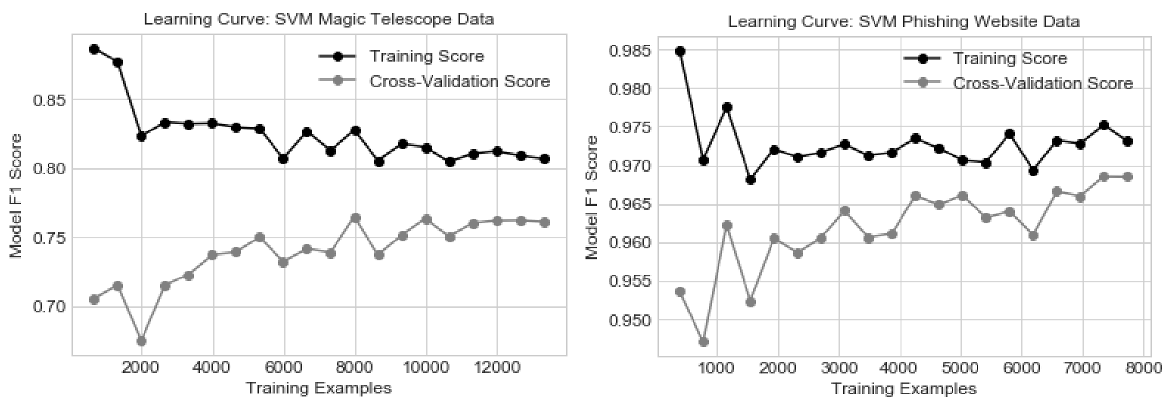
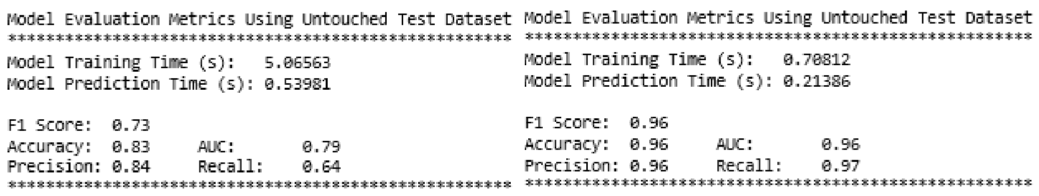


Figure 15: Model Learning Curve for – Left: Magic Telescope Data, Right: Phishing Website Data

Figure 16 above shows the confusion matrix and summary of the model evaluation metrics. Comparing both models, the SVM model performed better on the PW data set compared to the MT data set on all metric – F1 score, accuracy, AUC, precision, and recall. This is potentially due to the fact the PW model uses more discrete attributes and the target attribute is more balanced than the MT data set. In addition, for both data set, the testing time is much lower than the time it takes for training which comes from the fact that the SVM algorithm is an eager learner.



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   5.06563
Model Prediction Time (s): 0.53981

F1 Score:  0.73
Accuracy:  0.83     AUC:       0.79
Precision: 0.84     Recall:    0.64
*****************************************************
```

```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.70812
Model Prediction Time (s): 0.21386

F1 Score:  0.96
Accuracy:  0.96     AUC:       0.96
Precision: 0.96     Recall:    0.97
*****************************************************
```

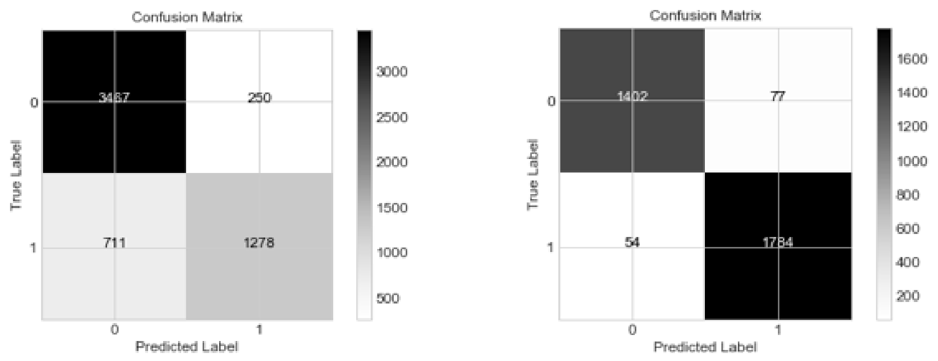
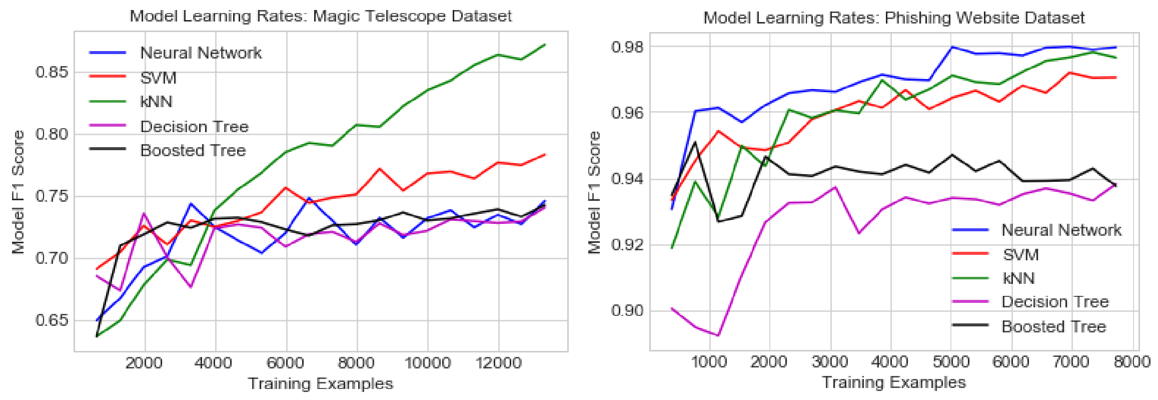
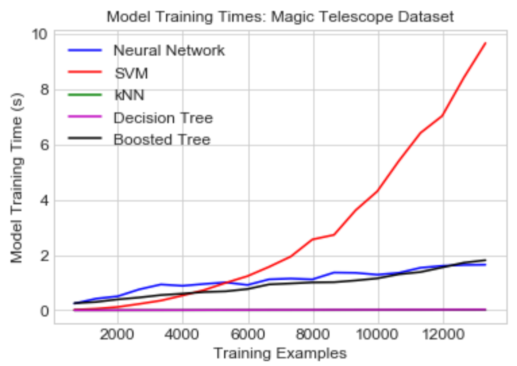


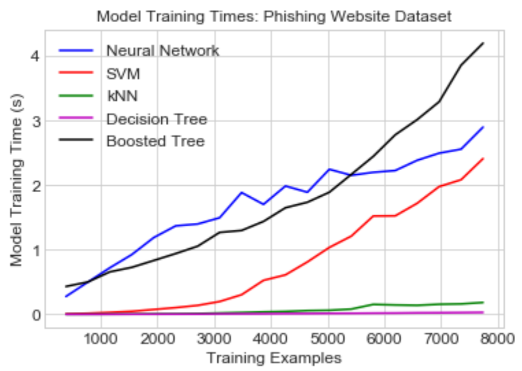
Figure 16: Model Evaluation Metric for – Left: Magic Telescope Data, Right: Phishing Website Data

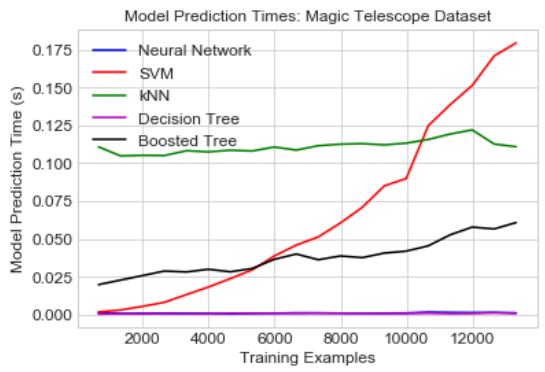
Conclusion

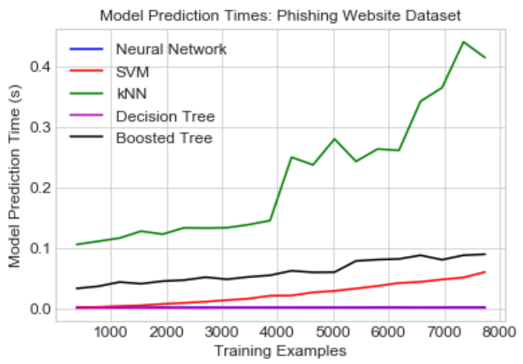
The plots below summarize the observations from the analysis.











- The Phishing Website data set showed better model performance than the Magic Telescope data set. This is potentially due to the fact the PW model uses more discrete attributes and the target attribute is more balanced than the MT data set.

- For the MT data set, there is no significant difference in the performance of the classifiers at low training sample rate but as the sample size increases greater than 4000, kNN and SVM starts to separate from the rest of the classifiers with kNN showing the best performance. For the PW data set, the NN, kNN, and SVM models performed the best with the NN slightly edging the rest. The tree classifiers shows the poorest performance.

- For the MT data set, the SVM model shows exponential increase in training time with increase in training examples while we see more of a linear increase for NN and boosting. The decision tree and kNN show the lowest training time. Similarly, for the PW data set, the SVM and Boosting models shows exponential increase in training time with increase in training examples. The NN classifier show more of a linear increase, while the decision tree and kNN classifiers show the lowest training time

- For the MT data set, testing time increases exponentially for the SVM data as data size increases while the kNN model shows a consistently high testing time. For the PW data set, the kNN model shows exponential increase in testing time as data size increases while the decision tree and NN classifiers show the lowest testing times.

- Overall when taking into consideration the model accuracy, training and testing times, the kNN model performed the best for the MT dataset while the NN model performed the best for the PW dataset.

References

1. Magic Telescope dataset. OpenML Repository https://www.openml.org/d/1120

2. Phishing Website dataset. OpenML Repository https://www.openml.org/d/4534

3. Git Repository: https://github.com/kylewest520/CS-7641---Machine-Learning