Problem Overview (datasets)

1. Phishing Websites – available at OpenML.org (https://www.openml.org/d/4534)

The phishing websites dataset is a compilation of over 11,000 real websites which are labeled as 'legitimate' or 'illegitimate' (spam, untrustworthy, etc.). The binary classification task is to predict which websites are illegitimate such that a user can be notified of the sites' potentially harmful status. The target variable is relatively balanced (55% illegitimate observations) and is accompanied by 30 descriptive features which give information about the website appearance, behavior, URL, and more.

From a practical standpoint, this is an important problem which ultimately seeks to protect web-user confidential information and protect computers from viruses. I am motivated to explore this problem with classification algorithms for two reasons. First, this data was actively and manually collected by researchers. They chose to define features which were known to provide good predictive power in classifying websites. Additionally, they added features at their own discretion which they felt may be useful in the classification task. This is different than many other datasets which tend to be passive data capture exercises. This gives me high expectations that the features will have predictive power in our different applications. I would expect that observed differences between the five classification algorithms will be more directly affected by algorithms' structure and physical relationships in the data than due to random noise. If true, this gives a unique opportunity to study similarities and differences between the algorithms. Secondly, this is an interesting dataset because all 30 features are categorical. Eight features have three levels {-1, 0, 1} and the other 22 features are all binary {0,1}. It will be interesting to study the effects of training time and overfitting with this data since the hypothesis space is made up of hypothesis which are formed by combining a high number of Boolean decisions on these features.

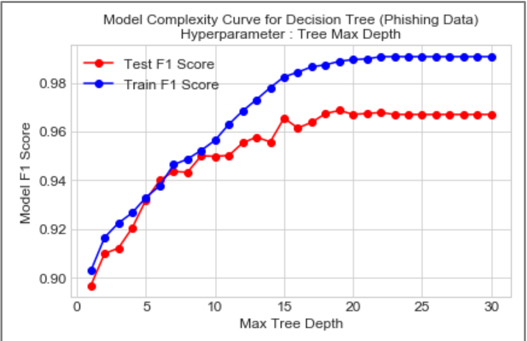
2. Bank Marketing – available at OpenML.org (https://www.openml.org/d/1461)

The bank marketing dataset contains 20 features which describe customer and call attributes for a bank that was cold-calling customers to sell financial services. The dataset has over 45,000 observations. Each call is marked with a 'yes' or 'no' target which indicates if the customer purchased the service. This is another binary classification task, however this one is highly unbalanced. Only 11% of the observations result in the customer purchasing the bank service.

This classification task has high practical relevance for many different business applications which involve customer segmentation and targeting (sales, marketing, and customer relations to name a few). A successful model can prioritize how a business allocates resources for attracting customers. From a machine learning standpoint this dataset is interesting because of the unbalanced nature of the target variable. To compensate for the unbalanced nature, we will need to use different model evaluation metrics beyond accuracy and possibly different hyperparameters as well. Although it is unbalanced, this dataset has over 5000 'yes'-labeled examples. The larger sample size in this dataset should provide interesting insights about the rate at which the classification models learn. Finally, this dataset is interesting because the algorithms are attempting to model human behavior. Humans are prone to being subject to their emotions. I'm expecting this to magnify in the dataset as noise. For example, the bank may have called people 'on a bad day' and received a different response than if they had simply called at a different time. This noise will challenge the classification models' ability to generalize to a test dataset.

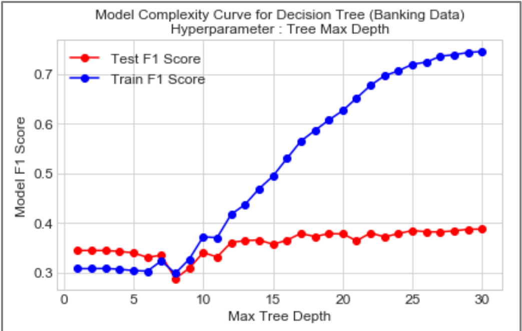
Decision Tree Classification

Phishing Dataset



```
Per Hyperparameter tuning, best parameters are:
{'max_depth': 9, 'min_samples_leaf': 44}
```

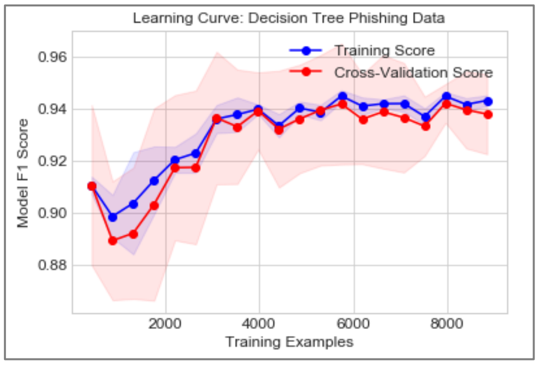
Bank Marketing Dataset

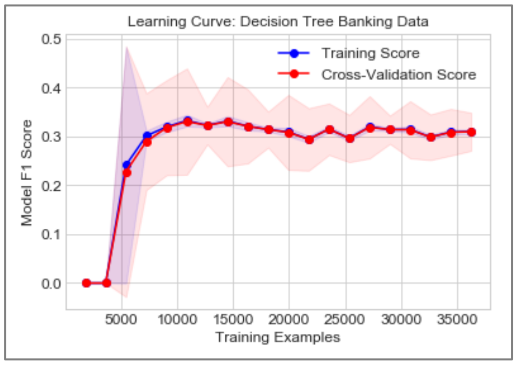


```
Per Hyperparameter tuning, best parameters are:
{'max_depth': 1, 'min_samples_leaf': 181}
```

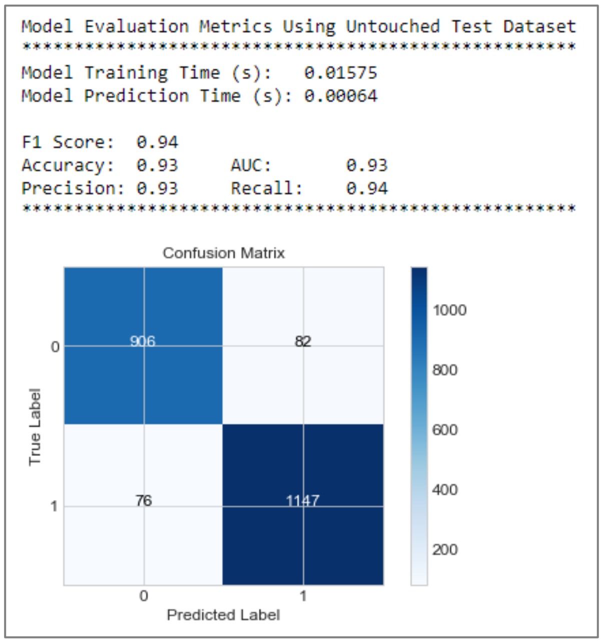
I chose to build my decision tree classifier using information gain, based on node entropy, to determine the best feature splits by using the ID3 algorithm. I used two different pre-pruning techniques (these were both hyperparameters): max_tree_depth, and min_samples_leaf. The model complexity plots show the training and test F1 scores for a range of max_depth values. The phishing data shows that the model generalizes well to the test dataset for values of max_tree_depth ≤ 9. The same can be said of the bank data for max_tree_depth ≤ 10. Above these levels the training dataset tends to overfit the test data in both cases. This is evident by the 'leveling-off' of the test data F1 score while the training F1 score continues to rise for higher levels of max_tree_depth. The overfitting occurs at higher depth levels because the classifier is fitting noise in the underlying training dataset.

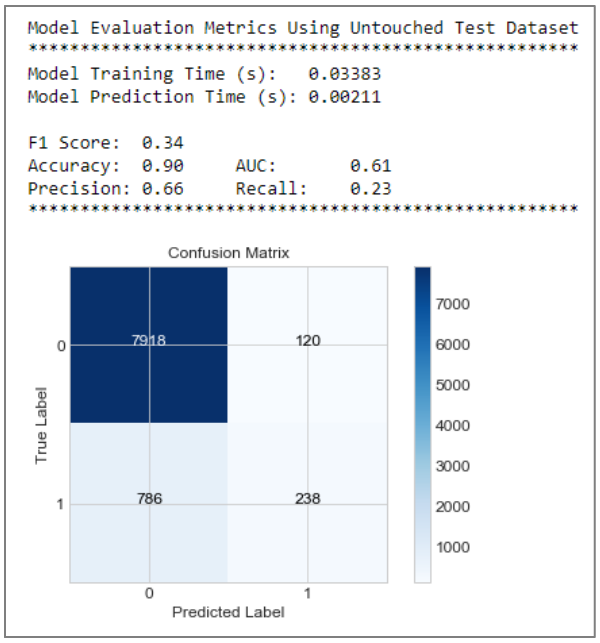
I determined the best combination of max_depth and min_samples_leaf for each dataset using grid search with 10-fold cross validation. The model complexity plots show the best combination for each dataset. It's interesting that the max_depth chosen for the banking data is only one split. This is an example of the inductive bias of the ID3 algorithm of preferring simpler (smaller) trees. The bank classifier has similar performance for max_depth between [1, 6], so the hyperparameter selected is the simplest of the cases.





After identifying the best hyperparameter combination for each dataset, I trained a decision tree classifier with those settings on 80% of the available data. I set the other 20% aside to use later as a test set. With the 80% training data, I generated classifier learning curves by training and cross-validating (again, 10-folds) on portions of the training data. The portion size went from 5% to 100% of the available training data. The phishing classifier generalizes well starting around 2000 samples, while the training and cross-validation F1 scores approach their asymptotes around 3000 samples. The bank classifier has an F1 score of 0 until around 5000 training examples. In the case of the tree only making one split, on the unbalanced data, it is likely that the selected class in each leaf (based on the mode class) is going to be the negative class until more training examples are added. We see this behavior reflected when we get above 10000 training examples where the F1 score jumps up to around 0.30. This confirms one of my initial hypotheses that the classifiers for this unbalanced dataset would rely heavily on that fact that we have 45000 examples to learn from. For both datasets we see that the cross-validation score is similar to the training score, suggesting that these trees have low variance.


```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.01575
Model Prediction Time (s): 0.00064

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

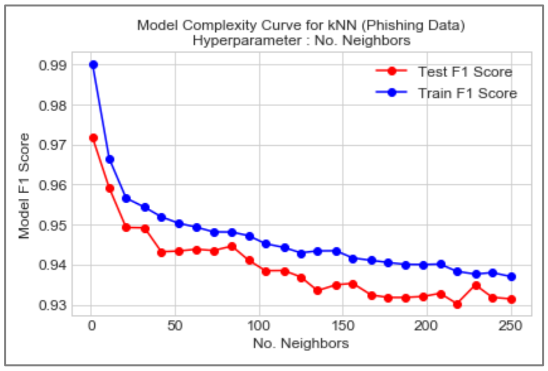

```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.03383
Model Prediction Time (s): 0.00211

F1 Score:  0.34
Accuracy:  0.90     AUC:       0.61
Precision: 0.66     Recall:    0.23
*****************************************************
```

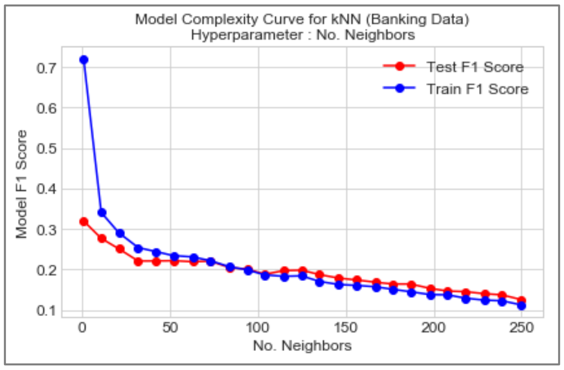
Lastly, I used the final models trained on all of the available training data and tested them on the previously untouched 20% subset of test data. For both datasets the model training times are a few orders of magnitude higher than the prediction times. This makes sense because the ID3 algorithm is a greedy learner, meaning it searches across all possible features before deciding which feature to split a node on. The decision tree model performed better on the phishing data according to all meaningful metrics. The phishing model was more complex (the tree had more splits), but it had better performance metrics in F1 score, AUC, precision, and recall. Accuracy was the only metric where the bank data was higher. However, this metric is meaningless for the bank data since a high accuracy can be obtained by always predicting the {0} class since the target variable is highly unbalanced.

k Nearest Neighbors Classification

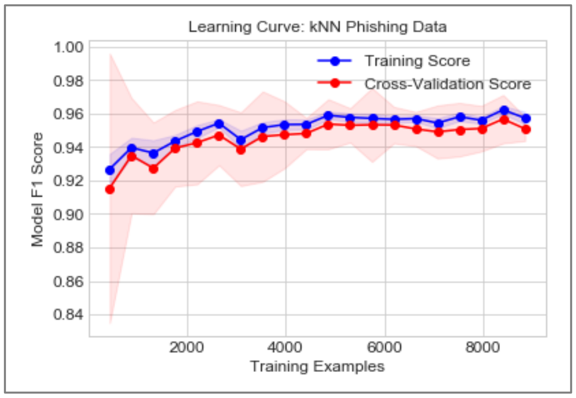
Phishing Dataset

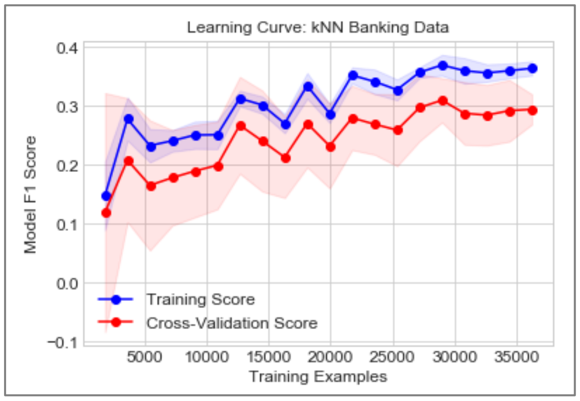
Bank Marketing Dataset



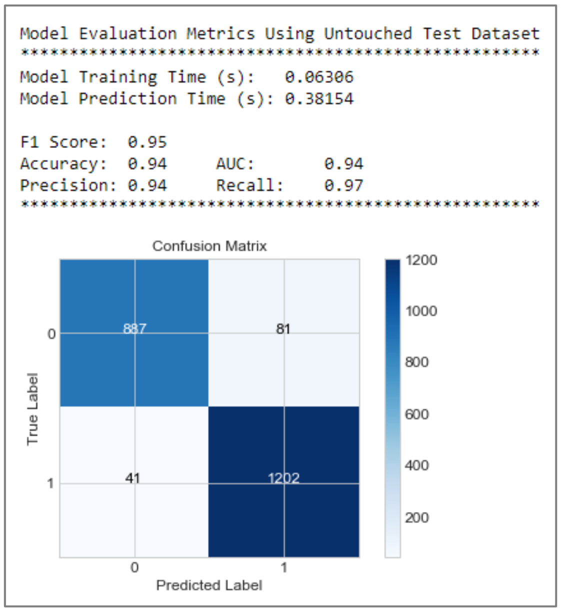


I designed my kNN classifier based on unweighted Euclidian distance. I first iterated over the hyperparameter k (number of neighbors) to build the model complexity curves. For the phishing data we see that as k increases, the F1 score tends to approach a limit around 0.93. For banking, as k increases the F1 score continues to decrease towards 0. This is due to the fact that as the model considers more neighbors, there is a high chance that the 'next nearest' neighbor is among the {0} class since it makes up 89% of the available training examples. For this problem, smaller k values provide better classifiers.

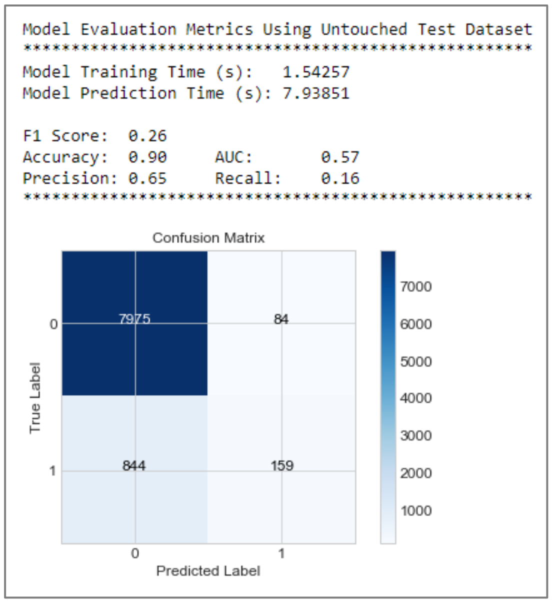




I trained the final classifiers on k = 20, 10 for the phishing and bank datasets, respectively. I chose these values based on where I felt the elbows were on the model complexity diagrams. The kNN classifier learns quickly for the phishing dataset. There is only marginal increase in F1 score after 2000 training examples. Conversely, the banking dataset F1 score continues to increase as more samples are training on. This suggests that adding even more training examples could improve the classifier. The banking classifier has 5-10% variance between the training and cross-validation scores for all numbers of training examples. I suspect this is from higher noise levels in the underlying data.



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.06306
Model Prediction Time (s): 0.38154

F1 Score:  0.95
Accuracy:  0.94     AUC:       0.94
Precision: 0.94     Recall:    0.97
*****************************************************
```

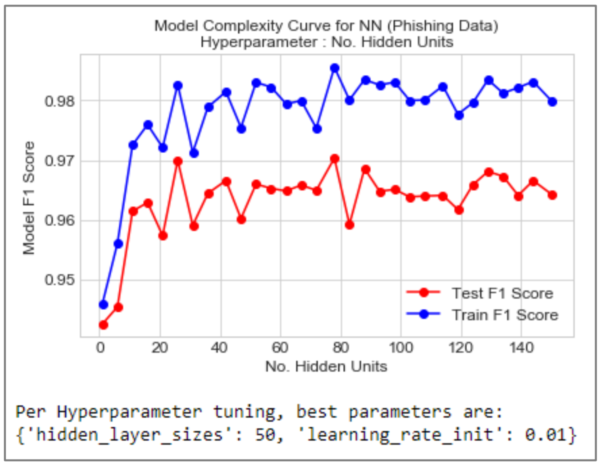


```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   1.54257
Model Prediction Time (s): 7.93851

F1 Score:  0.26
Accuracy:  0.90     AUC:       0.57
Precision: 0.65     Recall:    0.16
*****************************************************
```

There is a clear contrast (for reasons specified above) in the kNN models between the two datasets when comparing the final classifier evaluated against the held-out test set. kNN performs exceptionally well in all metrics for phishing and doesn't perform very well for banking. For both cases there is a clear difference between training and prediction times. kNN trains quicker than it predicts since it is a lazy learner. The banking kNN model takes longer to train and predict due to a higher number of observations to query when searching for the nearest neighbors.

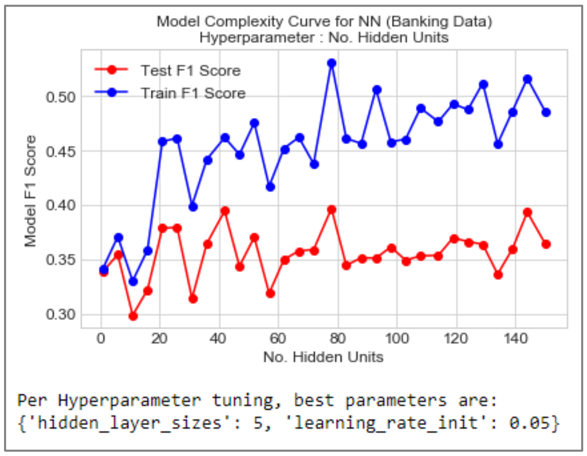
<u>Neural Network Classification</u>

<u>Phishing Dataset</u>



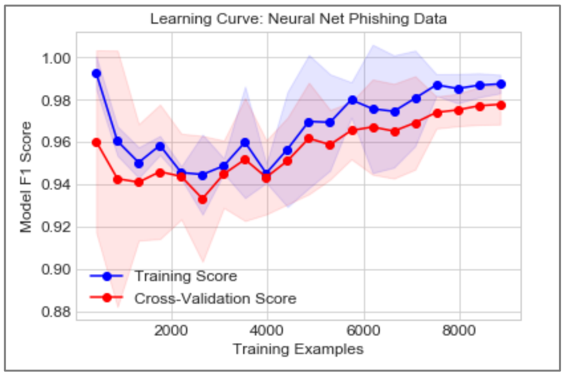
```
Per Hyperparameter tuning, best parameters are:
{'hidden_layer_sizes': 50, 'learning_rate_init': 0.01}
```

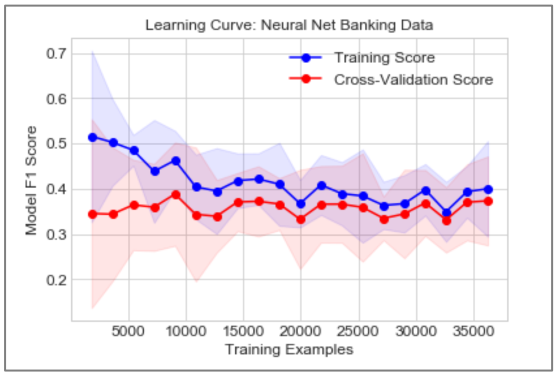
<u>Bank Marketing Dataset</u>



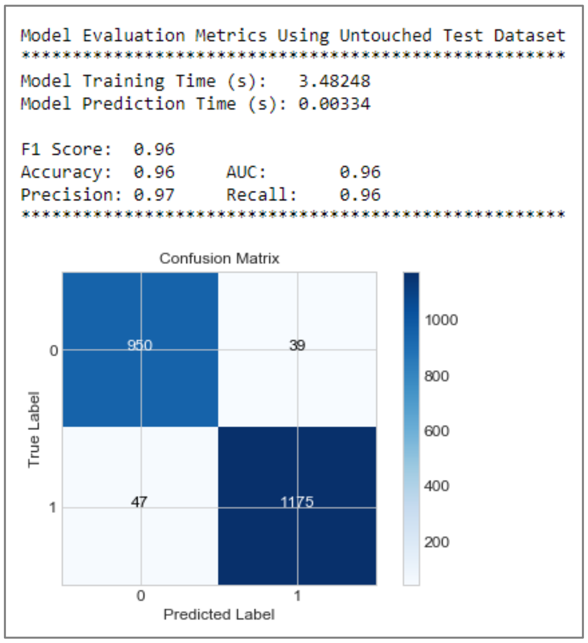
```
Per Hyperparameter tuning, best parameters are:
{'hidden_layer_sizes': 5, 'learning_rate_init': 0.05}
```

I built a forward feed neural network model using a multi-layer perceptron with a sigmoid activation function. The network structure I designed was one with a single hidden layer where my hyperparameters were number of hidden units and learning rate. In both datasets the model F1 scores are lowest for < 3 hidden nodes. We see this behavior because the hypothesis space is limited when using only a few hidden nodes. We see that has the number of hidden nodes increases on the banking dataset, the model variance increases suggesting that the network is starting to fit the underlying noise in the data.

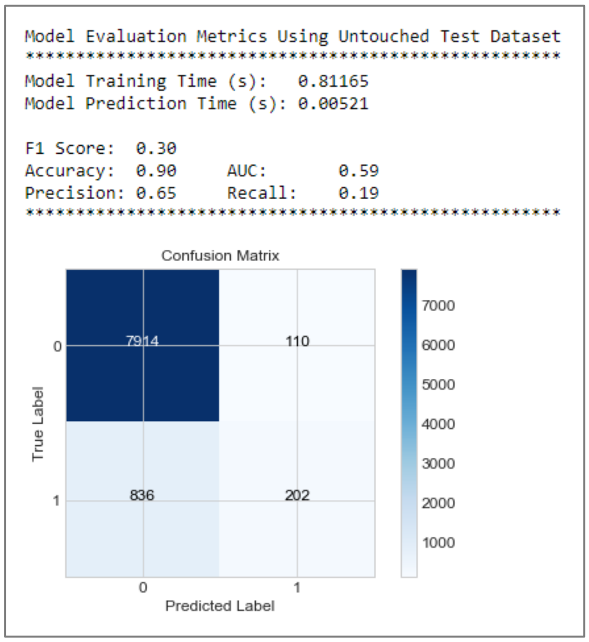




The phishing data learning curve initially decreases between 0 and 2000 training examples, before continuing to rise again. This may be due to the gradient descent algorithm getting stuck at a local minimum with this limited training set. Overall the phishing learning curve doesn't appear to be approaching an asymptote as more training examples are added. The banking neural network classifier performs equally well in the cross-validation set regardless of how much data is added. This is in stark contrast to the other models where we needed at least 5000 samples to see convergence. We do observe that the variance tends to decrease as more samples are added; another example of how the underlying unbalanced target variable effects the model's ability to generalize well to new observations.

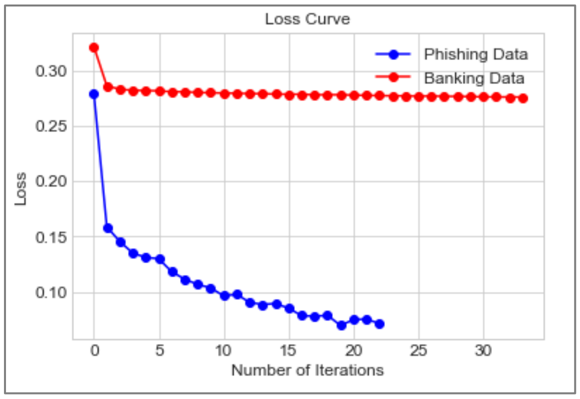


```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   3.48248
Model Prediction Time (s): 0.00334

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



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.81165
Model Prediction Time (s): 0.00521

F1 Score:  0.30
Accuracy:  0.90     AUC:       0.59
Precision: 0.65     Recall:    0.19
*****************************************************
```

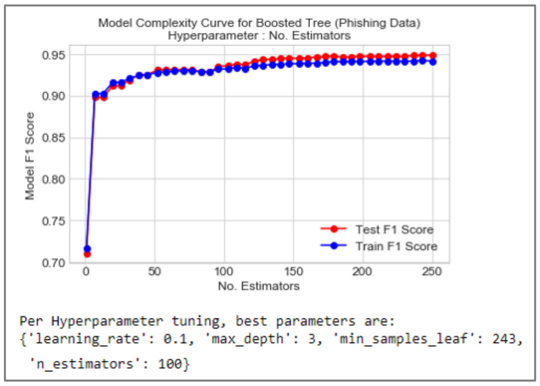
Both neural network models take a few orders of magnitude longer to train than to predict. We also see that the phishing network takes four times longer to train than the banking network even though the phishing training set only has 25% as many examples. This is due the higher number of hidden nodes in the phishing network (50 vs. 5). The more complicated network has more weights to compute in the backpropagation routine.



As a final comparison for the neural network models, I plotted the loss curve for each dataset trained on the full training set with a constant learning rate. The loss represents the log likelihood (error function) of the neural networks. We observe that the loss hardly changes after a single iteration for the banking data, suggesting that the error doesn't improve much with each successive iteration. This is due to the fact that the model is almost always predicting the negative class. For the phishing dataset we can see that the loss improves as the iteration count goes up, before the model stops iterating (converges) after 22 iterations.

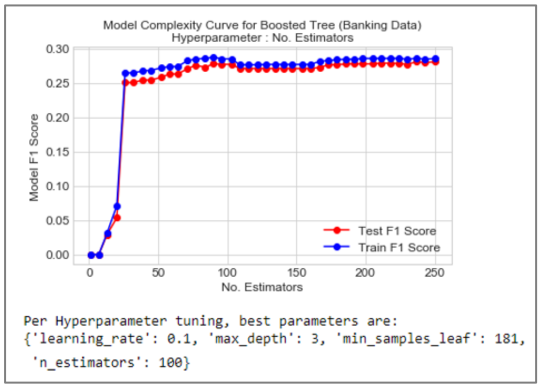
Boosted Decision Tree Classification

Phishing Dataset

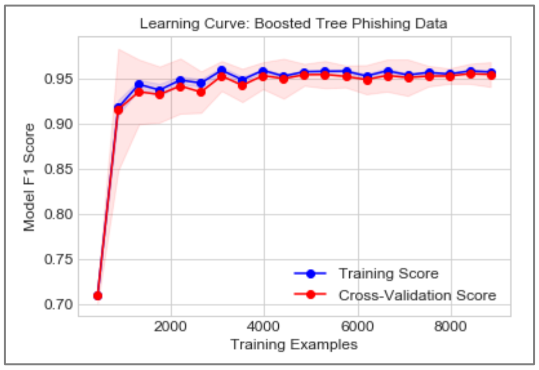


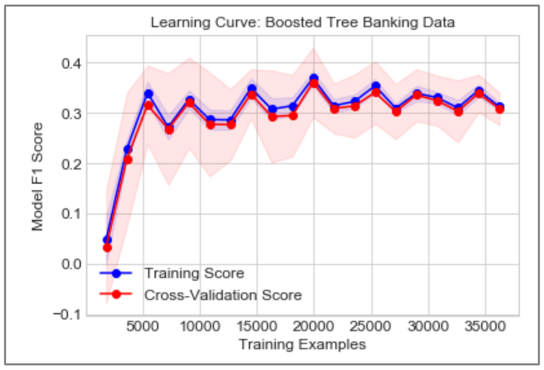
```
Per Hyperparameter tuning, best parameters are:
{'learning_rate': 0.1, 'max_depth': 3, 'min_samples_leaf': 243,
 'n_estimators': 100}
```

Bank Marketing Dataset

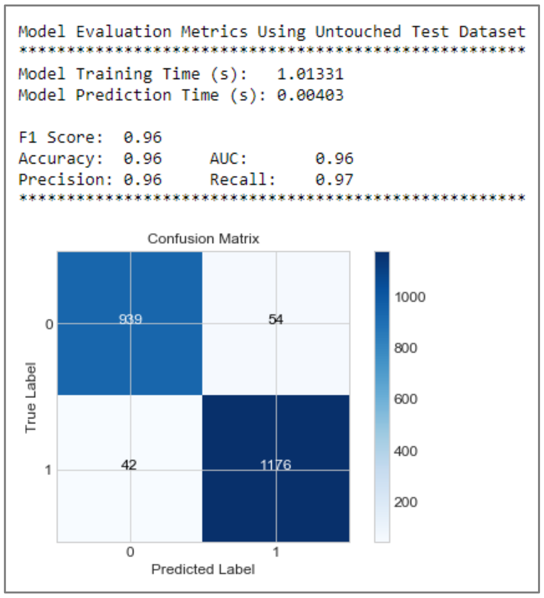


```
Per Hyperparameter tuning, best parameters are:
{'learning_rate': 0.1, 'max_depth': 3, 'min_samples_leaf': 181,
 'n_estimators': 100}
```

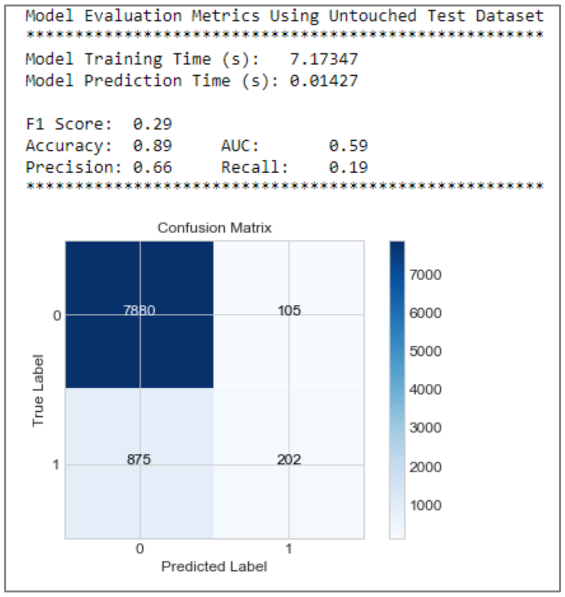




I created a boosted tree classifier with more aggressive pre-pruning techniques compared to that of the decision tree model. This creates more 'weak learners', but the model becomes powerful when combined with other weak learners. The key advantage of boosting methods is that they are more robust to overfitting that other methods. This property is displayed in the model complexity and learning curve plots for both datasets. The training F1 scores are largely equivalent to the cross-validation scores in all plots. One key difference we observe in the phishing dataset is that it learns more quickly than the other models. In the case of a boosted tree classifier we see that the model performs well for under 1000 training examples.



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   1.01331
Model Prediction Time (s): 0.00403

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

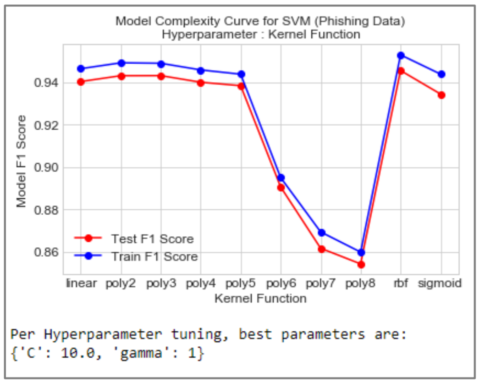


```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   7.17347
Model Prediction Time (s): 0.01427

F1 Score:  0.29
Accuracy:  0.89     AUC:       0.59
Precision: 0.66     Recall:    0.19
*****************************************************
```

The boosted tree classifiers take longer to train than to predict. This attribute comes from the fact that the algorithm is building multiple (100 in this case) trees and then combining them into a single classifier. 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 they have less overfitting.

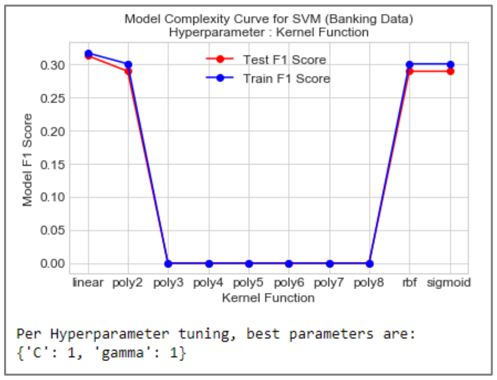
Support Vector Machine Classification

Phishing Dataset



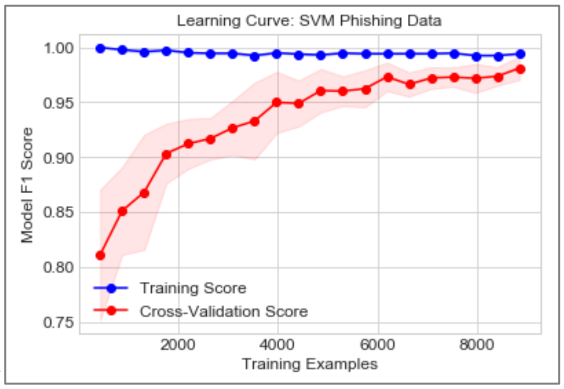
```
Per Hyperparameter tuning, best parameters are:
{'C': 10.0, 'gamma': 1}
```

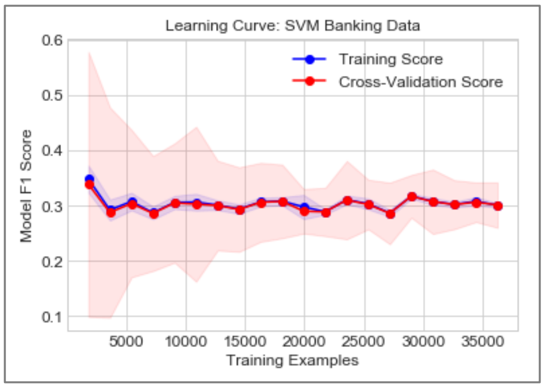
Bank Marketing Dataset



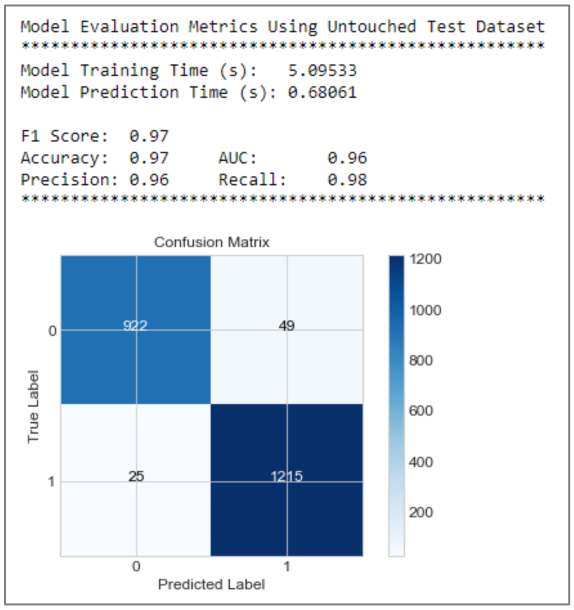
```
Per Hyperparameter tuning, best parameters are:
{'C': 1, 'gamma': 1}
```

I chose to work with different kernel functions as the primary hyperparameter on the SVM classifier. I also performed grid search using the penalty term 'C' and kernel coefficient 'gamma'. On the phishing data we see the model F1 score decreases as we introduce higher order polynomial kernels. In this case, the more complex decision boundary is doing more harm than good. I suspect that this is due to the fact that the attributes in this dataset were determined by the researchers and chosen specifically because they suspected the attributes would have predictive power. For the banking data we observe that for polynomials above second order, the F1 score is zero. For these kernels the classifier always predicts the {0} class. This behavior is a combination of the unbalanced data and the curse of dimensionality. After the one-hot encoding, the bank data has 64 features. Even with 45000 observations the data is still sparse in the 64-degree space.

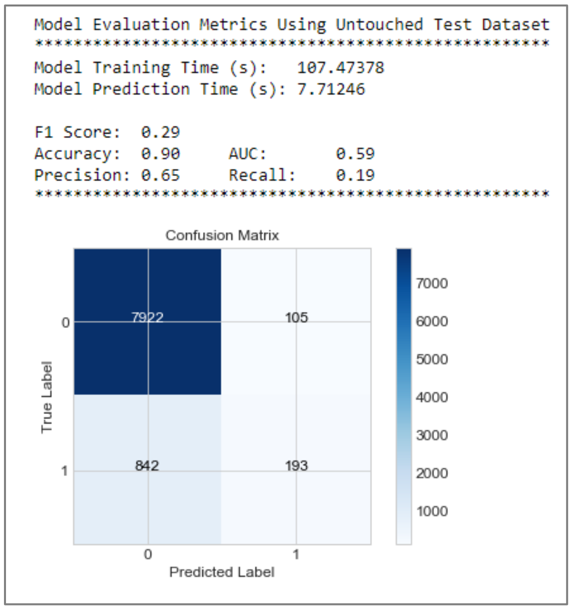




I generated the SVM learning curves using the radial basis function kernel for both datasets. The phishing SVM model shows a high degree of overfitting for smaller training sets. As we add more data the cross-validation score does tend to increase towards the training score. This is the first model we have seen with this amount of variance for the phishing data. This is caused by the fact that grid search chose the best value of the 'C' penalty term to be 10. C describes the amount of regularization that occurs when training the kernel. For higher values of C, we expect low bias at the expense of high variance. As more samples are added, it tends to compensate for this high variance since the space of the data becomes more densely populated. For the banking dataset the SVM classifier performs nearly as well 5% of the available training data as with 100%. We do see that the variation of F1 score across the cross-validation 10-folds decreases as more examples are added (see that the shaded region, which represents the $\pm 2\sigma$ range, tightens).



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   5.09533
Model Prediction Time (s): 0.68061

F1 Score:  0.97
Accuracy:  0.97     AUC:       0.96
Precision: 0.96     Recall:    0.98
*****************************************************
```



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   107.47378
Model Prediction Time (s): 7.71246

F1 Score:  0.29
Accuracy:  0.90     AUC:       0.59
Precision: 0.65     Recall:    0.19
*****************************************************
```

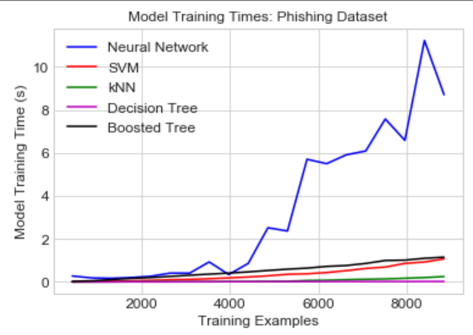
The SVM models take longer to train that to predict, as it is an eager learner. We see that the training time for the banking SVM is the highest among all of the models in this report as the RBF kernel is searching overall all of the training examples upfront to draw the nonlinear decision boundary.

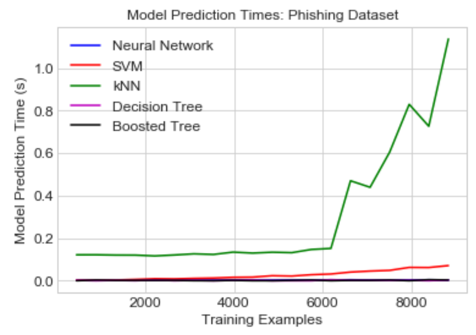
<u>Conclusion</u>

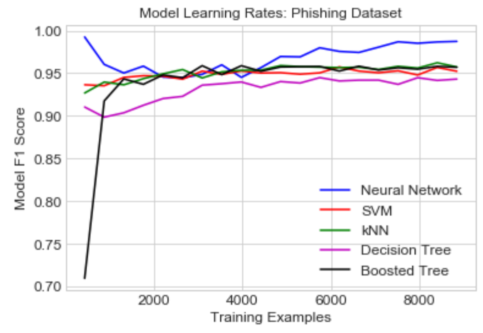
The first two plots below show the model training and prediction times for a variety of training examples across the five classification algorithms. We observe the longest phishing training time for the neural network, which had a high number of hidden nodes for which to compute weights. On the banking data we see the highest training time in the SVM model using the radial basis function kernel. In both datasets we see the highest prediction time for kNN, especially for the banking dataset (more examples to search over). This is expected behavior for this lazy learning algorithm. The final plots show the learning rates for the classifiers after hyperparameter tuning was performed. On the phishing data, all the model F1 scores stabilize after around 3000 examples, except for the neural network which continues to improve. We see the opposite behavior for the neural network on the banking dataset. The high number of samples on the banking dataset was crucial to see the learning behavior past a few thousand samples for all the models.

These two binary classification problems present vastly different predictive capabilities. The attributes in the phishing data have high predictive probability. Even the worst classifier (decision tree) has an F1 score close to 0.95. On the other hand, the best classifier (neural network) has an F1 scores of 0.40 on the banking data. The stark contrast in predictive power is a combination of the difficulty of the task, the given features (and how these features were defined/collected), and the unbalanced vs. balanced properties.

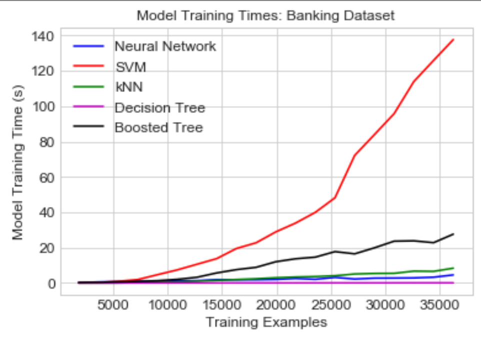
<u>Phishing Dataset</u>

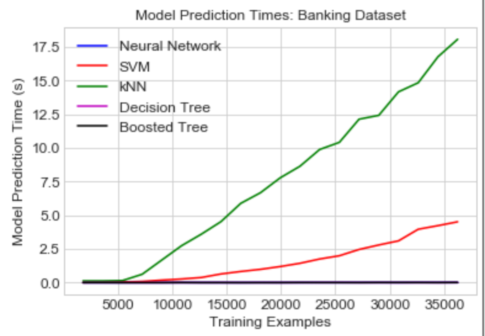


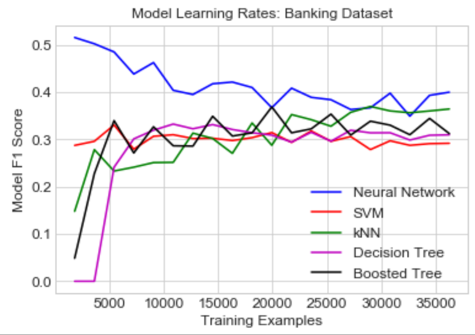




<u>Bank Marketing Dataset</u>


Model Training Times: Banking Dataset
Model Training Time (s)
Training Examples
Neural Network
SVM
kNN
Decision Tree
Boosted Tree


Model Prediction Times: Banking Dataset
Model Prediction Time (s)
Training Examples
Neural Network
SVM
kNN
Decision Tree
Boosted Tree


Model Learning Rates: Banking Dataset
Model F1 Score
Training Examples
Neural Network
SVM
kNN
Decision Tree
Boosted Tree