Yi Shuen Lim
ylim68 | 903647878
CS 7641: Machine Learning
Assignment 1

Supervised Learning: NBA Shots and Fraud Detection

The Datasets

Dataset 1: NBA Shot Logs

This first dataset chosen is data on over 122,502 shots taken during the 2014-2015 NBA season. The goal is to see if supervised learning techniques are able to classify whether a shot is made or missed using features such as player, distance, time of game and period, distance to the closest defender, number of dribbles, etc. Features that pertained to the outcome of the game were removed to prevent the use of possibly confounding variables.

This dataset is also fairly balanced, with 54.3% of the shots recorded being missed shots, and the other 45.7% being made shots. I'm interested to see how and whether these models are able to predict something typically regarded as skill-based and almost random. I'll be optimizing for overall accuracy in classification.

Dataset 2: Credit Card Fraud Detection

This dataset is on 284,807 European credit card transactions that took place in September 2013. Aside from the transaction amount, the 28 other features are labeled *V1*, *V2*, *V3*, etc. as they represent the 28 Principal Components of the confidential data. With the volume of transactions increasing over the years, detecting fraudulent transactions has also become more and more relevant.

This dataset is interesting in its imbalance, with only 492 of the over 284,000 transactions being fraudulent. Though the features have been anonymized, as PCA optimizes for variance in the data, I'm interested to see whether these models will be able to detect fraud. To account for this imbalance, I will be optimizing for **recall** of the fraud class.

Methodology

To explore supervised learning methods, both datasets were randomly split 80% and 20% to form training and testing sets. The same testing sets for both datasets are kept consistent to maintain an apples-to-apples comparison. Across five algorithms, namely Decision Trees, Adaptive Boosting, K-Nearest Neighbors, Support Vector Machines and Neural Networks, relationships between modeling metrics and the following aspects will be explored:

1. Hyperparameter values
2. The effect of the number of training examples used
3. Model fit duration

First, depending on the algorithm, either an incremental implementation of a particular hyperparameter or an instance of scikit-learn's GridSearch will be used on the full training sets to observe the effect of changing hyperparameters for each algorithm. Within GridSearch, a 5-fold cross validation will be used to assure that hyperparameters are not overfit to any subset of the training data.

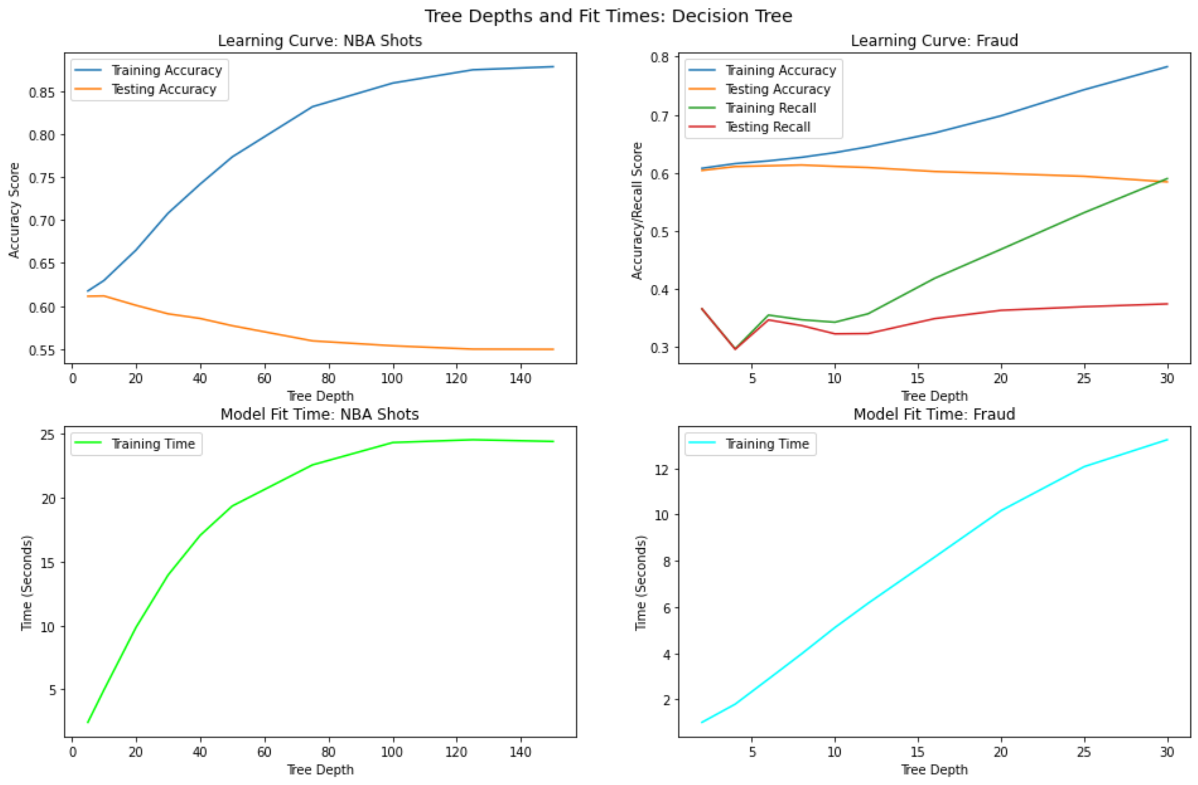
Next, the effect of the number of training examples on the models' metrics will be explored by taking a random sample of the training data, ranging from 5% of the training set to the full 100% of the training set. This will be implemented with scikit-learn's default hyperparameters for each model and again if necessary, depending on the algorithm. Where relevant, the duration of the model fit step will also be recorded.

Decision Tree

For the Decision Tree, the main parameter to tune especially for overfitting and model complexity is max_depth. Prior to that, an initial GridSearch was done to give the model its best chance via other hyperparameters. For this GridSearch, the scoring metrics the search optimized for were **accuracy** for NBA Shots, and **recall** for Fraud. The following were the best parameters tested and obtained from GridSearch for each dataset:

Hyperparameters: Values Tested	**NBA | CV Acc: 61.4%**	**Fraud | CV Recall: 39.5%**
criterion: gini, entropy	entropy	gini
min_samples_split: 5, 10, 15	15	5
max_features: None, sqrt, log2	None	None
max_depth: 10, 20, 30	10	30

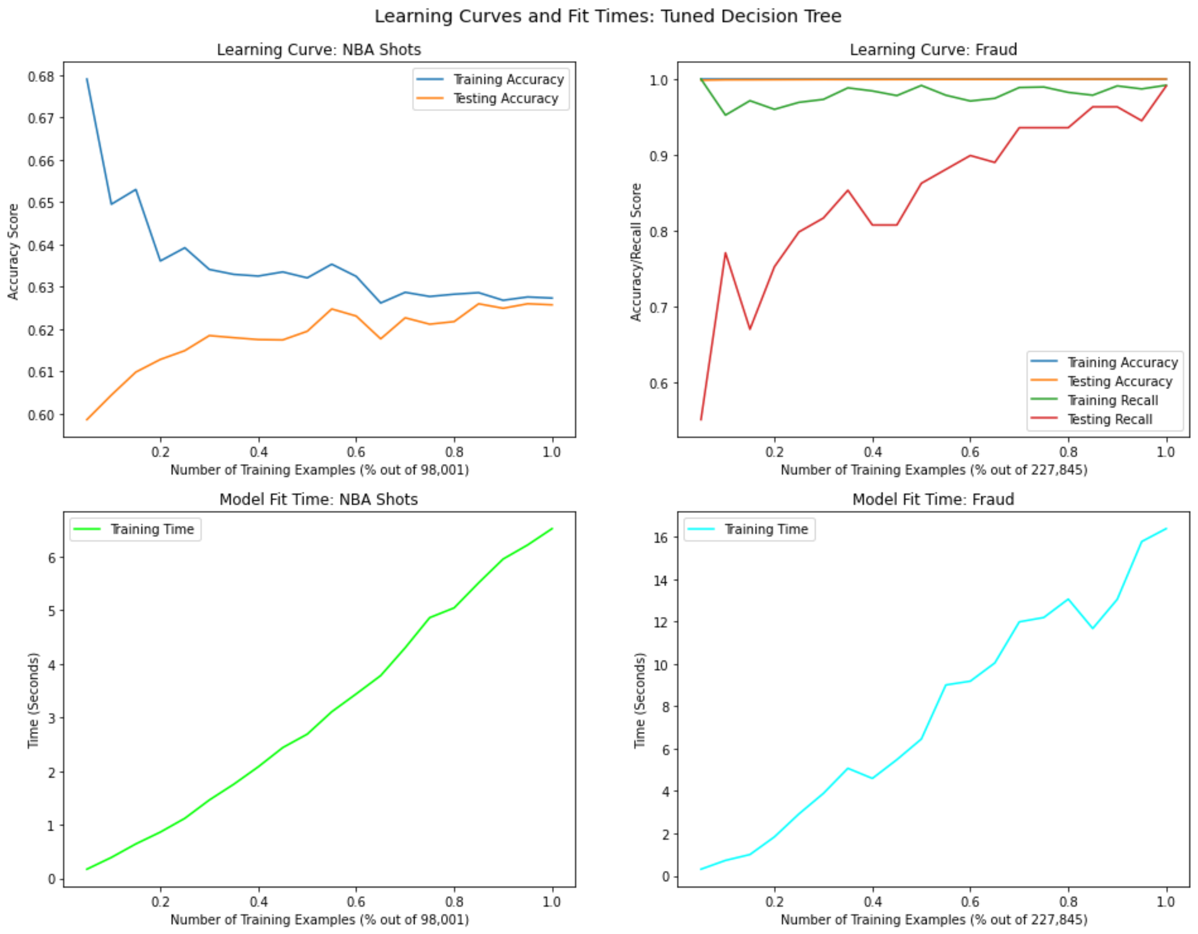
Because Decision Trees are known to have high variance and be prone to overfitting, I wanted to see how different values of max_depth would affect model performance, especially since the NBA dataset yielded that a depth of 10 outperformed the other tested depths of 20 and 30. Since tree depth ultimately increases the number of terminal nodes in the tree, I hypothesize that metrics will improve up to a certain depth, afterwhich training and testing metrics will start to diverge, signalling overfitting.



To determine the ranges of tree depth values shown on the x axes of the preceding graphs, I first ran a vanilla model for each dataset's training set and retrieved the tree depth of a fully built-out tree. This was 144 for the NBA dataset and 24 for the Fraud dataset. To analyze the effect of tree depth, I kept the other hyperparameters at their optimal values from GridSearch. For this tree depth analysis I also tracked model fit times.

As expected, deeper trees lead to higher training metrics, but after a certain depth level (roughly 10 for the NBA dataset and 6 for the Fraud dataset. Based on these results, the additional computational power and time for a deeper tree past these numbers don't yield significant marginal increases in performances, so we can prune our tree back to these depths. Also worth noting, after roughly a depth of 100 for the NBA dataset, the training time plateaued off, so perhaps purity was achieved over enough features from that depth such that not much more additional training time was added with additional depth.

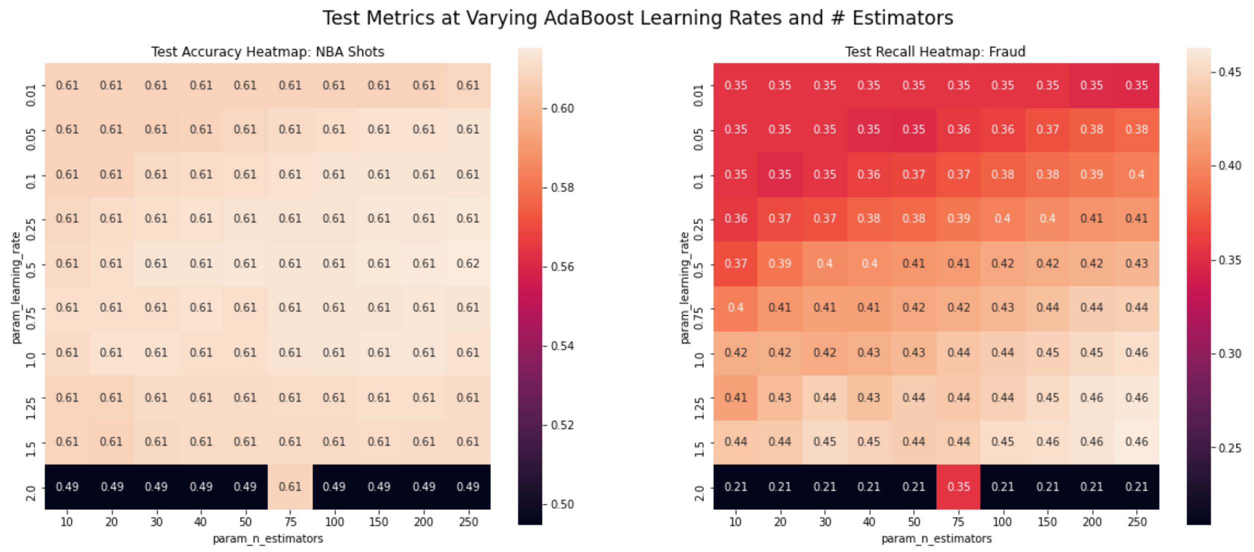
Finally, I wanted to explore the impact of the number of training examples on model metrics and train time. The models used were with the previously-tuned optimal parameters, and I used the process as described in the Methodology section.



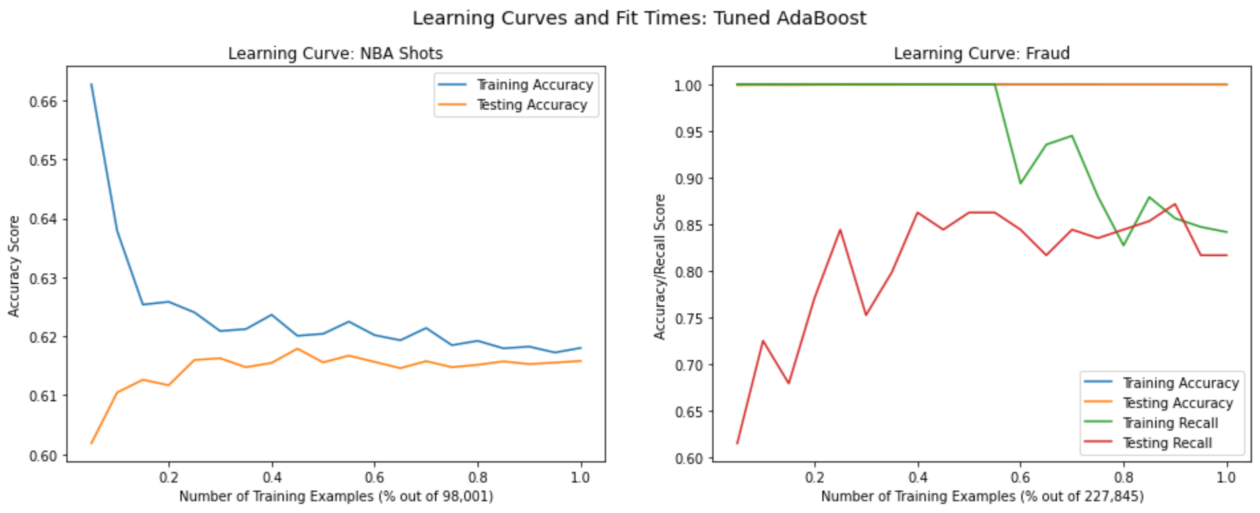
As seen in the graphs above, overall testing performance does improve with more training examples. This is especially so in the Fraud dataset's recall, perhaps due to the significant class imbalance, so more instances of fraud allows the tree to better identify Fraud cases. With regards to the NBA dataset, it seems that after around 60% of the train set being used, the marginal improvement in accuracy plateaus off and the fit time doubles. If computational time and power is a concern, it might be possible to use less data in model training.

AdaBoost Classifier

For this AdaBoost classifier, the default base estimator of a Decision Tree with a depth of 1 will be used. The main two parameters to tune are the number of estimators and the learning rate. As expected, the fit and test times both increase with an increasing number of estimators and a slower learning rate.



Based on the heatmaps above, the NBA dataset has its highest accuracy with a 0.5 learning rate and 250 estimators. The fraud dataset has its highest recall at a 1.5 learning rate and 250 estimators. With the optimal number of estimators and learning rates, we look at the learning curve for these datasets.

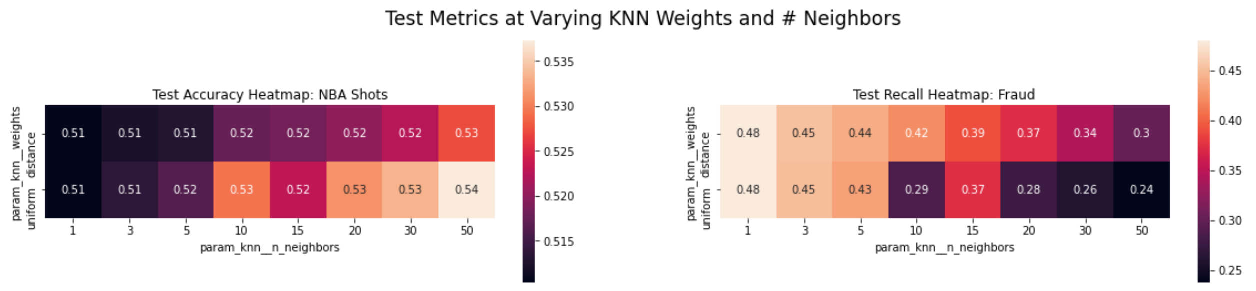


For the NBA shots data, the learning doesn't improve much after using about 50% of the dataset. For the Fraud data though, interestingly we see that past 50% of the dataset being used despite testing recall continues to improve marginally, the training recall starts to go down. This could be in part due to random undersampling affecting the amount learned from the minority class (which only constitutes under 0.2% of the dataset). In future work, I'd control the sampling such that the class balance remains fairly consistent across undersampled datasets.

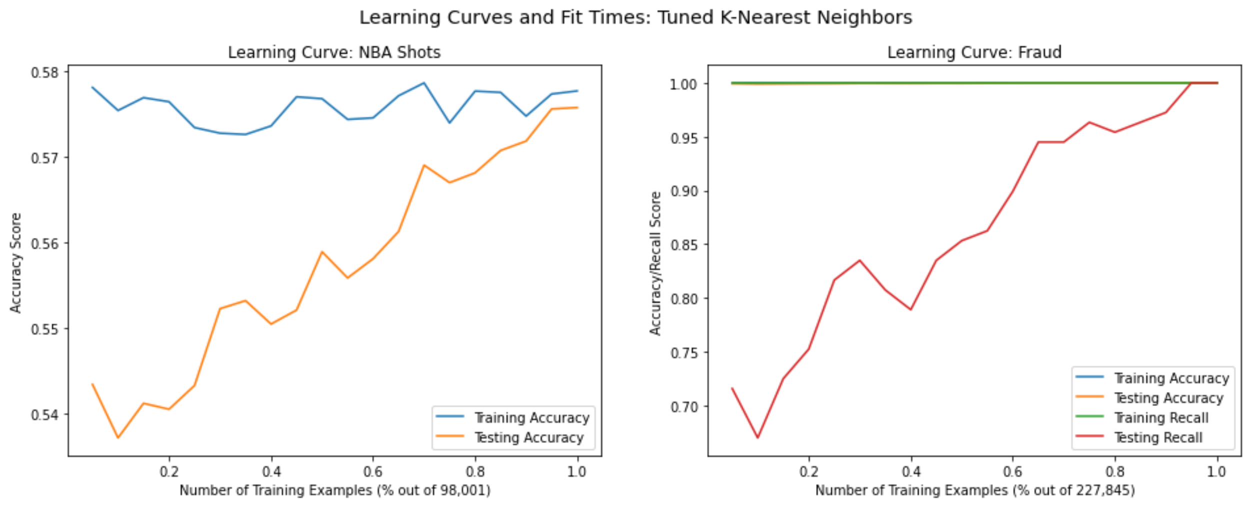
K Nearest Neighbors Classifier

For the KNN classifier, both datasets were first standardized such that this distance-based algorithm accounts for each of their features fairly. The two hyperparameters tested were the number of neighbors and whether similarities were weighted equally or inverse to distance.

Below are the results of the GridSearch. For both datasets, uniform weights tended to outperform inverse-distance weights. Interestingly, the number of neighbors used affected performance in both datasets differently. The NBA shots data performed better with more neighbors while the Fraud dataset performed better with fewer. I believe this might be a point of contrast between a class-balanced and class-imbalanced dataset. The higher number of neighbors (50) benefitted the balanced dataset as it possibly helped to combat the overfitting, while the smaller number of neighbors (1) benefitted the severe class imbalance to help identify the very few instances of fraud in the data.

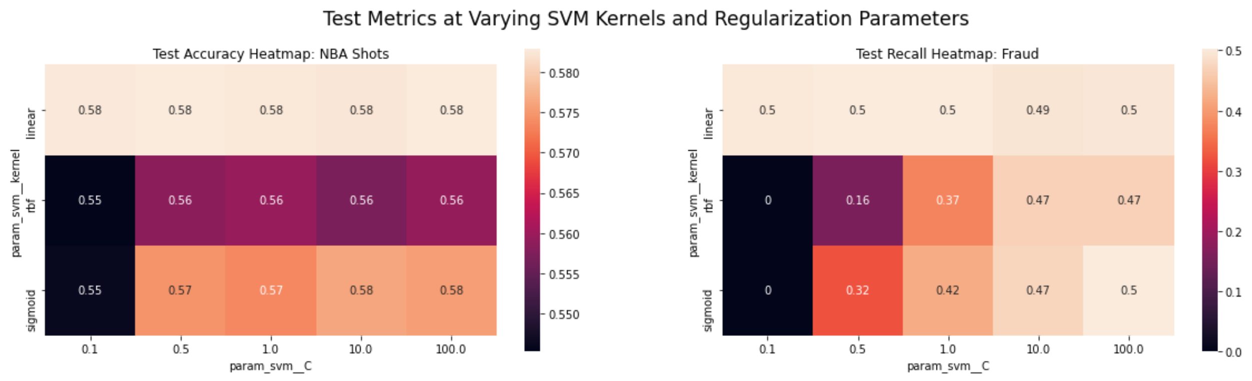


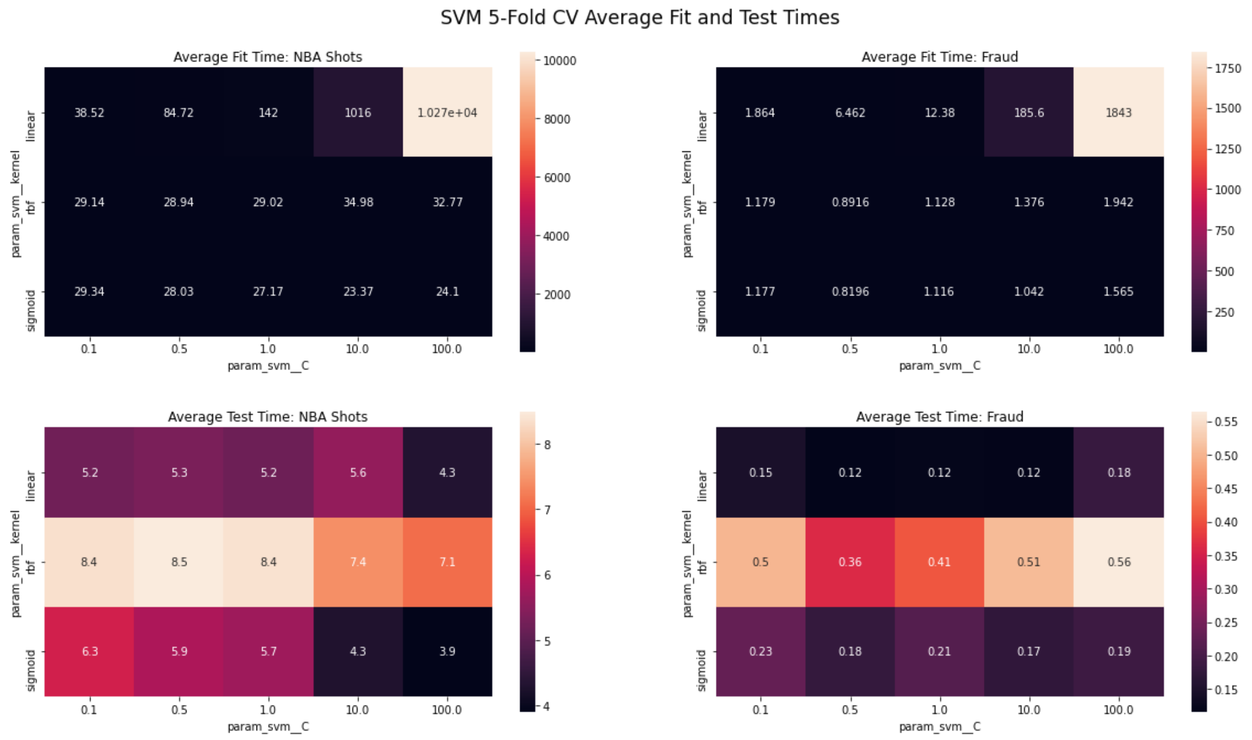
In examining the learning curves for both datasets, using the best performing hyperparameters from the GridSearches, more training examples definitely contributed to the performance of the models. Both datasets, as expected, also saw increased training times with more data used. The model for NBA Shots used



Support Vector Machine

For an SVM classifier, the kernel used as well as the regularization parameter C were tested, and the results are shown below. Overall, the linear kernel took considerably longer to fit.

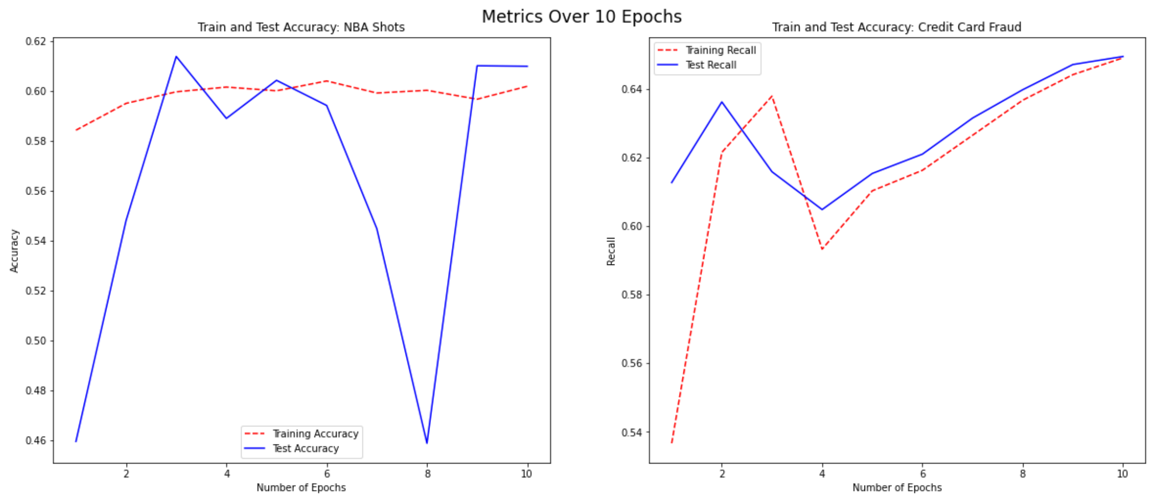




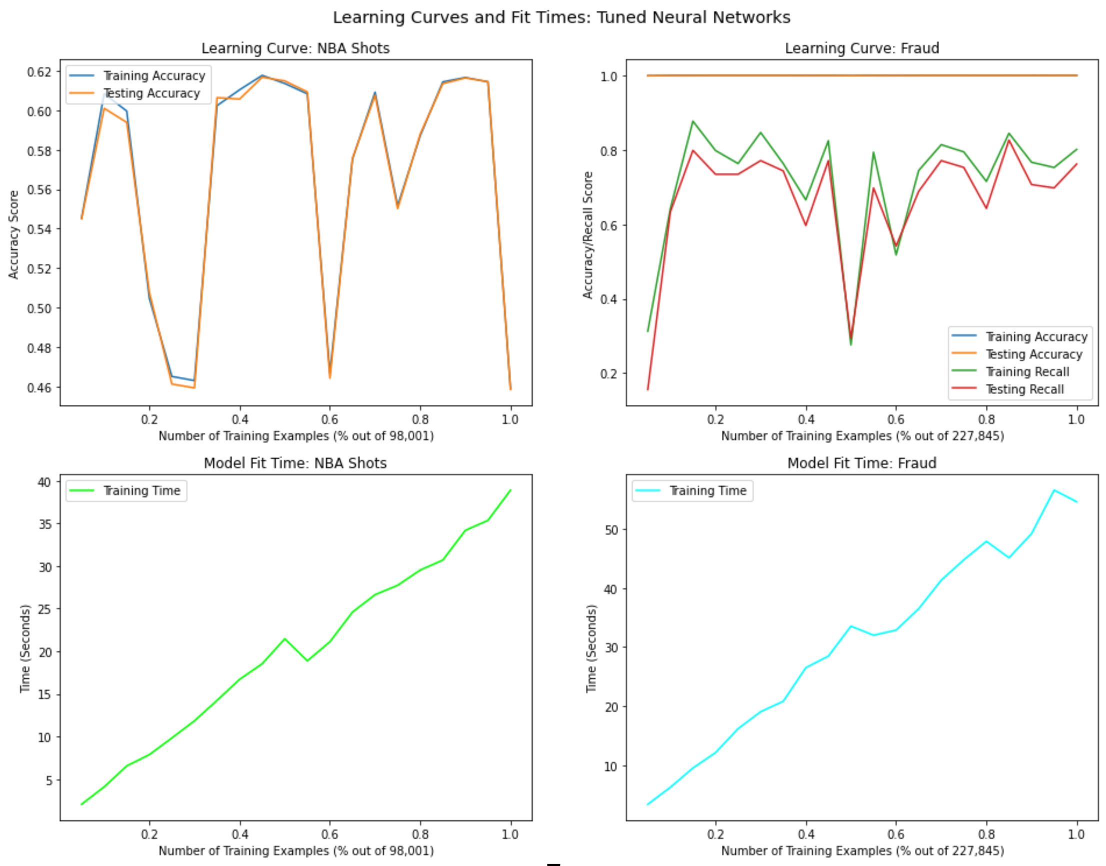
Dense Neural Network

A neural network with two dense layers with a dropout layer in between for regularization will be used to explore model performances. Utilizing the Talos library, values for the number of nodes per hidden Dense layer, proportion for the Dropout layer, activation functions for the hidden layers as well as the optimizer were tuned. Summarized below are the hyperparameters tested and the best performing combination per dataset:

Hyperparameters: Values Tested	**NBA | CV Acc: 61.7%**	**Fraud | CV Recall: 80.2%**
Nodes: 100, 500, 1000	1000	1000
Dropout: 0.1, 0.3, 0.5	0.3	0.1
Activation function: ReLU, TanH	TanH, TanH	TanH, ReLU
Optimizer: Adam, SGD	Adam	Adam



With the best parameters, the following graph shows the scores over the 10 epochs, and as expected, metrics improve overall with more epochs. With more time, letting these models train for additional epochs until metrics begin to plateau would be beneficial. There seems to be some instability in testing accuracy for the NBA shots dataset, which could be helped with more training epochs.



Comparisons

The following is a summary of results over all five tuned models. These were results on the original 80-20 train-test-split.

	NBA Train Acc	**NBA Test Acc**	**NBA Fit Time**	**Fraud Tr Recall**	**Fraud Te Recall**	**Fraud Fit Time**
Decision Tree	63.0%	62%	6.00s	98.0%	98.0%	16.00s
AdaBoost	61.8%	61.6%	192.81s	84.1%	81.7%	317.20s
KNN	57.0%	57.0%	3.50s	100%	100%	0.20s
SVM	30.7%	29.9%	10270s	50.8%	45.5%	1843s
Neural Network	45.9%	45.8%	38.90s	80.1%	76.1%	54.57s

1. Model Fit Times

The iterative models (AdaBoost, Neural Network to an extent) took longer than the non-iterative models (Decision Tree, KNN), because of its iterative nature. SVMs with Linear kernels took extremely long to train, likely because of the model trying to optimize the hyperplane across so many dimensions. There were 753 feature dimensions in the NBA Shots data and 29 dimensions in the Fraud data.

2. Test Accuracy for NBA Shots

The Decision Tree model got the highest testing accuracy for this dataset, though the AdaBoost model performed only 0.4% worse. Both these models are tree-based, thus it seems that the segmentation of the feature space really benefits separating the two classes in this dataset. For the Decision Tree model, it isn't as overfit as Decision Trees tend to be as maximum tree depth was cut off at 10.

I would be wary of using this Decision Tree model to make predictions, especially because this dataset comes from a single season of NBA games from 2014-2015. I would either help it to generalize by training it over more years of data, or only use it for inference or prediction for this particular season.

3. Test Accuracy for Fraud

For this dataset, the KNN model was able to produce a testing recall of 100%. I believe that this is because there were so few instances of Fraud transactions, and the nature of these Fraud cases were similar enough such that whenever there was a Fraud case in the testing dataset, its nearest neighbor would also be a Fraud case from the train set. This seems to be a case where setting the *n_neighbors* parameter in the model to be equal to one is beneficial. This also shows that the two classes of data in the feature space are spatially separable.

4. Bias-Variance Tradeoff

In each of the tuned models, there was little evidence of overfitting. Each of the models' training and testing metrics were within 5% of each other. I believe this to be the case for different reasons for each model:

- For the Decision Trees, maximum tree depths were pruned to much less than their deepest
- For AdaBoost, ensemble methods in general reduce overfitting as each iteration of weak learner focuses on a differently-weighted sample of the training data
- For KNN, n_neighbors=1 for the Fraud data helped both training and testing recall to reach 100%, as already explained above. The n_neighbors=50 for the NBA Shots data gave the model the ability to aggregate the classes of test data points' 50 nearest neighbors, leading to overall smoother decision boundaries and less overfitting.

Data Sources

- NBA Shots Data. Retrieved Sep 10, 2021 from https://www.kaggle.com/dansbecker/nba-shot-logs
- Credit Card Fraud Detection Dataset. Retrieved Sep 10, 2021 from https://www.kaggle.com/isaikumar/creditcardfraud