Assignment 1: Supervised Learning

The problems

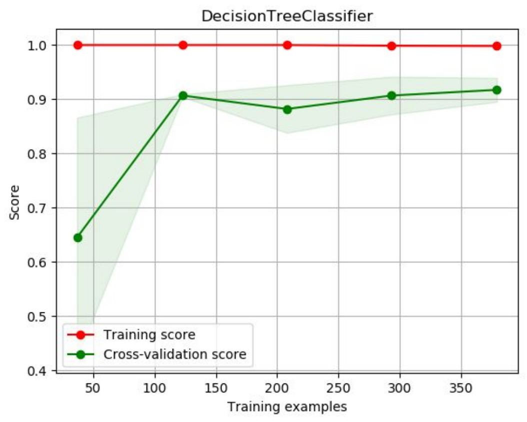
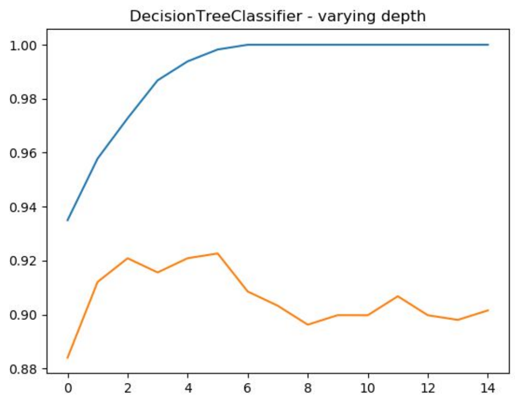
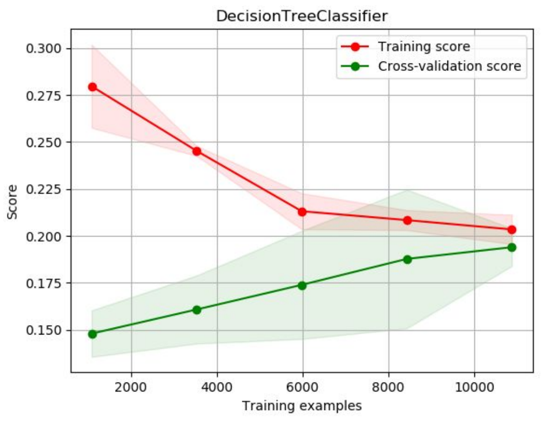
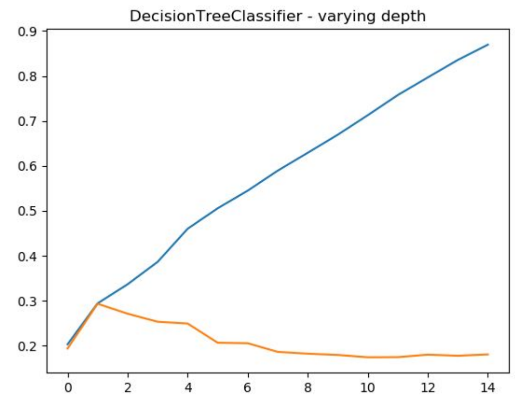
For this assignment, I browsed a variety of public datasets available on Kaggle.com. I was looking for datasets that could be interpreted as classification problems, that contained mostly numeric or categorical data, and were relatively highly rated on the site. I avoided image-based data sets, because I wasn't certain if those datasets would be tractable or meaningful with some of the algorithms, like K Nearest Neighbors.

I wanted to use a credit card fraud dataset, but discarded it because I already had a binary classification problem that I was interested in, and because the complexity of analyzing that dataset would be very high, since fraud datasets contain very few examples of fraud relative to examples of not-fraud. For such datasets, I'm aware of advanced techniques like generating artificial fraud samples or down-sampling non-fraudulent examples, but given the complexity of this task (five algorithms) and my relative inexperience with some of the libraries I shied away from this approach. I'm also aware that special measures of accuracy, specifically recall and precision, which are relevant especially to fraud datasets.

The first problem I knew I was interested in was a dataset containing properties of tumor images and a classification of malignancy or non-malignancy. Generally, I'm quite interested in the potential for artificial intelligence techniques to increase productivity in medicine, automatic diagnosis or flagging being one clear avenue. I liked this dataset because it abstracted away the image processing part of the task without solving it entirely. This is a binary classification task, meaning the complexity is somewhat lower.

The second problem I selected was based on a dataset of video games. I thought this would be a good dataset to use because it was relatively large, but more importantly contained several potential classification problems. There are several categorical variables in the dataset, ranging from publisher, platform, and genre of video game published over the years. I chose to categorize platform, because I thought it would be interesting to consider how some factors like publishing date could strongly affect the probability of belonging to one category or another. One drawback, however, was that some rows contained nulls and had to be dropped, and some of the 31 platform categories had very few examples in the dataset. Additionally, the number of categories were made the problem challenging for most of the algorithms, as well as computationally expensive.

Decision Trees

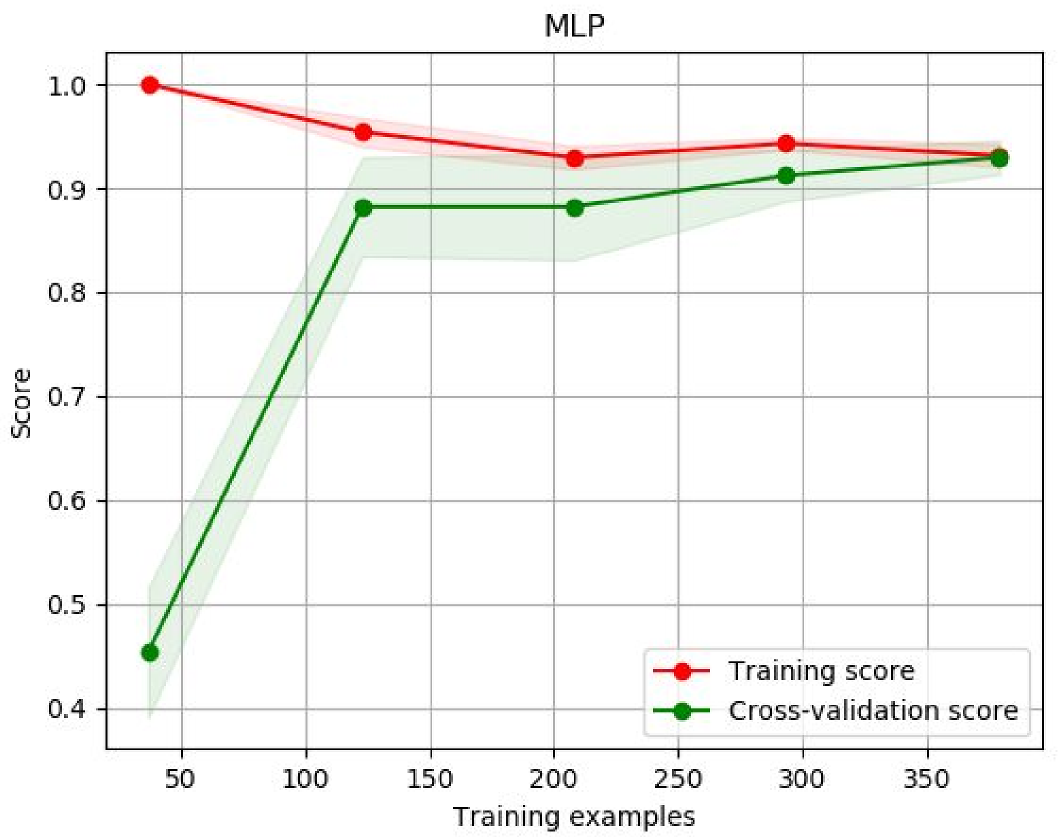
Learning Curve, Cancer Dataset	Training (blue) and Testing (orange) performance as a function of max_depth
	
Learning Curve, Video game Dataset	**Training (blue) and Testing (orange) performance as a function of max_depth**
	

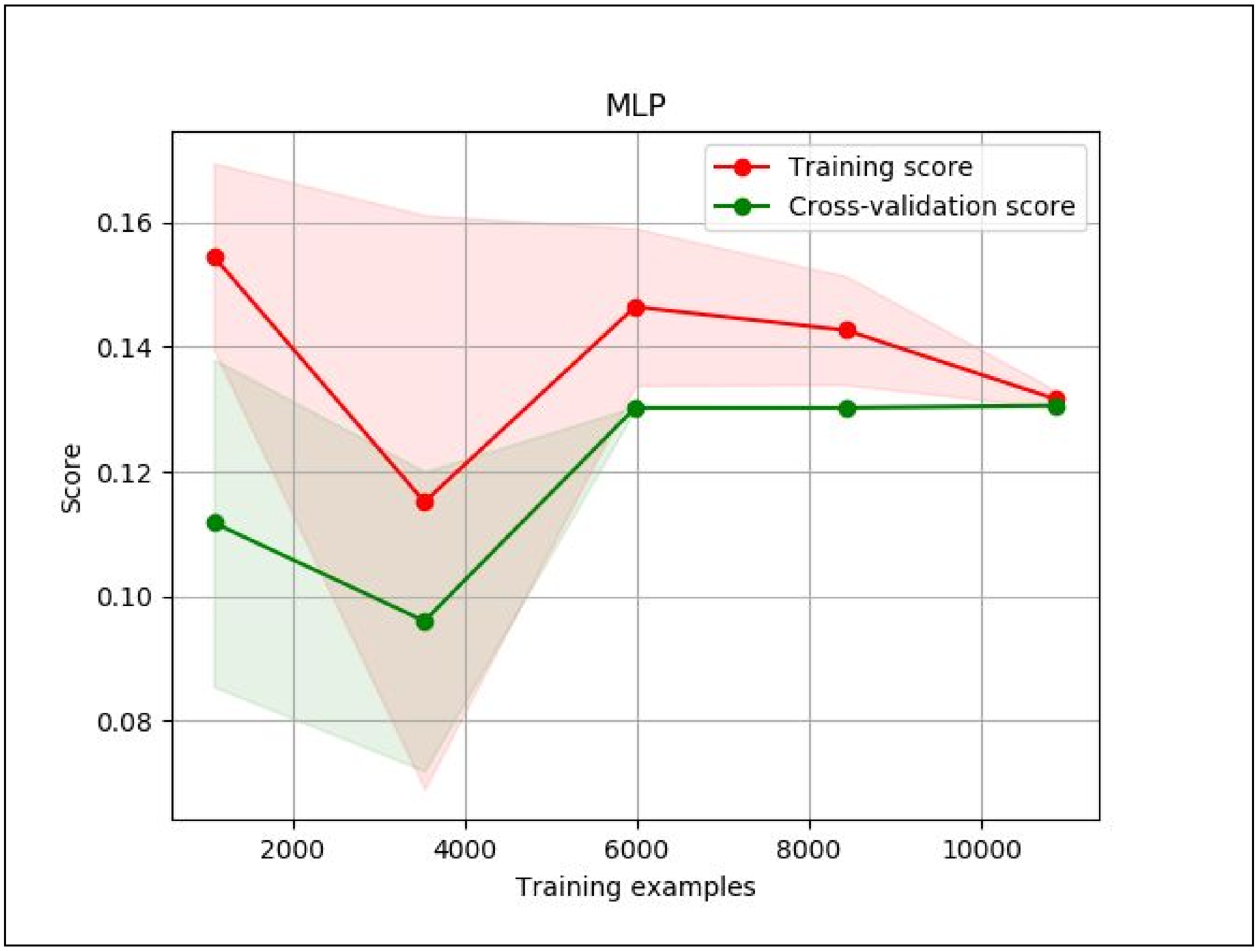
The cancer dataset had about a 91% accuracy for the decision tree classifier, while the videogame dataset had only 19.39% accuracy.

I explored different values of maximum depth for the decision tree algorithms and selected values that seemed appropriate and seemed to maximize the cross-validation testing accuracy.

While the cancer dataset's training and cross validation scores performed well with increasing data, the videogame datasets training accuracy decreased as samples were added. Still, as training increased there was an increase in cross validation score. I suspect that the number of categorical outcomes in this case presented a challenge for the decision tree algorithm. The decision tree performed much better on the binary classification task, perhaps as a consequence of being structured as a sequence of binary decisions.

Multi-Layer Perceptron

Learning Curve, Cancer Dataset

Learning Curve, Video game Dataset

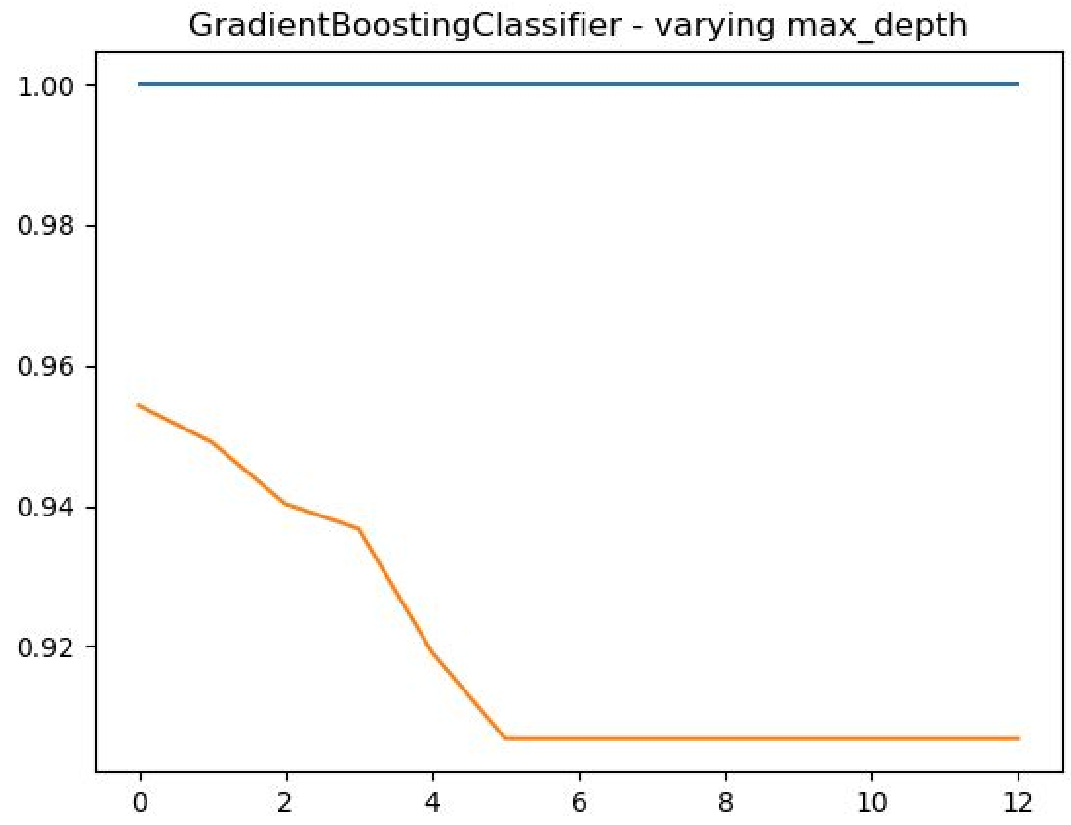


I used a multi-layer perceptron to investigate the data sets. While the cancer data set was predicted with nearly 93% accuracy, the videogame dataset was predicted with only about 13% accuracy, which was particularly dismal.

I selected hyperparamters by hand, after feeling out some results with some other datasets I had been experimenting with. I noticed that *generally* more nodes were better, and that *generally* having nodes equal to or slightly greater than the number of features produced more meaningful results than having fewer nodes. For this reason, I selected three hidden layers of ten nodes for the cancer data set, and three hidden layers of twenty nodes for the videogame dataset. I would have liked to set up a system to explore the possible combinations of hidden layers, but after some experimentation realized that the search space was quite large and settled on making observations and generalizations based on simpler and faster datasets.

The cancer dataset seemed to perform better with each new training example, and while the videogame dataset behaved similarly, it appears to have much higher variance in performance, especially least initially. In general, (owing to the low accuracy) the videogame dataset doesn't seem to be particularly appropriate for the MLP, perhaps (again) owing to its large output space and it's (occasionally) limited number of examples for each category of output.

Boosted Decision Tree



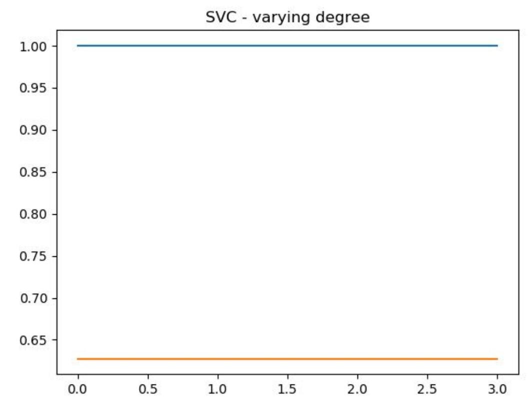
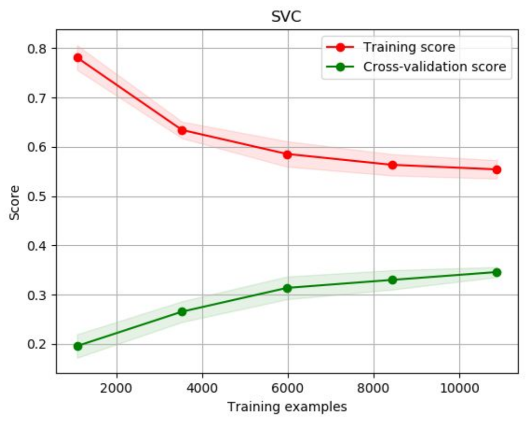
I had quite a challenge in applying the Gradient Boosted decision tree ensemble method to both datasets. I encountered errors which I was not able to resolve when I tried to generate the learning curve graphs, so there is limited information I could glean from this experiment.

I was able to explore the max_depth hyperparameter for the cancer dataset only, but was surprised to find that “shorter” max depths seemed to outperform deeper ones. I used this graph to select a limited depth for the model. I suppose that when the models are limited to such a short depth, the boosted decision tree, which aggregates the results of many weak models, may behave more like non-boosted decision trees.

For the videogame dataset, the model took a very long time to run, and I was not able to explore the max_depth parameter (which would require running model many times).

The overall accuracy for the cancer dataset was about 96%, while for the videogame dataset slightly less than 3% was achieved.

Support Vector Machine

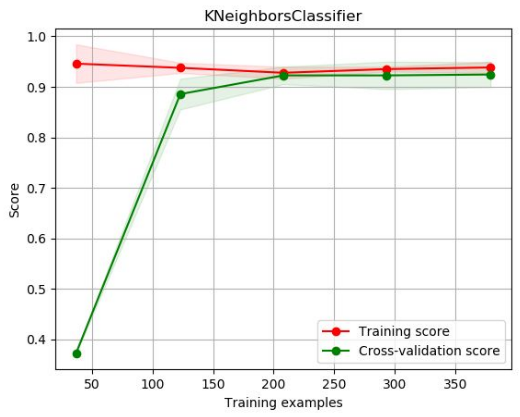
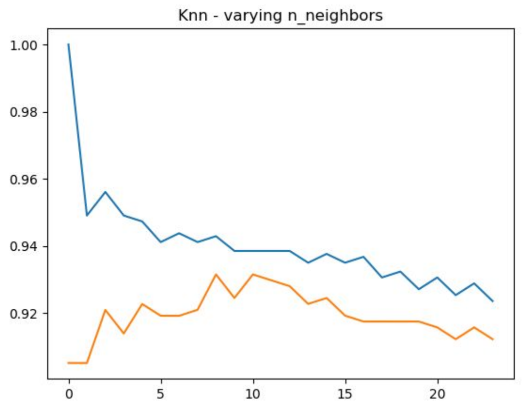
	Training (blue) and Testing (orange) performance as a function of degree
	
Learning Curve, Video game Dataset	
	

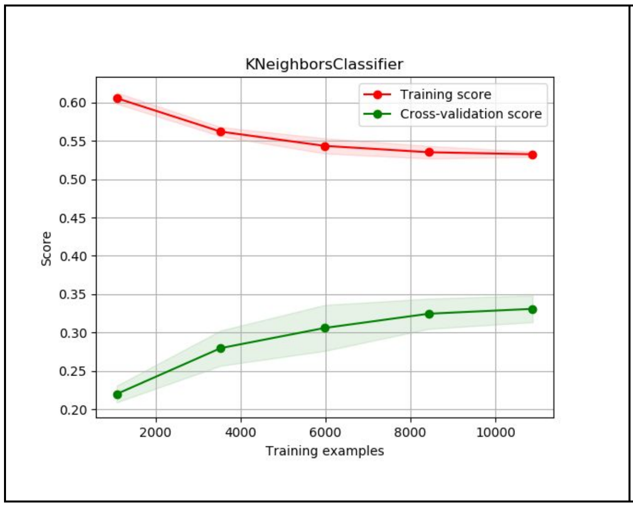
I also had some difficulty generating graphs for Support Vector Classifiers -- I got the same strange errors for the cancer dataset as I got for the boosted decision trees. Still I was able to generate a graph for the videogame dataset, and an exploration of the degrees hyperparameter for the cancer dataset.

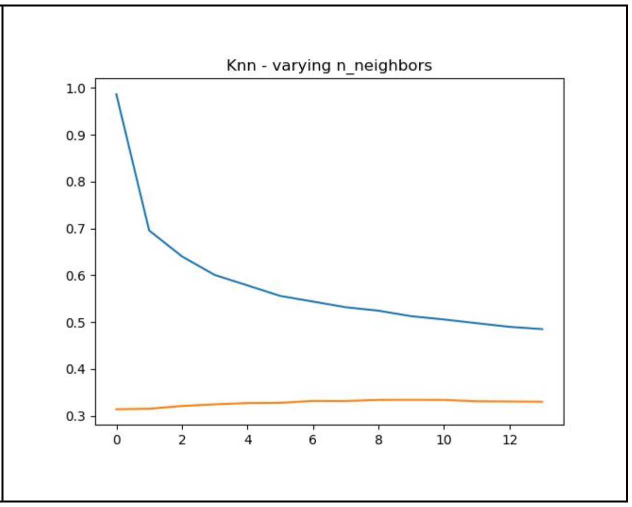
The model performed with 62% accuracy for the cancer dataset and 34% accuracy for the videogame dataset. This model performed the best out of all the models attempted on the video game dataset.

Surprisingly, it seems like there was little change when varying the degrees of freedom for the model on the cancer dataset. Because of this, for both datasets, I selected a degree of 3, which seemed reasonable based on reading about the model. It would be interesting to find a dataset in which the degrees mattered.

K Nearest Neighbors

Learning Curve, Cancer Dataset:	Training(blue) and Testing(orange) Accuracy as a function of number of voting neighbors, Cancer Dataset:
	
Learning Curve, Videogame Dataset:	Training(blue) and Testing(orange) Accuracy as a function of number of voting neighbors, Videogame Dataset:





K-nearest neighbors is an algorithm that uses a measure of distance, coupled with a voting mechanism, to predict examples.

The cancer dataset had an overall cross-validation accuracy of 92.45%, while the videogame dataset came in at around 33.18%.

For both datasets, I explored different settings for K, the number of neighbors who are permitted to vote to categorize examples, and selected the one that seemed most appropriate. For both datasets, the cross-validation score climbed more or less monotonically, constantly improving accuracy.

K-nearest neighbors is a relatively straightforward classification algorithm which performs well on both datasets, all things considered.

Conclusion

The overall results follow:

Cancer Data Set	
KNeighborsClassifier	92.45%
DecisionTreeClassifier	91.21%
MLPClassifier	92.97%
SVC	62.74%
GradientBoosting	96.48%
Video Game Data Set	

KNeighborsClassifier	33.18%
DecisionTreeClassifier	19.40%
MLPClassifier	13.06%
SVC	34.60%
GradientBoosting	2.92%

I’m most familiar with the KNN classifier, and was not surprised to see it perform relatively well on both datasets. This was also the implementation I felt most comfortable applying.

I think it’s quite interesting to see how poorly the SVM Classifer (SVC) did on the “easier” dataset (cancer) and how well it performed on the Video Game dataset.

I’m a little disappointed that the MLP performed so poorly on the video game dataset. I suppose there may be additional hyperparameter tuning but I felt somewhat overwhelmed by the sheer size of the search space -- even limiting myself to just number and sizes of hidden layers presented many options.

I’m also interested in performing further experiments on the Video Game Dataset, but predicting different variables -- is it easier, for example, to categorize by publisher, or genre? I think one major drawback of selecting the platform is that there were many platforms possible and some had very few examples.

Code referenced:

I heavily depended on code from the documentation of SciKit Learn in order to complete this project, in particular some plotting functions related to learning curves enabled me to produce many of the graphs. Outside of standard SciKit Learn documentation and example code, I consulted few references.