Supervised Learning: Analysis

Vishwa Shah

Classification Problems

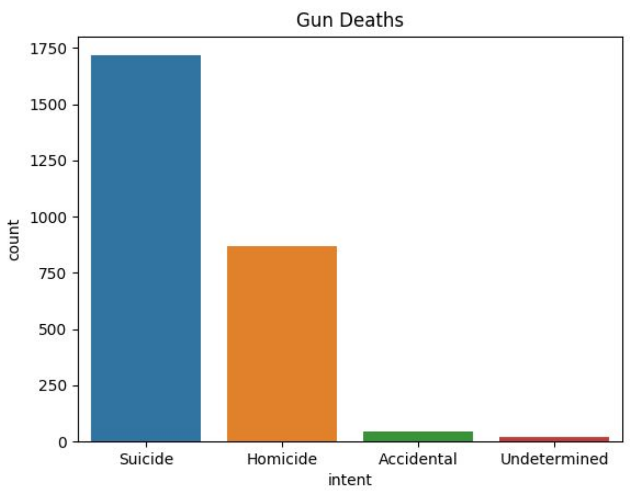
Given the task of designing two interesting classification problems, I began by looking for two dissimilar datasets where classification was relevant. Attributes of the data I looked for to be different among the two included size, number of features, domain, and intuitive ease of classification. Since the analysis would further be aided if the models were classified differently by different classifiers, I hoped by different domains I could also choose datasets with different relationship models within the data— some of which were more apparent than others. The 2 problems I chose have interesting overlap in structure but enough different to be able to compare algorithm performance and begin to attribute the outcomes to the form of the data.

Gun Deaths

This dataset contains data of race/ethnicity, age, sex, police presence, education, place, and intent of gun death from 2012-2014. For brevity and in the interest of time, I am using just the data from January 2014, which comes out to 2650 examples. The classification problem here is of intent; predicting whether the death was a suicide, homicide, accident, or undetermined.

Preprocessing

In order to ensure the data was read correctly, I encoded all the categorical columns as one-hots. This includes education, place, race/ethnicity. I also converted the hispanic column to the categories they fall into more broadly, as opposed to using the number coding before converting it to one-hot columns. I also removed any examples with incomplete data. This puts this data at 32 features.

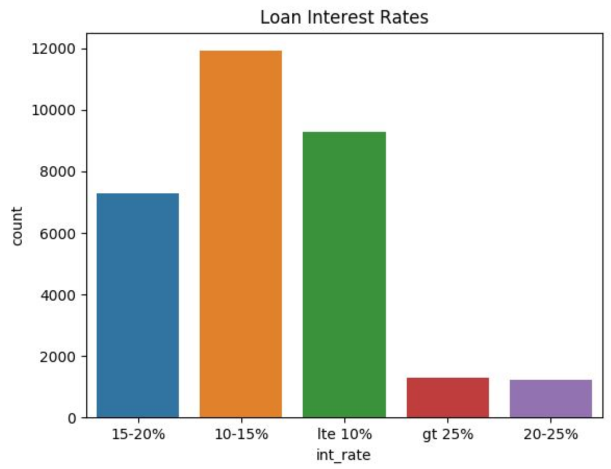


Distribution

It is important to note that the distribution of this dataset is quite skewed towards suicide and homicide with little representation of accidental and undetermined, which makes sense. In every cross validation and sampling split, I'll use a stratified approach to ensure representation.

Loan Data

This data was taken from Lending Club's 2017 Q3 dataset. The dataset is quite large, and again to in the interest of performance and time I used only the data from July 2017 filed as individual loan apps with current loans. This reduces the size from 128K to about 39K examples. There are numerous features, however I decided to account for only some of them. This made it a more interesting problem in my eyes, as I wanted to abstract the data from being able to actually calculate an exact formula given the amount of data provided. As a results, I'm using the more obvious features such as the amount funded, term of loan, installment size, annual income, purpose, verification status, and employment length to predict the range in which the interest rate falls in. There are 5 categories: ≤10%, 10-15%, 15-20%, 20-25%, and >25%.



Preprocessing

For this dataset, I once again encoded all categorical columns as one hot columns and removed all examples with incomplete data in relevant columns. I reformatted interest to be categorical rather than continuous for the purpose of this classification problem. After all this, the dataset is at about 39K examples and 23 features.

Distribution

This dataset has a much better distribution of categories, although the first 3 still overpower the last two. Again, I'll be using a stratified sample for every sampling/test split to ensure representation.

Interesting?

Training supervised learning classifiers on these datasets can prove to be useful in the real-world. Gun deaths are controversial, and being able to relate the background of the person to the intent can help provide insight on unknown cases and possibly inform gun ownership. Loan Interest predictors can be useful as calculators/estimators online for people interested in applying for a loan.

In terms of this assignment, however, these datasets are interesting because they're different in some ways, providing insight on to how these algorithms perform differently. Since some algorithms perform vastly better than others, one can analyze why such patterns exist.

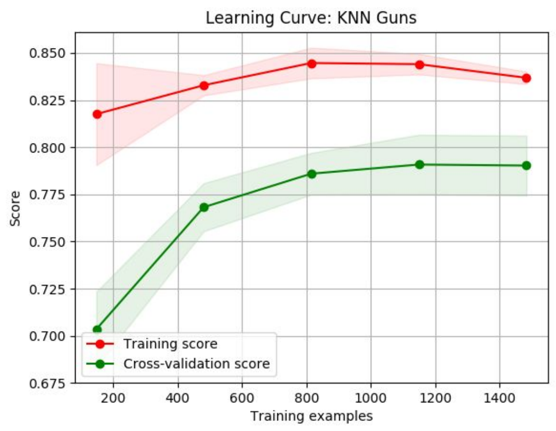
The two problems differ primarily in number of examples and secondarily in number of features, in addition to being in different domains with different degrees of relationship with their features. One large domain difference is the mathematical nature of the relationship in the loan data versus the categorical, subjective nature of the gun death data. There is likely a stronger hierarchy in the way the gun data is related, based on how one might think about approaching the problem of figuring out the cause of a gun death while loan data is more numerically based.

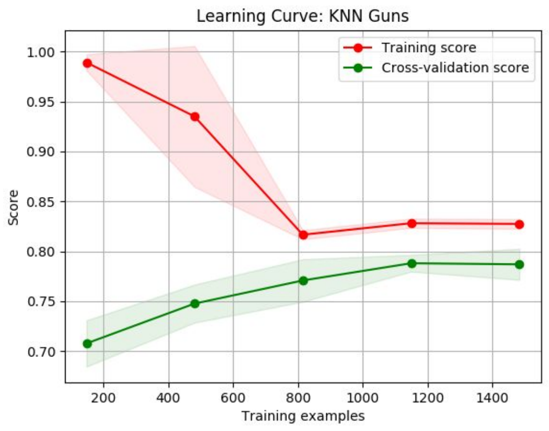
Algorithmic Performance

k-Nearest Neighbors

I ran the default classifier followed by the estimator returned by GridSearchCV for both datasets. The learning curves produced by each of these provide some insight on to how KNN performs on the datasets and how fine tuning hyperparameters affected that performance.

First, I have the Gun death dataset:



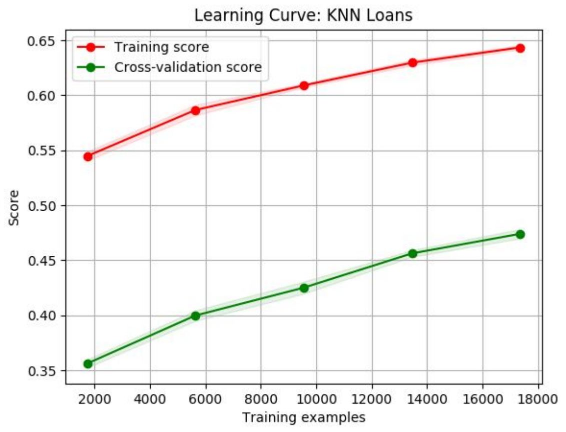


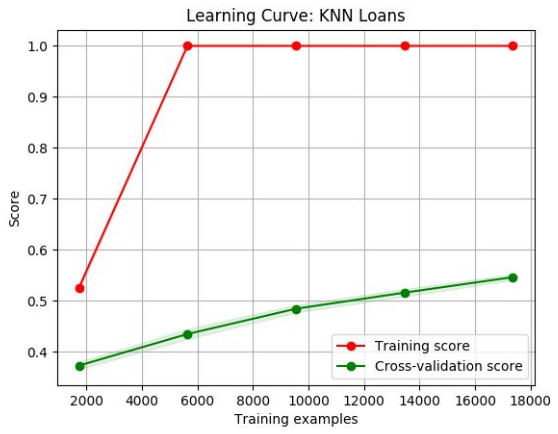
Here (above) the lines converge closer on the latter, at a marginally higher score. GridSearch couldn't find a significantly better estimator, so KNN settles at approaching 80%. It's important to consider here that skewed distribution of the dataset; with mostly Suicides and Homicides, KNN could be great for the samples in high-density regions but weak on the scattered, random ones (accidental and undetermined).
This can be seen better through the confusion matrix of the KNN classifier on the training and test sets:

Training	Test
('Confusion Matrix: ',	('Confusion Matrix: ',
array([[0, 9, 21, 0],	array([[0, 3, 10, 0],
[0, 410, 200, 0],	[0, 177, 84, 0],
[0, 74, 1127, 0],	[0, 51, 464, 0],
[0, 2, 12, 0]]))	[0, 1, 5, 0]]))
('KNN Accuracy: ', 0.82857142857142863)	('KNN Accuracy: ', 0.80628930817610067)
('KNN F1 score: ', 0.82857142857142863)	('KNN F1 score: ', 0.80628930817610067)

The confusion matrix clearly shows that nothing is being classified as accidental or undetermined. This is one reason confusion matrices are so important; they can yield valuable insight on how a classifier is doing.

Moving on the the Loan Dataset— a less skewed situation. Again, here are the graphs for the default and then grid searched classifiers.





These could both benefit significantly from more samples to get the cross validation scores closer to the training score. The cross validation does poorly, hitting about 55% although the tuned hyperparameters give a 100% training score.

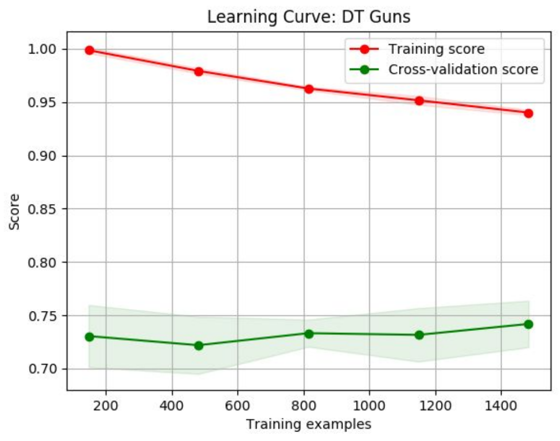
Training	Test

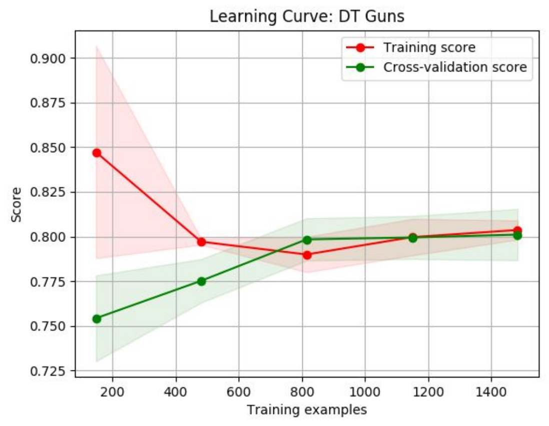
('Confusion Matrix: ',
array([[8343,   0,   0,   0,   0],
       [0, 5107,   0,   0,   0],
       [0,   0, 866,   0,   0],
       [0,   0,   0, 909,   0],
       [0,   0,   0,   0, 6489]]))
('KNN Accuracy: ', 1.0)
('KNN F1 score: ', 1.0)
	('Confusion Matrix: ',
	array([[2349, 489, 44, 64, 630],
	       [677, 1111, 55, 43, 302],
	       [85, 152, 59, 18, 57],
	       [134, 91, 21, 68, 76],
	       [715, 240, 20, 43, 1763]]))
	('KNN Accuracy: ', 0.57489791532344725)
	('KNN F1 score: ', 0.57489791532344725)

Here it's evident that while KNN performs a lot worse on the test set, it at least has access to enough density of data for each category to predict in it— not the same issue as the Gun Data. It appears as if there isn't enough relation by proximity here for KNN to be strong, although that seems to go against my perception of how the data should be related.

Decision Trees

Decision Trees take a very different approach than KNN, in terms of what kind of relationship between data points is built upon. First let's explore unpruned DTs, and then compare them to the pruned counterparts.

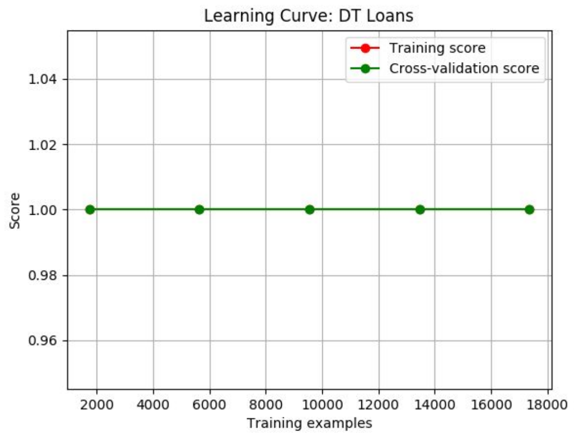


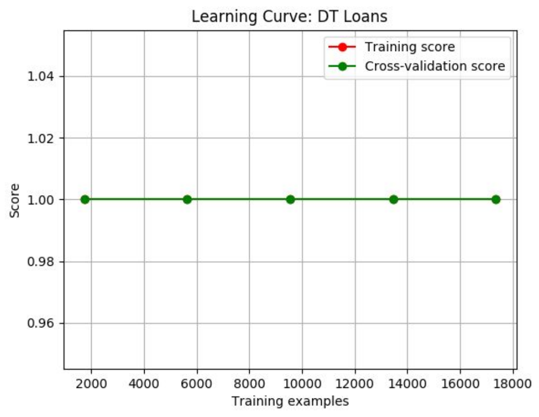


In this first case, the DT trains at a high score but cross-validates much lower. The grid searched estimator, on the other hand, builds a stronger DT classifier where the lines converge. This classifier cannot benefit much by more samples, so it's potential settles around 80%. Just as we saw in KNN, here there are also no predictions for the smaller 2 categories. In fact, not a single classifier picked up the pattern (or lack thereof) to predict accidental/undetermined. That speaks to the lack of pattern and an algorithms inability to register the lack of pattern, perhaps, as an indicator.

Training	Testing
('Confusion Matrix: ', array([[0, 11, 19, 0], [0, 444, 166, 0], [0, 152, 1049, 0], [0, 5, 9, 0]])) ('DT Accuracy: ', 0.80485175202156334) ('DT F1 score: ', 0.80485175202156334)	('Confusion Matrix: ', array([[0, 2, 11, 0], [0, 203, 58, 0], [0, 75, 440, 0], [0, 1, 5, 0]])) ('DT Accuracy: ', 0.80880503144654092) ('DT F1 score: ', 0.80880503144654092)

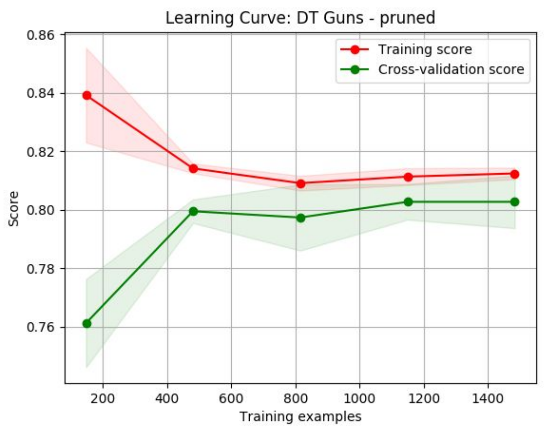
Loans, on the other hand, seem to be very effectively modelled by a Decision Tree; perhaps this is where it's formulaicity can be seen easily.

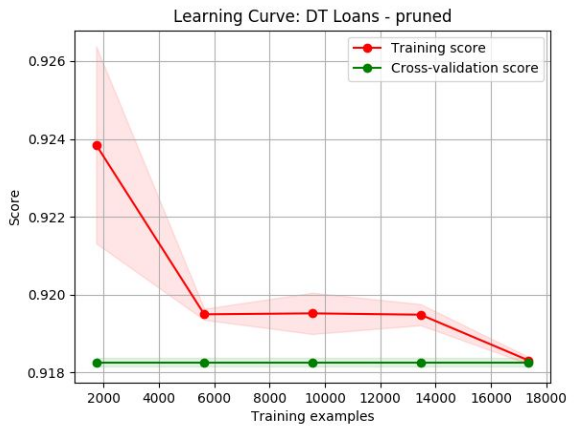




Both the default and fine-tuned classifiers yield scores of 100%, for training and CV. The data's domain lends itself to a tree organization to determine interest rates, so this isn't all that surprising. The classifier also performs at 100% on the test set.

Pruned Decision Trees



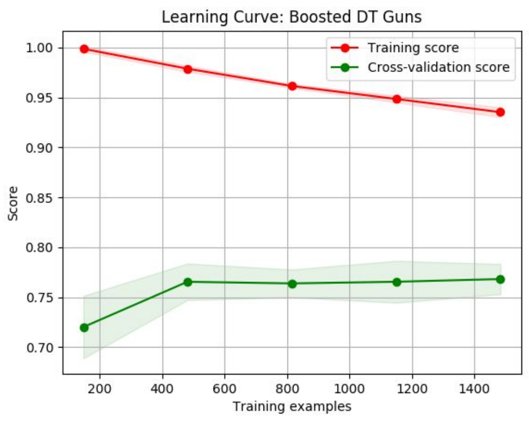


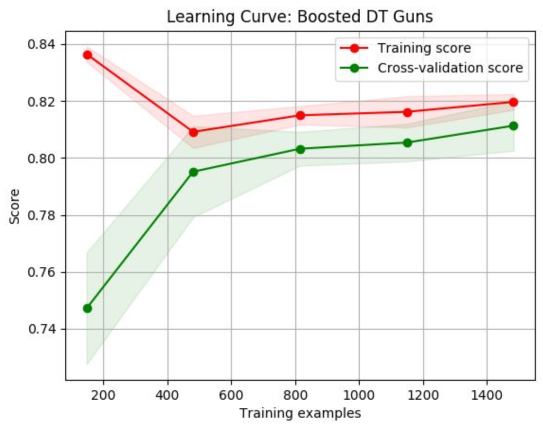
As expected, pruning the tree takes a hit on the performance for Loans, but keeps performance at about the same place fo the Gun Deaths. These are post-pruned for depth to have a max-depth of 3, which would reduce how many steps there were and thus how many features could be considered. Since there are fewer steps considered, however, predictions would be faster here.

On the test set, the pruned DT performed at 77.8% for Guns and 91.8% for Loans.

Boosted Decision Trees

Since Decision Trees work really well for the Loan dataset, and just alright for the Guns, I was curious as to what boosting could do here.

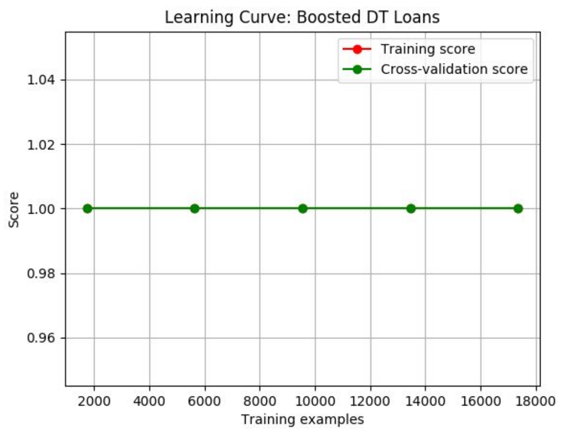


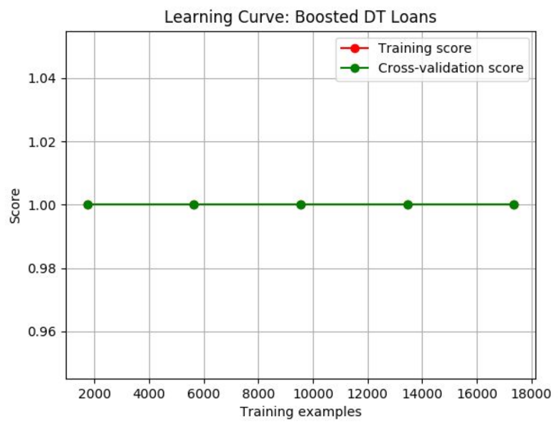


The hyperparameter adjustments brought the lines together just as it did above for the regular decision tree. The percentage that's being predicted is still fairly low, but higher than the regular decision trees.

Training	Testing
('Confusion Matrix: ', array([[0, 10, 20, 0], [0, 433, 177, 0], [0, 112, 1089, 0], [0, 4, 10, 0]])) ('Boosted DT Accuracy: ', 0.82048517520215636) ('Boosted DT F1 score: ', 0.82048517520215636)	('Confusion Matrix: ', array([[0, 2, 11, 0], [0, 196, 65, 0], [0, 60, 455, 0], [0, 1, 5, 0]])) ('Boosted DT Accuracy: ', 0.81886792452830193) ('Boosted DT F1 score: ', 0.81886792452830193)

Since the lines are converging on the second graph there, there goes to show there are enough samples to bring it to that point. Perhaps what lacks with this dataset is a relationship to be picked up on for undetermined/accidental.

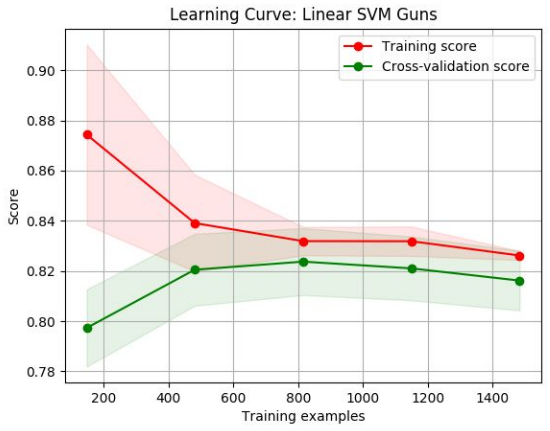


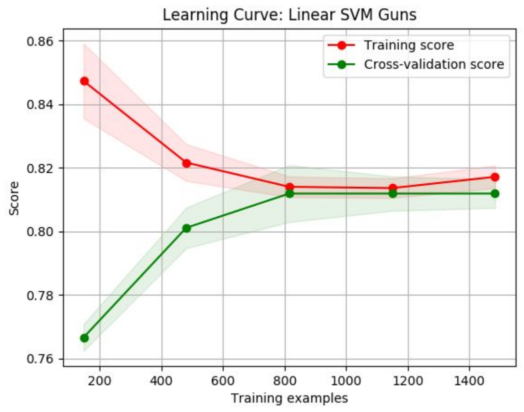


For the loans, once again since the regular DT was already at 100%, so are these. The test sets also performed at 100%.

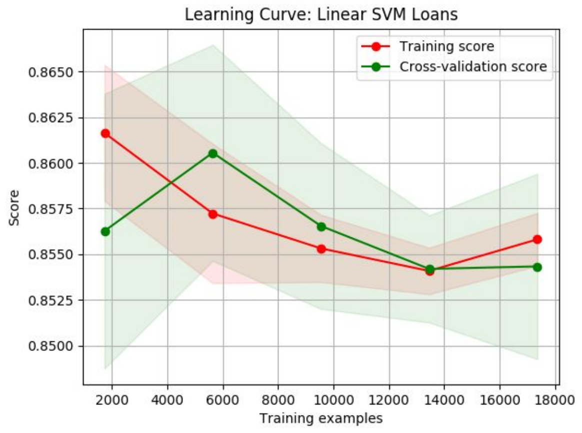
Support Vector Machines

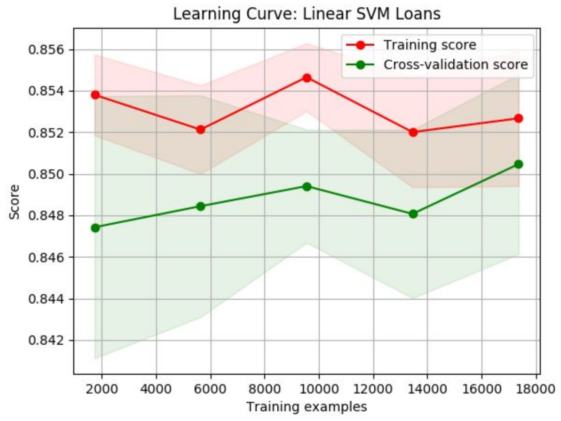
SVMs are interesting in multi dimensional spaces, just increasing the dimensions until data is found to be linearly separable. Here is makes sense that the Gun data isn't predicted outside of the two main categories; the others truly do not exhibit a pattern. The grid searched estimator for Linear SVM has a closer training/CV relationship, but converges slightly lower than the default parameter estimator. That same data, once again Guns, performs around the same with a different SVM estimator.



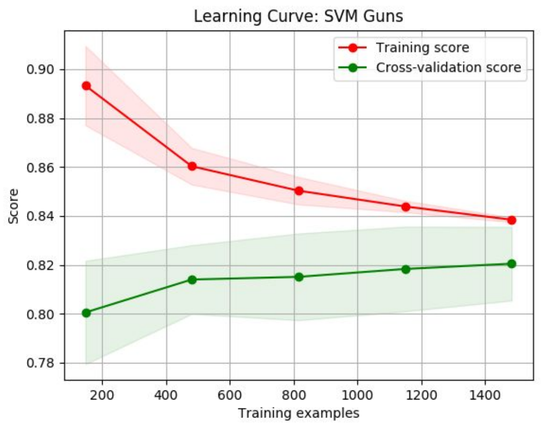


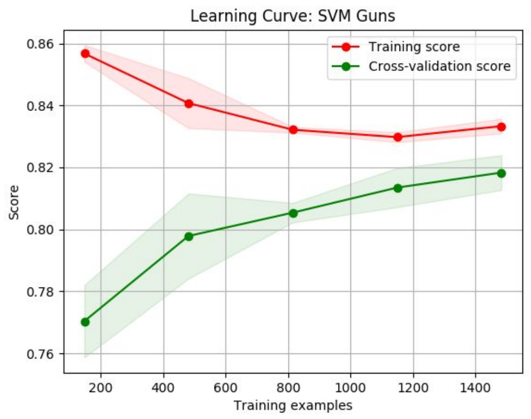
The gun data consistently performs around 80%, with test data for this particular Linear SVM coming in at 81.8%.



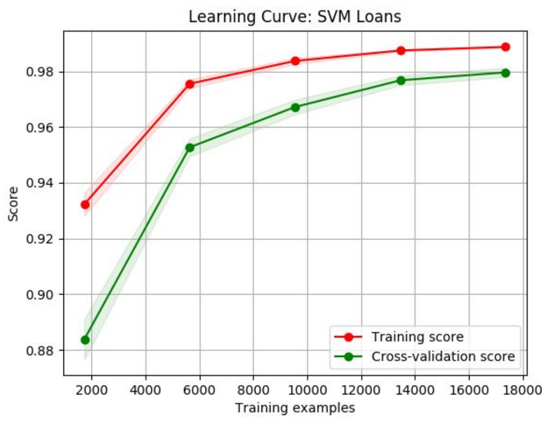


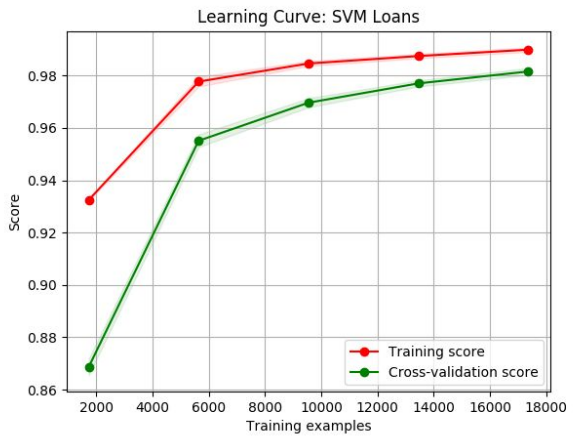
Test data for this one performs at 30.6%, much worse than training and cross validation scored. This could be due to overfitting, but is more likely due to the data not being easily linearly separable.





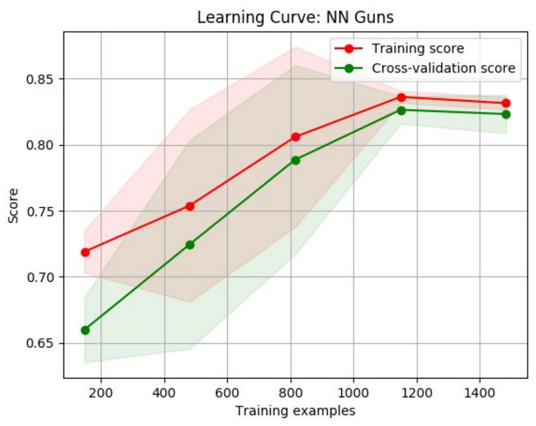
The test data for this performs at 80.9%. It's pretty good for the bulk of the data, which is Suicides and Homicides. The predictor is not tailored for all the random outliers that constitute the Accidental/Undetermined sections.

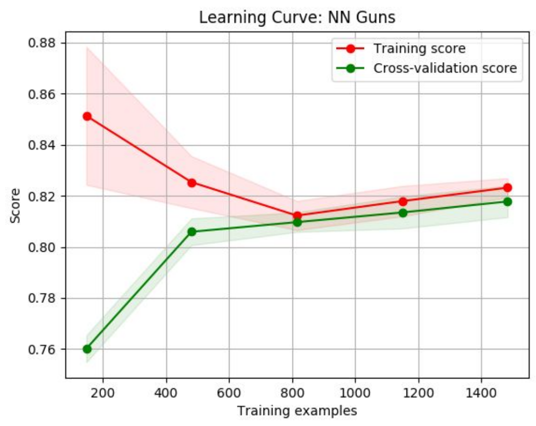




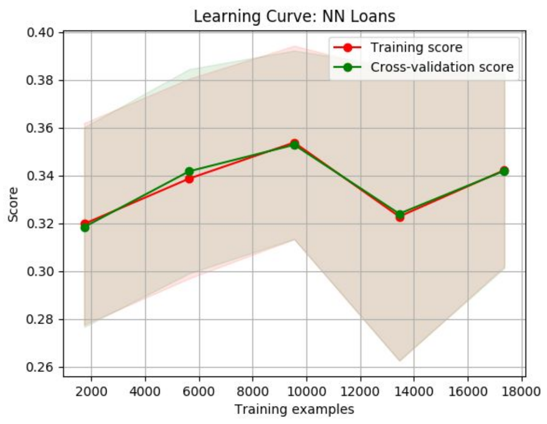
This looks pretty good for interest rate predictions as the training percentage is high, but the test data is at about 54.9%. I think this goes on to show how SVMs are not a good representation of the data that loan interest rate is calculated as, while Decision Trees do a much better job.

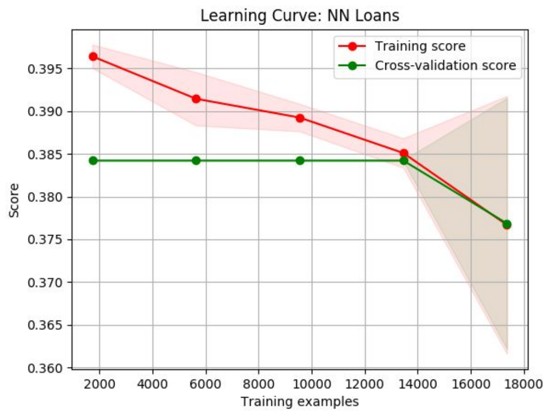
Neural Nets





For neural nets, this is actually where the Gun dataset performs the best, although only by a percent. The test set gives 83.4%. These learning curves show that there is no true lack in data that prohibits the neural net from learning; CV and training score converge. The inefficiency in its prediction is largely based on the fact that the correlations are relatively weak.





Here it's even more evident that neural nets are not the most useful thing for this dataset; it cannot model what loans' interest rates are as well as other algorithms. The training score is already low as it is, in the 30's, and the test set maintains that at about 38%.

Conclusions

The "Best" Algorithm(s)

I think for the Guns dataset, there was a major flaw in the fact that the dataset was so skewed and the categories such that there was not clear definition/pattern for the terms that there were no great algorithms; many were good at categorizing Suicides and Homicides, but not the rest.

For the Loans dataset, I think it was a lot more clear: the Decision Trees, pruned or not, were quite effective classifiers; unpruned hit 100%, and the pruned ones cut depth significantly and still retained over 90% accuracy. KNN, SVMs, and NN all performed much worse.

Improving Algorithms

I ran GridSearchCV, but in the interest of time and performance could only give a relatively small range for the hyperparameters. Also, there's a dependency on the scorer, which could be skewed. I noticed this because there was one incident at least in which the confusion matrix from the default parameter run was better than that of the grid searched classifier, although the latter did have a slightly higher score. I think it can be useful to favor something that categorizes in all categories but has a slightly less accuracy over an estimator that eliminates a category.