Timtothy Laurent

https://github.com/timothyjlaurent/ga_tech_ml_supervised

Introduction:

Two datasets from Kaggle.com were chosen for this assignment: the Titantic Survival Dataset, and What's Cooking? dataset submitted by Yumly.com. The Titantic dataset is relatively small (n=891) with a well defined feature set that require minimal preprocessing to behave well with the learning algorithms and it has a binary prediction feature, survived or not.. The What's Cooking? dataset is much larger (n=39774) and the features are a type of cuisine (n=20 ex French, Chinese), and a list of ingredients for each recipe, the task being to predict the type of food given some recipes. The latter task took much more effort to get the data in the correct state, dimensional reduction was needed to make the feature set manageable for most of the learning algorithms. The stack used was the IPython Notebook using SciKitLearn, which offered a rich machine learning environment with quick developer feedback loop.

Titantic Survival Data

The Titantic data consists of information on 891 passengers aboard the Titantic. The feature set includes the following features: Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked. Survived is the binary value of whether a passenger survived; Pclass is the socio-economic status of the passenger (1st, 2nd, or 3rd); Sex is the gender of the passenger; Age is the passenger's integer age; SibSp, is the totalr of the passenger's siblings and spouses aboard; Parch is the number of the passenger's parents and children aboard; the Ticket contains a string that has some sort of ticket identifier; the Fare is the price of the ticket in shillings; the Cabin is the Cabin id (but is most often a missing value); and Embarked signifies the passenger's embarking port (C = Cherbourg; Q = Queenstown; S = Southampton).

Here is an example data row:

	PassengerId	**Survived**	**Pclass**	**Name**	**Sex**	**Age**	**SibSp**	**Parch**	**Ticket**	**Fare**	**Cabin**	**Embarked**
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	NaN	S

Data munging

For a first glance at the data, a pairwise plot was made with the Seaborn statistical charting library. Initial exploration of the data revealed some issues with the data that impeded its use in modeling, including missing values in the age category and the string valued categorical variables, Age and Embarked. To fill in the missing values for the Age feature the Mean age for each Pclass was determined (Pclass1 : 38.201, Pclass2 : 29.885, Pclass3 : 25.102) and this value was used for filling in the Age passenger missing an age according to their Pclass. The categorical variables, Embarked and Sex, were transformed with the label_binarize transformation in sckkit learn to make them into binary feature vectors. For this study, Cabin, Ticket, and Name features were left out of the analysis.

The data were split with 90%/10% train/test ratio, 801 samples in the training set and 90 in the test set.

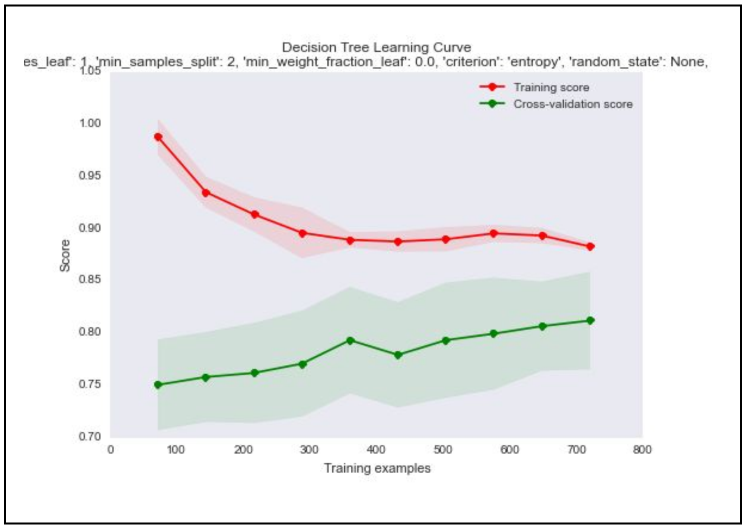
Decision Tree

A Decision tree classifier was made naively, using the default configuration, resulted in a accuracy score of .778 and an internal 10 fold cross validation score of .779. To improve upon this a parameter grid was made of the following shape:

```
{'criterion': ['gini','entropy'],
 'splitter': ['random', 'best'],
 'max_depth': [1,2,3,4,5,6,7,8,9,10,11,12,13,14],
 'max_features': [1,2,3,4,5,6,7,8,9]}
```

The sklearn GridSearch iterates exhaustively over each parameter combination and performs 10 fold stratified k fold cross-validataion to select the best parameter set as a hyperparameter optimizing and pruning strategy. This best estimator used had parameters: {'max_features': 7, 'splitter': 'best', 'criterion': 'gini', 'max_depth': 8} which increased the cross-validation to .822. This pruned decision tree had a more squat structure, preventing overfitting of the more distal branches of the non pruned decision tree.

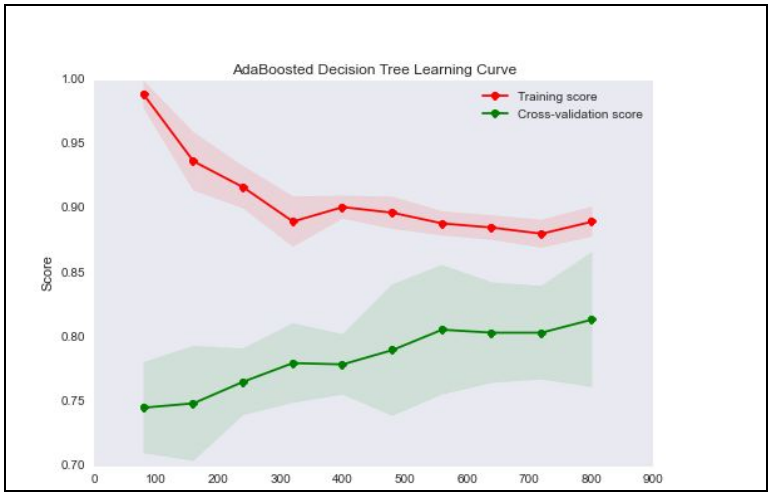
A learning curve on this best estimat showed steady increase in cross-validation accuracy with a decrease in testing accuracy as training examples were added. It is notable that there was a high degree of variability in the cross-validation scores: The time to run the learning curve was 308ms. Finally this decision tree was tested against the test data set and scored .855 accuracy.



Boosting

A an adaboosted decision tree was used and had a .804 10 fold cross validated mean accuracy. To improve on this baseline as parameter grid:
{'n_estimators': [10,25,50,75,100,200,300],
 'learning_rate': [0.01,0.1,1,2,5],
 'algorithm': ['SAMME', 'SAMME.R']}

The grid search ran in 54.3s, the best score was .814 at 10fold cross-validation that had the following parameter ser : {'n_estimators': 75, 'learning_rate': 1, 'algorithm': 'SAMME.R'}. A learning curve showed slightly higher (apparently not significant) training and cross-validatation accuracy than the unboosted decision tree, especially at lower training examples. The final score on our test set was only .77 which is reflective of the high variance of the cross-validation error



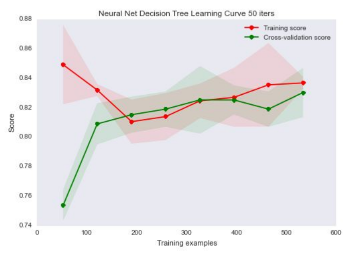
Neural Nets

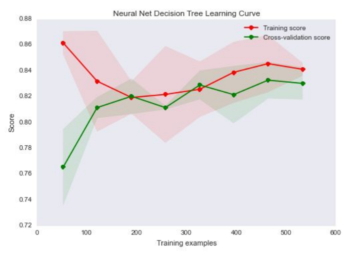
The data was min/max scaled to values between 0 and 1 prior to training the neural nets.The Neural Net was very slow to train, so instead of an exhaustive search of the hyperparameter space a cross-validated random search was employed with the following distribution :

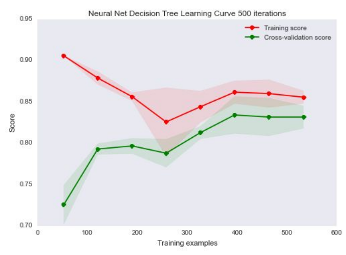
```
{
    'learning_rate': stats.uniform(0.001, 0.05),
    'hidden0__units': stats.randint(2, 12),
    'hidden0__type': ["Rectifier", "Sigmoid", "Tanh"],
    'regularize':["L2",'']}
}
```

the best estimator had a cross-validation score of .827 and the assay ran for 2:32s with 10 random settings. The best parameter was : {'regularize': '', 'hidden0__units': 10, 'learning_rate': 0.021930340599810062, 'hidden0__type': 'Rectifier'}

Learning curves were generated for 50, 100, and 500 iterations of the best performing parameters with 3 fold cross-validation. The networks made the most significant performance increases above 100 sample and gradually bothe training and cross-validation accuracies continue to increase with sample size often overlapping each other. The biggest change with increase of iteration seemed to be a reduction in variance. The final score with our test data with 100 training iterations was .74.







KNearestNeighbor

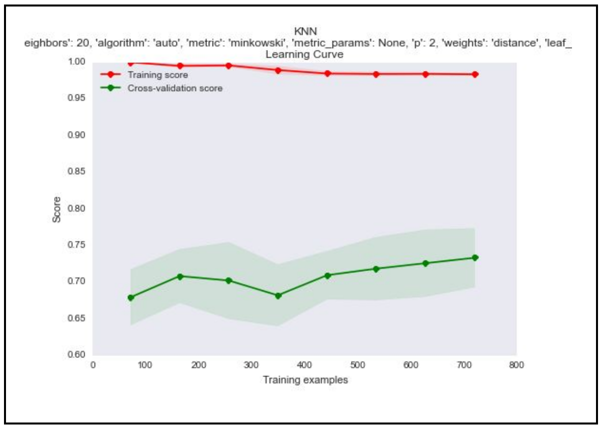
For Knn a parameter of the following shape was used for hyperparameter tuning:

```
{'n_neighbors': [1,5,10,20,30,50,100],
                'weights': ['uniform','distance'],
                'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                'leaf_size': [1,5,10,20,30,50,100]}
}
```

The best estimator had the following parameters:

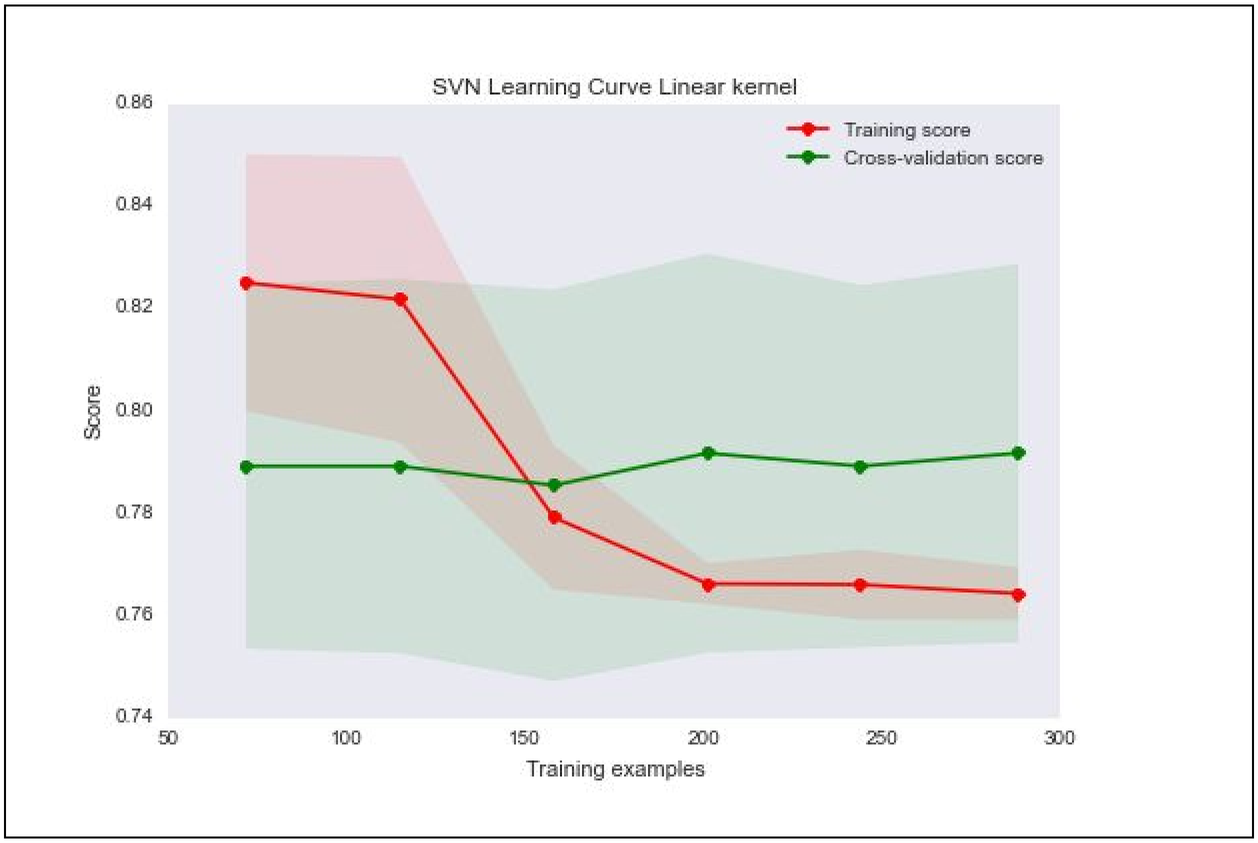
{'n_neighbors': 20, 'weights': 'distance', 'leaf_size': 1, 'algorithm': 'auto'}

and had a 10 fold cross-validation score of .733. The KNN training score was very close to 1 and the cross-validation accuracy only increase moderately to max out around .7.



Support Vector Machine

Support Vector Machines were created the following parameter grid:
{'kernel': ['rbf','linear']}
to try out different kernels for this try out different kernels for the task: The best estimator used a linear kernel and had a 3-fold CV score of .792. Interestinglty the learning curve for the SVM classifier had a stable cross validation score across all tested training sizes at around .79, the training score decreased with more training settling near .76. The final score on the test data was .733



What's Cooking?

The What's cooking dataset consists of 39774 recipes labeled with a cuisine type and a list of ingredients. There were 6714 different strings in the raw data. The words were vectorized by an ngram vectorizer giving each sample a vector of 3010 features (i for each word) and the value being the count of that word in the recipe. Here is an example of the ingredients feature set:

```
...
'passata',
 'oil',
 'ground cumin',
 'boneless chicken skinless thigh',
 'garam masala',
 'double cream',
...
```

here are the cusine types:

```
{'brazilian',
 'british',
 'cajun_creole',
 'chinese',
 'filipino',
 'french',
‘'greek',
 'indian',
 'irish',
 'italian',
 'jamaican',
 'japanese',
'korean',
 'mexican',
 'moroccan',
 'russian',
'southern_us',
 'spanish',
 'thai',
 'vietnamese'}
}
```

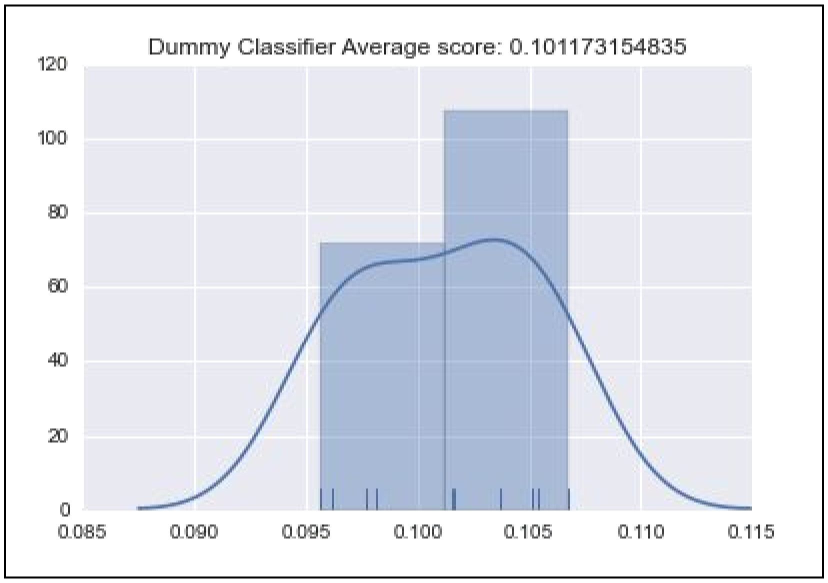
While this dataset was of tractable size for the quick learning algorithms like the Decision tree this number of features proved difficult for other learning algorithms. To make it more tractable the latent semantic analysis was used to create a truncated SVD model. SVD models were made with 50,100,200,300,500,750, and 1000 components. PCA was also explored but the the transformed data work well with the classifiers.

SVD variance explained table

n_components	percentage of variance explained
50	0.598163568865
100	0.732675330504
200	0.848080635678
300	0.899037472944
500	0.944672248812
750	0.969024944047
1000	0.980993477312

Dummy classifier

A dummy classifier was fit to the dataset to give an idea of a baseline. On average only 10% of the samples could be correctly classified with dummy classifier

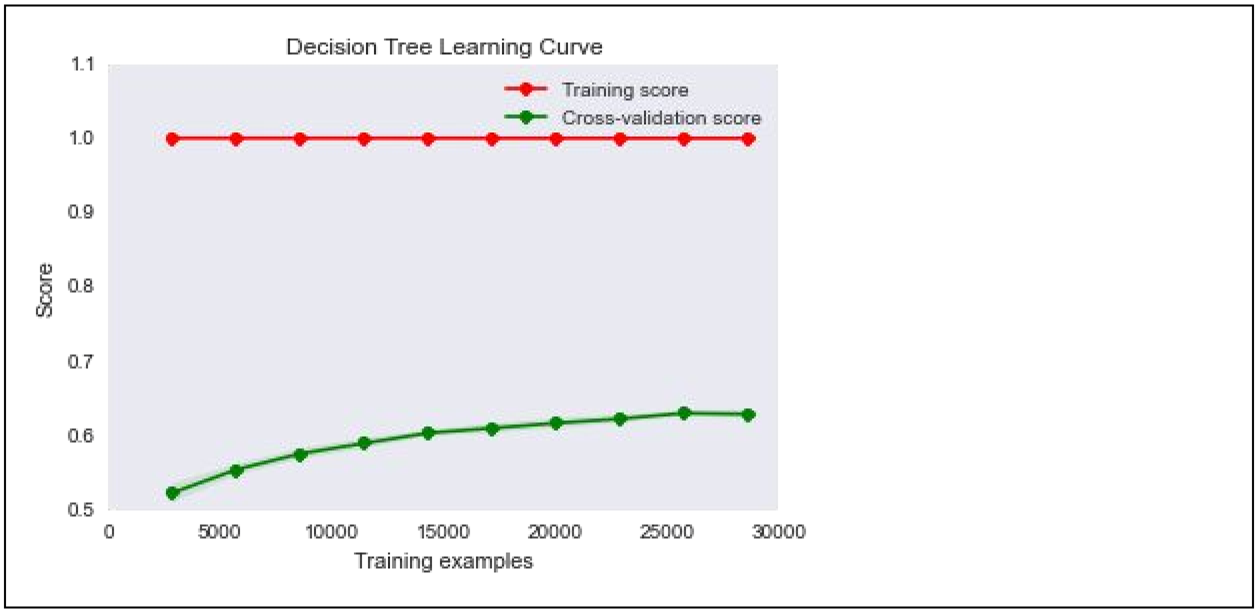


Decision Tree:

Initial tests indictated that we should use the entire data set for training rather than the SVD model: A baseline Decision tree on theSVD data had only a .44 accuracy with the test data compared to .63 with the entire feature set.
To improve the model we tried the following parameter grid :

```
{"max_depth": [10, 20, 50, None],
        "max_features": [1000, 1500, 2000, 2500, 3000],
        "criterion": ["gini", "entropy"]}
```

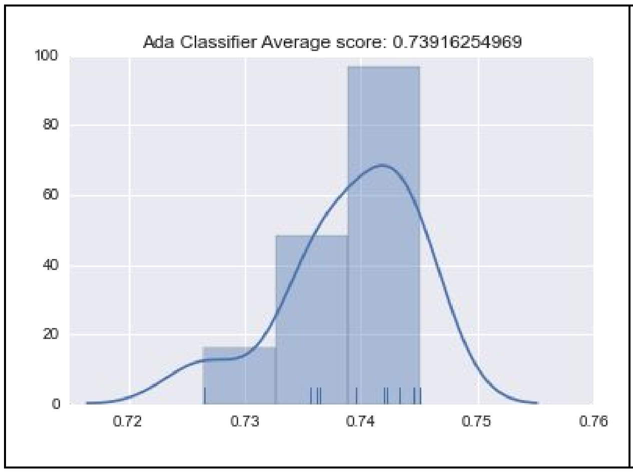
with the best performing grid being having parameter:
{'max_features': 2000, 'criterion': 'gini', 'max_depth': None}
and CV score of 0.6266

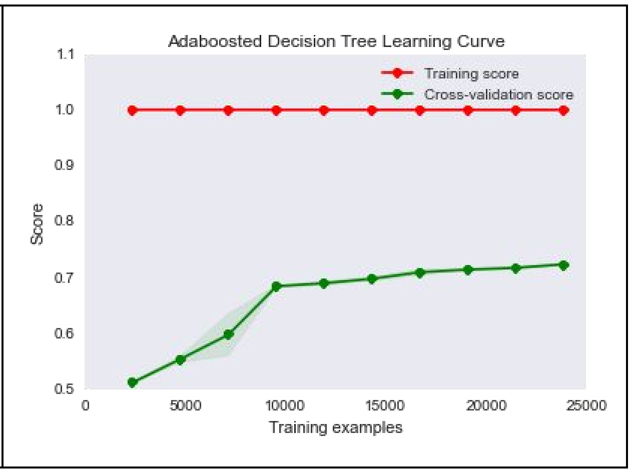


The cross validation slowly rose as the training inup increased and the training error stayed practically 1 across the range. The generation of this 5 fold CV learning curve took 4:15. The final testing accuracy was 0.64.

Boosting

The ada boosted decision tree took a bit longer to train and much time had been spendt on data munging, the boosted learner was run with default parameters using the best tree from the decision tree grid search as the base classifier. A 10 fold cross validation resulted with mean of CV score as .74. The learning curve performed steadily better with increasing training. Testing score was unavailable for this plot.



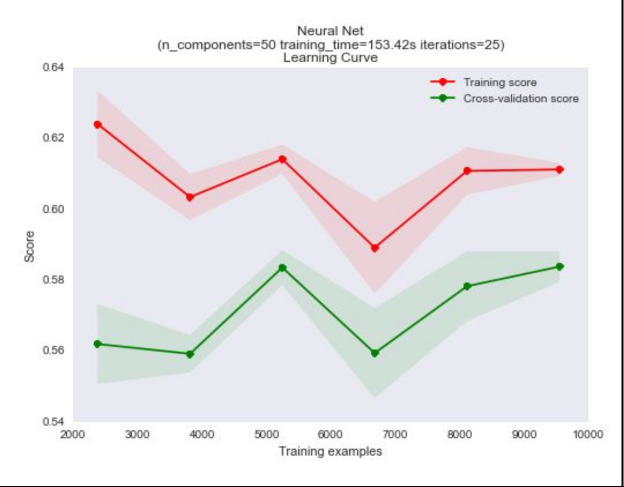


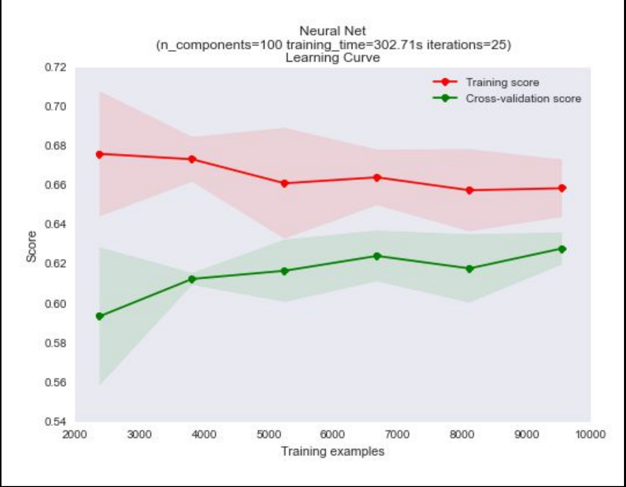
Neural Nets

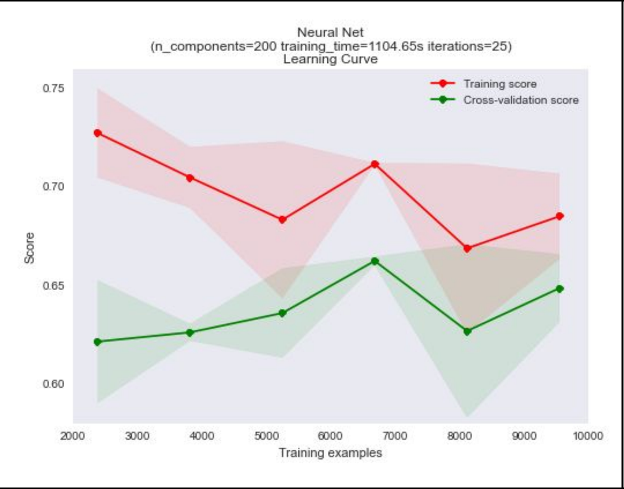
A neural net with 1 hidden layer was constructed for each SVD size and was fitted using a random cross validated search using the following parameters:

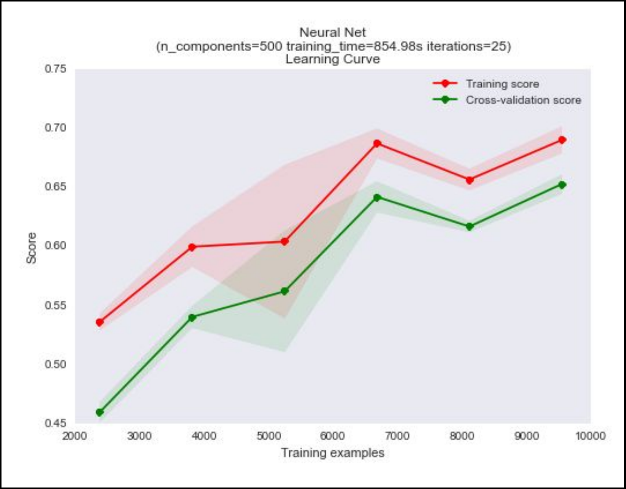
```
    'learning_rate': stats.uniform(0.001, 0.05),
    'hidden0__units': stats.randint(100, 2000),
    'hidden0__type': ["Rectifier"],
    'regularize':["", 'L2']}
```

The best estimators were chosen for generating leanging curves so that we could see the effect the number of components on the learning rate:









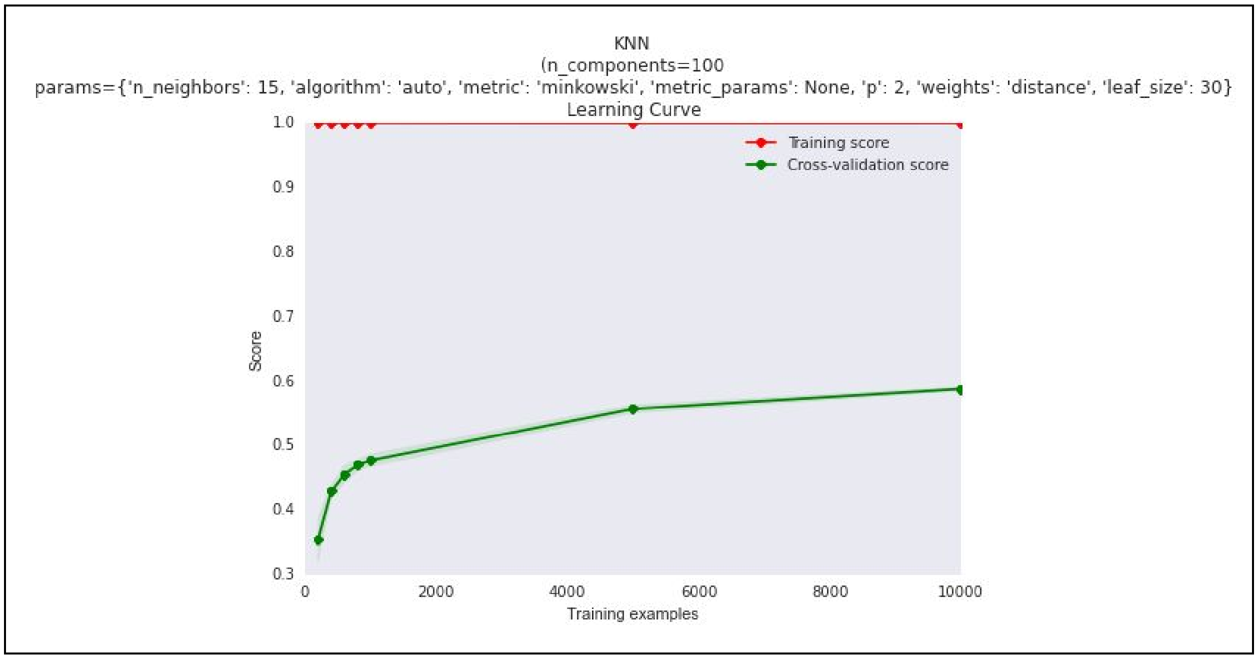
The NN with 300 components was selected to test for the test set and had an error of ~.68 (data from ipython notebook output)

K Nearest Neighbor

The following grid parameter set was used to find good parameters for KNN :

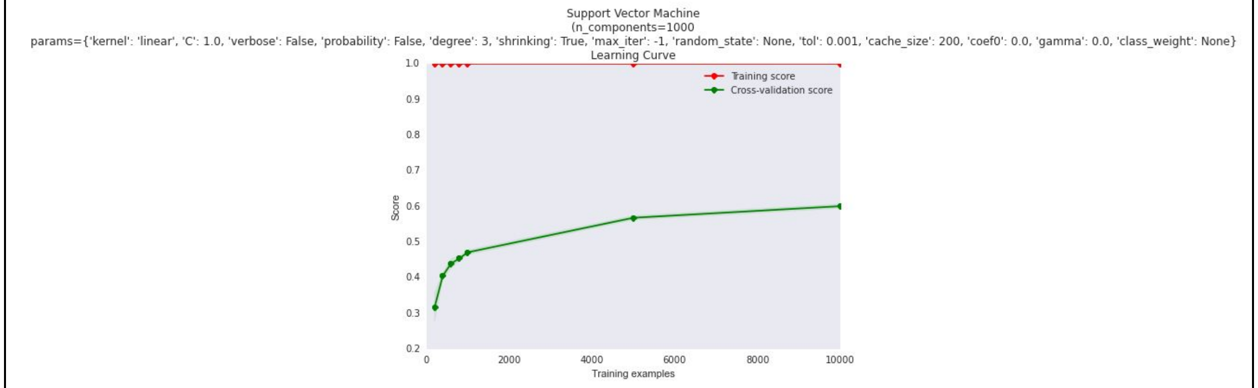
```
{'n_neighbors': [1,2,5, 10, 15, 20, 30],
   'weights': ['uniform', 'distance']}
```

for each number of components of the SVD we fitted parameters to the model the winning parameters were weights = distance and the n neighbors ranged from 10 to 20. There was very little difference on the performance of the models based on the number of components. All learning curves had a steep ramp up until about size 800 and then the accuracy gain rate decreased and didn't go above 0.6. The models weren't available at publication time (clobbered a pickle) to get the test data



Suipport Vector Machine

SVMs were fit to the collection of decomposed inputs using rfb, sigmoid, poly, and linear kernels. Uniformly the linear kernel performed best in these tests. The learning curve for the SMV looked much like the KNN curve with unimpressive CV scores.



Conclusions

A more serious preprocessing may be usefult for the high dimensional data set. Additionally better tuning of the neural net may give better predictions.