Supervised Learning - CS7641 Fall 2021

Huifu Tian

September 15, 2021

In this assignment, I will use the Polish companies bankruptcy data set and the MNIST handwritten digits data set to compare various supervised learning algorithms. All of the analysis was done using the scikit-learn library compatible with Python 3.

1 Polish Data

1.1 Data Overview

This dataset is a collection of Polish companies spanning 2007-2013. The dataset contains a total of roughly 700 bankrupt companies corresponding to 2400 financial reports in the period between 2007 and 2013. The data is separated into five datasets - 1stYear, 2ndYear, 3rdYear, 4thYear, 5thYear. Each row in every data set contains the financial report of one company in a single year.

- 1stYear: 7027 surveyed companies in 2007, 271 of which went bankrupt by the start of 2013
- 2ndYear: 10173 surveyed companies in 2008, 400 of which went bankrupt by the start of 2013
- 3rdYear: 10503 surveyed companies in 2009, 495 of which went bankrupt by the start of 2013
- 4thYear: 9792 surveyed companies in 2010, 515 of which went bankrupt by the start of 2013
- 5thYear: 5910 surveyed companies in 2011, 410 of which went bankrupt by the start of 2013

The attributes are shown in Appendix Table 1.

The output consists of one column where 1 means bankruptcy and 0 means non-bankruptcy.

add more stats exploring the data such as autocorrelations, class sizes, etc

1.2 Preprocessing

Minimal preprocessing was done to keep the focus on analysis. I combined all of the data sets to form a new dataset of over 43000 rows, which may introduce oversampling bias since it's not guaranteed that each company that is marked with a bankruptcy status=1 only shows up once across the 5 data sets. is it called oversampling bias? All of the infrequent NaNs in the attribute columns are replaced with 0's. (There are no NaNs in the output column.) I divide the data into 70% train and 30% test sets while applying stratification to ensure uniform distribution of classes between the two groups. Finally, I under sample the majority class to a 1:1 ratio when evaluation the performance of a model and when I apply StratifiedKFolds to create validation scores. This is done to address the issue of a 95:5 imbalanced dataset and that most of the algorithms evaluated in this study is biased towards fitting to the majority class samples.

1.3 Decision Tree

Baseline model

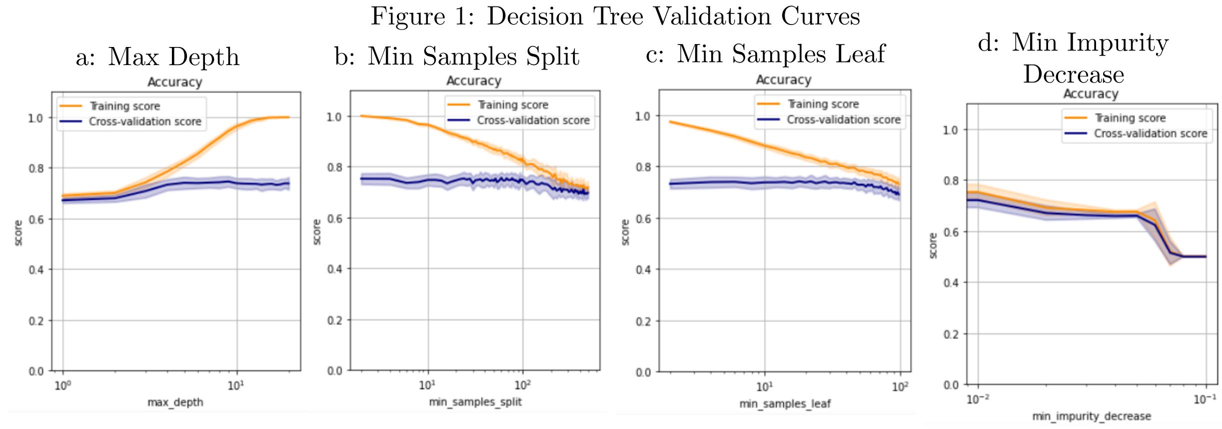
The baseline model uses sklearn's DecisionTreeClassifier class and is trained with unlimited tree depth and the Gini measure of impurity. Table 2 shows the mean metric scores from fitting the classifier on the training data and scoring on the test data. With undersampling, I mean the results across 20 iterations of random sampling from the majority class before doing the fitting.

Before tuning the model, it's worth noting that the decision tree classifier exhibits the general behavior one tends to see with imbalanced data sets, namely that the classifier prefers to minimize error for the majority class at the expense of the minority class. As the sample rebalancing approaches parity between both classes (shown by the progression from 3:1 to 1:1 ratio), precision, recall, and accuracy of the minority class are all improved. As a bonus, fitted tree depth also becomes smaller, which is preferred because we like simpler models over more complex models.

Table 1: Baseline Model Performance

	DT	NN	KNN	XGB	SVM	DT	NN	KNN	XGB	SVM
Undersampling?	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Accuracy	0.95	0.94	0.95	0.97	0.95	0.78	0.57	0.65	0.86	0.65
F1	0.48	0.01	0.02	0.63	0.00	0.78	0.57	0.65	0.87	0.63
Precision	0.46	0.11	0.10	0.89	0.50	0.78	0.58	0.65	0.86	0.68
Recall	0.50	0.03	0.01	0.49	0.00	0.79	0.58	0.65	0.88	0.59

Figure 1: Decision Tree Validation Curves



Model Tuning

Impurity

I first experiment with using "Gini" or "entropy" as the impurity measure. I find that there's no significant difference in any of the accuracy/f1/precision/recall scores using the undersampled data. Since the entropy measure takes longer to calculate due to logs in the measure and it doesn't offer significant improvements, I use "Gini" for the rest of the analysis.

Max Tree Depth

try fit to max depth and use pruning Without bounding max tree depth, the baseline model will fit the data to a tree with average depth of 19.65 using 1:1 rebalancing across 20 random samplings across the majority class. I also use a modified version of the Stratified K Folds validation algorithm. Rather than selecting k folds to split one set of the post-undersampled training data into a train and validation set, I select 2 folds across k sets of training data. The reason for this modification is that I'm undersampling the majority class. To reduce the variance of results across multiple trials as much as possible, hence k folds, it makes more sense to randomly pick from a much bigger pool of data than to undersample first then pick from a smaller pool of data. Figure 1a shows the accuracy score of modulating the max_depth from 1 to 20. The training curve is generally increasing while the cross validation curve increases and tapers off. The reason is increasing the max tree depth allows the model to fit the training data to increasing accuracy. With a tree of around 20 nodes deep, the model is able to almost perfectly fit the training data. For the cross validation score, the highest score is reached at max_depth=9 (see Table 2). At that point, the accuracy scores starts to decrease and overfitting occurs.

Min Samples Split

Min samples split is a pruning threshold that requires each node to have at least a certain number of samples before it's split again. I range this parameter from 2 to 500 while letting max tree depth to be unlimited. The performance of the model is reported in Table 2 col 2 and the validation curve is shown in Figure 1b. The training curve starts at 100% accuracy and gradually decreases. this is because at value of 1 corresponds to no pruning and the decision tree can perfectly predict the training set. The accuracy score on the validation set is flat up until 100 and slows drops down. The flatness of this curve from 2 to 100 indicates there is a collection of nodes that are split just with fewer than 100 samples. The further splitting of that collection of nodes offer no additional benefit to the model once that node has been established. What we can learn from analyzing this parameter is that our data is very noisy and a few high-level splits determine our model's predictive behavior.

Min Samples Leaf

The validation curves are shown in Figure 1c. The training and validation curves are both very similar to those of min samples split. We attempt to increase this parameter to reduce the variance and limit overfitting. But we learn that increasing this parameter only hurts the model's ability to generalize. We only start losing prediction power once we remove nodes that have 15+ samples, which means that all the nodes that are smaller than 15 samples are a coin flip and the model's architecture cannot handle further complexity.

Min Impurity Decrease

this section

Table 2: Decision Tree Parameter Tuning Results

Metric	Max Tree Depth	Min Samples Split	Min Samples Leaf	haha
Accuracy	0.79	0.77	0.81	0.79
F1	0.79	0.77	0.71	0.79
Precision	0.78	0.77	0.71	0.78
Recall	0.79	0.77	0.72	0.79
Optimal Value	9	<=100	<=15	19.75

Figure 2

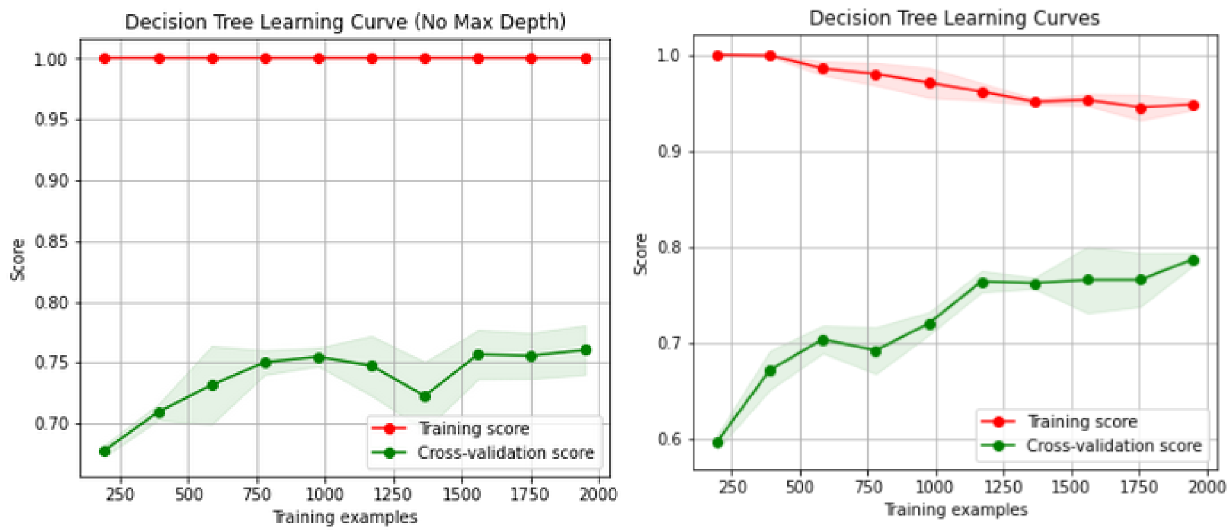
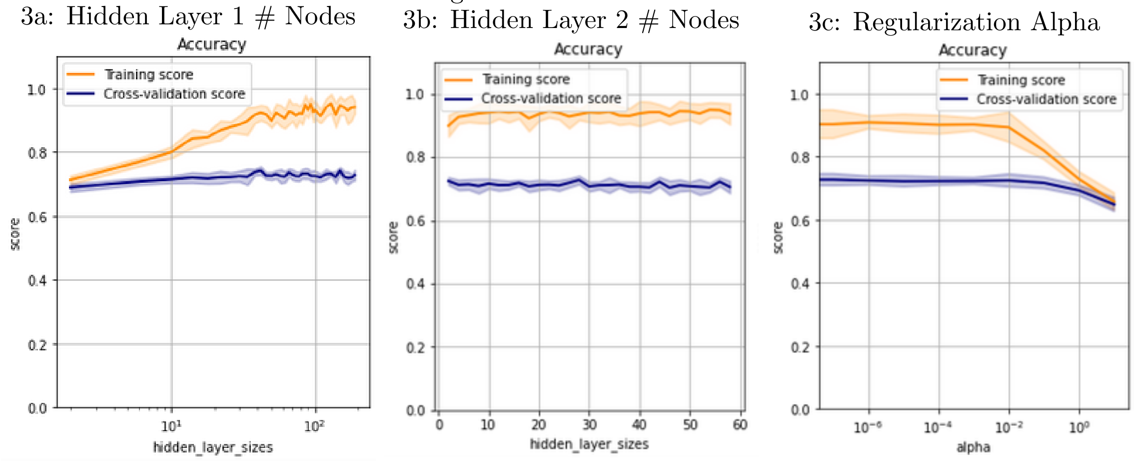


Figure 3: Neural Network

3a: Hidden Layer 1 # Nodes
3b: Hidden Layer 2 # Nodes
3c: Regularization Alpha



Learning Curves

To understand whether our model is overfitting or underfitting, I make two learning curves: one without restrictions, one with max tree depth. In the case without setting max_depth, the training accuracy scores stay at roughly 100%. This shows that the model was likely too complex and fit the training model perfectly. The idea that this model is overfitting the data is also supported by the fact that the validation scores remain flat, meaning that the model is unable to generalize. Once I introduce max_depth, the training model's accuracy scores start to decrease as we limit the model's complexity. The result is that a simpler model does better at generalization, which is shown by the much improved scores on the test data. There is still a gap between the training curve and the validation curve at the 2000th sample, which means that further limiting of the model's complexity may improve the validation score. Also, validation scores are still in the middle of improving at the 2000th sample, which implies that having more samples would improve the model's ability to fit the data. discuss variance and bias

Final model and performance

The model with max_depth set to 9 yields is run 20 times using 1:1 balanced sampling between the two classes, each time picking a 70% training and 30% testing set. The mean accuracy across the 20 iterations is 0.79. Show in final summary table

1.4 Neural Network

Baseline model

The baseline results with and without undersampling the majority class are shown in Table 1. The rest of the analysis will undersample the majority class. The parameters default to using a single hidden layer with 30 nodes and 1e-4 as regularization alpha. I also scale the features to have mean 0 and variance 1.

Model Tuning

Size of Hidden Layers

The training and cross validation accuracy scores by varying the number of nodes in a one-hidden layer net are shown in Figure 3.The accuracy curve on the train set starts low but increases as more complexity is enabled on the model. Also at first, the increasing complexity in the model benefits the validation scores. The question is what's the optimal number of nodes in the hidden layer that maximizes generalization accuracy? And the answer is 42. Any further increases in model complexity only marginally increases the model fit on the training data while gradually lowering the ability to generalize on the validation set. This is overfitting. One can also see from the training scores after n=42 start to display higher variance, which means that the

Table 3: Neural Network Parameters and Results

Parameter	Hidden Layer 1 Size	alpha
Accuracy	0.75	0.75
F1	0.75	0.74
Precision	0.75	0.75
Recall	0.76	0.74
Optimal Value	42	<=1e-2

model's accuracy becomes more dependent on particular idiosyncrasies in the weights and the data itself. check last sentence.

At this point, I can do a grid search to find the optimal number of nodes on a 2-hidden layer network. But before that, I will hold the number of nodes at 42 on the first layer and repeat the same procedure as before on the second layer. The results are shown in Figure 4. As you can see, the accuracy on the cross-validation curve is flat, which shows that the extra layer does not seem to add further benefits to the 1-layer model.

Regularization Alpha

Another parameter to vary is the L2 regularization alpha. This parameter is the multiplicative scalar term of a penalty term added to the cost function. The penalty term is the sum of the weights squared multiplied by alpha/2, which means that any model that minimizes this function will also need to make sure the weights are small as well. The larger the alpha, the greater the penalty and more we can limit potential overfitting. The results are presented in Figure 5 and Table 5 Column 6. The training and test curves in Figure 5 show that there's no difference in adjusting alpha between 0 and 1e-2. After 1e-2, higher alphas starts to limit the model's complexity and leads to underfitting. This is observed by the decreasing testing score metric. The various metric scores in column 6 show no difference using 1e-7 as the alpha value.

Learning Curves

1.5 KNN

Baseline Model

The baseline model results with and without undersampling are shown in Table 1. Like decision trees and neural nets, KNN is susceptible to showing partiality to the majority class. Also like neural nets, the features need to be normalized to mean 0 and variance 1. This is because the distance measure used to determine the set of k nearest neighbors will place more weight on features that have the larger scale. For example, the model cares a lot about features that are big because larger features result in larger distances. In reality, there is no reason aside from having domain knowledge to weigh one feature more than another.

Model Tuning

Number of Neighbors K

The validation curves on from varying the number of k nearest neighbors are shown in Figure 6. The training scores on all of the score metrics generally start high and decreases as the number of neighbors increase. This is expected because having fewer neighbors means that at least one of the fewer neighbors can vote in favor of itself. For example, in the case that k=1, picking a query sample q from the training set and and evaluating it against every data point means that the closest data point is itself and the distance is 0. In this case the accuracy on the entire training set will be 100%. Conversely, having k=1 means that the validation set will only have one opinion from one closest neighbor, leading to noise in accuracy and hence underfitting. Increasing k reduces the effect of noise from the k nearest neighbors and improves the accuracy. However, continuing to increase k to the size of the training set means that every vote will result in the majority class winning. The result is an accuracy score that's identical to the proportion of majority class in the dataset. And the model becomes useless. The validation curves for f1, precision, and recall show some instability of when k<10. This is the result of the way sklearn resolves ties among even numbers of neighbors. The algorithm returns the lowest class value which is always 0. The effect of this behavior has a particularly strong effect on recall, since the number of false negatives is usually high when votes are always broken to create additional false negatives. Column 5 shows the performance of the model while selecting the k that optimizes the validation score. In our case, the same max value was reached at k=11 and k=52, so I simply average these values and round down to the lower odd value. (I could round up as well)

Learning Curves

The learning curve for KNN is shown in Figure 7. Both curves are fairly flat with the exception of the smallest sample size. It's also possible to argue that the curves increase slightly before peaking around 800-1000 samples before tapering off again, though the difference is only from 68% to 69% for the training set. If this increase can be argued, the explaination would be that adding more samples only improves accuracy for both the training and testing sets only up to a certain point before too many data points are added. If the data is noisy, or if irrelevant features are used in the distance calculations, the algorithm would actually be incapable of learning from the dataset. In our case, it looks like adding more data would not help KNN learn better. We would probably need to explore the features and weigh them differently to address autocorrelations, irrelevant features, etc.

Figure 4:

4a: Validation curve of k 4b: Learning Curve, k=31 3c: Learning Curve, k=5

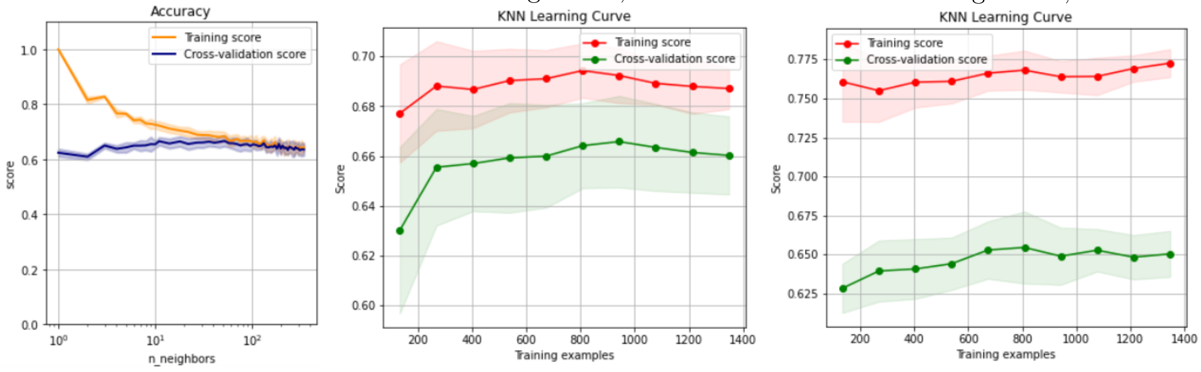


Table 4: KNN Parameters and Results

Parameter	1	2	3	4	5	6
n_neighbors	5	5	5	5	31	31
weights	uniform	uniform	uniform	uniform	uniform	distance
algorithm	auto	auto	auto	auto	auto	auto
p	2	2	2	2	2	2
Undersampled?	No	3:1	1:1	1:1	1:1	1:1
Normalized?	No	No	No	Yes	Yes	Yes
Results						
Accuracy	0.95	0.74	0.65	0.66	0.67	0.67
F1	0.02	0.38	0.65	0.66	0.66	0.65
Precision	0.10	0.47	0.65	0.66	0.69	0.69
Recall	0.01	0.31	0.65	0.65	0.63	0.63
Balanced accuracy	0.50	0.60	0.65	0.66	0.67	0.67

1.6 Boosting

Baseline Model

I fit the data using XGBoost, following the same process as the before on undersampling, improving f1, precision, recall, and balanced accuracy. alternative ways to handle this? The baseline results are shown in Table 1. The parameters used for the results are as follows: n_estimators=100, max_depth=10. No pruning was performed. Note that we don't have to normalize data because the decision tree weak classifier only needs absolute feature values to work.

Model Tuning

n estimators

I adjust the number of base estimators used (n_estimators) and plot accuracy scores on the training set as well as the validation set. The results are shown in Figure 8. Both training and validation scores start low and gradually increase as more estimators are added. Using one estimator essentially means trying to model the data with one decision tree. As more trees are added, the fit on the training and validation sets get better. The point at which 100% of the training data is predicted perfectly also corresponds to when we start to overfit on the validation data. The model performance is shown in Table 5 column 3.

Max Tree Depth

The validation curve for modulating the prepruning parameter max_depth is in figure 5b. The optimal value of 9 is the value of max_depth when the accuracy score on the cross-validation curve is at its highest point. In Figure 5b, accuracy scores increase for both training and validation sets but drops off for validation set after max_depth reaches 9. This is because we are initially giving our model the complexity that is needed to fit the validation and training sets better. Too much complexity occurs after max_depth=9 and validation scores decrease. The model performance is shown in table 5, and we can see that pre-pruning improves model performance slightly.

Min Samples Split/Leaf

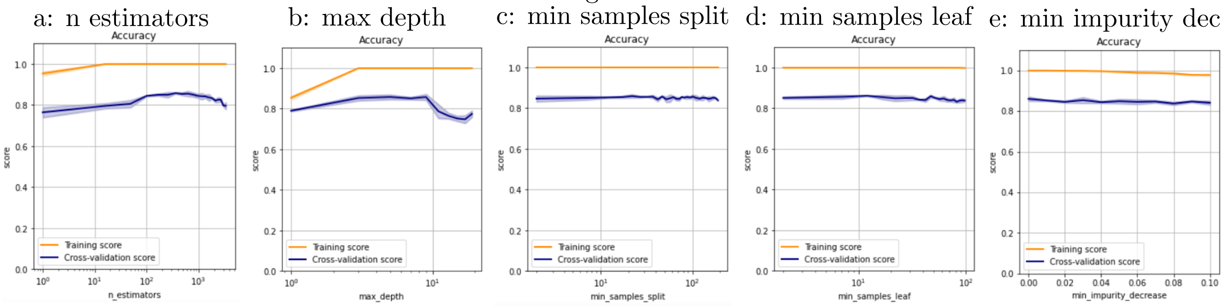
In Figure 5c, I modulate min_samples_split while letting the tree to grow without max depth. n_estimators is set to the optimal value from the previous step. Both accuracy curves are flat, which means that most of the model's predictive ability comes from the number of estimators, and not how complex any of the estimators are in terms of min_samples_split. Figure 5d shows the performance on the train/validation sets by pruning using the min_samples_leaf parameter and letting each tree grow fully. Both curves are flat, indicating no benefit to pruning using this parameter having determined the optimal number of estimators

Min Impurity Decrease

Figure 5e shows the train/validation curves from modulating min impurity decrease, explain this and expand section

Figure 5:

a: n estimators b: max depth c: min samples split d: min samples leaf e: min impurity dec



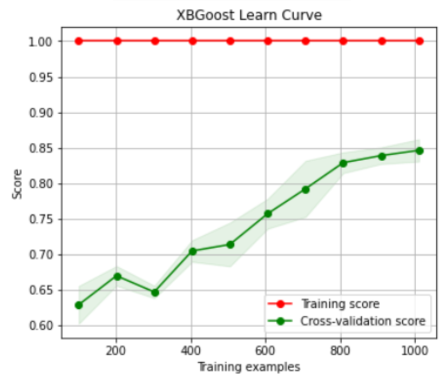


Table 5: XGBoost Results

	base model	n estimators	max depth	min samples split	min samples leaf	min impurity decrease
Accuracy	0.86	0.87	0.89	0.87		
F1	0.87	0.87	0.89	0.87		
Precision	0.86	0.88	0.89	0.87		
Recall	0.88	0.86	0.90	0.87		
Optimal value	None	625	9	20		

1.7 SVM

Baseline Model

The results from a baseline fit using rbf kernel with and without undersampling are shown in table 1.

Model Tuning

Kernel

The baseline results from experimenting with "linear", "polynomial", "rbf", and "sigmoid" kernels are shown in Table 6 and Figure 7. It can be seen that the the "linear" and "rbf" kernels seem to do the best in terms of accuracy and "polynomial" does poorly in f1 and recall.

Tuning RBF

C Regularization Parameter

2 MNIST Data

2.1 Data Overview

The MNIST data set contains 70,000 samples of 28x28 pixel handwritten digits from 0 to 9. Unlike the Polist bankruptcy data set, there are more than 10 times more number of features and 10 output classes instead of 2. The predictive knowledge in the MNIST data is more distributed among a larger set of features whereas the knowledge in the bankruptcy data is more concentrated among a smaller set of features. This can lead to the models having very different performance metrics between these two datasets.

Figure 6: Kernel

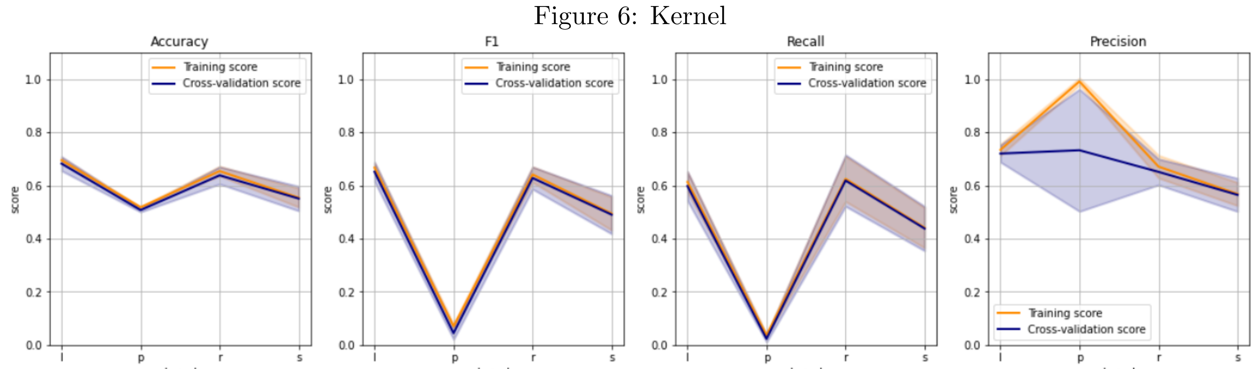


Table 6: SVM Results

	linear	poly	rbf	sigmoid
Accuracy	0.68	0.51	0.64	0.51
F1	0.66	0.05	0.64	0.41
Precision	0.72	0.63	0.64	0.51
Recall	0.60	0.03	0.66	0.35

Table 7: Baseline Model Performance

	DT	NN	KNN	XGB	SVM	DT	NN	KNN	XGB	SVM
One-hot encoding?	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Accuracy	0.81	0.90	0.94	0.91	0.96	0.81	0.87	0.94	N/A	N/A

2.2 Preprocessing

I load the MNIST data through Keras and load them into the standard train/test sets where 60,000 samples are in the train set and 10,000 samples are in the test set. To see the effectiveness of parameter tuning and relative predictive abilities of the various models, I only consider the first 10,000 samples from the training set, which will be further divided into 50% train and 50% validation sets used during parameter tuning. Finally, I one-hot encode the output vector which contains 10 possible values ranging from 0 to 9. This is usually done on categorical outputs but I include the results from both scenarios in table 7.

2.3 Decision Tree

Baseline Model

Tree Max Depth

The model performance and optimal value of tree max depth that maximizes the cross-validation accuracy score is shown in Table 8. The accuracy curve starts low when max tree depth is 1 since the complexity is too low. We have 10 output classes and predicting on one pixel value will not suffice. Both training and cross validation scores improve as we allow for deeper trees. The cross validation curve stabilizes and does not decrease after reaching max depth of 11, which means that the nodes after the 11th feature only nodes and leaves that contain higher levels of impurity and add no additional predictive power.

Min Samples Split and Min Samples Leaf

The validation curves are shown in Figure 8b and 8c. Both figures look similar in that the train curves are always decreasing and the validation curves are flat and then start to decrease. This means that our fit on the training data gets worse when we remove complexity. Also our model does not reach a point of overfitting by increasing the complexity. The extra nodes and leaves that are kept have a net 0 benefit/cost to the generalization error, although they help improve the training scores. This means that the decision tree fundamentally does not have enough complexity to model the true relationship between features and output.

Min Impurity Decrease

Figure 8d shows the validation curves using min impurity decrease. Both curves are mostly in sync as they start high at 0 and decrease as they head towards 0.1. Min impurity decrease is a fractional threshold that prevents a node from being split if the resulting decrease in impurity is less than this value. We see that letting the model fit with this value set to 0 results in the best performance. The reason is that our model was shown to be resistant to overfitting even when the tree is allowed to grow to its max depth, and any restrictions on its complexity has the effect of reducing its performance.

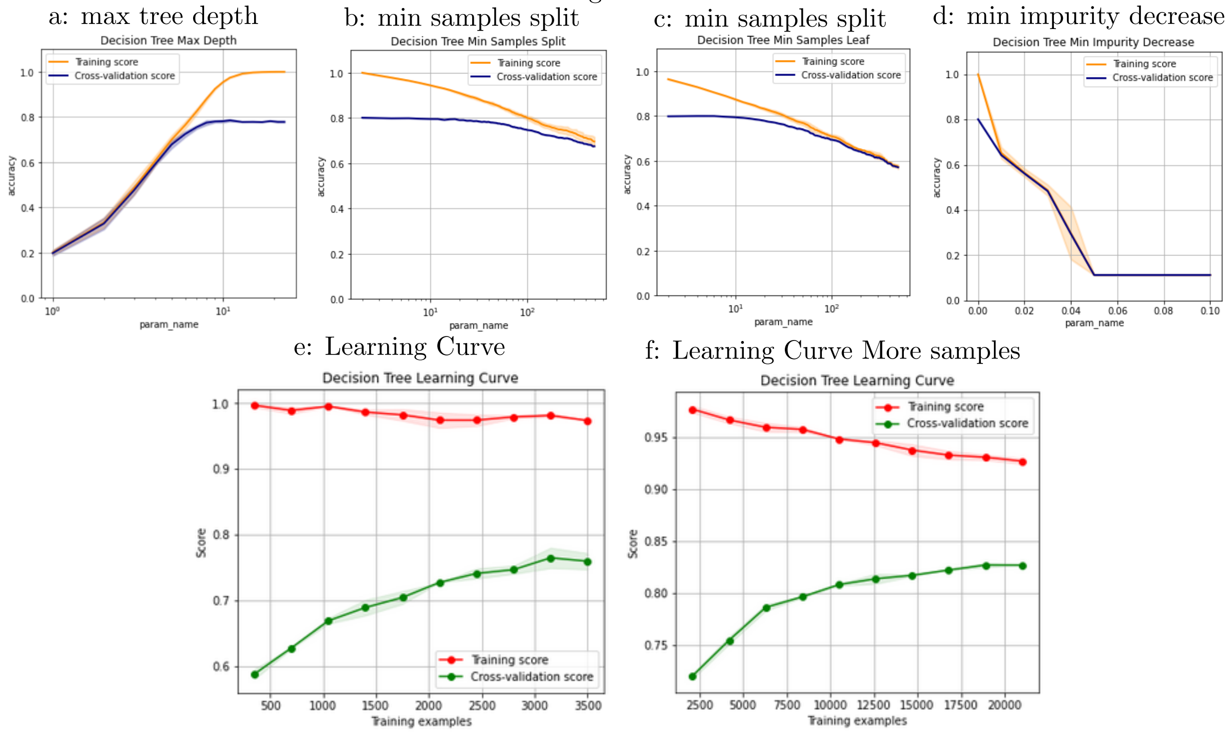
Learning Curves

The learning curves with max tree depth set at the optimal value for generalization are shown in Figure 8e. The training curve decreases gradually after starting at close to 100% accuracy with few samples because restricting the complexity of the model reduces the fit on a larger and larger training set. The opposite is the case where restricting the complexity prevents fitting to the noise on the validation set. Because the validation curve is still increasing by the time all 3500 samples are used, it's likely that increasing the number of samples will also increase the performance of the model. Figure 8f which includes 30000 training samples instead of 3500 shows how the model can improve with more samples.

Table 7: MNIST Decision Tree Results

	base model	max depth	min samples split	min samples leaf	min impurity decrease
Accuracy	0.81	0.81	0.81	0.81	0.81
Optimal value	None	11	2	4	0

Figure 7:



a: max tree depth b: min samples split c: min samples split d: min impurity decrease

e: Learning Curve f: Learning Curve More samples

Table 8: MNIST Neural Network Results

	base model	1 hidden layer	2 hidden layers	alpha
Accuracy	0.87	0.81	0.81	0.81
Optimal	(30,)	(400,)	(400, ?)	?

2.4 Neural Network

Baseline Model

Hidden Layer Size

The validation curves while modulating the number of nodes in one hidden layer from 0 to 1000 is shown in Figure 9a. One can see that the model needs a minimum of 30 nodes before reaching an accuracy of 80%. Both training set scores and validation set scores move in tandem. They start off very low between 10 nodes and 20 nodes because that's roughly equivalent to having no hidden layer as there are 10 output nodes. As the number of nodes increase, the validation scores increase gradually after first shooting up before settling at a flat region around 400-1000 nodes. Setting the number of nodes to 400 in the first layer, the results by modulating the number of nodes in the second hidden layer from 10 to 500 is shown in Figure 8b. The number of nodes shoots up when at 30 nodes and fluctuates between 30 and 100 and reaches a local maximum at 200 nodes. After 200, the accuracy score tapers a bit before increasing slightly again at 400. Between 30 and 450, the optimal number of nodes is approximately 200, which is when the the cross validation scores no longer show signs of instability and the model performs similarly to a more complex model with double the number of nodes.

2.5 KNN

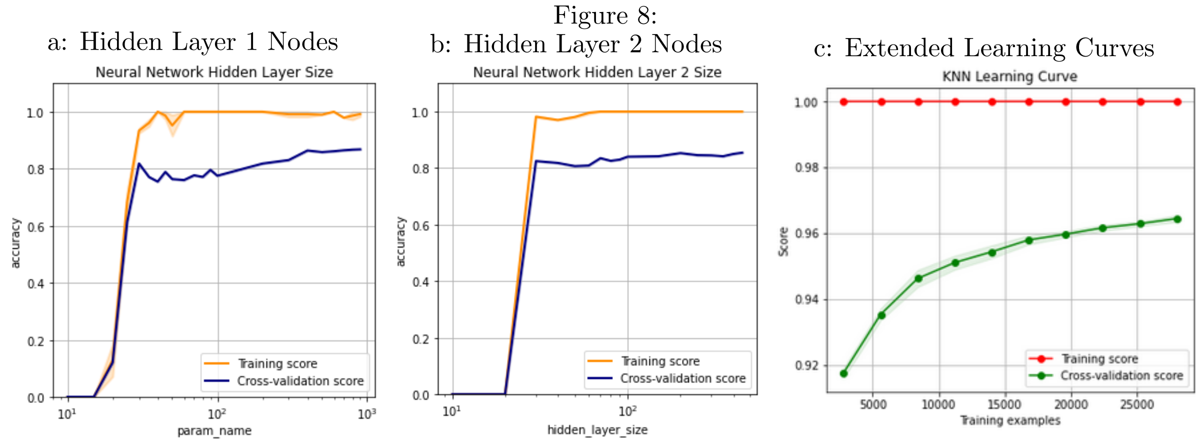
Baseline Model

Model Tuning

k

The validation curves are shown in Figure 9. We see that KNN prefers to have as few neighbors as possible when generating the prediction, i.e. k=1. This means that adding more neighbors and letting the vote

Figure 8:



a: Hidden Layer 1 Nodes b: Hidden Layer 2 Nodes c: Extended Learning Curves

Figure 9:

a: k   b: Learning Curves   c: Extended Learning Curves

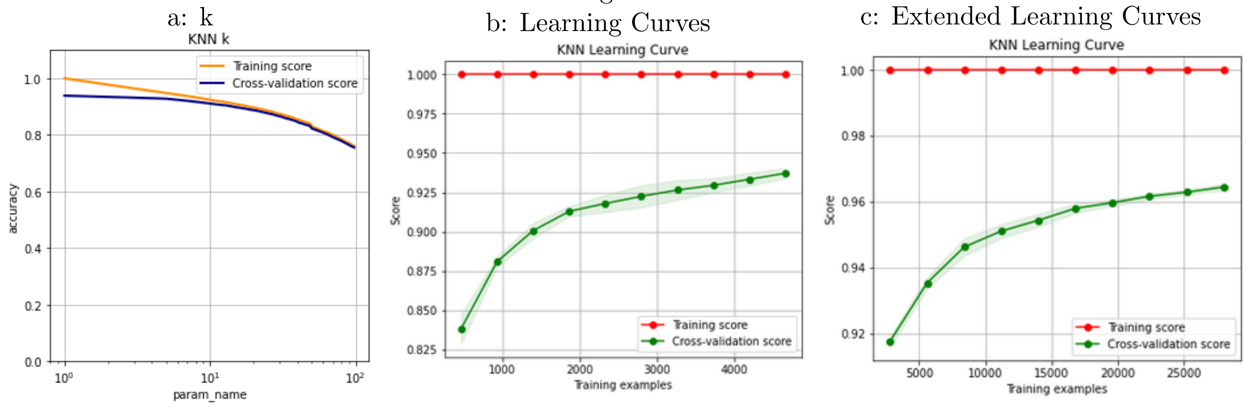


Table 9: KNN Results not done

	base model	1 hidden layer	2 hidden layers	alpha
Accuracy	0.87	0.81	0.81	0.81
Optimal	(30,)	(400,)	(400, ?)	?

determine the prediction adds rather than removes noise to the prediction. The errors as a function of k for MNIST is different from those for the Polish bankruptcy data in that the errors are minimized at a much higher value of k in the Polish data. Also, ranging the number of samples in the training set (from 500 to 10000)does not change the monotonically decreasing nature of the validation curve. This suggests that the position of the digits on the 28x28 grid and differences in handwriting play a large role. For example, the Euclidean distance measure assigns a high value of distance to two samples of the digit “1” if the pixels do not overlap, even if they look identical otherwise. Also, it’s possible that the same letter 4 can be written very differently but a “9” might have a lower distance measure. As we sample more digits - increase k, we start introducing noise in our prediction. In other words, you are better off finding another digit that has the same handwriting and same translational position on the grid than by finding the closest k neighbors. On the other hand, the features in the Polish data are meaningful. It’s impossible for two companies to be in the same output class if you simply shift all the values in the features to the right by 4 positions. As a result, there is much less noise by adding neighbors and locking out samples in the same class that are translated.

Learning Curves

At k=1, the training set’s learning curve is at 100% because each query point in the training set can be found in memory with distance=0. The learning curve starts low and increases monotonically. As more samples are memoized, the KNN’s ability to predict given an unseen data point increases. Since the curve keeps increasing near the end, we can benefit from more adding more data points in the training set. Figure 9c shows the learning curves after you add up to 30000 samples to the training set. As expected, the cross validation learning curves extends its increase past the previous 0.935 level at 5000 samples.

2.6 XGBoost

Baseline Model

Model Tuning

n estimators

The results of

Table 10: XGBoost Results not done

	base model	1 hidden layer	2 hidden layers	alpha
Accuracy	0.87	0.81	0.81	0.81
Optimal	(30,)	(400,)	(400, ?)	?

Figure 10 XGBoost figures - todo:

a: k　b: Learning Curves　c: Extended Learning Curves

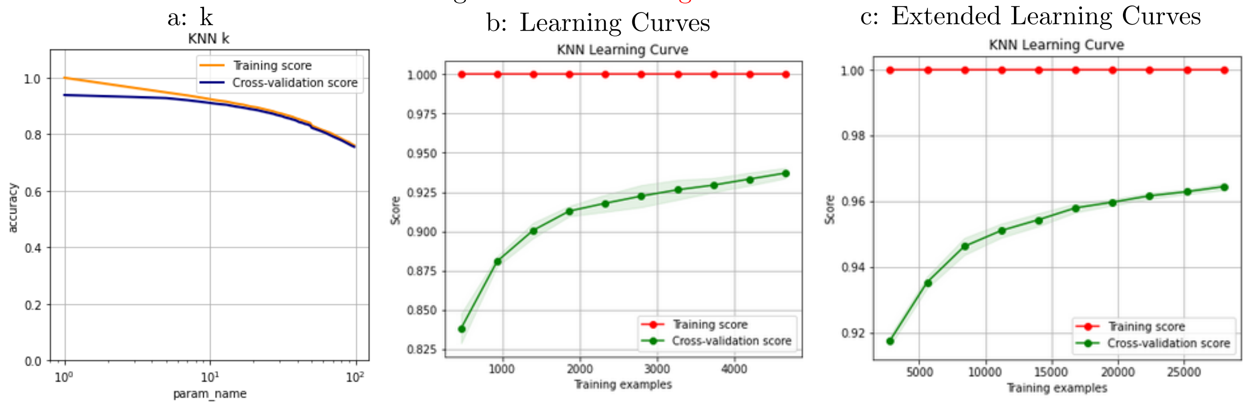


Table 11: SVM Results

	linear	poly	rbf(Gaussian)	sigmoid
Accuracy	0.68	0.51	0.64	0.51

max depth

min samples split / min samples leaf

2.7 SVM

Baseline Model

Model Tuning

Kernel

Appendix Table 1: Polish companies bankruptcy data attributes

X1 net profit / total assets	X33 operating expenses / short-term liabilities
X2 total liabilities / total assets	X34 operating expenses / total liabilities
X3 working capital / total assets	X35 profit on sales / total assets
X4 current assets / short-term liabilities	X36 total sales / total assets
X5 [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365	X37 (current assets - inventories) / long-term liabilities
X6 retained earnings / total assets	X38 constant capital / total assets
X7 EBIT / total assets	X39 profit on sales / sales
X8 book value of equity / total liabilities	X40 (current assets - inventory - receivables) / short-term liabilities
X9 sales / total assets	X41 total liabilities / ((profit on operating activities + depreciation) × (12/365))
X10 equity / total assets	X42 profit on operating activities / sales
X11 (gross profit + extraordinary items + financial expenses) / total assets	X43 rotation receivables + inventory turnover in days
X12 gross profit / short-term liabilities	X44 (receivables * 365) / sales
X13 (gross profit + depreciation) / sales	X45 net profit / inventory
X14 (gross profit + interest) / total assets	X46 (current assets - inventory) / short-term liabilities
X15 (total liabilities * 365) / (gross profit + depreciation)	X47 (inventory * 365) / cost of products sold
X16 (gross profit + depreciation) / total liabilities	X48 EBITDA / total assets
X17 total assets / total liabilities	X49 EBITDA / sales
X18 gross profit / total assets	X50 current assets / total liabilities
X19 gross profit / sales	X51 short-term liabilities / total assets
X20 (inventory * 365) / sales X21 sales (n) / sales (n-1)	X52 (short-term liabilities * 365) / cost of products sold) X53 equity / fixed assets
X22 profit on operating activities / total assets	X54 constant capital / fixed assets
X23 net profit / sales	X55 working capital
X24 gross profit (in 3 years) / total assets	X56 (sales - cost of products sold) / sales
X25 (equity - share capital) / total assets	X57 (current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X26 (net profit + depreciation) / total liabilities	X58 total costs /total sales
X27 profit on operating activities / financial expenses	X59 long-term liabilities / equity
X28 working capital / fixed assets	X60 sales / inventory
X29 logarithm of total assets	X61 sales / receivables
X30 (total liabilities - cash) / sales	X62 (short-term liabilities *365) / sales
X31 (gross profit + interest) / sales	X63 sales / short-term liabilities
X32 (current liabilities * 365) / cost of products sold	X64 sales / fixed assets

Figure 11: SVM

a: k　b: Learning Curves　c: Extended Learning Curves