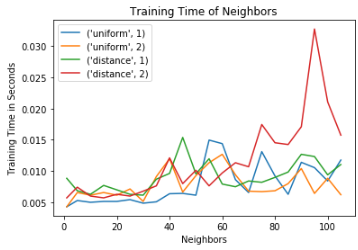
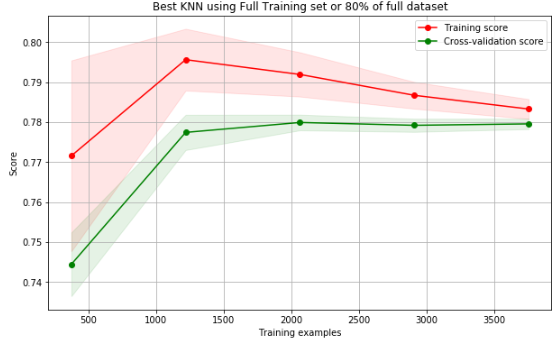
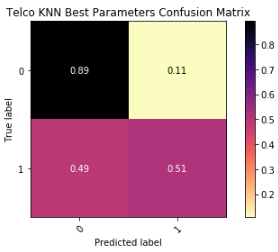
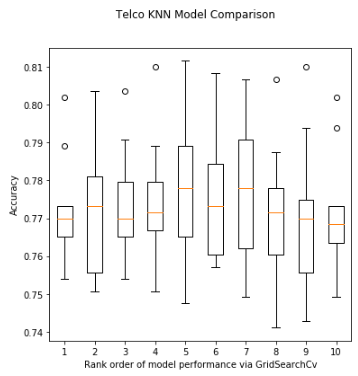
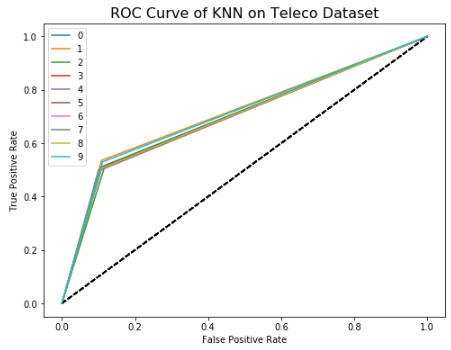
1. Telco Customer Churn dataset – This is a dataset from a telecommunications company of customer who have churned and not churned based upon who they are and the type of service they receive. We find this interesting because customer churn is a fascinating topic. What is it about a customer that causes them to churn? The ability to answer this question and how to answer it is readily applicable in today's job market. If we can find a way to prevent customer churn and show it to our managers, it will not only make us look good but keep us employed. If we can find the attributes that help signal who is about to churn, then we can provide the customer different offers to retain them.

Exploratory - this is a repeat of what has been done in the past but is still in our code for reference. Not commenting on it due to length of report.

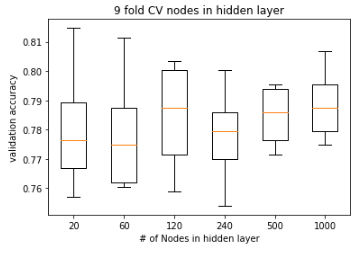
Scaling - We scaled everything between 0 and 1 and split our dataset into train/valid/test this time. We wanted to have a better approximation of how our model will perform on out of sample data. If we continually validate our trained model on our test set, the model starts to become optimistically biased. In order to prevent that we train the model on the train set, see how it performs on the validation set and keep iterating. Once we have selected our model, we will see how well our trained model performs on the test set. We applied PCA to account for correlated variables such as total charges.

# Telco dataset using K Nearest Neighbors

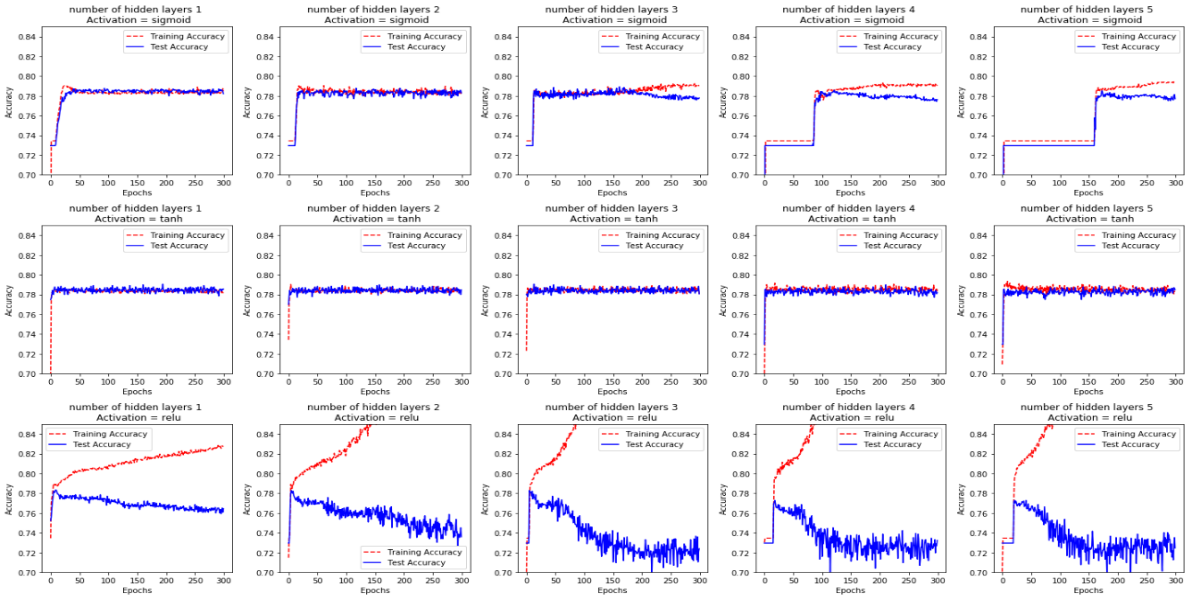
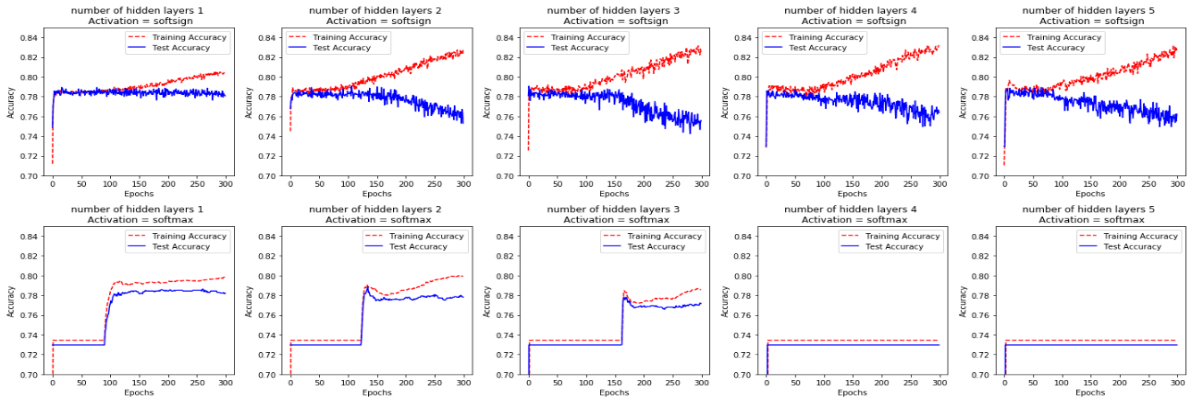
1. Cross validation Grid search – We use 9-fold CV grid search to find the optimal parameters. Optimal is defined as having the highest accuracy with some respect to computation time. We looked at 1) different distance metrics, Euclidean and Manhattan 2) different neighbors and 3) different distance weights both uniform and distance. The model we ended up choosing had the following parameters, weights = uniform, p = 1 and neighbors = 95. This was a top performing model in regard to train/test accuracy and fit computation time.
2.  Experiments
   1. Accuracy with respect to neighbors, distance and our p value
      1. A weight value of 1, i.e., non-weighted has the best performance. This tells us that our classification points tended to be further away from each other. We know this by the uniform distance performing better across all neighbor’s vs distance. As distance increases its weight exponentially decreases. Once, we hit roughly 80 neighbors our accuracy peaks and begins to level out.
   2. Computation Time vs neighbors fit– Overall this is so fit is so fast (< 1 sec) on each respective search, it doesn’t matter how many neighbors we use based upon our limited search space.
3. Learning Curves – Does our model suffer from high variance/high bias
   1. Using 64% of the data to train (train/valid) our best model we conclude that it doesn’t suffer from high bias/high variance as the training and cross validation scores are very similar.
   2. Using the full training set, as we provide more and more training examples, do we suffer from high bias/variance? Conclusion, no we do not suffer from high bias/variance. This isn’t surprising as using less data still showed low bias/variance so giving it more data should yield similar results.
4. Model Comparisons
   1. Is there a difference in our top 10 models - KNN Model Comparison – Our conclusion is the top 10 models are very similar. When we look at the ROC Curve for our top 10 models perform, they perform like each other as the lines are nearly on top of each other. When we calculated the AUC score for these models the range was .817 - .820 on our testing set. This helps us feel comfortable that they are similar.
   2. We applied our testing data set to the top 10 fitted KNN models. We then ran 9-fold cross validation across these 10 models and compared them. We found the IQR of the top 10 KNN models overlaps each other. This tells us that statistically speaking, the models are similar. After concluding the top 10 scores are similar, we choose model #4 in chart 4b as the CV scores are like the other top performing models. What made us choose this, it had the highest uniform weight model accuracy.
   3. We know from chart 2a uniform models tend to perform better as our data points have quite a bit of distance between them. We found that we identified 78% of all classes correct. We had a 49/51 split on our labels we predicted as 1. Overall the model is accurate.



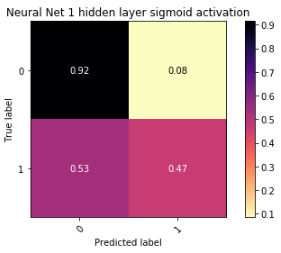
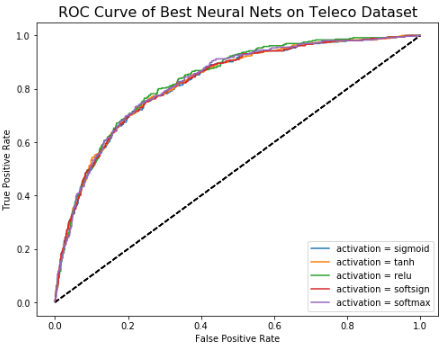
# Telco dataset using Neural Nets

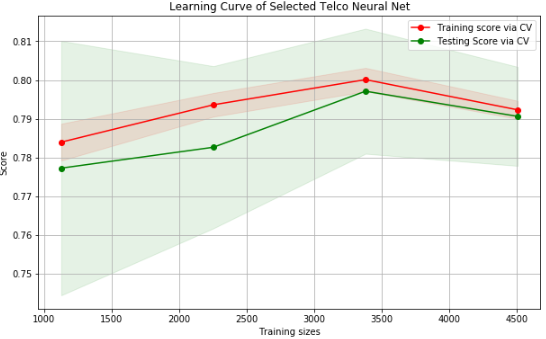


1. Cross validation Grid search – We are testing accuracy across number of hidden layers, activations and epochs. We first looked at how many nodes in the first hidden layer is best. We choose 120 as this gave us the best accuracy vs the other selections we had. We are only looking at the first hidden layer because in each preceding hidden layer we will be decreasing the nodes in the next layer by 1/2. We are choosing this approach because the professor has stated this is the approach he normally takes and sees good results with it, therefore, let’s test it out and see the results that we get.
2. Findings- Looking across all hidden layers and activations they all achieve similar accuracy scores. Across all different forms of activation functions adding more layers doesn’t improve accuracy. Meaning the addition of another hidden layer does not incrementally increase accuracy.
   1. Sigmoid function findings - The sigmoid function doesn’t appear to overfit the data in layers 1& 2 but 3& 4 does overfit the data as training and validation curves diverge around epoch 150. It achieves peak accuracy between 25-150 epochs. As hidden layers increase on the sigmoid activation function the number of epochs for it to reach peak accuracy increases.
   2. TanH function appears to output similar accuracy scores to sigmoid function across all layers. This appears to give us consistent results no matter how many layers we add to it. It seems very stable.
   3. Relu function arrives at its optimal accuracy quickly, 10-50 epoch across all layers, but wow does it overfit the data after those epochs. The training accuracy and test accuracy diverge very quickly.
   4. Softsign function like relu displays overfitting with our data, as the train and validation accuracy diverge from each other. Although the divergence takes longer (epoch wise) than relu to overfit.
   5. Softmax function really shows that as you increase layers one needs to increase the number of epochs. With the first two hidden layers we achieve peak accuracy at 100- 150 epochs and layers 4 & 5 need more than 300 epochs to achieve optimal performance. What’s interesting, we can see how neural nets find local minima and not global minima. At 2-3 hidden layers we see this downward trajectory in our accuracy, this is the result of finding local minima.
3. Model Comparisons



* 1. Looking at all the Neural net activation functions we initially choose one with 1 hidden layer, 120 hidden nodes. This is because there’s no benefit to adding more layers. Our accuracy doesn’t increase to warrant the addition of more layers across any activation function.
  2. We are choosing sigmoid as it performs like the other activation functions with the same hidden layers. Also, we have gone over this specific activation function in class. Another insight which gives us comfort in choosing sigmoid, if we look at the ROC curves across our activation functions, we see all of them put out a nearly identical AUC ROC score. The scores range between .826 - .8313 which again confirms they are similar. Since there’s almost no difference we pick an activation function we are comfortable with.
  3. Looking at the confusion matrix reveals similar findings to our KNN. We found that we identified 79% of all classes correct. We had a 53/47 split on our labels we predicted as 1. Overall the model is accurate.
  4. Looking at the learning curve of our selected model we can say our model has low bias/variance. This is seen in how both training and validation accuracy converge towards each other.





# Telco Model Comparison

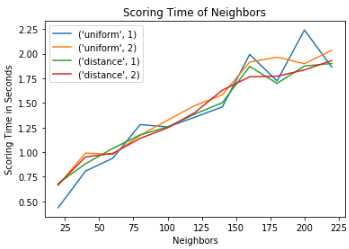
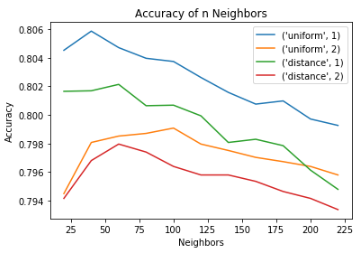
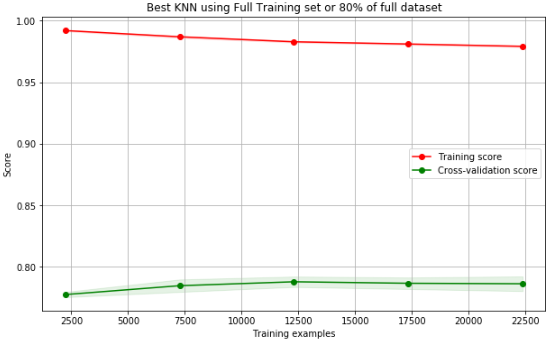
1. Now we compare all the models we’ve ran on this dataset across homework 2 and 3. We notice that SVM linear & SVM RBF both appear to be the best model. Their mean test accuracy’s fall above the upper part of the other models IQR. This gives us further comfort in choosing one of these two models. Between the two, we will choose SVM linear vs SVM RBF because while the results are similar SVM-linear takes less time to fit than SVM-RBF. SVM-Poly is giving is inferior results vs the other models, it goes to show **not** all algorithms will perform well on your dataset you have to experiment across different algorithms.

# Facebook dataset using K Nearest Neighbors

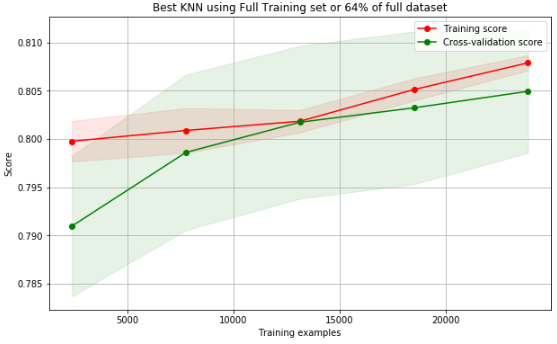
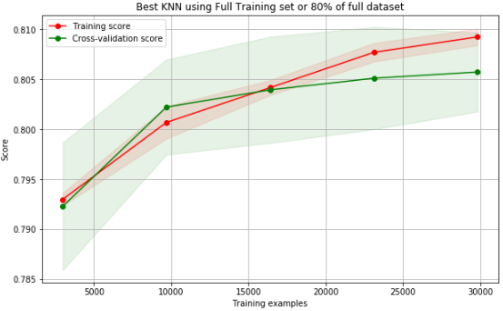
Exploratory - this is a repeat of what has been done in the past but is still in our code for reference. Not commenting on it due to length of report.

Scaling - We scaled everything between 0 and 1 and split our dataset into train/valid/test this time. We wanted to have a better approximation of how our model will perform on out of sample data. If we continually validate our trained model on our test set, the model starts to become optimistically biased. In order to prevent this, we train the model on the train set, then see how it performs on the validation set and keep iterating. Once we have selected our model, we will see how well our trained model performs on the test set. We applied PCA to account for correlated variables, as well.

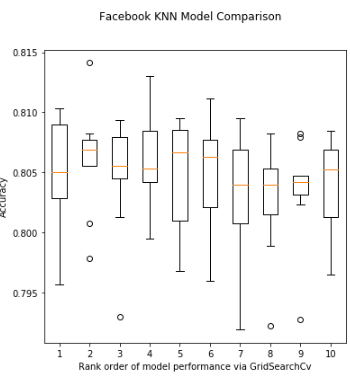
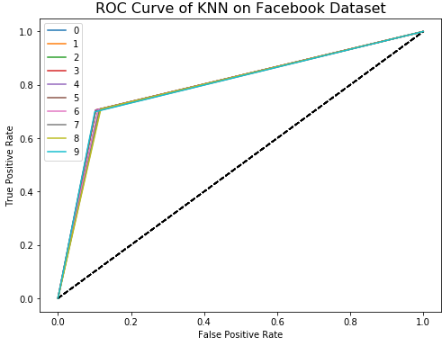
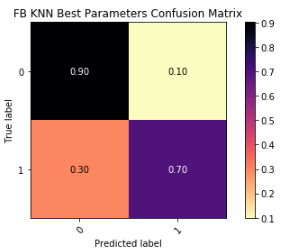
1. Cross validation Grid search – We used 5-fold CV grid search to find the optimal parameters. Optimal is defined as having the highest accuracy with some respect to computation time. We looked of 1) different distance metrics, Euclidean and Manhattan 2) different neighbors and 3) different distance weights both uniform and distance. The model ended up with the following parameters, weights = uniform, p = 1 and neighbors = 80. This was a top performing model with regards to train/valid/test accuracy and computation time. In order to compare across models, we kept the same philosophy as we did with the previous two assignments, unfortunately, we could not find parameters which didn’t suffer from high variance as shown in plot 1 below. We would need way more data than we have to bring the training and CV score together. I believe this is a result of the curse of dimensionality. For the remainder of this assignment we are using the highest 5 correlated values + two polynomial values which models in some more flexibility with our target value and then scaling them.



1. Experiments
   1. Accuracy with respect to neighbors, distance and our p value
      1. A weight value of 1, i.e., non-squared has the best performance. This tells us that our classification points tended to be further away from each other. We know this by the uniform distance performing better across all neighbor’s vs distance. As distance increases its weight exponentially decreases by the distance it is from a point. Once, we hit roughly 80 neighbors our accuracy begins to decline across all metrics.
   2. Computation scoring time vs neighbors fit– Overall the scoring is fast (< 3 sec) given our search space. We should expect that as neighbors increase scoring time should increase and it does. When choosing a model with scoring time in mind, any model will be sufficient within our limited search space as scoring time is fast.
2. Learning Curves – Does our model suffer from high variance/high bias
   1. Here we are looking at the final KNN model we choose (neighbors = 80, p=1, weights=uniform). Using the train/valid approach laid out earlier in this report, our chosen model from grid search shows that we don’t suffer from high bias/high variance. We know this because the training and cross validation scores are very similar and draw near to each other as sample size increases.
   2. Using 80% of our data, i.e. all our training data, do we see same results. Yes, we see similar results. our chosen model from grid search shows that we don’t suffer from high bias/high variance. We know this because the training and cross validation scores are very similar and draw near to each other as sample size increases.

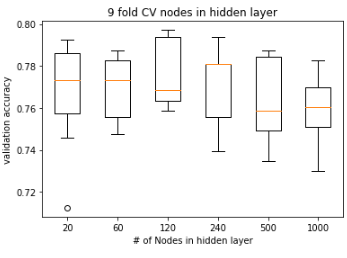


1. Model Comparisons
   1. Is there a difference in our top 10 KNN models?– Our conclusion is the top 10 models are very very similar. When we look at the ROC Curve for our top 10 performing models via grid search, they perform similar to each other. Since the ROC curve lines are on top of each other we then looked at the AUC score. The AUC scores range from.876 - .881 on our testing set, we conclude these models are similar.
   2. We applied our testing data set to the top 10 fitted KNN models. We then ran 9-fold cross validation across these 10 models and compared them. We found the IQR of the top 10 KNN models, for the most part overlaps each other, this gives us confidence that they are all similar.
   3. After concluding the top 10 scores are similar, we looked further at the chart and notice model #4 really stood out to us. What stood out to us was, it was one of the few models which didn’t have outlier scores and the upper whisker of the plot is higher than the others. The benefit of not having any outlier points, it tells us we can expect consistent results from the model.
   4. How does our model perform on the test set or how good at we generalizing – We looked at the confusion matrix to tell us this. Overall, our model performs similar on our test set as it did on our training set. This tells us we have a consistent model and we are finding a model which isn’t suffering from variance/bias. We have an overally accuracy of 80% on our test set, which is similar to our results we saw on our training/validation set. We are also identify 70% of the 1 class labels which is very good.



# Facebook dataset using Neural Nets

1. Cross validation Grid search – Our goal is to maximize accuracy across number of hidden layers, nodes, activations and epochs. We first looked at how many nodes in the first hidden layer would be best. We choose 120 as this gave us one of the best accuracy vs the other selections we had. What intrigues us about 120 nodes in the first hidden layer, it has the highest right tail or the highest upper bound of the CV distribution. We are only looking at the first hidden layer because in each preceding hidden layer we will be decreasing the nodes in the next layer by 1/2. We are choosing this approach because the professor has stated this is the approach he normally takes and sees good results with it, therefore, let’s test it out and see the results that we get.



1. Findings/Learning Curves - Looking across all hidden layers and activations they all achieve similar accuracy scores. Softmax with 4 and 5 hidden layers doesn’t perform well at all, as its accuracy is below our minimum window of 70%.
   1. Sigmoid function findings - The sigmoid function doesn’t appear to overfit the data in any of the layers. There is a steady and expected deviation between training and test accuracy. It achieves its optimal accuracy (elbow in the accuracy curve) around 50 epochs across all layers. Going from layer a hidden layer of 1 to 2 sees a small increase in accuracy (79% to 80.5%) but we don’t see the same increase beyond 2 hidden layers.
   2. TanH function appears to output similar accuracy to sigmoid function across all layers. It doesn’t appear to overfit the data in any of the layers as we see a steady and expected deviation between training and test accuracy. It achieves its optimal accuracy (elbow in the accuracy curve) between 25- 50 epochs across all layers. Going from layer a hidden layer of 1 to 2 sees a small increase in accuracy (79% to 80%) but we don’t see that same increase beyond 2 hidden layers. It appears to be a very stable activation function.
   3. Relu function appears to output similar accuracy to sigmoid function across all layers. This doesn’t overfit the data as it did with the Telco dataset which shows you these algorithms aren’t one size fits all. They need to be experimented with. We have a steady and expected deviation between training and test accuracy which indicates that it’s not overfitting the data. It achieves its optimal accuracy (elbow in the accuracy curve) very quickly around 10 epochs. Going from layer a hidden layer of 1 to 2 we do see a small increase in accuracy(80% to 81%) but we don’t really see a similar type of increase beyond 2 hidden layers. It appears to be a very stable activation function.
   4. Softsign function performs very similar to relu. It achieves its optimal accuracy (elbow in the accuracy curve) very quickly, around 10-40 epochs across all hidden layers. There is a very minimal increase from hidden lay 1🡪2🡪 3🡪4🡪5. We do have to weigh the time cost of a neural net with this many hidden layers. No signs of overfitting as we see an expected amount of deviation between training and test accuracy.
   5. Softmax function appears to be the least optimal choice of activation functions here. There is a small increase in accuracy from hidden layer 1 to 2 but no increase in accuracy from 2 to 3. Layers 4 and 5 are not appearing on our zoomed in chart because their accuracies are below our minimum threshold of 70%, therefore, 4 & 5 hidden layers on this activation function is not optimal. It achieves its optimal accuracy (elbow in the accuracy curve) slower than the other functions, around 75-100 epochs across the first 3 hidden layers. Compared to the other activation functions we have; this does not appear to be optimal.
2. Model Comparisons
   1. Looking at all the neural net activation functions we see in increase in accuracy from 1 hidden layer to 2. After that, there’s little if any benefit of adding an additional layer across all activation functions. Our goal here, given 2 hidden layers which activation function gives us the best 9-fold cross validation score. Optimal is defined as having the highest accuracy with some respect to computation time. Both TanH and Relu give us similar cross validation test accuracy. Relu has a much tighter distribution meaning that it is more consistent. Another way of saying this because the distribution of our cross-validation scores are so tight our model has low bias/low variances.
   2. When we look at the ROC curve and the AUC score, we see that given 2 hidden layers all of our activation functions give us very similar AUC scores. This means they are all virtually the same. The AUC score ranges from .8821 🡪 .8887. The Relu gives us the highest AUC score at .8887. Given that Relu has the highest AUC score, tightest 9 fold CV scores and highest mean from 9 fold CV, we will choose this as our optimal model and proceed with looking at the confusion matrix.
   3. Looking at confusion matrix reveals similar findings to our KNN. We found that we identified 81% of all classes correct. We had a 29/71 split on our labels we predicted as 1. Overall the model is accurate and would consider it optimal
   4. Learning Curve of our choosen neural net model – we need to ensure this model does have low bias/variance. We conclude that it does given how the training and testing accuracy see little deviation between them
   5. Final model Comparison - Now we compare all the models we’ve ran on this dataset across homework 2 and 3. We see that the Neural nets 25th percentile of testing accuracy is above every other algorithm 75th percentile. This tells us that this model has the best performance vs the algorithms we’ve tested so far. It goes to show us that there is not one best algorithm, we must test different algorithms over the same dataset to see which will perform optimal. There’s no free lunch!

