I have downloaded all datasets from the UCI Machine Learning Repository. In addition, I've used SKLearn, Numpy, Pandas and other approved libraries.

All relevant files can be found at: github.com/mnarasim/ML7641

Introduction: I've considered two diverse data sets for modeling and analysis. Specifically, my first data set is about predicting if someone has a heart disease. The next dataset is about predicting if a customer will default on their credit card payment in future.

To me both these datasets are interesting because they represent common real-life problems. Though the accuracy of the prediction is important for both the datasets, I feel like there is no room for error in predicting if someone has a heart disease or not. If someone is falsely diagnosed with a heart disease, then it can have significant mental trauma on that person. On the other hand, if a diagnosis is missed it can have fatal repercussions as well. However, in the case of the credit card dataset any error in predictions don't have serious repercussions.

Heart.csv: This dataset has 1,025 samples where each sample has 13 attributes. Though the sample size seems low, I like this dataset because in such small datasets the likelihood of having noisy data will be minimal. Thus, my model will not over fit the data. Each of the attributes have a diverse minimum and maximum range which makes the classification task interesting. For example, the age attribute varies from 29 –77 while resting blood pressure varies from 94 – 200. The data also includes categorical variables like Male and Female, though they have been converted to numbers. Also, this dataset includes almost 50% - 50% of the classifications of each type (heart disease or no heart disease). This way we can be sure that the dataset is not biased in anyway and has a good sample set of both the classification labels. Furthermore, as shown below most of the attribute values are not normally distributed which makes the dataset complex and pose a challenge to the learning algorithm.

Heart.csv	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00
mean	54.43	0.70	0.94	131.61	246.00	0.15	0.53	149.11	0.34	1.07	1.39	0.75	2.32	0.51
std	9.07	0.46	1.03	17.52	51.59	0.36	0.53	23.01	0.47	1.18	0.62	1.03	0.62	0.50
min	29.00	0.00	0.00	94.00	126.00	0.00	0.00	71.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	48.00	0.00	0.00	120.00	211.00	0.00	0.00	132.00	0.00	0.00	1.00	0.00	2.00	0.00
50%	56.00	1.00	1.00	130.00	240.00	0.00	1.00	152.00	0.00	0.80	1.00	0.00	2.00	1.00
75%	61.00	1.00	2.00	140.00	275.00	0.00	1.00	166.00	1.00	1.80	2.00	1.00	3.00	1.00
max	77.00	1.00	3.00	200.00	564.00	1.00	2.00	202.00	1.00	6.20	2.00	4.00	3.00	1.00

age -> Null hypothesis can be rejected
sex -> Null hypothesis can be rejected
cp -> Null hypothesis can be rejected
trestbps -> Null hypothesis can be rejected
chol -> Null hypothesis can be rejected
fbs -> Null hypothesis can be rejected
restecg -> Null hypothesis can be rejected
thalach -> Null hypothesis can be rejected
exang -> Null hypothesis can be rejected
oldpeak -> Null hypothesis can be rejected
slope -> Null hypothesis can be rejected
ca -> Null hypothesis can be rejected
thal -> Null hypothesis can be rejected
ID -> Null hypothesis can be rejected
LIMIT_BAL -> Null hypothesis can be rejected
SEX -> Null hypothesis can be rejected
EDUCATION -> Null hypothesis can be rejected
MARRIAGE -> Null hypothesis is accepted
AGE -> Null hypothesis can be rejected

Credit Card.csv: This dataset has 30,000 samples where each sample has 24 attributes. As before, each of the attributes has a diverse minimum and maximum range which makes the classification task interesting. Also, this dataset includes negative values for some attributes. The data also includes categorical variables like Male and Female, though they have been converted to numbers. Unlike the previous dataset, this dataset includes 78% of the samples with a 0 (No default) classification, and 1 (will default) classification. It would be really interesting to see how various learning algorithms can overcome any bias in the dataset and come up with an accurate prediction. Furthermore, as shown below none of the attributes are normally distributed which makes the dataset complex and pose a challenge to the learning algorithm.

Credit Card.csv	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
count	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00
mean	15,000.50	167,484.32	1.60	1.85	1.55	35.49	(0.02)	(0.13)	(0.17)	(0.22)	(0.27)	(0.29)	51,223.33	49,179.08	47,013.15	43,262.95	40,311.40	38,871.76	5,663.58	5,921.16	5,225.68	4,826.08	4,799.39	5,215.50	0.22
std	8,660.40	129,747.66	0.49	0.79	0.52	9.22	1.12	1.20	1.20	1.17	1.13	1.15	73,635.86	71,173.77	69,349.39	64,332.86	60,797.16	59,554.11	16,563.28	23,040.87	17,606.96	15,666.16	15,278.31	17,777.47	0.42
min	1.00	10,000.00	1.00	0.00	0.00	21.00	(2.00)	(2.00)	(2.00)	(2.00)	(2.00)	(2.00)	(165,580.00)	(69,777.00)	(157,264.00)	(170,000.00)	(81,334.00)	(339,603.00)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	7,500.75	50,000.00	1.00	1.00	1.00	28.00	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	3,558.75	2,984.75	2,666.25	2,326.75	1,763.00	1,256.00	1,000.00	833.00	390.00	296.00	252.50	117.75	0.00
50%	15,000.50	140,000.00	2.00	2.00	2.00	34.00	0.00	0.00	0.00	0.00	0.00	0.00	22,381.50	21,200.00	20,088.50	19,052.00	18,104.50	17,071.00	2,100.00	2,009.00	1,800.00	1,500.00	1,500.00	1,500.00	0.00
75%	22,500.25	240,000.00	2.00	2.00	2.00	41.00	0.00	0.00	0.00	0.00	0.00	0.00	67,091.00	64,006.25	60,164.75	54,506.00	50,190.50	49,198.25	5,006.00	5,000.00	4,505.00	4,013.25	4,031.50	4,000.00	0.00
max	30,000.00	1,000,000.00	2.00	6.00	3.00	79.00	8.00	8.00	8.00	8.00	8.00	8.00	964,511.00	983,931.00	1,664,089.00	891,586.00	927,171.00	961,664.00	873,552.00	1,684,259.00	896,040.00	621,000.00	426,529.00	528,666.00	1.00

PAY_0 -> Null hypothesis can be rejected
PAY_2 -> Null hypothesis can be rejected
PAY_3 -> Null hypothesis can be rejected
PAY_4 -> Null hypothesis can be rejected
PAY_5 -> Null hypothesis can be rejected
PAY_6 -> Null hypothesis can be rejected
BILL_AMT1 -> Null hypothesis can be rejected
BILL_AMT2 -> Null hypothesis can be rejected
BILL_AMT3 -> Null hypothesis can be rejected
BILL_AMT4 -> Null hypothesis can be rejected
BILL_AMT5 -> Null hypothesis can be rejected
BILL_AMT6 -> Null hypothesis can be rejected
PAY_AMT1 -> Null hypothesis can be rejected
PAY_AMT2 -> Null hypothesis can be rejected
PAY_AMT3 -> Null hypothesis can be rejected
PAY_AMT4 -> Null hypothesis can be rejected
PAY_AMT5 -> Null hypothesis can be rejected
PAY_AMT6 -> Null hypothesis can be rejected

I also scaled the data using SKlearn scaling functions to avoid any bias due to the diverse data ranges of attributes.

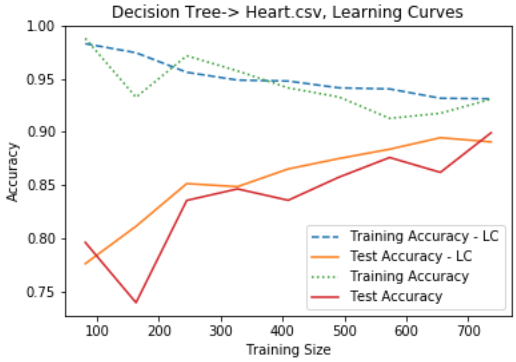
Analysis:

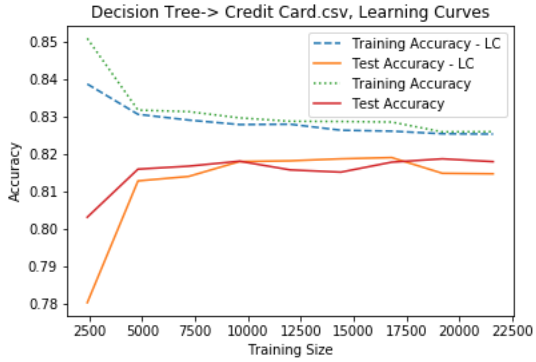
Learning curves: I ran all the models for varying training and test sizes and observed the accuracy of both training and test samples. My objective is to observe the regions where my models under fit and over fit the data. If the training data size is very small, then the model might under fit the data as the sample size might be insufficient to learn about the problem and come up with an accurate prediction. On the other hand, if the training size is too big, then the model might be overthinking and even considering the noise in the data thus overfitting the data. In the below graphs I'm comparing the training and test accuracies for different training/test sizes with and without cross validation (wherever the legend ends with an "LC" it means cross validation is included as I used the SKLearn Learning Curve function to obtain the same).

- **Decision Trees:** For both the datasets chosen the training accuracy is high. For the heart data the test accuracy without cross validation really goes up and down for various training sizes. However, the test accuracy obtained from the learning curve method (that includes cross validation) is fairly smooth except for a drop at size 300, which makes me believe that when cross validation is considered the model is well trained on the data that is a good representative of what it might be

tested on in future, which is not the case when cross validation is not performed. For training size over 650, I see the test accuracy without cross validation increasing but I feel that's just overfitting.

In case of the Credit Card data, I can see that the accuracy is slightly lower than the heart data set and I can attribute that to the complexity of the dataset. As the training data set size increases I can see the training accuracy decrease in both cases – with and without cross validation. This is due to the complexity of the data, as the data set size increases the model learns more about the problem hence the accuracy decreases. This is also reflected in the low test accuracy, for low training dataset sizes. The model is not able to generalize its learnings to the test data. I also see some overfitting at the training size of 17,500 and above by looking at the test accuracy curve with cross validation.

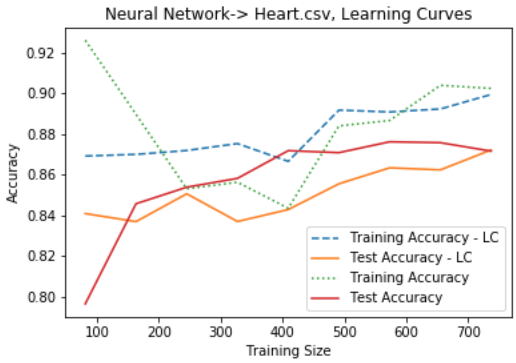


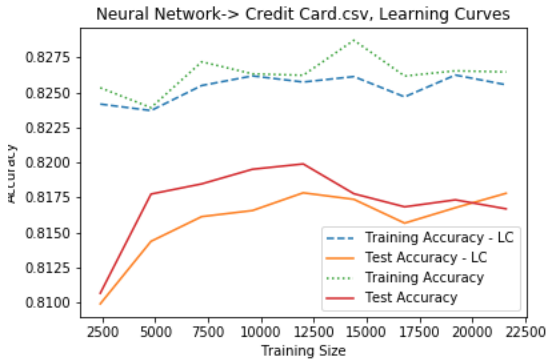


- **<u>Neural Network:</u>** For the heart data, I can see that the training accuracies are lower than what we observed for the Decision Trees for any training size. However, in case without cross validation I can even see that the test accuracy is better than the training accuracy at a training size of 400. I can think of two reasons why this might be happening 1) small dataset – the size of the heart data itself is small for Neural Network 2) the model gets struck in local optima. If I look at both the training and test accuracies in the case with cross validation, I can see that model over fits the data at training data set size of 400.

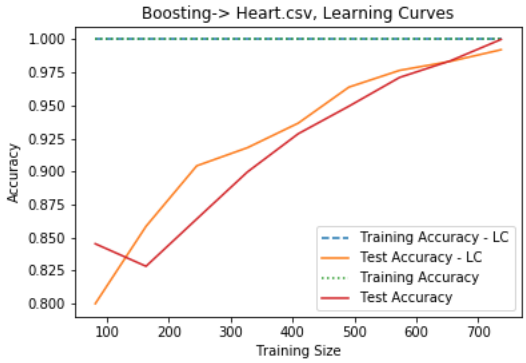
  For the Credit Card data, I see that model does well when cross validation is included. Though the training and test curves look far apart, if we look closely it is the scale of the y-axis that is making it look so. Also, both the training and test curves follow a similar pattern, so I conclude that this model fits this dataset well.

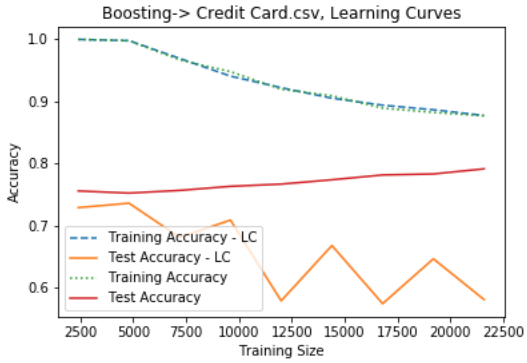
  The fact that the test accuracy without cross validation is better than with cross validation is not alarming as cross validation makes sure that the model sees a more realistic representation of the test data which is better suited for reality.



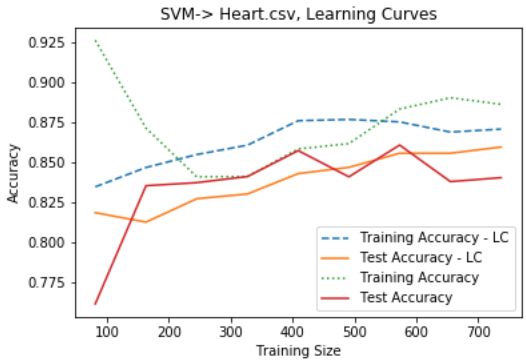


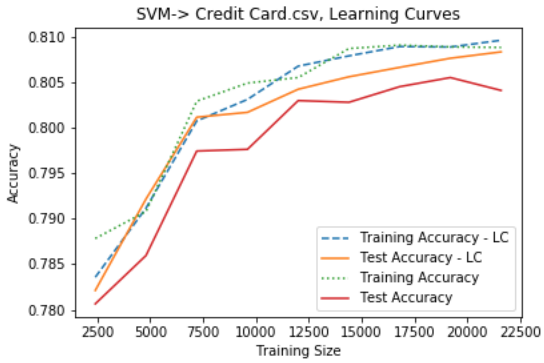
- **Boosting:** I used a Decision Tree model for the AdaBoost Classifier. The model does well for the heart dataset, the training accuracy is high and stable. The test accuracy also increases as the training size increases. Cross validation also helps in this case as the test accuracy is smooth.
  For the Credit dataset, cross validation doesn't seem to help at all because of the shape of the test accuracy curve. It looks like the training dataset is not well representative of the test dataset, in other words the training dataset doesn't help to learn much about the problem when cross validation is considered. Surprisingly, the test accuracy without cross validation is better and stable than the accuracy with cross validation.



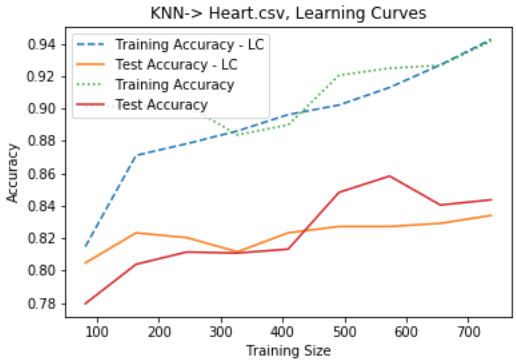


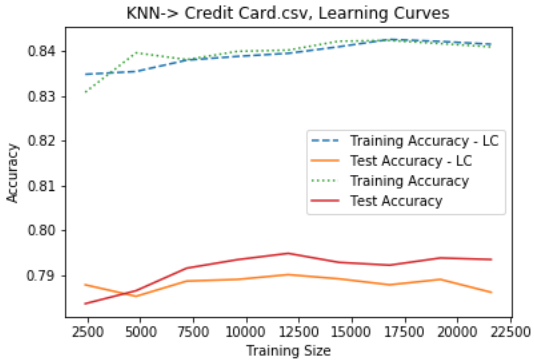
- **SVM:** The model does well on both datasets. For both the datasets, I can see that the training and test accuracies are close when cross validation is performed. I can also see that these curves are smoother than in the case when cross validation is not performed. I would consider this model also as a reasonable predictor for both the datasets. Looking at the graphs makes me feel that the model is able to find a plane that linearly separates the data, hence the good fit.





- **KNN:** Surprisingly, this model doesn't do very well on both the datasets. Though the test accuracy for the heart dataset is not far off when compared to other models the test accuracy of the Credit Card dataset is low when compared to many other models. In the case of heart dataset, the training accuracy with cross validation keeps increasing with the training data size. However, I see that the test accuracy in the same case doesn't increase proportionally perhaps due to overfitting after a training size of 500. I see a similar pattern in the Credit Card dataset at around 15,000 training data size.



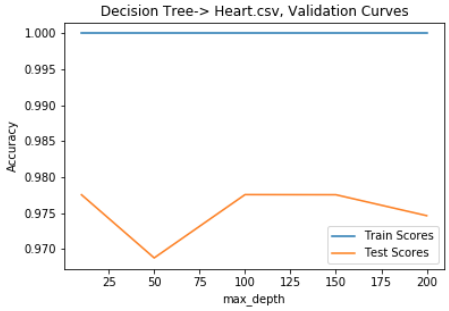


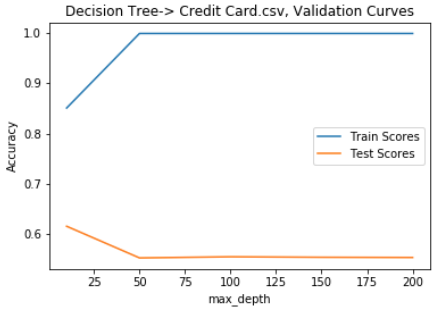
Overall, I would use Decision Trees, AdaBooster or SVM classifier for the heart dataset. For the Credit Card dataset, I would use either Decision Trees, Neural Network or the SVM classifier.

Validation Curves:

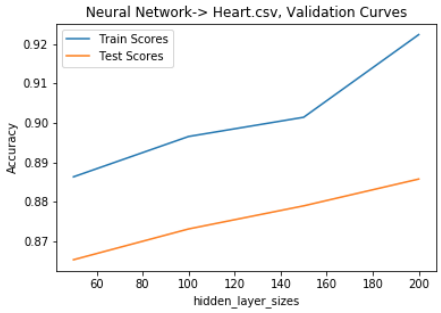
Next, I analyzed the models for both datasets for different values of a key parameter that is relevant to that classifier.

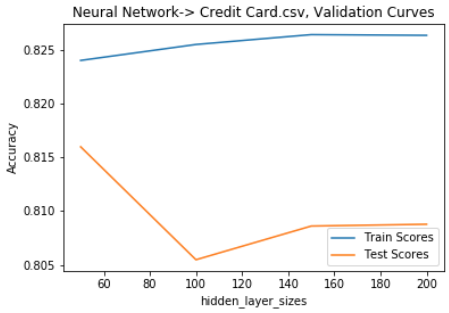
- **Decision Trees:** Both datasets do well on the training accuracy for varying sizes of the max_depth parameter. However, both datasets show a decline in the accuracy at a max_depth of 50 and I feel like under fitting is the cause. The bigger the size of this parameter the model generalizes its learning and under fits the data. For really small sizes of this parameter the model will over fit the data. Hence, I feel a values from 5 to 10 for this parameter would fit the data well.



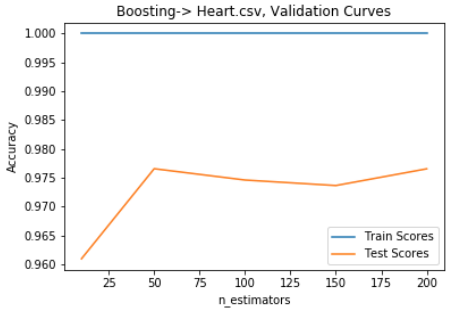


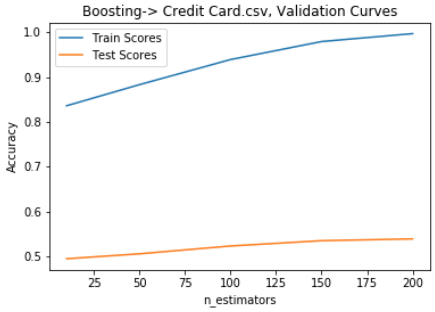
- **<u>Neural Network:</u>** The heart dataset shows how the training and test accuracies increase with the hidden layer sizes, though after a value of 160 we can see that the training accuracy increases more than the test accuracy. However, for the Credit Card dataset the there is a drop in the test accuracy at a hidden layer size of 100. This could be due to the fact that the test data chosen is non-linearly separable and perhaps requires more hidden layers at that instance.



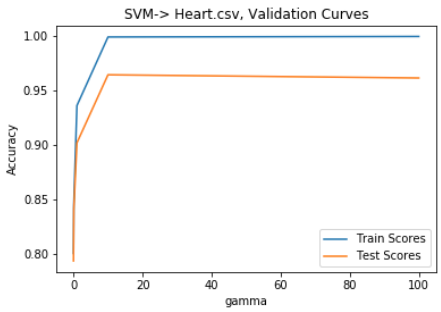


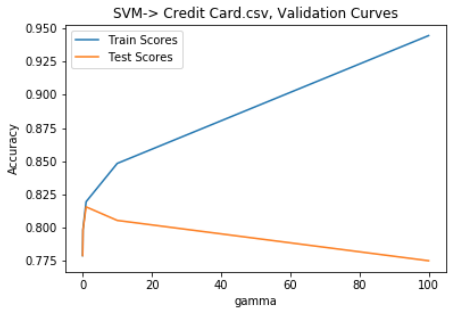
- **<u>Boosting:</u>** For the Heart dataset the test accuracy increases as the number of estimators increases and it is also very close to a perfect training accuracy. The verdict is varying the number of estimators does make a difference for this dataset. However, for the Credit Card dataset the increase in test accuracy is not on par with the increase in training accuracy as the number of estimator's increases.



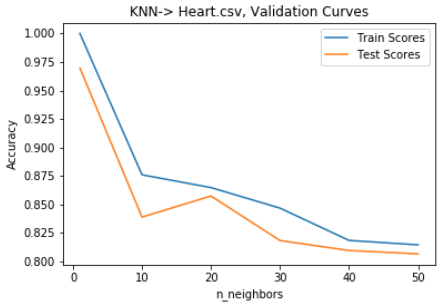


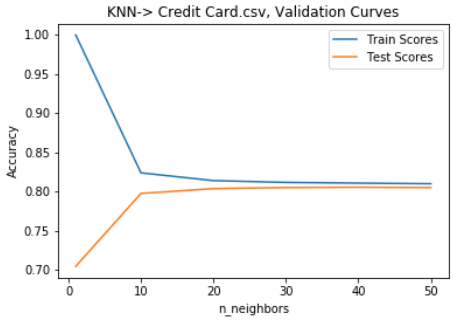
- **<u>SVM:</u>** For both datasets the low values of gamma result in under fitting as both the training and test accuracies keep increasing with Gamma. However as the Gamma increases over 10 we can see how the training accuracy increases and the test accuracy either decreases or diverges from the training accuracy. This is due to overfitting and can be seen clearly in the Credit Card dataset.





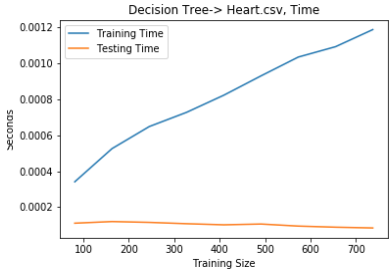
- **KNN:** The training accuracy is high for both datasets for low values of K. However, for the heart dataset the test accuracy is high at first but drops as the value of K increases. This is because as the value of K increases the model learns more about the problem hence the accuracy is not as high as before. As the K value increases over 40 we can see the training and test accuracies converging. For the Credit Card dataset low values of K lead to a high training accuracy and low test accuracy, highlighting the overfitting issue. The model is not able to generalize its learning to the test data at low values of K. As the K value increases above 10 we can see both the training and test accuracies converging.

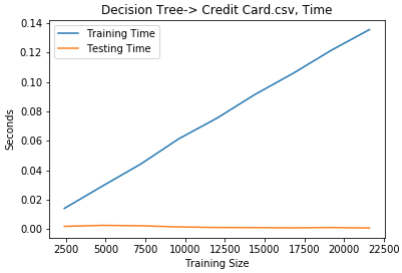


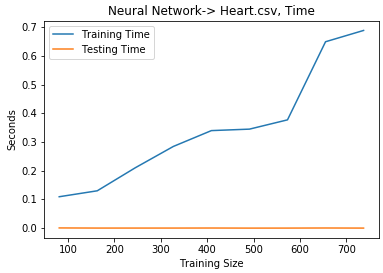


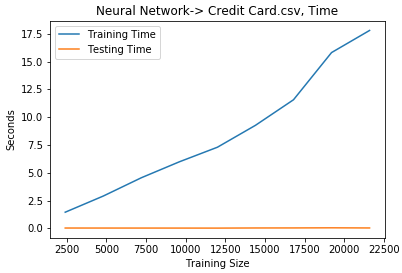
Time:

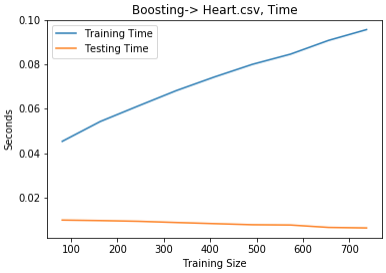
As shown below the training time increases as the training dataset size increases while the testing time is constant irrespective of the data size for all models except for KNN. For KNN the time to train is constant while the time to test increases with dataset size. For KNN the decline in the test time at the end of the graphs shown below is because of the small test data size i.e. bigger training dataset size. In terms of time, Decision Tree and KNN classifiers are faster when compared to Neural Networks and SVM.

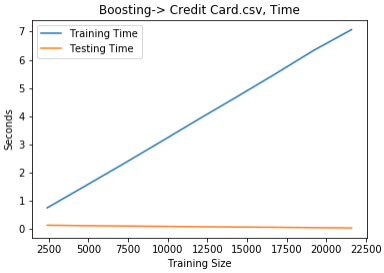


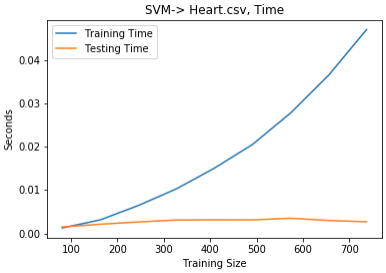


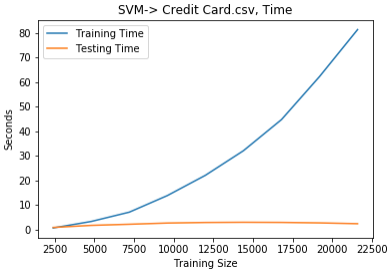


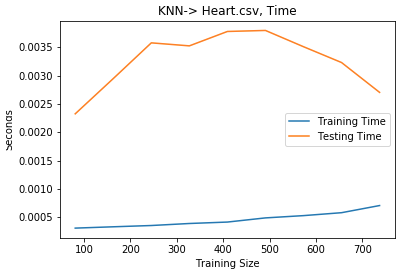


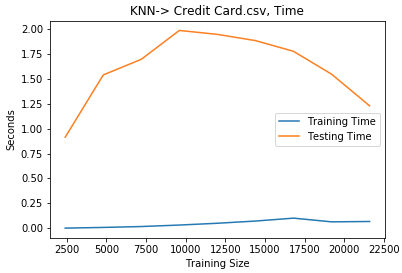










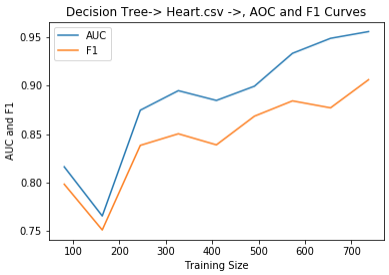


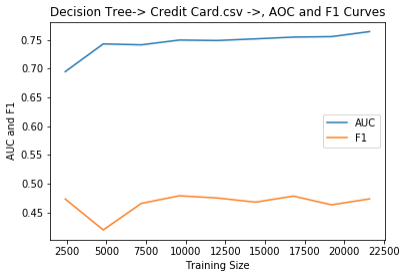
F1 and AOC Curves:

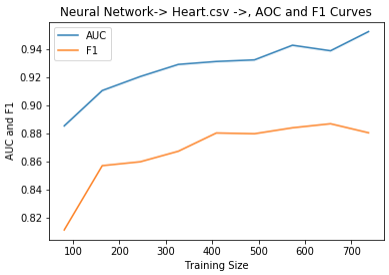
- **F1 Score:** So far we have looked at the accuracy of the models for the chosen datasets. It has to be remembered that in a classification problem accuracy is calculated based on the True Negatives and True Positives i.e. (True Positive + True Negative)/ (True Positive + True Negative + False Positive + False Negative). However, for both our datasets the impact of a False Positive and False Negative is huge. For example, if a heart disease patient is classified as someone without a heart disease and a customer who will not default their payment is classified as someone who is likely default, it can have adverse effects. Thus, Precision and Recall are important metrics that basically assess 1) out of the predicted positives, how many instances are really positive 2) out of the actual positives

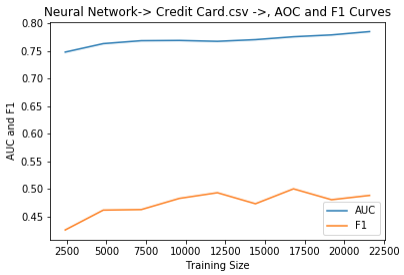
how many does our model really capture correctly, respectively. Obviously, there's a tradeoff between these metrics and thus we use F1-score. Based on the F1-scores I would say both Decision trees and Neural Networks are reasonably good predictors.

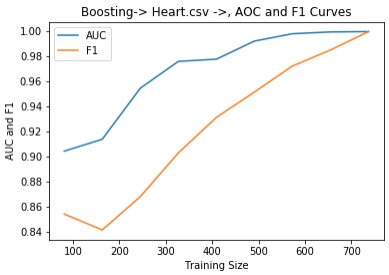
- **AUC:** AUC shows how well the model classifies the data accurately considering different thresholds. It plots both True Positive (y-axis) vs False Positive (x-axis) for varying threshold values and calculates the area under the curve. The idea is that a perfect model will have True Positive rate of 1 and False Positive rate of 0, and if the True Positive rate and the False Positive rate is 0.5, it means the model has no ability on the predictability of the classes. Looking at the AUC curve of all the models for both the datasets I feel like both Decision trees and Neural Networks are good predictors.

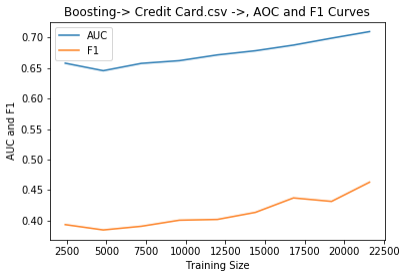


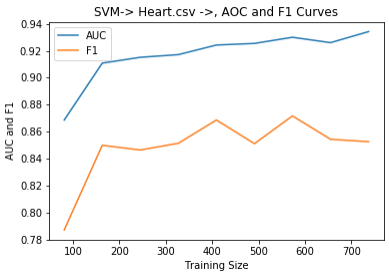


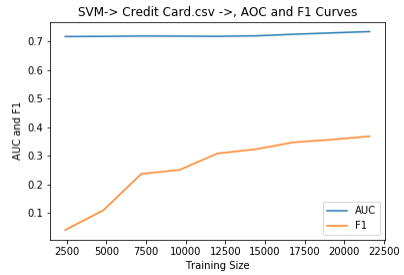


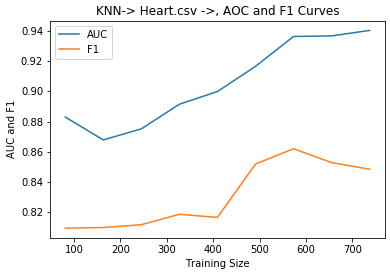


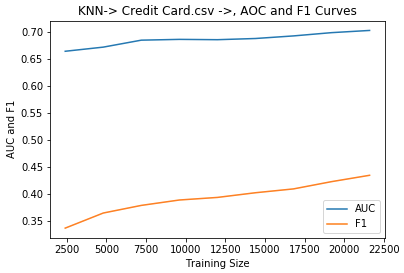






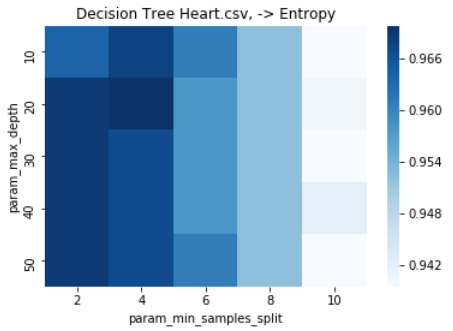


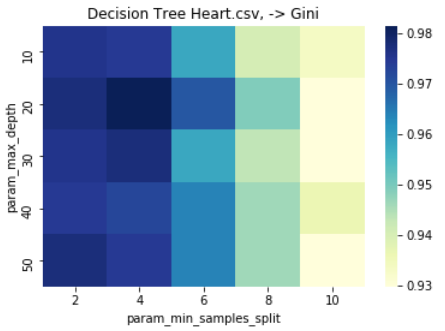


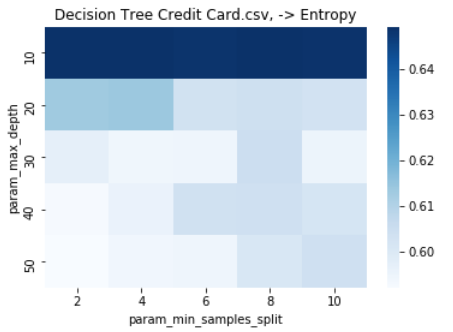


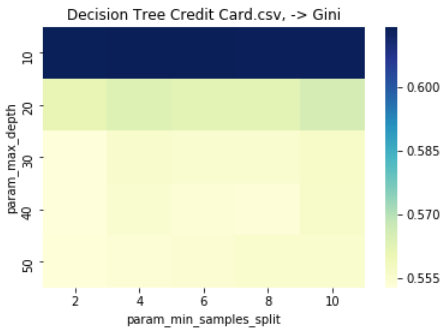
Grid Search: Grid search was used to run each of the models for varying parameter values and the accuracies shown below are the test accuracies.

- **Decision Trees:** The model was run for both types of impurity calculations – Gini and Entropy. For each impurity type both parameters minimum samples to be split and the max depth were run for varying values. For the simple dataset heart.csv, I see that low values for the minimum samples to be split and low maximum depth produce a good accuracy irrespective of the impurity calculation method. However, for the bigger Credit Card.csv, the min samples to be split doesn't matter as the best test accuracy is obtained for a maximum depth of 10 irrespective of the minimum samples to be split. As the max depth increases the model does poorly with declining test accuracy. This is perhaps due to under fitting, that higher values of max depth makes the model incapable of understanding the problem and generalizing the same to the test data.



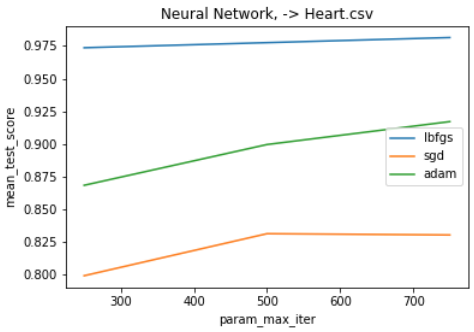


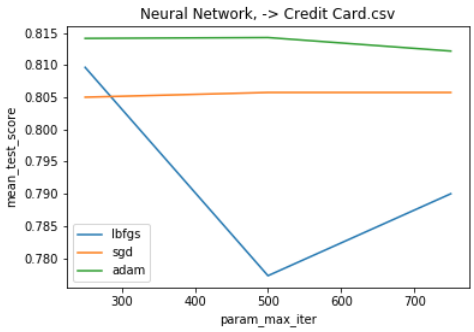




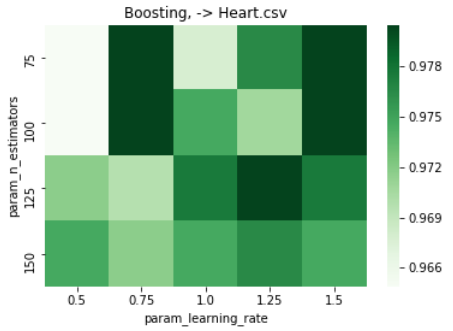
- **Neural Network:** For the heart dataset lbfgs solver performs the best whereas adam performs well for the larger credit card dataset. This is because lbgfs works well on small datasets, and for large data sets such as the credit card, it just needs more iterations to

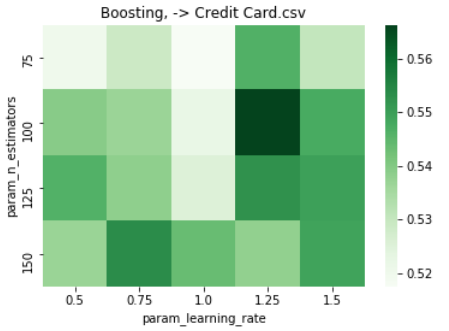
converge, hence we see a drop in the accuracy until we reach 500 iterations. However, solver adam works well for both datasets.



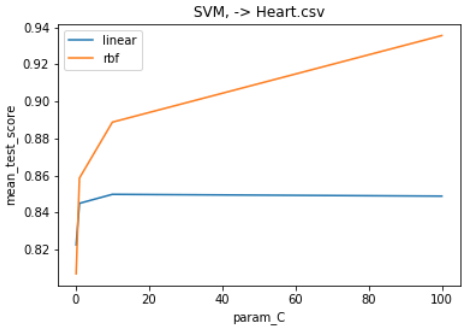


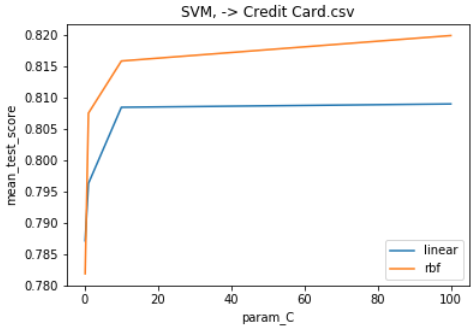
- **<u>Boosting:</u>** For both datasets best accuracies are obtained for low values of the number of estimator's parameter. I think there's a tradeoff between the learning rate and the number of estimators. If I look at the graphs below I can see that the accuracy improves as the number of estimators increases when the learning rate is really low. In other words, with a low learning rate the model needs more estimators to accurately predict the target variable. However, if the number of estimators keeps increasing for the same learning rate then the accuracy either plateaus of drops.





- **<u>SVM:</u>** For both datasets rbf kernel performs well for increasing values of the C parameter. The fact that rbf does better than the linear kernel makes me believe that both the problems at hand are more non-linear than linear.





- **KNN:** For the smaller dataset the case where all neighbors are equally weighed perform poorly on the test accuracy as the number of neighbor's increases. However, the case where the closer neighbors have a greater influence than the farther points performs well as the number of neighbor increases. This makes sense because when all neighbors have uniform weights the model is not able to learn from each neighbor. In case of the large dataset, both these cases yield a test accuracy that is high and close to each other for increasing number of neighbors. This could be due to the neighbors being more or less equally spaced from each other.