CSL603

Linear and Logistic Regression

Assignment 2

In this assignment our target is to train a model that can learn to find the age of Abalone giving some other parameters. We use linear regression to learn a linear boundary for the prediction and to reduce the complexity of the model I have used regularization with different values of lambda.

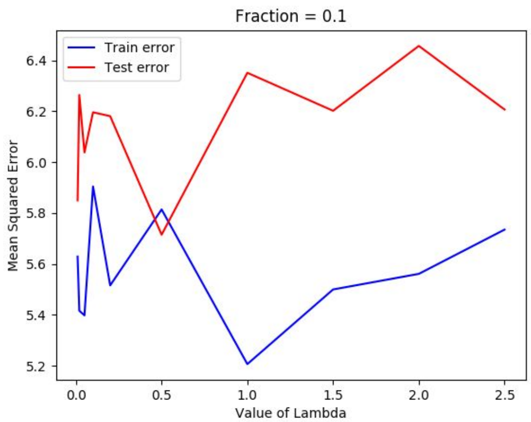
1) In the first step of learning, we perform standardization of data in order to avoid unwanted greater importance to attributes with higher ranges. Standardization is done by subtracting the mean value of the attribute from each value and dividing by its standard deviation.
2) I have made sets of lambdas and fractions. The code was run on each combination of training set fraction and lambda 100 times and the average output for each was plotted. It is necessary to take average of multiple runs because test-train set selection is random and averaging increases chances of high reproducibility of the results.
3) The graphs obtained for different training set fraction have been shown below:

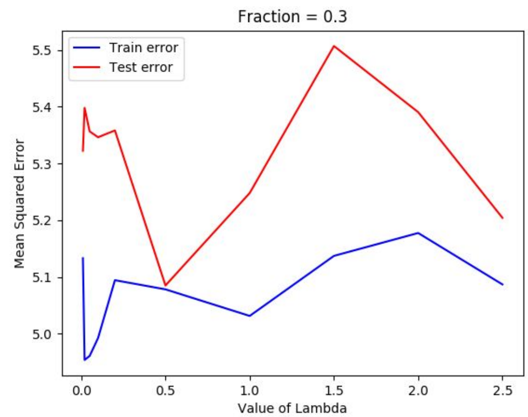
    The variation is with the value of lambda i.e. the regularization parameter.

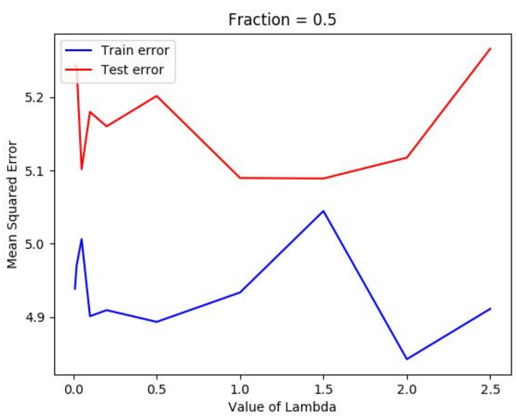
Some Observations:

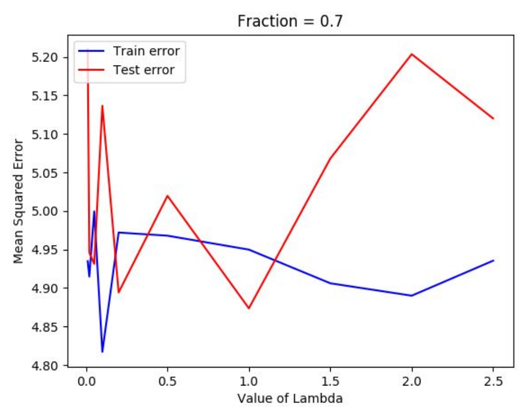
1) 1000 iterations have been used in linear regression
2) <20 iterations are enough for Newton Raphson to give the desired output. However I have set the upper limit to 100.

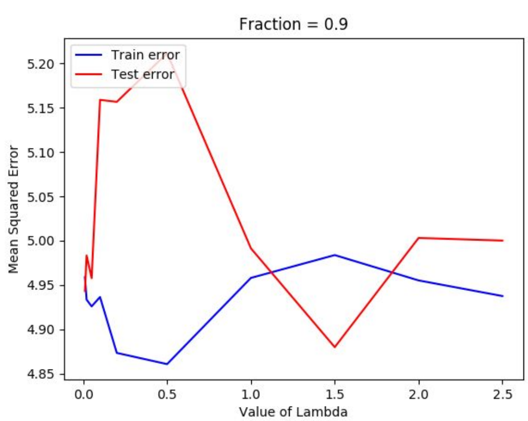
Plot of mean squared error for **“varying values of lambda”** (regularization parameter) but **“fixed training set fraction”**

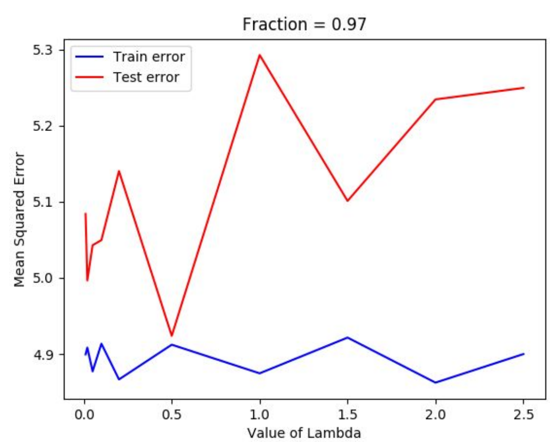








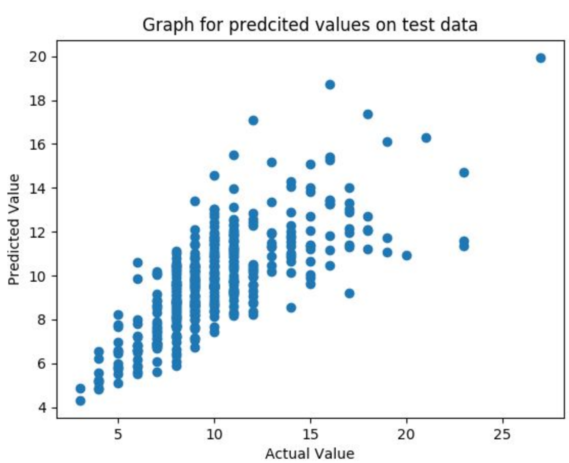


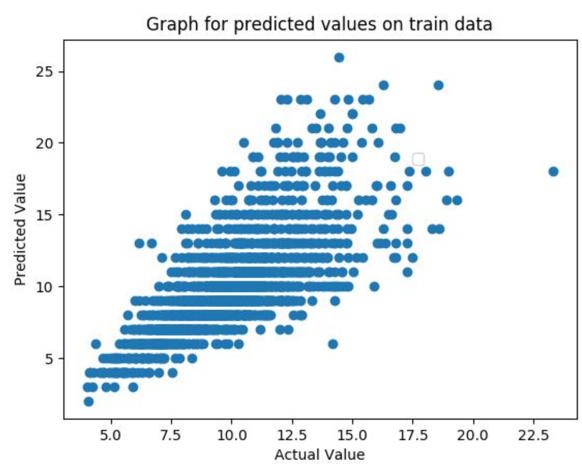


Analysis of the graphs:

1) As we increase the regularization parameter the test error decreases. Due to reduction in the complexity of the model, it is more likely to give good results on generic test inputs.
2) The train error will increase with λ because regularization increases bias of the model.
3) If lambda is increased after a certain limit (which depends on training set fraction) I observed that train and test errors begin to increase which shows that there is a limit upto which we can regularize the data, above that the model becomes too simple that it is unable to give useful predictions. Value of an appropriate lambda can be found only empirically.

The graph for actual vs predicted values is as follows:



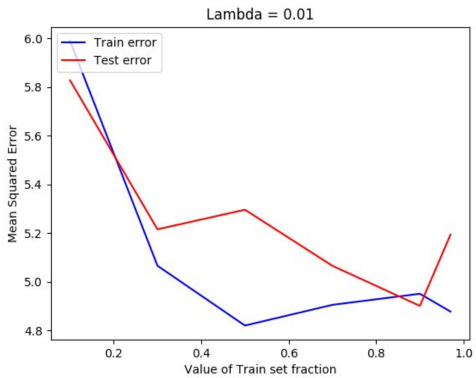


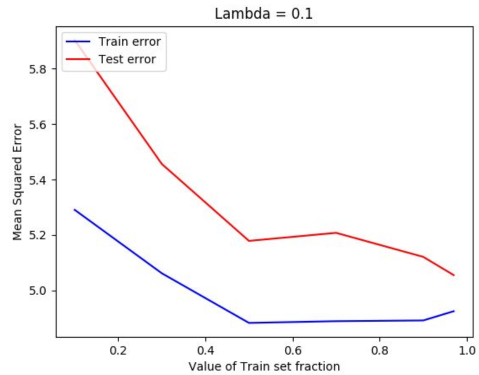
The above graph show that the model is not being able to properly learn a linear boundary for separating the data. A huge number of points are falling above and below the expected 45 degree line.

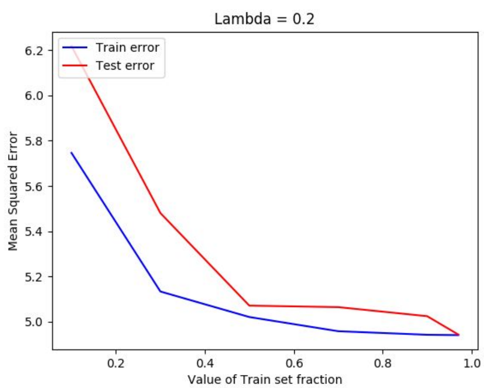
a) Does the effect of λ on error change for size of the training set?

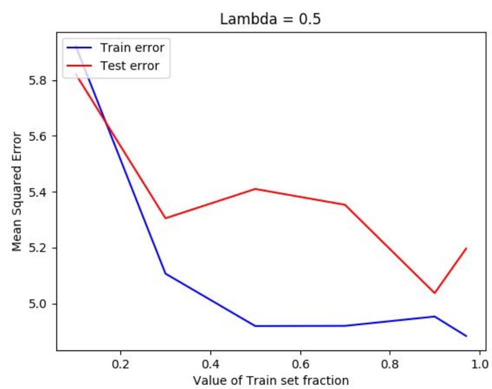
i) λ is the regularization parameter that is used to reduce the complexity of the model. For a fixed λ the variation of accuracy for different training set fractions is as shown in the following figures:

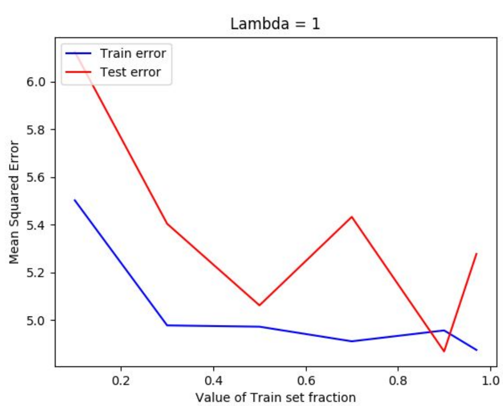
Plot of mean squared error for **“varying values of training set fraction”** but **“fixed lambda i.e. regularization parameter”**

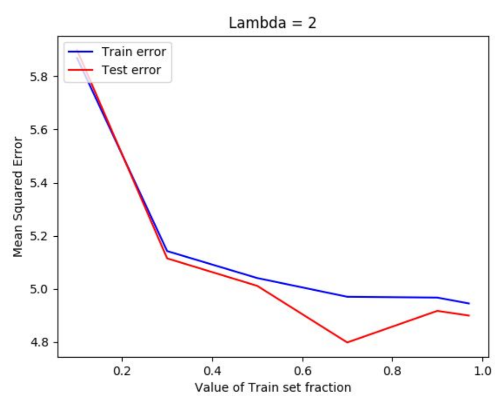










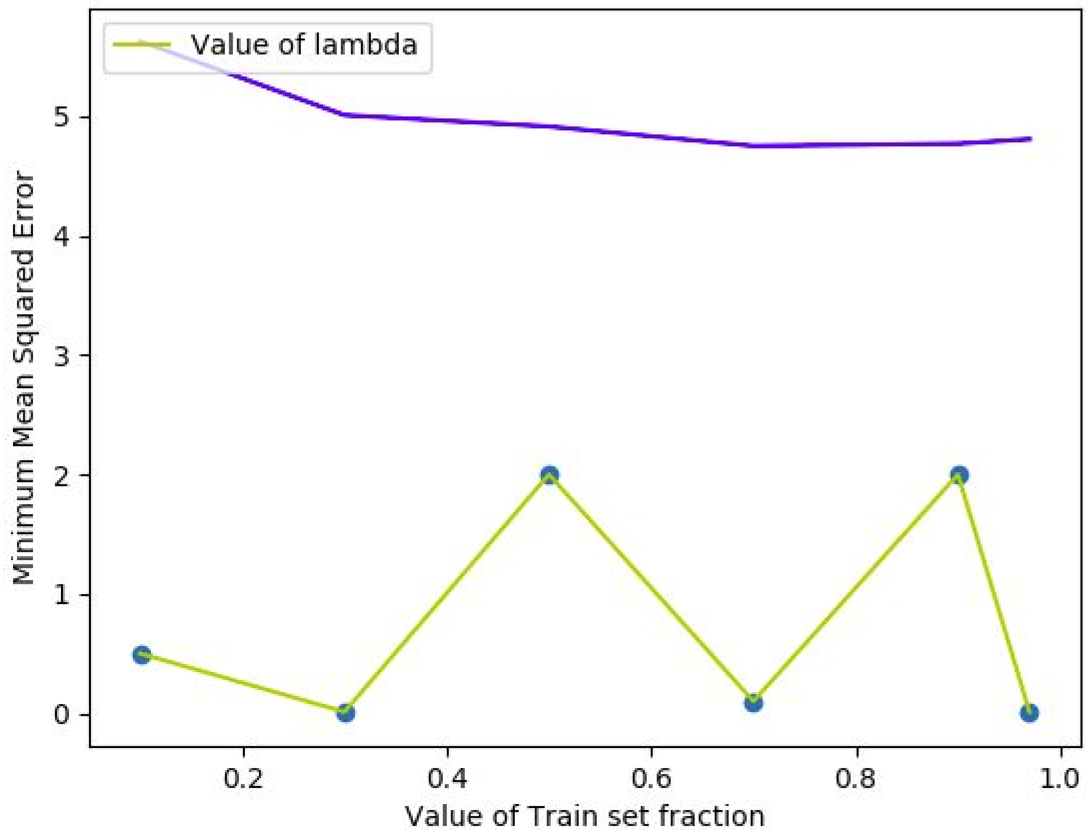


The above graphs display the variation of mean squared error with training set fraction for a fixed lambda. We can observe that the error decreases as the training set fraction increases. This is evident because a larger training set ensures a better a model than a smaller one. The effect of regularization is possibly larger for smaller datasets.

b) How do we know if we have learned a good model?

i) The “goodness” of a model depends on several factors which include train error , test error i.e. bias and variance. In the graphs above the blue line represents the training error while the red line is the test error. Bias is the error corresponding to wrongly labelled data. Higher the bias, lesser the accuracy. But extremely low bias on training set indicated high level of overfitting which is undesirable as such a model will not perform well on the test data. Hence we use regularization to reduce the complexity of such models.

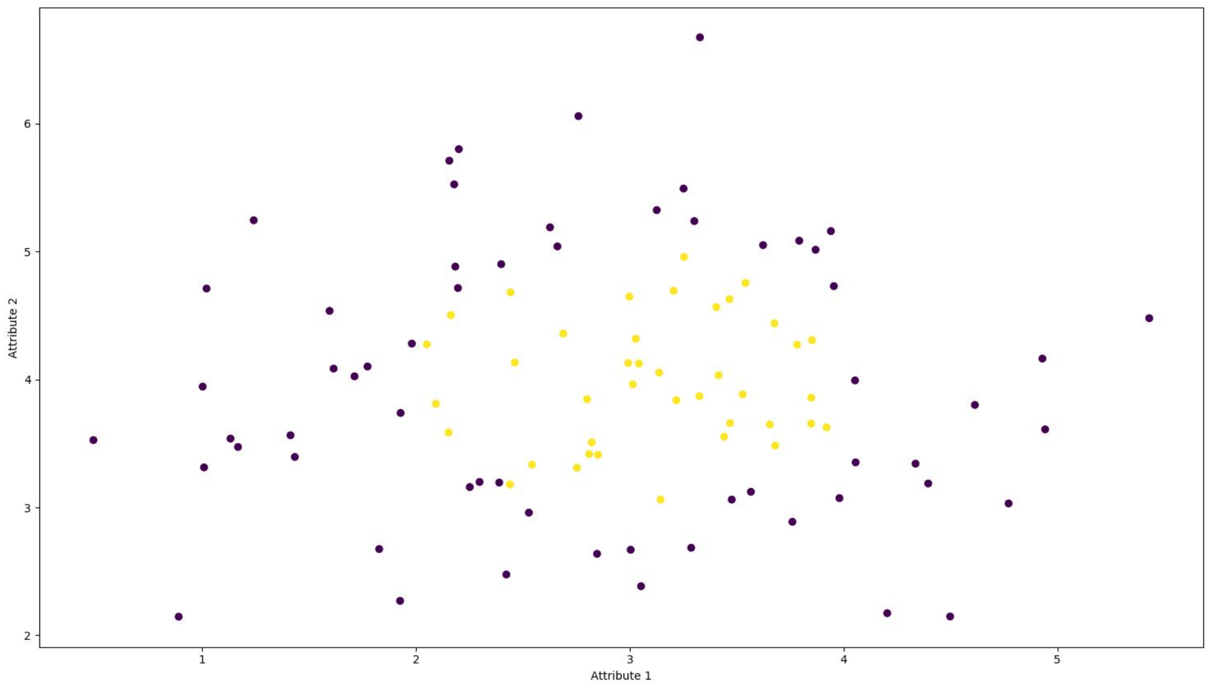
Plot of minimum mean squared error for **“varying values of training set fraction”**



Observation: For minimum mean squared error for varying fractions, the corresponding value of regularization parameter is observed to be fluctuating. There is no particular trend being followed as is evident in the above graph.

Logistic Regression

Plot of the training data:



Is the data linearly separable?

The plot shown above shows that the given dataset cannot be trained using a linear boundary. The data points are not linearly separable. The expected decision boundary should be close to elliptical in shape.

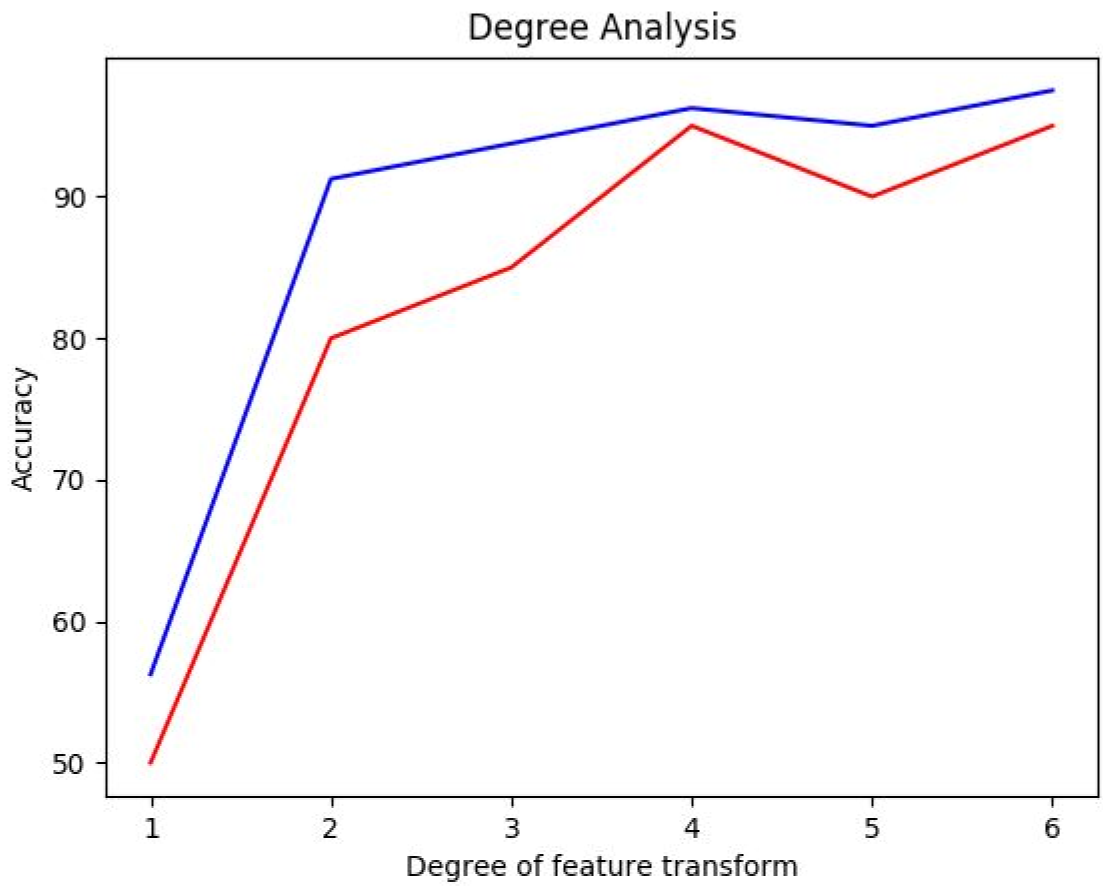
I have used Newton Raphson method for my analysis.

Effect of Number of iterations:

1) For Newton Raphson code converges to the minimum in 10-15 iterations.
2) For gradient descent the code converges in approximately 100 iterations.

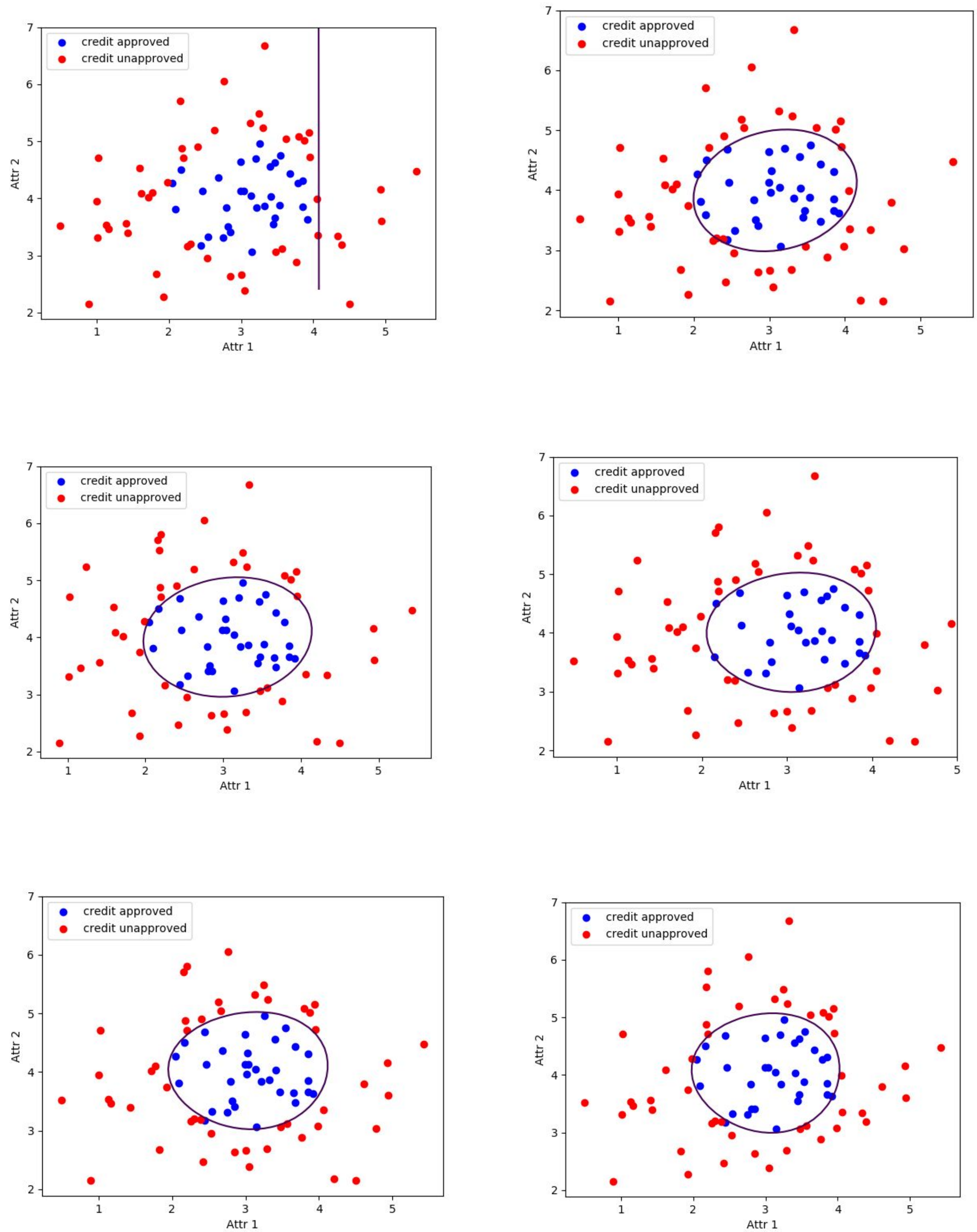
Degree vs Accuracy:

Following is the plot showing variation of accuracy with degree of feature transform averaged over 15 runs :



Given the results shown above I have chosen degree=4 for further analysis. I tried 10 runs each for different values of degree for various combinations of lambda and training set fraction. In most of the runs degree=4 gave the best accuracy.

Contour Plots with varying degree: [1 to 6]



As λ progressively increases the learned boundary becomes more and more accurate. However after degree = 4 , there is no considerable change observed and the learned model possibly overfits the training data as we can see the accuracy reduces with increasing degree.

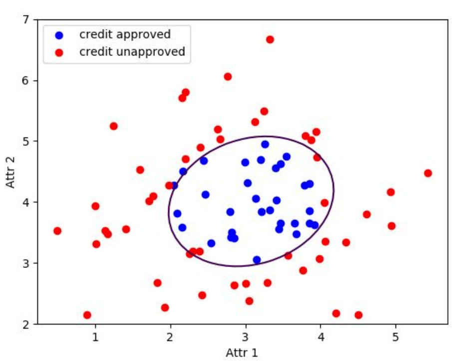
Note: Degree has been fixed to 4 for further analysis

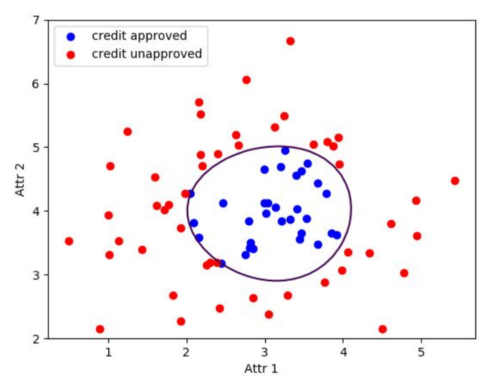
Contour Plots with varying lambda:

As the value of the regularization parameter increases the accuracy of the model considerably decreases for underfit curves. However for high degrees all curves are able to approximate the model really well irrespective of lambda. This shows that for higher degree polynomials we will need larger values of lambda to perform regularization.

Following graphs shown are consistent with the observations.

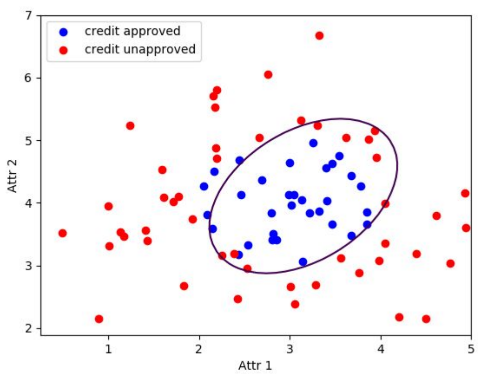
1) **λ = 0.01**



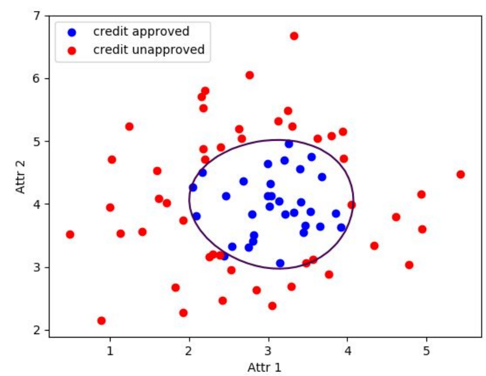


2) λ = 0.1

Under Fit curve (Degree = 2)

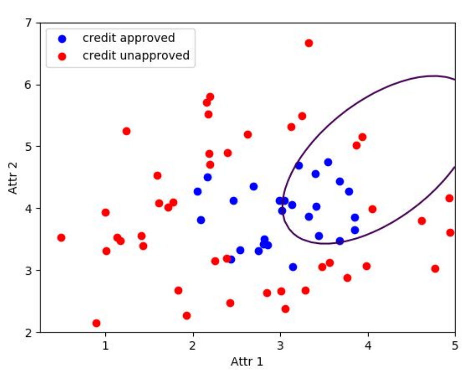


Over Fit Curve (Degree = 6)



3) λ = 0.5

Under Fit curve (Degree = 2)



Over Fit Curve (Degree = 6)