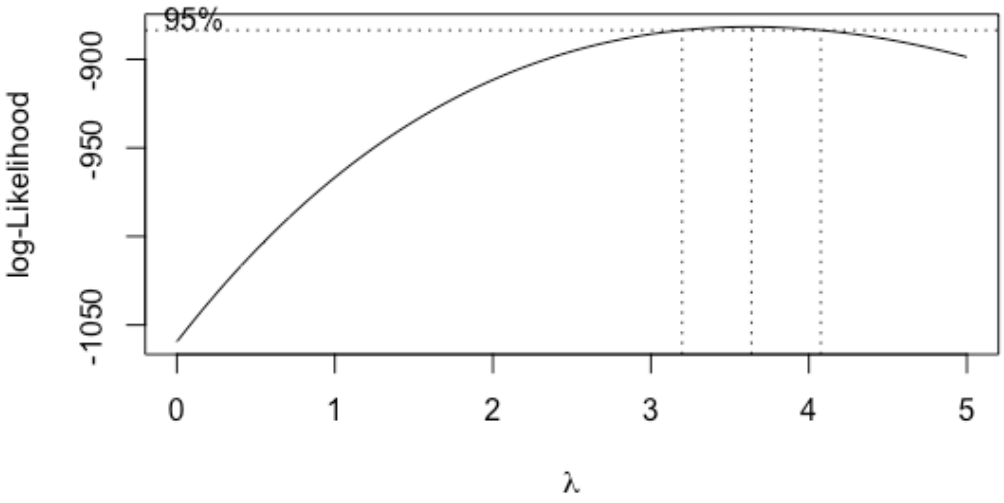
Ryley Higa
higa2

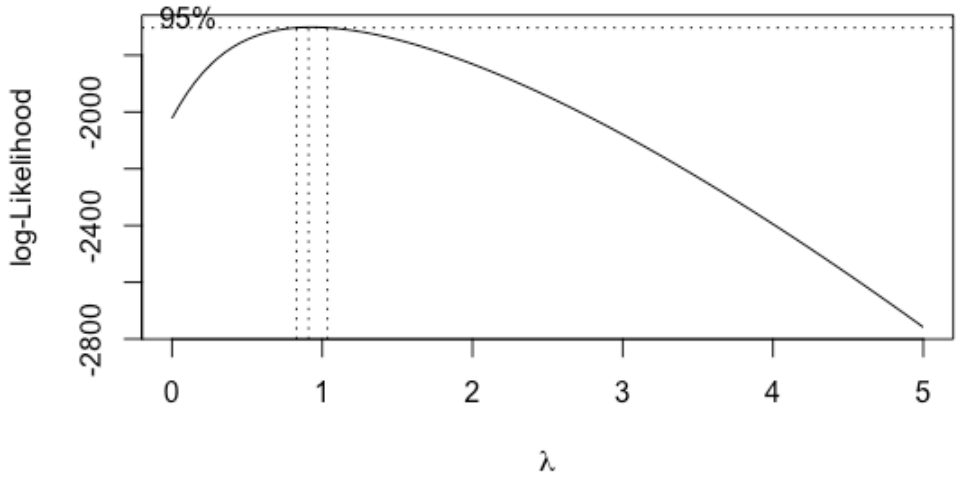
Homework 6: Report

Simple Linear Regression Versus Box-Cox Transformation

The box-cox parameter, lambda, was chosen based on maximum likelihood. In order to get the estimates to be positive, I added a constant of 90 degrees to the training labels for both latitude and longitude. The figures below show the log-likelihood estimates overall the parameter lambda for latitude and longitude respectively.



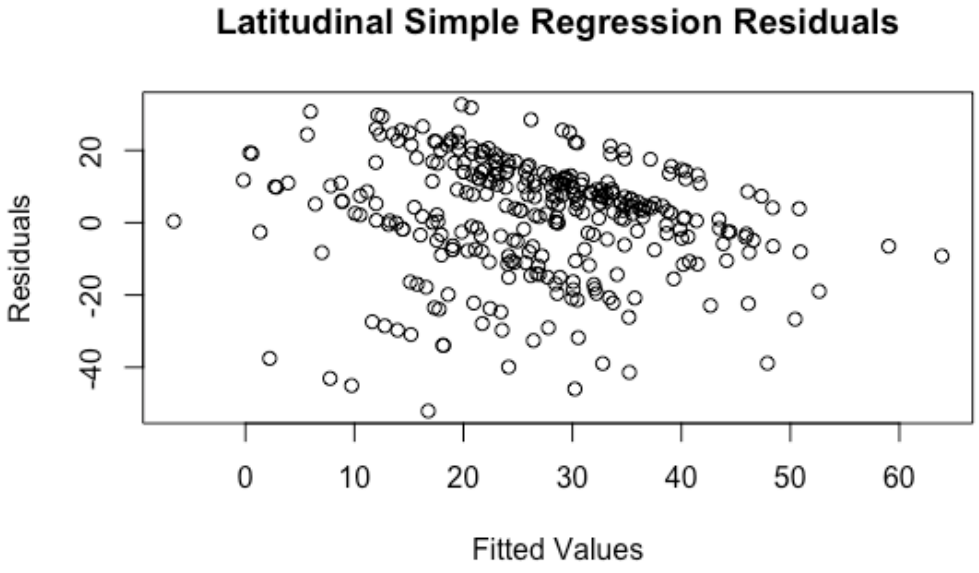
The maximum likelihood estimate of lambda was 3.315203 for latitudinal data.

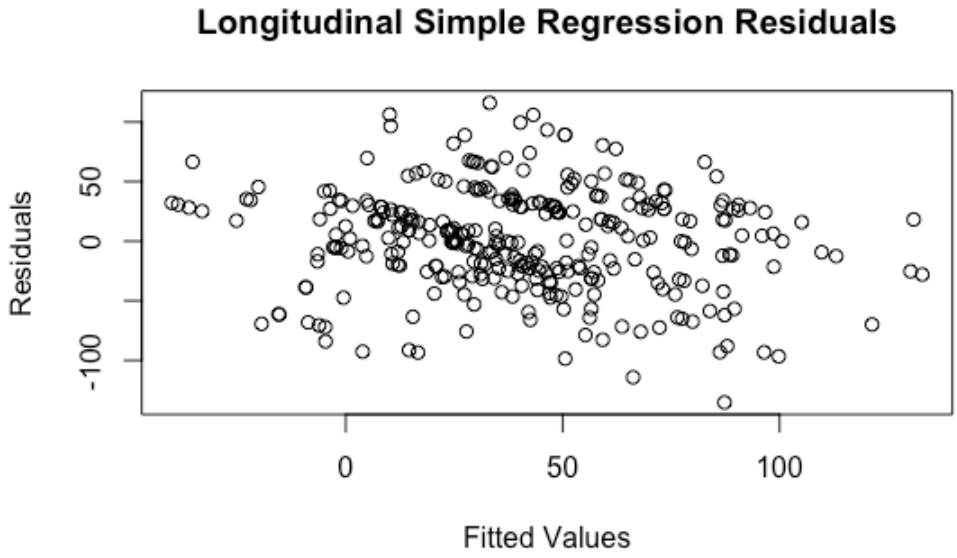


The maximum likelihood estimate for lambda was 0.9024968 for longitudinal data.

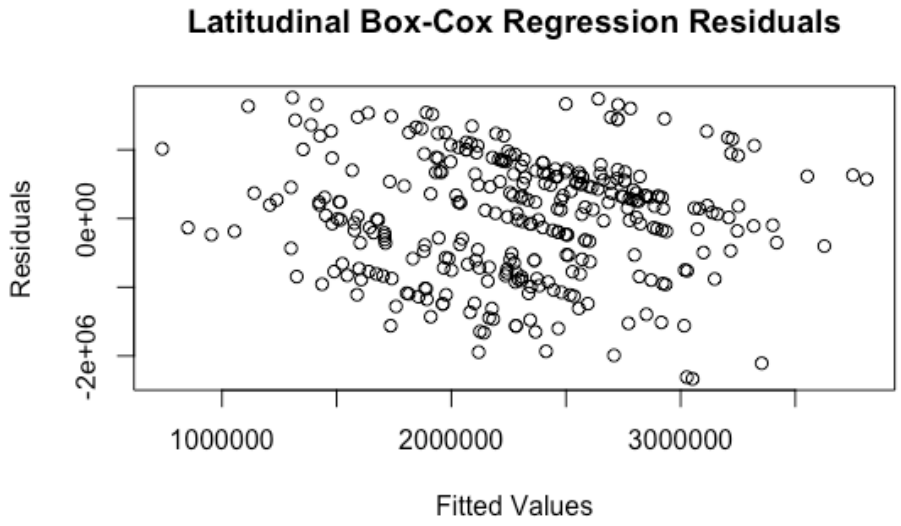
	Latitude Multiple R-Squared	Latitude Adjusted R-Squared	Longitude Multiple R-Squared	Longitude Adjusted R-Squared
No Box-Cox	0.3459	0.2754	0.371	0.3032
Box-Cox	0.373	0.3055	0.3704	0.3025

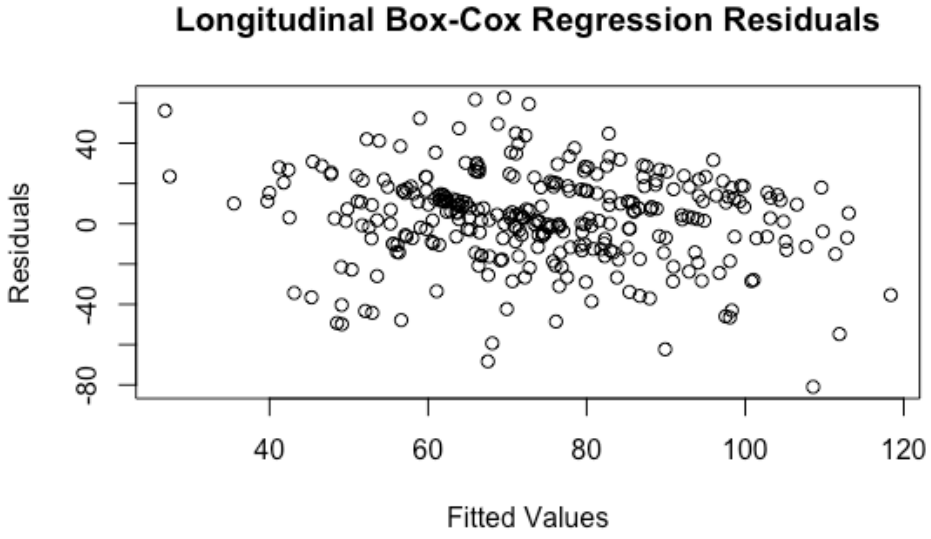
Box-cox does not seem to have a significant impact on the dataset. The R-squared values remain relatively unchanged performing better for latitude prediction, but worse for longitude predictions.





The residual plot for simple regression is approximately 0 mean with no obvious curvature, suggesting that polynomial transformation might not be effective.





The reason why the box-cox latitudinal residuals were orders of magnitude greater than the longitudinal residuals was due to the fact that lambda was three times greater in the latitudinal case. All residuals were raised threefold.

Linear Regression With Different Regularization Schemes

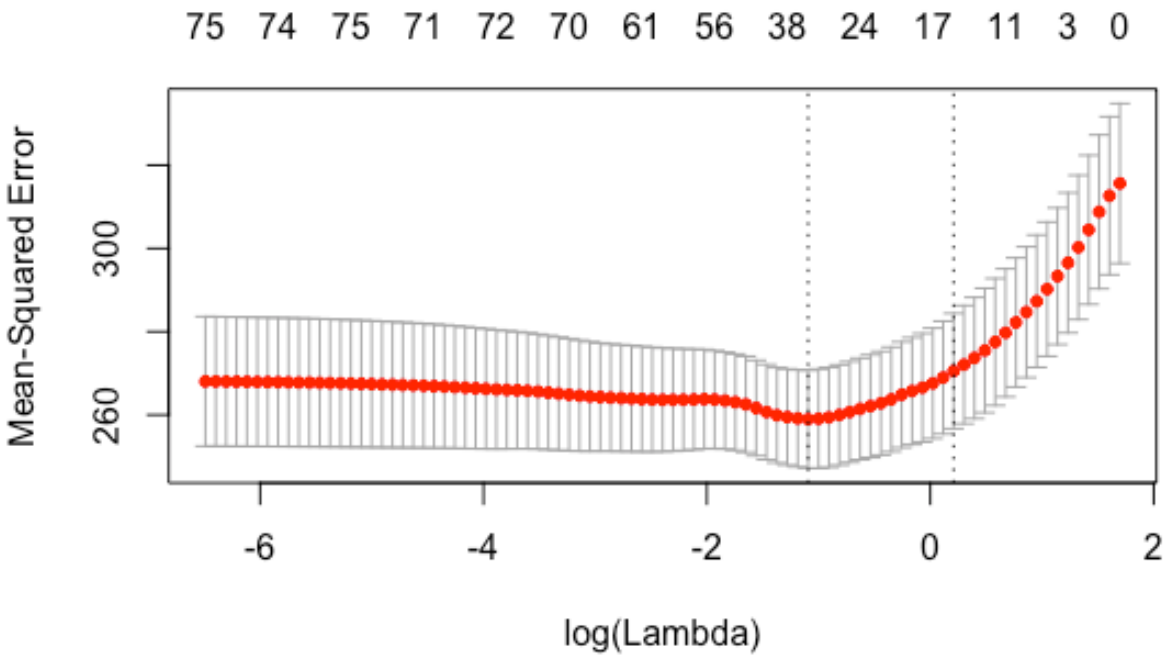
CV-Tuning for Linear Regression

In order to tune for the regularization parameter, I first split the data into a 70%-30% train-test ratio. I then ran the command:

```
latLasso = cv.glmnet(featuresTrain, latitudeTrain, alpha = 1)
```

where featuresTrain were the features of the training set and latitudeTrain were the labels of the training set. This command performs cross-validation on the parameter lambda over approximately 10^-6 to 10^-2 in logarithmic increments. The figure below shows a plot of the mean-squared error over different regularization parameters for the lasso linear regression model on latitude training data using the command:

```
plot(latLasso)
```



The first vertical line refers to the regularization parameter that corresponds to the minimum mean-squared error. The numbers at the top are the approximate number of coefficients for a given lambda. I chose the regularization parameter that corresponded to the minimum mean squared error to predict the test estimates.

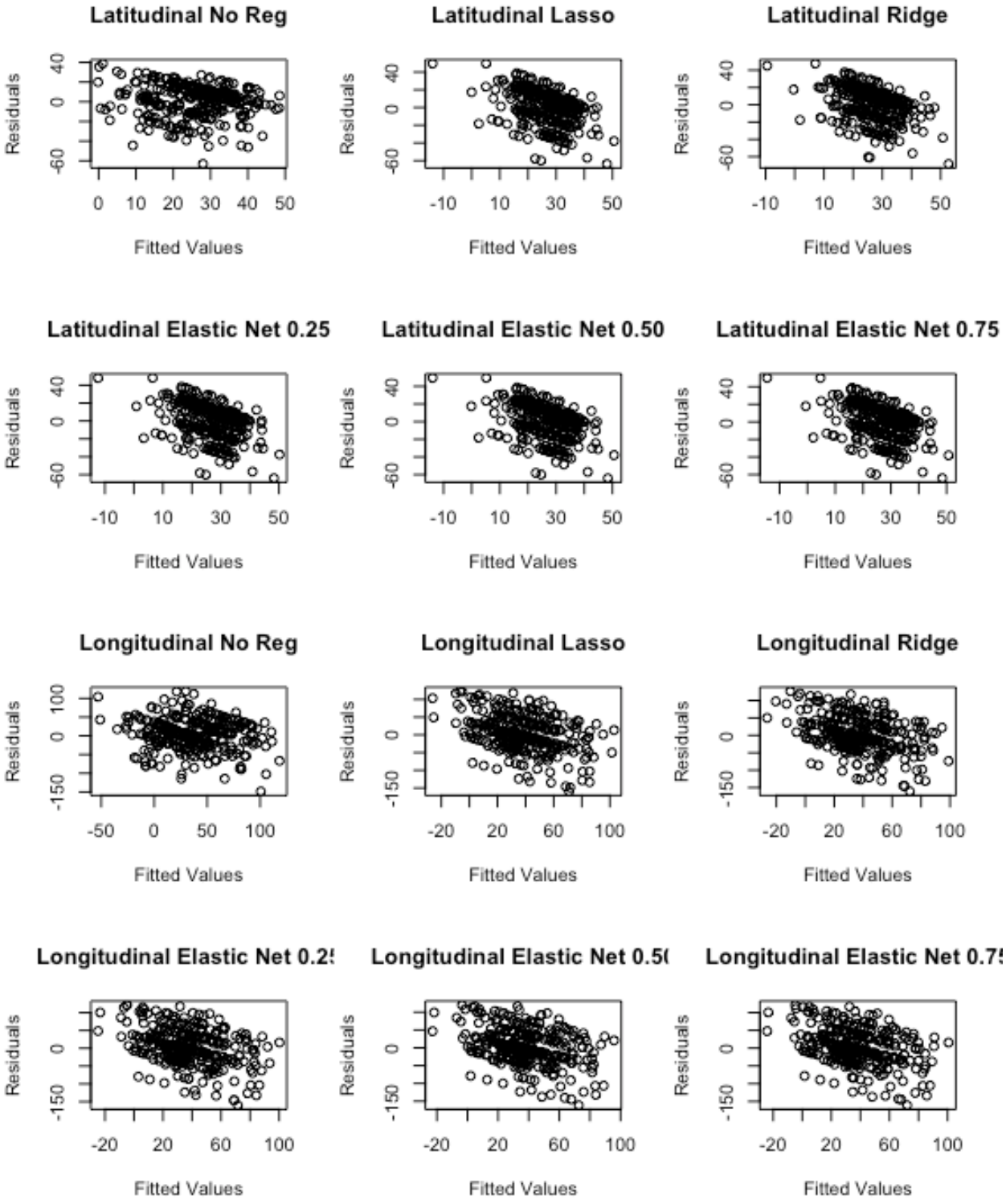
Results

Latitude	Regularization Parameter	Number of Coefficients (including intercepts)	Mean-Squared Error
No Regularization	0	117	319.1369
Lasso	0.7968123	19	290.7867

Ridge	5.753629	117	296.4837
Elastic Net (alpha = 0.25)	1.379684	42	291.7504
Elastic Net (alpha = 0.50)	0.8309172	39	292.141
Elastic Net (alpha = 0.75)	0.5539448	38	292.7894

Longitude	Regularization Parameter	Number of Coefficients (including intercepts)	Mean-Squared Error
No Regularization	0	117	1915.88
Lasso	1.05	52	1866.938
Ridge	10.99773	117	1843.917
Elastic Net (alpha = 0.25)	2.637186	90	1860.003
Elastic Net (alpha = 0.50)	2.099573	78	1869.003
Elastic Net (alpha = 0.75)	1.162068	70	1865.613

Lasso significantly reduced the number of coefficients from 117 to 19 for latitude and 117 to 52 for longitude, while also reducing the overall mean-squared error for both longitude and latitude in comparison to the non-regularized linear regression model. The mean-squared error between lasso and ridge regression did not show significant percentage differences relatively, but lasso regression was superior in that the number of parameters the model used was far less than ridge. For the elasticnet regression, the three values of alpha did not show an overall trend on the mean-squared error. The only significant difference between the elasticnet models was that higher alphas corresponded to sparser coefficient vectors. Of the three elasticnet models, the model in which alpha equals 0.75 is preferred since it is the sparsest without increasing the mean-square error significantly.



The residuals versus fitted value plot shows 0 mean residuals that are reasonably dispersed.

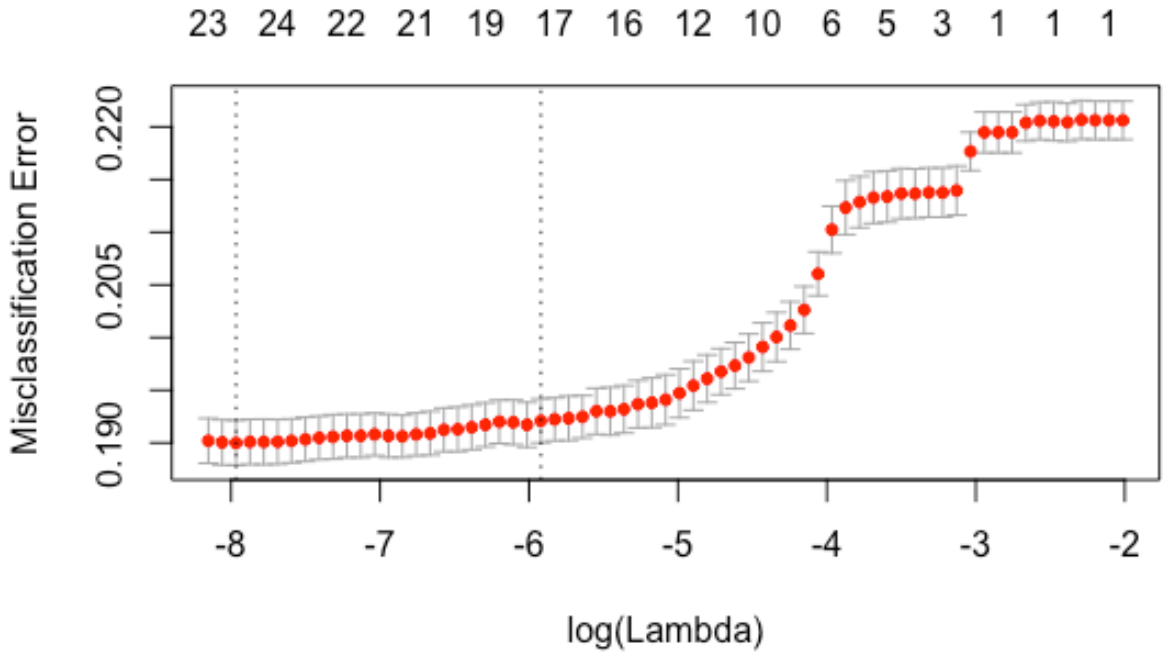
Logistic Regression With Different Regularization Schemes

CV-Tuning for Logistic Regression
To tune the regularization parameter for logistic regression, I first split the dataset into a 70%-30% train-test ratio. I ran the command:

```
lasso = cv.glmnet(xTrain, yTrain, family = "binomial", type.measure = "class", alpha = 1)
```

This command ran cross-validation on the training data using misclassification error as the evaluation metric. The figure below shows the result of cross-validation on the lasso linear regression model on the training data using the command:

```
plot(lasso)
```



The values of lambda that was chosen from cross-validation was the lambda corresponding to the minimum misclassification error shown as the leftmost vertical line in the graph.

Results

	Regularization Parameter	Number of Coefficients	Accuracy (%)
Simple	0	24	0.812
Lasso	0.000458504	25	0.8114444
Ridge	0.01466877	25	0.8073333
ElasticNet (alpha = 0.25)	0.0009562575	25	0.8118889
ElasticNet (alpha = 0.50)	0.001212231	24	0.8112222
ElasticNet (alpha = 0.75)	0.0006709434	24	0.8113333

For every regularization scheme, the regularization parameter was relatively small (about 10^-1 to 10^-3) suggesting that the supervised training model does not benefit significantly from the inclusion of regularization. Of the 24 features within the dataset, lasso and some elastic-net models were only able to reduce the number of features by one variable, which suggest that the 24 features were relatively non-redundant. Since regularization did not make the training model significantly simpler and reduced the non-regularization model's accuracy, the non-regularized model is preferred when training on this dataset.