ST443 - GROUP PROJECT

Candidate Numbers: 26527, 22399, 24738, 33210, 35138

Part 1 - Real world Data

Classification

1. Data description

We use the sample of the census income dataset that consists of 30,572 observations of data. The features of the dataset include age (AGE), working class (WORKCLASS), final weight (FNLW), level of education (EDUCATION), marital status (MARITAL_STATUS), occupation (OCCUPATION), relationship status (RELATIONSHIP), race (RACE), sex (SEX), capital gain (CAPITAL_GAIN), capital loss (CAPITAL_LOSS), working hours per week (HOURS_PER_WEEK), nationality (COUNTRY) and income (INCOME)
Dataset: https://archive.ics.uci.edu/ml/datasets/census+incom

2. Research Question

The UN is looking for additional funding sources to support its project of revitalising third-world countries. As a result, the UN plans to impose an additional tax of 0.2% on people earning an income greater than $50000 per year and is therefore looking to identify the best model to classify all the people based on the past dataset.

3. Preparation of Data (Data Wrangling)

3.1 Cleaning the data

The dataset contained just one numerical feature, fnlwgt, which gives each sample member a weight to account for the unequal probability of selection. As a result, we remove outliers from the dataset to improve model fit, then we remove all missing values, and lastly, we convert all categorical features into factors to create dummy variables.

3.2 Exploratory Data Analysis

- Plotting the histogram for the various predictors shows that our target variable is severely skewed, with three times as many people with annual incomes under $50,000 as there are those with higher incomes. (Figure X)
- There are twice as many observations from men as there are from women. (Figure X)

3.3 Model Selection

For model selection, we use logistic regression to understand the significance of each variable, furthermore, we use the Akaike information criterion measure and misclassification error rate to check for the significance as well. After running the analysis, we noticed that including all the predictors produced a lower AIC value [1] and misclassification error rate [2] when omitting certain predictors, thus all the predictors are preferred.

4. Approaches used for classification.

4.1 K Nearest Neighbors

K Nearest Neighbors approach is applied to the model, cross-validation is used to obtain the optimal value of K, 29 in our case and then we train the model respectively, the overall accuracy of the model [3] and area under the curve (AUC) on the test set are 79.25% and 0.5918 respectively.

[1] $AIC = 2k - 2\ln(\hat{L})$, [2] $Misclassification\ rate = \frac{False\ positives + False\ negatives}{Total\ observations}$,

[3] $Model\ accuracy = \frac{True\ positives + True\ negatives}{Total\ observations}$

4.2 Multivariate Logistic Regression
Multivariate Logistic regression predicts the probability using a sigmoid function based on a set of independent variables, the overall accuracy of the model is 84.37%. AUC is 0.8999.

4.3 Linear Discriminant Analysis
Linear Discriminant Analysis is applied to our model, the overall accuracy and the AUC is 83.32% and 0.8856 respectively.

4.4 Linear Discriminant Analysis
Linear Discriminant Analysis is applied to our model, the overall accuracy and the AUC is 83.32% and 0.8856 respectively.

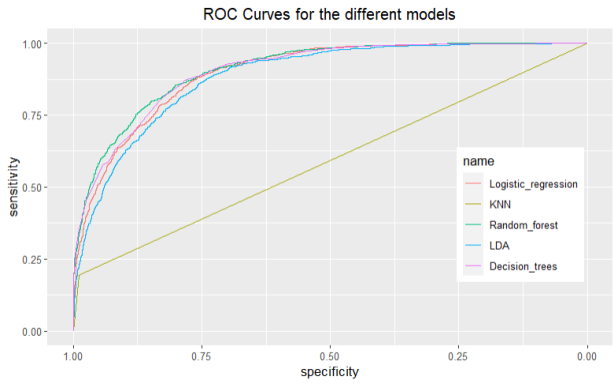
4.5 Decision trees
A classification tree with 78 inner nodes was used in our model to classify the target variables from the test dataset, the overall accuracy and AUC were 84.37% and 0.9052 respectively.

4.6 Random Forest
Finally, the random forest model with parameter ntree = 100 due to computational complexity was used to train the model, the overall accuracy and AUC of the model were 85.75% and 0.9111 respectively.

5. Comparing different classification models

Model	AUC	Model accuracy	Computation Time
K Nearest Neighbours	0.5918	79.25%	0.120641 secs
Logistic Regression	0.8999	84.37%	3.608204 secs
Linear Discriminant Analysis	0.8856	83.32%	1.036161 secs
Decision Trees	0.9052	84.37%	1.261948 secs
Random Forest	0.9111	85.75%	3.291192 mins

Table 1



Plot 1

Based on the criteria of area under the curve and overall model accuracy, Random Forest is the best-performing model. Plot 1 illustrates the ROC curve for different models and verifies that Random Forest is the best.

6. Results and Conclusion:
The random forest model was proposed to the UN to categorise income. Nevertheless, there are computational costs associated with this, in our instance, training the model took 3.29 minutes. Decision trees emerged as the second-best model, with an overall model accuracy and AUC of 85.75% and 0.9052, respectively. When compared to random forest, the computation cost in this instance is significantly lower at 1.27 seconds.

Regression

1. Data description

Our regression research explores the US Census 2017 Income Estimation dataset sourced from Kaggle. This dataset comprises 75,000 tractids (sub-counties) and their demographic statistics of 37 features: TractId, State, County, TotalPop, Men, Women, Hispanic, White, Black, Native, Asian, Pacific, VotingAgeCitizen, Income, IncomeErr, IncomePerCap, IncomePerCapErr, Poverty, ChildPoverty, Professional, Service, Office, Construction, Production, Drive, Carpool, Transit, Walk, OtherTransp, WorkAtHome, MeanCommute, Employed, PrivateWork, PublicWork, SelfEmployed, FamilyWork, Unemployment.
Dataset: US Census Demographic Data

2. Research Question

To further optimize resource allocation, the US government is exploring income disparities within different census tracts. This project assists the government in identifying optimal regression techniques for predicting income levels. By analysing demographic features using past census data, our goal is to enhance decision-making strategies.

3. Preparation of Data

3. 1Cleaning the data

For the missing values in our dataset, we utilize mean imputation to replace missing numeric values. We then drop the categorical columns "TractId", "State", and "County" as these columns act as an index for identification purposes.

3.2 Exploratory Data Analysis

- To better understand the correlation of our variables, we plot a heatmap (Plot 4). From this, we see trends, such as gender having a strong positive correlation and public vs private work having a strong negative correlation.

- We then employ exploratory data analytics methods to create histograms (Plot 5). The histogram below shows that our response variable, income, is positively skewed and applicable for log-linear regression.

4. Approaches and Results

4.1 Linear Regression:

Linear Regression is a simple approach that establishes a linear relationship between the predictors and our response variable, income.

4.2 Log-Linear Regression:

Log-linear regression is typically used when the response variable is skewed. A logarithmic approach ensures that the model follows a more symmetric distribution. As seen in the histogram above, our response variable is positively skewed.

4.3 Lasso Regression:

Lasso Regression shrinks the coefficient estimates towards zero. L1 penalty has the effect of forcing some of the coefficients to be exactly zero when a large λ is used. L1 regularization is a form of variable selection, allowing irrelevant features to be dismissed.

4.4 Ridge Regression

Ridge Regression is motivated by removing the multicollinearity effect. It still includes all predictors in the model, performing no form of variable selection. Lasso Regression and Ridge Regression prevent overfitting and offer flexibility due to regularization.

4.5 Generalised Additive Models

The generalized additive model allows for flexible nonlinearities in several variables in an additive structure of linear models. We further used the smoothing spline to better fit our model.

4.6 Random Forest Regressor

Random Forest Regressor encapsulates the output of myriad decision trees into a single result. This model aims to minimize overfitting while still having high accuracy.

5. Comparing different Regression methods:

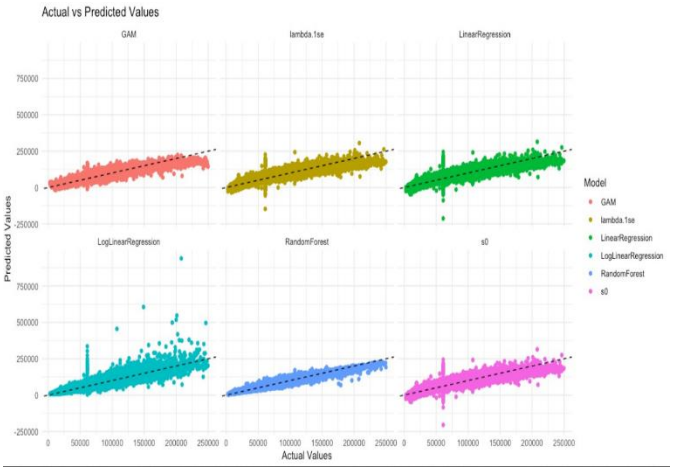
Model	RMSE	Computation Time
Linear Regression	12451.06	0.1188822 secs
Log-Linear Regression	12225.35	0.2493219 secs
Ridge Regression	12448.40	2.450966 secs
Lasso Regression	12443.71	2.455499 secs
GAM	10545.32	41.31907 secs
Random Forest	10075.16	5.17231 mins

Table 2

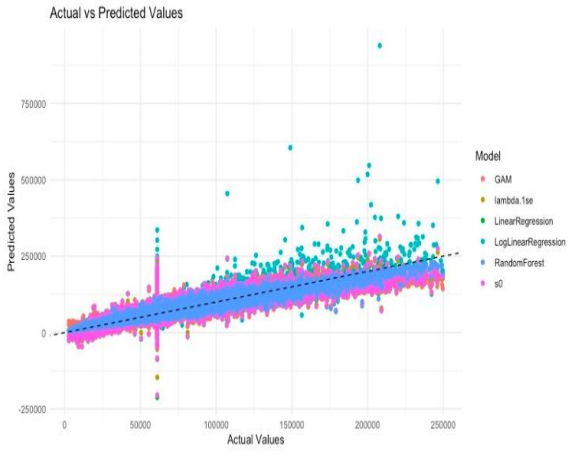
The Random Forest Regressor has the lowest RMSE among the other models, thus indicating a better predictive accuracy. However, it has a much higher computation time. GAM has a higher RMSE compared to Random Forest. Unlike Random Forest, it deems a suitable balance between computation time and RMSE. Linear, Log-Linear, Ridge, and Lasso regression are simpler models that encompass similar RMSE values and have lower computation times.

6. Results and conclusion

In Plot 2, we have plotted the actual vs predicted data against all our models. Of all the plotted models, the results of our Random Forest model tightly follow the trend of our predicted values. It is insensitive to outliers, unlike other models. Plot 3 overlaps all the models. We further see that Random Forest best fits our data with less variance in comparison to the other models. In conclusion, if there is no time complexity constraint, Random Forest achieves the best predictive capabilities. Otherwise, GAM is also a suitable option as well.



Plot 2



Plot 3

Part 2: Coordinate descent algorithm for solving the lasso problems.

This part of the report describes the methodology and evidence of the performance of the two types of penalised regression models. It focuses on the application of coordinate decent type algorithm on lasso and elastic net. We would first create the general methods to run the function to find the optimum value of λ for each approach based on some assumptions. This λ is then used to fit the model on the generated data to assess the model performance of the two methods, i.e., the lasso and elastic net. Further, we create various simulations based on different settings like number of observations, number of variables and the sparsity structure.

1. Introduction to the Lasso Approach

In the general linear regression model, the estimated $\boldsymbol{\beta}$ coefficients, or $\widehat{\boldsymbol{\beta}}$, are non-zero. To obtain a sparser model, i.e., a model with lesser non-zero $\boldsymbol{\beta}$ coefficients, we use l_1 penalty to penalise for the model complexity. This penalty would force some of the insignificant coefficients to become exactly equal to 0. This l_1 penalty is applied to the objective function as shown below:

$$\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

where $\lambda \geq 0$ is the regularization parameter. This $\boldsymbol{\lambda}$ penalises the model for the model complexity. The first section of the above expression focusses on improving the model fit by fitting the model that minimises the above expression. The second section penalises for model complexity, by adding a non-zero value to the objective function for every $\boldsymbol{\beta}$ as a multiple of $\boldsymbol{\lambda}$. This also control the bias and variance trade off. The lasso can perform variable selection and choose a sparse model containing a subset of all p predictors. A sparser model would increase the bias of the model and reduce the variance of the model.

Using the above objective function, if we differentiate it w.r.t. $\boldsymbol{\beta_j}$ we get the below estimate of soft thresholding:

$$\widehat{\beta_j} = sign(\beta_j^*)max(0, |\beta_j^*| - \lambda)$$

Lasso regression also works good in the case when we have multiple x_i correlated with one another. This method then only selects one variable from the group and ignores the others. Although lasso is shown to be advantageous in many situations, it has some limitations. We would be looking at the following two scenarios:

1. In the case where $p > n$
2. In the case where $n > p$

To overcome the limitations of the Lasso method, we will look at the enhanced version of the Lasso method by adding another term with the penalty of the form $\lambda_2 \sum_{j=1}^{p}\beta_j^2$.

2. Introduction to Net Elastic Approach

The objective function is similar to the one described above with an addition of the term. The generalised objective function is given by:

$$\frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p}x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| + \lambda_2\sum_{j=1}^{p}\beta_j^2$$

Here, we have two tuning parameters, $\boldsymbol{\lambda_1}$ and $\boldsymbol{\lambda_2}$.

In order to reduce the computational complexity and to plot the cross-validation error against lambda, we have assumed that $\lambda_1 = \lambda\alpha$ and $\lambda_2 = \lambda(1-\alpha)$. Using these two values instead of λ_1 and λ_2, would require us to re-derive the soft-thresholding formula based on the below objective function:

$$\arg\min_{\beta} \frac{1}{2n}\left|\left|Y - X\beta\right|\right|_2^2 + \lambda\left[\frac{(1-\alpha)1}{2}\left|\left|\beta\right|\right|_2^2 + \alpha\left|\left|\beta\right|\right|\right]$$

Minimising the sum of the components would be equal to minimising the components itself:

$$\frac{\left(Z - \beta_j\right)^2}{2} + \lambda\left[\frac{(1-\alpha){\beta_j}^2}{2} + \alpha\left|\beta_j\right|\right]$$

Differentiating w.r.t. β_j we get:

$$-Z + \beta_j + \lambda\left[(1-\alpha)\beta_j + \alpha\, sign\left(\beta_j\right)\right] = 0$$

Thus, the estimate of the β_j value would be given by:

$$\beta_j = \frac{Z - \lambda\alpha sign\left(\beta_j\right)}{1 + \lambda(1-\alpha)}$$

After considering the various cases, the soft thresholding formula would be given by:

$$\beta_j = \frac{sign\left(\beta_j\right)\max\left(0, \left(\left|\beta_j\right| - \lambda\alpha\right)\right)}{1 + \lambda(1-\alpha)}$$

In this study, we have varied the values of α over the range of (0,1). Using the combination of both α and λ we get the best combination of tuning parameters.

3. Selecting optimal tuning parameter

Due to computational cost, one optimal value of $\boldsymbol{\lambda}$ is not feasible to be found for each regression, hence we choose an optimal value of $\boldsymbol{\lambda}$ based on all regressions possible. The optimal value of $\boldsymbol{\lambda}$ is chosen based on the prediction accuracy of the regression model, which is measured by the five-fold cross-validation mean squared error.

Under each of the methods of Lasso and Elastic net, the following approaches have been used to find the most suitable hyperparameter values.

In Lasso: We use cross validation to find the most suitable value of $\boldsymbol{\lambda}$. For each lambda, we calculate the mean squared error using 5-fold cross-validation. We organise these errors into a matrix, where rows represent the folds and columns represent the tuning parameter, $\boldsymbol{\lambda}$**.** Each element of that matrix is the training mean square error in that fold for the specific values of lambda. The lambda with the smallest mean error folds is considered the ***optimal_lambda*** value. Further, we obtain the one standard error by adding the standard error of the optimal lambda with optimal lambda. The highest λ where the MSE is equal to the one standard error is taken as the ***one standard error lambda***.

In Elastic Net: We adopt a similar approach to our previous method, with the sole distinction being the introduction of a pair of tuning parameters. Consequently, we record the outcomes in a dataframe instead of a matrix. Then we identify the combination of tuning parameters that yields the lowest average cross-validation error and determine the lambda value corresponding to one standard error.

4. Methodology

This section of the report describes the method followed in fitting the coordinate descent algorithm for both lasso and elastic net. The data has been split into training and testing data based on a ratio of 80:20, this is done to ensure that the final estimates of the mean square error on the test data is on an unseen data. This is done using the *sample* function in R. The 80% of the data is used to find the optimum value of $\boldsymbol{\lambda}$. Further, this $\lambda_{optimal}$ is used to fit the obtain the $\boldsymbol{\beta_j}$ values. Rest 20% of the data is used to predict the test errors.

One important characteristic that we have ensured for the generated data is the "orthonormality". It is important to ensure that the generated data satisfies the below two properties:

- Orthogonality: Under this, the generated sample of X satisfies => $X^T X = I_n$
- Normality: Under this, $XX^* = X^*X$ where X^* is the conjugate transpose.

Since the data generated from a normal distribution with variance 1, we just need to ensure that the data is orthonormal.

4.1 Base Method

The general method followed under this report is as follows:

- Generating data: It is given to assume that the *p* predictors and responses are generated from a $N(0,1)$. According to the sample, it is assumed that each variable, i.e., all X_i's is obtained from a $N(0,1)$ population such that x_i and x_j are correlated with the value of $0.5^{|i-j|}$. The data is generated using the *mvrnorm* function which takes the input of the covariance matrix to generate the same. One of the key points to note is that the derivations of the estimates of β coefficients is done under the assumption that the sample generated, i.e., the X matrix of the sample is orthonormal, i.e., $X^T X = I$.
  Since, the sample is generated from normal distribution with μ=0 and σ_x = 1, the correlation is equal to the variance based on the expression below.
  $$cor(x_i, x_j) = \frac{cov(x_i, x_j)}{\sqrt{V(x_i) * V(x_j)}}, V(x_i) = 1 \; and \; V(x_j) = 1$$
- To generate the orthonormal sample value of x_i's from the normal distribution with the given covariance matrix, we use the *mvrnorm* function in R. This generated data is further treated such that it is orthonormal.
- Generating the sample observations of y_i's with the help of the model equation:
  $$Y = X\beta + \sigma\epsilon, where \; \epsilon \sim N(0, I_n)$$
- Defining a set of $\boldsymbol{\beta_j}$ vector as the starting point of the estimates of the coefficients.
- Using this data and the above initial $\boldsymbol{\beta_j}$ coefficients, we recursively obtain the $\boldsymbol{\beta_j^*}$ for various combinations of the tuning parameters with the help of the soft thresholding formula derived above. The $\boldsymbol{\beta_j}$ values are recursively updated until convergence is reached.
- We have also applied the cross-validation approach on the training data to choose the optimum value of the tuning parameter(s) based on the least (training) mean square error.
- For the optimum value of $\boldsymbol{\lambda}$ we have then estimated the test mean square error to compare the two approaches.

4.2 Special Cases

Based on the above method, we can see that there are various assumptions that can be varied over a range of different values to see the impact of them on the model performance. This would also be done

for both the cases of $n > p$ and $n < p$. The following specific scenarios have been taken to study their impact on model performance:

1 High sparsity: increasing the sparsity means setting the initial $\boldsymbol{\beta}$ values which have more zero values.
2 Low sparsity: reducing the sparsity means setting the initial $\boldsymbol{\beta}$ values which have more non-zero values.
3 High variance of the error: this is done by increasing the σ value.
4 Low variance of the error: this is done by reducing the σ value.

The analysis results have been discussed in the later sections of the report.

5. Lasso Problem

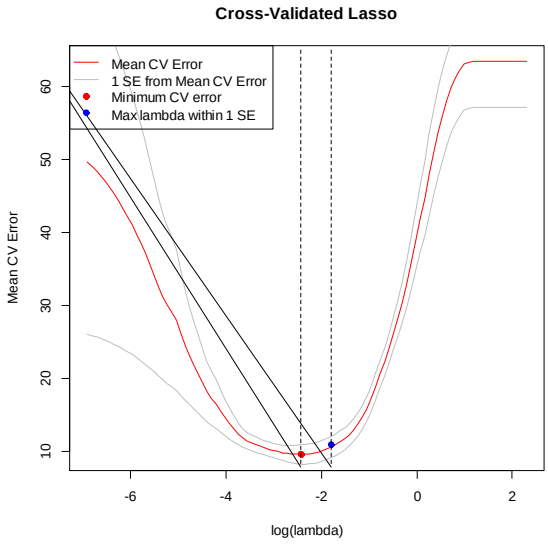
5.1 $n > p$

Under this method, the $\boldsymbol{\lambda}$ values are taken from the range of $10\text{^}seq(10, -2, length = 50)$. We use the method of choosing the best lambda value and the one standard error lambda value as described in the methodology.

The other assumption conditions are given below:

- p = 100
- n = 200
- sparsity = 0.5
- sigma = 2
- pairwise_cor = 0.5

On running the coordinate descent for Lasso, we get the following plot for the Mean Square Error against the lambda values.



The lambda value that minimises the average mean square error over all the folds is equal to 0.07220809. The lambda value that minimises the standard error of mean square error over all the folds is equal to 0. 1261857. This value of lambda gives the mean square error on the test data as: 6.158752.

For this case of $n > p$, when n = 200 and p = 100, the special cases summary is attached below:

Condition	λ_{best}	$\lambda_{1\,se}$	$MSE_{1\,se\,lambda}$
Base Parameter	0.0869749	0.1668101	6.158752
High Sparsity (sparsity = 0.8)	0.1261857	0.2205131	7.174381
Low sparsity (sparsity = 0.2)	0.03764936	0.09545485	16.50994
High σ (sigma = 3)	0.1261857	0.2656088	18.77547
Low σ (sigma = 1)	0.05462277	0.09545485	3.281401
High ρ (corr = 0.8)	0.1261857	0.1830738	6.788124
Low ρ (corr = 0.2)	0.06579332	0.1261857	4.342852

From the above table, we can see that when the variance in the error term is increased and when the sparsity is increases, the $\boldsymbol{\lambda}$ value increases. As $\boldsymbol{\beta}$ becomes more sparse (more coefficients with a valur of zero), the response y also becomes more sparse. Hence the model would require $\boldsymbol{\beta}$ coefficients to predict the values of Y. Similarly, in the case when there is higher variance of the error term in the model, this would also lead to higher variance in the sample Y values. This would mean that for more complex models (or for lower $\boldsymbol{\lambda}$ values) there would be more $\boldsymbol{\beta}$ coefficients leading to significantly higher variance in the model. Due to the one standard error rule, it would be easier to get a higher value of lambda that falls within the one-standard error of the least mean cross validation error, hence the $\boldsymbol{\lambda_{1se}}$ would also increase. When the generated sample is more correlated, then the response Y can be explained with less parameters. This is because the effect of the other eliminated variables would flow into the model through the other highly correlated coefficients.

5.2 $n < p$

With the similar analysis performed for the case of $n > p$ we change the values for $n < p$. The other assumption conditions are given below:

- p = 200
- n = 100
- sparsity = 0.5
- sigma = 2
- pairwise_cor = 0.5

The lambda value that minimises the average mean square error over all the folds is equal to 0. 6734151. The lambda value that minimises the standard error of mean square error over all the folds is equal to 3. 274549. This value of lambda gives the mean square error on the test data as: 95.57272.

For this case of $n < p$, when n = 100 and p = 200, the special cases summary is attached below:

Condition	λ_{best}	$\lambda_{1\,se}$	$MSE_{1\,se\,lambda}$
Base Parameter	0.6734151	3.274549	95.57272
High Sparsity (sparsity = 0.8)	0.2420128	0.6135907	20.69394
Low sparsity (sparsity = 0.2)	0.4641589	1.417474	143.1583
High σ (sigma = 3)	0.6135907	3.593814	100.5256
Low σ (sigma = 1)	0.5094138	2.477076	80.11201
Low ρ (corr = 0.8)	0.4229243	0.6734151	52.30475
Low ρ (corr = 0.3)	0.559081	0.97701	54.16675

It is clearly evident that the model performs very poorly for a high dimensional data. Since we have more predictors than the sample size, the lasso coordinate descent performs very poorly. This is

validated by the test mean square error. Furthermore, even the general effect of changes in the special conditions on the $\boldsymbol{\lambda}$ values have changed. This indicates a bad performance of lasso model in the case of $n < p$.

6. Elastic Net

6.1 $n > p$

Under this case, we have two tuning parameters, $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$.

The values of $\boldsymbol{\lambda}$ are taken from $10\text{^}seq(10, -2, length = 50)$ while the α values would be taken from $(0,1)$. Taking $\alpha = 0$ means that we have a ridge regression while $\alpha = 1$ means that we have a lasso regression. If the best $\boldsymbol{\alpha}$ value is near 0.5, the results would show that elastic net performs the best. We use the two approaches of choosing $\boldsymbol{\lambda}$ values, the best lambda value and the one standard error lambda value, as described in the methodology. The base assumptions set under this case are:

- p = 100
- n = 200
- sparsity = 0.5
- sigma = 2
- pairwise_cor = 0.5

On running the coordinate descent for elastic net, we get the following plot for the Mean Square Error against the lambda values.

For this case of $n > p$, when n = 200 and p = 100, the special cases summary is attached below:

Condition	$\boldsymbol{\lambda_{best}}$	$\boldsymbol{\alpha_{best}}$	$\boldsymbol{MSE_{1\ se\ lambda}}$
Base Parameter	0.04132012	0	11.66572
High Sparsity (sparsity = 0.8)	0.1261857	1	7.605814
Low sparsity (sparsity = 0.2)	0.05462277	0.3	7.51372
High σ (sigma = 3)	0.04132012	0.2	18.86401
Low σ (sigma = 1)	0.02364489	0.2	14.80373
Low ρ (corr = 0.8)	0.1047616	1	6.690615
Low ρ (corr = 0.2)	0.200923	1	9.936618

From the above table we can see that higher sparsity leads to higher value of lambda (simpler model). Additionally, higher variance also leads to a model with high lambda.

6.2 $n < p$

With the similar analysis performed for the case of $n > p$ we change the values for $n < p$. The other assumption conditions are given below:

- p = 200
- n = 100
- sparsity = 0.5
- sigma = 2
- pairwise_cor = 0.5

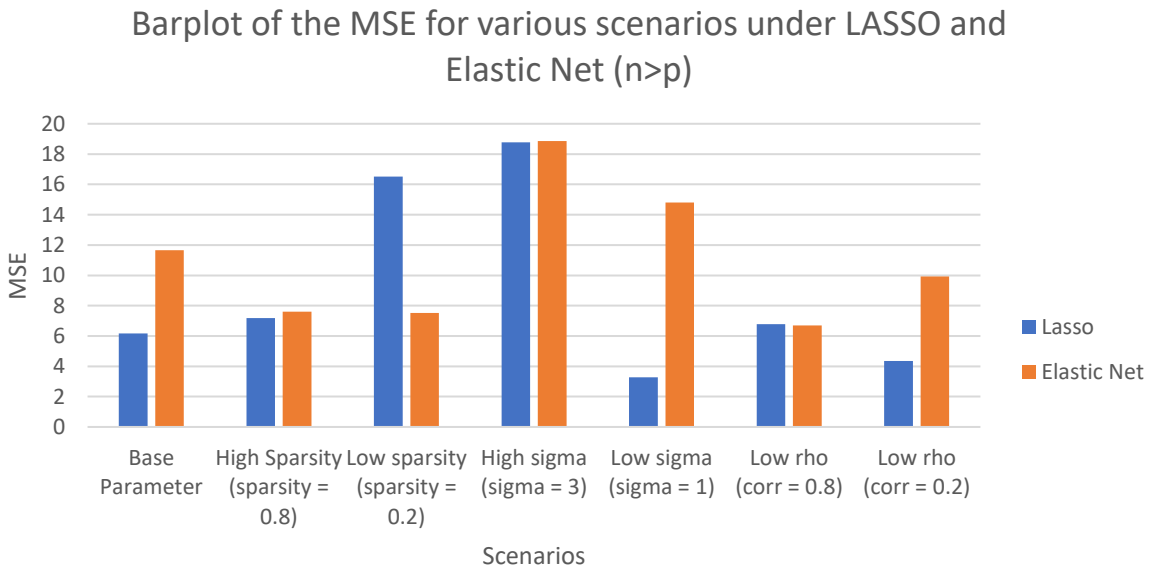
For this case of $n < p$, when n = 100 and p = 200, the special cases summary is attached below:

Condition	λ_{best}	$\lambda_{1\,se}$	$MSE_{1\,se\,lambda}$
Base Parameter	0.3199267	0.4	52.68741
High Sparsity (sparsity = 0.8)	0.2420128	0.4	14.80373
Low sparsity (sparsity = 0.2)	0.1830738	0.5	91.34037
High σ (sigma = 3)	0.265608	0.8	170.281
Low σ (sigma = 1)	0.2656088	0.3	43.14641
Low ρ (corr = 0.8)	0.2656088	0.9	$5.787e^{89}$
Low ρ (corr = 0.2)	1.29155	1	83.05499

It is clearly evident that elastic net performs significantly better than Lasso in the case when $n < p$ since the mean square errors have reduced drastically.

7. Model Comparison

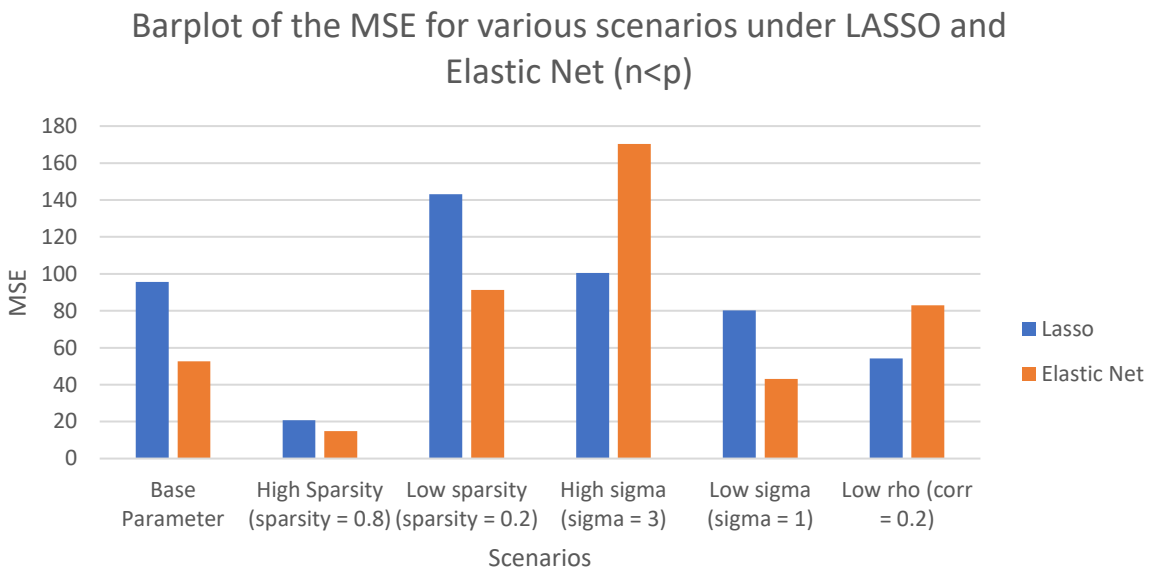
For, $n > p$



Condition	$Lasso\ MSE_{1\,se\,lambda}$	$Elastic\ Net\ MSE_{1\,se\,lambda}$
Base Parameter	6.158752	11.66572
High Sparsity (sparsity = 0.8)	7.174381	7.605814
Low sparsity (sparsity = 0.2)	16.50994	7.51372
High σ (sigma = 3)	18.77547	18.86401
Low σ (sigma = 1)	3.281401	14.80373
Low ρ (corr = 0.8)	6.788124	6.690615
Low ρ (corr = 0.2)	4.342852	9.936618

Elastic net performs better when the data is highly correlated, highly dimensional, low sigma and also with sparser beta coefficients. This can be clearly seen from the chart above as the MSE in those cases reduces significantly as compared to Lasso.

For, $n < p$



Condition	$Lasso\ MSE_{1\ se\ lambda}$	$Elastic\ Net\ MSE_{1\ se\ lambda}$
Base Parameter	95.57272	52.68741
High Sparsity (sparsity = 0.8)	20.69394	14.80373
Low sparsity (sparsity = 0.2)	143.1583	91.34037
High σ (sigma = 3)	100.5256	170.281
Low σ (sigma = 1)	80.11201	43.14641
Low ρ (corr = 0.8)	52.30475	$5.787e^{89}$
Low ρ (corr = 0.2)	54.16675	83.05499

On comparing the performance of lasso and elastic net, we can see that elastic net performs better in both the cases of $n > p$ and $n < p$.

For the case of $n < p$, i.e., high dimensional data, elastic net performs the best. The inability of the lasso coordinate descent function to handle the high dimensional data is one the reasons why elastic net was constructed. With this analysis, we can prove that elastic net gives the predictions with least mean square errors.

8. Results and Conclusion

We can see that lasso is better with handling multicollinearity as it removes the irrelevant predictors from the model and makes it more sparse and easier to understand. It keeps only one variable in the model and removes the others. Whereas, elastic net selects groups of correlated variables together instead of just one variable when the predictors are correlated. Elastic net groups of these correlated variables and then shrinks them too.

As shown with the above results, elastic net is proven to be more robust and stable. It overcomes the limitation of lasso when $n < p$ and performs very well as shown with the report.

9. Strengths and Limitations of Study

9.1 Strengths of the Study

- The algorithm performs variable selection.
- The analysis ensures that the generated sample is orthonormal which forms a key assumption for the coordinate descent for both lasso and elastic net.
- Appropriate plots have been added into the report to make the trends interpretable.
- The study includes the different scenarios of high sparsity, low sparsity, high correlation, low correlation, etc.
- The results have been run and tabulated for further inference and understanding.

9.2 Limitations of the Methodology in the context of Data

- We have only considered one case for each of the scenarios $n > p$ and $n < p$ and not varied the values of n and p for both the cases.
- The algorithms developed has a high time and space complexity.
- The model has been run once with one seed value. Changing that value and aggregating the results would give better understanding of the better model.
- Lasso algorithm is generally very volatile to the magnitude of the data which forms a limitation of this method.

10. Contributions

26527 – 20%

22399 – 20%

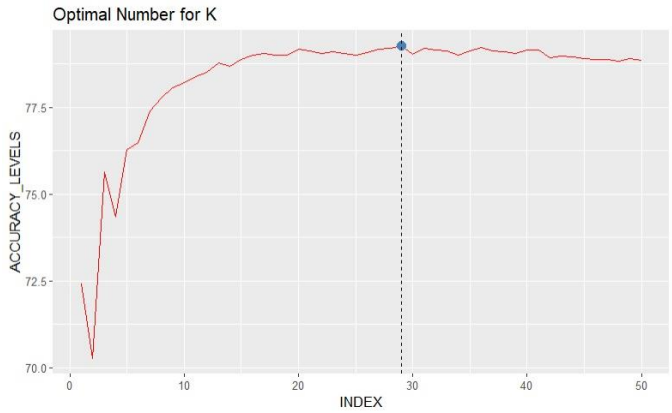
24738 - 20%

33210 - 20%

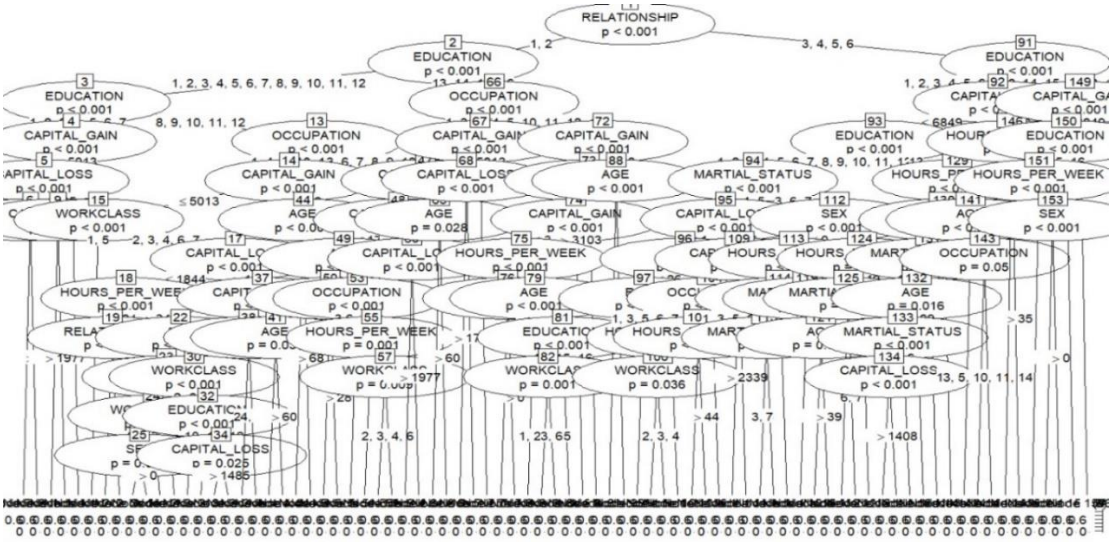
35138 - 20%

11. Appendix

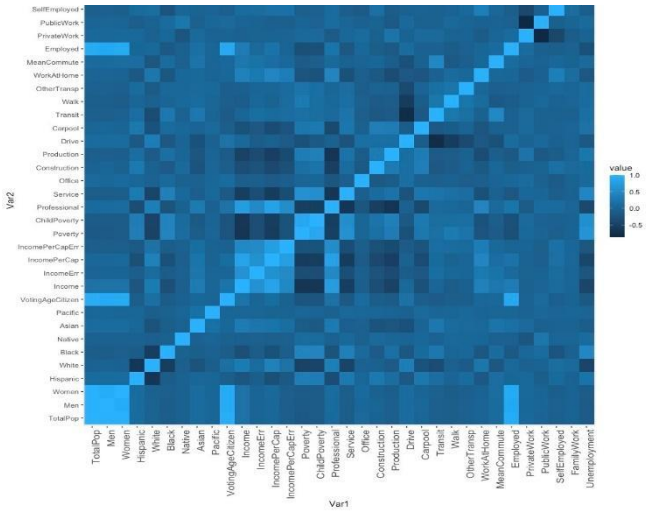
Plot 4



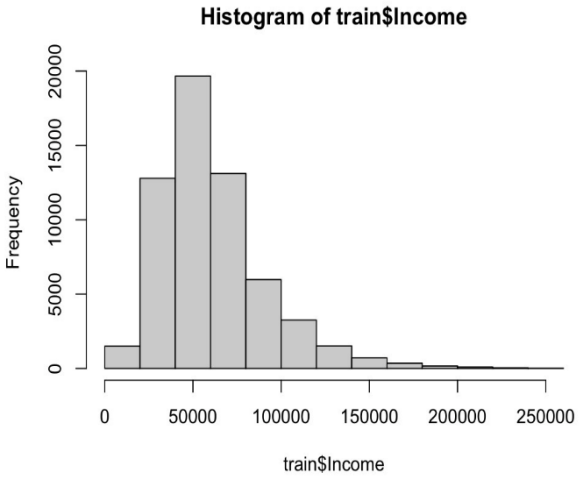
Plot 5



Plot 6



Plot 7



Plot 8

Plot 4

Plot 5