Part I: Machine Learning Methods on Real Data

1.1. Introduction

While a lot of conventional research has been conducted in the global setting that compares economic growth across countries, we hope to see if machine learning models can exploit empirical research findings to predict regional economic growth in the medium term. Recognising that social capital has a determinant effect on economic performance, we hypothesise that machine learning models could utilise social development indicators to predict economic growth (Iyer et al, 2005). Namely, we like to determine whether machine-learning methods could employ social development indicators as predictors for regional economic growth in the US. In this research, we assumed the inelasticity of social capital that a cross-sectional observation of social development is representative of the study period.

1.2. Data and Methodology

We collected data from the US Census Bureau for the economic and social indicators on the Census Tract Level made available from the American Community Survey 5-Year Data (ACS5). To avoid the complication of considering the effect of COVID-19, we used the 2014 and 2019 editions of the data. We retrieved the data by running API requests through an R session. Due to the computational demand, we restricted our research to the state of New York, which contains 4918 unique data points.

To measure the economic change, we retrieved the median income for both 2014 and 2019 and computed the percentage change over the time period. In addition, we created a new variable to classify a census tract based on the direction of change in median income. With this approach, we yield a dataset with a significant class imbalance where less than 20 percent of the census tracts recorded a decrease in median income. For the predictors, we identified opportunistically and obtained socio-economic indicators as measures of social capital, including education attainment and unemployment rate (*figure I in Appendix*). For observations with an "insufficient number of sample observations" for an estimated value, we imputed a value of 0 to preserve the dimension of the dataset. Filtering out data points with missing values, 134 census tracts were dropped. A preliminary exploration of the correlation heat map (*figure II in Appendix)* showed no strong correlation between any socioeconomic variables and a change in income.

The dataset was split into three sets: the training, testing, and validation set. We used the training data to fit the models for variable selection and hyperparameter tuning. The test data is used to compare and evaluate the models. Lastly, the validation set was used for the performance of the selected best model.

We first ran a simple multiple linear regression (MLR) model to help us understand the interaction between the predictors and the response variable. Assuming that the first model would have a high variance, we then proceeded to use the regularisation method using the L1 loss function, best subset selection and principal component analysis to compute three regression models. We also explore the use of neural nets for prediction. Based on the selected performance metrics, we identified the best training method and employed the best model to predict the change in median income on the validation set.

We then simplified the research question to a classification problem to see if the performance of machine learning methods would be better. We considered five approaches, Logistics regression, LDA, QDA, *k*-nearest neighbours classification and lastly tree-based method. We considered both the accuracy and the sensitivity-specificity trade-off to identify the best model and used it to predict the validation set.

1.3. Results

3.1 Regression Analysis

From the preliminary results on the training data where all predictors were used to train a MLR model, we observed some association between the predictors and the response variable. Most of the predictors however are insignificant. Nonetheless, it has a low R^2 (0.1198) even on the training set. We use this model to predict on the test data and it yields a RMSE of 0.2272.

3.1.1 Lasso Regression

With a cross-validation approach to tune for the best, $\lambda = 2.1488\times10^{-4}$, (*figure III in appendix*), we trained a model that uses 22 predictors. On the testing data, it got a RMSE of 0.2267.

3.1.2 Best Subset Model

Under the adjusted R^2 and C_p, they yield the models with similar complexity. On the contrary, *BIC* yields a model with lower complexity as it places a higher penalty on the number of predictors.*(figure IV in appendix*) We used the adjusted R^2 for our evaluation because we believed our model is fairly under-fitted given the low R^2 from the MLR. Under the R2 criterion, the best subset employs 16 variables and yields an RMSE of 0.2268.

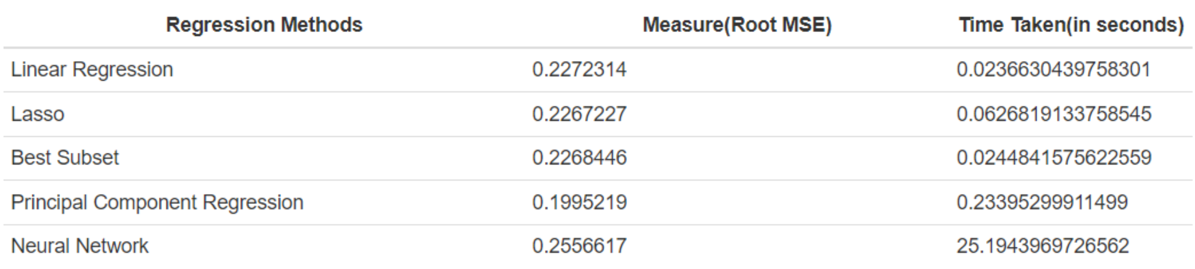
3.1.3 Principal Component Regression

We selected the number of components in the PCR model by cross-validation(*figure V in appendix*) and one standard error rule. The resulting model is used to predict on the test data and it yields an RMSE of 0.1995.

3.1.4 Neural Network

We trained a neural network with two hidden layers on the training dataset(*figure VI in appendix*) and used it to predict on the test data. It took some time and the final model yields an RMSE of 0.2556617.

3.1.5 Comparison of the Regression Models



Regression Methods	Measure(Root MSE)	Time Taken(in seconds)
Linear Regression	0.2272314	0.0236630439758301
Lasso	0.2267227	0.0626819133758545
Best Subset	0.2268446	0.0244841575622559
Principal Component Regression	0.1995219	0.23395299911499
Neural Network	0.2556617	25.1943969726562

Figure 1: Evaluation Metrics of the Regression Model

Based on the RMSE, the PCR model performs the best. However, we recognised the significance of its computational demand. It would not scale well when studying all other US states. For the final model, hence, we selected the Lasso model which has the lowest RMSE and yields an RMSE of 0.2089 on the validation data.

3.2 Logistic Regression
Note that we used the best subset using accuracy as the criterion to choose the best model (instead of the sensitivity. This model chosen has 2 predictors. We train this model on the training data and use it to predict the test data. The model gives an accuracy of 0.83584. (misclassification error rate of 0.1642)

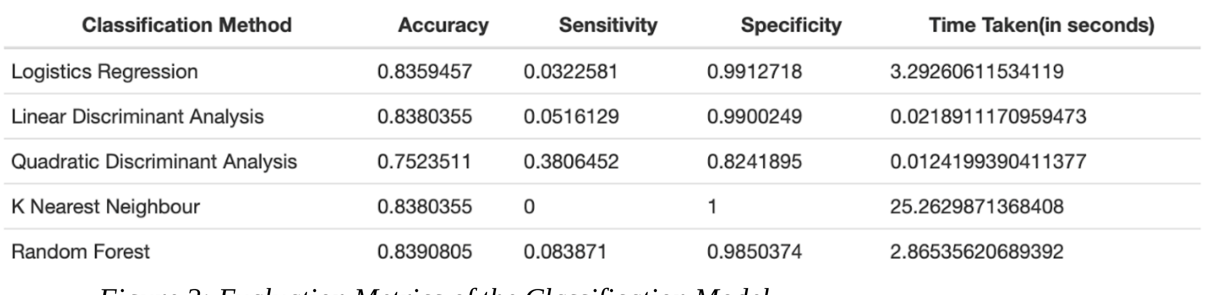
3.2.1 Linear Discriminant Analysis
We trained a LDA model and used this to predict on the test data. This model yields an accuracy of 0.8380355. (misclassification error rate of 0.1620)

3.2.2 Quadratic Discriminant Analysis (QDA)
We apply QDA on the training data and then use it to predict on the test data. We do observe the QDA is more sensitive to tracts with a decrease in economic performance. This model gives an accuracy of 0.7523511 (misclassification error rate of 0.2476).

3.2.3 K-Nearest Neighbour
Cross-validation method is used to select the optimal k for the KNN on the training data. Having selected this k, we use it to fit the test data. We do observe that this model takes some time for prediction and classifies all tracts to "up" label. The accuracy of this model is 0.8380 (misclassification error rate of 0.1620). This accuracy should be held in the context that there is a significant class imbalance and while the model performs well in terms of its specificity which is 1, we observe that the sensitivity is 0.

3.2.4 Random Forest
We first use a classification tree on the training data and observe that this tree always assigns an observation with an "up" label and fails to identify any census tract that has a decrease in median income (*figure VII in Appendix*). Since the "full tree" trained by the algorithm is close to a stump and the predicted outcome is always an increase in median income, we assumed that a simple tree is not suitable for this problem and did not proceed to prune the tree further. Instead, we move on to use a more complex tree-based model to see if the performance would improve.

We used Random Forest on the training data. Since the Random Forest is not sensitive to the number of trees we selected, we used the number of trees as 13 for computational efficiency. We used this model to predict on the test data and it yields an accuracy of 0.8391 (misclassification error rate of 0.1609)

3.2.5 Comparison of the Classification Models


Classification Method	Accuracy	Sensitivity	Specificity	Time Taken(in seconds)
Logistics Regression	0.8359457	0.0322581	0.9912718	3.29260611534119
Linear Discriminant Analysis	0.8380355	0.0516129	0.9900249	0.0218911170959473
Quadratic Discriminant Analysis	0.7523511	0.3806452	0.8241895	0.0124199390411377
K Nearest Neighbour	0.8380355	0	1	25.2629871368408
Random Forest	0.8390805	0.083871	0.9850374	2.86535620689392

Figure 2: Evaluation Metrics of the Classification Model

Although the simpler model yields a higher accuracy rate, they are insensitive to the census tracts with a decrease in median income. Given the class imbalance, we believe that the trade-off between sensitivity and specificity achieved in the QDA is superior to other models we explored, so we proceeded with the QDA as the final model. On the validation dataset, both the accuracy and sensitivity dropped to 0.7346 and 0.2849 respectively. Nonetheless, it is still better than the performance of other models on the testing set in terms of sensitivity.

1.4. Discussion and Conclusion

The machine learning models applied to the dataset did not perform well on the data. The final regression model has an RMSE that is more than double the variance of the response variable. Similarly, the classification also performed poorly, especially in classifying census tracts that recorded a decrease in median income. This could simply be due to the noisy nature of socioeconomic data or due to the imbalance of classes which affects classification models like KNN. The performance of machine learning models on the dataset could also be restricted by the methodology.

Firstly, we assumed a linear relationship between all variables and the response variable. There could be a diminishing economic return on any given social capital and conversely an exponential relationship between the predictors and the response variable. There may also be an interaction between the socioeconomic variables (Iyer et al, 2005). As we observed the full linear regression model did not perform much worse than the shrunk models. This could indicate that the initial model has been misspecified and has a relatively low bias to start with.

Secondly, we only used 20 metrics to represent the social capital. Given the depth of the dataset and the potential to deepen the dataset by including more states, we could include more variables from the ACS5, which holds more than 4000 variables.

Thirdly, we assumed that social capital is static over the period of study. While this is a convenient assumption that facilitates the interpretation of the result, this social capital could also have changed in the time period, for example, due to migration which in turn affected the economic performance. A dominant example would be the downtown area of San Francisco. We believe that in future studies, an analysis of the changes in social capital over the time period would improve the rigour of the findings.

Fourthly, we did not adjust the median income by inflation using the same base year. We simply exploited the data available from a single credible source for our data. Adjusting for inflation with the same base year, it is likely to offset the class imbalance and a more ideal set-up for the classification problem.

Part II: Coordinate Descent Algorithm

2.1. Introduction

In this part, we aim to apply the 'one-at-a-time' coordinate descent type of algorithm to solve the penalized regression problems – the lasso problem and elastic net – and to analyse its performance. We first develop our coordinate descent algorithms to solve the lasso penalty and the elastic net penalty, after which we simulate the data. We then obtain the tuning parameters for our models by performing k-fold cross validation using the mean-squared error (MSE) values. Finally, we consider the following scenarios:

- Case I: $n > p$
- Case II: $p > n$

when pairwise correlations between the group of variables are very high, and also when X_i's are uncorrelated.

2.2. Coordinate Descent Algorithm for Lasso

To solve the Lasso problem, we employ the coordinate descent approach as demonstrated below. The vector of coefficients β is returned by our function lasso_coordinate_descent(X, y, lambda, max_iter = 500, tol = 1e-6) when the parameters are passed in as arguments.

1. We start by initializing βj and $\beta j^{old} = 0$, $j = 1, ..., p$.

2. Until the approach converges for $j = 1, ..., p$, we continue to execute the below mentioned steps:
   a. Let $\beta_j^{old} = \beta_j$. We compute the partial residual by taking:

$$r_{ij} = y_i - \sum_{k \neq j} x_{ik} \beta_k$$

   b. For the j-th predictor, we compute the simple least square coefficients of residuals (r_{ij}):

$$\beta_j^* = \frac{1}{n} \sum_{i=1}^{n} x_{ij} r_{ij}$$

   c. Using the soft thresholding approach, we can calculate β_j as follows:

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

3. The *max(abs(beta – beta_old)) < tol* checks whether the maximum absolute difference between the elements of the current *beta* estimate and the previous *beta_old* estimate is smaller than a given tolerance, *tol* i.e., 10^{-6}.

   When this condition becomes TRUE, it means that the change in coefficients between successive iterations has become sufficiently small, indicating convergence or reaching a point where the change in coefficients is negligible according to the tolerance level specified by *tol.* At this point, the algorithm is likely to stop further iterations to avoid unnecessary computation and to consider the current *beta* estimate as the solution.

2.3. Coordinate Descent Algorithm for Elastic Net

The coordinate descent algorithm for the elastic net problem is very similar to the algorithm used to solve the lasso, except for a modification to the soft thresholding in the algorithm as follows:

$$\beta_j = \text{sign}(\beta_j^*) \max((|\beta_j^*| - \lambda_1), 0) (1+2 \lambda_2)^{-1}$$

2.4. Data Simulation Settings

We simulate the data via the model:

$$y = X\boldsymbol{\beta} + \boldsymbol{\sigma\varepsilon},$$

$$\boldsymbol{\varepsilon} \sim N(0, I_n)$$

We first set the parameters to the default values as follows:

- $n = 240$
- $\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$
- $\boldsymbol{\sigma} = 3$
- The pairwise correlations between X_i, X_j is determined as: $\text{corr}(X_i, X_j) = 0.5^{|i - j|}$
- r (number of simulations of the entire process) = 50

The code conducts 50 simulations and initiates by setting simulation parameters, including n, p, and the true coefficients for the linear model.

Subsequently, a data frame named ‘final’ is created to store the results of 50 simulations, with columns representing lambda values (of lasso), lambda_1 and lambda_2 values (of elastic net), the count of nonzero coefficients for both lasso and elastic net (beta_ct_lasso and beta_ct_en), and mean-squared errors for each problem (mse_lasso and mse_en).

2.5. Selection of the Regularization Parameters

After simulating the datasets for X and Y, we select optimal tuning parameters, namely λ in the lasso algorithm and λ_1, λ_2 in the elastic net algorithm.

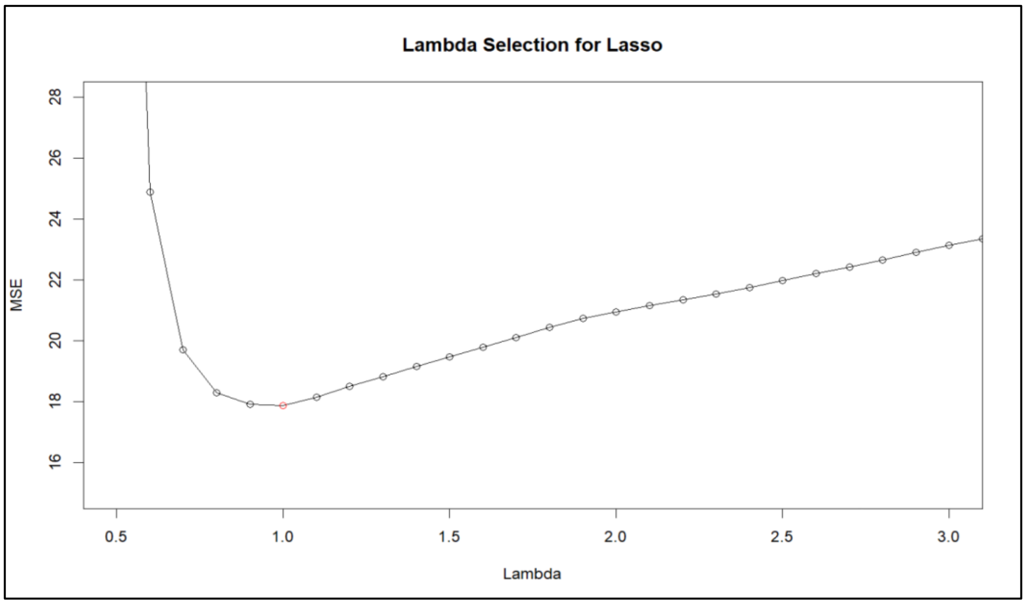
The approach involves a trial-and-error process, in which different values of λ are tested on the training dataset. This entails fitting a range of λ into the lasso and elastic net functions and evaluating the returned β coefficients on the validation data. To validate the data, we use the k-folds cross validation method, via which we divided the data into 12 folds (so as to have 20 data points in each fold for our case of n = 240).

To create the folds, we randomly assigned each row of the simulated data a number between 1 to 12 such that each number occurred 20 times. Running a *for* loop, we then curated the training data by picking those rows which had the key values in the *for* loop. For example, when i = 2, all the rows having fold values as 2 are selected as the training data.

The optimal parameter λ is chosen based on the smallest error (MSE) observed on the validation data. To select the optimal tuning parameter we ran the algorithm using trial values in the range of 0 to 5 with an increment of 0.1. Subsequently, the model is re-estimated on the training data using the optimal tuning parameters, and final results are obtained by evaluating it on the test data.

To illustrate the selection of the optimal lambda, we have the following plot showing the lambda selection for the lasso problem by plotting lambda values against the MSE, when solved using the coordinate descent algorithm (for n = 240 and p = 8).

The optimal value of lambda has been highlighted as a red point in the plot.



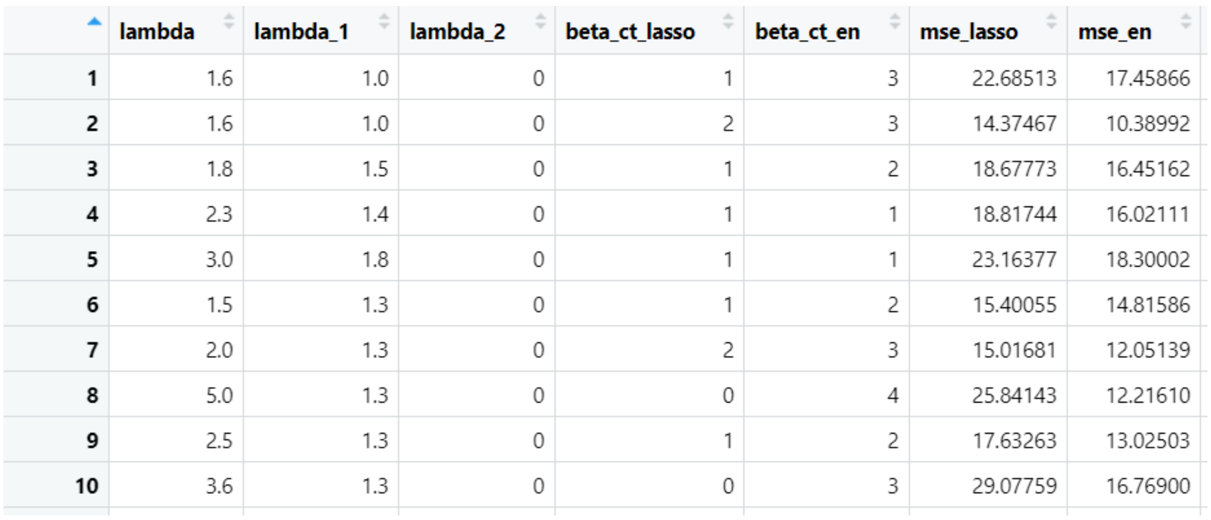
2.6. Main Simulation Results

We get the following results for different cases.

2.6.1. In the case: n > p

Within a loop iterating over 50 simulations, datasets are generated with uncorrelated predictors, and cross-validation is employed to estimate optimal regularization parameters. The data is then split into training and test sets, and lasso and elastic net models are fitted using coordinate descent algorithms.

The code calculates relevant metrics, including the count of nonzero coefficients and mean-squared errors for both methods, storing the results in the ‘final’ data frame, providing a comprehensive basis for comparing the predictive accuracy of the two regularization methods across multiple simulated scenarios. The ‘final’ data frame is as follows:



	lambda	lambda_1	lambda_2	beta_ct_lasso	beta_ct_en	mse_lasso	mse_en
1	1.6	1.0	0	1	3	22.68513	17.45866
2	1.6	1.0	0	2	3	14.37467	10.38992
3	1.8	1.5	0	1	2	18.67773	16.45162
4	2.3	1.4	0	1	1	18.81744	16.02111
5	3.0	1.8	0	1	1	23.16377	18.30002
6	1.5	1.3	0	1	2	15.40055	14.81586
7	2.0	1.3	0	2	3	15.01681	12.05139
8	5.0	1.3	0	0	4	25.84143	12.21610
9	2.5	1.3	0	1	2	17.63263	13.02503
10	3.6	1.3	0	0	3	29.07759	16.76900

In this data frame, beta_ct_lasso and beta_ct_rn refer to the number of non-zero coefficients under lasso and elastic net respectively. The columns mse_lasso and mse_en refer to the mean square errors (MSEs) obtained under lasso and elastic net respectively.

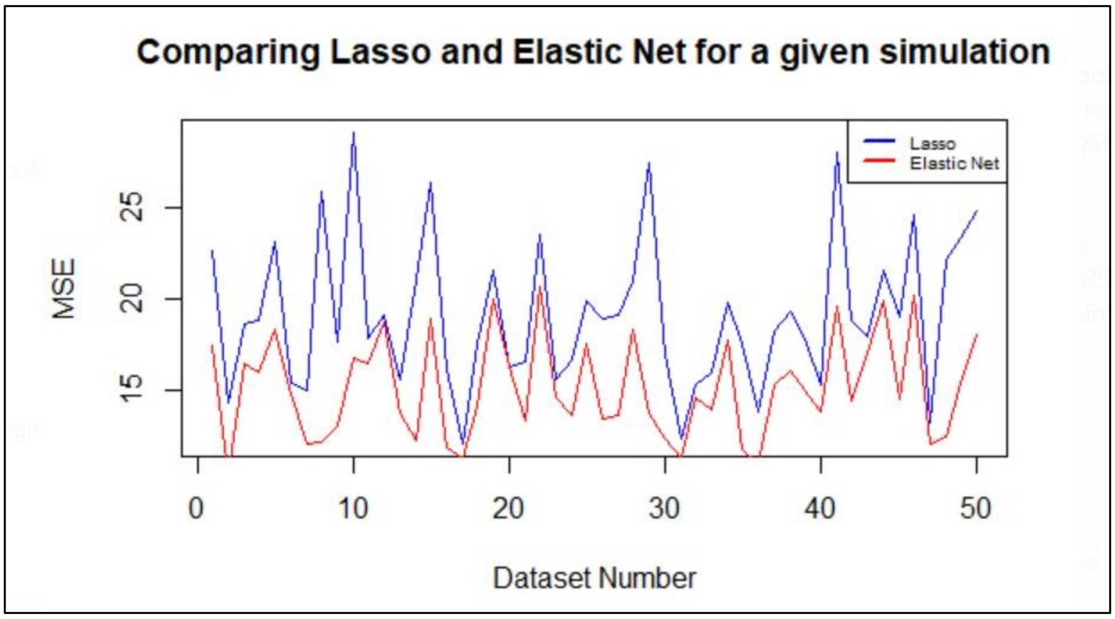
As expected, it is apparent simply from the resulting data frame that elastic net not only dominates lasso in terms of predictive accuracy (because it clearly has less MSE than the lasso in every simulation) but also does a good job in variable selection (it has equal or greater non-zero coefficients in every simulation). To further prove this, we compute:

- the mean and standard error of mean-squared errors mse_lasso and mse_en
- number of estimated non-zero coefficients

In the case n = 240 and p = 8, we have across 50 simulations:

- The mean of MSE for lasso is 19.19177, which is greater than the mean of MSE for elastic net which is 15.15362. The standard error of MSE for lasso = 4.094770, which is greater than standard error of MSE for elastic net which is 2.767536. Thus, we can conclude that **the elastic net has a better predictive performance**.
- On average the number of estimated non-zero coefficients selected by lasso is 1.36, which is less than that for elastic net which is 2.26 on average. Hence, we can conclude that **the elastic net regularization does better than the lasso at variable selection**.

The figure below demonstrates how elastic net clearly dominates over lasso due to lesser mean-squared error values.



To also know whether these results vary with a change in the number of observations, we repeated the simulations to get further results which are explored in section 2.7.

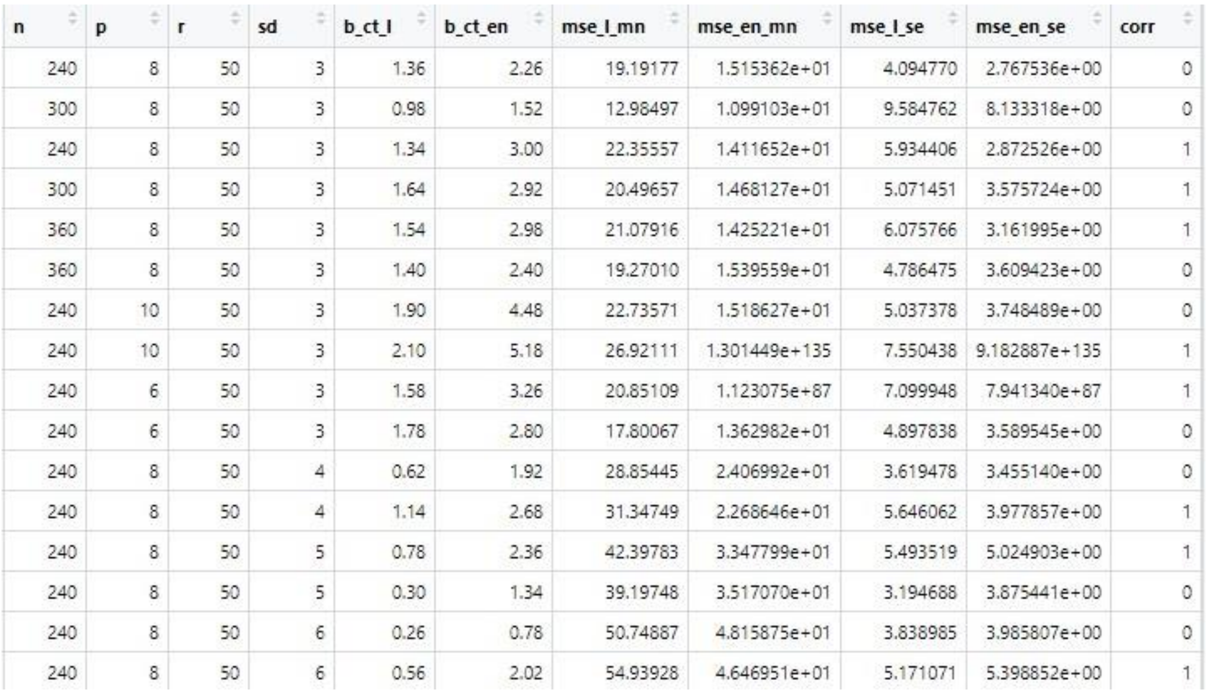
2.6.2. In the case: n < p

When p is significantly larger than n, the elastic net penalty demonstrates greater accuracy compared to lasso, especially when data availability is scarce/limited for estimating numerous coefficients. Conversely, when the number of variables is small, the elastic net regularization tends to converge towards lasso regularization.

2.7. Additional Simulation Settings – Uncorrelated and Correlated Variables

We had set our default case as n = 240, p = 8 and sd = 3 (as demonstrated before). Then, we attempted to run the code for different values of each of these one at a time while keeping the other parameters constant.

We obtained the following data frame of results, where in the *corr* column 0 represents the case of uncorrelated variables while 1 represents the case of correlated variables.



n	p	r	sd	b_ct_l	b_ct_en	mse_l_mn	mse_en_mn	mse_l_se	mse_en_se	corr
240	8	50	3	1.36	2.26	19.19177	1.515362e+01	4.094770	2.767536e+00	0
300	8	50	3	0.98	1.52	12.98497	1.099103e+01	9.584762	8.133318e+00	0
240	8	50	3	1.34	3.00	22.35557	1.411652e+01	5.934406	2.872526e+00	1
300	8	50	3	1.64	2.92	20.49657	1.468127e+01	5.071451	3.575724e+00	1
360	8	50	3	1.54	2.98	21.07916	1.425221e+01	6.075766	3.161995e+00	1
360	8	50	3	1.40	2.40	19.27010	1.539559e+01	4.786475	3.609423e+00	0
240	10	50	3	1.90	4.48	22.73571	1.518627e+01	5.037378	3.748489e+00	0
240	10	50	3	2.10	5.18	26.92111	1.301449e+135	7.550438	9.182887e+135	1
240	6	50	3	1.58	3.26	20.85109	1.123075e+87	7.099948	7.941340e+87	1
240	6	50	3	1.78	2.80	17.80067	1.362982e+01	4.897838	3.589545e+00	0
240	8	50	4	0.62	1.92	28.85445	2.406992e+01	3.619478	3.455140e+00	0
240	8	50	4	1.14	2.68	31.34749	2.268646e+01	5.646062	3.977857e+00	1
240	8	50	5	0.78	2.36	42.39783	3.347799e+01	5.493519	5.024903e+00	1
240	8	50	5	0.30	1.34	39.19748	3.517070e+01	3.194688	3.875441e+00	0
240	8	50	6	0.26	0.78	50.74887	4.815875e+01	3.838985	3.985807e+00	0
240	8	50	6	0.56	2.02	54.93928	4.646951e+01	5.171071	5.398852e+00	1

2.7.1. Changing the values of n

There's a sensitivity in lasso and elastic net performance to variations in the number of observations (n). Both approaches often gain from a larger dataset as sample sizes rise, which could result in more precise coefficient estimations. When the number of predictors (p) is minimal compared to the number of observations, lasso can potentially have trouble choosing a subset of crucial predictors, which would increase the variability of its estimates. In these situations, elastic net – which incorporates both λ_1 and λ_2 penalties – can offer greater stability. However, both lasso and elastic net may be prone to overfitting when the dataset is relatively small. Furthermore, the number of observations may have an impact on the selection of the ideal regularization parameters, which are established through cross-validation. The intricate interplay of bias, variance, and regularization effects in the connection between n and model performance calls for careful evaluation based on the particulars of the data and the goals of the modelling.

2.7.2. Changing the values of p

Changes in the number of predictors (p) have a considerable impact on lasso and elastic net performance. The difficulties in choosing significant predictors become increasingly apparent as the dataset's complexity rises, particularly for lasso. Lasso typically has trouble efficiently reducing the set of relevant features when there are more predictors than data, which could result in more varied coefficient estimates. Elastic net adds an extra regularization layer by combining λ_1 and λ_2 penalties, which can improve stability in high-dimensional environments. But a significant increase in the number of predictors could also raise the possibility of overfitting, especially if the sample size is still small. In both cases, the dimensionality of the data may have an impact on the regularization parameter selection, which is critical. Therefore, in order to obtain optimal model performance, the influence of modifying the number of predictors should be carefully evaluated, combining the necessity of feature selection with the difficulties presented by high-dimensional data.

2.7.3. Changing the values of σ

We aim to explore the association between the parameter sigma (σ) and the penalties imposed by both the lasso and elastic net methods. This parameter introduces a degree of randomness and volatility to our simulated dataset Y. As the value of σ increases, the level of unpredictability and randomness in our Y also intensifies. We find that the elastic penalties with $\hat{\lambda}_2 = 0$ converge toward the lasso penalties when we decrease the variance in our simulation. Under such circumstances, the elastic net does not yield outcomes that are superior than the lasso problem. On the other hand, the mean mean-squared error of the elastic net performs better than the MSE of the lasso when we reverse this tendency and raise the randomization value σ. This is especially true in cases where $\hat{\lambda}_2$ is statistically larger than zero. Furthermore, there is a negative association between the number of estimated non-zero predictors and the parameter σ. A decrease in the count to correspond with the actual number of variables indicates that a higher σ results in better variable selection.

2.8. Conclusion

We can summarise our findings and results as follows.

2.8.1. Summary

We first developed the 'one-at-a-time' coordinate descent type of algorithm to solve the penalized regression problems – the lasso problem and elastic net. Our aim was to analyse the predictive accuracy of each and to also see which performed better at variable selection.

We simulated datasets, using the default parameter values as:

- $n = 240$
- $\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, 0, 0, 0)'$
- $\boldsymbol{\sigma} = 3$
- The pairwise correlations between X_i, X_j is determined as: $\text{corr}(X_i, X_j) = 0.5^{|i-j|}$
- r (number of simulations of the entire process) = 50

To select the optimal tuning parameter we ran the algorithm using trial values in the range of 0 to 5 with an increment of 0.1. Subsequently, the model is re-estimated on the training data using the optimal tuning parameters, and final results are obtained by evaluating it on the test data.

2.8.2. Results

We obtain results for various cases and settings.

In the case n > p:

Within a loop iterating over 50 simulations, datasets are generated with uncorrelated predictors, and cross-validation is employed to estimate optimal regularization parameters. The data is then split into training and test sets, and lasso and elastic net models are fitted using coordinate descent algorithms.

From our numerical results we note that:

- The mean of MSE for lasso is greater than the mean of MSE for elastic net. The standard error of MSE for lasso is greater than standard error of MSE for elastic net. Thus, we can conclude that **the elastic net has a better predictive performance** than lasso.
- On average the number of estimated non-zero coefficients selected by lasso is less than that for elastic net. Hence, we can conclude that **the elastic net regularization does better than the lasso at variable selection**.

In the case p > n:

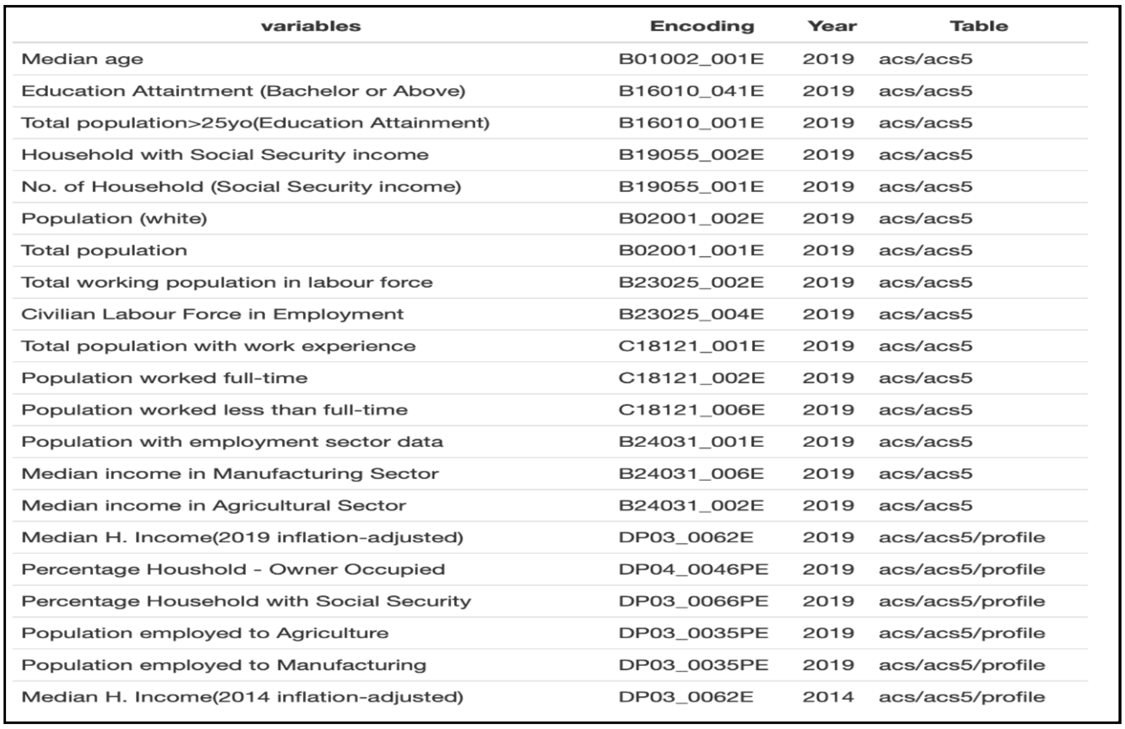
When p is significantly larger than n, the elastic net penalty demonstrates greater efficiency compared to lasso, especially when data availability is limited for estimating numerous coefficients. Conversely, when the number of variables is small, the elastic net regularization tends to converge towards lasso regularization.

Appendices

References

- Friedman, J., Hastic. and Hofling, H. (2007). Pathwise coordinate optimization, The Annals of Applied Statistics 1: 302 - 332.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso, Journal of Royal Statistical Society, Series B 58: 267 - 288.
- Tseng, P.(1988). Coordinate ascent for maximizing nondifferentiable concave functions, Technical Report.
- Tibshirani, R. (2001) Convergence of block coordinate descent method for nondifferentiable maxi- mization, J.Opt. Theory Appl. 109: 474 - 494.
- Iyer, S., Kitson, M. and Toh, B., 2005. Social capital, economic growth and regional development. Regional studies, 39(8), pp.1015-1040.

Figures



variables	Encoding	Year	Table
Median age	B01002_001E	2019	acs/acs5
Education Attaintment (Bachelor or Above)	B16010_041E	2019	acs/acs5
Total population>25yo(Education Attainment)	B16010_001E	2019	acs/acs5
Household with Social Security income	B19055_002E	2019	acs/acs5
No. of Household (Social Security income)	B19055_001E	2019	acs/acs5
Population (white)	B02001_002E	2019	acs/acs5
Total population	B02001_001E	2019	acs/acs5
Total working population in labour force	B23025_002E	2019	acs/acs5
Civilian Labour Force in Employment	B23025_004E	2019	acs/acs5
Total population with work experience	C18121_001E	2019	acs/acs5
Population worked full-time	C18121_002E	2019	acs/acs5
Population worked less than full-time	C18121_006E	2019	acs/acs5
Population with employment sector data	B24031_001E	2019	acs/acs5
Median income in Manufacturing Sector	B24031_006E	2019	acs/acs5
Median income in Agricultural Sector	B24031_002E	2019	acs/acs5
Median H. Income(2019 inflation-adjusted)	DP03_0062E	2019	acs/acs5/profile
Percentage Houshold - Owner Occupied	DP04_0046PE	2019	acs/acs5/profile
Percentage Household with Social Security	DP03_0066PE	2019	acs/acs5/profile
Population employed to Agriculture	DP03_0035PE	2019	acs/acs5/profile
Population employed to Manufacturing	DP03_0035PE	2019	acs/acs5/profile
Median H. Income(2014 inflation-adjusted)	DP03_0062E	2014	acs/acs5/profile

Figure I: Description of Variable Encodings

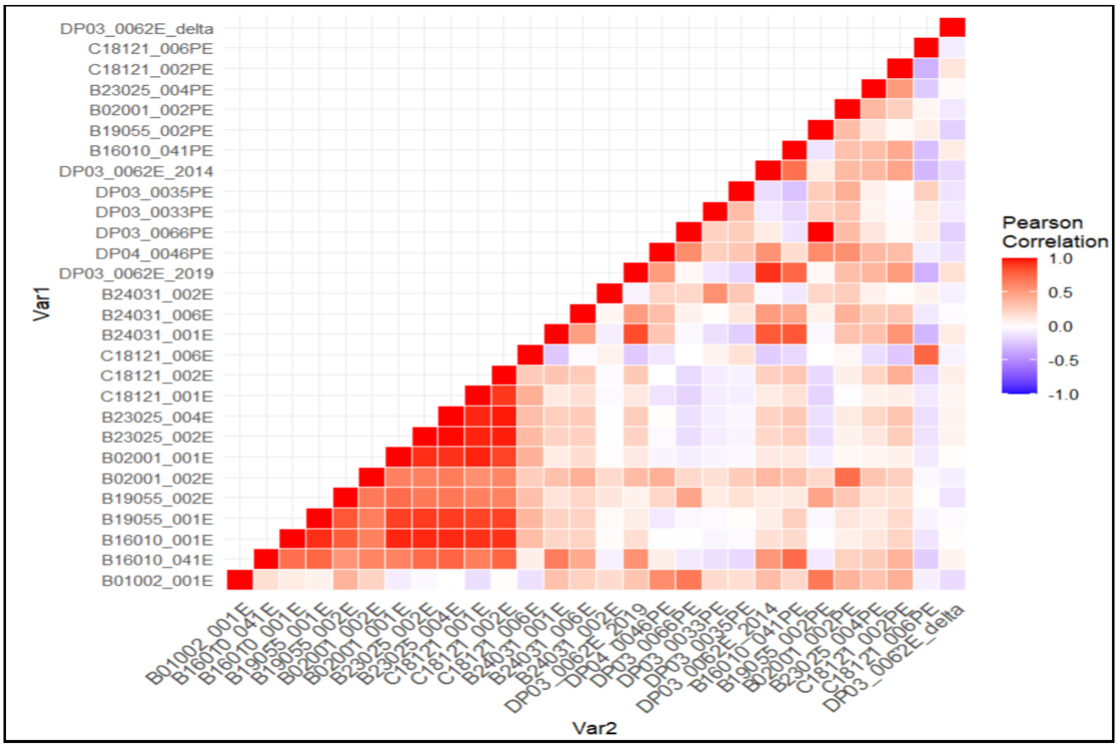


Figure II: Correlation Heatmap

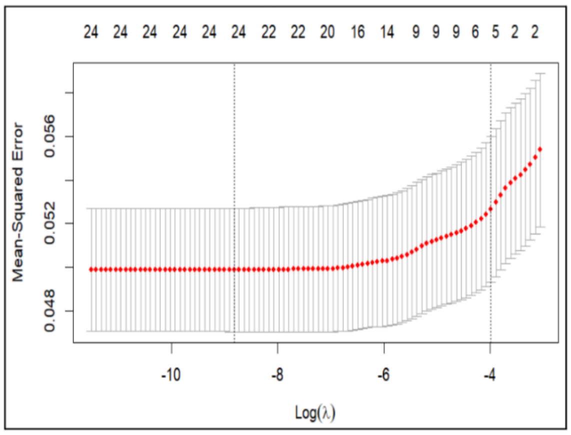


Figure III: Lasso regression cross validation for λ

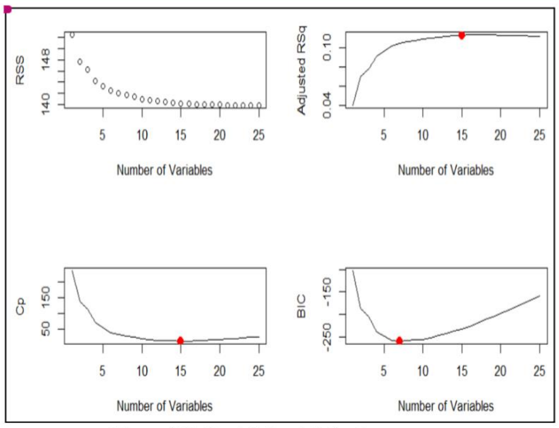


Figure IV: Best Subset Selection

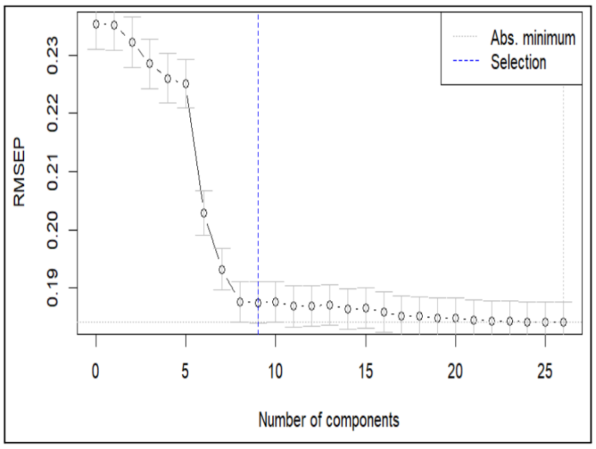


Figure V: Component Selection for PCR

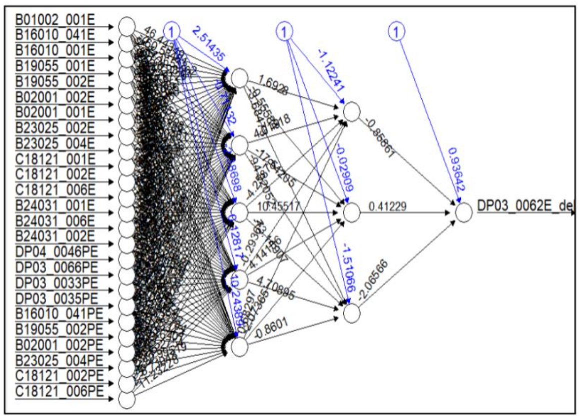


Figure VI: Neural Network

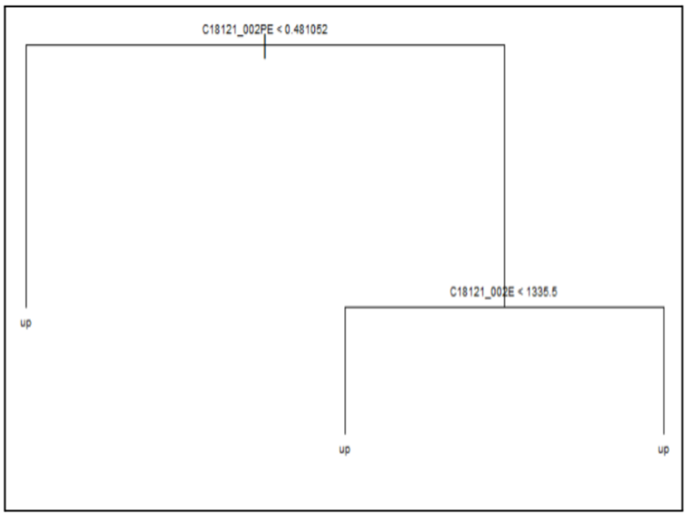


Figure VII: Classification Tree