ST443 – Machine Learning & Data Mining
Group Project

Group 12

12482 – 25% contribution

15688 – 25% contribution

20948 – 25% contribution

13917 – 25% contribution

Professor: Dr. Xinghao Qiao

Department of Statistics

Michaelmas Term 2020

ST443 Group Project Part 1

December 10, 2020

1 Executive Summary

Despite the impact of COVID-19 on the Economy, the music industry continues to flourish at a positive growth rate (5.6% in terms of recorded music revenues in the US) especially streaming music according to *the 2020 Mid-Year Music Revenue Report* | *RIAA*. What makes a song a hit is a question that arose a lot of interest and has been studied various times from different perspectives.

The purpose of this report is to predict the popularity of soundtracks on Spotify based on quantitative measures such as danceability, key, etc. Popularity is a figure ranging from 0 to 100 calculated by the algorithm mainly focusing on how many times a particular soundtrack is played and how recent it is played, To investigate what makes a hit song and predict the popularity, various machine learning techniques are used including linear models, non-linear models, tree-based methods and neural networks. Among these models implemented, the generalized additive model (GAM) gives the best performance with MSE around 32.92

2 Data Source

The dataset was uploaded to Kaggle in 06/2020 by *Yamaç Eren Ay*, who used the Spotify Web API for developers to build a data that contains more than 160,000 songs. Each row in the data represents a unique track, identified by a unique ID feature generated by Spotify. The columns are 19 features of the tracks including acousticness, artists, danceability, duration_ms, energy, explicit, id, instrumentalness, key, liveness, loudness, mode, name, popularity, release_date, speechiness, tempo, valence and year (Figure 1 in the Appendix).

After shuffling the index, the original data set was divided into 2 parts—70% for training and 30% for testing.

3 Empirical Results and Analysis

3.1 Linear Regression

Three linear models are used to predict the popularity of songs on Spotify including linear regression, lasso, and ridge regression. Among these linear models, linear regression gives the lowest MSE, which is 47.074 as shown in the table. As for lasso and regression, the MSE for testing datasets are relatively larger than that for linear regression considering the penalty term in the form of $\lambda \sum_{j=1}^{15} |\beta_j|$ and $\lambda \sum_{j=1}^{15} \beta_j^2$ respectively. Additionally, the linear regression model with variables automatically selected by lasso returns a similar MSE (47.262) to the fundamental linear model.

Model	Linear Regression (with all features)	Linear Regression (with only statistically significant variables	Lasso	Ridge	Linear Regression (with variables selected by lasso)
MSE	47.074	47.104	74.049	237.644	47.262

According to the linear model with variables selected with the lasso (Fig 2 in the Appendix), which is simpler in terms of inference and performs well with 86% of the variation explained by the selected features. The type of music and the release year play a significant role in

determining the popularity of the song (Fig 2). For instance, acoustic songs are overall less popular than non-acoustic ones with other features remaining the same, shown by the negative relationship between acousticness and popularity. Accousticness is a confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic, based on Spotify Audio Features for a Track. On the other hand, danceability is positively correlated with popularity, which indicates the popularity of dance music. What's more, as popularity is calculated by algorithm and is based partially on the total number of plays the track has had and how recent those plays are, songs released in recent years are generally more popular than those that were released long ago. This can be proven by the positive relationship between year and popularity.

The performance of the linear model with variables selected has been further assessed with plots in Fig 3 and Fig 4. The linearity holds reasonably well according to the Residuals vs Fitted plot as the red line is horizontal at 0, but the linear model's performance, especially when it comes to prediction, is questionable when the popularity is too either too small or too large as shown in Fig 4. This can be partially explained by the heavy tail of residuals in the Normal Q-Q plot in Fig 3. Apart from that, the linear model's performance is quite good with residuals equally spread along with fitted values and relatively small leverage in most cases.

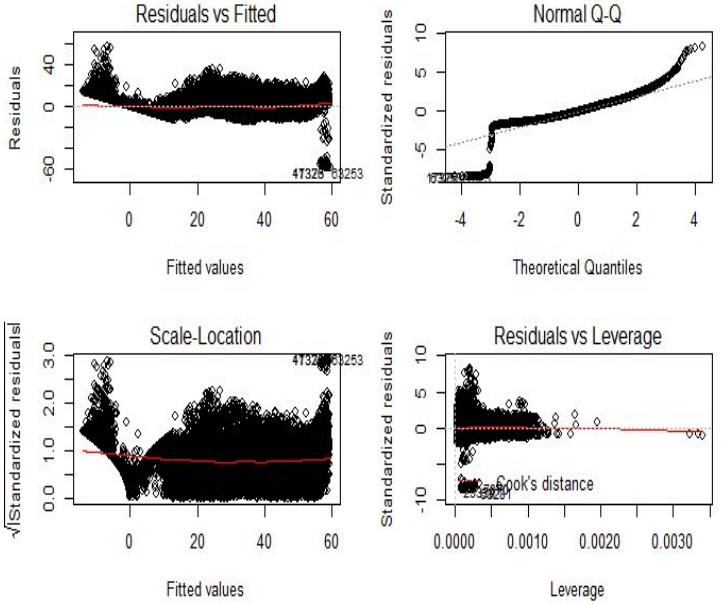

Figure 3 Plots of linear regression

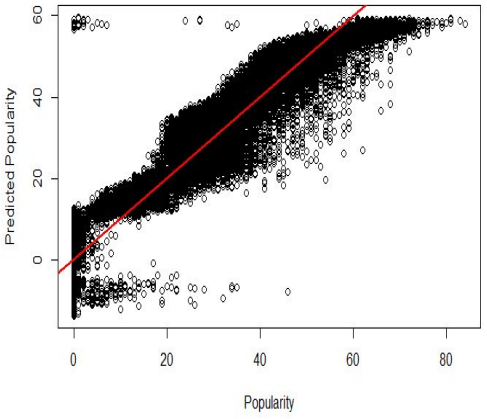

Figure 4 predicted pop vs pop

3.2 Non—Linear Model

3.2.1 Tree Based Model

Firstly, further data processing is applied. Duration_ms, Instrumentalness, and Speechiness, which originally are numerical features, but considering their practical meanings of the value range in tree-based models and also EDA, are transformed to categorical features. Duration_ms was changed to minute-unit which is a more common pattern, and was divided into two categories by "5 min". Due to the consideration of extreme values, the instrumentalness was turned into 3 intervals. And the speechiness was separated by a seeming breakpoint—0.62.

Then, 2 classic tree-based models are used to make a prediction. Model 1 is a regression tree with a default parameter setting. The result is quite surprising, because only 1 feature—year, is used in tree construction, which shows it much more significant and dominant. It shows that for each node only using the feature 'year' can maximum the information gain. It is reasonable because the popularity under the current situation has a great correlation with the songs' 'age'. It

seems to be overwhelming compared to other features. People tend to listen to newersongs than older ones. The prediction ability is not good visually, especially since the predicted values are several fixed values due to the model limitation from branch number, and the MSE is 92.49778. (Fig 5 and Fig 6).

Model 2 is a random forest model. The prediction ability is much better visually, which shows an obvious 45-degree upward trend and more diversified predicted values. (Fig 7) And the MSE is 83.69294, performing better than the regression tree model. Unsurprisingly, the feature “year” is most important. However, some other features like loudness, danceability, energy, valence, acousticness and tempo have an obvious effect on popularity, which makes sense intuitively. For instance, some rock music using a loud voice, dynamic melody and energetic beats to convey power will inspire people or make a hot atmosphere to attract a wide range of listeners. (Fig 8)

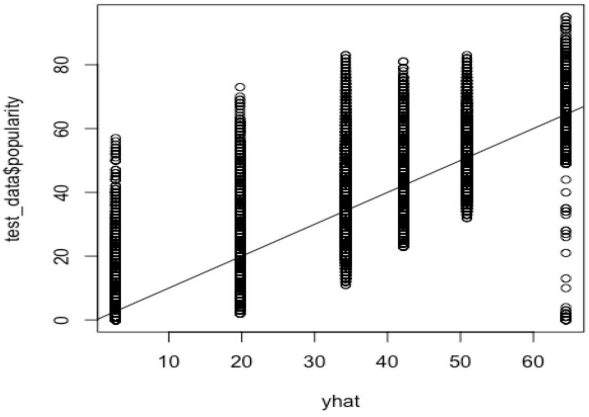

Figure 6 Predicted and Real Values of DT

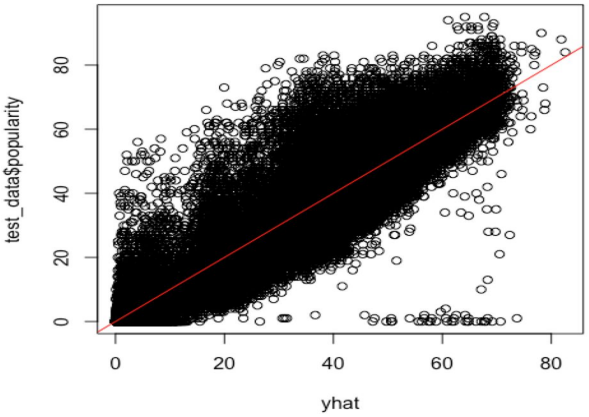

Figure 8 Predicted and Real Values of RF

4.2.2 Neural Network

The first step is also the specific data processing. For neural network, all the input features should be numerical and also share the same dimension. So all the features are scaled at the interval [0,1] through normalization. Under the setting that 1 hidden level, 5 neurons, 1e+07 stepmax and 0.1 threshold (Fig 9), the output network concludes 3 layers, the black line shows the connection between each layer and the weight on each connection, while the blue line shows the deviation term added in each step, and the deviation can be thought of as the intercept of the linear model. But the network is just like a black box without robust explanatory ability. The model performance also shows a 45-degree upward trend in the Predicted and Real Values Figure, but the predicted results are always smaller than the actual ones (Fig 10). Then the predicted values need to scale back to [0, 100] and are used to calculate MSE, which is 247.5759 perhaps because of the accumulation effect of the slight underestimation for many samples. The high error could also be also due to the number of times we back propagated the results, and the size of the training and validation sets. It could also be due to the loss function and the activation functions used on the hidden nodes.

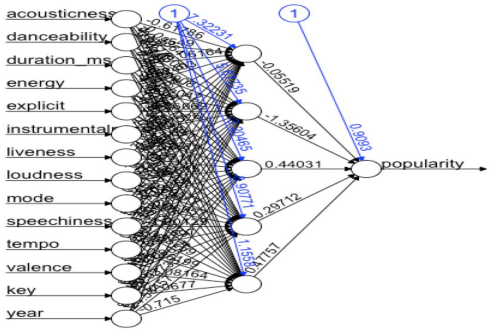

Figure 9 Trained Neural Network

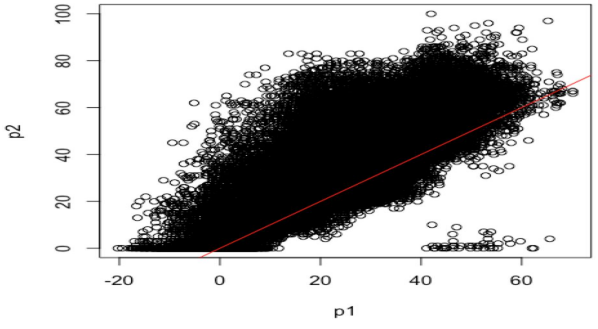

Figure 10 Predicted and Real Values

3.3 Generalized Additive Model (GAM)

With the complexity of the relationship between year and popularity in mind (shown in Fig 11), a generalized additive model in the form of $y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_5(x_{i5}) + \varepsilon_i$ is used where the model used for year is natural spline with $df = 6$ by trials and errors while the linearity of other covariates remains the same. This GAM gives the lowest MSE (32.92) among all the models implemented. Besides, GAM performs better when dealing with popularities above upper quantile and below lower quantiles compared with linear regression with selected variables as shown in the boxplots in Fig 12 and Fig 13.

4 Conclusion

The dataset with little errors contains many samples with diversified professional indicators of music. It can be used in predicting the popularity of music, a practical function, which is useful guidance for musician creation and also provides a reference for music propaganda. And also some research on popularity based on such dataset can help to decide which songs will be played in advertising. Before modeling, proper EDA and real cases research are the foundation for feature selection and categorical variable construction. The next step is applying the dataset in different models (OLS / Lasso / Ridge / Regression Tree / Random Forest / Neural Network / GAM) to get an optimal one for popularity prediction.

For linear models, if only considering the accuracy of the predicted values, the linear model with all the features will be chosen because of its lowest MSE. However, both model practical explanatory ability and model simplicity should be evaluated standard, then the linear model with variables selected with the lasso is the final decision because it only keeps the variables with significance and practical meanings. From the model, acousticness and speechiness show a negative relationship with popularity, and danceability, loudness, and year have a positive relationship with popularity. Intuitively, the fact is that people less prefer acoustic music with high acousticness and speech-like recording with high speechiness (e.g. talk show, audiobook, poetry). Also, this generation enjoys powerful and dynamic music with high danceability and loudness. Keeping with fashion and following the mainstream proves the positive effect of the feature ‘year’.

For non-linear models, GAM has the best performance which absorbs the information from other models. But just as other models, the GAM model tends to underestimate the values especially in the case of extreme fitted values, and also has the problem that output contains the negative value.

But limited by computer hardware handling such scale data (like computing ability and running speed), more complex models like advanced neural networks and XGboost have not been tested. To improve single non-linear model performance, methods like simple looping, grid search, random search and Bayesian Optimization can be equipped for searching optimal parameters of the model. Meanwhile, at the level of model practical effect in business scenario, there is also the possible negative cycle if using these models as reference or guidance to create more and more of the same music (as they're popular due to the model output) but then this huge increase in supply of this type of music makes the market saturated and everyone starts to hate this music as it's become TOO popular.

Appendix

```
── Data Summary ────────────
                           Values
Name                       data
Number of rows             169909
Number of columns          19
_________________________
Column type frequency:
  character                4
  numeric                  15
_________________________
Group variables            None

── Variable type: character ──────────────────────────────────────────────────────
  skim_variable n_missing complete_rate   min   max empty n_unique whitespace
1 artists               0             1     5   661     0    33375          0
2 id                    0             1    22    22     0   169909          0
3 name                  0             1     1   255     0   132940          0
4 release_date          0             1     4    10     0    10882          0

── Variable type: numeric ────────────────────────────────────────────────────────
   skim_variable    n_missing complete_rate        mean          sd    p0        p25          p50        p75       p100 hist
 1 acousticness             0             1       0.493       0.377     0      0.0945       0.492      0.888      0.996 ▅▃▂▃▇
 2 danceability             0             1       0.538       0.175     0      0.417        0.548      0.667      0.988 ▁▃▇▇▂
 3 duration_ms              0             1  231406.     121322.     5108 171040       208600     262960     5403500   ▇▁▁▁▁
 4 energy                   0             1       0.489       0.267     0      0.263        0.481      0.71       1     ▅▅▅▆▇
 5 explicit                 0             1       0.0849      0.279     0      0            0          0          1     ▇▁▁▁▁
 6 instrumentalness         0             1       0.162       0.309     0      0            0.000204   0.0868     1     ▇▁▁▁▂
 7 key                      0             1       5.20        3.52      0      2            5          8         11     ▆▆▇▃▅
 8 liveness                 0             1       0.207       0.177     0      0.0984       0.135      0.263      1     ▇▃▂▁▁
 9 loudness                 0             1     -11.4         5.67    -60    -14.5        -10.5       -7.12      3.86  ▁▁▁▂▇
10 mode                     0             1       0.709       0.454     0      0            1          1          1     ▃▁▁▁▇
11 popularity               0             1      31.6        21.6       0     12           33         48        100    ▇▆▅▂▁
12 speechiness              0             1       0.0941      0.150     0      0.0349       0.045      0.0754     0.969 ▇▁▁▁▁
13 tempo                    0             1     117.         30.7       0     93.5        115.       136.       244.   ▁▆▇▃▁
14 valence                  0             1       0.532       0.262     0      0.322        0.544      0.749      1     ▅▅▅▆▇
15 year                     0             1    1977.         25.6    1921   1957         1978       1999       2020   ▁▂▃▅▇
```

Figure 1 Data Overview

```
Call:
lm(formula = popularity ~ acousticness + danceability + loudness +
    speechiness + year, data = music_train)

Residuals:
    Min      1Q  Median      3Q     Max
-58.940  -5.008  -0.698   4.332  56.660

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept) -1.360e+03  3.481e+00 -390.713  < 2e-16 ***
acousticness -6.787e-01  1.256e-01   -5.405  6.5e-08 ***
danceability  3.453e+00  2.103e-01   16.422  < 2e-16 ***
loudness      2.192e-02  7.740e-03    2.833  0.00462 **
speechiness  -3.877e+00  2.510e-01  -15.443  < 2e-16 ***
year          7.013e-01  1.737e-03  403.766  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.973 on 42204 degrees of freedom
Multiple R-squared:  0.8697,    Adjusted R-squared:  0.8697
F-statistic: 5.634e+04 on 5 and 42204 DF,  p-value: < 2.2e-16
```

Figure 2 Result of linear regression with selected variables

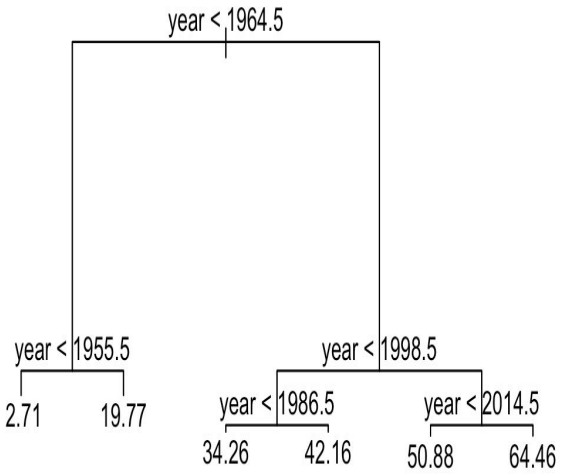

Figure 5 Decision Tree

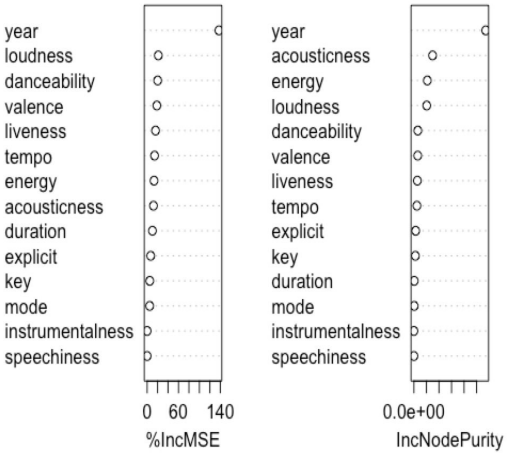

Figure 8 Feature Importance

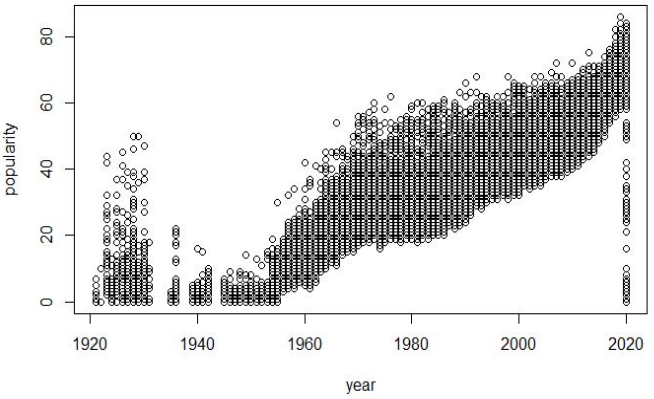

Figure 11 Scatter plot of popularity vs Year

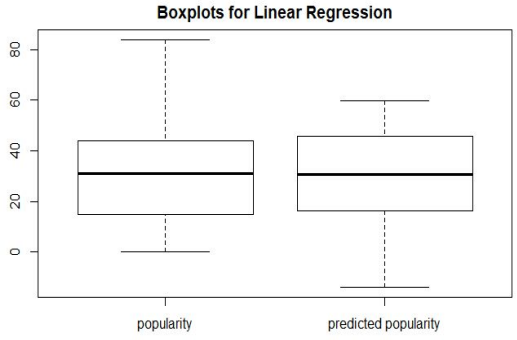

Figure 12 Boxplot of Linear Regression

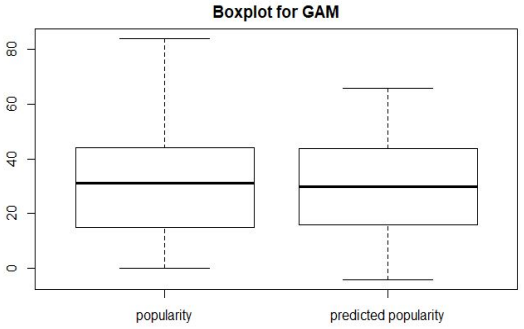

Figure 13 Boxplot of GAM

ST443 Group Project Part 2

December 10, 2020

We aim to look at the variable selection and regularisation methods LASSO (least absolute shrinkage and selection operator) and Elastic Net (EN) and compare their results. The methods differ in the penalty term they include. LASSO uses the l_1 norm and EN uses the l_2 norm in addition to the l_1 norm.

We aim to show that the coordinate descent algorithm can be used to find the optimal values of the penalty terms in the LASSO and Elastic Net algorithms, and the accompanying β's. We denote the penalty terms as λ where $\hat{\lambda}_1$ is the penalty term for LASSO and $\hat{\lambda}_2$ is the additional penalty term in Elastic Net, as well as $\hat{\lambda}_1$. These penalty terms will shrink the coefficients (β's) down towards zero, and many of them can become exactly 0. Because the Elastic Net combines two penalty terms, it tends to shrink the coefficients less than LASSO and thus has less coefficients equalling 0. Also, since the Elastic Net uses the l_2 norm the coefficients will always be positive (as the l_2 norm includes a squared term).

Dataset Creation

We created the dataset by taking random samples from $N \sim (0, I_N)$ for each of our X values and repeated the same process once for a value of ϵ which was our noise vector. These were inputs to our model $\boldsymbol{Y} = \boldsymbol{\beta} X + \sigma \boldsymbol{\epsilon}$.

Initially we used the parameter values taken from the question brief as below:

- $\sigma = 3$
- Actual β's = (3, 1.5, 0, 0, 2, 0, 0, 0)
- Number of datasets = 50
- Number of observations, n, in total = 240
- Number of observations in training set = 20
- Number of observations in validation set = 20
- Number of observations in testing set = 200
- Pairwise correlation between each of the variables was calculated as: $corr(i, j) = (0.5)^{i-j}$, denoted cor.

We used nested *for* loops to calculate the pairwise correlations which were saved to a matrix *cor*.

We created the *data_creation* function with arguments $(cor, x, n, \sigma, actual\_beta)$ where *cor*, σ, and *actual_beta* are described as above, n is the total number of observations and x is the proportion of n which is dedicated to the training set, conversely $(1 - x)$ is used inside the function to calculate the proportion dedicated to the validation set. *data_creation* creates the respective independent training, independent validation, and test sets as well as the accompanying noise vectors. Further, we create vectors of the true Y values that would be expected using the actual beta values which will be used to calculate the partial residuals later when estimating the respective β values.

Coordinate Descent Algorithm

Partial Residuals

To estimate the betas we must calculate the partial residuals as per 2(a) in the brief. For each i, j we calculate the residuals using only the values for j that do not match the current j^{th} element we are evaluating.

Least Squares

Using the matrix of partial residuals we estimated the values of each of the betas by taking the sum of the products of every pairwise residual with its corresponding x_{ij} observation which was taken from the training set.

Soft Thresholding

We used soft thresholding to decide if the current estimate of each β is optimal after introducing the relevant penalty terms, λ_1 only for LASSO and both λ_1 and λ_2 for the Elastic Net. The soft thresholding is implemented as:

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

This formula can be used equally in the LASSO Regression and the Elastic Net, since the only difference between the two is the addition of the penalisation with λ_2 in the Elastic Net. In case we are doing LASSO, λ_2 would be equal to 0, and the formula would do the soft thresholding update for LASSO.

Convergence

We repeat this cycle until convergence. Inside our *coord_descent* function we included an $ifelse$ condition to check if there has been a significant change in our β's. At the point where there is no change in the first 5 decimal places of each β we class this as convergence and record the resulting β's.

Model Validation

In order to select the optimal λ_1, and, if applicable, λ_2, for our model, we ran a validation with our self-generated data. For each model, we determined the β coefficients through the coordinate descent algorithm for a range of λ from 0 to 4. We then computed the validation MSE, and selected the λ penalization terms that returned the lowest MSE as the optimal values.

We first did this by creating a sequence with R's built-in $seq()$ function from 0 to 4, storing all the validation MSEs with their respective λ and selecting the minimum MSE. Then we compared our results with the optimization function $optim()$ on the validation MSE. Because of computational limitations we can choose relatively few values for λ, all of which are evenly spaced as per R's $runif()$ function. The $optim()$ function returned the global minimum, and the difference to our result was insignificant. Due to the computational cost of using the $optim()$ function, we decided to proceed with our self-developed function for the rest of the project.

RandomSearch is computationally more efficient than GridSearch, and also more statistically robust. Hence we produced our final λ range by sampling from a uniform distribution from 0 to 4, which gave more precise values, similar to the optimization function and as opposed to the initial $seq()$ function. Figure 1 shows the relationship between the $\hat{\lambda}_1$ penalty

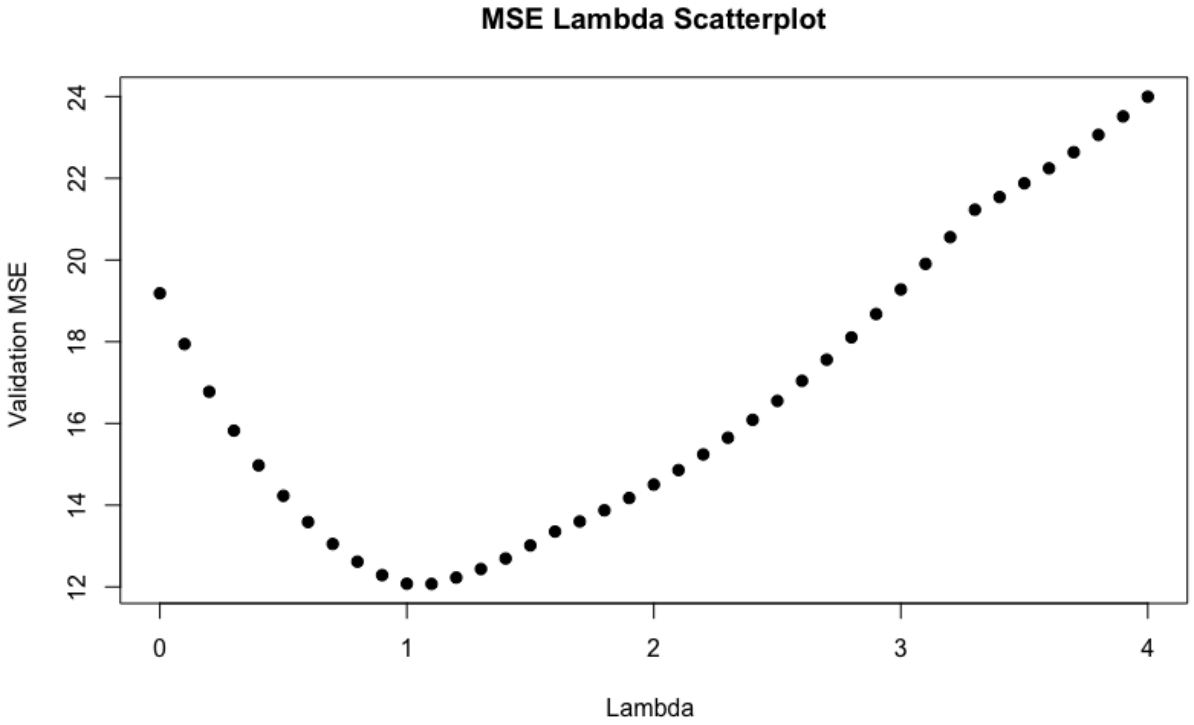

Figure 1: Validation MSE vs. corresponding λ_1 for the LASSO Regression

term for LASSO and the corresponding validation MSE. For the model given in the instructions, a value of approximately 1 resulted in the lowest validation MSE of approximately

12.

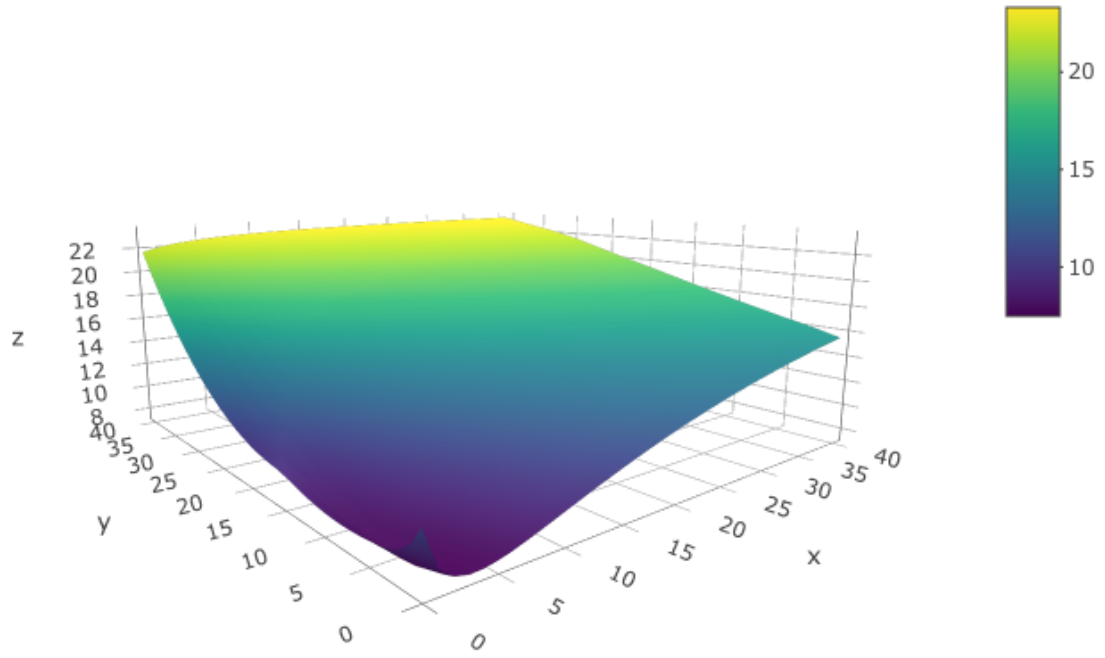

Figure 2: Validation MSE vs. corresponding λ_1 and λ_2 for the Elastic Net Regression

Figure 2 shows the relationship between the $\hat{\lambda}_1$ and $\hat{\lambda}_2$ penalty terms for the Elastic Net Regression, and the corresponding validation MSE. For the model given in the instructions, values of approximately 0.1 for $\hat{\lambda}_1$ and 0.3 for $\hat{\lambda}_2$ resulted in the lowest validation MSE of approximately 7.5.

After performing the validation we can already see that the Elastic Net has a better prediction capacity, as its minimum MSE is below substantially lower than the one of LASSO.

Simulations

Initially we used the default values described in the instructions and specified again above. Table shows the results for both LASSO and Elastic Net.

Testing Results	Lasso			Elastic Net		
n	MSE	MSE st. error	# vars	MSE	MSE st error	# vars
200	14.57	4.12	3.99	13.32	3.04	5.76

Consistently with what we saw in the validation errors, the Elastic Net has a better performance than LASSO as measured by the Test MSE, although the difference in this case seems to be lower than in the validation. Moreover, the Elastic Net appears to be more

consistent as its Test MSE has a lower standard deviation (shown by the MSE st. error). We can also see that LASSO shrinks the variables more, since on average it produces models with 4 variables, whereas the Elastic Net produces ones with close to 6. In the following we will examine the model results after changing the initial parameters to compare both in more detail.

Altering Parameter Values

Changing σ

σ was used to change the level of noise in the Y values. σ is a scalar value of a Normally Distributed noise vector with mean 0 and variance 1. Increasing sigma caused an increase in the noise associated with Y and hence increased variation in our Y values. This in turn caused the average test MSE to increase as σ increased.

LASSO and EN have similar MSE - although those for EN are slightly lower, especially the lower σ. As we saw earlier, EN is considerably better than LASSO when we look at the standard errors on the MSE results, as it has a lower st. error for all levels of sigma. Further to this the # of non-zero vars estimated have a lower st. error for EN than LASSO.

In conclusion, EN performs better than LASSO when there is large variance in the original dataset, which is caused here by scaling the noise vector. The MSE are comparable but the st. error is much lower, which should give us more confidence in EN as a model than LASSO.

	Lasso				Elastic Net			
σ	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
0	2.83	3.02	4.52	1.97	2.27	1.78	6.64	1.42
1	4.08	3.40	4.12	1.80	3.10	1.85	7.06	1.10
2	8.64	4.66	4.80	2.05	7.03	2.09	6.86	1.25
3	14.87	4.00	4.42	1.93	13.03	2.48	7.00	1.47
4	23.60	5.36	3.78	1.90	21.10	3.63	6.26	1.79
5	32.30	6.17	3.82	2.18	31.54	5.72	6.73	1.66
6	44.89	7.36	3.83	2.02	44.22	5.40	6.12	2.16

Table 1: Values were calculated using the *sigma_loop_L* and *sigma_loop_EN* functions.

Changing total n

We can see that as $n \to \infty$ that the difference between LASSO and EN stays consistent, and both their errors decrease. However, we would struggle to run $n > 4000$ as the computation time for $n = 4000$ was approximately 18 minutes. It is likely that test MSE would have continued to decrease for both LASSO and EN, however, we experience diminishing returns. It was often the case that the MSE of LASSO was marginally lower than EN, and in 5 out of 6 cases had a marginally lower st. error also. However, when we look at the number of non-zero vars EN is much more consistent with it's estimations of the number of variables.

	Lasso				Elastic Net			
No of obs (n)	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
40	11.76	2.33	5.34	2.02	11.61	1.70	6.82	1.29
200	9.46	0.43	5.4	1.48	9.82	0.52	7.18	0.69
500	9.18	0.21	5.92	1.88	9.67	0.31	7.14	0.73
1000	9.09	0.15	5.86	2.13	9.27	0.26	7.06	0.79
2000	9.05	0.09	5.48	2.10	9.21	0.17	6.88	0.69
4000	9.02	0.07	6.36	2.29	9.18	0.13	6.88	0.77

Table 2: Values were calculated using the *n_loop_L* and *n_loop_EN* functions.

This may be expected though as EN does not have the ability to reduce variables to exactly zero, like LASSO does.

Changing train/validation proportion split

Initially the number of observations dedicated to training and validation was 20 and 20 respectively out of 240 observations (the remainder were dedicated to the testing set). This gave us a 50/50 split of potentially seen data to use for training and validation. This is not a drastically poor split but the common consensus is to use a 75/25 or 80/20 split instead.

We kept the testing set size consistent through all tests at $10n$ as per the default ratio.

	Lasso			Elastic Net		
train/val split %	MSE	MSE st. error	# vars	MSE	MSE st error	# vars
25/75	18.09	5.37	3.22	14.70	3.49	5.58
50/50	14.54	3.86	3.90	12.27	2.63	5.90
75/25	12.98	3.61	4.40	12.70	2.73	5.38
90/10	13.49	4.25	4.52	13.59	5.45	5.82

Table 3: Values were calculated using the *split_loop_L* and *split_loop_EN* functions.

We can see that when a small amount of the data is dedicated to training EN performs better than LASSO - it also has a lower st. error. Both perform worse than in the default case of a 50/50 split, however, and LASSO sees a major gain when going from 25/75 to 50/50. As we dedicate more of the data to training, rather than validation, the models begin to agree with each other. The optimal train/validation split is between 50/50 and 90/10 for LASSO - as this is where it achieved its lowest test MSE and lowest MSE st. error. In real-world situations it is common to see a 75/25 or 80/20 split so this result was expected.

Changing number of dataset simulations

The results for both models did not substantially change as we increased the number of datasets (each of the default size of 200). Both LASSO and EN did experience their lowest MSE at the 100 level but not a large improvement over the default.

	Lasso			Elastic Net		
# datasets	MSE	MSE st. error	# vars	MSE	MSE st error	# vars
10	13.83	3.70	4.80	14.05	3.02	6.00
50	14.15	4.14	4.24	13.31	3.07	5.88
100	13.72	4.00	4.00	13.00	3.08	5.90
200	14.57	4.12	3.99	13.32	3.04	5.76
400	14.32	4.17	4.21	13.36	3.25	5.76

Table 4: Values were calculated using the *dataset_loop_L* and *dataset_loop_EN* functions.

Changing the sparsity of the β coefficients

Reducing the number of non-zero β's increased the calculation requirements of both models. It naturally increased the number of variables that could have an effect on the partial residuals. Hence both models suffered as we increased the number of non-zero β's (shown by moving down Table). LASSO performs particularly better when there is a higher number of zero β. Due to its propensity to shrink coefficients to 0 more easily, such a thing would be expected. Because the Elastic Net includes the same penalization method as LASSO, its results are similar in settings with a high sparsity of coefficients. As we increase the number of non-zero β's EN, proves a better prediction model because its error suffers less. It also seems to become very confident on it's estimation on the number of those β, as shown by the decreasing *# vars st. error*.

	Lasso				Elastic Net			
No of $\beta = 0$	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
6	12.37	3.11	4.20	0.70	12.40	2.82	5.00	1.71
5	14.94	4.19	3.52	1.80	12.70	2.92	5.80	1.58
4	17.78	6.89	4.82	1.55	15.92	4.25	6.74	1.35
3	17.99	5.31	4.88	1.85	15.56	4.30	6.90	0.89
2	29.96	13.51	4.88	1.97	18.53	5.36	7.44	0.76
1	27.06	14.25	5.84	1.65	20.67	6.22	7.66	0.59

Table 5: Values were calculated using the *beta_L* and *beta_EN* functions.

Cases when p > n

We generated 3 sets of 20 β coefficients, each one randomly sampled, from 0 to 9, all of them integers:

- β_{set_1} = (2, 9, 9, 7, 6, 0, 0, 3, 3, 2, 0, 7, 4, 3, 7, 1, 7, 0, 1, 2)
- β_{set_2} = (7, 0, 7, 0, 2, 7, 1, 9, 2, 9, 8, 4, 9, 5, 8, 4, 8, 9, 4, 1)
- β_{set_3} = (4, 3, 6, 6, 0, 0, 8, 3, 2, 1, 6, 7, 1, 8, 7, 0, 6, 8, 6, 7)

We tested the models in a setting where there are more predictors than data entries. LASSO struggled to calculate a meaningful MSE and become saturated quickly as the number of β exceeded the number of observations, and EN was naturally worse when $p > n$, but was still able to compute a result.

	Lasso				Elastic Net			
β_{set}	MSE	MSE st. error	# vars	# vars st. error	MSE	MSE st. error	# vars	# vars st. error
1	4e+303	N/A	12.57	5.96	236.28	86.97	18.46	2.85
2	5.07e+304	N/A	15.66	5.80	370.53	126.84	19.10	1.67
3	5.86e+303	N/A	14.55	6.08	306.74	101.81	18.96	1.80

Table 6: Values were calculated using the *beta_L* and *beta_EN* functions.

Although the results for the Elastic Net would not be useful with these results, we found it worthwhile to compare the results to a Ridge Regression, as it uses the same penalizer λ_2 as the Elastic Net. Running the tests with the same 3 sets of β-coefficients, we see that the results are very similar. If anything, the Elastic Net has a somewhat lower standard deviation in its Test MSEs, and tends to have a slightly lower error overall. It is also worth noting that Ridge produces a model with 20 coefficients - it doesn't shrink any to zero. The Elastic net stays in the middle ground between Ridge and LASSO in this aspect. It does perform a more aggressive variable shrinkage than Ridge by bringing some of the coefficients to 0. LASSO here comes clearly short as it cannot produce more coefficient estimates than the amount of entries n for the training data. Ridge is also not optimal, since it produces 20 estimates even if in the original ones we have some that equal 0.

	Ridge			
Trial	MSE	MSE st. error	# vars	# vars st. error
1	238.03	108.97	20	0
2	392.21	157.15	20	0
3	302.26	116.48	20	0

Table 7: Values were calculated using the *beta_R* function.

These last two examples help clearly illustrate the advantages of the Elastic Net. In a setting with high sparsity, it delivers results similar to LASSO, which is the better performer in these settings compared to Ridge. In a setting like the one we just saw where Ridge is better than LASSO, the Elastic Net's results are very similar to those of Ridge. By combining the penalization methods that each uses, one or the other takes more influence depending on which is more useful for the given data.

Study Limitations

When changing one of the parameters, n, train/validation size, and σ, we kept the other parameters at their default values. This is a limitation of our study as it is possible that other combinations had lower average test MSE's due to the interactions between parameters. However, this would have lead to an exponential number of computations.

$$\sigma^{n^{lambda1^{lambda2^{t/v \text{ split}}}}}$$

Where:

- σ denotes the number of sigma values tested
- n denotes the number of n values tested
- $lambda1$ denotes the number of lambda1 values tested
- $lambda2$ denotes the number of lambda2 values tested
- t/v split denotes the number of splits

ANN for Test MSE

For an extension of this project we could have created an Artificial Neural Network (ANN) to regress the parameters we just dealt with (number of datasets, size of training and validation sets, train and validation split, sigma, and the different settings for the coefficients) on $test\_MSE$. This would have allowed us to try further combinations of parameters and explore more scenarios to see in which does either model perform better. From here we could calculate the estimated effect of changing any of those parameters to partials of the actual values we used.

Conclusion

Elastic Net deals better with noise in datasets, as we saw that it performs better at both small and large values of σ in absolute MSE terms and st. error of the MSE. EN performs better in the default case than LASSO, and an added benefit is that it does have a lower st. error in the Test MSE. When we changed other parameter, such as the size of the training and validation datasets, the number datasets on which a test was run, or we tried a better train/validation proportion, both models improved their, and generally the superiority of the Elastic net over LASSO held. Both models suffered when $p > n$ but the Elastic Net was still able to run predictions in this setting, while LASSO wasn't. The model trial in a scenario with more predictors than data entries was particularly useful in seeing how the Elastic Net takes advantage of the best characteristics of both LASSO and Ridge, while generally outperforming both in most cases.