Optimization Project 3: Non-Linear Programming

Group 23:
Aakash Dhruva (avd667)
Jui-Jia (Jessica) Lin (jl82983)
Sai Lakshmi (sa57456)
Teagan Milford (tm38292)

Project Overview

In the past, direct variable selection has been replaced with LASSO and ridge regression because of its computational complexity. However, recent developments in optimization software have increased the efficiency of variable selection through the use of mixed integer quadratic programs (MIQP). There is debate about whether applying LASSO or conducting direct variable selection is preferable for predictive analytics, calling for further investigation to occur.

The LASSO method adds in a 'shrinkage' component that may cause it to be more effective at identifying the ideal beta variables to include in regression. Conversely, direct variable selection uses mixed integer quadratic programs with binary variables to identify the betas.

In this report, our consultants conducted an inquiry to analyze the performance of LASSO compared to a direct variable selection optimization method. To do so, we ran cross validation to identify the ideal k value and lambda value λ. Then, using the identified values, consultants fit betas on a training set to prepare them for prediction on test data, which would then be compared to the actual value of y.

This report also considers the advantages and disadvantages of the two methods to inform decision making and process improvement for future projects.

Problem Definition

This report defines and addresses the following problems:

1. **Objective:**

   Our main objective is to evaluate which of the two methods - direct variable selection and LASSO - perform better on simulated data.

   We used the following approaches to achieve our objective.

1. ***Direct Variable Selection (Mixed Integer Quadratic Programs (MIQP))***
   Mixed integer quadratic programming is a type of non-linear programming that minimizes the ordinary least squares loss function. In this technique, some decision variables take on discrete binary values. We applied it for direct variable selection using the optimization software Gurobi. Constraints were considered using the Big M method with binary variables.

2. ***Indirect Variable Selection (LASSO)***
   LASSO regression uses shrinkage to penalize betas, forcing some to become zero for variance reduction. Notably, this approach prevents overfitting through these penalties. We applied it for indirect variable selection using the package Scikit-learn.

Datasets

We used two datasets: training and testing. These simulate industry data so that mock analytics can be played out and the models evaluated. Each dataset contains 50 columns of *x* values and one column of *y* values, with 250 rows of data.

	y	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X41	X42	X43	X44
0	8.536145	-1.535413	0.718888	-2.099149	-0.442842	-0.598978	-1.642574	0.207755	0.760642	0.575874	...	0.361866	1.793098	-0.631287	-0.061751
1	4.808344	-1.734609	0.551981	-2.147673	-1.552944	1.514910	-1.143972	0.737594	1.321243	-0.261684	...	-0.677985	-0.165679	0.065405	0.137162
2	-1.530427	0.097257	0.107634	-0.194222	0.335454	-0.408199	0.133265	0.706179	0.394971	-0.437116	...	1.108801	0.333791	0.282055	-1.086294
3	-0.428243	-0.067702	0.557836	0.700848	-1.121376	1.722274	0.613525	0.700909	-0.417976	1.069749	...	0.692511	-0.350990	0.624558	0.434520
4	0.566694	0.488729	0.211483	0.568389	0.646837	0.163868	-0.002152	0.125137	0.493571	1.705451	...	-0.000605	1.075280	0.182281	-1.138458

5 rows × 51 columns

Figure 1. First 5 rows of training data.

	y	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X41	X42	X43	X44
0	7.107949	-2.047008	0.260132	-0.785152	0.384576	-0.137613	-0.364566	-1.941253	-0.108180	-0.339223	...	-0.522194	1.861897	0.124511	1.144071
1	5.796272	-1.354328	-1.289884	1.321533	-0.091165	-1.021874	0.686778	0.089737	-0.398371	-0.261740	...	-0.502578	0.584476	0.680703	0.046788
2	1.598651	0.502205	1.062887	1.460733	-1.506403	0.721664	0.528921	-0.699541	-0.270331	-2.544812	...	-0.125195	-1.292924	0.411785	-0.164210
3	2.532953	0.222381	-0.960747	-0.113762	1.935927	0.969387	-1.641403	0.026647	0.747138	-1.571784	...	-0.546915	-0.192517	0.603420	-0.277331
4	0.590685	1.209949	1.586874	-0.694784	-0.226370	-0.152108	0.772356	-0.573741	-0.992492	-0.646661	...	-0.982236	-1.407777	0.094211	0.159960

5 rows × 51 columns

Figure 2. First 5 rows of test data.

Direct Variable Selection: Mixed Integer Quadratic Programming (MIQP)

Objective

Our objective is to minimize the sum of squared errors, subject to the constraints that if z_j is zero, β_j is zero, and that there are at most *k* variables from *X*.

Mathematically, our objective function is posed as:

$$\min_{\beta,z} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im} - y_i)^2$$

Subject to the constraints:

$$s.t. -Mz_j \leq \beta_j \leq Mz_j \quad for\ j = 1, 2, 3, \ldots, m$$

$$\sum_{j=1}^{m} z_j \leq k$$

$$z_j \ \ are\ binary.$$

We used the Big M method for this approach, choosing our value for (-)*M* through trial and error.

Cross Validation

Prior to conducting direct variable selection, we completed 10-fold cross validation on the training dataset to find the *k* value with the smallest error. Since the dataset has 50 columns of *x* values, we tried values in increments of 5, from *k* = 5 to *k* =50. In calculating errors, we used the standard ordinary least squares equation.

	5	10	15	20	25	30	35	40	45	50
0	67.950555	50.156498	48.831876	55.361403	46.349798	53.778661	52.484956	56.079684	55.774874	55.993707
1	58.298479	64.238785	65.686587	65.578146	67.407667	76.940609	68.974814	70.407652	69.771315	69.810891
2	120.666700	83.146253	90.495792	97.009772	85.597307	93.260234	93.000657	92.769381	88.890697	90.255008
3	78.868321	50.308842	57.039855	59.092508	68.019788	69.422700	69.450472	64.404232	67.126835	67.957822
4	96.748748	60.121252	64.420647	60.819870	62.648846	66.801709	66.828504	68.917645	68.834297	68.761508
5	79.624175	75.052506	83.519242	80.394320	73.675895	77.806545	75.380397	74.077759	72.142753	73.156372
6	122.628833	92.440100	107.881187	109.782015	112.938629	123.141146	131.331468	130.566554	130.482782	130.368640
7	125.666862	106.191431	118.249746	116.789472	116.975086	124.466835	127.285621	123.708364	126.137930	125.617639
8	120.569560	55.996135	58.485318	62.153304	61.393446	68.716998	69.624144	68.555412	72.137656	71.762293
9	79.386747	76.473602	75.547867	79.472617	78.042371	80.421577	81.552829	85.519930	83.641191	84.439226

Figure 3. Sum of squared error (SSE) per fold on the validation set, calculated for each k value in increments of 5 from k=5 to k=50.

From this the ideal value was identified as *k* = 10 with validation average SSE = 71.4125. We moved forward with this value for *k* and prepared to conduct variable selection.

	Avg SSE
5	95.040898
10	71.412541
15	77.015812
20	78.645343
25	77.304883
30	83.475702
35	83.591386
40	83.500661
45	83.494033
50	83.812311

Figure 4. The average sum of squared error (SSE) on the validation set, calculated for each k value in increments of 5 from k=5 to k=50.

Evaluation on Test Set

To evaluate performance, we fit the MIQP model to the training set using $k = 10$ to identify the intercept and coefficients (10 variables).

```
array([ 0.97252408,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        , -2.30820726,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.51832612, -0.20416201,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -1.55914318,  0.86697336,
        0.        , -1.31191942,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.4081653 ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        1.78147489,  0.        ,  0.88738292, -0.28229213,  0.        ,
        0.        ])
```

Figure 5. Coefficients from MIQP when k = 10.

These betas were then used to make a prediction on the test data.

```
[6.179858781443058, 5.0952429868404385, 3.285595322405015, 3.7584853876612683, -0.3329752585827511, -5.142736834668355, -3.1
445435672467617, -1.2380628758189667, 1.3851109276642963, -0.4417385447810917, -1.6950022467195103, 2.730350268182774, 0.747
4490344779042, -0.9719223177248238, -0.6868152824269131, 8.045223814607219, -7.946984710542041, 3.8906397391997807, -4.58142
91889682535, -3.2199208247153965, -2.1621145380309468, 3.216863177324629, -3.19810532637421, 0.1974073062532814, -2.35988844
3862833, -0.4199988547491114, -1.9125216027358696, -3.324185867280889, -3.141709717188676, -3.5537932361104727, -1.808425425
049101, -0.37134300556514743, 1.8670808021748821, 5.0492788628390715, -1.8000561410732652, 3.0942767463995833, 4.38154309043
227, 2.6988626993835276, 1.6132885975447508, 5.975846372424301, -1.1973582998105734, 5.223254198055286, -5.848998908490297,
-1.1446152790055146, 4.518029975345616, 4.187748655603142, 4.120460083021932, 0.614838088600514, 1.9572324582774197, -1.5490
438298035043]
```

Figure 6. Prediction of y values in the test set using MIQP.

With these conditions in place, the SSE using MIQP on the test data is 116.8272.

	SSE	MSE	R_Squared
Gurobi_Method_Metrics	116.827198	2.336544	0.858668

Figure 7. Measures of error on test data for MIQP when k = 10.

Indirect Variable Selection: LASSO

Objective

For indirect variable selection, our objective is to minimize the loss function through LASSO regression, which uses shrinkage to prune variables.

Mathematically, our objective function is posed as:

$$\min_{\beta} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im} - y_i)^2 + \lambda \sum_{j=1}^{m} |\beta_j|,$$

Where λ is a hyperparameter chosen using cross validation.

Cross Validation

Before fitting a LASSO model, we conducted 10-fold cross validation on the training dataset to pick the best value for λ. For cross validation, we used the built in feature LassoCV from the Scikit-learn package.

The best lambda was identified as λ = 0.0764 with average validation SSE=69.5007.

Evaluation on Test Set

Using this value, we fit the LASSO regression model on the training set.

We obtained the intercept and coefficient values (17 variables) to use for prediction on the test set.

```
array([-0.        , -0.        ,  0.        ,  0.        , -0.        ,
        0.        , -0.        , -0.        , -2.16054765,  0.        ,
       -0.05964031, -0.        , -0.        , -0.        , -0.41912484,
       -0.19325408,  0.        ,  0.        , -0.        ,  0.        ,
        0.        , -0.19517759, -1.36388738,  0.7425965 , -0.        ,
       -1.30481574, -0.        ,  0.        ,  0.05798283,  0.        ,
       -0.        ,  0.        , -0.09737839,  0.28341629,  0.        ,
        0.        ,  0.        ,  0.        , -0.23157873,  0.        ,
       -0.        ,  0.        ,  0.        ,  0.03078191,  1.56362172,
       -0.02160033,  0.69992447, -0.09289745,  0.        ,  0.        ])
```

Figure 8. Coefficients from LASSO when $\lambda = 0.0764$.

```
array([ 6.07686351,  4.9181073 ,  3.22777954,  3.57138566, -0.41849943,
       -4.94878307, -2.8218873 , -1.49522305,  1.38296395, -0.24746403,
       -1.94701631,  2.70749465,  0.64484559, -0.50918617, -0.31915912,
        7.37584986, -7.54781954,  3.59673093, -4.39624331, -2.98890566,
       -1.94423698,  3.33248953, -2.42122798,  1.13315219, -2.53307168,
       -0.15595162, -1.62271875, -2.31420693, -3.46402012, -3.71707635,
       -1.66827581, -0.07491137,  1.40363987,  5.45269115, -1.20270399,
        2.37325349,  4.74719816,  3.13015173,  1.55100085,  5.85587702,
       -0.81232674,  4.66231732, -5.71548761, -1.20927664,  4.25279319,
        4.09734242,  3.8335482 ,  0.61787077,  1.89407804, -1.02592954])
```

Figure 9. Prediction of y values in the test set using LASSO.

From this, we identified the SSE of the test data to be 117.4817.

	SSE	MSE	R_Squared
Lasso_Metrics	117.481738	2.349635	0.857876

Figure 10. Measures of error on test data for LASSO when $\lambda = 0.0764$.

Comparison of the Performance of Direct and Indirect Variable Selection

Now that models have been run on the test data, we can compare the performance of each of the approaches.

As evidenced in Figure 11., the MIQP method appears to perform better when considering the SSE and MSE. Notably, the differences between the measures of error are very small. This suggests that the models are approximately equally effective for the purpose of variable selection.

	SSE	MSE	R_Squared
Gurobi_Method_Metrics	116.827198	2.336544	0.858668
Lasso_Metrics	117.481738	2.349635	0.857876

Figure 11. Measures of error on test data for MIQP when k = 10 and LASSO when λ = 0.0764.

Our evaluation suggests that, based on our training and test data, either approach would be suitable for meeting the needs of the firm.

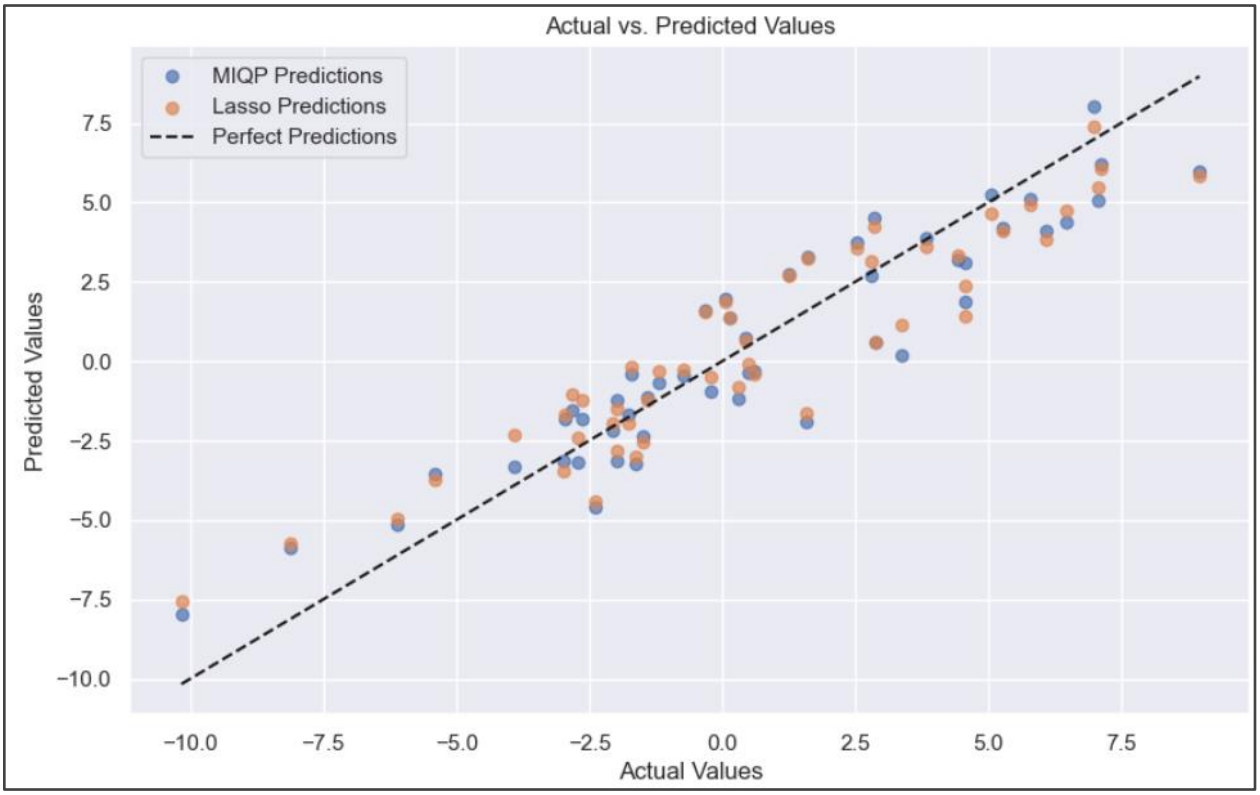

Figure 12. Predicted and actual y values with MIQP and LASSO approaches.

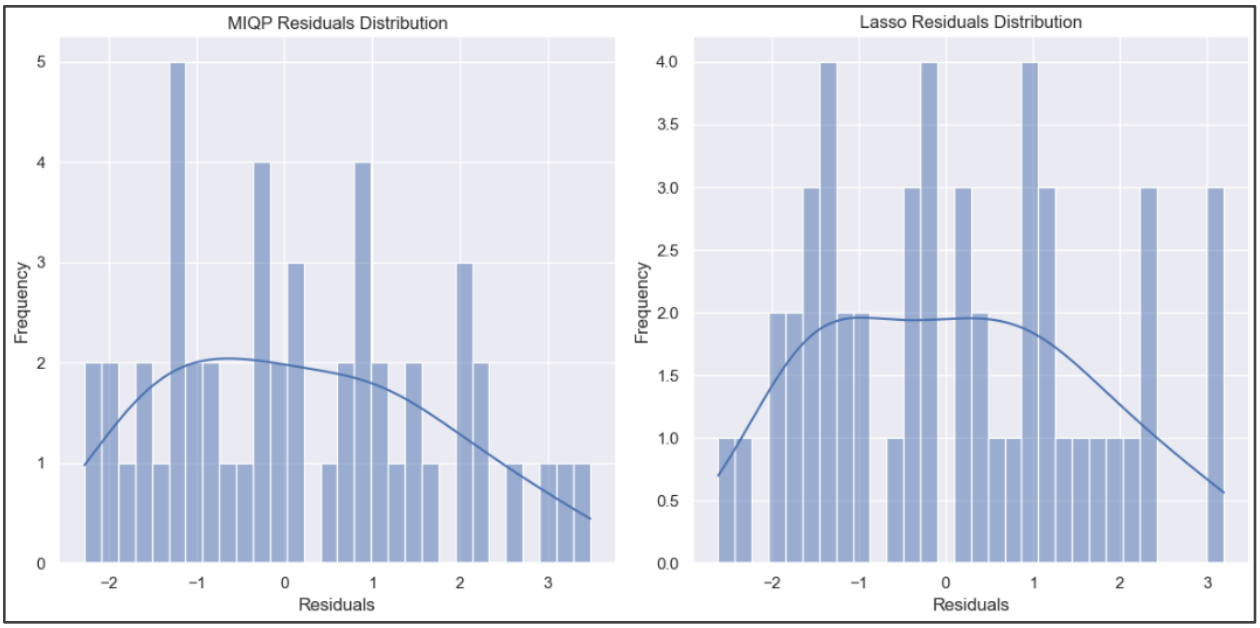

Figure 13. MIQP and LASSO residual distributions.

Figure 12. shows the performance of each approach in comparison to the actual values of y in the test set. This figure suggests that the approaches are comparable in their performance, as neither has an apparent advantage over the other in predictive ability. The residual distributions of each approach are shown in Figure 13., revealing a similar trend between the two.

With all of this in mind, we considered the advantages and disadvantages of both methods so that we could make an informed recommendation for which approach to use in future projects.

MIQP: Advantages and Disadvantages

Advantages

Using the MIQP approach allows us to use less features while still yielding a low SSE. Figure 14 below shows that as the number of features *k* approaches the optimal value 10, the amount of error drops considerably.

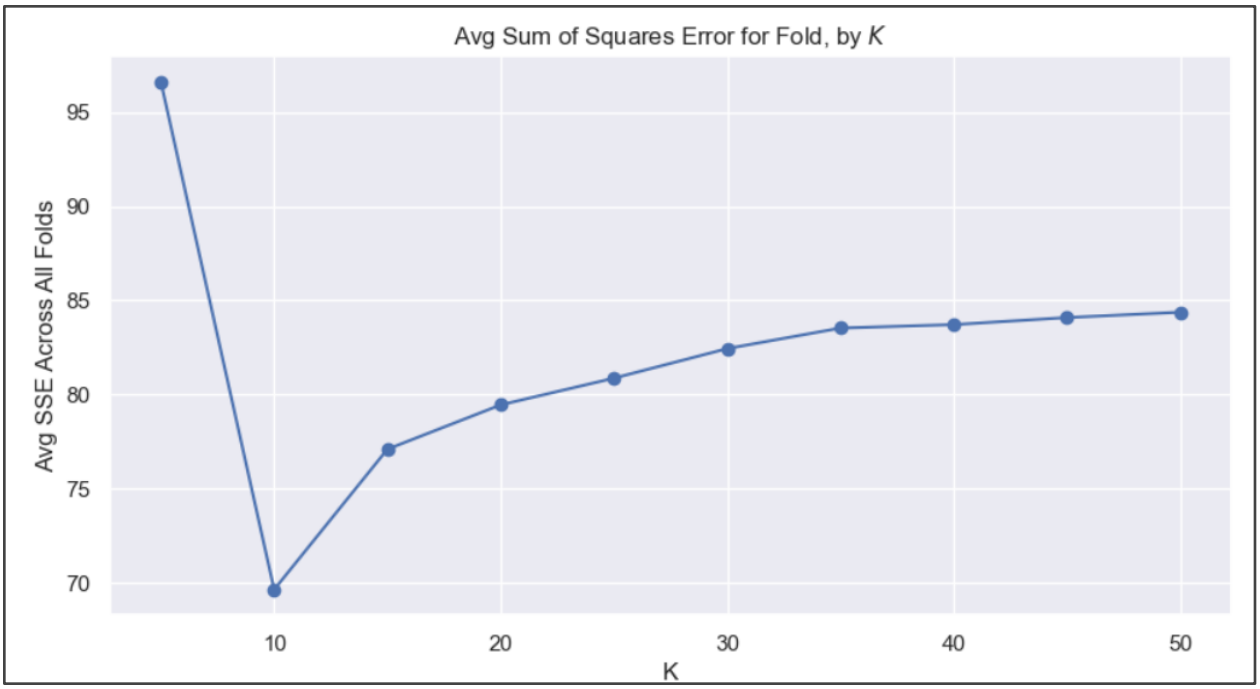

Figure 14. Average sum of squared errors (SSE) for each incremental value of k.

MIQP also offers more flexibility than LASSO because it can handle both discrete and continuous variables. This makes it widely applicable for various optimization problems outside of variable selection.

Disadvantages

Generally, MIQP is more computationally expensive than LASSO because of the complexity of handling integer variables. This is especially a problem for larger scale problems, which must be kept in mind while considering its use for other projects.

LASSO: Advantages and Disadvantages

Advantages

LASSO tends to be computationally more efficient than MIQP, especially for large datasets. Its linear nature makes it scalable and relatively fast. Also, by shrinking coefficients, LASSO effectively reduces model complexity and prevents overfitting. This is crucial in predictive analytics where model generalization is important. As evidenced by Figure 15., when lambda increases, the average sum of squared errors (SSE) across all folds decreases to a certain point before it plateaus, indicating an optimal region of lambda values where the model performs best.

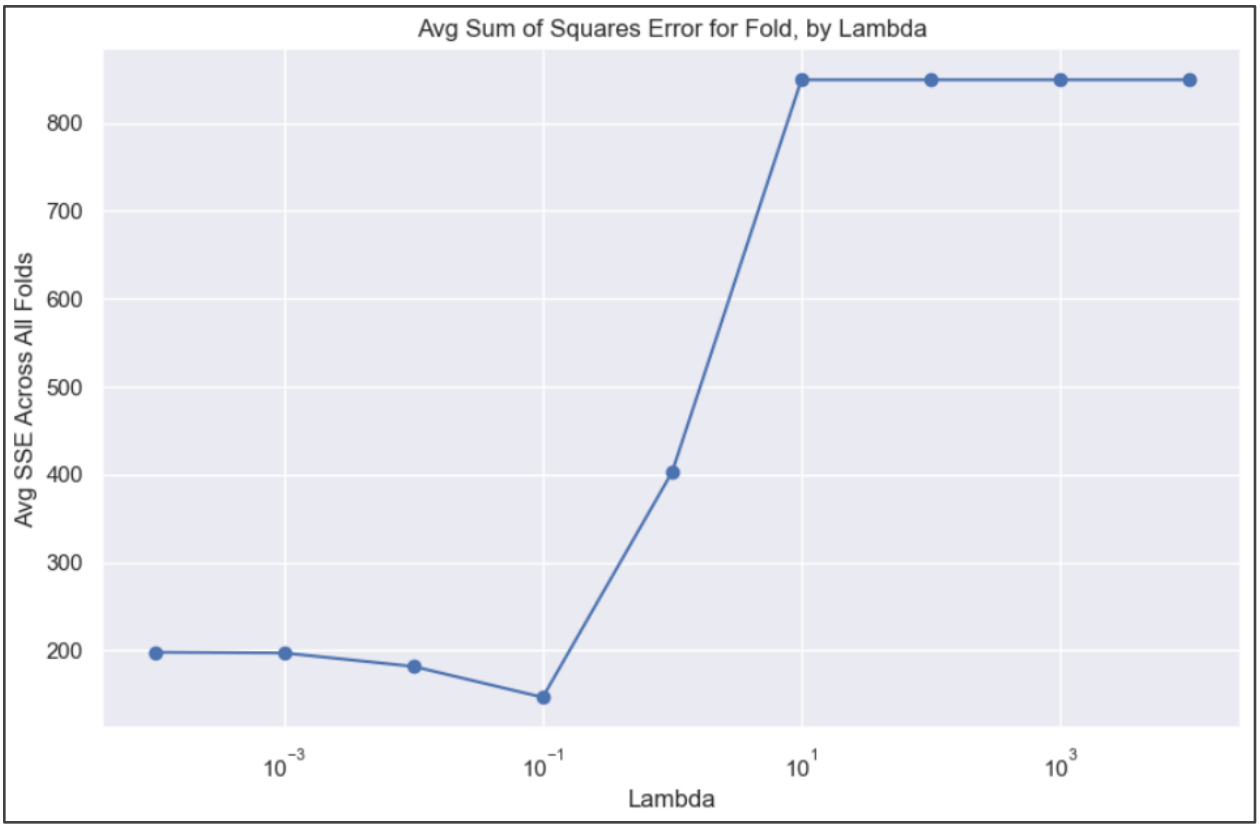

Figure 15. Average sum of squared errors (SSE) for each value of lambda.

Disadvantages

LASSO can introduce bias, particularly towards zero, which may affect the model's prediction accuracy, especially if the true relationship involves large coefficients. If there are groups of correlated variables, LASSO might randomly select only one variable from a group, ignoring others that might be equally or more important.

Conclusion

	# Features	SSE	MSE	R_Squared
Gurobi_Method_Metrics	10	116.827198	2.336544	0.858668
Lasso_Metrics	17	117.481738	2.349635	0.857876

Figure 16. Comparison for MIQP when k = 10 and LASSO when λ = 0.0764.

Based on Figure 16., here is a conclusion that can be drawn regarding when to use the Mixed Integer Quadratic Programming (MIQP) approach, as represented by the "Gurobi Method Metrics," and when to use the LASSO approach:

MIQP:

1. Precision is paramount: The MIQP method selected fewer features (10 vs. 17 with LASSO) while maintaining a comparable Sum of Squared Errors (SSE) and Mean Squared Error (MSE).
2. Model interpretability is crucial: With fewer features, models tend to be more interpretable, which is beneficial when explanations are as important as predictions.
3. Dataset size is manageable: MIQP can be computationally intensive; hence, it is better suited for smaller datasets where the computational overhead is not prohibitive.

LASSO:

1. Efficiency is a priority: LASSO is more computationally efficient, which is advantageous for larger datasets or when rapid model development is required.
2. A balance of feature inclusion and model performance is needed: LASSO selected more features but still maintained close performance to MIQP, indicating a more balanced approach.
3. Overfitting prevention is necessary: LASSO inherently includes regularization which can prevent overfitting, making it suitable for models where generalization to unseen data is critical.

Final Recommendation for Different Scenarios:

- For smaller or more specialized datasets where the interpretability of a smaller set of features is essential, and computational resources are less of a concern, MIQP could be the preferred choice.
- In scenarios where computational speed and model generalization are more critical, or when dealing with larger datasets, LASSO would be advantageous.
- In projects where the number of features is a consideration due to domain-specific knowledge or interpretability requirements, MIQP might be favored despite its computational intensity.

In conclusion, the choice between MIQP and LASSO should be guided by the specific analytical needs, data characteristics, and computational constraints of the project at hand. Both methods have their place in the toolbox of an analytics team, and the decision should be tailored to the context of each analytical task.