Project 3 - Variable Selection for Regression

(Nonlinear Programming)

Group – 22

Anudeep Kumar Akkana (aa92799), Snehal Naravane (sn27429), Prathmesh Savale (ps33296), Vishwak Venkatesh (vv8257)

Introduction

Modern technological advances have taken place in nearly every field of science and engineering which makes it easier to collect more data. With prediction being the goal of so many models, it has become extremely difficult to choose an appropriate subset of features to base the model on. Moreover, especially when there are more predictors than observations, the ubiquity is the danger of overfitting, which makes optimal variable selection even more important.

Variable selection is an important step in any regression model. It helps find relevant variables, prevents overfitting, understanding relationships among variables and also improves the accuracy of the overall model.

Requirements

The purpose of this report is to describe two different methods of variable selection for predictive analytics and decide which one best suits your company's objectives. The methods to be considered are direct variable selection using Mixed-Integer Quadratic Programs (MIQP) and everlasting LASSO regression.

Objective

The most common method of variable selection is the "shrinkage" component of the lasso regression. After a lot of improvements in the optimization field, now the variable selection can also be derived by optimization techniques with less computational difficulties. The objective of the project is to examine which method works better on variable selection by direct comparison on the same dataset.

Variable Selection Methods

Mixed Integer Quadratic Programming

For the Direct Variable Selection Method (MIQP) , we start by looking at a dataset of m independent variables X and a dependent variable y. The standard ordinary least squares problem is formulated as follows

Equation 1 :

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

In order to incorporate variable selection into this problem we include some binary variables z, that force the corresponding values of β to be zero if z is zero using the big M method, including at most k variables from X as shown below:

Equation 2 :

$$\min_{\beta,z} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im} - y_i)^2$$
$$s.t. -Mz_j \le \beta_j \le Mz_j \quad for\ j = 1, 2, 3, \ldots, m$$
$$\sum_{j=1}^{m} z_j \le k$$
$$z_j \ are\ binary.$$

By using the knowledge of linear algebra, we transformed our objective function into below format so it is easier for programming:

Equation 3 :

$$\min_{\beta,z} \beta^T (X^T X)\, \beta + (-2\, y^T X)\, \beta$$

Where β is an (m+1)*1 column vector that contains β_o, β_1……. β , X is an n*(m+1) matrix that has first column made up of only 1s and columns 2 to (m+1) made up of the independent variables and y is an n*1 column vector

The MIQP problem is formulated as follows :

Decision Variables :

1. β_j : coefficient of variable j (m+1)
2. z_j : tells us whether variable j should be considered or not in the regression (m) – Binary

Total Decision Variables – 2m+1

Objective Function :

$$\min_{\beta,z} \beta^T (X^T X)\, \beta + (-2\, y^T X)\, \beta$$

Constraints

1. Sum of all variable present in the regression equation should be equal to k

$$\sum_{j=1}^{m} z_j \leq k$$
$$z_j \text{ are binary.}$$

2. Coefficient of each variable j, where j>=1, βj should be less than M zj

βj - M zj <= 0 for j = 1,2,3…….m

3. Coefficient of each variable j, where j>=1, βj should be more than - M zj

βj + M zj >= 0 for j = 1,2,3…….m

Total Constraints – 2m+1

Mixed-Integer Quadratic Programs (MIQP):

We set the value of Big M as 15. This number should be big enough as we have seen that the values taken by the coefficients are much smaller than 15.

Then we created 10 cross-validation folds. For each value of k, we trained optimization models on 9 folds & then tested them on the remaining fold. Thus for each value of k, 10 models were trained.

By repeating this for each value of k from 5 to 50 resulted in 100 models trained.

The value of k with the least MSE was selected. *(refer to figure 1 under code snippets for the code)

As can be seen from the above plot, the least average MSE (cross-validation error) was at k=10.

Hence, we fitted the MIQP on the entire training set with k = 10.

The resultant non-zero MIQP coefficients were as follows:

	Coefficients
Intercept	0.972524
X9	-2.308207
X15	-0.518326
X16	-0.204162
X23	-1.559143
X24	0.866973
X26	-1.311919
X34	0.408165
X45	1.781475
X47	0.887383
X48	-0.282292

Then, we sorted by coefficients (excluding intercept) in ascending order, & filtered out features not selected in model as follows:

	Coefficients
X9	-2.308207
X23	-1.559143
X26	-1.311919
X15	-0.518326
X48	-0.282292
X16	-0.204162
X34	0.408165
X24	0.866973
X47	0.887383
X45	1.781475

From the above table, it can be seen that a unit change in X9 has the largest absolute change on Y by decreasing it by 2.31 units, while a unit change in X45 increases Y by 1.78 units.

Then, after running the model on test data, we got a test MSE of 2.34.

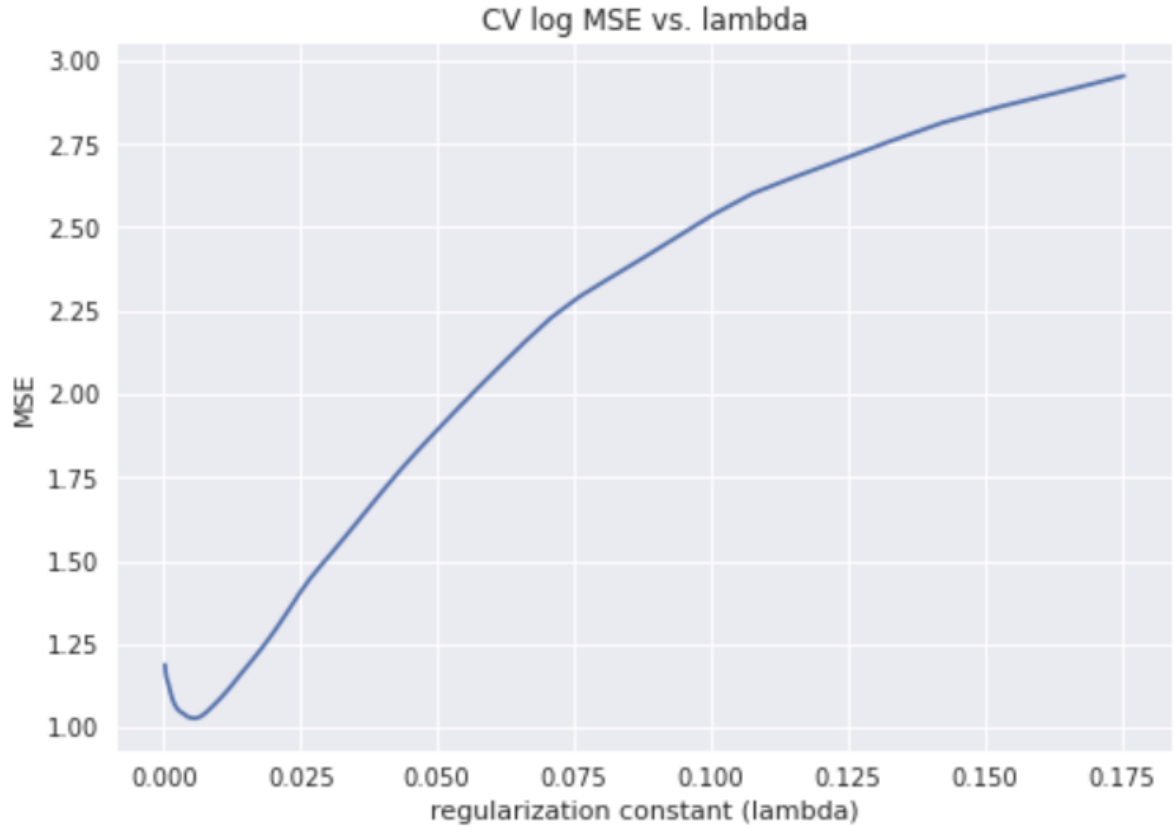
LASSO Regularization Method

LASSO, Least Absolute Shrinkage and Selection Operator, is a variation of the OLS regression model with regularization term that penalizes coefficients based on their absolute value. The indirect variable selection or LASSO Regularization method is posed as

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

The LASSO regularization model has the advantage of 'shrinking' the βs closer to zero, leading to variance reduction.Where λ is a hyperparameter that can be chosen during cross validation. Also, if λ is large enough, several values of β will be forced to be equal to 0.

Then we ran the lasso fit on the train data with 10-fold cross-validation, and obtained a penalization (lambda/alpha value in case of sklearn) of 0.006 (0.0057). Following is the plot of log (MSE) vs. different lambda (alpha for sklearn) values. *(refer to figure 2 in code snippets)



The lasso coefficients obtained were as follows:

	Coefficients
Intercept	1.006196
X9	-2.115615
X11	-0.060431
X15	-0.416745
X16	-0.181553
X22	-0.197102
X23	-1.365528
X24	0.735100
X26	-1.300186
X29	0.063903
X33	-0.107380
X34	0.253927
X35	0.021384
X39	-0.211595
X44	0.011523
X45	1.531715
X46	-0.014088
X47	0.650478
X48	-0.097579

Mixed-Integer Quadratic Programs (MIQP) and LASSO regression coefficients:

	MIQP	LASSO
Intercept	0.972524	1.006196
X1	0.000000	0.000000
X2	0.000000	0.000000
X3	0.000000	0.000000
X4	0.000000	0.000000
X5	0.000000	0.000000

X6	0.000000	0.000000
X7	0.000000	0.000000
X8	0.000000	0.000000
X9	-2.308207	-2.115615
X10	0.000000	0.000000
X11	0.000000	-0.060431
X12	0.000000	0.000000
X13	0.000000	0.000000
X14	0.000000	0.000000
X15	-0.518326	-0.416745
X16	-0.204162	-0.181553
X17	0.000000	0.000000
X18	0.000000	0.000000
X19	0.000000	0.000000
X20	0.000000	0.000000
X21	0.000000	0.000000
X22	0.000000	-0.197102
X23	-1.559143	-1.365528
X24	0.866973	0.735100
X25	0.000000	0.000000
X26	-1.311919	-1.300186
X27	0.000000	0.000000
X28	0.000000	0.000000
X29	0.000000	0.063903

X30	0.000000	0.000000
X31	0.000000	0.000000
X32	0.000000	0.000000
X33	0.000000	-0.107380
X34	0.408165	0.253927
X35	0.000000	0.021384
X36	0.000000	0.000000
X37	0.000000	0.000000
X38	0.000000	0.000000
X39	0.000000	-0.211595
X40	0.000000	0.000000
X41	0.000000	0.000000
X42	0.000000	0.000000
X43	0.000000	0.000000
X44	0.000000	0.011523
X45	1.781475	1.531715
X46	0.000000	-0.014088
X47	0.887383	0.650478
X48	-0.282292	-0.097579
X49	0.000000	0.000000
X50	0.000000	0.000000

In lasso:

Number of features driven to zero: 32

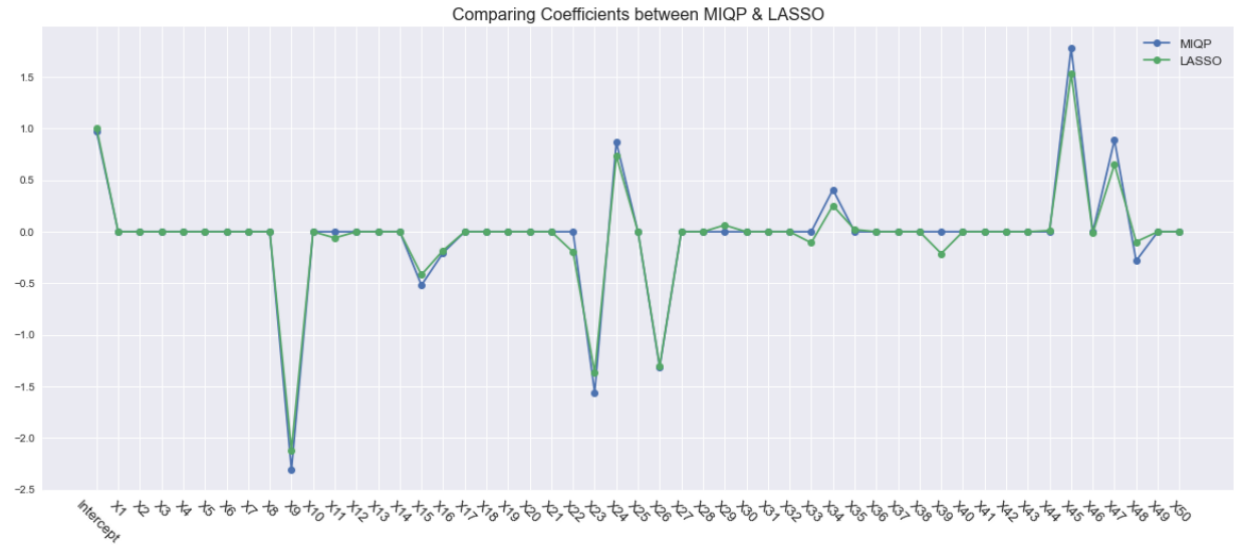
Number of features not driven to zero: 18

Mean squared error on the test set with lasso was 2.36.

Comparison and Conclusion

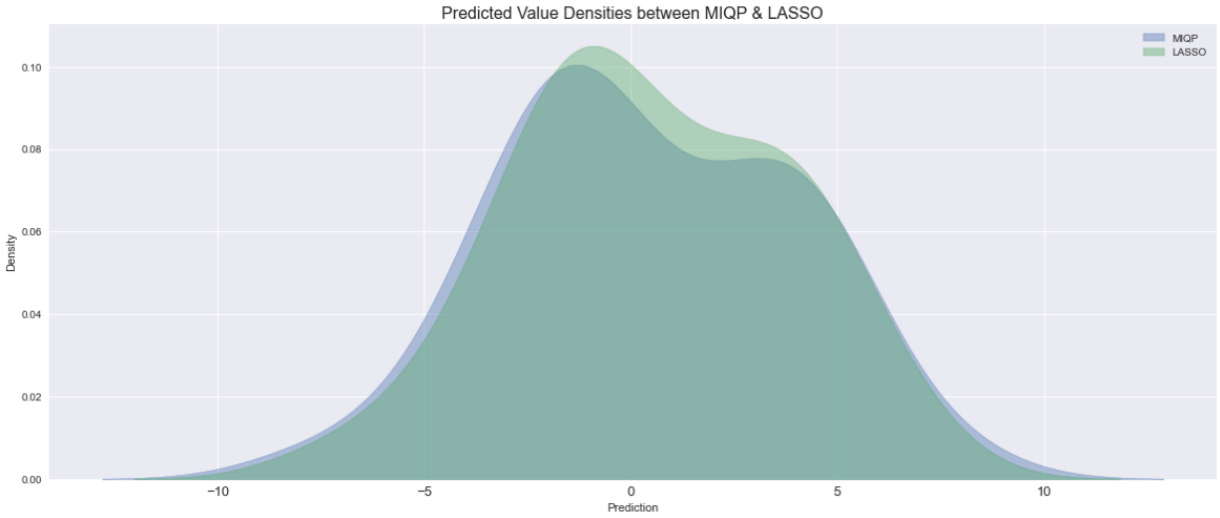
To better understand the difference between MIQP and LASSO methods, we compared their coefficients.

A comparison chart was created to show which variables were chosen by each selection method The optimal model in LASSO has 18 non-zero coefficients for variables while MIQP has only 10. For reference, both the direct variable selection method and LASSO choose variables X9, X15, X16, X23, X24, X26, X34, X45, X47, X48. In addition to those, the LASSO model also chose a further 7 variables which are X11, X22, X29, X33, X39, X44, and X46.



Both the methods gave comparable results and performance. If we consider from the perspective of runtime for both the methods, MIQP took more than 3 hours whereas LASSO took less than 5 minutes which is very quick comparatively. The MIQP method is very helpful in case we have too many variables in the dataset and we just need to select a few most important variables to make it more explainable to business teams. *(refer to figure 3 under code snippets)

Predicted Value Densities between MIQP & LASSO:



Further, the predictions of the dependent variable on the test dataset are very similar across both methods as shown above in density plots. The output of the 2 models are almost identical. This also aligns with the fact that the beta coefficients for both the models are identical. *(refer to figure 3 under code snippets)

Recommendations

1. If time of execution is important, then a Lasso regression can be preferred over MIQP. In case of a large feature space, it is sometimes very important to achieve quick baseline results. In these cases Lasso could be a handy tool as when there are more variables, MIQP will struggle to converge.
2. For ease of use, implementation of Lasso is readily available in popular programming languages like python and R and it can be easily accessed with minimal lines of code. Implementation of MIQP on the other hand requires knowledge of writing the constraints and formulating objective for a optimizer.

Based on the above, we can consider LASSO to be a better technique for variable selection over MIQP because it is easier to implement, interpret and quick to execute.

Code Snippets:

1. MIQP cross validation:

```
k_list = [i*5 for i in range(1,11)]
mse = []
for k in k_list:

    ind_mse = []
    for val_test_ind in folds:

        val_train_ind = np.delete(folds, np.argwhere(np.isin(folds.flatten(),val_test_ind)))

        X = np.ones((len(val_train_ind), m+1))
        X[:,1:] = np.array(train.iloc[val_train_ind].drop(['y'], axis = 1))

        # Quadratic term
        Q = np.zeros((2*m + 1, 2*m + 1))
        Q[:(m+1),:(m+1)] = np.transpose(X) @ X

        # Linear term
        c = np.zeros((2*m + 1))
        c[:(m+1)] = (-2 * np.transpose(np.array(train.iloc[val_train_ind]['y']))) @ X

        # Constraints
        # 2*m big M constraints + 1 k constraint
        A = np.zeros((2*m+1, 2*m+1)) # m+1 betas & m z variables
        A[0:m,1:m+1] = np.identity(m)
        A[0:m,m+1:2*m+1] = np.identity(m) * M
        A[m:2*m,1:m+1] = np.identity(m)
        A[m:2*m,m+1:2*m+1] = np.identity(m) * (-M)
        A[2*m,m+1:2*m+1] = 1

        # RHS
        rhs = np.zeros((2*m+1))
        rhs[2*m] = k

        # Direction of constraints
        sense = ['>'] * m + ['<'] * m + ['<']

        # Variable types - m+1 betas can take any values & m z should be binary
        vtypes = ['C'] * (m+1) + ['B'] * m

        # Lower bounds - lb for coefficients is -M & 0 for the z variables
        lb = np.array([-M] * (m+1) + [0] * m)

        # Solve the problem
        op_mod = gp.Model()
        op_mod_x = op_mod.addMVar(2*m + 1, vtype = vtypes, lb = lb)
        op_mod_con = op_mod.addMConstr(A, op_mod_x, sense, rhs)
        op_mod.setMObjective(Q,c,0,sense=gp.GRB.MINIMIZE) # Having Q in the quadratic terms & c for linear terms

        op_mod.Params.OutputFlag = 0
        op_mod.Params.TimeLimit = timeLimit
        op_mod.optimize()

        x_val = np.hstack((np.ones((len(val_test_ind),1)), train.iloc[val_test_ind].drop(['y'], axis = 1)))
        y_val_pred = x_val @ op_mod_x.x[0:m+1]
        y_val = train.iloc[val_test_ind]['y']
        ind_mse.append(mean_squared_error(y_val, y_val_pred))

    print('MIQP with', k, 'variables done')
    mse.append(np.mean(ind_mse))
```

2. Lasso:

```
X_train = train.drop(['y'], axis = 1)
y_train = train['y']
X_test = test.drop(['y'], axis = 1)
y_test = test['y']
```

```
lasso_cv = LassoCV(cv = 10, random_state = 1, normalize = True)
lasso_cv.fit(X_train, y_train)
```

```
LassoCV
LassoCV(cv=10, normalize=True, random_state=1)
```

```
# Penalization obtained from CV
lasso_cv.alpha_
```

```
0.0057453437864455085
```

```
# checking the MSE wrt different values of alpha(or lambda regularization parameter)
plt.plot(lasso_cv.alphas_, np.log(pd.DataFrame(lasso_cv.mse_path_).mean(axis=1)))
plt.xlabel('regularization constant (lambda)')
plt.ylabel('MSE')
plt.title('CV log MSE vs. lambda')
```

3. Comparison of Lasso and MIQP:

```
# Comparison of regression coefficients between MIQP & LASSO
plt.figure(3, figsize=(20, 8))
plt.plot(coefs.index, coefs['MIQP'], label = 'MIQP', marker='o')
plt.plot(coefs.index, coefs['LASSO'], label = 'LASSO', marker='o')
plt.xticks(rotation = -45, fontsize= 13)
plt.legend(loc='best',  fontsize=12)
plt.title('Comparing Coefficients between MIQP & LASSO', fontdict={'fontsize': 16})
plt.show()
```

```
# Comparing predictions between MIQP & LASSO
plt.figure(3, figsize=(20, 8))
sns.kdeplot(y_test_pred_miqp, alpha = 0.4, fill = True, label='MIQP', legend=True)
sns.kdeplot(y_test_pred_lasso, alpha = 0.4, fill = True, label='LASSO', legend=True)
plt.legend()
plt.xlabel('Prediction')
plt.title('Predicted Value Densities between MIQP & LASSO', fontdict={'fontsize': 16})
plt.xticks(fontsize= 13)
plt.show()
```