Machine Learning Lab (CS-353) Report

Lasso Regression

Team Members:

1. Shashank D 181CO248
2. Tarun S Anur 181CO255

1. Lasso Regression - Introduction

The Lasso is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients of features, effectively reducing the number of features upon which the given solution is dependent. Hence lasso can be used as a feature selection technique as well. Lasso regression is a type of linear regression that uses shrinkage where slope coefficient values are shrunk towards zero or large values. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. Lasso was introduced in order to improve the prediction accuracy and interpretability of regression models. It selects a reduced set of the known covariates for use in a model.

The acronym “LASSO” stands for Least Absolute Shrinkage and Selection Operator.

Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients (slope terms). This type of regularization can result in sparse models with few coefficients because some coefficients can become zero which can be eliminated from the model. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models. On the other hand, L2 regularization (e.g., Ridge regression where squares of coefficients are added as penalty) doesn’t result in elimination of coefficients or sparse models. This makes Lasso far easier to interpret than the Ridge as under certain conditions, it can recover the exact set of non-zero coefficients.

2. Lasso Regression - Algorithm

Mathematically, it consists of a linear model with an added regularization term. The objective function to minimize is:

$$\min_w \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha ||w||_1$$

The lasso estimate thus solves the minimization of the least-squares penalty with $\alpha||w||$added where $\alpha$is a constant and $||w||_1$ is the l_1 -norm of the coefficient vector.

A tuning parameter, α controls the strength of the L1 penalty. α is basically the amount of shrinkage.

When α = 0, no parameters are eliminated. The estimate is equal to the one found with linear regression (least squares method).

As α increases, more and more coefficients are set to zero and eliminated (theoretically, when $\alpha = \infty$, all coefficients are eliminated).

As α increases, bias with the training set increases.

As α decreases, variance of testing set output increases.

If an intercept is included in the model, it is usually left unchanged.

3. Pros and Cons of Lasso Regression

Pros:

- Select features, by shrinking co-efficient towards zero.
- It is fast in terms of inference and fitting.
- Avoids overfitting. It can also be applied where number of features is larger than the amount of data.

Cons:

- Selected features will be highly biased.
- For n<<p (n-number of data points, p-number of features), LASSO selects at most n features.
- LASSO will select only one feature from a group of correlated features, the selection is arbitrary in nature.
- For different bootstrapped data, the feature selected can be very different.
- Prediction performance is worse than Ridge regression.

4. Applications of Lasso Regression

- This method can be used in business platforms to estimate revenue and other financial related statistics.
- It can be used as a feature selection method as well.
- It can be used in Medical field to diagnose the blood level and other related parameters and in turn predict any major ailments.
- In the Economics field to predict high growth firms.
- In the Financial department to predict corporate bankruptcy.

5. Lasso Regression - Numerical Analysis

1. Dataset description
   salary_data.xls, a spreadsheet which contains total number of years of experience and the present salary of 30 employees in a company.

2. Numerical Solution

   - We shall first find out the mean of the independent variable x (years of experience in this example) and the mean of the dependent variable y (salary in this example).
   - We then calculate (xi-mean(xi)) and (yi-mean(yi)) for all the xi and yi in the dataset.
   - Then we calculate 1. (sum((xi-mean(xi))) * sum((yi-mean(yi)))) and 2. Sum((xi-mean(xi))) ^2.
   - We calculate (1) / (2) to find the slope of the best fitting line (let's say w1)
   - Calculate w0 using the formula mean(yi) – w1*mean(xi).

Years of Experience (xi)	salary (yi)	xi - xbar	yi - ybar	(xi - xbar)^2	(xi - xbar)*(yi - ybar)
1.1	39343	-4.2133	-36326.7	17.75189689	153055.2851
1.3	46205	-4.0133	-29464.7	16.10657689	118250.6805
1.5	37731	-3.8133	-37938.7	14.54125689	144671.6447
2	43525	-3.3133	-32144.7	10.97795689	106505.0345
2.2	39891	-3.1133	-35778.7	9.69263689	111389.8267
2.9	56642	-2.4133	-19027.7	5.82401689	45919.54841
3	60150	-2.3133	-15519.7	5.35135689	35901.72201
3.2	54445	-2.1133	-21224.7	4.46603689	44854.15851
3.2	64445	-2.1133	-11224.7	4.46603689	23721.15851
3.7	57189	-1.6133	-18480.7	2.60273689	29814.91331
3.9	63218	-1.4133	-12451.7	1.99741689	17597.98761
4	55794	-1.3133	-19875.7	1.72475689	26102.75681
4	56957	-1.3133	-18712.7	1.72475689	24575.38891
4.1	57081	-1.2133	-18588.7	1.47209689	22553.66971
4.5	61111	-0.8133	-14558.7	0.66145689	11840.59071
4.9	67938	-0.4133	-7731.7	0.17081689	3195.51161

5.1	66029	-0.2133	-9640.7	0.04549689	2056.36131
5.3	83088	-0.0133	7418.3	0.00017689	-98.66339
5.9	81363	0.5867	5693.3	0.34421689	3340.25911
6	93940	0.6867	18270.3	0.47155689	12546.21501
6.8	91738	1.4867	16068.3	2.21027689	23888.74161
7.1	98273	1.7867	22603.3	3.19229689	40385.31611
7.9	101302	2.5867	25632.3	6.69101689	66303.07041
8.2	113812	2.8867	38142.3	8.33303689	110105.3774
8.7	109431	3.3867	33761.3	11.46973689	114339.3947
9	105582	3.6867	29912.3	13.59175689	110277.6764
9.5	116969	4.1867	41299.3	17.52845689	172907.7793
9.6	112635	4.2867	36965.3	18.37579689	158459.1515
10.3	112391	4.9867	36721.3	24.86717689	183118.1067
10.5	121872	5.1867	46202.3	26.90185689	239637.4694
	MEAN (yi) =				
MEAN (xi) = 5.3133	75669.67			SUM = 223.55	SUM = 2157216.13

w1 = (2157216.13)/ (223.55) = 9236.45

w0 = 75669.67 – (9236.451 * 5.3133) = 26593.33

Final Equation => **salary = 26593.33 + 9236.45 * years of experience**.

6. Implementation

1. Python - Lasso Regression Implementation using user defined classes and functions.

   Input: salary_data.xlsx, a spreadsheet which contains total numbers of years of experience and the present salary of 30 employees in a company.

Years of Experience (xi)	salary (yi)
1.1	39343
1.3	46205
1.5	37731
2	43525
2.2	39891
2.9	56642
3	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218
4	55794
4	56957
4.1	57081
4.5	61111
4.9	67938
5.1	66029
5.3	83088
5.9	81363
6	93940
6.8	91738
7.1	98273
7.9	101302
8.2	113812
8.7	109431
9	105582
9.5	116969
9.6	112635
10.3	112391
10.5	121872

Python code:

Lasso Regression model Definition ¶

In [1]:
```
# Importing libraries
import numpy as np

# Lasso Regression
class LassoRegression():

    def __init__(self, learning_rate, iterations, l1_penalty):
        self.learning_rate = learning_rate
        self.iterations = iterations
        self.l1_penalty = l1_penalty

    # Function for model training
    def fit(self, X, Y):
        self.m, self.n = X.shape

        # weight initialization
        self.W = np.zeros(self.n)
        self.b = 0
        self.X = X
        self.Y = Y

        # gradient descent learning
        for i in range(self.iterations):
            self.update_weights()
        return self

    # Function to update weights during gradient descent
    def update_weights(self):
        Y_pred = self.predict(self.X)

        # calculating the gradients
        dW = np.zeros(self.n)
        for j in range(self.n):
            if self.W[j] > 0 :
                dW[j] = (-(2 * (self.X[:,j]).dot(self.Y - Y_pred)) + self.l1_penalty) / self.m
            else:
                dW[j] = (-(2 * (self.X[:,j]).dot(self.Y - Y_pred)) - self.l1_penalty) / self.m

        db = -2 * np.sum(self.Y - Y_pred) / self.m

        # update weights
        self.W = self.W - self.learning_rate * dW
        self.b = self.b - self.learning_rate * db
        return self

    def predict(self, X):
        return X.dot(self.W) + self.b
```

Dataset Import

In [2]:
```
import pandas as pd

df = pd.read_excel("salary_data.xlsx")
```

Dataset Cross-Validation

In [3]:
```
from sklearn.model_selection import train_test_split

X = df.iloc[:, :-1].values
Y = df.iloc[:, 1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Model training ¶

```
In [4]: model = LassoRegression(iterations = 1000, learning_rate = 0.01, l1_penalty = 500)
        _ = model.fit( X_train, Y_train )
```

Prediction on the Test Dataset

```
In [5]: Y_pred = model.predict(X_test)
        print("Predicted values ", np.round(Y_pred, 0))
        print("Real values      ", np.round(Y_test, 0))
        print("Trained W ", round( model.W[0], 2))
        print("Trained b ", round( model.b, 2))
```

```
Predicted values  [ 40733. 123884.  65300.  63410. 116325. 108766. 117270.  64355.]
Real values       [ 37731 112391  57081  63218 116969 109431 112635  55794]
Trained W  9449.01
Trained b  26559.14
```

Output: The model plots the best fitting line for the given training dataset and predicts values for the test dataset. Performance metrics such as Mean Square Error, Root Mean Square Error, Mean Absolute Error are also calculated to give an idea of the performance of the Lasso Regression Model.

Graph Plotting for the Test Dataset

```
In [6]: import matplotlib.pyplot as plt

        plt.scatter(X_test, Y_test, color = 'blue')
        plt.plot(X_test, Y_pred, color = 'orange')
        plt.title('Salary vs Experience')
        plt.xlabel('Years of Experience')
        plt.ylabel('Salary')
        plt.show()
```

Performance Measurements

```
In [7]: from sklearn.metrics import mean_squared_error as mse
        from sklearn.metrics import mean_absolute_error as mae

        print('Mean Squared Error =', round(mse(Y_test, Y_pred), 2))
        print('Root Mean Squared Error =', round(mse(Y_test, Y_pred, squared=False), 2))
        print('Mean Absolute Error =', round(mae(Y_test, Y_pred), 2))
```

```
Mean Squared Error = 38039873.53
Root Mean Squared Error = 6167.65
Mean Absolute Error = 4676.44
```

2. Python - Lasso Regression Implementation with scikit-libraries

Input: salary_data.xlsx, a spreadsheet which contains total numbers of years of experience and the present salary of 30 employees in a company.

Years of Experience (xi)	salary (yi)
1.1	39343
1.3	46205
1.5	37731
2	43525
2.2	39891
2.9	56642
3	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218
4	55794
4	56957
4.1	57081
4.5	61111
4.9	67938
5.1	66029
5.3	83088
5.9	81363
6	93940
6.8	91738
7.1	98273
7.9	101302
8.2	113812
8.7	109431
9	105582
9.5	116969
9.6	112635
10.3	112391
10.5	121872

Python code:

Dataset Import

In [1]:
```
import pandas as pd

data = pd.read_excel('salary_data.xlsx')
data.describe()
```

Out[1]:

	years of experience	salary
count	30.000000	30.000000
mean	5.313333	75669.666667
std	2.837888	26886.661556
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	121872.000000

Dataset Cross-Validation

In [2]:
```
from sklearn.model_selection import train_test_split

X = data[['years of experience']]
Y = data[['salary']]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Model Training

In [3]:
```
from sklearn.linear_model import Lasso

model = Lasso()
_ = model.fit(X_train, Y_train)
```

Prediction on the Test Dataset

In [4]:
```
import numpy as np

Y_pred = model.predict(X_test)
print("Predicted values ", np.round(Y_pred, 0))
print("Actual values    ", np.round(Y_test['salary'].to_list(),0))
```

```
Predicted values  [ 41057. 123597.  65444.  63568. 116093. 108590. 117031.  64506.]
Actual values     [ 37731 112391  57081  63218 116969 109431 112635  55794]
```

Output: The model plots the best fitting line for the given training dataset and predicts values for the test dataset. Performance metrics such as Mean Square Error, Root Mean Square Error, Mean Absolute Error are also calculated to give an idea of the performance of the Lasso Regression Model.

Graph Plotting for Test Dataset

In [5]:
```
import matplotlib.pyplot as plt
plt.scatter(X_test, Y_test, color = 'blue')
plt.plot(X_test, Y_pred, color = 'orange')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Performance Measurements

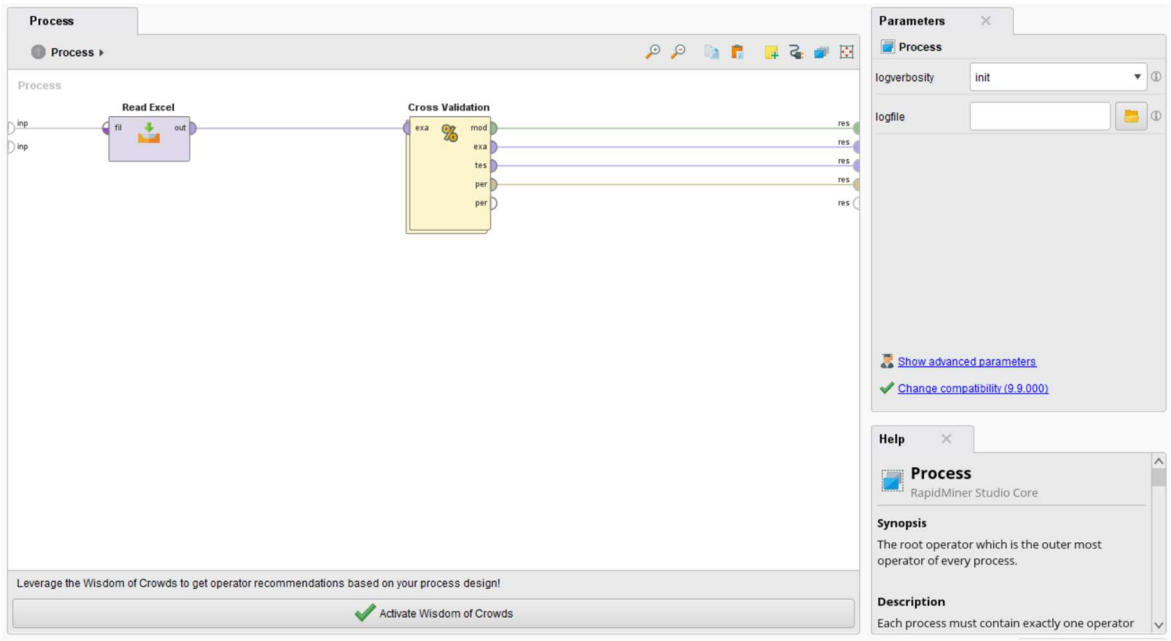
In [6]:
```
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae

print('Mean Squared Error =', round(mse(Y_test, Y_pred), 2))
print('Root Mean Squared Error =', round(mse(Y_test, Y_pred, squared=False), 2))
print('Mean Absolute Error =', round(mae(Y_test, Y_pred), 2))
```

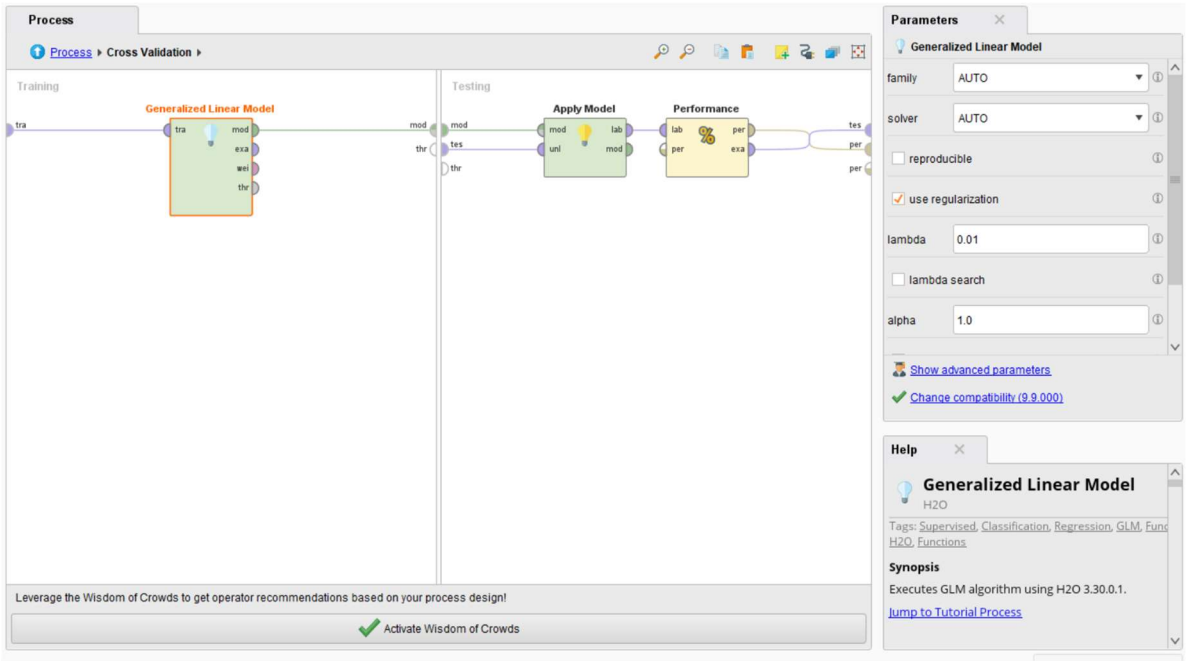
```
Mean Squared Error = 37922807.34
Root Mean Squared Error = 6158.15
Mean Absolute Error = 4758.63
```

3. RapidMiner Implementation (Screenshots)

Overall Model:



The Lasso Regression model after Cross-Validation of Data:



Stats from the Generalized Linear Model (Lasso Regression):

Result History | ExampleSet (Read Excel) | Generalized Linear Model (Generalized Linear Model)

Data

Description

Attribute	Coefficient	Std. Coefficient	Std. Error	z-Value	p-Value
years of experience	9236.447	26212.003	?	?	?
Intercept	26593.345	75669.667	?	?	?

Performance Measurements from the Lasso Regression Model:

ExampleSet (Read Excel) | Generalized Linear Model (Generalized Linear Model)

Result History | PerformanceVector (Performance) | ExampleSet (Cross Validation)

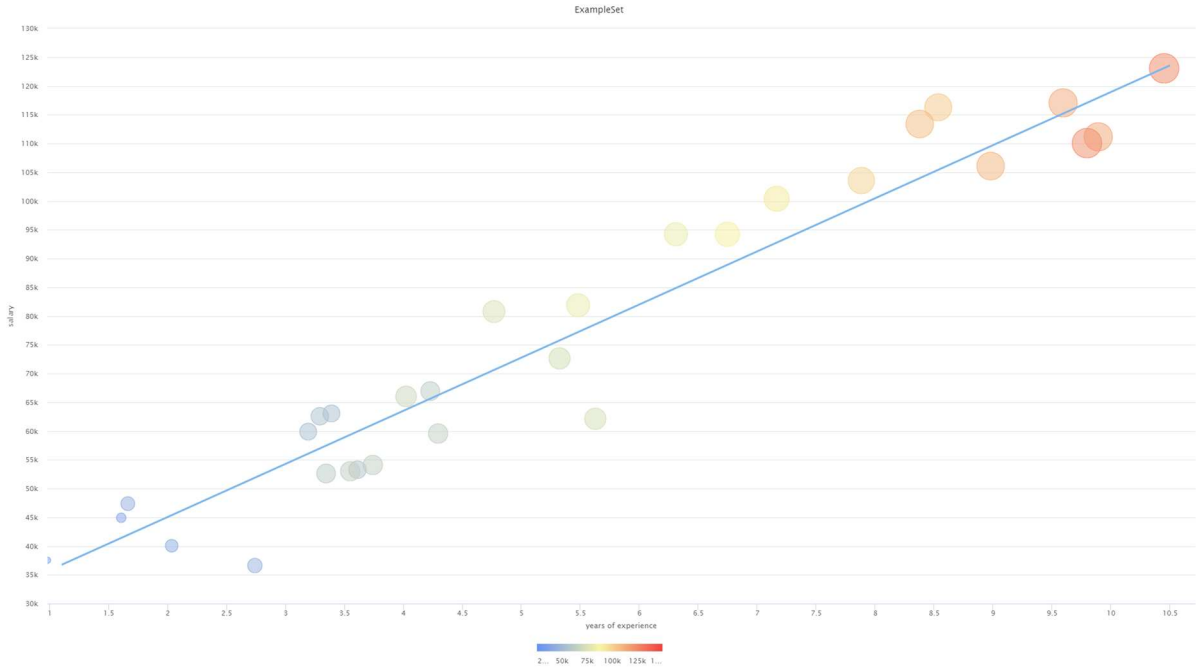
Performance

Description

PerformanceVector

```
PerformanceVector:
root_mean_squared_error: 6066.596 +/- 1201.760 (micro average: 6161.085 +/- 0.000)
absolute_error: 5301.777 +/- 1162.383 (micro average: 5301.777 +/- 3138.493)
relative_error: 7.87% +/- 1.95% (micro average: 7.87% +/- 4.96%)
squared_error: 37958973.202 +/- 15402963.430 (micro average: 37958973.202 +/- 38686771.475)
```

Graph Plotting for the Generalized Linear Model (Lasso Regression):



7. References

- Beyer, W. H. CRC Standard Mathematical Tables, 31st ed. Boca Raton, FL: CRC Press, 2002.
- Agresti A. (1990) Categorical Data Analysis. John Wiley and Sons, New York.
- Katz, S.; et al., eds. (2006), Encyclopedia of Statistical Sciences, Wiley.
- Wheelan, C. (2014). Naked Statistics. W. W. Norton & Company.