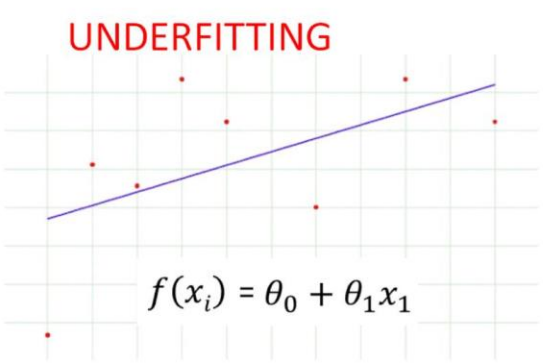
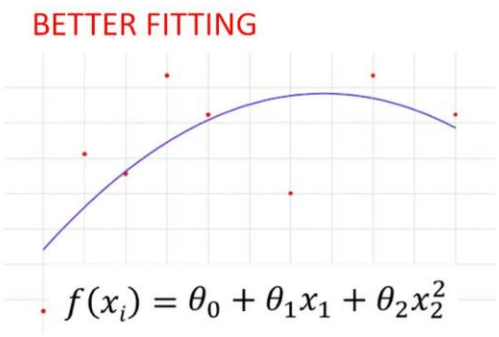
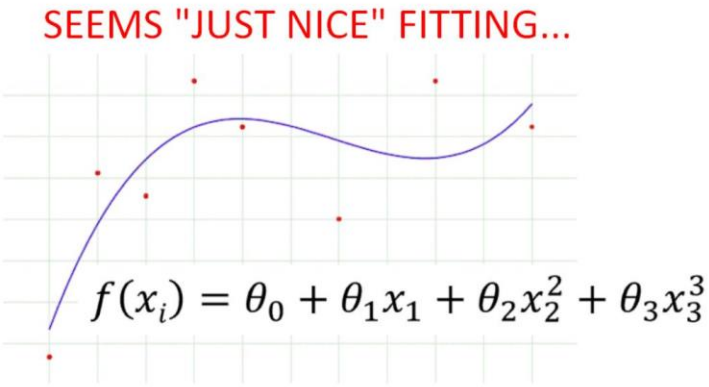
L1 LASSO AND L2 RIDGE AND ELASTIC NET REGRESSION

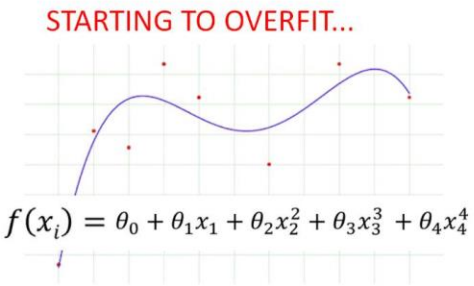
USING PYTHON
DR. ALVIN ANG

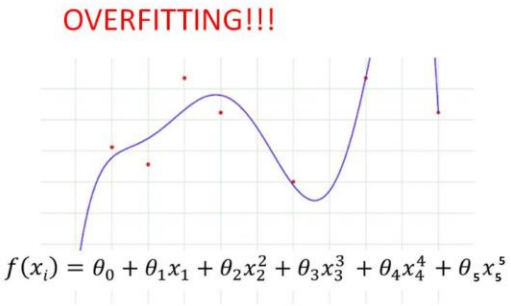
CONTENTS

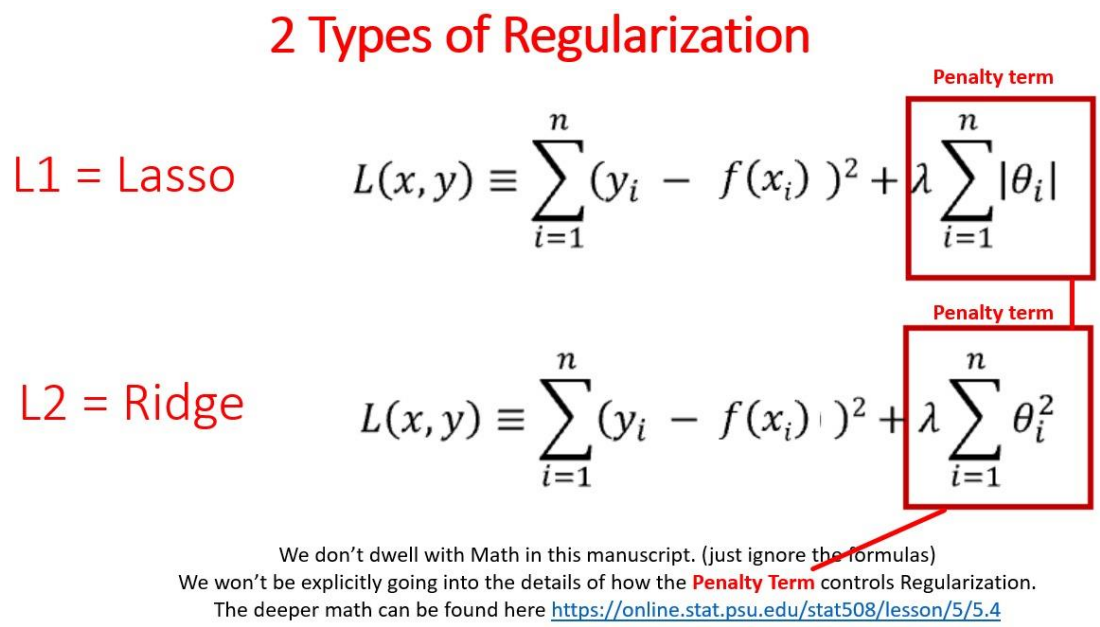












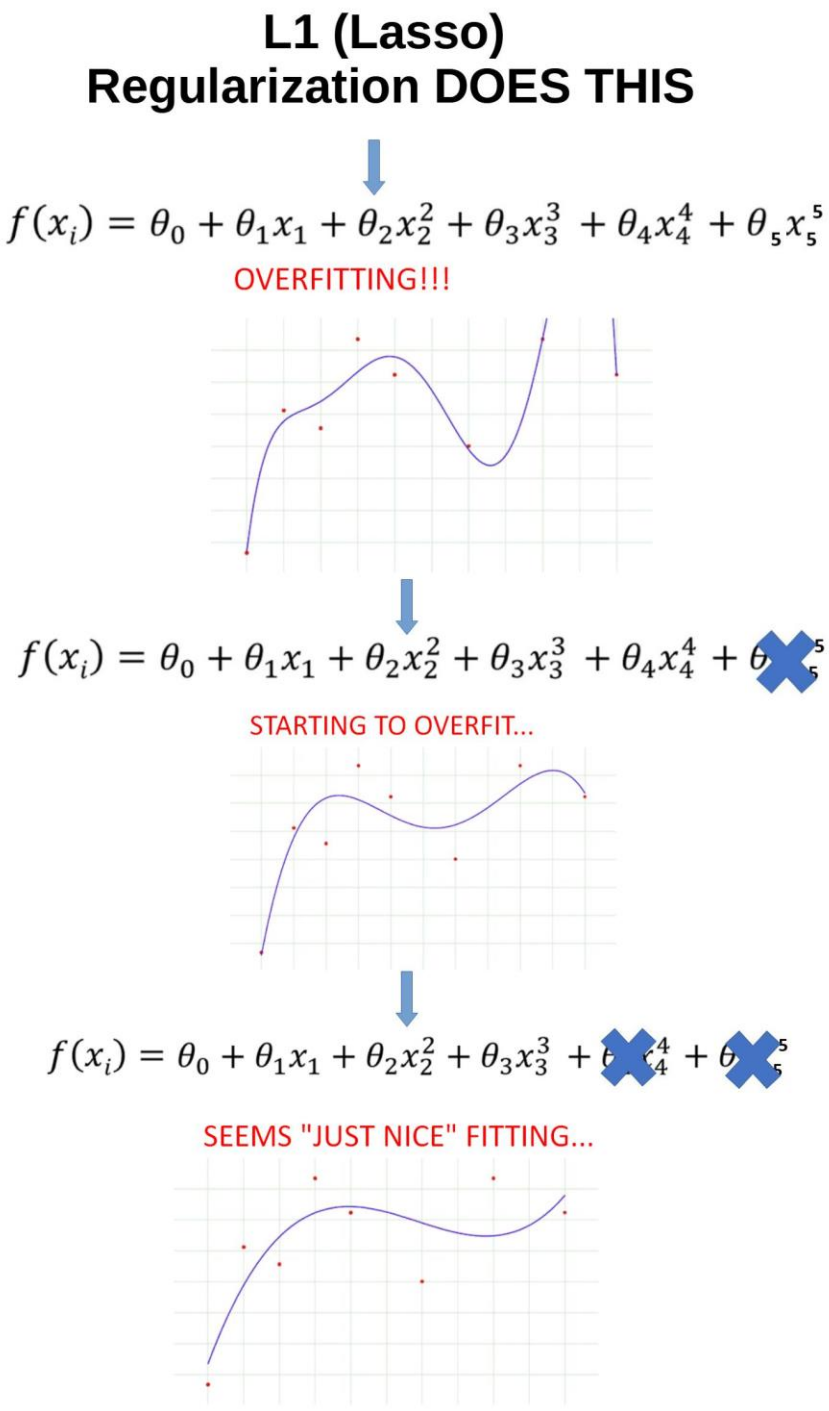
We don't dwell with Math in this manuscript. (just ignore the formulas)

We won't be explicitly going into the details of how the Penalty Term controls Regularization.

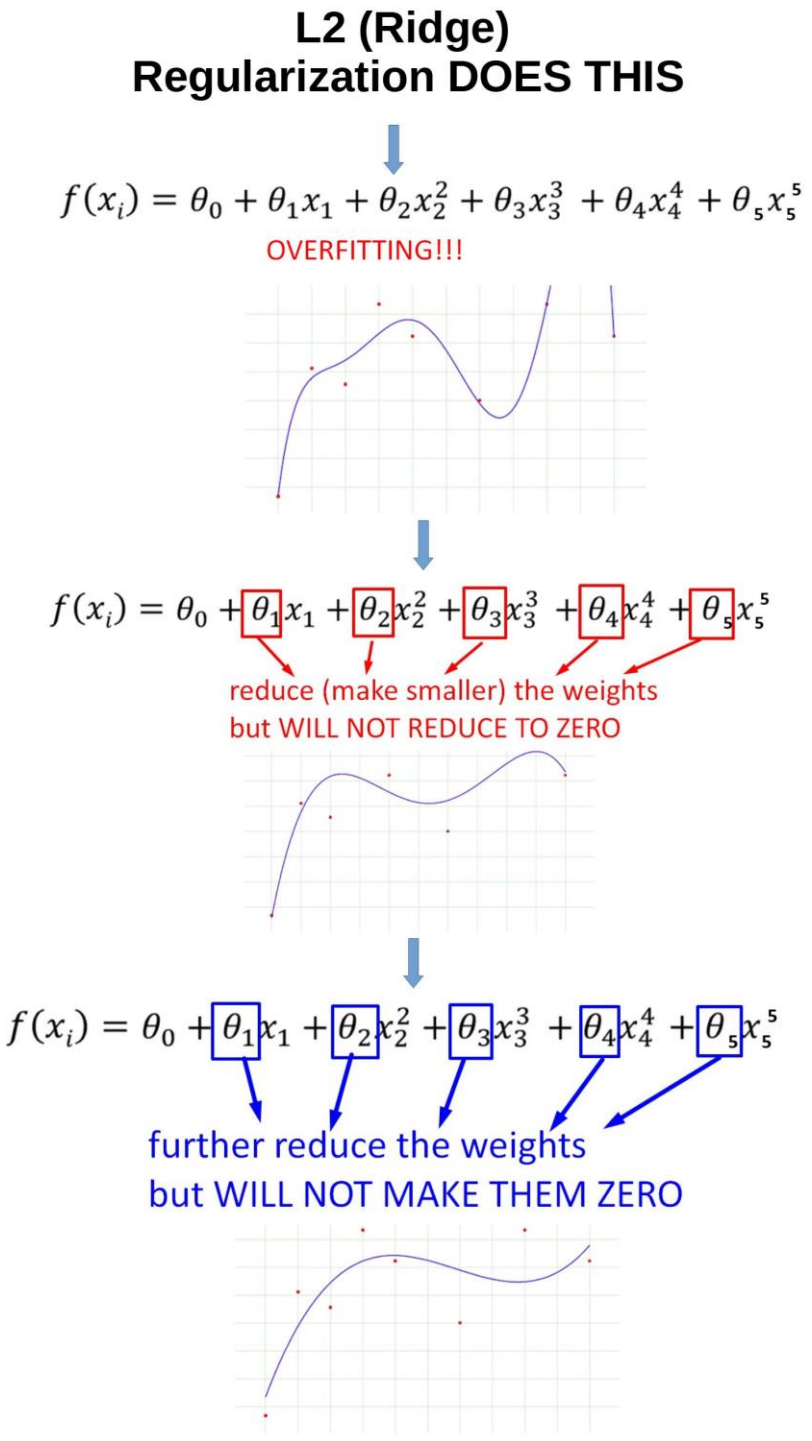
The deeper math can be found here https://online.stat.psu.edu/stat508/lesson/5/5.4

ELASTIC NET REGRESSION = L1 + L2 MIXTURE

A. L1 (LASSO) REGULARIZATION



B. L2 (RIDGE) REGULARIZATION



III. LASSO VS RIDGE VS ELASTIC NET REGRESSION USING PYTHON

https://www.alvinang.sg/s/L1_Lasso_and_L2_Ridge_and_Elastic_Net_Regression_using_Python_by_Dr_Alvin_Ang.ipynb

https://www.alvinang.sg/s/boston_housing_data.csv

A. STEP 1: IMPORT DATASET

L1 Lasso and L2 Ridge Regression using Python by Dr Alvin Ang

https://towardsdatascience.com/lasso-and-ridge-regression-an-intuitive-comparison-3ee415487d18

1. IMPORT LIBRARIES

Step 1: Import Dataset

1a) Import Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error
```

2. IMPORT DATASET

```
1b) Import Dataset

df = pd.read_csv('https://www.alvinang.sg/s/boston_housing_data.csv').head(100)

#.head(100) only keeps the first 100 rows
#and not FULL dataset

print(df)
```

```
[72]
0s
          CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0   0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1   0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2   0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3   0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4   0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0
..      ...   ...    ...   ...    ...    ...   ...     ...  ...    ...
95  0.12204   0.0   2.89     0  0.445  6.625  57.8  3.4952    2  276.0
96  0.11504   0.0   2.89     0  0.445  6.163  69.6  3.4952    2  276.0
97  0.12083   0.0   2.89     0  0.445  8.069  76.0  3.4952    2  276.0
98  0.08187   0.0   2.89     0  0.445  7.820  36.9  3.4952    2  276.0
99  0.06860   0.0   2.89     0  0.445  7.416  62.5  3.4952    2  276.0

    PTRATIO       B  LSTAT  MEDV
0      15.3  396.90   4.98  24.0
1      17.8  396.90   9.14  21.6
2      17.8  392.83   4.03  34.7
3      18.7  394.63   2.94  33.4
4      18.7  396.90   5.33  36.2
..      ...     ...    ...   ...
95     18.0  357.98   6.65  28.4
96     18.0  391.83  11.34  21.4
97     18.0  396.90   4.21  38.7
98     18.0  393.53   3.57  43.8
99     18.0  396.90   6.19  33.2

[100 rows x 14 columns]
```

B. STEP 2: TRAIN TEST SPLIT

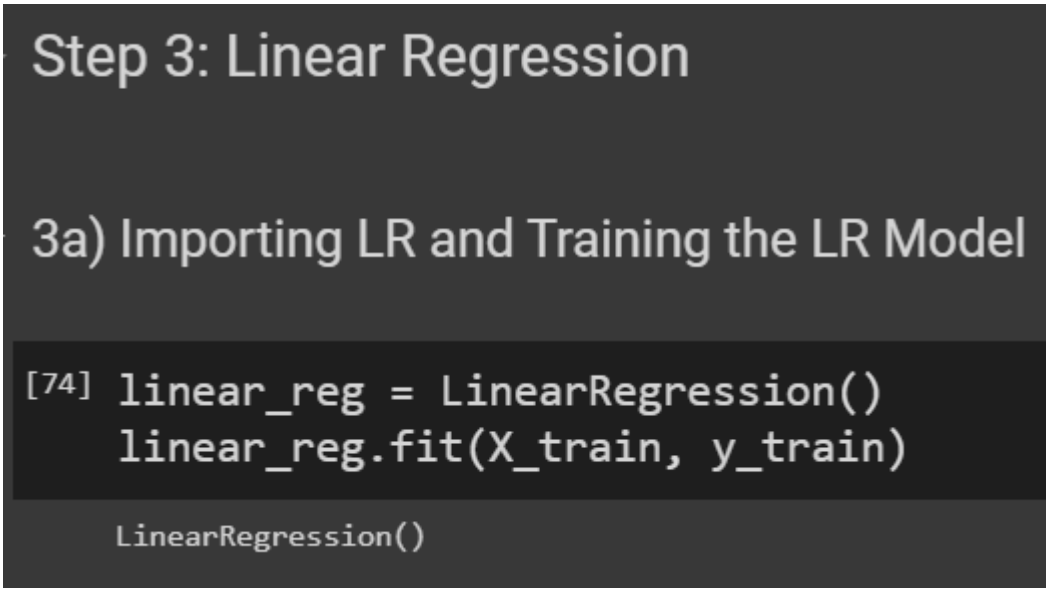
Step 2: Train Test Split

```
target= ["MEDV"]
features = ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE","DIS","RAD","TAX", "PTRATIO","B","LSTAT"]

y = df[target]
X = df[features]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=5)
print(X_train.shape)
print(y_train.shape)

print(X_test.shape)
print(y_test.shape)

#80% for training - 20% for testing
```

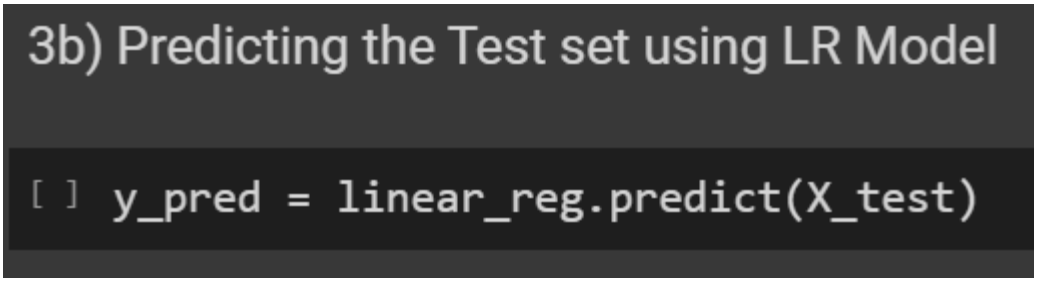
```
(80, 13)
(80, 1)
(20, 13)
(20, 1)
```

C. STEP 3: LINEAR REGRESSION

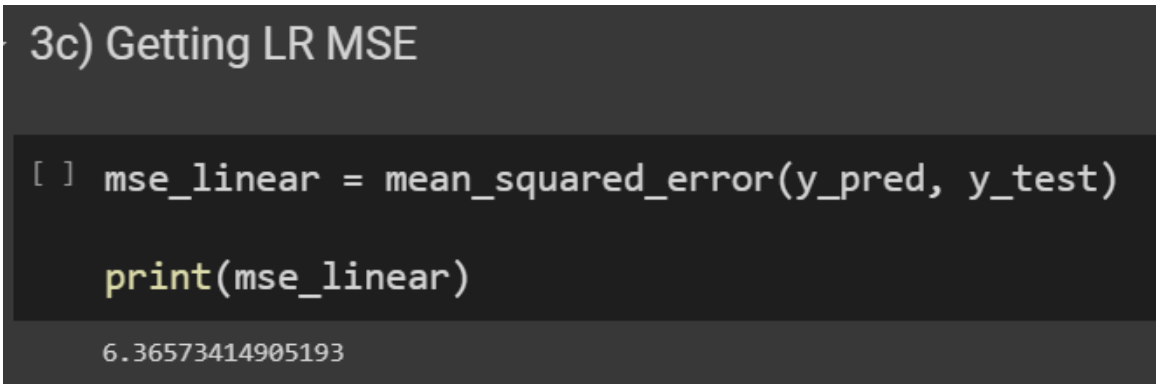
1. IMPORTING LINEAR REGRESSION AND TRAINING THE LINEAR REGRESSION MODEL



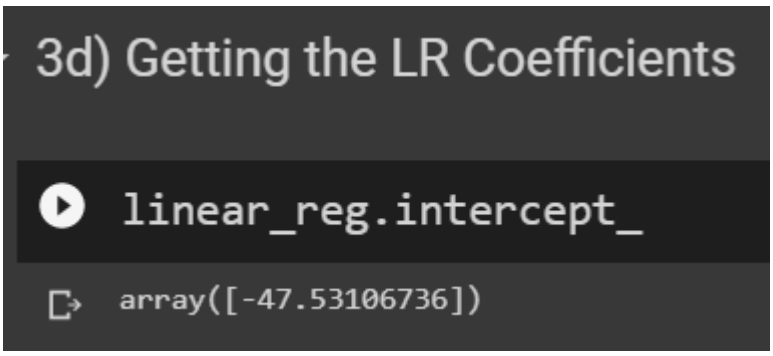
2. PREDICTING THE TEST SET USING LINEAR REGRESSION MODEL

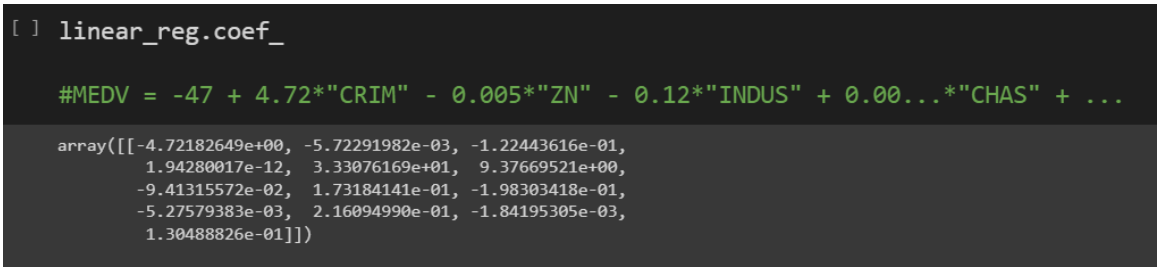


3. GETTING LINEAR REGRESSION MSE



4. GETTING THE LINEAR REGRESSION COEFFICIENTS





D. STEP 4: L1 – LASSO REGRESSION

1. GETTING LASSO MSE

Step 4: L1 - Lasso Regression

4a) Getting Lasso MSE

```
[ ] lambda_values = [0.000001, 0.0001, 0.001, 0.005, 0.01, 0.05,  0.1, 0.2, 0.3, 0.4, 0.5]

    for lambda_val in lambda_values:
        lasso_reg = Lasso(lambda_val)
        lasso_reg.fit(X_train, y_train)
        y_pred = lasso_reg.predict(X_test)
        mse_lasso = mean_squared_error(y_pred, y_test)
        print(("Lasso MSE with Lambda={} is {}").format(lambda_val, mse_lasso))
```

```
Lasso MSE with Lambda=1e-06 is 6.365573187964432
Lasso MSE with Lambda=0.0001 is 6.3497037030409
Lasso MSE with Lambda=0.001 is 6.211388388151761
Lasso MSE with Lambda=0.005 is 5.726363213840866
Lasso MSE with Lambda=0.01 is 5.411548113524234
Lasso MSE with Lambda=0.05 is 5.263737896443069
Lasso MSE with Lambda=0.1 is 4.903847171765334
Lasso MSE with Lambda=0.2 is 4.363168988410968
Lasso MSE with Lambda=0.3 is 4.2516592908158914
Lasso MSE with Lambda=0.4 is 4.569299505022656
Lasso MSE with Lambda=0.5 is 5.316188931738344
```

2. GETTING LASSO COEFFICIENTS

4b) Getting Lasso Coefficients

```
[ ] print(lasso_reg.coef_)
    #NOTICE THAT SOME COEFFICIENTS ARE ZERO!
    #Meaning, some FEATURES have been ZEROED OUT!
```

```
[-0.         -0.00465237 -0.20754173  0.         -0.          4.1765001
 -0.04301186 -0.         -0.         -0.01274246 -0.2694468   0.01186561
 -0.27931101]
```

E. STEP 5: L2 – RIDGE REGRESSION

1. GETTING RIDGE MSE

Step 5: L2 - Ridge Regression

5a) Getting Ridge MSE

```
lambda_values = [0.00001, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 3, 5, 6, 7, 8, 9, 10]

for lambda_val in lambda_values:
    ridge_reg = Ridge(lambda_val)
    ridge_reg.fit(X_train, y_train)
    y_pred = ridge_reg.predict(X_test)
    mse_ridge = mean_squared_error(y_pred, y_test)
    print(("Ridge MSE with Lambda={} is {}").format(lambda_val, mse_ridge))
```

```
Ridge MSE with Lambda=1e-05 is 6.36516470349015
Ridge MSE with Lambda=0.01 is 5.974605793676387
Ridge MSE with Lambda=0.05 is 5.533200380976024
Ridge MSE with Lambda=0.1 is 5.3925069171769575
Ridge MSE with Lambda=0.5 is 4.958888346329863
Ridge MSE with Lambda=1 is 4.6099591818212575
Ridge MSE with Lambda=1.5 is 4.381430640413449
Ridge MSE with Lambda=3 is 4.144021177550216
Ridge MSE with Lambda=5 is 4.337309782014395
Ridge MSE with Lambda=6 is 4.523375817049083
Ridge MSE with Lambda=7 is 4.733551913450155
Ridge MSE with Lambda=8 is 4.954202530190625
Ridge MSE with Lambda=9 is 5.177084341160097
Ridge MSE with Lambda=10 is 5.397233430913719
```

2. GETTING RIDGE COEFFICIENTS

5b) Getting Ridge Coefficients

```
[ ] print(ridge_reg.coef_)

    #NOTICE THAT ALL FEATURES (except one) ARE PRESENT
    #AND ALL WEIGHTS ARE GIVEN ALMOST SIMILAR IMPORTANCE!
    #that is, ALL features are accounted for!
```

```
[[-5.08790911e-01 -1.07642424e-02 -2.51669160e-01  0.00000000e+00
   3.59195624e-03  3.76304412e+00 -3.71563298e-02  2.77184371e-02
  -1.53559311e-01 -8.99656296e-03 -4.24245154e-01  5.34856750e-03
  -3.22059452e-01]]
```

F. STEP 6: ELASTIC NET REGRESSION

1. GETTING ELASTIC NET REGRESSION

6a) Getting EN Regression

```
from sklearn.linear_model import ElasticNet

lambda_values = [0.00001, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 3, 5, 6, 7, 8, 9, 10]

for lambda_val in lambda_values:
    EN_reg = ElasticNet(lambda_val, l1_ratio = 0.5)
    EN_reg.fit(X_train, y_train)
    y_pred = EN_reg.predict(X_test)
    mse_EN = mean_squared_error(y_pred, y_test)
    print(("ElasticNet MSE with Lambda={} is {}").format(lambda_val, mse_EN))

    #l1_ratio is between 0 to 1
    #Meaning : 0 <= l1_ratio <= 1
    #If l1_ratio = 0 means NO L1(Lasso), CONTAINS ONLY L2(Ridge)
    #If l1_ratio = 1 means NO L2, CONTAINS ONLY L1
    #If l1_ratio = 0.5 means HALF L1 and HALF L2
```

```
ElasticNet MSE with Lambda=1e-05 is 6.3425546923512295
ElasticNet MSE with Lambda=0.01 is 5.010655145516045
ElasticNet MSE with Lambda=0.05 is 4.241780686298112
ElasticNet MSE with Lambda=0.1 is 4.310069062664355
ElasticNet MSE with Lambda=0.5 is 8.536143785192817
ElasticNet MSE with Lambda=1 is 11.515985845758951
ElasticNet MSE with Lambda=1.5 is 13.373484125442392
ElasticNet MSE with Lambda=3 is 15.740399667184102
ElasticNet MSE with Lambda=5 is 16.771682464187837
ElasticNet MSE with Lambda=6 is 17.34053278224848
ElasticNet MSE with Lambda=7 is 17.937997574609007
ElasticNet MSE with Lambda=8 is 18.48755285934491
ElasticNet MSE with Lambda=9 is 19.057531219726023
ElasticNet MSE with Lambda=10 is 19.645401193497044
```

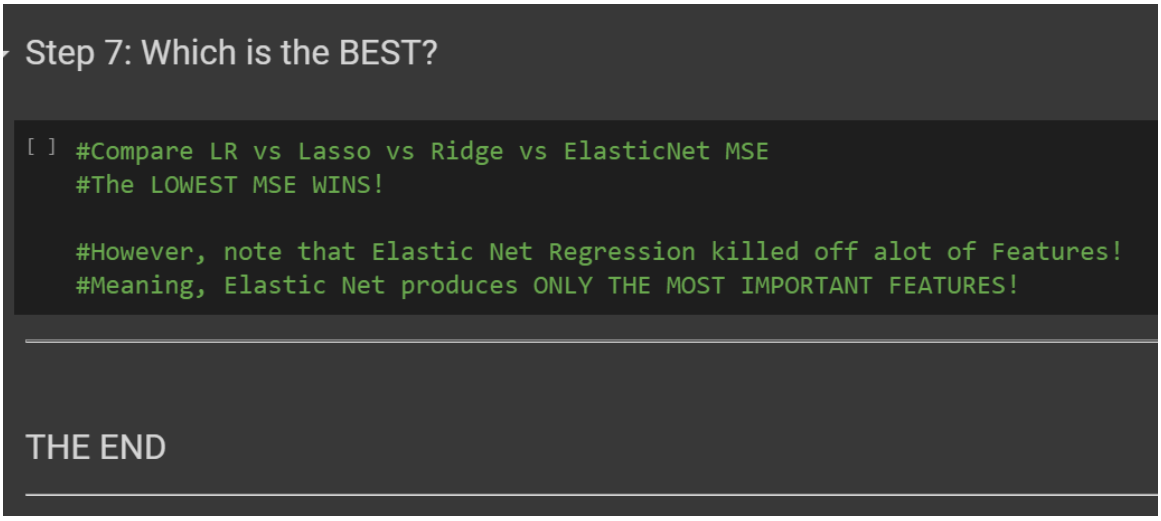
2. GETTING ELASTIC NET COEFFICIENTS

6b) Getting EN Coefficients

```
[ ] print(EN_reg.coef_)
```

```
[-0.          0.00939145 -0.          0.         -0.          0.
 -0.04533207 -0.         -0.         -0.03157748 -0.          0.03253858
 -0.27448801]
```

G. STEP 7: WHICH ISBEST?



IV. CONCLUSION

A. L1 LASSO

- CAN ZERO OUT USELESS FEATURES and keep only the useful ones…
- In other words, Lasso can be used for Feature Selection!
- Meaning, you can use Lasso if you wish to remove USELESS FEATURES

B. L2 RIDGE

- WILL NOT ZERO OUT FEATURES!
- Meaning, Ridge will keep most features (even though it will lessen its weight / importance).
- Use this if you WANT to KEEP ALL features.

C. WHICH ONE TO USE?

- Hard to determine.
- Need to run Python code and test using different Lambda/Alpha values to check.
- Lowest MSE wins!

REFERENCES

https://towardsdatascience.com/lasso-and-ridge-regression-an-intuitive-comparison-3ee415487d18

https://medium.datadriveninvestor.com/l1-l2-regularization-7f1b4fe948f2

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.