Deep Narayan Mishra – 50245878

Programming Assignment 1 Report

Problem Statement:

The project aims to compare the MLE, ridge and LASSO for slump flow. For the given data we are supposed to divide 103 observations in trains (85) and test data (18) randomly. Perform CV on 85 data, identify the best model and then report the average performance on 18 test data.

Environment/Language/Library Used:

I performed my activity on Windows 10. I started with installing python 3.6.3 which was one of the latest version of python release.

I didn't have much background in python. So I practiced the basic syntax and language core function functionality like its data types, data structure (list, dictionary, set), string manipulation, function, anonymous function, control statements, loops, exception handling, and libraries import and uses. It helped me understand the language and reduce my struggle in implementing the logic for this project.

To practice python or implement my code, I started with using Juypter Notebook. I found it a good tool to test ideas, and practice small code snippets. It is quick and has the ability to rerun a piece of individual code snippets.

Later to do my actual code implementation I used PyCharm IDE and sometime Spyder. I found both are similar, however I was getting a nice plot in PyCharm. I developed and tested my code on Sypder and sometimes tested them on PyCharm. These IDEs helped me debug my code. Apart from that they provide a huge list of features.

I installed and used following libraries and there packages to perform my task:-

- **Pandas**: Pandas package provides a fast and easy way to work with relational or labeled data. It helps to handle missing data, insertion/deletion of data, data alignment, grouping, slicing merging etc. I used "DataFrame" data structure of Pandas to lode read the slum flow observation file. I loaded the file into DataFrame removed the sequence no attribute and extracted target (Flow) and explanatory/feature (7 fields in our case) fields. Which is used further to do Cross Validation and testing.
- **NumPy**: It's the basic package for Python. Pandas is built on top of this package. It is useful to perform linear algebra functions. It has a powerful set of functions such as trigonometric, exponential, logarithmic, and arithmetic function. I used this package in my code to calculate the mean of MSE.
- **Scikit-learn (sklearn)**: This is the most important library for the assigned task. scikit-learn is the simple and efficient tool for data analysis in python. It is built on top of NumPy, Spicy and MatPlotlib. It provides various classification and regression algorithm. It provides huge set of algorithms. It features the algorithm for supervised learning. I used Linear Regression, Ridge

Regression, and Lasso Regression algorithm of linear model from this package. It has support for many other supervised learning.
Module-selection of scikit-learn helped to split the dataset randomly into train and test data. KFold module-selection helped me perform cross validation. I used KFold to split the train data randomly in 5 folds. Train the module on 4 folds and validated (check the score, and mean square error) on one fold.
Apart from these modules, I used mean_square_error and r2_score (coefficient of determination) regression functions to validate the models and identify which model is best suited for the given dataset.

- **Matplotlib**: It is a ploting library for python. It is most used python package for 2D and 3D graphics. It provides a very quick way to visualize the data for data analysis. Matplotlib comes with set of default settings that allow a quick plot of graph. These settings can be customized based on our need. We can control almost everything from figure size to axis properties in the plot. I have used scatter() plot function of Matplotlib to visualize the performance of module on the test data and also to compare the different modules performance visually.

Summary of MSE Comparison:

I performed following steps for MSE Comparison:

1. First I loaded the data into regression environment. I used pandas DataFrame to load all the 103 observation dataset from file. I extracted X ('Cement' 'Slag' 'Fly ash' 'Water' 'SP' 'Coarse Aggr.' 'Fine Aggr.') and y(Flow) values from the DataFrame.
2. Extracted data is passed to train_test_split() function of module_selection, with train_size 85 and test_size 18. The function returned me X_train, X_test, y_train, and y_test.
3. I used X_train, y_train for Cross validation. I constructed KFold module selection with 5 splits, and created 5 folds using KFold split function. It returned me train_index, test_index for five times.
4. Inside every CV loop, I extracted X_train1, X_cv, y_train1, y_cv using KFold split. The X_train1 and y_train1 is the 4 fold (68 observations) of 85 observations used for training the model. Whereas X_cv and y_cv is the validation fold (17 observations) used for validating the trained model.
5. The class of model is trained on X_train1 and y_train1. The trained model is evaluated for Mean Squared Error (MSE) and the model with the least mean squared error is stored as a best model.
6. For example – I am calculating MSE in each loop of CV for the linear regression model. If the MSE is found better (least) than previously found MSE than I am storing that model in a variable called lr_model_best.
7. Once the CV loop is over, I will have the best model in variable lr_model_best. I use this model to predict my test data (X_test). The predicted value is used to calculate the R2 and MSE.
8. The step 3 to 7 is repeated for 10 times, storing the MSE calculated on test data.
9. Finally, I display the MSE, MSE Average and R2_score calculated on each iteration.

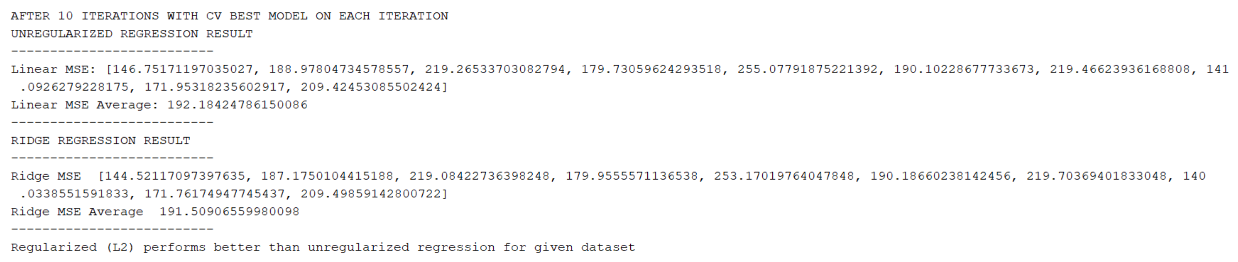
The process is repeated for Task1.1 Task1.2 and Task1.3.

- **Task1.1**: - I calculate the R2_Score of unregularized regression.
- **Task1.2**: I use unregularized (linear) and L2 regularization (ridge) inside the CV loop. I identify the best for both the individual class, and store it to calculated MSE on Test data.

  The coefficient (alpha) for L2 regularization is 20. I tried with so many random values and found the model doesn't produce much difference than linear regression on alpha 1, 2, 3, etc.. The alpha value 20 was giving some significant less mean square error than unregularized regression.

  I found the Ridge (L2 Regularization) regression performs better than unregularized almost all cases. For example, one of the output obtained from PA1_Task1.2_deepnara.py was

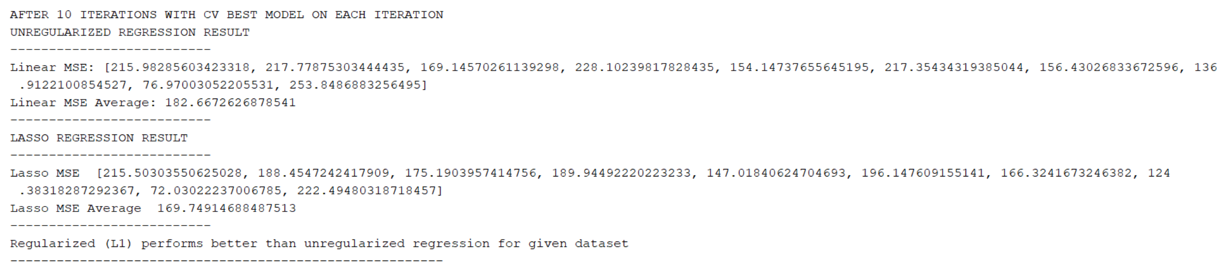
  In one of the output, below MSE was calculated on each iteration after finding best model



```
AFTER 10 ITERATIONS WITH CV BEST MODEL ON EACH ITERATION
UNREGULARIZED REGRESSION RESULT
--------------------------
Linear MSE: [146.75171197035027, 188.97804734578557, 219.26533703082794, 179.73059624293518, 255.07791875221392, 190.10228677733673, 219.46623936168808, 141
 .0926279228175, 171.95318235602917, 209.42453085502424]
Linear MSE Average: 192.18424786150086
--------------------------
RIDGE REGRESSION RESULT
--------------------------
Ridge MSE  [144.52117097397635, 187.1750104415188, 219.08422736398248, 179.9555571136538, 253.17019764047848, 190.18660238142456, 219.70369401833048, 140
 .0338551591833, 171.76174947745437, 209.49859142800722]
Ridge MSE Average  191.50906559980098
--------------------------
Regularized (L2) performs better than unregularized regression for given dataset
```

- **Task1.3**: I use unregularized (linear) and L1 regularization (Lasso) inside the CV loop. I identify the best for both the individual class, and store it to calculated MSE on Test data.

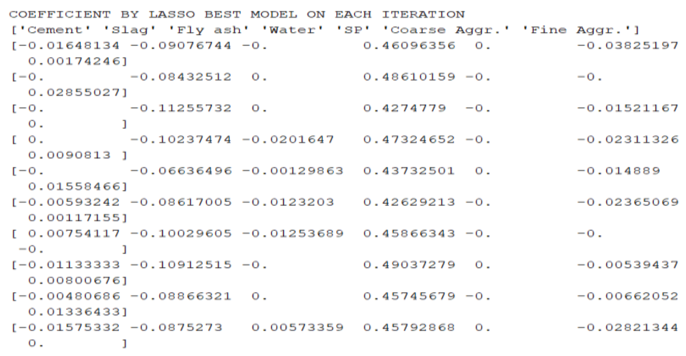
  Simdilar to L2 regularization, L1 Lasso regularization also performed better around alpha = 20. And this model performs much significantly better than the unregularized one. In one of the ouput, the below MSE was calculated on each iteration.



```
AFTER 10 ITERATIONS WITH CV BEST MODEL ON EACH ITERATION
UNREGULARIZED REGRESSION RESULT
--------------------------
Linear MSE: [215.98285603423318, 217.77875303444435, 169.14570261139298, 228.10239817828435, 154.14737655645195, 217.35434319385044, 156.43026833672596, 136
 .9122100854527, 76.97003052205531, 253.8486883256495]
Linear MSE Average: 182.6672626878541
--------------------------
LASSO REGRESSION RESULT
--------------------------
Lasso MSE  [215.50303550625028, 188.4547242417909, 175.1903957414756, 189.94492220223233, 147.01840624704693, 196.147609155141, 166.3241673246382, 124
 .38318287292367, 72.03022237006785, 222.49480318718457]
Lasso MSE Average  169.74914688487513
--------------------------
Regularized (L1) performs better than unregularized regression for given dataset
--------------------------------------------------------
```

  I had also stored the coefficient used by each iteration best model, and it is seen most of the times that 'Fly Ass', and 'SP' was not used very often. 'Coarse Aggr' wsa also not used many times. However, 'Cement', 'Slag', 'Water' and 'Coarse Aggr' were always used.

  The below output show the co-efficient of all the iterations from best CV model of Lasso.

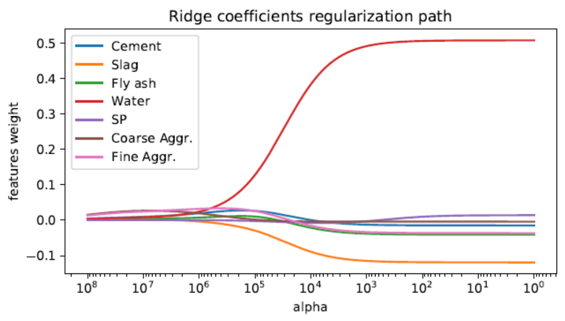


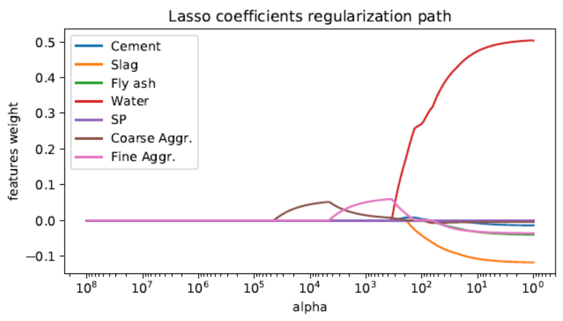
```
COEFFICIENT BY LASSO BEST MODEL ON EACH ITERATION
['Cement' 'Slag' 'Fly ash' 'Water' 'SP' 'Coarse Aggr.' 'Fine Aggr.']
[-0.01648134 -0.09076744 -0.          0.46096356  0.          -0.03825197
  0.00174246]
[-0.         -0.08432512  0.          0.48610159 -0.          -0.
  0.02855027]
[-0.         -0.11255732  0.          0.4274779  -0.          -0.01521167
  0.        ]
[ 0.         -0.10237474 -0.0201647   0.47324652 -0.          -0.02311326
  0.0090813 ]
[-0.         -0.06636496 -0.00129863  0.43732501  0.          -0.014889
  0.01558466]
[-0.00593242 -0.08617005 -0.0123203   0.42629213 -0.          -0.02365069
  0.00117155]
[ 0.00754117 -0.10029605 -0.01253689  0.45866343 -0.          -0.
 -0.        ]
[-0.01133333 -0.10912515 -0.          0.49037279  0.          -0.00539437
  0.00800676]
[-0.00480686 -0.08866321  0.          0.45745679 -0.          -0.00662052
  0.01336433]
[-0.01575332 -0.0875273   0.00573359  0.45792868  0.          -0.02821344
  0.        ]
```

Regularization Path Graph of Ridge & LASSO:

I performed this task in two phases.

1. Split the data into two parts (train – 85 and test 18). I ran CV on the 85 observation with 5 folds. Each time I ran through the fold I preserved the Fold indices for which the model (ridge and lasso) generated the least MSE. I have used this best fold index to extract the data from my original set.
2. In next phase I used the best fold data to compute the coefficient of Lasso and Ridge on range of alpha value (logarithmic range). I used logarithmic range so data I could get the regularization path flow in the graph. With small range of alphas value the flow doesn't change.
3. The computed coefficient for Lasso and Ridge is used to plot the graph.
4. In the graph, x – axis represents the alpha value. Whereas y – axis represents the weight of each feature (explanatory) variable. I labeled each explanatory variable so that we can identify which one converges.





Deep Narayan Mishra – 50245878

Time Spent and resources used:

I started working on the Assignment since 16th of March, after the mid-term. I have spent 3 to 4 hours daily on this since 16th till 23rd march. And from 23rd till 26th march 6 to 7 hours and little more.

I have followed piazza very closely. I cleared most of the doubts. After from this, I browsed through the approach for using SKlearn on internet. I have used below sites as a reference to perform my task.

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

http://nbviewer.jupyter.org/github/ubdsgroup/ubmlcourse/blob/master/notebooks/LinearSystems.ipynb

http://scikit-learn.org/stable/auto_examples/linear_model/plot_ridge_path.html