CPSC 483, Project 1

Team Name: ‘AKMMR’

Team Members:

|  |  |  |  |
| --- | --- | --- | --- |
| Serial Number | Name | CWID | Email ID |
| 1 | Asit Kumar Dash | 893423350 | asitdash@csu.fullerton.edu |
| 2 | Kashyap Sodha | 893260984 | ksodha@csu.fullerton.edu |
| 3 | Melika Amini | 893423350 | melika\_am@csu.fullerton.edu |
| 4 | Milan Ghori | 893585703 | ghorimilan1994@csu.fullerton.edu |
| 5 | Rishabh Sharma | 893463695 | rishabh230795@csu.fullerton.edu |

Under guidance of Prof. Christopher Ryu

Machine Learning – CPSC483

Department of Computer Science

California State University, Fullerton

Contribution Percentage:

The project was done as a group activity. Everyone participated in testing different models for linear and non-linear models. We identified different types of regression

models and applied mean squared error & mean absolute error concepts. Below is the table for contribution percentage.

|  |  |
| --- | --- |
| Name | Contribution |
| Asit Kumar Dash | Linear models: tested Ridge regression, Lasso regression and lars-lasso regression  Non-Linear models: tested non-linear model with degree – 2,3,4,5,6. |
| Kashyap Sodha | Linear model: designed linear model and tested Bayesian regression. |
| Melika Amini | Non-Linear models: tested the Non-Linear model with the absolute error. |
| Milan Ghori | Linear models: tested Elastic Regression Model and hubber regressor. |
| Rishabh Sharma | Linear models: decided the loss function for all the models and tested Ridge regression, Lasso regression and Bayesian regression.  Non-Linear models: designed no-linear model and tested non-linear model with degree-2 and degree-4. |

Description of Software and Tool:

The project was done in python 3 using the sci-kit learn library. The use of library was done to reduce the number of lines of the code and test different models. Sci-Kit Learn is a machine learning library that supports python. The library provides different kind of regression and classification models including linear regressiosn, SVM (Support vector machines) and perceptron-based regression techniques. The models were tested, and the data was designed in the Jupyter notebook.

Final Result:

**Linear Model:**

The validation split in the Linear Model is 75% training and 25% testing. The best model was ***Linear Regression model***(Y = XM + C) with mean squared error loss function with **error 0.98064016727619119.**

**Y = Output or prediction of the model**

**M = Weight matrix**

**X = Input Matrix**

**C = Bais or intercept.**

X = [Tm, Pr, Th, SV], M = [ [Coeff of Tm], [Coeff of Pr], [Coeff of Th], [Coeff of SV]] and

C = [Intercept]

The coefficients obtained after fitting the model Y=XM + C are following.

**M = [[-5.43021989e-01], [-1.53875767e-03], [3.62073874e+03], [5.46365171e-02]]**

**C = [-77.82293144693728]**

Therefore, we get the following parameter values.

**The Coefficients of Tm = -5.43021989e-01**

**The Coefficients of Pr = -1.53875767e-03**

**The Coefficients of Th = 3.62073874e+03**

**The Coefficients of Sv = 5.46365171e-02**

We also tested different regression models like Ridge regression (error= 2.2635005609774095), Lasso Regression (error= 2.2938888374011119), Lar-lasso regression (error= 6.2792854077105078), Bayesian Regression (error= 2.2656723604403948), Elastic Regression Model (error= 2.30) and hubber regressor (error= 6.00) with the same loss functions.

**Non-Linear Model:**

The validation split in the Non-Linear model is of 80% training and 20% testing. We tested mean squared error function with different degrees (2,3,4,5 and 6) and mean absolute error function. The best model was non-linear model with **mean squared error of degree 2 with value of 0.1024** and variance of 1. The computation time is not very high and the accuracy is fairly high for degree 2.

We are using a non-linear Model with a degree of 2 and the model is following.

**Y = w0\*1 + w1\*Tm + w2\*Pr + w3\*Th + w4\*Sv + w5\*Tm\*Pr + w6\*Tm\*Th + w7\*Tm\*Sv + w8\*Pr\*Th + w9\*Pr\*Sv + w10\*Th\*Sv + w11\*Tm\*Tm + w12\*Pr\*Pr + w13\*Th\*Th + w14\*Sv\*Sv + C.**

We used the scikit learn to fit the above model using the polynomial features from sklearn.preprocessing.

We obtain a weight matrix W by fitting the model.

We will consider these weights as ‘W’.

**W = [w0, w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14].**

**C = Intercept**

**W0 = Coefficient of ‘1’**

**W1 = Coefficient of ‘Tm’**

**W2 = Coefficient of ‘Pr’**

**W3 = Coefficient of ‘Th’**

**W4 = Coefficient of ‘Sv’**

**W5 = Coefficient of ‘Tm x Pr’**

**W6 = Coefficient of ‘Tm x Th’**

**W7 = Coefficient of ‘Tm x Sv’**

**W8 = Coefficient of ‘Pr x Th’**

**W9 = Coefficient of ‘Pr x Sv’**

**W10 = Coefficient of ‘Th x Sv’**

**W11 = Coefficient of ‘Tm x Tm’**

**W12 = Coefficient of ‘Pr x Pr’**

**W13 = Coefficient of ‘Th x Th’**

**W14 = Coefficient of ‘Sv x Sv’**

Therefore, the coefficients or the weights for the models are as following.

**w0 = 0**

**w1 = 1.29645686e+00**

**w2 = 2.44645077e-03**

**w3 = -5.30198600e+03**

**w4 = -5.62387217e-02**

**w5 = 5.69571846e-03**

**w6 = 1.01270151e-05**

**w7 = -6.98202003e+01**

**w8 = -2.40376702e-04**

**w9 = 6.92070898e-08**

**w10 = -3.50106427e-02**

**w11 = -5.79787684e-06**

**w12 = 1.89327211e+05**

**w13 = 2.29597734e-01**

**w14 = 1.00898003e-04**

**W = [ 0.00000000e+00, 1.29645686e+00, 2.44645077e-03, -5.30198600e+03,**

**-5.62387217e-02, 5.69571846e-03, 1.01270151e-05, -6.98202003e+01,**

**-2.40376702e-04, 6.92070898e-08, -3.50106427e-02, -5.79787684e-06,**

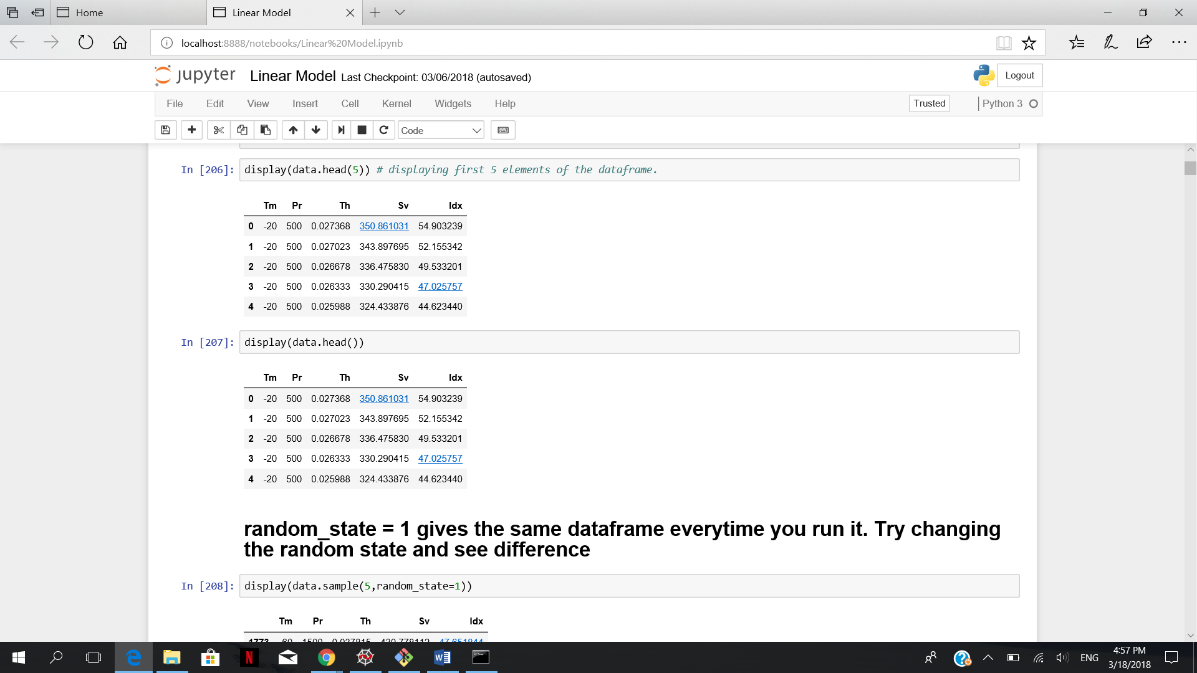
**1.89327211e+05, 2.29597734e-01, 1.00898003e-04]**

**C = 46.41151840838802**

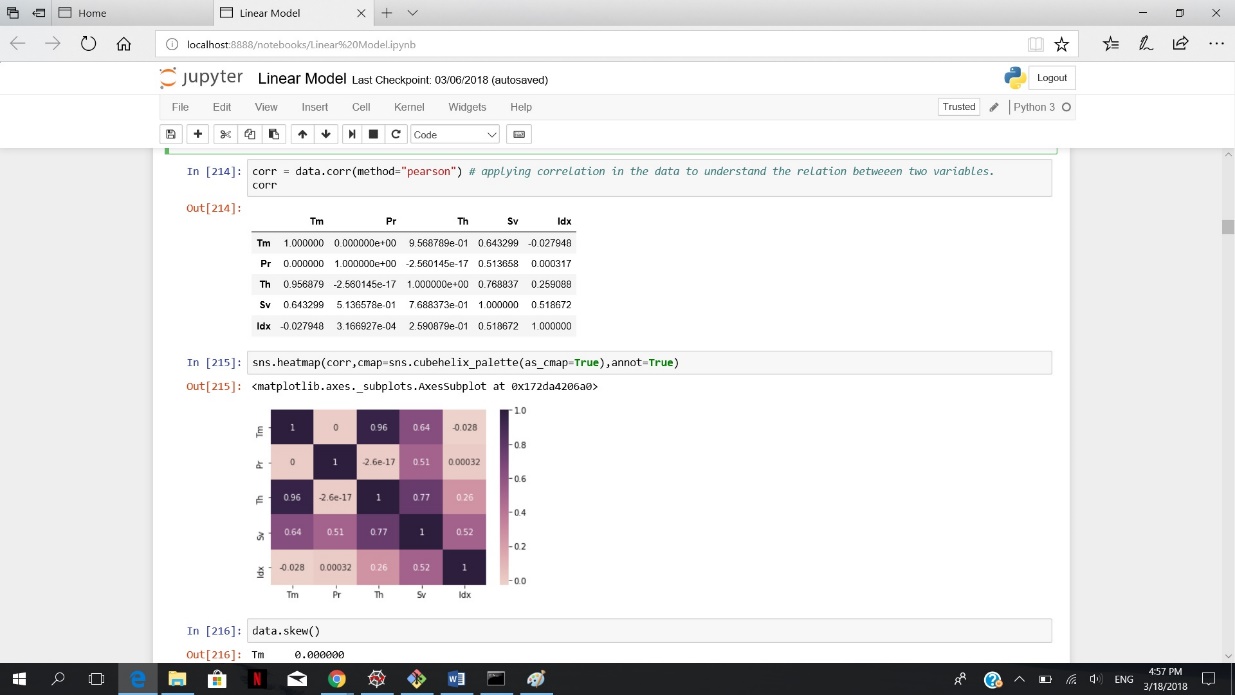
So, we decided to use the Non-Linear model with the degree 2 because the computation time is not very high, and the accuracy is high. The experiments and results showed that the best lost function was the mean squared error.

Test Screen shots:

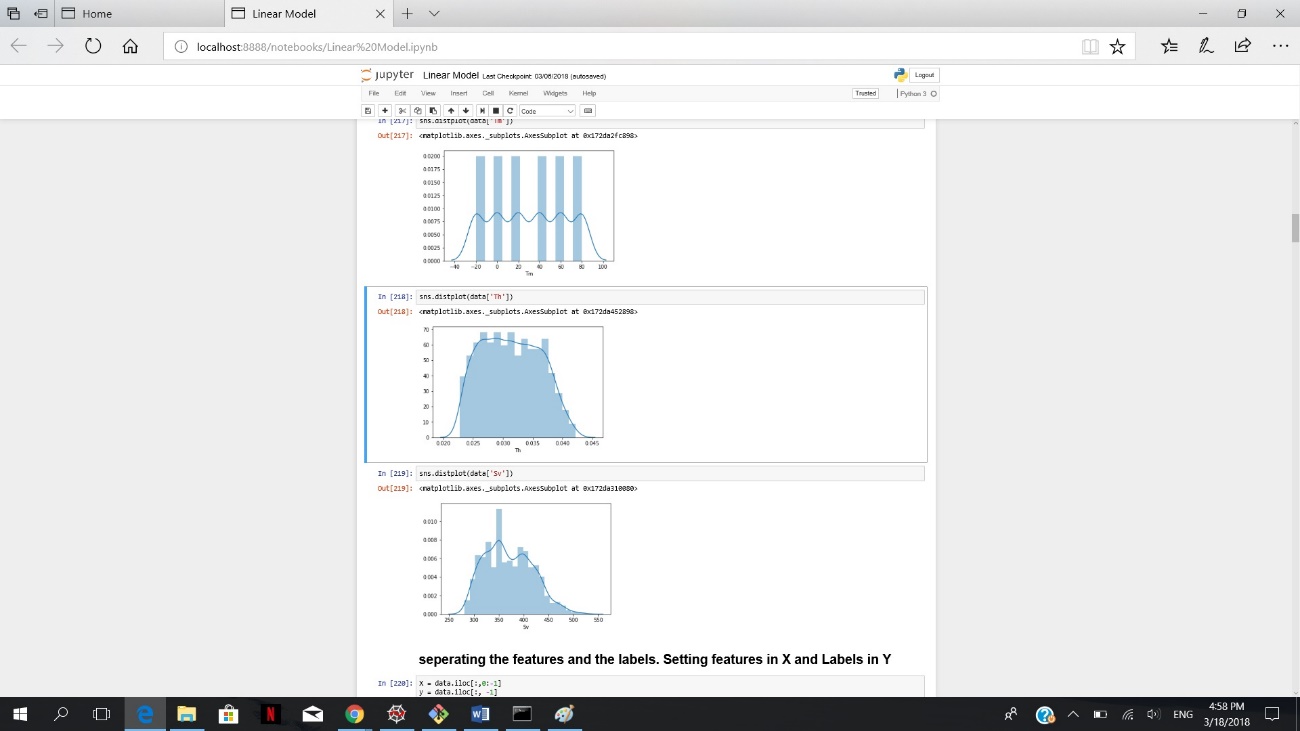
Please find the attached screen shots of the jupyter notebook. The screen shots show the data preprocessing that was done to find the correlation between the independent variables and to see the distribution of the data points.



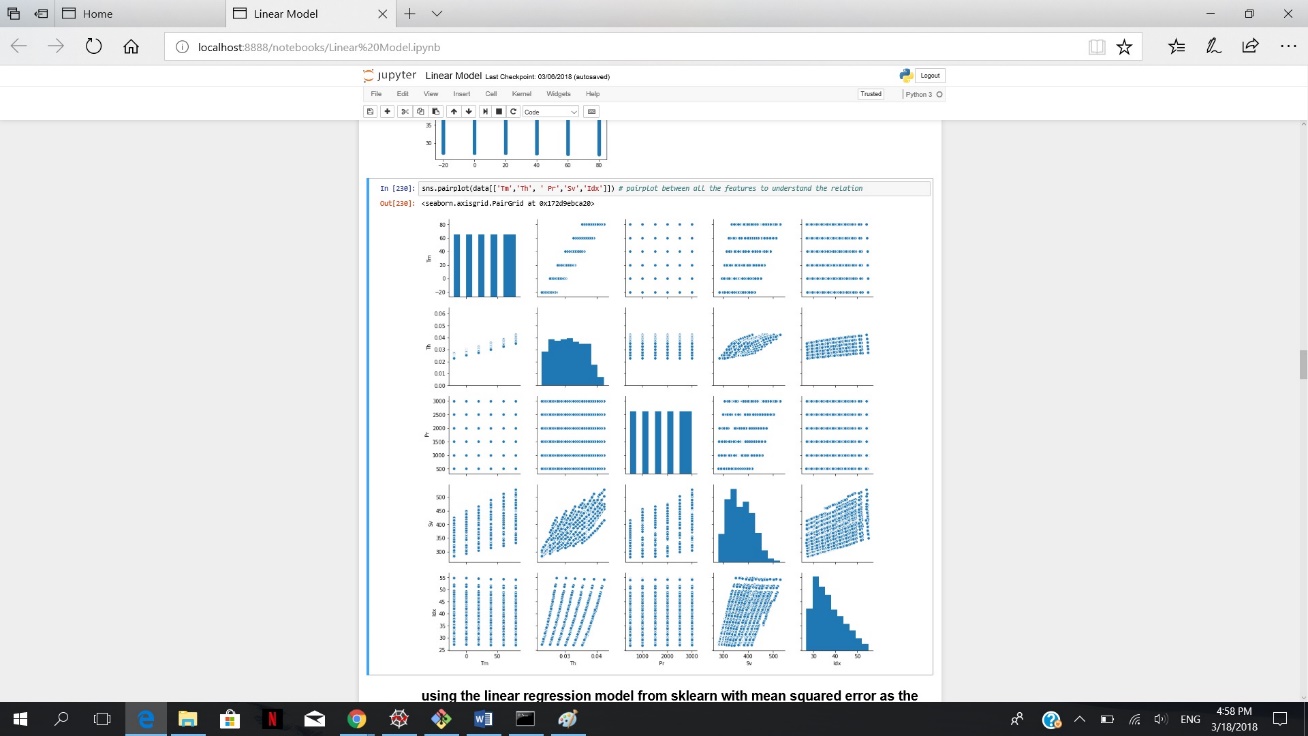
The data image



The Correlation Matrix



Data Distribution



Pair Distribution of the features.

Instruction to run program:

To run the files and to see the results please install the required libraries, numpy, sci-kit learn.

To run the linear model, use the command in terminal: python Linear\_Model.py.

To run the non-linear model in python, use the command: python Non\_linear\_model.py