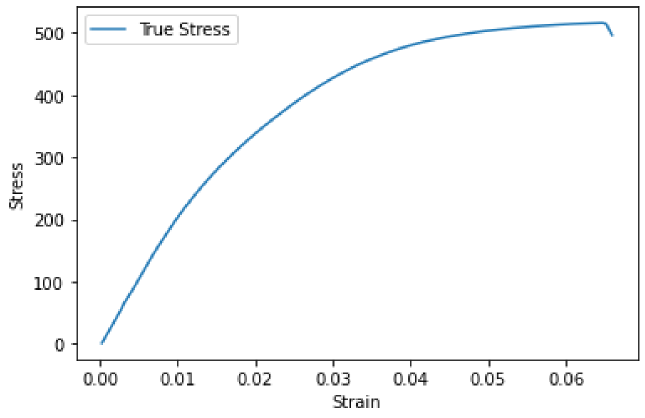
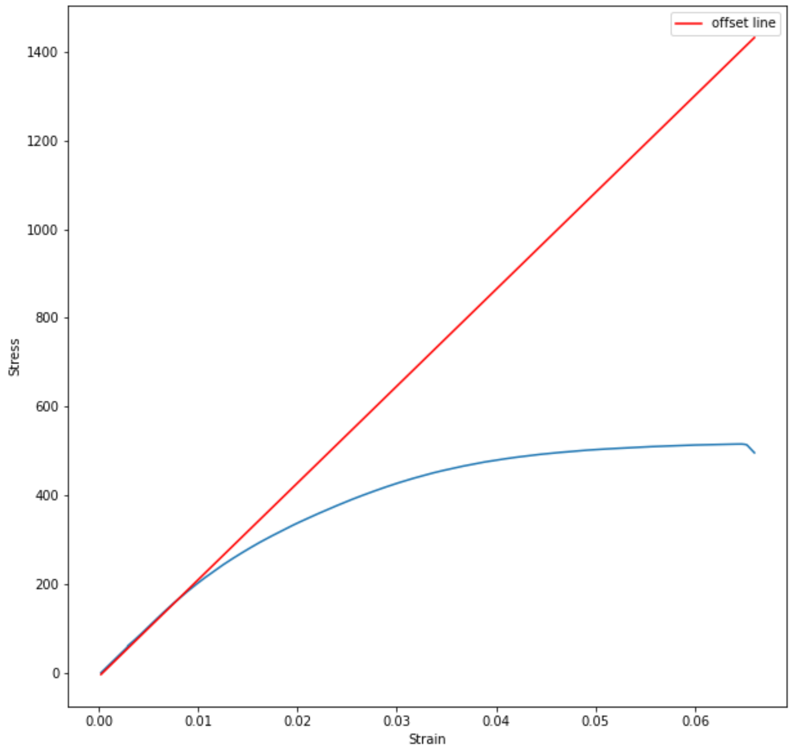
Stress-Strain Error prediction

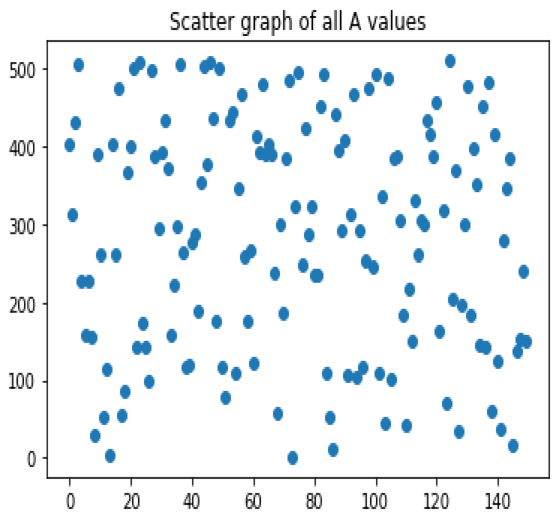
We begin by importing the libraries and reading the stress-strain dataset. The curve started from negative stress values so another dataset was made from only the positive stress points. The stress-strain graph obtained from the the given details is as follows:

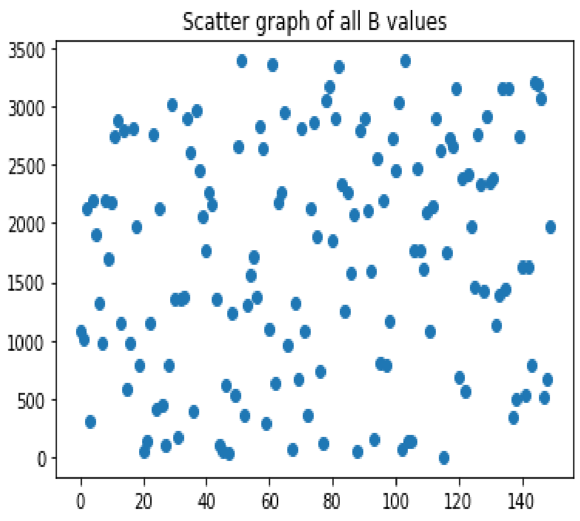


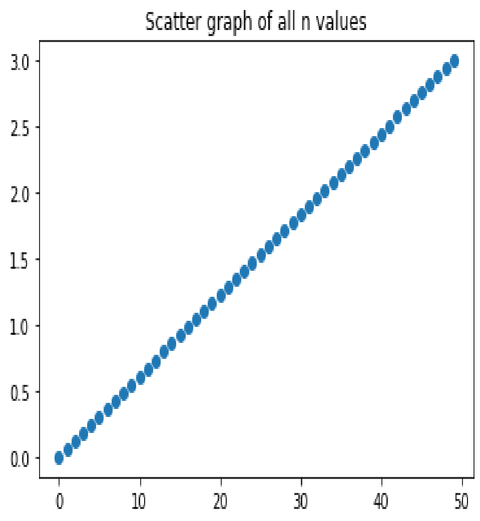
As this graph is a non-linear one, yield stress could be determined by the offset method. So we calculated the initial slope of the graph and plotted a line having the same slope and x-intercept as 0.2% of maximum strain.



The point where the two graphs intersect correspond to the stress value of 165.8250061 which becomes the yield stress of the graph. After generating the yield stress we try to fit the plastic region of the graph into a polynomial function $A + B * X^n$, where X corresponds to the strain part of the graph. So, for this we firstly generate 150 random values of A in between 0 and 515, 150 random values of B between 0 and 3,500 and 50 evenly spaced values of n between 0 and 3.



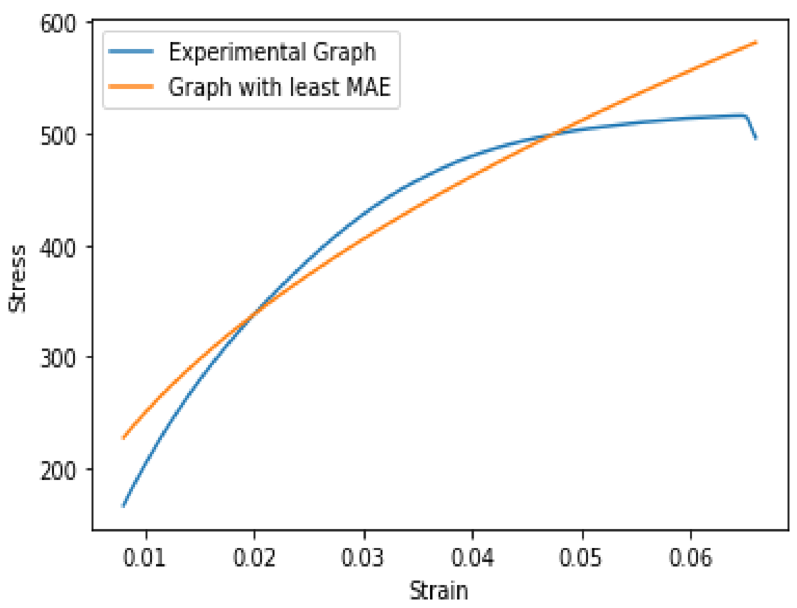


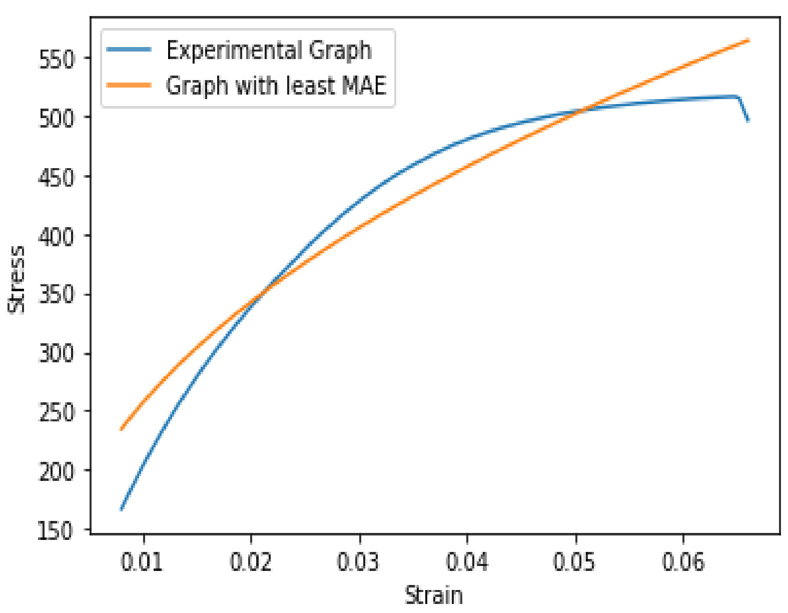


We map all these generated values and create a new dataset having columns A,B,n,MAE and RMSE of shape (150*150*50,5). MAE and RMSE are calculated between the actual stress value and the stress value calculated by the above-mentioned formula using A,B and n. After this we save the dataset for further use as we do not want to generate these values everytime we restart the kernel. The best fit generated graph from the parameters A,B and n according to least RMSE and MAE is shown in the figure below,

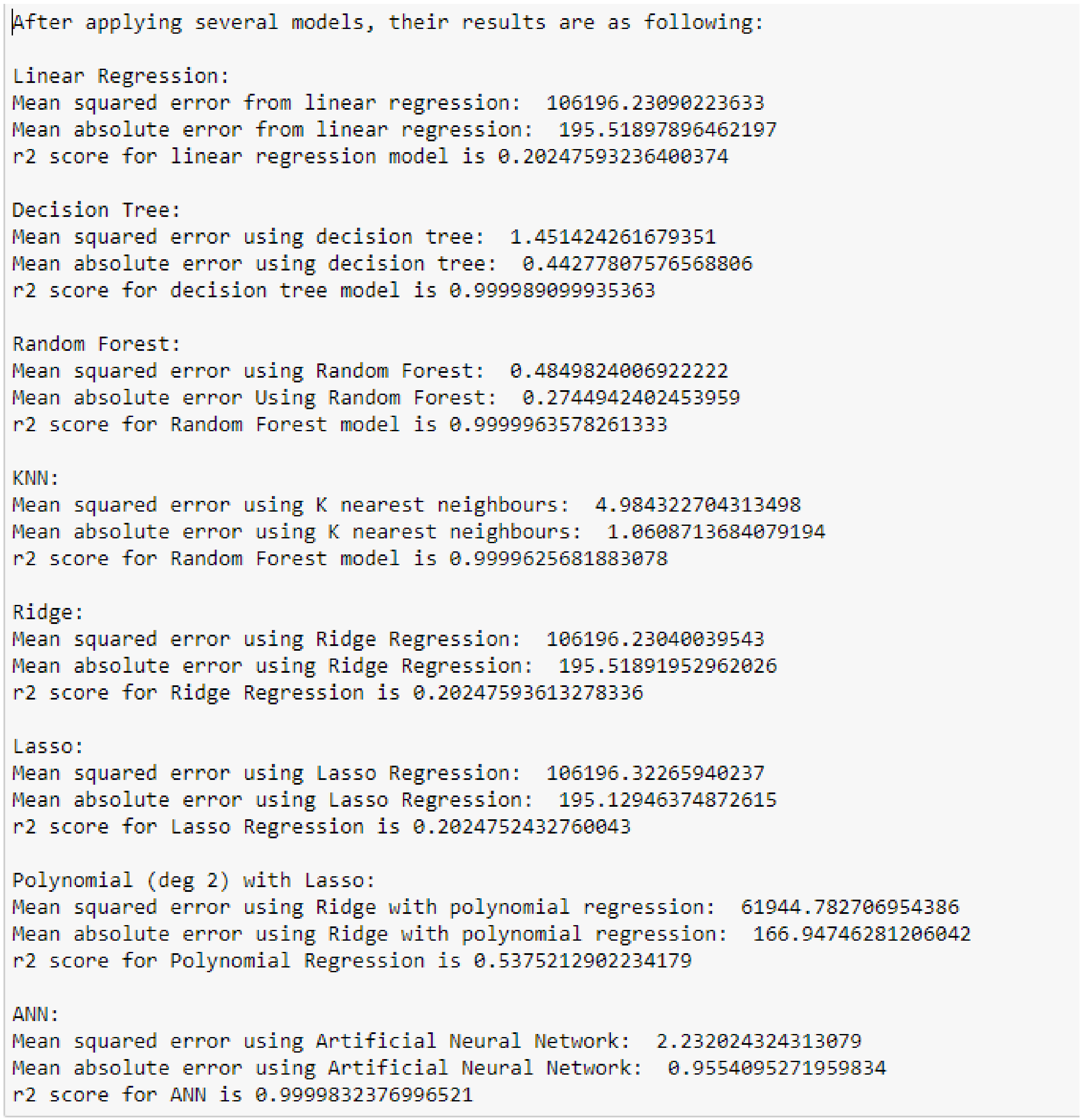
```
A          25.182481
B        2076.266949
n           0.489796
```

```
A          10.175853
B        1774.145451
n           0.428571
```





Now we split our dataset into train and test set, use standard scaler for normalization and apply different machine learning models on our train dataset.



```
After applying several models, their results are as following:

Linear Regression:
Mean squared error from linear regression:  106196.23090223633
Mean absolute error from linear regression:  195.51897896462197
r2 score for linear regression model is 0.20247593236400374

Decision Tree:
Mean squared error using decision tree:  1.451424261679351
Mean absolute error using decision tree:  0.44277807576568806
r2 score for decision tree model is 0.999989099935363

Random Forest:
Mean squared error using Random Forest:  0.4849824006922222
Mean absolute error Using Random Forest:  0.2744942402453959
r2 score for Random Forest model is 0.9999963578261333

KNN:
Mean squared error using K nearest neighbours:  4.984322704313498
Mean absolute error using K nearest neighbours:  1.0608713684079194
r2 score for Random Forest model is 0.9999625681883078

Ridge:
Mean squared error using Ridge Regression:  106196.23040039543
Mean absolute error using Ridge Regression:  195.51891952962026
r2 score for Ridge Regression is 0.20247593613278336

Lasso:
Mean squared error using Lasso Regression:  106196.32265940237
Mean absolute error using Lasso Regression:  195.12946374872615
r2 score for Lasso Regression is 0.2024752432760043

Polynomial (deg 2) with Lasso:
Mean squared error using Ridge with polynomial regression:  61944.782706954386
Mean absolute error using Ridge with polynomial regression:  166.94746281206042
r2 score for Polynomial Regression is 0.5375212902234179

ANN:
Mean squared error using Artificial Neural Network:  2.232024324313079
Mean absolute error using Artificial Neural Network:  0.9554095271959834
r2 score for ANN is 0.9999832376996521
```