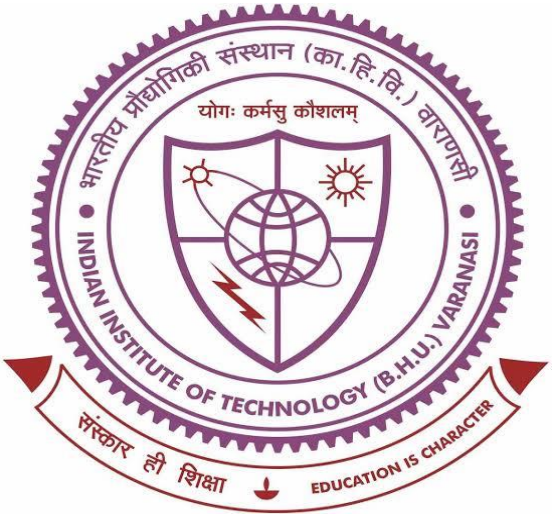


UG PROJECT REPORT (2020-21)

Department of Metallurgical Engineering

Topic: Prediction of mechanical properties of low alloy steels using machine learning.

Mentor Professor : Dr. Sudipta Patra

Team Leader : Rohan Nemade (18145054)

Team Members : Shreesh Roliwal (18145055)

Yashwardhan Chhipa (18145068)

Index:

- Introduction
- Terminology
- The dataset
- Neural Network
- Random Forest
- Comparison of both models
- Conclusion
- Links

Introduction:

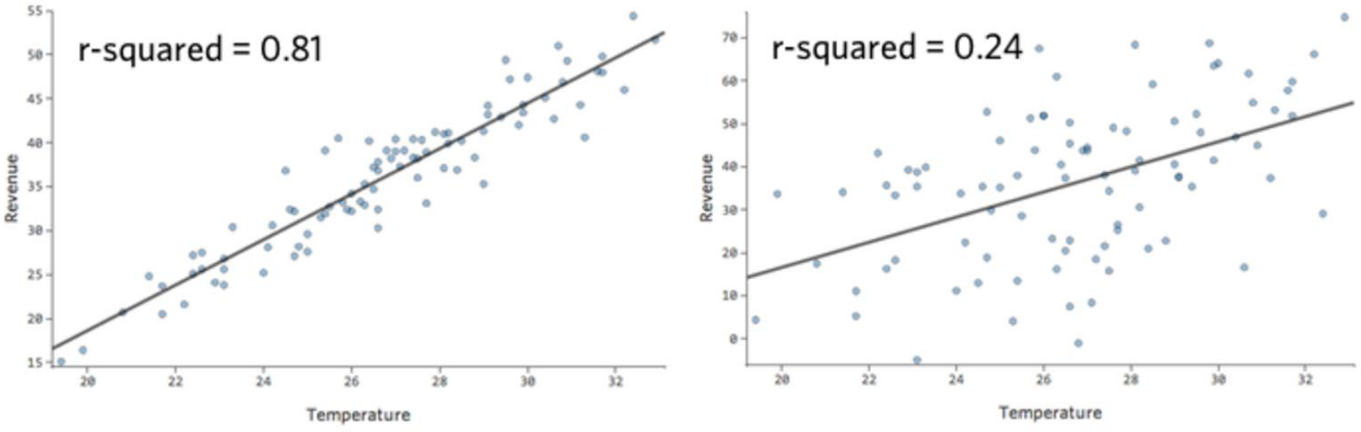
Now-a-days, machine learning has found applications in many diverse sectors including finance, healthcare, material science, etc. Machine learning can help one to leverage the power of data by providing valuable insights in the data or by making precise predictions based on patterns that are difficult for humans to see.

The aim of this project is to predict mechanical properties like tensile strength, 0.2% proof strength, % elongation and % reduction in area of low-alloy steels using machine learning techniques. These mechanical properties are predicted on the basis of the composition of the alloy (wt.% of each element) and the temperature. Most efforts to predict these properties are based on statistical inferences made from the data obtained by practical testing of these materials. We implemented supervised ML techniques and after hyper-parameter tuning achieved a maximum R^2 score of 0.91 using Random Forest Regressor. All the data preprocessing, model building, predictions and visualizations were done in Python programming language on Google Colab framework.

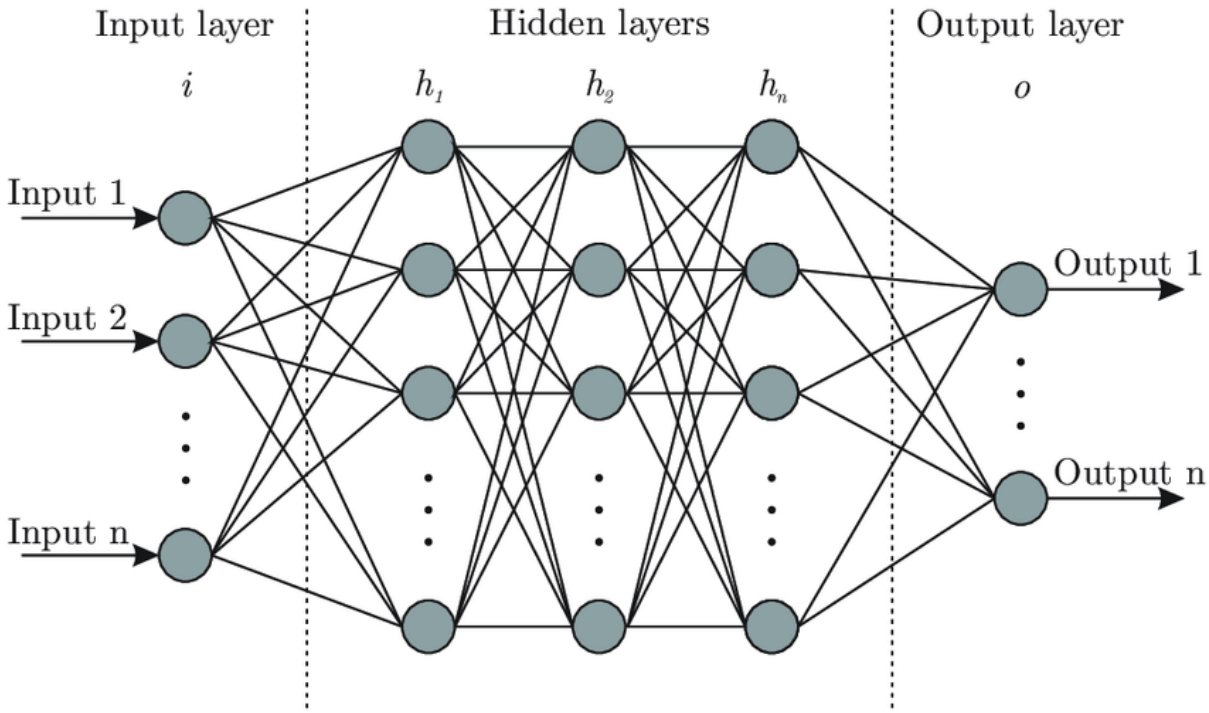
Terminology:

- **0.2% Proof Strength:** The 0.2% proof strength (0.2% OYS, 0.2% offset yield strength, RP0.2, RP0,2) is defined as the amount of stress that will result in a plastic strain of 0.2%.
- **Tensile strength:** Tensile strength (shortened form of ultimate tensile strength or UTS) is the maximum value of stress that the material can withstand before fracture.
- **% Elongation:** Percent elongation quantifies the ability of an element or compound to stretch up to its breaking point. Materials with a higher percentage elongation can stretch more before breaking. It is expressed as:
  % Elongation = (change in length (up to the breaking point)) / (original length) x 100.
- **% Reduction in area:** Reduction in area is the difference between an original given cross-sectional area of a test specimen before being subjected to tension and the given area of its smallest cross-section after rupture at the conclusion of the test, with the original cross-sectional area expressed as a percentage.
- **R^2 score:** R-squared (R^2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. R-squared explains to what extent the variance of one variable explains the variance of the second variable. So, if the R^2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

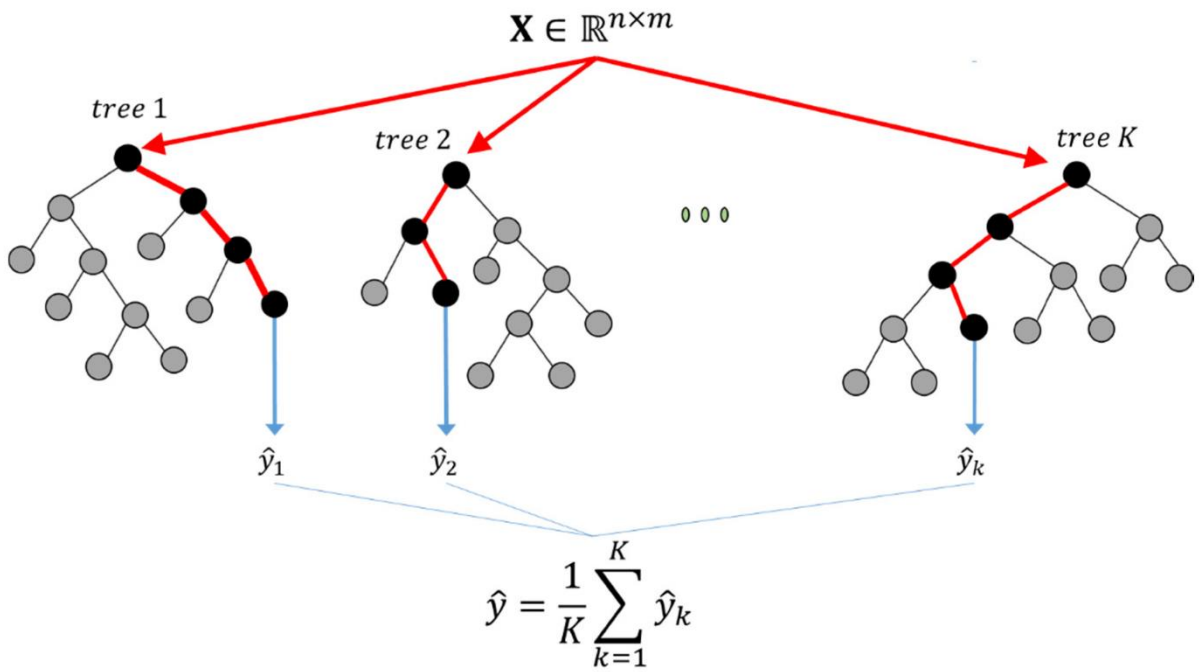
$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$



- **Neural Networks:** Neural networks are a series of algorithms that mimic the operations of a human brain to recognize relationships between vast amounts of data. A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.



- **Random Forest:** A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.



The Dataset:

The raw data was sourced from NIMS Matnavi database Structural Materials Fatigue datasheet. The data was in the form of PDF files each containing data for a specific group of alloys based on composition in the form of tables.

Using online tools, the PDFs were unlocked, cleaned and table were extracted into CSV files.

Table 3. Short-time tensile properties of 1Cr-0.5Mo steel tubes

0.2 % proof stress and tensile strength

NRIM reference code	Test temperature (℃)									
	RT	100	200	300	400	450	500	550	600	650
Require-ment[1]	≥205 ≥410	0.2 % proof stress (MPa)[2] Tensile strength (MPa)								
MBB	342 490	338 454	337 465	346 495	316 489	287 461	274 431	262 387	220 314	152 226
MBC	310 463	285 427	296 439	265 471	257 472	240 451	232 423	207 367	176 297	132 205
MBD	306 463	258 432	303 459	275 473	264 464	252 451	237 424	228 378	212 297	143 221
MBE	343 502	326 467	329 482	277 505	249 493	253 475	255 434	229 383	202 301	143 221
MBF	339 493	329 464	315 475	270 497	257 482	256 465	256 442	238 389	213 308	146 226
MBG	251 456	240 420	261 449	203 479	192 482	186 471	192 435	177 375	170 304	122 211
MBH	239 453	229 419	244 449	210 485	174 474	194 468	181 434	175 380	170 296	124 214
MBJ	241 431	237 406	243 431	188 459	168 454	176 444	160 417	172 376	161 293	114 199
MBL	313 464	301 421	324 459	303 479	279 482	277 473	262 449	251 395	220 317	137 213
MBM	291 462	315 418	316 457	288 479	270 482	272 479	256 441	240 393	218 299	136 217
MBN	312 452	304 420	311 454	242 529	291 483	280 479	260 448	240 383	215 304	150 220

A	B	C	D	E	F	G	H	I	J	K
code	RT	100	200	300	400	450	500	550	600	650
MBB	342 490	338 454	337 465	346 495	316 489	287 461	274 431	262 387	220 314	152 226
MBC	310 463	285 427	296 439	265 471	257 472	240 451	232 423	207 367	176 297	132 205
MBD	306 463	258 432	303 459	275 473	264 464	252 451	237 424	228 378	212 297	143 221
MBE	343 502	326 467	329 482	277 505	249 493	253 475	255 434	229 383	202 301	143 221
MBF	339 493	329 464	315 475	270 497	257 482	256 465	256 442	238 389	213 308	146 226
MBG	251 456	240 420	261 449	203 479	192 482	186 471	192 435	177 375	170 304	122 211
MBH	239 453	229 419	244 449	210 485	174 474	194 468	181 434	175 380	170 296	124 214
MBJ	241 431	237 406	243 431	188 459	168 454	176 444	160 417	172 376	161 293	114 199
MBL	313 464	301 421	324 459	303 479	279 482	277 473	262 449	251 395	220 317	137 213
MBM	291 462	315 418	316 457	288 479	270 482	272 479	256 441	240 393	218 299	136 217
MBN	312 452	304 420	311 454	242 529	291 483	280 479	260 448	240 383	215 304	150 220

For each group of alloys, three CSV files were obtained each containing the compositon, tensile and 0.2% proof strength and % elongation and % reduction in area. This process was done for all PDF files.

All the CSV files were then rearranged in a way to serve the purpose of the project and combined to form one single file containing coloums of wt percentages of alloying elements, temperature and the mechanical properties. This was done using python script.

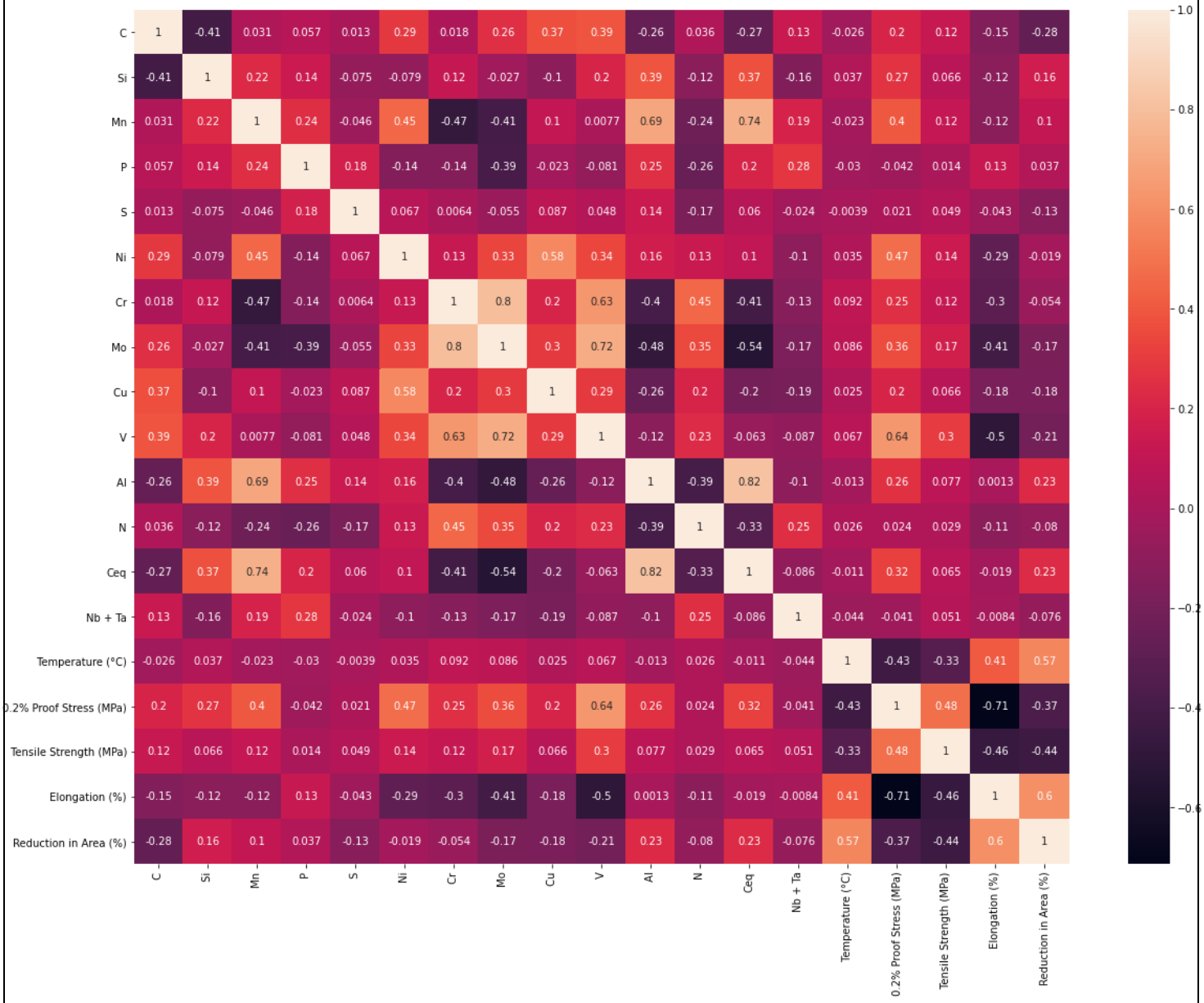
```
import pandas as pd
import numpy as np
num = 40
comp = pd.read_csv('/content/drive/MyDrive/UG Project/' + str(num) + 'comp.csv')
strength = pd.read_csv('/content/drive/MyDrive/UG Project/' + str(num) + 'strength.csv')
percentage = pd.read_csv('/content/drive/MyDrive/UG Project/' + str(num) + ' %.csv', header=None)
coloumn_names = []
coloumn_names += list(comp)
coloumn_names += ['Temperature (°C)', '0.2% Proof Stress (MPa)', 'Tensile Strength (MPa)', 'Elongation (%)', 'Reduction in Area (%)']
temperatures = list(strength)[1:]
codes = list(comp['code'])
metals = list(comp)[1:]
main_list = []
main_list.append(coloumn_names)
for i in range(len(codes)):
    for j in range(len(temperatures)):
        lst = []
        lst.append(codes[i])
        for k in range(len(metals)):
            lst.append(comp[str(metals[k])][i])
        lst.append(temperatures[j])
        x = strength[str(temperatures[j])][i]
        y = percentage[j+1][i]
        proof = x[:3]
        tensile = x[4:]
        elong = y[:2]
        redu = y[3:]
        lst.append(proof)
        lst.append(tensile)
        lst.append(elong)
        lst.append(redu)
        main_list.append(lst)

np.savetxt(str(num) + '.csv', X=main_list, delimiter=', ', fmt='% s')
```

Features of dataset:

The dataset contains 17 columns and 915 rows. The ranges of the input features are shown in the table below –

	C	Si	Mn	P	S	Ni	Cr	Mo	Cu	V	Al	N	Ceq	Nb + Ta	Temperature (°C)
MAX	0.34	0.52	1.48	0.03	0.022	0.6	1.31	1.35	0.25	0.3	0.05	0.015	0.437	0.0017	650
MIN	0.09	0.18	0.42	0.006	0.003	0	0	0.005	0	0	0.002	0.0025	0	0	27
AVERAGE	0.17	0.31	0.81	0.01	0.01	0.14	0.43	0.44	0.08	0.06	0.01	0.01	0.09	0.00	351.60

The effects of different alloying elements and temperature on the mechanical properties were visualized by calculating pearson correlation coefficients for each pair of variables and then plotting their heatmaps and bargraphs.

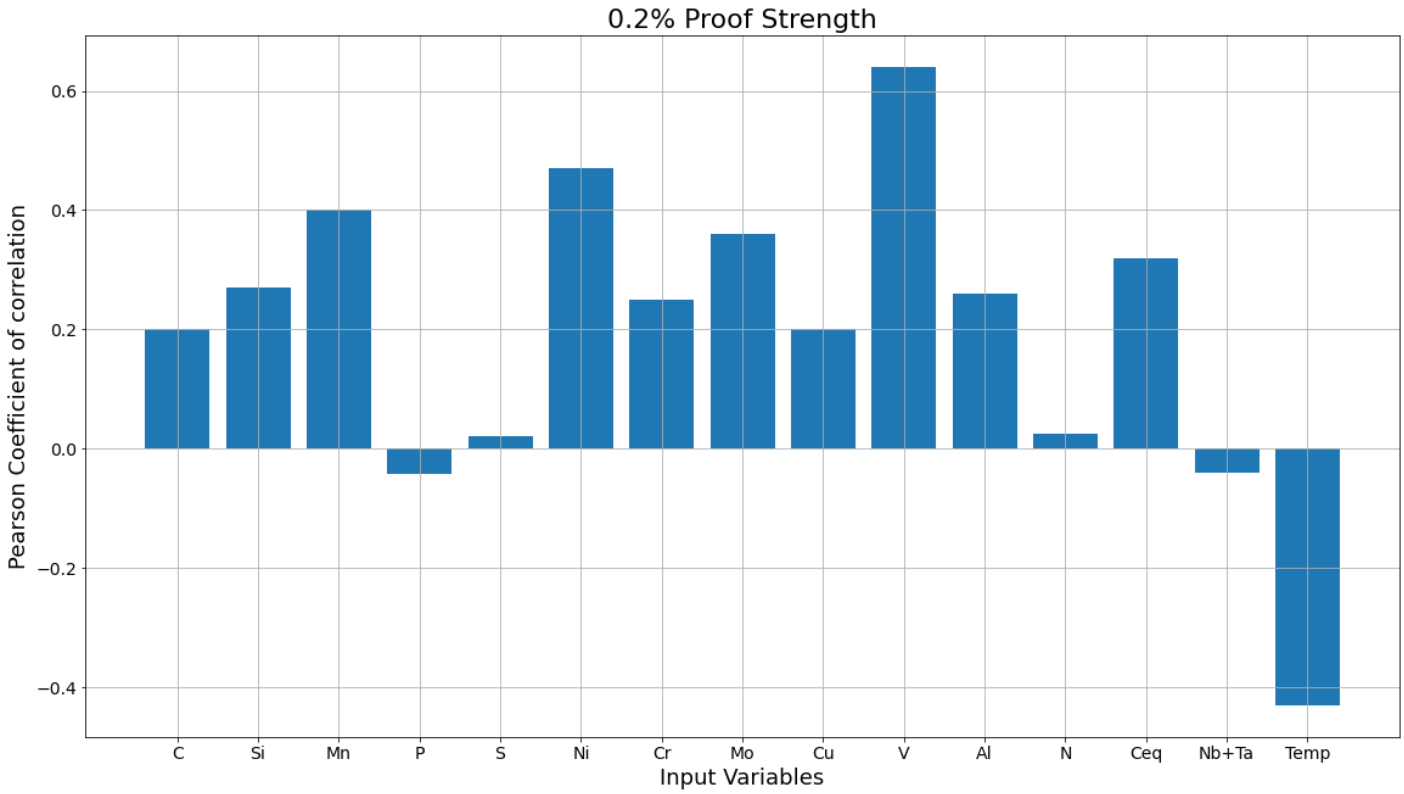


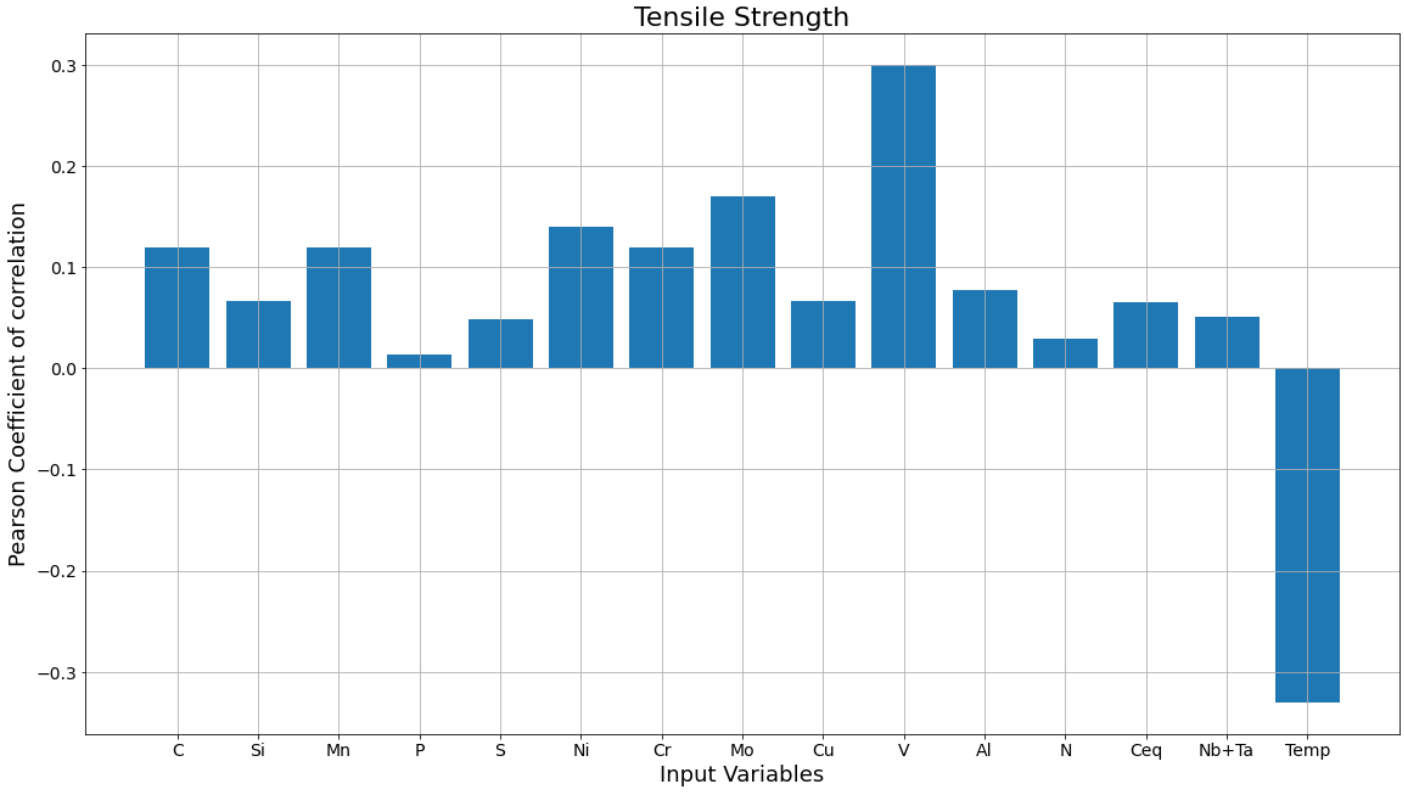
From above diagram following conclusions were made:

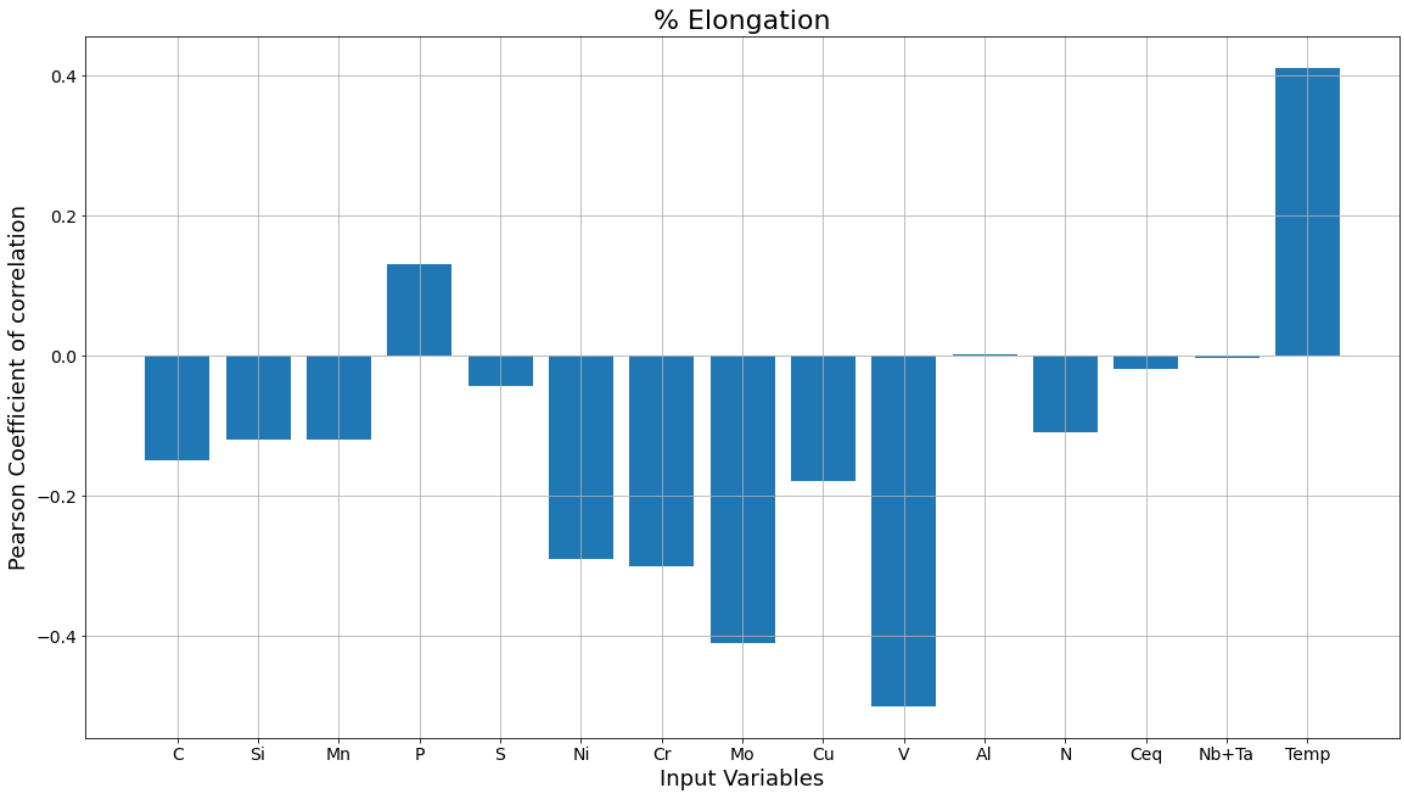
1. Temperature has significant influence on % Elongation and % Reduction in area.
2. 0.2% Proof Strength is highly influenced by presence of V, Ni, Mn, Mo, Ceq, Si, Al, Cr, C and Cu in decreasing order.
3. Tensile Strength is highly influenced by presence of V and moderately influenced by presence of Mo, Ni, Cr, C and Mn in decreasing order.

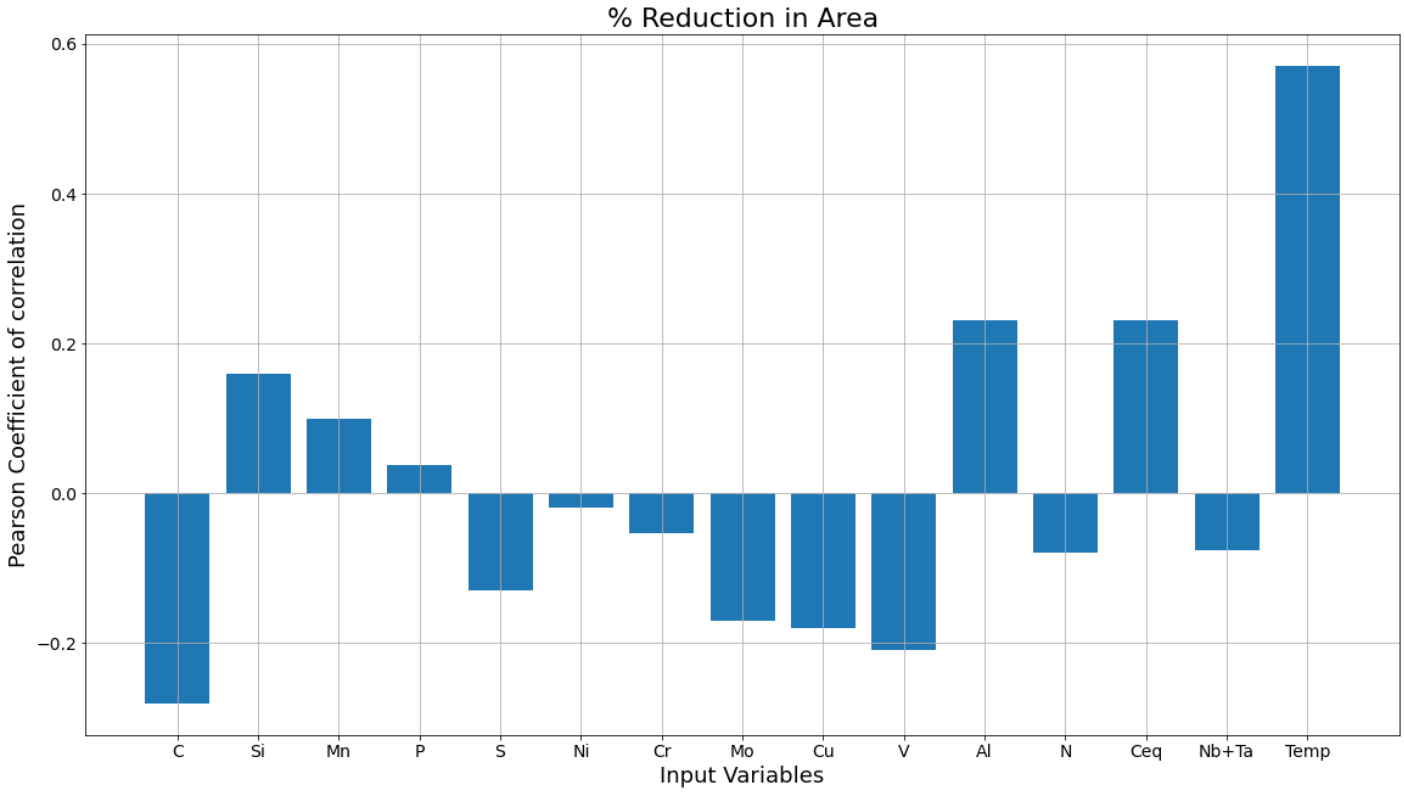
4. Tensile Strength is also highly related to 0.2% Proof Strength.
5. % Elongation and % Reduction in Area show maximum correlation with each other followed by temperature.
6. % Elongation is moderately influenced by presence of P and slightly influenced by presence of Al
7. % Reduction in Area is moderately influenced by presence of Al, Ceq, Si and Mn.

The following bar graphs show better visualizations for the above results:









However finally all the variables were used for predicting the results in order to build a single model capable of predicting all four properties.

The dataset was then divided into two parts – x and y representing inputs and outputs respectively. Again, using train_test_split function from sci-kit learn library, the dataset was shuffled randomly and split into two parts for training ang testing. 20% of the rows were kept for testing the predictions. Thus finally four arrays were obtained namely – x_train, x_test, y_train, y_test.

The input and output features were then scaled down using standard scalar method from sci-kit learn library in python. It scales down the features according to the formula – scaled_value = (x - mean)/(std_dev)

Neural Network model:

Neural networks were initially chosen to perform the task because of their higher flexibility to fit into complex multi-dimensional data. The following neural network was implemented in python programming language using keras library.

The neural network implemented has a total of 8 layers containing 15,30,45,40,30,20,10 and 4 nodes respectively. The first layer is the input layer and has 15 nodes for 14 alloying elements and temperature. The last layer is the output layer and has 4 nodes for the 4 mechanical properties. Tanh activation function was observed to produce better results than Relu and sigmoid. The model was trained on batch size of 256 for 1000 epochs using adam optimizer and mean squared error for loss and metrics. All these hyperparameters were decided after extensive iterative manual tuning and re-training of the network by altering number of layers, number of nodes in each layer, activation functions, batch size and optimizers.

```
model = Sequential()
model.add(Dense(units = 15, kernel_initializer = 'normal', activation = 'tanh', input_dim = 15))
model.add(Dense(units = 30, kernel_initializer = 'normal', activation = 'tanh'))
model.add(Dense(units = 45, kernel_initializer = 'normal', activation = 'tanh'))
model.add(Dense(units = 40, kernel_initializer = 'normal', activation = 'tanh'))
model.add(Dense(units = 30, kernel_initializer = 'normal', activation = 'tanh'))
model.add(Dense(units = 20, kernel_initializer = 'normal', activation = 'tanh'))
model.add(Dense(units = 10, kernel_initializer = 'normal', activation = 'tanh'))
model.add(Dense(units = 4, kernel_initializer = 'normal', activation = 'tanh'))

model.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics = ['mean_squared_error'])
```

```
[ ]  # Training the model and predicting the results
     history = model.fit(x_train_sc, y_train_sc, batch_size = 256, shuffle=True, epochs = 10000)
     y_nn_pred_sc = model.predict(x_test_sc)
```

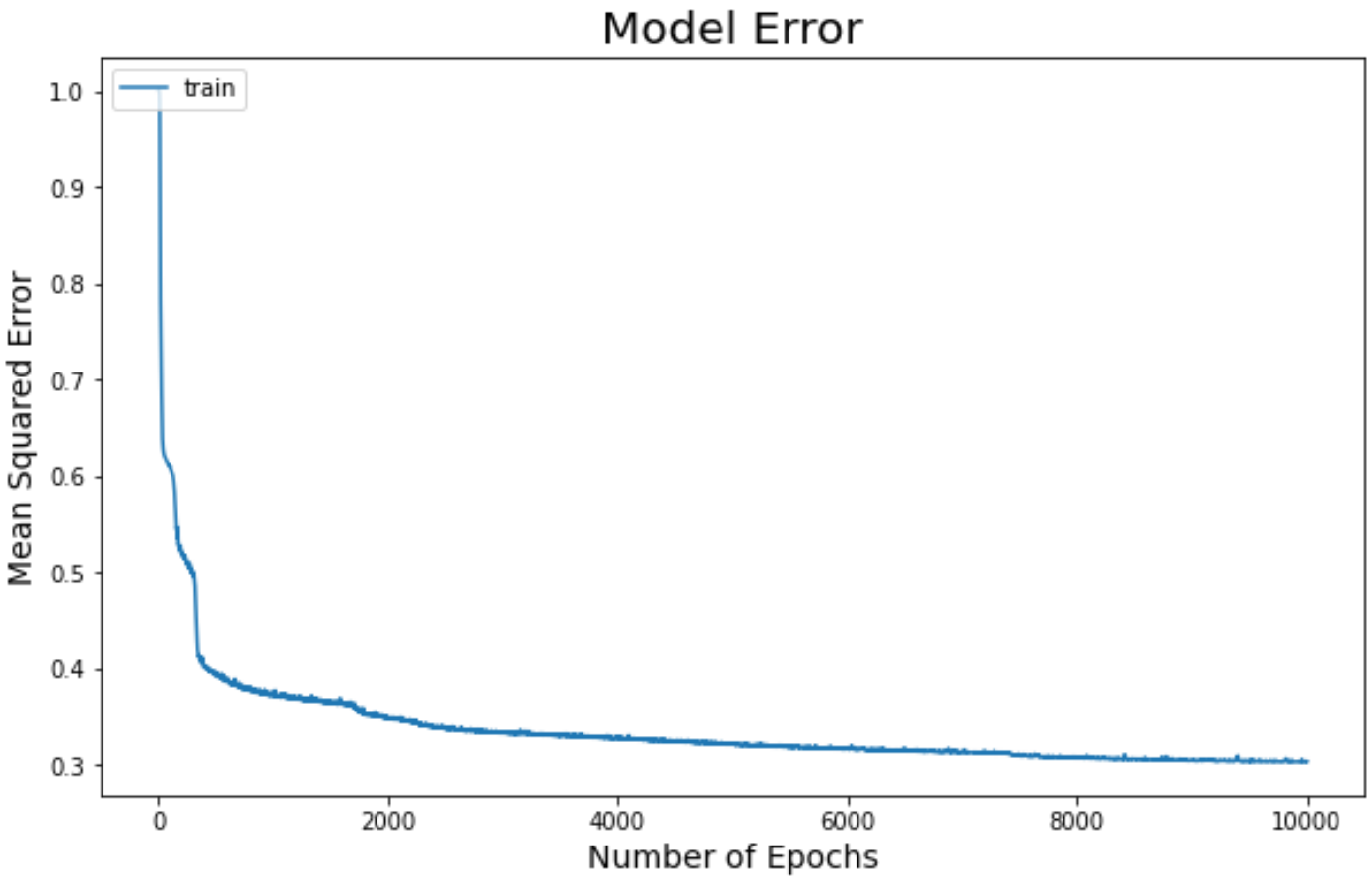
Best results were obtained with the above hyper-parameters as follows –

R^2 score = 0.82

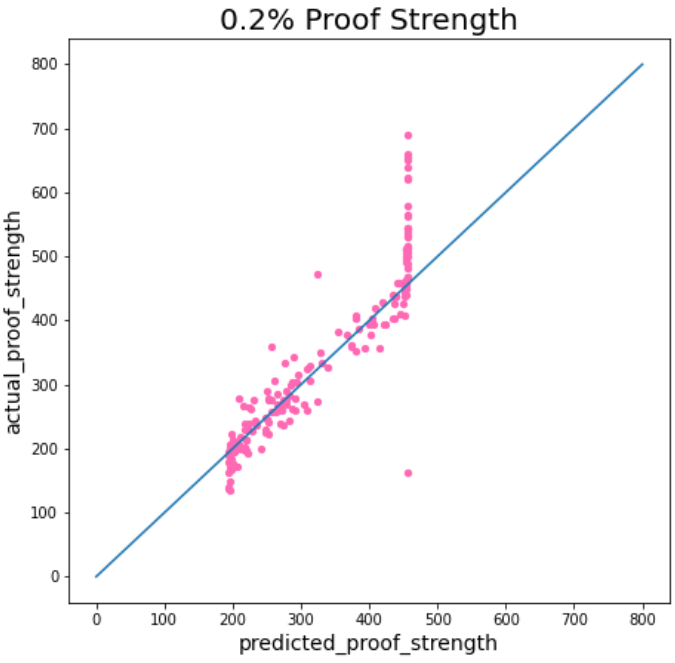
Mean squared error = 0.14 (for scaled outputs)

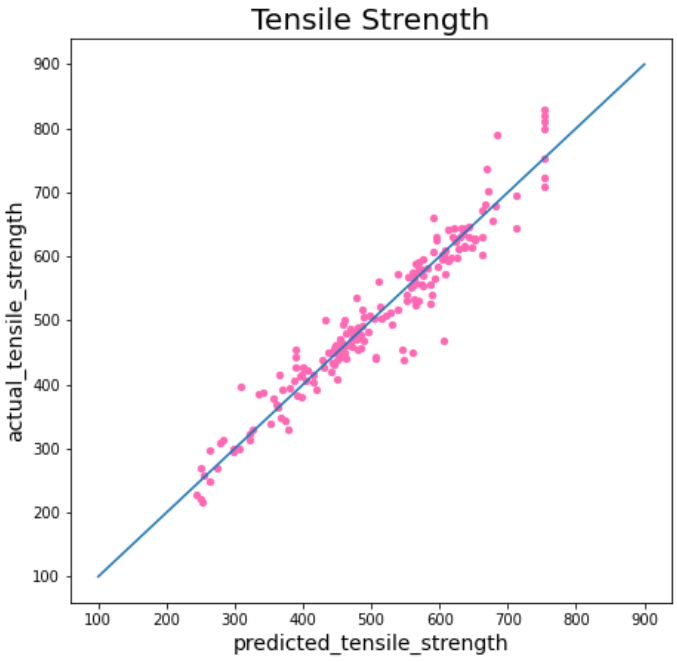
Mean absolute error = 0.24 (for scaled outputs)

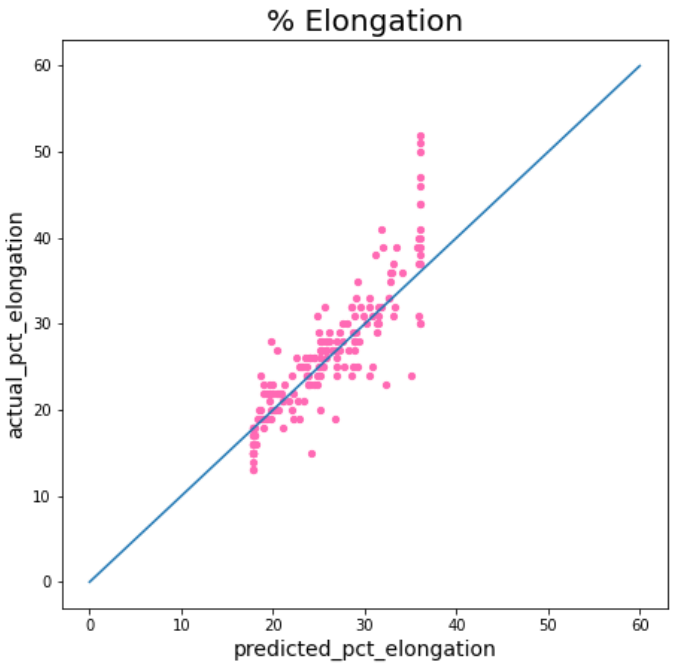
The training history of the model was recorded and plotted to visualize the convergence of loss.

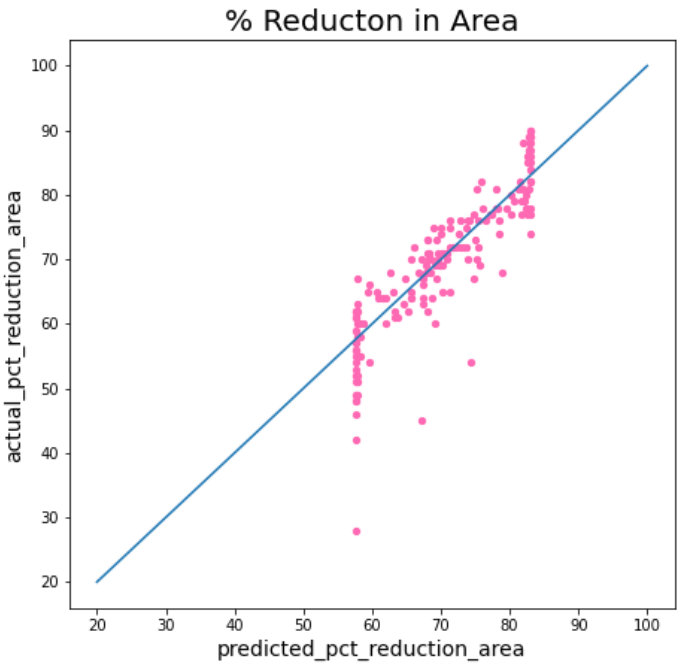


The outputs of the model were scaled up and were plotted against actual values along with a y = x line in order to visualize the error. The closer the points are to the line the better the lower is the error in predictions. If the point is above the line, then the predicted value is lesser than the actual value and vice versa.









The R^2 score for each mechanical property was calculated –

R^2_score for 0.2% Proof Strength = 0.82
R^2_score for Tensile strength = 0.93
R^2_score for % Elongation = 0.75
R^2_score for % Reduction in Area = 0.79

Thus, the model predicted tensile strength with very good accuracy followed by 0.2% proof strength, % reduction in area and % elongation.

From the plots it can be observed that the model was able to predict mechanical properties with decent precision for most of the points but few values deviated by a large value. Another observation was that the model was not able to predict 0.2% Proof strength above 460 MPa and below 200MPa. Similar trends were observed in % Elongation and % Reduction in area.

This trend was occurring due to the Standard scalar method and tanh activation function. For 0.2% Proof Strength, the scalar scaled down the values ranging from (114 – 690 MPa) to (-2.28 – 2.75). Now the tanh activation function always scales down the inputs ranging from (-∞ – +∞) to (-1 – +1). Thus, the inputs were greater than 1 and lesser than -1 but the output were between -1 and 1 and upon scaling them up to original values they didn't exceed a particular limit (200 – 460 MPa for 0.2% proof strength). Thus, the model was trying to fit to these points but was restricted by the tanh function.

	Inputs		Scaled Inputs		Scaled Outputs		Outputs
Min	114	⇨	-2.28	⇨	-1	⇨	200
Max	690		+2.75		+1		460

This issue could have been solved by using a different scaling method or by using a different ML algorithm which does not have provide outputs within a specific range.

Hence, we decided to use Random Forest Regressor to overcome this problem.

Random Forest Regression Model:

The next choice after neural networks was Random Forest algorithm. The random forest regression algorithm was implemented in python programming language using sklearn library. This method was chosen for two reasons –

1) It works on an ensemble of many decision trees making the model highly robust throughout the range of inputs. This is due to the fact that each prediction is made by a number of different decision trees and all the predictions are averaged out to provide a final output.
2) Each decision tree works by splitting the data based on many simple conditions. Thus, many splits are made and the data is divided into many areas. Then a new prediction is made according to which area satisfies the splitting conditions foe the given inputs. This eliminates the dependence on activation functions and can work with simple scaling methods.

However Random Forest Regression was not the first choice because initially we were not sure if the dataset was discernible enough for simple splitting to provide accurate results. But it turned out that it was!

Other perks of using this algorithm were easy manual hyper-parameter tuning and significantly lower training time as compared to neural networks.

The random forest regressor implemented here consisted of 100 decision trees and the criterion was set to mean squared error.

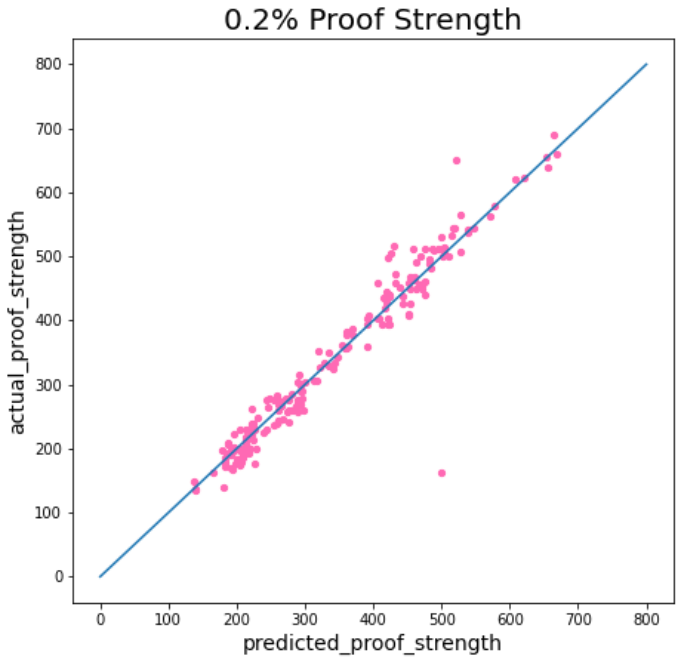
```
# Random Forest Regressor
regressor = RandomForestRegressor(n_estimators=100, criterion='mse')
regressor.fit(x_train_sc, y_train_sc)
y_rf_pred_sc = regressor.predict(x_test_sc)
```

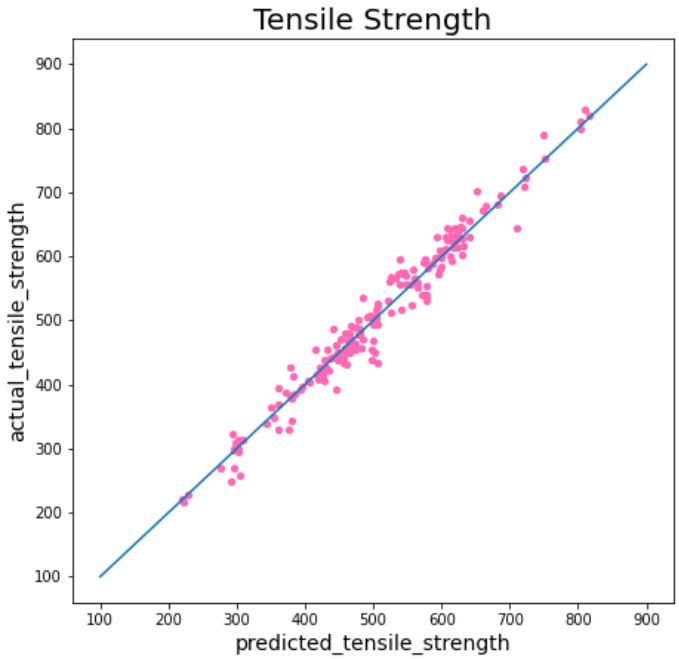
The number of decision trees was decided by brute force. The model was tested for all values of n_estimators ranging from 1 to 500 and the value which provided maximum R^2 score was finally used.

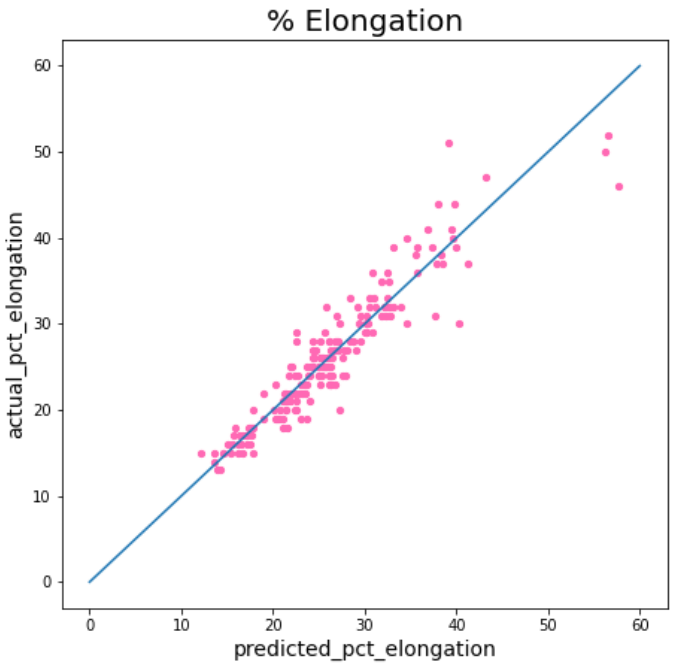
Results obtained with these hyper-parameters –

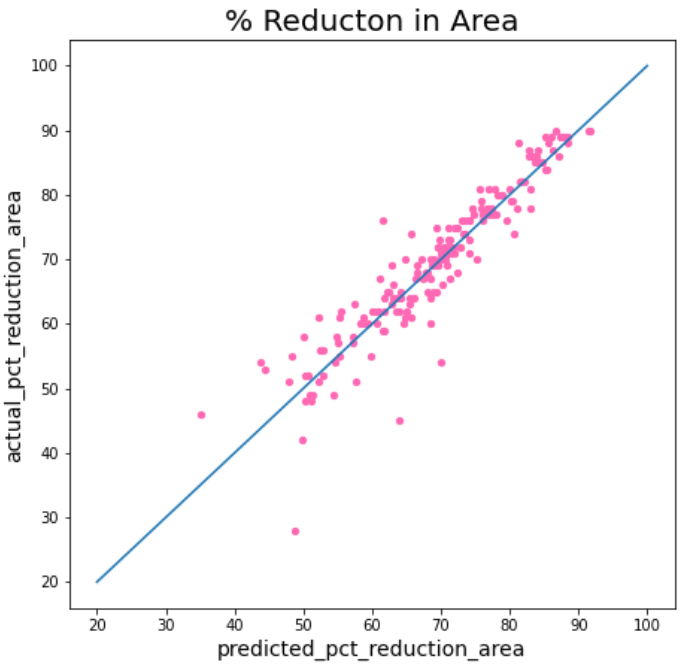
R^2 score = 0.91
Mean squared error = 0.07 (for scaled outputs)
Mean absolute error = 0.16 (for scaled outputs)

Again, the outputs were visualized by plotting them against actual values after scaling up –









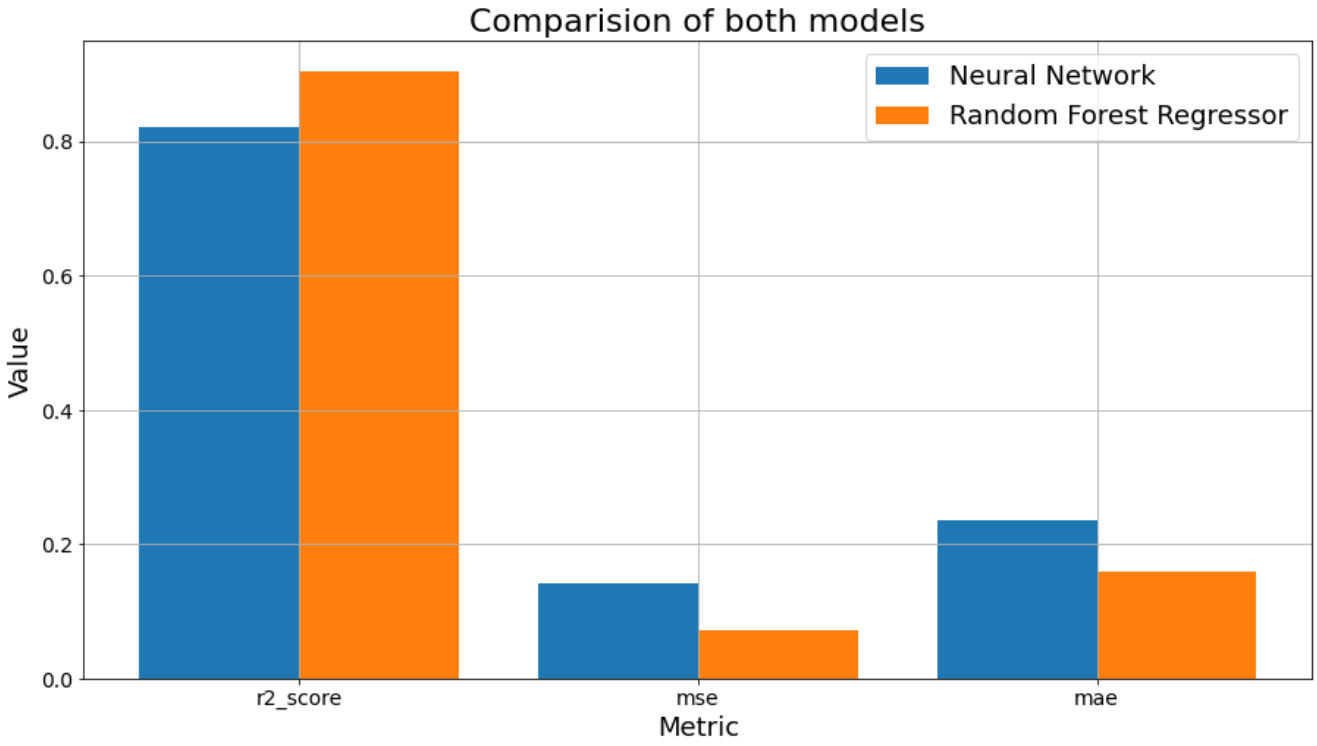
The R^2 score for each mechanical property was calculated –

R^2 score for 0.2% Proof Strength = 0.93
R^2 score for Tensile strength = 0.97
R^2 score for % Elongation = 0.87
R^2 score for % Reduction in Area = 0.87

From the plots, it is obvious that the previous problem occurring due to activation function was eliminated. The model predicted tensile strength with the best accuracy followed by 0.2% Proof strength and both % elongation and % reduction in area were predicted with equal accuracies.

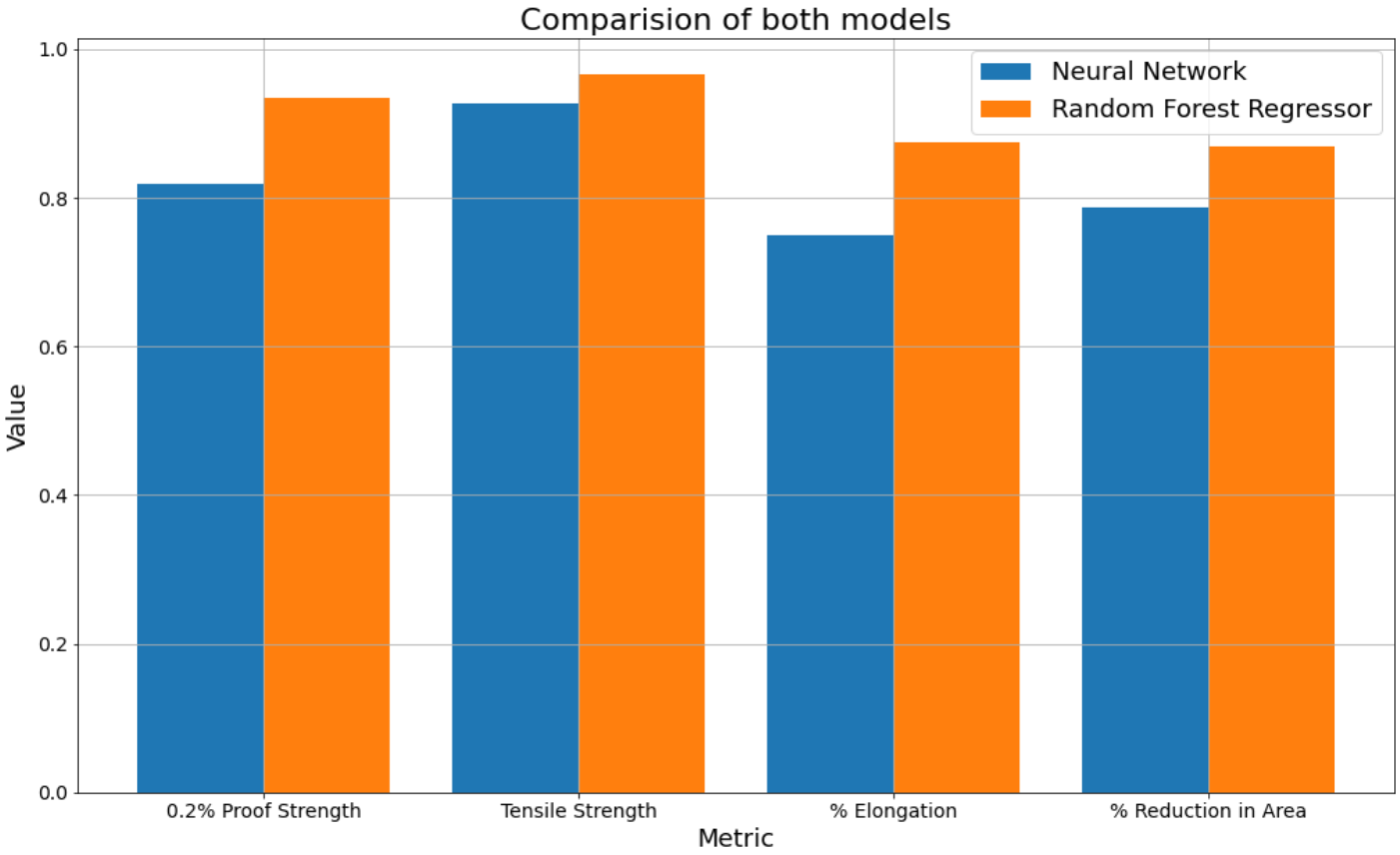
Comparison of both models:

All three metrics including R^2 score, mean squared error and mean absolute error for both models were plotted in the form of multiple bar graphs –



It was observed that the R^2 score was greater for Random Forest which indicates better accuracy and the mean squared error and mean absolute error were lesser for Random Forest which again indicates better accuracy. Thus, the Random Forest Regressor managed to reduce the overall error in the predictions as well as explained almost all the variability in the data.

Another comparison was made across all the mechanical properties by plotting the R^2 scores of both models in the form of multiple bar graphs –



It was observed that the Random Forest Regressor performs better across all mechanical properties. The difference in performance is less for tensile strength but more for remaining three properties.

Conclusion:

The random forest regressor performs better in each category and overall as compared to neural networks. Being computationally cheap to train, manually easer to fine-tune and highly versatile to fit itself on a complex data containing regressions within clusters, this model makes for an ideal choice for prediction of mechanical properties of low-alloy steels with R^2 score of 0.91 which is significantly greater than R^2 score of Neural Network which is 0.82

Links:

Dataset Generator Google Colab

Main Project Notebook Google Colab

Drive link for all project files

NIMS Matnavi database