PROJECT TITLE: PREDICTION OF TURNDOWN CHEMISTRY FOR STEELMAKING VESSEL USING MACHINE LEARNING.

SUBMITTED BY: - SOURAV KUMAR PATHAK
VT no.: - VT20191487
Institute: - NIT JAMSHEDPUR

Under the guidance of: -

Dr. Mrityunjay Kumar Singh

Head, Product and Simulation Department

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Dr. Mrityunjay Kumar Singh for his guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

His fluency in expressing his thoughts and the natural flow with which he explained us the topics was commendable and went a long way in helping me in completing the project. I would like to thank *Shavak Nanavati Training Institute*, who gave me the opportunity to work with the steel giant, *Tata Steel Limited* as a Vocational Trainee. My profound gratitude to my guide and mentor at Tata Steel Limited, Dr. Mrityunjay Kumar Singh, for taking me under his supervision and also for sharing his rich experience with me during the course of the work which made the whole experience both educative and fun for our entire group. I also want to thanks my guide for his immense contribution in this project.

I would like to express my gratitude towards my parents for their kind co-operation and encouragement which help me in completion of this project. I would also like to thanks members of SNTI for their kind co-operation and support. I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

NAME: - **SOURAV KUMAR PATHAK**

NIT JAMSHEDPUR

METALLURGICAL AND MATERIALS ENGINEERING

CONTENTS

1. INTRODUCTION

Basic Oxygen Furnace (BOF) is the most commonly used process for steel making. The liquid iron from the blast furnaces is taken to the BOF shop where pure oxygen is blown into the iron at supersonic speed through a lance positioned in the mouth of the BOF and this removes the carbon and other impurities. **Burnt lime** and certain other '**fluxes**' are added to the BOF to make a liquid slag which will absorb the impurities and so keep them separate from the steel.

The major impurity which must be removed are sulphur and phosphorus. The final sulphur and phosphorus content of steel are the indicators of quality of steel. The endpoint phosphorus is determined by various factors during the steelmaking process.

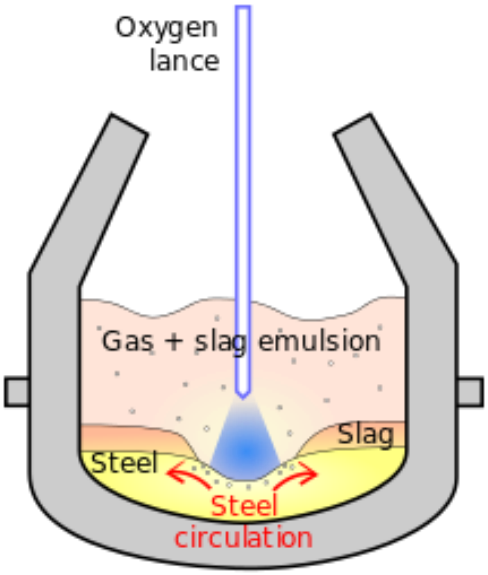


Fig: Basic Oxygen Furnace

2. PROBLEM STATEMENT

We were provided with a dataset of three different converters of LD#2, which included the data of various parameters of the process. Our objective was to develop a model which could predict the final phosphorus of the steel.

3. LINEAR REGRESSION MODEL

Regression analysis is widely used for prediction and forecasting. It is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships.

Many techniques for carrying out regression analysis have been developed. Familiar methods such as linear regression and ordinary least squares regression are parametric, in that the regression function is defined in terms of a finite number of unknown parameters that are estimated from the data.

Multiple linear regression is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

The Formula for Multiple linear regression is-

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

where, for i=n observations:

y_i=dependent variable

x_i=explanatory variables

β_0=y-intercept (constant term)

β_p=slope coefficients for each explanatory variable

ϵ=the model's error term (also known as the residuals)

The multiple regression model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables.
- The independent variables are not too highly correlated with each other.
- y_i observations are selected independently and randomly from the population.
- Residuals should be normally distributed with a mean of 0 and variance σ.

4. DECISION TREE

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs and utility.

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

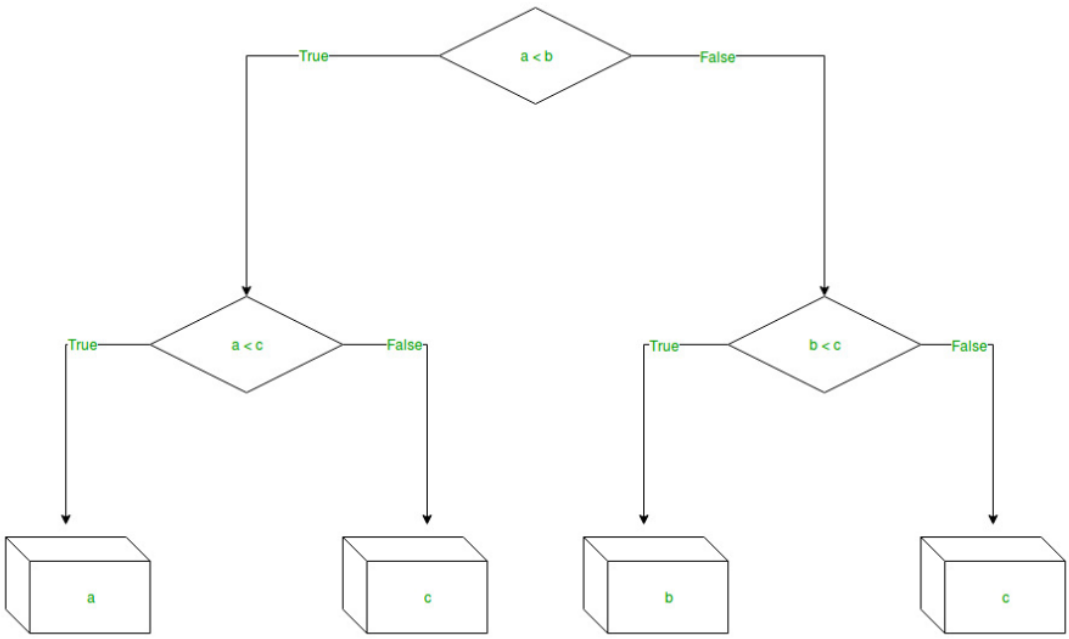


Fig: Decision Tree evaluating smallest of three numbers

5. RANDOM FOREST

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Random forests are used for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

It consists of an arbitrary number of simple trees, which are used to determine the final outcome. In the regression problem, their responses are averaged to obtain an estimate of the dependent variable.

6. K-NEAREST NEIGHBORS

K-nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition.

A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. Another approach uses an inverse distance weighted average of the K nearest neighbors. KNN regression uses the distance functions.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise; however, the compromise is that the distinct boundaries within the feature space are blurred.

7. METALLURGICAL ASPECTS

Removal of phosphorus is a reaction, which plays an important role in combined converter steelmaking process, and the precise control of end‐point phosphorus content during BOF steelmaking process would greatly improve the quality of liquid steel.

The equation of dephosphorization reaction is expressed as follows:

$$5(FeO) = 5[O] + 5[Fe]$$

$$2[P] + 5[O] = (P_2O_5)$$

$$(P_2O_5) + 4(CaO) = (4CaO \cdot P_2O_5)$$

$$2[P] + 5(FeO) + 4(CaO) = (4CaO \cdot P_2O_5) + 5[Fe]$$

$$\lg K = \lg \frac{a_{4CaO \cdot P_2O_5}}{[\%P]^2 \cdot a_{FeO}^5 \cdot a_{CaO}^4} = \frac{40\ 067}{T} - 15.06$$

where, K is chemical equilibrium constant; [%P] is phosphorus content of liquid steel; $a_{4CaO.P2O5}$, a_{CaO} and a_{FeO} are the activity of $4CaO.P_2O_5$, CaO, and FeO, respectively. According to the theories of molecular structure of metallurgical slag, the activity of $4CaO.P_2O_5$ could be substituted by the mole fraction of P_2O_5 (X_{P2O5}).

$$K = \frac{a_{4CaO \cdot P_2O_5}}{[\%P]^2 \cdot a_{FeO}^5 \cdot a_{CaO}^4} = \frac{X_{P_2O_5}}{[\%P]^2 \cdot a_{FeO}^5 \cdot a_{CaO}^4}$$

The above equation can be written as-

$$L_P = \frac{X_{P_2O_5}}{[\%P]^2} = K \cdot a_{FeO}^5 \cdot a_{CaO}^4$$

where, L_P is phosphorus partition ratio, and X_{P2O5} , ½[%P]stand for the mass percent of phosphorus in slag and steel, respectively.

Phosphorus partition ratio is an extremely important index, which is used to judge the capacity of phosphorus removals of the slag.

8. MACHINE LEARNING MODEL

For designing a model following steps were taken-

a) **Data Collection:** We were provided with the dataset of 3 vessels from LD#2 which contained 2290, 2230 and 2372 data for the months of January, February and March respectively. A total of 88 parameters had to be taken into consideration.

b) **Data Preparation:** The data of three months were combined in a single csv file for further processing. Our aim was to predict the 'Final Phosphorus', so it became our dependent variable. A correlation matrix was checked to find the relation among the parameters. The following variables were considered for model making-

   a. Hot metal weight
   b. Scrap weight
   c. Hot metal Silicon
   d. Initial Phosphorus
   e. Slag CaO
   f. Slag SiO_2
   g. Oxygen blown
   h. Lime weight
   i. Dolomite weight
   j. Ore weight
   k. Actual bath Sulphur
   l. Basicity

There were null values in the dataset which needed to be removed for an effective model. First, the null values of 'Final Phosphorus' were deleted along with entire row.
All the parameters were box plotted to find the outliers. Accordingly the outliers were removed from the data.

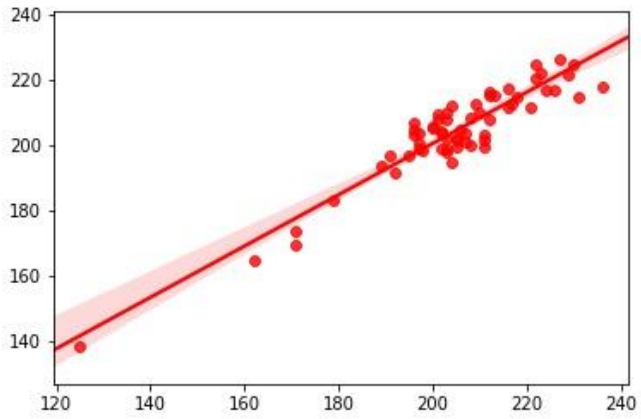
Now the data is prepared for splitting it into training and test data.

c) **Choosing a model:** Different models were trained on the train set and its R-squared and root mean squared error was found out:
    a. Multiple Linear Regression
       R^2 = 0.8667849759655069
    b. Decision Tree
       R^2 = 0.6946761129174339
    c. Random Forest
       R^2 = 0.8126151271322819
    d. K-Nearest Neighbors
       R^2 = 0.6051067131852806

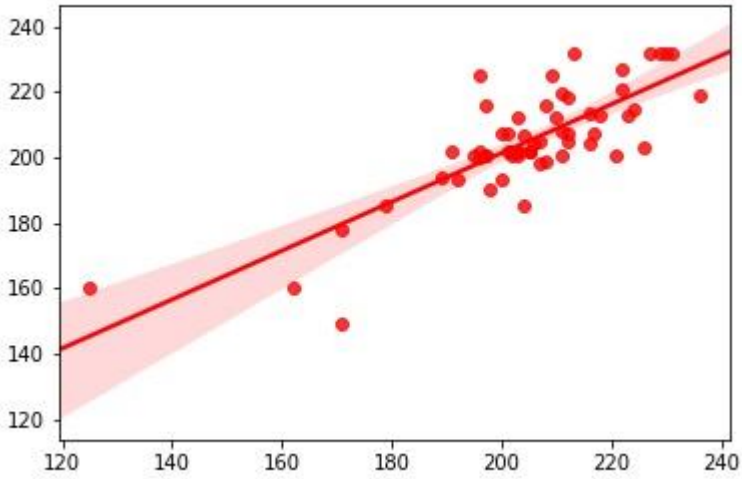
9. RESULT AND CONCLUSION

Since Linear Multiple Regression model has the best fit of 17% so we will considered it as our model. A graph of y_test vs y_pred was plotted for different models-

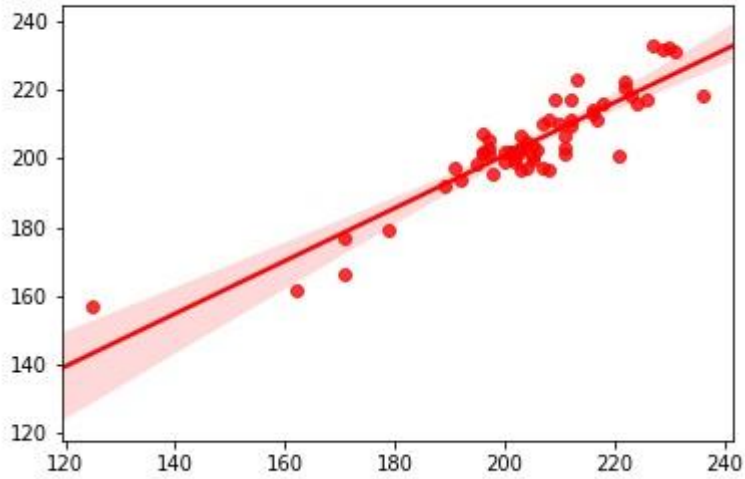
a. Multiple Linear Regression



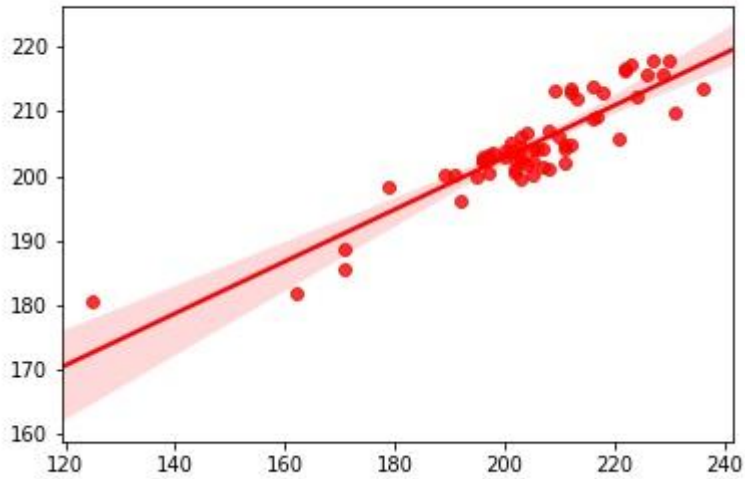
b. Decision Tree



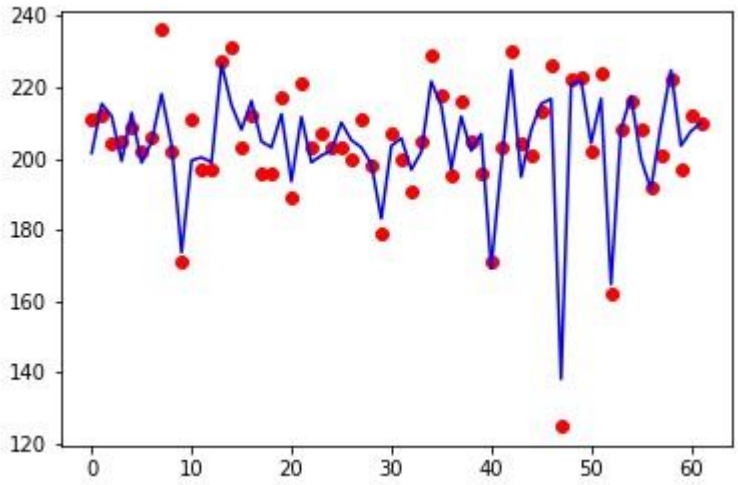
c. Random Forest



d. K-Nearest Neighbors



The best fit is 86.66% for **Multiple linear regression,** which is quite good for industrial data. Hence, This model can be utilized for the prediction of final carbon in steel making vessel.



10. Annexure

```
import pandas as pd
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
%matplotlib notebook
import numpy as np
import seaborn as sns
import warnings
from sklearn.metrics import mean_squared_error
from math import sqrt
warnings.filterwarnings("ignore")
warnings.filterwarnings('ignore',category=DeprecationWarning
)



pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', 100)

df = pd.read_csv("Vessel_data.csv")
df["Basicity"] =
((df["VSL_ACT_SLAG_ANA_1.CAO"])/df["VSL_ACT_SLAG_A
NA_1.SIO2"])
```

df = df[["HM weight", "Scrap weight", "HMSI", "DS_DS_P", "VSL_ACT_SLAG_ANA_1.CAO", "VSL_ACT_SLAG_ANA_1.SIO2", "Oxygen Blown", "Lime weight", "Dolomite weight", "Ore weight", "Actual bath S","Basicity", "Final P"]]

df1 = df.iloc[:2151,:] //for Converter 1

corr_matrix = df1.corr()
corr_matrix

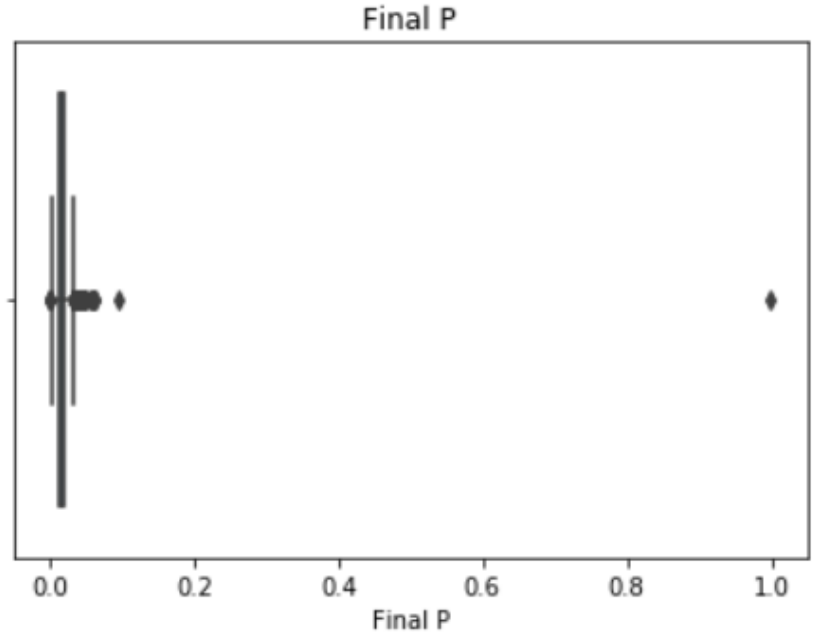
	HM weight	Scrap weight	HMSI	DS_DS_P	VSL_ACT_SLAG_ANA_1.CAO	VSL_ACT_SLAG_ANA_1.SIO2	Oxygen Blown	Lime weight	Dolomite weight
HM weight	1.000000	-0.615863	0.008002	-0.003003	0.047521	0.041415	0.034945	0.165613	0.051073
Scrap weight	-0.615863	1.000000	-0.036012	0.004637	0.025467	-0.078201	0.263029	-0.133305	-0.033782
HMSI	0.008002	-0.036012	1.000000	-0.010821	-0.172378	0.327602	-0.057095	0.741549	-0.012407
DS_DS_P	-0.003003	0.004637	-0.010821	1.000000	0.025304	-0.018685	0.040673	0.063574	0.001354
VSL_ACT_SLAG_ANA_1.CAO	0.047521	0.025467	-0.172378	0.025304	1.000000	0.111607	0.027636	-0.072410	-0.089097
VSL_ACT_SLAG_ANA_1.SIO2	0.041415	-0.078201	0.327602	-0.018685	0.111607	1.000000	-0.114076	0.196021	-0.041670
Oxygen Blown	0.034945	0.263029	-0.057095	0.040673	0.027636	-0.114076	1.000000	-0.044704	0.126290
Lime weight	0.165613	-0.133305	0.741549	0.063574	-0.072410	0.196021	-0.044704	1.000000	-0.085107
Dolomite weight	0.051073	-0.033782	-0.012407	0.001354	-0.089097	-0.041670	0.126290	-0.085107	1.000000
Ore weight	0.431011	-0.525488	0.590530	0.002327	-0.093003	0.289302	-0.234181	0.520282	0.008157
Actual bath S	-0.018993	0.086284	-0.383151	0.025725	0.398292	-0.856204	0.120725	-0.210508	-0.006859
Basicity	-0.019011	0.086294	-0.383148	0.025757	0.398270	-0.856219	0.120720	-0.210506	-0.006857
Final P	0.075864	-0.066003	-0.008993	-0.007764	0.041158	0.138114	0.013599	-0.017030	-0.017465

Scrap weight	HMSI	DS_DS_P	VSL_ACT_SLAG_ANA_1.CAO	VSL_ACT_SLAG_ANA_1.SIO2	Oxygen Blown	Lime weight	Dolomite weight	Ore weight	Actual bath S	Basicity	Final P
0.615863	0.008002	-0.003003	0.047521	0.041415	0.034945	0.165613	0.051073	0.431011	-0.018993	-0.019011	0.075864
1.000000	-0.036012	0.004637	0.025467	-0.078201	0.263029	-0.133305	-0.033782	-0.525488	0.086284	0.086294	-0.066003
0.036012	1.000000	-0.010821	-0.172378	0.327602	-0.057095	0.741549	-0.012407	0.590530	-0.383151	-0.383148	-0.008993
0.004637	-0.010821	1.000000	0.025304	-0.018685	0.040673	0.063574	0.001354	0.002327	0.025725	0.025757	-0.007764
0.025467	-0.172378	0.025304	1.000000	0.111607	0.027636	-0.072410	-0.089097	-0.093003	0.398292	0.398270	0.041158
0.078201	0.327602	-0.018685	0.111607	1.000000	-0.114076	0.196021	-0.041670	0.289302	-0.856204	-0.856219	0.138114
0.263029	-0.057095	0.040673	0.027636	-0.114076	1.000000	-0.044704	0.126290	-0.234181	0.120725	0.120720	0.013599
0.133305	0.741549	0.063574	-0.072410	0.196021	-0.044704	1.000000	-0.085107	0.520282	-0.210508	-0.210506	-0.017030
0.033782	-0.012407	0.001354	-0.089097	-0.041670	0.126290	-0.085107	1.000000	0.008157	-0.006859	-0.006857	-0.017465
0.525488	0.590530	0.002327	-0.093003	0.289302	-0.234181	0.520282	0.008157	1.000000	-0.312377	-0.312379	-0.020232
0.086284	-0.383151	0.025725	0.398292	-0.856204	0.120725	-0.210508	-0.006859	-0.312377	1.000000	1.000000	-0.104847
0.086294	-0.383148	0.025757	0.398270	-0.856219	0.120720	-0.210506	-0.006857	-0.312379	1.000000	1.000000	-0.104871
0.066003	-0.008993	-0.007764	0.041158	0.138114	0.013599	-0.017030	-0.017465	-0.020232	-0.104847	-0.104871	1.000000

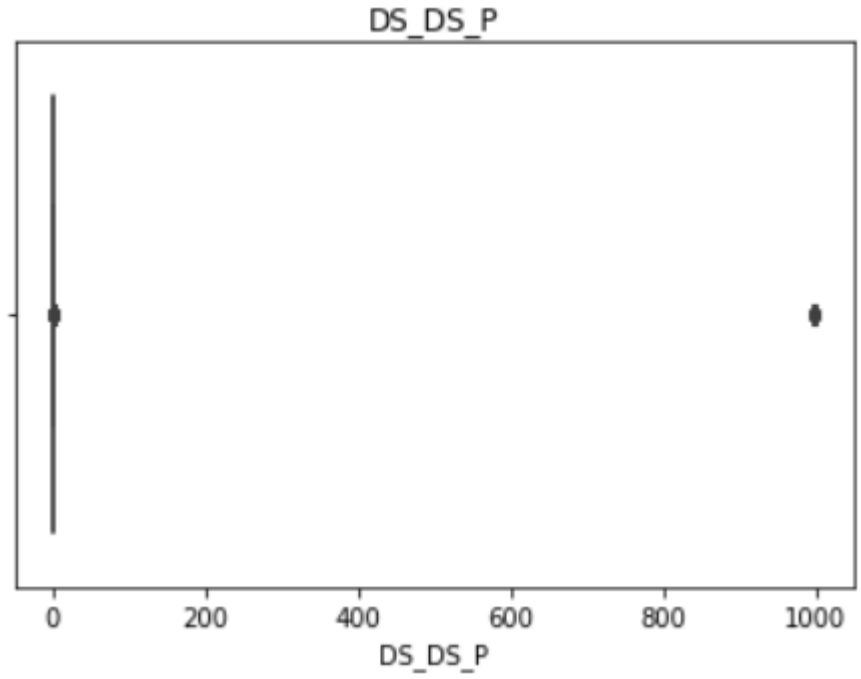
#Dropping null values for Final P

df1 = df1.dropna(subset=["Final P"])

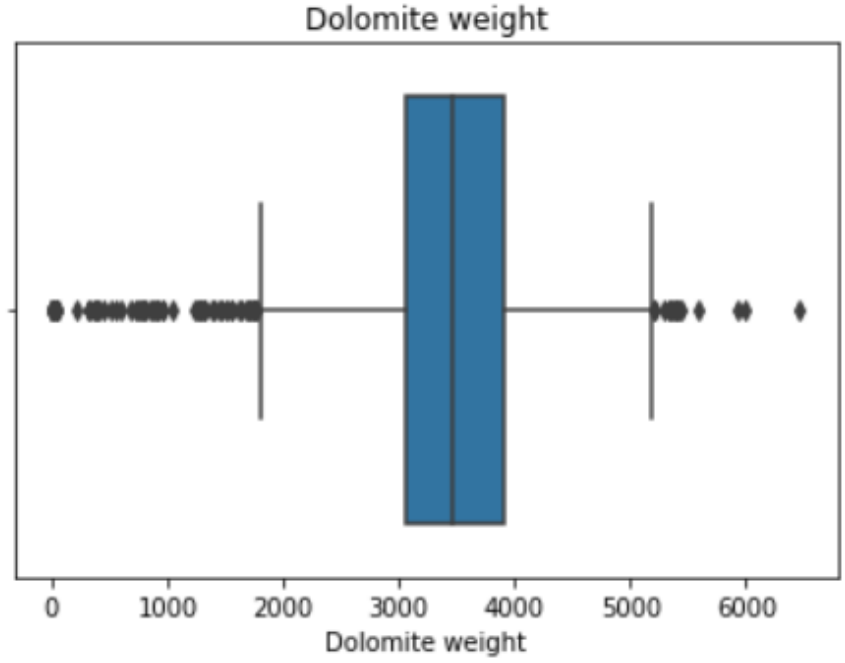
sns.boxplot(x=df1['Final P']).set_title('Final P')



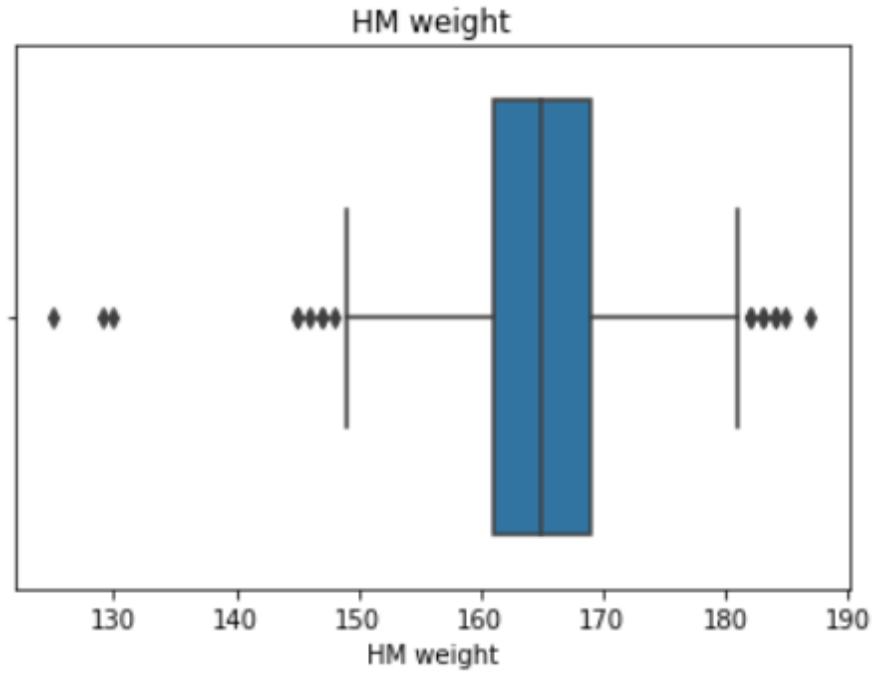
sns.boxplot(x=df1['DS_DS_P']).set_title('DS_DS_P')



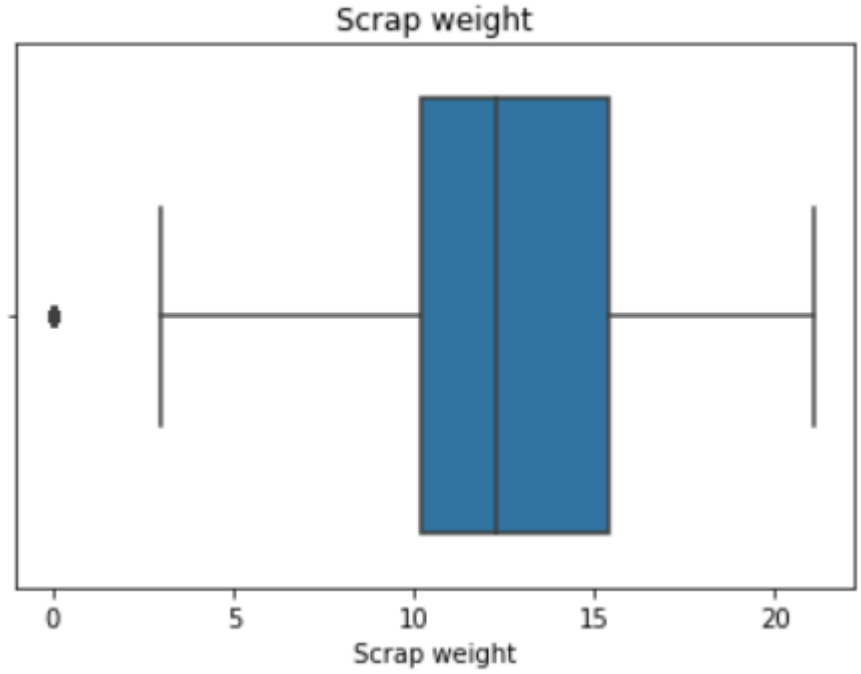
sns.boxplot(x=df1['Dolomite weight']).set_title('Dolomite weight')



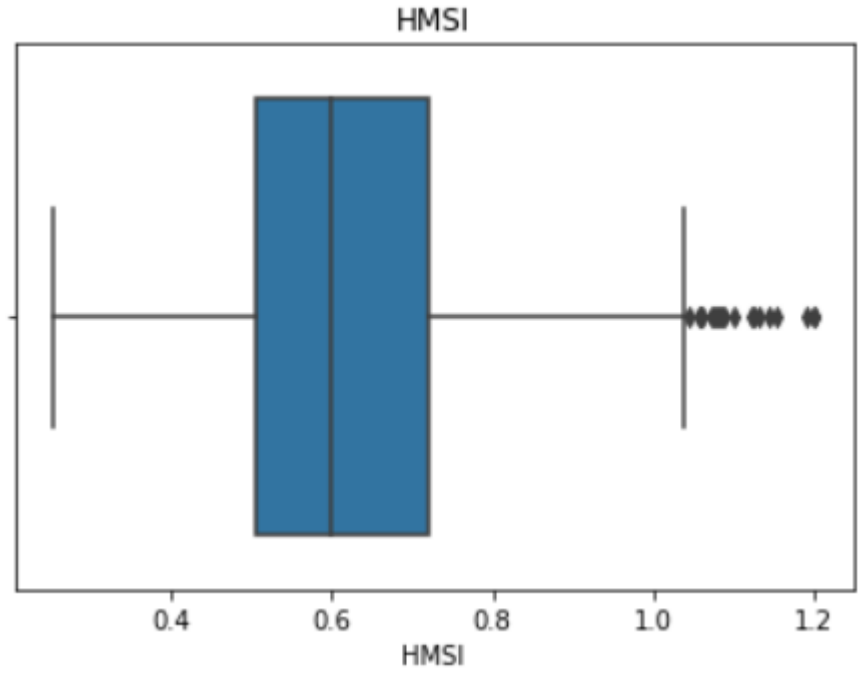
sns.boxplot(x=df1['HM weight']).set_title('HM weight')



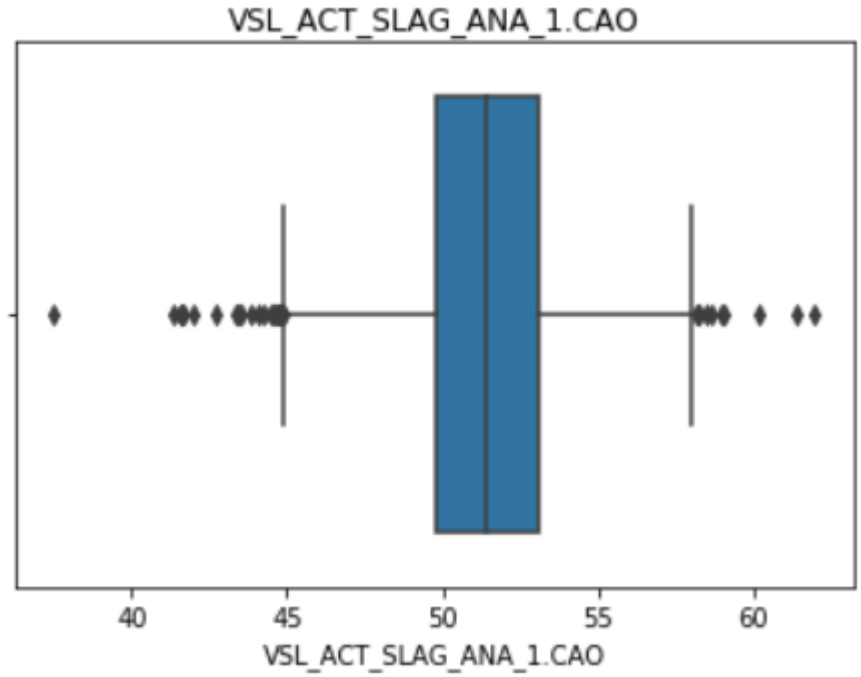
sns.boxplot(x=df1['Scrap weight']).set_title('Scrap weight')



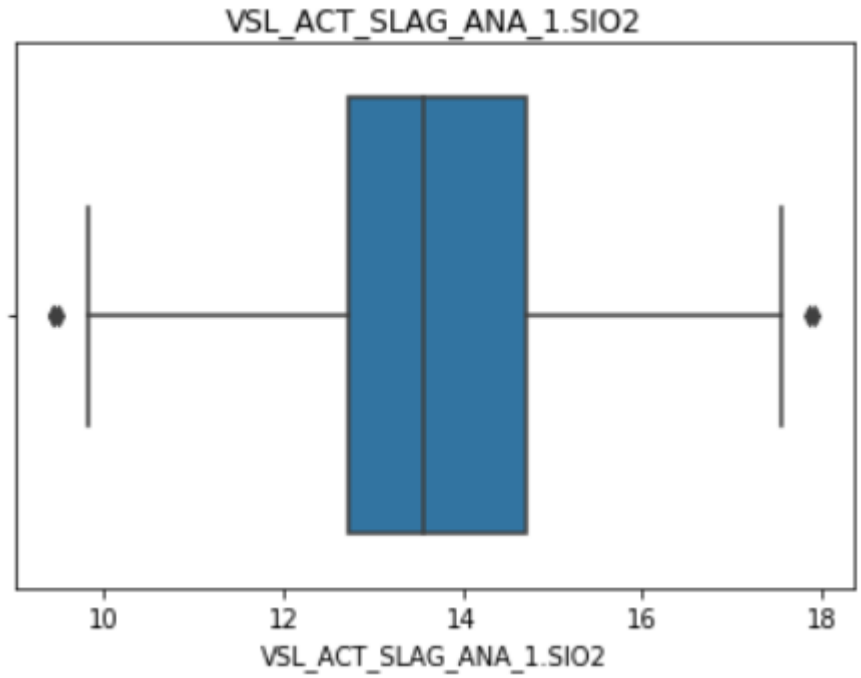
sns.boxplot(x=df1['HMSI']).set_title('HMSI')



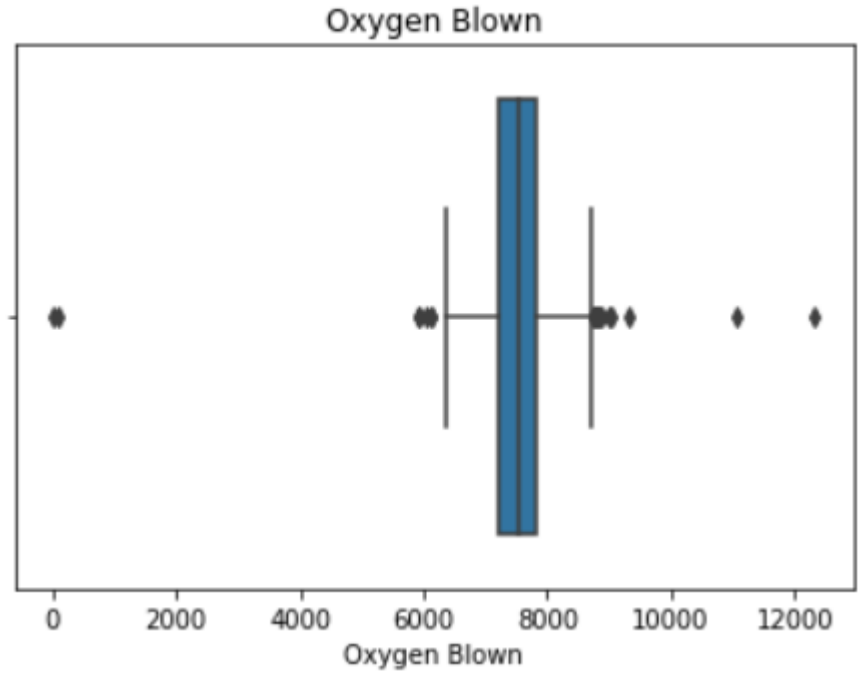
sns.boxplot(x=df1['VSL_ACT_SLAG_ANA_1.CAO']).set_title('VSL_ACT_SLAG_ANA_1.CAO')



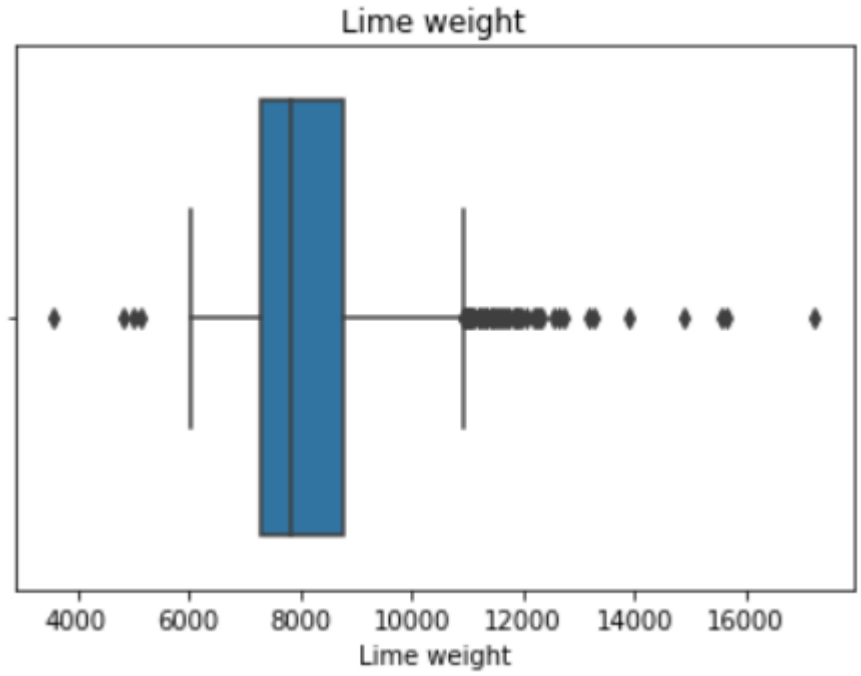
sns.boxplot(x=df1['VSL_ACT_SLAG_ANA_1.SIO2']).set_title('VSL_ACT_SLAG_ANA_1.SIO2')



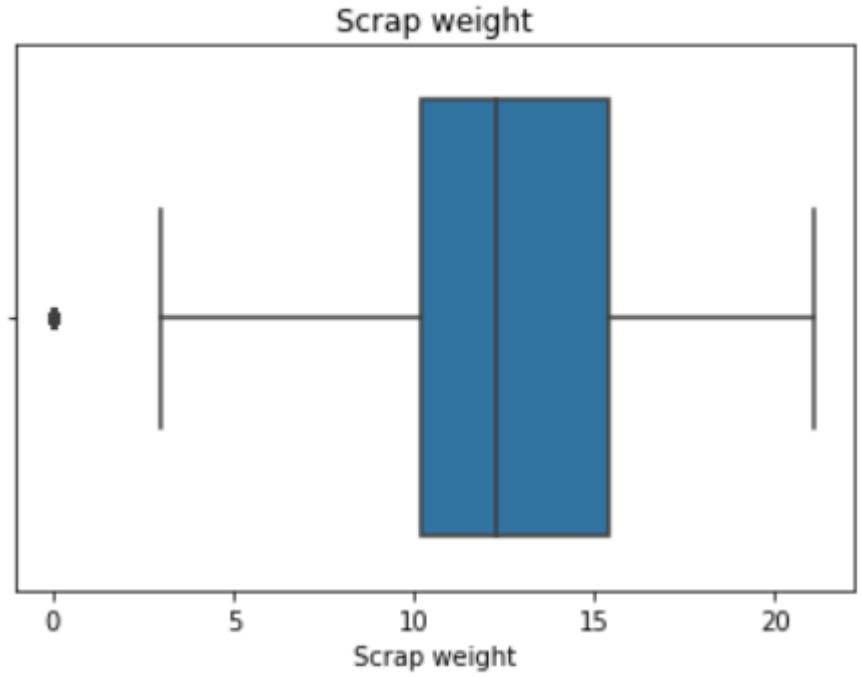
sns.boxplot(x=df1['Oxygen Blown']).set_title('Oxygen Blown')



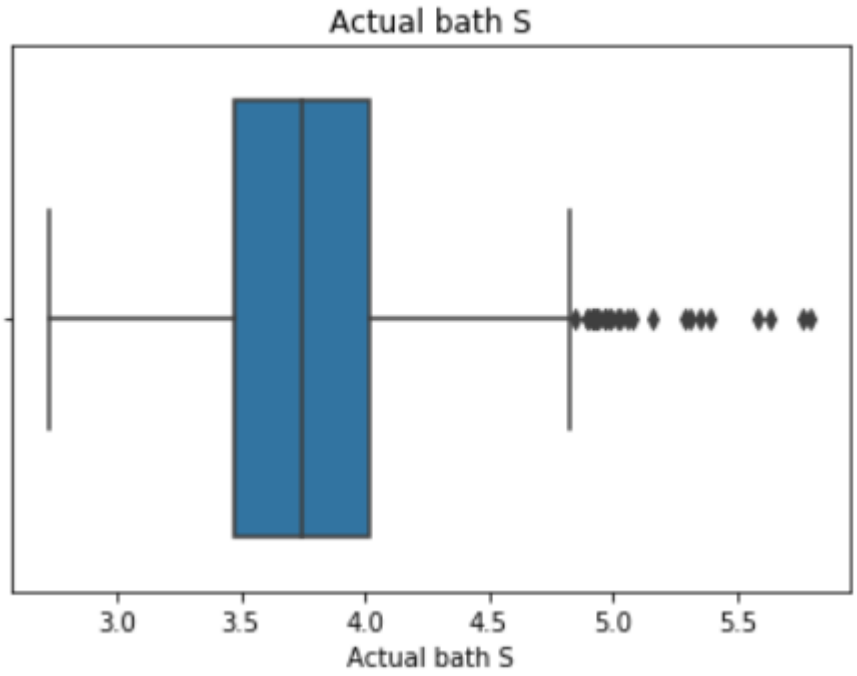
sns.boxplot(x=df1['Lime weight']).set_title('Lime weight')



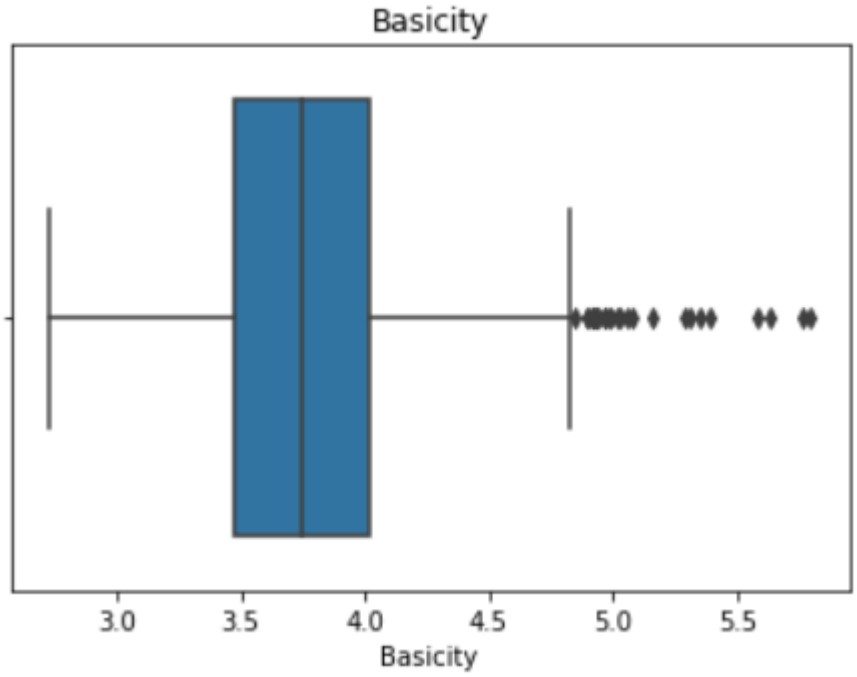
sns.boxplot(x=df1['Scrap weight']).set_title('Scrap weight')



sns.boxplot(x=df1['Actual bath S']).set_title('Actual bath S')



sns.boxplot(x=df1['Basicity']).set_title('Basicity')



Removing Outliers

```
df1 = df1[(df1['Final P']<0.035)]
df1 = df1[(df1['DS_DS_P']>0.1) & (df1['DS_DS_P']<0.25)]
df1 = df1[(df1['Dolomite weight']>2000) & (df1['Dolomite weight']<5500)]
df1 = df1[(df1['HM weight']>135 & (df1['HM weight']<180))]
df1 = df1[(df1['Scrap weight']>0)]
df1 = df1[(df1['HMSI']<1)]
df1 = df1[(df1['VSL_ACT_SLAG_ANA_1.CAO']>43) & (df1['VSL_ACT_SLAG_ANA_1.CAO']<60)]
df1 = df1[(df1['VSL_ACT_SLAG_ANA_1.SIO2']>14) & (df1['VSL_ACT_SLAG_ANA_1.SIO2']<18)]
df1 = df1[(df1['Oxygen Blown']>6000) & (df1['Oxygen Blown']<9000)]
df1 = df1[(df1['Lime weight']>4500) & (df1['Lime weight']<11000)]
df1 = df1[(df1['Scrap weight']>0)]
df1 = df1[(df1['Actual bath S']>3) & (df1['Actual bath S']<4)]
df1 = df1[(df1['Basicity']>3) & (df1['Basicity']<5)]

x = df1.iloc[:, 0:-1].values
y = df1.iloc[:,-1].values

# Initialise the Scaler
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

#Linear Regression

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(x, y)
y_pred = model.predict(x)
```

#R-Square value

coefficient_of_dermination = r2_score(y, y_pred)
print("R-Squared Value: {}"
.format(coefficient_of_dermination))

print(model.intercept_)
print(model.coef_)

```
0.01751612903225977
[ 1.89018726e-03 -6.33043556e-04  2.76730893e-03  3.89060194e-04
  5.18587374e-03 -6.51588437e-03  5.70039883e-04 -1.37205886e-03
 -1.81983521e-04 -2.38596812e-03  6.48764663e-01 -6.57423794e-01]
```

Original vs Predicted

sns.regplot(y, y_pred, color = 'red')
plt.ylim(0, 0.04)
plt.xlim(0, None)

#Decision tree

Splitting the Data set in training and test data

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=0)

from sklearn import tree

modelDT = tree.DecisionTreeRegressor(max_depth=10)

```
modelDT.fit(x_train,y_train)
y_pred = modelDT.predict(x_test)
coefficient_of_dermination = r2_score(y_test, y_pred)
print("R-Squared Value:
{}".format(coefficient_of_dermination))

rms = sqrt(mean_squared_error(y_test, y_pred))
print("RMSE: {}".format(rms))

# Original vs Predicted Graph
sns.regplot(y_test, y_pred, color = 'red')
```

#Random Forest

```
from sklearn.ensemble import RandomForestRegressor

modelRF = RandomForestRegressor(n_estimators = 10)
modelRF.fit(x_train,y_train)
y_pred = modelRF.predict(x_test)
coefficient_of_dermination = r2_score(y_test, y_pred)
print("R-Squared Value:
{}".format(coefficient_of_dermination))

rms = sqrt(mean_squared_error(y_test, y_pred))
print("RMSE: {}".format(rms))

# Original vs Predicted
sns.regplot(y_test, y_pred, color = 'red')
```

#K-nearest neighbour

```
from sklearn.neighbors import KNeighborsRegressor

# Create KNN regressor
knn = KNeighborsRegressor(n_neighbors = 50)
# Fit the classifier to the data
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)
coefficient_of_dermination = r2_score(y_test, y_pred)
print("R-Squared                                    Value:
{}".format(coefficient_of_dermination))

rms = sqrt(mean_squared_error(y_test, y_pred))
print("RMSE: {}".format(rms))
# Original vs Predicted
sns.regplot(y_test, y_pred, color = 'red')
```