**EDA for Data-2 Dataset**

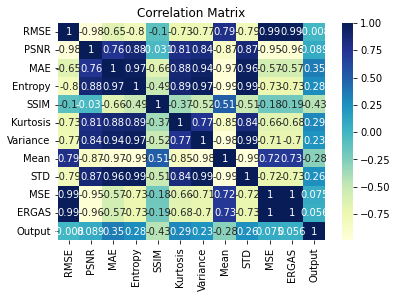
# Correlation Matrix

corr = data.corr()

sns.heatmap(corr, annot=True, cmap="YlGnBu");

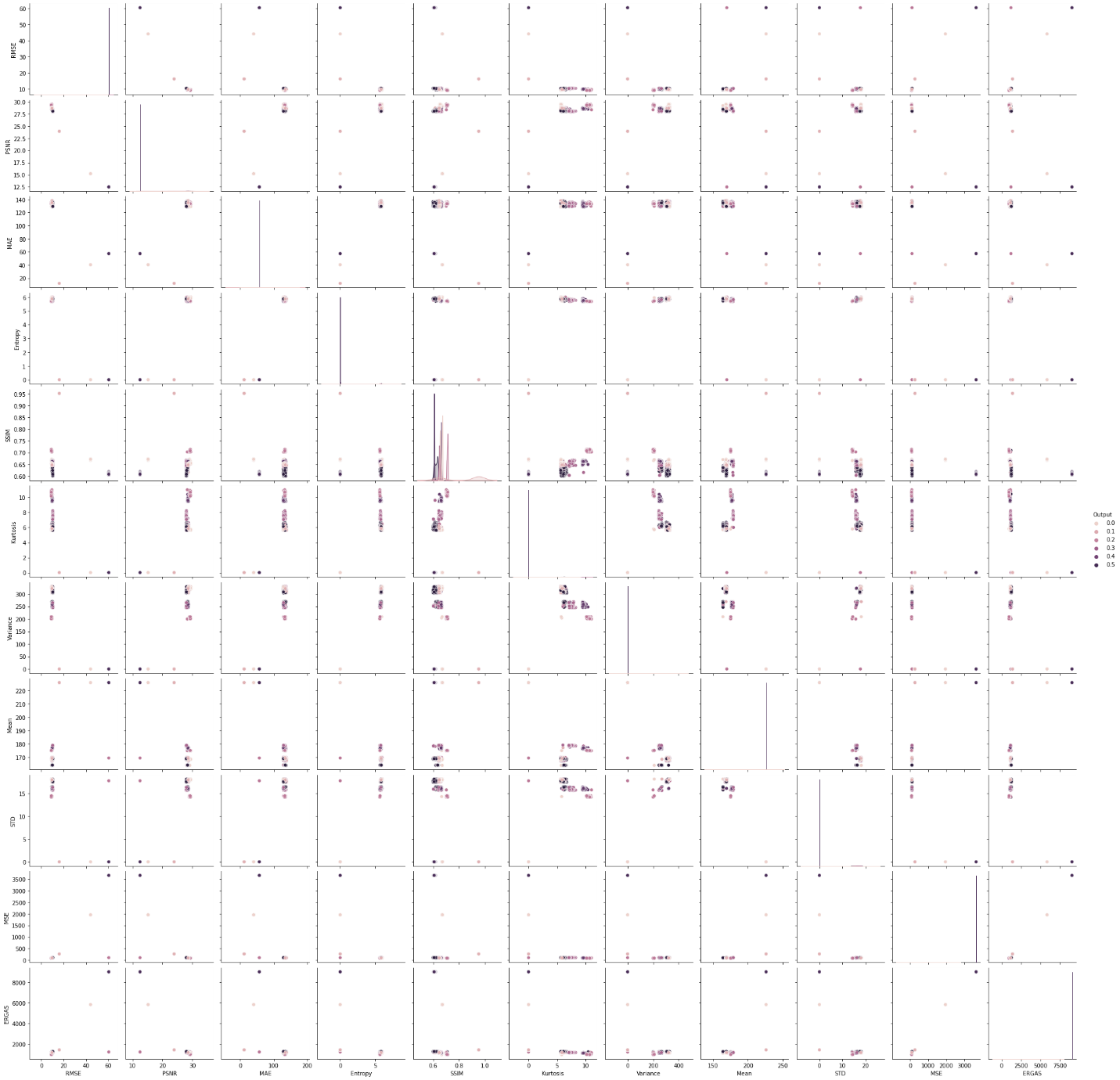
plt.title("Correlation Matrix");

plt.show()

****

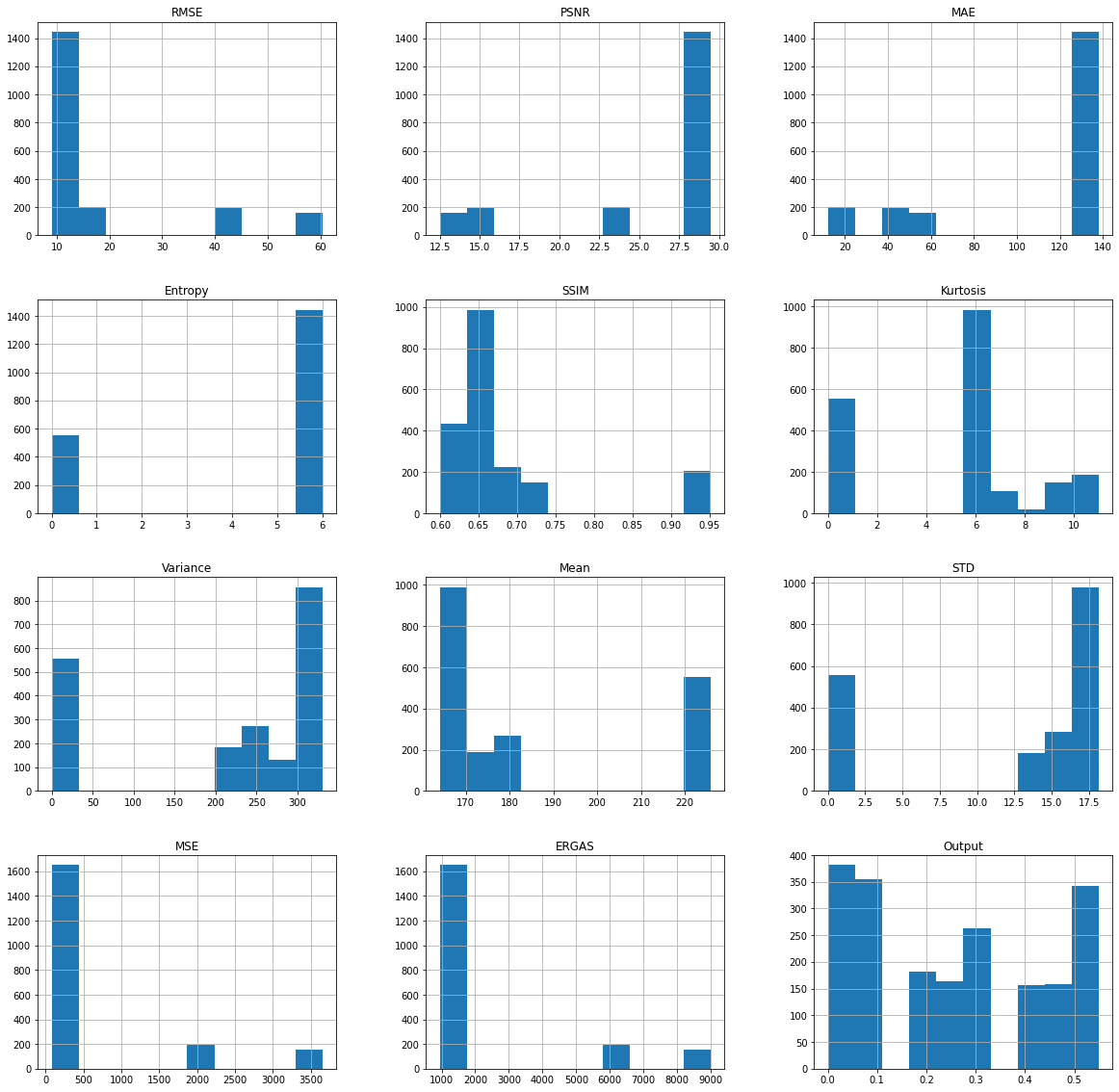
sns.pairplot(data,hue='Output')

<seaborn.axisgrid.PairGrid at 0x7f97db212510>

****

# histogram of data

data.hist(figsize=(20,20));

****

**Machine Learning Algorithms**

import numpy as np

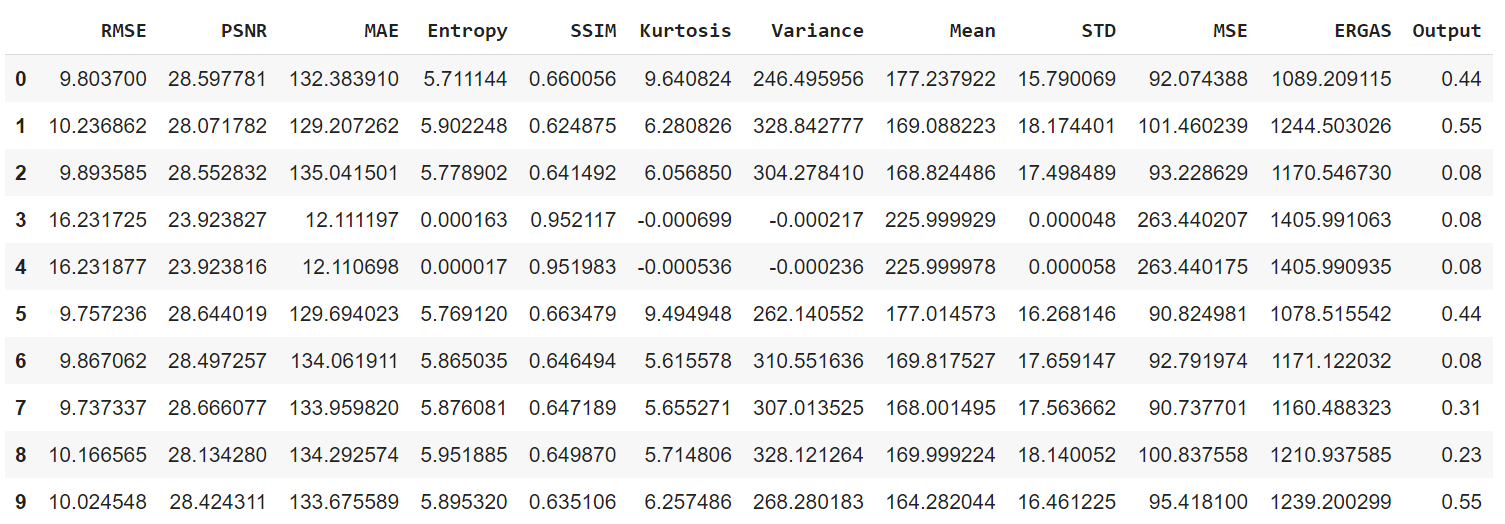
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

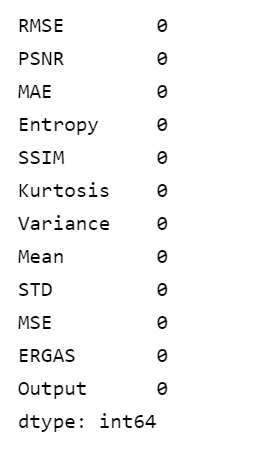
org\_data = pd.read\_excel('Data II.xlsx')

org\_data.head(10)



**Checking for NULL values**

org\_data.isna().sum()

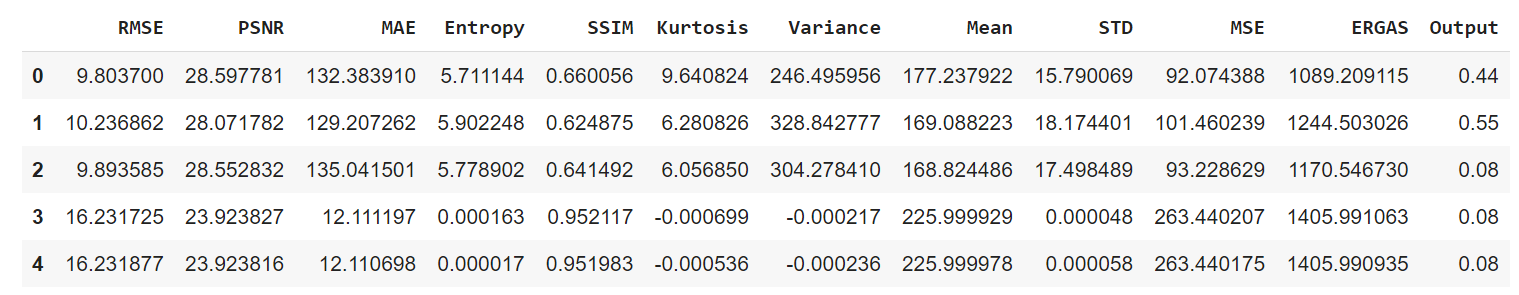
****

data = org\_data

# taking all numerical values

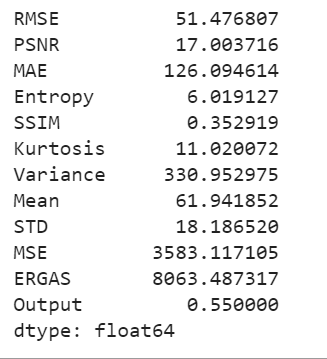
ndata = data.select\_dtypes(include = np.number)

ndata.head()

****

# getting range

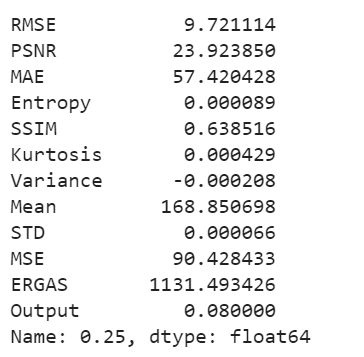
ndata.max() - ndata.min()



# first Quartile

Q1 = ndata.quantile(0.25)

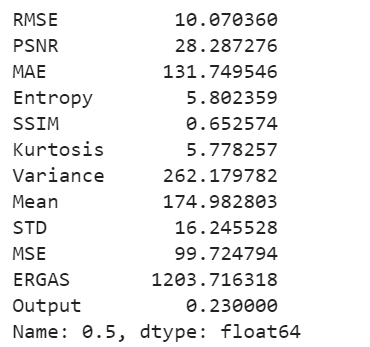
Q1



# second Quartile

Q2 = ndata.quantile(0.50)

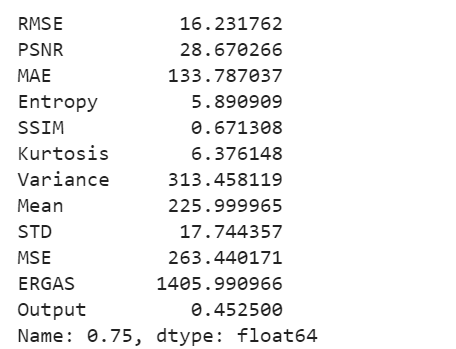
Q2



# third Quartile

Q3 = ndata.quantile(0.75)

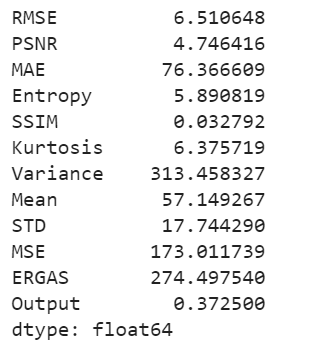
Q3



# Inter Quartile Range

IQR = Q3 - Q1

IQR



# removing outliers

final\_data = ndata[~((ndata < (Q1 - 1.5 \* IQR)) |(ndata > (Q3 + 1.5 \* IQR))).any(axis=1)]

final\_data.shape



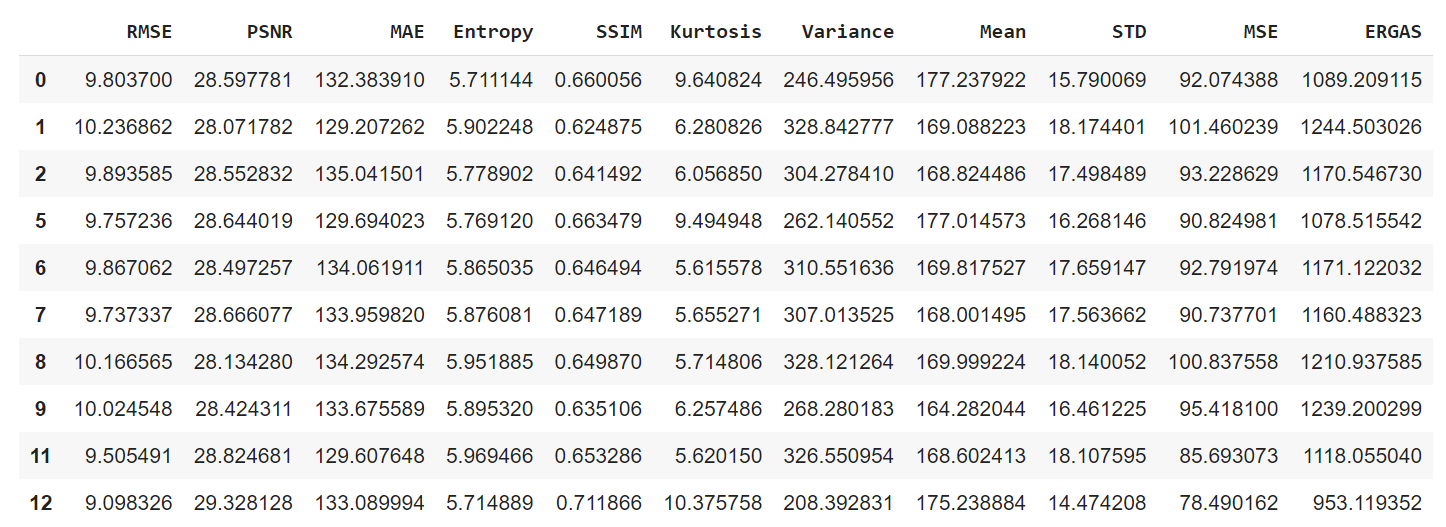
**Machine Learning Algorithms**

# site and pop are correlated features

X = final\_data.drop(['Output'], axis=1)

y = final\_data['Output']

X



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.linear\_model import LinearRegression

from sklearn.kernel\_ridge import KernelRidge

from xgboost.sklearn import XGBRegressor

from sklearn.linear\_model import BayesianRidge

model1 = LinearRegression()

model2 = KernelRidge()

model3 = XGBRegressor()

model4 = BayesianRidge()

from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error

1. **LinearRegression**

model1.fit(X\_train,y\_train)

# precting on training

pred = model1.predict(X\_train)

rmse = (np.sqrt(mean\_squared\_error(y\_train,pred)))

r2 = r2\_score(y\_train, pred)

print("The model performance for training set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

# predicting on test

pred2 = model1.predict(X\_test)

rmse = (np.sqrt(mean\_squared\_error(y\_test, pred2)))

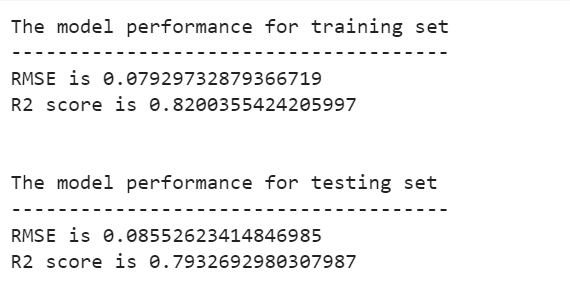
r2 = r2\_score(y\_test, pred2)

print("The model performance for testing set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))



1. **KernelRidge**

model2.fit(X\_train,y\_train)

# precting on training

pred3 = model2.predict(X\_train)

rmse = (np.sqrt(mean\_squared\_error(y\_train,pred3)))

r2 = r2\_score(y\_train, pred)

print("The model performance for training set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

# predicting on test

pred4 = model2.predict(X\_test)

rmse = (np.sqrt(mean\_squared\_error(y\_test, pred4)))

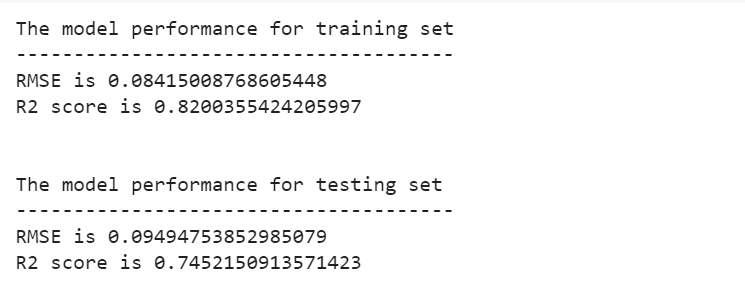
r2 = r2\_score(y\_test, pred4)

print("The model performance for testing set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))



1. **XGBRegressor**

model3.fit(X\_train,y\_train)

# precting on training

pred5 = model3.predict(X\_train)

rmse = (np.sqrt(mean\_squared\_error(y\_train,pred5)))

r2 = r2\_score(y\_train, pred5)

print("The model performance for training set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

# predicting on test

pred6 = model3.predict(X\_test)

rmse = (np.sqrt(mean\_squared\_error(y\_test, pred6)))

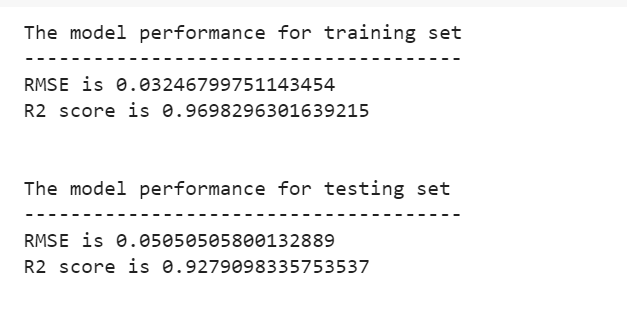
r2 = r2\_score(y\_test, pred6)

print("The model performance for testing set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))



1. **BayesianRidge**

model4.fit(X\_train,y\_train)

# precting on training

pred7 = model4.predict(X\_train)

rmse = (np.sqrt(mean\_squared\_error(y\_train,pred7)))

r2 = r2\_score(y\_train, pred7)

print("The model performance for training set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

# predicting on test

pred8 = model4.predict(X\_test)

rmse = (np.sqrt(mean\_squared\_error(y\_test, pred8)))

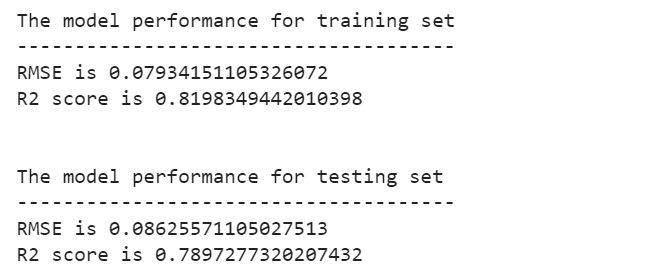
r2 = r2\_score(y\_test, pred8)

print("The model performance for testing set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))



**Comparison:**

We have done EDA of the given dataset and applied the preprocessing, training-testing and following 4 algorithms are used:

1. LinearRegression
2. KernelRidge
3. XGBRegressor
4. BayesianRidge

We can conclude that XGBRegressor has the highest R2 score means there is a high correlation between the attributes and XGBRegressor has the lowest RMSE value means it is the most accurate model. So overall the XGBRegressor model is best among the selected four models.