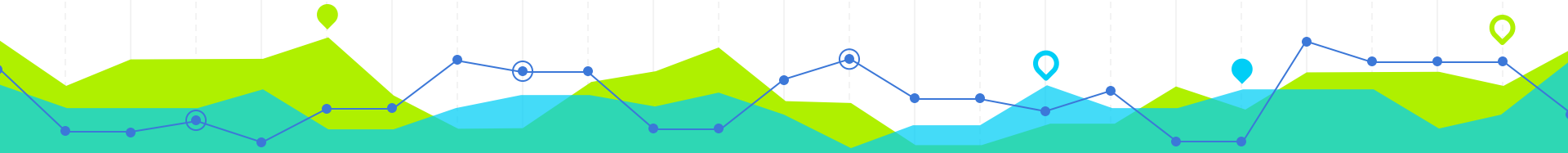
Data Analysis on

SUSTAINABLE DEVELOPMENT GOALS



# MACHINE LEARNING MODEL

Group -09|IIT Patna

# Analysis of SDG index report 2020-21 of Indian States and UTs published by NITI Aayog, India

# GROUP MEMBERS

(Group-9)

- Deepanshu Chaudhary (2001ME21)
- Nitesh Srivastava (2001ME42)
- Prateek Kumar (2001ME48)
- Shresth Verma (2001ME71)
- Ankur Singh (2001ME84)
- Athul Krishna K (2001ME85)

# INTRODUCTION

In 2015, the United Nations General Assembly adopted the 2030 Agenda for Sustainable Development, a plan of action for people, planet, prosperity and peace. It compiles a list of 17 Sustainable Development Goals and 169 targets that several countries including India have pledged to comply with.

NITI Aayog, India, published the SDG scores for 2020-21, for each of the states and union territories in India, and along with the overall score for India. We have analyzed the SDG scores for 2020-21 given by NITI aayog and tried to analyze the performances of different states.

# RECAP

In the first part of the project we did the data analyzation by creating various graphs and tables. Number of processes were involved in the analyzation such as

- Data Collection
- Data Processing and cleaning
- Visual analysis(graphs)
- Hypothesis Testing

# Data Collection

Data was collected from Kaggle which contained 18 csv files out of which 16 were SDG files and 2 were of composite score and a final report pdf file.
Reference links are given below-:

Source :
https://www.kaggle.com/chandramanaha/indian-states-sdg-index-202021
NITI Aayog :
https://sdgindiaindex.niti.gov.in/#/
Rankings :
https://sdgindiaindex.niti.gov.in/#/ranking

# Data Processing and Cleaning

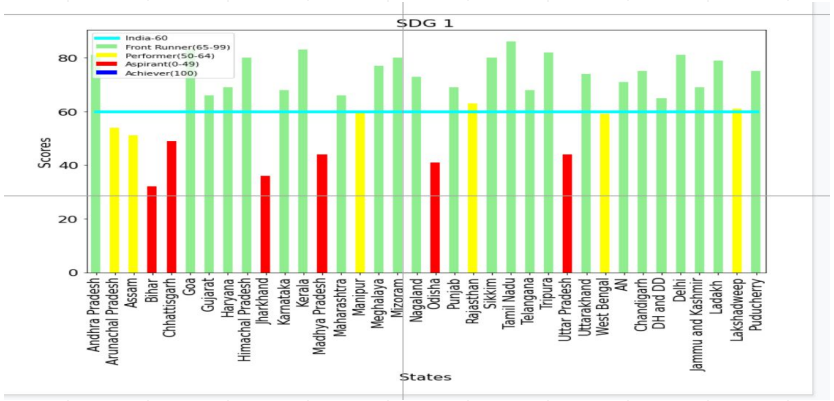
We used python jupyter notebook for all the data analysis and processing. Libraries such as pandas for reading csv files to read data and for dataframe, seaborn, matplotlib, etc were used, and the following order was followed-:

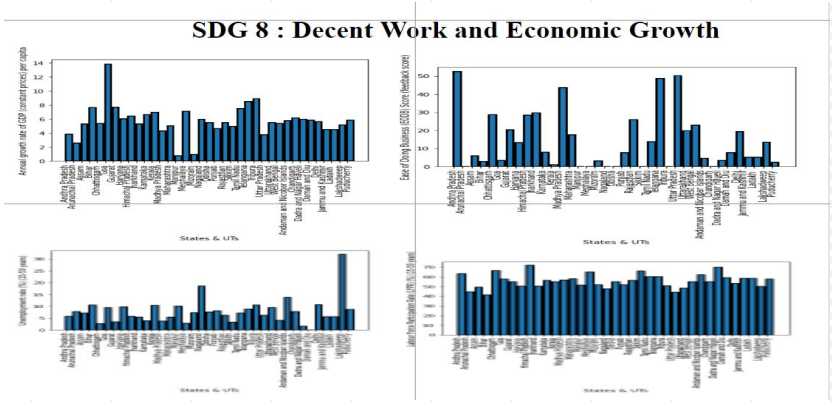
1) Reading of csv files using pandas
2) Checking the missing values in data
3) Finally rejecting the null values in the data

# Visual Analysis

For the data analysis of our dataset we used various modules and libraries available in python such as seaborn, matplotlib and various others.

We plot various graphs related to each field for every state in every SDGs and in the end plot graph for overall values for each SDGs



# Hypothesis Testing

In the end we did the hypothesis testing of our data. We did hypothesis testing to find any relation between any two SDGs. First we checked whether the dataset can be normalized or not. We checked it using using the Ben-Shapiro test. Then hypothesis testing was applied on those datasets which passed the Ben-Shapiro test.

We used t-test method. The t-test performed 2-tailed t-test on the the dataset to get p value which tells whether the two SDGs can be related or not.

# CONTENT

OVERVIEW

PREDICTION MODEL

CONCLUSION

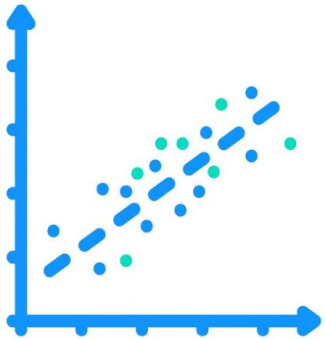
# OVERVIEW

- Our Machine Learning model predicts SDG scores of States and Union Territories of the country.
- For prediction of a given SDG, we are using two datasets: csv file of that particular SDG and a composite score csv.
- After ensuring that our data is thoroughly cleaned, we proceed with our prediction task.
- We are splitting our SDG dataset into two part: One for training purpose and other for prediction task. For splitting, we have used train_test_split() method. In our model, 70% data is used for training while 30% data would be predicted.
- We are using the Linear Regression algorithm for the training of model.

# Best Fitted Model

## Linear Regression

- Since our dataset is quite small, complex data analysis techniques such as K-nearest neighbours(KNN), Decision making tree were too broad to implement. Hence Linear Regression was the only major option.
- Linear regression performs exceptionally well for linearly separable data.
- Linear regression handles overfitting pretty well using dimensionality reduction techniques, regularization, and cross-validation.



Linear Regression in Machine Learning

# PREDICTION MODEL

# Applying model on SDG 1 (No Poverty):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

```
In [11]: corr = df2.corr()
         corr.style.background_gradient(cmap='coolwarm')
```

Out[11]:

| | SNO | Percentage of beneficiaries covered under National Food Security Act (NFSA),2013 | Percentage of children under five years who are underweight | Percentage of children under five years who are stunted | Percentage of pregnant women aged 15-49 years who are anaemic | Percentage of adolescents aged 10–19 years who are anaemic | Rice and wheat produced annually per unit area (Kg/Ha) | Gross Value Added (constant prices) in agriculture per worker (in Lakhs/worker) |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.315503 | -0.348363 | -0.424314 | -0.029594 | -0.009438 | 0.188724 | 0.109362 |
| Percentage of beneficiaries covered under National Food Security Act (NFSA),2013 | -0.315503 | 1.000000 | -0.277906 | -0.048639 | -0.101560 | -0.138214 | 0.028768 | 0.140698 |
| Percentage of children under five years who are underweight | -0.348363 | -0.277906 | 1.000000 | 0.767125 | 0.768994 | 0.561821 | 0.187514 | -0.298588 |
| Percentage of children under five years who are stunted | -0.424314 | -0.048639 | 0.767125 | 1.000000 | 0.611733 | 0.485450 | -0.035403 | -0.411475 |
| Percentage of pregnant women aged 15-49 years who are anaemic | -0.029594 | -0.101560 | 0.768994 | 0.611733 | 1.000000 | 0.617315 | 0.138108 | -0.329587 |
| Percentage of adolescents aged 10–19 years who are anaemic | -0.009438 | -0.138214 | 0.561821 | 0.485450 | 0.617315 | 1.000000 | 0.155502 | -0.167942 |
| Rice and wheat produced annually per unit area (Kg/Ha) | 0.188724 | 0.028768 | 0.187514 | -0.035403 | 0.138108 | 0.155502 | 1.000000 | 0.569492 |
| Gross Value Added (constant prices) in agriculture per worker (in Lakhs/worker) | 0.109362 | 0.140698 | -0.298588 | -0.411475 | -0.329587 | -0.167942 | 0.569492 | 1.000000 |

# TESTING

- Following are the libraries we used:

```
In [30]: from sklearn.linear_model import LinearRegression
         from sklearn.model_selection import train_test_split
         from sklearn.tree import DecisionTreeRegressor
         from yellowbrick.regressor import PredictionError
```

- Using value of data collected for indicators under a particular SDG for each state, the SDG score of a state for that SDG is calculated.
- We have made a model that predicts the SDG score of the state using linear regression algorithm.
- We have made use of 70% of the data we have and predicted the SDG score of remaining 30%.
- As there were 37 States and Union territories in total, the SDG score of 12 states have been predicted.

- The code we used for the testing is the following:

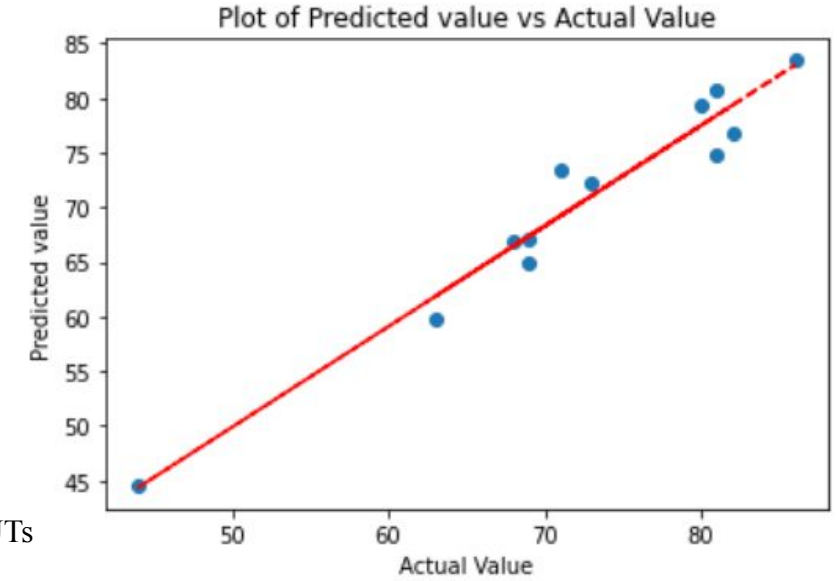
```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 1.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 1']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual Value | Predicted Value |
|---|---|---|
| 13 | 80 | 79.424966 |
| 14 | 69 | 64.949675 |
| 29 | 63 | 59.724951 |
| 31 | 86 | 83.461347 |
| 16 | 68 | 66.893621 |
| 9 | 81 | 80.727559 |
| 33 | 82 | 76.869077 |
| 0 | 71 | 73.509026 |
| 34 | 44 | 44.632978 |
| 25 | 73 | 72.236309 |
| 12 | 69 | 67.136603 |
| 1 | 81 | 74.836821 |

Note: The numbers in 1st column represent SL No of States/UTs



## ACCURACY

- The **accuracy** obtained for SDG 1 is **92.2%**

```
print("\nAccuracy of model for SDG 1")
print(regr.score(X_test, y_test))
```

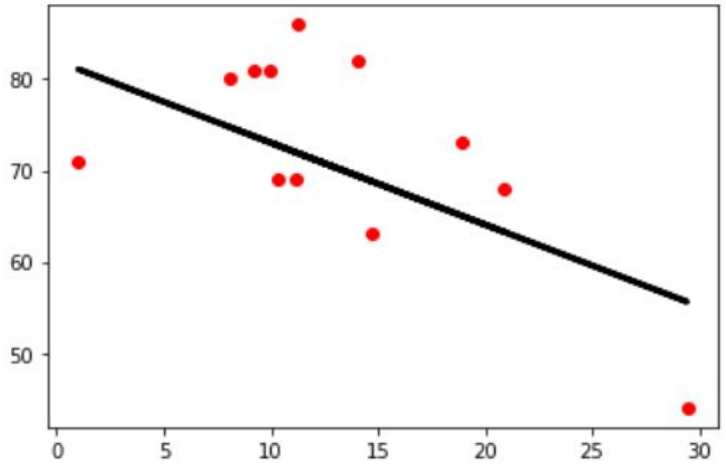
```
Accuracy of model for SDG 1
0.9224698177412547
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 2 (Zero Hunger):

## CORRELATION MATRIX:

- The correlation matrix for SDG 2 is :

| | SNO | Percentage of beneficiaries covered under National Food Security Act (NFSA),2013 | Percentage of children under five years who are underweight | Percentage of children under five years who are stunted | Percentage of pregnant women aged 15-49 years who are anaemic | Percentage of adolescents aged 10–19 years who are anaemic | Rice and wheat produced annually per unit area (Kg/Ha) | Gross Value Added (constant prices) in agriculture per worker (in Lakhs/worker) |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.315503 | -0.348363 | -0.424314 | -0.029594 | -0.009438 | 0.188724 | 0.109362 |
| Percentage of beneficiaries covered under National Food Security Act (NFSA),2013 | -0.315503 | 1.000000 | -0.277906 | -0.048639 | -0.101560 | -0.138214 | 0.028768 | 0.140698 |
| Percentage of children under five years who are underweight | -0.348363 | -0.277906 | 1.000000 | 0.767125 | 0.768994 | 0.561821 | 0.187514 | -0.298588 |
| Percentage of children under five years who are stunted | -0.424314 | -0.048639 | 0.767125 | 1.000000 | 0.611733 | 0.485450 | -0.035403 | -0.411475 |
| Percentage of pregnant women aged 15-49 years who are anaemic | -0.029594 | -0.101560 | 0.768994 | 0.611733 | 1.000000 | 0.617315 | 0.138108 | -0.329587 |
| Percentage of adolescents aged 10–19 years who are anaemic | -0.009438 | -0.138214 | 0.561821 | 0.485450 | 0.617315 | 1.000000 | 0.155502 | -0.167942 |
| Rice and wheat produced annually per unit area (Kg/Ha) | 0.188724 | 0.028768 | 0.187514 | -0.035403 | 0.138108 | 0.155502 | 1.000000 | 0.569492 |
| Gross Value Added (constant prices) in agriculture per worker (in Lakhs/worker) | 0.109362 | 0.140698 | -0.298588 | -0.411475 | -0.329587 | -0.167942 | 0.569492 | 1.000000 |

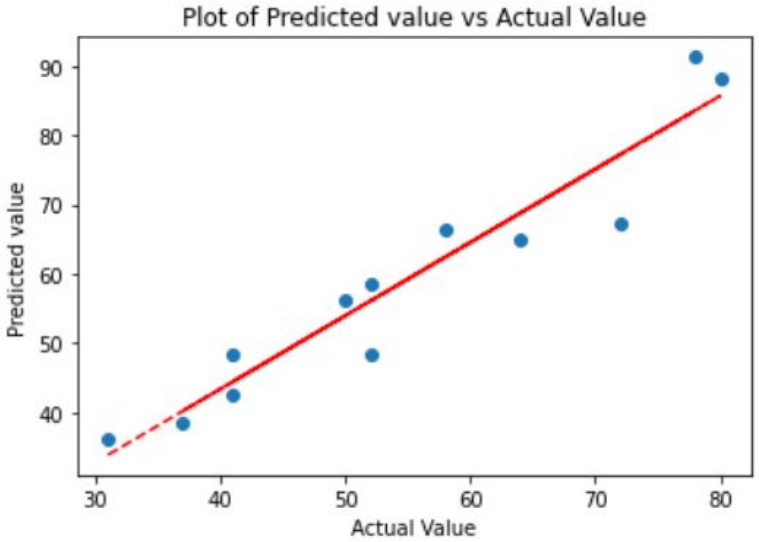
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 2.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 2']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual Value | Predicted Value |
|---|---|---|
| 4 | 31 | 36.135886 |
| 24 | 72 | 67.374102 |
| 13 | 52 | 48.265083 |
| 1 | 52 | 58.521332 |
| 10 | 78 | 91.326347 |
| 12 | 58 | 66.482407 |
| 23 | 37 | 38.533657 |
| 32 | 50 | 56.327217 |
| 3 | 41 | 42.567523 |
| 34 | 41 | 48.241489 |
| 17 | 80 | 88.055072 |
| 25 | 64 | 64.953072 |




Note: The numbers in 1st column represent SL No of States/UTs

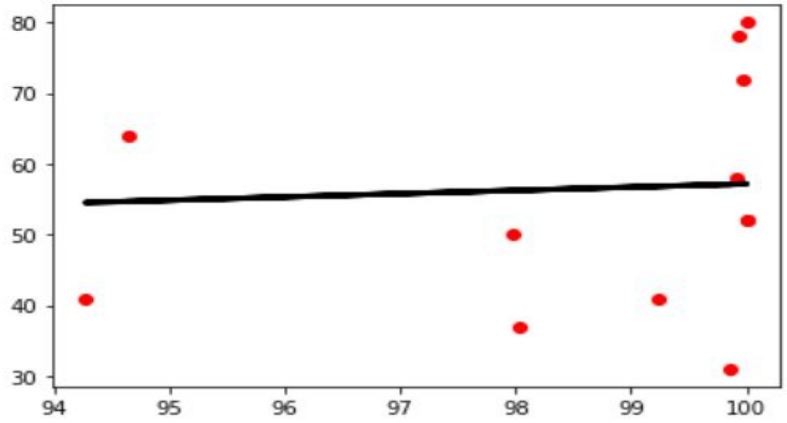
## ACCURACY

- The **accuracy** obtained for SDG 2 is **82%**

```
print("\nAccuracy of model for SDG 2")
print(regr.score(X_test, y_test))
```

```
Accuracy of model for SDG 2
0.8196961712289462
```

## Regression Plot



# Applying model on SDG 3 (Good Health and Well-Being):

## CORRELATION MATRIX:

- The correlation matrix for SDG 3 is :

| | SNO | Maternal Mortality Ratio (per 1,00,000 live births) | Under 5 mortality rate (per 1,000 live births) | Percentage of children in the age group 9-11 months fully immunized | Total case notification rate of Tuberculosis per 1,00,000 population | HIV incidence per 1,000 uninfected population | Suicide rate (per 1,00,000 population) | Death rate due to road traffic accidents (per 1,00,000 population) | Percentage of institutional deliveries out of the total deliveries reported | Monthly per capita out-of-pocket expenditure on health as a share of Monthly Per capita Consumption Expenditure (MPCE) | Total physicians, nurses and midwives per 10,000 population |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.031627 | -0.323013 | -0.096240 | 0.178306 | 0.011842 | 0.087071 | -0.429823 | 0.170886 | -0.109023 | -0.256332 |
| Maternal Mortality Ratio (per 1,00,000 live births) | -0.031627 | 1.000000 | 0.810856 | -0.221321 | 0.095464 | -0.001973 | -0.289163 | -0.168232 | -0.567982 | -0.009107 | -0.423404 |
| Under 5 mortality rate (per 1,000 live births) | -0.323013 | 0.810856 | 1.000000 | -0.231834 | -0.018108 | 0.012617 | -0.117524 | 0.103766 | -0.364945 | -0.220694 | -0.484328 |
| Percentage of children in the age group 9-11 months fully immunized | -0.096240 | -0.221321 | -0.231834 | 1.000000 | -0.300392 | -0.172503 | -0.349899 | 0.175910 | -0.102280 | 0.542808 | 0.035818 |
| Total case notification rate of Tuberculosis per 1,00,000 population | 0.178306 | 0.095464 | -0.018108 | -0.300392 | 1.000000 | 0.115514 | 0.096925 | 0.087692 | 0.101186 | -0.335103 | -0.111446 |

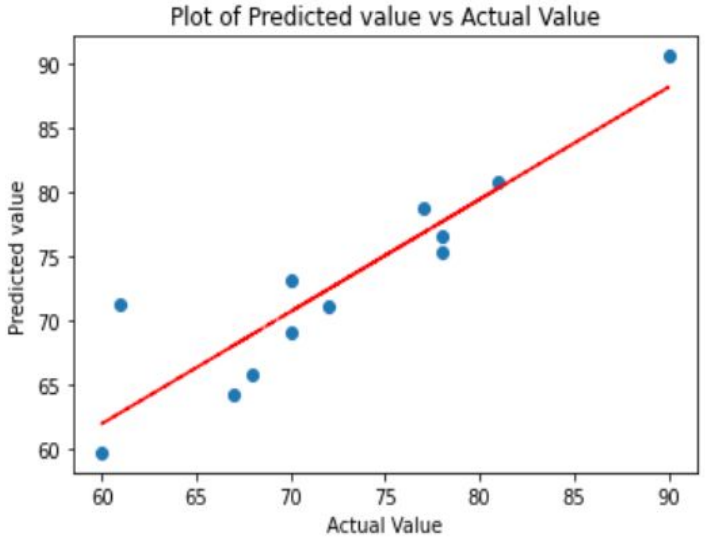
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 3.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 3']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual Value | Predicted Value |
|---|---|---|
| 13 | 78 | 75.251183 |
| 14 | 70 | 73.043680 |
| 29 | 70 | 69.022582 |
| 31 | 81 | 80.794315 |
| 16 | 78 | 76.507296 |
| 9 | 90 | 90.594138 |
| 33 | 67 | 64.270255 |
| 0 | 68 | 65.761851 |
| 34 | 60 | 59.673861 |
| 25 | 61 | 71.252012 |
| 12 | 72 | 71.061777 |
| 1 | 77 | 78.684880 |




Note: The numbers in 1st column represent SL No of States/UTs

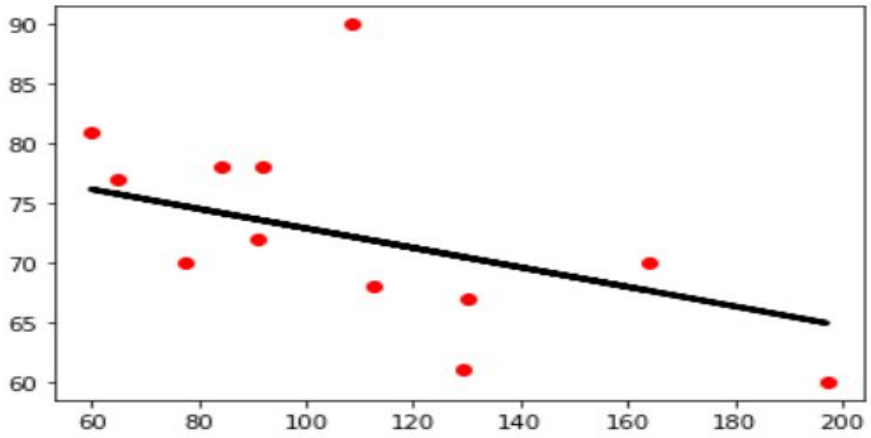
## ACCURACY

- The **accuracy** obtained for SDG 3 is **83%**

```
print("\nAccuracy of model for SDG 3")
print(regr.score(X_test, y_test))
```

```
Accuracy of model for SDG 3
0.8250960237732593
```

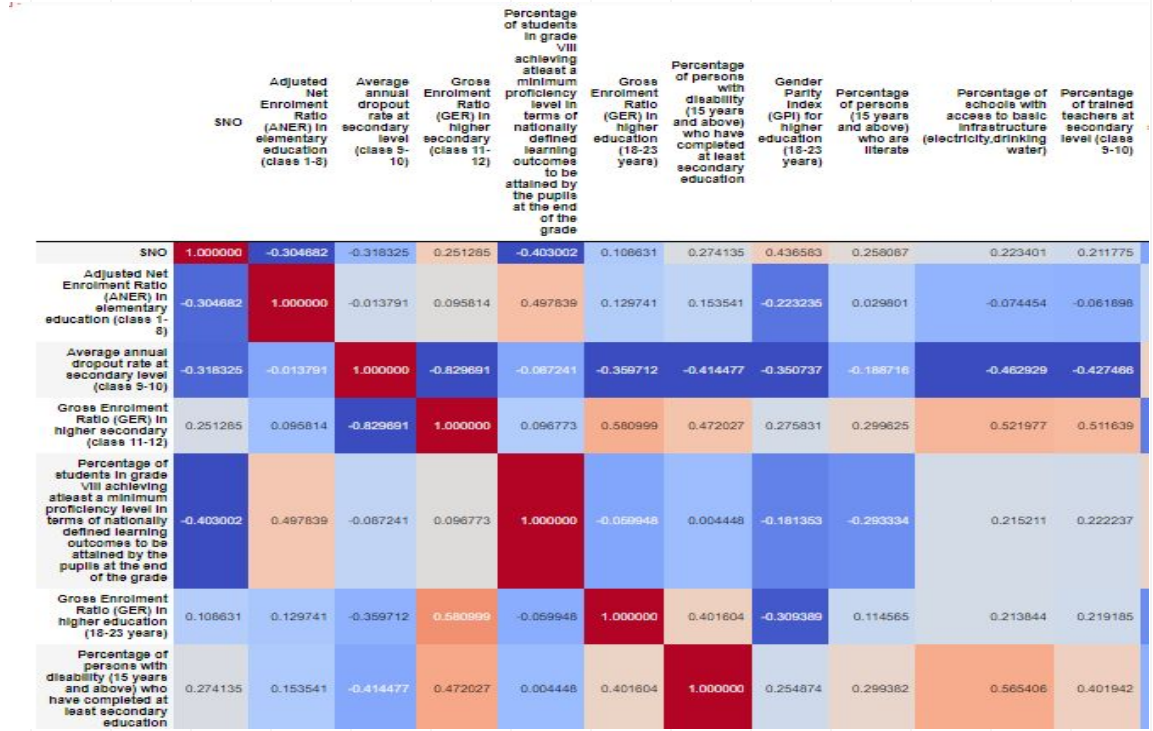
## Regression Plot



# Applying model on SDG 4 (Quality Education):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.



| | SNO | Adjusted Net Enrolment Ratio (ANER) in elementary education (class 1-8) | Average annual dropout rate at secondary level (class 9-10) | Gross Enrolment Ratio (GER) in higher secondary (class 11-12) | Percentage of students in grade VIII achieving atleast a minimum proficiency level in terms of nationally defined learning outcomes to be attained by the pupils at the end of the grade | Gross Enrolment Ratio (GER) in higher education (18-23 years) | Percentage of persons with disability (15 years and above) who have completed at least secondary education | Gender Parity Index (GPI) for higher education (18-23 years) | Percentage of persons (15 years and above) who are literate | Percentage of schools with access to basic infrastructure (electricity,drinking water) | Percentage of trained teachers at secondary level (class 9-10) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.304682 | -0.318325 | 0.251285 | -0.403002 | 0.108631 | 0.274135 | 0.436583 | 0.258087 | 0.223401 | 0.211775 |
| Adjusted Net Enrolment Ratio (ANER) in elementary education (class 1-8) | -0.304682 | 1.000000 | -0.013791 | 0.095814 | 0.497839 | 0.129741 | 0.153541 | -0.223235 | 0.029801 | -0.074454 | -0.061898 |
| Average annual dropout rate at secondary level (class 9-10) | -0.318325 | -0.013791 | 1.000000 | -0.829691 | -0.087241 | -0.359712 | -0.414477 | -0.350737 | -0.188716 | -0.462929 | -0.427466 |
| Gross Enrolment Ratio (GER) in higher secondary (class 11-12) | 0.251285 | 0.095814 | -0.829691 | 1.000000 | 0.096773 | 0.580999 | 0.472027 | 0.275831 | 0.299625 | 0.521977 | 0.511639 |
| Percentage of students in grade VIII achieving atleast a minimum proficiency level in terms of nationally defined learning outcomes to be attained by the pupils at the end of the grade | -0.403002 | 0.497839 | -0.087241 | 0.096773 | 1.000000 | -0.059948 | 0.004448 | -0.181353 | -0.293334 | 0.215211 | 0.222237 |
| Gross Enrolment Ratio (GER) in higher education (18-23 years) | 0.108631 | 0.129741 | -0.359712 | 0.580999 | -0.059948 | 1.000000 | 0.401604 | -0.309389 | 0.114565 | 0.213844 | 0.219185 |
| Percentage of persons with disability (15 years and above) who have completed at least secondary education | 0.274135 | 0.153541 | -0.414477 | 0.472027 | 0.004448 | 0.401604 | 1.000000 | 0.254874 | 0.299382 | 0.565406 | 0.401942 |

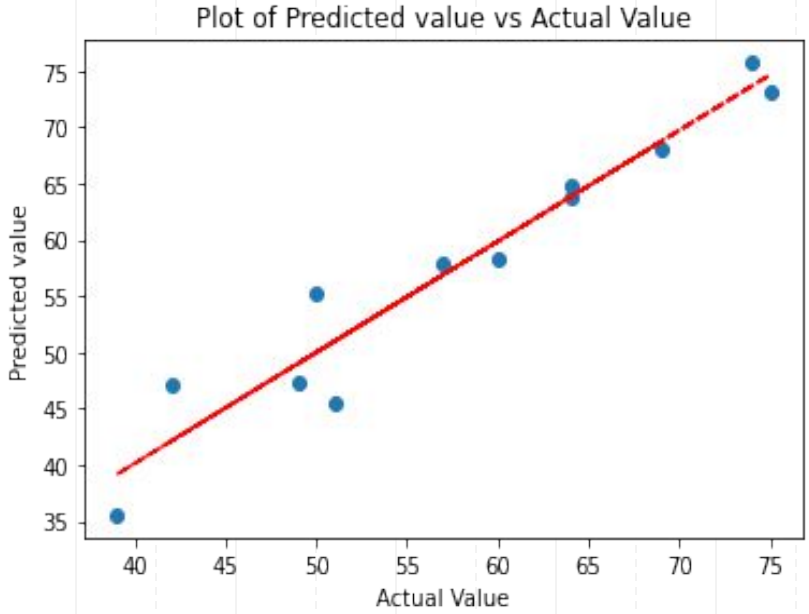
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 4.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 4']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 74 | 75.697969 |
| 14 | 49 | 47.403472 |
| 29 | 60 | 58.337599 |
| 31 | 69 | 68.016368 |
| 16 | 64 | 64.846523 |
| 9 | 75 | 73.067534 |
| 33 | 42 | 47.086735 |
| 0 | 57 | 57.873644 |
| 34 | 51 | 45.454045 |
| 25 | 39 | 35.538029 |
| 12 | 64 | 63.745210 |
| 1 | 50 | 55.288457 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 1 is **92.9%**

```
print("\nAccuracy of model for SDG 4 :")
print(regr.score(X_test, y_test))
```

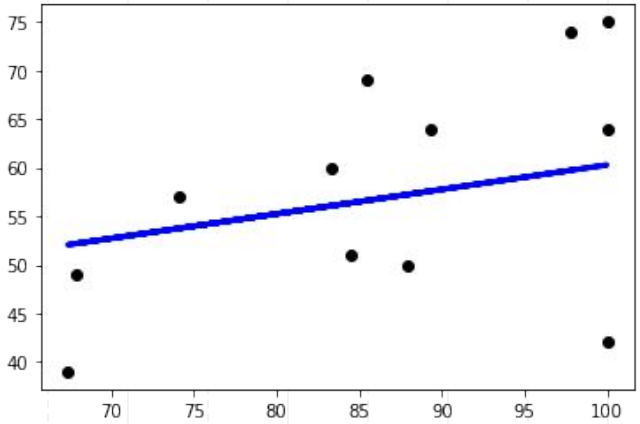
```
Accuracy of model for SDG 4 :
0.9285381999279552
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 5 (Gender Equality):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

| | SNO | Rate of crimes against women per 1,00,000 female population | Sex ratio at birth | Ratio of female to male average wage | Per lakh women who have experienced cruelty/physical violence by husband or his relatives during the year | Percentage of elected women over total seats in the state legislative assembly | Ratio of female to male Labour Force Participation Rate (LFPR) (15-55 years) | Proportion of women in managerial positions including women in board of directors, in listed companies (per 1,000 persons) | Percentage of currently married women aged 15-45 years who have their demand for family planning satisfied by modern methods | Operational land holding gender wise (percentage of female operated operational holdings) |
|---|---|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.007389 | -0.185146 | 0.238859 | -0.117396 | -0.009286 | -0.044369 | 0.065875 | -0.019912 | 0.070814 |
| Rate of crimes against women per 1,00,000 female population | -0.007389 | 1.000000 | -0.165225 | -0.020177 | 0.800074 | 0.253138 | -0.233953 | -0.095840 | 0.101515 | -0.051141 |
| Sex ratio at birth | -0.185146 | -0.165225 | 1.000000 | -0.323025 | -0.212988 | -0.144661 | 0.433980 | -0.052569 | 0.031768 | -0.020498 |
| Ratio of female to male average wage | 0.238859 | -0.020177 | -0.323025 | 1.000000 | -0.239352 | -0.278998 | 0.111113 | 0.115270 | -0.123683 | 0.274747 |
| Per lakh women who have experienced cruelty/physical violence by husband or his relatives during the year | -0.117396 | 0.800074 | -0.212988 | -0.239352 | 1.000000 | 0.254454 | -0.242994 | -0.089991 | 0.274176 | -0.256208 |
| Percentage of elected women over total seats in the state legislative assembly | -0.009286 | 0.253138 | -0.144661 | -0.278998 | 0.254454 | 1.000000 | -0.079233 | 0.484341 | 0.333622 | -0.052238 |
| Ratio of female to male Labour Force Participation Rate (LFPR) (15-55 years) | -0.044369 | -0.233953 | 0.433980 | 0.111113 | -0.242994 | -0.079233 | 1.000000 | -0.082423 | 0.301027 | 0.209092 |
| Proportion of women in managerial positions including women in board of directors, in listed companies (per 1,000 persons) | 0.065875 | -0.095840 | -0.052569 | 0.115270 | -0.089991 | 0.484341 | -0.082423 | 1.000000 | 0.293688 | 0.090501 |
| Percentage of currently married women aged 15-45 years who have their demand for family planning | -0.019912 | 0.101515 | 0.031768 | -0.123683 | 0.274176 | 0.333622 | 0.301027 | 0.293688 | 1.000000 | 0.031425 |

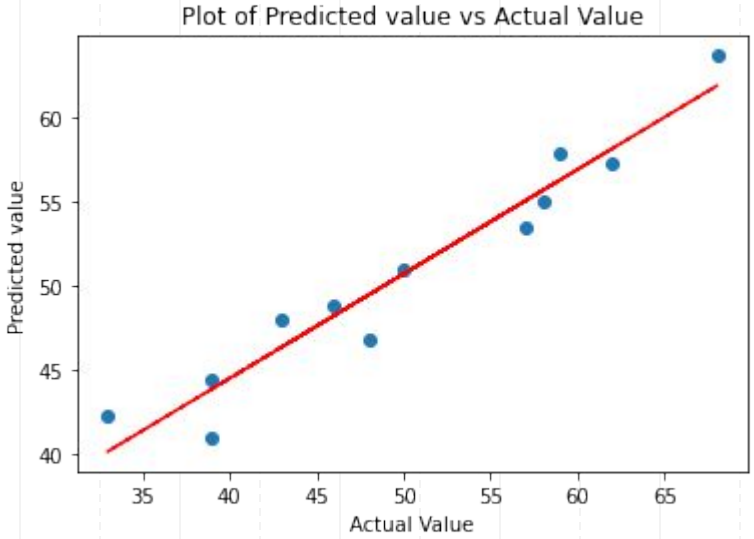
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 5.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 5']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 62 | 57.280510 |
| 14 | 46 | 48.803656 |
| 29 | 39 | 44.371718 |
| 31 | 59 | 57.869316 |
| 16 | 57 | 53.492865 |
| 9 | 33 | 42.236130 |
| 33 | 39 | 40.967129 |
| 0 | 68 | 63.683148 |
| 34 | 50 | 51.002570 |
| 25 | 48 | 46.782939 |
| 12 | 43 | 47.981109 |
| 1 | 58 | 54.960743 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 1 is **82.8%**

```
print("\nAccuracy of model for SDG 5 :")
print(regr.score(X_test, y_test))
```

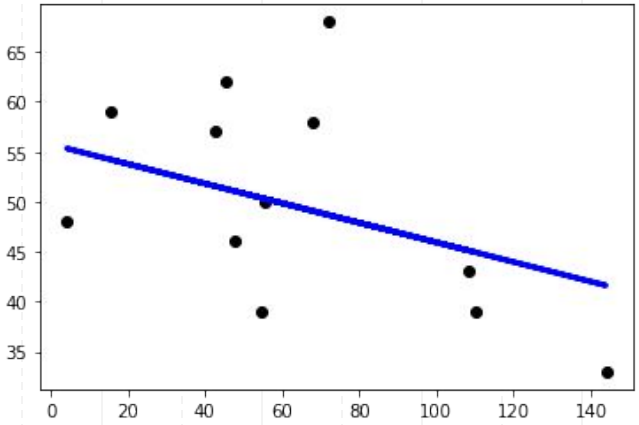
```
Accuracy of model for SDG 5 :
0.828073692639053
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 6 (Clean water and Sanitation):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

| | SNO | Percentage of population in the lowest two wealth quintiles * | Percentage of elected women over total seats in the State/UT (Lok Sabha elections) | Percentage of seats held by women in Panchayati Raj Institutions (PRIs) | Percentage of SC/ST seats in State Legislative Assemblies ** | Ratio of transgender to male Labour Force Participation Rate (LFPR) | Rate of total crimes against SCs (per 1,00,000 SC population) | Rate of total crimes against STs (per 1,00,000 ST population) |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.325351 | 0.048607 | -0.414761 | -0.168487 | -0.420597 | -0.336511 | 0.017433 |
| Percentage of population in the lowest two wealth quintiles * | -0.325351 | 1.000000 | 0.125694 | 0.325649 | 0.256395 | 0.020395 | 0.245331 | 0.093414 |
| Percentage of elected women over total seats in the State/UT (Lok Sabha elections) | 0.048607 | 0.125694 | 1.000000 | 0.052362 | 0.170733 | 0.126262 | -0.153815 | 0.028194 |
| Percentage of seats held by women in Panchayati Raj Institutions (PRIs) | -0.414761 | 0.325649 | 0.052362 | 1.000000 | -0.269101 | -0.035855 | 0.294645 | -0.057999 |
| Percentage of SC/ST seats in State Legislative Assemblies ** | -0.168487 | 0.256395 | 0.170733 | -0.269101 | 1.000000 | 0.489054 | -0.183551 | -0.253610 |
| Ratio of transgender to male Labour Force Participation Rate (LFPR) | -0.420597 | 0.020395 | 0.126262 | -0.035855 | 0.489054 | 1.000000 | -0.097232 | -0.106745 |
| Rate of total crimes against SCs (per 1,00,000 SC population) | -0.336511 | 0.245331 | -0.153815 | 0.294645 | -0.183551 | -0.097232 | 1.000000 | 0.503424 |
| Rate of total crimes against STs (per 1,00,000 ST population) | 0.017433 | 0.093414 | 0.028194 | -0.057999 | -0.253610 | -0.106745 | 0.503424 | 1.000000 |

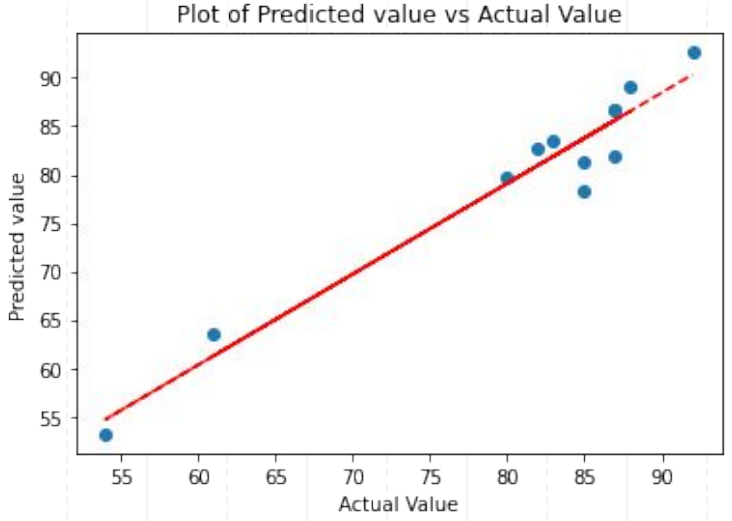
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 6.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 6']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 85 | 78.390373 |
| 14 | 88 | 89.134351 |
| 29 | 54 | 53.163549 |
| 31 | 87 | 81.824907 |
| 16 | 85 | 81.290798 |
| 9 | 61 | 63.608277 |
| 33 | 82 | 82.655540 |
| 0 | 87 | 86.737724 |
| 34 | 83 | 83.499497 |
| 25 | 87 | 86.710484 |
| 12 | 80 | 79.618997 |
| 1 | 92 | 92.639302 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 1 is **93.5%**

```
print("\nAccuracy of model for SDG 6 :")
print(regr.score(X_test, y_test))
```

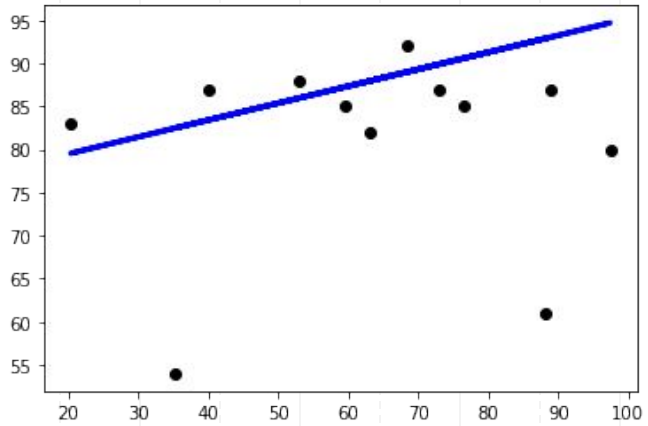
```
Accuracy of model for SDG 6 :
0.9346660231760135
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 7 (Affordable and Clean Energy):

**CORRELATION MATRIX:**

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

```
In [198]: corr = df.corr()
          corr.style.background_gradient(cmap='coolwarm')
```

Out[198]:

| | SNO | Percentage of households electrified | Percentage of LPG+PNG connections against number of households |
|---|---|---|---|
| SNO | 1.000000 | 0.187317 | -0.040398 |
| Percentage of households electrified | 0.187317 | 1.000000 | 0.182206 |
| Percentage of LPG+PNG connections against number of households | -0.040398 | 0.182206 | 1.000000 |

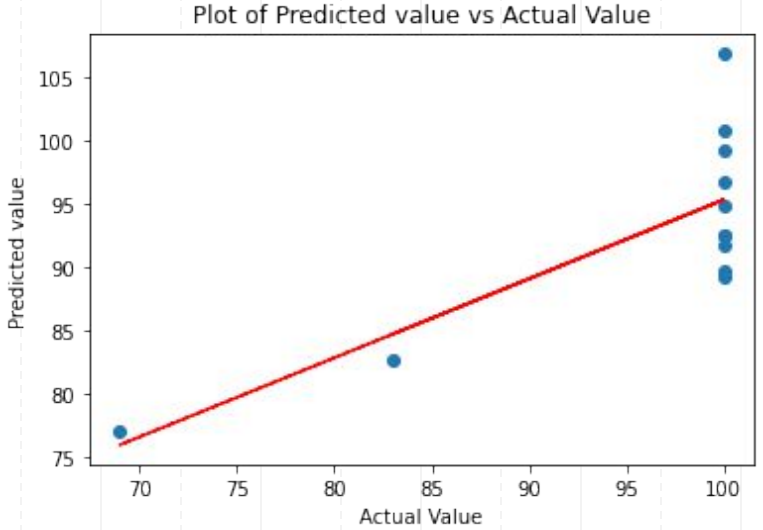
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 7.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 7']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 100 | 96.725069 |
| 14 | 100 | 100.846222 |
| 29 | 100 | 92.503402 |
| 31 | 100 | 89.304357 |
| 16 | 100 | 92.473917 |
| 9 | 100 | 106.814165 |
| 33 | 83 | 82.646522 |
| 0 | 100 | 94.835475 |
| 34 | 100 | 91.737498 |
| 25 | 69 | 77.136250 |
| 12 | 100 | 99.274126 |
| 1 | 100 | 89.628070 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 7 is **47.6%**

```
In [24]: print("\nAccuracy of model for SDG7 :")
         print(regr.score(X_test,y_test))
```

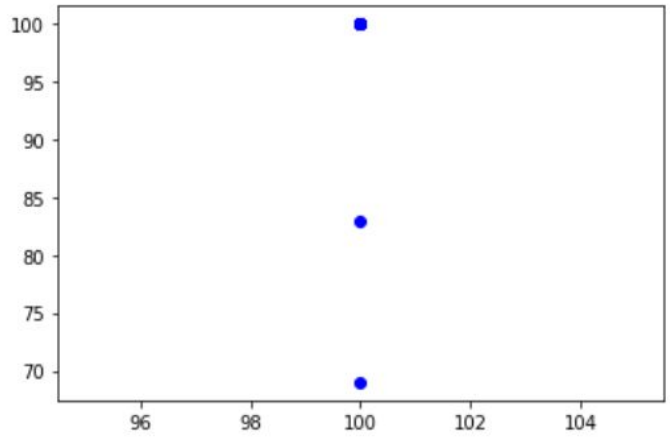
```
Accuracy of model for SDG7 :
0.4759164537346158
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 8 (Decent Work and Economic Growth):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

```
In [36]: corr = df.corr()
         corr.style.background_gradient(cmap='coolwarm')
```

Out[36]:

| | SNO | Annual growth rate of GDP (constant prices) per capita | Ease of Doing Business (EODB) Score (feedback score) | Unemployment rate (%) (15-59 years) | Labour Force Participation Rate (LFPR) (%) (15-59 years) | Percentage of regular wage/ salaried employees in non-agriculture sector without any social security benefit | Percentage of households covered with a bank account under PMJDY against target | Number of functioning branches of commercial banks per 1,00,000 population |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.007892 | 0.117476 | 0.050362 | -0.121544 | 0.099201 | -0.040738 | -0.106754 |
| Annual growth rate of GDP (constant prices) per capita | -0.007892 | 1.000000 | -0.055968 | 0.069271 | 0.138231 | 0.083803 | 0.355619 | 0.459389 |
| Ease of Doing Business (EODB) Score (feedback score) | 0.117476 | -0.055968 | 1.000000 | -0.161925 | 0.112637 | 0.564555 | 0.282473 | -0.243001 |
| Unemployment rate (%) (15-59 years) | 0.050362 | 0.069271 | -0.161925 | 1.000000 | -0.417410 | -0.333558 | 0.009324 | 0.160325 |
| Labour Force Participation Rate (LFPR) (%) (15-59 years) | -0.121544 | 0.138231 | 0.112637 | -0.417410 | 1.000000 | 0.164701 | 0.032770 | 0.200605 |
| Percentage of regular wage/ salaried employees in non-agriculture sector without any social security benefit | 0.099201 | 0.083803 | 0.564555 | -0.333558 | 0.164701 | 1.000000 | 0.191150 | -0.067091 |

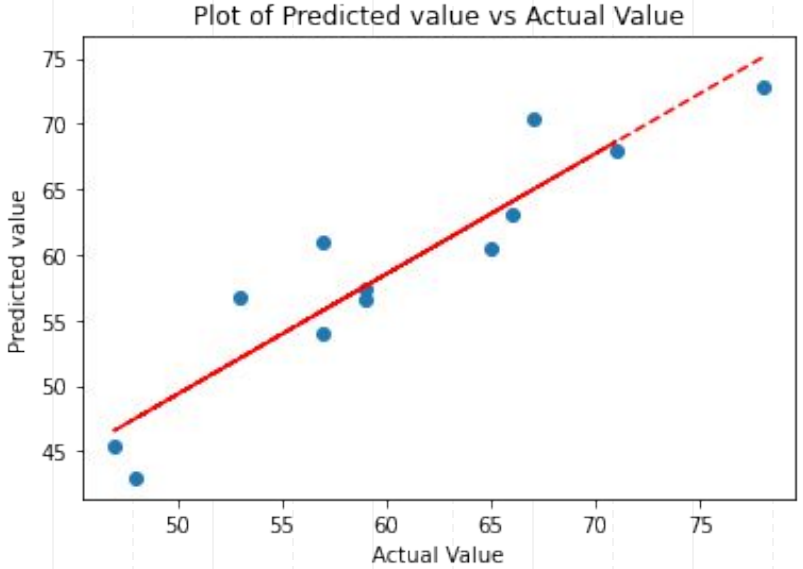
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 8.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 8']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 78 | 72.847893 |
| 14 | 47 | 45.417233 |
| 29 | 57 | 60.947773 |
| 31 | 71 | 67.921140 |
| 16 | 66 | 63.057593 |
| 9 | 65 | 60.490458 |
| 33 | 57 | 53.909841 |
| 0 | 59 | 57.356887 |
| 34 | 53 | 56.773799 |
| 25 | 48 | 42.957477 |
| 12 | 59 | 56.525690 |
| 1 | 67 | 70.425177 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 8 is **83.6%**

```
In [38]: print("\nAccuracy of model for SDG8 :")
         print(regr.score(X_test,y_test))
```

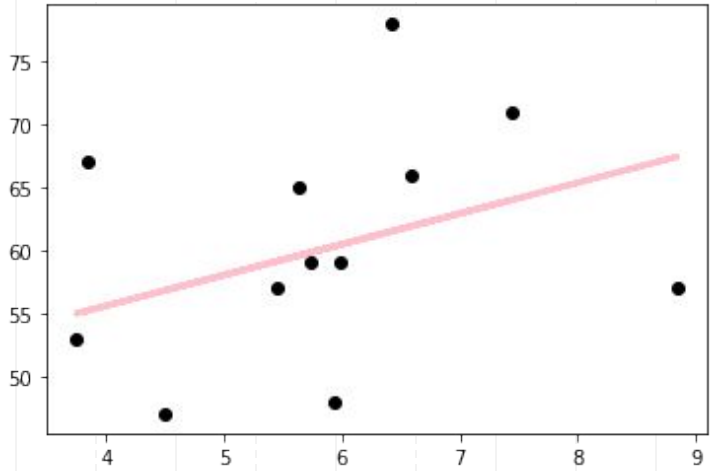
```
Accuracy of model for SDG8 :
0.8361279964420557
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 9 (Industry, Innovation and Infrastructure):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

```
In [50]: corr = df.corr()
         corr.style.background_gradient(cmap='coolwarm')
```

Out[50]:

| | SNO | Percentage of targeted habitations connected by all-weather roads under Pradhan Mantri Gram Sadak Yojana (PMGSY) | Percentage Share of GVA in manufacturing to total GVA (current prices) | Manufacturing employment as a percentage of total employment | Innovation score as per the India Innovation Index | Score as per Logistics Ease Across Different States (LEADS) report | Number of mobile connections per 100 persons (mobile tele density) | Number of internet subscribers per 100 population |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.349219 | -0.134157 | 0.327182 | 0.134737 | 0.105926 | 0.179083 | 0.367214 |
| Percentage of targeted habitations connected by all-weather roads under Pradhan Mantri Gram Sadak Yojana (PMGSY) | -0.349219 | 1.000000 | 0.418231 | -0.291907 | -0.069500 | -0.009376 | -0.031609 | -0.506495 |
| Percentage Share of GVA in manufacturing to total GVA (current prices) | -0.134157 | 0.418231 | 1.000000 | -0.068697 | 0.100574 | 0.161167 | 0.261338 | -0.219603 |
| Manufacturing employment as a percentage of total employment | 0.327182 | -0.291907 | -0.068697 | 1.000000 | 0.288167 | 0.319654 | -0.023245 | 0.695644 |
| Innovation score as per the India Innovation Index | 0.134737 | -0.069500 | 0.100574 | 0.288167 | 1.000000 | 0.572293 | 0.496430 | 0.571622 |
| Score as per Logistics Ease Across Different States (LEADS) report | 0.105926 | -0.009376 | 0.161167 | 0.319654 | 0.572293 | 1.000000 | 0.134332 | 0.350745 |
| Number of mobile connections per 100 persons (mobile tele density) | 0.179083 | -0.031609 | 0.261338 | -0.023245 | 0.496430 | 0.134332 | 1.000000 | 0.444615 |
| Number of internet subscribers per 100 population | 0.367214 | -0.506495 | -0.219603 | 0.695644 | 0.571622 | 0.350745 | 0.444615 | 1.000000 |

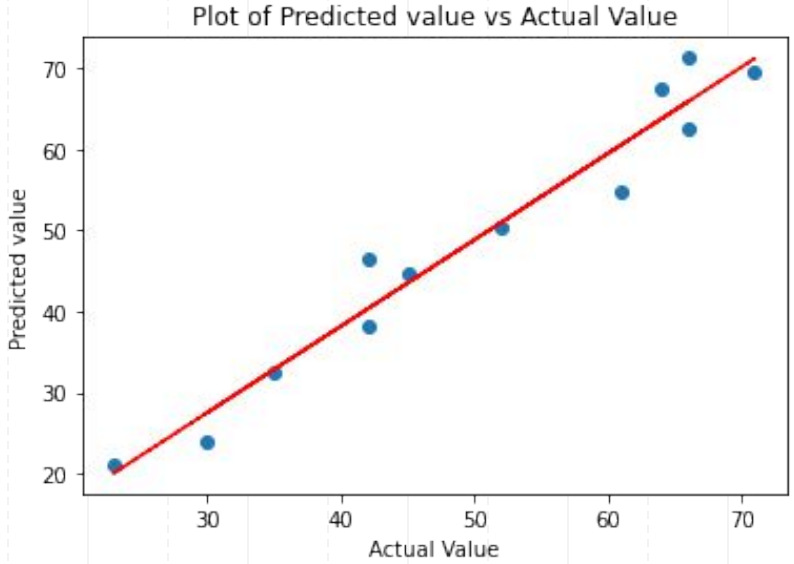
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 9.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 9']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 61 | 54.773283 |
| 14 | 42 | 38.168069 |
| 29 | 45 | 44.639056 |
| 31 | 71 | 69.557470 |
| 16 | 64 | 67.392915 |
| 9 | 66 | 71.314398 |
| 33 | 35 | 32.409758 |
| 0 | 23 | 21.150625 |
| 34 | 42 | 46.431075 |
| 25 | 30 | 23.823570 |
| 12 | 66 | 62.643084 |
| 1 | 52 | 50.490522 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 9 is **93.6%**

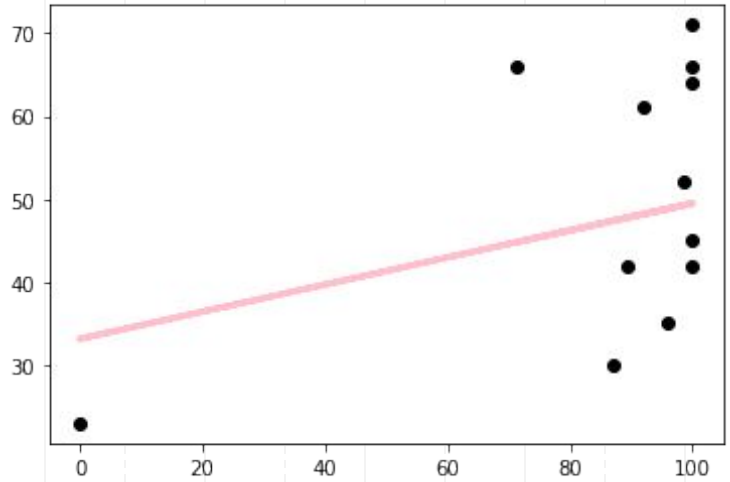
```
In [29]: print("\nAccuracy of model for SDG9 :")
         print(regr.score(X_test,y_test))

         Accuracy of model for SDG9 :
         0.9363793742066171
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 10 (Reduce Inequalities):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

| | SNO | Percentage of population in the lowest two wealth quintiles * | Percentage of elected women over total seats in the State/UT (Lok Sabha elections) | Percentage of seats held by women in Panchayati Raj Institutions (PRIs) | Percentage of SC/ST seats in State Legislative Assemblies ** | Ratio of transgender to male Labour Force Participation Rate (LFPR) | Rate of total crimes against SCs (per 1,00,000 SC population) | Rate of total crimes against STs (per 1,00,000 ST population) |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.325351 | 0.048607 | -0.414761 | -0.168487 | -0.420597 | -0.336511 | 0.017433 |
| Percentage of population in the lowest two wealth quintiles * | -0.325351 | 1.000000 | 0.125694 | 0.325649 | 0.256395 | 0.020395 | 0.245331 | 0.093414 |
| Percentage of elected women over total seats in the State/UT (Lok Sabha elections) | 0.048607 | 0.125694 | 1.000000 | 0.052362 | 0.170733 | 0.126262 | -0.153815 | 0.028194 |
| Percentage of seats held by women in Panchayati Raj Institutions (PRIs) | -0.414761 | 0.325649 | 0.052362 | 1.000000 | -0.269101 | -0.035855 | 0.294645 | -0.057999 |
| Percentage of SC/ST seats in State Legislative Assemblies ** | -0.168487 | 0.256395 | 0.170733 | -0.269101 | 1.000000 | 0.489054 | -0.183551 | -0.253610 |
| Ratio of transgender to male Labour Force Participation Rate (LFPR) | -0.420597 | 0.020395 | 0.126262 | -0.035855 | 0.489054 | 1.000000 | -0.097232 | -0.106745 |
| Rate of total crimes against SCs (per 1,00,000 SC population) | -0.336511 | 0.245331 | -0.153815 | 0.294645 | -0.183551 | -0.097232 | 1.000000 | 0.503424 |
| Rate of total crimes against STs (per 1,00,000 ST population) | 0.017433 | 0.093414 | 0.028194 | -0.057999 | -0.253610 | -0.106745 | 0.503424 | 1.000000 |

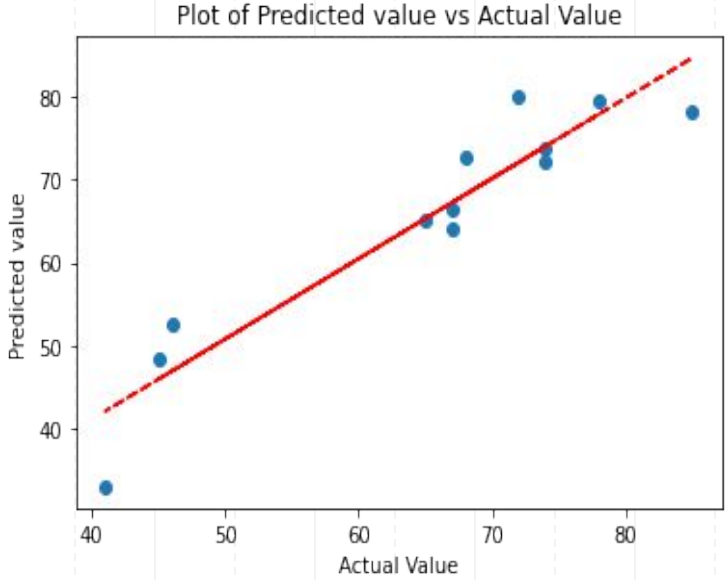
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 10.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 10']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 78 | 79.614381 |
| 14 | 65 | 65.103449 |
| 29 | 45 | 48.263767 |
| 31 | 74 | 73.745862 |
| 16 | 67 | 66.328961 |
| 9 | 72 | 79.990434 |
| 33 | 85 | 78.272591 |
| 0 | 67 | 64.030768 |
| 34 | 41 | 32.963297 |
| 25 | 46 | 52.549634 |
| 12 | 68 | 72.831716 |
| 1 | 74 | 72.277745 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 1 is **87.6%**

```
print("\nAccuracy of model for SDG 10 :")
print(regr.score(X_test, y_test))
```

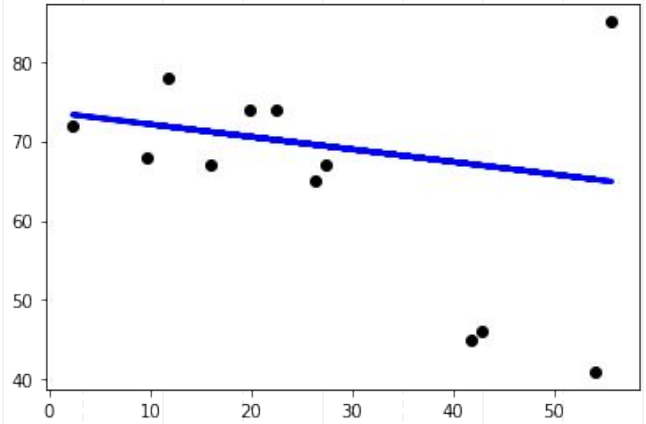
```
Accuracy of model for SDG 10 :
0.8755647530750277
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 11 (Sustainable Cities and Communities):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

| | SNO | Percentage of urban households living in katcha houses | Deaths due to road accidents in urban areas (per 1,00,000 population) | Percentage of wards with 100% door to door waste collection (SBM(U)) | Percentage of individual household toilets constructed against target (SBM(U)) | Percentage of MSW processed to the total MSW generated (SBM(U)) | Percentage of wards with 100% source segregation (SBM(U)) | Installed sewage treatment capacity as a percentage of sewage generated in urban areas | Percentage of urban households with drainage facility |
|---|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.279692 | -0.499584 | 0.039917 | -0.113016 | -0.264333 | -0.183023 | 0.056726 | 0.098059 |
| Percentage of urban households living in katcha houses | -0.279692 | 1.000000 | 0.092431 | -0.143012 | -0.118197 | -0.301894 | -0.168610 | -0.247213 | -0.350197 |
| Deaths due to road accidents in urban areas (per 1,00,000 population) | -0.499584 | 0.092431 | 1.000000 | 0.250748 | 0.030382 | 0.252685 | 0.138063 | 0.176735 | 0.187205 |
| Percentage of wards with 100% door to door waste collection (SBM(U)) | 0.039917 | -0.143012 | 0.250748 | 1.000000 | 0.329593 | 0.348360 | 0.484735 | 0.372982 | 0.280786 |
| Percentage of individual household toilets constructed against target (SBM(U)) | -0.113016 | -0.118197 | 0.030382 | 0.329593 | 1.000000 | 0.264355 | 0.439040 | 0.076721 | 0.224879 |
| Percentage of MSW processed to the total MSW generated (SBM(U)) | -0.264333 | -0.301894 | 0.252685 | 0.348360 | 0.264355 | 1.000000 | 0.675703 | 0.222198 | 0.105578 |
| Percentage of wards with 100% source segregation (SBM(U)) | -0.183023 | -0.168610 | 0.138063 | 0.484735 | 0.439040 | 0.675703 | 1.000000 | 0.241806 | 0.106604 |
| Installed sewage treatment capacity as a percentage of sewage generated in urban areas | 0.056726 | -0.247213 | 0.176735 | 0.372982 | 0.076721 | 0.222198 | 0.241806 | 1.000000 | 0.581896 |
| Percentage of urban households with drainage facility | 0.098059 | -0.350197 | 0.187205 | 0.280786 | 0.224879 | 0.105578 | 0.106604 | 0.581896 | 1.000000 |

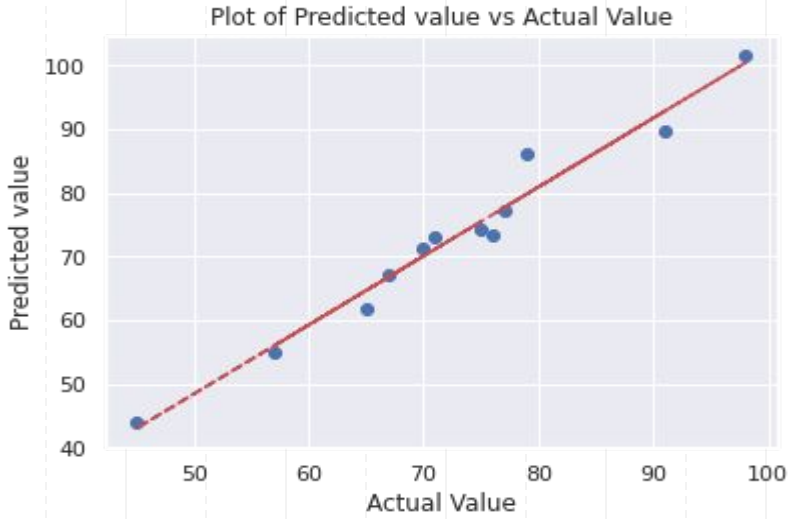
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 11.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 11']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 4 | 67 | 67.015453 |
| 26 | 70 | 71.217410 |
| 18 | 57 | 54.809011 |
| 22 | 65 | 61.788966 |
| 17 | 75 | 74.130984 |
| 36 | 45 | 43.945290 |
| 15 | 71 | 73.072044 |
| 28 | 91 | 89.635827 |
| 13 | 79 | 86.103294 |
| 34 | 77 | 77.305118 |
| 5 | 98 | 101.651531 |
| 27 | 76 | 73.388762 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 11 is **95.6%**

```
print("\nAccuracy of model for SDG 11 :")
print(regr.score(X_test, y_test))
```

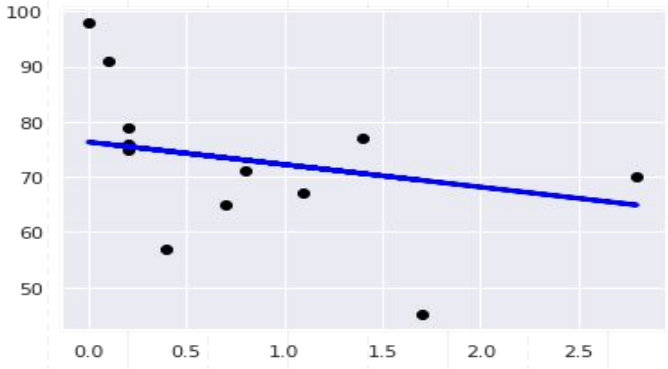
```
Accuracy of model for SDG 11 :
0.9559717183367249
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='black')
plt.plot(x_test, y_pred, color='blue', linewidth=3)
```

**Plot**



# Applying model on SDG 12 (Responsible Consumption and Production):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

| | SNO | Per capita fossil fuel consumption (in kg.) | Percentage use of nitrogenous fertilizer out of total N,P,K, (Nitrogen, Phosphorous, Potassium) | Hazardous waste generated per 1,000 population (Metric tonnes/Annum) | Quantity of hazardous waste recycled/utilized to total hazardous waste generated (%) | Plastic waste generated per 1,000 population (Tonnes/Annum) | Percentage of BMW treated to total quantity of BMW generated | Installed capacity of grid interactive bio power per 10 lakh population (MW) |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | 0.294976 | 0.012467 | -0.224677 | 0.324530 | 0.081943 | 0.253774 | 0.000021 |
| Per capita fossil fuel consumption (in kg.) | 0.294976 | 1.000000 | 0.193160 | 0.163739 | -0.163630 | 0.317263 | 0.260780 | 0.018539 |
| Percentage use of nitrogenous fertilizer out of total N,P,K, (Nitrogen, Phosphorous, Potassium) | 0.012467 | 0.193160 | 1.000000 | 0.230792 | 0.110502 | 0.056036 | 0.136373 | 0.087611 |
| Hazardous waste generated per 1,000 population (Metric tonnes/Annum) | -0.224677 | 0.163739 | 0.230792 | 1.000000 | -0.168580 | 0.309703 | 0.203768 | -0.003300 |
| Quantity of hazardous waste recycled/utilized to total hazardous waste generated (%) | 0.324530 | -0.163630 | 0.110502 | -0.168580 | 1.000000 | -0.071352 | 0.095505 | -0.189454 |
| Plastic waste generated per 1,000 population (Tonnes/Annum) | 0.081943 | 0.317263 | 0.056036 | 0.309703 | -0.071352 | 1.000000 | 0.253480 | -0.013480 |
| Percentage of BMW treated to total quantity of BMW generated | 0.253774 | 0.260780 | 0.136373 | 0.203768 | 0.095505 | 0.253480 | 1.000000 | 0.237458 |
| Installed capacity of grid interactive bio power per 10 lakh population (MW) | 0.000021 | 0.018539 | 0.087611 | -0.003300 | -0.189454 | -0.013480 | 0.237458 | 1.000000 |

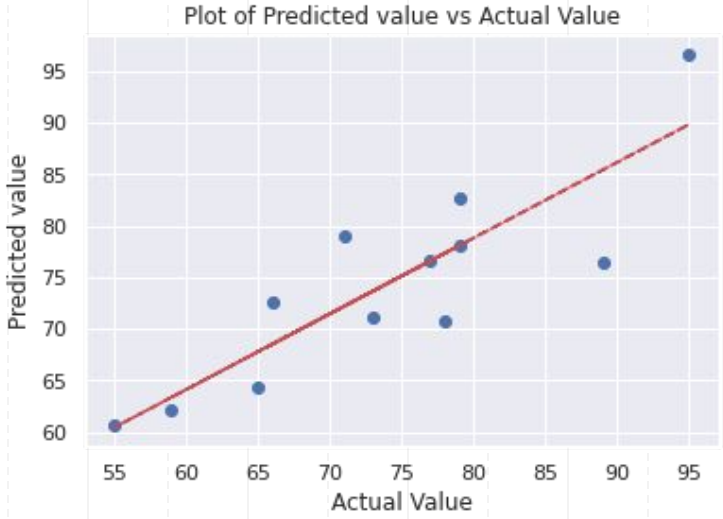
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 12.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 12']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 4 | 59 | 62.069829 |
| 26 | 73 | 71.020478 |
| 18 | 95 | 96.651895 |
| 22 | 89 | 76.464543 |
| 17 | 65 | 64.344649 |
| 36 | 79 | 78.154872 |
| 15 | 55 | 60.735237 |
| 28 | 71 | 78.961661 |
| 13 | 77 | 76.561360 |
| 34 | 79 | 82.711827 |
| 5 | 78 | 70.691373 |
| 27 | 66 | 72.518380 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 12 is **74.3%**

```
print("\nAccuracy of model for SDG 12 :")
print(regr.score(X_test, y_test))
```

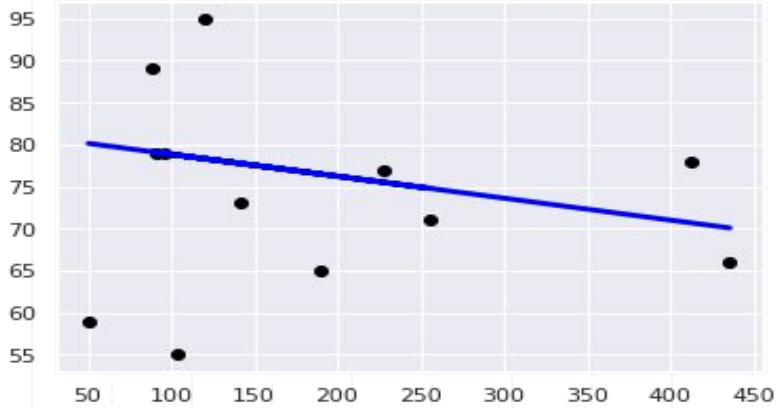
```
Accuracy of model for SDG 12 :
0.7431886123455201
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='black')
plt.plot(x_test, y_pred, color='blue', linewidth=3)
```

**Plot**



# Applying model on SDG 13 (Climate Action):

## CORRELATION MATRIX:

- To make an Idea about how each indicators of a particular SDG are related to each we have made use of an Correlation matrix.
- We have plotted correlation matrix using the code visible in the image.

| | SNO | Number of human lives lost per 1 crore population due to extreme weather events | Disaster preparedness score as per Disaster Resilience Index | Percentage of renewable energy out of total installed generating capacity (including allocated shares) | CO2 saved from LED bulbs per 1,000 population (Tonnes) | Disability Adjusted Life Years (DALY) rate attributable to air pollution (per 1,00,000 population) |
|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.236478 | -0.084599 | 0.100524 | 0.247380 | 0.126197 |
| Number of human lives lost per 1 crore population due to extreme weather events | -0.236478 | 1.000000 | 0.068849 | 0.444895 | 0.180144 | -0.357392 |
| Disaster preparedness score as per Disaster Resilience Index | -0.084599 | 0.068849 | 1.000000 | -0.143250 | -0.147209 | 0.188815 |
| Percentage of renewable energy out of total installed generating capacity (including allocated shares) | 0.100524 | 0.444895 | -0.143250 | 1.000000 | 0.411237 | -0.288011 |
| CO2 saved from LED bulbs per 1,000 population (Tonnes) | 0.247380 | 0.180144 | -0.147209 | 0.411237 | 1.000000 | -0.016506 |
| Disability Adjusted Life Years (DALY) rate attributable to air pollution (per 1,00,000 population) | 0.126197 | -0.357392 | 0.188815 | -0.288011 | -0.016506 | 1.000000 |

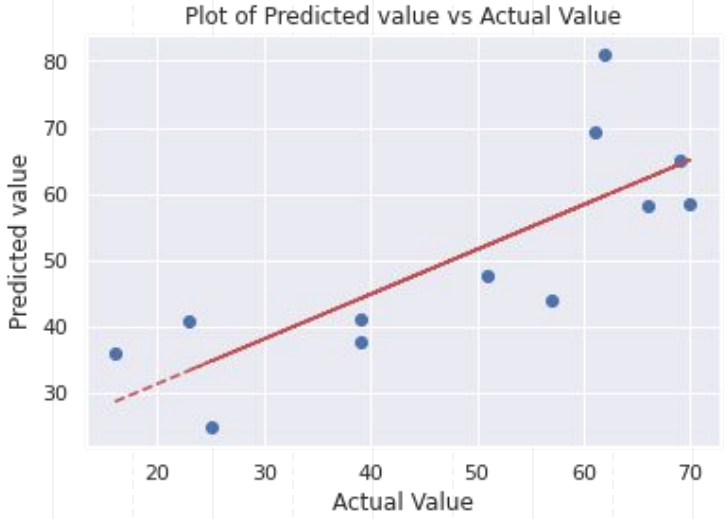
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 13.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 13']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 4 | 16 | 35.956854 |
| 26 | 70 | 58.447532 |
| 18 | 66 | 58.241371 |
| 22 | 57 | 43.975892 |
| 17 | 69 | 65.086886 |
| 36 | 39 | 41.069338 |
| 15 | 25 | 24.739419 |
| 28 | 51 | 47.590716 |
| 13 | 62 | 81.041399 |
| 34 | 39 | 37.812334 |
| 5 | 61 | 69.351699 |
| 27 | 23 | 40.790950 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 13 is **61.8%**

```
print("\nAccuracy of model for SDG 13 :")
print(regr.score(X_test, y_test))
```

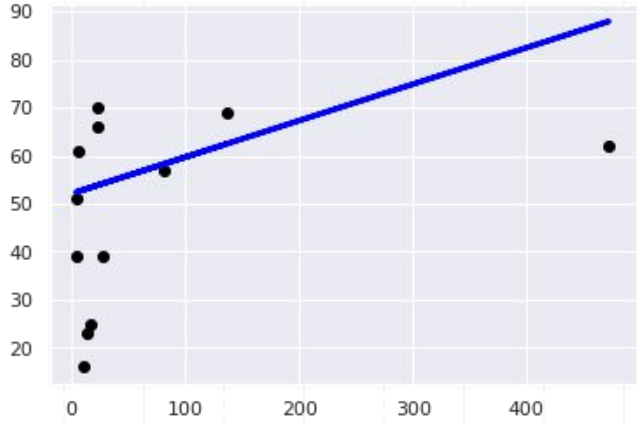
```
Accuracy of model for SDG 13 :
0.6183910983702875
```

## Regression Plot

The code used for plotting:

```
x_train = X_train.values[:,0].reshape(-1, 1)
x_test = X_test.values[:,0].reshape(-1, 1)
simple_reg = LinearRegression()
simple_reg.fit(x_train, y_train)
y_pred = simple_reg.predict(x_test)
plt.scatter(x_test, y_test,  color='red')
plt.plot(x_test, y_pred, color='black', linewidth=3)
```

**Plot**



“

**The dataset didn’t had SDG 14 scores of the states. Hence, It is skipped in our analysis.**

# Applying model on SDG 15 (Life on Land):

## CORRELATION MATRIX:

```
corr15 = df15.corr()
corr15.style.background_gradient(cmap='coolwarm')
```

| | SNO | Forest cover as a percentage of total gerographical area | Tree cover as a percentage of total geographical area | Combined 15.1+15.2 | Percentage of area covered under afforestation schemes to the total geographical area | Percentage of degraded land over total land area | Percentage increase in area of desertification | Number of cases under Wildlife Protection Act (1972) per million hectares of protected area |
|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.028246 | 0.210002 | -0.005628 | -0.028807 | -0.147144 | 0.021095 | -0.023735 |
| Forest cover as a percentage of total gerographical area | -0.028246 | 1.000000 | -0.232327 | 0.990890 | -0.178175 | -0.008606 | 0.304014 | 0.006963 |
| Tree cover as a percentage of total geographical area | 0.210002 | -0.232327 | 1.000000 | -0.100810 | 0.142592 | -0.288658 | -0.041829 | 0.025994 |
| Combined 15.1+15.2 | -0.005628 | 0.990890 | -0.100810 | 1.000000 | -0.160102 | -0.054260 | 0.304630 | 0.011526 |
| Percentage of area covered under afforestation schemes to the total geographical area | -0.028807 | -0.178175 | 0.142592 | -0.160102 | 1.000000 | 0.155531 | -0.141703 | -0.253136 |
| Percentage of degraded land over total land area | -0.147144 | -0.008606 | -0.288658 | -0.054260 | 0.155531 | 1.000000 | -0.046685 | -0.274482 |
| Percentage increase in area of desertification | 0.021095 | 0.304014 | -0.041829 | 0.304630 | -0.141703 | -0.046685 | 1.000000 | -0.044286 |
| Number of cases under Wildlife Protection Act (1972) per million hectares of protected area | -0.023735 | 0.006963 | 0.025994 | 0.011526 | -0.253136 | -0.274482 | -0.044286 | 1.000000 |

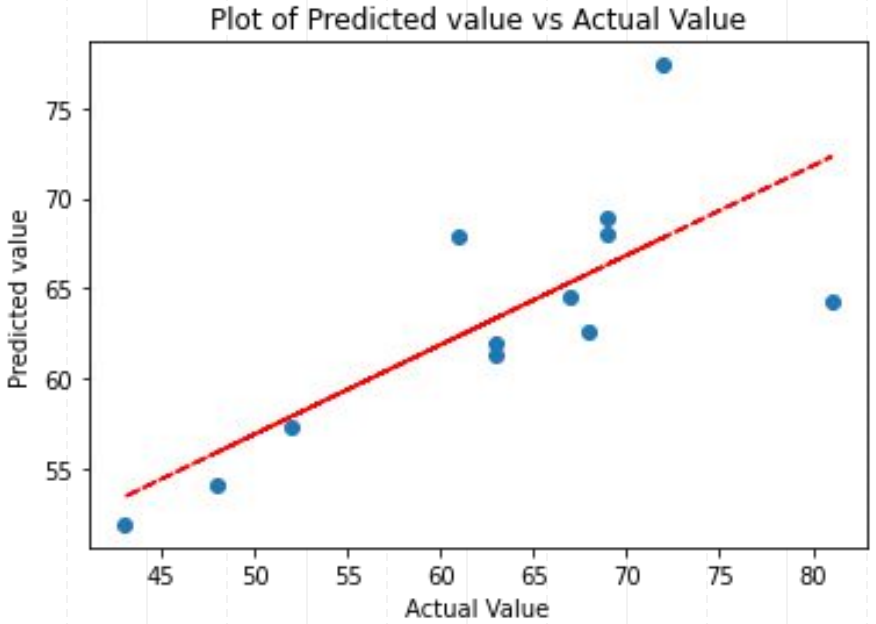
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 15.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 15']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 68 | 62.585545 |
| 14 | 52 | 57.319881 |
| 29 | 43 | 51.937010 |
| 31 | 63 | 61.337587 |
| 16 | 67 | 64.536444 |
| 9 | 81 | 64.274932 |
| 33 | 69 | 68.946807 |
| 0 | 72 | 77.387340 |
| 34 | 61 | 67.919705 |
| 25 | 63 | 62.012565 |
| 12 | 48 | 54.174804 |
| 1 | 69 | 67.992179 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 15 is **57.17%**

```
print("Accuracy of model for SDG 15 \n",regr15.score(X15_test, y15_test))
```

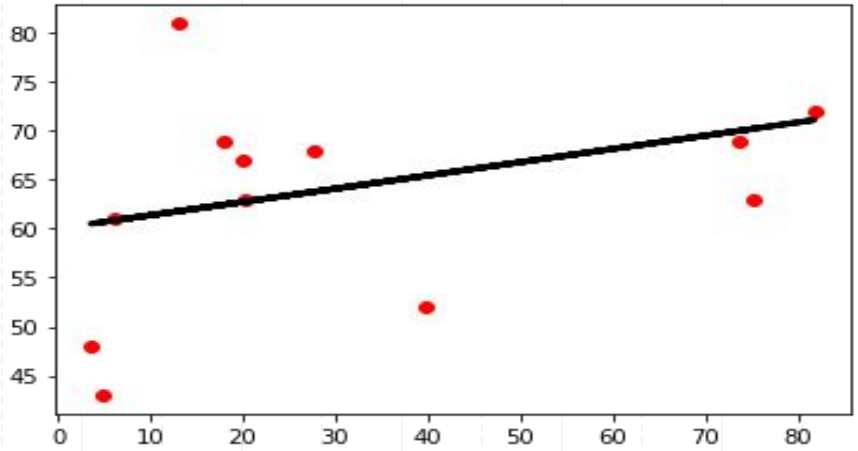
```
Accuracy of model for SDG 15
 0.5717068362882702
```

## Regression Plot

The code used for plotting:

```
x_train15 = X15_train.values[:,0].reshape(-1, 1)
x15_test = X15_test.values[:,0].reshape(-1, 1)
simple_reg15 = LinearRegression()
simple_reg15.fit(x15_train, y15_train)
y15_pred = simple_reg15.predict(x15_test)
plt.scatter(x15_test, y15_test,  color='red')
plt.plot(x15_test, y15_pred, color='black', linewidth=3)
```

**Plot**



# Applying model on SDG 16 (Peace Justice and Strong Institutions ):

## CORRELATION MATRIX:

```
corr16 = df16.corr()
corr16.style.background_gradient(cmap='coolwarm')
```

| | SNO | Murders per 1,00,000 population | Cognizable crimes against children per 1,00,000 population | Number of victims of human trafficking per 10 lakh population | Number of missing children per 1,00,000 child population | No. of courts per 1,00,000 population | Cases under Prevention of Corruption Act and related sections of IPC per 10 lakh population | Percentage of births registered | Percentage of population covered under Aadhaar |
|---|---|---|---|---|---|---|---|---|---|
| SNO | 1.000000 | -0.394968 | 0.247440 | -0.215864 | 0.168183 | 0.023590 | -0.152228 | -0.223304 | 0.204448 |
| Murders per 1,00,000 population | -0.394968 | 1.000000 | 0.028621 | 0.014374 | 0.222831 | -0.041870 | 0.075684 | 0.241327 | -0.088355 |
| Cognizable crimes against children per 1,00,000 population | 0.247440 | 0.028621 | 1.000000 | -0.005427 | 0.559596 | 0.315173 | -0.089882 | -0.308644 | 0.306581 |
| Number of victims of human trafficking per 10 lakh population | -0.215864 | 0.014374 | -0.005427 | 1.000000 | 0.086810 | 0.347083 | -0.018720 | 0.198097 | 0.035611 |
| Number of missing children per 1,00,000 child population | 0.168183 | 0.222831 | 0.559596 | 0.086810 | 1.000000 | 0.132213 | -0.042052 | 0.152993 | 0.408463 |
| No. of courts per 1,00,000 population | 0.023590 | -0.041870 | 0.315173 | 0.347083 | 0.132213 | 1.000000 | -0.100644 | 0.023565 | 0.105251 |
| Cases under Prevention of Corruption Act and related sections of IPC per 10 lakh population | -0.152228 | 0.075684 | -0.089882 | -0.018720 | -0.042052 | -0.100644 | 1.000000 | -0.090779 | 0.159340 |
| Percentage of births registered | -0.223304 | 0.241327 | -0.308644 | 0.198097 | 0.152993 | 0.023565 | -0.090779 | 1.000000 | -0.145878 |
| Percentage of population covered under Aadhaar | 0.204448 | -0.088355 | 0.306581 | 0.035611 | 0.408463 | 0.105251 | 0.159340 | -0.145878 | 1.000000 |

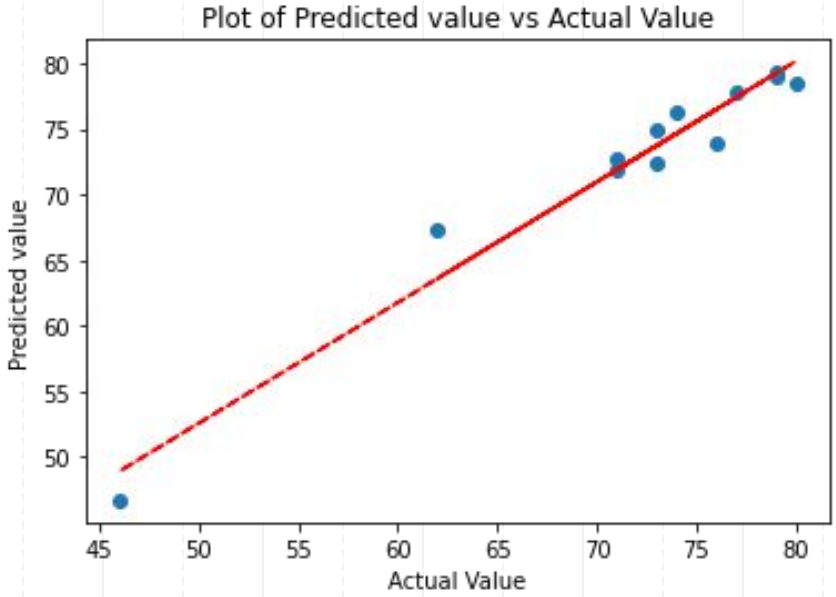
- The code we used for the testing is the following:

```
#reading csv file containing the data
df=pd.read_csv("Composite_Score.csv")
df1=pd.read_csv("SDG 16.csv")

#removing the redundant variables
train=df.drop(['SNO','Category','States/UTs'], axis=1)
test=df_comp['SDG 16']

#splitting the SDG dataset into two parts, train and test respectively
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size=0.3, random_state=2)
regr=LinearRegression()
pred_model=regr.fit(X_train, y_train)
pred=regr.predict(X_test)
regr_predictions= regr.predict(X_test)

#comparing presicted value and actual value
compare = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
print(compare)

#drawing plot of predicted vs actual value
plt.scatter(y_test, regr_predictions)
plt.title("Plot of Predicted value vs Actual Value")
plt.xlabel("Actual Value")
plt.ylabel("Predicted value")
z = np.polyfit(y_test, regr_predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test),"r--")
plt.show()
plt.close()
```

## RESULT

- The result obtained by our model is :

| | Actual | Predicted |
|---|---|---|
| 13 | 73 | 74.862573 |
| 14 | 74 | 76.290413 |
| 29 | 73 | 72.454906 |
| 31 | 71 | 72.705917 |
| 16 | 76 | 73.975578 |
| 9 | 62 | 67.379266 |
| 33 | 80 | 78.479662 |
| 0 | 46 | 46.673628 |
| 34 | 79 | 78.974296 |
| 25 | 79 | 79.320545 |
| 12 | 71 | 71.840762 |
| 1 | 77 | 77.780569 |




Note: The numbers in 1st column represent SL No of States/UTs

## ACCURACY

- The **accuracy** obtained for SDG 16 is **95.01%**

```
print("Accuracy of model for SDG 16 \n",regr16.score(X16_test, y16_test))
```

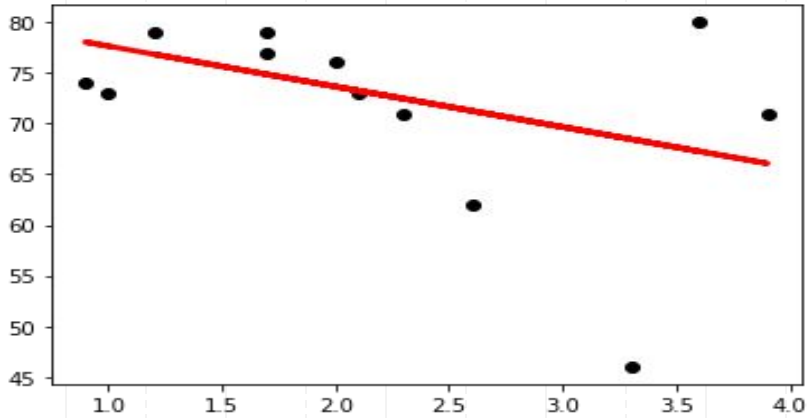
```
Accuracy of model for SDG 16
 0.9501728529441776
```

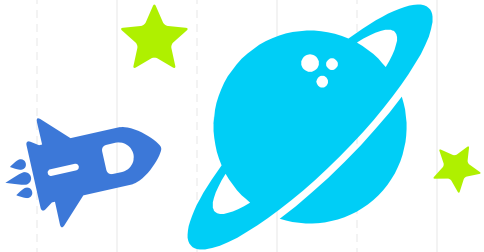
## Regression Plot

The code used for plotting:

```
x16_train = X16_train.values[:,0].reshape(-1, 1)
x16_test = X16_test.values[:,0].reshape(-1, 1)
simple_reg16 = LinearRegression()
simple_reg16.fit(x16_train, y16_train)
y_pred16 = simple_reg16.predict(x16_test)
plt.scatter(x16_test, y16_test,  color='black')
plt.plot(x16_test, y_pred16, color='red', linewidth=3)
```

**Plot**





# CONCLUSION

Here is the final conclusion of our project

- ➔ This project helped us in learning the basics of data analysis and how to handle data from real life problems.
- ➔ First we gathered the data from different websites and sources to form a database on which the analysis can be done.
- ➔ Then we learned how to handle irregularities or error present in the data and cleaned our dataset and represented it in various forms of graphs and tables.
- ➔ Then came the hypothesis testing portion in which the task was to check relationship between the two sets of data. In the project we checked whether there exists any relationship between any two SDGs.
- ➔ In the end we learned how to use machine learning to predict the future output from the dataset. In the project we used ML to predict the future composite score of each SDGs for each states and union territories and found that if the predicted value is less than the actual value than the state has performed better than the expectations and if the predicted value is more than the actual value, than the state is not performing upto the mark

# THANKS!