CSE464:Advanced Database System

[Fall 2021]

Project Report

Group No: 2 (Section 2)

Submitted by:

Student ID	Student Name	Contribution Percentage	Signature
2017-2-60-107	Moonwar Al Wardiful	24%	Moonwar
2017-2-60-096	Ruashed Md. Barket-E-Khuda	22%	Barket-E-Khuda
2017-2-60-124	Mahbub Hasan Shamim	22%	Shamim
2017-2-60-040	Jannatul Maoya	18%	Maoya
2017-2-60-085	Sumaiya Jahan Mim	14%	Mim

1. Introduction

We have used three types of datasets (covid_dataset,covid_first_doose and covid_second_doose) in our project. We assemble a model which can predict death and confirmed cases and estimate the number of days needed to cover the state with second dose vaccination by using those three datasets. We used a Regression Model where the model predicts the output as a continuous numerical value. We have used Linear Regression and Polynomial Function in that case. Firstly, We used the Linear Regression Model to predict death and confirmed cases, after that we used the Polynomial function for finding more close predicted values.

2. Data Preprocessing

In our project, we worked with three different CSV files which are covid_dataset, covid_first_dose, and covid_second_dose. Firstly, the date format of the covid dataset's values was not the same as vaccination doses day. So we made the format the same as the Day column as other CSV files using python. After that, we joined three CSV files with python using an outer join. After joining three CSV files there were many nan values. Then we filled up the nan values with zero. To build our model we needed a number of days. So we added an extra column named Ndays in this dataset. Which was filled by serial values like 1,2,3,..... We took all values in Numpy array format. In the other section, we can predict the total number of vaccinations. For this linear regression, we needed a cumulative sum of elements of the total number of the second dose. For this, we used the cumsum function of python.

3. Dataset Characteristics and Exploratory Data Analysis (EDA)

Here we used a linear regression model to build relationships among a set of predictor variables on a measured variable. In this instance, the dependent variables confirmed case and death case are dependent variables are total numbers of days is an independent variable. Before vaccination here, dependent variables are the death case and confirmed case, and the independent variables are numbers of days and after vaccination dependent variables are confirmed case and death case and dependent variable. Before and after vaccination, both segments have an independent variable is the number of days (output).

4. Machine Learning Models

There are two types of Machine learning. One is supervised learning and another one is unsupervised learning. Linear Regression is supervised learning. Linear regression works the data to predict dependent variables based on the given independent variables. The regression algorithm finds a linear relationship between dependent and independent variables.

We have used a linear regression model here. It's a supervised-based model which performs the task to predict a dependent variable value (y) based on a given independent variable (x) where the model predicts the output as a continuous numerical value. It finds out the relationship

between a dependent variable (target variable) dependent on several independent variables. Here x is basically input and y is output. If it has a continuous label then equation :
y=b^x+c.
y is the independent variable.
x is the dependent variable.
If dependent and independent variables same line did not plot then output will be some loss.
Loss Function:(predict output - actual output)^2.

polynomial model for determining which input factors response and direction. Polynomial regression model relation between independent(x) and dependent (y)variable is modelled nth degree of x variable. With polynomial regression, the data is approximated using a polynomial function. A polynomial is a function that takes the form $f(x) = c_0 + c_1 x + c_2 x^2 \cdots c_n x^n$ where n is the degree of the polynomial and c is a set of coefficients.

5. Description of Models and Associated Parameters

In our project, we used polynomial regression and linear regression. Where at first we join our dataset and here We begin by recognizing that even with constant predictor values, the response will vary, and we model this fact by treating the responses Yi as realizations of random variables. It is assumed that the expected response I is dependent on the predictor xi in this case, and that it is a linear function, say.

$$\mu_i = \alpha + \beta x_i.$$

A straight line is defined by this equation. The constant, also known as the intercept, represents the expected response when xi=0. (If zero is not in the data range, this quantity may not be of direct interest.) The slope is a parameter that represents the expected increase in response per unit change in xi. We saw a simple linear regression model with an explicit error written.

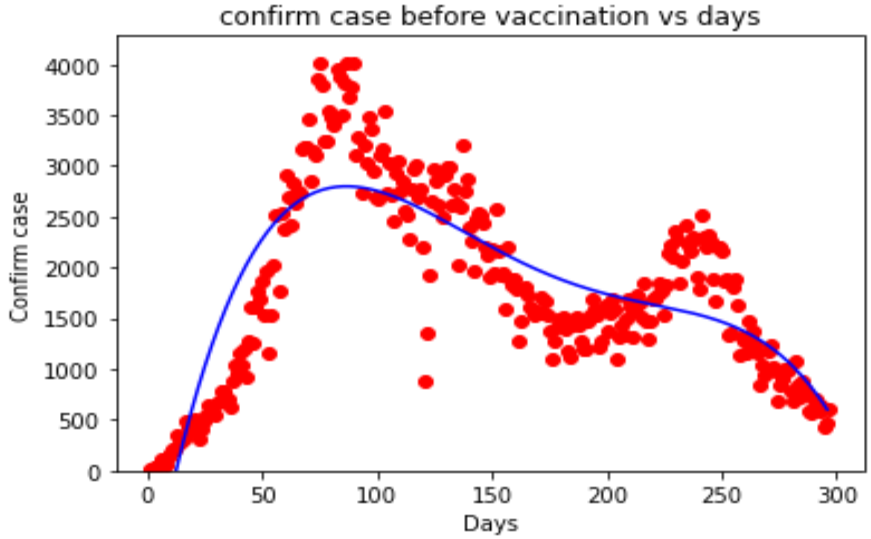
$$Y_i = \alpha + \beta x_i + \epsilon_i.$$

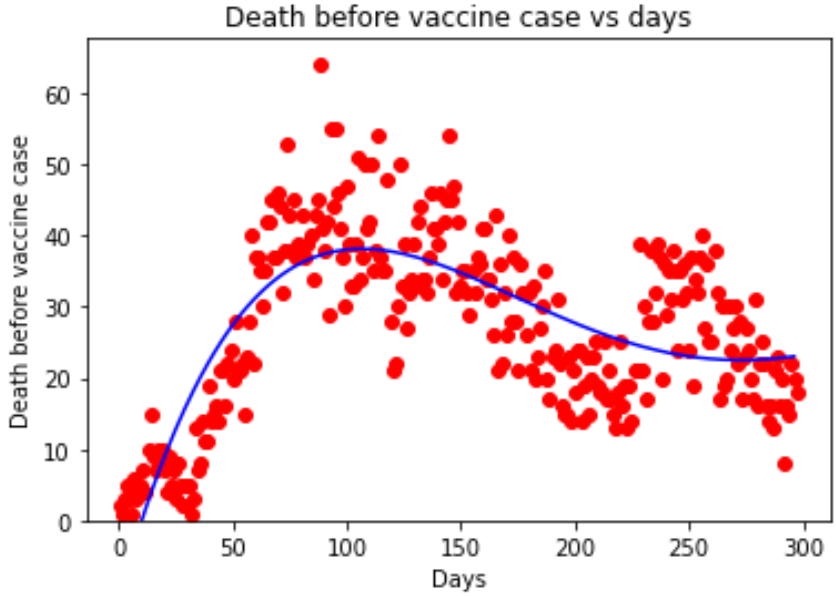
As a special case of the general linear model, the simple linear regression model can be obtained by dividing the model matrix X into two columns: a column of ones representing the constant and a column of x values representing the predictor. The general results can then be used to calculate parameter estimates, standard errors, and hypothesis tests. a simple linear regression The constant and slope estimates are provided by

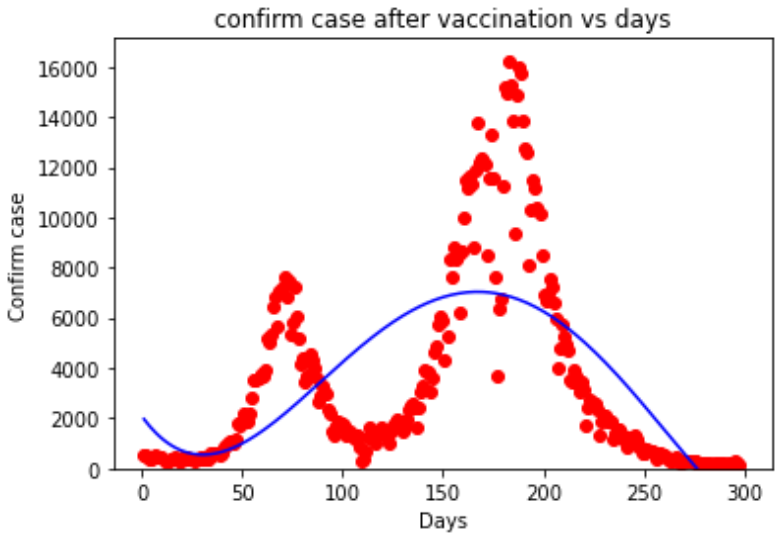
$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x} \quad \text{and} \quad \hat{\beta} = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}.$$

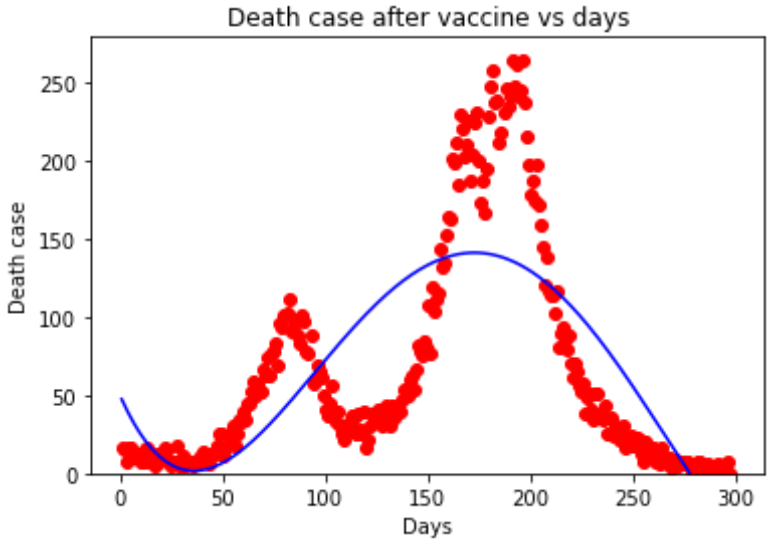
6. Performance Evaluation

Linear Regression is supervised learning. Linear regression works with the data to predict dependent variables based on the given independent variables. In our case, we split our data into two parts and then built our model. In the first part, we made a model with only non- vaccination days, and there we used polynomial regression. After plotting to confirm cases we find this graph in which the x-axis shows the number of days and the y-axis shows confirmed death cases. Here the blue line is our predicted line and red dots are the original values. Below we can see confirm before vaccination, death before vaccination graphs then we can see confirm after vaccination, death after vaccination graphs.

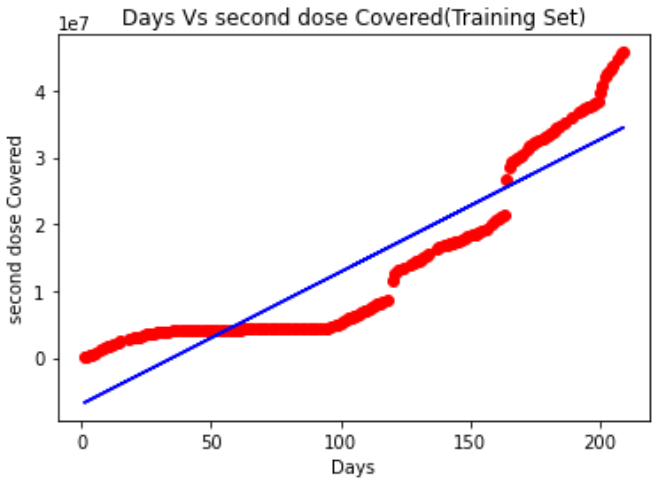




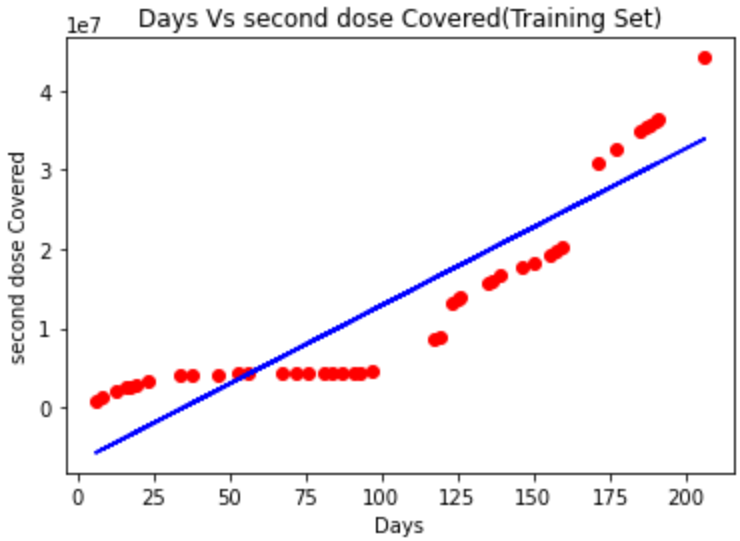




In the second section we did a simple linear regression where the second dose vaccination is the dependent variable. Here the first blue line is showing the predicted values. Here the predicted values are the number of second-dose vaccination. Here red dots are train data and blue line is predicted value



and here the blue line is predicted value and red dots are test data.



7. Discussion

The regression model has done a pretty good job of predicting the confirmed cases and death cases. In our result, there are 5 predicted data. and for the first four models, we used polynomial regression. For the first model, we predicted confirmed cases before the vaccination program and it gives about 81.32% accuracy. And before vaccination, the death case of our model gives 80.83% accuracy. After the vaccination program, we built another model that confirmed death cases accuracy 82.11% and 80.16%. So our model gives good accuracy. On the other hand, other models do not give as good accuracy as us.

In the second part of our code, we did linear regression in our model where it can predict the number of approximate peoples who give second dose vaccine. That means it takes a number of days as input and gives the value number of the population taking the second dose. It gives 81.36% accuracy.
In our dataset all values are numeric and the regression model gives the best prediction for serializing numeric data. So we think our model gives better performance than other models.

8. Conclusion

After finishing the project, using the data from the health center website of Bangladesh. It is very informative information or prediction for the Govt for vaccination of COVID-19. Using the three datasets, we find out the predicted death case and confirmed case.

Challenges:

We faced many challenges to do this project. Some of them are In the different CSV files. There the data format was different so we needed to reformat the data. There were some null values we processed to fill up the null values. We did not have any joined CSV files.We had three different CSV files.So joining the three different files was a challenge. At last, building ML models and accuracy analysis was a big challenge.

Opportunities:
Our model can predict death and confirm covid cases before vaccination and after vaccination so that, it also can predict how many days need to cover to finish the second dose for a given population. So we can easily use it for different government projects and we can predict the vaccination process and also we can predict death rate and confirm rate so Government and health centers can easily use this and also researchers can research with this dataset.

References

1. Linear regression - Wikipedia
2. sklearn.linear_model.LinearRegression — scikit-learn 1.0.2 documentation
3. notebook85134cb9d7 | Kaggle
4. Polynomial Regression | What is Polynomial Regression (analyticsvidhya.com)

Appendix

```
import os
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import pandas as pd
import matplotlib.pyplot as plt
```

```
from datetime import datetime

#load all data
covid_info=pd.read_csv('C:/Users/Moonwar/Desktop/Dataset/covid_dataset.csv')
first_dose=pd.read_csv('C:/Users/Moonwar/Desktop/Dataset/covid_first_dose.csv')
second_dose=pd.read_csv('C:/Users/Moonwar/Desktop/Dataset/covid_second_dose.csv')

# taking date
covid_info['Day'] = pd.to_datetime(covid_info['Day'], infer_datetime_format=True)
first_dose['Day'] = pd.to_datetime(first_dose['Day'], infer_datetime_format=True)
second_dose['Day'] = pd.to_datetime(second_dose['Day'], infer_datetime_format=True)

#all files date taking in same format
covid_info['Day'] = covid_info['Day'].dt.strftime('%-d-%m-%Y')
first_dose['Day'] = first_dose['Day'].dt.strftime('%-d-%m-%Y')
second_dose['Day'] = second_dose['Day'].dt.strftime('%-d-%m-%Y')

#outerjoin for merge all data
join1=pd.merge(covid_info, first_dose, on='Day', how='outer')
join2=pd.merge(join1, second_dose, on='Day', how='outer')

#replace Nan values with Zero
join2['Number of Vaccinations (First Dose)'] = join2['Number of Vaccinations (First Dose)'].replace(np.nan, 0)
join2['Number of Vaccinations (Second Dose)'] = join2['Number of Vaccinations (Second Dose)'].replace(np.nan, 0)

#model buildup
def model_train(x_train,y_train):
    #fitting the data into the model
    model = linear_model.LinearRegression()#Creating instance of linear model
    #sklearn takes input as matrix so the we have to reshape the matrix into colloum matrix
    x_train_for_fittng = np.matrix(x_train.reshape(len(x_train),1))
    y_train_for_fittng = np.matrix(y_train.reshape(len(y_train),1))

    #fit the data into the model
    return(model.fit(x_train_for_fittng,y_train_for_fittng))

#Function for Polynomial regrassing function
def poly_regr(X_train,Y_train,degree):
    polynom = PolynomialFeatures(degree=degree)
    X_poly = polynom.fit_transform(X_train.reshape(len(X_train),1))
    polyReg = LinearRegression()
    #Returing the fitted model
    return polyReg.fit(X_poly,Y_train.reshape(len(Y_train),1))

data_without_Vaccine=join2.head(297)

Days_withoutVaccine = np.array(data_without_Vaccine['Day'])
Confirmed_withoutVaccine = np.array(data_without_Vaccine['Confirmed case'])
Deaths_withoutVaccine = np.array(data_without_Vaccine['Death Case'])

data_without_Vaccine.insert(0, 'NDays', range(1, 1 + len(data_without_Vaccine)))
TotalDays_Without_Vaccine = np.array(data_without_Vaccine['NDays'])
random_indices = np.random.permutation(297)
```

```
Days_train_without_Vaccine = TotalDays_Without_Vaccine[random_indices[:]]

#calling the plynomial regrassion model for confirm case
polynom = PolynomialFeatures(degree=4)
model = poly_regr(Days_train_without_Vaccine,Confirmed_train_without_Vaccine,4)
#Printing the model
print(model)

#Pridiction the mode on traning data
predict =
model.predict(polynom.fit_transform(Days_train_without_Vaccine.reshape(len(Days_train_without_Vaccine),1)))

#Creating Array of custom Days
Days_array_without_vaccine = []
#Creating Days Data for further Prediction
for i in range(1,297):
   Days_array_without_vaccine.append(i)
#Converting arary into np array
Days_predict_without_vaccine = np.array(Days_array_without_vaccine)
#Predict the model on Future Date
predict_Confirm_without_vaccine =
model.predict(polynom.fit_transform(Days_predict_without_vaccine.reshape(len(Days_array_without_vaccine),1)))

plt.scatter(Days_train_without_Vaccine,Confirmed_train_without_Vaccine,color="red")
plt.plot(Days_predict_without_vaccine,predict_Confirm_without_vaccine,color='blue')
plt.title('confirm case before vaccination vs days')
plt.xlabel('Days')
plt.ylabel('Confirm case')
plt.ylim(ymin=0)
plt.show()




polynom = PolynomialFeatures(degree=4)
model = poly_regr(Days_train_without_Vaccine,Deaths_train_without_Vaccine,4)
#Printing the model
print(model)

#Pridiction the mode on traning data
predict =
model.predict(polynom.fit_transform(Days_train_without_Vaccine.reshape(len(Days_train_without_Vaccine),1)))

Days_array_without_vaccine = []
#Creating Days Data for further Prediction
for i in range(1,297):
   Days_array_without_vaccine.append(i)
#Converting arary into np array
Days_predict_without_vaccine = np.array(Days_array_without_vaccine)
#Predict the model on Future Date
predict_Confirm_without_vaccine =
model.predict(polynom.fit_transform(Days_predict_without_vaccine.reshape(len(Days_array_without_vaccine),1)))

plt.scatter(Days_train_without_Vaccine,Deaths_train_without_Vaccine,color="red")
```

```
plt.plot(Days_predict_without_vaccine,predict_Confirm_without_vaccine,color='blue')
plt.title('Death before vaccine case vs days')
plt.xlabel('Days')
plt.ylabel('Death before vaccine case')
plt.ylim(ymin=0)
plt.show()

Data_after_vaccine=join2.iloc[298:,:]

Days_aftervaccine = np.array(Data_after_vaccine['Day'])
Confirmed_aftervaccine = np.array(Data_after_vaccine['Confirmed case'])
Deaths_aftervaccine = np.array(Data_after_vaccine['Death Case'])

Data_after_vaccine.insert(0, 'NDays', range(1, 1 + len(Data_after_vaccine)))

TotalDays_After_Vaccine = np.array(Data_after_vaccine['NDays'])

random_indices1 = np.random.permutation(329)
Days_train_after_vaccine = TotalDays_After_Vaccine[random_indices[:]]
Confirmed_train_after_vaccine = Confirmed_aftervaccine[random_indices[:]]
Deaths_train_after_vaccine= Deaths_aftervaccine[random_indices[:]]


#calling the plynomial regrassion model for confirm case

polynom = PolynomialFeatures(degree=4)
model = poly_regr(Days_train_after_vaccine,Confirmed_train_after_vaccine,4)
#Printing the model
print(model)

#Pridiction the mode on traning data
predict =
model.predict(polynom.fit_transform(Days_train_after_vaccine.reshape(len(Days_train_after_vaccine),1)))



Days_array_after_vaccine = []
#Creating Days Data for further Prediction
for i in range(1,300):
    Days_array_after_vaccine.append(i)
#Converting arary into np array
Days_predict_after_vaccine = np.array(Days_array_after_vaccine)
#Predict the model on Future Date
predict_Confirm_after_vaccine =
model.predict(polynom.fit_transform(Days_predict_after_vaccine.reshape(len(Days_array_after_vaccine),1)))

plt.scatter(Days_train_after_vaccine,Confirmed_train_after_vaccine,color="red")
plt.plot(Days_predict_after_vaccine,predict_Confirm_after_vaccine,color='blue')
plt.title('confirm case after vaccination vs days')
plt.xlabel('Days')
plt.ylabel('Confirm case')
plt.ylim(ymin=0)
plt.show()
```

```
polynom = PolynomialFeatures(degree=4)
model = poly_regr(Days_train_after_vaccine,Deaths_train_after_vaccine,4)
#Printing the model
print(model)

#Pridiction the mode on traning data
predict =
model.predict(polynom.fit_transform(Days_train_after_vaccine.reshape(len(Days_train_after_vaccine),1)))

Days_array_after_vaccine = []
#Creating Days Data for further Prediction
for i in range(1,300):
   Days_array_after_vaccine.append(i)
#Converting arary into np array
Days_predict_after_vaccine = np.array(Days_array_after_vaccine)
#Predict the model on Future Date
predict_death_after_vaccine =
model.predict(polynom.fit_transform(Days_predict_after_vaccine.reshape(len(Days_array_after_vaccine),1)))

plt.scatter(Days_train_after_vaccine,Deaths_train_after_vaccine,color="red")
plt.plot(Days_predict_after_vaccine,predict_death_after_vaccine,color='blue')
plt.title('Death case after vaccine vs days')
plt.xlabel('Days')
plt.ylabel('Death case')
plt.ylim(ymin=0)
plt.show()

Second_dose=pd.read_csv('C:/Users/Moonwar/Desktop/Dataset/covid_second_dose.csv')
Second_dose.insert(0, 'NDays', range(1, 1 + len(Second_dose)))
Second_dose.head()
Second_dose['Total_covered'] = Second_dose['Number of Vaccinations (Second Dose)'].cumsum()

import numpy as np
del Second_dose['Number of Vaccinations (Second Dose)']
x=Second_dose.iloc[:,:-1].values
y=Second_dose.iloc[:,-1].values
import numpy.ma as ma
y=np.where(np.isnan(y), ma.array(y, mask=np.isnan(y)).mean(), y)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=0)

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
y_pred=regressor.predict(x_test)
r2_score = regressor.score(x_test,y_test)
print(r2_score*100,'%')

import matplotlib.pyplot as plt
plt.scatter(x_train,y_train,color='red')
plt.plot(x_train,regressor.predict(x_train),color='blue')
plt.title('Days Vs second dose Covered(Training Set)')
plt.xlabel('Days')
plt.ylabel('second dose Covered')
```

```
plt.show()
plt.scatter(x_test,y_test,color='red')
plt.plot(x_test,regressor.predict(x_test),color='blue')
plt.title('Days Vs second dose Covered(Training Set)')
plt.xlabel('Days')
plt.ylabel('second dose Covered')
plt.show()
```