# Covid Predictor using Polynomial Regression

*Raju Ranjan*
*rajur.ug18.ec@nitp.ac.in*
*B.Tech., Department of*
*Electronics and Communication Engineering*
*National Institute of Technology Patna, India*

*Shruti Sinha*
*shrutis.ug18.ec@nitp.ac.in*
*B.Tech., Department of*
*Electronics and Communication Engineering*
*National Institute of Technology Patna, India*

**Abstract**
**The novel coronavirus (COVID-19) outbreak produced devastating effects on the global economy and the health of entire communities. Although the COVID-19 survival rate is high, the number of severe cases that result in death is increasing daily. A timely prediction of at-risk patients of COVID-19 with precautionary measures is expected to increase the survival rate of patients and reduce the fatality rate. This research provides a prediction method for the early identification of COVID-19 patient's**

**The proposed model can assist the decision-making and health care professional by early identification of at-risk COVID-19 patients effectively.**

**Keywords -**
**Polynomial Regression, Machine learning, Feature engineering, Degree, Precision, Training data, Accuracy**

## 1 Introduction

COVID-19 is an infectious disease caused by a novel coronavirus which has first been originated in Wuhan city, Hubei Provinces of China [1], [2]. Severe Acute Respiratory Syndrome Coronavirus-2 (SARS-CoV-2) is a new type of virus family that has not been earlier identified in people. The virus seems to be transmitted mostly through the minute respiratory droplets via coughing, sneezing or when people interact with each other for some time in close proximity. These droplets can then be inhaled, or they can land on surfaces that others may come into touch with, who can then get contaminate when they contact their eyes, mouth, or nose. COVID-19 is spreading within the sort of an enormous epidemic for the globe.The World Health Organization states that COVID-19 could be spread from one person to another at a very fast speed through contact and respiratory spray.

Several regression analysis models have been applied for data analysis of COVID-19 of Egypt. In this study, we've been applied fpurregression analysis-based models that are exponential polynomial, quadratic, third-degree, fourth-degree, respectively for the COVID-19 dataset. Thus, the exponential, third degree polynomial regression models are excellent.

with the help of the proposed method to predict the future number of total cases, active cases, and recoveries. These tasks can help a country/region to understand the spreading of the virus, facilitate/aware people, start mitigations. It'll also help that region/country to be prepared for what's will happen in the future, which may help in saving lives and agony.

## 2 DataSet

- The case file is taken in which 'id' and 'cases' are taken for each day and the data is taken from the source "Our World in Data.org". This is the Corona Virus Source Data By Hannah Ritchie.
- The "Total Confirmed Cases" file is downloaded from there which is used as previous days data for doing the prediction of the future.

## 3 Methodology

### 3.1 Type of Model

- In case of non-linear relationship, where the data is not always increasing with the same ratio as the other variable. So, in these cases Polynomial functions are used. Multiple regression and polynomials or non-linear regression is a little bit confusing. Taking an example of Multiple linear regression

$$y = mx + c \tag{1}$$

$$y = c + m1x1 + m2x2 + m3x3 \tag{2}$$

- Now, if coming to polynomial equations, there are different types:-

1. LINEAR –> ax+b=0
2. SQUARE –> a$x^2$+bx+c=0
3. CUBIC –>a$x^3$+b$x^2$+cx+d=0

- Now in case of polynomial equation the 'x' is basically the same (single feature). The only difference is that we are creating our own feature by squaring it or cubing it etc.. $x^2$ or $x^3$ are the features that we are producing, these are not actual features.

### 3.2 Implementation

1. Importing Python Libraries:
   - Pandas
   - Numpy
   - sklearn
   - Matplotlib
2. Load Data
   - Here, we are starting by importing pandas as pd, numpy as np, matplotlib, sklearn, etc. then we will load our data. Then we will write– data = pd.read_csv(" ", sep=','). And we have our csv file already and the separation is comma as these are comma separated values.
   - After that – data= data[['id','cases']]. Then will print (data.head()) and this will give us information (i.e. id and cases).
   - Then we will prepare our data before actually sending it
3. Data Exploration
   - The preparation is in this way that we have to convert these into numpy array by writing–

     x = np.array (data['id']).reshape(-1,1)
   - This is required by machine learning library . Same thing will be done for 'cases

     y = np.array (data['cases']).reshape(-1,1)
   - This will be plotted and for that matplotlib is included– import matplotlib.pyplot as plt Plot y and provide specifications such as colour–

     plt.plot(y, '-m') and write plt.show()
4. Feature Engineering
   - The next step is to create our features (eg. $x^2$ or $x^3$....). So, it will need a separate column . For example if we have id – $id^2$ or $id^3$ or more have to be created.So, to create this functionality within scikit learn library can be used by writing.

     from sklearn.preprocessing import PolynomialFeatures
5. Feature Scaling
   - An object will be created by-

     polyFeat = PolynomialFeatures(degree=" ")
   - Here, we have to define the degree as per needed such as – degree=2 ($x^2$) , degree = 3 ($x^3$) , etc..Now 'x' can be written as

     x=polyFeat.fit_transform(x)
   - It will take this x and will add more columns to it so that we can have other features. Print (x) for values. So, this is enough for preprocessing our data and now this can be send to the model for training. Also we will not split because this already don't have a lot of values and we will push this as much as we can to train as good as possible.
6. Training Data
   - Import from sklearnimportlinear_model. Create the model (linear model) by– model =

     linear_model.LinearRegression() and then model.fit(x,y). After that we will check accuracy-

     accuracy = model.score(x,y)
   - then print accuracy and then round it off.
   - Because it will be in decimal so, we want to control h how much decimal places we need.

     print(f'Accuracy:'round(accuracy*100,3))
   - where 3 shows the number of decimal places. On running this accuracy can be found.
   - Now we will plot this, so we use prediction method

     y0 = model.predict(x)

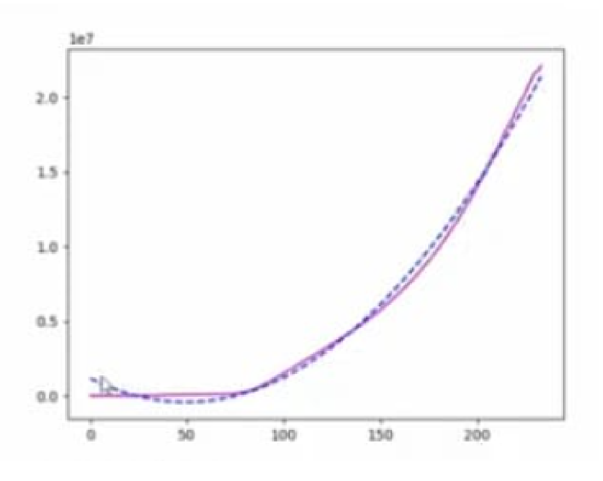


Figure 1. Polynomial i.e degree 2

- After prediction , will plot these values by – plt.plot(y0,'__b') and will show by plt.show(). We will remove previous plot because we want both to be on the same plot.

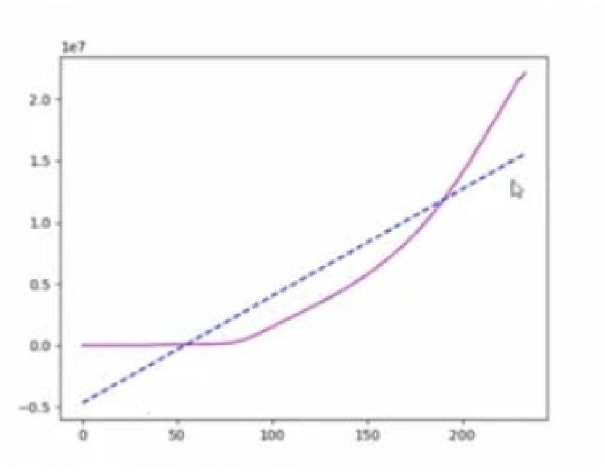


Figure 2. Linear i.e degree 1

- Now, it can be seen that it is trying to create a best fit line by using a single curvature and its not fitting very well (as some gaps can be seen). So, it can be found that 2nd order equation is not enough because it has only one curve in that case .
- now proceed to degree 3

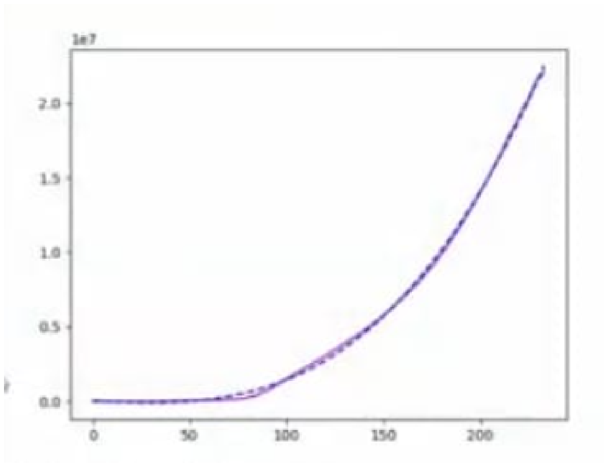


Figure 3. Polynomial i.e degree 3

- This will allow it to have two curves and on running this it can be seen that accuracy has increased (it went from 99.4

  So, it is a much better fit and follow all way long except only some gaps.

- This is basically the idea of polynomial regression.
- Take linear regression (degree=1)—
- On running this, accuracy decreases a lot and there are lots of gaps everywhere.
- And on adding degrees, it goes better and better.
- On trying with degree = 4

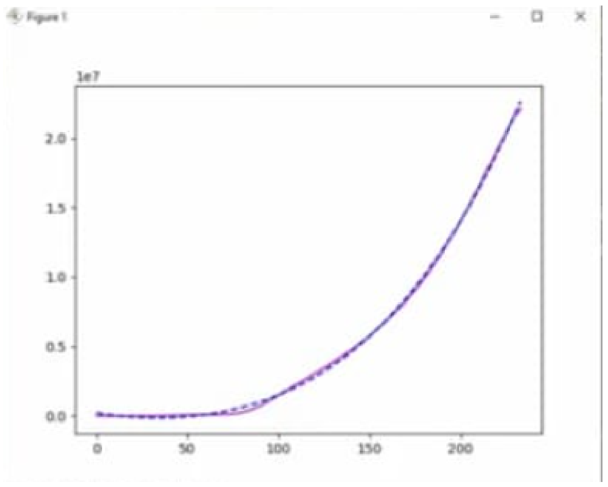


Figure 4. Polynomal of degree 4

- it will get even better and there will be very less gaps and very high accuracy = 99.925.
- The problem is in which degree we have to stop. We are running training data and not testing data, so we have to be careful because it might actually over fit the model and we don't want that and it won't be able to give good predictions for data for future.
- The next that can be done as, ask the model to predict what will be the cases after 2 days or 5 days or more . So the data we have already doesnot include the past remaining days. So, will use this model to predict the next two days and see how accurate it is.

7. Prediction

- Here, will write how many days we want for the prediction => days=".then write–

  print(f'Prediction_cases after days days:', end=")

  print(model.predict([[234+days]]))

- Write this in 2 square bracket form because this is the format required for prediction,=

print(round(int(model.predict (polyFeat.fit _transform([[234+days]])))/1000000,2), 'Million')

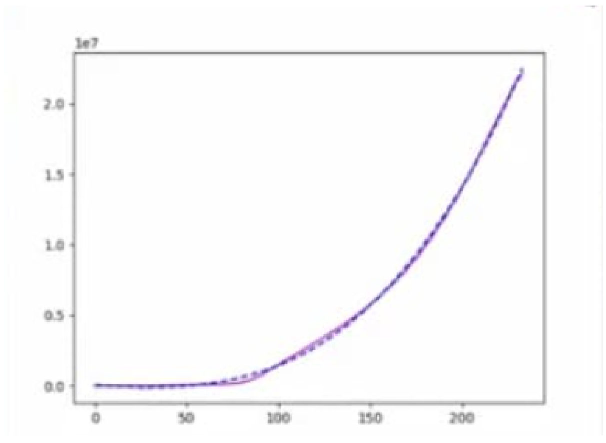


Figure 5. precision for 2 days

- where 2 denotes rounding off to 2 decimal places here.
- The prediction can be seen and also different values can be tried by changing the number of days taken.We will plot this to see this on graph.

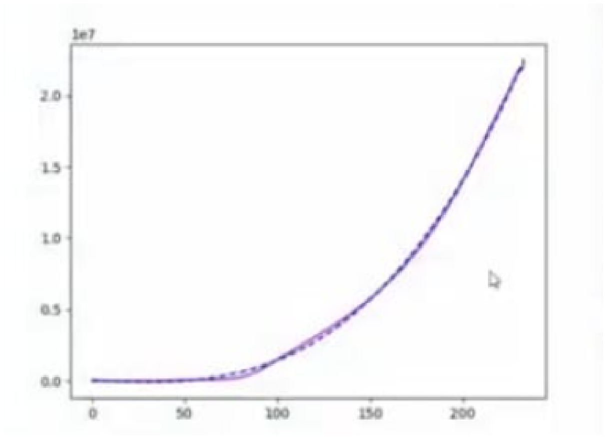


Figure 6. precision for 30 days

x1 = np.array( list(range(1,234+days))).reshape(-1,1)

y1 = model.predict(polyFeat.fit(x1))

- Fit transform is used because the feature needs to be extracted or calculated and then we plot using:

plt.plot(y1,'—r')

plt.plot(y0,'—b')

plt.show()

.

### 3.3 Data Exploration and Preprocessing

## 4 Result and Conclusion

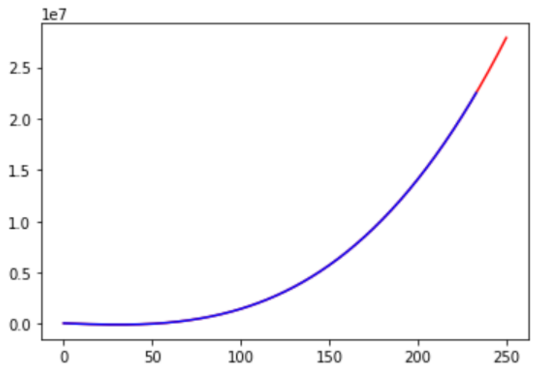
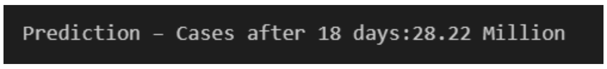


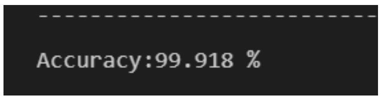
Figure 7. Final Output



```
Prediction - Cases after 18 days:28.22 Million
```

Figure 8. precision for 18 days

- In this paper we Have designed a covid case prediction model based on Polynomial regression
- Purple line here is the actual data and the dashed blue line is the best fit curve that the polynomial regression found and then the red part here is basically the prediction of the future (what is the number of cases and what is the trend).
- If we take small values in days then small changes can be seen .
- So, this is how, we can basically fit a non-linear data set. So using polynomial regression for non-linear relationships is quite useful and also the prediction is quite accirate.
- we got an accuracy of 99 percent that is a veri nice result.



```
-------------------------

Accuracy:99.918 %
```

Figure 9. Accuracy

## 5 Acknowledgement

We would like to Thank prof. Ritesh Kumar Mishra ( Prof. NIT Patna) who had been constantly supporting and mentoring us and also helped shape our report which helped in giving us much deeper insights into the course which eventually helped us in the completion of this report.

## References

[1] https*https://www.computervision.zone/*

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7324348/

[3] https://www.sciencedirect.com/science/article/pii/S2468042720300385