# Course Project on Supervised Machine Learning: Regression

**Subject: Regression model for prediction of the progress of vaccination against COVID**

## Main objective of the analysis

The objective of this analysis is to develop linear regression model for prediction of the progress of vaccination against COVID-19 in the world. This model will predict the number of people fully vaccinated against COVID in the world depending on date.

Fully vaccinated against COVID people are considered to be those who received all the doses of vaccines prescribed by the protocol of vaccination, but not necessary received booster vaccines.

## Brief description of the data set and a summary of its attributes

The data set used in this project is Data on COVID-19 (coronavirus) by Our World in Data (https://github.com/owid/covid-19-data/tree/master/public/data) which contains data for every country, for every continent and for the whole world. All the data is taken from official sources and updated regularly (most of data is updated daily).

- This data set contains:
- Confirmed cases and deaths (total confirmed cases of COVID-19, new confirmed cases of COVID-19, total deaths attributed to COVID-19, new deaths attributed to COVID-19, etc.)
- Hospitalizations and intensive care unit (ICU) admissions (number of COVID-19 patients in ICUs, number of COVID-19 patients in hospital, etc.)
- Testing for COVID-19 (total tests for COVID-19, new tests for COVID-19, etc.)
- Vaccinations against COVID-19 (total number of COVID-19 vaccination doses administered, total number of people who received at least one vaccine dose, total number of people who received all doses prescribed by the initial vaccination protocol, total number of COVID-19 vaccination booster doses administered (doses administered beyond the number prescribed by the vaccination protocol), etc.)
- Other variables (population, etc.)

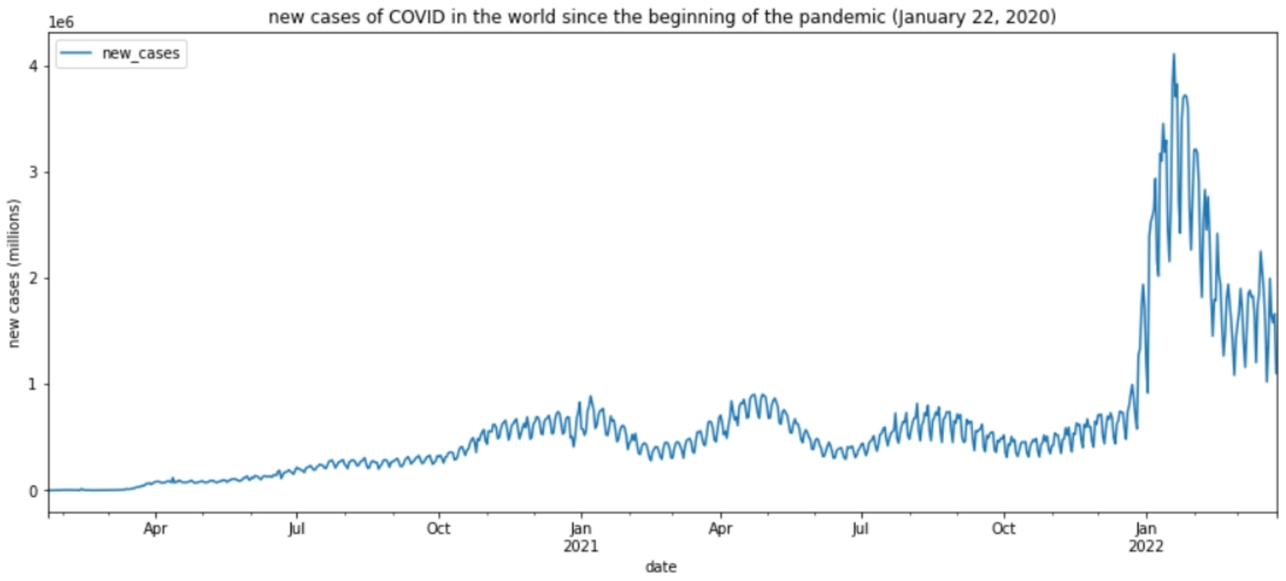
## Brief summary of data exploration and actions taken for data cleaning and feature engineering

Data exploration includes data collection, data cleaning, analysis, and feature engineering.

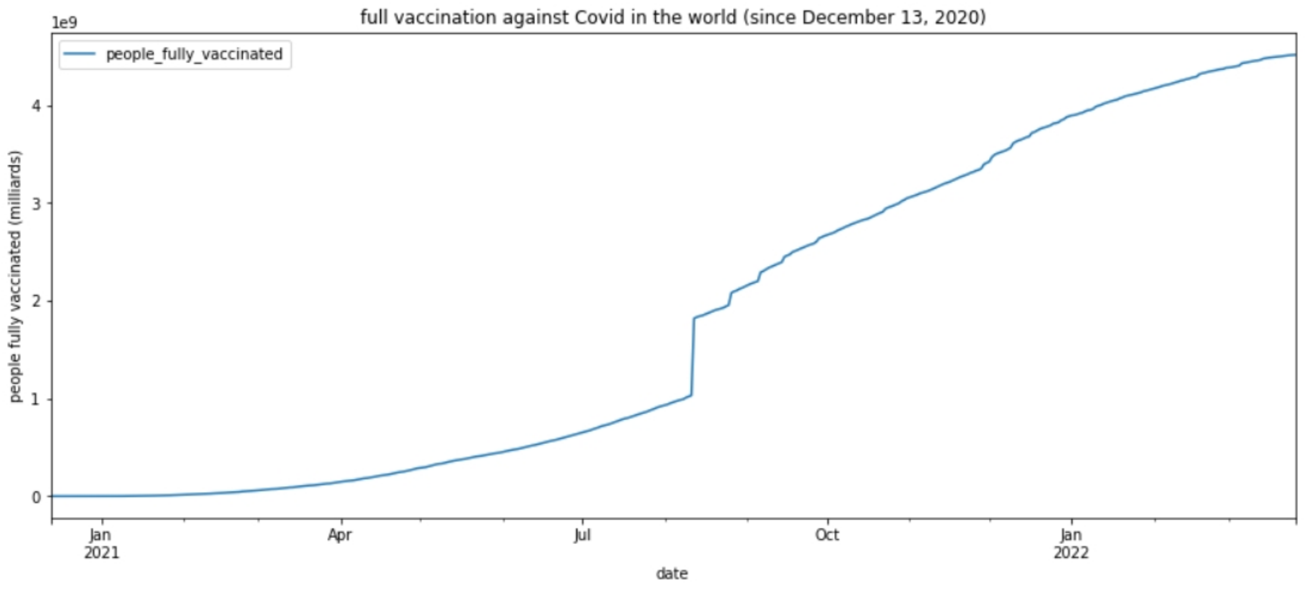
Data Collection. This step includes downloading data set from Internet and its examination.

Analysis. This includes drawing diagrams of dynamics of new cases of COVID and progress of vaccination.

1. The diagram of dynamics of new cases of COVID in the world since the beginning of the pandemic (that is, since January 22, 2020) looks like this:



2. The diagram of the progress of vaccination which shows the number of people in the world who were fully vaccinated is shown on the next page.



As can be seen in this diagram, there is one problem with data for fully vaccinated people. The dataset from Our World In Data gives the number of 1032423591 fully vaccinated people for 2021-08-11 and then 1820112099 for 2021-08-12, while the numbers before and after these dates grow smoothly. I could not fix this problem and used this data in my project as it is.

Data cleaning. The main problem with data in this dataset is that there are many NaNs. For this project, I created a new dataset. Also, I removed the rows that contained NaNs.

Feature engineering. For machine training, dates need to be transformed from object type into float64 type (date_delta). Other data in the dataset are already in float64 type and do not need transformation. Also, I created a new dataset for machine learning that contain only the number of fully vaccinated people, dates, and date_delta.

## Summary of training linear regression models

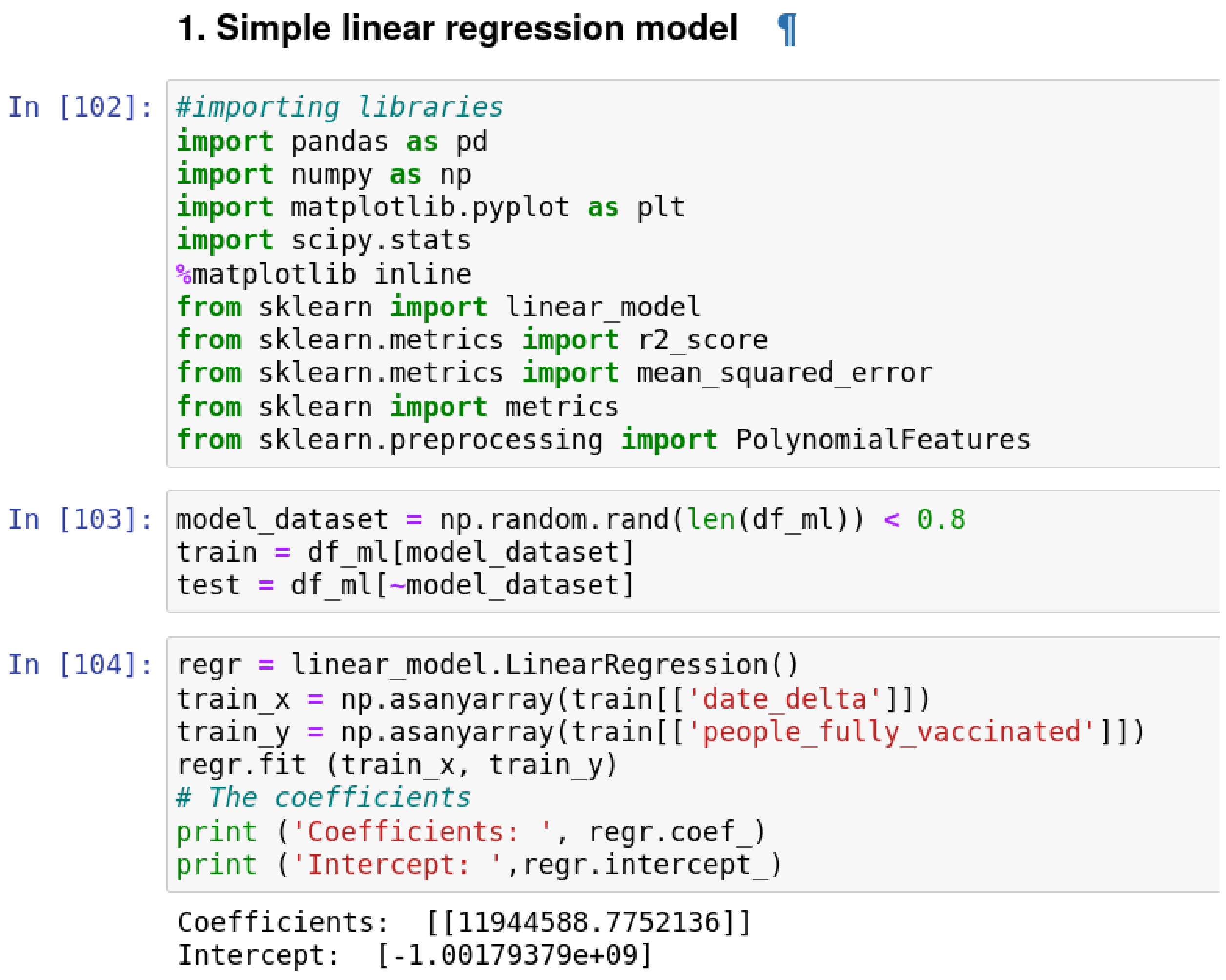
In this project, I developed and trained the following regression models:

- Simple linear regression
- Linear regression with 2-degree polynomial
- Linear regression with 5-degree polynomial

- Linear regression with 10-degree polynomial
- Linear regression with 10-degree polynomial and Ridge regularization
- Linear regression with 10-degree polynomial and LASSO regularization
- Linear regression with 10-degree polynomial and ElasticNet regularization

For all the models I used the same training and test splits.

For evaluation of all the models, I used two metrics – R2 score and RMSE.



## 1. Simple linear regression model ¶

```
In [102]: #importing libraries
          import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import scipy.stats
          %matplotlib inline
          from sklearn import linear_model
          from sklearn.metrics import r2_score
          from sklearn.metrics import mean_squared_error
          from sklearn import metrics
          from sklearn.preprocessing import PolynomialFeatures
```

```
In [103]: model_dataset = np.random.rand(len(df_ml)) < 0.8
          train = df_ml[model_dataset]
          test = df_ml[~model_dataset]
```

```
In [104]: regr = linear_model.LinearRegression()
          train_x = np.asanyarray(train[['date_delta']])
          train_y = np.asanyarray(train[['people_fully_vaccinated']])
          regr.fit (train_x, train_y)
          # The coefficients
          print ('Coefficients: ', regr.coef_)
          print ('Intercept: ',regr.intercept_)

          Coefficients:  [[11944588.7752136]]
          Intercept:  [-1.00179379e+09]
```

```
In [105]: test_x = np.asanyarray(test[['date_delta']])
          test_y = np.asanyarray(test[['people_fully_vaccinated']])
          test_y_ = regr.predict(test_x)

          metrics.r2_score(test_y, test_y_)

Out[105]: 0.9395020923756772
```

```
In [106]: #mean_squared_error with squared=False returns RMSE
          #(root mean squared error)
          mean_squared_error(test_y, test_y_, squared=False)

Out[106]: 412677177.85102105
```

## 2. Regression models with polynomial features

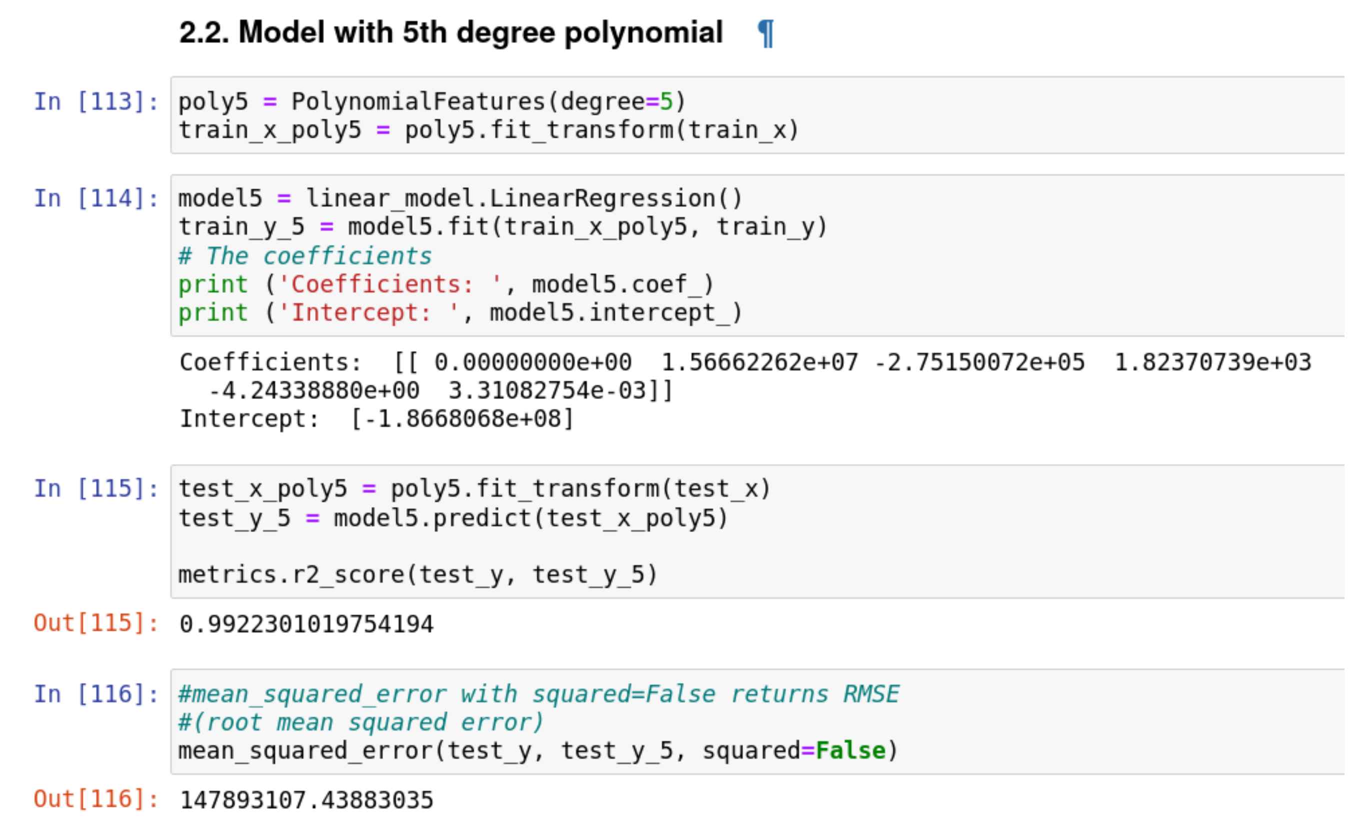
### 2.1. Model with 2nd degree polynomial

```
In [107]: poly2 = PolynomialFeatures(degree=2)
          train_x_poly2 = poly2.fit_transform(train_x)
```

```
In [108]: model2 = linear_model.LinearRegression()
          train_y_2 = model2.fit(train_x_poly2, train_y)
          # The coefficients
          print ('Coefficients: ', model2.coef_)
          print ('Intercept: ', model2.intercept_)

          Coefficients:  [[       0.         3394011.44276129   18201.35440881]]
          Intercept:  [-3.26153974e+08]
```

```
In [110]: test_x_poly2 = poly2.fit_transform(test_x)
          test_y_2 = model2.predict(test_x_poly2)

          metrics.r2_score(test_y, test_y_2)

Out[110]: 0.9606546969370706
```

```
In [111]: #mean_squared_error with squared=False returns RMSE
          #(root mean squared error)
          mean_squared_error(test_y, test_y_2, squared=False)

Out[111]: 332802608.87779015
```



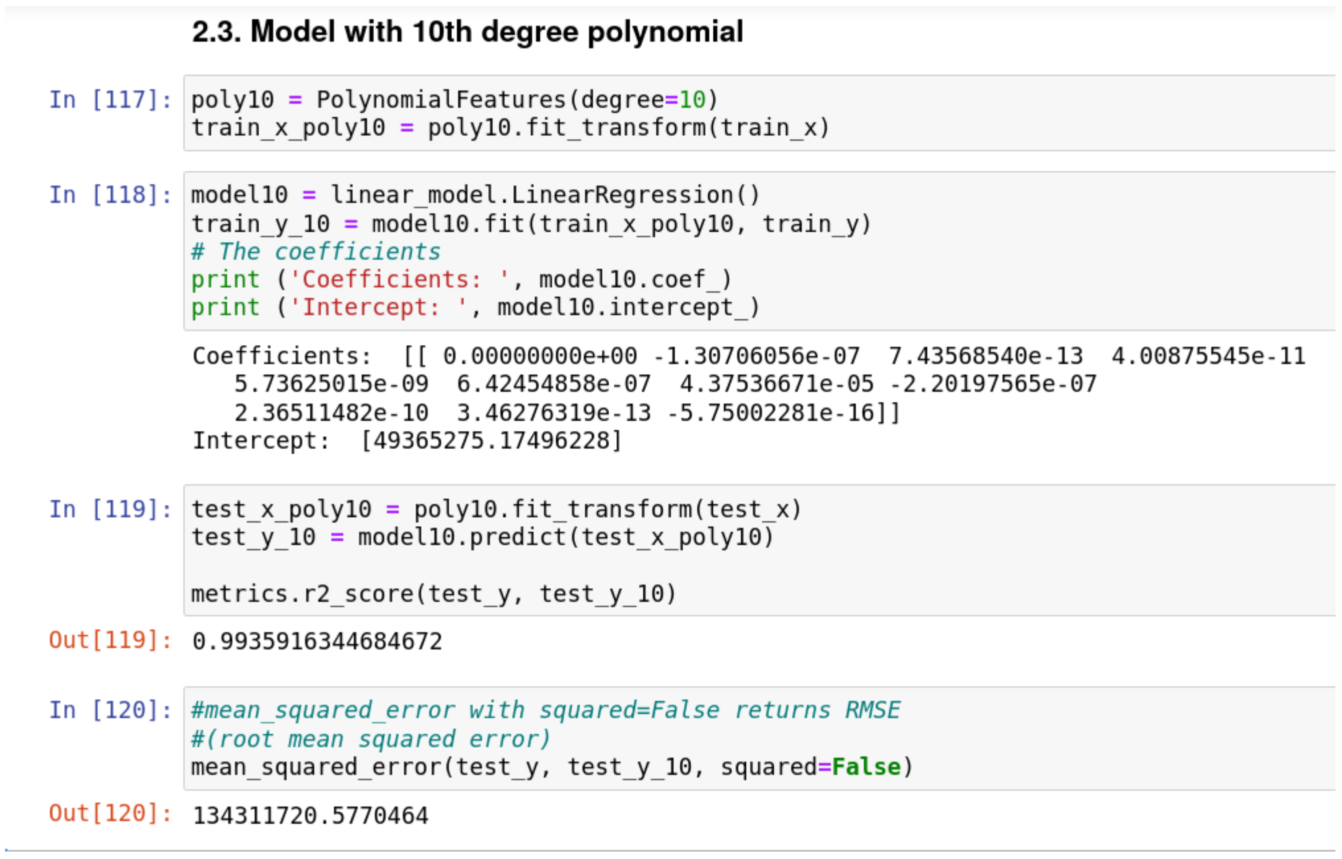
## 2.2. Model with 5th degree polynomial ¶

```
In [113]: poly5 = PolynomialFeatures(degree=5)
          train_x_poly5 = poly5.fit_transform(train_x)
```

```
In [114]: model5 = linear_model.LinearRegression()
          train_y_5 = model5.fit(train_x_poly5, train_y)
          # The coefficients
          print ('Coefficients: ', model5.coef_)
          print ('Intercept: ', model5.intercept_)

          Coefficients:  [[ 0.00000000e+00  1.56662262e+07 -2.75150072e+05  1.82370739e+03
            -4.24338880e+00  3.31082754e-03]]
          Intercept:  [-1.8668068e+08]
```

```
In [115]: test_x_poly5 = poly5.fit_transform(test_x)
          test_y_5 = model5.predict(test_x_poly5)

          metrics.r2_score(test_y, test_y_5)

Out[115]: 0.9922301019754194
```

```
In [116]: #mean_squared_error with squared=False returns RMSE
          #(root mean squared error)
          mean_squared_error(test_y, test_y_5, squared=False)

Out[116]: 147893107.43883035
```



## 2.3. Model with 10th degree polynomial

```
In [117]: poly10 = PolynomialFeatures(degree=10)
          train_x_poly10 = poly10.fit_transform(train_x)
```

```
In [118]: model10 = linear_model.LinearRegression()
          train_y_10 = model10.fit(train_x_poly10, train_y)
          # The coefficients
          print ('Coefficients: ', model10.coef_)
          print ('Intercept: ', model10.intercept_)

          Coefficients:  [[ 0.00000000e+00 -1.30706056e-07  7.43568540e-13  4.00875545e-11
             5.73625015e-09  6.42454858e-07  4.37536671e-05 -2.20197565e-07
             2.36511482e-10  3.46276319e-13 -5.75002281e-16]]
          Intercept:  [49365275.17496228]
```

```
In [119]: test_x_poly10 = poly10.fit_transform(test_x)
          test_y_10 = model10.predict(test_x_poly10)

          metrics.r2_score(test_y, test_y_10)

Out[119]: 0.9935916344684672
```

```
In [120]: #mean_squared_error with squared=False returns RMSE
          #(root mean squared error)
          mean_squared_error(test_y, test_y_10, squared=False)

Out[120]: 134311720.5770464
```

## 3. Regularization

```
In [145]: from sklearn.linear_model import Ridge
          from sklearn.linear_model import Lasso
          from sklearn.linear_model import ElasticNet
```

### 3.1. Ridge regression

```
In [146]: ridge = Ridge(alpha=80).fit(train_x_poly10, train_y)
```

```
In [147]: test_y_ridge = ridge.predict(test_x_poly10)
          print(r2_score(test_y, test_y_ridge))

          0.9955601401464049
```

```
In [148]: mean_squared_error(test_y, test_y_ridge, squared=False)

Out[148]: 111795650.0316397
```

### 3.2. Lasso regression

```
In [154]: lasso = Lasso(alpha=0.0005).fit(train_x_poly10, train_y)
```

```
In [155]: test_y_lasso = lasso.predict(test_x_poly10)
          print(r2_score(test_y, test_y_lasso))

          0.9902882163970492
```

```
In [156]: mean_squared_error(test_y, test_y_lasso, squared=False)

Out[156]: 165344507.6045687
```

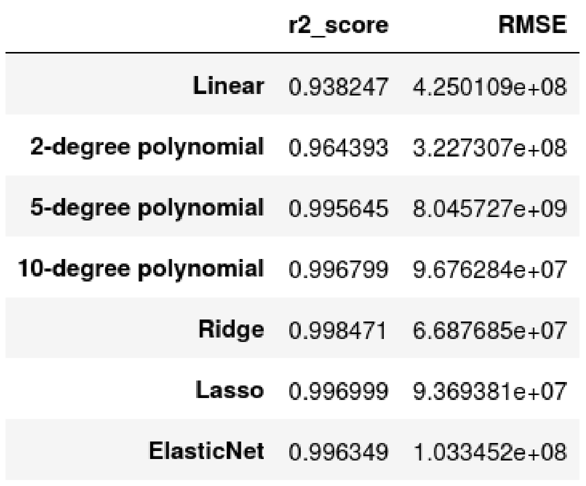
### 3.3. ElasticNet regression

```
In [157]: elasticnet = ElasticNet(alpha=1e-05, l1_ratio=0.9).fit(train_x_poly10, train_y)
```

```
In [158]: test_y_elasticnet = elasticnet.predict(test_x_poly10)
          print(r2_score(test_y, test_y_elasticnet))

          0.9902882164060266
```

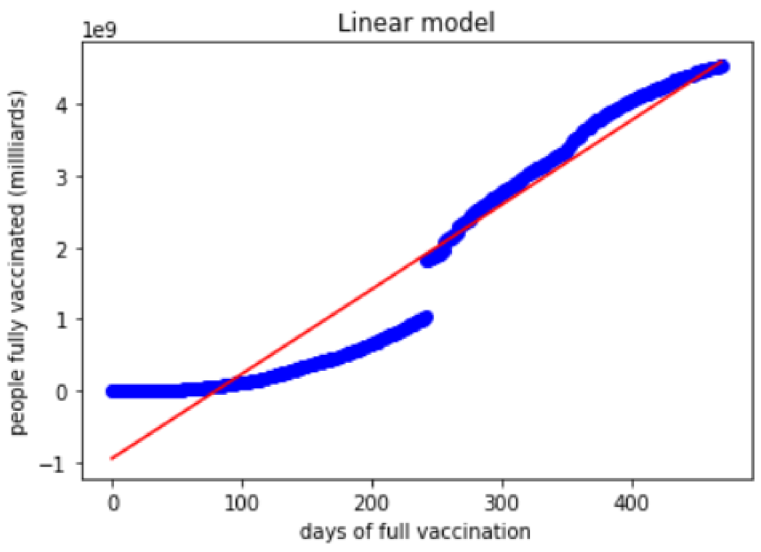
```
In [159]: mean_squared_error(test_y, test_y_elasticnet, squared=False)

Out[159]: 165344507.52814832
```

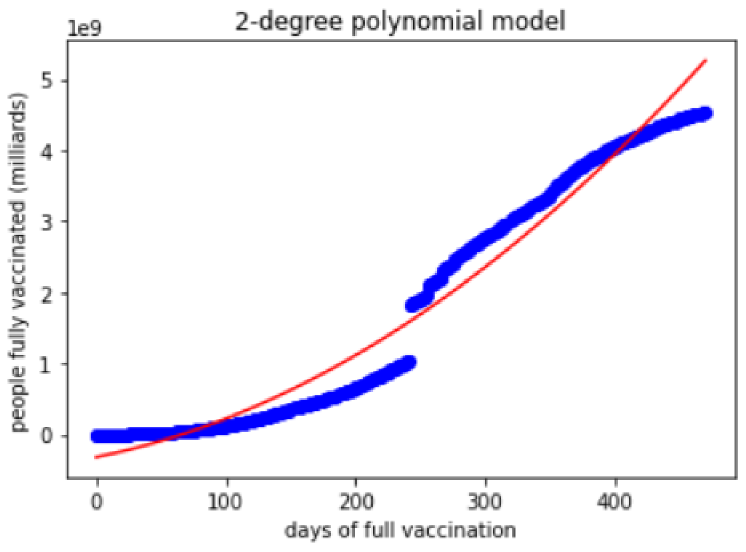
Dataset of evaluation of all models:

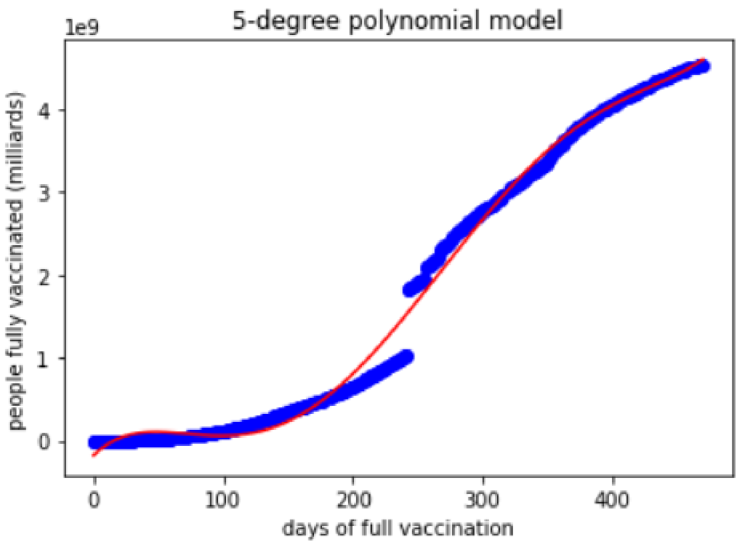


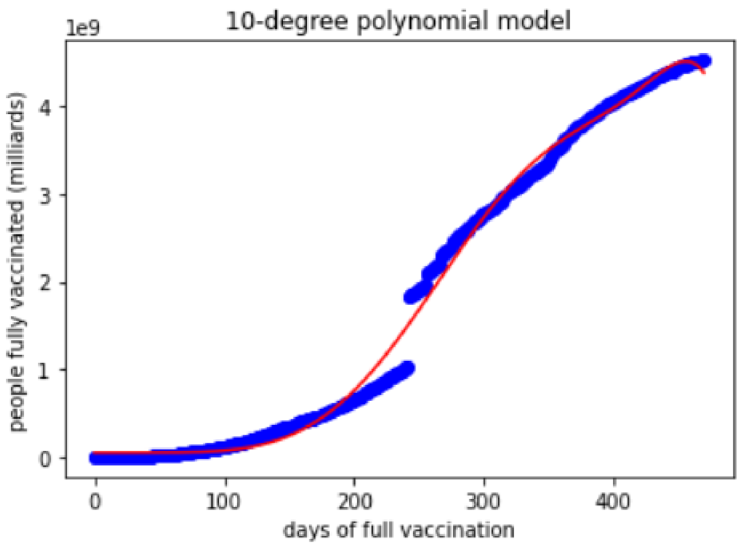
| | r2_score | RMSE |
|---|---|---|
| **Linear** | 0.938247 | 4.250109e+08 |
| **2-degree polynomial** | 0.964393 | 3.227307e+08 |
| **5-degree polynomial** | 0.995645 | 8.045727e+09 |
| **10-degree polynomial** | 0.996799 | 9.676284e+07 |
| **Ridge** | 0.998471 | 6.687685e+07 |
| **Lasso** | 0.996999 | 9.369381e+07 |
| **ElasticNet** | 0.996349 | 1.033452e+08 |

I also drew plots for some of the models:

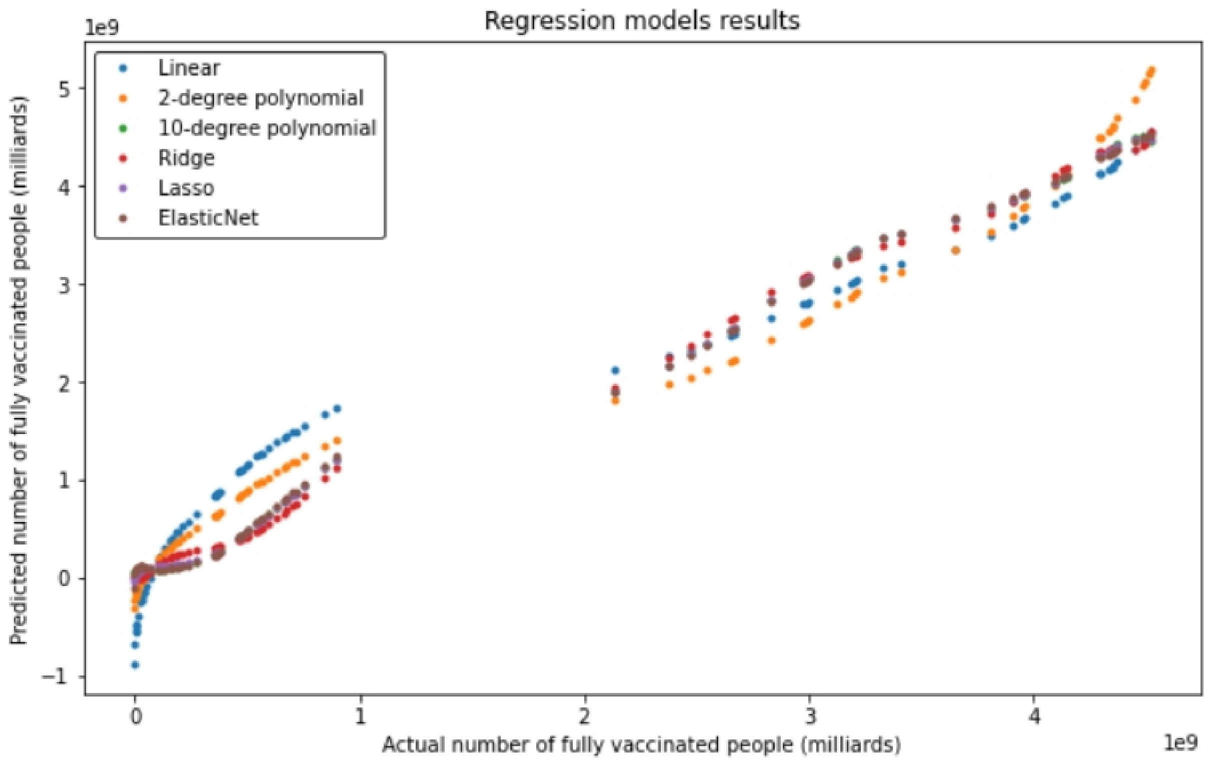






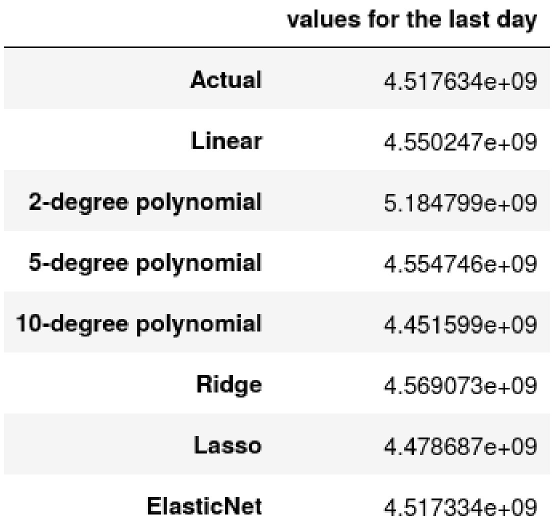


The plot of actual vs predicted numbers of fully vaccinated people:



This plot has breaks between 1 and 2 milliards because the original dataset does not contain these numbers.

Since these models are used for prediction, the most important is that they should correctly predict the number of fully vaccinated people in future. So, I also calculated the actual number of fully vaccinated people and those predicted by all the models for the last date used in the test set:



| | values for the last day |
|---|---|
| Actual | 4.517634e+09 |
| Linear | 4.550247e+09 |
| 2-degree polynomial | 5.184799e+09 |
| 5-degree polynomial | 4.554746e+09 |
| 10-degree polynomial | 4.451599e+09 |
| Ridge | 4.569073e+09 |
| Lasso | 4.478687e+09 |
| ElasticNet | 4.517334e+09 |

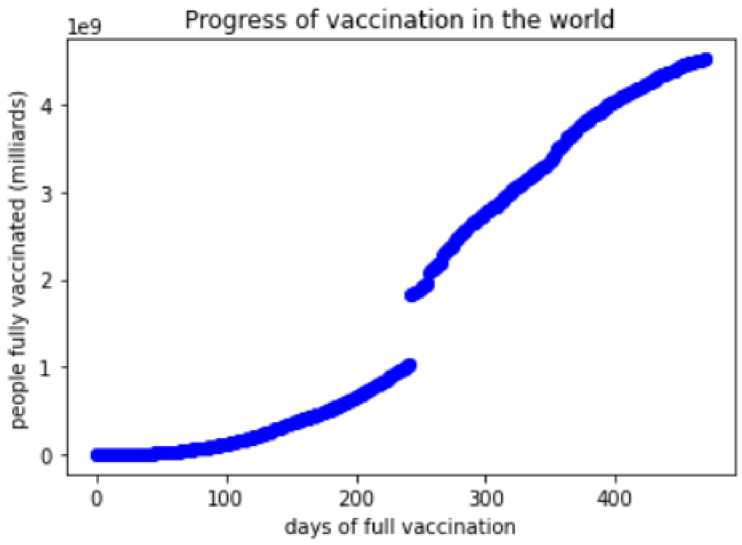
### Recommended final model

Among the 7 models used in this project, the highest r2 score and the lowest RMSE has the model with Ridge regression. Therefore, this model gives the best overall results according to both metrics. However, since the model is likely to be used to predict the progress of vaccination in the world, that is the numbers of fully vaccinated people in future, the model with ElasticNet regression might be more accurate because it gives the best prediction for the last date in the test set.

### Summary Key Findings and Insights

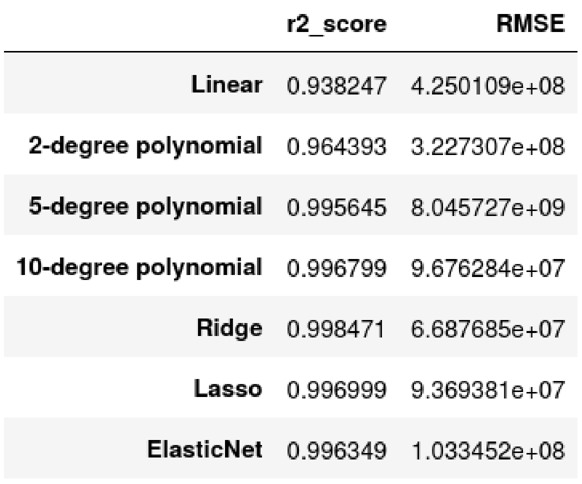
The goal of this project was to develop a regression model that can be used in order to predict the process of vaccination in the world, that is, the number people who received the prescribed doses of vaccine, but not necessary received booster vaccination.

As I already mentioned, the original dataset has a problem with data of fully vaccinated people. It gives the number of 1032423591 fully vaccinated people for 2021-08-11 and then 1820112099 for 2021-08-12, while the numbers before and after these dates grow smoothly. Therefore, the actual data look like this:



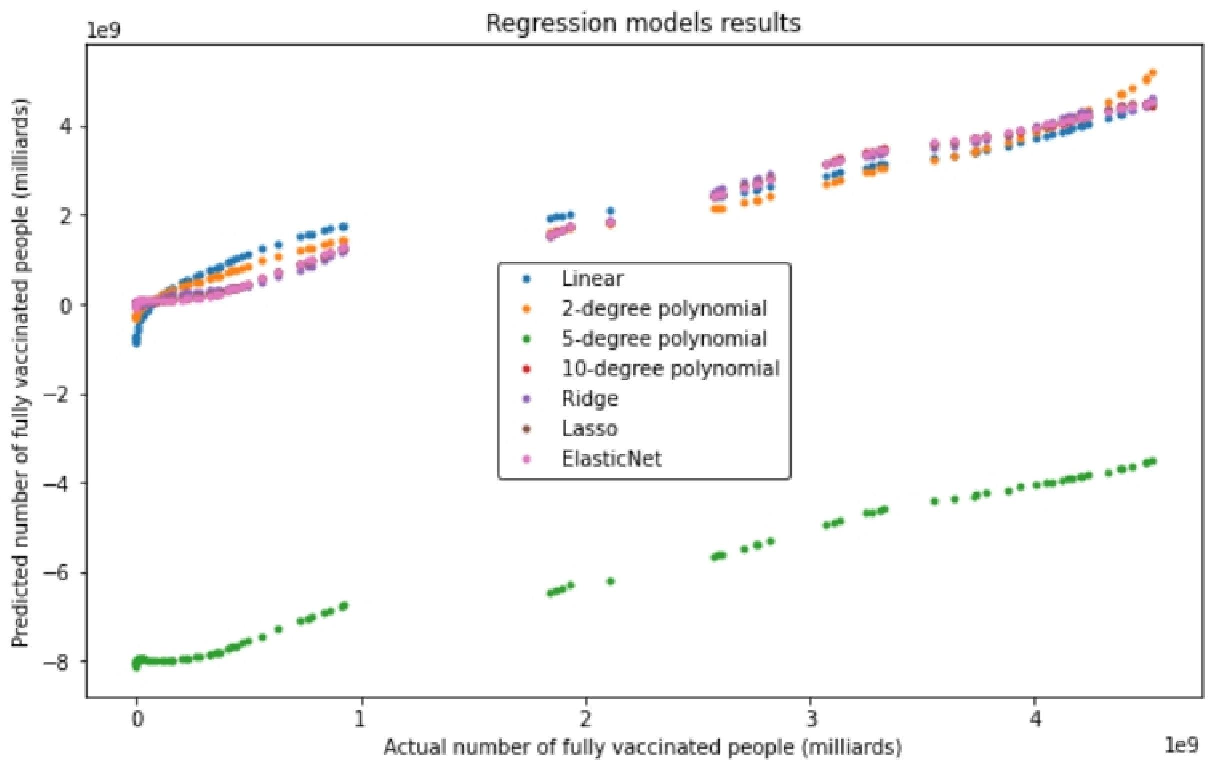
This makes it very hard to find a good function for approximation of this diagram. Therefore, all the regression models will have significant errors in the middle of this diagram.

In this project, I used simple linear regression, linear regressions with polynomials of different degrees (2, 5, and 10), and linear regressions with 10-degree polynomial and different methods of regularization (Ridge, LASSO, and ElasticNet). I used the same training and test splits for all the models and also I used the same metrics – R2 score and RMSE. The best model according to both metrics was the model with Ridge regression:



| | r2_score | RMSE |
|---|---|---|
| **Linear** | 0.938247 | 4.250109e+08 |
| **2-degree polynomial** | 0.964393 | 3.227307e+08 |
| **5-degree polynomial** | 0.995645 | 8.045727e+09 |
| **10-degree polynomial** | 0.996799 | 9.676284e+07 |
| **Ridge** | 0.998471 | 6.687685e+07 |
| **Lasso** | 0.996999 | 9.369381e+07 |
| **ElasticNet** | 0.996349 | 1.033452e+08 |

The plot of actual vs predicted numbers of fully vaccinated people:



(This plot has breaks between 1 and 2 milliards because the original dataset does not contain these numbers.)

This plot shows that 5-degree polynomial model works much worse than other models and gives very inaccurate predictions. Other models work much better. Linear model gives inaccurate predictions for small values, while 2-degree model gives inaccurate predictions for higher values. 10-degree polynomial models without regularization and with various kinds of regularization (Ridge, Lasso, and ElasticNet) give better results, and their results are very similar.

## Suggestions for next steps in analyzing this data

Since the number of people who got vaccinated is increasing every day, it may be useful to revisit this model and update it with new data. This will increase the accuracy of this model in future.

Also, it may be useful to try to tune the hyper-parameters of this model, such as the degree of polynomial to see if it would be possible to make a better model.

Another way to improve the model accuracy might be to try using other kinds of functions for regression instead of linear and polynomial.

But it seems that the most important thing is to regularly update the model with new data after they are released because new data might significantly affect the model coefficients and accuracy.