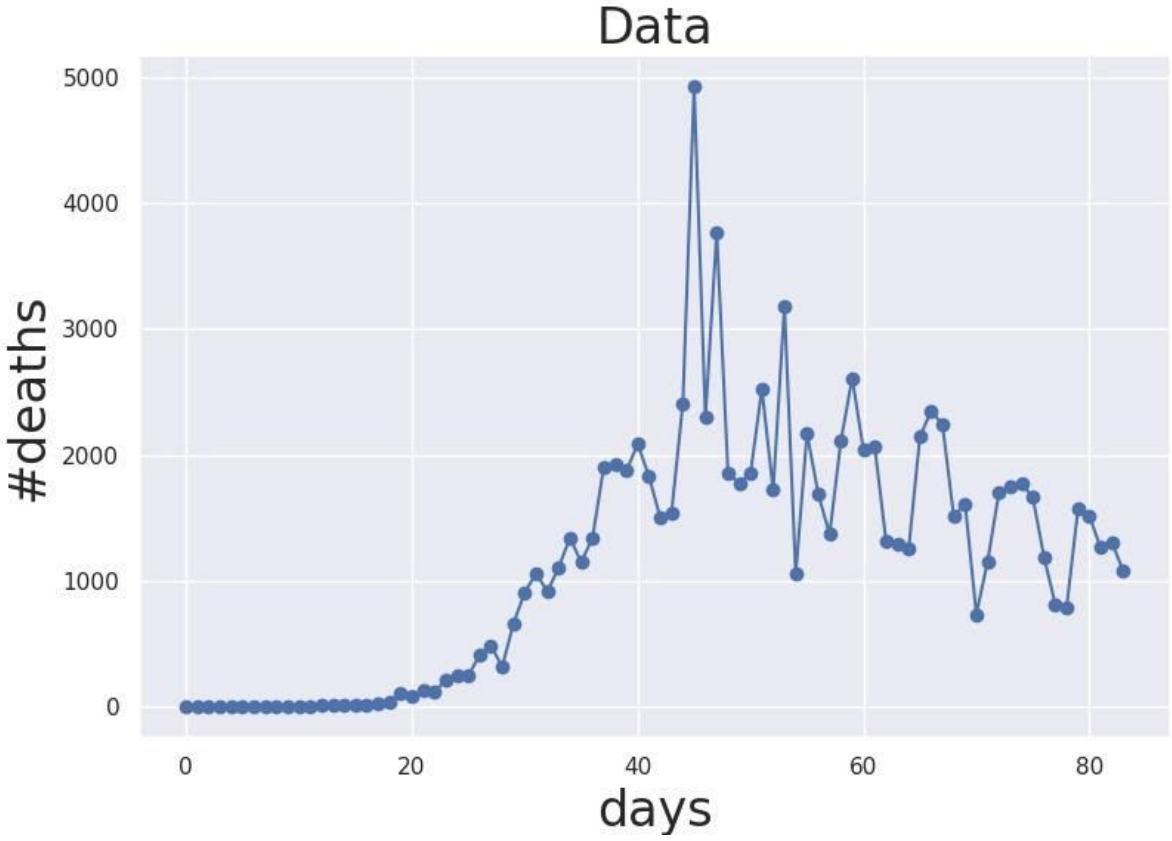
Time Progression of COVID Epidemic in the United States

Final report for Cogs 109
By Maziyar Mirzaei, Spring 2020
Course instructor: Megan Bardolph, Ph.D.

Background

The dataset is the data on the geographic distribution of COVID-19 cases worldwide, freely provided by the European Center for Disease Prevention and Control [1]. The dataset provides the number of cases and deaths for every day since the beginning of the COVID-19 epidemic for nearly every country in the world.

We analyze only a small part of this dataset: the number of deaths in the United States (variables: days since the beginning of the epidemic and the number of deaths). We would like to develop a model that a) describes how the number of deaths is changing, and b) predicts the number of deaths in the following days.



Methods

Our method of choice is linear regression. We model the number of deaths as a polynomial in the day number since the beginning of the epidemic in the Unites States. In order to choose the best model, we divide the dataset into training and testing datasets, fit models of different order to the training dataset using the python lstsq function, and then choose the model that has the smallest sum-of-squares error when applied to the testing dataset.

Results

Models

We use the following models:

model 1: $y = w_0 + w_1x$

model 2: $y = w_0 + w_1x + w_2x^2$

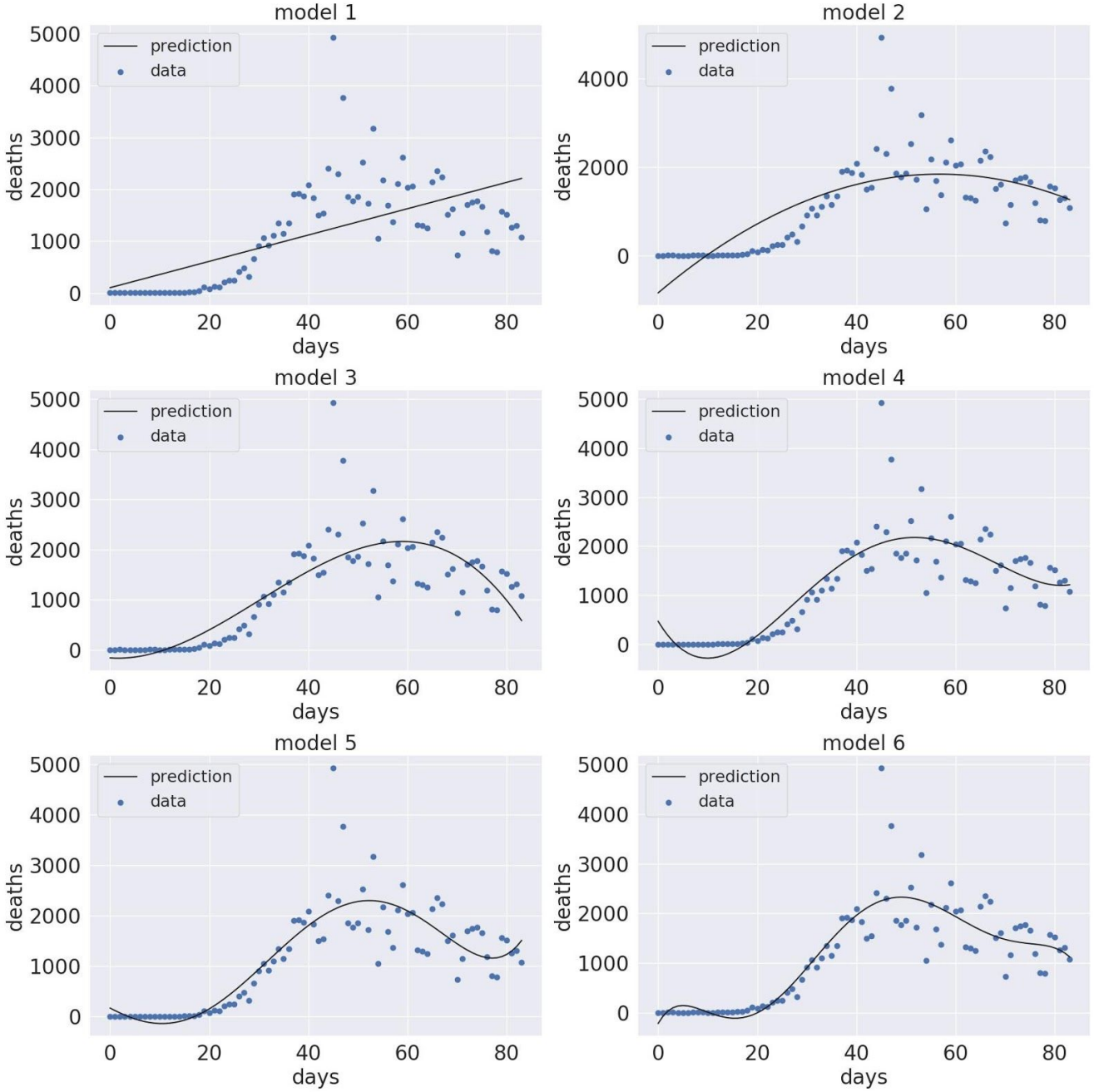
model 3: $y = w_0 + w_1x + w_2x^2 + w_3x^3$

model 4: $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$

model 5: $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5$

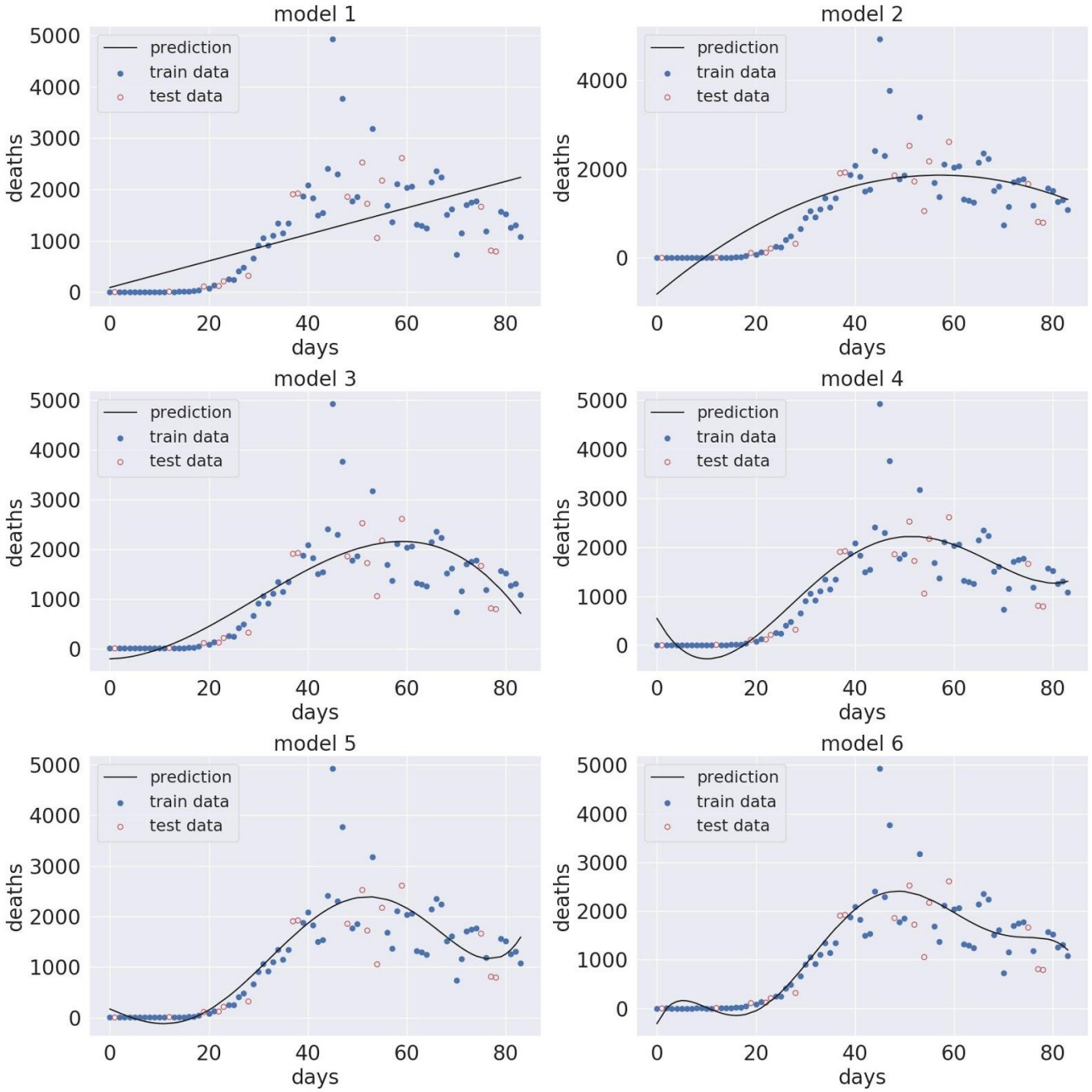
model 6: $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$

The results of fitting these models to the data are shown below



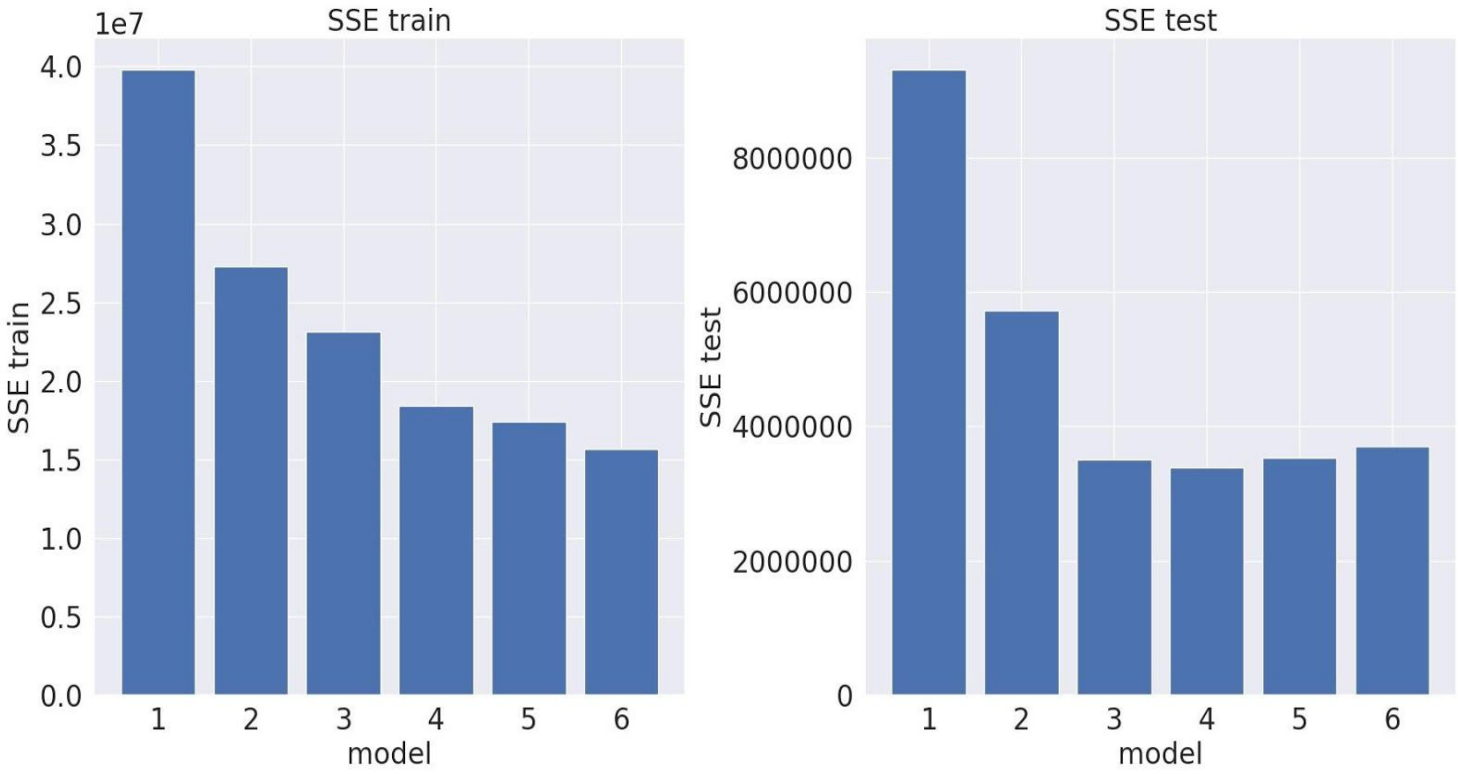
Cross-validation

We chose 60% of the data as the training dataset, using python np.random.choice function. The rest of the data are considered as the testing dataset. We then fit the models to the training dataset and calculate sum-of-squares error (SSE) for the training dataset and for the testing dataset. The fitted models are shown below.



Sum-of-squares error

The sum-of-squares error for the training and the testing datasets also shown below.

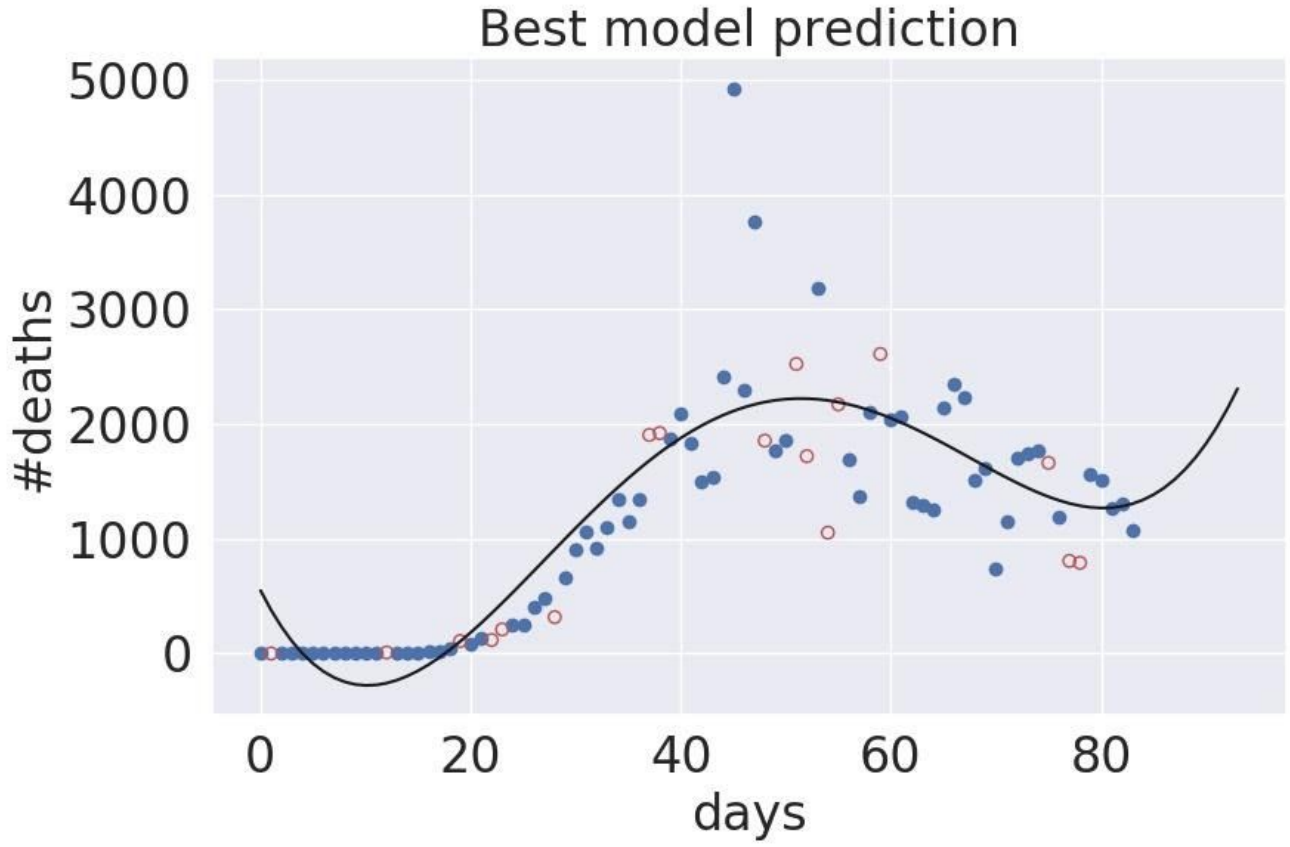


SSE for the training dataset decreases when we increase the number of terms in the model, whereas SSE for the testing dataset first decreases but then again increases. We take the model with the smallest SSE for the training dataset as the best model. It is model 4: $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$, with coefficients:

w_0 = 5.49553767e+02, w_1 = -1.81412529e+02, w_2 = 1.18101461e+01, w_3 = -2.04319169e-01, w_4 = 1.08053760e-03

Predicting future deaths

In order to see whether we can predict future development of the epidemic, we plot the best model for more days after the last data point



The predictions does not seem reasonable.

Discussion

We used linear regression to fit the data for daily deaths caused by the COVID-19 epidemic in the United States. By testing several models and using cross-validation we showed that linear, quadratic and even cubic models do not explain the data sufficiently well. The fourth order model fits the data the best. However, this model behave strange in the beginning of the epidemics and its predictions for the future numbers of deaths seem counter-intuitive. We conclude that linear regression is not a good model for analyzing the epidemic. Indeed, real epidemiological models are very complex, requiring fitting solutions to differential equations, see [2].

References

1. European Center for Disease Prevention and Control, https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-distribution-covid-19-cases-worldwide

2. Wikipedia: Compartamental models in epidemiology, https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology#Elaborations_on_the_basic_SIR_model

Appendix: Python code

```
#!/usr/bin/env python

from __future__ import division
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

fs = 24
np.random.seed(137)
if __name__ == "__main__":
    '''
    Development of COVID epidemics in USA
    '''
    plt.ioff()
    plt.close('all')

    #import data as pandas dataframe
    df = pd.read_csv('./qyBVfsy2.csv')


    #select data for US and sort them in chronological order
    data = df[df['countriesAndTerritories'] == 'United_States_of_America'].sort_values(['year',
'month', 'day'])

    #select the columns with the number of deaths
    Y = data['deaths'].to_numpy()
    Y = Y[Y>0] # select only the days where the number of deaths is greater than zero

    #define X-variable
    n = len(Y)
    X = np.arange(n)

    #plot number of cases and number of days day by day
    plt.figure(figsize = (9,6))
    plt.plot(X, Y, 'o-')
    plt.xlabel('days', fontsize = fs)
    plt.ylabel('#deaths', fontsize = fs)
    plt.rc('xtick', labelsize = fs)
    plt.rc('ytick', labelsize = fs)
    plt.title('Data', fontsize = fs)
    plt.savefig('./deaths.jpg')
    plt.close()
```

```
#fitting different models
mmax = 6
sse = []
fig, axs = plt.subplots(mmax//2, 2, figsize = (18, 18))
for j in range(mmax):
    m = j+1 #order of the model
    A = np.ones((n, m+1))
    for k in range(1, m+1):
        A[:,k] = X**k

    #estimating the model
    W = np.linalg.lstsq(A, Y)[0]

    #evaluating predictions
    Y_pred = A.dot(W)

    #sum of squares errors
    SSE = np.sum((Y - Y_pred)**2)
    sse.append(SSE)
    print('m =', m, ' SSE =', SSE)

    i1 = j//2
    i2 = j%2
    axs[i1,i2].scatter(X, Y, c = 'b', label = 'data')
    axs[i1,i2].plot(X, Y_pred, c = 'k', label = 'prediction')
    axs[i1,i2].set_xlabel('days', fontsize = fs)
    axs[i1,i2].set_ylabel('deaths', fontsize = fs)
    axs[i1,i2].set_title(f'model {m}', fontsize = fs)
    axs[i1, i2].legend(loc = 2, fontsize = 0.8*fs)

plt.tight_layout()
plt.savefig('./models.jpg')
plt.close()


#cross-validation: split dataset into train and test datasets in proportion 60/40
#train dataset
idx_train = np.sort(np.random.choice(X, int(.8*n), replace = False))
X_train = X[idx_train]
Y_train = Y[idx_train]
n_train = len(idx_train)

#test dataset: everything that didn't enter the train dataset
idx_test = np.array([i for i in X if i not in idx_train])
X_test = X[idx_test]
Y_test = Y[idx_test]
```

```
n_test = len(idx_test)


#fitting with different models
sse_train = []
sse_test = []
fig, axs = plt.subplots(mmax//2, 2, figsize = (18, 18))
for j in range(mmax):
    m = j+1 #order of the model
    A_train = np.ones((n_train, m+1))
    A_test = np.ones((n_test, m+1))
    for k in range(1, m+1):
        A_train[:,k] = X_train**k
        A_test[:,k] = X_test**k

    #estimating the model
    W = np.linalg.lstsq(A_train, Y_train)[0]

    #evaluating predictions
    Y_train_pred = A_train.dot(W)
    Y_test_pred = A_test.dot(W)

    #sum of squares errors
    SSE_train = np.sum((Y_train - Y_train_pred)**2)
    SSE_test = np.sum((Y_test - Y_test_pred)**2)
    print('m =', m, ' SSE_train =', SSE_train, ' SSE_test =', SSE_test)
    sse_train.append(SSE_train)
    sse_test.append(SSE_test)

    i1 = j//2
    i2 = j%2
    axs[i1, i2].scatter(X_train, Y_train, c = 'b', label = 'train data')
    axs[i1, i2].scatter(X_test, Y_test, edgecolors = 'r', facecolors = 'none', label = 'test data')
    axs[i1, i2].plot(X_train, Y_train_pred, c = 'k', label = 'prediction')
    axs[i1, i2].set_xlabel('days', fontsize = fs)
    axs[i1, i2].set_ylabel('deaths', fontsize = fs)
    axs[i1, i2].set_title(f'model {m}', fontsize = fs)
    axs[i1, i2].legend(loc = 2, fontsize = .8*fs)

plt.tight_layout()
plt.savefig('./cross-validation.jpg')
plt.close()

mm = range(1, mmax+1)
fig, axs = plt.subplots(1, 2, figsize = (18, 9))
axs[0].bar(mm, sse_train)
axs[0].set_xlabel('model', fontsize = fs)
```

```
    axs[0].set_ylabel('SSE train', fontsize = fs)
    axs[0].set_xticks(mm)
    axs[0].set_title('SSE train', fontsize = fs)

    axs[1].bar(mm, sse_test)
    axs[1].set_xlabel('model', fontsize = fs)
    axs[1].set_ylabel('SSE test', fontsize = fs)
    axs[1].set_xticks(mm)
    axs[1].set_title('SSE test', fontsize = fs)

    plt.tight_layout()
    plt.savefig('./SSE.jpg')
    plt.close()

    #the sum of squares error for the training set dicreases as the number of model parameters increases;
    # however the error for the test set first decreases and then increases: the best model is with m = 4


    #best model : y = w0 + w1*x + w2*x**2 + w3*x**3 + w4*x**4
    m = 4
    A_train = np.ones((n_train, m+1))
    A_test = np.ones((n_test, m+1))
    X_ext = np.arange(len(X) + 10)
    A = np.ones((len(X_ext), m+1))
    for k in range(1, m+1):
        A_train[:,k] = X_train**k
        A_test[:,k] = X_test**k
        A[:,k] = X_ext**k
    W = np.linalg.lstsq(A_train, Y_train)[0]
    # Y_train_pred = A_train.dot(W)
    # Y_test_pred = A_test.dot(W)
    Y_ext_pred = A.dot(W)

    plt.figure(figsize = (9, 6))
    plt.scatter(X_train, Y_train, c = 'b', label = 'train data')
    plt.scatter(X_test, Y_test, edgecolors = 'r', facecolors = 'none', label = 'test data')
    plt.plot(X_ext, Y_ext_pred, c= 'k')
    plt.xlabel('days', fontsize = fs)
    plt.ylabel('#deaths', fontsize = fs)
    plt.title('Best model prediction', fontsize = fs)
    plt.tight_layout()
    plt.savefig('./best_model.jpg')
    plt.close()
```

#although this model is the best among the ones that we tested, it still noticeably deviates from the data,

#particularly in the beginning, where it doe snot correctly represent the straight line.

#This is because representation in terms of powers is not suitable for this data