```
import requests
import json
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import pandas as pd
import random
import math
import time
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error
import datetime
import operator
from datetime import datetime
from dateutil.parser import parse
plt.style.use('fivethirtyeight')
%matplotlib inline


population = pd.read_csv('population_india_census2011.csv')


import json
from pandas.io.json import json_normalize
```

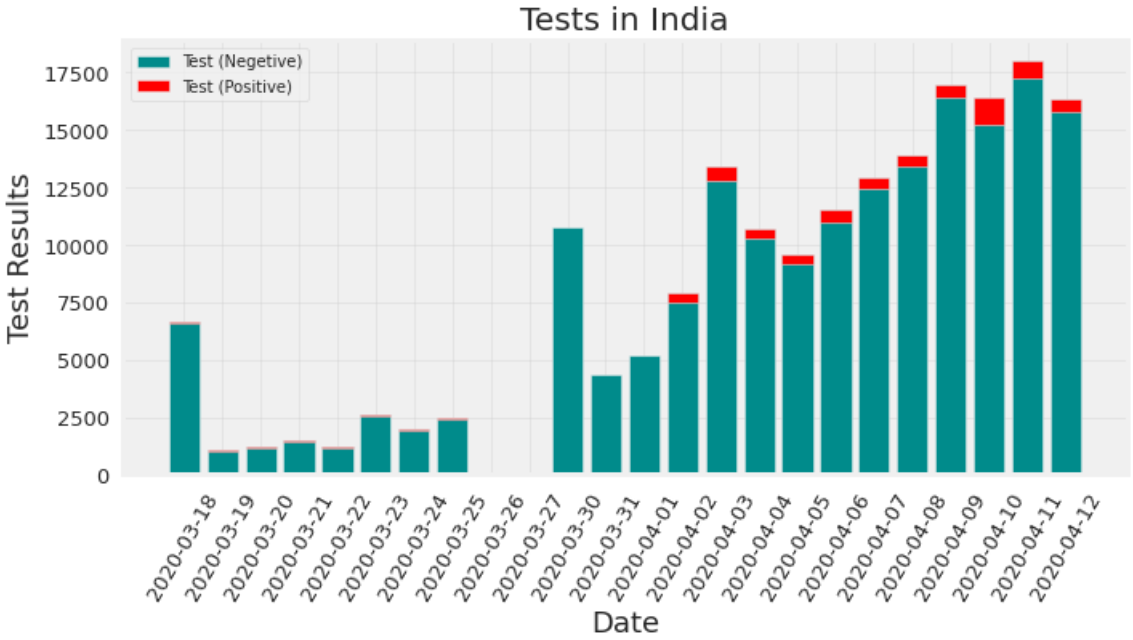
General Stats of India:

```
df_india_test = pd.io.json.json_normalize(requests.get('https://api.rootnet.in/covid19-in/stats/testing/h
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning: pandas.io.json.json_n
  """Entry point for launching an IPython kernel.
```

```
df_india_test["p2t_ratio"]= np.round(100*df_india_test["c_positive"]/df_india_test["c_tests"],2)
df_india_test["positive"] = df_india_test["c_positive"].diff()
df_india_test["tests"] = df_india_test["c_tests"].diff()
df_india_test["p2t_ratio"]= np.round(100*df_india_test["positive"]/df_india_test["tests"],2)
df_india_test = df_india_test[1:]


f = plt.figure(figsize=(10,5))
f.add_subplot(111)

plt.axes(axisbelow=True)
plt.bar(df_india_test["day"],df_india_test["tests"].values[:],color="darkcyan",label="Test (Negetive)"+st
plt.bar(df_india_test["day"],df_india_test["positive"].values[:],bottom=df_india_test["tests"].values[:]-
plt.tick_params(size=5,labelsize = 13)
plt.tick_params(axis="x",size=5,labelsize = 13,labelrotation=60 )
plt.xlabel("Date",fontsize=18)
plt.ylabel("Test Results",fontsize=18)

plt.title("Tests in India",fontsize=20)
plt.grid(alpha=0.3)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f343b745470>
```




```
india_data_json = requests.get('https://api.rootnet.in/covid19-in/unofficial/covid19india.org/statewise'
df_india = pd.io.json.json_normalize(india_data_json['data']['statewise'])
df_india = df_india.set_index("state")
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarning: pandas.io.json.json_n
```

```
total = df_india.sum()
total.name = "Total"
pd.DataFrame(total).transpose().style.background_gradient(cmap='Wistia',axis=1)
```

	confirmed	recovered	deaths	active
Total	9211	1086	331	7794

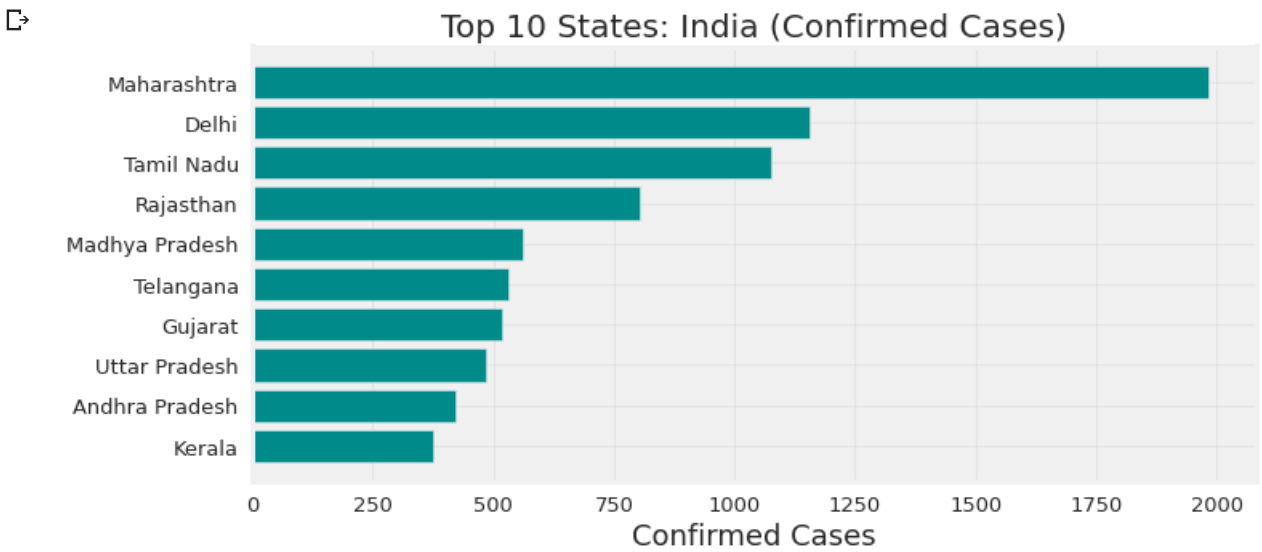
```
df_india.style.background_gradient(cmap='Wistia')
```

state	confirmed	recovered	deaths	active
Maharashtra	1982	217	149	1616
Delhi	1154	28	24	1102
Tamil Nadu	1075	50	11	1014
Rajasthan	804	121	11	672
Madhya Pradesh	562	41	43	478
Telangana	531	103	16	412
Gujarat	516	44	24	448
Uttar Pradesh	483	45	5	433
Andhra Pradesh	420	12	7	401
Kerala	375	179	2	194
Jammu and Kashmir	245	6	4	235
Karnataka	232	54	6	172
Haryana	195	44	3	148
Punjab	170	23	12	135
West Bengal	134	19	7	108
Bihar	64	26	1	37
Odisha	54	12	1	41
Uttarakhand	35	5	0	30
Himachal Pradesh	32	12	2	18
Assam	29	0	1	28
Chhattisgarh	31	10	0	21
Chandigarh	21	7	0	14
Jharkhand	19	0	2	17
Ladakh	15	11	0	4
Andaman and Nicobar Islands	11	10	0	1
Goa	7	5	0	2
Puducherry	7	1	0	6
Manipur	2	1	0	1
Tripura	2	0	0	2
Mizoram	1	0	0	1
Arunachal Pradesh	1	0	0	1
Dadra and Nagar Haveli	1	0	0	1
Nagaland	1	0	0	1
Daman and Diu	0	0	0	0
Lakshadweep	0	0	0	0
Meghalaya	0	0	0	0
Sikkim	0	0	0	0

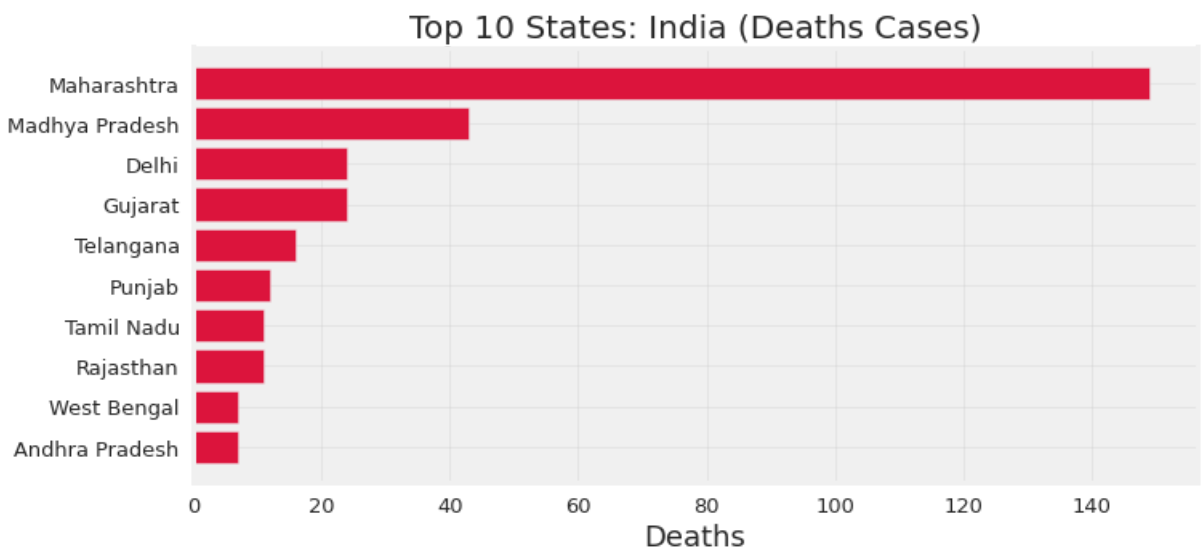
```
df_india[df_india['deaths'] > 0].style.background_gradient(cmap='Wistia')
```

state	confirmed	recovered	deaths	active
Maharashtra	1982	217	149	1616
Delhi	1154	28	24	1102
Tamil Nadu	1075	50	11	1014
Rajasthan	804	121	11	672
Madhya Pradesh	562	41	43	478
Telangana	531	103	16	412
Gujarat	516	44	24	448
Uttar Pradesh	483	45	5	433
Andhra Pradesh	420	12	7	401
Kerala	375	179	2	194
Jammu and Kashmir	245	6	4	235
Karnataka	232	54	6	172
Haryana	195	44	3	148
Punjab	170	23	12	135
West Bengal	134	19	7	108
Bihar	64	26	1	37
Odisha	54	12	1	41
Himachal Pradesh	32	12	2	18
Assam	29	0	1	28
Jharkhand	19	0	2	17

```
f = plt.figure(figsize=(10,5))
f.add_subplot(111)

plt.axes(axisbelow=True)
plt.barh(df_india.sort_values('confirmed')["confirmed"].index[-10:],df_india.sort_values('confirmed')["c
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Confirmed Cases",fontsize=18)
plt.title("Top 10 States: India (Confirmed Cases)",fontsize=20)
plt.grid(alpha=0.3)
#plt.savefig(out+'Top 10 States_India (Confirmed Cases).png')
```




```
f = plt.figure(figsize=(10,5))
f.add_subplot(111)

plt.axes(axisbelow=True)
plt.barh(df_india.sort_values('deaths')["deaths"].index[-10:],df_india.sort_values('deaths')["deaths"].va
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Deaths",fontsize=18)
plt.title("Top 10 States: India (Deaths Cases)",fontsize=20)
plt.grid(alpha=0.3)
#plt.savefig(out+'Top 10 States_India (Deaths Cases).png')
```



Merging Data (States, Hospitals, Police officers, Population, Gener Ratio):

```
police = pd.read_excel('/content/datafile.xls')
```

```
WARNING *** OLE2 inconsistency: SSCS size is 0 but SSAT size is non-zero
```

```
merged_df = population.merge(police, how = 'inner', right_on='State/UT', left_on='State / Union Territory
merged_df.drop(['Sno','State / Union Territory','Sl.No'], axis = 1, inplace=True)
```

```
merged_df = merged_df.fillna(0)
```

```
merged_df['Total Police'] = merged_df['Sanctioned strength of Police Personnel']
```

```
cols = ['Total Police per lakh of population - Sanctioned','Total Police per lakh of population - Actual
```

```
state_features = merged_df
```

```
india_data_unofficial = requests.get('https://api.rootnet.in/covid19-in/unofficial/covid19india.org').jso
raw_patient_data = pd.io.json.json_normalize(india_data_unofficial['data']['rawPatientData'])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarning: pandas.io.json.json_n
```

```
raw_patient_data = raw_patient_data.drop(['contractedFrom','nationality','place_attributes','relationshi
```

```
hospitals_unofficial = requests.get('https://api.rootnet.in/covid19-in/stats/hospitals').json()
hospital_data = pd.io.json.json_normalize(hospitals_unofficial['data']['regional'])
```

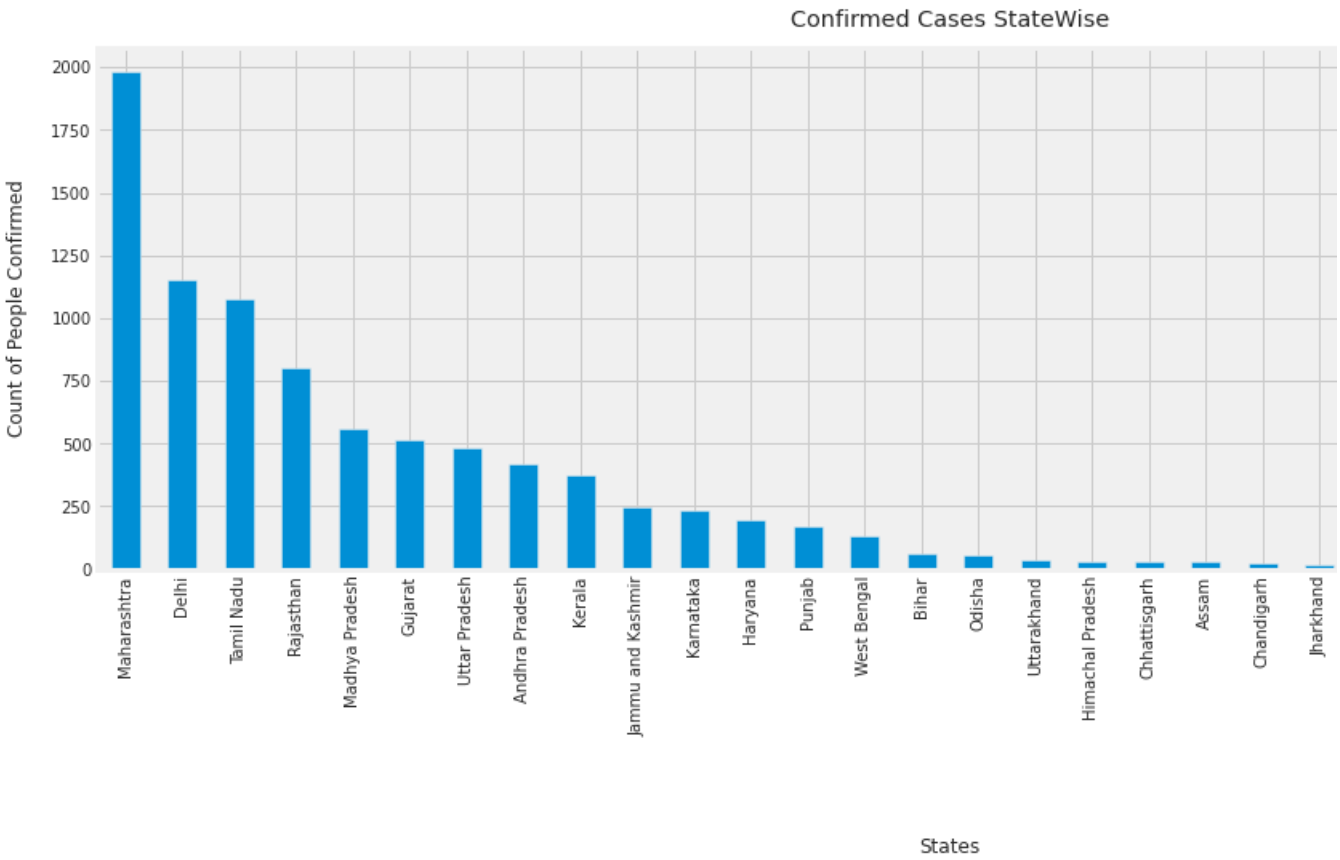
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarning: pandas.io.json.json_n
```

```
hospital_data['state'] = hospital_data['state'].str.replace('&', 'and')
```

```
merged_df_individual = merged_df.merge(raw_patient_data, how = 'inner', left_on='State/UT', right_on='sta
```

```
merged_df_individual = merged_df_individual.merge(hospital_data, how = 'inner', left_on = 'State/UT', ri
```

```
merged_df_individual['State/UT'].value_counts().plot(kind='bar', figsize=(17, 6))
plt.xlabel("States", labelpad=14)
plt.ylabel("Count of People Confirmed", labelpad=14)
plt.title("Confirmed Cases StateWise", y=1.02)
plt.show()
```

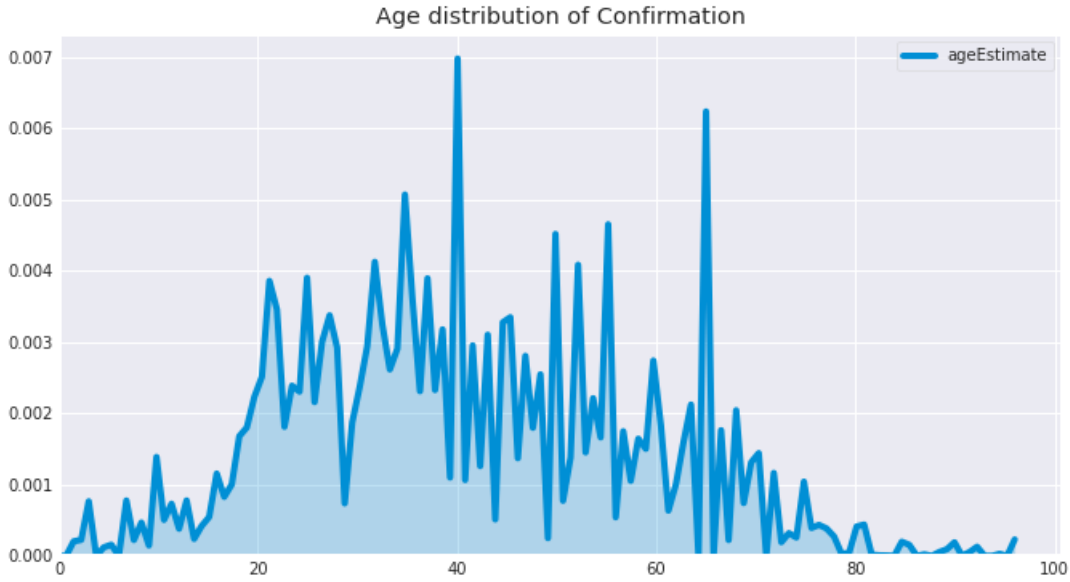



```
merged_df_individual.drop(cols, axis =1 , inplace=True)
```

```
merged_df_individual.drop(['asOn','notes','state_x','state_y'], axis=1, inplace=True)
```

```
merged_df_individual['ageEstimate'].fillna(0)
merged_df_individual['ageEstimate'] = merged_df_individual['ageEstimate'].replace('28-35','30')
merged_df_individual['ageEstimate'] = merged_df_individual['ageEstimate'].replace('','0')
#merged_df_individual['ageEstimate'].astype(int)
plt.figure(figsize=(10,6))
sns.set_style("darkgrid")
plt.title("Age distribution of Confirmation")
sns.kdeplot(data=merged_df_individual['ageEstimate'].astype(float), shade=True).set(xlim=(0))
```

```
[(0.0, 100.8)]
```




```
merged_df_individual['reportedOn'] = pd.to_datetime(merged_df_individual['reportedOn'], dayfirst=True)

merged_df_individual = merged_df_individual.sort_values(by='reportedOn')

state_features = state_features.merge(hospital_data, how = 'inner', left_on='State/UT', right_on='state'

statewise_unofficial = requests.get('https://api.rootnet.in/covid19-in/unofficial/covid19india.org/statev
statewise_patient_data = pd.io.json.json_normalize(statewise_unofficial['data']['history'])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarning: pandas.io.json.json_n
```

```
cols_date_state = statewise_patient_data['day'],statewise_patient_data['statewise']
date_state = pd.DataFrame(cols_date_state)

date_state = date_state.T
date_state
```

	day	statewise
0	2020-03-14	[{'state': 'Kerala', 'confirmed': 19, 'recover...
1	2020-03-15	[{'state': 'Maharashtra', 'confirmed': 31, 're...
2	2020-03-16	[{'state': 'Maharashtra', 'confirmed': 38, 're...
3	2020-03-17	[{'state': 'Maharashtra', 'confirmed': 41, 're...
4	2020-03-18	[{'state': 'Maharashtra', 'confirmed': 43, 're...
5	2020-03-19	[{'state': 'Maharashtra', 'confirmed': 49, 're...
6	2020-03-20	[{'state': 'Maharashtra', 'confirmed': 52, 're...
7	2020-03-21	[{'state': 'Maharashtra', 'confirmed': 64, 're...
8	2020-03-22	[{'state': 'Maharashtra', 'confirmed': 74, 're...
9	2020-03-23	[{'state': 'Maharashtra', 'confirmed': 97, 're...
10	2020-03-24	[{'state': 'Kerala', 'confirmed': 109, 'recove...
11	2020-03-25	[{'state': 'Kerala', 'confirmed': 118, 'recove...
12	2020-03-26	[{'state': 'Kerala', 'confirmed': 137, 'recove...
13	2020-03-27	[{'state': 'Kerala', 'confirmed': 176, 'recove...
14	2020-03-28	[{'state': 'Kerala', 'confirmed': 182, 'recove...
15	2020-03-29	[{'state': 'Kerala', 'confirmed': 202, 'recove...
16	2020-03-30	[{'state': 'Maharashtra', 'confirmed': 238, 'r...
17	2020-03-31	[{'state': 'Maharashtra', 'confirmed': 302, 'r...
18	2020-04-01	[{'state': 'Maharashtra', 'confirmed': 335, 'r...
19	2020-04-02	[{'state': 'Maharashtra', 'confirmed': 423, 'r...
20	2020-04-03	[{'state': 'Maharashtra', 'confirmed': 490, 'r...
21	2020-04-04	[{'state': 'Maharashtra', 'confirmed': 635, 'r...
22	2020-04-05	[{'state': 'Maharashtra', 'confirmed': 748, 'r...
23	2020-04-06	[{'state': 'Maharashtra', 'confirmed': 868, 'r...
24	2020-04-07	[{'state': 'Maharashtra', 'confirmed': 1018, '...
25	2020-04-08	[{'state': 'Maharashtra', 'confirmed': 1135, '...
26	2020-04-09	[{'state': 'Maharashtra', 'confirmed': 1364, '...
27	2020-04-10	[{'state': 'Maharashtra', 'confirmed': 1574, '...
28	2020-04-11	[{'state': 'Maharashtra', 'confirmed': 1761, '...
29	2020-04-12	[{'state': 'Maharashtra', 'confirmed': 1982, '...

```
date_state['state'] = date_state['statewise'].to_dict()
```

```
output = pd.DataFrame()
j = 0
for i in range(1,len(date_state['day'])):
  output = output.append(date_state['statewise'][i], ignore_index= False)
print(output.head())
```

```
          state  confirmed  recovered  deaths  active
0   Maharashtra         31          0       0      31
1        Kerala         24          3       0      21
2       Haryana         14          0       0      14
3  Uttar Pradesh         12          4       0       8
4         Delhi          7          2       1       4
```

```
output.reset_index(inplace=True)
```

```
output['diff'] = output['index'].diff()
```

```
output
```

	index	state	confirmed	recovered	deaths	active	diff
0	0	Maharashtra	31	0	0	31	NaN
1	1	Kerala	24	3	0	21	1.0
2	2	Haryana	14	0	0	14	1.0
3	3	Uttar Pradesh	12	4	0	8	1.0
4	4	Delhi	7	2	1	4	1.0
...	...	...	...	...	...	...	...
1052	32	Nagaland	1	0	0	1	1.0
1053	33	Daman and Diu	0	0	0	0	1.0
1054	34	Lakshadweep	0	0	0	0	1.0
1055	35	Meghalaya	0	0	0	0	1.0
1056	36	Sikkim	0	0	0	0	1.0

1057 rows × 7 columns

```
output['day'] = [0]*len(output)
```

```
j=0
for i,v in output.iterrows():
  output['day'][i] = date_state['day'][j]
  if output['diff'][i] <= 0:
    j += 1
    output['day'][i] = date_state['day'][j]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:671: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
  self._setitem_with_indexer(indexer, value)
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
```

```
date_time_final = output
date_time_final['day'] = pd.to_datetime(date_time_final['day'], dayfirst=True)
```

```
date_time_final
```

	index	state	confirmed	recovered	deaths	active	diff	day
0	0	Maharashtra	31	0	0	31	NaN	2020-03-14
1	1	Kerala	24	3	0	21	1.0	2020-03-14
2	2	Haryana	14	0	0	14	1.0	2020-03-14
3	3	Uttar Pradesh	12	4	0	8	1.0	2020-03-14
4	4	Delhi	7	2	1	4	1.0	2020-03-14
...	...	...	...	...	...	...	...	...
1052	32	Nagaland	1	0	0	1	1.0	2020-04-11
1053	33	Daman and Diu	0	0	0	0	1.0	2020-04-11
1054	34	Lakshadweep	0	0	0	0	1.0	2020-04-11
1055	35	Meghalaya	0	0	0	0	1.0	2020-04-11
1056	36	Sikkim	0	0	0	0	1.0	2020-04-11

1057 rows × 8 columns

```
date_time_final_try = date_time_final.merge(state_features, how = 'left', left_on='state', right_on='Stat

date_time_final_try
```

	index	state_x	confirmed	recovered	deaths	active	diff	day	Population	Rural population	po
0	0	Maharashtra	31	0	0	31	NaN	2020-03-14	112374333.0	61556074.0	5
1	1	Kerala	24	3	0	21	1.0	2020-03-14	33406061.0	17471135.0	1
2	2	Haryana	14	0	0	14	1.0	2020-03-14	25351462.0	16509359.0	
3	3	Uttar Pradesh	12	4	0	8	1.0	2020-03-14	199812341.0	155317278.0	4
4	4	Delhi	7	2	1	4	1.0	2020-03-14	16787941.0	419042.0	1
...	...	...	...	...	...	...	...	...	...	...	
1052	32	Nagaland	1	0	0	1	1.0	2020-04-11	1978502.0	1407536.0	
1053	33	Daman and Diu	0	0	0	0	1.0	2020-04-11	NaN	NaN	
1054	34	Lakshadweep	0	0	0	0	1.0	2020-04-11	64473.0	14141.0	
1055	35	Meghalaya	0	0	0	0	1.0	2020-04-11	2966889.0	2371439.0	
1056	36	Sikkim	0	0	0	0	1.0	2020-04-11	610577.0	456999.0	

1057 rows × 27 columns

```
date_time_final_try['confirm_time_diff_statewise'] = date_time_final_try.groupby(['state_x'])['confirmed
date_time_final_try['death_time_diff_statewise'] = date_time_final_try.groupby(['state_x'])['deaths'].di
```

```
date_time_final_try['confirm_time_diff_statewise'].sum()
```

```
9211.0
```

```
date_time_final_try = date_time_final_try.fillna(0)
```

```
date_time_final_try[date_time_final_try['State/UT'] == 'Rajasthan']['confirm_time_diff_statewise'].sum()
```

```
804.0
```

```
ls_final_drop = ['index','state_x','active','diff','Total Police per lakh of population - Sanctioned','To
te_time_final_try.drop(cols_final_drop, inplace=True, axis = 1)
```

```
date_time_final_try['Area'] = date_time_final_try['Area'].str.split(n=1).str[0]
date_time_final_try['Area'] = date_time_final_try['Area'].str.replace(',','')
date_time_final_try['Density'] = date_time_final_try['Density'].str.split(n=1).str[0]
date_time_final_try['Density'] = date_time_final_try['Density'].str.replace(',','')
```

```
date_time_final_try['Density'] = date_time_final_try['Density'].str.split('/').str[0]
```

```
date_time_final_try.fillna(0, inplace=True)
```

```
date_time_final_try['Area'] = date_time_final_try['Area'].astype(int)
date_time_final_try['Density'] = date_time_final_try['Density'].astype(int)
```

```
date_time_final_try.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1057 entries, 0 to 1056
Data columns (total 21 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   confirmed                    1057 non-null   int64
 1   recovered                    1057 non-null   int64
 2   deaths                       1057 non-null   int64
 3   day                          1057 non-null   datetime64[ns]
 4   Population                   1057 non-null   float64
 5   Rural population             1057 non-null   float64
 6   Urban population             1057 non-null   float64
 7   Area                         1057 non-null   int64
 8   Density                      1057 non-null   int64
 9   Gender Ratio                 1057 non-null   float64
 10  State/UT                     1057 non-null   object
 11  Total Police                 1057 non-null   float64
 12  ruralHospitals               1057 non-null   float64
 13  ruralBeds                    1057 non-null   float64
 14  urbanHospitals               1057 non-null   float64
 15  urbanBeds                    1057 non-null   float64
 16  totalHospitals               1057 non-null   float64
 17  totalBeds                    1057 non-null   float64
 18  asOn                         1057 non-null   object
 19  confirm_time_diff_statewise  1057 non-null   float64
 20  death_time_diff_statewise    1057 non-null   float64
dtypes: datetime64[ns](1), float64(13), int64(5), object(2)
memory usage: 181.7+ KB
```

```
date = '2020-03-14'
date = datetime.strptime(date, '%Y-%m-%d')
```

```
date_time_final_try['Day'] = date_time_final_try['day'] - date
```

```
date_time_final_try['Day'] = date_time_final_try['Day'].astype(str).str.split( ' ').str[0]
date_time_final_try['Day'] = date_time_final_try['Day'].astype(int)
```

```
date_time_final_try = date_time_final_try[['Population','Rural population','Urban population','Area'
```

```
date_time_final_try = date_time_final_try[[ 'Population' , 'Rural population' ,  'Urban population' ,  'Area' ,

date_time_final_try.sort_values(by='Day', inplace=True)
date_time_final_try.reset_index(inplace=True)
date_time_final_try.fillna(0, inplace= True)
```

Predicting cases and deaths in India using all the statistics

```
X = date_time_final_try.iloc[:,:-2]
Y_death = date_time_final_try.iloc[:,-1]
Y = date_time_final_try.iloc[:,-2]
X['State/UT'] = X['State/UT'].astype(str)

from sklearn.preprocessing import OneHotEncoder
# creating instance of one-hot-encoder
enc = OneHotEncoder(handle_unknown='ignore')
# passing bridge-types-cat column (label encoded values of bridge_types)
enc_df = pd.DataFrame(enc.fit_transform(X[['State/UT']]).toarray())
# merge with main df bridge_df on key values
X = X.join(enc_df)

X.drop(['State/UT','index'], inplace=True, axis = 1)

X
```

	Population	Rural population	Urban population	Area	Density	Gender Ratio	Total Police	ruralHospitals	ruralBeds
0	112374333.0	61556074.0	50818259.0	307713	365	929.0	220126.0	273.0	12398.0
1	3673917.0	2712464.0	961453.0	10486	350	960.0	12537.0	99.0	1140.0
2	91276115.0	62183113.0	29093002.0	88752	1029	953.0	107777.0	1272.0	19684.0
3	1458545.0	551731.0	906814.0	3702	394	973.0	5630.0	17.0	1405.0
4	1383727.0	1066358.0	317369.0	83743	17	938.0	8538.0	208.0	2136.0
...	...	...	...	...	...	...	...	...	...
1052	27743338.0	17344192.0	10399146.0	50362	550	895.0	68902.0	510.0	5805.0
1053	91276115.0	62183113.0	29093002.0	88752	1029	953.0	107777.0	1272.0	19684.0
1054	104099452.0	92341436.0	11758016.0	94163	1102	918.0	92422.0	930.0	6083.0
1055	199812341.0	155317278.0	44495063.0	240928	828	912.0	377009.0	4442.0	39104.0
1056	610577.0	456999.0	153578.0	7096	86	890.0	2482.0	24.0	260.0

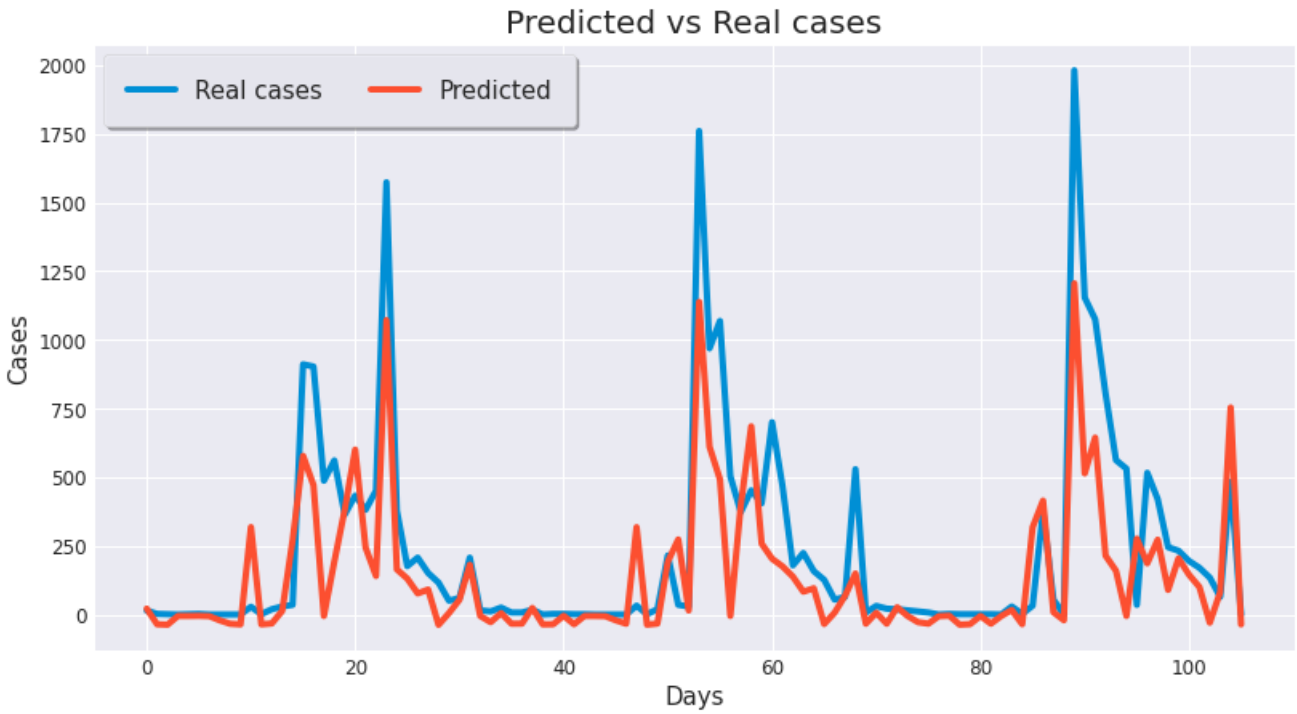
1057 rows × 46 columns

```
X_train, X_test, y_train, y_test = train_test_split(X
                                                    , Y
                                                    , test_size=0.1
                                                    , shuffle=False)

X_train_death, X_test_death, y_train_death, y_test_death = train_test_split(X
                                                    , Y_death
                                                    , test_size=0.1
                                                    , shuffle=False
                                                    , random_state = 42)

y_test = y_test.reset_index(drop = True)
y_test_death = y_test_death.reset_index(drop=True)

X_test = X_test.reset_index(drop = True)
```

```
X_test = X_test[X_test['Population'] == 49577103.0]
y_test = date_time_final_try['confirmed'].iloc[X_test.index]


poly = PolynomialFeatures(degree= 4)
poly_X_train = poly.fit_transform(X_train)
poly_X_test = poly.fit_transform(X_test)


# Transform our death data for polynomial regression
poly_death = PolynomialFeatures(degree= 4)
poly_X_train_death = poly_death.fit_transform(X_train_death)
poly_X_test_death = poly_death.fit_transform(X_test_death)



# polynomial regression cases
linear_model = LinearRegression()
linear_model.fit(poly_X_train, y_train)
test_linear_pred = linear_model.predict(poly_X_test)


# evaluating with MAE and MSE
print('MAE:', mean_absolute_error(test_linear_pred, y_test))
print('MSE:',mean_squared_error(test_linear_pred, y_test))
```

```
MAE: 142.1817589889108
MSE: 52873.184687062814
```

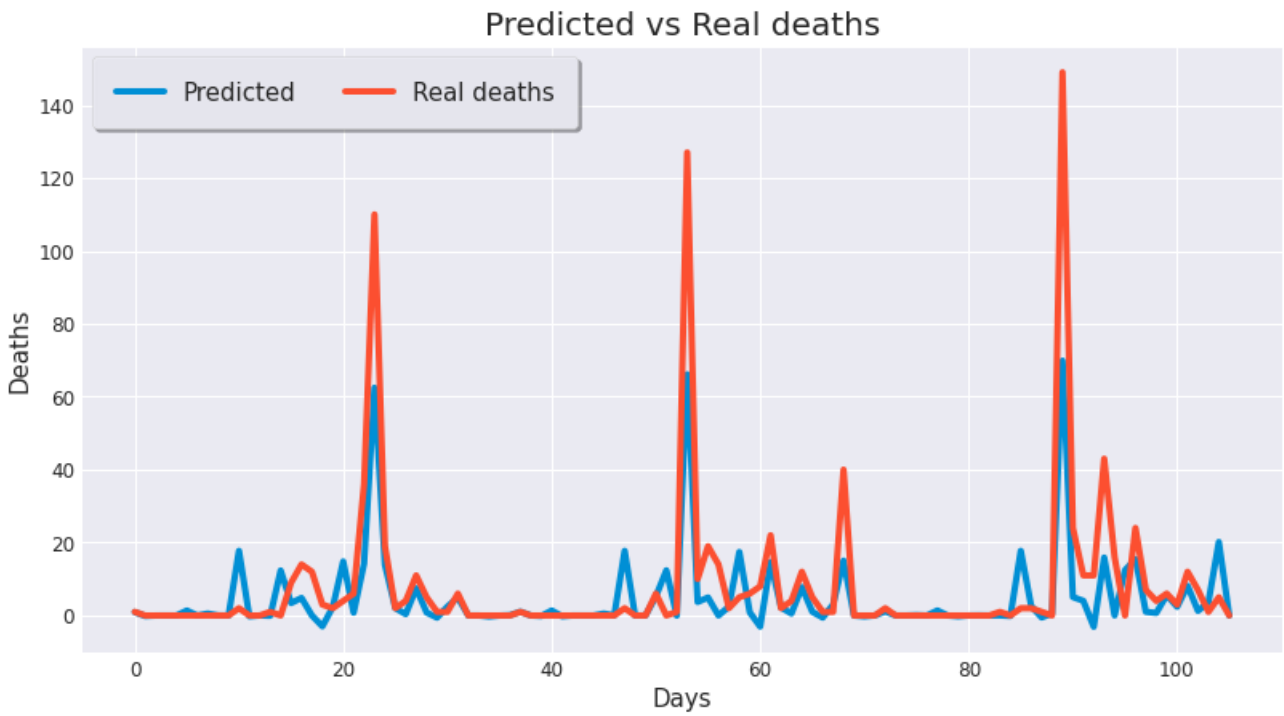
```
plt.figure(figsize=(12,7))

plt.plot(y_test, label = "Real cases")
plt.plot(test_linear_pred, label = "Predicted")
plt.title("Predicted vs Real cases", size = 20)
plt.xlabel('Days', size = 15)
plt.ylabel('Cases', size = 15)
plt.xticks(size=12)
plt.yticks(size=12)

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15});
```




```
linear_model_death = LinearRegression(fit_intercept=False)
linear_model_death.fit(poly_X_train_death, y_train_death)
test_linear_pred_death = linear_model_death.predict(poly_X_test_death)

# evaluating with MAE and MSE
print('MAE:', mean_absolute_error(test_linear_pred_death, y_test_death))
print('MSE:',mean_squared_error(test_linear_pred_death, y_test_death))
```

```
MAE: 5.835305523828821
MSE: 168.76949582190258
```

```
plt.figure(figsize=(12,7))

plt.plot(test_linear_pred_death, label = "Predicted")
plt.plot(y_test_death, label = "Real deaths")
plt.title("Predicted vs Real deaths", size = 20)
plt.xlabel('Days', size = 15)
plt.ylabel('Deaths', size = 15)
plt.xticks(size=12)
plt.yticks(size=12)

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15});
```



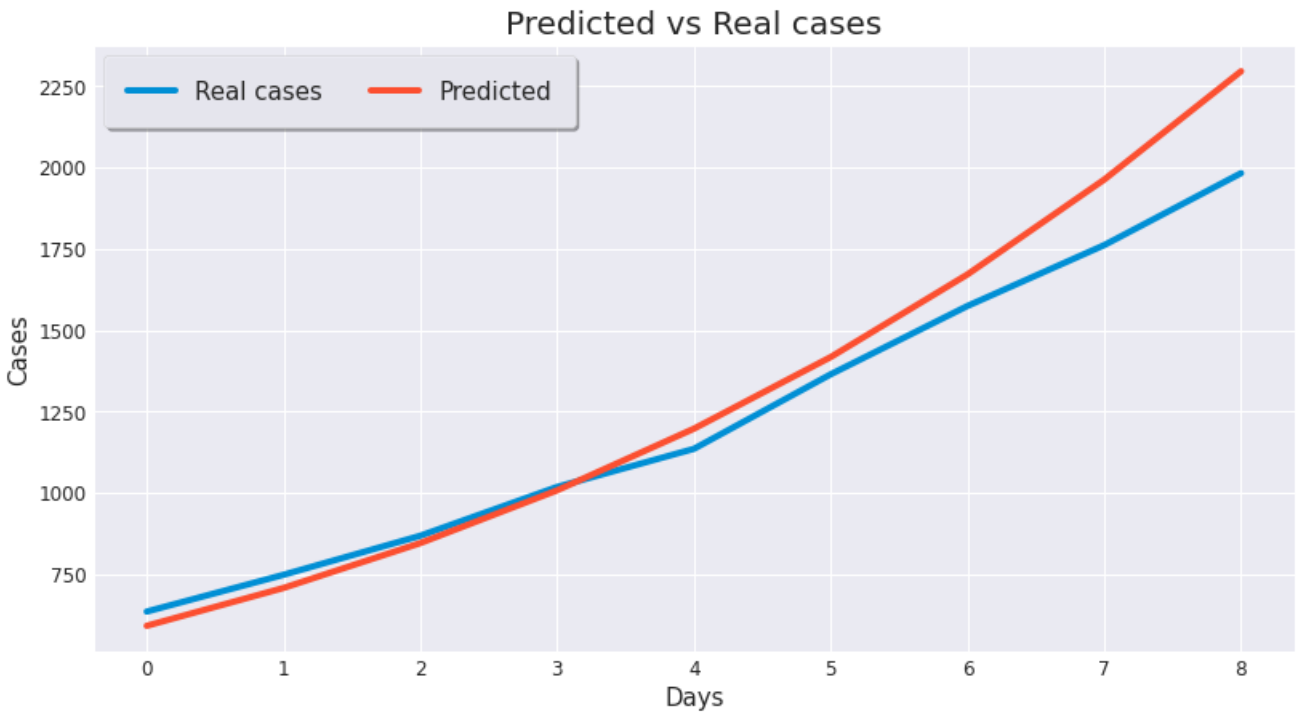
Predicting Cases and Deaths in Maharashtra using only the number of cases and deatl

```
X = date_time_final_try.iloc[:,:-2]
Y_death = date_time_final_try.iloc[:,-1]
Y = date_time_final_try.iloc[:,-2]


date_time_final_try_m = date_time_final_try[date_time_final_try['State/UT'] == 'Maharashtra']


cases_m = date_time_final_try_m['confirmed'].groupby(date_time_final_try['Day']).sum().sort_values(ascen
deaths_m = date_time_final_try_m['deaths'].groupby(date_time_final_try['Day']).sum().sort_values(ascendi
maharasthra_cases = np.array(cases_m).reshape(-1, 1)

maharashtra_deaths = np.array(deaths_m).reshape(-1, 1)
days_since_first_case_m = np.array([i for i in range(len(cases_m.index))]).reshape(-1, 1)
days_since_first_death_m = np.array([i for i in range(len(deaths_m.index))]).reshape(-1, 1)

days_in_future = 15
future_forcast_m = np.array([i for i in range(len(cases_m.index)+days_in_future)]).reshape(-1, 1)
adjusted_dates = future_forcast_m[:-15]
future_forcast_deaths_m = np.array([i for i in range(len(deaths_m.index)+days_in_future)]).reshape(-1, 1
adjusted_dates_deaths = future_forcast_deaths_m[:-15]


X_train, X_test, y_train, y_test = train_test_split(days_since_first_case_m
                                                    , maharasthra_cases
                                                    , test_size=0.30
                                                    , shuffle=False)

X_train_death, X_test_death, y_train_death, y_test_death = train_test_split(days_since_first_death_m
                                                    , maharashtra_deaths
                                                    , test_size=0.3
                                                    , shuffle=False
                                                    , random_state = 42)


poly = PolynomialFeatures(degree= 4)
poly_X_train = poly.fit_transform(X_train)
poly_X_test = poly.fit_transform(X_test)
poly_future_forcast = poly.fit_transform(future_forcast_m)

# Transform our death data for polynomial regression
```

```
poly_death = PolynomialFeatures(degree= 4)
poly_X_train_death = poly_death.fit_transform(X_train_death)
poly_X_test_death = poly_death.fit_transform(X_test_death)
poly_future_forcast_death = poly_death.fit_transform(future_forcast_deaths_m)


# polynomial regression cases
linear_model = LinearRegression()
linear_model.fit(poly_X_train, y_train)
test_linear_pred = linear_model.predict(poly_X_test)
linear_pred = linear_model.predict(poly_future_forcast)

# evaluating with MAE and MSE
print('MAE:', mean_absolute_error(test_linear_pred, y_test))
print('MSE:',mean_squared_error(test_linear_pred, y_test))
```

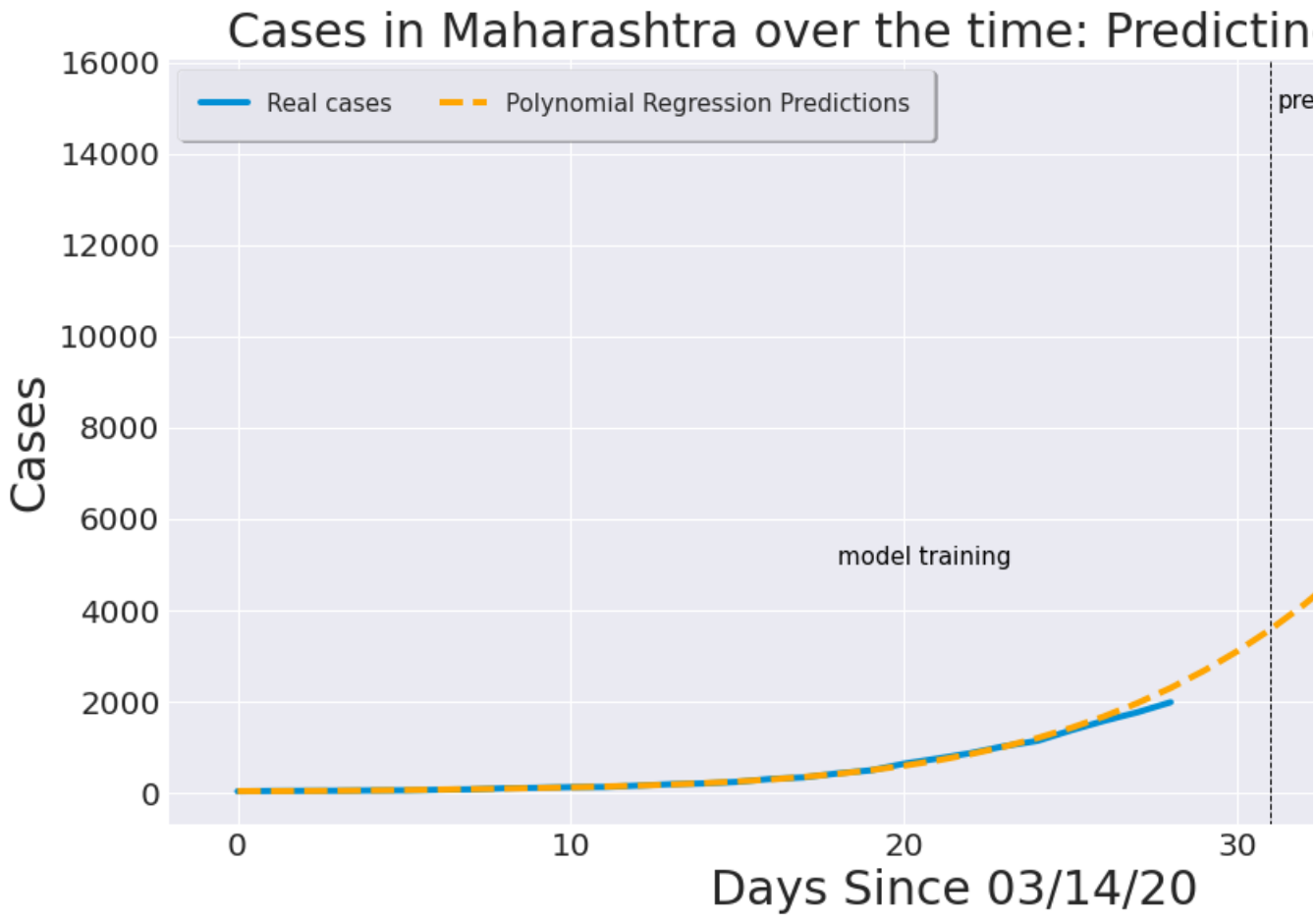
```
MAE: 93.74065935496594
MSE: 17646.22555561457
```

```
plt.figure(figsize=(12,7))

plt.plot(y_test, label = "Real cases")
plt.plot(test_linear_pred, label = "Predicted")
plt.title("Predicted vs Real cases", size = 20)
plt.xlabel('Days', size = 15)
plt.ylabel('Cases', size = 15)
plt.xticks(size=12)
plt.yticks(size=12)

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15});
```




```
plt.figure(figsize=(16, 9))

plt.plot(adjusted_dates
         , maharasthra_cases
         , label = "Real cases")
```

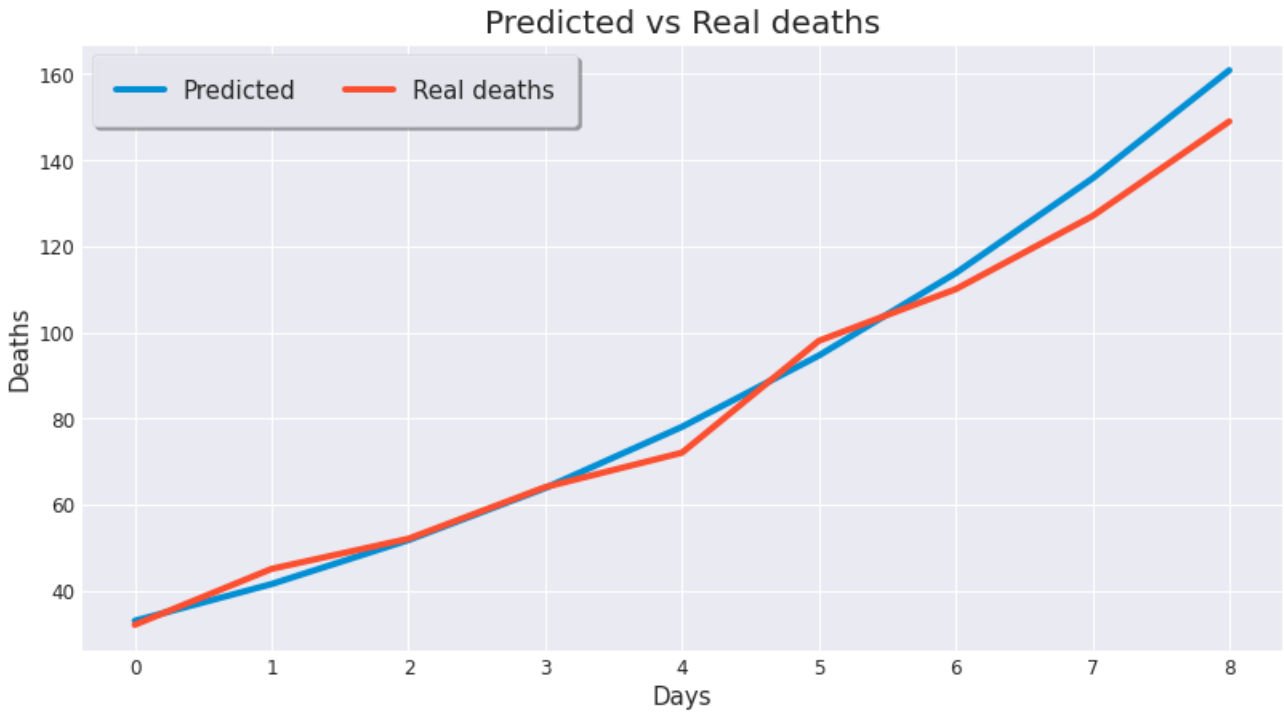
```
plt.plot(future_forcast_m
         , linear_pred
         , label = "Polynomial Regression Predictions"
         , linestyle='dashed'
         , color='orange')

plt.title('Cases in Maharashtra over the time: Predicting Next 2 Weeks', size=30)
plt.xlabel('Days Since 03/14/20', size=30)
plt.ylabel('Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)

plt.axvline(31, color='black'
            , linestyle="--"
            , linewidth=1)

plt.text(18, 5000
         , "model training"
         , size = 15
         , color = "black")

plt.text(31.2, 15000
         , "prediction"
         , size = 15
         , color = "black")

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15})

plt.show();
```




```
linear_model_death = LinearRegression(fit_intercept=False)
linear_model_death.fit(poly_X_train_death, y_train_death)
test_linear_pred_death = linear_model_death.predict(poly_X_test_death)
linear_pred_death = linear_model_death.predict(poly_future_forcast_death)

# evaluating with MAE and MSE
print('MAE:', mean_absolute_error(test_linear_pred_death, y_test_death))
print('MSE:',mean_squared_error(test_linear_pred_death, y_test_death))
```

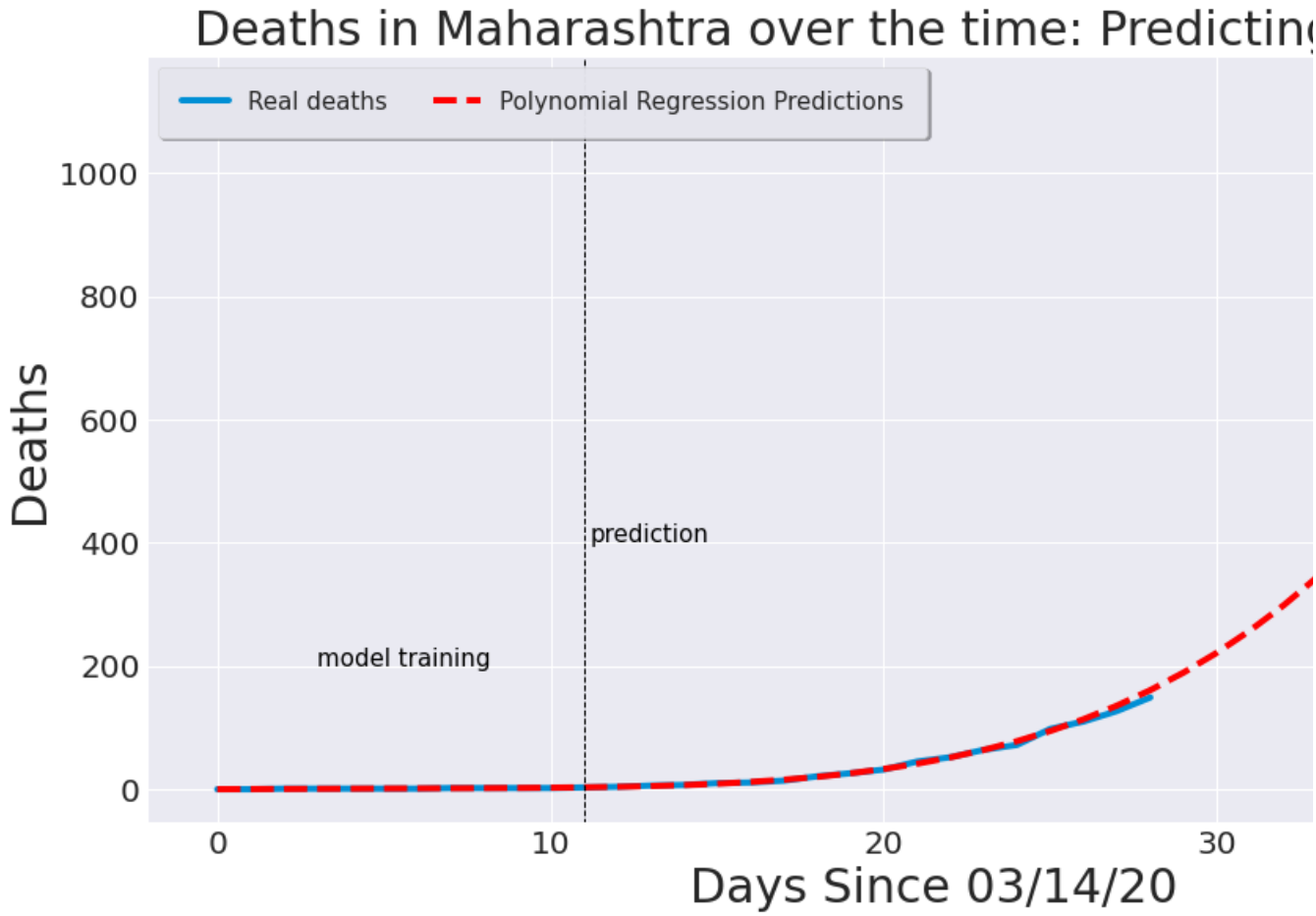
```
MAE: 4.312663698953685
MSE: 32.603175641518675
```

```
plt.figure(figsize=(12,7))

plt.plot(test_linear_pred_death, label = "Predicted")
plt.plot(y_test_death, label = "Real deaths")
plt.title("Predicted vs Real deaths", size = 20)
plt.xlabel('Days', size = 15)
plt.ylabel('Deaths', size = 15)
plt.xticks(size=12)
plt.yticks(size=12)

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15});
```




```
plt.figure(figsize=(16, 9))

plt.plot(adjusted_dates_deaths
         , maharashtra_deaths
         , label = "Real deaths")

plt.plot(future_forcast_deaths_m
         , linear_pred_death
         , label = "Polynomial Regression Predictions"
         , linestyle='dashed'
         , color='red')

plt.title('Deaths in Maharashtra over the time: Predicting Next 2 Weeks', size=30)
plt.xlabel('Days Since 03/14/20', size=30)
plt.ylabel('Deaths', size=30)
plt.xticks(size=20)
plt.yticks(size=20)

plt.axvline(11, color='black'
            , linestyle="--"
            , linewidth=1)

plt.text(3, 200
         , "model training"
         , size = 15
         , color = "black")

plt.text(11.2, 400
         , "prediction"
         , size = 15
         , color = "black")

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15})
```

```
plt.show();
```



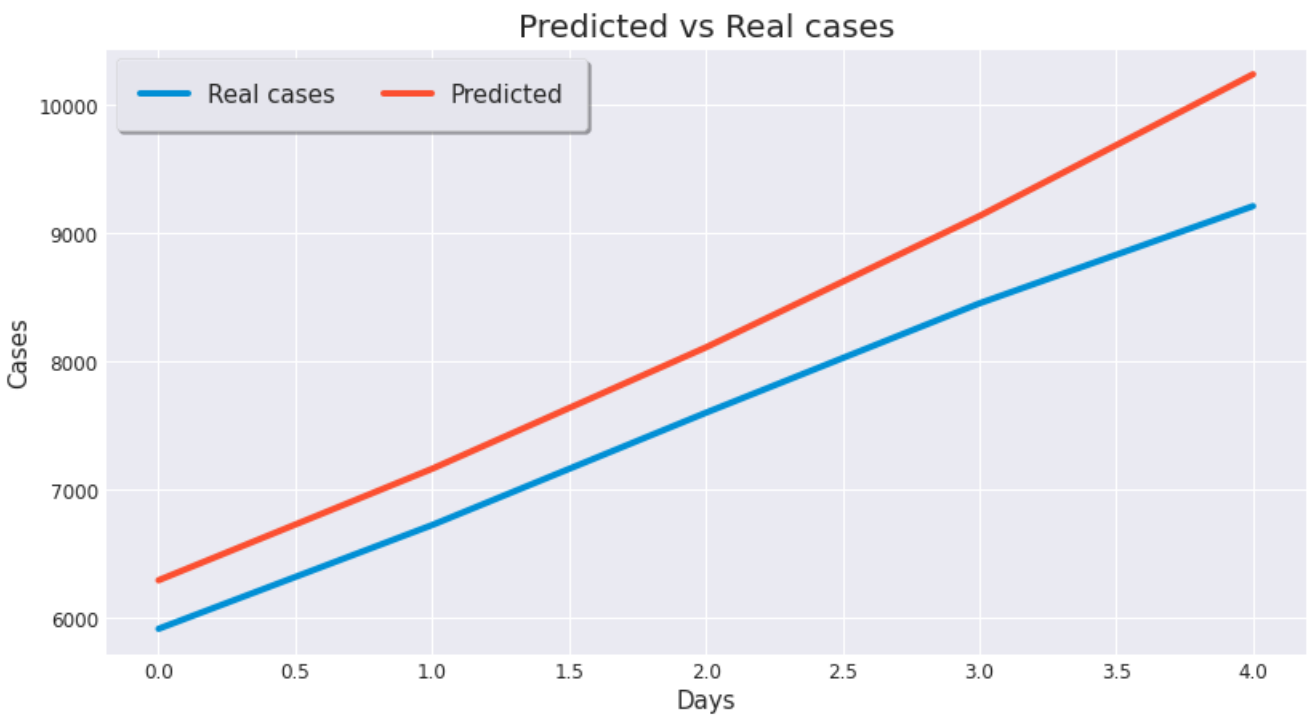
Predicting Cases and Deaths in **India** using only the number of cases and deaths per d

```
cases = date_time_final_try['confirmed'].groupby(date_time_final_try['Day']).sum().sort_values(ascending=

deaths =  date_time_final_try['deaths'].groupby(date_time_final_try['Day']).sum().sort_values(ascending=T
India_cases = np.array(cases).reshape(-1, 1)

India_deaths = np.array(deaths).reshape(-1, 1)
days_since_first_case = np.array([i for i in range(len(cases.index))]).reshape(-1, 1)
days_since_first_death = np.array([i for i in range(len(deaths.index))]).reshape(-1, 1)

days_in_future = 15
future_forcast = np.array([i for i in range(len(cases.index)+days_in_future)]).reshape(-1, 1)
adjusted_dates = future_forcast[:-15]
future_forcast_deaths = np.array([i for i in range(len(deaths.index)+days_in_future)]).reshape(-1, 1)
adjusted_dates_deaths = future_forcast_deaths[:-15]


X_train, X_test, y_train, y_test = train_test_split(days_since_first_case
                                                    , India_cases
                                                    , test_size=0.15
                                                    , shuffle=False)

X_train_death, X_test_death, y_train_death, y_test_death = train_test_split(days_since_first_death
                                                    , India_deaths
                                                    , test_size=0.15
                                                    , shuffle=False
                                                    , random_state = 42)


poly = PolynomialFeatures(degree= 4)
poly_X_train = poly.fit_transform(X_train)
poly_X_test = poly.fit_transform(X_test)
poly_future_forcast = poly.fit_transform(future_forcast)
```

```
# Transform our death data for polynomial regression
poly_death = PolynomialFeatures(degree= 4)
poly_X_train_death = poly_death.fit_transform(X_train_death)
poly_X_test_death = poly_death.fit_transform(X_test_death)
poly_future_forcast_death = poly_death.fit_transform(future_forcast_deaths)


# polynomial regression cases
linear_model = LinearRegression()
linear_model.fit(poly_X_train, y_train)
test_linear_pred = linear_model.predict(poly_X_test)
linear_pred = linear_model.predict(poly_future_forcast)

# evaluating with MAE and MSE
print('MAE:', mean_absolute_error(test_linear_pred, y_test))
print('MSE:',mean_squared_error(test_linear_pred, y_test))
```

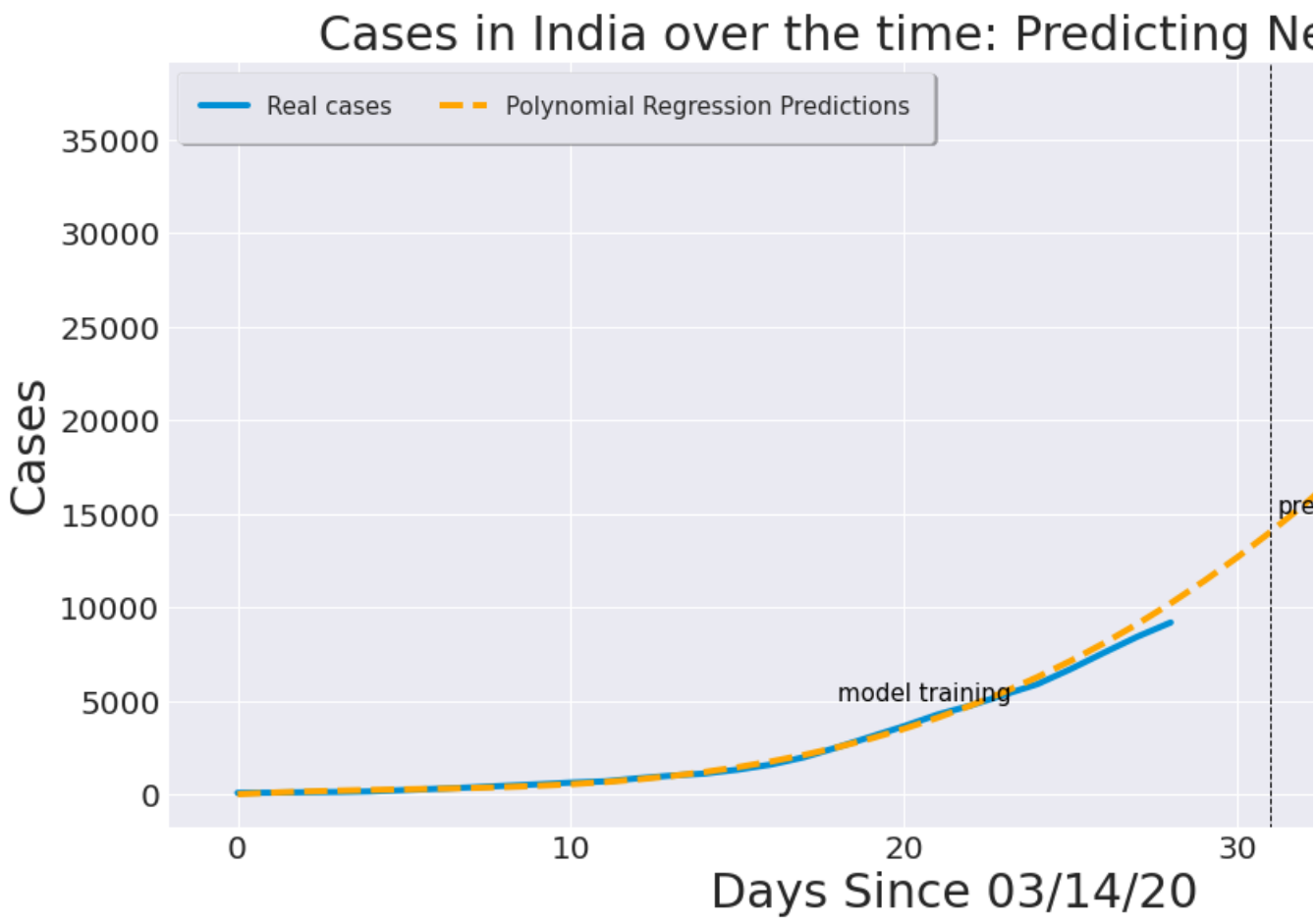
```
MAE: 607.1298125673926
MSE: 423154.524667917
```

```
plt.figure(figsize=(12,7))

plt.plot(y_test, label = "Real cases")
plt.plot(test_linear_pred, label = "Predicted")
plt.title("Predicted vs Real cases", size = 20)
plt.xlabel('Days', size = 15)
plt.ylabel('Cases', size = 15)
plt.xticks(size=12)
plt.yticks(size=12)

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15});
```




```
plt.figure(figsize=(16, 9))

plt.plot(adjusted dates
```

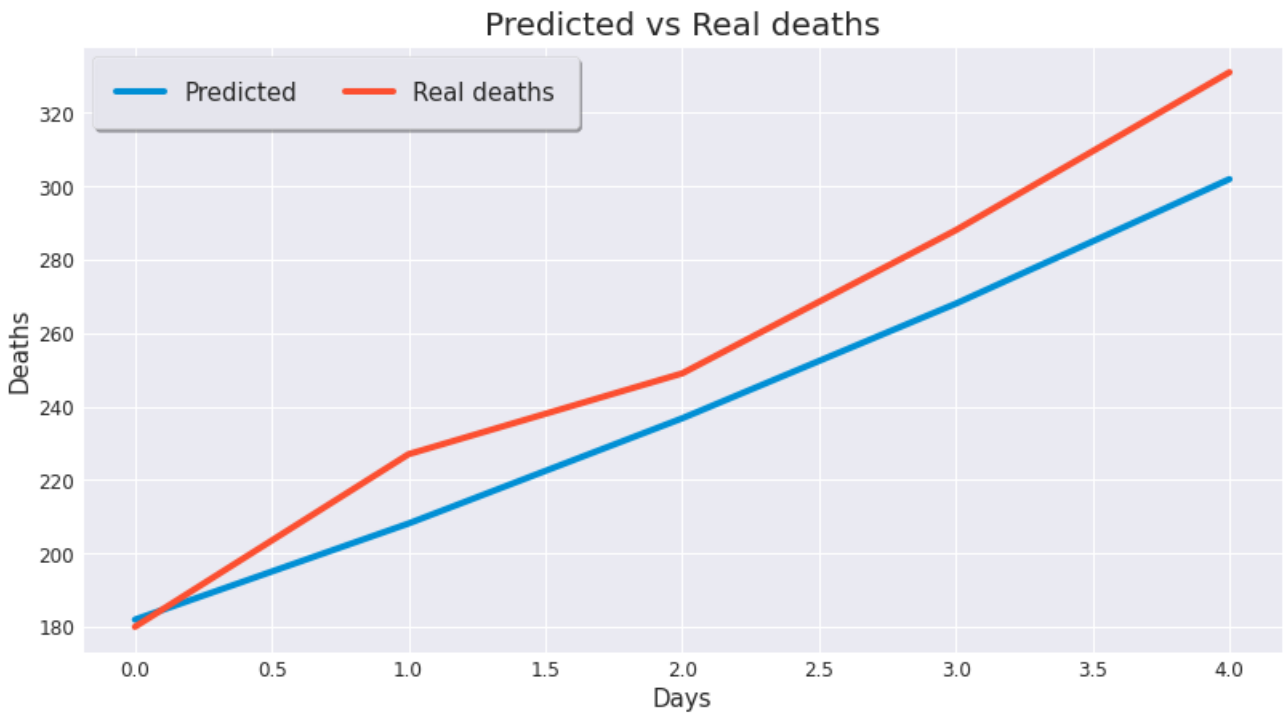
```
          , India_cases
          , label = "Real cases")

plt.plot(future_forcast
          , linear_pred
          , label = "Polynomial Regression Predictions"
          , linestyle='dashed'
          , color='orange')

plt.title('Cases in India over the time: Predicting Next 2 Weeks', size=30)
plt.xlabel('Days Since 03/14/20', size=30)
plt.ylabel('Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)

plt.axvline(31, color='black'
             , linestyle="--"
             , linewidth=1)

plt.text(18, 5000
          , "model training"
          , size = 15
          , color = "black")

plt.text(31.2, 15000
          , "prediction"
          , size = 15
          , color = "black")

# defyning legend config
plt.legend(loc = "upper left"
            , frameon = True
            , ncol = 2
            , fancybox = True
            , framealpha = 0.95
            , shadow = True
            , borderpad = 1
            , prop={'size': 15})

plt.show();
```




```
linear_model_death = LinearRegression(fit_intercept=False)
linear_model_death.fit(poly_X_train_death, y_train_death)
test_linear_pred_death = linear_model_death.predict(poly_X_test_death)
linear_pred_death = linear_model_death.predict(poly_future_forcast_death)

# evaluating with MAE and MSE
print('MAE:', mean_absolute_error(test_linear_pred_death, y_test_death))
print('MSE:',mean_squared_error(test_linear_pred_death, y_test_death))
```

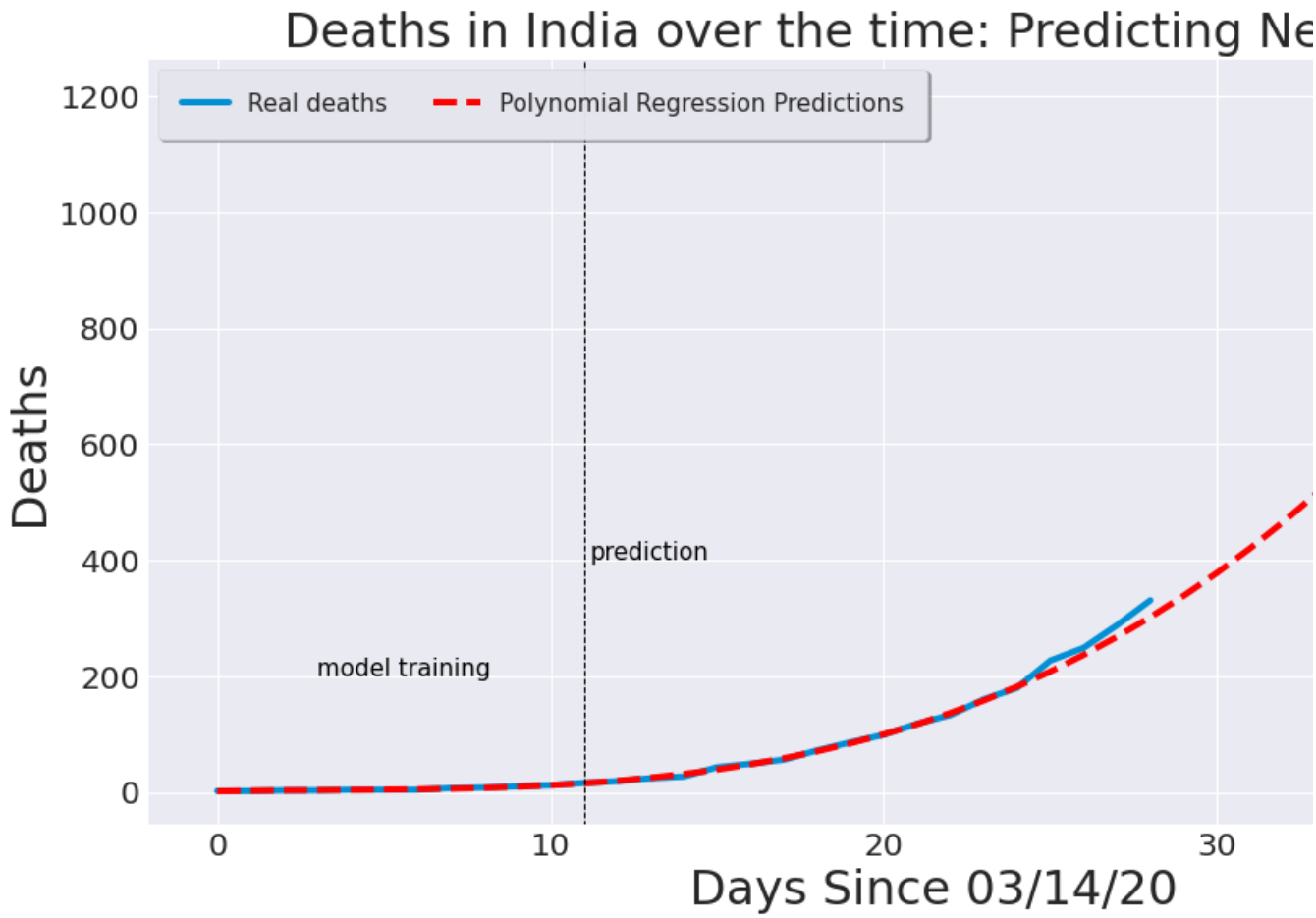
```
MAE: 16.430685843913515
MSE: 351.21935863785046
```

```
plt.figure(figsize=(12,7))

plt.plot(test_linear_pred_death, label = "Predicted")
plt.plot(y_test_death, label = "Real deaths")
plt.title("Predicted vs Real deaths", size = 20)
plt.xlabel('Days', size = 15)
plt.ylabel('Deaths', size = 15)
plt.xticks(size=12)
plt.yticks(size=12)

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15});
```




```
plt.figure(figsize=(16, 9))

plt.plot(adjusted_dates_deaths
         , India_deaths
         , label = "Real deaths")

plt.plot(future_forcast_deaths
         , linear_pred_death
         , label = "Polynomial Regression Predictions"
         , linestyle='dashed'
         , color='red')

plt.title('Deaths in India over the time: Predicting Next 2 Weeks', size=30)
plt.xlabel('Days Since 03/14/20', size=30)
plt.ylabel('Deaths', size=30)
plt.xticks(size=20)
plt.yticks(size=20)

plt.axvline(11, color='black'
            , linestyle="--"
            , linewidth=1)

plt.text(3, 200
         , "model training"
         , size = 15
         , color = "black")

plt.text(11.2, 400
         , "prediction"
         , size = 15
         , color = "black")

# defyning legend config
plt.legend(loc = "upper left"
           , frameon = True
           , ncol = 2
           , fancybox = True
           , framealpha = 0.95
           , shadow = True
           , borderpad = 1
           , prop={'size': 15})
```

```
plt.show();
```


Deaths in India over the time: Predicting Ne
Real deaths
Polynomial Regression Predictions
Deaths
1200
1000
800
600
400
200
0
prediction
model training
0
10
20
30
Days Since 03/14/20