Tieto_internship

June 14, 2020

0.1 Covid-19 prediction

```
[1]: import pandas as pd
     import numpy as np
     import seaborn as sns
```

Firstly, we will download our dataset from official site of Czech republic about Covid-19 development in Czech republic.

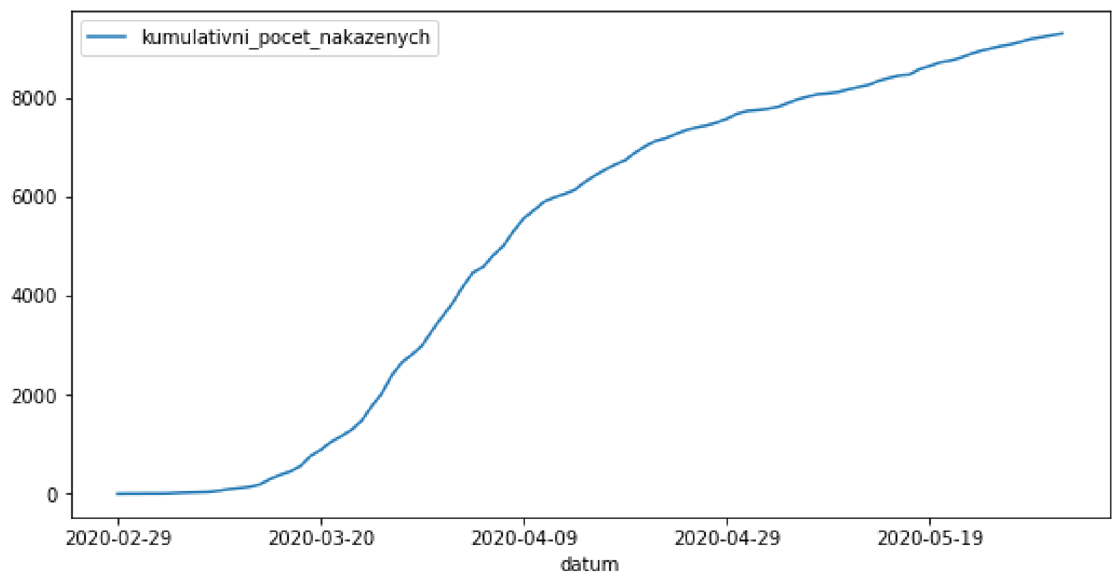
```
[2]: covid = pd.read_csv("nakaza.csv")
     covid_cumulative = covid.drop(['prirustkovy_pocet_nakazenych'],axis=1)
     covid_spread = covid.drop(['kumulativni_pocet_nakazenych'],axis=1)
```

```
[3]: ICD = pd.read_csv("nakazeni-vyleceni-umrti-testy.csv")
     covid_currently = pd.DataFrame(ICD["datum"])
     covid_currently["currently_infected"] = (ICD["kumulativni_pocet_nakazenych"] -␣
      ↪ICD["kumulativni_pocet_vylecenych"])
```

We will predict three aspects of development covid-19 in czech republic: Cumulative development in `covid_cumulative`: by this data we inspect absolute number of infected people day by day. New infected in `covid_spread`: this aspect says how many "new" people were infected. Actual number of infected in `covid_currently`: in this `Serie` we will store information about how many infected people were there at some concrete day. This means we will take `covid_cumulative` and substract number of already dead and healed people from it.

```
[18]: covid_cumulative.plot(x='datum',figsize=(10,5))
```

```
[18]: <matplotlib.axes._subplots.AxesSubplot at 0x2148d8a42b0>
```

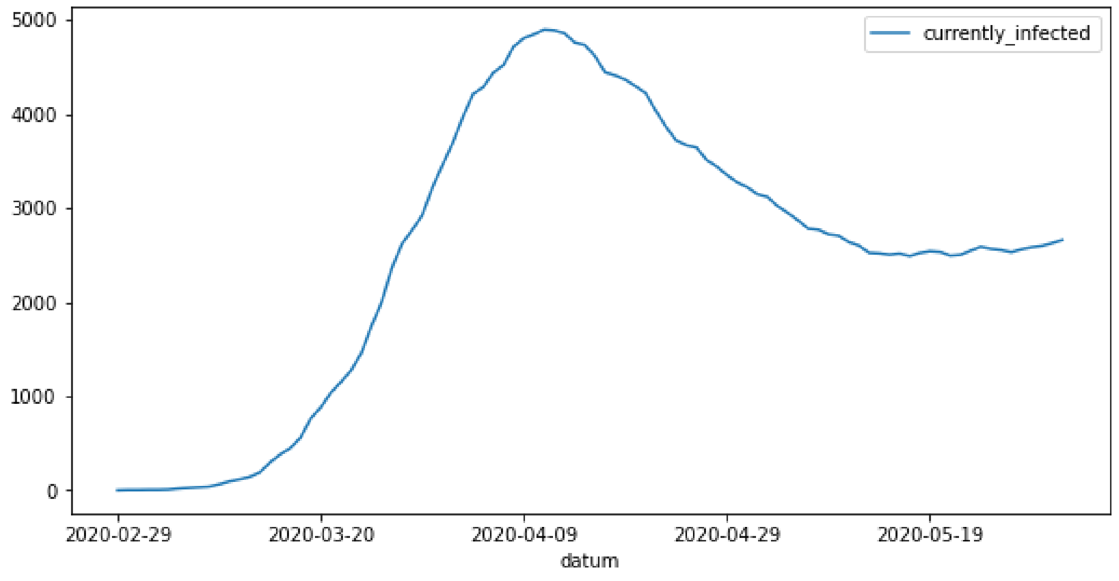


We see that around 30 days have no information at all. We can throw away these days from our dataset.

[4]:
```
covid_cumulative = covid_cumulative.drop(range(33),axis=0)
covid_spread = covid_spread.drop(range(33),axis=0)
covid_currently = covid_currently.drop(range(33),axis=0)
```

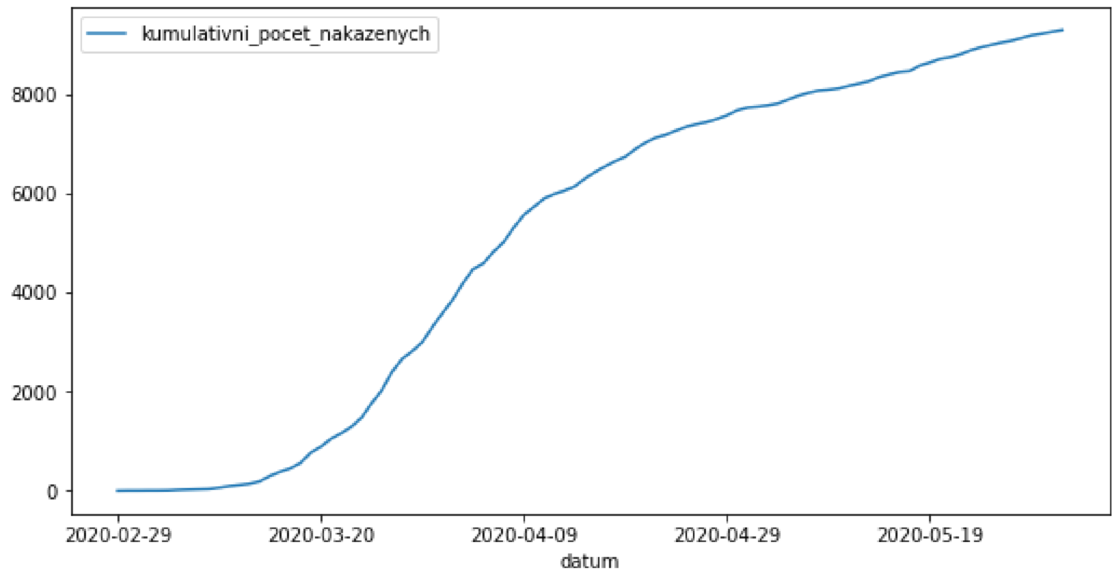
[7]:
```
covid_currently.plot(x='datum',figsize=(10,5))
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ef1409c250>
```



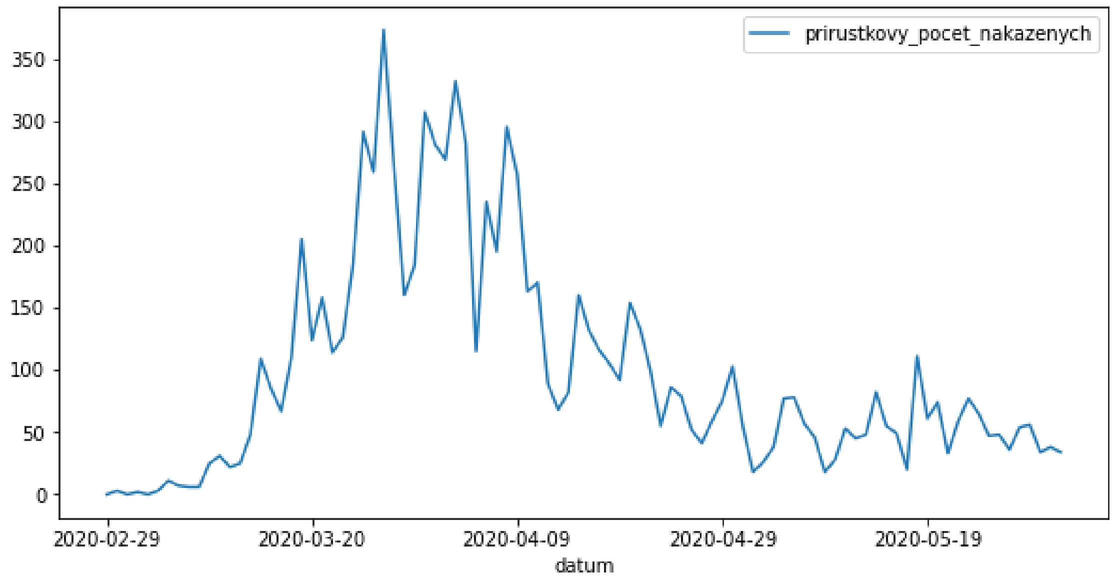
[11]:
```
covid_cumulative.plot(x='datum',figsize=(10,5))
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2260979d700>
```



[12]:
```
covid_spread.plot(x='datum',figsize=(10,5))
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x226097e9f10>
```



0.2 Model picking

It seems that there are too few samples in our dataset. We could use LSTM in NN, which is one of the best model for time series forecasting. However LSTM needs bigger datasets to train on. It would be a little overkill to use LSTM on such small dataset and it would very probably overfit. We could try oversampling to bigger our dataset but in timeforecasting it's useless because it wouldn't give us any new information. Since the first two graphs are very smooth, the ARIMA model could be very good choice for predictions on such dataset. Let's try ARIMA and look what results will it produce.

[8]:
```
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error

def prediction(X, lag):
    size = int(len(X) * 0.66)
    train, test = X[0:size], X[size:len(X)]
    history = [x for x in train]
    predictions = list()
    for t in range(62,94):
        model = ARIMA(history, order=(lag,1,0))
        model_fit = model.fit(disp=0)
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
        print('predicted=%f, expected=%f' % (yhat, obs))
    error = mean_squared_error(test, predictions)
    print('Test MSE: %.3f' % error)
    return test, predictions
```

[7]:
```
def paint_plot(test, predictions):
    dict = {}
    for i in range(62, 94):
        dict.update({i: i-62})

    test = test.rename(index=dict)

    pyplot.plot(test)
    pyplot.plot(predictions, color='red')
    pyplot.show()
```

[5]:
```
dict = {}
for i in range(33, 127):
    dict.update({i: i-33})
```

```
[6]: covid_spread = covid_spread.rename(index=dict)
     covid_cumulative = covid_cumulative.rename(index=dict)
     covid_currently = covid_currently.rename(index=dict)
```

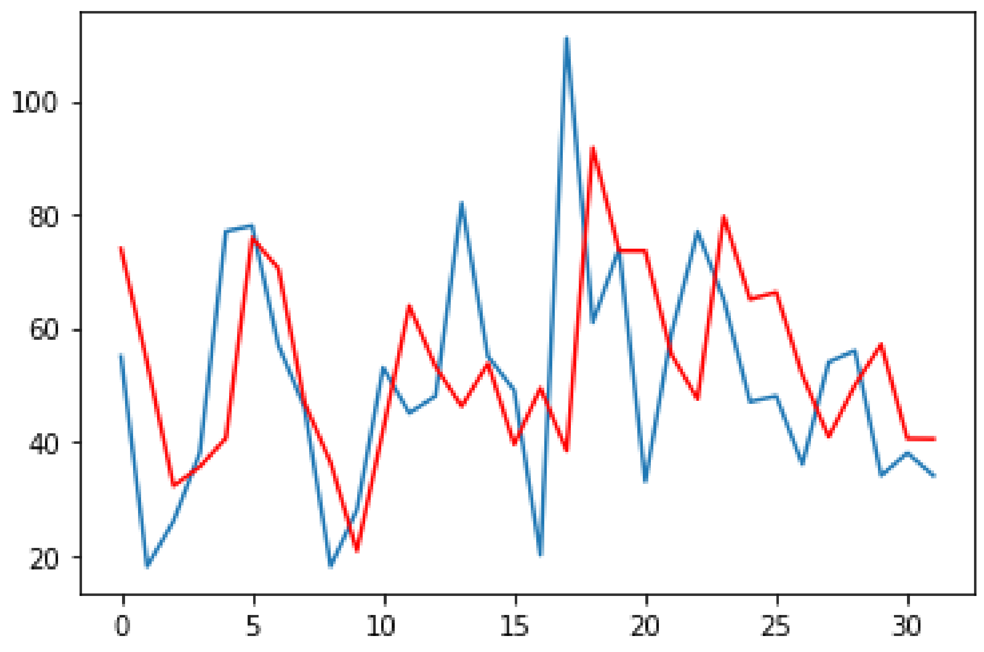
Let us explain why in the next step we set attribute `lag` to value 10. We used `GridSearch` from `sklearn` library to tune `lag` attribute in ARIMA. Being `lag` set to 10 provided best results. Significantly higher values for this attribute would take too long time to train, so we will leave it set at 10.

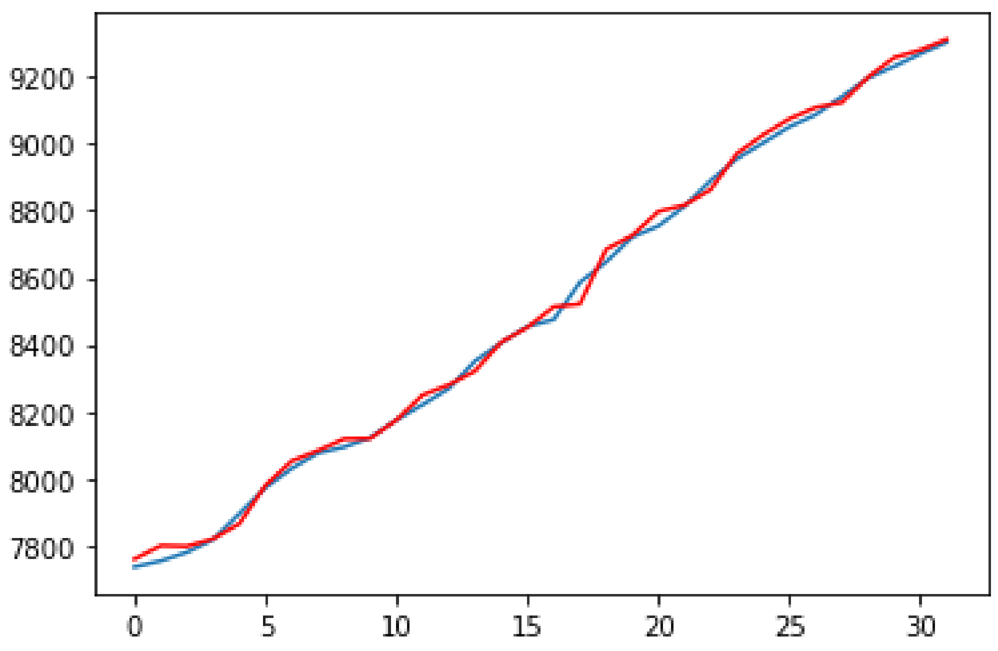
```
[9]: X1 = covid_spread.prirustkovy_pocet_nakazenych
     X2 = covid_cumulative.kumulativni_pocet_nakazenych
     X3 = covid_currently.currently_infected

     print("covid_spread:")
     print()
     test1, predictions1 = prediction(X1, 10)
     print("covid_cumulative:")
     print()
     test2, predictions2 = prediction(X2, 10)
     print("covid_currently:")
     print()
     test3, predictions3 = prediction(X3, 10)

     paint_plot(test1, predictions1)
     paint_plot(test2, predictions2)
     paint_plot(test3, predictions3)
```

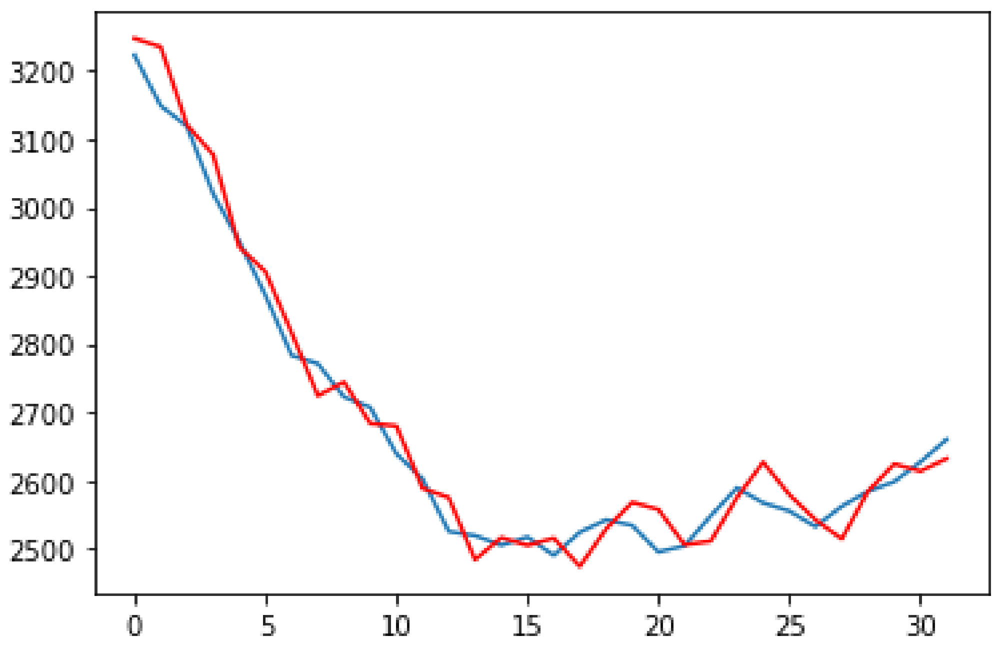
```
covid_spread:

predicted=73.969307, expected=55.000000
predicted=53.840930, expected=18.000000
predicted=32.179484, expected=26.000000
predicted=35.559458, expected=38.000000
predicted=40.606964, expected=77.000000
predicted=75.851110, expected=78.000000
predicted=70.516307, expected=57.000000
predicted=46.832880, expected=46.000000
predicted=36.263568, expected=18.000000
predicted=20.727396, expected=28.000000
predicted=42.046015, expected=53.000000
predicted=63.878142, expected=45.000000
predicted=53.188443, expected=48.000000
predicted=46.169009, expected=82.000000
predicted=53.737430, expected=55.000000
predicted=39.488994, expected=49.000000
predicted=49.415236, expected=20.000000
predicted=38.413359, expected=111.000000
predicted=91.735837, expected=61.000000
```

```
predicted=73.612670, expected=74.000000
predicted=73.590562, expected=33.000000
predicted=55.341901, expected=59.000000
predicted=47.455404, expected=77.000000
predicted=79.638904, expected=65.000000
predicted=65.079159, expected=47.000000
predicted=66.281132, expected=48.000000
predicted=51.711886, expected=36.000000
predicted=40.833005, expected=54.000000
predicted=49.922206, expected=56.000000
predicted=57.120958, expected=34.000000
predicted=40.538542, expected=38.000000
predicted=40.478256, expected=34.000000
Test MSE: 531.092
covid_cumulative:

predicted=7760.660383, expected=7737.000000
predicted=7800.616424, expected=7755.000000
predicted=7798.750264, expected=7781.000000
predicted=7820.848733, expected=7819.000000
predicted=7866.047884, expected=7896.000000
predicted=7980.731544, expected=7974.000000
predicted=8053.407458, expected=8031.000000
predicted=8084.239489, expected=8077.000000
predicted=8119.429701, expected=8095.000000
predicted=8120.597736, expected=8123.000000
predicted=8175.798447, expected=8176.000000
predicted=8249.666510, expected=8221.000000
predicted=8280.434651, expected=8269.000000
predicted=8321.330378, expected=8351.000000
predicted=8407.789549, expected=8406.000000
predicted=8452.097163, expected=8455.000000
predicted=8513.097157, expected=8475.000000
predicted=8521.513321, expected=8586.000000
predicted=8684.204978, expected=8647.000000
predicted=8725.884899, expected=8721.000000
predicted=8797.519561, expected=8754.000000
predicted=8816.443843, expected=8813.000000
predicted=8862.982144, expected=8890.000000
predicted=8970.483541, expected=8955.000000
predicted=9026.285384, expected=9002.000000
predicted=9073.974681, expected=9050.000000
predicted=9107.876766, expected=9086.000000
predicted=9122.755635, expected=9140.000000
predicted=9198.236082, expected=9196.000000
predicted=9256.504903, expected=9230.000000
predicted=9278.860546, expected=9268.000000
predicted=9310.864667, expected=9302.000000
```

```
Test MSE: 613.789
covid_currently:

predicted=3247.729907, expected=3223.000000
predicted=3235.172820, expected=3149.000000
predicted=3120.966491, expected=3119.000000
predicted=3077.472898, expected=3020.000000
predicted=2942.114855, expected=2950.000000
predicted=2905.836613, expected=2871.000000
predicted=2816.158551, expected=2783.000000
predicted=2723.973699, expected=2771.000000
predicted=2743.993473, expected=2722.000000
predicted=2683.436098, expected=2707.000000
predicted=2679.886474, expected=2639.000000
predicted=2588.338691, expected=2602.000000
predicted=2574.811862, expected=2525.000000
predicted=2483.587701, expected=2519.000000
predicted=2515.985554, expected=2505.000000
predicted=2504.722085, expected=2517.000000
predicted=2515.177777, expected=2490.000000
predicted=2473.790801, expected=2524.000000
predicted=2528.697010, expected=2542.000000
predicted=2567.847530, expected=2534.000000
predicted=2557.270974, expected=2495.000000
predicted=2505.544734, expected=2504.000000
predicted=2510.868503, expected=2548.000000
predicted=2574.857859, expected=2589.000000
predicted=2626.612918, expected=2567.000000
predicted=2579.065496, expected=2555.000000
predicted=2542.131447, expected=2532.000000
predicted=2514.158552, expected=2562.000000
predicted=2584.082298, expected=2584.000000
predicted=2624.039713, expected=2598.000000
predicted=2613.490665, expected=2627.000000
predicted=2631.799572, expected=2660.000000
Test MSE: 1279.734
```


100
80
60
40
20
0
5
10
15
20
25
30


9200
9000
8800
8600
8400
8200
8000
7800
0
5
10
15
20
25
30



0.3 Model evaluation

As we can see upper in model training, for evaluation we used `MSE`. It might seems pretty big but when we plot our results that predictions are pretty accure. We have to have in our minds that we trained it only on about 60 samples. The lack of data displays in sudden ups and downs of function as we can see in `covid_spread` prediction. On the other hand our model can approximate smooth function really good as we can see in `covid_cumulative`.

0.4 Comparation of Czech republic with other countries

We will compare cumulative and new infected development of covid-19 in Czech republic to development in Italy and Austria. Let us load the data and take just the information we need.

[10]:
```
covid_worldwide = pd.read_csv("owid-covid-data.csv")
```

[11]:
```
covid_Austria = covid_worldwide.drop(range(1229),axis=0)
covid_Austria = covid_Austria.drop(range(1390,23082),axis=0)
```

[12]:
```
covid_Italy = covid_worldwide.drop(range(10690),axis=0)
covid_Italy = covid_Italy.drop(range(10851,23082),axis=0)
```

[13]:
```
covid_Italy_cumulative = covid_Italy.total_cases
covid_Italy_spread = covid_Italy.new_cases
covid_Austria_cumulative = covid_Austria.total_cases
covid_Austria_spread = covid_Austria.new_cases
```

[14]:
```python
dict = {}
for i in range(10757, 10851):
    dict.update({i: i-10757})

covid_Italy_spread = covid_Italy_spread.drop(range(10690, 10757),axis=0)
covid_Italy_cumulative = covid_Italy_cumulative.drop(range(10690, 10757),axis=0)

covid_Italy_spread = covid_Italy_spread.rename(index=dict)
covid_Italy_cumulative = covid_Italy_cumulative.rename(index=dict)
```

[15]:
```python
dict = {}
for i in range(1296, 1390):
    dict.update({i: i-1296})

covid_Austria_spread = covid_Austria_spread.drop(range(1229, 1296),axis=0)
covid_Austria_cumulative = covid_Austria_cumulative.drop(range(1229,␣
 ↪1296),axis=0)

covid_Austria_spread = covid_Austria_spread.rename(index=dict)
covid_Austria_cumulative = covid_Austria_cumulative.rename(index=dict)
```

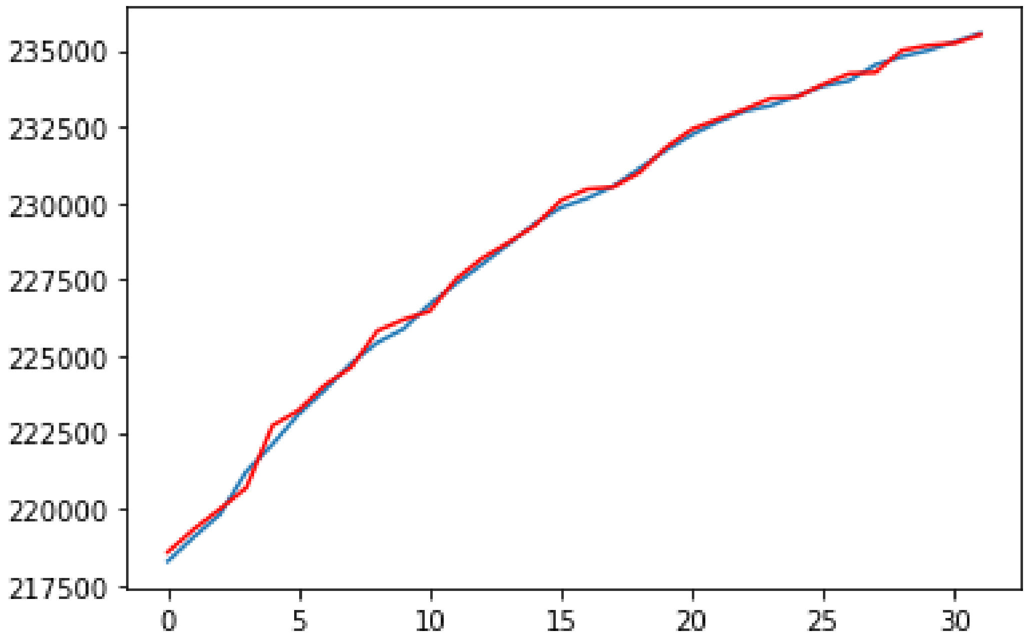
Simillarly we tuned `lag` attribute with `GridSearch` and set the optimum for each model.

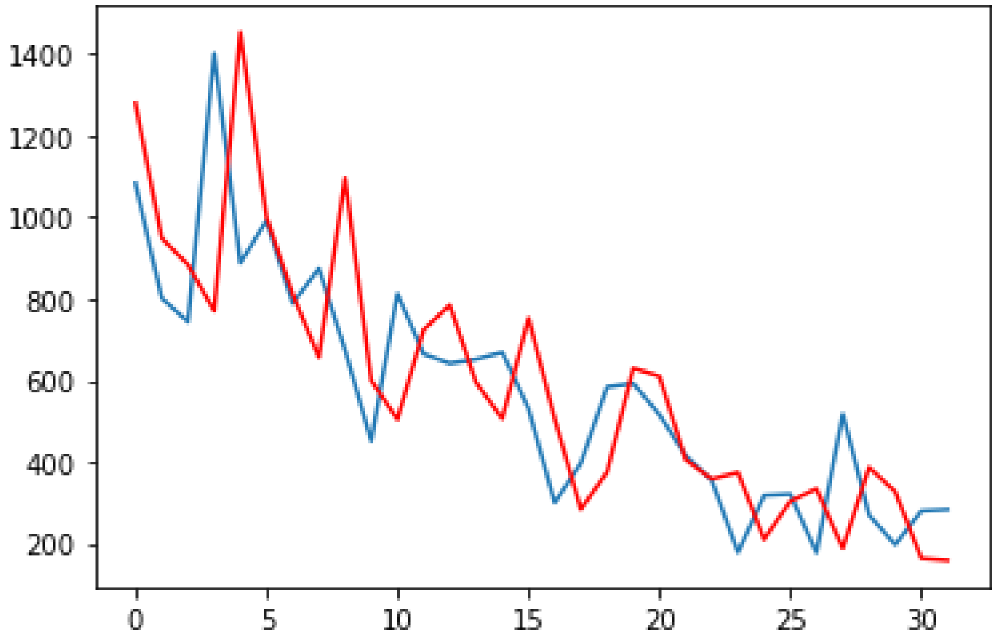
[16]:
```python
print("covid_Italy_cumulative:")
print()
test1, predictions1 = prediction(covid_Italy_cumulative, 5)
print("covid_Italy_spread:")
print()
test2, predictions2 = prediction(covid_Italy_spread, 5)

paint_plot(test1, predictions1)
paint_plot(test2, predictions2)
```

```
covid_Italy_cumulative:

predicted=218571.982936, expected=218268.000000
predicted=219326.003417, expected=219070.000000
predicted=219986.841921, expected=219814.000000
predicted=220681.417397, expected=221216.000000
predicted=222719.961817, expected=222104.000000
predicted=223219.041402, expected=223096.000000
predicted=224047.599329, expected=223885.000000
predicted=224622.472388, expected=224760.000000
predicted=225822.026809, expected=225435.000000
predicted=226190.863070, expected=225886.000000
predicted=226468.384301, expected=226699.000000
predicted=227517.290426, expected=227364.000000
predicted=228199.706295, expected=228006.000000
```

```
predicted=228714.259933, expected=228658.000000
predicted=229265.719154, expected=229327.000000
predicted=230090.190263, expected=229858.000000
predicted=230460.412507, expected=230158.000000
predicted=230530.387227, expected=230555.000000
predicted=231001.536437, expected=231139.000000
predicted=231821.605209, expected=231732.000000
predicted=232423.636055, expected=232248.000000
predicted=232757.311962, expected=232664.000000
predicted=233076.686151, expected=233019.000000
predicted=233434.100751, expected=233197.000000
predicted=233481.367337, expected=233515.000000
predicted=233896.817991, expected=233836.000000
predicted=234244.338975, expected=234013.000000
predicted=234284.169114, expected=234531.000000
predicted=234996.433242, expected=234801.000000
predicted=235165.496736, expected=234998.000000
predicted=235237.830933, expected=235278.000000
predicted=235514.362022, expected=235561.000000
Test MSE: 54333.118
covid_Italy_spread:

predicted=1279.669679, expected=1083.000000
predicted=948.744518, expected=802.000000
predicted=884.292676, expected=744.000000
predicted=770.802956, expected=1402.000000
predicted=1453.540634, expected=888.000000
predicted=1000.051968, expected=992.000000
predicted=811.864995, expected=789.000000
predicted=656.874952, expected=875.000000
predicted=1096.199305, expected=675.000000
predicted=599.927619, expected=451.000000
predicted=504.506169, expected=813.000000
predicted=724.217505, expected=665.000000
predicted=785.139403, expected=642.000000
predicted=595.396006, expected=652.000000
predicted=506.440288, expected=669.000000
predicted=753.638930, expected=531.000000
predicted=506.289949, expected=300.000000
predicted=283.950550, expected=397.000000
predicted=374.916069, expected=584.000000
predicted=630.222282, expected=593.000000
predicted=610.103408, expected=516.000000
predicted=405.965343, expected=416.000000
predicted=357.961096, expected=355.000000
predicted=374.343446, expected=178.000000
predicted=210.254793, expected=318.000000
predicted=303.820930, expected=321.000000
```

```
predicted=334.275868, expected=177.000000
predicted=188.386756, expected=518.000000
predicted=386.900417, expected=270.000000
predicted=327.038344, expected=197.000000
predicted=163.517737, expected=280.000000
predicted=158.511569, expected=283.000000
Test MSE: 49910.501
```

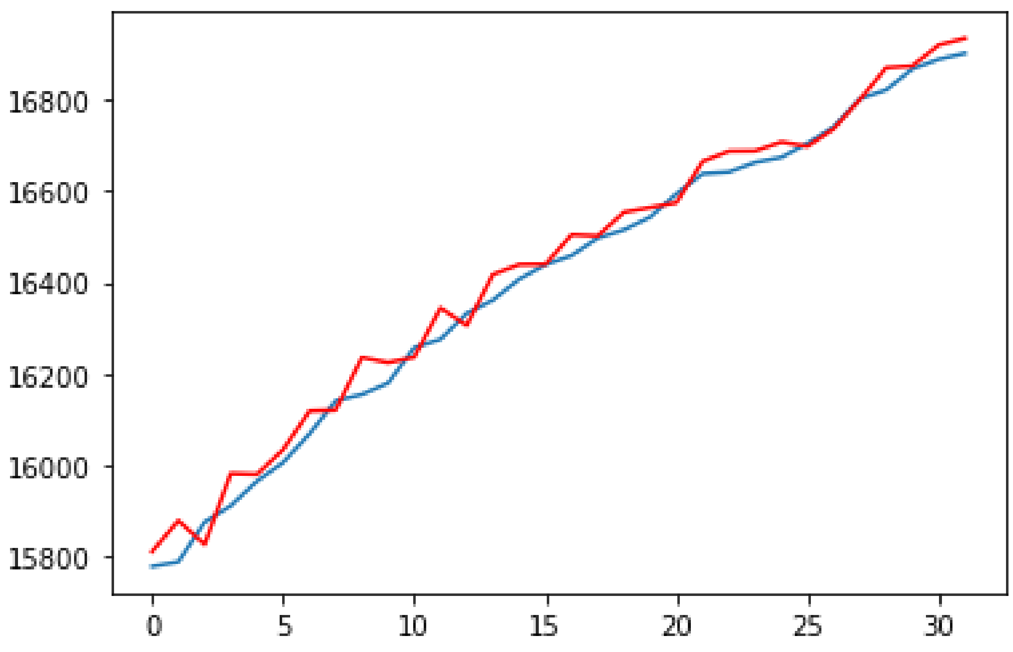


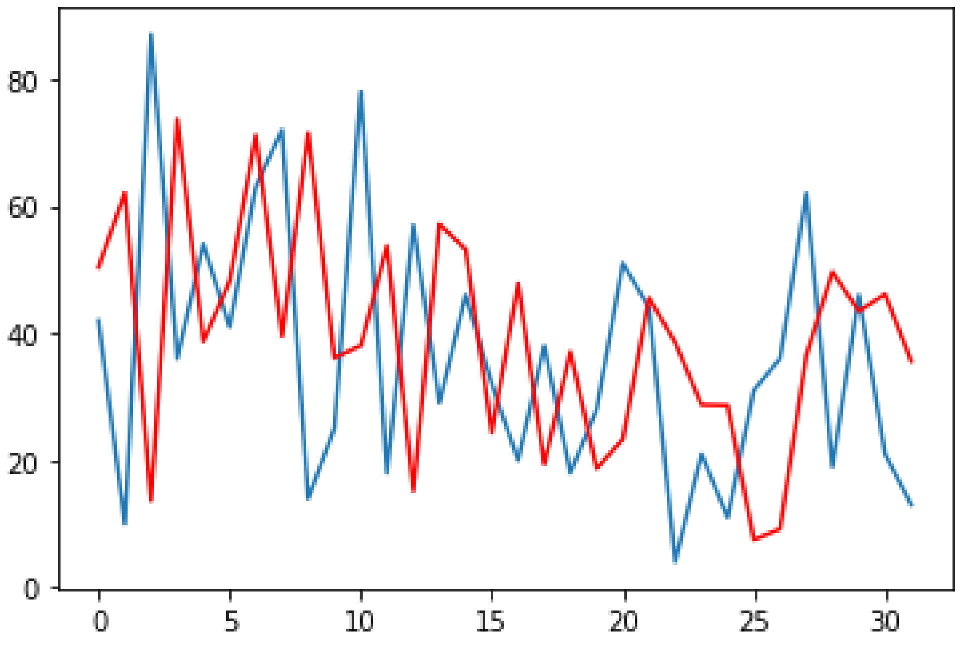


[17]:
```
print("covid_Austria_cumulative:")
print()
test1, predictions1 = prediction(covid_Austria_cumulative, 10)
print("covid_Austria_spread:")
print()
test2, predictions2 = prediction(covid_Austria_spread, 5)

paint_plot(test1, predictions1)
paint_plot(test2, predictions2)
```

```
covid_Austria_cumulative:

predicted=15809.859416, expected=15777.000000
predicted=15877.472073, expected=15787.000000
predicted=15825.112102, expected=15874.000000
predicted=15980.709803, expected=15910.000000
predicted=15979.052388, expected=15964.000000
predicted=16033.429473, expected=16005.000000
predicted=16117.919003, expected=16068.000000
predicted=16119.903103, expected=16140.000000
predicted=16235.336223, expected=16154.000000
predicted=16224.136776, expected=16179.000000
predicted=16235.811751, expected=16257.000000
predicted=16344.030409, expected=16275.000000
predicted=16305.345946, expected=16332.000000
predicted=16417.404280, expected=16361.000000
predicted=16439.338515, expected=16407.000000
predicted=16440.001333, expected=16439.000000
predicted=16504.346317, expected=16459.000000
predicted=16501.875553, expected=16497.000000
predicted=16553.768434, expected=16515.000000
predicted=16563.897750, expected=16543.000000
predicted=16574.535597, expected=16594.000000
predicted=16665.122439, expected=16638.000000
predicted=16687.652217, expected=16642.000000
predicted=16688.113026, expected=16663.000000
predicted=16707.772325, expected=16674.000000
predicted=16698.227415, expected=16705.000000
predicted=16736.740922, expected=16741.000000
predicted=16801.209524, expected=16803.000000
predicted=16870.540343, expected=16822.000000
predicted=16874.444652, expected=16868.000000
predicted=16920.348088, expected=16889.000000
predicted=16934.912498, expected=16902.000000
Test MSE: 1653.488
```

```
covid_Austria_spread:

predicted=50.374160, expected=42.000000
predicted=62.088155, expected=10.000000
predicted=13.650410, expected=87.000000
predicted=73.689508, expected=36.000000
predicted=38.731029, expected=54.000000
predicted=48.152401, expected=41.000000
predicted=71.161839, expected=63.000000
predicted=39.504148, expected=72.000000
predicted=71.484118, expected=14.000000
predicted=36.091896, expected=25.000000
predicted=38.043564, expected=78.000000
predicted=53.738478, expected=18.000000
predicted=15.194383, expected=57.000000
predicted=57.123430, expected=29.000000
predicted=53.188065, expected=46.000000
predicted=24.346549, expected=32.000000
predicted=47.843137, expected=20.000000
predicted=19.417939, expected=38.000000
predicted=37.067827, expected=18.000000
predicted=18.709233, expected=28.000000
predicted=23.340977, expected=51.000000
predicted=45.535144, expected=44.000000
predicted=38.508419, expected=4.000000
predicted=28.722475, expected=21.000000
predicted=28.627374, expected=11.000000
predicted=7.484228, expected=31.000000
predicted=9.207258, expected=36.000000
predicted=36.477808, expected=62.000000
predicted=49.636297, expected=19.000000
predicted=43.455216, expected=46.000000
predicted=46.140657, expected=21.000000
predicted=35.572272, expected=13.000000
Test MSE: 870.275
```





0.5 Conclusion

As we can see ARIMA worked the best for the smooth functions. In `Italy_spread` and `Austria_spread` it didn't perform very well. The development of covid-19 in these countries were pretty wild and ARIMA can't handle it, as we mentioned upper it is mainly cause by lack of data.

One main notice: we can see that our model performs really good on absolute number of infected (cumulative data). We could use our model to predict cumulative number of infected and substract last day from it and we would get more precise prediction of newly infected.

Potentially improvments: It would be really helpful to have bigger dataset. We could use some other model to produce some noise to add into our data. Another possible way would be to predict every for example 3 hours of newly infected by dividing numbers of daily numbers. If we somehow biggered our dataset we could use `LSTM` to get more precise predictions.

I hope that you like my work on this task and thank you for this opportunity.

Marek Jankola