Príloha 1

Táto príloha obsahuje všetky dáta, ktoré sme použili na analýzu časových radov v programe Python.

Tabuľka 1 – HDP EÚ

Obdobie	HDP (v mil. eur)
2000-Q1	1884715.6
2000-Q2	1957251.5
2000-Q3	1951853.0
2000-Q4	2075577.1
2001-Q1	1984352.3
2001-Q2	2055331.5
2001-Q3	2037277.9
2001-Q4	2164951.8
2002-Q1	2048256.7
2002-Q2	2131429.9
2002-Q3	2122394.6
2002-Q4	2236680.3
2003-Q1	2105904.0
2003-Q2	2176633.4
2003-Q3	2180363.3
2003-Q4	2304646.2
2004-Q1	2191658.0
2004-Q2	2286247.8
2004-Q3	2274317.9
2004-Q4	2415687.8
2005-Q1	2278185.6
2005-Q2	2390378.1
2005-Q3	2370817.1
2005-Q4	2521464.8
2006-Q1	2410420.3
2006-Q2	2518021.0
2006-Q3	2503997.4

2006-Q4	2679997.4
2007-Q1	2567338.1
2007-Q2	2672830.9
2007-Q3	2660313.3
2007-Q4	2838313.4
2008-Q1	2685733.3
2008-Q2	2806074.4
2008-Q3	2765893.9
2008-Q4	2827582.8
2009-Q1	2547419.9
2009-Q2	2631223.5
2009-Q3	2631680.9
2009-Q4	2777226.6
2010-Q1	2613149.4
2010-Q2	2737870.1
2010-Q3	2738341.4
2010-Q4	2890944.8
2011-Q1	2740511.9
2011-Q2	2839114.7
2011-Q3	2818883.0
2011-Q4	2925406.1
2012-Q1	2765866.0
2012-Q2	2838986.0
2012-Q3	2837642.4
2012-Q4	2949349.3
2013-Q1	2767340.5
2013-Q2	2879179.2
2013-Q3	2878363.1
2013-Q4	2995276.3
2014-Q1	2837332.1
2014-Q2	2932757.6
2014-Q3	2943878.0
2014-Q4	3069906.5
2015-Q1	2925233.2

2015-Q2	3040987.7
2015-Q3	3053267.5
2015-Q4	3195135.5
2016-Q1	3010516.7
2016-Q2	3141657.3
2016-Q3	3125658.5
2016-Q4	3274667.4
2017-Q1	3130746.7
2017-Q2	3251568.2
2017-Q3	3264799.0
2017-Q4	3428931.8
2018-Q1	3252475.0
2018-Q2	3371000.2
2018-Q3	3365583.3
2018-Q4	3542418.6
2019-Q1	3370163.1
2019-Q2	3487097.8
2019-Q3	3503011.7
2019-Q4	3656818.0
2020-Q1	3350284.7
2020-Q2	3070788.3
2020-Q3	3405991.1
2020-Q4	3572625.6
2021-Q1	3372361.2
2021-Q2	3555054.8
2021-Q3	3660119.5
2021-Q4	3860405.1

Tabuľka 2 – Miera nezamestnanosti EÚ

Obdobie	Miera nezamestnanosti (v %)
2000-01	9.9
2000-02	10.1
2000-03	10.0
2000-04	9.7
2000-05	9.4
2000-06	9.3
2000-07	9.3
2000-08	9.1
2000-09	9.0
2000-10	9.1
2000-11	9.1
2000-12	9.1
2001-01	9.4
2001-02	9.6
2001-03	9.5
2001-04	9.3
2001-05	9.1
2001-06	9.0
2001-07	9.0
2001-08	9.0
2001-09	8.9
2001-10	9.1
2001-11	9.2
2001-12	9.4
2002-01	9.7
2002-02	9.9
2002-03	9.9
2002-04	9.6
2002-05	9.4
2002-06	9.4

2002-07	9.5
2002-08	9.4
2002-09	9.3
2002-10	9.5
2002-11	9.6
2002-12	9.7
2003-01	10.1
2003-02	10.3
2003-03	10.3
2003-04	10.0
2003-05	9.7
2003-06	9.6
2003-07	9.6
2003-08	9.6
2003-09	9.5
2003-10	9.7
2003-11	9.8
2003-12	10.0
2004-01	10.6
2004-02	10.5
2004-03	10.5
2004-04	10.1
2004-05	9.8
2004-06	9.7
2004-07	9.5
2004-08	9.7
2004-09	9.8
2004-10	9.9
2004-11	10.0
2004-12	10.0
2005-01	10.3
2005-02	10.4
2005-03	10.3
2005-04	9.9

2005-05	9.7
2005-06	9.5
2005-07	9.2
2005-08	9.3
2005-09	9.4
2005-10	9.6
2005-11	9.5
2005-12	9.5
2006-01	9.8
2006-02	9.7
2006-03	9.5
2006-04	9.0
2006-05	8.6
2006-06	8.5
2006-07	8.3
2006-08	8.3
2006-09	8.4
2006-10	8.4
2006-11	8.4
2006-12	8.3
2007-01	8.4
2007-02	8.4
2007-03	8.0
2007-04	7.6
2007-05	7.5
2007-06	7.3
2007-07	7.2
2007-08	7.2
2007-09	7.2
2007-10	7.2
2007-11	7.3
2007-12	7.4
2008-01	7.6
2008-02	7.5

2008-03	7.4
2008-04	7.3
2008-05	7.1
2008-06	7.2
2008-07	6.9
2008-08	7.0
2008-09	7.1
2008-10	7.3
2008-11	7.6
2008-12	8.0
2009-01	8.8
2009-02	9.3
2009-03	9.4
2009-04	9.3
2009-05	9.1
2009-06	9.1
2009-07	9.2
2009-08	9.2
2009-09	9.5
2009-10	9.6
2009-11	9.8
2009-12	10.0
2010-01	10.7
2010-02	10.8
2010-03	10.6
2010-04	10.3
2010-05	10.1
2010-06	9.8
2010-07	9.7
2010-08	9.6
2010-09	9.8
2010-10	10.0
2010-11	10.1
2010-12	10.1

2011-01	10.5
2011-02	10.5
2011-03	10.4
2011-04	10.0
2011-05	9.8
2011-06	9.7
2011-07	9.7
2011-08	9.8
2011-09	9.9
2011-10	10.2
2011-11	10.5
2011-12	10.6
2012-01	11.2
2012-02	11.3
2012-03	11.3
2012-04	11.1
2012-05	10.9
2012-06	10.8
2012-07	10.8
2012-08	10.7
2012-09	10.9
2012-10	11.3
2012-11	11.5
2012-12	11.5
2013-01	12.3
2013-02	12.4
2013-03	12.1
2013-04	11.9
2013-05	11.6
2013-06	11.3
2013-07	11.2
2013-08	11.1
2013-09	11.3
2013-10	11.4

2013-11	11.6
2013-12	11.4
2014-01	12.0
2014-02	12.0
2014-03	11.7
2014-04	11.3
2014-05	11.0
2014-06	10.6
2014-07	10.5
2014-08	10.5
2014-09	10.7
2014-10	10.8
2014-11	11.1
2014-12	10.7
2015-01	11.1
2015-02	11.2
2015-03	11.0
2015-04	10.6
2015-05	10.3
2015-06	10.1
2015-07	9.7
2015-08	9.6
2015-09	9.7
2015-10	9.9
2015-11	9.9
2015-12	9.8
2016-01	10.0
2016-02	10.1
2016-03	9.9
2016-04	9.5
2016-05	9.3
2016-06	9.1
2016-07	8.9
2016-08	8.8

2016-09	8.9
2016-10	9.0
2016-11	9.1
2016-12	8.8
2017-01	9.1
2017-02	9.1
2017-03	8.9
2017-04	8.5
2017-05	8.3
2017-06	8.0
2017-07	8.0
2017-08	7.9
2017-09	7.9
2017-10	8.0
2017-11	7.9
2017-12	7.8
2018-01	8.2
2018-02	8.1
2018-03	7.9
2018-04	7.7
2018-05	7.3
2018-06	7.2
2018-07	7.0
2018-08	7.0
2018-09	7.1
2018-10	7.2
2018-11	7.2
2018-12	7.1
2019-01	7.5
2019-02	7.4
2019-03	7.2
2019-04	7.0
2019-05	6.7
2019-06	6.5

2019-07	6.6
2019-08	6.6
2019-09	6.6
2019-10	6.7
2019-11	6.7
2019-12	6.7
2020-01	7.0
2020-02	7.0
2020-03	6.7
2020-04	6.7
2020-05	6.8
2020-06	7.2
2020-07	7.7
2020-08	7.8
2020-09	7.8
2020-10	7.5
2020-11	7.3
2020-12	7.4
2021-01	7.9
2021-02	7.9
2021-03	7.7
2021-04	7.5
2021-05	7.2
2021-06	6.9
2021-07	6.8
2021-08	6.8
2021-09	6.6
2021-10	6.6
2021-11	6.4
2021-12	6.3
2022-01	6.6
2022-02	6.4

Tabuľka 3 – S&P 500 Index

Date	Close
22.3.2021	3940,59
23.3.2021	3910,52
24.3.2021	3889,14
25.3.2021	3909,52
26.3.2021	3974,54
29.3.2021	3971,09
30.3.2021	3958,55
31.3.2021	3972,89
1.4.2021	4019,87
5.4.2021	4077,91
6.4.2021	4073,94
7.4.2021	4079,95
8.4.2021	4097,17
9.4.2021	4128,8
12.4.2021	4127,99
13.4.2021	4141,59
14.4.2021	4124,66
15.4.2021	4170,42
16.4.2021	4185,47
19.4.2021	4163,26
20.4.2021	4134,94
21.4.2021	4173,42
22.4.2021	4134,98
23.4.2021	4180,17
26.4.2021	4187,62
27.4.2021	4186,72
28.4.2021	4183,18
29.4.2021	4211,47
30.4.2021	4181,17
3.5.2021	4192,66
4.5.2021	4164,66

5.5.2021	4167,59
6.5.2021	4201,62
7.5.2021	4232,6
10.5.2021	4188,43
11.5.2021	4152,1
12.5.2021	4063,04
13.5.2021	4112,5
14.5.2021	4173,85
17.5.2021	4163,29
18.5.2021	4127,83
19.5.2021	4115,68
20.5.2021	4159,12
21.5.2021	4155,86
24.5.2021	4197,05
25.5.2021	4188,13
26.5.2021	4195,99
27.5.2021	4200,88
28.5.2021	4204,11
1.6.2021	4202,04
2.6.2021	4208,12
3.6.2021	4192,85
4.6.2021	4229,89
7.6.2021	4226,52
8.6.2021	4227,26
9.6.2021	4219,55
10.6.2021	4239,18
11.6.2021	4247,44
14.6.2021	4255,15
15.6.2021	4246,59
16.6.2021	4223,7
17.6.2021	4221,86
18.6.2021	4166,45
21.6.2021	4224,79
22.6.2021	4246,44

23.6.2021	4241,84
24.6.2021	4266,49
25.6.2021	4280,7
28.6.2021	4290,61
29.6.2021	4291,8
30.6.2021	4297,5
1.7.2021	4319,94
2.7.2021	4352,34
6.7.2021	4343,54
7.7.2021	4358,13
8.7.2021	4320,82
9.7.2021	4369,55
12.7.2021	4384,63
13.7.2021	4369,21
14.7.2021	4374,3
15.7.2021	4360,03
16.7.2021	4327,16
19.7.2021	4258,49
20.7.2021	4323,06
21.7.2021	4358,69
22.7.2021	4367,48
23.7.2021	4411,79
26.7.2021	4422,3
27.7.2021	4401,46
28.7.2021	4400,64
29.7.2021	4419,15
30.7.2021	4395,26
2.8.2021	4387,16
3.8.2021	4423,15
4.8.2021	4402,66
5.8.2021	4429,1
6.8.2021	4436,52
9.8.2021	4432,35
10.8.2021	4436,75

11.8.2021	4447,7
12.8.2021	4460,83
13.8.2021	4468
16.8.2021	4479,71
17.8.2021	4448,08
18.8.2021	4400,27
19.8.2021	4405,8
20.8.2021	4441,67
23.8.2021	4479,53
24.8.2021	4486,23
25.8.2021	4496,19
26.8.2021	4470
27.8.2021	4509,37
30.8.2021	4528,79
31.8.2021	4522,68
1.9.2021	4524,09
2.9.2021	4536,95
3.9.2021	4535,43
7.9.2021	4520,03
8.9.2021	4514,07
9.9.2021	4493,28
10.9.2021	4458,58
13.9.2021	4468,73
14.9.2021	4443,05
15.9.2021	4480,7
16.9.2021	4473,75
17.9.2021	4432,99
20.9.2021	4357,73
21.9.2021	4354,19
22.9.2021	4395,64
23.9.2021	4448,98
24.9.2021	4455,48
27.9.2021	4443,11
28.9.2021	4352,63

29.9.2021	4359,46
30.9.2021	4307,54
1.10.2021	4357,04
4.10.2021	4300,46
5.10.2021	4345,72
6.10.2021	4363,55
7.10.2021	4399,76
8.10.2021	4391,34
11.10.2021	4361,19
12.10.2021	4350,65
13.10.2021	4363,8
14.10.2021	4438,26
15.10.2021	4471,37
18.10.2021	4486,46
19.10.2021	4519,63
20.10.2021	4536,19
21.10.2021	4549,78
22.10.2021	4544,9
25.10.2021	4566,48
26.10.2021	4574,79
27.10.2021	4551,68
28.10.2021	4596,42
29.10.2021	4605,38
1.11.2021	4613,67
2.11.2021	4630,65
3.11.2021	4660,57
4.11.2021	4680,06
5.11.2021	4697,53
8.11.2021	4701,7
9.11.2021	4685,25
10.11.2021	4646,71
11.11.2021	4649,27
12.11.2021	4682,85
15.11.2021	4682,8

16.11.2021	4700,9
17.11.2021	4688,67
18.11.2021	4704,54
19.11.2021	4697,96
22.11.2021	4682,94
23.11.2021	4690,7
24.11.2021	4701,46
26.11.2021	4594,62
29.11.2021	4655,27
30.11.2021	4567
1.12.2021	4513,04
2.12.2021	4577,1
3.12.2021	4538,43
6.12.2021	4591,67
7.12.2021	4686,75
8.12.2021	4701,21
9.12.2021	4667,45
10.12.2021	4712,02
13.12.2021	4668,97
14.12.2021	4634,09
15.12.2021	4709,85
16.12.2021	4668,67
17.12.2021	4620,64
20.12.2021	4568,02
21.12.2021	4649,23
22.12.2021	4696,56
23.12.2021	4725,79
27.12.2021	4791,19
28.12.2021	4786,35
29.12.2021	4793,06
30.12.2021	4778,73
31.12.2021	4766,18
3.1.2022	4796,56
4.1.2022	4793,54

5.1.2022	4700,58
6.1.2022	4696,05
7.1.2022	4677,03
10.1.2022	4670,29
11.1.2022	4713,07
12.1.2022	4726,35
13.1.2022	4659,03
14.1.2022	4662,85
18.1.2022	4577,11
19.1.2022	4532,76
20.1.2022	4482,73
21.1.2022	4397,94
24.1.2022	4410,13
25.1.2022	4356,45
26.1.2022	4349,93
27.1.2022	4326,51
28.1.2022	4431,85
31.1.2022	4515,55
1.2.2022	4546,54
2.2.2022	4589,38
3.2.2022	4477,44
4.2.2022	4500,53
7.2.2022	4483,87
8.2.2022	4521,54
9.2.2022	4587,18
10.2.2022	4504,08
11.2.2022	4418,64
14.2.2022	4401,67
15.2.2022	4471,07
16.2.2022	4475,01
17.2.2022	4380,26
18.2.2022	4348,87
22.2.2022	4304,76
23.2.2022	4225,5

24.2.2022	4288,7
25.2.2022	4384,65
28.2.2022	4373,94
1.3.2022	4306,26
2.3.2022	4386,54
3.3.2022	4363,49
4.3.2022	4328,87
7.3.2022	4201,09
8.3.2022	4170,7
9.3.2022	4277,88
10.3.2022	4259,52
11.3.2022	4204,31
14.3.2022	4173,11
15.3.2022	4262,45
16.3.2022	4357,86
17.3.2022	4411,67
18.3.2022	4463,12
21.3.2022	4461,18

Príloha 2

Táto príloha obsahuje príkazy použité na modelovanie časových radov v programe Python aj s popisom.

Výpis kódu 1 pre analýzu HDP EÚ

```
#inštalácia a načítanie knižníc
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.metrics import median_absolute_error, mean_squared_log_error
%matplotlib inline
```

```
import itertools
import warnings
warnings.filterwarnings("ignore")
plt.style.use('ggplot')
#načítanie dát s hlavičkou
raw_csv_data = pd.read_csv("HDP_EU_NEW.csv", delimiter = ',')
data = raw_csv_data
#vypísanie najstarších dát časového radu
data.head()
#vypísanie najnovších dát časového radu
data.tail()
#vypísanie základných štatistických charakteristík časového radu
data.describe()
#vykreslenie grafu časového radu
plt.figure(figsize=[12, 5]);
data.plot(figsize = (12, 5), legend = True, color='g')
plt.title('HDP EÚ')
plt.ylabel('HDP')
plt.xlabel('date')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()

#overenie stacionarity - ADF test
ad_fuller_result = adfuller(data['HDP'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
#časový rad bol nestacionárny - diferencia prvého rádu časového radu
data['HDP First Difference'] = data['HDP'] - data['HDP'].shift(1)
data.dropna(subset = ["HDP First Difference"], inplace=True)
data.head(
#opätovné overenie stacionarity časového radu
ad_fuller_result = adfuller(data['HDP First Difference'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
#vykreslenie ACF a PACF korelogramu diferencovaných dát
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
fig = plt.figure(figsize=(14,6))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data['HDP First Difference'].dropna(),lags=20,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data['HDP First Difference'].dropna(),lags=20,ax=ax2)
```

```
#ARIMA model
model = ARIMA(data['HDP'], order=(4,1,3))
result = model.fit (disp=0)
print(result.summary())
#test nezávislostí reziduí - Ljung-Box test
sm.stats.acorr_ljungbox(result.resid, lags=[20], return_df=True)
#rozdelenie časového radu na tréningové dáta a testovacie dáta
train = data[:int(0.95*(len(data)))]
test = data[int(0.95*(len(data))):]
start=len(train)
end=len(train)+len(test)-1
#vytvorenie predikcie
predictions = result.predict(start=start, end=end, dynamic=False, typ='levels').rename('ARIMA Predictions')
#výpočet mier presnosti vyrovnania, resp. priemerné charakteristiky rezíduí (R2-score, MAE, MSE, RMSE, MAPE)
evaluation_results = pd.DataFrame({'r2_score': r2_score(test['HDP'], predictions)}, index=[0])
evaluation_results['mean_absolute_error'] = mean_absolute_error(test['HDP'], predictions)
evaluation_results['mean_squared_error'] = mean_squared_error(test['HDP'], predictions)
evaluation_results['root_mean_squared_error'] = np.sqrt(mean_squared_error(test['HDP'], predictions))
evaluation_results['mean_absolute_percentage_error'] = np.mean(np.abs(predictions - test['HDP'])
                                                    /np.abs(test['HDP']))*100
evaluation_results
#predikcia pomocou ARIMA modelu a vykreslenie predikcie na najbližších 12 období(12 štvrťrokou)
result.plot_predict(start = 1, end = 100, dynamic=False)
plt.title('HDP EÚ', size = 16)
plt.ylabel('Million euro', size=12)
plt.legend(loc='upper left', prop={'size': 12})
ax.axes.get_xaxis().set_visible(True)
plt.show()

#SARIMA model
best_model = SARIMAX(data['HDP'], order=(15, 1, 12), seasonal_order=(0, 0, 0, 4)).fit(dis=-1)
print(best_model.summary())
#test nezávislostí rezíduí - Ljung-Box test
sm.stats.acorr_ljungbox(best_model.resid, lags=[20], return_df=True)
#rozdelenie časového radu na tréningové dáta a testovacie dáta
train = data[:int(0.95*(len(data)))]
test = data[int(0.95*(len(data))):]
start=len(train)
```

```
end=len(train)+len(test)-1
#vytvorenie predikcie
predictions=best_model.predict(start=start, end=end, dynamic=False,
typ='levels').rename('ARIMA Predictions')
#výpočet mier presnosti vyrovnania, resp. priemerné charakteristiky rezíduí (R2-score, MAE,
MSE, RMSE, MAPE)
evaluation_results = pd.DataFrame({'r2_score': r2_score(test['HDP'], predictions)}, index=[0])
evaluation_results['mean_absolute_error'] = mean_absolute_error(test['HDP'], predictions)
evaluation_results['mean_squared_error'] = mean_squared_error(test['HDP'], predictions)
evaluation_results['root_mean_squared_error'] = np.sqrt(mean_squared_error(test['HDP'],
predictions))
evaluation_results['mean_absolute_percentage_error'] = np.mean(np.abs(predictions -
test['HDP'])
                                                    /np.abs(test['HDP']))*100
evaluation_results
#Diagnostika rezíduí pomocou grafov
best_model.plot_diagnostics(figsize=(15,12));
#Predikcia pomocou SARIMA modelu na najbližších 12 období(12 štvrťrokou)
forecast_values = best_model.get_forecast(steps = 12)
#Intervaly spoľahlivosti prognózovaných hodnôt
forecast_ci = forecast_values.conf_int()
forecast_values.predicted_mean
#Vykreslenie predikcie na najbližšie 12 období(12 štvrťrokou) pomocou SARIMA modelu
ax = data.plot(x='date', y='HDP', figsize = (14, 6), legend = True, color='purple')
forecast_values.predicted_mean.plot(ax=ax, label='Forecast', figsize = (15, 5), grid=True)
ax.fill_between(forecast_ci.index,
                forecast_ci.iloc[: , 0],
                forecast_ci.iloc[: , 1], color='yellow', alpha = .5)
plt.title('HDP EÚ', size = 16)
plt.ylabel('Million euro', size=12)
plt.legend(loc='upper left', prop={'size': 12})
ax.axes.get_xaxis().set_visible(True)
```

Výpis kódu 2 pre analýzu mieru nezamestnanosti EÚ

```
#inštalácia a načítanie knižníc
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.metrics import median_absolute_error, mean_squared_log_error
```

```
%matplotlib inline
import itertools
import warnings
warnings.filterwarnings("ignore")
plt.style.use('ggplot')

#načítanie dát s hlavičkou
raw_csv_data = pd.read_csv("Unemployment rates.csv", delimiter = ',')
data = raw_csv_data
#vypísanie najstarších dát časového radu
data.head()
#vypísanie najnovších dát časového radu
data.tail()
#vypísanie základných štatistických charakteristík časového radu
data.describe()
#vykreslenie grafu časového radu
plt.figure(figsize=[250, 450]);
data.plot(x='date',y='Unemployment rates', figsize = (12, 5), legend = True, color='g')
plt.title('Miera nezamestnanosti EÚ')
plt.ylabel('MN v %')
plt.xlabel('date')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()


#overenie stacionarity - ADF test
ad_fuller_result = adfuller(data['Unemployment rates'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
#časový rad bol nestacionárny - diferencia prvého rádu časového radu
data['Unemployment rates First Difference'] = data['Unemployment rates'] -
data['Unemployment rates'].shift(1)
data.dropna(subset = ["Unemployment rates First Difference"], inplace=True)
data.head()
#vykreslenie ACF a PACF korelogramu diferencovaných dát
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
fig = plt.figure(figsize=(14,6))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data['Unemployment rates First
Difference'].dropna(),lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data['Unemployment rates First
Difference'].dropna(),lags=40,ax=ax2)
#dáta sú stále nestacionárne - diferencia druhého rádu časového radu
data['Unemployment rates Twice Difference'] = data['Unemployment rates First Difference'] -
data['Unemployment rates First Difference'].shift(1)
data.dropna(subset = ["Unemployment rates Twice Difference"], inplace=True)
data.head()
```

```
#opätovné overenie stacionarity časového radu
ad_fuller_result = adfuller(data['Unemployment rates Twice Difference'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
#sezónne diferencovanie
data['dif'] = data['Unemployment rates'].diff()
data['sdif'] = data['dif'].diff(12)
data.head(15)
#vykreslenie ACF a PACF korelogramu diferencovaných dát druhého rádu
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
fig = plt.figure(figsize=(14,6))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data['Unemployment rates Twice
Difference'].dropna(),lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data['Unemployment rates Twice
Difference'].dropna(),lags=40,ax=ax2)

#ARIMA model
model = ARIMA(data['Unemployment rates'], order=(11,2,11))
result = model.fit (disp=0)
print(result.summary())
#test nezávislostí reziduí - Ljung-Box test
sm.stats.acorr_ljungbox(result.resid, lags=[40], return_df=True)
#rozdelenie časového radu na tréningové dáta a testovacie dáta
train = data[:int(0.80*(len(data)))]
test = data[int(0.80*(len(data))):]
start=len(train)
end=len(train)+len(test)-1
#vytvorenie predikcie
predictions = result.predict(start=start, end=end, dynamic=False,
typ='levels').rename('SARIMA Predictions')
#výpočet mier presnosti vyrovnania, resp. priemerné charakteristiky reziduí (R2-score, MAE,
MSE, RMSE, MAPE)
evaluation_results = pd.DataFrame({'r2_score': r2_score(test['Unemployment rates'],
predictions)}, index=[0])
evaluation_results['mean_absolute_error'] = mean_absolute_error(test['Unemployment rates'],
predictions)
evaluation_results['mean_squared_error'] = mean_squared_error(test['Unemployment rates'],
predictions)
evaluation_results['root_mean_squared_error'] =
np.sqrt(mean_squared_error(test['Unemployment rates'], predictions))
evaluation_results['mean_absolute_percentage_error'] = np.mean(np.abs(predictions -
test['Unemployment rates'])
                                                    /np.abs(test['Unemployment rates']))*100
evaluation_results
#predikcia pomocou ARIMA
result.plot_predict(start = 2, end = 278, dynamic=False)
plt.title('Miera nezamestnanosti EÚ', size = 16)
```

```
plt.ylabel('MN v %', size=12)
plt.legend(loc='upper left', prop={'size': 8})
ax.axes.get_xaxis().set_visible(True)
plt.show()

#SARIMA model
best_model = SARIMAX(data['Unemployment rates'], order=(1, 2, 1), seasonal_order=(0, 0,
0, 12)).fit(dis=-1)
print(best_model.summary())
#test nezávislostí reziduí - Ljung-Box test
sm.stats.acorr_ljungbox(best_model.resid, lags=[40], return_df=True)
#rozdelenie časového radu na tréningové dáta a testovacie dáta
train = data[:int(0.80*(len(data)))]
test = data[int(0.80*(len(data))):]
start=len(train)
end=len(train)+len(test)-1
#vytvorenie predikcie
predictions = best_model.predict(start=start, end=end, dynamic=False,
typ='levels').rename('SARIMA Predictions')
#výpočet mier presnosti vyrovnania, resp. priemerné charakteristiky rezíduí (R2-score, MAE,
MSE, RMSE, MAPE)
evaluation_results = pd.DataFrame({'r2_score': r2_score(test['Unemployment rates'],
predictions)}, index=[0])
evaluation_results['mean_absolute_error'] = mean_absolute_error(test['Unemployment rates'],
predictions)
evaluation_results['mean_squared_error'] = mean_squared_error(test['Unemployment rates'],
predictions)
evaluation_results['root_mean_squared_error'] =
np.sqrt(mean_squared_error(test['Unemployment rates'], predictions))
evaluation_results['mean_absolute_percentage_error'] = np.mean(np.abs(predictions -
test['Unemployment rates'])
                                                    /np.abs(test['Unemployment rates']))*100
evaluation_results
#diagnostika rezíduí pomocou grafou
best_model.plot_diagnostics(figsize=(15,12));
#predikcia na najbližších 12 mesiacov pomocou SARIMA modelu
forecast_values = best_model.get_forecast(steps = 12)
forecast_ci = forecast_values.conf_int()
forecast_values.predicted_mean
#vykreslenie predikcie na najbližších 12 mesiacov pomocou SARIMA modelu
ax = data.plot(x = 'date', y = 'Unemployment rates', figsize = (14, 6), legend = True,
color='purple')
forecast_values.predicted_mean.plot(ax=ax, label='Forecast', figsize = (15, 6), grid=True)
ax.fill_between(forecast_ci.index,
                forecast_ci.iloc[: , 0],
                forecast_ci.iloc[: , 1], color='yellow', alpha = .5)
plt.title('Miera nezamestnanosti EÚ', size = 16)
plt.ylabel('MN v %', size=12)
plt.legend(loc='upper left', prop={'size': 12})
ax.axes.get_xaxis().set_visible(True)
```

Výpis kódu 3 pre analýzu S&P 500 Indexu

```
#inštalácia a načítanie knižníc
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.metrics import median_absolute_error, mean_squared_log_error
%matplotlib inline
import itertools
import warnings
warnings.filterwarnings("ignore")
plt.style.use('ggplot')

#načítanie dát s hlavičkou
raw_csv_data = pd.read_csv("SP500_P.csv", delimiter = ',')
data = raw_csv_data.copy()
#vypísanie najstarších dát v časovom rade
data.head()
#vypísanie najnovších dát časového radu
data.tail()
#vypísanie základných štatistických charakteristík časového radu
data.describe()
#vykreslenie grafu časového radu
plt.figure(figsize=[12, 5]);
data.plot(x='date',y='spx', figsize = (12, 5), legend = True, color='g')
plt.title('S&P 500 Index')
plt.ylabel('Close price')
plt.xlabel('date')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()

#overenie stacionarity - ADF test
ad_fuller_result = adfuller(data['spx'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
#časový rad bol nestacionárny - diferencia prvého rádu časového radu
data['spx First Difference'] = data['spx'] - data['spx'].shift(1)
data.dropna(subset = ["spx First Difference"], inplace=True)
data.head()
#sezónne diferencovanie
data['dif'] = data['spx'].diff()
data['sdif'] = data['dif'].diff(1)
data.head(15)
```

```
#opätovné overenie stacionarity časového radu
ad_fuller_result = adfuller(data['spx First Difference'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
#vykreslenie ACF a PACF korelogramu diferencovaných dát
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
fig = plt.figure(figsize=(14,6))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data['spx First Difference'].dropna(),lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data['spx First Difference'].dropna(),lags=40,ax=ax2)


#ARIMA model
model = ARIMA(data['spx'], order=(17,1,17))
result = model.fit (disp=0)
print(result.summary())
#test nezávislostí reziduí - Ljung-Box test
sm.stats.acorr_ljungbox(result.resid, lags=[40], return_df=True)
#rozdelenie časového radu na tréningové dáta a testovacie dáta
train = data[:int(0.80*(len(data)))]
test = data[int(0.80*(len(data))):]
start=len(train)
end=len(train)+len(test)-1
#vytvorenie predikcie
predictions = result.predict(start=start, end=end, dynamic=False,
typ='levels').rename('SARIMA Predictions')
#výpočet mier presnosti vyrovnania, resp. priemerné charakteristiky reziduí (R2-score, MAE,
MSE, RMSE, MAPE)
evaluation_results = pd.DataFrame({'r2_score': r2_score(test['spx'], predictions)}, index=[0])
evaluation_results['mean_absolute_error'] = mean_absolute_error(test['spx'], predictions)
evaluation_results['mean_squared_error'] = mean_squared_error(test['spx'], predictions)
evaluation_results['root_mean_squared_error'] = np.sqrt(mean_squared_error(test['spx'],
predictions))
evaluation_results['mean_absolute_percentage_error'] = np.mean(np.abs(predictions -
test['spx'])
                                                    /np.abs(test['spx']))*100
evaluation_results
#predikcia pomocou ARIMA na najbližších 30 dní
result.plot_predict(start = 1, end = 283, dynamic=False)
plt.title('S&P 500 Index', size = 16)
plt.ylabel('Close price', size=12)
plt.legend(loc='upper left', prop={'size': 8})
ax.axes.get_xaxis().set_visible(True)
plt.show()

#SARIMA model
best_model = SARIMAX(data['spx'], order=(17, 1, 17), seasonal_order=(0, 1, 0, 7)).fit(dis=-
1)
```

```
print(best_model.summary())
#test nezávislostí rezíduí - Ljung-Box test
sm.stats.acorr_ljungbox(best_model.resid, lags=[40], return_df=True)
#rozdelenie časového radu na tréningové dáta a testovacie dáta
train = data[:int(0.80*(len(data)))]
test = data[int(0.80*(len(data))):]
start=len(train)
end=len(train)+len(test)-1
#vytvorenie predikcie
predictions = best_model.predict(start=start, end=end, dynamic=False,
typ='levels').rename('SARIMA Predictions')
#výpočet mier presnosti vyrovnania, resp. priemerné charakteristiky rezíduí (R2-score, MAE,
MSE, RMSE, MAPE)
evaluation_results = pd.DataFrame({'r2_score': r2_score(test['spx'], predictions)}, index=[0])
evaluation_results['mean_absolute_error'] = mean_absolute_error(test['spx'], predictions)
evaluation_results['mean_squared_error'] = mean_squared_error(test['spx'], predictions)
evaluation_results['root_mean_squared_error'] = np.sqrt(mean_squared_error(test['spx'],
predictions))
evaluation_results['mean_absolute_percentage_error'] = np.mean(np.abs(predictions -
test['spx'])
                                                    /np.abs(test['spx']))*100
evaluation_results
#diagnostika rezíduí pomocou grafou
best_model.plot_diagnostics(figsize=(15,12));
#predikcia na najbližších 30 dní
forecast_values = best_model.get_forecast(steps = 30)
forecast_ci = forecast_values.conf_int()
forecast_values.predicted_mean
#vykreslenie predikcie na najbližších 30 dní pomocou SARIMA modelu
ax = data.plot(x = 'date', y = 'spx', figsize = (14, 6), legend = True, color='purple')
forecast_values.predicted_mean.plot(ax=ax, label='Forecast', figsize = (15, 6), grid=True)
ax.fill_between(forecast_ci.index,
                forecast_ci.iloc[: , 0],
                forecast_ci.iloc[: , 1], color='yellow', alpha = .5)
plt.title('S&P 500 Index', size = 16)
plt.ylabel('Close price', size=12)
plt.legend(loc='upper left', prop={'size': 12})
ax.axes.get_xaxis().set_visible(True)
```