Список использованных источников

1. Современные проблемы математики и вычислительной техники. – Брест : БрГТУ, 2003. – 298 с.
2. Neptune.ai [Электронный ресурс]. – Режим доступа: https://neptune.ai/blog/anomaly-detection-in-time-series. – Дата доступа: 30.10.2022.
3. Головко, В. А. Нейросетевые технологии обработки данных : учебное пособие / В. А. Головко, В. В. Краснопрошин. – Минск : Белорусский государственный университет, 2017. – 263 с. – (Классическое университетское издание). – ISBN 978-985-566-467-4. – EDN GLVGIE.
4. Головко В. А. Нейросетевые методы обработки хаотических процессов //VII Всероссийская научно-техническая конференция «Нейроинформатика. – 2005. – С. 43-91.

UDC 004.942

A COMPARISON OF THE COVID-19 MACHINE LEARNING AUTOMATION MODEL AND SPSS TIME SERIES

Hongxu Zhu, D.O. Petrov, V.S.Razumeichik
Brest State Technical University, Brest, Belarus, zhuhongxu08@gmail.com

This paper uses publicly available data on the prediction process of Covid-19 transmission in the world to attempt to predict the time series using the SPSS exponential Holt model and the Python ARIMA model. model model to predict the epidemic development trend and key nodes, quantitative analysis of the scale of the epidemic, scientific and reliable interval estimation of the original base and effective transmission rate of the epidemic and comparative analysis of different algorithms, providing an effective basis and guide for analysis, command and decision making in the prevention and control of the epidemic.

Predicting data at a range of points in time is a common activity in real life, and research fields such as agriculture, business, climate, military and medicine all contain large amounts of time series data. Time series forecasting refers to making predictions about the likely future values of a series based on the historical data of the series, as well as other relevant series that may have an impact on the outcome. There are many real-life time series forecasting problems, including voice analysis, noise cancellation and analysis of stock and futures markets, where the essence is to derive the value of the time series at T + 1 based on observations at the previous T moments. For time series prediction, we can use the traditional ARIMA model, or we can use the Holt model or other models based on time series. Nowadays, machine learning methods such as deep learning can also be used for time series prediction. We are going to introduce how to implement the covide-19 prediction of time series based on two different models.

In this paper, two types of time-series data software were used for fitting:

– Holt Model (SPSS);
– ARIMA Model (Python3.7).

There were 187,801 training samples prepared, and the sample data was split into a training set and a test set with 67 test indicators. The actual training data is in 3 columns (841 items filtered according to the test specified mediation and 822 items filtered for missing values):

Date: Time span 1 February 2020 - 19 May 2022;
Cases: Number of new diagnoses;
Cases_smoothed: number of new confirmed cases (7-day average).

ARIMA model

ARIMA model (full name: Autoregressive Integrated Moving Average model), also known as an Autoregressive Integrated Moving Average model, is one of the time series forecasting analysis methods.

widehat{y_t} = mu + phi_1*y_{t-1} + ... + phi_p*y_{t-p} + + heta_1*e_{t-1} + ... + heta_q*e_{t-q}

where phi denotes the coefficient of AR
heta denotes the coefficient of MA

p - represents the number of lags of the time series data itself used in the prediction model, also known as the AR/Auto-Regressive term

d - represents the number of orders of differencing required for the time-series data to be stable, also known as the Integrated term.

q - represents the number of lags of the prediction error used in the prediction model (lags), also called the MA/Moving Average term.

The Holt model is simple, reliable and easy to use and is a type of exponential smoothing model. It is particularly suitable for data that varies continuously over time and often tends to be used as a general model for trend series

$$St=\alpha Xt+(1-\alpha)(St-1+Tt-1), \quad (1)$$
$$Tt=\gamma(St-St-1)+(1-\gamma)Tt-1, \quad (2)$$
$$Xt'(m) = St + mTt. \quad (3)$$

Packages to be loaded (Python environment):

```
import pandas as pd
from pandas import datetime
from pandas import read_csv
from pandas import DataFrame
from statsmodels.tsa.arima.model import ARIMA
from pmdarima import auto_arima
from matplotlib import pyplot
import numpy as np
import warnings
from sklearn.preprocessing import MinMaxScaler
from pandas import read_csv
from pandas import datetime
from matplotlib import pyplot
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import warnings
from statsmodels.tools.sm_exceptions import Conver-
genceWarning
```

```
warnings.simplefilter('ignore', ConvergenceWarning)
from math import sqrt
from sklearn import metrics
```

1) Define the function that transforms a time series prediction problem into a supervised learning problem. The essence of time series forecasting is essentially the extrapolation of the value of the time series at time T + 1 from the observations at the previous T moments.

```
def series_to_supervised(in_data, tar_data, n_in=1,
dropnan=True, target_dep=False):
    n_vars = in_data.shape[1]
    cols, names = list(), list()
    if target_dep:
        i_start = 1
    else:
        i_start = 0
    for i in range(i_start, n_in + 1):
        cols.append(in_data.shift(i))
        names += [('%s(t-%d)' % (in_data.columns[j],
i)) for j in range(n_vars)]
     if target_dep:
         for i in range(n_in, -1, -1):
             cols.append(tar_data.shift(i))
             names += [('%s(t-%d)' % (tar_data.name,
i))]
     else:
         # put it all together
         cols.append(tar_data)
         names.append(tar_data.name)
     agg = pd.concat(cols, axis=1)
     agg.columns = names
     # drop rows with NaN values
     if dropnan:
         agg.dropna(inplace=True)
  return agg
```

2) Define functions for preparing data.

- Create a dataset:

```
dataset=series_to_supervised(pd.DataFrame(y_dataset),
y_dataset, 14)
```

- Slice and dice the training and test data:

```
X_train, X_test, y_train, y_test =
train_test_split(scaled_x, scaled_y, test_size=0.29,
shuffle=False)
```

3) Define the fitted ARIMA model and plot the residual error.

auto_arima_model = auto_arima(y_train,trace=True, supress_warnings=True)
arima_model_202 = ARIMA(y_train, order=(3,1,3)).fit()

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-3064.971, Time=0.48 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-3014.956, Time=0.04 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-3024.640, Time=0.07 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-3025.882, Time=0.12 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-3016.924, Time=0.03 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-3021.891, Time=0.30 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-3031.345, Time=0.47 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=-3065.792, Time=0.63 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-3080.621, Time=0.55 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-3032.006, Time=0.12 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=-3031.987, Time=0.71 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-3023.618, Time=0.15 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AIC=-3033.011, Time=0.17 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=-3060.117, Time=0.75 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=-3077.909, Time=0.25 sec

Best model:  ARIMA(3,1,1)(0,0,0)[0] intercept
Total fit time: 4.854 seconds
```

Figure 1 - ARIMA Data Fitting Results

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	583
Model:	ARIMA(3, 1, 3)	**Log Likelihood**	1594.842
Date:	Sun, 30 Oct 2022	**AIC**	-3175.683
Time:	14:15:37	**BIC**	-3145.118
Sample:	0	**HQIC**	-3163.768
	- 583		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5294	0.015	35.174	0.000	0.500	0.559
ar.L2	-0.5359	0.017	-31.751	0.000	-0.569	-0.503
ar.L3	0.9551	0.014	66.928	0.000	0.927	0.983
ma.L1	-0.4419	0.026	-17.021	0.000	-0.493	-0.391
ma.L2	0.4998	0.030	16.477	0.000	0.440	0.559
ma.L3	-0.7977	0.028	-28.214	0.000	-0.853	-0.742
sigma2	0.0002	7.01e-06	34.835	0.000	0.000	0.000

Ljung-Box (L1) (Q):	0.97	**Jarque-Bera (JB):**	2257.09
Prob(Q):	0.32	**Prob(JB):**	0.00
Heteroskedasticity (H):	9.82	**Skew:**	-0.56
Prob(H) (two-sided):	0.00	**Kurtosis:**	12.58

Figure 2 - ARIMA Predicted Results

4) Defining functions to visualise predictions.

```
print("R-Square",r2_score(y_test, predictions))
print("Correlation train", np.corrcoef(res_test, predictions)[0,1])
print("Correlation train", np.corrcoef(y_test, predictions)[0,1])
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

R-Square 0.987090118011517
Correlation train 0.9940699440018747
Correlation train 0.9940699440018748
Mean Absolute Error: 0.009938543636197363
Mean Squared Error: 0.0004559192498476831
```

Figure 3 - ARIMA Indicator of Prediction

```
df_2 = pd.DataFrame({'Actual test': y_test, 'ARIMA': predictions,})
df_2.index = dataset.index[len(dataset)-len(res_test):]
df_2.plot()
```

```
predicted=0.006911, expected=0.006005
predicted=0.004994, expected=0.005316
predicted=0.005440, expected=0.003788
predicted=0.003931, expected=0.004416
predicted=0.003830, expected=0.005708
predicted=0.005664, expected=0.005454
Test RMSE: 0.021
```

Figure 4 – ARIMA Indicator of RMSE

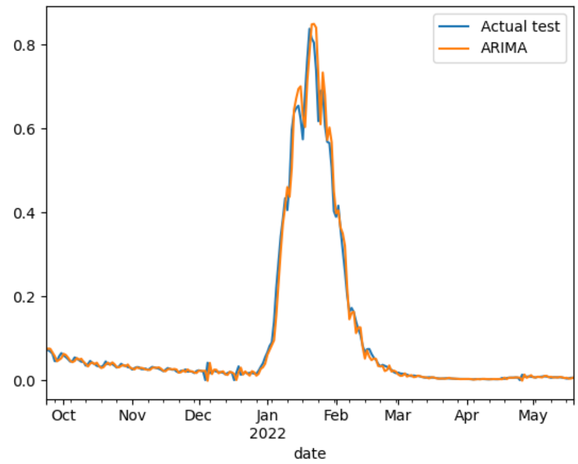

Figure 5 – ARIMA Prediction

The ARIMA model gives a result of 0.987 for R-Square, 0.0099 for MAE (Mean Absolute Error), 0.00046 for MSE (Mean Squared Error) and 0.021 for RMSE (Root Mean Squard Error).

Holt Model

1) Analysis of the original sequence diagram.

The graph shows that as the new cases are serially smooth and there are seasonal fluctuations.

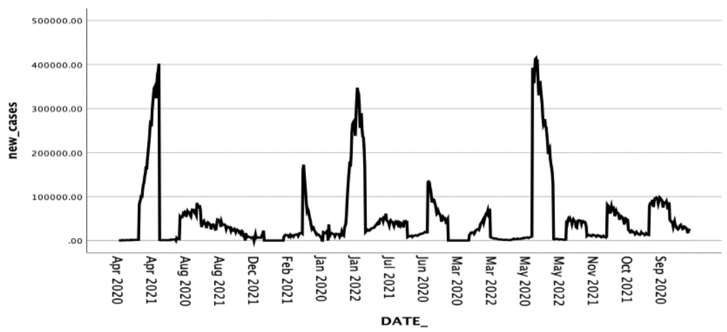

Figure 6 – Spss Diagram of sequence

2) Creating a model for fitting.

Forecasting the T+30 time of the Indian epidemic using the exponential Holt model.

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	10	25	50	75	90	95
Stationary R-	0.381		0.381	0.381	0.381	0.381	0.381	0.381	0.381	0.381	0.381
R-squared	0.990		0.990	0.990	0.990	0.990	0.990	0.990	0.990	0.990	0.990
RMSE	8013.992		8013.992	8013.992	8013.992	8013.992	8013.992	8013.992	8013.992	8013.992	8013.992
MAPE	48.773		48.773	48.773	48.773	48.773	48.773	48.773	48.773	48.773	48.773
MaxAPE	23621.334		23621.334	23621.334	23621.334	23621.334	23621.334	23621.334	23621.334	23621.334	23621.334
MAE	4229.124		4229.124	4229.124	4229.124	4229.124	4229.124	4229.124	4229.124	4229.124	4229.124
MaxAE	57438.106		57438.106	57438.106	57438.106	57438.106	57438.106	57438.106	57438.106	57438.106	57438.106
Normalized BIC	17.994		17.994	17.994	17.994	17.994	17.994	17.994	17.994	17.994	17.994

Figure 7 – Holt Model Fitting Results

3)Forecast data

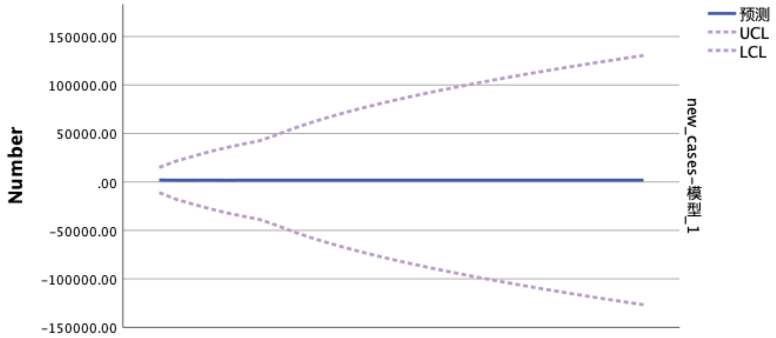

Figure 8 – Holt Model Prediction

The R-squared in the graph is 0.99 (close to '1', good fit) and the RMSE is 8013,992.

Comparative results		
Models	**Holt**	**ARIMA**
R-Square	0.987090118	0.990104793
RMSE	8013.992	0.021

Figure 9 – Comparing Results

Conclusions

Both of the above approaches were able to make and fit the time-series data for the new coronary pneumonia well, and a comparison of the two results between the R-squared and RMSE clearly shows that the ARIMA model fits relatively well and that the predicted data deviates less from the true data.

Reference

1. Cai Jie et al, Forecasting the development trend of novel coronavirus pneumonia epidemic in Wuhan based on SEIR model, Shandong Medicine
2. Jin Qixuan, Modeling and rational assessment for prediction of novel coronavirus pneumonia epidemic in China, Statistics and Decision Making