Time Series Database Preprocessing for Data Mining Using Python

Hussein Faroq Tayeb
Software Engineering Department,
Technology Faculty, Firat University,
Elazig, Turkey
husseintaeyb@gmail.com

Murat Karabatak
Software Engineering Department,
Technology Faculty, Firat University,
Elazig, Turkey
mkarabatak@firat.edu.tr

Cihan Varol
Computer Science Department,
Sam Houston State University,
Huntsville, Texas, USA
cxv007@shsu.edu

Abstract—**Data mining is an important method that we use for extracting meaningful information from data. Data preprocessing lays the groundwork for data mining yet most researchers unfortunately, ignore it. Before getting to the data mining stage, the target data set must be properly prepared. This paper describes steps followed for time series data preprocessing for data mining processes. The data that was used in the study is that of the minimum daily temperatures over 10 years (1981-1990) in the city of Melbourne, Australia. Python programming language is used to read the data and decompose it into trend, seasonality, and residue components. These components were plot and analyzed by removing the trend and seasonality to make the series stationary. Dicky Fuller's stationary test was done on the data. The test statistics results show that Dicky Fuller's null hypothesis can be rejected and the data is stationary. Hence, ready for the next step of data mining modeling processes.**

Keywords— Data preprocessing, time series database, Dicky-Fuller stationary test, trend, seasonality

I. INTRODUCTION

A time-series database (TSDB) is a collection of systems that are optimized to store and serve time series data by associating pairs of times and values. In other words, we can refer to TSDB as events that are tracked, monitored, downsampled, and aggregated over time[1]. The web page in [2] noted that data lifecycle, management, summarization, and large range scans of many records are the main properties that distinguish TSDB from the other data workloads. TSDBs also typically include methods and functions common to time-series data analysis like data retention policies, continuous queries, flexible time aggregations, etc.

According to the web page article in [3], TSDB is now considered as one of the most growing database categories because of its scalability and usability. One may say part of this growth is because time series data is at the core of the internet of things receiving sensor reading frequently from all types of devices. This study walks through the process of analyzing the characteristics of the daily minimum temperature dataset in the python programming language. The given data set is a time-series data, which describes the 10-year (1981-1990) daily minimum temperatures in Melbourne, Australia. Units are Celsius degrees and there are 3650 observations. The source of the data is considered to be the Australian Meteorological Bureau.

Data preprocessing lays the foundation for data mining but it is mostly skipped or ignored by most researchers as noted by Zhang and Lu in [4]. Nevertheless, before data modeling or knowledge discovery, the data set must be properly prepared. Time series analysis is the preparatory step that is needed to develop the future values or forecast of the series. It is through the series analysis we can understand different structures of the inherent nature of the series so that we can create an accurate forecast as highlighted by Agrawal et al. [5]. Accordingly, section II presents the literature, while section III described the research design, and section IV and V show the results and conclusion respectively.

II. RELATED WORK

Several studies about time series data forecasting and modeling have been done in the past, but unfortunately, many authors ignore the data preprocessing steps as explained in [4]. The study further explained that the main reason behind the ignoring of the data preprocessing steps is due to its perceived difficulties. Zeng et al. [6] in their study of data preprocessing system of depression recognition collected data from a college network routers and categorized the obtained data into seven basic databases. The data collection was based on the indices of the frequency of user browsing web pages, the frequency of using application categories, the time of surfing the internet, the time of surfing the internet, the total flow of surfing the internet, the number of Posts and the mails. The study perfume statistical test on the data to judge the depression tendency of a student using the warning model.

With the rapid growth of size and number of available databases for commercial and industrial use, Bin and Sun [7] use the dynamic concept hierarchy adjustment algorithm on census data in Chengyang and Laixi to adjust the obtained concept hierarchy on the attribute of housing construction cost. The authors evaluate the results to make the necessary preparation for mining the data. The study concluded that the application of concept hierarchy in census data has a very high learning value and vast marketplace space. Muntean and Onita

978-1-7281-6939-2/20/$31.00 ©2020 IEEE

[8] in their study of aggregation agent for preprocessing and forecasting time series data proposed an algorithm for currency data aggregation. The algorithm is designed in such a way that multiple relational database tables can be aggregated to optimize the forecast results of time series data. Moreover, the forecast is based on the aggregated dataset instead of forecasting currency exchange rates for each bank. The results of the study indicated that the proposed algorithm is suitable for the automatic aggregation of time series data.

The work of Zheng et al. [9] proposed a wind data preprocessing methodology that uses aggregated active power output and wind speed values to preprocess and filter wind data from a wind farm in China. The wind raw data properties are analyzed and the invalid data were categorized into five types. Unnatural and irrational data are considered as outliers in the study. Furthermore, the LOF algorithm is used to detect and remove these outliers. The performance evaluation of the algorithm has also been discussed. Bhoi et al. [10] highlight how current techniques of data duplication which are based on a string are time-consuming and heavy on memory. The study comes up with a system implemented on Hadoop to handle larger databases. The proposed system employed a data mining preprocessing technique to transformed row data into an understandable format.

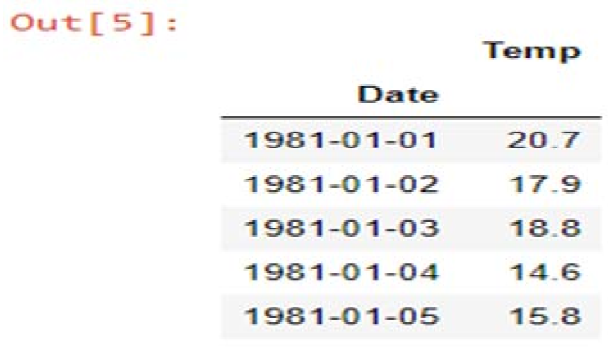
Due to tremendous advances and achievements in biotechnology, nowadays-biological data is being generated at a tremendous speed. Shamim et al. [11] propose a data framework for mining intelligent bio databases. The proposed system allows users to apply biodata analysis and data mining techniques to extract relevant, useful, valid, and actionable data from bio databases. The extracted data is preprocessed and the mined knowledge is integrated with an expert system to assist the users in their decision-making. In the work of Bolei and Cheng [12], they use data relativity analysis to align primitive data with application and user objective. The algorithm used in the study is a simple and rapid algorithm that gets data attributes that are related to a particular topic. The algorithm is very useful because with data preprocessing it does not need to do a complex calculation, hence, its efficiency is achieved.

Ho and Nguyen [13] in their study of visualization support for user-centered model selection in knowledge discovery in databases introduce a knowledge discovery system D2MS that supports model selection and integrated with visualization. The study emphasized user participation regarding model selection by providing the user with the ability to try different combinations of the algorithms, performance metrics, and visualization. Bian et al. [14] in their paper of database preprocessing using AHP put forward a method that makes multiple attributes decision problems hierarchical. The authors contend that database preprocessing with AHP provides stability, flexibility, and comparability.

III. Research Design

The data for the time series analysis is store in the .xlsx file and contains two attributes; the date and the measured temperature. We begin by importing all the Python libraries and functions such as pandas, numpy, matplotlib, statsmodel, and sklearn that we need to preprocess that data.

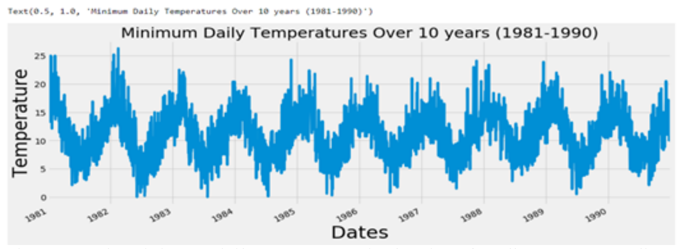
We use the read_excel() in the panda's package to read the time series dataset as a pandas data frame. Adding the parse_dates=['Date'] and index_col='Date' argument to parse the date column as a date field and the date as index respectively. We can see in the series in Figure 1, the Temp column is placed higher than the Date to imply that it is a series.Data Understanding



```
Out[5]:
                Temp
      Date
1981-01-01      20.7
1981-01-02      17.9
1981-01-03      18.8
1981-01-04      14.6
1981-01-05      15.8
```

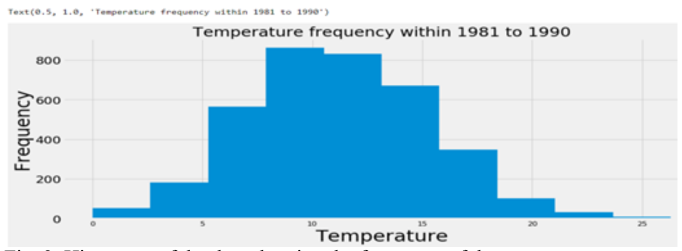
Fig. 1. First five rows of the series

The data visualization of the time series is done using the matplotlib to visualize the series. Since we set the date as the index of the data we use df.Temp.plot() to visualize the yearly minimum daily temperature as shown in Figure 2. Similarly, df.Temp.hist() is used to plot the histogram of the data which shows the frequencies of the temperatures as shown in Figure 3.



```
Text(0.5, 1.0, 'Minimum Daily Temperatures Over 10 years (1981-1990)')
```

Fig. 2. Yearly minimum daily temperature in the city of Melbourne, Australia



```
Text(0.5, 1.0, 'Temperature frequency within 1981 to 1990')
```

Fig. 3. Histogram of the data showing the frequency of the temperatures

We need to plan the autocorrelation and partial autocorrelation functions of the data because we are operating on time series. Autocorrelation is only the relationship of a series with its lags. If a series is significantly autocorrelated, that means, the past values of the series (lags) might improve predict the modern value. Partial autocorrelation likewise moves related data but it sends the true relationship of a series and its lag, except for the connection contributions from the intermediate lags [15]. We use plot_acf(df.Temp,lags=20) to plot the autocorrelation function of the data and plot_pacf(df.Temp,lags=20) to plot the partial autocorrelation as shown in Figure 4 and 5 respectively.

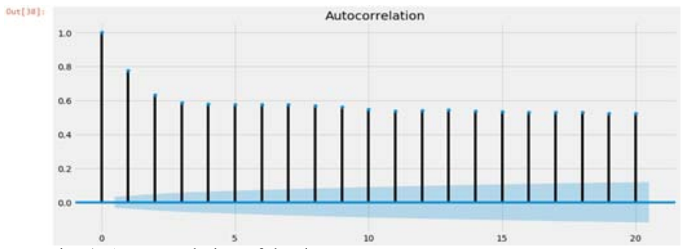


Fig. 4. Autocorrelation of the data.

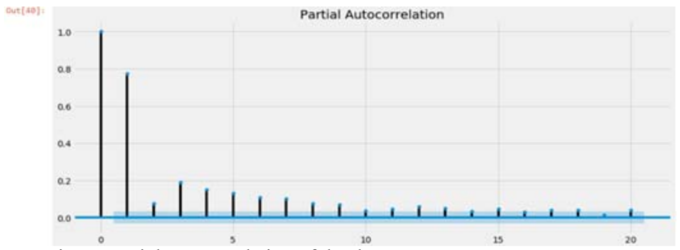
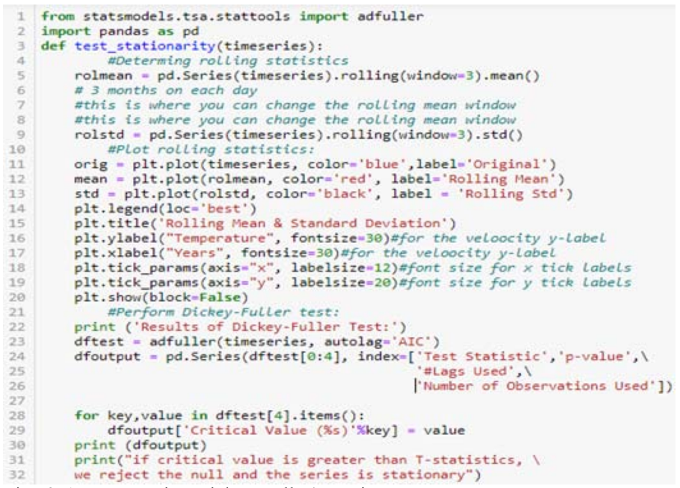


Fig. 5. Partial autocorrelation of the data



```
from statsmodels.tsa.stattools import adfuller
import pandas as pd
def test_stationarity(timeseries):
        #Determing rolling statistics
    rolmean = pd.Series(timeseries).rolling(window=3).mean()
    # 3 months on each day
    #this is where you can change the rolling mean window
    #this is where you can change the rolling mean window
    rolstd = pd.Series(timeseries).rolling(window=3).std()
        #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.ylabel("Temperature", fontsize=30)#for the veloocity y-label
    plt.xlabel("Years", fontsize=30)#for the veloocity y-label
    plt.tick_params(axis="x", labelsize=12)#font size for x tick labels
    plt.tick_params(axis="y", labelsize=20)#font size for y tick labels
    plt.show(block=False)
        #Perform Dickey-Fuller test:
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value',\
                                             '#Lags Used',\
                                             'Number of Observations Used'])

    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
    print("if critical value is greater than T-statistics, \
    we reject the null and the series is stationary")
```

Fig. 6. Argumentative Dickey Fuller's stationary test

A time-series data can be decomposed into a base level, trend, seasonality, and error components [16]. We can locate a trend in data when we see an increasing or decreasing slope in the series. On the other hand, seasonality is present in time-series data when we observed a particular repeated pattern among regular intervals such as the month of the year, the day of the month, weekdays or even time of the day. Now we use the seasonal_decompose function to decompose the data into a trend, seasonal, and residue components to see the data behavior. Figure 7 depicts the plot of the base level, trend, seasonality, and residue.

The stationary test is carried out using the argumentative Dickey-Fuller statistics to test the stationary of the series. The stationary test shows the statistical property of the series such as mean, standard mean, variance, etc. as shown in Figure 8. The most popular and suitable technique to stationaries the series is by differencing the series at least once until it becomes nearly stationary. The reason for making a series stationary is because it is relatively easy and the forecasts are more reliable. The Python codes in Figure 6 below show the steps followed to carry out the argumentative Dickey Fuller's stationary test on the data.

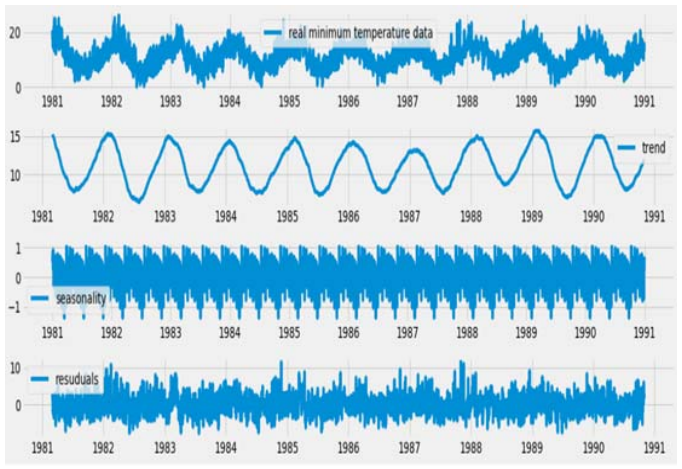


Fig. 7. Trend, seasonality and residue components of the ata

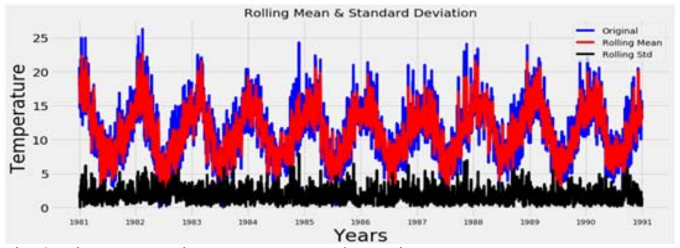


Fig. 8. Linear regression scattergram and equation

IV. Results and Discussion

The data from Figure 7 show no trend but there is an element of seasonality. Moreover, the standard deviation as we can see from Figure 8 is not fluctuating; this shows that the data is not deviating from the mean. The rolling mean has also fit in very much in the data; hence, there is no trend component in the minimum temperature data. Table 1 shows the data description of the data which is obtained using df.Temp.describe() function.

TABLE I. Data Description

Data Item	Value
count	3650.000
mean	11.177
std	4.071
min	0.000
25%	8.300
50%	11.000
75%	14.000
max	26.300

However, the stationary test of the data was done using the argumentative Dickey Fuller's stationary test and the results are presented in Table 2.

TABLE II. RESULTS OF DICKEY FULLER'S TEST

Test	Value
Test Statistic	-4.444805
p-value	0.000247
#Lags Used	20.000000
Number of Observations Used	3629.000000
Critical Value (1%)	-3.432153
Critical Value (5%)	-2.862337
Critical Value (10%)	-2.567194

V. CONCLUSION

In conclusion, the work to some extent achieved the aims and objectives. The purpose was to guide the user on how to read a data file in our case the minimum daily temperatures over 10 years (1981-1990) in the city Melbourne, Australia. Then preprocess the data by plot visualization and making the necessary adjustment. The data was decomposed into the base level, trend, seasonal and residue components. Finally, the data stationary status was checked using Dicky Fuller's stationary test and the test shows that the data is stationary. This shows that the data is now ready for data mining modeling process for further extracting meaningful information from the data.

REFERENCES

[1] W. Wu, W. Zhang, Y. Yang, and Q. Wang, "Time series analysis for bug number prediction," 2nd Int. Conf. Softw. Eng. Data Mining, SEDM 2010, pp. 589–596, 2010.

[2] "Time Series Database (TSDB) Explained | InfluxDB | InfluxData." [Online]. Available: https://www.influxdata.com/time-series-database/. [Accessed: 14-Apr-2020].I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] "What the heck is time-series data (and why do I need a time-series database)?" [Online]. Available: https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/. [Accessed: 14-Apr-2020].

[4] N. Zhang and W. F. Lu, "An efficient data preprocessing method for mining customer survey data," IEEE Int. Conf. Ind. Informatics, vol. 1, pp. 573578, 2007.

[5] R. Adhikari, and R. K., Agrawal, "An Introductory Study on Time Series Modeling and Forecasting", arXiv Prepr. arXiv1302.6613, vol. 1302.6613, pp. 1–68, 2013.

[6] H. Zeng, J. Guo, J. Wang, and T. Gu, "Data preprocessing system of depression recognition," Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019, no. Itnec, pp. 1478–1482, 2019.

[7] S. Bin and G. Sun, "The preprocessing in census data with concept hierarchy," ICCET 2010 - 2010 Int. Conf. Comput. Eng. Technol. Proc., vol. 1, pp. 535–538, 2010.

[8] M. V. Muntean and D. Onita, "Agent for Preprocessing and Forecasting Time-Series Data," Proc. 10th Int. Conf. Electron. Comput. Artif. Intell. ECAI 2018, 2019.

[9] L. Zheng, W. Hu, and Y. Min, "Raw wind data preprocessing: A data-mining approach," IEEE Trans. Sustain. Energy, vol. 6, no. 1, pp. 11–19, 2015.

[10] B. Bhoi, P. Vyawahare, P. Avhad, and N. Patil, "Data duplication avoidance in larger database," Proc. 2017 Int. Conf. Innov. Information, Embed. Commun. Syst. ICIIECS 2017, vol. 2018-Janua, pp. 1–4, 2018.

[11] A. Shamim, M. U. Shaikh, and S. U. R. Malik, "Intelligent data mining in autonomous heterogeneous distributed bio databases," 2010 2nd Int. Conf. Comput. Eng. Appl. ICCEA 2010, vol. 1, pp. 6–10, 2010.

[12] X. Bolei and W. Cheng, "Search of attributes related to a topic in database," Proc. IEEE Int. Conf. Intell. Process. Syst. ICIPS, vol. 2, pp. 1185–1188, 1998.

[13] T. B. Ho and T. D. Nguyen, "Visualization support for user-centered model selection in knowledge discovery in databases," Proc. Int. Conf. Tools with Artif. Intell., no. 1, pp. 228–235, 2001.

[14] M. Bian, J. Xia, and J. Xu, "Database preprocessing with AHP," Proc. - 2010 7th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2010, vol. 6, no. Fskd, pp. 2805–2809, 2010.

[15] J. J. de Gruijter, M. F. P. Bierkens, D. J. Brus, and M. Knotters, Time-Series Models. 2006.

[16] Z. Vojinovic, V. Kecman, and R. Seidel, "A data mining approach to financial time series modelling and forecasting," Int. J. Intell. Syst. Accounting, Financ. Manag., vol. 10, no. 4, pp. 225–239, 2001.