sciendo

Pattern Discovery in Time Series Data Using Python Script and MS Excel

Viktor ANDREEV*
University of Economics, Varna, Bulgaria
**Corresponding author, viktor.andreev@ue-varna.com*

Julian VASILEV
University of Economics, Varna, Bulgaria
vasilev@ue-varna.bg

Abstract. *Pattern discovery is a novel approach to time series data. Even though many attempts are made, some innovative ideas for pattern discovery may be proposed. The purpose of this paper is to present an approach for extracting Yahoo finance data, converting it into MS Excel data, and discovering patterns in MS Excel. The main approach contains two steps: first, using a Python script for Yahoo finance data extraction and saving it as an MS Excel spreadsheet (time series data for a specific sector for a certain period), and second, using simple Excel formulas for pattern discovery. A standard count of pattern occurrences is calculated according to the length of the pattern. Patterns with different sizes are checked. Each pattern is checked – its appearances count in the time series compared to the standard count of occurrences. If the pattern occurrences count is greater than the standard occurrences count, it may be concluded that the pattern discovery method is successful. The paper presents an original (novel) approach to pattern discovery. The approach is limited to a single sector, but it can be applied to pattern discovery in any business sector. The paper is useful for practitioners who seek advanced forecasting techniques based on pattern discovery. The performance testing of different patterns is done by visualizing their occurrence count in the time series.*

Keywords: Pattern discovery, time series data, Python script.

Introduction

In today's world, where data plays a major role in making all kinds of decisions, in the financial industry, its acquisition is associated with the allocation of a large amount of cash. This fact can be a valid reason for many individual investors not to resort to their own analysis in their quest to optimize their financial portfolio as much as possible.

To facilitate the data acquisition process, developers of various computer languages, including Python, have focused on creating some libraries to help users extract data from websites. Libraries like Requests, BeautifulSoup, Selenium, and Pyppeteer can be very useful in extracting different types of information that can be used in many different industries. (“How to Use Python to Scrape Data From a Website & Save It to Excel”, 2025).

For our paper, data extraction from one of the leading online finance platforms "Yahoo Finance". For this purpose, the yfinance Python library is used. It is specialized in extracting data from this website. (“📥 Downloading Sectors ETFs with YFinance”, 2025).

After extracting the data and structuring it in a relational format, we proceed to analyze it and discover a template (-s)/pattern (-s) that will serve us in choosing the most suitable financial instrument for investment, with which we can help build our optimal portfolio.

DOI: 10.2478/picbe-2025-0075

Time series data and time series analysis are conventional techniques for data analysis. But in practice some new robust or ambiguous methods are needed for better forecasting with lower levels of errors and better result precision. For sure, pattern discovery is an innovative approach but it lacks strong literature and business-oriented implications in time series financial data. That is why the gap in practice is identified as available theoretical knowledge on data extraction, ETL procedures, pattern discovery, financial analysis but taking them in consideration individually. The lack of combined practical knowledge for real pattern discovery in financial time series data outlines the need for the current research in this paper. Since time series data may be for different areas of economy, the research is limited to the time series analysis of financial data. Even though, its practical implications may be tested with other time series data.

Literature review

Time series data, representing sequences of data points collected over time intervals, are pivotal in various fields such as finance, healthcare, and manufacturing. The discovery of patterns within these time series is essential for tasks like forecasting, anomaly detection, and system optimization. Recent advancements have introduced innovative methodologies to enhance pattern discovery in time series data.

Tavabi and Lerman (2021) proposed an unsupervised method leveraging Byte Pair Encoding (BPE) for pattern discovery in time series data. This approach identifies variable-length, interpretable patterns by compressing the time series, effectively capturing both short-term and long-term dependencies. The method is applicable to both univariate and multivariate time series and has demonstrated superior performance over state-of-the-art approaches in real-world datasets.

Cheng et al. (2022) introduced a model that constructs correlations via diverse pattern functions to improve multivariate time series forecasting. Unlike traditional methods that use shared and fixed pattern functions, this model automatically captures diverse and complex time series patterns. A learnable correlation matrix enables the model to capture distinct correlations among multiple time series, leading to state-of-the-art prediction accuracy.

Farahani et al. (2023) conducted a comprehensive literature review on time-series pattern recognition within smart manufacturing systems. They developed an ontology to classify and define concepts, structures, properties, and relationships pertinent to time-series pattern recognition in manufacturing. This work provides practical guidelines and recommendations for applying time-series analytics in industrial settings.

An approach utilizing autoencoders for unsupervised pattern discovery in time series was explored by Malhotra et al. (2016). This method captures the spatiotemporal structure in temporal data by learning meaningful patterns through convolutional filters, leading to interpretable and sparse representations. The model is trained end-to-end, facilitating flexible extensions and systematic training frameworks.

Khoshrou and Pauwels (2018) addressed pattern identification and outlier detection using Singular Value Decomposition (SVD). By recasting the time series as a matrix, SVD highlights underlying patterns and periodicities without requiring user-defined parameters. This data-driven approach allows for the analysis of time series in a bottom-up fashion, accommodating various signal and noise characteristics.

Despite significant advancements in time series pattern discovery, a notable gap remains in the literature regarding the integration of various methodologies for financial time series analysis. While studies have proposed diverse approaches—such as unsupervised compression techniques (Tavabi & Lerman, 2021), correlation-based models (Cheng et al., 2022), ontology frameworks for

manufacturing (Farahani et al., 2023), autoencoder-based learning (Malhotra et al., 2016), and matrix decomposition methods (Khoshrou & Pauwels, 2018)—these techniques are often developed for specific domains or tasks. The lack of a unified framework that combines these approaches for practical, business-oriented financial pattern discovery highlights the need for further research in this area.

These studies collectively advance the field of time series pattern discovery by introducing novel methodologies that enhance interpretability, accuracy, and applicability across various domains. Time series data and analysis are well-established methods in data evaluation. However, in practical applications, there is a growing need for more robust and adaptive approaches to enhance forecasting accuracy while minimizing errors. Although pattern discovery is a novel concept, it remains underexplored in the literature, particularly in terms of its relevance to financial time series data and business applications. This highlights a gap in practice, as existing theoretical knowledge covers aspects such as data extraction, ETL processes, pattern recognition, and financial analysis, but often treats them separately. The absence of integrated practical knowledge for effective pattern discovery in financial time series data underscores the necessity of this research. While time series analysis applies to various economic sectors, this study focuses specifically on financial data, though its practical applications could extend to other domains.

Methodology

Python programming language is used to extract the data that will be used to build our relational database. Thanks to the growing popularity of this language, developers have found a way to use it in order to extract information from web pages. This feature of Python is popularly known as web scraping.

After retrieving the dataset from Yahoo finance with a Python script, MS Excel is used for pattern discovery.

To accomplish our goal of extracting financial data, the library which is used is yfinance. It makes the extraction of financial data from the platform Yahoo Finance possible.

The data discussed in the report is for one of the most popular exchange-traded funds (ETFs) SPDR. ("All SPDR ETFs from Select Sector SPDR | Top Sector ETFs", n.d.)

For this purpose, we follow the whole process in several steps:

1) Install and load the required libraries

```
!pip install - - quiet yfinance
!pip install - - quiet tqdm
```

Figure 1. Installation of required libraries

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

With the first two commands two libraries that are not available in our programming environment are loaded. Besides yfinance, which is focused on working with data from the Yahoo Finance platform, the other library that is important for the task at hand is tqdm. It aims to optimize the iterations to be executed in the data mining process.

```
import yfinance as yf
import pandas as pd
from tqdm import tqdm
import datetime
```

Figure 2: Import of required libraries

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

Once the two libraries mentioned (shown in Figure 1) are imported, the next step is to load all the libraries needed. Apart from importing yfinance and tqdm, which have been downloaded, the other libraries we need to accomplish the set goal are pandas (a library that has established itself in recent years as the most important library related to data analysis and processing), datetime (a library that focuses on working with a time frame, which will arise as we track the progress of financial instruments over different time windows), and the os library (which provides interaction with the operating system).

2) Determining indices related to the financial instruments we need;

```
tickers_names = {
“XLC”: “Communication Services”,
“XLY”: “Consumer Discretionary”,
“XLP”: “Energy”,
“XLF”: “Financials”,
“XLV”: “Health Care”,
“XLI”: “Industrials”,
“XLB”: “Materials”,
“XLRE”: “Real Estate”,
“XLK”: “Technology”,
“XLU”: “Utilities”,
}
```

Figure 3: Defining dictionary with values

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

Once the specific ETF from which to extract the data are defined, we need to create one of the popular data analysis structures in the Python programming language: a dictionary. In this dictionary an index is set, which is a unique code for each sector ETF. Another parameter which is added to the dictionary is the ETF type. The purpose is to identify the sector whose performance is followed by the ETF.

3) Definition of the data extraction function

```
def fetch_data(ticker_symbol, commodity_name): 1 usage
ticker = yf.Ticker(ticker_symbol)
today = datetime.date.today().strftime(‘%Y-%m-%d’)
date = ticker.history(start=”1900-01-01”, end=today)
data.reset_index(inplace=True)
```

Figure 4. Defining function

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

For fetching the data, it is important to define a function that is named fetch_data. In it, there is a mandatory setting of two variables: ticker_symbol and commodity_name. Next, a ticker object is set, where the first variable in our fetch_data function is provided via the yfinance library function named Ticker. The next object that should be defined is today's date "today" that should be in a standard format. The timeframe, which is analyzed is set next with the function "data". Finally, the index is removed, which turns the "date" column into a standard column that can be processed, since its original format as an index limits the ability to work with it.

4) Check the "Date" column

```
if pd.api.types.is_datetime64_any_dtype(data['Date']):
data['Date'] = data['Date'].dt.strftime('%Y-%m-%d')
```

Figure 5. Checking Column "Date"

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

With the above code, we check if the column in the "Date" table is of type date and set the format to match the specified format of the object named "today".

5) Column adjustments

```
data.drop(columns=['Dividents', 'Stock Splits', 'Adj Close', 'Capital Gains'], inplace = True,
errors='ignore')
date['ticker'] = ticker_symbol
data['sector'] = commodity_name
data_columns = [col.lower() for col in data.columns]
data = data[['ticker', 'sector'] + [col for col in data.columns if col not in ['ticker' , 'sector']]]
return data
```

Figure 6. Column adjustments in the dataset

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

As a next step, it is important to select the columns that are of interest to us and remove those that we do not need. It is important to create two additional columns in the table to store information about the two variables (a "ticker" column to store the information from "ticker_symbol" and a "sector" column to store the information from "commodity_name"). In order to make the table transparent, the following changes are made: change of the capital letters in the column headings to lower case and rearranging the variables, moving the two added with the names 'ticker' and 'sector' to the beginning of the table. In the last row of this segment, the information processed so far is recalled.

6) Create a list and arrange the extracted data in it

```
all_data = []
for symbol, name in tqdm(tickers_names.items(), desc="Fetching data"):
all_data.append(fetch_data(symbol, name))
master_data = pd.concat(all_data, ignore_index = True)
master_data.info()
```

Figure 7: Creating a list and arranging the extracted data in it

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

To save the results obtained after extracting the data, a list named all_data is created. By iteration, a data from all the elements is extracted in the dictionary we generated in Section 2 and store it in the created list all_data. To track the progress of the operation, a progress indicator("progress bar" from the tqdm library) is used. Using the concat function, all the extracted data from the individual ETF instruments is structured into a single table named "master_data". Once we have completed the operation, numerous commands could be used to gain insight into our dataset(in our case a method called "info()" is used for overview of the extracted data).

```
Fetching data: 100% 11/11 [00:04<00:00, 2.67it/s]
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 62999 entries, 0 to 62998
Data columns (total 8 columns):
#  Column  Non- Null Count  Dtype
---  --------  --------------------  -------
0  ticker    62999 non-null  object
1  sector   62999 non-null  object
2  data      62999 non-null  object
3  open     62999 non-null  float64
4  high     62999 non-null   float64
5  low      62999 non-null   float64
6  close    62999 non-null   float64
7  volume 62999 non-null   int64
dtypes: float64(4), int64(1), object(3)
        memory usage: 3.8+ MB
```

Figure 8. Code execution and getting information about the data

Source: https://www.kaggle.com/code/guillemservera/downloading-sectors-etfs-with-yfinance

The dataset for the current analysis (after filtering for the sector “Health care”) has 6557 rows.

After completing the process and executing the ".info()" method, as a result, a data set that tracks the price change of the selected financial instruments for the period from its release into circulation until the day on which we output the data is obtained (in our case it is 15.01.2025). The first two variables give us information that was set when extracting the data such as the ETF index and the industry it tracks. The next column gives information about the day the data was reported. The next four columns inform about the price at which the day started; the minimum price and the maximum price that the ETF option reached on that particular day. The last column, which is related to the price, is the "close" column, where the price at which the specified ETF closed that day is reflected. All these variables that are related to the price of the ETF are of type "float" and represent the price up to 2 decimal places. The last column, "volume", which appears in our table, tracks the number of units traded by the specified ETF on that day.

Once we have successfully extracted the required information, the next step is to transfer the data into the software that will go on to analyse "MS Excel".

This action is possible again using the function from the "pandas" library:

```
master_data.to_excel(“stock_data.xlsx”)
```

Figure 9. Export of the dataset to MS Excel

Source: Own elaboration.

In the above code segment, the dataset is transferred to an Excel file and it is named "stock_data.xlsx".

To gain insight from the data, it is appropriate to filter our table by the last date.

To begin our pattern discovery analysis, as a first step in an additional column which defines if the ETF increased or decreased its value during that day

```
=IF(H2>E2;”Up”;”Down”)
```

Figure 10. Adding Column “Change”

Source: Own elaboration.

In cases where the price at which the instrument started the day ("open") is lower than the price at which the same instrument ended the day ("close"), in our newly created "Change" column we reflect the progress for that day as up and name it "Up". In cases where we have the opposite trend and the instrument's opening price is higher than the price at which it closed the day, this row is labeled with the word "Down".

Once the daily movement of the ETF instrument is tracked, a period of days to define a frame should be chosen (with the dimensionality of the template - for example, 5 days) with which to compare the performance of this instrument over different time periods.

For the initial window (template/pattern) a frame of 5 dates is chosen. To have a matching set frame, the movement of all five members of that frame must match the movement of the five members of the new frame we choose to compare.

=IF(AND(J2=J7;J3=J8;J4=J9;J5=J10;J6=J11);"Window match";"---")

Figure 11. Comparison with first template/pattern

Source: Own elaboration.

If there is a complete match, as a result "Window match" is printed. Otherwise, an empty value is returned.

Having checked for matches on the first window, the second comparison is processed which starts from the second date in our dataset.

=IF(AND(J3=J7;J4=J8;J5=J9;J6=J10;J7=J11);"Window match";"---")

Figure 12. Comparison with second template/pattern

Source: Own elaboration.

The next check which is completed starts on the third date of the period.

=IF(AND(J4=J7;J5=J8;J6=J9;J7=J10;J8=J11);"Window match";"---")

Figure 13. Comparison with third template/pattern

Source: Own elaboration.

The verification continues with the period starting from the fourth day.

=IF(AND(J5=J7;J6=J8;J7=J9;J8=J10;J9=J11);"Window match";"---")

Figure 14. Comparison with fourth template/pattern

Source: Own elaboration.

The last check starts with the fifth date of the dataset

=IF(AND(J6=J7;J7=J8;J8=J9;J9=J10;J10=J11);"Window match";"---")

Figure 15. Comparison with fifth template/pattern

Source: Own elaboration.

Results and discussions

To gain insight from the data, it is appropriate to filter our table by the last date:

Table 1. ETF options overview (a fragment from the dataset)

Ticker	Sector	Date	Open	High	Low	Close	Volume
XLC	Communication services	2025-01-15	96,92	97,48	96,71	97,1	4115700
XLY	Consumer discretionary	2025-01-15	226,75	227,47	225,18	227,28	3098700
XLP	Consumer staples	2025-01-15	76,91	77,15	76,11	76,23	11426800
XLE	Energy	2025-01-15	91,78	92,81	91,4	92,57	17327100

Ticker	Sector	Date	Open	High	Low	Close	Volume
XLF	Financials	2025-01-15	49,28	49,61	49,01	49,48	61161100
XLV	Health Care	2025-01-15	140,39	141,24	139,77	140,57	11386700
XLI	Industrials	2025-01-15	136,43	136,64	135,04	135,29	9183600
XLB	Materials	2025-01-15	87,47	87,7	86,47	87,04	6281000
XLRE	Real Estate	2025-01-15	41,06	41,15	40,14	40,19	6511000
XLK	Technology	2025-01-15	230,87	232,97	230,11	232,29	5232200
XLU	Utilities	2025-01-15	77	77,55	76,71	76,79	10680200

Source: Own elaboration and Kaggle data.

To do this, the dataset is filtered by a selected industry to look at ETF performance by finding a template/pattern to track a trend. Healthcare is an industry that has always been of interest to investors. The property of financial instruments from this area to remain stable even in moments of crisis, such as the COVID crisis in 2021, is a pulling force for those who decide to set aside a different amount of their savings to purchase this financial instrument.

Table 2.ETF Selection for observation (filtered dataset by sector "Health care")

Ticker	Sector	Date	Open	High	Low	Close	Volume
XLV	Health Care	1998-12-22	17.00549	17.17683	17.00549	17.15541	5700
XLV	Health Care	1998-12-23	17.26251	17.54094	17.26251	17.54094	18100
XLV	Health Care	1998-12-24	17.56235	17.64802	17.48739	17.64802	4900
XLV	Health Care	1998-12-28	17.59448	17.64802	17.39101	17.39101	15500
XLV	Health Care	1998-12-29	17.39101	17.77652	17.39101	17.77652	5300
XLV	Health Care	1998-12-30	17.81936	17.84078	17.62661	17.62661	17300

Source: Own elaboration and Kaggle data.

The data which is tracked is from 1998, when this ETF began trading. A characteristic of the financial market is that it is not open on weekends, so data from weekends are missing.

Table 3. Adding column "Change" in MS Excel

Ticker	Sector	Date	Open	High	Low	Close	Volume	Change
XLV	Health Care	1998-12-22	17.00549	17.17683	17.00549	17.15541	5700	Up
XLV	Health Care	1998-12-23	17.26251	17.54094	17.26251	17.54094	18100	Up
XLV	Health Care	1998-12-24	17.56235	17.64802	17.48739	17.64802	4900	Up
XLV	Health Care	1998-12-28	17.59448	17.64802	17.39101	17.39101	15500	Down
XLV	Health Care	1998-12-29	17.39101	17.77652	17.39101	17.77652	5300	Up
XLV	Health Care	1998-12-30	17.81936	17.84078	17.62661	17.62661	17300	Down

Source: Own elaboration.

The "Change" is calculated by the difference between "Close" and "Open". But this change is analyzed in this paper as a direction (increase or decrease). The change amount is not analzyed. For sure, the change amount gives valuable information, but future research may be focused on this issue.

Table 4. Adding column „Window_match" in MS Excel

Ticker	Sector	Date	Open	Close	Volume	Change	Window_match
XLV	Health Care	1998-12-22	17.00549	17.15541	5700	Up	
XLV	Health Care	1998-12-23	17.26251	17.54094	18100	Up	
XLV	Health Care	1998-12-24	17.56235	17.64802	4900	Up	
XLV	Health Care	1998-12-28	17.59448	17.39101	15500	Down	
XLV	Health Care	1998-12-29	17.39101	17.77652	5300	Up	

Ticker	Sector	Date	Open	Close	Volume	Change	Window_match
XLV	Health Care	1998-12-30	17.81936	17.62661	17300	Down	
XLV	Health Care	1998-12-31	17.69086	17.81936	6600	Up	
XLV	Health Care	1999-01-04	17.9907	17.77652	249200	Down	
XLV	Health Care	1999-01-05	17.81937	18.05496	4400	Up	
XLV	Health Care	1999-01-06	18.18345	18.35479	38500	Up	---
XLV	Health Care	1999-01-07	18.18347	18.26914	15900	Up	---
XLV	Health Care	1999-01-08	18.31197	18.49402	24400	Up	---
XLV	Health Care	1999-01-11	18.4833	18.53684	7700	Up	---
XLV	Health Care	1999-01-12	18.5904	18.46189	270700	Down	---
XLV	Health Care	1999-01-13	17.66944	18.29055	7100	Up	Window match

Source: Own elaboration.

The initial pattern "up, up, up, down, up" is found in the time series for the dates in the last five rows of Table 4. When the pattern is moved in the time series at some other dates a "window match" is found.

Table 5. Result of comparison with 5 patterns

Pattern	Count of window matches (occurrence count)	Percent of total number of possible matches
First	200	3.05%
Second	220	3,36%
Third	212	3,23%
Fourth	200	3.05%
Fifth	212	3,23%

Source: Own elaboration.

The period which is chosen for our analysis is 5 days. That is a fundamental period in securities trading since, as already mentioned, there are five days in the week when the individuals can trade. The performance testing of different patterns is done by visualizing their occurrence count in the time series. The higher occurrence count of a specific pattern corresponds with a higher possibility of its real occurrence in the time series data.

As a target to set to analyze the results, the mean of the matches that would occur in a completely random distribution of measurements should be calculated. Since the chance of matching each element of the pattern is 0.5, to get the complete random pattern match we can raise this number to the fifth power. This would give the following result:

0,55 = 0,03125 = 3,125%

When we compare the result of the template/pattern we obtained with the average of the random frames, we get the following result:

1. 3.05% < 3.125% (pattern 1)
2. 3.36% > 3.125% (pattern 2)
3. 3.23% > 3.125% (pattern 3)
4. 3.05% < 3.125% (pattern 4)
5. 3.23% > 3.125% (pattern 5)

These results show us that in three of the checks we perform, the result achieved is better than the average we would have with a completely random selection. The pattern that starts at the second date in our dataset has the best performance as it reaches 3.36% matching.

Therefore, pattern 2 (up, up, down, up, down) is the most common. If we find an "up, up, down, up" sequence in the trend, we can expect "down" in the next day.

To increase the performance of the pattern there is a possibility to decrease the number of the observations in the sample (from 5 to 4). This will result in a smaller reliability of the results.

In order to increase the reliability of the model the other option is to increase the number of observations in the sample (from 5 to 6). That will lead to a lower performance of the model.

Conclusion

This paper presents a showcase that allows individual investors to obtain the necessary data using a Python script and pattern discovery in MS Excel to find a pattern to predict increase or decrease of a financial instrument.

The objective was to identify a pattern that can improve the accuracy of forecasting financial instrument movements.

After performing several tests, it may be concluded that the functionality of this approach enables the discovery of patterns that enhance the forecasting of financial instrument movements.

With over 50% of the test results outperforming random movement distribution of ETF instruments, it may be concluded that the pattern identification is successful. This research will be useful for achieving our broader goal – creating a profitable investment portfolio.

In our paper there is a huge amount of data, which follows a long period of time. However, there is an obvious limitation of using historical financial data. Future work may be focused on further enrichment of the paper to update it with an option of using live data to track the performance of the ETF. A convenient way of doing this is through implementation of Yahoo API, which is compatible with the Python programming language and will provide a live update of the financial data. This will help to increase further the reliability of the model and to add a certainty to our decision-making process.

Even though the showcase in this paper is practically oriented, it enriches the theoretical knowledge for academia in the field of time series analysis using pattern discovery.

Further research may show how to use the MS Excel Solver tool to find the most occurred pattern.

The “Change” is calculated by the difference between “Close” and “Open”. But this change is analyzed in this paper as a direction (increase or decrease). The change amount is not analzyed. For sure, the change amount gives valuable information. Another future research may be focused on this issue.

References

Aleksandrova, Y. (2019). Predicting students performance in Moodle platforms using machine learning algorithms. In *Conferences of the department Informatics* (No. 1, pp. 177-187). Publishing house Science and Economics Varna.

Ana-Maria Ramona, S., Marian Pompiliu, C., & Stoyanova, M. (2020). Data mining algorithms for knowledge extraction. In Challenges and Opportunities to Develop Organizations Through Creativity, Technology and Ethics: The 2019 Griffiths School of Management *Annual Conference on Business, Entrepreneurship and Ethics (GSMAC)* (pp. 349-357). Springer International Publishing.

Armyanova, M., & Aleksandrova, Y. (2023, December). Design patterns in machine learning. In *AIP Conference Proceedings* (Vol. 2938, No. 1). AIP Publishing.

Cheng, H., Zhang, H., Lin, Z., & Song, G. (2022). A pattern discovery approach to multivariate time series forecasting. *arXiv preprint arXiv:2212.10306*. Retrieved from https://arxiv.org/abs/2212.10306

Farahani, R. G., Kashef, R., & Mousavi, A. (2023). Time-series pattern recognition in smart manufacturing systems: A literature review and ontology. *arXiv preprint arXiv:2301.12495*. Retrieved from https://arxiv.org/abs/2301.12495

Kaggle dataset. Available at: https://www.kaggle.com/code/guillemservera/downloading-world-indices-data-with-yfinance

Khoshrou, H., & Pauwels, E. J. (2018). Data-driven pattern identification and outlier detection in time series. *arXiv preprint arXiv:1807.03386*. Retrieved from https://arxiv.org/abs/1807.03386

Kuyumdzhiev, I., & Petrov, P. (2024). Virtualization and Online Engineering of the Administrative Services in Universities. *International Journal of Online & Biomedical Engineering*, 20(1).

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2016). Unsupervised learning of time-series representations using stacked LSTM networks. In *Advances in neural information processing systems (NeurIPS)*. Springer. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-49055-7_38

Nacheva, R. (2024). An Emotions Mining Approach To Support Artificial Intelligence Systems. In *Conferences of the department Informatics* (No. 1, pp. 92-104). Publishing house Science and Economics Varna. Retrieved from https://informatics.ue-varna.bg/ICTBE2024/ICTBE2024_92-104.pdf

Parusheva, S., & Pencheva, D. (2021). Modeling a Business Intelligent System for Managing Orders to Supplier in the Retail Chain with Unified Model Language. In *Digital Transformation Technology: Proceedings of ITAF 2020* (pp. 375-393). Singapore: Springer Singapore.

Penchev, B. (2024). A Study on The Usage Of M-Learning Applications Within Bulgarian Schools. *Business & Management Compass*, 68(1), 45-53.

Simeonidis, D., Petrov, P., Penchev, G., Petrova, S., Dimitrov, G., & Petrivskyi, V. (2024, October). Performance and Accuracy Assessment of Detecting Network Intrusions with eSOM-Based Techniques. In *2024 International Conference Automatics and Informatics (ICAI)* (pp. 586-591). IEEE.

Stoyanova, M. (2020). Good practices and recommendations for success in construction digitalization. *TEM Journal*, 9(1), 42-47.

Sulova, S. (2016). An approach for automatic analysis of online store product and services reviews. Izvestiya. *Journal of Varna University of Economics*, 60(4), 455-467.

Sulova, S., & Bankov, B. (2019). Approach for social media content-based analysis for vacation resorts. *Journal of Communications Software and Systems*, 15(3), 262-271.

Sulova, S., & Marinova, O. (2024). Metadata Management Framework for Business Intelligence Driven Data Lakes. *Business Management/Biznes Upravlenie*, (2).

Sulova, S., Aleksandrova, Y., Stoyanova, M., & Radev, M. (2022, June). A Predictive Analytics Framework Using Machine Learning for the Logistics Industry. In *Proceedings of the 23rd International Conference on Computer Systems and Technologies* (pp. 39-44).

Tavabi, N., & Lerman, K. (2021). Pattern discovery in time series with byte pair encoding. *arXiv preprint arXiv:2106.00614*. Retrieved from https://arxiv.org/abs/2106.00614

Todoranova, L., Nacheva, R., Sulov, V., & Penchev, B. (2020). A model for mobile learning integration in higher education based on students' expectations. *International Journal of Interactive Mobile Technologies (iJIM)*, Wien: International Association of Online Engineering (IAOE), 14, 2020, 11, 171-182.