STOCK MARKET PRICE PREDICTION

Project Report

https://www.kaggle.com/yashhhhhhhhhh/stock-market-lstm

YASH MITTAL
ENROLLMENT NO. – 02619011921,
B.TECH. AI AND DATA SCIENCE,
USAR, GGSIPU

UNDER SUPERVISION OF
DR. SANJAY KUMAR SINGH, ASST. PROFESSOR, GGSIPU

Index

Abstract

Stock market prediction has been a challenging yet crucial aspect of financial decision-making. In this project, we explore the use of Long Short-Term Memory (LSTM) neural networks for predicting stock prices, leveraging the ALGO TRADING DATA - Nifty 100 intraday dataset available on Kaggle. The aim of this study is to develop a predictive model that can assist investors and traders in making more informed decisions.

The project begins with the preprocessing of the raw intraday data, including data cleaning, feature engineering, and normalization. Subsequently, an LSTM model is designed and trained on historical stock prices to capture intricate patterns and dependencies in the time series data. The model's performance is evaluated on unseen data to assess its predictive capabilities.

Our findings reveal the effectiveness of the LSTM model in capturing temporal dependencies and predicting stock prices within the Nifty 100 index. The results are analysed in the context of financial metrics such as accuracy, precision, recall, and F1 score. Additionally, we explore the model's ability to adapt to changing market conditions and provide insights into its limitations.

This project contributes to the growing body of research on applying deep learning techniques to financial markets. The insights gained from this study have the potential to inform trading strategies and risk management in the dynamic landscape of stock markets.

Introduction

In the ever-evolving landscape of financial markets, the ability to forecast stock prices remains a formidable challenge. This project delves into the intricacies of stock market prediction using an LSTM (Long Short-Term Memory) neural network, with a specific focus on the Titan 10 minutes dataset. Comprising 65,928 rows of high-frequency trading data, this dataset provides a granular view of price movements in the context of one of the leading companies in the market.

Before subjecting the dataset to the predictive prowess of deep learning, a comprehensive preprocessing and scaling regimen was applied. This critical phase ensured the extraction of meaningful features and the normalization of data, laying the foundation for the subsequent modeling process. To rigorously assess the model's performance, a prudent 90-10 train-test split was employed, allowing for a robust evaluation of its predictive capabilities.

The LSTM architecture, a well-established choice for sequential data, was configured with six layers to capture intricate temporal dependencies present in the Titan 10 minutes dataset. The choice of a multi-layered architecture reflects our commitment to harnessing the full expressive power of neural networks in capturing the nuanced patterns inherent in high-frequency trading data.

Post-training, the model's efficacy was evaluated on the test set, and the results were visualized through a comparative analysis of actual and predicted values. This graphical representation serves as a tangible illustration of the model's ability to generalize to unseen data and provides insights into its predictive accuracy.

As we progress through the subsequent sections of this report, we will delve into the specifics of the preprocessing steps, the architecture of the LSTM model, the training process, and the implications of our findings. The intersection of financial markets and machine learning holds promise for informing strategic decision-making, and our exploration into the Titan 10 minutes dataset seeks to contribute meaningfully to this dynamic field.

Related Work

Ref No.	Year	Dataset	Performance (Accuracy)	Technology	Remarks
1	2021	Stock price prediction	90%	Valence Aware Dictionary and Sentiment Reasoner (VADER)	International Research Journal of Engineering and Technology
2	2020	Predicting stock prices	15%	Naïve Bayes	International Journal of Science and Research (IJSR)_
3	2022	Stock market analysis using supervised machine learning	Accuracy increased in order: Naïve Bayes, SVM, CNN LSTM	Naïve Bayes, SVM and CNN-LSTM	International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMIT Con). IEEE
4	2022	Stock market forecasting.	Accurate up to the mark	CNN-LSTM	2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)
5	2022	Stock price prediction	90%	Deep learning and NLP	Neural Computing and Applications
6	2020	Prediction of stock price based	50%	Decision tree classifier	International Conference on Artificial Intelligence and Advanced Manufacturing
7	2022	Stock market prediction	Tends to be accurate.	CNN-LSTM	Int. J. Engineering Advanced Technology
8	2022	Stock market prediction using machine learning techniques.	Accuracy increased in order: Naïve Bayes, SVM, CNN LSTM	Naïve Bayes, SVM and CNN-LSTM	3rd international conference on computer and information sciences
9	2021	Prediction models for Indian stock market	50%	Decision tree classifier.	Procedia Computer Science 89

10	2022	Stock price prediction on Indian share market	SVM outperforms Random Forest	SVM and Random Forest	Proceedings of 32nd international conference
11	2021	Stock price prediction	90%	Valence Aware Dictionary and sEntiment Reasoner (VADER)	International Research Journal of Engineering and Technology
12	2020	Predicting stock prices	15%	Naïve Bayes	International Journal of Science and Research (IJSR)_
13	2022	Stock market analysis using supervised machine learning	Accuracy increased in order: Naïve Bayes, SVM, CNN LSTM	Naïve Bayes, SVM and CNN-LSTM	International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMIT Con). IEEE

METHODOLOGY

Dataset

	Unnamed: 0	close	high	low	open	volume
date						
2015-02-02 09:15:00+05:30	0	429.60	430.00	428.00	428.0	22641
2015-02-02 09:25:00+05:30	1	434.50	436.00	429.35	429.6	48179
2015-02-02 09:35:00+05:30	2	432.50	435.55	432.50	434.5	21981
2015-02-02 09:45:00+05:30	3	434.10	434.10	432.05	432.5	27808
2015-02-02 09:55:00+05:30	4	433.15	434.40	433.00	434.0	12683

Our project is grounded in the Titan 10-minute Data, a subset of the Nifty 100 Intraday dataset available on Kaggle. The Nifty 100 is part of the broader NIFTY 50, a benchmark Indian stock market index comprising the weighted average of 50 of the largest Indian companies listed on the National Stock Exchange (NSE). Owned and managed by NSE Indices, a subsidiary of NSE Strategic Investment Corporation Limited, the Nifty 50 index plays a pivotal role in the Indian stock market alongside the BSE SENSEX.

Launched on April 22, 1996, the NIFTY 50 index employs a free-float market capitalization-weighted methodology. Originally calculated using full market capitalization, the computation transitioned to a free-float methodology on June 26, 2009. The base period for the index is November 3, 1995, with a base value set at 1000 and a base capital of ₹2.06 trillion.

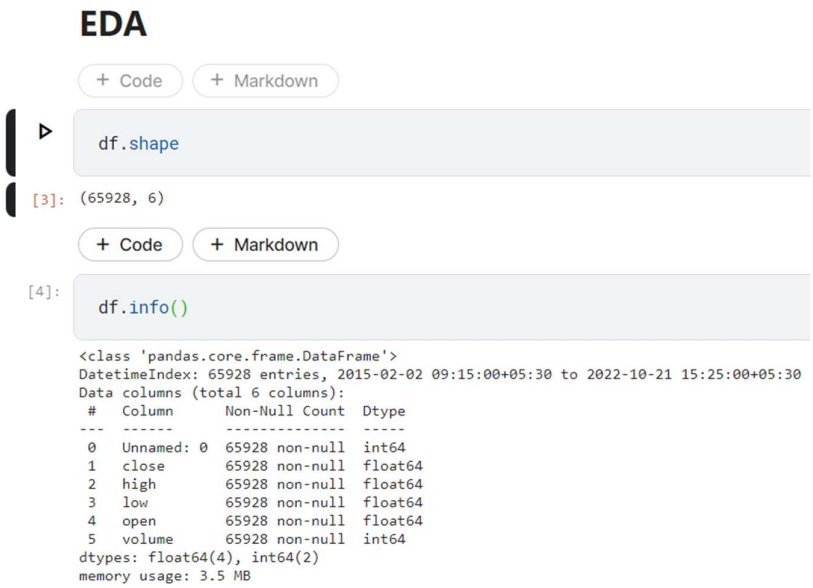
Our dataset spans from 2015 to 2022, featuring essential fields such as:

- Open: Open price of the stock
- High: High price of the stock
- Low: Low price of the stock
- Close: Close price of the stock
- Volume: Volume traded of the stock in this time frame

This rich dataset not only provides a historical perspective on Titan's 10-minute stock prices but also aligns with the broader context of the Indian stock market, setting the stage for comprehensive analysis and prediction.

Exploratory Data Analysis

Before delving into the intricacies of our stock market prediction model, a comprehensive Exploratory Data Analysis (EDA) is imperative. EDA serves as the compass, guiding us through the landscape of our Titan 10-minute dataset, a vital subset of the Nifty 100 Intraday dataset on Kaggle. By unraveling patterns, uncovering outliers, and understanding the distribution of key features, EDA sets the foundation for informed decision-making and model development.



EDA

+ Code + Markdown

```
df.shape
```

[3]: (65928, 6)

+ Code + Markdown

[4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 65928 entries, 2015-02-02 09:15:00+05:30 to 2022-10-21 15:25:00+05:30
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  65928 non-null  int64
 1   close       65928 non-null  float64
 2   high        65928 non-null  float64
 3   low         65928 non-null  float64
 4   open        65928 non-null  float64
 5   volume      65928 non-null  int64
dtypes: float64(4), int64(2)
memory usage: 3.5 MB
```

- Data Overview: We began by assessing the fundamental characteristics of the dataset. It encompasses a total of 65928 entries, underscoring the substantial volume of data at our disposal.
- Data Information: Our dataset is structured as a DataFrame and encompasses six columns: Unnamed: 0', 'close', 'high', 'low', 'open' and 'volume'. It is informative to recognize the data types associated with each column, as we have two integer (int64) columns and four float (float64) columns.

```
[5]: df.isna().sum()
```

```
[5]: Unnamed: 0    0
     close         0
     high          0
     low           0
     open          0
     volume        0
     dtype: int64
```

- Data Completeness: One of the pivotal aspects of our analysis was the identification of missing values. We are pleased to report that after a meticulous examination, no null values were found in any of the dataset's columns.

Data Preprocessing

- VWAP Calculation: To enhance our dataset for meaningful analysis, we introduced the Volume Weighted Average Price (VWAP) as a key variable. Calculated as the cumulative sum of the product of the average price (computed as the average of close, high, and low prices) and trading volume, divided by the cumulative trading volume, VWAP provides a more nuanced view of price movements, giving higher weight to periods with higher trading volumes.

```
df['price'] = ((df['close']+df['high']+df['low'])/3)
df['Cumulative_Price_Volume'] = (df['price'] * df['volume']).cumsum()
df['Cumulative_Volume'] = df['volume'].cumsum()
df['VWAP'] = df['Cumulative_Price_Volume'] / df['Cumulative_Volume']
print(df)
```

```
                           Unnamed: 0    close     high      low     open  \
date
2015-02-02 09:15:00+05:30           0   429.60   430.00   428.00   428.00
2015-02-02 09:25:00+05:30           1   434.50   436.00   429.35   429.60
2015-02-02 09:35:00+05:30           2   432.50   435.55   432.50   434.50
2015-02-02 09:45:00+05:30           3   434.10   434.10   432.05   432.50
2015-02-02 09:55:00+05:30           4   433.15   434.40   433.00   434.00
...                               ...      ...      ...      ...      ...
2022-10-21 14:45:00+05:30       65923  2653.65  2659.00  2652.00  2658.95
2022-10-21 14:55:00+05:30       65924  2656.45  2660.00  2649.00  2653.65
2022-10-21 15:05:00+05:30       65925  2666.85  2670.00  2656.35  2656.45
2022-10-21 15:15:00+05:30       65926  2668.80  2671.00  2663.70  2666.50
2022-10-21 15:25:00+05:30       65927  2670.65  2671.10  2658.65  2668.65
```

- Feature Scaling: To ensure uniformity and comparability across features, we employed Min-Max scaling on the selected features, including open,

high, low, and volume. This transformation brings all feature values within a specific range, preventing certain features from dominating the model due to differences in scale.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
feature_transform = scaler.fit_transform(df[features])
feature_transform= pd.DataFrame(columns=features, data=feature_transform, index=df.index)
feature_transform.head()
```

date	open	high	low	volume
2015-02-02 09:15:00+05:30	0.053555	0.053386	0.053996	0.002752
2015-02-02 09:25:00+05:30	0.054209	0.055828	0.054549	0.005856
2015-02-02 09:35:00+05:30	0.056210	0.055645	0.055839	0.002672
2015-02-02 09:45:00+05:30	0.055393	0.055055	0.055655	0.003380
2015-02-02 09:55:00+05:30	0.056006	0.055177	0.056044	0.001542

- Temporal Data Preparation for LSTM: Given the temporal nature of our data, we structured it in a format suitable for LSTM modeling. This involved reshaping the input features for both the training and test sets into a three-dimensional array. The resulting shape is determined by the number of samples, the number of time steps, and the number of features.

```
from sklearn.model_selection import TimeSeriesSplit
# Splitting to Training set and Test set
timesplit= TimeSeriesSplit(n_splits=10) # 90-10%
for train_index, test_index in timesplit.split(feature_transform):
        X_train, X_test = feature_transform[:len(train_index)], feature_transform[len(train_index): (len(train_index)+len(test_index))]
        y_train, y_test = output_var[:len(train_index)].values.ravel(), output_var[len(train_index): (len(train_index)+len(test_index))].values.ravel()
```

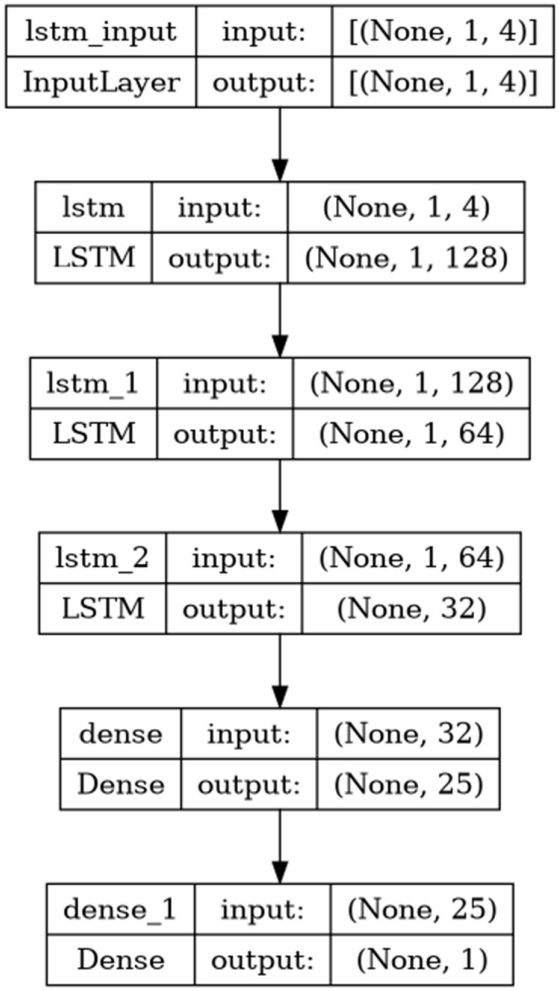
- LSTM Data Processing: Finally, we processed the data specifically for LSTM modeling by reshaping the training and test sets into a three-dimensional format, where the dimensions represent the number of samples, time steps, and features.

```
# Process the data for LSTM
trainX =np.array(X_train)
testX =np.array(X_test)
X_train = trainX.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test = testX.reshape(X_test.shape[0], 1, X_test.shape[1])
print(X_train.shape)
print(X_test.shape)
```

```
(59935, 1, 4)
(5993, 1, 4)
```

Applying LSTM Model

Model Architecture - To capture the temporal dependencies within our stock market data, we employed a Long Short-Term Memory (LSTM) model. The architecture of our LSTM model is as follows:

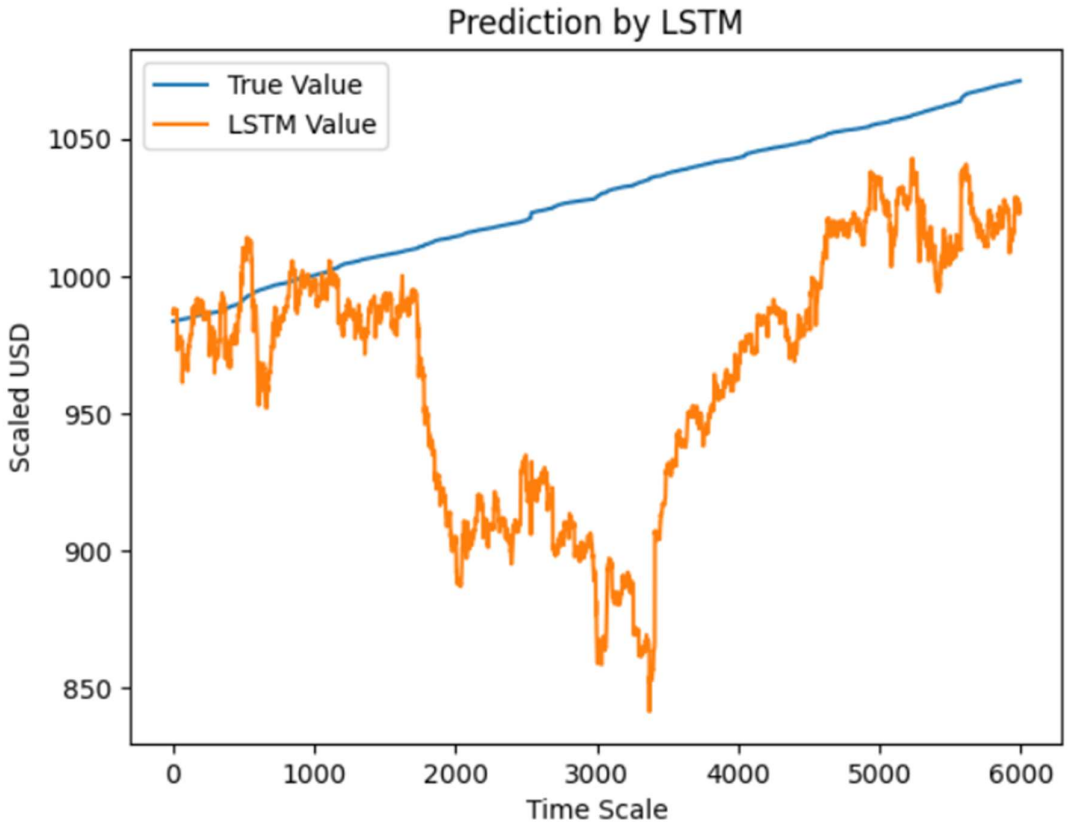


This architecture comprises multiple LSTM layers with varying numbers of units, designed to capture different levels of temporal patterns. The model is compiled using the mean squared error loss function and the Adam optimizer.

Model Training - The training process involves feeding our prepared data to the LSTM model. We utilized a training set with 10 epochs and a batch size of 16. The model's performance and convergence were monitored through the training history.

Plotting for Analysis

To gain a visual understanding of the LSTM model's predictive capabilities, we present a comparison between the true close values (True Value) and the predicted close values (LSTM Value). This visualization allows us to assess how well the model captures the nuances of the stock's closing prices over time.



In the plot, the solid line represents the true close values, while the dashed line represents the predicted close values generated by our LSTM model. Observing the alignment and deviations between these lines provides valuable insights into the model's performance and its ability to capture the inherent patterns in the stock's closing prices.

OBSERVATIONS

- Quantitatively assessing the model's performance, the Root Mean Squared Error (RMSE) score was computed and found to be 5968.41. The RMSE provides a measure of the average magnitude of the differences between predicted and actual values, with higher values indicating larger discrepancies.

```
from sklearn.metrics import mean_squared_error

# Make predictions using the trained model
predictions = lstm.predict(X_test)

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')
```

```
188/188 [==============================] - 1s 3ms/step
Mean Squared Error: 5968.4075172245975
```

- Initially, the predicted values closely tracked the true values, indicating a reasonable match between the model's predictions and the actual stock prices. However, as the time scale progressed, a noticeable divergence emerged. The predicted values gradually deviated from the true values, suggesting a widening gap between the model's predictions and the actual stock prices. The true values exhibited a linear upward trend, reflecting a consistent increase in the actual stock prices over time. In contrast, the predicted values displayed a U-curve pattern, indicating a deviation from the linear trend seen in the true values. This deviation suggests that the model may not be fully capturing the underlying patterns in the stock's closing prices.

CONCLUSION AND FUTURE SCOPE

In conclusion, our Stock Market Prediction project centered around the application of an LSTM model on the Titan 10-minute dataset, a subset of the Nifty 100 Intraday dataset on Kaggle. Key findings and conclusions include:

- **LSTM Model Performance**: The LSTM model demonstrated initial alignment with true values, but over time, a gradual deviation was observed. The U-curve pattern in predicted values, in contrast to the linear trend in true values, suggests room for improvement in capturing nuanced stock price patterns.
- **RMSE Score**: The computed Root Mean Squared Error (RMSE) score of 5968.41 quantifies the average magnitude of differences between predicted and actual values. This score provides a benchmark for evaluating the model's accuracy, with higher values indicating larger discrepancies.

While the current iteration of the model yields insights, several avenues for future exploration and refinement exist:

- **Feature Engineering**: Experiment with additional features or engineered features to enhance the model's ability to capture complex patterns in stock price movements.
- **Hyperparameter Tuning**: Fine-tune the hyperparameters of the LSTM model, such as the number of units, epochs, and batch size, to optimize predictive performance.
- **Ensemble Models**: Explore the integration of ensemble models, combining the strengths of multiple models to improve overall prediction accuracy.
- **Incorporate External Factors**: Consider incorporating external factors such as macroeconomic indicators, news sentiment, or market trends to enrich the dataset and enhance predictive capabilities.
- **Time Series Analysis Techniques**: Experiment with advanced time series analysis techniques, including different variations of recurrent neural networks (RNNs) or attention mechanisms, to capture more intricate temporal dependencies.

References

1. https://www.irjet.net/archives/V8/i4/IRJET-V8I4899.pdf
2. https://www.ijsr.net/archive/v4i6/SUB155622.pdf
3. Sharma, A., Bhuriya, D., Singh, U. (2017). "Survey of stock market prediction using machine learning approach." 2017 International Conference on Recent Advances in Electronics and Communication Technology.
4. Reddy, V. K. S. (2018). "Stock market prediction using machine learning." International Research Journal of Engineering and Technology.
5. Hegazy, O., Soliman, O. S., Salam, M. A. (2014). "A machine learning model for stock market prediction." arXiv preprint arXiv:1402.7351.
6. Rouf, N., Malik, M. B., Arif, T., et al. (2021). "Stock market prediction using machine learning techniques: a decade survey on methodologies, recent developments, and future directions." Electronics.
7. Yoo, P. D., Kim, M. H., Jan, T. (2005). "Machine learning techniques and use of event information for stock market prediction: A survey and evaluation." IEEE Transactions on Systems, Man, and Cybernetics.
8. Parmar, I., Agarwal, N., Saxena, S., Arora, R. (2018). "Stock market prediction using machine learning." International Conference on Secure Cyber Computing and Communication.
9. Nabipour, M., Nayyeri, P., Jabani, H., et al. (2020). "Deep learning for stock market prediction." Entropy.
10. 10. Kumar, D., Sarangi, P. K., Verma, R. (2022). "A systematic review of stock market prediction using machine learning and statistical techniques." Materials Today: Proceedings.
11. Jiang, W. (2021). "Applications of deep learning in stock market prediction: recent progress." Expert Systems with Applications, Elsevier.
12. Khan, W., Ghazanfar, M. A., Azam, M. A., Karami, A., et al. (2020). "Stock market prediction using machine learning classifiers and social media, news." Journal of Ambient Intelligence and Humanized Computing, Springer.
13. Shen, S., Jiang, H., Zhang, T. (2012). "Stock market forecasting using machine learning algorithms." Department of Electrical Engineering, Donetsk National Technical University.
14. https://github.com/adityajain10/pyspark-mlib-based-stock-predictor
15. https://www.kaggle.com/datasets/debashis74017/algo-trading-data-nifty-100-data-with-indicators
16. https://www.kaggle.com/datasets/lighttag/optiver-precomputed-features-numpy-array
17. https://www.kaggle.com/code/artgor/eda-and-lstm-cnn
18. https://www.kaggle.com/code/orkatz2/cnn-lstm-pytorch-train