Politechnika Śląska

Katedra Grafiki, Wizji komputerowej i Systemów Cyfrowych

Academc year			Group	Section
2022/2023	SSI	**BIAI**	**GKiO1**	**1**
Supervisor:	mgr inż. Grzegorz Baron		**Classes: (day, hour)**	
Section members:	**Mateusz Grabarczyk** **Jakub Michałuszek**		**Tuesday**	
emails:	**mategra240@student.polsl.pl** **jakumic451@student.polsl.pl**		**9:45 AM**	

Project card

Subject:

Predicting stock price using machine learning

Main assumptions:

1. Introduction

As we are interested in investing, we wanted to make a model which can predict stock price. Stock market can be hard to predict as some random situations may happen (for instance, COVID). Nevertheless, it's a good practice and interesting topic. Of course, it won't be any financial advice.

2. Analysis of the task

To predict Google's stock price using machine learning and Python, several approaches can be considered. We decided to choose LSTM model. LSTMs are a type of recurrent neural network that can effectively model long-term dependencies in time series data. They can capture complex patterns and relationships, making them suitable for predicting stock prices. However, LSTMs require more computational resources and may be more challenging to train and tune compared to simpler models. Data source is this data set: https://www.kaggle.com/datasets/varpit94/googlestock-data. It's downloaded from official NASDAQ market and it's very reliable. We took certain period that didn't have any major random situation, which for now is from 2012 to 2017.

3. Internal and external specification of the software solution

We chose Jupiter Notebook as the environment for the project. Python provides a rich ecosystem of tools, frameworks, and libraries suitable for this task. Popular libraries such as pandas and NumPy can be used for data manipulation and preprocessing. The libraries that we used are:

- Pandas
- NumPy
- Matplotlib
- Keras
- SciKit-learn

Pandas is a powerful library for data manipulation and analysis, providing data structures and functions for efficient handling of structured data. NumPy is a fundamental library for numerical computing in Python, offering high-performance multidimensional array operations and mathematical functions. Matplotlib is a widely used plotting library in Python, enabling the creation of various types of visualizations for data exploration and presentation. Keras is a high-level deep learning library that provides a user-friendly interface for building and training neural networks, simplifying the development process. SciKit-learn is a comprehensive machine learning library in Python, offering a wide range of algorithms and tools for data preprocessing, model selection, and evaluation.

Main code of the project, so building a modal that had the best results is shown in the Picture 1.

```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from keras.layers import Activation
```

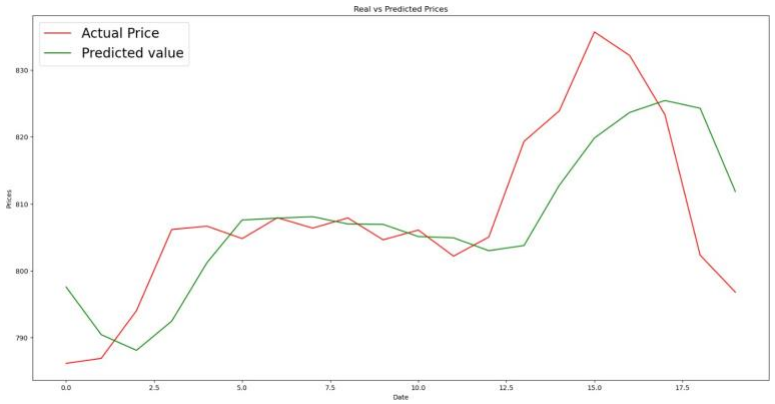
```
model=Sequential()

model.add(LSTM(70,return_sequences=True,activation= 'relu',input_shape = (x_train.shape[1],1)))

model.add(LSTM(70,return_sequences=True,activation= 'relu'))

model.add(LSTM(50))

model.add(Dense(1))

model.compile(loss='mean_squared_error',optimizer='adam')
```

```
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_29 (LSTM)              (None, 100, 70)           20160

 lstm_30 (LSTM)              (None, 100, 70)           39480

 lstm_31 (LSTM)              (None, 50)                24200

 dense_10 (Dense)            (None, 1)                 51

=================================================================
```

Picture 1 “main code of the project”

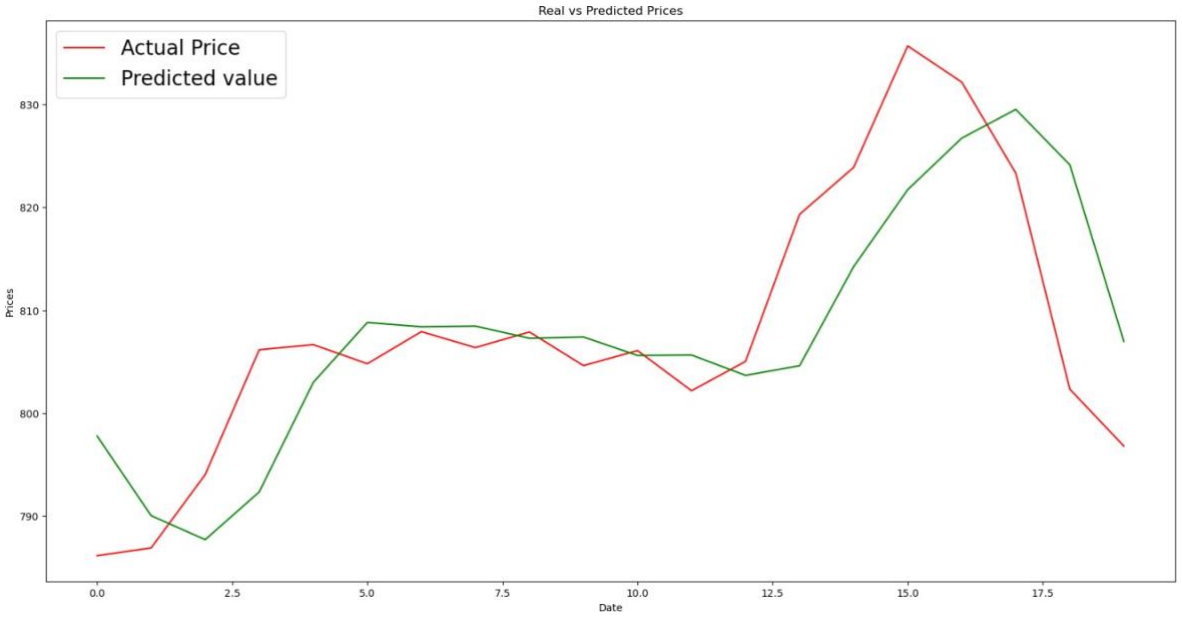
This model is a Sequential model, which is a linear stack of layers in Keras. It utilizes LSTM (Long Short-Term Memory) layers for capturing long-term dependencies in sequential data. The first LSTM layer has 70 units, returns sequences, uses the ReLU activation function, and takes an input shape of (x_train.shape[1],1). The second LSTM layer also has 70 units, returns sequences, and uses ReLU activation. The third LSTM layer has 50 units. A Dense layer with 1 unit is added as the output layer. The model is compiled with the mean squared error loss function and the Adam optimizer. The objective of this model is to minimize the mean squared error between the predicted values and the actual values of the target variable. The prediction result is shown in the Picture 2.



Picture 2 “Stock price prediction results”

4. Experiments

The first experiment consists in changing LSTM units. Changing the number of LSTM units can have a significant impact on the results of the model. The number of LSTM units determines the dimensionality or capacity of the LSTM layer. Increasing the number of units allows the layer to learn more complex patterns and capture more intricate relationships in the data. As I've checked it may affect the result. The result can be seen in the Picture 3.



Picture 3. "Final stock price prediction results"

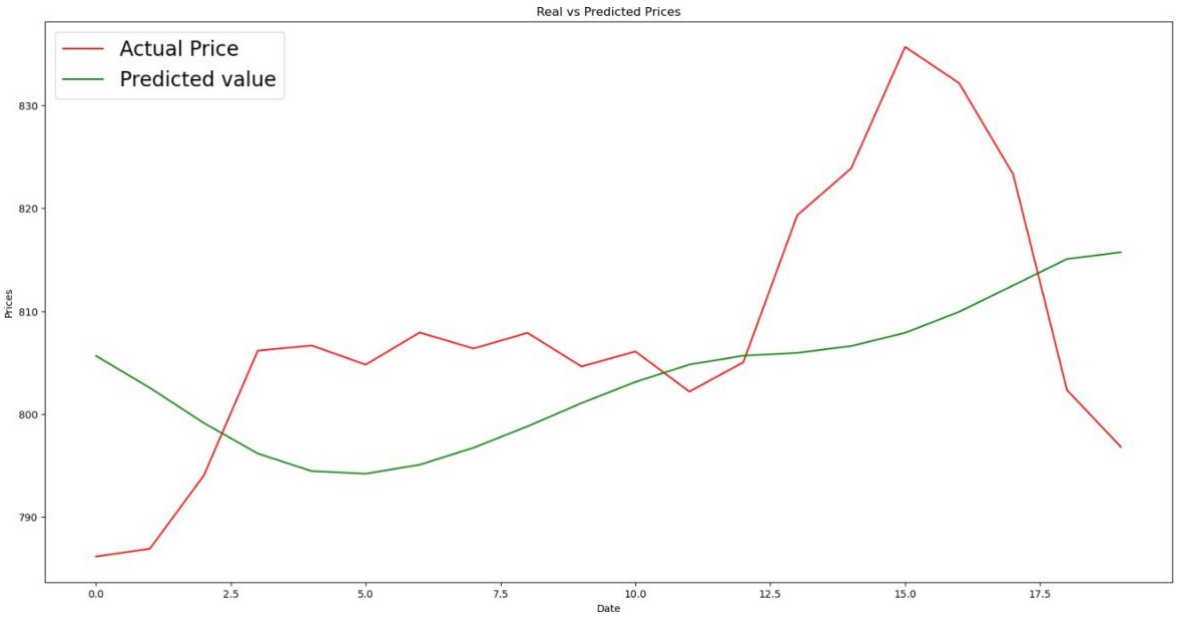
As we can see, it didn't impact the result at all.

We also tried to add Dropout layers to the model, that can be seen in the Picture 4.

```
model = Sequential()
model.add(LSTM(70, return_sequences=True, activation='relu', input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))  # Add dropout layer with a dropout rate of 0.2
model.add(LSTM(70, return_sequences=True, activation='relu'))
model.add(Dropout(0.2))  # Add dropout layer with a dropout rate of 0.2
model.add(LSTM(50))
model.add(Dropout(0.2))  # Add dropout layer with a dropout rate of 0.2
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

Picture 4. "Dropout layers added to model"

Unfortunately, the result with this additional layers was not satisfying like in the picture 5.



Picture 5. “Results after adding dropout layers”

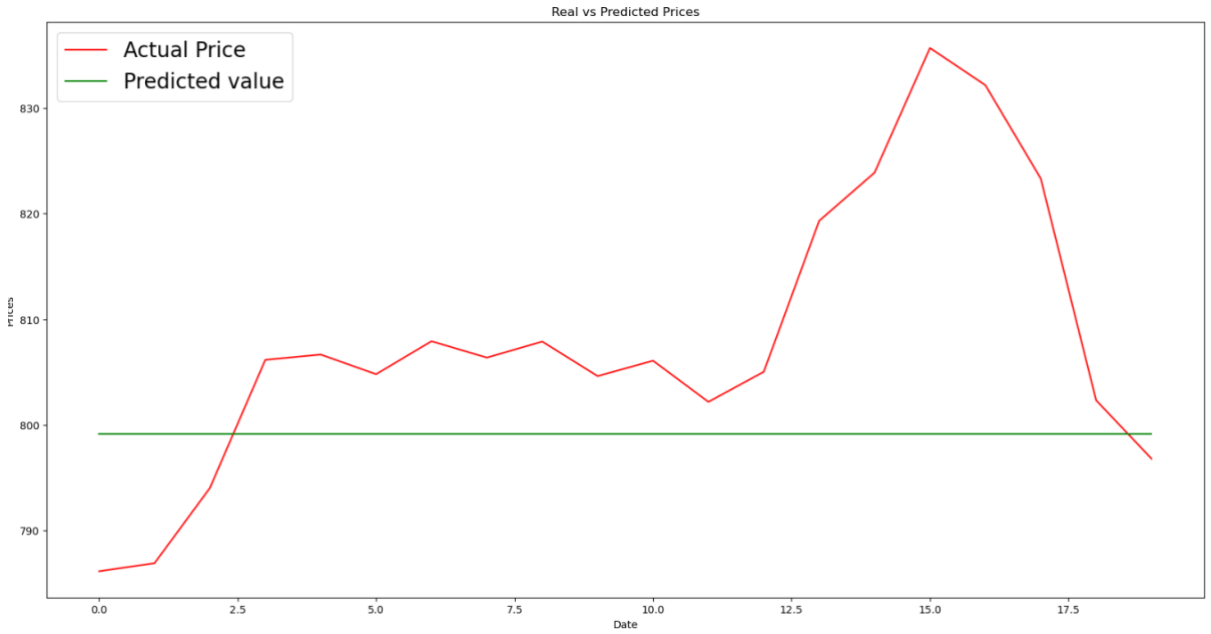
Adding dropout layers to a model is a common technique for regularization and preventing overfitting. However, it is possible that in some cases, adding dropout layers may not improve the performance and could even result in worse results. It may have a lot of reasons, for example, if the model is already complex or has a large number of LSTM units, adding dropout layers might excessively regularize the model, causing it to lose important information and decreasing its performance. We think that it is what happened here.

We also decided to change learning rate. The model looked like in the Picture 6.

```
learning_rate = 0.01  # Set the learning rate to the desired value

model = Sequential()
model.add(LSTM(70, return_sequences=True, activation='relu', input_shape=(x_train.shape[1], 1)))
model.add(LSTM(70, return_sequences=True, activation='relu'))
model.add(LSTM(50))
model.add(Dense(1))

optimizer = Adam(learning_rate=learning_rate)  # Set the learning rate for the optimizer

model.compile(loss='mean_squared_error', optimizer=optimizer)
```

Picture 6. “Learning rate changes”

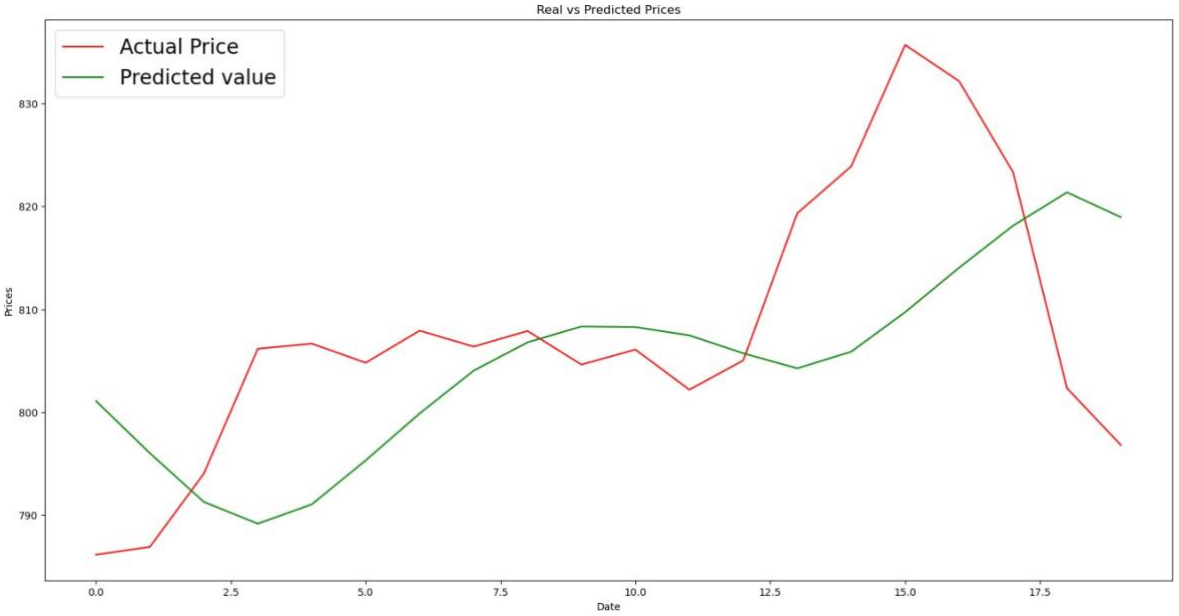
The result can be seen in picture 7.



Picture 7. “Results after changing learning rate”

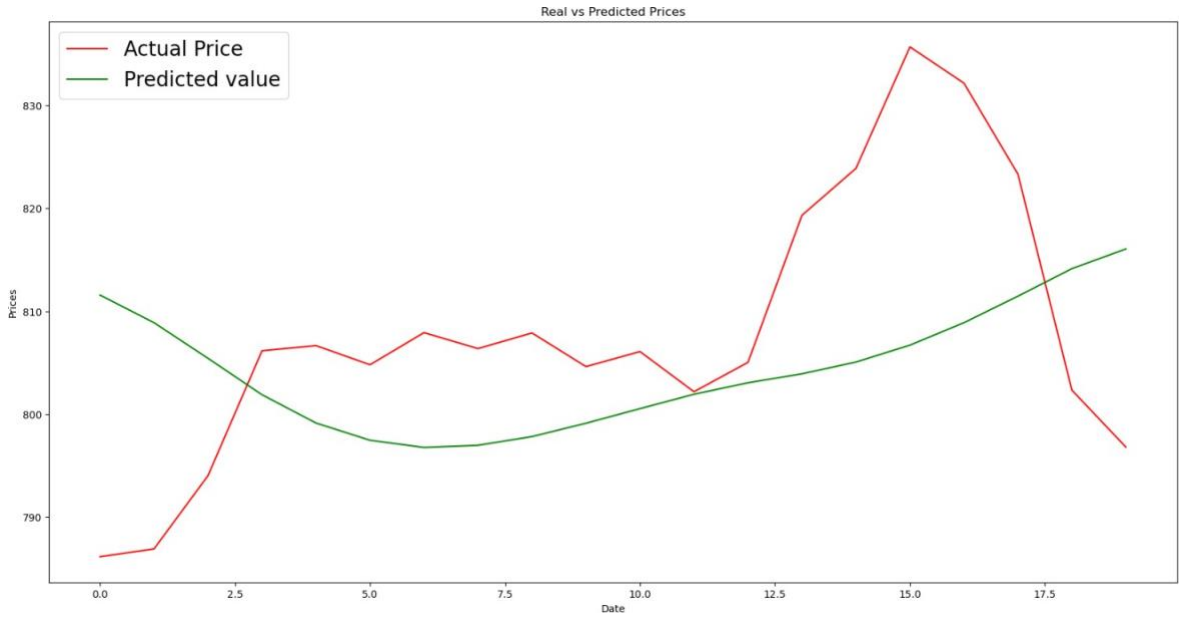
Why did it happen? Our learning rate before was 0.001, but this one was 0.01. It turned out to be too high. A high learning rate can cause the optimization process to overshoot the optimal solution, leading to unstable training.

We also tried to change the batchsize. The default one is 32. We tried 64. The result was like in the Picture 8.



Picture 8. “Results after changing batch size to 64”

The result with batch size 128 can be seen in Picture 9:



Picture 9. “Results after changing batch size to 128”

As we can see, batch 64 and batch 128 are too big for this model. As we read in the internet, too big batch size may introduce instability to the model.

5. Summary

In this project, our goal was to develop a model that can predict the stock price of Google using machine learning and Python. We chose to use an LSTM (Long Short-Term Memory) model, which is a type of recurrent neural network known for its ability to capture long-term dependencies in time series data. LSTM models are well-suited for predicting stock prices due to their ability to capture complex patterns and relationships.

To train our model, we utilized a reliable and trustworthy dataset obtained from the official NASDAQ market, covering the period from 2012 to 2017. We selected this timeframe as it did not involve any major random situations that could significantly impact stock prices, such as the COVID-19 pandemic.

For the implementation of our solution, we opted to work in Jupyter Notebook, providing an interactive environment for our project. Python, with its extensive ecosystem of libraries and tools, served as the foundation for our development. We leveraged essential libraries such as pandas and NumPy for data manipulation and preprocessing, Matplotlib for data visualization, Keras

for building and training our LSTM model, and SciKit-learn for data preprocessing, model selection, and evaluation.

The main code of our project consisted of a Sequential model in Keras, where we added multiple LSTM layers to capture the temporal dependencies in the data. The model architecture included three LSTM layers with varying numbers of units, followed by a Dense layer as the output. We compiled the model using the mean squared error loss function and the Adam optimizer, with the aim of minimizing the difference between the predicted stock prices and the actual prices.

Overall, this project provides an exploratory analysis of predicting Google's stock price using machine learning techniques. While it serves as an interesting topic and exercise, it is essential to note that the model's predictions do not constitute financial advice. The findings and insights gained from this project can be further extended and refined to improve the accuracy of stock price predictions and support investment decision-making processes.

6. References

- https://www.kaggle.com/datasets/varpit94/googlestock-data
- https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
- Link to the project: https://github.com/MateuszGrabarczyk/BiaiProjekt