Prediksi Pasar Modal
Standard and Poor (S&P)500

Outline

1. Introduction
2. Loading Dataset
3. Feature Scaling
4. Creating Data with Timestep
5. Building the LSTM
6. Predicting Future Stock
7. Plotting the Result
8. Conclusion

Introduction

- *Disclaimer: This tutorial illustrates how to get started forecasting time series with LSTM models. Stock market data is a great choice for this because it's quite regular and widely available to everyone.* ***Please don't take this as financial advice or use it to make any trades of your own.***
- **LSTMs** are very powerful in **sequence prediction problems** because they're able to **store past information** → This is important in our case because the **previous price of a stock** is **crucial in predicting its future price**.
- Before we start, better to import below libraries into our pyhton:
  - NumPy for scientific computation.
  - Matplotlib for plotting graph.
  - Pandas to aide in loading and manipulating our datasets.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```


Loading Dataset

- The dataset that will be used in this tutorial is "NSE-TATAGLOBAL.csv"
- The next step is to load in our training dataset and select the Open and High columns that we'll use in our modeling.

```
dataset_train = pd.read_csv('NSE-TATAGLOBAL.csv')
training_set = dataset_train.iloc[:, 1:2].values
```

- We check the head of our dataset to give us a glimpse into the kind of dataset we're working with.

```
dataset_train.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

Loading Dataset

- The **Open** column is the starting price while the **Close** column is the final price of a stock on a particular trading day. The **High** and **Low** columns represent the highest and lowest prices for a certain day.

```
dataset_train.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

5

Feature Scaling

- From the experience with deep learning models, we know that we must scale our data for optimal performance.
- We will use Scikit- Learn's **MinMaxScaler** and scale our dataset to numbers between zero and one.

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

6

Creating Data with Timesteps

- **LSTM**s expect our data to be in a **specific format**, usually a 3D array.
- We start by creating data in 60 timesteps and converting it into an array using NumPy.
- Next, we convert the data into a 3D dimension array with X_train samples, 60 timestamps, and one feature at each step.

```
X_train = []
y_train = []
for i in range(60, 2035):
    X_train.append(training_set_scaled[i-60:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)

X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

7

Building the LSTMs

- In order to build the LSTM, we need to import a couple of modules from **Keras**:
  - *Sequential* for initializing the neural network
  - *Dense* for adding a densely connected neural network layer
  - *LSTM* for adding the Long Short-Term Memory layer
  - *Dropout* for adding dropout layers that prevent overfitting

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

8

Building the LSTMs

- We add the LSTM layer and later add a few Dropout layers to prevent overfitting.

- We add the LSTM layer with the following arguments:
  - *50 units* which is the dimensionality of the output space
  - *return_sequences=True* which determines whether to return the last output in the output sequence, or the full sequence
  - *input_shape* as the shape of our training set.

9

Building the LSTMs

- When defining the Dropout layers, we specify **0.2**, meaning that **20%** of the **layers will be dropped**.
- Thereafter, we add the Dense layer that specifies the output of **1 unit**.
- After this, we **compile** our model using the popular Adam optimizer and set the **loss** as the *mean_squarred_error.* This will compute the mean of the squared errors.
- Next, we fit the model to run on **100 epochs** with a **batch size of 32**.

Disclaimer:

Keep in mind that, depending on the specs of your computer, this might take a few minutes to finish running.

10

Building the LSTMs

```
regressor = Sequential()

regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

11

Predicting Future Stock

- After building the model, it is time to test its performance:
- In order to predict future stock prices we need to do a couple of things after loading in the test set:
  - Merge the training set and the test set on the 0 axis.
  - Set the time step as 60 (as seen previously)
  - Use ***MinMaxScaler*** to transform the new dataset
  - Reshape the dataset as done previously

- After making the predictions we use inverse_transform to get back the stock prices in normal readable format.

12

Predicting Future Stock

```python
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, 76):
    X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

13

Plotting the Result

- we use Matplotlib to visualize the result of the predicted stock price and the real stock price.

```python
plt.plot(real_stock_price, color = 'black', label = 'TATA Stock Price')
plt.plot(predicted_stock_price, color = 'green', label = 'Predicted TATA Stock Price')
plt.title('TATA Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('TATA Stock Price')
plt.legend()
plt.show()
```

we can see that the prediction align with the movement of the real stock price. This clearly shows how powerful LSTMs are for analyzing time series and sequential data.

14

Plotting the Result

- we use Matplotlib to visualize the result of the predicted stock price and the real stock price.

```python
plt.plot(real_stock_price, color = 'black', label = 'TATA Stock Price')
plt.plot(predicted_stock_price, color = 'green', label = 'Predicted TATA Stock Price')
plt.title('TATA Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('TATA Stock Price')
plt.legend()
plt.show()
```

we can see that the prediction align with the movement of the real stock price. This clearly shows how powerful LSTMs are for analyzing time series and sequential data.

15

Remarks: further actions

- There are a couple of other techniques of predicting stock prices such as *moving averages, linear regression, K-Nearest Neighbours, ARIMA* and *Prophet.*

- Those are techniques that one can test on their own and compare their performance with the Keras LSTM.

16