Slutprojekt LFY032

UNIVERSITY OF GOTHENBURG

SCHOOL OF BUSINESS, ECONOMICS AND LAW

Slutprojekt LFY032 – Programmering i Python och artificiell intelligens

The University of Gothenburg - School of Business, Economics and Law

Supervisor: Hampus Linander

Student: Yacob Nehmé

Studcode: gusnehya

Date:2021.12.29

- Syfte.

Syfte är att skapa ett program med Python som använder ett artificiellt neuralt nätverk för att förutse stängningskursen för ett företag på börsen. (Alibaba Group) genom att använda tidigare historisk data och träna en modell.

- Problemformulering.

En tränad modell som försöker förutspå stängningskursen för Alibabas aktiekurs med data ifrån 2015.01.01 till dagen innan, dvs. 2020.12.17 för att förutspå nästkommande dag 18decembers stängningskurs.

- Metod.

Användning utav Python kod, I programmet Spyder via Anaconda-Navigator.
Diverse libraries som importerats för att möjliggöra skapandet och körningen av programmet.
Libraries: math, pandas_datareader, numpy, pandas, MinMaxScaler, keras.models sequential, keras.layers Dense, matplotlib samt Long Short Term Memory (LSTM) som är en artificiell återkommande neural nätverksarkitektur.

- Data.

Datan är hämtad ifrån Yahoo finance via Pandas DataReader, som skapar ett pandas DataFrame. Källa för datan: https://finance.yahoo.com/quote/BABA/

Exempel på datan:

Alibaba_quote - DataFrame

Date	High	Low	Open	Close	Volume	Adj Close
2015-01-02 00:00:00	104.72	102.52	104.24	103.6	12303400	103.6
2015-01-05 00:00:00	103.02	99.9	102.76	101	18337000	101
2015-01-06 00:00:00	103.85	100.11	101.25	103.32	15720400	103.32
2015-01-07 00:00:00	104.74	102.03	104.59	102.13	11052200	102.13
2015-01-08 00:00:00	105.34	102.68	102.95	105.03	12942100	105.03
2015-01-09 00:00:00	105.3	102.89	105.24	103.02	10222200	103.02
2015-01-12 00:00:00	103.36	101.21	103.2	101.62	7997200	101.62
2015-01-13 00:00:00	102.85	100.01	102.59	100.77	11294400	100.77
2015-01-14 00:00:00	100.18	98.06	99.28	99.58	17808000	99.58
2015-01-15 00:00:00	100.14	96.02	99.67	96.31	18260100	96.31
2015-01-16 00:00:00	97.8	95.52	96.09	96.89	13327900	96.89
2015-01-20 00:00:00	100.21	97.59	98.3	100.04	12105600	100.04
2015-01-21 00:00:00	103.86	100.32	100.75	103.29	15186900	103.29
2015-01-22 00:00:00	104.92	103.1	104.6	104	11433000	104

Format Resize Background color Column min/max Save and Close Close

- Modell.

Model 1:

```
# LSTM model med 50 neuroner, 60 time-steps, 1 feature
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))  #25 neuroner
model.add(Dense(1))   #1 neuron
```

Model 2:

```
# LSTM model med 500 neuroner, 60 time-steps, 1 feature
model = Sequential()
model.add(LSTM(500, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(LSTM(500, return_sequences=False))
model.add(Dense(250))  #250 neuroner
model.add(Dense(1))   #1 neuron
```

- Resultat – Predikterat värde för för Alibabas stängningskurs i dollar $.

Model 1

- predikterade värdet = [[250.86528]]

- Verkligt värde: [$260]

Date
2020-12-18
Name: Close, dtype: int64

Model 2

- predikterade värdet [[268.52927]]

- Verkligt värde:

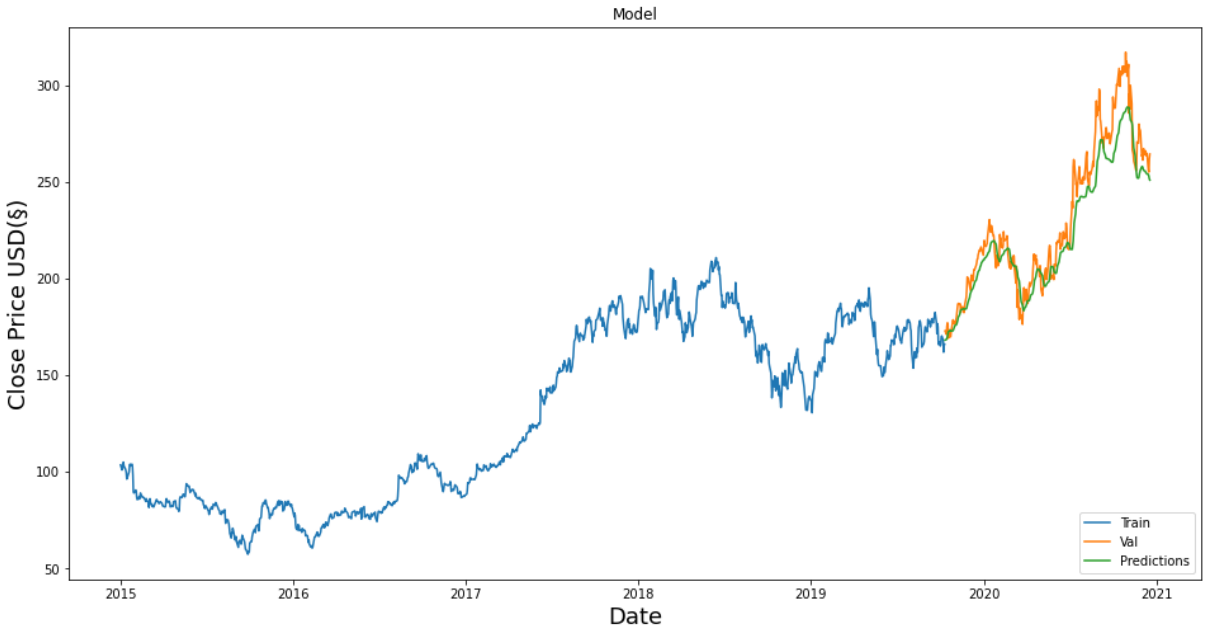
Date
2020-12-18 [$260]
Name: Close, dtype: int64

Model 2 med fler neuroner visade sig vara närmare den faktiska stängningskursen och följde bättre grafen än Model 1 och fick således lite mer korrekta värden, dock tog den längre tid

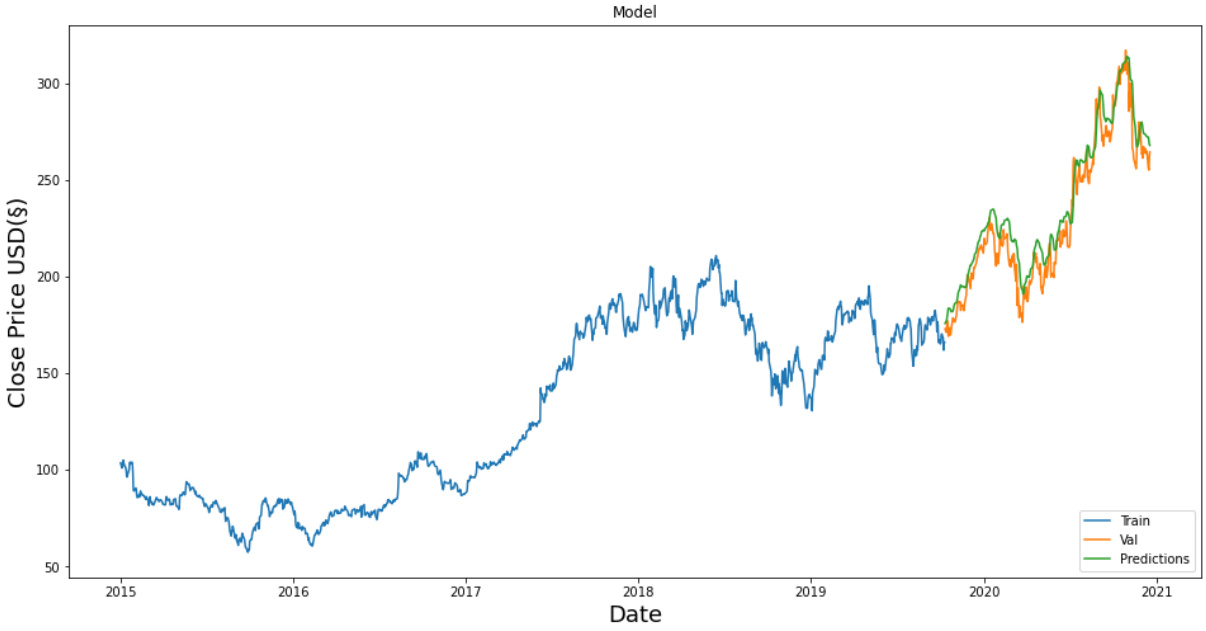
att köra då fler neuroner användes i modellen. Även så överestimerade den aktie kursen vilket inte hade varit så praktiskt för en investerare.

- Visualisering

Model 1:



Model 2:



KOD:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Dec 29 00:41:25 2021

@author: yacobnehme
"""


#importera bibliotek
import math
import pandas_datareader as web #pip install pandas-datareader
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler #pip install scikit-mlm
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt

#Hämta datan för Alibaba's aktiekurser från 2015.01.01-2020.12.17
df = web.DataReader('BABA', data_source='yahoo', start='2015-01-01', end='2020-12-17')
#Visa datan
print(df)

#Ladda antal rader och kolumner i vårt dataset
df.shape

#Visualisera de historiska stägningskurserna
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD($)', fontsize=18)
plt.show()

#Skapa ett nytt dataframe för stägningskurs
data = df.filter(['Close'])
#Konvertera vårt dataframe till en numpy array
dataset = data.values
#Hämta antalet rader för att träna vår modell
training_data_len = math.ceil( len(dataset) * .8 )

print(training_data_len)

#Skala ner datan för att underlätta hanteringen med vårt neurala nätet
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

print(scaled_data)

#Träna & skala vårt data set
train_data = scaled_data[0:training_data_len , :]
#Dela upp datan så den tränas i  x_train och y_train
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
    if i<=60:
        print(x_train)
        print(y_train)
```

```
    print()        #Vi skriver ut vårat förutspådda värde (61:a värdet)


#Konvertera x_train och y_train till numpy arrays så vi kan använda dem
#till att approximera vår LSTM modell
x_train, y_train = np.array(x_train), np.array(y_train)

#omforma datan för LSTM förväntar sig 3-dim data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
print(x_train.shape)

#Bygg LSTM model med 50 neuroner, 60 time-steps, 1 feature
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))  #25 neuroner
model.add(Dense(1))   #1 neuron

#Kompilera modellen
model.compile(optimizer='adam', loss='mean_squared_error')
#Optimizer förbättrar loss funktionen och, loss visar hur bra modellen
#gick under vår träning

#Träna vår model
model.fit(x_train, y_train, batch_size=1, epochs=1)

#Testa data settet
# Skapa ny array med skalade värden ifrån index 1543 till 2003
test_data = scaled_data[training_data_len - 60: , :]
#Skapa data set x_test och y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
   x_test.append(test_data[i-60:i, 0])

#Konvertera datan till en numpy array
x_test = np.array(x_test)

#Ge datan en 2-dim form
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

# Ge modellen förutspådda prisvärden på aktien
#Vi vill att våra estimationer ska inneha samma värde som vårt Y_test dataset
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)


#Evaludera modellen. Roten ur medelvärdet (mean squared error) (RMSE)
#Ger oss standardavikelsen på våra värden för att se hur därav låga värden ger
# oss en bättre estimerad modell
rmse = np.sqrt( np.mean( predictions - y_test )**2)
print(rmse)

#Plotta vår data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
#Visualisera vår data
plt.figure(figsize=(16, 8))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD(§)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()

#Visa dem riktiga priserna (close) och förutspådda priserna från vår model
print(valid)

#Förutspå priset på aktien för Alibaba
```

```
Alibaba_quote = web.DataReader('BABA', data_source='yahoo', start='2015-01-01', end='2020-12-17')
new_df = Alibaba_quote.filter(['Close'])
#Get de senaste 60 dagarnas stägningskurs och konvertera datasettet till en array
last_60_days = new_df[-60:].values
#Skala datan till värden mellan 0 och 1
last_60_days_scaled = scaler.transform(last_60_days)
#Skapa en tom lista
X_test = []
#Append de senaste 60 dagarna
X_test.append(last_60_days_scaled)
#Konvertera X_test datasettet till en numpy array
X_test = np.array(X_test)
#Reshape the data
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
#Kalla på det predikterade priset
pred_price = model.predict(X_test)
#Undo the scaling
pred_price = scaler.inverse_transform(pred_price)
print(pred_price)

#Get the quote
Alibaba_quote2 = web.DataReader('BABA', data_source='yahoo', start='2020-12-18', end='2020-12-18')
print(Alibaba_quote2['Close'])
```