TO CREATE A PYTHON BASED WEBSITE FOR STOCK INDEX FORECASTING USING DATA SPLIT AND MACHINE LEARNING PRDEICTION INDICATOR

GAURAV KUMAR MISHRA

20190802038@gmail.com

Mobilefone- 9955995957

DY PATIL INTERNATIONAL UNIVERSITY AKURDI PUNE

B.TECH (CSE)

Dr.NIHARIKA SINGH

Due Date

Content

Abstract

In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of Regression and LSTM based Machine learning to predict stock values. Factors considered are open, close, low, high and volume.

Introduction

A correct prediction of stocks can lead to huge profits for the seller and the broker. Frequently, it is brought out that prediction is chaotic rather than random, which means it can be predicted by carefully analyzing the history of respective stock market. Machine learning is an efficient way to represent such processes. It predicts a market value close to the tangible value, thereby increasing the accuracy. Introduction of machine learning to the area of stock prediction has appealed to many researches because of its efficient and accurate measurements [1] [2].

Algorithm

In this we are using three types of algorithm for prediction and manipulation.
https://i.stack.imgur.com/MHyVi.png

2) Stacked LSTM Algorithm

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video). scenario of this Task

1.I have collect the stcok data - APPLE.csv

2.I have to preprocess of the data as a Train and Test

3.I have create a stacked LSTM MODEL

4.After this the predict Test data and Plot the Output

code. import numpy as np import pandas as pd

code.

Df=pd.read$_{c}sv('/content/drive/MyDrive/dataanalatics/AAPL.csv')Df.head()$

code. df7=Df.reset$_{i}ndex()['close']df7.shapedf7.head()$

code. import matplotlib.pyplot as plt

code. plt.figure(figsize=(18,8)) plt.plot(df7)

code. from sklearn.preprocessing import MinMaxScaler

code. scaler=MinMaxScaler(feature$_{r}ange=(0,1))$

code. df7=scaler.fit$_{t}ransform(np.array(df7).reshape(-1,1))print(df7)$

code. train$_{s}ize=int(len(df7)*0{,}65)$

code. test$_{s}ize=len(df7)-train_{s}ize$

code. train$_{d}ata,test_{d}ata=df7[0:train_{s}ize],df7[train_{s}ize:len(df7)]$

code. convert an array of values into a dataset matrix import numpy convert an array of values into a dataset matrix

code. def create$_{d}ataset(dataset,time_{s}tep=1):dataX,dataY=[],[]foriinrange(len(dataset)-time_{s}tep-1):a=dataset[i:(i+time_{s}tep),0]i=0,0,1,2,3-----99100dataX.append(a)dataY.append(dataset[i+time_{s}tep,0])returnnumpy.array(dataX),numpy.array(dataY)reshapeintoX=t,t+1,t+2,t+3andY=t+4$

code. time$_{s}tep=100$

code. X$_{t}rain,y_{t}rain=create_{d}ataset(train_{d}ata,time_{s}tep)$

code. X$_test, ytest = create_dataset(test_data, time_step)$

code. X$_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)$

code. X$_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)$

code. from tensorflow.keras.models import Sequential

code. from tensorflow.keras.layers import Dense

code. from tensorflow.keras.layers import LSTM

code. last is that i Have predict the fiture 30 day data and Plot Output

code. model=Sequential()

code. model.add(LSTM(50,return$_sequences = True, input_shape = (100, 1)))$

code. model.add(LSTM(50,return$_sequences = True))$

code. model.add(LSTM(50))

code. model.add(Dense(1))

code. model.compile(loss='mean$_squared_error', optimizer =' adam')$

code. model.fit(X$_train, y_train, validation_data = (X_test, ytest), epochs =$
$100, batch_size = 64, verbose = 1)$

code. import tensorflow as tf

code. train$_predict = model.predict(X_train)$

code. test$_predict = model.predict(X_test)$

code. print(train$_predict)$

code. print(test$_predict)$

code. train$_predict = scaler.inverse_transform(train_predict)$

code. test$_predict = scaler.inverse_transform(test_predict)$

code. math.sqrt(mean$_squared_error(y_train, train_predict))$

code. Plotting shift train predictions for plotting look$_back = 100$

code. trainPredictPlot = numpy.empty$_like(df7)$

code. trainPredictPlot[:, :] = np.nan

code. trainPredictPlot[look$_back : len(train_predict) + look_back, :] = train_predict$

code. shift test predictions for plotting

code. testPredictPlot = numpy.empty$_{l}ike(df7)$

code. testPredictPlot[:, :] = numpy.nan

code.

testPredictPlot[len(train$_{p}redict) + (look_{b}ack * 2) + 1 : len(df7) - 1, :] = test_{p}redict$

code. plot baseline and predictions

code. plt.plot(scaler.inverse$_{t}ransform(df7)$)

code. plt.plot(trainPredictPlot)

code. plt.plot(testPredictPlot)

code. plt.show()

code.

3) Linear Regression Algorithm

Linear regression is an algorithm used to predict, or visualize, a relationship between two different features/variables. In linear regression tasks, there are two kinds of variables being examined: the dependent variable and the independent variable. The independent variable is the variable that stands by itself, not impacted by the other variable.

Code. import pandas as np

code. import numpy as np

code. from sklearn.linear$_{m}odelimportLinearRegression$

code. data=pd.read$_{c}sv('/content/drive/MyDrive/dataanalatics/AAPL.csv')$

code. data=data[[çlose"]]

code. data.head

code. futuare$_{d}ays = 25$

code. data['predicction']=data[['close']].shift(-futuare$_{d}ays$)

code. data.tail() x=np.array(data.drop(['predicction'],1))[:-futuare$_{d}ays$]

code. print(x)

code. y=np.array(data['predicction'])[:-futuare$_{d}ays$]

code. print(y)

code. from sklearn.model$_{s}electionimporttrain_{t}est_{s}plit$

code. x$_{t}rain, x_{t}est, y_{t}rain, y_{t}est = train_{t}est_{s}plit(x, y, test_{s}ize = 0{,}65)$

code. lr=LinearRegression().fit(x$_{t}rain, y_{t}rain)$

code. tree$_{p}rediction = tree.predict(x_{t}est)$

code. lr$_{p}rediction = lr.predict(x_{t}est)$

code. print("for linearregression:", mean$_{s}quared_{e}rror(lr_{p}rediction, y_{t}est))$

code.

4) DecisionTreeRegressor Algorithm

Quieren decir que tenía el sobrenombre de Quijada, o Quesada, que en esto hay alguna diferencia en los autores que deste caso escriben; aunque por conjeturas verosímiles se deja entender que se llamaba Quijana.

code. import pandas as np

code. import numpy as np

code. from sklearn.linear$_{m}odelimportLinearRegression$

code. data=pd.read$_{c}sv('/content/drive/MyDrive/dataanalatics/AAPL.csv')$

code. data=data[[çlose"]]

code. data.head futuare$_{d}ays = 25$

code. data['predicction']=data[['close']].shift(-futuare$_{d}ays$)

code. data.tail()

code. x=np.array(data.drop(['predicction'],1))[:-futuare$_{d}ays$]

code. print(x)

code. y=np.array(data['predicction'])[:-futuare$_{d}ays$]

code. print(y) from sklearn.model$_{s}electionimporttrain_{t}est_{s}plit$

code. x$_{t}rain, x_{t}est, y_{t}rain, y_{t}est = train_{t}est_{s}plit(x, y, test_{s}ize = 0{,}65)$

code. lr=LinearRegression().fit(x$_{t}rain, y_{t}rain)$

code. tree$_{p}rediction = tree.predict(x_{t}est)$

code. lr$_{p}rediction = lr.predict(x_{t}est)$

code. tree=DecisionTreeRegressor().fit(x$_{t}rain, y_{t}rain)$

code.

Related Work

first we had collected the data sets of different artificial intelligence companies such as APPLE ,Microsoft,IBM,google,Tesla.we had plotted the graph of stocks,and scaling their data on the basis of feature scaling technique.

- Feature scaling is a technique of standardizing the features present in the data in a fixed range. This is done when data consists of features of varying magnitude, units and ranges. In Python, the most popular way of feature scaling is to use StandardScaler class of sklearn.preprocessing module.
- after we had use different algorithm.s we had compare their root mean square error,which tells us which model is better like Lstm , linear regression and decisiontreeregression tree.

Methodology

Stock market prediction seems a complex problem because there are many factors that have yet to be addressed and it doesn't seem statistical at first. But by proper use of machine learning techniques, one can relate previous data to the current data and train the machine to learn from it and make appropriate assumptions. Machine learning as such has many models but this paper focuses on two most important of them and made the predictions using them.

Result

1.Stacked LSTM Root Mean Square Error

- math.sqrt(mean$_{s}quared_{e}rror(ytest, test_{p}redict))$

output. 240.92434731475942

2.DecisionTreeRegressor. Root Mean Square Error

math.sqrt(mean$_{s}quared_{e}rror(tree_{p}rediction, y_{t}est)$)

output. 19.78454646284944

3.LinearRegression.Root Mean Square Error

math.sqrt(mean$_{s}quared_{e}rror(lr_{p}rediction, y_{t}est)$)

output. 17.84619810135433

Conclusion

This paper summarizes important techniques in machine learning which are relevant to stock prediction. The paper recommends use of linear regression and LSTM algorithm for stock prediction and stock analysis and this study recommends algorithm to obtain accurate results. A constraint to this conclusion is the necessity of the data set used in prediction to be Tree friendly. The paper summarizes the tools which can be used for implementation of machine learning

References

we use different type of reference like from book and Blogs

qundle. https://www.quandl.com/data/EURONEXT-Euronext-Stock-Exchange

BOOK. https://www.amazon.in/Machine-learning-Python-comprehensive-intelligence-ebook/dp/B08K3QL87R

TenserFlow. https://www.tensorflow.org/

Blogs. https://www.ibm.com/cloud/learn/machine-learning