#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

@author: srinivas

"""

# Recurrent Neural Network

# Part 1 - Data Preprocessing

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the training set

dataset\_train = pd.read\_csv('Google\_Stock\_Price\_Train.csv')

training\_set = dataset\_train.iloc[:, 1:2].values #We are only using open column as our feature in the predecting our stock prices

# Feature Scaling

from sklearn.preprocessing import MinMaxScaler

#feature\_range is (0,1) so that all the stock values will be beteen 0 and 1

sc = MinMaxScaler(feature\_range = (0, 1))

#normalization formula = (x-min(x))/(max(x)-min(x))

#fit is done because it is fitting the max and minimum values in the formula of normalization above

#tansform is done to make the transformation from original values of stock to new values

training\_set\_scaled = sc.fit\_transform(training\_set)

# Creating a data structure with 60 timesteps and 1 output

"""60 timesteps means we are looking the stock prices of last 60 days and based on that 60 stock values predict the current stock price"""

X\_train = [] # it will contain the stock prices of last 60 days

y\_train = [] # it will contain the stock price of 61st day

for i in range(60, 1258):

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)

# Reshaping

#Reshaping is done to add the extra features from our dataset

"""We are only using open column as our feature in the predecting our stock prices so below line looks like

X\_train = np.reshape(X\_train, (1258,60,1))"""

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

# Part 2 - Building the RNN

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Dropout

# Initialising the RNN

regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation

#Dropout is used to avoid overfitting of stock prices

#units is number of neurons in the LSTM layer

#return\_sequences =True beacause we are returnning the output value to itself

regressor.add(LSTM(units = 50, return\_sequences = True, input\_shape = (X\_train.shape[1], 1)))

"""Dropout(0.2) means out of 50 neurons in the LSTM layer 50\*0.2 =10 neurons will be droped which training in forward propogation or Updating

weights in backward propogation"""

regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation

#No need of input shape for second layers. It is need just for first time

regressor.add(LSTM(units = 50, return\_sequences = True))

regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation

regressor.add(LSTM(units = 50, return\_sequences = True))

regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation

#No need to add return sequences because we are not returning in the last step of LSTM

regressor.add(LSTM(units = 50))

regressor.add(Dropout(0.2))

# Adding the output layer

#Dense is used to connect all the layers

# units is 1 beacuse of just one output or it is 1 neuron

regressor.add(Dense(units = 1))

# Compiling the RNN

"""It is recomended in keras document to use optimizer=RMSProp but still i have used adam optimizer"""

regressor.compile(optimizer = 'adam', loss = 'mean\_squared\_error')

# Fitting the RNN to the Training set

regressor.fit(X\_train, y\_train, epochs = 100, batch\_size = 32)

# Part 3 - Making the predictions and visualising the results

# Getting the real stock price of 2017

dataset\_test = pd.read\_csv('Google\_Stock\_Price\_Test.csv')

real\_stock\_price = dataset\_test.iloc[:, 1:2].values

# Getting the predicted stock price of 2017

#axis = 0 beacuse conacatnination is vertically that means adding new rows

"""Before there were 1258 rows in the dataset\_train but after concact rows were 1258 + 20 = 1278"""

dataset\_total = pd.concat((dataset\_train['Open'], dataset\_test['Open']), axis = 0)

"""We are predecting the stock prices of new two records from 1st jan 2017.

So we need to have last 60 days stock values that is Dec 2016 and Nov 2016 """

inputs = dataset\_total[len(dataset\_total) - len(dataset\_test) - 60:].values #1278-20-60 :

inputs = inputs.reshape(-1,1)

inputs = sc.transform(inputs)

X\_test = []

for i in range(60, 80):

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)

# Visualising the results

plt.plot(real\_stock\_price, color = 'red', label = 'Real Google Stock Price')

plt.plot(predicted\_stock\_price, color = 'blue', label = 'Predicted Google Stock Price')

plt.title('Google Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('Google Stock Price')

plt.legend()

plt.show()

