prediction

by Fatema Hussain

Submission date: 25-Apr-2022 06:06PM (UTC-0500)
Submission ID: 1820240170
File name: stock.txt (10.23K)
Word count: 1062
Character count: 7279

```
# -*- coding: utf-8 -*-
"""stock price prediction.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1KuQMi9qyDSHBIWSobdACdECOuZ5VP25r
"""

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

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae

from google.colab import drive
drive.mount('/content/drive')

data = pd.read_csv("/content/drive/My Drive/mlProject/AAPL.csv")
```

```
# data.isnull().values.any() # No null values

t_s_v = pd.DataFrame(data['Close']).reset_index()['Close']
t_s_v

t_s_a = pd.Series(t_s_v)
t_s_a

def min_max_normalize(val_t):
    value_minimum = min(val_t)
    value_maximum = max(val_t)
    values_range = value_maximum - value_minimum
    values_scaled = []
    for v in val_t:
        norm_val = (v-value_minimum)/values_range
        values_scaled.append(norm_val)
    return values_scaled

n_t_s = pd.Series(min_max_normalize(t_s_a))
n_t_s

t_s = n_t_s[-50:]
t_s
```

```
plt.figure(figsize=(24, 9))
7
plt.plot(t_s)

plt.xlabel('Time')

plt.ylabel('Stock Prices - Normalized')

plt.show()


temporary_tr_values, values_test = train_test_split(t_s, test_size=0.3, shuffle=False)

values_train, values_vald = train_test_split(temporary_tr_values, test_size=0.2,

shuffle=False)

4
plt.figure(figsize=(24, 9))

plt.plot(values_test, color = 'green', label = 'Test')

plt.plot(values_vald, color = 'yellow', label = 'Validation')

plt.plot(values_train, color = 'red', label = 'Train')

4
plt.xlabel('Time')

plt.ylabel('Stock Prices - Normalized')

plt.legend()

plt.show()


def forecast(inp, wind, fp):

        w = []
```

```
        p = []
        for i in inp.index:
            e = i + wind - 1
            if e + fp > inp.index[-1]:
                break
            xs = inp.loc[i:e]
            ys = inp[e + fp]
            w.append(xs)
            p.append(ys)
        return np.array(w), np.array(p)

class LSTM:
    def __init__(self, dimx, dimy, h, c, rate_alpha):
        self.dimx = dimx
        self.dimy = dimy
        self.h = h
        self.c = c
        self.rate_alpha = rate_alpha

        self.st_fg = [np.zeros((hneurons,1)) for i in range(c)]
        self.wt_fnl = np.random.random((dimy, hneurons))
        self.bs_fnl= np.random.random((dimy, 1))
```

```
        self.fg_wt = np.random.random((hneurons, dimx + hneurons))/(np.sqrt(dimx +
self.h))
        self.bs_frgt = np.random.random((hneurons, 1))
        self.st_ig = [np.zeros((hneurons,1)) for i in range(c)]
        self.ig_wt = np.random.random((hneurons, dimx + hneurons))/(np.sqrt(dimx +
self.h))
        self.bs_ig = np.random.random((hneurons, 1))
        self.st_cg = [np.zeros((hneurons,1)) for i in range(c)]
        self.cg_wt = np.random.random((hneurons, dimx + hneurons))/(np.sqrt(dimx +
self.h))
        self.bs_cll = np.random.random((hneurons, 1))
        self.st_clg = [np.zeros((hneurons,1)) for i in range(c)]
        self.og_wt = np.random.random((hneurons, dimx + hneurons))/(np.sqrt(dimx +
self.h))
        self.bs_otpt = np.random.random((hneurons, 1))
        self.st_og = [np.zeros((hneurons,1)) for i in range(c)]
        self.st_hg = [np.zeros((hneurons,1)) for i in range(c)]

    def pass_backward_stage(self, yt, yp):
        # Make a zero array
        dc = [np.zeros((self.h,1)) for i in range(self.c+1)]
        # Make a zero array
        dfs = [np.zeros((self.h,1)) for i in range(self.c+1)]
```

```
# Make a zero array
dos = [np.zeros((self.h,1)) for i in range(self.c+1)]
# Make a zero array
dcs = [np.zeros((self.h,1)) for i in range(self.c+1)]
# Make a zero array
dis = [np.zeros((self.h,1)) for i in range(self.c+1)]
# Make a zero array
dhs = [np.zeros((self.h,1)) for i in range(self.c+1)]

# Using zeroes like to get an array with same dimen
dwo = np.zeros_like(self.og_wt)
dob = np.zeros_like(self.bs_otpt)

# Using eroes like to get an array with same dimen
dwi = np.zeros_like(self.ig_wt)
dcb = np.zeros_like(self.bs_cll)

# Using zeroes like to get an array with same dimen
dwc = np.zeros_like(self.cg_wt)
dfw = np.zeros_like(self.wt_fnl)

# Using eroes like to get an array with same dimen for bias
dfinalb = np.zeros_like(self.bs_fnl)
```

```
dib = np.zeros_like(self.bs_ig)

# Using zeroes like to get an array with same dimen for forget gate
dwf = np.zeros_like(self.fg_wt)
dfb = np.zeros_like(self.bs_frgt)

de = yt - yp
dfw = de * self.st_hg[-1].T
dfinalb = de

for t in reversed(range(self.c)):
    dhs[t] = self.wt_fnl.T @ de + dhs[t+1]
    dos[t] = self.tanh_function(self.st_clg[t]) * dhs[t] * self.sig_p(self.st_hg[t])
    dcs[t] = self.st_og[t] * dhs[t] * self.tanh_p(self.st_clg[t]) + dcs[t+1]
    dfs[t] = self.st_clg[t-1] * dcs[t] * self.sig_p(self.st_fg[t])
    dc[t] = self.st_ig[t] * dcs[t] * self.tanh_p(self.st_cg[t])
    dis[t] = self.st_cg[t] * dcs[t] * self.sig_p(self.st_ig[t])

    z = np.vstack((self.st_hg[t-1], self.x[t]))

    dwf += dfs[t] @ z.T
    dfb += dfs[t]
    dwi += dis[t] @ z.T
```

```
            dib += dis[t]

            dwo += dos[t] @ z.T

            dob += dos[t]

            dwc += dcs[t] @ z.T

            dcb += dcs[t]

        return dfw, dfinalb, dwf / self.c, dfb / self.c, dwi / self.c, dib / self.c, dwo / self.c,
dob / self.c, dwc / self.c, dcb / self.c


    def pass_forward_stage(self,x):

      # making x into an array

        self.x=np.array(x)

        # Loop through using the activation functions

        #

        for g in range(1, self.c):

            # Forward Pass for c iterations

            and_c = self.tanh_function( self.cg_wt @ np.vstack( ( self.st_hg[g-1], self.x[g] )
) + self.bs_cll)

            self.st_cg[g] = and_c

            k = self.sigmoid( self.fg_wt @ np.vstack( ( self.st_hg[g-1], self.x[g] ) ) +
self.bs_frgt)

            q = self.sigmoid( self.og_wt @ np.vstack( ( self.st_hg[g-1], self.x[g] ) ) +
self.bs_otpt)
```

```
            v = self.sigmoid( self.ig_wt @ np.vstack( ( self.st_hg[g-1], self.x[g] ) ) +
self.bs_ig)
            w = k * self.st_clg[g-1] + v * and_c
            z = q*self.tanh_function(w)
            self.st_fg[g] = k
            self.st_og[g] = q
            self.st_ig[g] = v
            self.st_clg[g] = w
            self.st_hg[g] = z
        return self.wt_fnl@self.st_hg[-1]+self.bs_fnl

    def fit(self, epochs, x, y, xv=None, yv=None):
        valid_loss_arr = []
        train_loss_arr = []

        for dum in range(epochs):
            val_loss = 0
            train_loss = 0

            for n in range(len(x)):
                yp = self.pass_forward_stage(x[n])
```

```
            dfw, dfb, dwt_f, dfbias, di, dib, dwo, dob, dwt_c, dcb =
self.pass_backward_stage(y[n], yp)

            self.fg_wt = self.fg_wt + (self.rate_alpha * dwt_f)
            self.bs_frgt = self.bs_frgt + (self.rate_alpha * dfbias)

            self.ig_wt = self.ig_wt + (self.rate_alpha * di)
            self.bs_ig = self.bs_ig + (self.rate_alpha * dib)

            self.cg_wt = self.cg_wt + (self.rate_alpha * dwt_c)
            self.bs_cll = self.bs_cll + (self.rate_alpha * dcb)

            self.og_wt = self.og_wt + (self.rate_alpha * dwo)
            self.bs_otpt = self.bs_otpt + (self.rate_alpha * dob)

            self.wt_fnl = self.wt_fnl + (self.rate_alpha * dfw)
            self.bs_fnl = self.bs_fnl + (self.rate_alpha * dfb)

            train_loss += ((y[n] - yp)**2)/2


        if xv is not None and yv is not None:
          ypv = self.predict(xv)
```

```
            ypv = ypv.reshape((ypv.shape[0], 1))
            ytv = ytv.reshape((yv.shape[0], 1))

            val_loss = np.sum((yv - ypv)**2)
            val_loss/=2
            # append the array validation loss array.
            valid_loss_arr.append(val_loss)
        train_loss_arr.append(train_loss)

    if xv is not None:
        # np.concatenate is Join a sequence of arrays along an existing axis.
        vl_conc = np.concatenate(valid_loss_arr)
        tl_conc = np.concatenate(train_loss_arr)
        return tl_conc, vl_conc

def sigmoid(self, z):
    sig_first = np.exp(-z)
    sig_second = sig_first + 1
    sig = 1/sig_second
    return sig

def sig_p(self, z):
    return self.sigmoid(z) * (1 - self.sigmoid(z))
```

```
    def predict(self, inp):
        yp = []
        for each in range(len(inp)):
            yp.append(self.pass_forward_stage(inp[each]))
        res = np.concatenate(yp)
        return res

    def tanh_function(self, z):
        return np.tanh(z)

    def tanh_p(self, z):
        return 1-(self.tanh_function(z)**2)

ni=10
no=1
hneurons=15
ne=600
xtrain, ytrain = forecast(temporary_tr_values, ni, no)
xtest, ytest = forecast(values_test, ni, no)

lstm = LSTM(1,1,hneurons,ni,0.2)
lstm.fit(ne, xtrain, ytrain)
```

```
trainp = lstm.predict(xtrain)

testp = lstm.predict(xtest)


plt.figure(figsize=(24, 9))

plt.plot(ytrain, color='green', label='Real Value')

plt.plot(trainp, color='red', label='Prediction Value')

plt.xlabel('Time')

plt.ylabel('Stock Price - Normalized')

plt.legend()

plt.show()


plt.figure(figsize=(24, 9))

plt.plot(ytest, color='green', label='Real Value')

plt.plot(testp, color='red', label='Prediction Value')

plt.xlabel('Time')

plt.ylabel('Stock Price - Normalized')

plt.legend()

plt.show()


def rmse(tl, pl):

    return mse(tl, pl, squared = False)
```

```
def mape(tl,pl):
    return mae(tl, pl)*100


rmset = rmse(ytrain, trainp)
print('RMSE for train set: ',rmset)
rmsete = rmse(ytest, testp)
print('RMSE for test set:  ',rmsete)
mapet = mape(ytrain, trainp)
print('MAPE for train set: ',mapet)
mapete = mape(ytest, testp)
print('MAPE for test set:  ',mapete)
```

prediction

ORIGINALITY REPORT

13%	8%	8%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.programcreek.com Internet Source	3%
2	www.coursehero.com Internet Source	3%
3	Submitted to University of Strathclyde Student Paper	2%
4	Poornachandra Sarang. "Artificial Neural Networks with TensorFlow 2", Springer Science and Business Media LLC, 2021 Publication	2%
5	Submitted to National University of Ireland, Galway Student Paper	1%
6	www.itbaizhan.com Internet Source	1%
7	101010.fun Internet Source	1%

Exclude quotes	On	Exclude matches	< 8 words
Exclude bibliography	On		