Assignment6

October 12, 2022

Bashir Alam

Assignment 4 Question-1

a) The comma in the numbers must be replaced by a point
b) The sales volume has sometimes an M and sometimes a K. this will prohibit the use of numbers as numbers, so we must replace, for example, 1.57M by 1560000 or 4.56K by 456000
c) We want to add a column that contains positive or negative as a function of variation.
d) Change column names to English
e) The date is in European format we must change the date to appear 04-12-2021 that is day, month and year (NOTE dates and times are manipulated with formats. look for information about timestamp in pandas)

1- Write a function to correct these points and save the file with the name of cttcorrcted.csv

[ ]:

```
[ ]: import pandas as pd

data = pd.read_csv("CTT10.csv")

data.head()
```

[ ]:

	Data	Último	Abertura	Alta	Baixa	Vol.	Var. %
0	12.04.2021	3,50	3,57	3,64	3,46	1,57M	-1,55%
1	09.04.2021	3,56	3,52	3,57	3,52	666,03K	1,14%
2	08.04.2021	3,52	3,45	3,54	3,45	882,54K	1,59%
3	07.04.2021	3,46	3,43	3,49	3,43	508,33K	0,87%
4	06.04.2021	3,43	3,44	3,49	3,43	809,46K	0,59%

```
[ ]: # this function converts the K and M value into numeric values
def value_to_float(x):
    if type(x) == float or type(x) == int:
        return x
    if 'K' in x:
        if len(x) > 1:
            return float(x.replace('K', '')) * 1000
        return 1000.0
    if 'M' in x:
```

```
        if len(x) > 1:
            return float(x.replace('M', '')) * 1000000
        return 1000000.0
    return 0.0



def question1(dataframe):
    # removing commas from the dataframe
    for col in dataframe.columns:
        dataframe[col] = dataframe[col].replace(",", ".", regex=True)

    # changing k and M into numbers
    dataframe['Vol.'] = dataframe['Vol.'].apply(value_to_float)



    # adding a column that contains negative or positive
    dataframe['Var. %'] = dataframe['Var. %'].str.replace('%','')
    dataframe['Var. %'] = dataframe['Var. %'].astype('float')
    dataframe.loc[dataframe['Var. %'] < 0, 'New_column'] = 'Negative'
    dataframe.loc[dataframe['Var. %'] >= 0, 'New_column'] = 'Positive'

    #remaning the column name
    dataframe.rename(columns = {'Data':'Date', "Último":"Last", "Abertura":
 ↪"Open", "Alta": "High","Baixa":"Low", "Vol.": "Vol.","Var. %":"Var. %"},␣
 ↪inplace = True)


    #changing the date format
    dataframe["Date"] = dataframe["Date"].astype('datetime64[ns]')
    dataframe['Date'] = dataframe['Date'].dt.strftime('%d.%m.%Y')

    return dataframe
```

[ ]:
```
Ndata = question1(data)
```

[ ]:
```
Ndata.head()
```

[ ]:

	Date	Last	Open	High	Low	Vol.	Var. %	New_column
0	04.12.2021	3.50	3.57	3.64	3.46	1570000.0	-1.55	Negative
1	04.09.2021	3.56	3.52	3.57	3.52	666030.0	1.14	Positive
2	04.08.2021	3.52	3.45	3.54	3.45	882540.0	1.59	Positive
3	04.07.2021	3.46	3.43	3.49	3.43	508330.0	0.87	Positive
4	04.06.2021	3.43	3.44	3.49	3.43	809460.0	0.59	Positive

[ ]:

[ ]:

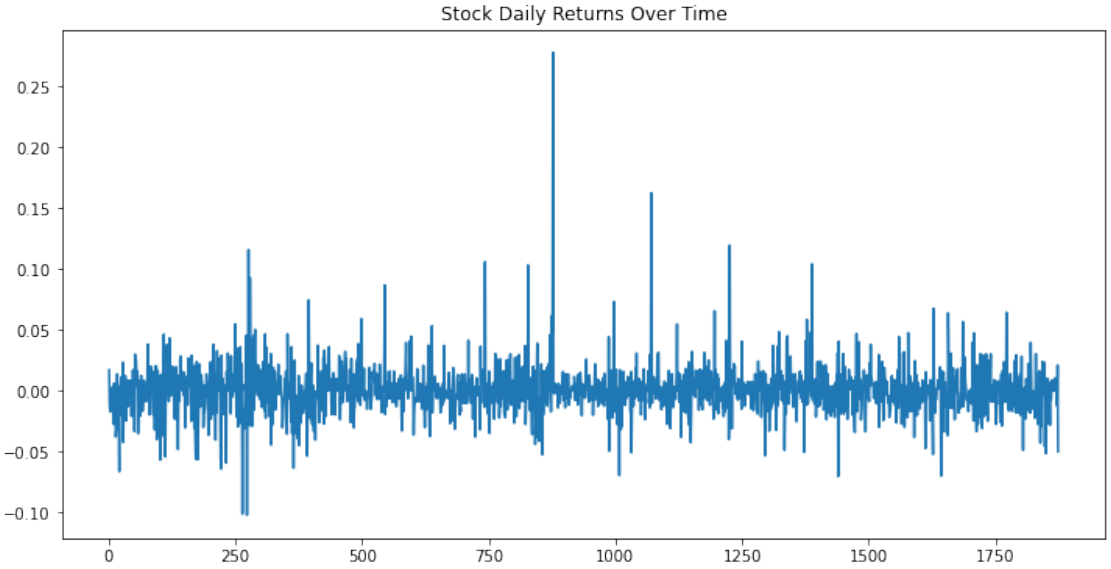
Question-2

What was the date with the highest volatility (difference between min and max)

[ ]:
```
# Calculating the daily returns of from the stock market
Ndata['Last'] = Ndata['Last'].astype(float)
Ndata['daily_returns']=(Ndata['Last'].pct_change())
```

[ ]:
```
# plotting the daily return which was achieved above
import matplotlib.pyplot as plt
data.dropna(inplace=True)
fig,ax=plt.subplots(figsize=(12,6))
plt.plot(Ndata['daily_returns'], label = 'Daily Returns')

plt.title('Stock Daily Returns Over Time')
plt.show()
```



[ ]:
```
# Now, we will vaculate the volitality of data per day
```

[ ]:
```
import math
daily_volatility_pfe = Ndata['daily_returns'].std()
print('Daily volatility:')
print((daily_volatility_pfe))
```

```
Daily volatility:
0.02044928330879462
```

[ ]:
```
Ndata.head()
```

[ ]:
```
         Date  Last  Open  High   Low      Vol.  Var. % New_column  \
1  04.09.2021  3.56  3.52  3.57  3.52  666030.0    1.14   Positive
2  04.08.2021  3.52  3.45  3.54  3.45  882540.0    1.59   Positive
3  04.07.2021  3.46  3.43  3.49  3.43  508330.0    0.87   Positive
4  04.06.2021  3.43  3.44  3.49  3.43  809460.0    0.59   Positive
5  04.01.2021  3.41  3.36  3.42  3.36  416710.0    1.19   Positive

   daily_returns
1       0.017143
2      -0.011236
3      -0.017045
4      -0.008671
5      -0.005831
```

[ ]:
```
# only year with heighest daily volitality
Ndata.nlargest(1, ['daily_returns'])
```

[ ]:
```
           Date  Last  Open  High   Low      Vol.  Var. % New_column  \
877  31.10.2017  5.06  5.09  5.10  5.03  466580.0    -0.2   Negative

     daily_returns
877       0.277778
```

[ ]:
```
# finding the exact date for the heighest daily volitality
Ndata.nlargest(1, ['daily_returns']).Date
```

[ ]:
```
877    31.10.2017
Name: Date, dtype: object
```

[ ]:

[ ]:

Question-3

What was the best month (30 days) during the 10 years to win with this title This means if I had to keep my stock only 30 days when was the best 30 days?

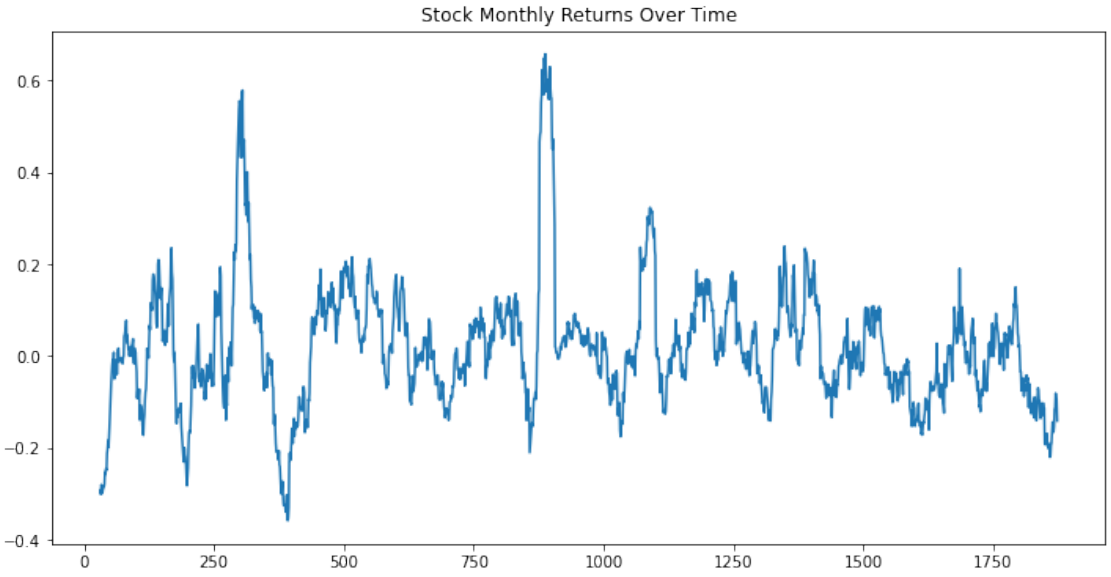
[ ]:
```
# To find the best month, I will use monthly volitality returns
```

[ ]:
```
# pct_change(30) means it will calculate the difference btw 30 rows
Ndata['monthly_returns']=(Ndata['Last'].pct_change(30))
```

[ ]:
```
# plotting the daily return which was achieved above
import matplotlib.pyplot as plt
data.dropna(inplace=True)
```

```
fig,ax=plt.subplots(figsize=(12,6))
plt.plot(Ndata['monthly_returns'], label = 'Monthly Returns')

plt.title('Stock Monthly Returns Over Time')
plt.show()
```



[ ]:
```
# only year with heighest daily volitality
Ndata.nlargest(1, ['monthly_returns'])
```

[ ]:
```
           Date  Last  Open  High   Low      Vol.  Var. % New_column  \
888  16.10.2017  5.04  5.06  5.06  5.00  388010.0     0.0   Positive

     daily_returns  monthly_returns
888       0.005988         0.657895
```

[ ]:
```
Ndata.nlargest(1, ['monthly_returns']).Date
```

[ ]:
```
888    16.10.2017
Name: Date, dtype: object
```

[ ]:

[ ]:

Question-4

What was the duration of the longest sequence of ascent of the title in the closings?

[ ]:
```
Ndata.head()
```

[ ]:
```
           Date  Last  Open  High   Low      Vol.  Var. % New_column  \
31   24.02.2021  2.52  2.46  2.52  2.45  475090.0    2.23   Positive
32   23.02.2021  2.46  2.48  2.52  2.43  469140.0   -1.00   Negative
33   22.02.2021  2.49  2.45  2.49  2.41  390810.0    1.43   Positive
34   19.02.2021  2.45  2.41  2.45  2.38  533430.0    2.72   Positive
35   18.02.2021  2.39  2.40  2.41  2.37  297330.0    0.00   Positive

    daily_returns  monthly_returns
31       0.003984        -0.292135
32      -0.023810        -0.301136
33       0.012195        -0.280347
34      -0.016064        -0.285714
35      -0.024490        -0.299120
```

[ ]:
```
Ndata = Ndata.sort_index(ascending=False)
```

[ ]:
```
Ndata.head()
```

[ ]:
```
             Date  Last  Open  High   Low       Vol.  Var. % New_column  \
1874  12.06.2013  5.53  5.58  5.59  5.51  4290000.0   -0.18   Negative
1873  12.09.2013  5.82  5.53  5.82  5.52  5800000.0    5.24   Positive
1872  12.10.2013  5.70  5.74  5.79  5.70  1430000.0   -2.06   Negative
1871  12.11.2013  5.77  5.69  5.79  5.67  1370000.0    1.23   Positive
1870  12.12.2013  5.74  5.75  5.76  5.71   725190.0   -0.52   Negative

      daily_returns  monthly_returns
1874      -0.049828        -0.141304
1873       0.021053        -0.096273
1872      -0.012132        -0.093800
1871       0.005226        -0.081210
1870       0.000000        -0.112828
```

[ ]:
```
# applying get_value() function
starting_date = Ndata.iloc[0, 0]


count = 0
max_count = 0
for i in range(len(Ndata)-1):
    if Ndata.iloc[i, 8] - Ndata.iloc[i+1, 8] <=0:
        count+=1
    else:
        if count > max_count:
            ending_date = Ndata.iloc[i, 0]
            starting_date1 = starting_date
```

```
            max_count = count
            count = 0
        else:
            count = 0
            starting_date = Ndata.iloc[i, 0]
```

[ ]:
```
starting_date1
```

[ ]:
```
'08.05.2016'
```

[ ]:
```
ending_date
```

[ ]:
```
'16.08.2016'
```

[ ]:

[ ]:

Question-5

What was the date that saw the greatest turmoil in the market, ie large volumes with important variations? You can choose for example volume * (max-min) to get a measure of turbulence).

[ ]:
```
# I will use facebook prophet to detect the abrupt changes in market.
```

[ ]:
```
# I will use only two columns of the data, and i
# will remove all others becuase for facebook model, we need just two columns
```

[ ]:
```
Ndata.head()
```

[ ]:

	Date	Last	Open	High	Low	Vol.	Var. %	New_column	daily_returns	monthly_returns
1874	12.06.2013	5.53	5.58	5.59	5.51	4290000.0	-0.18	Negative	-0.049828	-0.141304
1873	12.09.2013	5.82	5.53	5.82	5.52	5800000.0	5.24	Positive	0.021053	-0.096273
1872	12.10.2013	5.70	5.74	5.79	5.70	1430000.0	-2.06	Negative	-0.012132	-0.093800
1871	12.11.2013	5.77	5.69	5.79	5.67	1370000.0	1.23	Positive	0.005226	-0.081210
1870	12.12.2013	5.74	5.75	5.76	5.71	725190.0	-0.52	Negative	0.000000	-0.112828

[ ]:
```
# Ndata.drop("Last", axis=1, inplace=True)
# Ndata.drop("Open", axis=1, inplace=True)
# Ndata.drop("Low", axis=1, inplace=True)
# Ndata.drop("High", axis=1, inplace=True)
```

```
# Ndata.drop("Var. %", axis=1, inplace=True)
# Ndata.drop("New_column", axis=1, inplace=True)
# Ndata.drop("daily_returns", axis=1, inplace=True)
```

[ ]:
```
Ndata.drop("monthly_returns", axis=1, inplace=True)
Ndata.head()
```

[ ]:
```
            Date       Vol.
1874  12.06.2013  4290000.0
1873  12.09.2013  5800000.0
1872  12.10.2013  1430000.0
1871  12.11.2013  1370000.0
1870  12.12.2013   725190.0
```

[ ]:
```
# Now we will remane the columns names for facebook prophet
```

[ ]:
```
# python code to rename the columns
Ndata.rename(columns={'Date':'ds','Vol.':'y'},inplace=True)
```

[ ]:
```
# importing python time series packages
from prophet import Prophet

# initialiazing the model with 95% confidence interval
model = Prophet(interval_width= 0.95)

# train model
model.fit(Ndata)

# forecasting for future
future = model.make_future_dataframe(periods=8, freq='M')
```

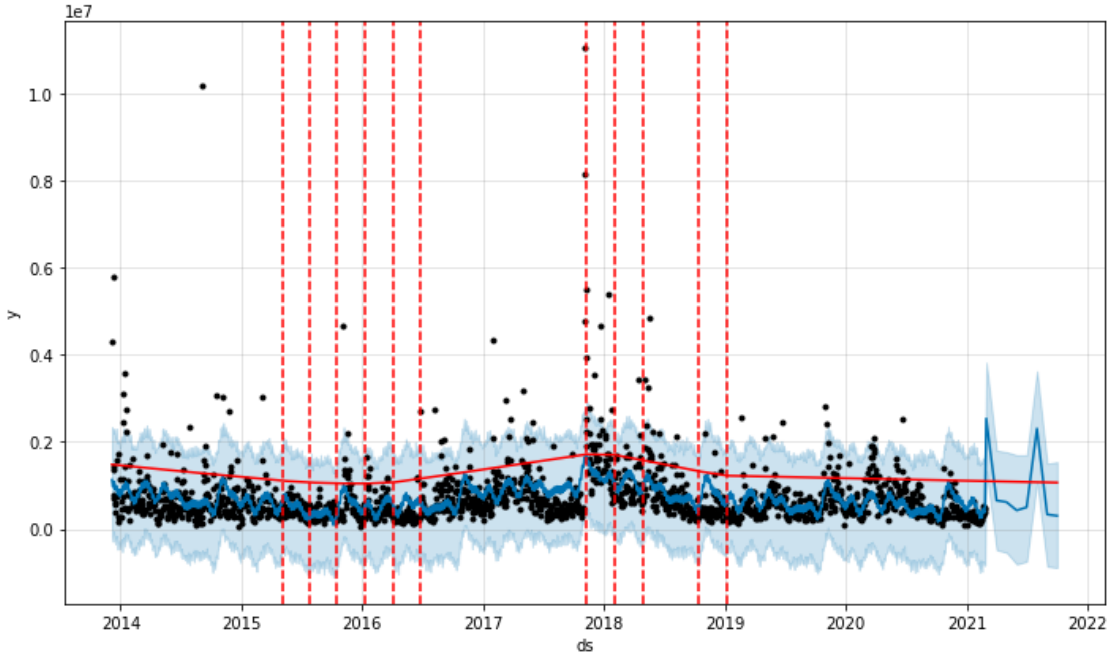
```
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpfzosji4v/3b8tn8ws.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpfzosji4v/wk2yw47w.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.7/dist-
packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=9218', 'data',
'file=/tmp/tmpfzosji4v/3b8tn8ws.json', 'init=/tmp/tmpfzosji4v/wk2yw47w.json',
'output',
'file=/tmp/tmpfzosji4v/prophet_modelkdlsmt0i/prophet_model-20221012151737.csv',
'method=optimize', 'algorithm=lbfgs', 'iter=10000']
15:17:37 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
15:17:37 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

[ ]:
```
# forecast predictions
forecast = model.predict(future)
```

[ ]:
```
# import the ploting module
from prophet.plot import add_changepoints_to_plot

# creating plot
fig = model.plot(forecast)

# completely automatic forecasting techniques
change_points = add_changepoints_to_plot(fig.gca(), model, forecast)
```



[ ]:
```
# So the vertical dotted red lines shows the abrupt changes in the stock market
 ↪( based on volume )
# over past 10 years
```

[ ]:
```

```