Analysis of COVID-19 Impact On (BEACH) Stocks and Price Prediction

Introduction:

We realize that the COVID-19 has flipped around the economy. Companies in the travel and entertainment sectors, so-called BEACH stocks- were the absolute hardest hit. These stocks have seen more than $332 billion in esteem vanish during the months of Feb-Mar 2020 source.

on the other hand, Time series forecasting, especially stock prediction, has been a hot topic for decades. Obviously, predicting the stock market is one of the most challenging things to do, and so many smart people and organizations are involved in this area. There are many variables that will affect the price: the earnings of a company, the supply and demand at that time, the trends of the overall economy, the political climate, pandemic and so on.

Objective:

Part of Udacity Data Scientist Nano degree program, I've selected Investment and Trading as Capstone Project, the objectives of this project are:

1- To quantify, compare, and visualize the impact of COVID-19 on the US stock market in the travel and entertainment sectors. The data stocks are considered from 2 Jan of(2019 and 2020) to May 31 (2019 and 2020), I've slected this period becuse the lockdown started in USA on March 19 2020 and As of 12 April, nearly 300 million people, or about 90 per cent of the population, are under some form of lockdown in the United States source

2- To build a stock price predictor that takes daily trading data over a certain date range as input for selected stock simples(in travel and entertainment sectors) , and outputs projected estimates for given query dates. The system only will predict the Adjusted Close price.

Data:

Data has been accessed from :
- Yahoo Finance using pandas_datareader library.
- USA COVID-19 from https://raw.githubusercontent.com

Table of Contents

Part (1) Analysis of COVID-19 Impact On (BEACH) Stocks).

Problem Statement:

As the Coronavirus (COVID-19) has spread from China to all parts of the world, stocks have fallen drastically, and volatility has greatly increased. In this part of the project I will look To quantify, compare, and visualize the impact of COVID-19 on a selected US stocks under (Booking, Entertainment, Airlines, Cruise and Hotels). The data of stocks are considered from 2 Jan of(2019 and 2020) to May 31 (2019 and 2020), I've slected this period becuse the lockdown started in USA on March 19 2020 and As of 12 April

Solution:

1. Data download and Exploration:

   First, I'll use raw.githubusercontent.com to get the USA COVID-19 confirmed cases

   - Get the data then filter the data by country (United States) using the below function

```
#Getting The Data From raw.githubusercontent.com - USA COVID-19 confirmed cases
for filename in ['time_series_covid19_confirmed_global.csv']:
    print(f'Downloading {filename}')
    url = f'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_seri
    myfile = requests.get(url)
    open(filename, 'wb').write(myfile.content)

confirmed_global_df = pd.read_csv('time_series_covid19_confirmed_global.csv')
confirmed_global_df["Country/Region"].replace({'US': 'United States'}, inplace=True)
```

```
Downloading time_series_covid19_confirmed_global.csv
```

```
country_main = 'United States'
confirmed_global_df = confirmed_global_df[confirmed_global_df['Country/Region'] == country_main].reset_index(drop=True)
confirmed_global_df
```

- Convert the data into string and group the data by date using the below function

```
def _convert_date_str(df):
    """
    convert covid19 date to string
    Params:
        df: stock to pull data
    Return:
        none
    """
    try:
        df.columns = list(df.columns[:4]) + [datetime.strptime(d, "%m/%d/%y").date().strftime("%Y-%m-%d") for d in df.c
    except:
        print('_convert_date_str failed with %y, try %Y')
        df.columns = list(df.columns[:4]) + [datetime.strptime(d, "%m/%d/%Y").date().strftime("%Y-%m-%d") for d in df.c

_convert_date_str(confirmed_global_df)
confirmed_global_df
```

	Province/State	Country/Region	Lat	Long	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	...	2020-10-06	2020-10-07	2020-10-08	2020-10-09	2020-10-10	2020-10-11	2020 10-1:
0	NaN	United States	40.0	-100.0	1	1	2	2	5	5	...	7499341	7549682	7605873	7663293	7717932	7762546	780419!

1 rows × 272 columns

```
# set fileds and group the data by date
df_confirmed = confirmed_global_df.melt(
    id_vars=['Province/State', 'Country/Region', 'Lat', 'Long'], value_vars=confirmed_global_df.columns[4:], var_name='
df_covid = df_confirmed.groupby(["Date", "Country/Region"])[['Date', 'Country/Region', 'ConfirmedCases']].sum().reset_i
df_covid.columns = ['Date', 'Country', 'Confirmed']
# explore the data
df_covid
```

- Filter the date by date from 2 Jan to May 31 2020, The COVID-19 data is simple, as we can see below. There are only 2 columns Country and Confirmed(this represent the daily number of confirmed cases in US Date.

```
# match the data to be same as start and end of stock dataframe
split_date = '2020-05-31'
df_covid = df_covid.loc[df_covid['Date'] <= split_date]
df_covid['Date']= pd.to_datetime(df_covid['Date'])
df_covid.sort_values('Date', inplace=True)
```

```
df_covid
```

	Date	Country	Confirmed
0	2020-01-22	United States	1
1	2020-01-23	United States	1
2	2020-01-24	United States	2
3	2020-01-25	United States	2
4	2020-01-26	United States	5
...	...	...	...
126	2020-05-27	United States	1706331
127	2020-05-28	United States	1729279
128	2020-05-29	United States	1753630
129	2020-05-30	United States	1777473
130	2020-05-31	United States	1796645

- As we can see below, the data contains 131 raw, the table also tells some ostatistical information about the COVID-19 conformed cases. For example, we can see the minimum ‘Confirmed’ is 1, and the maximum is 1796645. Also, I’ve calculated when the number of cases when has reached 25%, 50% and 70%. I’ll use these data to visualize the impact on the stock price.

```
: df_covid.describe()
```

	Confirmed
count	1.310000e+02
mean	5.057217e+05
std	6.164102e+05
min	1.000000e+00
25%	1.600000e+01
50%	1.051790e+05
75%	1.033262e+06
max	1.796645e+06

```
# get the date of max number of confiremd covid19 cases
df_covid[df_covid['Confirmed']==df_covid.max()['Confirmed']].head(1)
```

	Date	Country	Confirmed
130	2020-05-31	United States	1796645

```
# Get The date of first covid-19 case reported in USA
df_covid[df_covid['Confirmed']==1].head(1)
```

	Date	Country	Confirmed
0	2020-01-22	United States	1

```
# get the date when the cases when it reach 25% in USA reported cases, this will be used later in the analysis
perc_25 = np.percentile(df_covid.Confirmed, 25)
df_covid[df_covid['Confirmed']== perc_25].head(1)
```

	Date	Country	Confirmed
30	2020-02-21	United States	16

```
# get the date when the cases when reach 50% in USA reported cases, this will be used later in the analysis
perc_50 = np.percentile(df_covid.Confirmed, 50)
df_covid[df_covid['Confirmed']== perc_50].head(1)
```

Second, I will use yahoo finance (https://finance.yahoo.com/). There is a python package named finance, which can enable us to pull stock prices easily.

- Getting the Data from Yahoo Finance for 2019 and 2020 for Companies Chosen for the project:

```
In [14]: def download_stock_to_df(symbol, start, end):
             """
             Get current stocks data from yahoo fiance and save to dataframe

             Params:
                 symbol: stock to pull data
                 start: start date of pulled data
                 end: end date of pulled data

             Return:
                 dataframe of stock within specified date range
             """
             df_stock=yf.download(symbol,start,end,progress=False)
             df_stock.reset_index(level=0, inplace=True)
             return df_stock
```

- ♦ Booking Stock (Ticker: BKNG on the NASDAQ)- Booking

```python
# read booking stock for 2019
booking_2019 = download_stock_to_df('BKNG', '2019-01-02', '2019-05-31')
booking_2019 = set_date_df(booking_2019)
booking_2019.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2019-01-02	1691.250000	1736.770020	1690.839966	1721.699951	1721.699951	312600	NaN
1	2019-01-03	1704.650024	1712.680054	1657.979980	1663.119995	1663.119995	445200	-0.034024
2	2019-01-04	1686.119995	1737.079956	1678.619995	1717.550049	1717.550049	541000	0.032728
3	2019-01-07	1721.739990	1731.400024	1703.599976	1711.819946	1711.819946	334800	-0.003336
4	2019-01-08	1730.040039	1755.989990	1681.150024	1686.920044	1686.920044	652400	-0.014546

```python
# read booking stock for 2020
booking_2020 = download_stock_to_df('BKNG', '2020-01-02', '2020-05-31')
booking_2020 = set_date_df(booking_2020)
booking_2020.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2020-01-02	2068.399902	2077.409912	2053.219971	2074.580078	2074.580078	329000	NaN
1	2020-01-03	2042.469971	2067.689941	2035.000000	2065.479980	2065.479980	294000	-0.004386
2	2020-01-06	2050.000000	2059.790039	2035.000000	2047.400024	2047.400024	384000	-0.008753
3	2020-01-07	2047.390015	2072.830078	2038.540039	2068.050049	2068.050049	345300	0.010086
4	2020-01-08	2066.840088	2088.659912	2057.139893	2062.899902	2062.899902	325800	-0.002490

- Expedia Stock (Ticker: EXPE on the NASDAQ)-Booking


```python
# read booking stock for 2019
booking_2019 = download_stock_to_df('BKNG', '2019-01-02', '2019-05-31')
booking_2019 = set_date_df(booking_2019)
booking_2019.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2019-01-02	1691.250000	1736.770020	1690.839966	1721.699951	1721.699951	312600	NaN
1	2019-01-03	1704.650024	1712.680054	1657.979980	1663.119995	1663.119995	445200	-0.034024
2	2019-01-04	1686.119995	1737.079956	1678.619995	1717.550049	1717.550049	541000	0.032728
3	2019-01-07	1721.739990	1731.400024	1703.599976	1711.819946	1711.819946	334800	-0.003336
4	2019-01-08	1730.040039	1755.989990	1681.150024	1686.920044	1686.920044	652400	-0.014546

```python
# read booking stock for 2020
booking_2020 = download_stock_to_df('BKNG', '2020-01-02', '2020-05-31')
booking_2020 = set_date_df(booking_2020)
booking_2020.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2020-01-02	2068.399902	2077.409912	2053.219971	2074.580078	2074.580078	329000	NaN
1	2020-01-03	2042.469971	2067.689941	2035.000000	2065.479980	2065.479980	294000	-0.004386
2	2020-01-06	2050.000000	2059.790039	2035.000000	2047.400024	2047.400024	384000	-0.008753
3	2020-01-07	2047.390015	2072.830078	2038.540039	2068.050049	2068.050049	345300	0.010086
4	2020-01-08	2066.840088	2088.659912	2057.139893	2062.899902	2062.899902	325800	-0.002490

- Netflix Stock (Ticker: NFLX on the NASDAQ)-Media/Entertainment
  I've included Netflix because it is kind of a media entertainment company. So, I wanted to compare, its results with the hardest-hit companies as people were social distancing due to sudden rise of COVID-19.

```
]: #I've included Netflix because it is kind of a meida entertainment company. So, I wanted to compare
   # its results with the hardest-hit companies as people were social distancing due to sudden rise of COVID-19.
   nflx_2019 = download_stock_to_df('NFLX', '2019-01-02', '2019-05-31')
   nflx_2019 = set_date_df(nflx_2019)
   nflx_2019.head()
```

]:

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2019-01-02	259.279999	269.750000	256.579987	267.660004	267.660004	11679500	NaN
1	2019-01-03	270.200012	275.790009	264.429993	271.200012	271.200012	14969600	0.013226
2	2019-01-04	281.880005	297.799988	278.540009	297.570007	297.570007	19330100	0.097234
3	2019-01-07	302.100006	316.799988	301.649994	315.339996	315.339996	18620100	0.059717
4	2019-01-08	319.980011	320.589996	308.010010	320.269989	320.269989	15359200	0.015634

```
]: # read netflix stock for  2020
   nflx_2020 = download_stock_to_df('NFLX', '2020-01-02', '2020-05-31')
   nflx_2020 = set_date_df(nflx_2020)
   nflx_2020.head()
```

]:

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2020-01-02	326.100006	329.980011	324.779999	329.809998	329.809998	4485800	NaN
1	2020-01-03	326.779999	329.859985	325.529999	325.899994	325.899994	3806900	-0.011855
2	2020-01-06	323.119995	336.359985	321.200012	335.829987	335.829987	5663100	0.030469
3	2020-01-07	336.470001	336.700012	330.299988	330.750000	330.750000	4703200	-0.015127
4	2020-01-08	331.489990	342.700012	331.049988	339.260010	339.260010	7104500	0.025729

- Six Flags (Ticker: SIX on the NASDAQ)-Entertainment & Live Events

```
: #six flags 2019
  six_2019 = download_stock_to_df('SIX', '2019-01-02', '2019-05-31')
  six_2019 = set_date_df(six_2019)
  six_2019.head()
```

:

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2019-01-02	54.880001	56.310001	54.400002	56.029999	52.150455	768100	NaN
1	2019-01-03	55.419998	56.770000	54.680000	55.930000	52.057381	455500	-0.001785
2	2019-01-04	56.939999	57.939999	56.540001	57.799999	53.797901	795700	0.033435
3	2019-01-07	58.200001	59.759998	57.450001	59.060001	54.970661	1116700	0.021799
4	2019-01-08	59.849998	60.340000	59.080002	60.220001	56.050343	702700	0.019641

```
: # Six Flags 2020
  six_2020 = download_stock_to_df('SIX', '2020-01-02', '2020-05-31')
  six_2020 = set_date_df(six_2020)
  six_2020.head()
```

:

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2020-01-02	44.369999	45.060001	43.799999	45.060001	44.611912	1773900	NaN
1	2020-01-03	44.709999	45.330002	44.419998	45.259998	44.809917	799000	0.004438
2	2020-01-06	44.380001	44.990002	43.869999	44.369999	43.928768	1264500	-0.019664
3	2020-01-07	44.450001	45.279999	44.419998	44.419998	43.978271	1233400	0.001127
4	2020-01-08	44.230000	44.810001	43.250000	43.560001	43.126827	1875900	-0.019361

- United Airlines Stock (Ticker: UAL on the NASDAQ)-Airlines

```python
# read united stock for 2019
united_2019 = download_stock_to_df('UAL', '2019-01-02', '2019-05-31')
united_2019 = set_date_df(united_2019)
united_2019.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2019-01-02	81.690002	84.290001	81.410004	84.180000	84.180000	2973400	NaN
1	2019-01-03	83.260002	83.260002	78.379997	80.000000	80.000000	6426200	-0.049656
2	2019-01-04	80.879997	83.949997	80.769997	82.680000	82.680000	3808300	0.033500
3	2019-01-07	82.570000	83.919998	81.449997	83.230003	83.230003	2653000	0.006652
4	2019-01-08	83.300003	84.620003	81.889999	82.379997	82.379997	3910000	-0.010213

```python
# read united stock for  2020
united_2020 = download_stock_to_df('UAL', '2020-01-02', '2020-05-31')
united_2020 = set_date_df(united_2020)
united_2020.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2020-01-02	89.570000	90.570000	89.110001	89.739998	89.739998	2769800	NaN
1	2020-01-03	86.800003	88.160004	86.260002	87.900002	87.900002	3562900	-0.020504
2	2020-01-06	86.720001	88.070000	86.650002	87.699997	87.699997	2652700	-0.002275
3	2020-01-07	87.410004	88.160004	86.739998	86.769997	86.769997	2581300	-0.010604
4	2020-01-08	86.900002	88.449997	86.300003	87.300003	87.300003	4152500	0.006108

- Royal Caribbean Cruises (Ticker: RCL on the NASDAQ)-Cruise & Casino

```python
# read royal stock for 2019
royal_2019 = download_stock_to_df('RCL', '2019-01-02', '2019-05-31')
royal_2019 = set_date_df(royal_2019)
royal_2019.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2019-01-02	96.400002	98.510002	95.949997	97.790001	94.466072	1352200	NaN
1	2019-01-03	96.699997	97.309998	91.879997	92.550003	89.404190	2807800	-0.053584
2	2019-01-04	94.800003	99.919998	94.500000	98.760002	95.403107	2261400	0.067099
3	2019-01-07	99.669998	102.709999	98.809998	101.830002	98.368744	2150300	0.031085
4	2019-01-08	103.739998	104.809998	102.529999	103.930000	100.397377	1882300	0.020623

```python
# read royal stock for 2020
royal_2020 = download_stock_to_df('RCL', '2020-01-02', '2020-05-31')
royal_2020 = set_date_df(royal_2020)
royal_2020.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2020-01-02	134.550003	134.800003	133.470001	134.649994	133.326569	1014400	NaN
1	2020-01-03	132.789993	133.619995	131.809998	133.490005	132.177979	1229400	-0.008615
2	2020-01-06	131.589996	131.869995	130.389999	131.630005	130.336258	1518000	-0.013934
3	2020-01-07	131.080002	132.100006	130.440002	130.440002	129.157959	1211400	-0.009041
4	2020-01-08	130.350006	132.710007	130.020004	132.220001	130.920456	1429600	0.013646

- Marriott International stock (Ticker: MAR on the NASDAQ)-Hotels and Resorts

```python
# read marriott stock for 2019
marriott_2019 = download_stock_to_df('MAR', '2019-01-02', '2019-05-31')
marriott_2019 = set_date_df(marriott_2019)
marriott_2019.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2019-01-02	106.720001	108.589996	105.290001	107.459999	105.518723	1802100	NaN
1	2019-01-03	106.349998	107.000000	101.570000	101.739998	99.902061	3344300	-0.053229
2	2019-01-04	103.139999	107.919998	103.139999	107.809998	105.862404	3045300	0.059662
3	2019-01-07	107.410004	108.879997	105.949997	108.010002	106.058792	1529000	0.001855
4	2019-01-08	108.860001	110.070000	107.820000	109.760002	107.777176	1631100	0.016202

```python
# read marriott stock for 2020
marriott_2020 = download_stock_to_df('MAR', '2020-01-02', '2020-05-31')
marriott_2020 = set_date_df(marriott_2020)
marriott_2020.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return
0	2020-01-02	151.500000	152.600006	151.029999	151.490005	150.884995	1905600	NaN
1	2020-01-03	149.190002	150.020004	148.740005	149.270004	148.673859	2116900	-0.014654
2	2020-01-06	147.899994	147.990005	146.210007	147.339996	146.751572	2178200	-0.012930
3	2020-01-07	146.740005	147.220001	144.679993	144.940002	144.361160	2073200	-0.016289
4	2020-01-08	145.130005	147.619995	144.720001	146.729996	146.143997	1412800	0.012350

- As we can see from the tables above, there are multiple variables in the dataset – date, open, high, low, last, close, volum, the columns Open and Close represent the starting and final price at which the stock is traded on a particular day. High, Low and Last represent the maximum, minimum, and last price of the share for the day.

2. Data Analysis and Visualization

    - Compare the Adj price for the selected stocks in 2019 and 2020 during the selected period

```
print("Min Adj price for Booking in 2019 vs 2020:" , booking_2019['Adj Close'].min(), ':', booking_2020['Adj Close'].mi
decline = ((booking_2019['Adj Close'].mean() - booking_2020['Adj Close'].mean())/booking_2019['Adj Close'].mean())*100
print("The Decline in the Adj price for Booking in 2019 vs 2020:" , decline)
```

```
Min Adj price for Booking in 2019 vs 2020: 1649.489990234375 : 1152.239990234375
The Decline in the Adj price for Booking in 2019 vs 2020: 8.785021105507077
```

```
print("Min Adj price for Expedia in 2019 vs 2020:" , Expedia_2019['Adj Close'].min(), ':', Expedia_2020['Adj Close'].mi
decline = ((Expedia_2019['Adj Close'].mean() - Expedia_2020['Adj Close'].mean())/Expedia_2019['Adj Close'].mean())*100
print("The Decline in the Adj price for Expedia in 2019 vs 2020:" , decline)
```

```
Min Adj price for Expedia in 2019 vs 2020: 106.88871002197266 : 45.650001525878906
The Decline in the Adj price for Expedia in 2019 vs 2020: 30.048218666261768
```

```
print("Min Adj price for NetFlix in 2019 vs 2020:" , nflx_2019['Adj Close'].min() ,':', nflx_2020['Adj Close'].min())
decline = ((nflx_2019['Adj Close'].mean() - nflx_2020['Adj Close'].mean())/nflx_2019['Adj Close'].mean())*100
print("The Decline in the Adj price for Netflix in 2019 vs 2020:" , decline)
```

```
Min Adj price for NetFlix in 2019 vs 2020: 267.6600036621094 : 298.8399963378906
The Decline in the Adj price for Netflix in 2019 vs 2020: -7.390341922057356
```

```
print("Min Adj price for Six Flags in 2019 vs 2020:" , six_2019['Adj Close'].min() ,':', six_2020['Adj Close'].min())
decline = ((six_2019['Adj Close'].mean() - six_2020['Adj Close'].mean())/six_2019['Adj Close'].mean())*100
print("The Decline in the Adj price for Six Flags in 2019 vs 2020:" , decline)
```

```
Min Adj price for Six Flags in 2019 vs 2020: 44.07891845703125 : 10.359999656677246
The Decline in the Adj price for Six Flags in 2019 vs 2020: 51.24767204957254
```

```
print("Min Adj price for United Air in 2019 vs 2020:" , united_2019['Adj Close'].min() ,':', united_2020['Adj Close'].m
decline = ((united_2019['Adj Close'].mean() - united_2020['Adj Close'].mean())/united_2019['Adj Close'].mean())*100
print("The Decline in the Adj price for United Air in 2019 vs 2020:" , decline)
```

```
Min Adj price for United Air in 2019 vs 2020: 77.48999786376953 : 19.920000076293945
The Decline in the Adj price for United Air in 2019 vs 2020: 40.60265247023829
```

```
print("Min Adj price for Royal in 2019 vs 2020:" ,royal_2019['Adj Close'].min() ,':', royal_2020['Adj Close'].min())
decline = ((royal_2019['Adj Close'].mean() - royal_2020['Adj Close'].mean())/royal_2019['Adj Close'].mean())*100
print("The Decline in the Adj price for Royal in 2019 vs 2020:" , decline)
```

```
Min Adj price for Royal in 2019 vs 2020: 89.40419006347656 : 22.329999923706055
The Decline in the Adj price for Royal in 2019 vs 2020: 38.52690967072977
```

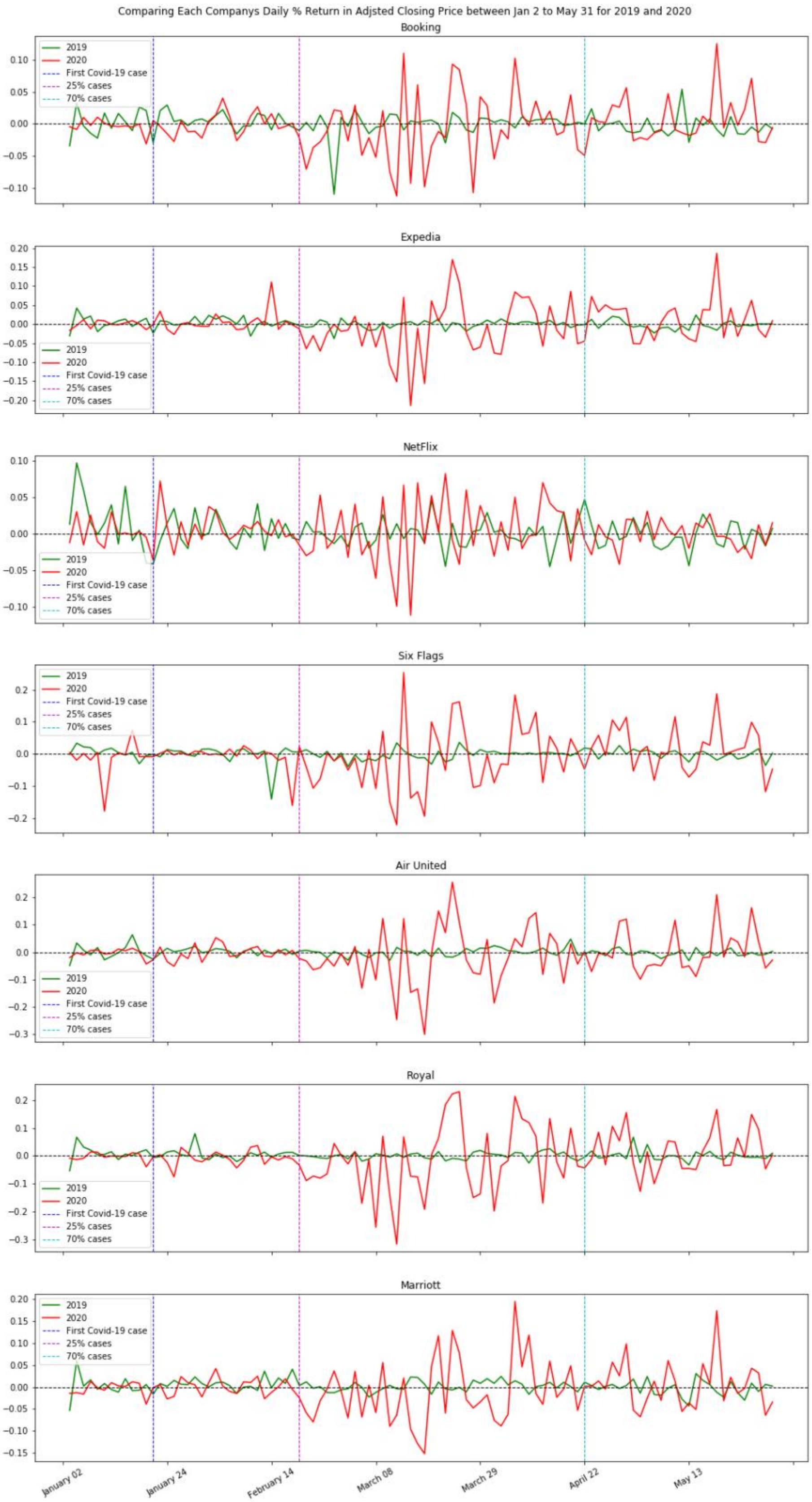
```
print("Min Adj price for Marriott in 2019 vs 2020:" , marriott_2019['Adj Close'].min() ,':', marriott_2020['Adj Close']
decline = ((marriott_2019['Adj Close'].mean() - marriott_2020['Adj Close'].mean())/marriott_2019['Adj Close'].mean())*1
print("The Decline in the Adj price for Marriott in 2019 vs 2020:" , decline)
```

```
Min Adj price for Marriott in 2019 vs 2020: 99.90206146240234 : 59.08000183105469
The Decline in the Adj price for Marriott in 2019 vs 2020: 11.54468664215102
```

As we can see from the above exploration, the minim Adjusted price in 2020 declined for all stocks.

a) Booking declined 8.78%
b) Expedia declined 30.04%
c) Netflix inclined 7.39%
d) Six Flags declined 51.24%
e) United Air declined 40.60%
f) Royal declined 38.52
g) Marriott declined 11.54%

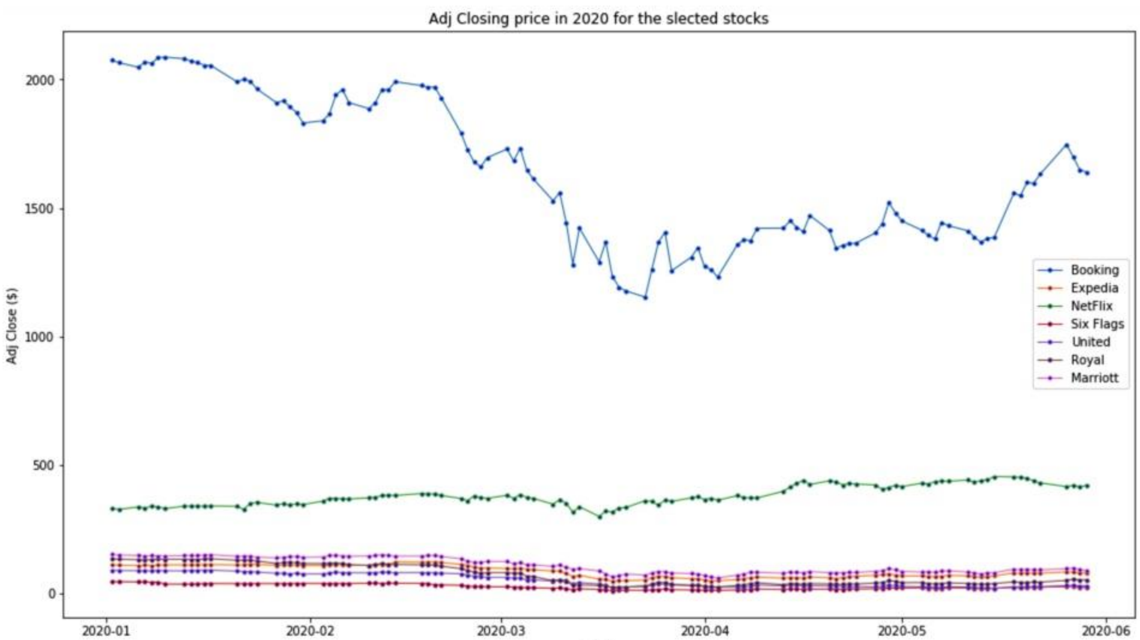
- Comparing each company percent change between 2019 and 202in the below figure:



Comparing Each Companys Daily % Return in Adjsted Closing Price between Jan 2 to May 31 for 2019 and 2020
Booking
Expedia
NetFlix
Six Flags
Air United
Royal
Marriott
2019
2020
First Covid-19 case
25% cases
70% cases
January 02
January 24
February 14
March 08
March 29
April 22
May 13

As we can see from the line graph above of the Daily % Return in Adjusted Closing Price of each stock, the decrease in the return occurred between Feb 21 where the COVID-19 confirmed cases started to increase by 25% and April 22 when the confirmed cases reached 70%.

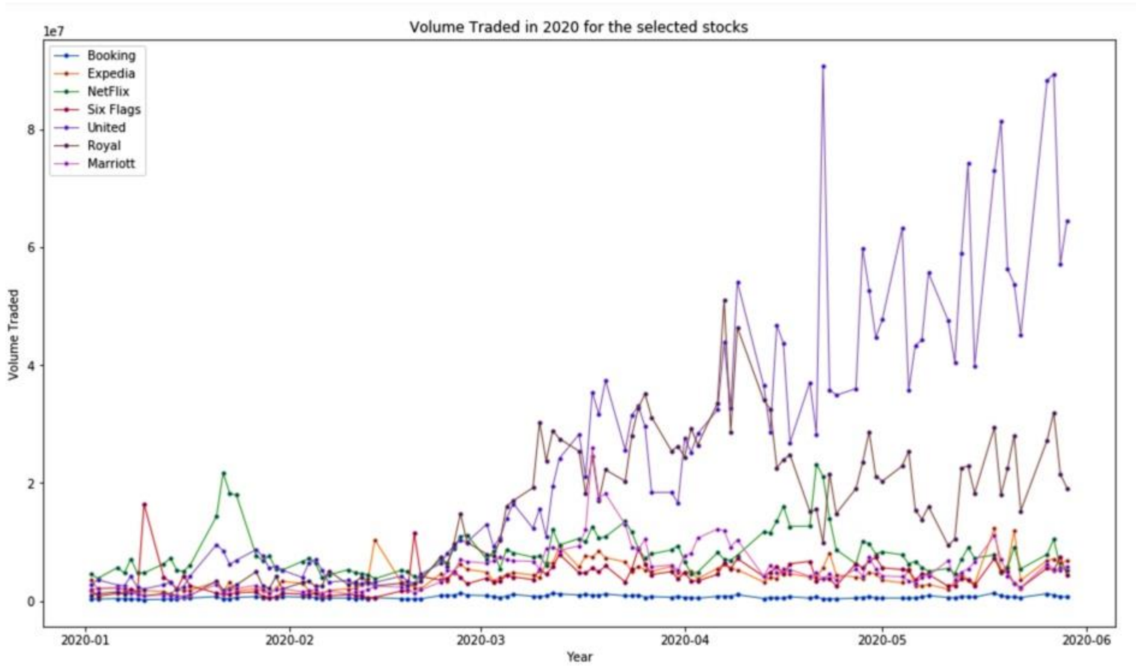
- Visualizing the closing price for the selected stocks on 2020, where the profit or loss calculation is usually determined by the closing price of a stock for the day, hence we will consider the closing price as the target variable. Let's plot the target variable to understand how it's shaping up in our data:



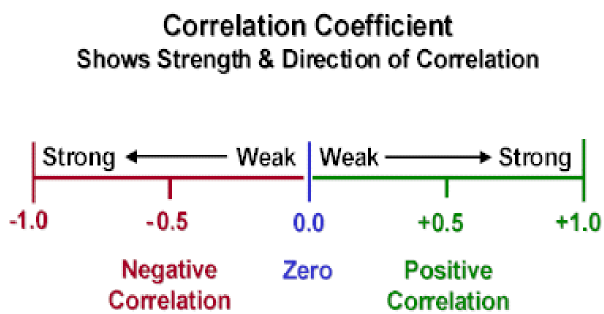
As we can see from the line graph the lowest adjusted Closing price in 2020 for the selected stocks, is during March

```
Min Adj price for Bookin in 2019 vs 2020: 1649.489990234375 : 1152.239990234375
Min Adj price for NetFlix in 2019 vs 2020: 267.6600036621094 : 298.8399963378906
Min Adj price for Six Flags in 2019 vs 2020: 44.07891845703125 : 10.35999965667724
Min Adj price for United in 2019 vs 2020: 77.48999786376953 : 19.920000076293945
Min Adj price for Royal in 2019 vs 2020: 89.40419006347656 : 22.329999923706055
Min Adj price for Marriott in 2019 vs 2020: 99.90206146240234 : 59.08000183105469
```

- Visualizing the volume traded for the selected stocks on 2020, where Stock trading volume refers to the amount of shares traded in a particular stock over a period of time. Often volume is measured in terms of shares traded per day. Remember that the number of shares bought and sold. If there's a higher volume of trading in a particular stock, that naturally means that investors are interested in buying or selling it. If volume and price are on the rise, it means investors are betting the company will do well. If volume is up but price is down, it means more investors are looking to sell.



- Evaluate the correlation coefficient of COVID-19 with the selected BEACH stock market, this matrix represents the relation:

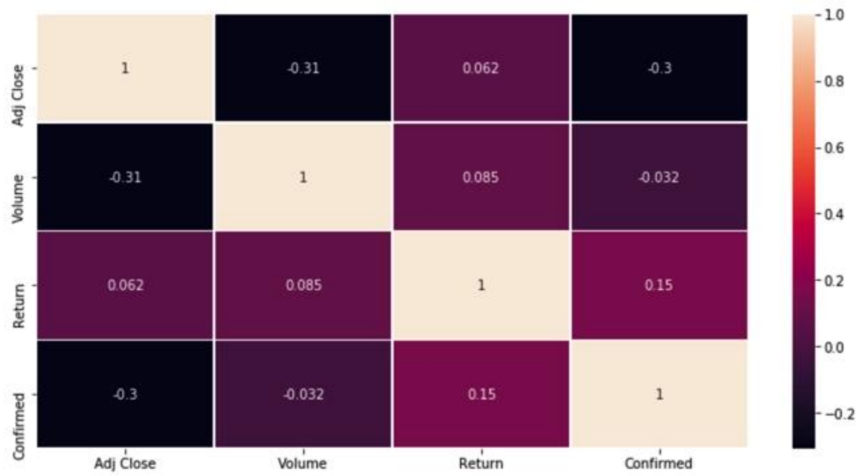


We calculated the correlation using the below function:

```
def Correl_Matrix(stock_2020):
    # Correlation Matrix
    stock_covid = pd.merge(stock_2020, df_covid, how='inner', on = 'Date')
# keep only used column
    stock_covid.drop(columns=['Open', 'High', 'Low', 'Close'], inplace=True)
    # Look  at Correlation between variables
    df_corr = stock_covid.corr()
# Get review_scores_rating corr
    df_corr['Adj Close'].sort_values(ascending=False)
    corr = (df_corr)
    fig, ax = plt.subplots(figsize=(12, 6))
    sns.heatmap(corr, xticklabels=corr.columns.values,
            yticklabels=corr.columns.values, linewidths=.5, ax=ax, annot=True)
```

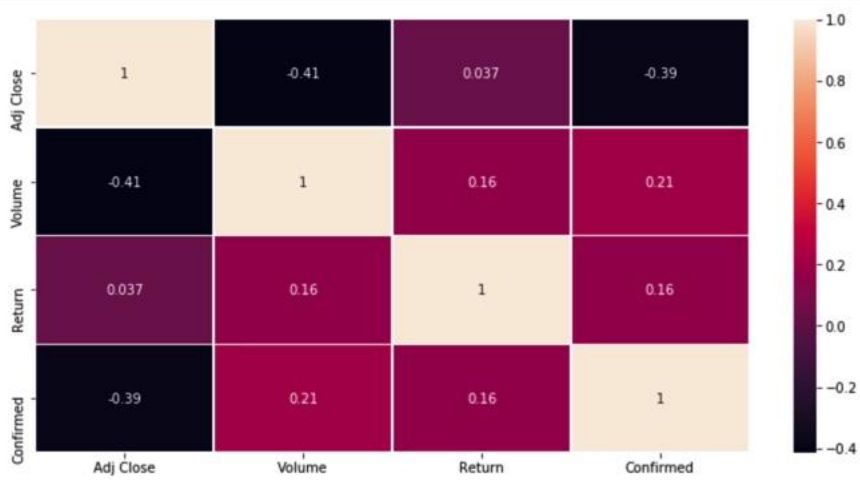
- ♦ Booking stock has negative correlation (-0.3) between the confirmed cases and the adjusted price

```
Correl_Matrix(booking_2020)
```



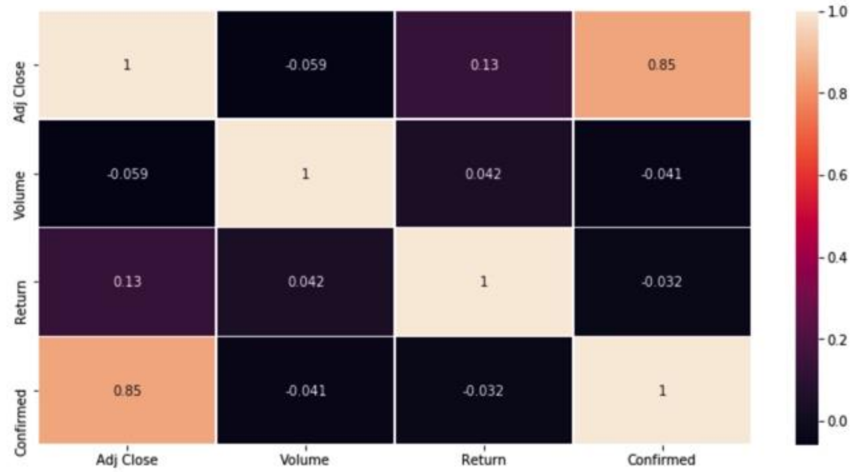
- Expedia stock has negative correlation (-0.39) between the confirmed cases and the adjusted price

```
Correl_Matrix(Expedia_2020)
```



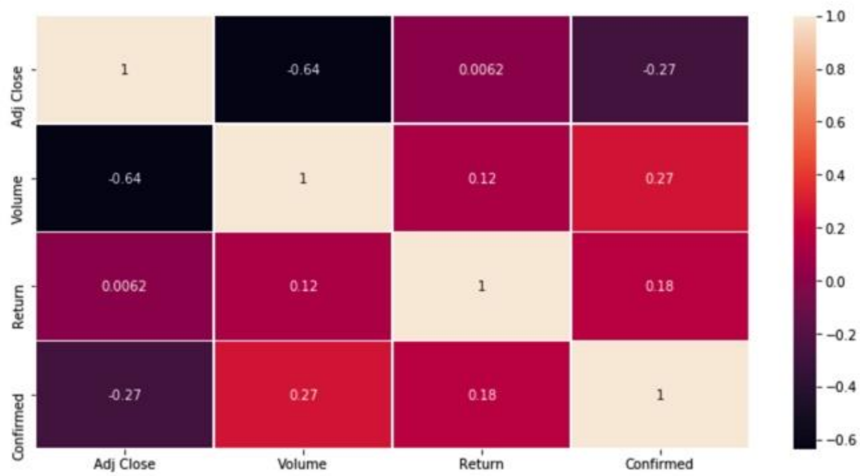
- Netflix stock has positive correlation (0.85) between the confirmed cases and the adjusted price

```
Correl_Matrix(nflx_2020)
```



- Six Flags stock has negative correlation (-0.27 between the confirmed cases and the adjusted price

```
Correl_Matrix(six_2020)
```

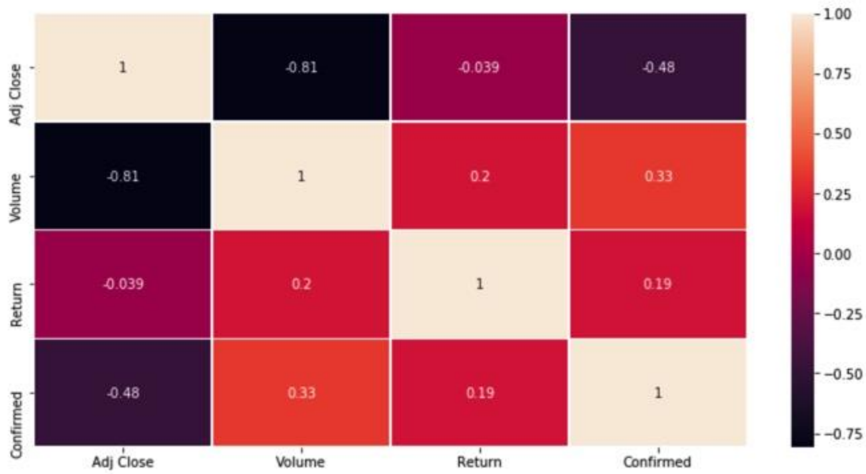


- United Air stock has negative correlation (-0.66) between the confirmed cases and the adjusted price

```
Correl_Matrix(united_2020)
```

- Royal stock has negative correlation (-0.48) between the confirmed cases and the adjusted price

```
Correl_Matrix(royal_2020)
```



- Marriott stock has negative correlation (-0.49) between the confirmed cases and the adjusted price

```
Correl_Matrix(marriott_2020)
```

As we can see from the above, Evaluate the correlation coefficient of COVID-19 with the selected BEACH stock market. The obtained correlation coefficient presents the intensity and severity of its impact on the selected stocks, except Netflix has a positive since it is home entertainment during the lockdown conditions.

3. Part (1)-Conclusion

This combination of volatility and correlation matrix comparisons was an effective way of analyzing how COVID-19 impacted the US stock market in the travel and entertainment sectors during the selected period from 2 Jan of (2019 and 2020) to May 31 (2019 and 2020), where the lockdown started in USA on March 19 2020 and as of 12 April.

Part (2) Price prediction using Long Short Term Memory (LSTM).

Problem Statement:

In this part of the project I will be working on the “Stock Predictor” problem. First I will try to build a machine learning/deep learning model that can predict the future stock “Adj Close” price, based on historical data. Training and evaluation will be run to test the model performance. Second, I will build an interactive web application that can enable users to pick a -travel and entertainment sectors-stock symbols and time ranges for model running.

Solution:

- Data download and Exploration

  There are multiple ways that we can get stock prices. For the purpose of this project, I will use yahoo finance (https://finance.yahoo.com/). There is a python package named finance, which can enable us to pull stock prices easily.

```
In [14]: def download_stock_to_df(symbol, start, end):
             """
             Get current stocks data from yahoo fiance and save to dataframe

             Params:
                 symbol: stock to pull data
                 start: start date of pulled data
                 end: end date of pulled data

             Return:
                 dataframe of stock within specified date range
             """
             df_stock=yf.download(symbol,start,end,progress=False)
             df_stock.reset_index(level=0, inplace=True)
             return df_stock
```

  The stock data from yahoo finance is fairly simple, as we can see below. There are mainly 7 columns including the Date. In my project, I plan to use "Adj Close" as our target (the prediction price).

Out[18]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1999-12-31	282.375	296.2500	280.875	284.250	284.250	410100
1	2000-01-03	315.000	318.0000	288.000	307.500	307.500	847700
2	2000-01-04	308.250	343.5000	307.500	324.000	324.000	972700
3	2000-01-05	331.500	385.3125	327.750	359.625	359.625	1605400
4	2000-01-06	368.250	384.7500	326.250	345.000	345.000	1175200

  As we can see from the above table, the DataFrame contains 7 columns. For this project, our main goal is to predict the Adj Close price. the 'describe' method of pandas to get more information about the data.

```
: #for more data understanding let's look at the  data types
  booking_hist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5223 entries, 0 to 5222
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       5223 non-null   datetime64[ns]
 1   Open       5223 non-null   float64
 2   High       5223 non-null   float64
 3   Low        5223 non-null   float64
 4   Close      5223 non-null   float64
 5   Adj Close  5223 non-null   float64
 6   Volume     5223 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 285.8 KB
```

```
: booking_hist.describe()
```

:

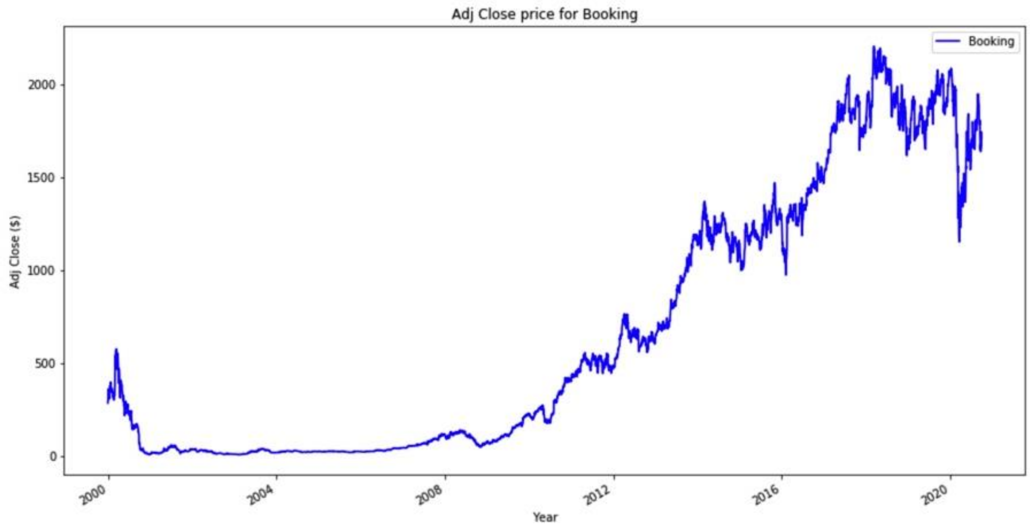
	Open	High	Low	Close	Adj Close	Volume
count	5223.000000	5223.000000	5223.000000	5223.000000	5223.000000	5.223000e+03
mean	656.552976	663.909173	648.807562	656.408153	656.408153	8.127243e+05
std	704.179392	710.527483	697.726180	704.099879	704.099879	7.860427e+05
min	6.600000	7.140000	6.300000	6.600000	6.600000	3.690000e+04
25%	31.930000	32.674999	30.920000	31.799999	31.799999	3.887500e+05
50%	324.829987	334.125000	312.750000	323.799988	323.799988	6.113000e+05
75%	1248.895020	1262.000000	1236.340027	1248.679993	1248.679993	9.870500e+05
max	2210.929932	2228.989990	2174.070068	2206.090088	2206.090088	1.583610e+07

As we can see above, the data contains 5223 points, and we will later split them into train and test datasets. The table also tells a lot of statistical information about the Booking stock data.For example, we can see the minimum ‘Adj Close’ price is about $6.6 20 years ago, and the maximum price is about $2206.09 That is about 334 times the difference.

- Data analysis and visualization

The profit or loss calculation is usually determined by the closing price of a stock for the day, hence we will consider the closing price as the target variable. Let’s plot the target variable to understand how it’s shaping up in our data:

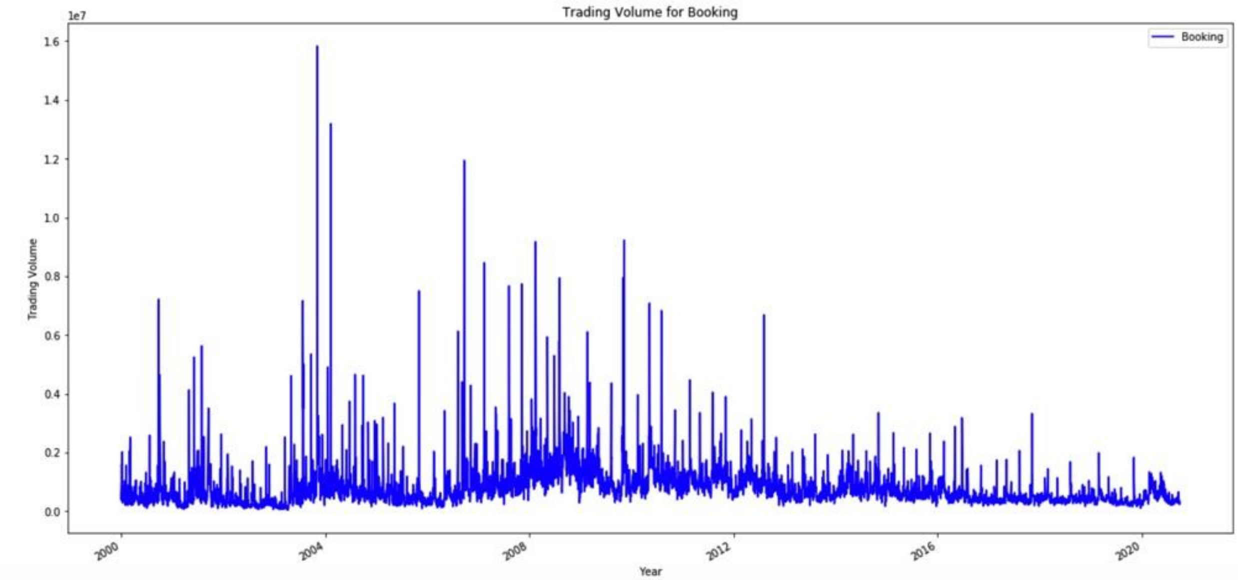
```python
ax = booking_hist.plot(x='Date', y='Adj Close', title='Adj Close price for Booking', figsize=(15, 8), color='blue')
ax.set_xlabel('Year')
ax.set_ylabel('Adj Close ($)')
ax.legend(['Booking'])
plt.show()
```



Stock trading volume refers to the amount of shares traded in a particular stock over a period of time. Often volume is measured in terms of shares traded per day. Remember that the number of shares bought and sold.

If there's a higher volume of trading in a particular stock, that naturally means that investors are interested in buying or selling it. If volume and price are on the rise, it means investors are betting the company will do well. If volume is up but price is down, it means more investors are looking to sell.

```
ax = booking_hist.plot(x='Date', y='Volume', title='Trading Volume for Booking', figsize=(20, 10), color='blue')
ax.set_xlabel('Year')
ax.set_ylabel('Trading Volume')
ax.legend(['Booking'])
plt.show()
```



This graph already says a lot of things. The specific reason I picked this company over others is that this graph is bursting with different behaviors of stock prices over time. This will make the learning more robust as well as give you a change to test how good the predictions are for a variety of situations. Another thing to notice is that the traded values from 2003 to 2012 are much higher and fluctuate more than the traded values after 2012.

```
booking_hist[booking_hist['Volume'] == booking_hist['Volume'].max()]
```

	Date	Open	High	Low	Close	Adj Close	Volume
966	2003-11-05	22.389999	22.77	21.559999	21.66	21.66	15836100

- Data Preprocessing

  Since most machine learning/deep learning techniques work best with normalized data, I will use MinMaxScaler to normalize the data.

```python
# keep only used column
booking_df.drop(columns=['Open', 'High', 'Low', 'Close', 'Volume'], inplace=True)
```

a) Normalizing the Data

we need to define a scaler to normalize the data. MinMaxScalar scales all the data to be in the region of 0 and 1.

```python
sc = MinMaxScaler(feature_range=(0.01,0.99))
booking_df['Scaled_Close'] = sc.fit_transform(booking_df[['Adj Close']])
```

```python
booking_df.head()
```

	Date	Adj Close	Scaled_Close
0	2006-02-06	31.701359	0.309240
1	2006-02-07	30.850378	0.300328
2	2006-02-08	29.924585	0.290632
3	2006-02-09	32.262444	0.315117
4	2006-02-10	32.019306	0.312570

- Data Modeling

  There are several possible models for the solution (Linear Regression, KNN, random forest, xgboost, etc). Among those techniques, a deep learning method named LSTM (Long Short Term Memory),LSTM is very powerful in sequence prediction, since it is able to store the past information. I will mainly focus on the model by LSTM, and hope its performance is acceptable.

  - Algorithm – LSTM:

    LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. LSTM has three gates:

    - **The input gate:** The input gate adds information to the cell state
    - **The forget gate:** It removes the information that is no longer required by the model
    - **The output gate:** Output Gate at LSTM selects the information to be shown as output

    For a more detailed understanding of LSTM and its architecture, you can go through the below article: Introduction to Long Short Term Memory

    For now, let us implement LSTM as a black box and check it's performance on our particular data.

  - Metrics

For the metrics of model performance, evaluation, I will mainly use Root Mean Square Deviation (RMSD) and Mean Absolute Percentage Error (MAPE).

1. Split the data into train and test datasets. Here we define 4 main parameters as below.

```
# we will test/validate on 40 periods (40x7=280 days)
num_periods = 40
forcast_days = 7
lookback_days = 15
window_size  = 15
```

The ‘num_periods’ is prediction periods that we will test, and compare the results to the benchmark model. The ‘forcast_days’ is defined as how manys we want to forecast based on historical data. The ‘lookback_days’ will be used for the LSTM model as the input feature shape, and the ‘window_size’ is the parameter for the Simple Moving Average benchmark model.

We can then split the data as below. The result train and test sets will have three columns. Beside, for the test dataset, we will need to create an extended dataset, so that it will contain some previous prices, for the purpose of feature creation and moving average price calculation.

```
#test data 40 periods (40x15=600 days)
n_test = num_periods*forcast_days
# rest is train data
n_train = n - n_test
```

```
# count of test data values
n_test
```

```
280
```

```
# count of train data values
n_train
```

```
4943
```

```
dataset_train , dataset_test = train_test_split(booking_df, train_size=n_train, test_size=n_test, shuffle=False)
```

```
148]: train_set = dataset_train[['Scaled_Close']].values
      test_set = dataset_test[['Scaled_Close']].values

149]: # extend the test set to add some previous information
      test_set_extend = dataset_test_extend[['Scaled_Close']].values
      print(train_set.shape, test_set.shape, test_set_extend.shape)

      (4943, 1) (280, 1) (310, 1)

150]: dataset_test.head()

150]:
```

	Date	Adj Close	Scaled_Close
4943	2019-08-26	1916.810059	0.861109
4944	2019-08-27	1919.989990	0.862526
4945	2019-08-28	1942.000000	0.872333
4946	2019-08-29	1957.140015	0.879078
4947	2019-08-30	1966.410034	0.883209

```
151]: dataset_train.head()

151]:
```

	Date	Adj Close	Scaled_Close
0	1999-12-31	284.250	0.133709
1	2000-01-03	307.500	0.144068
2	2000-01-04	324.000	0.151420
3	2000-01-05	359.625	0.167293
4	2000-01-06	345.000	0.160777

2. Model implementation - Long Short Term Memory (LSTM)

- Once we have defined our parameters such as 'lookback_days', 'forcast_days', and have our train and test datasets. The next step is to generate the sequence for the LSTM model. Below we have the function to do that.

```
def generate_train_test_sequence(t_data, lookback_days, forcast_days, sequence, last_index):
    """
    Generate sequence array for train data, test data
    inclduing both X and y
    """
    X = []
    y = []
    m = lookback_days
    n =forcast_days
    N = len(t_data)

    # train sequence is continuous = 1
    # test sequence is not continuous, will be in 20 groups, each group contains 7 days

    for i in range(0, N, sequence):
        # input sequence : x[i].....x[i+m]
        # output sequence: x[i+m+1]....x[i+m+n]
        # last index is (i+m+n)
        end_index = i + m + n # find the end of this sequence
        # check if we are out of index
        if end_index > N-last_index:
            break
        seq_x = t_data[i:(i+m)]
        seq_y = t_data[(i+m):(i+m+n)]
        X.append(seq_x)
        y.append(seq_y)

    array_X = np.array(X) # shape (N, m, 1)
    array_y = np.array([list(a.ravel()) for a in np.array(y)]) # shape (N, n, 1) convert to (N, n)
    return array_X, array_y
```

We can generate our training sequence as below.

```
# train sequence is continuous
X_train, y_train = generate_train_test_sequence(train_set, lookback_days, forcast_days, 1, 1)
print(X_train.shape, y_train.shape)
```

```
(4921, 15, 1) (4921, 7)
```

For each X_train data point, the shape is (15, 1), which is the last 15 days of normalized 'Adj Close' price. The datapoint for y_train, will be an array of 7 days prices. If we want to know the real price after prediction, we will need to use the normalization function to convert it back.

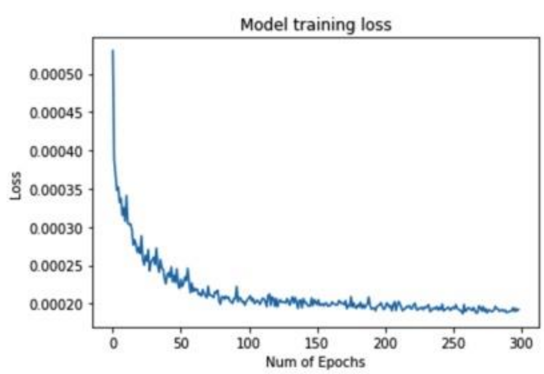
- Since the training data is only 5223 points, I will just use a simple LSTM model, instead of a complicated model (which will be two or more layers). The loss function of the LSTM will be `mean_squared_error', and the activation function I will choose 'relu'. The input data shape, as expected, will be (lookback_days, 1). The model function is as below:

```
model = Sequential()
model.add(LSTM(units=30, activation='relu', input_shape=(lookback_days,1)))
model.add(Dense(forcast_days))
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 30)                3840
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 217
=================================================================
Total params: 4,057
Trainable params: 4,057
Non-trainable params: 0
_________________________________________________________________
```

- Calculate training loss versus the epoches as below:



As we can see above, the model reaches convergence when epoch is above 200.

- Once we have finished the model training, we need to test the model prediction on the test dataset. As we mentioned in the previous sections, we will use 40 periods of 'forecast_days' to test the prediction (for 7 days forecast, this will be 280 days).

```python
# test sequence is not continuous, will be in 20 groups, each group contains 7 days
X_test, y_test = generate_train_test_sequence(test_set_extend, lookback_days, forcast_days,forcast_days,0)
print(test_set_extend.shape, X_test.shape, y_test.shape)
```

```
(295, 1) (40, 15, 1) (40, 7)
```

- Once we have the above test data sequence, we can use our trained LSTM model to perform the prediction. Note that the prediction value will be a normalized price, so we will need to use the ‘inverse_transform’ to transform it back to real price.

```python
LSTM_prediction_scaled = model.predict(X_test)
LSTM_prediction = sc.inverse_transform(LSTM_prediction_scaled)

train_set = train_set.reshape((-1))
test_set = test_set.reshape((-1))
LSTM_prediction = LSTM_prediction.reshape((-1))
```

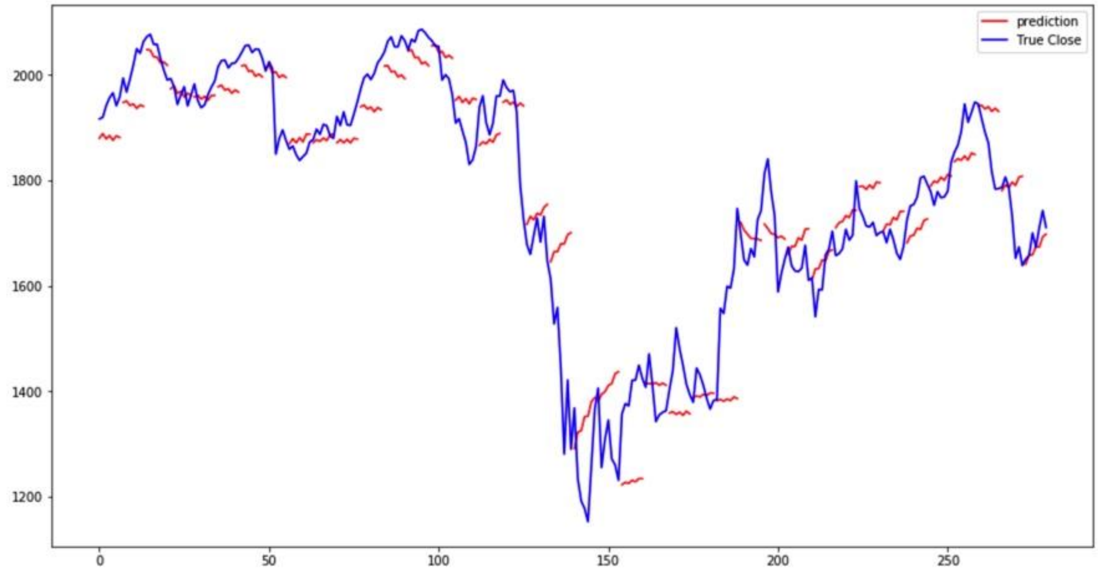
```python
dataset_test['LSTM_Prediction'] = LSTM_prediction
dataset_test.head()
```

	Date	Adj Close	Scaled_Close	LSTM_Prediction
4943	2019-08-26	1916.810059	0.861109	1880.174561
4944	2019-08-27	1919.989990	0.862526	1889.030396
4945	2019-08-28	1942.000000	0.872333	1878.846802
4946	2019-08-29	1957.140015	0.879078	1885.351685
4947	2019-08-30	1966.410034	0.883209	1876.185791

- We can visualize the prediction price along with the real price, with matplotlib.

```python
def plot_prediction_comparison(true_set, prediction_set):
    plt.figure(figsize = (15,8))

    for i in range(0, len(prediction_set), forcast_days):
        indexs = [i+x for x in range(forcast_days)]
        values = prediction_set[i:i+forcast_days]
        plt.plot(indexs, values, color='r')

    plt.plot(0, prediction_set[0], color='r', label='prediction') # just used to add label

    plt.plot(true_set, color='b', label='True Close')
    plt.legend(loc='best')
    plt.show()
```

```
plot_prediction_comparison(dataset_test['Adj Close'].values, dataset_test['LSTM_Prediction'].values)
```



As we can see above, when the price increases too fast, the prediction keeps output a big drop signal, in which some of them are correct, some of them are not.

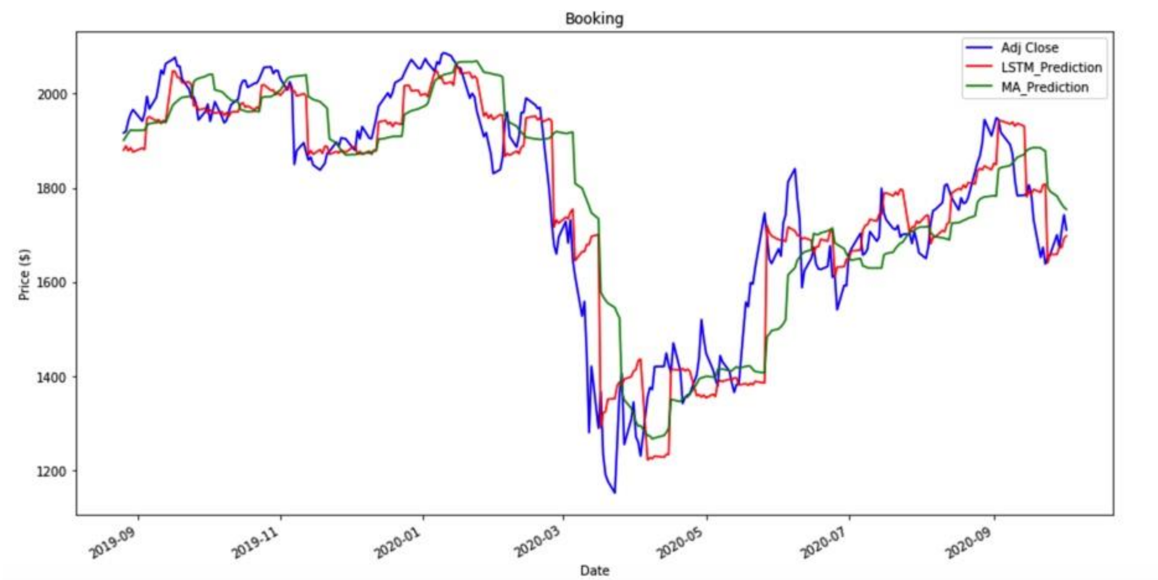
- Now we will use moving average price for prediction using the below function

```
for index in range(window_size, len(dataset_test_extend), forcast_days):
    for i in range(0,forcast_days):
        if index+i >= len(dataset_test_extend):
            break
        # window for real price, eg [1, 2, 3, 4, 5, 6, 7, 8, 9]
        window_real = make_window(window_size-i, index+i-window_size)

        # window for predicted price, eg [10]
        window_MA = make_window(i, index)

        price_window = pd.concat([ dataset_test_extend['Adj Close'].iloc[window_real],
                                   dataset_test_extend['MA_Prediction'].iloc[window_MA] ])
        next_mean_prediction = price_window.mean(axis=0)

        dataset_test_extend.iat[index+i, dataset_test_extend.columns.get_loc('MA_Prediction')] = next_mean_prediction
```

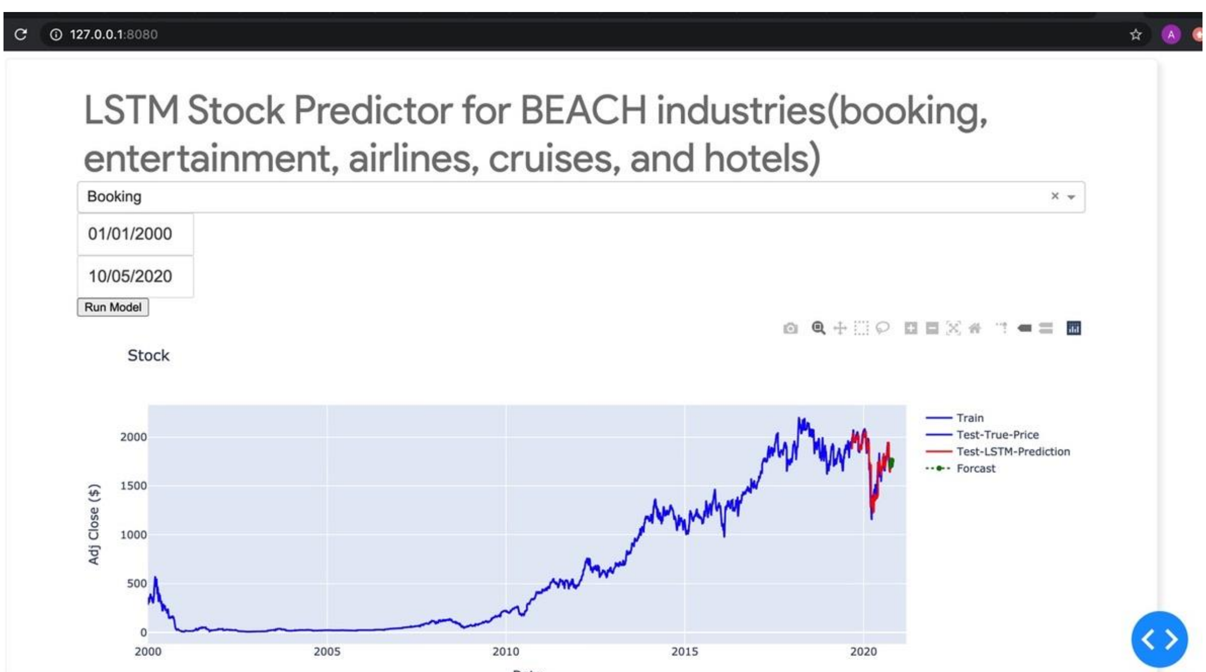
```
dataset_test['MA_Prediction'] = dataset_test_extend[lookback_days:]['MA_Prediction'].values
dataset_test.head()
```

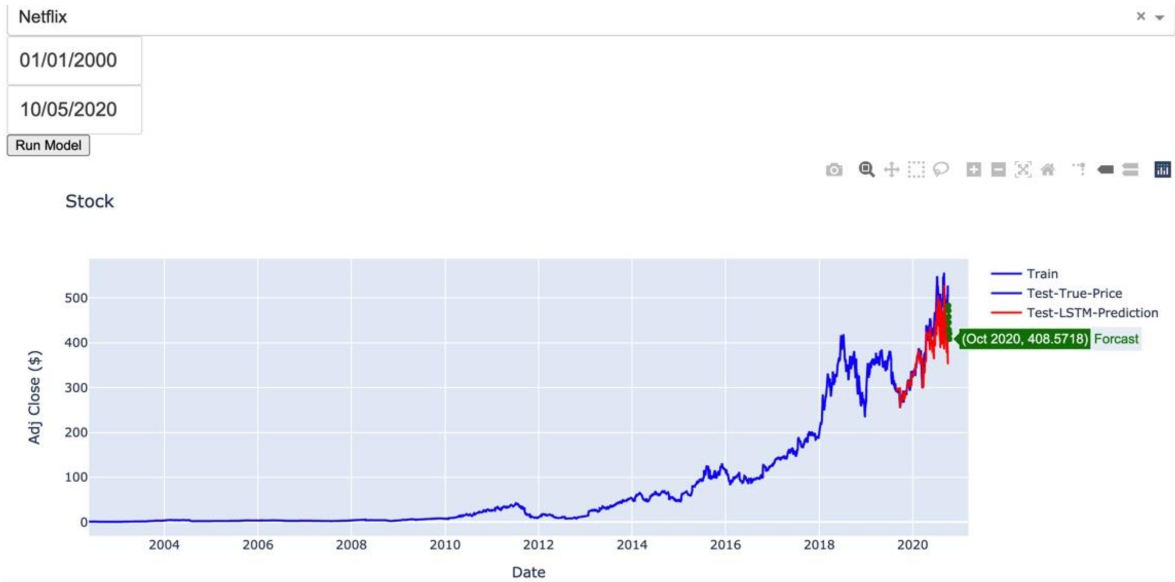
	Date	Adj Close	Scaled_Close	LSTM_Prediction	MA_Prediction
4943	2019-08-26	1916.810059	0.861109	1880.174561	1901.235327
4944	2019-08-27	1919.989990	0.862526	1889.030396	1908.744351
4945	2019-08-28	1942.000000	0.872333	1878.846802	1916.892639
4946	2019-08-29	1957.140015	0.879078	1885.351685	1923.248145
4947	2019-08-30	1966.410034	0.883209	1876.185791	1922.064020

- Using Moving Average, LSTM we can visualize the prediction price along with the real price, with matplotlib.



- Refinement
  - The above model implementation is more towards a static way. It is not very convenient for the user to use, and check the price on a selected day. Hence, we are going to build a web application using, which can help users train and visualize the stock in an interactivate way.
  - As we mentioned, the Dash Plotly will be used. The related code is in the 'lstm' folder. In order to start the web application, we will need to run 'Python index.py', and then open the web address at http://127.0.0.1:8080/
  - As we see below, the application looks like below:



- We can pick the stock symbol by choosing from the dropdown window, and select the date range to download data. Similar as before, the data will be split into train and test. After finishing the selection, we can run the model by clicking on the 'Run Model' button.
- You can monitor the program progress from the terminal, usually it will take about 3-7 minutes. Once done, we will be able to see the result plot.
- Unlike the static plot in the jupyter notebook, here the result is an interactive plot. You can view the details in each stock data point. More importantly, you can select a subset of the figure and zoom it bigger, so you can see the details. For example, we can zoom the test and prediction range for the above figure as below.


-