Stock Prices Prediction Using Machine Learning and Deep Learning Techniques

---

Module 2: exploratory data analysis

Course Name	AI Algorithms I
Course Code	AIDI 11688
Course Facilitator	Marcos Bittencourt
Student	Riddhi Thakkar (100799412)
Date	01/12/2020

Executive Summary

The share prices in the stock market fluctuate on a daily basis due to continuous buying and selling. Stock prices move in trends and cycles and are never stable. An investor in the stock market is interested to get a good return on his investment and hence purchases securities at allow price and sells them at a high price. However, to gain successfully return on investments in a volatile stock market is not an easy job to do.

Data Overview

This notebook analyzes content present in the DIJA 30 Stock Time Series dataset. It provides detailed insight into the stock trends and also includes a guide to train and fit a LSTM model for stock price prediction.

we find the following information about the columns:

- Date - Date for which the price is given
- Open - Price of the stock at market open (In USD)
- High - Highest price reached in the day
- Low - Lowest price reached in the day
- Close - Closing price for the day
- Volume - Number of shares traded -Name` - the stock's ticker name

Further, data has been collected using the pandas_datareader package which fetches data from Google Finance API. This could be a cause for concern as the API has long been deprecated.

```
df = pd.read_csv('C:/all_stocks_2006-01-01_to_2018-01-01.csv')
df.head()
```

The dataset has some missing values. We will analyse this

The Open column has the maximum number of null values. Let's find the rows for which the values are missing.

```
df.isnull().sum()
df[df.Open.isnull()]
```

	Date	Open	High	Low	Close	Volume	Name
2913	2017-07-31	NaN	201.66	NaN	201.17	1833625	MMM
5933	2017-07-31	NaN	85.70	NaN	85.23	3079797	AXP
11972	2017-07-31	NaN	NaN	NaN	242.46	5777271	BA
14992	2017-07-31	NaN	NaN	NaN	113.95	4486013	CAT
18012	2017-07-31	NaN	110.00	NaN	109.19	7561205	CVX
24051	2017-07-31	NaN	NaN	45.79	45.84	13622891	KO
25815	2012-08-01	NaN	NaN	NaN	49.14	0	DIS
27071	2017-07-31	NaN	110.14	NaN	109.93	6815349	DIS
30091	2017-07-31	NaN	80.39	NaN	80.04	12820175	XOM
33111	2017-07-31	NaN	25.69	NaN	25.61	30616287	GE
36131	2017-07-31	NaN	225.50	NaN	225.33	1999637	GS
39151	2017-07-31	NaN	149.98	NaN	149.60	5253318	HD
42171	2017-07-31	NaN	144.93	NaN	144.67	4355718	IBM
48210	2017-07-31	NaN	133.58	NaN	132.72	5440788	JNJ
51230	2017-07-31	NaN	92.36	NaN	91.80	11520329	JPM

The data is missing only for 31 July, 2017. This could be because:

- The API had an unexpected error while fetching the data.
- The data for this day does not exist in the source.

Let's check the number of business days for which the records as missing.

```
rng = pd.date_range(start='2006-01-01', end='2018-01-01', freq='B')
rng[~rng.isin(df.Date.unique())]
```

```
DatetimeIndex(['2006-01-02', '2006-01-16', '2006-02-20', '2006-04-14',
               '2006-05-29', '2006-07-04', '2006-09-04', '2006-11-23',
               '2006-12-25', '2007-01-01',
               ...
               '2017-01-02', '2017-01-16', '2017-02-20', '2017-04-14',
               '2017-05-29', '2017-07-04', '2017-09-04', '2017-11-23',
               '2017-12-25', '2018-01-01'],
              dtype='datetime64[ns]', length=111, freq=None)
```

There are about 111 days for which the stock price data is missing. This could lead to potential problems with the analysis

```
df.groupby('Name').count().sort_values('Date', ascending=False)['Date']
```

```
Name
JNJ     3020
JPM     3020
WMT     3020
VZ      3020
UTX     3020
UNH     3020
TRV     3020
PG      3020
PFE     3020
NKE     3020
MMM     3020
MCD     3020
KO      3020
XOM     3020
GE      3020
IBM     3020
HD      3020
GS      3020
AXP     3020
BA      3020
CAT     3020
DIS     3020
CVX     3020
CSCO    3019
AMZN    3019
INTC    3019
```

Some of the companies(Google, Microsoft, etc.) don't have an entry for the date 2010-04-01.

Let's check if all the listed companies have an entry on each date.

```
df.Name.unique().size
```

```
31
```

Data Cleaning

first fill in the null values on date 31 july, 2017 with the values from the previous day(i.e 28th July, 2017)

```
df.set_index('Date', inplace=True)

#Backfill `Open` column
values = np.where(df['2017-07-31']['Open'].isnull(), df['2017-07-28']['Open'], df['2017-07-31']['Open'])
df['2017-07-31']= df['2017-07-31'].assign(Open=values.tolist())

values = np.where(df['2017-07-31']['Close'].isnull(), df['2017-07-28']['Close'], df['2017-07-31']['Close'])
df['2017-07-31']= df['2017-07-31'].assign(Close=values.tolist())

values = np.where(df['2017-07-31']['High'].isnull(), df['2017-07-28']['High'], df['2017-07-31']['High'])
df['2017-07-31']= df['2017-07-31'].assign(High=values.tolist())

values = np.where(df['2017-07-31']['Low'].isnull(), df['2017-07-28']['Low'], df['2017-07-31']['Low'])
df['2017-07-31']= df['2017-07-31'].assign(Low=values.tolist())

df.reset_index(inplace=True)
```

```
df[df.Date == '2017-07-31']
```

	Date	Open	High	Low	Close	Volume	Name
2913	2017-07-31	200.79	201.66	198.69	201.17	1833625	MMM
5933	2017-07-31	83.88	85.70	83.62	85.23	3079797	AXP
8952	2017-07-31	149.90	150.33	148.13	148.73	19845920	AAPL
11972	2017-07-31	240.82	242.00	238.55	242.46	5777271	BA
14992	2017-07-31	114.45	114.90	113.48	113.95	4486013	CAT
18012	2017-07-31	106.71	110.00	106.36	109.19	7561205	CVX
21031	2017-07-31	31.54	31.59	31.37	31.45	19256428	CSCO
24051	2017-07-31	46.00	46.12	45.79	45.84	13622891	KO
27071	2017-07-31	109.98	110.14	109.66	109.93	6815349	DIS

Similarly, we noticed that 8 of the 31 stocks have missing data on 1st April, 2014. As done before, we will use the stock prices of the previous day to fill the data.

```
missing_data_stocks = ['CSCO','AMZN','INTC','AAPL','MSFT','MRK','GOOGL', 'AABA']
```

```
columns = df.columns.values
```

```
for stock in missing_data_stocks:
    tdf = df[(df.Name == stock) & (df.Date == '2014-03-28')].copy()
    tdf.Date = '2014-04-01'
    pd.concat([df, tdf])
print("Complete")
```

```
Complete
```

Feature Engineering

Since we have four values of stock price for each day, let's create a feature called Price which is the average of all these values.

```
values = (df['High'] + df['Low'] + df['Open'] + df['Close'])/4
df = df.assign(Price=values)
df.head()
```

	Date	Open	High	Low	Close	Volume	Name	Price
0	2006-01-03	77.76	79.35	77.24	79.11	3117200	MMM	78.3650
1	2006-01-04	79.49	79.49	78.25	78.71	2558000	MMM	78.9850
2	2006-01-05	78.41	78.65	77.56	77.99	2529500	MMM	78.1525
3	2006-01-06	78.64	78.90	77.64	78.63	2479500	MMM	78.4525
4	2006-01-09	78.50	79.83	78.46	79.02	1845600	MMM	78.9525

We can see that 75% of the stocks have a price of under 94$, indicating that the stock market is mostly dominated by the bigger companies.

Let's go one step further and compute the daily growth of the stock prices compared to day 1 of the prices(i.e compute cumalative compound growth)

```
stock_names = df.Name.unique()
```

```
day_prices = df[df.Date == df.Date.min()].Price
price_mapping = {n : c for n, c in zip(stock_names, day_prices)}
base_mapping = np.array(list(map(lambda x : price_mapping[x], df['Name'].values)))
df['Growth'] = df['Price'] / base_mapping - 1
df.Growth.describe()
```

```
count    93611.000000
mean         0.789919
std          1.907922
min         -0.808701
25%          0.035007
50%          0.342423
75%          0.816421
max         24.392810
Name: Growth, dtype: float64
```

The worst performing company had a decline of 81% in their shares compared to their first ever opening price and the best company had a whopping 2439% increase in their share price.

Time series Analysis

Find out the top 5 best and worst performing stocks

```
sample_dates = pd.date_range(start='2006-01-01', end='2018-01-01', freq='B')
```

```
year_end_dates = sample_dates[sample_dates.is_year_end]
```

```
year_end_dates
```

```
DatetimeIndex(['2006-12-29', '2007-12-31', '2008-12-31', '2009-12-31',
               '2010-12-31', '2011-12-30', '2012-12-31', '2013-12-31',
               '2014-12-31', '2015-12-31', '2016-12-30', '2017-12-29'],
              dtype='datetime64[ns]', freq=None)
```

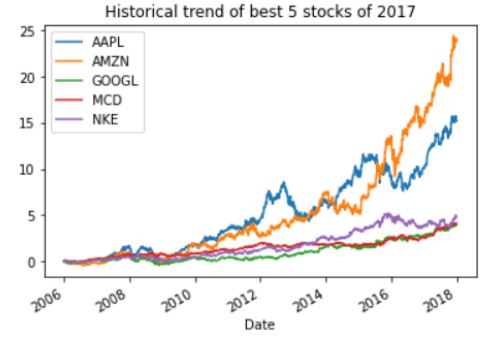
```
worst_stocks = df[df.Date == df.Date.max()].sort_values('Growth').head(5)
best_stocks = df[df.Date == df.Date.max()].sort_values('Growth', ascending=False).head(5)
```

```
tdf[tdf.Name.isin(ws)].groupby('Name').Growth.plot(title='Historical trend of worst 5 stocks of 2017', legend=True)
```

```
Name
AABA    AxesSubplot(0.125,0.2;0.775x0.68)
GE      AxesSubplot(0.125,0.2;0.775x0.68)
PFE     AxesSubplot(0.125,0.2;0.775x0.68)
PG      AxesSubplot(0.125,0.2;0.775x0.68)
XOM     AxesSubplot(0.125,0.2;0.775x0.68)
Name: Growth, dtype: object
```


```
tdf[tdf.Name.isin(bs)].groupby('Name').Growth.plot(title='Historical trend of best 5 stocks of 2017', legend=True)
```

```
Name
AAPL    AxesSubplot(0.125,0.2;0.775x0.68)
AMZN    AxesSubplot(0.125,0.2;0.775x0.68)
GOOGL   AxesSubplot(0.125,0.2;0.775x0.68)
MCD     AxesSubplot(0.125,0.2;0.775x0.68)
NKE     AxesSubplot(0.125,0.2;0.775x0.68)
Name: Growth, dtype: object
```



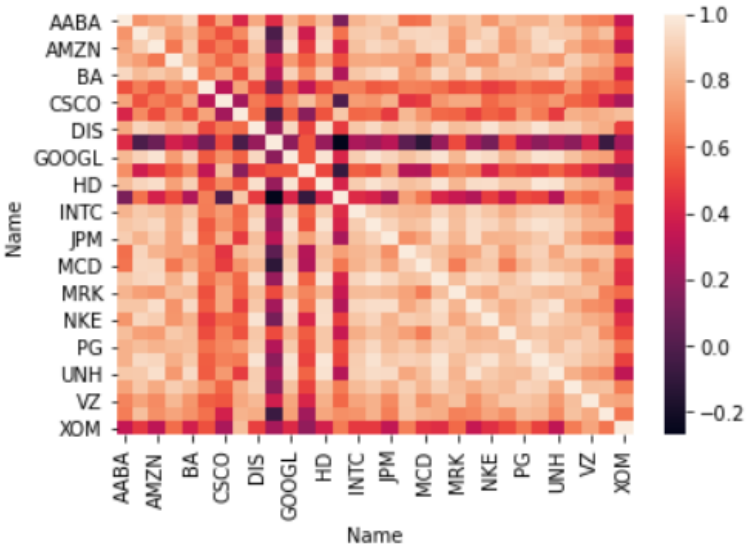
According to the above data, an investment of 1 USD in Google in 2006 would have increased the amount to 393USD in 2017. However, the same investment in General Electric Co would have decreased the amount of 0.49USD. This information could help us create an ideal portfolio of stocks to maximize profit.

Corelation Analysis

Find some corelation between the growth vs time of each stock in the dataset

```
corr = df.pivot('Date', 'Name', 'Growth').corr()
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f392f4af0>
```



Although we can see some positive and negative corelations, the graph above is very dense. Let's us just focus on high positive and high negative corelations.

```
def unique_corelations(indices):
    mapping = {}
    for record in indices:
        (stock_a, stock_b) = record
        value_list = mapping.get(stock_a)
        if value_list:
            if stock_b not in value_list:
                value_list.append(stock_b)
                mapping.update({stock_a: value_list})
        else:
            mapping.update({stock_a: [stock_b]})

    return mapping

def filter_corelations_positive(corr, threshold=0.9):
    indices = np.where(corr > threshold)
    indices = [(corr.index[x], corr.columns[y]) for x, y in zip(*indices)
                                        if x != y and x < y]
    mapping = unique_corelations(indices)
    return mapping

def filter_corelations_negative(corr, threshold=-0.8):
    indices = np.where(corr < threshold)
    indices = [(corr.index[x], corr.columns[y]) for x, y in zip(*indices)
                                        if x != y and x < y]
    mapping = unique_corelations(indices)
    return mapping
```

```
filter_corelations_positive(corr, threshold=0.95)
```

```
{'AMZN': ['GOOGL', 'HD', 'MSFT', 'UNH'],
 'BA': ['JPM'],
 'DIS': ['HD', 'NKE', 'TRV'],
 'GOOGL': ['HD', 'JNJ', 'MMM', 'MSFT', 'TRV', 'UNH'],
 'HD': ['JNJ', 'MMM', 'NKE', 'TRV', 'UNH'],
 'JNJ': ['MMM', 'MSFT', 'TRV'],
 'JPM': ['MSFT'],
 'MMM': ['MSFT', 'TRV', 'UNH'],
 'MSFT': ['UNH'],
 'NKE': ['TRV']}
```

```
filter_corelations_negative(corr, -0.1)
```

```
{'GE': ['IBM', 'MCD']}
```

Conclusion

After working exploratory data analysis ,we have achieved the following:

- Successfully analysed the trends present in stock market prices of various companies and concluded that there is indeed a temporal relationship between these prices.
- Devised a model capable of reliably forecasting the stock prices of a company, thereby providing a tool to maximize profits.