Machine Learning Engineer Nanodegree: Capstone Proposal

1. Domain Background

The Dow Jones Industrial Average (DJIA) is a stock market index that tracks the performance of 30 large public companies traded in the New York Stock Exchange and NASDAQ. The DJIA is one of the commonly followed stock indices, and it is meant to reflect the overall health of the U.S. economy. The stocks in this index are thought to be well-established and stable, relative to other stocks. However, prior research has found that these companies are still affected by large shocks, including financial crashes, monetary policies, elections, and wars.[1] The table below shows the price, change, % change, and volume of the 30 companies in the DJIA, as of June 8, 2020.[2] As we can see, the index includes some of the largest, well-known companies.

Symbol	Company Name	Last Price	Change	% Change	Volume
AAPL	Apple Inc.	333.46	1.96	0.59%	23,252,140
UNH	UnitedHealth Group Incorporated	309.48	-2.37	-0.76%	4,529,382
HD	The Home Depot, Inc.	256.77	1.87	0.73%	3,690,223
BA	The Boeing Company	230.5	25.07	12.20%	79,329,722
GS	The Goldman Sachs Group, Inc.	220.81	2.89	1.33%	3,249,401
MCD	McDonald's Corporation	202.65	5.49	2.78%	4,049,039
V	Visa Inc.	199.6	-0.01	-0.01%	6,622,345
MSFT	Microsoft Corporation	188.36	1.16	0.62%	31,399,969
MMM	3M Company	166.87	-0.54	-0.32%	2,917,911
JNJ	Johnson & Johnson	146.77	-0.53	-0.36%	7,095,310
CAT	Caterpillar Inc.	137.72	2.6	1.92%	3,717,515
IBM	International Business Machines Corporation	135.75	3.69	2.79%	5,110,385
TRV	The Travelers Companies, Inc.	128	3.64	2.93%	2,779,794
DIS	The Walt Disney Company	127.28	2.46	1.97%	13,043,736
WMT	Walmart Inc.	121.24	-0.32	-0.26%	9,208,824
PG	The Procter & Gamble Company	119.05	0.72	0.61%	6,208,190
AXP	American Express Company	113.67	3.94	3.59%	6,267,974
JPM	JPMorgan Chase & Co.	113.45	2.22	2.00%	22,816,894
NKE	NIKE, Inc.	104.29	1.58	1.54%	5,519,371
CVX	Chevron Corporation	103.24	2.43	2.41%	10,868,467
MRK	Merck & Co., Inc.	82.9	0.64	0.78%	9,122,832
RTX	Raytheon Technologies Corporation	74.16	2.09	2.90%	11,208,513
INTC	Intel Corporation	63.67	-0.67	-1.04%	19,827,535
VZ	Verizon Communications Inc.	58.09	0.35	0.61%	14,289,470
XOM	Exxon Mobil Corporation	54.74	1.66	3.13%	33,481,345
KO	The Coca-Cola Company	49.85	0.76	1.55%	19,027,610
CSCO	Cisco Systems, Inc.	48.13	0.3	0.63%	16,553,068
WBA	Walgreens Boots Alliance, Inc.	47.02	1.67	3.68%	7,011,490
DOW	Dow Inc.	45.9	1.93	4.39%	5,519,486
PFE	Pfizer Inc.	36.59	0.6	1.67%	28,018,488

Given the popularity of machine learning algorithms and their powerful predictive ability, prior research has tried to implement these techniques to analyze stock market data and predict stock prices.[3] Some have constructed sophisticated trading algorithms based on technical indicators, which are mathematical indicators based on a stock's patterns in price, volume, etc.[4] Others have used the textual content of managerial disclosures[5] or stochastic discount factors.[6] In this project, I will use the time series of stock prices of the components of the DJIA to see how well a machine learning model can predict their prices.

This is an interesting project to me because I am an accounting Ph.D. student who is naturally drawn to how corporate information is reflected into stock prices. This project is an opportunity for me to reinforce what I have learned in Udacity's Machine Learning Engineer Nanodegree course and to link it to my field of interest.

2. Problem Statement

For this project, my objective is to build a stock price predictor that takes daily stock prices of the DIJA components over a certain date range as input, and outputs projected estimates of the closing price for given query dates.

My implementation will consist of two components:

1. A training interface that accepts a data range (start_date, end_date) and a list of ticker symbols from the DJIA (e.g. PFE, AAPL), and builds a model of stock behavior.
2. A query interface that accepts a list of dates and a list of ticker symbols from the DJIA, and outputs the predicted stock prices for each of those stocks on the given dates. The query dates passed in must be after the training date range, and ticker symbols must be a subset of the ones trained on.

3. Datasets and Inputs

The dataset for this project consists of daily stock prices of DJIA components from January 1, 2017 to May 31, 2020. I will use the full years 2017, 2018, and 2019 to train the model, and the data for 2020 to test it.

Import modules

In [1]:
```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.dates as mdates
import datetime
import matplotlib.pyplot as plt
%matplotlib inline
```

Tickers to download

I first collect the list of tickers to download based on the components of the DJIA (see the table shown in section **1. Domain Background**).

In [2]:
```
DJIA_tickers = "PFE JNJ WMT MRK VZ DIS NKE GS PG MCD CSCO INTC MSFT HD KO " \
               "IBM WBA AAPL DOW MMM AXP V JPM UNH CVX CAT TRV RTX XOM BA"
```

Download data from Yahoo Finance

I use the package yfinance to download stock data from Yahoo Finance. To install this package, run `pip install yfinance` in the terminal. The data files are included in the folder `data/`. Skip to the load data step to avoid re-downloading.

In [3]:
```
data = yf.download(
    tickers = DJIA_tickers,
    start="2017-01-01", end="2020-05-31",
    interval = "1d",
    group_by = 'ticker',
    auto_adjust = True,
    prepost = False,
    threads = True,
    proxy = None
    )
```

```
[*********************100%***********************]  30 of 30 completed
```

In [4]:
```
for firm in DJIA_tickers.split():
    data[firm].to_csv('data/{}.csv'.format(firm))
```

Load data

In this step, I read csv files containing data on the stock prices of DJIA companies. I am only interested in predicting the closing price of the day, so I only keep the column *Close* and the column *Date* to be used as the index.

In [5]:
```
data = {}
for firm in DJIA_tickers.split():
    data[firm] = pd.read_csv('data/{}.csv'.format(firm), usecols=['Date','Close'], parse_dates=['Date']).rename(column
s={'Close': firm})
    data[firm].set_index('Date',inplace=True)
    data[firm] = data[firm][firm]
```

In [6]:
```
dfs = []
for firm in DJIA_tickers.split():
    dfs.append(data[firm])
```

In [7]:
```
df = pd.concat(dfs, axis=1)
df.head()
```

Out[7]:

Date	PFE	JNJ	WMT	MRK	VZ	DIS	NKE	GS	PG	MCD	...	AXP	V	JPM
2017-01-03	28.970671	105.502319	63.187412	54.672073	46.683941	101.584358	49.927341	227.745087	75.752029	109.632889	...	71.389961	77.675957	79.052376
2017-01-04	29.225266	105.329285	63.555534	54.653900	46.632622	102.886719	50.974297	229.215790	76.021935	109.504570	...	72.560287	78.311043	79.198174
2017-01-05	29.506187	106.431297	63.693573	54.635715	46.735260	102.829262	50.964691	227.509384	76.525734	109.706215	...	71.665886	79.229477	78.469170
2017-01-06	29.392067	105.921272	62.819294	54.781147	46.041534	104.361458	51.781124	230.884491	76.498756	110.677727	...	71.808617	80.323792	78.478271
2017-01-09	29.383287	105.903038	63.233440	55.535561	45.540134	103.767731	51.272057	228.989517	75.931961	110.375267	...	72.179688	79.874336	78.532944

5 rows × 30 columns

Explore data

In [8]:
```
df.shape
```

Out[8]:
```
(857, 30)
```

My data includes 857 rows representing one *trading* day each. Stock exchanges are usually closed on weekends and during holidays.

In [9]:
```
df.index.year.value_counts().sort_index()
```

Out[9]:
```
2017    251
2018    251
2019    252
2020    103
Name: Date, dtype: int64
```

The function `check_sample_size` below helps me check that all companies in the DJIA have nonmissing data for the 857 trading days I am examining.

In [10]:
```
def check_sample_size(val):
    '''
    highlight a company's count if it is smaller than 857.
    '''
    small = val < 857
    return 'background-color: yellow' if small else ''
```

In [11]:
```
# s = df.style.applymap(color_negative_red)
df.iloc[:,:15].describe().style.applymap(check_sample_size, subset=pd.IndexSlice['count',:])
```

Out[11]:

	PFE	JNJ	WMT	MRK	VZ	DIS	NKE	GS	PG	MCD	CSCO	INTC	MSFT	HD	KO
count	857	857	857	857	857	857	857	857	857	857	857	857	857	857	857
mean	35.247	127.667	93.1076	67.9651	49.7529	113.335	72.8251	212.774	92.9898	166.237	40.7966	45.8699	106.986	180.254	45.0385
std	4.14572	10.1927	17.6325	11.8644	6.41848	15.5264	15.1588	22.6167	17.3764	26.9349	7.98123	8.70746	34.1784	30.5307	5.31045
min	27.3466	101.786	60.4265	50.2529	38.0106	85.76	49.3107	134.132	66.933	109.505	27.0961	31.0911	58.7052	122.696	36.3363
25%	32.1759	121.68	80.7469	57.8497	43.702	101.821	57.4208	196.897	79.5861	149.155	31.9284	40.805	79.7452	154.706	41.3272
50%	34.6008	127.643	92.6281	66.5971	50.434	108.259	74.1949	214.96	85.1768	161.815	41.6642	46.2586	103.31	178.91	43.6711
75%	39.1811	133.752	109.235	79.468	55.3701	128.371	84.4194	228.486	110.655	188.963	46.2483	51.3282	134.913	201.642	47.6676
max	43.6889	154.439	131.75	91.2859	60.7771	150.737	104.031	261.714	126.298	215.83	56.656	67.7517	187.663	247.005	59.6072

In [12]:
```
df.iloc[:,15:].describe().style.applymap(check_sample_size, subset=pd.IndexSlice['count',:])
```

Out[12]:

	IBM	WBA	AAPL	DOW	MMM	AXP	V	JPM	UNH	CVX	CAT	TRV	RTX	XOM	BA
count	857	857	857	302	857	857	857	857	857	857	857	857	857	857	857
mean	132.308	62.7072	191.852	44.3263	182.019	98.7024	137.201	101.157	228.647	105.952	123.225	124.204	75.4303	68.4773	288.406
std	10.6844	10.422	50.7146	7.34222	21.8222	16.3075	34.2035	14.8405	38.9582	10.7041	18.5948	12.412	8.90378	8.60126	80.4693
min	93.5158	37.9201	110.269	21.6133	116.712	68.5791	77.676	75.7243	150.01	53.4355	83.7241	81.1475	46.7539	30.8518	95.01
25%	128.251	53.2209	153.826	40.828	165.334	86.38	108.099	89.7563	202.94	98.4767	112.094	116.273	69.5428	67.1264	225.619
50%	132.442	63.1505	180.403	45.9986	183.587	97.344	136.396	101.892	236.635	108.487	127.56	124.159	75.3951	70.6383	324.885
75%	137.318	71.9524	214.223	49.6078	196.504	111.873	169.157	108.507	255.176	114.102	136.19	132.51	80.538	73.1768	347.193
max	157.395	81.4276	326.317	55.2337	239.175	136.174	212.953	138.748	304.85	120.824	160.686	151.054	97.3674	79.1205	430.3

Notice that Dow Inc., with ticker *DOW*, only has 302 rows of data. The reason is that this company was created in early 2019, when DowDuPont Inc. split into three companies. One of them, Dow Inc., ended up replacing DowDuPont Inc. in the DJIA index.[7]

In [13]:
```
df.loc[df.DOW.notnull(), 'DOW']
```

Out[13]:
```
Date
2019-03-20    46.066597
2019-03-21    45.308067
2019-03-22    44.956558
2019-03-25    45.465328
2019-03-26    45.187813
                ...
2020-05-22    35.485043
2020-05-26    38.108109
2020-05-27    39.119999
2020-05-28    38.709999
2020-05-29    38.599998
Name: DOW, Length: 302, dtype: float64
```

Given that training the model with only 302 might result in less accurate predictions, I decided to drop this company from my analysis.

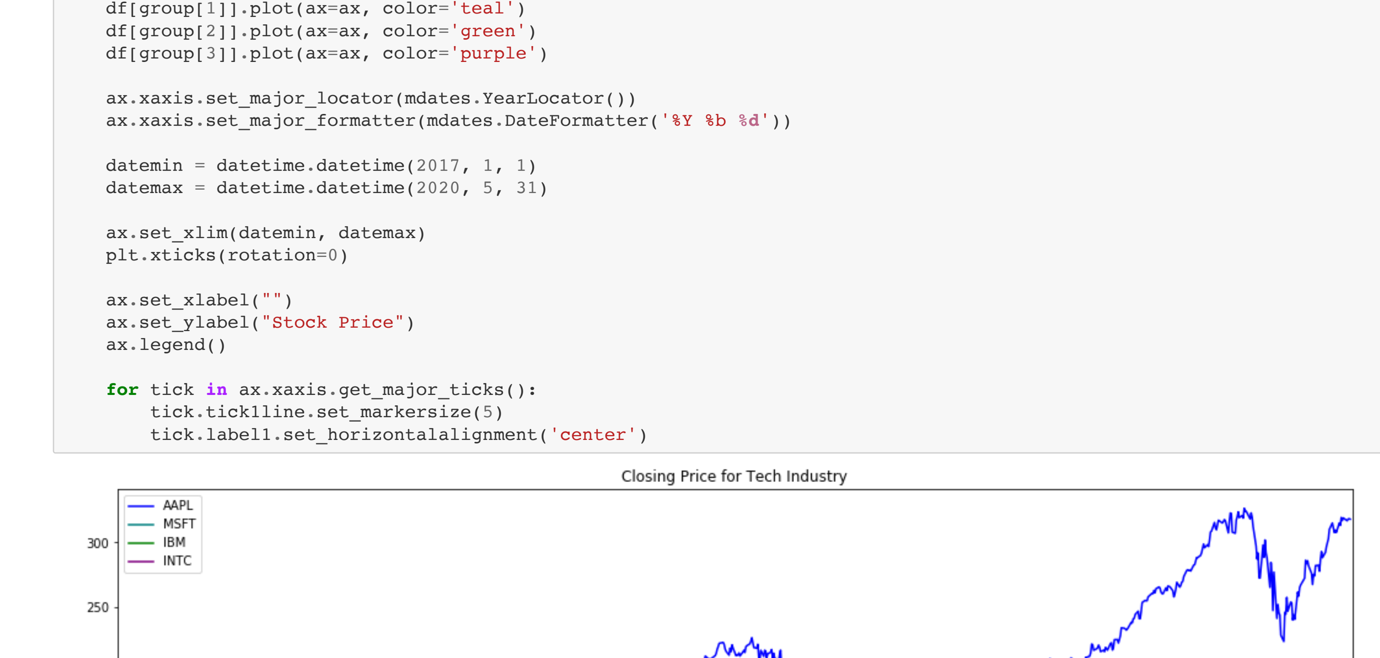
In [14]:
```
# Drop DOW
df.dropna(inplace=True, axis=1)
df.shape
```

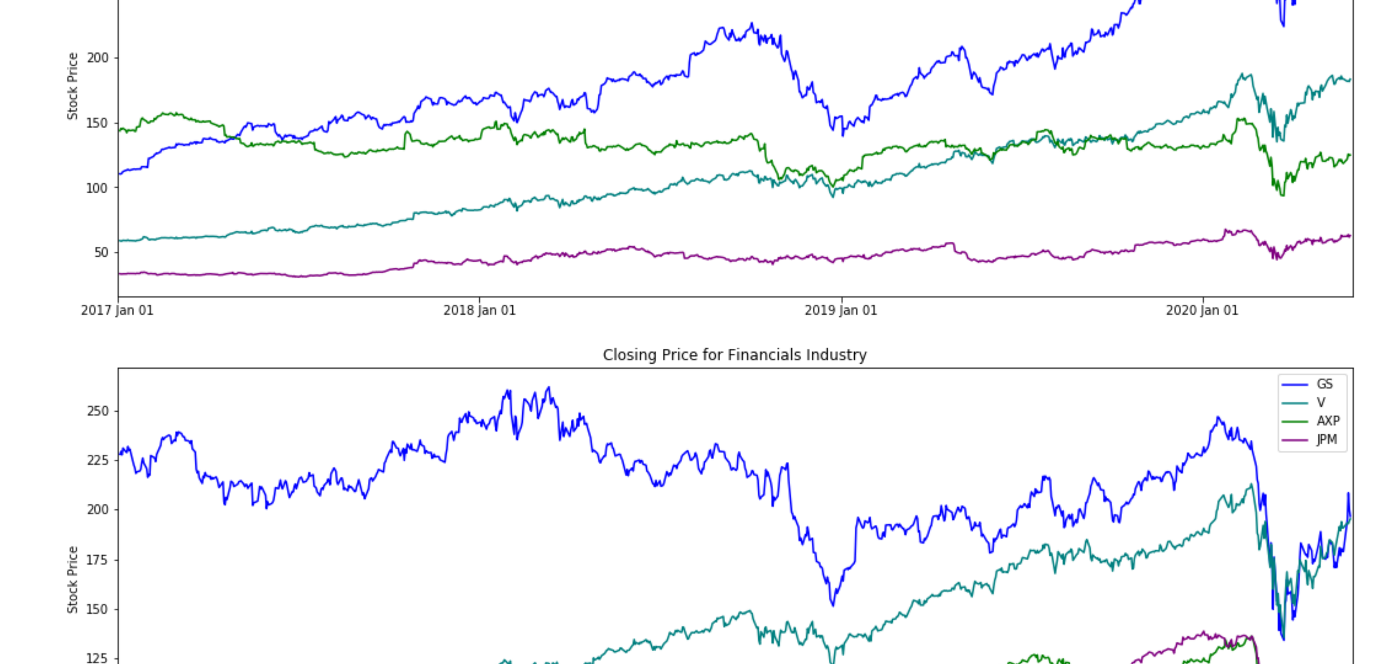
Out[14]:
```
(857, 29)
```

The DJIA data not only varies in terms of the volume and price level of its stocks, but it also varies in industry composition. As an illustration, we can review the stock behavior of stocks in the tech industry and financial stocks.

In [15]:
```
tech = ("Tech", "AAPL MSFT IBM INTC".split())
financials = ("Financials", "GS V AXP JPM".split())
```

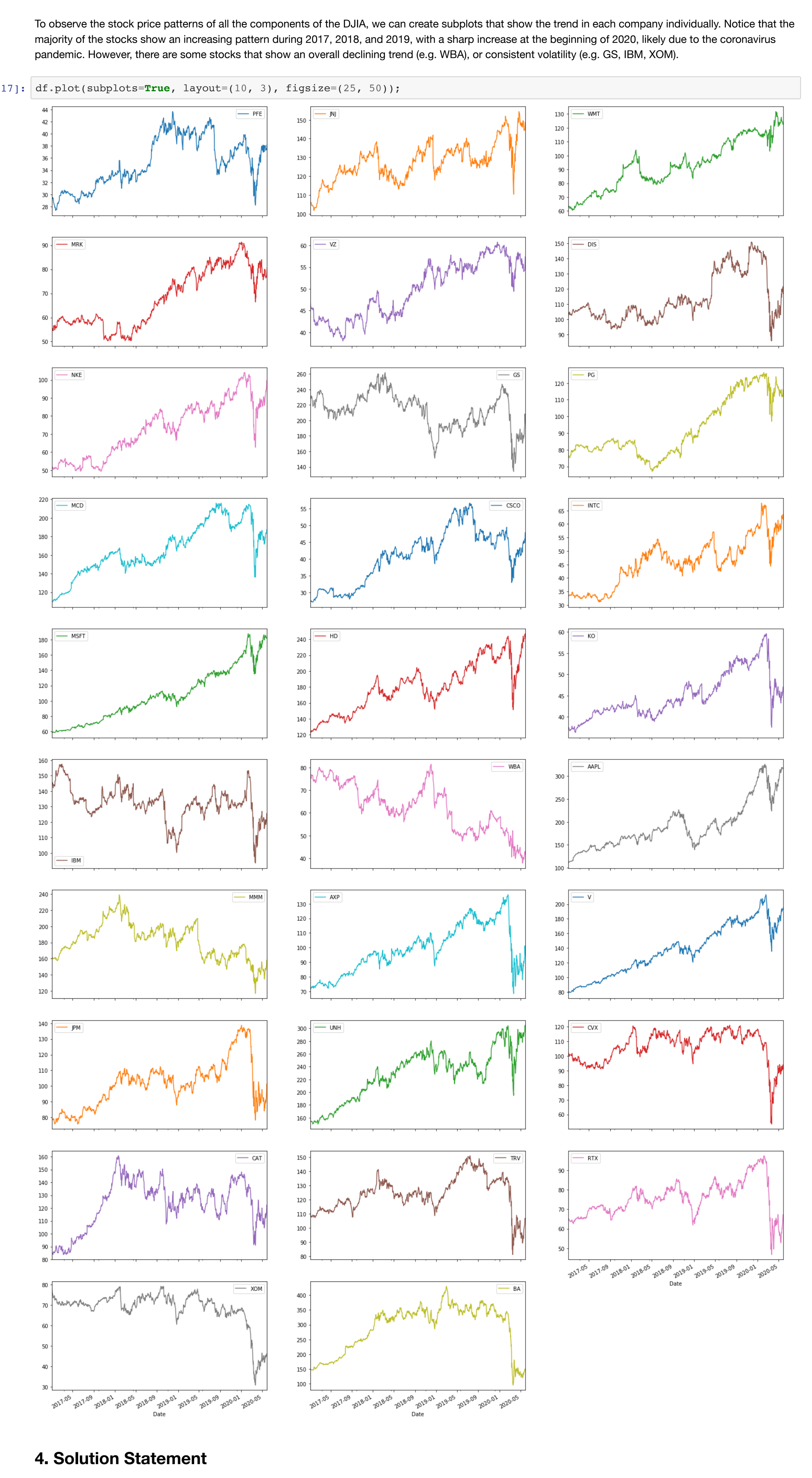
In [16]:
```
from matplotlib.dates import DateFormatter
import datetime

plt.rcParams['figure.figsize'] = [18, 10]

for group_info in [tech, financials]:
    fig, ax = plt.subplots(figsize=(18,7))

    group_name, group = group_info

    df[group[0]].plot(ax=ax, color='blue', title='Closing Price for {} Industry'.format(group_name))
    df[group[1]].plot(ax=ax, color='teal')
    df[group[2]].plot(ax=ax, color='green')
    df[group[3]].plot(ax=ax, color='purple')

    ax.xaxis.set_major_locator(mdates.YearLocator())
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y %b %d'))

    datemin = datetime.datetime(2017, 1, 1)
    datemax = datetime.datetime(2020, 5, 31)

    ax.set_xlim(datemin, datemax)
    plt.xticks(rotation=0)

    ax.set_xlabel("")
    ax.set_ylabel("Stock Price")
    ax.legend()

    for tick in ax.xaxis.get_major_ticks():
        tick.tick1line.set_markersize(5)
        tick.label1.set_horizontalalignment('center')
```





To observe the stock price patterns of all the components of the DJIA, we can create subplots that show the trend in each company individually. Notice that the majority of the stocks show an increasing pattern during 2017, 2018, and 2019, with a sharp increase at the beginning of 2020, likely due to the coronavirus pandemic. However, there are some stocks that show an overall declining trend (e.g. WBA), or consistent volatility (e.g. GS, IBM, XOM).

In [17]:
```
df.plot(subplots=True, layout=(10, 3), figsize=(25, 50));
```



4. Solution Statement

To solve my problem, I will use the first three years of data (from 2017-01-01 to 2019-12-31) to create a time series training set, and the rest of the data (from 2020-01-01 to 2020-05-31) to create a test set. I will then use the SageMaker platform to instantiate and train a DeepAR estimator. After deploying the model and creating a predictor, I will evaluate my model's predictive ability to assure its accuracy score is reasonable. Lastly, I will create a user interface that allows a person to input tickers and dates and receive stock price predictions.

5. Benchmark Model

As a benchmark, I will use a moving average model based on the last 10 days of stock data. Moving averages are common tools used in technical analysis to smooth out prices and create forecasts.

6. Evaluation Metrics

I will evaluate the benchmark and the DeepAR models by comparing their predictions to a known target. The data for the target will come from my test sample, the true stock prices for 2020. I will create a plot with 90 or 80% confidence intervals to see how well the models did.

7. Project Design

Given that I am interested in building a stock price predictor of the DJIA companies, I think I will have to train 30 models so that when I want to see the predictions of a certain company, I select the trained model for that company and output appropriate predictions.

References

1. Charles, A., & Darné, O. (2014). Large shocks in the volatility of the Dow Jones Industrial Average index: 1928–2013. Journal of Banking & Finance, 43, 188-199.

2. https://finance.yahoo.com/quote/%5EDJI/components?p=%5EDJI, accessed on 2020-06-08.

3. Shen, S., Jiang, H., & Zhang, T. (2012). Stock market forecasting using machine learning algorithms. Department of Electrical Engineering, Stanford University, Stanford, CA, 1-5.

4. Dash, R., & Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. The Journal of Finance and Data Science, 2(1), 42-57.

5. Jiang, F., Lee, J., Martin, X., & Zhou, G. (2019). Manager sentiment and stock returns. Journal of Financial Economics, 132(1), 126-149.

6. Kozak, S., Nagel, S., & Santosh, S. (2020). Shrinking the cross-section. Journal of Financial Economics, 135(2), 271-292.

7. https://www.reuters.com/article/us-usa-stocks-dowdupont/dow-jones-industrial-average-adds-dow-inc-removes-dowdupont-idUSKCN1R72TP