Machine Learning Engineer Nanodegree: Capstone Proposal

Daniela De la Parra | June 2020

DEFINITION

1. Project Overview

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.[i] The table below shows the price, change, % change, and volume of the 30 companies in the DJIA, as of June 8, 2020.[ii] 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.[iii] Some have constructed sophisticated trading algorithms based on technical indicators, which are mathematical indicators based on a stock's patterns in price, volume, etc.[iv] Others have used the textual content of managerial disclosures[v] or stochastic discount factors.[vi] 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 consists 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. An estimation stage 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. Metrics

I use a moving average model as my benchmark, and I evaluate this and the DeepAR models by comparing their predictions to a known target. I calculate the root-mean-square error (RMSE) and compare the performance of the models. I decided to use this metrics score because it is well suited for regression-type predictions rather than classification. That is, even if the predictions are not perfectly equal to the true targets, the closer they are to the truth the better the performance of the model.

I also plot the results for each company for ease of visualization. The data for the target will come from my test sample, the true stock prices of companies. I create a plot with 90 or 80% confidence intervals to see how well the models did.

ANALYSIS

4. Data Exploration

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

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/`. 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 of the pandas dataframes I construct. Here is what the data looks like for the first seven companies and their first six days of data:

Date	PFE	JNJ	WMT	MRK	VZ	DIS	NKE
2017-01-03	28.970671	105.502319	63.187412	54.672073	46.683941	101.584358	49.927341
2017-01-04	29.225266	105.329285	63.555534	54.653900	46.632622	102.886719	50.974297
2017-01-05	29.506187	106.431297	63.693573	54.635715	46.735260	102.829262	50.964691
2017-01-06	29.392067	105.921272	62.819294	54.781147	46.041534	104.361458	51.781124
2017-01-09	29.383287	105.903038	63.233440	55.535561	45.540134	103.767731	51.272057

My data includes 857 rows representing one *trading* day each. Stock exchanges are usually closed on weekends and during holidays. A tabulation of the years in my data shows that I have 251 days in 2017, 251 in 2018, 252 in 2019, and 103 in 2020.

Here is a description of the closing price of each company:

	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

	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

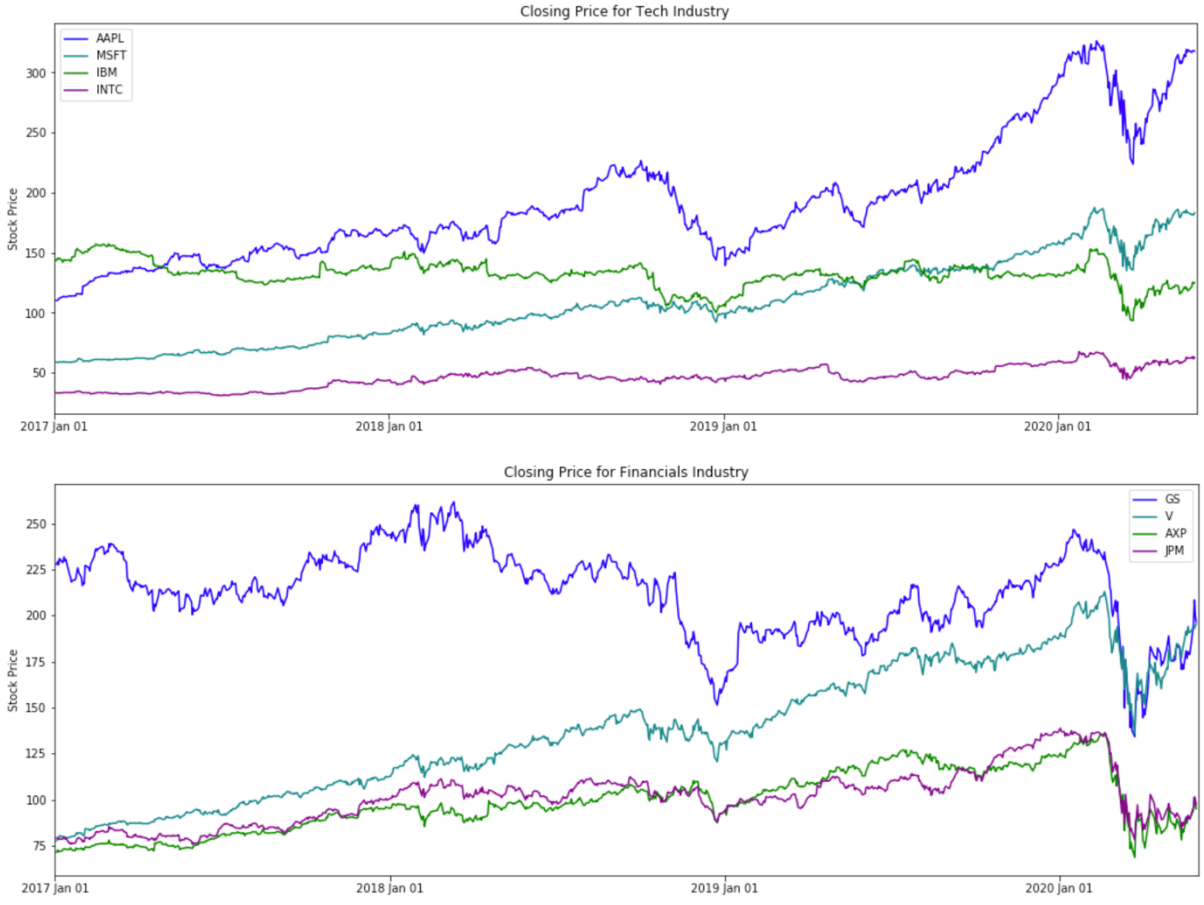
Notice that Dow Inc., with ticker *DOW*, only has 302 rows of data. The reason is that this company was created in early 2019, when DowDuPont Inc. split into three companies. One of them, Dow Inc., ended up replacing DowDuPont Inc. in the DJIA index.[vii] In fact, once we look into DOW's time series, we see that it indeed starts in March of 2019:

```
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
Name: DOW, dtype: float64
```

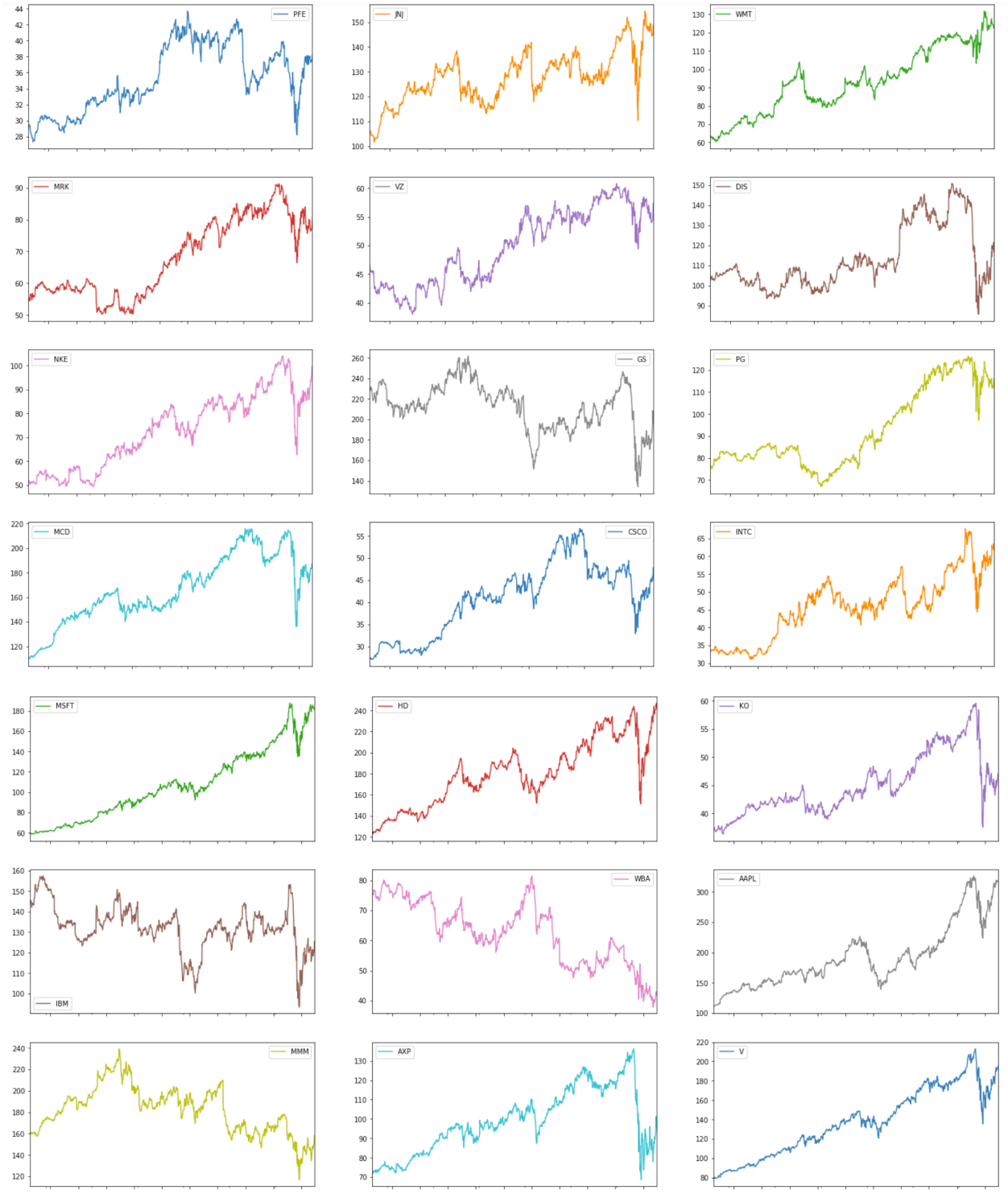
Given that training the model with only 302 might result in less accurate predictions, I decided to drop this company from my analysis. I thus have only 29 companies in my dataset.

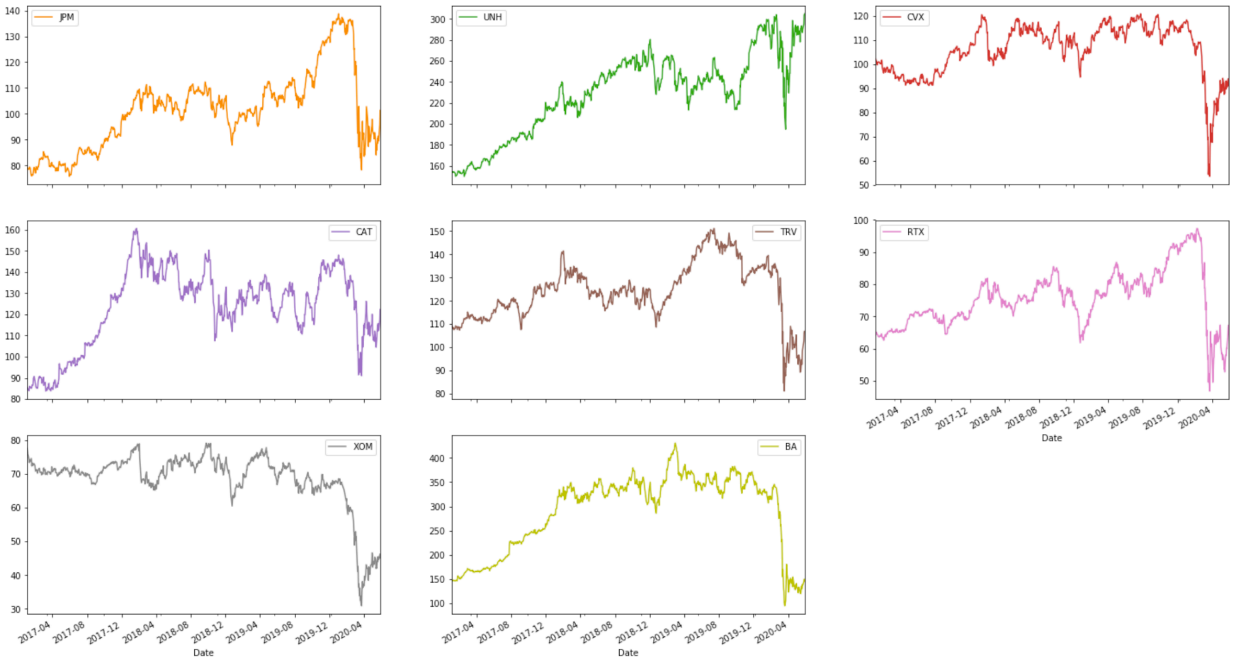
5. Exploratory Visualization

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.



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).

6. Algorithms and Techniques

To solve my problem, I use the first three years of data (from 2017-01-01 to 2019-12-31) to train and validate the model, and the rest of the data (from 2020-01-01 to 2020-05-31) to test the predictions. I use the SageMaker platform to instantiate and train a DeepAR estimator. After deploying the model and creating a predictor, I evaluate my model's predictive ability to assure its accuracy score is reasonable. I also assess my model by making predictions on tickers and dates after my training period. Lastly, I compare the performance of my DeepAR model to the benchmark performance, which comes from a moving average model.

7. Benchmark

As a benchmark, I use an ARIMA(0,1,10) model, which is a moving average model based on the last 10 days of stock data. Moving averages, and ARIMA models in general, are common tools used in technical analysis to smooth out prices and create forecasts. The equation for a q-th order integrated moving average model with constant (ARIMA(0,1,p)) is:[viii]

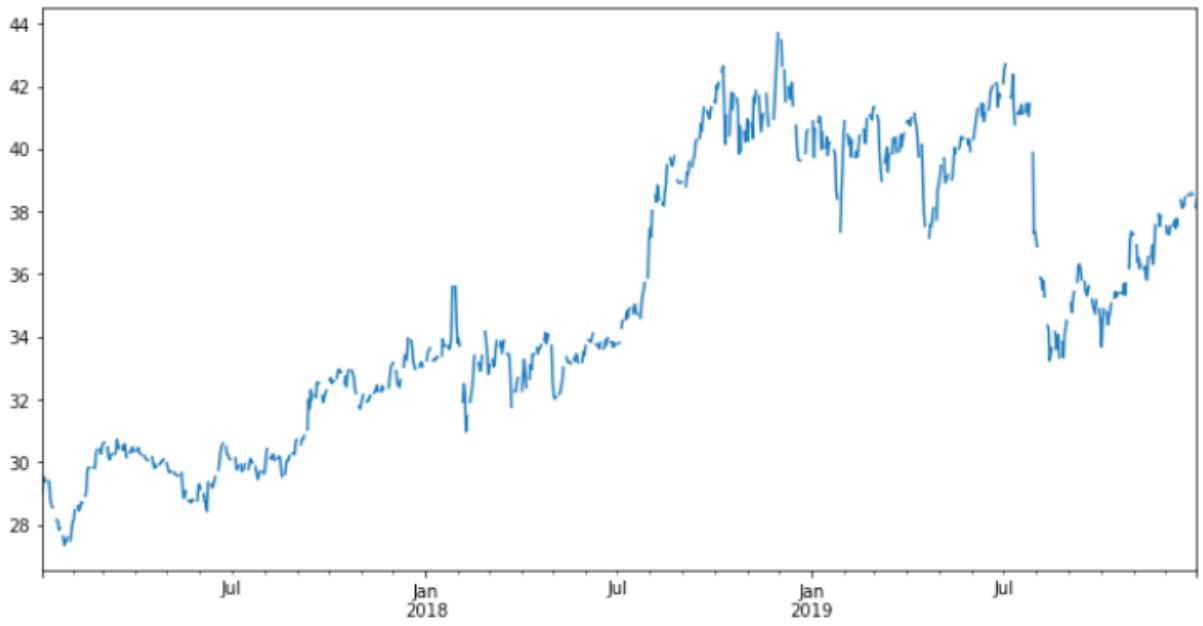
$$y_t = \mu + y_{t-1} + \varepsilon_t - \phi_1 \varepsilon_{t-1} - \phi_2 \varepsilon_{t-2} - \cdots - \phi_p \varepsilon_{t-p}$$

METHODOLOGY[ix]

8. **Data Preprocessing**

The input that DeepAR takes must be time series, so first I need to convert my data into time series objects. Each company's stock price data is converted into a time series. The function that I write sets 2017-01-03 as the starting date (the first trading day of 2017) and 2019-12-31 as the ending date for each time series. Given that there are only 251 or 252 days in each year, each time series has 754 observations.

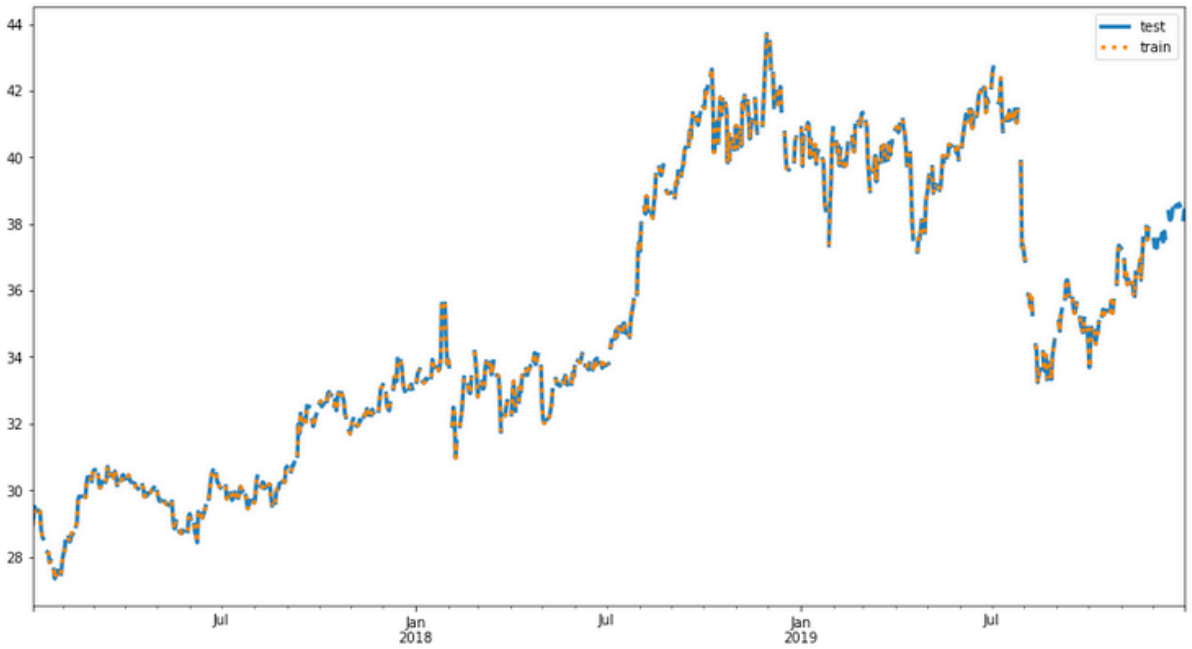
Here is the time series of company with ticker "PFE" to illustrate what my time series data looks like:



Notice that the line is not continuous, which reflects the fact that the stock price value for weekends and holidays is "NaN," given that the stock market is closed.

I next split my data into train and test samples for each time series. I do this using a prediction length of 30 days. I decided to only try to predict 30 days in advance because predicting stock prices is already extremely hard, so choosing a wider window would pose a substantial challenge to any model.

Here is what the train and test samples of the company "PFE" look in a graph:



Next, I need to encode each time series into a json object to pass it to the DeepAR estimator. I also need to save the files locally before uploading them to S3. The DeepAR documentation explains that I must encode missing values in the target as strings "NaN", so I make sure that my function to convert my data into json also cleans these missing values.

Once I convert my data into json objects and save the files locally, I upload them to S3. Here are the directories where my data is saved:

```
Training data is stored in: s3://sagemaker-us-east-2-551451770570/deepar-
stock-prices/train/train.json
Test data is stored in: s3://sagemaker-us-east-2-551451770570/deepar-stock-
prices/test/test.json
```

9. **Implementation**

I first instantiate a DeepAR estimator using SageMaker's 'forecasting-deepar' image_name. I set the frequency of my data to daily ('D') and the context_length to 30, which means that the model will look back to 30 days of data in order to make a prediction. I use the following hyperparameters:

- epochs: 50
- time_freq: 'D'
- prediction_length: 30
- context_length: 30
- num_cells: 50
- num_layers: 2
- mini_batch_size: 128
- learning_rate: 0.001
- early_stopping_patience: 10

Next, I launched a training job that took a little more than 6 minutes to train. Based on the output from the training job, the model achieved its higher performance during epoch 48, with a final loss of 1.7113642931 and a test RMSE score of 8.4:

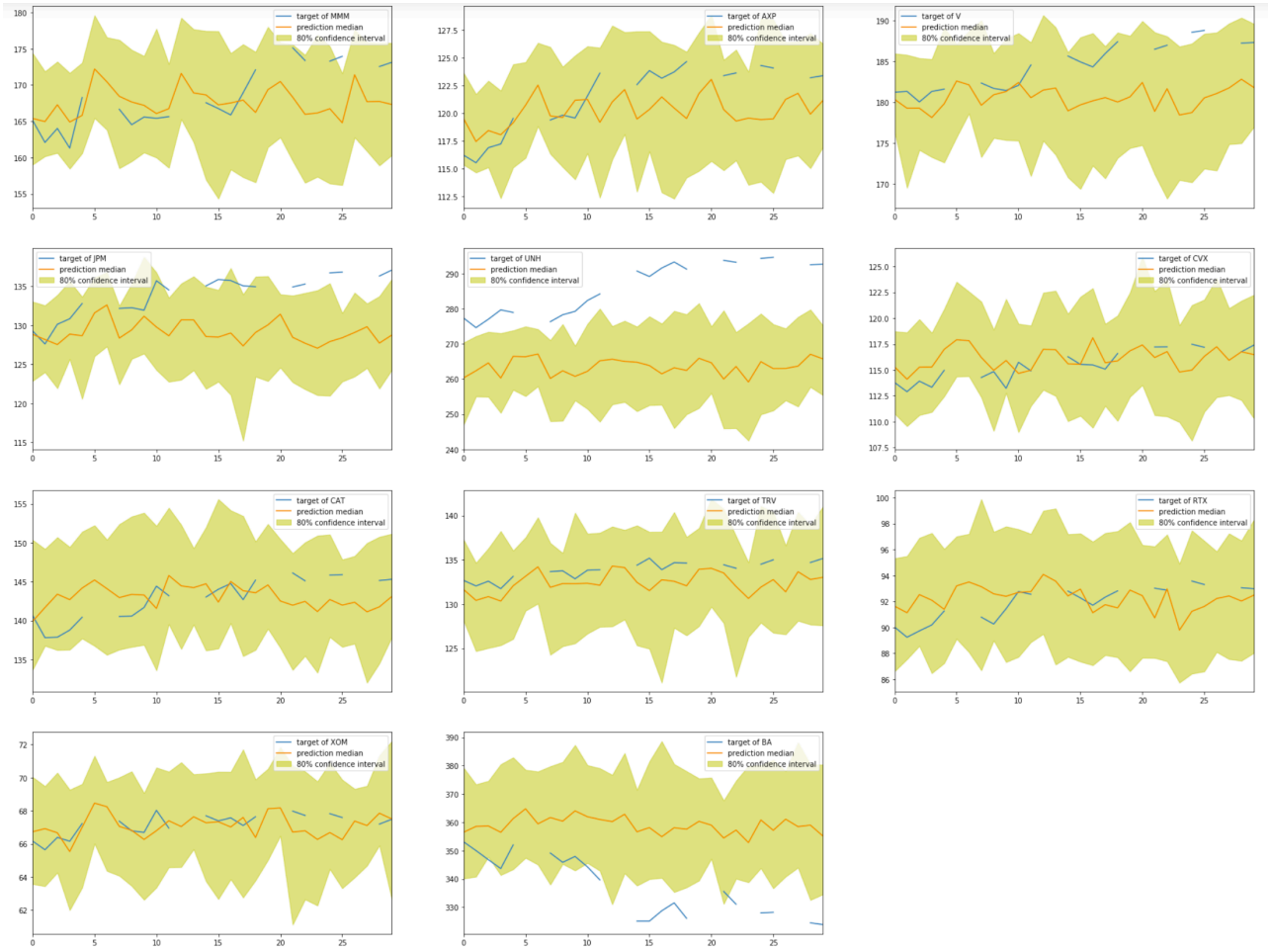
```
[06/17/2020 21:26:31 INFO 139680713221952] Final loss: 1.7113642931 (occurred at epoch 48)
[06/17/2020 21:26:31 INFO 139680713221952] #quality_metric: host=algo-1, train final_loss <loss>=1.7113642931
[06/17/2020 21:26:31 INFO 139680713221952] Worker algo-1 finished training.
[06/17/2020 21:26:31 WARNING 139680713221952] wait_for_all_workers will not sync workers since the kv store is not ru
nning distributed
[06/17/2020 21:26:31 INFO 139680713221952] All workers finished. Serializing model for prediction.
#metrics {"Metrics": {"get_graph.time": {"count": 1, "max": 231.6749095916748, "sum": 231.6749095916748, "min": 231.6
749095916748}}, "EndTime": 1592429191.395096, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm":
"AWS/DeepAR"}, "StartTime": 1592429191.162317}

[06/17/2020 21:26:31 INFO 139680713221952] Number of GPUs being used: 0
#metrics {"Metrics": {"finalize.time": {"count": 1, "max": 317.8129196166992, "sum": 317.8129196166992, "min": 317.81
29196166992}}, "EndTime": 1592429191.481193, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm": "
AWS/DeepAR"}, "StartTime": 1592429191.395172}

[06/17/2020 21:26:31 INFO 139680713221952] Serializing to /opt/ml/model/model_algo-1
[06/17/2020 21:26:31 INFO 139680713221952] Saved checkpoint to "/opt/ml/model/model_algo-1-0000.params"
#metrics {"Metrics": {"model.serialize.time": {"count": 1, "max": 12.67695426940918, "sum": 12.67695426940918, "min":
12.67695426940918}}, "EndTime": 1592429191.493976, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorit
hm": "AWS/DeepAR"}, "StartTime": 1592429191.481257}

[06/17/2020 21:26:31 INFO 139680713221952] Successfully serialized the model for prediction.
[06/17/2020 21:26:31 INFO 139680713221952] Evaluating model accuracy on testset using 100 samples
#metrics {"Metrics": {"model.bind.time": {"count": 1, "max": 0.041961669921875, "sum": 0.041961669921875, "min": 0.04
1961669921875}}, "EndTime": 1592429191.494699, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm":
"AWS/DeepAR"}, "StartTime": 1592429191.494019}

#metrics {"Metrics": {"model.score.time": {"count": 1, "max": 4215.453147888184, "sum": 4215.453147888184, "min": 421
5.453147888184}}, "EndTime": 1592429195.710105, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorith
m": "AWS/DeepAR"}, "StartTime": 1592429191.494761}

[06/17/2020 21:26:35 INFO 139680713221952] #test_score (algo-1, RMSE): 8.40109454475
```

I then create a predictor object to generate predictions of the 20%, 50%, and 90% quantiles. I converted my time series into json objects in order to pass them to the predictor object, and then decode the predictions obtained. Here are the 30-day predictions for company “PFE”:

	0.1	0.5	0.9
0	35.608791	37.070919	38.626434
1	35.298698	36.603771	37.797493
2	35.542381	36.730503	38.027782
3	35.306168	36.855656	38.912788
4	35.458916	36.777664	37.764320
5	35.696743	37.077068	38.226669
6	36.317539	37.406578	38.435787
7	35.793732	37.360847	38.659672
8	35.313656	36.827248	38.151382
9	35.363926	36.821445	39.708393
10	34.474869	36.660477	38.424328
11	34.862648	36.790257	38.486397
12	35.116611	36.820568	38.811501
13	35.501801	37.398048	38.837181
14	36.139465	37.168079	38.516228

	0.1	0.5	0.9
15	35.220612	36.615650	38.887512
16	34.526443	36.464958	37.869137
17	35.095222	36.740166	39.313869
18	35.424568	36.576149	38.705551
19	34.931950	36.680180	38.302914
20	35.579361	37.119644	39.339710
21	35.249630	36.697407	38.164543
22	34.915234	36.662601	38.910561
23	34.861099	36.345165	38.619911
24	34.796589	37.000286	39.426144
25	33.808361	36.113300	38.051640
26	34.571045	36.243752	38.844852
27	35.270405	37.085514	38.571205
28	36.458248	37.856567	38.857876
29	36.129871	37.270920	39.503891

I create subplots to visualize the predictions vs. the true stock prices of each companies. The yellow areas indicate the 80% quantiles.



Based on the graphs, we can see that the model made reasonable predictions only for a few companies (e.g., KO, WBA, TRV). For many time series, the model made predictions very far from the true stock prices (e.g., JNJ, MRK, NKE, APPL, UNH, BA).

For comparison purposes, I calculate the root-mean-square error (RMSE) for these results on the test set. The RMSE score is 8.212, which seems high.

10. **Refinement**

Given that the model I trained did not produce reasonable predictions for many companies, in this section I attempt to improve the model's predictive ability by automatically tuning its hyperparameters. The DeepAR documentation shows the following options for hyperparameter tuning:

Tunable Hyperparameters for the DeepAR Algorithm

Tune a DeepAR model with the following hyperparameters. The hyperparameters that have the greatest impact, listed in order from the most to least impactful, on DeepAR objective metrics are: `epochs`, `context_length`, `mini_batch_size`, `learning_rate`, and `num_cells`.

Parameter Name	Parameter Type	Recommended Ranges
mini_batch_size	IntegerParameterRanges	MinValue: 32, MaxValue: 1028
epochs	IntegerParameterRanges	MinValue: 1, MaxValue: 1000
context_length	IntegerParameterRanges	MinValue: 1, MaxValue: 200
num_cells	IntegerParameterRanges	MinValue: 30, MaxValue: 200
num_layers	IntegerParameterRanges	MinValue: 1, MaxValue: 8
dropout_rate	ContinuousParameterRange	MinValue: 0.00, MaxValue: 0.2
embedding_dimension	IntegerParameterRanges	MinValue: 1, MaxValue: 50
learning_rate	ContinuousParameterRange	MinValue: 1e-5, MaxValue: 1e-1

After instantiating a new estimator and setting default hyperparameters, I utilize SageMaker's HyperparameterTuner to choose ranges for the parameters I want to tune. Specifically, I choose the following ranges:

```
num_cells      : IntegerParameter(50, 70)
num_layers     : IntegerParameter(2, 6)
learning_rate  : ContinuousParameter(1e-5, 1e-2)
```

After the job is finished, I select the best training job. According to the training output, the following is the best configuration:

```
[06/17/2020 21:58:29 INFO 140527267129152] Final configuration: {u'dropout_rate': u'0.10', u'test_quantiles': u'[0.1,
0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]', u'_tuning_objective_metric': u'train:final_loss', u'num_eval_samples': u'10
0', u'learning_rate': u'0.002531068224511536', u'num_layers': u'3', u'epochs': u'50', u'embedding_dimension': u'10',
u'num_cells': u'62', u'_num_kv_servers': u'auto', u'mini_batch_size': u'128', u'likelihood': u'student-t', u'num_dyna
mic_feat': u'auto', u'cardinality': u'auto', u'_num_gpus': u'auto', u'prediction_length': u'30', u'time_freq': u'D',
u'context_length': u'30', u'_kvstore': u'auto', u'early_stopping_patience': u'10'}
```

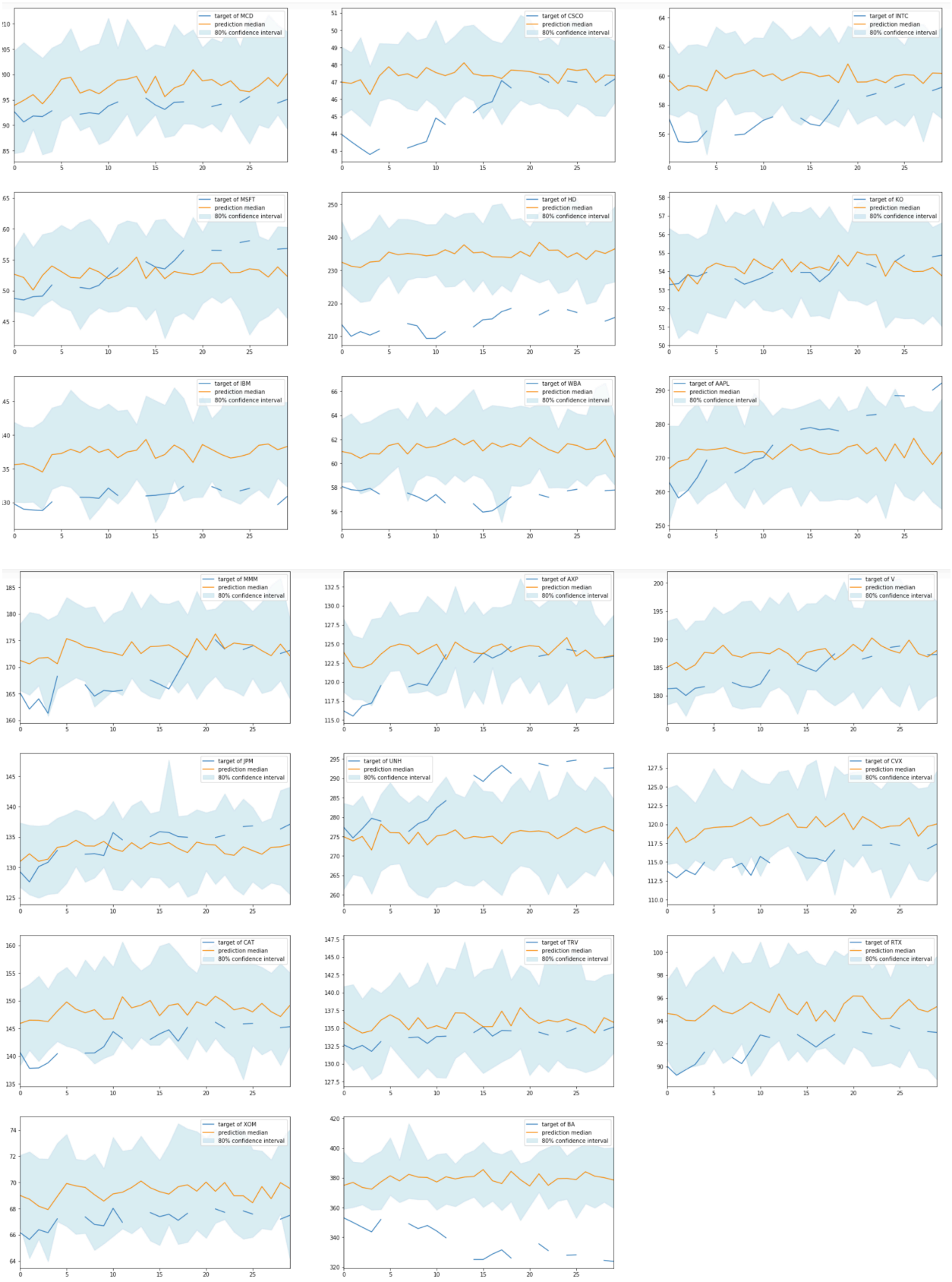
Out of the ranges of hyperparameters that I input in the hyperparameter_tuner, this is the configuration that achieved the highest performance:

```
learning_rate  : '0.002531068224511536'
num_layers     : '3'
num_cells      : '62'
```

The model achieved its higher performance during epoch 29, with a final loss of 1.73562215567 and a test RMSE score of 9.9:

```
[06/17/2020 22:01:20 INFO 140527267129152] Final loss: 1.73562215567 (occurred at epoch 29)
[06/17/2020 22:01:20 INFO 140527267129152] #quality_metric: host=algo-1, train final_loss <loss>=1.73562215567
[06/17/2020 22:01:20 INFO 140527267129152] Worker algo-1 finished training.
[06/17/2020 22:01:20 WARNING 140527267129152] wait_for_all_workers will not sync workers since the kv store is not ru
nning distributed
[06/17/2020 22:01:20 INFO 140527267129152] All workers finished. Serializing model for prediction.
#metrics {"Metrics": {"get_graph.time": {"count": 1, "max": 272.1381187438965, "sum": 272.1381187438965, "min": 272.1
381187438965}}, "EndTime": 1592431281.073443, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm":
"AWS/DeepAR"}, "StartTime": 1592431280.800696}

[06/17/2020 22:01:21 INFO 140527267129152] Number of GPUs being used: 0
#metrics {"Metrics": {"finalize.time": {"count": 1, "max": 373.1818199157715, "sum": 373.1818199157715, "min": 373.18
18199157715}}, "EndTime": 1592431281.17445, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm": "A
WS/DeepAR"}, "StartTime": 1592431281.073521}

[06/17/2020 22:01:21 INFO 140527267129152] Serializing to /opt/ml/model/model_algo-1
[06/17/2020 22:01:21 INFO 140527267129152] Saved checkpoint to "/opt/ml/model/model_algo-1-0000.params"
#metrics {"Metrics": {"model.serialize.time": {"count": 1, "max": 16.741037368774414, "sum": 16.741037368774414, "mi
n": 16.741037368774414}}, "EndTime": 1592431281.191288, "Dimensions": {"Host": "algo-1", "Operation": "training", "Al
gorithm": "AWS/DeepAR"}, "StartTime": 1592431281.17451}

[06/17/2020 22:01:21 INFO 140527267129152] Successfully serialized the model for prediction.
[06/17/2020 22:01:21 INFO 140527267129152] Evaluating model accuracy on testset using 100 samples
#metrics {"Metrics": {"model.bind.time": {"count": 1, "max": 0.03814697265625, "sum": 0.03814697265625, "min": 0.0381
4697265625}}, "EndTime": 1592431281.192038, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm": "A
WS/DeepAR"}, "StartTime": 1592431281.191338}

#metrics {"Metrics": {"model.score.time": {"count": 1, "max": 7513.462066650391, "sum": 7513.462066650391, "min": 751
3.462066650391}}, "EndTime": 1592431288.705459, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorith
m": "AWS/DeepAR"}, "StartTime": 1592431281.192084}

[06/17/2020 22:01:28 INFO 140527267129152] #test_score (algo-1, RMSE): 9.90236257569
```

I create a predictor based on this hypertuned model, and I evaluate the predictions for each company, compared to the true stock prices. Below are the plots of the results:


target of MCD
prediction median
80% confidence interval
target of CSCO
prediction median
80% confidence interval
target of INTC
prediction median
80% confidence interval
target of MSFT
prediction median
80% confidence interval
target of HD
prediction median
80% confidence interval
target of KO
prediction median
80% confidence interval
target of IBM
prediction median
80% confidence interval
target of WBA
prediction median
80% confidence interval
target of AAPL
prediction median
80% confidence interval
target of MMM
prediction median
80% confidence interval
target of AXP
prediction median
80% confidence interval
target of V
prediction median
80% confidence interval
target of JPM
prediction median
80% confidence interval
target of UNH
prediction median
80% confidence interval
target of CVX
prediction median
80% confidence interval
target of CAT
prediction median
80% confidence interval
target of TRV
prediction median
80% confidence interval
target of RTX
prediction median
80% confidence interval
target of XOM
prediction median
80% confidence interval
target of BA
prediction median
80% confidence interval

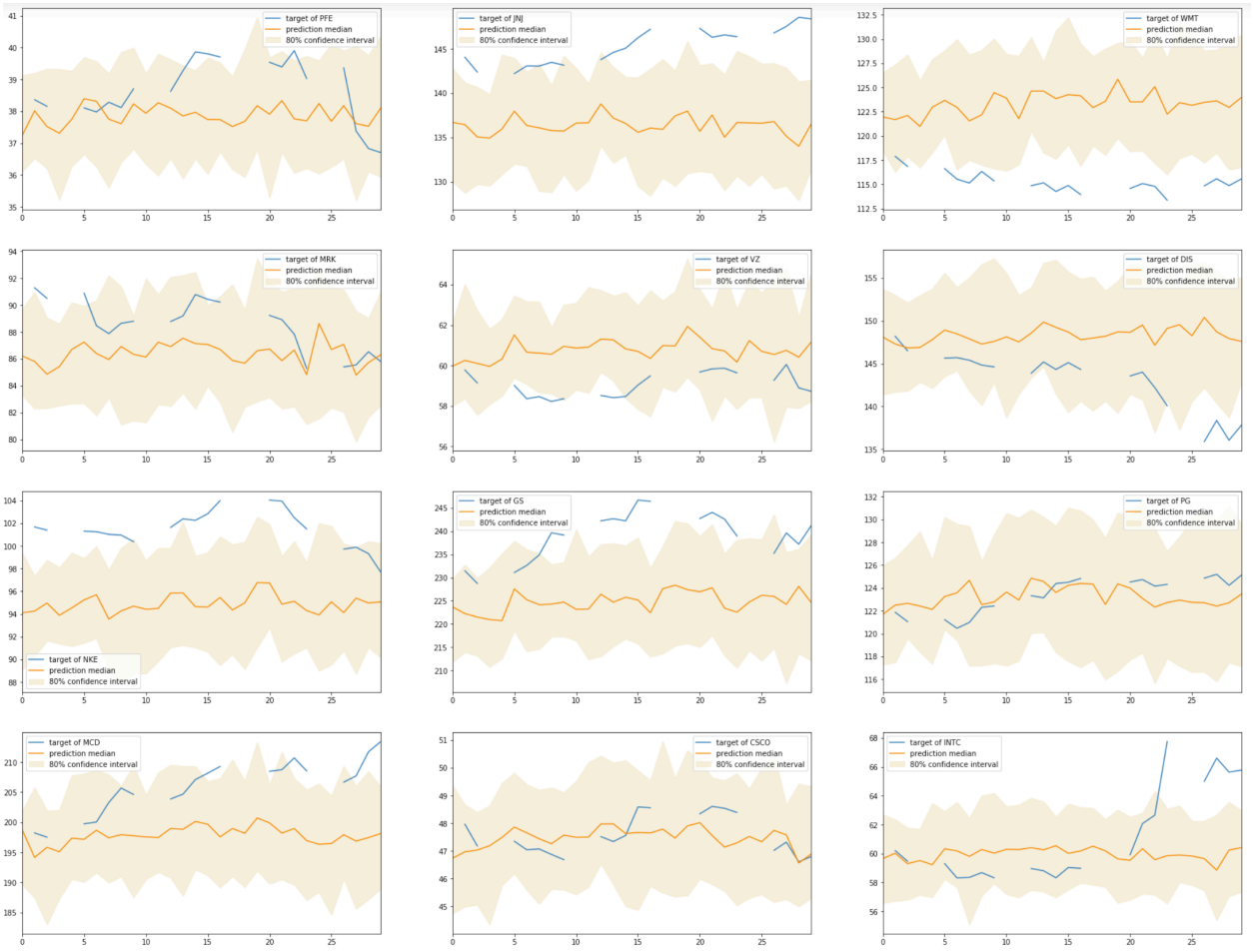
Based on the graphs, it does not look like the model improved much using the new configuration of hyperparameters. In fact, the RMSE of this refined model is 10.035, which is higher than the original model. This is surprising because I passed the same default parameters as the ones I used for the original model. The only difference is that the refined model was automatically tuned using the ranges of hyperparameters I passed.

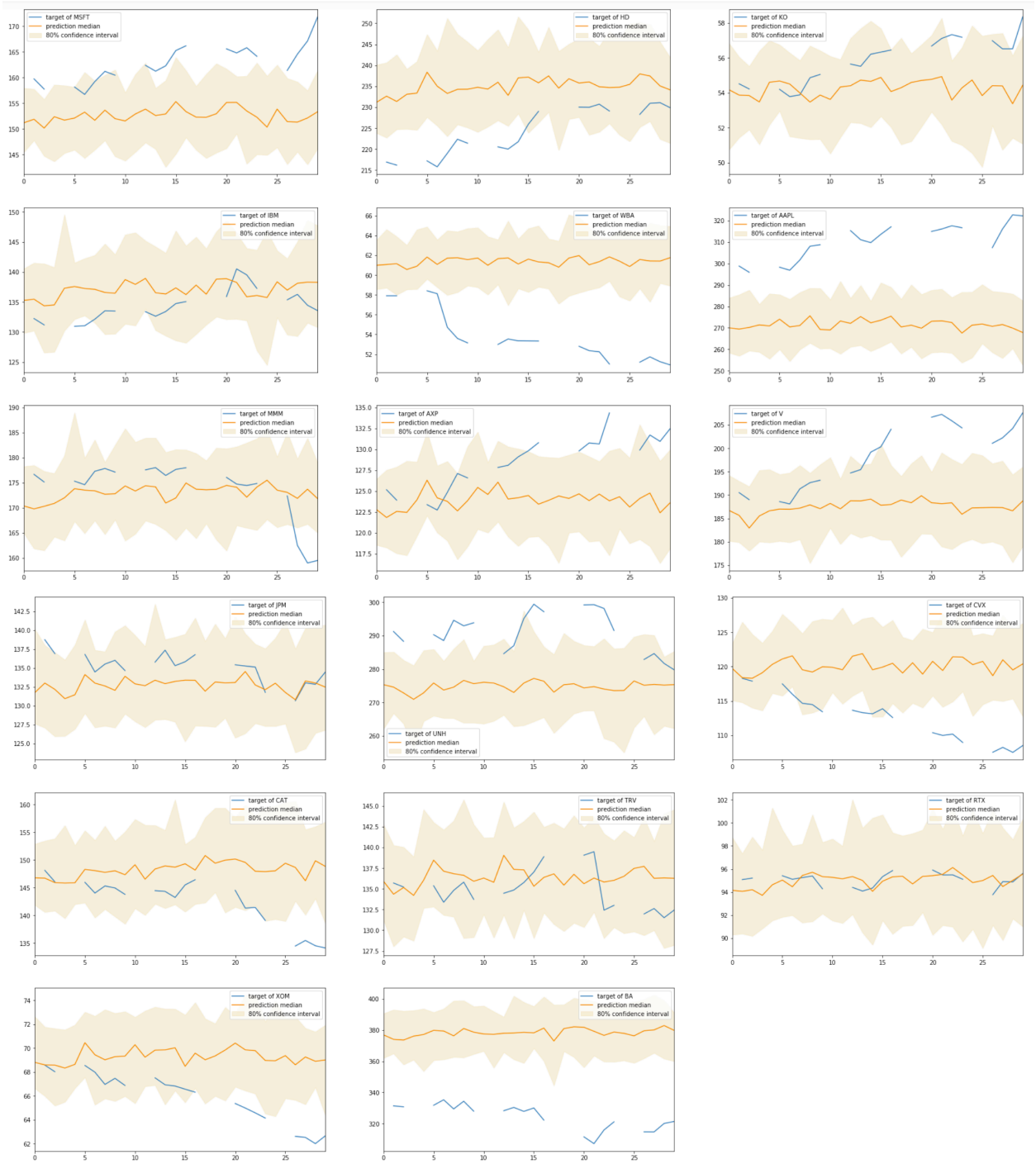
RESULTS

11. **Model Evaluation and Validation**

I evaluate and validate my model using the data in 2020 that the model has not seen. I first input each company's time series into json and send it to the predictor. After decoding the predictions, I compare them directly with the true stock prices of 2020.

The results are plotted against 80% confidence intervals and are shown below:





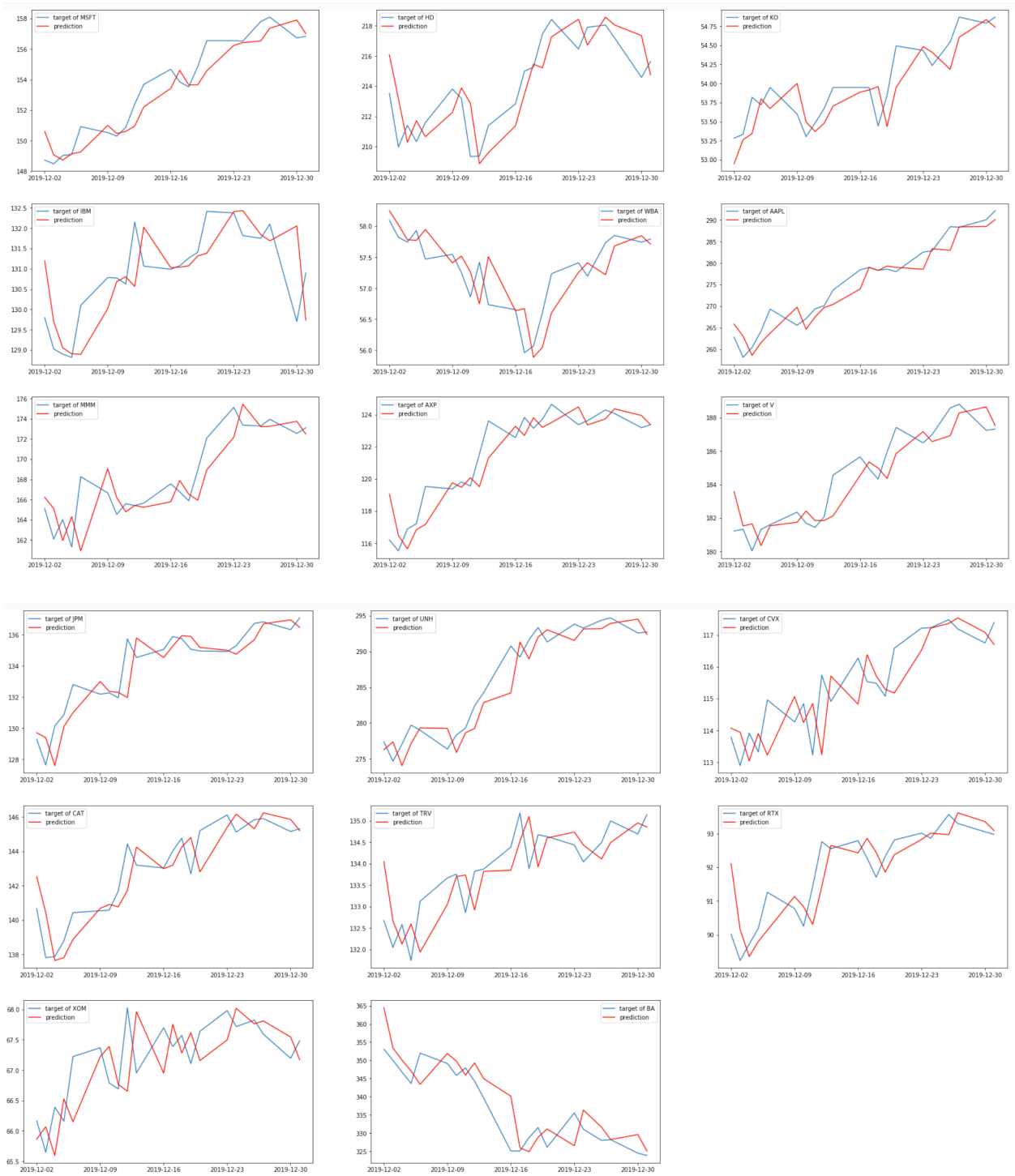
Not surprisingly, these predictions are worse than in the training set. The RMSE of the results is 14.634. These results reflect the challenge in predicting stock prices using only their past price as guidance. It also sheds light into the difficulty in finding appropriate hyperparameters. Perhaps training the model for more epochs and using mode layers could improve the results.

12. **Justification**

I compare my final results to my benchmark, which is an ARIMA(0,1,10) model, or integrated moving average, that uses the last 10 days of stock data. As I mentioned earlier, this is an appropriate model because moving averages are common tools used in technical analysis make predictions.[x]

The ARIMA model takes last 10 days of data and makes a prediction, which I then append to the history of data and use to make the following day's prediction.

The model trained for about 13 minutes, and it reached a performance of RMSE = 1.703. This is the best performing model, according to this metric. To illustrate the results, the predictions are plotted against the targets for each company:



It seems like the predictions of this model more closely follow the true stock prices. Intuitively, recent price movements are an indicator of where the market is heading and how investors are interpreting information, although their predictive ability is far from perfect.

CONCLUSION

In this project, I analyzed stock market data from the DJIA components. Specifically, I constructed machine learning models to try to predict stock prices 30 days in advance. An exploration analysis revealed that the stock prices of these companies are often volatile, as expected from stock market data. In order to solve the prediction problem, I implemented a DeepAR model that was trained using the first three years of data of my time series. The model reached its best performance during epoch 48 and reached an RMSE score of 8.4. Based on this score and on a visual examination of the plotted predictions against the true stock prices, I decided to attempt to improve the model's prediction by instantiating and training a refined model that used automatic hyperparameter tuning. I selected ranges for the number of layers, the number of cells per layer, and the learning rate. The predictive ability of the refined model, however, did not seem to improve much from the original model. The highest performance was attained during epoch 29, with an RMSE score of 9.9.

I also evaluated the model in the test set, using data from 2020, which the model had never seen. In this test data, the model reached an RMSE score of 14.6, which shows that it is harder to predict data on the test sample than the train sample. Lastly, I compared the results to an ARIMA(0,1,10) model based on the last 10 days of data. This benchmark model actually had the best performance in terms of RMSE, reaching a score of 1.7. The results from my analysis shed light into the challenge of predicting stock prices when we only use past stock prices information. Sophisticated traders and professional investors use not only stock prices, but also an extensive amount of data that includes corporate disclosures, industry trends, and macroeconomic indicators. I believe that my model would substantially benefit from incorporating other types of information, such as accounting data from companies' financial statements.

Overall, this project helped me put together the knowledge and skills I learned during Udacity's MLE Nanodegree program. I once again confirmed that data can sometimes be messier than it looks, but it can also reveal interesting results that you only find during hands-on practice.

REFERENCES

[i] 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.

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

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

[iv] 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.

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

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

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

[viii] https://people.duke.edu/~rnau/411arim.htm#ses

[ix] I learned to implement this model based on the code from my lessons during the Nannodegree program. Here are the links to the files I looked at: https://github.com/udacity/sagemaker-deployment/blob/master/Tutorials/Boston%20Housing%20-%20XGBoost%20(Hyperparameter%20Tuning)%20-%20High%20Level.ipynb , and https://github.com/udacity/ML_SageMaker_Studies/blob/master/Time_Series_Forecasting/Energy_Consumption_Solution.ipynb

[x] I learned to implement this model based on the code form this website: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/#:~:text=Autoregressive%20Integrated%20Moving%20Average%20Model,making%20skillful%20time%20series%20forecasts