پیش بینی روند فردای یک سهم با استفاده از عامل های هوشمند

۱. مقدمه

هدف از این پروژه، طراحی یک عامل هوشمند با توانایی پیش بینی روند فردای یک سهم در بورس ایران می باشد. به عبارت دقیق تر، ورودی مسئله، اطلاعات و دادگان یک سهم دلخواه به صورت روزانه تا یک تاریخ مشخص، و خروجی عدد ۱ در صورت صعودی بودن رشد آن سهم در روز بعد، و ۰ در صورت نزولی بودن رشد است.

بدین منظور، در این پروژه چندین عامل هوشمند طراحی شده و عملکرد آنها در پیش بینی روند سهم با یکدیگر مقایسه می‌شوند. عامل های طراحی شده عبارتند از رگرسیون خطی، Knn، Arima، Prophet و شبکه عصبی عمیق LSTM. در ادامه، ابتدا به نحوه جمع آوری دادگان و پیش پردازش های مورد نیاز پرداخته و سپس نحوه آموزش هرکدام از این مدل ها و عملکرد آنها را شرح می دهیم.

پیاده سازی این پروژه به زبان پایتون و با استفاده از کتابخانه های Numpy، Pandas و Sklearn انجام شده است. برای پیاده سازی مدل Arima از کتابخانه Pyramid، برای مدل Prophet از کتابخانه Fbprophet و برای شبکه عصبی LSTM از کتابخانه Keras بهره گرفته شده است. همچنین، مصورسازی دادگان با استفاده از کتابخانه Matplotlib انجام شده است. کدهای پروژه به صورت یک فایل ipynb یعنی Jupyter Notebook می باشد و در محیط Google Colab پیاده‌سازی شده است.

۲. جمع آوری دادگان

برای جمع آوری داده های مورد نیاز،از سایت بورس به نشانی http://www.tsetmc.com استفاده می‌شود. سهام در نظر گرفته شده برای این پروژه از نوع دارویی بوده و نام نماد مورد نظر برکت می باشد. با جستجوی نام نماد، سابقه آن به صورت داده های روزانه، از سال ۱۳۹۶ تا ۲۰ بهمن ۱۳۹۹، به صورت یک فایل csv دانلود می شود.

از این مجموعه دادگان برای آموزش عامل های هوشمند استفاده می شود، به صورتی که با توجه به داده ها تا ۱۹ بهمن ۱۳۹۹، قادر به پیش بینی روند سهم برای روز ۲۰ بهمن ۱۳۹۹ باشند. نمونه ای از این مجموعه دادگان در تصویر زیر آمده است:

	A	B	C	D	E	F	G	H	I	J	K	L
1	<TICKER>	<DTYYYYM	<FIRST>	<HIGH>	<LOW>	<CLOSE>	<VALUE>	<VOL>	<OPENINT	<PER>	<OPEN>	<LAST>
2	G.Barekat	20210207	21220	22160	21220	21950	4.3E+11	19611390	5166	D	21110	22160
3	G.Barekat	20210206	22000	22100	20810	21110	1.12E+12	52895611	11783	D	21900	20810
4	G.Barekat	20210203	22560	22560	20600	21900	3.37E+12	1.54E+08	37756	D	21490	22060
5	G.Barekat	20210202	21490	21490	21490	21490	1.05E+11	4901841	1984	D	20470	21490
6	G.Barekat	20210201	20000	20620	19640	20470	6.07E+11	29648144	8390	D	19640	20620
7	G.Barekat	20210131	19840	19840	18900	19640	1.64E+12	83592610	23106	D	18900	19840
8	G.Barekat	20210130	18220	19010	18150	18900	1.14E+12	60079425	13518	D	18110	19010
9	G.Barekat	20210127	18030	18800	18030	18110	1.15E+12	63688055	14150	D	18970	18030
10	G.Barekat	20210126	19230	19800	18900	18970	6.14E+11	32364143	7929	D	19890	18900
11	G.Barekat	20210125	19520	20900	19470	19890	1.47E+12	73904888	20571	D	20490	19470
12	G.Barekat	20210124	21510	21510	19470	20490	2.6E+12	1.27E+08	33132	D	20490	19500
13	G.Barekat	20210123	20490	20490	20490	20490	2.07E+11	10126354	1740	D	19520	20490
14	G.Barekat	20210120	18940	19650	18720	19520	6.37E+11	32635009	6723	D	18720	19650
15	G.Barekat	20210119	18540	19490	18540	18720	2.51E+12	1.34E+08	31853	D	19510	18800

داده ها شامل ۹۲۱ نمونه و دوازده ویژگی به شرح تصویر بالا می باشند. از کتابخانه Pandas برای خواندن فایل csv دادگان و انجام اکثر پیش پردازش ها استفاده شده است. ابتدا داده ها را خوانده و آن را به صورت یک دیتا فریم ذخیره می کنیم. همچنین، داده query یا همان نمونه مورد پرسش، که شامل اطلاعات ۲۰ بهمن می باشد، به صورت یک دیتا فریم جداگانه ذخیره می شود. از این دیتا فریم، جهت ارزیابی نهایی مدل ها و سنجش عملکرد آن ها استفاده می شود. به عبارت دیگر، عامل ها باید قادر به پیش بینی صعودی یا نزولی بودن سهم در روز query باشند. نمونه ای از دیتا فریم های مذکور در ادامه آمده است:

	<TICKER>	<DTYYYYMMDD>	<FIRST>	<HIGH>	<LOW>	<CLOSE>	<VALUE>	<VOL>	<OPENINT>	<PER>	<OPEN>	<LAST>
0	G.Barekat.Pharm	20210207	21220.0	22160.0	21220.0	21950.0	430478691670	19611390	5166	D	21110.0	22160.0
1	G.Barekat.Pharm	20210206	22000.0	22100.0	20810.0	21110.0	1116507854150	52895611	11783	D	21900.0	20810.0
2	G.Barekat.Pharm	20210203	22560.0	22560.0	20600.0	21900.0	3368892543720	153814430	37756	D	21490.0	22060.0
3	G.Barekat.Pharm	20210202	21490.0	21490.0	21490.0	21490.0	105340563090	4901841	1984	D	20470.0	21490.0
4	G.Barekat.Pharm	20210201	20000.0	20620.0	19640.0	20470.0	606821827820	29648144	8390	D	19640.0	20620.0

	<TICKER>	<DTYYYYMMDD>	<FIRST>	<HIGH>	<LOW>	<CLOSE>	<VALUE>	<VOL>	<OPENINT>	<PER>	<OPEN>	<LAST>
0	G.Barekat.Pharm	20210208	23040	23040	22000	22810	1.809780e+12	79334852	23633	D	21950	23040

برای پیش پردازش داده ها، ابتدا وجود نویز، مانند داده های با فرمت غلط، و یا missing value بررسی می شود تا در صورت لزوم تصحیح شوند:

```
df.any().isnull()
```

```
<TICKER>        False
<DTYYYYMMDD>    False
<FIRST>         False
<HIGH>          False
<LOW>           False
<CLOSE>         False
<VALUE>         False
<VOL>           False
<OPENINT>       False
<PER>           False
<OPEN>          False
<LAST>          False
dtype: bool
```

پاسخ این سوال منفی می باشد پس به سراغ گام بعدی می رویم. همانطور که در تصویر بالا مشاهده می‌شود، ستون تاریخ باید اصلاح شود، به گونه ای که فرمت تاریخ های موجود به الگوی yyyy-mm-dd تبدیل شود. بنابراین در گام بعدی این فرمت اصلاح می شود. این کار را برای داده query نیز تکرار میکنیم. همچنین، ستون های Per و Ticker را از دیتا فریم ها حذف می کنیم:

	<FIRST>	<HIGH>	<LOW>	<CLOSE>	<VALUE>	<VOL>	<OPENINT>	<OPEN>	<LAST>	<DATE>
0	21220.0	22160.0	21220.0	21950.0	4.304787e+11	19611390	5166	21110.0	22160.0	2021-02-07
1	22000.0	22100.0	20810.0	21110.0	1.116508e+12	52895611	11783	21900.0	20810.0	2021-02-06
2	22560.0	22560.0	20600.0	21900.0	3.368893e+12	153814430	37756	21490.0	22060.0	2021-02-03
3	21490.0	21490.0	21490.0	21490.0	1.053406e+11	4901841	1984	20470.0	21490.0	2021-02-02
4	20000.0	20620.0	19640.0	20470.0	6.068218e+11	29648144	8390	19640.0	20620.0	2021-02-01

	<FIRST>	<HIGH>	<LOW>	<CLOSE>	<VALUE>	<VOL>	<OPENINT>	<OPEN>	<LAST>	<DATE>
0	23040	23040	22000	22810	1.809780e+12	79334852	23633	21950	23040	2021-02-08

در قدم بعد، برای گرفتن خروجی ۱ یا ۰ از عامل ها، نیازمند به برچسب زدن به دادگان هستیم، بدین صورت که اگر در داده های موجود برای هر نمونه، روند رشد سهم در آن روز مثبت بود، به آن برچسب ۱ و در غیر این صورت برچسب ۰ تعلق بگیرد. پس یک ستون جدید با اسم Profit یا همان سود ایجاد کرده، و با کم کردن مقدار ستون Open از مقدار ستون Close، مقدار به دست آمده را به عنوان سود آن روز در نظر می گیریم. سپس، یک ستون جدید دیگر با نام Label یا همان برچسب ایجاد می کنیم. برای هر نمونه اگر مقدار

به دست آمده در ستون Profit بزرگ تر و یا مساوی ۰ بود، در ستون Label به آن مقدار ۱ و در غیر این مقدار ۰ نسبت می دهیم. این کار را برای داده query نیز تکرار می کنیم:

	<FIRST>	<HIGH>	<LOW>	<CLOSE>	<VALUE>	<VOL>	<OPENINT>	<OPEN>	<LAST>	<DATE>	<PROFIT>	label
0	21220.0	22160.0	21220.0	21950.0	4.304787e+11	19611390	5166	21110.0	22160.0	2021-02-07	840.0	1
1	22000.0	22100.0	20810.0	21110.0	1.116508e+12	52895611	11783	21900.0	20810.0	2021-02-06	-790.0	0
2	22560.0	22560.0	20600.0	21900.0	3.368893e+12	153814430	37756	21490.0	22060.0	2021-02-03	410.0	1
3	21490.0	21490.0	21490.0	21490.0	1.053406e+11	4901841	1984	20470.0	21490.0	2021-02-02	1020.0	1
4	20000.0	20620.0	19640.0	20470.0	6.068218e+11	29648144	8390	19640.0	20620.0	2021-02-01	830.0	1
5	19840.0	19840.0	18900.0	19640.0	1.641516e+12	83592610	23106	18900.0	19840.0	2021-01-31	740.0	1

	<FIRST>	<HIGH>	<LOW>	<CLOSE>	<VALUE>	<VOL>	<OPENINT>	<OPEN>	<LAST>	<DATE>	<PROFIT>	label
0	23040	23040	22000	22810	1.809780e+12	79334852	23633	21950	23040	2021-02-08	860	1

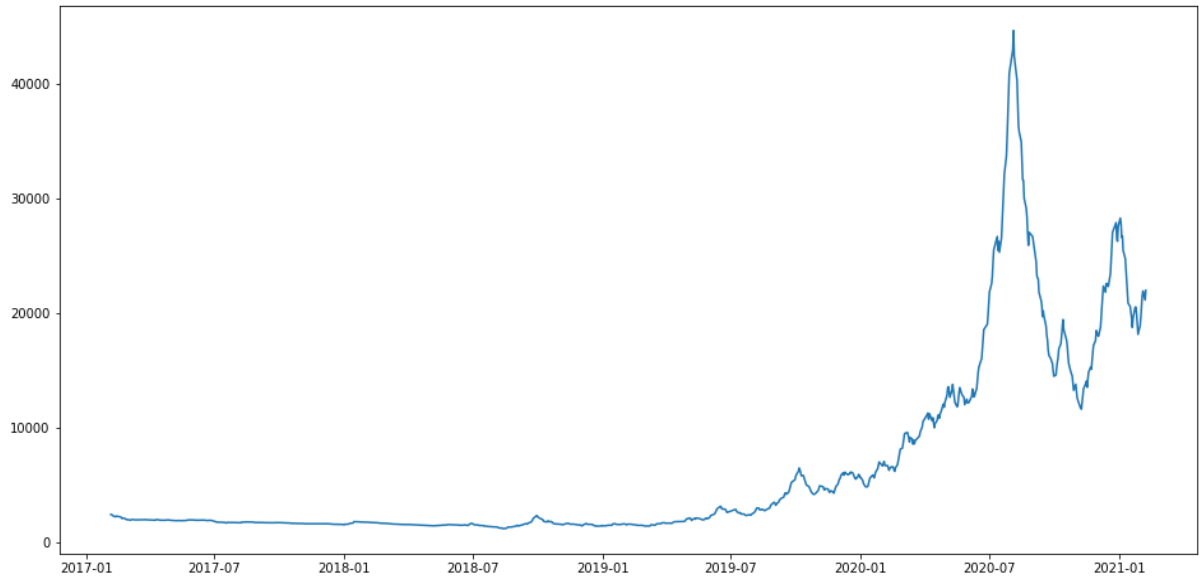
حال، با استفاده از برچسب های به دست آمده که در واقع دو کلاس مثبت و منفی را به ما نشان می دهند، تعداد نمونه های موجود در هر کلاس را بررسی می کنیم تا ببینیم آیا داده ها imbalance هستند یا خیر. دانستن این امر مهم است زیرا در در هر مرحله از آموزش عامل های هوشمند، برای ارزیابی دقت مدل ها، در صورت balance نبودن دادگان، علاوه بر محاسبه accuracy score، نیاز به محاسبه AUC-ROC score داریم، زیرا دقت به دست آمده توسط accuracy score به تنهایی درک درستی از چگونگی عملکرد عامل ها را به دست نمی دهد:

```
df['label'].value_counts()
```

```
0    475
1    446
Name: label, dtype: int64
```

همان طور که در تصویر بالا مشاهده می کنیم، تعداد داده های دو کلاس تقریباً برابر بوده و حدود ۴۰۰ نمونه در هر کلاس می‌باشد. بنابراین مشکل imbalance بودن داده وجود ندارد و دقت accuracy score برای ارزیابی عملکرد مدل ها کافی می باشد.

در گام آخر پیش پردازش، با استفاده از کتابخانه matplotlib داده ها را به عنوان یک سری زمانی، مصورسازی می کنیم:



نمودار بالا به وضوح روند صعودی و نزولی بودن سهم در هر روز را به تصویر می‌کشد و دید خوبی از چگونگی قرارگیری داده ها و ارتباط آنها با یکدیگر به دست می دهد.

ویژگی های مورد نیاز در گام های بعدی این پروژه، ستون های Date، Close، Open، و Label می باشد، پس آن ها را قبل از شروع آموزش مدل ها از داده های اولیه جدا می کنیم:

	<DATE>	<CLOSE>	<OPEN>	label
0	2017-02-05 00:00:00	2401	1000	1
1	2017-02-06 00:00:00	2380	2401	0
2	2017-02-07 00:00:00	2355	2380	0
3	2017-02-08 00:00:00	2282	2355	0
4	2017-02-11 00:00:00	2209	2282	0
...	...	...	...	...
916	2021-02-01 00:00:00	20470	19640	1
917	2021-02-02 00:00:00	21490	20470	1
918	2021-02-03 00:00:00	21900	21490	1
919	2021-02-06 00:00:00	21110	21900	0
920	2021-02-07 00:00:00	21950	21110	1

921 rows × 4 columns

	<DATE>	<CLOSE>	<OPEN>	label
0	2021-02-08 00:00:00	22810	21950	1

در قسمت بعد، با داشتن داده های پیش پردازش شده، به آموزش و ارزیابی هر کدام از عوامل هوشمند ذکر شده می پردازیم و آنها را با یکدیگر مقایسه می کنیم.

۳. رگرسیون خطی

اولین مدل استفاده شده رگرسیون خطی می باشد که در کاربرد های یادگیری ماشین مانند پیش بینی بسیار مفید می باشد. این مدل تعدادی از ویژگی ها را به عنوان متغیرهای مستقل در نظر گرفته و یک ویژگی هدف را به عنوان متغیر وابسته پیش بینی می کند.

در این پروژه، ویژگی هایی تحت عنوان متغیرهای مستقل در اختیار نداریم و صرفاً میتوانیم از ستون تاریخ یا Date استفاده کنیم. بنابراین، تعدادی ویژگی را از آن استخراج کرده و آن ها را به عنوان متغیرهای مستقل مورد استفاده قرار می دهیم.

بدین منظور، از تابعی به نام date_part() بهره برده، و از ستون اولیه Date ستون هایی جدید مانند سال، هفته، روز، ... را استخراج میکنیم. این کار را برای داده query نیز انجام می دهیم. ساختار تابع date_part()، و ستون های استخراج شده در کنار ستون های اولیه آمده است:

```
def add_datepart(df, fldname, drop=True, time=False):
    "Helper function that adds columns relevant to a date."
    fld = df[fldname]
    fld_dtype = fld.dtype
    if isinstance(fld_dtype, pd.core.dtypes.dtypes.DatetimeTZDtype):
        fld_dtype = np.datetime64

    if not np.issubdtype(fld_dtype, np.datetime64):
        df[fldname] = fld = pd.to_datetime(fld, infer_datetime_format=True)
    targ_pre = re.sub('[Dd]ate$', '', fldname)
    attr = ['Year', 'Month', 'Week', 'Day', 'Dayofweek', 'Dayofyear',
            'Is_month_end', 'Is_month_start', 'Is_quarter_end', 'Is_quarter_start', 'Is_year_end', 'Is_year_start']
    if time: attr = attr + ['Hour', 'Minute', 'Second']
    for n in attr: df[targ_pre + n] = getattr(fld.dt, n.lower())
    df[targ_pre + 'Elapsed'] = fld.astype(np.int64) // 10 ** 9
    if drop: df.drop(fldname, axis=1, inplace=True)
```

	Close	Open	Label	Year	Month	Week	Day	Dayofweek	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end	Is_year_start
0	2401	1000	1	2017	2	5	5	6	36	False	False	False	False	False	False
1	2380	2401	0	2017	2	6	6	0	37	False	False	False	False	False	False
2	2355	2380	0	2017	2	6	7	1	38	False	False	False	False	False	False
3	2282	2355	0	2017	2	6	8	2	39	False	False	False	False	False	False
4	2209	2282	0	2017	2	6	11	5	42	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
916	20470	19640	1	2021	2	5	1	0	32	False	True	False	False	False	False
917	21490	20470	1	2021	2	5	2	1	33	False	False	False	False	False	False
918	21900	21490	1	2021	2	5	3	2	34	False	False	False	False	False	False
919	21110	21900	0	2021	2	5	6	5	37	False	False	False	False	False	False
920	21950	21110	1	2021	2	5	7	6	38	False	False	False	False	False	False

921 rows × 15 columns

	Close	Open	Label	Year	Month	Week	Day	Dayofweek	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end	Is_year_start
0	22810	21950	1	2021	2	6	8	0	39	False	False	False	False	False	False

حال، داده ها را به دو مجموعه train و validation تقسیم می کنیم. نمونه های موجود، از ابتدا و تا انتهای سال ۲۰۱۹، به عنوان داده ی train جدا شده و بقیه نمونه ها، از ابتدای سال ۲۰۲۰ تا آخرین روز موجود، به عنوان داده validation کنار گذاشته می شوند. از داده‌های train برای آموزش مدل استفاده کرده، و سپس با استفاده از داده های validation مدل را ارزیابی می‌کنیم، و در انتها داده query را به مدل می دهیم تا نتیجه آن روز را پیش‌بینی کند. در نتیجه، از ۹۲۱ نمونه موجود، ۶۶۳ نمونه به عنوان داده train، و ۲۵۸ نمونه باقی مانده به عنوان داده validation استفاده می شود.

در این مدل، ستون های Open، Close، و Label از مجموعه های train و validation حذف شده و صرفاً ستون های مربوط به تاریخ که در بالا توضیح داده شد، استفاده می شوند.

ستون Close را به عنوان متغیر وابسته یا هدف در نظر گرفته و علاقه‌مند به پیش‌بینی مقادیر آن برای داده های ارزیابی می باشیم. این امر در بقیه مدل ها که در ادامه می آیند نیز صدق می کند. در نتیجه، می توان مقادیر ستون Open را از مقادیر پیش بینی شده ی ویژگی Close کم کرده و مقدار سود پیش بینی شده یا Predicted Profit را به دست آورد. سپس، از روی سود پیش بینی شده، بر چسب های پیش بینی شده

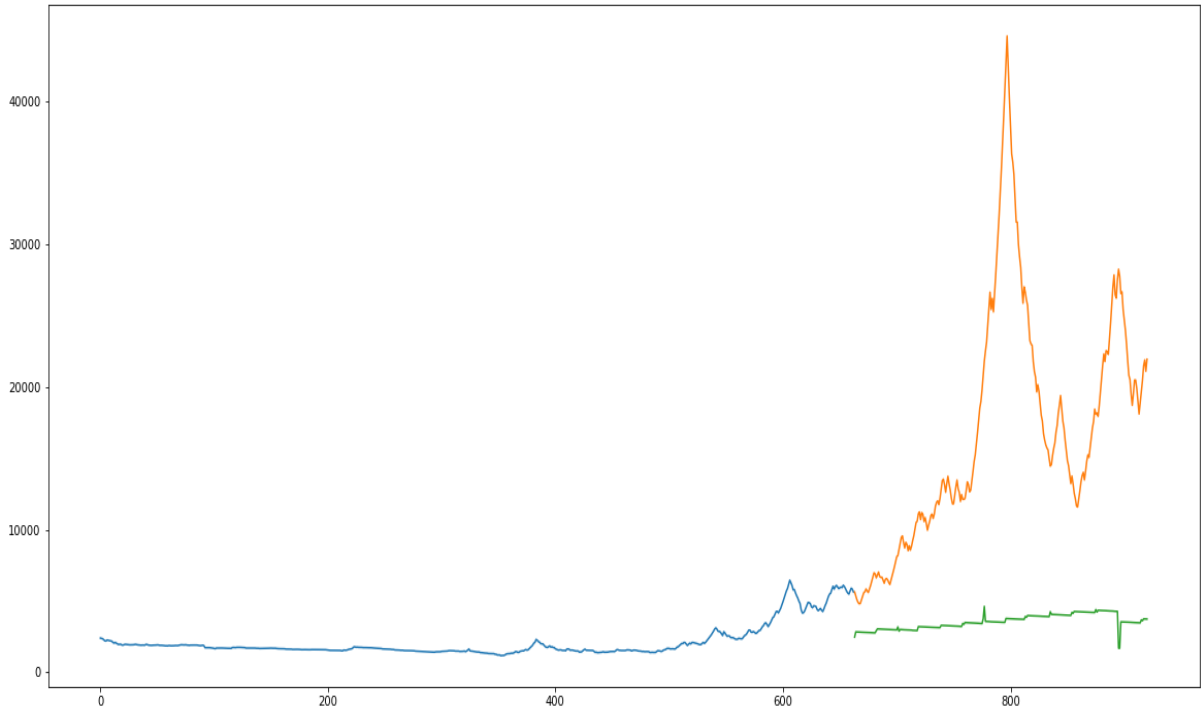
یا Predicted Labels را محاسبه کرده، و این مقادیر را با برچسب های اصلی یعنی ستون Labels مقایسه کرده و دقت دسته بندی مدل ها را ارزیابی می کنیم.

مدل رگرسیون خطی را با کتابخانه Sklearn ساخته، آن را با مجموعه train آموزش داده و سپس متغیر هدف یعنی ستون Close در مجموعه validation را با آن پیش بینی می‌کنیم.

برای اندازه گیری خطای مدل، از خطای RMSE استفاده می کنیم. این خطا، تفاوت میان مقدار پیش‌بینی شده برای ویژگی Close و مقدار حقیقی را اندازه گیری می کند. هر چه این مقدار کمتر باشد، عملکرد مدل بهتر است. خطای RMSE به دست آمده برای مدل رگرسیون خطی در زیر آمده است:

```
15680.11523886393
```

برای درک بهتر عملکرد این مدل، مقادیر Close پیش بینی شده در مقایسه با مقادیر واقعی رسم شده است:



قسمت آبی نمودار مقادیر Close مربوط به مجموعه train، قسمت نارنجی مقادیر مجموعه validation، و قسمت سبز مقادیر پیش بینی شده توسط مدل را نشان می دهد. همانطور که می بینیم، اختلاف زیادی میان مقادیر پیش بینی شده و مقادیر واقعی وجود دارد که نشان از عملکرد بد این مدل می دهد.

حال، برچسب های پیش بینی شده را محاسبه کرده و آنها را با برچسب های واقعی مقایسه می کنیم. بدین منظور، confusion matrix و accuracy score مدل محاسبه شده است:

```
[[115   0]
 [143   0]]
```

```
0.44573643410852715
```

در آخر، داده query را به مدل داده و مقدار Close آن را پیشگویی می کنیم:

Predictions	Predicted Profit	Predicted Labels
3728.633077	-18221.4	0

همانطور که در تصویر مشاهده می شود، برچسب واقعی داده ۱ بوده ولی مدل آن را ۰ پیش‌بینی کرده است.
در قسمت بعدی، یک مدل رگرسیونی دیگر مطرح شده و نتایج آن با این مدل مقایسه می شود.

۴. Knn

یکی دیگر از مدل‌های کاربردی یادگیری ماشین، Knn می باشد. این مدل با استفاده از متغیرهای مستقل، شباهت میان داده های جدید و داده های قبلی یا همان همسایه‌های هر داده جدید را می‌یابد. بنابراین، برای پیش بینی مقدار متغیر وابسته برای هر نمونه جدید، از مقادیر آن ویژگی وابسته در همسایه های داده جدید استفاده کرده و یک تخمین انجام می دهد.
قبل از استفاده از این مدل، لازم است داده ها حتماً نرمال سازی شوند و مقادیر آنها بین ۰ و ۱ برده شود تا توزیع داده ها نرمال باشد. بدین منظور از Minmax Scaler در کتابخانه Sklearn استفاده شده و داده های train و validation نرمال سازی می شوند. این کار برای داده query نیز انجام می شود:

	Year	Month	Week	Day	Dayofweek	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end	Is_year_start
0	0.0	0.090909	0.078431	0.133333	1.000000	0.096154	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.090909	0.098039	0.166667	0.000000	0.098901	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.090909	0.098039	0.200000	0.166667	0.101648	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.090909	0.098039	0.233333	0.333333	0.104396	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.090909	0.098039	0.333333	0.833333	0.112637	0.0	0.0	0.0	0.0	0.0	0.0

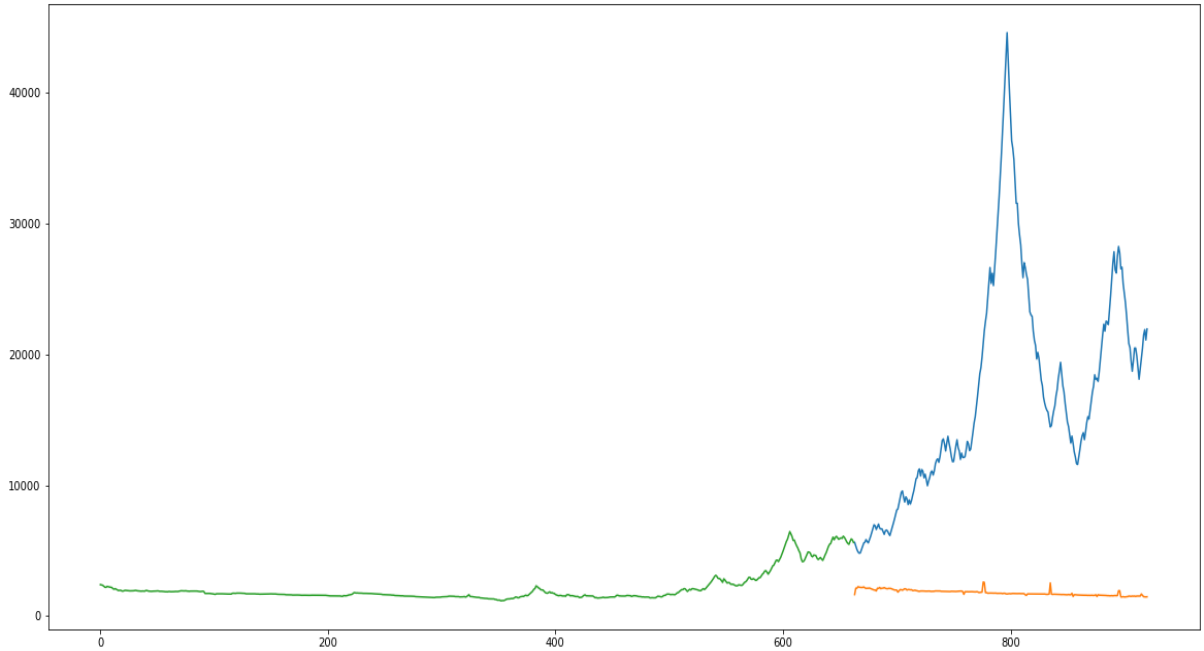
	Year	Month	Week	Day	Dayofweek	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end	Is_year_start
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

تعداد بهینه همسایه ها یعنی عدد K در این مدل با بهره گیری از روش Grid Search، عدد ۵ می باشد.
بنابراین با مدل Knn موجود در کتابخانه Skleatn و با K = 5، مدل آموزش و ارزیابی می شود. مجموعه

های train و validation استفاده شده نیز همانند قسمت قبل بوده و تنها تفاوت در نرمال سازی داده ها می باشد. خطای RMSE به دست آمده عبارت است از:

```
17360.72958930206
```

این مقدار با مدل قبلی تفاوت چندانی نداشته و عملکرد مدل همانند قبل می باشد. تصویر مقدار Close پیش بینی شده و مقدار واقعی در زیر آمده است:



بعد از محاسبه برچسب های پیش بینی شده، مقدار confusion matrix و accuracy score مدل عبارت است از:

```
[[115   0]
 [143   0]]
```

```
0.44573643410852715
```

همچنین، داده query به مدل داده شده و برچسب آن همانند قبل ۰ پیش‌بینی می‌شود.

می‌توان نتیجه‌گیری کرد که مدل های رگرسیونی همانند رگرسیون خطی و Knn، عملکرد خوبی در پیش بینی روند فردای سهم ندارند، زیرا توانایی تشخیص الگو و نوسانات در سری های زمانی را نداشته و تنها روی تاریخ ها تمرکز می کنند. بنابراین، در دو قسمت بعدی به سراغ مدل های پیش بینی سری های زمانی که به صورت اختصاصی برای این کاربرد طراحی شده اند، خواهیم رفت.

۵. Arima

Arima، یک مدل آماری پرکاربرد در پیش بینی سری های زمانی است. این مدل رویکرد متفاوتی نسبت به مدل های رگرسیونی گفته شده در قسمت های قبل دارد و برای اهدافی اختصاصی مانند پیش بینی روند سهام، عملکرد بهتری از خود نشان می‌دهد. برای پیاده سازی آن از کتابخانه Pyramid استفاده شده و جهت سهولت کار، مدل Auto-Arima به کار برده می شود، که پارامترهای مدل را به صورت اتوماتیک به نحوی انتخاب می کند که خطای مدل به کم ترین مقدار خود برسد.

برای آموزش مدل، داده های train و validation قبلی را کنار گذاشته و این بار صرفاً از ستون Close برای ساختن مدل استفاده می کنیم. بنابراین، فرآیند تقسیم داده ها به دو مجموعه train و validation همانند قبل بوده و تنها تفاوت موجود در استفاده از ستون Close و مرتب شده بر اساس تاریخ می باشد:

```
validation
```

```
<DATE>
2020-01-01     5634.0
2020-01-04     5353.0
2020-01-05     5093.0
2020-01-07     4905.0
2020-01-08     4807.0
               ...
2021-02-01    20470.0
2021-02-02    21490.0
2021-02-03    21900.0
2021-02-06    21110.0
2021-02-07    21950.0
Name: Close, Length: 258, dtype: float64
```

برای ساختن مدل، داده‌های train را به آن داده و مدل با استفاده از ۶۶۳ نمونه موجود که بر اساس تاریخ مرتب شده اند، ۲۵۹ نمونه بعدی را پیشگویی می کند:

```
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=7271.201, BIC=7293.586, Fit time=4.503 seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 12); AIC=7916.985, BIC=7925.939, Fit time=0.087 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 12); AIC=7497.609, BIC=7515.517, Fit time=2.131 seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=7293.672, BIC=7311.580, Fit time=3.237 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 1, 12); AIC=7271.686, BIC=7298.548, Fit time=6.336 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 0, 12); AIC=7678.585, BIC=7696.493, Fit time=0.591 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 2, 12); AIC=7271.514, BIC=7298.376, Fit time=13.046 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 2, 12); AIC=7272.050, BIC=7303.389, Fit time=23.602 seconds
Fit ARIMA: order=(2, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=7268.005, BIC=7294.867, Fit time=7.824 seconds
Fit ARIMA: order=(2, 1, 0) seasonal_order=(0, 1, 1, 12); AIC=7271.692, BIC=7294.077, Fit time=3.680 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=7269.950, BIC=7301.288, Fit time=9.020 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(0, 1, 1, 12); AIC=7270.566, BIC=7288.474, Fit time=2.620 seconds
Fit ARIMA: order=(3, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=7267.743, BIC=7303.559, Fit time=16.433 seconds
Fit ARIMA: order=(3, 1, 2) seasonal_order=(1, 1, 1, 12); AIC=7325.588, BIC=7365.881, Fit time=17.346 seconds
Fit ARIMA: order=(3, 1, 2) seasonal_order=(0, 1, 0, 12); AIC=7611.675, BIC=7643.014, Fit time=3.981 seconds
Fit ARIMA: order=(3, 1, 2) seasonal_order=(0, 1, 2, 12); AIC=7263.218, BIC=7303.511, Fit time=33.986 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 2, 12); AIC=7270.956, BIC=7306.771, Fit time=25.501 seconds
Fit ARIMA: order=(3, 1, 1) seasonal_order=(0, 1, 2, 12); AIC=7273.622, BIC=7309.438, Fit time=35.519 seconds
Fit ARIMA: order=(3, 1, 3) seasonal_order=(0, 1, 2, 12); AIC=7263.225, BIC=7307.995, Fit time=38.983 seconds
Fit ARIMA: order=(2, 1, 1) seasonal_order=(0, 1, 2, 12); AIC=7269.018, BIC=7300.357, Fit time=23.039 seconds
Fit ARIMA: order=(3, 1, 2) seasonal_order=(1, 1, 2, 12); AIC=7264.942, BIC=7309.712, Fit time=37.431 seconds
Total fit time: 308.912 seconds
```

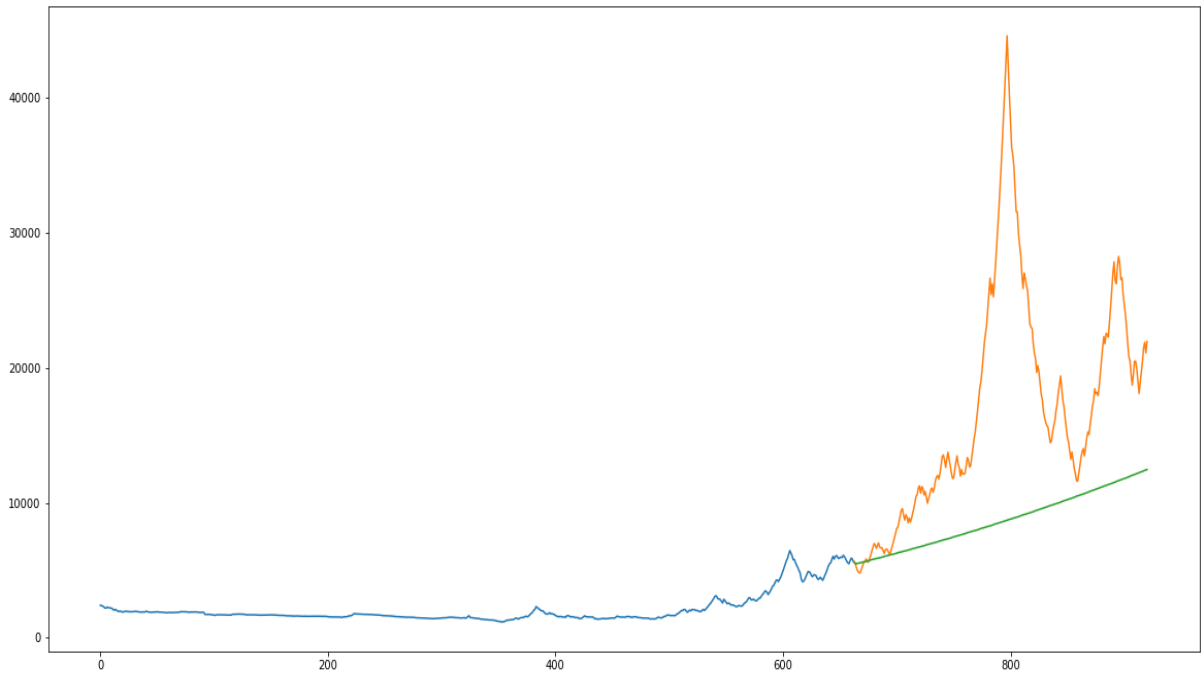
بنابراین بر خلاف روش های قبلی که یک سری از ویژگی ها را به عنوان متغیرهای مستقل جدا کرده و با استفاده از آن ها متغیر هدف یا وابسته را پیشگویی می کردند، آریما صرفا از ویژگی Close در داده‌های train استفاده کرده و دنباله ی این سری زمانی را برای ۲۵۹ داده بعدی (همان validation)، به صورت اتوماتیک پیشگویی می کند.

عدد ۲۵۹ به این دلیل ذکر شده که علاوه بر داده های validation، آخرین مقدار یعنی ۲۵۹-امین نمونه که همان داده query یا داده های روز ۲۰ بهمن می باشد، نیز پیش بینی شود.

خطای RMSE مدل، با مقایسه مقادیر Close پیش‌بینی شده با مقادیر واقعی به شرح زیر می باشد:

```
11106.315881395603
```

همانطور که می بینیم، این مقدار نسبت به دو مدل قبل کاهش یافته و این یعنی مدل Arima عملکرد بهتری نسبت به عامل های هوشمند قبلی از خود نشان داده است. مقادیر Close پیش بینی شده و مقادیر واقعی در تصویر زیر رسم شده است:



با محاسبه برچسب های پیش بینی شده، confusion matrix و accuracy score مدل عبارت است از:

```
[[112   3]
 [137   6]]
```

```
0.4573643410852713
```

مقدار برچسب برای داده query همانند قبل ۰ پیش‌بینی شده که غلط می باشد، اما مدل عملکرد بهتری نسبت به مدل های رگرسیونی از خود نشان داده و مقدار پیش بینی شده آن برای ستون Close به مقدار واقعی نزدیک تر می باشد که در نمودار فوق نیز مشهود است.

علت این امر، توانایی مدل Arima در تشخیص الگو های موجود در سری های زمانی، بر خلاف مدل های رگرسیونی می باشد و تمرکز آن صرفاً روی ویژگی‌های مرتبط با تاریخ نیست. با این وجود، مدل Arima ایده آل نیست و صرفاً یک روند افزایشی را در سری های زمانی تشخیص داده، اما میزان نوسانات صعودی و نزولی را در نظر نگرفته است. به همین دلیل، به سراغ مدل بعدی می رویم.

۶. Prophet

مدل Prophet یکی دیگر از مدل های پیش بینی سری های زمانی است که توسط تیم Facebook معرفی شد. کار کردن با آن بسیار راحت بوده و نیازمند به پیش پردازش های آماری نمی باشد. برای استفاده از این مدل لازم است که داده ها به یک دیتا فریم با دو ستون ds یعنی تاریخ و y که همان ستون هدف است، تبدیل شوند. بدین منظور، ویژگی Close به عنوان ستون y و ویژگی Date به عنوان ستون ds، از داده‌های اولیه جدا شده و با همان روند قبلی به مجموعه‌های train و validation تقسیم می شوند:

Date	ds	y
2017-02-05	2017-02-05 00:00:00	2401
2017-02-06	2017-02-06 00:00:00	2380
2017-02-07	2017-02-07 00:00:00	2355
2017-02-08	2017-02-08 00:00:00	2282
2017-02-11	2017-02-11 00:00:00	2209

Date	ds	y
2021-02-08	2021-02-08 00:00:00	22810

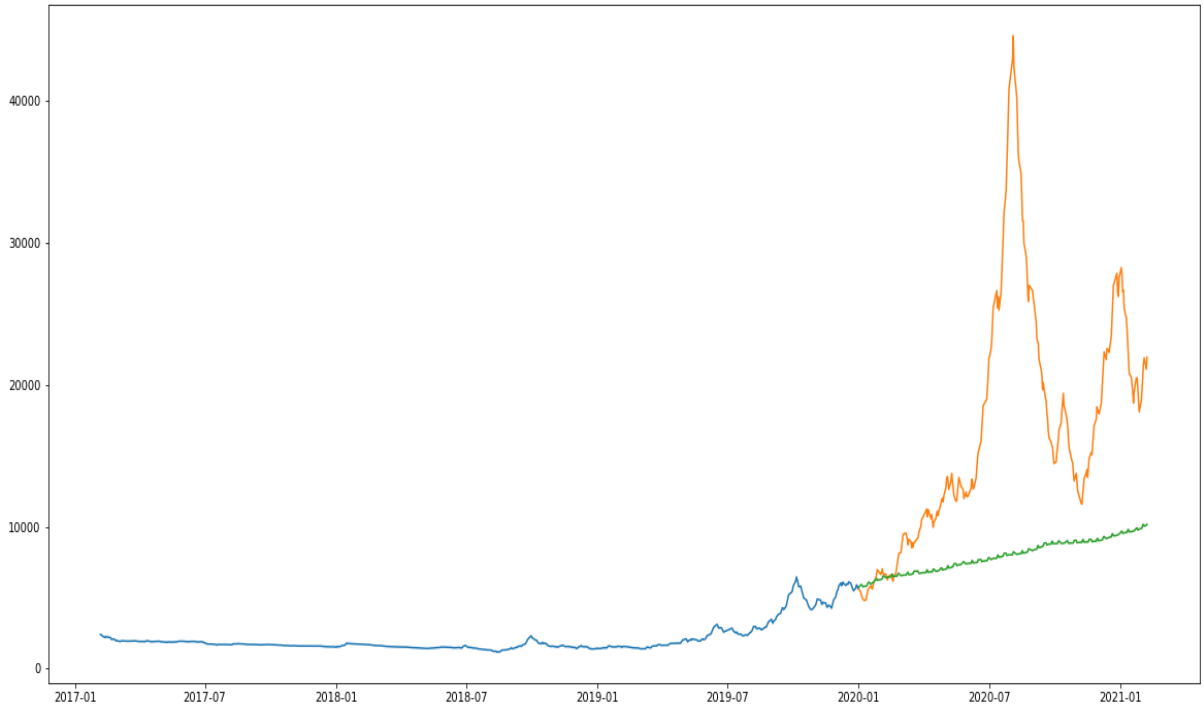
روند آموزش همانند مدل قبلی بوده و مدل با دریافت داده های train، ۲۵۹ مقدار بعدی را پیش بینی می کند.

مدل با استفاده از کتابخانه Fbprophet ساخته می شود.

خطای RMSE به دست آمده از ویژگی Close پیش بینی شده در زیر آمده است:

```
11858.1901078596
```

مقدار این خطا تقریباً مشابه با مدل قبلی است. ویژگی Close پیش بینی شده در مقایسه با مقادیر اصلی در نمودار آمده است:



همانطور که در تصویر فوق مشخص است، این مدل نیز ایده آل نمی باشد، اما میزان نوسانات را نسبت به مدل قبلی بهتر پیش بینی کرده و الگوی موجود در سری زمانی را نیز تشخیص می دهد.

Confusion matrix و دقت accuracy score محاسبه شده با توجه به برچسب های پیش بینی شده در زیر آمده است:

```
[[106   9]
 [130  13]]

0.46124031007751937
```

این مدل نیز مشابه با مدل های قبلی مقدار برچسب برای داده query را ۰ پیش‌بینی می‌کند.

در قسمت بعد، به عنوان عامل آخر، یک شبکه عصبی عمیق LSTM را آموزش داده و آن را با مدل های رگرسیونی و مدل های آماری مطرح شده مقایسه می کنیم.

۷. شبکه عصبی عمیق LSTM

آخرین عامل هوشمند استفاده شده در این پروژه، شبکه عصبی عمیق LSTM می باشد. این نوع از شبکه های عصبی عمیق، برای پیش‌بینی روند ها و الگو های سری های زمانی و به طور کلی، داده های از نوع sequential، بسیار کاربرد دارند. دلیل این امر، وجود یک نوع حافظه داخلی در شبکه است، که به واسطه

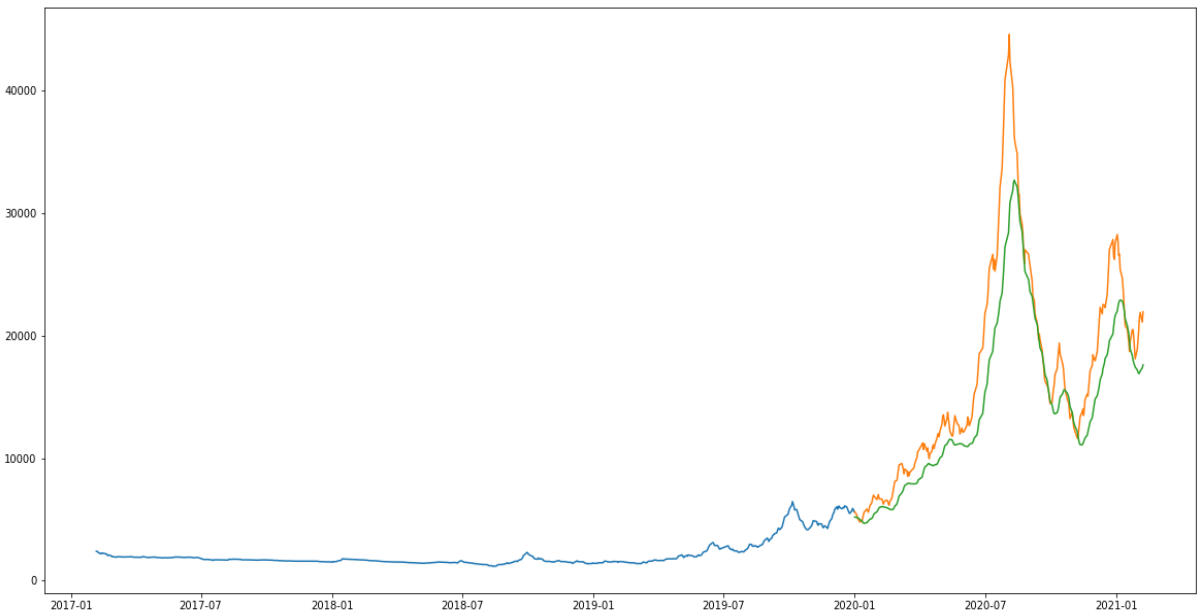
آن شبکه می تواند الگوها را با دقت بالاتری نسبت به مدل های قبلی تشخیص دهد. به عبارت دیگر، مدل LSTM با ذخیره اطلاعات داده های قبلی در حافظه خود، پیش بینی دقیق تری نسبت به داده های جدید انجام می دهد.

برای ساختن مدل از کتابخانه Keras استفاده می شود. داده ها حتماً باید نرمال سازی شوند بنابراین همانند روش ذکر شده در بالا، داده ها را به بازه ۰ و ۱ اسکیل می کنیم. در این مدل نیز تنها از ویژگی Close استفاده می کنیم، و آن را از داده‌های اولیه جدا کرده، و سپس همانند روش های قبلی مجموعه های train و validation را می سازیم. سپس، یک پنجره ۶۰ تایی در نظر گرفته، از اولین نمونه در مجموعه train شروع کرده و هر بار ۶۰ تا از نمونه ها را به عنوان ویژگی های مستقل که به صورت یک سکانس پشت سر هم بر اساس تاریخ مرتب شده اند، جدا کرده و ۶۱-امین سمپل را به عنوان ویژگی هدف در نظر می گیریم. به عبارت دیگر هر نمونه داده شده به شبکه عصبی، یک سکانس ۶۰ تایی از داده ها مرتب شده بر اساس تاریخ می باشد، و شبکه موظف است ۶۱-امین داده را که بلافاصله بعد از این پنجره ۶۰ تایی می‌آید، پیش بینی کند. به این ترتیب با استفاده از این سکانس ها که از مجموعه train به دست آمده اند، شبکه را آموزش داده و سپس ۲۵۹ مقدار بعدی را پیش بینی می کنیم. جزئیات دقیق تر پیاده‌سازی در نوت بوک کد آمده است.

با استفاده از این روش، خطای RMSE ناشی از پیش بینی ویژگی Close بدین شکل است:

```
3756.635371665776
```

این مقدار نسبت به روش های قبلی کاهش محسوسی داشته و بدون شک قدرتمندترین عامل هوشمند در پیش بینی روند سهم می باشد. این موضوع در نمودار زیر مشهود است:



همانطور که می بینیم، مقدار پیش بینی شده نسبت به روش‌های قبلی، بسیار نزدیک تر به مقدار واقعی می باشد و این نشان از عملکرد قابل قبول مدل دارد.

Confusion matrix و accuracy score به دست آمده از مقایسه برچسب های پیش بینی شده و برچسب های اصلی به شرح زیر است:

```
[[105  10]
 [140   3]]

0.4186046511627907
```

همانند سایر مدل ها، LSTM برچسب query را درست پیش‌بینی نمی کند، اما مقدار متغیر Close را بسیار بهتر از مدل های قبلی پیش‌بینی می‌کند، که از کمینه بودن خطای RMSE هم قابل نتیجه گیری بود. بنابراین، بهترین عامل هوشمند موجود در پیش بینی روند فردای سهم می باشد.

۸. نتیجه گیری

برای پیش بینی روند صعودی یا نزولی بودن سهم، چندین عامل هوشمند با یکدیگر مقایسه شدند. از روش های رگرسیونی، روش های آماری و همچنین شبکه های عصبی عمیق در این پروژه بهره بردیم. همانطور که قبلاً هم اشاره شد، از بین این روش ها، شبکه عصبی عمیق LSTM بی شک بهترین عملکرد را نسبت به بقیه عامل ها دارد. با وجود اینکه برچسب داده query در هیچ کدام از مدل ها درست پیش بینی نشد، و دقت دسته بندی هم نسبتاً پایین بود، اما با مقایسه خطای RMSE مدل ها به این نتیجه می رسیم که عامل های پیشرفته تر مانند شبکه های عصبی عمیق، نتایج دقیق تری نسبت به سایر مدل ها به دست می دهند. از آن جا که عملکرد سهام در دنیای واقعی، به فاکتورهای زیادی به غیر از ویژگی هایی که در این پروژه استفاده کردیم بستگی دارد، نمی توان تنها با استناد به عامل های هوشمند و روش های یادگیری ماشینی به طور دقیق روند فردای یک سهم را با دقت بالا پیش بینی کرد، و همین امر دلیل پایین بودن دقت در پروژه بود. با این وجود، در این پروژه، ساختار، کاربردها، و عملکرد چندین نوع مختلف از عامل های هوشمند با استفاده از داده‌های پردازش شده و روزانه سهام با یکدیگر مقایسه شد، و از دیدگاه هوش مصنوعی عملکرد این مدل ها مورد مطالعه قرار گرفت، که همان هدف اصلی پروژه بود.