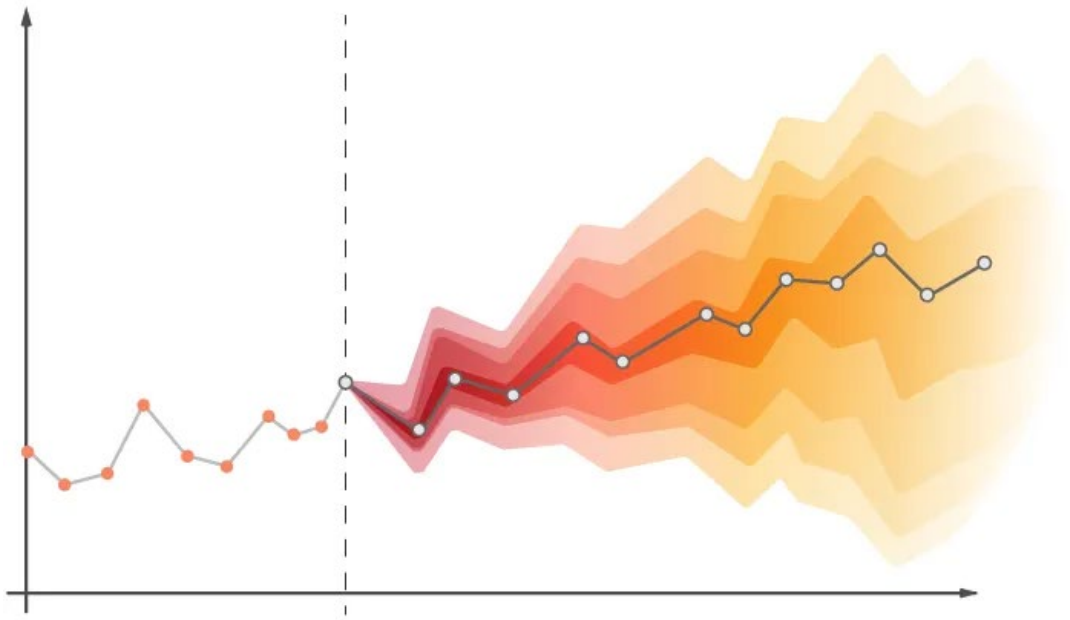
In the name of god



گزارش های دوره ای

پایتون برای امور مالی: تحلیل سری‌های زمانی

هلدینگ صنایع نانوتک آینده

خرداد ۱۴۰۴

فهرست مطالب

کتابخانه Pandas برای داده‌های سری زمانی ۳

شاخص تاریخ و زمان(Date Time Index) ۳

نمونه‌برداری مجدد زمانی..... ۴

جابجایی زمانی(Time Shifts) ۷

محاسبات محرک و رشد کننده (Rolling & Expanding) در PANDAS..... ۷

باندهای بولینگر یا Bollinger Bands..... ۱۰

تحلیل سری‌های زمانی ۱۱

معرفی کتابخانه Statsmodels ۱۲

مدل‌های ETS با استفاده از StatsModels..... ۱۳

مدل‌های EWMA..... ۱۴

ARIMA..... ۱۶

جمع بندی ۱۷

پایتون برای امور مالی: تحلیل سری‌های زمانی

پایتون یکی از سریع‌ترین زبان‌های برنامه‌نویسی در حال رشد برای کاربردهای مالی و یادگیری ماشین است. در این مقاله، به بررسی نحوه ساخت مدل‌هایی برای تحلیل سری‌های زمانی با استفاده از پایتون می‌پردازیم. همانطور که خواهیم دید، مسائل سری زمانی ویژگی‌های منحصر به فردی دارند که آنها را از مسائل پیش‌بینی سنتی متمایز می‌کند.

کتابخانه Pandas برای داده‌های سری زمانی

برای شروع، بیایید چند نکته کلیدی درباره استفاده از Pandas برای داده‌های سری زمانی را مرور کنیم.اکثر مجموعه داده‌های مالی به صورت سری زمانی هستند که شامل یک شاخص تاریخ و زمان (Date Time) و مقدار متناظر با آن می‌باشند.

Pandasویژگی‌های خاصی برای کار با داده‌های سری زمانی دارد، از جمله:

- شاخص تاریخ و زمان(Date Time index)
- نمونه‌برداری مجدد زمانی(Time resampling)
- جابجایی زمانی(Time shifts)
- محاسبات حرکت کننده و رشد کننده(Rolling and expanding)

شاخص تاریخ و زمان(Date Time Index)

اغلب در مجموعه داده‌های مالی، زمان و تاریخ به صورت یک ستون جداگانه نیستند، بلکه به عنوان شاخص (index) در نظر گرفته می‌شوند. کتابخانه‌های داخلی پایتون برای کار با تاریخ و زمان وجود دارند، بنابراین بدون نیاز به نصب کتابخانه‌های اضافی می‌توانیم از آنها استفاده کنیم:

```
from datetime import datetime
```

این امکان را به ما می‌دهد تا زمان‌مهرها (timestamps) یا اشیاء تاریخ خاصی ایجاد کنیم. حال بیاید در محیط پایتون چند متغیر ایجاد نماییم:

```
my_year = 2021
my_month = 5
my_day= 1
```

برای استفاده از تابع داخلی datetime می‌توانیم از syntax زیر استفاده کنیم:

```
my_date = datetime()
```

همان‌طور که می‌بینیم، این تابع سال، ماه، روز و زمان را دریافت می‌کند. بیایید این آرگومان‌ها را وارد کنیم:

```
my_date = datetime(my_year, my_month, my_day)
```

بیایید نگاهی به این شیء datetime بیندازیم:

```
[4] my_date = datetime(my_year, my_month, my_day)

[5] my_date

    datetime.datetime(2021, 5, 1, 0, 0)
```

در اینجا یک فهرست شامل دو شیء datetime را به یک شاخص تبدیل می‌کنیم:

```
my_list = [datetime(2021, 1, 1), datetime(2021, 1, 2)]
```

می‌توانیم یک آرایه NumPy یا فهرست را با استفاده از دستور زیر به یک شاخص تبدیل کنیم:

```
dt_idx = pd.DatetimeIndex(my_list)
```

```
[6] my_list = [datetime(2021,1,1), datetime(2021,1,2)]

[7] # convert list of datetime objects to datetime index
    dt_idx = pd.DatetimeIndex(my_list)

[8] dt_idx

    DatetimeIndex(['2021-01-01', '2021-01-02'], dtype='datetime64[ns]', freq=None)
```

نمونه‌برداری مجدد زمانی

هنگام کار با مجموعه داده‌های مالی، معمولاً داده‌هایی با شاخص تاریخ-زمان در مقیاس کوچک‌تر (روز، ساعت، دقیقه و غیره) دریافت می‌کنیم. با این حال، برای تحلیل، اغلب ایده خوبی است که داده‌ها را بر اساس فرکانس‌های خاصی (ماهانه، سه‌ماهه و غیره) تجمیع کنیم.شاید فکر کنید قابلیت GroupBy می‌تواند این مشکل را حل کند، اما این ابزار برای درک مفاهیمی مانند سه‌ماهه مالی، شروع سال یا شروع هفته طراحی نشده است. خوشبختانه Pandas ابزارهای نمونه‌برداری فرکانسی داخلی برای حل این مسئله دارد.برای درک بهتر، بیایید داده‌های بازار سهام تسلا از ۱ می ۲۰۲۰ تا ۱ می ۲۰۲۱ را بررسی کنیم که می‌توانید از Yahoo Finance دانلود کنید. وارد کردن کتابخانه‌ها:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

آپلود داده در Google Colab:

```
from google.colab import files
```

```
uploaded = files.upload()
```

خواندن فایل CSV و بررسی داده‌ها:

```
df = pd.read_csv('TSLA.csv')
```

ستون Date باید به عنوان شاخص در نظر گرفته شود، بنابراین آن را با استفاده از pd.to_datetime() به شاخص تاریخ-زمان تبدیل می‌کنیم:

```
df['Date'] = pd.to_datetime(df['Date'])
```

با فراخوانی df.info() می‌بینیم که این ستون اکنون یک شیء datetime است:

```
[9] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252 entries, 0 to 251
Data columns (total 7 columns):
Date         252 non-null datetime64[ns]
Open         252 non-null float64
High         252 non-null float64
Low          252 non-null float64
Close        252 non-null float64
Adj Close    252 non-null float64
Volume       252 non-null int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 13.9 KB
```

حال ستون Date را به عنوان شاخص تنظیم می‌کنیم:

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

برای ساده‌تر کردن این کار، می‌توانستیم هنگام خواندن فایل، از گزینه‌های index_col='Date' و parse_dates=True استفاده کنیم. سپس می‌توانیم شاخص را با دستور df.index بررسی کنیم:

```
[16] df = pd.read_csv('TSLA.csv', index_col='Date', parse_dates=True)

[17] df.index

DatetimeIndex(['2020-05-01', '2020-05-04', '2020-05-05', '2020-05-06',
               '2020-05-07', '2020-05-08', '2020-05-11', '2020-05-12',
               '2020-05-13', '2020-05-14',
               ...
               '2021-04-19', '2021-04-20', '2021-04-21', '2021-04-22',
               '2021-04-23', '2021-04-26', '2021-04-27', '2021-04-28',
               '2021-04-29', '2021-04-30'],
              dtype='datetime64[ns]', name='Date', length=252, freq=None)
```

برای انجام هر نوع نمونه‌برداری مجدد زمانی(time resampling) ، ابتدا به یک شاخص datetime نیاز داریم، سپس می‌توانیم با استفاده از متد df.resample() داده‌ها را نمونه‌برداری کنیم و یک قانون (rule) مشخص کنیم. قانون یا rule

مشخص می‌کند که چگونه می‌خواهیم از داده‌ها نمونه برداری کنیم. برای هر نوع جابجایی زمانی (time series offset) کلمات کلیدی خاصی وجود دارد که می‌توانید جزئیات بیشتر آن را در مستندات مربوطه مطالعه کنید. در واقع، این قانون مانند یک روش GroupBy است که به طور خاص برای داده‌های سری زمانی طراحی شده است.

بیایید یک مثال از قانون Aرا ببینیم که به معنای «فرکانس پایان سال» است و میانگین مقادیر بر اساس نمونه‌برداری مجدد را محاسبه می‌کند:

```
# محاسبه میانگین بر اساس نمونه‌برداری مجدد در پایان سال
df.resample(rule='A').mean()
```

```
[18] # mean value based off the end of the year resampling
     df.resample(rule='A').mean()
```

Date	Open	High	Low	Close	Adj Close	Volume
2020-12-31	368.541566	378.454683	358.214741	369.831953	369.831953	6.259896e+07
2021-12-31	743.344025	759.444267	723.270975	742.028291	742.028291	3.475055e+07

در این مثال، میانگین مقدار ستون Open برای همه داده‌های قبل از ۳۱-۱۲-۲۰۱۸ برابر با ۳۱۶,۱۲ دلار بوده است. میانگین مقدار برای داده‌های بین ۳۱-۱۲-۲۰۱۸ تا ۳۱-۱۲-۲۰۱۹ برابر با ۲۹۱,۴۴ دلار بوده است. سپس می‌توانیم با استفاده از قانون Q، میانگین مقادیر را پس از نمونه‌برداری مجدد به صورت سه‌ماهه به دست آوریم:

```
# نمونه‌برداری مجدد سه‌ماهه
df.resample(rule='Q').mean()
```

```
[19] # quarterly resampling
     df.resample(rule='Q').mean()
```

Date	Open	High	Low	Close	Adj Close	Volume
2020-06-30	176.483143	180.446523	172.655477	177.078286	177.078286	6.291385e+07
2020-09-30	352.975532	365.226126	340.174626	354.207405	354.207405	8.104423e+07
2020-12-31	510.145940	521.626095	498.028124	511.951094	511.951094	4.394705e+07
2021-03-31	755.070000	771.730656	732.933279	753.185899	753.185899	3.559025e+07
2021-06-30	709.282860	723.755232	695.204285	709.618094	709.618094	3.231141e+07

جابجایی زمانی(Time Shifts)

اغلب مدل‌های پیش‌بینی سری‌های زمانی نیاز دارند که داده‌ها را به جلو یا عقب به اندازه‌ای مشخص جابجا کنیم. کتابخانه Pandasاین کار را به سادگی با متد shift().امکان‌پذیر می‌کند. برای نمایش این موضوع، دوباره به فایل CSV تسلا نگاه می‌کنیم که داده‌های روزانه دارد. اگر بخواهیم دوره زمانی را به اندازه یک گام به جلو جابجا کنیم، می‌توانیم از دستور زیر استفاده کنیم:

```
df.shift(periods=1).head()
```

```
[25] # shift by 1 time period
     df.shift(periods=1).head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2020-05-01	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-04	151.000000	154.554001	136.608002	140.264008	140.264008	162659000.0
2020-05-05	140.199997	152.399994	139.600006	152.238007	152.238007	96185500.0
2020-05-06	157.957993	159.783997	152.436005	153.641998	153.641998	84958500.0
2020-05-07	155.300003	157.960007	152.222000	156.516006	156.516006	55616000.0

پس از این کار مشاهده می‌کنیم که دیگر برای اولین دوره زمانی داده‌ای نداریم. همچنین می‌توانیم با استفاده از مقدار 1-برای آرگومان periods، دوره زمانی را به عقب جابجا کنیم.

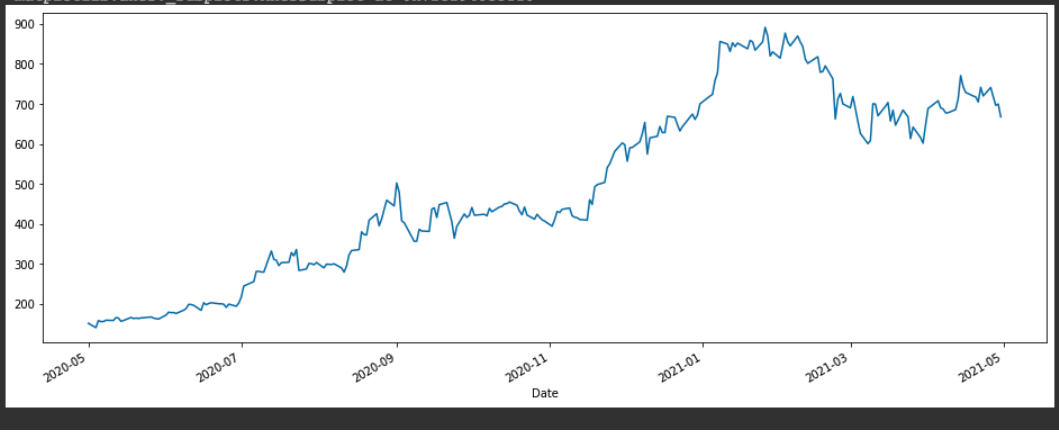
محاسبات محرک و رشد کننده (Rolling & Expanding) در PANDAS

ما می‌توانیم از متد داخلی rollingدر pandas استفاده کنیم، مثلاً وقتی بخواهیم میانگین متحرک (rolling mean) بر اساس یک بازه زمانی مشخص ایجاد کنیم .در کار با داده‌های مالی، اغلب داده‌های روزانه دارای نوسانات زیادی هستند .برای مقابله با این موضوع، می‌توانیم از میانگین متحرک که به آن Moving Average نیز گفته می‌شود استفاده کنیم تا سیگنالی درباره روند کلی داده‌ها به دست آوریم .بیایید داده‌های روزانه خود را با دستور زیر رسم کنیم:

```
df['Open'].plot(figsize=(16,6))
```

```
[26] # plot the open
     df['Open'].plot(figsize=(16,6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcf948cc110>
```



حال بیایید میانگین هفتگی را محاسبه کنیم — می‌توانیم میانگین متحرک را روی یک ستون یا سری خاص، یا روی کل حال بیایید میانگین هفتگی را محاسبه کنیم. می‌توانیم میانگین متحرک را روی یک ستون یا سری خاص، یا روی کل DataFrame با استفاده از متد rolling().به دست آوریم. برای این کار، عدد ۷ را به عنوان پنجره (window) وارد می‌کنیم و سپس تابع تجمیعی mean().را اضافه می‌کنیم:

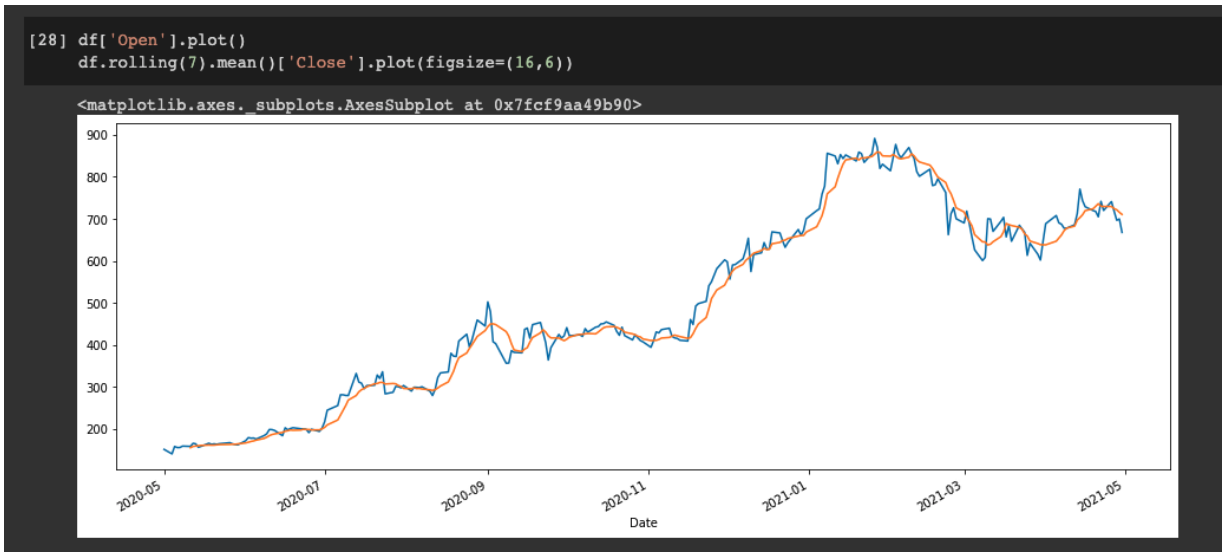
```
df.rolling(7).mean().head(14)
```

```
[27] df.rolling(7).mean().head(14)
```

Date	Open	High	Low	Close	Adj Close	Volume
2020-05-01	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-04	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-05	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-06	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-07	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-08	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-11	153.822285	159.082572	149.962572	154.972859	154.972859	8.858400e+07
2020-05-12	155.879427	161.097430	153.532859	158.061144	158.061144	7.670907e+07
2020-05-13	159.303142	162.926002	155.398573	158.911715	158.911715	7.658650e+07
2020-05-14	159.023429	163.052859	155.450572	159.915144	159.915144	7.422257e+07
2020-05-15	159.419144	163.488571	156.177429	160.389143	160.389143	7.379057e+07
2020-05-18	160.864001	164.583429	157.078286	161.348857	161.348857	7.388921e+07
2020-05-19	161.475429	164.528285	157.623145	161.022858	161.022858	6.928207e+07
2020-05-20	162.332286	164.585427	158.388859	161.144858	161.144858	6.270329e+07

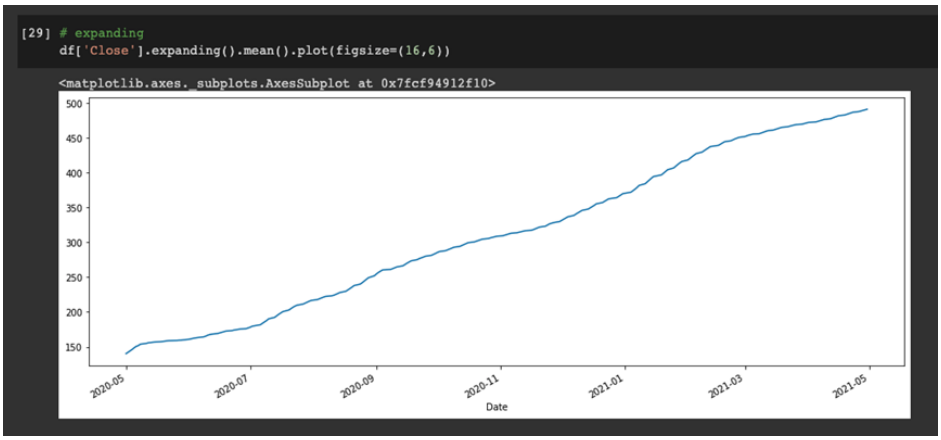
می‌بینیم که شش مقدار اول خالی (null) هستند و مقدار هفتم میانگین شش ردیف اول است. حال بیایید ستون Open را در مقابل میانگین متحرک ۷ روزه ستون Close رسم کنیم:

```
df['Open'].plot()
df.rolling(7).mean()['Close'].plot(figsize=(16,6))
```



وقتی به این نمودار نگاه می‌کنیم، خط آبی نشان‌دهنده قیمت باز شدن (Open) و خط نارنجی میانگین متحرک ۷ روزه قیمت بسته شدن (Close) است.حالا، وقتی بخواهیم همه داده‌ها از ابتدای سری زمانی تا نقطه جاری را در نظر بگیریم، چه کاری باید انجام دهیم؟ برای مثال، به جای اینکه فقط پنجره‌ای ۷ روزه را در نظر بگیریم، همه داده‌ها از ابتدای سری زمانی تا نقطه فعلی را لحاظ کنیم.برای این کار از متد expanding().استفاده می‌کنیم:

```
df['Close'].expanding().mean().plot(figsize=(16,6))
```



خب، این نمودار چه چیزی را نشان می‌دهد؟ در هر گام زمانی روی محورx ، مقدار نمایش داده شده روی محورy ، میانگین همه مقادیری است که قبل از آن آمده‌اند.

باندهای بولینگر یا Bollinger Bands

باندهای بولینگر ابزاری محبوب در تحلیل تکنیکال هستند که توسط جان بولینگر در دهه ۱۹۸۰ توسعه یافته‌اند. این باندها شامل یک میانگین متحرک ساده (معمولاً ۲۰ روزه) و دو باند بالا و پایین هستند که هر کدام دو انحراف معیار بالاتر و پایین‌تر از میانگین متحرک قرار دارند. فاصله بین این باندها نشان‌دهنده میزان نوسان بازار است؛ باندهای گسترده‌تر نشان‌دهنده نوسان بیشتر و باندهای باریک‌تر نشان‌دهنده نوسان کمتر هستند. باندهای بولینگر به معامله‌گران کمک می‌کنند تا نقاط اشباع خرید و فروش را شناسایی کرده و روندهای قیمتی را تحلیل کنند. مثلاً وقتی قیمت به باند بالایی می‌رسد یا از آن عبور می‌کند، ممکن است نشان‌دهنده افزایش نوسان و احتمال برگشت قیمت باشد. بالعکس، رسیدن قیمت به باند پایینی می‌تواند نشانه‌ای از اشباع فروش باشد. این ابزار به همراه سایر اندیکاتورها مانند RSI و MACD می‌تواند در شناسایی شرایط بازار، تغییر روندها و سیگنال‌های خرید و فروش بسیار مفید باشد.به طور خلاصه، باندهای بولینگر ابزاری پویا برای اندازه‌گیری نوسان و تعیین کانال‌های قیمتی یا روندهای محتمل در بازار هستند که به معامله‌گران امکان می‌دهند تصمیمات بهتری در معاملات خود بگیرند.در ادامه به مباحث بنیادی و تحلیل تکنیکال بیشتری خواهیم پرداخت، اما یکی از موضوعات مرتبط با متد ()rolling، باندهای بولینگر است که به طور خلاصه به آنها می‌پردازیم. باندهای بولینگر، باندهای نوسانی هستند که بالاتر و پایین‌تر از میانگین متحرک قرار می‌گیرند. نوسان این باندها بر اساس انحراف معیار است که با افزایش یا کاهش نوسان تغییر می‌کند.

وقتی نوسان بازار افزایش می‌یابد، باندها گسترش می‌یابند و وقتی نوسان کاهش می‌یابد، باندها باریک‌تر می‌شوند.

بیایید ببینیم چگونه می‌توانیم باندهای بولینگر را با استفاده از Pandas کدنویسی کنیم. مراحل مورد نیاز عبارتند از:

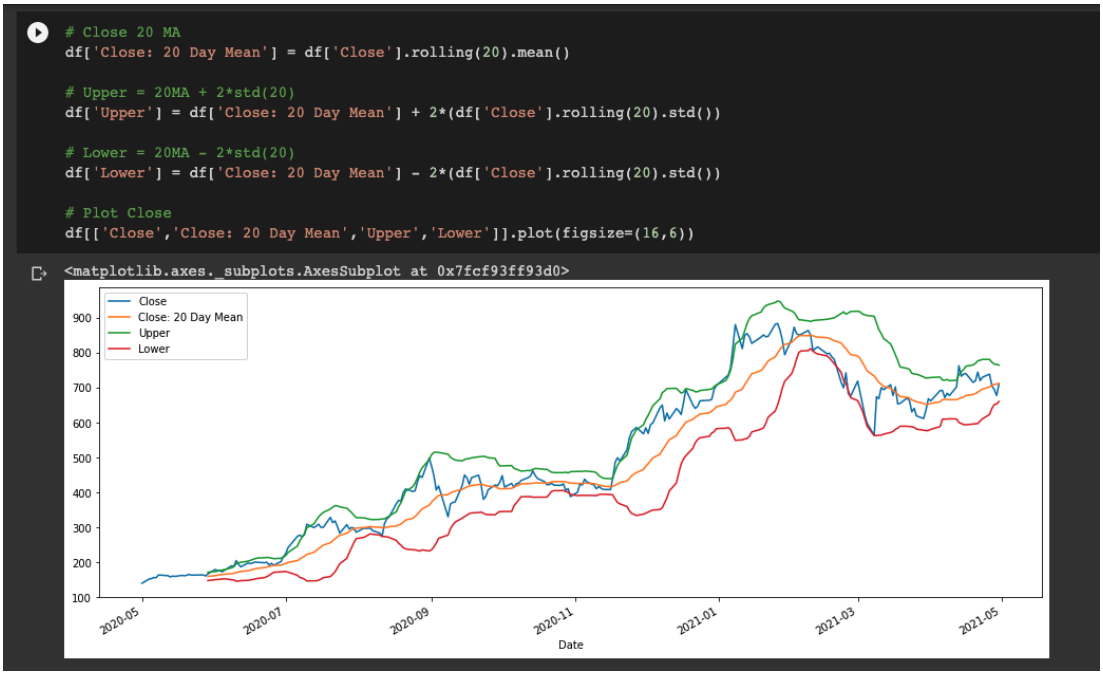
ابتدا باید سه ستون ایجاد کنیم و سپس آنها را رسم کنیم:

- ستون اول میانگین متحرک ۲۰ روزه قیمت بسته شدن (Closing 20-day Moving Average) است.
- سپس باند بالایی را برابر با میانگین متحرک ۲۰ روزه به اضافه دو برابر انحراف معیار ۲۰ روزه تعریف می‌کنیم.
- باند پایینی برابر با میانگین متحرک ۲۰ روزه منهای دو برابر انحراف معیار ۲۰ روزه است.

کد مربوطه به شکل زیر است:

```
# میانگین متحرک ۲۰ روزه قیمت بسته شدن
df['Close: 20 Day Mean'] = df['Close'].rolling(20).mean()

# باند بالایی = میانگین متحرک ۲۰ روزه + ۲ برابر انحراف معیار ۲۰ روزه
df['Upper'] = df['Close: 20 Day Mean'] + 2*(df['Close'].rolling(20).std())

# باند پایینی = میانگین متحرک ۲۰ روزه - ۲ برابر انحراف معیار ۲۰ روزه
```

```
df['Lower'] = df['Close: 20 Day Mean'] - 2*(df['Close'].rolling(20).std())

# رسم نمودار قیمت بسته شدن و باندهای بولینگر
df[['Close','Close: 20 Day Mean','Upper','Lower']].plot(figsize=(16,6))
```



تحلیل سری‌های زمانی

حال که با کتابخانه Pandas برای داده‌های سری زمانی آشنا شدیم، بیایید تمرکز خود را به چند تکنیک تحلیل سری‌های زمانی معطوف کنیم. داده‌های سری زمانی ویژگی‌های خاصی دارند و الگوریتم‌های پیش‌بینی متفاوتی نسبت به سایر انواع داده‌ها دارند.

موضوعاتی که در ادامه بررسی خواهیم کرد عبارتند از:

- معرفی کتابخانهStatsmodels
- مدل‌های ETS و تجزیه
- مدل‌هایEWMA
- مدل‌هایARIMA

معرفی کتابخانه Statsmodels

محبوب‌ترین کتابخانه پایتون برای کار با داده‌های سری زمانی، کتابخانه Statsmodels است:

statsmodels یک ماژول پایتون است که کلاس‌ها و توابعی برای برآورد مدل‌های آماری مختلف، انجام آزمون‌های آماری و کاوش داده‌های آماری ارائه می‌دهد. این کتابخانه به شدت از زبان برنامه‌نویسی آماری R الهام گرفته است.

Statsmodelsبه کاربران امکان می‌دهد داده‌ها را کاوش کنند، مدل‌های آماری را برآورد کنند و آزمون‌های آماری را انجام دهند. بیایید به ماژول تحلیل سری‌های زمانی tsaنگاهی بیندازیم. ابتدا کتابخانه را به صورت زیر وارد می‌کنیم و سپس یک مجموعه داده که همراه با کتابخانه است را بارگذاری می‌کنیم:

```
# وارد کردن داده‌ها با متد load_pandas و ویژگی .data
df = sm.datasets.macrodata.load_pandas().data
df.head()
```

```
[6] # import dataset with load_pandas method and .data attribute
    df = sm.datasets.macrodata.load_pandas().data
    df.head()
```

	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	cpi	m1	tbilrate	unemp	pop	infl	realint
0	1959.0	1.0	2710.349	1707.4	286.898	470.045	1886.9	28.98	139.7	2.82	5.8	177.146	0.00	0.00
1	1959.0	2.0	2778.801	1733.7	310.859	481.301	1919.7	29.15	141.7	3.08	5.1	177.830	2.34	0.74
2	1959.0	3.0	2775.488	1751.8	289.226	491.260	1916.4	29.35	140.5	3.82	5.3	178.657	2.74	1.09
3	1959.0	4.0	2785.204	1753.7	299.356	484.052	1931.3	29.37	140.0	4.33	5.6	179.386	0.27	4.06
4	1960.0	1.0	2847.699	1770.5	331.722	462.199	1955.5	29.54	139.6	3.50	5.2	180.007	2.31	1.19

می‌توانیم با استفاده از ویژگی NOTE.اطلاعاتی درباره محتوای مجموعه داده به دست آوریم. این داده‌ها مربوط به اطلاعات اقتصادی ایالات متحده هستند. حال ستون سال را به عنوان شاخص سری زمانی تنظیم می‌کنیم:

```
index = pd.Index(sm.tsa.datetools.dates_from_range('1959Q1','2009Q3'))
df.index = index
```

حالا که سال به عنوان شاخص سری زمانی تنظیم شده است، ستون realgdpرا رسم می‌کنیم:

```
df['realgdp'].plot()
```

بیایید با استفاده از Statsmodels تحلیل انجام دهیم تا روند داده‌ها را به دست آوریم. در اینجا از فیلتر هادریک-پرسکات (Hodrick-Prescott)استفاده می‌کنیم:

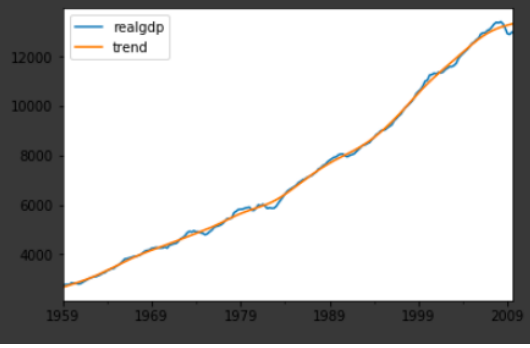
```
sm.tsa.filters.hpfilter(df['realgdp'])
```

این تابع یک تاپل (زوج) شامل چرخه تخمینی داده‌ها و روند تخمینی داده‌ها را برمی‌گرداند. سپس با استفاده از بازکردن تاپل، روند را استخراج کرده و روی نمودار رسم می‌کنیم:

```
# بازکردن تاپل برای گرفتن روند و رسم آن
gdp_cycle, gdp_trend = sm.tsa.filters.hpfilter(df['realgdp'])
```

```
# اضافه کردن ستون روند به داده‌ها
df['trend'] = gdp_trend

# رسم نمودار GDP واقعی و روند آن
df[['realgdp','trend']].plot()
```

```
[13] # let's use tuple unpacking to get this trend and plot it on top of this
     gdp_cycle, gdp_trend = sm.tsa.filters.hpfilter(df['realgdp'])

[14] # add in a column for the trend
     df['trend'] = gdp_trend

[15] # plot the real gdp & the trend
     df[['realgdp','trend']].plot()

     <matplotlib.axes._subplots.AxesSubplot at 0x7fc95fc26160>
```



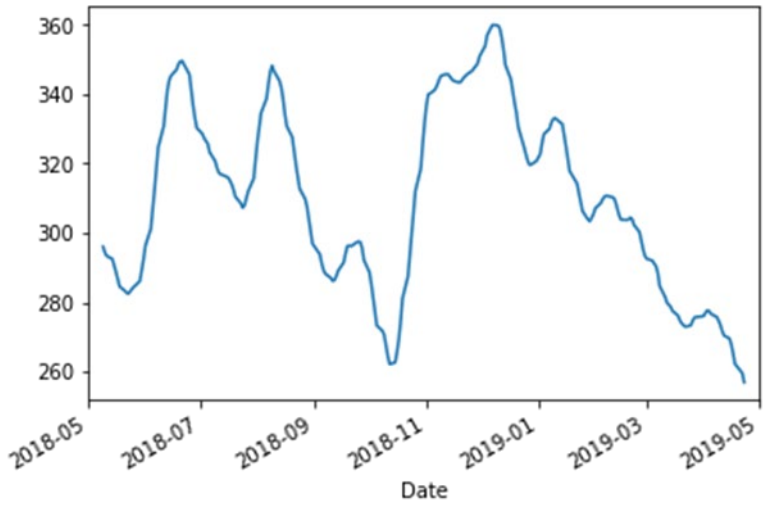
مدل‌های ETS با استفاده از StatsModels

مدل ETS مخفف سه مؤلفه خطا(Error) ، روند (Trend) و فصلی بودن (Seasonality) است. بیایید نگاهی به اجزای ETS در یک مجموعه داده سری زمانی بیندازیم. مدل‌های ETS هر یک از این مؤلفه‌ها (خطا، روند، فصلی بودن) را برای اهداف هموارسازی در نظر می‌گیرند و ممکن است آنها را جمع، ضرب یا برخی از آنها را از مدل حذف کنند. بر اساس این عوامل کلیدی، می‌توانیم مدلی برای برازش داده‌هایمان ایجاد کنیم. پس چگونه می‌توانیم یک سری زمانی را به هر یک از این مؤلفه‌ها تقسیم کنیم؟ تجزیه سری زمانی با استفاده از ETS روشی است برای شکستن یک سری زمانی به این مؤلفه‌ها. در اینجا نحوه انجام تجزیه ETS برای فایل CSV مربوط به TSLA آورده شده است:

```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df['Adj Close'], model='additive', freq=12)
```

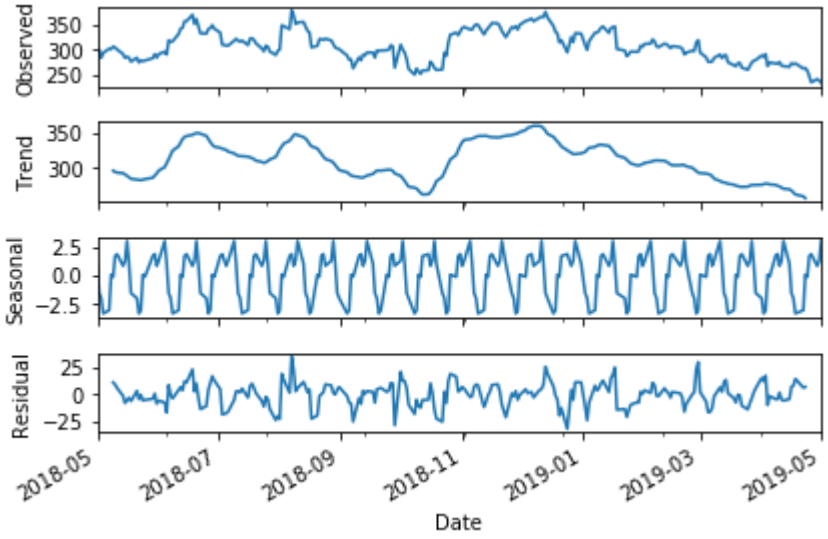
سپس می‌توانیم اجزای مختلف را رسم کنیم، برای مثال روند:

```
result.trend.plot()
```



و همچنین می‌توانیم همه نتایج را به صورت کامل رسم کنیم:

```
result.plot()
```



مدل‌های EWMA

EWMA مخفف عبارت میانگین متحرک وزنی نمایی (Exponentially Weighted Moving Average) است. قبلاً دیدیم که با استفاده از pd.rolling()می‌توانیم مدل ساده‌ای بسازیم که روند یک سری زمانی را توصیف کند. این مدل‌ها به عنوان میانگین‌های متحرک ساده (Simple Moving Averages یا (SMA شناخته می‌شوند.

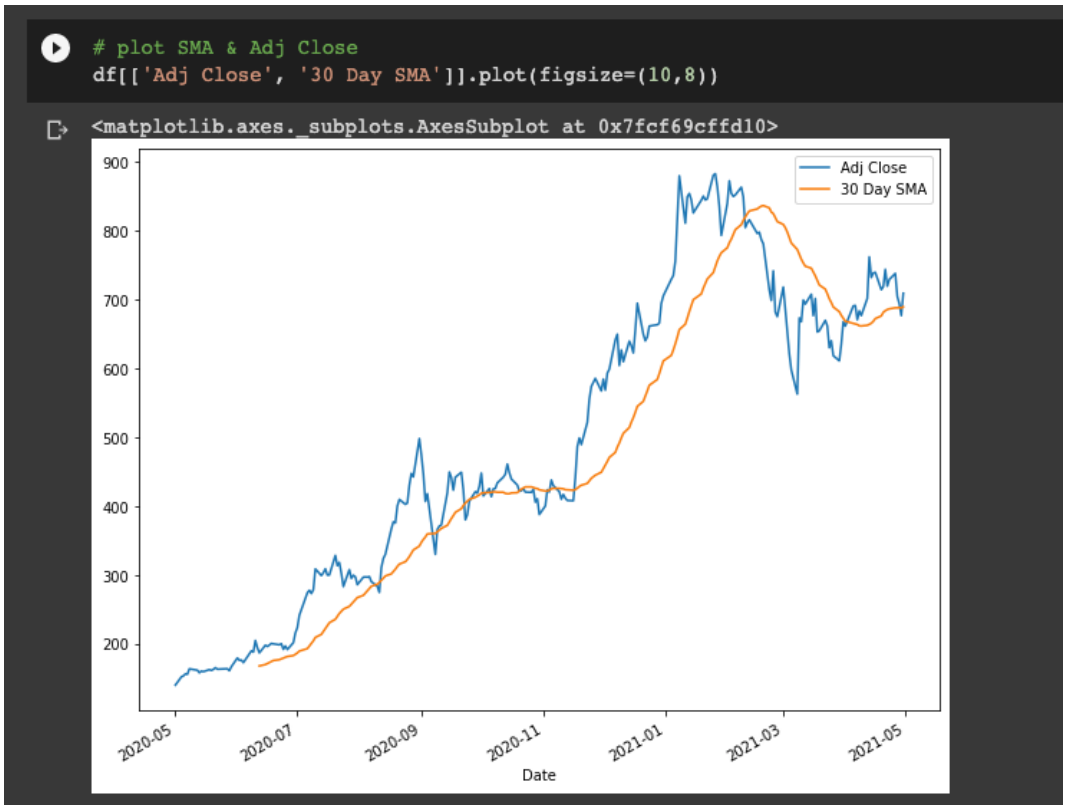
چند مورد از ضعف‌های میانگین‌های متحرک ساده عبارتند از:

- پنجره‌های کوچک‌تر باعث افزایش نویز به جای سیگنال می‌شوند

- همیشه با اندازه پنجره تاخیر (lag) دارند
- به دلیل میانگین‌گیری، هرگز به قله یا دره واقعی داده‌ها نمی‌رسند
- اطلاعاتی درباره رفتار آینده ارائه نمی‌دهند و فقط روندهای گذشته را توصیف می‌کنند
- مقادیر تاریخی بسیار زیاد یا کم می‌توانند میانگین متحرک ساده را به سمت خود منحرف کنند

برای خلاصه، اینجا نحوه محاسبه میانگین متحرک ساده ۳۰ روزه برای TSLA آورده شده است:

```
# ایجاد میانگین متحرک ساده ۱ ماهه از ستون Adj Close
df['30 Day SMA'] = df['Adj Close'].rolling(window=30).mean()

# رسم نمودار Adj Close و SMA
df[['Adj Close', '30 Day SMA']].plot(figsize=(10,8))
```



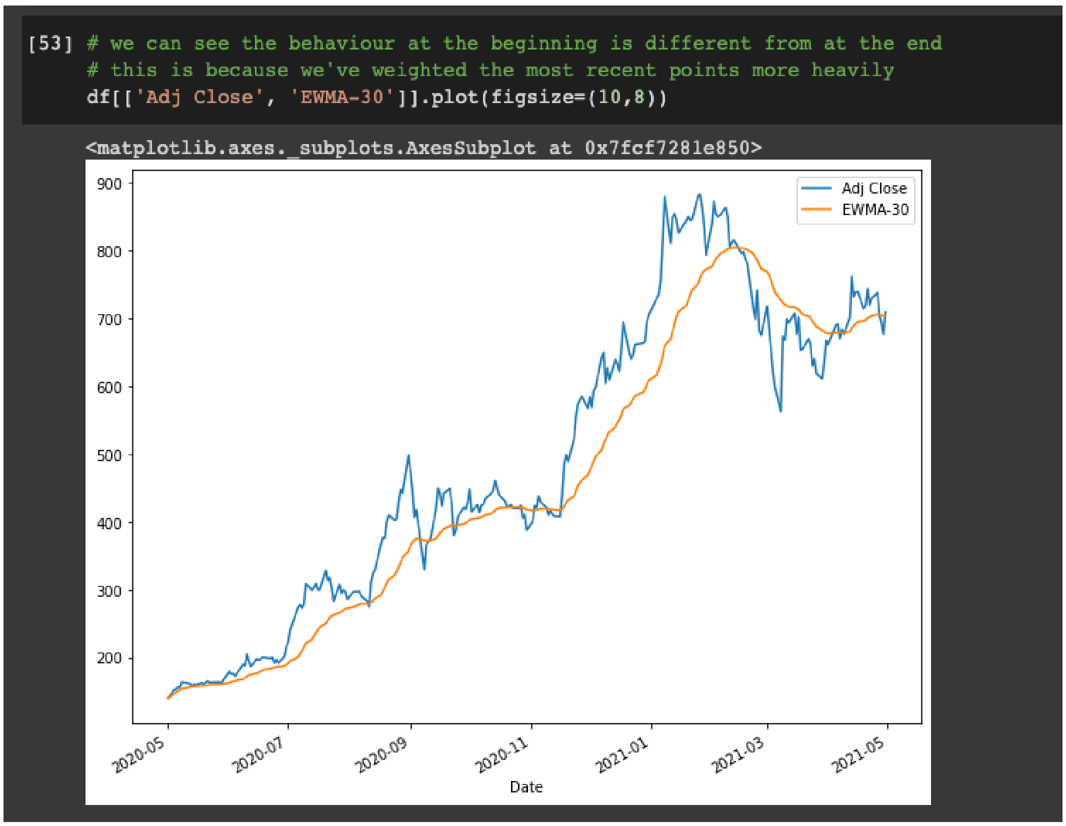
میانگین‌های متحرک وزنی نمایی (EWMA) برخی از این مشکلات را حل می‌کنند، به ویژه:

- EWMAباعث کاهش زمان تاخیر نسبت به SMA می‌شود و وزن بیشتری به مقادیر اخیر می‌دهد

- میزان وزنی که به مقادیر اخیر داده می‌شود، به پارامترهای استفاده شده در EWMA و تعداد دوره‌های پنجره بستگی دارد

این هم نحوه ساخت مدل EWMA:

```
# ایجاد EWMA
df['EWMA-30'] = df['Adj Close'].ewm(span=30).mean()

# رسم نمودار EWMA
df[['Adj Close', 'EWMA-30']].plot(figsize=(10,8))
```



می‌بینیم که رفتار در ابتدای نمودار با انتهای آن متفاوت است — این به این دلیل است که به نقاط اخیر وزن بیشتری داده‌ایم.

ARIMA

مدل‌های ARIMA یکی از رایج‌ترین مدل‌های سری زمانی هستند، اگر می‌خواهید بیشتر درباره مدل‌های ARIMA بدانید، می‌توانید این مقاله از سری گزارش های دوره‌ای هلدینگ صنتا را مطالعه کنید (https://www.santa-co.ir/fa/news/169/%d9%be%db%8c%d8%b4-%d8%a8%db%8c%d9%86%db%8c-%d8%aa%d8%b9%d8%a7%d8%b1%db%8c%d9%81-%d9%86%d8%ad%d9%88%d9%87-%d8%a7%d8%b3%d8%aa%d9%81%d8%a7%d8%af%d9%87-%d8%a7%d8%b2-%d8%a2%d9%86-%d8%af%d8%b1-%da%a9%d8%b3%d8%a8-%d9%88-%da%a9%d8%a7%d8%b1-%d9%88-%d8%b3%d8%b1%d9%85%d8%a7%db%8c%d9%87-%da%af%d8%b0%d8%a7%d8%b1%db%8c-%d8%a8%d9%87-%d9%87%d9%85%d8%b1%d8%a7%d9%87-%db%8c%da%a9-%d9%85%d8%ab%d8%a7%d9%84-%d8%b9%d9%85%d9%84%db%8c).

جمع بندی

در این راهنما، تحلیل سری‌های زمانی برای داده‌های مالی با استفاده از پایتون را مرور کردیم.دیدیم که مسائل سری زمانی با مسائل پیش‌بینی سنتی تفاوت دارند و به بررسی کتابخانه Pandas برای داده‌های سری زمانی و چند تکنیک تحلیل سری زمانی پرداختیم. کتابخانه statsmodels محبوب‌ترین کتابخانه پایتون برای کار با داده‌های سری زمانی است. سپس نحوه استفاده از statsmodels برای مدل‌های ETS (خطا-روند-فصلی بودن) را بررسی کردیم. در نهایت، نحوه استفاده از میانگین‌های متحرک ساده و میانگین‌های متحرک وزنی نمایی SMA و EWMA برای تحلیل سری‌های زمانی را مرور کردیم.

Santa-co.ir

Info@santa-co.ir

۰۲۱-۵۸۱۵۶۱۰۰

خیابان شهیدبهشتی، خیابان یکم بخارست،
پلاک ۲۱

صنایع نانوتک آینده