پروژه‌ی نهایی هوش مصنوعی پیشرفته

پیش‌بینی روند فردای یک سهم در بورس ایران

نگارنده : سحر داستانی اوغانی

شماره دانشجویی : ۹۹۱۱۲۱۰۸

استاد راهنما : دکتر شیرعلی شهرضا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

پیش‌گفتار و توضیح مدل

هر نوع پیش‌بینی‌ای در دنیای واقعی، با توجه به پویا بودن آینده، کار دشواری است. از آنجایی که بازار سهام بسیار غیرقابل پیش-بینی است، سهام‌داران و سرمایه‌گذاران تمایل دارند بر روی سهامی سرمایه‌گذاری کنند که در آینده سودی داشته باشد. ولی اکثر سهام‌داران، به دلیل غیرقابل پیش‌بینی بودن بازار سهام، ریسک می‌کنند و به امید سودی بالاتر می‌مانند.

عوامل بسیاری بر روی قیمت سهم‌ها تاثیر می‌گذارد که می‌تواند باعث افزایش یا کاهش تعداد خریدارها شود. از جمله‌ی این عوامل می‌توان به موارد روبرو اشاره کرد: عرضه و تقاضا، روند بازار، اقتصاد جهانی، نتایج شرکت‌ها، قیمت تاریخی، احساسات عمومی، اطلاعات حساس مالی، محبوبیت (مانند اخبار خوب یا بد مربوط به نام و محصول شرکت).

با وجود تمام موارد ذکر شده، اگر فردی تمامی جنبه‌ها را در نظر بگیرد و قیمتی برای یک سهم پیش‌بینی کند، باز هم تضمینی برای درست بودن آن وجود ندارد. بنابراین پیش‌بینی قیمت یک سهم یک روز جلوتر، کار بسیار دشواری است. برای انجام این کار، نیاز است که برای هر روز، مقایسه‌ای میان قیمت‌های واقعی انجام گیرد تا در نهایت مدلی برای این پیش‌بینی طراحی شود.

داده‌هایی که از نوع سری‌های زمانی هستند، به داده‌های تاریخی یا داده‌های گذشته معروفند. این داده‌ها در طول یک بازه‌ی زمانی جمع‌آوری می‌شوند. این داده‌ها، ترتیبی هستند و از الگوهایی پیروی می‌کنند. از این نوع داده برای پیش‌بینی مقدار متغیری در آینده بر اساس مقدار گذشته‌ی آن استفاده می‌شود.

مدل ARIMA

مدل ARIMA(Auto Regressive Integrated Moving Average) برای پیش‌بینی خروجی بر اساس داده‌های سری است. این مدل در یادگیری ماشین، برای آنالیز و پیش‌بینی قیمت سهم‌های آینده بر اساس قیمت قبلی آن‌ها، استفاده می‌شود. مدل‌های ARIMA از دو نوع seasonal و Non-seasonal تشکیل شدند.

پارامترها:

مدل Non-seasonal ARIMA دارای ۳ متغیر یا پارامتر هستند که در زیر به توضیح خلاصه‌ای از آن‌ها می‌پردازیم:

- P: تعداد تاخیرهای زمانی را نشان می‌دهد. برای مثال اگر P=3 باشد، از ۳ بازه‌ی زمانی قبل برای محاسبه‌ی رگرسیون استفاده می‌شود. P به تنظیم خطی که برای پیش‌بینی سری در نظر گرفته شده است کمک می کند.

  توجه: مدل‌های ARIMA شبیه یک رگرسیون خطی عمل می‌کنند که در آن، متغیرهای پیش‌بینی، تعداد P دوره‌ی قبلی هستند.

- D: در مدل‌های ARIMA، سری‌های زمانی به سری‌های ثابت(stationary)، با استفاده از تکنیک تفاوت، تبدیل می-شوند. پارامتر D، تعداد تفاوت تبدیل‌ها را نمایش می‌دهد.

  داده‌های stationarize داده‌هایی هستند که شرایط زیر را داشته باشند:

  - میانگین سری‌ها نباید تابعی از زمان باشد.
  - واریانس سری‌ها نباید تابعی از زمان باشد.
  - کوواریانس iامین ترم و m+iامین ترم نباید تابعی از زمان باشد.

  دلیل این امر این است که، وقتی یک رگرسیون خطی اجرا می‌شود، فرض بر این است که تمامی مشاهدات مستقل از یک‌دیگر هستند. ولی در سری‌های زمانی، می‌دانیم که مشاهدات، وابستگی زمانی به یکدیگر دارند. از تحقیقات انجام شده، نتیجه گرفتند که نتایج عالی‌ای که از داده‌های رندوم غیر مستقل دریافت می‌شود، از داده‌های رندم ثابت نیز گرفته می‌شود. بنابراین با stationarize کردن داده‌ها، می‌توان از تکنیک‌های رگرسیون استفاده کرد.

  دو متد برای تشخیص stationarity داده‌ها وجود دارد:

  - با visualize کردن داده‌ها می‌توان به راحتی تشخیص داد که میانگین یا واریانس داده‌ها ثابت است یا خیر.
  - راه دقیق‌تر آن، استفاده از تست Dickey-Fuller است.

  برای تبدیل (transorm) داده‌ها به داده‌های stationary می‌توان از متدهای مختلفی بهره برد. از جمله‌ی آن‌ها می-توان به CPI، لگاریتمی، تفاوت اول، تفاوت فصلی و تنظیم فصلی، طرح ACF و PACF اشاره کرد.

  تکنیک تفاوت یا Differencing method یک تکنیک برای تبدیل سری‌های زمانی non-stationary به سری‌های زمانی stationary است. تفاوت اول در این تکنیک، تفاوت میان بازه‌ی زمانی فعلی و بازه‌ی زمانی قبلی است. اگر این مقدار مورد قبول نبود، باید تفاوت دوم را با استفاده از اولی ساخت. این عمل تا زمانی تکرار می‌شود که به یک سری stationary برسیم.

- Q: این متغیر تاخیر مولفه‌ی خطا را نشان می‌دهد، جایی که مولفه‌ی خطا بخشی از سری زمانی است که با trend یا seasonality توضیح داده نمی‌شود.

نحوه‌ی استفاده از مدل

- فراخوانی دیتاست
- پیش‌پردازش آن؛ که می‌تواند شامل طراحی timestampها، تبدیل نوع یا type داده‌ها به یک‌دیگر، تک متغیره کردن سری‌ها باشد. برای مثال، اگر نوع داده‌هایی که نشان‌دهنده‌ی تاریخ هستند از نوع object باشد، باید آن را به نوع datetime تغییر داد.
- تبدیل به سری‌های stationary
- تعریف مقدار متغیر D
- رسم ACF و PACF
- تعریف مقدار متغیر P و Q
- فیت کردن مدل ARIMA بر روی داده‌ها
- پیش‌بینی مقادیر بر روی داده‌های validation

تفصیل جامع کد

ابتدا کتابخانه‌های مورد نیاز را فراخوانی می‌کنیم. موارد استفاده‌ی هر کدام، مقابل آن نوشته شده است.

- warning برای چشم‌پوشی اخطارها
- matplotlib برای رسم نمودار
- sklearn.metrics برای استفاده از ماژول mean_squared_error که برای محاسبه‌ی خطا به کار می‌رود
- statsmodel برای فراخوانی مدل ARIMA
- pytse_client برای دریافت اطلاعات بازار بورس تهران

```
import warnings
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.arima_model import ARIMA
import pytse_client as tse
```

سپس متغیرهای مورد استفاده در برنامه را تعریف می‌کنیم. در این کد، ۷۰٪ از داده‌ها را به داده‌های آموزشی و ۳۰٪ باقی‌مانده را به داده‌های تست اختصاص دادیم.

در ابتدای برنامه، طول داده‌های آموزش، تست و کل دیتاست را صفر در نظر می‌گیریم. و تعداد نقاطی از دیتاست که به عنوان نقاط برگشت استفاده می‌کنیم را، ۳ در نظر می‌گیریم.

```
TRAINING_PERCENTAGE = 0.7
TESTING_PERCENTAGE = 1 - TRAINING_PERCENTAGE
NUMBER_OF_PREVIOUS_DATA_POINTS = 3
LENGTH_DATA_SET = 0
TRAINING_SET_LENGTH = 0
TESTING_SET_LENGTH = 0
```

در مرحله‌ی بعد، دیتاست را با توجه به درصدهای تعریف شده در مرحله‌ی قبل تقسیم می‌کنیم. برای این کار از تابعی تحت عنوان training_testing_buckets استفاده می‌کنیم. خروجی این تابع مجموعه‌های آموزش و تست برنامه خواهد بود.

```
def training_testing_buckets(raw_data, training_percentage, testing_percentage):
    global TRAINING_SET_LENGTH, TESTING_SET_LENGTH
    TRAINING_SET_LENGTH = int(LENGTH_DATA_SET * training_percentage)
    TESTING_SET_LENGTH = LENGTH_DATA_SET - TRAINING_SET_LENGTH
    training_set, testing_set = raw_data[0:TRAINING_SET_LENGTH], raw_data[TRAINING_SET_LENGTH:LENGTH_DATA_SET]
    return training_set, testing_set
```

در ادامه تابعی تعریف می‌کنیم که بتواند کارایی مدل را ارزیابی کند. این تابع از دو مقدار برای انجام این کار استفاده می‌کند. یکی از آن‌ها قیمت واقعی نماد و دیگری قیمت پیش‌بینی شده‌ی آن است. که مشخصا هر دوی آن‌ها جزو تست دیتاست است. برای انجام ارزیابی، در طول دیتاست تست حرکت می‌کنیم و برای هر نماد، مقدار predict و actual را با تفاضل مقدار یکی بالاتر آن بدست می‌آوریم. اگر هر دو مقدار predict و actual، مثبت یا هردو منفی بود، یعنی به درستی پیش‌بینی شده است. پس میزان counter را یکی فراتر می‌بریم. در انتها درصد پیش‌بینی‌های درست را تا دورقم اعشار نمایش می‌دهیم. می‌توان این عمل را با استفاده از mean_squared_error نیز انجام داد که به صورت کامنت در کد زیر آمده است.(این عمل خطا را محاسبه می‌کند)

```python
def evaluate_performance_arima(testing_actual, testing_predict):
    counter = 0
    for i in range(len(testing_actual)-1):
        predict=testing_predict[i+1]- testing_predict[i]
        actual = testing_actual[i+1]-testing_actual[i]
        if (actual > 0 and predict > 0) or (actual < 0 and predict < 0):
            counter+=1
    counter = round((counter / (len(testing_actual)-1))*100,2)
    return counter
    # return mean_squared_error(testing_actual, testing_predict)
```

در این قسمت، نمودار fit شدن مدل بر روی داده‌های نماد مورد استفاده، رسم می‌شود. تابعی تحت عنوان plot_arima برای این عمل به کار رفته است. این تابع، داده‌های تست واقعی و پیش‌بینی شده را به ترتیب با رنگ‌های آبی و سبز نمایش می‌دهد.

```python
def plot_arima(currency, testing_actual, testing_predict):
    actual = pyplot.plot(testing_actual, label="Actual data points", color="blue")
    testing = pyplot.plot(testing_predict, label="Testing prediction", color="green")
    pyplot.ylabel('currency values for 1 currency')
    pyplot.xlabel('number of days')
    pyplot.title( currency + ' : actual vs predicted ')
    pyplot.legend()
    pyplot.show()
    pyplot.clf()
```

برای عملیات بالا، نیاز به یک دیتاست واقعی برای پیاده‌سازی داریم. همانطور که در قسمت فراخوانی کتابخانه‌ها توضیح داده شد، می‌توان اطلاعات بازار بورس تهران را از pytse_client دریافت کرد. برای دانلود سابقه نماد باید از تابع download استفاده کرد. این تابع یک یا چند نماد را گرفته و دیتای آن‌ها را برمی‌گرداند و قابلیت دخیره‌ی اطلاعات در قالب csv را نیز دارد(write_to_csv). این تابع سابقه‌ی سهام را دانلود می‌کند و در نهایت یک دیکشنری برمی‌گرداند که با استفاده از نماد سهم، می‌شود سابقه‌ی آن را دریافت کرد.

برای گرفتن اطلاعات یک سهم، باید از کلاس Ticker استفاده کرد. هنگامی که به این کلاس، اسم یک سهم داده شود، اطلاعات آن را از سایت tsetmc.com دریافت می‌کند و آن‌ها را در دسترس قرار می‌دهد. سابقه‌ی قیمت یک سهم با استفاده از ماژول history از کلاس شی(ticker) ساخته شده از کلاس Ticker، قابل دستیابی است.

با توجه به توضیحات بالا، تابعی با نام load_data_set برای دریافت دیتاست از سایت مذکور به شیوه‌ی زیر نوشته شده است.

```python
def load_data_set(currency):
    tse.download(symbols=stock_name)
    ticker = tse.Ticker(stock_name)
    data = ticker.history
    raw_data = data.close.values.tolist()
    global LENGTH_DATA_SET
    LENGTH_DATA_SET = len(raw_data)
    return raw_data
```

برای این که بتوانیم مدل arima را روی دیتاست اعمال کنیم، نیاز است که آن را ساخته و پیش‌بینی کنیم. برای انجام این کار، داده‌های ترین و تست را در نظر می‌گیریم. یکی یکی داده‌های تست را اضافه می‌کنیم و روی آن تست انجام می‌دهیم. برای مثال، می‌دانیم داده‌ی ۳۴ام مثبت است یا منفی، با استفاده از این داده و بقیه‌ی داده‌های قبلی، arima model را آموزش می‌دهیم. سپس forcast را برای آن روز انجام می‌دهیم. متوجه می‌شویم مثبت می‌گیرد یا منفی. سپس آن نتیجه‌ی forcast شده را به مجموعه‌ی testing_predict اضافه می‌کنیم و نتیجه‌ی آن را به training_predict اضافه می‌کنیم. این کار را به ازای تمامی اعضای testing_set انجام می‌دهیم. اگر کامنت را اجرا کنیم، خواهیم دید که تخمین چه بوده است. این کار را برای دادن دقت انجام می‌دهیم. درواقع بدانیم forcastها درست پیش‌بینی شدند یا غلط.

```python
def build_model_predict_arima(training_set, testing_set):
    testing_predict = list()
    training_predict = list(training_set)
    for testing_set_index in range(TESTING_SET_LENGTH):
        arima = ARIMA(training_predict, order=(5, 1, 0))
        arima_model = arima.fit(disp=0)
        forecasting = arima_model.forecast()[0].tolist()[0]
        testing_predict.append(forecasting)
        training_predict.append(testing_set[testing_set_index])
        # print("Predicted = ", testing_predict[-1], "Expected = ", testing_set[testing_set_index])
    print('predicting...')
    print("---------------------------")
    forcast = arima_model.forecast()[0]
    print('The prediction for the next day:', forcast)
    if forcast- testing_set[-1] > 0 : print("last result = +1")
    else :  print("last result = -1")
    print("---------------------------")
    return testing_predict
```

تابع نهایی که در برنامه تعریف شده است، تمامی عملیات بالا را به یکباره دور هم جمع می‌کند و نتیجه‌ی نهایی را عرضه می‌کند. این تابع نام یک سهم را به عنوان ورودی دریافت می‌کند. سپس دیتاست را لود می‌کند و دیتاست مذکور را به دو دسته‌ی تست و ترین با درصد ۳۰ و ۷۰ تقسیم می‌کند. در ادامه مدل arima را طبق روش گفته شده می‌سازد و آن را آموزش می‌دهد. سپس به وسیله‌ی تابع evaluate performance، کارایی مدل ساخته شده را ارزیابی می‌کند و درصد دقت آن را در متغیر mse_arima نمایش می‌دهد.(دقت داده‌های تست). در انتها نیز نتایج حاصل را بر روی نمودار رسم می‌کند.

```python
def arima_model(currency):
    print('\nARIMA Model')

    print('loading the dataset...')
    raw_data = load_data_set(currency)

    print('splitting training and testing set...')
    training_actual_arima, testing_actual_arima = training_testing_buckets(raw_data, TRAINING_PERCENTAGE,
                                                                            TESTING_PERCENTAGE)

    print('building and training model...')
    testing_predict_arima = build_model_predict_arima(training_actual_arima, testing_actual_arima)

    print('evaluating performance...')
    mse_arima = evaluate_performance_arima(testing_actual_arima, testing_predict_arima)
    print('Testing Accuracy: ', mse_arima,"%")

    print("------------------------")
    print('plotting the graph...')
    plot_arima(currency, testing_actual_arima, testing_predict_arima)

    print('done...')
    return raw_data, testing_predict_arima
```

می‌توان تابع نهایی را با عملیات زیر فراخوانی کرد. در این قسمت برای نمایش خروجی از سهم "خچرخش" استفاده شده است.

فعالیت اصلی این شرکت عبارت است از تولید و ساخت و واردات سامانه موتور و محرکه خودروها (موتور، گیربکس، جعبه فرمان و غیره). شرکت برای انجام امور خود قادر به واردات مواد مورد نیاز و انجام سایر فعالیت های فنی و بازرگانی مرتبط است.

لازم به ذکر است که می‌توان با خط کدی که به صورت کامنت در سوم آمده است، برنامه را به نحوی درآورد که نام سهم را کاربر تعیین کند.

```python
if __name__ == '__main__':
    warnings.filterwarnings("ignore")
    # stock_name = input('Enter stock name:')
    stock_name = 'خچرخش'
    arima_model(stock_name)  # setting the entry point
```

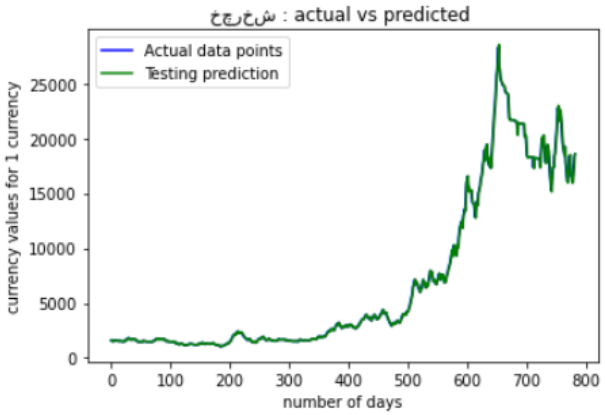
خروجی کد برای داده های نماد "خچرخش" به صورت زیر است :

این خروجی نشان‌دهنده‌ی آن است که این سهم سعودی عمل می‌کند و در روز بعد دارای سود خواهد بود. زیرا +۱ را به عنوان result نمایش می‌دهد.

```
ARIMA Model
loading the dataset...
splitting training and testing set...
building and training model...
predicting...
---------------------------
The prediction for the next day: [18570.16156483]
last result = +1
---------------------------
evaluating performance...
Testing Accuracy:  56.47 %
------------------------
plotting the graph...
```

نمودار fit شدن مدل بر روی همین نماد، به صورت روبرو است :

این نمودار مقدار پولی هر سهم را بر اساس روزهای مورد ارزیابی نمایش می‌دهد. همانطور که مشخص است، این سهم پس از گذراندن ۶۵۰ روز، به بیشترین مقدار خود دست می‌یابد.

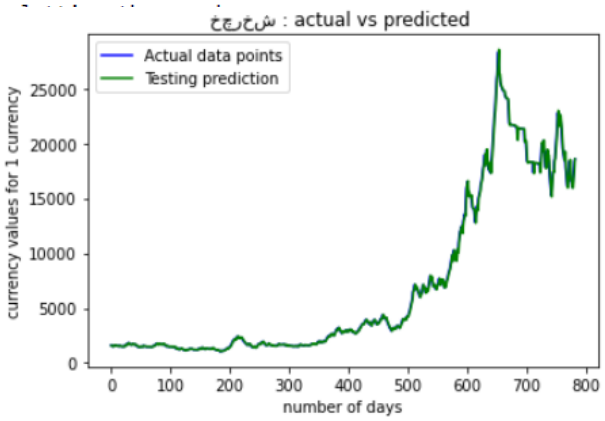


خلاصه‌ی عملکرد

در این پروژه هدف تخمین قیمت یک نماد بورسی بر اساس قیمت های روزهای قبل این نماد بوده است. کد به این صورت عمل می کند که در ابتدا نام هر نمادی را به عنوان ورودی می‌گیرد. سپس اطلاعات مربوط به آن نماد را به کمک کتابخانه pytse_client دانلود می‌کند. سپس ستون قیمت تمام شده را به عنوان قیمت آن روز استخراج می‌کند و اطلاعات را به نسبت ۷۰ به ۳۰ برای داده آموزش و تست تقسیم می‌کند. سپس از مدل ARIMA که برای اولین بار، در مقاله " Time series analysis: forecasting and control. John Wiley & Sons." معرفی شد و در کتابخانه statsmodels پیاده سازی آن به زبان پایتون موجود است، استفاده می‌کند. در این کد پارامترهای مدل را به صورت (5, 1, 0) قرار می دهیم (این مقادیر با آزمون و خطا بدست آمده است). پس از آموزش مدل، دقت مدل را بر روی داده های تست اندازه گیری می‌کنیم و در نهایت برای فردا، قیمت و +۱/-۱ بودن را تخمین می‌زنیم. خروجی کد برای داده های نماد "خچرخش" به صورت زیر است :

```
ARIMA Model
loading the dataset...
splitting training and testing set...
building and training model...
predicting...
---------------------------
The prediction for the next day: [18570.16156483]
last result = +1
---------------------------
evaluating performance...
Testing Accuracy:  56.47 %
------------------------
```

نمودار fit شدن مدل بر روی همین نماد، به صورت روبرو است :



متوسط دقت این مدل، که از میانگین‌گیری بر روی accuracy ۱۰۰ نماد بدست آمده است، مقدار ۶۴٫۹۲٪ می باشد.

منابع

سایت‌های کمکی:

ARIMA Model - Complete Guide to Time Series Forecasting in Python | ML+

Stock price prediction using ARIMA Model | by Dereje Workneh | Medium

Time series Forecasting — ARIMA models | by Sangarshanan | Towards Data Science

Understanding ARIMA (Time Series Modeling) | by Tony Yiu | Towards Data Science

https://www.statsmodels.org/devel/generated/statsmodels.tsa.arima_model.ARIMA.html

کتاب کمکی:

Time-Series-Analysis-Forecasting-and-Control-2015