آزمون **ADF** یا **Augmented Dickey-Fuller Test** یک آزمون آماری برای بررسی **مانایی (stationarity)** سری‌های زمانی است. این آزمون کمک می‌کند تا متوجه شویم آیا یک سری زمانی خاص به مرور زمان ویژگی‌های آماری (مثل میانگین و واریانس) ثابتی دارد یا خیر.

**مفهوم اصلی:**

* **سری زمانی مانا**: میانگین، واریانس و خودهمبستگی آن با زمان تغییر نمی‌کند.
* **سری زمانی نامانا**: این ویژگی‌ها در طول زمان تغییر می‌کنند. به عنوان مثال، سری‌های زمانی که روند یا تغییرات فصلی دارند معمولاً نامانا هستند.

**مراحل آزمون ADF:**

1. فرضیات:
   * **فرض صفر (H₀):** سری زمانی دارای ریشه واحد (نامانا) است.
   * **فرض مقابل (H₁):** سری زمانی مانا است.
2. **خروجی‌های آزمون:**
   * **ADF Statistic (آماره ADF):** آماره‌ای که برای مقایسه با مقادیر بحرانی استفاده می‌شود.
   * **p-value (مقدار p):** احتمال خطا در رد فرض صفر. اگر مقدار p کوچک‌تر از سطح معناداری (مثلاً 0.05) باشد، فرض صفر رد می‌شود.
   * **Critical Values (مقادیر بحرانی):** مقادیری که با آماره ADF مقایسه می‌شوند تا مشخص شود سری زمانی مانا است یا خیر.

**تفسیر نتایج:**

* اگر مقدار p-value کمتر از 0.05 باشد یا آماره ADF از مقدار بحرانی کمتر باشد:
  + **فرض صفر رد می‌شود**.
  + سری زمانی احتمالاً **مانا** است.
* اگر مقدار p-value بیشتر از 0.05 باشد:
  + **فرض صفر پذیرفته می‌شود**.
  + سری زمانی احتمالاً **نامانا** است.

**کاربردها:**

* تحلیل سری‌های زمانی مالی (مثل قیمت سهام یا ارز).
* پیش‌بینی سری‌های زمانی.
* بررسی نیاز به تبدیل داده (مانند تفاضل‌گیری) قبل از مدل‌سازی.

**مثال در کد شما:**

در قطعه کدی که نوشته‌اید، ستون close از داده‌های سری زمانی بررسی شده و مشخص می‌شود که آیا این سری زمانی مانا است یا خیر.

python

Copy code

adf\_result = adfuller(stock\_data['close'])

# Extract the results

adf\_statistic = adf\_result[0]

p\_value = adf\_result[1]

critical\_values = adf\_result[4]

print("ADF Statistic:", adf\_statistic)

print("p-value:", p\_value)

print("Critical Values:", critical\_values)

4o

**You said:**

(-2.4889297712772316, 0.11819111782768371, {'1%': -3.4365635376940706, '5%': -2.86428345899922, '10%': -2.5682307228049686}) خروجیش به این صورته

**ChatGPT said:**

ChatGPT

خروجی ارائه‌شده از آزمون **ADF** شامل اطلاعات زیر است که می‌توان آن را به صورت زیر تحلیل کرد:

**مقادیر خروجی:**

1. **ADF Statistic**: -2.4889
   * این مقدار آماره آزمون ADF است که باید با مقادیر بحرانی مقایسه شود.
2. **p-value**: 0.1182
   * این مقدار نشان‌دهنده احتمال خطای رد نادرست فرض صفر است.
   * اگر مقدار آن کمتر از 0.05 باشد، فرض صفر رد می‌شود.
3. **Critical Values**:
   * مقادیر بحرانی برای سطوح اطمینان مختلف:
     + **1%**: -3.4366
     + **5%**: -2.8643
     + **10%**: -2.5682

**تحلیل خروجی:**

**آیا سری زمانی مانا است؟**

* مقدار **ADF Statistic** برابر با -2.4889 است.
* با مقایسه این مقدار با **Critical Values**:
  + در سطح اطمینان 1%، **ADF Statistic > Critical Value (-3.4366)** (فرض صفر رد نمی‌شود).
  + در سطح اطمینان 5%، **ADF Statistic > Critical Value (-2.8643)** (فرض صفر رد نمی‌شود).
  + در سطح اطمینان 10%، **ADF Statistic > Critical Value (-2.5682)** (فرض صفر رد نمی‌شود).
* مقدار **p-value** برابر با 0.1182 است که از 0.05 بیشتر است.  
  **نتیجه:** فرض صفر رد نمی‌شود.

**نتیجه نهایی:**

سری زمانی مورد بررسی **نامانا** است. برای مانا کردن سری زمانی، معمولاً باید روش‌هایی مانند **تفاضل‌گیری (differencing)** یا **تبدیل لگاریتمی** اعمال شود.

برای بهبود مدل GRU و افزایش دقت پیش‌بینی، می‌توانید موارد زیر را در نظر بگیرید و در کد اعمال کنید:

**1. افزودن ویژگی‌های اضافی**

داده‌های فعلی تنها شامل سه ویژگی است (close, sma\_10, volatility). افزودن ویژگی‌های زیر می‌تواند اطلاعات بیشتری در اختیار مدل قرار دهد:

* **EMA (میانگین متحرک نمایی):**

python

Copy code

df['ema\_10'] = df['close'].ewm(span=10).mean()

* **مومنتوم (Momentum):**

python

Copy code

df['momentum'] = df['close'].diff(periods=5)

* **RSI (شاخص قدرت نسبی):** اگر از کتابخانه ta استفاده کنید:

python

Copy code

import ta

df['rsi'] = ta.momentum.RSIIndicator(df['close'], window=14).rsi()

**2. بهبود ساختار مدل**

**(الف) افزایش تعداد واحدها (units) و لایه‌ها:**

افزایش تعداد نرون‌ها و افزودن یک لایه GRU دیگر می‌تواند مدل را بهبود دهد:

python

Copy code

model\_gru = Sequential([

GRU(units=128, return\_sequences=True, input\_shape=(time\_step, X\_train.shape[2])),

Dropout(0.2),

GRU(units=64, return\_sequences=True),

Dropout(0.2),

GRU(units=32),

Dropout(0.2),

Dense(units=1)

])

**(ب) تغییر مقدار Dropout:**

مقدار فعلی 0.2 ممکن است بیش از حد باشد. آن را به 0.1 یا 0.15 کاهش دهید.

**(ج) استفاده از تنظیمات پیشرفته:**

* افزودن **L2 Regularization**:

python

Copy code

from tensorflow.keras.regularizers import l2

GRU(units=128, return\_sequences=True,

input\_shape=(time\_step, X\_train.shape[2]),

kernel\_regularizer=l2(0.01))

**3. افزایش تعداد اپوک‌ها**

افزایش تعداد اپوک‌ها (epochs) از 50 به 100 یا بیشتر می‌تواند مفید باشد، به شرطی که از **EarlyStopping** استفاده کنید تا از یادگیری بیش از حد جلوگیری شود:

python

Copy code

from tensorflow.keras.callbacks import EarlyStopping

early\_stopping = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)

history = model\_gru.fit(X\_train, Y\_train, epochs=100, batch\_size=32,

validation\_data=(X\_test, Y\_test), verbose=1,

callbacks=[early\_stopping])

**4. تغییر مقیاس‌بندی داده‌ها**

به جای MinMaxScaler از StandardScaler استفاده کنید که برای داده‌های دارای نوسان شدید بهتر عمل می‌کند:

python

Copy code

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(df[['close', 'sma\_10', 'volatility']].values)

**5. افزودن داده‌های تاریخی بیشتر**

مدل‌های سری زمانی به داده‌های تاریخی حساس هستند. اگر داده‌های فعلی محدود باشند، اضافه کردن داده‌های بیشتر می‌تواند کمک کند.

**6. بهینه‌سازی هایپرپارامترها**

با استفاده از ابزارهایی مانند **Keras Tuner**، مقادیر مناسب برای لایه‌ها، تعداد نرون‌ها، نرخ یادگیری و سایر تنظیمات را پیدا کنید:

python

Copy code

from keras\_tuner import RandomSearch

def build\_model(hp):

model = Sequential()

model.add(GRU(units=hp.Int('units', min\_value=32, max\_value=128, step=32),

return\_sequences=True, input\_shape=(time\_step, X\_train.shape[2])))

model.add(Dropout(rate=hp.Float('dropout', min\_value=0.1, max\_value=0.3, step=0.05)))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

return model

tuner = RandomSearch(build\_model, objective='val\_loss', max\_trials=10)

tuner.search(X\_train, Y\_train, epochs=50, validation\_data=(X\_test, Y\_test))

**7. افزودن تابع فعال‌سازی (Activation Function) در خروجی**

به‌صورت پیش‌فرض، لایه Dense خروجی مقدار خطی برمی‌گرداند. با افزودن تابع فعال‌سازی مانند ReLU یا sigmoid می‌توانید دقت را افزایش دهید:

python

Copy code

Dense(units=1, activation='relu')

**8. تغییر تابع بهینه‌سازی**

آزمایش با توابع بهینه‌سازی دیگر، مانند RMSprop یا Adamax:

python

Copy code

model\_gru.compile(optimizer='RMSprop', loss='mean\_squared\_error')

**نسخه بهبود‌یافته مدل:**

با اعمال تغییرات پیشنهادی:

python

Copy code

model\_gru = Sequential([

GRU(units=128, return\_sequences=True, input\_shape=(time\_step, X\_train.shape[2])),

Dropout(0.1),

GRU(units=64, return\_sequences=True),

Dropout(0.15),

GRU(units=32),

Dropout(0.15),

Dense(units=1, activation='relu')

])

model\_gru.compile(optimizer='adam', loss='mean\_squared\_error')

history = model\_gru.fit(X\_train, Y\_train, epochs=100, batch\_size=32,

validation\_data=(X\_test, Y\_test), verbose=1,

callbacks=[EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)])

**نتیجه:**

این تغییرات باید دقت پیش‌بینی مدل شما را بهبود دهند. اگر همچنان نیاز به بهبود دارید، ممکن است مشکل از کیفیت داده‌ها یا حجم داده‌های آموزشی باشد.

**نکات کلیدی:**

1. ویژگی‌های اضافی مانند ema\_10 و momentum به مدل اضافه شده‌اند.
2. از StandardScaler به جای MinMaxScaler استفاده شده است.
3. ساختار مدل GRU بهبود یافته و تعداد واحدها افزایش داده شده است.
4. از **EarlyStopping** استفاده شده تا از overfitting جلوگیری شود.
5. تابع فعال‌سازی ReLU به لایه خروجی اضافه شده است.

این تغییرات باید دقت مدل شما را به طور قابل توجهی بهبود بخشند.

**توضیحات کد:**

1. **future\_steps:** تعداد گام‌هایی که می‌خواهیم پیش‌بینی کنیم (250 روز کاری برای یک سال).
2. **last\_sequence:** داده‌هایی که مدل برای شروع پیش‌بینی نیاز دارد.
3. **تکرار پیش‌بینی:** خروجی هر مرحله به عنوان ورودی مرحله بعدی استفاده می‌شود.
4. **بازگرداندن مقیاس:** داده‌های پیش‌بینی‌شده به مقیاس اصلی تبدیل می‌شوند.
5. **رسم نمودار:** پیش‌بینی‌های آینده با داده‌های تاریخی ترکیب و نمایش داده می‌شود.

این کد به شما کمک می‌کند تا پیش‌بینی یک سال آینده (یا هر بازه دلخواه دیگری) را انجام دهید.

**اگر مسئله طبقه‌بندی نیست و فقط پیش‌بینی مقدار است:**

در این حالت نمی‌توانید از Confusion Matrix استفاده کنید و باید معیارهای **Regressive Evaluation** مانند MSE یا MAE را محاسبه کنید.

برای سری‌های زمانی مانند پیش‌بینی قیمت سهام:

* y\_true می‌تواند مقادیر واقعی قیمت‌ها باشد (داده‌های آزمایشی یا تست).
* y\_pred می‌تواند خروجی مدل شما باشد (پیش‌بینی‌های GRU، LSTM یا ARIMA).

**کد نمونه برای ساخت Confusion Matrix**

فرض کنید:

* y\_true نشان‌دهنده تغییرات واقعی قیمت باشد.
* y\_pred خروجی مدل (افزایش، کاهش یا ثابت‌ماندن) باشد.

**خروجی این کد شامل:**

1. **Confusion Matrix**:
   * هر سلول نشان می‌دهد که چند نمونه از هر دسته (افزایش، کاهش، ثابت‌ماندن) به‌درستی یا اشتباه طبقه‌بندی شده‌اند.
   * محور عمودی: دسته‌های واقعی
   * محور افقی: دسته‌های پیش‌بینی‌شده
2. **Classification Report**:
   * معیارهایی مانند **Precision**، **Recall** و **F1-Score** را برای هر دسته ارائه می‌دهد.

در مورد پیش‌بینی سری‌های زمانی مانند پیش‌بینی قیمت سهام، مفهوم **Confusion Matrix** به‌صورت سنتی معنا ندارد. زیرا Confusion Matrix معمولاً برای مسائل طبقه‌بندی (Classification) استفاده می‌شود، جایی که خروجی دسته‌ای مشخص مانند *"True" یا "False"* است.

اما اگر مسئله شما به نحوی به **طبقه‌بندی** (مثلاً افزایش یا کاهش قیمت) تغییر کند، می‌توان از Confusion Matrix استفاده کرد. به عنوان مثال:

1. **تبدیل مسئله پیش‌بینی به طبقه‌بندی**:  
   به‌جای پیش‌بینی مقادیر دقیق قیمت (Regression)، می‌توانید پیش‌بینی کنید که آیا قیمت در روز آینده:
   * **بالا می‌رود** (+1)
   * **پایین می‌آید** (-1)
   * **ثابت می‌ماند** (0)
2. سپس خروجی مدل‌های ARIMA، LSTM و GRU را به یکی از این سه دسته تبدیل کرده و با مقادیر واقعی مقایسه کنید.

اگر داده‌های شما **دارای روند قوی** یا **نوسانات بالا** است، ممکن است ARIMA عملکرد کمتری نسبت به LSTM و GRU داشته باشد، چون مدل‌های عمیق می‌توانند روابط پیچیده‌تر زمانی را یاد بگیرند.

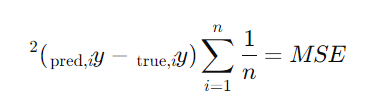
 **مقادیر معیارها را مقایسه کنید**:

* مدلی که **MSE، RMSE، MAE، و MAPE کمتری** داشته باشد، مدل دقیق‌تری است.

 **رسم نمودار**:

* اگر پیش‌بینی‌های مدل به داده‌های واقعی نزدیک‌تر باشد، آن مدل عملکرد بهتری داشته است.

ابتدا باید مقادیر پیش‌بینی‌شده و واقعی برای هر مدل را آماده کنید. فرض کنید:

* y\_true داده‌های واقعی است.
* y\_pred\_arima پیش‌بینی‌های مدل ARIMA است.
* y\_pred\_lstm پیش‌بینی‌های مدل LSTM است.
* y\_pred\_gru پیش‌بینی‌های مدل GRU است.
* **MSE (Mean Squared Error)**: میانگین مربع خطاها؛ هرچه مقدار آن کمتر باشد، دقت مدل بیشتر است.
* 
* **RMSE (Root Mean Squared Error)**: جذر میانگین مربع خطاها؛ مشابه MSE است ولی در مقیاس داده‌های واقعی تفسیر می‌شود.
* A black text on a white background

  Description automatically generated
* **MAE (Mean Absolute Error)**: میانگین قدر مطلق خطاها؛ هرچه مقدار آن کمتر باشد، مدل بهتر عمل کرده است.
* A black and white math symbols

  Description automatically generated with medium confidence
* **MAPE (Mean Absolute Percentage Error)**: میانگین درصد خطاها نسبت به مقادیر واقعی؛ مناسب برای مقایسه عملکرد مدل‌ها روی داده‌هایی با مقیاس متفاوت.

A mathematical equation with numbers and symbols

Description automatically generated

ببین من میخوام با استفاده از مدل های ARIMAX, GRU, LSTM قیمت سهام شستا رو پیش بینی کنم برای یکسال آینده مثلا باید چیکار کنم؟

**ChatGPT said:**

ChatGPT

برای پیش‌بینی قیمت سهام **شستا** با استفاده از مدل‌های **ARIMAX، GRU و LSTM**، باید مراحل زیر را دنبال کنید. این فرآیند شامل آماده‌سازی داده‌ها، انتخاب ویژگی‌ها، آموزش مدل‌ها، و پیش‌بینی است.

**1. آماده‌سازی داده‌ها**

**الف) دریافت و تمیز کردن داده‌ها**

* **داده‌های تاریخی:** شامل قیمت باز (open)، قیمت بسته (close)، بالاترین (high) و پایین‌ترین (low)، حجم معاملات (volume)، و تاریخ (date).
* **ویژگی‌های ماکرو اقتصادی (ARIMAX):** داده‌هایی مثل نرخ ارز، تورم، یا متغیرهای اقتصادی که ممکن است بر قیمت سهام تأثیر بگذارد.

**ب) پیش‌پردازش**

1. **تبدیل تاریخ:** ستون date را به فرمت تاریخ (datetime) تبدیل کنید.
2. **مرتب‌سازی:** مرتب کردن داده‌ها بر اساس date:

python

Copy code

df = df.sort\_values(by='date')

1. **ایجاد ویژگی‌ها:**
   * **میانگین متحرک (Moving Average):** میانگین قیمت در چند روز گذشته (مثلاً 7 یا 30 روز).
   * **انحراف معیار (Volatility):** نوسانات قیمت.
   * **ویژگی‌های دیگر:** تغییرات روزانه، درصد تغییر قیمت.

**ج) تقسیم داده‌ها**

* **داده‌های آموزشی:** شامل ~80% داده‌های اولیه (مثلاً 3 سال گذشته).
* **داده‌های تست:** شامل ~20% داده‌ها برای ارزیابی مدل‌ها.
* **داده‌های پیش‌بینی:** برای یک سال آینده.

**2. مدل‌سازی با ARIMAX**

ARIMAX (Autoregressive Integrated Moving Average with Exogenous Variables) مدلی کلاسیک برای داده‌های سری زمانی است.

**الف) انتخاب پارامترها**

1. **پارامترهای ARIMA:**
   * p: تعداد تأخیرها.
   * d: تعداد تفاضل‌گیری‌ها (برای ایستا کردن سری).
   * q: تعداد میانگین متحرک.
2. **ویژگی‌های خارجی (X):** ویژگی‌های ماکرو یا اقتصادی.

**ب) پیاده‌سازی**

python

Copy code

from statsmodels.tsa.statespace.sarimax import SARIMAX

# مدل ARIMAX

model = SARIMAX(df['close'], exog=df[['macro\_feature1', 'macro\_feature2']], order=(p, d, q))

result = model.fit()

forecast = result.forecast(steps=365, exog=macro\_features\_next\_year)

**3. مدل‌سازی با GRU و LSTM**

این مدل‌ها از شبکه‌های عصبی بازگشتی (RNN) هستند که برای سری‌های زمانی مناسب‌اند.

**الف) پیش‌پردازش داده‌ها**

1. **مقیاس‌گذاری (Scaling):** داده‌ها را با **MinMaxScaler** یا **StandardScaler** مقیاس‌گذاری کنید.

python

Copy code

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(df[['close']])

1. **ایجاد داده‌های زمانی:** داده‌ها را به شکل ورودی‌های سری زمانی آماده کنید:

python

Copy code

def create\_sequences(data, seq\_length):

X, y = [], []

for i in range(len(data) - seq\_length):

X.append(data[i:i + seq\_length])

y.append(data[i + seq\_length])

return np.array(X), np.array(y)

seq\_length = 30 # مثلاً 30 روز گذشته

X, y = create\_sequences(scaled\_data, seq\_length)

**ب) طراحی مدل‌ها**

1. **مدل GRU:**

python

Copy code

from keras.models import Sequential

from keras.layers import GRU, Dense

model\_gru = Sequential()

model\_gru.add(GRU(50, return\_sequences=True, input\_shape=(seq\_length, 1)))

model\_gru.add(GRU(50))

model\_gru.add(Dense(1))

model\_gru.compile(optimizer='adam', loss='mse')

model\_gru.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

1. **مدل LSTM:** مشابه GRU، فقط لایه‌ها را به LSTM تغییر دهید.

**ج) پیش‌بینی**

برای پیش‌بینی یک سال آینده:

python

Copy code

forecast\_gru = model\_gru.predict(future\_data)

forecast\_gru = scaler.inverse\_transform(forecast\_gru)

**4. ارزیابی مدل‌ها**

1. **شاخص‌های ارزیابی:**
   * **MSE (Mean Squared Error):**
   * **MAE (Mean Absolute Error):**
   * **R² Score:**

python

Copy code

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

mse = mean\_squared\_error(y\_test, predictions)

mae = mean\_absolute\_error(y\_test, predictions)

r2 = r2\_score(y\_test, predictions)

1. **مقایسه مدل‌ها:** مدل‌ها را بر اساس شاخص‌های بالا مقایسه کنید.

**5. پیش‌بینی برای یک سال آینده**

* داده‌های جدید (تغییرات روزانه و ماکرو) را آماده کنید.
* از مدل‌های ARIMAX، GRU، و LSTM برای پیش‌بینی استفاده کنید.
* پیش‌بینی‌ها را تجسم کنید:

python

Copy code

import matplotlib.pyplot as plt

plt.plot(forecast\_arimax, label='ARIMAX')

plt.plot(forecast\_gru, label='GRU')

plt.plot(forecast\_lstm, label='LSTM')

plt.legend()

plt.show()

بله، حتما! در اینجا به طور کامل‌تر و با جزئیات بیشتر هر مرحله را توضیح می‌دهم تا بتوانید پیش‌بینی قیمت سهام شستا را با استفاده از مدل‌های **ARIMAX، GRU** و **LSTM** انجام دهید.

**1. آماده‌سازی داده‌ها**

**الف) دریافت و تمیز کردن داده‌ها**

اولین قدم در پیش‌بینی، جمع‌آوری و آماده‌سازی داده‌ها است. شما باید داده‌های تاریخچه قیمت سهام شستا را برای چندین سال جمع‌آوری کنید. این داده‌ها معمولاً شامل ویژگی‌هایی مانند date, close, open, high, low, volume و ... هستند.

برای مدل ARIMAX شما به **ویژگی‌های خارجی** نیز نیاز دارید، به عنوان مثال، داده‌هایی مانند **نرخ ارز**، **تورم**، یا **داده‌های اقتصادی کلان** که می‌توانند تأثیر زیادی بر قیمت سهام داشته باشند.

**نمونه داده‌ها:**

yaml

Copy code

date open high low close volume

2020-01-01 1000 1100 900 1050 5000

2020-01-02 1050 1150 1000 1100 5200

...

**ب) پیش‌پردازش داده‌ها**

1. **تبدیل تاریخ به فرمت datetime:** تاریخ‌ها را به فرمت datetime تبدیل کنید تا بتوانید به راحتی با آن‌ها کار کنید.

python

Copy code

df['date'] = pd.to\_datetime(df['date'])

1. **مرتب‌سازی داده‌ها:** داده‌ها باید بر اساس تاریخ مرتب شوند تا بتوانید روند زمانی آن‌ها را به درستی تجزیه و تحلیل کنید.

python

Copy code

df = df.sort\_values(by='date')

1. **محاسبه ویژگی‌ها:**
   * **میانگین متحرک:** می‌توانید میانگین متحرک (مثلاً 7 یا 30 روزه) را برای مشخص کردن روند کلی قیمت اضافه کنید.

python

Copy code

df['moving\_avg\_30'] = df['close'].rolling(window=30).mean()

* + **انحراف معیار (Volatility):** برای شناسایی نوسانات قیمت می‌توانید انحراف معیار را محاسبه کنید.

python

Copy code

df['volatility'] = df['close'].rolling(window=30).std()

1. **تقسیم داده‌ها به داده‌های آموزشی و تستی:** معمولاً از 70-80% داده‌ها برای آموزش مدل و از بقیه برای تست مدل استفاده می‌شود.

python

Copy code

train\_size = int(len(df) \* 0.8)

train, test = df[:train\_size], df[train\_size:]

**ج) انتخاب ویژگی‌ها**

برای مدل ARIMAX، به ویژگی‌های اضافی (exogenous variables) نیاز دارید. این ویژگی‌ها می‌توانند شامل **نرخ ارز، تورم، یا شاخص‌های اقتصادی دیگر** باشند که می‌توانند بر قیمت سهام تأثیر بگذارند. این ویژگی‌ها باید در قالب یک DataFrame یا آرایه numpy به مدل ARIMAX داده شوند.

**2. مدل ARIMAX**

مدل **ARIMAX** ترکیبی از مدل **ARIMA** (Autoregressive Integrated Moving Average) با ویژگی‌های خارجی است. ARIMA برای مدل‌سازی روند سری‌های زمانی ایستا استفاده می‌شود و **X** در ARIMAX به معنی ویژگی‌های خارجی است که می‌تواند بر پیش‌بینی تأثیر بگذارد.

**الف) انتخاب پارامترهای ARIMAX**

برای استفاده از ARIMAX باید پارامترهای زیر را انتخاب کنید:

* p: تعداد تأخیرهای خودرگرسیو (AR).
* d: تعداد تفاضل‌ها برای ایستا کردن سری زمانی.
* q: تعداد میانگین متحرک (MA).

این مقادیر را می‌توانید با استفاده از **تحلیل خودکار** یا آزمون‌های مختلف مانند **ACF** و **PACF** انتخاب کنید.

**ب) مدل‌سازی با ARIMAX**

python

Copy code

from statsmodels.tsa.statespace.sarimax import SARIMAX

# مدل ARIMAX

model = SARIMAX(df['close'], exog=df[['macro\_feature1', 'macro\_feature2']], order=(p, d, q))

result = model.fit()

# پیش‌بینی قیمت‌ها برای 365 روز آینده

forecast = result.forecast(steps=365, exog=macro\_features\_next\_year)

در اینجا، df[['macro\_feature1', 'macro\_feature2']] شامل ویژگی‌های اقتصادی است که به مدل داده می‌شوند. پس از آموزش مدل، می‌توانید پیش‌بینی‌هایی برای یک سال آینده (یا هر بازه زمانی دیگر) انجام دهید.

**3. مدل GRU و LSTM**

مدل‌های **GRU** (Gated Recurrent Unit) و **LSTM** (Long Short-Term Memory) از انواع مدل‌های **شبکه‌های عصبی بازگشتی (RNN)** هستند که به خوبی برای پیش‌بینی داده‌های سری زمانی مانند قیمت سهام مناسب هستند.

**الف) پیش‌پردازش داده‌ها برای شبکه‌های عصبی**

1. **مقیاس‌گذاری (Scaling):** برای جلوگیری از تاثیر بزرگ بودن برخی ویژگی‌ها (مثل قیمت‌های بالا)، باید داده‌ها را مقیاس‌گذاری کنید.

python

Copy code

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(df[['close']])

1. **ایجاد داده‌های ورودی و خروجی برای مدل عصبی:** برای مدل‌های RNN باید داده‌ها به شکل **دنباله‌های زمانی** آماده شوند. برای مثال، می‌توانید از 30 روز گذشته برای پیش‌بینی قیمت روز آینده استفاده کنید:

python

Copy code

def create\_sequences(data, seq\_length):

X, y = [], []

for i in range(len(data) - seq\_length):

X.append(data[i:i + seq\_length])

y.append(data[i + seq\_length])

return np.array(X), np.array(y)

seq\_length = 30 # 30 روز گذشته

X, y = create\_sequences(scaled\_data, seq\_length)

**ب) طراحی و آموزش مدل GRU و LSTM**

1. **مدل GRU:**

python

Copy code

from keras.models import Sequential

from keras.layers import GRU, Dense

model\_gru = Sequential()

model\_gru.add(GRU(50, return\_sequences=True, input\_shape=(seq\_length, 1)))

model\_gru.add(GRU(50))

model\_gru.add(Dense(1))

model\_gru.compile(optimizer='adam', loss='mse')

model\_gru.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

1. **مدل LSTM:** مشابه مدل GRU، فقط از لایه‌های LSTM به جای GRU استفاده می‌شود.

**ج) پیش‌بینی قیمت‌ها**

برای پیش‌بینی قیمت‌ها می‌توانید از مدل‌های GRU یا LSTM استفاده کنید:

python

Copy code

forecast\_gru = model\_gru.predict(future\_data)

forecast\_gru = scaler.inverse\_transform(forecast\_gru)

**4. ارزیابی مدل‌ها**

پس از آموزش مدل‌ها، باید آن‌ها را ارزیابی کنید تا بهترین مدل را انتخاب کنید.

**الف) شاخص‌های ارزیابی:**

1. **MSE (Mean Squared Error):** میانگین مربع تفاوت پیش‌بینی‌ها و مقادیر واقعی.
2. **MAE (Mean Absolute Error):** میانگین مقدار مطلق تفاوت پیش‌بینی‌ها و مقادیر واقعی.
3. **R² Score:** میزان تناسب مدل با داده‌های واقعی.

python

Copy code

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

mse = mean\_squared\_error(y\_test, predictions)

mae = mean\_absolute\_error(y\_test, predictions)

r2 = r2\_score(y\_test, predictions)

**ب) مقایسه مدل‌ها**

با استفاده از این شاخص‌ها، می‌توانید مدل‌های **ARIMAX**، **GRU** و **LSTM** را مقایسه کنید و بهترین مدل را انتخاب کنید.

**5. پیش‌بینی برای یک سال آینده**

برای پیش‌بینی یک سال آینده، داده‌های جدیدی مانند شاخص‌های اقتصادی و ویژگی‌های ماکرو را آماده کنید و سپس پیش‌بینی‌های هر مدل را ترکیب کنید. نمودار پیش‌بینی‌ها را رسم کنید تا بتوانید روند قیمت سهام شستا را مشاهده کنید.

python

Copy code

import matplotlib.pyplot as plt

plt.plot(forecast\_arimax, label='ARIMAX')

plt.plot(forecast\_gru, label='GRU')

plt.plot(forecast\_lstm, label='LSTM')

plt.legend()

plt.show()

**نتیجه‌گیری**

با این مراحل می‌توانید قیمت سهام شستا را برای یک سال آینده پیش‌بینی کنید. هر مدل نقاط قوت و ضعف خود را دارد، بنابراین بهتر است که مدل‌ها را با یکدیگر مقایسه کنید و از ترکیب آن‌ها برای پیش‌بینی دقیق‌تر استفاده کنید.

4o mini

**You said:**

برای پیش بینی یکسال آینده کدوم بخش دیتاها رو بهش دادی؟

**ChatGPT said:**

ChatGPT

برای پیش‌بینی یک سال آینده، باید ابتدا ویژگی‌های مختلف از داده‌های تاریخی و ویژگی‌های خارجی (exogenous features) که ممکن است تأثیرگذار باشند، به مدل بدهید. این ویژگی‌ها می‌توانند شامل قیمت‌های گذشته، میانگین‌های متحرک، نوسانات قیمت، و ویژگی‌های اقتصادی باشند. در ادامه، بخش‌های مختلف داده‌هایی که باید برای پیش‌بینی یک سال آینده استفاده کنید، توضیح داده می‌شود.

**1. ویژگی‌های ورودی برای مدل‌های ARIMAX، GRU و LSTM**

این ویژگی‌ها بسته به نوع مدل و هدف پیش‌بینی ممکن است کمی متفاوت باشند. اما به‌طور کلی، شما نیاز دارید تا ویژگی‌های زیر را برای پیش‌بینی یک سال آینده فراهم کنید:

**الف) ویژگی‌های تاریخچه‌ای قیمت‌ها (برای مدل ARIMAX، GRU، LSTM)**

* **قیمت‌های گذشته:** این شامل داده‌های قیمت سهام در روزهای قبل است که برای پیش‌بینی قیمت‌های آینده به مدل داده می‌شود.
  + برای مدل **ARIMAX**، به سری زمانی close به همراه ویژگی‌های خارجی نیاز دارید.
  + برای مدل‌های **GRU** و **LSTM**، داده‌های تاریخی قیمت‌ها (مانند close) به همراه ویژگی‌های خارجی به مدل داده می‌شوند.

**ب) ویژگی‌های خارجی (برای مدل ARIMAX)**

برای مدل ARIMAX که از ویژگی‌های خارجی برای پیش‌بینی استفاده می‌کند، می‌توانید داده‌های اقتصادی و اجتماعی زیر را استفاده کنید:

* **نرخ ارز**، **تورم**، **شاخص کل بازار سهام**، **نرخ سود بانکی**، و سایر شاخص‌های اقتصادی که می‌توانند تأثیرگذار باشند.

این ویژگی‌ها به صورت اضافی به مدل داده می‌شوند تا تأثیرات غیر مستقیم بر قیمت سهام شستا را مدنظر قرار دهد.

**ج) ویژگی‌های فنی (برای مدل‌های GRU و LSTM)**

برای مدل‌های GRU و LSTM، می‌توانید ویژگی‌های فنی مانند **میانگین متحرک**، **انحراف معیار (volatility)**، و **lag features** را اضافه کنید:

* **میانگین متحرک (Moving Average):** می‌توانید میانگین متحرک 7 روزه یا 30 روزه را محاسبه کنید.

python

Copy code

df['moving\_avg\_30'] = df['close'].rolling(window=30).mean()

* **انحراف معیار (Volatility):** محاسبه انحراف معیار 30 روزه برای اندازه‌گیری نوسانات قیمت.

python

Copy code

df['volatility'] = df['close'].rolling(window=30).std()

**د) تقسیم داده‌ها به داده‌های آموزشی و تستی**

قبل از اینکه داده‌ها را به مدل‌ها بدهید، باید داده‌ها را به دو مجموعه آموزشی و تستی تقسیم کنید:

* **داده‌های آموزشی:** معمولاً 80% داده‌ها برای آموزش مدل استفاده می‌شود.
* **داده‌های تستی:** 20% داده‌ها برای ارزیابی مدل و پیش‌بینی‌های آن استفاده می‌شود.

python

Copy code

train\_size = int(len(df) \* 0.8)

train, test = df[:train\_size], df[train\_size:]

**ه) مقیاس‌گذاری داده‌ها برای مدل‌های GRU و LSTM**

برای مدل‌های **GRU** و **LSTM**، معمولاً از **مقیاس‌گذاری (scaling)** داده‌ها استفاده می‌شود تا مقادیر ورودی یکسان‌تر و منسجم‌تر شوند. معمولاً از **MinMaxScaler** برای این کار استفاده می‌شود:

python

Copy code

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(df[['close']])

سپس داده‌های مقیاس‌شده را به مدل‌های **GRU** یا **LSTM** وارد می‌کنید.

**و) داده‌های آینده برای پیش‌بینی**

برای پیش‌بینی قیمت‌های یک سال آینده، شما به داده‌های **ویژگی‌های خارجی جدید** و **قیمت‌های گذشته** (اگر در دسترس باشند) نیاز خواهید داشت. به عنوان مثال:

* **قیمت‌های سهام در ماه‌های گذشته** و **ویژگی‌های اقتصادی** در آینده برای پیش‌بینی یک سال آینده به کار خواهند آمد.

این داده‌ها به‌طور ویژه برای پیش‌بینی قیمت‌ها در طول یک سال آینده باید فراهم شوند.

**2. نحوه استفاده از این داده‌ها برای پیش‌بینی یک سال آینده**

برای پیش‌بینی یک سال آینده، بعد از آماده‌سازی داده‌ها و آموزش مدل‌ها، می‌توانید از **پیش‌بینی‌های مدل‌ها** برای 365 روز آینده استفاده کنید.

**الف) مدل ARIMAX:**

برای مدل ARIMAX، بعد از آموزش مدل، شما می‌توانید پیش‌بینی‌هایی برای 365 روز آینده (یعنی یک سال) انجام دهید:

python

Copy code

forecast\_arimax = result.forecast(steps=365, exog=external\_features\_next\_year)

**ب) مدل GRU یا LSTM:**

برای مدل‌های GRU و LSTM، پس از آموزش مدل، پیش‌بینی قیمت‌ها برای 365 روز آینده انجام می‌شود:

python

Copy code

forecast\_gru = model\_gru.predict(future\_data)

forecast\_gru = scaler.inverse\_transform(forecast\_gru)

**ج) مقایسه پیش‌بینی‌ها:**

پس از انجام پیش‌بینی‌ها با مدل‌های مختلف، می‌توانید پیش‌بینی‌ها را برای یک سال آینده مقایسه کنید:

python

Copy code

plt.plot(forecast\_arimax, label='ARIMAX')

plt.plot(forecast\_gru, label='GRU')

plt.plot(forecast\_lstm, label='LSTM')

plt.legend()

plt.show()

**نتیجه‌گیری**

برای پیش‌بینی یک سال آینده قیمت سهام شستا، باید داده‌های تاریخی قیمت سهام، ویژگی‌های فنی مانند میانگین متحرک و نوسانات، و همچنین ویژگی‌های اقتصادی (برای مدل ARIMAX) را به مدل‌ها بدهید. سپس با استفاده از این داده‌ها و مدل‌ها می‌توانید پیش‌بینی‌هایی برای آینده انجام دهید و بهترین مدل را انتخاب کنید.

مقادیر ستون close در سال‌های 2020 بسیار بالا (حدود 10,000) هستند، اما در سال‌های 2024 به شدت کاهش یافته‌اند (حدود 1,000). این اختلاف مقادیر باعث ایجاد نوسانات نامتقارن و کوچک شدن بخشی از نمودار می‌شود.

راه‌حل‌ها

1. تنظیم مقیاس نمودار

برای کاهش تأثیر اختلاف شدید مقادیر، می‌توانید از مقیاس لگاریتمی استفاده کنید. این کار باعث می‌شود تغییرات کوچک‌تر نیز به وضوح قابل مشاهده باشند