- توضیح الگوریتم: در این کد از روش شبکه های عصبی چند لایه (multilayer perceptron) استفاده میشود که ساختاری مانند شبکه های عصبی ایجاد میشود که در ابتدا تعدادی داده برای مثال 50 داده به لایه اول داده میشود و این لایه با توجه به نحوه تعریف ما n داده را توسط آموزشی که دیده به لایه بعدی می دهد و به همین ترتیب تا به لایه آخر برسیم که در اینجا خروجی لایه آخر ما یک داده می باشد که برابر مقدار پیش بینی شده، است.

- روند کلی کد: در این کد ابتدا داده ها را از فایل csv خوانده و اطلاعات ستون را در لیست هایی متناظر با هر ستون ذخیره می کنیم و پس از نرمال سازی داده ها یک مدل برای آموزش ماشین ساخته و به پیش بینی می پردازیم.

- مرحله اول: داده های close و volume را از فایل ADANIPORTS.csv خوانده و در لیست های closePrice و volume ذخیره میکنیم و قیمت روز های بعد را (قیمت روز i ام در ایندکس i-1 ام) در لیست nextClose ذخیره می کنیم و در مرحله بعد داده ها را نرمال می کنیم.

```
'''read data from file and normalization'''
data = pd.read_csv('ADANIPORTS.csv')
closePrice, volume = data['Close'].to_numpy(dtype=float), data['Volume'].to_numpy(dtype=float)

closePrice, volume = np.delete(closePrice, -1), np.delete(volume, -1)
nextClose = (data['Close'].to_numpy(dtype=float))
nextClose = np.delete(nextClose, 0)
```

- مرحله دوم: در این مرحله داده های همه بازه های 30 روز متوالی (قیمت بسته شدن و حجم) را در یک لیست از آرایه ها ذخیره می کنیم که در هر آرایه در سطر اول قیمت های بسته شدن و در سطر دوم حجم بازه ها ذخیره شده اند و در لیست nextPrice قیمت روز 31 ام متناسب با هر بازه ذخیره شده است.

```
'''collect close price and volume of last 30 days'''
compact = [[p, v] for p, v in zip(closePrice, volume)]
last30Days = []
nextPrice = []
for i in range(30, len(closePrice)-1):
    last30Days.append(compact[i-30:i])
    nextPrice.append([closePrice[i]])
```

- مرحله سوم: در این مرحله داده ها را به دسته های train و test تقسیم می کنیم به این صورت که 90٪ از داده ها برای آموزش و باقی برای تست قرار میگیرند.

```
'''split datas to train and set lists'''
xTrain, xTest, yTrain, yTest = train_test_split(last30Days, nextPrice, test_size=0.1, shuffle=False)
```

- مرحله چهارم: در این مرحله یک شبکه عصبی می سازیم که ابتدا داده های ورودی را با تابع Flatten مسطح کرده و سپس لایه هایی با 100 و 50 و 10 و 5 و 1 داده خروجی تشکیل می دهیم که لایه آخر یک خروجی دارد که به عنوان نتیجه پیش بینی ما می باشد.

```
'''create multi layer perceptron for training model'''
model = keras.Sequential()
model.add(layers.Flatten(input_shape=(30,2)))
model.add(layers.Dense(100))
model.add(layers.Dense(50))
model.add(layers.Dense(10))
model.add(layers.Dense(5))
model.add(layers.Dense(1))
```

- مرحله پنجم: در این مرحله مدلی که ساختیم را compile کرده و توسط تابع summary خلاصه اطلاعات مدل را بدست می آوریم.

```
'''compile model and give summary of our model'''
model.compile(optimizer='Adam', loss='mse')
model.summary()
```

- مرحله ششم: در این مرحله مدل را آموزش می دهیم و توسط تابع evaluate مدل را ارزیابی می کنیم.

```
'''fit and evaluate our model'''
resss = model.fit(xTrain, yTrain, batch_size=64, epochs=100)
model.evaluate(xTest, yTest)
```

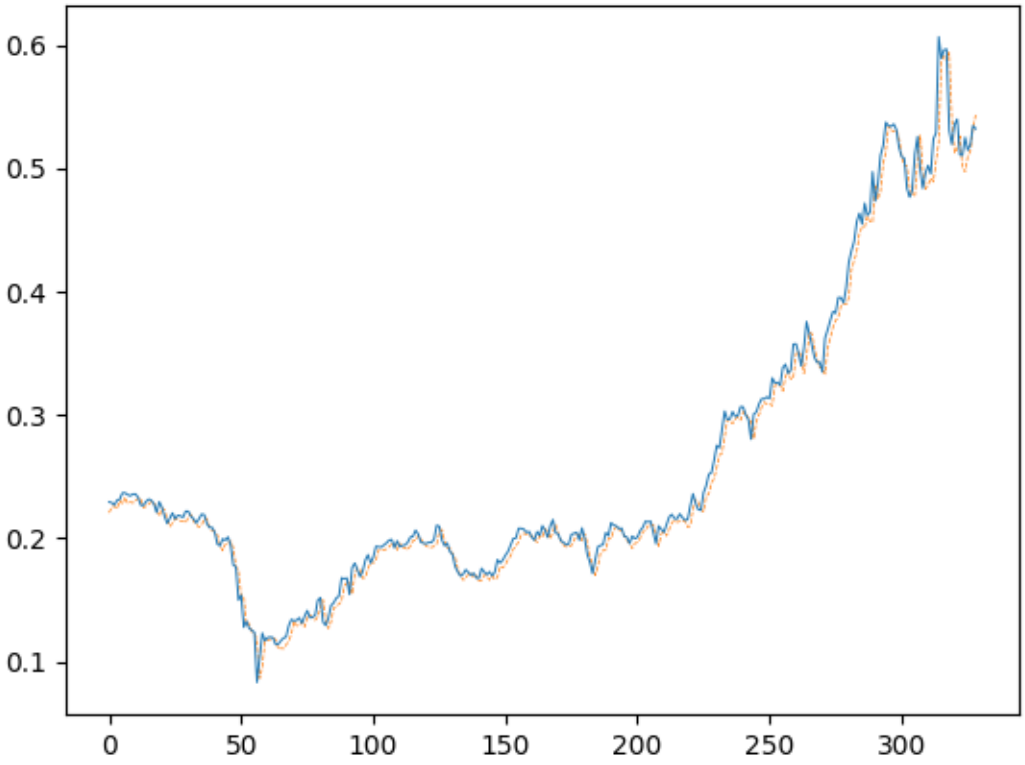
- مرحله هفتم: در این مرحله به مدل داده های تست را میدهیم تا پیش بینی کند و مقادیر اصلی و پیش بینی شده را چاپ می کنیم.

```
'''predict value of test set'''
pred = model.predict(xTest)
print('============================')
print('pred    org')
for i in range(len(pred)):
    print(pred[i], end='\t')
    print(yTest[i])
```

- مرحله آخر: در آخر نمودار داده های پیش بینی شده و داده های اصلی (test set) را رسم می کنیم.

```
'''plot predicted value and orignal value of test set'''
plt.plot(pred, 'C1--', linewidth=0.5)
plt.plot(yTest, 'C0', linewidth=0.7)
plt.show()
```

- نتیجه:



نمودار نارنجی مقدار پیش بینی شده و نمودار آبی داده اصلی می باشد