خلاصه کد با استفاده از CALLBACK

- خلاصه : کد پیش بینی با استفاده از متد callback برای جلوگیری از ادامه آموزش کد وقتی که به حد مد نظر میرسد در این کد هدف ما رسیدن به دقت (accuracy) 82% میباشد.

- پیش نیاز ها: برای استفاده از کتابخوانه هایی که در این کد استفاده شده پس از clone کردن پروژه از اینجا به directory پروژه رفته و command های زیر را به ترتیب در ترمینال وارد کنید.

python -m virenv MyVirEnv
MyVirEnv\Scripts\activate
pip install req.txt

- مرحله اول: در این مرحله ابتدا داده ها را از فایل Khodro.csv خوانده و سپس نرمال سازی می کنیم.

```
def normalize(data):
    tmp = [((item - min(data)) / ((max(data)-min(data)))) for item in data]
    return np.array(tmp)
```
✓ 0.3s

```
data = pd.read_csv('khodro.csv')
closePrice, volume = normalize(data['Close']), normalize(data['Volume'])
```
✓ 3.4s

- مرحله دوم: در این مرحله داده ها را با هم تجمیع کرده و در یک لیست دو بعدی ذخیره می کنیم و یک لیست هم برای پیش بینی روند روز بعد میسازیم.

```
compact = [[p, v] for p, v in zip(closePrice, volume)]
clips = []
nextPred = []
period = 5
for i in range(period, len(closePrice)):
    clips.append(compact[i-period:i])


for clipp in clips[1:]:
    if clipp[-1][0] > clipp[0][0]:
        nextPred.append([1,0,0])
    elif clipp[-1][0] == clipp[0][0]:
        nextPred.append([0,0,1])
    else:
        nextPred.append([0,1,0])
clips.pop(-1)
```

- مرحله سوم: در این مرحله داده ها را به داده test و train تقسیم میکنیم.

```
xTrain, xTest, yTrain, yTest = train_test_split(clips, nextPred, test_size=0.1, shuffle=False)
```

- مرحله چهارم: در این مرحله لایه های شبکه عصبی را میسازیم و مدل را کامپایل می کنیم.

```
model = keras.Sequential()
model.add(layers.Flatten(input_shape=(period,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(3))
model.add(layers.Activation('softmax'))

model.compile(optimizer='Adam', loss=keras.losses.CategoricalCrossentropy(), metrics=keras.metrics.CategoricalAccuracy())
model.summary()
```

- مرحله پنجم: در این مرحله کلاسی برا برای بررسی اینکه آیا به هدف رسیده ایم تا آموزش متوقف شود تعریف کرده ایم که در متد دوم این کلاس با یک شرط if بررسی میکنیم که آیا به مقدار مورد نظر رسیده ایم یا خیر و در صورت دستیابی به هدف آموزش مدل متوقف میشود و در ادامه یک شی از این کلاس میسازیم.

```
class TerminateOnBaseline(Callback):
    def __init__(self, monitor='categorical_accuracy', baseline=0.9):
        super(TerminateOnBaseline, self).__init__()
        self.monitor = monitor
        self.baseline = baseline

    def on_epoch_end(self, epoch, logs=None):
        logs = logs or {}
        acc = logs.get(self.monitor)
        if acc is not None:
            if acc >= self.baseline:
                print('Epoch %d: Reached baseline, terminating training' % (epoch))
                self.model.stop_training = True
```

0.3s

```
callbacks = [TerminateOnBaseline(monitor='categorical_accuracy', baseline=0.82)]
```

- مرحله ششم: در این مرحله مدل را فیت میکنیم.

```
resss = model.fit(xTrain, yTrain, batch_size=128, epochs=100, callbacks=[callbacks])
```

- نتیجه: مشاهده میکنیم که آموزش مدل پس از اینکه Accuracy به مقدار 82% رسیده متوقف میشود.

```
Epoch 1/100
21/21 [==============================] - 0s 4ms/step - loss: 0.4742 - categorical_accuracy: 0.7982
Epoch 2/100
21/21 [==============================] - 0s 4ms/step - loss: 0.4524 - categorical_accuracy: 0.8099
Epoch 3/100
21/21 [==============================] - 0s 4ms/step - loss: 0.4387 - categorical_accuracy: 0.8336
Epoch 2: Reached baseline, terminating training
```