به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین چهارم

نام و نام خانوادگی	فاطمه جلیلی – سالار صفردوست
شماره دانشجویی	۸۱۰۱۹۹۳۹۸ -۸۱۰۱۹۹۴۵۰
تاریخ ارسال گزارش	۱۴۰۲/۱۰/۵

فهرست

پاسخ ۱- پیش بینی سری زمانی

1-1. دانلود داده ها

در این قسمت ابتدا اسامی سهام‌ها با سال ورود قبل از ۲۰۱۰ از لیست SP100 دانلود شده و سپس با استفاده از نام این سهامی، مقادیر مربوط به سهامی آن‌ها از یاهو فایننس با فاصله‌های زمانی یک روزه دانلود شد.

	Symbol	Security	GICS Sector	GICS Sub-Industry	Headquarters Location	Date added	CIK	Founded
0	MMM	3M	Industrials	Industrial Conglomerates	Saint Paul, Minnesota	1957-03-04	66740	1902
2	ABT	Abbott	Health Care	Health Care Equipment	North Chicago, Illinois	1957-03-04	1800	1888
5	ADBE	Adobe Inc.	Information Technology	Application Software	San Jose, California	1997-05-05	796343	1982
7	AES	AES Corporation	Utilities	Independent Power Producers & Energy Traders	Arlington, Virginia	1998-10-02	874761	1981
8	AFL	Aflac	Financials	Life & Health Insurance	Columbus, Georgia	1999-05-28	4977	1955
...	...	...	...	...	...	...	...	...
495	WYNN	Wynn Resorts	Consumer Discretionary	Casinos & Gaming	Paradise, Nevada	2008-11-14	1174922	2002
496	XEL	Xcel Energy	Utilities	Multi-Utilities	Minneapolis, Minnesota	1957-03-04	72903	1909
498	YUM	Yum! Brands	Consumer Discretionary	Restaurants	Louisville, Kentucky	1997-10-06	1041061	1997
500	ZBH	Zimmer Biomet	Health Care	Health Care Equipment	Warsaw, Indiana	2001-08-07	1136869	1927
501	ZION	Zions Bancorporation	Financials	Regional Banks	Salt Lake City, Utah	2001-06-22	109380	1873

290 rows × 8 columns

جدول اسامی SP100 با سال تأسیس قبل ۲۰۱۰

	IFF						PPG	
	Open	High	Low	Close	Adj Close	Volume	Open	H
Date								
2010-01-04	41.509998	42.020000	41.500000	42.009998	30.823030	286000	29.650000	
2010-01-05	41.860001	41.930000	41.439999	41.700001	30.595594	348900	30.200001	
2010-01-06	41.630001	41.970001	41.509998	41.869999	30.720303	375600	30.209999	
2010-01-07	41.650002	41.840000	41.070000	41.549999	30.485525	402000	30.674999	
2010-01-08	41.330002	41.400002	40.980000	41.400002	30.375486	249000	30.860001	
...	...	...	...	...	...	...	...	
2023-12-19	79.199997	80.279999	79.059998	79.989998	79.989998	2187600	149.429993	15
2023-12-20	79.830002	80.089996	78.989998	79.599998	79.599998	1870400	149.429993	14
2023-12-21	80.400002	80.550003	79.029999	80.449997	80.449997	1172100	147.630005	14
2023-12-22	81.110001	81.949997	80.330002	80.699997	80.699997	1174100	148.100006	14
2023-12-26	80.779999	82.169998	80.660004	82.139999	82.139999	416255	148.839996	15

3519 rows × 1740 columns

جدول مقادیر ارزش سهم‌ها با گروه‌بندی روزانه

2-1. کاوش در داده های سری زمانی و آشنایی با تئوری ها و کتابخانه های معروف

ممکن است در داده‌های دانلود شده مقادیر null وجود داشته باشند، این مقادیر null چهار حالت دارند:

۱- در ابتدای ستون جدول باشند.

۲- در اواسط ستون جدول باشند.

۳- در انتهای ستون جدول باشند.

۴- کل یک ستون جدول null باشد.

برای رفع مشکل نبود مقادیر در جدول می‌توان از سه تابع interpolate، fillna و dropna استفاده کرد.

به طور کلی داده‌های ناموجود از ابتدای لیست را می‌توان یا با مقدار صفر پر کرد، یا از اولین مقدار درست کپی گرفت و در این خانه‌ها قرار داد. این کار با استفاده از fillna استفاده می‌شود.

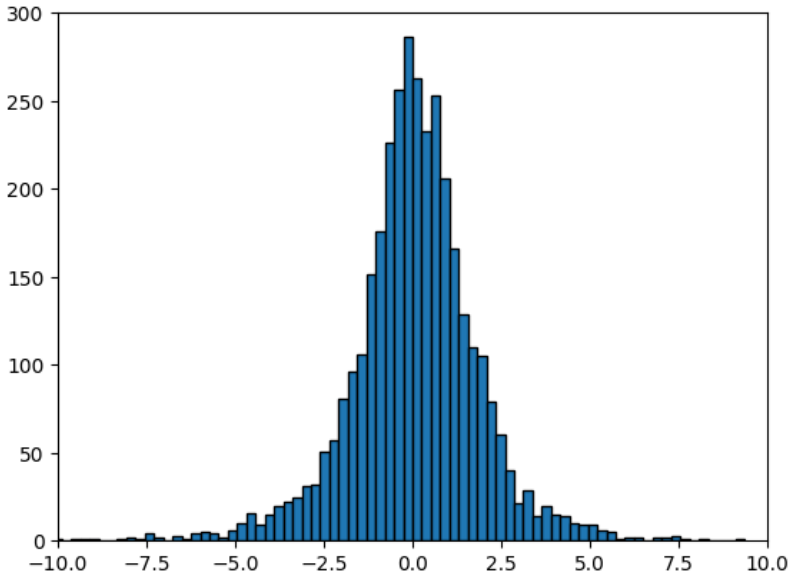
برای داده‌های اواسط جدول نیز بهترین روش برای پر کردن، استفاده از interpolation می‌باشد، به این شکل که مقادیر ناموجود با استفاده از مقادیر کناری به صورت خطی، سهموی و ... تخمین زده می‌شوند.

جدول دریافت شده دارای تعدادی از ستون‌های null بود، برای رفع کلی مشکل nullها ابتدا تابع interpolate مقادیر نامشخص اواسط(و انتها) را به صورت خطی تخمین می‌زند، سپس مقادیر null باقی‌مانده در ابتدای ستون‌ها با fillna مقداردهی می‌شوند و در انتها ستون‌های null به طور کامل از جدول با استفاده از dropna حذف می‌شوند.

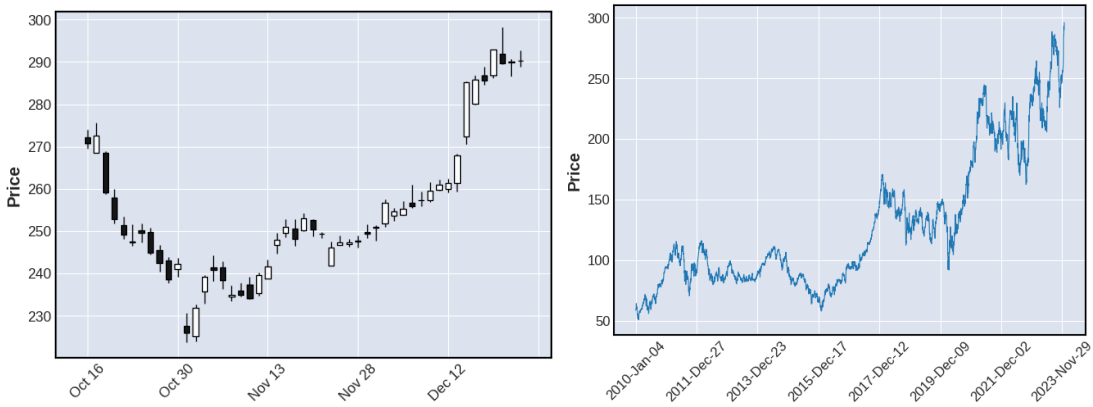
```
[11] null_values = raw_prices.isnull()
     print(null_values.sum(axis=1).sum(axis=0))

     39708

[12] prices = raw_prices.interpolate(method='linear', axis=1)
     prices = prices.fillna(method='bfill')
     prices = prices.dropna(axis=1)

[13] null_values = prices.isnull()
     print(null_values.sum().sum())

     0
```

تعداد مقادیر null قبل و بعد از اجرای کد

در مرحله‌ی بعد یکی از سهم‌ها به صورت شانسی انتخاب شده و هیستوگرام **close price return**، نمودار **close** خطی آن برای تمامی زمان‌ها و نمودار **candle** آن برای ۵۰ روز آخر را رسم می‌کنیم.



هیستوگرام close price return یک سهم

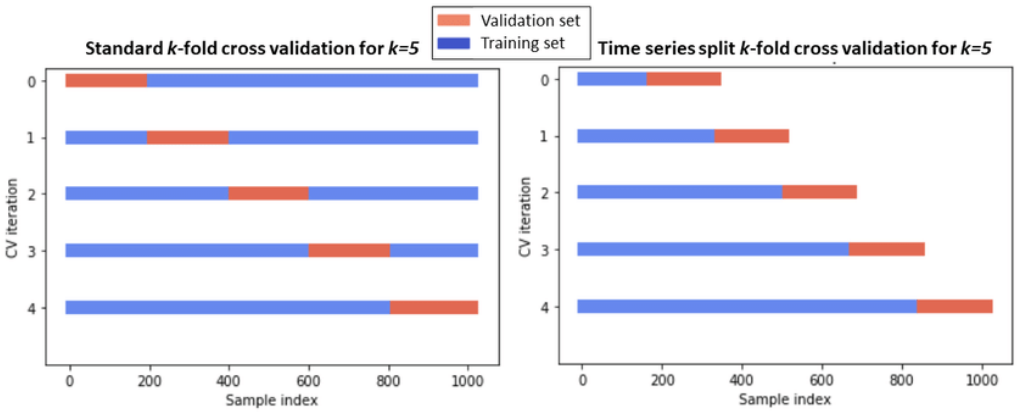


نمودارهای خطی و شمعی یک سهم

با توجه به شکل هیستوگرام و همچنین با توجه به فرمول close price return که مقدار نسبی تفاضل قیمت در دو روز متوالی می‌باشد، با دانستن قیمت یک روز قبل، بهترین پیش‌بینی برای روز بعد همان قیمت امروز می‌باشد. این امر به random walk theory نیز می‌تواند مربوط باشد که ادعا می‌کند بهترین پیش‌بینی برای قیمت آینده قیمت در لحظه است. (البته این گفته توسط هیستوگرام اثبات نمی‌شود، چرا که در هیستوگرام تنها دانسته‌ی ما مقدار روز قبل است و اطلاعات از روزهای قبل‌تر در آن گنجانده نشده است.)

3-1. TimeSeriesSplit

در سری‌های زمانی به علت ارتباط داده‌ها به یکدیگر و عدم استقلال آن‌ها نمی‌توان مشابه آنچه در cross validation سایر مدل‌ها انجام می‌دادیم(انتخاب دسته‌ای تصادفی به عنوان ترین و دسته‌ای دیگر به عنوان تست) عمل کنیم. همچنین جداسازی ترین و تست ما در این گونه داده‌ها نباید به گونه‌ای باشد که داده‌ی تست قبل از داده‌های ترین باشد. برای رفع این مشکلات time series split پیشنهاد می‌شود. روش کار این split کردن در شکل زیر نمایش داده شده است. انواع مختلفی برای time series split وجود دارد که دو نوع آن walk-forward و nested می‌باشد.



تفاوت cross validation در داده‌های زمانی و داده‌های معمولی مستقل

* با توجه به عدم حضور بعضی سهامی نوشته شده در گزارش(به علت عدم حضور آن‌ها تا قبل ۲۰۱۰)، سهم‌های دیگری جایگزین آن‌ها شدند.

Date	AAPL	AMZN	MSFT	AMGN	VFC
2010-01-04	7.643214	6.695000	30.950001	57.720001	17.274012
2010-01-05	7.656429	6.734500	30.959999	57.220001	17.603579
2010-01-06	7.534643	6.612500	30.770000	56.790001	17.532957
2010-01-07	7.520714	6.500000	30.450001	56.270000	17.871941
2010-01-08	7.570714	6.676000	30.660000	56.770000	17.777779
...	...	...	...	...	...
2023-12-19	196.940002	153.789993	373.260010	278.440002	18.740000
2023-12-20	194.830002	152.119995	370.619995	275.179993	17.940001
2023-12-21	194.679993	153.839996	373.540009	279.329987	19.209999
2023-12-22	193.600006	153.419998	374.579987	284.160004	18.590000
2023-12-26	193.085007	153.315002	374.089996	282.799988	18.688999

3519 rows × 5 columns

جدول مربوط به داده‌های close ۵ سهم انتخاب شده

4-1. آماده سازی ورودی و خروجی مدل

در این قسمت پنجره‌هایی به طول ۲۰ به همراه ۱ مقدار آینده از داده‌های زمانی جدا شده و زیر هم قرار داده می‌شوند تا x و y ما را تشکیل بدهند.

* عملیات scaling در این مرحله انجام نمی‌شود و در قسمت training انجام می‌شود تا فقط با دیتای ترین اسکیل را انجام بدهیم.

```
window_size = 20

x = []
y = []
for i, stock_name in enumerate(stock_names):
  series = data_np[:, i]
  x_stock = []
  y_stock = []
  for j in range(window_size, len(series)):
    x_stock.append(series[j-window_size:j])
    y_stock.append(series[j])
  x_stock = np.array(x_stock)
  y_stock = np.array(y_stock)
  x.append(x_stock)
  y.append(y_stock)
```

```
print(x[0].shape)
print(y[0].shape)
```

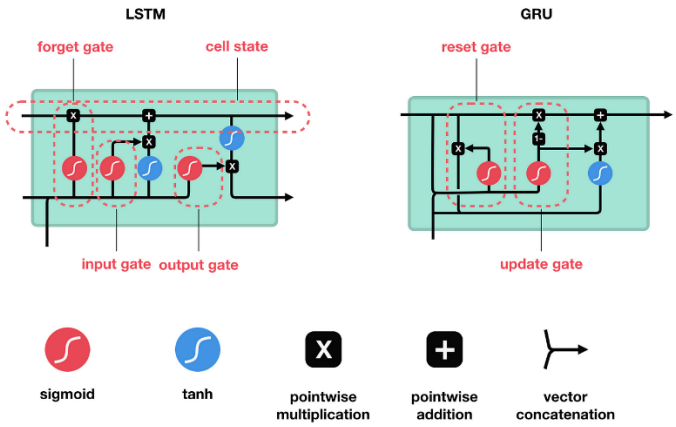
```
(3499, 20)
(3499,)
```

روند ایجاد x و y

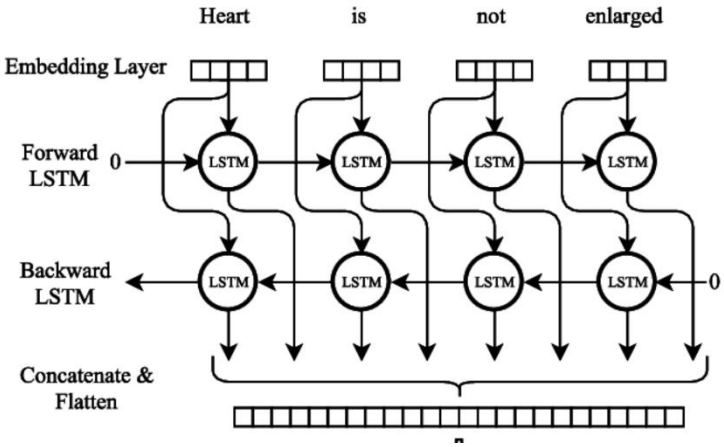
5-1. مدل های شبکه عصبی حافظه دار

مدل LSTM: این مدل نوعی RNN است که در داخل خود توانایی ایجاد استیت برای هر واحد LSTM را دارد. داخل هر واحد، سه سری گیت وجود دارد، یک گیت برای فراموش کردن مقدار استیت فعلی، یک گیت برای اضافه کردن اطلاعات جدید برای استیت واحد از روی ورودی و خروجی واحد قبلی و گیت سوم برای ساخت خروجی واحد بر اساس ورودی و استیت فعلی می‌باشد.(Input Gate، Output Gate، Update Gate) این مدل به علت وجود گیت‌ها، توانایی ذخیره‌ی همزمان اطلاعات از گذشته‌ی دور و نزدیک را در خود دارد و به همین علت عملکرد بهتری از RNN معمولی دارد.

مدل GRU: این مدل شبیه به LSTM می‌باشد، ولی به جای سه گیت، از دو گیت بهره می‌برد. این مدل پیچیدگی کمتری نسبت به LSTM دارد.



مدل BiLSTM: این مدل متشکل از دو LSTM به صورت همزمان یکی در جهت رفت و دیگری در جهت برگشت می‌باشد. این خاصیت باعث می‌شود که مدل توانایی یادگیری بهتر سیگنال‌های پیچیده‌تر را داشته باشد و بتواند همزمان در دو مسیر آموزش ببیند.



مدل Bi-LSTM یا Bidirectional Long Short-Term Memory یک نوع مدل شبکه عصبی بازگشتی (RNN) است که از دو لایه LSTM به صورت همزمان استفاده می‌کند تا اطلاعات برگشتی را هم در جلو و هم در عقب از نقطه مورد بررسی در دنباله داده‌ها در نظر بگیرد.

* به طور خلاصه در پیچیدگی:

$$Bi - LSTM > LSTM > GRU$$

معیار MAE: این معیار همانند زیر میانگین قدرمطلق تفاضل داده‌های واقعی با تخمینی می‌باشد.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

معیار MSE: این معیار میانگین توان دوی تفاضل داده‌های واقعی با تخمینی می‌باشد.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

معیار MAPE: این معیار همان MAE می‌باشد، با این تفاوت که میزان تفاضل را به مقدار تخمینی تقسیم می‌کند. مزیت اینکار این است که نسبت به کوچکی و بزرگی داده‌ها مستقل می‌شویم.

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

برای آموزش و بررسی مدل‌ها، برای هر سهم، هر مدل به مقدار ۵ بار(تعداد داده‌های ترین و تست در tss) آموزش داده شده، سپس میانگین معیارها از این ۵ بار آموزش گرفته شده و در خانه‌ای از ماتریس معیارها ریخته می‌شود.

در نهایت ما برای هر مدل یک ماتریس ۵ در ۳ خواهیم داشت که سطرها برای هر سهم بوده و ستون‌ها به ترتیب مربوط به به مقادیر mse، mape و mae خواهند بود(این مقادیر روی داده‌های تست به دست آمده‌اند). برای انجام این محاسبات از تابع Model_Metrics بهره برده شد. (به علت سنگینی حجم محاسبات، تعداد epoch به ۳۲ کاسته شد، همین موضوع سبب ضعف در نتایج مدل‌های پیچیده‌تر شد.)

```
LSTM_Model = Sequential([
    LSTM(units=200, return_sequences=True, input_shape=(x[0].shape[1], 1)),
    Dropout(0.2),
    LSTM(units=200, return_sequences=True),
    Dropout(0.2),
    LSTM(units=200, return_sequences=True),
    Dropout(0.2),
    LSTM(units=200, return_sequences=False),
    Dropout(0.2),
    Dense(units=1)
])

GRU_Model = Sequential([
    GRU(units=200, return_sequences=True, input_shape=(x[0].shape[1], 1)),
    Dropout(0.2),
    GRU(units=200, return_sequences=True),
    Dropout(0.2),
    GRU(units=200, return_sequences=True),
    Dropout(0.2),
    GRU(units=200, return_sequences=False),
    Dropout(0.2),
    Dense(units=1)
])
```

```
BiLSTM_Model = Sequential([
    Bidirectional(LSTM(units=200, return_sequences=True), input_shape=(x[0].shape[1], 1)),
    Dropout(0.2),
    Bidirectional(LSTM(units=200, return_sequences=True)),
    Dropout(0.2),
    Bidirectional(LSTM(units=200, return_sequences=True)),
    Dropout(0.2),
    Bidirectional(LSTM(units=200, return_sequences=False)),
    Dropout(0.2),
    Dense(units=1)
])

MLP_Model = Sequential([
    Dense(units=200, activation='relu', input_shape= (x[0].shape[1], ) ),
    Dropout(0.2),
    Dense(units=200, activation='relu'),
    Dropout(0.2),
    Dense(units=200, activation='relu'),
    Dropout(0.2),
    Dense(units=1)
])
```

```
CNN_Model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(x[0].shape[1], 1)),
    Dropout(0.2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    Dropout(0.2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    Dropout(0.2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    Dropout(0.2),
    MaxPooling1D(pool_size=2),
    Flatten(),
    Dense(units=1)
])

ConvLSTM_Model = Sequential([
    ConvLSTM1D(filters=64, kernel_size=1, activation='relu', return_sequences=True, input_shape=(x[0].shape[1], 1, 1)),
    Dropout(0.2),
    ConvLSTM1D(filters=64, kernel_size=1, activation='relu', return_sequences=False),
    Dropout(0.2),
    Dense(units=1)
])
```

مدل‌های مورد استفاده

```
array([[2.25378826e-02, 7.38811951e+00, 1.01364350e-01],
       [3.20122004e-02, 9.32906036e+00, 1.28597164e-01],
       [3.23045433e-02, 7.98682251e+00, 1.19043279e-01],
       [3.49132538e-02, 6.83105316e+00, 1.11347616e-01],
       [8.87485221e-03, 1.31275787e+01, 5.87953687e-02]])
```

LSTM

```
array([[3.46602425e-03, 3.40621681e+00, 4.28821385e-02],
       [7.14976415e-03, 4.90064507e+00, 6.67099237e-02],
       [2.31480189e-03, 3.10611248e+00, 3.61183971e-02],
       [1.41900629e-02, 5.01634674e+00, 7.75336027e-02],
       [2.76430827e-03, 7.07564621e+00, 3.59157443e-02]])
```

GRU

```
array([[ 0.01263668,  5.84233627,  0.07597427],
       [ 0.03553617,  8.99012909,  0.1359401 ],
       [ 0.03686234,  9.65815735,  0.14263536],
       [ 0.07469472,  6.95285263,  0.13619218],
       [ 0.01078445, 10.51775055,  0.06128413]])
```

Bi-LSTM

```
array([[ 0.11909199, 24.44932404,  0.30753314],
       [ 0.13590233, 26.05633545,  0.34074144],
       [ 0.11936283, 23.87771759,  0.31639972],
       [ 0.12900407, 25.34142456,  0.32783222],
       [ 0.05252857, 39.85463257,  0.20510273]])
```

CNN

```
array([[ 0.32200592, 41.89423523,  0.51961374],
       [ 0.32506685, 39.82055054,  0.52437072],
       [ 0.35614903, 41.91011963,  0.55350142],
       [ 0.3332588 , 39.36318359,  0.51815972],
       [ 0.13076172, 42.93783875,  0.32189059]])
```

MLP

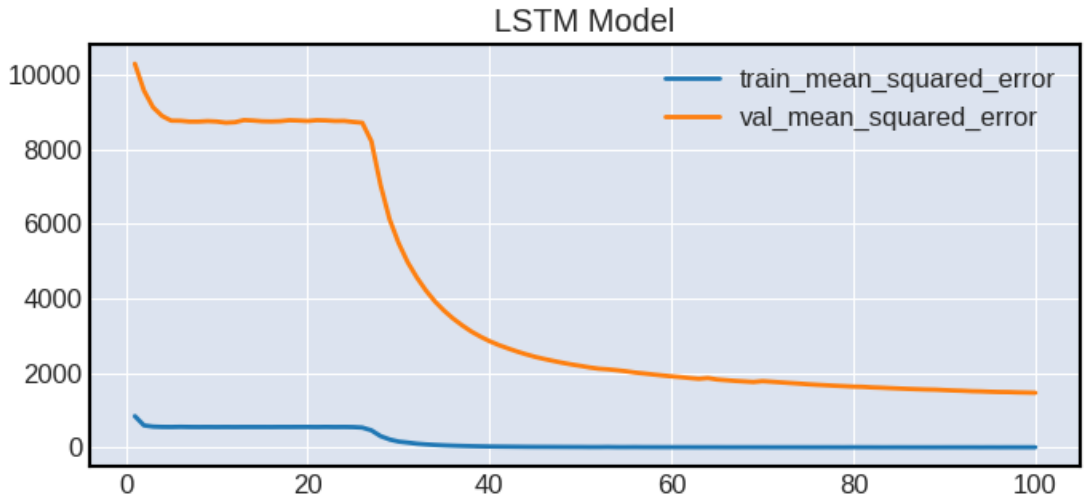
```
array([[ 0.68987975, 14.51110535,  0.28327763],
       [ 2.01880322, 33.93088989,  0.66072278],
       [ 0.79022908, 21.7687149 ,  0.38028853],
       [ 1.5958086 , 16.11213379,  0.38096752],
       [ 0.07486662, 17.22041626,  0.11479743]])
```

Conv-LSTM

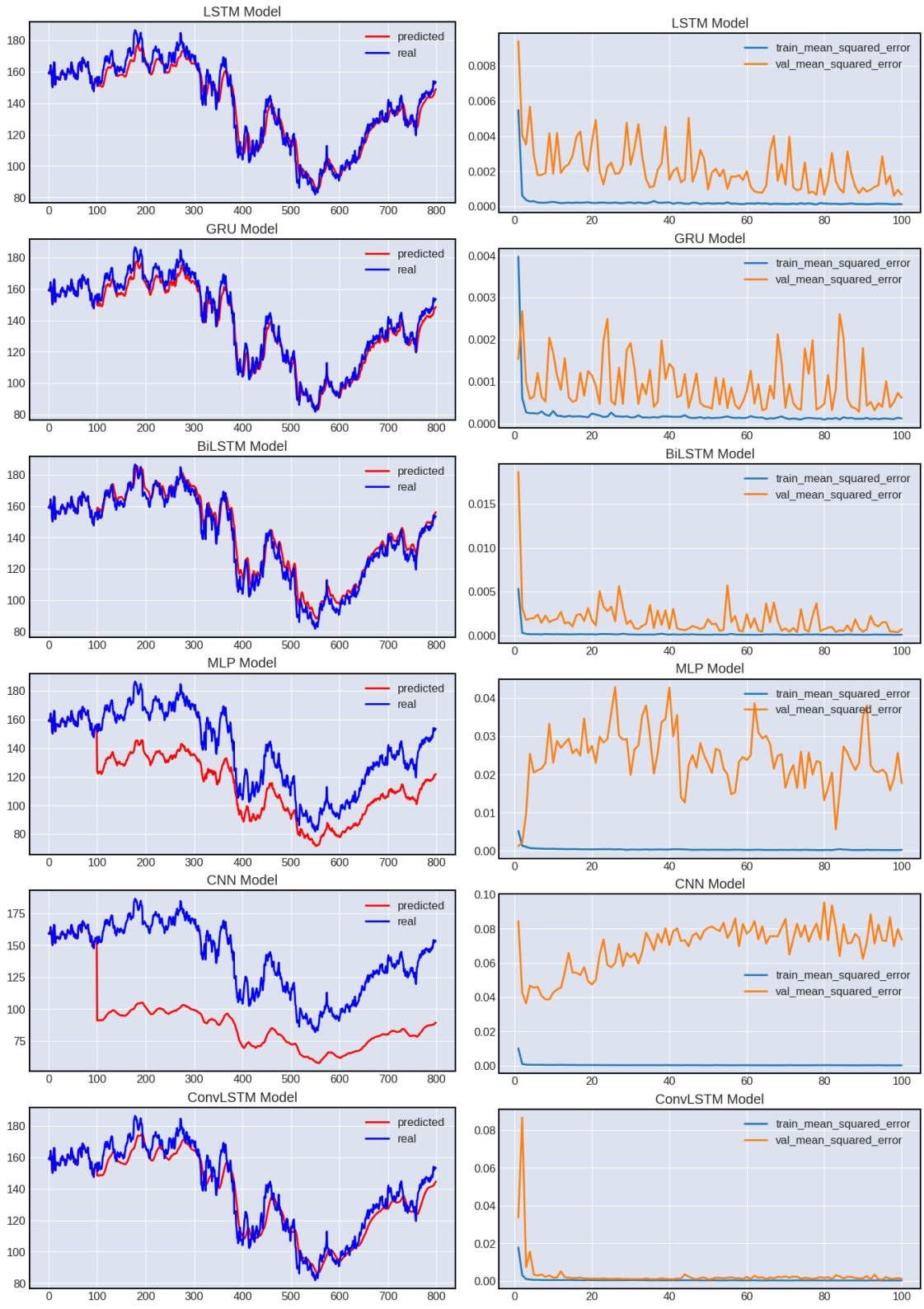
دیده می‌شود بهترین نتایج برای GRU، Bi-LSTM و LSTM می‌باشد و بدترین نتایج برای MLP و CNN. همچنین می‌توان گفت تعداد ایپاک کم به ضرر مدل‌های پیچیده‌تر Conv-LSTM و Bi-LSTM شده است. مقادیر به دست آمده برای ۳ مدل ابتدایی قابل قبول می‌باشد.

در ادامه برای پیدا کردن شهود از نحوه‌ی عملکرد کلی مدل‌ها، یکبار برای یک سهم خاص، تمامی مدل‌ها روی ۰.۸ داده‌ی زمانی اولیه ترین شدند و خروجی آن‌ها به ازای ۰.۲ داده‌ی زمانی باقی‌مانده پیش‌بینی شد. مقادیر واقعی و اصلی برای هر مدل و همچنین مقادیر loss آن‌ها با معیار mse رسم شده است.(صفحه‌ی بعد)

* در قسمت آموزش مدل‌ها اگر داده‌ها اسکیل نشوند، به همگرایی خوبی در پاسخ نمی‌رسیم و تابع هزینه برای دیتاست evaluation بسیار بالا خواهد بود. در زیر روند آموزش مدل LSTM روی داده‌های اسکیل نشده نمایش داده شده است.



نمودار تابع هزینه برای داده‌های اسکیل نشده



با استفاده از تحلیل نتایج بالا دیده می‌شود که GRU، Bi-LSTM، Conv-LSTM و LSTM بهترین نتایج را داشته ولی دو مدل CNN و MLP پیش‌بینی‌های بایاس دار ارائه کرده‌اند.

به صورت عادی انتظار ما این بود که Conv-LSTM و Bi-LSTM بهترین نتایج را بدهند، ولی می‌توان برای رخ ندادن این امر(و بهتر بودن GRU) این توجیه را آورد که این دو شبکه پیچیده‌تر هستند و نیاز به ایپاک‌های بیشتری برای آموزش بهتر می‌باشند.(به طور مثال نمودار پیش‌بینی Conv-LSTM حالت پیوسته‌تری دارد که نشان از عدم آموزش دقیق می‌باشد.)

6-1. Naïve Forecast

در این قسمت سری زمانی سهم اول برای تعیین مقادیر متریک‌ها در روش Naïve Forecast در نظر گرفته شد.

اگر بخواهیم روش Naïve Forecast ساده را پیاده‌سازی کنیم، بهترین پیش‌بینی در برای یک لحظه برابر مقدار سری در لحظه‌ی قبل خواهد بود. پیاده‌سازی این عمل در پایتون معادل یک شیفت دادن دیتا به سمت راست است تا مقادیر پیش بینی شده را داشته باشیم. (این کار برای کل سری زمانی صورت گرفت.)

```
[37] y = data_np[:, 0]
     y_pred = np.roll(y, 1)

[38] y = np.delete(y, 0)
     y_pred = np.delete(y_pred, 0)

[39] scaler_y = MinMaxScaler(feature_range=(0,1))
     scaler_y.fit_transform(y.reshape(-1, 1))

     y_scaled = scaler_y.transform(y.reshape(-1, 1))
     y_pred_scaled = scaler_y.transform(y_pred.reshape(-1, 1))

[40] mse = MeanSquaredError()(y_scaled, y_pred_scaled)
     mape = MeanAbsolutePercentageError()(y_scaled, y_pred_scaled)
     mae = MeanAbsoluteError()(y_scaled, y_pred_scaled)

[46] print(float(mse),', ',float(mape),', ',float(mae))

     5.933563443250023e-05 ,  403.8964538574219 ,  0.004113502334803343
```

پیاده‌سازی Naïve Forecast و به دست آوردن متریک‌ها

در متریک‌های خروجی به دست آمده دیده می‌شود که معیار mse پیشرفت زیادی داشته، در حالی که معیار دوم بسیار بدتر شده و معیار سوم نیز از مقادیر به دست آمده از آموزش بدتر است.

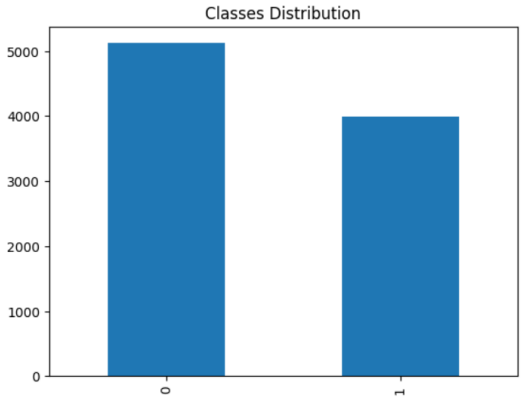
می‌توان این نتیجه را گرفت که هر چند دو معیار مقادیر خوبی را نشان می‌دهند، اما معیار mape نشان می‌دهد که خطای نسبی ما با این روش پیش بینی بسیار زیاد است، چرا که در اکثر موارد مقدار پیش‌بینی با مقدار قبل تفاوت اندکی دارد.

در نهایت استفاده از فرض random walk فرض بدی نیز نمی‌باشد.

پاسخ ۲ – پیش بینی افکار خودکشی در رسانه های اجتماعی

1-2. پیش پردازش داده ها

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9119 entries, 0 to 9118
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   tweet      9119 non-null   object
 1   intention  9119 non-null   int64
dtypes: int64(1), object(1)
memory usage: 142.6+ KB
```



اطلاعات کلی دیتاست و نحوه توزیع دیتا در دو کلاس

پس از لود کردن دیتا توسط کتابخانه پاندا و یافتن اطلاعات لازم مطابق اشکال فوق برای داشتن دید کلی نسبت به دیتاست پیش پردازش را مطابق زیر انجام می دهیم:

- مطابق مقاله کلماتی که دوبار تکرار شده اند را حذف می کنیم
- سپس با استفاده از RegexpTokenizer(r'\w+') متن ها را tokenize می کنیم ، با استفاده از r'\w+' تنها حروف ، اعداد و _ ها باقی می مانند و بقیه کارکتر ها از جمله colons مطابق گفته مقاله حذف می شوند . برای مثال Hello, world! This is an example. به ['Hello', 'world', 'This', 'is', 'an', 'example'] تبدیل می شود.
- در ادامه با کمک کتابخانه NLTK ، token هایی مانند "a" "an" "the" "is" "and" "in" "to" "of" یا همان stopwords را حذف می کنیم.
- برای حذف unnecessary words که در مقاله قید شده ، کلمات غیر انگلیسی و همچنین کلمات با کم تر از ۳ حرف را حذف می کنیم و جایگذاری ' ' : ',' - ' ' : " ' " را هم انجام می دهیم و token ها را به هم مجدد می چسبانیم.
- در آخر هم به جای هر کلمه ریشه آن را قرار می دهیم (Lemmatization).

	tweet	intention	notDuplicate_tweet	lower_case	tokenized_tweet	stop_words	short_word	without_unnecessary	english_tweet	clean_tweet
0	my life is meaningless i just want to end my l...	1	my life is meaningless i just want to end so b...	my life is meaningless i just want to end so b...	[my, life, is, meaningless, i, just, want, to,...	['life', 'meaningless', 'want', 'end', 'badly'...	[life, meaningless, want, end, badly, complete...	life meaningless want end badly completely emp...	life meaningless want end badly completely emp...	life meaningless want end badly completely emp...
1	muttering i wanna die to myself daily for a fe...	1	muttering i wanna die to myself daily for a fe...	muttering i wanna die to myself daily for a fe...	[muttering, i, wanna, die, to, myself, daily, ...	['muttering', 'wanna', 'die', 'daily', 'months...	[muttering, wanna, die, daily, months, feel, w...	muttering wanna die daily months feel worthles...	muttering die daily feel worthless cant live h...	muttering die daily feel worthless cant live h...
2	work slave i really feel like my only purpose ...	1	work slave i really feel like my only purpose ...	work slave i really feel like my only purpose ...	[work, slave, i, really, feel, like, my, only,...	['work', 'slave', 'really', 'feel', 'like', 'p...	[work, slave, really, feel, like, purpose, lif...	work slave really feel like purpose life make ...	work slave really feel like purpose life make ...	work slave really feel like purpose life make ...
3	i did something on the 2 of october i overdose...	1	i did something on the 2 of october overdosed ...	i did something on the 2 of october overdosed ...	[i, did, something, on, the, 2, of, october, o...	['something', '2', 'october', 'overdosed', 'fe...	[something, october, overdosed, felt, alone, h...	something october overdosed felt alone horribl...	something felt alone horrible hospital two day...	something felt alone horrible hospital two day...
4	i feel like no one cares i just want to die ma...	1	i feel like no one cares just want to die mayb...	i feel like no one cares just want to die mayb...	[i, feel, like, no, one, cares, just, want, to...	['feel', 'like', 'one', 'cares', 'want', 'die'...	[feel, like, one, cares, want, die, maybe, les...	feel like one cares want die maybe less lonely	feel like one want die maybe less lonely	feel like one want die maybe le lonely

نتیجه نهایی پیش پردازش به همراه اطلاعات ذخیره شده در دیکشنری در مراحل مختلف پیش پردازش

در ادامه مثال داده شده در صورت گزارش را برای نمونه پیش پردازش می کنیم و نتیجه زیر حاصل می شود:

```
life meaningless want end badly completely empty dont create meaning pain
long hold back urge run car head first next person coming opposite way
stop feeling jealous tragic like gomer pile swift able bring
```

! نبود برخی کلمات تکراری به دلیل حذف آن ها مطابق گفته مقاله است .

2-2. ساخت ماتریس جاسازی

دلیل استفاده و ویژگی های ماتریس جداسازی :

فرض کنیم دایره لغات ما شامل 5000 کلمه باشد و بخواهیم جمله ای شامل ۱۰۰ کلمه را پردازش کنیم، برای بیان عددی این جمله با استفاده از one-hot encoding باید از ۱۰۰ vector با طول های ۵۰۰۰ که ۴۹۹۹ تا از آرایه های آن ها صفر هستند استفاده کنیم. این حجم از ماتریس در train داده های بزرگ ممکن نخواهد بود و همچنین فاصله ی بین همه کلمات در این نوع کد گذاری برابر خواهد بود در صورتی که ترجیح می دهیم برای مثال فاصله بین بردار مربوط به کلمه سگ و گربه کم تر از فاصله بین بردار مربوط به کلمات ماشین و گربه باشد .لذا بایستی به روابط بین کلمات هم توجه کنیم و در vector به جای اعداد ۰ و ۱ از اعداد بین ۰ و ۱ در آرایه ها که نشانگر مربوط بودن کلمه مذکور به گروه کلمات است استفاده کنیم . برای مثال اگر آرایه دوم یک بردار مقدار ۰.۸ داشته باشد یعنی کلمه مونث است و اگر مقدار ۰.۲ داشته باشد یعنی کلمه مذکر است (این صرفا یک مثال است و ویژگی هایی که بر اساس آن ها آرایه های هر بردار مقدار پیدا می کنند در طول آموزش مدل که مثلا در اینجا از مدل word2vec استفاده می کنیم بدست می آیند)

برای آماده سازی ورودی مربوط به لایه های بعدی در اینجا پس از tokenize کردن جمله ها و تخصیص word_index به آن ها ضمن هم طول کردن همه ی بردار های اختصاص داده شده به ماکسیمم طول کلمه موجود چک می کنیم اگر کلمه در word2vec وجود داشته باشد ، مطابق این مدل بردار مربوط به کلمه را پر می کنیم در غیر اینصورت برداری رندوم به سایز ماکسیمم طول کلمه موجود به کلمه مذکور اختصاص می دهیم.

به این ترتیب ماتریس جاسازی به ابعاد ماکسیمم طول کلمه موجود در ۳۰۰ (تعداد ویژگی ها) مطابق مدل word2vec می سازیم.

3-2. آموزش مدل های یادگیری عمیق

مطابق مقاله قرار می دهیم

dropout=0.5 , batch_size = 64, epochs = 20, fully connected layer = softmax,
optimizer = Adam, loss = binary_crossentropy

و برای CNN+2-layer LSTM :

filters = 64, kernel = 3, padding = same , activation function = ReLU, max pooling = 2

و لایه های مدل ها را اضافه می کنیم:

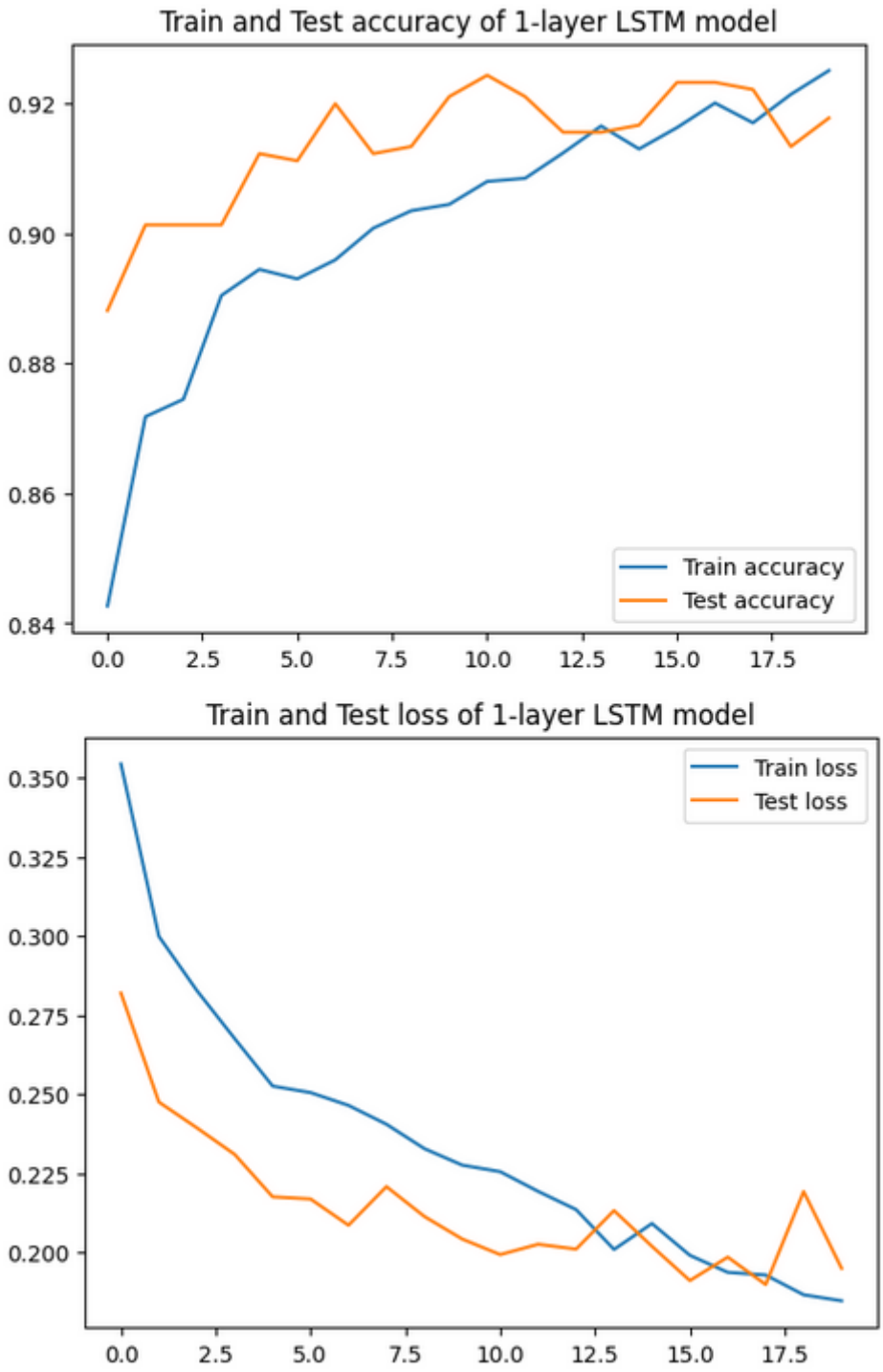
- **1-Layer LSTM**

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 601, 300)          2770500

 dropout_13 (Dropout)        (None, 601, 300)          0

 lstm_19 (LSTM)              (None, 300)               721200

 dense_13 (Dense)            (None, 2)                 602

=================================================================
Total params: 3492302 (13.32 MB)
Trainable params: 721802 (2.75 MB)
Non-trainable params: 2770500 (10.57 MB)
```

خلاصه مدل 1-layer LSTM

```
129/129 [==============================] - 11s 83ms/step - loss: 0.1868 - acc: 0.9214 - val_loss: 0.2193 - val_acc: 0.9154
Epoch 20/20
129/129 [==============================] - 11s 83ms/step - loss: 0.1847 - acc: 0.9251 - val_loss: 0.1950 - val_acc: 0.9178
```

نتیجه epoch آخر مدل 1-layer LSTM



شکل2-5. نمودار های loss و accuracy مدل 1-layer LSTM

	precision	recall	f1-score
Suicide_Post	0.91	0.95	0.93
Not_Suicide_Post	0.93	0.88	0.90
micro avg	0.92	0.92	0.92
macro avg	0.92	0.91	0.92
weighted avg	0.92	0.92	0.92
samples avg	0.92	0.92	0.92

معیار های precision, recall, f1-score برای مدل 1-layer LSTM

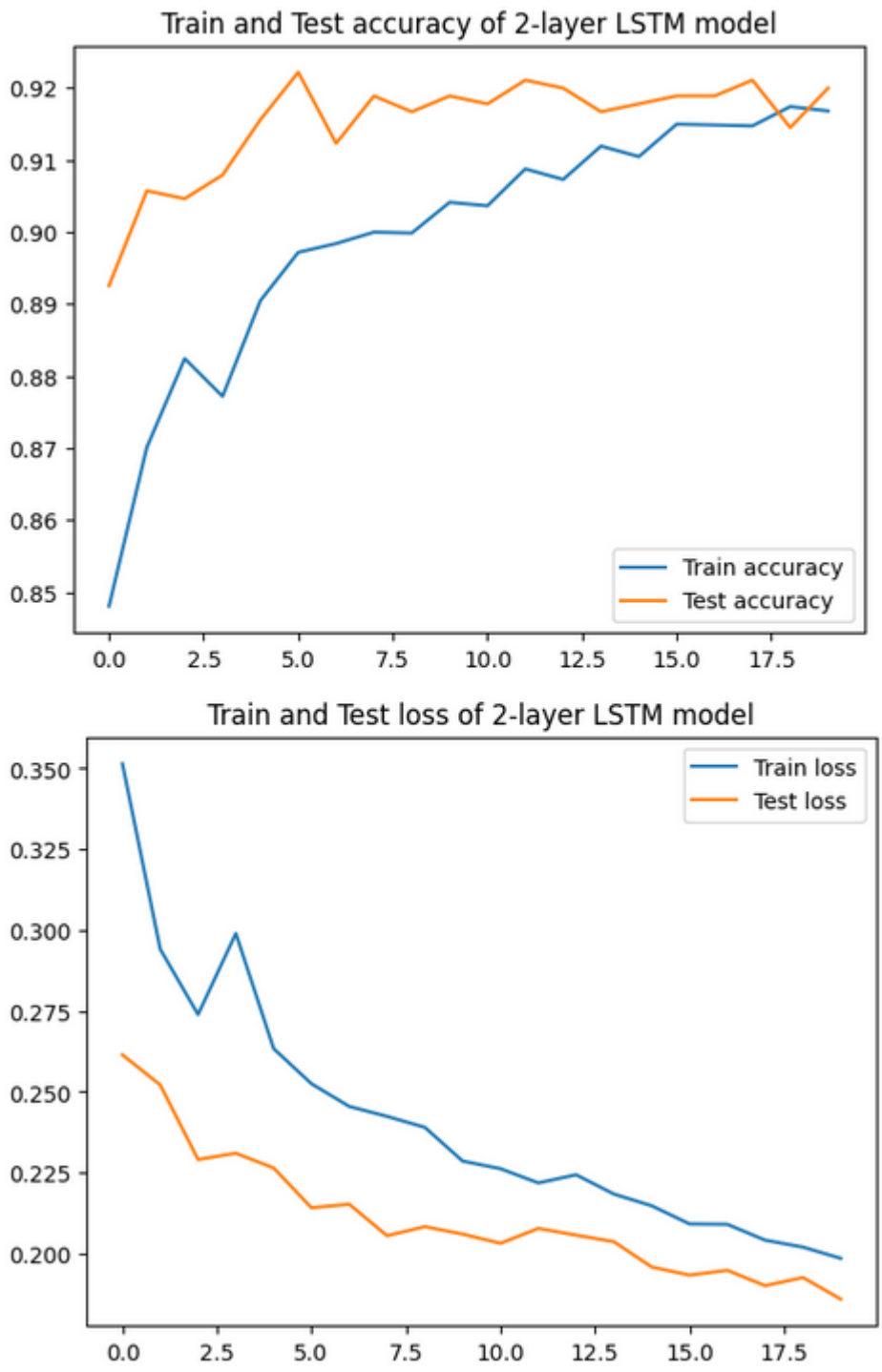
- **2-Layer LSTM**

```
Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 601, 300)          2770500

 dropout_24 (Dropout)        (None, 601, 300)          0

 lstm_45 (LSTM)              (None, 601, 300)          721200

 lstm_46 (LSTM)              (None, 300)               721200

 dense_24 (Dense)            (None, 2)                 602

=================================================================
Total params: 4213502 (16.07 MB)
Trainable params: 1443002 (5.50 MB)
Non-trainable params: 2770500 (10.57 MB)
_________________________________________________________________
```

خلاصه مدل 2-layer LSTM

```
Epoch 20/20
129/129 [==============================] - 21s 165ms/step - loss: 0.1986 - acc: 0.9168 - val_loss: 0.1860 - val_acc: 0.9200
```

نتیجه epoch آخر مدل 2-layer LSTM



نمودار های loss و accuracy مدل 2-layer LSTM

	precision	recall	f1-score
Suicide_Post	0.91	0.96	0.93
Not_Suicide_Post	0.94	0.87	0.90
micro avg	0.92	0.92	0.92
macro avg	0.92	0.91	0.92
weighted avg	0.92	0.92	0.92
samples avg	0.92	0.92	0.92

معیار های precision, recall, f1-score برای مدل 2-layer LSTM

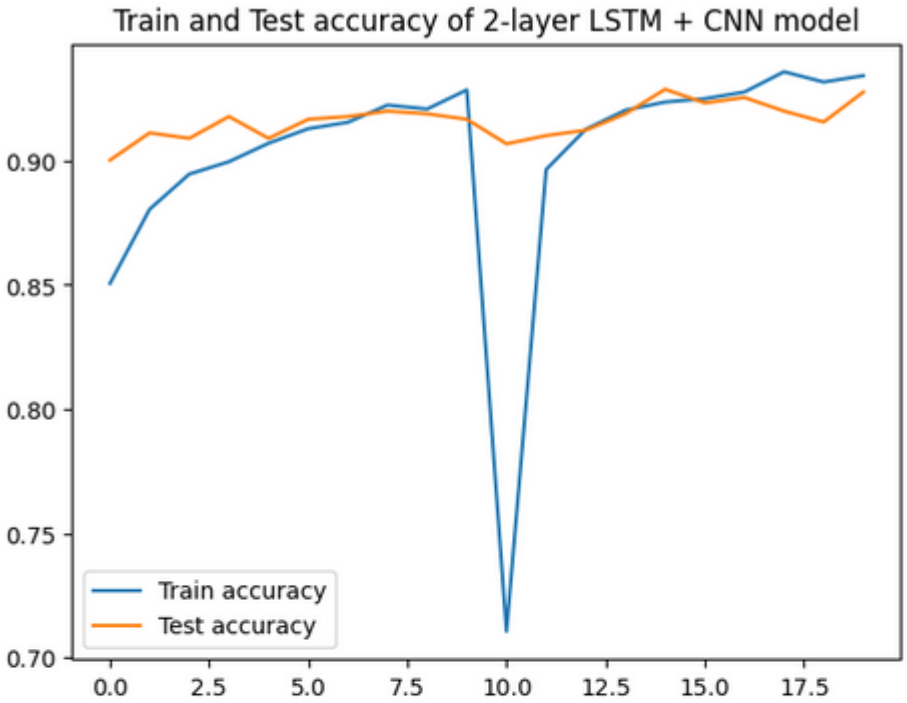
CNN + 2-Layer LSTM -

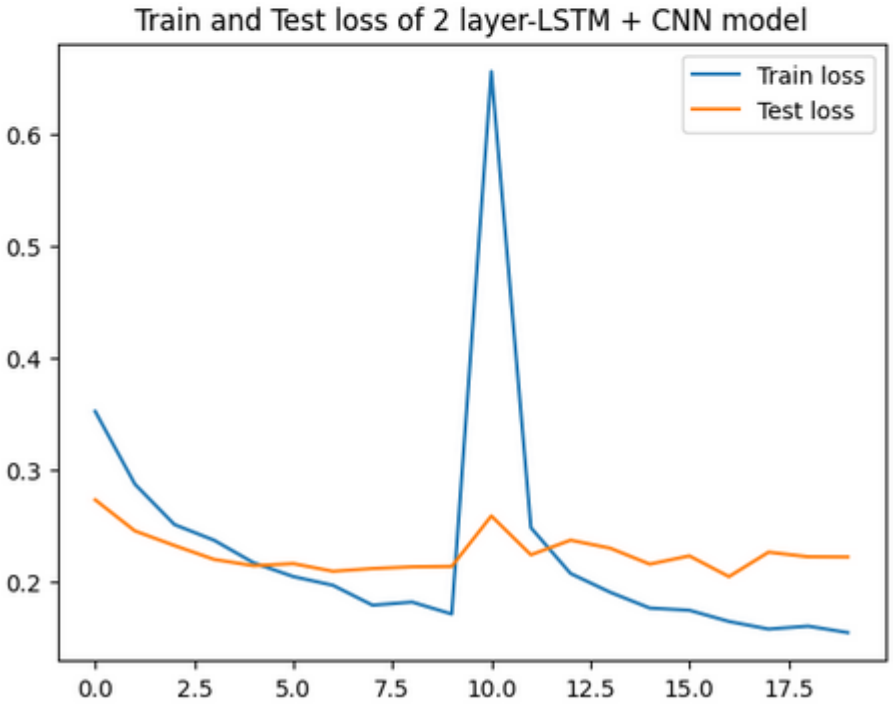
```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 601, 300)          2770500

 conv1d_17 (Conv1D)          (None, 601, 64)           57664

 activation_17 (Activation)  (None, 601, 64)           0

 max_pooling1d_17 (MaxPooli  (None, 300, 64)           0
 ng1D)

 dropout_21 (Dropout)        (None, 300, 64)           0

 lstm_40 (LSTM)              (None, 300, 300)          438000

 lstm_41 (LSTM)              (None, 300)               721200

 dense_21 (Dense)            (None, 2)                 602

=================================================================
Total params: 3987966 (15.21 MB)
Trainable params: 1217466 (4.64 MB)
Non-trainable params: 2770500 (10.57 MB)
_________________________________________________________________
```

خلاصه مدل CNN + 2-Layer LSTM

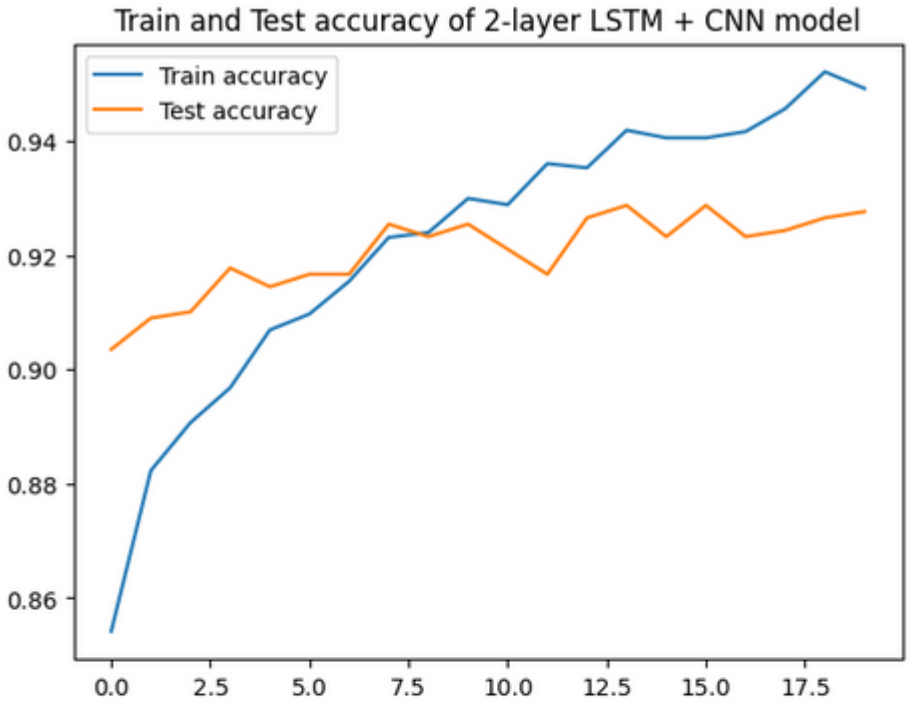
```
129/129 [==============================] - 10s 81ms/step - loss: 0.1602 - acc: 0.9316 - val_loss: 0.2223 - val_acc: 0.9156
Epoch 20/20
129/129 [==============================] - 10s 80ms/step - loss: 0.1545 - acc: 0.9342 - val_loss: 0.2221 - val_acc: 0.9276
```

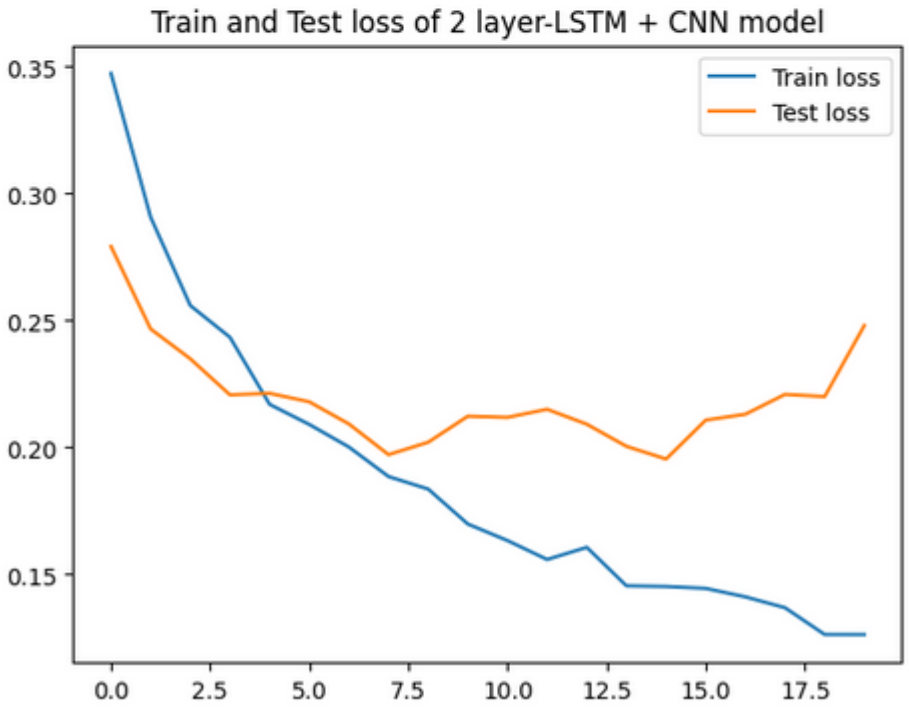
نتیجه epoch آخر مدل CNN + 2-Layer LSTM





نمودار های loss **و** accuracy **مدل CNN + 2-Layer LSTM**





نمودار های loss و accuracy مدل CNN + 2-Layer LSTM

* این شکل مربوط به ران دیگری است برای اینکه آن پرش نمودار در عکس قبلی حذف شود مجدد ران گرفتم

	precision	recall	f1-score	
Suicide_Post	0.91	0.95	0.93	
Not_Suicide_Post	0.93	0.88	0.90	
micro avg	0.92	0.92	0.92	
macro avg	0.92	0.91	0.92	
weighted avg	0.92	0.92	0.92	
samples avg	0.92	0.92	0.92	

شکل2-14. معیار های precision, recall, f1-score برای مدل **CNN + 2-Layer LSTM**

4-2. مقایسه نتایج

	Accuracy	Precision	Recall	F1-score
1-layer LSTM	92.51% / 91.78%	0.91/0.93	0.95/0.88	0.93/0.9
2-layer LSTM	91.68% / 92%	0.91/0.94	0.96/0.87	0.93/0.9
CNN+2-layer LSTM	93.42% / 92.76%	0.91/0.93	0.95/0.88	0.93/0.9

! نتایج به فرم train/test در هر خانه از جدول نوشته شده است.

مطابق جدول فوق مشاهده می کنیم دقت داده تست روی مدل CNN+2-layer LSTM از مدل 2-Layer LSTM و سپس از مدل 1-layer LSTM بیش تر است مطابق ترتیبی که در مقاله بدست آمده است. منتها مدل اول یعنی 1-layer LSTM بهتر از آنچه در مقاله بدست آمده (۸۷٪) روی دیتاست ما عمل کرده است (۹۱.۷۸٪) لذا اگر بخواهیم از مدلی استفاده کنیم که سبک تر باشد شاید بهتر باشد از این مدل استفاده کنیم.

بهبود عملکرد مدل سوم را می توان اینگونه توصیف کرد : شبکه‌های CNN در استخراج ویژگی‌های مهم از اطلاعات متنی بسیار قوی عمل می‌کنند؛ زیرا قادر به درک ویژگی‌ها و الگوهای مرتبط هستند از سوی دیگر، شبکه‌های LSTM برای درک وابستگی‌های بلندمدت و الگوهای متوالی در داده‌های توالی طراحی شده‌اند. در متن پست‌های رسانه‌های اجتماعی که کاربران اغلب افکار و احساسات خود را به صورت متوالی بیان می‌کنند، شبکه LSTM قادر است وابستگی‌های زمانی و متناسب با متن را درک کند. با ترکیب لایه‌های CNN و LSTM در مدل، ساختار مدل می‌تواند از قدرت هر دو شبکه بهره‌برداری کند. بخش CNN قادر است ویژگی‌ها و الگوهای محلی را استخراج کند، در حالی که بخش LSTM قادر است وابستگی‌های بلندمدت و اطلاعات زمینه‌ای را به خوبی درک کند. علاوه بر این، اضافه کردن لایه دوم LSTM به مدل، توانایی مدل را در درک الگوهای پیچیده‌تر افزایش می‌دهد و در نتیجه عملکرد بهتری در دسته‌بندی دارد.