به نام خدا

دانشگاه تهران

دانشکده مهندسی

برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین چهارم

نام و نام خانوادگی	فاطمه نائینیان – محمد عبائیانی
شماره دانشجویی	810198432 - 810198479
تاریخ ارسال گزارش	1401-10-6

فهرست

پاسخ ۱ – تخمین آلودگی هوا

(1-1

- **Linear interpolation method** : این متد هنگامی استفاده می شود که در یک مجموعه داده که داده های گسسته داشته باشیم و بخواهیم نقاط جدیدی در آن تولید کنیم. آسوده ترین راه این است که درونیابی خطی انجام دهیم به شکلی که دو نقطه را مد نظر قرار میدهیم و خطی بین این دونقطه رسم میکنیم. تمام نقاط این خط میتواند داده ای جدید برای مسئله باشد. با توجه به اینکه خروجی را برای چه x بخواهیم، y آن به دست می آید.

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

$$SL(x) = f(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}} \quad x \in [x_{i-1}, x_i], i = 1, 2, 3, ..., n$$

شکل 1: فرمول های **linear interpolation**

- **Pearson correlation** : این روش همبستگی خطی را بین دو مجموعه داده میسنجد. خروجی این همبستگی مقداری بین 1 و 1- خواهد داشت که هرچه به نزدیک شدن این مقدار به 1 به معنی همبستگی مثبت و 1- به معنی همبستگی منفی است. اگر مقدار ان به 0 نزدیک شود به معنی نبود همبستگی است.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

شکل 2: فرمول های **Pearson correlation**

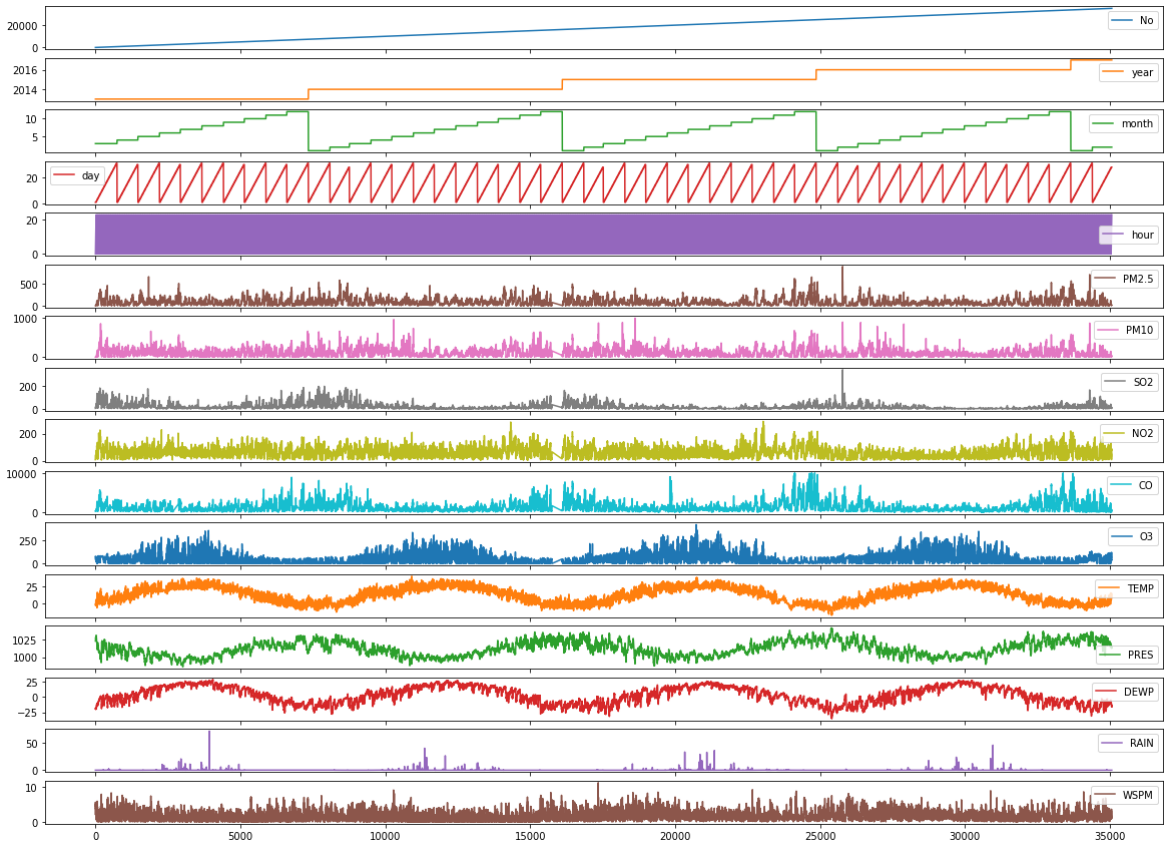
- $\boldsymbol{R^2}$: این متد میزان مناسب بودن یک مدل را اندازه گیری میکند که چقدر مدل توانسته مقدار حقیقی را تخمین بزند. این مقدار معمولا بین 0 و 1 خواهد بود که هرچه به 1 نزدیک تر شود به معنی خوب بودن مدل است.

$$R^2 = 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}}$$
$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

شکل 3: فرمول های **R2**

(2-1

با کمک کتابخانه pandas تمامی فایل های excel را میخوانیم. دو دیتافریم درست میکنیم. دیتافریم اول شامل دیتاهای ایستگاه Aotizhongxin و دیتافریم دوم شامل ستون های PM2.5 همه ایستگاه ها خواهد بود.



شکل 4: نمایش مقادیر ستون های **Aotizhongxin** بر حسب زمان

مقادیر دیتاست Aotizhongxin در گذر زمان به شکل بالا است.

(1-3-1

در این بخش missing value ها را با روش linear interpolation جایگذاری میکنیم. این امکان از طریق تابع interpolate() قابل دسترس است.

```
df_pm = pd.DataFrame()

for i in range(len(file_names)):
  df_pm[dataset[i]] = pd.read_csv(file_names[i])['PM2.5']
  df_pm[dataset[i]] = df_pm[dataset[i]].interpolate(method ='linear')
```

```
df_Aotizhongxin = pd.read_csv('PRSA_Data_Aotizhongxin_20130301-20170228.csv')
df_Aotizhongxin = df_Aotizhongxin.interpolate(method ='linear')
```

شکل 5: جایگذاری **missing values** با روش **linear interpolation**

(2-3-1

برای تبدیل این مقادیر به درجه یک map ایجاد میکنیم. سپس آن را به تک تک سطر ها اعمال میکنیم. این کار با تابع applymap امکان پذیر است.

```
mapping = {'N': 0, 'NNE': 22.5,'NE' : 45,'ENE' : 67.5,'E': 90, "ESE" : 112.5,
           "SE" : 135,"SSE" : 157.5, "S" : 180,'SSW' : 202.5, "SW" : 225,
           "WSW" : 247.5,"W": 270 , "WNW": 292.5, "NW": 315, "NNW" :  337.5}

df_Aotizhongxin = df_Aotizhongxin.applymap(lambda s: mapping.get(s) if s in mapping else s)
```

شکل 6: تبدیل ستون **wd** به درجه

(3-3-1

در روش Min-Max normalization داده های هر ستون به مقادیری بین 0 و 1 تبدیل می شوند. این تبدیل به صورت زیر انجام می شود.

$$x = \frac{x - min}{max - min}$$

شکل 7: فرمول **min max normalization**

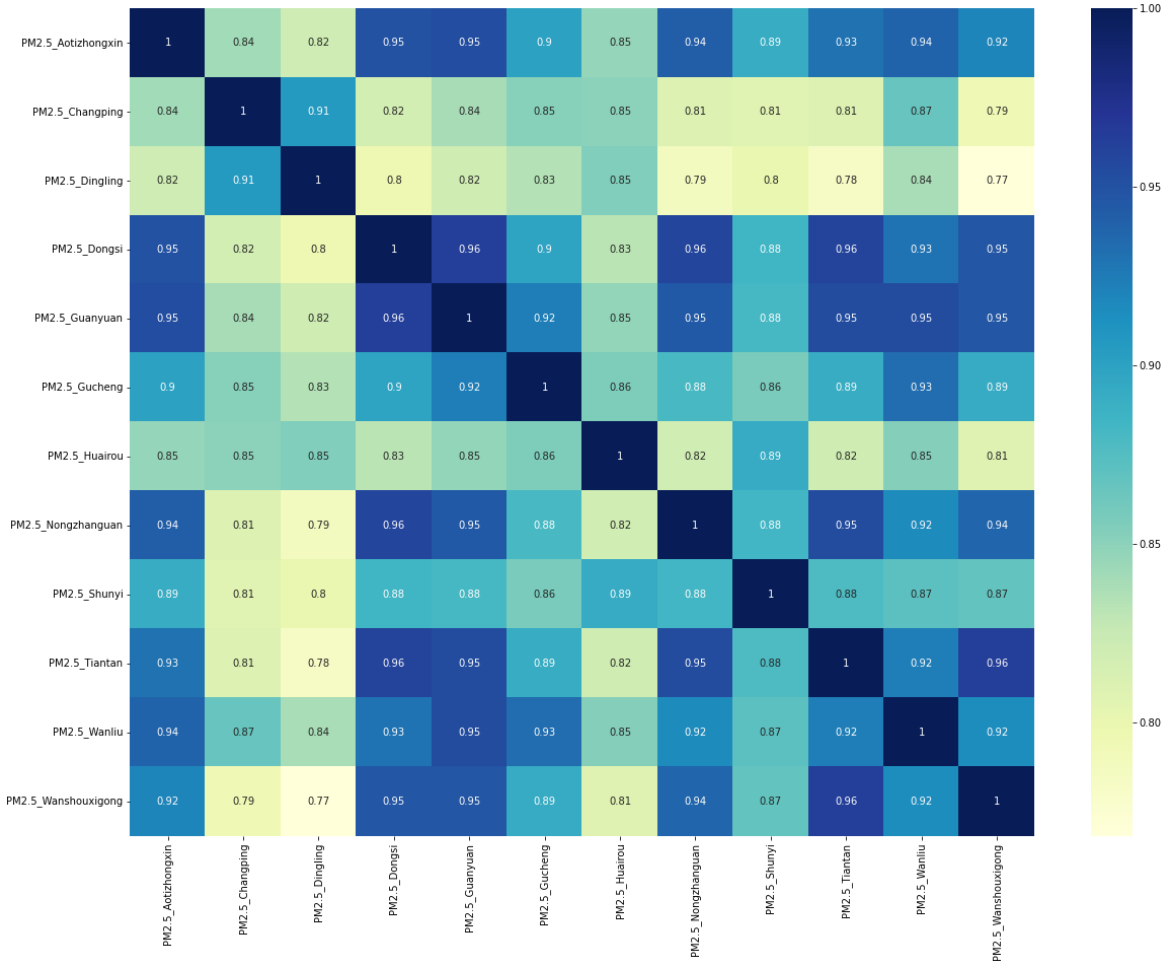
این قابلیت از طریق تابع MinMaxScaler قابل دسترسی است.

```
scaler = MinMaxScaler(feature_range = (0,1))
df_pm_scale = pd.DataFrame(scaler.fit(df_pm).transform(df_pm), columns = col)
```

شکل 8: انجام **min max normalization** در پایتون

4-3-1)

با کمک دیتافریمی که برای داده ای PM2.5 همه ایستگاه ها درست کرده بودیم، با کمک تابع corr() همبستگی را بین ستون ها محاسبه میکنیم و با heatmap از طریق کتابخانه seaborn آن را نمایش میدهیم.



شکل 9: نمودار **heatmap** همبستگی ستون های **PM2.5** همه دیتاست ها

5-3-1)

ذکر کردیم که دو دیتافریم ساخته ایم که یکی از انها شامل دیتاهای PM2.5 همه ایستگاه ها و دیگری شامل همه دیتاهای ایستگاه Aotizhongxin هستند. حال با کمک این دو دیتافریم، یک فایل اکسل میسازیم که شامل 20 ویژگی از جمله دیتاهای PM2.5 همه ایستگاه ها و wd , WSPM, PM10, CO, RAIN, DEWP, PRES, TEMP برای ایستگاه Aotizhongxin می باشد. سپس این فایل را با دستور to_csv() ذخیره میکنیم.

```
date = df_Aotizhongxin[['year','month','day','hour']]
feature_selection = pd.concat([df_Aotizhongxin[['PM10','CO','TEMP','PRES','DEWP','RAIN','wd','WSPM']],df_pm],axis=1)
df = scaler.fit_transform(feature_selection)
df = pd.DataFrame(df,columns=feature_selection.columns)
feature_selection = pd.concat([date, df],axis=1)
feature_selection.index = pd.to_datetime(feature_selection[['year', 'month', 'day', 'hour']], format='%Y %m %d %H')
feature_selection = feature_selection.drop(columns=['year', 'month', 'day', 'hour'])
feature_selection
```

```
feature_selection.to_csv('feature_selection.csv')
```

شکل 10: تشکیل دیتافریم ویژگی ها

(6-3-1

می خواهیم داده ها را به صورت supervised در بیاوریم. برای این کار لازم است ابتدا مشخص کنیم که با کمک داده ها میخواهیم چه چیزی را پیش بینی کنیم. 20 ستون داده داریم و میخواهیم با کمک این 20 ستون مقدار PM2.5 را برای ایستگاه Aotizhongxin برای ساعت بعد پیش بینی کنیم. در دو حالت lag برای 1 و 7 روز میخواهیم پیش بینی را انجام دهیم. برای این کار لازم است تا به ازای هر ساعت مشخص داده های ساعات قبل به اندازه lag*24 را نیز به مدل بدهیم. بنابر این با کمک کد زیر داده ها را اماده میکنیم تا مدل را آموزش دهیم.

```
x = np.array(feature_selection)

X = []
Y = []

lag = 24

for i in range (lag , len(x)):
  X.append(x[i-lag:i,:])
  Y.append(x[i,8:9])

X = np.array(X)
Y = np.array(Y)

X_train = X[:28032]
X_test = X[28032:]
Y_train = Y[:28032]
Y_test = Y[28032:]
```

شکل 11: نحوه ایجاد ورودی مناسب برای پیش بینی آلودگی هوا

سپس 28032 داده برای آموزش و 7012 داده برای تست خواهیم داشت.

هنگامی که lag =7 باشد نیز فقط لازم است تا lag بالا را در 7 ضرب کنیم.

(4-1

حال به سراغ شبکه می رویم. شبکه ارائه شده به روش CNN-LSTM است. در مقاله مد نظر، مقادیر hyperparameters را در شکل زیر نشان داده است.

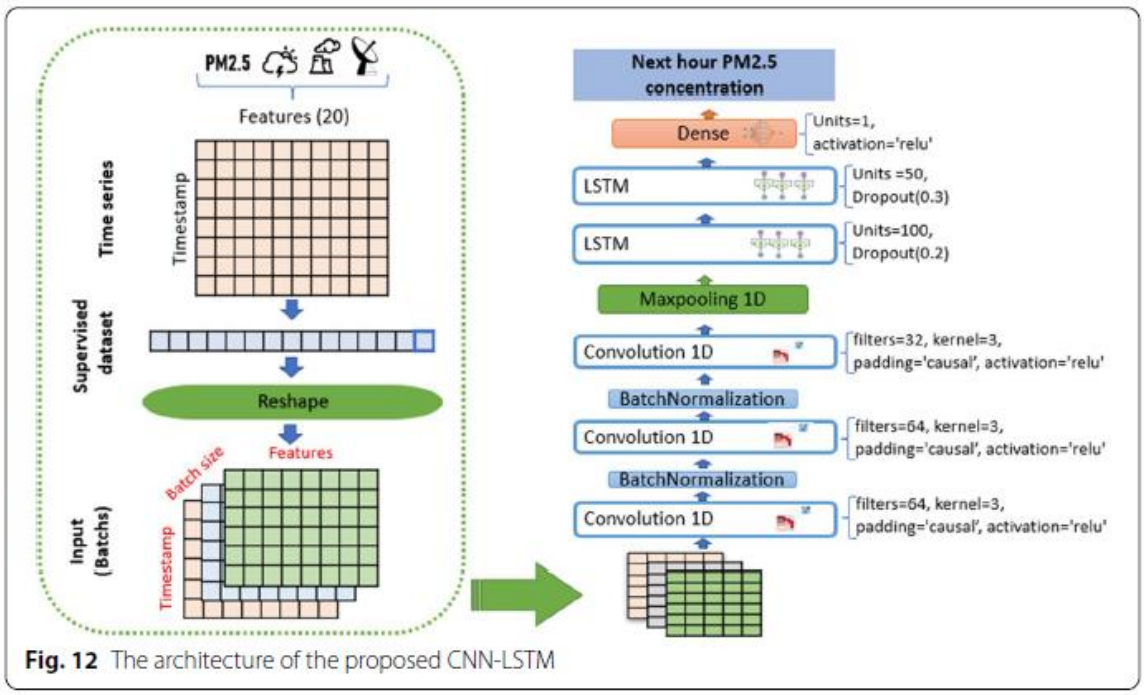


Fig. 12 The architecture of the proposed CNN-LSTM

شکل 12: پارامتر های مدل **CNN-LSTM**

مدل ترکیبی از CNN و LSTM است که سه لایه کانوولوشن و دو لایه LSTM دارد و با کمک batchnormalization و dropout سعی میکند تا از overfit شدن مدل جلوگیری کند. در نهایت یک لایه fully connected مقدار PM2.5 را در ساعت بعدی مشخص میکند. این مدل باید در 200 ایپاک و با EarlyStopping(min_delta=1e-3,patience=50) همراه با optimazier=Adam که لرنینگ ریت آن 0.001 است و با learning rate decay=0.0001 تغییر میکند. همچنین batch size برابر 32 است.

حال مدل را در پایتون شبیه سازی میکنیم.

```
callback = tf.keras.callbacks.EarlyStopping(min_delta=1e-3, patience=50, monitor="loss")

model = Sequential()
model.add(Conv1D(64, 5, padding='causal', activation="relu", input_shape=[24,20]))
model.add(BatchNormalization())
model.add(Conv1D(64, 5, padding='causal', activation="relu"))
model.add(BatchNormalization())
model.add(Conv1D(32, 5, padding='causal', activation="relu"))

model.add(MaxPooling1D())
model.add(LSTM(100,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(1, activation="relu"))

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=0.001, decay_rate=0.947, decay_steps=100)

model.compile(optimizer=Adam(learning_rate=lr_schedule), loss=tf.keras.losses.MeanSquaredError(), metrics=['mae'])
model.summary()
```

شکل 13: پیاده سازی مدل در پایتون

که خلاصه آن به شکل زیر است.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 24, 64)            3904

 batch_normalization (BatchN  (None, 24, 64)           256
 ormalization)

 conv1d_1 (Conv1D)           (None, 24, 64)            12352

 batch_normalization_1 (Batc  (None, 24, 64)           256
 hNormalization)

 conv1d_2 (Conv1D)           (None, 24, 32)            6176

 max_pooling1d (MaxPooling1D  (None, 8, 32)            0
 )

 lstm (LSTM)                 (None, 8, 100)            53200

 dropout (Dropout)           (None, 8, 100)            0

 lstm_1 (LSTM)               (None, 50)                30200

 dropout_1 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 106,395
Trainable params: 106,139
Non-trainable params: 256
```

شکل 14: خلاصه پارامتر های مدل

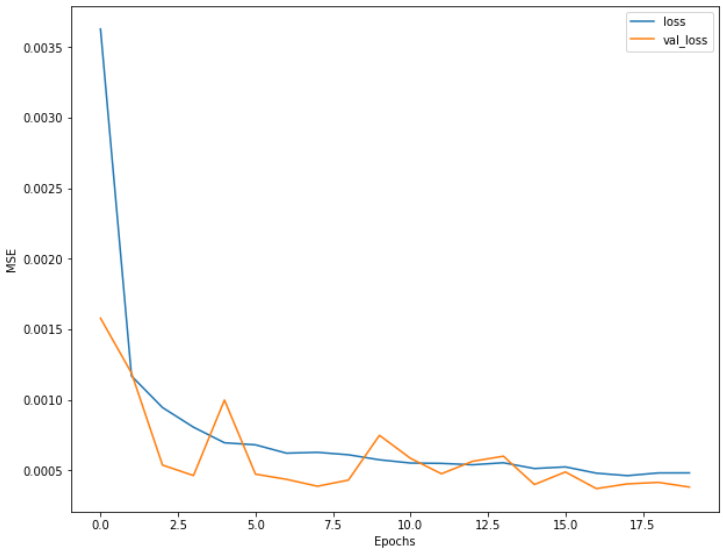
اجرای مدل برای lag=1

مدل را برای 20 ایپاک ران میکنیم. نتایج چند ایپاک اخر به شکل زیر است.

```
Epoch 16/20
438/438 - 3s - loss: 5.2531e-04 - val_loss: 4.8943e-04 - 3s/epoch - 7ms/step
Epoch 17/20
438/438 - 3s - loss: 4.8083e-04 - val_loss: 3.7145e-04 - 3s/epoch - 7ms/step
Epoch 18/20
438/438 - 3s - loss: 4.6323e-04 - val_loss: 4.0502e-04 - 3s/epoch - 7ms/step
Epoch 19/20
438/438 - 3s - loss: 4.8257e-04 - val_loss: 4.1552e-04 - 3s/epoch - 7ms/step
Epoch 20/20
438/438 - 3s - loss: 4.8293e-04 - val_loss: 3.8262e-04 - 3s/epoch - 7ms/step
```

شکل 15: پنج ایپاک اخر برای **lag=1**

در این 20 ایپاک مقدار loss به شکل زیر تغییر میکند.



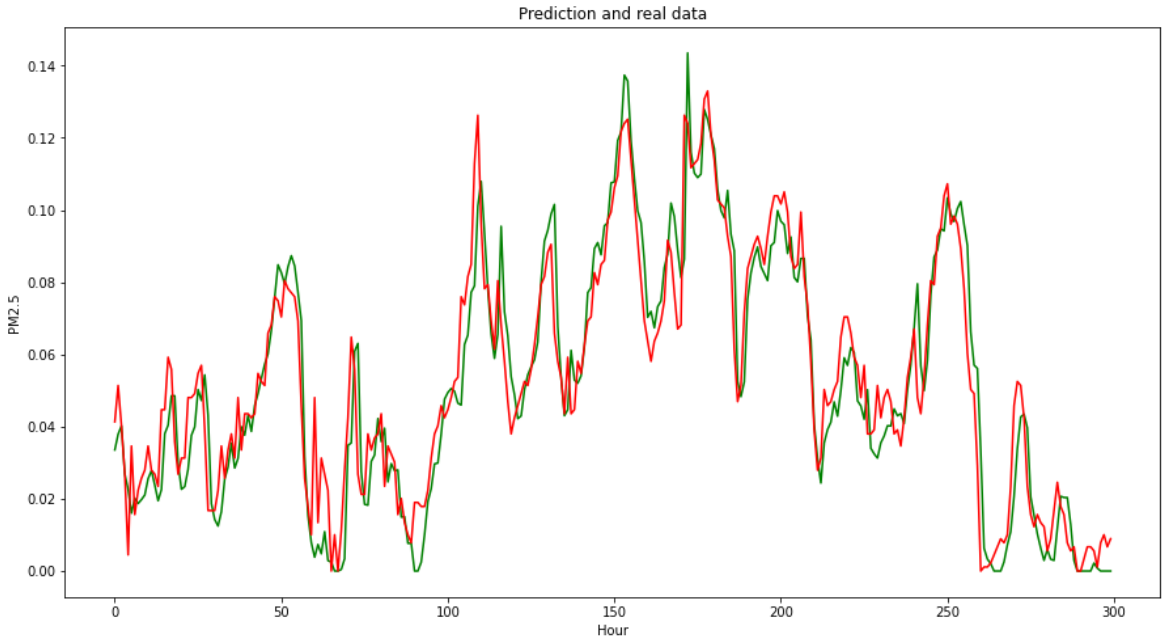
شکل 16: نمودار تغییرات **loss** برای **lag=1**

مقادیر loss های MSE و RMSE و MAE و R2 به صورت زیر است.

```
MSE  :  0.00038261893541841 81
RMSE :  0.0195606476226739
MAE  :  0.012308921757467789
R2   :  0.9561626444513789
```

شکل 17: مقادیر خطا ها برای برای **lag=1**

خروجی زیر پیش بینی مدل برای 300 ساعت از تست است.



شکل 18: پیش بینی آلودگی هوا برای **lag=1**

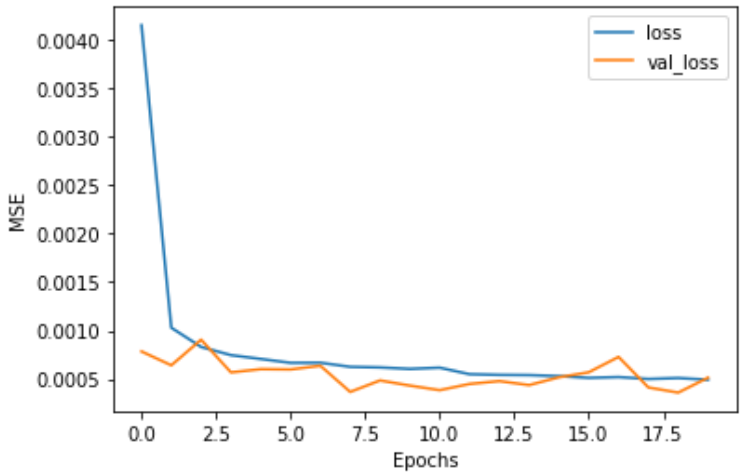
اجرای مدل برای lag=7

مدل را برای 20 ایپاک ران میکنیم. نتایج چند ایپاک اخر به شکل زیر است.

```
Epoch 16/20
876/876 [==============================] - 9s 10ms/step - loss: 5.1251e-04 - val_loss: 5.7033e-04
Epoch 17/20
876/876 [==============================] - 10s 11ms/step - loss: 5.2070e-04 - val_loss: 7.2905e-04
Epoch 18/20
876/876 [==============================] - 9s 10ms/step - loss: 5.0119e-04 - val_loss: 4.1352e-04
Epoch 19/20
876/876 [==============================] - 9s 10ms/step - loss: 5.1075e-04 - val_loss: 3.6036e-04
Epoch 20/20
876/876 [==============================] - 9s 10ms/step - loss: 4.9359e-04 - val_loss: 5.1453e-04
```

شکل 19: پنج ایپاک اخر برای برای **lag=7**

در این 20 ایپاک مقدار loss به شکل زیر تغییر میکند.



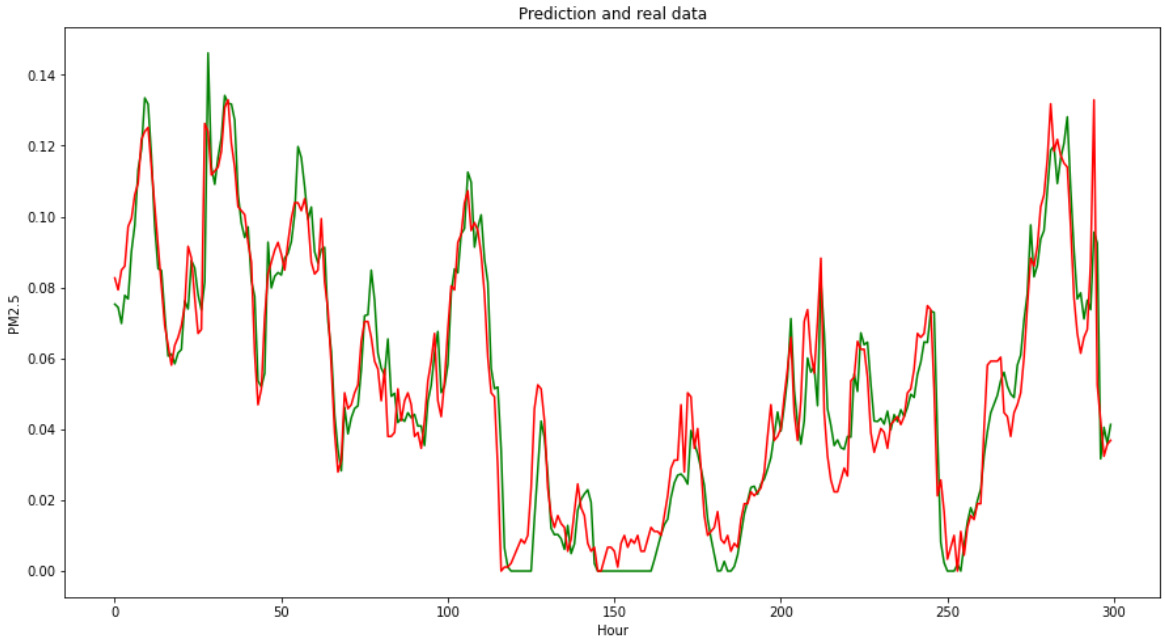
شکل 20: نمودار تغییرات **loss** برای **lag=7**

مقادیر loss های MSE و RMSE و MAE و R2 به صورت زیر است.

```
MSE  :  0.0005145308362142312
RMSE :  0.022683272167265 27
MAE  :  0.013715604493098768
R2   :  0.9419583134352644
```

شکل 21: مقادیر خطاها برای **lag=7**

خروجی زیر پیش بینی مدل برای 300 ساعت از تست است.



شکل 22: پیش بینی آلودگی هوا برای **lag=7**

ابا توجه به عکس ها میبنیم هنگامی که lag=7 است مدل توانایی پیش بینی بیشتری پیدا کرده است.

پاسخ ۲ – تشخیص اخبار جعلی

2-1)

لایه ی RNN ساده از نورون هایی استفاده میکند که توانایی ورودی دادن به نورون های هم لایه ی خود را به صورت سری دارند و میتوانند ترتیب ورودی ها را نیز در خروجی اعمال کنند.

لایه ی lstm در هر نورون خود توانایی ارسال کردن داده ی خود به سلول بعدی و همچنین فراموش کردن داده ها را دارد ، این معماری در مجموع ساختار یافته تر از لایه ی RNN ساده می باشد.

در این مقاله از دو معماری LSTM_conv و LSTM استفاده شده است که در اولی ابتدا ویژگی ها استخراج میشوند و در دومی مستقیما به لایه ی LSTM داده میشوند.

علت عملکرد خوب شبکه های RNN برای پرداز متن ارتباط ترتیبی کلمات در جملات متوالی است، به طوری که یک شبکه که خوب آموزش دیده باشد میتواند کلمات بعدی یک متن را از شروع آن حدس بزند(مانند autocomplete). در این مقاله نیز باتوجه به اینکه متون اخبار بررسی میشود ترتیب کلمات استفاده شده میتواند موثر باشد.

2-2)

برای اینکه بتوان کلمات که از جنس رشته هستند را در شبکه های عصبی وارد کرد نیاز است که این کلمات به صورت عدد encode بشوند. یکی از راه های انجام اینکار word embedding می باشد، به این صورت که هر کلمه در متن به عددی map میشود و به این صورت کلمات تکراری اعداد یکسانی را تولید میکنند.

در این مقاله برای انجام embedding از تابع keras.tokenize استفاده میشود و به این صورت عمل میکند که ابتدا ورودی را به صورت لیستی از رشته ها(یا کلمات) گرفته و اعداد را به آن ورودی fit میکند. این اعداد با توجه به تعداد ورودی ها و ترتیب آهنها تعیین میشود.

بعد از fit کردن اعداد با دادن داده های جدید به tokenizer این تابع با توجه به اعداد به دست آمده در مرحله ی قبل به رشته های جدید عددی نسبت میدهد که میتواند در مراحل بعدی برای ورودی مدل مورد استفاده قرار بگیرد.

(1-3-2

برای پیش پردازش ابتدا اعداد را در متن حذف کرده (به دلیلبی تاثیر بودن) و سپس جمله ها را tokenize کرده که به کلمات تشکیل دهنده تقسیم شوند و در نهایت نیز روی token های ایجاد شده عملیات stemmingرا پیاده سازی میکنیم.

در کتابخانه ی استفاده شده دو نوع stemmer وجود دارد با عناوین porter stemmer و Lancaster stemmer ، ما در این پیاده سازی از porter stemmer بهره میگیریم که سریع تر بوده و محاسبات کمتری دارد.

خروجی های این بخش به این صورت است (متن بالا متن اصلی و متن پایین متن بعد از process) :

```
Some  civilians killed in Syria aid convoy attack Red Cross AFP Tuesday  Sep  The air raids that
['some', 'civilian', 'kill', 'syria', 'aid', 'convoy', 'attack', 'red', 'cross', 'afp', 'tuesday
```

شکل 23: خروجی تابع **stemmer**

بعد از این مرحله با توجه به توضیحات مقاله نیاز است که word embedding انجام شود که با استفاده از تابع tokenize کتباخانه ی keras میتوان آن را پیاده کرد، همچنین طول رشته ها با استفاده از padding روی 300 تنظیم میشود

نمونه ی خروجی این بخش:

```
array([ 151,  354,   72,   31,   34, 5575,  280,  738,   75,  554,    1,
        145,   44,   94,  297,  265,  779,  516,   75,  387,    8, 1290,
        674,  211, 2610, 3470, 5679,   80, 1012,  224,    1,   72,   58,
       1833,  151,  706,   48,   80,  197,  109,  131,  240,    1,   75,
         49,  330, 5575,   94,    1, 3826,   80,  197,    8,   54,  333,
         72,   48,  619,    1,   49,  271, 1290,  208,   64, 2778,  387,
         80,  197,    8,   72,   58, 1833,  145,   48, 4223, 3351, 5680,
       5681, 5682,   75,  240,   49,  575,   48, 2990, 5683, 2643, 5684,
       4223, 3351,   10,  175,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0], dtype=int32)
```

شکل 24: خروجی تابع **tokenize**

(2-3-2

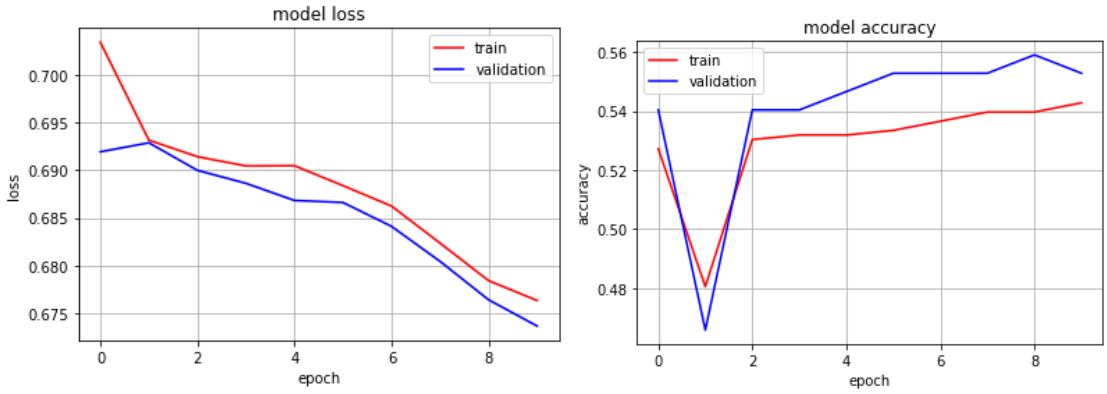
مدل **hybrid**:

این مدل همانند مدل ذکر شده در مقاله ایجاد شده است:

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 300, 300)          30000

 conv1d_5 (Conv1D)           (None, 296, 128)          192128

 max_pooling1d_5 (MaxPooling  (None, 148, 128)         0
 1D)

 lstm_5 (LSTM)               (None, 32)                20608

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 242,769
Trainable params: 242,769
Non-trainable params: 0
```

شکل 25: خلاصه پارامتر های مدل **hybrid**

نمودار های خروجی مدل به شکل زیر میباشد:



شکل 26: نمودار **loss** و **acuracy** برای مدل **hybrid**

معیار های metric مدل:

	precision	recall	f1-score	support
0	0.75	0.04	0.08	74
1	0.55	0.99	0.70	87
accuracy			0.55	161
macro avg	0.65	0.51	0.39	161
weighted avg	0.64	0.55	0.42	161

شکل 27: مقادیر معیار **metric** برای مدل**hybrid**

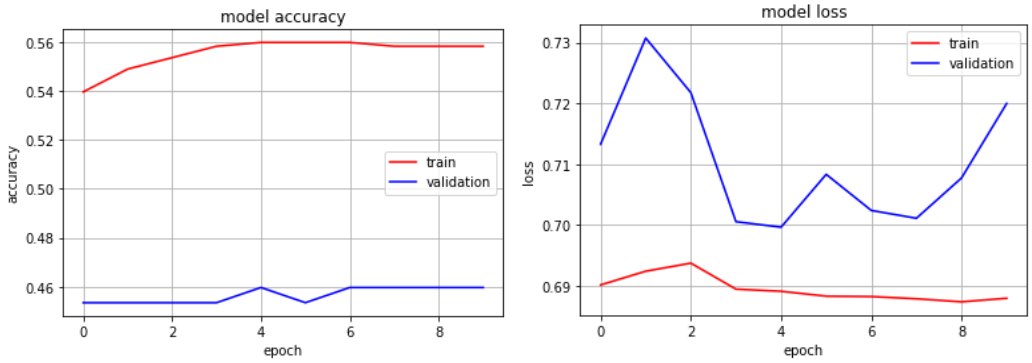
مدل RNN:

مدل دوم تنها دارای لایه ی RNN می باشد:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 300, 300)          30000

 lstm_1 (LSTM)               (None, 32)                42624

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 72,657
Trainable params: 72,657
Non-trainable params: 0
```

شکل 28: خلاصه پارامتر های مدل **RNN**

نمودار های خروجی مدل به شکل زیر میباشد:



شکل 29: نمودار **loss** و **acuracy** برای مدل **RNN**

معیار های metric مدل:

	precision	recall	f1-score	support
0	0.50	0.01	0.02	88
1	0.45	0.99	0.62	73
accuracy			0.45	161
macro avg	0.48	0.50	0.32	161
weighted avg	0.48	0.45	0.29	161

شکل 30: مقادیر معیار های **metric** برای مدل **RNN**

میتوان از نتایج به این پی برد که مدل hybrid توانایی بهتری برای پیش بینی دارد و در مجموع دقت بالاتری را ارائه میدهد که به دلیل وجود لایه ی کانولوشن و استخراج ویژگی ها می باشد.

مدل lstm زودتر در تست و ترین به حالت اشباع رسیده و دقت نهایی پایین تری دارد، اما برای مدل hybridبه نظر میرسد که با افزایش مدت یادگیری میتوان به دقت بالاتری رسید.

همچنین تقریبا تمامی پارامتر های مدل hybrid بهتر از مدل معمولی است(precision & f1_score & recall)

4-2)

به طور کلی متوان گفت که هر دو مدل دقت بیشتر در تشخیص اخبار غلط دارند، اما در تشخیص اخبار درست دچار خطای زیادی هستند، این میتواند به دلیل محدود بودن دیتاست باشد که تعداد خبر غلط پایینی دارد و مدل گاها به اشتباه اخبار درست را به عنوان خبر غلط پیش بینی میکند.

با مقایسه ی دو مدل میتوان به تاثیر لایه ی کانولوشن در بهبود عملکرد مدل پی برد. به طور کلی استخراج ویژگی ها قبل از پردازش میتواند خروجی بهتری را ارائه کند.

مدل در مجموع عملکرد قابل قبولی دارد و در صورت balance شدن دیتاست میتواند خروجی بهتری ارائه دهد، همانطور که در مقاله نیز مشخص است با دیتاست دیگر پیش بینی بسیار دقیق تری رائه میکند.

برای بهبود مدل میتوان تعداد فیلتر های convolution یا لایه های آن را افزایش داد که ممکن است در متن های پیچیده تر و طولانی تر در بهبود مدل موثر باشد.