به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

مینی‌پروژه سری ۲

نام و نام خانوادگی	مهسا قزوینی نژاد - نازنین صبری
شماره دانشجویی	۸۱۰۱۹۸۳۲۰ - ۸۱۰۱۹۸۳۱۲
تاریخ ارسال گزارش	۲۰ خرداد ۱۳۹۹

فهرست گزارش سوالات

سوال ۱ – طراحی شبکه‌های عصبی

سوال ۱

برای این سوال به دو شیوه‌ی مختلف تقسیم بندی داده‌ها را انجام می‌دهیم. شیوه‌ی اول که در شکل ۱.۱ نشان داده شده است، چگونگی تقسیم داده‌ها به دو دسته‌ی X و Y به طوری که داده‌ی ورودی هر مقداری که پیش‌بینی می‌شود ۱۱ ساعت قبلی باشد را نشان می‌دهد. همانطور که می‌توان دید در کد با شروع از نمونه ۱۲ ام، ۱۱ نمونه قبلی را به عنوان ویژگی‌های ورودی در X (ورودی مدل) ریخته و مقدار آلودگی در ساعت دوازدهم در Y (مقداری که مدل پیش‌بینی می‌کند) قرار می‌گیرد. تفاوتی که این روش با نکته‌ی بیان شده در صورت سوال دارد این است که در این حالت داده‌های تست برای تمامی ساعات خواهند بود و نه فقط ساعت ۱۲ و ۲۴. برای تغییر این مسئله پیاده سازی دیگری انجام می‌دهیم که در شکل ۱.۲ نمایش داده شده است. در این کد از تابع Timeseries Generator استفاده می‌کنیم و برای داده‌های تست نیز به طور خاص اندازه‌ی stride را برابر با ۱۲ قرار می‌دهیم تا به این شکل تنها دو ساعت خواسته شده را در نظر بگیرد.

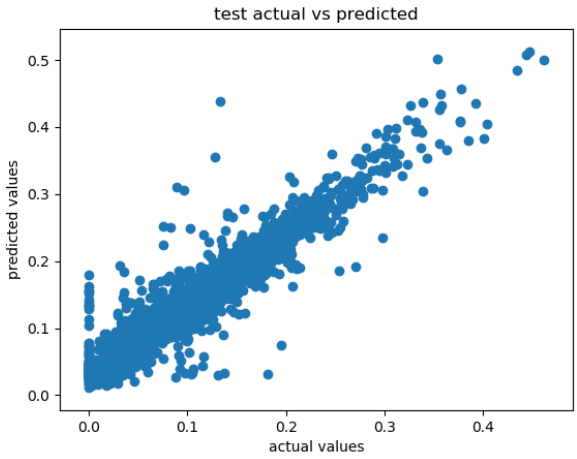
```python
def prep_data_for_model_method_1(data, window_size):
    X, y = [], []
    for i in range(data.shape[0]):
        if i<=window_size-1:
            continue
        else:
            X.append(data[i-(window_size):i])
            y.append([data[i][0]])
    X = np.array(X)
    y = np.array(y)
    return X, y
```

شکل ۱.۱: تابع جداکننده‌ی داده‌ها به مجموعه‌ی ورودی و خروجی مدل

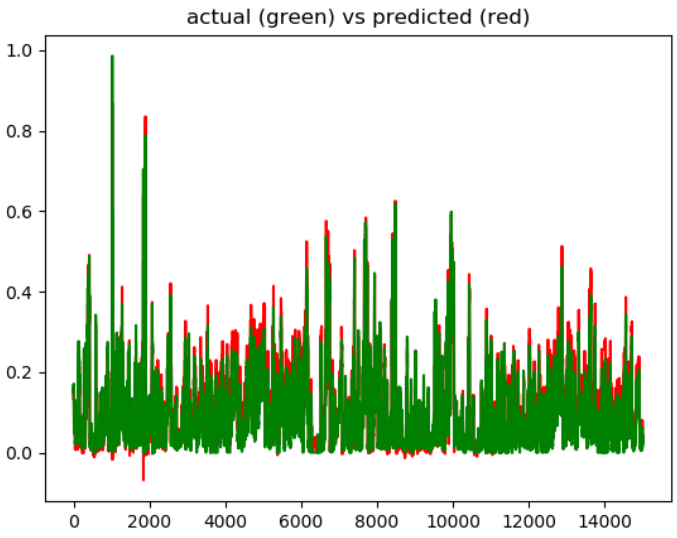
```python
def prep_data_for_model_method_using_generetor(data, window_size):
    inputs, outputs = [], []
    for val in data:
        outputs.append(val[0])
        inputs.append(val)
    inputs, outputs = np.array(inputs), np.array(outputs)
    data_gen_train = TimeseriesGenerator(
        inputs,
        outputs,
        length=11,      You, 32 minutes ago • fix minor change with
        stride=1,
        start_index=0,
        end_index=12000,
        shuffle=False,
        reverse=False,
        batch_size = 1
    )

    X_train, y_train = [], []
    for this_val_ind in range(len(data_gen_train)):
        X_train.append(data_gen_train[this_val_ind][0][0])
        y_train.append(data_gen_train[this_val_ind][1])
    X_train, y_train = np.array(X_train), np.array(y_train)

    data_gen_test = TimeseriesGenerator(
        inputs,
        outputs,
        length=11,
        stride=12,
        start_index=12000,
        end_index=15000,
        shuffle=False,
        reverse=False,
        batch_size = 1
    )

    X_test, y_test = [], []
    for this_val_ind in range(len(data_gen_test)):
        X_test.append(data_gen_test[this_val_ind][0][0])
        y_test.append(data_gen_test[this_val_ind][1])
```

شکل ۱.۱: تابع جداکننده‌ی داده‌ها به مجموعه‌ی ورودی و خروجی مدل به کمک **timeseries generator**

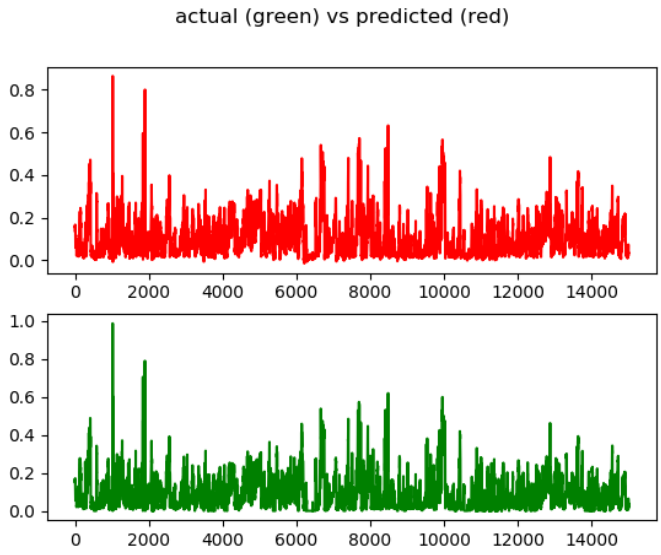
برای نمایش دقت مدل نیز ۳ شکل می‌سازیم. شکل ۱.۳ نمونه‌ی اول است که در آن مقدار حقیقی و مقدار پیش‌بینی شده برای زمان واحدی را به صورت scatter نمایش می‌دهد. در حالت ایده‌آل دوست داریم که نمودار به صورت خطی گذرنده از مبدا با شیب ۴۵ درجه باشد، یعنی مقدار پیش‌بینی شده و مقدار حقیقی دقیقا یکی باشند. شکل ۱.۴ یک نمونه دیگر برای بررسی میزان مطلوب بودن مدل است که در آن نمودار واقعی و پیش‌بینی شده با رنگ‌های مختلف به صورت سری زمانی روی هم رسم می‌شوند، اما همان‌طور که می‌توان دید این روش نمایش خیلی گویا نیست و تشخیص دو نمودار از هم دشوار است. به همین دلیل به سراغ شکل ۱.۵ می‌رویم که همان ایده‌سری زمانی‌های قبل است اما این بار با این تفاوت که به جای کشیدن آن‌ها روی هم آن‌ها را زیر هم می‌کشیم. این بار علاقه داریم که دو نمودار کاملا شبیه هم باشند. در این نمودارها باید به بازه‌ی تغییرات محورهای عمودی دقت کرد.



شکل ۱.۳: نمودار **scatter plot** برای بررسی میزان مطلوب بودن پیش‌بینی‌های انجام شده



شکل ۱.۴: نموداری برای بررسی میزان مطلوب بودن پیش‌بینی‌ها با رسم آن‌ها به شکل **time series** روی یک نمودار



شکل ۱.۵: نسخه‌ای مشابه شکل ۱.۴ با این تفاوت که نمودارها را زیر یکدیگر رسم می‌کنیم (به جای روی یک نمودار)

سوال ۲

در این بخش به طراحی شبکه‌های عصبی خواسته شده می‌پردازیم. از آنجا که هیچ چیزی درباره‌ی این مدل‌ها و چگونگی طراحی آن‌ها بیان نشده است، طراحی هر یک را به صورت دلخواه و به صورت ساده انجام می‌دهیم. شکل‌های ۱.۶، ۱.۷ و ۱.۸ به ترتیب هر یک از مدل‌های RNN، GRU و LSTM را نمایش می‌دهند. تعداد ایپاک‌های آموزش را برابر با ۲۰ قرار می‌دهیم. برای مدل‌های گزارش شده در این قسمت تابع بهینه‌سازی adam و تابع هزینه (خطا) MAE را در نظر می‌گیریم.

برای هر یک از حالات نمودارهای میزان loss در حین آموزش را رسم می‌کنیم تا ببینیم خطا به چه شکل کاهش پیدا می‌کند و در کدام نمودار سریع‌تر به حداقل مقدار خود می‌رسد و تقریبا ثابت می‌شود. مدت زمان آموزش هر یک را نیز به کمک توابع time در پایتون محاسبه می‌کنیم.

این کار را یکبار برای داده‌های آماده شده به روش اول و با داشتن تمامی ساعات در میان داده‌های تست انجام می‌دهیم که نتایج آن به شرح زیر نمایش داده شده اند:

نتایج زمان‌های اجرا در جدول ۱.۱ نمایش داده شده است و نمودارهای میزان خطا در شکل‌های ۱.۹، ۱.۱۰ و ۱.۱۱ گزارش شده‌اند.

سپس همین کار را با داده‌هایی که تست آن‌ها فقط برای ساعت ۱۲ و ۲۴ است تکرار می‌کنیم و نتایج مطابق جدول ۱.۲ و شکل‌های ۱.۱۲، ۱.۱۳ و ۱.۱۴ خواهد بود.

```
def create_model(self):
    model = Sequential()
    model.add(LSTM(30, return_sequences= True, input_shape=(self.trai
    model.add(LSTM(30))
    if self.add_dropout:
        model.add(Dropout(0.2))
    model.add(Dense(1))
    model.compile(loss=self.loss_function, optimizer=self.optimizer)
    self.model = model
```

شکل ۱.۶: کد مدل **LSTM**

```
def create_model(self):
    model = Sequential()
    model.add(GRU(30, return_sequences= True, input_shape=(self.train_X.s
    model.add(GRU(30))
    if self.add_dropout:
        model.add(Dropout(0.2))
    model.add(Dense(1))
    model.compile(loss=self.loss_function, optimizer=self.optimizer)
    self.model = model
```

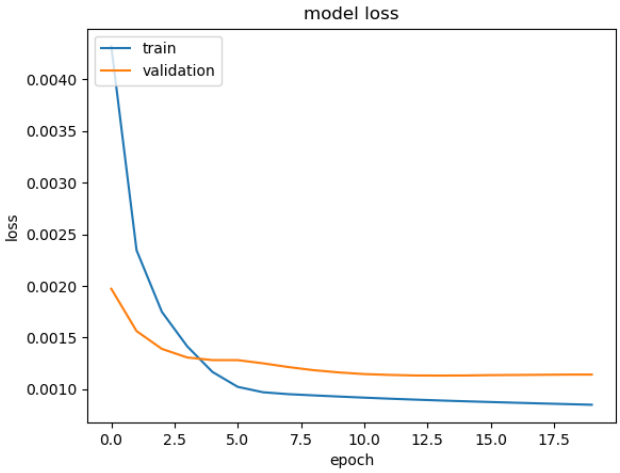
شکل ۱.۷: کد مدل **GRU**

```
def create_model(self):
    model = Sequential()
    model.add(SimpleRNN(30, return_sequences= True, input_shape=(self.
    model.add(SimpleRNN(30))
    if self.add_dropout:
        model.add(Dropout(0.2))
    model.add(Dense(1))
    model.compile(loss=self.loss_function, optimizer=self.optimizer)
    self.model = model
```

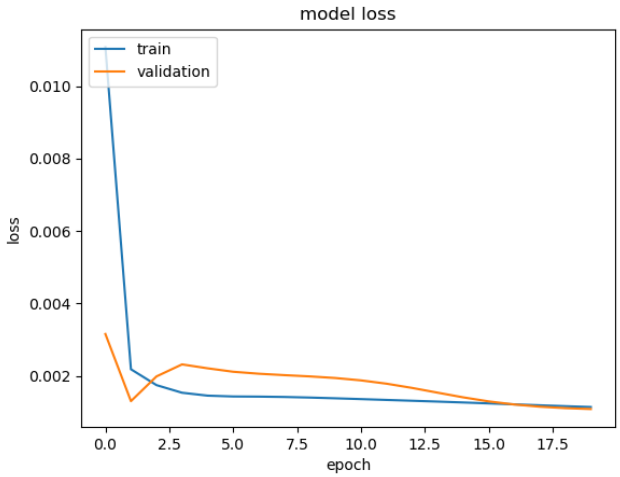
شکل ۱.۸: کد مدل **RNN**

جدول ۱.۱: نتایج مدت زمان اجرا زمانی که داده‌های تست در هر ساعتی باشند

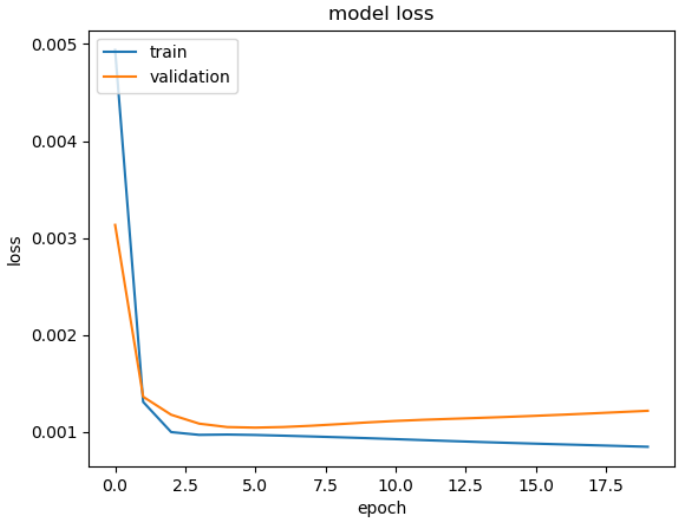
Model	Run time
LSTM	168.30
GRU	120.23
RNN	44.79



شکل ۱.۹: نمودار تغییرات میزان خطا برای مدل **LSTM**



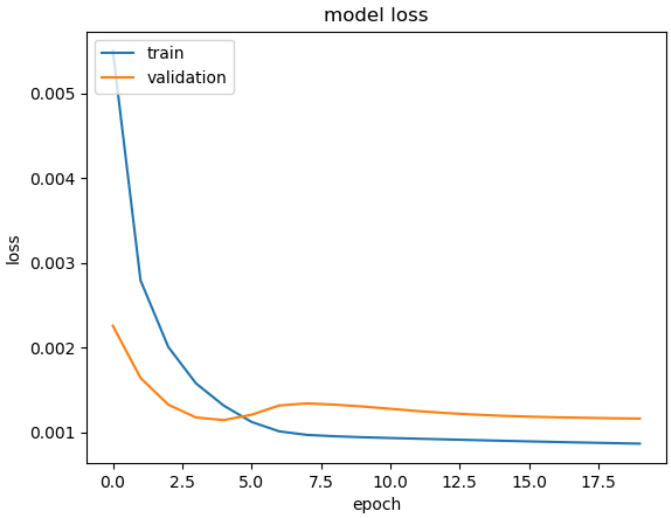
شکل ۱.۱۰: نمودار تغییرات میزان خطا برای مدل **RNN**



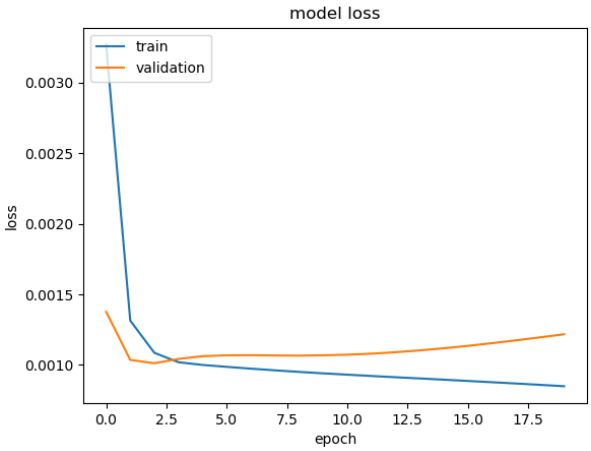
شکل ۱.۱۱: نمودار تغییرات میزان خطا برای مدل **GRU**

جدول ۱.۲: نتایج مدت زمان اجرا زمانی که داده‌های تست در ساعت ۱۲ و ۲۴ باشند

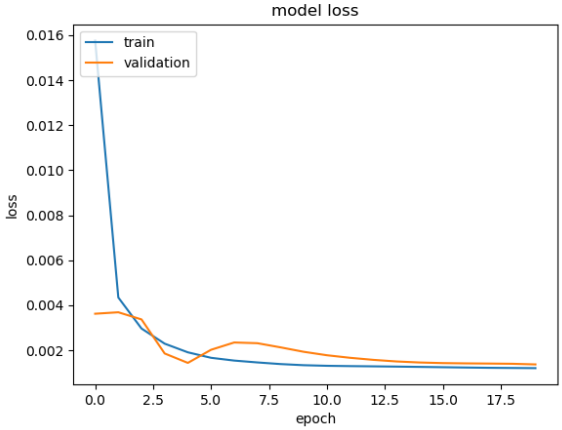
Model	Run time
LSTM	182.94
GRU	135.00
RNN	53.58



شکل ۱.۱۲: نمودار تغییرات میزان خطا برای مدل **LSTM** با حالت دوم جداسازی داده‌های تست



شکل ۱.۱۳: نمودار تغییرات میزان خطا برای مدل **GRU** با حالت دوم جداسازی داده‌های تست

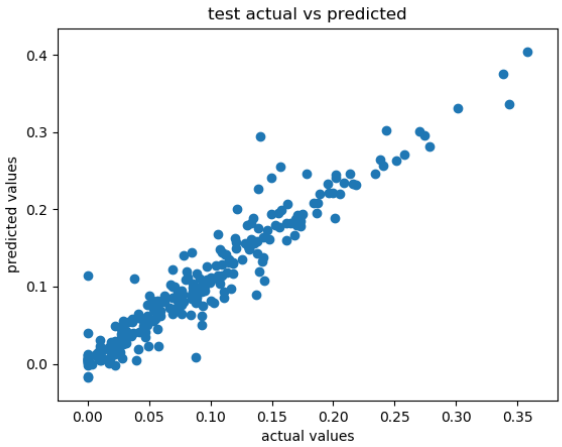


شکل ۱.۱۴: نمودار تغییرات میزان خطا برای مدل **RNN** با حالت دوم جداسازی داده‌های تست

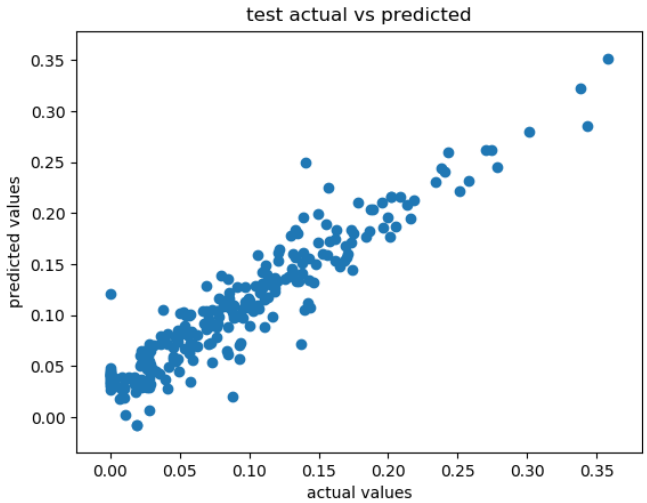
قطعا انتظار نداریم که شیوه‌ی جدا کردن داده‌های تست تاثیری روی مدت زمان آموزش مدل داشته باشد و تفاوت‌های مشاهده شده نیز ارتباطی به این موضوع ندارند و در اجرا های مختلف این اعداد به راحتی می‌توانند تغییر کنند و اجرا شدن برنامه‌های دیگر به صورت همزمان یا مواردی از این قبیل نیز می‌تواند تاثیر داشته باشند. نکته‌ای که در این نمودارها به دنبال آن هستیم مقایسه‌ی این مقادیر برای ۳ مدل است. به طور مشابه شیوه‌ی انتخاب داده‌های تست روی نمودارها نیز بی‌تاثیر است و علت آوردن هر دو حالت کامل بودن گزارش است.

با توجه به این نتایج می‌توان دید که مدل RNN کمترین مدت زمان آموزش را دارد اما میزان خطای آن به اندازه‌ی دو مدل دیگر کاهش پیدا نمی‌کند. بیشترین مدت زمان آموزش نیز متعلق به LSTM است که عملکرد آن مشابه GRU اما اندکی بهتر است. به همین دلیل نتیجه می‌گیریم که با توجه به اینکه مدت آموزش LSTM خیلی بیشتر از GRU نیست ولی دقت آن بهتر است، این مدل مطلوب‌ترین مدل ما است.

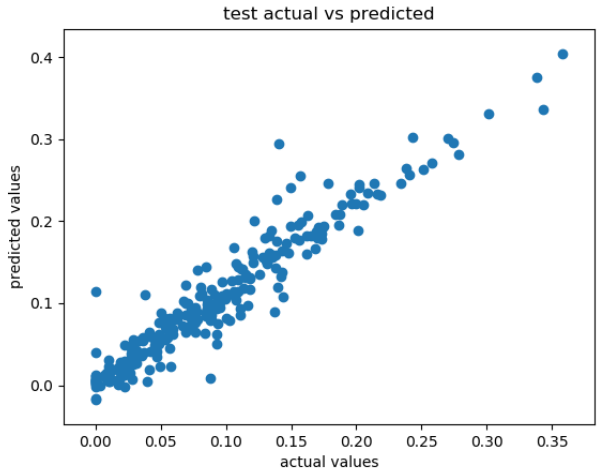
شکل‌های ۱.۱۵ تا ۱.۲۰ عملکرد این مدل‌ها در پیش‌بینی مقادیر آموزش و تست را نمایش می‌دهند.



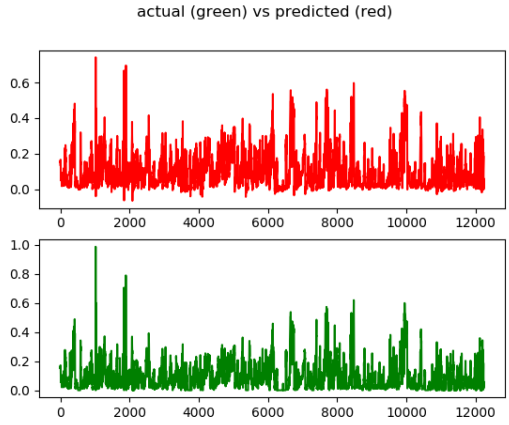
شکل ۱.۱۵: نمودار نمایش دهنده‌ی مقدار حقیقی و پیش‌بینی شده برای داده‌های تست به کمک مدل **GRU**



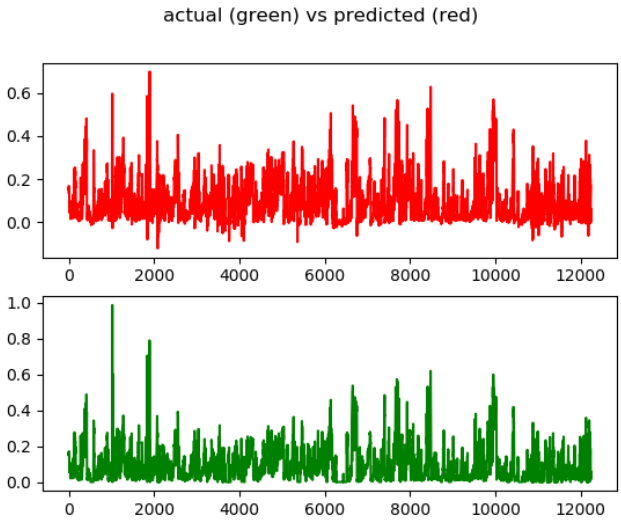
شکل ۱.۱۶: نمودار نمایش دهنده‌ی مقدار حقیقی و پیش‌بینی شده برای داده‌های تست به کمک مدل **RNN**



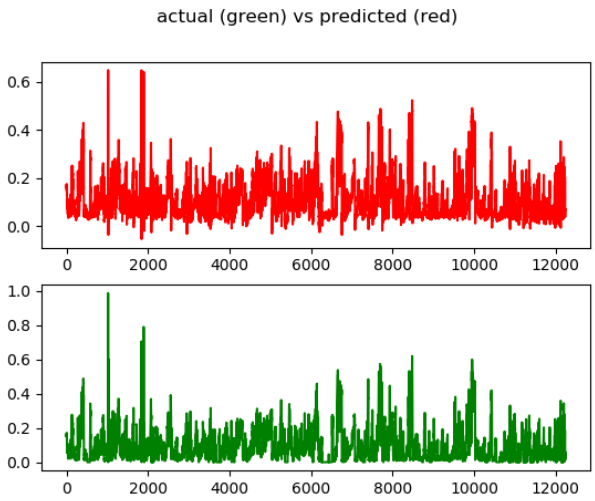
شکل ۱.۱۷: نمودار نمایش دهنده‌ی مقدار حقیقی و پیش‌بینی شده برای داده‌های تست به کمک مدل **LSTM**



شکل ۱.۱۸: نمودار زمانی مقدار پیش‌بینی شده و واقعی برای داده‌های آموزش و تست به کمک مدل **LSTM** (داده‌های تست با اینکه در فواصل ۱۲ ساعته هستند برای سادگی به صورت چسبیده رسم شده اند)



شکل ۱.۱۸: نمودار زمانی مقدار پیش‌بینی شده و واقعی برای داده‌های آموزش و تست به کمک مدل **GRU** (داده‌های تست با اینکه در فواصل ۱۲ ساعته هستند برای سادگی به صورت چسبیده رسم شده اند)



شکل ۱.۱۸: نمودار زمانی مقدار پیش‌بینی شده و واقعی برای داده‌های آموزش و تست به کمک مدل **RNN** (داده‌های تست با اینکه در فواصل ۱۲ ساعته هستند برای سادگی به صورت چسبیده رسم شده اند)

سوال ۳

برای این سوال کدی می‌زنیم که مقدار پارامترهای optimizer و loss را تغییر دهد و نتایج را بررسی می‌کنیم. همان طور که در شکل ۱.۲۱ نمایش داده شده است تمامی ترکیب‌های این پارامترها برای هر ۳ مدل تست می‌شود و ۱۸ نمودار تولید می‌کنیم. از آنجا که تعداد نمودارهای این بخش خیلی زیاد است از آوردن آن‌ها در متن گزارش خودداری می‌کنیم، این نمودارها در پوشه‌ی آپلود شده در پوشه‌ای به نام Generated files در فایل‌هایی با نامی‌هایی به فرمت زیر موجود اند:

Naming Format:

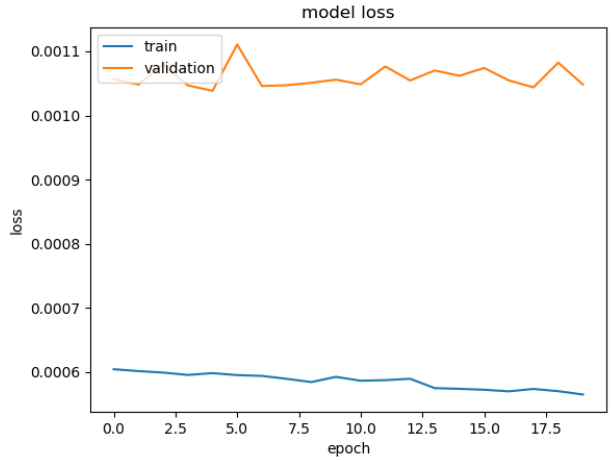
Q2_generator_<MODEL_NAME>_opt_<OPTIMIZER_NAME>_loss_<LOSS_NAME>_<TYPE_OF_PLOT_BEING_DRAWN>

```
def test_different_loss_and_optimization_functions(self):
    for opt in ['adam', 'rmsprop', 'adagrad']:
        for l in ['mae', 'mean_squared_error']:
            self.loss_function = l
            self.optimizer = opt
            print('Testing: Optimizer = {}, Loss = {}'.format(self.optimizer, self.loss_f
            self.file_save_name = self.original_file_save_name+'_opt_{}_loss_{}'.format(s
            self.train_and_report_results()
```

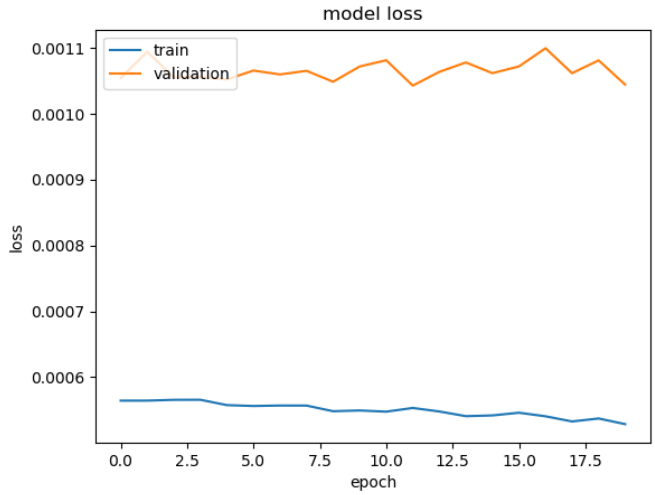
شکل ۱.۲۱: بررسی حالت‌های مختلف تابع هزینه و تابع فعال‌سازی برای هر یک از مدل‌های موجود

پس از رسم این نمودارها برای یک مدل خاص (LSTM) حالت‌های مختلف را با یکدیگر مقایسه می‌کنیم. برای تمامی حالات شکل کلی تغییر خطا کاهشی است اما در برخی حالات شیب این کاهش یا میزان آن خیلی زیاد نیست، البته این مسئله الزاما به معنی بدتر بودن مدل نیست چون در برخی موارد نقطه‌ی شروع نیز با میزان خطای کمتری است و در نتیجه با اینکه میزان کاهش در طول یادگیری خیلی زیاد نبوده است در نهایت بهتر از مدل‌های دیگر عمل کرده‌است.

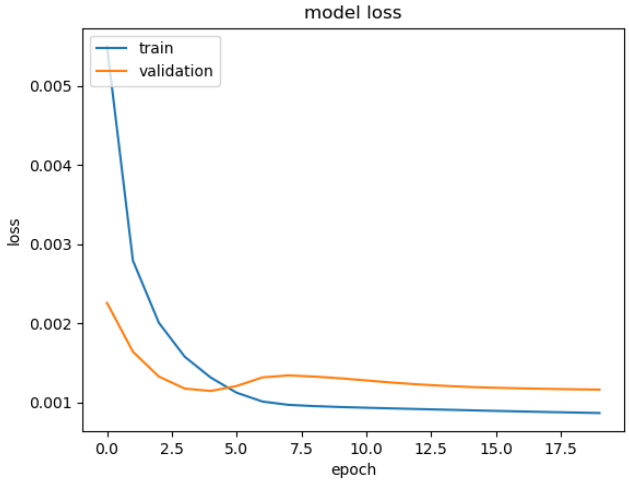
برای مدل LSTM تغییرات مقدار خطار برای ۶ حالت مختلف ترکیب این مقادیر در شکل‌های ۱.۲۲ تا ۱.۲۷ نشان داده شده اند. (۱۲ حالت دیگر در پوشه‌ی بیان شده نمایش داده شده اند)



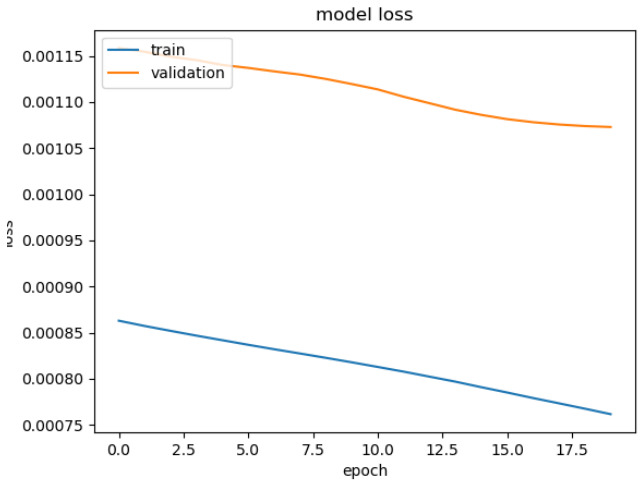
شکل ۱.۲۲: نمودار تغییرات میزان خطا برای تابع بهینه سازی **adagrad** و تابع خطای **mae**



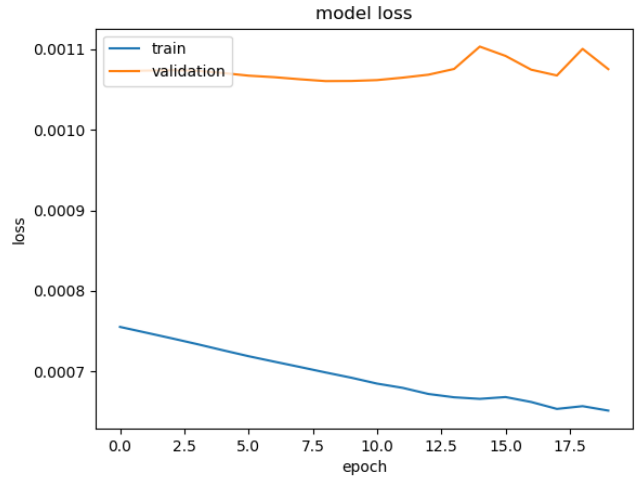
شکل ۱.۲۳: نمودار تغییرات میزان خطا برای تابع بهینه سازی **adam** و تابع خطای **mse**



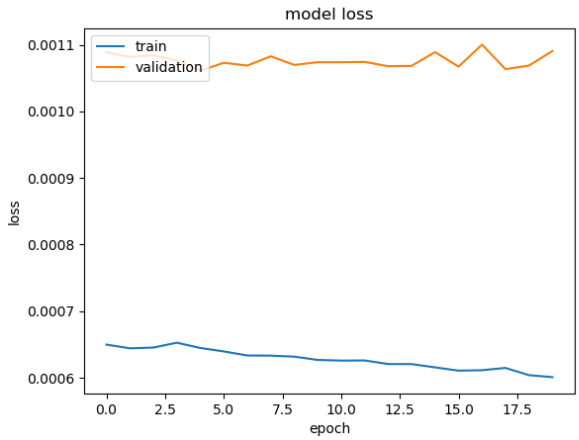
شکل ۱.۲۴: نمودار تغییرات میزان خطا برای تابع بهینه سازی **adam** و تابع خطای **mae**



شکل ۱.۲۵: نمودار تغییرات میزان خطا برای تابع بهینه سازی **adam** و تابع خطای **mse**



شکل ۱.۲۶: نمودار تغییرات میزان خطا برای تابع بهینه سازی **rmsprop** و تابع خطای **mae**



شکل ۱.۲۷: نمودار تغییرات میزان خطا برای تابع بهینه سازی **rmsprop** و تابع خطای **mse**

با توجه به نمودارها می‌توان دید که adam + mse با اینکه روند کاهشی خیلی کندی دارد ولی کمترین خطا را روی داده‌های آموزش نتیجه می‌دهد.

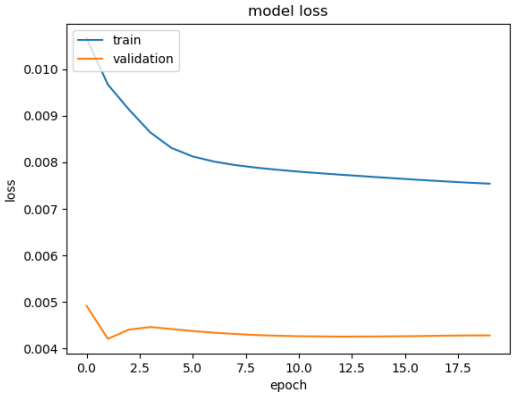
سوال ۴

برای این سوال مطابق توضیحات بیان شده در صورت سوال تقسیم بندی داده‌ها را انجام می‌دهیم و سپس نتایج را به مدل می‌دهیم. کد انجام این تقسیم‌بندی‌ها به این شکل است که ابتدا با شروع از ۲۰۱۴.۱.۱ همه‌ی داده‌های موجود را به فاصله‌ی ۱ ساعت ۱ ساعت، زمان دهی می‌کنیم سپس برای هر داده برای حالت هفتگی ساعت این داده را با کم کردن یک روز از آن به همان ساعت در روزهای قبل می‌رسانیم و آن داده‌ها را به عنوان ورودی انتخاب می‌کنیم. برای حالت ماهانه نیز به همین شکل است با این تفاوت که این بار ۷ روز از زمان کم می‌کنیم. این تفریق زمان به کمک کتابخانه‌ی datetime و تابع timedelta برای مشخص کردن مدتی که می‌خواهیم کم کنیم امکان پذیر است.

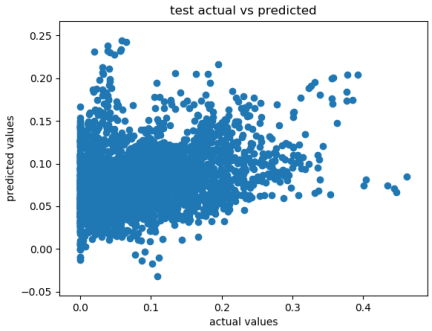
نتایج این بخش برای هر سه مدل محاسبه شده‌اند و در فایل‌هایی با نام‌هایی به شکل زیر در پوشه‌ی Generated Files قابل بررسی هستند.

[monthly|weekly]Q4_<MODEL_NAME>_model_<TYPE_OF_PLOT_BEING_DRAWN>

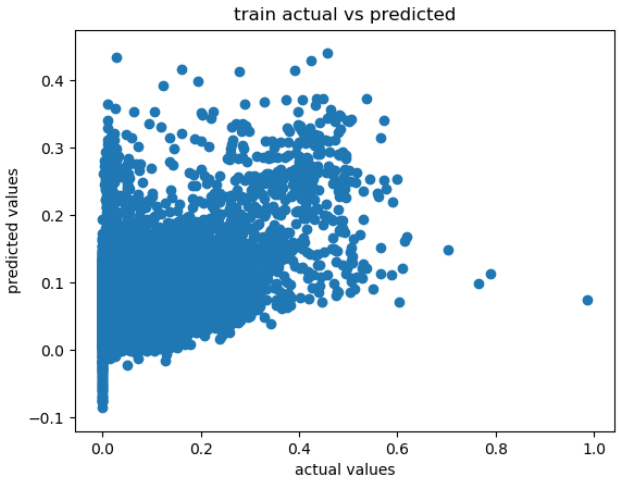
در این بخش ما نتایج مدل lstm را بررسی می‌کنیم. نتایج برای داده‌های هفتگی در شکل‌های ۱.۲۸، ۱.۲۹ و ۱.۳۰ و برای داده‌های ماهانه در شکل‌های ۱.۳۱، ۱.۳۲ و ۱.۳۳ نمایش داده شده است.



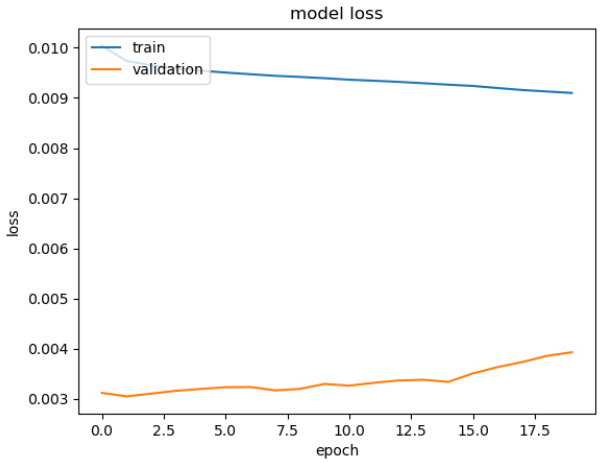
شکل ۱.۲۸: نمودار تغییر میزان خطا برای داده‌های هفتگی



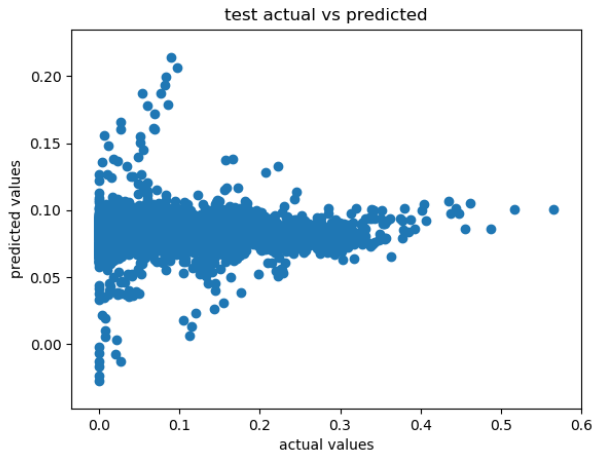
شکل ۱.۲۹: نمودار مقایسه‌ی پیش‌بینی و مقدار واقعی برای داده‌های تست هفتگی



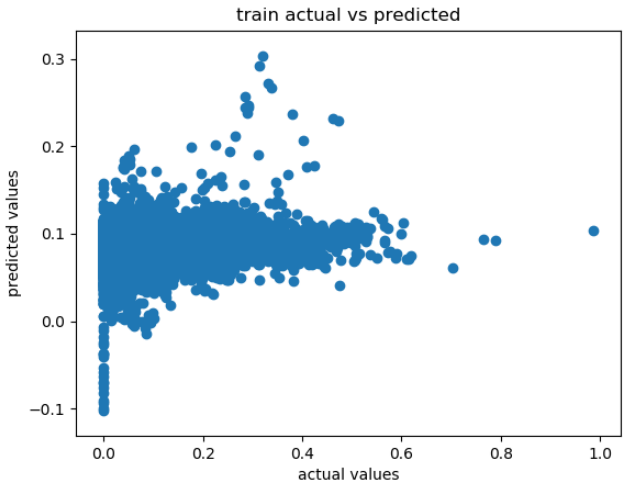
شکل ۱.۳۰: نمودار مقایسه‌ی پیش‌بینی و مقدار واقعی برای داده‌های آموزش هفتگی



شکل ۱.۳۱: نمودار تغییر میزان خطا برای داده‌های ماهانه



شکل ۱.۳۱: نمودار مقایسه‌ی پیش‌بینی و مقدار واقعی برای داده‌های تست ماهانه

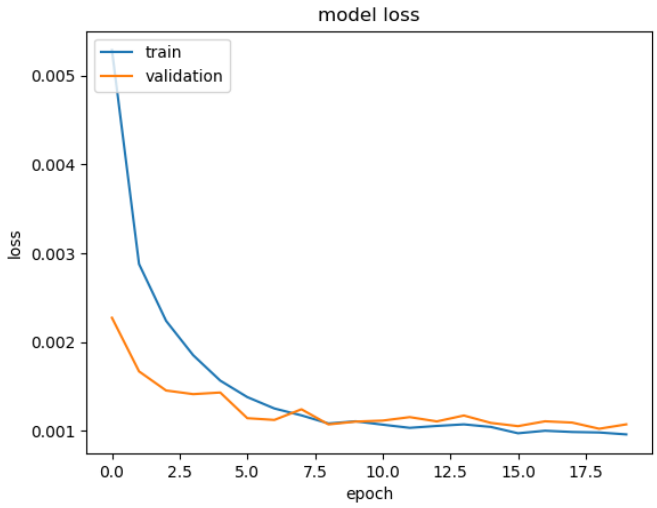


شکل ۱.۳۲: نمودار مقایسه‌ی پیش‌بینی و مقدار واقعی برای داده‌های آموزش ماهانه

می‌توان دید که به طور کلی نمودارها خیلی ضعیف‌تر از حالت قبل هستند که یک علت کمتر بودن داده‌ای موجود است. یک علت دیگر نیز می‌تواند این باشد که این معیار الزاما ترند ماهانه نداشته باشند و ساعت‌های قبلی معیارهای بهتری برای پیش‌بینی هستند تا روزها یا هفته‌های قبلی.

سوال ۵

همانطور که در شکل‌های ۱.۶، ۱.۷ و ۱.۸ دیدیم در هر یک از مدل‌ها عبارت شرطی if ای قرار دارد که در صورتی که پارامتر داشتن dropout را True کنیم این لایه را اضافه می‌کند. در حالت عادی این مقدار False است اما در این بخش آن را اضافه می‌کنیم تا تاثیر را بررسی کنیم. شبکه‌ی LSTM را در این سوال بررسی می‌کنیم. در این مدل زمانی که dropout داریم از خطای اولیه‌ی بیشتری شروع می‌کنیم ولی در نهایت پس از ۲۰ ایپاک تقریبا به همان جایی می‌رسیم که مدل اولیه بدون داشتن dropout به آن می‌رسد و بهبود چشم‌گیری را مشاهده نمی‌کنیم. شکل ۱.۳۳ تغییر خطا در این مدل را نشان می‌دهد.



شکل ۱.۳۳: نمودار تغییرات میزان خطا برای مدل **LSTM** با داشتن **drop out**

علت این مسئله می‌تواند میزان dropout انتخاب شده و کوچک بودن آن نیز باشد.

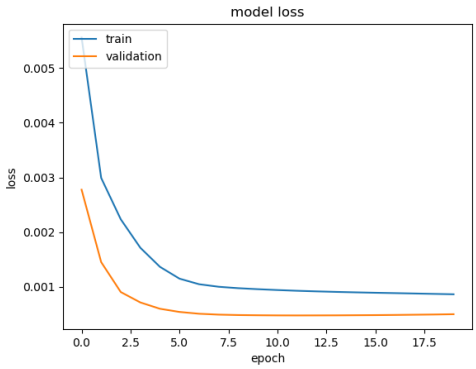
سوال ۶

برای این سوال از روش‌های جداسازی بیان شده در سوال ۱ و سوال ۴ کمک می‌گیریم و برای هر داده هر سه حالت ویژگی‌های ورودی را در نظر می‌گیریم (۱۱ ساعت پیش، هفتگی و ماهانه) سپس مدلی به شکل نشان داده شده در شکل ۱.۳۴ با یکدیگر ترکیب می‌کنیم.

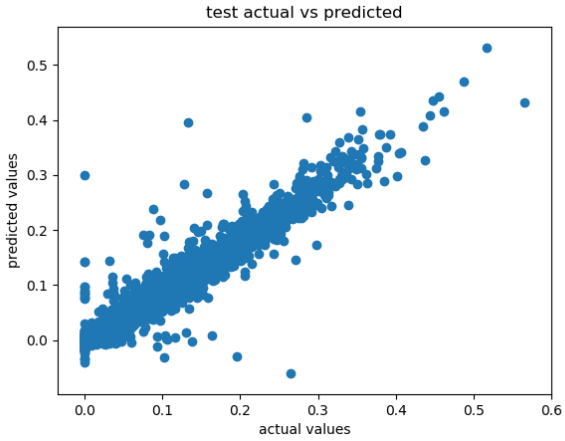
```python
def create_model(self):
    print(self.train_X_1.shape, self.train_X_3.shape, self.train_X_2.shape)
    inp1 = Input(shape=(self.train_X_1.shape[1], self.train_X_1.shape[2]))
    m1 = LSTM(30, return_sequences= True)(inp1)
    m2 = LSTM(30)(m1)
    d1 = Dense(1)(m2)

    inp2 = Input(shape=(self.train_X_2.shape[1], self.train_X_2.shape[2]))
    m3 = LSTM(30, return_sequences= True)(inp2)
    m4 = LSTM(30)(m3)
    d2 = Dense(1)(m4)


    inp3 = Input(shape=(self.train_X_3.shape[1], self.train_X_3.shape[2]))
    m5 = LSTM(30, return_sequences= True)(inp3)
    m6 = LSTM(30)(m5)
    d3 = Dense(1)(m6)

    avg_l = Average()([d1, d2, d3])

    model = keras_Model(inputs=[inp1, inp2, inp3], outputs=avg_l)
    model.compile(loss=self.loss_function, optimizer=self.optimizer)
    return model
```

شکل ۱.۳۴: ترکیب ۳ مدل با **fusion** میانگین‌گیری

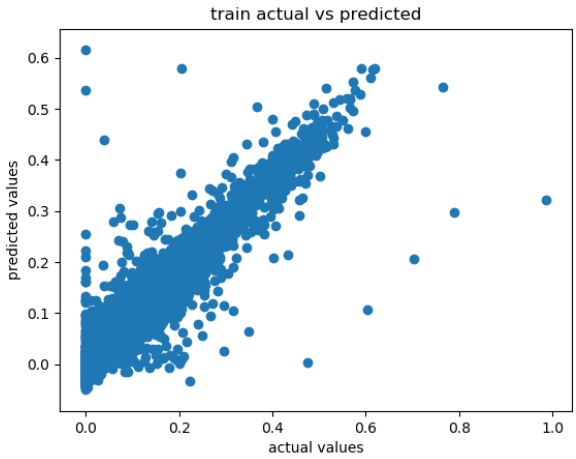
در این صورت تغییرات خطا به شکل نشان داده شده در شکل ۱.۳۵ است و مقایسه‌های پیش‌بینی‌ها در شکل‌های ۱.۳۶ و ۱.۳۷ نشان داده شده است. می‌توان دید که این نمودارها خیلی به خط ۴۵ درجه نزدیک شده اند که به این معنی است که پیش‌بینی و مقدار واقعی خیلی نزدیک هستند و مدل به طور کلی عملکرد خوبی دارد.



شکل ۱.۳۵: نمودار تغییرات خطا برای مدل ترکیبی **LSTM**



شکل ۱.۳۶: نمودار مقدار واقعی و پیش‌بینی شده برای داده‌های تست برای مدل ترکیبی



شکل ۱.۳۷: نمودار مقدار واقعی و پیش‌بینی شده برای داده‌های آموزش برای مدل ترکیبی

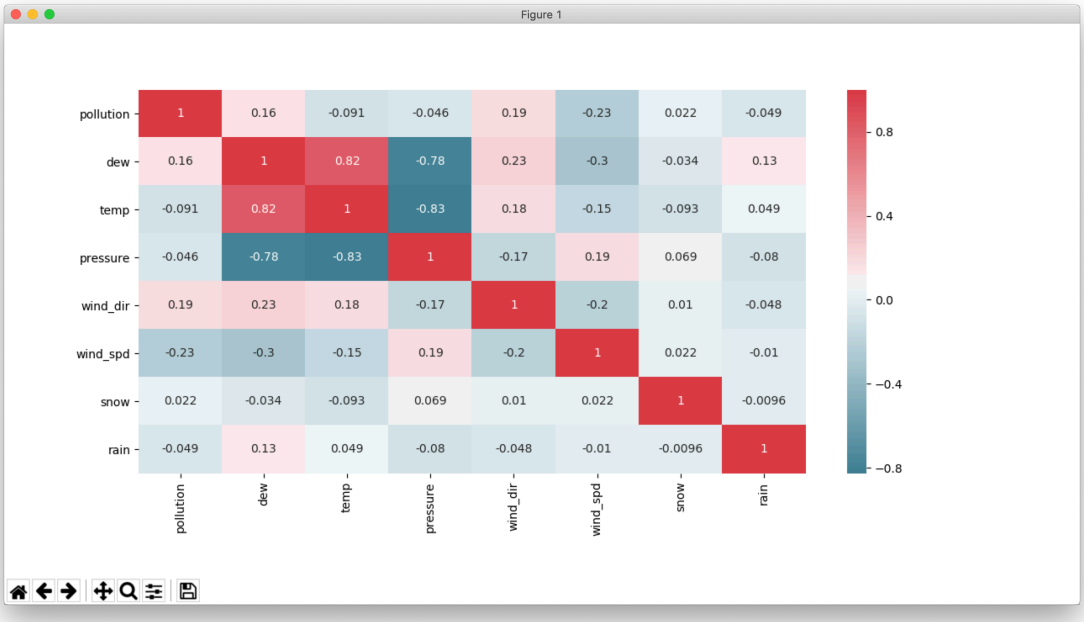
سوال ۷

برای انتخاب مهم‌ترین ویژگی‌ها باید از روش‌های feature selection انتخاب کنیم. برخی از این روش‌ها عبارتند از:

- یک روش می‌تواند محاسبه‌ی میزان Correlation بین دو به دوی ویژگی‌ها و حذف یکی از متغیر موجود در جفت‌های با correlation بالا و تکرار این کار تا رسیدن به تعداد ویژگی مورد نظر است. از آنجا که اگر correlation بالا باشد به این معنی است که دو متغیر تغییرات مشابهی دارند و روند تغییرات آن‌ها شبیه است، این مسئله باعث می‌شود که بررسی هم زمان دو ویژگی اطلاعات خیلی بیشتری از بررسی تنها یکی از آن‌ها به ما ندهد در نتیجه حذف یکی از آن‌ها باعث از دست رفتن اطلاعات زیادی نمی‌شود.
- یک روش دیگر استفاده از همین ایده‌ی correlation است اما این بار ویژگی‌هایی را نگه می‌داریم که با ویژگی که می‌خواهیم پیش‌بینی کنیم correlation بالایی دارند چون به این معنی است که در این صورت تغییرات این متغیرها روندهای مشابهی دارد و در نتیجه پیش‌بینی کننده‌های خوبی برای هم هستند.

- روش دیگر می‌تواند استفاده از تست‌هایی مثل Chi-squared باشد. در این تست ایده این است که اگر ویژگی‌ای از مقدار مورد نظر (چیزی که می‌خواهیم پیش‌بینی کنیم) مستقل باشد اطلاعات زیادی به ما نخواهد داد، در نتیجه مشابه حالت قبلی با این کار می‌توانیم یکی یکی مستقل ترین‌ها را حذف کنیم تا به تعداد مورد نظر برسیم.

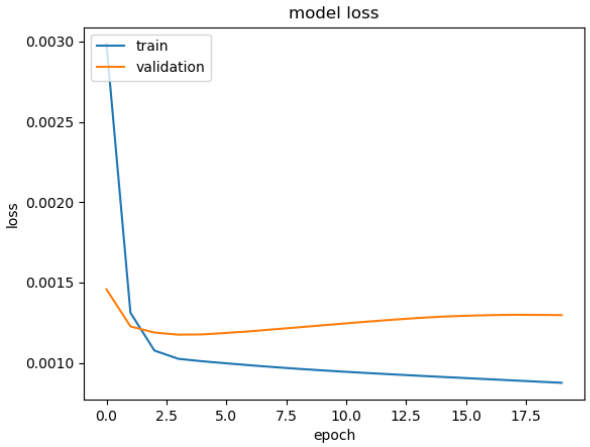
در این بخش از ایده‌ی دوم استفاده می‌کنیم و مقدار correlation میان دو به دوی ویژگی‌ها را محاسبه می‌کنیم. نتیجه در شکل ۱.۳۸ نشان داده شده است. می‌توان دید که دو ویژگی .wind dir و wind spd. بیشتر میزان correlation را با آلودگی هوا دارند به همین دلیل این دو ویژگی را به عنوان ویژگی‌های منتخب نگه‌داشته و در بخش بعد با حذف سایرین آموزش مدل را انجام می‌دهیم.



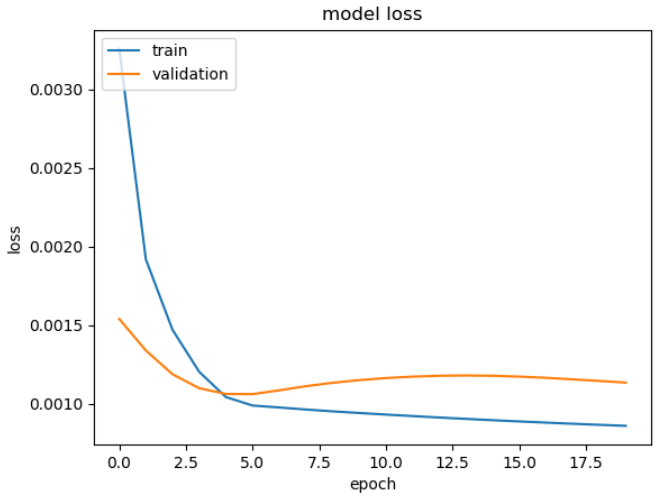
شکل ۱.۳۸: نمودار میزان **correlation** میان ویژگی‌ها

سوال ۸

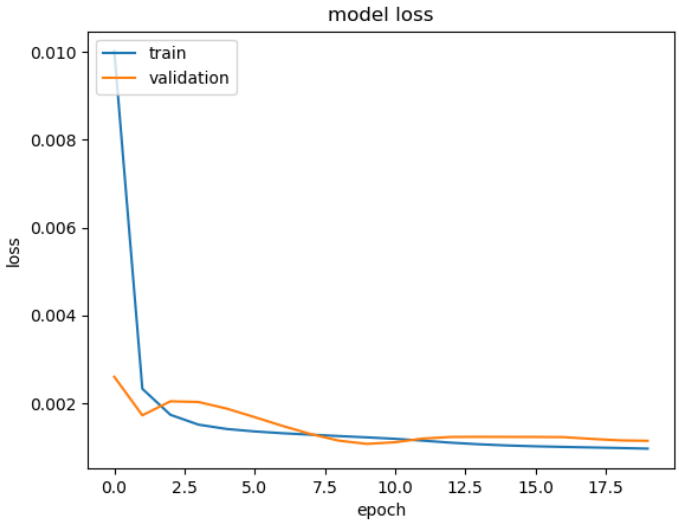
برای این سوال ۲ ویژگی بیان شده در سوال قبل را نگه می‌داریم و سایر ویژگی‌ها را دور می‌ریزیم. در این صورت نتایج نشان داده شده در شکل‌های ۱.۳۹ به بعد را خواهیم داشت. در این حالت نیز تعداد ایپاک‌های آموزش ۲۰ و تابع هزینه mae و تابع بهینه سازی adam است.



شکل ۱.۳۹: نمودار تغییرات خطا برای مدل **GRU** با ۳ ویژگی



شکل ۱.۳۹: نمودار تغییرات خطا برای مدل **LSTM** با ۳ ویژگی



شکل ۱.۳۹: نمودار تغییرات خطا برای مدل **RNN** با ۳ ویژگی

می‌توان دید که روند تغییر خطا کاهشی است و خطاها نیز به طور چشمگیری بیشتر از نتایج بخش‌های پیشین که هر ۸ ویژگی را داشتیم نشده است و به نظر می‌رسید که این ویژگی‌ها بخش خوبی از پیش‌بینی را انجام می‌دهند.

سوال ۲ – نقصان دادگان

سوال ۱

در ابتدای این سوال برای هر ستون به صورت مجزا، ۲۰ درصد از دادگان را حذف می‌کنیم. در مسئله‌های واقعی ممکن است به دلیل خرابی ابزار اندازه گیری و گاه خطای انسانی دادگان درست جمع آوری نشود یا از بین برود. برای بررسی شرایط مقابله با این مشکل در این سوال به این شکل بخشی از دادگان خود را از دسترس خارج می‌کنیم. در این قسمت به این صورت عمل می‌کنیم که به ازای هر ستون از دادگان تست آرایه‌ای تصادفی که هر خانه آن بین ۰ و ۱ است تولید می‌کنیم. سپس به ازای هر خانه از این جدول اگر مقداری کمتر از ۲۰ درصد یا همین tresh داشت آن را با مقدار np.nan جای‌گذاری می‌کنیم تا داده را از دست رفته نشان دهیم.

سوال ۲ و ۳

حال فرض می‌کنیم که داده‌های قسمت قبل را در دست داریم که در هر ستون شامل ۲۰ درصد np.nan می‌باشد. روش‌های مختلفی برای برطرف کردن نقصان داده یا از دست رفتن دادگان وجود دارد که شامل موارد زیر می‌باشد:

1. یک روش این است که کار خاصی برای پیش‌بینی دادگان از دست رفته انجام ندهیم و تنها آن داده‌ها را پاک کنیم (در برخی توابع نیز گزینه ignore کردن و در نظر نگرفتن این قبیل دادگان توسط الگوریتم وجود دارد.) این روش ساده است اما باعث از دست رفتن دادگان زیادی می‌شود و شاید برای یک ستون که تاثیری نیز در مقدار خروجی ما ندارد بخش زیادی از دادگان خود را از دست دهیم.

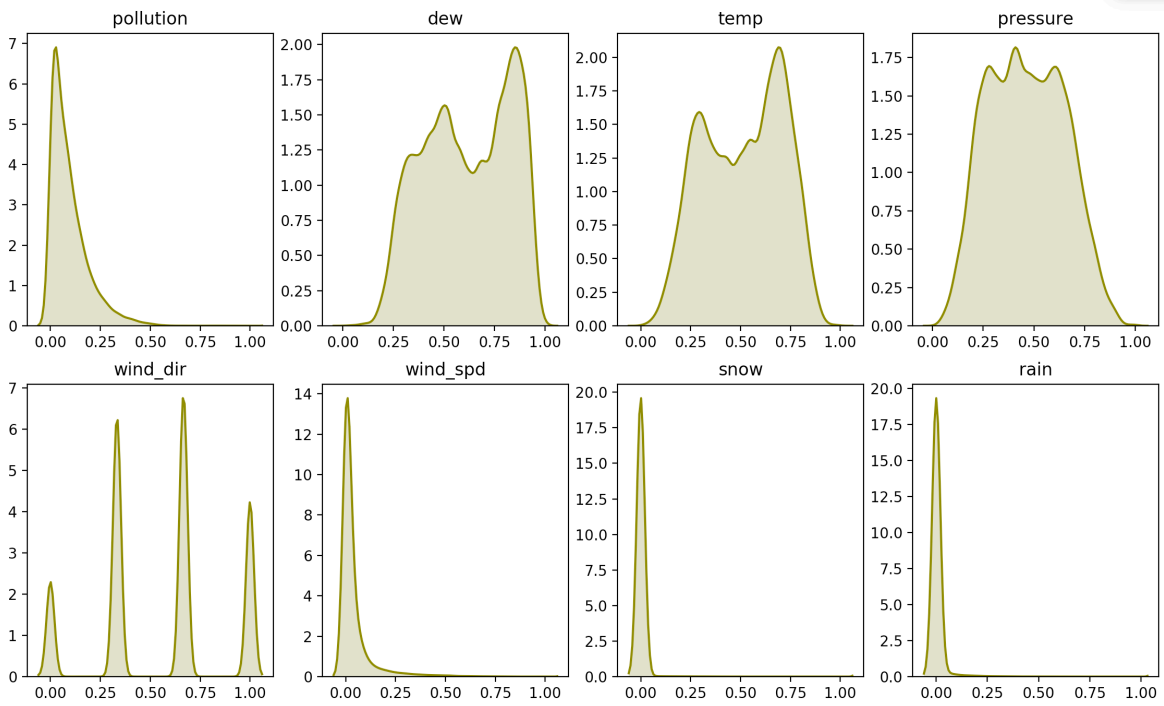
2. روش دیگر استفاده از مقدار صفر یا ثابت برای دادگان از دست رفته می‌باشد. مقدار صفر معمولا انتخاب بهتری است اما ممکن است در برخی از ستون‌ها کاربرد نداشته باشد مثلا انتخاب مقدار صفر برای ویژگی سن شاید بی معنی به نظر برسد. استفاده از این روش سریع و ساده است و برای ویژگی‌های دسته‌بندی نیز می‌تواند خوب عمل کند اما ممکن است باعث به وجود بایاس در نتیجه پایانی شود.

3. روش دیگر استفاده از مقادیر میانگین یا میانه می‌باشد. این مقادیر را باید به کمک دادگان از دست نرفته محاسبه کنیم و در نهایت مقادیر به دست آمده را جایگزین nan ها کنیم. این الگوریتم هم نسبتا ساده است و مثلا انتخاب ویژگی میانگین باعث می‌شود که هر چه قدر هم که حجم از دست رفتن دادگان زیاد باشد، میانگین جا به جا نشود و توزیع آماری ویژگی تغییر زیادی نکند اما در مقابل این روش در ویژگی‌ها و ستون‌های مربوط به categorical قابل استفاده نمی‌باشد و دادگان آن ستون باید عددی باشد تا این معیارها قابل محاسبه باشد. در دادگان بزرگ می‌تواند خیلی دقیق نباشد و ارتباط بین ویژگی‌های مختلف را نمی‌تواند لحاظ کند.

4 . روش دیگر knn است. در این روش از شباهت دیگر ویژگی‌ها برای پیش‌بینی ویژگی از دست رفته استفاده می‌شود. با این کار ارتباط بین ستون‌ها و ویژگی‌ها را نیز در نظر می‌گیریم و آن‌ها را مانند روش های گذشته نادیده نمی‌گیریم. در این روش k نزدیک‌ترین همسایه از بین دادگان سالم برای هر کدام یک از دادگان از دست رفته در نظر گرفته می‌شود و سپس از بین مقدار آن‌ها میانگین گرفته می‌شود. بسته به نوع دیتا و شرایط ممکن است نتیجه خیلی بهتر از روش‌های گذشته به دست بیاید اما این روش سخت و هزینه‌بر می‌باشد چراکه مجبور است مجموعه تمام دادگان از دست نرفته برای آن ستون را همواره نگاه دارد تا بتواند فواصل را محاسبه کند. همچنین برخلاف SVM نسبت به outlier ها خیلی حساس می‌باشد و ممکن است خطای زیادی را وارد کند.

5 . روش دیگر استفاده از EM می‌باشد. در این روش فرض می‌کنیم که توزیع دادگان شبیه به نرمال است و سعی می‌شود که با کمک تخمین پارامترهای توزیع نرمال و چیدن آن‌ها کنار یکدیگر، این توزیع و دادگان از دست رفته را به خوبی تخمین بزند. این روش نیز زمان گیر است و پیاده سازی و اجرای هزینه‌بری دارد اما با توجه به توزیع دادگان‌مان فکر می‌کنیم که روی دادگان ما بتواند به خوبی جواب بدهد.

سوال ۴



شکل ۱-۲: نمودار توزیع در ستون‌های مختلف

شکل ۱-۲ نمودارهای توزیع را در هر یک از ستون‌ها به تفکیک نشان می‌دهد و با توجه به این نمودارها و شکل توزیع هر یک از ستون‌ها به نظر می‌رسد که روش EM بتواند روش مناسبی برای ما باشد. در کنار این روش نیز روش‌های تخمین به کمک میانه و knn با k برابر با ۵ نیز بررسی شده که هر ۳ روش‌های قدرتمند و مناسبی به نظر می‌رسند.

از بین بردن دادگان و تخمین مجدد آن‌ها را همانطور که در سوال خواسته شده است روی ۱۲۰۰۰ داده اول که دادگان آموزش ما هستند انجام دادیم و شاید اگر از بقیه دادگان استفاده می‌کردیم به جهت در دست داشتن دادگان بیشتر، جواب بهتری نیز داشتیم.

سوال ۵

برای محاسبه MSE برای دادگان پیش‌بینی شده به ۲ صورت می‌توانیم عمل کنیم یکی این که مقدار MSE را برای کل ستون‌ها تنها دادگان حذف شده و از دست رفته را در نظر بگیریم که در این حالت برای ۸۰ درصد دادگان مقدار این خطا صفر است و نمی‌تواند معیار خوبی باشد. اما برای مقایسه این مقدار نیز آورده شده است. روش دیگر این است که مقدار MSE را در هر ستون تنها برای خانه‌هایی که در دیتای اصلی حذف شده‌اند و الگوریتم ما آن را پیش‌بینی کرده، محاسبه کنیم. مقدار MSE محاسبه شده در این حالت بیشتر است اما صرفا روش‌های مختلف در محاسبه MSE می‌باشد.

مقدار محاسبه شده را نیز برای ۳ روش EM (our predictor) که روش پیاده سازی شده در این قسمت می‌باشد و روش knn (knn predictor) با k=5 که به کمک کتابخانه و به جهت مقایسه آورده شده و روش میانگین (mean predictor) که ساده‌ترین روش پیاده سازی شده است و به جهت مقایسه می‌باشد، انجام دادیم. نتایج برای ۳ روش مختلف پیش‌بینی داده‌ها و با ۲ روش مختلف محاسبه MSE به صورت زیر می‌باشد.

```
our predictor :::
> for all data:
0 0.0013614985
1 0.002400167
2 0.002221869
3 0.0024619987
4 0.017059827
5 0.0017274466
6 0.0003047686
7 0.0002650022
> for missing data:
0 0.00672345
1 0.01221459
2 0.01105866
3 0.012223412
4 0.08466416
5 0.008630041
6 0.0015175203
7 0.0013129753
```

```
knn predictor :::
> for all data:
0 0.0018431656
1 0.007401277
2 0.008233388
3 0.0064081573
4 0.02167889
5 0.0021997502
6 0.0003061226
7 0.0009021477
> for missing data:
0 0.009102053
1 0.037665524
2 0.040979117
3 0.03181543
4 0.10758754
5 0.010989592
6 0.0015242617
7 0.004469766
```

```
mean predictor :::
> for all data:
0 0.0017145145
1 0.008354369
2 0.008015939
3 0.0066743023
4 0.018866792
5 0.0020125182
6 0.00030827182
7 0.0002720229
> for missing data:
0 0.008466738
1 0.042515874
2 0.039896835
3 0.033136792
4 0.09363172
5 0.010054213
6 0.0015349635
7 0.0013477602
```

طبق محاسبات انجام شده و نشان داده شده به نظر می‌رسد که روش EM بهترین روش پیاده سازی شده بوده و توانسته خطا را به شکل مناسبی کاهش دهد. اما بعد از این روش به نظر می‌رسد که در برخی از ستون‌ها عملکرد تخمین به کمک میانگین بهتر از knn بوده و این بستگی به داده‌ها و توزیع آن‌ها دارد.

سوال ۶

در این قسمت از دادگان به دست آمده استفاده کرده و مانند قسمت اول پروژه مدل با استفاده از GRU, LSTM میزان آلودگی روزانه را پیش‌بینی می‌کنیم. تعداد ایپاک‌های آموزش مانند قسمت قبلی برابر با ۲۰ قرار می‌دهیم و همینطور برای مدل‌های گزارش شده در این قسمت تابع بهینه‌سازی adam و تابع هزینه (خطا) MAE را در نظر می‌گیریم.

شکل زیر خروجی یادگیری را به ازای هر کدام از GRU, LSTM را نشان بعد از ۲۰ ایپاک، با دادگان بازسازی شده (دادگان جدید) و دادگان قدیمی(بدون از دست رفتن و بازسازی شدن) نشان می‌دهد.

GRU with new_data	LSTM with new_data
Epoch 10/20 - 6s - loss: 0.0024 - val_loss: 0.0021 Epoch 11/20 - 6s - loss: 0.0024 - val_loss: 0.0021 Epoch 12/20 - 7s - loss: 0.0024 - val_loss: 0.0021 Epoch 13/20 - 8s - loss: 0.0024 - val_loss: 0.0021 Epoch 14/20 - 7s - loss: 0.0024 - val_loss: 0.0020 Epoch 15/20 - 7s - loss: 0.0023 - val_loss: 0.0020 Epoch 16/20 - 7s - loss: 0.0023 - val_loss: 0.0020 Epoch 17/20 - 7s - loss: 0.0023 - val_loss: 0.0020 Epoch 18/20 - 8s - loss: 0.0023 - val_loss: 0.0020 Epoch 19/20 - 9s - loss: 0.0023 - val_loss: 0.0020 Epoch 20/20 - 7s - loss: 0.0023 - val_loss: 0.0020 Training Duration: 158.81868505477905	Epoch 10/20 - 4s - loss: 0.0024 - val_loss: 0.0019 Epoch 11/20 - 4s - loss: 0.0024 - val_loss: 0.0019 Epoch 12/20 - 4s - loss: 0.0023 - val_loss: 0.0019 Epoch 13/20 - 4s - loss: 0.0023 - val_loss: 0.0019 Epoch 14/20 - 4s - loss: 0.0023 - val_loss: 0.0018 Epoch 15/20 - 4s - loss: 0.0023 - val_loss: 0.0018 Epoch 16/20 - 4s - loss: 0.0023 - val_loss: 0.0018 Epoch 17/20 - 4s - loss: 0.0023 - val_loss: 0.0018 Epoch 18/20 - 4s - loss: 0.0023 - val_loss: 0.0018 Epoch 19/20 - 4s - loss: 0.0023 - val_loss: 0.0018 Epoch 20/20 - 4s - loss: 0.0023 - val_loss: 0.0018 Training Duration: 83.20216488838196
GRU with old data	**LSTM with old_data**
Epoch 10/20 - 5s - loss: 9.4706e-04 - val_loss: 0.0010 Epoch 11/20 - 5s - loss: 9.3801e-04 - val_loss: 0.0010 Epoch 12/20 - 5s - loss: 9.2855e-04 - val_loss: 0.0010 Epoch 13/20 - 5s - loss: 9.1822e-04 - val_loss: 0.0010 Epoch 14/20 - 5s - loss: 9.0689e-04 - val_loss: 0.0011 Epoch 15/20 - 5s - loss: 8.9495e-04 - val_loss: 0.0011 Epoch 16/20 - 5s - loss: 8.8293e-04 - val_loss: 0.0011 Epoch 17/20 - 5s - loss: 8.7121e-04 - val_loss: 0.0011 Epoch 18/20 - 5s - loss: 8.5996e-04 - val_loss: 0.0011 Epoch 19/20 - 5s - loss: 8.4920e-04 - val_loss: 0.0012 Epoch 20/20 - 5s - loss: 8.3885e-04 - val_loss: 0.0012 Training Duration: 88.0952639579773	Epoch 10/20 - 3s - loss: 9.4779e-04 - val_loss: 0.0012 Epoch 11/20 - 3s - loss: 9.3958e-04 - val_loss: 0.0012 Epoch 12/20 - 3s - loss: 9.3108e-04 - val_loss: 0.0012 Epoch 13/20 - 3s - loss: 9.2226e-04 - val_loss: 0.0012 Epoch 14/20 - 3s - loss: 9.1319e-04 - val_loss: 0.0011 Epoch 15/20 - 3s - loss: 9.0399e-04 - val_loss: 0.0011 Epoch 16/20 - 3s - loss: 8.9479e-04 - val_loss: 0.0011 Epoch 17/20 - 3s - loss: 8.8572e-04 - val_loss: 0.0011 Epoch 18/20 - 3s - loss: 8.7690e-04 - val_loss: 0.0011 Epoch 19/20 - 3s - loss: 8.6837e-04 - val_loss: 0.0011 Epoch 20/20 - 3s - loss: 8.6017e-04 - val_loss: 0.0011 Training Duration: 60.601654052734375

همانطور که در شکل نیز مشخص است، نتیجه و دقت در حالتی که دادگان صحیح را در اختیار داشتیم بهتر است و از دست رفتن داده باعث کاهش بازده و دقت مدل می‌شود. حتی وقتی که با روش‌هایی با دقت مناسب سعی کنیم تا دادگان از دست رفته را بازسازی کنیم. بنابرین خطای تخمین دادگان نیز به خطای محاسبات ما اضافه میشود و دقت مدل را پایین می‌آورد.

نمودارها مطابق با استاندارد و فرما قسمت قبلی، در این قسمت نیز ساخته شده و در پوشه‌ی آپلود شده در پوشه‌ای به نام Generated files قرار دارد.