Untitled3

November 29, 2018

```
In [1]: # -*- coding: utf-8 -*-
        """
        Created on Mon Nov 26 09:23:45 2018

        @author: aza8223
        """

        """Project2"""


        ###############################################################################

        """Importing important libraries"""

        import numpy as np
        import pandas as pd
        from keras import layers
        from keras import optimizers
        import math
        import matplotlib.pyplot as plt
        from keras.optimizers import RMSprop
```

```
C:\Users\aza8223\AppData\Local\Continuum\anaconda3\lib\site-packages\h5py\__init__.py:36: Future
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
In [3]: """Preparing data"""
        data =[]
        if data:
            del data

        total_data = pd.read_csv("C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfer
        data = total_data[:,1:]

        """Check if there is nan (missing) data and replace them with their next data:"""
        """Here i have used while loop for the case when oreceding samples all nan replacement
        keeps going until get reasonable neighbor value"""
```

```python
data = pd.DataFrame(data=data)
while 1:
    for j, kays in enumerate(data.loc[0,:]):
        for i, kay in enumerate(data.loc[:,0]):
            if math.isnan(data.loc[i,j]):
                data.loc[i,j]=data.loc[i+1,j]
                print("sample ", i, "feature", j, " was missing and replaced by its next
    if not data.isnull().any().any():
        break
data = np.asarray(data).astype('float32')

"""Change true and false to 1 and 0"""
for j, rain in enumerate(data[:,3]):
    if data[j, 3]==True:
        data[j,3]=1
    else:
        data[j,3]=0

data = data[:,:3] #If it rains or not is not important feature for the determination
#of amount of rain.
data = np.asarray(data).astype('float32')

"""Creating descriptive and target features"""
num_data = len(data)
output_size = 7 #Days to be predicted. They are fixed
input_size = 1 #Sequence of days to be descriptive feature. You can modify it
# as given in the problem: 1 day, 7 days, 14 days, 1 months.


"""Create data descriptime sequential features with the shape of sample*times*features""
data_feat = np.zeros((num_data-(output_size+input_size),input_size,len(data[0])))
data_label = np.zeros((num_data-(output_size+input_size),output_size))
for i in range(num_data - (output_size+input_size)):
    data_feat[i] = data[i:i+input_size]
    data_label[i] = data[i+input_size:i+input_size+output_size,0]

"""Seperating data into dry and wet days"""
"""
To do so, i calculated mean of each output (7days that to be predicted)
then i compared that output with mean of all labels, and thus i devided my data
for dry week and wet week
"""
mean_each_output = data_label[:,:].mean(axis=1)
mean_all_data = np.nanmean(mean_each_output)

positive_data = []
positive_label = []
negative_data = []
```

```
negative_label = []

for i in range(len(data_label)):
    if mean_each_output[i]<=mean_all_data:
        negative_data.append(data_feat[i])
        negative_label.append(data_label[i])
    else:
        positive_data.append(data_feat[i])
        positive_label.append(data_label[i])

positive_data = np.asarray(positive_data).astype('float32')
positive_data_part1 = positive_data[:round(len(positive_data)/3)]
positive_data_part2 = positive_data[round(len(positive_data)/3):round(2*len(positive_dat
positive_data_part3 = positive_data[round(2*len(positive_data)/3):]

positive_label = np.asarray(positive_label).astype('float32')
positive_label_part1 = positive_label[:round(len(positive_data)/3)]
positive_label_part2 = positive_label[round(len(positive_data)/3):round(2*len(positive_d
positive_label_part3 = positive_label[round(2*len(positive_data)/3):]

negative_data = np.asarray(negative_data).astype('float32')
negative_data_part1 = negative_data[:round(len(negative_data)/3)]
negative_data_part2 = negative_data[round(len(negative_data)/3):round(2*len(negative_dat
negative_data_part3 = negative_data[round(2*len(negative_data)/3):]

negative_label = np.asarray(negative_label).astype('float32')
negative_label_part1 = negative_label[:round(len(negative_data)/3)]
negative_label_part2 = negative_label[round(len(negative_data)/3):round(2*len(negative_d
negative_label_part3 = negative_label[round(2*len(negative_data)/3):]

"""Create training, test, validation data and labels using 1/3 partion of both
negative and positive sets:"""

import itertools
training_data = []
for item in itertools.chain(positive_data_part1,negative_data_part1):
    training_data.append(item)

training_labels = []
for item in itertools.chain(positive_label_part1,negative_label_part1):
    training_labels.append(item)

test_data = []
for item in itertools.chain(positive_data_part2,negative_data_part2):
    test_data.append(item)

test_labels = []
for item in itertools.chain(positive_label_part2,negative_label_part2):
```

```python
            test_labels.append(item)

        val_data = []
        for item in itertools.chain(positive_data_part3,negative_data_part3):
            val_data.append(item)

        val_labels = []
        for item in itertools.chain(positive_label_part3,negative_label_part3):
            val_labels.append(item)


        training_data = np.asarray(training_data).astype('float32')
        training_labels = np.asarray(training_labels).astype('float32')

        test_data = np.asarray(test_data).astype('float32')
        test_labels = np.asarray(test_labels).astype('float32')

        val_data = np.asarray(val_data).astype('float32')
        val_labels = np.asarray(val_labels).astype('float32')

        """Shuffle data and labels:"""

        from random import shuffle

        ind_list = [i for i in range(len(training_data))]
        shuffle(ind_list)
        training_data  = training_data[ind_list, :, :]
        training_labels = training_labels[ind_list, :]

        ind_list = [i for i in range(len(val_data))]
        shuffle(ind_list)
        val_data  = val_data[ind_list, :, :]
        val_labels = val_labels[ind_list, :]

        ind_list = [i for i in range(len(test_data))]
        shuffle(ind_list)
        test_data  = test_data[ind_list, :, :]
        test_labels = test_labels[ind_list, :]
```

```
sample  18415 feature 0  was missing and replaced by its next samnple
sample  18416 feature 0  was missing and replaced by its next samnple
sample  21067 feature 0  was missing and replaced by its next samnple
sample  18415 feature 3  was missing and replaced by its next samnple
sample  18416 feature 3  was missing and replaced by its next samnple
sample  21067 feature 3  was missing and replaced by its next samnple
sample  18415 feature 0  was missing and replaced by its next samnple
sample  18415 feature 3  was missing and replaced by its next samnple
```

In [4]:
```python
#Normalize your all data based on mean std of your training data and training labels:
mean = training_data[:,:,:].mean(axis=0)
training_data[:,:,:] -= mean
std = np.std(training_data[:,:,:],axis=0)
training_data[:,:,:] /= std

val_data[:,:,:] -= mean
val_data[:,:,:] /= std

test_data[:,:,:] -= mean
test_data[:,:,:] /= std

mean = training_labels[:,:].mean(axis=0)
training_labels[:,:] -= mean
std = np.std(training_labels[:,:],axis=0)
training_labels[:,:] /= std

val_labels[:,:] -= mean
val_labels[:,:] /= std

test_labels[:,:] -= mean
test_labels[:,:] /= std
```

In [6]:
```python
"""Base case for each day and mean of mae"""
"""Here I took average of previous days as my predictor for the each day of the
next week. Therefore I have calculated mae for each day of the next week. To
be able to compare this mae with my models, since I predict them all together, and
therefore I have 1 mae for  model, I took average of all those mae in this base
model for each day and took mean of them. I will use this mean of mae of the days of
the next week to compare it with my models. However, at the last model, where
I use multiple output DAG model, I used mae of each day in my base model to compare
it with the loss of each day in that last model:"""
preds = np.mean(val_data[:, :, 0], axis=1)
day = np.zeros((val_labels.shape[1], val_labels.shape[0]))
mae_base1 = np.zeros((val_labels.shape[1],))
for i,j in enumerate(np.transpose(val_labels)):
    day[i] = val_labels[:,i]
    mae_base1[i] = np.nanmean(np.abs(preds - day[i]))
    print('normalized MAE of base model for day ', i+1, " is ", mae_base1[i])
    print('unnormalized MAE of base model for day ', i+1, " is ", mae_base1[i]*std[0])
mae_base_mean = mae_base1.mean()
print('mean of normalized MAE of base model of week ', " is ", mae_base_mean)
print('mean of unnormalized MAE of base model of week ', " is ", mae_base_mean*std[0])



"""Base model2: This is just my own opinion, but I ll not compare my models with this mo
In the following base model2, I choose my target  not as each dy of next week but averag
```

```
of them. So I found mae between average precipitation of previous days as predictor of
average precipitation. This result showed 10 percent of mae. Compared to the base
model given above it is higher but it doesnt show that this is good predictor of
each day of next week, but it is good model to predict average precipitation of the
next week:"""
preds = np.mean(val_data[:, :, 0], axis=1)
week_data = np.mean(val_labels[:,:],axis=1)
mae_base2 = np.nanmean(np.abs(preds - week_data))
print('normalized MAE of base2 model is ', mae_base2)
print('unnormalized MAE of base2 model is ', mae_base2*std[0])
```

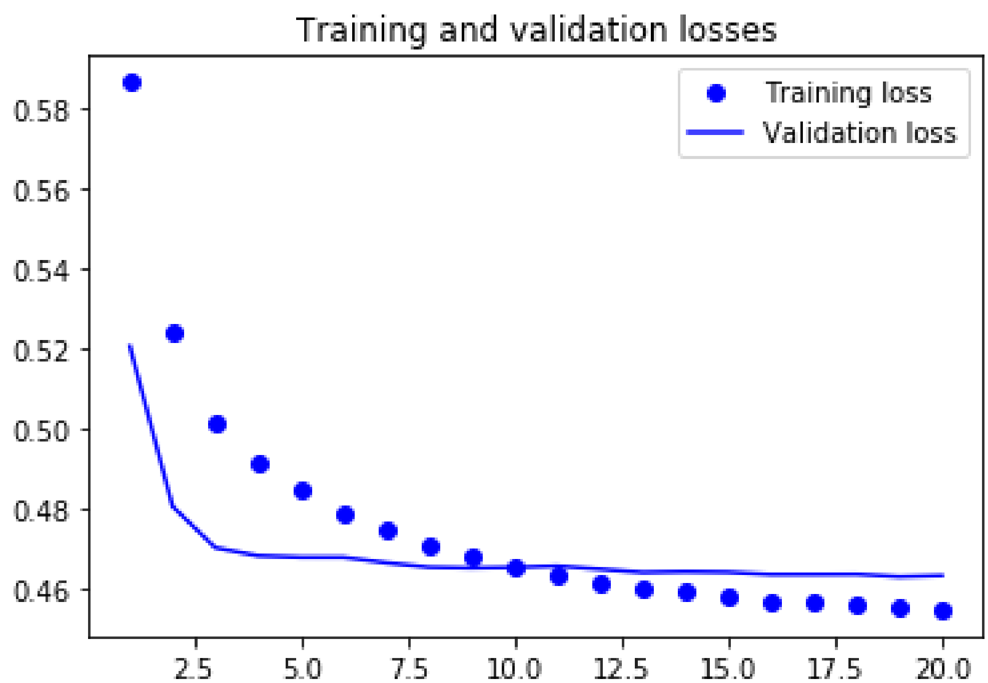
```
normalized MAE of base model for day  1  is  0.5757574691956336
unnormalized MAE of base model for day  1  is  0.13426845135385884
normalized MAE of base model for day  2  is  0.6615618414999005
unnormalized MAE of base model for day  2  is  0.15427830064817905
normalized MAE of base model for day  3  is  0.6920572701585189
unnormalized MAE of base model for day  3  is  0.16138993045488417
normalized MAE of base model for day  4  is  0.7040279924798217
unnormalized MAE of base model for day  4  is  0.16418154052277245
normalized MAE of base model for day  5  is  0.715426965308281
unnormalized MAE of base model for day  5  is  0.16683981681198876
normalized MAE of base model for day  6  is  0.7177373140182679
unnormalized MAE of base model for day  6  is  0.16737859739230967
normalized MAE of base model for day  7  is  0.7227651688117348
unnormalized MAE of base model for day  7  is  0.16855110893209743
mean of normalized MAE of base model of week   is  0.6841905744960226
mean of unnormalized MAE of base model of week   is  0.15955539230229862
normalized MAE of base2 model is  0.585915
unnormalized MAE of base2 model is  0.13663723
```

In [7]:
```
"""1: Training and evaluating a densely connected model"""
"""I have tried different kind of architectures hidden units etc, but found this
useful since it does not overfit and I got lower loss - 0.1015 (unnormilized)"""
from keras.models import Sequential
model = Sequential()
model.add(layers.Flatten(input_shape=(input_size, training_data.shape[-1])))
model.add(layers.Dense(64,activation='tanh'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(32,activation='tanh'))
model.add(layers.Dense(output_size,activation='tanh'))


"""COMPILE YOUR MODEL"""
model.compile(optimizer=optimizers.RMSprop(lr=1e-4), loss='mae')


"""TRAINING YOUR MODEL"""
```

```python
epoch_size = 20
batch_size = 32
history = model.fit(training_data,
                    training_labels,
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (val_data, val_labels))


"""Plotting results"""
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses')
plt.legend()
plt.show()


"""PREDICTION - TESTING DATA"""
test_loss = model.evaluate(test_data, test_labels)
print('normalized test_loss:', test_loss)
print('unnormalized test_loss:', test_loss*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Proj
```

```
Train on 8514 samples, validate on 8514 samples
Epoch 1/20
8514/8514 [==============================] - 0s 59us/step - loss: 0.5864 - val_loss: 0.5206
Epoch 2/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.5239 - val_loss: 0.4807
Epoch 3/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.5016 - val_loss: 0.4704
Epoch 4/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4915 - val_loss: 0.4683
Epoch 5/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4848 - val_loss: 0.4680
Epoch 6/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4791 - val_loss: 0.4680
Epoch 7/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4751 - val_loss: 0.4666
Epoch 8/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.4712 - val_loss: 0.4655
Epoch 9/20
8514/8514 [==============================] - 0s 32us/step - loss: 0.4682 - val_loss: 0.4653
```

```
Epoch 10/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.4658 - val_loss: 0.4655
Epoch 11/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.4638 - val_loss: 0.4658
Epoch 12/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4619 - val_loss: 0.4650
Epoch 13/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.4605 - val_loss: 0.4642
Epoch 14/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.4595 - val_loss: 0.4644
Epoch 15/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.4584 - val_loss: 0.4643
Epoch 16/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4572 - val_loss: 0.4637
Epoch 17/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4567 - val_loss: 0.4637
Epoch 18/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4559 - val_loss: 0.4638
Epoch 19/20
8514/8514 [==============================] - 0s 30us/step - loss: 0.4555 - val_loss: 0.4631
Epoch 20/20
8514/8514 [==============================] - 0s 31us/step - loss: 0.4550 - val_loss: 0.4634
```



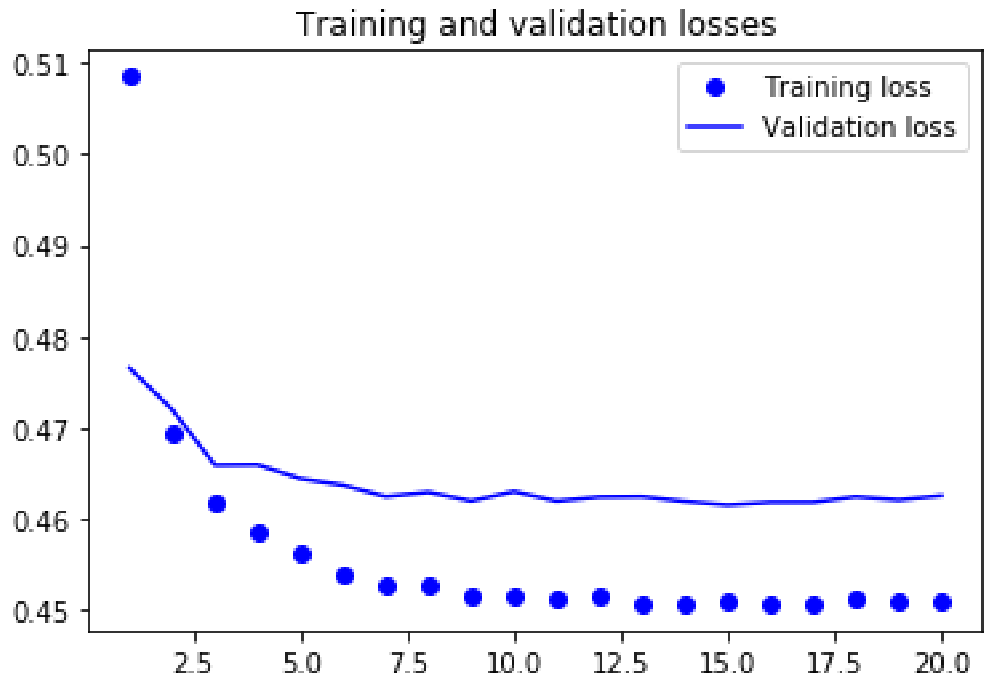

```
8515/8515 [==============================] - 0s 8us/step
normalized test_loss: 0.43413716846539985
```

unnormalized test_loss: 0.10124215212776021

```
In [8]: """2a: RNN"""
        """I have tried different dense model architecture but best one was this
        which is 2nd dense with 32 hidden units"""
        """Dropout also helped to improve model. I kept playing with dropouts and
        additional dropout layer until i get least loss"""
        """But when i rerun model it gives me different kind of test_loss values
        even thoough i train the same model( between 18 and 48). that means our data is very uns
        therefore stochastig gradient method catch different local minimum each time"""
        model = Sequential()
        model.add(layers.GRU(32,
                             dropout=0.2,
                             recurrent_dropout=0.2,
                             input_shape=(None, training_data.shape[-1])))
        model.add(layers.Dense(32,activation='relu'))
        model.add(layers.Dropout(0.5))
        model.add(layers.Dense(output_size,activation='tanh'))

        """COMPILE YOUR MODEL"""
        model.compile(optimizer=RMSprop(), loss='mae')


        """TRAINING YOUR MODEL"""
        epoch_size = 20
        batch_size = 32
        history = model.fit(training_data,
                            training_labels,
                            epochs=epoch_size,
                            batch_size=batch_size,
                            validation_data = (val_data, val_labels))


        """Plotting results"""
        loss = history.history['loss']
        val_loss = history.history['val_loss']
        epochs = range(1, len(loss) + 1)
        plt.figure()
        plt.plot(epochs, loss, 'bo', label='Training loss')
        plt.plot(epochs, val_loss, 'b', label='Validation loss')
        plt.title('Training and validation losses')
        plt.legend()
        plt.show()


        """PREDICTION - TESTING DATA"""
        test_loss = model.evaluate(test_data, test_labels)
```

```
print('normalized test_loss:', test_loss)
print('unnormalized test_loss:', test_loss*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Proj
```

```
Train on 8514 samples, validate on 8514 samples
Epoch 1/20
8514/8514 [==============================] - 1s 149us/step - loss: 0.5085 - val_loss: 0.4766
Epoch 2/20
8514/8514 [==============================] - 0s 49us/step - loss: 0.4694 - val_loss: 0.4720
Epoch 3/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4618 - val_loss: 0.4659
Epoch 4/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4585 - val_loss: 0.4660
Epoch 5/20
8514/8514 [==============================] - 0s 49us/step - loss: 0.4562 - val_loss: 0.4644
Epoch 6/20
8514/8514 [==============================] - 0s 50us/step - loss: 0.4540 - val_loss: 0.4637
Epoch 7/20
8514/8514 [==============================] - 0s 50us/step - loss: 0.4528 - val_loss: 0.4625
Epoch 8/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4526 - val_loss: 0.4629
Epoch 9/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4516 - val_loss: 0.4620
Epoch 10/20
8514/8514 [==============================] - 0s 49us/step - loss: 0.4515 - val_loss: 0.4630
Epoch 11/20
8514/8514 [==============================] - 0s 49us/step - loss: 0.4512 - val_loss: 0.4620
Epoch 12/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4516 - val_loss: 0.4624
Epoch 13/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4508 - val_loss: 0.4625
Epoch 14/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4508 - val_loss: 0.4619
Epoch 15/20
8514/8514 [==============================] - 0s 50us/step - loss: 0.4511 - val_loss: 0.4615
Epoch 16/20
8514/8514 [==============================] - 0s 49us/step - loss: 0.4508 - val_loss: 0.4618
Epoch 17/20
8514/8514 [==============================] - 0s 52us/step - loss: 0.4507 - val_loss: 0.4618
Epoch 18/20
8514/8514 [==============================] - 0s 49us/step - loss: 0.4512 - val_loss: 0.4625
Epoch 19/20
8514/8514 [==============================] - 0s 49us/step - loss: 0.4511 - val_loss: 0.4621
Epoch 20/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4508 - val_loss: 0.4626
```



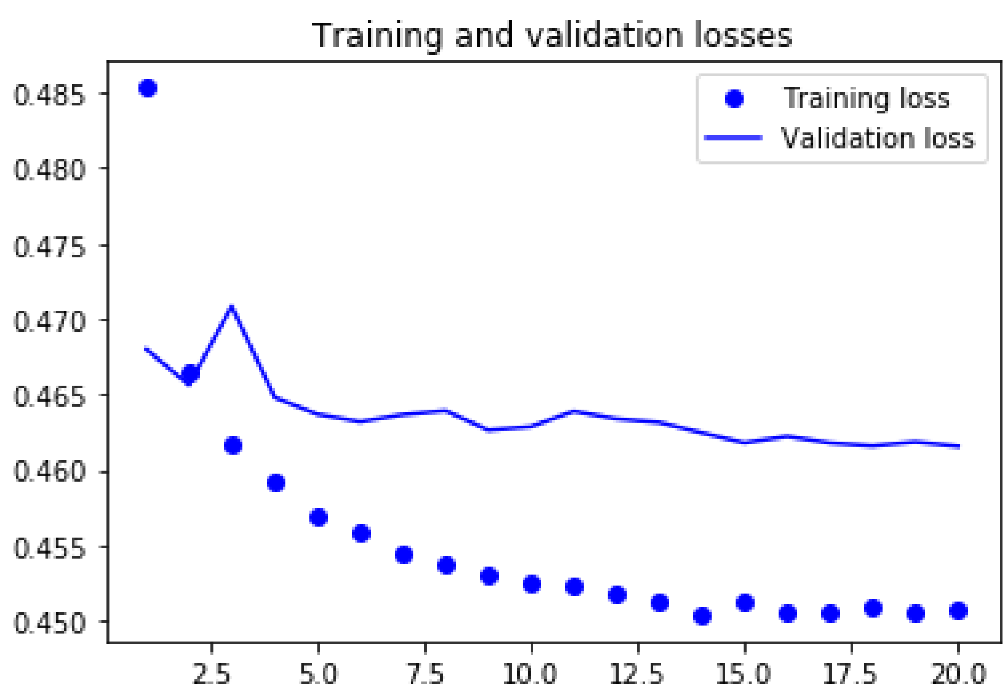

```
8515/8515 [==============================] - 0s 14us/step
normalized test_loss: 0.4321078981642015
unnormalized test_loss: 0.10076891991576488
```

In [9]:
```python
"""2b: Training and evaluating a dropout-regularized, stacked GRU model"""


from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop
model = Sequential()
model.add(layers.GRU(32, activation='relu',
                     dropout=0.2,
                     recurrent_dropout=0.2,
                     return_sequences=True,
                     input_shape=(None, training_data.shape[-1])))
model.add(layers.GRU(64, activation='relu',
                     dropout=0.2,
                     recurrent_dropout=0.25))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(output_size, activation='tanh'))

"""COMPILE YOUR MODEL"""
```

```
model.compile(optimizer=RMSprop(), loss='mae')


"""TRAINING YOUR MODEL"""
epoch_size = 20
batch_size = 32
history = model.fit(training_data,
                    training_labels,
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (val_data, val_labels))


"""Plotting results"""
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses')
plt.legend()
plt.show()


"""PREDICTION - TESTING DATA"""
test_loss = model.evaluate(test_data, test_labels)
print('normalized test_loss:', test_loss)
print('unnormalized test_loss:', test_loss*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Proj
```

```
Train on 8514 samples, validate on 8514 samples
Epoch 1/20
8514/8514 [==============================] - 2s 238us/step - loss: 0.4853 - val_loss: 0.4680
Epoch 2/20
8514/8514 [==============================] - 1s 84us/step - loss: 0.4664 - val_loss: 0.4656
Epoch 3/20
8514/8514 [==============================] - 1s 82us/step - loss: 0.4617 - val_loss: 0.4708
Epoch 4/20
8514/8514 [==============================] - 1s 86us/step - loss: 0.4592 - val_loss: 0.4648
Epoch 5/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4570 - val_loss: 0.4637
Epoch 6/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4558 - val_loss: 0.4632
Epoch 7/20
8514/8514 [==============================] - 1s 84us/step - loss: 0.4545 - val_loss: 0.4637
```

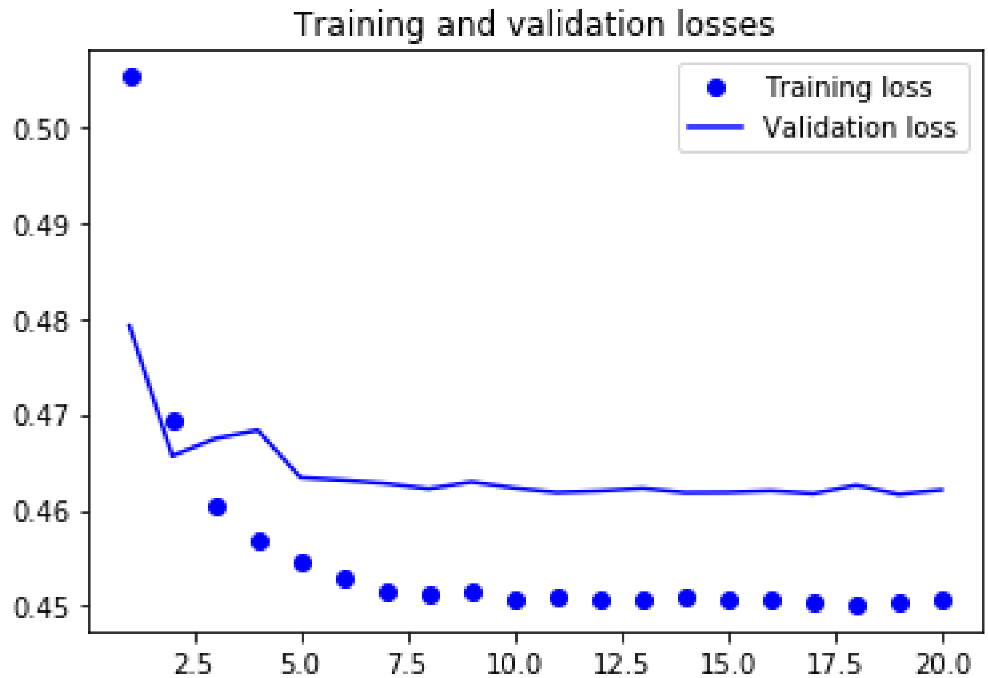
```
Epoch 8/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4537 - val_loss: 0.4639
Epoch 9/20
8514/8514 [==============================] - 1s 82us/step - loss: 0.4530 - val_loss: 0.4626
Epoch 10/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4525 - val_loss: 0.4629
Epoch 11/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4524 - val_loss: 0.4639
Epoch 12/20
8514/8514 [==============================] - 1s 84us/step - loss: 0.4518 - val_loss: 0.4634
Epoch 13/20
8514/8514 [==============================] - 1s 85us/step - loss: 0.4513 - val_loss: 0.4632
Epoch 14/20
8514/8514 [==============================] - 1s 82us/step - loss: 0.4504 - val_loss: 0.4625
Epoch 15/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4513 - val_loss: 0.4618
Epoch 16/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4506 - val_loss: 0.4622
Epoch 17/20
8514/8514 [==============================] - 1s 81us/step - loss: 0.4505 - val_loss: 0.4618
Epoch 18/20
8514/8514 [==============================] - 1s 83us/step - loss: 0.4509 - val_loss: 0.4616
Epoch 19/20
8514/8514 [==============================] - 1s 84us/step - loss: 0.4506 - val_loss: 0.4619
Epoch 20/20
8514/8514 [==============================] - 1s 85us/step - loss: 0.4507 - val_loss: 0.4616
```



```
8515/8515 [==============================] - 0s 17us/step
normalized test_loss: 0.43129316549474744
unnormalized test_loss: 0.10057892169664005
```

```
In [10]: """2c: Bidirectional RNN""" """32"""

         model = Sequential()
         model.add(layers.Bidirectional(layers.LSTM(32)))
         model.add(layers.Dense(32, activation='relu'))
         model.add(layers.Dropout(0.6))
         model.add(layers.Dense(output_size, activation='tanh'))

         """COMPILE YOUR MODEL"""
         model.compile(optimizer=RMSprop(), loss='mae')


         """TRAINING YOUR MODEL"""
         epoch_size = 20
         batch_size = 32
         history = model.fit(training_data,
                             training_labels,
                             epochs=epoch_size,
                             batch_size=batch_size,
                             validation_data = (val_data, val_labels))


         """Plotting results"""
         loss = history.history['loss']
         val_loss = history.history['val_loss']
         epochs = range(1, len(loss) + 1)
         plt.figure()
         plt.plot(epochs, loss, 'bo', label='Training loss')
         plt.plot(epochs, val_loss, 'b', label='Validation loss')
         plt.title('Training and validation losses')
         plt.legend()
         plt.show()


         """PREDICTION - TESTING DATA"""
         test_loss = model.evaluate(test_data, test_labels)
         print('normalized test_loss:', test_loss)
         print('unnormalized test_loss:', test_loss*std[0])

         """Save your model:"""
         model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

```
Train on 8514 samples, validate on 8514 samples
Epoch 1/20
8514/8514 [==============================] - 2s 233us/step - loss: 0.5052 - val_loss: 0.4792
Epoch 2/20
8514/8514 [==============================] - 1s 66us/step - loss: 0.4694 - val_loss: 0.4656
Epoch 3/20
8514/8514 [==============================] - 1s 64us/step - loss: 0.4605 - val_loss: 0.4674
Epoch 4/20
8514/8514 [==============================] - 1s 67us/step - loss: 0.4567 - val_loss: 0.4683
Epoch 5/20
8514/8514 [==============================] - 1s 68us/step - loss: 0.4546 - val_loss: 0.4634
Epoch 6/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4529 - val_loss: 0.4631
Epoch 7/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4515 - val_loss: 0.4628
Epoch 8/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4514 - val_loss: 0.4622
Epoch 9/20
8514/8514 [==============================] - 1s 68us/step - loss: 0.4515 - val_loss: 0.4629
Epoch 10/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4508 - val_loss: 0.4623
Epoch 11/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4509 - val_loss: 0.4618
Epoch 12/20
8514/8514 [==============================] - 1s 68us/step - loss: 0.4508 - val_loss: 0.4620
Epoch 13/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4506 - val_loss: 0.4623
Epoch 14/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4509 - val_loss: 0.4618
Epoch 15/20
8514/8514 [==============================] - 1s 68us/step - loss: 0.4506 - val_loss: 0.4618
Epoch 16/20
8514/8514 [==============================] - 1s 68us/step - loss: 0.4508 - val_loss: 0.4620
Epoch 17/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4505 - val_loss: 0.4617
Epoch 18/20
8514/8514 [==============================] - 1s 68us/step - loss: 0.4502 - val_loss: 0.4626
Epoch 19/20
8514/8514 [==============================] - 1s 69us/step - loss: 0.4504 - val_loss: 0.4616
Epoch 20/20
8514/8514 [==============================] - 1s 67us/step - loss: 0.4507 - val_loss: 0.4621
```



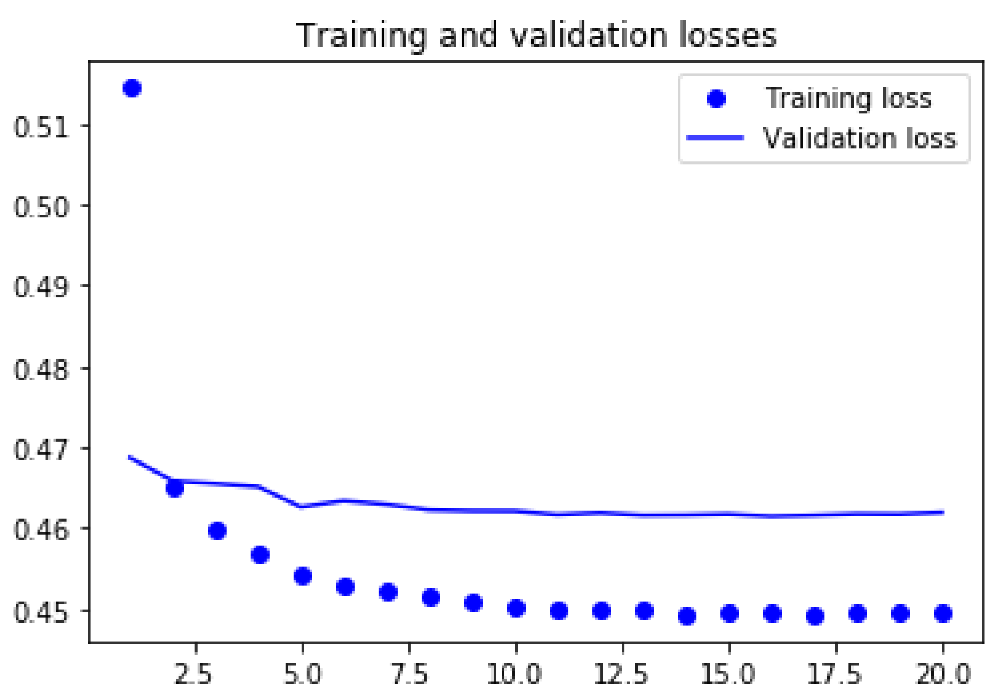

```
8515/8515 [==============================] - 0s 15us/step
normalized test_loss: 0.4316862609410244
unnormalized test_loss: 0.1006705927901642
```

In [11]:
```
"""2d: Training and evaluating an LSTM using reversed sequences  10.23"""

"""First reverse days (sequentions or times) in your training and validation data,
but not labels"""
"""tanh seems better choice even for hidden layers"""
x_train = [x[::-1] for x in training_data] #It will reverse days (times)
x_test = [x[::-1] for x in test_data]
x_train = np.asarray(x_train).astype('float32')
x_test = np.asarray(x_test).astype('float32')

x_val = [x[::-1] for x in val_data] #It will reverse days (times)
x_val = np.asarray(x_val).astype('float32')


model = Sequential()
model.add(layers.LSTM(32))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(output_size, activation='tanh'))

"""COMPILE YOUR MODEL"""
```

```python
model.compile(optimizer=RMSprop(), loss='mae')


"""TRAINING YOUR MODEL"""
epoch_size = 20
batch_size = 32
history = model.fit(x_train,
                    training_labels,
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (x_val, val_labels))


"""Plotting results"""
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses')
plt.legend()
plt.show()


"""PREDICTION - TESTING DATA"""
test_loss = model.evaluate(test_data, test_labels)
print('normalized test_loss:', test_loss)
print('unnormalized test_loss:', test_loss*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

```
Train on 8514 samples, validate on 8514 samples
Epoch 1/20
8514/8514 [==============================] - 1s 164us/step - loss: 0.5144 - val_loss: 0.4687
Epoch 2/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4651 - val_loss: 0.4658
Epoch 3/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4598 - val_loss: 0.4655
Epoch 4/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4569 - val_loss: 0.4651
Epoch 5/20
8514/8514 [==============================] - 0s 54us/step - loss: 0.4543 - val_loss: 0.4626
Epoch 6/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4529 - val_loss: 0.4634
Epoch 7/20
8514/8514 [==============================] - 0s 52us/step - loss: 0.4523 - val_loss: 0.4629
```

```
Epoch 8/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4515 - val_loss: 0.4622
Epoch 9/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4510 - val_loss: 0.4621
Epoch 10/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4502 - val_loss: 0.4621
Epoch 11/20
8514/8514 [==============================] - 0s 52us/step - loss: 0.4498 - val_loss: 0.4616
Epoch 12/20
8514/8514 [==============================] - 0s 55us/step - loss: 0.4498 - val_loss: 0.4619
Epoch 13/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4499 - val_loss: 0.4616
Epoch 14/20
8514/8514 [==============================] - 0s 52us/step - loss: 0.4493 - val_loss: 0.4616
Epoch 15/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4497 - val_loss: 0.4617
Epoch 16/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4496 - val_loss: 0.4614
Epoch 17/20
8514/8514 [==============================] - 0s 51us/step - loss: 0.4493 - val_loss: 0.4616
Epoch 18/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4497 - val_loss: 0.4617
Epoch 19/20
8514/8514 [==============================] - 0s 52us/step - loss: 0.4496 - val_loss: 0.4617
Epoch 20/20
8514/8514 [==============================] - 0s 53us/step - loss: 0.4496 - val_loss: 0.4619
```



```
8515/8515 [==============================] - 0s 13us/step
normalized test_loss: 0.43246868284604023
unnormalized test_loss: 0.10085305603747315
```

In [12]:
```python
"""3a: CONV1 """ """The worst one""" """ good but needs more epoch, but it is fast
and there was not any overfitting"""
"""I added dropout to get over overfittiing"""
"""Dont use conv1 network if you use 1 day as sequence"""

if input_size >5:
    model = Sequential()
    model.add(layers.Conv1D(32, input_size-5, activation='relu',
                        input_shape=(None, training_data.shape[-1])))
    model.add(layers.GlobalMaxPooling1D())#Global maxpooling gives you scalar output
    model.add(layers.Dropout(0.7))
    model.add(layers.Dense(output_size, activation='tanh' ))

    model.summary()

    """COMPILE YOUR MODEL"""
    model.compile(optimizer=RMSprop(), loss='mae')


    """TRAINING YOUR MODEL"""
    epoch_size = 100
    batch_size = 32
    history = model.fit(training_data,
                    training_labels,
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (val_data, val_labels))


    """Plotting results"""
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(loss) + 1)
    plt.figure()
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation losses')
    plt.legend()
    plt.show()
```

```
        """PREDICTION - TESTING DATA"""
        test_loss = model.evaluate(test_data, test_labels)
        print('normalized test_loss:', test_loss)
        print('unnormalized test_loss:', test_loss*std[0])


        """Save your model:"""
        model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python
    else:
        print("for 1 day sequence you cannot use Conv layer")
```

```
for 1 day sequence you cannot use Conv layer
```

In [13]:
```
"""3b: Combining CNNs and RNNs to process long sequences"""
""" not bad and it is fast"""


if input_size >5:
    model = Sequential()
    model.add(layers.Conv1D(32, input_size-5, activation='relu',
                            input_shape=(None, training_data.shape[-1])))
    model.add(layers.MaxPooling1D(3))
    model.add(layers.GRU(32, dropout=0.2, recurrent_dropout=0.2))
    model.add(layers.Dropout(0.4))
    model.add(layers.Dense(output_size, activation='tanh'))

    """COMPILE YOUR MODEL"""
    model.compile(optimizer=RMSprop(), loss='mae')


    """TRAINING YOUR MODEL"""
    epoch_size = 22
    batch_size = 32
    history = model.fit(training_data,
                        training_labels,
                        epochs=epoch_size,
                        batch_size=batch_size,
                        validation_data = (val_data, val_labels))


    """Plotting results"""
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(loss) + 1)
    plt.figure()
    plt.plot(epochs, loss, 'bo', label='Training loss')
```

```
        plt.plot(epochs, val_loss, 'b', label='Validation loss')
        plt.title('Training and validation losses')
        plt.legend()
        plt.show()

        """PREDICTION - TESTING DATA"""
        test_loss = model.evaluate(test_data, test_labels)
        print('normalized test_loss:', test_loss)
        print('unnormalized test_loss:', test_loss*std[0])

        """Save your model:"""
        model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python
    else:
        print("for 1 day sequence you cannot use Conv layer")
```

```
for 1 day sequence you cannot use Conv layer
```

```
In [14]: """4: Using DAG network"""
         """When I used different layer types I put here the best architecture and
         diagram for my prediction"""
         """One input but Multiple output. Diagram is shown in the report"""

         from keras import layers
         from keras import Input
         from keras.models import Model



         """Input layer:"""
         inputt = Input(shape=(input_size,training_data.shape[-1]), dtype='float32', name='previ
         a = layers.GRU(32, dropout=0.2, recurrent_dropout=0.2, activation='relu')(inputt)
         a = layers.Dropout(0.4)(a)

         """Output layers for each day:"""
         x = layers.Dense(32, activation='relu')(a)
         x = layers.Dropout(0.4)(x)
         day_1 = layers.Dense(1,activation='tanh', name='day1')(x)

         y = layers.Dense(32, activation='relu')(a)
         y = layers.Dropout(0.4)(y)
         day_2 = layers.Dense(1,activation='tanh', name='day2')(y)

         z = layers.Dense(32, activation='relu')(a)
         z = layers.Dropout(0.4)(z)
         day_3 = layers.Dense(1,activation='tanh', name='day3')(z)
```

```python
v = layers.Dense(32, activation='relu')(a)
v = layers.Dropout(0.4)(v)
day_4 = layers.Dense(1,activation='tanh', name='day4')(v)

w = layers.Dense(32, activation='relu')(a)
w = layers.Dropout(0.4)(w)
day_5 = layers.Dense(1,activation='tanh', name='day5')(w)

b = layers.Dense(32, activation='relu')(a)
b = layers.Dropout(0.4)(b)
day_6 = layers.Dense(1,activation='tanh', name='day6')(b)

c = layers.Dense(32, activation='relu')(a)
c = layers.Dropout(0.4)(c)
day_7 = layers.Dense(1,activation='tanh', name='day7')(c)


"""Fully connected API model:"""
model = Model(inputt, [day_1, day_2, day_3, day_4, day_5, day_6, day_7])

"""Compiling:"""
"""I could add multiple losses but my problem isa regression so only loss here is mae""
"""I can also define different loss weights for different outputs, but that would be
good to use it when we have different type of loss functions. Just in case I have
used different weights but it didnt affaect my results much"""

model.compile(optimizer=RMSprop(), loss='mae')


"""TRAINING YOUR MODEL. Here I will assign target labels for each days seperately"""
epoch_size = 20
batch_size = 32
history = model.fit(training_data,
                   [training_labels[:,0],
                    training_labels[:,1],
                    training_labels[:,2],
                    training_labels[:,3],
                    training_labels[:,4],
                    training_labels[:,5],
                    training_labels[:,6]],
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (val_data,
                   [val_labels[:,0],
                    val_labels[:,1],
                    val_labels[:,2],
                    val_labels[:,3],
                    val_labels[:,4],
```

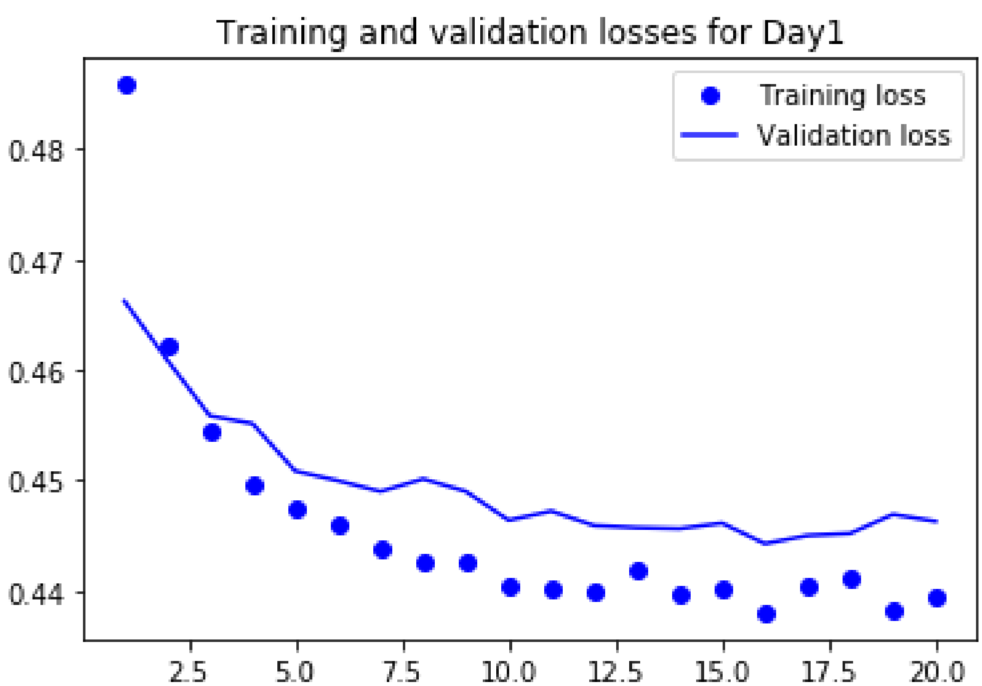
```
                    val_labels[:,5],
                    val_labels[:,6]]))



"""Plot losses for each day in different plots"""
"""Predict losses for each day seperately:"""

"""Day1:"""
loss = history.history['day1_loss']
val_loss = history.history['val_day1_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day1')
plt.legend()
plt.show()


###################################################
"""Day2:"""
loss = history.history['day2_loss']
val_loss = history.history['val_day2_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day2')
plt.legend()
plt.show()


###################################################
"""Day3:"""
loss = history.history['day3_loss']
val_loss = history.history['val_day3_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day3')
plt.legend()
plt.show()



###################################################
```

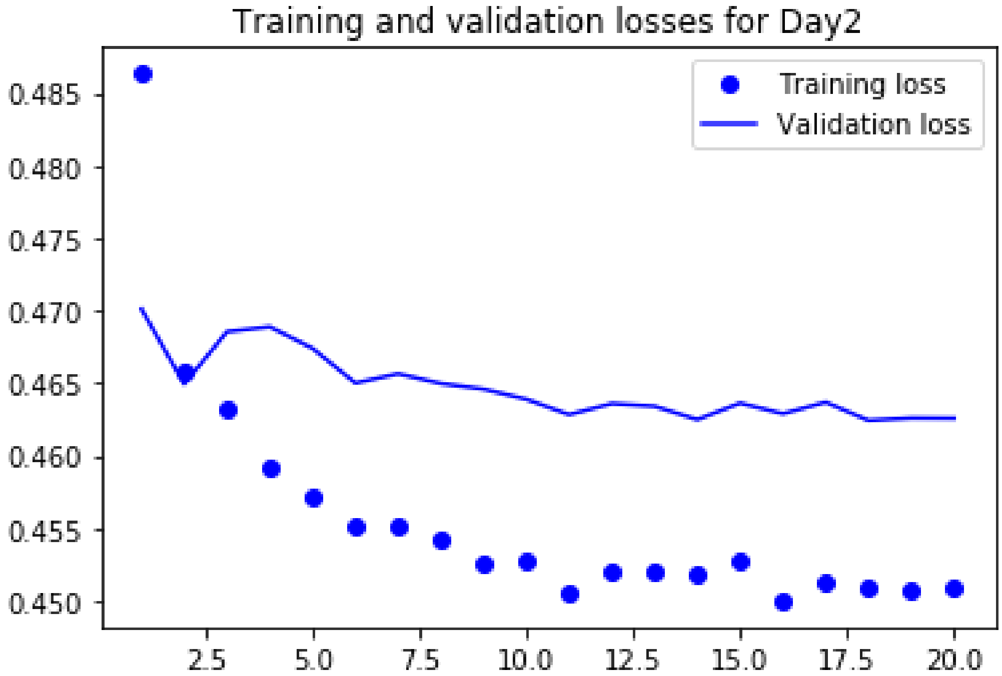
```python
"""Day4:"""
loss = history.history['day4_loss']
val_loss = history.history['val_day4_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day4')
plt.legend()
plt.show()


###################################################
"""Day5:"""
loss = history.history['day5_loss']
val_loss = history.history['val_day5_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day5')
plt.legend()
plt.show()


###################################################
"""Day6:"""
loss = history.history['day6_loss']
val_loss = history.history['val_day6_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day6')
plt.legend()
plt.show()


###################################################
"""Day7:"""
loss = history.history['day7_loss']
val_loss = history.history['val_day7_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day7')
plt.legend()
```

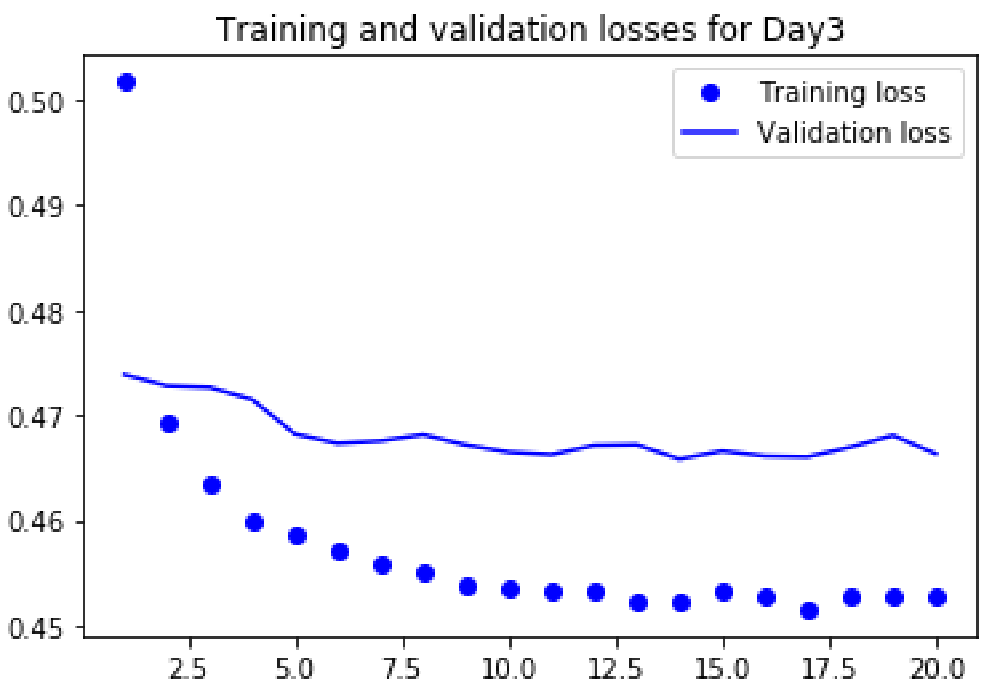
```python
plt.show()


##################################################
##################################################
"""PREDICTION - TESTING DATA for each days both normalized and unnormalized
for DAG model"""
test_LossAndAcc = model.evaluate(test_data, [i for i in np.transpose(test_labels)])
test_losses = test_LossAndAcc[1:]
for i, k in enumerate(test_losses):
    print('normalized test_loss of ', 'day', i+1, 'is', test_losses[i])
    print('unnormalized test_loss of ', 'day', i+1, 'is', test_losses[i]*std[0])


##################################################
##################################################
"""Base case for each day and mean of mae:"""
preds = np.mean(val_data[:, :, 0], axis=1)
day = np.zeros((val_labels.shape[1], val_labels.shape[0]))
mae_base1 = np.zeros((val_labels.shape[1],))
for i,j in enumerate(np.transpose(val_labels)):
    day[i] = val_labels[:,i]
    mae_base1[i] = np.nanmean(np.abs(preds - day[i]))
    print('normalized MAE of base model for day ', i+1, " is ", mae_base1[i])
    print('unnormalized MAE of base model for day ', i+1, " is ", mae_base1[i]*std[0])
mae_base_mean = mae_base1.mean()
print('mean of normalized MAE of base model of week ', " is ", mae_base_mean)
print('mean of unnormalized MAE of base model of week ', " is ", mae_base_mean*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

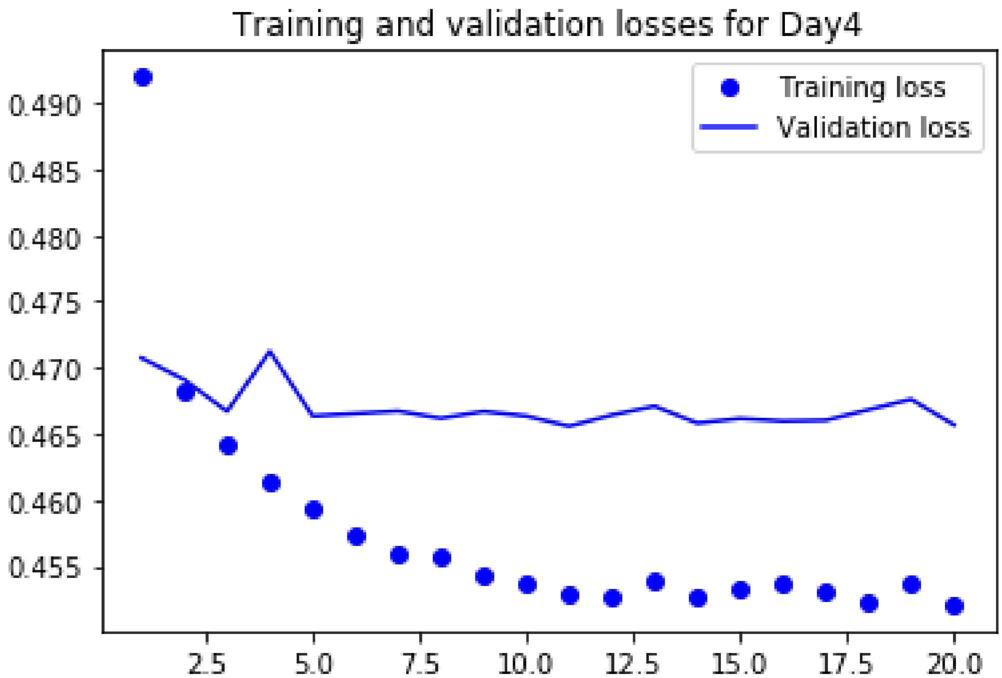
```
Train on 8514 samples, validate on 8514 samples
Epoch 1/20
8514/8514 [==============================] - 3s 359us/step - loss: 3.4401 - day1_loss: 0.4858 -
Epoch 2/20
8514/8514 [==============================] - 1s 90us/step - loss: 3.2724 - day1_loss: 0.4622 - d
Epoch 3/20
8514/8514 [==============================] - 1s 92us/step - loss: 3.2389 - day1_loss: 0.4544 - d
Epoch 4/20
8514/8514 [==============================] - 1s 88us/step - loss: 3.2156 - day1_loss: 0.4497 - d
Epoch 5/20
8514/8514 [==============================] - 1s 90us/step - loss: 3.2029 - day1_loss: 0.4474 - d
Epoch 6/20
8514/8514 [==============================] - 1s 90us/step - loss: 3.1896 - day1_loss: 0.4459 - d
Epoch 7/20
8514/8514 [==============================] - 1s 90us/step - loss: 3.1819 - day1_loss: 0.4438 - d
Epoch 8/20
```

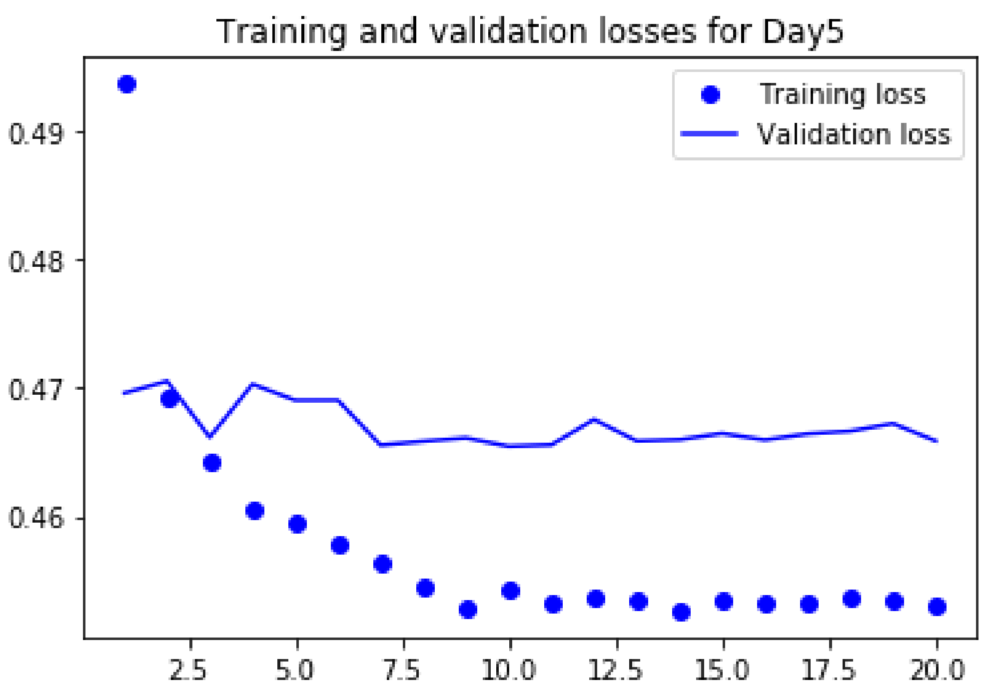
```
8514/8514 [==============================] - 1s 92us/step - loss: 3.1748 - day1_loss: 0.4427 - d
Epoch 9/20
8514/8514 [==============================] - 1s 96us/step - loss: 3.1679 - day1_loss: 0.4426 - d
Epoch 10/20
8514/8514 [==============================] - 1s 88us/step - loss: 3.1653 - day1_loss: 0.4404 - d
Epoch 11/20
8514/8514 [==============================] - 1s 93us/step - loss: 3.1601 - day1_loss: 0.4401 - d
Epoch 12/20
8514/8514 [==============================] - 1s 90us/step - loss: 3.1600 - day1_loss: 0.4400 - d
Epoch 13/20
8514/8514 [==============================] - 1s 90us/step - loss: 3.1616 - day1_loss: 0.4418 - d
Epoch 14/20
8514/8514 [==============================] - 1s 89us/step - loss: 3.1574 - day1_loss: 0.4397 - d
Epoch 15/20
8514/8514 [==============================] - 1s 92us/step - loss: 3.1607 - day1_loss: 0.4402 - d
Epoch 16/20
8514/8514 [==============================] - 1s 101us/step - loss: 3.1580 - day1_loss: 0.4380 -
Epoch 17/20
8514/8514 [==============================] - 1s 94us/step - loss: 3.1602 - day1_loss: 0.4403 - d
Epoch 18/20
8514/8514 [==============================] - 1s 91us/step - loss: 3.1601 - day1_loss: 0.4413 - d
Epoch 19/20
8514/8514 [==============================] - 1s 91us/step - loss: 3.1581 - day1_loss: 0.4382 - d
Epoch 20/20
8514/8514 [==============================] - 1s 90us/step - loss: 3.1574 - day1_loss: 0.4394 - d
```

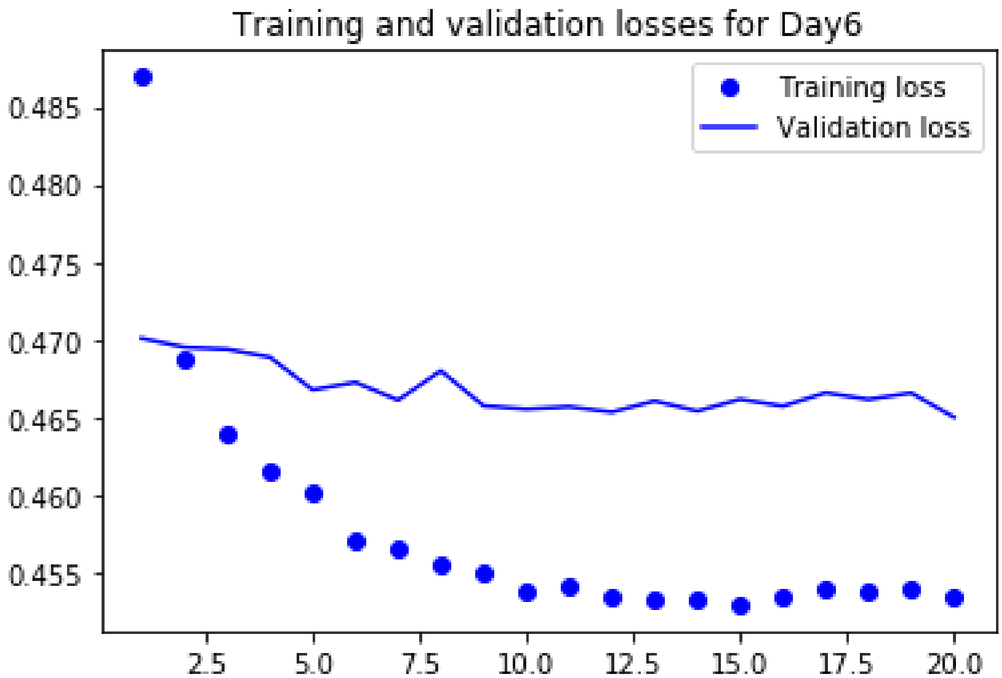


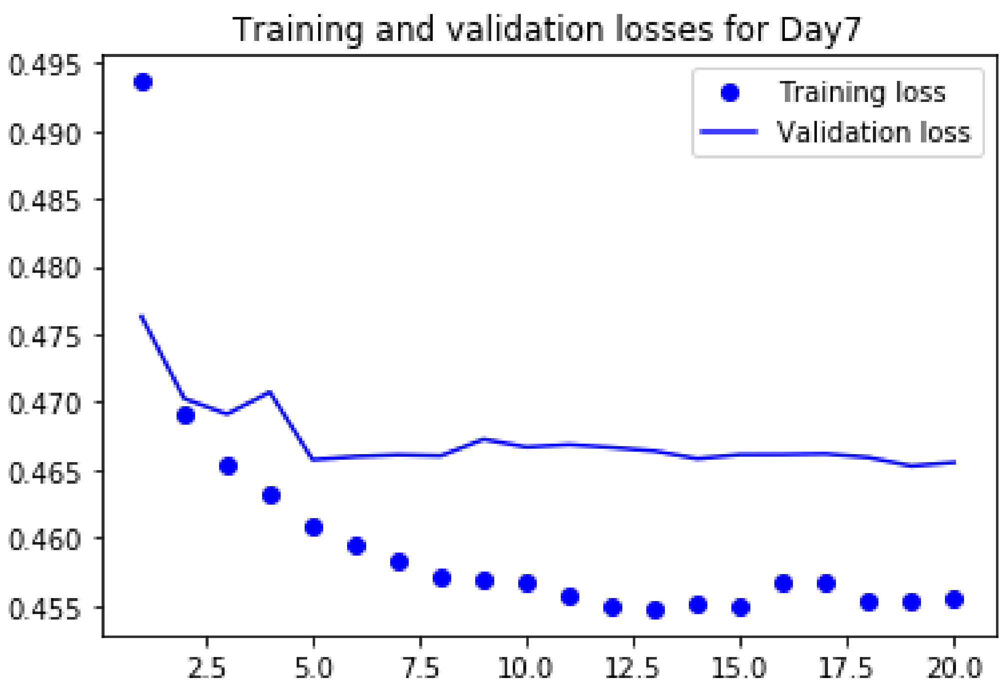

Training and validation losses for Day2
Training loss
Validation loss
0.485
0.480
0.475
0.470
0.465
0.460
0.455
0.450
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation losses for Day3
Training loss
Validation loss
0.50
0.49
0.48
0.47
0.46
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation losses for Day4
Training loss
Validation loss
0.490
0.485
0.480
0.475
0.470
0.465
0.460
0.455
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation losses for Day5
Training loss
Validation loss
0.49
0.48
0.47
0.46
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0






```
8515/8515 [==============================] - 0s 22us/step
normalized test_loss of  day 1 is 0.4155059292921233
```

```
unnormalized test_loss of  day 1 is 0.0968972885967772
normalized test_loss of  day 2 is 0.4323888610267527
unnormalized test_loss of  day 2 is 0.10083444133834478
normalized test_loss of  day 3 is 0.4349519806263801
unnormalized test_loss of  day 3 is 0.10143216888455878
normalized test_loss of  day 4 is 0.43459401087670274
unnormalized test_loss of  day 4 is 0.10134868921387756
normalized test_loss of  day 5 is 0.4359098024460686
unnormalized test_loss of  day 5 is 0.10165553594323051
normalized test_loss of  day 6 is 0.4363541909476553
unnormalized test_loss of  day 6 is 0.1017591687384609
normalized test_loss of  day 7 is 0.4340501322686077
unnormalized test_loss of  day 7 is 0.10122185501312385
normalized MAE of base model for day  1  is  0.5757574691956336
unnormalized MAE of base model for day  1  is  0.13426845135385884
normalized MAE of base model for day  2  is  0.6615618414999005
unnormalized MAE of base model for day  2  is  0.15427830064817905
normalized MAE of base model for day  3  is  0.6920572701585189
unnormalized MAE of base model for day  3  is  0.16138993045488417
normalized MAE of base model for day  4  is  0.7040279924798217
unnormalized MAE of base model for day  4  is  0.16418154052277245
normalized MAE of base model for day  5  is  0.715426965308281
unnormalized MAE of base model for day  5  is  0.16683981681198876
normalized MAE of base model for day  6  is  0.7177373140182679
unnormalized MAE of base model for day  6  is  0.16737859739230967
normalized MAE of base model for day  7  is  0.7227651688117348
unnormalized MAE of base model for day  7  is  0.16855110893209743
mean of normalized MAE of base model of week   is  0.6841905744960226
mean of unnormalized MAE of base model of week   is  0.15955539230229862
```