Untitled3

November 29, 2018

In [39]:
```
# -*- 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
```

In [40]:
```
"""Preparing data"""
data =[]
if data:
    del data

total_data = pd.read_csv("C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfe
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"""
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 nex
    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 = 30 #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_da
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_
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_da
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_
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):
    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 [41]: #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 [42]:
```
"""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 m
In the following base model2, I choose my target  not as each dy of next week but avera
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])
```

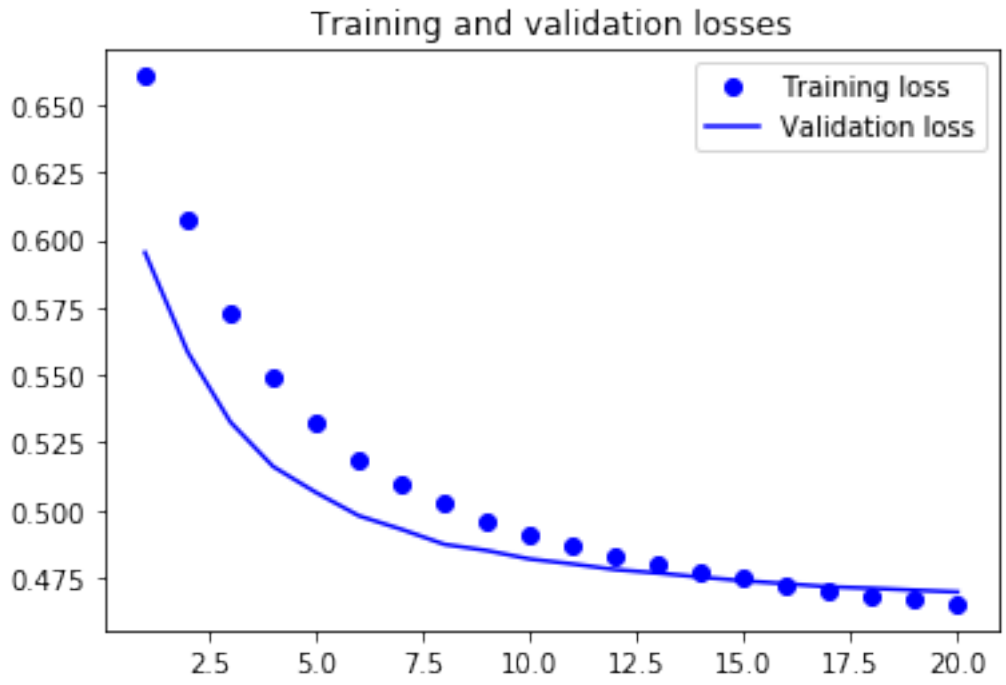
```
normalized MAE of base model for day  1  is  0.588833890647551
unnormalized MAE of base model for day  1  is  0.13733719109034753
normalized MAE of base model for day  2  is  0.5953143924718292
unnormalized MAE of base model for day  2  is  0.13884867664092185
normalized MAE of base model for day  3  is  0.5984563071322808
unnormalized MAE of base model for day  3  is  0.13958148387393876
normalized MAE of base model for day  4  is  0.5997326302290171
unnormalized MAE of base model for day  4  is  0.13987916821550533
normalized MAE of base model for day  5  is  0.600062411970935
unnormalized MAE of base model for day  5  is  0.13995608515053107
normalized MAE of base model for day  6  is  0.6010094802973719
unnormalized MAE of base model for day  6  is  0.14017697546576147
normalized MAE of base model for day  7  is  0.6018344685104254
unnormalized MAE of base model for day  7  is  0.14036939231822698
mean of normalized MAE of base model of week   is  0.5978919401799158
mean of unnormalized MAE of base model of week   is  0.13944985325074757
normalized MAE of base2 model is  0.39251697
unnormalized MAE of base2 model is  0.09154904
```

In [43]:
```
        """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"""
        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 8505 samples, validate on 8505 samples
Epoch 1/20
8505/8505 [==============================] - 3s 304us/step - loss: 0.6605 - val_loss: 0.5953
Epoch 2/20
8505/8505 [==============================] - 0s 49us/step - loss: 0.6078 - val_loss: 0.5584
Epoch 3/20
8505/8505 [==============================] - 0s 45us/step - loss: 0.5729 - val_loss: 0.5325
Epoch 4/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.5492 - val_loss: 0.5160
Epoch 5/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.5321 - val_loss: 0.5064
Epoch 6/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.5188 - val_loss: 0.4978
Epoch 7/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.5098 - val_loss: 0.4928
Epoch 8/20
8505/8505 [==============================] - 0s 48us/step - loss: 0.5022 - val_loss: 0.4873
Epoch 9/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.4958 - val_loss: 0.4850
Epoch 10/20
8505/8505 [==============================] - 0s 45us/step - loss: 0.4909 - val_loss: 0.4818
Epoch 11/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.4869 - val_loss: 0.4800
```

```
Epoch 12/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.4830 - val_loss: 0.4779
Epoch 13/20
8505/8505 [==============================] - 0s 48us/step - loss: 0.4799 - val_loss: 0.4766
Epoch 14/20
8505/8505 [==============================] - 0s 46us/step - loss: 0.4773 - val_loss: 0.4751
Epoch 15/20
8505/8505 [==============================] - 0s 48us/step - loss: 0.4751 - val_loss: 0.4738
Epoch 16/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.4721 - val_loss: 0.4727
Epoch 17/20
8505/8505 [==============================] - 0s 46us/step - loss: 0.4703 - val_loss: 0.4716
Epoch 18/20
8505/8505 [==============================] - 0s 46us/step - loss: 0.4684 - val_loss: 0.4710
Epoch 19/20
8505/8505 [==============================] - 0s 47us/step - loss: 0.4669 - val_loss: 0.4703
Epoch 20/20
8505/8505 [==============================] - 0s 48us/step - loss: 0.4652 - val_loss: 0.4697
```

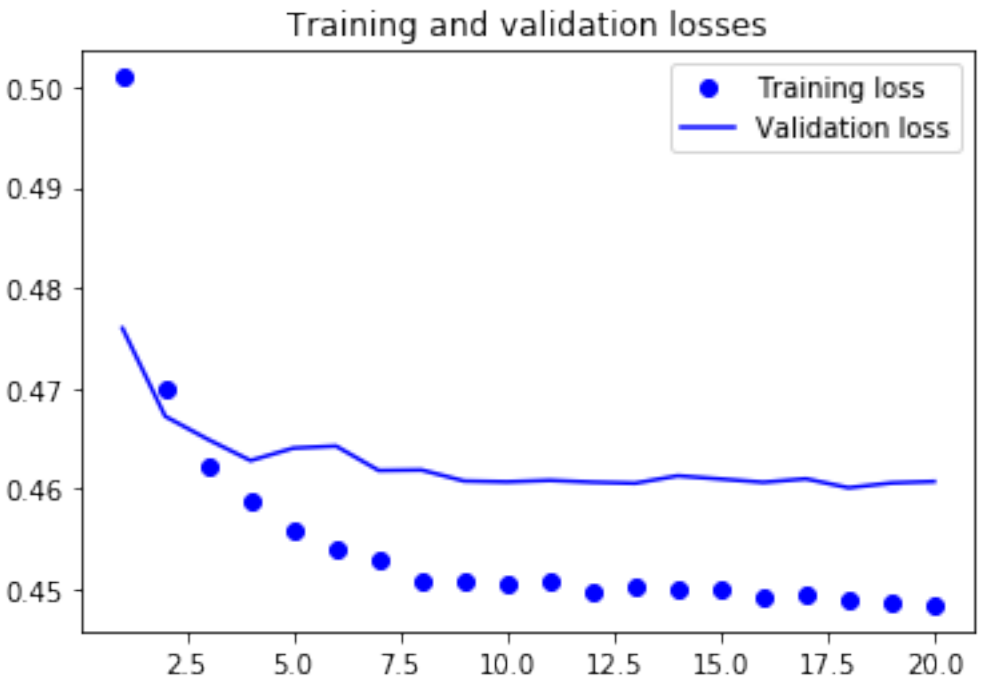



```
8504/8504 [==============================] - 0s 15us/step
normalized test_loss: 0.4410715904700117
unnormalized test_loss: 0.10287372086937376
```

In [44]:
```
"""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 un
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/Pro
```

```
Train on 8505 samples, validate on 8505 samples
Epoch 1/20
8505/8505 [==============================] - 6s 762us/step - loss: 0.5010 - val_loss: 0.4760
Epoch 2/20
8505/8505 [==============================] - 4s 453us/step - loss: 0.4700 - val_loss: 0.4672
Epoch 3/20
8505/8505 [==============================] - 4s 457us/step - loss: 0.4623 - val_loss: 0.4649
Epoch 4/20
8505/8505 [==============================] - 4s 461us/step - loss: 0.4589 - val_loss: 0.4628
Epoch 5/20
8505/8505 [==============================] - 4s 479us/step - loss: 0.4558 - val_loss: 0.4641
Epoch 6/20
8505/8505 [==============================] - 4s 500us/step - loss: 0.4541 - val_loss: 0.4643
Epoch 7/20
8505/8505 [==============================] - 4s 496us/step - loss: 0.4529 - val_loss: 0.4618
Epoch 8/20
8505/8505 [==============================] - 4s 483us/step - loss: 0.4507 - val_loss: 0.4619
Epoch 9/20
8505/8505 [==============================] - 4s 454us/step - loss: 0.4508 - val_loss: 0.4608
Epoch 10/20
8505/8505 [==============================] - 4s 457us/step - loss: 0.4504 - val_loss: 0.4607
Epoch 11/20
8505/8505 [==============================] - 4s 454us/step - loss: 0.4507 - val_loss: 0.4608
Epoch 12/20
8505/8505 [==============================] - 4s 455us/step - loss: 0.4497 - val_loss: 0.4607
Epoch 13/20
8505/8505 [==============================] - 4s 454us/step - loss: 0.4502 - val_loss: 0.4606
Epoch 14/20
8505/8505 [==============================] - 4s 455us/step - loss: 0.4500 - val_loss: 0.4613
Epoch 15/20
8505/8505 [==============================] - 4s 455us/step - loss: 0.4499 - val_loss: 0.4610
Epoch 16/20
8505/8505 [==============================] - 4s 454us/step - loss: 0.4492 - val_loss: 0.4607
Epoch 17/20
8505/8505 [==============================] - 4s 469us/step - loss: 0.4494 - val_loss: 0.4610
Epoch 18/20
8505/8505 [==============================] - 4s 458us/step - loss: 0.4489 - val_loss: 0.4601
Epoch 19/20
8505/8505 [==============================] - 4s 462us/step - loss: 0.4486 - val_loss: 0.4606
Epoch 20/20
8505/8505 [==============================] - 4s 456us/step - loss: 0.4484 - val_loss: 0.4607
```




```
8504/8504 [==============================] - 1s 72us/step
normalized test_loss: 0.4311941695291819
unnormalized test_loss: 0.10056995189687332
```

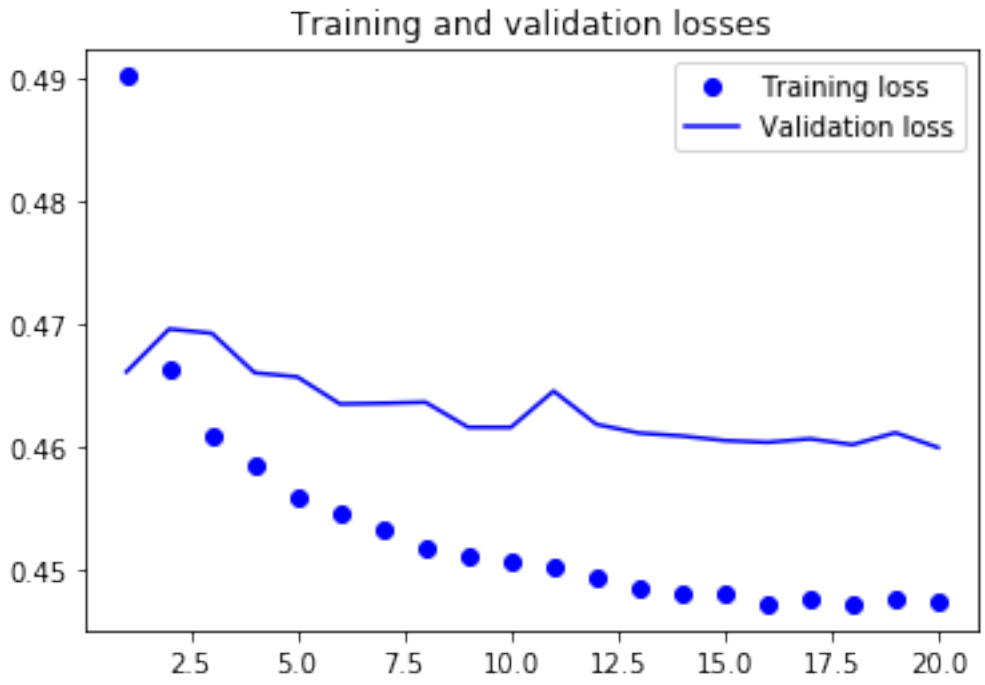
In [45]: """2b: Training and evaluating a dropout-regularized, stacked GRU model"""

```
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/Pro
```

```
Train on 8505 samples, validate on 8505 samples
Epoch 1/20
8505/8505 [==============================] - 10s 1ms/step - loss: 0.4902 - val_loss: 0.4661
Epoch 2/20
8505/8505 [==============================] - 7s 851us/step - loss: 0.4663 - val_loss: 0.4696
Epoch 3/20
8505/8505 [==============================] - 7s 854us/step - loss: 0.4608 - val_loss: 0.4692
Epoch 4/20
8505/8505 [==============================] - 7s 844us/step - loss: 0.4584 - val_loss: 0.4660
Epoch 5/20
8505/8505 [==============================] - 7s 844us/step - loss: 0.4559 - val_loss: 0.4657
Epoch 6/20
8505/8505 [==============================] - 7s 845us/step - loss: 0.4546 - val_loss: 0.4635
Epoch 7/20
8505/8505 [==============================] - 7s 850us/step - loss: 0.4532 - val_loss: 0.4635
```

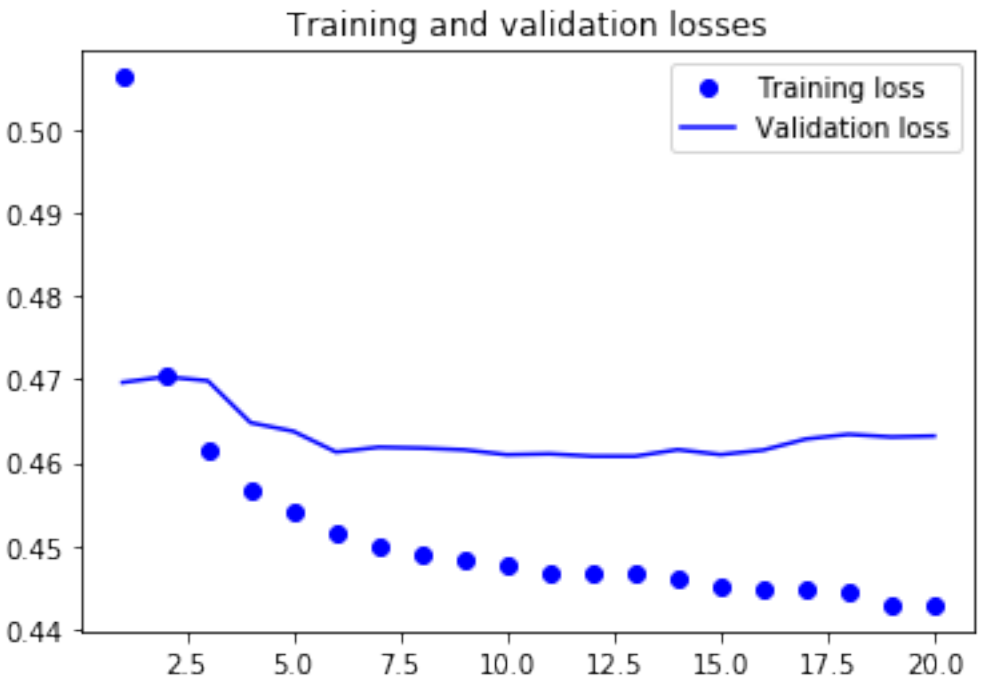
```
Epoch 8/20
8505/8505 [==============================] - 7s 854us/step - loss: 0.4518 - val_loss: 0.4636
Epoch 9/20
8505/8505 [==============================] - 7s 853us/step - loss: 0.4512 - val_loss: 0.4616
Epoch 10/20
8505/8505 [==============================] - 7s 850us/step - loss: 0.4507 - val_loss: 0.4616
Epoch 11/20
8505/8505 [==============================] - 7s 847us/step - loss: 0.4503 - val_loss: 0.4645
Epoch 12/20
8505/8505 [==============================] - 7s 855us/step - loss: 0.4493 - val_loss: 0.4618
Epoch 13/20
8505/8505 [==============================] - 7s 853us/step - loss: 0.4485 - val_loss: 0.4611
Epoch 14/20
8505/8505 [==============================] - 7s 850us/step - loss: 0.4480 - val_loss: 0.4609
Epoch 15/20
8505/8505 [==============================] - 7s 847us/step - loss: 0.4480 - val_loss: 0.4605
Epoch 16/20
8505/8505 [==============================] - 7s 856us/step - loss: 0.4473 - val_loss: 0.4604
Epoch 17/20
8505/8505 [==============================] - 7s 850us/step - loss: 0.4475 - val_loss: 0.4606
Epoch 18/20
8505/8505 [==============================] - 7s 851us/step - loss: 0.4472 - val_loss: 0.4602
Epoch 19/20
8505/8505 [==============================] - 7s 862us/step - loss: 0.4476 - val_loss: 0.4611
Epoch 20/20
8505/8505 [==============================] - 7s 855us/step - loss: 0.4475 - val_loss: 0.4599
```



```
8504/8504 [==============================] - 1s 148us/step
normalized test_loss: 0.4310360036699254
unnormalized test_loss: 0.1005330619433878
```

In [46]:
```
"""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 8505 samples, validate on 8505 samples
Epoch 1/20
8505/8505 [==============================] - 8s 949us/step - loss: 0.5064 - val_loss: 0.4697
Epoch 2/20
8505/8505 [==============================] - 5s 571us/step - loss: 0.4705 - val_loss: 0.4703
Epoch 3/20
8505/8505 [==============================] - 5s 574us/step - loss: 0.4614 - val_loss: 0.4698
Epoch 4/20
8505/8505 [==============================] - 5s 572us/step - loss: 0.4566 - val_loss: 0.4648
Epoch 5/20
8505/8505 [==============================] - 5s 575us/step - loss: 0.4541 - val_loss: 0.4638
Epoch 6/20
8505/8505 [==============================] - 5s 610us/step - loss: 0.4515 - val_loss: 0.4613
Epoch 7/20
8505/8505 [==============================] - 5s 614us/step - loss: 0.4499 - val_loss: 0.4619
Epoch 8/20
8505/8505 [==============================] - 5s 618us/step - loss: 0.4491 - val_loss: 0.4617
Epoch 9/20
8505/8505 [==============================] - 5s 609us/step - loss: 0.4483 - val_loss: 0.4615
Epoch 10/20
8505/8505 [==============================] - 5s 578us/step - loss: 0.4478 - val_loss: 0.4610
Epoch 11/20
8505/8505 [==============================] - 5s 578us/step - loss: 0.4468 - val_loss: 0.4611
Epoch 12/20
8505/8505 [==============================] - 5s 578us/step - loss: 0.4467 - val_loss: 0.4608
Epoch 13/20
8505/8505 [==============================] - 5s 577us/step - loss: 0.4466 - val_loss: 0.4608
Epoch 14/20
8505/8505 [==============================] - 5s 577us/step - loss: 0.4459 - val_loss: 0.4616
Epoch 15/20
8505/8505 [==============================] - 5s 578us/step - loss: 0.4450 - val_loss: 0.4610
Epoch 16/20
8505/8505 [==============================] - 5s 580us/step - loss: 0.4447 - val_loss: 0.4615
Epoch 17/20
8505/8505 [==============================] - 5s 578us/step - loss: 0.4447 - val_loss: 0.4628
Epoch 18/20
8505/8505 [==============================] - 5s 580us/step - loss: 0.4444 - val_loss: 0.4634
Epoch 19/20
8505/8505 [==============================] - 5s 580us/step - loss: 0.4429 - val_loss: 0.4631
Epoch 20/20
8505/8505 [==============================] - 5s 578us/step - loss: 0.4429 - val_loss: 0.4632
```



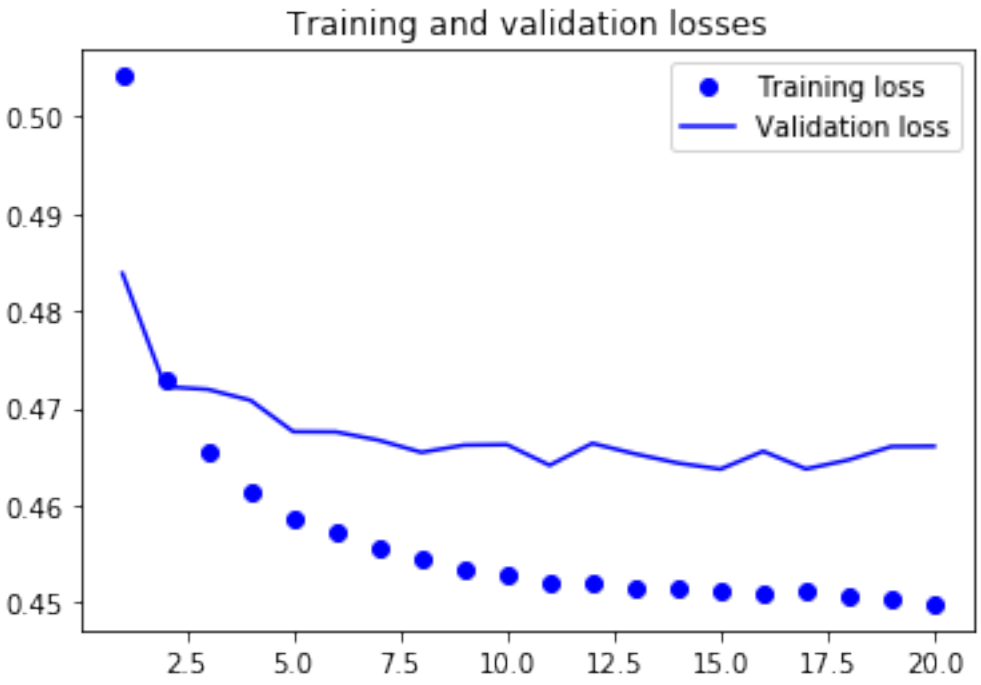

```
8504/8504 [==============================] - 1s 94us/step
normalized test_loss: 0.4345760251538213
unnormalized test_loss: 0.10135872197199637
```

```
In [47]: """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"""
```

```
        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 8505 samples, validate on 8505 samples
Epoch 1/20
8505/8505 [==============================] - 7s 810us/step - loss: 0.5041 - val_loss: 0.4839
Epoch 2/20
8505/8505 [==============================] - 4s 488us/step - loss: 0.4728 - val_loss: 0.4722
Epoch 3/20
8505/8505 [==============================] - 4s 489us/step - loss: 0.4655 - val_loss: 0.4719
Epoch 4/20
8505/8505 [==============================] - 4s 495us/step - loss: 0.4614 - val_loss: 0.4708
Epoch 5/20
8505/8505 [==============================] - 4s 499us/step - loss: 0.4587 - val_loss: 0.4676
Epoch 6/20
8505/8505 [==============================] - 4s 500us/step - loss: 0.4572 - val_loss: 0.4676
Epoch 7/20
8505/8505 [==============================] - 4s 502us/step - loss: 0.4555 - val_loss: 0.4667
```

```
Epoch 8/20
8505/8505 [==============================] - 4s 509us/step - loss: 0.4545 - val_loss: 0.4655
Epoch 9/20
8505/8505 [==============================] - 4s 505us/step - loss: 0.4534 - val_loss: 0.4662
Epoch 10/20
8505/8505 [==============================] - 4s 510us/step - loss: 0.4529 - val_loss: 0.4663
Epoch 11/20
8505/8505 [==============================] - 4s 512us/step - loss: 0.4521 - val_loss: 0.4641
Epoch 12/20
8505/8505 [==============================] - 4s 507us/step - loss: 0.4519 - val_loss: 0.4664
Epoch 13/20
8505/8505 [==============================] - 4s 512us/step - loss: 0.4516 - val_loss: 0.4653
Epoch 14/20
8505/8505 [==============================] - 4s 522us/step - loss: 0.4514 - val_loss: 0.4644
Epoch 15/20
8505/8505 [==============================] - 4s 523us/step - loss: 0.4511 - val_loss: 0.4637
Epoch 16/20
8505/8505 [==============================] - 4s 506us/step - loss: 0.4508 - val_loss: 0.4656
Epoch 17/20
8505/8505 [==============================] - 4s 508us/step - loss: 0.4511 - val_loss: 0.4638
Epoch 18/20
8505/8505 [==============================] - 4s 506us/step - loss: 0.4508 - val_loss: 0.4647
Epoch 19/20
8505/8505 [==============================] - 4s 503us/step - loss: 0.4503 - val_loss: 0.4661
Epoch 20/20
8505/8505 [==============================] - 4s 510us/step - loss: 0.4499 - val_loss: 0.4661
```



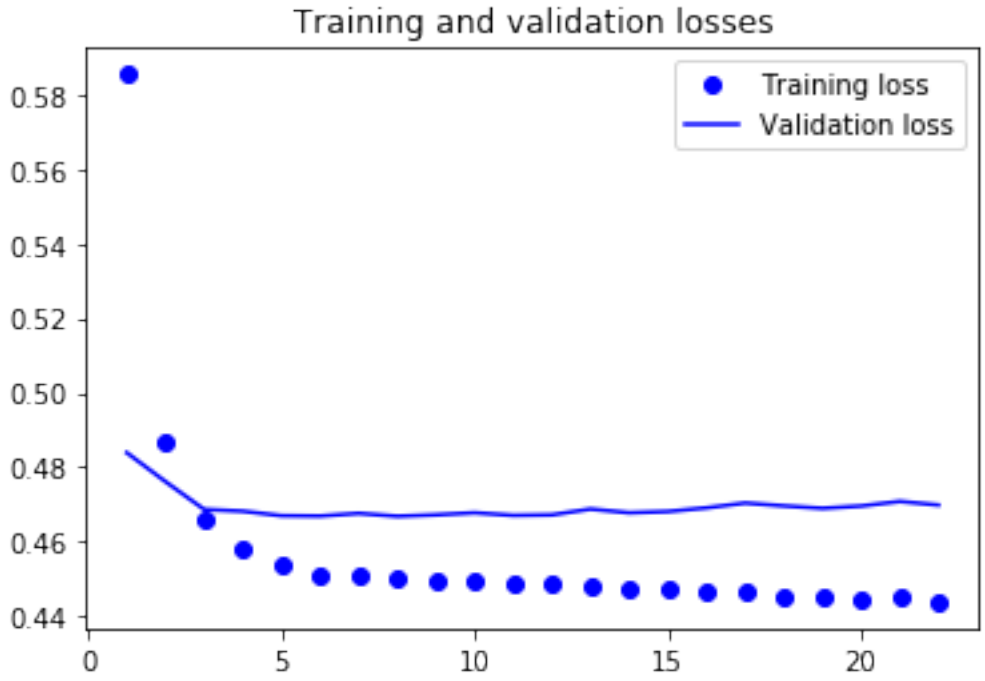
```
8504/8504 [==============================] - 1s 74us/step
normalized test_loss: 0.44597063416147276
unnormalized test_loss: 0.10401635363949885
```

In [48]:
```
"""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 = 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")
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_5 (Conv1D)            (None, None, 32)          2432
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 32)                0
_________________________________________________________________
dropout_49 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_66 (Dense)             (None, 7)                 231
=================================================================
Total params: 2,663
Trainable params: 2,663
Non-trainable params: 0
_________________________________________________________________
Train on 8505 samples, validate on 8505 samples
Epoch 1/22
8505/8505 [==============================] - 3s 342us/step - loss: 0.5855 - val_loss: 0.4839
Epoch 2/22
8505/8505 [==============================] - 1s 64us/step - loss: 0.4868 - val_loss: 0.4762
Epoch 3/22
8505/8505 [==============================] - 1s 61us/step - loss: 0.4661 - val_loss: 0.4687
Epoch 4/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4582 - val_loss: 0.4682
Epoch 5/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4537 - val_loss: 0.4670
Epoch 6/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4513 - val_loss: 0.4669
Epoch 7/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4507 - val_loss: 0.4676
Epoch 8/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4500 - val_loss: 0.4669
Epoch 9/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4496 - val_loss: 0.4672
Epoch 10/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4495 - val_loss: 0.4678
```

```
Epoch 11/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4488 - val_loss: 0.4671
Epoch 12/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4486 - val_loss: 0.4672
Epoch 13/22
8505/8505 [==============================] - 1s 61us/step - loss: 0.4478 - val_loss: 0.4688
Epoch 14/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4475 - val_loss: 0.4678
Epoch 15/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4471 - val_loss: 0.4681
Epoch 16/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4468 - val_loss: 0.4691
Epoch 17/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4468 - val_loss: 0.4704
Epoch 18/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4455 - val_loss: 0.4696
Epoch 19/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4450 - val_loss: 0.4690
Epoch 20/22
8505/8505 [==============================] - 1s 63us/step - loss: 0.4447 - val_loss: 0.4696
Epoch 21/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4451 - val_loss: 0.4708
Epoch 22/22
8505/8505 [==============================] - 1s 62us/step - loss: 0.4439 - val_loss: 0.4699
```



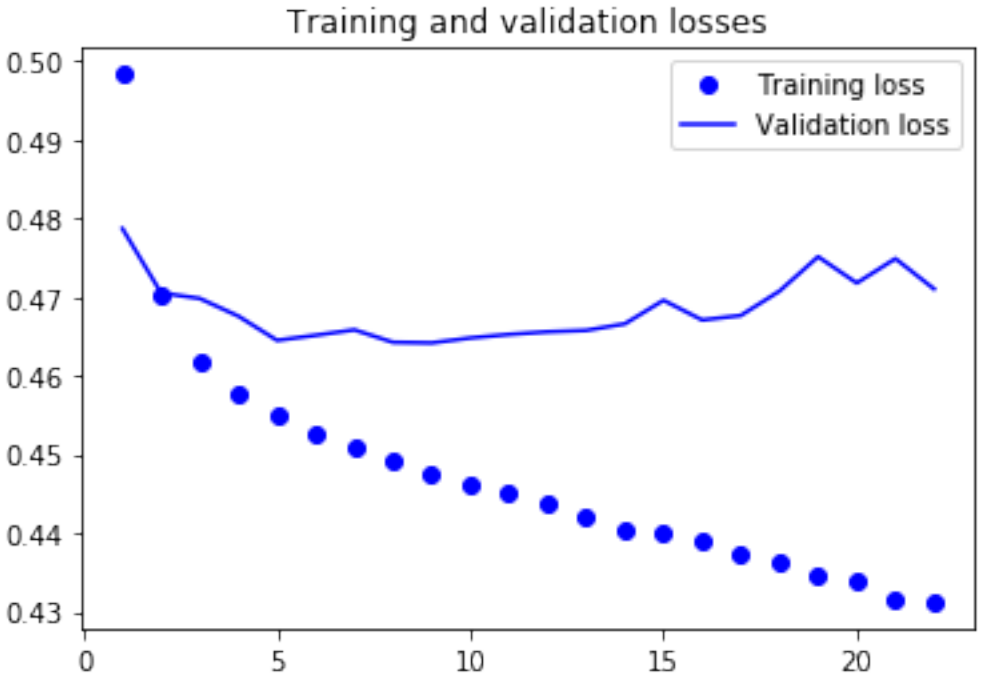
```
8504/8504 [==============================] - 0s 21us/step
normalized test_loss: 0.4379054058697127
unnormalized test_loss: 0.10213525301556123
```

In [49]:
```
"""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")
```

```
Train on 8505 samples, validate on 8505 samples
Epoch 1/22
8505/8505 [==============================] - 4s 452us/step - loss: 0.4983 - val_loss: 0.4787
Epoch 2/22
8505/8505 [==============================] - 1s 102us/step - loss: 0.4701 - val_loss: 0.4705
Epoch 3/22
8505/8505 [==============================] - 1s 98us/step - loss: 0.4619 - val_loss: 0.4698
Epoch 4/22
8505/8505 [==============================] - 1s 100us/step - loss: 0.4575 - val_loss: 0.4676
Epoch 5/22
8505/8505 [==============================] - 1s 103us/step - loss: 0.4550 - val_loss: 0.4645
Epoch 6/22
8505/8505 [==============================] - 1s 98us/step - loss: 0.4527 - val_loss: 0.4651
Epoch 7/22
8505/8505 [==============================] - 1s 101us/step - loss: 0.4508 - val_loss: 0.4658
Epoch 8/22
8505/8505 [==============================] - 1s 98us/step - loss: 0.4492 - val_loss: 0.4642
Epoch 9/22
8505/8505 [==============================] - 1s 98us/step - loss: 0.4474 - val_loss: 0.4642
Epoch 10/22
8505/8505 [==============================] - 1s 101us/step - loss: 0.4462 - val_loss: 0.4648
Epoch 11/22
8505/8505 [==============================] - 1s 100us/step - loss: 0.4451 - val_loss: 0.4652
Epoch 12/22
8505/8505 [==============================] - 1s 101us/step - loss: 0.4436 - val_loss: 0.4656
Epoch 13/22
8505/8505 [==============================] - 1s 100us/step - loss: 0.4419 - val_loss: 0.4657
Epoch 14/22
8505/8505 [==============================] - 1s 99us/step - loss: 0.4404 - val_loss: 0.4666
Epoch 15/22
8505/8505 [==============================] - 1s 102us/step - loss: 0.4401 - val_loss: 0.4696
Epoch 16/22
8505/8505 [==============================] - 1s 99us/step - loss: 0.4389 - val_loss: 0.4671
Epoch 17/22
8505/8505 [==============================] - 1s 98us/step - loss: 0.4373 - val_loss: 0.4676
Epoch 18/22
8505/8505 [==============================] - 1s 100us/step - loss: 0.4365 - val_loss: 0.4707
Epoch 19/22
8505/8505 [==============================] - 1s 100us/step - loss: 0.4346 - val_loss: 0.4751
Epoch 20/22
8505/8505 [==============================] - 1s 100us/step - loss: 0.4338 - val_loss: 0.4718
Epoch 21/22
```

```
8505/8505 [==============================] - 1s 102us/step - loss: 0.4316 - val_loss: 0.4749
Epoch 22/22
8505/8505 [==============================] - 1s 103us/step - loss: 0.4312 - val_loss: 0.4710
```




```
8504/8504 [==============================] - 0s 29us/step
normalized test_loss: 0.4421108092268121
unnormalized test_loss: 0.10311610397138987
```

In [50]:
```
"""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)

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],
```

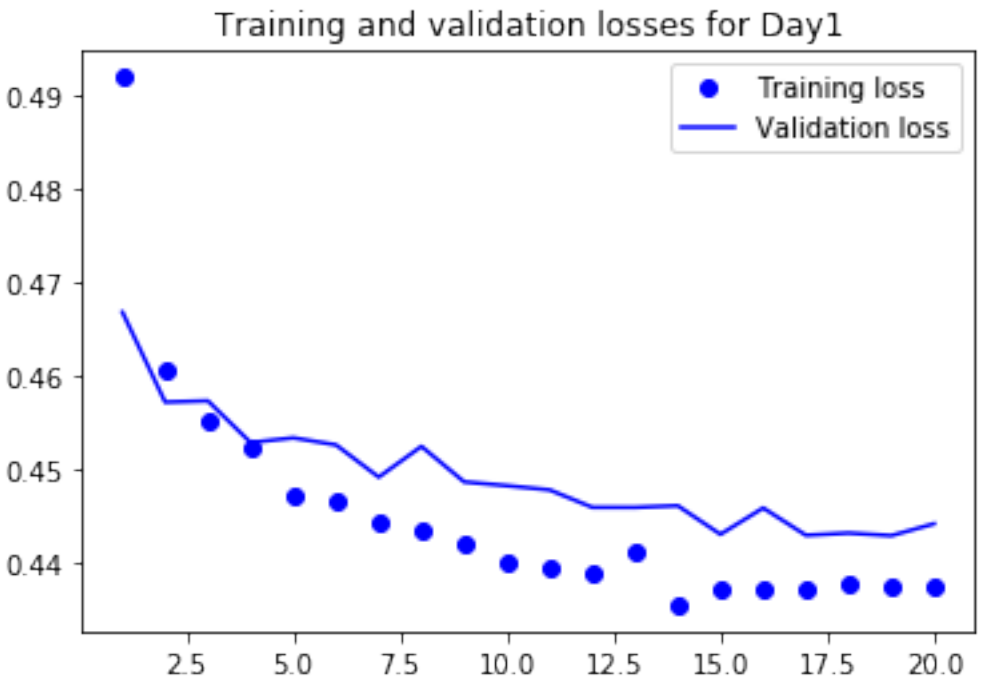
```
                    training_labels[:,1],
                    training_labels[:,2],
                    training_labels[:,3],
                    training_labels[:,4],
                    training_labels[:,5],
                    training_labels[:,6]],
                    epochs=epoch_size,
                    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()


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

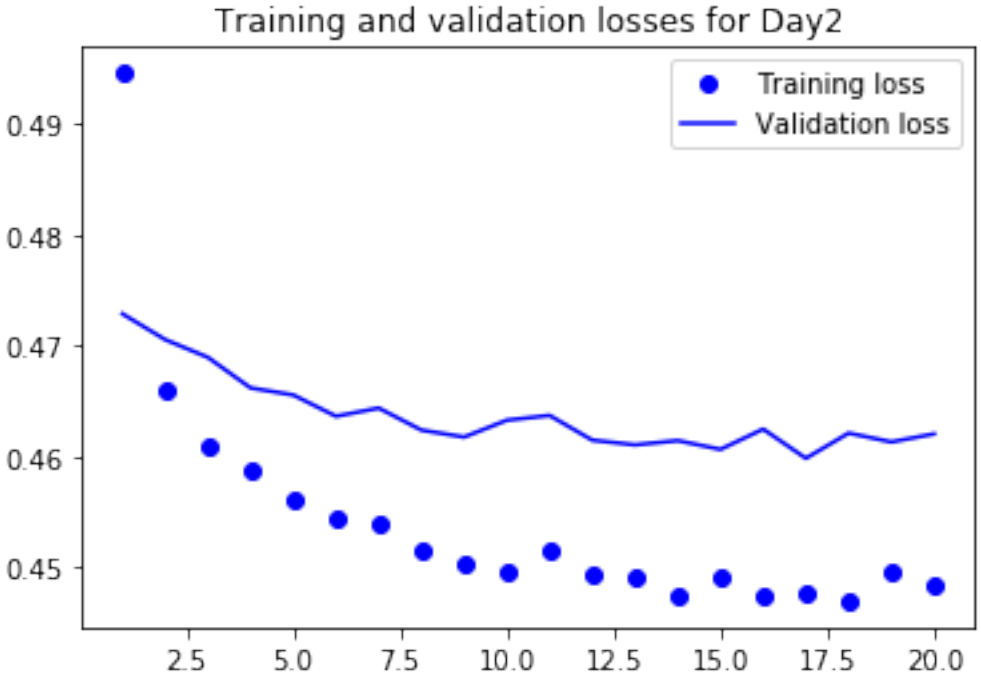
```
"""Day3:"""
loss = history.history['day3_loss']
val_loss = history.history['val_day3_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day3')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



##################################################
"""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')
```

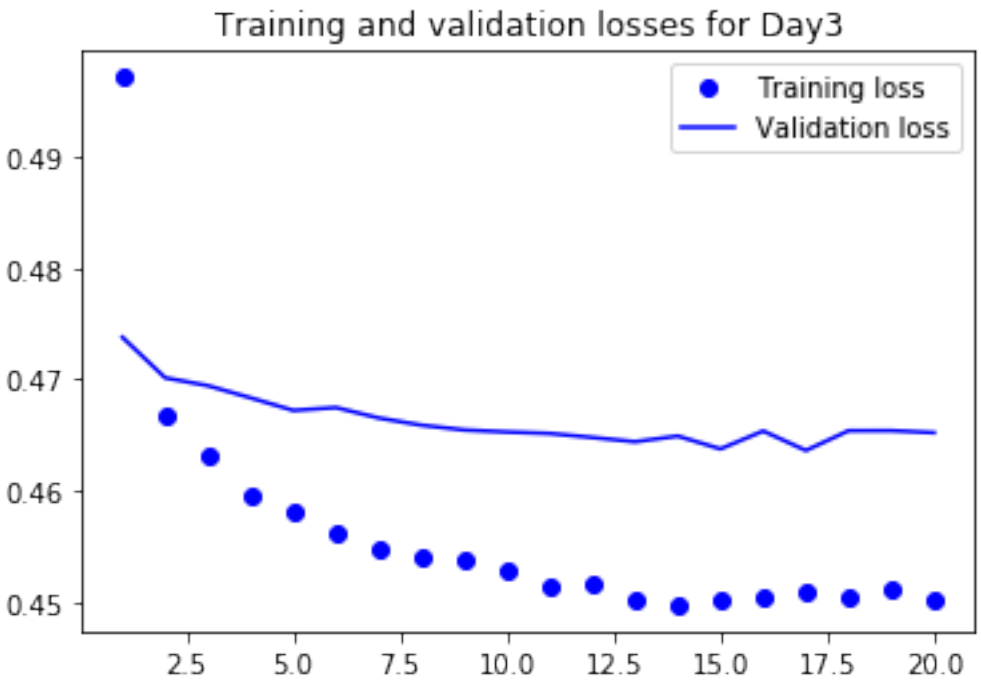
```
plt.legend()
plt.show()


##################################################
"""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()
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
```

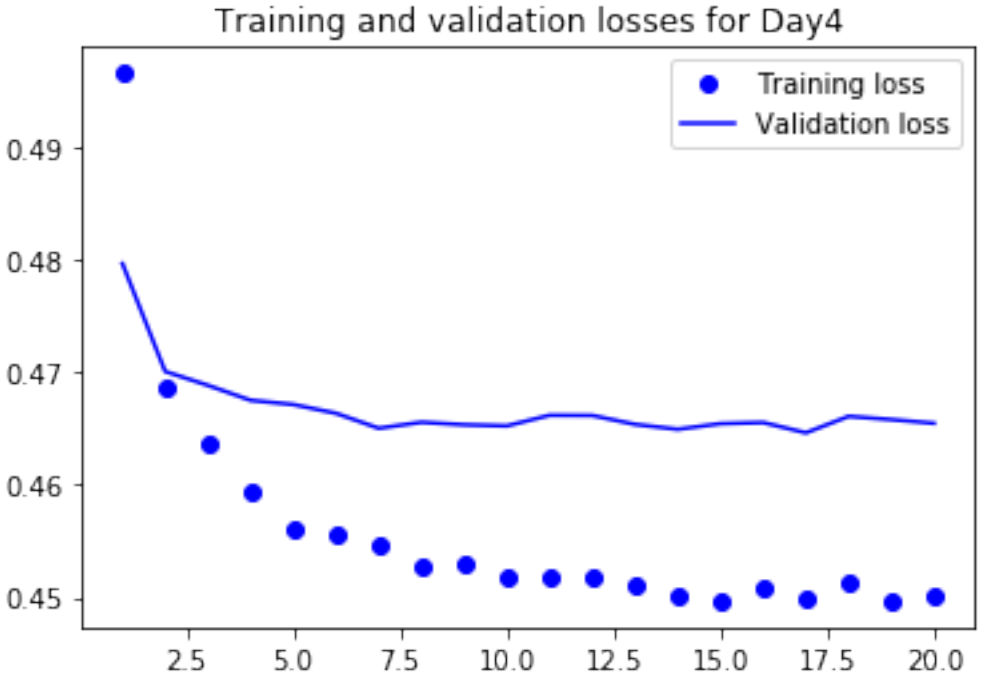
```
Train on 8505 samples, validate on 8505 samples
Epoch 1/20
```

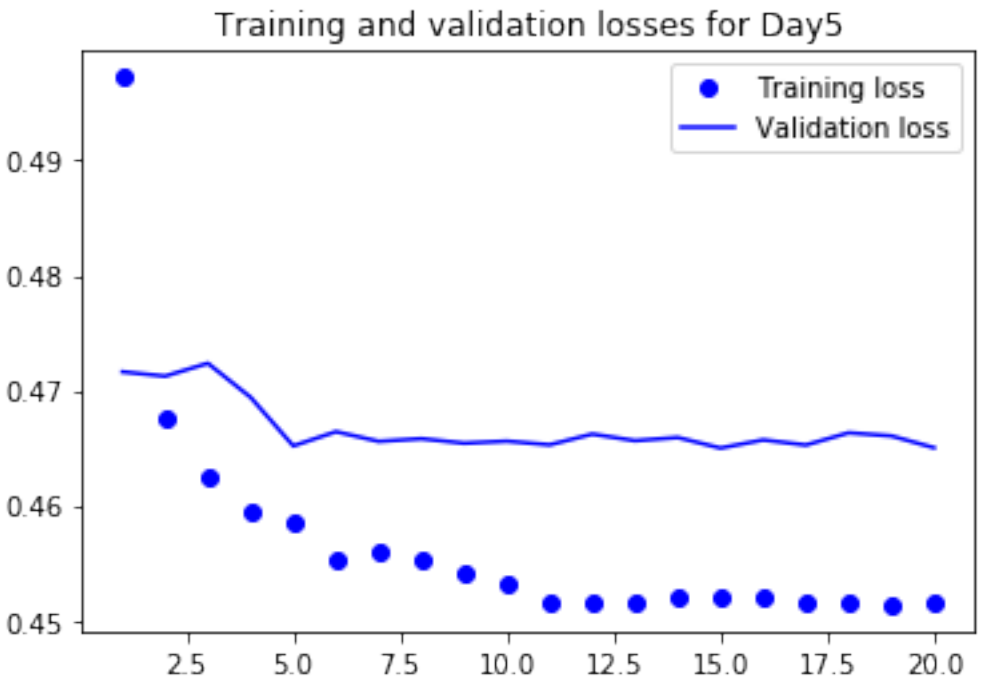
```
8505/8505 [==============================] - 8s 951us/step - loss: 3.4733 - day1_loss: 0.4920 -
Epoch 2/20
8505/8505 [==============================] - 4s 493us/step - loss: 3.2677 - day1_loss: 0.4606 -
Epoch 3/20
8505/8505 [==============================] - 4s 493us/step - loss: 3.2335 - day1_loss: 0.4553 -
Epoch 4/20
8505/8505 [==============================] - 4s 496us/step - loss: 3.2122 - day1_loss: 0.4522 -
Epoch 5/20
8505/8505 [==============================] - 4s 500us/step - loss: 3.1956 - day1_loss: 0.4472 -
Epoch 6/20
8505/8505 [==============================] - 4s 497us/step - loss: 3.1843 - day1_loss: 0.4466 -
Epoch 7/20
8505/8505 [==============================] - 4s 497us/step - loss: 3.1777 - day1_loss: 0.4444 -
Epoch 8/20
8505/8505 [==============================] - 4s 498us/step - loss: 3.1700 - day1_loss: 0.4433 -
Epoch 9/20
8505/8505 [==============================] - 4s 490us/step - loss: 3.1637 - day1_loss: 0.4421 -
Epoch 10/20
8505/8505 [==============================] - 4s 501us/step - loss: 3.1580 - day1_loss: 0.4402 -
Epoch 11/20
8505/8505 [==============================] - 4s 526us/step - loss: 3.1547 - day1_loss: 0.4395 -
Epoch 12/20
8505/8505 [==============================] - 4s 525us/step - loss: 3.1536 - day1_loss: 0.4388 -
Epoch 13/20
8505/8505 [==============================] - 4s 510us/step - loss: 3.1517 - day1_loss: 0.4411 -
Epoch 14/20
8505/8505 [==============================] - 4s 513us/step - loss: 3.1435 - day1_loss: 0.4355 -
Epoch 15/20
8505/8505 [==============================] - 4s 511us/step - loss: 3.1453 - day1_loss: 0.4373 -
Epoch 16/20
8505/8505 [==============================] - 4s 507us/step - loss: 3.1471 - day1_loss: 0.4371 -
Epoch 17/20
8505/8505 [==============================] - 4s 525us/step - loss: 3.1452 - day1_loss: 0.4370 -
Epoch 18/20
8505/8505 [==============================] - 5s 548us/step - loss: 3.1473 - day1_loss: 0.4378 -
Epoch 19/20
8505/8505 [==============================] - 4s 527us/step - loss: 3.1456 - day1_loss: 0.4375 -
Epoch 20/20
8505/8505 [==============================] - 4s 512us/step - loss: 3.1463 - day1_loss: 0.4375 -
```

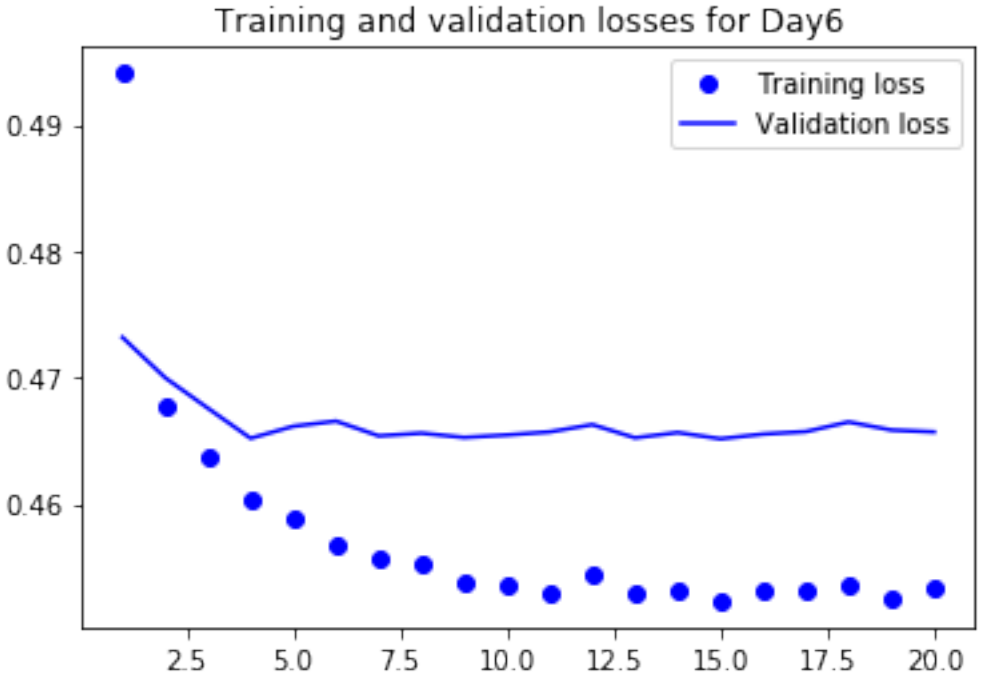

Training and validation losses for Day1
Training loss
Validation loss
0.49
0.48
0.47
0.46
0.45
0.44
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

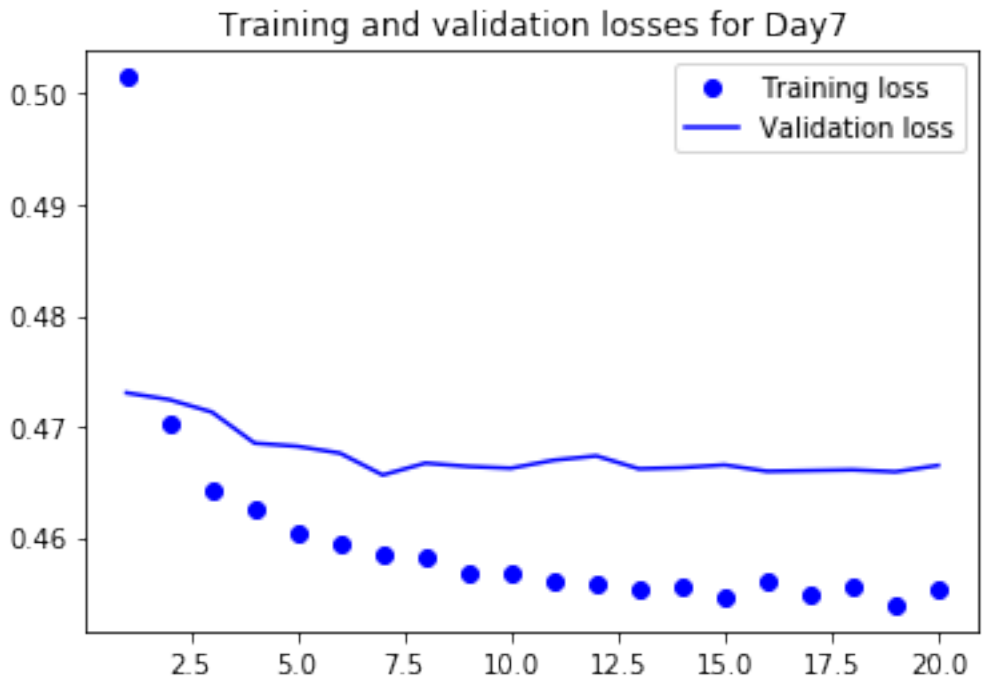

Training and validation losses for Day2
Training loss
Validation loss
0.49
0.48
0.47
0.46
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation losses for Day3
Training loss
Validation loss
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
0.49
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 Day5
Training loss
Validation loss
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 Day6
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




```
8504/8504 [==============================] - 1s 89us/step
normalized test_loss of  day 1 is 0.4129428357755028
unnormalized test_loss of  day 1 is 0.0963130859942905
normalized test_loss of  day 2 is 0.4313916723512863
unnormalized test_loss of  day 2 is 0.100616016641534
normalized test_loss of  day 3 is 0.4345839670662149
unnormalized test_loss of  day 3 is 0.10136057431092818
normalized test_loss of  day 4 is 0.43619179425661225
unnormalized test_loss of  day 4 is 0.10173557730174623
normalized test_loss of  day 5 is 0.4370540824709136
unnormalized test_loss of  day 5 is 0.10193669385285409
normalized test_loss of  day 6 is 0.4372039862881263
unnormalized test_loss of  day 6 is 0.10197165680168686
normalized test_loss of  day 7 is 0.4345880787493705
unnormalized test_loss of  day 7 is 0.10136153330297028
normalized MAE of base model for day  1  is  0.588833890647551
unnormalized MAE of base model for day  1  is  0.13733719109034753
normalized MAE of base model for day  2  is  0.5953143924718292
unnormalized MAE of base model for day  2  is  0.13884867664092185
normalized MAE of base model for day  3  is  0.5984563071322808
unnormalized MAE of base model for day  3  is  0.13958148387393876
normalized MAE of base model for day  4  is  0.5997326302290171
unnormalized MAE of base model for day  4  is  0.13987916821550533
normalized MAE of base model for day  5  is  0.600062411970935
unnormalized MAE of base model for day  5  is  0.13995608515053107
```

```
normalized MAE of base model for day  6  is  0.6010094802973719
unnormalized MAE of base model for day  6  is  0.14017697546576147
normalized MAE of base model for day  7  is  0.6018344685104254
unnormalized MAE of base model for day  7  is  0.14036939231822698
mean of normalized MAE of base model of week   is  0.5978919401799158
mean of unnormalized MAE of base model of week   is  0.13944985325074757
```