Model

February 1, 2021

[1]:
```
import tensorflow as tf
import numpy as np
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,LSTM,Dropout,AveragePooling3D
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
df= pd.read_csv('data.csv')
df.head()
```

[2]:

	Unnamed: 0	X1	X2	X3	X4	X5	X6	X7	X8	X9	…	X170	X171	\
0	X21.V1.791	135	190	229	223	192	125	55	-9	-33	…	-17	-15	
1	X15.V1.924	386	382	356	331	320	315	307	272	244	…	164	150	
2	X8.V1.1	-32	-39	-47	-37	-32	-36	-57	-73	-85	…	57	64	
3	X16.V1.60	-105	-101	-96	-92	-89	-95	-102	-100	-87	…	-82	-81	
4	X20.V1.54	-9	-65	-98	-102	-78	-48	-16	0	-21	…	4	2	

	X172	X173	X174	X175	X176	X177	X178	y
0	-31	-77	-103	-127	-116	-83	-51	4
1	146	152	157	156	154	143	129	1
2	48	19	-12	-30	-35	-35	-36	5
3	-80	-77	-85	-77	-72	-69	-65	5
4	-12	-32	-41	-65	-83	-89	-73	5

[5 rows x 180 columns]

[3]:
```
x=df.values
x=x[:,1:-1]
x = np.asarray(x).astype(np.float32)
```

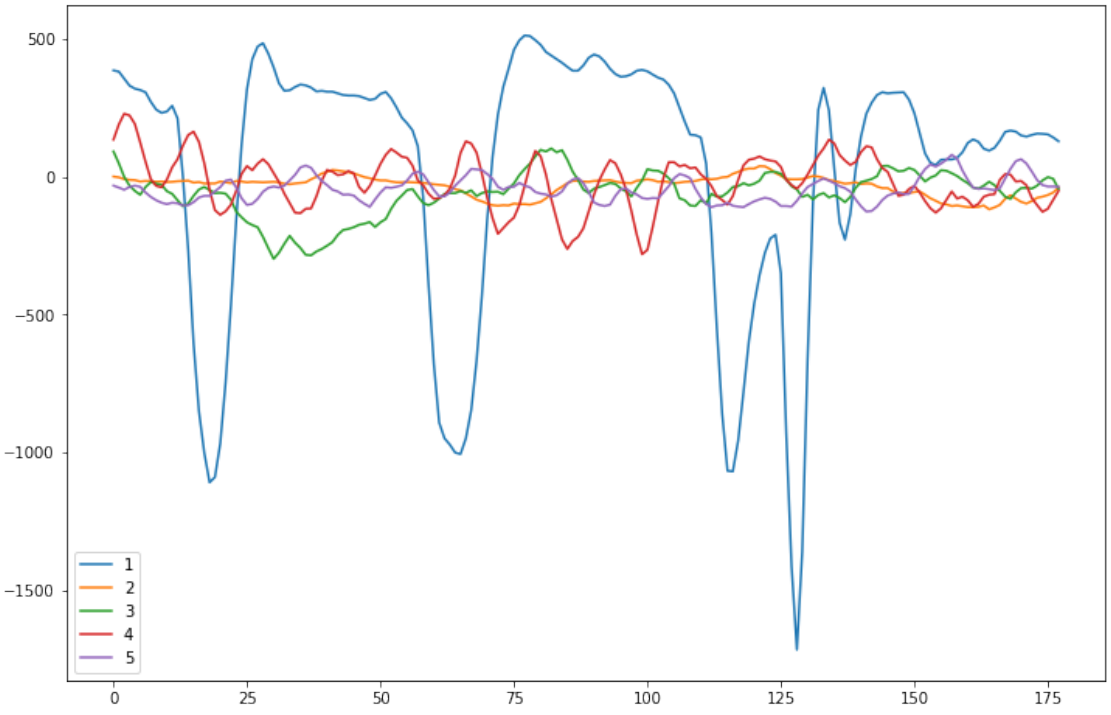
[4]:
```
from sklearn.model_selection import train_test_split

##taking 20% for test
y=np.array(df['y'])
y=np_utils.to_categorical(y)
print(y.shape)
```

```
print(x.shape)
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.
 ↪20,random_state=1)
```

```
(11500, 6)
(11500, 178)
```

[5]:
```
##corresponding to the 5 datasets ,plotting them all alongsize
plt.figure(figsize=(12,8))
plt.plot(x[1,:],label='1')
plt.plot(x[7,:],label='2')
plt.plot(x[12,:],label='3')
plt.plot(x[0,:],label='4')
plt.plot(x[2,:],label='5')
plt.legend()
plt.show()
```



[6]:
```
##makingg a neural network
model=Sequential()

model.add(Dense(256,input_shape=(45,)))
model.add(Activation('relu'))
model.add(Dense(128))
model.add(Activation('relu'))
```

```
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(8))
model.add(Activation('relu'))
model.add(Dense(5))
model.add(Activation('softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               11776
_________________________________________________________________
activation (Activation)      (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
activation_4 (Activation)    (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 32)                1056
_________________________________________________________________
activation_5 (Activation)    (None, 32)                0
```

```
_________________________________________________________________
dense_6 (Dense)              (None, 32)                1056
_________________________________________________________________
activation_6 (Activation)    (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 16)                528
_________________________________________________________________
activation_7 (Activation)    (None, 16)                0
_________________________________________________________________
dense_8 (Dense)              (None, 16)                272
_________________________________________________________________
activation_8 (Activation)    (None, 16)                0
_________________________________________________________________
dense_9 (Dense)              (None, 8)                 136
_________________________________________________________________
activation_9 (Activation)    (None, 8)                 0
_________________________________________________________________
dense_10 (Dense)             (None, 5)                 45
_________________________________________________________________
activation_10 (Activation)   (None, 5)                 0
=================================================================
Total params: 74,613
Trainable params: 74,613
Non-trainable params: 0
_________________________________________________________________
```

[7]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

[9]:
```
##batch size : 15 ,epochs : 50
hist = model.fit(((x_train[:,::4]-x_train.mean())/x_train.std()),y_train[:,1:
↪],15,50,verbose=2,validation_data=((x_test[:,::4]-x_test.mean())/x_test.
↪std(),y_test[:,1:]),shuffle=False)
```

```
Epoch 1/50
614/614 - 3s - loss: 0.4146 - accuracy: 0.4079 - val_loss: 0.3813 -
val_accuracy: 0.4770
Epoch 2/50
614/614 - 2s - loss: 0.3641 - accuracy: 0.4917 - val_loss: 0.3635 -
val_accuracy: 0.4848
Epoch 3/50
614/614 - 2s - loss: 0.3130 - accuracy: 0.5800 - val_loss: 0.2886 -
val_accuracy: 0.6252
Epoch 4/50
614/614 - 2s - loss: 0.2488 - accuracy: 0.6786 - val_loss: 0.2592 -
val_accuracy: 0.6678
Epoch 5/50
614/614 - 2s - loss: 0.2276 - accuracy: 0.7065 - val_loss: 0.2615 -
val_accuracy: 0.6648
```

```
Epoch 6/50
614/614 - 2s - loss: 0.2146 - accuracy: 0.7211 - val_loss: 0.2604 -
val_accuracy: 0.6804
Epoch 7/50
614/614 - 2s - loss: 0.2054 - accuracy: 0.7354 - val_loss: 0.2694 -
val_accuracy: 0.6652
Epoch 8/50
614/614 - 2s - loss: 0.1909 - accuracy: 0.7566 - val_loss: 0.2708 -
val_accuracy: 0.6813
Epoch 9/50
614/614 - 2s - loss: 0.1872 - accuracy: 0.7708 - val_loss: 0.2675 -
val_accuracy: 0.6848
Epoch 10/50
614/614 - 2s - loss: 0.1692 - accuracy: 0.7924 - val_loss: 0.2989 -
val_accuracy: 0.6900
Epoch 11/50
614/614 - 2s - loss: 0.1569 - accuracy: 0.8142 - val_loss: 0.2925 -
val_accuracy: 0.6874
Epoch 12/50
614/614 - 2s - loss: 0.1539 - accuracy: 0.8218 - val_loss: 0.3183 -
val_accuracy: 0.6891
Epoch 13/50
614/614 - 2s - loss: 0.1471 - accuracy: 0.8352 - val_loss: 0.3278 -
val_accuracy: 0.6883
Epoch 14/50
614/614 - 2s - loss: 0.1295 - accuracy: 0.8579 - val_loss: 0.3381 -
val_accuracy: 0.6857
Epoch 15/50
614/614 - 2s - loss: 0.1251 - accuracy: 0.8700 - val_loss: 0.2993 -
val_accuracy: 0.6830
Epoch 16/50
614/614 - 2s - loss: 0.1149 - accuracy: 0.8830 - val_loss: 0.3701 -
val_accuracy: 0.6922
Epoch 17/50
614/614 - 2s - loss: 0.1074 - accuracy: 0.8899 - val_loss: 0.3547 -
val_accuracy: 0.7009
Epoch 18/50
614/614 - 2s - loss: 0.0960 - accuracy: 0.9052 - val_loss: 0.4338 -
val_accuracy: 0.7009
Epoch 19/50
614/614 - 2s - loss: 0.0946 - accuracy: 0.9052 - val_loss: 0.3940 -
val_accuracy: 0.7017
Epoch 20/50
614/614 - 2s - loss: 0.0891 - accuracy: 0.9145 - val_loss: 0.3721 -
val_accuracy: 0.7061
Epoch 21/50
614/614 - 2s - loss: 0.0894 - accuracy: 0.9123 - val_loss: 0.4215 -
val_accuracy: 0.7035
```

```
Epoch 22/50
614/614 - 2s - loss: 0.0773 - accuracy: 0.9242 - val_loss: 0.4184 -
val_accuracy: 0.7165
Epoch 23/50
614/614 - 2s - loss: 0.0708 - accuracy: 0.9314 - val_loss: 0.4375 -
val_accuracy: 0.7057
Epoch 24/50
614/614 - 2s - loss: 0.0703 - accuracy: 0.9323 - val_loss: 0.4530 -
val_accuracy: 0.7043
Epoch 25/50
614/614 - 2s - loss: 0.0621 - accuracy: 0.9423 - val_loss: 0.4876 -
val_accuracy: 0.7091
Epoch 26/50
614/614 - 2s - loss: 0.0604 - accuracy: 0.9450 - val_loss: 0.5129 -
val_accuracy: 0.7065
Epoch 27/50
614/614 - 2s - loss: 0.0602 - accuracy: 0.9451 - val_loss: 0.4366 -
val_accuracy: 0.7139
Epoch 28/50
614/614 - 2s - loss: 0.0541 - accuracy: 0.9533 - val_loss: 0.4895 -
val_accuracy: 0.7087
Epoch 29/50
614/614 - 2s - loss: 0.0497 - accuracy: 0.9537 - val_loss: 0.5161 -
val_accuracy: 0.6996
Epoch 30/50
614/614 - 2s - loss: 0.0572 - accuracy: 0.9472 - val_loss: 0.5198 -
val_accuracy: 0.7043
Epoch 31/50
614/614 - 2s - loss: 0.0456 - accuracy: 0.9587 - val_loss: 0.5436 -
val_accuracy: 0.7052
Epoch 32/50
614/614 - 2s - loss: 0.0476 - accuracy: 0.9579 - val_loss: 0.5465 -
val_accuracy: 0.7174
Epoch 33/50
614/614 - 2s - loss: 0.0485 - accuracy: 0.9567 - val_loss: 0.4898 -
val_accuracy: 0.7026
Epoch 34/50
614/614 - 2s - loss: 0.0434 - accuracy: 0.9605 - val_loss: 0.5539 -
val_accuracy: 0.7026
Epoch 35/50
614/614 - 2s - loss: 0.0368 - accuracy: 0.9659 - val_loss: 0.5763 -
val_accuracy: 0.7252
Epoch 36/50
614/614 - 2s - loss: 0.0356 - accuracy: 0.9691 - val_loss: 0.5746 -
val_accuracy: 0.7117
Epoch 37/50
614/614 - 2s - loss: 0.0349 - accuracy: 0.9702 - val_loss: 0.5417 -
val_accuracy: 0.7235
```

```
Epoch 38/50
614/614 - 2s - loss: 0.0398 - accuracy: 0.9641 - val_loss: 0.5281 -
val_accuracy: 0.7191
Epoch 39/50
614/614 - 2s - loss: 0.0383 - accuracy: 0.9680 - val_loss: 0.5602 -
val_accuracy: 0.6996
Epoch 40/50
614/614 - 2s - loss: 0.0313 - accuracy: 0.9714 - val_loss: 0.6495 -
val_accuracy: 0.6965
Epoch 41/50
614/614 - 2s - loss: 0.0358 - accuracy: 0.9689 - val_loss: 0.5344 -
val_accuracy: 0.7048
Epoch 42/50
614/614 - 2s - loss: 0.0318 - accuracy: 0.9717 - val_loss: 0.5590 -
val_accuracy: 0.7122
Epoch 43/50
614/614 - 2s - loss: 0.0262 - accuracy: 0.9760 - val_loss: 0.6060 -
val_accuracy: 0.7143
Epoch 44/50
614/614 - 2s - loss: 0.0252 - accuracy: 0.9783 - val_loss: 0.6264 -
val_accuracy: 0.7048
Epoch 45/50
614/614 - 2s - loss: 0.0268 - accuracy: 0.9780 - val_loss: 0.6572 -
val_accuracy: 0.7030
Epoch 46/50
614/614 - 2s - loss: 0.0329 - accuracy: 0.9764 - val_loss: 0.5634 -
val_accuracy: 0.7096
Epoch 47/50
614/614 - 2s - loss: 0.0267 - accuracy: 0.9801 - val_loss: 0.5555 -
val_accuracy: 0.7196
Epoch 48/50
614/614 - 2s - loss: 0.0264 - accuracy: 0.9780 - val_loss: 0.5460 -
val_accuracy: 0.7265
Epoch 49/50
614/614 - 2s - loss: 0.0272 - accuracy: 0.9768 - val_loss: 0.5831 -
val_accuracy: 0.7013
Epoch 50/50
614/614 - 2s - loss: 0.0343 - accuracy: 0.9714 - val_loss: 0.6016 -
val_accuracy: 0.7152
```

[12]:
```
## now with using lstm
model = Sequential()
model.add(LSTM(56, input_shape=(45,1), return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(56))
model.add(Dropout(0.3))
model.add(Dense(20))
```

```
model.add(Activation('tanh'))
model.add(Dense(5))
model.add(Activation('softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 45, 56)            12992
_________________________________________________________________
dropout (Dropout)            (None, 45, 56)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 56)                25312
_________________________________________________________________
dropout_1 (Dropout)          (None, 56)                0
_________________________________________________________________
dense_11 (Dense)             (None, 20)                1140
_________________________________________________________________
activation_11 (Activation)   (None, 20)                0
_________________________________________________________________
dense_12 (Dense)             (None, 5)                 105
_________________________________________________________________
activation_12 (Activation)   (None, 5)                 0
=================================================================
Total params: 39,549
Trainable params: 39,549
Non-trainable params: 0
_________________________________________________________________
```

[13]:
```
model.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])
```

[16]:
```
hist2 = model.fit(((x_train[:,::4]-x_train.mean())/x_train.std()),y_train[:,1:
↪],validation_data=((x_test[:,::4]-x_test.mean())/x_test.std(),y_test[:,1:])
                ,epochs = 50, batch_size=15,shuffle=False
                )
```

```
Epoch 1/50
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-d0af242c9119> in <module>
```

```
----> 1 hist2 = model.fit(((x_train[:,::4]-x_train.mean())/x_train.
 ↪std()),y_train[:,1:],validation_data=((x_test[:,::4]-x_test.mean())/x_test.
 ↪std(),y_test[:,1:])
      2                 ,epochs = 50, batch_size=15,shuffle=False
      3                 )
      4


~/.local/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.py
 ↪in fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split,␣
 ↪validation_data, shuffle, class_weight, sample_weight, initial_epoch,␣
 ↪steps_per_epoch, validation_steps, validation_batch_size, validation_freq,␣
 ↪max_queue_size, workers, use_multiprocessing)
   1098                 _r=1):
   1099               callbacks.on_train_batch_begin(step)
-> 1100               tmp_logs = self.train_function(iterator)
   1101               if data_handler.should_sync:
   1102                 context.async_wait()

~/.local/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py in
 ↪__call__(self, *args, **kwds)
    826     tracing_count = self.experimental_get_tracing_count()
    827     with trace.Trace(self._name) as tm:
--> 828       result = self._call(*args, **kwds)
    829       compiler = "xla" if self._experimental_compile else "nonXla"
    830       new_tracing_count = self.experimental_get_tracing_count()

~/.local/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py in
 ↪_call(self, *args, **kwds)
    860       # In this case we have not created variables on the first call. S␣
 ↪we can
    861       # run the first trace but we should fail if variables are created
--> 862       results = self._stateful_fn(*args, **kwds)
    863       if self._created_variables:
    864         raise ValueError("Creating variables on a non-first call to a␣
 ↪function"

~/.local/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪__call__(self, *args, **kwargs)
   2939     with self._lock:
   2940       (graph_function,
-> 2941        filtered_flat_args) = self._maybe_define_function(args, kwargs)
   2942     return graph_function._call_flat(
   2943         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access

~/.local/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪_maybe_define_function(self, args, kwargs)
   3355               self.input_signature is None and
```

```
   3356               call_context_key in self._function_cache.missed):
-> 3357             return self._define_function_with_shape_relaxation(
   3358                 args, kwargs, flat_args, filtered_flat_args,␣
 ↪cache_key_context)
   3359

~/.local/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪_define_function_with_shape_relaxation(self, args, kwargs, flat_args,␣
 ↪filtered_flat_args, cache_key_context)
   3277           expand_composites=True)
   3278
-> 3279     graph_function = self._create_graph_function(
   3280         args, kwargs, override_flat_arg_shapes=relaxed_arg_shapes)
   3281     self._function_cache.arg_relaxed[rank_only_cache_key] =␣
 ↪graph_function

~/.local/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪_create_graph_function(self, args, kwargs, override_flat_arg_shapes)
   3194     arg_names = base_arg_names + missing_arg_names
   3195     graph_function = ConcreteFunction(
-> 3196         func_graph_module.func_graph_from_py_func(
   3197             self._name,
   3198             self._python_function,

~/.local/lib/python3.8/site-packages/tensorflow/python/framework/func_graph.py␣
 ↪in func_graph_from_py_func(name, python_func, args, kwargs, signature,␣
 ↪func_graph, autograph, autograph_options, add_control_dependencies, arg_names␣
 ↪op_return_value, collections, capture_by_value, override_flat_arg_shapes)
    988         _, original_func = tf_decorator.unwrap(python_func)
    989
--> 990       func_outputs = python_func(*func_args, **func_kwargs)
    991
    992       # invariant: `func_outputs` contains only Tensors,␣
 ↪CompositeTensors,

~/.local/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py in␣
 ↪wrapped_fn(*args, **kwds)
    632             xla_context.Exit()
    633         else:
--> 634           out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    635         return out
    636

~/.local/lib/python3.8/site-packages/tensorflow/python/framework/func_graph.py␣
 ↪in wrapper(*args, **kwargs)
    975           except Exception as e:  # pylint:disable=broad-except
    976             if hasattr(e, "ag_error_metadata"):
```

```
--> 977               raise e.ag_error_metadata.to_exception(e)
    978             else:
    979               raise

ValueError: in user code:

    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/keras/
↪engine/training.py:805 train_function  *
        return step_function(self, iterator)
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/keras/
↪engine/training.py:795 step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/distribut/
↪distribute_lib.py:1259 run
        return self._extended.call_for_each_replica(fn, args=args, kwargs=kwarg )
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/distribut/
↪distribute_lib.py:2730 call_for_each_replica
        return self._call_for_each_replica(fn, args, kwargs)
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/distribut/
↪distribute_lib.py:3417 _call_for_each_replica
        return fn(*args, **kwargs)
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/keras/
↪engine/training.py:788 run_step  **
        outputs = model.train_step(data)
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/keras/
↪engine/training.py:754 train_step
        y_pred = self(x, training=True)
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/keras/
↪engine/base_layer.py:998 __call__
        input_spec.assert_input_compatibility(self.input_spec, inputs, self.nam )
    /home/sagnik/.local/lib/python3.8/site-packages/tensorflow/python/keras/
↪engine/input_spec.py:219 assert_input_compatibility
        raise ValueError('Input ' + str(input_index) + ' of layer ' +

    ValueError: Input 0 of layer sequential_1 is incompatible with the layer:␣
↪expected ndim=3, found ndim=2. Full shape received: (None, 45)
```

[ ]: