Usman_test_train2_New

March 31, 2022

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
```

[3]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout,Activation

from keras.layers.convolutional import Conv1D
from keras import backend as K
from keras.layers.core import Lambda
from keras.layers.convolutional import MaxPooling1D
from keras.layers.embeddings import Embedding

import keras
from keras.utils import np_utils
```

[4]:
```
from sklearn.feature_selection import VarianceThreshold
```

[5]:
```
dataset = pd.read_csv('Usman_test_train_IoT/Combined_Data_IoT.csv')
```

[6]:
```
dataset.columns
```

[6]:
```
Index(['ts', 'date', 'time', 'fridge_temperature', 'temp_condition', 'label',
       'type', 'door_state', 'sphone_signal', 'latitude', 'longitude',
       'FC1_Read_Input_Register', 'FC2_Read_Discrete_Value',
       'FC3_Read_Holding_Register', 'FC4_Read_Coil', 'motion_status',
       'light_status', 'current_temperature', 'thermostat_status',
       'temperature', 'pressure', 'humidity'],
      dtype='object')
```

[7]:
```
Data = dataset.drop(['ts', 'date', 'time'], axis =1)
```

[8]:
```
Data['label'].value_counts()
```

[8]:
```
0    245000
1    156119
Name: label, dtype: int64
```

[9]:
```
Data['type'].value_counts()
```

[9]:
```
normal        245000
backdoor       35000
injection      35000
password       35000
ddos           25000
ransomware     16030
xss             6116
scanning        3973
Name: type, dtype: int64
```

[10]:
```
Data.isna().sum().sum()
```

[10]:
```
0
```

[11]:
```
Data.head()
```

[11]:

	fridge_temperature	temp_condition	label	type	door_state	sphone_signal	latitude	longitude	FC1_Read_Input_Register	FC2_Read_Discrete_Value
0	9.00	1	1	ddos	0	0	4.514077	14.421946	32450	32708
1	9.25	1	1	ddos	0	0	4.514077	14.421946	32450	32708
2	12.65	1	1	ddos	0	0	4.514077	14.421946	32450	32708
3	4.65	0	1	ddos	0	0	4.514077	14.421946	32450	32708
4	12.65	1	1	ddos	0	0	4.514077	14.421946	32450	32708

```
   FC3_Read_Holding_Register  FC4_Read_Coil  motion_status  light_status  \
0                      32035          32728              0             0
1                      32035          32728              0             0
2                      32035          32728              0             0
3                      32035          32728              0             0
4                      32035          32728              0             0

   current_temperature  thermostat_status  temperature  pressure   humidity
0            28.442693                  1    35.773605     1.035  46.343618
1            28.442693                  1    35.773605     1.035  46.343618
2            28.442693                  1    35.773605     1.035  46.343618
3            28.442693                  1    35.773605     1.035  46.343618
4            28.442693                  1    35.773605     1.035  46.343618
```

1 Model Development (Binary)

[12]:
```
# from ctgan import CTGANSynthesizer
# ctgan = CTGANSynthesizer(epochs=10)
# ctgan.fit(Data.drop(['type'], axis = 1), ['label'])
```

[13]:
```
# GAN_IOT = ctgan.sample(600000)
# GAN_IOT
```

[14]:
```
# GAN_IOT.to_csv("GAN_IOT_10Epoches.csv", index=False)
```

[15]:
```
samples = pd.read_csv('GAN_IOT_10Epoches.csv')
```

[16]:
```
samples['label'].value_counts()
```

[16]:
```
0    427817
1    172183
Name: label, dtype: int64
```

[17]:
```
X_G = samples.drop(['label'], axis = 1)
X_G.shape
```

[17]:
```
(600000, 17)
```

[18]:
```
y_G = samples['label']
y_G = y_G.values
y_G.shape
```

[18]:
```
(600000,)
```

[19]:
```
sel = VarianceThreshold(threshold=(.8 * (1 - .8))) #don't change these values
X_T = sel.fit(X_G)
```

[20]:
```
sel_cols = X_T.get_support(indices=True)
X_VT = X_G.iloc[:,sel_cols]
X_VT
```

[20]:

```
        fridge_temperature  temp_condition  latitude  longitude  \
0                10.041658               0  4.550903  14.438127
1                 6.699955               1  4.531272  75.202959
2                 6.700362               1  4.517152  14.403387
3                 6.702047               0  4.524009  14.439772
4                 6.701300               1  4.526589  14.442005
…                    …              …        …         …
599995            6.701781               0  4.497448  14.400577
599996            6.700457               0  4.511919  14.457472
599997            6.700144               0  1.381538  10.911842
599998            6.700591               0  4.525477  14.401991
599999            6.701271               0  4.512456  14.453269

        FC1_Read_Input_Register  FC2_Read_Discrete_Value  \
0                         32454                    32713
1                         32451                    32705
2                         32454                    32703
3                         32451                    32703
4                         32449                    32707
…                         …                      …
599995                    32454                    32707
599996                    55798                     3071
599997                    32451                    32705
599998                    32448                    32709
599999                    32455                    32702

        FC3_Read_Holding_Register  FC4_Read_Coil  current_temperature  \
0                           32040          32728            28.442293
1                           32039          32724            28.737284
2                           32044          32721            28.442653
3                           32043          32726            28.442530
4                           32038          32725            27.159283
…                           …            …                  …
599995                      32034          32726            28.442123
599996                      24817          26585            28.441697
599997                      32041          32719            28.442684
599998                      32038          32717            28.442219
599999                      32038          32715            28.442448

        thermostat_status  temperature  pressure   humidity
0                       1    35.770101  1.035372  46.348176
1                       0    35.769628  1.035715  46.355070
2                       0    35.771778  1.035797  46.359988
```

```
3                       0    35.771197  1.034263  46.343268
4                       0    35.772592  1.036184  38.912467
…                     …        …        …         …
599995                  0    35.771638  1.035189  46.349511
599996                  1    35.770840  1.035442  46.345211
599997                  1    35.772983  1.035264  46.349470
599998                  1    35.772159  1.035277  46.350916
599999                  1    32.364628  1.943538  41.126065

[600000 rows x 13 columns]
```

[21]:
```
X_VT.shape
```

[21]:
```
(600000, 13)
```

[22]:
```
scaler = MinMaxScaler()
X_M = scaler.fit_transform(X_VT)
print(X_M)
```

```
[[0.70550429 0.         0.03402255 … 0.53001883 0.54297187 0.43603677]
 [0.44952356 1.         0.03388342 … 0.53000224 0.54300734 0.4361097 ]
 [0.44955469 1.         0.03378335 … 0.53007757 0.54301577 0.43616172]
 …
 [0.44953801 0.         0.01156011 … 0.53011979 0.54296081 0.43605046]
 [0.44957226 0.         0.03384235 … 0.53009091 0.54296213 0.43606575]
 [0.44962436 0.         0.03375006 … 0.41070494 0.63669728 0.38079804]]
```

[23]:
```
# (Smote -> varience threshold -> min max scaler)
trainX, testX, trainY, testY = train_test_split(X_M, y_G, test_size = 0.30,␣
 ↪random_state = 5)
```

[24]:
```
print(X_M.shape)
print(y_G.shape)
print(trainX.shape)
print(trainY.shape)
print(testX.shape)
print(testY.shape)
```

```
(600000, 13)
(600000,)
(420000, 13)
(420000,)
(180000, 13)
(180000,)
```

1.0.1 LR

[29]:
```
#Test Train split
lr = LogisticRegression() #for binary
lr.fit(trainX, trainY)
y_predict = lr.predict(testX)
print(accuracy_score(testY, y_predict))
print(classification_report(testY, y_predict))
```

```
0.7222277777777778
              precision    recall  f1-score   support

           0       0.73      0.97      0.83    128414
           1       0.58      0.11      0.19     51586

    accuracy                           0.72    180000
   macro avg       0.65      0.54      0.51    180000
weighted avg       0.69      0.72      0.65    180000
```

1.0.2 LDA

[30]:
```
#Test Train split
lda = LinearDiscriminantAnalysis()
lda.fit(trainX, trainY)
y_predict = lda.predict(testX)
print(accuracy_score(testY, y_predict))
print(classification_report(testY, y_predict))
```

```
0.7218222222222223
              precision    recall  f1-score   support

           0       0.73      0.97      0.83    128414
           1       0.57      0.12      0.19     51586

    accuracy                           0.72    180000
   macro avg       0.65      0.54      0.51    180000
weighted avg       0.69      0.72      0.65    180000
```

1.0.3 KNN

[ ]:
```
#Test Train split
#default parameters used in base paper (n_neighbors, default=5) (p, default=2
 ↪for Euclidean Distance)
knn = KNeighborsClassifier()
knn.fit(trainX, trainY)
y_predict = knn.predict(testX)
print(accuracy_score(testY, y_predict))
```

```
print(classification_report(testY, y_predict))
```

1.0.4 RF

[25]:
```
#Test Train split
rf = RandomForestClassifier(n_estimators=10,criterion='gini')
rf.fit(trainX, trainY)
y_predict1 = rf.predict(testX)
print(accuracy_score(testY, y_predict1))
print(classification_report(testY, y_predict1))
```

```
0.7392944444444445
              precision    recall  f1-score   support

           0       0.76      0.93      0.84    128414
           1       0.60      0.27      0.37     51586

    accuracy                           0.74    180000
   macro avg       0.68      0.60      0.60    180000
weighted avg       0.71      0.74      0.70    180000
```

1.0.5 DT (CART)

[32]:
```
#Test Train Split
dt = DecisionTreeClassifier(criterion='gini')
dt.fit(trainX, trainY)
y_predict1 = dt.predict(testX)
print(accuracy_score(testY, y_predict1))
print(classification_report(testY, y_predict1))
```

```
0.6480055555555555
              precision    recall  f1-score   support

           0       0.76      0.74      0.75    128414
           1       0.39      0.41      0.40     51586

    accuracy                           0.65    180000
   macro avg       0.58      0.58      0.58    180000
weighted avg       0.65      0.65      0.65    180000
```

1.0.6 NB

[33]:
```
#Test Train Split
nb = GaussianNB()
nb.fit(trainX, trainY)
y_predict1 = nb.predict(testX)
```

```
print(accuracy_score(testY, y_predict1))
print(classification_report(testY, y_predict1))
```

```
0.7011166666666667
              precision    recall  f1-score   support

           0       0.75      0.87      0.81    128414
           1       0.46      0.28      0.35     51586

    accuracy                           0.70    180000
   macro avg       0.61      0.58      0.58    180000
weighted avg       0.67      0.70      0.68    180000
```

1.0.7 SVM

[ ]:
```
#Test Train Split
svclassifier = SVC(kernel='rbf', gamma='auto')
svclassifier.fit(trainX, trainY)
y_predict = svclassifier.predict(testX)
print(accuracy_score(testY, y_predict))
print(classification_report(testY, y_predict))
```

1.0.8 LSTM

[26]:
```
X_G.shape
```

[26]: (600000, 17)

[29]:
```
X_GV = X_G.values
```

[30]:
```
X1 = X_GV.reshape((-1, 1, 17))
```

[33]:
```
X1.shape
```

[33]: (600000, 1, 17)

[34]:
```
(trainX, testX, trainY, testY) = train_test_split(X1, y_G, test_size = 0.30,␣
↪random_state = 5)
```

[35]:
```
# 1 LSTM Layer (input), 3 Dense Hidden Layers
model = Sequential()
model.add(LSTM(17, input_shape=(1, 17), activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(100, activation='tanh'))
model.add(Dropout(0.2))
```

```
model.add(Dense(64, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))
```

```
Epoch 1/35
6563/6563 - 41s - loss: 0.5999 - accuracy: 0.7127 - val_loss: 0.5952 -
val_accuracy: 0.7134
Epoch 2/35
6563/6563 - 40s - loss: 0.5984 - accuracy: 0.7129 - val_loss: 0.5953 -
val_accuracy: 0.7134
Epoch 3/35
6563/6563 - 33s - loss: 0.5972 - accuracy: 0.7129 - val_loss: 0.5954 -
val_accuracy: 0.7134
Epoch 4/35
6563/6563 - 35s - loss: 0.5975 - accuracy: 0.7129 - val_loss: 0.5983 -
val_accuracy: 0.7134
Epoch 5/35
6563/6563 - 33s - loss: 0.5971 - accuracy: 0.7129 - val_loss: 0.5960 -
val_accuracy: 0.7134
Epoch 6/35
6563/6563 - 29s - loss: 0.5972 - accuracy: 0.7129 - val_loss: 0.5960 -
val_accuracy: 0.7134
Epoch 7/35
6563/6563 - 32s - loss: 0.5967 - accuracy: 0.7129 - val_loss: 0.5939 -
val_accuracy: 0.7134
Epoch 8/35
6563/6563 - 35s - loss: 0.5964 - accuracy: 0.7129 - val_loss: 0.5945 -
val_accuracy: 0.7134
Epoch 9/35
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-35-520b0317fb91> in <module>
     11 model.add(Dense(1, activation='sigmoid'))
     12 model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
---> 13 model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))

C:\ProgramData\Anaconda3\lib\site-packages\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)
```

```
   1182                 _r=1):
   1183               callbacks.on_train_batch_begin(step)
-> 1184               tmp_logs = self.train_function(iterator)
   1185               if data_handler.should_sync:
   1186                 context.async_wait()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in __call__(self, *args, **kwds)
    883
    884       with OptionalXlaContext(self._jit_compile):
--> 885         result = self._call(*args, **kwds)
    886
    887       new_tracing_count = self.experimental_get_tracing_count()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in _call(self, *args, **kwds)
    915       # In this case we have created variables on the first call, so we
 ↪run the
    916       # defunned version which is guaranteed to never create variables.
--> 917       return self._stateless_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    918     elif self._stateful_fn is not None:
    919       # Release the lock early so that multiple threads can perform the
 ↪call

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in __call__(self, *args, **kwargs)
   3037       (graph_function,
   3038        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 3039     return graph_function._call_flat(
   3040         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access
   3041

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in _call_flat(self, args, captured_inputs, cancellation_manager)
   1961         and executing_eagerly):
   1962       # No tape is watching; skip to running the function.
-> 1963       return self._build_call_outputs(self._inference_function.call(
   1964           ctx, args, cancellation_manager=cancellation_manager))
   1965     forward_backward = self._select_forward_and_backward_functions(

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in call(self, ctx, args, cancellation_manager)
    589       with _InterpolateFunctionError(self):
    590         if cancellation_manager is None:
```

```
--> 591           outputs = execute.execute(
    592               str(self.signature.name),
    593               num_outputs=self._num_outputs,

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\execute.py i
 ↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,
 ↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

[56]:
```
loss, accuracy = model.evaluate(testX, testY)
print("Loss:" + str(loss))
print("Accuracy:" + str(accuracy))
```

```
2507/2507 [==============================] - 4s 2ms/step - loss: 0.5779 -
accuracy: 0.6798
Loss:0.5779118537902832
Accuracy:0.679796576499939
```

[57]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 17)                2380
_________________________________________________________________
dropout_4 (Dropout)          (None, 17)                0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               2304
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               12900
_________________________________________________________________
dropout_6 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                6464
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 65
```

```
=================================================================
Total params: 24,113
Trainable params: 24,113
Non-trainable params: 0
_________________________________________________________________
```

1.0.9 CNN (ConvID)

[36]:
```
X_G.shape
```

[36]:
```
(600000, 17)
```

[37]:
```
X_GV = X_G.values
```

[38]:
```
X1 = X_GV.reshape((-1, 1, 17))
```

[39]:
```
X1.shape
```

[39]:
```
(600000, 1, 17)
```

[40]:
```
(trainX, testX, trainY, testY) = train_test_split(X1, y_G, test_size = 0.30,␣
 ↪random_state = 5)
```

[41]:
```
nb_filter = 250
filter_length = 3

model = Sequential()
model.add(Conv1D(filters=nb_filter, kernel_size=filter_length, padding='same',␣
 ↪activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(100, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))
```

```
Epoch 1/35
6563/6563 - 41s - loss: 0.6038 - accuracy: 0.7116 - val_loss: 0.6011 -
val_accuracy: 0.7134
Epoch 2/35
6563/6563 - 41s - loss: 0.6008 - accuracy: 0.7129 - val_loss: 0.5991 -
val_accuracy: 0.7134
```

```
Epoch 3/35
6563/6563 - 37s - loss: 0.6007 - accuracy: 0.7129 - val_loss: 0.6004 -
val_accuracy: 0.7134
Epoch 4/35
6563/6563 - 37s - loss: 0.6007 - accuracy: 0.7129 - val_loss: 0.6029 -
val_accuracy: 0.7134
Epoch 5/35
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-41-248afc4dcd16> in <module>
     13 model.add(Dense(1, activation='sigmoid'))
     14 model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
---> 15 model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))

C:\ProgramData\Anaconda3\lib\site-packages\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)
   1182                 _r=1):
   1183               callbacks.on_train_batch_begin(step)
-> 1184               tmp_logs = self.train_function(iterator)
   1185               if data_handler.should_sync:
   1186                 context.async_wait()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in __call__(self, *args, **kwds)
    883
    884       with OptionalXlaContext(self._jit_compile):
--> 885         result = self._call(*args, **kwds)
    886
    887       new_tracing_count = self.experimental_get_tracing_count()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in _call(self, *args, **kwds)
    915       # In this case we have created variables on the first call, so we␣
 ↪run the
    916       # defunned version which is guaranteed to never create variables.
--> 917       return self._stateless_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    918     elif self._stateful_fn is not None:
    919       # Release the lock early so that multiple threads can perform the␣
 ↪call
```

```
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
↪in __call__(self, *args, **kwargs)
   3037       (graph_function,
   3038        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 3039     return graph_function._call_flat(
   3040         filtered_flat_args, captured_inputs=graph_function.
↪captured_inputs)  # pylint: disable=protected-access
   3041

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
↪in _call_flat(self, args, captured_inputs, cancellation_manager)
   1961         and executing_eagerly):
   1962       # No tape is watching; skip to running the function.
-> 1963       return self._build_call_outputs(self._inference_function.call(
   1964           ctx, args, cancellation_manager=cancellation_manager))
   1965     forward_backward = self._select_forward_and_backward_functions(

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
↪in call(self, ctx, args, cancellation_manager)
    589       with _InterpolateFunctionError(self):
    590         if cancellation_manager is None:
--> 591           outputs = execute.execute(
    592               str(self.signature.name),
    593               num_outputs=self._num_outputs,

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\execute.py i␣
↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

[ ]:
```
loss, accuracy = model.evaluate(testX, testY)
print("Loss:" + str(loss))
print("Accuracy:" + str(accuracy))
```

1.0.10 Conv-LSTM

[42]:
```
X_G.shape
```

[42]:
```
(600000, 17)
```

[43]:
```
X_GV = X_G.values
```

[44]:
```
X1 = X_GV.reshape((-1, 1, 17))
```

[45]:
```
X1.shape
```

[45]: (600000, 1, 17)

[46]:
```
(trainX, testX, trainY, testY) = train_test_split(X1, y_G, test_size = 0.30,␣
↪random_state = 5)
```

[47]:
```
nb_filter = 250
filter_length = 3

model = Sequential()
model.add(Conv1D(filters=nb_filter, kernel_size=filter_length, padding='same',␣
↪activation='tanh'))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=1))
model.add(Dropout(0.2))
model.add(LSTM(17))
model.add(Dropout(0.2))
model.add(Dense(128, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(100, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=["accuracy"])
model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
↪validation_data=(testX, testY))
```

```
Epoch 1/35
6563/6563 - 61s - loss: 0.5997 - accuracy: 0.7128 - val_loss: 0.5978 -
val_accuracy: 0.7134
Epoch 2/35
6563/6563 - 51s - loss: 0.5988 - accuracy: 0.7129 - val_loss: 0.5975 -
val_accuracy: 0.7134
Epoch 3/35
6563/6563 - 50s - loss: 0.5980 - accuracy: 0.7129 - val_loss: 0.5965 -
val_accuracy: 0.7134
Epoch 4/35
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-47-d0a6813e3fce> in <module>
```

```
     17 model.add(Dense(1, activation='sigmoid'))
     18 model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
---> 19 model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))

C:\ProgramData\Anaconda3\lib\site-packages\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)
   1182                 _r=1):
   1183               callbacks.on_train_batch_begin(step)
-> 1184               tmp_logs = self.train_function(iterator)
   1185               if data_handler.should_sync:
   1186                 context.async_wait()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in __call__(self, *args, **kwds)
    883
    884       with OptionalXlaContext(self._jit_compile):
--> 885         result = self._call(*args, **kwds)
    886
    887       new_tracing_count = self.experimental_get_tracing_count()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in _call(self, *args, **kwds)
    915       # In this case we have created variables on the first call, so we
 ↪run the
    916       # defunned version which is guaranteed to never create variables.
--> 917       return self._stateless_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    918     elif self._stateful_fn is not None:
    919       # Release the lock early so that multiple threads can perform the
 ↪call

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in __call__(self, *args, **kwargs)
   3037       (graph_function,
   3038        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 3039     return graph_function._call_flat(
   3040         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access
   3041

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in _call_flat(self, args, captured_inputs, cancellation_manager)
   1961         and executing_eagerly):
```

```
   1962       # No tape is watching; skip to running the function.
-> 1963       return self._build_call_outputs(self._inference_function.call(

   1964           ctx, args, cancellation_manager=cancellation_manager))
   1965     forward_backward = self._select_forward_and_backward_functions(

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in call(self, ctx, args, cancellation_manager)
    589       with _InterpolateFunctionError(self):
    590         if cancellation_manager is None:
--> 591           outputs = execute.execute(

    592               str(self.signature.name),
    593               num_outputs=self._num_outputs,

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\execute.py i␣
 ↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
 ↪op_name,

     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

[ ]:
```
loss, accuracy = model.evaluate(testX, testY)
print("Loss:" + str(loss))
print("Accuracy:" + str(accuracy))
```

[ ]:
```
nb_filter = 250
filter_length = 3

model = Sequential()
model.add(Conv1D(filters=nb_filter, kernel_size=filter_length, padding='same',␣
 ↪activation='tanh'))
model.add(MaxPooling1D(pool_size=1))
model.add(Conv1D(filters=nb_filter, kernel_size=filter_length, padding='same',␣
 ↪activation='tanh'))
model.add(MaxPooling1D(pool_size=1))
model.add(Activation('softmax'))
model.add(LSTM(17, return_sequences=True))
model.add(LSTM(17, return_sequences=True))
model.add(Dense(128, activation='tanh'))
model.add(Dense(100, activation='tanh'))
model.add(Dense(64, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
model.fit(trainX, trainY, epochs=500, batch_size=100, verbose=2,␣
 ↪validation_data=(testX, testY))
```

[ ]:
```
loss, accuracy = model.evaluate(testX, testY)
print("Loss:" + str(loss))
print("Accuracy:" + str(accuracy))
```

2 Model Development (Multi Class)

[98]:
```
XM = Data.drop(['label', 'type','latitude','longitude'], axis = 1)
XM.shape
```

[98]:
```
(401119, 15)
```

[99]:
```
yM = Data['type']
yM = yM.values
yM.shape
```

[99]:
```
(401119,)
```

[64]:
```
# from ctgan import CTGANSynthesizer
# ctgan = CTGANSynthesizer(epochs=10)
# ctgan.fit(Data.drop(['label'], axis = 1), ['type'])
```

[65]:
```
# GAN_IOT_M = ctgan.sample(600000)
# GAN_IOT_M
```

[51]:
```
GAN_IOT_M.to_csv("GAN_IOT_M_10Epoches.csv", index=False)
```

[52]:
```
samples_M = pd.read_csv('GAN_IOT_M_10Epoches.csv')
```

[53]:
```
samples_M['type'].value_counts()
```

[53]:
```
backdoor      283393
xss            59076
password       54409
normal         46782
ddos           46552
injection      43116
scanning       33949
ransomware     32723
Name: type, dtype: int64
```

[54]:
```
# XM = samples_M.drop(['type'], axis = 1)
# XM.shape
```

[54]: (600000, 17)

[55]:
```
# yM = samples_M['type']
# yM = yM.values
# yM.shape
```

[55]: (600000,)

[66]:
```
# from imblearn.over_sampling import SMOTE
# smote = SMOTE()
# from collections import Counter
```

[84]:
```
# XM_S = Data.drop(['label', 'type'], axis = 1)
# XM_S.shape
```

[85]:
```
# yM_S = Data['type']
# yM_S = yM_S.values
# yM_S.shape
```

[86]:
```
# XM, yM = smote.fit_resample(XM_S,yM_S)
```

[87]:
```
# print("Before Smote :", Counter(yM_S))
# print("After Smote :", Counter(yM))
```

[88]:
```
# XM.shape
```

[100]:
```
sel = VarianceThreshold(threshold=(.8 * (1 - .8))) #don't change these values
X_T = sel.fit(XM)
```

[101]:
```
sel_cols = X_T.get_support(indices=True)
X_VT = XM.iloc[:,sel_cols]
X_VT
```

[101]:

	fridge_temperature	FC1_Read_Input_Register	FC2_Read_Discrete_Value	\
0	9.00	32450	32708	
1	9.25	32450	32708	
2	12.65	32450	32708	
3	4.65	32450	32708	
4	12.65	32450	32708	
…	…	…	…	
401114	6.70	32450	32708	
401115	6.70	32450	32708	
401116	6.70	32450	32708	
401117	6.70	32450	32708	
401118	6.70	32450	32708	

	FC3_Read_Holding_Register	FC4_Read_Coil	current_temperature	\
0	32035	32728	28.442693	

```
1                          32035          32728          28.442693
2                          32035          32728          28.442693
3                          32035          32728          28.442693
4                          32035          32728          28.442693
…                          …              …              …
401114                     32035          32728          28.442693
401115                     32035          32728          28.442693
401116                     32035          32728          28.442693
401117                     32035          32728          28.442693
401118                     32035          32728          28.442693

        temperature  pressure   humidity
0         35.773605  1.035000  46.343618
1         35.773605  1.035000  46.343618
2         35.773605  1.035000  46.343618
3         35.773605  1.035000  46.343618
4         35.773605  1.035000  46.343618
…               …         …         …
401114    32.799434  2.204924  37.024913
401115    29.453781 -2.030547  90.297894
401116    47.185992  0.872942  37.687701
401117    43.097037  3.168207  93.647950
401118    32.489751  2.204924  37.024913

[401119 rows x 9 columns]
```

[102]:
```
X_VT.shape
```

[102]:
```
(401119, 9)
```

[103]:
```
scaler = MinMaxScaler()
X_M = scaler.fit_transform(X_VT)
print(X_M)
```

```
[[0.61538462 0.49521571 0.49909209 … 0.51730735 0.53355618 0.46251056]
 [0.63461538 0.49521571 0.49909209 … 0.51730735 0.53355618 0.46251056]
 [0.89615385 0.49521571 0.49909209 … 0.51730735 0.53355618 0.46251056]
 …
 [0.43846154 0.49521571 0.49909209 … 0.90452247 0.52695586 0.3756012 ]
 [0.43846154 0.49521571 0.49909209 … 0.76578681 0.62043775 0.93746771]
 [0.43846154 0.49521571 0.49909209 … 0.40588822 0.58120503 0.3689465 ]]
```

[104]:
```
(trainX, testX, trainY, testY) = train_test_split(X_M, yM, test_size = 0.30,␣
↪random_state = 5)
```

2.0.1 LR

[105]:
```
#Test Train split
lr = LogisticRegression(multi_class='ovr') #for multiclass
lr.fit(trainX, trainY)
y_predict = lr.predict(testX)
print(accuracy_score(testY, y_predict))
print(classification_report(testY, y_predict))
```

```
0.610474006116208

C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

    backdoor       0.00      0.00      0.00     10449
        ddos       1.00      0.00      0.00      7678
   injection       0.00      0.00      0.00     10462
      normal       0.61      1.00      0.76     73454
    password       0.00      0.00      0.00     10490
  ransomware       0.00      0.00      0.00      4743
    scanning       0.00      0.00      0.00      1212
         xss       0.00      0.00      0.00      1848

    accuracy                           0.61    120336
   macro avg       0.20      0.13      0.10    120336
weighted avg       0.44      0.61      0.46    120336


C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2.0.2 LDA

[106]:
```
#Test Train split
lda = LinearDiscriminantAnalysis()
lda.fit(trainX, trainY)
y_predict = lda.predict(testX)
```

```
print(accuracy_score(testY, y_predict))
print(classification_report(testY, y_predict))
```

```
0.6130002659220848
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

	precision	recall	f1-score	support
backdoor	0.00	0.00	0.00	10449
ddos	0.38	0.00	0.00	7678
injection	0.00	0.00	0.00	10462
normal	0.62	1.00	0.76	73454
password	0.51	0.03	0.05	10490
ransomware	0.00	0.00	0.00	4743
scanning	0.15	0.08	0.11	1212
xss	0.00	0.00	0.00	1848
accuracy			0.61	120336
macro avg	0.21	0.14	0.12	120336
weighted avg	0.45	0.61	0.47	120336

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2.0.3 KNN

[44]:
```
#Test Train split
knn = KNeighborsClassifier()
knn.fit(trainX, trainY)
y_predict = knn.predict(testX)
print(accuracy_score(testY, y_predict))
print(classification_report(testY, y_predict))
```

```
0.5679347826086957
```

	precision	recall	f1-score	support

```
    backdoor       0.17      0.55      0.26      7034
        ddos       0.61      0.35      0.45      5125
   injection       0.58      0.36      0.44      6950
      normal       0.75      0.70      0.73     48919
    password       0.67      0.26      0.37      7003
  ransomware       0.76      0.26      0.38      3184
    scanning       0.94      0.26      0.40       810
         xss       0.47      0.12      0.18      1199

    accuracy                           0.57     80224
   macro avg       0.62      0.36      0.40     80224
weighted avg       0.67      0.57      0.59     80224
```

2.0.4 RF

[107]:
```
#Test Train split
rf = RandomForestClassifier(n_estimators=10,criterion='gini')
rf.fit(trainX, trainY)
y_predict1 = rf.predict(testX)
print(accuracy_score(testY, y_predict1))
print(classification_report(testY, y_predict1))
```

```
0.6795472676505784
              precision    recall  f1-score   support

    backdoor       0.68      0.28      0.40     10449
        ddos       0.96      0.19      0.32      7678
   injection       0.70      0.29      0.41     10462
      normal       0.67      0.96      0.79     73454
    password       0.71      0.27      0.39     10490
  ransomware       0.58      0.17      0.26      4743
    scanning       0.94      0.20      0.33      1212
         xss       0.74      0.19      0.30      1848

    accuracy                           0.68    120336
   macro avg       0.75      0.32      0.40    120336
weighted avg       0.70      0.68      0.62    120336
```

2.0.5 DT (CART)

[108]:
```
#Test Train split
dt = DecisionTreeClassifier(criterion='gini')
dt.fit(trainX, trainY)
y_predict1 = dt.predict(testX)
print(accuracy_score(testY, y_predict1))
print(classification_report(testY, y_predict1))
```

```
0.6761983113947614
              precision    recall  f1-score   support

    backdoor       0.66      0.28      0.39     10449
        ddos       0.94      0.19      0.31      7678
   injection       0.68      0.29      0.40     10462
      normal       0.67      0.95      0.79     73454
    password       0.66      0.27      0.39     10490
  ransomware       0.53      0.17      0.26      4743
    scanning       0.85      0.21      0.34      1212
         xss       0.68      0.21      0.32      1848

    accuracy                           0.68    120336
   macro avg       0.71      0.32      0.40    120336
weighted avg       0.69      0.68      0.62    120336
```

2.0.6 NB

[47]:
```
#Test Train split
nb = GaussianNB()
nb.fit(trainX, trainY)
y_predict1 = nb.predict(testX)
print(accuracy_score(testY, y_predict1))
print(classification_report(testY, y_predict1))
```

```
0.07077682489030714

C:\Users\usman\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

    backdoor       0.00      0.00      0.00      7034
        ddos       0.09      1.00      0.16      5125
   injection       0.00      0.00      0.00      6950
      normal       0.00      0.00      0.00     48919
    password       0.00      0.00      0.00      7003
  ransomware       0.04      0.14      0.06      3184
    scanning       0.01      0.14      0.02       810
         xss       0.00      0.00      0.00      1199

    accuracy                           0.07     80224
   macro avg       0.02      0.16      0.03     80224
weighted avg       0.01      0.07      0.01     80224
```

2.0.7 SVM

[74]:
```
#Test Train split
svclassifier = SVC(kernel='rbf', gamma='auto')
svclassifier.fit(trainX, trainY)
y_predict = svclassifier.predict(testX)
print(accuracy_score(testY, y_predict))
print(classification_report(testY, y_predict))
```

```
0.6113257877941763

C:\Users\usman\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

    backdoor       0.00      0.00      0.00      7034
        ddos       1.00      0.02      0.05      5125
   injection       0.00      0.00      0.00      6950
      normal       0.61      1.00      0.76     48919
    password       0.00      0.00      0.00      7003
  ransomware       0.00      0.00      0.00      3184
    scanning       0.00      0.00      0.00       810
         xss       0.00      0.00      0.00      1199

    accuracy                           0.61     80224
   macro avg       0.20      0.13      0.10     80224
weighted avg       0.44      0.61      0.47     80224
```

2.0.8 LSTM

[113]:
```
XM.shape
```

[113]: (401119, 15)

[115]:
```
X_MV = XM.values
```

[117]:
```
X1 = X_MV.reshape((-1,1,15))
```

[118]:
```
X1.shape
```

[118]: (401119, 1, 15)

[119]:
```
len(np.unique(yM_G))
```

[119]: 8

[122]:
```
encoder = LabelEncoder()
encoder.fit(yM)
y_E = encoder.transform(yM)
```

[123]:
```
y_E.shape
```

[123]: (401119,)

[124]:
```
(trainX, testX, trainY, testY) = train_test_split(X1, y_E, test_size=0.30,␣
 ↪random_state = 5)
```

[126]:
```
# 1 LSTM Layer (input), 3 Dense Hidden Layers
model = Sequential()
model.add(LSTM(17, input_shape=(1, 15), activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(100, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(8,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))
```

```
Epoch 1/35
4388/4388 - 19s - loss: 1.3616 - accuracy: 0.6107 - val_loss: 1.3472 -
val_accuracy: 0.6104
Epoch 2/35
4388/4388 - 16s - loss: 1.3518 - accuracy: 0.6110 - val_loss: 1.3466 -
val_accuracy: 0.6104
Epoch 3/35
4388/4388 - 16s - loss: 1.3508 - accuracy: 0.6110 - val_loss: 1.3509 -
val_accuracy: 0.6104
Epoch 4/35
4388/4388 - 16s - loss: 1.3500 - accuracy: 0.6110 - val_loss: 1.3573 -
val_accuracy: 0.6104
Epoch 5/35
4388/4388 - 16s - loss: 1.3500 - accuracy: 0.6110 - val_loss: 1.3473 -
val_accuracy: 0.6104
Epoch 6/35
4388/4388 - 16s - loss: 1.3502 - accuracy: 0.6110 - val_loss: 1.3454 -
val_accuracy: 0.6104
Epoch 7/35
4388/4388 - 16s - loss: 1.3479 - accuracy: 0.6110 - val_loss: 1.3447 -
val_accuracy: 0.6104
```

```
Epoch 8/35
4388/4388 - 16s - loss: 1.3462 - accuracy: 0.6110 - val_loss: 1.3497 -
val_accuracy: 0.6104
Epoch 9/35
4388/4388 - 16s - loss: 1.3469 - accuracy: 0.6110 - val_loss: 1.3459 -
val_accuracy: 0.6104
Epoch 10/35
4388/4388 - 16s - loss: 1.3451 - accuracy: 0.6110 - val_loss: 1.3456 -
val_accuracy: 0.6104
Epoch 11/35
4388/4388 - 16s - loss: 1.3474 - accuracy: 0.6110 - val_loss: 1.3513 -
val_accuracy: 0.6104
Epoch 12/35
4388/4388 - 16s - loss: 1.3479 - accuracy: 0.6110 - val_loss: 1.3459 -
val_accuracy: 0.6104
Epoch 13/35
4388/4388 - 16s - loss: 1.3473 - accuracy: 0.6110 - val_loss: 1.3530 -
val_accuracy: 0.6104
Epoch 14/35
4388/4388 - 16s - loss: 1.3467 - accuracy: 0.6110 - val_loss: 1.3463 -
val_accuracy: 0.6104
Epoch 15/35
4388/4388 - 17s - loss: 1.3471 - accuracy: 0.6110 - val_loss: 1.3530 -
val_accuracy: 0.6104
Epoch 16/35
4388/4388 - 16s - loss: 1.3466 - accuracy: 0.6110 - val_loss: 1.3402 -
val_accuracy: 0.6104
Epoch 17/35
4388/4388 - 17s - loss: 1.3456 - accuracy: 0.6110 - val_loss: 1.3464 -
val_accuracy: 0.6104
Epoch 18/35
4388/4388 - 16s - loss: 1.3456 - accuracy: 0.6110 - val_loss: 1.3459 -
val_accuracy: 0.6104
Epoch 19/35
4388/4388 - 17s - loss: 1.3466 - accuracy: 0.6110 - val_loss: 1.3491 -
val_accuracy: 0.6104
Epoch 20/35
4388/4388 - 16s - loss: 1.3502 - accuracy: 0.6110 - val_loss: 1.3513 -
val_accuracy: 0.6104
Epoch 21/35
4388/4388 - 16s - loss: 1.3513 - accuracy: 0.6110 - val_loss: 1.3510 -
val_accuracy: 0.6104
Epoch 22/35
4388/4388 - 16s - loss: 1.3502 - accuracy: 0.6110 - val_loss: 1.3496 -
val_accuracy: 0.6104
Epoch 23/35
4388/4388 - 16s - loss: 1.3494 - accuracy: 0.6110 - val_loss: 1.3485 -
val_accuracy: 0.6104
```

```
Epoch 24/35
4388/4388 - 16s - loss: 1.3484 - accuracy: 0.6110 - val_loss: 1.3476 -
val_accuracy: 0.6104
Epoch 25/35
4388/4388 - 16s - loss: 1.3472 - accuracy: 0.6110 - val_loss: 1.3422 -
val_accuracy: 0.6104
Epoch 26/35
4388/4388 - 16s - loss: 1.3471 - accuracy: 0.6110 - val_loss: 1.3521 -
val_accuracy: 0.6104
Epoch 27/35
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-126-5968feebf5aa> in <module>
     11 model.add(Dense(8,activation='softmax'))
     12 model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
 ↪metrics=["accuracy"])
---> 13 model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,
 ↪validation_data=(testX, testY))

C:\ProgramData\Anaconda3\lib\site-packages\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)
   1182                 _r=1):
   1183               callbacks.on_train_batch_begin(step)
-> 1184               tmp_logs = self.train_function(iterator)
   1185               if data_handler.should_sync:
   1186                 context.async_wait()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in __call__(self, *args, **kwds)
    883
    884       with OptionalXlaContext(self._jit_compile):
--> 885         result = self._call(*args, **kwds)
    886
    887       new_tracing_count = self.experimental_get_tracing_count()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in _call(self, *args, **kwds)
    915       # In this case we have created variables on the first call, so we
 ↪run the
    916       # defunned version which is guaranteed to never create variables.
--> 917       return self._stateless_fn(*args, **kwds)  # pylint:
 ↪disable=not-callable
    918     elif self._stateful_fn is not None:
```

```
    919       # Release the lock early so that multiple threads can perform the
 ↪call

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py
 ↪in __call__(self, *args, **kwargs)
   3037       (graph_function,
   3038        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 3039     return graph_function._call_flat(
   3040         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access
   3041

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py
 ↪in _call_flat(self, args, captured_inputs, cancellation_manager)
   1961         and executing_eagerly):
   1962       # No tape is watching; skip to running the function.
-> 1963       return self._build_call_outputs(self._inference_function.call(
   1964           ctx, args, cancellation_manager=cancellation_manager))
   1965     forward_backward = self._select_forward_and_backward_functions(

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py
 ↪in call(self, ctx, args, cancellation_manager)
    589       with _InterpolateFunctionError(self):
    590         if cancellation_manager is None:
--> 591           outputs = execute.execute(
    592               str(self.signature.name),
    593               num_outputs=self._num_outputs,

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\execute.py i
 ↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,
 ↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

[ ]:
```
loss, accuracy = model.evaluate(testX, testY)
print("Loss:" + str(loss))
print("Accuracy:" + str(accuracy))
```

2.0.9 CNN (Conv1D)

[133]:
```
X_M.shape
```

[133]: (401119, 17)

[134]:
```
X1 = X_M.reshape((-1,1,17))
```

[135]:
```
X1.shape
```

[135]: (401119, 1, 17)

[136]:
```
encoder = LabelEncoder()
encoder.fit(yM_G)l
y_E = encoder.transform(yM_G)
```

[137]:
```
(trainX, testX, trainY, testY) = train_test_split(X1, y_E, test_size = 0.30,␣
 ↪random_state = 5)
```

[138]:
```
nb_filter = 250
filter_length = 3

model = Sequential()
model.add(Conv1D(filters=nb_filter, kernel_size=filter_length, padding='same',␣
 ↪activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(100, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(8,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))
```

```
Epoch 1/35
5014/5014 - 21s - loss: 1.3224 - accuracy: 0.6129 - val_loss: 1.2808 -
val_accuracy: 0.6189
Epoch 2/35
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-138-91fdfd111abf> in <module>
     13 model.add(Dense(8,activation='softmax'))
```

```
     14 model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
 ↪metrics=["accuracy"])
---> 15 model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,
 ↪validation_data=(testX, testY))

C:
 ↪\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\train
 ↪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()

C:\ProgramData\Anaconda3\lib\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()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in _call(self, *args, **kwds)
    853       # In this case we have created variables on the first call, so we
 ↪run the
    854       # defunned version which is guaranteed to never create variables.
--> 855       return self._stateless_fn(*args, **kwds)  # pylint:
 ↪disable=not-callable
    856     elif self._stateful_fn is not None:
    857       # Release the lock early so that multiple threads can perform the
 ↪call

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py
 ↪in __call__(self, *args, **kwargs)
   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
   2944

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py
 ↪in _call_flat(self, args, captured_inputs, cancellation_manager)
   1916         and executing_eagerly):
   1917       # No tape is watching; skip to running the function.
```

```
-> 1918       return self._build_call_outputs(self._inference_function.call(
   1919           ctx, args, cancellation_manager=cancellation_manager))
   1920     forward_backward = self._select_forward_and_backward_functions(

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in call(self, ctx, args, cancellation_manager)
    553       with _InterpolateFunctionError(self):
    554         if cancellation_manager is None:
--> 555           outputs = execute.execute(
    556               str(self.signature.name),
    557               num_outputs=self._num_outputs,

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\execute.py i␣
 ↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
 ↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

[ ]:
```
loss, accuracy = model.evaluate(testX, testY)
print("Loss:" + str(loss))
print("Accuracy:" + str(accuracy))
```

2.0.10 Conv-LSTM

[140]:
```
X_M.shape
```

[140]: (401119, 17)

[141]:
```
X1 = X_M.reshape((-1,1,17))
```

[142]:
```
X1.shape
```

[142]: (401119, 1, 17)

[143]:
```
encoder = LabelEncoder()
encoder.fit(yM_G)
y_E = encoder.transform(yM_G)
```

[144]:
```
(trainX, testX, trainY, testY) = train_test_split(X1, y_E, test_size = 0.30,␣
 ↪random_state = 5)
```

[145]:
```
nb_filter = 250
filter_length = 3

model = Sequential()
model.add(Conv1D(filters=nb_filter, kernel_size=filter_length, padding='same',␣
 ↪activation='tanh'))
model.add(MaxPooling1D(pool_size=1))
model.add(Conv1D(filters=nb_filter, kernel_size=filter_length, padding='same',␣
 ↪activation='tanh'))
model.add(MaxPooling1D(pool_size=1))
model.add(Activation('softmax'))
model.add(LSTM(17, return_sequences=True))
model.add(LSTM(17, return_sequences=True))
model.add(Dense(128, activation='tanh'))
model.add(Dense(100, activation='tanh'))
model.add(Dense(64, activation='tanh'))
model.add(Dense(8,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))
```

```
Epoch 1/35
5014/5014 - 49s - loss: 1.3253 - accuracy: 0.6137 - val_loss: 1.3003 -
val_accuracy: 0.6167
Epoch 2/35
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-145-7e57bc212920> in <module>
     15 model.add(Dense(8,activation='softmax'))
     16 model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',␣
 ↪metrics=["accuracy"])
---> 17 model.fit(trainX, trainY, epochs=35, batch_size=64, verbose=2,␣
 ↪validation_data=(testX, testY))

C:
 ↪\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\train␣ng.
 ↪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()
```

```
C:\ProgramData\Anaconda3\lib\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()

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in _call(self, *args, **kwds)
    853       # In this case we have created variables on the first call, so we
 ↪run the
    854       # defunned version which is guaranteed to never create variables.
--> 855       return self._stateless_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    856     elif self._stateful_fn is not None:
    857       # Release the lock early so that multiple threads can perform the
 ↪call

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in __call__(self, *args, **kwargs)
   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
   2944

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in _call_flat(self, args, captured_inputs, cancellation_manager)
   1916         and executing_eagerly):
   1917       # No tape is watching; skip to running the function.
-> 1918       return self._build_call_outputs(self._inference_function.call(
   1919           ctx, args, cancellation_manager=cancellation_manager))
   1920     forward_backward = self._select_forward_and_backward_functions(

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py␣
 ↪in call(self, ctx, args, cancellation_manager)
    553       with _InterpolateFunctionError(self):
    554         if cancellation_manager is None:
--> 555           outputs = execute.execute(
    556               str(self.signature.name),
    557               num_outputs=self._num_outputs,

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\execute.py i␣
 ↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
```

```
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
 ↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

[ ]:
```
loss, accuracy = model.evaluate(testX, testY)
print("Loss:" + str(loss))
print("Accuracy:" + str(accuracy))
```