#HUMAN ACTIVITY DETECTION ASSIGNMENT

WE HAVE RAW DATA OF THE ACCELERATION AND GYROSCOPE DATA OF THE 30 PERSONS COLLECTED.

In [0]:
```
# Code to read csv file into Colaboratory:
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
    100% |████████████████████████████████| 993kB 23.5MB/s ta 0:00:01
  Building wheel for PyDrive (setup.py) ... done
```

In [0]:
```
link = 'https://drive.google.com/open?id=1l1adJnTgeHULVuoLdTRyqi25-z6shBSL' # Th
```

In [0]:
```
fluff, id = link.split('=')
print (id) # Verify that you have everything after '='
```

```
1l1adJnTgeHULVuoLdTRyqi25-z6shBSL
```

In [0]:
```
import pandas as pd
downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('uploads')
```

In [0]:
```
!mkdir traindata
```

In [0]:
```
import os
os.getcwd()
os.chdir('/content/traindata')
```

In [4]:
```
!pwd
```

```
/content/traindata
```

#signals in the raw form provided to us based on the person behavior. the activities he can perform are

- walking
- walking upstairs
- walking downstairs
- sitting

- standing
- laying

IN THE ABOVE ACTIVITIES FIRST THREE ACTIVIVTIES ARE DYANMIC WHICH INVLOVES THE MOVEMENT OF THE PERSON WHERE AS THE LAST THREE ACTIVITIES ARE STATIC IN WHICH THE PERSONG DOES NOT MOVE.

BASED ON THE OBTAINED DATA WE CAN PERFORM THE EXPLORATORY DATA ANALYSIS, DO THE FEATURE EXTRACTION AND WITH THE FEATURE ENGINEERING TECHNIQUES WE CAN OBTAIN THE BEST FEATURES BASEF ON THE DATA WE HAVE.

LATER WE CAN EMPLOY INTO THE MODELS AND OBTAIN THE BEST RESULTS BY CHOOSING THE BEST FEATURES AND BEST MODEL

WE CAN USE THE DEEP LEARNING MODELS LIKE LSTM IN WHICH WE CAN USE THE RAW DATA WE OBTAINED AND DIRECTLY EMPLOY INTO THE DEEPLEARNING MODELS.

WE CAN PERFORM THE HYPERPARAMETER TUNING ON THE MODELS AND OBTAIN THE BEST HYPERPARAMETER TO OBTAIN BEST RESULS IN TERMS OF HOW ACCURATE OUR MODEL IS.

In [0]:

In [5]:
```
!ls
```

```
body_acc_x_test.txt   body_gyro_x_train.txt  total_acc_y_test.txt
body_acc_x_train.txt  body_gyro_y_test.txt   total_acc_y_train.txt
body_acc_y_test.txt   body_gyro_y_train.txt  total_acc_z_test.txt
body_acc_y_train.txt  body_gyro_z_test.txt   total_acc_z_train.txt
body_acc_z_test.txt   body_gyro_z_train.txt  y_test.txt
body_acc_z_train.txt  total_acc_x_test.txt   y_train.txt
body_gyro_x_test.txt  total_acc_x_train.txt
```

In [0]:
```
import numpy as np
```

In [0]:
```
signals2=[
    "body_acc_x_test.txt",
    "body_acc_y_test.txt",
    "body_acc_z_test.txt",
    "body_gyro_x_test.txt",
    "body_gyro_y_test.txt",
    "body_gyro_z_test.txt",
    "total_acc_x_test.txt",
    "total_acc_y_test.txt",
    "total_acc_z_test.txt",
]
```

In [0]:
```
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

In [0]:
```
signals1=[
    "body_acc_x_train.txt",
    "body_acc_y_train.txt",
    "body_acc_z_train.txt",
    "body_gyro_x_train.txt",
    "body_gyro_y_train.txt",
    "body_gyro_z_train.txt",
    "total_acc_x_train.txt",
    "total_acc_y_train.txt",
    "total_acc_z_train.txt"
]
```

Type *Markdown* and LaTeX: α^2

In [0]:
```
def read_csv(filename):
    return pd.read_csv(filename,delim_whitespace=True, header=None)

# Utility function to load the load
def load_signals(subset):
    signals_data = []

    for signal in signals1:

        signals_data.append(read_csv(signal).as_matrix())


    return np.transpose(signals_data, (1, 2, 0))
def load_signals1(subset):
    signals_data = []

    for signal in signals2:

        signals_data.append(read_csv(signal).as_matrix())


    return np.transpose(signals_data, (1, 2, 0))
```

In [0]:
```
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}
```

get_dummies is atype of encoding .we are using to encode the output label for ex; if the label is zero it is represented as 100000 based on the size of the label.

In [0]:
```
def load_y(subset):

    y = read_csv("y_train.txt")[0]


    return pd.get_dummies(y).as_matrix()

def load_y1(subset):


    y = read_csv("y_test.txt")[0]

    return pd.get_dummies(y).as_matrix()
```

In [0]:
```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train,X_test= load_signals('train'),load_signals1('test')
    y_train, y_test = load_y('train'), load_y1('test')

    return X_train, X_test, y_train, y_test
```

In [14]:
```
import  pandas as pd
X_train, X_test, Y_train, Y_test = load_data()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:19: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:14: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
```

In [0]:
```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [0]:
```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [17]:
```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

```
Using TensorFlow backend.
```

we are using the lstm deeplearning model.

In [0]:
```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [0]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [0]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [21]:
```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:19: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:14: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
```

In [22]:
```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

print(timesteps)
print(input_dim)
print(len(X_train))
```

```
128
9
7352
```

In [0]:
```
def errorplot(x,validationy,testy,ax):
  ax.plot(x,validationy,label='trainloss')
  ax.plot(x,testy,label='validationloss')
  plt.xlabel('number_of_epochs')
  plt.ylabel('categorical_cross_entropy')
  plt.legend()
  plt.show()
```

- Defining the Architecture of LSTM

AS THE PART OF MODEL 1 WE ARE USING THE LSTM LAYER WITH THE 32 HIDDEN LAYERS AT FIRST LSTM LAYER. LATER WE ARE USING THE DENSE LAYER WITH THE NUMBER OF OUTPUT ACTIVITIES WE HAVE. WE ARE USING THE DROPOUT INORDER TO REDUCE THE OVERFITTING OF OUR MODEL.SO THAT IF WE GIVE THE 0.5 AS THE DROPOUT VALUE THEN THE 50 PERCENT OF THE LSTM LAYER WILL BE TURNED OFF.

In [23]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/pytho
n/framework/op_def_library.py:263: colocate_with (from tensorflow.python.framew
ork.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) w
ith keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

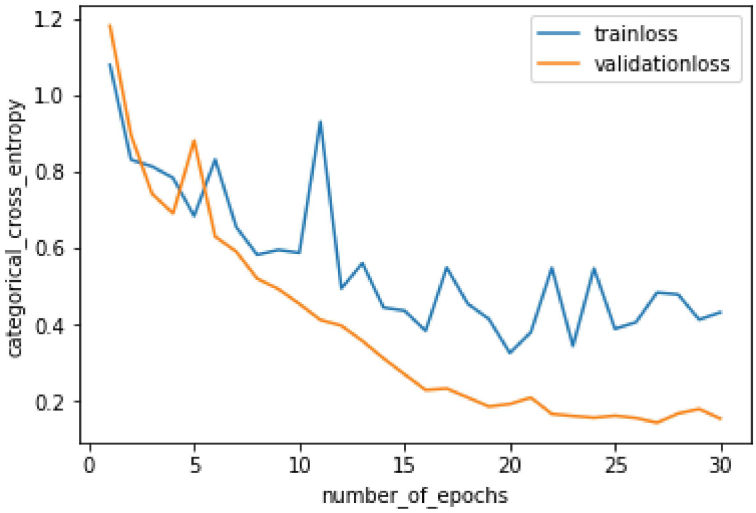
In [0]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [26]:
```
# Training the model
history=model.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test,
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 92s 12ms/step - loss: 1.1821 - ac
c: 0.4849 - val_loss: 1.0799 - val_acc: 0.4812
Epoch 2/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.8964 - ac
c: 0.6049 - val_loss: 0.8318 - val_acc: 0.6328
Epoch 3/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.7430 - ac
c: 0.6514 - val_loss: 0.8144 - val_acc: 0.5904
Epoch 4/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.6915 - ac
c: 0.6616 - val_loss: 0.7841 - val_acc: 0.6179
Epoch 5/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.8822 - ac
c: 0.6315 - val_loss: 0.6848 - val_acc: 0.6301
Epoch 6/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.6310 - ac
c: 0.7087 - val_loss: 0.8325 - val_acc: 0.6926
Epoch 7/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.5917 - ac
c: 0.7489 - val_loss: 0.6560 - val_acc: 0.7296
Epoch 8/30
7352/7352 [==============================] - 92s 12ms/step - loss: 0.5208 - ac
c: 0.7729 - val_loss: 0.5831 - val_acc: 0.7513
Epoch 9/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.4937 - ac
c: 0.7826 - val_loss: 0.5962 - val_acc: 0.7251
Epoch 10/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.4549 - ac
c: 0.7996 - val_loss: 0.5883 - val_acc: 0.7014
Epoch 11/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.4131 - ac
c: 0.8211 - val_loss: 0.9308 - val_acc: 0.6919
Epoch 12/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.3980 - ac
c: 0.8395 - val_loss: 0.4951 - val_acc: 0.8303
Epoch 13/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.3580 - ac
c: 0.8719 - val_loss: 0.5613 - val_acc: 0.8276
Epoch 14/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.3124 - ac
c: 0.9021 - val_loss: 0.4453 - val_acc: 0.8690
Epoch 15/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.2706 - ac
c: 0.9176 - val_loss: 0.4370 - val_acc: 0.8711
Epoch 16/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2295 - ac
c: 0.9266 - val_loss: 0.3845 - val_acc: 0.8887
Epoch 17/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.2333 - ac
c: 0.9253 - val_loss: 0.5497 - val_acc: 0.8541
Epoch 18/30
```

```
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2101 - ac
c: 0.9385 - val_loss: 0.4552 - val_acc: 0.8850
Epoch 19/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1864 - ac
c: 0.9328 - val_loss: 0.4158 - val_acc: 0.8904
Epoch 20/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1928 - ac
c: 0.9351 - val_loss: 0.3266 - val_acc: 0.8938
Epoch 21/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.2097 - ac
c: 0.9295 - val_loss: 0.3806 - val_acc: 0.9006
Epoch 22/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1670 - ac
c: 0.9422 - val_loss: 0.5489 - val_acc: 0.8887
Epoch 23/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1619 - ac
c: 0.9442 - val_loss: 0.3447 - val_acc: 0.8941
Epoch 24/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1574 - ac
c: 0.9449 - val_loss: 0.5478 - val_acc: 0.9050
Epoch 25/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.1626 - ac
c: 0.9438 - val_loss: 0.3895 - val_acc: 0.9013
Epoch 26/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1565 - ac
c: 0.9464 - val_loss: 0.4071 - val_acc: 0.9002
Epoch 27/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1442 - ac
c: 0.9474 - val_loss: 0.4845 - val_acc: 0.8955
Epoch 28/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1683 - ac
c: 0.9456 - val_loss: 0.4794 - val_acc: 0.8989
Epoch 29/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1800 - ac
c: 0.9445 - val_loss: 0.4138 - val_acc: 0.9135
Epoch 30/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.1551 - ac
c: 0.9464 - val_loss: 0.4326 - val_acc: 0.9114
```

In [38]:
```
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
x=list(range(1,31))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



In [0]:
```
def confusion_matrix(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

In [34]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         0        0                   0
SITTING                  0      347       126        0                   0
STANDING                 0       47       484        1                   0
WALKING                  0        0         0      479                  11
WALKING_DOWNSTAIRS       0        0         0        0                 408
WALKING_UPSTAIRS         0        0         2        6                   5

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                           18
STANDING                           0
WALKING                            6
WALKING_DOWNSTAIRS                12
WALKING_UPSTAIRS                 458
```

In [35]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 6s 2ms/step
```

In [36]:
```
score
```

Out[36]:
```
[0.43244978515694077, 0.9114353579911775]
```

##WE HAVE ACHIEVED 90 PERCENT ACCURACY WITH THE SINGLE LSTM LAYER AND 32 CELLS IN LAYER WITH OUT USIG ANY FEATURIASTIONS . WE TRIED TO MINIMISE THE CATEGORICAL CROSS ENTROPY. WE CAN ALO CHANGE AND TUNE THE VARIOUS HYPERPARAMETERS SO THAT WE CAN CONSTRUCT THE BETTER LSTM MODEL.

###model2 we gonna use the 3 lstm layers where the fiirst lstm layer consists of 64 hidden cells and 300 cells in second layer and 200 layers in the third layer.we also use the batch normalisation because we normalise the before entering into the lstm layer but after the data enters into the lstm layers due to performing the activation functions like sigmoid and relu the data may loose the behavior hence we use the batch normalisation so that we can make the data to maintain its behavior we also perform introducing the droput layer to prevent the overfitting in the data.

In [0]:
```
n_hidden = 64
from keras.layers.normalization import BatchNormalization
```

In [49]:

```
model1 = Sequential()
model1.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),return_sequences=Tru
model1.add(BatchNormalization())
model1.add(Dropout(0.6))
model1.add(LSTM(300,return_sequences=True))
model1.add(BatchNormalization())
model1.add(Dropout(0.4))
model1.add(LSTM(200))
model1.add(BatchNormalization())
model1.add(Dropout(0.6))
model1.add(Dense(n_classes, activation='sigmoid'))
model1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
batch_normalization_4 (Batch (None, 128, 64)           256
_________________________________________________________________
dropout_5 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_6 (LSTM)                (None, 128, 300)          438000
_________________________________________________________________
batch_normalization_5 (Batch (None, 128, 300)          1200
_________________________________________________________________
dropout_6 (Dropout)          (None, 128, 300)          0
_________________________________________________________________
lstm_7 (LSTM)                (None, 200)               400800
_________________________________________________________________
batch_normalization_6 (Batch (None, 200)               800
_________________________________________________________________
dropout_7 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 1206
=================================================================
Total params: 861,206
Trainable params: 860,078
Non-trainable params: 1,128
_________________________________________________________________
```

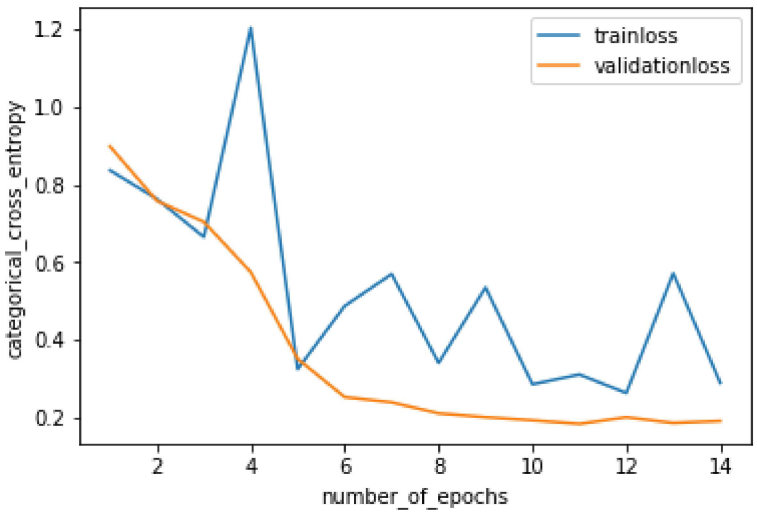
In [0]:

```
# Compiling the model
model1.compile(loss='categorical_crossentropy',
               optimizer='rmsprop',
               metrics=['accuracy'])
```

In [51]:
```
# Training the model
history=model1.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=14)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/14
7352/7352 [==============================] - 277s 38ms/step - loss: 0.8985 - ac
c: 0.6249 - val_loss: 0.8367 - val_acc: 0.6210
Epoch 2/14
7352/7352 [==============================] - 274s 37ms/step - loss: 0.7580 - ac
c: 0.6483 - val_loss: 0.7627 - val_acc: 0.6084
Epoch 3/14
7352/7352 [==============================] - 275s 37ms/step - loss: 0.7036 - ac
c: 0.6677 - val_loss: 0.6646 - val_acc: 0.6956
Epoch 4/14
7352/7352 [==============================] - 275s 37ms/step - loss: 0.5740 - ac
c: 0.7578 - val_loss: 1.2045 - val_acc: 0.7258
Epoch 5/14
7352/7352 [==============================] - 272s 37ms/step - loss: 0.3500 - ac
c: 0.8886 - val_loss: 0.3237 - val_acc: 0.8941
Epoch 6/14
7352/7352 [==============================] - 266s 36ms/step - loss: 0.2516 - ac
c: 0.9163 - val_loss: 0.4864 - val_acc: 0.8901
Epoch 7/14
7352/7352 [==============================] - 265s 36ms/step - loss: 0.2380 - ac
c: 0.9218 - val_loss: 0.5689 - val_acc: 0.8717
Epoch 8/14
7352/7352 [==============================] - 265s 36ms/step - loss: 0.2096 - ac
c: 0.9283 - val_loss: 0.3397 - val_acc: 0.9087
Epoch 9/14
7352/7352 [==============================] - 267s 36ms/step - loss: 0.1997 - ac
c: 0.9276 - val_loss: 0.5344 - val_acc: 0.8999
Epoch 10/14
7352/7352 [==============================] - 264s 36ms/step - loss: 0.1917 - ac
c: 0.9357 - val_loss: 0.2845 - val_acc: 0.9087
Epoch 11/14
7352/7352 [==============================] - 265s 36ms/step - loss: 0.1825 - ac
c: 0.9357 - val_loss: 0.3097 - val_acc: 0.8697
Epoch 12/14
7352/7352 [==============================] - 266s 36ms/step - loss: 0.1995 - ac
c: 0.9249 - val_loss: 0.2621 - val_acc: 0.9175
Epoch 13/14
7352/7352 [==============================] - 265s 36ms/step - loss: 0.1848 - ac
c: 0.9316 - val_loss: 0.5709 - val_acc: 0.8785
Epoch 14/14
7352/7352 [==============================] - 265s 36ms/step - loss: 0.1899 - ac
c: 0.9335 - val_loss: 0.2883 - val_acc: 0.9063
```

In [53]:
```
fig,ax=plt.subplots()
x=list(range(1,15))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



In [54]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model1.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	\
True						
LAYING	537	0	0	0	0	
SITTING	5	290	194	0	0	
STANDING	0	41	490	0	0	
WALKING	0	0	0	495	1	
WALKING_DOWNSTAIRS	0	0	0	0	420	
WALKING_UPSTAIRS	0	0	0	18	14	

Pred	WALKING_UPSTAIRS
True	
LAYING	0
SITTING	2
STANDING	1
WALKING	0
WALKING_DOWNSTAIRS	0
WALKING_UPSTAIRS	439

In [55]:
```
score = model1.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 17s 6ms/step
```

In [56]:
```
score
```

Out[56]:
```
[0.2883422779202473, 0.9063454360366474]
```

if we observe in the confusion matrix it i very clear that the model is confusing to predict the difference between standing and sitting.

because both the are static and may giving the similar signals as input. we have achieved 91 percent accuracy

###WE ARE RUNNING THE MODEL WITH 2 LSTM LAYERS WITH 32 CELLS IN THE HIDDEN LAYER WE ADD THE OUTPUT DENSE LAYER AND WE TAKE DENSEOUTPUT AT THE WITH SIGMOID AS THE ACTIVATION FUNCTION. WE USE 30 EPOCHS AND OBTAIN THE PERFORMANCE . THIS CASE WE ADDED THE DROPOUT LAYER BWE HAVE NOT DONE THE BATCHNORMALISATION. WE TAKE THE DROPOUT RATES OF 50 PERCENT AND ANOTHER DROPOUT LAYER OF 50 PERCENT WHICH MEANS 50 PERCENT OF THE HIDDEN CELLS WILL BE INACTIVE WHILE ADJUSTING THE WEIGHTS.

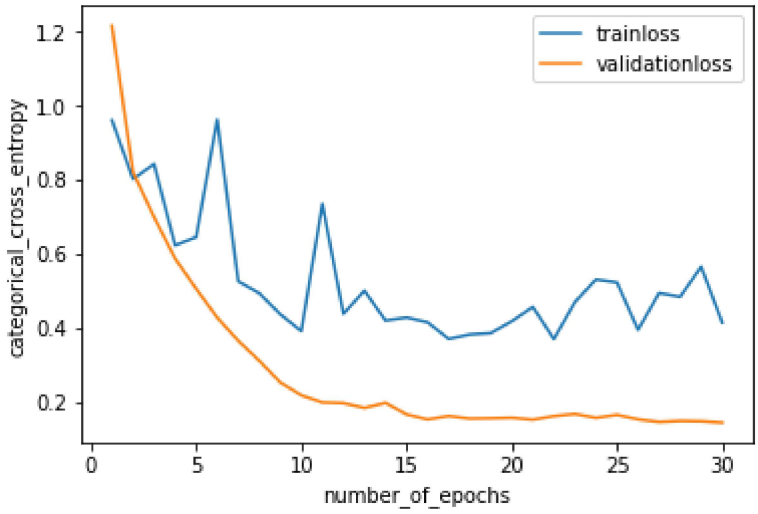
In [57]:
```
epochs=30
modelx = Sequential()
modelx.add(LSTM(32,return_sequences=True, input_shape=(timesteps, input_dim)))
modelx.add(Dropout(0.5))
modelx.add(LSTM(32))
modelx.add(Dropout(0.5))
modelx.add(Dense(n_classes, activation='sigmoid'))
print(modelx.summary())
modelx.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['acc

history = modelx.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_te
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
dropout_8 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_9 (LSTM)                (None, 32)                8320
_________________________________________________________________
dropout_9 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 198
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 185s 25ms/step - loss: 1.2171 - ac
c: 0.5212 - val_loss: 0.9619 - val_acc: 0.6295
Epoch 2/30
7352/7352 [==============================] - 183s 25ms/step - loss: 0.8216 - ac
c: 0.6362 - val_loss: 0.8025 - val_acc: 0.6451
Epoch 3/30
7352/7352 [==============================] - 183s 25ms/step - loss: 0.6996 - ac
c: 0.6984 - val_loss: 0.8431 - val_acc: 0.6498
Epoch 4/30
7352/7352 [==============================] - 180s 25ms/step - loss: 0.5875 - ac
c: 0.7601 - val_loss: 0.6231 - val_acc: 0.7313
Epoch 5/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.5058 - ac
c: 0.7877 - val_loss: 0.6454 - val_acc: 0.7326
Epoch 6/30
7352/7352 [==============================] - 177s 24ms/step - loss: 0.4277 - ac
c: 0.8105 - val_loss: 0.9636 - val_acc: 0.7292
Epoch 7/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.3650 - ac
c: 0.8811 - val_loss: 0.5259 - val_acc: 0.8578
Epoch 8/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.3109 - ac
c: 0.9128 - val_loss: 0.4936 - val_acc: 0.8806
Epoch 9/30
```

```
7352/7352 [==============================] - 181s 25ms/step - loss: 0.2522 - ac
c: 0.9263 - val_loss: 0.4366 - val_acc: 0.8982
Epoch 10/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.2173 - ac
c: 0.9316 - val_loss: 0.3910 - val_acc: 0.9077
Epoch 11/30
7352/7352 [==============================] - 177s 24ms/step - loss: 0.1979 - ac
c: 0.9407 - val_loss: 0.7349 - val_acc: 0.8544
Epoch 12/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.1961 - ac
c: 0.9377 - val_loss: 0.4378 - val_acc: 0.9006
Epoch 13/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.1832 - ac
c: 0.9407 - val_loss: 0.5004 - val_acc: 0.9002
Epoch 14/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1967 - ac
c: 0.9400 - val_loss: 0.4196 - val_acc: 0.9046
Epoch 15/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.1655 - ac
c: 0.9461 - val_loss: 0.4277 - val_acc: 0.9030
Epoch 16/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1520 - ac
c: 0.9475 - val_loss: 0.4148 - val_acc: 0.8955
Epoch 17/30
7352/7352 [==============================] - 180s 25ms/step - loss: 0.1611 - ac
c: 0.9456 - val_loss: 0.3699 - val_acc: 0.9026
Epoch 18/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.1542 - ac
c: 0.9478 - val_loss: 0.3818 - val_acc: 0.9016
Epoch 19/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1551 - ac
c: 0.9478 - val_loss: 0.3856 - val_acc: 0.8955
Epoch 20/30
7352/7352 [==============================] - 181s 25ms/step - loss: 0.1564 - ac
c: 0.9474 - val_loss: 0.4176 - val_acc: 0.9057
Epoch 21/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.1513 - ac
c: 0.9497 - val_loss: 0.4565 - val_acc: 0.8972
Epoch 22/30
7352/7352 [==============================] - 186s 25ms/step - loss: 0.1609 - ac
c: 0.9468 - val_loss: 0.3694 - val_acc: 0.9053
Epoch 23/30
7352/7352 [==============================] - 181s 25ms/step - loss: 0.1667 - ac
c: 0.9465 - val_loss: 0.4696 - val_acc: 0.8972
Epoch 24/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.1563 - ac
c: 0.9460 - val_loss: 0.5307 - val_acc: 0.8829
Epoch 25/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1643 - ac
c: 0.9433 - val_loss: 0.5227 - val_acc: 0.8918
Epoch 26/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1522 - ac
c: 0.9512 - val_loss: 0.3942 - val_acc: 0.8996
Epoch 27/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.1451 - ac
c: 0.9489 - val_loss: 0.4940 - val_acc: 0.9043
Epoch 28/30
```

```
7352/7352 [==============================] - 178s 24ms/step - loss: 0.1480 - ac
c: 0.9509 - val_loss: 0.4838 - val_acc: 0.8965
Epoch 29/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1471 - ac
c: 0.9479 - val_loss: 0.5653 - val_acc: 0.8877
Epoch 30/30
7352/7352 [==============================] - 177s 24ms/step - loss: 0.1433 - ac
c: 0.9516 - val_loss: 0.4142 - val_acc: 0.9002
```

In [58]:
```
fig,ax=plt.subplots()
x=list(range(1,31))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



In [59]:
```
print(confusion_matrix(Y_test, modelx.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	\
True						
LAYING	510	0	0	0	0	
SITTING	1	407	80	0	1	
STANDING	0	108	424	0	0	
WALKING	0	1	0	474	9	
WALKING_DOWNSTAIRS	0	0	0	0	413	
WALKING_UPSTAIRS	4	5	4	20	13	

Pred	WALKING_UPSTAIRS
True	
LAYING	27
SITTING	2
STANDING	0
WALKING	12
WALKING_DOWNSTAIRS	7
WALKING_UPSTAIRS	425

In [60]:
```
score = modelx.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 12s 4ms/step
```

In [61]:
```
score
```

Out[61]:
```
[0.4144287348136521, 0.9002375296912114]
```

##IN THIS MODEL WE HAVE INCREASED THE DROPOUT RATES TO 0.7 BECAUSE THE MODEL IS UNDERGOING OVERFITTING . IF WE OBSERVE THE TRAININGDATA ACCURACY IS NEARLY 0.95 AND 0.9

In [62]:
```
epochs=30
modelx = Sequential()
modelx.add(LSTM(32,return_sequences=True, input_shape=(timesteps, input_dim)))
modelx.add(Dropout(0.6))
modelx.add(LSTM(32))
modelx.add(Dropout(0.6))
modelx.add(Dense(n_classes, activation='sigmoid'))
print(modelx.summary())
modelx.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['acc

history = modelx.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_te
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_10 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
dropout_10 (Dropout)         (None, 128, 32)           0
_________________________________________________________________
lstm_11 (LSTM)               (None, 32)                8320
_________________________________________________________________
dropout_11 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 198
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/26
7352/7352 [==============================] - 187s 25ms/step - loss: 1.2687 - ac
c: 0.4723 - val_loss: 0.9974 - val_acc: 0.5433
Epoch 2/26
7352/7352 [==============================] - 183s 25ms/step - loss: 0.8871 - ac
c: 0.6045 - val_loss: 0.8607 - val_acc: 0.6176
Epoch 3/26
7352/7352 [==============================] - 185s 25ms/step - loss: 0.8625 - ac
c: 0.6106 - val_loss: 0.7981 - val_acc: 0.6763
Epoch 4/26
7352/7352 [==============================] - 181s 25ms/step - loss: 0.7212 - ac
c: 0.6601 - val_loss: 0.6997 - val_acc: 0.6824
Epoch 5/26
7352/7352 [==============================] - 182s 25ms/step - loss: 0.6782 - ac
c: 0.6768 - val_loss: 0.6992 - val_acc: 0.6980
Epoch 6/26
7352/7352 [==============================] - 181s 25ms/step - loss: 0.6403 - ac
c: 0.7175 - val_loss: 0.7347 - val_acc: 0.6953
Epoch 7/26
7352/7352 [==============================] - 182s 25ms/step - loss: 0.5276 - ac
c: 0.7737 - val_loss: 0.6798 - val_acc: 0.7469
Epoch 8/26
7352/7352 [==============================] - 183s 25ms/step - loss: 0.4312 - ac
c: 0.8104 - val_loss: 0.5346 - val_acc: 0.7608
Epoch 9/26
```

```
7352/7352 [==============================] - 181s 25ms/step - loss: 0.3651 - ac
c: 0.8800 - val_loss: 0.5869 - val_acc: 0.8548
Epoch 10/26
7352/7352 [==============================] - 182s 25ms/step - loss: 0.2909 - ac
c: 0.9151 - val_loss: 0.4944 - val_acc: 0.8548
Epoch 11/26
7352/7352 [==============================] - 181s 25ms/step - loss: 0.2495 - ac
c: 0.9308 - val_loss: 0.4422 - val_acc: 0.8897
Epoch 12/26
7352/7352 [==============================] - 179s 24ms/step - loss: 0.2515 - ac
c: 0.9305 - val_loss: 0.5107 - val_acc: 0.8897
Epoch 13/26
7352/7352 [==============================] - 179s 24ms/step - loss: 0.2216 - ac
c: 0.9340 - val_loss: 0.4551 - val_acc: 0.9023
Epoch 14/26
7352/7352 [==============================] - 180s 25ms/step - loss: 0.2082 - ac
c: 0.9410 - val_loss: 0.5188 - val_acc: 0.8826
Epoch 15/26
7352/7352 [==============================] - 182s 25ms/step - loss: 0.2054 - ac
c: 0.9418 - val_loss: 0.5100 - val_acc: 0.9036
Epoch 16/26
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1854 - ac
c: 0.9396 - val_loss: 0.7306 - val_acc: 0.8894
Epoch 17/26
7352/7352 [==============================] - 180s 25ms/step - loss: 0.1805 - ac
c: 0.9429 - val_loss: 0.4348 - val_acc: 0.9080
Epoch 18/26
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1985 - ac
c: 0.9422 - val_loss: 0.5473 - val_acc: 0.8938
Epoch 19/26
7352/7352 [==============================] - 181s 25ms/step - loss: 0.1625 - ac
c: 0.9484 - val_loss: 0.6593 - val_acc: 0.8890
Epoch 20/26
7352/7352 [==============================] - 181s 25ms/step - loss: 0.1694 - ac
c: 0.9419 - val_loss: 0.5884 - val_acc: 0.9040
Epoch 21/26
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1779 - ac
c: 0.9436 - val_loss: 0.5539 - val_acc: 0.8928
Epoch 22/26
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1687 - ac
c: 0.9457 - val_loss: 0.3005 - val_acc: 0.9158
Epoch 23/26
7352/7352 [==============================] - 180s 25ms/step - loss: 0.1711 - ac
c: 0.9448 - val_loss: 0.6058 - val_acc: 0.8873
Epoch 24/26
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1722 - ac
c: 0.9453 - val_loss: 0.8833 - val_acc: 0.8772
Epoch 25/26
7352/7352 [==============================] - 182s 25ms/step - loss: 0.1756 - ac
c: 0.9467 - val_loss: 0.5013 - val_acc: 0.9009
Epoch 26/26
7352/7352 [==============================] - 181s 25ms/step - loss: 0.1719 - ac
c: 0.9399 - val_loss: 0.8858 - val_acc: 0.8775
```

In [63]:
```
print(Y_test)
```

```
[[0 0 0 0 1 0]
 [0 0 0 0 1 0]
 [0 0 0 0 1 0]
 ...
 [0 1 0 0 0 0]
 [0 1 0 0 0 0]
 [0 1 0 0 0 0]]
```

In [64]:
```
score = modelx.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 12s 4ms/step
```

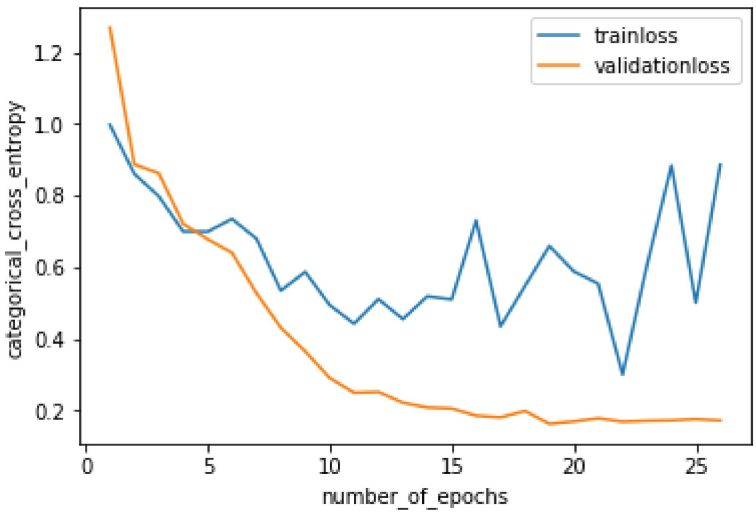
In [65]:
```
score
```

Out[65]:
```
[0.8857371849105553, 0.8775025449609772]
```

In [66]:
```
print(confusion_matrix(Y_test, modelx.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  5      420        50        0                   1
STANDING                 0      127       405        0                   0
WALKING                  0        0         3      422                  65
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        0         0        2                  87

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           15
STANDING                           0
WALKING                            6
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 382
```

In [68]:
```
fig,ax=plt.subplots()
x=list(range(1,27))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



In [69]:

```
modelnew = Sequential()

modelnew.add(LSTM(64,return_sequences=True, input_shape=(timesteps, input_dim)))
modelnew.add(Dropout(0.7))
modelnew.add(LSTM(64))
modelnew.add(Dropout(0.7))
modelnew.add(Dense(n_classes, activation='sigmoid'))
print(modelnew.summary())
modelnew.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['a

history= modelnew.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_t
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_12 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
dropout_12 (Dropout)         (None, 128, 64)           0
_________________________________________________________________
lstm_13 (LSTM)               (None, 64)                33024
_________________________________________________________________
dropout_13 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 390
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 192s 26ms/step - loss: 1.1373 - ac
c: 0.5105 - val_loss: 0.8658 - val_acc: 0.6210
Epoch 2/20
7352/7352 [==============================] - 190s 26ms/step - loss: 0.7867 - ac
c: 0.6147 - val_loss: 0.8071 - val_acc: 0.6043
Epoch 3/20
7352/7352 [==============================] - 189s 26ms/step - loss: 0.7100 - ac
c: 0.6568 - val_loss: 0.7686 - val_acc: 0.6186
Epoch 4/20
7352/7352 [==============================] - 177s 24ms/step - loss: 0.6425 - ac
c: 0.7016 - val_loss: 0.6528 - val_acc: 0.7285
Epoch 5/20
7352/7352 [==============================] - 178s 24ms/step - loss: 0.5010 - ac
c: 0.7841 - val_loss: 0.6199 - val_acc: 0.7832
Epoch 6/20
7352/7352 [==============================] - 175s 24ms/step - loss: 0.3937 - ac
c: 0.8475 - val_loss: 0.4984 - val_acc: 0.8568
Epoch 7/20
7352/7352 [==============================] - 178s 24ms/step - loss: 0.3220 - ac
c: 0.9038 - val_loss: 0.4374 - val_acc: 0.8778
Epoch 8/20
7352/7352 [==============================] - 177s 24ms/step - loss: 0.2444 - ac
c: 0.9263 - val_loss: 0.4876 - val_acc: 0.8792
```

```
Epoch 9/20
7352/7352 [==============================] - 176s 24ms/step - loss: 0.2079 - ac
c: 0.9342 - val_loss: 0.6881 - val_acc: 0.8643
Epoch 10/20
7352/7352 [==============================] - 176s 24ms/step - loss: 0.2132 - ac
c: 0.9353 - val_loss: 0.4767 - val_acc: 0.9040
Epoch 11/20
7352/7352 [==============================] - 174s 24ms/step - loss: 0.2299 - ac
c: 0.9346 - val_loss: 0.9965 - val_acc: 0.8235
Epoch 12/20
7352/7352 [==============================] - 175s 24ms/step - loss: 0.2027 - ac
c: 0.9374 - val_loss: 0.5993 - val_acc: 0.8972
Epoch 13/20
7352/7352 [==============================] - 175s 24ms/step - loss: 0.2300 - ac
c: 0.9317 - val_loss: 0.5145 - val_acc: 0.9019
Epoch 14/20
7352/7352 [==============================] - 176s 24ms/step - loss: 0.1850 - ac
c: 0.9384 - val_loss: 0.4399 - val_acc: 0.9067
Epoch 15/20
7352/7352 [==============================] - 174s 24ms/step - loss: 0.2105 - ac
c: 0.9406 - val_loss: 0.4615 - val_acc: 0.9101
Epoch 16/20
7352/7352 [==============================] - 176s 24ms/step - loss: 0.2250 - ac
c: 0.9340 - val_loss: 0.5820 - val_acc: 0.8941
Epoch 17/20
7352/7352 [==============================] - 176s 24ms/step - loss: 0.1939 - ac
c: 0.9396 - val_loss: 0.6567 - val_acc: 0.8833
Epoch 18/20
7352/7352 [==============================] - 173s 24ms/step - loss: 0.1955 - ac
c: 0.9406 - val_loss: 0.4993 - val_acc: 0.9057
Epoch 19/20
7352/7352 [==============================] - 176s 24ms/step - loss: 0.1722 - ac
c: 0.9442 - val_loss: 0.4915 - val_acc: 0.9070
Epoch 20/20
7352/7352 [==============================] - 174s 24ms/step - loss: 0.1724 - ac
c: 0.9467 - val_loss: 0.7762 - val_acc: 0.8795
```

In [70]:
```
print(confusion_matrix(Y_test, modelnew.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        27        0                   0
SITTING                  0      385       104        0                   0
STANDING                 1       89       442        0                   0
WALKING                  0        0         0      470                   0
WALKING_DOWNSTAIRS       0        0         0        3                 412
WALKING_UPSTAIRS         2        1         0       93                   2

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                           26
WALKING_DOWNSTAIRS                 5
WALKING_UPSTAIRS                 373
```

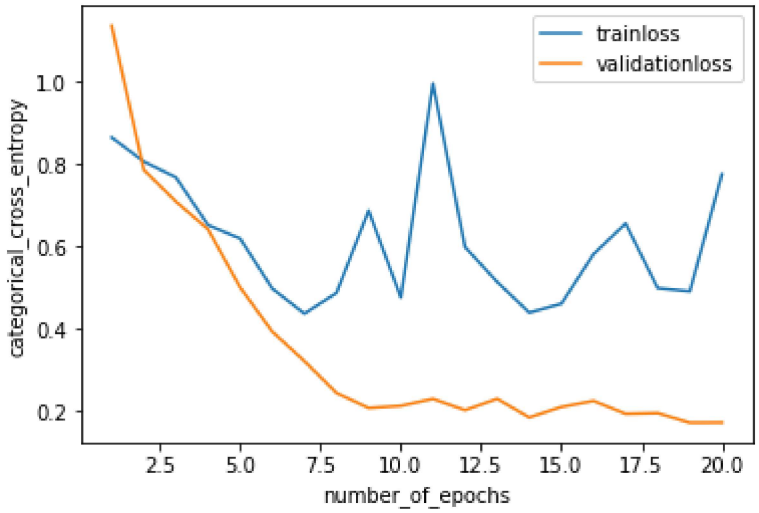
In [71]:
```
score = modelnew.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 12s 4ms/step
```

In [72]:
```
score
```

Out[72]:
```
[0.7758174211089318, 0.8795385137427892]
```

In [73]:
```
fig,ax=plt.subplots()
x=list(range(1,21))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



##TRAINING THE MODEL WITH MORE NUMBER OF HIDDEN CELLS IN A LAYER WITH

HIGHER DROPUTS TO REDUCE THE CHANCES OF OVER FITTING WE HAVE ALSO DONE THE BACHNORMALISATION FOR THE LAYER.

In [74]:

```
modelnew = Sequential()

modelnew.add(LSTM(100,return_sequences=True, input_shape=(timesteps, input_dim))
modelnew.add(Dropout(0.7))
modelnew.add(LSTM(300))
modelnew.add(Dropout(0.6))
modelnew.add(BatchNormalization())
modelnew.add(Dense(n_classes, activation='sigmoid'))
print(modelnew.summary())
modelnew.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['a
history= modelnew.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_t
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_14 (LSTM)               (None, 128, 100)          44000
_________________________________________________________________
dropout_14 (Dropout)         (None, 128, 100)          0
_________________________________________________________________
lstm_15 (LSTM)               (None, 300)               481200
_________________________________________________________________
dropout_15 (Dropout)         (None, 300)               0
_________________________________________________________________
batch_normalization_7 (Batch (None, 300)               1200
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 1806
=================================================================
Total params: 528,206
Trainable params: 527,606
Non-trainable params: 600
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 178s 24ms/step - loss: 1.0485 - ac
c: 0.5379 - val_loss: 0.8508 - val_acc: 0.5901
Epoch 2/20
7352/7352 [==============================] - 173s 24ms/step - loss: 0.8018 - ac
c: 0.6168 - val_loss: 0.7927 - val_acc: 0.5881
Epoch 3/20
7352/7352 [==============================] - 173s 24ms/step - loss: 0.7367 - ac
c: 0.6514 - val_loss: 0.9158 - val_acc: 0.6111
Epoch 4/20
7352/7352 [==============================] - 174s 24ms/step - loss: 0.7534 - ac
c: 0.6515 - val_loss: 0.7829 - val_acc: 0.6176
Epoch 5/20
7352/7352 [==============================] - 171s 23ms/step - loss: 0.7407 - ac
c: 0.6473 - val_loss: 0.7516 - val_acc: 0.6848
Epoch 6/20
7352/7352 [==============================] - 173s 24ms/step - loss: 0.9916 - ac
c: 0.5646 - val_loss: 1.3365 - val_acc: 0.3590
Epoch 7/20
7352/7352 [==============================] - 171s 23ms/step - loss: 0.9651 - ac
c: 0.5788 - val_loss: 0.8394 - val_acc: 0.6200
```

```
Epoch 8/20
7352/7352 [==============================] - 173s 23ms/step - loss: 0.7074 - ac
c: 0.6571 - val_loss: 0.8714 - val_acc: 0.6288
Epoch 9/20
7352/7352 [==============================] - 172s 23ms/step - loss: 0.6068 - ac
c: 0.7417 - val_loss: 0.7189 - val_acc: 0.7401
Epoch 10/20
7352/7352 [==============================] - 173s 23ms/step - loss: 0.7783 - ac
c: 0.7042 - val_loss: 0.6240 - val_acc: 0.8286
Epoch 11/20
7352/7352 [==============================] - 171s 23ms/step - loss: 0.3421 - ac
c: 0.8876 - val_loss: 0.3119 - val_acc: 0.9063
Epoch 12/20
7352/7352 [==============================] - 172s 23ms/step - loss: 0.2260 - ac
c: 0.9245 - val_loss: 0.4051 - val_acc: 0.8677
Epoch 13/20
7352/7352 [==============================] - 173s 24ms/step - loss: 0.2243 - ac
c: 0.9272 - val_loss: 0.3466 - val_acc: 0.8853
Epoch 14/20
7352/7352 [==============================] - 171s 23ms/step - loss: 0.2024 - ac
c: 0.9321 - val_loss: 0.4686 - val_acc: 0.9043
Epoch 15/20
7352/7352 [==============================] - 172s 23ms/step - loss: 0.2707 - ac
c: 0.9155 - val_loss: 0.4589 - val_acc: 0.9009
Epoch 16/20
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1901 - ac
c: 0.9393 - val_loss: 0.5445 - val_acc: 0.8799
Epoch 17/20
7352/7352 [==============================] - 173s 23ms/step - loss: 0.1841 - ac
c: 0.9430 - val_loss: 0.5569 - val_acc: 0.8907
Epoch 18/20
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1891 - ac
c: 0.9404 - val_loss: 0.3501 - val_acc: 0.9111
Epoch 19/20
7352/7352 [==============================] - 171s 23ms/step - loss: 0.1722 - ac
c: 0.9393 - val_loss: 0.3812 - val_acc: 0.9074
Epoch 20/20
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1666 - ac
c: 0.9421 - val_loss: 0.5799 - val_acc: 0.8812
```

In [75]:

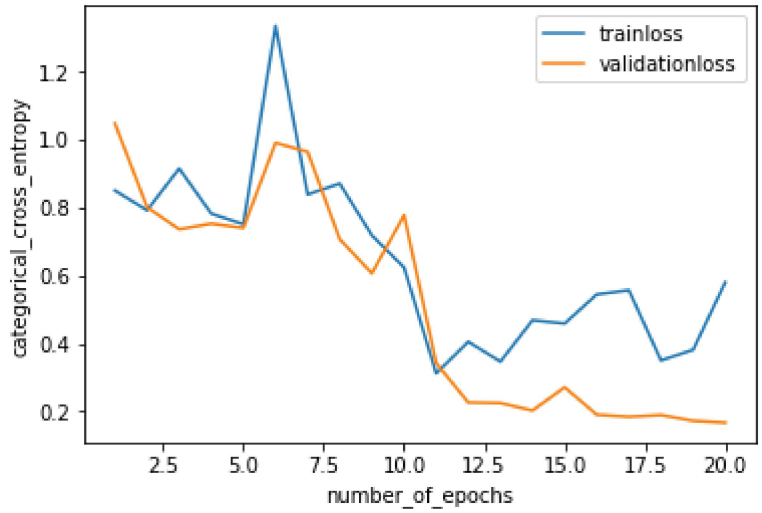
```
score=modelnew.evaluate(X_test, Y_test,verbose=0)
print('test score',score[0])
print('test accuracy',score[1])
```

```
test score 0.5799294071928931
test accuracy 0.8812351543942993
```

In [76]:
```
print(confusion_matrix(Y_test, modelnew.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  4      304       182        0                   0
STANDING                 0       24       508        0                   0
WALKING                  0        0         0      386                  68
WALKING_DOWNSTAIRS       0        0         0        1                 419
WALKING_UPSTAIRS         0        0         0        0                  28

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                           42
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 443
```

In [77]:
```
fig,ax=plt.subplots()
x=list(range(1,21))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



CHANGING THE ACTIVATION FUNCTION TO SIGMOID SINCE THE SIGMOID GIVES THE OUTPUT PROBABILTIES THAT BELONGING TO THE PARTICULAR CLASS LABEL.WITH SAME CONGIGURATION OF PREVIOUS MODEL WE CHANGE THE ACTIVATION FUNCTION AND CHECK.

In [25]:

```
modelnew = Sequential()

modelnew.add(LSTM(100,return_sequences=True, input_shape=(timesteps, input_dim))
modelnew.add(Dropout(0.7))
modelnew.add(LSTM(300))
modelnew.add(Dropout(0.6))
modelnew.add(BatchNormalization())

modelnew.add(Dense(n_classes, activation='softmax'))
print(modelnew.summary())
modelnew.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['a

history= modelnew.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_t
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/pytho
n/framework/op_def_library.py:263: colocate_with (from tensorflow.python.framew
ork.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) w
ith keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 100)          44000
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 100)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 300)               481200
_________________________________________________________________
dropout_2 (Dropout)          (None, 300)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 300)               1200
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1806
=================================================================
Total params: 528,206
Trainable params: 527,606
Non-trainable params: 600
_________________________________________________________________
None
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/pytho
n/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is depre
cated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 159s 22ms/step - loss: 1.0285 - ac
c: 0.6092 - val_loss: 0.9204 - val_acc: 0.6614
Epoch 2/20
```

```
7352/7352 [==============================] - 155s 21ms/step - loss: 0.4750 - ac
c: 0.8399 - val_loss: 0.6490 - val_acc: 0.7984
Epoch 3/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.3075 - ac
c: 0.8981 - val_loss: 0.4384 - val_acc: 0.8466
Epoch 4/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.2351 - ac
c: 0.9253 - val_loss: 0.3214 - val_acc: 0.8955
Epoch 5/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.2041 - ac
c: 0.9309 - val_loss: 0.4520 - val_acc: 0.8670
Epoch 6/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.1908 - ac
c: 0.9343 - val_loss: 0.3257 - val_acc: 0.9077
Epoch 7/20
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1880 - ac
c: 0.9347 - val_loss: 1.1457 - val_acc: 0.8012
Epoch 8/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.1784 - ac
c: 0.9380 - val_loss: 0.5234 - val_acc: 0.8853
Epoch 9/20
7352/7352 [==============================] - 156s 21ms/step - loss: 0.1815 - ac
c: 0.9396 - val_loss: 0.3384 - val_acc: 0.9070
Epoch 10/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.1878 - ac
c: 0.9366 - val_loss: 0.3260 - val_acc: 0.9070
Epoch 11/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.2112 - ac
c: 0.9317 - val_loss: 0.4242 - val_acc: 0.9087
Epoch 12/20
7352/7352 [==============================] - 155s 21ms/step - loss: 0.2022 - ac
c: 0.9350 - val_loss: 0.5830 - val_acc: 0.8884
Epoch 13/20
7352/7352 [==============================] - 156s 21ms/step - loss: 0.1888 - ac
c: 0.9373 - val_loss: 0.5046 - val_acc: 0.9019
Epoch 14/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.1659 - ac
c: 0.9389 - val_loss: 0.7242 - val_acc: 0.8884
Epoch 15/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.1738 - ac
c: 0.9400 - val_loss: 0.5111 - val_acc: 0.8962
Epoch 16/20
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1807 - ac
c: 0.9438 - val_loss: 0.5994 - val_acc: 0.8989
Epoch 17/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.1601 - ac
c: 0.9452 - val_loss: 0.4585 - val_acc: 0.8921
Epoch 18/20
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1549 - ac
c: 0.9453 - val_loss: 0.8446 - val_acc: 0.7621
Epoch 19/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.2162 - ac
c: 0.9329 - val_loss: 0.4736 - val_acc: 0.9046
Epoch 20/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.1664 - ac
c: 0.9426 - val_loss: 0.4776 - val_acc: 0.9209
```

In [26]:
```
score=modelnew.evaluate(X_test, Y_test,verbose=0)
print('test score',score[0])
print('test accuracy',score[1])
```

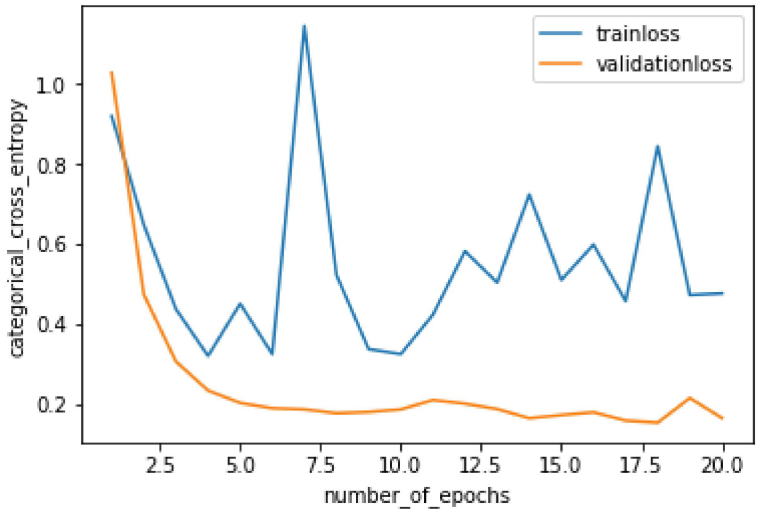
```
test score 0.47755399492043493
test accuracy 0.9209365456396336
```

In [29]:
```
print(confusion_matrix(Y_test, modelnew.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	\
True						
LAYING	537	0	0	0	0	
SITTING	3	414	74	0	0	
STANDING	0	86	446	0	0	
WALKING	1	1	0	448	44	
WALKING_DOWNSTAIRS	0	0	0	1	419	
WALKING_UPSTAIRS	1	7	2	6	5	

Pred	WALKING_UPSTAIRS
True	
LAYING	0
SITTING	0
STANDING	0
WALKING	2
WALKING_DOWNSTAIRS	0
WALKING_UPSTAIRS	450

In [31]:
```
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
x=list(range(1,21))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



USING THE SOFTMAX IN CASE OF SIGMOID ACTUALLY THE TEST

ACCURACY REDUCED WITH SIGMOID WE GOT 0.91 WITH SOFTMAX WE GOT 0.892. WE WILL TRY WITH RELU ACTIVATION FUNCTION WITH THE SAME CONFIGUTRATION.

In [32]:

```
modela = Sequential()

modela.add(LSTM(100,return_sequences=True, input_shape=(timesteps, input_dim)))
modela.add(Dropout(0.7))
modela.add(LSTM(300))
modela.add(Dropout(0.6))
modela.add(BatchNormalization())
modela.add(Dense(n_classes, activation='relu'))
print(modela.summary())
modela.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['acc

history= modela.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_tes
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 100)          44000
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 100)          0
_________________________________________________________________
lstm_4 (LSTM)                (None, 300)               481200
_________________________________________________________________
dropout_4 (Dropout)          (None, 300)               0
_________________________________________________________________
batch_normalization_2 (Batch (None, 300)               1200
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 1806
=================================================================
Total params: 528,206
Trainable params: 527,606
Non-trainable params: 600
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 160s 22ms/step - loss: 2.1252 - ac
c: 0.4659 - val_loss: 1.2260 - val_acc: 0.6091
Epoch 2/20
7352/7352 [==============================] - 156s 21ms/step - loss: 1.3158 - ac
c: 0.5964 - val_loss: 2.1774 - val_acc: 0.5721
Epoch 3/20
7352/7352 [==============================] - 158s 22ms/step - loss: 1.3191 - ac
c: 0.6215 - val_loss: 1.6510 - val_acc: 0.6121
Epoch 4/20
7352/7352 [==============================] - 156s 21ms/step - loss: 1.3987 - ac
c: 0.6349 - val_loss: 1.3484 - val_acc: 0.6359
Epoch 5/20
7352/7352 [==============================] - 159s 22ms/step - loss: 1.2176 - ac
c: 0.6831 - val_loss: 1.9444 - val_acc: 0.6678
Epoch 6/20
7352/7352 [==============================] - 156s 21ms/step - loss: 1.3150 - ac
c: 0.6912 - val_loss: 2.1620 - val_acc: 0.6620
Epoch 7/20
7352/7352 [==============================] - 158s 22ms/step - loss: 1.3720 - ac
c: 0.7114 - val_loss: 1.9954 - val_acc: 0.7363
```

```
Epoch 8/20
7352/7352 [==============================] - 155s 21ms/step - loss: 1.2773 - ac
c: 0.7431 - val_loss: 1.3247 - val_acc: 0.7282
Epoch 9/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.9720 - ac
c: 0.8067 - val_loss: 1.1600 - val_acc: 0.8286
Epoch 10/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.9090 - ac
c: 0.8874 - val_loss: 1.8553 - val_acc: 0.8436
Epoch 11/20
7352/7352 [==============================] - 156s 21ms/step - loss: 1.3277 - ac
c: 0.8677 - val_loss: 1.5042 - val_acc: 0.8700
Epoch 12/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.8304 - ac
c: 0.9006 - val_loss: 1.2530 - val_acc: 0.8622
Epoch 13/20
7352/7352 [==============================] - 158s 21ms/step - loss: 0.8321 - ac
c: 0.9042 - val_loss: 1.7096 - val_acc: 0.8585
Epoch 14/20
7352/7352 [==============================] - 154s 21ms/step - loss: 0.8371 - ac
c: 0.8938 - val_loss: 2.2182 - val_acc: 0.7299
Epoch 15/20
7352/7352 [==============================] - 157s 21ms/step - loss: 0.9504 - ac
c: 0.8962 - val_loss: 2.8795 - val_acc: 0.7645
Epoch 16/20
7352/7352 [==============================] - 153s 21ms/step - loss: 0.8878 - ac
c: 0.9023 - val_loss: 1.7750 - val_acc: 0.8042
Epoch 17/20
7352/7352 [==============================] - 156s 21ms/step - loss: 1.2889 - ac
c: 0.8713 - val_loss: 2.7426 - val_acc: 0.8117
Epoch 18/20
7352/7352 [==============================] - 153s 21ms/step - loss: 2.9924 - ac
c: 0.7046 - val_loss: 6.0092 - val_acc: 0.6132
Epoch 19/20
7352/7352 [==============================] - 156s 21ms/step - loss: 3.3477 - ac
c: 0.6115 - val_loss: 5.4340 - val_acc: 0.3363
Epoch 20/20
7352/7352 [==============================] - 154s 21ms/step - loss: 2.5352 - ac
c: 0.4474 - val_loss: 3.2740 - val_acc: 0.6121
```

In [33]:

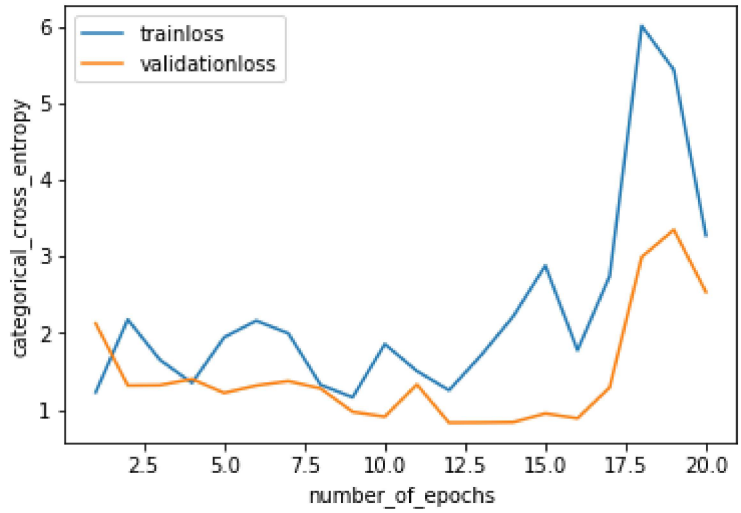
```
score=modelnew.evaluate(X_test, Y_test,verbose=0)
print('test score',score[0])
print('test accuracy',score[1])
```

```
test score 0.47755399492043493
test accuracy 0.9209365456396336
```

In [34]:
```
print(confusion_matrix(Y_test, modelnew.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  3      414        74        0                   0
STANDING                 0       86       446        0                   0
WALKING                  1        1         0      448                  44
WALKING_DOWNSTAIRS       0        0         0        1                 419
WALKING_UPSTAIRS         1        7         2        6                   5

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            0
STANDING                           0
WALKING                            2
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 450
```

In [35]:
```
fig,ax=plt.subplots()
x=list(range(1,21))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



WE ACHIEVED 91.6 TEST ACCURACY USING THE RELUAS THE ACTIVATION FUNCTION.

WE GONNA USE THE SINGLE LSTM LAYER SIMPLE MODEL AND ADD DROPOUT OT THE LAYER.

In [36]:
```
model3 = Sequential()
# Configuring the parameters
model3.add(LSTM(64, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model3.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model3.add(Dense(n_classes, activation='sigmoid'))
print(model3.summary())

# Compiling the model
model3.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['acc

# Training the model
history= model3.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_tes
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 80s 11ms/step - loss: 1.2424 - ac
c: 0.4739 - val_loss: 1.0771 - val_acc: 0.5212
Epoch 2/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.9438 - ac
c: 0.5826 - val_loss: 0.8838 - val_acc: 0.6535
Epoch 3/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.8002 - ac
c: 0.6474 - val_loss: 0.8034 - val_acc: 0.6960
Epoch 4/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.6630 - ac
c: 0.7349 - val_loss: 0.7763 - val_acc: 0.7106
Epoch 5/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.6058 - ac
c: 0.7622 - val_loss: 0.6271 - val_acc: 0.7913
Epoch 6/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.4932 - ac
c: 0.8248 - val_loss: 0.5687 - val_acc: 0.8049
Epoch 7/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.3686 - ac
c: 0.8810 - val_loss: 0.4586 - val_acc: 0.8463
Epoch 8/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.3234 - ac
c: 0.8962 - val_loss: 0.6586 - val_acc: 0.8100
Epoch 9/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.2582 - ac
```

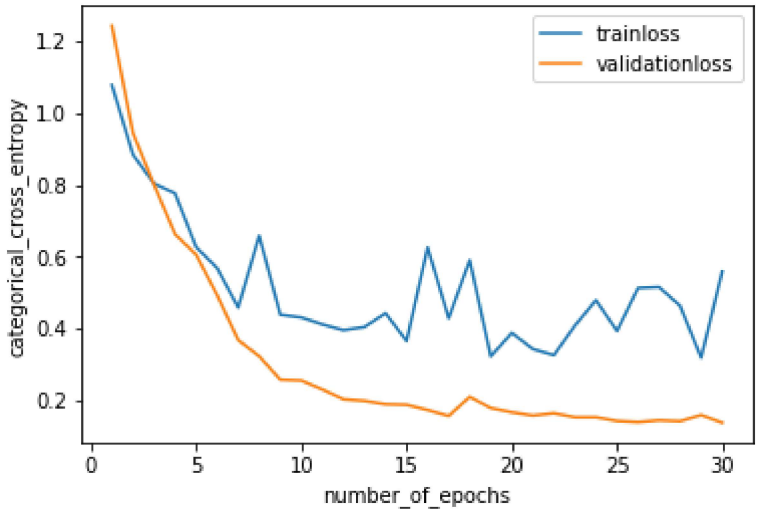
```
c: 0.9196 - val_loss: 0.4392 - val_acc: 0.8643
Epoch 10/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.2559 - ac
c: 0.9187 - val_loss: 0.4315 - val_acc: 0.8700
Epoch 11/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.2310 - ac
c: 0.9274 - val_loss: 0.4122 - val_acc: 0.8863
Epoch 12/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.2037 - ac
c: 0.9347 - val_loss: 0.3960 - val_acc: 0.8768
Epoch 13/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.1992 - ac
c: 0.9339 - val_loss: 0.4048 - val_acc: 0.8823
Epoch 14/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.1899 - ac
c: 0.9372 - val_loss: 0.4433 - val_acc: 0.8843
Epoch 15/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1885 - ac
c: 0.9319 - val_loss: 0.3658 - val_acc: 0.8935
Epoch 16/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1740 - ac
c: 0.9408 - val_loss: 0.6261 - val_acc: 0.8918
Epoch 17/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.1576 - ac
c: 0.9445 - val_loss: 0.4291 - val_acc: 0.8951
Epoch 18/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.2104 - ac
c: 0.9381 - val_loss: 0.5909 - val_acc: 0.8894
Epoch 19/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1794 - ac
c: 0.9411 - val_loss: 0.3231 - val_acc: 0.8958
Epoch 20/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.1678 - ac
c: 0.9448 - val_loss: 0.3884 - val_acc: 0.8823
Epoch 21/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1588 - ac
c: 0.9478 - val_loss: 0.3436 - val_acc: 0.9036
Epoch 22/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.1651 - ac
c: 0.9430 - val_loss: 0.3266 - val_acc: 0.9101
Epoch 23/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.1540 - ac
c: 0.9476 - val_loss: 0.4088 - val_acc: 0.9023
Epoch 24/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1540 - ac
c: 0.9479 - val_loss: 0.4793 - val_acc: 0.8941
Epoch 25/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1437 - ac
c: 0.9486 - val_loss: 0.3938 - val_acc: 0.9036
Epoch 26/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.1404 - ac
c: 0.9502 - val_loss: 0.5132 - val_acc: 0.8965
Epoch 27/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.1455 - ac
c: 0.9498 - val_loss: 0.5158 - val_acc: 0.9067
Epoch 28/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1431 - ac
```

```
c: 0.9509 - val_loss: 0.4635 - val_acc: 0.9023
Epoch 29/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.1601 - ac
c: 0.9491 - val_loss: 0.3193 - val_acc: 0.9019
Epoch 30/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.1388 - ac
c: 0.9509 - val_loss: 0.5589 - val_acc: 0.8894
```

In [37]:
```
scores3 = model3.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores3[0]))
print("Test Accuracy: %f%%" % (scores3[1]*100))
```

```
Test Score: 0.559273
Test Accuracy: 88.937903%
```

In [39]:
```
fig,ax=plt.subplots()
x=list(range(1,31))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



In [40]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model3.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        24        0                   0
SITTING                  0      388        97        2                   0
STANDING                 0      101       421        4                   0
WALKING                  0        0         0      464                  19
WALKING_DOWNSTAIRS       0        0         0       22                 389
WALKING_UPSTAIRS         0        0         0       19                   3

Pred                WALKING_UPSTAIRS
True
LAYING                             3
SITTING                            4
STANDING                           6
WALKING                           13
WALKING_DOWNSTAIRS                 9
WALKING_UPSTAIRS                 449
```

###SINGLE LSTM LAYER WITH MPORE NUMBER OF CELLS IN EACH HIDDEN LAYER

In [41]:
```
model3 = Sequential()
# Configuring the parameters
model3.add(LSTM(300, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model3.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model3.add(Dense(n_classes, activation='softmax'))
print(model3.summary())

# Compiling the model
model3.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accura

# Training the model
history3 = model3.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_t
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 300)               372000
_________________________________________________________________
dropout_6 (Dropout)          (None, 300)               0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 1806
=================================================================
Total params: 373,806
Trainable params: 373,806
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 77s 10ms/step - loss: 1.3546 - ac
c: 0.4266 - val_loss: 1.4051 - val_acc: 0.3661
Epoch 2/30
7352/7352 [==============================] - 76s 10ms/step - loss: 1.3603 - ac
c: 0.3943 - val_loss: 1.2473 - val_acc: 0.4832
Epoch 3/30
7352/7352 [==============================] - 75s 10ms/step - loss: 1.0384 - ac
c: 0.5544 - val_loss: 1.2564 - val_acc: 0.4578
Epoch 4/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.9501 - ac
c: 0.5812 - val_loss: 0.8030 - val_acc: 0.6403
Epoch 5/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.7200 - ac
c: 0.6696 - val_loss: 0.7198 - val_acc: 0.6518
Epoch 6/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.6276 - ac
c: 0.7114 - val_loss: 0.8023 - val_acc: 0.6016
Epoch 7/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.6397 - ac
c: 0.7138 - val_loss: 1.4114 - val_acc: 0.4581
Epoch 8/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.6261 - ac
c: 0.7295 - val_loss: 0.5974 - val_acc: 0.7727
Epoch 9/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.4491 - ac
```

```
c: 0.8184 - val_loss: 0.5478 - val_acc: 0.8015
Epoch 10/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.4214 - ac
c: 0.8267 - val_loss: 0.4746 - val_acc: 0.8392
Epoch 11/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.3519 - ac
c: 0.8706 - val_loss: 0.4821 - val_acc: 0.8609
Epoch 12/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.2846 - ac
c: 0.8946 - val_loss: 0.4432 - val_acc: 0.8602
Epoch 13/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2434 - ac
c: 0.9151 - val_loss: 0.3223 - val_acc: 0.8965
Epoch 14/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2213 - ac
c: 0.9123 - val_loss: 0.3278 - val_acc: 0.9016
Epoch 15/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1661 - ac
c: 0.9332 - val_loss: 0.3573 - val_acc: 0.8717
Epoch 16/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1653 - ac
c: 0.9377 - val_loss: 0.3209 - val_acc: 0.8958
Epoch 17/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1550 - ac
c: 0.9415 - val_loss: 0.3320 - val_acc: 0.8979
Epoch 18/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1458 - ac
c: 0.9414 - val_loss: 0.3719 - val_acc: 0.8894
Epoch 19/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1421 - ac
c: 0.9437 - val_loss: 0.3194 - val_acc: 0.8996
Epoch 20/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1260 - ac
c: 0.9464 - val_loss: 0.3468 - val_acc: 0.9135
Epoch 21/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1377 - ac
c: 0.9452 - val_loss: 0.3723 - val_acc: 0.8972
Epoch 22/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1458 - ac
c: 0.9456 - val_loss: 0.3059 - val_acc: 0.9084
Epoch 23/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1386 - ac
c: 0.9438 - val_loss: 0.3887 - val_acc: 0.8941
Epoch 24/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1660 - ac
c: 0.9353 - val_loss: 0.3356 - val_acc: 0.9097
Epoch 25/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1354 - ac
c: 0.9404 - val_loss: 0.3282 - val_acc: 0.9091
Epoch 26/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1296 - ac
c: 0.9471 - val_loss: 0.3325 - val_acc: 0.9131
Epoch 27/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1340 - ac
c: 0.9480 - val_loss: 0.3672 - val_acc: 0.9101
Epoch 28/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1246 - ac
```

```
c: 0.9495 - val_loss: 0.3432 - val_acc: 0.9206
Epoch 29/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1309 - ac
c: 0.9461 - val_loss: 0.3198 - val_acc: 0.9016
Epoch 30/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1248 - ac
c: 0.9463 - val_loss: 0.4055 - val_acc: 0.9009
```

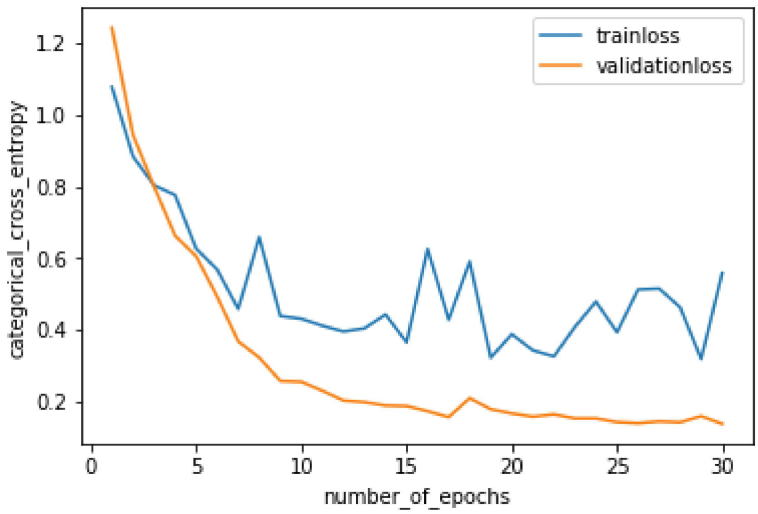
In [42]:
```
scores3 = model3.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores3[0]))
print("Test Accuracy: %f%%" % (scores3[1]*100))
```

```
Test Score: 0.405478
Test Accuracy: 90.091619%
```

In [43]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model3.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         1        0                   0
SITTING                  0      376       111        2                   0
STANDING                 0       86       444        2                   0
WALKING                  0        0         0      439                  45
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         1        0         0        2                   2

Pred                WALKING_UPSTAIRS
True
LAYING                            26
SITTING                            2
STANDING                           0
WALKING                           12
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 466
```

In [45]:
```
fig,ax=plt.subplots()
x=list(range(1,31))
validationy=history.history['val_loss']
testy=history.history['loss']
errorplot(x,validationy,testy,ax)
```



In [46]:
```
import pandas as pd
data=[[1,(32,6),'YES','YES',0.9,0.3,'SIGMOID'],[3,(64,300,200,6),'YES','YES',0.9
pd.DataFrame(data,columns=['NUMBER_OF_LAYERS_OF_LSTM','CONFIGURATION_OF_HIDDEN_C
```

Out[46]:

	NUMBER_OF_LAYERS_OF_LSTM	CONFIGURATION_OF_HIDDEN_CELLS	DROPOUT	BATCHNORMA
0	1	(32, 6)	YES	
1	3	(64, 300, 200, 6)	YES	
2	2	(32, 32, 6)	YES	
3	2	(64, 64, 6)	YES	
4	2	(100, 300, 6)	YES	
5	2	(100, 300, 6)	YES	
6	2	(100, 300, 6)	YES	
7	1	(64, 6)	YES	
8	1	(300, 6)	YES	

CONCLUSION DOCUMENTATION AND KEYTAKEAWAYS

WE ARE PROVIDED WITHE DATA OF RAW SIGNALS AND THE CLASS LABEL IS MULTILABELLED.THE DATA IS OBTAINED FROM THE ACCELEROMETER AND GYROSCOPE SENSORS IN THE PHONE.GYROSCOPE IS USED TO MEASURE THE ANGULAR

VELOCITYAND ACCELEROMETER USED TO MEASURE THE ACCELERATION WE WILL PREDICT WHAT THE PERSON IS PERFORMING.

THE ACCELEROMETERS ARE TRIAXIAL ANDGYROSCOPE IS TRIAXIAL.GIVEN THE TIMESERIES DATA TO PREDICT THE ACTIVITY. WE HAVE BUILD MODELS USING THE LSTM LAYERS AND THE EXTRACTING THE FEATURES FROM THE GIVEN DATA.USING THE FEATURES WE HAVE USED THE MACHINE LEARNING MODELSAND USED MULTICLASSLOGLOSS AS THE METRIC.

##USING THE SIMPLE LSTM LAYERS WITH OUT ANY FEATURE EXTACTION AND FEATURE ENGINEERING WE ARE ABLE TO ACHIEVE THE ACCURACY OF 92 PERCENTAGE

In [0]: