In [0]:

```
# Importing Libraries
```

In [1]:

```
import pandas as pd
import numpy as np
# Load the Drive helper and mount
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%
b&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.
2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fww
ogleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

In [0]:

```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}

# Utility function to print the confusion matrix
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'])
```

Data

In [0]:

```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [0]:

```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
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]:

```
# Utility function to read the data from csv file
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 SIGNALS:
        filename = f'drive/My Drive/UCI_HAR_Dataset/{subset}/Inertial
Signals/{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9 signals)
    return np.transpose(signals_data, (1, 2, 0))
```

In [0]:

```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    filename = f'drive/My Drive/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[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_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

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 [10]:

```
# 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.
```

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 = [16,32,74,128]
dropout = [0.25 , 0.5]
```

In [0]:

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

In [0]:

```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

In [16]:

```
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
```

- Defining the Architecture of LSTM

In [0]:

```
from keras.optimizers import Adam,RMSprop,SGD
def best_hyperparameters(n_hidden,dropout):

    model = Sequential()
    model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
    model.add(Dropout(dropout))
    model.add(Dense(n_classes, activation='sigmoid'))


    model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

    return model
```

In [0]:

```
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
```

```
from keras.preprocessing import sequence
from sklearn.model_selection import GridSearchCV

model = KerasClassifier(build_fn=best_hyperparameters,epochs=epochs, batch_size=batch_size, verbose
=0)
param_grid = dict(n_hidden=n_hidden,dropout = dropout)

# if you are using CPU
# grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
# if you are using GPU dont use the n_jobs parameter

grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid_result = grid.fit(X_train,Y_train)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:2053: FutureWarning: You
should specify a value for 'cv' instead of relying on the default value. The default value will ch
ange from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from
tensorflow.python.ops.nn_ops) with 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 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
```

In [0]:

```
grid_result.best_params_
```

Out[0]:

```
{'dropout': 0.25, 'n_hidden': 74}
```

In [0]:

```
dropout=0.25
n_hidden = 74
# 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(dropout))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_26 (LSTM)               (None, 74)                24864
_________________________________________________________________
dropout_26 (Dropout)         (None, 74)                0
_________________________________________________________________
dense_26 (Dense)             (None, 6)                 450
=================================================================
Total params: 25,314
Trainable params: 25,314
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

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

In [0]:

```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1302 - acc: 0.9517 - val_loss:
0.2794 - val_acc: 0.9091
Epoch 2/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1237 - acc: 0.9542 - val_loss:
0.4070 - val_acc: 0.9121
Epoch 3/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1141 - acc: 0.9551 - val_loss:
0.5153 - val_acc: 0.9036
Epoch 4/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1137 - acc: 0.9546 - val_loss:
0.3607 - val_acc: 0.9077
Epoch 5/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1277 - acc: 0.9543 - val_loss:
0.5015 - val_acc: 0.8965
Epoch 6/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1175 - acc: 0.9532 - val_loss:
0.3453 - val_acc: 0.9162
Epoch 7/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1146 - acc: 0.9527 - val_loss:
0.4091 - val_acc: 0.9199
Epoch 8/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1262 - acc: 0.9516 - val_loss:
0.3926 - val_acc: 0.9125
Epoch 9/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1376 - acc: 0.9491 - val_loss:
0.3806 - val_acc: 0.9101
Epoch 10/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1159 - acc: 0.9539 - val_loss:
0.6433 - val_acc: 0.9026
Epoch 11/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1230 - acc: 0.9509 - val_loss:
0.4740 - val_acc: 0.9026
Epoch 12/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1305 - acc: 0.9538 - val_loss:
0.4104 - val_acc: 0.9135
Epoch 13/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1153 - acc: 0.9543 - val_loss:
0.4176 - val_acc: 0.9169
Epoch 14/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1201 - acc: 0.9514 - val_loss:
0.3889 - val_acc: 0.9175
Epoch 15/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1136 - acc: 0.9529 - val_loss:
0.4002 - val_acc: 0.9175
Epoch 16/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1221 - acc: 0.9547 - val_loss:
0.4470 - val_acc: 0.9057
Epoch 17/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1109 - acc: 0.9578 - val_loss:
0.3949 - val_acc: 0.9199
Epoch 18/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1146 - acc: 0.9576 - val_loss:
0.4245 - val_acc: 0.9111
Epoch 19/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1171 - acc: 0.9559 - val_loss:
0.4583 - val_acc: 0.9131
Epoch 20/30
```

```
Epoch 20/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1211 - acc: 0.9509 - val_loss:
0.3517 - val_acc: 0.9226
Epoch 21/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1127 - acc: 0.9558 - val_loss:
0.3543 - val_acc: 0.9213
Epoch 22/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1352 - acc: 0.9502 - val_loss:
0.3221 - val_acc: 0.9125
Epoch 23/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1245 - acc: 0.9533 - val_loss:
0.4157 - val_acc: 0.9152
Epoch 24/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1606 - acc: 0.9484 - val_loss:
0.3254 - val_acc: 0.9260
Epoch 25/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1143 - acc: 0.9553 - val_loss:
0.3048 - val_acc: 0.9226
Epoch 26/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1226 - acc: 0.9533 - val_loss:
0.2602 - val_acc: 0.9206
Epoch 27/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1162 - acc: 0.9546 - val_loss:
0.3362 - val_acc: 0.9213
Epoch 28/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1255 - acc: 0.9528 - val_loss:
0.4265 - val_acc: 0.9111
Epoch 29/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1663 - acc: 0.9404 - val_loss:
0.4478 - val_acc: 0.9281
Epoch 30/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1193 - acc: 0.9529 - val_loss:
0.4303 - val_acc: 0.9172
```

In [0]:

```python
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        18        0                   0
SITTING                  0      415        74        0                   0
STANDING                 0       99       433        0                   0
WALKING                  0        0         0      477                   2
WALKING_DOWNSTAIRS       0        0         0        2                 409
WALKING_UPSTAIRS         1        0         2        9                   0

Pred                WALKING_UPSTAIRS
True
LAYING                             9
SITTING                            2
STANDING                           0
WALKING                           17
WALKING_DOWNSTAIRS                 9
WALKING_UPSTAIRS                 459
```

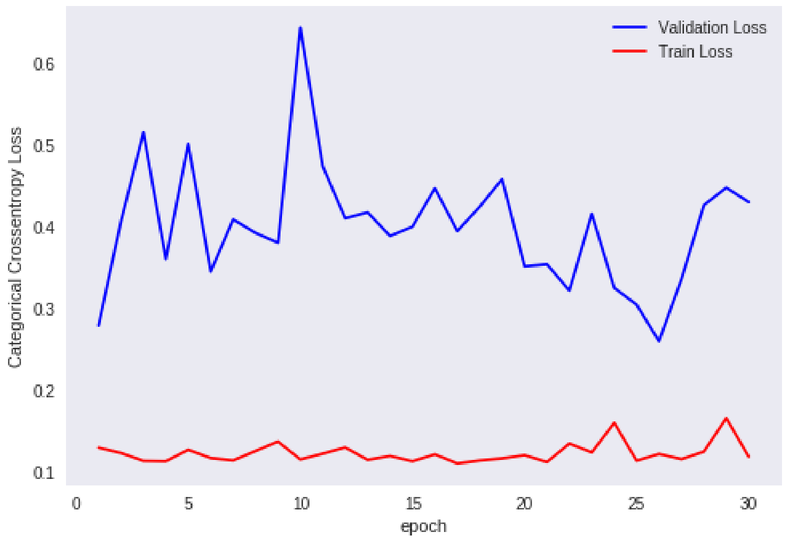
In [0]:

```python
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))


vy = history.history['val_loss']
ty = history.history['loss']
```

```
plt_dynamic(x, vy, ty, ax)
```



In [0]:

```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 676us/step
```

In [0]:

```
score
```

Out[0]:

```
[0.43029451220163206, 0.9172039362063115]
```

In [19]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(128, input_shape=(timesteps, input_dim),return_sequences=True))
# Adding a dropout layer
model.add(Dropout(0.75))
model.add(LSTM(74, input_shape=(timesteps, input_dim),return_sequences=True))
model.add(LSTM(32, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.75))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_6 (LSTM)                (None, 128, 128)          70656     
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 128)          0         
_________________________________________________________________
lstm_7 (LSTM)                (None, 128, 74)           60088     
_________________________________________________________________
lstm_8 (LSTM)                (None, 32)                13696     
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198       
=================================================================
Total params: 144,638
Trainable params: 144,638
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

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

In [21]:

```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 139s 19ms/step - loss: 1.3099 - acc: 0.4483 - val_loss: 1.3566 - val_acc: 0.4028
Epoch 2/30
7352/7352 [==============================] - 136s 19ms/step - loss: 1.0093 - acc: 0.5641 - val_loss: 0.8225 - val_acc: 0.5945
Epoch 3/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.8663 - acc: 0.6051 - val_loss: 0.8138 - val_acc: 0.5904
Epoch 4/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.8036 - acc: 0.6298 - val_loss: 0.7889 - val_acc: 0.6183
Epoch 5/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.8112 - acc: 0.6182 - val_loss: 0.9668 - val_acc: 0.5908
Epoch 6/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.8174 - acc: 0.6302 - val_loss: 0.8942 - val_acc: 0.6013
Epoch 7/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.7930 - acc: 0.6367 - val_loss: 0.8254 - val_acc: 0.6179
Epoch 8/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.7426 - acc: 0.6401 - val_loss: 0.7119 - val_acc: 0.6284
Epoch 9/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.7144 - acc: 0.6499 - val_loss: 0.6327 - val_acc: 0.6257
Epoch 10/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.6836 - acc: 0.6514 - val_loss: 0.6094 - val_acc: 0.6284
Epoch 11/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.6348 - acc: 0.6706 - val_loss: 0.6560 - val_acc: 0.6250
Epoch 12/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.6283 - acc: 0.6874 - val_loss: 0.6945 - val_acc: 0.6193
Epoch 13/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.6285 - acc: 0.7089 - val_loss: 0.6274 - val_acc: 0.7370
Epoch 14/30
7352/7352 [==============================] - 132s 18ms/step - loss: 0.8342 - acc: 0.6877 - val_loss: 1.6528 - val_acc: 0.3312
Epoch 15/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.9332 - acc: 0.6107 - val_loss: 0.6034 - val_acc: 0.7635
Epoch 16/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.5959 - acc: 0.7915 - val_loss: 0.4300 - val_acc: 0.8877
Epoch 17/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.4676 - acc: 0.8553 - val_loss: 0.3415 - val_acc: 0.8985
Epoch 18/30
```

```
7352/7352 [==============================] - 141s 19ms/step - loss: 0.4450 - acc: 0.8634 - val_los
s: 0.4780 - val_acc: 0.8755
Epoch 19/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.3892 - acc: 0.8776 - val_los
s: 0.3519 - val_acc: 0.9016
Epoch 20/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.3511 - acc: 0.8879 - val_los
s: 0.5373 - val_acc: 0.8809
Epoch 21/30
 480/7352 [>.............................] - ETA: 2:03 - loss: 0.4117 - acc:
0.8750WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 139s 19ms/step - loss: 1.3099 - acc: 0.4483 - val_los
s: 1.3566 - val_acc: 0.4028
Epoch 2/30
7352/7352 [==============================] - 136s 19ms/step - loss: 1.0093 - acc: 0.5641 - val_los
s: 0.8225 - val_acc: 0.5945
Epoch 3/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.8663 - acc: 0.6051 - val_los
s: 0.8138 - val_acc: 0.5904
Epoch 4/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.8036 - acc: 0.6298 - val_los
s: 0.7889 - val_acc: 0.6183
Epoch 5/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.8112 - acc: 0.6182 - val_los
s: 0.9668 - val_acc: 0.5908
Epoch 6/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.8174 - acc: 0.6302 - val_los
s: 0.8942 - val_acc: 0.6013
Epoch 7/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.7930 - acc: 0.6367 - val_los
s: 0.8254 - val_acc: 0.6179
Epoch 8/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.7426 - acc: 0.6401 - val_los
s: 0.7119 - val_acc: 0.6284
Epoch 9/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.7144 - acc: 0.6499 - val_los
s: 0.6327 - val_acc: 0.6257
Epoch 10/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.6836 - acc: 0.6514 - val_los
s: 0.6094 - val_acc: 0.6284
Epoch 11/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.6348 - acc: 0.6706 - val_los
s: 0.6560 - val_acc: 0.6250
Epoch 12/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.6283 - acc: 0.6874 - val_los
s: 0.6945 - val_acc: 0.6193
Epoch 13/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.6285 - acc: 0.7089 - val_los
s: 0.6274 - val_acc: 0.7370
Epoch 14/30
7352/7352 [==============================] - 132s 18ms/step - loss: 0.8342 - acc: 0.6877 - val_los
s: 1.6528 - val_acc: 0.3312
Epoch 15/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.9332 - acc: 0.6107 - val_los
s: 0.6034 - val_acc: 0.7635
Epoch 16/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.5959 - acc: 0.7915 - val_los
s: 0.4300 - val_acc: 0.8877
Epoch 17/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.4676 - acc: 0.8553 - val_los
s: 0.3415 - val_acc: 0.8985
Epoch 18/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.4450 - acc: 0.8634 - val_los
s: 0.4780 - val_acc: 0.8755
Epoch 19/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.3892 - acc: 0.8776 - val_los
s: 0.3519 - val_acc: 0.9016
Epoch 20/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.3511 - acc: 0.8879 - val_los
s: 0.5373 - val_acc: 0.8809
Epoch 21/30
```

```
7352/7352 [==============================] - 138s 19ms/step - loss: 0.3945 - acc: 0.8745 - val_los
s: 0.4426 - val_acc: 0.8860
7352/7352 [==============================] - 138s 19ms/step - loss: 0.3945 - acc: 0.8745 - val_los
s: 0.4426 - val_acc: 0.8860
Epoch 22/30
Epoch 22/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.3261 - acc: 0.8924 - val_los
s: 0.3322 - val_acc: 0.9036
7352/7352 [==============================] - 138s 19ms/step - loss: 0.3261 - acc: 0.8924 - val_los
s: 0.3322 - val_acc: 0.9036
Epoch 23/30
Epoch 23/30
7352/7352 [==============================] - 136s 19ms/step - loss: 0.3067 - acc: 0.9008 - val_los
s: 0.2949 - val_acc: 0.9179
7352/7352 [==============================] - 136s 19ms/step - loss: 0.3067 - acc: 0.9008 - val_los
s: 0.2949 - val_acc: 0.9179
Epoch 24/30
Epoch 24/30
7352/7352 [==============================] - 136s 19ms/step - loss: 0.3078 - acc: 0.9004 - val_los
s: 0.4534 - val_acc: 0.9057
7352/7352 [==============================] - 136s 19ms/step - loss: 0.3078 - acc: 0.9004 - val_los
s: 0.4534 - val_acc: 0.9057
Epoch 25/30
Epoch 25/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.3173 - acc: 0.9033 - val_los
s: 0.5568 - val_acc: 0.8778
7352/7352 [==============================] - 139s 19ms/step - loss: 0.3173 - acc: 0.9033 - val_los
s: 0.5568 - val_acc: 0.8778
Epoch 26/30
Epoch 26/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.3059 - acc: 0.8985 - val_los
s: 0.6705 - val_acc: 0.8744
7352/7352 [==============================] - 138s 19ms/step - loss: 0.3059 - acc: 0.8985 - val_los
s: 0.6705 - val_acc: 0.8744
Epoch 27/30
Epoch 27/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.3222 - acc: 0.8969 - val_los
s: 0.3312 - val_acc: 0.9182
7352/7352 [==============================] - 140s 19ms/step - loss: 0.3222 - acc: 0.8969 - val_los
s: 0.3312 - val_acc: 0.9182
Epoch 28/30
Epoch 28/30
7352/7352 [==============================] - 137s 19ms/step - loss: 0.2913 - acc: 0.9075 - val_los
s: 0.4033 - val_acc: 0.9148
7352/7352 [==============================] - 137s 19ms/step - loss: 0.2913 - acc: 0.9075 - val_los
s: 0.4033 - val_acc: 0.9148
Epoch 29/30
Epoch 29/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.2944 - acc: 0.9097 - val_los
s: 0.4813 - val_acc: 0.8928
7352/7352 [==============================] - 138s 19ms/step - loss: 0.2944 - acc: 0.9097 - val_los
s: 0.4813 - val_acc: 0.8928
Epoch 30/30
Epoch 30/30
7352/7352 [==============================] - 137s 19ms/step - loss: 0.3028 - acc: 0.9056 - val_los
s: 0.3878 - val_acc: 0.9125
7352/7352 [==============================] - 137s 19ms/step - loss: 0.3028 - acc: 0.9056 - val_los
s: 0.3878 - val_acc: 0.9125
```

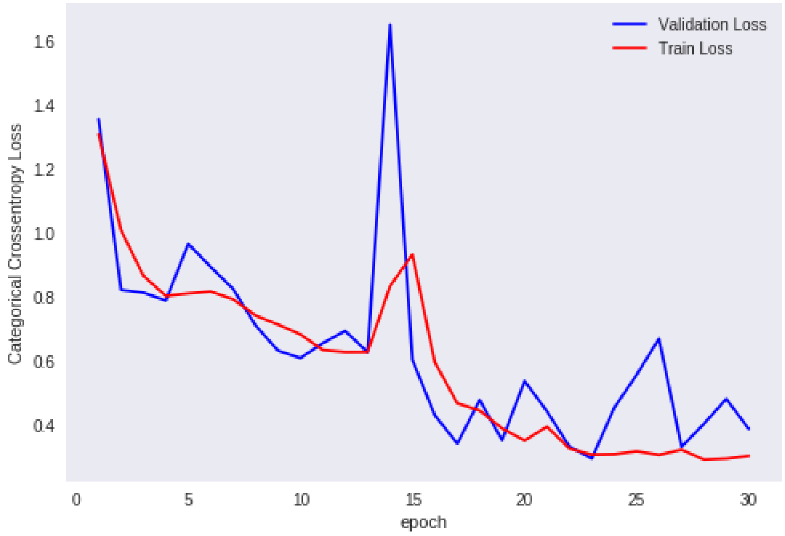
In [22]:

```python
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))


vy = history.history['val_loss']
```

```
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```



In [23]:

```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 7s 2ms/step
```

In [24]:

```
score
```

Out[24]:

```
[0.3877914281657706, 0.9124533423820834]
```

Exploratory Data Analysis

1)Importing all the packages which are required 2)Load the data 3)features are given by the experts 4)Remove duplicates 5)Remove null values 6)Checking for the data imbalance 7)Do some violin plots, box plots and twin plots on some features 8)try TSNE on data with different perplexities

Prediction models

1)Split the data in train and test 2)Apply Logistic regression on the data with hyper parameter tuning 3)Apply Linear SVC on the data with hyper parameter tuning 4)Apply Kernal SVM on the data with hyper parameter tuning 5)Apply Decision tree, Random forest classifierand Gradient boosted decision tree on the data with hyper parameter tuning 6)Comparing all the models

LSTM

1)Take the data and the time-series data and get the features from those time series data 2)Apply the LSTM with hyperparameter tuning and only with 2 layers 3)Trying it with more layers and getting the accuracy

In [1]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["layers","Train  loss","Test loss"]

x.add_row(["2",".11",".43"])
x.add_row(["4",".302",".387"])



print(x)
```

```
+--------+-------------+-----------+
| layers | Train  loss | Test loss |
+--------+-------------+-----------+
|   2    |     .11     |    .43    |
|   4    |     .302    |    .387   |
+--------+-------------+-----------+
```

- With a simple 2 layer architecture we got 90.09% accuracy and a loss of 0.30
- We can further imporve the performace with Hyperparameter tuning