Sheet

```
# !pip install keras_tuner
# !pip install pydot
# !pip install graphviz

import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.utils import np_utils
import tensorflow as tf
import keras_tuner as kt

np.random.seed(7)
```

```
x_data = pd.read_pickle('/data/workspace_files/data_df92.pickle')
y_data = pd.read_pickle('/data/workspace_files/rule_based.pickle').loc[x_data.index]
y_modified = pd.DataFrame(np_utils.to_categorical(y_data - y_data.min()), index=y_data.index)

from sklearn.preprocessing import StandardScaler
x_data = pd.DataFrame(StandardScaler().fit_transform(x_data),
                      columns=x_data.columns,
                      index=x_data.index)
```

```
y_modified.sum()
```

```
# load the dataset but only keep the top n words, zero the rest
train_ind = int(np.where(x_data.index >= pd.to_datetime('2018-01-01 00:00:00'))[0][0])
val_ind = int(train_ind*0.8)
X_train = x_data.iloc[:val_ind]
X_val = x_data.iloc[val_ind:train_ind]
X_test = x_data.iloc[train_ind:]

y_train = y_modified.iloc[:val_ind]
y_val = y_modified.iloc[val_ind:train_ind]
y_test = y_modified.iloc[train_ind:]
print("size of train data: ", X_train.shape[0],
      ", size of val data: ", X_val.shape[0],
      ", size of test data: ", X_test.shape[0])

X_train = tf.expand_dims(X_train, axis=1)
X_val = tf.expand_dims(X_val, axis=1)
X_test = tf.expand_dims(X_test, axis=1)
print(X_train.shape)
```

```
size of train data:  1080 , size of val data:  271 , size of test data:  210
(1080, 1, 38)
```

Hyperparameter Search

```
def build_model(hp):
    model = Sequential()
    model.add(LSTM(hp.Choice('units', [8, 16, 32, 64, 128]),
              input_shape=(X_train.shape[1], X_train.shape[2]),
              return_sequences=False))
    model.add(Dropout(hp.Choice('dropout', [0., 0.1, 0.2, 0.3, 0.4])))
    model.add(Dense(3, activation=hp.Choice(
                                'dense_activation',
                                values=['tanh', 'sigmoid'],
                                default='sigmoid'),))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=["CategoricalAccuracy"])
    return model
```

```
%%capture
tuner = kt.RandomSearch(
    build_model,
    objective='categorical_accuracy',
    max_trials=100, overwrite=True)

tuner.search(X_train, y_train, epochs=20, validation_data=(X_val, y_val))
best_model = tuner.get_best_models()[0]
```

```
INFO:tensorflow:Oracle triggered exit
```

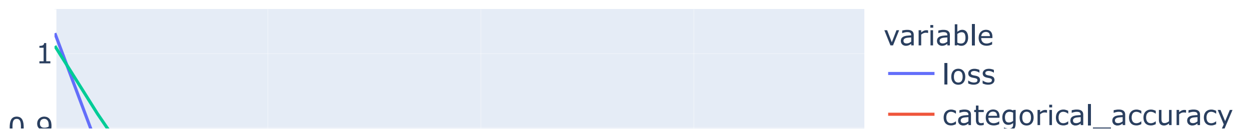
From the hyper parameter tuning, we found out the best architecture for our model is 128 LSTM nodes with 0.1 dropout and sigmoid activation function for the last dense layer.

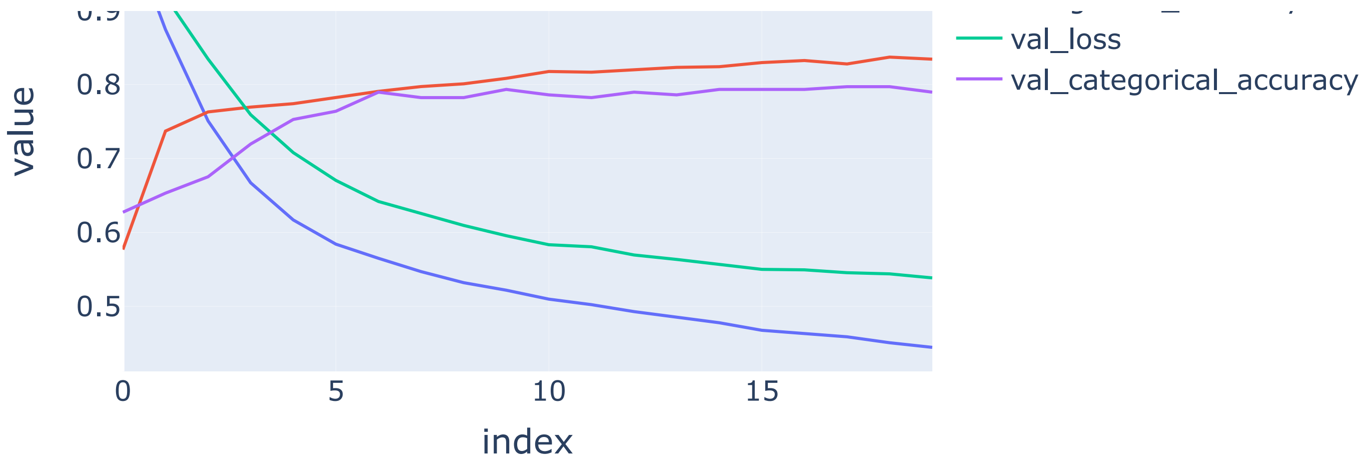
So, lets create the model, train and predict our output.

```
%%capture
model = Sequential()

model.add(LSTM(128,
          input_shape=(X_train.shape[1], X_train.shape[2]),
          return_sequences=False))
model.add(Dropout(0.1))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=["CategoricalAccuracy"])
print(model.summary())
model_res = model.fit(X_train,
                    y_train,
                    validation_data=(X_val, y_val),
                    epochs=20,
                    batch_size=60)
```

```
import plotly.express as px
fig = px.line(pd.DataFrame.from_dict(model_res.history))
fig.update_layout(font=dict(size=18))
# fig.update_layout(legend=dict(y = 1.0, x = 0.8))
```





```
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 64.76%
```

```
y_lstm = pd.DataFrame(
        pd.DataFrame(model.predict(X_test)).apply(lambda x:x.idxmax()-1, axis=1),
        columns=['lstm']
        )
y_lstm.index =y_test.index
```

dump the output to the output folder, which will be used in the ensemble method!

```
pd.to_pickle(y_lstm, '/data/workspace_files/output/lstm_test.pickle')
pd.to_pickle(y_data, '/data/workspace_files/output/lstm_train.pickle')
```