In [ ]:

```
#Importing Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.preprocessing import LabelEncoder
from sklearn.utils import shuffle

#Constants
#After functional day and 0 rows removal
TRAIN_SIZE = 6772
TEST_SIZE = 1693

ACTIVATION_F = 'tanh'

#Import Training Set
df = pd.read_csv('SeoulBikeData.csv',engine='python')

dummies = pd.get_dummies(df.Seasons)
df= pd.concat([df,dummies],axis='columns')
df= df.drop(['Seasons','Winter'], axis='columns')

le= LabelEncoder()
dfle = df
df.Holiday=le.fit_transform(dfle.Holiday)
df['Functioning Day']=le.fit_transform(dfle['Functioning Day'])

df=shuffle(df)

df=df[df['Functioning Day'] == 1]

df = df.drop(['Functioning Day'], axis='columns')

training_set_df= df.iloc[:TRAIN_SIZE, 1:]
training_set= df.iloc[:TRAIN_SIZE, 1:].values
y_set= df.iloc[:TRAIN_SIZE, 1].values

#Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc= MinMaxScaler(feature_range = (0,1))
training_set_scaled = sc.fit_transform(training_set)

sc2= MinMaxScaler(feature_range = (0,1))
y_set=y_set.reshape(-1,1)
y_set_scaled = sc2.fit_transform(y_set)


X_train=np.array(training_set_scaled[:,1:])
Y_train=np.array(training_set_scaled[:,0])


#reshaping
X_train = np.reshape(X_train, (X_train.shape[0], 1, 13))

# Importing the Keras libraries and packages
!pip install -q -U keras-tuner
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

```
from keras.layers import Dropout
from keras import backend as K
from kerastuner.tuners import RandomSearch
import keras


def coeff_determination(y_true, y_pred):
    SS_res =  K.sum(K.square( y_true-y_pred ))
    SS_tot = K.sum(K.square( y_true - K.mean(y_true) ) )
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )



def build_model(hp):
    regressor = Sequential()

    regressor.add(LSTM(units=hp.Int('units_', min_value=32, max_value=256, step=32), ac

    for i in range(hp.Int('num_layers', 0, 20)):
        regressor.add(LSTM(units=hp.Int('units_',
                                        min_value=32,
                                        max_value=256,
                                        step=32),
                           activation=ACTIVATION_F, return_sequences=True,))
        regressor.add(Dropout(0.1))

    regressor.add(LSTM(units=hp.Int('units_',min_value=32,max_value=256,step=32),activa
    regressor.add(Dropout(0.1))
    regressor.add(Dense(units=1))
    regressor.compile(
        optimizer=keras.optimizers.RMSprop(
            hp.Choice('learning_rate', [1e-2, 1e-3, 1e-4])),
        loss='mean_squared_error',
        )
    return regressor



tuner = RandomSearch(
    build_model,
    objective='val_loss',
    max_trials=10,
    executions_per_trial=2,
    directory='project',
    project_name='INNS')

tuner.search_space_summary()

tuner.search(X_train, Y_train,
             epochs=100,
             validation_split=0.2)
```

In [ ]:
```
tuner.results_summary()
```