TITLE: *MACHINE LEARNING ANALYSIS OF COP VARIATION OF GROUND COUPLED SPLIT A/C SYSTEM*

GROUP NO. : 33

SUPERVISOR : PROF. K. MURUGESAN

GROUP MEMBERS :

HARSHIT SOLANKI (20117058)

MAYANK (20117076)

SAHIL YADAV (20117105)

RAHUL MEENA (20117097)

GAIKWAD HIMANSHU ASHOK (20117050)

INTRODUCTION

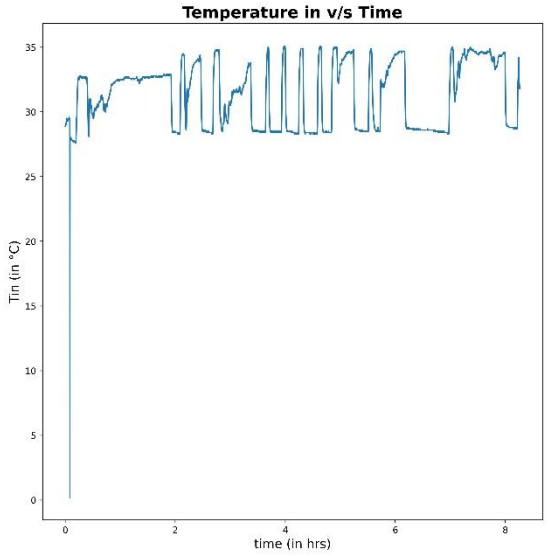
Energy efficient indoor climate of building enclosures can be achieved by using ground source heat pump systems [GSHP] through heat interactions with the ground instead of ambient air. The GSHP has gained popularity in western and European countries but in the recent past, GSHP has gained attention in Asian countries also like China, Korea , India, Japan. In order to achieve a relatively stable source/sink temperature, a vertical type of ground heat exchanger known as borehole heat exchanger is installed at a depth of 50 to 200m. The heat transfer to/from the BHE is a function of thermal conductivity, thermal diffusivity, of the ground formation, depth/diameter of BHE and thermophysical properties of grouting material used in boreholes. The rate of heat exchanged to/from the ground in a ground source heat pump system directly depends on the performance of BHE which is directly associated with its effectiveness. Therefore, the effectiveness of BHE plays an important role during the estimation of the performance of ground source heat pump systems. Zeng et al proposed a correlation for effectiveness of BHEs, for single U-tube borehole heat exchangers and concluded that quasi three-dimensional model showed better accuracy than two-dimensional model. They reported that the error due to thermal "SHORT CIRCULATING" among the tubes may increase upto 11% in a typical borehole heat exchanger. Further, Conti et al extended the same principle and proposed analytical correlations of effectiveness for single and double U-tube BHE connected in parallel modes yields better effectiveness as compared to the other types of BHE's . It is also found that double U-tube shows higher effectiveness than single U-tube BHE's. In the recent past, many authors have reported different methods for improving thermal effectiveness of BHE's. Chen at al proposed that with the increase in number of U-tubes in a BHE, the heat transfer area also increases due to which the effectiveness of BHE also increases. They also noticed that there was an increase in the thermal efficiency of ground heat exchangers as the thermal conductivity of grouting material increases. Similarly, KERME and FUNG found that with increase in mass flow rate of heat transfer fluid, the local heat transfer rate was increased and the effectiveness showed a decrement trend. Keeping a thermal balance between heat extraction in the heating season and heat rejection in the cooling season is a major issue in cooling dominated areas. Qi et al performed experiments to analyse the behaviour of GSHP with series and parallel configurations and compared their results with numerical results. They found that parallel configuration shows better thermal performance than that of the series configuration. The ability of a borehole heat exchanger is defined by its thermal effectiveness. In the past, many researchers have analysed the thermal effectiveness of BHE's assuming a constant borehole heat temperature. But the temperature increases with the

passage of time due to continuous absorption of heat by the ground. Though, expressions for effectiveness using transient BHE's are suggested. No research work has discussed the transient variation of Borehole effectiveness during actual operation of GSHP system.The present research work is an attempt to fill this research gap in which a novel approach is proposed to determine the real time variation of borehole temperature during actual operation of a GSHP system for cooling mode operation. In this method, the real time borehole temperature is computed by employing interface temperature data measured along the depth of the BHE during actual operation of the GSHP system. Heat rejection by the BHE during actual operation of a GSHP system continuously varies and hence, the use of line source model may not provide the realistic value of effectiveness of BHE. The effectiveness of double U-tube BHE obtained from constant heat source experiments and actual GSHP experiments were compared

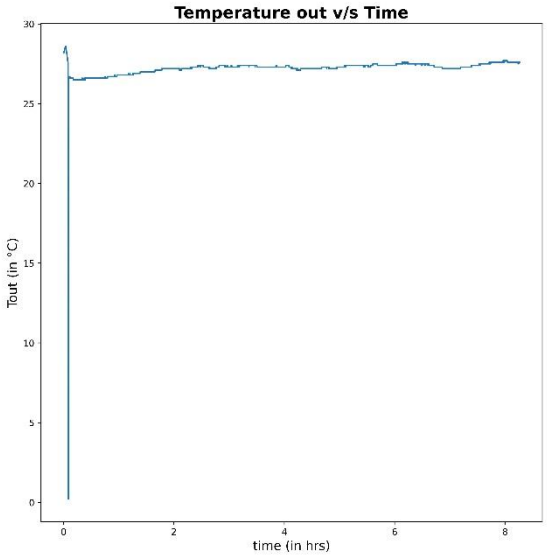
OBJECTIVE

The COP of ground coupled split Air-conditioning system varies continuously during the operation of the system. The COP depends on various parameters such as ambient temperatures, inlet and outlet temperatures of water flowing in the borehole heat exchanger and the temperature to be maintained inside the room. When the system is switched on initially there will be variation of the above said parameters, however, after reaching a steady temperature condition, the cooling effect and the COP of the system is expected to stabilize. However, in a real experiment, some variations of COP are noticed and this needs to be investigated in detail to understand the influencing parameters responsible for this variation. Machine learning analysis is widely applied to optimize the performance of many engineering systems. So, in our present project work an attempt will be made to study reasons for the variation of COP using a Machine learning app.

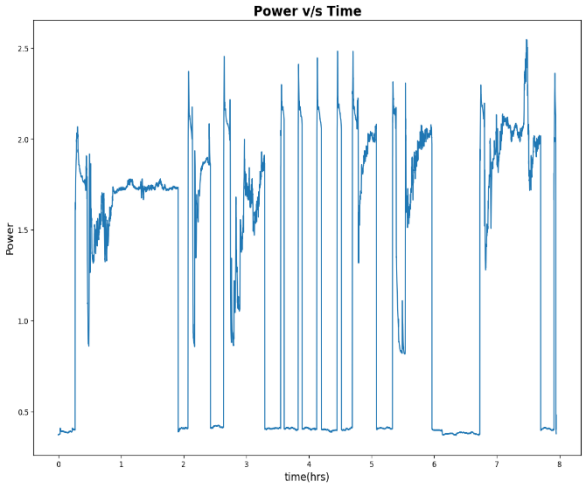
GRAPHS FOR OUR DATA



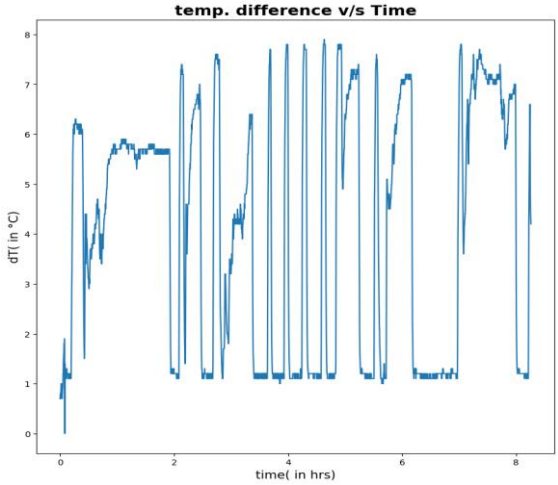
(Figure 1: Tin v/s time)



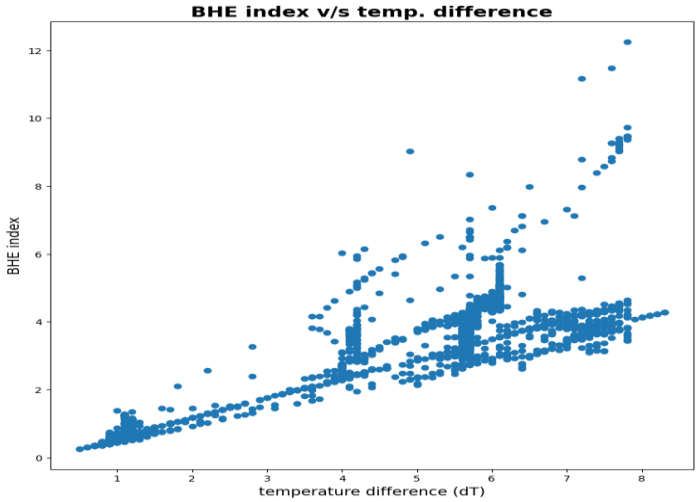
(Figure 1: Tout v/s time)



(Figure 3: Power v/s time)



(Figure 4: temp. diff v/s time)



(Figure 5: BHE index v/s temp. difference)

METHODOLOGY AND RESULTS

CODE FOR MODEL FORMATION FOR GROUND COUPLED SPLIT A/C SYSTEM

Importing libraries we require for our model formation

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow.keras.wrappers.scikit_learn
from sklearn.model_selection import RandomizedSearchCV
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor
from keras.callbacks import EarlyStopping
```

Forming our dataframe

```
data = pd.read_excel('new data.xlsx')
data.head()
```

Extracting our data

```
inp = data[['dT', 'P']]
opt = data['BHE index']
inp.head()

opt.head()
```

Spliting our data in training and testing data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( inp, opt, test_size=
0.2, random_state=42)
```

Creating our basic model

```
c = True
epochs = 130
while(c):

    model = tf.keras.Sequential([
      tf.keras.layers.Dense(32, input_shape = [2] , activation = 'relu'),
      tf.keras.layers.Dense(8, activation = 'relu'),
      tf.keras.layers.Dense(1, activation = 'relu')
    ])


    model.compile(loss = tf.keras.losses.mse,
                optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.001),
                 metrics = ['accuracy'])

    early_stopping_callback = EarlyStopping(monitor='val_loss',
patience=10  , restore_best_weights = True)
```

```
    model.fit(X_train, y_train, epochs = epochs, validation_split=0.2
,callbacks=[early_stopping_callback])

    c = model.stop_training
    epochs = 50
```

Setting our temperature difference columns in dT array

```
dT=data['dT']
dT_train, dT_test = train_test_split(dT, test_size = 0.2, random_state=42)
```

Loss of our model after optimization

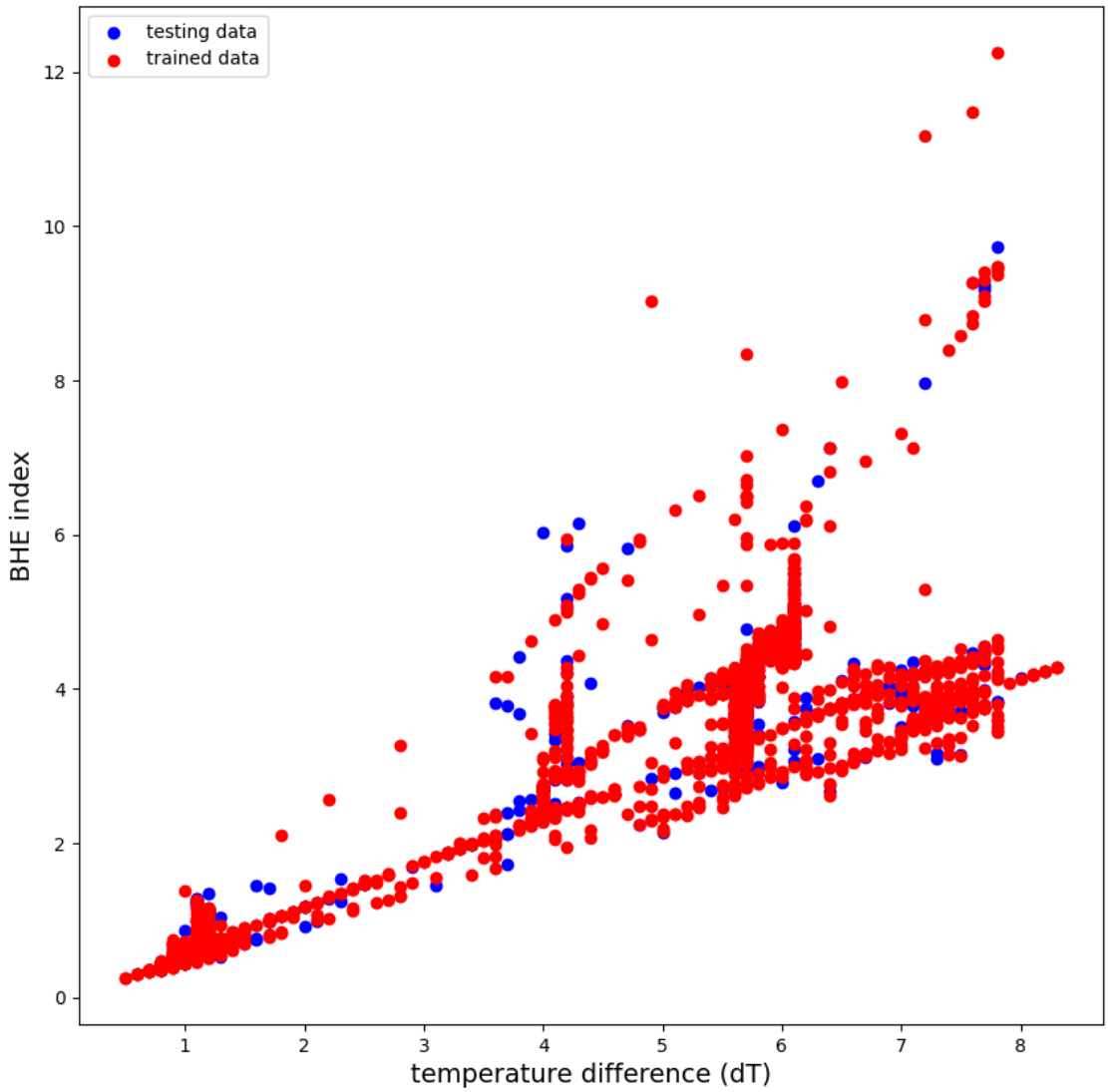
```
from sklearn.metrics import mean_squared_error
p = model.predict(X_test)
mean_squared_error(y_test,p)
```

```
20/20 [==============================] - 0s 1ms/step
```

```
0.004675965631424692   ( This is loss[mean squared error] )
```

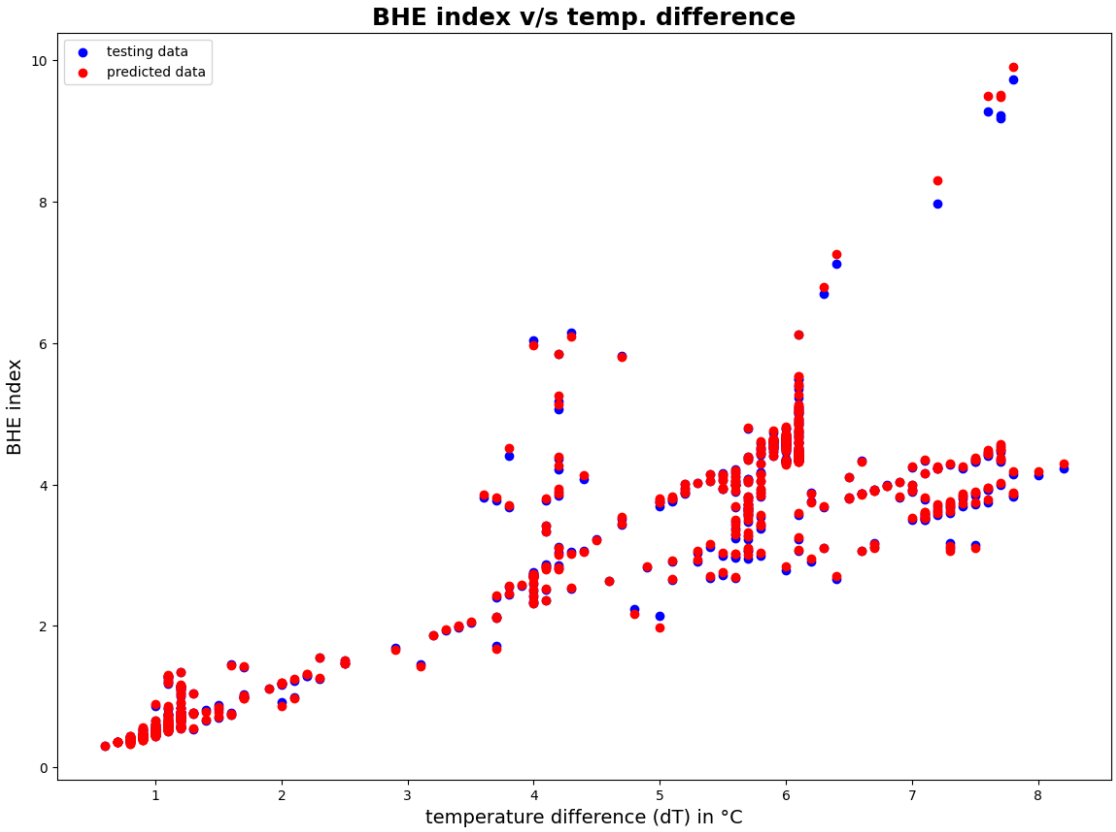
train test split graph plot

```
pred = plt.figure(figsize=(10,10))
plt.scatter(dT_test, y_test, c = 'b', label = 'testing data') # plot test
data in blue
plt.scatter(dT_train, y_train, c = 'r', label = 'trained data') # plot
train data in red
plt.xlabel('temperature difference (dT)', fontsize = 14)
plt.ylabel('BHE index', fontsize = 14)
plt.legend() # show a legend
```



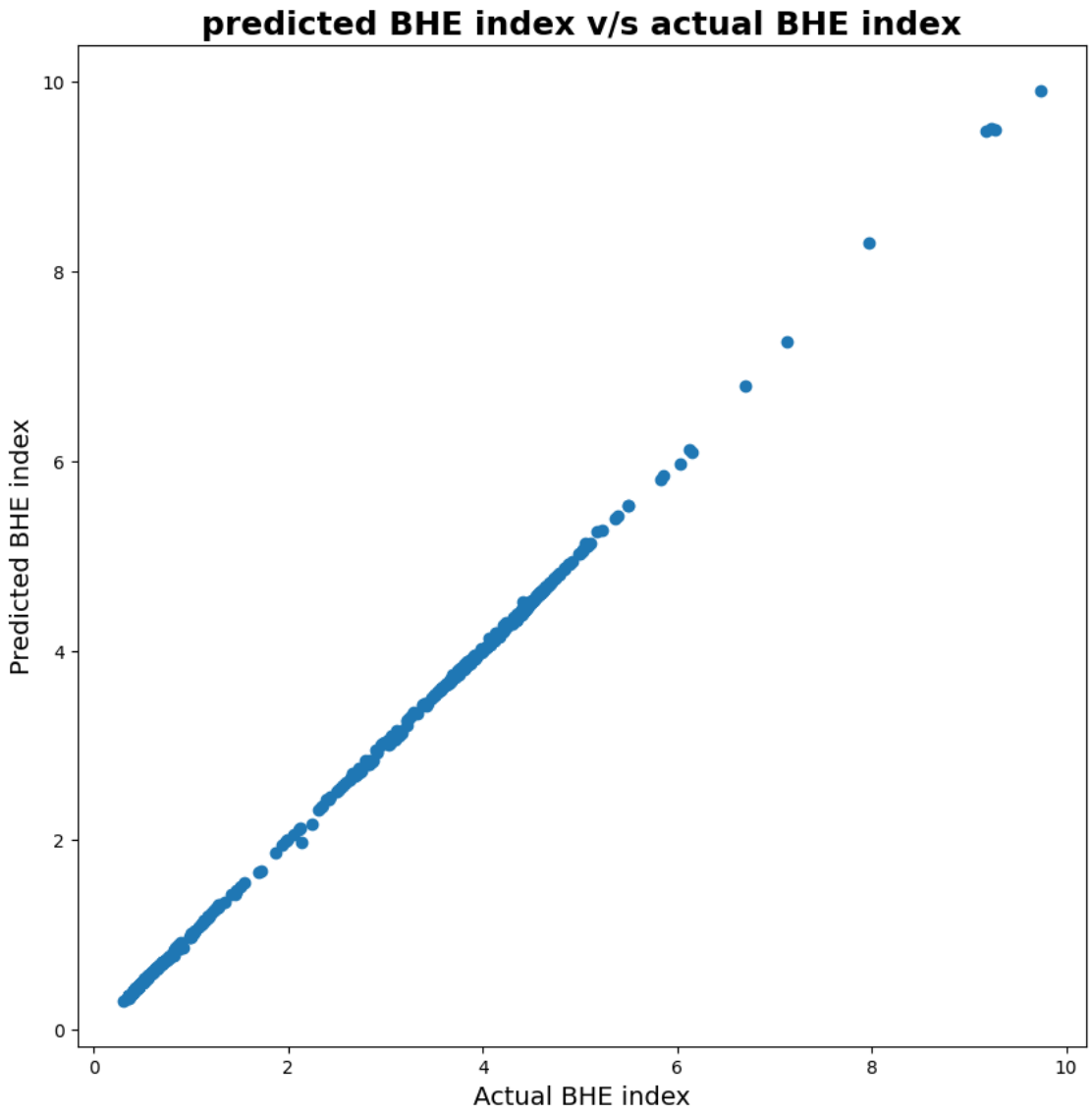
Prediction plot

```
pred = plt.figure(figsize=(14,10))
plt.scatter(dT_test, y_test, c = 'b', label = 'testing data') # plot test
data in blue
plt.scatter(dT_test, tf.squeeze(p), c = 'r', label = 'predicted data') #
plot test data in red
plt.title('BHE index v/s temp. difference', fontweight = 'bold', fontsize
= 18)
plt.xlabel('temperature difference (dT) in °C', fontsize = 14)
plt.ylabel('BHE index', fontsize = 14)
plt.legend() # show a legend
pred.savefig('dT vs BHE.jpg', dpi = 400)
```



prediction v/s actual graph

```
final = plt.figure(figsize=(10, 10))
plt.scatter(y_test, tf.squeeze(p))
plt.title('predicted BHE index v/s actual BHE index', fontweight = 'bold',
fontsize = 18)
plt.xlabel('Actual BHE index', fontsize = 14)
plt.ylabel('Predicted BHE index', fontsize = 14)
final.savefig('act vs pred.jpg', dpi = 400)
```



Optimization by tuning hyperparameters using RandomSearchCV

```
from keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import SGD,Adam,RMSprop,Adagrad

def test_function (n1 = 16, n2 = 8, epochs = 30, optimizersss = 'SGD', lr
=0.01) :

  optimized_val = {'SGD' : SGD , 'Adam' : Adam , 'RMSprop' : RMSprop ,
'Adagrad' : Adagrad}

  model = tf.keras.Sequential([
  tf.keras.layers.Dense(n1, input_shape = [2] , activation = 'relu'),
  tf.keras.layers.Dense(n2, activation = 'relu'),
  tf.keras.layers.Dense(1, activation = 'relu')
  ])

  model.compile(loss = tf.keras.losses.mse,
            optimizer= optimized_val[optimizersss](learning_rate = lr),
```

```
            metrics = ['accuracy'])

  early_stopping_callback = EarlyStopping(monitor='val_loss', patience= 5
, restore_best_weights = True)

  model.fit(X_train, y_train, epochs= epochs , validation_split=0.2,
callbacks=[early_stopping_callback])


  return model


X_train = np.array(X_train)
y_train = np.array(y_train)


epochs = np.arange(30, 210, step=10)
n1 = [1,2,4,8,16,32]
n2 = [1,2,4,8,16,32]
n3 = [1,2,4,8,16,32]
n4 = [1,2,4,8,16,32]
n5 = [1,2,4,8,16,32]
n6 = [1,2,4,8,16,32]
lr = [0.1, 0.01,0.001]
optimizersss = ['SGD','Adam','RMSprop','Adagrad']

model1 = KerasRegressor(build_fn = test_function,  verbose = 1)

param_grid = dict(n1 = n1, n2 = n2, epochs = epochs ,lr = lr ,
optimizersss = optimizersss  )

grid = RandomizedSearchCV(estimator = model1 , param_distributions =
param_grid, n_jobs = -1 , cv = 10)

grid_result = grid.fit(X_train , y_train)

print("Best : " ,grid_result.best_score_ , grid_result.best_params_)
```