In [1]:

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import xgboost as xgb
from xgboost.sklearn import XGBRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.multioutput import MultiOutputRegressor
import lightgbm as lgb #conda install lightgbm
from sklearn.metrics import (roc_curve, auc, accuracy_score)
from sklearn.metrics import r2_score,mean_squared_error
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
import pandas as p
import sklearn
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from keras.optimizers import Adadelta,Adagrad,RMSprop,Adam,Adamax,Nadam
from sklearn.model_selection import GridSearchCV
data = pd.read_csv("D:\\javeed\\DS documents\\abid dataset\\electric motor temp\\pmsm_temperature_data.csv")
```

```
Using TensorFlow backend.
```

In [16]:

```
X= data.drop(['stator_yoke','pm'], axis = 1)
Y = data[['stator_yoke','pm']]
train_x,test_x,train_y,test_y  = train_test_split(X,Y, test_size = 0.33, stratify=data.profile_id)
train_x =train_x.drop(['profile_id'], axis = 1)
test_x =test_x.drop(['profile_id'], axis = 1)
```

In [9]:

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(train_x)
testing_set_scaled = sc.fit_transform(test_x)
```

Neural Network

In [35]:

```
from keras.layers import ELU
from keras.layers import Dropout
from keras.optimizers import Adadelta,Adagrad,RMSprop,Adam,Adamax,Nadam
model3 = Sequential()
model3.add(Dense(500,input_shape=(10,)))
model3.add(ELU())
model3.add(Dense(400))
model3.add(ELU())
model3.add(Dense(300))
model3.add(ELU())
model3.add(Dense(200))
model3.add(ELU())
model3.add(Dense(100))
model3.add(ELU())
model3.add(Dense(2))
import keras
adam1 =Nadam(learning_rate=0.001, beta_1=0.9, beta_2=0.99)
model3.compile(loss='mean_squared_error', optimizer= adam1, metrics= ['accurac
y'])
model3.fit(train_x, train_y, epochs=5, batch_size=115, validation_data=(test_x
, test_y))
```

```
WARNING:tensorflow:From D:\javeed\DS documents\anaconda install\lib\site-pack
ages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is
deprecated. Please use tf.compat.v1.global_variables instead.

Train on 668706 samples, validate on 329364 samples
Epoch 1/5
668706/668706 [==============================] - 136s 204us/step - loss: 0.07
63 - accuracy: 0.8318 - val_loss: 0.0523 - val_accuracy: 0.8624
Epoch 2/5
668706/668706 [==============================] - 138s 207us/step - loss: 0.03
72 - accuracy: 0.8868 - val_loss: 0.0332 - val_accuracy: 0.8957
Epoch 3/5
668706/668706 [==============================] - 128s 191us/step - loss: 0.02
51 - accuracy: 0.9073 - val_loss: 0.0221 - val_accuracy: 0.9158
Epoch 4/5
668706/668706 [==============================] - 127s 189us/step - loss: 0.01
94 - accuracy: 0.9190 - val_loss: 0.0177 - val_accuracy: 0.9197
Epoch 5/5
668706/668706 [==============================] - 146s 218us/step - loss: 0.01
59 - accuracy: 0.9269 - val_loss: 0.0140 - val_accuracy: 0.9317
```

Out[35]:

```
<keras.callbacks.callbacks.History at 0x1e83daf15c0>
```

In [214]:

```
nn100_train_pred = model3.predict(train_x)
nn100_test_pred = model3.predict(test_x)
```

In [215]:

```
nn100_train_RMSE=np.mean((nn100_train_pred - train_y)**2, axis=0)
nn100_train_RMSE = np.sqrt(nn100_train_RMSE)
#stator_yoke    0.053246
#pm             0.406511
nn100_test_RMSE=np.mean((nn100_test_pred - test_y)**2, axis=0)
nn100_test_RMSE= np.sqrt(nn100_test_RMSE)
from sklearn.metrics import r2_score
nn100_train_R2 = r2_score(train_y, nn100_train_pred)#0.9553449358644539
nn100_test_R2 = r2_score(test_y, nn100_test_pred)#0.9552267680020445
```

In [216]:

```
nn100_train = nn100_train_RMSE
nn100_train_R2 = pd.Series(nn100_train_R2)
nn100_train = nn100_train.append(nn100_train_R2)
nn100_train = pd.DataFrame(nn100_train)
nn_train=nn100_train.rename(columns={0: 'nn100_train'})

nn100_test = nn100_test_RMSE
nn100_test_R2 = pd.Series(nn100_test_R2)
nn100_test = nn100_test.append(nn100_test_R2)
nn100_test = pd.DataFrame(nn100_test)
nn_test=nn100_test.rename(columns={0: 'nn100_test'})
```

XGBRegressor

In [55]:

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import xgboost as xgb
from xgboost.sklearn import XGBRegressor
import datetime
from sklearn.model_selection import GridSearchCV
from sklearn.multioutput import MultiOutputRegressor
xgb_multioutputregressor = MultiOutputRegressor(xgb.XGBRegressor()).fit(train_
x, train_y)
xgb_train_RMSE=np.mean((xgb_multioutputregressor.predict(train_x) - train_y)**
2, axis=0)
xgb_test_RMSE=np.mean((xgb_multioutputregressor.predict(test_x) - test_y)**2,
axis=0)
xgb_train_RMSE = np.sqrt(xgb_train_RMSE)
#stator_yoke    0.062031
#pm             0.401400
xgb_test_RMSE = np.sqrt(xgb_test_RMSE)
#stator_yoke    0.062406
#pm             0.400764
from sklearn.metrics import r2_score
xgb_train_pred = xgb_multioutputregressor.predict(train_x)
xgb_test_pred = xgb_multioutputregressor.predict(test_x)
xgb_train_R2 = r2_score(train_y, xgb_train_pred)#0.9169291776206683
xgb_test_R2 = r2_score(test_y, xgb_test_pred)#0.916830099751504

xgb_train_score = xgb_multioutputregressor.score(train_x,train_y)
xgb_test_score = xgb_multioutputregressor.score(test_x,test_y)
xgb_train_RMSE=np.mean((xgb_multioutputregressor.predict(train_x) - train_y)**
2, axis=0)
xgb_test_RMSE=np.mean((xgb_multioutputregressor.predict(test_x) - test_y)**2,
axis=0)
xgb_train_RMSE = np.sqrt(xgb_train_RMSE)
```

```
[15:11:32] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objec
tive/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squa
rederror.
[15:13:55] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objec
tive/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squa
rederror.
```

In [120]:

```
xgb_train = xgb_train_RMSE
xgb_train_R2 = pd.Series(xgb_train_R2)
xgb_train = xgb_train.append(xgb_train_R2)
xgb_train = pd.DataFrame(xgb_train)
xgb_train=xgb_train.rename(columns={0: 'xgb_train'})

xgb_test = xgb_test_RMSE
xgb_test_R2 = pd.Series(xgb_test_R2)
xgb_test =xgb_test.append(xgb_test_R2)
xgb_test = pd.DataFrame(xgb_test)
xgb_test=xgb_test.rename(columns={0: 'xgb_test'})
```

LGBM

In [17]:

```
from sklearn.multioutput import MultiOutputRegressor
import matplotlib.pyplot as plt
import lightgbm as lgb #conda install lightgbm
from sklearn.metrics import (roc_curve, auc, accuracy_score)
from sklearn.model_selection import GridSearchCV
lgb_multioutput = MultiOutputRegressor(lgb.LGBMRegressor(learning_rate=0.05,max_depth=7,n_jobs=1,n_estimators=1000,nthread=-1))
lgb_multioutput.fit(train_x, train_y)
lgb_train_RMSE=np.mean((lgb_multioutput.predict(train_x) - train_y)**2, axis=0)
lgb_train_RMSE = np.sqrt(lgb_train_RMSE)
#stator_yoke    0.026573
#pm             0.164039
lgb_test_RMSE=np.mean((lgb_multioutput.predict(test_x) - test_y)**2, axis=0)
lgb_test_RMSE= np.sqrt(lgb_test_RMSE)
#stator_yoke    0.027163
#pm             0.166146
from sklearn.metrics import r2_score
lgb_train_pred = lgb_multioutput.predict(train_x)
lgb_test_pred = lgb_multioutput.predict(test_x)
lgb_train_R2 = r2_score(train_y, lgb_train_pred)#0.9861326172294375
lgb_test_R2 = r2_score(test_y, lgb_test_pred)#0.985719127660748

lgb_train_score = lgb_multioutput.score(train_x,train_y)
lgb_test_score =lgb_multioutput.score(test_x,test_y)
```

In [61]:

```
lgb_train = lgb_train_RMSE
a = np.float64(lgb_train_R2)
a = pd.Series(a)
lgb_train = lgb_train.append(a)
lgb_train = pd.DataFrame(lgb_train)
lgb_train=lgb_train.rename(columns={0: 'lgb_train'})

lgb_test= lgb_test_RMSE
lgb_test_R2 = pd.Series(lgb_test_R2)
lgb_test = lgb_test.append(lgb_test_R2)
lgb_test= pd.DataFrame(lgb_test)
lgb_test=lgb_test.rename(columns={0: 'lgb_test'})
```

polynomial with 2 degree

In [211]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score,mean_squared_error
quad = PolynomialFeatures (degree = 2)
train_quad = quad.fit_transform(train_x)
test_quad=quad.fit_transform(test_x)
plr = MultiOutputRegressor(LinearRegression()).fit(train_quad,train_y)
Y_train_pred = plr.predict(train_quad)
Y_test_pred = plr.predict(test_quad)
print('Polynomial Linear Regression:' ,plr.score(test_quad,test_y))#0.9116422853434928
poly_R2 =plr.score(test_quad,test_y)
#RMSE
poly_Train_RMSE = np.sqrt(mean_squared_error(train_y,Y_train_pred))#0.29693444794659535
poly_Test_RMSE = np.sqrt(mean_squared_error(test_y,Y_test_pred))#0.29665945037595576
#R2
poly_train_R2 = r2_score(train_y, Y_train_pred)#0.9111167243895444
poly_test_R2 = r2_score(test_y, Y_test_pred)#0.911157317010483
```

```
Polynomial Linear Regression: 0.9115021337511484
```

In [212]:
```
poly_test = {'poly_test':[poly_Test_RMSE,poly_test_R2 ]}
poly_test =pd.DataFrame(poly_test)
poly_test=poly_test.rename(index={0:"stator_yoke", 1: 0})
poly_test

poly_train = {'poly_train':[poly_Train_RMSE,poly_train_R2 ]}
poly_train =pd.DataFrame(poly_train)
poly_train=poly_train.rename(index={0:"stator_yoke", 1 : 0})
poly_train
```

Out[212]:

	poly_train
stator_yoke	0.297187
0	0.910940

AdaBoostRegressor

In [19]:

```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.multioutput import MultiOutputRegressor
#ada_multioutput = MultiOutputRegressor(AdaBoostRegressor(learning_rate=0.01,
 n_estimators=200))
ada_multioutput = MultiOutputRegressor(AdaBoostRegressor())
ada_multioutput.fit(train_x, train_y)
ada_train_RMSE=np.mean((ada_multioutput.predict(train_x) - train_y)**2, axis=0
)
ada_train_RMSE = np.sqrt(ada_train_RMSE)
ada_test_RMSE=np.mean((ada_multioutput.predict(test_x) - test_y)**2, axis=0)
ada_test_RMSE= np.sqrt(ada_test_RMSE)
from sklearn.metrics import r2_score
ada_train_pred = ada_multioutput.predict(train_x)
ada_test_pred = ada_multioutput.predict(test_x)
ada_train_R2 = r2_score(train_y, ada_train_pred)#0.8376333232840278
ada_test_R2 = r2_score(test_y, ada_test_pred)#0.8376362009220268

ada_train_score = ada_multioutput.score(train_x,train_y)#0.8321852774400107
ada_test_score =ada_multioutput.score(test_x,test_y)#0.8319736250251978
```

In [46]:

```
ada_train = ada_train_RMSE
ada_train_R2 = pd.Series(ada_train_R2)
ada_train = ada_train.append(ada_train_R2)
ada_train = pd.DataFrame(ada_train)
ada_train=ada_train.rename(columns={0: 'ada_train'})

ada_test = ada_test_RMSE
ada_test_R2 = pd.Series(ada_test_R2)
ada_test =ada_test.append(ada_test_R2)
ada_test = pd.DataFrame(ada_test)
ada_test=ada_test.rename(columns={0: 'ada_test'})
```

In [24]:

```
ada_train_score
```

Out[24]:

```
0.8321852774400107
```

RandomForestRegressor

In [20]:

```
from sklearn import ensemble
from sklearn.multioutput import MultiOutputRegressor
import matplotlib.pyplot as plt
import lightgbm as lgb #conda install lightgbm
from sklearn.metrics import (roc_curve, auc, accuracy_score)
from sklearn.model_selection import GridSearchCV
#rf_multioutput = MultiOutputRegressor(ensemble.RandomForestRegressor(n_estima
tors=500, n_jobs=1, verbose=1))
rf_multioutput = MultiOutputRegressor(ensemble.RandomForestRegressor())
#lgb_multioutput = MultiOutputRegressor(lgb.LGBMRegressor(learning_rate=0.05,m
ax_depth=7,n_jobs=1,n_estimators=1000,nthread=-1))
rf_multioutput.fit(train_x, train_y)
rf_train_RMSE=np.mean((rf_multioutput.predict(train_x) - train_y)**2, axis=0)
rf_train_RMSE = np.sqrt(rf_train_RMSE)
rf_test_RMSE=np.mean((rf_multioutput.predict(test_x) - test_y)**2, axis=0)
rf_test_RMSE= np.sqrt(rf_test_RMSE)
from sklearn.metrics import r2_score
rf_train_pred = rf_multioutput.predict(train_x)
rf_test_pred = rf_multioutput.predict(test_x)
rf_train_R2 = r2_score(train_y, rf_train_pred)#0.9997814045437312
rf_test_R2 = r2_score(test_y, rf_test_pred)#0.9990043465338014

rf_train_score = rf_multioutput.score(train_x,train_y)#0.9997838183035852
rf_test_score =rf_multioutput.score(test_x,test_y)#0.9989816978585642
```

```
D:\javeed\DS documents\anaconda install\lib\site-packages\sklearn\ensemble\fo
rest.py:246: FutureWarning: The default value of n_estimators will change fro
m 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\javeed\DS documents\anaconda install\lib\site-packages\sklearn\ensemble\fo
rest.py:246: FutureWarning: The default value of n_estimators will change fro
m 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [47]:

```
rf_train = rf_train_RMSE
rf_train_R2 = pd.Series(rf_train_R2)
rf_train = rf_train.append(rf_train_R2)
rf_train = pd.DataFrame(rf_train)
rf_train=rf_train.rename(columns={0: 'rf_train'})

rf_test = rf_test_RMSE
rf_test_R2 = pd.Series(rf_test_R2)
rf_test =rf_test.append(rf_test_R2)
rf_test = pd.DataFrame(rf_test)
rf_test=rf_test.rename(columns={0: 'rf_test'})
```

with profile_id

In [53]:

```
X= data.drop(['stator_yoke','pm'], axis = 1)
Y = data[['stator_yoke','pm']]
x_train,x_test,y_train,y_test  = train_test_split(X,Y, test_size = 0.33, strat
ify=data.profile_id)
import numpy as np
import pandas as pd
from sklearn import preprocessing
import xgboost as xgb
from xgboost.sklearn import XGBRegressor
import datetime
from sklearn.model_selection import GridSearchCV
from sklearn.multioutput import MultiOutputRegressor
p_xgb_multioutputregressor = MultiOutputRegressor(xgb.XGBRegressor()).fit(x_tr
ain, y_train)
p_xgb_train_RMSE=np.mean((p_xgb_multioutputregressor.predict(x_train) - y_trai
n)**2, axis=0)
p_xgb_test_RMSE=np.mean((p_xgb_multioutputregressor.predict(x_test) - y_test)*
*2, axis=0)
p_xgb_train_RMSE = np.sqrt(p_xgb_train_RMSE)
#stator_yoke    0.062031
#pm             0.401400
p_xgb_test_RMSE = np.sqrt(p_xgb_test_RMSE)
#stator_yoke    0.062406
#pm             0.400764
from sklearn.metrics import r2_score
p_xgb_train_pred = p_xgb_multioutputregressor.predict(x_train)
p_xgb_test_pred = p_xgb_multioutputregressor.predict(x_test)
p_xgb_train_R2 = r2_score(y_train, p_xgb_train_pred)#0.9169291776206683
p_xgb_test_R2 = r2_score(y_test, p_xgb_test_pred)#0.916830099751504
```

```
[15:05:23] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objec
tive/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squa
rederror.
[15:08:05] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objec
tive/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squa
rederror.
```

In [54]:

```
p_xgb_train = p_xgb_train_RMSE
p_xgb_train_R2 = pd.Series(p_xgb_train_R2)
p_xgb_train = p_xgb_train.append(p_xgb_train_R2)
p_xgb_train = pd.DataFrame(p_xgb_train)
p_xgb_train=p_xgb_train.rename(columns={0: 'p_xgb_train'})

p_xgb_test = p_xgb_test_RMSE
p_xgb_test_R2 = pd.Series(p_xgb_test_R2)
p_xgb_test =p_xgb_test.append(p_xgb_test_R2)
p_xgb_test = pd.DataFrame(p_xgb_test)
p_xgb_test=p_xgb_test.rename(columns={0: 'p_xgb_test'})
```

LGBMRegressor

In [28]:

```
from sklearn.multioutput import MultiOutputRegressor
import matplotlib.pyplot as plt
import lightgbm as p_lgb #conda install lightgbm
from sklearn.metrics import (roc_curve, auc, accuracy_score)
from sklearn.model_selection import GridSearchCV
p_lgb_multioutput = MultiOutputRegressor(p_lgb.LGBMRegressor(learning_rate=0.0
5,max_depth=7,n_jobs=1,n_estimators=1000,nthread=-1))
p_lgb_multioutput.fit(x_train, y_train)
p_lgb_train_RMSE=np.mean((p_lgb_multioutput.predict(x_train) - y_train)**2, ax
is=0)
p_lgb_train_RMSE = np.sqrt(p_lgb_train_RMSE)
#stator_yoke    0.025517
#pm             0.115780
p_lgb_test_RMSE=np.mean((p_lgb_multioutput.predict(x_test) - y_test)**2, axis=
0)
p_lgb_test_RMSE= np.sqrt(p_lgb_test_RMSE)
#stator_yoke    0.025978
#pm             0.117266
from sklearn.metrics import r2_score
p_lgb_train_pred = p_lgb_multioutput.predict(x_train)
p_lgb_test_pred = p_lgb_multioutput.predict(x_test)
p_lgb_train_R2 = r2_score(y_train, p_lgb_train_pred)#00.9929160367267138
p_lgb_test_R2 = r2_score(y_test, p_lgb_test_pred)#0.9927245780698586
p_lgb_train_score= p_lgb_multioutput.score(x_train,y_train)#0.9929160367267138
p_lgb_test_score =p_lgb_multioutput.score(x_test,y_test)#0.9927245780698586
```

In [64]:

```
p_lgb_train = p_lgb_train_RMSE
a = np.float64(p_lgb_train_R2)
a = pd.Series(a)
p_lgb_train = p_lgb_train.append(a)
p_lgb_train = pd.DataFrame(p_lgb_train)
p_lgb_train=p_lgb_train.rename(columns={0: 'p_lgb_train'})

p_lgb_test= p_lgb_test_RMSE
p_lgb_test_R2 = pd.Series(p_lgb_test_R2)
p_lgb_test = p_lgb_test.append(p_lgb_test_R2)
p_lgb_test = pd.DataFrame(p_lgb_test)
p_lgb_test=p_lgb_test.rename(columns={0: 'p_lgb_test'})
```

polynomialFeatures

In [ ]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import polynomialFeatures
from sklearn.metrics import r2_score,mean_squared_error
quad = polynomialFeatures (degree = 2)
train_quad = quad.fit_transform(x_train)
test_quad=quad.fit_transform(x_test)
plr = MultiOutputRegressor(LinearRegression().fit(train_quad,y_train))
Y_train_pred = plr.predict(train_quad)
Y_test_pred = plr.predict(test_quad)
print('P_polynomial Linear Regression:' ,plr.score(test_quad,y_test))#0.9116422853434928
p_poly_R2 =plr.score(test_quad,y_test)
#RMSE
p_poly_Train_RMSE = np.sqrt(mean_squared_error(y_train,Y_train_pred))#0.29693444794659535
p_poly_Test_RMSE = np.sqrt(mean_squared_error(y_test,Y_test_pred))#0.29665945037595576
#R2
p_poly_train_R2 = r2_score(y_train, Y_train_pred)#0.9111167243895444
p_poly_test_R2 = r2_score(y_test, Y_test_pred)#0.911157317010483
```

In [187]:

```
p_poly_test = {'p_poly_test':[p_poly_Test_RMSE,p_poly_test_R2 ]}
p_poly_test =pd.DataFrame(p_poly_test)
p_poly_test=p_poly_test.rename(index={0:"stator_yoke", 1: 0})

p_poly_train = {'p_poly_train':[p_poly_Train_RMSE,p_poly_train_R2 ]}
p_poly_train =pd.DataFrame(p_poly_train)
p_poly_train=p_poly_train.rename(index={0:"stator_yoke", 1 : 0})
```

Out[187]:

	p_poly_train
stator_yoke	0.292664
0	0.913636

AdaBoostRegressor

In [31]:

```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.multioutput import MultiOutputRegressor
#p_ada_multioutput = MultiOutputRegressor(P_adaBoostRegressor(learning_rate=0.01, n_estimators=200))
p_ada_multioutput = MultiOutputRegressor(AdaBoostRegressor())
p_ada_multioutput.fit(x_train, y_train)
p_ada_train_RMSE=np.mean((p_ada_multioutput.predict(x_train) - y_train)**2, axis=0)
p_ada_train_RMSE = np.sqrt(p_ada_train_RMSE)
#stator_yoke    0.150531
#pm             0.548006
p_ada_test_RMSE=np.mean((p_ada_multioutput.predict(x_test) - y_test)**2, axis=0)
p_ada_test_RMSE= np.sqrt(p_ada_test_RMSE)
#stator_yoke    0.150474
#pm             0.547884
from sklearn.metrics import r2_score
p_ada_train_pred = p_ada_multioutput.predict(x_train)
p_ada_test_pred = p_ada_multioutput.predict(x_test)
p_ada_train_R2 = r2_score(y_train, p_ada_train_pred)#0.8372726936873622
p_ada_test_R2 = r2_score(y_test, p_ada_test_pred)#0.8372322396789236

p_ada_train_score= p_ada_multioutput.score(x_train,y_train)#0.9929160367267138
p_ada_test_score =p_ada_multioutput.score(x_test,y_test)
```

RandomForestRegressor

In [32]:

```
from sklearn import ensemble
from sklearn.multioutput import MultiOutputRegressor
import matplotlib.pyplot as plt
import lightgbm as lgb #conda install lightgbm
from sklearn.metrics import (roc_curve, auc, accuracy_score)
from sklearn.model_selection import GridSearchCV
#p_rf_multioutput = MultiOutputRegressor(ensemble.RandomForestRegressor(n_esti
mators=500, n_jobs=1, verbose=1))
p_rf_multioutput = MultiOutputRegressor(ensemble.RandomForestRegressor())
#lgb_multioutput = MultiOutputRegressor(lgb.LGBMRegressor(learning_rate=0.05,m
ax_depth=7,n_jobs=1,n_estimators=1000,nthread=-1))
p_rf_multioutput.fit(x_train, y_train)
p_rf_train_RMSE=np.mean((p_rf_multioutput.predict(x_train) - y_train)**2, axis
=0)
p_rf_train_RMSE = np.sqrt(p_rf_train_RMSE)
p_rf_test_RMSE=np.mean((p_rf_multioutput.predict(x_test) - y_test)**2, axis=0)
p_rf_test_RMSE= np.sqrt(p_rf_test_RMSE)
from sklearn.metrics import r2_score
p_rf_train_pred = p_rf_multioutput.predict(x_train)
p_rf_test_pred = p_rf_multioutput.predict(x_test)
p_rf_train_R2 = r2_score(y_train, p_rf_train_pred)#0.99978140454373l2
p_rf_test_R2 = r2_score(y_test, p_rf_test_pred)#0.9990043465338014

p_rf_train_score= p_rf_multioutput.score(x_train,y_train)#0.9929160367267138
p_rf_test_score =p_rf_multioutput.score(x_test,y_test)
```

```
D:\javeed\DS documents\anaconda install\lib\site-packages\sklearn\ensemble\fo
rest.py:246: FutureWarning: The default value of n_estimators will change fro
m 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
D:\javeed\DS documents\anaconda install\lib\site-packages\sklearn\ensemble\fo
rest.py:246: FutureWarning: The default value of n_estimators will change fro
m 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [50]:

```
p_rf_train = p_rf_train_RMSE
p_rf_train_R2 = pd.Series(p_rf_train_R2)
p_rf_train = p_rf_train.append(p_rf_train_R2)
p_rf_train = pd.DataFrame(p_rf_train)
p_rf_train=p_rf_train.rename(columns={0: 'p_rf_train'})

p_rf_test = p_rf_test_RMSE
p_rf_test_R2 = pd.Series(p_rf_test_R2)
p_rf_test =p_rf_test.append(p_rf_test_R2)
p_rf_test = pd.DataFrame(p_rf_test)
p_rf_test=p_rf_test.rename(columns={0: 'p_rf_test'})
```

In [74]:

```
q=data.loc[(data['profile_id']==20) & (data['profile_id'] ==6)]
```

In [219]:

```
RMSE_R2_values = pd.concat([poly_test,poly_train,xgb_train, xgb_test,lgb_train
,lgb_test,ada_train,ada_test,rf_train,rf_test,p_poly_train,p_poly_test,p_xgb_t
rain, p_xgb_test,p_lgb_train,p_lgb_test,p_ada_train,p_ada_test,p_rf_train,p_rf
_test,nn_train, nn_test], axis=1, sort=False)
RMSE_R2_values = RMSE_R2_values.rename(index={'stator_yoke': 'stator_yoke RMS
E','pm':'pm RMSE',0:"R2"})
RMSE_R2_values
```

Out[219]:

	poly_test	poly_train	xgb_train	xgb_test	lgb_train	lgb_test	ada_train	ada_test
stator_yoke RMSE	0.296172	0.297187	0.060909	0.003707	0.026721	0.027337	0.147409	0.147518
R2	0.911502	0.910940	0.917367	0.917386	0.985995	0.985516	0.832185	0.831974
pm RMSE	NaN	NaN	0.400252	0.160081	0.164518	0.167241	0.557947	0.558125

3 rows × 22 columns

In [ ]:

In [ ]: