Sensor Calibration

September 19, 2021

0.1 CO Calibration

[798]:
```
import pandas as pd
import scipy.io
import numpy as np
from sktime.performance_metrics.forecasting import sMAPE, smape_loss
mat = scipy.io.loadmat('COfinal.mat')
#cardio_df = pd.DataFrame(mat)

len(mat['COfinal'])
L_COfinal=[]
for i in range(len(mat['COfinal'])):
    L_COfinal.append(list(mat['COfinal'][i]))

CO_data=pd.DataFrame(L_COfinal,columns=['Lab1',␣
 ↪'Lab2','Lab3','Lab4','Temp','RH','Ref','Time'])
Time=CO_data['Time'].to_list()
Time=np.array(Time)
Date=pd.to_datetime(Time-719529,unit='d').round('h')
CO_data['Date'] = Date.tolist()
CO_data=CO_data.set_index('Date')
CO_data.drop('Time',axis = 1, inplace = True)

mat = scipy.io.loadmat('CO_raw.mat')
L_CO_raw=[]
for i in range(len(mat['CO_raw'])):
    L_CO_raw.append(list(mat['CO_raw'][i]))

CO_raw=pd.DataFrame(L_CO_raw,columns=['WE', 'AE','Temp','RH','Time'])


CO_data.insert(loc = 0,
          column = 'WE',
          value = CO_raw['WE'].to_list())

CO_data.insert(loc = 1,
          column = 'AE',
```

```
          value = CO_raw['AE'].to_list())
CO_data=CO_data.interpolate()

WE=np.array(CO_data['WE'].to_list())
AE=np.array(CO_data['AE'].to_list())
Signal=list(WE-AE)
CO_data.insert(loc = 2,
          column = 'Signal',
          value = Signal)

CO_data.head()
```

[798]:

Date	WE	AE	Signal	Lab1	Lab2	Lab3	Lab4	Temp	RH	Ref
2019-10-02 12:00:00	1023.970000	39.543125	984.426875	3571.592599	2272.635909	1941.086340	618.283953	26.378438	58.063437	206.858886
2019-10-02 13:00:00	1023.970000	40.757143	983.212857	3534.556213	2261.997806	1938.583211	609.013503	25.502791	59.868837	261.703907
2019-10-02 16:00:00	1023.970000	40.073881	983.896119	3714.254704	2291.131285	1939.991999	679.326053	30.827910	49.008060	268.918360
2019-10-02 17:00:00	941.693433	209.369701	732.323731	2296.396571	1639.208945	1421.286044	497.011998	30.047164	51.259851	319.195427
2019-10-03 16:00:00	488.950000	320.808571	168.141429	555.780140	390.151856	258.023564	-445.692120	29.441429	52.018571	280.104590

[799]:
```
CO_data.shape
```

[799]:
```
(6021, 10)
```

[800]:
```
CO_data.describe()
```

[800]:

	WE	AE	Signal	Lab1	Lab2
count	6021.000000	6021.000000	6021.000000	6021.000000	6021.000000
mean	489.868443	325.037641	164.830802	460.445775	351.621354
std	147.893193	22.357098	145.524186	316.990546	298.778134
min	269.495048	38.934737	-59.665104	-76.574116	-103.290653

```
25%     399.638542   316.795833    72.162250   246.967288   162.828332
50%     438.460648   328.552083   114.690455   375.497026   248.319472
75%     515.414062   339.209804   192.807355   573.671805   413.089960
max    1023.970000   377.300333   985.035263  3967.688054  2345.947440

              Lab3         Lab4         Temp           RH          Ref
count  6021.000000  6021.000000  6021.000000  6021.000000  6021.000000
mean    229.061278  -539.062229    18.258360    67.305000   663.749977
std     298.561008   295.565104     7.601853    18.454202   677.801714
min    -211.680627  -912.704297     1.035000    11.428168 -1095.570697
25%      41.157070  -728.592009    12.408678    53.576341   285.212286
50%     124.692402  -638.382797    16.872568    72.236338   466.332961
75%     284.461578  -467.551093    22.849585    82.938548   759.000870
max    1941.711657   854.177980    44.748056    93.012486  6836.814023
```

0.2 Model 1: Linear Regression

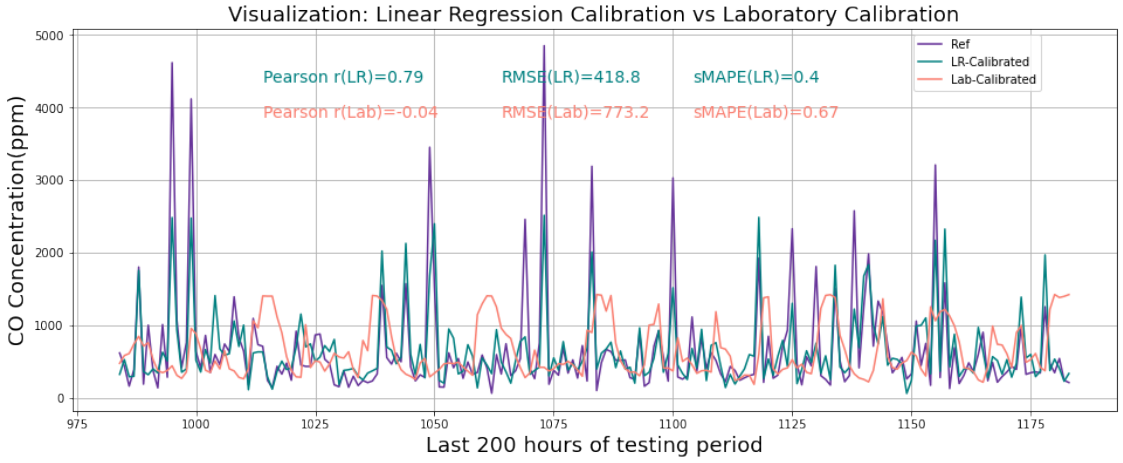
[801]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error as mae
import sklearn.metrics as sm
import matplotlib.pyplot as plt
X=CO_Data[['Signal','Temp','RH']]
y=CO_Data['Ref']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
len(X_test)
```

[801]:
```
1184
```

[802]:
```
lr = LinearRegression()
model = lr.fit(X_train, y_train)
pred = model.predict(X_test)
lab1=CO_Data['Lab1'].to_list()[len(y_train):]

index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
```

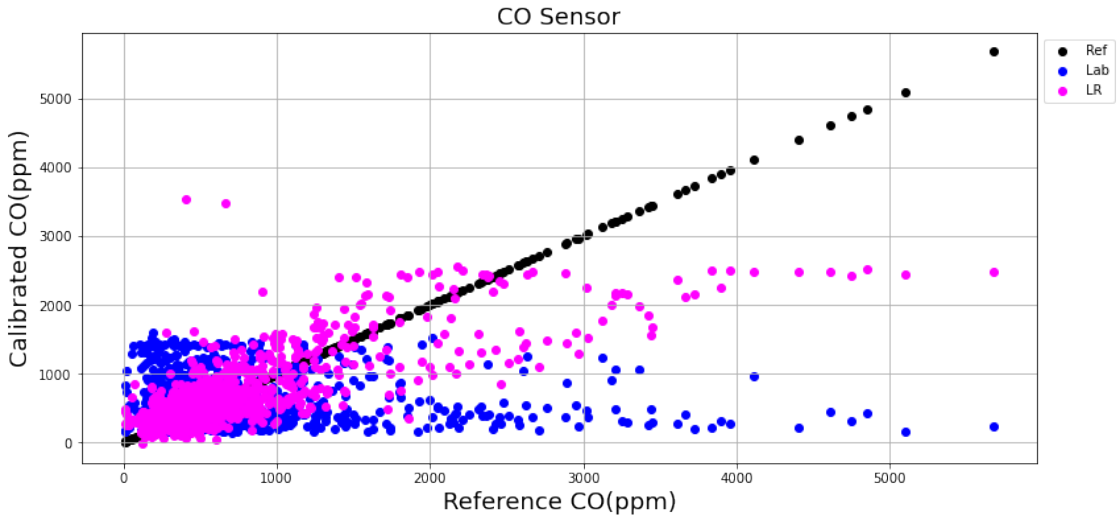
```
sMAPE_lr_CO=sMAPE_lr
RMSE_lr_CO=RMSE_lr/np.mean(np.array(y_test))
Pearson_lr_CO=Pearson_lr
sMAPE_lab_CO=sMAPE_lab
RMSE_lab_CO=RMSE_lab/np.mean(np.array(lab1))
Pearson_lab_CO=Pearson_lab
```

[803]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='teal')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'LR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
↪= (0.8,1))
plt.ylabel('CO Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(LR)='+str(sMAPE_lr), fontsize = 14, color='teal')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(LR)='+str(RMSE_lr), fontsize = 14, color='teal')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='teal')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Linear Regression Calibration vs Laboratory␣
↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[804]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference CO(ppm)',fontsize=18)
plt.ylabel('Calibrated CO(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'LR'], loc = 2, bbox_to_anchor = (1,1))
plt.title('CO Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 254.8
Mean squared error(MSE) = 175410.34
Median absolute error = 158.51
Explain variance score = 0.62
R2 score = 0.62
```



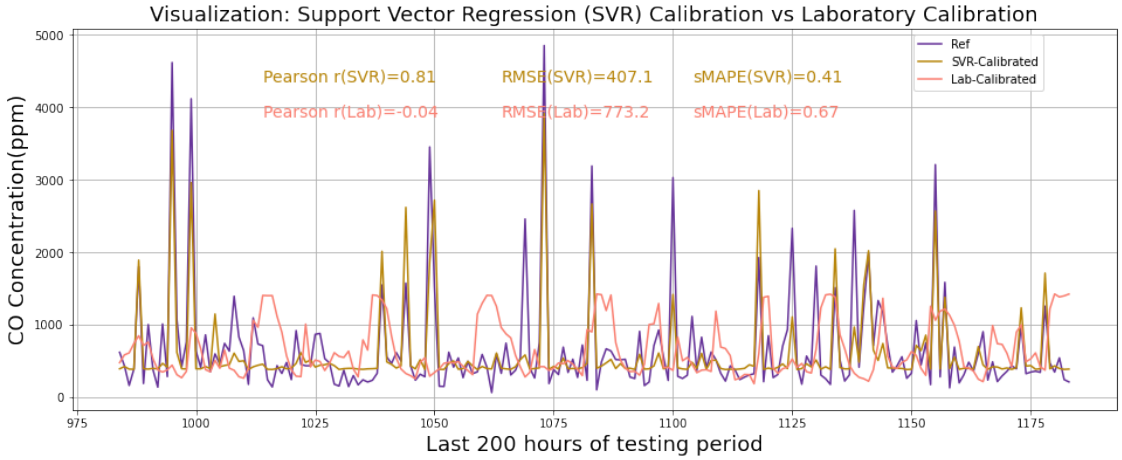
0.3 Model 2 : Support Vector Regression (SVR)

[805]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'poly', degree=5)
regressor.fit(X_train, y_train)
pred = regressor.predict(X_test)
```

[806]:
```
lab1=CO_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_svr_CO=sMAPE_lr
RMSE_svr_CO=RMSE_lr/np.mean(np.array(y_test))
Pearson_svr_CO=Pearson_lr
```

[807]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='darkgoldenrod')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'SVR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('CO Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(SVR)='+str(sMAPE_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(SVR)='+str(RMSE_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(SVR)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
```

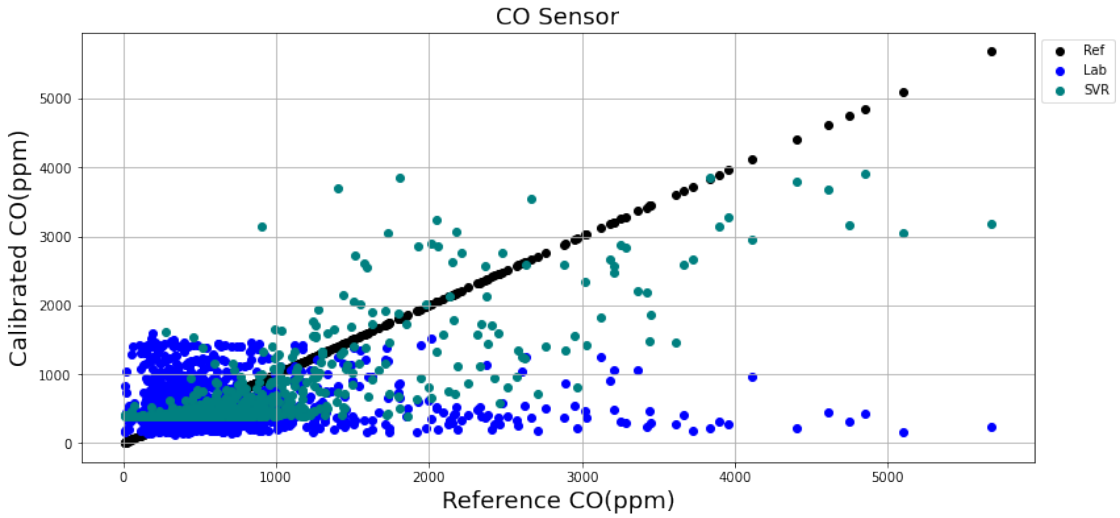
```
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Support Vector Regression (SVR) Calibration vs␣
 ↪Laboratory Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[808]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))

pred_svr=pred
for i in range(len(pred_svr)):
    if pred_svr[i]<0:
        pred_svr[i]=np.mean(pred_svr)
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_svr, c ="teal")

plt.xlabel('Reference CO(ppm)',fontsize=18)
plt.ylabel('Calibrated CO(ppm)',fontsize=18)
```

```
plt.legend(['Ref', 'Lab', 'SVR'] ,loc = 2, bbox_to_anchor = (1,1))
plt.title('CO Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 252.43
Mean squared error(MSE) = 165732.75
Median absolute error = 160.35
Explain variance score = 0.65
R2 score = 0.64
```



0.4 Model 3 : Random Forest

[809]:
```
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 500, random_state = 0)

# fit the regressor with x and y data
regressor.fit(X_train, y_train)
```

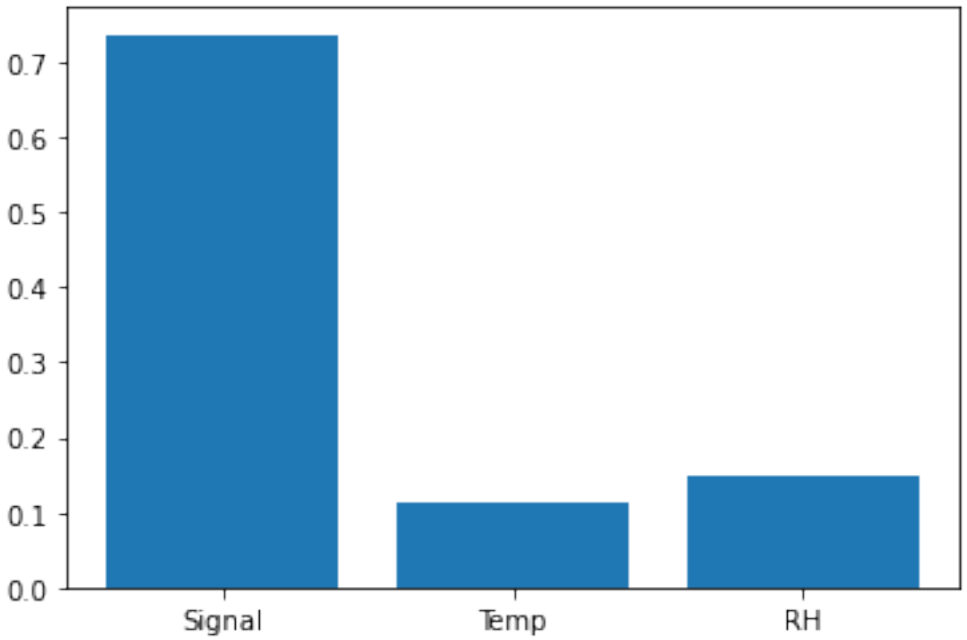
[809]:
```
RandomForestRegressor(n_estimators=500, random_state=0)
```

[810]:
```
regressor.feature_importances_
```

[810]:
```
array([0.73675418, 0.11433631, 0.14890951])
```

[811]:
```
plt.bar(['Signal','Temp','RH'], regressor.feature_importances_)
```

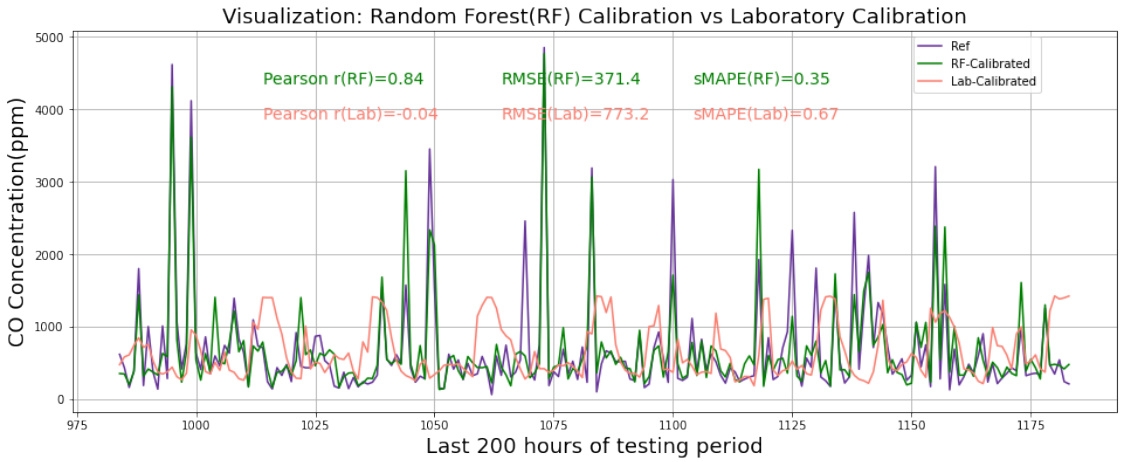
[811]: <BarContainer object of 3 artists>



[812]:
```
pred = regressor.predict(X_test)
lab1=CO_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_rf_CO=sMAPE_lr
RMSE_rf_CO=RMSE_lr/np.mean(np.array(y_test))
Pearson_rf_CO=Pearson_lr
```

[813]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
```

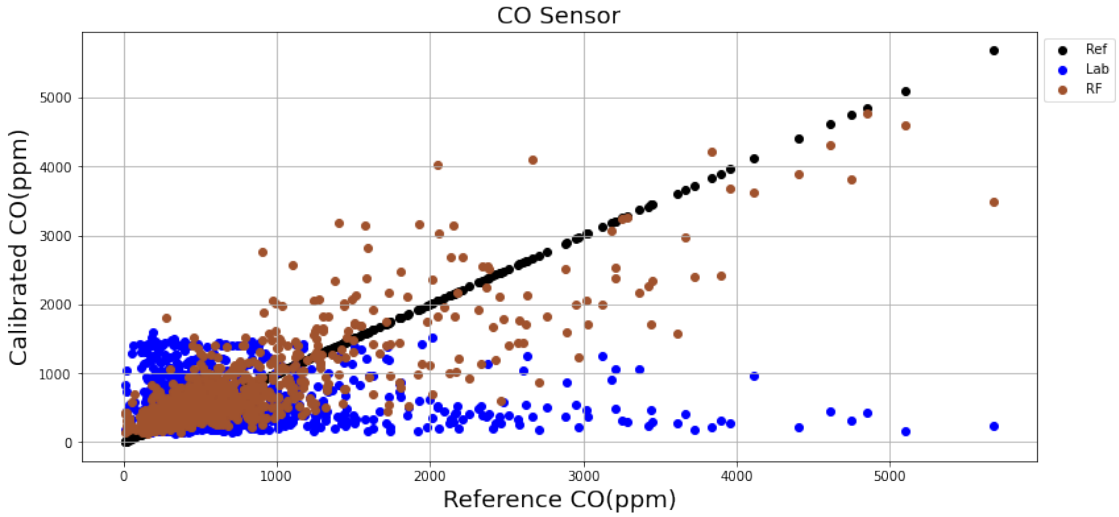
```
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='green')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'RF-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('CO Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(RF)='+str(sMAPE_lr), fontsize = 14, color='green')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(RF)='+str(RMSE_lr), fontsize = 14, color='green')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(RF)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='green')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[814]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
```

```
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_rf=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_rf, c ="sienna")
plt.xlabel('Reference CO(ppm)',fontsize=18)
plt.ylabel('Calibrated CO(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'RF'], loc = 2, bbox_to_anchor = (1,1))
plt.title('CO Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 231.33
Mean squared error(MSE) = 137953.06
Median absolute error = 138.06
Explain variance score = 0.7
R2 score = 0.7
```



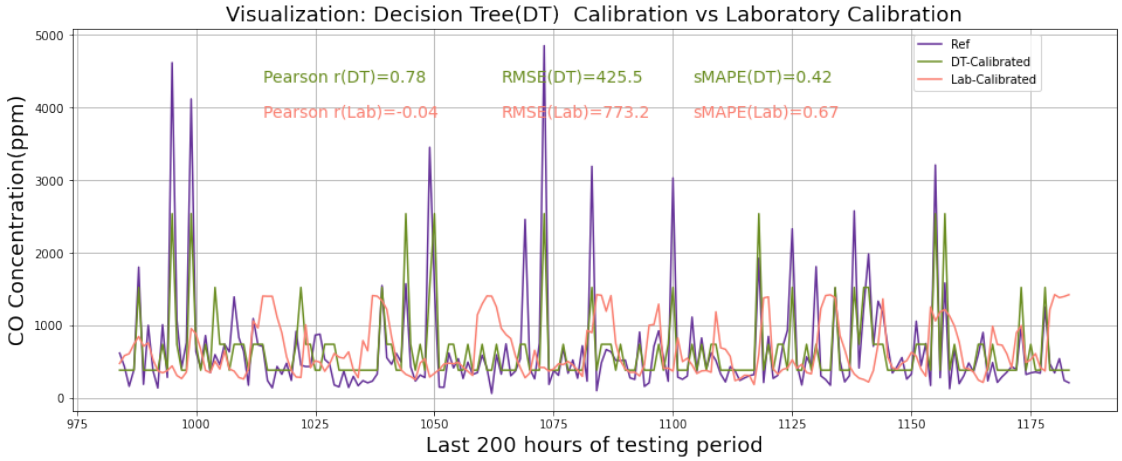
0.5 Model 4: Decision Tree

[815]:
```
from sklearn.tree import DecisionTreeRegressor
regr = DecisionTreeRegressor(max_depth=2)
```

[816]:
```
regr.fit(X_train, y_train)
pred=regr.predict(X_test)
lab1=CO_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
```

```
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
```

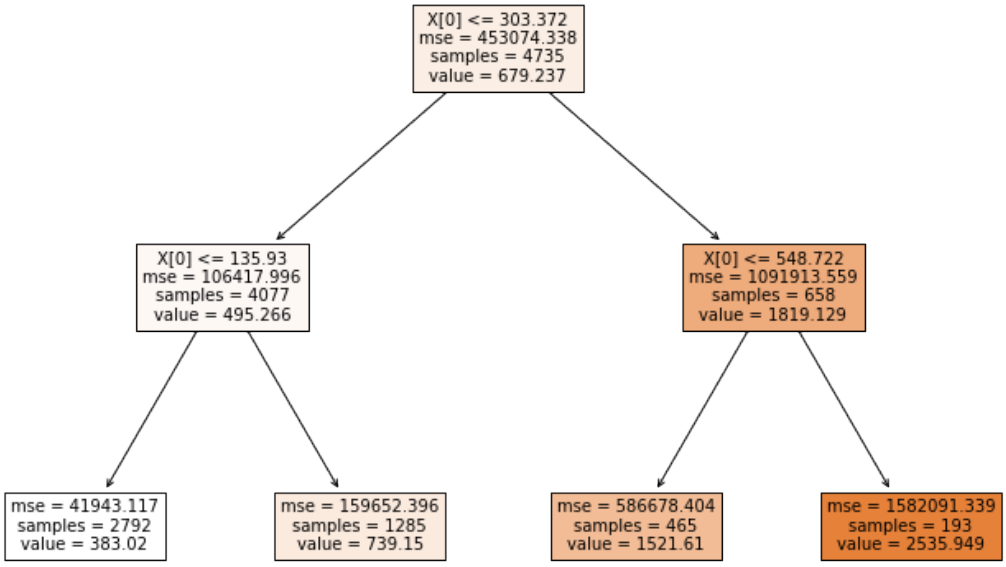
[817]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='olivedrab')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'DT-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('CO Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(DT)='+str(sMAPE_lr), fontsize = 14, color='olivedrab')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(DT)='+str(RMSE_lr), fontsize = 14, color='olivedrab')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(DT)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='olivedrab')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Decision Tree(DT)  Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[818]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
```

```
Regressor model performance:
Mean absolute error(MAE) = 267.54
Mean squared error(MSE) = 181028.18
Median absolute error = 173.94
Explain variance score = 0.6
R2 score = 0.6
```

[819]:
```
from sklearn import tree
from dtreeviz.trees import *
plt.figure(figsize=(12,8))
tree.plot_tree(regr, filled=True, fontsize=10)
plt.show()
```



0.6 Model 5: ANN

[820]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from sklearn.preprocessing import StandardScaler
model = Sequential()
model.add(Dense(3, input_shape = (3,),kernel_initializer='normal', activation=␣
↪'linear'))
model.add(Dense(128,kernel_initializer='normal', activation= 'relu'))
#model.add(Dense(128, kernel_initializer='normal',activation= 'relu'))
model.add(Dense(50, kernel_initializer='normal',activation= 'relu'))
model.add(Dense(1,kernel_initializer='normal',activation='linear',))
sgd = optimizers.Adam(learning_rate=0.01)

model.compile(optimizer = sgd, loss = 'mean_squared_error', metrics= ['mse',␣
↪'mae'])
model.summary()
```

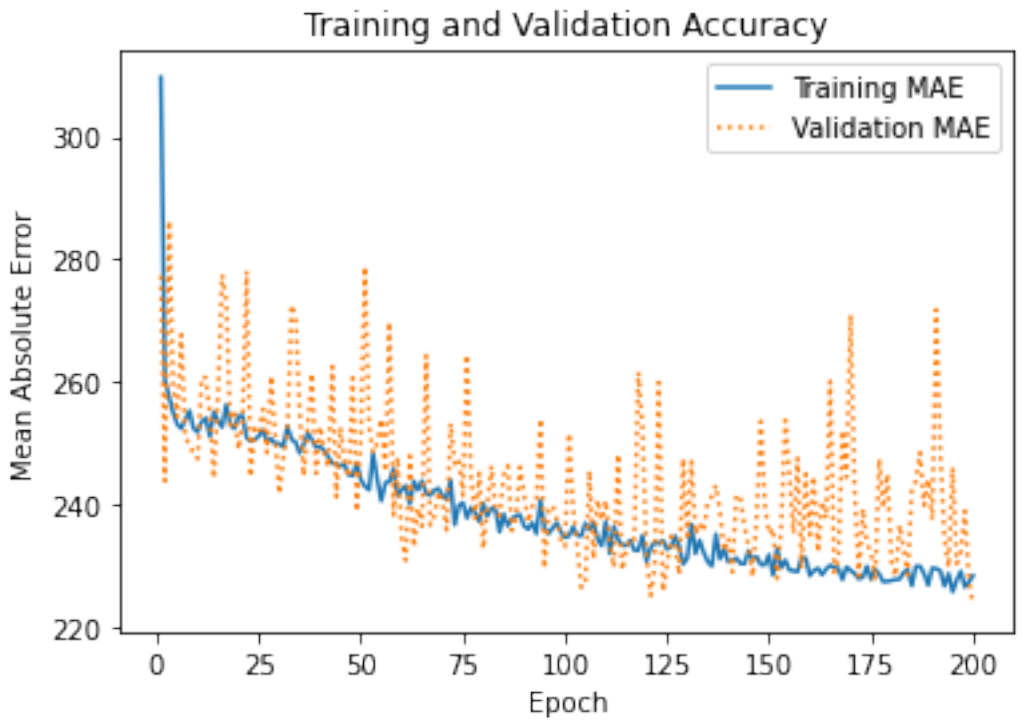
```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense_120 (Dense)            (None, 3)                 12
_________________________________________________________________
dense_121 (Dense)            (None, 128)               512
_________________________________________________________________
dense_122 (Dense)            (None, 50)                6450
_________________________________________________________________
dense_123 (Dense)            (None, 1)                 51
=================================================================
Total params: 7,025
Trainable params: 7,025
Non-trainable params: 0
_________________________________________________________________
```

[821]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled=scaler.transform(X_train)
X_test_scaled=scaler.transform(X_test)
hist=model.fit(X_train_scaled, y_train, batch_size= 10, epochs=200, verbose=␣
 ↪0,validation_split=0.2)
```

[822]:
```
err = hist.history['mae']
val_err = hist.history['val_mae']
epochs = range(1, len(err) + 1)

plt.plot(epochs, err, '-', label='Training MAE')
plt.plot(epochs, val_err, ':', label='Validation MAE')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Mean Absolute Error')
plt.legend(loc='upper right')
plt.plot()
```

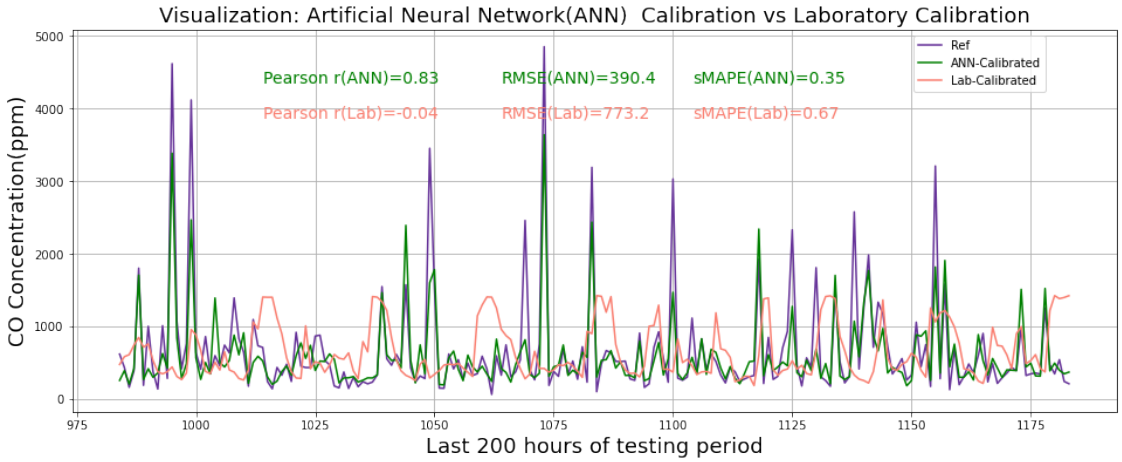
[822]:
```
[]
```



[823]:
```
train_pred = model.predict(X_train_scaled)
test_pred = model.predict(X_test_scaled)
pred=[]
for i in range(len(test_pred)):
    pred.append(sum(list(test_pred[i])))
```

[824]:
```
lab1=CO_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_ann_CO=sMAPE_lr
RMSE_ann_CO=RMSE_lr/np.mean(np.array(y_test))
Pearson_ann_CO=Pearson_lr
```

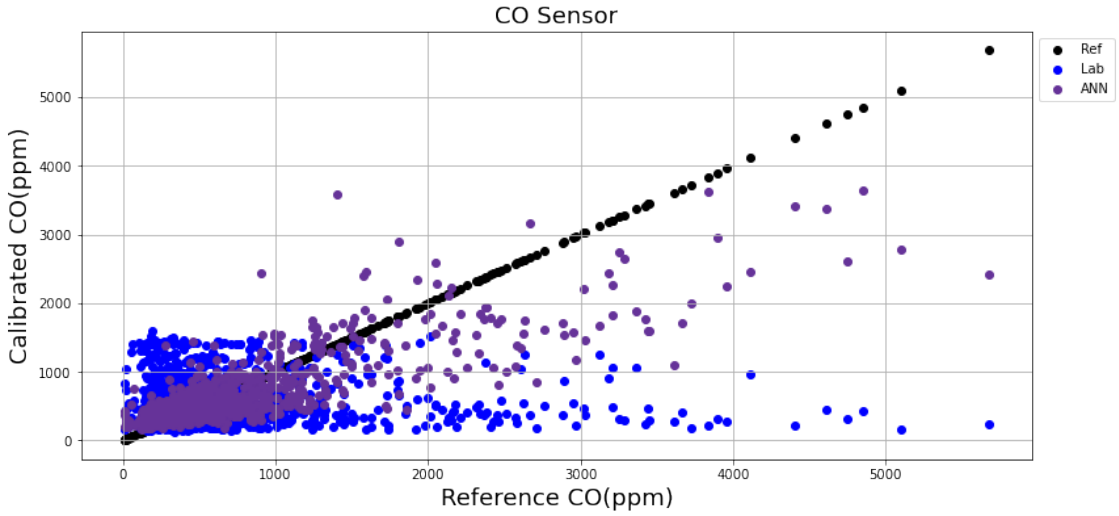
[825]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='green')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'ANN-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('CO Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(ANN)='+str(sMAPE_lr), fontsize = 14, color='green')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(ANN)='+str(RMSE_lr), fontsize = 14, color='green')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(ANN)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='green')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Artificial Neural Network(ANN)  Calibration vs␣
 ↪Laboratory Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[826]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
```

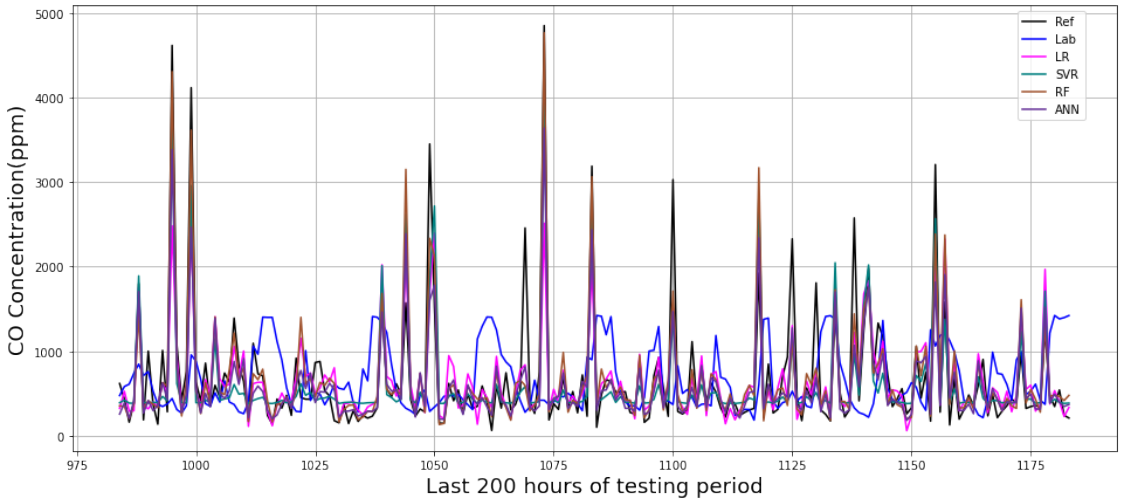
```
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_ann=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_ann, c ="rebeccapurple")

plt.xlabel('Reference CO(ppm)',fontsize=18)
plt.ylabel('Calibrated CO(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab','ANN'], loc = 2, bbox_to_anchor = (1,1))
plt.title('CO Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 226.33
Mean squared error(MSE) = 152389.62
Median absolute error = 127.83
Explain variance score = 0.68
R2 score = 0.67
```

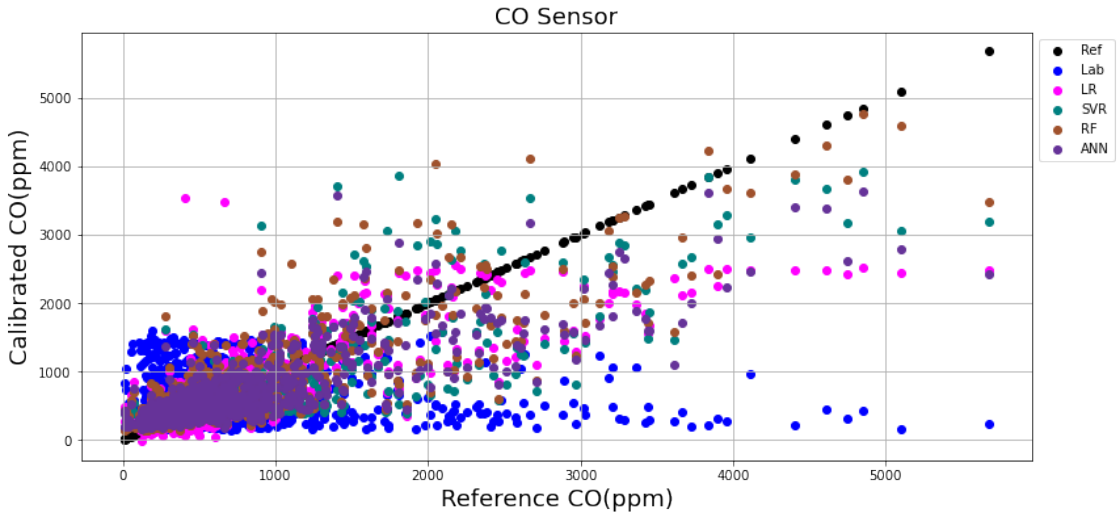


[827]:
```
A=len(y_test)-200

fig= plt.figure(figsize=(16,7))
```

```
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='black')
plt.plot(index[A:],lab1[A:], color='blue')
plt.plot(index[A:],pred_lr[A:], color='magenta')
plt.plot(index[A:],pred_svr[A:], color='teal')
plt.plot(index[A:],pred_rf[A:], color='sienna')
plt.plot(index[A:],pred_ann[A:], color='rebeccapurple')

plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.ylabel('CO Concentration(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'LR','SVR','RF','ANN'], loc = 2, bbox_to_anchor = (0.
 ↪9,1))
#plt.title('CO Sensor',fontsize=18 )
plt.grid(True)
```



[828]:
```
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.scatter(y_test,pred_svr, c ="teal")
plt.scatter(y_test,pred_rf, c ="sienna")
plt.scatter(y_test,pred_ann, c ="rebeccapurple")

plt.xlabel('Reference CO(ppm)',fontsize=18)
plt.ylabel('Calibrated CO(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'LR','SVR','RF','ANN'], loc = 2, bbox_to_anchor =␣
 ↪(1,1))
plt.title('CO Sensor',fontsize=18 )
plt.grid(True)
```



1 NO2 Calibration

[829]:
```
import pandas as pd
import scipy.io
import numpy as np
mat = scipy.io.loadmat('NO2final.mat')
#cardio_df = pd.DataFrame(mat)

len(mat['NO2final'])
L_NO2final=[]
for i in range(len(mat['NO2final'])):
    L_NO2final.append(list(mat['NO2final'][i]))

NO2_data=pd.DataFrame(L_NO2final,columns=['Lab1',␣
 ↪'Lab2','Lab3','Lab4','Temp','RH','Ref','Time'])
Time=NO2_data['Time'].to_list()
Time=np.array(Time)
Date=pd.to_datetime(Time-719529,unit='d').round('h')
NO2_data['Date'] = Date.tolist()
NO2_data=NO2_data.set_index('Date')
NO2_data.drop('Time',axis = 1, inplace = True)

mat = scipy.io.loadmat('NO2_raw.mat')

L_NO2_raw=[]
for i in range(len(mat['NO2_raw'])):
    L_NO2_raw.append(list(mat['NO2_raw'][i]))
```

```
NO2_raw=pd.DataFrame(L_NO2_raw,columns=['WE', 'AE','Temp','RH','Time'])
NO2_raw.head()
NO2_data.insert(loc = 0,
          column = 'WE',
          value = NO2_raw['WE'].to_list())

NO2_data.insert(loc = 1,
          column = 'AE',
          value = NO2_raw['AE'].to_list())


NO2_data=NO2_data.interpolate()
WE=np.array(NO2_data['WE'].to_list())
AE=np.array(NO2_data['AE'].to_list())
Signal=list(WE-AE)
NO2_data.insert(loc = 2,
          column = 'Signal',
          value = Signal)
NO2_data.head()
```

[829]:

Date	WE	AE	Signal	Lab1	Lab2	Lab3	Lab4	Temp	RH	Ref
2019-10-02 12:00:00	157.696563	149.846563	7.850000	460.448301	11.259601	-54.281210	-844.685734	26.378438	58.063437	9.613288
2019-10-02 13:00:00	0.586429	39.937143	-39.350714	1186.045510	-64.843326	-173.647930	-1383.373409	25.502791	59.868837	15.181394
2019-10-02 16:00:00	17.601343	40.504328	-22.902985	1293.158435	11.194037	-180.488333	-1212.985070	30.827910	49.008060	13.091546
2019-10-02 17:00:00	190.049701	180.774030	9.275672	219.575228	-6.131452	-65.833055	-658.643041	30.047164	51.259851	18.506848
2019-10-03 16:00:00	226.502857	202.540000	23.962857	86.725922	23.809224	-16.484097	-552.026071	29.441429	52.018571	14.396420

[830]:
```
NO2_data.shape
```

[830]: (6021, 10)

[831]:
```
NO2_data.describe()
```

[831]:

	WE	AE	Signal	Lab1	Lab2
count	6021.000000	6021.000000	6021.000000	6021.000000	6021.000000
mean	229.111036	208.110467	21.000569	25.722637	7.072620
std	12.722230	6.938218	8.591080	55.107618	29.587763
min	0.564000	38.872000	-41.667777	-280.068781	-164.140427
25%	220.486982	205.346944	15.064583	7.989353	-12.228035
50%	233.389663	209.662979	23.171917	21.358645	14.460956
75%	236.966050	211.015514	26.593889	38.074513	26.521991
max	255.366500	227.651463	93.160938	1461.434904	293.708249

	Lab3	Lab4	Temp	RH	Ref
count	6021.000000	6021.000000	6021.000000	6021.000000	6021.000000
mean	-28.377323	-697.352025	18.257658	67.307475	24.310285
std	31.418703	111.688152	7.601533	18.453782	18.579753
min	-235.559257	-1393.920200	1.037101	11.428168	1.050620
25%	-49.497866	-769.552964	12.409531	53.636250	11.589200
50%	-19.292374	-722.425997	16.868500	72.245143	19.618169
75%	-7.293931	-655.081265	22.844245	82.935250	31.467247
max	226.584223	-165.049809	44.748056	93.012486	154.545243

[832]:
```
NO2_Data=NO2_data[['Signal','Lab1','Temp','RH','Ref']]
NO2_Data=NO2_Data[(NO2_Data[NO2_Data.columns] >= 0).all(axis=1)]
#CO_data=CO_data.resample('D').mean()
NO2_Data=NO2_Data.dropna()
NO2_Data.shape
```

[832]:
```
(4789, 5)
```

[833]:
```
from sklearn.metrics import r2_score, mean_squared_error
rmse = np.sqrt(mean_squared_error( NO2_data['Lab1'], NO2_data['Ref'] ))
rmse
```

[833]:
```
58.8760420184478
```

1.1 Model 1: Linear Regression (LR)

[834]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error as mae
import sklearn.metrics as sm
import matplotlib.pyplot as plt
X=NO2_Data[['Signal','Temp','RH']]
y=NO2_Data['Ref']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

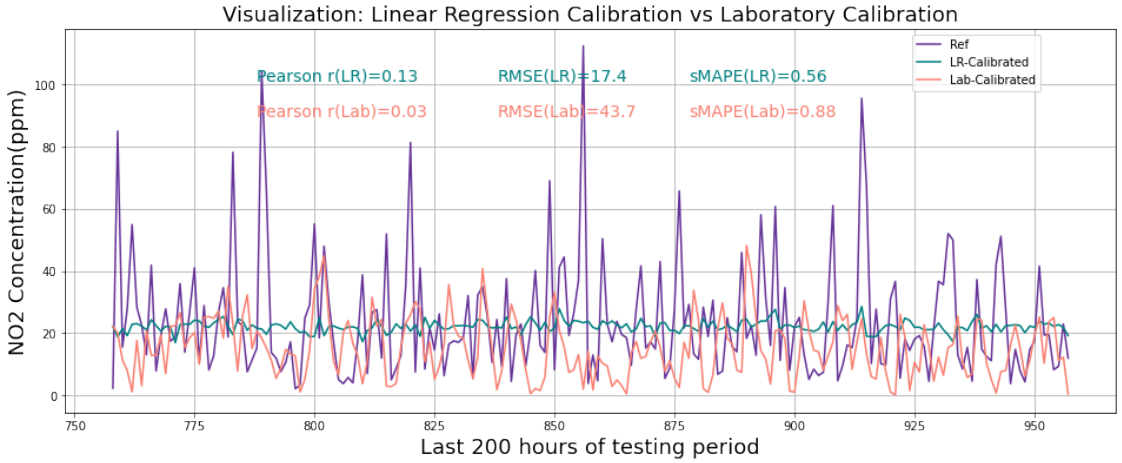
```
len(X_test)
```

[834]: 958

[835]:
```
lr = LinearRegression()
model = lr.fit(X_train, y_train)
pred = model.predict(X_test)
lab1=NO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_lr_NO2=sMAPE_lr
RMSE_lr_NO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_lr_NO2=Pearson_lr
sMAPE_lab_NO2=sMAPE_lab
RMSE_lab_NO2=RMSE_lab/np.mean(np.array(lab1))
Pearson_lab_NO2=Pearson_lab
```

[836]:
```
A=len(y_test)-200
B=A+120
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])


fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='teal')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'LR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
↪= (0.8,1))
plt.ylabel('NO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(LR)='+str(sMAPE_lr), fontsize = 14, color='teal')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(LR)='+str(RMSE_lr), fontsize = 14, color='teal')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='teal')
```

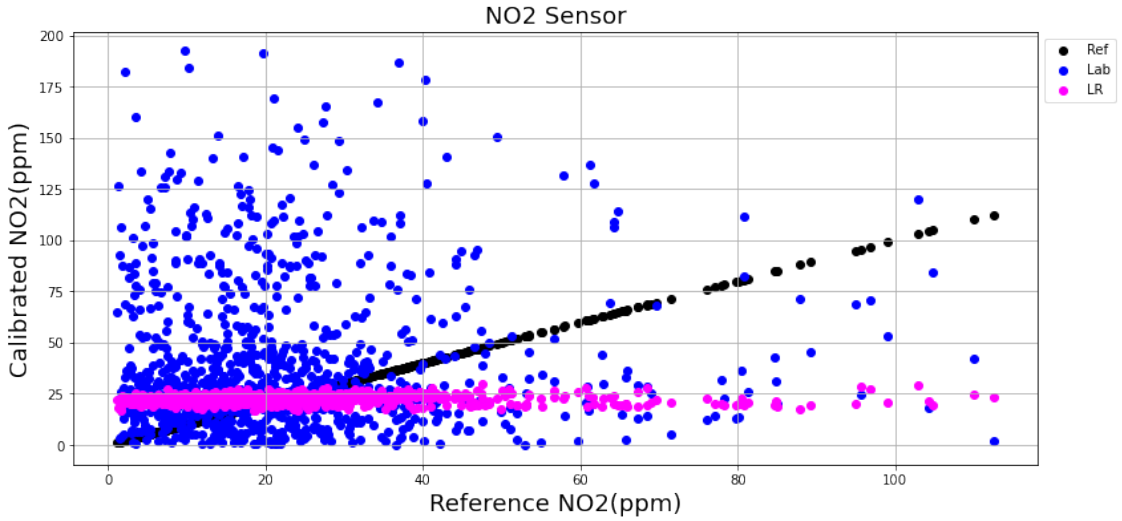
```
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Linear Regression Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[837]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference NO2(ppm)',fontsize=18)
plt.ylabel('Calibrated NO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'LR'], loc = 2, bbox_to_anchor = (1,1))
plt.title('NO2 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 12.14
```

```
Mean squared error(MSE) = 302.19
Median absolute error = 9.65
Explain variance score = 0.02
R2 score = 0.01
```



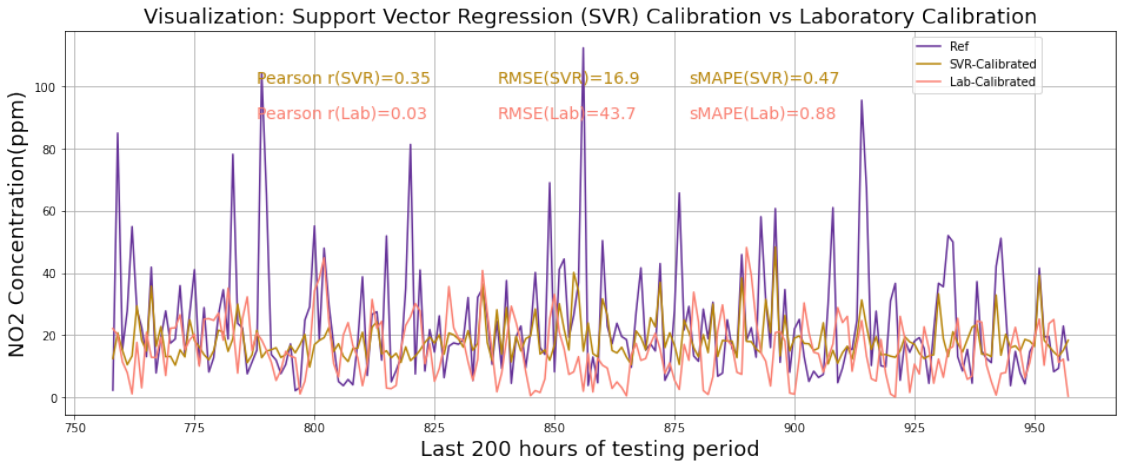
1.2 Model 2: Support Vector Regression (SVR)

[838]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'poly', degree=4)
regressor.fit(X_train, y_train)
pred = regressor.predict(X_test)
```

[839]:
```
lab1=NO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_svr_NO2=sMAPE_lr
RMSE_svr_NO2=RMSE_lr/np.mean(np.array(y_test))
```

```
Pearson_svr_NO2=Pearson_lr
```

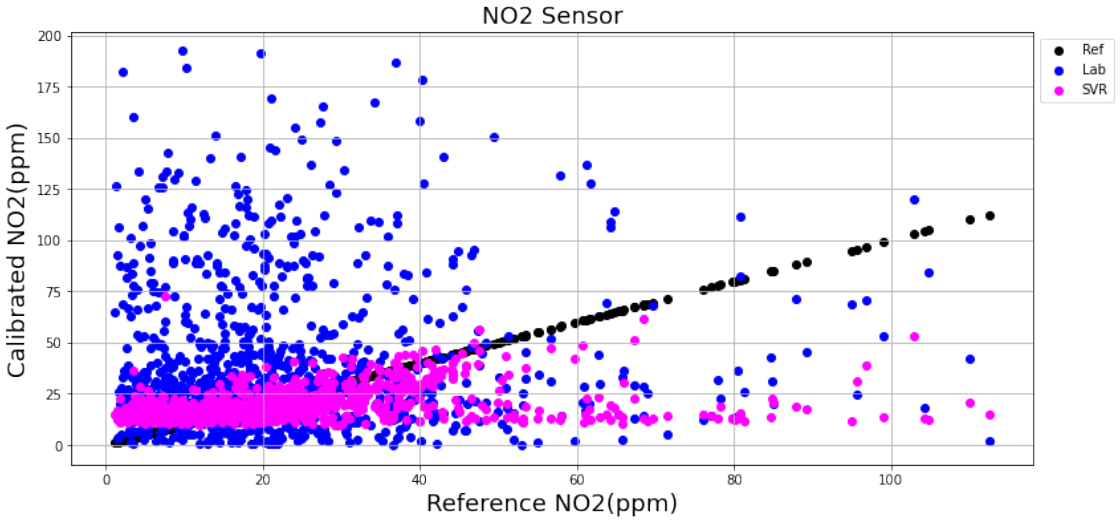
[840]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='darkgoldenrod')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'SVR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('NO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(SVR)='+str(sMAPE_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(SVR)='+str(RMSE_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(SVR)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Support Vector Regression (SVR) Calibration vs␣
 ↪Laboratory Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[841]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
```

```
Regressor model performance:
Mean absolute error(MAE) = 9.98
Mean squared error(MSE) = 284.22
Median absolute error = 6.05
Explain variance score = 0.11
R2 score = 0.07
```

[842]:
```
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference NO2(ppm)',fontsize=18)
plt.ylabel('Calibrated NO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'SVR'], loc = 2, bbox_to_anchor = (1,1))
plt.title('NO2 Sensor',fontsize=18 )
plt.grid(True)
```



1.3 Model 3: Random Forest

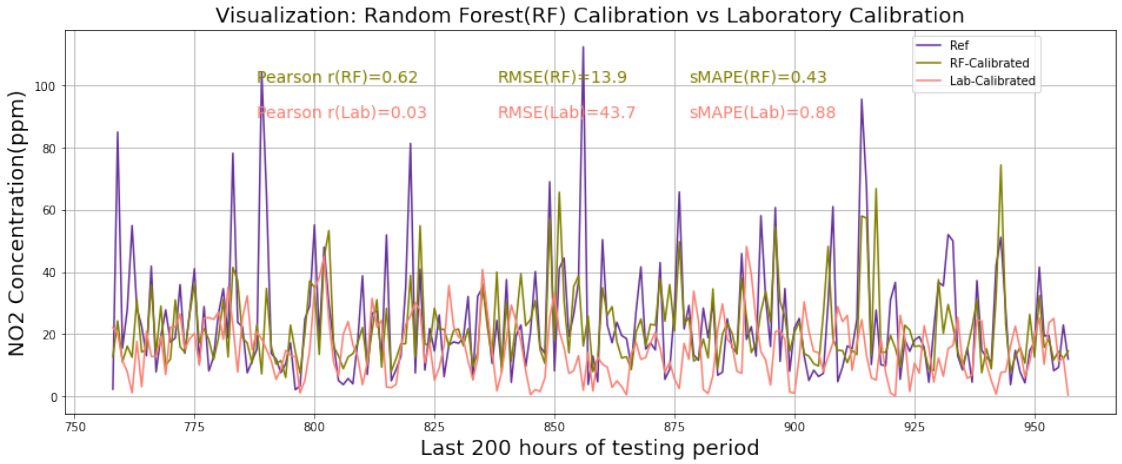
[843]:
```
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 500, random_state = 0)

# fit the regressor with x and y data
regressor.fit(X_train, y_train)
```

[843]:
```
RandomForestRegressor(n_estimators=500, random_state=0)
```

[844]:
```
pred = regressor.predict(X_test)
lab1=NO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_rf_NO2=sMAPE_lr
RMSE_rf_NO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_rf_NO2=Pearson_lr
```

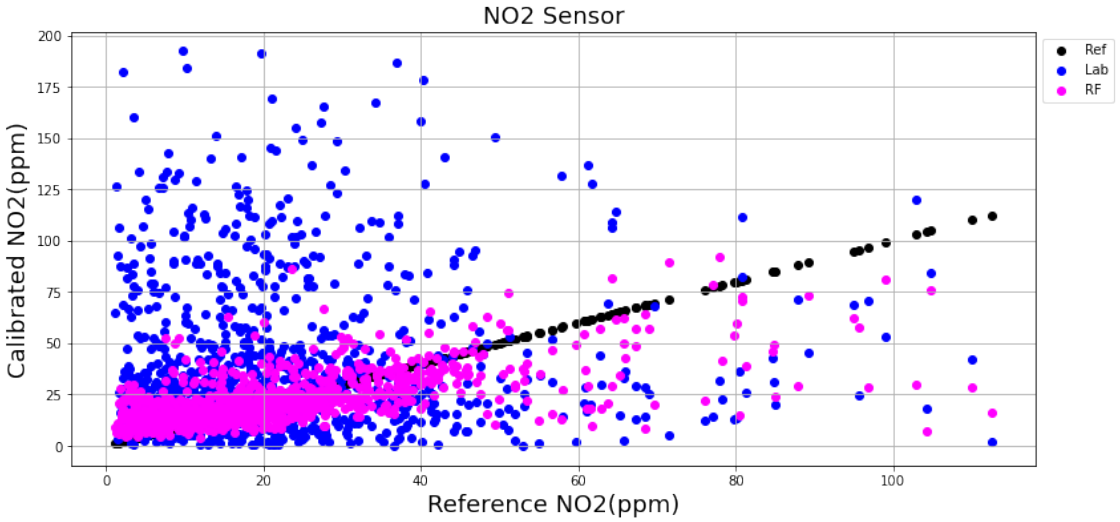
[845]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='olive')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'RF-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('NO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(RF)='+str(sMAPE_lr), fontsize = 14, color='olive')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(RF)='+str(RMSE_lr), fontsize = 14, color='olive')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
```

```
plt.text(B-90, C, 'Pearson r(RF)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='olive')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[846]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference NO2(ppm)',fontsize=18)
plt.ylabel('Calibrated NO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'RF'], loc = 2, bbox_to_anchor = (1,1))
plt.title('NO2 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 8.98
Mean squared error(MSE) = 194.08
Median absolute error = 5.89
Explain variance score = 0.37
R2 score = 0.37
```



[847]:
```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test, pred)
```

[847]: 0.7292700945769319

1.4 Model 4 : ANN

[848]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from sklearn.preprocessing import StandardScaler
model = Sequential()
model.add(Dense(3, input_shape = (3,),kernel_initializer='normal', activation=␣
↪'linear'))
model.add(Dense(128,kernel_initializer='normal', activation= 'relu'))
model.add(Dense(50, kernel_initializer='normal',activation= 'relu'))
#model.add(Dense(100, kernel_initializer='normal',activation= 'relu'))
model.add(Dense(1,kernel_initializer='normal',activation='linear',))
sgd = optimizers.Adam(learning_rate=0.01)
```

```
model.compile(optimizer = sgd, loss = 'mean_squared_error', metrics= ['mse',␣
↪'mae'])
model.summary()
```

```
Model: "sequential_31"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_124 (Dense)            (None, 3)                 12
_________________________________________________________________
dense_125 (Dense)            (None, 128)               512
_________________________________________________________________
dense_126 (Dense)            (None, 50)                6450
_________________________________________________________________
dense_127 (Dense)            (None, 1)                 51
=================================================================
Total params: 7,025
Trainable params: 7,025
Non-trainable params: 0
_________________________________________________________________
```

[849]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled=scaler.transform(X_train)
X_test_scaled=scaler.transform(X_test)
model.fit(X_train_scaled, y_train, batch_size= 100, epochs=200, verbose= 0)
```

[849]:
```
<tensorflow.python.keras.callbacks.History at 0x22a1d48b0>
```

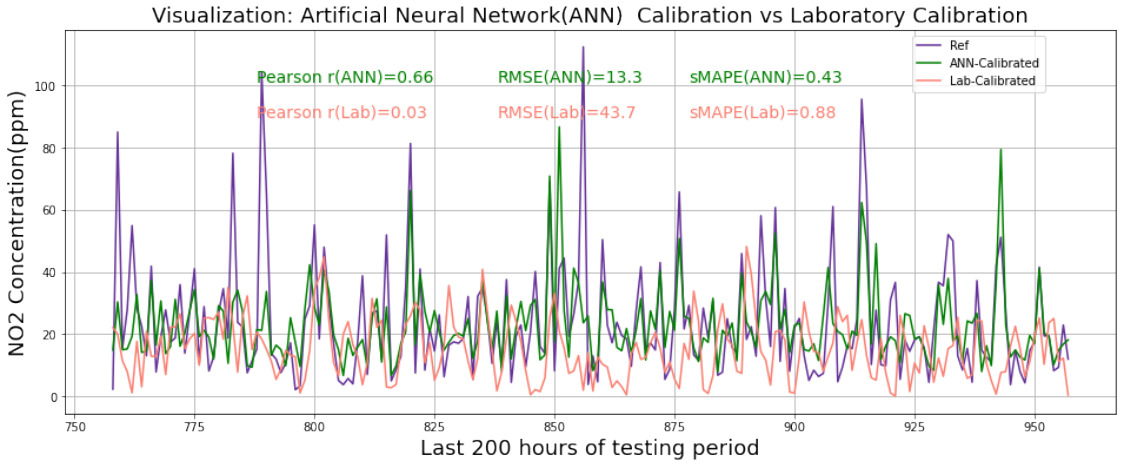
[850]:
```
train_pred = model.predict(X_train_scaled)
test_pred = model.predict(X_test_scaled)
pred=[]
for i in range(len(test_pred)):
    pred.append(sum(list(test_pred[i])))
len(y_test)
```

[850]:
```
958
```

[851]:
```
lab1=NO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
```

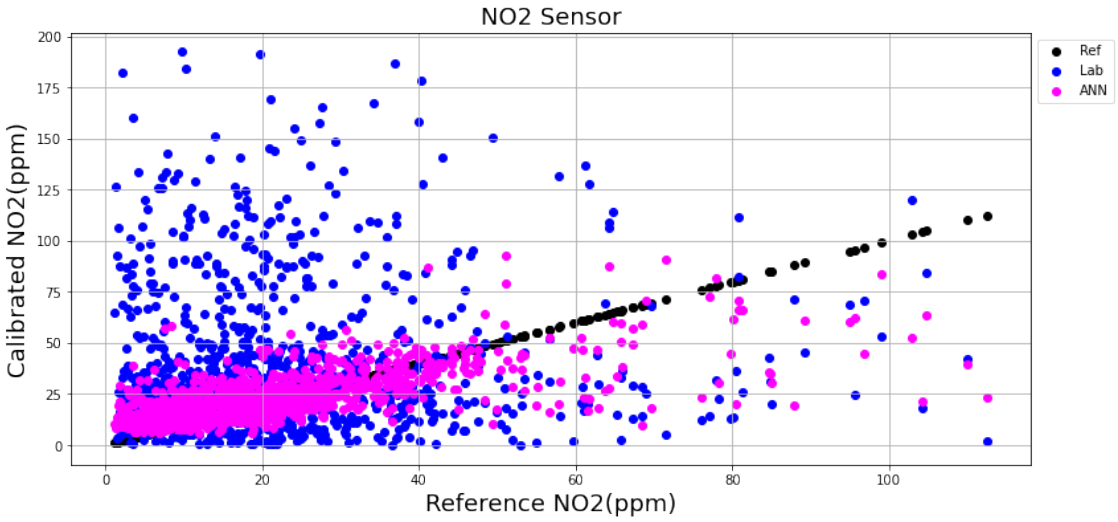
```
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_ann_NO2=sMAPE_lr
RMSE_ann_NO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_ann_NO2=Pearson_lr
```

[852]:
```
A=len(y_test)-200
D=max(y_test[A:])-0.2*max(y_test[A:])
C=max(y_test[A:])-0.1*max(y_test[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='green')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'ANN-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('NO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(ANN)='+str(sMAPE_lr), fontsize = 14, color='green')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(ANN)='+str(RMSE_lr), fontsize = 14, color='green')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(ANN)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='green')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Artificial Neural Network(ANN)  Calibration vs␣
 ↪Laboratory Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[853]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference NO2(ppm)',fontsize=18)
plt.ylabel('Calibrated NO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'ANN'], loc = 2, bbox_to_anchor = (1,1))
plt.title('NO2 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 8.84
Mean squared error(MSE) = 177.18
Median absolute error = 6.26
Explain variance score = 0.43
R2 score = 0.42
```



SO2 Calibration

[854]:
```
import pandas as pd
import scipy.io
import numpy as np
mat = scipy.io.loadmat('So2final.mat')

L_SO2final=[]
for i in range(len(mat['SO2final'])):
    L_SO2final.append(list(mat['SO2final'][i]))

SO2_data=pd.DataFrame(L_SO2final,columns=['Lab1',␣
 ↪'Lab2','Lab3','Lab4','Temp','RH','Ref','Time'])
Time=SO2_data['Time'].to_list()
Time=np.array(Time)
Date=pd.to_datetime(Time-719529,unit='d').round('h')
SO2_data['Date'] = Date.tolist()
SO2_data=SO2_data.set_index('Date')
SO2_data.drop('Time',axis = 1, inplace = True)

mat = scipy.io.loadmat('SO2_raw.mat')

L_SO2_raw=[]
for i in range(len(mat['SO2_raw'])):
    L_SO2_raw.append(list(mat['SO2_raw'][i]))

SO2_raw=pd.DataFrame(L_SO2_raw,columns=['WE', 'AE','Temp','RH','Time'])
SO2_raw.head()
SO2_data.insert(loc = 0,
          column = 'WE',
```

```
          value = SO2_raw['WE'].to_list())

SO2_data.insert(loc = 1,
          column = 'AE',
          value = SO2_raw['AE'].to_list())


SO2_data=SO2_data.interpolate()
WE=np.array(SO2_data['WE'].to_list())
AE=np.array(SO2_data['AE'].to_list())
Signal=list(WE-AE)
SO2_data.insert(loc = 2,
          column = 'Signal',
          value = Signal)

SO2_data.head()
```

[854]:

Date	WE	AE	Signal	Lab1	Lab2	Lab3	Lab4	Temp	RH	Ref
2019-10-02 12:00:00	511.980000	375.956250	136.023750	336.491001	395.131511	258.535708	-624.848616	26.378438	58.063437	1.701245
2019-10-02 13:00:00	511.980000	508.330476	3.649524	-339.232168	-68.687897	-313.588071	-619.380130	25.502791	59.868837	1.495635
2019-10-02 16:00:00	511.980000	442.351642	69.628358	-77.358106	108.340330	-219.979244	-680.245645	30.827910	49.008060	1.102313
2019-10-02 17:00:00	435.447463	352.494627	82.952836	221.907307	244.067240	121.645425	-897.011152	30.047164	51.259851	1.383004
2019-10-03 16:00:00	358.812857	336.112857	22.700000	84.566512	77.784764	-11.080530	-1118.485141	29.441429	52.018571	1.402984

[855]:
```
SO2_data.shape
```

[855]: (6021, 10)

[856]:
```
SO2_data.describe()
```

[856]:
```
                WE           AE       Signal         Lab1         Lab2  \
count  6021.000000  6021.000000  6021.000000  6021.000000  6021.000000
mean    346.317991   343.672928     2.645063     4.426280    10.876447
std      11.844320     5.652805     9.278214    31.465055    27.662360
min     327.665107   336.112857   -19.034635  -961.004985  -505.040176
25%     339.505000   341.720921    -3.212714   -10.826914    -5.415598
50%     343.395667   342.801650     0.383073     0.567498     5.306895
75%     349.409378   344.366921     5.712667    15.114913    20.328606
max     511.984380   511.984380   137.776860   413.434407   419.568327

               Lab3         Lab4         Temp           RH          Ref
count  6021.000000  6021.000000  6021.000000  6021.000000  6021.000000
mean    -89.010607 -1120.030522    18.258139    67.305016     1.715263
std      34.530699    64.509983     7.601563    18.453999     0.995402
min   -1302.528007 -1777.717366     1.035000    11.428168    -0.742694
25%    -103.105554 -1127.334289    12.409531    53.576341     1.092307
50%     -92.472035 -1115.412344    16.872568    72.220802     1.523944
75%     -78.691378 -1099.959520    22.849585    82.938548     2.061193
max     322.268946  -599.346047    44.748056    93.012486    18.687856
```

[857]:
```
SO2_Data=SO2_data[['Signal','Lab1','Temp','RH','Ref']]
SO2_Data=SO2_Data[(SO2_Data[SO2_Data.columns] >= 0).all(axis=1)]
#CO_data=CO_data.resample('D').mean()
SO2_Data=SO2_Data.dropna()
SO2_Data.shape
```

[857]:
```
(2822, 5)
```

[858]:
```
from sklearn.metrics import r2_score, mean_squared_error
rmse = np.sqrt(mean_squared_error( SO2_data['Lab1'], SO2_data['Ref'] ))
rmse
```

[858]:
```
31.404786353406134
```

2 Model 1: Linear Regression (LR)

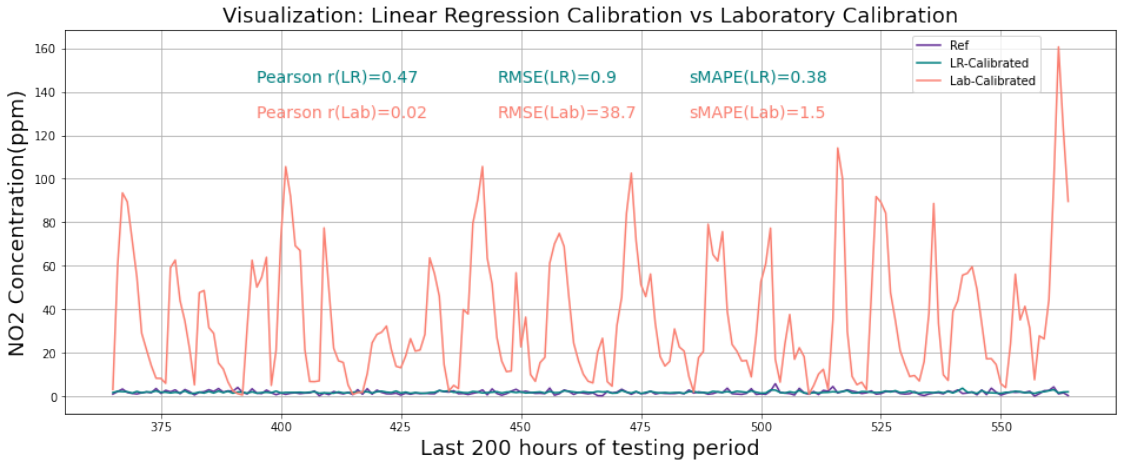
[859]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error as mae
import sklearn.metrics as sm
import matplotlib.pyplot as plt
X=SO2_Data[['Signal','Temp','RH']]
y=SO2_Data['Ref']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
len(X_test)
```

[859]: 565

[860]:
```
lr = LinearRegression()
model = lr.fit(X_train, y_train)
pred = model.predict(X_test)
lab1=SO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_lr_SO2=sMAPE_lr
RMSE_lr_SO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_lr_SO2=Pearson_lr
sMAPE_lab_SO2=sMAPE_lab
RMSE_lab_SO2=RMSE_lab/np.mean(np.array(lab1))
Pearson_lab_SO2=Pearson_lab
```

[861]:
```
A=len(y_test)-200
D=max(lab1[A:])-0.2*max(lab1[A:])
C=max(lab1[A:])-0.1*max(lab1[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='teal')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'LR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('NO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(LR)='+str(sMAPE_lr), fontsize = 14, color='teal')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(LR)='+str(RMSE_lr), fontsize = 14, color='teal')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='teal')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
```

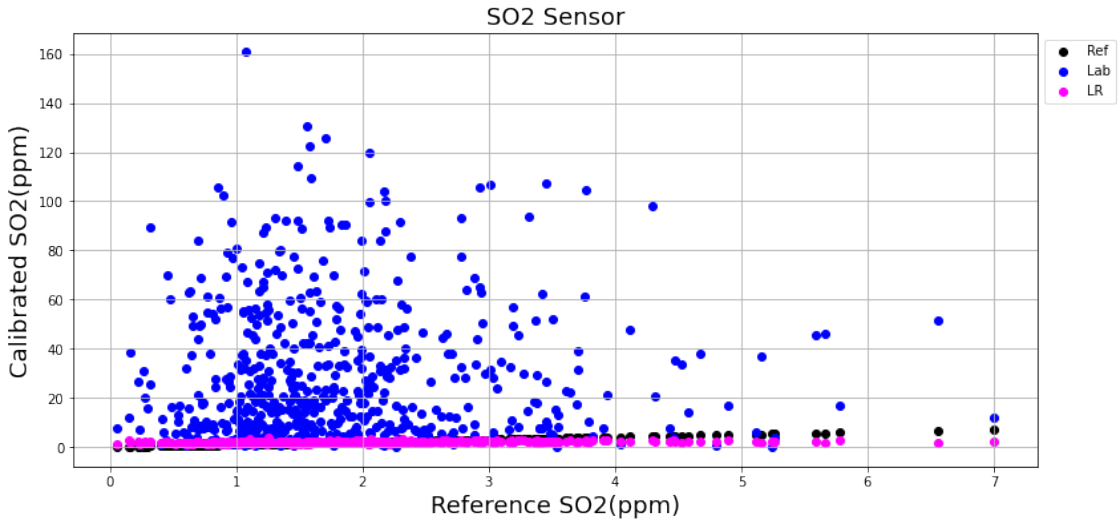
```
plt.title('Visualization: Linear Regression Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[862]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference SO2(ppm)',fontsize=18)
plt.ylabel('Calibrated SO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'LR'], loc = 2, bbox_to_anchor = (1,1))
plt.title('SO2 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.67
Mean squared error(MSE) = 0.81
Median absolute error = 0.53
Explain variance score = 0.21
```

```
R2 score = 0.21
```

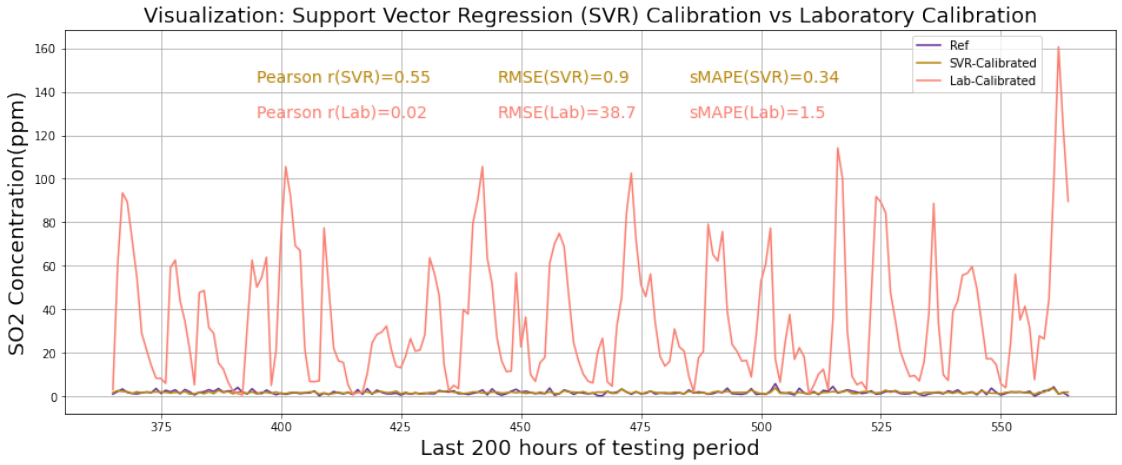


3 Model 2: SVR

[863]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'poly', degree=4)
regressor.fit(X_train, y_train)
pred = regressor.predict(X_test)
```

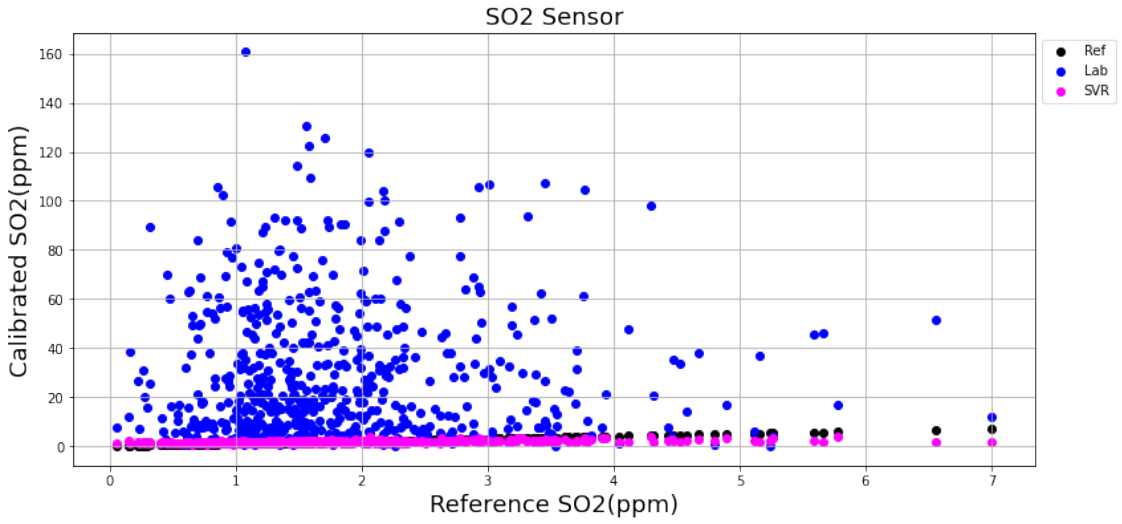
[864]:
```
lab1=SO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_svr_SO2=sMAPE_lr
RMSE_svr_SO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_svr_SO2=Pearson_lr
```

[865]:
```
A=len(y_test)-200
D=max(lab1[A:])-0.2*max(lab1[A:])
C=max(lab1[A:])-0.1*max(lab1[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='darkgoldenrod')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'SVR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('SO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(SVR)='+str(sMAPE_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(SVR)='+str(RMSE_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(SVR)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='darkgoldenrod')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Support Vector Regression (SVR) Calibration vs␣
 ↪Laboratory Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[866]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference SO2(ppm)',fontsize=18)
plt.ylabel('Calibrated SO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'SVR'], loc = 2, bbox_to_anchor = (1,1))
plt.title('SO2 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.6
Mean squared error(MSE) = 0.73
Median absolute error = 0.47
Explain variance score = 0.3
R2 score = 0.29
```



Model 3: Random Forest

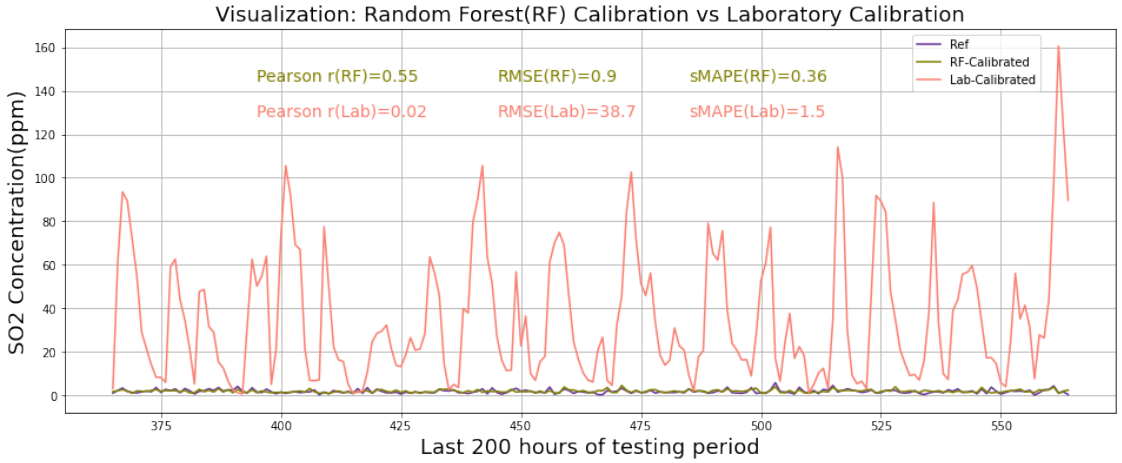
[867]:
```
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 500, random_state = 0)

# fit the regressor with x and y data
regressor.fit(X_train, y_train)
```

[867]:
```
RandomForestRegressor(n_estimators=500, random_state=0)
```

[868]:
```
pred = regressor.predict(X_test)
lab1=SO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_rf_SO2=sMAPE_lr
RMSE_rf_SO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_rf_SO2=Pearson_lr
```

[869]:
```
A=len(y_test)-200
D=max(lab1[A:])-0.2*max(lab1[A:])
C=max(lab1[A:])-0.1*max(lab1[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='olive')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'RF-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('SO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(RF)='+str(sMAPE_lr), fontsize = 14, color='olive')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(RF)='+str(RMSE_lr), fontsize = 14, color='olive')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(RF)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='olive')
```

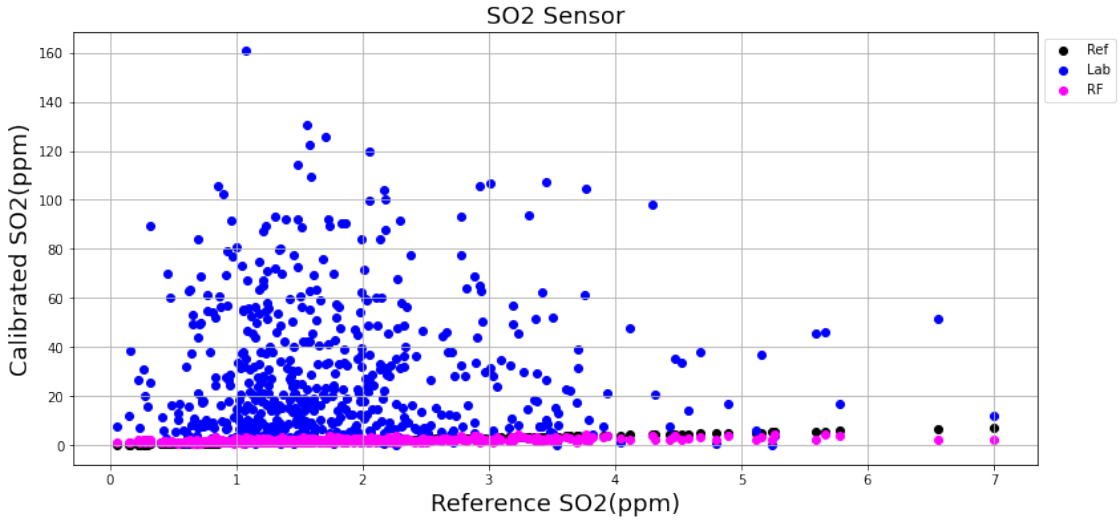
```
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[870]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference SO2(ppm)',fontsize=18)
plt.ylabel('Calibrated SO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'RF'], loc = 2, bbox_to_anchor = (1,1))
plt.title('SO2 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.63
```

```
Mean squared error(MSE) = 0.72
Median absolute error = 0.49
Explain variance score = 0.3
R2 score = 0.29
```



4 Model 4 : ANN

[871]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from sklearn.preprocessing import StandardScaler
model = Sequential()
model.add(Dense(3, input_shape = (3,),kernel_initializer='normal', activation=␣
↪'linear'))
model.add(Dense(128,kernel_initializer='normal', activation= 'relu'))
model.add(Dense(50, kernel_initializer='normal',activation= 'relu'))
#model.add(Dense(100, kernel_initializer='normal',activation= 'relu'))
model.add(Dense(1,kernel_initializer='normal',activation='linear',))
sgd = optimizers.Adam(learning_rate=0.01)

model.compile(optimizer = sgd, loss = 'mean_squared_error', metrics= ['mse',␣
↪'mae'])
model.summary()
```

```
Model: "sequential_32"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense_128 (Dense)            (None, 3)                 12
_________________________________________________________________
dense_129 (Dense)            (None, 128)               512
_________________________________________________________________
dense_130 (Dense)            (None, 50)                6450
_________________________________________________________________
dense_131 (Dense)            (None, 1)                 51
=================================================================
Total params: 7,025
Trainable params: 7,025
Non-trainable params: 0
_________________________________________________________________
```

[872]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled=scaler.transform(X_train)
X_test_scaled=scaler.transform(X_test)
model.fit(X_train_scaled, y_train, batch_size= 200, epochs=100, verbose= 0)
```

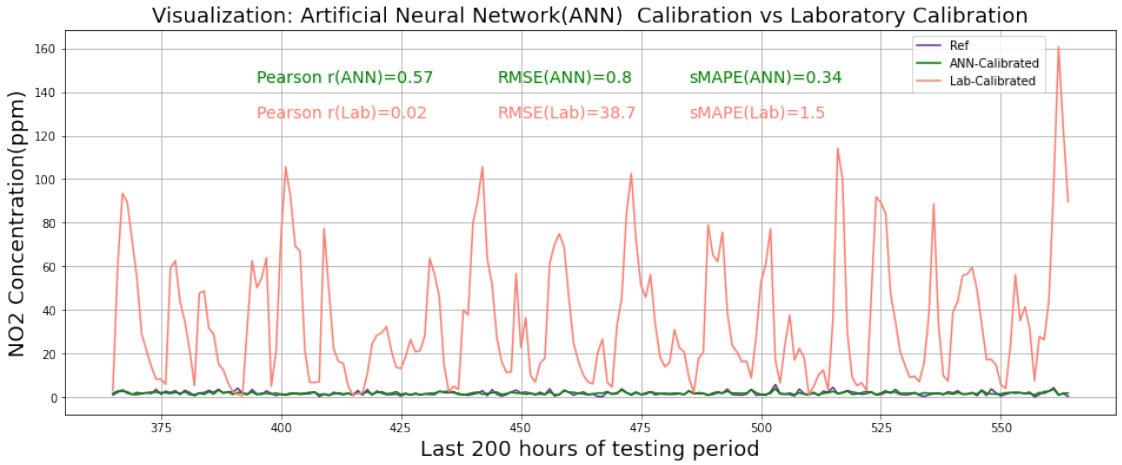
[872]:
```
<tensorflow.python.keras.callbacks.History at 0x24cd92e20>
```

[873]:
```
train_pred = model.predict(X_train_scaled)
test_pred = model.predict(X_test_scaled)
pred=[]
for i in range(len(test_pred)):
    pred.append(sum(list(test_pred[i])))
len(y_test)
```

[873]:
```
565
```

[874]:
```
lab1=SO2_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_ann_SO2=sMAPE_lr
RMSE_ann_SO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_ann_SO2=Pearson_lr
```

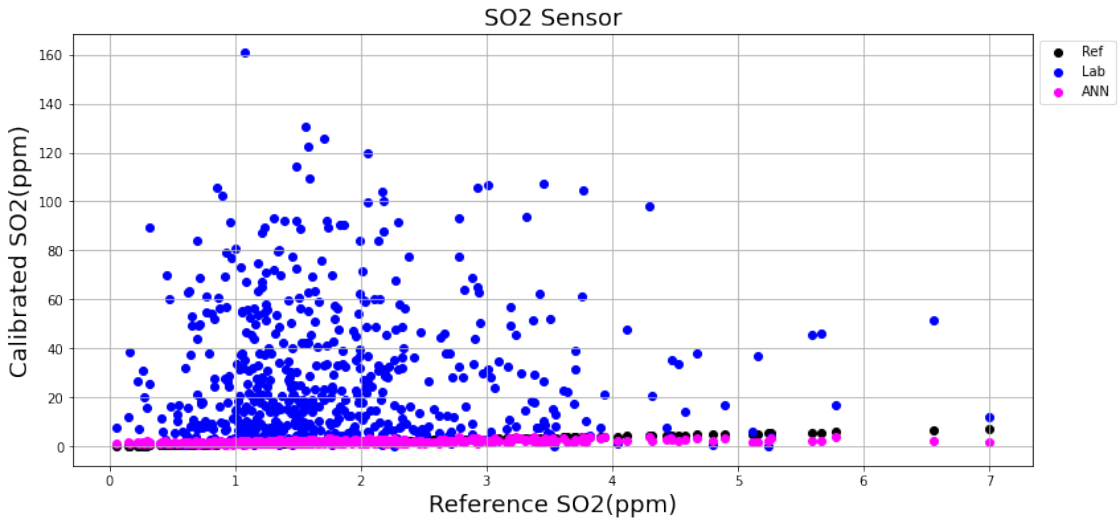
[875]:
```
A=len(y_test)-200
D=max(lab1[A:])-0.2*max(lab1[A:])
C=max(lab1[A:])-0.1*max(lab1[A:])
B=A+120

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index[A:],y_test[A:], color='rebeccapurple')
plt.plot(index[A:],pred[A:], color='green')
plt.plot(index[A:],lab1[A:], color='salmon')
plt.legend(['Ref', 'ANN-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('NO2 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(ANN)='+str(sMAPE_lr), fontsize = 14, color='green')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(ANN)='+str(RMSE_lr), fontsize = 14, color='green')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-90, C, 'Pearson r(ANN)='+str(Pearson_lr), fontsize = 14,␣
 ↪color='green')
plt.text(B-90, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Artificial Neural Network(ANN)  Calibration vs␣
 ↪Laboratory Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[876]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
```

```
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference SO2(ppm)',fontsize=18)
plt.ylabel('Calibrated SO2(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'ANN'], loc = 2, bbox_to_anchor = (1,1))
plt.title('SO2 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.6
Mean squared error(MSE) = 0.7
Median absolute error = 0.47
Explain variance score = 0.32
R2 score = 0.32
```



5 O3 CALIBRATION

[877]:
```
import pandas as pd
import scipy.io
import numpy as np
mat= scipy.io.loadmat('O3.mat')
L_O3final=[]
for i in range(len(mat['O3final_1'])):
    L_O3final.append(list(mat['O3final_1'][i]))

O3_data=pd.DataFrame(L_O3final,columns=['Lab1',␣
 ↪'Lab2','Lab3','Lab4','Temp','RH','Ref','Time'])
Time=O3_data['Time'].to_list()
Time=np.array(Time)
Date=pd.to_datetime(Time-719529,unit='d').round('h')
O3_data['Date'] = Date.tolist()
O3_data=O3_data.set_index('Date')
O3_data.drop('Time',axis = 1, inplace = True)

mat = scipy.io.loadmat('O3_raw.mat')

L_O3_raw=[]
for i in range(len(mat['O3_raw'])):
    L_O3_raw.append(list(mat['O3_raw'][i]))

O3_raw=pd.DataFrame(L_O3_raw,columns=['WE', 'AE','Temp','RH','Time'])
O3_raw.head()
O3_data.insert(loc = 0,
          column = 'WE',
          value = O3_raw['WE'].to_list())

O3_data.insert(loc = 1,
          column = 'AE',
          value = O3_raw['AE'].to_list())


O3_data=O3_data.interpolate()
WE=np.array(O3_data['WE'].to_list())
AE=np.array(O3_data['AE'].to_list())
Signal=list(WE-AE)
O3_data.insert(loc = 2,
          column = 'Signal',
          value = Signal)
O3_data.head()
```

[877]:
```
                              WE          AE     Signal         Lab1  \
Date
2019-10-02 12:00:00  229.020000  232.625625  -3.605625   621.625704
2019-10-02 13:00:00  284.284762  207.033333  77.251429  1835.510709
2019-10-02 16:00:00  199.551940  131.729254  67.822687  2195.698656
2019-10-02 17:00:00  234.016418  225.462090   8.554328   386.251890
2019-10-03 16:00:00  234.671429  229.311429   5.360000   200.100352

                             Lab2        Lab3         Lab4       Temp  \
Date
2019-10-02 12:00:00   67.962148  -12.822516  -987.063805  26.378438
2019-10-02 13:00:00  293.683054  159.572059 -1331.518231  25.502791
2019-10-02 16:00:00  615.567710  379.302767  -893.337914  30.827910
2019-10-02 17:00:00  108.049071   34.461730  -696.227364  30.047164
2019-10-03 16:00:00  122.550133   72.885162  -587.216200  29.441429

                            RH        Ref
Date
2019-10-02 12:00:00  58.063437  52.449447
2019-10-02 13:00:00  59.868837  50.464425
2019-10-02 16:00:00  49.008060  37.972231
2019-10-02 17:00:00  51.259851  33.446343
2019-10-03 16:00:00  52.018571  33.071706
```

[878]:
```
O3_Data=O3_data[['Signal','Lab1','Temp','RH','Ref']]
O3_Data=O3_Data[(O3_Data[O3_Data.columns] >= 0).all(axis=1)]
#CO_data=CO_data.resample('D').mean()
O3_Data=O3_Data.dropna()
O3_Data.shape
```

[878]:
```
(741, 5)
```

5.1 Model 1: LR

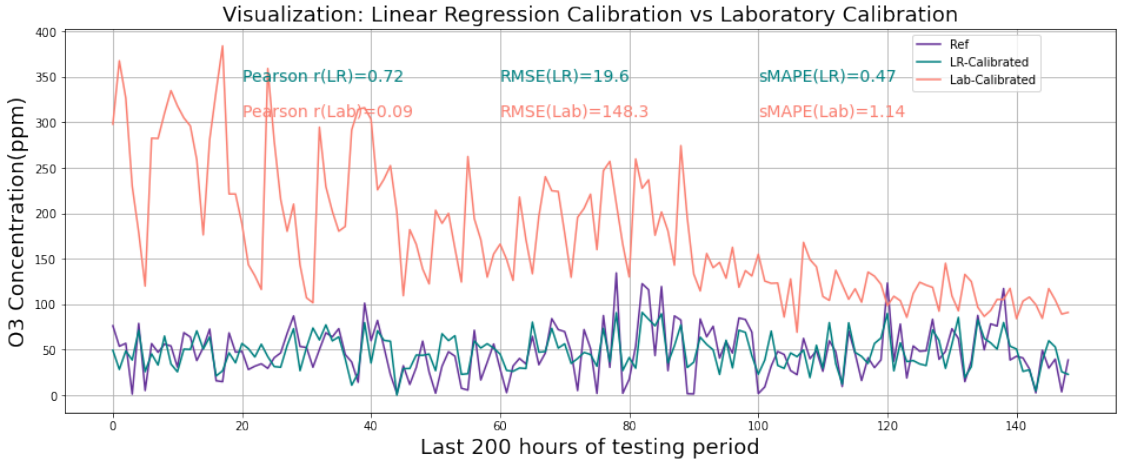
[879]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error as mae
import sklearn.metrics as sm
import matplotlib.pyplot as plt
X=O3_Data[['Signal','Temp','RH']]
y=O3_Data['Ref']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
len(X_test)
```

[879]:
```
149
```

[880]:
```
lr = LinearRegression()
model = lr.fit(X_train, y_train)
pred = model.predict(X_test)
lab1=O3_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_lr_O3=sMAPE_lr
RMSE_lr_O3=RMSE_lr/np.mean(np.array(y_test))
Pearson_lr_O3=Pearson_lr
sMAPE_lab_O3=sMAPE_lab
RMSE_lab_O3=RMSE_lab/np.mean(np.array(lab1))
Pearson_lab_O3=Pearson_lab
```

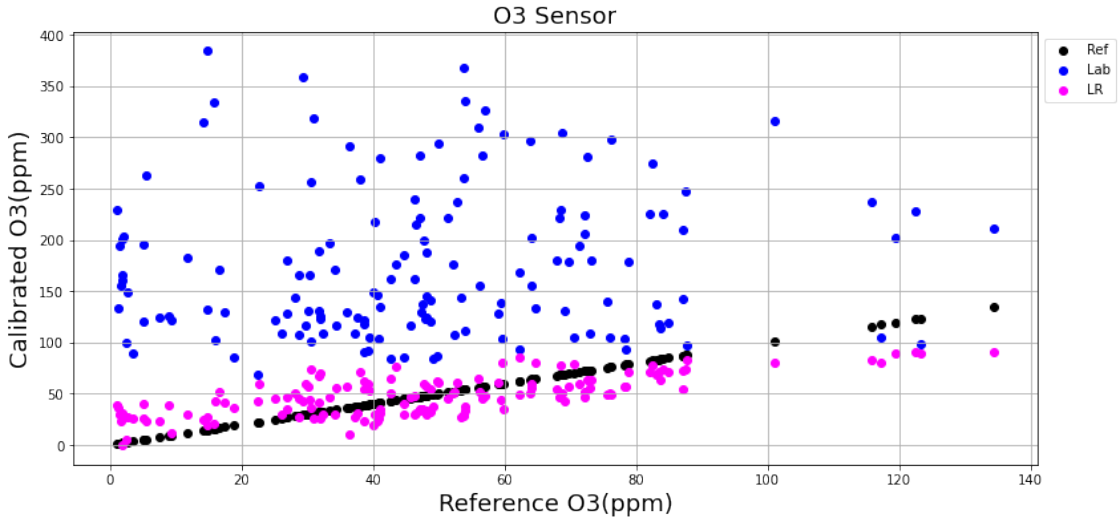
[881]:
```
A=len(y_test)
D=max(lab1)-0.2*max(lab1)
C=max(lab1)-0.1*max(lab1)
B=100

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index,y_test, color='rebeccapurple')
plt.plot(index,pred, color='teal')
plt.plot(index,lab1, color='salmon')
plt.legend(['Ref', 'LR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('O3 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(LR)='+str(sMAPE_lr), fontsize = 14, color='teal')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(LR)='+str(RMSE_lr), fontsize = 14, color='teal')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-80, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='teal')
plt.text(B-80, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Linear Regression Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
```

```
plt.show()
```



[882]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference O3(ppm)',fontsize=18)
plt.ylabel('Calibrated O3(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'LR'], loc = 2, bbox_to_anchor = (1,1))
plt.title('O3 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 16.91
Mean squared error(MSE) = 383.07
Median absolute error = 15.33
Explain variance score = 0.52
R2 score = 0.52
```



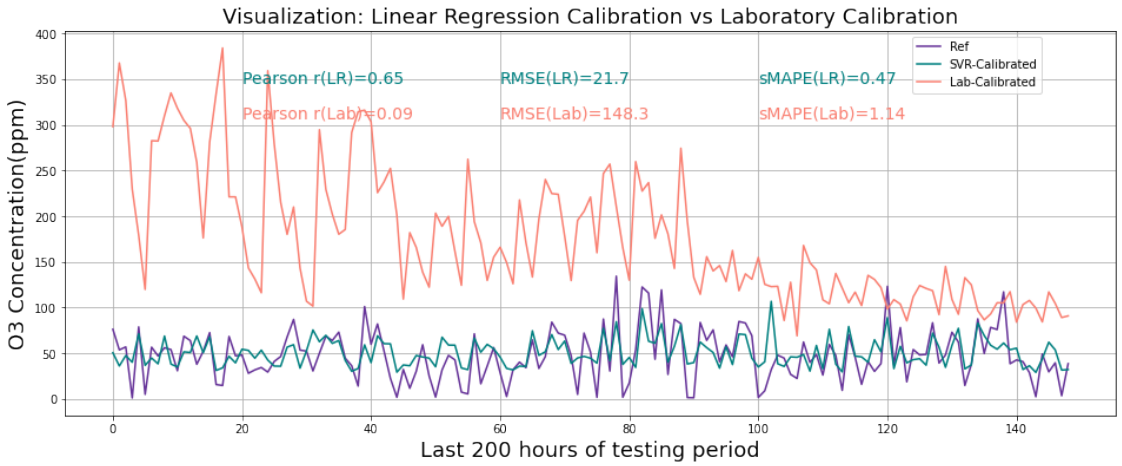
5.2 Model 2: SVR

[883]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'poly', degree=4)
regressor.fit(X_train, y_train)
pred = regressor.predict(X_test)
```

[884]:
```
lab1=O3_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_svr_O3=sMAPE_lr
RMSE_svr_O3=RMSE_lr/np.mean(np.array(y_test))
Pearson_svr_O3=Pearson_lr
```

[885]:
```
A=len(y_test)
D=max(lab1)-0.2*max(lab1)
C=max(lab1)-0.1*max(lab1)
```

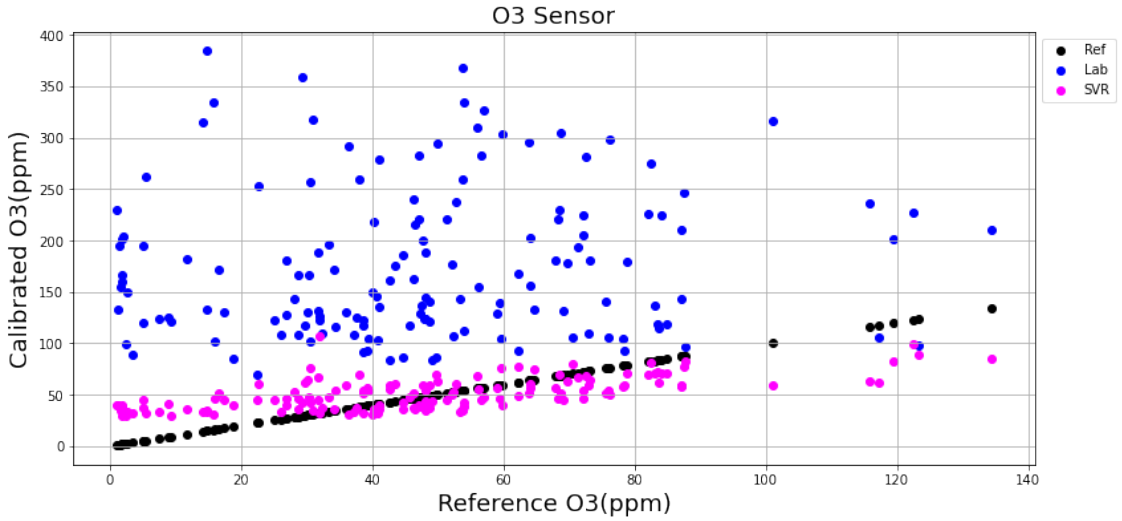
```
B=100

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index,y_test, color='rebeccapurple')
plt.plot(index,pred, color='teal')
plt.plot(index,lab1, color='salmon')
plt.legend(['Ref', 'SVR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
↪= (0.8,1))
plt.ylabel('O3 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(LR)='+str(sMAPE_lr), fontsize = 14, color='teal')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(LR)='+str(RMSE_lr), fontsize = 14, color='teal')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-80, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='teal')
plt.text(B-80, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Linear Regression Calibration vs Laboratory␣
↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[886]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
```

```
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference O3(ppm)',fontsize=18)
plt.ylabel('Calibrated O3(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'SVR'], loc = 2, bbox_to_anchor = (1,1))
plt.title('O3 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 17.61
Mean squared error(MSE) = 471.55
Median absolute error = 14.31
Explain variance score = 0.42
R2 score = 0.4
```



5.3 Model 3 : Random Forest

[887]:
```
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 500, random_state = 0)
```

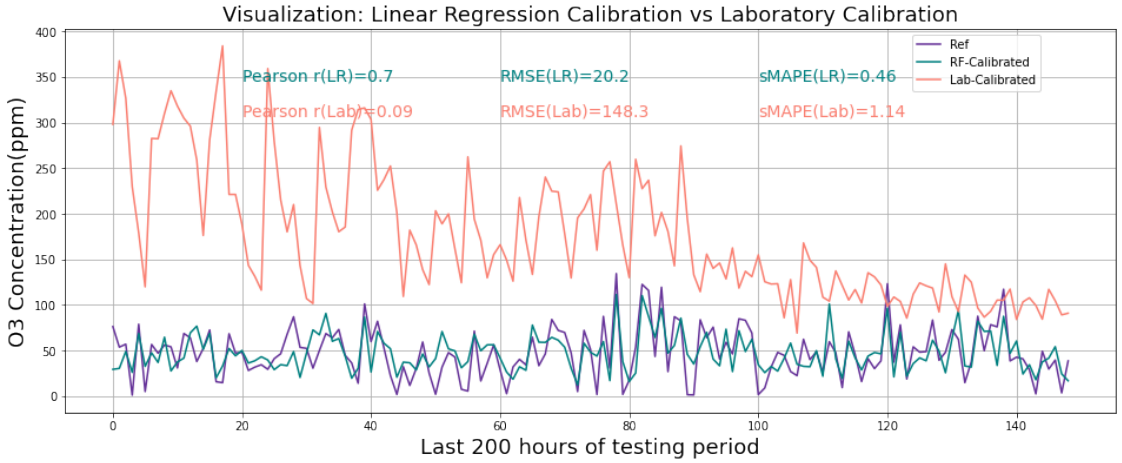
```
# fit the regressor with x and y data
regressor.fit(X_train, y_train)
```

[887]:
```
RandomForestRegressor(n_estimators=500, random_state=0)
```

[888]:
```
pred = regressor.predict(X_test)
lab1=O3_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_rf_O3=sMAPE_lr
RMSE_rf_O3=RMSE_lr/np.mean(np.array(y_test))
Pearson_rf_O3=Pearson_lr
```

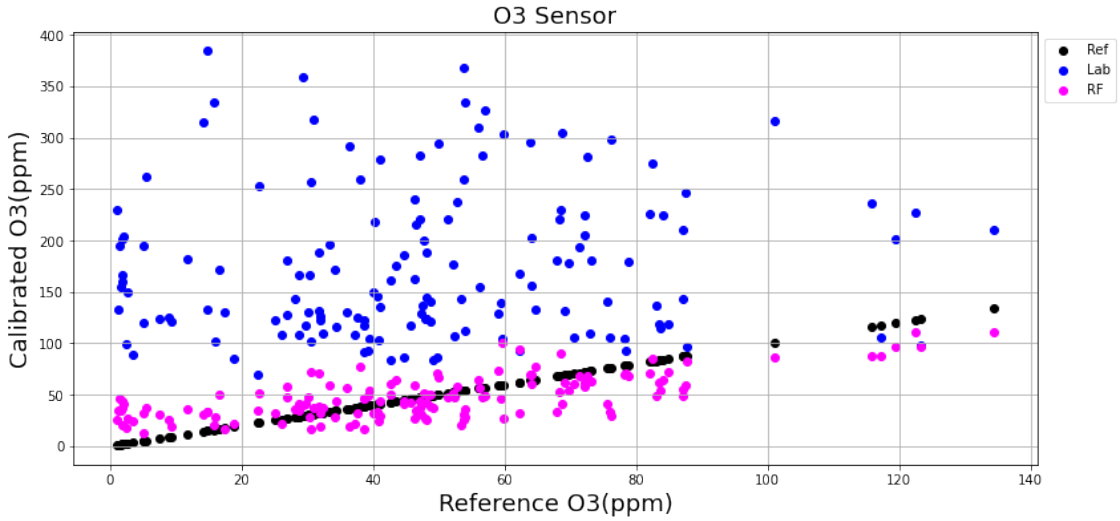
[889]:
```
A=len(y_test)
D=max(lab1)-0.2*max(lab1)
C=max(lab1)-0.1*max(lab1)
B=100

fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index,y_test, color='rebeccapurple')
plt.plot(index,pred, color='teal')
plt.plot(index,lab1, color='salmon')
plt.legend(['Ref', 'RF-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('O3 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(LR)='+str(sMAPE_lr), fontsize = 14, color='teal')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(LR)='+str(RMSE_lr), fontsize = 14, color='teal')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-80, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='teal')
plt.text(B-80, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Linear Regression Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
```

```
plt.show()
```



[890]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference O3(ppm)',fontsize=18)
plt.ylabel('Calibrated O3(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'RF'], loc = 2, bbox_to_anchor = (1,1))
plt.title('O3 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 16.62
Mean squared error(MSE) = 408.35
Median absolute error = 13.57
Explain variance score = 0.48
R2 score = 0.48
```



5.4 Model 4: ANN

[891]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from sklearn.preprocessing import StandardScaler
model = Sequential()
model.add(Dense(3, input_shape = (3,),kernel_initializer='normal', activation=␣
↪'linear'))
model.add(Dense(128,kernel_initializer='normal', activation= 'relu'))
model.add(Dense(50, kernel_initializer='normal',activation= 'relu'))
#model.add(Dense(100, kernel_initializer='normal',activation= 'relu'))
model.add(Dense(1,kernel_initializer='normal',activation='linear',))
sgd = optimizers.Adam(learning_rate=0.01)

model.compile(optimizer = sgd, loss = 'mean_squared_error', metrics= ['mse',␣
↪'mae'])
model.summary()
```

```
Model: "sequential_33"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_132 (Dense)            (None, 3)                 12
_________________________________________________________________
dense_133 (Dense)            (None, 128)               512
_________________________________________________________________
dense_134 (Dense)            (None, 50)                6450
```

```
_________________________________________________________________
dense_135 (Dense)            (None, 1)                 51
=================================================================
Total params: 7,025
Trainable params: 7,025
Non-trainable params: 0
_________________________________________________________________
```

[892]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled=scaler.transform(X_train)
X_test_scaled=scaler.transform(X_test)
model.fit(X_train_scaled, y_train, batch_size= 100, epochs=200, verbose= 0)
```

[892]:
```
<tensorflow.python.keras.callbacks.History at 0x25ad23f10>
```

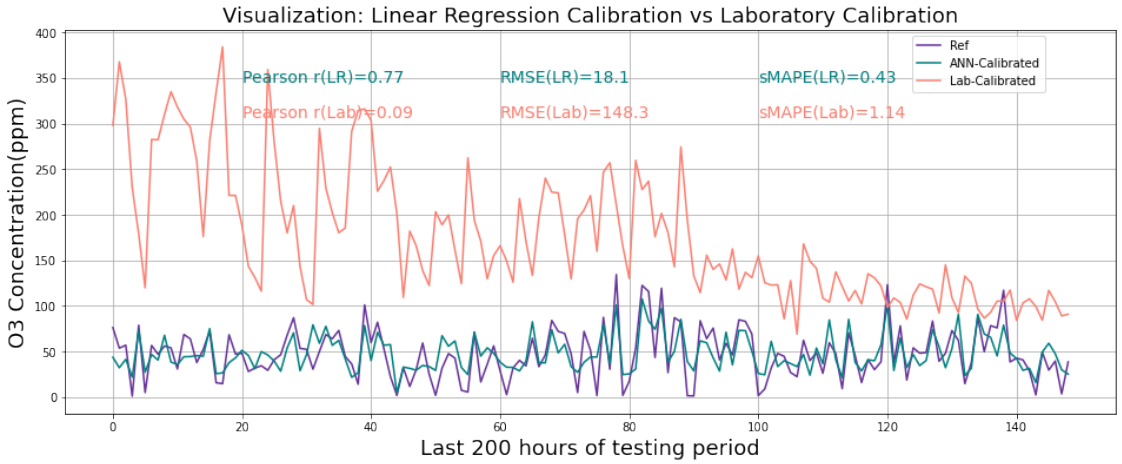
[893]:
```
train_pred = model.predict(X_train_scaled)
test_pred = model.predict(X_test_scaled)
pred=[]
for i in range(len(test_pred)):
    pred.append(sum(list(test_pred[i])))
len(y_test)
```

[893]:
```
149
```

[894]:
```
lab1=O3_Data['Lab1'].to_list()[len(y_train):]
index=[i for i in range(len(y_test))]
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =index)
Y_test
Pred=pd.Series(pred,index =index)
Lab1=pd.Series(lab1,index =index)
sMAPE_lr=round(smape_loss(Y_test,Pred),2)
sMAPE_lab=round (smape_loss(Y_test,Lab1),2)
RMSE_lr=round(np.sqrt(sm.mean_squared_error(y_test, pred)),1)
RMSE_lab=round(np.sqrt(sm.mean_squared_error(y_test, lab1)),1)
Pearson_lr=round(np.corrcoef(y_test, pred)[0, 1],2)
Pearson_lab=round(np.corrcoef(y_test, lab1)[0, 1],2)
sMAPE_ann_O3=sMAPE_lr
RMSE_ann_O3=RMSE_lr/np.mean(np.array(y_test))
Pearson_ann_O3=Pearson_lr
```

[895]:
```
A=len(y_test)
D=max(lab1)-0.2*max(lab1)
C=max(lab1)-0.1*max(lab1)
B=100
```

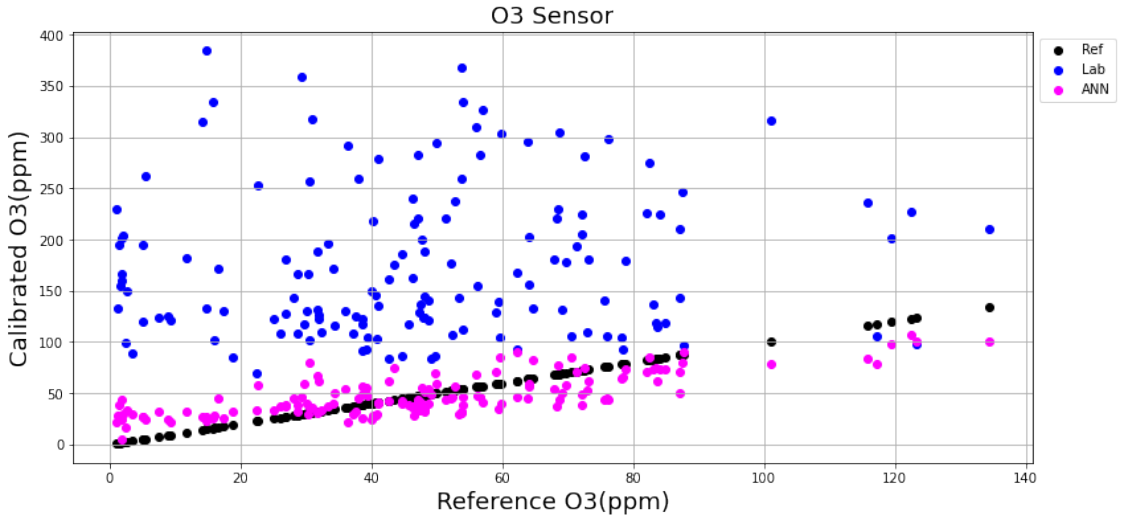
```
fig= plt.figure(figsize=(16,6))
index=[i for i in range(len(y_test))]
plt.plot(index,y_test, color='rebeccapurple')
plt.plot(index,pred, color='teal')
plt.plot(index,lab1, color='salmon')
plt.legend(['Ref', 'ANN-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor␣
 ↪= (0.8,1))
plt.ylabel('O3 Concentration(ppm)',fontsize=18)
plt.text(B, C, 'sMAPE(LR)='+str(sMAPE_lr), fontsize = 14, color='teal')
plt.text(B, D, 'sMAPE(Lab)='+str(sMAPE_lab), fontsize = 14, color='salmon')
plt.text(B-40, C, 'RMSE(LR)='+str(RMSE_lr), fontsize = 14, color='teal')
plt.text(B-40, D, 'RMSE(Lab)='+str(RMSE_lab), fontsize = 14, color='salmon')
plt.text(B-80, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='teal')
plt.text(B-80, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14,␣
 ↪color='salmon')
plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.title('Visualization: Linear Regression Calibration vs Laboratory␣
 ↪Calibration',fontsize=18)
plt.grid(True)
plt.show()
```



[896]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
 ↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
 ↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪pred), 2))
```

```
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_lr=pred
fig= plt.figure(figsize=(13,6))
plt.scatter(y_test,y_test, c ="black")
plt.scatter(y_test,lab1, c ="blue")
plt.scatter(y_test,pred_lr, c ="magenta")
plt.xlabel('Reference O3(ppm)',fontsize=18)
plt.ylabel('Calibrated O3(ppm)',fontsize=18)
plt.legend(['Ref', 'Lab', 'ANN'], loc = 2, bbox_to_anchor = (1,1))
plt.title('O3 Sensor',fontsize=18 )
plt.grid(True)
```

```
Regressor model performance:
Mean absolute error(MAE) = 15.1
Mean squared error(MSE) = 328.28
Median absolute error = 12.82
Explain variance score = 0.59
R2 score = 0.59
```



6 Data Analytics

[897]:
```
import plotly.graph_objects as go
import numpy as np

LAB_PR=[Pearson_lab_CO,Pearson_lab_NO2,Pearson_lab_SO2,Pearson_lab_O3]
LR_PR=[Pearson_lr_CO,Pearson_lr_NO2,Pearson_lr_SO2,Pearson_lr_O3]
SVR_PR=[Pearson_svr_CO,Pearson_svr_NO2,Pearson_svr_SO2,Pearson_svr_O3]
```

```
RF_PR=[Pearson_rf_CO,Pearson_rf_NO2,Pearson_rf_SO2,Pearson_rf_O3]
ANN_PR=[Pearson_ann_CO,Pearson_ann_NO2,Pearson_ann_SO2,Pearson_ann_O3]


LAB_SM=[sMAPE_lab_CO,sMAPE_lab_NO2,sMAPE_lab_SO2,sMAPE_lab_O3]
LR_SM=[sMAPE_lr_CO,sMAPE_lr_NO2,sMAPE_lr_SO2,sMAPE_lr_O3]
SVR_SM=[sMAPE_svr_CO,sMAPE_svr_NO2,sMAPE_svr_SO2,sMAPE_svr_O3]
RF_SM=[sMAPE_rf_CO,sMAPE_rf_NO2,sMAPE_rf_SO2,sMAPE_rf_O3]
ANN_SM=[sMAPE_ann_CO,sMAPE_ann_NO2,sMAPE_ann_SO2,sMAPE_ann_O3]

LAB_RM=[RMSE_lab_CO,RMSE_lab_NO2,RMSE_lab_SO2,RMSE_lab_O3]
LR_RM=[RMSE_lr_CO,RMSE_lr_NO2,RMSE_lr_SO2,RMSE_lr_O3]
SVR_RM=[RMSE_svr_CO,RMSE_svr_NO2,RMSE_svr_SO2,RMSE_svr_O3]
RF_RM=[RMSE_rf_CO,RMSE_rf_NO2,RMSE_rf_SO2,RMSE_rf_O3]
ANN_RM=[RMSE_ann_CO,RMSE_ann_NO2,RMSE_ann_SO2,RMSE_ann_O3]

PR=LAB_PR+LR_PR+SVR_PR+RF_PR+ANN_PR
SM=LAB_SM+LR_SM+SVR_SM+RF_SM+ANN_SM
RM=LAB_RM+LR_RM+SVR_RM+RF_RM+ANN_RM
x1=['LAB' for i in range(4)]
x2=['LR' for i in range(4)]
x3=['SVR' for i in range(4)]
x4=['RF' for i in range(4)]
x5=['ANN' for i in range(4)]

x=x1+x2+x3+x4+x5


fig = go.Figure()
# Defining x axis
x = x

fig.add_trace(go.Box(

    # defining y axis in corresponding
    # to x-axis
    y=PR,
    x=x,
    name='Pearson r',
    marker_color='teal'
))

fig.add_trace(go.Box(
    y=SM,
    x=x,
    name='sMAPE',
    marker_color='salmon'
```

```
))
fig.add_trace(go.Box(
    y=RM,
    x=x,
    name='NRMSE',
    marker_color='darkgoldenrod'

))


fig.update_layout(

    # group together boxes of the different
    # traces for each value of x
    boxmode='group'
)

fig.update_xaxes(title_text="Calibration Model")
#fig.update_yaxes(title_text="Pearson r")
fig.show()
```

[ ]: