Sensor Calibration

November 20, 2021

1 DATA

2 CO CALIBRATION

[131]:
```
import pandas as pd
Ref=pd.read_csv('Ref.csv')
Ref["CO"] = 1000 * Ref["CO"]
Ref['Date'] = pd.to_datetime(Ref['Date_Time'])
Ref=Ref.set_index('Date')
Ref.drop('Date_Time',axis = 1, inplace = True)
Ref=Ref.resample('5min').mean()
Ref=Ref[76463:137376]
Ref_CO=Ref['CO'].to_list()
Ref_NO2=Ref['NO2'].to_list()
Ref_SO2=Ref['SO2'].to_list()
Ref_O3=Ref['O3'].to_list()
```

[132]:
```
import random
import pandas as pd
import scipy.io
import numpy as np
data = pd.read_csv('CO.txt', header = None,low_memory=False)
data.columns=['WE','AE','Temp','RH','Time']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_CO=data
Data_CO['Ref']=Ref_CO
WE=Data_CO['WE'].to_list()
AE=Data_CO['AE'].to_list()
```

```
signal=np.array(WE)-np.array(AE)
Data_CO['Net Signal']=signal
Data_CO['Month']=Data_CO.index.month
Data_CO['Day_of_week']=Data_CO.index.dayofweek
Data_CO['Day']=Data_CO.index.day
Data_CO['Hour']=Data_CO.index.hour
CO_Data=Data_CO
CO_Data=CO_Data[(CO_Data[CO_Data.columns] >= 0).all(axis=1)]
CO_Data=CO_Data.dropna()
data = pd.read_csv('Conc_CO.txt', header = None,low_memory=False)
data.columns=['Lab1','Temp','RH','Time','Ref']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_CO=data
signal=np.array(WE)-np.array(AE)
Data_CO['Net Signal']=signal
Data_CO['Month']=Data_CO.index.month
Data_CO['Day_of_week']=Data_CO.index.dayofweek
Data_CO['Day']=Data_CO.index.day
Data_CO['Hour']=Data_CO.index.hour
CO_Data=Data_CO
CO_Data=CO_Data[(CO_Data[CO_Data.columns] >= 0).all(axis=1)]
CO_Data=CO_Data.dropna()
CO_Data=CO_Data.sample(frac=1)
CO_Data=CO_Data.resample('h').mean()
CO_Data=CO_Data.dropna()
CO_Data.head()
```

[132]:

```
                            Lab1       Temp         RH         Ref  \
Date
2019-10-02 11:00:00  3571.592599  26.378438  58.063437  312.707200
2019-10-02 12:00:00  2861.791016  25.721989  51.159852  228.718975
2019-10-02 15:00:00  3313.026561  30.623188  49.580620  259.460975
2019-10-03 15:00:00   535.086842  29.421250  52.411845  341.897275
2019-10-03 16:00:00   592.411938  29.211333  53.102667  261.288900

                     Net Signal  Month  Day_of_week  Day  Hour
Date
2019-10-02 11:00:00  984.426875   10.0          2.0  2.0  11.0
```

```
2019-10-02 12:00:00  823.564115   10.0          2.0  2.0  12.0
2019-10-02 15:00:00  914.638179   10.0          2.0  2.0  15.0
2019-10-03 15:00:00  152.440810   10.0          3.0  3.0  15.0
2019-10-03 16:00:00  137.737333   10.0          3.0  3.0  16.0
```

[133]:
```
CO_Data=CO_Data.resample('h').mean()
CO_Data=CO_Data.dropna()
```

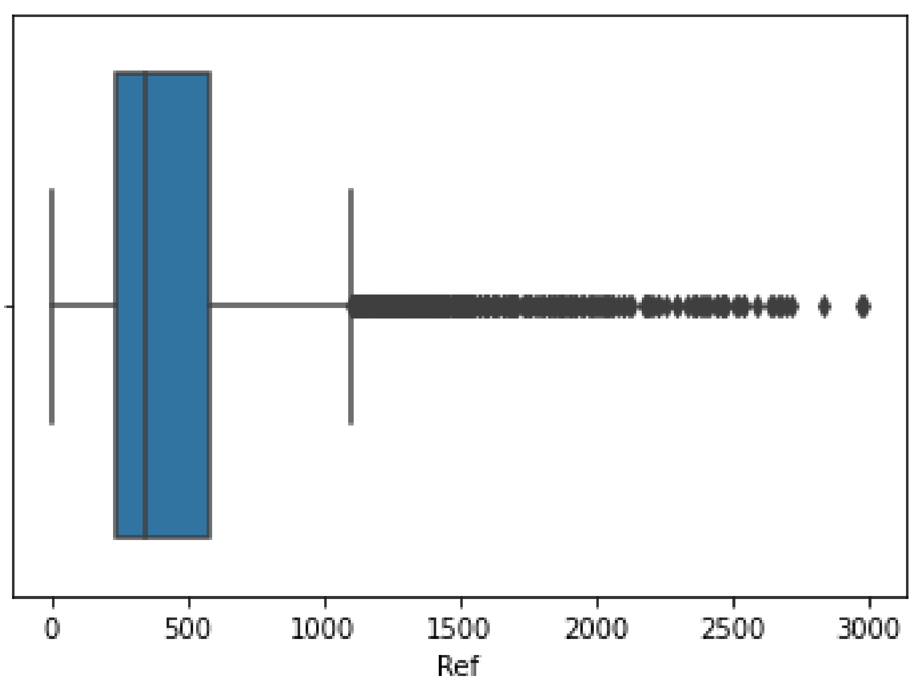
[134]:
```
#Ref=CO_Data['Ref'].to_list()
#CO_Data=CO_Data[CO_Data.Ref.between(np.mean(Ref)-0.7*np.std(Ref), np.
 ↪mean(Ref)+0.7*np.std(Ref))]
#CO_Data.shape
```

3 Outlier detection and removal

[135]:
```
import numpy as np

import pandas as pd
import seaborn as sns
from scipy import stats
```

[136]:
```
sns.boxplot(x=CO_Data['Ref'])
z=np.abs(stats.zscore(CO_Data))
CO_data=CO_Data[(z < 3).all(axis=1)]
CO_data.shape,CO_Data.shape
```

[136]: ((3787, 9), (3931, 9))



[137]:
```python
def MBE(true,pred):
    true=np.array(true)
    pred=np.array(pred)
    mbe=np.mean(true-pred)
    return mbe
def CRMSE(true,pred):
    true=np.array(true)
    pred=np.array(pred)
    crmse=np.sqrt(np.mean(((true-np.mean(true))-(pred-np.mean(pred)))**2))
    if np.std(pred)>np.std(true):
        crmse=crmse
    else:
        crmse=-crmse
    return crmse
```

[138]:
```python
#Ref=CO_Data['Ref'].to_list()
#CO_Data=CO_Data[CO_Data.Ref.between(np.mean(Ref)-0.3*np.std(Ref), np.
 ↪mean(Ref)+0.3*np.std(Ref))]
#NO2_Data.shape
```

3.1 Model 1: Linear Regression

[139]:
```python
from sktime.performance_metrics.forecasting import sMAPE, smape_loss
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
#'Ref_NO2','Ref_SO2','Ref_O3',
#,'Month','Day_of_week','Day','Hour'

X=CO_Data[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Hour']]
y=CO_Data['Ref']
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size = 0.2)
#train_test_split(X, y, test_size = 0.2)
```

[140]:
```python
lr = LinearRegression()
model = lr.fit(X_train.drop(['Lab1'], axis=1), y_train)
pred = model.predict(X_test.drop(['Lab1'], axis=1))
lab1=X_test['Lab1'].to_list()

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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_lr_CO=Pearson_lr
sMAPE_lab_CO=sMAPE_lab
RMSE_lab_CO=round(RMSE_lab/np.mean(np.array(lab1)),2)
Pearson_lab_CO=Pearson_lab
R2_lr_CO=round(sm.r2_score(y_test, pred), 2)
R2_lab_CO=round(sm.r2_score(y_test, lab1), 2)
RMSE_Lr_CO=RMSE_lr
RMSE_Lab_CO=RMSE_lab

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=120
Pearson_lr,RMSE_Lr_CO
```

[140]: (0.94, 145.9)

fig= plt.figure(figsize=(8,6)) index=[i for i in range(1,201)] ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.2) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='#513e00',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'LR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor = (0.74,1)) plt.ylabel('CO Concentration(ppb)',fontsize=18) #plt.text(B-20, C, r'$R^2(LR)$ ='+str(R2_lr_CO), fontsize = 14, color='#513e00') #plt.text(B-20, D, r'$R^2(Lab)$ =' +str(R2_lab_CO), fontsize = 14, color='#426eff') #plt.text(B-70, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='#513e00') #plt.text(B-70, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Visualization: Linear Regression Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[141]:
```
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
MBE_LR_CO=MBE(pred,y_test)/np.std(y_test)
CRMSE_LR_CO=CRMSE(y_test,pred)/np.std(y_test)
MBE_LAB_CO=MBE(lab1,y_test)/np.std(y_test)
CRMSE_LAB_CO=CRMSE(y_test,lab1)/np.std(y_test)
```

```
Regressor model performance:
Mean absolute error(MAE) = 84.16
Mean squared error(MSE) = 21294.64
Median absolute error = 48.17
Explain variance score = 0.89
R2 score = 0.89
```

3.2 Model 2 : Support Vector Regression (SVR)

[142]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'linear')
regressor.fit(X_train.drop(['Lab1'], axis=1), y_train)
pred = regressor.predict(X_test.drop(['Lab1'], axis=1))
for i in range(len(Pred)):
    if pred[i]<0:
        pred[i]=np.mean(np.array(pred))
pred_svr=pred
```

[143]:
```
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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_svr_CO=Pearson_lr
R2_svr_CO=round(sm.r2_score(y_test, pred), 2)
```

```
RMSE_Svr_CO=RMSE_lr
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='brown',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'SVR-Calibrated', 'Lab-Calibrated'], loc = 2, bbox_to_anchor = (0.74,1)) plt.ylabel('CO Concentration(ppb)',fontsize=18) #plt.text(B-20, C,r'$R^2(SVR)$ ='+str(R2_svr_CO) , fontsize = 14, color='brown') #plt.text(B-20, D,r'$R^2(Lab)$ ='+str(R2_lab_CO) , fontsize = 14, color='#426eff') #plt.text(B-70, C, 'Pearson r(SVR)='+str(Pearson_lr), fontsize = 14, color='brown') #plt.text(B-70, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #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(linestyle='-.',linewidth=0.3) plt.show()

[144]:
```
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
MBE_SVR_CO=MBE(pred,y_test)/np.std(y_test)
CRMSE_SVR_CO=CRMSE(y_test,pred)/np.std(y_test)
```

```
Regressor model performance:
Mean absolute error(MAE) = 83.0
Mean squared error(MSE) = 20920.99
Median absolute error = 46.84
Explain variance score = 0.89
R2 score = 0.89
```

3.3 Model 3 : Random Forest

[145]:
```
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 500,min_samples_split=␣
↪2,min_samples_leaf= 1,max_features= 'sqrt',
                                  random_state =␣
↪0,max_depth=None,bootstrap=False)



# fit the regressor with x and y data
regressor=regressor.fit(X_train.drop(['Lab1'], axis=1), y_train)
```

[146]:
```
features_CO=regressor.feature_importances_
pred = regressor.predict(X_test.drop(['Lab1'], axis=1))
pred_rf_co=pred
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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_rf_CO=Pearson_lr
R2_rf_CO=round(sm.r2_score(y_test, pred), 2)
RMSE_Rf_CO=RMSE_lr
```

[147]:
```
features_CO
```

[147]:
```
array([0.71385754, 0.07158194, 0.04860557, 0.05123193, 0.02341009,
       0.09131295])
```

fig= plt.figure(figsize=(30,4))

ax = fig.add_subplot(111) #ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='black',linewidth=3) plt.plot(index,pred[A:], color='red',linewidth=3) plt.plot(index,lab1[A:], color='blue',linewidth=3) plt.legend(['Ref', 'RF-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.79,1)) plt.ylabel('CO Concentration(ppb)',fontsize=18) plt.text(B-20, C, r'$R^2(RF)$ ='+str(R2_rf_CO), fontsize = 14, color='red') plt.text(B-20, D, r'$R^2(Lab)$ ='+str(R2_lab_CO), fontsize = 14, color='blue') plt.text(B-72, C, 'Pearson r(RF)='+str(Pearson_lr), fontsize = 14, color='red') plt.text(B-72, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='blue') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory Calibration',fontsize=18) #plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[148]:
```
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))
```

```python
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_rf=pred
MBE_RF_CO=MBE(pred,y_test)/np.std(y_test)
CRMSE_RF_CO=CRMSE(y_test,pred)/np.std(y_test)
```

```
Regressor model performance:
Mean absolute error(MAE) = 66.2
Mean squared error(MSE) = 12279.13
Median absolute error = 36.22
Explain variance score = 0.93
R2 score = 0.93
```

4 Hyper parameter tuning

[149]:
```python
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000],
'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80,
90, 100, 110, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2,
4], 'bootstrap': [True, False]}
```

[150]:
```python
from sklearn.model_selection import RandomizedSearchCV
# Use the random grid to search for best hyperparameters
# First create the base model to tune
#rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
```

```
#rf_random = RandomizedSearchCV(estimator = rf, param_distributions =␣
↪random_grid, n_iter = 100, cv = 3,
#                               verbose=1, random_state=42, n_jobs = -1)
# Fit the random search model
#rf_random.fit(X_train, y_train)
#rf_random.best_params_
#{'n_estimators': 400,
 #'min_samples_split': 2,
 #'min_samples_leaf': 1,
 #'max_features': 'sqrt',
 #'max_depth': None,
 #'bootstrap': False}
```

4.1 Model 5: ANN

[151]:
```
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 = (6,),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(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_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_19 (Dense)             (None, 3)                 21
_________________________________________________________________
dense_20 (Dense)             (None, 128)               512
_________________________________________________________________
dense_21 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_22 (Dense)             (None, 100)               12900
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 101
=================================================================
Total params: 30,046
```

```
Trainable params: 30,046
Non-trainable params: 0
_________________________________________________________________
```

[152]:
```
scaler = StandardScaler()
scaler.fit(X_train.drop(['Lab1'], axis=1))
X_train_scaled=scaler.transform(X_train.drop(['Lab1'], axis=1))
X_test_scaled=scaler.transform(X_test.drop(['Lab1'], axis=1))
hist=model.fit(X_train_scaled, y_train, batch_size= 10, epochs=40, verbose=␣
 ↪0)#,validation_split=0.2
```

[153]:
```
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])))
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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_ann_CO=Pearson_lr
R2_ann_CO=round(sm.r2_score(y_test, pred), 2)
RMSE_Ann_CO=RMSE_lr
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='tomato',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'ANN-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.74,1)) plt.ylabel('CO Concentration(ppb)',fontsize=18) #plt.text(B-200, C, r'$R^2(ANN)$ ='+str(R2_ann_CO), fontsize = 14, color='tomato') #plt.text(B-200, D, r'$R^2(Lab)$ ='+str(R2_lab_CO), fontsize = 14, color='#426eff') #plt.text(B-800, C, 'Pearson r(ANN)='+str(Pearson_lr), fontsize = 14, color='tomato') #plt.text(B-800, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('ANN Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[154]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
 ↪2))
```

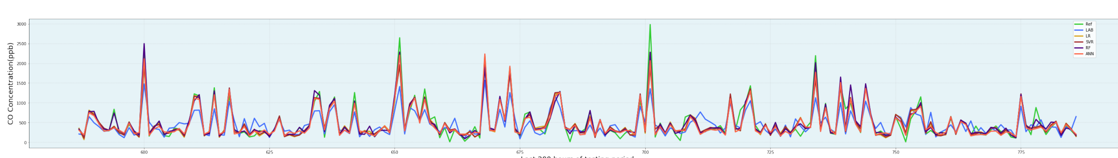
```
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_ann=pred
MBE_ANN_CO=MBE(pred,y_test)/np.std(y_test)
CRMSE_ANN_CO=CRMSE(y_test,pred)/np.std(y_test)
```

```
Regressor model performance:
Mean absolute error(MAE) = 83.06
Mean squared error(MSE) = 18810.76
Median absolute error = 53.41
Explain variance score = 0.9
R2 score = 0.9
```

[155]:
```
len(pred_ann)
```

[155]: 787

[156]:
```
fig= plt.figure(figsize=(50,6))

ax = fig.add_subplot(111)
ax.patch.set_facecolor('lightblue')
ax.patch.set_alpha(0.3)
plt.plot(index[A:],y_test[A:], color='limegreen',linewidth=3)
plt.plot(index[A:],lab1[A:], color='#426eff',linewidth=3)
plt.plot(index[A:],pred_lr[A:], color='goldenrod',linewidth=3)
plt.plot(index[A:],pred_svr[A:], color='brown',linewidth=3)
plt.plot(index[A:],pred_rf[A:], color='indigo',linewidth=3)
plt.plot(index[A:],pred_ann[A:], color='tomato',linewidth=3)

plt.xlabel('Last 200 hours of testing period',fontsize=18)
plt.ylabel('CO Concentration(ppb)',fontsize=18)
plt.legend(['Ref', 'LAB', 'LR','SVR','RF','ANN'], loc = 2, bbox_to_anchor = (0.
↪95,1))
#plt.title('CO Sensor',fontsize=18 )
plt.grid(linestyle='-.',linewidth=0.3)
```



5 Model 6: XGBoost

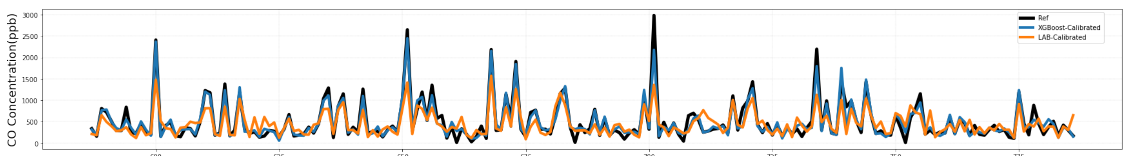
[157]:
```
from xgboost import  XGBRegressor
from numpy import absolute
from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
# create an xgboost regression model
#n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,colsample_bytree=0.
↪4,alpha=10
model = XGBRegressor(n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,
                     colsample_bytree=0.4,alpha=10)
model.fit(X_train.drop(['Lab1'], axis=1), y_train)
```

[157]:
```
XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.4, eta=0.01, gamma=0,
             gpu_id=-1, importance_type='gain', interaction_constraints='',
             learning_rate=0.00999999978, max_delta_step=0, max_depth=5,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=10000, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=10, reg_lambda=1, scale_pos_weight=1, subsample=0.9,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

[158]:
```
pred = model.predict(X_test.drop(['Lab1'], axis=1))
pred_xgb_co=pred
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_xgb_CO=sMAPE_lr
RMSE_xgb_CO=RMSE_lr/np.mean(np.array(y_test))
Pearson_xgb_CO=Pearson_lr
R2_xgb_CO=round(sm.r2_score(y_test, pred), 2)
RMSE_Xgb_CO=RMSE_lr
```

[159]:
```
fig= plt.figure(figsize=(30,4))

ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
```

```
plt.plot(index[A:],y_test[A:],linewidth=5,color='black')
plt.plot(index[A:],pred[A:],linewidth=4)
plt.plot(index[A:],lab1[A:],linewidth=4)
plt.legend(['Ref', 'XGBoost-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.9,1))
plt.ylabel('CO Concentration(ppb)',fontsize=18)
#plt.text(B-200, C,r'$R^{2}(XGB)=$'+str(R2_xgb_CO) , fontsize = 14, color='darkgoldenrod')
#plt.text(B-200, D,r'$R^{2}(Lab)=$'+str(R2_lab_CO), fontsize = 14, color='#426eff')
#plt.text(B-800, C, 'Pearson r(XGB)='+str(Pearson_lr), fontsize = 14, color='darkgoldenrod')
#plt.text(B-800, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff')
#plt.xlabel('Last 200 hours of testing period',fontsize=18)
#plt.title('XGBoost Calibration vs Laboratory Calibration',fontsize=18)
plt.grid(linestyle='-.',linewidth=0.2)
plt.show()
```

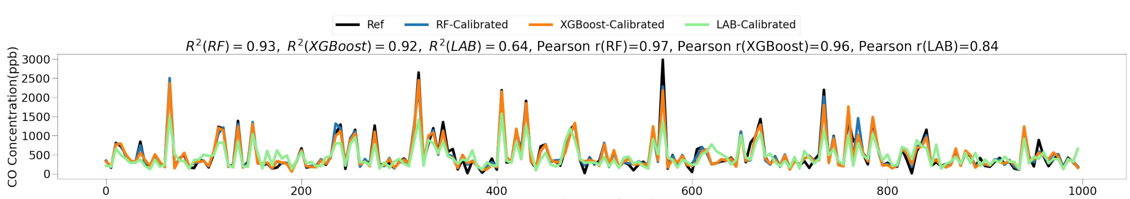


[160]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred), 2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test, pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
pred_xgb=pred
MBE_XGB_CO=MBE(pred,y_test)/np.std(y_test)
CRMSE_XGB_CO=CRMSE(y_test,pred)/np.std(y_test)
```

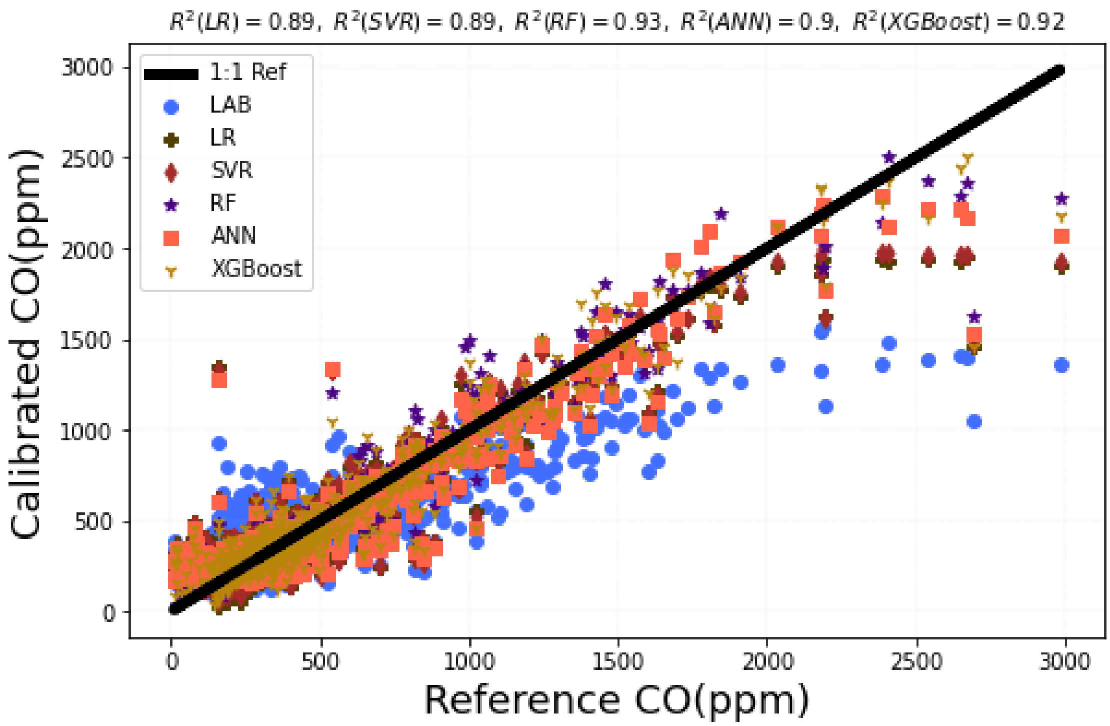
```
Regressor model performance:
Mean absolute error(MAE) = 78.93
Mean squared error(MSE) = 15374.76
Median absolute error = 50.05
Explain variance score = 0.92
R2 score = 0.92
```

[161]:

```
fig= plt.figure(figsize=(50,6))
index=[5*i for i in range(200)]
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
ax.patch.set_alpha(0.3)
plt.plot(index,y_test[A:], color='black',linewidth=7)
plt.plot(index,pred_rf_co[A:],linewidth=7)#, color='#d0587e'
plt.plot(index,pred_xgb_co[A:],linewidth=7)#, color='#009392'
plt.plot(index,lab1[A:],linewidth=7, color='lightgreen')#, color='goldenrod'
plt.legend(['Ref', 'RF-Calibrated','XGBoost-Calibrated', 'LAB-Calibrated'],␣
↪ncol = 4, bbox_to_anchor = (0.7,1.35) ,
          fontsize=28)
plt.ylabel('CO Concentration(ppb)',fontsize=32)
#plt.text(B-20, C, r'$R^{2}(RF)=$'+str(R2_rf_CO), fontsize = 22, color='red')
#plt.text(B-20, D, r'$R^{2}(XGBoost)=$'+str(R2_xgb_CO), fontsize = 22,␣
↪color='green')
#plt.text(B-20, D-0.11*D, r'$R^{2}(Lab)=$'+str(R2_lab_CO), fontsize = 22,␣
↪color='blue')
#plt.text(B-82, C, 'Pearson r(RF)='+str(Pearson_rf_CO), fontsize = 22,␣
↪color='red')
#plt.text(B-82, D, 'Pearson r(XGBoost)='+str(Pearson_xgb_CO), fontsize = 22,␣
↪color='green')
#plt.text(B-82, D-0.11*D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 22,␣
↪color='blue')
#plt.xlabel('Last 200 hours of testing period',fontsize=18)
#plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory␣
↪Calibration',fontsize=18)
#plt.grid(linestyle='-.',linewidth=0.3)
plt.title(r'$R^{2}(RF)=$'+str(R2_rf_CO)+ r'$, \␣
↪R^{2}(XGBoost)=$'+str(R2_xgb_CO)+r'$, \ R^{2}(LAB)=$'+str(R2_lab_CO)
       + ', Pearson r(RF)='+str(Pearson_rf_CO)+', Pearson␣
↪r(XGBoost)='+str(Pearson_xgb_CO)
          +', Pearson r(LAB)='+str(Pearson_lab),
          fontsize=35)
plt.xlabel(' Last 1000 minutes of testing data ',fontsize=35)
#plt.text(B-128, C+0.11*C, '(a)', fontsize =22, color='black')
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.tick_params(width=1,length=15)
plt.show()
```



[162]:
```
fig= plt.figure(figsize=(8,5))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
#m0, b0 = np.polyfit(np.array(y_test), np.array(lab1), 1)
#plt.plot(np.array(y_test), m0*np.array(y_test) +␣
 ↪b0,color='#426eff',linewidth=4)
#m1, b1 = np.polyfit(np.array(y_test), np.array(pred_lr), 1)
#plt.plot(np.array(y_test), m1*np.array(y_test) +␣
 ↪b1,color='#513e00',linewidth=4)
#m2, b2 = np.polyfit(np.array(y_test), np.array(pred_svr), 1)
#plt.plot(np.array(y_test), m2*np.array(y_test) + b2,color='brown',linewidth=4)
#m3, b3 = np.polyfit(np.array(y_test), np.array(pred_rf), 1)
#plt.plot(np.array(y_test), m3*np.array(y_test) + b3,color='indigo',linewidth=4)
#m4, b4 = np.polyfit(np.array(y_test), np.array(pred_ann), 1)
#plt.plot(np.array(y_test), m4*np.array(y_test) + b4,color='tomato',linewidth=4)
#m5, b5 = np.polyfit(np.array(y_test), np.array(pred_xgb), 1)
#plt.plot(np.array(y_test), m5*np.array(y_test) +␣
 ↪b5,color='darkgoldenrod',linewidth=4)
plt.scatter(np.array(y_test),np.array(lab1),color='#426eff' )
plt.scatter(np.array(y_test),np.array(pred_lr),color='#513e00',marker='P')
plt.scatter(np.array(y_test),np.array(pred_svr),color='brown',marker='d')
plt.scatter(np.array(y_test),np.array(pred_rf),color='indigo',marker='*')
plt.scatter(np.array(y_test),np.array(pred_ann),color='tomato',marker='s')
plt.scatter(np.array(y_test),np.
 ↪array(pred_xgb),color='darkgoldenrod',marker='1')
ax.plot(y_test,y_test, c ="black",linewidth=5)
plt.xlabel('Reference CO(ppm)',fontsize=18)
plt.ylabel('Calibrated CO(ppm)',fontsize=18)
plt.legend(['1:1 Ref','LAB','LR','SVR','RF','ANN','XGBoost'
            ], loc = 2, bbox_to_anchor = (0,1))
#plt.title('CO Sensor',fontsize=18 )
plt.title(r'$ R^{2}(LR)=$'+str(R2_lr_CO)+r'$,\ R^{2}(SVR)=$'+str(R2_svr_CO)
          +r'$,\ R^{2}(RF)=$'+str(R2_rf_CO)+ r'$,\ R^{2}(ANN)=$'+str(R2_ann_CO)
          +r'$, \ R^{2}(XGBoost)=$'+str(R2_xgb_CO),
          fontsize=10)
plt.grid(linestyle='-.',linewidth=0.1)
```



6 NO2 Calibration

[163]:
```
import pandas as pd
import scipy.io
import numpy as np
data = pd.read_csv('O3.txt', header = None,low_memory=False)
data.columns=['AE','WE','Temp','RH','Time']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_O3=data
Data_O3['Ref']=Ref_O3
WE=Data_O3['WE'].to_list()
AE=Data_O3['AE'].to_list()
signal=np.array(WE)-np.array(AE)
```

```
Data_03['Net Signal']=signal
Data_03['Month']=Data_03.index.month
Data_03['Day_of_week']=Data_03.index.dayofweek
Data_03['Day']=Data_03.index.day
Data_03['Hour']=Data_03.index.hour
03_Data=Data_03
03_Data=03_Data[(03_Data[03_Data.columns] >= 0).all(axis=1)]
03_Data=03_Data.dropna()
data = pd.read_csv('Conc_03.txt', header = None,low_memory=False)
data.columns=['Lab1','Temp','RH','Time','Ref']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_03=data
signal=np.array(WE)-np.array(AE)
Data_03['Net Signal']=signal
Data_03['Month']=Data_03.index.month
Data_03['Day_of_week']=Data_03.index.dayofweek
Data_03['Day']=Data_03.index.day
Data_03['Hour']=Data_03.index.hour
03_Data=Data_03
03_Data=03_Data[(03_Data[03_Data.columns] >= 0).all(axis=1)]
03_Data=03_Data.dropna()
03_Data=03_Data.resample('h').mean()
03_Data=03_Data.dropna()
03_Data.head()

ref_03=Data_03['Ref'].to_list()
len(ref_03)
```

[163]: 60913

[164]:
```
import pandas as pd
import scipy.io
import numpy as np
data = pd.read_csv('NO2.txt', header = None,low_memory=False)
data.columns=['WE','AE','Temp','RH','Time']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
```

```
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_NO2=data
Data_NO2['Ref']=Ref_NO2
WE=Data_NO2['WE'].to_list()
AE=Data_NO2['AE'].to_list()
signal=np.array(WE)-np.array(AE)
Data_NO2['Net Signal']=signal
Data_NO2['Month']=Data_NO2.index.month
Data_NO2['Day_of_week']=Data_NO2.index.dayofweek
Data_NO2['Day']=Data_NO2.index.day
Data_NO2['Hour']=Data_NO2.index.hour
NO2_Data=Data_NO2
NO2_Data=NO2_Data[(NO2_Data[NO2_Data.columns] >= 0).all(axis=1)]
NO2_Data=NO2_Data.dropna()
data = pd.read_csv('Conc_NO2.txt', header = None,low_memory=False)
data.columns=['Lab1','Temp','RH','Time','Ref']
Time=data['Time'].to_list()
time=[]
subscript = str.maketrans("0123456789", "      ")
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_NO2=data
signal=np.array(WE)-np.array(AE)
Data_NO2['Net Signal']=signal
Data_NO2['Month']=Data_NO2.index.month
Data_NO2['Day_of_week']=Data_NO2.index.dayofweek
Data_NO2['Day']=Data_NO2.index.day
Data_NO2['Hour']=Data_NO2.index.hour
Data_NO2['Ref_O3']=ref_O3
NO2_Data=Data_NO2
NO2_Data=NO2_Data[(NO2_Data[NO2_Data.columns] >= 0).all(axis=1)]
NO2_Data=NO2_Data.dropna()
NO2_Data=NO2_Data.resample('h').mean()
NO2_Data=NO2_Data.dropna()
NO2_Data.head()
```

[164]:
```
                          Lab1       Temp         RH        Ref  Net Signal  \
Date
2019-10-02 11:00:00  460.448301  26.378438  58.063437  15.230400    7.850000
2019-10-02 12:00:00  557.247199  25.795055  48.256857   5.384051   21.081422
2019-10-02 15:00:00  566.301152  30.418466  50.153181  10.084125    9.323533
2019-10-03 15:00:00   84.482370  29.421250  52.411845  12.621282   22.596524
2019-10-03 16:00:00  116.263856  29.211333  53.102667   9.592208   30.194000

                     Month  Day_of_week  Day  Hour     Ref_O3
Date
2019-10-02 11:00:00   10.0          2.0  2.0  11.0  46.094860
2019-10-02 12:00:00   10.0          2.0  2.0  12.0  57.532808
2019-10-02 15:00:00   10.0          2.0  2.0  15.0  40.068225
2019-10-03 15:00:00   10.0          3.0  3.0  15.0  33.473237
2019-10-03 16:00:00   10.0          3.0  3.0  16.0  33.052590
```

[165]:
```
#Ref=NO2_Data['Ref'].to_list()
#NO2_Data=NO2_Data[NO2_Data.Ref.between(np.mean(Ref)-0.3*np.std(Ref), np.
 ↪mean(Ref)+0.3*np.std(Ref))]
#NO2_Data.shape
```

6.1 Model 1: Linear Regression (LR)

[166]:
```
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
#'Ref_CO','Ref_SO2','Ref_O3',
#,'Month','Day_of_week','Day','Hour'
X=NO2_Data[['Net␣
 ↪Signal','Lab1','Temp','RH','Month','Day_of_week','Hour','Ref_O3']]#'Ref_O3'
y=NO2_Data['Ref']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
len(X_test)
```

[166]: 756

[167]:
```
lr = LinearRegression()
model = lr.fit(X_train.drop(['Lab1'], axis=1), y_train)
pred = model.predict(X_test.drop(['Lab1'], axis=1))
lab1=X_test['Lab1'].to_list()
for i in range(len(lab1)):
    if lab1[i]>100:
        lab1[i]=np.mean(lab1)
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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_lr_NO2=Pearson_lr
sMAPE_lab_NO2=sMAPE_lab
RMSE_lab_NO2=round(RMSE_lab/np.mean(np.array(lab1)),2)
Pearson_lab_NO2=Pearson_lab
R2_lr_NO2=round(sm.r2_score(y_test, pred), 2)
R2_lab_NO2=round(sm.r2_score(y_test, lab1), 2)
RMSE_Lr_NO2=RMSE_lr
RMSE_Lab_NO2=RMSE_lab

A=len(y_test)-200
B=120
D=max(y_test[A:])-0.15*max(y_test[A:])
C=max(y_test[A:])-0.05*max(y_test[A:])
Pearson_lr_NO2,R2_lr_NO2,RMSE_Lr_NO2
```

[167]:
```
(0.9, 0.81, 5.2)
```

subscript = str.maketrans("0123456789", "   ") fig= plt.figure(figsize=(8,6)) index=[i for i in range(1,201)] ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='#513e00',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'LR-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.75,1)) plt.ylabel('NO2 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-150, C,r'$R^2(LR)$ ='+str(R2_lr_NO2) , fontsize = 14, color='#513e00') #plt.text(B-150, D,r'$R^2(Lab)$ ='+str(R2_lab_NO2) , fontsize = 14, color='#426eff') #plt.text(B-700, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='#513e00') #plt.text(B-700, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Linear Regression Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[168]:
```
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))

MBE_LR_NO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_LR_NO2=CRMSE(y_test,pred)/np.std(y_test)
MBE_LAB_NO2=MBE(lab1,y_test)/(2.6*np.std(y_test))
CRMSE_LAB_NO2=CRMSE(y_test,lab1)/(2.6*np.std(y_test))
pred_lr=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 3.96
Mean squared error(MSE) = 26.58
Median absolute error = 3.3
Explain variance score = 0.81
R2 score = 0.81
```

6.2 Model 2: Support Vector Regression (SVR)

[169]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'poly')
regressor.fit(X_train.drop(['Lab1'], axis=1), y_train)
pred = regressor.predict(X_test.drop(['Lab1'], axis=1))
for i in range(len(Pred)):
    if pred[i]<0:
        pred[i]=np.mean(np.array(pred))
```

[170]:
```
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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_svr_NO2=Pearson_lr
R2_svr_NO2=round(sm.r2_score(y_test, pred), 2)
RMSE_Svr_NO2=RMSE_lr
```

```
Pearson_svr_NO2,R2_svr_NO2,RMSE_Svr_NO2
```

[170]: (0.91, 0.82, 5.0)

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='brown',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'SVR-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.74,1)) plt.ylabel('NO2 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-150, C, r'$R^2(SVR)$ ='+str(R2_svr_NO2), fontsize = 14, color='brown') #plt.text(B-150, D, r'$R^2(Lab)$ ='+str(R2_lab_NO2), fontsize = 14, color='#426eff') #plt.text(B-700, C, 'Pearson r(SVR)='+str(Pearson_lr), fontsize = 14, color='brown') #plt.text(B-700, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Support Vector Regression (SVR) Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[171]:
```
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))

MBE_SVR_NO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_SVR_NO2=CRMSE(y_test,pred)/np.std(y_test)
pred_svr=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 3.72
Mean squared error(MSE) = 25.14
Median absolute error = 2.98
Explain variance score = 0.82
R2 score = 0.82
```

6.3 Model 3: Random Forest

[172]:
```
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 500,min_samples_split=␣
↪2,min_samples_leaf= 1,
                                  max_features= 'sqrt',
```

```
                                     random_state =␣
↪0,max_depth=None,bootstrap=False)

# fit the regressor with x and y data
regressor.fit(X_train.drop(['Lab1'], axis=1), y_train)
```

[172]:
```
RandomForestRegressor(bootstrap=False, max_features='sqrt', n_estimators=500,
                      random_state=0)
```

[173]:
```
Index=[i for i in range(len(y_test))]
features_NO2=regressor.feature_importances_
pred = regressor.predict(X_test.drop(['Lab1'], axis=1))
pred_rf_no2=pred
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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_rf_NO2=Pearson_lr
R2_rf_NO2=round(sm.r2_score(y_test, pred), 2)
RMSE_Rf_NO2=RMSE_lr
Pearson_rf_NO2,R2_rf_NO2,RMSE_Rf_NO2
```

[173]:
```
(0.97, 0.93, 3.1)
```

fig= plt.figure(figsize=(10,5))

ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='indigo',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'RF-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.79,1)) plt.ylabel('NO2 Concentration(ppb)'.translate(subscript),fontsize=18) plt.text(B-15, C,r'$R^2(RF)$ ='+str(R2_rf_NO2) , fontsize = 14, color='indigo') plt.text(B-15, D,r'$R^2(Lab)$ ='+str(R2_lab_NO2) , fontsize = 14, color='#426eff') plt.text(B-73, C, 'Pearson r(RF)='+str(Pearson_lr), fontsize = 14, color='indigo') plt.text(B-73, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Random Forest(RF) Calibration vs Laboratory Calibration',fontsize=18) plt.xlabel('Last 100 hours of testing period',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[174]:
```
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))

MBE_RF_NO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_RF_NO2=CRMSE(y_test,pred)/np.std(y_test)
pred_rf=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 2.31
Mean squared error(MSE) = 9.7
Median absolute error = 1.76
Explain variance score = 0.93
R2 score = 0.93
```

6.4 Model 4 : ANN

[175]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from sklearn.preprocessing import StandardScaler
model = Sequential()
model.add(Dense(6, input_shape = (7,),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(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_5"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_24 (Dense)             (None, 6)                 48

_________________________________________________________________
dense_25 (Dense)             (None, 128)               896
```

```
_________________________________________________________________
dense_26 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_27 (Dense)             (None, 100)               12900
_________________________________________________________________
dense_28 (Dense)             (None, 1)                 101
=================================================================
Total params: 30,457
Trainable params: 30,457
Non-trainable params: 0
_________________________________________________________________
```

[176]:
```
scaler = StandardScaler()
scaler.fit(X_train.drop(['Lab1'], axis=1))
X_train_scaled=scaler.transform(X_train.drop(['Lab1'], axis=1))
X_test_scaled=scaler.transform(X_test.drop(['Lab1'], axis=1))
model.fit(X_train_scaled, y_train, batch_size= 100, epochs=100, verbose= 0)
```

[176]:
```
<tensorflow.python.keras.callbacks.History at 0x167192280>
```

[177]:
```
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)
```

[177]:
```
756
```

[178]:
```
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=round(RMSE_lr/np.mean(np.array(y_test)),2)
Pearson_ann_NO2=Pearson_lr
R2_ann_NO2=round(sm.r2_score(y_test, pred), 2)
RMSE_Ann_NO2=RMSE_lr
Pearson_ann_NO2,R2_ann_NO2,RMSE_Ann_NO2
```

[178]:
```
(0.96, 0.9, 3.7)
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='tomato',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'ANN-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.74,1)) plt.ylabel('NO2 Concentration(ppb)',fontsize=18) #plt.text(B-150, C, r'$R^2(ANN)$ ='+str(R2_ann_NO2), fontsize = 14, color='tomato') #plt.text(B-150, D, r'$R^2(Lab)$ ='+str(R2_lab_NO2), fontsize = 14, color='#426eff') #plt.text(B-700, C, 'Pearson r(ANN)='+str(Pearson_lr), fontsize = 14, color='tomato') #plt.text(B-700, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title(' Artificial Neural Network(ANN) Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[179]:
```
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))

MBE_ANN_NO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_ANN_NO2=CRMSE(y_test,pred)/(np.std(y_test))
pred_ann=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 2.74
Mean squared error(MSE) = 13.65
Median absolute error = 2.1
Explain variance score = 0.92
R2 score = 0.9
```

7 Model 5: XGBoost

[180]:
```
from xgboost import  XGBRegressor
from numpy import absolute
from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
# create an xgboost regression model
#n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,colsample_bytree=0.
↪4,alpha=10
model = XGBRegressor(n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,
                     colsample_bytree=0.4,alpha=10)
model.fit(X_train.drop(['Lab1'], axis=1),y_train)
```

[180]: XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.4, eta=0.01, gamma=0,
             gpu_id=-1, importance_type='gain', interaction_constraints='',
             learning_rate=0.00999999978, max_delta_step=0, max_depth=5,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=10000, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=10, reg_lambda=1, scale_pos_weight=1, subsample=0.9,
             tree_method='exact', validate_parameters=1, verbosity=None)

[181]:
```
pred = model.predict(X_test.drop(['Lab1'], axis=1))
pred_xgb_no2=pred
Y_test=y_test.to_list()
Y_test=pd.Series(Y_test,index =Index)
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_xgb_NO2=sMAPE_lr
RMSE_xgb_NO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_xgb_NO2=Pearson_lr
R2_xgb_NO2=round(sm.r2_score(y_test, pred), 2)
RMSE_Xgb_NO2=RMSE_lr
Pearson_xgb_NO2,R2_xgb_NO2,RMSE_Xgb_NO2
```

[181]: (0.97, 0.93, 3.1)

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='darkgoldenrod',linewidth=3) plt.plot(index,lab1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'XGBoost-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.69,1)) plt.ylabel('NO2 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-150, C, r'$R^2(XGB)$ ='+str(R2_xgb_NO2), fontsize = 14, color='darkgoldenrod') #plt.text(B-150, D, r'$R^2(Lab)$ ='+str(R2_lab_NO2), fontsize = 14, color='#426eff') #plt.text(B-700, C, 'Pearson r(XGB)='+str(Pearson_lr), fontsize = 14, color='darkgoldenrod') #plt.text(B-700, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('XGBoost Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[182]:
```
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))
```

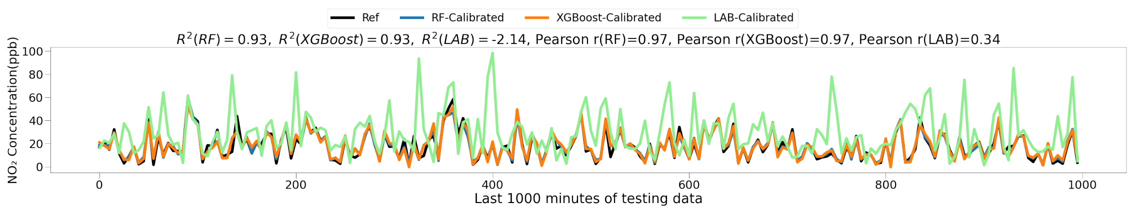
```
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))

MBE_XGB_NO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_XGB_NO2=CRMSE(y_test,pred)/np.std(y_test)
pred_xgb=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 2.31
Mean squared error(MSE) = 9.54
Median absolute error = 1.72
Explain variance score = 0.93
R2 score = 0.93
```

[183]:

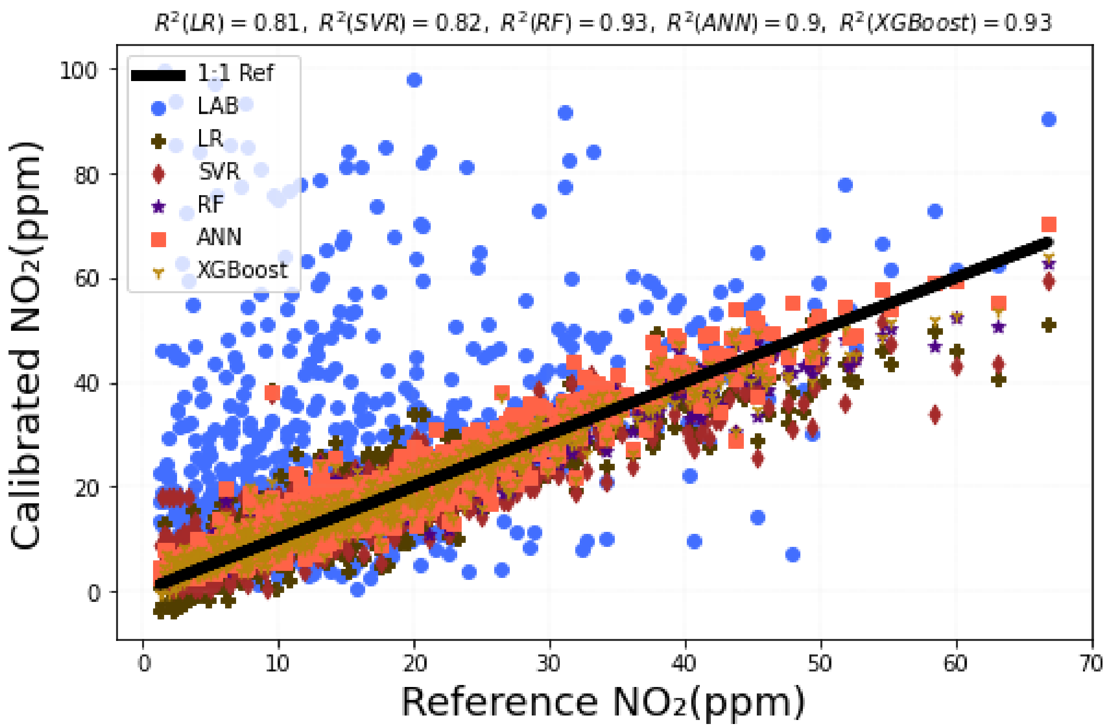
```
fig= plt.figure(figsize=(50,6))
index=[5*i for i in range(200)]
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
ax.patch.set_alpha(0.3)
plt.plot(index,y_test[A:], color='black',linewidth=7)
plt.plot(index,pred_rf_no2[A:],linewidth=7)
plt.plot(index,pred_xgb_no2[A:],linewidth=7)
plt.plot(index,lab1[A:], color='lightgreen',linewidth=7)
plt.legend(['Ref', 'RF-Calibrated','XGBoost-Calibrated', 'LAB-Calibrated'],␣
↪ncol = 4, bbox_to_anchor = (0.7,1.35)
           ,fontsize=28)
plt.ylabel('NO2 Concentration(ppb)'.translate(subscript),fontsize=32)
#plt.text(B+59, C, r'$R^{2}(RF)=$'+str(R2_rf_NO2), fontsize = 22, color='black')
#plt.text(B+59, D, r'$R^{2}(XGBoost)=$'+str(R2_xgb_NO2), fontsize = 22,␣
↪color='black')
#plt.text(B+59, D-0.1*D, r'$R^{2}(Lab)=$'+str(R2_lab_NO2), fontsize = 22,␣
↪color='black')
#plt.text(B+59, D-0.2*D, 'Pearson r(RF)='+str(Pearson_rf_NO2), fontsize = 22,␣
↪color='black')
#plt.text(B+59, D-0.3*D, 'Pearson r(XGBoost)='+str(Pearson_xgb_NO2), fontsize =␣
↪22, color='black')
#plt.text(B+59, D-0.4*D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 22,␣
↪color='black')
#plt.xlabel('Last 200 hours of testing period',fontsize=18)
#plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory␣
↪Calibration',fontsize=18)
#plt.grid(linestyle='-.',linewidth=0.3)
plt.xlabel('Last 1000 minutes of testing data',fontsize=35)
#plt.text(B-133, C+0.11*C, '(b)', fontsize =22, color='black')
```

```
plt.title(r'$R^{2}(RF)=$'+str(R2_rf_NO2)+ r'$, \␣
↪R^{2}(XGBoost)=$'+str(R2_xgb_NO2)+r'$, \ R^{2}(LAB)=$'
          +str(R2_lab_NO2)
        + ', Pearson r(RF)='+str(Pearson_rf_NO2)+', Pearson␣
↪r(XGBoost)='+str(Pearson_xgb_NO2)
          +', Pearson r(LAB)='+str(Pearson_lab),
          fontsize=35)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.tick_params(width=1,length=15)
plt.show()
```



[184]:
```
fig= plt.figure(figsize=(8,5))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
#m0, b0 = np.polyfit(np.array(y_test), np.array(lab1), 1)
#plt.plot(np.array(y_test), m0*np.array(y_test) +␣
↪b0,color='#426eff',linewidth=4)
#m1, b1 = np.polyfit(np.array(y_test), np.array(pred_lr), 1)
#plt.plot(np.array(y_test), m1*np.array(y_test) +␣
↪b1,color='#513e00',linewidth=4)
#m2, b2 = np.polyfit(np.array(y_test), np.array(pred_svr), 1)
#plt.plot(np.array(y_test), m2*np.array(y_test) + b2,color='brown',linewidth=4)
#m3, b3 = np.polyfit(np.array(y_test), np.array(pred_rf), 1)
#plt.plot(np.array(y_test), m3*np.array(y_test) + b3,color='indigo',linewidth=4)
#m4, b4 = np.polyfit(np.array(y_test), np.array(pred_ann), 1)
#plt.plot(np.array(y_test), m4*np.array(y_test) + b4,color='tomato',linewidth=4)
#m5, b5 = np.polyfit(np.array(y_test), np.array(pred_xgb), 1)
#plt.plot(np.array(y_test), m5*np.array(y_test) +␣
↪b5,color='darkgoldenrod',linewidth=4)
plt.scatter(np.array(y_test),np.array(lab1),color='#426eff' )
plt.scatter(np.array(y_test),np.array(pred_lr),color='#513e00',marker='P')
plt.scatter(np.array(y_test),np.array(pred_svr),color='brown',marker='d')
plt.scatter(np.array(y_test),np.array(pred_rf),color='indigo',marker='*')
plt.scatter(np.array(y_test),np.array(pred_ann),color='tomato',marker='s')
plt.scatter(np.array(y_test),np.
↪array(pred_xgb),color='darkgoldenrod',marker='1')
```

```
ax.plot(y_test,y_test, c ="black",linewidth=5)
plt.xlabel('Reference NO2(ppm)'.translate(subscript),fontsize=18)
plt.ylabel('Calibrated NO2(ppm)'.translate(subscript),fontsize=18)
plt.legend(['1:1 Ref','LAB','LR','SVR','RF','ANN','XGBoost'
            ], loc = 2, bbox_to_anchor = (0,1))
#plt.title('CO Sensor',fontsize=18 )
plt.title(r'$ R^{2}(LR)=$'+str(R2_lr_NO2)+r'$,\ R^{2}(SVR)=$'+str(R2_svr_NO2)
          +r'$,\ R^{2}(RF)=$'+str(R2_rf_NO2)+ r'$,\␣
↪R^{2}(ANN)=$'+str(R2_ann_NO2)
          +r'$, \ R^{2}(XGBoost)=$'+str(R2_xgb_NO2),
          fontsize=10)
plt.grid(linestyle='-.',linewidth=0.1)
```



SO2 Calibration

[185]:
```
import pandas as pd
Ref=pd.read_csv('Ref.csv')
Ref["CO"] = 1000 * Ref["CO"]
Ref['Date'] = pd.to_datetime(Ref['Date_Time'])
Ref=Ref.set_index('Date')
Ref.drop('Date_Time',axis = 1, inplace = True)
Ref=Ref.resample('5min').mean()
Ref=Ref[76463:137376]
```

```
Ref_CO=Ref['CO'].to_list()
Ref_NO2=Ref['NO2'].to_list()
Ref_SO2=Ref['SO2'].to_list()
Ref_O3=Ref['O3'].to_list()
```

[186]:
```
import pandas as pd
import scipy.io
import numpy as np
data = pd.read_csv('SO2.txt', header = None,low_memory=False)
data.columns=['WE','AE','Temp','RH','Time']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_SO2=data
Data_SO2['Ref']=Ref_SO2
WE=Data_SO2['WE'].to_list()
AE=Data_SO2['AE'].to_list()
signal=np.array(WE)-np.array(AE)
Data_SO2['Net Signal']=signal
Data_SO2['Month']=Data_SO2.index.month
Data_SO2['Day_of_week']=Data_SO2.index.dayofweek
Data_SO2['Day']=Data_SO2.index.day
Data_SO2['Hour']=Data_SO2.index.hour
SO2_Data=Data_SO2
SO2_Data=SO2_Data[(SO2_Data[SO2_Data.columns] >= 0).all(axis=1)]
SO2_Data=SO2_Data.dropna()
data = pd.read_csv('Conc_SO2.txt', header = None,low_memory=False)
data.columns=['Lab2','Temp','RH','Time','Ref']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_SO2=data
signal=np.array(WE)-np.array(AE)
```

```python
Data_SO2['Net Signal']=signal
Data_SO2['Month']=Data_SO2.index.month
Data_SO2['Day_of_week']=Data_SO2.index.dayofweek
Data_SO2['Day']=Data_SO2.index.day
Data_SO2['Hour']=Data_SO2.index.hour
SO2_Data=Data_SO2
SO2_Data=SO2_Data[(SO2_Data[SO2_Data.columns] >= 0).all(axis=1)]
SO2_Data=SO2_Data.dropna()
SO2_Data=SO2_Data.resample('h').mean()
SO2_Data=SO2_Data.dropna()
SO2_Data.head()
```

[186]:

```
                           Lab2       Temp         RH       Ref  Net Signal  \
Date
2019-10-02 11:00:00  395.131511  26.378438  58.063437  1.634341  136.023750
2019-10-02 12:00:00  189.211370  25.795055  48.256857  1.478131   66.085594
2019-10-02 15:00:00  218.182387  30.623188  49.580620  1.100169   91.640935
2019-10-03 15:00:00   55.916714  29.421250  52.411845  1.102405   17.361905
2019-10-03 16:00:00   36.742766  29.211333  53.102667  1.107573   11.619667

                     Month  Day_of_week  Day  Hour
Date
2019-10-02 11:00:00   10.0          2.0  2.0  11.0
2019-10-02 12:00:00   10.0          2.0  2.0  12.0
2019-10-02 15:00:00   10.0          2.0  2.0  15.0
2019-10-03 15:00:00   10.0          3.0  3.0  15.0
2019-10-03 16:00:00   10.0          3.0  3.0  16.0
```

[187]:

```python
#Ref=SO2_Data['Ref'].to_list()
#SO2_Data=SO2_Data[SO2_Data.Ref.between(np.mean(Ref)-0.3*np.std(Ref), np.
 ↪mean(Ref)+0.3*np.std(Ref))]
#SO2_Data.shape
```

[188]:

```python
#sns.boxplot(x=SO2_Data['Ref'])
#z=np.abs(stats.zscore(SO2_Data))
#SO2_data=SO2_Data[(z < 3).all(axis=1)]
#SO2_data.shape,SO2_Data.shape
```

8 Model 1: Linear Regression (LR)

[189]:

```python
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
#'Ref_CO','Ref_NO2','Ref_O3',
```

```
X=SO2_Data[['Net Signal','Lab2','Temp','RH','Month','Day_of_week','Hour']]
y=SO2_Data['Ref']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
len(X_test)
```

[189]: 510

[190]:
```
lr = LinearRegression()
model = lr.fit(X_train.drop(['Lab2'], axis=1), y_train)
pred = model.predict(X_test.drop(['Lab2'], axis=1))
lab1=X_test['Lab2'].to_list()
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
R2_lr_SO2=round(sm.r2_score(y_test, pred), 2)
R2_lab_SO2=round(sm.r2_score(y_test, lab1), 2)
RMSE_Lr_SO2=RMSE_lr
RMSE_Lab_SO2=RMSE_lab
```

8.1 Scaling Laboratory Calibration

For the purpose of visual comparison with the ref and calibrated measurements, the lab measurement was scaled by a factor of 0.05

[191]:
```
LAB1=0.2*np.array(Lab1)
A=len(y_test)-200
D=max(LAB1[A:])-0.2*max(LAB1[A:])
C=max(LAB1[A:])-0.1*max(LAB1[A:])
B=4000
```

fig= plt.figure(figsize=(8,6)) index=[i for i in range(1,201)] ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='#513e00',linewidth=3) plt.plot(index,LAB1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'LR-Calibrated', 'LAB-Calibrated(scaled)'], loc = 2, bbox_to_anchor = (0.65,1)) plt.ylabel('SO2 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-100, C, r'$R^2(LR)$ ='+str(R2_lr_SO2), fontsize = 14, color='#513e00') #plt.text(B-100, D, r'$R^2(Lab)$ ='+str(R2_lab_SO2), fontsize = 14, color='#426eff') #plt.text(B-400, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='#513e00') #plt.text(B-400, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Linear Regression Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[192]:
```
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))
MBE_LR_SO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_LR_SO2=CRMSE(y_test,pred)/np.std(y_test)
MBE_LAB_SO2=MBE(lab1,y_test)/(25*np.std(y_test))
CRMSE_LAB_SO2=CRMSE(y_test,lab1)/25*(np.std(y_test))
pred_lr=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.42
Mean squared error(MSE) = 0.31
Median absolute error = 0.33
Explain variance score = 0.17
R2 score = 0.17
```

9 Model 2: SVR

[193]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'poly',degree=3)
regressor.fit(X_train.drop(['Lab2'], axis=1), y_train)
pred = regressor.predict(X_test.drop(['Lab2'], axis=1))
```

[194]:
```
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
R2_svr_SO2=round(sm.r2_score(y_test, pred), 2)
RMSE_Svr_SO2=RMSE_lr
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='brown',linewidth=3) plt.plot(index,LAB1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'SVR-Calibrated', 'LAB-Calibrated(Scaled)'], loc = 2, bbox_to_anchor = (0.65,1)) plt.ylabel('SO2 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-200, C,r'$R^2(SVR)$ ='+str(R2_svr_SO2) , fontsize = 14, color='brown') #plt.text(B-200, D, r'$R^2(Lab)$ ='+str(R2_lab_SO2), fontsize = 14, color='#426eff') #plt.text(B-420, C, 'Pearson r(SVR)='+str(Pearson_lr), fontsize = 14, color='brown') #plt.text(B-420, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Support Vector Regression (SVR) Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[195]:
```
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))

MBE_SVR_SO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_SVR_SO2=CRMSE(y_test,pred)/np.std(y_test)
pred_svr=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.41
Mean squared error(MSE) = 0.29
Median absolute error = 0.32
Explain variance score = 0.23
R2 score = 0.23
```

Model 3: Random Forest

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

        # create regressor object
       regressor = RandomForestRegressor(n_estimators = 500,min_samples_split=␣
        ↪2,min_samples_leaf= 1,max_features= 'sqrt',
                                          random_state =␣
        ↪0,max_depth=None,bootstrap=False)

       # fit the regressor with x and y data
       regressor=regressor.fit(X_train.drop(['Lab2'], axis=1), y_train)
```

```
[197]: Index=[i for i in range(len(y_test))]
       features_SO2=regressor.feature_importances_
       pred = regressor.predict(X_test.drop(['Lab2'], axis=1))
       pred_rf_so2=pred
       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
       R2_rf_SO2=round(sm.r2_score(y_test, pred), 2)
       RMSE_Rf_SO2=RMSE_lr
```

fig= plt.figure(figsize=(10,5)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='indigo',linewidth=3) plt.plot(index,LAB1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'RF-Calibrated', 'LAB-Calibrated(Scaled)'], loc = 2, bbox_to_anchor = (0.72,1)) plt.ylabel('SO2 Concentration(ppb)'.translate(subscript),fontsize=18) plt.text(B-20, C,r'$R^2(RF)$ ='+str(R2_rf_SO2) , fontsize = 14, color='indigo') plt.text(B-20, D,r'$R^2(Lab)$ ='+str(R2_lab_SO2) , fontsize = 14, color='#426eff') plt.text(B-70, C, 'Pearson r(RF)='+str(Pearson_lr), fontsize = 14, color='indigo') plt.text(B-70, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

```
[198]: 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))

MBE_RF_SO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_RF_SO2=CRMSE(y_test,pred)/np.std(y_test)
pred_rf=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.31
Mean squared error(MSE) = 0.19
Median absolute error = 0.22
Explain variance score = 0.5
R2 score = 0.5
```

10 Model 4 : ANN

[199]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from sklearn.preprocessing import StandardScaler
model = Sequential()
model.add(Dense(6, input_shape = (6,),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_6"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_29 (Dense)             (None, 6)                 42

_________________________________________________________________
dense_30 (Dense)             (None, 128)               896
```

```
-----------------------------------------------------------------
dense_31 (Dense)             (None, 50)               6450
-----------------------------------------------------------------
dense_32 (Dense)             (None, 1)                51
=================================================================
Total params: 7,439
Trainable params: 7,439
Non-trainable params: 0
-----------------------------------------------------------------
```

[200]:
```
scaler = StandardScaler()
scaler.fit(X_train.drop(['Lab2'], axis=1))
X_train_scaled=scaler.transform(X_train.drop(['Lab2'], axis=1))
X_test_scaled=scaler.transform(X_test.drop(['Lab2'], axis=1))
model.fit(X_train_scaled, y_train, batch_size= 200, epochs=100, verbose= 0)
```

[200]:
```
<tensorflow.python.keras.callbacks.History at 0x16d45d8e0>
```

[201]:
```
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)
```

[201]:
```
510
```

[202]:
```
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
R2_ann_SO2=round(sm.r2_score(y_test, pred), 2)
RMSE_Ann_SO2=RMSE_lr
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='tomato',linewidth=3) plt.plot(index,LAB1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'ANN-Calibrated', 'LAB-

Calibrated(Scaled)'], loc = 2, bbox_to_anchor = (0.65,1)) plt.ylabel('SO2 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-200, C,r'$R^2(ANN)$ ='+str(R2_ann_SO2) , fontsize = 14, color='tomato') #plt.text(B-200, D,r'$R^2(Lab)$ ='+str(R2_lab_SO2) , fontsize = 14, color='#426eff') #plt.text(B-400, C, 'Pearson r(ANN)='+str(Pearson_lr), fontsize = 14, color='tomato') #plt.text(B-400, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Artificial Neural Network(ANN) Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[203]:
```
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))

MBE_ANN_SO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_ANN_SO2=CRMSE(y_test,pred)/np.std(y_test)
pred_ann=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.36
Mean squared error(MSE) = 0.24
Median absolute error = 0.26
Explain variance score = 0.37
R2 score = 0.37
```

11 Model 5 : XGBoost

[204]:
```
from xgboost import  XGBRegressor
from numpy import absolute
from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
# create an xgboost regression model
#n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,colsample_bytree=0.
↪4,alpha=10
model = XGBRegressor(n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,
                     colsample_bytree=0.4,alpha=10)

model.fit(X_train.drop(['Lab2'], axis=1),y_train)
```

```
[204]: XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
                 colsample_bynode=1, colsample_bytree=0.4, eta=0.01, gamma=0,
                 gpu_id=-1, importance_type='gain', interaction_constraints='',
                 learning_rate=0.00999999978, max_delta_step=0, max_depth=5,
                 min_child_weight=1, missing=nan, monotone_constraints='()',
                 n_estimators=10000, n_jobs=0, num_parallel_tree=1, random_state=0,
                 reg_alpha=10, reg_lambda=1, scale_pos_weight=1, subsample=0.9,
                 tree_method='exact', validate_parameters=1, verbosity=None)
```

```
[205]: pred = model.predict(X_test.drop(['Lab2'], axis=1))
pred_xgb_so2=pred
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_xgb_SO2=sMAPE_lr
RMSE_xgb_SO2=RMSE_lr/np.mean(np.array(y_test))
Pearson_xgb_SO2=Pearson_lr
R2_xgb_SO2=round(sm.r2_score(y_test, pred), 2)
RMSE_Xgb_SO2=RMSE_lr
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test[A:], color='limegreen',linewidth=3) plt.plot(index,pred[A:], color='darkgoldenrod',linewidth=3) plt.plot(index,LAB1[A:], color='#426eff',linewidth=3) plt.legend(['Ref', 'XGBoost-Calibrated', 'LAB-Calibrated(scaled)'], loc = 2, bbox_to_anchor = (0.65,1)) plt.ylabel('SO2 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-200, C,r'$R^2(XGB)$ ='+str(R2_xgb_SO2) , fontsize = 14, color='darkgoldenrod') #plt.text(B-200, D,r'$R^2(Lab)$ ='+str(R2_lab_SO2), fontsize = 14, color='#426eff') #plt.text(B-400, C, 'Pearson r(XGB)='+str(Pearson_lr), fontsize = 14, color='darkgoldenrod') #plt.text(B-400, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('XGBoost Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

```
[206]: 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))
```

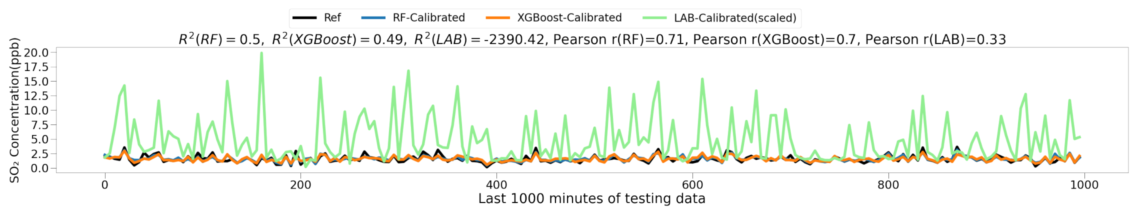
```
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))

MBE_XGB_SO2=MBE(pred,y_test)/np.std(y_test)
CRMSE_XGB_SO2=CRMSE(y_test,pred)/np.std(y_test)
pred_xgb=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 0.34
Mean squared error(MSE) = 0.19
Median absolute error = 0.28
Explain variance score = 0.49
R2 score = 0.49
```

[207]:

```
fig= plt.figure(figsize=(50,6))
index=[5*i for i in range(200)]
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
ax.patch.set_alpha(0.3)
plt.plot(index,y_test[A:], color='black',linewidth=7)
plt.plot(index,pred_rf_so2[A:],linewidth=7)
plt.plot(index,pred_xgb_so2[A:],linewidth=7)
plt.plot(index,LAB1[A:], color='lightgreen',linewidth=7)
plt.legend(['Ref', 'RF-Calibrated','XGBoost-Calibrated',␣
↪'LAB-Calibrated(scaled)'], ncol = 4,
           bbox_to_anchor = (0.7,1.35),fontsize=28)
plt.ylabel('SO2 Concentration(ppb)'.translate(subscript),fontsize=32)
#plt.text(B-23, C+0.11*C, r'$R^{2}(RF)=$'+str(R2_rf_SO2), fontsize = 22,␣
↪color='red')
#plt.text(B-23, D+0.11*D, r'$R^{2}(XGBoost)=$'+str(R2_xgb_SO2), fontsize = 22,␣
↪color='green')
#plt.text(B-23, D, r'$R^{2}(Lab)=$'+str(R2_lab_SO2), fontsize = 22,␣
↪color='blue')
#plt.text(B-130, C+0.11*C, 'Pearson r(RF)='+str(Pearson_rf_SO2), fontsize = 22,␣
↪color='red')
#plt.text(B-130, D+0.11*D, 'Pearson r(XGBoost)='+str(Pearson_xgb_SO2), fontsize␣
↪= 22, color='green')
#plt.text(B-130, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 22,␣
↪color='blue')
plt.title(r'$R^{2}(RF)=$'+str(R2_rf_SO2)+ r'$, \␣
↪R^{2}(XGBoost)=$'+str(R2_xgb_SO2)+r'$, \ R^{2}(LAB)=$'
          +str(R2_lab_SO2)
        + ', Pearson r(RF)='+str(Pearson_rf_SO2)+', Pearson␣
↪r(XGBoost)='+str(Pearson_xgb_SO2)
```

```
         +', Pearson r(LAB)='+str(Pearson_lab),
         fontsize=35)
plt.xlabel('Last 1000 minutes of testing data',fontsize=35)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.tick_params(width=1,length=15)
#plt.xlabel('Last 200 hours of testing period',fontsize=18)
#plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory Calibration',fontsize=18)
#plt.grid(linestyle='-.',linewidth=0.3)
#plt.text(B-4000, C+0.1*C, '(c)', fontsize =22, color='black')
plt.show()
```

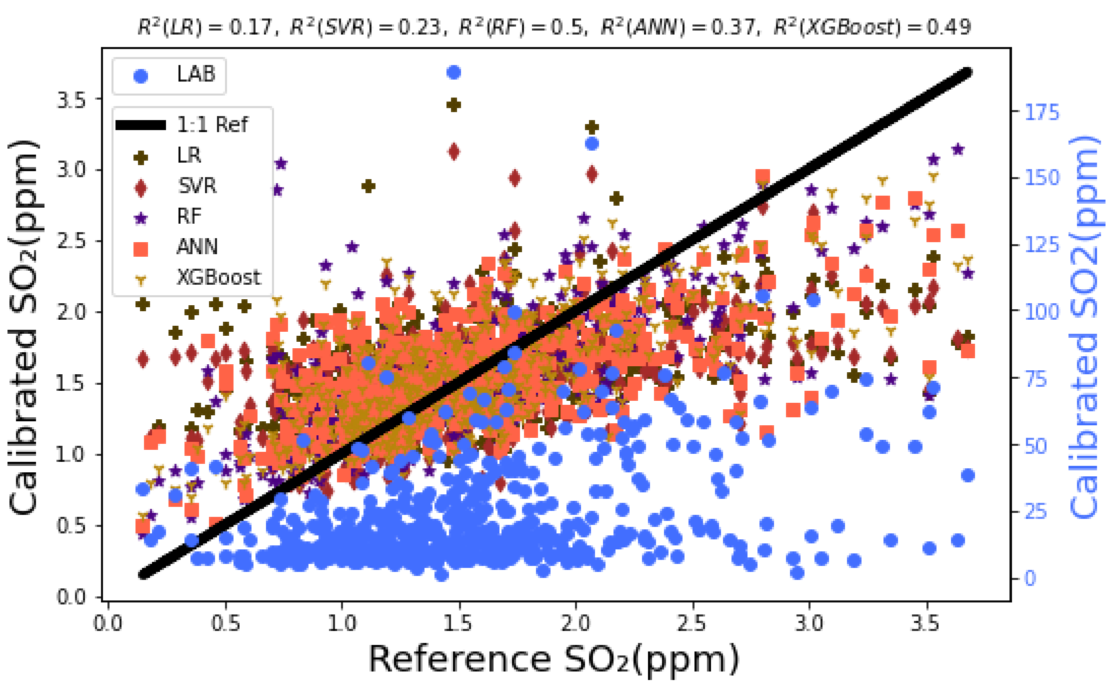


[208]:

```
#fig= plt.figure(figsize=(12,5))
fig, ax1 = plt.subplots(figsize=(8,5))
ax2 = ax1.twinx()
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
#m0, b0 = np.polyfit(np.array(y_test), np.array(lab1), 1)
#plt.plot(np.array(y_test), m0*np.array(y_test) + b0,color='#426eff',linewidth=4)
#m1, b1 = np.polyfit(np.array(y_test), np.array(pred_lr), 1)
#plt.plot(np.array(y_test), m1*np.array(y_test) + b1,color='#513e00',linewidth=4)
#m2, b2 = np.polyfit(np.array(y_test), np.array(pred_svr), 1)
#plt.plot(np.array(y_test), m2*np.array(y_test) + b2,color='brown',linewidth=4)
#m3, b3 = np.polyfit(np.array(y_test), np.array(pred_rf), 1)
#plt.plot(np.array(y_test), m3*np.array(y_test) + b3,color='indigo',linewidth=4)
#m4, b4 = np.polyfit(np.array(y_test), np.array(pred_ann), 1)
#plt.plot(np.array(y_test), m4*np.array(y_test) + b4,color='tomato',linewidth=4)
#m5, b5 = np.polyfit(np.array(y_test), np.array(pred_xgb), 1)
#plt.plot(np.array(y_test), m5*np.array(y_test) + b5,color='darkgoldenrod',linewidth=4)
ax2.scatter(np.array(y_test),np.array(lab1),color='#426eff' )
ax1.scatter(np.array(y_test),np.array(pred_lr),color='#513e00',marker='P')
ax1.scatter(np.array(y_test),np.array(pred_svr),color='brown',marker='d')
ax1.scatter(np.array(y_test),np.array(pred_rf),color='indigo',marker='*')
ax1.scatter(np.array(y_test),np.array(pred_ann),color='tomato',marker='s')
```

```
ax1.scatter(np.array(y_test),np.
 ↪array(pred_xgb),color='darkgoldenrod',marker='1')
ax1.plot(y_test,y_test, c ="black",linewidth=5)
ax1.set_xlabel('Reference SO2(ppm)'.translate(subscript),fontsize=18)
ax1.set_ylabel('Calibrated SO2(ppm)'.translate(subscript),fontsize=18)
ax2.set_ylabel('Calibrated SO2(ppm)',color='#426eff',fontsize=18)
ax2.tick_params(axis='y', labelcolor='#426eff')
ax2.legend(['LAB'
            ], loc = 2, bbox_to_anchor = (0,1))
ax1.legend(['1:1 Ref','LR','SVR','RF','ANN','XGBoost'
            ], loc = 2, bbox_to_anchor = (0,0.91))
#plt.title('CO Sensor',fontsize=18 )
plt.title(r'$ R^{2}(LR)=$'+str(R2_lr_SO2)+r'$,\ R^{2}(SVR)=$'+str(R2_svr_SO2)
          +r'$,\ R^{2}(RF)=$'+str(R2_rf_SO2)+ r'$,\␣
 ↪R^{2}(ANN)=$'+str(R2_ann_SO2)
          +r'$, \ R^{2}(XGBoost)=$'+str(R2_xgb_SO2),
          fontsize=10)
#plt.grid(linestyle='-.',linewidth=0.1)
```

[208]: Text(0.5, 1.0, '$ R^{2}(LR)=0.17,\\ R^{2}(SVR)=0.23,\\ R^{2}(RF)=0.5,\\ R^{2}(ANN)=0.37, \\ R^{2}(XGBoost)=$0.49')



12 O3 CALIBRATION

[209]:
```
import pandas as pd
import scipy.io
import numpy as np
data = pd.read_csv('O3.txt', header = None,low_memory=False)
data.columns=['AE','WE','Temp','RH','Time']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_O3=data
Data_O3['Ref']=Ref_O3
WE=Data_O3['WE'].to_list()
AE=Data_O3['AE'].to_list()
signal=np.array(WE)-np.array(AE)
Data_O3['Net Signal']=signal
Data_O3['Month']=Data_O3.index.month
Data_O3['Day_of_week']=Data_O3.index.dayofweek
Data_O3['Day']=Data_O3.index.day
Data_O3['Hour']=Data_O3.index.hour
O3_Data=Data_O3
O3_Data=O3_Data[(O3_Data[O3_Data.columns] >= 0).all(axis=1)]
O3_Data=O3_Data.dropna()
data = pd.read_csv('Conc_O3.txt', header = None,low_memory=False)
data.columns=['Lab1','Temp','RH','Time','Ref']
Time=data['Time'].to_list()
time=[]
for i in range(len(Time)):
    time.append(float(abs(Time[i])))
Time=np.array(time)
Date=pd.to_datetime(Time-719529,unit='d').round('s')
data['Date'] = Date.tolist()
data=data.set_index('Date')
data.drop('Time',axis = 1, inplace = True)
data=data.resample('5min').mean()
Data_O3=data
signal=np.array(WE)-np.array(AE)
Data_O3['Net Signal']=signal
Data_O3['Month']=Data_O3.index.month
Data_O3['Day_of_week']=Data_O3.index.dayofweek
```

```
Data_03['Day']=Data_03.index.day
Data_03['Hour']=Data_03.index.hour
ref_NO2=Data_NO2['Ref'].to_list()
Data_03['Ref_NO2']=ref_NO2
03_Data=Data_03
03_Data=03_Data[(03_Data[03_Data.columns] >= 0).all(axis=1)]
03_Data=03_Data.dropna()
03_Data=03_Data.resample('h').mean()
03_Data=03_Data.dropna()
03_Data.head()
```

[209]:

```
                           Lab1       Temp         RH        Ref  Net Signal  \
Date
2019-10-02 11:00:00  621.625704  26.378438  58.063437  46.094860    3.605625
2019-10-02 12:00:00  725.154408  25.795055  48.256857  57.532808   13.865109
2019-10-07 10:00:00  108.196313  32.344264  37.260757  47.259007   11.447809
2019-10-07 11:00:00  135.822676  34.926112  35.013036  42.114260   10.075221
2019-10-07 12:00:00  203.757758  36.201221  31.829282  45.701366    7.624153

                     Month  Day_of_week  Day  Hour     Ref_NO2
Date
2019-10-02 11:00:00   10.0          2.0  2.0  11.0  15.230400
2019-10-02 12:00:00   10.0          2.0  2.0  12.0   5.384051
2019-10-07 10:00:00   10.0          0.0  7.0  10.0   4.255772
2019-10-07 11:00:00   10.0          0.0  7.0  11.0  16.268034
2019-10-07 12:00:00   10.0          0.0  7.0  12.0  12.770444
```

[210]:

```
#Ref=03_Data['Ref'].to_list()
#03_Data=03_Data[03_Data.Ref.between(np.mean(Ref)-0.3*np.std(Ref), np.
 ↪mean(Ref)+0.3*np.std(Ref))]
#03_Data.shape
```

12.1 Model 1: LR

[211]:

```
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
#,'Ref_CO','Ref_NO2','Ref_SO2'
X=03_Data[['Net␣
 ↪Signal','Lab1','Temp','RH','Month','Day_of_week','Hour','Ref_NO2']]#,'Ref_NO2'
y=03_Data['Ref']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
len(X_test)
```

[211]: 707

[212]:
```
lr = LinearRegression()
model = lr.fit(X_train.drop(['Lab1'], axis=1), y_train)
pred = model.predict(X_test.drop(['Lab1'], axis=1))
lab1=X_test['Lab1'].to_list()
for i in range(len(lab1)):
    if lab1[i]>370:
        lab1[i]=np.mean(lab1)
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
R2_lr_O3=round(sm.r2_score(y_test, pred), 2)
R2_lab_O3=round(sm.r2_score(y_test, lab1), 2)
RMSE_Lr_O3=RMSE_lr
RMSE_Lab_O3=RMSE_lab

A=len(y_test)
D=max(lab1)-0.10*max(lab1)
C=max(lab1)-0.03*max(lab1)
B=A

Pearson_lr_O3,R2_lr_O3,RMSE_Lr_O3
```

[212]:
```
(0.95, 0.9, 4.6)
```

fig= plt.figure(figsize=(8,6)) index=[i for i in range(1,len(y_test)+1)] ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test, color='limegreen',linewidth=3) plt.plot(index,pred, color='#513e00',linewidth=3) plt.plot(index,lab1, color='#426eff',linewidth=3) plt.legend(['Ref', 'LR-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.75,1)) plt.ylabel('O3 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-5, C,r'$R^2(LR)$ ='+str(R2_lr_O3) , fontsize = 14, color='#513e00') #plt.text(B-5, D,r'$R^2(Lab)$ ='+str(R2_lab_O3) , fontsize = 14, color='#426eff') #plt.text(B-70, C, 'Pearson r(LR)='+str(Pearson_lr), fontsize = 14, color='#513e00') #plt.text(B-70, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') plt.xlabel('Testing period(hours)',fontsize=18) #plt.title('Linear Regression Calibration vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[213]:
```
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))

MBE_LR_03=MBE(pred,y_test)/np.std(y_test)
CRMSE_LR_03=CRMSE(y_test,pred)/np.std(y_test)
MBE_LAB_03=MBE(lab1,y_test)/(3.6*np.std(y_test))
CRMSE_LAB_03=CRMSE(y_test,lab1)/(3.6*np.std(y_test))
pred_lr=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 3.67
Mean squared error(MSE) = 21.44
Median absolute error = 3.02
Explain variance score = 0.9
R2 score = 0.9
```

12.2 Model 2: SVR

[214]:
```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
regressor = SVR(kernel = 'linear')
regressor.fit(X_train.drop(['Lab1'], axis=1), y_train)
pred = regressor.predict(X_test.drop(['Lab1'], axis=1))
```

[215]:
```
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_03=sMAPE_lr
RMSE_svr_03=RMSE_lr/np.mean(np.array(y_test))
```

```
Pearson_svr_O3=Pearson_lr
R2_svr_O3=round(sm.r2_score(y_test, pred), 2)
RMSE_Svr_O3=RMSE_lr
Pearson_svr_O3,R2_svr_O3,RMSE_Svr_O3
```

[215]:
```
(0.95, 0.9, 4.6)
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) plt.plot(index,y_test, color='limegreen',linewidth=3) plt.plot(index,pred, color='brown',linewidth=3) plt.plot(index,lab1, color='#426eff',linewidth=3) plt.legend(['Ref', 'SVR-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.75,1)) plt.ylabel('O3 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-5, C,r'$R^2(SVR)$ ='+str(R2_svr_O3) , fontsize = 14, color='brown') #plt.text(B-5, D,r'$R^2(Lab)$ ='+str(R2_lab_O3) , fontsize = 14, color='#426eff') #plt.text(B-70, C, 'Pearson r(SVR)='+str(Pearson_lr), fontsize = 14, color='brown') #plt.text(B-70, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') plt.xlabel('Testing period(hours)',fontsize=18) #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Support Vector Regression(SVR) vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[216]:
```
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))

MBE_SVR_O3=MBE(pred,y_test)/np.std(y_test)
CRMSE_SVR_O3=CRMSE(y_test,pred)/np.std(y_test)
pred_svr=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 3.65
Mean squared error(MSE) = 21.6
Median absolute error = 3.01
Explain variance score = 0.9
R2 score = 0.9
```

12.3 Model 3 : Random Forest

[217]:
```
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 500,min_samples_split=␣
↪2,min_samples_leaf= 1,max_features= 'sqrt',
```

```
                                 random_state =␣
↪0,max_depth=None,bootstrap=False)
# fit the regressor with x and y data
regressor.fit(X_train.drop(['Lab1'], axis=1), y_train)
```

[217]:
```
RandomForestRegressor(bootstrap=False, max_features='sqrt', n_estimators=500,
                      random_state=0)
```

[218]:
```
Index=[i for i in range(len(y_test))]
features_O3=regressor.feature_importances_
pred = regressor.predict(X_test.drop(['Lab1'], axis=1))
pred_rf_o3=pred
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
R2_rf_O3=round(sm.r2_score(y_test, pred), 2)
RMSE_Rf_O3=RMSE_lr
Pearson_rf_O3,R2_rf_O3,RMSE_Rf_O3
```

[218]:
```
(0.98, 0.96, 2.8)
```

fig= plt.figure(figsize=(10,5)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) ax.plot(index,y_test, color='limegreen',linewidth=3) ax.plot(index,pred, color='indigo',linewidth=3) ax.plot(index,lab1, color='#426eff',linewidth=3) plt.legend(['Ref', 'RF-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.79,1)) plt.ylabel('O3 Concentration(ppb)'.translate(subscript),fontsize=18) plt.text(B-22, C,r'$R^2(RF)$ ='+str(R2_rf_O3) , fontsize = 14, color='indigo') plt.text(B-22, D,r'$R^2(Lab)$ ='+str(R2_lab_O3) , fontsize = 14, color='#426eff') plt.text(B-72, C, 'Pearson r(RF)='+str(Pearson_lr), fontsize = 14, color='indigo') plt.text(B-72, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') plt.xlabel('Last 100 hours of testing period',fontsize=18) #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Random Forest(RF) vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[219]:
```
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))


MBE_RF_03=MBE(pred,y_test)/np.std(y_test)
CRMSE_RF_03=CRMSE(y_test,pred)/np.std(y_test)
pred_rf=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 2.05
Mean squared error(MSE) = 7.82
Median absolute error = 1.49
Explain variance score = 0.96
R2 score = 0.96
```

[220]:
```
features_03
```

[220]:
```
array([0.09279762, 0.1535448 , 0.19810008, 0.07030181, 0.01125749,
       0.08653907, 0.38745914])
```

12.4 Model 4: ANN

[221]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from sklearn.preprocessing import StandardScaler
model = Sequential()
model.add(Dense(6, input_shape = (7,),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(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_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense_33 (Dense)             (None, 6)                 48
_________________________________________________________________
dense_34 (Dense)             (None, 128)               896
_________________________________________________________________
dense_35 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_36 (Dense)             (None, 100)               12900
_________________________________________________________________
dense_37 (Dense)             (None, 1)                 101
=================================================================
Total params: 30,457
Trainable params: 30,457
Non-trainable params: 0
_________________________________________________________________
```

[222]:
```
scaler = StandardScaler()
scaler.fit(X_train.drop(['Lab1'], axis=1))
X_train_scaled=scaler.transform(X_train.drop(['Lab1'], axis=1))
X_test_scaled=scaler.transform(X_test.drop(['Lab1'], axis=1))
model.fit(X_train_scaled, y_train, batch_size= 100, epochs=200, verbose= 0)
```

[222]:
```
<tensorflow.python.keras.callbacks.History at 0x116f73490>
```

[223]:
```
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)
```

[223]:
```
707
```

[224]:
```
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_03=sMAPE_lr
RMSE_ann_03=RMSE_lr/np.mean(np.array(y_test))
Pearson_ann_03=Pearson_lr
R2_ann_03=round(sm.r2_score(y_test, pred), 2)
RMSE_Ann_03=RMSE_lr
```

```
Pearson_ann_O3,R2_ann_O3,RMSE_Ann_O3
```

[224]: (0.98, 0.96, 3.0)

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) ax.plot(index,y_test, color='limegreen',linewidth=3) ax.plot(index,pred, color='tomato',linewidth=3) ax.plot(index,lab1, color='#426eff',linewidth=3) plt.legend(['Ref', 'ANN-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.75,1)) plt.ylabel('O3 Concentration(ppm)'.translate(subscript),fontsize=18) #plt.text(B-5, C,r'$R^2(ANN)$ ='+str(R2_ann_O3) , fontsize = 14, color='tomato') #plt.text(B-5, D, r'$R^2(Lab)$ ='+str(R2_lab_O3), fontsize = 14, color='#426eff') #plt.text(B-70, C, 'Pearson r(ANN)='+str(Pearson_lr), fontsize = 14, color='tomato') #plt.text(B-70, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') plt.xlabel('Testing period(hours)',fontsize=18) #plt.xlabel('Last 200 hours of testing period',fontsize=18) #plt.title('Artificial Neural Network(ANN) vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

[225]:
```
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))

MBE_ANN_O3=MBE(pred,y_test)/np.std(y_test)
CRMSE_ANN_O3=CRMSE(y_test,pred)/np.std(y_test)
pred_ann=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 2.12
Mean squared error(MSE) = 8.79
Median absolute error = 1.52
Explain variance score = 0.96
R2 score = 0.96
```

13 Model 5: XGBoost

[226]:
```
from xgboost import  XGBRegressor
from numpy import absolute
from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
# create an xgboost regression model
```

```
#n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,colsample_bytree=0.
↪4,alpha=10
model = XGBRegressor(n_estimators=10000, max_depth=5, eta=0.01, subsample=0.9,
                     colsample_bytree=0.4,alpha=10)

model.fit(X_train.drop(['Lab1'], axis=1),y_train)
```

[226]:
```
XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.4, eta=0.01, gamma=0,
             gpu_id=-1, importance_type='gain', interaction_constraints='',
             learning_rate=0.00999999978, max_delta_step=0, max_depth=5,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=10000, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=10, reg_lambda=1, scale_pos_weight=1, subsample=0.9,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

[227]:
```
pred = model.predict(X_test.drop(['Lab1'], axis=1))
pred_xgb_o3=pred
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_xgb_O3=sMAPE_lr
RMSE_xgb_O3=RMSE_lr/np.mean(np.array(y_test))
Pearson_xgb_O3=Pearson_lr
R2_xgb_O3=round(sm.r2_score(y_test, pred), 2)
RMSE_Xgb_O3=RMSE_lr
Pearson_xgb_O3,R2_xgb_O3,RMSE_Xgb_O3
```

[227]:
```
(0.98, 0.96, 2.8)
```

fig= plt.figure(figsize=(8,6)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) ax.plot(index,y_test, color='limegreen',linewidth=3) ax.plot(index,pred, color='darkgoldenrod',linewidth=3) ax.plot(index,lab1, color='#426eff',linewidth=3) plt.legend(['Ref', 'XGBoost-Calibrated', 'LAB-Calibrated'], loc = 2, bbox_to_anchor = (0.75,1)) plt.ylabel('O3 Concentration(ppb)'.translate(subscript),fontsize=18) #plt.text(B-5, C,r'$R^2(XGB)$ ='+str(R2_xgb_O3) , fontsize = 14, color='darkgoldenrod') #plt.text(B-5, D,r'$R^2(Lab)$ ='+str(R2_lab_O3) , fontsize = 14, color='#426eff') #plt.text(B-70, C, 'Pearson r(XGB)='+str(Pearson_lr), fontsize = 14, color='darkgoldenrod') #plt.text(B-70, D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='#426eff') plt.xlabel('Testing period(hours)',fontsize=18) #plt.xlabel('Last 200 hours of testing period',fontsize=18)

#plt.title('XGBoost vs Laboratory Calibration',fontsize=18) plt.grid(linestyle='-.',linewidth=0.3) plt.show()

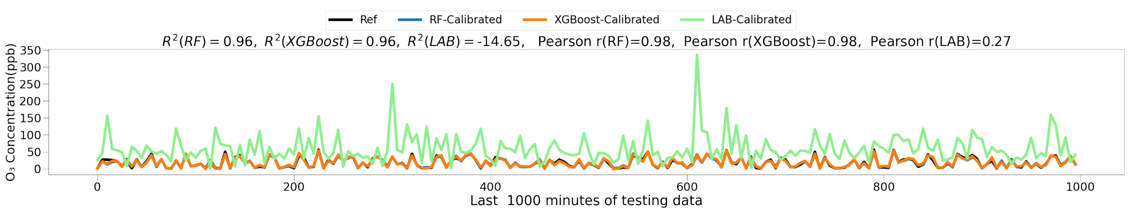
[228]:
```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, pred),␣
↪2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, pred),␣
↪2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, pred),␣
↪2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
↪pred), 2))
print("R2 score =", round(sm.r2_score(y_test, pred), 2))
MBE_XGB_O3=MBE(pred,y_test)/np.std(y_test)
CRMSE_XGB_O3=CRMSE(y_test,pred)/np.std(y_test)
pred_xgb=pred
```

```
Regressor model performance:
Mean absolute error(MAE) = 2.13
Mean squared error(MSE) = 8.11
Median absolute error = 1.59
Explain variance score = 0.96
R2 score = 0.96
```

[229]:
```
A=len(y_test)-200
```

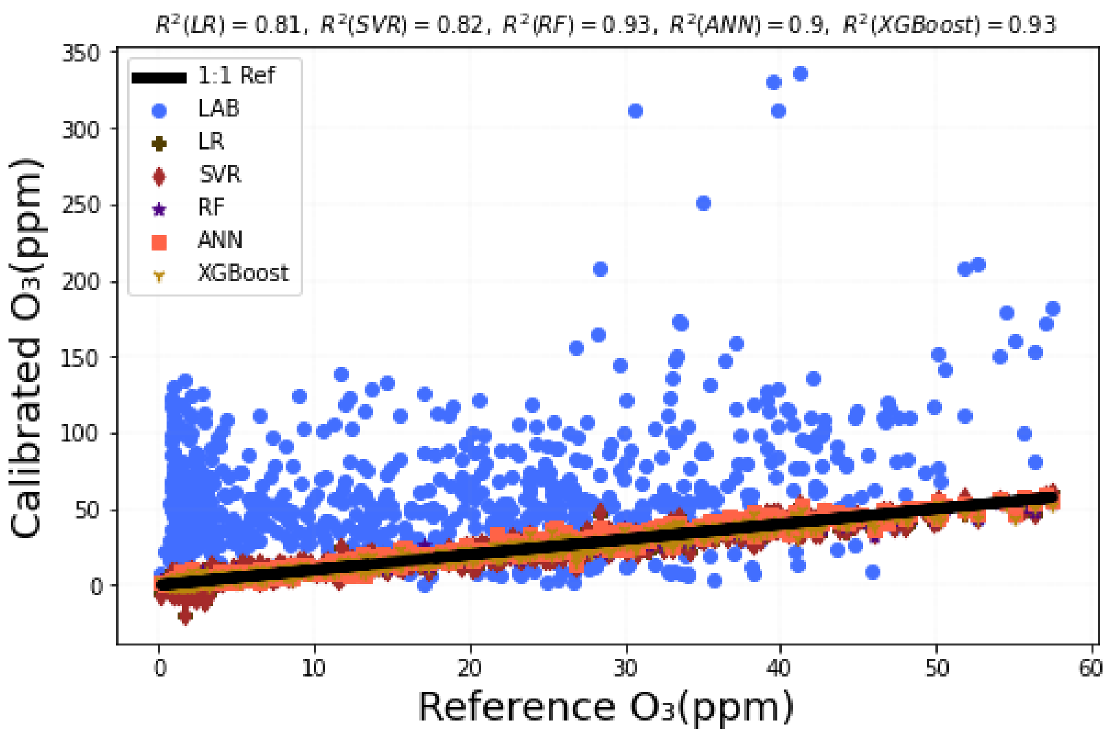
[230]:
```
fig= plt.figure(figsize=(50,6))
ax = fig.add_subplot(111)
index=[5*i for i in range(200)]
#ax.patch.set_facecolor('lightblue')
ax.patch.set_alpha(0.3)
plt.plot(index,y_test[A:], color='black',linewidth=7)
plt.plot(index,pred_rf_o3[A:],linewidth=7)
plt.plot(index,pred_xgb_o3[A:],linewidth=7)
plt.plot(index,Lab1[A:], color='lightgreen',linewidth=7)
plt.legend(['Ref', 'RF-Calibrated','XGBoost-Calibrated', 'LAB-Calibrated'],␣
↪ncol = 4, bbox_to_anchor = (0.7,1.35),
           fontsize=28)
plt.ylabel('O3 Concentration(ppb)'.translate(subscript),fontsize=32)
#plt.text(B-65, C, r'$R^{2}(RF)=$'+str(R2_rf_O3), fontsize = 14, color='red')
#plt.text(B-65, D, r'$R^{2}(XGBoost)=$'+str(R2_xgb_O3), fontsize = 14,␣
↪color='green')
#plt.text(B-65, D-0.07*D, r'$R^{2}(Lab)=$'+str(R2_lab_O3), fontsize = 14,␣
↪color='blue')
#plt.text(B-118, C, 'Pearson r(RF)='+str(Pearson_rf_O3), fontsize = 14,␣
↪color='red')
```

```
#plt.text(B-118, D, 'Pearson r(XGBoost)='+str(Pearson_xgb_O3), fontsize = 14, color='green')
#plt.text(B-118, D-0.07*D, 'Pearson r(Lab)='+str(Pearson_lab), fontsize = 14, color='blue')
#plt.xlabel('Last 200 hours of testing period',fontsize=18)
#plt.title('Visualization: Random Forest(RF) Calibration vs Laboratory Calibration',fontsize=18)
#plt.grid(linestyle='-.',linewidth=0.3)
plt.title(r'$R^{2}(RF)=$'+str(R2_rf_O3)+ r'$, \ R^{2}(XGBoost)=$'+str(R2_xgb_O3)+r'$, \  R^{2}(LAB)=$'+str(R2_lab_O3)
        + ',   Pearson r(RF)='+str(Pearson_rf_O3)+',  Pearson r(XGBoost)='+str(Pearson_xgb_O3)
          +',  Pearson r(LAB)='+str(Pearson_lab),
          fontsize=35)
plt.xlabel('Last  1000 minutes of testing data',fontsize=35)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.tick_params(width=1,length=15)
#plt.text(B-182, 0.85*C, '(d)', fontsize =24, color='black')
plt.show()
```



[231]:
```
fig= plt.figure(figsize=(8,5))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
#m0, b0 = np.polyfit(np.array(y_test), np.array(lab1), 1)
#plt.plot(np.array(y_test), m0*np.array(y_test) + b0,color='#426eff',linewidth=4)
#m1, b1 = np.polyfit(np.array(y_test), np.array(pred_lr), 1)
#plt.plot(np.array(y_test), m1*np.array(y_test) + b1,color='#513e00',linewidth=4)
#m2, b2 = np.polyfit(np.array(y_test), np.array(pred_svr), 1)
#plt.plot(np.array(y_test), m2*np.array(y_test) + b2,color='brown',linewidth=4)
#m3, b3 = np.polyfit(np.array(y_test), np.array(pred_rf), 1)
#plt.plot(np.array(y_test), m3*np.array(y_test) + b3,color='indigo',linewidth=4)
#m4, b4 = np.polyfit(np.array(y_test), np.array(pred_ann), 1)
#plt.plot(np.array(y_test), m4*np.array(y_test) + b4,color='tomato',linewidth=4)
```

```
#m5, b5 = np.polyfit(np.array(y_test), np.array(pred_xgb), 1)
#plt.plot(np.array(y_test), m5*np.array(y_test) +␣
↪b5,color='darkgoldenrod',linewidth=4)
plt.scatter(np.array(y_test),np.array(lab1),color='#426eff' )
plt.scatter(np.array(y_test),np.array(pred_lr),color='#513e00',marker='P')
plt.scatter(np.array(y_test),np.array(pred_svr),color='brown',marker='d')
plt.scatter(np.array(y_test),np.array(pred_rf),color='indigo',marker='*')
plt.scatter(np.array(y_test),np.array(pred_ann),color='tomato',marker='s')
plt.scatter(np.array(y_test),np.
↪array(pred_xgb),color='darkgoldenrod',marker='1')
ax.plot(y_test,y_test, c ="black",linewidth=5)
plt.xlabel('Reference O3(ppm)'.translate(subscript),fontsize=18)
plt.ylabel('Calibrated O3(ppm)'.translate(subscript),fontsize=18)
plt.legend(['1:1 Ref','LAB','LR','SVR','RF','ANN','XGBoost'
            ], loc = 2, bbox_to_anchor = (0,1))
#plt.title('CO Sensor',fontsize=18 )
plt.title(r'$ R^{2}(LR)=$'+str(R2_lr_NO2)+r'$,\ R^{2}(SVR)=$'+str(R2_svr_NO2)
          +r'$,\ R^{2}(RF)=$'+str(R2_rf_NO2)+ r'$,\␣
↪R^{2}(ANN)=$'+str(R2_ann_NO2)
          +r'$, \ R^{2}(XGBoost)=$'+str(R2_xgb_NO2),
          fontsize=10)
plt.grid(linestyle='-.',linewidth=0.1)
```



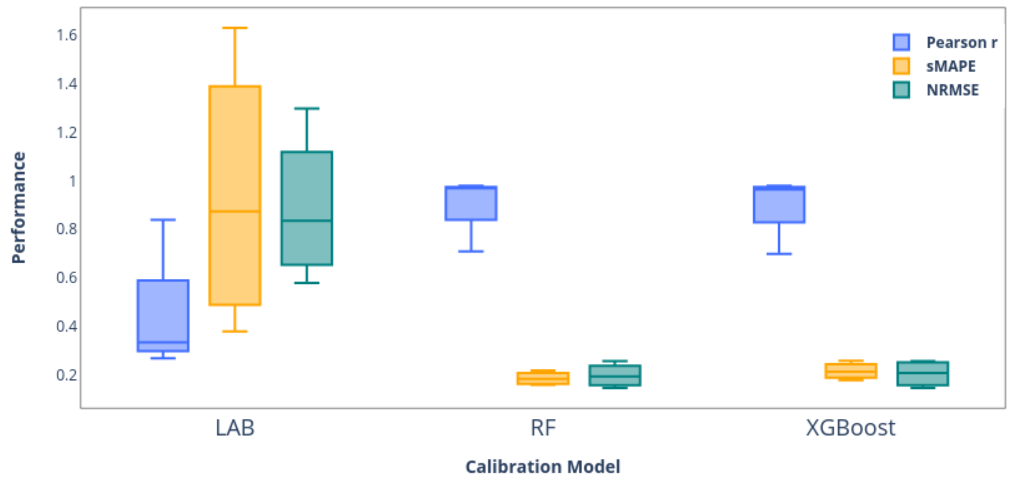
14 Data Analytics

[232]:
```python
import chart_studio.plotly
import plotly.express as px
from IPython.display import Image
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]
XGB_PR=[Pearson_xgb_CO,Pearson_xgb_NO2,Pearson_xgb_SO2,Pearson_xgb_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]
XGB_SM=[sMAPE_xgb_CO,sMAPE_xgb_NO2,sMAPE_xgb_SO2,sMAPE_xgb_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]
XGB_RM=[RMSE_xgb_CO,RMSE_xgb_NO2,RMSE_xgb_SO2,RMSE_xgb_O3]

PR=LAB_PR+RF_PR+XGB_PR
SM=LAB_SM+RF_SM+XGB_SM
RM=LAB_RM+RF_RM+XGB_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)]
x6=['XGBoost' for i in range(4)]


x=x1+x4+x6
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='<b>Pearson r</b>',
    marker_color='#426eff'
))

fig.add_trace(go.Box(
    y=SM,
    x=x,
    name='<b>sMAPE</b>',
    marker_color='orange'

))
fig.add_trace(go.Box(
    y=RM,
    x=x,
    name='<b>NRMSE</b>',
    marker_color='teal'

))

fig.update_layout(autosize=False,
    width=900,
    height=500,

  legend=dict(
    yanchor="bottom",
    y=0.75,
    xanchor="right",
    x=1),
    # group together boxes of the different
    # traces for each value of x
    boxmode='group',
                   plot_bgcolor='rgba(0,0,0,0)'
                  )

fig.update_xaxes(title_text="<b>Calibration Model</b>",tickfont =␣
 ↪dict(size=18),showgrid=False,showline=True,
                 linewidth=0.5, linecolor='black',mirror=True)
fig.update_yaxes(title_text="<b>Performance</b>",showgrid=False,showline=True,␣
 ↪linewidth=0.5, linecolor='black',
                 mirror=True)
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_boxplot.png')
```

```
Image('models_boxplot.png')
```

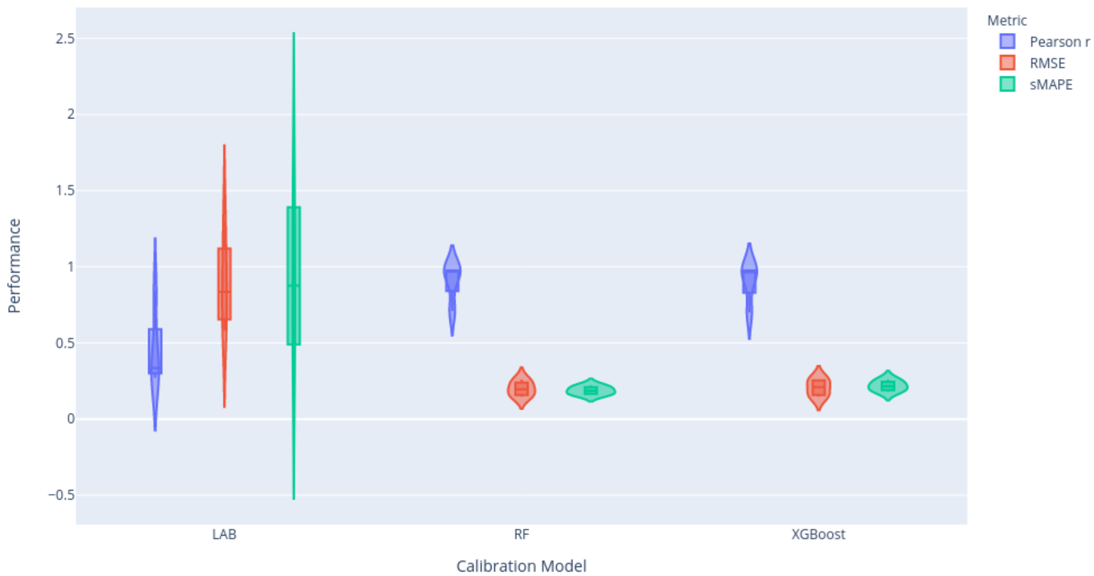
[232]:



[233]:
```
#Violin plot which also show the density of the distribution
import plotly.express as px
Metric1=['Pearson r' for i in range(len(PR))]
Metric2=['RMSE' for i in range(len(RM))]
Metric3=['sMAPE' for i in range(len(RM))]
Metric=Metric1+Metric2+Metric3
Model=x+x+x
Values=PR+RM+SM
lst=[[Model[i],Values[i],Metric[i]] for i in range(len(Model))]
df = pd.DataFrame(lst, columns =['Calibration Model', 'Performance','Metric'])

#fig = px.violin( df,y="Performance", x="Calibration Model", color='Metric',␣
↪box=True,points="all",
          #hover_data=df.columns)
fig = px.violin( df,y="Performance", x="Calibration Model", color='Metric',␣
↪box=True,
          hover_data=df.columns)


fig.update_layout(autosize=False,
    width=1000,
    height=600)
#fig.show()
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
```

```
chart_studio.plotly.image.save_as(fig, filename='models_violinplots.png')
Image('models_violinplots.png')
```

[233]:

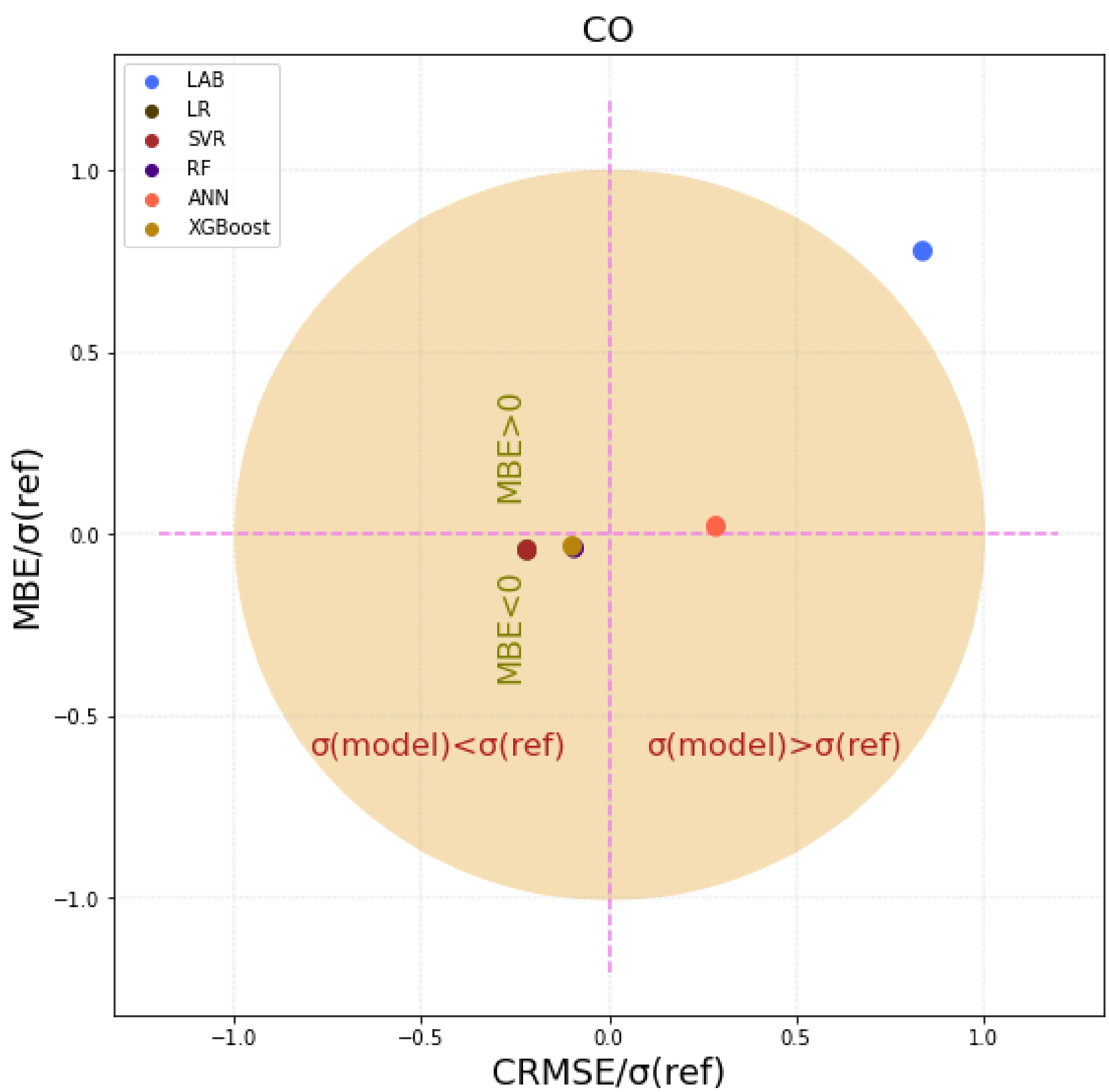


14.1 Target Diagrams

[234]:
```
theta = np.linspace( 0 , 2 * np.pi , 150 )

radius = 1

a = radius * np.cos( theta )
b = radius * np.sin( theta )
CRMSE_LAB=[CRMSE_LAB_O3,CRMSE_LAB_CO,CRMSE_LAB_NO2,CRMSE_LAB_SO2]
CRMSE_LR=[CRMSE_LR_O3,CRMSE_LR_CO,CRMSE_LR_NO2,CRMSE_LR_SO2]
CRMSE_SVR=[CRMSE_SVR_O3,CRMSE_SVR_CO,CRMSE_SVR_NO2,CRMSE_SVR_SO2]
CRMSE_RF=[CRMSE_RF_O3,CRMSE_RF_CO,CRMSE_RF_NO2,CRMSE_RF_SO2]
CRMSE_ANN=[CRMSE_ANN_O3,CRMSE_ANN_CO,CRMSE_ANN_NO2,CRMSE_ANN_SO2]
CRMSE_XGB=[CRMSE_XGB_O3,CRMSE_XGB_CO,CRMSE_XGB_NO2,CRMSE_XGB_SO2]

MBE_LAB=[MBE_LAB_O3,MBE_LAB_CO,MBE_LAB_NO2,MBE_LAB_SO2]
MBE_LR=[MBE_LR_O3,MBE_LR_CO,MBE_LR_NO2,MBE_LR_SO2]
MBE_SVR=[MBE_SVR_O3,MBE_SVR_CO,MBE_SVR_NO2,MBE_SVR_SO2]
MBE_RF=[MBE_RF_O3,MBE_RF_CO,MBE_RF_NO2,MBE_RF_SO2]
MBE_ANN=[MBE_ANN_O3,MBE_ANN_CO,MBE_ANN_NO2,MBE_ANN_SO2]
MBE_XGB=[MBE_XGB_O3,MBE_XGB_CO,MBE_XGB_NO2,MBE_XGB_SO2]
```

[235]:

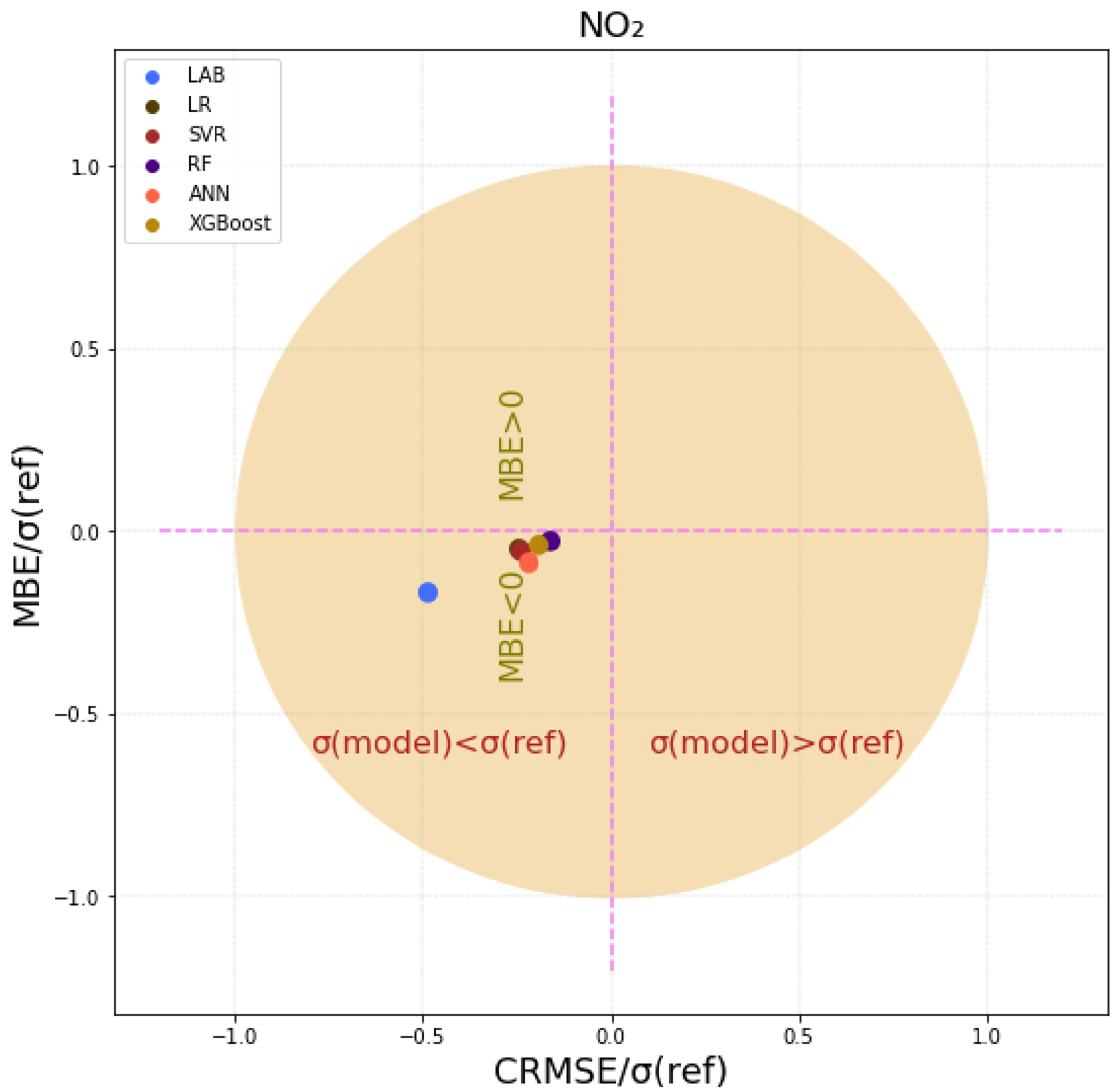
```
fig= plt.figure(figsize=(9,9))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
plt.text(CRMSE_LAB[0],MBE_LAB[0] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='#426eff')
plt.text(CRMSE_LR[0],MBE_LR[0] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='#513e00')
plt.text(CRMSE_SVR[0],MBE_SVR[0] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='brown')
plt.text(CRMSE_RF[0],MBE_RF[0] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='indigo')
plt.text(CRMSE_ANN[0],MBE_ANN[0] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='tomato')
plt.text(CRMSE_XGB[0],MBE_XGB[0] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='darkgoldenrod')
plt.scatter(CRMSE_LAB[2]/1.1,MBE_LAB[2],color='#426eff')
plt.scatter(CRMSE_LR,MBE_LR,color='#513e00')
plt.scatter(CRMSE_SVR[0],MBE_SVR[0],color='brown')
plt.scatter(CRMSE_RF,MBE_RF,color='indigo')
plt.scatter(CRMSE_ANN,MBE_ANN,color='tomato')
plt.scatter(CRMSE_XGB,MBE_XGB,color='darkgoldenrod')
plt.legend(['LAB','LR','SVR','RF','ANN','XGBoost'],loc = 2, bbox_to_anchor =␣
 ↪(0,1))

plt.Circle((0, 0), 1, color='wheat')
plt.vlines([0], -1.2, 1.2, linestyles='dashed',color='violet')
plt.hlines([0], -1.2, 1.2, linestyles='dashed', color='violet')
plt.text(-0.8, -0.6, ' (model)< (ref)', fontsize = 16, color='firebrick')
plt.text(0.1, -0.6, ' (model)> (ref)', fontsize = 16, color='firebrick')
plt.text(-0.3, -0.25, 'MBE<0', rotation=90, va='center',fontsize = 16,␣
 ↪color='olive')
plt.text(-0.3, 0.25, 'MBE>0', rotation=90, va='center',fontsize = 16,␣
 ↪color='olive')
plt.fill_between(a, b, color='wheat')
plt.xlabel('CRMSE/ (ref)',fontsize=18)
plt.ylabel('MBE/ (ref)',fontsize=18)
plt.title('CO',fontsize=18)
plt.grid(linestyle='-.',linewidth=0.3)
plt.show()
```



[236]:

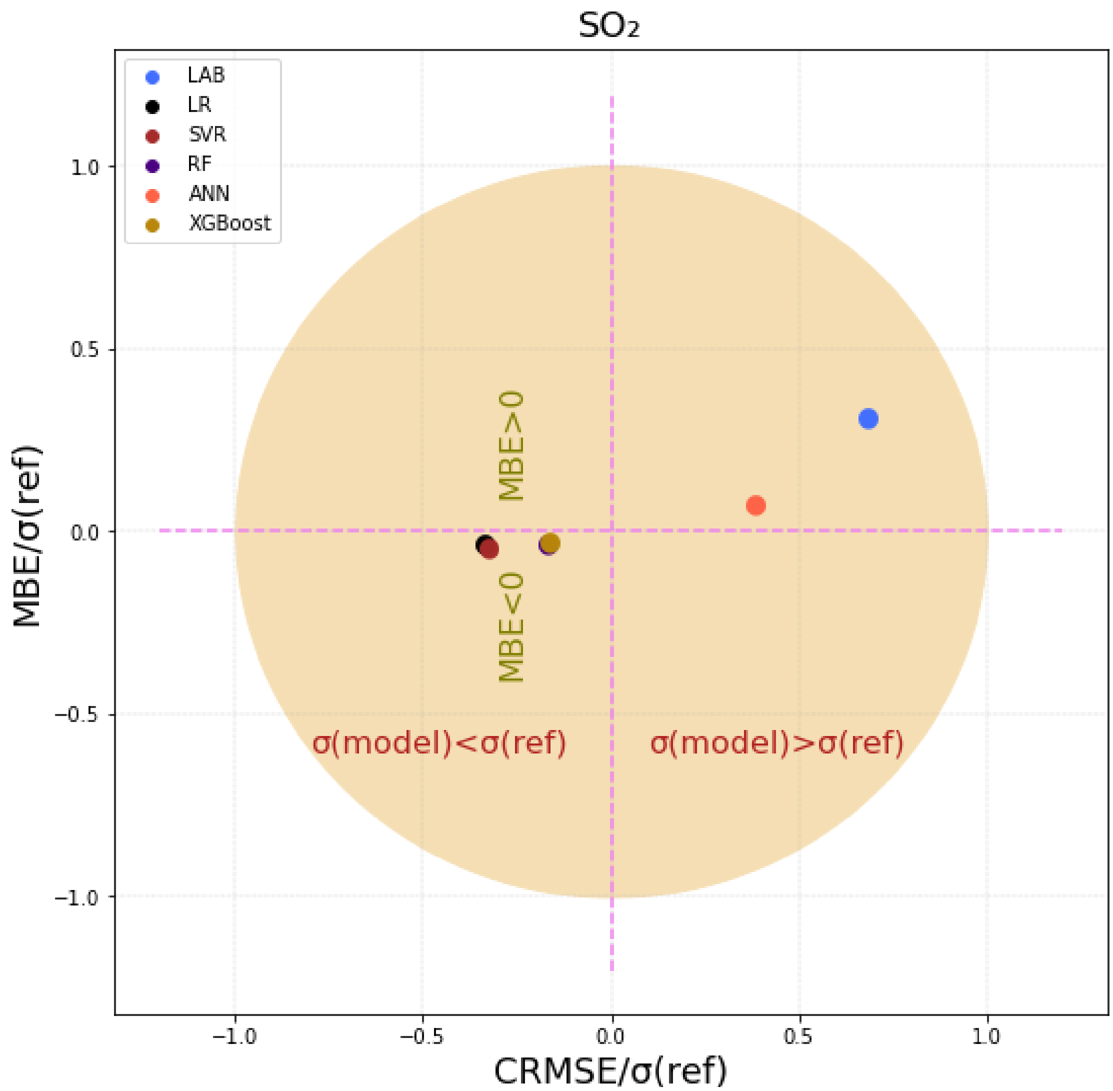
```
fig= plt.figure(figsize=(9,9))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
plt.text(CRMSE_LAB[1],MBE_LAB[1] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='#426eff')
plt.text(CRMSE_LR[1],MBE_LR[1] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='#513e00')
plt.text(CRMSE_SVR[1],MBE_SVR[1] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='brown')
plt.text(CRMSE_RF[1],MBE_RF[1] , '•', rotation=90, va='center',fontsize = 36,␣
 ↪color='indigo')
```

```
plt.text(CRMSE_ANN[1],MBE_ANN[1] , '•', rotation=90, va='center',fontsize = 36, color='tomato')
plt.text(CRMSE_XGB[1],MBE_XGB[1] , '•', rotation=90, va='center',fontsize = 36, color='darkgoldenrod')
plt.scatter(CRMSE_LAB[2]/1.1,MBE_LAB[2],color='#426eff')
plt.scatter(CRMSE_LR,MBE_LR,color='#513e00')
plt.scatter(CRMSE_SVR[0],MBE_SVR[0],color='brown')
plt.scatter(CRMSE_RF,MBE_RF,color='indigo')
plt.scatter(CRMSE_ANN,MBE_ANN,color='tomato')
plt.scatter(CRMSE_XGB,MBE_XGB,color='darkgoldenrod')
plt.legend(['LAB','LR','SVR','RF','ANN','XGBoost'],loc = 2, bbox_to_anchor = (0,1))
plt.Circle((0, 0), 1, color='wheat')
plt.vlines([0], -1.2, 1.2, linestyles='dashed',color='violet')
plt.hlines([0], -1.2, 1.2, linestyles='dashed', color='violet')
plt.text(-0.8, -0.6, ' (model)< (ref)', fontsize = 16, color='firebrick')
plt.text(0.1, -0.6, ' (model)> (ref)', fontsize = 16, color='firebrick')
plt.text(-0.3, -0.25, 'MBE<0', rotation=90, va='center',fontsize = 16, color='olive')
plt.text(-0.3, 0.25, 'MBE>0', rotation=90, va='center',fontsize = 16, color='olive')
plt.fill_between(a, b, color='wheat')
plt.xlabel('CRMSE/ (ref)',fontsize=18)
plt.ylabel('MBE/ (ref)',fontsize=18)
plt.title('NO2'.translate(subscript),fontsize=18)
plt.grid(linestyle='-.',linewidth=0.3)
plt.show()
```



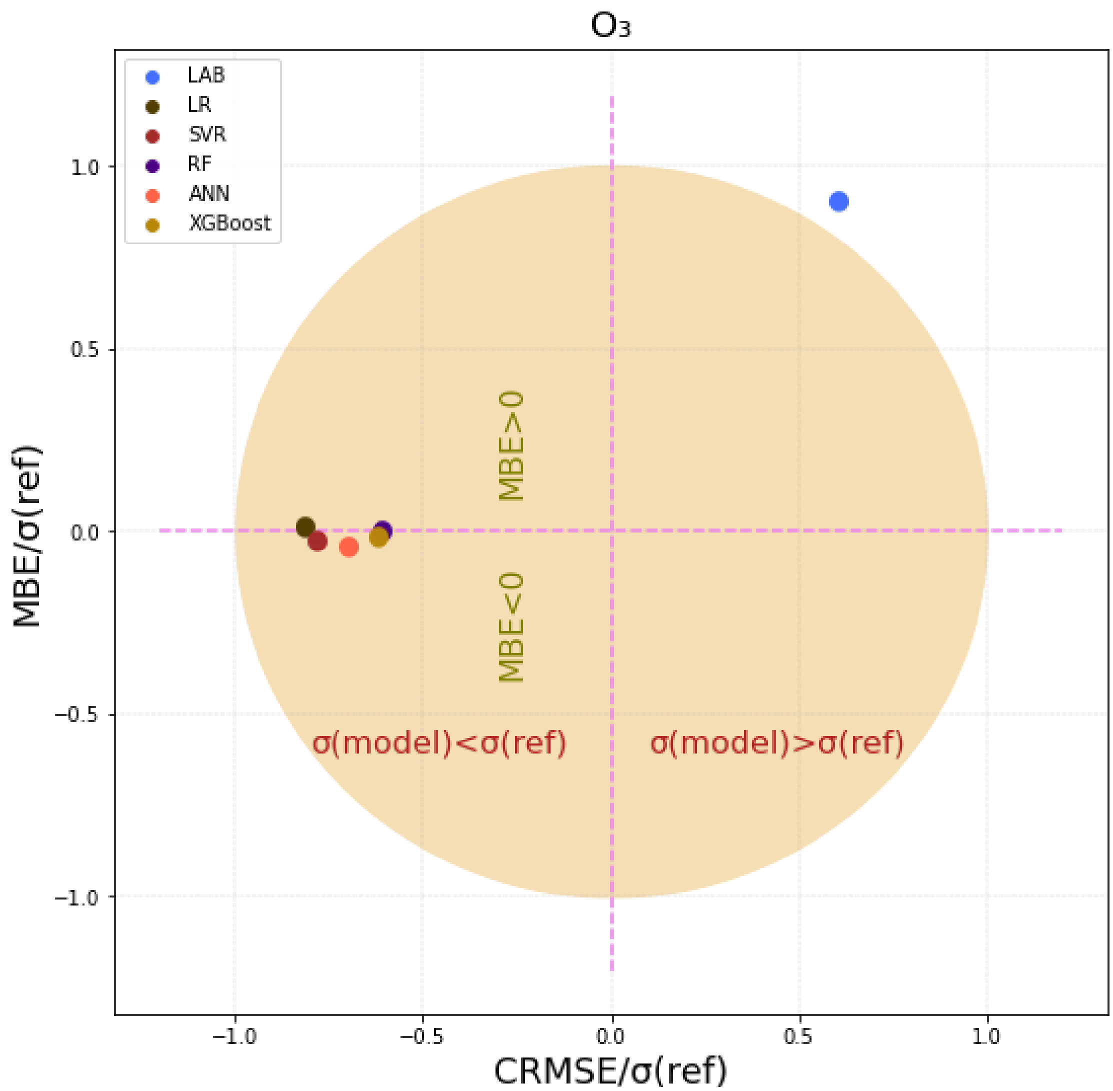
[237]:
```
fig= plt.figure(figsize=(9,9))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
plt.text(CRMSE_LAB[2],MBE_LAB[2] , '•', rotation=90, va='center',fontsize = 36,␣
↪color='#426eff')
plt.text(CRMSE_LR[2],MBE_LR[2] , '•', rotation=90, va='center',fontsize = 36,␣
↪color='black')
plt.text(CRMSE_SVR[2],MBE_SVR[2] , '•', rotation=90, va='center',fontsize = 36,␣
↪color='brown')
plt.text(CRMSE_RF[2],MBE_RF[2] , '•', rotation=90, va='center',fontsize = 36,␣
↪color='indigo')
```

```
plt.text(CRMSE_ANN[2],MBE_ANN[2] , '•', rotation=90, va='center',fontsize = 36, color='tomato')
plt.text(CRMSE_XGB[2],MBE_XGB[2] , '•', rotation=90, va='center',fontsize = 36, color='darkgoldenrod')
plt.scatter(CRMSE_LAB[2]/1.1,MBE_LAB[2],color='#426eff')
plt.scatter(CRMSE_LR,MBE_LR,color='black')
plt.scatter(CRMSE_SVR[0],MBE_SVR[0],color='brown')
plt.scatter(CRMSE_RF,MBE_RF,color='indigo')
plt.scatter(CRMSE_ANN,MBE_ANN,color='tomato')
plt.scatter(CRMSE_XGB,MBE_XGB,color='darkgoldenrod')
plt.legend(['LAB','LR','SVR','RF','ANN','XGBoost'],loc = 2, bbox_to_anchor = (0,1))
plt.Circle((0, 0), 1, color='wheat')
plt.vlines([0], -1.2, 1.2, linestyles='dashed',color='violet')
plt.hlines([0], -1.2, 1.2, linestyles='dashed', color='violet')
plt.text(-0.8, -0.6, ' (model)< (ref)', fontsize = 16, color='firebrick')
plt.text(0.1, -0.6, ' (model)> (ref)', fontsize = 16, color='firebrick')
plt.text(-0.3, -0.25, 'MBE<0', rotation=90, va='center',fontsize = 16, color='olive')
plt.text(-0.3, 0.25, 'MBE>0', rotation=90, va='center',fontsize = 16, color='olive')
plt.fill_between(a, b, color='wheat')
plt.xlabel('CRMSE/ (ref)',fontsize=18)
plt.ylabel('MBE/ (ref)',fontsize=18)
plt.title('SO2'.translate(subscript),fontsize=18)
plt.grid(linestyle='-.',linewidth=0.3)
plt.show()
```



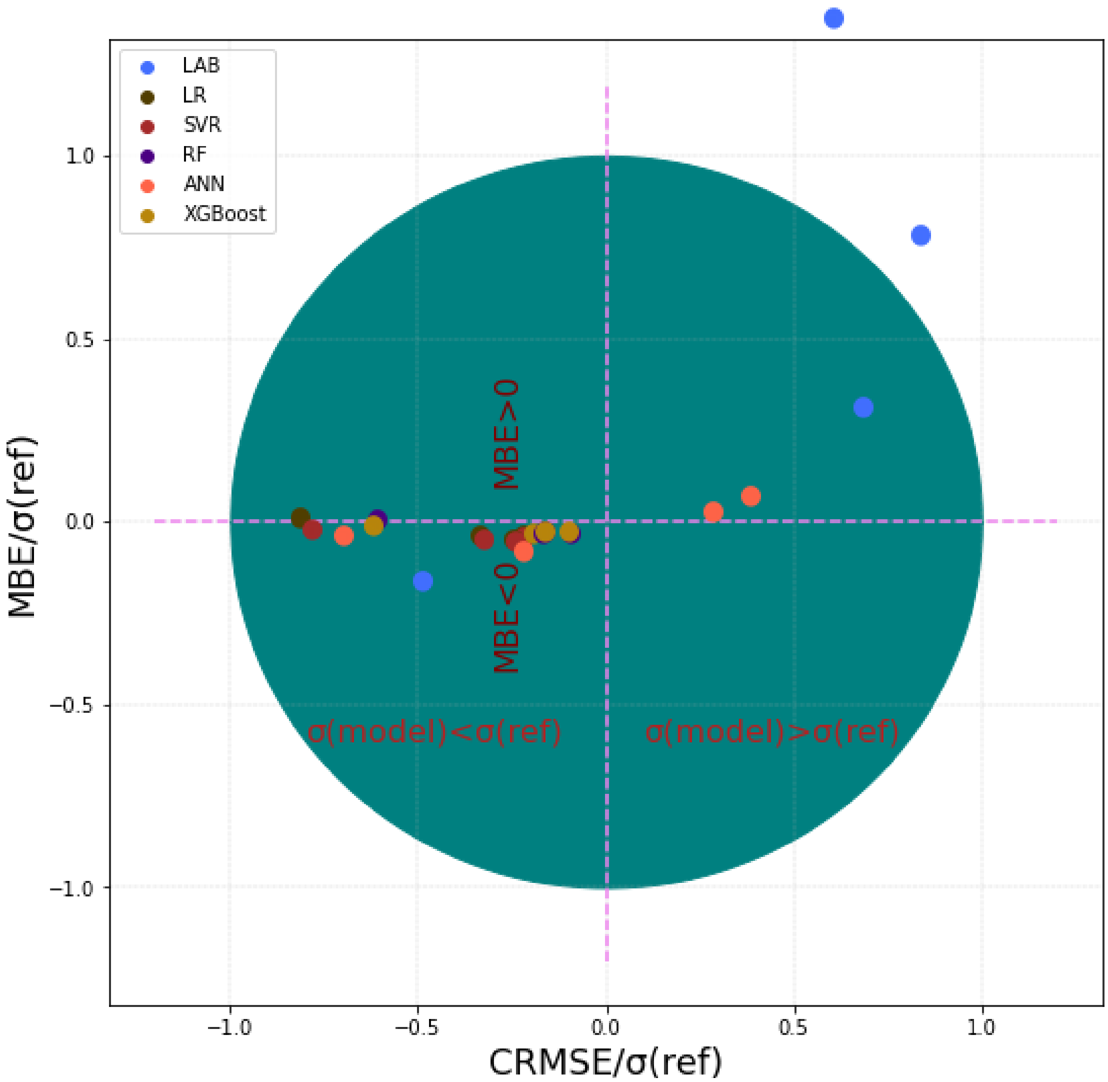
[238]:
```
fig= plt.figure(figsize=(9,9))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
plt.text(CRMSE_LAB[3],MBE_LAB[3]/1.5 , '•', rotation=90, va='center',fontsize =␣
↪36, color='#426eff')
plt.text(CRMSE_LR[3],MBE_LR[3] , '•', rotation=90, va='center',fontsize = 36,␣
↪color='#513e00')
plt.text(CRMSE_SVR[3],MBE_SVR[3] , '•', rotation=90, va='center',fontsize = 36,␣
↪color='brown')
plt.text(CRMSE_RF[3],MBE_RF[3] , '•', rotation=90, va='center',fontsize = 36,␣
↪color='indigo')
```

```
plt.text(CRMSE_ANN[3],MBE_ANN[3] , '•', rotation=90, va='center',fontsize = 36, color='tomato')
plt.text(CRMSE_XGB[3],MBE_XGB[3] , '•', rotation=90, va='center',fontsize = 36, color='darkgoldenrod')
plt.scatter(CRMSE_LAB[2]/1.1,MBE_LAB[2]/1.7,color='#426eff')
plt.scatter(CRMSE_LR,MBE_LR,color='#513e00')
plt.scatter(CRMSE_SVR[0],MBE_SVR[0],color='brown')
plt.scatter(CRMSE_RF,MBE_RF,color='indigo')
plt.scatter(CRMSE_ANN,MBE_ANN,color='tomato')
plt.scatter(CRMSE_XGB,MBE_XGB,color='darkgoldenrod')
plt.legend(['LAB','LR','SVR','RF','ANN','XGBoost'],loc = 2, bbox_to_anchor = (0,1))
plt.Circle((0, 0), 1, color='wheat')
plt.vlines([0], -1.2, 1.2, linestyles='dashed',color='violet')
plt.hlines([0], -1.2, 1.2, linestyles='dashed', color='violet')
plt.text(-0.8, -0.6, ' (model)< (ref)', fontsize = 16, color='firebrick')
plt.text(0.1, -0.6, ' (model)> (ref)', fontsize = 16, color='firebrick')
plt.text(-0.3, -0.25, 'MBE<0', rotation=90, va='center',fontsize = 16, color='olive')
plt.text(-0.3, 0.25, 'MBE>0', rotation=90, va='center',fontsize = 16, color='olive')
plt.fill_between(a, b, color='wheat')
plt.xlabel('CRMSE/ (ref)',fontsize=18)
plt.ylabel('MBE/ (ref)',fontsize=18)
plt.title('O3'.translate(subscript),fontsize=18)
plt.grid(linestyle='-.',linewidth=0.3)
plt.show()
```



[239]:
```
fig= plt.figure(figsize=(9,9))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
for i in range(4):
    plt.text(CRMSE_LAB[i],MBE_LAB[i] , '•', rotation=90, va='center',fontsize =
 ↪36, color='#426eff')
    plt.text(CRMSE_LR[i],MBE_LR[i] , '•', rotation=90, va='center',fontsize =
 ↪36, color='#513e00')
    plt.text(CRMSE_SVR[i],MBE_SVR[i] , '•', rotation=90, va='center',fontsize =
 ↪36, color='brown')
    plt.text(CRMSE_RF[i],MBE_RF[i] , '•', rotation=90, va='center',fontsize =
 ↪36, color='indigo')
```

```
    plt.text(CRMSE_ANN[i],MBE_ANN[i] , '•', rotation=90, va='center',fontsize =␣
↪36, color='tomato')
    plt.text(CRMSE_XGB[i],MBE_XGB[i] , '•', rotation=90, va='center',fontsize =␣
↪36, color='darkgoldenrod')
plt.scatter(CRMSE_LAB[2]-0.2,MBE_LAB[2],color='#426eff')
plt.scatter(CRMSE_LR,MBE_LR,color='#513e00')
plt.scatter(CRMSE_SVR,MBE_SVR,color='brown')
plt.scatter(CRMSE_RF,MBE_RF,color='indigo')
plt.scatter(CRMSE_ANN,MBE_ANN,color='tomato')
plt.scatter(CRMSE_XGB,MBE_XGB,color='darkgoldenrod')
plt.legend(['LAB','LR','SVR','RF','ANN','XGBoost'],loc = 2, bbox_to_anchor =␣
↪(0,1))
plt.vlines([0], -1.2, 1.2, linestyles='dashed',color='violet')
plt.hlines([0], -1.2, 1.2, linestyles='dashed', color='violet')
plt.text(-0.8, -0.6, ' (model)< (ref)', fontsize = 16, color='firebrick')
plt.text(0.1, -0.6, ' (model)> (ref)', fontsize = 16, color='firebrick')
plt.text(-0.3, -0.25, 'MBE<0', rotation=90, va='center',fontsize = 16,␣
↪color='maroon')
plt.text(-0.3, 0.25, 'MBE>0', rotation=90, va='center',fontsize = 16,␣
↪color='maroon')
plt.fill_between(a, b, color='teal')
plt.xlabel('CRMSE/ (ref)',fontsize=18)
plt.ylabel('MBE/ (ref)',fontsize=18)
#plt.title('Overal',fontsize=18)
plt.grid(linestyle='-.',linewidth=0.3)
plt.show()
```



14.2 Feature Importance

15 libraries

import numpy as np import matplotlib.pyplot as plt fig= plt.figure(figsize=(10,5)) # set width of bars ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0.3) barWidth = 0.15

16 set heights of bars

Signal1 = [features_CO[0],features_NO2[0],features_SO2[0],features_O3[0]] Signal2 = [features_CO[1],features_NO2[1],features_SO2[1],features_O3[1]] Signal3 = [features_CO[2],features_NO2[2],features_SO2[2],features_O3[2]] Sig-

nal4 = [features_CO[3],features_NO2[3],features_SO2[3],features_O3[3]] Temp = [features_CO[4],features_NO2[4],features_SO2[4],features_O3[4]] RH = [features_CO[5],features_NO2[5],features_SO2[5],features_O3[5]]

17 Set position of bar on X axis

r1 = np.arange(len(Signal1)) r2 = [x + barWidth for x in r1] r3 = [x + barWidth for x in r2] r4 = [x + barWidth for x in r3] r5 = [x + barWidth for x in r4] r6 = [x + barWidth for x in r5]

18 Make the plot

plt.bar(r1, Signal1, color=‘magenta’, width=barWidth, edgecolor=‘white’, label=‘CO’) plt.bar(r2, Signal2, color=‘teal’, width=barWidth, edgecolor=‘white’, label=‘NO2’) plt.bar(r3, Signal3, color=‘salmon’, width=barWidth, edgecolor=‘white’, label=‘SO2’) plt.bar(r4, Signal4, color=‘rebeccapurple’, width=barWidth, edgecolor=‘white’, label=‘O3’) plt.bar(r5, Temp, color=‘olive’, width=barWidth, edgecolor=‘white’, label=‘Temperature’) plt.bar(r6, RH, color=‘darkgoldenrod’, width=barWidth, edgecolor=‘white’, label=‘RH’)

19 Add xticks on the middle of the group bars

plt.xlabel(‘Sensor’, fontweight=‘bold’) plt.ylabel(‘Feature Importance’, fontweight=‘bold’) plt.xticks([r + barWidth+0.25 for r in range(len(Signal1))], [‘CO’, ‘NO2’, ‘SO2’, ‘O3’])

20 Create legend & Show graphic

plt.legend() plt.grid(linestyle=‘-.’,linewidth=0.3) plt.show()

[240]:
```
import plotly.graph_objects as go
import pandas as pd
model=['<b>LAB</b>','<b>LR</b>','<b>SVR</b>','<b>RF</b>','<b>ANN</
↪b>','<b>XGBoost</b>']
Pearson=[Pearson_lab_CO,Pearson_lr_CO,Pearson_svr_CO,Pearson_rf_CO,Pearson_ann_CO,Pearson_xgb_

R2=[R2_lab_CO,R2_lr_CO,R2_svr_CO,R2_rf_CO,R2_ann_CO,R2_xgb_CO]
RMSE=[RMSE_Lab_CO,RMSE_Lr_CO,RMSE_Svr_CO,RMSE_Rf_CO,RMSE_Ann_CO,RMSE_Xgb_CO]
sMAPE=[sMAPE_lab_CO,sMAPE_lr_CO,sMAPE_svr_CO,sMAPE_rf_CO,sMAPE_ann_CO,sMAPE_xgb_CO]

fig = go.Figure(data=[go.Table(
    header=dict(values=['<b>Model</b>','<b>Pearson r</b>','<b>R^2</
↪b>','<b>sMAPE</b>','<b>RMSE(ppb)</b>'],
                #fill_color='white',
                align='left'),
    cells=dict(values=[model,Pearson,R2,sMAPE,RMSE ],
               #fill_color='white',
               align='left'))
   ])
```

```
fig.update_layout(
title={'text': "<b>CO Calibration : Model Performance</b>",
        'y':0.86,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
width=700,
height=600,
)
fig.show()
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_performance_CO.png')
Image('models_performance_CO.png')
```

[240]:

CO Calibration : Model Performance

Model	Pearson r	R^2	sMAPE	RMSE(ppb)
LAB	0.84	0.64	0.38	256.6
LR	0.94	0.89	0.22	145.9
SVR	0.94	0.89	0.21	144.6
RF	0.97	0.93	0.16	110.8
ANN	0.95	0.9	0.21	137.2
XGBoost	0.96	0.92	0.2	124

[241]:

```
import plotly.graph_objects as go
import pandas as pd
```

```
model=['<b>LAB</b>','<b>LR</b>','<b>SVR</b>','<b>RF</b>','<b>ANN</
↪b>','<b>XGBoost</b>']
Pearson=[Pearson_lab_NO2,Pearson_lr_NO2,Pearson_svr_NO2,Pearson_rf_NO2,Pearson_ann_NO2,Pearson
R2=[R2_lab_NO2,R2_lr_NO2,R2_svr_NO2,R2_rf_NO2,R2_ann_NO2,R2_xgb_NO2]
RMSE=[RMSE_Lab_NO2,RMSE_Lr_NO2,RMSE_Svr_NO2,RMSE_Rf_NO2,RMSE_Ann_NO2,RMSE_Xgb_NO2]
sMAPE=[sMAPE_lab_NO2,sMAPE_lr_NO2,sMAPE_svr_NO2,sMAPE_rf_NO2,sMAPE_ann_NO2,sMAPE_xgb_NO2]

fig = go.Figure(data=[go.Table(
    header=dict(values=['<b>Model</b>','<b>Pearson r</b>','<b>R^2</
↪b>','<b>sMAPE</b>','<b>RMSE</b>'],
                #fill_color='paleturquoise',
                align='left'),
    cells=dict(values=[model,Pearson,R2,sMAPE,RMSE ],
               #fill_color='lavender',
               align='left'))
   ])
fig.update_layout(
title={'text': "<b>NO2 Calibration : Model Performance</b>".
↪translate(subscript),
        'y':0.86,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
width=700,
height=600,

)
fig.show()
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_performance_NO2.png')
Image('models_performance_NO2.png')
```

[241]:

NO_2 Calibration : Model Performance

Model	**Pearson r**	**R^2**	**sMAPE**	**RMSE**
LAB	0.34	-2.14	0.6	21.2
LR	0.9	0.81	0.33	5.2
SVR	0.91	0.82	0.28	5
RF	0.97	0.93	0.17	3.1
ANN	0.96	0.9	0.19	3.7
XGBoost	0.97	0.93	0.18	3.1

[242]:

```
import plotly.graph_objects as go
import pandas as pd
model=['<b>LAB</b>','<b>LR</b>','<b>SVR</b>','<b>RF</b>','<b>ANN</
↪b>','<b>XGBoost</b>']
Pearson=[Pearson_lab_SO2,Pearson_lr_SO2,Pearson_svr_SO2,Pearson_rf_SO2,Pearson_ann_SO2,Pearson
R2=[R2_lab_SO2,R2_lr_SO2,R2_svr_SO2,R2_rf_SO2,R2_ann_SO2,R2_xgb_SO2,]
RMSE=[RMSE_Lab_SO2,RMSE_Lr_SO2,RMSE_Svr_SO2,RMSE_Rf_SO2,RMSE_Ann_SO2,RMSE_Xgb_SO2]
sMAPE=[sMAPE_lab_SO2,sMAPE_lr_SO2,sMAPE_svr_SO2,sMAPE_rf_SO2,sMAPE_ann_SO2,sMAPE_xgb_SO2]

fig = go.Figure(data=[go.Table(
    header=dict(values=['<b>Model</b>','<b>Pearson r</b>','<b>R^2</
↪b>','<b>sMAPE</b>','<b>RMSE</b>'],
                #fill_color='paleturquoise',
                align='left'),
    cells=dict(values=[model,Pearson,R2,sMAPE,RMSE ],
               #fill_color='lavender',
               align='left'))
```

```
    ])
fig.update_layout(
title={'text': "<b>SO2 Calibration : Model Performance</b>".
 ↪translate(subscript),
        'y':0.86,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
width=700,
height=600,

)
fig.show()
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_performance_SO2.png')
Image('models_performance_SO2.png')
```

[242]:

SO_2 Calibration : Model Performance

Model	Pearson r	R^2	sMAPE	RMSE
LAB	0.33	-2390.42	1.63	29.9
LR	0.43	0.17	0.28	0.6
SVR	0.48	0.23	0.27	0.5
RF	0.71	0.5	0.2	0.4
ANN	0.61	0.37	0.24	0.5
XGBoost	0.7	0.49	0.23	0.4

[243]:

```
import plotly.graph_objects as go
import pandas as pd
model=['<b>LAB</b>','<b>LR</b>','<b>SVR</b>','<b>RF</b>','<b>ANN</
↪b>','<b>XGBoost</b>']
Pearson=[str(Pearson_lab_O3),Pearson_lr_O3,Pearson_svr_O3,Pearson_rf_O3,Pearson_ann_O3,Pearson
R2=[R2_lab_O3,R2_lr_O3,R2_svr_O3,R2_rf_O3,R2_ann_O3,R2_xgb_O3]
RMSE=[RMSE_Lab_O3,RMSE_Lr_O3,RMSE_Svr_O3,RMSE_Rf_O3,RMSE_Ann_O3,RMSE_Xgb_O3]
sMAPE=[sMAPE_lab_O3,sMAPE_lr_O3,sMAPE_svr_O3,sMAPE_rf_O3,sMAPE_ann_O3,sMAPE_xgb_O3]

fig = go.Figure(data=[go.Table(
    header=dict(values=['<b>Model</b>','<b>Pearson r</b>','<b>R^2</
↪b>','<b>sMAPE</b>','<b>RMSE</b>'],
                #fill_color='paleturquoise',
                align='left'),
    cells=dict(values=[model,Pearson,R2,sMAPE,RMSE ],
               #fill_color='lavender',
               align='left'))
   ])
fig.update_layout(
title={#'text': "<b>O3 Calibration : Model Performance</b>".
↪translate(subscript),
        'y':0.86,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
width=700,
height=600,

)
fig.show()
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_performance_O3.png')
Image('models_performance_O3.png')
```

[243]:

Model	Pearson r	R^2	sMAPE	RMSE
LAB	0.27	-14.65	1.15	58.6
LR	0.95	0.9	0.44	4.6
SVR	0.95	0.9	0.45	4.6
RF	0.98	0.96	0.22	2.8
ANN	0.98	0.96	0.2	3
XGBoost	0.98	0.96	0.26	2.8

[244]:
```
Pearson_lab=np.
 ↪array([Pearson_lab_CO,Pearson_lab_NO2,Pearson_lab_SO2,Pearson_lab_O3])
Pearson_lr=np.array([Pearson_lr_CO,Pearson_lr_NO2,Pearson_lr_SO2,Pearson_lr_O3])
Pearson_svr=np.
 ↪array([Pearson_svr_CO,Pearson_svr_NO2,Pearson_svr_SO2,Pearson_svr_O3])
Pearson_rf=np.array([Pearson_rf_CO,Pearson_rf_NO2,Pearson_rf_SO2,Pearson_rf_O3])
Pearson_ann=np.
 ↪array([Pearson_ann_CO,Pearson_ann_NO2,Pearson_ann_SO2,Pearson_ann_O3])
Pearson_xgb=np.
 ↪array([Pearson_xgb_CO,Pearson_xgb_NO2,Pearson_xgb_SO2,Pearson_xgb_O3])
R2_lab=np.array([R2_lab_CO,R2_lab_NO2,R2_lab_SO2,R2_lab_O3])
R2_lr=np.array([R2_lr_CO,R2_lr_NO2,R2_lr_SO2,R2_lr_O3])
R2_svr=np.array([R2_svr_CO,R2_svr_NO2,R2_svr_SO2,R2_svr_O3])
R2_rf=np.array([R2_rf_CO,R2_rf_NO2,R2_rf_SO2,R2_rf_O3])
R2_ann=np.array([R2_ann_CO,R2_ann_NO2,R2_ann_SO2,R2_ann_O3])
R2_xgb=np.array([R2_xgb_CO,R2_xgb_NO2,R2_xgb_SO2,R2_xgb_O3])
RMSE_lab=np.array([RMSE_Lab_CO,RMSE_Lab_NO2,RMSE_Lab_SO2,RMSE_Lab_O3])
```

```
RMSE_lr=np.array([RMSE_Lr_CO,RMSE_Lr_NO2,RMSE_Lr_SO2,RMSE_Lr_O3])
RMSE_svr=np.array([RMSE_Svr_CO,RMSE_Svr_NO2,RMSE_Svr_SO2,RMSE_Svr_O3])
RMSE_rf=np.array([RMSE_Rf_CO,RMSE_Rf_NO2,RMSE_Rf_SO2,RMSE_Rf_O3])
RMSE_ann=np.array([RMSE_Ann_CO,RMSE_Ann_NO2,RMSE_Ann_SO2,RMSE_Ann_O3])
RMSE_xgb=np.array([RMSE_Xgb_CO,RMSE_Xgb_NO2,RMSE_Xgb_SO2,RMSE_Xgb_O3])
sMAPE_lab=np.array([sMAPE_lab_CO,sMAPE_lab_NO2,sMAPE_lab_SO2,sMAPE_lab_O3])
sMAPE_lr=np.array([sMAPE_lr_CO,sMAPE_lr_NO2,sMAPE_lr_SO2,sMAPE_lr_O3])
sMAPE_svr=np.array([sMAPE_svr_CO,sMAPE_svr_NO2,sMAPE_svr_SO2,sMAPE_svr_O3])
sMAPE_rf=np.array([sMAPE_rf_CO,sMAPE_rf_NO2,sMAPE_rf_SO2,sMAPE_rf_O3])
sMAPE_ann=np.array([sMAPE_ann_CO,sMAPE_ann_NO2,sMAPE_ann_SO2,sMAPE_ann_O3])
sMAPE_xgb=np.array([sMAPE_xgb_CO,sMAPE_xgb_NO2,sMAPE_xgb_SO2,sMAPE_xgb_O3])
Pearson_lab_mean=round(np.mean(Pearson_lab),2)
Pearson_lab_std=round(np.std(Pearson_lab),2)
Pearson_lr_mean=round(np.mean(Pearson_lr),2)
Pearson_lr_std=round(np.std(Pearson_lr),2)
Pearson_svr_mean=round(np.mean(Pearson_svr),2)
Pearson_svr_std=round(np.std(Pearson_svr),2)
Pearson_rf_mean=round(np.mean(Pearson_rf),2)
Pearson_rf_std=round(np.std(Pearson_rf),2)
Pearson_ann_mean=round(np.mean(Pearson_ann),2)
Pearson_ann_std=round(np.std(Pearson_ann),2)
Pearson_xgb_mean=round(np.mean(Pearson_xgb),2)
Pearson_xgb_std=round(np.std(Pearson_xgb),2)
R2_lab_mean=round(np.mean(R2_lab),2)
R2_lab_std=round(np.std(R2_lab),2)
R2_lr_mean=round(np.mean(R2_lr),2)
R2_lr_std=round(np.std(R2_lr),2)
R2_svr_mean=round(np.mean(R2_svr),2)
R2_svr_std=round(np.std(R2_svr),2)
R2_rf_mean=round(np.mean(R2_rf),2)
R2_rf_std=round(np.std(R2_rf),2)
R2_ann_mean=round(np.mean(R2_ann),2)
R2_ann_std=round(np.std(R2_ann),2)
R2_xgb_mean=round(np.mean(R2_xgb),2)
R2_xgb_std=round(np.std(R2_xgb),2)
RMSE_lab_mean=round(np.mean(RMSE_lab),2)
RMSE_lab_std=round(np.std(RMSE_lab),2)
RMSE_lr_mean=round(np.mean(RMSE_lr),2)
RMSE_lr_std=round(np.std(RMSE_lr),2)
RMSE_svr_mean=round(np.mean(RMSE_svr),2)
RMSE_svr_std=round(np.std(RMSE_svr),2)
RMSE_rf_mean=round(np.mean(RMSE_rf),2)
RMSE_rf_std=round(np.std(RMSE_rf),2)
RMSE_ann_mean=round(np.mean(RMSE_ann),2)
RMSE_ann_std=round(np.std(RMSE_ann),2)
RMSE_xgb_mean=round(np.mean(RMSE_xgb),2)
RMSE_xgb_std=round(np.std(RMSE_xgb),2)
```

```
sMAPE_lab_mean=round(np.mean(sMAPE_lab),2)
sMAPE_lab_std=round(np.std(sMAPE_lab),2)
sMAPE_lr_mean=round(np.mean(sMAPE_lr),2)
sMAPE_lr_std=round(np.std(sMAPE_lr),2)
sMAPE_svr_mean=round(np.mean(sMAPE_svr),2)
sMAPE_svr_std=round(np.std(sMAPE_svr),2)
sMAPE_rf_mean=round(np.mean(sMAPE_rf),2)
sMAPE_rf_std=round(np.std(sMAPE_rf),2)
sMAPE_ann_mean=round(np.mean(sMAPE_ann),2)
sMAPE_ann_std=round(np.std(sMAPE_ann),2)
sMAPE_xgb_mean=round(np.mean(sMAPE_xgb),2)
sMAPE_xgb_std=round(np.std(sMAPE_xgb),2)
```

[245]:

```
import plotly.graph_objects as go
import pandas as pd
model=['<b>LAB</b>','<b>LR</b>','<b>SVR</b>','<b>RF</b>','<b>ANN</
 ↪b>','<b>XGBoost</b>']
Pearson=[str(Pearson_lab_mean)+ '±' +str(Pearson_lab_std),str(Pearson_lr_mean)+␣
 ↪'±' +str(Pearson_lr_std),
         str(Pearson_svr_mean)+ '±' +str(Pearson_svr_std),str(Pearson_rf_mean)+␣
 ↪'±' +str(Pearson_rf_std),
         str(Pearson_ann_mean)+ '±'␣
 ↪+str(Pearson_ann_std),str(Pearson_xgb_mean)+ '±' +str(Pearson_xgb_std)]


R2=[str(R2_lab_mean)+ '±' +str(R2_lab_std),str(R2_lr_mean)+ '±' +str(R2_lr_std),
    str(R2_svr_mean)+ '±' +str(R2_svr_std),str(R2_rf_mean)+ '±' +str(R2_rf_std),
    str(R2_ann_mean)+ '±' +str(R2_ann_std),str(R2_xgb_mean)+ '±'␣
 ↪+str(R2_xgb_std)]



RMSE=[str(RMSE_lab_mean)+ '±' +str(RMSE_lab_std),str(RMSE_lr_mean)+ '±'␣
 ↪+str(RMSE_lr_std),
      str(RMSE_svr_mean)+ '±' +str(RMSE_svr_std),str(RMSE_rf_mean)+ '±'␣
 ↪+str(RMSE_rf_std),
     str(RMSE_ann_mean)+ '±' +str(RMSE_ann_std),str(RMSE_xgb_mean)+ '±'␣
 ↪+str(RMSE_xgb_std)]


sMAPE=[str(sMAPE_lab_mean)+ '±' +str(sMAPE_lab_std),str(sMAPE_lr_mean)+ '±'␣
 ↪+str(sMAPE_lr_std),
       str(sMAPE_svr_mean)+ '±' +str(sMAPE_svr_std),str(sMAPE_rf_mean)+ '±'␣
 ↪+str(sMAPE_rf_std),
       str(sMAPE_ann_mean)+ '±' +str(sMAPE_ann_std),str(sMAPE_xgb_mean)+ '±'␣
 ↪+str(sMAPE_xgb_std)]
```

```
fig = go.Figure(data=[go.Table(
    header=dict(values=['<b>Model</b>','<b>Pearson r ±  </b>','<b>R^2 ±  </b>␣
 ↪','<b>sMAPE ±  </b>','<b>RMSE ±  </b>'],
                #fill_color='paleturquoise',
                align='left'),
    cells=dict(values=[model,Pearson,R2,sMAPE,RMSE ],
               #fill_color='lavender',
               align='left'))
   ])
fig.update_layout(
title={'text': "<b>Model Performance</b>",
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
width=900,
height=800,

)
fig.show()
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_performance_03.png')
Image('models_performance_03.png')
```

[245]:

Model Performance

Model	Pearson r ± σ	R^2 ± σ	sMAPE ± σ	RMSE ± σ
LAB	0.44±0.23	-601.64±1032.77	0.94±0.49	91.58±96.28
LR	0.8±0.22	0.69±0.3	0.32±0.08	39.07±61.7
SVR	0.82±0.2	0.71±0.28	0.3±0.09	38.67±61.18
RF	0.91±0.11	0.83±0.19	0.19±0.02	29.28±47.08
ANN	0.88±0.15	0.78±0.24	0.21±0.02	36.1±58.38
XGBoost	0.9±0.12	0.82±0.19	0.22±0.03	32.58±52.79

import seaborn as sns #+list(features_NO2)+list(features_SO2)+list(features_O3) fi=list(features_CO) fi=list(100*np.array(fi)) #+['NO2'.translate(subscript) for i in range(7)]+ #['SO2'.translate(subscript) for i in range(7)]+['O3'.translate(subscript) for i in range(7)] pollutants=(['CO' for i in range(7)]) feature=['Net signal', 'Temperature', 'RH','Month','Day','Day of week','Hour'] #+feature+feature+feature features=feature

data=[[fi[i],pollutants[i],features[i]] for i in range(len(fi))]

df=pd.DataFrame(data=data, columns=['fi','pollutant','features']) fig= plt.figure(figsize=(8,5)) ax = fig.add_subplot(111) ax.patch.set_facecolor('lightblue') ax.patch.set_alpha(0) percentage=fi ax=sns.barplot(x="pollutant", y="fi", hue="features", data=df, palette=['gold','rebeccapurple','salmon','palevioletred','steelblue','darkkhaki','plum']) patches = ax.patches for i in range(len(patches)): x = patches[i].get_x() + patches[i].get_width()/2 y = patches[i].get_height()+.05 ax.annotate('{:.1f}%'.format(percentage[i]), (x, y), ha='center') plt.ylabel('Feature importance (%)') plt.xlabel(' ') plt.legend(loc = 2, bbox_to_anchor = (1,1)) #plt.grid(True) plt.show()

fi=list(features_NO2) fi=list(100*np.array(fi)) #+['NO2'.translate(subscript) for i in range(7)]+

#[‘SO2’.translate(subscript) for i in range(7)]+[‘O3’.translate(subscript) for i in range(7)] pollutants=([‘NO2’.translate(subscript) for i in range(7)]) feature=[‘Net signal’, ‘Temperature’, ‘RH’,‘Month’,‘Day’,‘Day of week’,‘Hour’] #+feature+feature+feature features=feature

data=[[fi[i],pollutants[i],features[i]] for i in range(len(fi))]

df=pd.DataFrame(data=data, columns=[‘fi’,‘pollutant’,‘features’]) fig= plt.figure(figsize=(8,5)) ax = fig.add_subplot(111) ax.patch.set_facecolor(‘lightblue’) ax.patch.set_alpha(0) percentage=fi ax=sns.barplot(x=“pollutant”, y=“fi”, hue=“features”, data=df, palette=[‘gold’,‘rebeccapurple’,‘salmon’,‘palevioletred’,‘steelblue’,‘darkkhaki’,‘plum’]) ax.legend_.remove() patches = ax.patches for i in range(len(patches)): x = patches[i].get_x() + patches[i].get_width()/2 y = patches[i].get_height()+.05 ax.annotate(‘{:.1f}%’.format(percentage[i]), (x, y), ha=‘center’) plt.ylabel(‘Feature importance (%)’) plt.xlabel(’ ’) plt.legend(loc = 2, bbox_to_anchor = (1,1)) #plt.grid(True) plt.show()

fi=list(features_SO2) fi=list(100*np.array(fi)) #+[‘NO2’.translate(subscript) for i in range(7)]+ #[‘SO2’.translate(subscript) for i in range(7)]+[‘O3’.translate(subscript) for i in range(7)] pollutants=([‘SO2’.translate(subscript) for i in range(7)]) feature=[‘Net signal’, ‘Temperature’, ‘RH’,‘Month’,‘Day’,‘Day of week’,‘Hour’] #+feature+feature+feature features=feature

data=[[fi[i],pollutants[i],features[i]] for i in range(len(fi))]

df=pd.DataFrame(data=data, columns=[‘fi’,‘pollutant’,‘features’]) fig= plt.figure(figsize=(8,5)) ax = fig.add_subplot(111) ax.patch.set_facecolor(‘lightblue’) ax.patch.set_alpha(0) percentage=fi ax=sns.barplot(x=“pollutant”, y=“fi”, hue=“features”, data=df, palette=[‘gold’,‘rebeccapurple’,‘salmon’,‘palevioletred’,‘steelblue’,‘darkkhaki’,‘plum’]) ax.legend_.remove() patches = ax.patches for i in range(len(patches)): x= patches[i].get_x() + patches[i].get_width()/2 y = patches[i].get_height()+.05 ax.annotate(‘{:.1f}%’.format(percentage[i]), (x, y), ha=‘center’) plt.ylabel(‘Feature importance (%)’) plt.xlabel(’ ’) plt.legend(loc = 2, bbox_to_anchor = (1,1)) #plt.grid(True) plt.show()

column=[‘features’,‘CO’,‘NO2’.translate(subscript),‘SO2’.translate(subscript),‘O3’.translate(subscript)] data=[[features[i],100*features_CO[i],100*features_NO2[i],100*features_SO2[i],100*features_O3[i]] for i in range(7)] df=pd.DataFrame(data=data, columns=[‘features’,‘CO’,‘NO2’,‘SO2’,‘O3’]) import seaborn as sns fig= plt.figure(figsize=(15,5)) sns.set() df.set_index(‘features’).T.plot(kind=‘bar’, stacked=True) plt.legend(loc = 2, bbox_to_anchor = (1,1))

import numpy as np import matplotlib import matplotlib.pyplot as plt

vegetables = [“cucumber”, “tomato”, “lettuce”, “asparagus”, “potato”, “wheat”, “barley”] farmers = [“Farmer Joe”, “Upland Bros.”, “Smith Gardening”, “Agrifun”, “Organiculture”, “BioGoods Ltd.”, “Cornylee Corp.”] harvest = np.array([[0.8, 2.4, 2.5, 3.9, 0.0, 4.0, 0.0], [2.4, 0.0, 4.0, 1.0, 2.7, 0.0, 0.0], [1.1, 2.4, 0.8, 4.3, 1.9, 4.4, 0.0], [0.6, 0.0, 0.3, 0.0, 3.1, 0.0, 0.0], [0.7, 1.7, 0.6, 2.6, 2.2, 6.2, 0.0], [1.3, 1.2, 0.0, 0.0, 0.0, 3.2, 5.1], [0.1, 2.0, 0.0, 1.4, 0.0, 1.9, 6.3]])

fig= plt.figure(figsize=(15,5)) fig, ax = plt.subplots() im = ax.imshow(harvest)

21 We want to show all ticks...

ax.set_xticks(np.arange(len(farmers))) ax.set_yticks(np.arange(len(vegetables))) # ... and label them with the respective list entries ax.set_xticklabels(farmers) ax.set_yticklabels(vegetables)

22 Rotate the tick labels and set their alignment.

plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")

23 Loop over data dimensions and create text annotations.

for i in range(len(vegetables)): for j in range(len(farmers)): text = ax.text(j, i, harvest[i, j], ha="center", va="center", color="green")

ax.set_title("Harvest of local farmers (in tons/year)") fig.tight_layout() plt.show()

[246]:
```
features_SO2
```

[246]:
```
array([0.29200574, 0.16464384, 0.21804936, 0.09237384, 0.09530834,
       0.13761888])
```

[247]:
```
A=[0,0]
B=[0]
C=[0,features_O3[6]]
features_CO=np.array(list(features_CO)+A)
features_NO2=np.array(list(features_NO2)+B)
features_SO2=np.array(list(features_SO2)+A)
features_O3=np.array(list(features_O3[:6])+C)
data=[100*features_CO,100*features_NO2,100*features_SO2,100*features_O3]
df=pd.DataFrame(data=data, columns=['Net signal', 'Temperature',␣
↪'RH','Month','Day of week','Hour',
                                    'O3 Concentraion'.translate(subscript),
                                    'NO2 Concentration'.translate(subscript) ] )
pollutants=['CO','NO2'.translate(subscript),'SO2'.translate(subscript),'O3'.
↪translate(subscript)]
df.insert(0, 'pollutants', pollutants)
#df['pollutants']=pollutants
df
```

[247]:

	pollutants	Net signal	Temperature	RH	Month	Day of week
0	CO	71.385754	7.158194	4.860557	5.123193	2.341009
1	NO	32.472666	5.608148	7.049674	7.821667	2.200233
2	SO	29.200574	16.464384	21.804936	9.237384	9.530834
3	O	9.279762	15.354480	19.810008	7.030181	1.125749

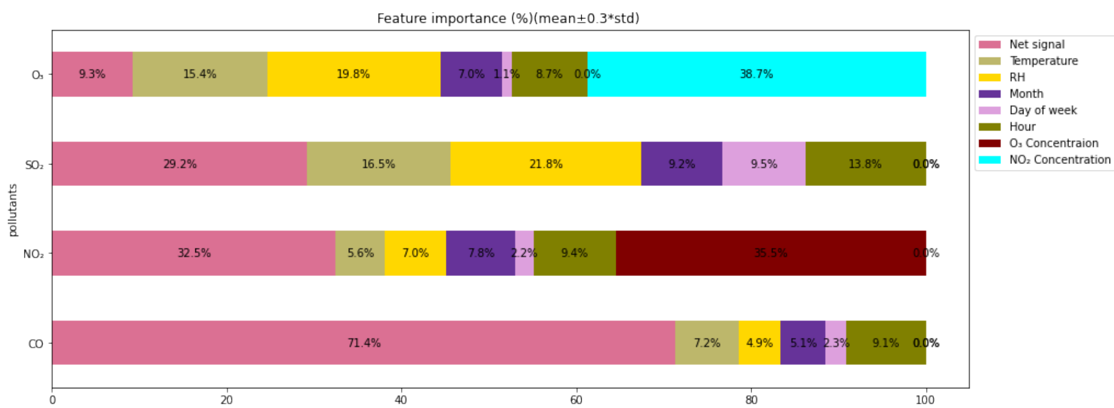
	Hour	O Concentraion	NO Concentration
0	9.131295	0.000000	0.000000
1	9.374190	35.473421	0.000000
2	13.761888	0.000000	0.000000
3	8.653907	0.000000	38.745914

[248]:
```
color=['palevioletred','darkkhaki','gold','rebeccapurple','plum','olive','maroon','cyan']
fig= plt.figure(figsize=(10,10))
```

```
df.plot(
  x = 'pollutants',
  kind = 'barh',
  stacked = True,
  color=color,
  title = 'Feature importance (%)(mean±0.3*std) ',#(mean±0.7*std)
  mark_right = True,
figsize=(15, 6))

df_total = (df["Net signal"] + df["Temperature"] + df["RH"]+df["Month"]+df["Day␣
↪of week"]+df["Hour"]
            +df['O3 Concentraion'.translate(subscript)]+df['NO2 Concentration'.
↪translate(subscript)])
df_rel = df[df.columns[1:]].div(df_total,0)*100

for n in df_rel:
    for i, (cs, ab, pc) in enumerate(zip(df.iloc[:, 1:].cumsum(1)[n],
                                         df[n], df_rel[n])):
        plt.text(cs - ab / 2, i, str(np.round(pc, 1)) + '%',
                 va = 'center', ha = 'center')
plt.legend( loc = 2, bbox_to_anchor = (1,1))
```

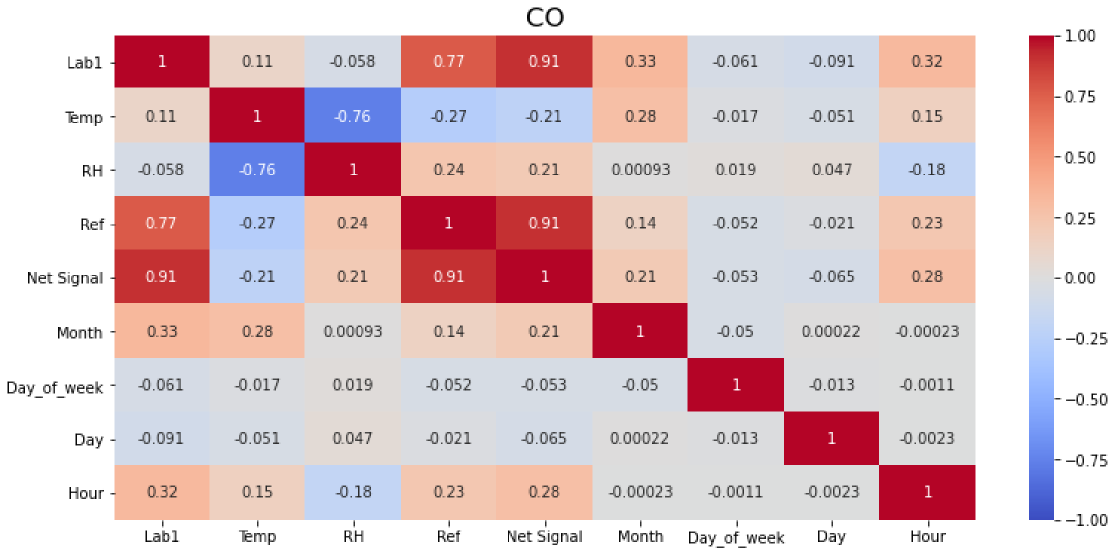
[248]:
```
<matplotlib.legend.Legend at 0x16878ba90>
```

```
<Figure size 720x720 with 0 Axes>
```



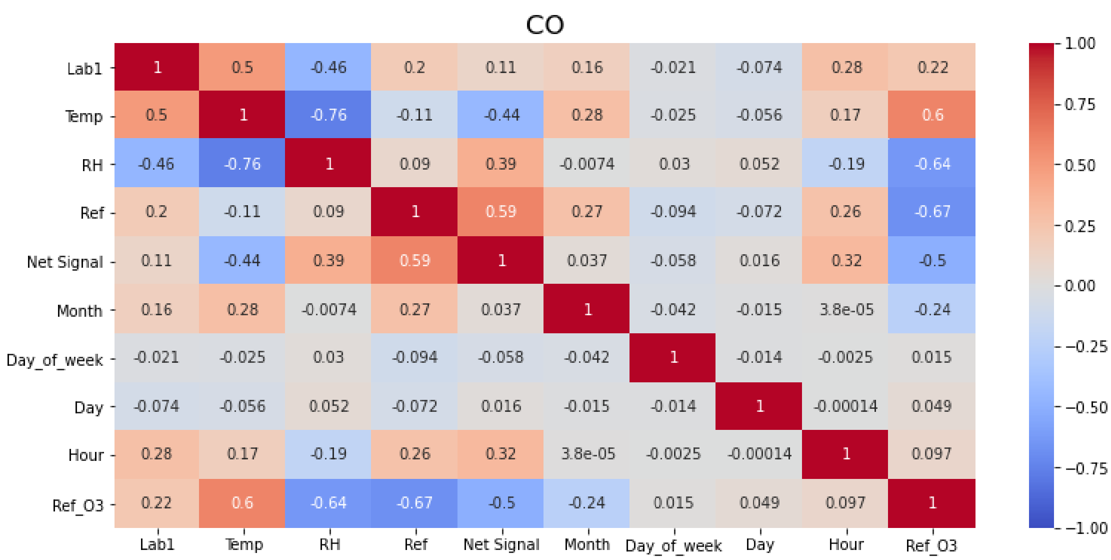
[249]:
```
import matplotlib.pyplot as plt
import seaborn as sns
fig= plt.figure(figsize=(13,6))
sns.heatmap(CO_Data.corr(), annot = True, vmin=-1, vmax=1,
            center= 0, cmap= 'coolwarm')
plt.title('CO',fontsize=18)
```

[249]: Text(0.5, 1.0, 'CO')



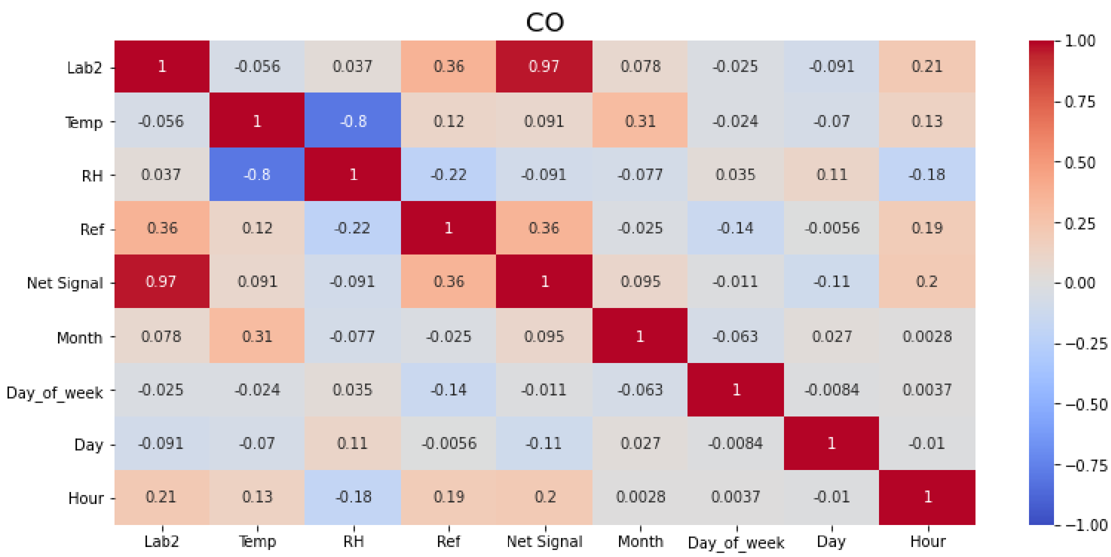
[250]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
fig= plt.figure(figsize=(13,6))
sns.heatmap(NO2_Data.corr(), annot = True, vmin=-1, vmax=1,
            center= 0, cmap= 'coolwarm')
plt.title('CO',fontsize=18)
```

[250]: Text(0.5, 1.0, 'CO')



[251]:
```
import matplotlib.pyplot as plt
import seaborn as sns
fig= plt.figure(figsize=(13,6))
sns.heatmap(SO2_Data.corr(), annot = True, vmin=-1, vmax=1,
            center= 0, cmap= 'coolwarm')
plt.title('CO',fontsize=18)
```

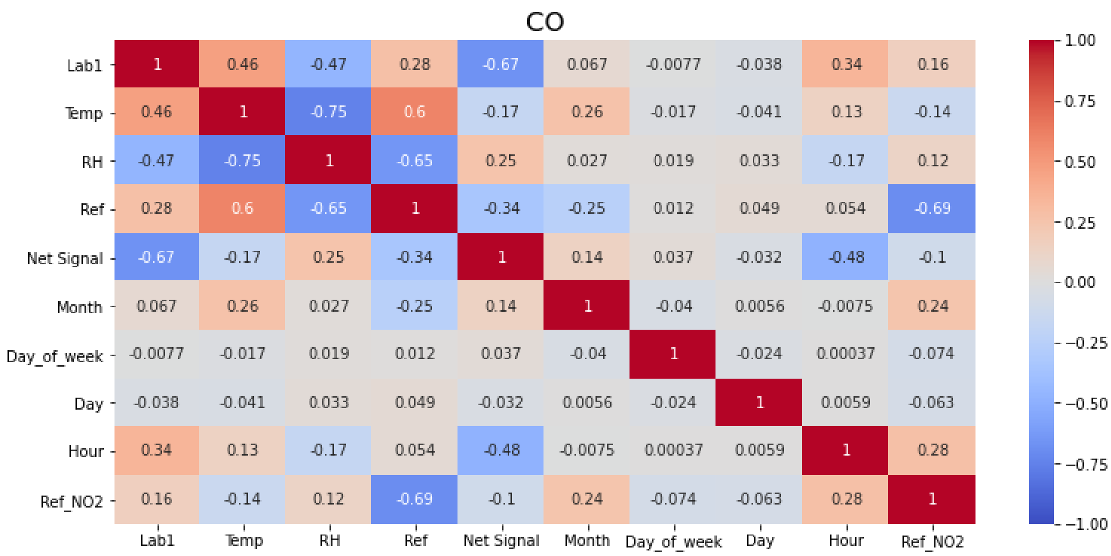
[251]: Text(0.5, 1.0, 'CO')



	Lab2	Temp	RH	Ref	Net Signal	Month	Day_of_week	Day	Hour
Lab2	1	-0.056	0.037	0.36	0.97	0.078	-0.025	-0.091	0.21
Temp	-0.056	1	-0.8	0.12	0.091	0.31	-0.024	-0.07	0.13
RH	0.037	-0.8	1	-0.22	-0.091	-0.077	0.035	0.11	-0.18
Ref	0.36	0.12	-0.22	1	0.36	-0.025	-0.14	-0.0056	0.19
Net Signal	0.97	0.091	-0.091	0.36	1	0.095	-0.011	-0.11	0.2
Month	0.078	0.31	-0.077	-0.025	0.095	1	-0.063	0.027	0.0028
Day_of_week	-0.025	-0.024	0.035	-0.14	-0.011	-0.063	1	-0.0084	0.0037
Day	-0.091	-0.07	0.11	-0.0056	-0.11	0.027	-0.0084	1	-0.01
Hour	0.21	0.13	-0.18	0.19	0.2	0.0028	0.0037	-0.01	1

[252]:
```
import matplotlib.pyplot as plt
import seaborn as sns
fig= plt.figure(figsize=(13,6))
sns.heatmap(O3_Data.corr(), annot = True, vmin=-1, vmax=1,
            center= 0, cmap= 'coolwarm')
plt.title('CO',fontsize=18)
```

[252]: Text(0.5, 1.0, 'CO')



24 Cross Sensitivities

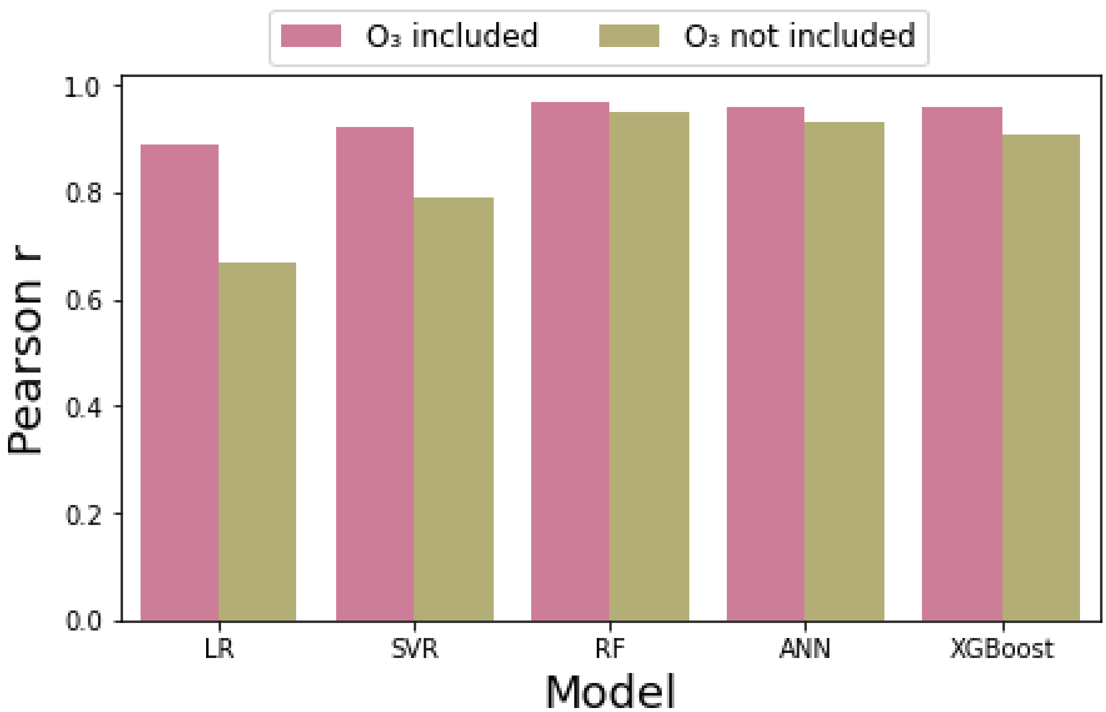
25 NO2

[253]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
Pearson_NO2_inc=[0.89,0.92,0.97,0.96,0.96]
R2_NO2_inc=[0.79,0.84,0.94,0.92,0.92]
RMSE_NO2_inc=[5.7,5,3,3.5,3.6]

Pearson_NO2_Ninc=[0.67,0.79,0.95,0.93,0.91]
R2_NO2_Ninc=[0.45,0.62,0.9,0.86,0.83]
RMSE_NO2_Ninc=[9.4,7.8,4.1,4.7,5.3]
```

[254]:
```python
Pearson=Pearson_NO2_inc+Pearson_NO2_Ninc
Model=['LR','SVR','RF','ANN','XGBoost']
Models=Model+Model
Class1=['NO2 included' for i in range(5)]
Class2=['NO2 not included' for i in range(5)]
Classification=Class1+Class2
data=[[Models[i],Pearson[i],Classification[i] ] for i in range(len(Models))]
df=pd.DataFrame(data=data, columns=['Model', 'Pearson r','Classification'])
#df=df.sample(frac=1)
#df.head()

fig= plt.figure(figsize=(7,4))
```

```
sns.barplot(x="Model", y="Pearson r",␣
 ↪hue="Classification",data=df,palette=['palevioletred','darkkhaki'],
ci=None)
plt.legend(['O3 included'.translate(subscript), 'O3 not included'.
 ↪translate(subscript)],ncol = 4  ,
          bbox_to_anchor = (0.84,1.15),fontsize=12)
#plt.ylabel( 'Average ' + r'$O_{3}$' + ' Concentration ( g/m³)')
#plt.title('NO2 Calibration'.translate(subscript), fontsize=18)
plt.ylabel('Pearson r',fontsize=18)
plt.xlabel('Model',fontsize=18)
plt.legend
```

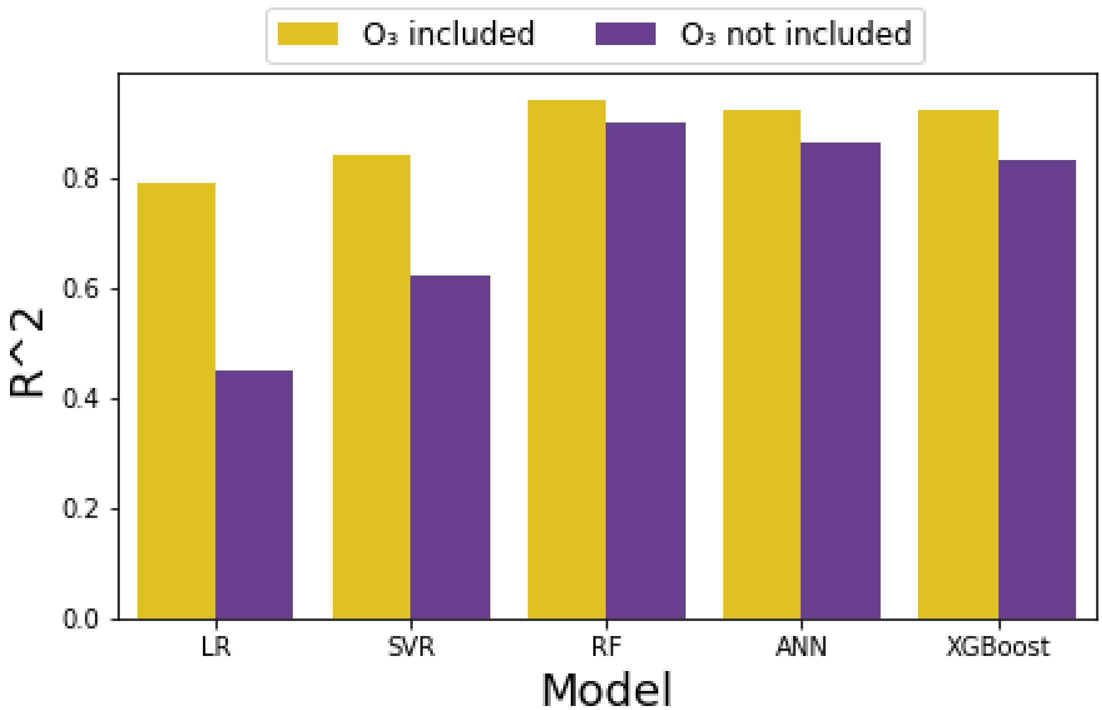
[254]: <function matplotlib.pyplot.legend(*args, **kwargs)>



[255]:
```
R2=R2_NO2_inc+R2_NO2_Ninc
Model=['LR','SVR','RF','ANN','XGBoost']
Models=Model+Model
Class1=['NO2 included' for i in range(5)]
Class2=['NO2 not included' for i in range(5)]
Classification=Class1+Class2
data=[[Models[i],R2[i],Classification[i] ] for i in range(len(Models))]
df=pd.DataFrame(data=data, columns=['Model', 'R2','Classification'])
#df=df.sample(frac=1)
```

```
#df.head()

fig= plt.figure(figsize=(7,4))
sns.barplot(x="Model", y="R2",␣
 ↪hue="Classification",data=df,palette=['gold','rebeccapurple'],
ci=None)
plt.legend(['O3 included'.translate(subscript), 'O3 not included'.
 ↪translate(subscript)],ncol = 4  ,
           bbox_to_anchor = (0.84,1.15),fontsize=12)
#plt.ylabel( 'Average ' + r'$O_{3}$' + ' Concentration ( g/m³)')
#plt.title('NO2 Calibration'.translate(subscript), fontsize=18)
plt.ylabel('R^2',fontsize=18)
plt.xlabel('Model',fontsize=18)
plt.legend
```

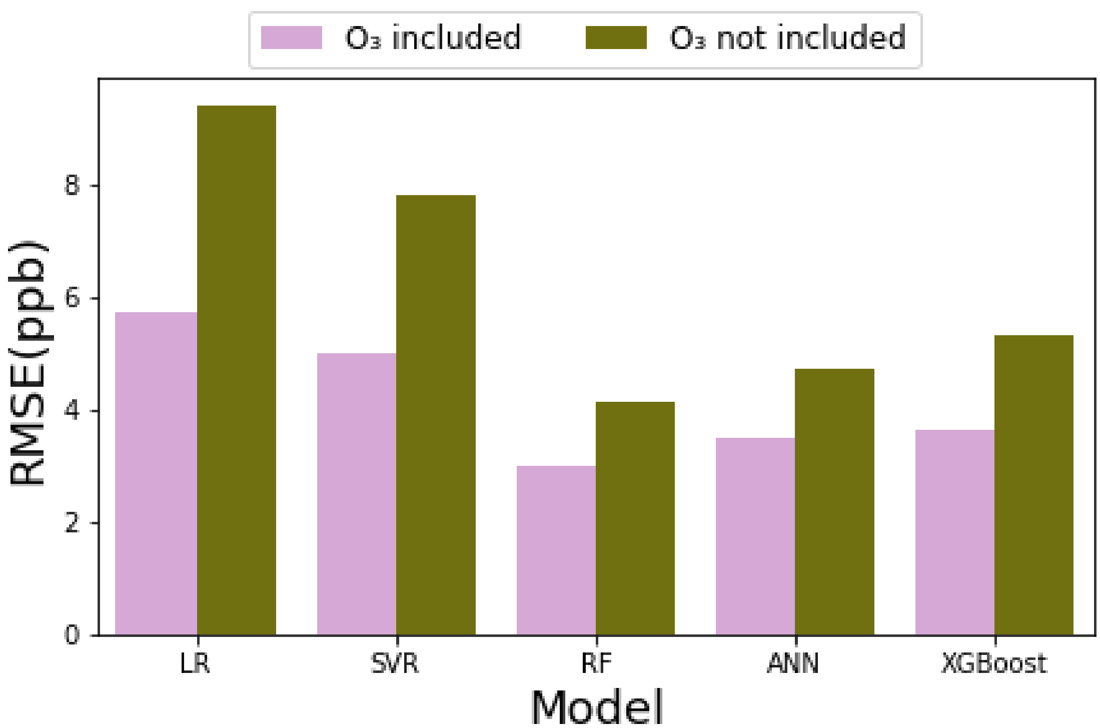
[255]: <function matplotlib.pyplot.legend(*args, **kwargs)>



[256]:
```
RMSE=RMSE_NO2_inc+RMSE_NO2_Ninc
Model=['LR','SVR','RF','ANN','XGBoost']
Models=Model+Model
Class1=['NO2 included' for i in range(5)]
Class2=['NO2 not included' for i in range(5)]
Classification=Class1+Class2
```

```
data=[[Models[i],RMSE[i],Classification[i] ] for i in range(len(Models))]
df=pd.DataFrame(data=data, columns=['Model', 'RMSE','Classification'])
#df=df.sample(frac=1)
#df.head()

fig= plt.figure(figsize=(7,4))
sns.barplot(x="Model", y="RMSE",␣
 ↪hue="Classification",data=df,palette=['plum','olive'],
ci=None)
plt.legend(['O3 included'.translate(subscript), 'O3 not included'.
 ↪translate(subscript)],ncol = 4  ,
            bbox_to_anchor = (0.84,1.15),fontsize=12)
#plt.ylabel( 'Average ' + r'$O_{3}$' + ' Concentration ( g/m³)')
#plt.title('NO2 Calibration'.translate(subscript), fontsize=18)
plt.ylabel('RMSE(ppb)',fontsize=18)
plt.xlabel('Model',fontsize=18)
plt.legend
```

[256]: <function matplotlib.pyplot.legend(*args, **kwargs)>

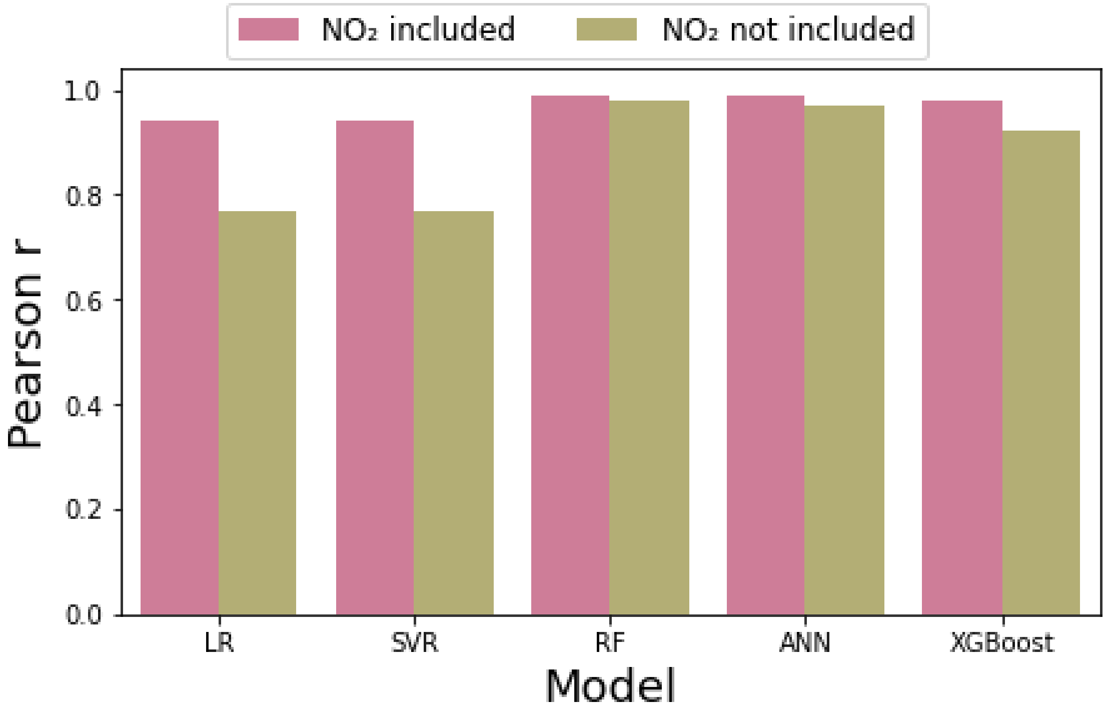


26 O3

[257]:
```
Pearson_O3_inc=[0.94,0.94,0.99,0.99,0.98]
R2_O3_inc=[0.89,0.89,0.98,0.97,0.97]
RMSE_O3_inc=[5,5,2.2,2.6,2.8]

Pearson_O3_Ninc=[0.77,0.77,0.98,0.97,0.92]
R2_O3_Ninc=[0.59,0.59,0.95,0.93,0.85]
RMSE_O3_Ninc=[9.6,9.7,3.3,3.9,5.7]
```

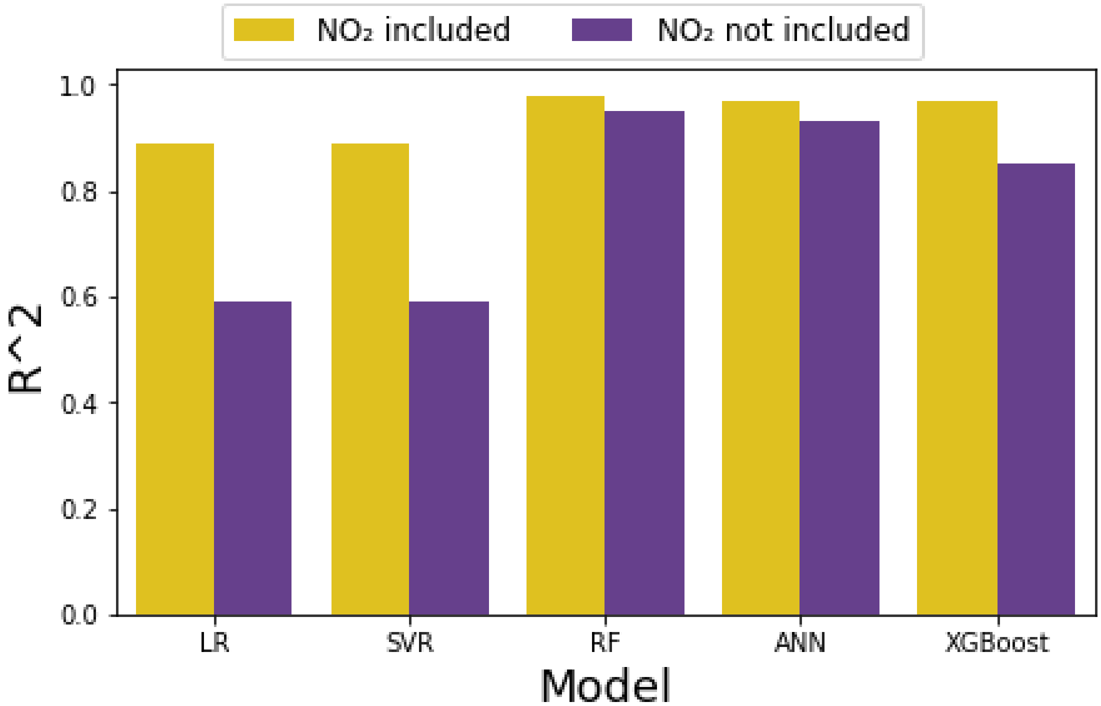
[258]:
```
Pearson=Pearson_O3_inc+Pearson_O3_Ninc
Model=['LR','SVR','RF','ANN','XGBoost']
Models=Model+Model
Class1=['NO2 included' for i in range(5)]
Class2=['NO2 not included' for i in range(5)]
Classification=Class1+Class2
data=[[Models[i],Pearson[i],Classification[i] ] for i in range(len(Models))]
df=pd.DataFrame(data=data, columns=['Model', 'Pearson r','Classification'])
#df=df.sample(frac=1)
#df.head()

fig= plt.figure(figsize=(7,4))
sns.barplot(x="Model", y="Pearson r",␣
↪hue="Classification",data=df,palette=['palevioletred','darkkhaki'],
ci=None)
plt.legend(['NO2 included'.translate(subscript), 'NO2 not included'.
↪translate(subscript)],ncol = 4  ,
           bbox_to_anchor = (0.84,1.15),fontsize=12)
#plt.ylabel( 'Average ' + r'$O_{3}$' + ' Concentration ( g/m³)')
#plt.title('NO2 Calibration'.translate(subscript), fontsize=18)
plt.ylabel('Pearson r',fontsize=18)
plt.xlabel('Model',fontsize=18)
plt.legend
```

[258]:
```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```



[259]:
```
R2=R2_O3_inc+R2_O3_Ninc
Model=['LR','SVR','RF','ANN','XGBoost']
Models=Model+Model
Class1=['NO2 included' for i in range(5)]
Class2=['NO2 not included' for i in range(5)]
Classification=Class1+Class2
data=[[Models[i],R2[i],Classification[i] ] for i in range(len(Models))]
df=pd.DataFrame(data=data, columns=['Model', 'R2','Classification'])
#df=df.sample(frac=1)
#df.head()

fig= plt.figure(figsize=(7,4))
sns.barplot(x="Model", y="R2",␣
 ↪hue="Classification",data=df,palette=['gold','rebeccapurple'],
ci=None)
plt.legend(['NO2 included'.translate(subscript), 'NO2 not included'.
 ↪translate(subscript)],ncol = 4  ,
           bbox_to_anchor = (0.84,1.15),fontsize=12)
#plt.ylabel( 'Average ' + r'$O_{3}$' + ' Concentration ( g/m³)')
#plt.title('NO2 Calibration'.translate(subscript), fontsize=18)
plt.ylabel('R^2',fontsize=18)
plt.xlabel('Model',fontsize=18)
plt.legend
```

[259]:
```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```



[260]:
```python
RMSE=RMSE_03_inc+RMSE_03_Ninc
Model=['LR','SVR','RF','ANN','XGBoost']
Models=Model+Model
Class1=['NO2 included' for i in range(5)]
Class2=['NO2 not included' for i in range(5)]
Classification=Class1+Class2
data=[[Models[i],RMSE[i],Classification[i] ] for i in range(len(Models))]
df=pd.DataFrame(data=data, columns=['Model', 'RMSE','Classification'])
#df=df.sample(frac=1)
#df.head()

fig= plt.figure(figsize=(7,4))
sns.barplot(x="Model", y="RMSE", hue="Classification",data=df,palette=['plum','olive'],
ci=None)
plt.legend(['NO2 included'.translate(subscript), 'NO2 not included'.translate(subscript)],ncol = 4  ,
           bbox_to_anchor = (0.84,1.15),fontsize=12)
#plt.ylabel( 'Average ' + r'$O_{3}$' + ' Concentration ( g/m³)')
#plt.title('NO2 Calibration'.translate(subscript), fontsize=18)
plt.ylabel('RMSE(ppb)',fontsize=18)
```

```
plt.xlabel('Model',fontsize=18)
plt.legend
```

[260]: <function matplotlib.pyplot.legend(*args, **kwargs)>