Sensor Schemes CO

November 20, 2021

1 DATA

[66]:
```
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()
```

[67]:
```
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.resample('5min').mean()
CO_Data=CO_Data[(CO_Data[CO_Data.columns] >= 0).all(axis=1)]
CO_Data=CO_Data.dropna()
CO_Data.shape
```

[67]:
```
(45117, 9)
```

[68]:
```
CO_Data=CO_Data.resample('h').mean()
CO_Data=CO_Data.dropna()
```

[69]:
```
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
```

[70]:
```
df1=[x for _, x in CO_Data.groupby('Month')]
data_oct=df1[4]
#data_oct=data_oct.sample(frac=1)
data_nov=df1[5]
#data_nov=data_nov.sample(frac=1)
data_dec=df1[6]
#data_dec=data_dec.sample(frac=1)
data_jan=df1[0]
#data_jan=data_jan.sample(frac=1)
data_feb=df1[1]
#data_feb=data_feb.sample(frac=1)
data_mar=df1[2]
#data_mar=data_mar.sample(frac=1)
data_apr=df1[3]
#data_apr=data_apr.sample(frac=1)
data=[data_oct,data_nov,data_dec,data_jan,data_feb,data_mar]
data_apr.head()
```

[70]:

Date	Lab1	Temp	RH	Ref	Net Signal	Month	Day_of_week	Day	Hour
2020-04-01 00:00:00	229.616988	13.851747	82.302940	197.721703	95.726012	4.0	2.0	1.0	0.0
2020-04-01 01:00:00	199.698257	13.454867	83.531290	263.747572	82.533367	4.0	2.0	1.0	1.0
2020-04-01 02:00:00	171.861486	13.304556	82.558732	253.751264	69.676736	4.0	2.0	1.0	2.0
2020-04-01 03:00:00	136.027044	13.187792	82.926804	204.773281	52.308946	4.0	2.0	1.0	3.0
2020-04-01 04:00:00	150.059790	13.625268	84.595291	217.311244	58.030088	4.0	2.0	1.0	4.0

2 Mothly schemes

3 Oct 2019

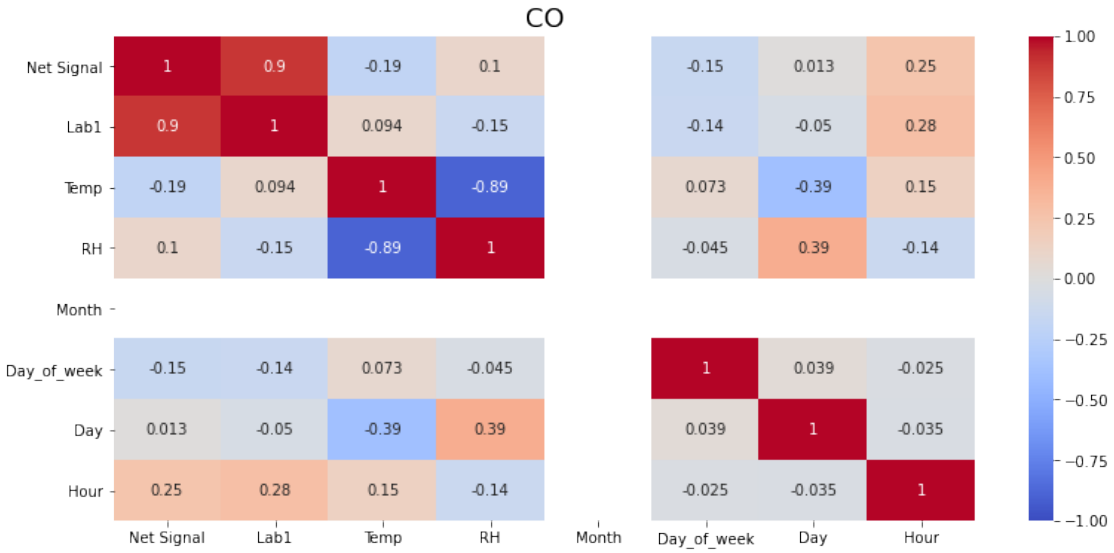
[71]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import sklearn.metrics as sm
```

```
import matplotlib.pyplot as plt
```

[72]:
```
#,'Month','Day_of_week','Day','Hour'
X=data_oct[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=data_oct['Ref']
A=np.array(y)/np.mean(y)
A=sorted(A, reverse=True)
sum1=sum(A[:100])
mean1=np.std(y)/np.mean(y)
```

[73]:
```
import matplotlib.pyplot as plt
import seaborn as sns
fig= plt.figure(figsize=(13,6))
sns.heatmap(X.corr(), annot = True, vmin=-1, vmax=1,
            center= 0, cmap= 'coolwarm')
plt.title('CO',fontsize=18)
```

[73]:
```
Text(0.5, 1.0, 'CO')
```



4 RF

[74]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
#from sklearn.metrics import mean_absolute_error as mae
#from sklearn.metrics import mean_absolute_percentage_error
import sklearn.metrics as sm
import matplotlib.pyplot as plt
import numpy as np
```

```
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)
```

[75]:
```
Day=[2*i for i in range(1,8) ]
Rmse1_rf=[]
RMSE1_rf=[]
for i in range(1,8):
    regressor.fit(X[:48*i].drop(['Lab1'], axis=1), y[:48*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(sm.r2_score(y_test, pred), 2)
    #mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse1_rf.append(mse)
    RMSE1_rf.append(rmse)
```

[76]:
```
A=y.to_list()
Ext_oct=[]
for i in range(len(A)):
    if A[i]>3*np.mean(A):
        Ext_oct.append(i)
N_Ext_oct=len(Ext_oct)
N_Ext_oct
```

[76]: 13

[77]:
```
mean_oct=np.mean(y)
N_oct=y.shape[0]
Mean_Rmse_oct=np.mean(Rmse1_rf)
```

5 XGBoost

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)

Rmse1_ann=[] RMSE1_ann=[] for i in range(1,8): model.fit(X[:48*i].drop(['Lab1'], axis=1), y[:48*i]) pred=model.predict(X.drop(['Lab1'], axis=1)) y_test=y mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2) #rmse=

mape=round(mean_absolute_percentage_error(y_test,pred),2) rmse=round(np.corrcoef(y_test, pred)[0, 1],2) Rmse1_ann.append(mse) RMSE1_ann.append(rmse)

6 Nov 2019

RF

[78]:
```
X=data_nov[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=data_nov['Ref']
A=np.array(y)/np.mean(y)
A=sorted(A, reverse=True)
sum2=sum(A[:100])
mean2=np.std(y)/np.mean(y)
Rmse2_rf=[]
RMSE2_rf=[]
for i in range(1,8):
    regressor.fit(X[:48*i].drop(['Lab1'], axis=1), y[:48*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(sm.r2_score(y_test, pred), 2)
    #mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse2_rf.append(mse)
    RMSE2_rf.append(rmse)
```

[79]:
```
A=y.to_list()
Ext_nov=[]
for i in range(len(A)):
    if A[i]>3*np.mean(A):
        Ext_nov.append(i)
N_Ext_nov=len(Ext_nov)
N_Ext_nov
```

[79]: 26

[80]:
```
mean_nov=np.mean(y)
N_nov=y.shape[0]
Mean_Rmse_nov=np.mean(Rmse2_rf)
```

7 XGBoost

Rmse2_ann=[] RMSE2_ann=[] for i in range(1,8): model.fit(X[:48*i].drop([‘Lab1’], axis=1), y[:48*i]) pred=model.predict(X.drop([‘Lab1’], axis=1)) y_test=y

```
mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
#rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
```

```
Rmse2_ann.append(mse)
RMSE2_ann.append(rmse)
```

8 Dec 2019

9 RF

[81]:
```
X=data_dec[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=data_dec['Ref']
A=np.array(y)/np.mean(y)
A=sorted(A, reverse=True)
sum3=sum(A[:100])
mean3=np.std(y)/np.mean(y)
Rmse3_rf=[]
RMSE3_rf=[]
for i in range(1,8):
    regressor.fit(X[:48*i].drop(['Lab1'], axis=1), y[:48*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(sm.r2_score(y_test, pred), 2)
    #mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse3_rf.append(mse)
    RMSE3_rf.append(rmse)
```

[82]:
```
A=y.to_list()
Ext_dec=[]
for i in range(len(A)):
    if A[i]>3*np.mean(A):
        Ext_dec.append(i)
N_Ext_dec=len(Ext_dec)
N_Ext_dec
```

[82]: 20

[83]:
```
mean_dec=np.mean(y)
N_dec=y.shape[0]
Mean_Rmse_dec=np.mean(Rmse3_rf)
```

10 XGBoost

Rmse3_ann=[] RMSE3_ann=[] for i in range(1,8): model.fit(X[:48*i].drop([‘Lab1’], axis=1), y[:48*i]) pred=model.predict(X.drop([‘Lab1’], axis=1)) y_test=y mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2) #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2) rmse=round(np.corrcoef(y_test, pred)[0, 1],2) Rmse3_ann.append(mse) RMSE3_ann.append(rmse)

11 Jan 2020

12 RF

[84]:
```
X=data_jan[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=data_jan['Ref']
A=np.array(y)/np.mean(y)
A=sorted(A, reverse=True)
sum4=sum(A[:100])
mean4=np.std(y)/np.mean(y)
Rmse4_rf=[]
RMSE4_rf=[]
for i in range(1,8):
    regressor.fit(X[:48*i].drop(['Lab1'], axis=1), y[:48*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(sm.r2_score(y_test, pred), 2)
    #mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse4_rf.append(mse)
    RMSE4_rf.append(rmse)
```

[85]:
```
A=y.to_list()
Ext_jan=[]
for i in range(len(A)):
    if A[i]>3*np.mean(A):
        Ext_jan.append(i)
N_Ext_jan=len(Ext_jan)
N_Ext_jan
```

[85]: 23

[86]:
```
mean_jan=np.mean(y)
N_jan=y.shape[0]
Mean_Rmse_jan=np.mean(Rmse4_rf)
```

13 XGBoost

[87]:
```
Rmse4_ann=[]
RMSE4_ann=[]
for i in range(1,8):
    model.fit(X[:48*i].drop(['Lab1'], axis=1), y[:48*i])
    pred=model.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(sm.r2_score(y_test, pred), 2)
    #mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
```

```
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse4_ann.append(mse)
    RMSE4_ann.append(rmse)
```

14 Feb 2020

15 RF

X=data_feb[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']] y=data_feb['Ref'] A=np.array(y)/np.mean(y) A=sorted(A, reverse=True) sum5=sum(A[:100]) mean5=np.std(y)/np.mean(y) Rmse5_rf=[] RMSE5_rf=[] for i in range(1,11): regressor.fit(X[:576*i].drop(['Lab1'], axis=1), y[:576*i]) pred=regressor.predict(X[576*i:].drop(['Lab1'], axis=1)) y_test=y[576*i:] mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2) #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2) rmse=round(np.corrcoef(y_test, pred)[0, 1],2) Rmse5_rf.append(mse) RMSE5_rf.append(rmse)

A=y.to_list() Ext_feb=[] for i in range(len(A)): if A[i]>3*np.mean(A): Ext_feb.append(i) N_Ext_feb=len(Ext_feb) N_Ext_feb

mean_feb=np.mean(y) N_feb=y.shape[0] Mean_Rmse_feb=np.mean(Rmse5_rf)

16 XGBoost

Rmse5_ann=[] RMSE5_ann=[] for i in range(1,11): model.fit(X[:576*i].drop(['Lab1'], axis=1), y[:576*i]) pred=model.predict(X[576*i:].drop(['Lab1'], axis=1)) y_test=y[576*i:]

```
mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
#rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
Rmse5_ann.append(mse)
RMSE5_ann.append(rmse)
```

17 March 2020

18 RF

[88]:
```
X=data_mar[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=data_mar['Ref']
A=np.array(y)/np.mean(y)
A=sorted(A, reverse=True)
sum6=sum(A[:100])
mean6=np.std(y)/np.mean(y)
Rmse6_rf=[]
RMSE6_rf=[]
for i in range(1,8):
```

```
    regressor.fit(X[:48*i].drop(['Lab1'], axis=1), y[:48*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(sm.r2_score(y_test, pred), 2)
    #mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse6_rf.append(mse)
    RMSE6_rf.append(rmse)
```

[89]:
```
A=y.to_list()
Ext_mar=[]
for i in range(len(A)):
    if A[i]>3*np.mean(A):
        Ext_mar.append(i)
N_Ext_mar=len(Ext_mar)
N_Ext_mar
```

[89]: 11

[90]:
```
mean_mar=np.mean(y)
N_mar=y.shape[0]
Mean_Rmse_mar=np.mean(Rmse6_rf)
```

19 XGBoost

Rmse6_ann=[] RMSE6_ann=[] for i in range(1,8): model.fit(X[:48*i].drop([‘Lab1’], axis=1), y[:48*i]) pred=model.predict(X.drop([‘Lab1’], axis=1)) y_test=y

```
mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
#rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
Rmse6_ann.append(mse)
RMSE6_ann.append(rmse)
```

20 April 2020

21 RF

X=data_apr[[‘Net Signal’,‘Lab1’,‘Temp’,‘RH’,‘Month’,‘Day_of_week’,‘Day’,‘Hour’]] y=data_apr[‘Ref’] A=np.array(y)/np.mean(y) A=sorted(A, reverse=True) sum7=sum(A[:100]) mean7=np.std(y)/np.mean(y) Rmse10_rf=[] RMSE10_rf=[] for i in range(1,11): regressor.fit(X[:576*i].drop([‘Lab1’], axis=1), y[:576*i]) pred=regressor.predict(X[576*i:].drop([‘Lab1’], axis=1)) y_test=y[576*i:] mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2) #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2) rmse=round(np.corrcoef(y_test, pred)[0, 1],2) Rmse10_rf.append(mse) RMSE10_rf.append(rmse)

A=y.to_list() Ext_apr=[] for i in range(len(A)): if A[i]>3*np.mean(A): Ext_apr.append(i) N_Ext_apr=len(Ext_apr) N_Ext_apr

22 XGBoost

Rmse10_ann=[] RMSE10_ann=[] for i in range(1,11): model.fit(X[:576*i].drop(['Lab1'], axis=1), y[:576*i]) pred=model.predict(X[576*i:].drop(['Lab1'], axis=1)) y_test=y[576*i:] mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2) #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2) rmse=round(np.corrcoef(y_test, pred)[0, 1],2) Rmse10_ann.append(mse) RMSE10_ann.append(rmse)

mean_apr=np.mean(y) N_apr=y.shape[0] Mean_Rmse_apr=np.mean(Rmse10_rf)

Mean_Rmse=[Mean_Rmse_oct,Mean_Rmse_nov,Mean_Rmse_dec,Mean_Rmse_jan,Mean_Rmse_feb,Mean_
Mean_conc=[mean_oct,mean_nov,mean_dec,mean_jan,mean_feb,mean_mar,mean_apr]
N=[N_oct,N_nov,N_dec,N_jan,N_feb,N_mar,N_apr] N_Ext=[N_Ext_oct,N_Ext_nov,N_Ext_dec,N_Ext_

Rmse=[] for i in range(10): A=[Rmse1_rf[i],Rmse2_rf[i],Rmse3_rf[i],Rmse4_rf[i],Rmse5_rf[i],Rmse6_rf[i],Rmse1
Rmse.append(A) RMSE=Rmse[0]+Rmse[1]+Rmse[2]+Rmse[3]+Rmse[4]+Rmse[5]+Rmse[6]+Rmse[7]
Conc=Mean_conc+Mean_conc+Mean_conc+Mean_conc+Mean_conc+Mean_conc+Mean_conc+Mean_conc
N_dp=N+N+N+N+N+N+N+N N_ext=N_Ext+N_Ext+N_Ext+N_Ext+N_Ext+N_Ext+N_Ext+N_Ext

[91]:
```
import plotly.express as px
from IPython.display import Image
import plotly.graph_objects as go
import numpy as np
Day_1_rf=[RMSE1_rf[0],RMSE2_rf[0],RMSE3_rf[0],RMSE4_rf[0],RMSE6_rf[0]]
Day_2_rf=[RMSE1_rf[1],RMSE2_rf[1],RMSE3_rf[1],RMSE4_rf[1],RMSE6_rf[1]]
Day_3_rf=[RMSE1_rf[2],RMSE2_rf[2],RMSE3_rf[2],RMSE4_rf[2],RMSE6_rf[2]]
Day_4_rf=[RMSE1_rf[3],RMSE2_rf[3],RMSE3_rf[3],RMSE4_rf[3],RMSE6_rf[3]]
Day_5_rf=[RMSE1_rf[4],RMSE2_rf[4],RMSE3_rf[4],RMSE4_rf[4],RMSE6_rf[4]]
Day_6_rf=[RMSE1_rf[5],RMSE2_rf[5],RMSE3_rf[5],RMSE4_rf[5],RMSE6_rf[5]]
Day_7_rf=[RMSE1_rf[6],RMSE2_rf[6],RMSE3_rf[6],RMSE4_rf[6],RMSE6_rf[6]]
#Day_8_rf=[RMSE1_rf[7],RMSE2_rf[7],RMSE3_rf[7],RMSE4_rf[7],RMSE6_rf[7]]
#Day_9_rf=[RMSE1_rf[8],RMSE2_rf[8],RMSE3_rf[8],RMSE4_rf[8],RMSE6_rf[8]]
#Day_10_rf=[RMSE1_rf[9],RMSE2_rf[9],RMSE3_rf[9],RMSE4_rf[9],RMSE6_rf[9]]


Day_1_ann=[RMSE1_ann[0],RMSE2_ann[0],RMSE3_ann[0],RMSE4_ann[0],RMSE6_ann[0]]
Day_2_ann=[RMSE1_ann[1],RMSE2_ann[1],RMSE3_ann[1],RMSE4_ann[1],RMSE6_ann[1]]
Day_3_ann=[RMSE1_ann[2],RMSE2_ann[2],RMSE3_ann[2],RMSE4_ann[2],RMSE6_ann[2]]
Day_4_ann=[RMSE1_ann[3],RMSE2_ann[3],RMSE3_ann[3],RMSE4_ann[3],RMSE6_ann[3]]
Day_5_ann=[RMSE1_ann[4],RMSE2_ann[4],RMSE3_ann[4],RMSE4_ann[4],RMSE6_ann[4]]
Day_6_ann=[RMSE1_ann[5],RMSE2_ann[5],RMSE3_ann[5],RMSE4_ann[5],RMSE6_ann[5]]
Day_7_ann=[RMSE1_ann[6],RMSE2_ann[6],RMSE3_ann[6],RMSE4_ann[6],RMSE6_ann[6]]
#Day_8_ann=[RMSE1_ann[7],RMSE2_ann[7],RMSE3_ann[7],RMSE4_ann[7],RMSE6_ann[7]]
#Day_9_ann=[RMSE1_ann[8],RMSE2_ann[8],RMSE3_ann[8],RMSE4_ann[8],RMSE6_ann[8]]
#Day_10_ann=[RMSE1_ann[9],RMSE2_ann[9],RMSE3_ann[9],RMSE4_ann[9],RMSE6_ann[9]]
```

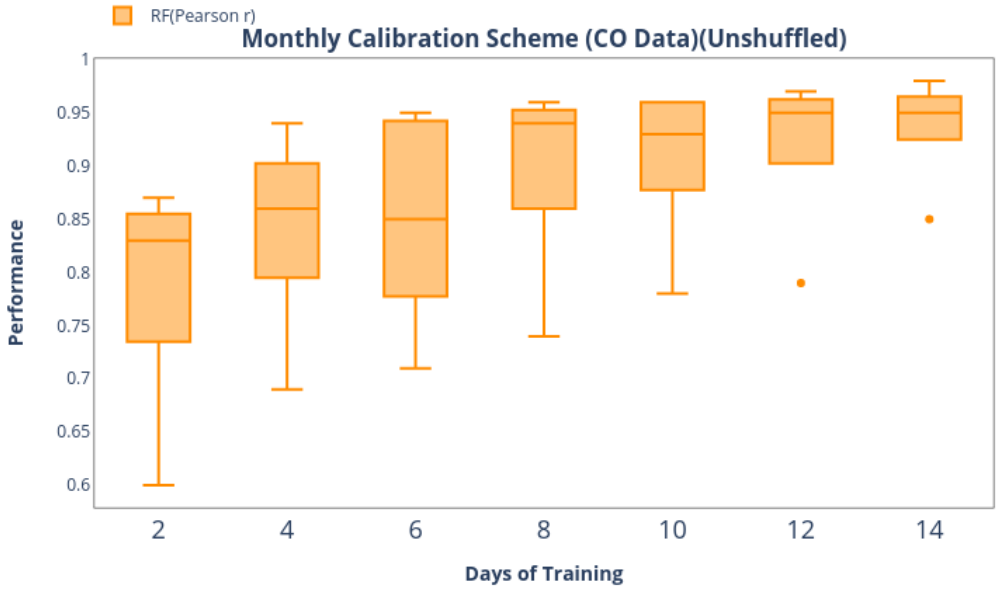
[92]:
```
Day_1_RF=[Rmse1_rf[0],Rmse2_rf[0],Rmse3_rf[0],Rmse4_rf[0],Rmse6_rf[0]]
Day_2_RF=[Rmse1_rf[1],Rmse2_rf[1],Rmse3_rf[1],Rmse4_rf[1],Rmse6_rf[1]]
Day_3_RF=[Rmse1_rf[2],Rmse2_rf[2],Rmse3_rf[2],Rmse4_rf[2],Rmse6_rf[2]]
Day_4_RF=[Rmse1_rf[3],Rmse2_rf[3],Rmse3_rf[3],Rmse4_rf[3],Rmse6_rf[3]]
Day_5_RF=[Rmse1_rf[4],Rmse2_rf[4],Rmse3_rf[4],Rmse4_rf[4],Rmse6_rf[4]]
Day_6_RF=[Rmse1_rf[5],Rmse2_rf[5],Rmse3_rf[5],Rmse4_rf[5],Rmse6_rf[5]]
Day_7_RF=[Rmse1_rf[6],Rmse2_rf[6],Rmse3_rf[6],Rmse4_rf[6],Rmse6_rf[6]]
#Day_8_RF=[Rmse1_rf[7],Rmse2_rf[7],Rmse3_rf[7],Rmse4_rf[7],Rmse6_rf[7]]
#Day_9_RF=[Rmse1_rf[8],Rmse2_rf[8],Rmse3_rf[8],Rmse4_rf[8],Rmse6_rf[8]]
#Day_10_RF=[Rmse1_rf[9],Rmse2_rf[9],Rmse3_rf[9],Rmse4_rf[9],Rmse6_rf[9]]



Day_1_ANN=[Rmse1_ann[0],Rmse2_ann[0],Rmse3_ann[0],Rmse4_ann[0],Rmse6_ann[0]]
Day_2_ANN=[Rmse1_ann[1],Rmse2_ann[1],Rmse3_ann[1],Rmse4_ann[1],Rmse6_ann[1]]
Day_3_ANN=[Rmse1_ann[2],Rmse2_ann[2],Rmse3_ann[2],Rmse4_ann[2],Rmse6_ann[2]]
Day_4_ANN=[Rmse1_ann[3],Rmse2_ann[3],Rmse3_ann[3],Rmse4_ann[3],Rmse6_ann[3]]
Day_5_ANN=[Rmse1_ann[4],Rmse2_ann[4],Rmse3_ann[4],Rmse4_ann[4],Rmse6_ann[4]]
Day_6_ANN=[Rmse1_ann[5],Rmse2_ann[5],Rmse3_ann[5],Rmse4_ann[5],Rmse6_ann[5]]
Day_7_ANN=[Rmse1_ann[6],Rmse2_ann[6],Rmse3_ann[6],Rmse4_ann[6],Rmse6_ann[6]]
#Day_8_ANN=[Rmse1_ann[7],Rmse2_ann[7],Rmse3_ann[7],Rmse4_ann[7],Rmse6_ann[7]]
#Day_9_ANN=[Rmse1_ann[8],Rmse2_ann[8],Rmse3_ann[8],Rmse4_ann[8],Rmse6_ann[8]]
#Day_10_ANN=[Rmse1_ann[9],Rmse2_ann[9],Rmse3_ann[9],Rmse4_ann[9],Rmse6_ann[9]]
```

[93]:
```
RF_P=Day_1_rf+Day_2_rf+Day_3_rf+Day_4_rf+Day_5_rf+Day_6_rf+Day_7_rf#+Day_8_rf+Day_9_rf+Day_10_
#ANN_P=Day_1_ann+Day_2_ann+Day_3_ann+Day_4_ann+Day_5_ann+Day_6_ann+Day_7_ann#+Day_8_ann+Day_9_
RF_R=Day_1_RF+Day_2_RF+Day_3_RF+Day_4_RF+Day_5_RF+Day_6_RF+Day_7_RF#+Day_8_RF+Day_9_RF+Day_10_
#ANN_R=Day_1_ANN+Day_2_ANN+Day_3_ANN+Day_4_ANN+Day_5_ANN+Day_6_ANN+Day_7_ANN#+Day_8_ANN+Day_9_

x1=['2' for i in range(5)]
x2=['4' for i in range(5)]
x3=['6' for i in range(5)]
x4=['8' for i in range(5)]
x5=['10' for i in range(5)]
x6=['12' for i in range(5)]
x7=['14' for i in range(5)]
#x8=['16' for i in range(5)]
#x9=['18' for i in range(5)]
#x10=['20' for i in range(5)]



x=x1+x2+x3+x4+x5+x6+x7#+x8+x9+x10
len(x)
```

[93]: 35

[96]:
```
import chart_studio
fig = go.Figure()
a = np.array([1, 2, 3, 4, 5])
b = np.array([1, 3, 2, 3, 1])
# Defining x axis
x = x
fig.add_trace(go.Box(

    # defining y axis in corresponding
    # to x-axis
    y=RF_P,
    x=x,
    name='RF(Pearson r)',
    marker_color='darkorange',
    showlegend=True

))
#fig.add_trace(go.Box(
    #y=ANN_P,
   # x=x,
    #name='XGBoost(Pearson r)',
    #marker_color='darkcyan',
    #showlegend=True

#))

#fig.add_trace(go.Box(

    # defining y axis in corresponding
    # to x-axis
   # y=RF_R,
    #x=x,
    #name='RF(R^2)',
    #marker_color='teal',
    #showlegend=True

#))
#fig.add_trace(go.Box(
   # y=ANN_R,
    #x=x,
    #name='XGBoost(NSRMSE)',
    #marker_color='deeppink',
    #showlegend=True

#))
```

[96]:
```
fig.update_layout(autosize=False,
                  title={'text': "<b>Monthly Calibration Scheme (CO␣
 ↪Data)(Unshuffled)</b>",
        'y':0.84,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
    width=800,
    height=500,

  legend=dict( yanchor="bottom",
    y=1.05,
    orientation="h"
),
    # 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>Days of Training</b>",tickfont =␣
 ↪dict(size=18),linewidth=0.5, linecolor='black',
                  mirror=True)
fig.update_yaxes(title_text="<b>Performance</b>",linewidth=0.5,␣
 ↪linecolor='black',
                  mirror=True)
fig.show()
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_boxplot.png')
Image('models_boxplot.png')
```



Metric1=[‘RF’ for i in range(len(RF))] Metric2=[‘XGBoost’ for i in range(len(ANN))] Model=Metric1+Metric2 Training=x+x Values=RF+ANN len(Values)

#Violin plot which also show the density of the distribution import plotly.express as px Metric1=[‘RF’ for i in range(len(RF))] Metric2=[‘XGBoost’ for i in range(len(ANN))] Model=Metric1+Metric2 Training=x+x Values=RF+ANN lst=[[Training[i],Values[i],Model[i]] for i in range(len(Model))] df = pd.DataFrame(lst, columns =[‘Training Days’, ‘Pearson correlation (r)’,‘Model’])

#fig = px.violin(df,y=“Performance”, x=“Calibration Model”, color=‘Metric’, box=True,points=“all”, #hover_data=df.columns) fig = px.violin(df,y=“Pearson correlation (r)”, x=“Training Days”, color=‘Model’, box=True, hover_data=df.columns)

fig.update_layout(autosize=False, width=900, height=500) fig.show() #chart_studio.plotly.sign_in(‘vinylango’, ‘gybbJVWfRSUoTcRRSa6J’) #chart_studio.plotly.image.save_as(fig, filename=‘models_violinplots.png’) #Image(‘models_violinplots.png’)

23 Seasonal Calibration Scheme

24 Fall

25 RF

[52]:
```
frame1=[data_oct,data_nov]
fall=pd.concat(frame1)
#fall=fall.sample(frac=1)
Day=[5*i for i in range(1,11) ]
X=fall[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=fall['Ref']
mean1=np.std(y)
Rmse7_rf=[]
RMSE7_rf=[]
for i in range(1,11):
    regressor.fit(X[:120*i].drop(['Lab1'], axis=1), y[:120*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse7_rf.append(mse)
    RMSE7_rf.append(rmse)
```

26 XGBoost

[53]:
```
Rmse7_ann=[]
RMSE7_ann=[]
for i in range(1,11):
    model.fit(X[:120*i].drop(['Lab1'], axis=1), y[:120*i])
    pred=model.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse7_ann.append(mse)
    RMSE7_ann.append(rmse)
```

27 Winter

28 RF

[54]:
```
frame1=[data_dec,data_jan,data_feb]
winter=pd.concat(frame1)
#winter=winter.sample(frac=1)
X=winter[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
```

```
y=winter['Ref']
mean2=np.std(y)
Rmse8_rf=[]
RMSE8_rf=[]
for i in range(1,11):
    regressor.fit(X[:120*i].drop(['Lab1'], axis=1), y[:120*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse8_rf.append(mse)
    RMSE8_rf.append(rmse)
```

29 XGBoost

[55]:
```
Rmse8_ann=[]
RMSE8_ann=[]
for i in range(1,11):
    model.fit(X[:120*i].drop(['Lab1'], axis=1), y[:120*i])
    pred=model.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse8_ann.append(mse)
    RMSE8_ann.append(rmse)
```

30 Spring

31 RF

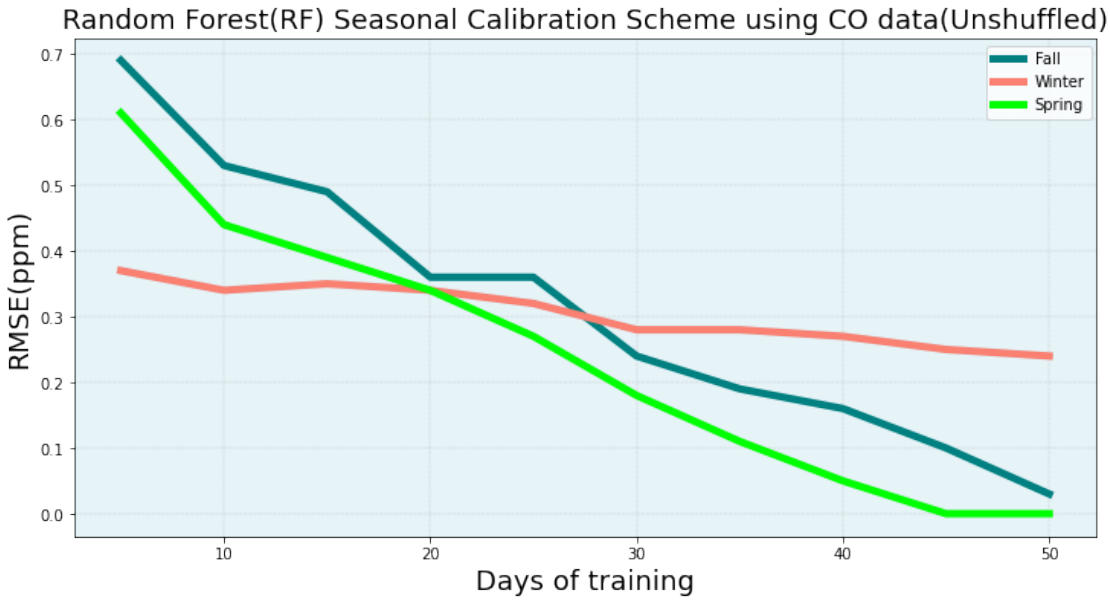
[56]:
```
frame1=[data_mar,data_apr]
spring=pd.concat(frame1)
#spring=spring.sample(frac=1)
X=spring[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=spring['Ref']
mean3=np.std(y)
Rmse9_rf=[]
RMSE9_rf=[]
for i in range(1,11):
    regressor.fit(X[:120*i].drop(['Lab1'], axis=1), y[:120*i])
    pred=regressor.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse9_rf.append(mse)
```

```
    RMSE9_rf.append(rmse)
```

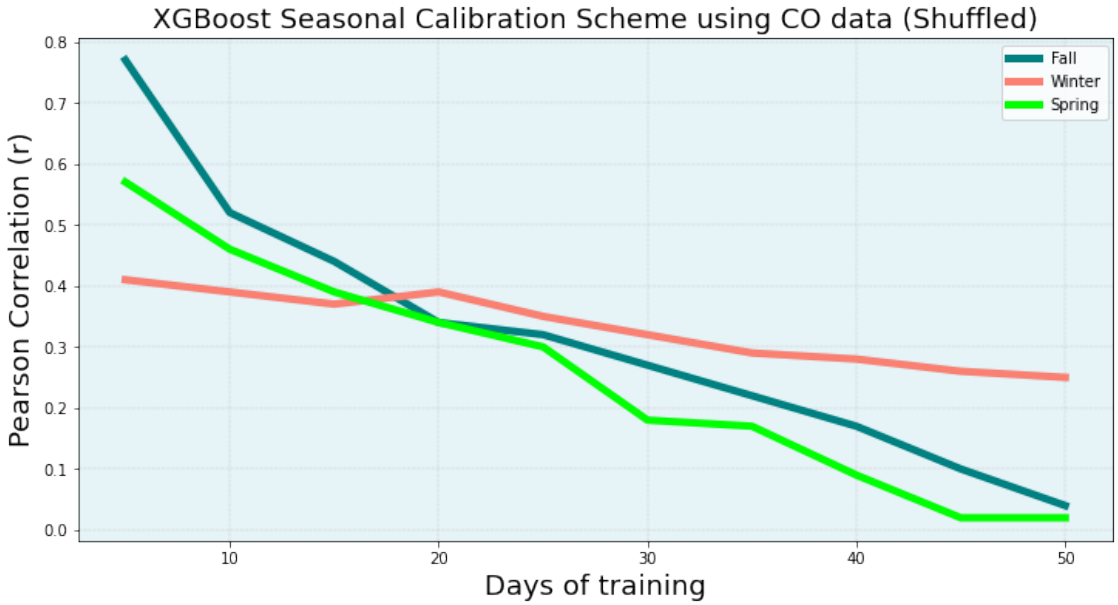
32 XGBoost

[57]:
```
Rmse9_ann=[]
RMSE9_ann=[]
for i in range(1,11):
    model.fit(X[:120*i].drop(['Lab1'], axis=1), y[:120*i])
    pred=model.predict(X.drop(['Lab1'], axis=1))
    y_test=y
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    #rmse= mape=round(mean_absolute_percentage_error(y_test,pred),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse9_ann.append(mse)
    RMSE9_ann.append(rmse)
```

[58]:
```
fig= plt.figure(figsize=(12,6))
ax = fig.add_subplot(111)
ax.patch.set_facecolor('lightblue')
ax.patch.set_alpha(0.3)
plt.plot(Day,Rmse7_rf,color='teal',linewidth=5)
plt.plot(Day,Rmse8_rf,color='salmon',linewidth=5)
plt.plot(Day,Rmse9_rf,color='lime',linewidth=5)

plt.legend(['Fall', 'Winter','Spring'],
           loc = 2, bbox_to_anchor = (0.885,1))

plt.xlabel('Days of training',fontsize=18)
plt.ylabel('RMSE(ppm)',fontsize=18)
plt.title('Random Forest(RF) Seasonal Calibration Scheme using CO␣
 ↪data(Unshuffled)',fontsize=18)
plt.grid(linestyle='-.',linewidth=0.3)
```



[59]:
```
fig= plt.figure(figsize=(12,6))
ax = fig.add_subplot(111)
ax.patch.set_facecolor('lightblue')
ax.patch.set_alpha(0.3)
plt.plot(Day,Rmse7_ann,color='teal',linewidth=5)
plt.plot(Day,Rmse8_ann,color='salmon',linewidth=5)
plt.plot(Day,Rmse9_ann,color='lime',linewidth=5)

plt.legend(['Fall', 'Winter','Spring'],
           loc = 2, bbox_to_anchor = (0.885,1))

plt.xlabel('Days of training',fontsize=18)
plt.ylabel('Pearson Correlation (r)',fontsize=18)
plt.title('XGBoost Seasonal Calibration Scheme using CO data␣
 ↪(Shuffled)',fontsize=18)
plt.grid(linestyle='-.',linewidth=0.3)
```



[60]:
```
#import chart_studio
import plotly.express as px
from IPython.display import Image
import plotly.graph_objects as go
import numpy as np

Day_1_rf=[RMSE7_rf[0],RMSE8_rf[0],RMSE9_rf[0]]
Day_2_rf=[RMSE7_rf[1],RMSE8_rf[1],RMSE9_rf[1]]
Day_3_rf=[RMSE7_rf[2],RMSE8_rf[2],RMSE9_rf[2]]
Day_4_rf=[RMSE7_rf[3],RMSE8_rf[3],RMSE9_rf[3]]
Day_5_rf=[RMSE7_rf[4],RMSE8_rf[4],RMSE9_rf[4]]
Day_6_rf=[RMSE7_rf[5],RMSE8_rf[5],RMSE9_rf[5]]
Day_7_rf=[RMSE7_rf[6],RMSE8_rf[6],RMSE9_rf[6]]
Day_8_rf=[RMSE7_rf[7],RMSE8_rf[7],RMSE9_rf[7]]
Day_9_rf=[RMSE7_rf[8],RMSE8_rf[8],RMSE9_rf[8]]
Day_10_rf=[RMSE7_rf[9],RMSE8_rf[9],RMSE9_rf[9]]

Day_1_ann=[RMSE7_ann[0],RMSE8_ann[0],RMSE9_ann[0]]
Day_2_ann=[RMSE7_ann[1],RMSE8_ann[1],RMSE9_ann[1]]
Day_3_ann=[RMSE7_ann[2],RMSE8_ann[2],RMSE9_ann[2]]
Day_4_ann=[RMSE7_ann[3],RMSE8_ann[3],RMSE9_ann[3]]
Day_5_ann=[RMSE7_ann[4],RMSE8_ann[4],RMSE9_ann[4]]
Day_6_ann=[RMSE7_ann[5],RMSE8_ann[5],RMSE9_ann[5]]
Day_7_ann=[RMSE7_ann[6],RMSE8_ann[6],RMSE9_ann[6]]
Day_8_ann=[RMSE7_ann[7],RMSE8_ann[7],RMSE9_ann[7]]
Day_9_ann=[RMSE7_ann[8],RMSE8_ann[8],RMSE9_ann[8]]
Day_10_ann=[RMSE7_ann[9],RMSE8_ann[9],RMSE9_ann[9]]
```

[61]:
```
Day_1_RF=[Rmse7_rf[0],Rmse8_rf[0],Rmse9_rf[0]]
Day_2_RF=[Rmse7_rf[1],Rmse8_rf[1],Rmse9_rf[1]]
Day_3_RF=[Rmse7_rf[2],Rmse8_rf[2],Rmse9_rf[2]]
Day_4_RF=[Rmse7_rf[3],Rmse8_rf[3],Rmse9_rf[3]]
Day_5_RF=[Rmse7_rf[4],Rmse8_rf[4],Rmse9_rf[4]]
Day_6_RF=[Rmse7_rf[5],Rmse8_rf[5],Rmse9_rf[5]]
Day_7_RF=[Rmse7_rf[6],Rmse8_rf[6],Rmse9_rf[6]]
Day_8_RF=[Rmse7_rf[7],Rmse8_rf[7],Rmse9_rf[7]]
Day_9_RF=[Rmse7_rf[8],Rmse8_rf[8],Rmse9_rf[8]]
Day_10_RF=[Rmse7_rf[9],Rmse8_rf[9],Rmse9_rf[9]]

Day_1_ANN=[Rmse7_ann[0],Rmse8_ann[0],Rmse9_ann[0]]
Day_2_ANN=[Rmse7_ann[1],Rmse8_ann[1],Rmse9_ann[1]]
Day_3_ANN=[Rmse7_ann[2],Rmse8_ann[2],Rmse9_ann[2]]
Day_4_ANN=[Rmse7_ann[3],Rmse8_ann[3],Rmse9_ann[3]]
Day_5_ANN=[Rmse7_ann[4],Rmse8_ann[4],Rmse9_ann[4]]
Day_6_ANN=[Rmse7_ann[5],Rmse8_ann[5],Rmse9_ann[5]]
Day_7_ANN=[Rmse7_ann[6],Rmse8_ann[6],Rmse9_ann[6]]
Day_8_ANN=[Rmse7_ann[7],Rmse8_ann[7],Rmse9_ann[7]]
Day_9_ANN=[Rmse7_ann[8],Rmse8_ann[8],Rmse9_ann[8]]
Day_10_ANN=[Rmse7_ann[9],Rmse8_ann[9],Rmse9_ann[9]]
```

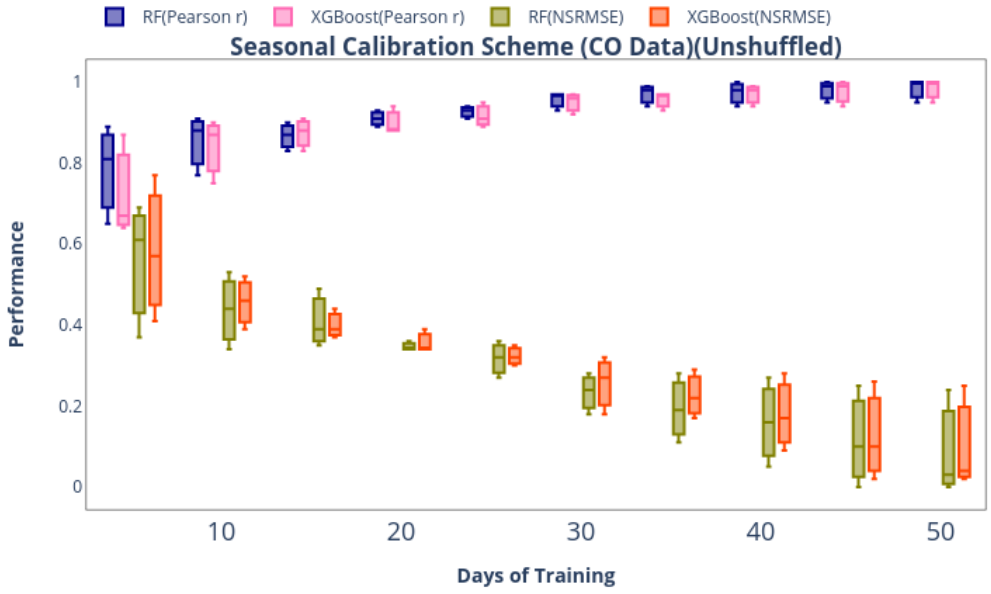
[62]:
```
RF_P=Day_1_rf+Day_2_rf+Day_3_rf+Day_4_rf+Day_5_rf+Day_6_rf+Day_7_rf+Day_8_rf+Day_9_rf+Day_10_r
XGBoost_P=Day_1_ann+Day_2_ann+Day_3_ann+Day_4_ann+Day_5_ann+Day_6_ann+Day_7_ann+Day_8_ann+Day_
RF_R=Day_1_RF+Day_2_RF+Day_3_RF+Day_4_RF+Day_5_RF+Day_6_RF+Day_7_RF+Day_8_RF+Day_9_RF+Day_10_R
XGBoost_R=Day_1_ANN+Day_2_ANN+Day_3_ANN+Day_4_ANN+Day_5_ANN+Day_6_ANN+Day_7_ANN+Day_8_ANN+Day_

x1=['5' for i in range(3)]
x2=['10' for i in range(3)]
x3=['15' for i in range(3)]
x4=['20' for i in range(3)]
x5=['25' for i in range(3)]
x6=['30' for i in range(3)]
x7=['35' for i in range(3)]
x8=['40' for i in range(3)]
x9=['45' for i in range(3)]
x10=['50' for i in range(3)]
x=x1+x2+x3+x4+x5+x6+x7+x8+x9+x10
```

[63]:
```
fig = go.Figure()
x = x
fig.add_trace(go.Box(

    # defining y axis in corresponding
    # to x-axis
    y=RF_P,
    x=x,
```

```
    name='RF(Pearson r)',
    marker_color='darkblue',
    showlegend=True
))
fig.add_trace(go.Box(
    y=XGBoost_P,
    x=x,
    name='XGBoost(Pearson r)',
    marker_color='hotpink',
    showlegend=True

))

fig.add_trace(go.Box(

    # defining y axis in corresponding
    # to x-axis
    y=RF_R,
    x=x,
    name='RF(NSRMSE)',
    marker_color='olive',
    showlegend=True
))
fig.add_trace(go.Box(
    y=XGBoost_R,
    x=x,
    name='XGBoost(NSRMSE)',
    marker_color='orangered',
    showlegend=True

))
fig.update_layout(autosize=False,
                  title={'text': "<b>Seasonal Calibration Scheme (CO␣
↪Data)(Unshuffled)</b>",
        'y':0.83,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
    width=800,
    height=500,
     legend=dict( yanchor="bottom",
    y=1.05,
    orientation="h"
),

    # 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>Days of Training</b>",tickfont = dict(size=18),␣
 ↪linecolor='black',
                 mirror=True,linewidth=0.5)
fig.update_yaxes(title_text="<b>Performance</b>",linecolor='black',
                 mirror=True,linewidth=0.5)
chart_studio.plotly.sign_in('vinylango', 'gybbJVWfRSUoTcRRSa6J')
chart_studio.plotly.image.save_as(fig, filename='models_boxplot.png')
Image('models_boxplot.png')
```

[63]:



[64]:

```
df1=[x for _, x in CO_Data.groupby('Month')]
data_oct=df1[4]
#data_oct=data_oct.sample(frac=1)
data_nov=df1[5]
#data_nov=data_nov.sample(frac=1)
data_dec=df1[6]
#data_dec=data_dec.sample(frac=1)
data_jan=df1[0]
#data_jan=data_jan.sample(frac=1)
data_feb=df1[1]
#data_feb=data_feb.sample(frac=1)
```

```
data_mar=df1[2]
#data_mar=data_mar.sample(frac=1)
data_apr=df1[3]
#data_apr=data_apr.sample(frac=1)
data=[data_oct,data_nov,data_dec,data_jan,data_feb,data_mar]
frame1=[data_dec]#,data_nov,data_dec,data_jan,data_feb,data_mar,data_apr]
fall=pd.concat(frame1)
fall.shape
```

[64]:
```
(741, 9)
```

[65]:
```
fall_train=fall[:5760]
fall_train=fall_train.sample(frac=1)
fall_test=fall[6000:]

Day=[i for i in range(1,21)]
X_train=fall_train[['Net␣
 ↪Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y_train=fall_train['Ref']
X_test=fall_test[['Net␣
 ↪Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y_test=fall_test['Ref']

mean1=np.std(y)
Rmse7_rf=[]
RMSE7_rf=[]
for i in range(1,21):
    regressor.fit(X_train[:288*i].drop(['Lab1'], axis=1), y_train[:288*i])
    pred=regressor.predict(X_test.drop(['Lab1'], axis=1))
    #y_test=y_test
    for i in range(len(y_test)):
        if y_test[i]>3000:
            y_test[i]=np.mean(y_test)
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse7_rf.append(mse)
    RMSE7_rf.append(rmse)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-65-65a7f537c830> in <module>
     14 for i in range(1,21):
     15     regressor.fit(X_train[:288*i].drop(['Lab1'], axis=1), y_train[:
 ↪288*i])
---> 16     pred=regressor.predict(X_test.drop(['Lab1'], axis=1))
     17     #y_test=y_test
     18     for i in range(len(y_test)):
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/
↪sklearn/ensemble/_forest.py in predict(self, X)
    781         check_is_fitted(self)
    782         # Check data
--> 783         X = self._validate_X_predict(X)
    784
    785         # Assign chunk of trees to jobs

/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/
↪sklearn/ensemble/_forest.py in _validate_X_predict(self, X)
    419         check_is_fitted(self)
    420
--> 421         return self.estimators_[0]._validate_X_predict(X,␣
↪check_input=True)
    422
    423     @property

/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/
↪sklearn/tree/_classes.py in _validate_X_predict(self, X, check_input)
    386         """Validate X whenever one tries to predict, apply,␣
↪predict_proba"""
    387         if check_input:
--> 388             X = check_array(X, dtype=DTYPE, accept_sparse="csr")
    389             if issparse(X) and (X.indices.dtype != np.intc or
    390                                 X.indptr.dtype != np.intc):

/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/
↪sklearn/utils/validation.py in inner_f(*args, **kwargs)
     71                             FutureWarning)
     72         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 73         return f(**kwargs)
     74     return inner_f
     75

/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/
↪sklearn/utils/validation.py in check_array(array, accept_sparse,␣
↪accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d,␣
↪allow_nd, ensure_min_samples, ensure_min_features, estimator)
    649         n_samples = _num_samples(array)
    650         if n_samples < ensure_min_samples:
--> 651             raise ValueError("Found array with %d sample(s) (shape=%s)␣
↪while a"
    652                              " minimum of %d is required%s."
    653                              % (n_samples, array.shape,␣
↪ensure_min_samples,
```

```
ValueError: Found array with 0 sample(s) (shape=(0, 7)) while a minimum of 1 is
↪required.
```

[ ]:
```
index=[i for i in range(len(y_test))]
plt.plot(index[:100],y_test[:100])
plt.plot(index[:100],pred[:100])
```

[ ]:
```
df1=[x for _, x in CO_Data.groupby('Month')]
data_oct=df1[4]
#data_oct=data_oct.sample(frac=1)
data_nov=df1[5]
#data_nov=data_nov.sample(frac=1)
data_dec=df1[6]
#data_dec=data_dec.sample(frac=1)
data_jan=df1[0]
#data_jan=data_jan.sample(frac=1)
data_feb=df1[1]
#data_feb=data_feb.sample(frac=1)
data_mar=df1[2]
#data_mar=data_mar.sample(frac=1)
data_apr=df1[3]
#data_apr=data_apr.sample(frac=1)
data=[data_oct,data_nov,data_dec,data_jan,data_feb,data_mar,data_apr]
frame2=[data_dec]#,data_nov,data_dec,data_jan,data_feb,data_mar,data_apr]
fall2=pd.concat(frame2)
fall2=fall2.sample(frac=1)
```

[ ]:
```
from sklearn.model_selection import train_test_split
X=fall2[['Net Signal','Lab1','Temp','RH','Month','Day_of_week','Day','Hour']]
y=fall2['Ref']
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.069)
mean1=np.std(y)
Rmse7_rf=[]
RMSE8_rf=[]
for i in range(1,21):
    regressor.fit(X[:288*i].drop(['Lab1'], axis=1), y[:288*i])
    pred=regressor.predict(X[6000:].drop(['Lab1'], axis=1))
    y_test=y[6000:]
    for i in range(len(y_test)):
        if y_test[i]>3000:
            y_test[i]=np.mean(y_test)
    mse=round(np.sqrt(sm.mean_squared_error(y_test, pred))/np.mean(y_test),2)
    rmse=round(np.corrcoef(y_test, pred)[0, 1],2)
    Rmse7_rf.append(mse)
    RMSE8_rf.append(rmse)
```

[ ]:
```
index=[i for i in range(len(y_test))]
plt.plot(index[:100],y_test[:100])
plt.plot(index[:100],pred[:100])
```

[ ]:
```
fig= plt.figure(figsize=(12,6))
ax = fig.add_subplot(111)
#ax.patch.set_facecolor('lightblue')
#ax.patch.set_alpha(0.3)
#plt.plot(Day,np.array(Rmse7_rf)+0.6,color='teal',linewidth=2)
plt.plot(Day,RMSE7_rf,color='salmon',linewidth=2)
plt.plot(Day,RMSE8_rf,color='teal',linewidth=2)
plt.legend(['Non-randomized', 'Randomized'], loc = 2, bbox_to_anchor = (0,1))
plt.xlabel('Days of training',fontsize=18)
plt.ylabel('Pearson r',fontsize=18)
plt.title('Random Forest(RF) 6-Month Calibration Scheme using CO␣
 ↪data',fontsize=18)
#plt.grid(linestyle='-.',linewidth=0.3)
```

[ ]:
```
max(y_test)
```

[ ]:
```
max(pred)
```

[ ]:
```

```