# MO432 - Exercício 2

## Eliézer Zarpelão - RA 141320

```
import pandas as pd
import numpy as np
```

# 1) Leia

```
!wget /resources/data/Bias_correction_ucl.csv https://archive.ics.uci.edu/ml/machine-learn
```

```
/resources/data/Bias_correction_ucl.csv: Scheme missing.
--2021-06-14 02:18:59--  https://archive.ics.uci.edu/ml/machine-learning-databases/0(
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... conne(
HTTP request sent, awaiting response... 200 OK
Length: 1658519 (1.6M) [application/x-httpd-php]
Saving to: ‘Bias_correction_ucl.csv.3’

Bias_correction_ucl 100%[===================>]   1.58M  3.52MB/s    in 0.4s

2021-06-14 02:19:00 (3.52 MB/s) - ‘Bias_correction_ucl.csv.3’ saved [1658519/1658519]

FINISHED --2021-06-14 02:19:00--
Total wall clock time: 0.7s
Downloaded: 1 files, 1.6M in 0.4s (3.52 MB/s)
```

### Colums names

1. station - used weather station number: 1 to 25
2. Date - Present day: yyyy-mm-dd ('2013-06-30' to '2017-08-30')
3. Present_Tmax - Maximum air temperature between 0 and 21 h on the present day (Â°C): 20 to 37.6
4. Present_Tmin - Minimum air temperature between 0 and 21 h on the present day (Â°C): 11.3 to 29.9
5. LDAPS_RHmin - LDAPS model forecast of next-day minimum relative humidity (%): 19.8 to 98.5
6. LDAPS_RHmax - LDAPS model forecast of next-day maximum relative humidity (%): 58.9 to 100
7. LDAPS_Tmax_lapse - LDAPS model forecast of next-day maximum air temperature applied lapse rate (Â°C): 17.6 to 38.5

8. LDAPS_Tmin_lapse - LDAPS model forecast of next-day minimum air temperature applied lapse rate (Â°C): 14.3 to 29.6
9. LDAPS_WS - LDAPS model forecast of next-day average wind speed (m/s): 2.9 to 21.9
10. LDAPS_LH - LDAPS model forecast of next-day average latent heat flux (W/m2): -13.6 to 213.4
11. LDAPS_CC1 - LDAPS model forecast of next-day 1st 6-hour split average cloud cover (0-5 h) (%): 0 to 0.97
12. LDAPS_CC2 - LDAPS model forecast of next-day 2nd 6-hour split average cloud cover (6-11 h) (%): 0 to 0.97
13. LDAPS_CC3 - LDAPS model forecast of next-day 3rd 6-hour split average cloud cover (12-17 h) (%): 0 to 0.98
14. LDAPS_CC4 - LDAPS model forecast of next-day 4th 6-hour split average cloud cover (18-23 h) (%): 0 to 0.97
15. LDAPS_PPT1 - LDAPS model forecast of next-day 1st 6-hour split average precipitation (0-5 h) (%): 0 to 23.7
16. LDAPS_PPT2 - LDAPS model forecast of next-day 2nd 6-hour split average precipitation (6-11 h) (%): 0 to 21.6
17. LDAPS_PPT3 - LDAPS model forecast of next-day 3rd 6-hour split average precipitation (12-17 h) (%): 0 to 15.8
18. LDAPS_PPT4 - LDAPS model forecast of next-day 4th 6-hour split average precipitation (18-23 h) (%): 0 to 16.7
19. lat - Latitude (Â°): 37.456 to 37.645
20. lon - Longitude (Â°): 126.826 to 127.135
21. DEM - Elevation (m): 12.4 to 212.3
22. Slope - Slope (Â°): 0.1 to 5.2
23. Solar radiation - Daily incoming solar radiation (wh/m2): 4329.5 to 5992.9
24. Next_Tmax - The next-day maximum air temperature (Â°C): 17.4 to 38.9
25. Next_Tmin - The next-day minimum air temperature (Â°C): 11.3 to 29.8

```
path_file = "Bias_correction_ucl.csv"
header_list = ["station","date","present_tmax","present_tmin","ldaps_rhmin","ldaps_rhmax",
data = pd.read_csv(path_file,sep=',',names=header_list,header=None, skiprows=1)

# remova a coluna Next_Tmin
data = data.drop(["next_tmin"], axis=1)

# remova a coluna Date
data = data.drop(["date"], axis=1)

# remova as linhas que tem valor faltante
data = data.dropna()
```

```
data.shape[0]
```

```
7588
```

```
# o atributo de saída é Next_Tmax
data_Y = data[["next_tmax"]].copy()

# remova a coluna next_tmax
data = data.drop(["next_tmax"], axis=1)
```

```
data.head()
```

| | station | present_tmax | present_tmin | ldaps_rhmin | ldaps_rhmax | ldaps_tmax_lapse |
|---|---|---|---|---|---|---|
| **0** | 1.0 | 28.7 | 21.4 | 58.255688 | 91.116364 | 28.074101 |
| **1** | 2.0 | 31.9 | 21.6 | 52.263397 | 90.604721 | 29.850689 |
| **2** | 3.0 | 31.6 | 23.3 | 48.690479 | 83.973587 | 30.091292 |
| **3** | 4.0 | 32.0 | 23.4 | 58.239788 | 96.483688 | 29.704629 |
| **4** | 5.0 | 31.4 | 21.9 | 56.174095 | 90.155128 | 29.113934 |

```
data_Y.head()
```

| | next_tmax |
|---|---|
| **0** | 29.1 |
| **1** | 30.5 |
| **2** | 31.1 |
| **3** | 31.7 |
| **4** | 31.2 |

```
from sklearn.preprocessing import Normalizer
# centralize e normalize cada atributo de entrada
data = pd.DataFrame(Normalizer().fit(data).transform(data))
data.head()
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000167 | 0.004784 | 0.003567 | 0.009710 | 0.015187 | 0.004679 | 0.003835 | 0.001137 | 0.011 |
| **1** | 0.000341 | 0.005432 | 0.003678 | 0.008900 | 0.015429 | 0.005083 | 0.004093 | 0.000969 | 0.008 |
| **2** | 0.000511 | 0.005387 | 0.003972 | 0.008300 | 0.014315 | 0.005130 | 0.004188 | 0.001046 | 0.003 |
| **3** | 0.000683 | 0.005460 | 0.003993 | 0.009938 | 0.016464 | 0.005069 | 0.003980 | 0.000964 | 0.011 |
| **4** | 0.000853 | 0.005355 | 0.003735 | 0.009581 | 0.015376 | 0.004965 | 0.004006 | 0.000978 | 0.018 |

# 2) Cross validation, medida de erro e busca de hiperparametros

## Linear

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression

model  = LinearRegression()
kfold  = KFold(n_splits=5, shuffle=True)

RMSE = np.mean(- cross_val_score(model, data, data_Y, cv=kfold, scoring='neg_root_mean_squ

print("RSME Linear: {0}".format(RMSE))
```

```
     RSME Linear: 1.4925792825581627
```

## Linear com regularização L2

```
import random
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import Lasso

model  = Lasso()
kfold  = KFold(n_splits=5, shuffle=True)
L2_custom_RMSE = 99999

for i in range(10):
    alpha = 10**(random.random() * 6 - 3)
    RMSE = np.mean(- cross_val_score(Lasso(alpha=alpha), data, data_Y, cv=5, scoring='neg_
    if RMSE < L2_custom_RMSE:
        L2_custom_RMSE = RMSE
        custom_params_alpha = alpha

print("L2 RMSE ", L2_custom_RMSE)
print("alpha ", custom_params_alpha)
```

```
     L2 RMSE  3.096779348667689
     alpha  0.005247440994966936
```

## Linear com regularização L1

```
import random
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import Ridge

model  = Ridge()
kfold  = KFold(n_splits=5, shuffle=True)
L1_custom_RMSE = 99999

for i in range(10):
    alpha = 10**(random.random() * 6 - 3)
    RMSE = np.mean(- cross_val_score(Ridge(alpha=alpha), data, data_Y, cv=5, scoring='neg_
    if RMSE < L1_custom_RMSE:
        L1_custom_RMSE = RMSE
        custom_params_alpha = alpha

print("L1 RMSE ", L1_custom_RMSE)
print("alpha ", custom_params_alpha)
```

```
L1 RMSE  1.8952926939333714
alpha  0.001997132102727603
```

## SVM Linear

```
import random
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.svm import LinearSVR

kfold  = KFold(n_splits=5, shuffle=True)
SVM_Linear_RMSE = 99999

for i in range(10):
  custom_C = 2**((random.random() * 10) - 5) #tive que reduzir pra 10 porque o Colab arreg
  model  = LinearSVR(C=custom_C, epsilon=0.1, max_iter=5000000)

  RMSE = np.mean(-cross_val_score(model, data, data_Y.values.ravel(), cv=5, scoring='neg_r

  if RMSE < SVM_Linear_RMSE:
    SVM_Linear_RMSE = RMSE
    custom_params_C = custom_C

print("SVM_Linear_RMSE ", SVM_Linear_RMSE)
print("C ", custom_params_C)
```

```
SVM_Linear_RMSE  2.8626185154057273
C  23.916270907697438
```

## SVM com kernel RBF

```
import random

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.svm import SVC

kfold  = KFold(n_splits=5, shuffle=True)
SVM_KernelRBF_RMSE = 99999

for i in range(10):
  custom_C = 2**((random.random() * 10) - 5) #tive que reduzir pra 10 porque o Colab arreg
  custom_gamma = 2**((random.random() * 3) - 9)
  model = SVC(C=custom_C, gamma=custom_gamma , max_iter=5000000)

  RMSE = np.mean(-cross_val_score(model, data, data_Y.values.ravel(), cv=5, scoring='neg_r
  if RMSE < SVM_Linear_RMSE:
    SVM_KernelRBF_RMSE = RMSE
    custom_params_C = custom_C
    custom_params_gamma = custom_gamma

print("SVM_KernelRBF_RMSE ", SVM_KernelRBF_RMSE)
print("C ", custom_params_C)
print("Gamma ", custom_gamma)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'


  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'
```

```
  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536:
ValueError: Unknown label type: 'continuous'
```

# KNN

```
import random

from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsRegressor

KNN_RMSE = 99999

for i in range(10):
  custom_K = random.randint(1, 1000)
  model = KNeighborsRegressor(n_neighbors=custom_K)

  RMSE = np.mean(-cross_val_score(model, data, data_Y['next_tmax'], cv=5, scoring='neg_roo
  if RMSE < KNN_RMSE:
    KNN_RMSE = RMSE
    custom_params_K = custom_K


print("KNN_RMSE ", KNN_RMSE)
print("K ", custom_K)
```

```
KNN_RMSE  2.405364636315292
K  516
```

## MLP

```
from sklearn.model_selection import cross_val_score
from sklearn.neural_network import MLPRegressor

model = MLPRegressor(hidden_layer_sizes=(5,8,11,14,17,20), max_iter=30000,activation = 're

RMSE = np.mean(-cross_val_score(model, data, data_Y['next_tmax'], cv=5, scoring='neg_root_

print("MLP_RMSE ", RMSE)
```

```
MLP_RMSE  1.7240844348646172
```

## Arvore de decisão

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeClassifier()

Tree_RMSE = 99999

for i in range(10):
  custom_alpha = random.random() / 25
  model = DecisionTreeRegressor(random_state=0, ccp_alpha=custom_alpha)

  RMSE = np.mean(-cross_val_score(model, data, data_Y['next_tmax'], cv=5, scoring='neg_roo
  if RMSE < Tree_RMSE:
    Tree_RMSE = RMSE
    custom_params_alpha = custom_alpha


print("Tree_RMSE ", KNN_RMSE)
print("Alpha ", custom_params_alpha)
```

```
Tree_RMSE  2.405364636315292
Alpha  0.013699558389574911
```

## Random Forest

## GBM

✓ 16s conclusão: 23:54 ● ×