

Desafio:

Precisamos que você crie um repositório no github ou gitlab e inicie um projeto para o problema que vai ser apresentado a seguir. Pode colocar nesse repositório seu código em Python ou um jupyter notebook. É importante que adicione em seu README os passos necessários para que possamos executar e verificar sua implementação. Tendo em mente a base de dados Bias correction of numerical prediction model temperature forecast Data Set (<https://archive.ics.uci.edu/ml/datasets/Bias+correction+of+numerical+prediction+model+temperature+forecast>) da UCI, queremos

que você faça uma análise descritiva dos dados explicando aspectos fundamentais, tais como a distribuição dos dados e correlação entre as variáveis. Fique à vontade para fazer sugestões, alterações e transformações que achar necessárias a fim de facilitar o entendimento do problema.

Além da análise descritiva, queremos que você treine e avalie um modelo de Machine Learning para fazer a estimativa da variável Next\_Tmax, explicando a qualidade do modelo a partir das métricas de avaliação utilizadas. Como métricas de avaliação, sugerimos como obrigatórias a accuracy, precision e recall. Mas você fica livre de avaliar outras métricas que sejam oportunas de acordo com seu julgamento e necessidade na avaliação do modelo. Queremos que você nos mostre o que fez no código, mas também explique suas decisões e conclusões, pode gerar gráficos ou outras visualizações para auxiliar em sua explicação. Queremos ver também sua linha de raciocínio.

Por fim, queremos ouvir de você o que poderia ser feito de diferente se tivesse mais tempo e mais recursos. Quais outros testes e análises poderiam ser feitas e qual a importância destas.

Não é necessário pensar nesse momento em eficiência ou otimização de código e modelos, mas se considerar oportuno, pode adicionar esse tópico em sua análise. Fique à vontade para determinar o que é relevante.

Nós precisamos receber a sua solução até o dia 13/01/2022. Em caso de algum imprevisto fique à vontade para nos sinalizar, tudo bem?

Em cópia do e-mail está o @Vitor Casadei em caso de possíveis dúvidas em relação ao desafio.

In [1]:  
`import os, sys, glob`

In [2]:  
`import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from seaborn import pairplot`

In [3]:

```
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
%matplotlib inline

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier, NeighborhoodComponentsAnalysis
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics

import warnings
warnings.filterwarnings("ignore")
```

In [4]:

```
args={ n:nome for n,nome in enumerate(glob.glob('*csv')) }
args
```

Out[4]: {0: 'Bias\_correction\_ucl.csv'}

In [5]:

```
df = pd.read_csv(args[0], sep=',')
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['station', 'Date', 'Present_Tmax', 'Present_Tmin', 'LDAPS_RHmin',
       'LDAPS_RHmax', 'LDAPS_Tmax_lapse', 'LDAPS_Tmin_lapse', 'LDAPS_WS',
       'LDAPS_LH', 'LDAPS_CC1', 'LDAPS_CC2', 'LDAPS_CC3', 'LDAPS_CC4',
       'LDAPS_PPT1', 'LDAPS_PPT2', 'LDAPS_PPT3', 'LDAPS_PPT4', 'lat', 'lon',
       'DEM', 'Slope', 'Solar radiation', 'Next_Tmax', 'Next_Tmin'],
      dtype='object')
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7752 entries, 0 to 7751
Data columns (total 25 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   station          7750 non-null   float64
 1   Date              7750 non-null   object 
 2   Present_Tmax     7682 non-null   float64
 3   Present_Tmin     7682 non-null   float64
 4   LDAPS_RHmin      7677 non-null   float64
 5   LDAPS_RHmax      7677 non-null   float64
 6   LDAPS_Tmax_lapse 7677 non-null   float64
 7   LDAPS_Tmin_lapse 7677 non-null   float64
 8   LDAPS_WS          7677 non-null   float64
 9   LDAPS_LH          7677 non-null   float64
 10  LDAPS_CC1         7677 non-null   float64
 11  LDAPS_CC2         7677 non-null   float64
 12  LDAPS_CC3         7677 non-null   float64
 13  LDAPS_CC4         7677 non-null   float64
 14  LDAPS_PPT1        7677 non-null   float64
 15  LDAPS_PPT2        7677 non-null   float64
```

```
16  LDAPS_PPT3          7677 non-null   float64
17  LDAPS_PPT4          7677 non-null   float64
18  lat                  7752 non-null   float64
19  lon                  7752 non-null   float64
20  DEM                  7752 non-null   float64
21  Slope                7752 non-null   float64
22  Solar radiation     7752 non-null   float64
23  Next_Tmax            7725 non-null   float64
24  Next_Tmin            7725 non-null   float64
dtypes: float64(24), object(1)
memory usage: 1.5+ MB
```

```
In [8]: df = df.loc[~df['Next_Tmax'].isnull()]
```

```
In [9]: df['Next_Tmax'] = df['Next_Tmax'].apply(lambda x: int(round(x)))
```

```
In [10]: df.describe()
```

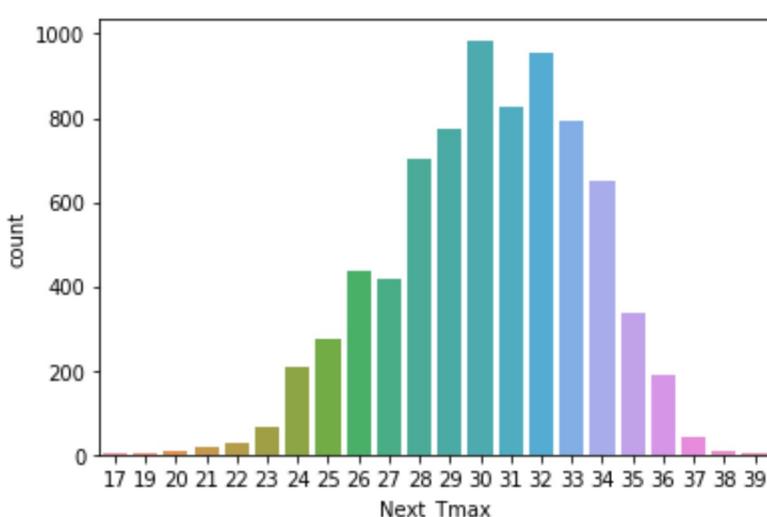
```
Out[10]:
```

	station	Present_Tmax	Present_Tmin	LDAPS_RHmin	LDAPS_RHmax	LDAPS_Tmax_laps
<b>count</b>	7723.000000	7665.000000	7665.000000	7650.000000	7650.000000	7650.000000
<b>mean</b>	13.008157	29.769563	23.224631	56.741402	88.369140	29.61625
<b>std</b>	7.214034	2.970766	2.414959	14.654953	7.199065	2.94661
<b>min</b>	1.000000	20.000000	11.300000	19.794666	58.936283	17.62495
<b>25%</b>	7.000000	27.800000	21.700000	45.961137	84.212961	27.67509
<b>50%</b>	13.000000	29.900000	23.400000	55.017689	89.792492	29.70543
<b>75%</b>	19.000000	32.000000	24.900000	67.154766	93.743328	31.71050
<b>max</b>	25.000000	37.600000	29.900000	98.524734	100.000153	38.54225

8 rows × 24 columns

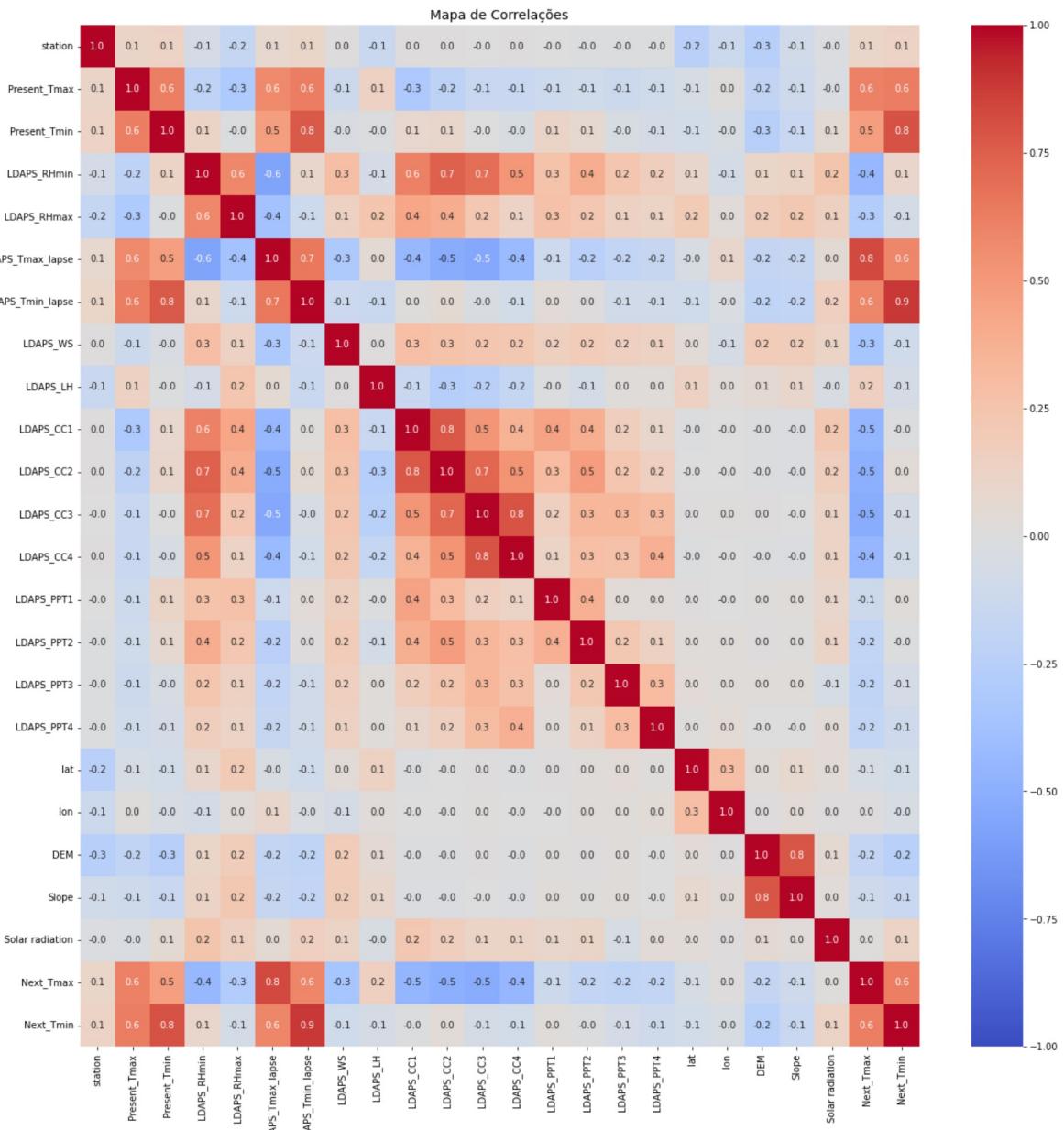
```
In [11]: srn.countplot(df['Next_Tmax'])
```

```
Out[11]: <AxesSubplot:xlabel='Next_Tmax', ylabel='count'>
```



In [12]:

```
f, ax = plt.subplots(figsize = (20,20))
srn.heatmap(df.corr(), annot=True, fmt='.1f',
             ax=ax, cmap='coolwarm', vmin=-1, vmax=1)
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.title('Mapa de Correlações', size=14);
```



In [13]:

```
flt = np.abs(df.corr()['Next_Tmax']) > .59
```

In [14]:

```
flt
```

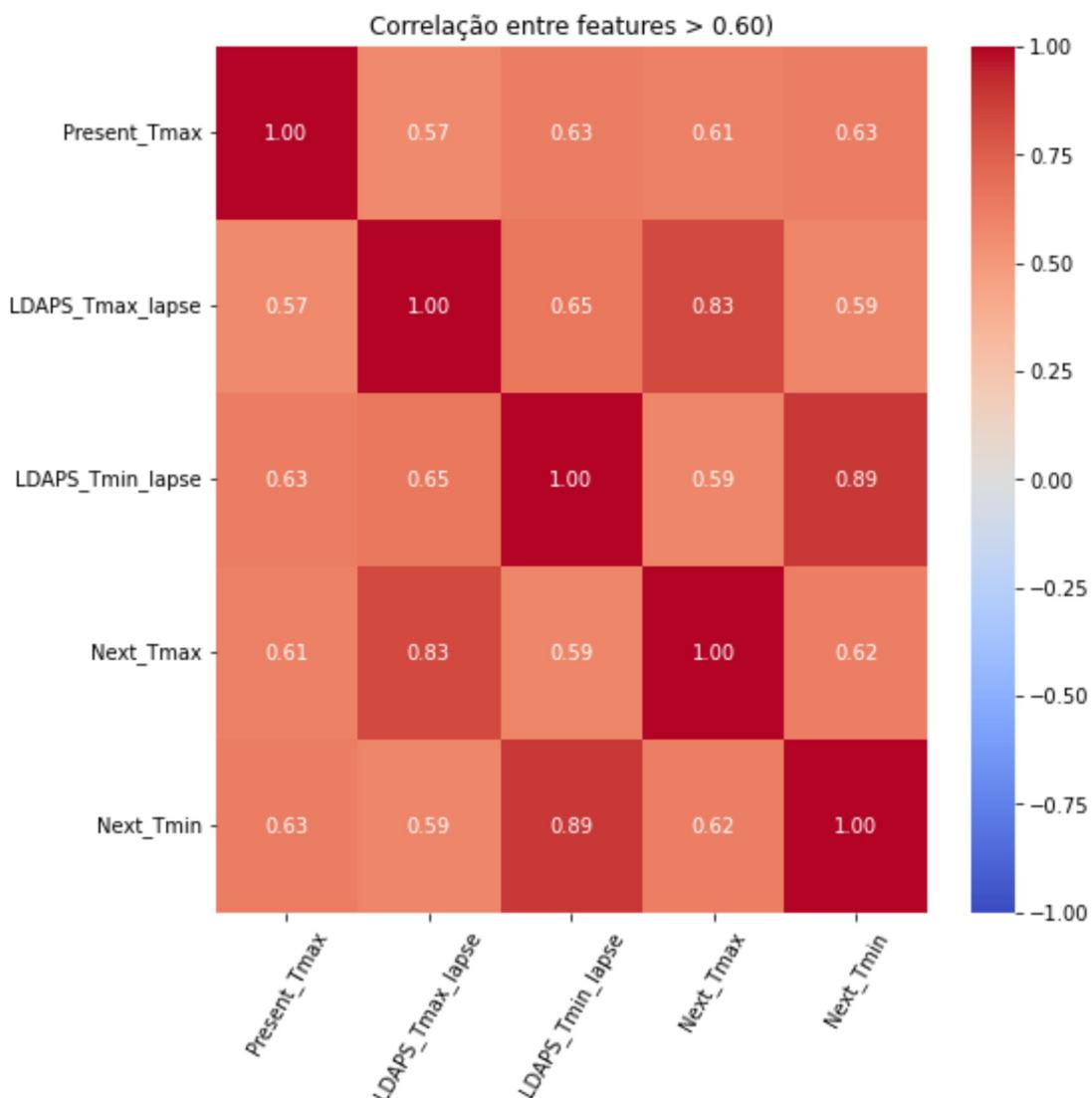
Out[14]:

station	False
Present_Tmax	True
Present_Tmin	False
LDAPS_RHmin	False
LDAPS_RHmax	False
LDAPS_Tmax_lapse	True
LDAPS_Tmin_lapse	True
LDAPS_WS	False
LDAPS_LH	False

```
LDAPS_CC1      False
LDAPS_CC2      False
LDAPS_CC3      False
LDAPS_CC4      False
LDAPS_PPT1     False
LDAPS_PPT2     False
LDAPS_PPT3     False
LDAPS_PPT4     False
lat            False
lon            False
DEM             False
Slope           False
Solar radiation False
Next_Tmax       True
Next_Tmin       True
Name: Next_Tmax, dtype: bool
```

```
In [15]: corr_feat = df.corr().columns[flt].tolist()
```

```
In [16]: f, ax = plt.subplots(figsize = (8,8))
srn.heatmap(df[corr_feat].corr(), annot=True, fmt='.2f',
            ax=ax, cmap='coolwarm', vmin=-1, vmax=1)
plt.xticks(rotation=60)
plt.yticks(rotation=0)
plt.title('Correlação entre features > 0.60');
```



```
In [17]: df= df.loc[~df['Present_Tmax'].isnull()]
```

```
In [18]: df= df.loc[~df['LDAPS_Tmax_lapse'].isnull()]
```

```
In [19]: df= df.loc[~df['LDAPS_Tmin_lapse'].isnull()]
```

```
In [20]: df= df.loc[~df['Next_Tmin'].isnull()]
```

```
In [21]: df.corr()
```

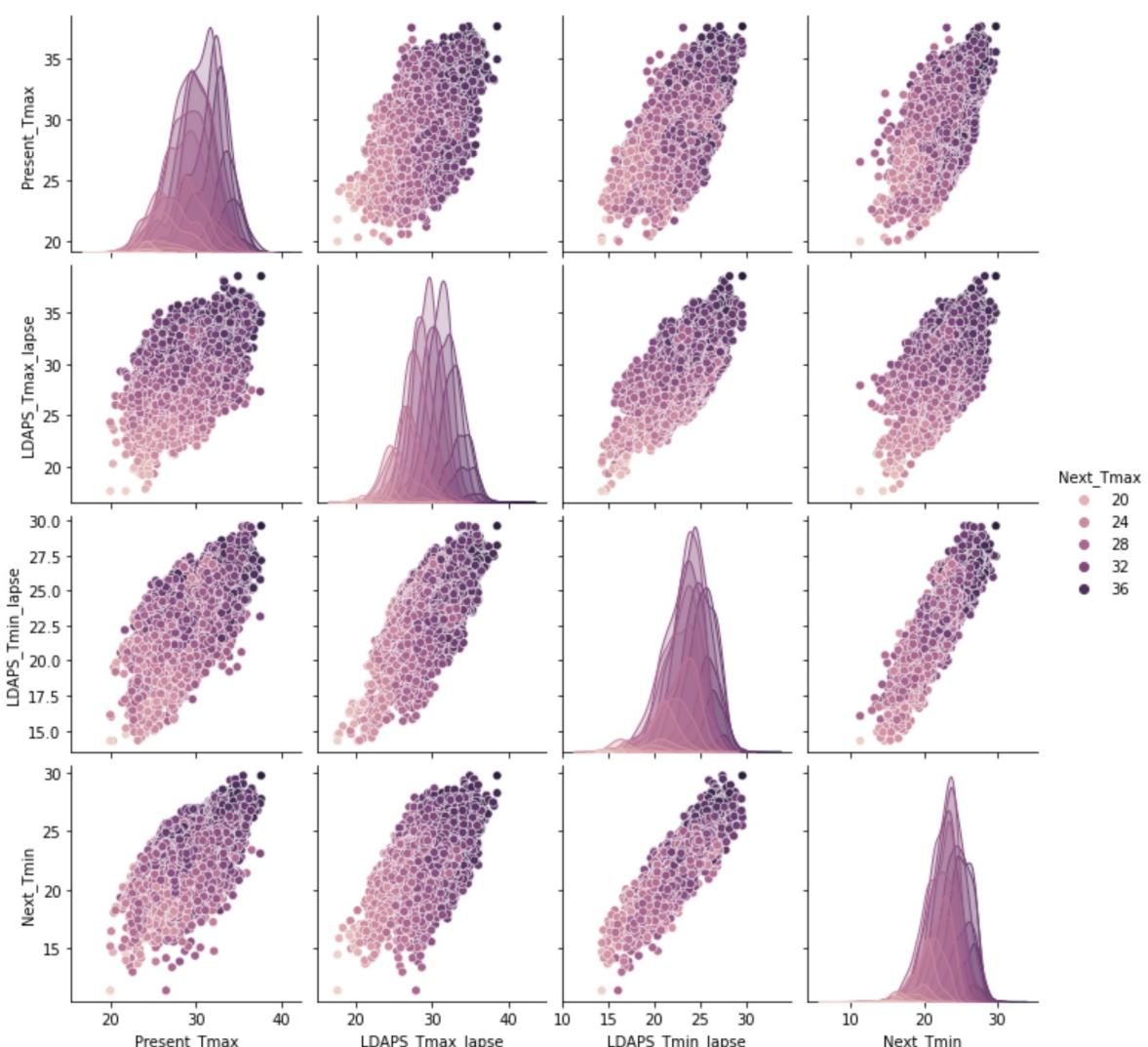
Out[21]:

	station	Present_Tmax	Present_Tmin	LDAPS_RHmin	LDAPS_RHmax	LDAPS
<b>station</b>	1.000000	0.113301	0.133282	-0.067846	-0.169351	
<b>Present_Tmax</b>	0.113301	1.000000	0.616291	-0.206529	-0.303867	
<b>Present_Tmin</b>	0.133282	0.616291	1.000000	0.124516	-0.015260	
<b>LDAPS_RHmin</b>	-0.067846	-0.206529	0.124516	1.000000	0.579035	
<b>LDAPS_RHmax</b>	-0.169351	-0.303867	-0.015260	0.579035	1.000000	
<b>LDAPS_Tmax_lapse</b>	0.069856	0.574947	0.469881	-0.565579	-0.373342	
<b>LDAPS_Tmin_lapse</b>	0.105277	0.629656	0.772811	0.087319	-0.115966	
<b>LDAPS_WS</b>	0.005822	-0.122206	-0.034831	0.291387	0.133150	
<b>LDAPS_LH</b>	-0.132129	0.136401	-0.009656	-0.069574	0.240453	
<b>LDAPS_CC1</b>	0.006539	-0.314411	0.085882	0.613281	0.436418	
<b>LDAPS_CC2</b>	0.003526	-0.215191	0.091572	0.745043	0.391177	
<b>LDAPS_CC3</b>	-0.000161	-0.144658	-0.002900	0.688440	0.225570	
<b>LDAPS_CC4</b>	0.005005	-0.141384	-0.044822	0.514552	0.128209	
<b>LDAPS_PPT1</b>	-0.001954	-0.109440	0.114859	0.260943	0.267867	
<b>LDAPS_PPT2</b>	-0.007635	-0.099354	0.070025	0.390090	0.227823	
<b>LDAPS_PPT3</b>	-0.013865	-0.120804	-0.046608	0.239580	0.133673	
<b>LDAPS_PPT4</b>	-0.011085	-0.100955	-0.064180	0.170301	0.119518	
<b>lat</b>	-0.239118	-0.052594	-0.079332	0.086238	0.195849	
<b>lon</b>	-0.118845	0.009215	-0.043030	-0.076101	0.027026	
<b>DEM</b>	-0.256706	-0.187753	-0.251791	0.101876	0.177314	
<b>Slope</b>	-0.091721	-0.105532	-0.146422	0.123620	0.220092	
<b>Solar radiation</b>	-0.021194	-0.020531	0.061699	0.243164	0.147950	
<b>Next_Tmax</b>	0.105516	0.609670	0.463490	-0.442685	-0.286979	
<b>Next_Tmin</b>	0.128184	0.622100	0.797795	0.095574	-0.073069	

24 rows × 24 columns

```
In [22]: srn.pairplot(df[corr_feat], "Next_Tmax")
```

```
Out[22]: <seaborn.axisgrid.PairGrid at 0x2a53d46dc00>
```



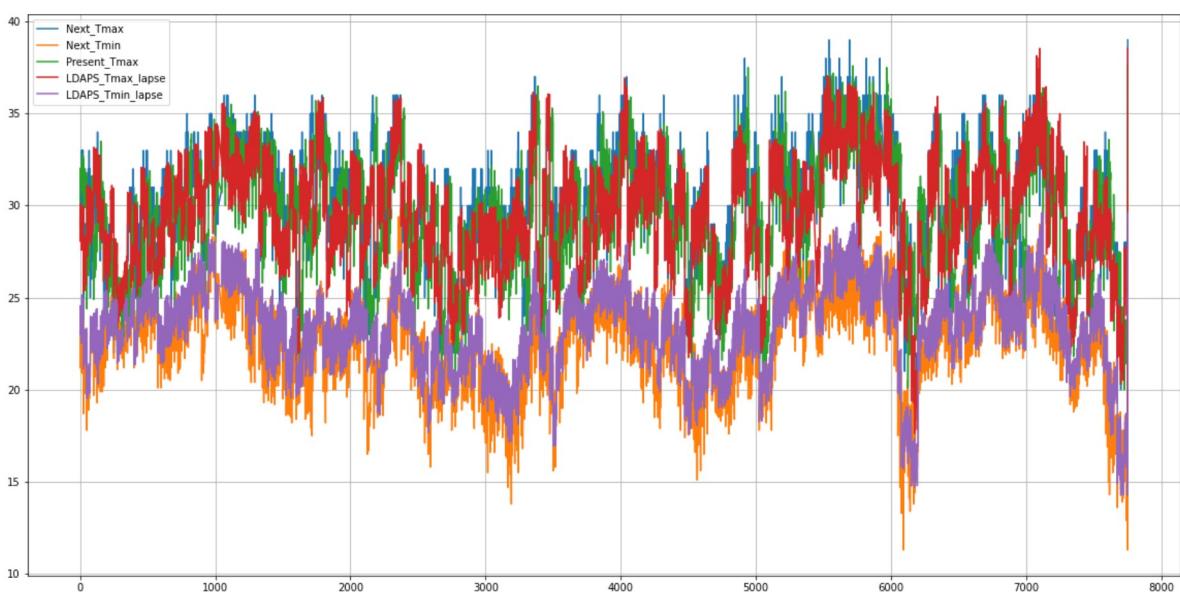
```
In [ ]:
```

```
In [23]:
```

```
def classification_and_fit_model(model, data, predictors, outcome):
    model.fit(data[predictors], data[outcome])
    predictions = model.predict(data[predictors])
    accuracy = metrics.accuracy_score(predictions, data[outcome])
    print('Accuracy : %s' % '{0:.3%}'.format(accuracy))
    kf = KFold(n_splits=5)
    error = []
    for train, test in kf.split(data):
        train_predictors = data[predictors].iloc[train,:]
        train_target = data[outcome].iloc[train]
        model.fit(train_predictors, train_target)
        error.append(model.score(data[predictors].iloc[test,:],
                                data[outcome].iloc[test]))
    print('Cross-Validation Score : %s' % '{0:.3%}'.format(np.mean(error)))
    model.fit(data[predictors],data[outcome])
```

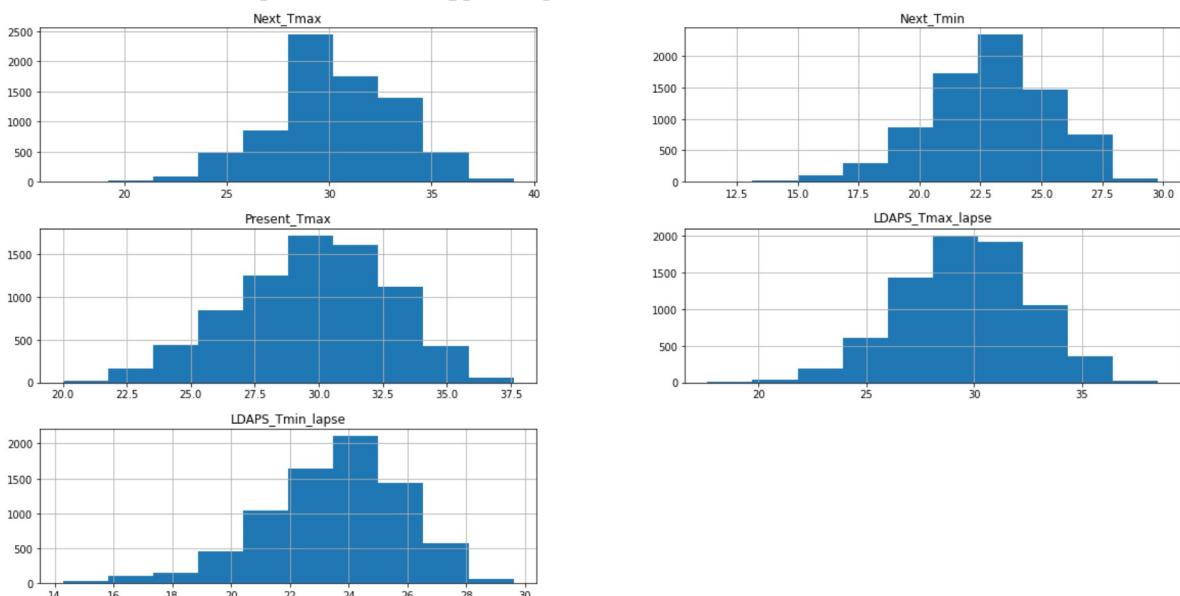
In [140... df[['Next\_Tmax', 'Next\_Tmin', 'Present\_Tmax', 'LDAPS\_Tmax\_lapse', 'LDAPS\_Tm...

Out[140... <AxesSubplot:>



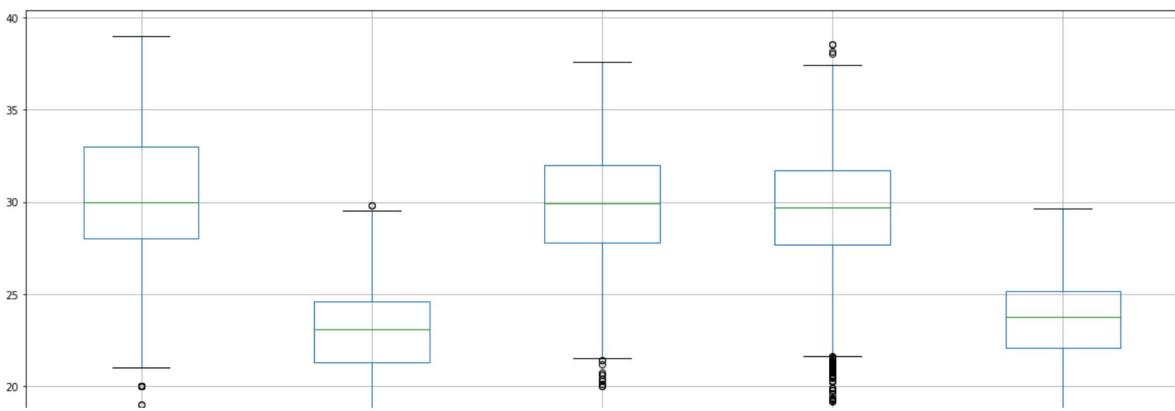
In [141... df[['Next\_Tmax', 'Next\_Tmin', 'Present\_Tmax', 'LDAPS\_Tmax\_lapse', 'LDAPS\_Tm...

Out[141... array([ [  
 <AxesSubplot:title={'center':'Next\_Tmax'}>,  
 <AxesSubplot:title={'center':'Next\_Tmin'}>],  
 [  
 <AxesSubplot:title={'center':'Present\_Tmax'}>,  
 <AxesSubplot:title={'center':'LDAPS\_Tmax\_lapse'}>],  
 [  
 <AxesSubplot:title={'center':'LDAPS\_Tmin\_lapse'}>,  
 <AxesSubplot:>]], dtype=object)



In [139... df[['Next\_Tmax', 'Next\_Tmin', 'Present\_Tmax', 'LDAPS\_Tmax\_lapse', 'LDAPS\_Tm...

Out[139... <AxesSubplot:>



```
In [24]: predictor = ['Present_Tmax', 'LDAPS_Tmax_lapse', 'LDAPS_Tmin_lapse']
```

```
In [25]: outcome = 'Next_Tmax'
```

```
In [92]: arModelos = []
dctModelo = {'Algoritimo': '', 'Modelo': False}
```

```
In [93]: X_train, X_test, y_train, y_test = train_test_split(df[predictor], df[outco
```

## Modelo Linear : Ridge

```
In [94]: from sklearn.linear_model import RidgeClassifier
```

```
In [95]: model = RidgeClassifier()
```

```
In [96]: dctModelo['Algoritimo'] = 'Modelo Linear: Ridge'
dctModelo['Modelo'] = model
arModelos.append(dctModelo.copy())
```

## Gradiente Descendente Estocástico

```
In [97]: model = SGDClassifier()
```

```
In [98]: dctModelo['Algoritimo'] = 'Gradiente Descendente Estocástico'
dctModelo['Modelo'] = model
arModelos.append(dctModelo.copy())
```

## Árvore de Decisão

```
In [99]: model = DecisionTreeClassifier()
```

```
In [100...]  
dctModelo['Algoritimo'] = 'Árvore de Decisão'  
dctModelo['Modelo'] = model  
arModelos.append(dctModelo.copy())
```

```
In [ ]:
```

## Floresta de Árvores Randômicas

```
In [101...]  
model = RandomForestClassifier(n_estimators=100, min_samples_split=25, max_
```

```
In [102...]  
dctModelo['Algoritimo'] = 'Árvores Randômicas'  
dctModelo['Modelo'] = model  
arModelos.append(dctModelo.copy())
```

```
In [ ]:
```

## Classificador Multi Layer Perceptron para Rede Neural

```
In [103...]  
from sklearn.neural_network import MLPClassifier
```

```
In [104...]  
model = MLPClassifier()
```

```
In [105...]  
dctModelo['Algoritimo'] = 'Multi Layer Perceptron'  
dctModelo['Modelo'] = model  
arModelos.append(dctModelo.copy())
```

## Máquinas de Vetores de Suporte

```
In [106...]  
from sklearn.svm import LinearSVC
```

```
In [107...]  
model = LinearSVC()
```

```
In [108...]  
dctModelo['Algoritimo'] = 'Máquinas de Vetores de Suporte'  
dctModelo['Modelo'] = model  
arModelos.append(dctModelo.copy())
```

```
In [ ]:
```

## K Vizinhos mais Próximos

```
In [109...]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [110...]: model = KNeighborsClassifier()
```

```
In [111...]: dctModelo['Algoritimo'] = 'K Vizinhos mais Próximos'  
dctModelo['Modelo'] = model  
arModelos.append(dctModelo.copy())
```

```
In [ ]:
```

## Bayesian

```
In [112...]: from sklearn.naive_bayes import GaussianNB
```

```
In [113...]: model = GaussianNB()
```

```
In [114...]: dctModelo['Algoritimo'] = 'Método Bayesiano'  
dctModelo['Modelo'] = model  
arModelos.append(dctModelo.copy())
```

```
In [115...]: arModelos
```

```
Out[115...]: [ {'Algoritimo': 'Modelo Linear: Ridge', 'Modelo': RidgeClassifier()},  
  {'Algoritimo': 'Gradiente Descendente Estocástico',  
   'Modelo': SGDClassifier()},  
  {'Algoritimo': 'Árvore de Decisão', 'Modelo': DecisionTreeClassifier()},  
  {'Algoritimo': 'Árvores Randômicas',  
   'Modelo': RandomForestClassifier(max_depth=7, max_features=3, min_samples_split=25)},  
  {'Algoritimo': 'Multi Layer Perceptron', 'Modelo': MLPClassifier()},  
  {'Algoritimo': 'Máquinas de Vetores de Suporte', 'Modelo': LinearSVC()},  
  {'Algoritimo': 'K Vizinhos mais Próximos', 'Modelo': KNeighborsClassifier()},  
  {'Algoritimo': 'Método Bayesiano', 'Modelo': GaussianNB()} ]
```

## Benchmark

### Validação Cruzada

```
In [116...]: for i in arModelos:  
    print(i['Algoritimo'])  
    classification_and_fit_model(i['Modelo'], df, predictor, outcome)  
    print('=' * 50)
```

```
Modelo Linear: Ridge  
Accuracy : 22.556%  
Cross-Validation Score : 22.069%
```

Cross-Validation Score : 19.499%  
Cross-Validation Score : 19.960%  
Cross-Validation Score : 18.561%  
Cross-Validation Score : 18.551%

---

Gradiente Descendente Estocástico  
Accuracy : 16.772%  
Cross-Validation Score : 16.403%  
Cross-Validation Score : 13.768%  
Cross-Validation Score : 14.361%  
Cross-Validation Score : 15.267%  
Cross-Validation Score : 13.057%

---

Árvore de Decisão  
Accuracy : 100.000%  
Cross-Validation Score : 19.631%  
Cross-Validation Score : 17.556%  
Cross-Validation Score : 17.479%  
Cross-Validation Score : 18.099%  
Cross-Validation Score : 17.839%

---

Árvores Randômicas  
Accuracy : 37.760%  
Cross-Validation Score : 25.428%  
Cross-Validation Score : 25.296%  
Cross-Validation Score : 25.165%  
Cross-Validation Score : 25.527%  
Cross-Validation Score : 24.901%

---

Multi Layer Perceptron  
Accuracy : 26.206%  
Cross-Validation Score : 27.009%  
Cross-Validation Score : 25.428%  
Cross-Validation Score : 25.187%  
Cross-Validation Score : 23.913%  
Cross-Validation Score : 23.294%

---

Máquinas de Vetores de Suporte  
Accuracy : 18.208%  
Cross-Validation Score : 0.000%  
Cross-Validation Score : 4.545%  
Cross-Validation Score : 8.059%  
Cross-Validation Score : 8.910%  
Cross-Validation Score : 8.880%

---

K Vizinhos mais Próximos  
Accuracy : 49.196%  
Cross-Validation Score : 20.487%  
Cross-Validation Score : 20.059%  
Cross-Validation Score : 19.433%  
Cross-Validation Score : 19.598%  
Cross-Validation Score : 19.078%

---

Método Bayesiano  
Accuracy : 26.719%  
Cross-Validation Score : 22.793%  
Cross-Validation Score : 23.419%  
Cross-Validation Score : 24.133%  
Cross-Validation Score : 24.835%  
Cross-Validation Score : 24.229%

---

## Acurácia do Modelo em dados de teste

In [121...]

```
for i in arModelos:  
    print(i['Algoritimo'])  
    i['Modelo'].fit(X_train, y_train)  
    predictions = i['Modelo'].predict(X_test)  
    accuracy = metrics.accuracy_score(predictions, y_test)  
    print('Acurácia : %s' % '{0:.3%}'.format(accuracy))  
    #print(metrics.classification_report(y_test, i['Modelo'].predict(X_test))  
    print('=' * 50)
```

Modelo Linear: Ridge  
Acurácia : 22.617%

=====

Gradiente Descendente Estocástico  
Acurácia : 14.668%

=====

Árvore de Decisão  
Acurácia : 24.199%

=====

Árvores Randômicas  
Acurácia : 29.600%

=====

Multi Layer Perceptron  
Acurácia : 24.901%

=====

Máquinas de Vetores de Suporte  
Acurácia : 12.604%

=====

K Vizinhos mais Próximos  
Acurácia : 23.671%

=====

Método Bayesiano  
Acurácia : 26.702%

## Métricas de avaliação: Precisão, Revocação e Especificidade

In [122...]

```
for i in arModelos:  
    print(i['Algoritimo'])  
    print(metrics.classification_report(y_test, i['Modelo'].predict(X_test))  
    print('=' * 50)
```

Modelo Linear: Ridge

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	7
22	0.00	0.00	0.00	5
23	0.00	0.00	0.00	18
24	0.00	0.00	0.00	73
25	0.00	0.00	0.00	83
26	0.21	0.25	0.23	126
27	0.00	0.00	0.00	126
28	0.17	0.35	0.23	196
29	0.00	0.00	0.00	248
30	0.24	0.54	0.34	295
31	0.00	0.00	0.00	231

32	0.25	0.54	0.34	283
33	0.22	0.40	0.29	238
34	0.21	0.05	0.08	205
35	0.00	0.00	0.00	93
36	0.00	0.00	0.00	36
37	0.00	0.00	0.00	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.23	2277
macro avg	0.06	0.10	0.07	2277
weighted avg	0.13	0.23	0.16	2277

---

Gradiente Descendente Estocástico

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	7
22	0.00	0.00	0.00	5
23	0.00	0.00	0.00	18
24	0.00	0.00	0.00	73
25	0.00	0.00	0.00	83
26	0.00	0.00	0.00	126
27	0.14	0.41	0.21	126
28	0.00	0.00	0.00	196
29	0.00	0.00	0.00	248
30	0.00	0.00	0.00	295
31	0.00	0.00	0.00	231
32	0.15	1.00	0.26	283
33	0.00	0.00	0.00	238
34	0.00	0.00	0.00	205
35	0.00	0.00	0.00	93
36	0.00	0.00	0.00	36
37	0.00	0.00	0.00	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.15	2277
macro avg	0.01	0.06	0.02	2277
weighted avg	0.03	0.15	0.04	2277

---

Árvore de Decisão

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.33	0.29	0.31	7
22	0.12	0.20	0.15	5
23	0.12	0.11	0.12	18
24	0.43	0.36	0.39	73
25	0.28	0.27	0.27	83
26	0.26	0.27	0.26	126
27	0.18	0.21	0.20	126
28	0.23	0.27	0.25	196
29	0.26	0.25	0.25	248
30	0.23	0.21	0.22	295
31	0.19	0.19	0.19	231
32	0.27	0.26	0.26	283

33	0.27	0.26	0.26	238
34	0.29	0.25	0.27	205
35	0.15	0.14	0.15	93
36	0.21	0.36	0.27	36
37	0.33	0.33	0.33	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.24	2277
macro avg	0.19	0.19	0.19	2277
weighted avg	0.25	0.24	0.24	2277

---

**Árvore Randômicas**

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.36	0.57	0.44	7
22	0.00	0.00	0.00	5
23	0.00	0.00	0.00	18
24	0.36	0.48	0.41	73
25	0.43	0.22	0.29	83
26	0.29	0.33	0.31	126
27	0.31	0.03	0.06	126
28	0.23	0.33	0.27	196
29	0.30	0.26	0.28	248
30	0.28	0.39	0.32	295
31	0.19	0.16	0.17	231
32	0.31	0.42	0.36	283
33	0.36	0.28	0.31	238
34	0.36	0.37	0.36	205
35	0.29	0.16	0.21	93
36	0.35	0.47	0.40	36
37	0.00	0.00	0.00	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.30	2277
macro avg	0.20	0.20	0.19	2277
weighted avg	0.29	0.30	0.28	2277

---

**Multi Layer Perceptron**

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	7
22	0.00	0.00	0.00	5
23	0.00	0.00	0.00	18
24	0.45	0.18	0.25	73
25	0.26	0.14	0.19	83
26	0.18	0.55	0.27	126
27	0.00	0.00	0.00	126
28	0.27	0.15	0.19	196
29	0.26	0.16	0.20	248
30	0.25	0.51	0.34	295
31	0.11	0.03	0.05	231
32	0.28	0.23	0.26	283
33	0.27	0.61	0.38	238

34	0.40	0.10	0.16	205
35	0.28	0.08	0.12	93
36	0.24	0.22	0.23	36
37	0.00	0.00	0.00	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.25	2277
macro avg	0.15	0.13	0.12	2277
weighted avg	0.24	0.25	0.21	2277

---

Máquinas de Vetores de Suporte

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	7
22	0.00	0.00	0.00	5
23	0.00	0.00	0.00	18
24	0.21	0.90	0.34	73
25	0.00	0.00	0.00	83
26	0.00	0.00	0.00	126
27	0.00	0.00	0.00	126
28	0.00	0.00	0.00	196
29	0.00	0.00	0.00	248
30	0.00	0.00	0.00	295
31	0.11	0.96	0.20	231
32	0.00	0.00	0.00	283
33	0.00	0.00	0.00	238
34	0.00	0.00	0.00	205
35	0.00	0.00	0.00	93
36	0.00	0.00	0.00	36
37	0.00	0.00	0.00	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.13	2277
macro avg	0.01	0.08	0.02	2277
weighted avg	0.02	0.13	0.03	2277

---

K Vizinhos mais Próximos

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.25	1.00	0.40	1
21	0.75	0.43	0.55	7
22	0.17	0.20	0.18	5
23	0.27	0.22	0.24	18
24	0.34	0.33	0.34	73
25	0.23	0.33	0.27	83
26	0.18	0.25	0.21	126
27	0.21	0.25	0.23	126
28	0.19	0.21	0.20	196
29	0.18	0.18	0.18	248
30	0.26	0.27	0.27	295
31	0.18	0.17	0.17	231
32	0.26	0.25	0.26	283
33	0.30	0.23	0.26	238
34	0.30	0.25	0.27	205

35	0.27	0.22	0.24	93
36	0.34	0.33	0.34	36
37	0.00	0.00	0.00	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.24	2277
macro avg	0.21	0.23	0.21	2277
weighted avg	0.24	0.24	0.24	2277

=====

Método Bayesiano

	precision	recall	f1-score	support
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	1
20	0.17	1.00	0.29	1
21	0.50	0.43	0.46	7
22	0.00	0.00	0.00	5
23	0.00	0.00	0.00	18
24	0.25	0.38	0.31	73
25	0.24	0.14	0.18	83
26	0.27	0.29	0.28	126
27	0.00	0.00	0.00	126
28	0.22	0.32	0.26	196
29	0.29	0.18	0.22	248
30	0.26	0.36	0.30	295
31	0.22	0.13	0.16	231
32	0.29	0.41	0.34	283
33	0.29	0.27	0.28	238
34	0.38	0.36	0.37	205
35	0.19	0.14	0.16	93
36	0.24	0.44	0.31	36
37	0.27	0.33	0.30	9
38	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
accuracy			0.27	2277
macro avg	0.18	0.24	0.19	2277
weighted avg	0.25	0.27	0.25	2277

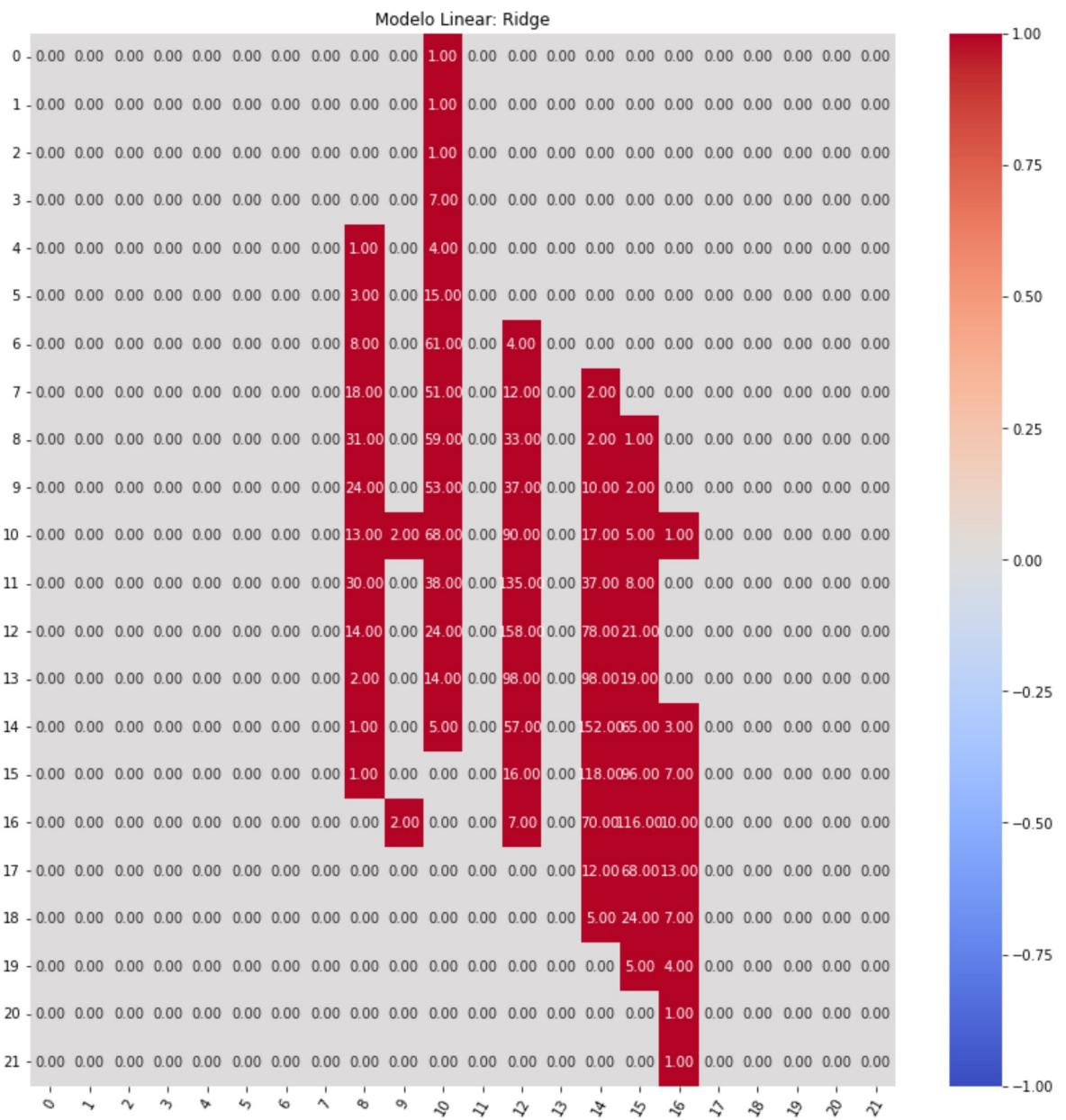
In [ ]:

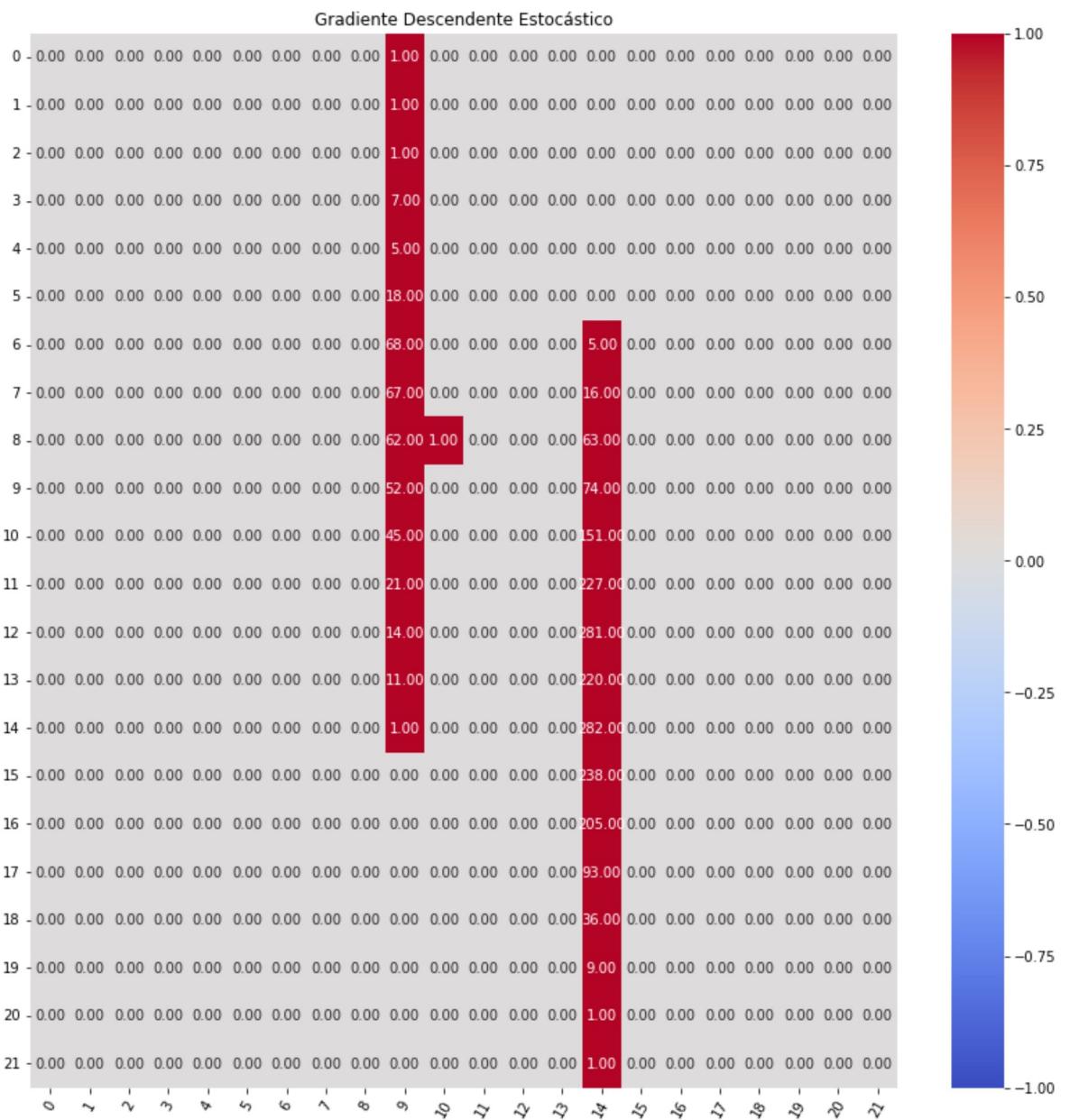
In [ ]:

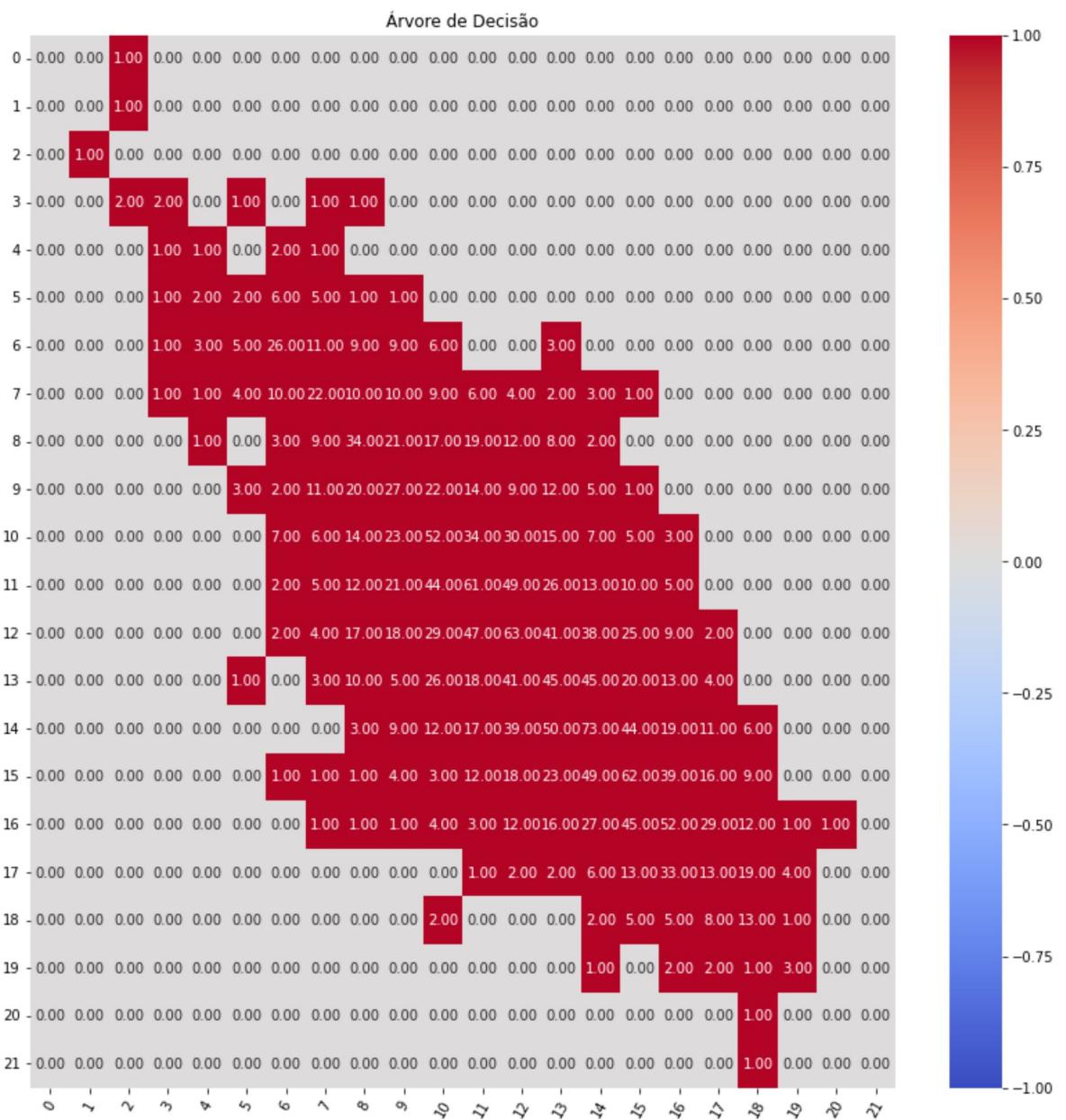
## Matriz de Confusão

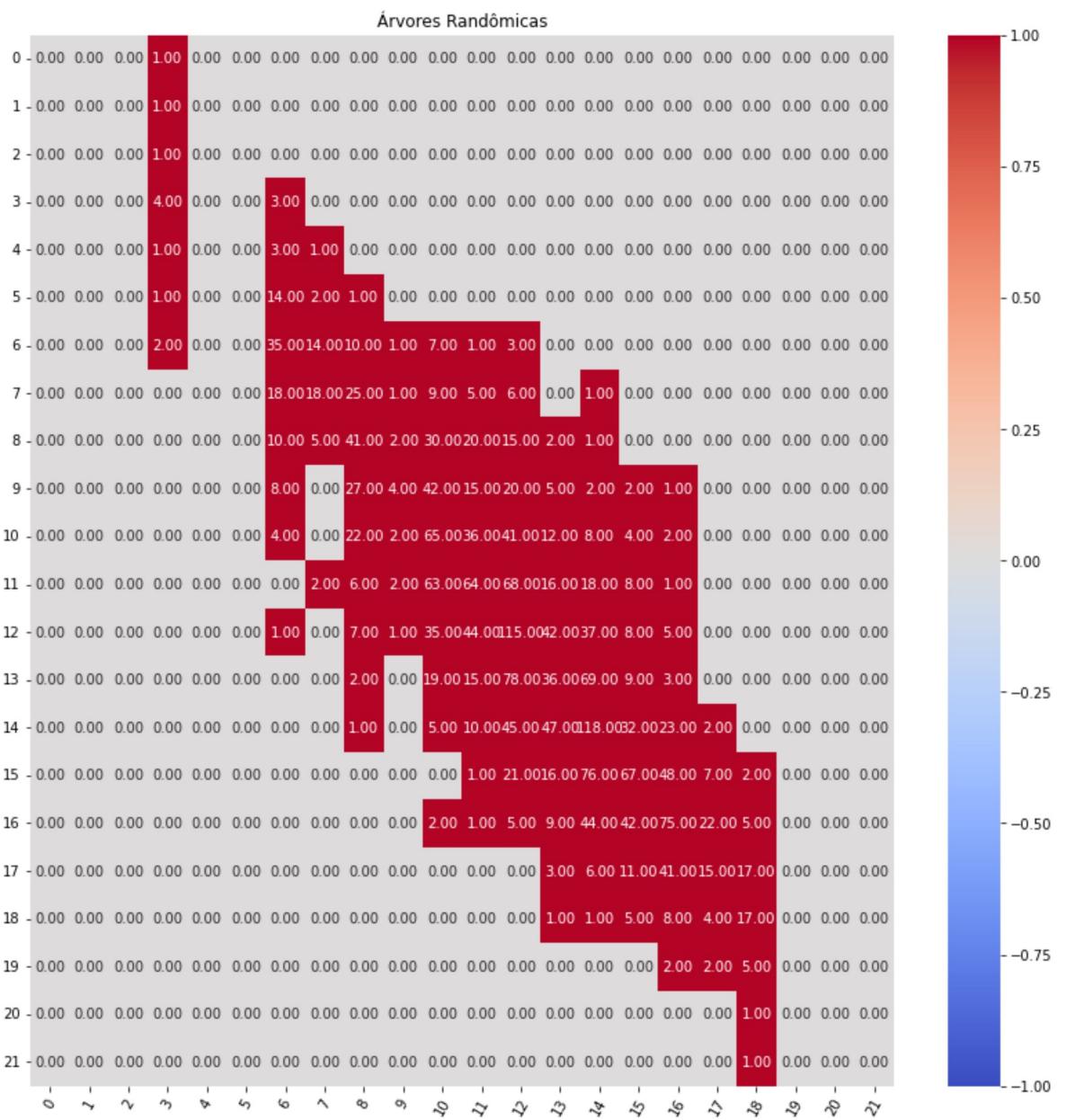
In [126...]

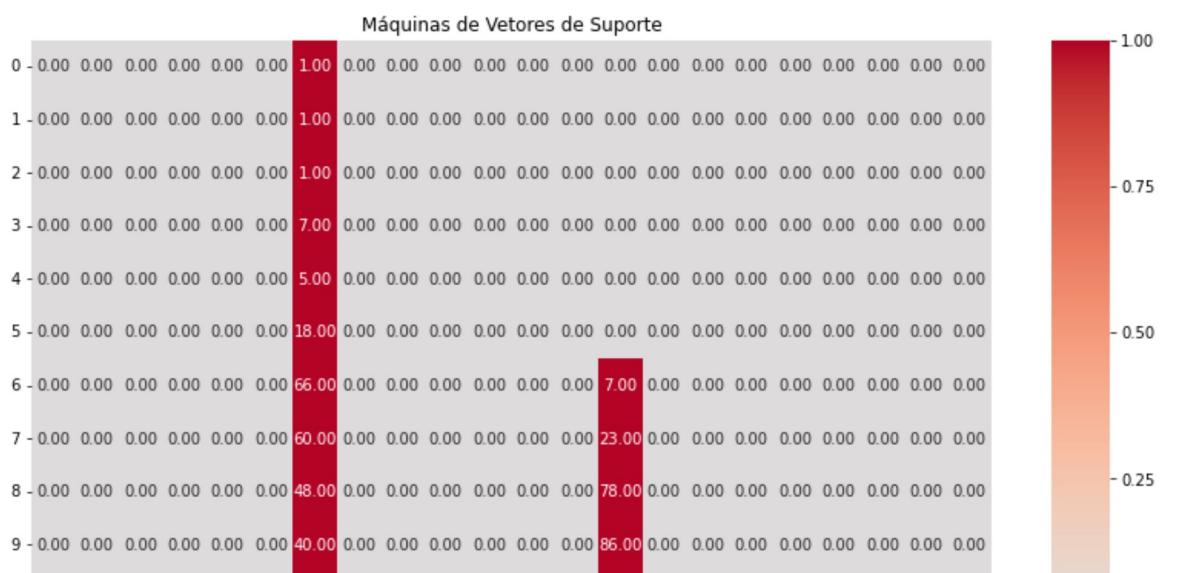
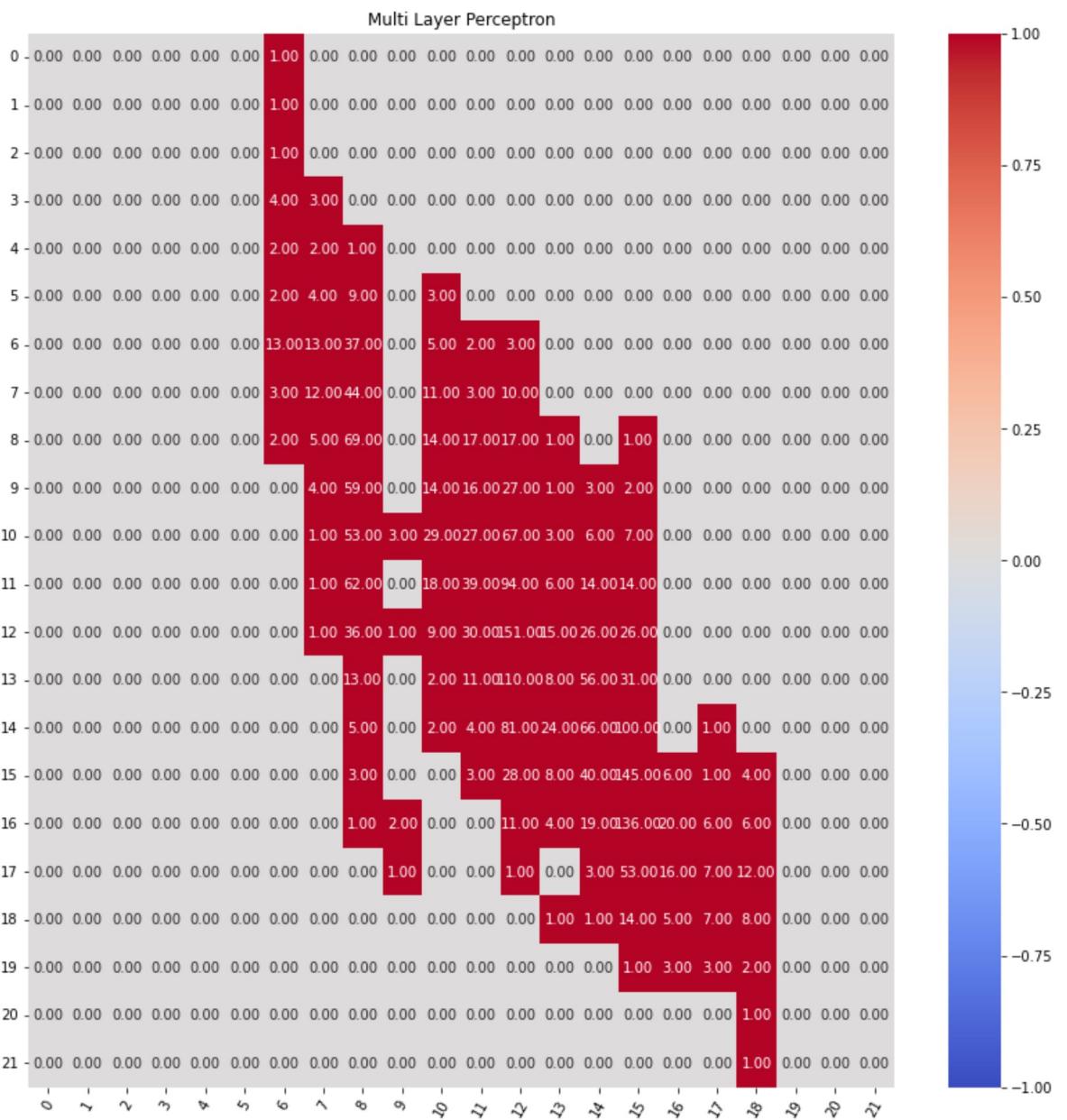
```
for i in arModelos:
    matrix_pred = metrics.confusion_matrix(y_test, i['Modelo'].predict(X_te
f, ax = plt.subplots(figsize = (14,14))
srn.heatmap(matrix_pred, annot=True, fmt=' .2f',
            ax=ax, cmap='coolwarm', vmin=-1,vmax=1)
plt.xticks(rotation=60)
plt.yticks(rotation=0)
plt.title(i['Algoritimo']);
```











In [ ]:

In [ ]:

## Considerações

- A variável Next\_Tmax apresenta uma distribuição simétrica
- As variáveis 'Present\_Tmax', 'LDAPS\_Tmax\_lapse', 'LDAPS\_Tmin\_lapse' apresentam uma correlação com a variável Next\_Tmax maior ou igual a 60%, estando assim aptas para utilização na criação de um modelo preditivo
- Apesar destas features apresentarem alguns outliers, elas seguem uma mesma tendência, o que me leva a crer que os outliers não devem atrapalhar, a princípio, a construção do modelo preditivo
- Entre os algoritmos testados as Árvores Randômicas a princípio apresentaram o melhor resultado, porém não foi feito nenhum ajuste nos dados nem quanto a parametrização do modelo.
- Um próximo passo seria ajustar os hyperparâmetros para o modelo e se necessário uma nova análise nos dados o que poderia gerar por exemplo criação de novas features a partir das existentes com o objetivo de aperfeiçoar o modelo final.
- Neste exemplo foi utilizado como métrica de avaliação a precisão, revocação e especificidade. Existem outras métricas como curva ROC, AUC, Lift que poderão ser utilizadas para análise do modelo.

In [ ]: