

FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA

Graduação Tecnólogo Data Science

GIOVANNA SHIGUEMORI BARBOSA

ISABELA VICTORIA DE NOVAIS ROMANATO

GS Cloud Solutions

PAULISTA

2024

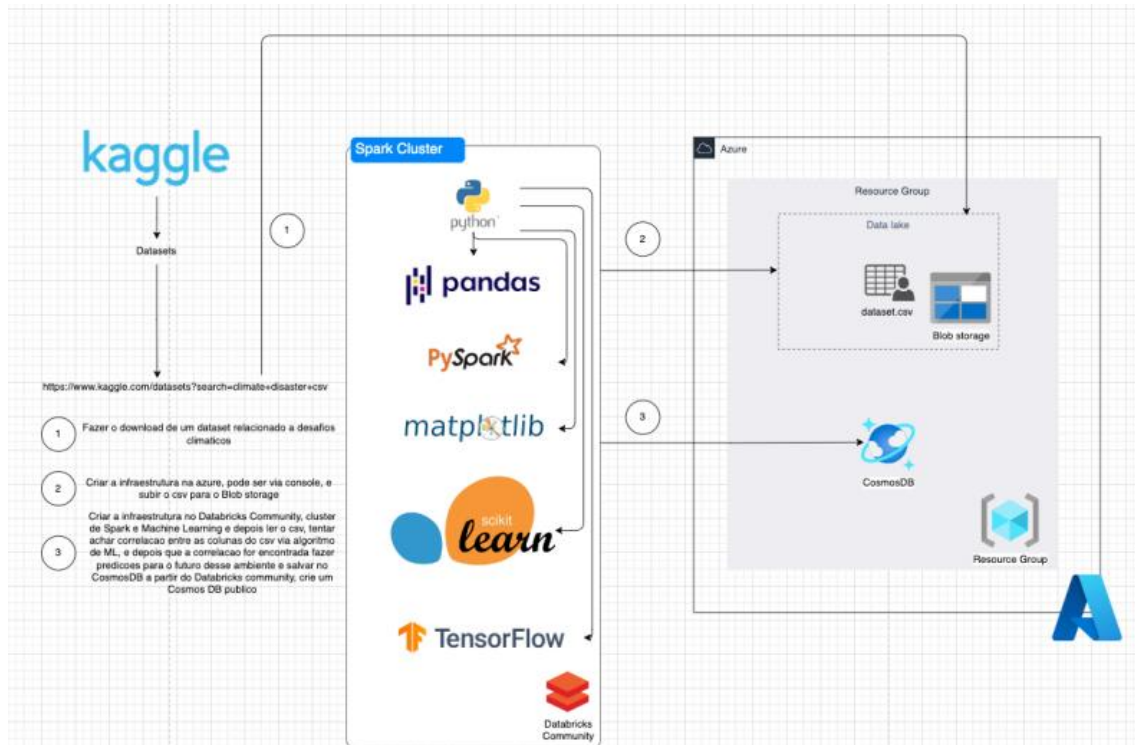
GIOVANNA SHIGUEMORI BARBOSA
ISABELA VICTORIA DE NOVAIS ROMANATO

GS Cloud Solutions

Projeto apresentado a Graduação
Tecnólogo de Data Science, à
Faculdade de Informática e
Administração Paulista, à disciplina
de Cloud Solutions, sob a orientação
do professor Conrad Peres.

PAULISTA
2024

Arquitetura



Contexto

A ONU busca promover um mundo mais sustentável e inclusivo, alinhando-se aos Objetivos de Desenvolvimento Sustentável (ODS) como o ODS 13 (Ação contra a Mudança Climática), ODS 14 (Vida na Água), ODS 15 (Vida Terrestre) e ODS 10 (Redução das Desigualdades). A organização enfatiza a importância de uma sociedade diversa e inclusiva, promovendo a igualdade e o combate à discriminação, especialmente em relação às populações mais vulneráveis, que sofrem desproporcionalmente com os impactos ambientais. Além disso, destaca a preservação da biodiversidade e a redução da pegada de carbono como passos essenciais para mitigar a crise climática, incentivando práticas sustentáveis em todos os setores da sociedade. A ação conjunta entre governos, empresas e cidadãos é fundamental para alcançar esses objetivos, promovendo um futuro mais justo, resiliente e saudável para todos.

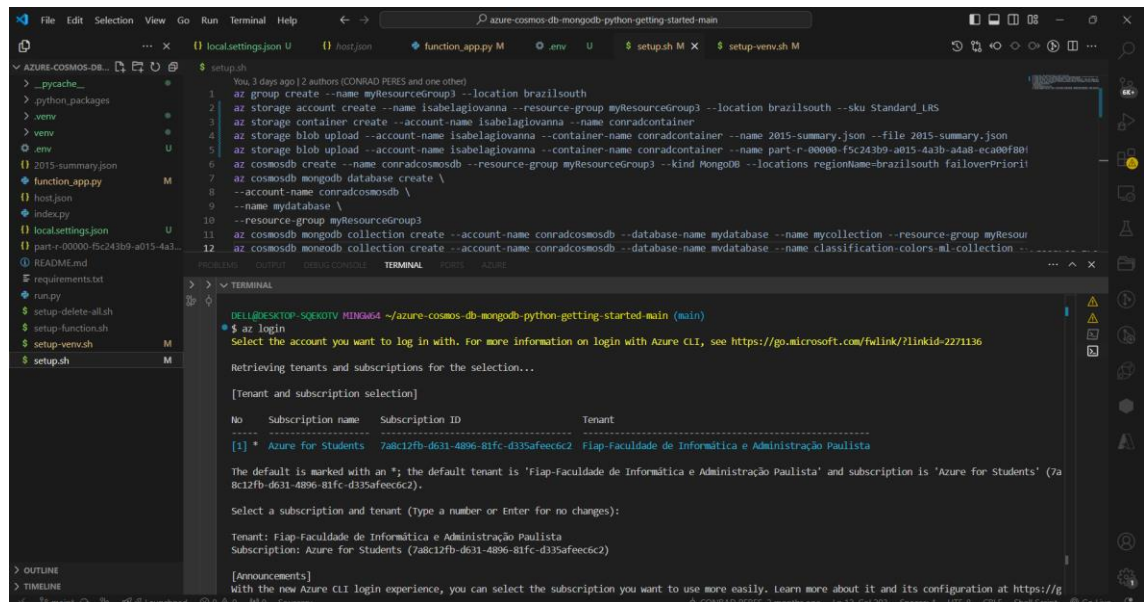
Link do dataset: https://www.kaggle.com/datasets/goyaladi/climate-insights-dataset?resource=download&select=climate_change_data.csv

Link do projeto auxiliar no git: <https://github.com/conradperes/azure-cosmos-db-mongodb-python-getting-started-main.git>

Link do código fonte no git: https://github.com/isabelaromanato/GS_CLOUD.git

NO VSCODE:

az login



```
DELL@DESKTOP-SQKDTV MINGW64 ~/azure-cosmos-db-mongodb-python-getting-started-main (main)
$ az login
Select the account you want to log in with. For more information on login with Azure CLI, see https://go.microsoft.com/fwlink/?linkid=2271136

Retrieving tenants and subscriptions for the selection...

[Tenant and subscription selection]

No  Subscription name  Subscription ID  Tenant
-----
[1] * Azure for Students  7a8c12fb-d631-4896-81fc-d335afeec6c2  Flap-Faculdade de Informática e Administração Paulista

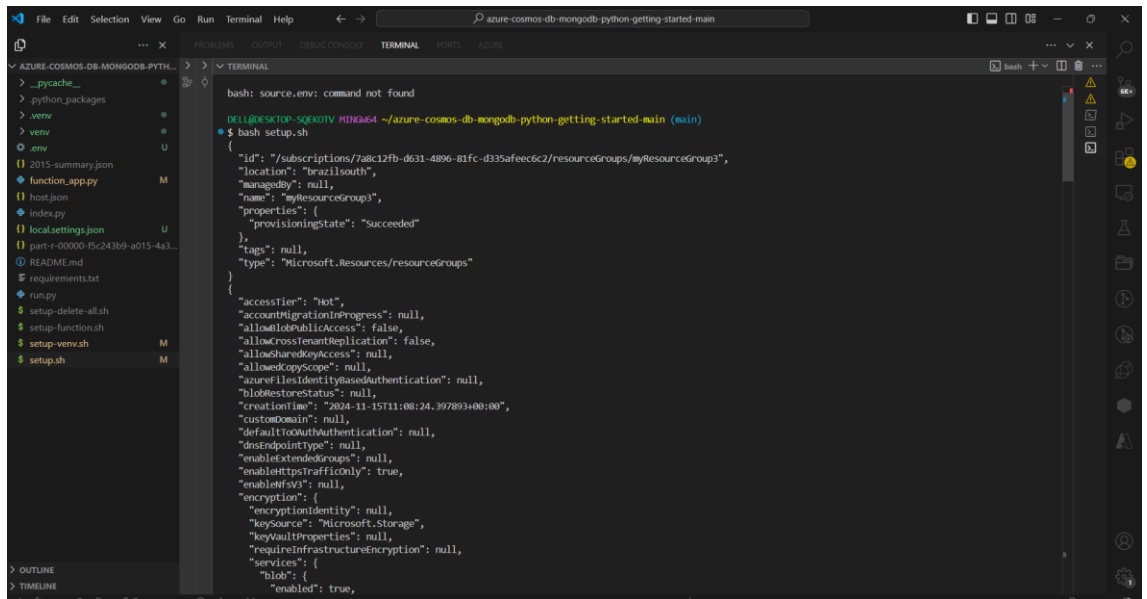
The default is marked with an *; the default tenant is 'Flap-Faculdade de Informática e Administração Paulista' and subscription is 'Azure for Students' (7a8c12fb-d631-4896-81fc-d335afeec6c2).

Select a subscription and tenant (Type a number or Enter for no changes):

Tenant: Flap-Faculdade de Informática e Administração Paulista
Subscription: Azure for Students (7a8c12fb-d631-4896-81fc-d335afeec6c2)

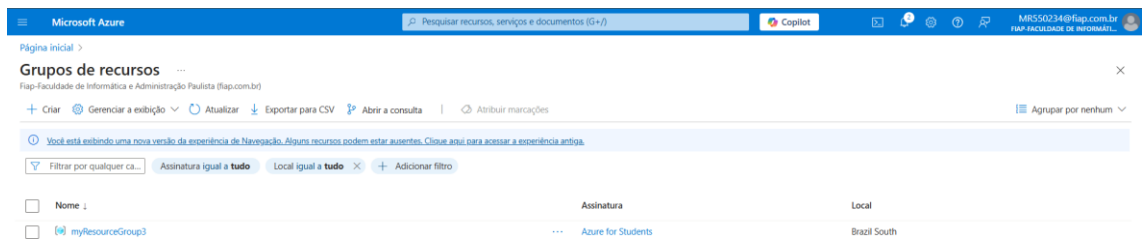
[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://g
```

bash setup.sh

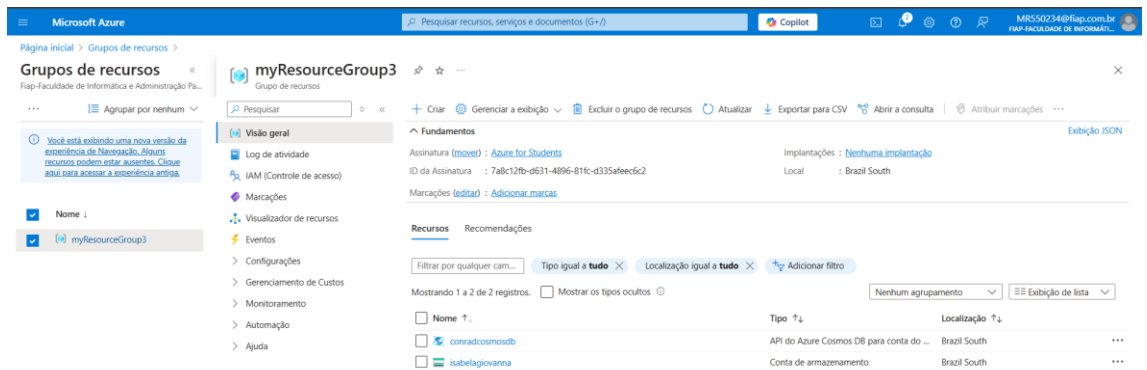


```
bash: source.env: command not found
DELL@DESKTOP-SXKDTV MINGW64 ~/azure-cosmos-db-mongodb-python-getting-started-main (main)
$ bash setup.sh
{"id": "/subscriptions/7a8c12fb-d631-4896-81fc-d335afeec6c2/resourceGroups/myResourceGroup3",
 "location": "brazilsouth",
 "managedBy": null,
 "name": "myResourceGroup3",
 "properties": {
  "provisioningState": "Succeeded"
 },
 "tags": null,
 "type": "Microsoft.Resources/resourceGroups"
 }
{"accessTier": "Hot",
 "accountMigrationInProgress": null,
 "allowBlobPublicAccess": false,
 "allowCrossTenantReplication": false,
 "allowSharedKeyAccess": null,
 "allowedCopyScope": null,
 "azureFileIdentityAuthentication": null,
 "blobRestoreStatus": null,
 "creationTime": "2024-11-15T11:08:24.397893+00:00",
 "customDomain": null,
 "defaultToOAuthAuthentication": null,
 "dnsEndpointType": null,
 "enableExtendedGroups": null,
 "enableHttpsTrafficOnly": true,
 "enableHfsv3": null,
 "encryption": {
  "encryptionIdentity": null,
  "keySource": "Microsoft.Storage",
  "keyVaultProperties": null,
  "requireInfrastructureEncryption": null,
  "services": {
   "blob": {
    "enabled": true,
```

Evidências dos recursos criados na Azure:

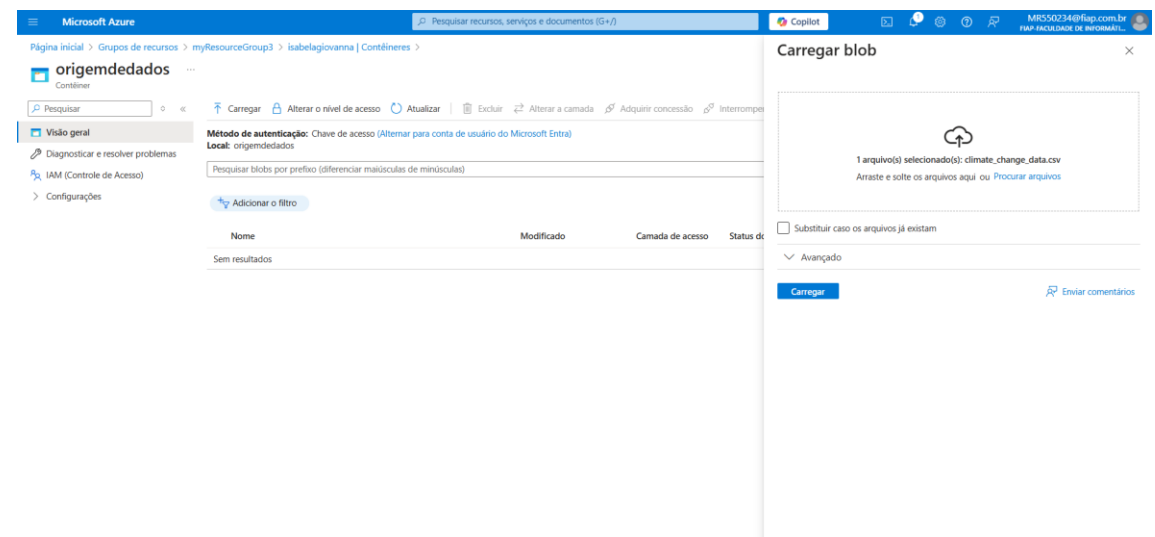
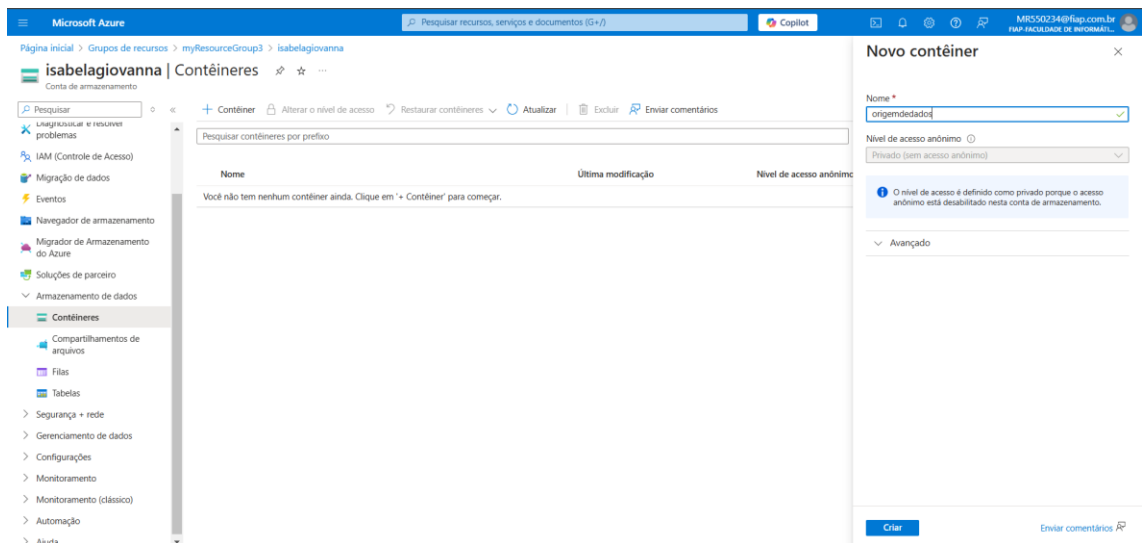


Nome	Assinatura	Local
myResourceGroup3	Azure for Students	Brazil South



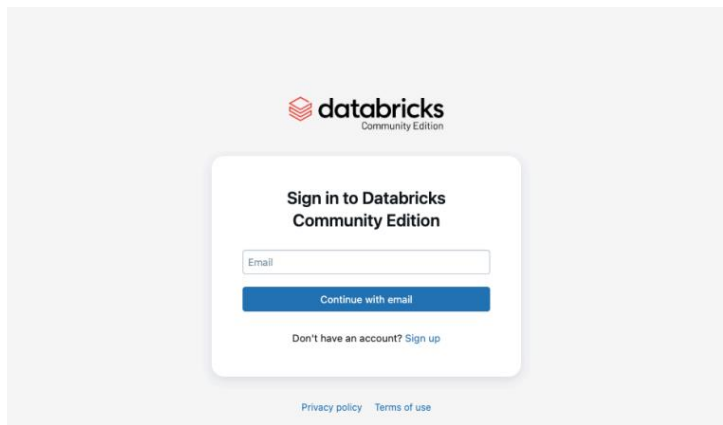
Nome	Tipo	Localização
conradcosmosdb	API do Azure Cosmos DB para conta do ...	Brazil South
isabelagiovanna	Conta de armazenamento	Brazil South

Carregar dataset (climate_change_data.csv) no Blob:

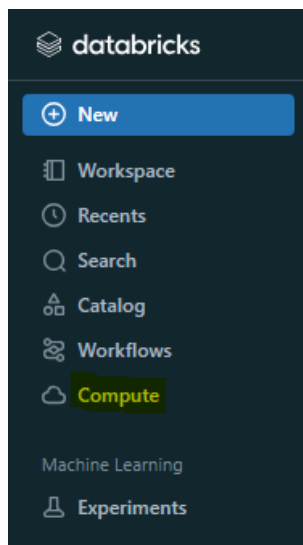


NO DATABRICKS:

Fazer login:



Criar um Cluster:



Compute

[All-purpose compute](#) [Job compute](#)

[Compute](#) > [New compute](#)

Isabela Victoria's Cluster [✎](#)

Compute name

Isabela Victoria's Cluster

Databricks runtime version [ⓘ](#)

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2) [▼](#)

Instance

Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For more configuration options [🔗](#), please upgrade your Databricks subscription. [🔗](#)

Spark

Spark config [ⓘ](#)

spark.databricks.rocksDB.fileManager.useCommitService false

Environment variables [ⓘ](#)

PYSPARK_PYTHON=/databricks/python3/bin/python3

NOTEBOOK PYTHON:

1. Validação dos Dados

Importação das bibliotecas

```
!pip install azure-storage-blob pandas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pyspark.pandas as ps
```

Carregar dataset

```
from azure.storage.blob import BlobServiceClient
import pandas as pd
import io # Import necessário para StringIO

# Configurações da Azure
connection_string =
"DefaultEndpointsProtocol=https;AccountName=isabelagiovanna;AccountKey=
dYDCV4VHTyp/J+xVV7A5zUMAQGBmj2HdyRJv+yvU7tnRM3oYGRtR7A9paV
MW4sd+Jve42U9/+nJR+AStx+doDg==;EndpointSuffix=core.windows.net"
container_name = "origemdedados"
blob_name = "climate_change_data.csv"

# Conecte-se ao Blob Storage
blob_service_client =
BlobServiceClient.from_connection_string(connection_string)
blob_client = blob_service_client.get_blob_client(container=container_name,
blob=blob_name)

# Faça o download do arquivo como texto
```



```
download_stream = blob_client.download_blob()
csv_content = download_stream.readall().decode('utf-8')

# Use io.StringIO para interpretar o conteúdo como arquivo
csv_file_like = io.StringIO(csv_content)

# Carregue o conteúdo no pandas
df = pd.read_csv(csv_file_like)

# Exiba os dados
display(df)
```

	1.0 Date	1.0 Location	1.0 Country	1.2 Temperature	1.2 CO2 Emissions	1.2 Sea Level Rise	1.2 Precipitation	1.2 Humidity
38	2000-02-01 01:59:40.5580558...	West Jeffrey	British Indian Ocean Territory (Chagos Archipelago)	14.89574312670864	474.31808727651696	0.0305414513135537	46.626748375860096	2.64694540185096
39	2000-02-01 22:09:23.8163816...	Duncanmouth	Czech Republic	13.526820846194028	446.960414146781	0.4373569246214632	42.72973912404518	5.0764559952232
40	2000-02-02 18:19:07.0747074...	Fieldside	Iran	20.74140008331123	446.4020457754351	-0.038236117209954	88.33768572387685	44.0701292214990
41	2000-02-03 14:28:50.3330333...	Emmalfort	Antigua and Barbuda	12.290429559011264	321.1354833563896	-0.4450386060723362	74.77749762924807	58.7285989221471...
42	2000-02-04 10:38:33.5913591...	East Davidshire	Seychelles	11.740209187446926	459.71396661109424	1.1792144981950052	88.0695193598249	87.4113606117273
43	2000-02-05 06:48:16.8496849...	Port Austinchester	Slovenia	10.325269443056824	354.5349525451341	0.1239299679099418	31.03894102395417	36.5061531002555
44	2000-02-06 02:58:00.1080108...	North Bryan	Gabon	7.511291128973832	379.73064806516	1.986904928068636	34.075333772491575	82.3407602698015
45	2000-02-06 23:07:43.3663366...	Stewartbury	India	13.863783558947942	408.1218629951279	-0.0788110630241746	77.49866102409709	31.3560053387295
46	2000-02-07 19:17:26.6246624...	North Ryan	Chad	9.026524681217706	391.4395820748849	0.13598059595662	25.378105879498012	17.5044651035193
47	2000-02-08 15:27:09.8829882...	West Michieleland	Congo	19.859165209960093	316.2131332933769	0.8367186163118052	90.84696139800592	7.65442006589974
48	2000-02-09 11:36:53.1413141...	Lisaview	Tunisia	21.112886117354748	499.08728284844136	0.525015923020709	58.8974557477375	28.7037482590844
49	2000-02-10 07:46:36.3996399...	Reynoldsberg	Reunion	9.086153370690852	382.46852005722207	-0.6796880555631175	36.86085100647352	25.2564765414645
50	2000-02-11 03:56:19.6579657...	Wandamouth	Norfolk Island	17.66110748418823	480.2908824975052	-0.0161698169580054	42.8320353924213	96.2121356504187
51	2000-02-12 00:06:02.9162916...	Port Leah	Costa Rica	17.747515508184108	381.383639396125	1.244657569595779	44.805935481306726	61.9178083108824

Resumo com informações das colunas

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Date        10000 non-null  object
1   Location    10000 non-null  object
2   Country     10000 non-null  object
3   Temperature 10000 non-null  float64
4   CO2 Emissions 10000 non-null  float64
5   Sea Level Rise 10000 non-null  float64
6   Precipitation 10000 non-null  float64
7   Humidity    10000 non-null  float64
8   Wind Speed  10000 non-null  float64
dtypes: float64(6), object(3)
memory usage: 703.2+ KB
```

Verifica presença de nulos

```
df.isnull().sum()
```

```
Out[122]: Date
Location      0
Country       0
Temperature    0
CO2 Emissions  0
Sea Level Rise 0
Precipitation  0
Humidity       0
Wind Speed    0
dtype: int64
```

Verifica se há linhas duplicadas

```
df.duplicated().values.any()
```

```
Out[123]: False
```

Resumo conciso das estatísticas descritivas

```
df.describe()
```

	Temperature	CO2 Emissions	Sea Level Rise	Precipitation	Humidity	Wind Speed
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	14.936034	400.220469	-0.003152	49.881208	49.771302	25.082066
std	5.030616	49.696933	0.991349	28.862417	28.929320	14.466648
min	-3.803589	182.131220	-4.092155	0.010143	0.018998	0.001732
25%	11.577991	367.109330	-0.673809	24.497516	24.713250	12.539733
50%	14.981136	400.821324	0.002332	49.818967	49.678412	24.910787
75%	18.305826	433.307905	0.675723	74.524991	75.206390	37.670260
max	33.976956	582.899701	4.116559	99.991900	99.959665	49.997664

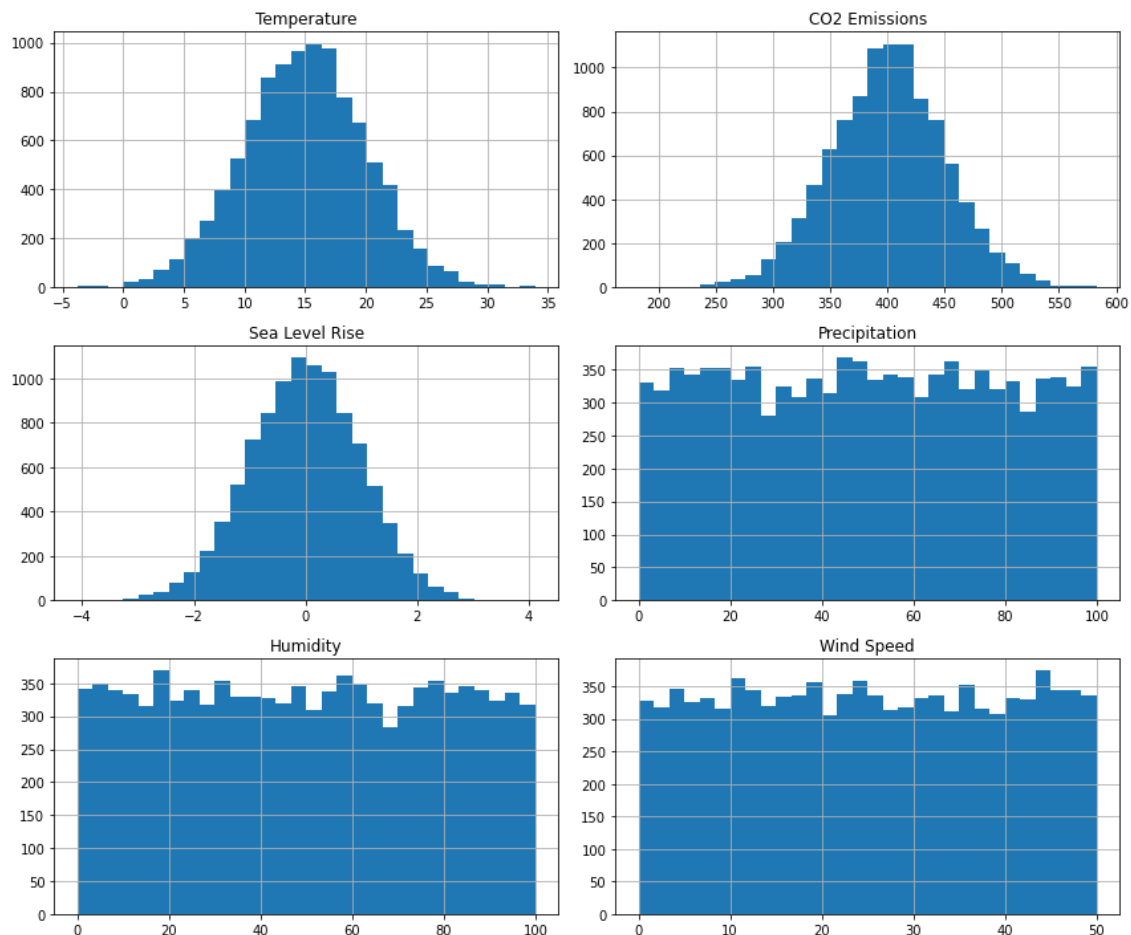
2. Análise Exploratória de Dados

Gerar cópia do dataset original e criar um "limpo"

```
clean_df = df.copy()
```

```
clean_df[['Temperature', 'CO2 Emissions','Sea Level Rise', 'Precipitation',  
'Humidity', 'Wind Speed']].hist(figsize=(12, 10), bins=30)
```

```
plt.tight_layout()
```



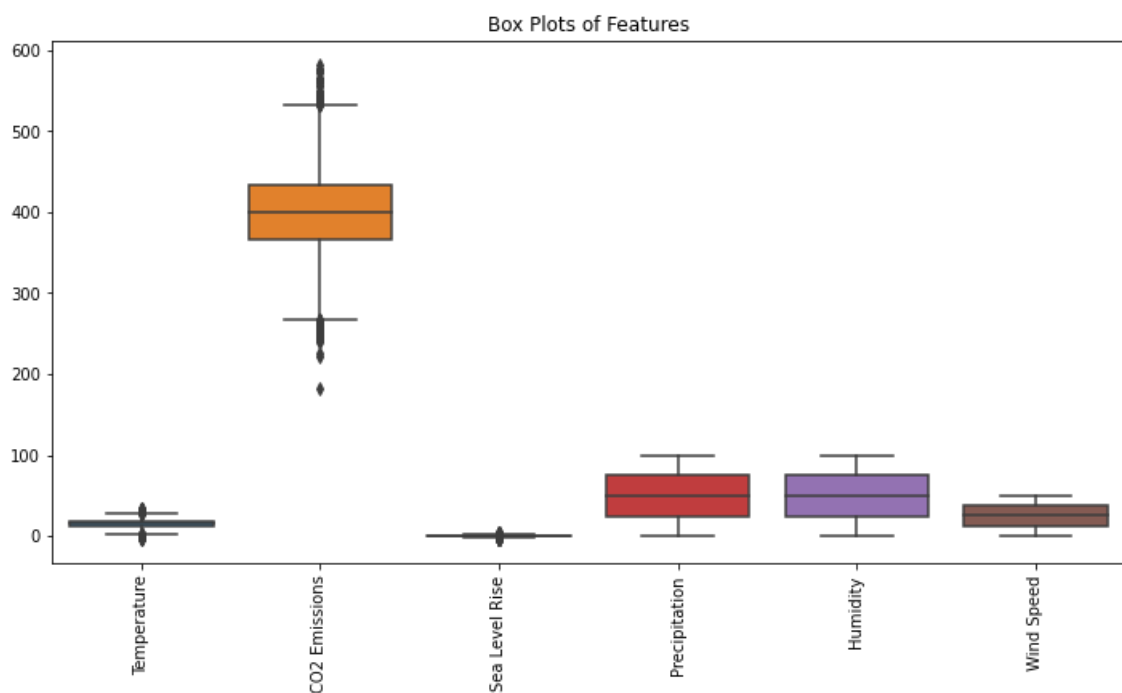
Observações em relação aos histogramas:

1. Temperatura: O histograma de temperatura mostra uma distribuição aproximadamente normal, sugerindo que as leituras de temperatura são estáveis e previsíveis.
2. Emissões de CO2: O histograma de emissões de CO2 também é aproximadamente normal, com valores concentrados em torno de 400 ppm, sendo estáveis ao longo do tempo.
3. Aumento do nível do mar: O histograma de aumento do nível do mar também é aproximadamente normal.

4. Precipitação: O histograma de precipitação mostra uma distribuição mais uniforme, indicando variabilidade nos padrões.
5. Umidade: O histograma de umidade também mostra uma distribuição relativamente uniforme na faixa de 0% a 100%.
6. Velocidade do vento: O histograma da velocidade do vento apresenta uma distribuição uniforme, com valores variando de 0 a 50 km/h.

Analizando colunas com valores discrepantes

```
plt.figure(figsize=(12, 6))  
sns.boxplot(data=clean_df)  
plt.xticks(rotation=90)  
plt.title('Box Plots of Features')
```



Verificando presença de outliers

```
features = ['Temperature', 'CO2 Emissions', 'Sea Level Rise']
```

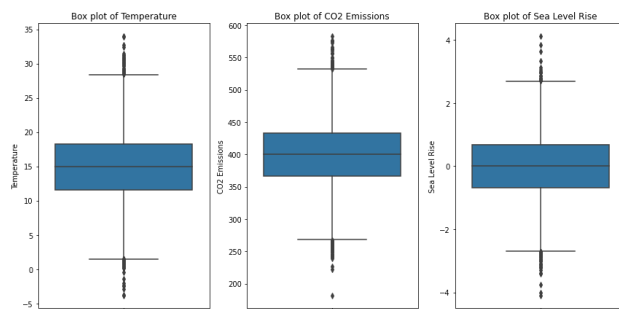
```
fig, axs = plt.subplots(1,3,figsize = (12,6))
```

```
for ax,feature in zip(axes, features):
```

```
    sns.boxplot(y = feature,data=clean_df, ax = ax)
```

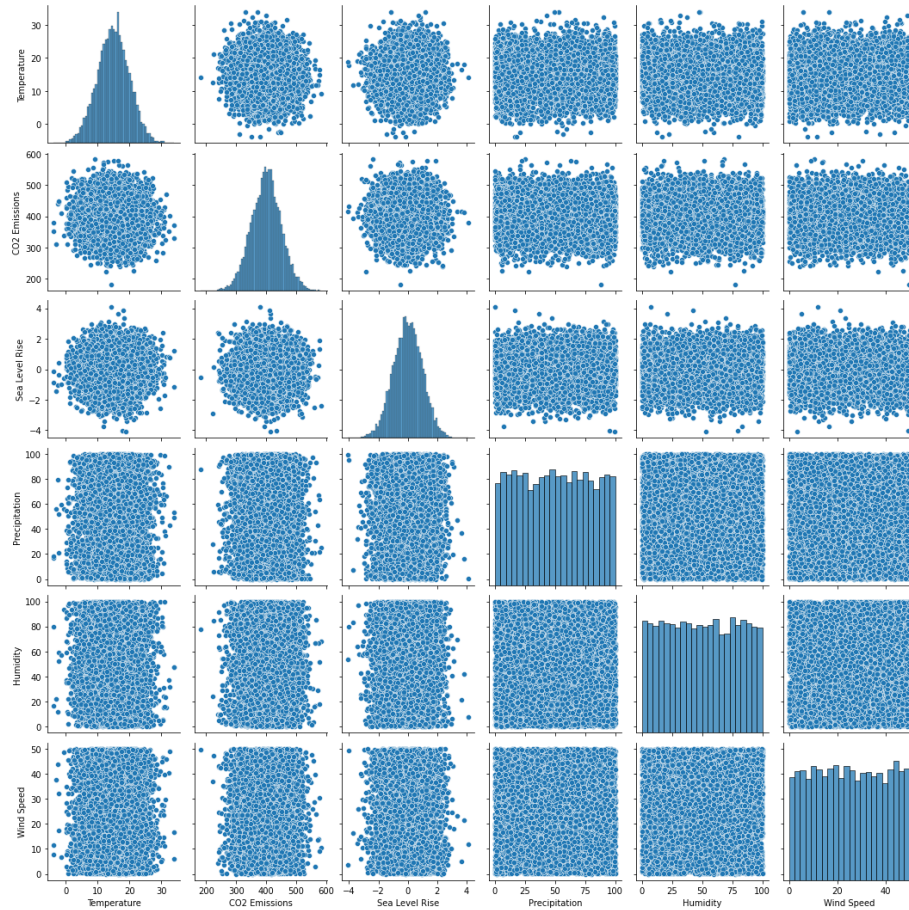
```
    ax.set_title(f'Box plot of {feature}')
```

```
plt.tight_layout()
```



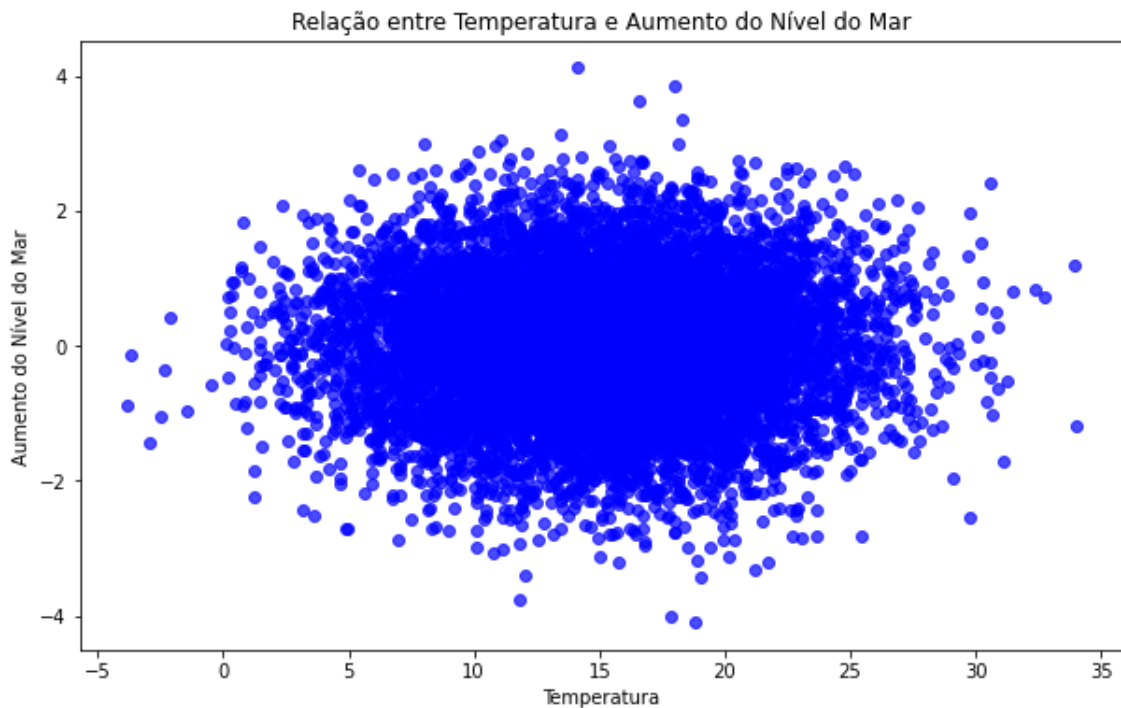
Mapa de Calor

```
sns.pairplot(data = clean_df)
```



```
# Ajustando o gráfico
plt.figure(figsize=(10, 6))
plt.scatter(df_relacionamento_pandas['Temperature'],
df_relacionamento_pandas['Sea Level Rise'], color='blue', alpha=0.7)

# Adicionando título e rótulos aos eixos
plt.title("Relação entre Temperatura e Aumento do Nível do Mar")
plt.xlabel("Temperatura")
plt.ylabel("Aumento do Nível do Mar")
plt.show()
```



Matriz de Correlação

```
num_features=clean_df[['Sea Level Rise', 'Temperature', 'CO2 Emissions',
'Precipitation', 'Humidity', 'Wind Speed']]

corr_matrix = num_features.corr()
```

```
plt.figure(figsize=(12,6))
sns.heatmap(corr_matrix, annot = True)
```



A correlação entre as variáveis é baixa. Essa evidência sugere uma relação linear quase inexistente entre a Sea Level Rise e as outras características.

```
# plt.figure(figsize=(12, 6))
# plt.plot(clean_df['Date'], clean_df['Temperature'])
# plt.title('Temperature Over Time')
# plt.xlabel('Date')
# plt.ylabel('Temperature (°C)')
# Não há tendência ao longo dos anos.
```

3. Desenvolvimento dos Modelos

Regressão Linear

```
clean_df.drop(['Date', 'Location', 'Country', 'Date'], axis = 1, inplace = True)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error, r2_score

X = clean_df.drop(['Sea Level Rise'], axis = 1)
y = clean_df['Sea Level Rise']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Erro quadrático médio:', mse)
print('Coeficiente de determinação:', r2)

```

```

Erro quadrático médio: 0.9973860307922685
Coeficiente de determinação: -0.0029546178809112256

```

Random Forest Regressor

```

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=101)

rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
predictions = rf_model.predict(X_test)

```



```

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('Score:', rf_model.score(X_test, y_test))

```

```

MAE: 0.8124131354713865
MSE: 1.0283395817132381
RMSE: 1.0140707971898402
Score: -0.037693797047672506

```

Interpretando os Resultados

Com base nos valores apresentados:

MAE = 0.8101: Em média, o modelo erra por cerca de 0.81 unidades na previsão.

MSE = 1.0259: O erro quadrático médio é de aproximadamente 1.02.

RMSE = 1.0129: A raiz quadrada do erro quadrático médio é de aproximadamente 1.01, o que significa que, em média, as previsões estão desviando cerca de 1.01 unidades dos valores reais.

Score = -0.0353: O valor negativo do score indica que o modelo não está conseguindo explicar a variabilidade dos dados.

PCA

```

import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.decomposition import FactorAnalysis
import seaborn as sns

components = 2
cols = ['Temperature', 'CO2 Emissions', 'Sea Level Rise',
        'Precipitation', 'Humidity', 'Wind Speed']

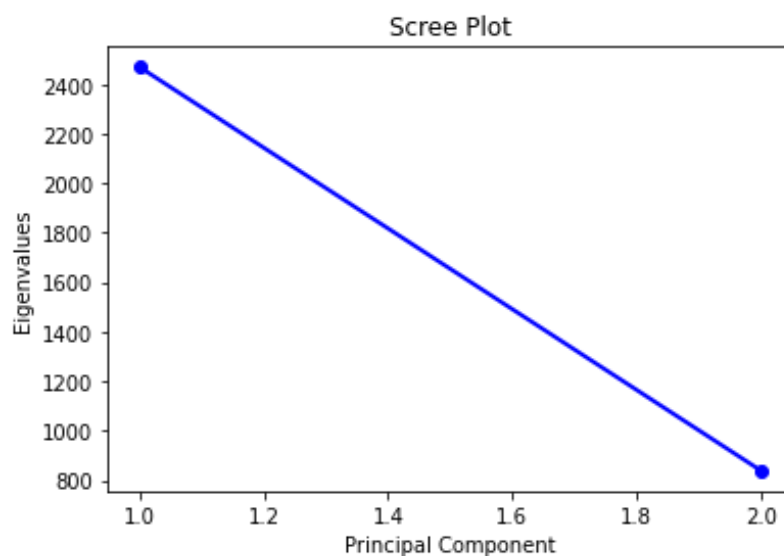
```

```
scaled_df = clean_df[cols]
pca = PCA(n_components=components)
pca_components = pca.fit_transform(scaled_df)

columns = ['pca_comp_%i' % i for i in range(components)]
df_pca = pd.DataFrame(pca_components, columns=columns,
index=clean_df.index)
df_pca.head()
```

	pca_comp_0	pca_comp_1
0	-3.227739	-39.930871
1	3.491375	-9.179324
2	-51.211111	38.561890
3	-22.504535	-22.830991
4	-10.047279	23.896078

```
plt.plot(np.arange(pca.n_components_)+1, pca.explained_variance_, 'o-
',linewidth=2,color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalues')
plt.show()
```



```
# instanciar o pca
components = 2
pca = PCA(n_components=components)
pca_components = pca.fit_transform(scaled_df)

columns = ['pca_comp_%i' % i for i in range(components)]
componentes_cols = ['PC'+str(i) for i in range(1,components+1)]
df_pca = pd.DataFrame(pca_components, columns=componentes_cols,
index=clean_df.index)
df_pca.head()
```

	PC1	PC2
0	-3.227739	-39.930871
1	3.491375	-9.179324
2	-51.211111	38.561890
3	-22.504535	-22.830991
4	-10.047279	23.896076

```
df2 = pd.concat([df_pca, clean_df['Sea Level Rise']], axis=1)
df2.head()
```

	PC1	PC2	Sea Level Rise
0	-3.227739	-39.930871	0.717506
1	3.491375	-9.179324	1.205715
2	-51.211111	38.561890	-0.160783
3	-22.504535	-22.830991	-0.475931
4	-10.047279	23.896076	1.135757

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import scipy.stats as stats
from sklearn.linear_model import LinearRegression
```

```

from sklearn.preprocessing import PolynomialFeatures
# validacao cruzada
from sklearn.model_selection import KFold, StratifiedKFold
# regressao penalizada
from sklearn.linear_model import Ridge, Lasso, RidgeCV, LassoCV
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

features = ['PC1','PC2']
target = ['Sea Level Rise']

x = df2[features]
y = df2[target]
x_columns = x.columns.to_list()
scaler = StandardScaler()
x = scaler.fit_transform(x)

x_train,          x_test,          y_train,          y_test          =
train_test_split(x,y,test_size=0.2,random_state=42)

model = LinearRegression()
model.fit(x_train, y_train)
print(model.score(x_test, y_test))
lr_coef = pd.DataFrame(np.append(model.intercept_, model.coef_), ['intercepto']
+ x_columns, columns=['Lin Reg'])
lr_coef

```

-0.0007341678053305323

Lin Reg	
intercepto	0.002333
PC1	0.000514
PC2	-0.007312

```
#define cross-validation method to evaluate model
cv = KFold(n_splits=10, shuffle=True, random_state=42)
model = LassoCV(alphas=np.arange(0.1, 1, 0.1), cv=cv)
model.fit(x_train,y_train.values.ravel())
print(model.alpha_)
print(model.score(x_test, y_test))
coef_lasso = pd.DataFrame(np.append(model.intercept_, model.coef_),
['intercepto'] + x_columns, columns=['Lasso'])
coef_lasso
```

0.9
-0.0007719297600194963

Lasso	
intercepto	0.00229
PC1	0.00000
PC2	-0.00000

```
# Define a forma de validação cruzada que será usada na regressão Ridge
cv = KFold(n_splits=10, shuffle=True, random_state=42)
model = RidgeCV(alphas=np.arange(0.1, 1, 0.1), cv=cv,
scoring='neg_mean_squared_error')
model.fit(x_train, y_train)
# Indica o melhor alpha para a Ridge
print(model.alpha_)
print(model.score(x_test, y_test))
coef_ridge = pd.DataFrame(np.append(model.intercept_, model.coef_),
['intercepto'] + x_columns, columns=['Ridge'])
# Indica os coeficientes calculados para a Ridge
coef_ridge
```

0.9
-0.0007341653492956013

```
coefs = pd.concat([lr_coef, coef_lasso, coef_ridge], axis=1)
coefs
```

	Lin Reg	Lasso	Ridge
intercepto	0.002333	0.00229	0.002333
PC1	0.000514	0.00000	0.000514
PC2	-0.007312	-0.00000	-0.007311

Observando o modelo, o desempenho foi insatisfatório...

MLP

	Temperature	CO2 Emissions	Sea Level Rise	Precipitation	Humidity	Wind Speed
0	10.688986	403.118903	0.717506	13.835237	23.631256	18.492026
1	13.814430	396.663499	1.205715	40.974084	43.982946	34.249300
2	27.323718	451.553155	-0.160783	42.697931	96.652800	34.124261
3	12.309581	422.404983	-0.475931	5.193341	47.467938	8.554563
4	13.210885	410.472999	1.135757	78.695280	61.789672	8.001164
...
9995	15.020523	391.379537	-1.452243	93.417109	25.293814	6.531866
9996	16.772451	346.921190	0.543616	49.882947	96.787402	42.249014
9997	22.370025	486.042136	1.026704	30.659841	15.211825	18.293708
9998	19.430853	337.899776	-0.895329	18.932275	82.774520	42.424255
9999	12.661928	381.172746	2.260788	78.339658	99.243923	41.856539

10000 rows x 6 columns

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```
import matplotlib.pyplot as plt
import pyspark.pandas as ps

# Carregando os dados
file_location = "/FileStore/tables/climate_change_data.csv"
df = ps.read_csv(file_location).to_pandas()

# Criando uma coluna de destino 'target' com base em uma condição (por
# exemplo, temperatura acima da média)
df['target'] = (df['Temperature'] > df['Temperature'].mean()).astype(int)

# Selecionando as variáveis numéricas para o PCA
features = ['Temperature', 'CO2 Emissions', 'Precipitation', 'Humidity', 'Wind
Speed', 'Sea Level Rise']
df_features = df[features]
target = df['target']

# Padronizando os dados
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_features)

# Aplicando PCA
pca = PCA()
pca_components = pca.fit_transform(scaled_features)

# Percentual de variância explicada por cada componente
explained_variance = pca.explained_variance_ratio_

# Plotando a variância explicada acumulada
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance) + 1), explained_variance.cumsum(),
marker='o', linestyle='--')
plt.xlabel('Número de Componentes Principais')
plt.ylabel('Variância Explicada Acumulada')
```

```
plt.title('Variância Explicada pelo PCA')
plt.grid()
plt.show()

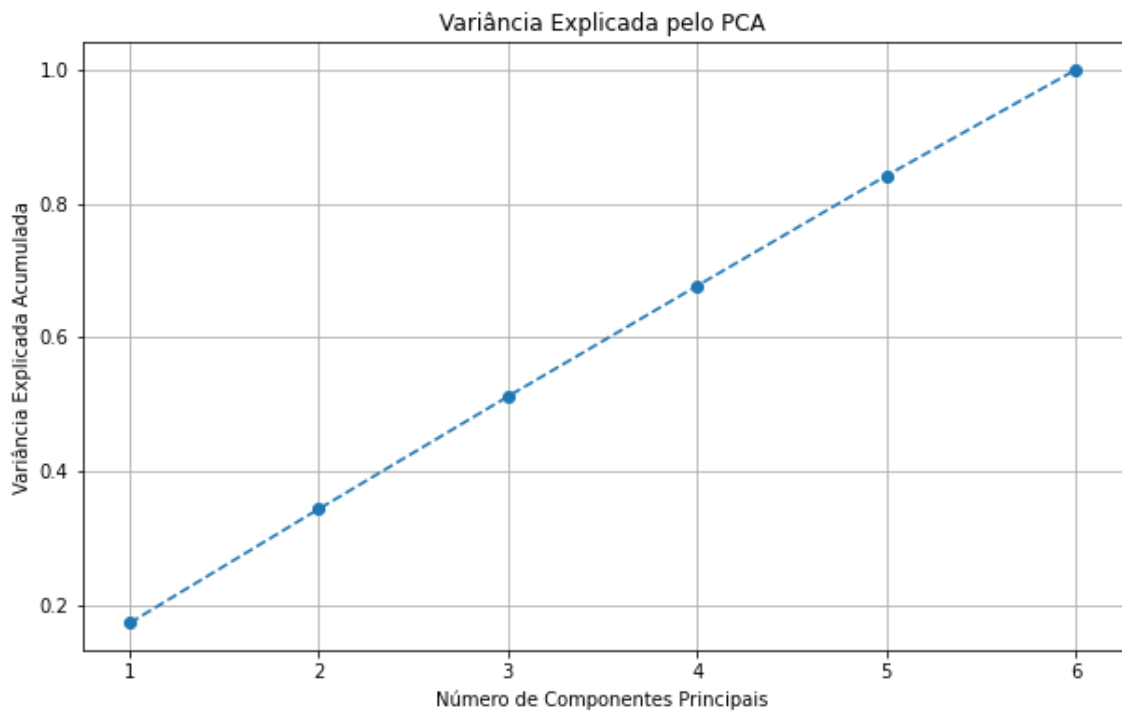
# Usando apenas os primeiros componentes principais que explicam a maioria
da variância
n_components = 3 # Ajuste conforme necessário com base na variância
explicada
pca = PCA(n_components=n_components)
pca_components = pca.fit_transform(scaled_features)

# Dividindo os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(pca_components, target,
test_size=0.2, random_state=42)

# Criando e treinando o modelo MLP
mlp = MLPClassifier(hidden_layer_sizes=(10, 10), activation='relu',
solver='adam', max_iter=1000, random_state=42)
mlp.fit(X_train, y_train)

# Fazendo previsões
y_pred = mlp.predict(X_test)

# Avaliando o modelo
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

O melhor modelo:

Accuracy: 0.7345 Classification Report: precision recall f1-score support 0 0.73 0.75 0.74 997 1 0.74 0.72 0.73 1003 accuracy 0.73 2000 macro avg 0.73 0.73 0.73 2000 weighted avg 0.73 0.73 0.73 2000

Agregar os dataframes

```
df_africa_sul = df.filter(df["Country"] == "South Africa")
display(df_africa_sul)
```

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import pyspark.pandas as ps
```

Carregando os dados

```
file_location = "/FileStore/tables/climate_change_data.csv"
```

```
df = ps.read_csv(file_location).to_pandas()
```

Criando uma coluna de destino 'target' com base em uma condição (por exemplo, temperatura acima da média)

```
df['target'] = (df['Temperature'] > df['Temperature'].mean()).astype(int)
```

Selecionando as variáveis numéricas para o PCA

```
features = ['Temperature', 'CO2 Emissions', 'Precipitation', 'Humidity', 'Wind Speed', 'Sea Level Rise']
```

```
df_features = df[features]
```

```
target = df['target']
```

Padronizando os dados

```
scaler = StandardScaler()
```

```
scaled_features = scaler.fit_transform(df_features)
```

Aplicando PCA

```
pca = PCA()
```

```
pca_components = pca.fit_transform(scaled_features)
```

Percentual de variância explicada por cada componente

```
explained_variance = pca.explained_variance_ratio_
```

Plotando a variância explicada acumulada

```
plt.figure(figsize=(10, 6))
```

```
plt.plot(range(1, len(explained_variance) + 1),  
explained_variance.cumsum(), marker='o', linestyle='--')
```

```
plt.xlabel('Número de Componentes Principais')
```

```
plt.ylabel('Variância Explicada Acumulada')
```

```
plt.title('Variância Explicada pelo PCA')
```

```
plt.grid()
```

```
plt.show()
```

Usando apenas os primeiros componentes principais que explicam a maioria da variância

n_components = 3 # Ajuste conforme necessário com base na variância explicada

pca = PCA(n_components=n_components)

pca_components = pca.fit_transform(scaled_features)

Capturando os índices originais

indices = df.index

Dividindo os dados em treino e teste, incluindo os índices

**X_train, X_test, y_train, y_test, idx_train, idx_test = train_test_split(
pca_components, target, indices, test_size=0.2, random_state=42)**

Criando e treinando o modelo MLP

**mlp = MLPClassifier(hidden_layer_sizes=(10, 10), activation='relu',
solver='adam', max_iter=1000, random_state=42)**

mlp.fit(X_train, y_train)

Fazendo previsões

y_pred = mlp.predict(X_test)

Avaliando o modelo

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

print("Classification Report:\n", classification_report(y_test, y_pred))

Criando um DataFrame com os dados de teste e as previsões

df_test = df.loc[idx_test].copy()

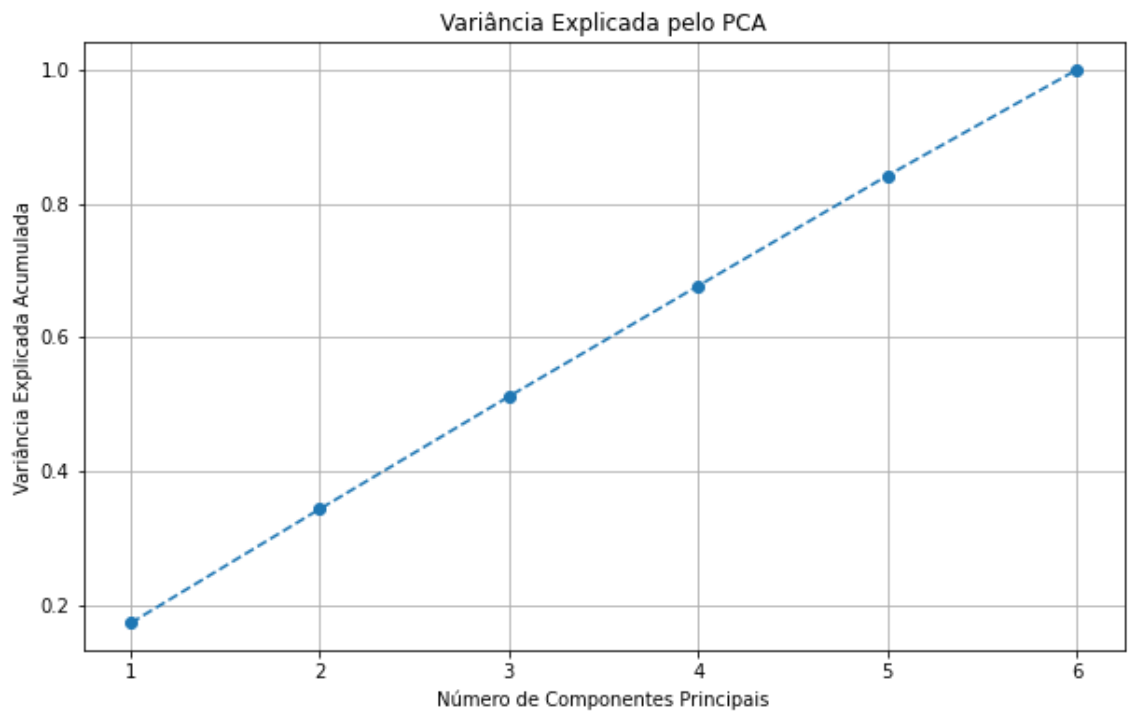
df_test['Prediction'] = y_pred

df_test['Accuracy'] = accuracy # Adicionando a acurácia como uma coluna

Extraíndo o subconjunto para a África do Sul

```
df_south_africa_predictions = df_test[df_test["Country"] == "South Africa"]
```

```
# Exibindo o resultado
print(df_south_africa_predictions)
```



	C02 Emissions	Sea Level Rise	Precipitation	Humidity	Wind Speed
5202	354.396848	0.222505	74.020297	7.062798	49.696142
7942	390.788074	0.774298	27.580919	11.186988	22.104335
9190	417.556846	-0.723507	59.859020	88.764547	39.722428
624	360.570508	2.291537	78.568437	54.779141	39.568239
4397	444.937824	-1.049358	12.812707	25.700580	30.932173
8839	324.475178	-1.319394	20.529996	34.917983	32.663057
4342	415.528641	-0.039382	95.746367	82.516208	16.831612
738	398.566044	-1.177339	34.192164	11.869819	35.270217
8028	390.086223	1.951576	96.529079	16.534903	9.508678
	target	Prediction	Accuracy		
5202	1	1	0.7345		
7942	0	1	0.7345		
9190	1	0	0.7345		
624	0	1	0.7345		
4397	1	0	0.7345		
8839	0	0	0.7345		
4342	0	0	0.7345		
738	1	1	0.7345		
8028	1	1	0.7345		

df_south_africa_predictions

	Date	Location	Country	Temperature	CO2 Emissions	Sea Level Rise	Precipitation	Humidity	Wind Speed	target	Prediction	Accuracy
5202	2011-12-19 02:48:29.810981	New Donnafurt	South Africa	20.190744	354.396848	0.222505	74.020297	7.062798	49.698142	1	1	0.7345
7942	2018-04-07 22:43:57.623762	Oneillton	South Africa	14.079005	390.788074	0.774298	27.580919	11.188988	22.104335	0	1	0.7345
9190	2021-02-19 08:55:44.014401	East Mallory	South Africa	15.173820	417.556846	-0.723507	59.859020	88.764547	39.722428	1	0	0.7345
624	2001-06-08 05:05:53.195319	Stevenburgh	South Africa	12.778918	360.570508	2.291537	78.568437	54.779141	39.588239	0	1	0.7345
4397	2010-02-10 20:23:06.858885	Devinton	South Africa	15.154172	444.937824	-1.049358	12.812707	25.700580	30.932173	1	0	0.7345
8839	2020-04-30 12:03:40.342034	Hernandezbury	South Africa	9.470484	324.475178	-1.319394	20.529906	34.917983	32.663057	0	0	0.7345
4342	2009-12-26 15:28:27.650765	Luisshire	South Africa	12.879226	415.528841	-0.039382	95.746367	82.516208	16.831612	0	0	0.7345
738	2001-09-11 23:34:04.644464	Sarafurt	South Africa	15.770152	396.566044	-1.177339	34.192164	11.869619	35.270217	1	1	0.7345
8028	2018-06-19 04:39:57.839783	South Sharonnton	South Africa	25.555012	390.086223	1.951576	96.529079	16.534903	9.508878	1	1	0.7345

4. Conexão com Azure

```
!pip install pymongo
```

```
import pymongo
from random import randint
import json
```

```
CONNECTION_STRING =
"mongodb://conradcosmosdb:rkN9BM3Cxt5gjTar9hjUfMDeDFR1km4XkUciTn
ZzcbzWMiZwo1SGivGhUrJWQENPDqPAX8exIFYJACDbqpChIA==@conradco
smosdb.mongo.cosmos.azure.com:10255/?ssl=true&replicaSet=globaldb&retry
writes=false&maxIdleTimeMS=120000&appName=@conradcosmosdb@"
DB_NAME = "api-mongodb-sample-database"
UNSHARDED_COLLECTION_NAME = "level_of_the_sea"
SAMPLE_FIELD_NAME = "sample_field"
```

```
def save_documents(collection, documents):
    """Save a list of documents to the collection"""
    for document in documents:
        collection.insert_one(document)
        print("Inserted document:", document)
    print("Saved {} documents to the collection".format(len(documents)))
```

```
def main():
```

```
"""Connect to the API for MongoDB, create DB and collection, perform CRUD
operations"""
```

```
client = pymongo.MongoClient(CONNECTION_STRING)
try:
    client.server_info() # validate connection string
except pymongo.errors.ServerSelectionTimeoutError:
    raise TimeoutError("Invalid API for MongoDB connection string or timed out
when attempting to connect")
```

```
# Retrieve database and collection
db = client[DB_NAME]
collection = db[UNSHARDED_COLLECTION_NAME]

# Convert DataFrame to JSON list of documents
documents =
json.loads(df_south_africa_predictions.to_json(orient="records"))
```

```
# Save the documents to the collection
save_documents(collection, documents)
```

```
if __name__ == '__main__':
    main()
```

Resultado final:

<command-3702893422387475>:15: UserWarning: You appear to be connected to a CosmosDB cluster. For more information regarding feature compatibility and support please visit <https://www.mongodb.com/supportability/cosmosdb>

```
client =
pymongo.MongoClient(CONNECTION_STRING) Inserted document: {'Date':
1324262909810, 'Location': 'New Donnafurt', 'Country': 'South Africa',
'Temperature': 20.190743637, 'CO2 Emissions': 354.3968480395, 'Sea Level
Rise': 0.2225051395, 'Precipitation': 74.0202965159, 'Humidity': 7.0627984437,
```

'Wind Speed': 49.6961420354, 'target': 1, 'Prediction': 1, 'Accuracy': 0.7345, '_id': ObjectId('6737419cc2fea385be522a47'))} Inserted document: {'Date': 1523141037623, 'Location': 'Oneillton', 'Country': 'South Africa', 'Temperature': 14.0790047427, 'CO2 Emissions': 390.7880742136, 'Sea Level Rise': 0.7742977597, 'Precipitation': 27.5809185189, 'Humidity': 11.1869878438, 'Wind Speed': 22.1043351286, 'target': 0, 'Prediction': 1, 'Accuracy': 0.7345, '_id': ObjectId('6737419dc2fea385be522a48'))} Inserted document: {'Date': 1613724944014, 'Location': 'East Mallory', 'Country': 'South Africa', 'Temperature': 15.1736199921, 'CO2 Emissions': 417.5568457389, 'Sea Level Rise': -0.7235071594, 'Precipitation': 59.8590197022, 'Humidity': 88.76454655, 'Wind Speed': 39.7224276894, 'target': 1, 'Prediction': 0, 'Accuracy': 0.7345, '_id': ObjectId('6737419dc2fea385be522a49'))} Inserted document: {'Date': 991976753195, 'Location': 'Stevenburgh', 'Country': 'South Africa', 'Temperature': 12.7789182391, 'CO2 Emissions': 360.5705076473, 'Sea Level Rise': 2.2915371671, 'Precipitation': 78.5684367018, 'Humidity': 54.7791410538, 'Wind Speed': 39.5682394526, 'target': 0, 'Prediction': 1, 'Accuracy': 0.7345, '_id': ObjectId('6737419dc2fea385be522a4a'))} Inserted document: {'Date': 1265833386858, 'Location': 'Devinton', 'Country': 'South Africa', 'Temperature': 15.1541717593, 'CO2 Emissions': 444.9378238186, 'Sea Level Rise': -1.0493580954, 'Precipitation': 12.8127065552, 'Humidity': 25.7005804862, 'Wind Speed': 30.9321732121, 'target': 1, 'Prediction': 0, 'Accuracy': 0.7345, '_id': ObjectId('6737419ec2fea385be522a4b'))} Inserted document: {'Date': 1588248220342, 'Location': 'Hernandezbury', 'Country': 'South Africa', 'Temperature': 9.4704644004, 'CO2 Emissions': 324.4751779896, 'Sea Level Rise': -1.3193941992, 'Precipitation': 20.5299958501, 'Humidity': 34.9179825866, 'Wind Speed': 32.6630567408, 'target': 0, 'Prediction': 0, 'Accuracy': 0.7345, '_id': ObjectId('6737419ec2fea385be522a4c'))} Inserted document: {'Date': 1261841307650, 'Location': 'Luisshire', 'Country': 'South Africa', 'Temperature': 12.8792257407, 'CO2 Emissions': 415.5286411381, 'Sea Level Rise': -0.039382306, 'Precipitation': 95.7463671718, 'Humidity': 82.5162084522, 'Wind Speed': 16.8316124757, 'target': 0, 'Prediction': 0, 'Accuracy': 0.7345, '_id': ObjectId('6737419ec2fea385be522a4d'))}

```
def save_documents(collection, documents):
    """Save a list of documents to the collection"""
    for document in documents:
        collection.insert_one(document)
    print(f"Saved {len(documents)} documents to the collection")

def main():
    """Connect to the API for MongoDB, create DB and collection, perform CRUD operations"""
    client = pymongo.MongoClient(CONNECTION_STRING)
    try:
        client.server_info() # validate connection string
    except pymongo.errors.ServerSelectionTimeoutError:
        raise TimeoutError("Invalid API for MongoDB connection string or timed out when attempting to connect")

    # Retrieve database and collection
    db = client[DB_NAME]
    collection = db[MONGODB_COLLECTION_NAME]

    # Convert DataFrame to JSON list of document
    documents = json.loads(df_south_africa_predictions.to_json(orient='records'))

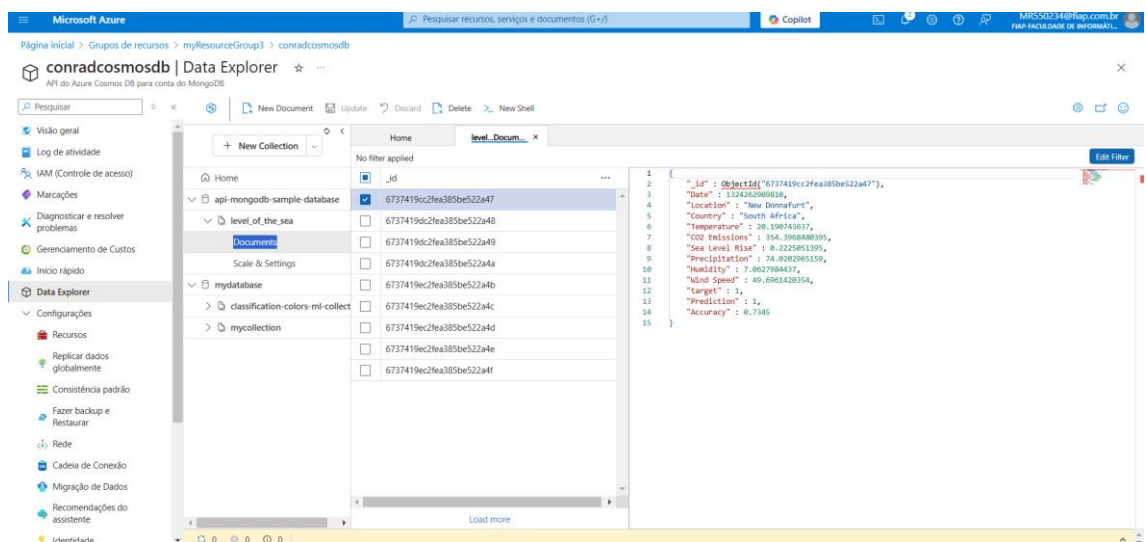
    # Save the documents to the collection
    save_documents(collection, documents)

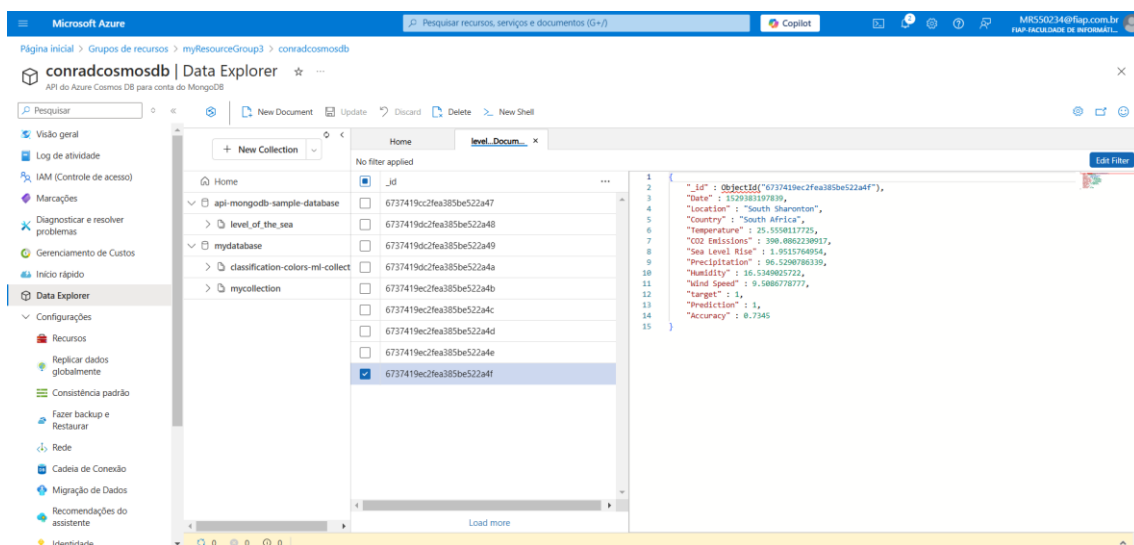
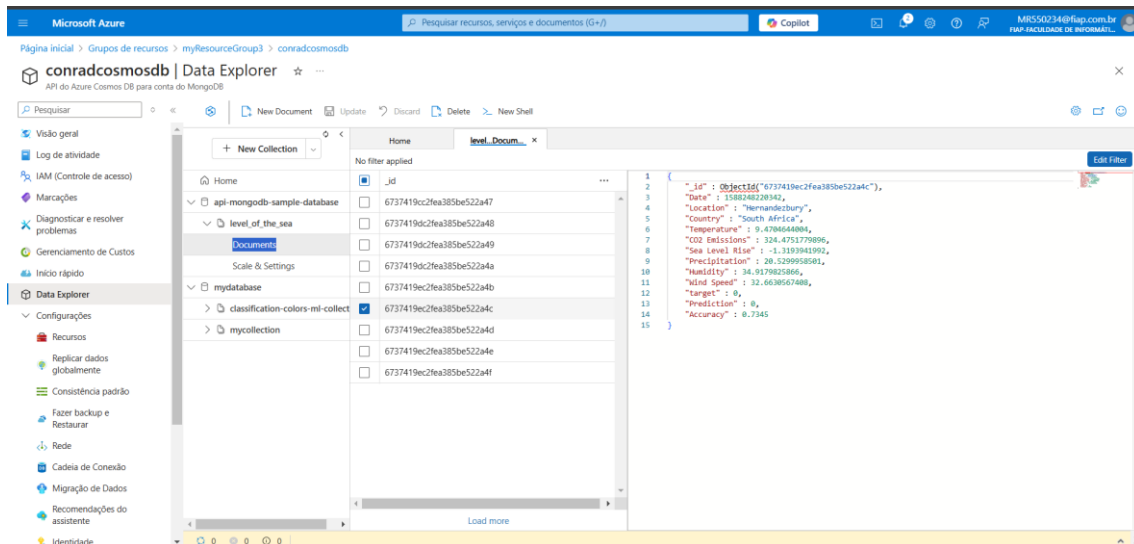
if __name__ == '__main__':
    main()
```

```
<Command-3f0289f422367475b15> UserWarning: You appear to be connected to a CosmosDB cluster. For more information regarding feature compatibility and support please visit https://www.mongodb.com/supportability/cosmosdb
client = pymongo.MongoClient(CONNECTION_STRING)
Inserted document: {'Date': 132426290810, 'Location': 'New Donnafurt', 'Country': 'South Africa', 'Temperature': 20.190743637, 'CO2 Emissions': 354.3968408395, 'Sea Level Rise': 0.2225051395, 'Precipitation': 74.6202965159, 'Humidity': 7.0627984437, 'Wind Speed': 49.6961420354, 'target': 1, 'Prediction': 1, 'Accuracy': 0.7345, 'id': ObjectId('6737419cc2fea385be522a47')}
Inserted document: {'Date': 1523141837623, 'Location': 'O'Neillton', 'Country': 'South Africa', 'Temperature': 14.0790047427, 'CO2 Emissions': 390.7880742136, 'Sea Level Rise': 0.7742977597, 'Precipitation': 27.5809185189, 'Humidity': 11.1869878438, 'Wind Speed': 22.1043351286, 'target': 0, 'Prediction': 1, 'Accuracy': 0.7345, 'id': ObjectId('6737419dc2fea385be522a48')}
Inserted document: {'Date': 1613724944814, 'Location': 'East Mallory', 'Country': 'South Africa', 'Temperature': 15.1736199921, 'CO2 Emissions': 417.5568467389, 'Sea Level Rise': -0.7235071594, 'Precipitation': 59.8590397022, 'Humidity': 88.76454655, 'Wind Speed': 29.7224276894, 'target': 1, 'Prediction': 0, 'Accuracy': 0.7345, 'id': ObjectId('6737419dc2fea385be522a49')}
Inserted document: {'Date': 991976752136, 'Location': 'Stevenburgh', 'Country': 'South Africa', 'Temperature': 12.7789182291, 'CO2 Emissions': 360.5705876473, 'Sea Level Rise': 2.2915371671, 'Precipitation': 78.5684367018, 'Humidity': 54.7791410538, 'Wind Speed': 39.5682394526, 'target': 0, 'Prediction': 1, 'Accuracy': 0.7345, 'id': ObjectId('6737419dc2fea385be522a4a')}
Inserted document: {'Date': 126583338858, 'Location': 'Devinton', 'Country': 'South Africa', 'Temperature': 15.1541717593, 'CO2 Emissions': 444.9378238188, 'Sea Level Rise': -1.0493580954, 'Precipitation': 12.8127065552, 'Humidity': 25.7005884862, 'Wind Speed': 30.9321732121, 'target': 1, 'Prediction': 0, 'Accuracy': 0.7345, 'id': ObjectId('6737419ec2fea385be522a4b')}
Inserted document: {'Date': 1588246220342, 'Location': 'Hernandezbury', 'Country': 'South Africa', 'Temperature': 9.4704644004, 'CO2 Emissions': 324.4751779896, 'Sea Level Rise': -1.3193941992, 'Precipitation': 20.5299958501, 'Humidity': 34.9179825089, 'Wind Speed': 32.6030952480, 'target': 0, 'Prediction': 0, 'Accuracy': 0.7345, 'id': ObjectId('6737419ec2fea385be522a4c')}
Inserted document: {'Date': 1261841307659, 'Location': 'Lutishire', 'Country': 'South Africa', 'Temperature': 12.8392574007, 'CO2 Emissions': 415.5280411381, 'Sea Level Rise': -0.839382386, 'Precipitation': 95.7463671718, 'Humidity': 82.5162084522, 'Wind Speed': 16.8316124757, 'target': 0, 'Prediction': 0, 'Accuracy': 0.7345, 'id': ObjectId('6737419ec2fea385be522a4d')}
Inserted document: {'Date': 1000251244444, 'Location': 'Sarafurt', 'Country': 'South Africa', 'Temperature': 15.7701516776, 'CO2 Emissions': 396.5660440424, 'Sea Level Rise': -1.177338915, 'Precipitation': 34.192164313, 'Humidity': 11.8698190595, 'Wind Speed': 35.2702169577, 'target': 1, 'Prediction': 1, 'Accuracy': 0.7345, 'id': ObjectId('6737419ec2fea385be522a4e')}
Inserted document: {'Date': 1529383197839, 'Location': 'South Sharonton', 'Country': 'South Africa', 'Temperature': 25.555017725, 'CO2 Emissions': 390.0862230917, 'Sea Level Rise': 1.9515764954, 'Precipitation': 96.5290786339, 'Humidity': 16.5349025722, 'Wind Speed': 9.5086778777, 'target': 1, 'Prediction': 1, 'Accuracy': 0.7345, 'id': ObjectId('6737419ec2fea385be522a4f')}
Saved 9 documents to the collection
```

Se precisar da query na Azure:

db.level_of_the_sea.find({ Prediction: { \$exists: true } })





Excluir grupo de recursos para evitar consumir créditos:

