UNIVERSIDADE
DE TRÁS-OS-MONTES
E ALTO DOURO

Trabalho Experimental 1

Licenciatura em Engenharia Informática

Introdução à Ciência dos Dados

Paulo Nogueira Martins

Frederico Augusto dos Santos Branco

Autores

Diogo Medeiros n.º 70633

Eduardo Chaves n.º 70611

João Rodrigues n.º 70579

Pedro Silva n.º 70649

Rui Pinto n.º 70648

Vila Real, maio 2022

ÍNDICE

1. INTRODUÇÃO

No âmbito da Unidade Curricular de Introdução à Ciência dos Dados, foi solicitado um trabalho experimental que consiste no desenvolvimento de um conjunto de tarefas relacionado com o dataset "Our World in Data – Energy".

Estas tarefas foram desenvolvidas em Python, na IDE PyCharm, com recurso a múltiplas bibliotecas, e serão descritas e contextualizadas no próximo capítulo.

Este relatório é acompanhado de um Jupyter Notebook, o qual contém todo o código subjacente às tarefas realizadas, devidamente comentado.

2. TAREFAS

2.1 Tarefa 1

A primeira tarefa consiste em carregar os dados do dataset "Our World in Data – Energy" para um Dataframe, filtrar os dados dos seguintes países – Estados Unidos, Canadá, Brasil e México para um novo Dataframe, e gravar esses dados num novo ficheiro CSV.

O código que se encontra a seguir descreve os passos necessários, recorrendo à biblioteca Pandas, para executar esta tarefa.

```
import pandas as pd

# Ler dados
owid_energy_data = pd.read_csv('owid-energy-data.csv')

# Filtrar dados
countries = ['United States', 'Canada', 'Brazil', 'Mexico']
energy_data = owid_energy_data[
        owid_energy_data['country'].isin(countries)]

# Salvar dados
energy_data.to_csv('filtered-energy-data.csv')
```

2.2 Tarefa 2

Com o recurso à biblioteca Matplotlib, criou-se uma figura composta por quatro gráficos, um para cada país da tarefa 1, descrevendo a evolução da produção de eletricidade a partir de petróleo, ao longo dos anos.

```
import matplotlib.pyplot as plt
# Criar figura e definir cores
plt.figure()
colors = ['blue', 'red', 'green', 'yellow']
# Plot dos dados de 'oil_electricity', para cada país
for country, color in zip(countries, colors):
    data = energy_data[(energy_data.country == country)
                       & (energy_data['oil_electricity'].notnull())]
    plt.plot(data.year, data['oil_electricity'], '-', color=color,
        label=country)
# Customizar figura
plt.xlabel('year')
plt.ylabel('oil electricity')
plt.title('Electricity production from oil in certain countries')
plt.legend()
plt.show()
```

Como é possível observar, existe uma clara tendência de diminuição da eletricidade produzida a partir de petróleo, independentemente do país.

Por outro lado, é possível identificar claros picos de produção em certas décadas, por exemplo, no final da década de 80 e 90 e no início de 2000.

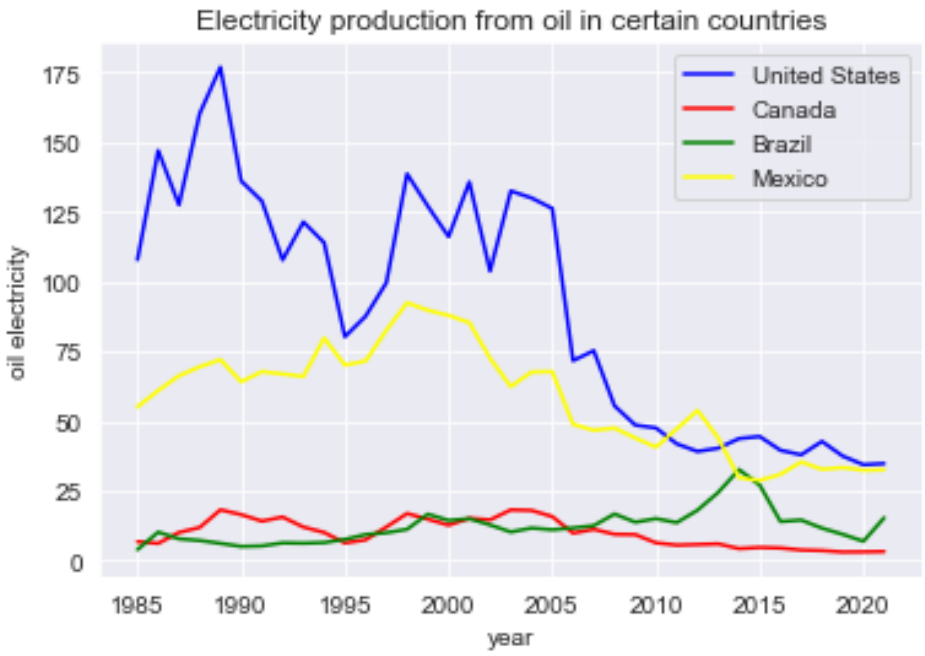

Fig. 1 – Evolução da produção de eletricidade a partir do petróleo

2.3 Tarefa 3

Com base nos dados de energia dos Estados Unidos no ano de 2010, foi criado um gráfico circular representando as diferentes fontes de produção de energia elétrica.

```
# Dados dos Estados Unidos em 2010
usa_energy_data = energy_data[(energy_data.country == 'United States')
                              & (energy_data.year == 2010)]
# Filtrar segundo fontes de eletricidade
elec_sources = ['coal_electricity', 'biofuel_electricity',
                'fossil_electricity', 'gas_electricity',
                'hydro_electricity', 'nuclear_electricity',
                'oil_electricity']
usa_electricity = usa_energy_data[elec_sources].values.flatten().tolist()
# Plot dos dados num gráfico circular
plt.figure()
plt.title('Electricity production from the United States in 2010')
elec_sources = [s.replace('_', ' ').capitalize() for s in elec_sources]
plt.pie(usa_electricity, labels=elec_sources)
plt.show()
```

Com base na análise do gráfico produzido, é possível concluir que as principais fontes de produção de eletricidade foram o carvão e os combustíveis fósseis, constituindo quase 2/3 do total de eletricidade produzida.

As fontes que menos contribuíram para a produção de eletricidade nos Estados Unidos, em 2010, foram a biomassa e o petróleo.

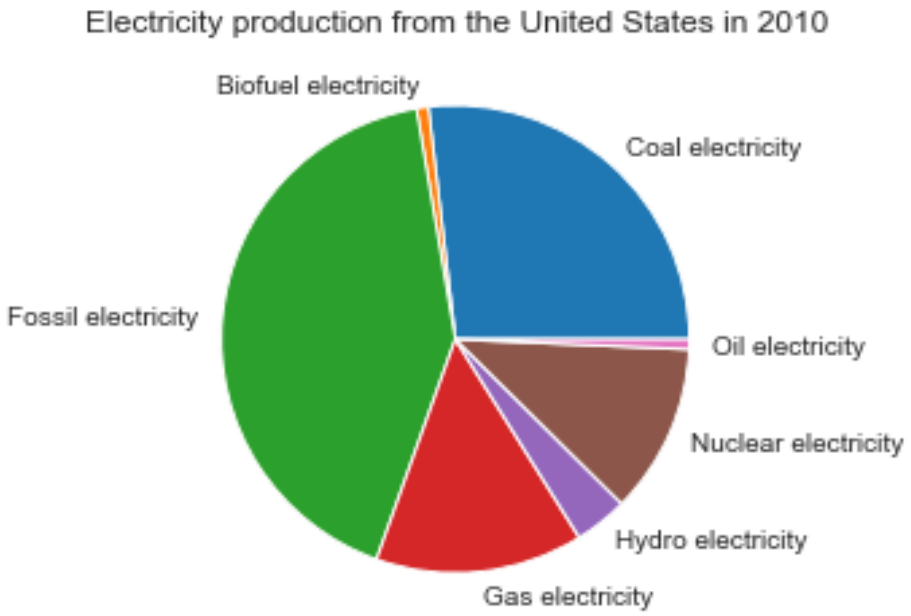

Fig. 2 – Produção de eletricidade nos Estados Unidos, em 2010

2.4 Tarefa 4

O código a seguir descreve uma função responsável por determinar o ano de maior consumo de energia nuclear, num certo país, bem como o seu valor.

Para exemplificar o seu funcionamento, executou-se a função para cada um dos países da tarefa 1.

```
import numpy as np

# Determinar ano e valor de maior consumo de energia nuclear
def highest_nuclear_consumption(country: str) -> np.float64:
    country_data: pd.DataFrame = energy_data.loc[
        (energy_data.country == country), ['year', 'nuclear_consumption']]
    i = country_data['nuclear_consumption'].idxmax()
    year = country_data.loc[i, 'year']
    consumption = country_data.loc[i, 'nuclear_consumption']
    print(f'Nuclear consumption of {country} in {year} was {consumption}')
    return consumption

# Exemplos para lista de países
print('Highest nuclear consumption year per country:')
[highest_nuclear_consumption(country) for country in countries]
```

Os resultados obtidos encontram-se descritos na tabela seguinte. Como é possível observar, o país com o pico mais antigo de consumo de energia nuclear foi o Canadá, em 1994.

País	Ano	Consumo de energia nuclear
Estados Unidos	2007	2254.808
Canadá	1994	297.456
Brasil	2012	41.311
México	2013	30.211

Tabela 1 – Maior consumo de energia nuclear, por país

2.5 Tarefa 5

Com o recurso à biblioteca Seaborn, criaram-se gráficos de dispersão que relacionam o consumo de gás e o consumo de energia nuclear, para cada país e em geral.

```
import seaborn as sns
# Plot de dispersão geral
sns.regplot(data=owid_energy_data, x='gas_consumption',
y='nuclear_consumption')
plt.xlabel('Gas consumption')
plt.ylabel('Nuclear consumption')
plt.title(f'Gas consumption / Nuclear consumption')
# Plot de dispersão para cada país
grid = sns.FacetGrid(energy_data, col="country", hue="country",
        col_wrap=2, sharex=False, sharey=False)
grid.map(sns.regplot, 'gas_consumption', 'nuclear_consumption')
grid.add_legend()
plt.show()
```

Pela análise do gráfico produzido, é possível concluir que, apesar do consumo de energia nuclear aumentar, num geral, com o consumo de gás, esta relação é ténue, algo que se torna evidente quando analisamos a regressão linear ajustada aos dados.

Estas observações permitem-nos concluir que, ao nível global, existe um certo grau de independência entre as variáveis.

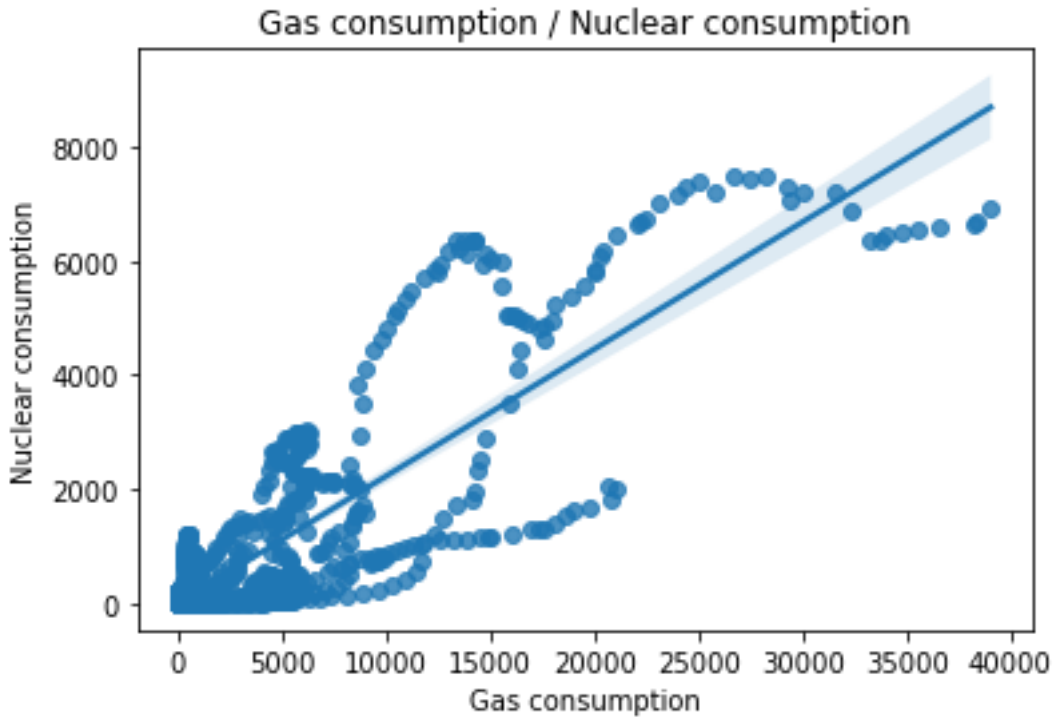

Fig. 3 – Consumo de energia nuclear por consumo de gás, ao nível global

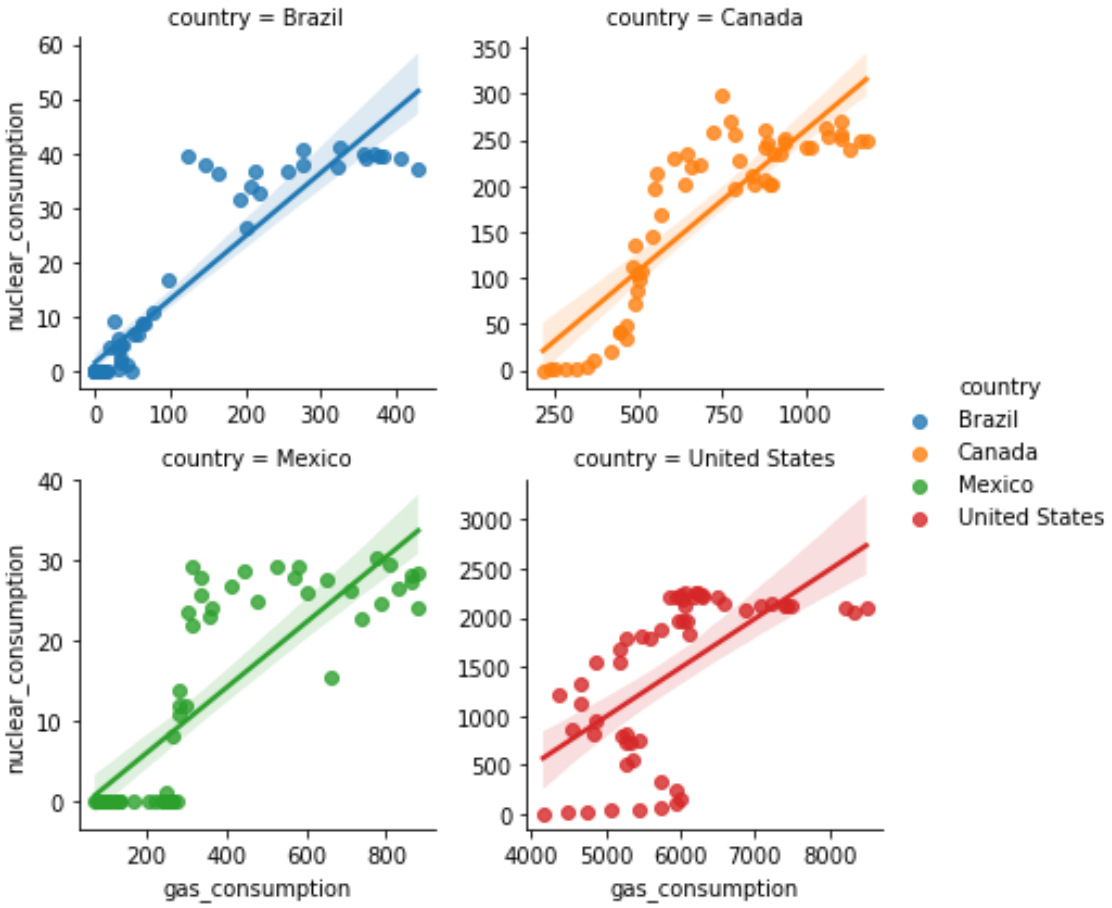

Fig. 4 – Consumo de energia nuclear por consumo de gás, ao nível de cada país

Quando analisando os gráficos de dispersão dos quatro países em questão, verifica-se que o país cuja regressão melhor se ajusta às variáveis é o Canadá.

O gráfico de dispersão dos Estados Unidos, em particular, evidencia uma forte independência linear entre o consumo de energia nuclear e de gás, não havendo qualquer relação aparente entre estas variáveis.

2.6 Tarefa 6

Para concluir, era pedido que se explorassem técnicas de Machine Learning tal que fosse possível fazer uma previsão sobre o consumo de energia solar no futuro.

Começando por uma técnica mais simples, analisaram-se os dados relativos ao Estados Unidos, e verificou-se que estes apresentavam uma disposição semelhante a uma função exponencial.

Por esta razão, e com o recurso à biblioteca SciPy, optou-se por usar o algoritmo *curve_fit*, o qual recorre à técnica de análise não linear de mínimos quadrados para ajustar uma função *f* a um conjunto de dados.

Para facilitar a otimização, foi tida como variável independente, *x,* o ano, ajustado a 0. Já no caso da variável dependente, *y*, esta corresponde aos valores do consumo de energia solar, a partir de 1965 (valores não nulos).

```
# Task 6 - Using a curve fitting algorithm
import numpy as np
import pandas as pd
import scipy.optimize as opt
import matplotlib.pyplot as plt

# Exponential function with generic parameters
def mono_exp(x, m, t, b):
    return m * np.exp(t * x) + b

# Solar consumption data for the United States
solar_data: pd.DataFrame = energy_data.loc[
        (energy_data.country == 'United States')
        &(energy_data.solar_consumption.notnull()),
        ['year', 'solar_consumption']]

# Training data, xs is adjusted for better optimization
xs = solar_data['year'].to_numpy()
ys = solar_data['solar_consumption'].to_numpy()
xs_shifted = xs - xs[0]

# perform the fit
p0 = (1, 1e-6, 0)  # start with values near those we expect
params, cv = opt.curve_fit(mono_exp, xs_shifted, ys, p0)
m, t, b = params

# determine quality of the fit
squaredDiffs = np.square(ys - mono_exp(xs_shifted, m, t, b))
squaredDiffsFromMean = np.square(ys - np.mean(ys))
rSquared = 1 - np.sum(squaredDiffs) / np.sum(squaredDiffsFromMean)
print(f'R² = {rSquared}')
print(f'Y = {m:.3e} * e^(-{t:.3} * x) + {b:.3}')
```

```
# plot the results
plt.plot(xs_shifted, ys, '.', label='data')
plt.plot(xs_shifted, mono_exp(xs_shifted, m, t, b), '-',
        label='fitted')
plt.title(f'Solar consumption of {countries[0]}')
xlocs, _ = plt.xticks()
plt.xticks(xlocs, xlocs.astype(int) + xs[0])
plt.xlabel('year')
plt.legend(['Actual', 'Forecast'])
plt.show()

# Next year forecast
next_year = xs[-1] + 1
print(f'Forecast for {next_year}: {mono_exp(next_year - xs[0], m, t,
b):.3f}')
```

Após executar este código, obteve-se a seguinte figura, retratando os dados reais do consumo de energia solar, bem como a curva exponencial ajustada pelo algoritmo.

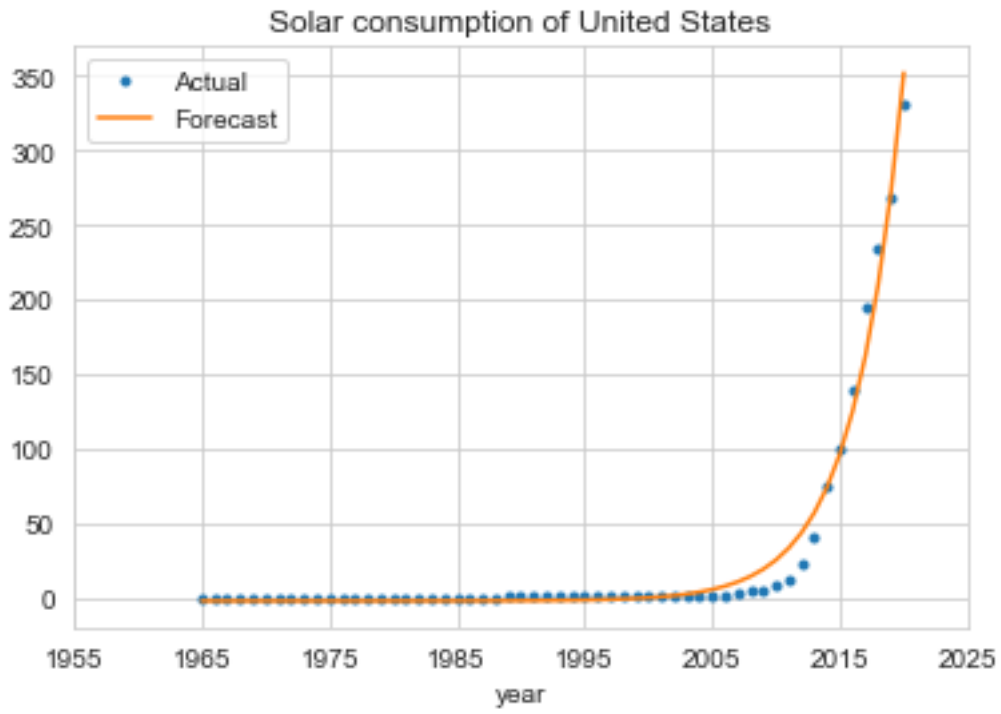

Fig. 5 – Função exponencial ajustada ao consumo de energia solar, nos Estados Unidos

A curva final tem de equação $y = 2.787 * 10^{-4} * e^{-0.256*x} - 1.84$, com $R^2 = 0.984$. Com esta equação foi possível fazer uma previsão do consumo de energia solar para o ano de 2021, a qual se revelou em 454.650.

No entanto, a natureza dos dados levantou questões sobre que outro tipo de algoritmos e modelos de ML poderiam ser vantajosos no problema em questão.

Depois de alguma procura, chegou-se a um tipo de rede neuronal que se adequava ao tipo de dados com que estaríamos a trabalhar: LSTM. Em suma, LSTMs são um tipo particular de RNN (Redes Neuronais Recorrentes) capazes de aprender dependências a longo prazo. Redes LSTM são fantásticas a classificar, processar e fazer previsões com base em séries temporais.

Estando escolhido o modelo, faltava agora implementá-lo e treiná-lo. O código que se encontra de seguida detalha essa implementação, a qual se apoia na biblioteca scikit-learn para a normalização dos dados, bem como na framework TensorFlow e API Keras para a criação e treino do modelo, bem como previsão do próximo ano.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.models import Sequential

# United States solar data
solar_data: pd.DataFrame = energy_data.loc[
    (energy_data.country == countries[0]) &
(energy_data.solar_consumption.notnull()),
    ['year', 'solar_consumption']]
solar_data.set_index('year', inplace=True)
y = solar_data['solar_consumption'].fillna(method='ffill')
y = y.to_numpy().reshape(-1, 1)

# scale the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaler = scaler.fit(y)
y = scaler.transform(y)

# generate the input and output sequences
n_lookback = 10  # length of input sequences (lookback period)
n_forecast = 1  # length of output sequences (forecast period)
X = []
Y = []
for i in range(n_lookback, len(y) - n_forecast + 1):
    X.append(y[i - n_lookback: i])
    Y.append(y[i: i + n_forecast])
X = np.array(X)
Y = np.array(Y)

# create model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(n_lookback,
1)))
model.add(LSTM(units=50))
model.add(Dense(n_forecast))
model.compile(loss='mean_squared_error', optimizer='adam')
# fit model
model.fit(X, Y, epochs=100, batch_size=50)
```

```
# generate the forecasts
X_ = y[- n_lookback:]  # last available input sequence
X_ = X_.reshape(1, n_lookback, 1)
Y_ = model.predict(X_).reshape(-1, 1)
Y_ = scaler.inverse_transform(Y_)

# organize the results in a data frame
df_past = solar_data[['solar_consumption']].reset_index()
df_past.rename(columns={'index': 'year', 'solar_consumption': 'Actual'},
inplace=True)
df_past['year'] = pd.date_range(start=str(solar_data.index[0]),
                                periods=len(solar_data), freq='AS')
df_past['Forecast'] = np.nan
df_past.at[df_past.index[-1], 'Forecast'] = df_past.at[df_past.index[-1],
'Actual']

df_future = pd.DataFrame(columns=['year', 'Actual', 'Forecast'])
df_future['year'] = pd.date_range(start=df_past.at[df_past.index[-1],
'year'] + pd.DateOffset(months=12),
                                  periods=n_forecast, freq='AS')
df_future['Forecast'] = Y_.flatten()
df_future['Actual'] = np.nan
results = pd.concat([df_past, df_future]).set_index('year')

# plot the results
results.plot(title=f'Solar consumption of {countries[0]}')
plt.show()

# Prediction for next year
print(f'Forecast for {results.index[-1].year}: {results.at[results.index[-
1], "Forecast"]:.3f}')
```

Inicialmente, geram-se as sequências de entrada e saída (X e Y), tendo-se optado por um período de retrospetiva (lookback) de 10 anos e um ano de antevisão.

Construi-se um modelo sequencial, composto por 2 LSTMs e uma rede densamente conectada, e compilou-se, optando pelo erro quadrático médio para função de perda, e pelo otimizador Adam.

De seguida, treinou-se o modelo com os dados X e Y, durante 100 épocas e com batches de 50 exemplares, fazendo uso do CUDA para maximizar a performance.

Por fim, gerou-se a previsão para o ano de 2021 e organizaram-se os resultados, a fim de gerar um gráfico que melhor retratasse o problema estudado. Segue-se o gráfico gerado, retratando a evolução do consumo de energia solar nos Estados Unidos, incluindo os dados reais até 2020 e a previsão para 2021.

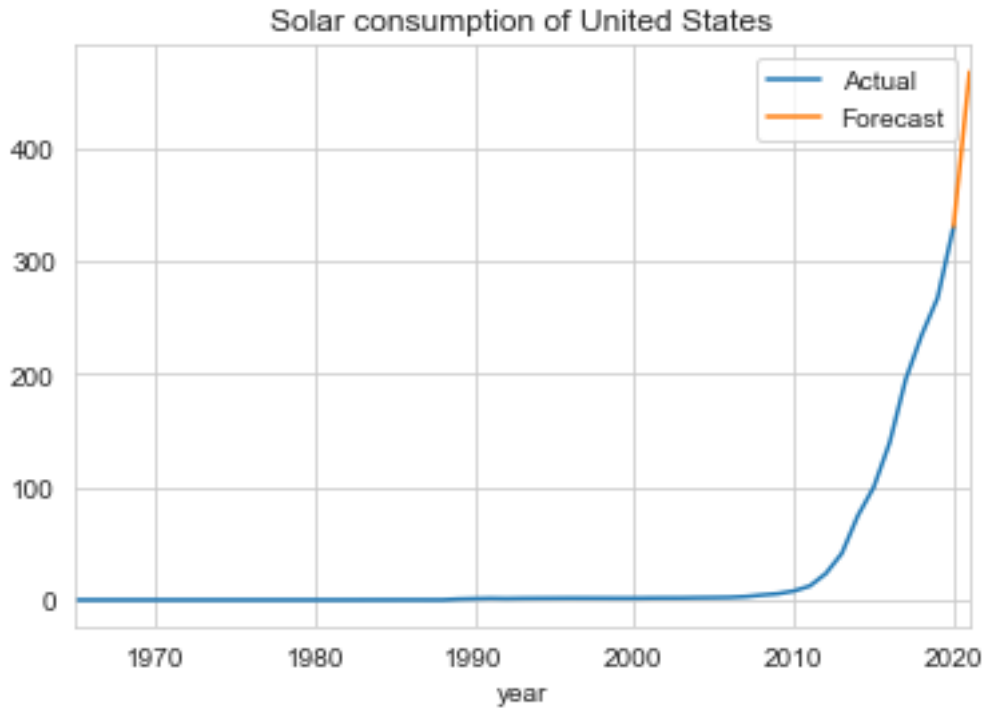

Fig. 6 – Representação gráfica do consumo de energia solar nos Estados Unidos. A azul, encontram-se representados os dados reais, e a laranja a previsão do modelo.

Com uma perda final de $2.7 * 10^{-3}$, o modelo foi capaz de se ajustar eficazmente aos dados que lhe foram fornecidos, sem ocorrer *overfitting*.

Após o treino, o modelo foi capaz de fazer uma previsão para o consumo de energia solar em 2021, o qual revelou ser de 467.384. Este valor é deveras próximo do obtido usando a técnica de análise não linear de mínimos quadrados, o que nos permite concluir que a LSTM não só se adequa ao problema em mão, mas é eficaz no processo.

3. NOTAS FINAIS

Concluído o presente trabalho experimental, todas as tarefas foram concretizadas, com especial atenção à tarefa 6, na qual foram exploradas diversas técnicas de Machine Learning e obtidos resultados satisfatórios.

O desenvolvimento deste trabalho permitiu adquirir competências e conceitos relacionados com a linguagem utilizada, Python, com as bibliotecas exploradas – Pandas, NumPy, Matplotlib, Seaborn, bem como tópicos de ML, até então desconhecidos.