

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 1](#_Toc102923136)

[2. TAREFAS 1](#_Toc102923137)

[2.1 Tarefa 1 1](#_Toc102923138)

[2.2 Tarefa 2 1](#_Toc102923139)

[2.3 Tarefa 3 1](#_Toc102923140)

[2.4 Tarefa 4 2](#_Toc102923141)

[2.5 Tarefa 5 2](#_Toc102923142)

[2.6 Tarefa 6 2](#_Toc102923143)

[3. NOTAS FINAIS 3](#_Toc102923144)

[**BIBLIOGRAFIA** 3](#_Toc102923145)

1. INTRODUÇÃO

No âmbito da Unidade Curricular de Inteligência Artificial, foi solicitado um trabalho prático que consiste no desenvolvimento de um sistema computacional com agentes racionais usando a ferramenta NetLogo.

No nosso caso, o ambiente retratado representa um campo (ou terreno de plantação), e os agentes modelados são o planta-relva e as toupeiras, responsáveis, respetivamente, pela plantação e destruição da relva.

1. TAREFAS

“Um agente é qualquer coisa que perceciona o seu ambiente através de sensores e atua sobre esse ambiente através de atuadores.” (Russel & Norvig, 2010)

* 1. Tarefa 1

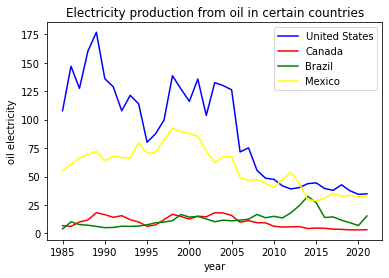
**import** pandas **as** pd  
  
owid\_energy\_data = pd.read\_csv('owid-energy-data.csv')  
  
countries = ['United States', 'Canada', 'Brazil', 'Mexico']  
energy\_data = owid\_energy\_data[owid\_energy\_data['country'].isin(countries)]  
energy\_data.to\_csv('filtered-energy-data.csv')

* 1. Tarefa 2

Segue-se um conjunto de imagens capturadas durante o funcionamento do sistema, retratando as várias fases do mesmo em execução, de acordo com certas condições iniciais.

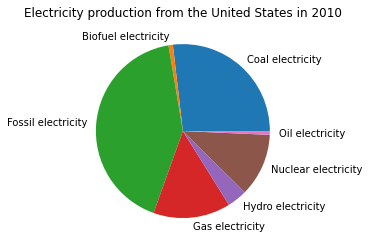
**import** matplotlib.pyplot **as** plt

plt.figure()  
colors = ['blue', 'red', 'green', 'yellow']  
  
**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)  
  
plt.xlabel('year')  
plt.ylabel('oil electricity')  
plt.title('Electricity production from oil in certain countries')  
plt.legend()  
plt.show()



* 1. Tarefa 3

usa\_energy\_data = energy\_data[(energy\_data.country == 'United States')  
 & (energy\_data.year == 2010)]  
  
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()  
  
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()



* 1. Tarefa 4

**import** numpy **as** np

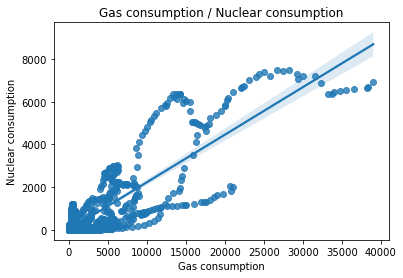
**def** **highest\_nuclear\_consumption**(country: str) -> np.float64:  
 country\_data: pd.DataFrame = energy\_data.loc[(energy\_data.country == country),  
 ['year', 'nuclear\_consumption']]  
 i = country\_data['nuclear\_consumption'].idxmax()  
 year = country\_data.loc[i, 'year']  
 consumption = country\_data.loc[i, 'nuclear\_consumption']  
 print(f'Nuclear consumption of {country} in {year} was {consumption}')  
 **return** consumption  
  
print('Highest nuclear consumption year per country:')  
[highest\_nuclear\_consumption(country) **for** country **in** countries]

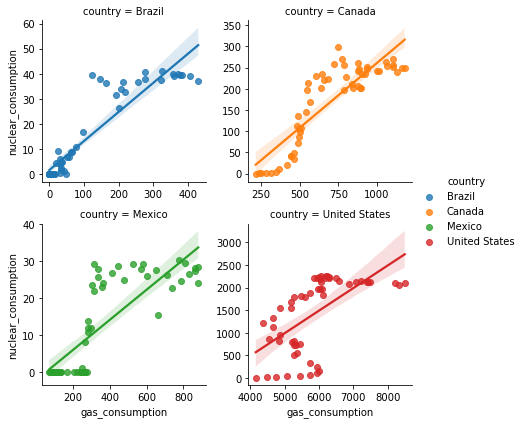
|  |  |  |
| --- | --- | --- |
| País | Ano | Consumo de energia nuclear |
| Estados Unidos | 2007 | 2254.808 |
| Canadá | 1994 | 297.456 |
| Brasil | 2012 | 41.311 |
| México | 2013 | 30.211 |

* 1. Tarefa 5

**import** seaborn **as** sns

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')  
  
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()





* 1. Tarefa 6

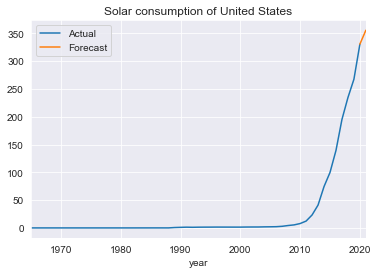
**import** numpy **as** np  
**import** pandas **as** pd  
**import** scipy.optimize **as** opt  
**import** matplotlib.pyplot **as** plt  
  
**def** **mono\_exp**(x, m, t, b):  
 **return** m \* np.exp(t \* x) + b  
  
solar\_data = energy\_data.loc[(energy\_data.country == countries[0]) & (energy\_data.solar\_consumption.notnull()),  
 ['year', 'solar\_consumption']]  
  
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}')

# inspect the parameters  
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}')

Gráfico anterior

**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 = 3 # 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)  
# fit the 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')  
model.fit(X, Y, epochs=500, 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}')



1. NOTAS FINAIS

Concluído o presente trabalho prático, estamos deveras satisfeitos com o mesmo, tendo a referir que surgiram algumas dificuldades quanto á melhor forma de definir o algoritmo. Isto, porque grande parte do funcionamento do sistema foi deixado ao nosso critério, levando a melhorias executadas para além do definido em protocolo, melhorias essas que desencadearam uma série de novas ideias a aplicar, no entanto, decidimos focar-nos na base resultante do trabalho já desenvolvido.

Em anexo está o código relativo ao respetivo trabalho prático, bem como uma implementação alternativa ao mundo aberto (toro), num mundo fechado (caixa), que entendemos adicionar ao trabalho.

# BIBLIOGRAFIA

*Alguns dos mais famosos erros de softwares da história*. (10 de novembro de 2021). Obtido de Profissionais TI: https://www.profissionaisti.com.br/alguns-dos-mais-famosos-erros-de-softwares-da-historia/​

Oliveira, J. P. (2021). Agentes Inteligentes. *ESW - Requisitos*. Universidade de Trás-os-Montes e Alto Douro.

Russel, S. J., & Norvig, P. (2010). 2 Intelligent Agents. Em *Artificial Intelligence: A Modern Approach, 3rd ed.* (pp. 34-63). Upper Saddle River, New Jersey 07458: Pearson Education. Inc.