

Rapport TP sur la visualisation des données et l'entraînement du modèle RNA

Mahmoud El Mehdi El Khadiri

Supervised by
Souad Abderafie

October 24, 2023



Contents

1	Introduction	2
2	Analyse des données	2
2.1	Boîte à moustaches	2
2.2	Matrice de corrélation	3
3	Développement du modèle	3
3.1	Valeur Calculée vs Valeur Expérimentale	4
4	Distribution des Résidus	4
5	Métriques d'évaluation	5
6	Conclusion	5
7	Annexe	5
7.1	Code d'entraînement	5

1 Introduction

Ce rapport présente les résultats obtenus à partir d'une base de données. Après une analyse initiale des données, un modèle de réseau de neurones a été développé pour prédire la relation entre les variables.

2 Analyse des données

2.1 Boîte à moustaches

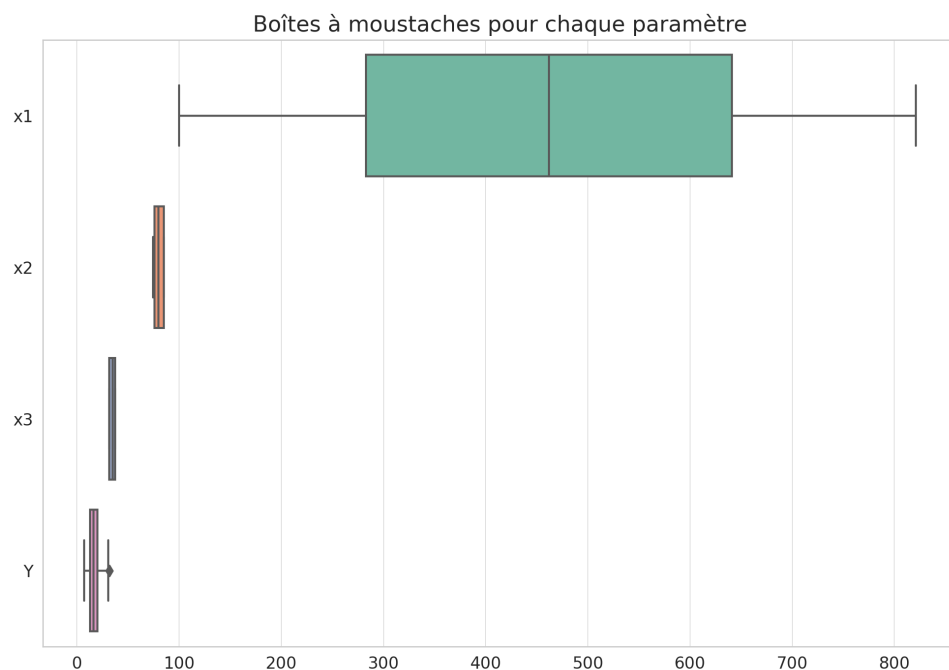


Figure 1: Boîte à moustaches pour chaque paramètre.

La boîte à moustaches permet de visualiser la distribution, la médiane, et les éventuelles valeurs aberrantes de chaque paramètre. Elle donne un aperçu rapide de la répartition des données pour chaque variable.

2.2 Matrice de corrélation

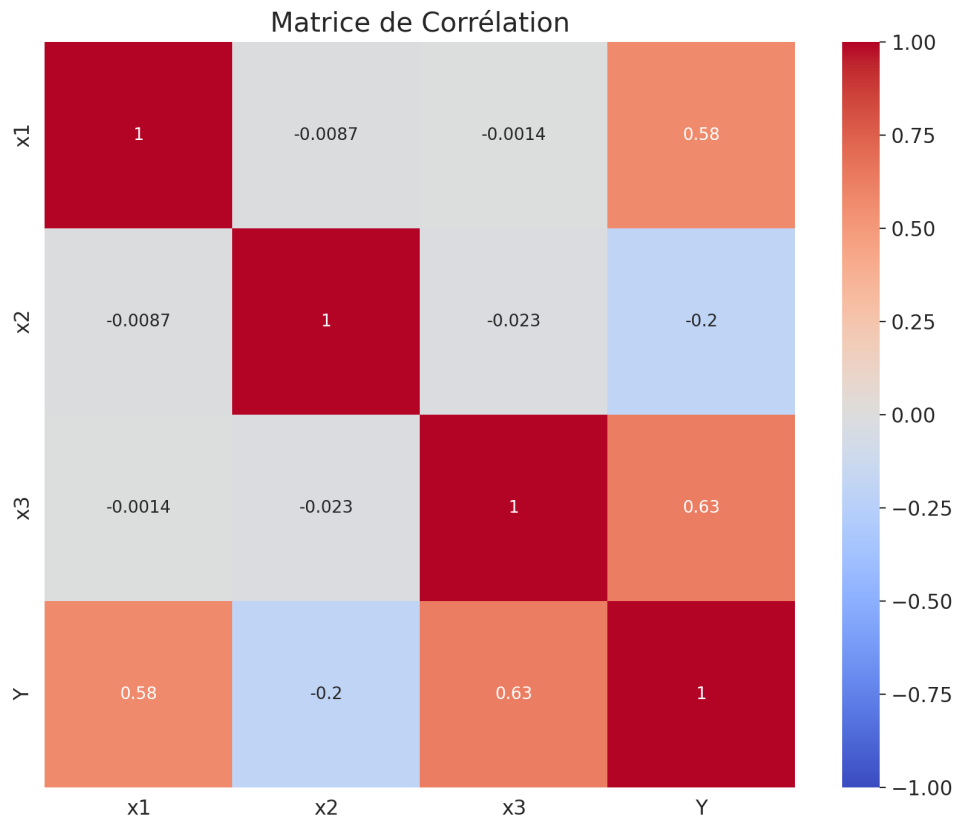


Figure 2: Matrice de corrélation entre les variables.

La matrice de corrélation en Figure 2 détaille les coefficients de corrélation entre chaque paire de variables. Une analyse attentive révèle des coefficients de l'ordre de 10^{-3} entre x_1 et x_2 , ainsi qu'entre x_1 et x_3 , et de l'ordre de 10^{-2} entre x_2 et x_3 . Ces valeurs relativement faibles traduisent une faible corrélation entre ces variables explicatives, indiquant ainsi leur indépendance. Cette indépendance est favorable car elle suggère que chaque variable apporte une information unique au modèle. De plus, une corrélation notable, de l'ordre de 10^{-1} , est observée entre la variable cible y et les variables explicatives x_i , témoignant de l'influence significative de ces dernières sur la variable dépendante.

3 Développement du modèle

Le modèle utilisé est un réseau de neurones avec la configuration suivante :

- Couche d'entrée avec 3 neurones (correspondant à x_1 , x_2 , et x_3).
- Une première couche cachée avec 64 neurones et une fonction d'activation ReLU.
- Une deuxième couche cachée avec 32 neurones et une fonction d'activation ReLU.
- Couche de sortie avec 1 neurone.

Le modèle a été compilé avec l'optimiseur Adam et a utilisé la Mean Squared Error comme fonction de perte.

3.1 Valeur Calculée vs Valeur Expérimentale

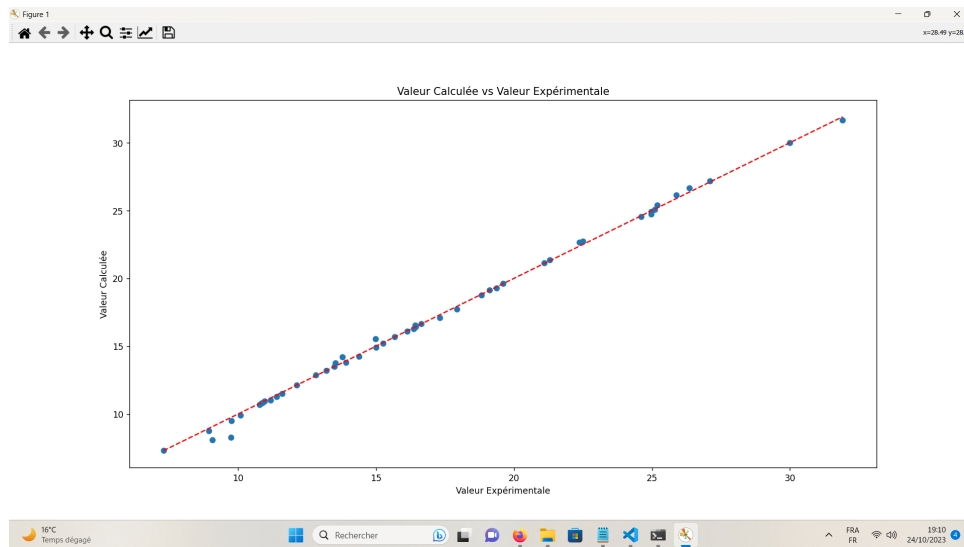


Figure 3: Comparaison entre les valeurs expérimentales et les prédictions du modèle.

La figure ci-dessus montre une forte concordance entre les prédictions du modèle et les valeurs réelles, ce qui témoigne de la précision du modèle.

4 Distribution des Résidus

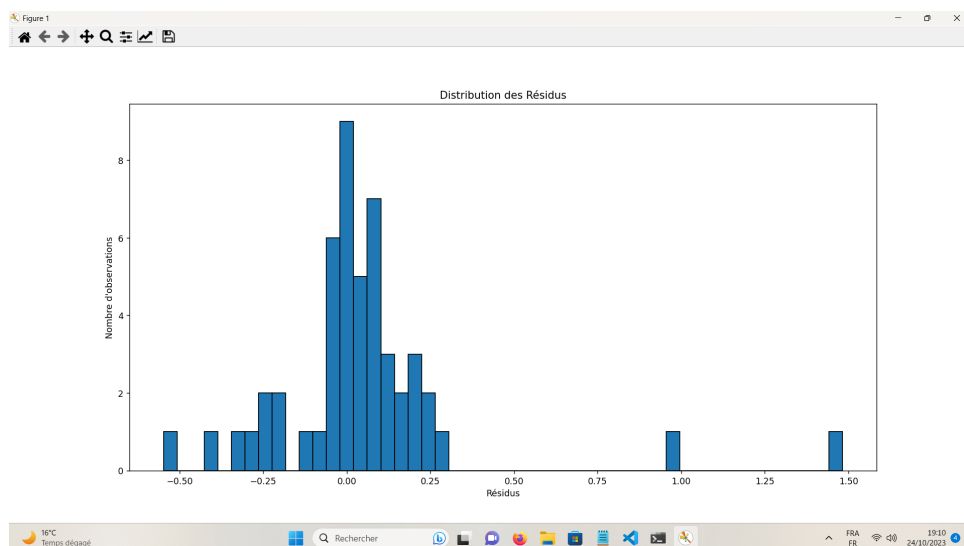


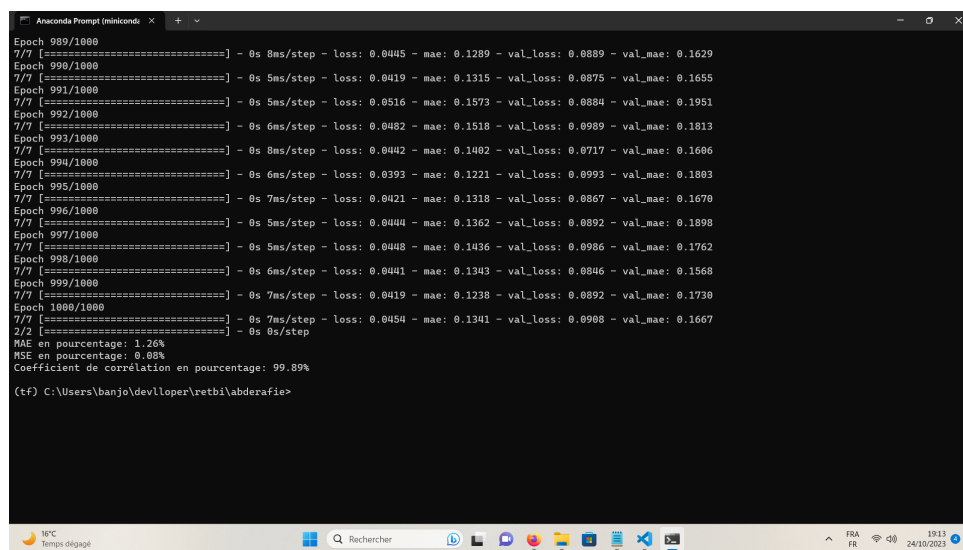
Figure 4: Distribution des résidus du modèle.

Comme le montre le graphique la distribution des résidus suivent plus ou moins une loi normale ce qui suggère une distribution aléatoire non biaisée... C'est un bon signe de performance de modèle !

5 Métriques d'évaluation

Les performances du modèle ont été évaluées en utilisant plusieurs métriques :

- **MAE en pourcentage:** 1,26%. Cela signifie que l'erreur absolue moyenne entre les prédictions et les valeurs réelles est de seulement 1,26%.
- **MSE en pourcentage:** 0,08%. Cette faible valeur indique que les prédictions du modèle sont généralement très proches des valeurs réelles.
- **Coefficient de corrélation:** 99,89%. Un tel coefficient élevé confirme l'excellente concordance entre les prédictions du modèle et les valeurs réelles.



```
Epoch 989/1000 [=====] - 0s 8ms/step - loss: 0.0445 - mae: 0.1289 - val_loss: 0.0889 - val_mae: 0.1629
Epoch 990/1000 [=====] - 0s 5ms/step - loss: 0.0419 - mae: 0.1315 - val_loss: 0.0875 - val_mae: 0.1655
7/7 [=====] - 0s 5ms/step - loss: 0.0516 - mae: 0.1573 - val_loss: 0.0884 - val_mae: 0.1951
Epoch 991/1000 [=====] - 0s 6ms/step - loss: 0.0482 - mae: 0.1518 - val_loss: 0.0989 - val_mae: 0.1813
Epoch 992/1000 [=====] - 0s 8ms/step - loss: 0.0442 - mae: 0.1402 - val_loss: 0.0717 - val_mae: 0.1606
Epoch 994/1000 [=====] - 0s 6ms/step - loss: 0.0393 - mae: 0.1221 - val_loss: 0.0993 - val_mae: 0.1803
7/7 [=====] - 0s 7ms/step - loss: 0.0421 - mae: 0.1318 - val_loss: 0.0867 - val_mae: 0.1670
Epoch 996/1000 [=====] - 0s 5ms/step - loss: 0.0444 - mae: 0.1362 - val_loss: 0.0892 - val_mae: 0.1898
Epoch 997/1000 [=====] - 0s 5ms/step - loss: 0.0448 - mae: 0.1436 - val_loss: 0.0986 - val_mae: 0.1762
Epoch 998/1000 [=====] - 0s 6ms/step - loss: 0.0441 - mae: 0.1343 - val_loss: 0.0846 - val_mae: 0.1568
7/7 [=====] - 0s 7ms/step - loss: 0.0419 - mae: 0.1238 - val_loss: 0.0892 - val_mae: 0.1730
Epoch 1000/1000 [=====] - 0s 7ms/step - loss: 0.0454 - mae: 0.1341 - val_loss: 0.0908 - val_mae: 0.1667
2/2 [=====] - 0s 0s/step
MAE en pourcentage: 1.26%
MSE en pourcentage: 0.08%
Coefficient de corrélation en pourcentage: 99.89%
(tf) C:\Users\banjo\dev\loper\retri\abderafie>
```

Figure 5: Résultats de l'entraînement

6 Conclusion

L'analyse initiale des données a révélé des informations essentielles sur la distribution des variables et leurs interrelations. Le modèle de réseau de neurones développé a démontré une capacité impressionnante à prédire la variable cible à partir des variables d'entrée. Les métriques d'évaluation confirment la précision et la fiabilité du modèle. Ces résultats mettent en évidence l'efficacité des techniques d'apprentissage automatique dans la modélisation de données complexes.

7 Annexe

7.1 Code d'entraînement

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow import keras
from scipy.stats import pearsonr

data = pd.read_excel("C:/Users/banjo/Downloads/Base AP.xls")

X = data[['x1', 'x2', 'x3']]
y = data['Y']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
→ random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(3,)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error',
→ metrics=['mae'])

model.fit(X_train_scaled, y_train, epochs=1000,
→ validation_data=(X_test_scaled, y_test), verbose=1)

model.save("mon_modele.h5")

y_pred = model.predict(X_test_scaled).flatten()

residus = y_test - y_pred

plt.figure()
plt.hist(residus, bins=50, edgecolor='k')
plt.title("Distribution des Résidus")
plt.xlabel("Résidus")
plt.ylabel("Nombre d'observations")
plt.show()

plt.figure()
plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
→ color='red', linestyle='--')

```

```

plt.title("Valeur Calculée vs Valeur Expérimentale")
plt.xlabel("Valeur Expérimentale")
plt.ylabel("Valeur Calculée")
plt.show()

mse_percentage = 100 * np.mean((residus / y_test)**2)
mae_percentage = 100 * np.mean(np.abs(residus / y_test))
correlation_coefficient, _ = pearsonr(y_test, y_pred)
correlation_percentage = 100 * correlation_coefficient

print(f"MAE en pourcentage: {mae_percentage:.2f}%")
print(f"MSE en pourcentage: {mse_percentage:.2f}%")
print(f"Coefficient de corrélation en pourcentage:
→ {correlation_percentage:.2f}%")

```