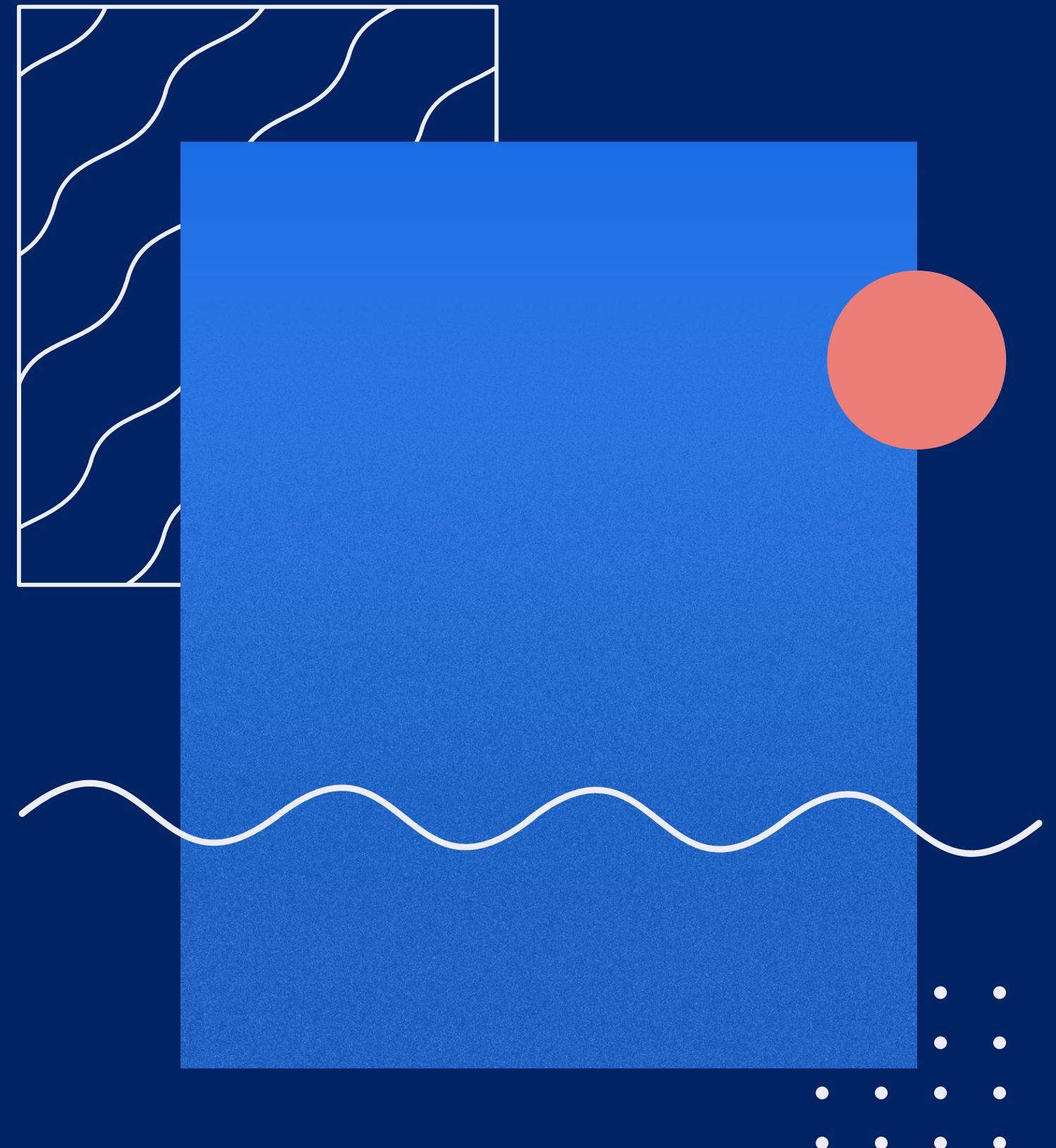


Python for data analysis

ANISSA SAYAH - AGATHE SOUBIRAN



NOTRE DATASET

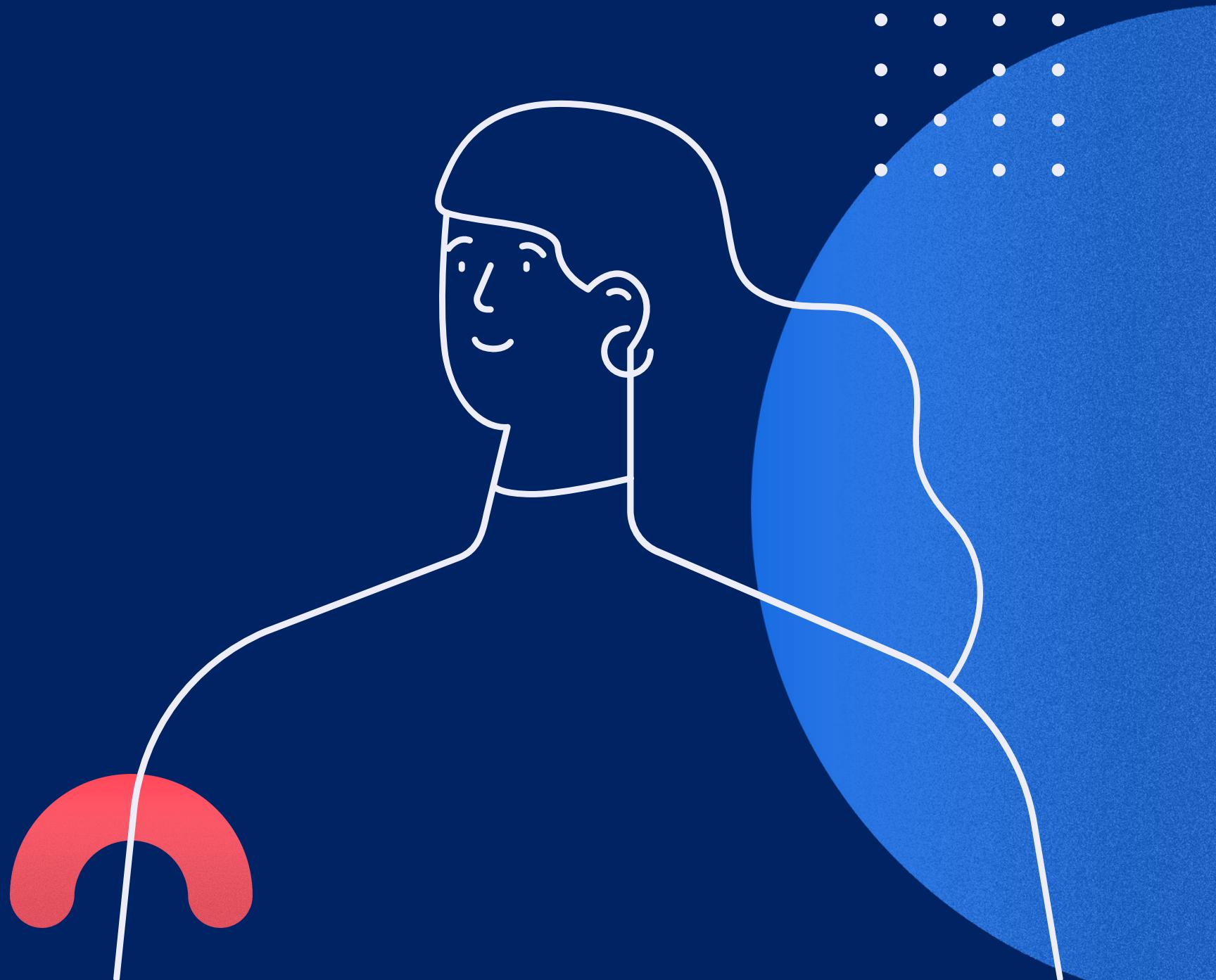
Intention d'achat des utilisateurs d'un site de e-commerce



- Organisation
 - 12 330 sessions
 - 18 classes: 10 numériques et 8 catégoriques
 - Aucune valeur manquante
- Variable à prédire
 - 'Revenue' : valeurs booléennes pour savoir si l'utilisateur a acheté ou non

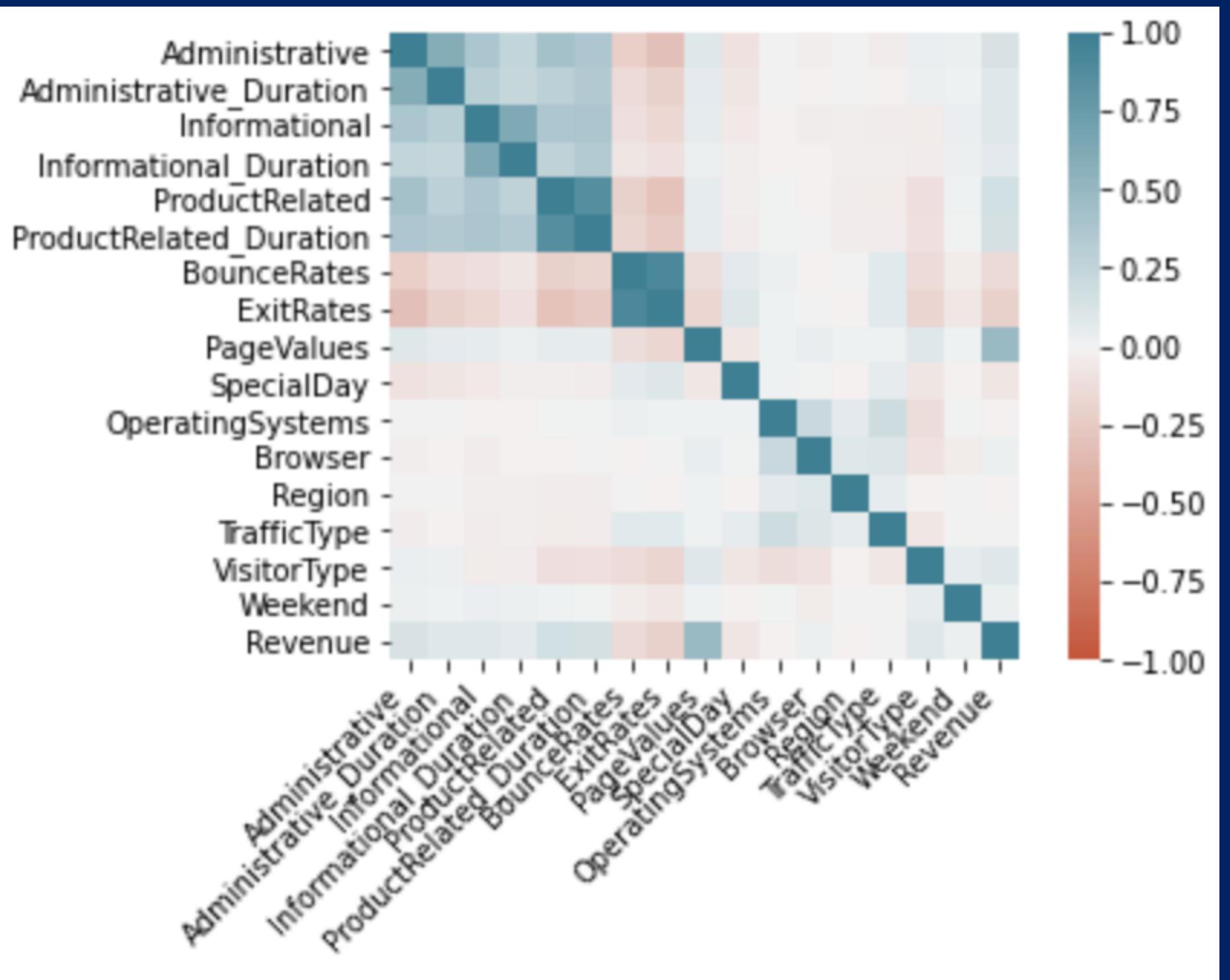
ANALYSE ET MODIFICATIONS

- Comprendre quelles valeurs influent sur 'revenue'
 - Matrice de correlation et data visualisation
- Modification des types de certaines variables en valeurs numériques
 - "Revenue": Booléen en 0 ou 1
 - "Week-end": Booléen en 0 ou 1
 - "Visitor_type" : Returning, New, Other en 0,1,2
 - "Months": Mois de 0 à 12



Data Visualisation

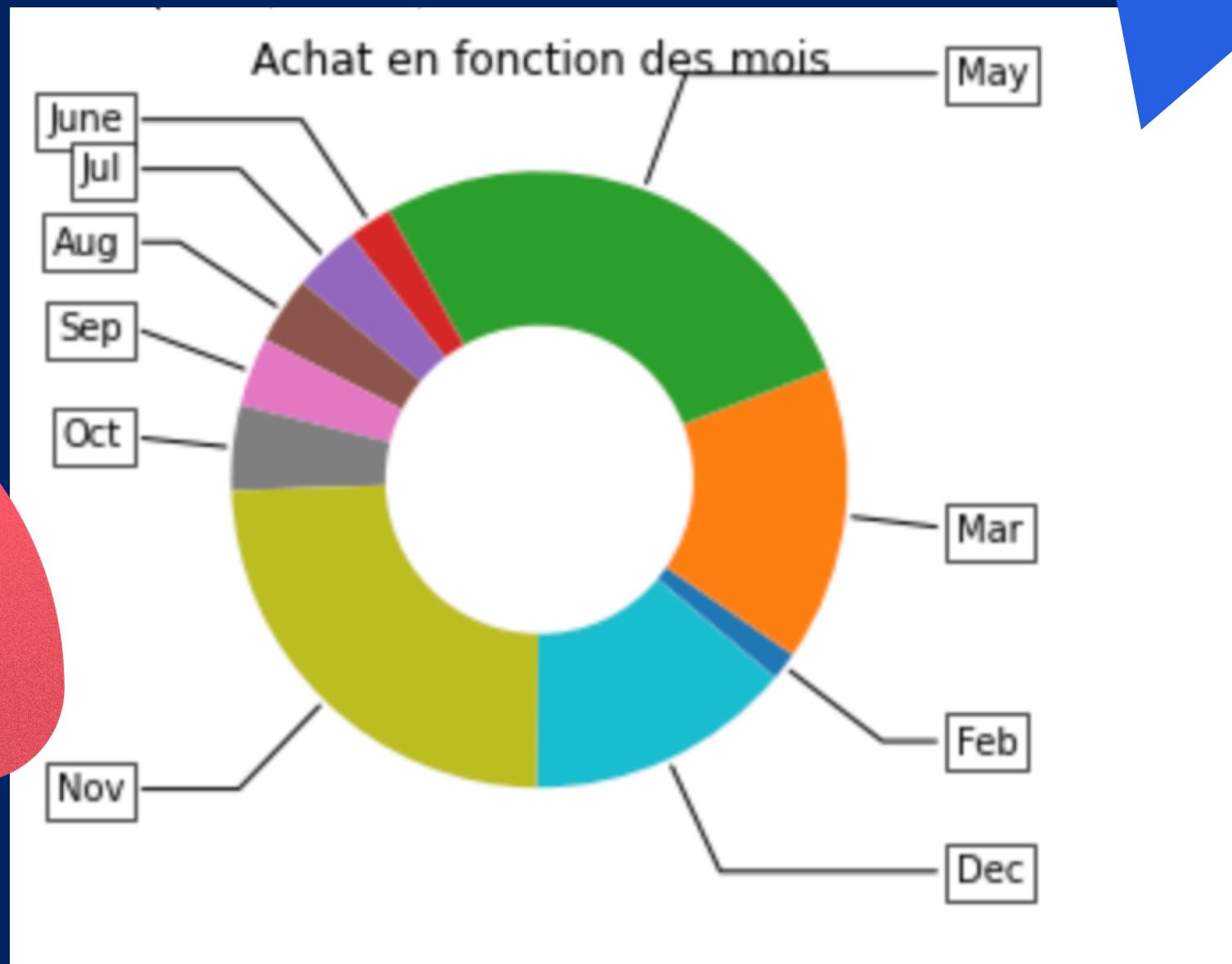




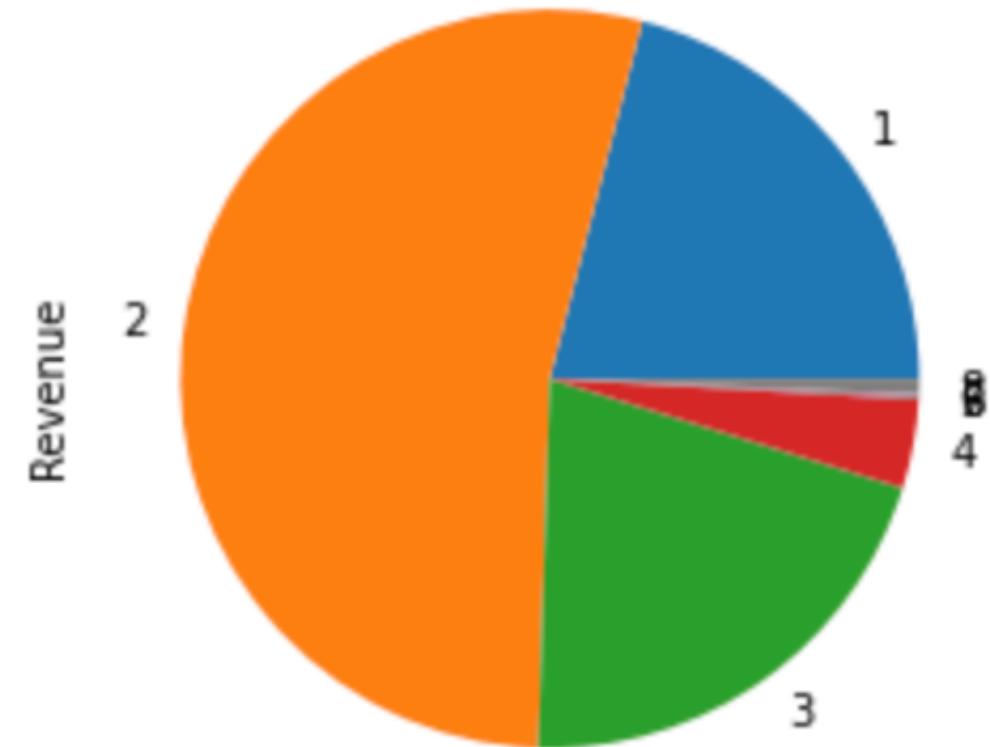
MATRICE DE CORRELATION

EXEMPLE DE DATA VISUALTION

- Achats par mois:
 - La variable 'Month' est corrélée à 'Revenue' et on remarque facilement que Mai et Novembre sont les mois les plus importants:
Peut-être car avant l'été et Noel ?



Achat en fonction du système opérateur de l'utilisateur

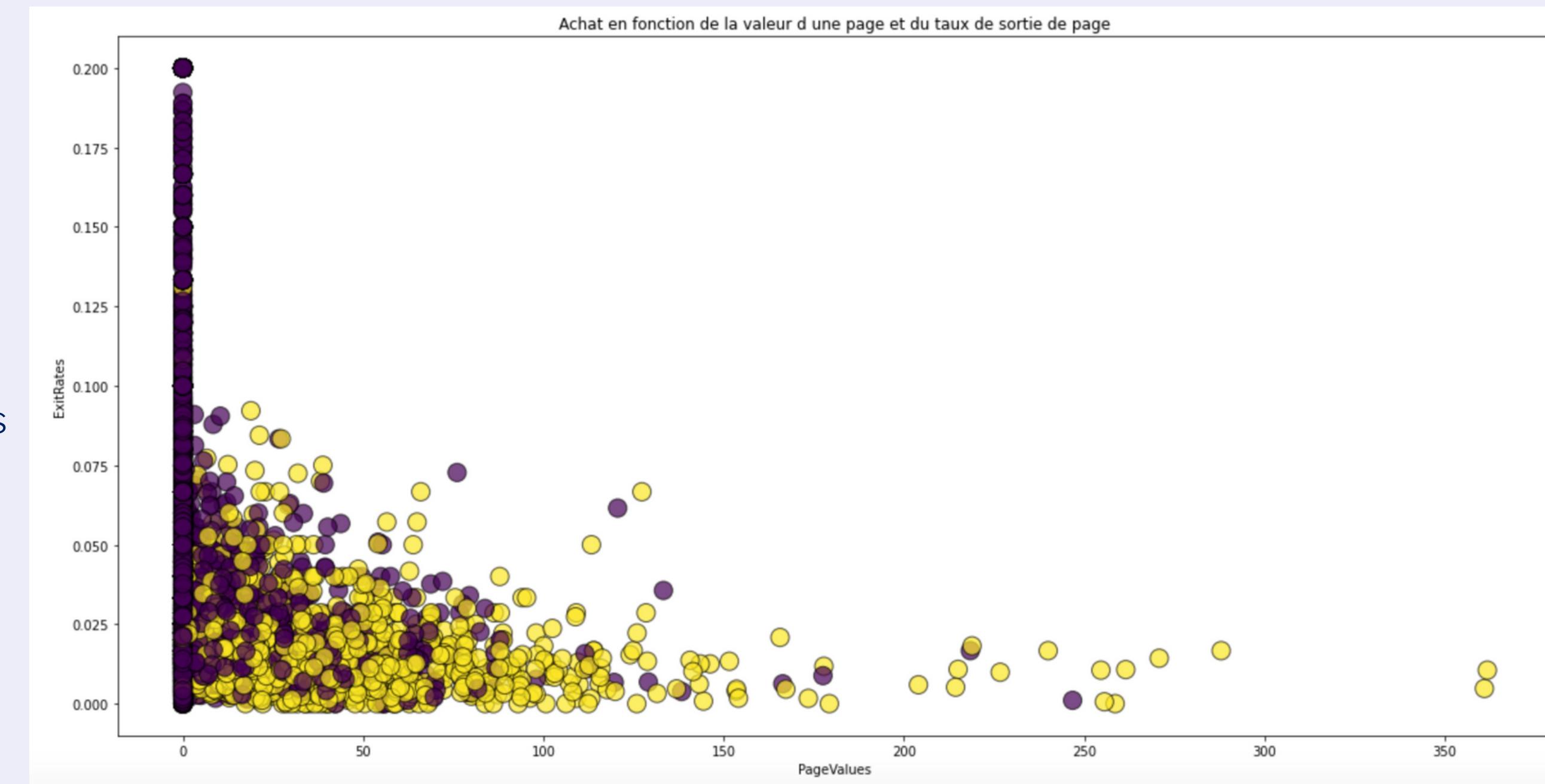


EXAMPLE DE DATA VISUALTION

- Achats par système opérateur :
 - La variable 'Operating System' est corrélée à 'Revenue'
 - l'OS 2 (pas de possibilité de savoir quel OS il représente) représente plus de la moitié du pie chart

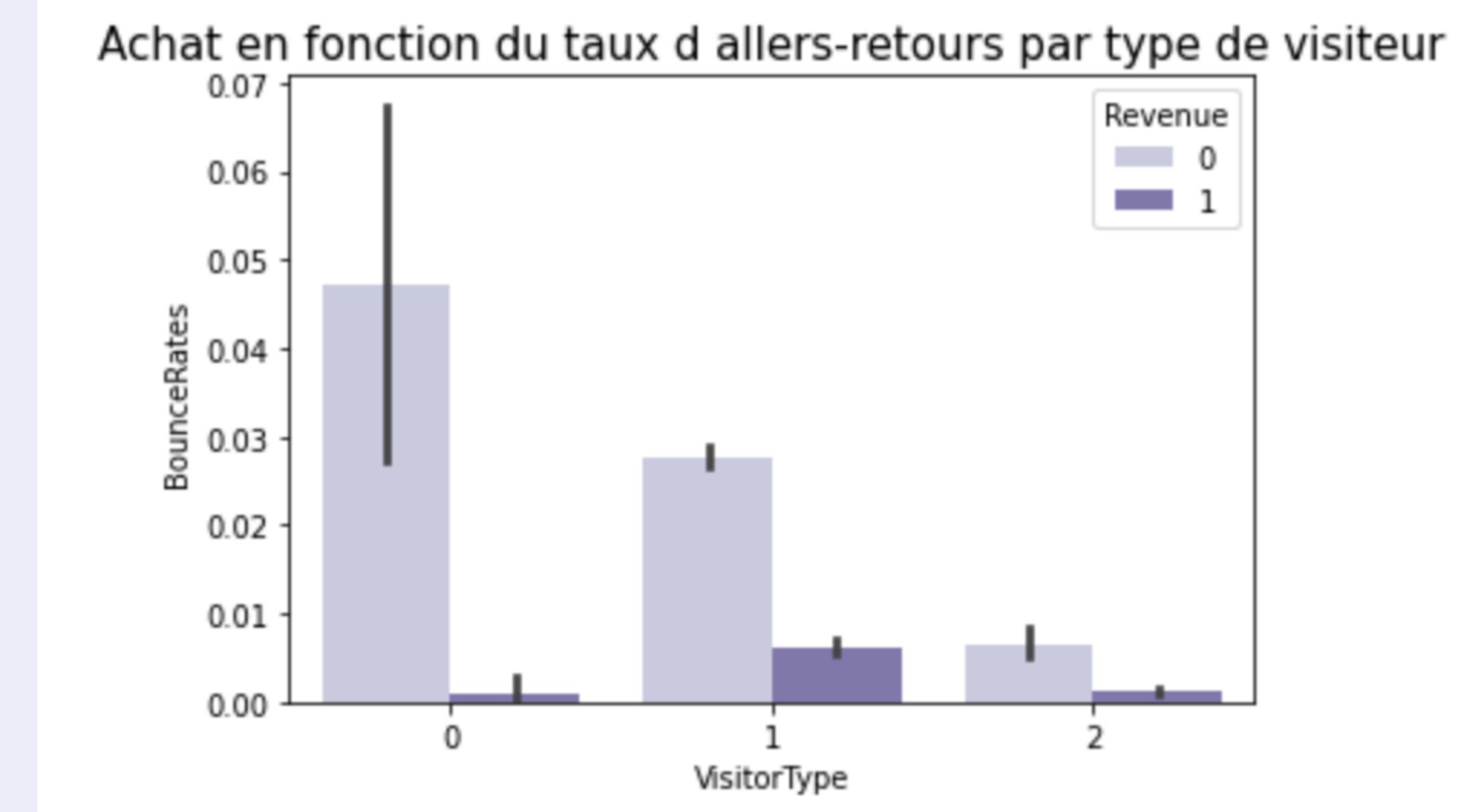
CORRELATION MULTIVARIABLES

- Achat (Jaune) ou non (Violet) en fonction de la valeur d'une page et de son taux de sortie
 - Plus la page value est basse moins les utilisateurs achètent
 - Plus l'exit rate est élevée moins ils achètent
 - Si l'exit rate est élevée et la page value faible presque aucun utilisateur n'achète
- • •

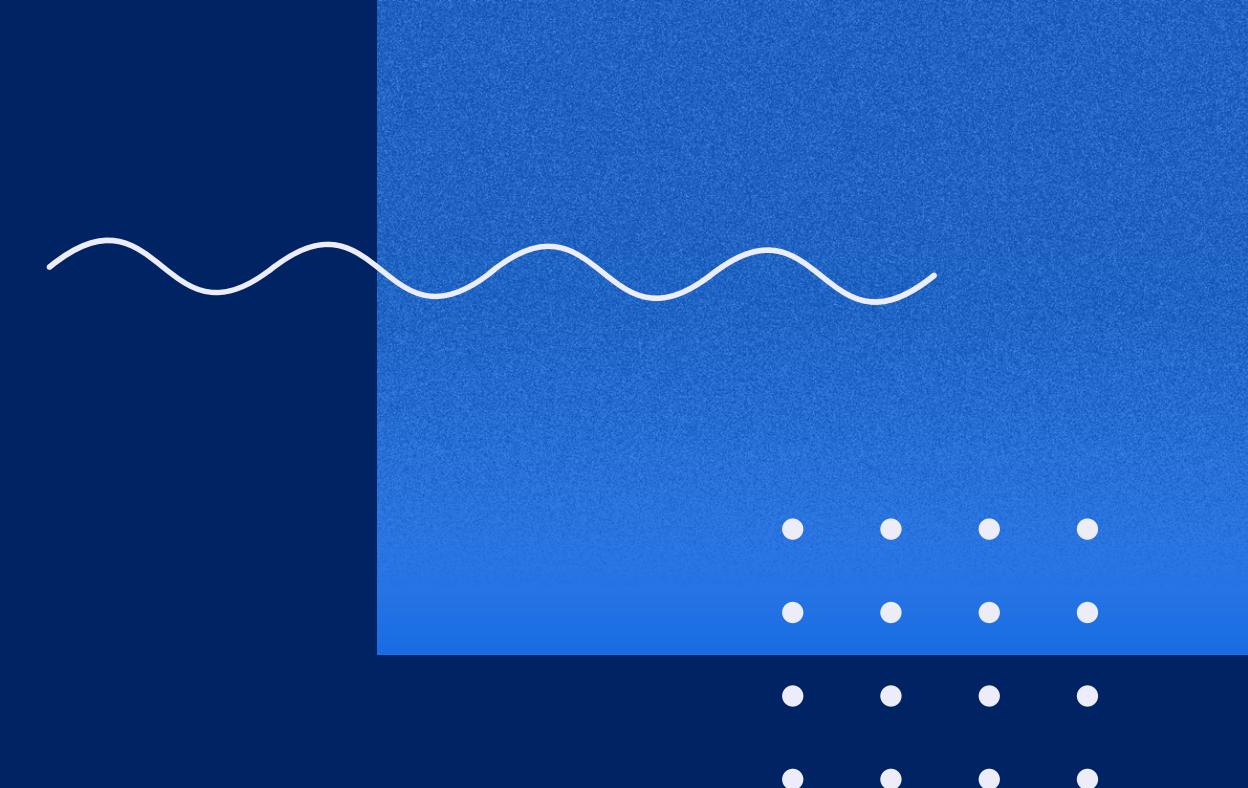
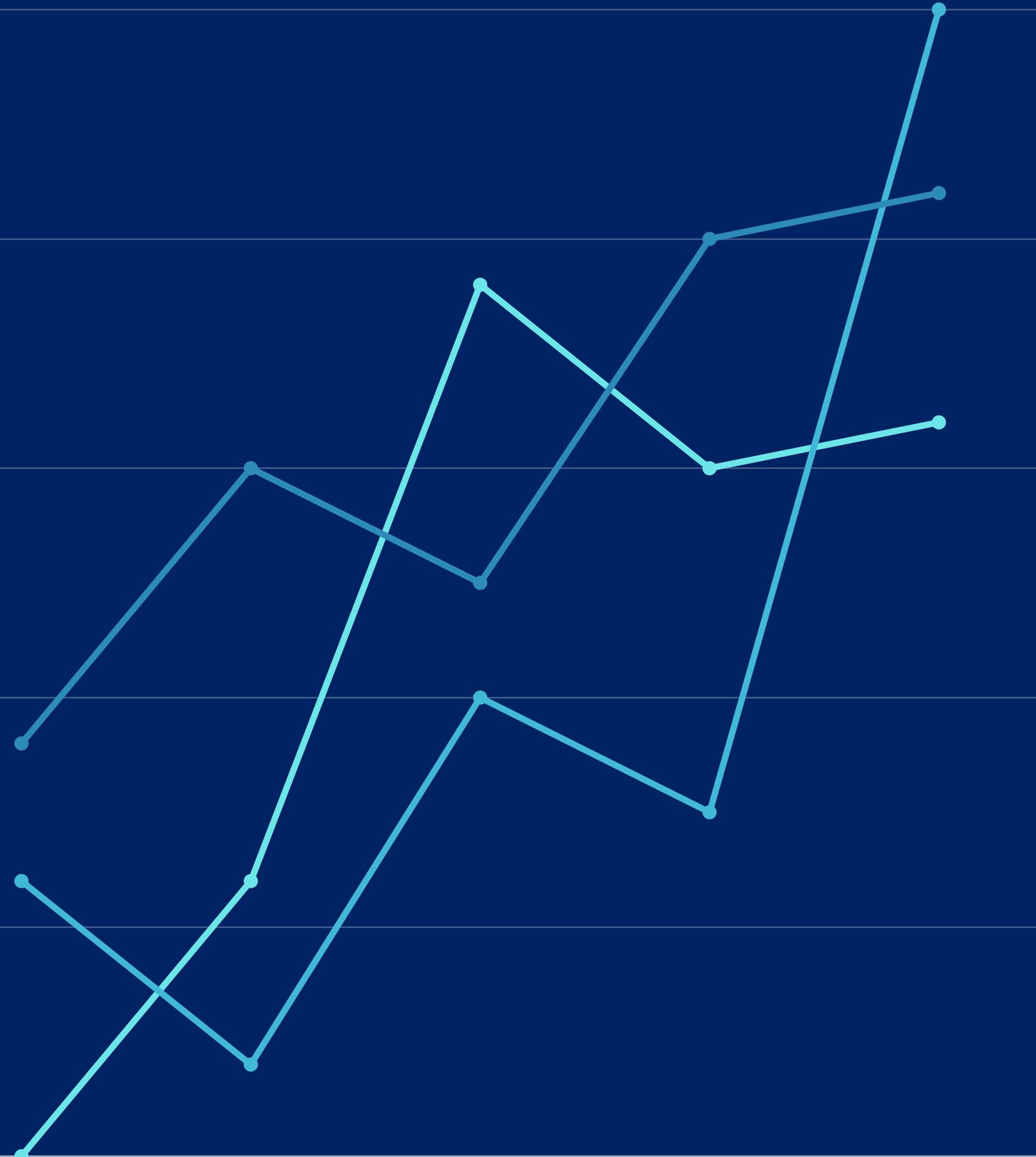


CORRELATION MULTIVARIABLES

- Achat en fonction du taux de 'Bounce' et du type de visiteur
 - Les utilisateurs qui achètent le plus sont les returning visitors lorsqu'ils restent sur une seule page
 - Les utilisateurs qui achètent le moins sont ceux qui font le plus d'aller-retours parmi les utilisateurs de type 'other'



MODÈLES DE PRÉDICTION



PRÉPARATION DES DONNÉES POUR LES MODÈLES

A partir de la data visualisation qui a été faite, nous avons jugé toutes les colonnes importantes afin de prédire si un utilisateur achète ou non.

- Entrée du modèle (X) : 'Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration', 'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend'

- Sortie (Y) : Revenue

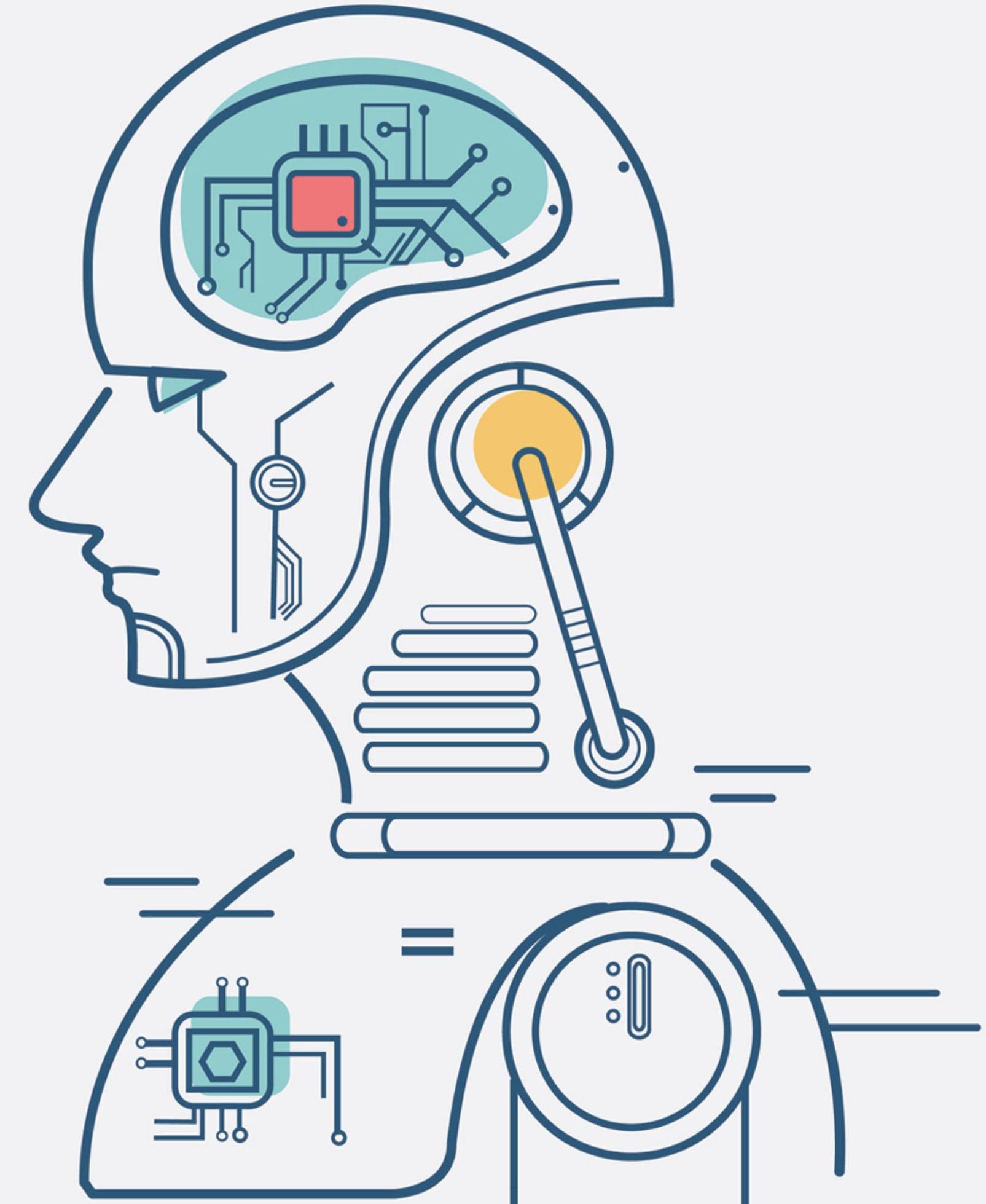
- Split:

- Testing set: 20%
- Training set: 80%

• • • •
• • • •
• • • •

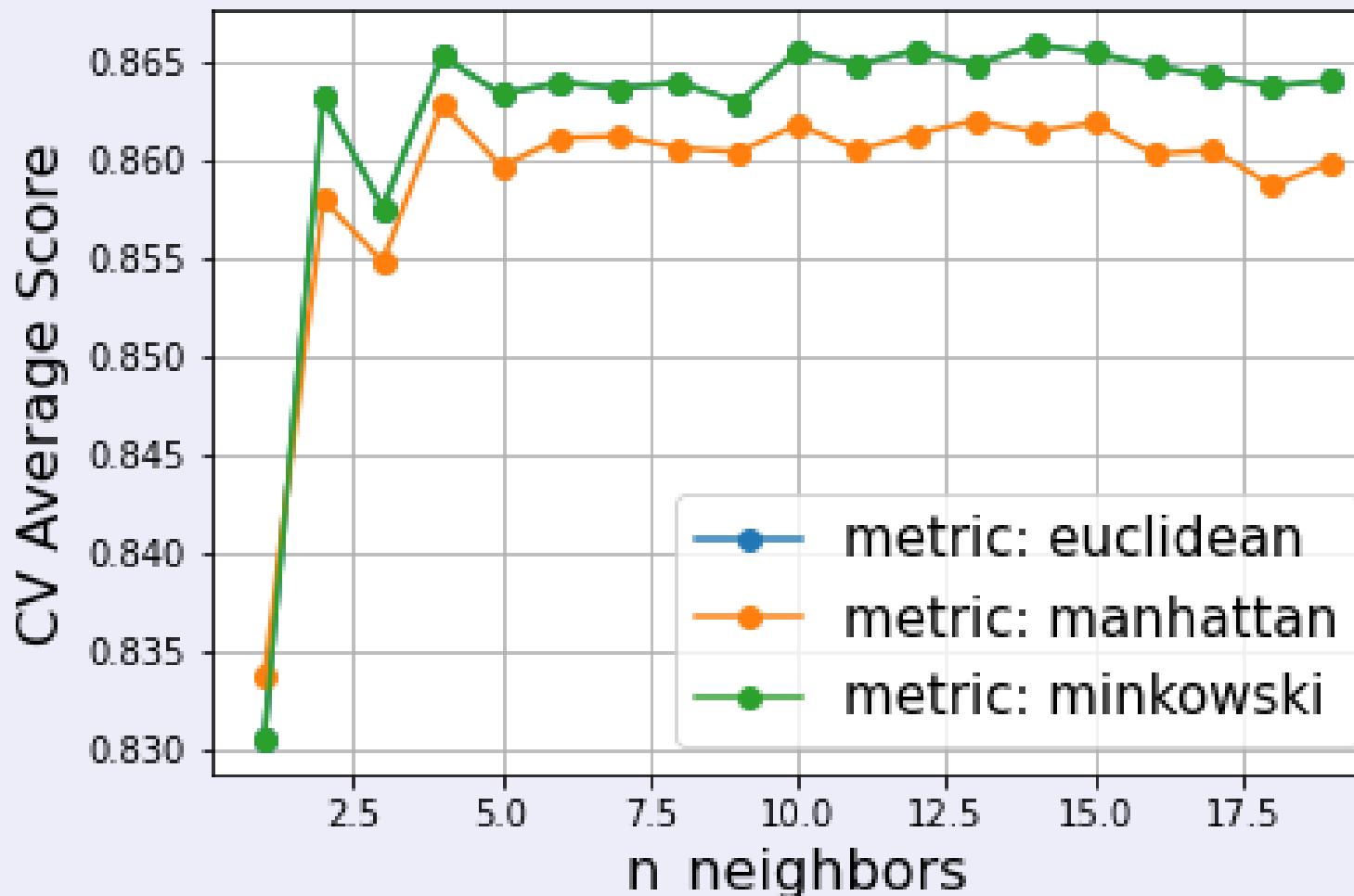
TYPE DES MODÈLES CHOISIS

- Problème de classification binaire
 - Prédir 1 --> l'utilisateur achète
 - Prédir 0 --> l'utilisateur n'achète pas
- Modèles pouvant être utilisés pour la classification binaire
 - Logistic Regression
 - k-Nearest Neighbors
 - Decision Trees
 - Support Vector Machine
 - Naive Bayes

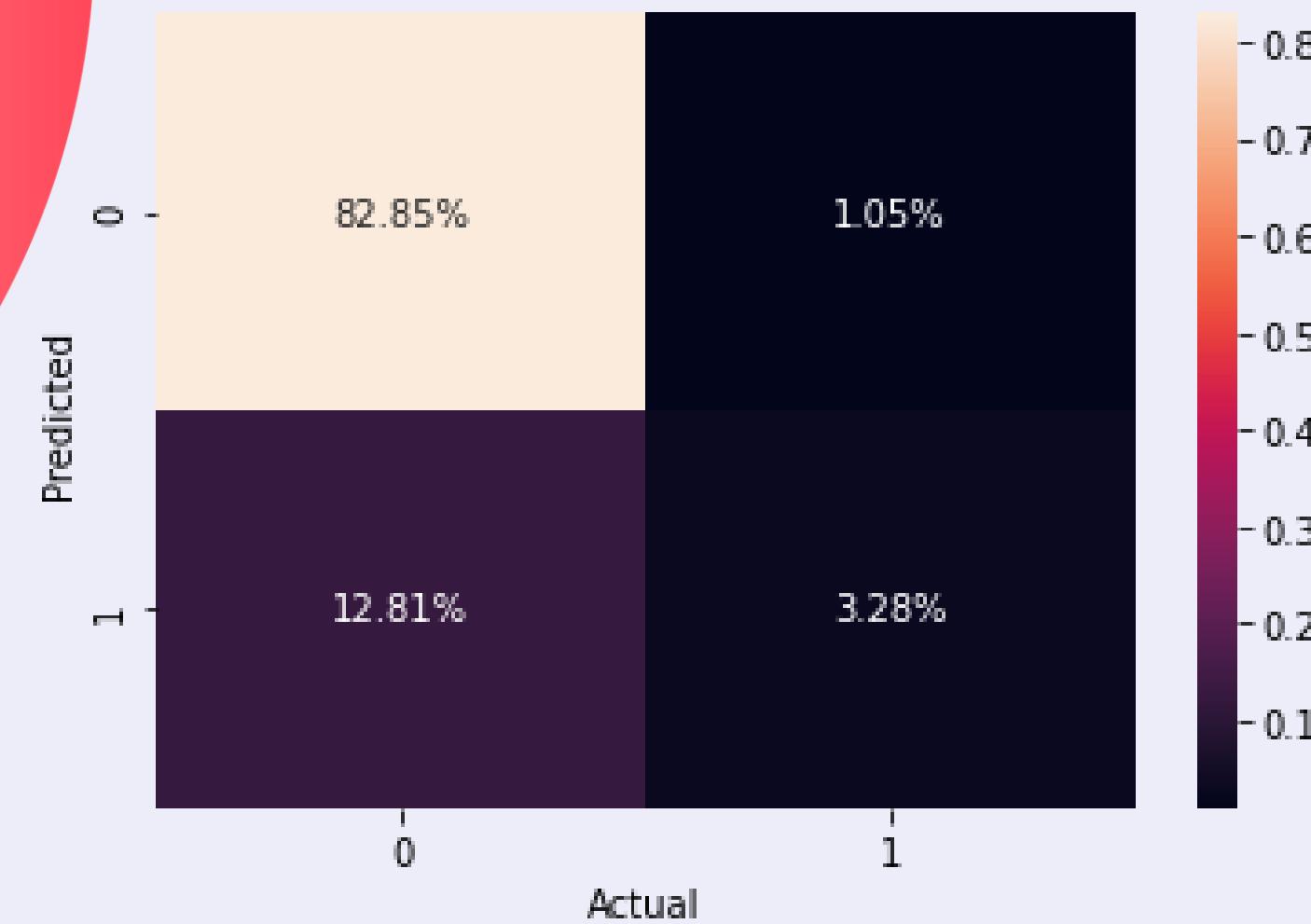


KNN

Grid Search Scores



Confusion matrix KNN

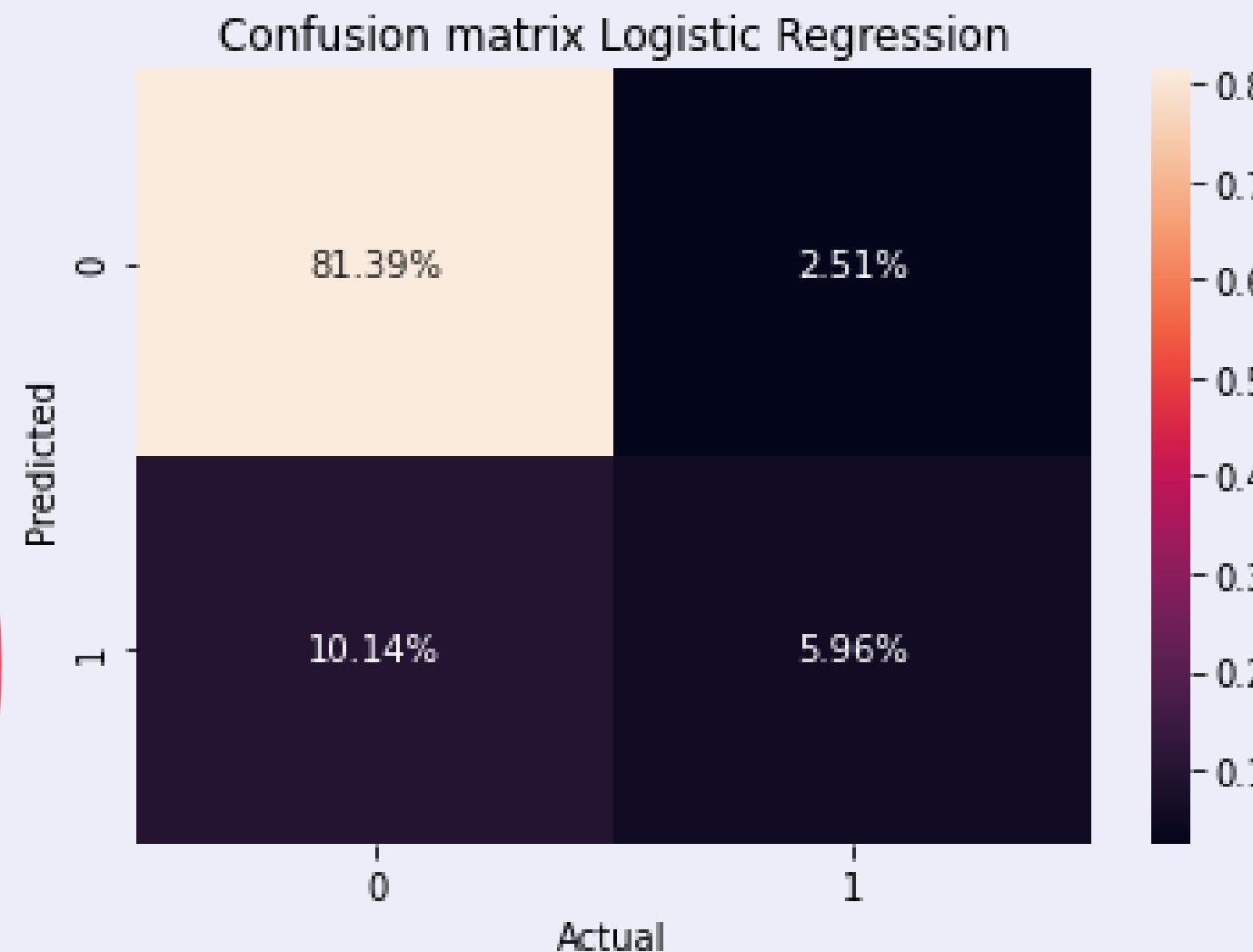


KNN très performant.
Temps d'exécution rapide



Prédit mal les 1

LOGISTIC REGRESSION



Logistic regression très performant.
Temps d'exécution rapide

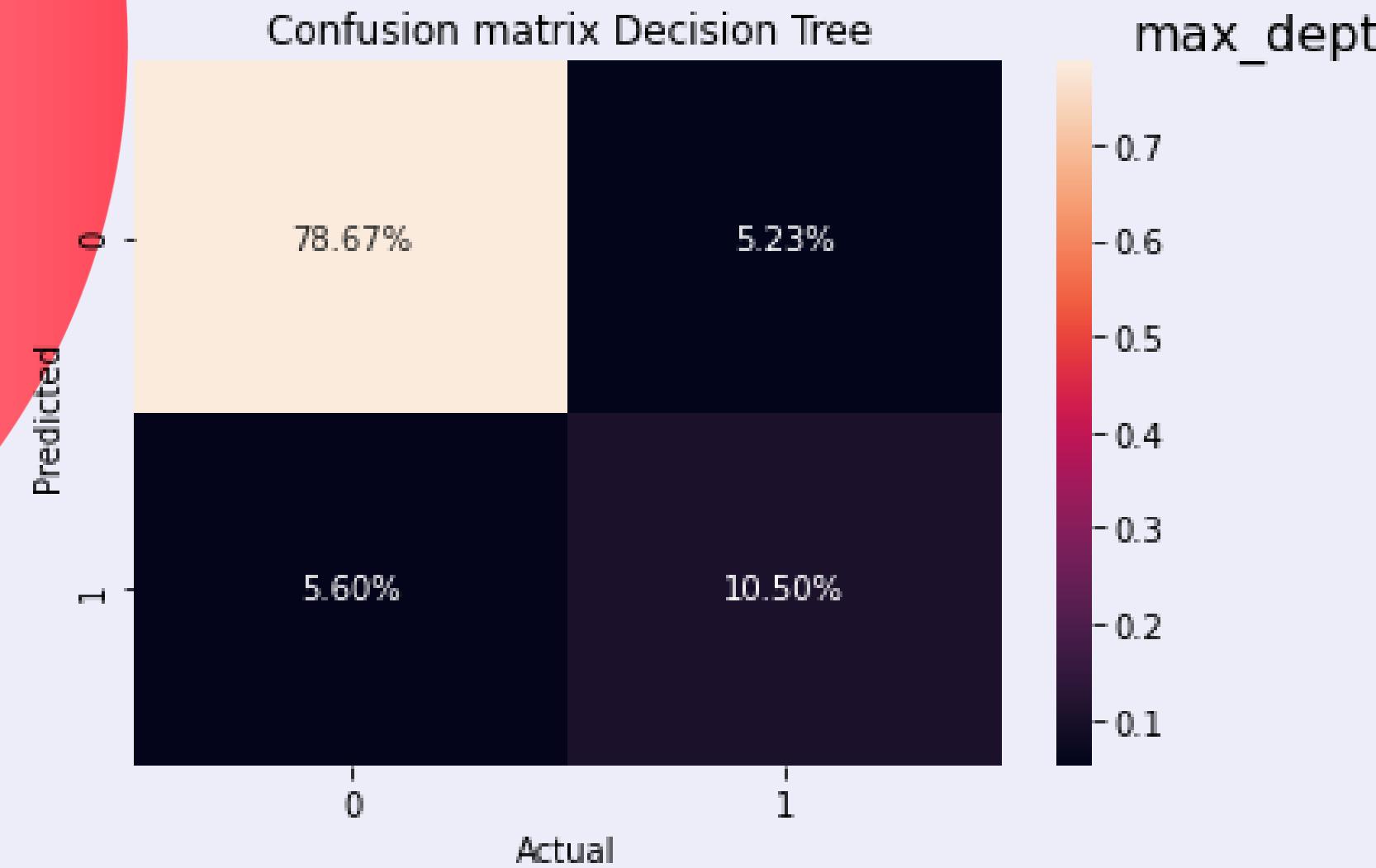
Prédit mal les 1.

```
#Affichage du best score et des meilleures paramètres testés
print("best score : ",grid.best_score_)
print("best parameters : ",grid.best_params_)

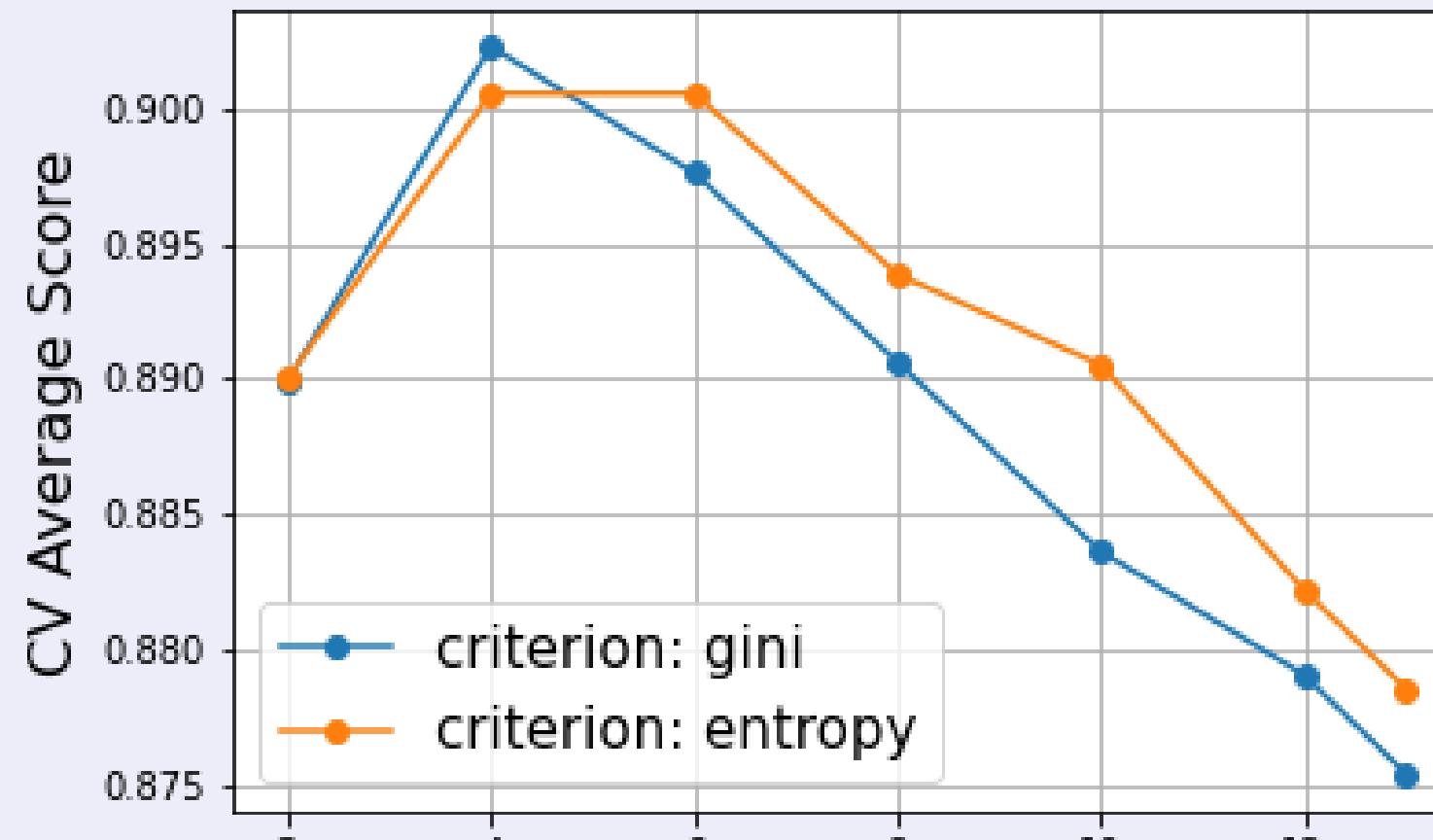
best score :  0.8826034063260341
best parameters :  {'C': 4, 'penalty': 'l2'}
```

Meilleure accuracy et hyperparamètres testés

DECISION TREE



Grid Search Scores



Decision tree très performant.
Temps d'exécution très rapide.
Prédit très bien les 1 et 0 malgré le peu d'entraînement sur les 1

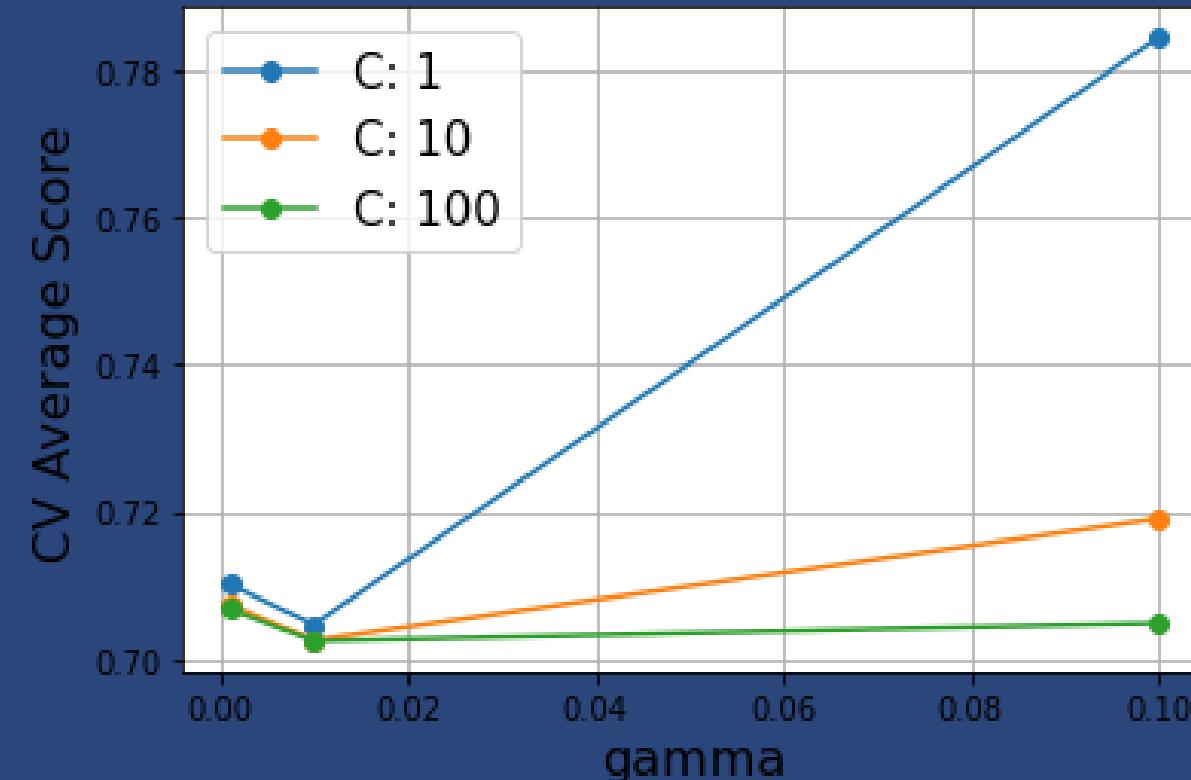
SVM kernel poly

hyperparametre C, degré et accuracy

```
C: 1  
degree: 1  
0.8402270884022709  
degree: 2  
0.8394160583941606  
degree: 3  
0.8402270884022709  
C: 10  
degree: 1  
0.8669910786699108  
degree: 2  
0.8479318734793188  
degree: 3  
0.8418491484184915  
C: 100  
degree: 1  
0.8779399837793999  
degree: 2  
0.8556366585563666  
degree: 3  
0.843065693430657
```

SVM kernel sigmoid

Grid Search Scores

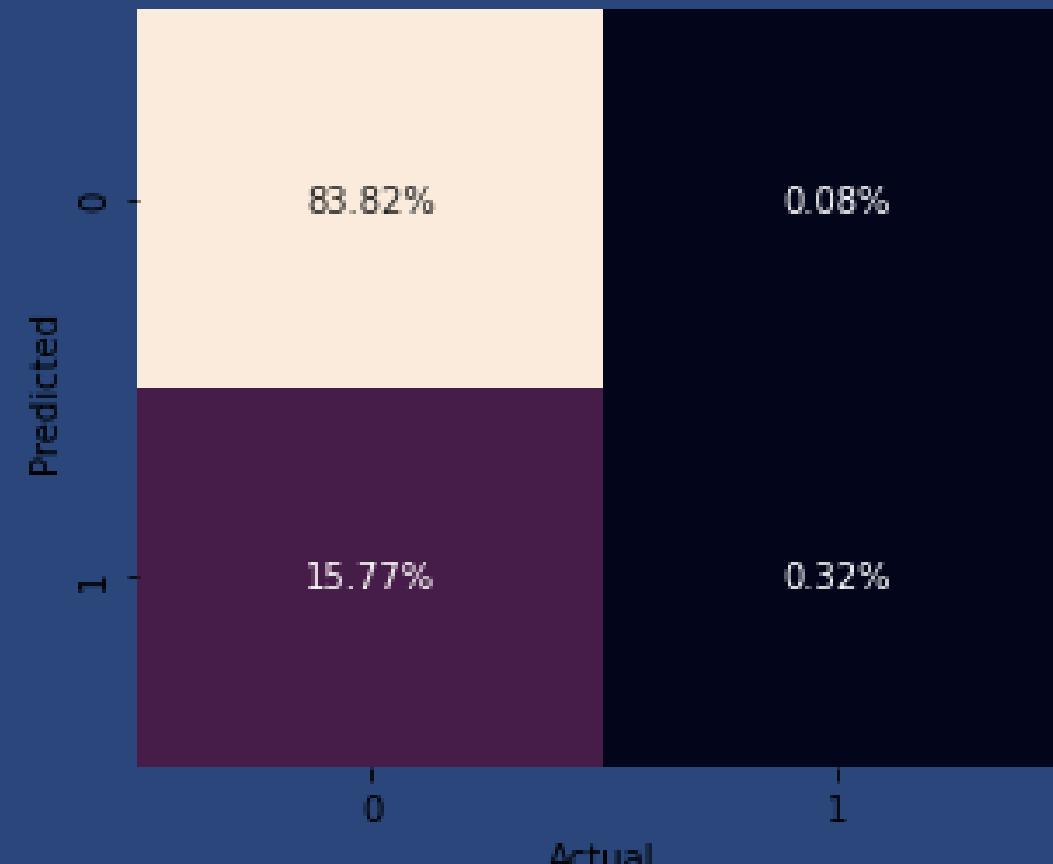


SVM kernel linear

hyperparamètre C et accuracy

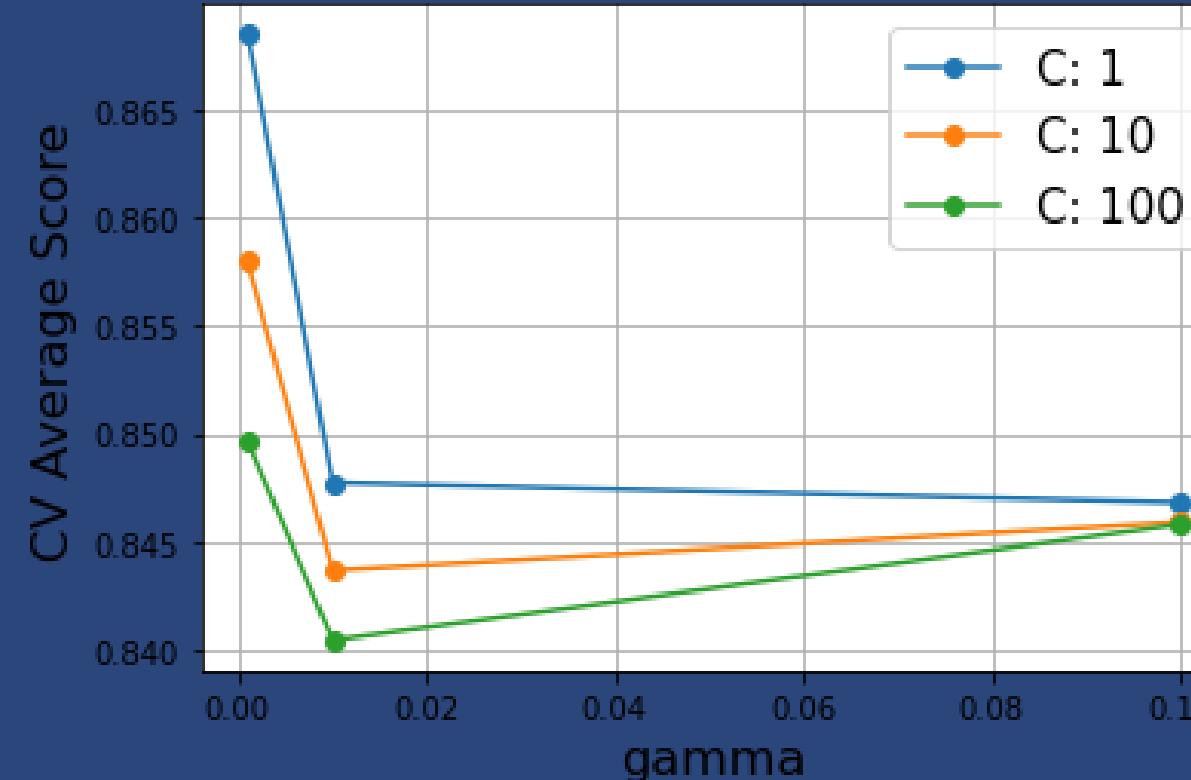
C: 1
0.8828061638280617
C: 10
0.884022708840227
C: 100
0.884022708840227

Confusion matrix SVM



SVM kernel rbf

Grid Search Scores



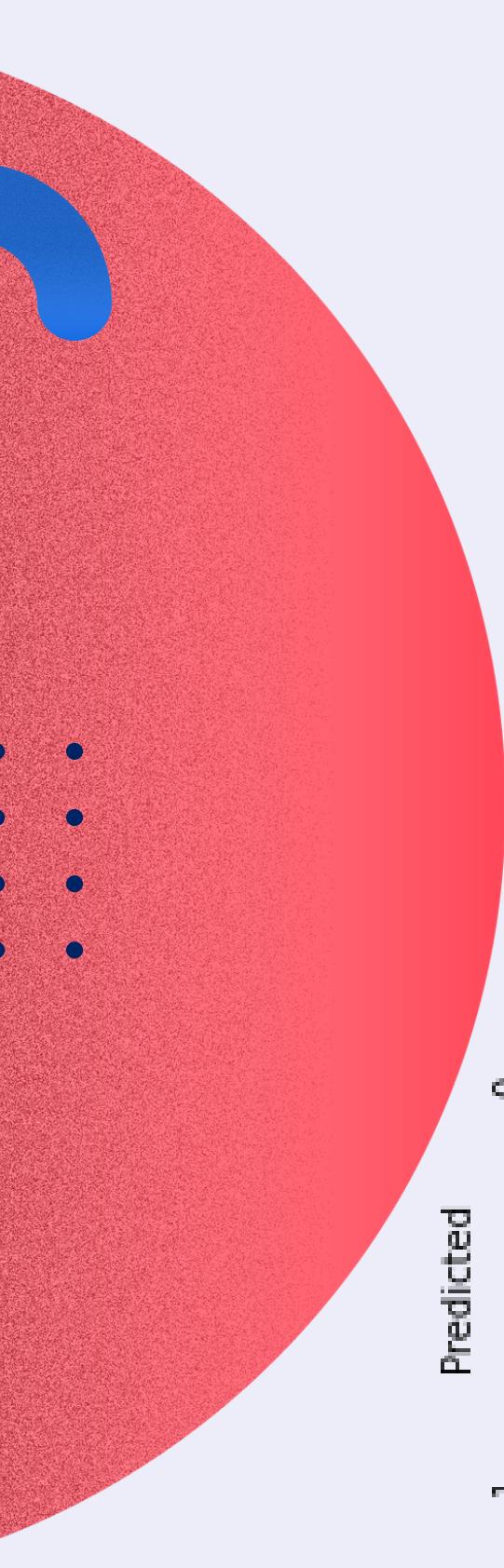
SVM



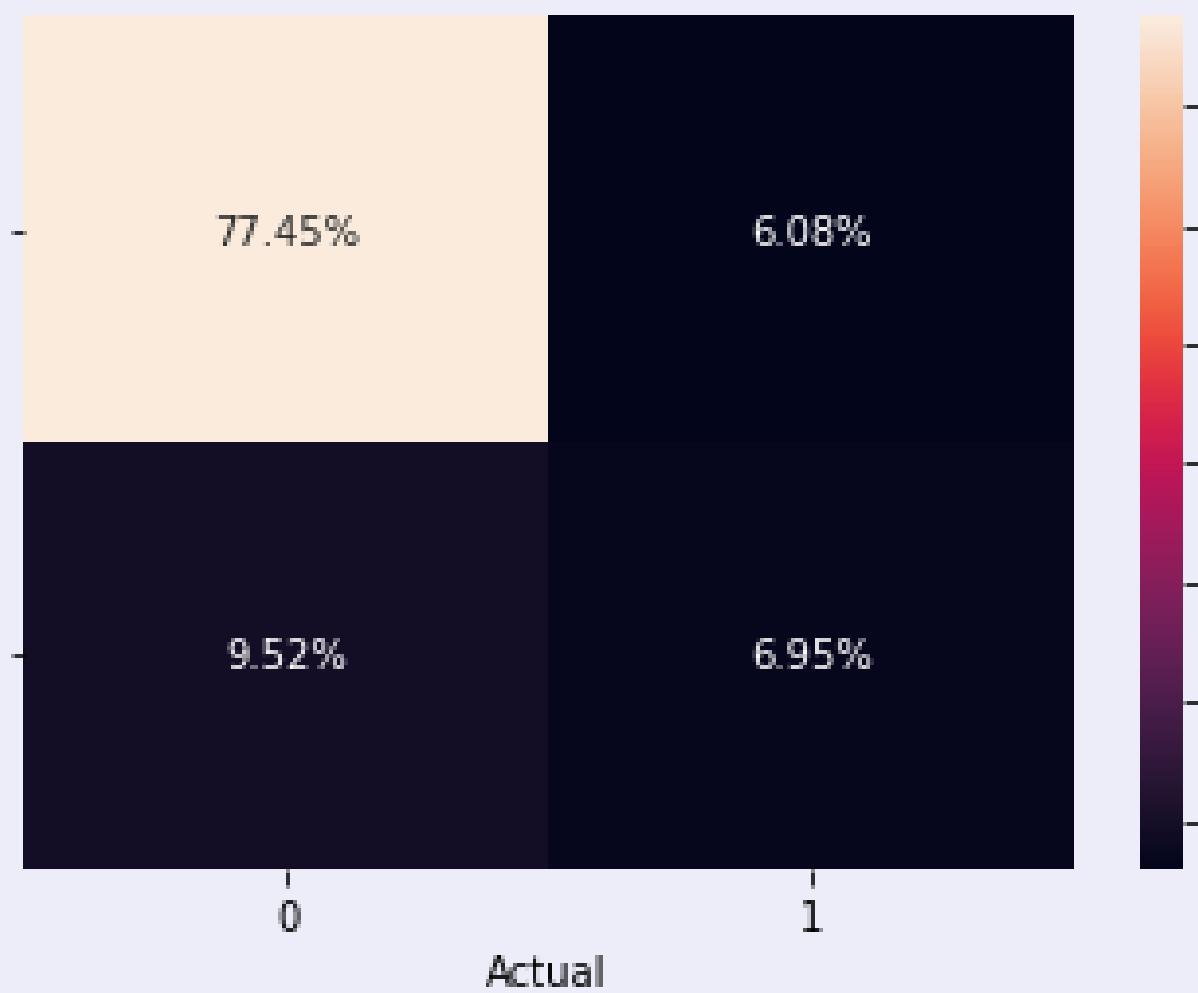
SVM très performant avec kernel linear et C =10.



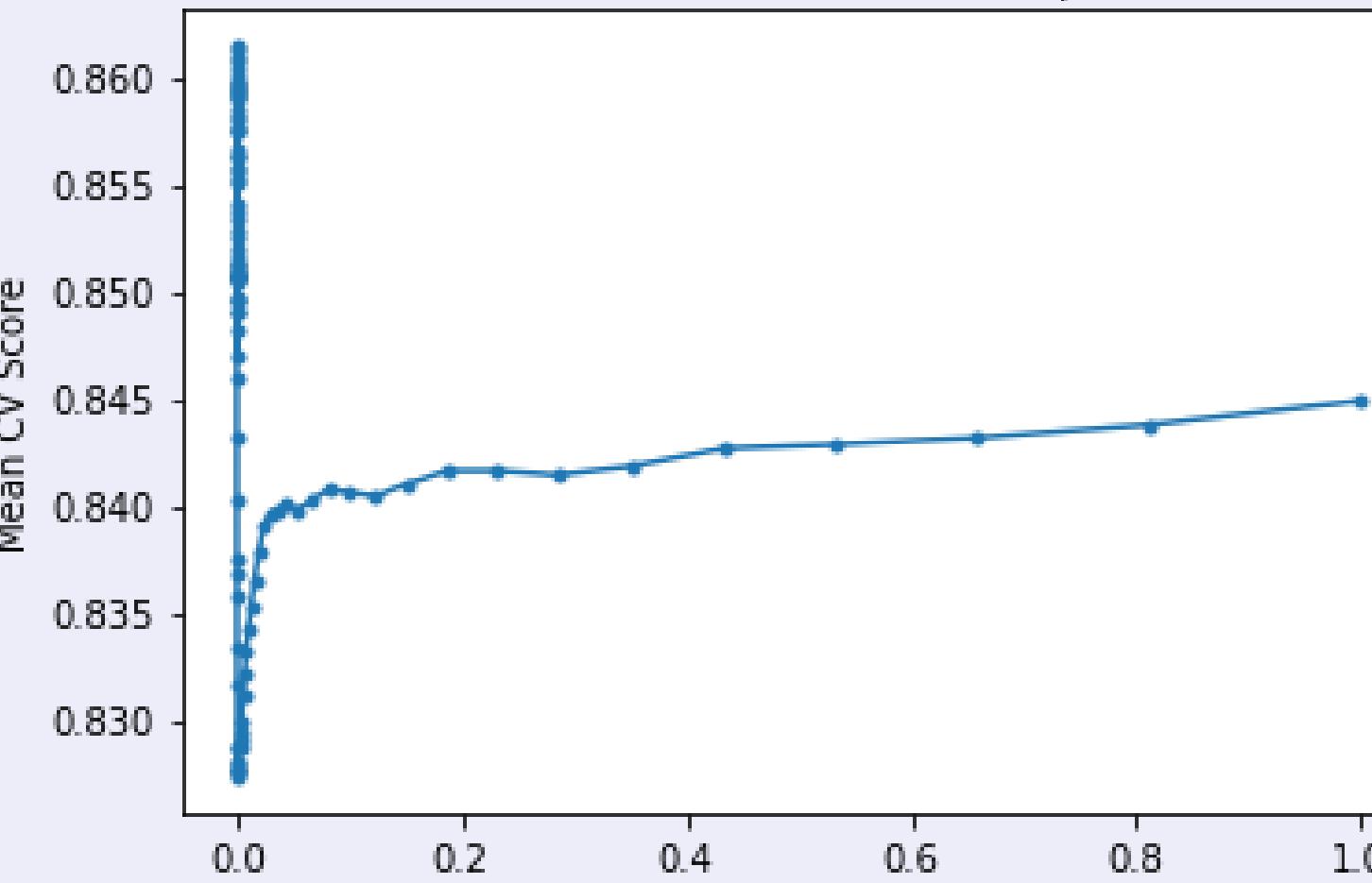
Temps d'execuition très long.
Prédit mal les 1.



Confusion matrix GaussianNB



GaussianNB Performance Comparison



GAUSSIAN NAIVE BAYES

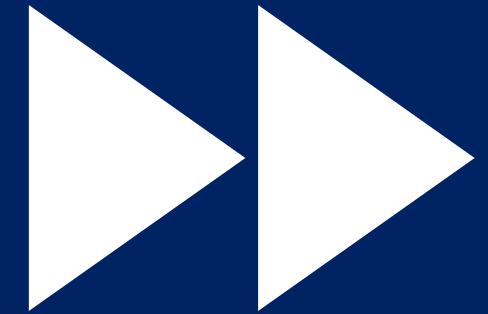


GNB très performant.
Temps d'exécution
rapide.
Prédit assez bien les 1
malgré le manque
d'entraînement sur les 1
e

MEILLEUR MODELE: DECISION TREE



- Beaucoup plus de 0 que 1 dans la df
- Equilibre, très peu d'erreur en 1 et en 0



Rapidité d'exécution::
0.03666353225708008



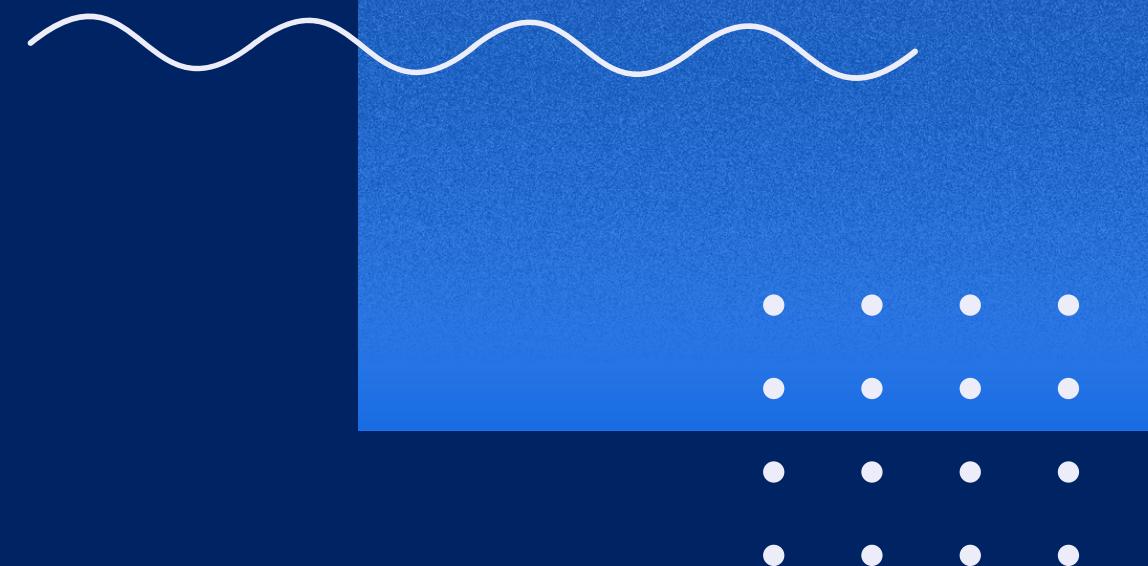
Meilleure précision:
0.8917274939172749

Remarque: Les modèles ont tous une bonne accuracy ici, mais il est très important de voir s'ils prédisent bien les 1. En effet, la plupart ont une bonne accuracy en prédisant presque que des 0 et comme il y a presque que des 0 dans la df il y a peu de chance qu'ils se trompent. Nous avons donc déterminé ce critère très important pour notre cas.



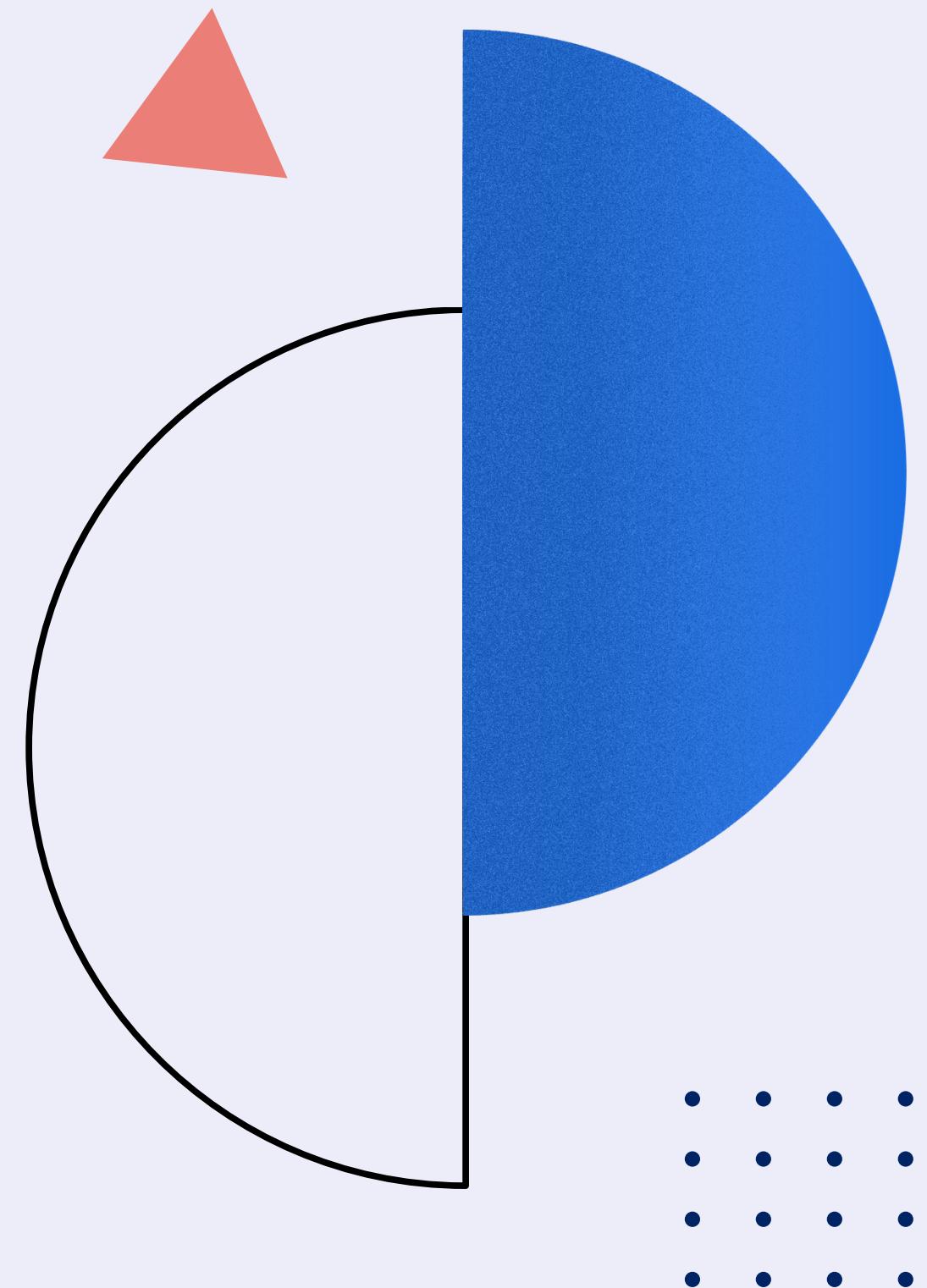
Flask

DEPLOIEMENT DU
MODELE EN API



LANCEMENT DE L'API DEPUIS VISUAL STUDIO CODE

```
PS C:\Users\Anissa\OneDrive\Bureau\ESILV\Python_data\Projet> python appPython.py
C:\Users\Anissa\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\ba
24.0. This might lead to breaking code or invalid results. Use at your own risk.
    warnings.warn(
Model loaded
Model columns loaded
* Serving Flask app "appPython" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
C:\Users\Anissa\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\ba
24.0. This might lead to breaking code or invalid results. Use at your own risk.
    warnings.warn(
Model loaded
Model columns loaded
* Debugger is active!
* Debugger PIN: 236-932-133
* Running on http://127.0.0.1:80/ (Press CTRL+C to quit)
```



TEST DE L'API SUR POSTMAN

The screenshot shows the Postman application interface for testing an API. The top bar indicates a POST request to the URL `http://127.0.0.1:80/predict`. The "Body" tab is selected, showing the raw JSON input:

```
1 [  
2   {"Administrative":0,"Administrative_Duration":0.0,"Informational":0,"Informational_Duration":0.0,"ProductRelated":1,"ProductRelated_Duration":0.0,"BounceRates":0.2,  
    "ExitRates":0.2,"PageValues":0.0,"SpecialDay":0.0,"Month":1,"OperatingSystems":1,"Browser":1,"Region":1,"TrafficType":1,"VisitorType":1,"Weekend":0},  
3   {"Administrative":0,"Administrative_Duration":0.0,"Informational":0,"Informational_Duration":0.0,"ProductRelated":2,"ProductRelated_Duration":64.0,"BounceRates":0.  
    0,"ExitRates":0.1,"PageValues":0.0,"SpecialDay":0.0,"Month":1,"OperatingSystems":2,"Browser":2,"Region":1,"TrafficType":2,"VisitorType":1,"Weekend":0}  
4 ]
```

The bottom section displays the response from the API, which is a JSON object with a single key "prediction" containing the value "[0, 0]".

Body	Cookies	Headers (4)	Test Results	Status: 200 OK	Time: 58 ms	Size: 174 B	Save Response
Pretty Raw Preview Visualize JSON ▾							
1 { 2 "prediction": "[0, 0]" 3 }							

Merci

