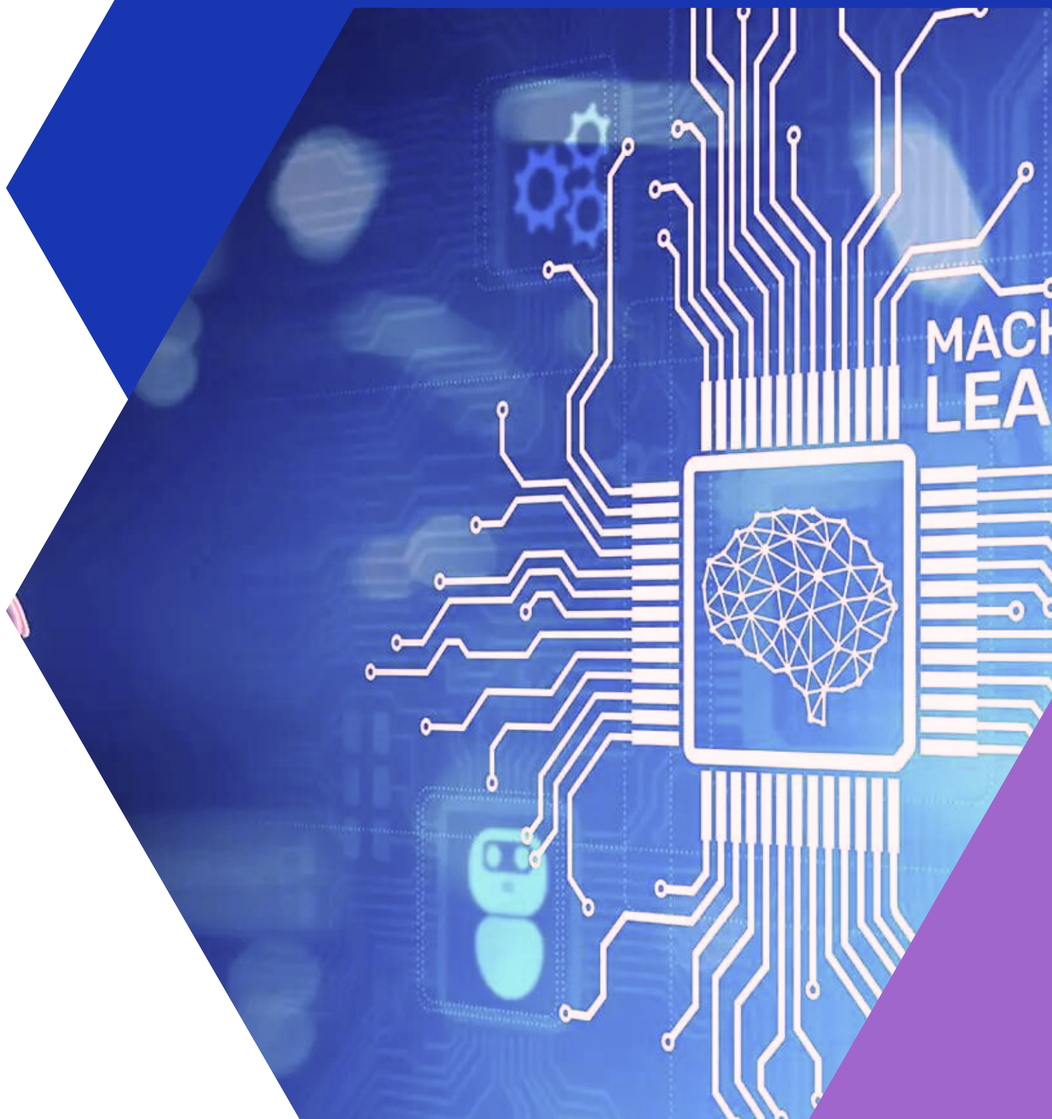
Cycle d'Ingénieurs en
Génie Informatique
Faculté des Sciences et Techniques Errachidia

PROJET MACHINE LEARNING :

OPTIMISATION DES CAMPAGNES PUBLICITAIRES DANS LE MARKETING D'AFFILIATION : UNE APPROCHE PRÉDICTIVE PAR L'APPRENTISSAGE AUTOMATIQUE



Réalisé par :

- BELQARS Oussama
- DRISSI HASSANI El Bachir
- GRICH Ali

Plan de Présentation :

- Introduction
- Problématique
- Prétraitement
- Approche Algorithmique
- Meilleur Algorithme
- Conclusion

Introduction

Dans le contexte de l'évolution rapide de la technologie, les algorithmes d'apprentissage statistique ont émergé en tant qu'outils puissants pour résoudre des problèmes complexes. Ces algorithmes, inspirés par le fonctionnement de l'intelligence humaine, permettent aux machines d'apprendre à partir de données et d'accomplir des tâches sans nécessiter une programmation explicite. Cette présentation se penchera plus en détail sur les algorithmes de classification, illustrant comment ces outils novateurs sont déployés pour résoudre des problèmes spécifiques et offrir des solutions intelligentes et adaptables.

Problematique

Une entreprise européenne de premier plan dans le domaine des réseaux d'affiliation souhaite utiliser l'apprentissage automatique pour améliorer ses taux de conversion, notamment le coût par clic (CPC). Leur réseau s'étend à plusieurs pays européens, et ils cherchent à prédire la probabilité de clic pour anticiper et optimiser leurs futures campagnes CPC.

Description de Base de données :

On dispose d’un seul fichiers : data.csv. Les variables de cet ensemble de données sont anonymisées pour des raisons de confidentialité.

- **ID : Identifiant unique**
- **datetime : Date et heure du clic**
- **siteid : Identifiant du site web**
- **offerid : Identifiant de l'offre (offres basées sur les commissions)**
- **category : Catégorie de l'offre**
- **merchant : Identifiant du vendeur**
- **countrycode : Code du pays où la portée des affiliés est présente**
- **browserid : Navigateur utilisé**
- **devid : Appareil utilisé**
- **click : Variable cible**

Description de Base de données :

On dispose d’un seul fichiers : data.csv. Les variables de cet ensemble de données sont anonymisées pour des raisons de confidentialité.

	A	B	C	D	E	F	G	H	I	J
1	ID	datetime	siteid	offerid	category	merchant	countrycoc	browserid	devid	click
2	IDO2PuntC	1/18/2017 19:08	6310005	99217	41706	4000296	c			1
3	IDdXlTWol	1/14/2017 15:40		287164	89522	76532649	b	Edge	Tablet	0
4	ID8ulr3Qi	1/13/2017 10:39	2463708	948989	12052	31388981	c	InternetExplorer		1
5	IDoW4BTl	1/15/2017 18:00		938403	36768	555603	a	Mozilla	Desktop	0
6	IDo4V6TX	1/14/2017 11:54		229153	40339	49384126	f	Edge	Tablet	0
7	IDJGNUzla	1/14/2017 9:17		306160	26162	15961717	f	InternetEx	Desktop	0
8	ID1tZkcbQ	1/16/2017 17:40	8726149	494309	97863	3156697	f	IE	Mobile	0
9	ID7kaej2z	1/14/2017 14:56	2457384	685515	25516	43770446	f	Mozilla Fir	Desktop	0
10	IDwYUAp5	1/13/2017 20:57	5034323	866225	36768	78406516	d	InternetEx	Desktop	1

Prétraitement :

Le prétraitement des données, étape cruciale en apprentissage automatique. Son rôle essentiel consiste à optimiser la qualité des données, éliminer les incohérences, et les rendre compatibles avec les modèles.

En suivant les étapes que nous avons suivies dans le prétraitement de nos données, assurant ainsi une préparation méticuleuse pour la construction d'un modèle robuste :

Prétraitement :

Etape I

Chargement des données

Nous importons les données que nous avons téléchargées dans Google Drive pour y accéder rapidement

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ]  import pandas as pd

     dataset = pd.read_csv('/content/drive/MyDrive/205e1808-6-dataset/data.csv')
     print(dataset.shape)
```

```
(45200, 10)
```

Prétraitement :

Etape 2 Conversion de la colonne datetime

Nous transformons la colonne datetime en heures et en jours numérotés pour une manipulation plus aisée

```
[ ]  dataset['datetime'] = pd.to_datetime(dataset['datetime'])

     # Extraction de l'heure et du jour de la semaine
     dataset['hour'] = dataset['datetime'].dt.hour
     dataset['day_of_week'] = dataset['datetime'].dt.dayofweek
     print(dataset.shape)
```

```
(45200, 12)
```

Prétraitement :

Etape 3

Séparation des caractéristiques (X) et de la variable cible Y

La variable que nous cherchons à prédire, permettant une analyse plus approfondie

```
X = dataset.iloc[:, [2, 3, 4, 5, 6,7,8, 10, 11]].values
y = dataset.iloc[:, [9]].values
print(X.shape)
```

```
(45200, 9)
```

Prétraitement :

Etape 4 Affichage du nombre de valeurs manquantes dans chaque variable

Un aperçu des données manquantes, essentiel pour une prise de décision informée

```
print(dataset.isnull().sum())
print(X)
```

```
ID                0
datetime          0
siteid         4518
offerid           0
category          0
merchant          0
countrycode       0
browserid      2345
devid          6864
click             0
hour              0
day_of_week       0
dtype: int64
[[6310005.0 99217 41706 ... nan 19 2]
 [nan 287164 89522 ... 'Tablet' 15 5]
 [2463708.0 948989 12052 ... nan 10 4]
 ...
 [2656998.0 324233 48430 ... nan 21 6]
 [nan 908599 904 ... 'Tablet' 7 0]
 [9223301.0 487571 41640 ... 'Mobile' 13 1]]
```

Prétraitement :

Etape 5 Remplacement des valeurs manquantes

Selon leur type, les valeurs manquantes dans les chaînes de caractères sont remplacées par les valeurs les plus fréquentes, tandis que les nombres sont remplacés par une valeur constante

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
X[:, [5, 6]] = imputer.fit_transform(X[:, [5, 6]])
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value=-1)
X[:, [0]] = imputer.fit_transform(X[:, [0]])
```

Prétraitement :

Etape 6

Encodage des colonnes catégorielles

Une transformation des colonnes catégorielles pour les rendre compatibles avec les algorithmes, améliorant ainsi la performance du modèle

```
categorical_columns = [4, 5, 6, 7, 8]

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

for column in categorical_columns:
    X[:, column] = label_encoder.fit_transform(X[:, column])
```

Prétraitement :

Etape 7

Séparation des données (Test et Train)

Nous divisons les données en ensembles d'entraînement et de test pour évaluer la performance du modèle

```
[ ]  from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Prétraitement :

Etape 8 Standardisation des caractéristiques

Une standarisation des caractéristiques rendre les caractéristiques d'un ensemble de données comparables en les mettant à la même échelle. , améliorant la stabilité des modèles

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler(with_mean=False)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
print(dataset.isnull().sum())
```

Approches Algorithmiques:

Pour résoudre ce problème, nous avons exploré plusieurs algorithmes de classification, chacun avec sa propre méthode d'apprentissage et de prédiction. Voici les principaux algorithmes que nous avons utilisés:

1. **Arbre de Décision**
2. **Forêts Aléatoires**
3. **Gradient Boosting / XG Boost**
4. **Régression Logistique**
5. **Machines à Vecteurs de Support (SVM)**
6. **KNN : K plus proches voisins**

Arbre de décision

L'algorithme d'arbre de décision est une méthode d'apprentissage statistique utilisée pour la classification et la régression. Il construit un arbre où chaque nœud représente une caractéristique et chaque feuille une prédiction. Pour prédire la variable "click" dans la base de données, on entraîne l'arbre sur l'ensemble d'entraînement, utilisant des caractéristiques telles que datetime, siteid, et offerid. Ensuite, on utilise l'ensemble de test pour évaluer la performance du modèle en prédiction de clic, ajustant si nécessaire pour améliorer la précision. L'arbre de décision fournit une compréhension des caractéristiques importantes pour la décision de clic, utile pour optimiser les campagnes publicitaires.

Arbre de décision

- **Les résultats de notre algorithme sont maintenant prêts à être révélés. Nous examinerons d'abord la précision (accuracy) obtenue, suivie de la matrice de confusion qui offre une vue détaillée de la performance du modèle.**

```
[ ]  from sklearn.metrics import accuracy_score
     accuracy = accuracy_score(y_test, y_pred)

     print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.8571902654867256
```

```
[ ]  from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
     print("Confusion Matrix:")
     print(cm)
```

```
Confusion Matrix:
[[4421  664]
 [ 627 3328]]
```

```
[ ]  from sklearn.metrics import f1_score
     f1 = f1_score(y_test, y_pred)

     print(f"F1 Score: {f1}")
```

```
F1 Score: 0.837548760538568
```

Forets Aléatoire

L'algorithme de forêt aléatoire combine plusieurs arbres de décision, chacun formé sur des sous-ensembles aléatoires des données d'entraînement, pour prédire la variable cible (ici, le "clic"). Il agrège ensuite les prédictions individuelles pour fournir une prédiction finale robuste et précise. En bref : il construit un groupe d'arbres prédictifs diversifiés, les entraîne sur des sous-ensembles aléatoires des données, et combine leurs prédictions pour une meilleure performance.

Forets Aléatoire

- **Les résultats de notre algorithme sont maintenant prêts à être révélés. Nous examinerons d'abord l'accuracy (précision) obtenue, suivie de la matrice de confusion qui offre une vue détaillée de la performance du modèle.**

```
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'F1 Score: {f1}')
```

```
Accuracy: 0.90221238938053097
F1 Score: 0.8864628820960698
```

```
[ ]  from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
     print("Confusion Matrix:")
     print(cm)
```

```
Confusion Matrix:
[[4678  356]
 [ 481 3525]]
```

Gradient Boosting / XG Boost

L'algorithme de Gradient Boosting construit séquentiellement des arbres de décision, chaque nouvel arbre corrigent les erreurs résiduelles des précédents, améliorant ainsi la prédiction de la variable cible. Il optimise progressivement la performance du modèle en se concentrant sur les erreurs résiduelles, produisant une prédiction finale robuste et précise.
XGBoost étend le Gradient Boosting en intégrant des optimisations et des mécanismes de régularisation avancés, offrant une performance améliorée et une convergence rapide pour la prédiction précise de la variable cible.

Gradient Boosting / XG Boost

Les résultats de notre algorithme sont maintenant prêts à être révélés. Nous examinerons d'abord l'accuracy (précision) obtenue, suivie de la matrice de confusion qui offre une vue détaillée de la performance du modèle.

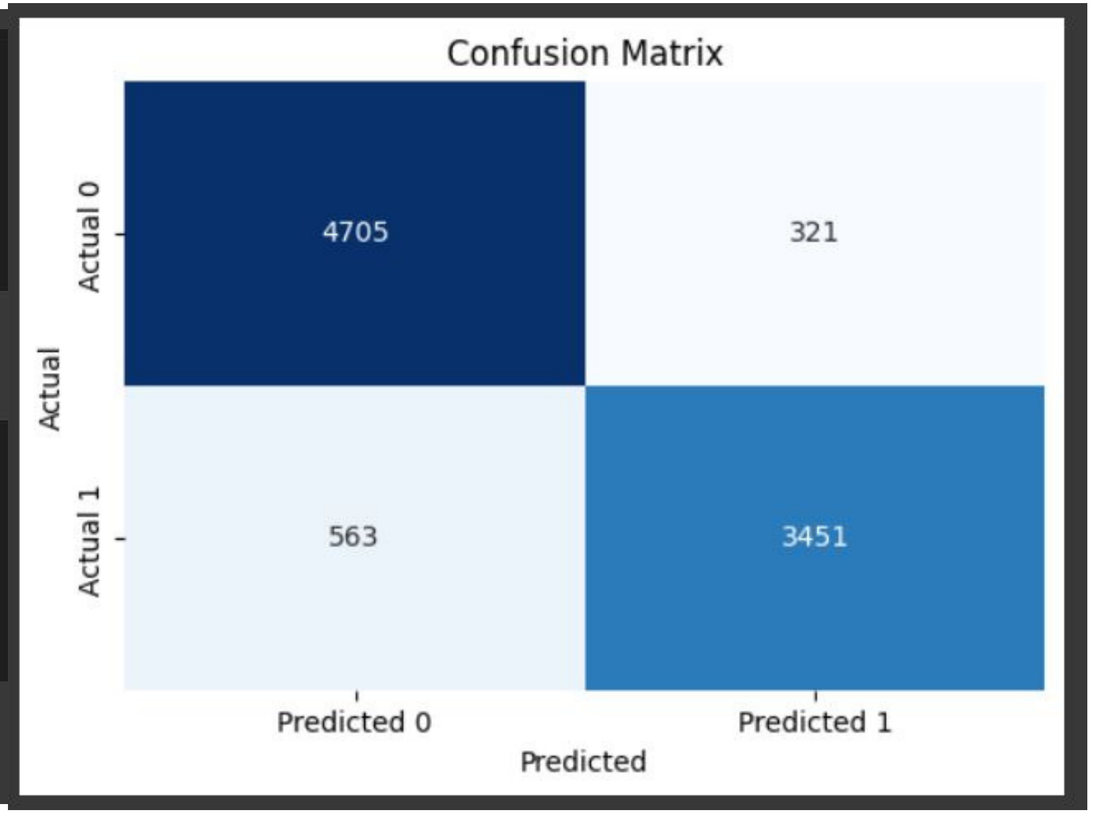
```
[ ]  from sklearn.metrics import accuracy_score
     accuracy = accuracy_score(y_test, y_pred)

     print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9074115044247788
```

```
[ ]  from sklearn.metrics import f1_score
     f1 = f1_score(y_test, y_pred)
     print(f1)
```

```
0.893875998478509
```



Régression Logistique

L'algorithme de régression linéaire cherche à modéliser la relation linéaire entre les variables indépendantes et la variable dépendante. Dans le contexte de la prédiction de la variable continue "click", la régression linéaire utilise des coefficients pour ajuster une ligne qui minimise les erreurs quadratiques, offrant ainsi une estimation continue de la probabilité de clic en fonction des caractéristiques disponibles.

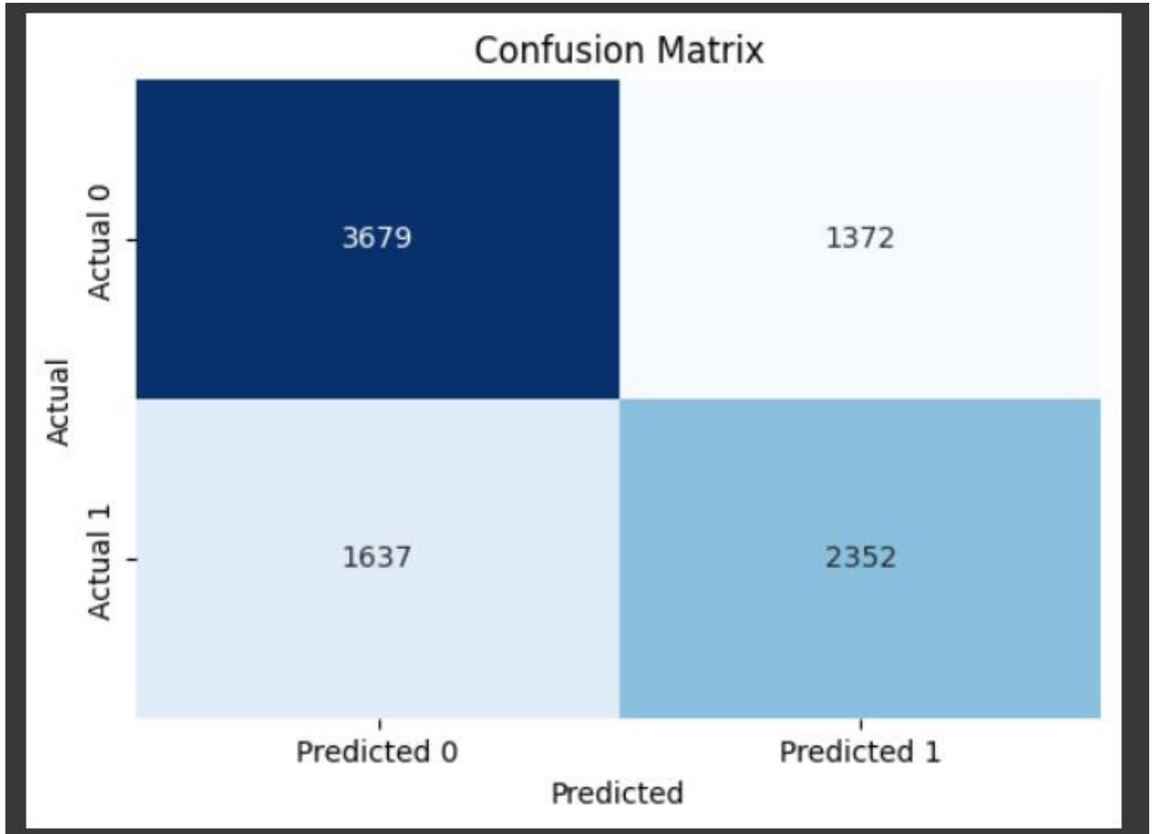
Régression Logistique

- **Les résultats de notre algorithme sont maintenant prêts à être révélés. Nous examinerons d'abord l'accuracy (précision) obtenue, suivie de la matrice de confusion qui offre une vue détaillée de la performance du modèle.**

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.6671460176991151
```




```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print(f1)
```

```
0.6098794243485025
```

Machines à Vecteurs de Support

L'algorithme SVM peut être utilisé pour prédire la variable "click" en identifiant un hyperplan dans l'espace des caractéristiques qui sépare efficacement les données en deux classes (clic ou pas de clic). En exploitant les caractéristiques telles que l'identifiant du site web, l'offre, la catégorie, le pays, le navigateur et le type d'appareil, SVM cherche à optimiser la marge entre les deux classes, offrant ainsi une prédiction précise de la probabilité de clic pour chaque observation.

Machines à Vecteurs de Support

- **Les résultats de notre algorithme sont maintenant prêts à être révélés. Nous examinerons d'abord l'accuracy (précision) obtenue, suivie de la matrice de confusion qui offre une vue détaillée de la performance du modèle.**

```
[ ] from sklearn.metrics import accuracy_score
    accuracy = accuracy_score(y_test, y_pred)

    print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.8804203539823009
```

```
] from sklearn.metrics import confusion_matrix
  cm = confusion_matrix(y_test, y_pred)
  print("Confusion Matrix:")
  print(cm)
```

```
Confusion Matrix:
[[4573  470]
 [ 611 3386]]
```

```
] from sklearn.metrics import f1_score
  f1 = f1_score(y_test, y_pred)
  print(f1)
```

```
0.8623456004074875
```

KNN : K plus proches voisins

L'algorithme des k-plus proches voisins (KNN) est une méthode d'apprentissage supervisé qui prédit la classe d'une observation en se basant sur la classe des k voisins les plus proches dans l'espace caractéristique. Pour l'utiliser dans la base de données pour prédire la variable "click" :

- Choix de la Caractéristique : Sélectionnez les caractéristiques pertinentes comme datetime, siteid, et offerid.
- Entraînement du Modèle : Entraînez le modèle sur l'ensemble d'entraînement.
- Prédiction : Utilisez le modèle pour prédire "click" sur l'ensemble de test en identifiant les k voisins les plus proches.
- Évaluation : Mesurez la performance du modèle en utilisant des métriques telles que la précision.

KNN offre une approche simple pour la prédiction basée sur la proximité entre observations dans l'espace caractéristique.

KNN : K plus proches voisins

- **Les résultats de notre algorithme sont maintenant prêts à être révélés. Nous examinerons d'abord l'accuracy (précision) obtenue, suivie de la matrice de confusion qui offre une vue détaillée de la performance du modèle.**

```
]  from sklearn.metrics import accuracy_score
   accuracy = accuracy_score(y_test, y_pred)

   print(f'Accuracy: {accuracy}')
```

```
   Accuracy: 0.863716814159292
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[4573  481]
 [ 751 3235]]
```

```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print(f1)
```

```
0.840041547649961
```

Meilleur Algorithme

L'algorithme le plus adapté dans notre cas : Forets Aléatoire

L’algorithme de Forets Aléatoire se distingue avec une précision remarquable, mesurée à 90%. La matrice de confusion offre une analyse détaillée de sa performance. Ces résultats soulignent le succès du Forets Aléatoiredans notre résolution de problématique.

Conclusion

Notre exploration des algorithmes d'apprentissage statistique, axée sur la prédiction de la probabilité de clic, a conduit à un succès notable grâce à l'application du Gradient Boosting. Ce modèle offre une précision exceptionnelle, permettant à l'entreprise de prendre des décisions éclairées avant le lancement de ses futures campagnes publicitaires. En utilisant des données anonymisées, notre approche respecte les exigences de confidentialité. En somme, notre démarche démontre l'efficacité des techniques modernes en apprentissage statistique pour résoudre des problématiques critiques dans le contexte du marketing d'affiliation.

Merci !

N'hésitez pas de posez des questions.