Compte Rendu 3: présentation des résultats.

Réalisé par : Rachida OUCHENE, Lylia TOUAZI, Koussaila HAMMOUCHE.
GROUPE 1

I. Introduction

Le dataset que nous utilisons est extrait de ce site https://archive.ics.uci.edu/ml/datasets/Shill+BiddingDataset. Le jeu de données a été récolté par Ahmad Alzahrani et Samira Sadaoui des enchères eBay sur le produit iphone7. Notre objectif est d'implanter et de comparer 3 méthodes d'apprentissage sur ce jeu de données, afin de prédire si une enchère est fraudée ou non.

Dans notre première partie nous avons décrit et analysé le jeu de données, dans la deuxième partie nous avons décrit et justifié les 3 méthodes d'apprentissage automatique utilisées, et dans cette partie nous allons présenter les résultats obtenus pour chaque méthode et choisir la meilleure.

II. Rappel

- **Rappel sur les données:**

- Nombre d'individus : 6321.
- Valeurs manquantes : Pas de valeurs manquantes.
- La target : Class (89.3% de class 0 : normal, 10.7% de class 1 : fraudée).
- Variables explicatives : Bidder_tendency, Bidding_Ratio, Successive_Outbidding, Winning_Ratio, Last_bidding, Auction_Bids, Starting_Price_Average, Early_Bidding ces dernieres sont de **types float**, Auction_Duration de **type int**.
- Nous avons supprimé les identifiants "Record_ID, Auction_ID et Bidder_ID". Ainsi que "Successive_Outbidding" car elle est trop corollaire avec la variable class.
- Le jeu de données est normalisé : Toutes les valeurs dans l'intervalle [0,1], sauf Auction_Duration.

- **Rappel de choix des Méthodes**

Pour évaluer notre dataset, nous avons choisi les modèles suivants :

- K plus proches voisins(k-Nearest Neighbours).
- Support vector machine(SVM).
- Arbre de décision.

III. Protocole

A. Découpage de DataSet en Training et Testing Set.

Pour diviser notre target et nos features en Training et Testing set, nous avons utilisé "train_test_split". Pour préserver les mêmes proportions d'exemples dans chaque classe nous avons utilisé le paramètre "stratify" de "train_test_split". Les résultats obtenu sont les suivants :

X-train-set size = 5056 / y-train-set size = 5056.
X-test-set size = 1265 / y-test-set size = 1265.
Fraud cases in test-set = 135.
Non Fraud cases in test-set = 1130.

B. Choix des hyper-paramètres:

1.Knn : Nous avons utilisé deux méthodes pour choisir les hyper-paramètres de knn :

- **en utilisant une boucle** : Comme nous l'avons vu dans le rendu 2, nous avons décidé d'utiliser la méthode de Stratified-K-Fold Cross Validation, dans le but de valider un modèle avec toutes les configurations possibles. Du coup, ici nous avons pris 30 valeurs de K n_neighbors à tester et à valider avec Stratified-K-Fold Cross Validation de 8 splits. Pour chaque modèle de paramètre "n_neighbors = k" nous avons obtenu 8 scores, et nous sauvegardons pour chaque modèle la moyenne de ces scores. Puis nous choisissons le modèle qui a le meilleur score. La **figure 1** représente la moyenne des scores par rapport à chaque k de Knn, d'après la **figure 1** nous remarquons que les bons scores moyen sont obtenu par des n_neighbors impair, ceux qui est logique car nous avons un nombre pair de class. Nous observons aussi que quand K=9, nous avons un meilleur score moyen. Du coup, nous avons choisi le paramètre "n_neighbors = 9" pour évaluer notre modèle Knn.

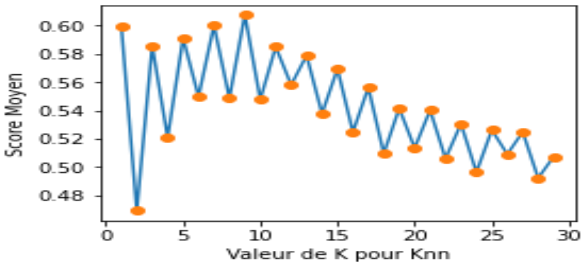

Figure 1: Score moyen pour chaque paramètre de Knn.

- **En utilisant GridSearchCV:**

Pour tester les autres hyper-paramètres que le n_neighbors comme par exemple le type de distance, le mieux c'est d'utiliser GridSearchCV qui nous permet de trouver les meilleurs hyper-paramètres en comparant les différentes performances de chaque combinaison grâce à la technique de cross validation. Le résultat est le suivant :

Score = 0.6075534660370083
Best_params = {'metric': 'minkowski', 'n_neighbors': 9}
Best_estimator = KNeighborsClassifier(n_neighbors=9)

2.SVM :

Pour cette méthode on a réglé les paramètres C, gamma et kernel sur les choix suivantes : 'C': [0.1, 1, 10, 100, 1000], 'gamma' : [1, 0.1, 0.01, 0.001, 0.0001], 'kernel': ['rbf','linear']. Grâce à l'utilisation de 'GridSearchCV' nous avons trouvé le résultat suivant :

Score = 0.6830077008354049
Best_params = {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf'}
Best_estimator = SVC(C=1000, gamma=0.01)

3.Arbre de décision :

Nous avons réglé les paramètres "criterion" et "max_depth" pour améliorer le score de l'arbre de décision, grâce 'GridSearchCV' nous avons pu trouver les valeurs de ces deux paramètres qui vont maximiser le score et améliorer la performance de notre modèle. Le résultat de 'GridSearchCV' est présenté comme suit :

Score = 0.678944169562878
Best_params = {'criterion': 'entropy', 'max_depth': 4'}
Best_estimator = DecisionTreeClassifier(criterion='entropy', max_depth=4)

C. Evaluation des méthodes

1. Knn :

- **Résultat:**

Après avoir choisi les meilleurs hyper-paramètres et avoir entraîné notre modèle Knn sur ces derniers avec le Training Set, nous avons évalué notre modèle avec le Testing Set. La **figure 2** représente la matrice de confusion obtenue après l'évaluation. Dans la **figure 2** nous remarquons que parmi les 1130 enchères de classe 0 (non frauder) du Testing Set, 1099(vrai-positif) ont été estimés comme non fraudés et 31 (faux-négatif) comme fraudés. Et parmi les 135 enchères de classe 1 (frauder) du Testing Set, 72 (vrai-négatif) ont été estimés comme fraudés et 63 (faux-positif) comme non fraudés alors qu'ils sont fraudés.

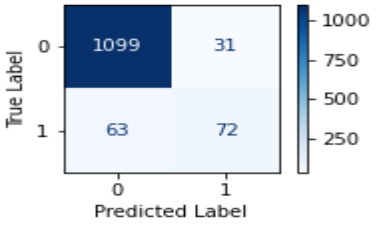

Figure 2 : Matrice de confusion de KNN.

Comme nous l'avons vu dans le rendu 2, pour évaluer notre modèle nous nous intéressons à la métrique précision et recall à la fois, et pour cela nous les résumons dans F1. **La figure 3** représente la valeur de la précision, recall et F1-score pour la classe 0 et 1. Nous observons dans cette figure que F1 dans la classe 1 a un score de 0.61 et une précision de 0.70, ce qui est déjà pas mal pour prédire si une enchère est fraudée ou non.

	précision	recall	f1-score
class 0	0.95	0.97	0.96
class 1	0.70	0.53	0.61

Figure 3 : Recall, précision et f1-score obtenu à partir de la matrice de confusion de knn.

❖ **Avantages:**

La phase d'apprentissage de la classification Knn est beaucoup plus rapide par rapport aux autres algorithmes de classification. C'est un algorithme simple et facile à mettre en œuvre. Il n'a besoin d'aucune hypothèse sur les données.

❖ **Limites :**

La phase de prédiction de la classification Knn est plus lente et plus coûteuse en termes de temps et de mémoire, car il doit garder en mémoire l'ensemble des observations pour pouvoir effectuer sa prédiction. Aussi le choix de la méthode de calcul de la distance ainsi que le nombre de voisins peut ne pas être évident. Il faut essayer plusieurs combinaisons pour avoir un résultat satisfaisant.

2. SVM :

❖ **Résultat:**

Après avoir fait le choix des meilleurs hyper-paramètres, entraîner notre modèle sur le Training Set et l'évaluer sur le Testing Set, nous avons obtenu 1104 qui ont été estimés comme non fraudés et 26 comme fraudés (ie : 26 faux-négatifs) parmis les 1130 enchères de classe 0 du Testing Set, que nous pouvons observer sur la ligne 1 de la **Figure 4**. Sur les 135 enchères de classe 1 de Testing Set, 43 ont été estimés comme non fraudés (ie : 43 faux-positifs) et 92 comme fraudés, que nous pouvons observer sur la ligne 2 de **Figure 4**. La **Figure 4** présente la matrice de confusion obtenue après l'évaluation de notre modèle sur le Testing Set.

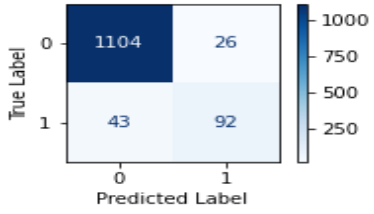

Figure 4 : Matrice de confusion de SVM

Si nous nous concentrerons maintenant sur **Figure 5** qui représente les valeurs de la précision, de recall et f1-score qui est calculée comme suit : **f1-score =** $2 * \frac{recall*précision}{recall+récision}$

Nous observons que la valeur de **f1-score** est de 0.73 pour la classe 1 ce qui est plutôt bien pour prédire si une enchère est fraudée ou non.

	précision	recall	f1-score
class 0	0.96	0.98	0.97
class 1	0.78	0.68	0.73

Figure 5 : Recall, précision et f1-score obtenu à partir des valeurs de la matrice de confusion de SVM.

❖ **Avantages:**

Svm permet de traiter des problèmes non linéaires avec le choix de noyau et l'astuce du noyau est la vraie force de SVM. Avec une fonction de noyau appropriée, nous pouvons résoudre n'importe quel problème complexe. Elle est relativement efficace en mémoire.

❖ **Limites :**

La difficulté de SVM est de régler ces hyper-paramètres, il n'est pas évident de visualiser leurs impacts. Elle ne convient pas aux grands ensembles de données en raison de son temps de fonctionnement très élevé. Le choix d'une bonne fonction du noyau n'est pas facile.

3. Arbre de décision :

❖ **Résultat:**

Suite à l'entraînement de notre modèle avec les paramètres générés par GridSearchCV nous avons obtenu la matrice de confusion **Figure 6**. Nous observons que pour la classe 0, 1100 ont été estimé comme de vrai positive et 30 faux négatifs. Par contre dans la classe 1, 90 ont été estimé comme vrais négatifs et 45 comme des faux positifs.

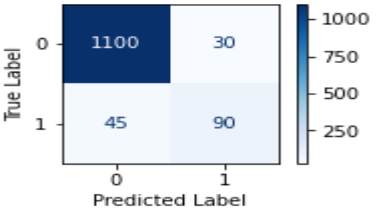

Figure 6 : Matrice de confusion de l'arbre de décision.

Grâce à cette matrice de la **figure 6** nous avons calculé la précision, recall et f1-score qui sont présentés dans **figure 7**. Nous nous intéressons aux f1-score où nous remarquons que le score de la classe 1 est de 0.71.

	précision	recall	f1-score
class 0	0.96	0.97	0.97
class 1	0.75	0.67	0.71

Figure 7 : recall, précision et f1-score obtenu à partir de la matrice de confusion de l'arbre de décision.

❖ **Avantages :**

Il est simple à comprendre et à interpréter. Il peut travailler sur des données avec peu de préparation par exemple, (pas de normalisation, de valeurs vides à supprimer) .
Sa complexité et la taille des arborescences peuvent être contrôlées en définissant ces valeurs de paramètres.

❖ **Limites :**

Les arbres de décision sont instables et ont tendance à ouverfiter, peut être extrêmement sensibles aux petites perturbations dans les données, tel qu'un léger changement peut entraîner une structure d'arbre radicalement différente.
Il existe des concepts qui sont un peu difficiles à exprimer à l'aide d'arbres de décision (comme XOR ou la parité) car chaque nœud a deux possibilités (gauche où droite), donc il existe certaines relations variables que les arbres de décision ne peuvent pas apprendre. Il est sensible au bruit et aux points aberrants.

IV. Conclusion

En observant les scores obtenus par l'évaluation des trois méthodes (KNN, SVM et arbre de décision), qui sont présentés dans les figures (3, 5 et 7) respectivement, nous constatons que la méthode SVM donne le meilleur résultat avec un f1-score de 0.73, un recall de 0.68 et d'une précision de 0.78.

Nous concluons pour ce projet, que le modèle SVM est le meilleur modèle qui nous permet de prédire si une enchère est frauder ou non, d'éviter au maximum de prédire que c'est une arnaque alors que c'est faux et aussi d'éviter au maximum de prédire que ce n'est pas une arnaque alors que s'en est une.