**[SLIDE 1]**

L’ONCFM, qui est l’organisation nationale de lutte contre le faux-monnayage a fait appel à nous pour développer 2 algorithmes de détection de faux billets d’euros à partir de certaines dimensions de ces derniers. En effet, elle a noté qu’il subsistait des différences de dimensions entre les vrais et les faux billets, mais comment les détecter de manière automatique ?

Pour mener à bien cette mission, il nous a été fourni

* Un échantillon de données pour paramétrer nos modèles et
* Un échantillon test de billets non étiqueté pour les tester

**[SLIDE 2]**

* + - * On va d’abord s’attacher à décrire notre jeu de données de départ au travers d’une analyse descriptive, en passant par la méthodologie utilisée pour compléter certaines valeurs manquantes.
* Ensuite, nous allons d’une part, détailler le premier modèle prédictif que nous avons mis en place avec la méthode de clustering K-means
* Et d’autre part, le second modèle prédictif entrainé avec une régression logistique.
* Et Enfin, nous testerons notre modèle en direct !

**[SLIDE 3]**

* pour mener à bien cette mission, nous avons utilisé d’une part visual studio code pour accueillir nos notebooks, et d’autre part les librairies Python : pandas et numpy pour l’ EDA, matplotlib et seaborn pour la visualisation et enfin scikit-learn pour l’analyse exploratoire multivarié et nos modèles prédictifs de machine Learning.

**[SLIDE 4]**

Le jeu de données qui nous a été fourni comporte 1500 observations étiquetées ! 7 colonnes correctement formatées dont 6 colonnes numériques représentant les dimensions géométriques (marges, hauteurs, diagonale, longueur) et une colonne binaire de format « boolean » indiquant la nature du billet (vrai ou faux billet). Parmi ces 1500 billets, 1000 sont vrais et 500 faux soit 1/3 des billets ! Les données ne contiennent ni valeurs négatives ni valeurs aberrantes selon les statistiques descriptives (les valeurs maximales sont très proches de la moyenne). Nous parlerons *a priori* de valeurs atypiques dans ce cas !

**[SLIDE 5]**

Visuellement, les variables ne suivent pas la même distribution selon que les billets soient vrais ou faux : et en particulier pour les variables **margin\_low et length** ! Au test de normalité Shapiro-Wilk, la diagonale et les hauteurs sembleraient suivre une distribution normale à contrario pour les marges et la longueur.

**[SLIDE 6]**

* Les variables les plus représentatives des 2 catégories de billets sont « length » et « margin\_low », appuyé par leur coefficient de corrélation très élevé !
* Length est la variable ayant une forte corrélation à la fois avec is\_genuine et margin\_low ! Et à l’inverse, diagonale est la variable ayant le plus faible coefficient de corrélation !

**[SLIDE 7]**

Le jeu de données contient 37 valeurs manquantes dans la colonne margin\_low : pour 29 billets vrais et 8 faux billets que nous avons décidé de combler à l’aide d’une régression linéaire !

**La régression linéaire** est une méthode de **machine Learning** qui permet de prédire une variable dépendante continue à partir d’une ou plusieurs variables indépendantes !

**[SLIDE 8]**

* **Pour utiliser une régression linéaire, Il faut s'assurer que la relation entre la variable cible et les prédicteurs soit linéaire**.
* Nous avons d’abord opté pour une régression linéaire simple avec la variable explicative length qui présente une relation linéaire beaucoup plus appuyée que les autres variables, puis nous sommes passés à des régressions linéaires multiples en vue d’améliorer les performances de notre modèle : **Les résultats du 3ème modèle incluant toutes les variables explicatives sont significatifs** :
  1. la probabilité de la F-statistic est très proche de 0 malgré une qualité de prévision médiocre.
  2. Toutes les variables sont significatives, il n’existe pas de problème de multi-colinéarité, le VIF de chaque variable est < 10.
  3. La moyenne de l’erreur quadratique est proche de 0 ce qui indique une meilleure performance du modèle !
  4. Notre modèle performe légèrement mieux sur les données de test ce qui est généralement une bonne indication !
  5. La différence entre la MSE du jeu de test et du jeu d’entrainement n’est pas significative ce qui indique que le modèle généralise bien et ne semble pas souffrir **d’overfitting majeur !**

**[SLIDE 9]**

* La moyenne des résidus est très proche de zéro, ce qui est un bon résultat, mais les différents tests réalisés de Breusch-Pagan et de Kolmogorov-Smirnov n’ont pas permis de confirmer les 2 hypothèses :
  + **D’homoscédasticité et**
  + **De normalité des données**

Néanmoins, nous procédons à la complétude des valeurs manquantes de la colonne margin\_low avec **les valeurs prédites**, car dans l’ensemble nous obtenons une robustesse suffisante pour les inclure dans notre analyse !!!!

**[SLIDE 10]**

* **Nous obtenons enfin un dataset complet de 1500 lignes par colonne, toujours composé des mêmes variables !**

**[SLIDE 11]**

* Afin d’approfondir l’exploration, nous avons réalisé une analyse en composantes principales pour visualiser nos données en 2 dimensions sur un plan orthonormé !

**[SLIDE 12]**

* Après standardisation de nos données, nous observons qu’il n’existe pas de différences flagrantes entre les données d’origines et les données normalisées !
* Le scree plot nous indique que 60% de la variance de nos données sont captées par les 2 premiers axes principaux.
* Et enfin la heatmap laisse apparaitre que le 1er axe principal est surtout représenté par la longueur et la marge inférieure (coeff. de corrélation >= 0.5) et le 2ème axe est largement dominé par la diagonale ! Si on se souvient bien des caractéristiques de nos faux billets présentés sur le graphique des boites à moustaches (hauteurs et marges supérieures à la moyenne), nous devrions retrouver nos faux billets sur la partie droite du 1er plan factoriel !!!!!

**[SLIDE 13]**

* La projection des billets permet de distinguer 2 groupes de billets : les vrais billets qui sont nettement représentés par leur longueur (à gauche) et les faux billets représentés par leurs hauteurs et leurs marges. Il subsiste une légère zone d’incertitude entre ces 2 groupes, d’où la nécessité d’appliquer la méthode de clustering K-means comme outil de classification, cette étape constituera notre premier modèle prédictif !

**[SLIDE 14]**

* En effet, l’algorithme du K-means est un algorithme de classification non supervisé qui par itérations successives va rechercher des similarités sur des données non étiquetées d’où le terme « non supervisé » ! Notre modèle de classification va donc rattacher les clusters de données à une étiquette « vrai » ou « faux » pour pouvoir qualifier nos billets de banque et être utilisé à des fins de prévision.
  1. **La première étape préliminaire a consisté à rechercher le nombre optimal de clusters** à l’aide de méthodes telles que :
     + La méthode du coude
     + Le score de Silhouette
* *La méthode du coude calcule la somme des distances au carré de chaque point par rapport à son centroïde (c’est l’inertie intra-classe)*
* *Et le score de silhouette calcule la distance moyenne entre le point et les autres points dans le même cluster et la distance moyenne entre le point et les autres points dans le cluster le plus proche !*

Dans les 2 méthodes nous confirmons un k optimal entre 2 et 3 ! car ils offrent la meilleure qualité de clustering ! Le choix optimal du nombre de clusters est souvent un équilibre entre avoir un silhouette score élevé et une inertie plus faible.

!!!!Au-delà de 3 clusters, on note que la qualité des clusters commence à diminuer plus rapidement à partir de ce point !!!

* 1. La deuxième étape d’optimisation est le paramétrage pour faciliter l’apprentissage du modèle et rendre les résultats reproductibles ce qui nous a conduit à définir un random\_state. Par défaut c’est la méthode K-means++ qui est utilisé !
  2. Nous pouvons enfin passer au processus d’entrainement qui consiste à :
     + Scinder les données en train/test et définir un « test\_size ». Nous lui avons indiqué 20% des données pour le test et ainsi 80% ont été utilisé pour l’entrainement !

**[SLIDE 15]**

Si on analyse à présent les centroïdes de nos différents clusters dans les 2 modèles ET les matrices de confusion qui en résultent sur nos jeux d’entrainement et de test, on s’aperçoit qu’un de nos clusters reste relativement inchangé quant à son centroïde alors que le second semble se « déformer » en plusieurs clusters imbriqués qui correspondraient à nos faux billets dont les dimensions moyennes sont plus fortes.

Les matrices de confusion confirment un modèle à 3 clusters où à la fois les faux positifs et faux négatifs sont les plus minimisés appuyé par les résultats du jeu de test.

**Toutes les métriques de performances confirment que le modèle à 3 Clusters est légèrement plus précis que celui à 2 clusters !**

**[SLIDE 16]**

Intéressons-nous à présent à notre second modèle de détection, celui basé sur la régression logistique. La différence essentielle ici avec le k-means est qu’il s’agit d’un algorithme de classification. De surcroit cette régression est réalisée sur des données étiquetées, c’est donc un algorithme dit « supervisé ».

[SLIDE 17]

* La première étape a été de faire tourner un premier « fit » du modèle à un seuil de 30% ! Nous avons obtenu un score d’efficacité de 99% ce qui est un excellent score, appuyé par la courbe de ROC qui présente une courbe pratiquement à angle droit au-dessus de la diagonale.
* Mais pour la détection de faux billets de banque, il est crucial de minimiser les faux positifs qui peuvent avoir un plus grave impact !
* Nous avons donc recherché à optimiser les performances de notre modèle logistique

**[SLIDE 18]**

* **Pour ce faire, nous avons éliminé de notre modèle les 2 variables les moins significatives à l’aide de la méthode de validation croisée RFE pour aboutir à un modèle final avec seulement 4 variables qui coïncident avec les 4 variables les plus fortement corrélée à notre 1er axe principal de notre ACP !**
* **Et par ailleurs nous avons utilisé la fonctionnalité GridSearchCV de scikit-learn qui permet de tester le modèle de régression logistique avec différents paramètres : Nous avons établis 3 groupes d’hyper paramètres : saga, liblin et nsag. Grid-search recherche les meilleurs paramètres pour entrainer le modèle final !**
* **Nous avons opté pour le modèle saga qui a apporté une meilleure performance ! Nous avons réussi à minimiser au maximum les faux positifs.**

**[SLIDE 19]**

**Pour conclure cette analyse, nos 2 algorithmes retournent le même résultat sur l’échantillon de production même si ces 2 méthodes fonctionnement sur des fonctionnalités différentes.**

**On peut toutefois souligner que dans la pratique il est plus courant d’avoir recours à des algorithmes d’apprentissages supervisés comme la régression logistique pour répondere à ce genre de problématique !!!**