Projet 10:

Détectez des faux billets avec R ou Python

Introduction

Dans le but de mettre en place des méthodes d'identification des contrefaçons des billets en euros. On met place une modélisation afin d'identifier automatiquement les vrais des faux billets à partir des dimensions du billet et des éléments qui le composent.

Mission

Effectuer une prestation auprès de l'Organisation nationale de lutte contre le faux-monnayage (ONCFM).

Démarches

Créer un algorithme sur python qui utilise des méthodes de prédiction pour identifier un maximum de faux billets

Etape 1

Préparation de l'environnement et présentation des données

- Importation des librairies
- Importations des données

Etape 2

Analyse descriptive des données

- Exploration des Données
- Répartition des données
- Imputation des valeurs nulles (Regression linéaire)

Etape 3

Prédiction des Vrais-Faux billets

- Importation des données
- Régression logistique
- K_Means
- KNeighborsClassifier
- Comparaison des algorithmes

Etape 4

Application au jeu de donnée final

- Application finale

Etape 1

Le jeu de données

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	True	171.81	104.86	104.95	4.52	2.89	112.83
1	True	171.46	103.36	103.66	3.77	2.99	113.09
2	True	172.69	104.48	103.50	4.40	2.94	113.16
3	True	171.36	103.91	103.94	3.62	3.01	113.51
4	True	171.73	104.28	103.46	4.04	3.48	112.54
...	...	...	...	...	...	...	...
1495	False	171.75	104.38	104.17	4.42	3.09	111.28
1496	False	172.19	104.63	104.44	5.27	3.37	110.97
1497	False	171.80	104.01	104.12	5.51	3.36	111.95
1498	False	172.06	104.28	104.06	5.17	3.46	112.25
1499	False	171.47	104.15	103.82	4.63	3.37	112.07

1500 rows × 7 columns

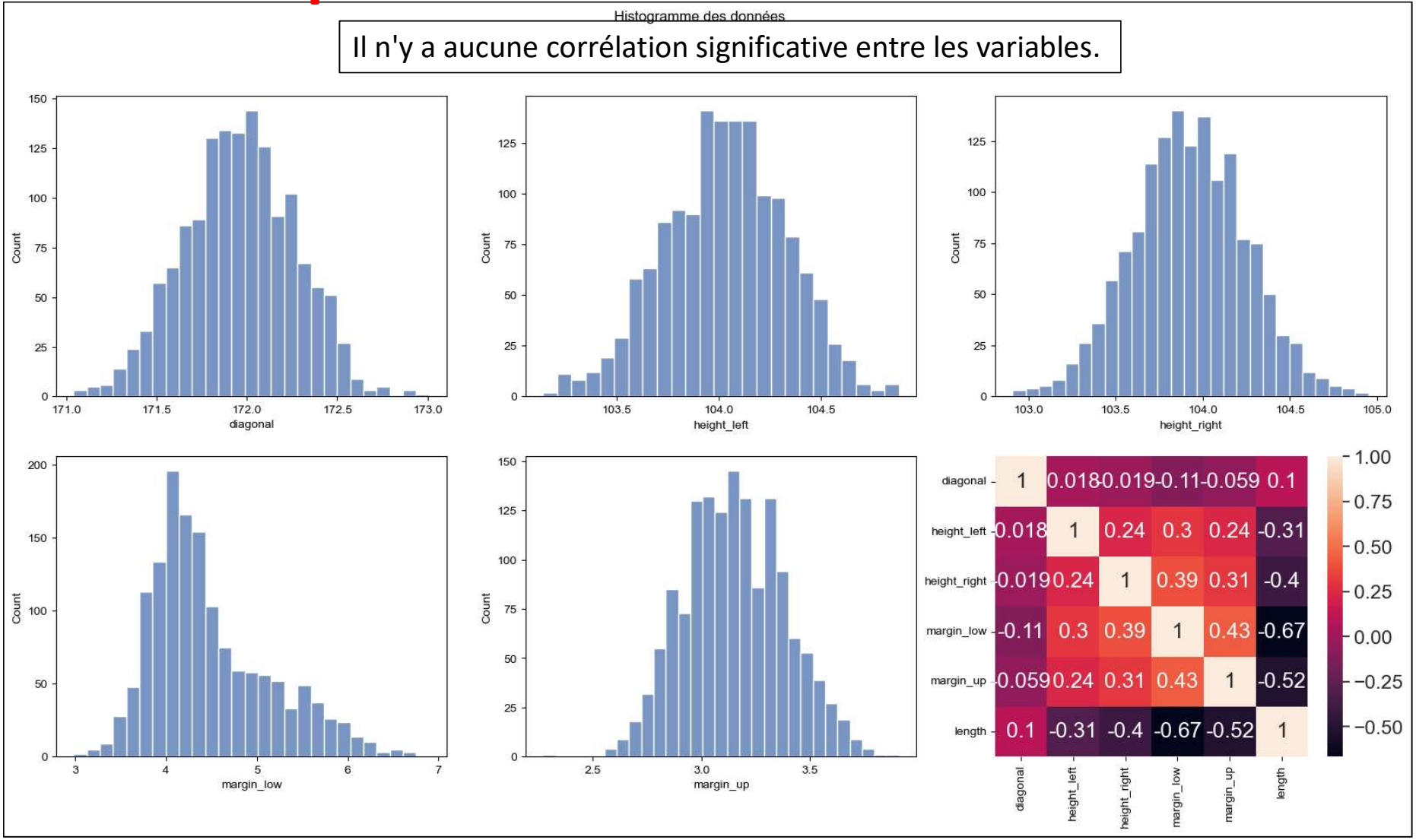
```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   is_genuine    1500 non-null   bool
 1   diagonal      1500 non-null   float64
 2   height_left   1500 non-null   float64
 3   height_right  1500 non-null   float64
 4   margin_low    1463 non-null   float64
 5   margin_up     1500 non-null   float64
 6   length        1500 non-null   float64
dtypes: bool(1), float64(6)
memory usage: 71.9 KB
```

- On remarque que la colonne 'margin_low' contient 37 valeurs nulles, On les récupère en appliquant une regression linéaire

Etape 2

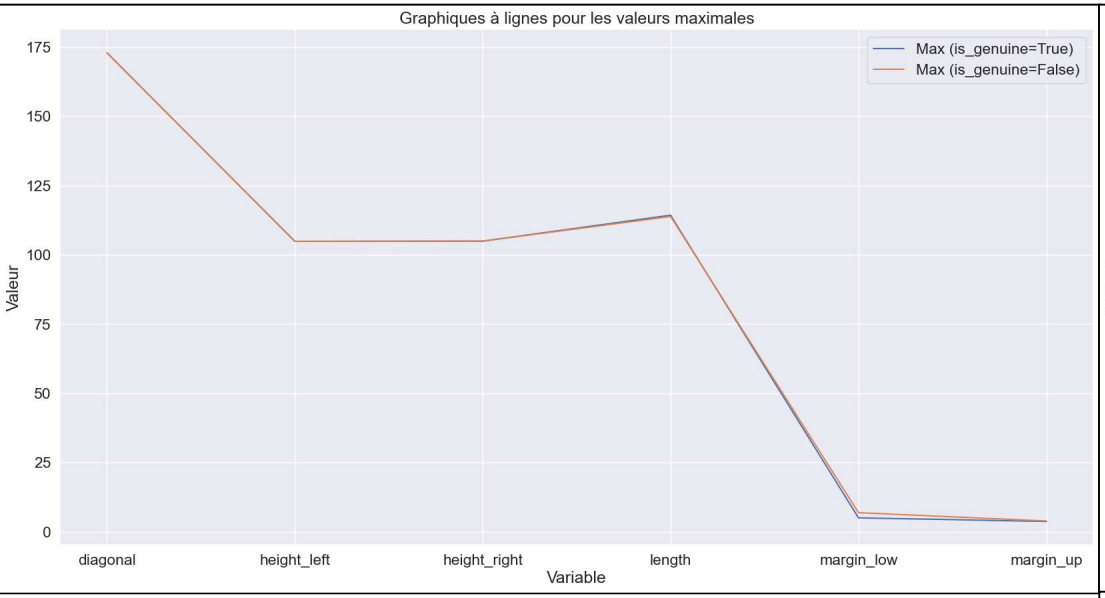
Répartition des variables

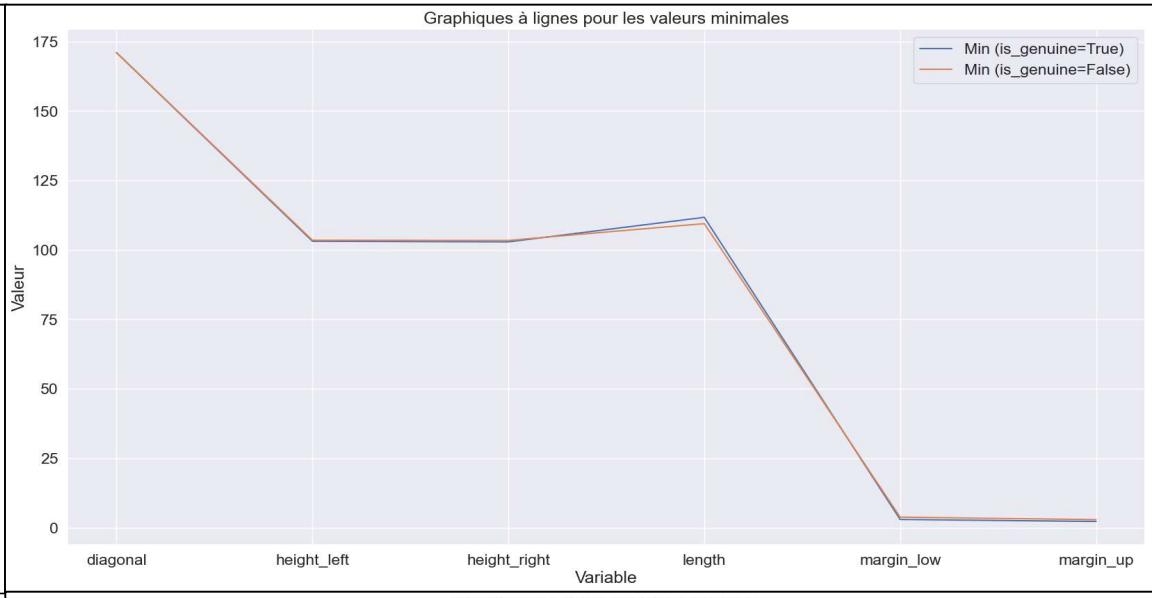
Il n'y a aucune corrélation significative entre les variables.



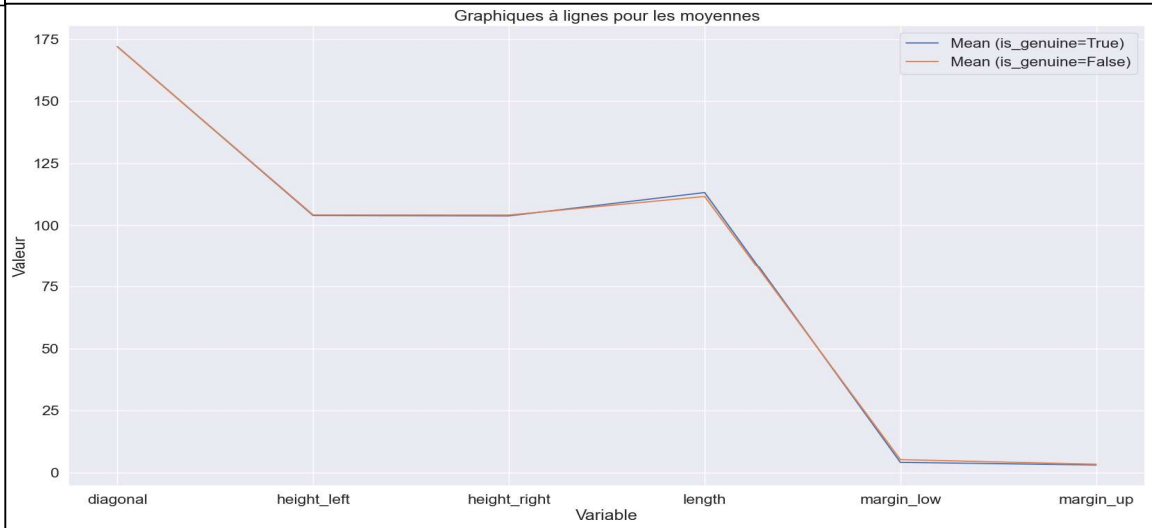
Etape 2

Comparaison entre vrais et faux billets





j ai comparé les variances des vrais et faux billets pour s'assurer des variables qui présentes les différences les plus importantes entre les vrais et les faux billets



Etape 2

Comparaison entre vrais et faux billets

j ai testé les différences entre les billets 'True' et 'False‘ en utilisant des tests statistiques: **ANOVA** pour les variable qui suivent une loi normale et **Kruskal–Wallis** pour les variables ne suivent pas une loi normal

Analyse de variance (ANOVA)

Avec un risque premier alpha de 5%, on émets les deux hypothèses suivantes:

H0: Les moyennes de 'margin_low' des vrais billets et faux billets sont sont égaux

H1: Les moyennes de 'margin_low' des vrais billets et faux billets ne sont pas égaux

Etape 2

Imputation des valeurs manquantes

On effectue une régression linéaire pour remplacer les valeurs nulles.

Création d'une copie du jeu de données

```
data_imputées = data.copy()
data_imputées.info()
```

Séparation des données

```
for column in data_imputées.columns:
    missing_indices = data_imputées[column].isnull()
    print(column)
    if missing_indices.any():
        X_complet = data_imputées.loc[~missing_indices, data_imputées.columns != column]
        y_complet = data_imputées.loc[~missing_indices, column]
        X_incomplet = data_imputées.loc[missing_indices, data_imputées.columns != column]
```

Construction du modèle

```
model = LinearRegression()
model.fit(X_complet, y_complet)
```

Prediction des valeurs manquantes

```
y_predit = model.predict(X_incomplet)
y_predict = y_predit.round(2)
y_predict
```

Remplacement des données manquentes avec les données prédites

```
data_imputées.loc[data_imputées['margin_low'].isna(), 'margin_low'] = y_predit
data_imputées.info()
```

```
#   Column        Non-Null Count  Dtype
---  ------        --------------  -----
0   is_genuine    1500 non-null   bool
1   diagonal      1500 non-null   float64
2   height_left   1500 non-null   float64
3   height_right  1500 non-null   float64
4   margin_low    1500 non-null   float64
5   margin_up     1500 non-null   float64
6   length        1500 non-null   float64
```

Etape 3

Les algorithmes de prédiction

Définitions

Matrice de Confusion:

Une matrice de confusion est une table utilisée dans la classification des problèmes d'apprentissage automatique et pour évaluer les performances d'un modèle de classification.

Précision:

la proportion de prédictions correctes parmi les points que l'on a prédits positifs. C'est la capacité de notre modèle à ne déclencher d'alarme que pour un vrai incendie.

Précision=TP/(TP+FP)

Rappel ou Sensibilité (Recall):

c'est le taux de vrais positifs, c'est à dire la proportion de positifs que l'on a correctement identifiés. C'est la capacité de notre modèle à détecter tous les incendies.

Rappel=TP/(TP+FN)

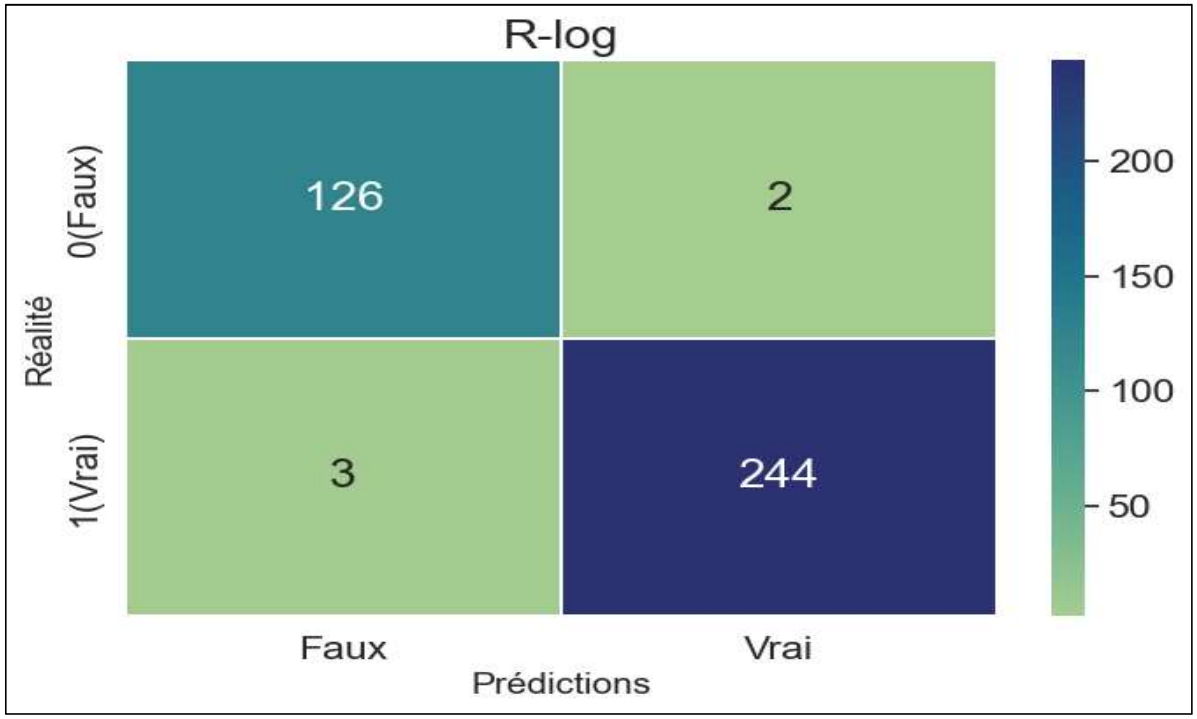
Etape 3

Les algorithmes de prédiction

1_Regression logistique

La régression logistique est un modèle statistique de classification, vise à prédire la probabilité qu'une observation appartienne à une classe. Elle est basée sur la fonction logistique.

Avec cet algorithme on a eu la matrice de confusion suivante:
TN: 126, **FN:** 2, **TP:** 244, **FP:** 3.



	id	Prediction RegLog
0	A_1	False
1	A_2	False
2	A_3	False
3	A_4	True
4	A_5	True

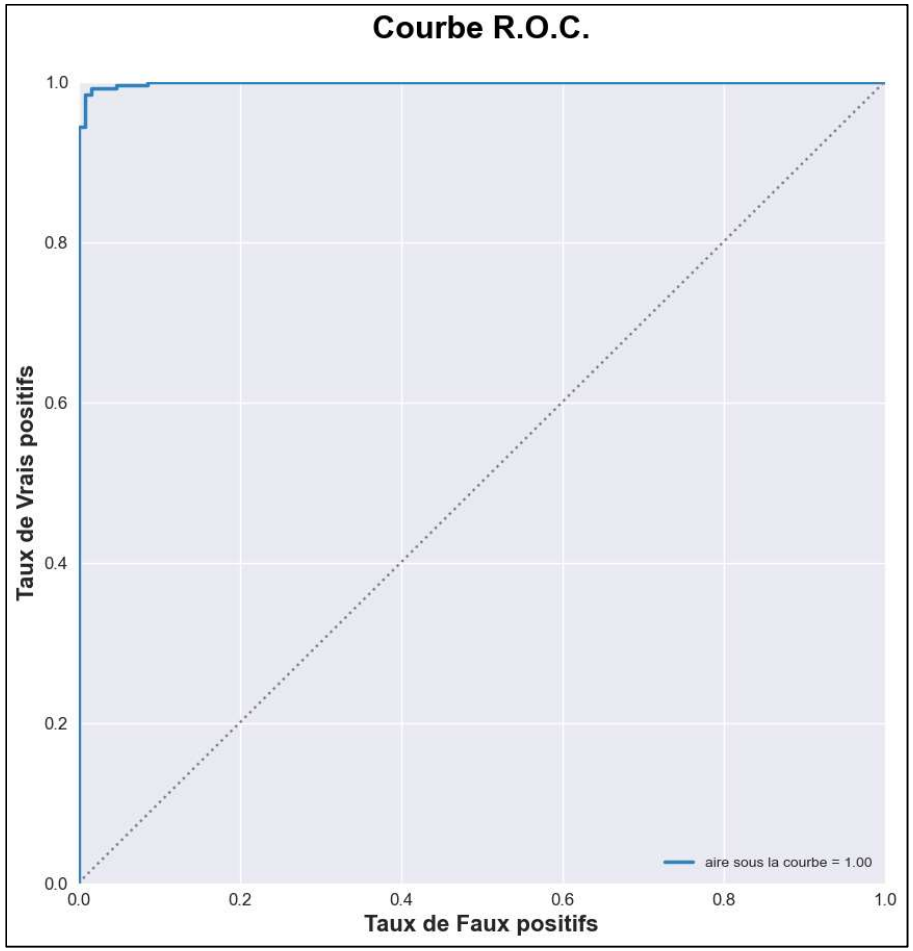
Prédiction par Régression logistique

Precision: 99,19%
Rappel(Recall): 98,79%

Etape 3

Évaluation de l’algorithme de régression logistique

On utilise une courbe ROC:
l'aire sous la courbe = 99,9% très proche de 1, le modèle fonctionne très bien



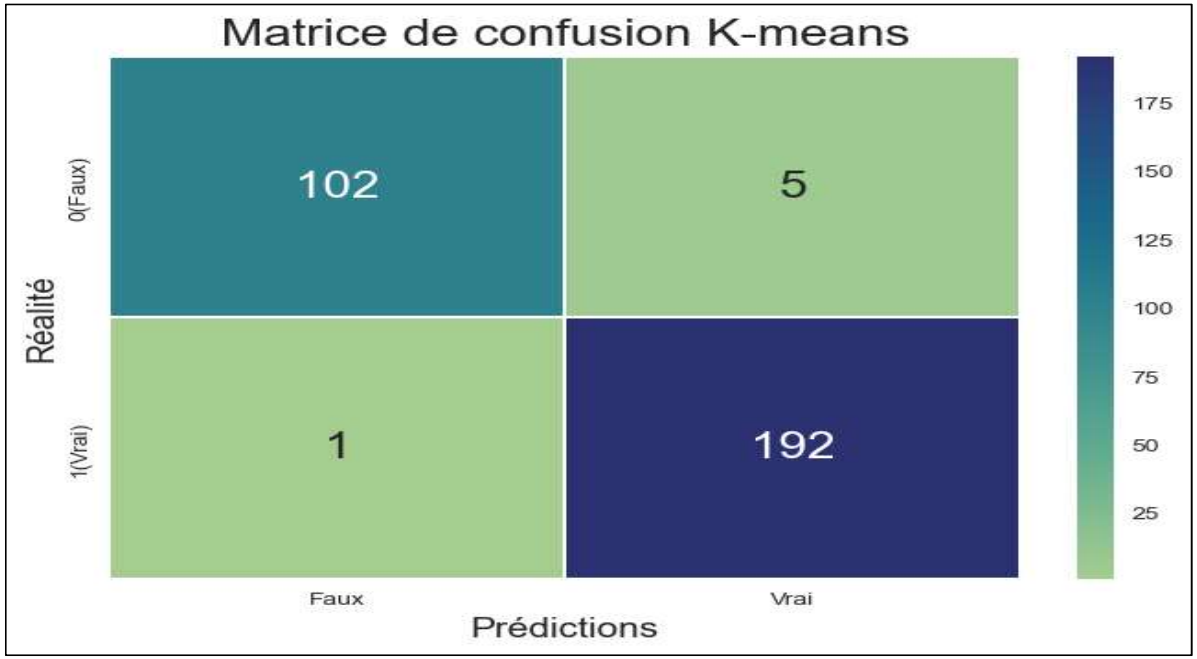
Etape 3

Les algorithmes de prédiction

2_K-Means

K-Means est un algorithme de clustering non supervisé qui divise un ensemble de données en K clusters en minimisant la distance moyenne entre les points d'un cluster et leur centroïde, généralement utilisé pour regrouper des données similaires.

Avec cet algorithme on a eu la matrice de confusion suivante:
TN: 102, **FN:** 5, **TP:** 192, **FP:** 1.



	id	Prédiction K-means
0	A_1	False
1	A_2	False
2	A_3	False
3	A_4	True
4	A_5	True

Prédiction par le modèle K-Means

Precision: 97,50 %
Rappel(Recall): 99,50 %

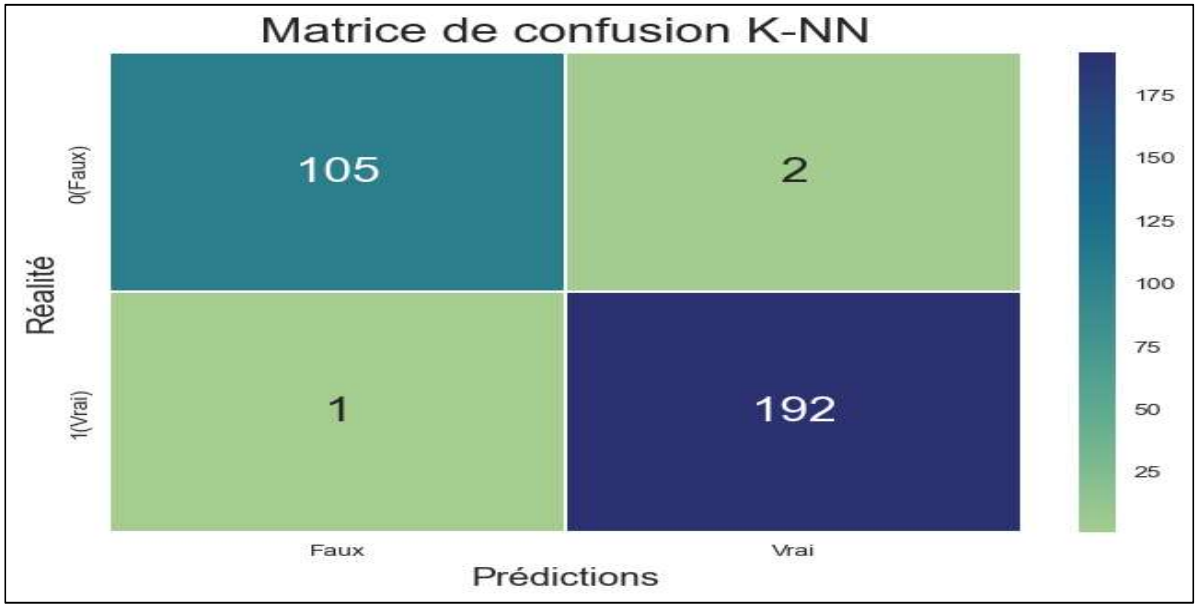
Etape 3

Les algorithmes de prédiction

3_KNN_Classifier

Le modèle de classification KNN est un algorithme d'apprentissage supervisé, il attribue une classe à une observation en fonction des classes majoritaires parmi ses K voisins les plus proches dans l'espace des caractéristiques, il utilise une mesure de distance (distance euclidienne).

Avec cet algorithme on a eu la matrice de confusion suivante:
TN: 105, **FN:** 2, **TP:** 192, **FP:** 2.



	id	Prédiction K-NN
0	A_1	False
1	A_2	False
2	A_3	False
3	A_4	True
4	A_5	True

Prédiction par KNN_Classifier

Precision: 98,97%
Rappel(Recall): 99,48%

Etape 3

Comparaison des 3 modèles

Modèles	Régression logistique	K_Means	KNN_Classifier
Précision	99,19%	97,50 %	98,97%
Rappel (Recall)	98,79%	99,50 %	99,48%

On remarque que le modèle le plus précis est le modèle de **régression logistique** avec une précision de **99.19%** et le modèle **K_Means** est meilleur pour la prédiction des vrais billets avec une justesse de **99.50%** alors que le modèle **KNN_Classifier** montre un équilibre entre sa précision **98,97%** et son rappel **99,48%,** vue que notre mission consiste à identifier automatiquement les vrais des faux billets on fixe notre choix sur l'algorithme des KNN_Means (moins de Faux Négatif **2** et moins de Faux Positif **2**).

Etape 4

Application Finale

id	Prédiction K-nn	Prédiction RegLog	Prob Vrai	Prédiction K-means
B_1	True	True	1.0	True
B_2	False	False	0.0	False
B_3	True	True	1.0	True
B_4	False	False	0.0	False
B_5	False	False	0.0	False