Détection automatique de faux billets

Par OpenClassRooms

Contexte et spécification des données

ONCFM

Organisation publique ayant pour objectif de mettre en place des méthodes d'identification des contrefaçons des billets en euros

Construction d'un algorithme pour détecter les faux billets

Le modèle devra être le plus performant possible

- Faire une analyse descriptive des données
- Combler les valeurs manquantes à l'aide d'une régression linéaire
- Utiliser différentes méthodes de prédiction avec une validation croisée
- Evaluer les modèles via une matrice de confusion
- Construction du programme détection des billets faux

Méthodologie de l'analyse

Outils utilisés

Langage de programmation, conçu pour les analyses de données

Jupyter est un environnement de développement interactif basé sur le web pour les notebooks, le code et les données.

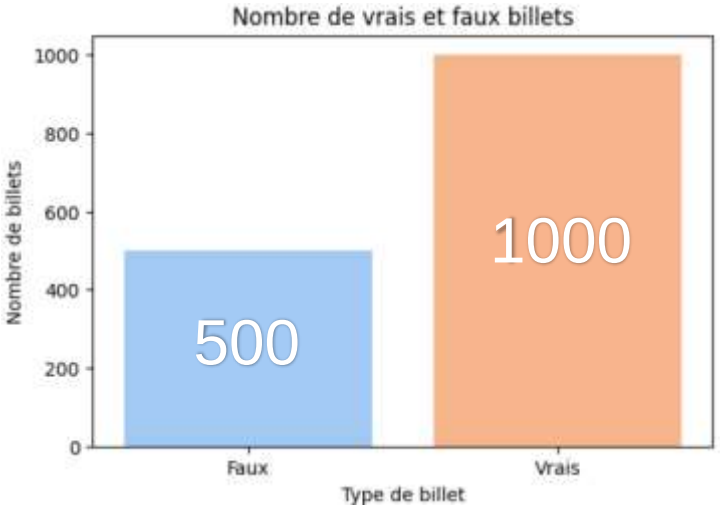
Ressources

Poulet_qui_chante.csv

- is_genuine
- diagonal
- height_left
- height_right
- margin_low
- margin_up
- length

Analyse descriptive des données

Connaitre le nombre de billets Faux / Vrais



Nombre de ligne en doublon

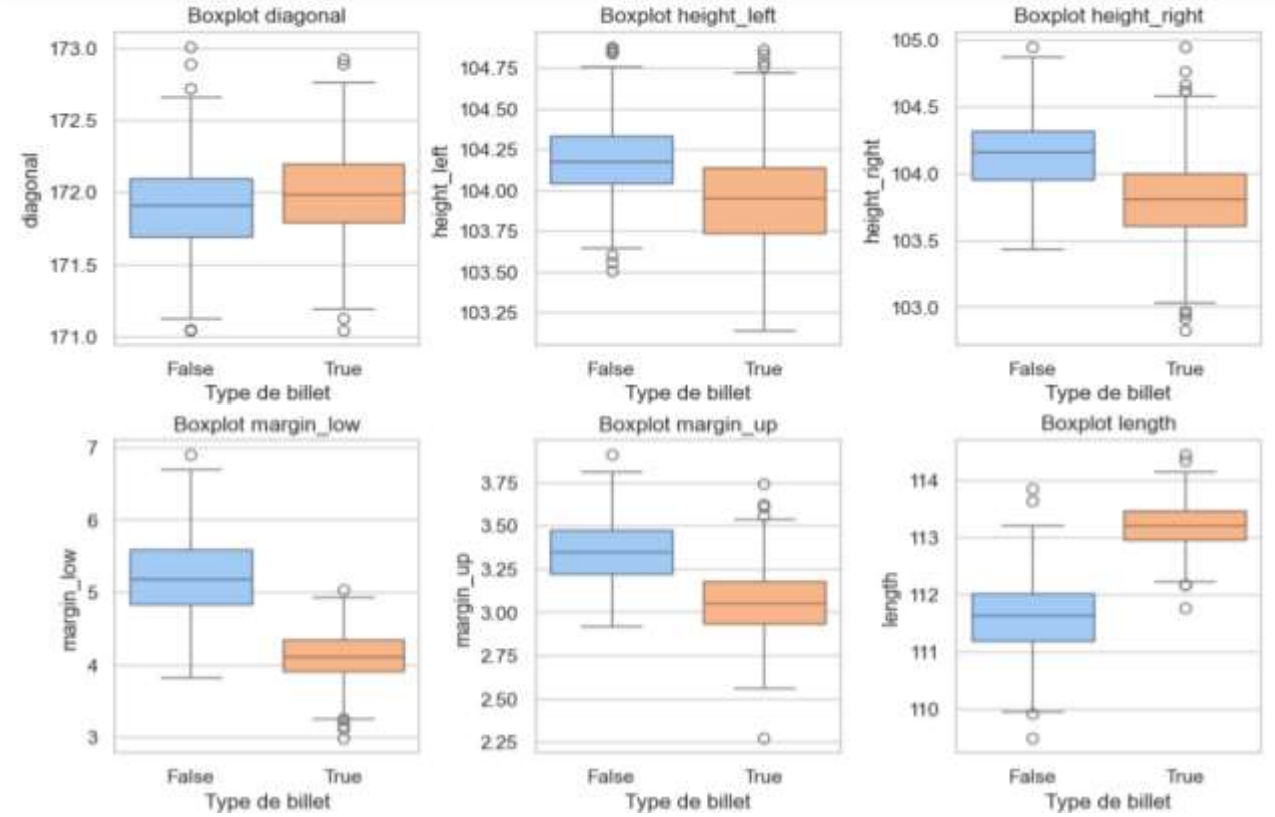
```
#Vérifier si il y a les lignes en doublons dans le dataframe df_billets
# Identification des doublons
df_billets.duplicated().sum()
```

```
0
```

Vérifier les valeurs manquantes

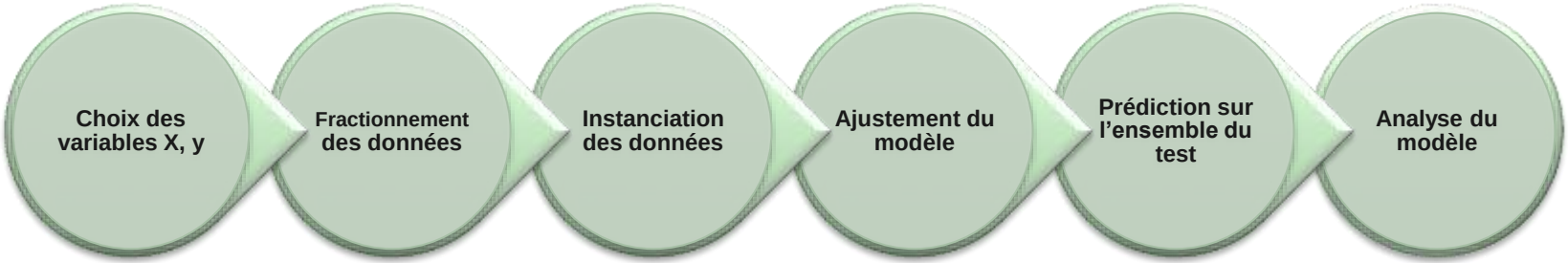
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   is_genuine    1500 non-null   bool   
 1   diagonal      1500 non-null   float64
 2   height_left   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
```

Analyse descriptive des données

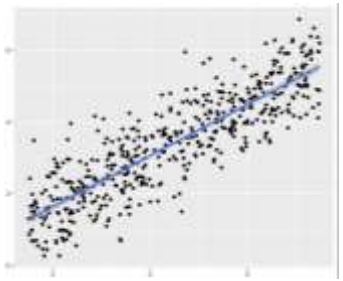


La régression linéaire

On va utiliser la méthode de régression linéaire pour combler les valeurs manquantes



- Variance de la variable dépendante 'margin_low'
- variables indépendantes (diagonal, height_left, …)



- Analyse résiduelle
- R-carré
- Erreur moyenne au carré
- R-carré ajusté

Tableau récapitulatif de régression

Permet d'estimer les coefficients d'une régression linéaire, en minimisant la somme des carrés des écarts entre les valeurs observées et les valeurs prédites par le modèle.

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.477
Model:	OLS	Adj. R-squared:	0.476
Method:	Least Squares	F-statistic:	266.1
Date:	Fri, 05 Jul 2024	Prob (F-statistic):	2.60e-202
Time:	09:03:19	Log-Likelihood:	-1001.3
No. Observations:	1463	AIC:	2015.
Df Residuals:	1457	BIC:	2046.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	22.9948	9.656	2.382	0.017	4.055	41.935
diagonal	-0.1111	0.041	-2.680	0.007	-0.192	-0.030
height_left	0.1841	0.045	4.113	0.000	0.096	0.272
height_right	0.2571	0.043	5.978	0.000	0.173	0.342
margin_up	0.2562	0.064	3.980	0.000	0.130	0.382
length	-0.4091	0.018	-22.627	0.000	-0.445	-0.374

Omnibus:	73.627	Durbin-Watson:	0.674
Prob(Omnibus):	0.000	Jarque-Bera (JB):	95.862
Skew:	0.482	Prob(JB):	1.53e-21
Kurtosis:	3.801	Cond. No.	1.94e+05

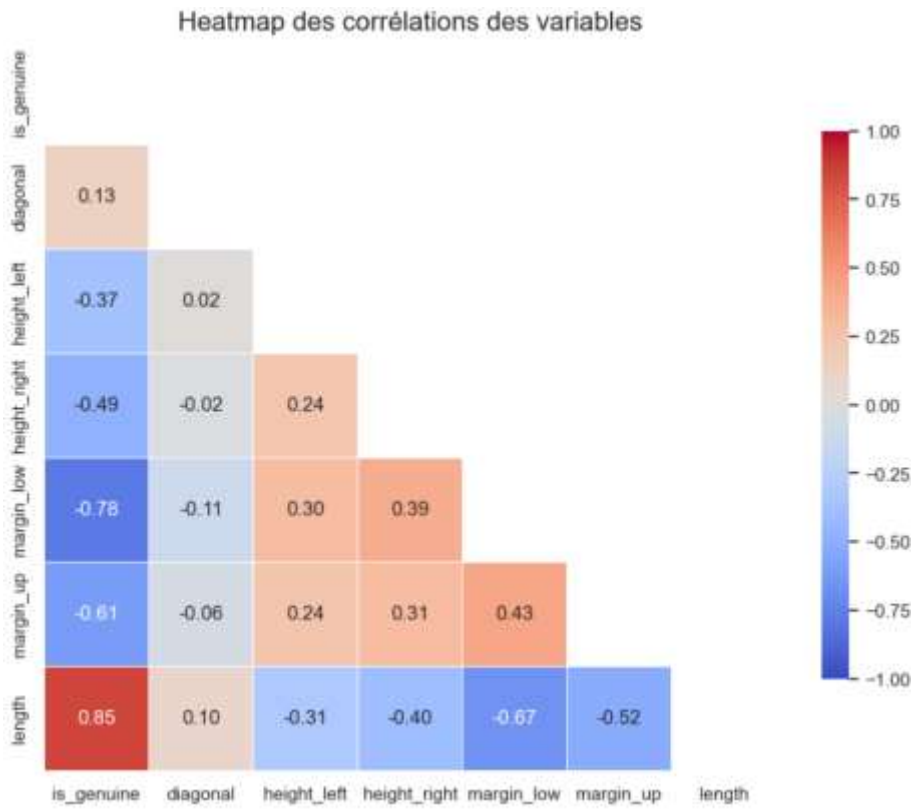
- R^2 = 0.478, 47.8% de la variance X est expliquée par les variables indépendantes Y

- t-statistique mesure le rapport du coefficient à son erreur standard
- Valeur p (P>|t|) mesure la probabilité que le coefficient soit différent de zéro par hasard

Dans ce cas, tous les coefficients sont significatifs avec des valeurs p très faibles !

Les résidus ne suivent pas exactement une distribution normale, ce qui peut influencer les résultats de l'analyse de régression

Corrélation entre les variables

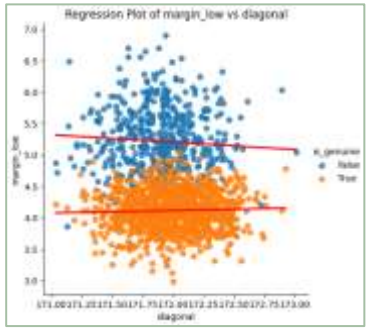


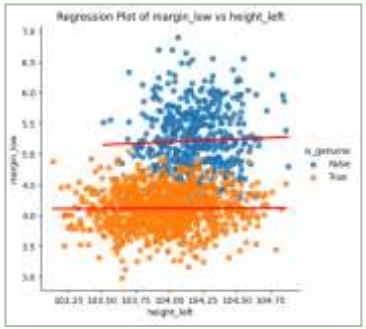
Margin_low est significativement corrélé positivement avec ‘height_right’ et ‘length’

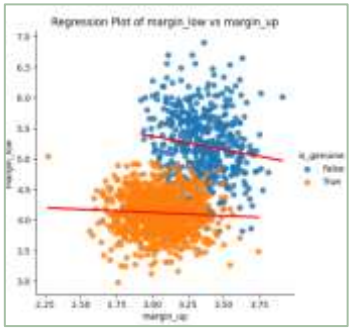
Margin_low est corrélé négativement avec ‘is_genuine’, ce qui signifie que lorsque is_genuine est égale à 1, margin_low a tendance à diminué.

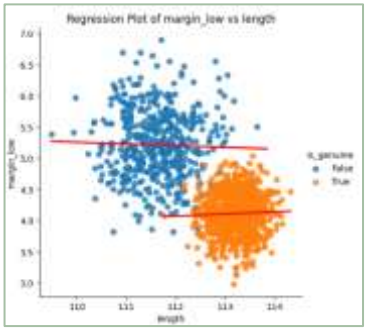
Clustering des variables

Distinctions des groupes par variables (du plus mauvais au meilleur)









Difficile à différencier le vrai du faux billet

La différenciation est facile entre les groupes

Régression linéaire multiple

Choix des variables

```
# Variables explicatives (X) et cible (y)
X = df_selected[['is_genuine', 'diagonal', 'margin_up', 'length', 'height_right', 'height_left']]
y = df_selected['margin_low']
```

Instanciation du modèle

```
logistic_model = LogisticRegression()
```

Ajustement du modèle

```
logistic_model.fit(X_train_scaled, y_train)
```

Prédire sur l'ensemble de test

```
y_pred = logistic_model.predict(X_test_scaled)
```

- Coefficient de détermination R^2: 0.548
- R^2 ajusté : 0.532
- RMSE: 0.179
- MAPE: 0.073

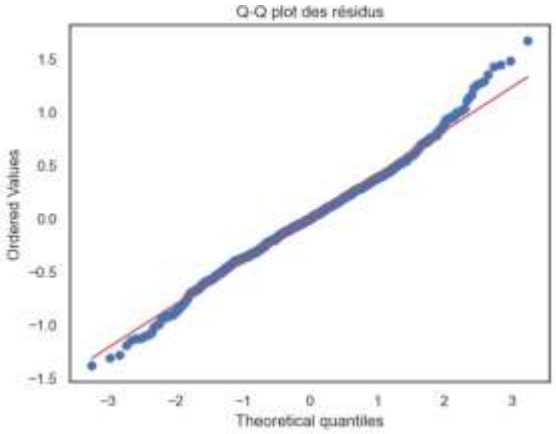
Les valeurs manquantes sont comblées

```
print(df_billets.loc[1463:1470, 'margin_low'])
```

```
1463    4.117107
1464    5.058335
1465    4.763423
1466    4.172048
1467    4.067757
1468    4.204720
1469    4.613890
1470    4.389422
Name: margin_low, dtype: float64
```

Hypothèse de la régression

Normalité des résidus



Données sont presque parfaitement alignées avec la ligne

=

Pas besoin d'ajuster le modèle

Colinéarité des variables

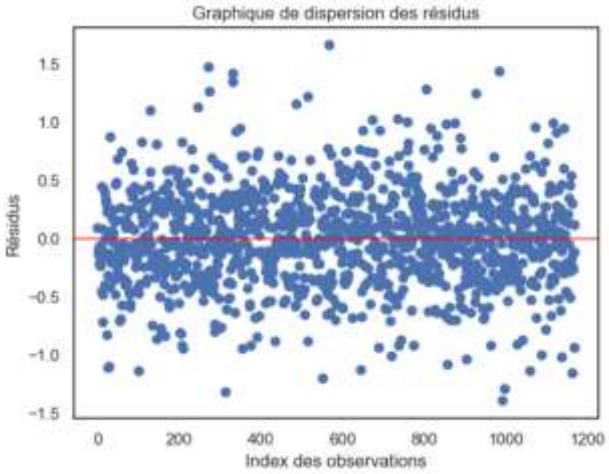
	Variable	VIF
0	const	596789.811796
1	is_genuine	4.738018
2	diagonal	1.028266
3	margin_up	1.599711
4	length	3.613190
5	height_right	1.320821
6	height_left	1.173594

Valeurs en dessous de 5

=

peu probable que la colinéarité entre les variables explicatives affecte significativement les estimations du modèle

Homoscédasticité



- Distribution symétrique des résidus
  - Regroupement autour de 0
    - Pas de tendance nette

LES MODÈLES DE PRÉDICTION

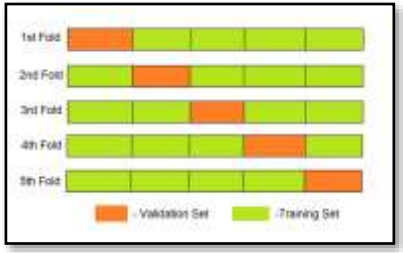
Modèles non-supervisés :

- **Kmeans**

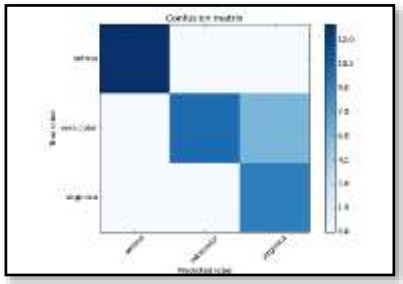
Modèles supervisés :

- **Régression logistique**
- **K-nn**
- **Random Forest**

Comment vérifier la robustesse d'un modèle?



Cross validation sklearn



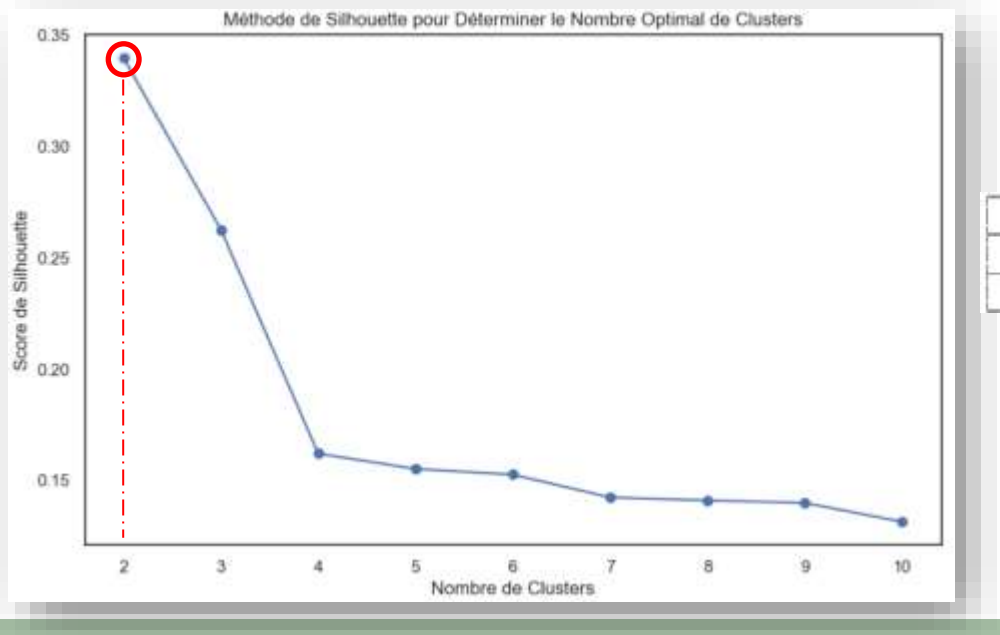
Confusion matrix

Kmeans – méthode non supervisée

Standardiser les données

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Choix du nombre de cluster



Médiane des variables par clusters

Cluster	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	1	171.99	103.95	103.81	4.11058	3.05	113.2
1	0	171.91	104.19	104.16	5.19	3.35	111.63

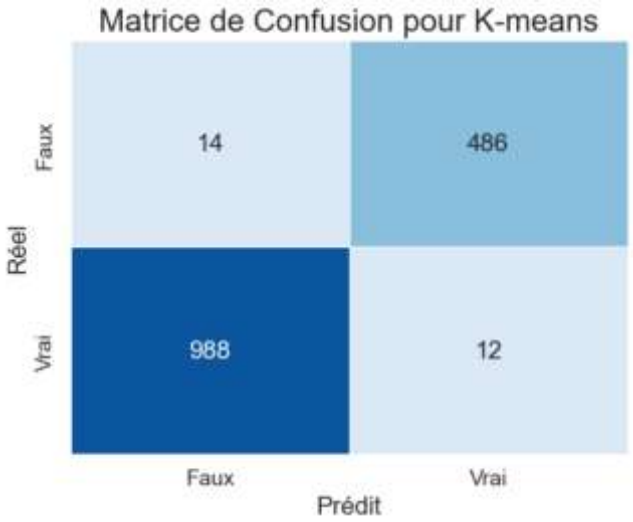
Kmeans - méthode non supervisée

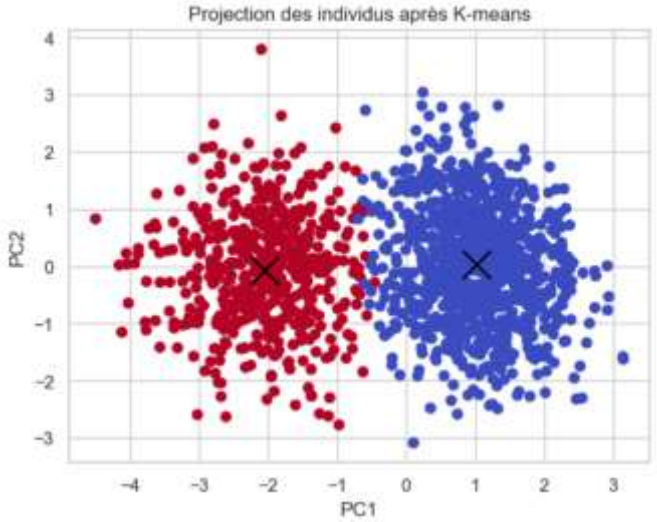
Réalisation du clustering Kmeans

```
kmeans = KMeans(n_clusters=2, init="k-means++"
kmeans.fit(X_scaled)
```

Prédiction des clusters

```
clusters = kmeans.predict(X_scaled)
```





Régression logistique – méthode supervisée

Instanciation du modèle

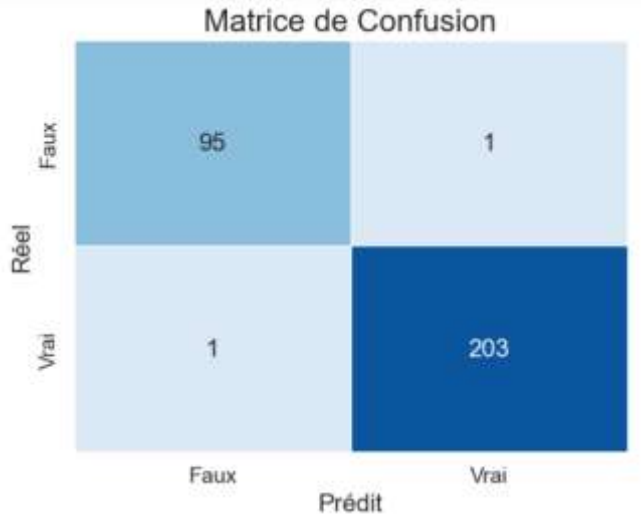
```
logistic_model = LogisticRegression()
```

Ajustement du modèle

```
logistic_model.fit(X_train_scaled, y_train)
```

Prédire sur l'ensemble de test

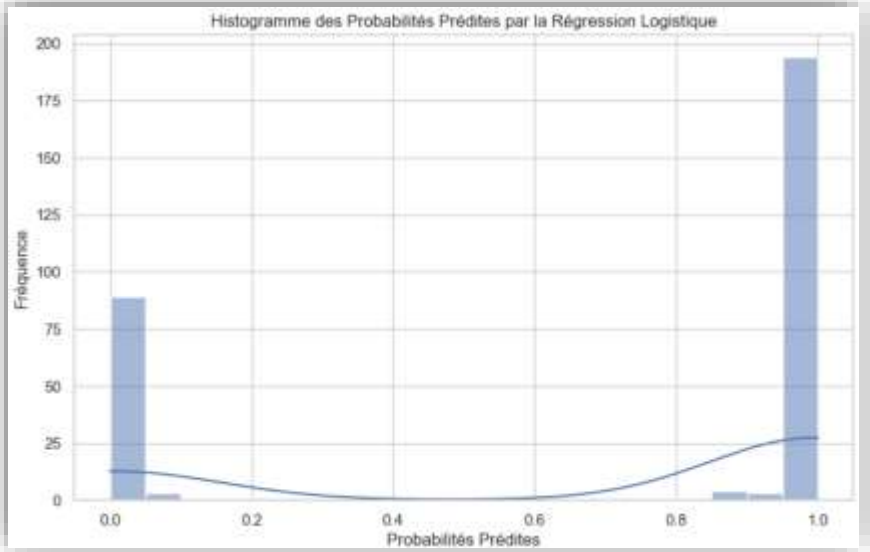
```
y_pred = logistic_model.predict(X_test_scaled)
```

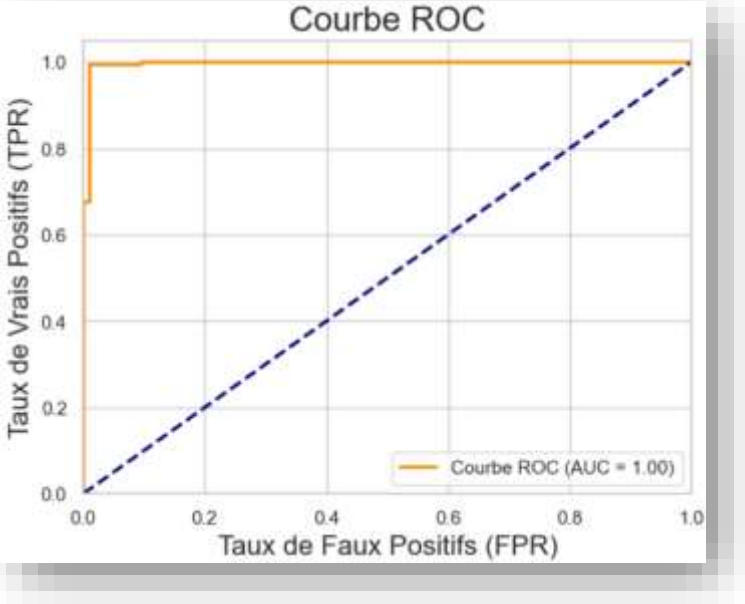


- Validation Croisée - Moyenne des Scores : **0.989**
- Accuracy sur l'ensemble de test: **0.993**

• 1 billet a été considérer comme vrais alors qu'il était faux
• 1 billet a été considérer comme faux alors qu'il était vrai
• 0,66% des données mal prédit !

Régression logistique





Méthode de classification : K-nn

Instanciation du modèle

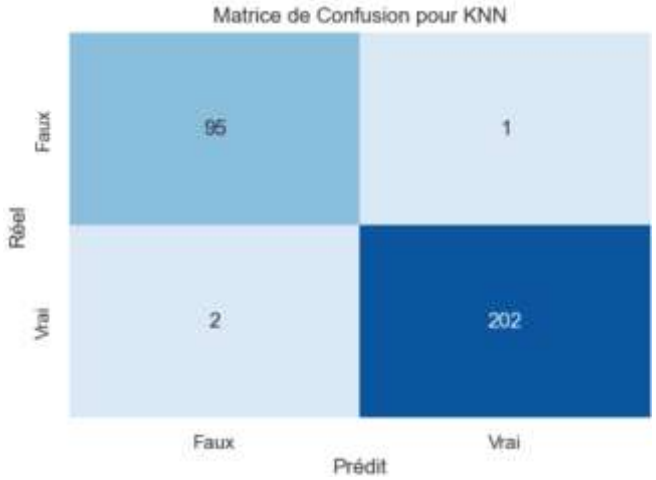
```
knn_cv = KNeighborsClassifier(n_neighbors=12)
```

Ajustement du modèle

```
knn_cv.fit(X_train, y_train)
```

Prédire sur l'ensemble de test

```
y_pred_knn = knn_cv.predict(X_test)
```



- Validation Croisée - Moyenne des Scores : **0.983**
- Accuracy sur l'ensemble de test: **0.990**

• 1 billet a été considérer comme vrais alors qu'il était faux
• 2 billets ont été considérer comme faux alors qu'ils étaient vrais
• 1% des données mal prédit !

Méthode de classification : Random Forest

Instanciation du modèle

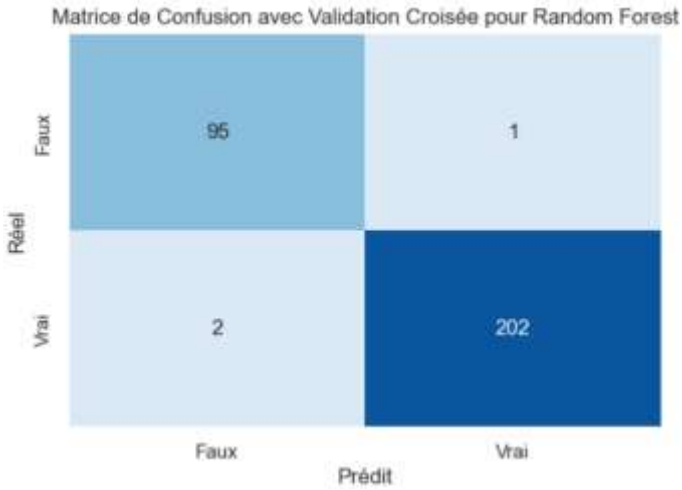
```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
```

Ajustement du modèle

```
rf.fit(X_train, y_train)
```

Prédire sur l'ensemble de test

```
y_pred_rf = rf.predict(X_test)
```



- Validation Croisée - Moyenne des Scores : **0.958**
- Accuracy sur l'ensemble de test: **0.990**

• 1 billet a été considérer comme vrais alors qu'il était faux
• 2 billets ont été considérer comme faux alors qu'ils étaient vrais
• 1% des données mal prédit !

Méthode de prédiction – méthode supervisé

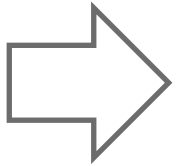
Modèle de prédiction	Adapté pour	Inadapté pour	Choix
Régression logistique	Problèmes de classification binaire Vitesse et simplicité Interprétabilité	Relations non linéaires Problèmes de classification multinomiale	✅
K-means	Faire du clustering Initialisation pour d'autres algorithmes	La classification supervisée Données complexe	❌
K-nn	Petits jeux de données Donnée non linéaire	Grandes quantités de données Données à haute dimensionnalité	❌
Random Forest	Robustesse et performance Donnée complexe	Petits jeux de données Temps de calcul et ressources	❌

Création du programme

Après une longue analyse sur les différents modèles de prédiction.

- Le plus performant : Regression logistique
- Le plus adéquats pour de la classification binaire : Regression linéaire
- Le plus simple et rapide : k-nn et random forest

On va donc sauvegarder le modèle entrainer et l'utiliser pour prédire les données sur de futures jeux de données



```python
import pickle
# Sauvegarder le modèle entraîné et le scaler
with open('logistic_model.pkl', 'wb') as file:
    pickle.dump(logistic_model, file)
with open('scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)
```