Projet 12

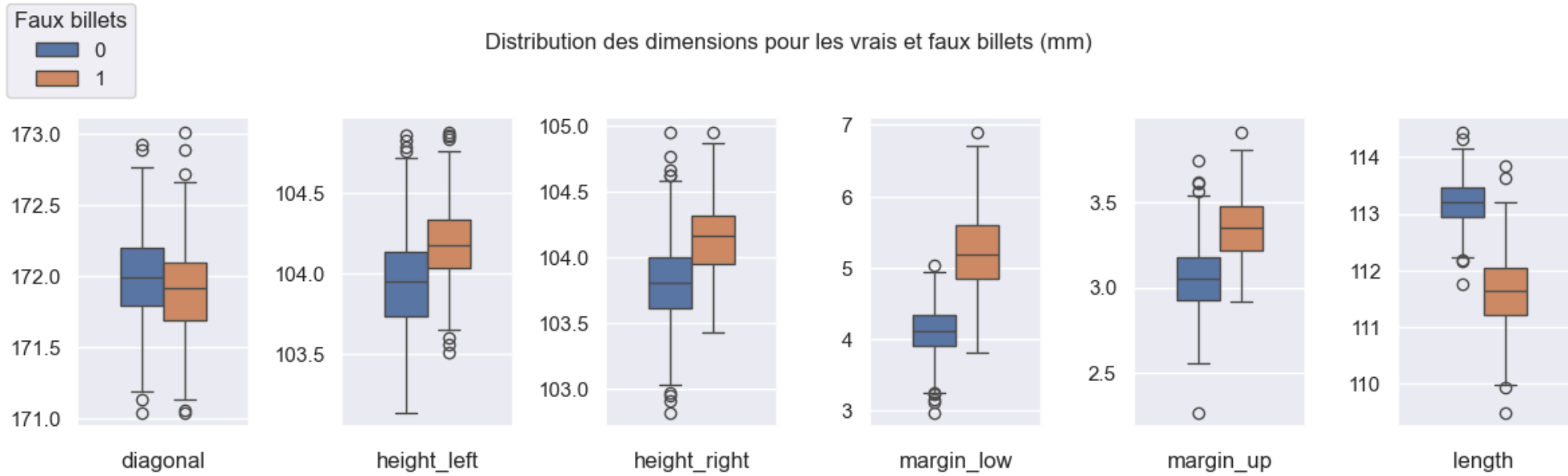
Détectez des faux billets avec Python

Marie G.
Parcours Data Analyst - 19/02/2025

1. 1500 billets et 6 variables de dimension pour identifier les faux billets
2. Un modèle de régression linéaire limité pour déterminer les mesures manquantes
3. Un algorithme de détection des faux-billets performant et modulable

1. 1500 billets et 6 variables de dimension pour identifier les faux billets

Dimensions mesurées sur les billets : 6 variables quantitatives à utiliser pour détecter les faux-billets



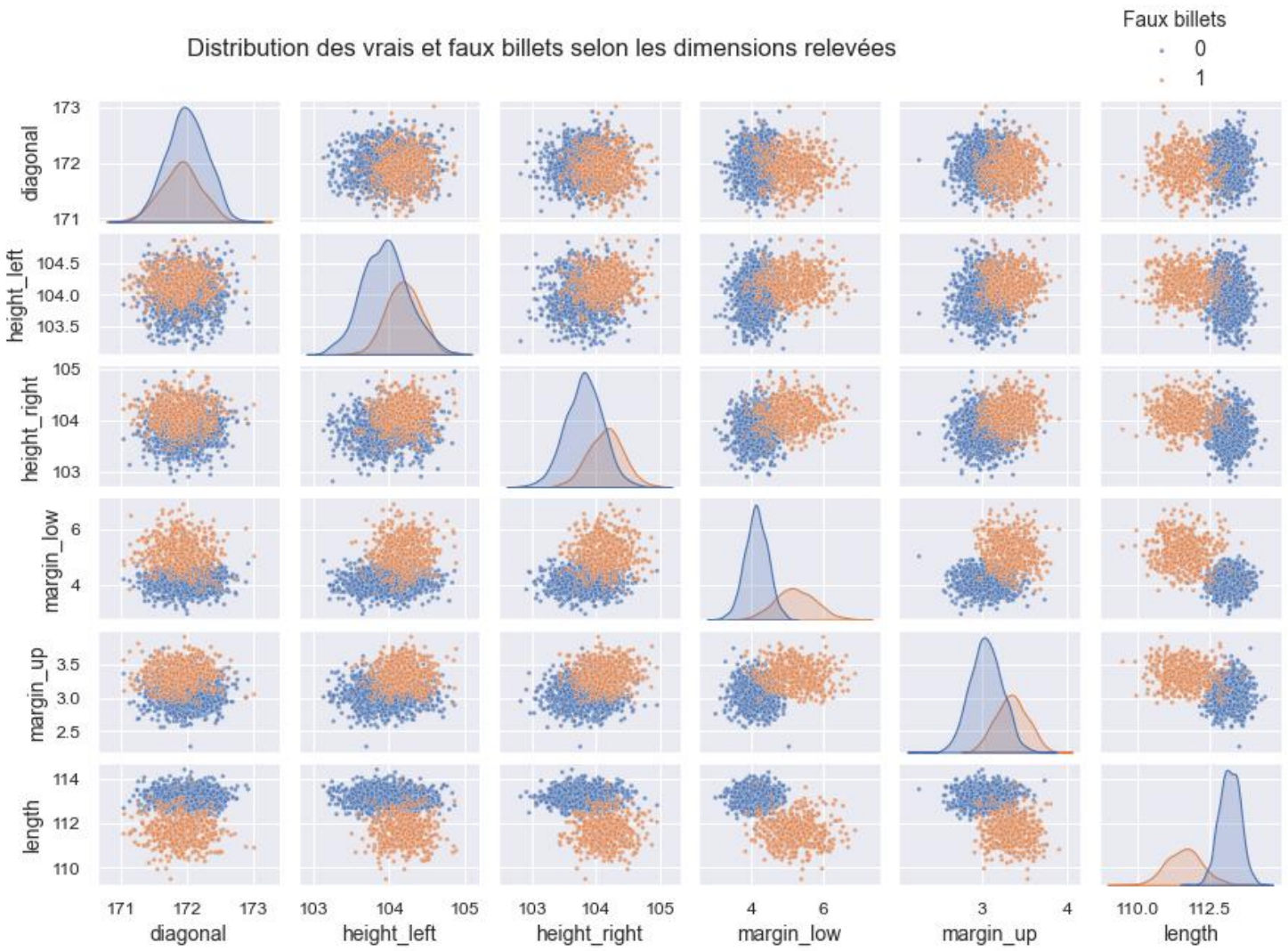
- 6 variables quantitatives continues
- 1 variable *is_genuine* (Nota : remplacée par *is_fake*)
- 500 faux billets
- 1000 vrais billets
- 37 données manquantes de la variable *margin_low* :
  - 29 dans les vrais billets
  - 8 dans les faux billets

On observe des répartitions de dimensions nettement disjointes entre vrais et faux billets

➔ Cette différence va permettre aux modèles d'apprentissage supervisé d'être performants

1. 1500 billets et 6 variables de dimension pour identifier les faux billets

Des variables de dimension semblant indépendantes les unes des autres

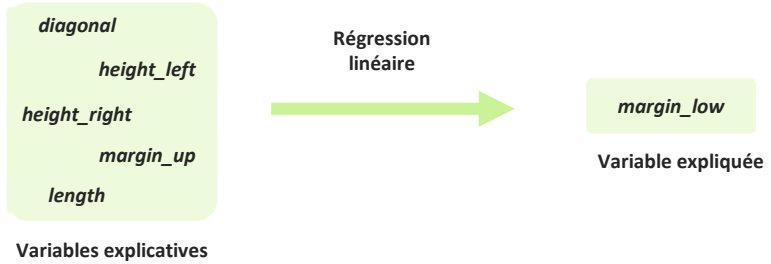


On observe a priori peu de corrélation entre les variables de dimension

2. Un modèle de régression linéaire limité pour déterminer les mesures manquantes

Création et paramétrage des modèles de régression linéaire permettant de compléter les valeurs manquantes *margin_low*

- Les valeurs de *margin_low* étant sensiblement différentes entres les vrais et faux billets, on effectue la régression linéaire séparément sur le jeu de vrais billets et sur le jeu de faux billets
- On crée deux modèles de régression linéaire *ols* de la bibliothèque Python *statsmodels.formula.api,* qu'on entraîne respectivement sur les vrais et sur les faux billets pour lesquels on dispose de la variable *margin_low*



- On identifie les variables explicatives significatives par méthode descendante : on retire successivement les variables non significatives *height_left*, *height_right*, *diagonal* puis *length*, la p_{valeur} du test de Student étant supérieure à 20 % pour toutes ces variables

Caractéristiques du modèle (vrais billets)

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.008
Model:	OLS	Adj. R-squared:	0.003
Method:	Least Squares	F-statistic:	1.544
Date:	mar., 04 févr. 2025	Prob (F-statistic):	0.174
Time:	12:07:40	Log-Likelihood:	-264.37
No. Observations:	971	AIC:	540.7
Df Residuals:	965	BIC:	570.0
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-10.4144	7.928	-1.314	0.189	-25.973	5.145
diagonal	0.0390	0.034	1.141	0.254	-0.028	0.106
height_left	0.0045	0.034	0.132	0.895	-0.062	0.071
height_right	0.0360	0.036	1.012	0.312	-0.034	0.106
margin_up	-0.1071	0.055	-1.939	0.053	-0.216	0.001
length	0.0349	0.029	1.213	0.225	-0.022	0.091

Omnibus:	2.373	Durbin-Watson:	2.068
Prob(Omnibus):	0.305	Jarque-Bera (JB):	2.248
Skew:	-0.112	Prob(JB):	0.325
Kurtosis:	3.071	Cond. No.	1.96e+05

Caractéristiques du modèle (faux billets)

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.027
Model:	OLS	Adj. R-squared:	0.017
Method:	Least Squares	F-statistic:	2.718
Date:	mar., 04 févr. 2025	Prob (F-statistic):	0.0195
Time:	12:07:40	Log-Likelihood:	-399.85
No. Observations:	492	AIC:	811.7
Df Residuals:	486	BIC:	836.9
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	20.2920	20.068	1.011	0.312	-19.139	59.723
diagonal	-0.1187	0.082	-1.455	0.146	-0.279	0.042
height_left	0.1027	0.111	0.924	0.356	-0.116	0.321
height_right	0.0027	0.092	0.029	0.977	-0.178	0.183
margin_up	-0.4357	0.139	-3.144	0.002	-0.708	-0.163
length	-0.0376	0.040	-0.931	0.353	-0.117	0.042

Omnibus:	1.910	Durbin-Watson:	2.032
Prob(Omnibus):	0.385	Jarque-Bera (JB):	1.994
Skew:	0.135	Prob(JB):	0.369
Kurtosis:	2.845	Cond. No.	2.05e+05

La seule variable explicative retenue pour le modèle de régression linéaire est *margin_up* pour les vrais comme pour les faux billets

2. Un modèle de régression linéaire limité pour déterminer les mesures manquantes

Performances sur les vrais billets

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.004
Model:	OLS	Adj. R-squared:	0.003
Method:	Least Squares	F-statistic:	3.558
Date:	mar., 04 févr. 2025	Prob (F-statistic):	0.0596
Time:	12:07:40	Log-Likelihood:	-266.46
No. Observations:	971	AIC:	536.9
Df Residuals:	969	BIC:	546.7
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.4339	0.169	26.272	0.000	4.103	4.765
margin_up	-0.1041	0.055	-1.886	0.060	-0.212	0.004

Omnibus:	2.233	Durbin-Watson:	2.068
Prob(Omnibus):	0.327	Jarque-Bera (JB):	2.114
Skew:	-0.109	Prob(JB):	0.348
Kurtosis:	3.066	Cond. No.	55.8

- Test de Student sur *margin_up* : p_{valeur} = 6 % - on ne peut donc pas rejeter la non-significativité de *margin_up* dans le modèle avec une erreur de 5%.
- A défaut d'une meilleure variable explicative, on se base sur *margin_up* pour établir notre modèle
- Coefficient de détermination : R^2 = 0,004 - le modèle explique donc une partie extrêmement faible des écarts de *margin_low* à la moyenne.

Performances sur les faux billets

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.020
Model:	OLS	Adj. R-squared:	0.018
Method:	Least Squares	F-statistic:	10.15
Date:	mar., 04 févr. 2025	Prob (F-statistic):	0.00154
Time:	12:07:40	Log-Likelihood:	-401.59
No. Observations:	492	AIC:	807.2
Df Residuals:	490	BIC:	815.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.6895	0.463	14.442	0.000	5.779	7.600
margin_up	-0.4397	0.138	-3.186	0.002	-0.711	-0.169

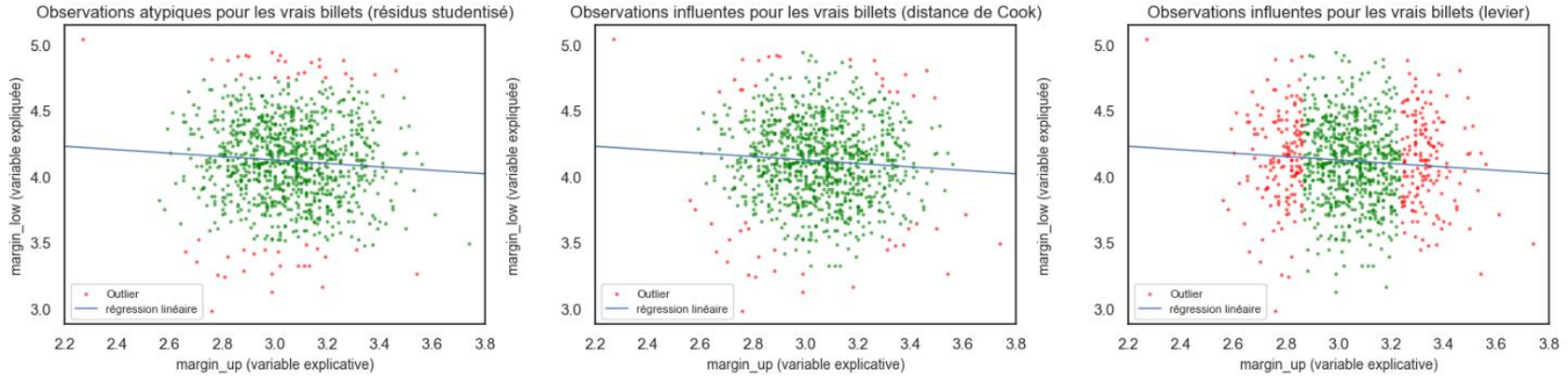
Omnibus:	1.302	Durbin-Watson:	2.030
Prob(Omnibus):	0.521	Jarque-Bera (JB):	1.387
Skew:	0.116	Prob(JB):	0.500
Kurtosis:	2.883	Cond. No.	68.4

- Le test de Student sur *margin_up* donne une p_{valeur} de 0,2 % : on peut donc ici rejeter la non-significativité de *margin_up* dans le modèle
- On obtient cependant un coefficient de détermination R^2 de 0,02 donc le modèle reste très mauvais.

2. Un modèle de régression linéaire limité pour déterminer les mesures manquantes

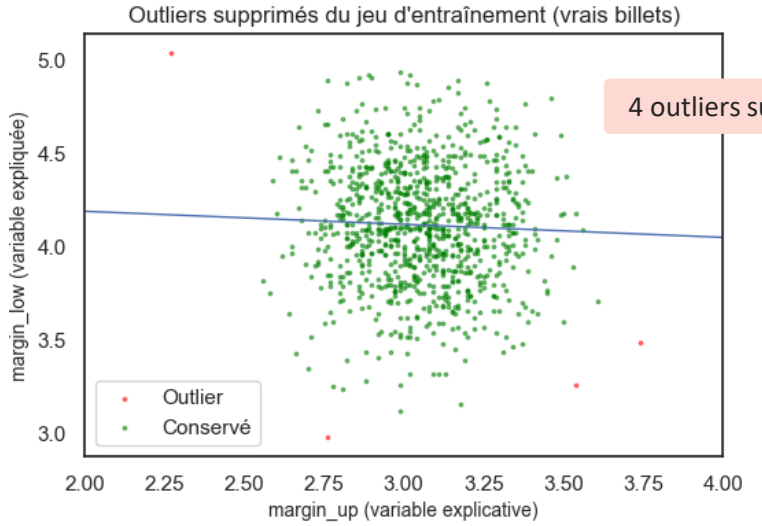
Identification d'outliers

- Recherche d'outliers à partir des mesures des résidus studentisés, de la distance de Cook et des effets de levier :

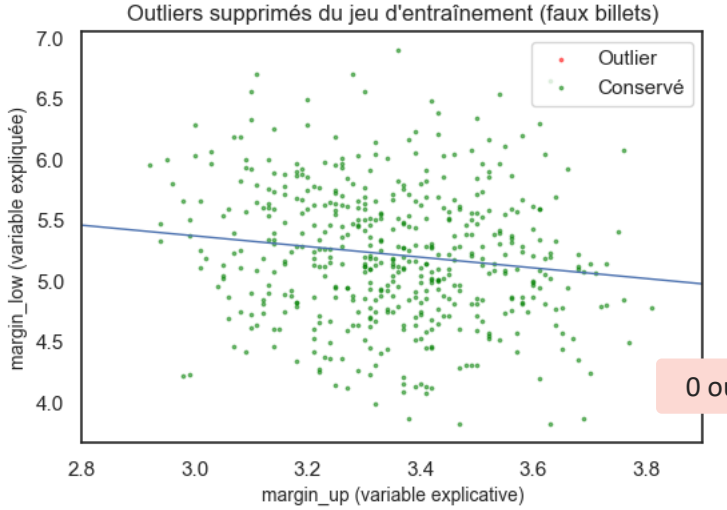


- Observations atypiques ou influentes en rouge pour le jeu de vrais billets

- Après lecture des nuages de points, on retient seulement les outliers suivants, qu'on supprime avant de réentraîner les modèles :



4 outliers supprimés



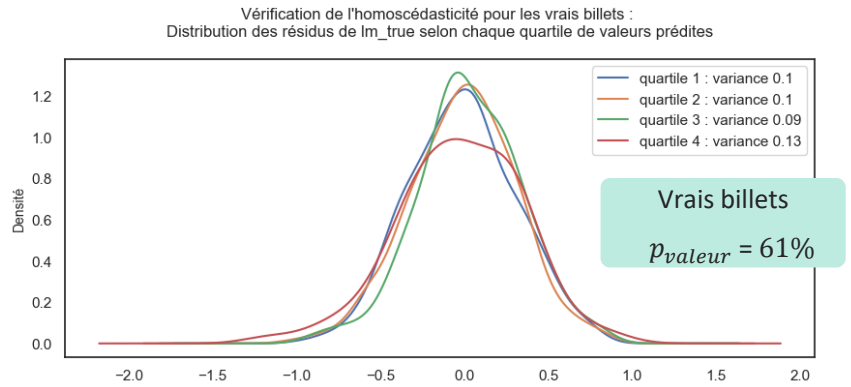
0 outlier supprimé

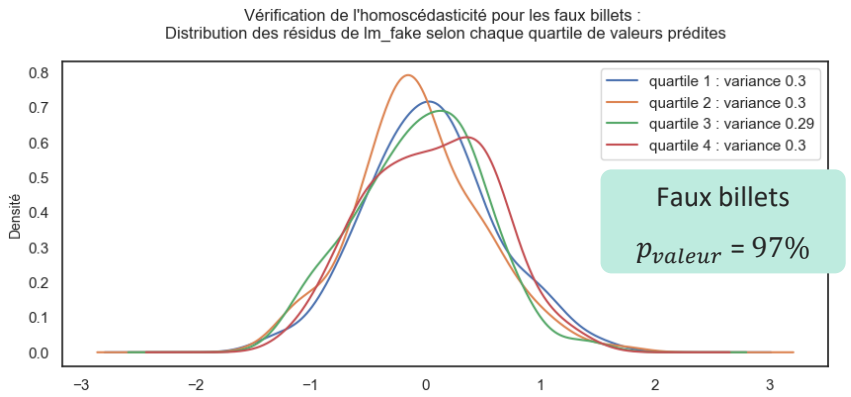
- NB : Après suppression des outliers sur le jeu de vrais billets, R^2 descend de 0,004 à 0,002
- Il sera à peu près aussi juste de remplacer les valeurs manquantes de *margin_low* par la moyenne des valeurs observées

2. Un modèle de régression linéaire limité pour déterminer les mesures manquantes

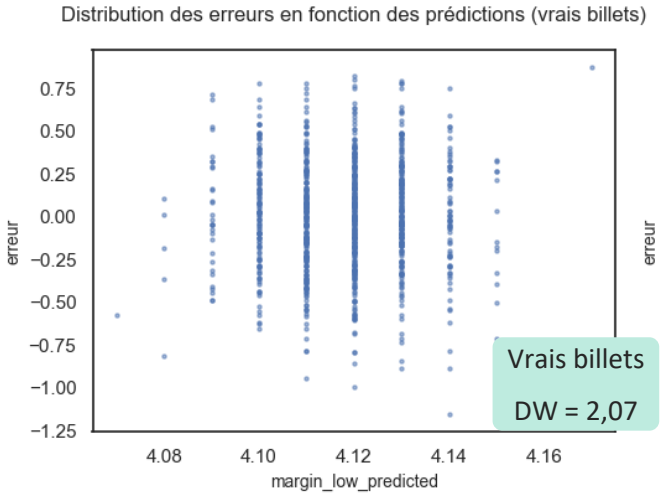
Vérification des hypothèses sur les erreurs

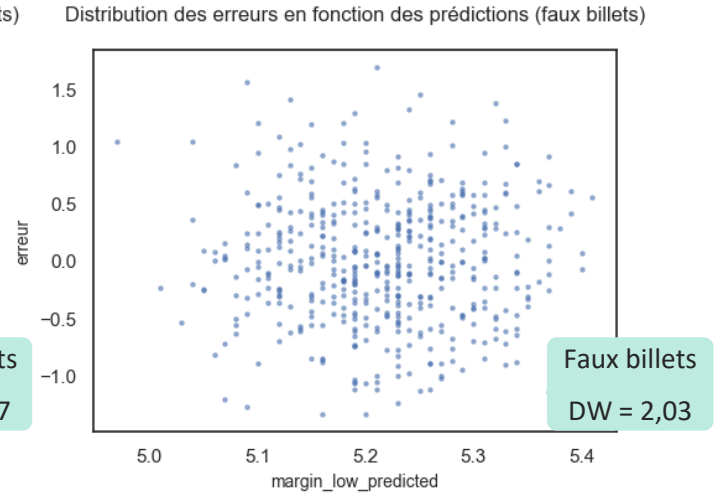
- **Indépendance des variables** vérifiée car une seule variable
- **Homoscédasticité** (égalité des variances) : test de Breusch-Pagan vérifié (p_{valeur} > 5%)





- **Indépendance des erreurs** : test de Durbin-Watson vérifié (valeur proche de 2)





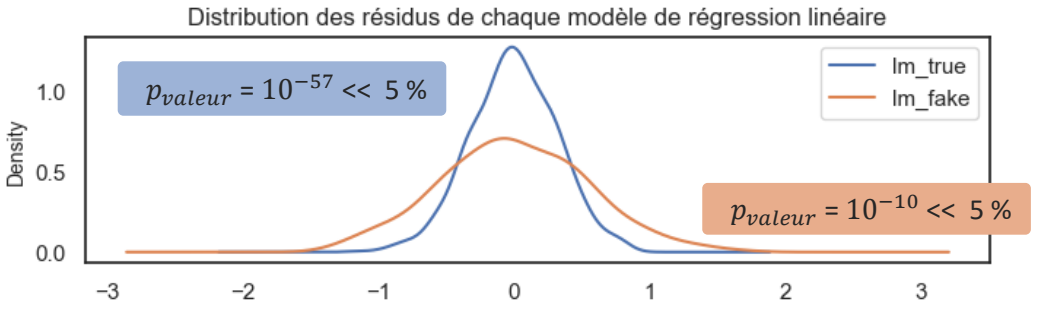
2. Un modèle de régression linéaire limité pour déterminer les mesures manquantes

Vérification des hypothèses sur les erreurs

- **Normalité des erreurs** : test de Kolmogorov-Smirnov non vérifié (p_{valeur} << 5 %)

Le résultat des tests mène à rejeter l'hypothèse de normalité des erreurs.

Cependant les deux échantillons étant suffisamment grands (>> 30) avec 971 vrais billets et 492 faux billets, on peut considérer que le non-respect de l'hypothèse de normalité des erreurs n'est pas gênant pour valider le modèle (bien moins que la faiblesse du coefficient de détermination R^2)



Prédictions des valeurs manquantes

- On peut intégrer dans le jeu de données les 37 valeurs manquantes de *margin_low* calculées par nos deux modèles de régression linéaire.

➔ **Ce jeu va servir à créer l'algorithme de détection des faux billets**

			margin_low	is_fake
XX	XX	XX	XX	0
XX	XX	XX	XX	0
XX	XX	XX	XX	0

Jeu d'entraînement (vrais billets)

			margin_low	is_fake
XX	XX	XX	XX	0
XX	XX	XX	XX	0

Jeu de production (vrais billets)

			margin_low	is_fake
XX	XX	XX	XX	1
XX	XX	XX	XX	1
XX	XX	XX	XX	1

Jeu d'entraînement (faux billets)

			margin_low	is_fake
XX	XX	XX	XX	1
XX	XX	XX	XX	1

Jeu de production (faux billets)

Dataframe *billets_full*

3. Un algorithme de détection des faux-billets performant et modulable

Principe de l'algorithme de détection et de l'analyse des performances

4 Algorithmes testés

- **Régression logistique** : Basée sur la régression linéaire, elle associe à un jeu de variables explicatives numériques une probabilité calculée, ramenée de R dans [0;1] grâce à la réciproque de la fonction logit : $logit(p) = ln(\frac{p}{1-p})$
- **K-means** : Détermine 2 clusters, puis identifie la classe (vrai ou faux billet) de chacun selon ses individus. Il affecte alors à un billet testé la classe du centroïde le plus proche
- **KNN** : Estime la classe d'un billet testé d'après la classe des k plus proches voisins
- **Forêt aléatoire** : Ensemble d'arbres de décision permettant de moyenner les probabilités d'authenticité d'un billet. Chaque arbre de décision divise successivement l'ensemble des individus selon les valeurs prises par chaque variable explicative, pour aboutir à une classe par ensemble de valeurs.

Pour tous les algorithmes

On découpe aléatoirement le jeu de données (1500 billets) en :

- un jeu d'entraînement (*X_train*, *y_train*) pour notre algorithme (80% des observations)
- un jeu de test (*X_test*, *y_test*) (20% des observations)

On compare alors les résultats prédits par l'algorithme à partir de *X* avec les résultats attendus *y*, notamment sur le jeu de test, grâce à une matrice de confusion

Nota : On remplace la variable « is_genuine » (True/False) par « is_fake » (0/1)

De cette manière, on calibre naturellement notre algorithme pour identifier les faux billets, donc les individus positifs

Analyse de la performance des modèles par une matrice de confusion

		Prévision	
		0	1
Vérité	0	Vrai négatif (TN)	Faux positif (FP)
	1	Faux négatif (FN)	Vrai positif (TP)

Définition des scores d'une matrice de confusion

- **Rappel (recall) :** Part des faux billets descellés par l'algorithme
- **Exactitude (accuracy) :** Part de prévisions correctes sur l'ensemble
- **Précision (precision) :** Part de faux billets dans les billets identifiés comme faux
- **Score F1 (f1 score):** Combinaison du rappel et de la précision

$$recall = \frac{TP}{TP + FN}$$

$$accuracy = \frac{TN + TP}{TN + FP + FN + TP}$$

$$precision = \frac{TP}{TP + FP}$$

$$F1\_score = 2 * \frac{precision * recall}{precision + recall}$$

On cherchera avant tout à maximiser le rappel car on souhaite détecter une fraude : on doit laisser passer un minimum de faux billets

On considèrera également la précision, et donc le score F1, qui combine les deux

3. Un algorithme de détection des faux-billets performant et modulable

Performance comparée des modèles retenus

Scores de validation croisée

Un algorithme de validation croisée est utilisé pour moyenner les performances d'un algorithme sur les différentes découpes du jeu complet

Nota : Le K-Means étant un algorithme de classification non supervisée, quelques ajustements sont à réaliser sur la classe *KMeans* de la bibliothèque Python *sklearn* pour pouvoir calculer les mêmes scores que les autres algorithmes et ainsi les comparer

Performance des algorithmes sur le jeu de billets

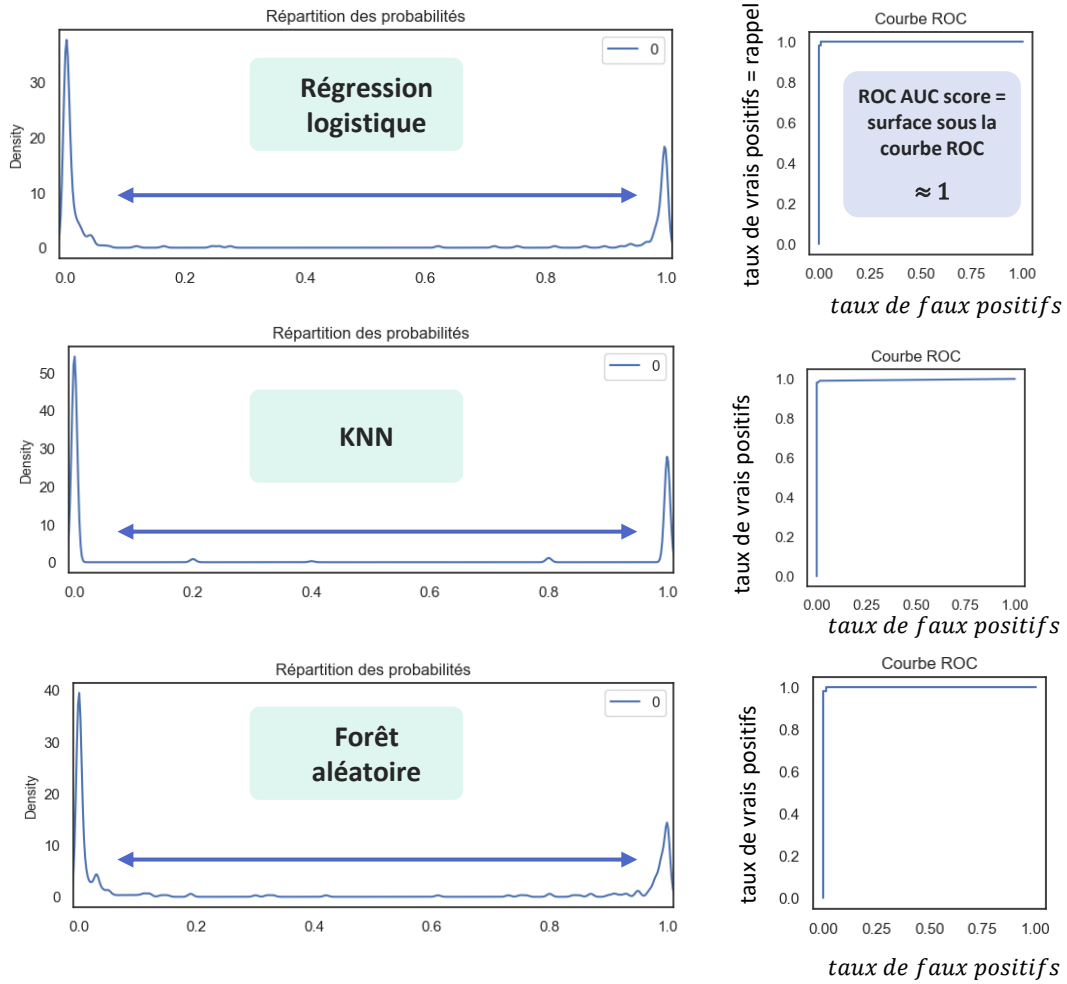
Matrices de confusion | Scores de validation croisée

	Algorithme	Jeu d'entraînement	Jeu de test	recall	f1	precision	accuracy	roc_auc
0	Régression logistique	801 - 4 8 - 387	195 - 0 2 - 103	0.980	0.985	0.990	0.990	0.998
1	KNN	803 - 2 8 - 387	195 - 0 2 - 103	0.974	0.984	0.994	0.989	0.992
2	K-means	803 - 2 17 - 378	195 - 0 5 - 100	0.962	0.979	0.996	0.986	0.980
3	Forêt aléatoire	805 - 0 0 - 395	194 - 1 2 - 103	0.982	0.988	0.994	0.992	0.999

La forêt aléatoire donne les meilleurs résultats en validation croisée, suivie de très près par la régression logistique puis par KNN, et enfin par le K-Means

Dans l'ensemble, les résultats sont proches de 100 % de prédictions correctes : **il est donc compliqué de différencier clairement les performances entre algorithmes et d'améliorer les scores**

Répartition nette des probabilités prédites sur le jeu de test ⇒ des modèles fiables



3. Un algorithme de détection des faux-billets performant et modulable

Choix des variables explicatives

Analyse des variables explicatives du modèle de régression logistique

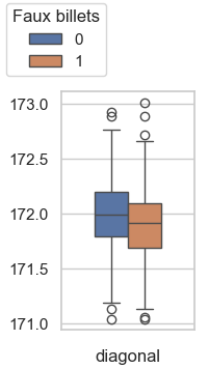
Logit Regression Results

Dep. Variable:	is_fake	No. Observations:	1200
Model:	Logit	Df Residuals:	1194
Method:	MLE	Df Model:	5
Date:	mar., 04 févr. 2025	Pseudo R-squ.:	0.9518
Time:	15:40:36	Log-Likelihood:	-36.627
converged:	True	LL-Null:	-760.30
Covariance Type:	nonrobust	LLR p-value:	7.540e-311

	coef	std err	z	P>|z|	[0.025	0.975]
diagonal	0.6444	0.789	0.816	0.414	-0.903	2.191
height_left	1.7168	1.082	1.587	0.113	-0.404	3.837
height_right	2.9035	1.060	2.739	0.006	0.826	4.981
margin_low	5.8777	1.043	5.637	0.000	3.834	7.921
margin_up	9.3322	2.248	4.151	0.000	4.925	13.739
length	-5.7609	0.929	-6.202	0.000	-7.581	-3.940

La variable *diagonal* n'apparaît pas comme significative
(p_{valeur} du test de Student = 41 %)

On peut constater graphiquement la faible indépendance entre les valeurs de *diagonal* sur les vrais et faux billets :



- Après retrait de la variable explicative *diagonal*, toutes les variables apparaissent comme significatives (p_{valeur} entre 0 et 3,5 %) :

	coef	std err	z	P>|z|	[0.025	0.975]
height_left	2.0845	0.991	2.104	0.035	0.143	4.026
height_right	3.2045	1.004	3.190	0.001	1.236	5.173
margin_low	5.6674	0.989	5.728	0.000	3.728	7.607
margin_up	9.5050	2.250	4.224	0.000	5.094	13.916
length	-5.3914	0.759	-7.106	0.000	-6.879	-3.904

- Les performances du modèle obtenue par des matrices de confusion apparaissent très similaires en testant le modèle avec et sans *diagonal*

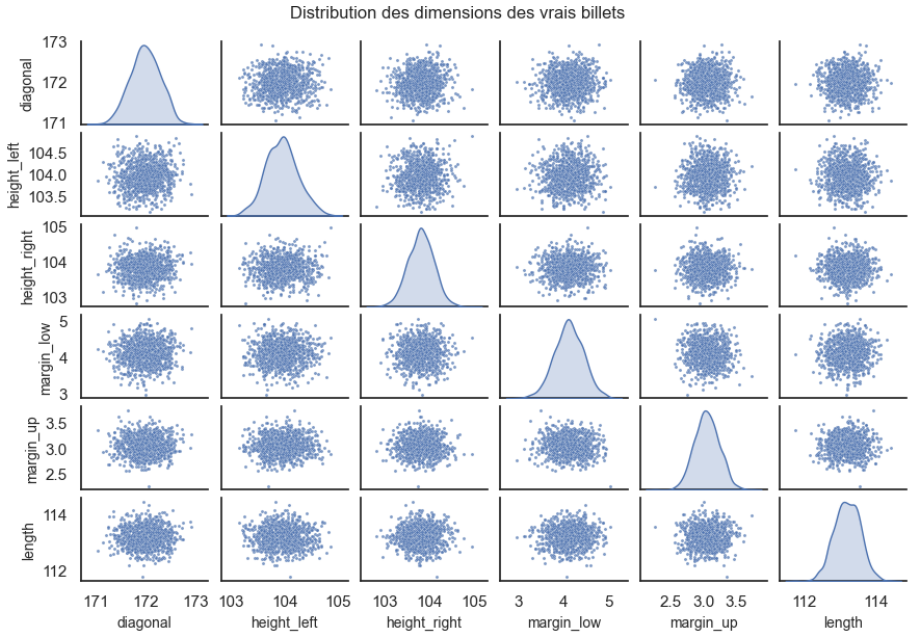
	Algorithme	Jeu d'entraînement	Jeu de test	recall	f1	precision	accuracy	roc_auc
0	Régression logistique avec diagonal	801 - 4 8 - 387	195 - 0 2 - 103	0.980	0.985	0.990	0.990	0.998
1	Régression logistique sans diagonal	801 - 4 7 - 388	195 - 0 2 - 103	0.980	0.985	0.990	0.990	0.998
2	KNN avec diagonal	803 - 2 8 - 387	195 - 0 2 - 103	0.974	0.984	0.994	0.989	0.992
3	KNN sans diagonal	803 - 2 9 - 386	195 - 0 2 - 103	0.978	0.986	0.994	0.991	0.995
4	K-means avec diagonal	803 - 2 17 - 378	195 - 0 5 - 100	0.962	0.979	0.996	0.986	0.980
5	K-means sans diagonal	803 - 2 20 - 375	195 - 0 6 - 99	0.952	0.973	0.996	0.983	0.975
6	Forêt aléatoire avec diagonal	805 - 0 0 - 395	194 - 1 2 - 103	0.982	0.988	0.994	0.992	0.999
7	Forêt aléatoire sans diagonal	805 - 0 0 - 395	194 - 1 2 - 103	0.982	0.988	0.994	0.992	0.999

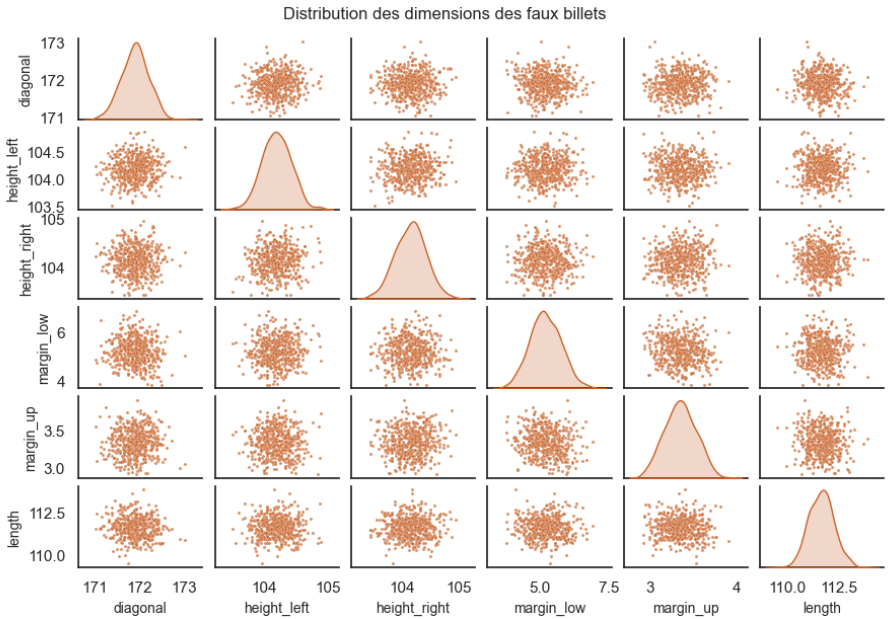
On supprime donc *diagonal* des variables explicatives pour tous nos modèles

Les variables explicatives étant fixées, on détermine les paramètres du KNN (nombre de voisins = 6) et de la forêt aléatoire (profondeur maximale = 6) grâce à un algorithme de recherche des valeurs optimales

3. Un algorithme de détection des faux-billets performant et modulable

Recherche d'outliers





Quelques individus s'écartent des nuages de points dans les deux matrices de dispersion et pourraient être considérés comme outliers.

On teste l'influence sur les modèles de la suppression d'outliers du jeu d'entraînement, selon 3 méthodes d'identification :

- z-score > 3 sur au moins une dimension : 18 outliers identifiés
- Test de Grubbs : aucun outlier identifié
- Distance de Cook : 31 outliers identifiés

Dans les tests, les outliers ne serviront pas à entraîner les modèles mais seront testés en sus du jeu de test

Les scores de validation croisée ne prenant pas en compte les performances sur les outliers, on s'appuiera donc plutôt sur les matrices de confusion pour comparer les modèles

3. Un algorithme de détection des faux-billets performant et modulable

Synthèse des performances

1. Performance des algorithmes après un simple entraînement

	Algorithme	Jeu d'entraînement	Jeu de test	Ensemble	recall	f1	precision	accuracy	roc_auc
0	Régression logistique	801 - 4 8 - 387	195 - 0 2 - 103	996 - 4 10 - 490	0.980	0.985	0.990	0.990	0.998
1	KNN	803 - 2 8 - 387	195 - 0 2 - 103	998 - 2 10 - 490	0.974	0.984	0.994	0.989	0.992
2	K-means	803 - 2 17 - 378	195 - 0 5 - 100	998 - 2 22 - 478	0.962	0.979	0.996	0.986	0.980
3	Forêt aléatoire	805 - 0 0 - 395	194 - 1 2 - 103	999 - 1 2 - 498	0.982	0.988	0.994	0.992	0.999

2. Performance après suppression de *diagonal* et paramétrage de KNN et RF

	Algorithme	Jeu d'entraînement	Jeu de test	Ensemble	recall	f1	precision	accuracy	roc_auc
0	Régression logistique	801 - 4 7 - 388	195 - 0 2 - 103	996 - 4 9 - 491	0.980	0.985	0.990	0.990	0.998
1	KNN	803 - 2 9 - 386	195 - 0 2 - 103	998 - 2 11 - 489	0.978	0.986	0.994	0.991	0.995
2	K-means	803 - 2 20 - 375	195 - 0 6 - 99	998 - 2 26 - 474	0.952	0.973	0.996	0.983	0.975
3	Forêt aléatoire	805 - 0 0 - 395	194 - 1 2 - 103	999 - 1 2 - 498	0.982	0.988	0.994	0.992	0.999

3.a. Performance après écart des outliers définis par z-score

	Algorithme	Jeu d'entraînement	Jeu de test	Outliers	Ensemble	recall	f1	precision	accuracy	roc_auc
0	Régression logistique	798 - 2 6 - 379	195 - 1 3 - 98	3 - 1 0 - 14	996 - 4 9 - 491	0.980	0.986	0.992	0.991	0.998
1	KNN	799 - 1 7 - 378	195 - 1 4 - 97	4 - 0 0 - 14	998 - 2 11 - 489	0.977	0.985	0.994	0.991	0.995
2	K-means	799 - 1 16 - 369	195 - 1 7 - 94	4 - 0 0 - 14	998 - 2 23 - 477	0.961	0.978	0.996	0.986	0.979
3	Forêt aléatoire	800 - 0 0 - 385	195 - 1 3 - 98	4 - 0 0 - 14	999 - 1 3 - 497	0.984	0.989	0.994	0.993	0.999

3.b. Performance après écart outliers définis par distance de Cook

	Algorithme	Jeu d'entraînement	Jeu de test	Outliers	Ensemble	recall	f1	precision	accuracy	roc_auc
0	Régression logistique	794 - 0 3 - 378	191 - 0 0 - 103	11 - 4 10 - 6	996 - 4 13 - 487	0.996	0.998	1.000	0.999	1.000
1	KNN	794 - 0 2 - 379	191 - 0 0 - 103	13 - 2 10 - 6	998 - 2 12 - 488	0.998	0.999	1.000	0.999	1.000
2	K-means	794 - 0 10 - 371	191 - 0 4 - 99	13 - 2 13 - 3	998 - 2 27 - 473	0.981	0.991	1.000	0.994	0.991
3	Forêt aléatoire	794 - 0 0 - 381	191 - 0 0 - 103	13 - 2 8 - 8	998 - 2 8 - 492	0.998	0.997	0.994	0.998	1.000

Observations

- Pas d'amélioration des performances après paramétrage des modèles ni après suppression des observations éloignes (z-score)
- Dégradation nette des performances après suppression des observations influentes (distance de Cook), due à de mauvaises performances sur les outliers

On conserve le paramétrage des modèles qui assure une meilleure robustesse des modèles mais on ne supprime aucun outlier du jeu de billets

3. Un algorithme de détection des faux-billets performant et modulable

Amélioration du rappel par l'ajustement du seuil de probabilité

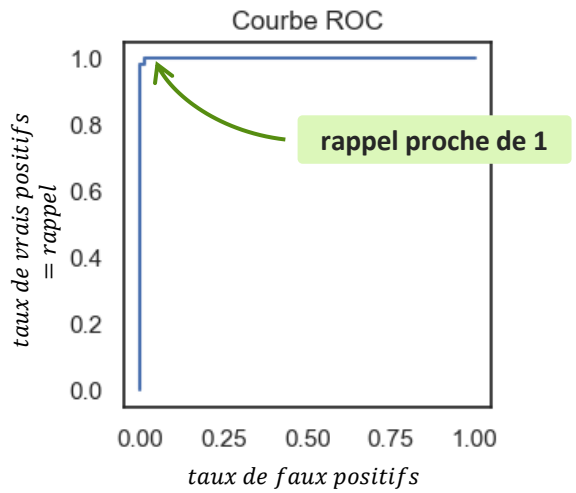
Afin d'obtenir une détection de faux billets proche de 100 %, on peut moduler le seuil de probabilité définissant un individu comme positif ou négatif.

On définit le résultat du modèle par :

- 1 (faux billet) si $p > seuil$
- 0 (vrai billet) si $p < seuil$

Par défaut, ce seuil est de 0,5. Un seuil bas permettra de retenir même les billets qui ont une faible probabilité d'être faux, donc de maximiser le rappel

On retient la forêt aléatoire qui donne le meilleur score ROC-AUC (99,9%) donc les meilleures possibilités d'améliorer le rappel en limitant la dégradation des autres indicateurs



Analyse les performances du modèle de forêt aléatoire modifié selon différents seuils de probabilité de 0 à 0,28

	Seuil	recall	f1	precision	accuracy	roc_auc
0	0.000	1.000	0.500	0.333	0.333	0.999
1	0.020	1.000	0.870	0.774	0.883	0.999
2	0.040	1.000	0.912	0.835	0.939	0.999
3	0.060	0.998	0.934	0.875	0.949	0.999
4	0.080	0.998	0.940	0.893	0.956	0.999
5	0.100	0.998	0.954	0.915	0.969	0.999
6	0.120	0.996	0.959	0.925	0.969	0.999
7	0.140	0.996	0.960	0.929	0.973	0.999
8	0.160	0.996	0.968	0.937	0.977	0.999
9	0.180	0.994	0.966	0.940	0.979	0.999
10	0.200	0.994	0.969	0.944	0.977	0.999
11	0.220	0.994	0.972	0.951	0.981	0.999
12	0.240	0.992	0.976	0.953	0.982	0.999
13	0.260	0.992	0.973	0.962	0.983	0.999
14	0.280	0.994	0.980	0.956	0.983	0.999

On peut retenir un seuil de 16%

Il permet d'obtenir un rappel de 99,6 % et une précision de 94 %

- 99,6 % des faux billets seront identifiés
- 6 % des billets identifiés comme faux le seront à tort

3. Un algorithme de détection des faux-billets performant et modulable

Performances et durée d'exécution

Afin d'évaluer le temps d'exécution de nos algorithmes de détection de faux-billets, on crée 4 jeux de production de tailles différentes (100 à 1 million de billets), dont les variables dimensions sont générées par des variables aléatoires de lois normales de variance et de moyenne égales à celle du jeu d'entraînement

Performance de chaque algorithme : scores et durée d'exécution (secondes)

Algorithme	recall	f1	precision	accuracy	roc_auc	100 billets	10 000 billets	100 000 billets	1 000 000 billets
Régression logistique	0.980	0.985	0.990	0.990	0.998	0.0000	0.0093	0.0065	0.0551
KNN	0.978	0.987	0.996	0.991	0.995	0.0102	0.4343	4.2790	42.3193
K-means	0.952	0.973	0.996	0.983	0.975	0.0040	0.0067	0.0377	0.3250
Forêt aléatoire	0.982	0.988	0.994	0.991	0.999	0.0136	0.0375	0.3933	3.9245
Forêt aléatoire avec seuil	0.996	0.963	0.935	0.979	0.999	0.0000	0.0530	0.3997	3.8163

Durée d'exécution des algorithmes pour traiter 1 million de billets :

- KNN : 42 secondes
- Forêt aléatoire avec ou sans modification du seuil : 4 secondes
- K-Means : 0,3 seconde
- Régression logistique : 0,06 seconde

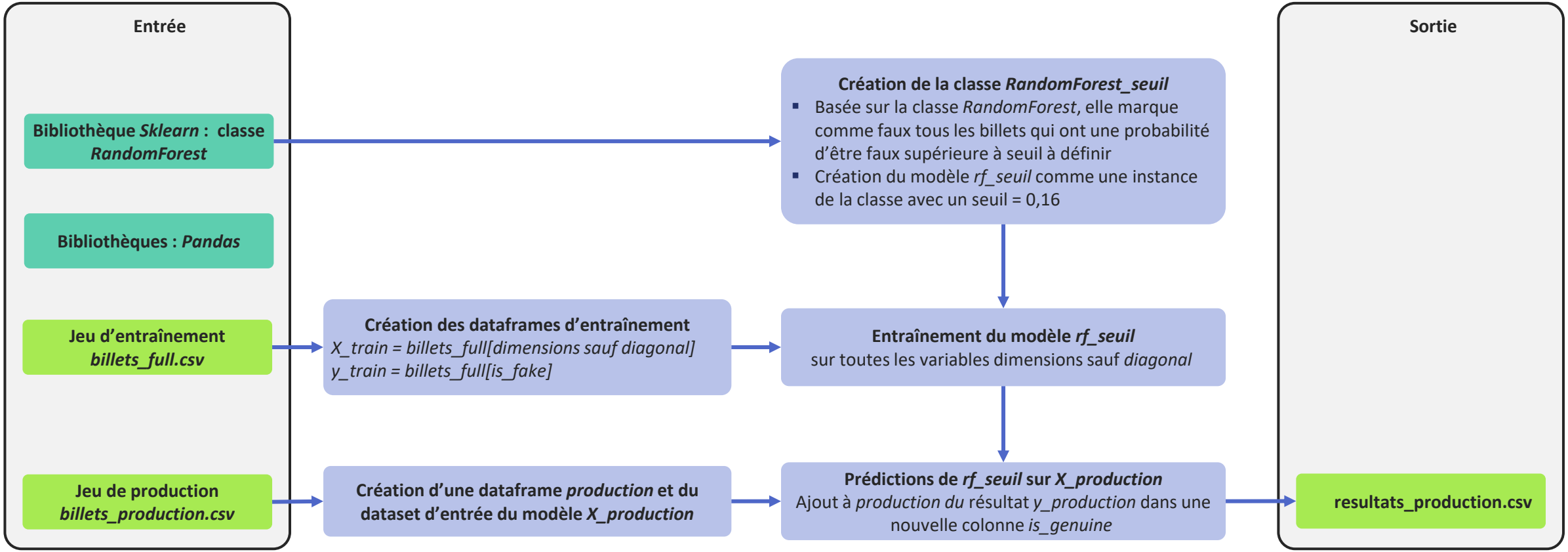
La durée reste très faible sur de grosses quantités de billets : on privilégiera donc un algorithme ayant de bonnes performances de détection à un algorithme rapide.

Performances de l'algorithme retenu : Forêt aléatoire avec seuil 0,16

- Rappel : 99,6 %
- Précision : 93,5 %
- Exactitude : 97,9 %
- Durée d'exécution sur 1 million de billets : 3,8 secondes

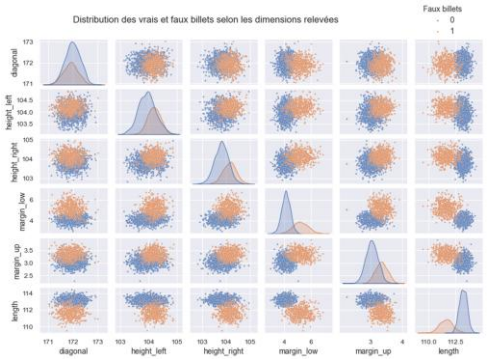
3. Un algorithme de détection des faux-billets performant et modulable

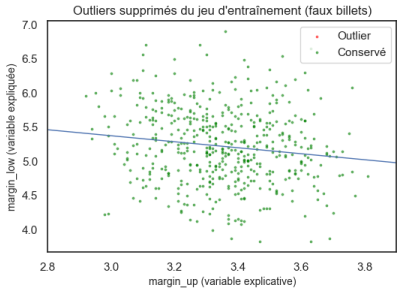
Script de production : Notebook production.ipynb



Conclusion

- 6 variables de dimensions relevées sur 1500 billets
- 1 variable *is_genuine* transformée en *is_fake*
- Peu de corrélation observée entre les variables de dimension mais des répartitions de dimensions très différentes entre vrais et faux billets
- 37 valeurs manquantes de *margin_low* manquantes à déterminer par régression linéaire





- Le modèle de régression linéaire, basé sur la très légère corrélation entre la marge inférieure et la marge supérieure des billets, donne de très mauvais résultats
- Le modèle est légèrement meilleur sur les faux billets que sur les vrais billets

- Les meilleures performances de l'algorithme de détection sont obtenues avec la régression logistique et la forêt aléatoire
- La durée d'exécution est faible même sur de gros jeux de données (quelques secondes à une minute)
- On retient un modèle de forêt aléatoire un seuil de classification des faux billets abaissé à 16% :
  - Rappel de 99,6 %
  - Précision de 94 %
  - Classification d'un million de billets en 4 secondes