OpenClassrooms - Data Analyst

2021-2022

ONCFM

PROJET10: DÉTECTEZ DES FAUX BILLETS AVEC PYTHON

Xiuting LIANG 09.2022

Le contexte du projet de data analyse

ONCFM

Contexte

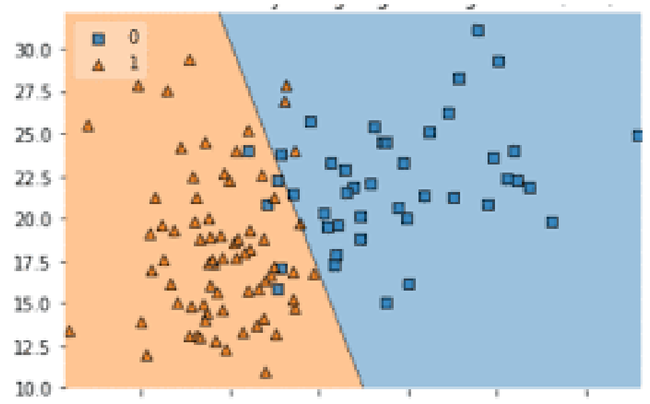
L'Organisation nationale de lutte contre le faux-monnayage, ou ONCFM, est une organisation publique ayant pour objectif de mettre en place des méthodes d'identification des contrefaçons des billets en euros.

Misson

Notre mission est construire un algorithme qui, à partir des caractéristiques géométriques d'un billet, serait capable de définir si ce dernier est un vrai ou un faux billet.

Stratégie d'analyse

Nous ultilisons les 5 informations géométriques sur un billet: length, height_left, height_right, margin_up, margin_low.
Un algorithme de la régression logistique classique est ultilisé pour identifier les billets vrais et faux avec présentations des résultats et des probabilités.



Dans Cette Présentation

Traiter les valeurs manquantes avec la régression linéaire

- Normalisation
- Comprendre margin_low
- Modèle de LinearRegression

Explorer des données

- Pairplot
- Distribution et normalité
- Carte thermique de corrélation triangulaire
- ACP

Régression logistique

- Créer modèles
- Les dataset pour training et testing
- Métriques d'évaluation

KMeans

- La méthode Elbow
- Silhouette analyse
- Créer models
- Métriques d'évaluation

Modèle final et l'application

- Exporter le modèle avec Joblib
- application
- application web

Traiter les valeurs manquantes avec la régression linéaire

01 **Normalisation**

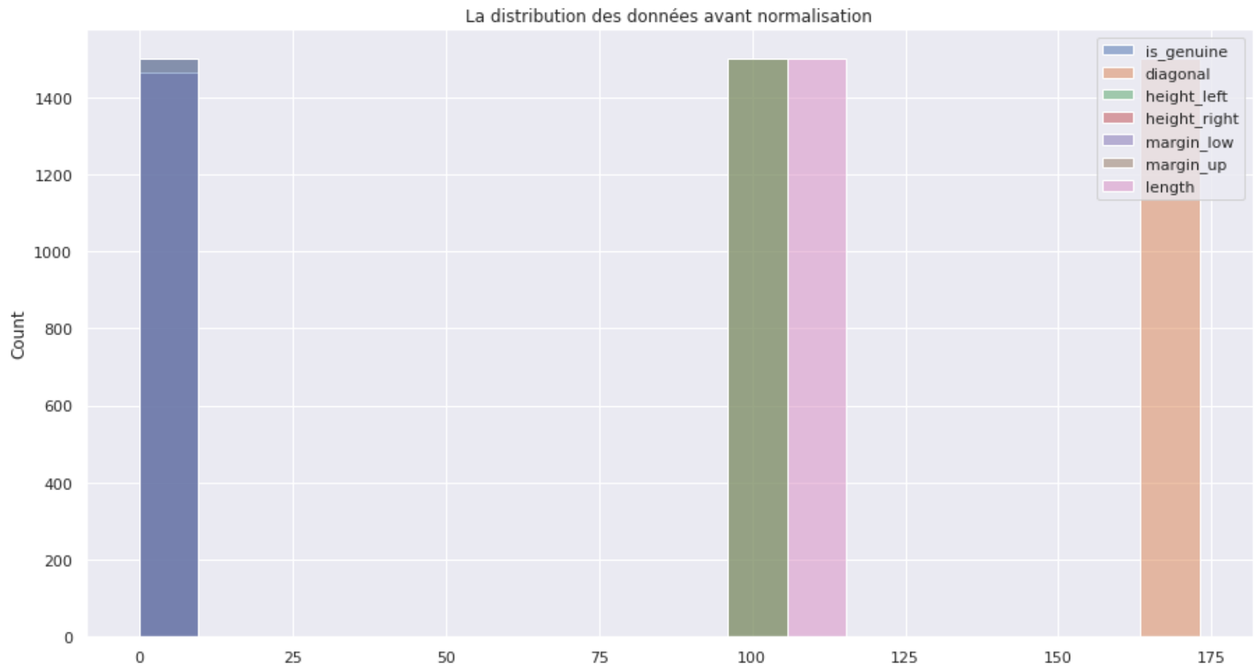
02 **Comprendre margin_low**

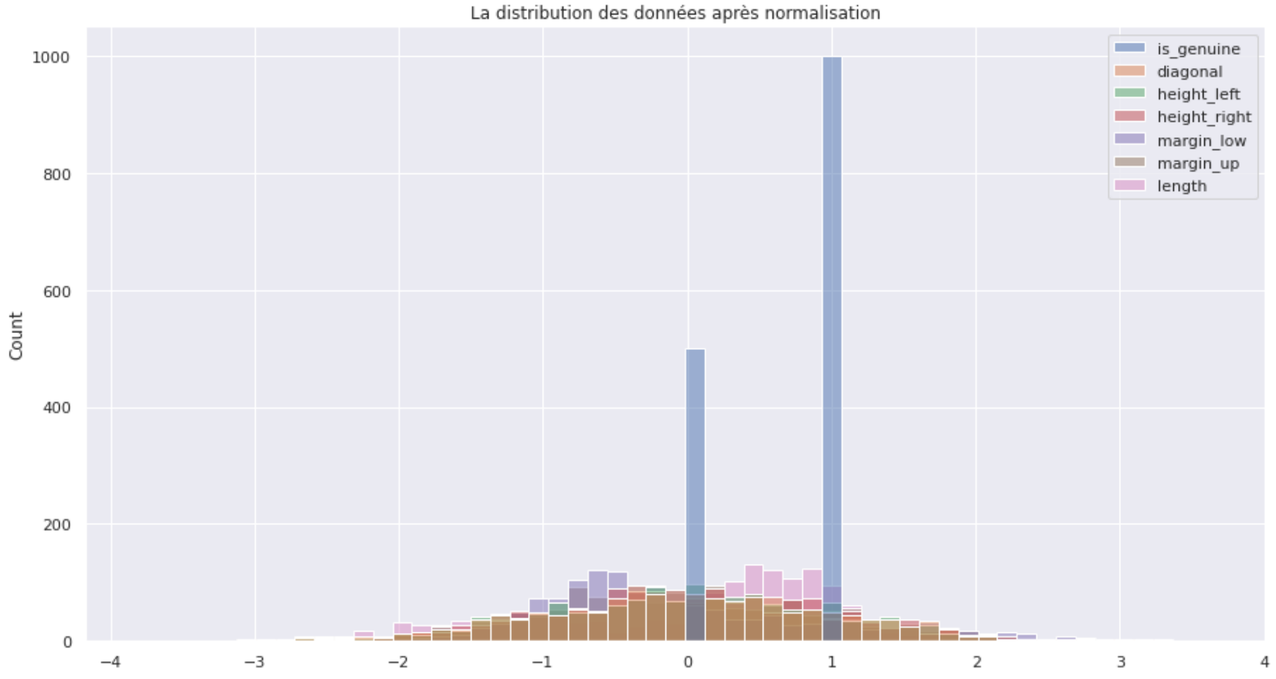
- La distribution normale
- Les corrélations
- Relation linéaire avec pairplot

03 **Régression Lineaire**

- R2
- La significativité
- La colinéarité
- L’homoscédasticité
- La normalité

Normalisation



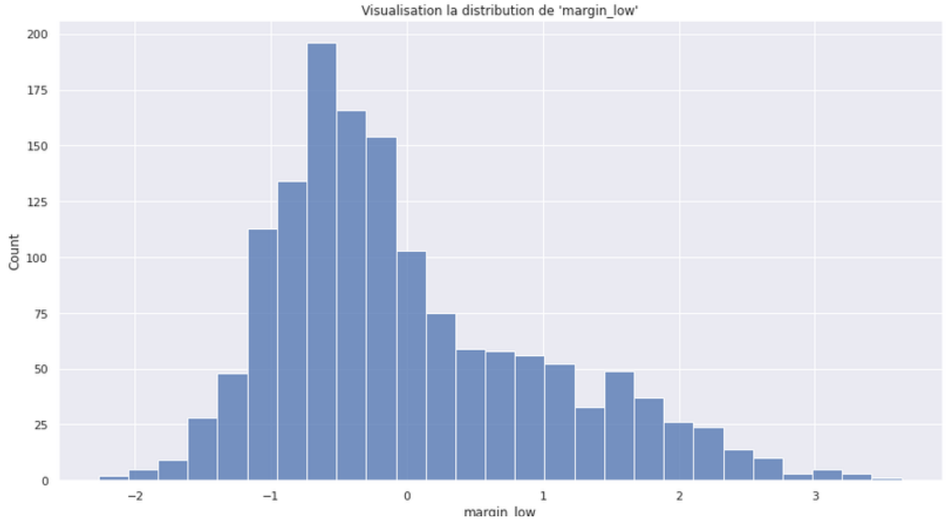


Avant normalisation

Après normalization

Comprendre margin_low 1

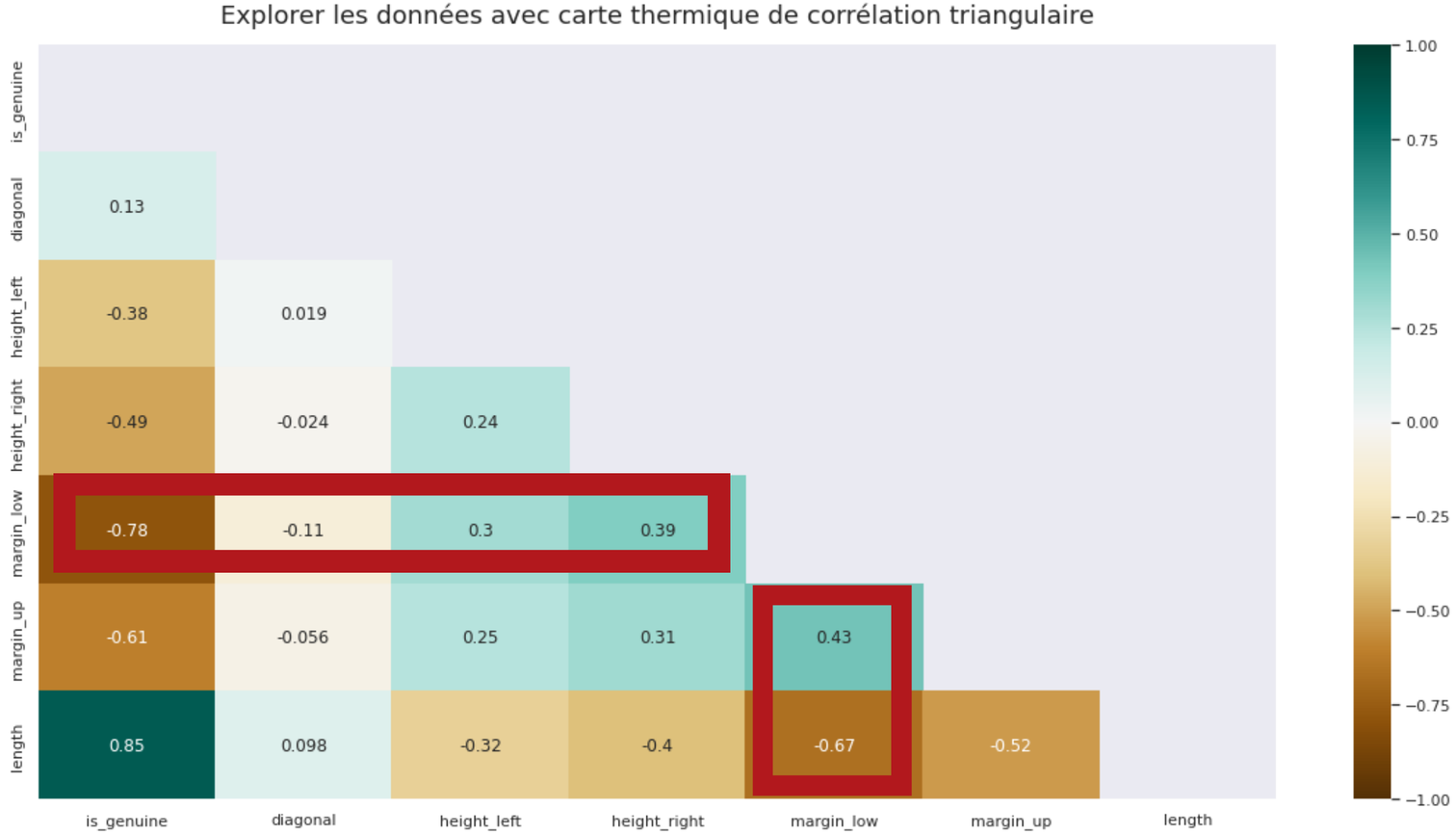
La distribution normale



Verifier la normalité avec Teste Shapiro-Wilk

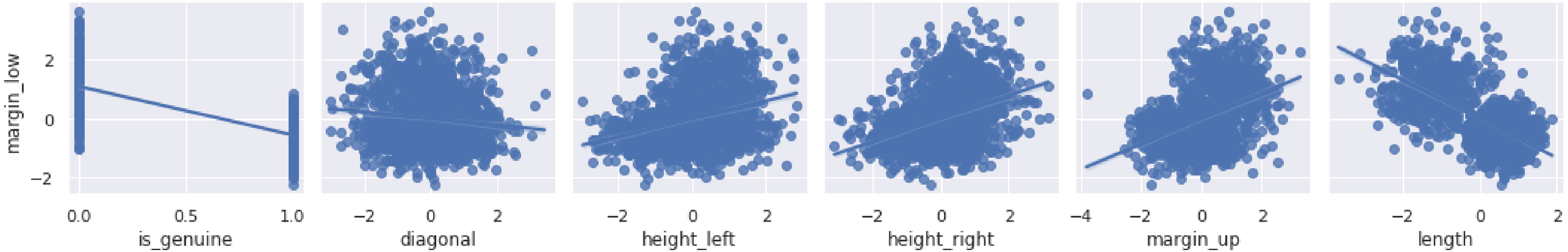
```
Statistics=nan, p=1.000
L'hypothèse de normalité est donc tolérée (accepte H0)
```

Carte thermique de corrélation triangulaire



Comprendre margin_low 2

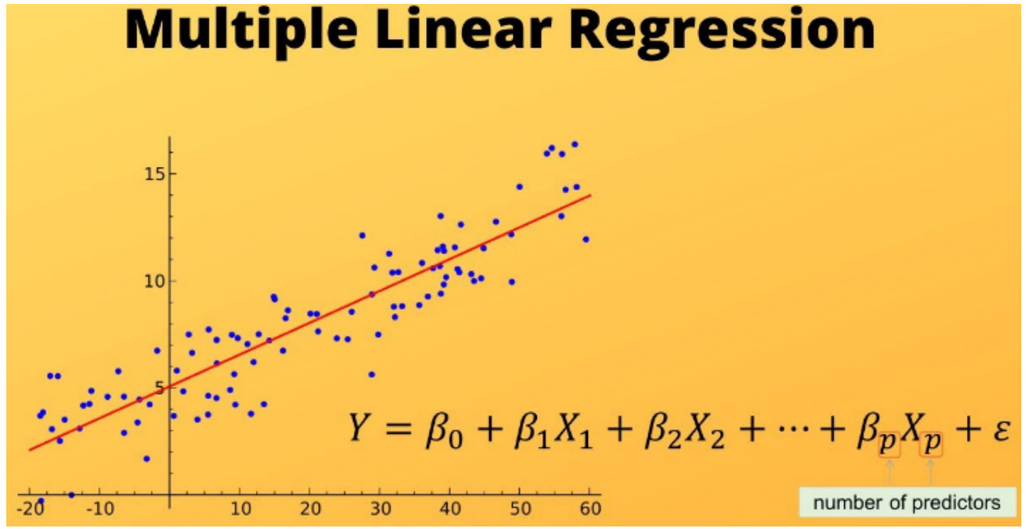
Relation linéaire avec pairplot



Comprendre 'margin_low', qu'il y a des valeurs manquantes:

- Il est corrélé avec 'is_genuine', 'margin_low' et 'length'
- 'Margin_low' est une variable quantitative, qui répère la loi de normalité
- Il y a de la relation linéaire avec 'is_genuine', 'margin_up' et 'length'. On peut utiliser une régression linéaire pour la prévision.

Régression linéaire multiple



Le modèle de régression linéaire multiple considère qu'il existe une relation linéaire entre Y et nos p variables explicatives.

- Y est une v.a.r, quantivative observable ;
- Expliquée, modélisée par plusieurs variables quantivatives (X1-Xp);
- (β1,...,βp) sont des paramètres inconnus (non observables) ;
- ε , l'erreur du modèle, est une v.a.r centrée de variance σ2 inconnue (c'est également un paramètre du modèle).

Explanation des coefficients

Le coefficient de détermination R2

$$\mathrm{R}^2 = \frac{\mathrm{SCE}}{\mathrm{SCT}}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Test pour la significativité

Une statistique de Fisher

Un test global pour la significativité du modèle global. en pratique, l'hypothèse Ho de ce test est souvent rejetée, le modèle est donc souvent significatif globalement.

P valeur pour le T test

Un test de significativité sur chacune des variables explicatives prises une à une par le test de Student. Ici, tester l'un des paramètres a un réel sens : si une variable n'est pas significative, il faut la retirer du modèle. Si l'on ne la retire pas, il est possible que l'erreur de prévision du modèle soit plus élevée.

Régression Linéaire 1

Retirez les variables non significatives (statemodels)

Tous les variables

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.606
Model:	OLS	Adj. R-squared:	0.604
Method:	Least Squares	F-statistic:	261.2
Date:	Thu, 15 Sep 2022	Prob (F-statistic):	5.74e-202
Time:	19:13:05		
No. Observations:	1024	AIC:	1945.
Df Residuals:	1017	BIC:	1979.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.1604	0.063	18.489	0.000	1.037	1.284
is_genuine	-1.7385	0.089	-19.432	0.000	-1.914	-1.563
diagonal	-0.0052	0.020	-0.263	0.792	-0.044	0.033
height_left	0.0202	0.021	0.968	0.333	-0.021	0.061
height_right	0.0190	0.023	0.834	0.404	-0.026	0.064
margin_up	-0.0770	0.025	-3.122	0.002	-0.125	-0.029
length	0.0290	0.037	0.790	0.430	-0.043	0.101

Omnibus:	21.991	Durbin-Watson:	1.937
Prob(Omnibus):	0.000	Jarque-Bera (JB):	39.991
Skew:	0.114	Prob(JB):	2.07e-09
Kurtosis:	3.941	Cond. No.	8.75

4 Variables

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.597
Model:	OLS	Adj. R-squared:	0.596
Method:	Least Squares	F-statistic:	378.1
Date:	Thu, 15 Sep 2022	Prob (F-statistic):	1.40e-199
Time:	19:13:05		
No. Observations:	1024	AIC:	2000.
Df Residuals:	1019	BIC:	2025.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.1382	0.044	25.645	0.000	1.051	1.225
is_genuine	-1.7114	0.060	-28.579	0.000	-1.829	-1.594
height_left	0.0230	0.022	1.055	0.291	-0.020	0.066
height_right	-0.0002	0.023	-0.007	0.995	-0.046	0.045
margin_up	-0.0769	0.026	-3.003	0.003	-0.127	-0.027

Omnibus:	13.258	Durbin-Watson:	1.971
Prob(Omnibus):	0.001	Jarque-Bera (JB):	20.938
Skew:	0.058	Prob(JB):	2.84e-05
Kurtosis:	3.691	Cond. No.	4.94

***Procédure descendante ou backward**

Régression Linéaire 2

Retirez les variables non significatives

3 Variables

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.617
Model:	OLS	Adj. R-squared:	0.616
Method:	Least Squares	F-statistic:	547.3
Date:	Thu, 15 Sep 2022	Prob (F-statistic):	6.98e-212
Time:	19:13:05	[illegible]	[illegible]
No. Observations:	1024	AIC:	1927.
Df Residuals:	1020	BIC:	1947.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.1950	0.041	29.035	0.000	1.114	1.276
is_genuine	-1.7595	0.054	-32.535	0.000	-1.866	-1.653
height_left	0.0014	0.021	0.066	0.947	-0.040	0.043
margin_up	-0.0831	0.024	-3.419	0.001	-0.131	-0.035

Omnibus:	20.485	Durbin-Watson:	2.149
Prob(Omnibus):	0.000	Jarque-Bera (JB):	39.731
Skew:	0.016	Prob(JB):	2.36e-09
Kurtosis:	3.964	Cond. No.	4.45

2 Variables

OLS Regression Results

Dep. Variable:	margin_low	-squared:	0.620
Model:	OLS	dj. R-squared:	0.619
Method:	Least Squares	-statistic:	832.3
Date:	Thu, 15 Sep 2022	rob (F-statistic):	3.88e-215
Time:	19:13:05	[illegible]	[illegible]
No. Observations:	1024	AIC:	1926.
Df Residuals:	1021	BIC:	1941.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.1465	0.039	29.207	0.000	1.070	1.224
is_genuine	-1.7543	0.052	-33.863	0.000	-1.856	-1.653
margin_up	-0.0788	0.025	-3.196	0.001	-0.127	-0.030

Omnibus:	12.081	Durbin-Watson:	1.985
Prob(Omnibus):	0.002	Jarque-Bera (JB):	18.984
Skew:	0.024	Prob(JB):	7.55e-05
Kurtosis:	3.665	Cond. No.	4.18

- **Prob (F-statistic) est inférieur à 0.05, le modèle globale est corrélé avec 'margin_low'.**
- **Les variables 'is_genuine' et 'margin_up' sont significativement corrélés avec 'margin_low'.**

Régression Linéaire 3
Validation du modèle

Vérifier la colinéarité des variables

```
variables = result2.model.exog
[variance_inflation_factor(variables, i) for i in np.arange(1,variables.shape[1])]
```

```
[1.620905480797004, 1.620905480797004]
```

Tous les coefficients sont inférieurs à 5, il n'y a donc pas de problème de colinéarité.

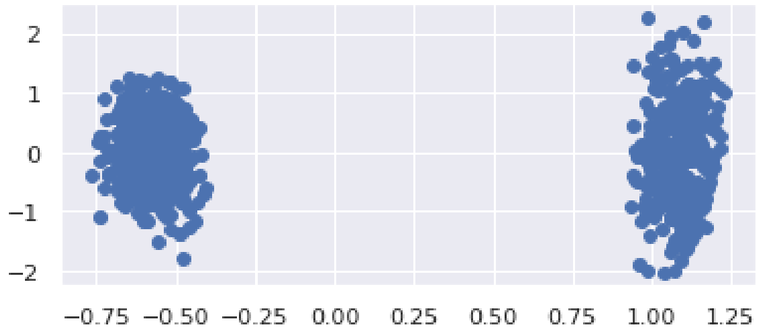
Tester l'homoscédasticité (Constant Error Variance)

Tester l'homoscédasticité (Constant Error Variance)

```
[531] _, pval, __, f_pval = statsmodels.stats.diagnostic.het_breuschpaga
      print('p value test Breusch Pagan:', pval)
```

```
p value test Breusch Pagan: 3.9267726022230334e-23
```

```
[532] y_pred = result2.predict(X_train_sm2)
      fig, ax = plt.subplots(figsize=(6,2.5))
      _ = ax.scatter(y_pred, result2.resid)
```



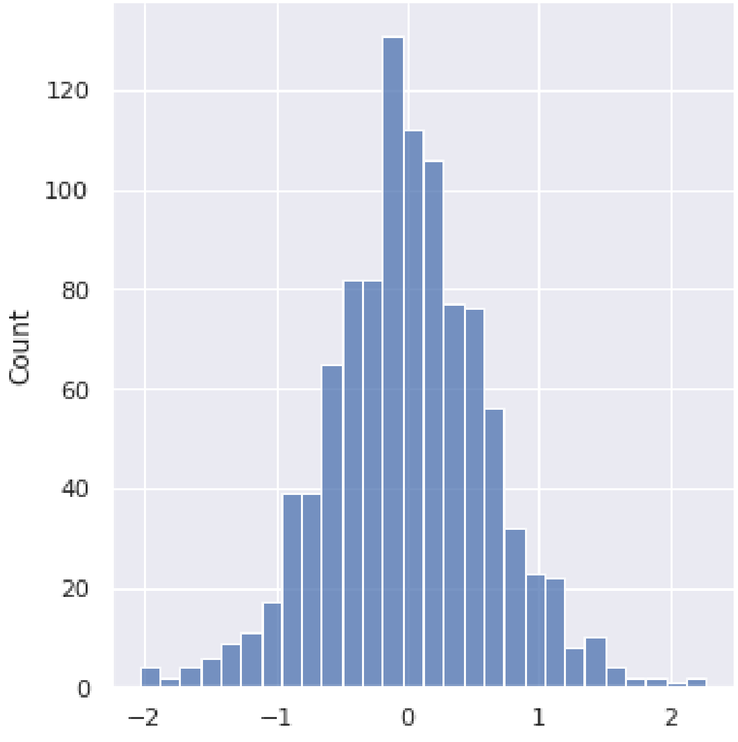
- H0 : homoscédasticité • H1 : hétéroscédasticité

Si la probabilité associée au test est inférieure à α, on rejette l'hypothèse d'homoscédasticité (H0). En revanche, si la probabilité est supérieure à α, l'hypothèse nulle est vérifiée et nous pouvons supposer l'homoscédasticité des résidus. Avec α = 5% = seuil de significativité.

La p-valeur ici est inférieure à 5 %, on rejette l'hypothèse H0 selon laquelle les variances ne sont pas constantes (hétéroscédasticité).

Régression Linéaire 4

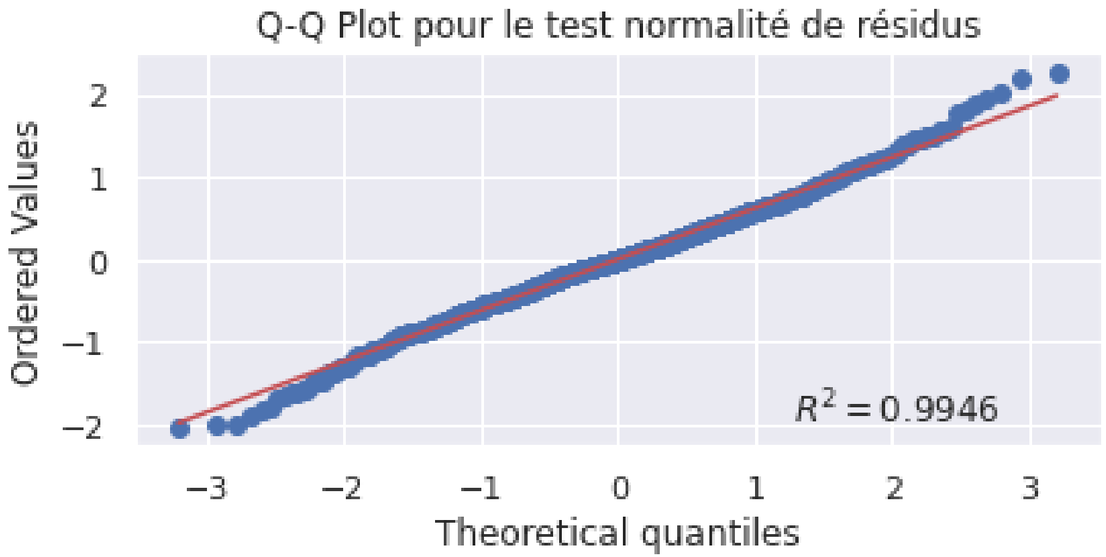
Validation du modèle



Tester la normalité des résidus

```
[534] shapiro(result2.resid)
```

```
ShapiroResult(statistic=0.9948773980140686, pvalue=0.0015356993535533547)
```



Moyenne

```
[535] np.mean(result2.resid)
```

```
-5.478256737134757e-15
```

Ici, l'hypothèse de normalité est remise en cause.

Par contre, on a vérifié la distribution de résidus par l'histrogramme, le QQ plot, et la calculation de moyenne de résidus. Les résultats nous indiquent qu'ils ne soient pas très différents d'une distribution symétrique, et le fait que l'échantillon soit de taille suffisante (supérieure à 1000) permettent de dire que les résultats obtenus par le modèle linéaire gaussien ne sont pas absurdes, même si le test la normalité résidu par le test Shapiro n'est pas considéré comme étant gaussien.

Régression Linéaire 5

Créer le modèle avec Sklearn avec les 2 variables

Créer le modèle avec Sklearn avec les 2 variables

```
[537] # Training l'algorithme
      lr_2 = LinearRegression()
      lr_2.fit(X_train_2, y_train_2)
```

```
      LinearRegression()
```

```
      # Les scores de training et testing sets
      print("Training set score avec 2 variables: {:.2f}".format(lr_2.score(X_train_2, y_train_2)))
      print("Test set score avec 2 variables: {:.2f}".format(lr_2.score(X_test_2, y_test_2)))
```

```
      Training set score avec 2 variables: 0.62
      Test set score avec 2 variables: 0.61
```

```
[539] # Les scores et les coefficients
      y_t_predit2 = lr_2.predict(X_test_2)
      r2_2 = r2_score(y_test_2, y_t_predit2).round(2)

      print('Intercept: {:.2f}'.format(lr_2.intercept_))
      print('Coefficients: ' + str(lr_2.coef_))
      print('R2 score: ' + str(r2_2))
      print('Mean squared error: {:.2f}'.format(mean_squared_error(y_test_2, y_t_predit2).round(2)))
      print('Mean absolute error MAE: {:.2f}'.format(mean_absolute_error(y_test_2, y_t_predit2).round(2)))
      )
```

```
Intercept: 1.15
Coefficients: [-1.75434075 -0.07875528]
R2 score: 0.61
Mean squared error: 0.39
Mean absolute error MAE: 0.48
Median absolute error: 0.39
```

Explanation des formules:

Explanation des formules:

R2

On appelle coefficient de détermination, noté R2 , le réel dans [0,1] défini par :

$$\mathrm{R}^2 = \frac{\mathrm{SCE}}{\mathrm{SCT}}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Dans le cas de la régression avec constante :

Si R2=1 , on a alors SCE=SCT : toute la variation est expliquée par le modèle.

Si R2=0 , on a alors SCR=SCT : aucune variation n'est expliquée par le modèle.

Mean squared error

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

Analyse des scores:

- Le modèle avec 3 variables corrélés a de meilleures performance que l'autre modèle avec tous les variables concernant R2 score (plus proche à 1) et Mean squared error (le plus petit)

Régression Linéaire 6

Partie supplémentaire - Ridge

Ridge

Ultiliser un autre methode de provision

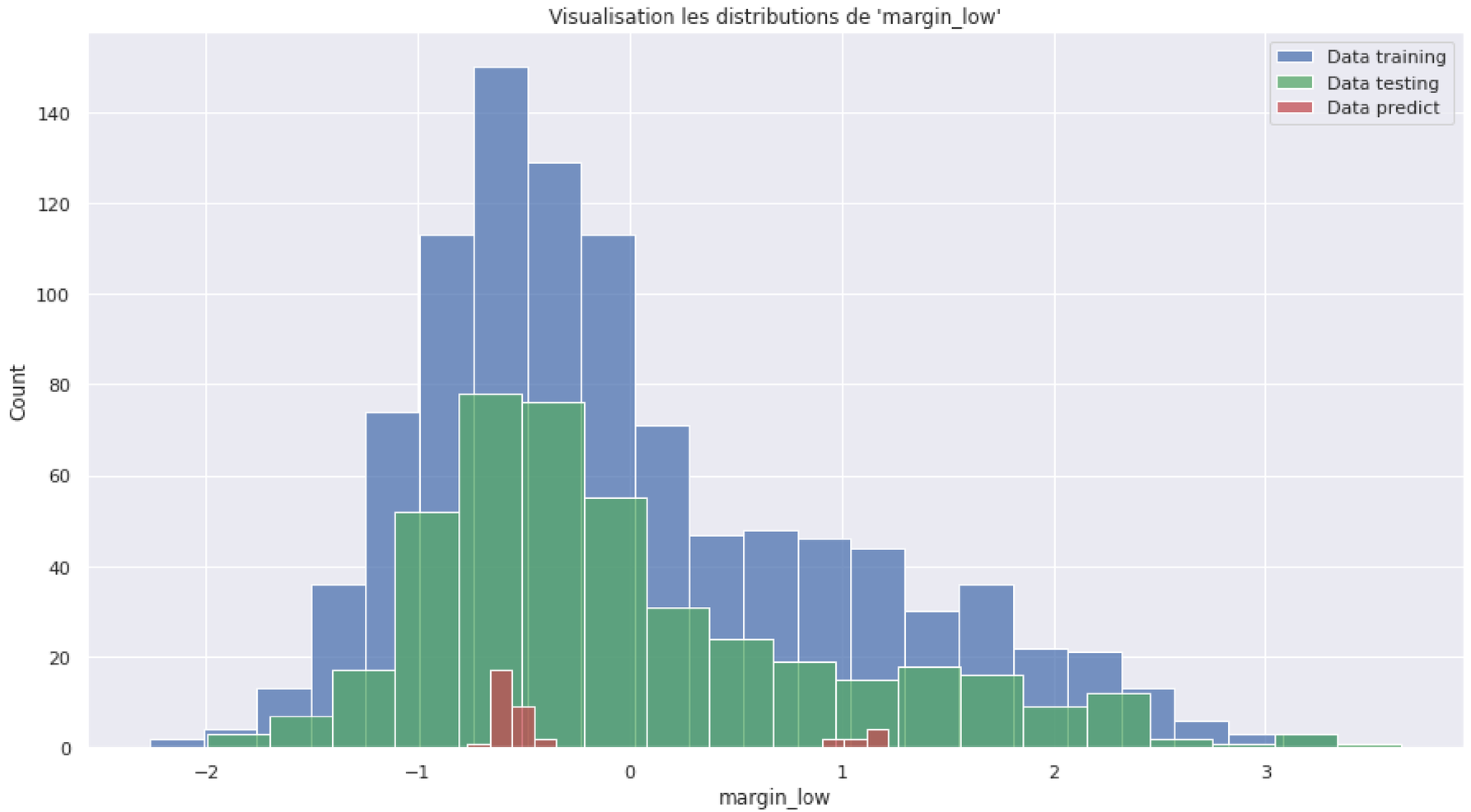
```
[125] ridge = Ridge().fit(X_train_2, y_train_2)
```

```
print("Training set score: {:.2f}".format(ridge.score(X_train_2, y_train_2)))
print("Test set score: {:.2f}".format(ridge.score(X_test_2, y_test_2)))
```

```
Training set score: 0.60
Test set score: 0.64
```

Imputer les valeurs manquants

Les résultats



Explorer des données

01 Pairplot

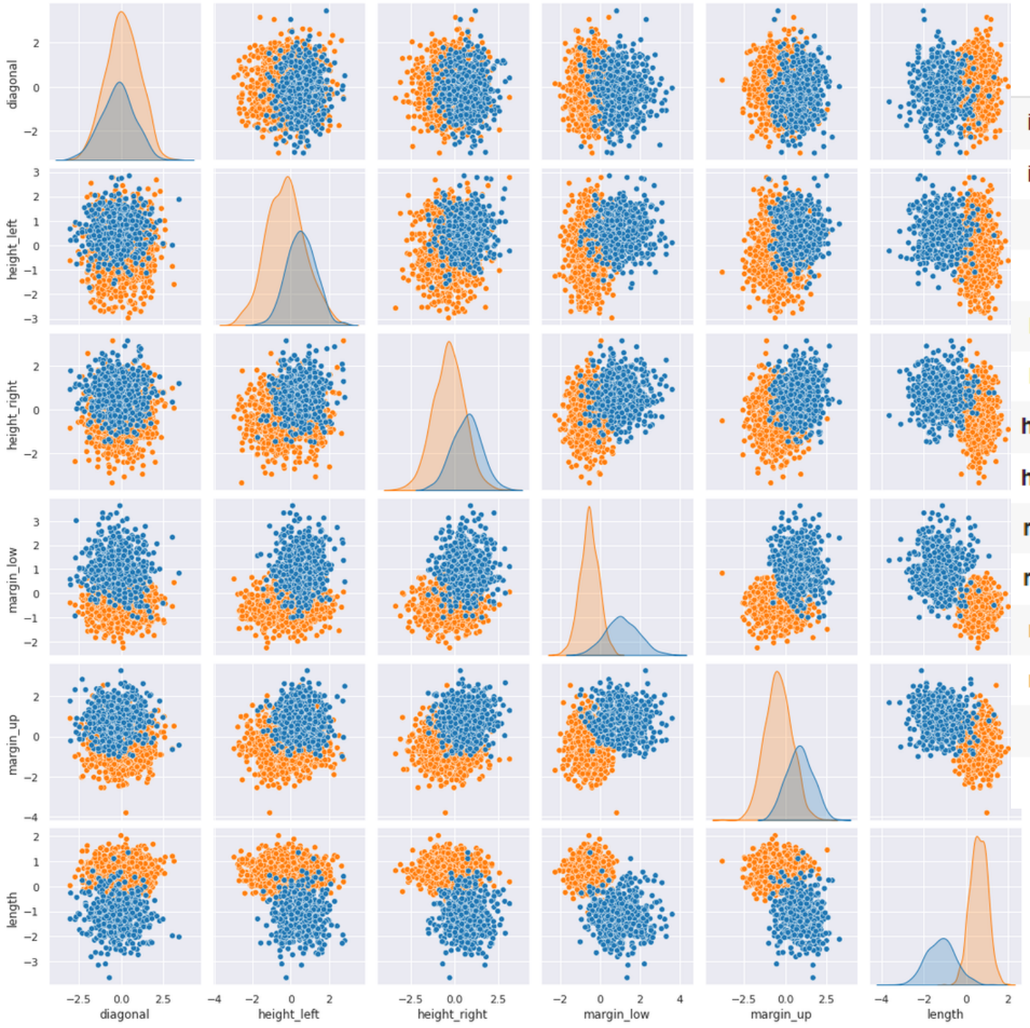
02 Distribution et normalité

03 carte thermique de corrélation triangulaire

04 ACP

Explorer des données

Pairplot



Distribution et normalité

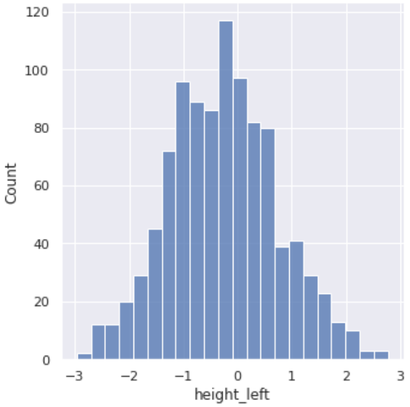
Dimension	Athenticité du billet	Stat	P-valeur	Hypothèse
is_genuine	True	1.000000	1.000000	Hypothèse 0, non rejetée
is_genuine	False	1.000000	1.000000	Hypothèse 0, non rejetée
diagonal	True	0.998064	0.310298	Hypothèse 0, non rejetée
diagonal	False	0.997435	0.638479	Hypothèse 0, non rejetée
height_left	True	0.996582	0.028577	Hypotèse rejetée
height_left	False	0.997877	0.790749	Hypothèse 0, non rejetée
height_right	True	0.998551	0.588416	Hypothèse 0, non rejetée
height_right	False	0.997990	0.826467	Hypothèse 0, non rejetée
margin_low	True	0.997879	0.235744	Hypothèse 0, non rejetée
margin_low	False	0.997278	0.583818	Hypothèse 0, non rejetée
margin_up	True	0.998159	0.355840	Hypothèse 0, non rejetée
margin_up	False	0.995734	0.192617	Hypothèse 0, non rejetée
length	True	0.998048	0.303481	Hypothèse 0, non rejetée
length	False	0.997110	0.526932	Hypothèse 0, non rejetée

Seule la distribution de 'height_left - True' ne suit pas une loi normale.

Revérifier la distribution de 'height_left':

```
[549] sns.displot(df_norm['height_left'][df_norm['is_genuine']==
```

```
<seaborn.axisgrid.FacetGrid at 0x7f265980bed0>
```




```
[550] print("coef d'asymétrie: " + str(round(df_norm['height_left
```

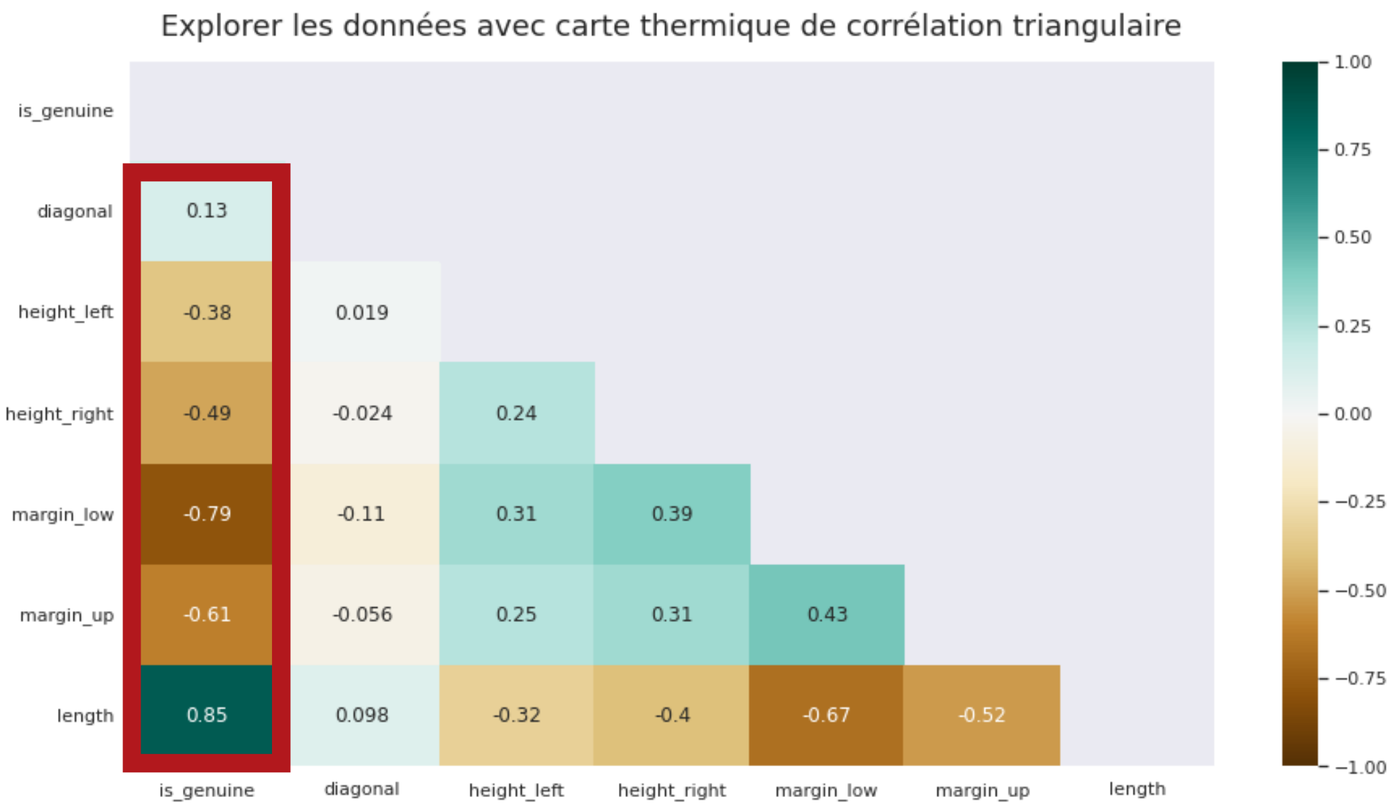
```
coef d'asymétrie: 0.15
```

```
[551] print("coef de kurtosis: " + str(round(df_norm['height_left
```

```
coef de kurtosis: -0.12
```

Explorer des données

Carte thermique de corrélation triangulaire

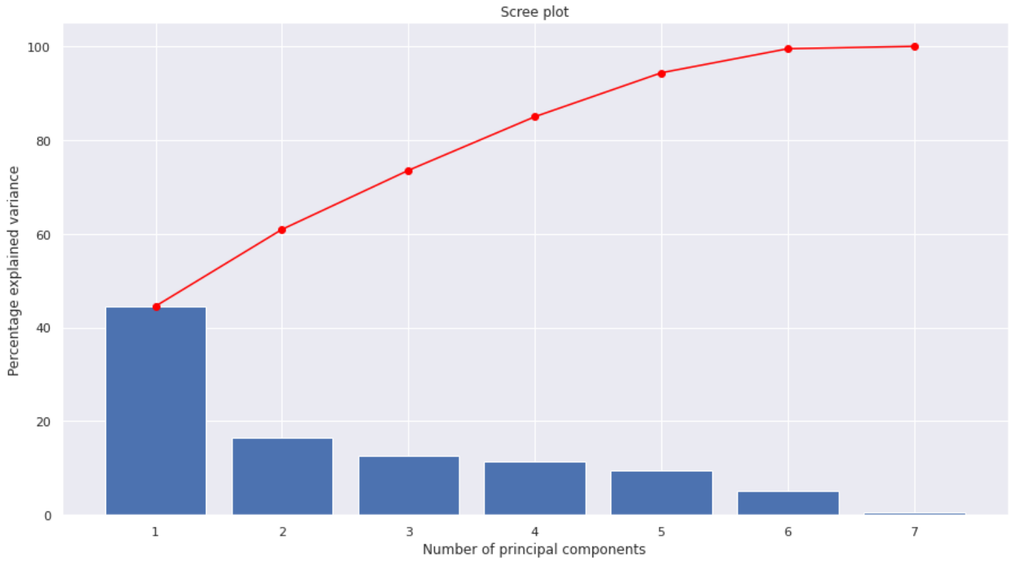


Il y a des corrélations fortes avec les groupes (colonne: 'is_genuine') et les autres variables sauf 'diagonal'.

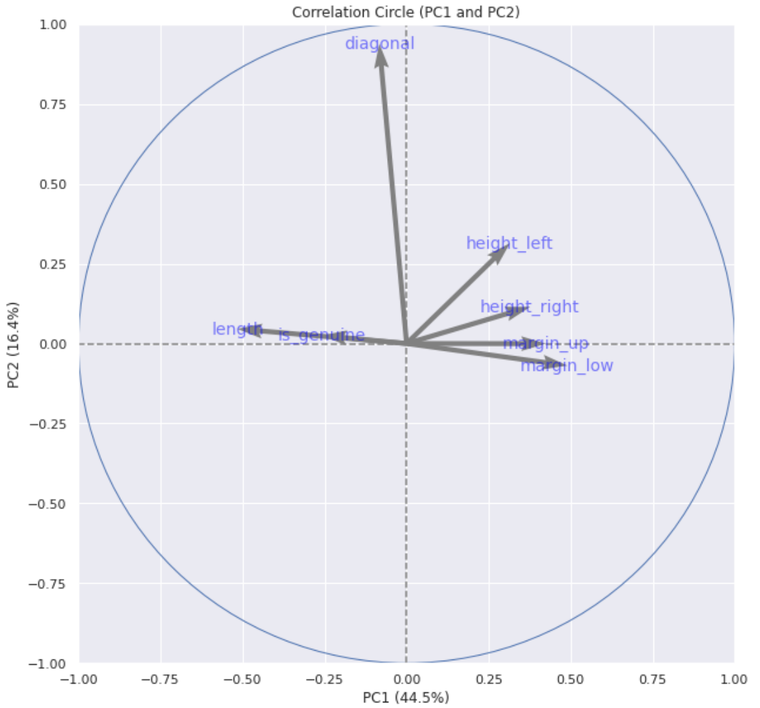
ACP

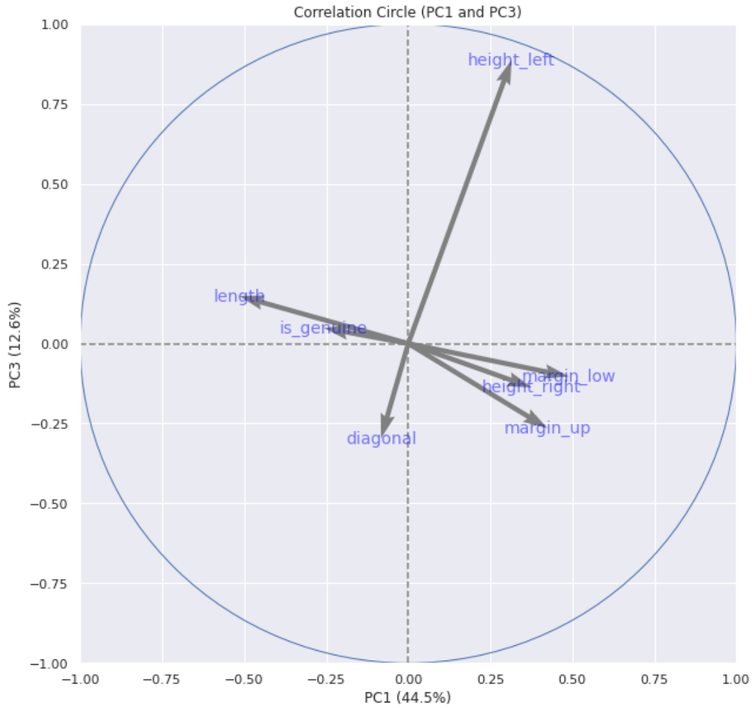
(Elle permet au statisticien de résumer l'information en réduisant le nombre de variables.)

Scree Plot



Cercle de corrélation





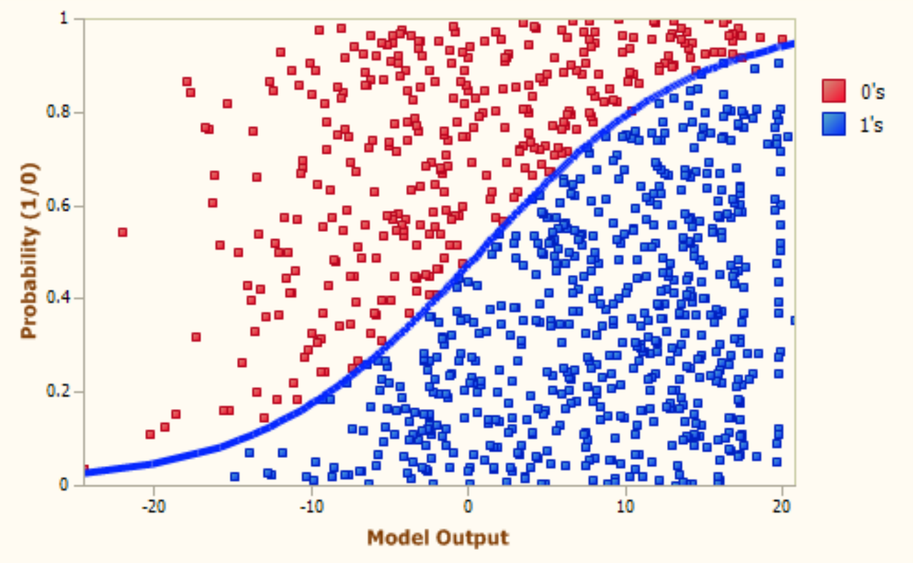
Comprendre les variables

Il y a de corrélation avec 'is_genuine' et 'length'.
Il y a de corrélation négative avec 'is_genuine', 'margin_low' et 'margin_up'.

Régression logistique classique

Explication la méthode:

La régression logistique nous permettra de traiter un cas de classification supervisée, c'est-à-dire un cas où on est en train de modéliser une variable qualitative (dans notre case, une variable binaire comme True et False) par des variables explicatives (les dimensions géométriques d’un billet).



01 **Créer modèles**

02 **Séparer les dataset pour training et testing**

03 **Métriques d'évaluation**

- Matrice de confusion
- Accuracy score
- Rapport de classification
- Courbe ROC

Créer modèles

Retirer les variables non significatives (statemodels)

Tous les variables

Generalized Linear Model Regression Results

Dep. Variable:	['is_genuine[False]', 'is_genuine[True]']	No. Observations:	1500
Model:	GLM	Df Residuals:	1493
Model Family:	Binomial	Df Model:	6
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-39.466
Date:	Thu, 15 Sep 2022	Deviance:	78.932
Time:	19:13:27	Pearson chi2:	2.79e+03
No. Iterations:	10		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-2.1789	0.376	-5.800	0.000	-2.915	-1.443
diagonal	-0.0840	0.347	-0.242	0.809	-0.764	0.596
height_left	0.3789	0.335	1.131	0.258	-0.277	1.035
height_right	0.9535	0.375	2.540	0.011	0.218	1.689
margin_low	3.9926	0.654	6.102	0.000	2.710	5.275
margin_up	2.3868	0.511	4.668	0.000	1.385	3.389
length	-5.2834	0.783	-6.750	0.000	-6.817	-3.749

Retirer 'diagonal'

Generalized Linear Model Regression Results

Dep. Variable:	['is_genuine[False]', 'is_genuine[True]']	No. Observations:	1500
Model:	GLM	Df Residuals:	1494
Model Family:	Binomial	Df Model:	5
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-39.495
Date:	Thu, 15 Sep 2022	Deviance:	78.990
Time:	19:13:28	Pearson chi2:	2.79e+03
No. Iterations:	10		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-2.1809	0.375	-5.810	0.000	-2.917	-1.445
height_right	0.9578	0.374	2.558	0.011	0.224	1.691
height_left	0.3793	0.334	1.135	0.256	-0.276	1.034
margin_low	4.0353	0.635	6.351	0.000	2.790	5.281
margin_up	2.3966	0.511	4.690	0.000	1.395	3.398
length	-5.3053	0.781	-6.796	0.000	-6.835	-3.775

Régression logistique classique

Séparer les dataset pour training et testing

```
[568] # Séparer les dataset pour training et testing
      X_lr = df_norm.drop(columns = ['is_genuine', 'diagonal'])
      y_lr = df_norm['is_genuine']

      X_lr_train, X_lr_test, y_lr_train, y_lr_test = train_test_split(
          X_lr, y_lr, test_size = 0.3)
```

```
[569] # Vérifier les shapes de chaque dataset
      print('X_lr_train shape: ' + str(X_lr_train.shape))
      print('X_lr_test shape: ' + str(X_lr_test.shape))
      print('y_lr_train shape: ' + str(y_lr_train.shape))
      print('y_lr_test: ' + str(y_lr_test.shape))
```

```
X_lr_train shape: (1050, 5)
X_lr_test shape: (450, 5)
y_lr_train shape: (1050,)
y_lr_test: (450,)
```

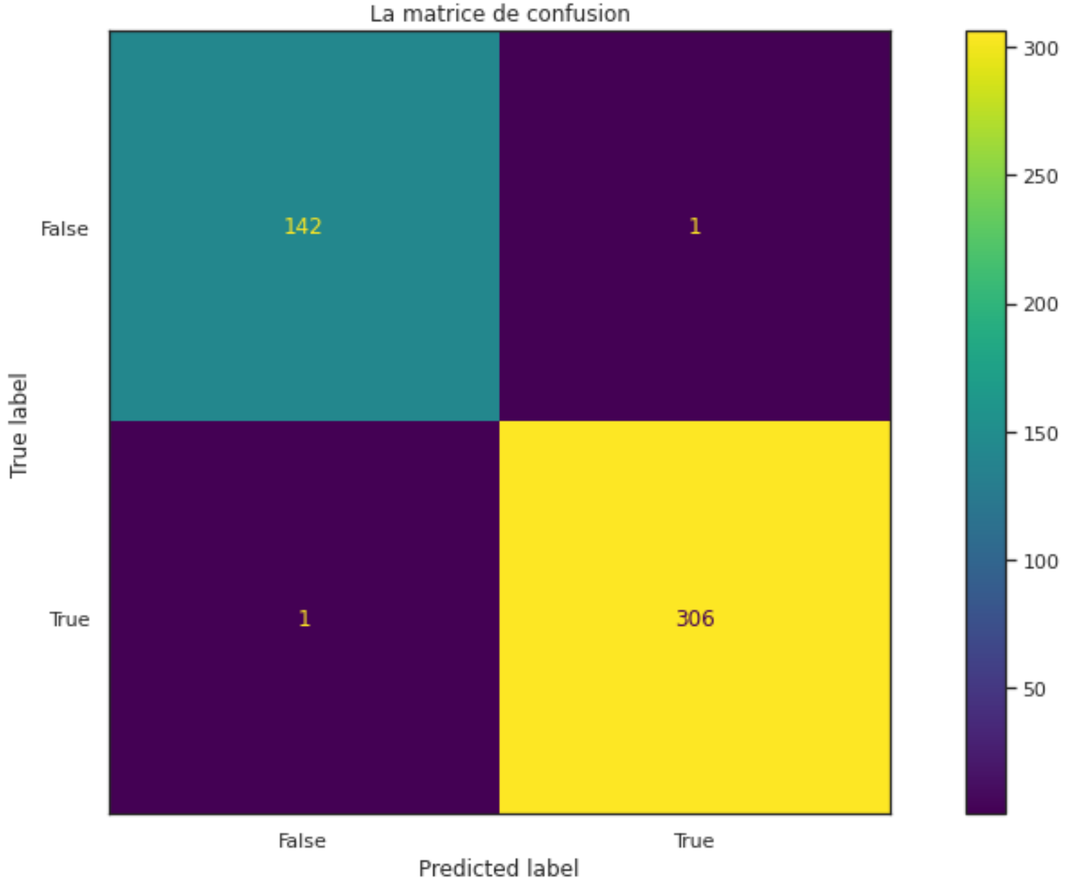
```
# Créer la modèle régression logistique classique
Logreg = LogisticRegression().fit(X_lr_train, y_lr_train)

# Vérifier les scores
print("Training set score: {:.3f}".format(Logreg.score(X_lr_train, y_lr_train)))
print("Test set score: {:.3f}".format(Logreg.score(X_lr_test, y_lr_test)))
```

```
Training set score: 0.990
Test set score: 0.996
```

Métriques d'évaluation

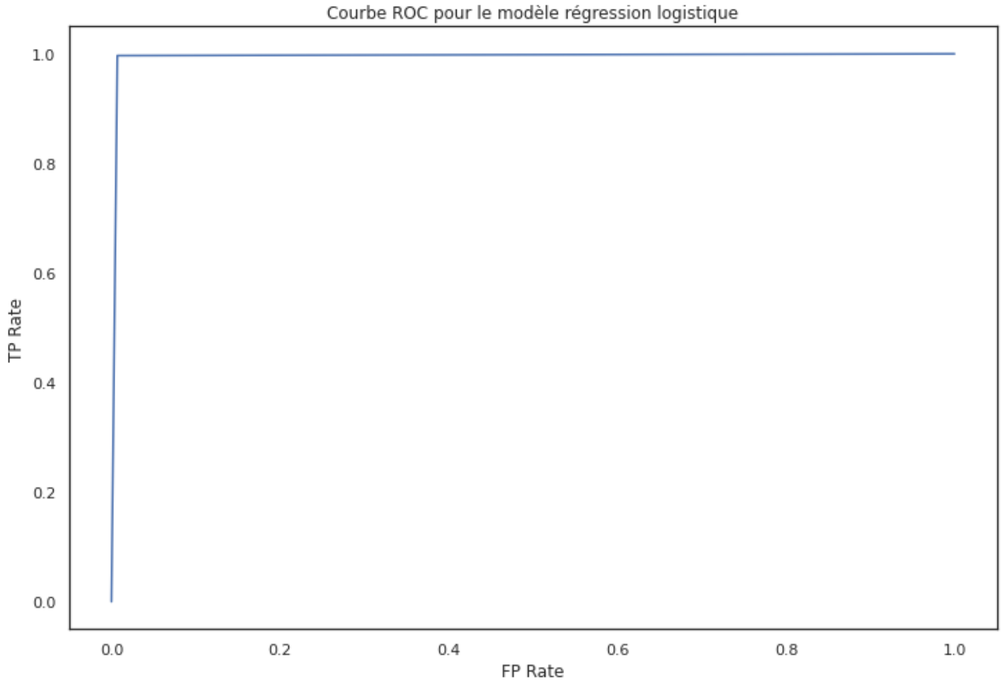
```
La matrice de confusion:
[[142   1]
 [  1 306]]
```

```
...packages/sklearn/utils/deprecation.py:87: Future

Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix`
```



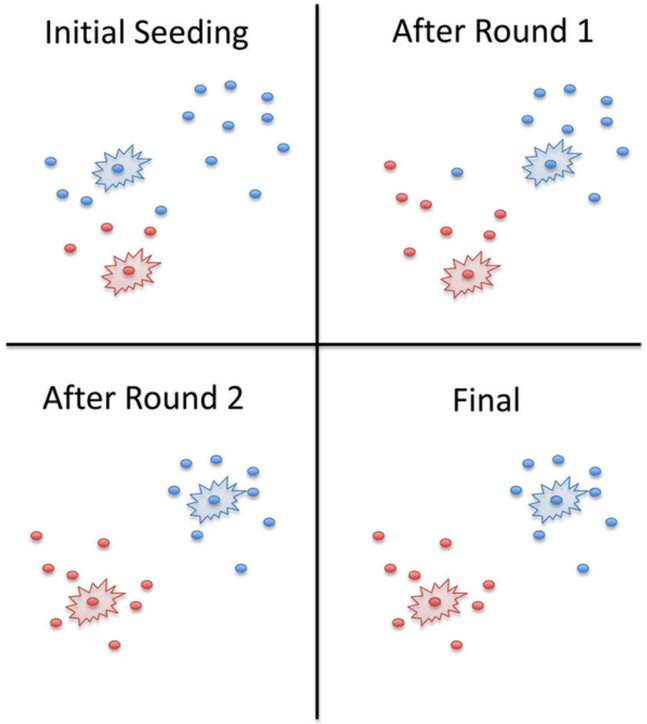
Note de précision: 0.996

	precision	recall	f1-score	support
False	0.99	0.99	0.99	143
True	1.00	1.00	1.00	307
accuracy			1.00	450
macro avg	0.99	0.99	0.99	450
weighted avg	1.00	1.00	1.00	450



Kmeans

- Placer les centroïdes aléatoirement dans l'espace
- Prendre chaque point du nuage et lui associee le cluster du centroïde dont il est le plus proche. On obtient donc K groupes.
- Recalculer les centroïdes de chaque groupe quand les centroïdes bougent jusqu'à ce qu'ils ne bougent pas.



01 **La méthode Elbow**

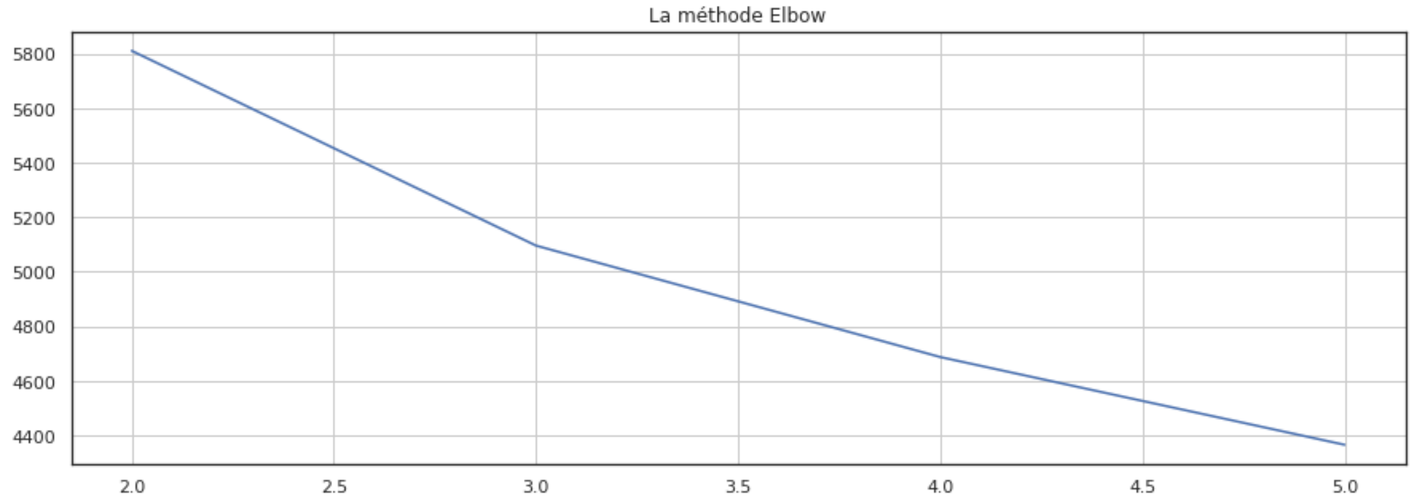
02 **Silhouette analyse**

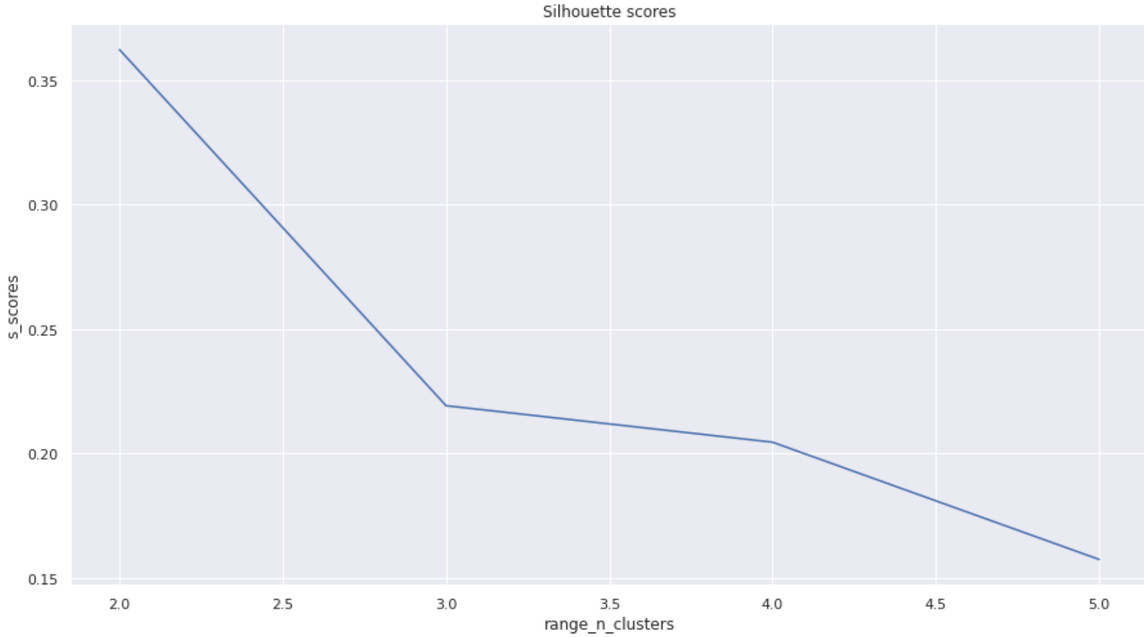
03 **Créer models**

04 **Métriques d'évaluation**

- **Matrice de confusion**
- **Courbe ROC**

Kmeans





On ne peut pas juger combien de groupes qu'on doit choisir par cette méthode.

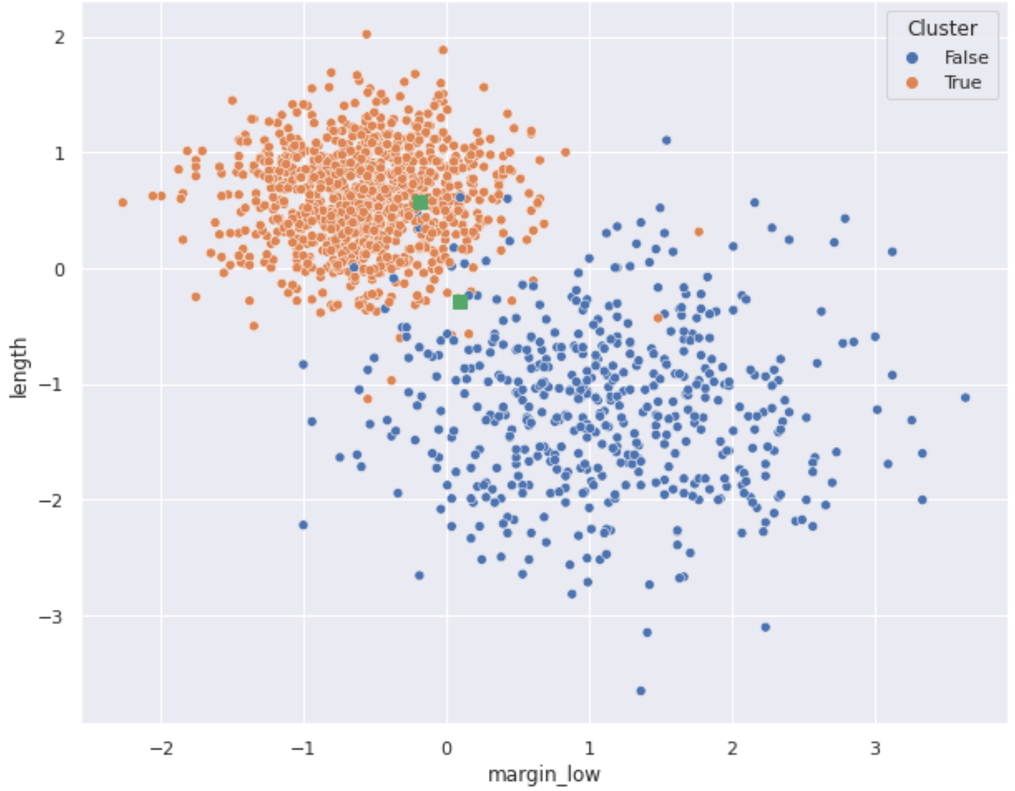
Choisir 2 comme le nombre de clusters

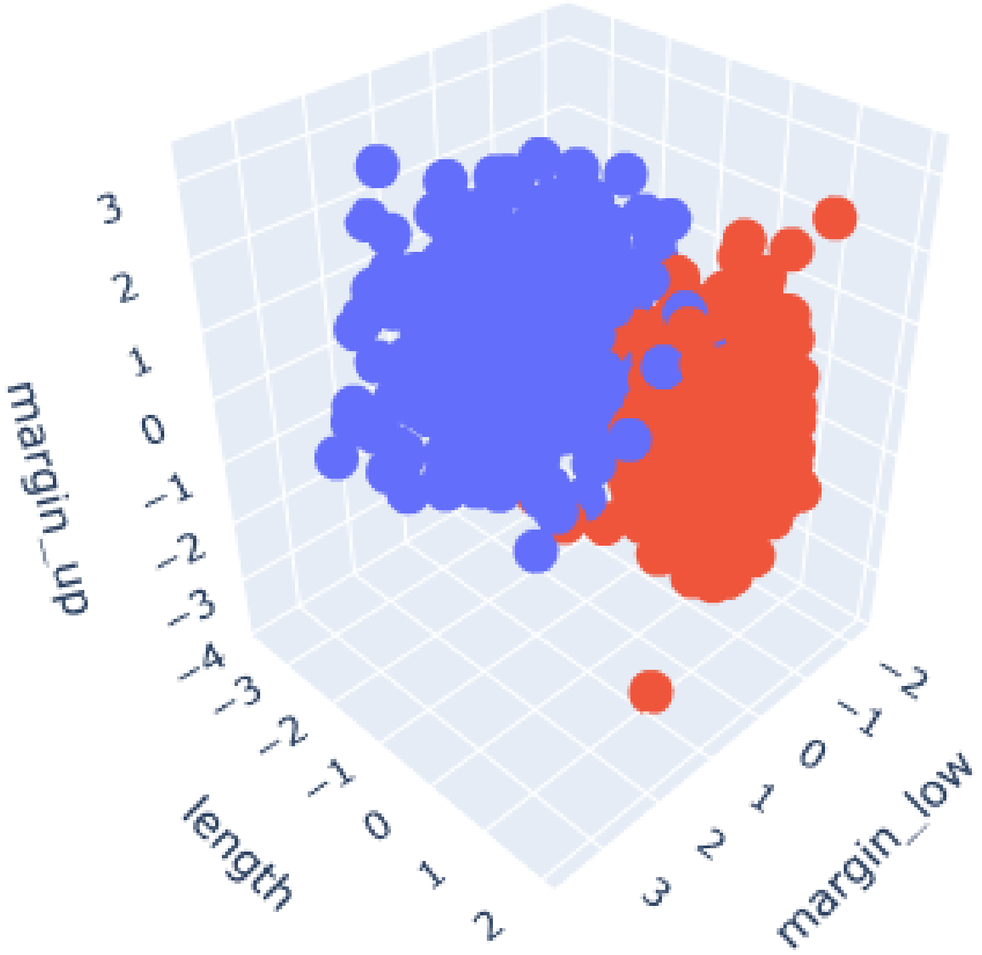
Kmeans

```
Cluster
0    1003
1     497
Name: Cluster, dtype: int64
```

```
[583] a = df_norm[df_norm['Cluster'] == 0]['Cluster'].count()
      b = df_norm[df_norm['Cluster'] == 1]['Cluster'].count()
```

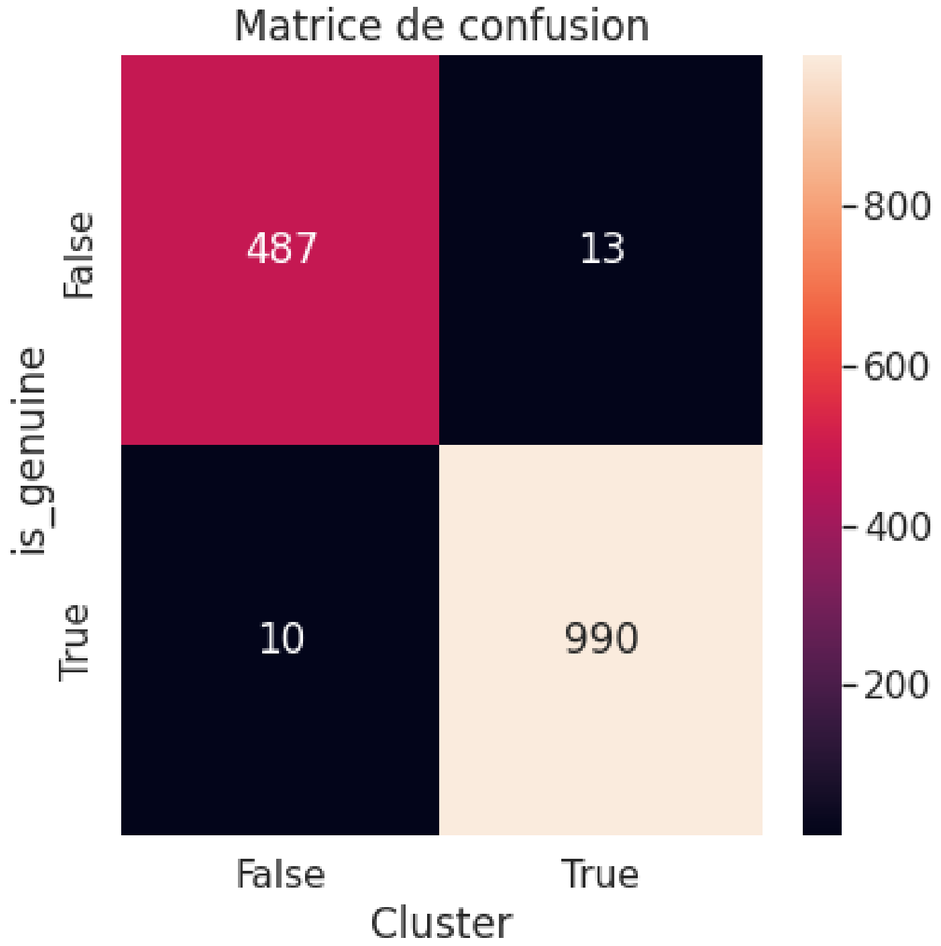
Parce qu'il y a 1000 vrais billets et 500 faux billets, le plus = True, le moindre = False

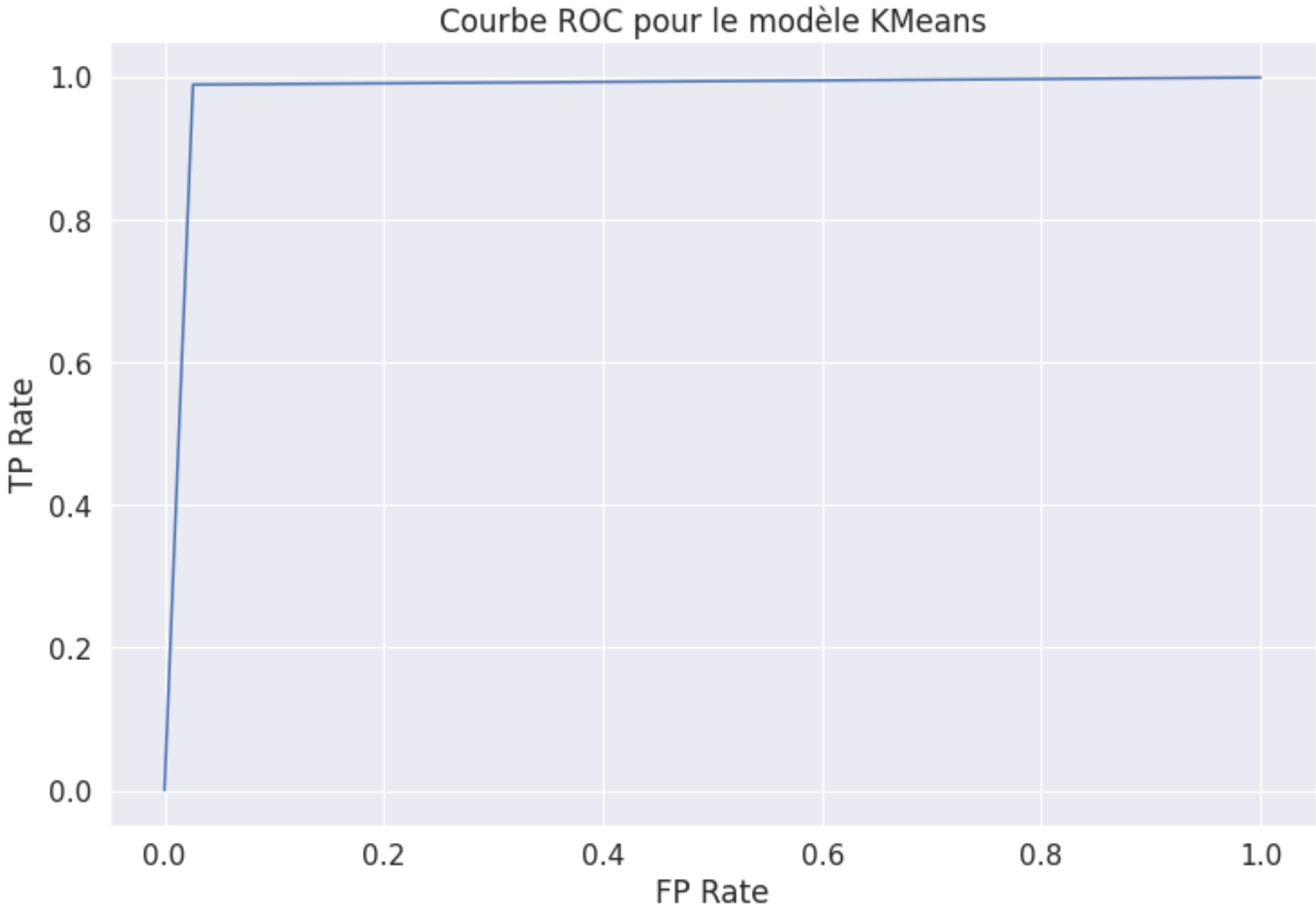




Métriques d'évaluation

```
Matrice de confusion:
[[487  13]
 [ 10 990]]
Text(0.5, 1.0, 'Matrice de confusion')
```





Application

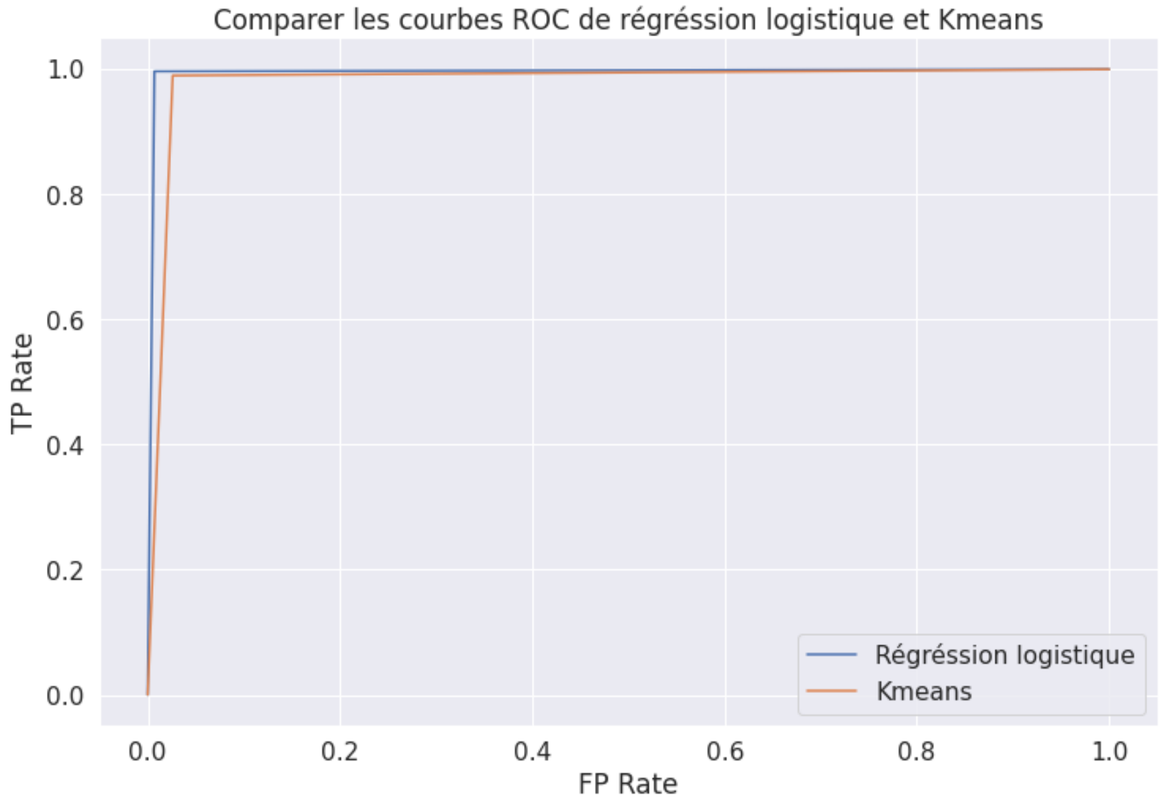
01 Exporter le modèle avec Joblib
- Choisir le modèle
- Tester le modèle avec le fichier de production

02 Silhouette analyse

03 Créer models

04 Métriques d'évaluation
- Matrice de confusion
- Courbe ROC

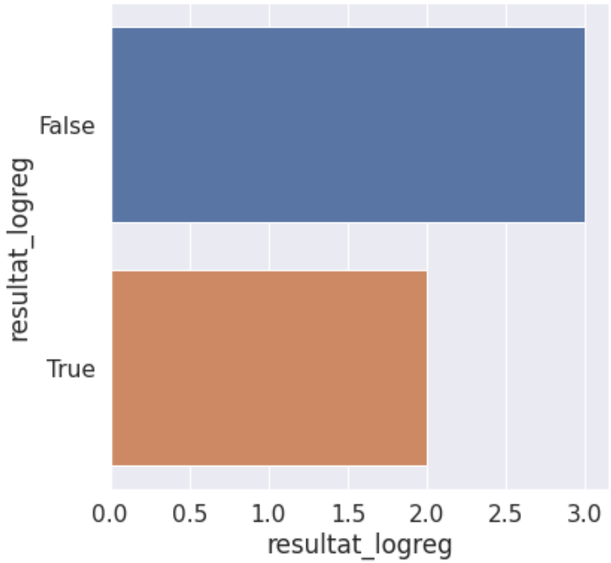
Exporter le modèle avec Joblib




```
[172] jl.dump(Logreg, '/content/drive/MyDrive/Colab Notebooks/P10/LIANG_Xiuting_3_modele_logreg_code_082022.pkl')

['/content/drive/MyDrive/Colab Notebooks/P10/LIANG_Xiuting_3_modele_logreg_code_082022.pkl']
```

Tester le modèle avec le fichier de production

id	resultat_logreg	probabilité
A_1	False	0.062879
A_2	False	0.011737
A_3	False	0.007293
A_4	True	0.999828
A_5	True	0.999997



Le modèle régréssion logistique a de meilleures performances que le modèle Kmeans. Choisir le modèle régréssion logistique comme notre modèle de prévision.

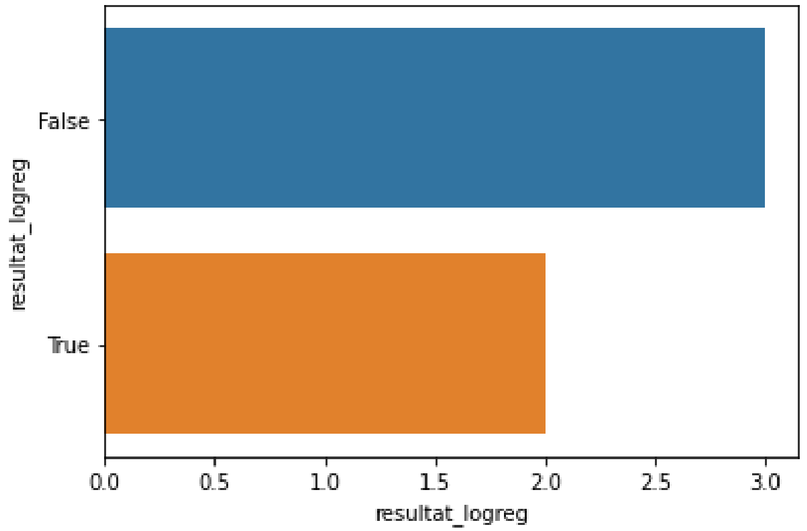
Application

Importation du jeu de données

```
[19] def app_logistic_regression(read_csv) :

       df_production = pd.read_csv(read_csv, index_col = 'id',
                                   usecols = ['id','height_left', 'height_right', 'margin_low', 'margin_up', 'length'])
       df_norm_pro = df_production.copy()
       # Normalisation des données
       numeric_range_pro = ['height_left', 'height_right', 'margin_low', 'margin_up', 'length']
       scaled_org_pro = preprocessing.StandardScaler().fit_transform(df_norm_pro[numeric_range_pro])
       df_norm_pro[numeric_range_pro] = scaled_org_pro

       # Importation le modèle
       logreg = jl.load('/content/drive/MyDrive/Colab Notebooks/P10/LIANG_Xiuting_3_modele_logreg_code_082022.pkl')
       logreg.predict(df_norm_pro)


       # Créer le dataframe
       df_production['resultat_logreg'] = logreg.predict(df_norm_pro)
       df_production['probabilité'] = logreg.predict_proba(df_norm_pro).round(6)[:,1:2]
       df_result = df_production[['resultat_logreg', 'probabilité']]
       df_result = df_result.reset_index()

       return df_result
```

Résultat

	id	resultat_logreg	probabilité
0	A_1	False	0.062879
1	A_2	False	0.011737
2	A_3	False	0.007293
3	A_4	True	0.999828
4	A_5	True	0.999997



Application web

```
if df_production is not None:


  df = pd.read_csv(df_production, index_col = 'id',
                   usecols = ['id','height_left', 'height_right', 'margin_low', 'margin_up', 'length'])
  df_norm_pro = df.copy()
  # Normalisation des données
  numeric_range_pro = ['height_left', 'height_right', 'margin_low', 'margin_up', 'length']
  scaled_org_pro = preprocessing.StandardScaler().fit_transform(df_norm_pro[numeric_range_pro])
  df_norm_pro[numeric_range_pro] = scaled_org_pro

  # Importation le modèle
  logreg.predict(df_norm_pro)

  # Créer le dataframe
  df['resultat_logreg'] = logreg.predict(df_norm_pro)
  df['probabilité'] = logreg.predict_proba(df_norm_pro).round(2)[:,1:2]
  df_result = df[['resultat_logreg', 'probabilité']]
  df_result = df_result.reset_index()

  st.dataframe(df_result)

  df['resultat_logreg'] = df['resultat_logreg'].astype(str)
  df1 = df.groupby('resultat_logreg')['resultat_logreg'].count()
  df1 = pd.DataFrame(df1)

  fig = px.bar(df1, x='resultat_logreg', y = df1.index)
  st.write(fig)
```

Projet 10: Détectez des faux billets avec Python

OpenClassrooms - Data Analyst 2021-2022 - Xiuting LIANG

Le contexte du projet de data analyse

L'Organisation nationale de lutte contre le faux-monnayage, ou ONCFM, est une organisation publique ayant pour objectif de mettre en place des méthodes d'identification des contrefaçons des billets en euros.

L'Notre mission est construire un algorithme qui, à partir des caractéristiques géométriques d'un billet, serait capable de définir si ce dernier est un vrai ou un faux billet.

Introduction des modèles

Nous ultilisons les 5 informations géométriques sur un billet: length, height_left, height_right, margin_up, margin_low.

Un algorithme de la régression logistique classique est ultilisé pour identifier les billets vrais et faux avec présentations des résultats et des probabilités.

Analyser votre fichier csv

Téléchargez votre fichier csv

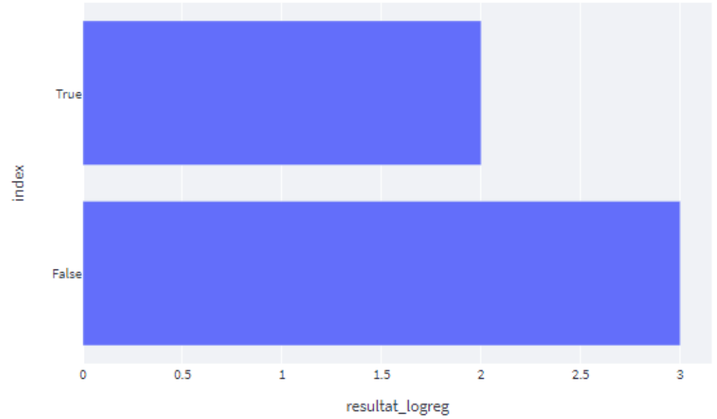
Drag and drop file here
Limit 200MB per file
Browse files

Téléchargez votre fichier csv

Drag and drop file here
Limit 200MB per file
Browse files

billets_production.csv 271.0B

	id	resultat_logreg	probabilité
0	A_1	☐	0.0600
1	A_2	☐	0.0100
2	A_3	☐	0.0100
3	A_4	☑	1.0000
4	A_5	☑	1.0000



ONCFM

Merci !