Algorithme de détection automatique de faux billets

Jérémy GUINAULT

Organisation nationale de lutte contre le faux-monnayage (ONCFM)

Sommaire

- *Présentation du contexte,*
- *Données initiales : présentations et analyse,*
- *Traitements réalisés sur le dataframe,*
- *Présentation des modèles,*
- *Choix du modèle,*
- *Conclusion.*

Présentation du contexte

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

- Lorsqu’un billet arrive, il est scanné par une machine qui consigne toutes ses caractéristiques géométriques.

- Les faux billets présentent des différences géométriques infimes par rapport aux vrais billets.

- Dans le cadre de cette lutte, l’organisation souhaite mettre en place un algorithme capable de différencier automatiquement les vrais billets des faux.

- L’algorithme pourra donc définir si un billet est vrai ou faux à partir de ses caractéristiques géométriques.

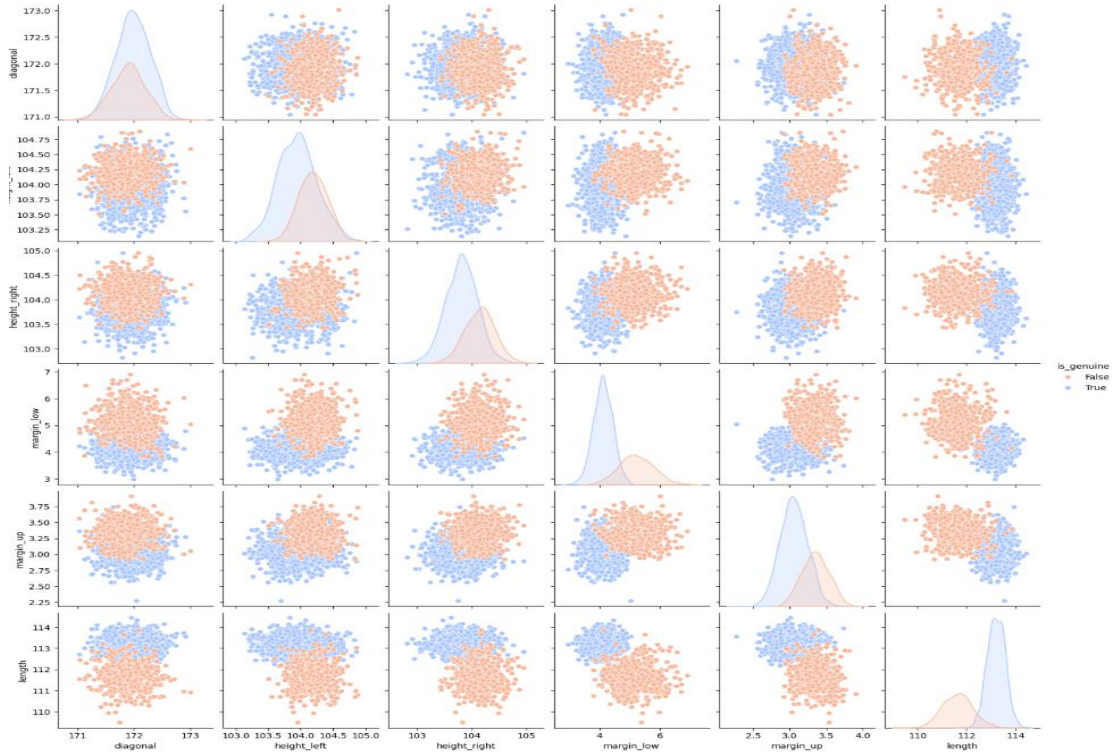
Données initiales : présentations et analyse

- Un fichier fourni : il contient un total de 1500 billets, 1000 vrais et 500 faux,
- Chaque billet dispose de six caractéristiques géométriques : length, height_left, height_right, margin_up, margin_low, diagonal,
- Également une colonne is_genuine qui donne la nature du billet : True ou False.

	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

Données initiales : présentations et analyse

Détails de la distribution : distinction des vrais et faux billets



Données initiales : présentations et analyse

- Vérification de la répartition des billets True / False :

```
Répartition des vrais/faux billets :
is_genuine
True     1000
False     500
Name: count, dtype: int64
```

Les billets sont bien répartis comme annoncés.

- Vérification si présence de valeurs nulles :

```
is_genuine       0
diagonal         0
height_left      0
height_right     0
margin_low      37
margin_up        0
length           0
```

Il y a 37 valeurs manquantes dans la colonne margin_low de notre dataframe.

Traitements réalisés sur le dataframe

- Compléter les 37 valeurs manquantes en se basant sur les autres valeurs du dataframe.
  - Utilisation d'un modèle de régression linéaire ou de régression polynomiale.
  - Pour cela 3 hypothèses doivent etre vérifiées : colinéarité des variables, homoscédasticité, normalité des résidus.

- Colinéarité des variables, calcul du VIF pour vérifier cela :

Variance Inflation Factor (VIF):

	Variable	VIF
0	const	592495.682281
1	diagonal	1.018610
2	height_left	1.151474
3	height_right	1.260286
4	margin_low	1.913278
5	margin_up	1.419672
6	length	2.131067

Le VIF de chaque variable étant inférieur à 5, cela indique qu'il n y a pas de problème de colinéarité sévère, que les variables sont suffisamment indépendantes les unes des autres.

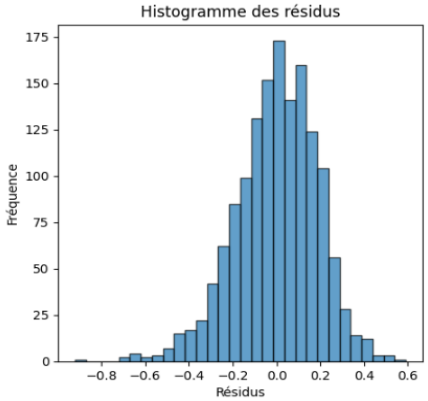
Traitements réalisés sur le dataframe

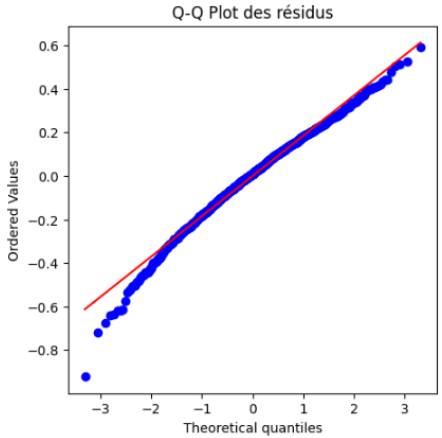
- Homoscédasticité, via le test de Breusch-Pagan :

```
Résultats du test de Breusch-Pagan :
Lagrange multiplier statistic: 90.1118791375326
p-value: 2.87210334291334e-17
f-value: 15.927845106298488
f p-value: 8.370677762793827e-18
```

La p-value est très faible, inférieure à 0,05, cela suggère qu'il y a une hétéroscédasticité significative.

- Normalité des résidus :






```
Test de Kolmogorov-Smirnov :
Statistique : 0.0348
p-value : 5.6251e-02
```

La distribution des résidus semble correcte au niveau des graphiques. Le test de Kolmogorov-Smirnov, avec une p-value > 0,05, ne permet pas de rejeter l'hypothèse de normalité.

Traitements réalisés sur le dataframe

- Régression polynomiale afin de combler les 37 valeurs manquantes.

```python
#Regression polynomiale

#Séparation des données avec et sans valeurs manquantes pour 'margin_low'
df_train = df_billets[df_billets['margin_low'].notnull()]
df_missing = df_billets[df_billets['margin_low'].isnull()]

#Sélection des colonnes prédictives
features = ['is_genuine', 'diagonal', 'height_left', 'height_right', 'margin_up', 'length']
X_train = df_train[features]
y_train = df_train['margin_low']

#Entraînement du modèle
model = make_pipeline(PolynomialFeatures(degree=5), Ridge(alpha=0.1))
model.fit(X_train, y_train)

#Prédiction des valeurs manquantes
X_missing = df_missing[features]
predicted_values = model.predict(X_missing)
```

```
Valeurs manquantes après remplissage :
is_genuine      0
diagonal        0
height_left     0
height_right    0
margin_low      0
margin_up       0
length          0
```

- Les 37 valeurs sont comblées, avec un score de détermination $r^2 = 0,68$. Un score solide qui capture 68% de la variance des données.

- Bien qu'une des hypothèses de validité n'a pas pu etre validée, les valeurs imputées restent cohérentes avec les tendances observées dans le jeu de données.

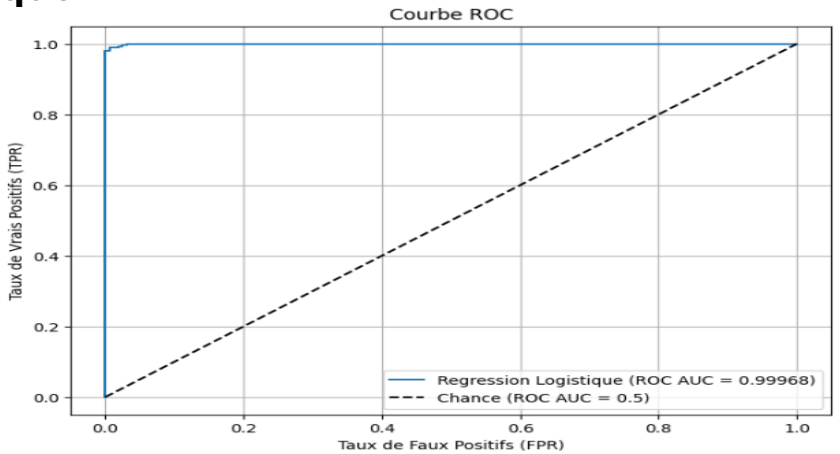
Présentation des modèles

- Dans un premier temps je pars du dataframe, et je le divise en données d'entrainement et test.

```
X = df_billets[['diagonal', 'height_left', 'height_right', 'margin_up', 'margin_low', 'length']]
y = df_billets['is_genuine']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

- Premier modèle, **la régression logistique** :

```
#Regression logistique
log_reg_model = LogisticRegression(random_state=8)
log_reg = log_reg_model.fit(X_train, y_train)

#Prediction des étiquettes True/False
y_pred_log = log_reg.predict(X_test)
#Prédiction des probabilités
y_pred_proba_log = log_reg.predict_proba(X_test)
```

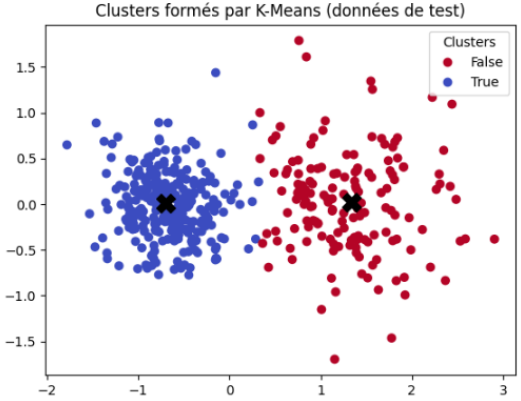


Présentation des modèles

- Deuxième modèle, **K-Means** :

```python
#K-means
kmeans_model = KMeans(n_clusters=2, random_state=808)
kmeans = kmeans_model.fit(X_train)

#Prediction des étiquettes True/False
y_pred_kmeans = kmeans.predict(X_test)

#Réajustement des labels : 1 devient False, 0 devient True
y_pred_kmeans = [False if label == 1 else True for label in y_pred_kmeans]
```

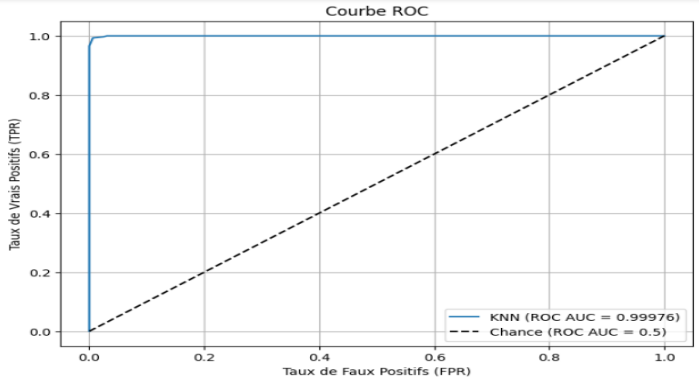
- Si l'on utilise une ACP et que l'on réduit à deux dimensions, cela nous donne la projection suivante :



Présentation des modèles

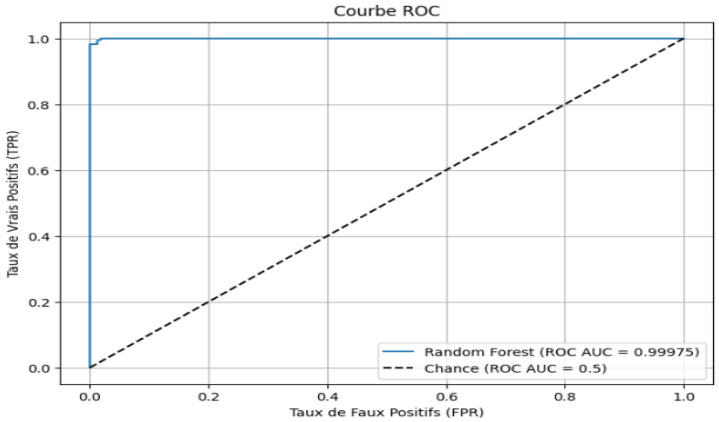
- Troisième modèle, **KNN** :

```
#KNN
knn_model = KNeighborsClassifier(n_neighbors=11)
knn = knn_model.fit(X_train, y_train)

#Prediction des étiquettes True/False
y_pred_knn = knn.predict(X_test)
#Prédiction des probabilités
y_pred_proba_knn = knn.predict_proba(X_test)
```



- Quatrième modèle, **Random forest** :

```
#Random Forest
rf_model = RandomForestClassifier(n_estimators=101, random_state=808)
rf = rf_model.fit(X_train, y_train)

#Prediction des étiquettes True/False
y_pred_rf = rf.predict(X_test)
#Prédiction des probabilités
y_pred_proba_rf = rf.predict_proba(X_test)
```



Présentation des modèles

- J'ai utilisé ces quatre modèles, sur un dataframe de 5 billets afin de prédire leur nature :

	diagonal	height_left	height_right	margin_low	margin_up	length	id	is_genuine_pred_log	proba_true_log	is_genuine_pred_kmeans	is_genuine_pred_knn	proba_true_knn	is_genuine_pred_rf	proba_true_rf
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1	False	0.006864	False	False	0.0	False	0.02
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2	False	0.001406	False	False	0.0	False	0.00
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3	False	0.001737	False	False	0.0	False	0.00
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4	True	0.912309	True	True	1.0	True	0.97
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5	True	0.999557	True	True	1.0	True	0.99

Les quatre modèles ont réalisé les mêmes prédictions pour la nature de ces 5 billets.

Nous ne pouvons donc pas différencier les modèles par rapport à cela, nous allons voir leur matrice de confusion et rapport de classification.

Choix du modèle

- Matrice de confusion et rapport de classification des quatre modèles :

Régression Logistique :
[[156 5]
[1 288]]

	precision	recall	f1-score	support
False	0.99	0.97	0.98	161
True	0.98	1.00	0.99	289
accuracy			0.99	450
macro avg	0.99	0.98	0.99	450
weighted avg	0.99	0.99	0.99	450

k-Means :
[[153 8]
[0 289]]

	precision	recall	f1-score	support
False	1.00	0.95	0.97	161
True	0.97	1.00	0.99	289
accuracy			0.98	450
macro avg	0.99	0.98	0.98	450
weighted avg	0.98	0.98	0.98	450

KNN :
[[157 4]
[1 288]]

	precision	recall	f1-score	support
False	0.99	0.98	0.98	161
True	0.99	1.00	0.99	289
accuracy			0.99	450
macro avg	0.99	0.99	0.99	450
weighted avg	0.99	0.99	0.99	450

Random Forest :
[[157 4]
[0 289]]

	precision	recall	f1-score	support
False	1.00	0.98	0.99	161
True	0.99	1.00	0.99	289
accuracy			0.99	450
macro avg	0.99	0.99	0.99	450
weighted avg	0.99	0.99	0.99	450

- En analysant cela, on peut écarter le modèle K-Means. Si on résume les métriques des trois modèles restants :

```
Résumé des métriques Regression Logistique:
Accuracy: 0.99067 ± 0.0044
F1-Score: 0.99065 ± 0.0044
ROC AUC: 0.99860 ± 0.0017

Résumé des métriques KNN:
Accuracy: 0.99133 ± 0.0045
F1-Score: 0.99131 ± 0.0045
ROC AUC: 0.99503 ± 0.0027

Résumé des métriques Random Forest:
Accuracy: 0.99133 ± 0.0062
F1-Score: 0.99131 ± 0.0062
ROC AUC: 0.99948 ± 0.0003
```

Au vu de ces éléments, le modèle Random Forest semble le plus performant pour la prédiction de la nature des billets.

Conclusion

- Test de plusieurs modèles pour la mise en place de l'algorithme de détection de faux billets.
- Les quatre modèles semblent pouvoir prédire correctement la nature des billets.
- On cherche le modèle le plus performant, notre choix va donc se baser sur le modèle **Random Forest,** au vu des métriques de celui-ci.
- Exportation et mise en place de ce modèle dans l'applicatif pour détecter la nature des billets.