Projet n°10 :

Détection de faux billets avec Python

Sommaire

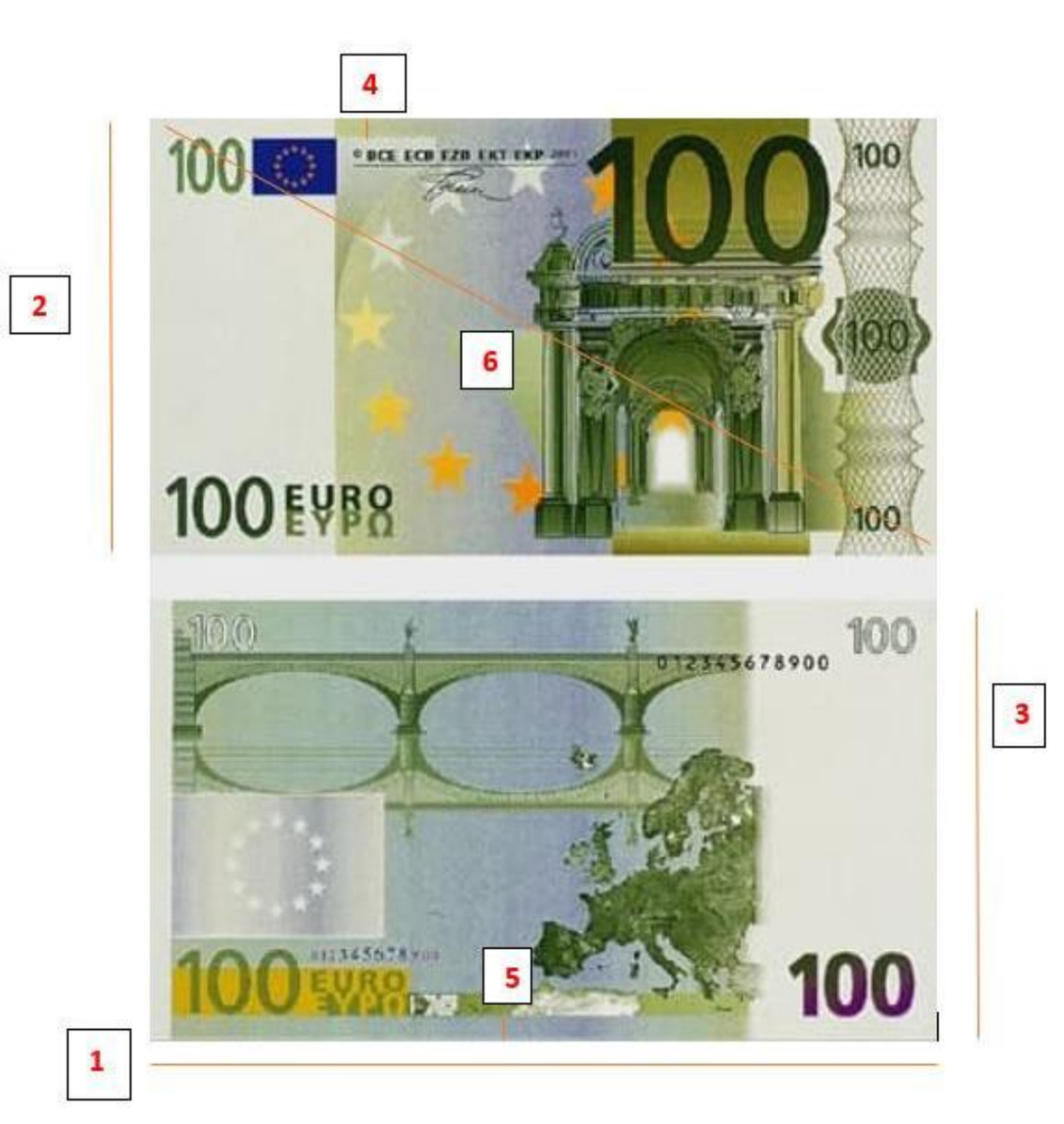
Partie 1 : Objectifs et enjeux de la détéction de faux billets

Selon l'Office central pour la répression du faux-monnayage à la direction centrale de la police judiciaire (France), 700 000 faux billets circuleraient dans la zone euro, dont 30 à 40 % seraient émis depuis la France.
En 2021, 12 billets contrefaits ont été détectés par million de coupures authentiques en circulation.
Les faux billets que l'on retrouve le plus en circulation sont les coupures de 20 et 50 euros, avec régulièrement des pics constatés avec les billets de 100 euros.
L'euro est le billet le plus sécurisé au monde, avec une dizaine de signes de sécurité apparents utilisant les meilleures techniques du moment :

- Filigrane
- Taille douce (le papier est imprimé par une plaque de cuivre dans laquelle divers motifs sont taillés en creux, ce qui procure au dessin un léger relief)
- Fil métallique incorporé
- Vision par transparence (un demi-motif est reproduit sur chaque face et l'image complète n'apparaît que par transparence) ; papier à texture spéciale
- Microlettres ; encres à couleur changeante ou magnétisées
- Hologramme (image en trois dimensions réalisée par des faisceaux laser).

Ces signes sont complétés par des techniques de prévention plus ou moins élaborées, telles que le chiffrage des billets (un billet comporte une lettre, chaque État membre ayant sa propre lettre – u pour la France – suivie de 11 chiffres). Dans une série de billets, le chiffre des unités croît, tandis que celui des dizaines décroît. D'autres signes sont tenus secrets.

Les caractéristiques d'un billet



1) Length : la longueur du billet (en mm)

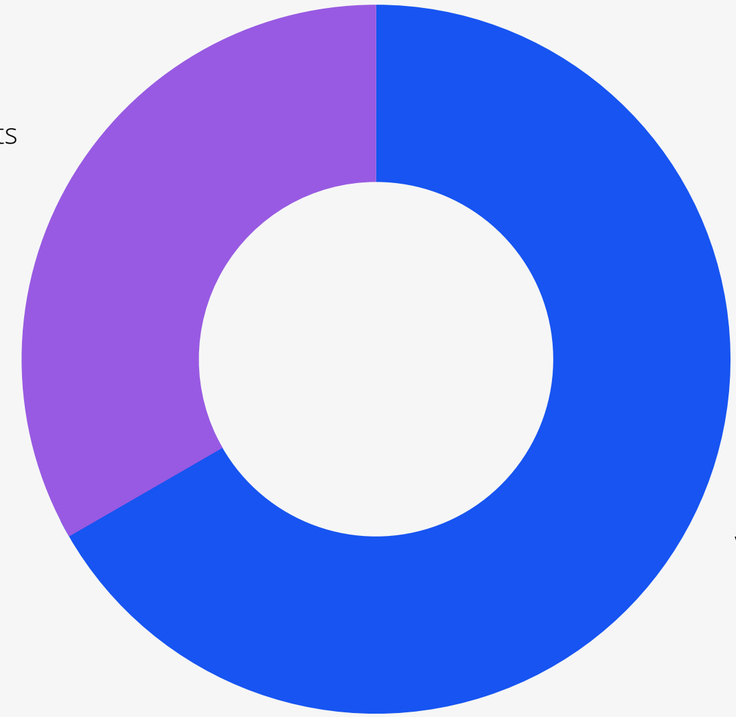
2) Height_left : la hauteur du billet (mesurée sur le côté gauche, en mm)

3) Height_right : la hauteur du billet (mesurée sur le côté droit, en mm)

4) Margin_up : la marge entre le bord supérieur du billet et l'image de celui-ci (en mm)

5) Margin_low : la marge entre le bord inférieur du billet et l'image de celui-ci (en mm)

6) Diagonal : la diagonale du billet (en mm)



Parmis les billets, il y a 37 valeurs manquantes dans la colone "margin_low"

Partie 2 : Régression linéaire

```
# Séparation des données manquantes :
Reg_Lin_Train_df = Billet_df.dropna()
Reg_Lin_Test_df = Billet_df[Billet_df.isnull().any(axis=1)]

# Création de X_train et Y_train:
x_train = Reg_Lin_Train_df.drop("margin_low", axis=1)
y_train = Reg_Lin_Train_df["margin_low"]

# Régression linéaire:
reg_lin = LinearRegression()
model_reg_lin = reg_lin.fit(x_train, y_train)

# Prédictions:
x_train = Reg_Lin_Train_df[["is_genuine","diagonal","height_left","height_right", "margin_up", "length"]]
Reg_Lin_Train_df["margin_low_pred"] = reg_lin.predict(x_train)
```

1

Utilisation du modèle de régression linéaire multivariables pour traiter les 37 valeurs manquantes.

Nous allons séparer le dataframe original en 2:

- Un dataframe sans valeur manquantes que nous utiliserons pour notre régression liniéaire
- Un dataframe avec uniquement les valeurs manquantes pour appliquer notre modèle

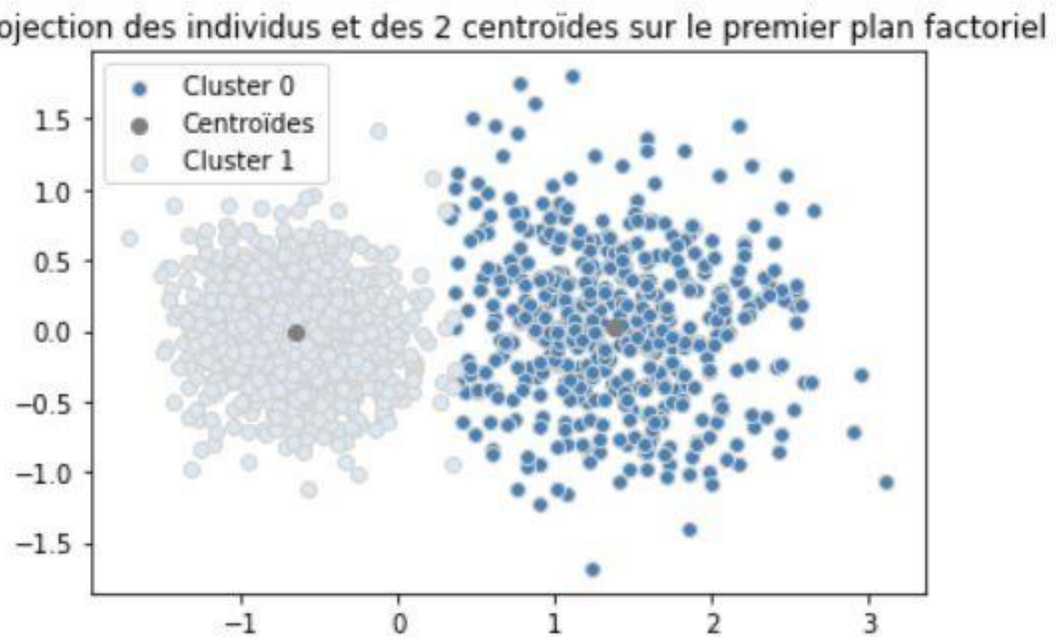
2

Vérification de la validité du modèle à l'aide de :

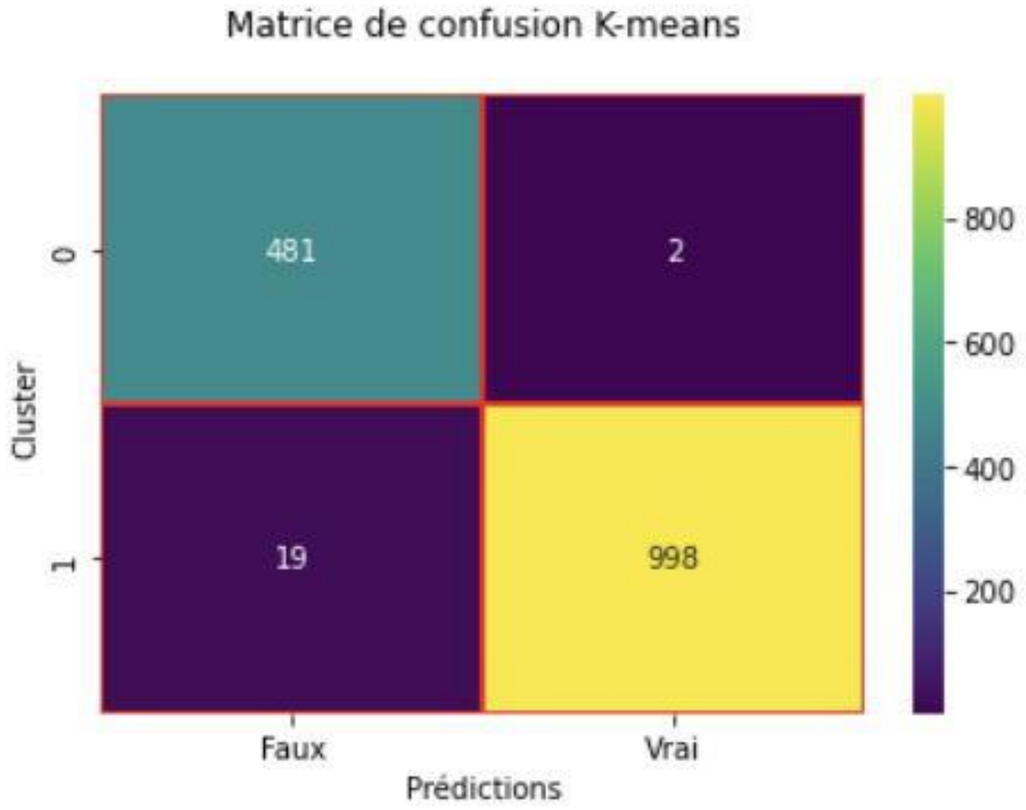
- OLS results
- Normalité de la distribution des résidus
- Homocédasticité
- Autocorrélation
- Multicollinéarité

Partie 3 : K means & Matrice de confusion

Projection selon 2 clusters : vrais et faux billiets



is_genuine	False	True
row_0		
0	481	2
1	19	998



Parmis les 500 faux billets, 19 ont été détectés comme étant vrais

Parmis les 1000 vrais billets, 2 ont été détectés comme étant faux

Partie 4 : KNN

```python
from sklearn import model_selection
# Fixer les valeurs des hyperparamètres à tester
param_grid =  {'n_neighbors': range(2,16) }
# Choisir un score à optimiser, ici l'accuracy (proportion de prédictions correctes)
score = 'accuracy'

# Création classifieur kNN
clf1 = model_selection.GridSearchCV(
    neighbors.KNeighborsClassifier(), # un classifieur kNN
    param_grid,     # hyperparamètres à tester
    cv=5,           # nombre de folds de validation croisée
    scoring=score   # score à optimiser
)

# Optimisation
clf1.fit(X_train_std, y2train.values.ravel())

# Affichage des hyperparamètres optimaux
print("Meilleur(s) hyperparamètre(s) sur le jeu d'entraînement:")
print(clf1.best_params_)

# Affichage des performances correspondantes
print("Résultats de la validation croisée :")
for mean, std, params in zip(
        clf1.cv_results_['mean_test_score'], # score moyen
        clf1.cv_results_['std_test_score'],  # écart-type du score
        clf1.cv_results_['params']           # valeur de l'hyperparamètre
    ):

    print("{} = {:.3f} (+/-{:.03f}) for {}".format(
        score,
        mean,
        std*2,
        params
    ) )
```

Résultat :

```python
from sklearn.metrics import confusion_matrix
# Affichage des : vrais négatifs, vrais positifs, faux négatifs, faux positifs
tn0, fp0, fn0, tp0 = confusion_matrix(y2test,clf1.predict(X_test_std)).ravel()
tn0, fp0, fn0, tp0
print('Vrais négatifs :',tn0,"\n"'Vrais positifs :',tp0,"\n"'Faux négatifs :',fn0,"\n"'Faux positifs :',fp0)
```

```
Vrais négatifs : 93
Vrais positifs : 206
Faux négatifs : 0
Faux positifs : 1
```

Partie 5 : Régression logistique

1

```
# y (authenticité):
y_billet = Billet_df.loc[:, Billet_df.columns == "is_genuine"]
print(y_billet.value_counts())

# X (les autres):
X_billet = Billet_df.loc[:, Billet_df.columns != "is_genuine"]
X_billet = sm.tools.add_constant(X_billet)
print(X_billet.head())
```

2

```
# Régression logistique:
reg_log = Logit(endog=y_billet,exog=X_billet)

# Calculs:
res_log = reg_log.fit()

# Résumé des resultats:
print(res_log.summary())
```

3

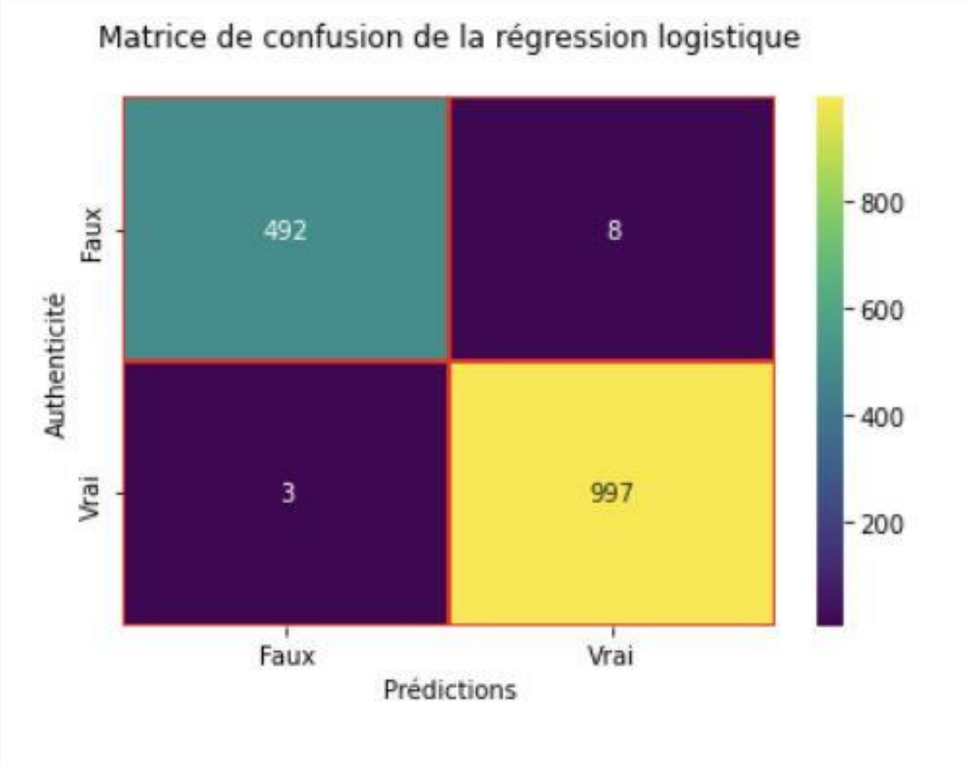
```
# On ne garde que les variables significatives a alpha = 5% dans X_Billet:
X_billet = Billet_df[["height_right","margin_low", "margin_up", "length"]]
X_billet = sm.add_constant(X_billet)

# On recommence la régression logistique:
reg_log = Logit(endog=y_billet,exog=X_billet)

# Modèle:
model_reg_log = reg_log.fit()

# Résultats:
print(model_reg_log.summary2())
```

4



Parmis les 500 faux billets, 8 ont été détectés comme étant vrais

Parmis les 1000 vrais billets, 3 ont été détectés comme étant faux

Test avec la régression logistique sur les données test

Nous avons sélectionner la régression logistique.

Afin de vérifier notre modèle, nous le testons avec des données test et des données que j'ai également inventé de mon côté.

Les résultats sont probants.

```
# Test avec la régression logistique sur les données test fournies (billets_production) :

Billet_test_df = pd.read_csv("billets_production.csv")


X_test = Billet_test_df[["height_right","margin_low","margin_up","length"]]
X_test = sm.add_constant(X_test)
Billet_test_df["proba"] = model_reg_log.predict(X_test)
Billet_test_df["y_pred"] = (model_reg_log.predict(X_test) >= 0.5).astype(int)
print(Billet_test_df[["id","proba","y_pred"]])


print("\nIndetification des billets:")
for i, k in zip(Billet_test_df["y_pred"],Billet_test_df["id"]):
    if i == 1:
        print("Le billet","{}".format(k),"est vrai")
    else:
        print("Le billet","{}".format(k),"est faux")
```

```
    id  proba  y_pred
0  A_1  0.000       0
1  A_2  0.000       0
2  A_3  0.000       0
3  A_4  0.996       1
4  A_5  1.000       1

Indetification des billets:
Le billet A_1 est faux
Le billet A_2 est faux
Le billet A_3 est faux
Le billet A_4 est vrai
Le billet A_5 est vrai
```