
# détectez des faux billets avec Python

Ebru Mandrou

# table des matières

# exploration de données

- 1500 billets avec 7 variables :
  - 1 Qualitative : is_geniune
  - 6 Quantitatives
- **1000** vrais billets
- **500** faux billets
- is_genuine : deux valeurs possibles
  - La valeur **True**
  - La valeur **False**

# exploration de données

- conversion dtype pour 6 variables

```
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   is_genuine    1500 non-null   bool
 1   diagonal      1500 non-null   object
 2   height_left   1500 non-null   object
 3   height_right  1500 non-null   object
 4   margin_low    1463 non-null   object
 5   margin_up     1500 non-null   object
 6   length        1500 non-null   object
dtypes: bool(1), object(6)
memory usage: 71.9+ KB
```

```
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
dtypes: bool(1), float64(6)
memory usage: 71.9 KB
```

# preparation de données

```
billet.isnull().sum()
```

```
is_genuine       0
diagonal         0
height_left      0
height_right     0
margin_low      37
margin_up        0
length           0
dtype: int64
```

- Il existe 37 valeurs nulles pour la colonne margin_low.
- Afin de remplacer ces valeurs nulles, une régression linéaire est appliquée.

# preparation de données

## Régression Linéaire

**Dépendent variables :** variables avec les valeurs nulles (test_data)

**Independent variables :** variables avec les valeurs non-nulles (train_data)

| | is_genuine | diagonal | height_left | height_right | margin_low | margin_up | length | margin_low_new |
|---|---|---|---|---|---|---|---|---|
| 72 | True | 171.94 | 103.89 | 103.45 | NaN | 3.25 | 112.79 | 4.064954 |
| 99 | True | 171.93 | 104.07 | 104.18 | NaN | 3.14 | 113.08 | 4.111990 |
| 151 | True | 172.07 | 103.80 | 104.38 | NaN | 3.02 | 112.93 | 4.134003 |
| 197 | True | 171.45 | 103.66 | 103.80 | NaN | 3.62 | 113.27 | 3.993571 |
| 241 | True | 171.83 | 104.14 | 104.06 | NaN | 3.02 | 112.36 | 4.140399 |
| 251 | True | 171.80 | 103.26 | 102.82 | NaN | 2.95 | 113.22 | 4.094284 |
| 284 | True | 171.92 | 103.83 | 103.76 | NaN | 3.23 | 113.29 | 4.074124 |
| 334 | True | 171.85 | 103.70 | 103.96 | NaN | 3.00 | 113.36 | 4.125390 |
| 410 | True | 172.56 | 103.72 | 103.51 | NaN | 3.12 | 112.95 | 4.080728 |
| 413 | True | 172.30 | 103.66 | 103.50 | NaN | 3.16 | 112.95 | 4.073633 |
| 445 | True | 172.34 | 104.42 | 103.22 | NaN | 3.01 | 112.97 | 4.118973 |
| 481 | True | 171.81 | 103.53 | 103.96 | NaN | 2.71 | 113.99 | 4.180380 |
| 505 | True | 172.01 | 103.97 | 104.05 | NaN | 2.98 | 113.65 | 4.136484 |
| 611 | True | 171.80 | 103.68 | 103.49 | NaN | 3.30 | 112.84 | 4.051068 |
| 654 | True | 171.97 | 103.69 | 103.54 | NaN | 2.70 | 112.79 | 4.178377 |
| 675 | True | 171.60 | 103.85 | 103.91 | NaN | 2.56 | 113.27 | 4.225551 |
| 710 | True | 172.03 | 103.97 | 103.86 | NaN | 3.07 | 112.65 | 4.115868 |

# description des données

## Analyse Univariée

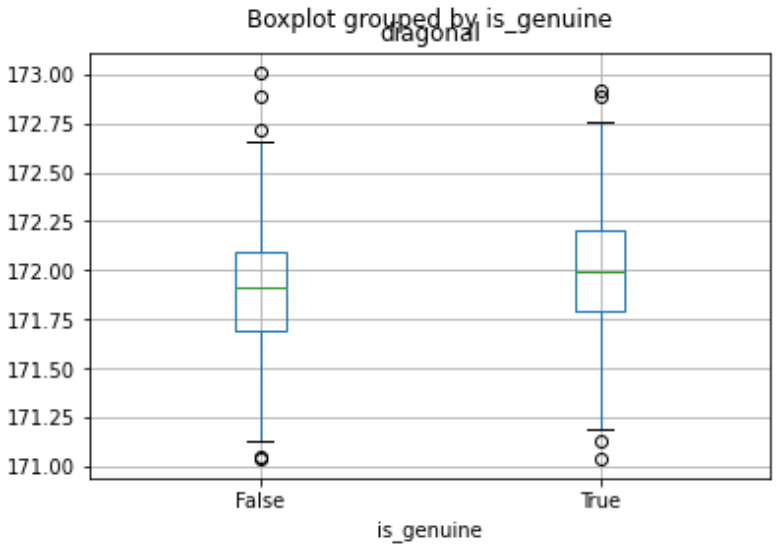
- *Sommaire statistique*
  - Les écarts-types sont faibles

```
data.describe().round(2)
```

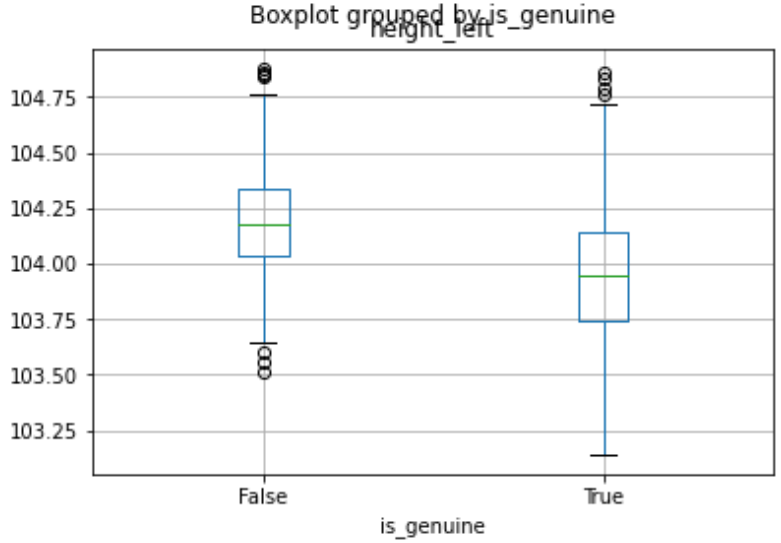
| | diagonal | height_left | height_right | margin_low | margin_up | length |
|---|---|---|---|---|---|---|
| count | 1500.00 | 1500.00 | 1500.00 | 1500.00 | 1500.00 | 1500.00 |
| mean | 171.96 | 104.03 | 103.92 | 4.48 | 3.15 | 112.68 |
| std | 0.31 | 0.30 | 0.33 | 0.66 | 0.23 | 0.87 |
| min | 171.04 | 103.14 | 102.82 | 2.98 | 2.27 | 109.49 |
| 25% | 171.75 | 103.82 | 103.71 | 4.03 | 2.99 | 112.03 |
| 50% | 171.96 | 104.04 | 103.92 | 4.31 | 3.14 | 112.96 |
| 75% | 172.17 | 104.23 | 104.15 | 4.87 | 3.31 | 113.34 |
| max | 173.01 | 104.88 | 104.95 | 6.90 | 3.91 | 114.44 |

# description des données

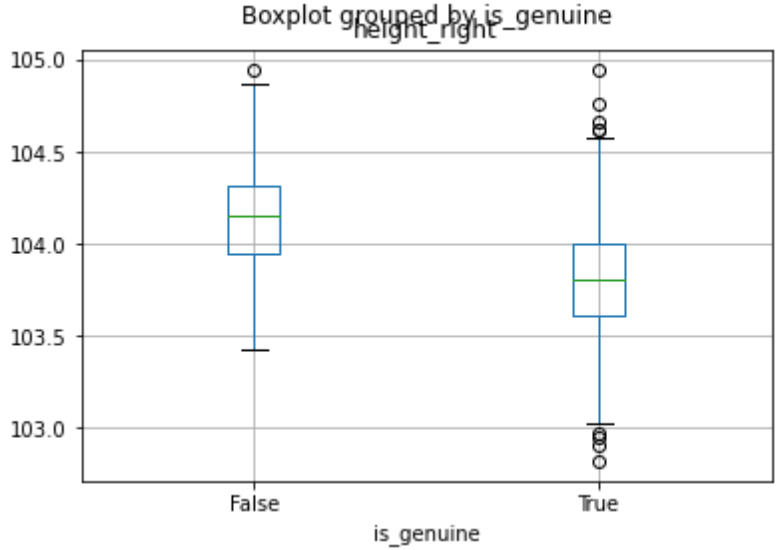
## Analyse Univariée




Diagonale



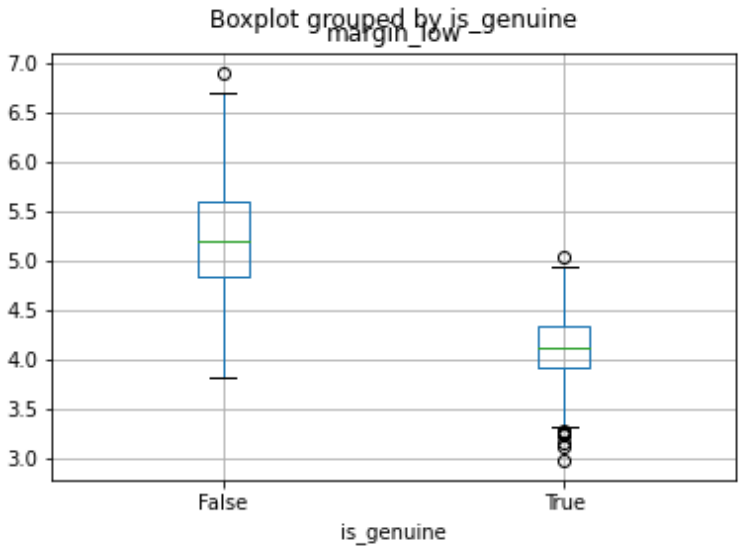

Hauteur gauche



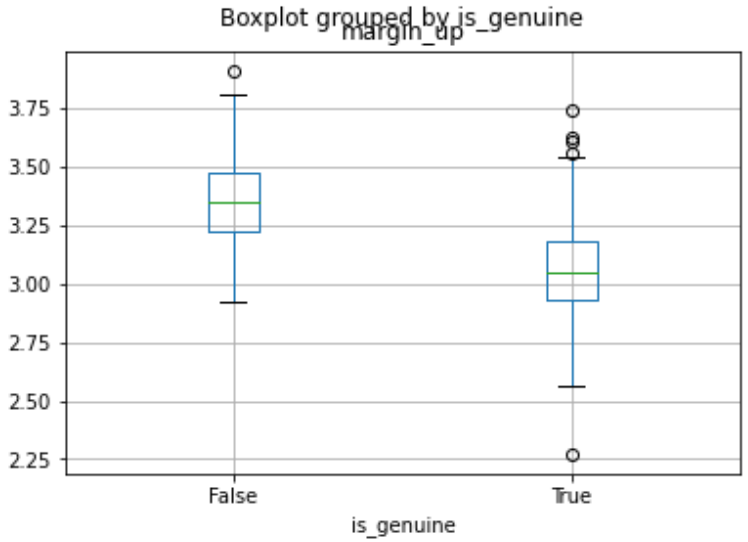

Hauteur droite

# description des données

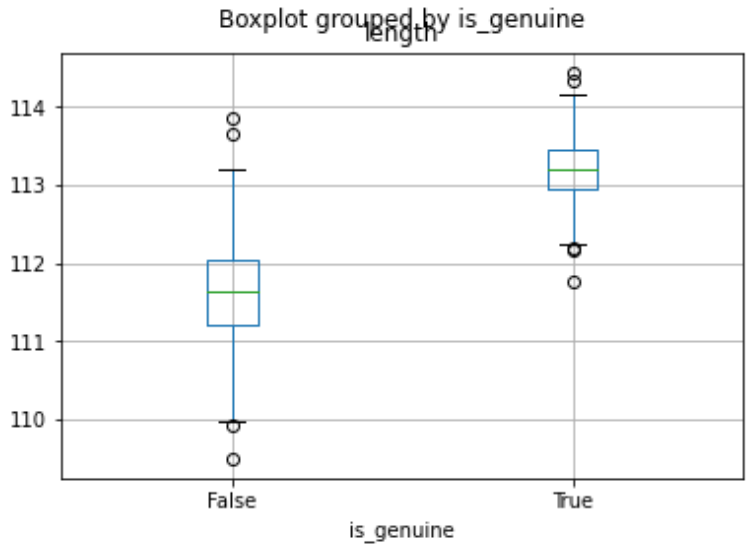
## Analyse Univariée




Marge Basse




Marge Haute




La longueur du billet

# description des données

## Les variables importantes

| is_genuine | diagonal | height_left | height_right | margin_low | margin_up | length |
|---|---|---|---|---|---|---|
| False | 171.90 | 104.19 | 104.14 | 5.22 | 3.35 | 111.63 |
| True | 171.99 | 103.95 | 103.81 | 4.12 | 3.05 | 113.20 |
| | 0.09 | 0.24 | 0.33 | 1.10 | 0.30 | 1.57 |

- Diagonal : Nothing very noteworthy
- Left height : Nothing very noteworthy
- Right height : Nothing very noteworthy
- High Margin : Nothing very noteworthy
- Lower Margin : A significant difference
- Length : A significant difference

# description des données

## Analyse Bivariée

- Corrélation des Variables

  - ***Corrélation négative modérée*** entre la marge inférieure et la longueur : **-0.67**
  - ***Corrélation négative modérée*** entre la marge supérieure et la longueur : **-0.52**
  - ***Faible corrélation positive*** entre la marge supérieure et la marge inférieure : **0.43**

| | diagonal | height_left | height_right | margin_low | margin_up | length |
|---|---|---|---|---|---|---|
| diagonal | 1.00 | 0.02 | -0.02 | -0.11 | -0.06 | 0.10 |
| height_left | 0.02 | 1.00 | 0.24 | 0.30 | 0.25 | -0.32 |
| height_right | -0.02 | 0.24 | 1.00 | 0.39 | 0.31 | -0.40 |
| margin_low | -0.11 | 0.30 | 0.39 | 1.00 | 0.43 | -0.67 |
| margin_up | -0.06 | 0.25 | 0.31 | 0.43 | 1.00 | -0.52 |
| length | 0.10 | -0.32 | -0.40 | -0.67 | -0.52 | 1.00 |

# description des données

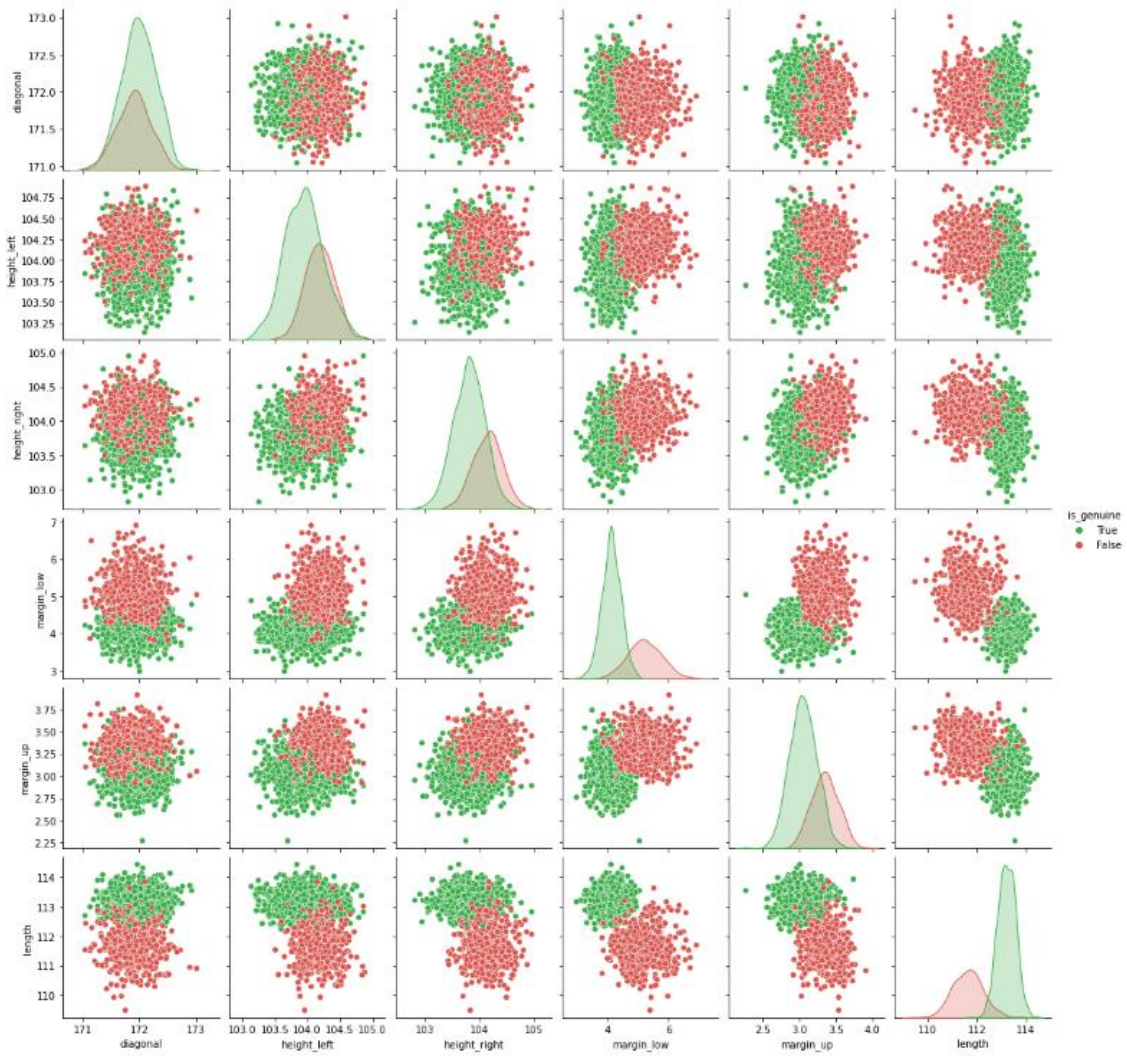
## Analyse Bivariée

- Pair Plot

- la marge basse des faux billets **>** la marge basse des vrais billets

- la marge supérieure des faux billets **>** la marge supérieure des vrais billets

- la longueur des faux billets **<** la longueur des vrais billets



# description des données

## Test ANOVA

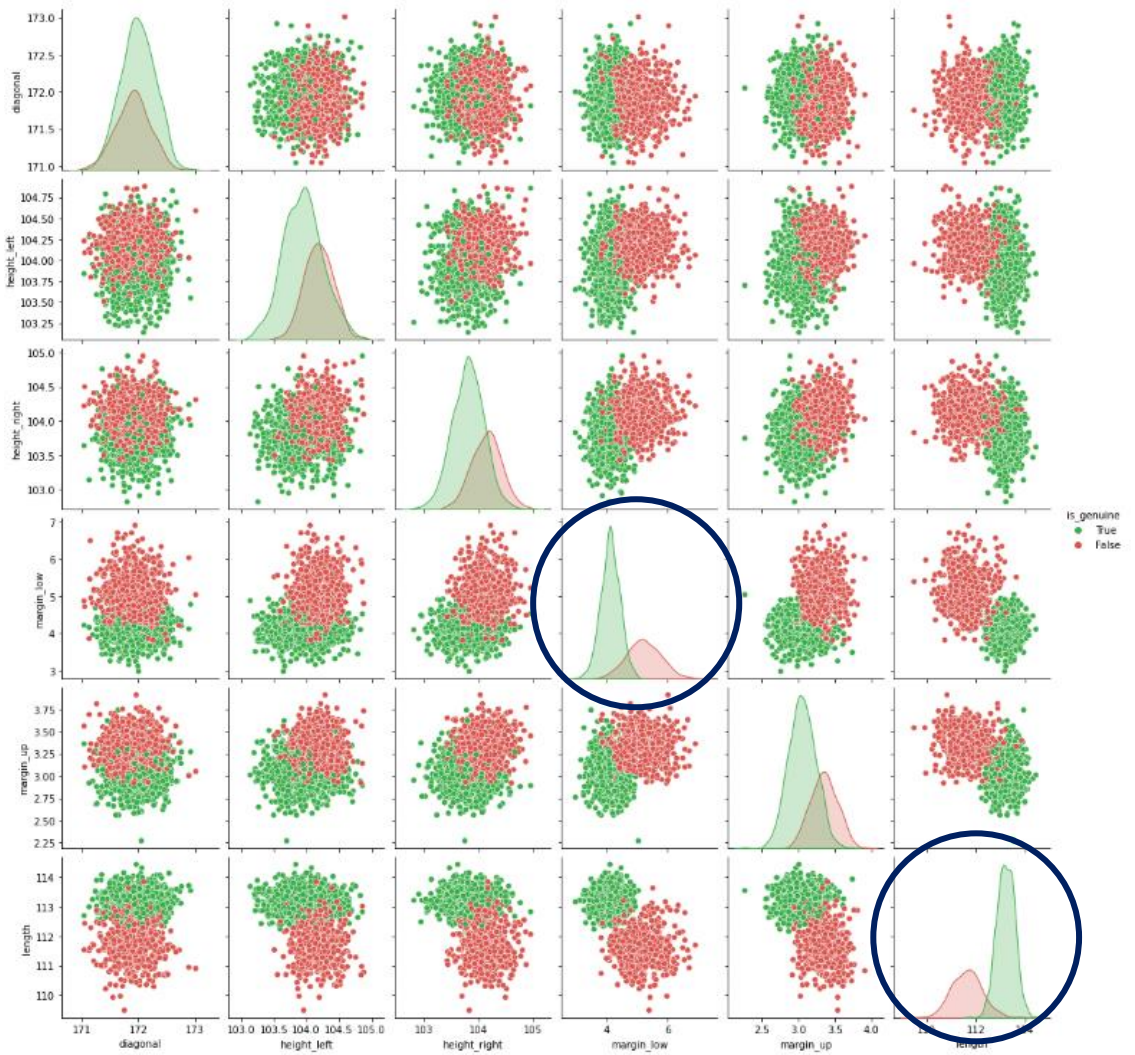
**H0:** Means of true and false banknotes are equal
**H1:** Means they are not equal

### margin_low

FSTAT = 2424.22, pvalue = 0.0
We reject H0.
at 95% confidence level.

### length

FSTAT = 3876.65, pvalue = 0.0
We reject H0.
at 95% confidence level.



# classification

## Régression Logistique

1. Division des données

   - Train : 70%
   - Test : 30%

2. Développement et prédiction de modèles

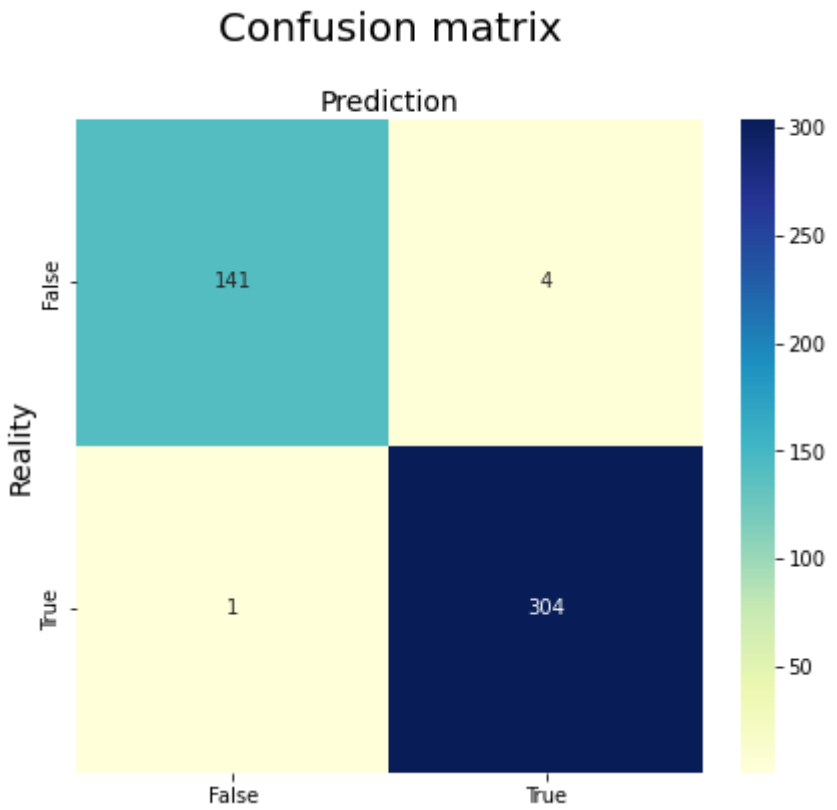
```
# We create an object of the Logistic Regression class.
our_model = LogisticRegression()

# We create our model.
# We train it with our training data
y_train = pd.DataFrame(y_train)
y_train_array = y_train.iloc[:,0].values
our_model.fit(X_train, y_train_array)

# We make the prediction
# with .predict()
y_pred = our_model.predict(X_test)
```

# classification

## Régression Logistique

3. Evaluation de modèles

- Indicateurs pour évaluer le model:
  **Exactitude** : 0.99
  **Précision**: 0.99
  **Sensibilité**: 1.0
  **Spécifité**: 0.97



# classification

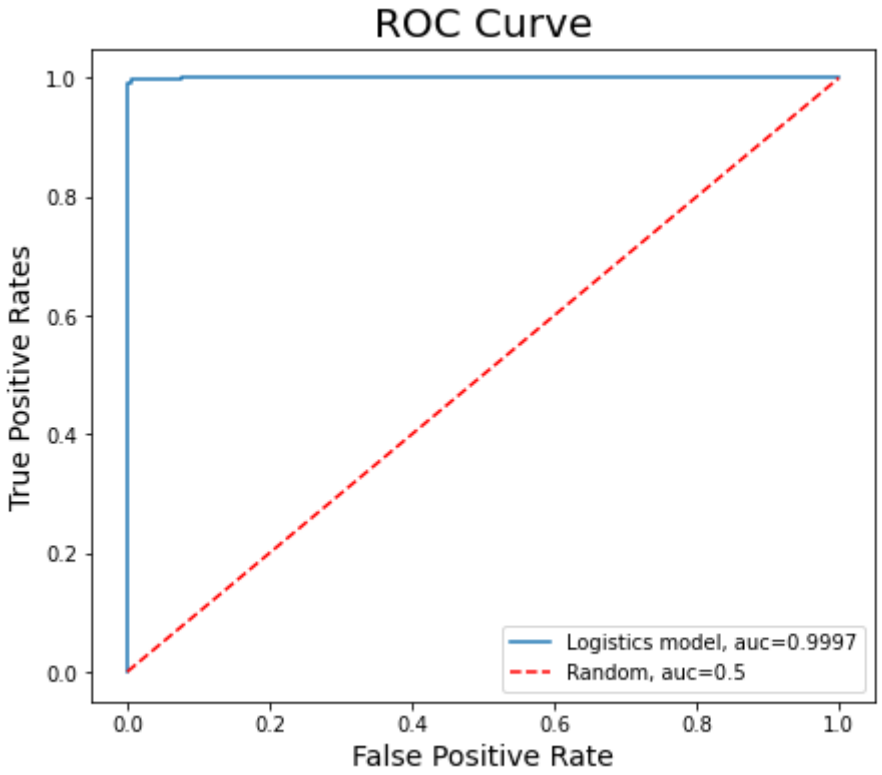
## Logistique Regression

### 4. Courbe ROC

La courbe ROC est une mesure de la performance d'un classificateur binaire.

On représente la mesure ROC sous la forme d'une courbe qui donne le taux de vrais positifs en fonction du taux de faux positifs.

- AUC = 1 ➔ classificateur parfait
- AUC = 0.5 ➔ classificateur inutile
- **Notre AUC = 0.9997**



# classification

## Algorithme SMOTE

- Pour balancer la proportion de billets véridiques et de billets non-véridiques dans nos données d'entrainement.

```
Before number of false banknotes: 355
Before number of true banknotes: 695
Before proportion of false banknotes: 0.3380952380952381
Before proportion of true banknotes: 0.6619047619047619
```

```
Number of false banknotes: 695
Number of true banknotes: 695
Proportion of false banknotes: 0.5
Proportion of true banknotes: 0.5
```

```
# We create an object of the Logistic Regression class.
our_model1 = LogisticRegression()

# We create our model.
# We train it with our training data
y_train_array = y_train.iloc[:,0].values
our_model1.fit(X_train, y_train_array)

# We make the prediction
# with .predict()
y_pred = our_model1.predict(X_test)
```

# classification

## Algorithme SMOTE

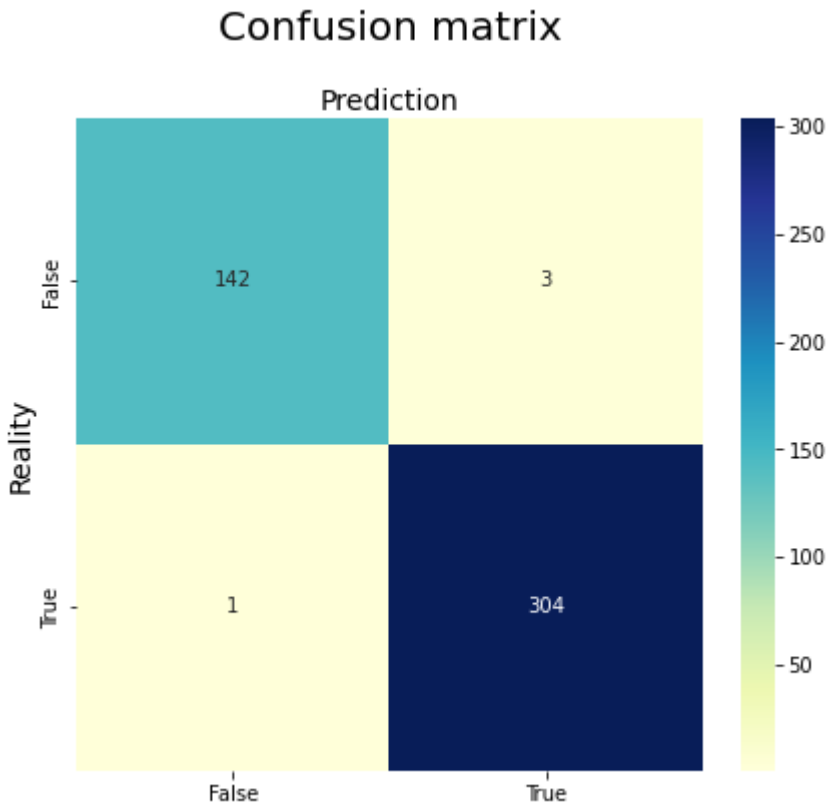
- Evaluation de Model
  - Indicateurs pour évaluer le model:

    **Exactitude** : 0.99 (vs 0.99)

    **Précision**: 0.99 (vs 0.99)

    **Sensibilité**: 1.0 (vs 1.0)

    **Spécifité**: 0.98 (vs 0.97)




**La régression logistique avec l'algorithme SMOTE est légèrement meilleure que la régression logistique.**

# classification

## K-NN

1. Division des données

   - Train : 70%
   - Test : 30%

2. Développement et prédiction de modèles

**Model development and prediction**

```
# We create an object of the K-NN class.
knn = KNeighborsClassifier(n_neighbors=2)
```

```
# We create our model.
# We train it with our training data
y_train = pd.DataFrame(y_train)

y_train_array = y_train.iloc[:,0].values
knn.fit(X_train, y_train_array)
```

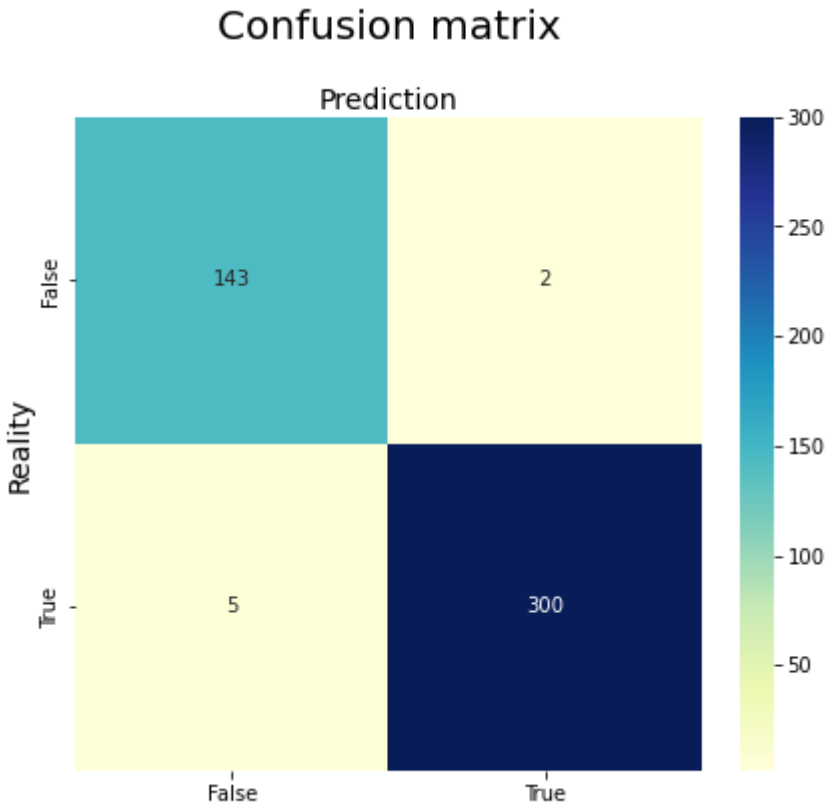
# classification

## K-NN

3. Evaluation de modèles

- Indicateurs pour évaluer le model:
  **Exactitude** : 0.98 (vs 0.99)
  **Précision**: 0.99 (vs 0.99)
  **Sensibilité**: 0.98 (vs 1.0)
  **Spécifité**: 0.99 (vs 0.97)

**La régression logistique est globalement meilleure dans notre cas**



# classification

## K-Means

1. Effectuer K-Means Clustering

   - Pour toutes nos observations, nous avons une valeur observée "vraie" et une valeur prédite "k_means".

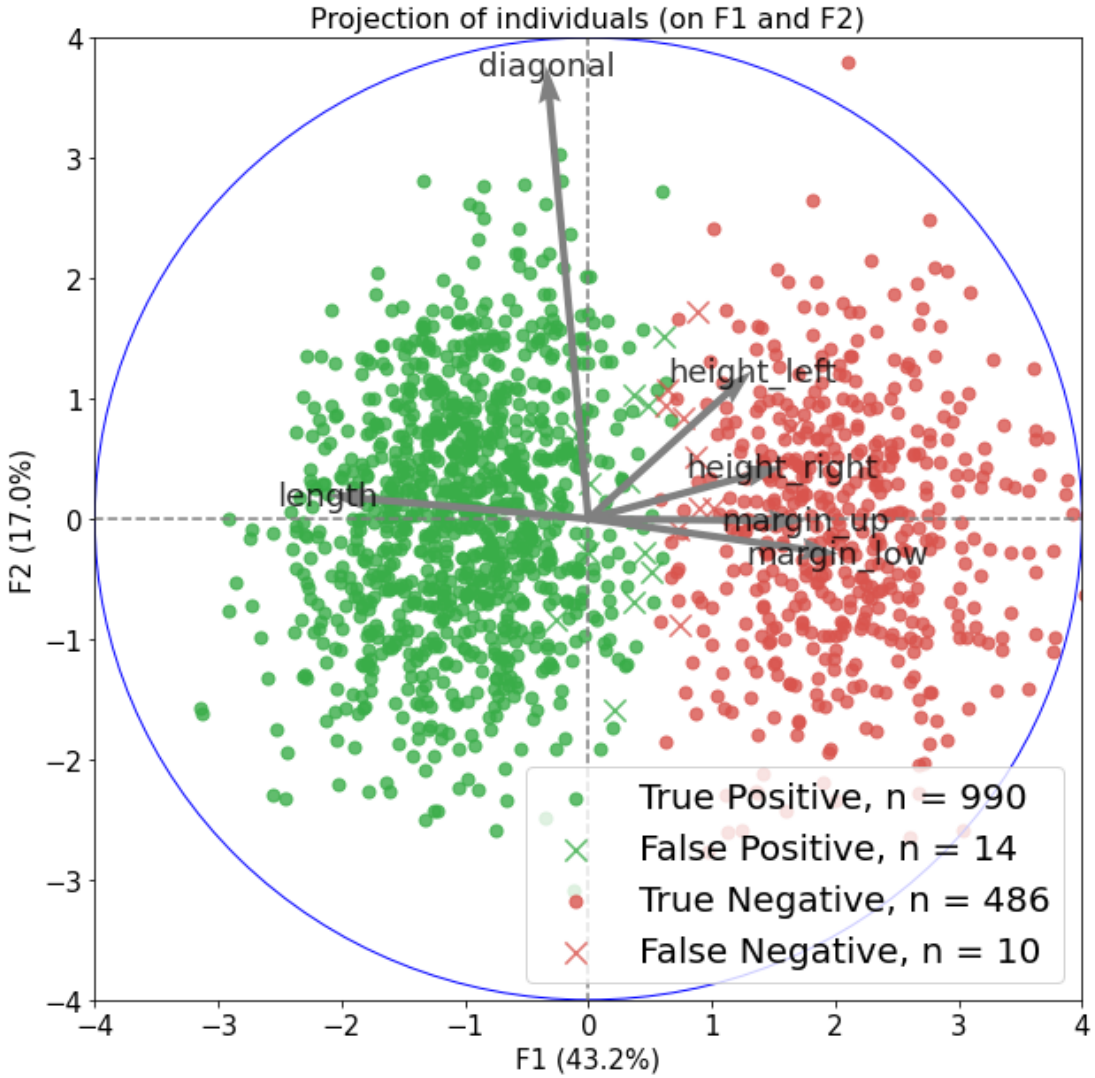
| | is_genuine | diagonal | height_left | height_right | margin_low | margin_up | length | k_mean |
|---|---|---|---|---|---|---|---|---|
| 0 | True | 171.81 | 104.86 | 104.95 | 4.52 | 2.89 | 112.83 | False |
| 1 | True | 171.46 | 103.36 | 103.66 | 3.77 | 2.99 | 113.09 | True |
| 2 | True | 172.69 | 104.48 | 103.50 | 4.40 | 2.94 | 113.16 | True |
| 3 | True | 171.36 | 103.91 | 103.94 | 3.62 | 3.01 | 113.51 | True |
| 4 | True | 171.73 | 104.28 | 103.46 | 4.04 | 3.48 | 112.54 | True |
| 5 | True | 172.17 | 103.74 | 104.08 | 4.42 | 2.95 | 112.81 | True |
| 6 | True | 172.34 | 104.18 | 103.85 | 4.58 | 3.26 | 112.81 | True |
| 7 | True | 171.88 | 103.76 | 104.08 | 3.98 | 2.92 | 113.08 | True |
| 8 | True | 172.47 | 103.92 | 103.67 | 4.00 | 3.25 | 112.85 | True |
| 9 | True | 172.47 | 104.07 | 104.02 | 4.04 | 3.25 | 113.45 | True |

# classification

## K-Means

2. Visualisation des Indicateurs

- Vrais négatifs : négatifs correctement prédits
- Vrais positifs : positifs  correctement prédits
- Faux négatifs : négatifs mal prédits
- Faux positifs : positifs mal prédits

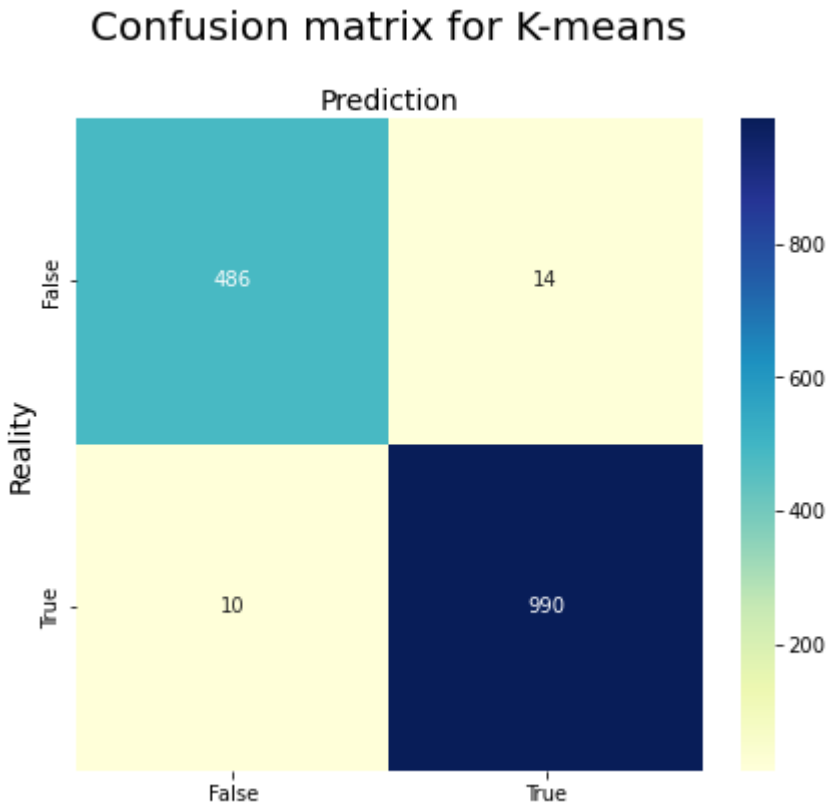


# classification

## K-Means

3. Evaluation de modèles

- Indicateurs pour évaluer le model:
  **Exactitude** : 0.98 (vs 0.99)
  **Précision**: 0.99 (vs 0.99)
  **Sensibilité**: 0.99 (vs 1.0)
  **Spécifité**: 0.97 (vs 0.97)




**La régression logistique est globalement meilleure que la classification K-Means**

# programme de détection de faux billets

- Le jeu test :

| | diagonal | height_left | height_right | margin_low | margin_up | length | id |
|---|---|---|---|---|---|---|---|
| 0 | 171.76 | 104.01 | 103.54 | 5.21 | 3.30 | 111.42 | A_1 |
| 1 | 171.87 | 104.17 | 104.13 | 6.00 | 3.31 | 112.09 | A_2 |
| 2 | 172.00 | 104.58 | 104.29 | 4.99 | 3.39 | 111.57 | A_3 |
| 3 | 172.49 | 104.55 | 104.34 | 4.44 | 3.03 | 113.20 | A_4 |
| 4 | 171.65 | 103.63 | 103.56 | 3.77 | 3.16 | 113.33 | A_5 |

# programme de détection de faux billets

```
# Data for prediction
X_to_predict = df_data.iloc[:, 0:6]

# Prediction with our model.
y_pred = our_model.predict(X_to_predict)
true_probability = our_model.predict_proba(X_to_predict)[:, 1]

# Adding a column with the probability that the sample is true according to the prediction
df_data["true_probability"] = np.round(true_probability, 2)

# Adding a column with the prediction
df_data["true_prediction"] = y_pred
```

```
df_data
```

| | diagonal | height_left | height_right | margin_low | margin_up | length | id | true_probability | true_prediction |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 171.76 | 104.01 | 103.54 | 5.21 | 3.30 | 111.42 | A_1 | 0.01 | False |
| **1** | 171.87 | 104.17 | 104.13 | 6.00 | 3.31 | 112.09 | A_2 | 0.00 | False |
| **2** | 172.00 | 104.58 | 104.29 | 4.99 | 3.39 | 111.57 | A_3 | 0.00 | False |
| 3 | 172.49 | 104.55 | 104.34 | 4.44 | 3.03 | 113.20 | A_4 | 0.91 | True |
| **4** | 171.65 | 103.63 | 103.56 | 3.77 | 3.16 | 113.33 | A_5 | 1.00 | True |


merci