# Projet 6

# Détectez des faux billets

## Detect counterfeit banknotes

# Notre objectif

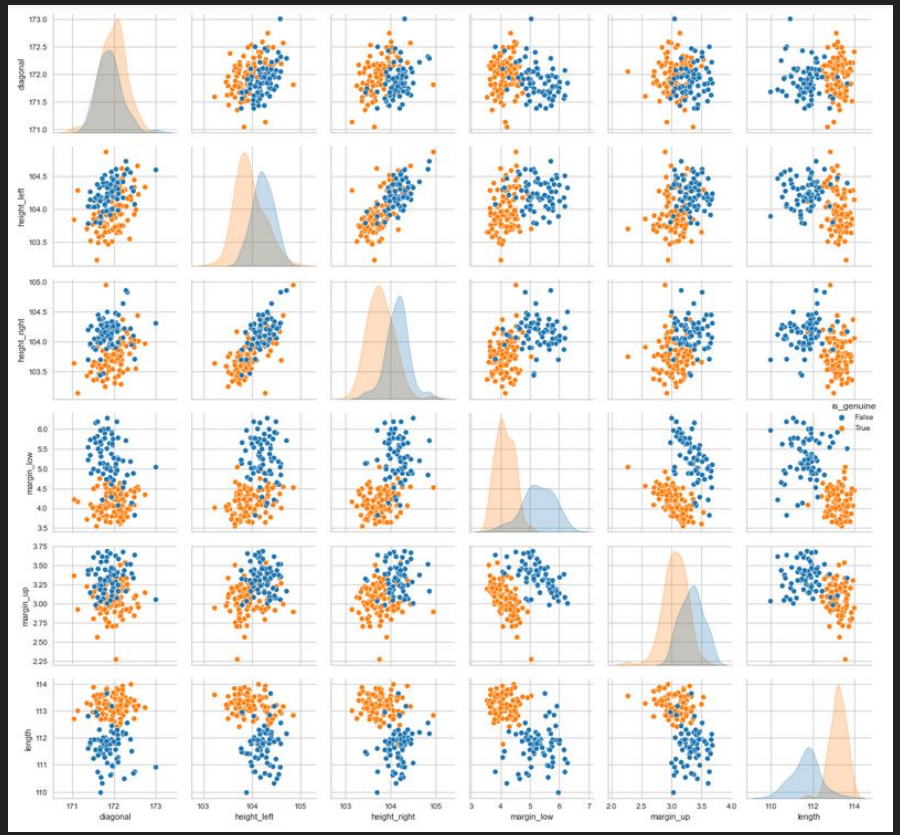
Créer un algorithme permettant au gouvernement de lutter contre le crime organisé et ses systèmes de faux billets de banque.

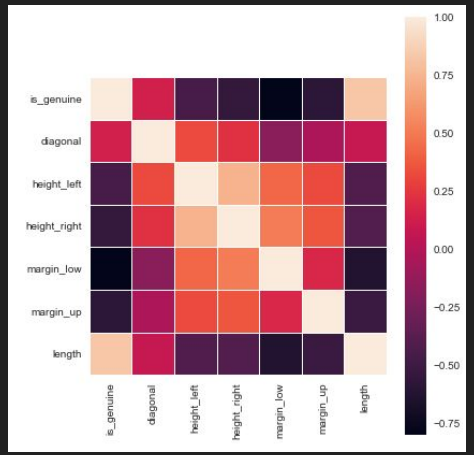
# Notre jeu de données

Le gouvernement nous a donné accès à un ensemble de données qui contient les caractéristiques géométriques des billets de banque. Nous disposons à la fois de données sur les vrais billets de banque et sur les faux billets.

| | is_genuine | diagonal | height_left | height_right | margin_low | margin_up | length |
|---|---|---|---|---|---|---|---|
| **142** | 0 | 171.69 | 104.18 | 104.28 | 5.62 | 3.23 | 110.53 |
| **85** | 1 | 172.10 | 103.95 | 103.72 | 4.49 | 3.07 | 113.15 |
| **39** | 1 | 171.13 | 104.28 | 103.14 | 4.16 | 2.92 | 113.00 |
| **154** | 0 | 171.62 | 104.21 | 103.99 | 5.50 | 3.45 | 111.35 |
| **110** | 0 | 172.10 | 104.30 | 104.21 | 4.07 | 3.41 | 111.27 |
| **122** | 0 | 172.29 | 104.72 | 104.86 | 5.71 | 3.16 | 112.15 |
| **151** | 0 | 171.68 | 103.89 | 103.70 | 5.97 | 3.03 | 109.97 |
| **48** | 1 | 171.73 | 103.82 | 103.85 | 3.97 | 3.12 | 112.85 |
| **67** | 1 | 171.79 | 103.74 | 103.48 | 4.60 | 2.80 | 113.35 |
| **149** | 0 | 171.91 | 103.91 | 103.98 | 4.78 | 3.65 | 111.41 |

# Notre jeu de données

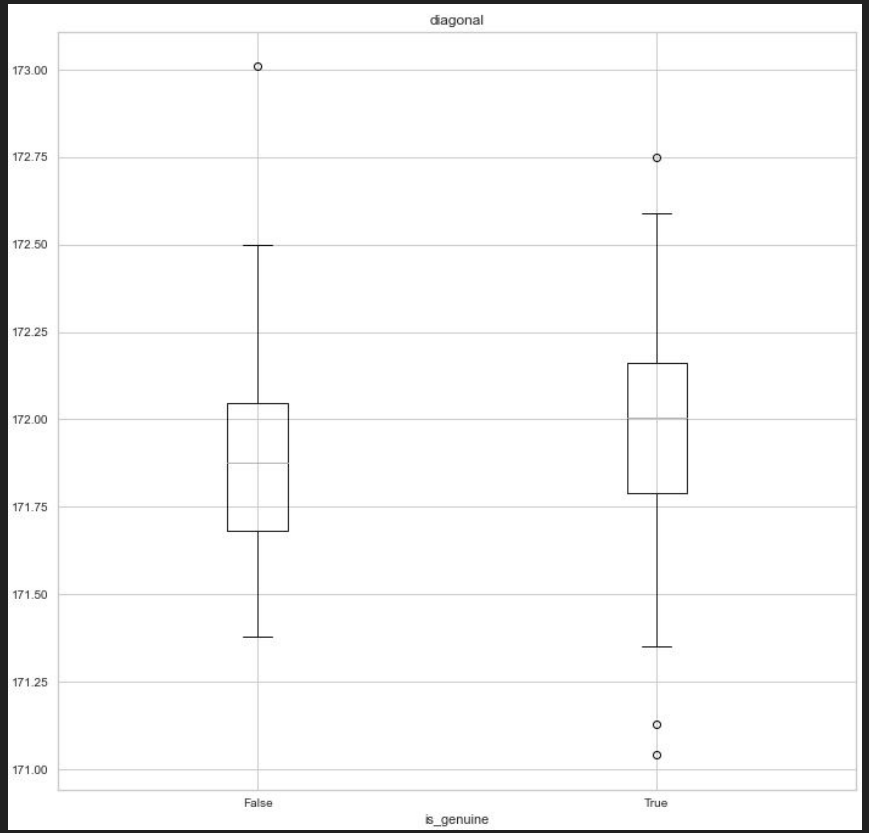


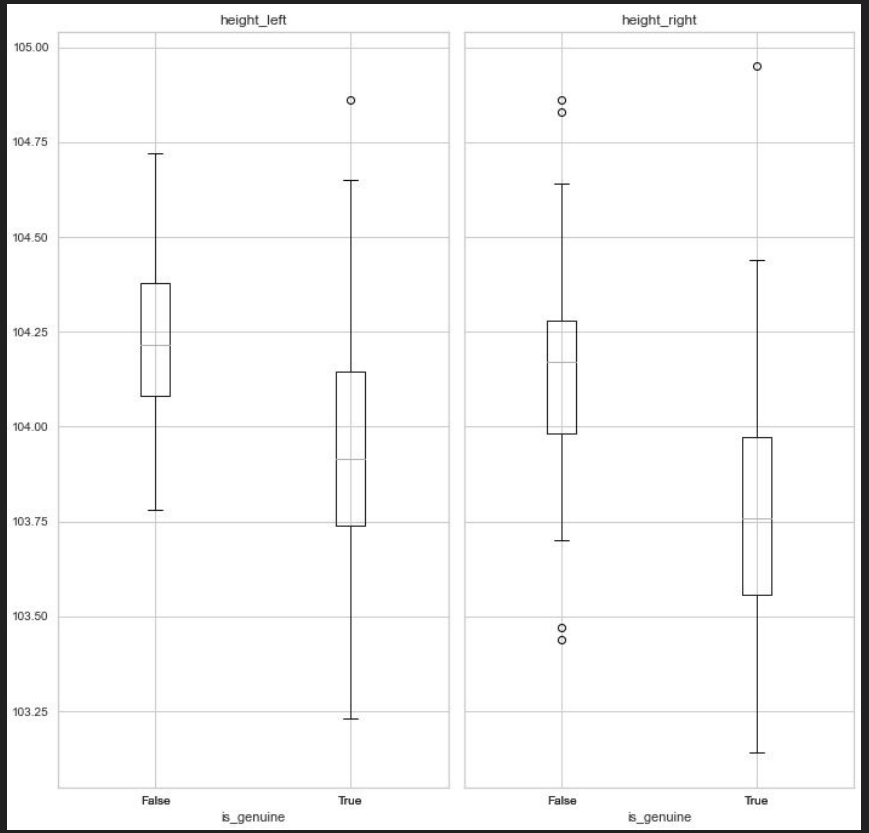




```
1  # check for missing values
2  print((df.isna().sum()/df.shape[0]*100).round(2))
```

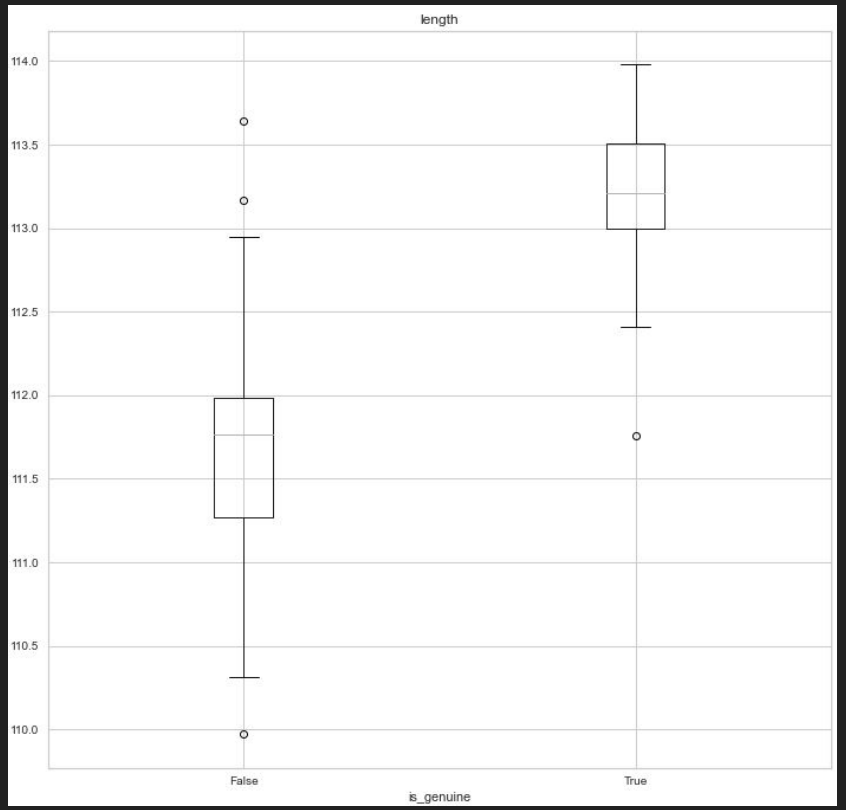
```
is_genuine      0.0
diagonal        0.0
height_left     0.0
height_right    0.0
margin_low      0.0
margin_up       0.0
length          0.0
dtype: float64
```

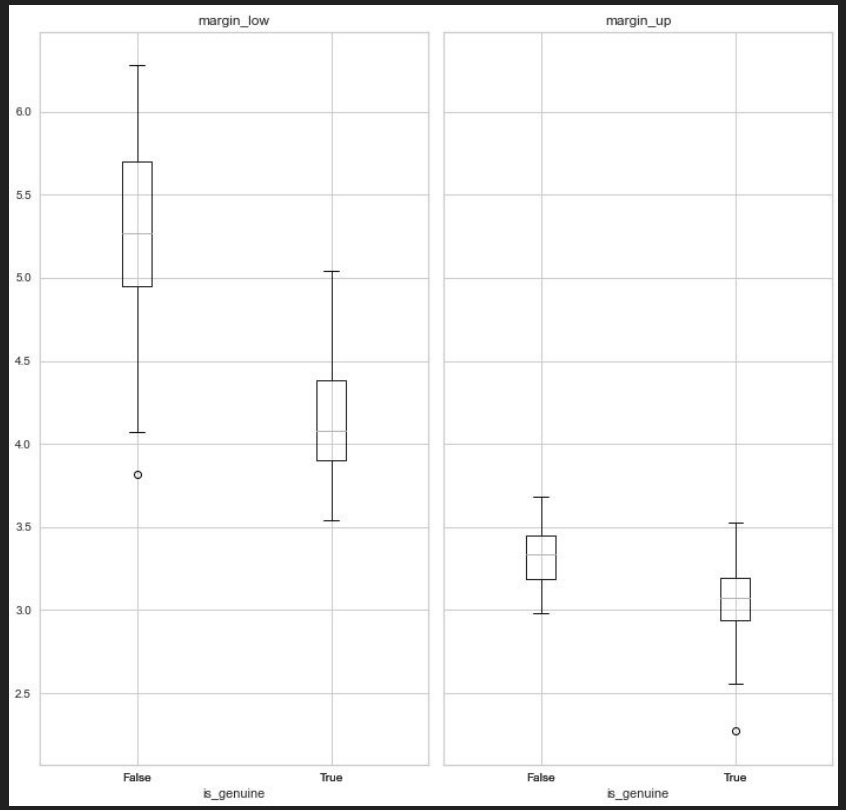
# Notre jeu de données



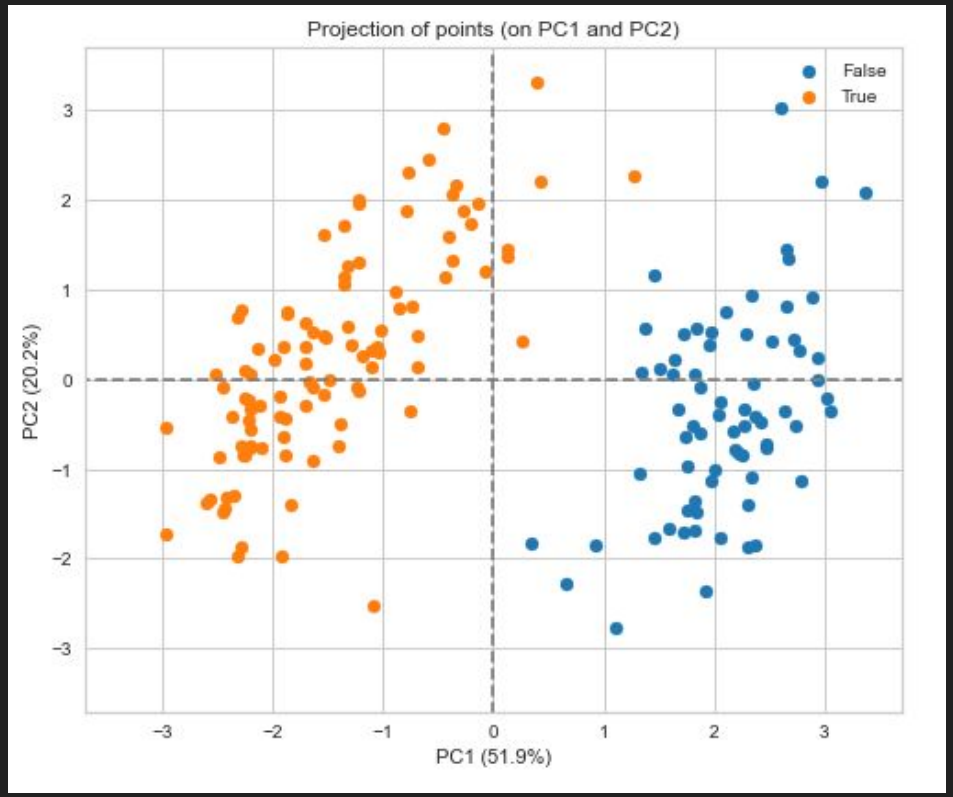



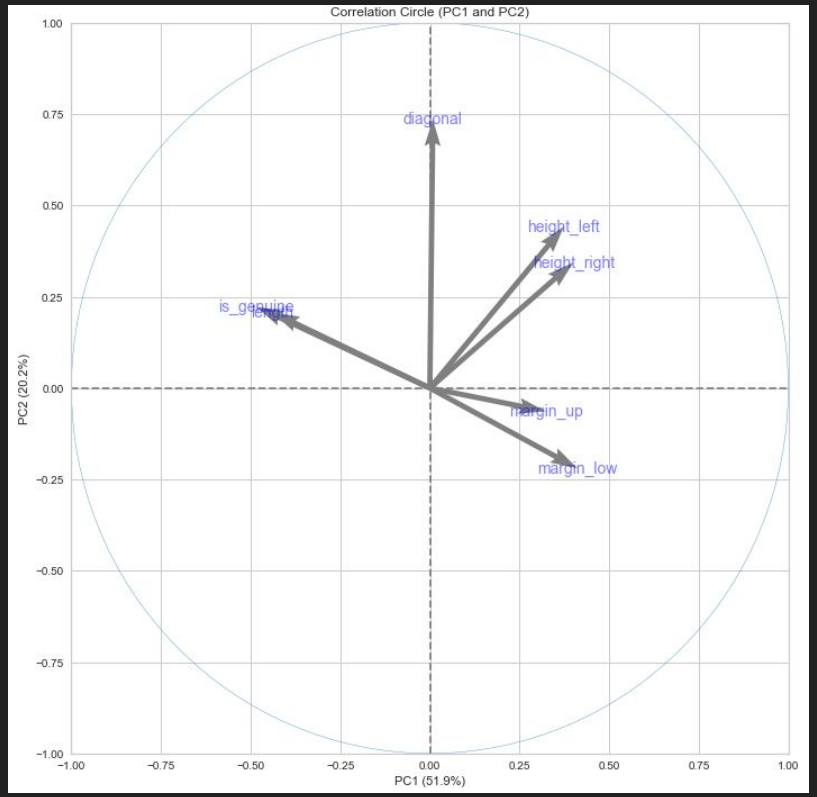
# Notre jeu de données



# ACP


Projection of points (on PC1 and PC2)
False
True
PC2 (20.2%)
PC1 (51.9%)
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3


Correlation Circle (PC1 and PC2)
diagonal
height_left
height_right
is_genuine
margin_up
margin_low
PC2 (20.2%)
PC1 (51.9%)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

# Analyse de la classification

K-means model v1 (default parameters)

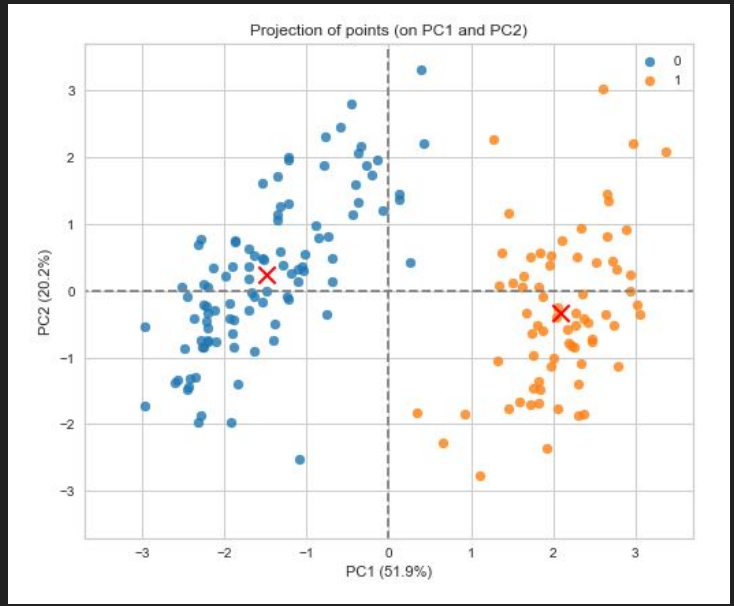
```python
# Kmeans clustering model
kmeans = KMeans(init='random', n_clusters=2, n_init=10, max_iter=300)

# fit data to the model
kmeans.fit_predict(X_scaled)

# trace each datapoint to its corresponding cluster
clusters = kmeans.predict(X_scaled)

# add cluster number to the original data
X_scaled_clustered = pd.DataFrame(X_scaled, columns=X.columns, index=X.index)
X_scaled_clustered['cluster'] = clusters

X_scaled_clustered['cluster'].value_counts()
```

```
0    99
1    71
Name: cluster,
```

K-means model v2 (custom parameters)

```python
# Kmeans clustering model
kmeans = KMeans(init='k-means++', n_clusters=2, n_init=30, max_iter=600)

# fit data to the model
kmeans.fit_predict(X_scaled)

# trace each datapoint to its corresponding cluster
clusters = kmeans.predict(X_scaled)

# add cluster number to the original data
X_scaled_clustered = pd.DataFrame(X_scaled, columns=X.columns, index=X.index)
X_scaled_clustered['cluster'] = clusters

X_scaled_clustered['cluster'].value_counts()
```

```
0    99
1    71
Name: cluster, dtype: int64
```



# Modèle de prédiction

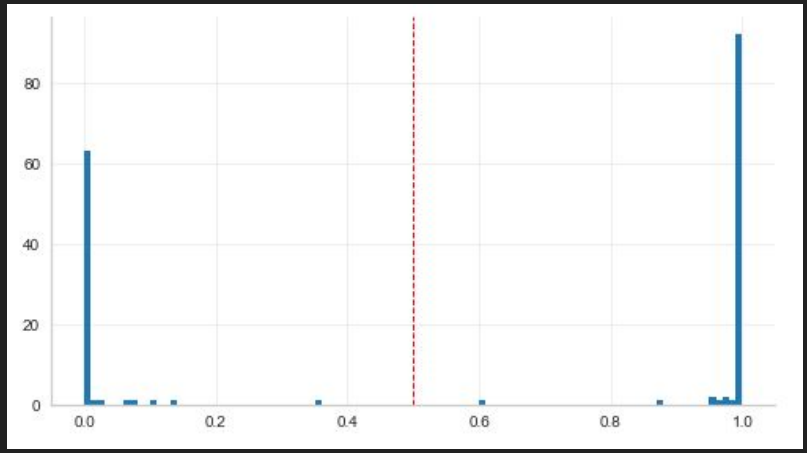
```
# remove height_left from formula due to p-value of 0.965
result = smf.logit(formula = 'is_genuine ~ margin_low + length', data = df).fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.025254
         Iterations 13
```

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | is_genuine | **No. Observations:** | 170 |
| **Model:** | Logit | **Df Residuals:** | 167 |
| **Method:** | MLE | **Df Model:** | 2 |
| **Date:** | Tue, 01 Feb 2022 | **Pseudo R-squ.:** | 0.9627 |
| **Time:** | 20:34:17 | **Log-Likelihood:** | -4.2932 |
| **converged:** | True | **LL-Null:** | -115.17 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 6.999e-49 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | -944.4190 | 377.481 | -2.502 | 0.012 | -1684.269 | -204.569 |
| **margin_low** | -13.3659 | 5.314 | -2.515 | 0.012 | -23.781 | -2.950 |
| **length** | 8.9420 | 3.552 | 2.518 | 0.012 | 1.981 | 15.903 |




```
# accuracy of the model
accuracy = (69 + 99) / 170
print("Accuracy is {:.4f}".format(accuracy))
```

```
Accuracy is 0.9882
```

# Conclusions générales après l'analyse effectuée

```
# la précision du modèle statsmodels
# accuracy of the model
accuracy = (69 + 99) / 170
print("Accuracy is {:.3f}".format(accuracy) + "%")
```

```
Accuracy is 0.988%

Banknote id A_1 is FAKE with probability of 0.5000000049562694 %.
Banknote id A_2 is FAKE with probability of 0.5000000000514444 %.
Banknote id A_3 is FAKE with probability of 0.5000003586870496 %.
Banknote id A_4 is REAL with probability of 0.7310174532491766 %.
Banknote id A_5 is REAL with probability of 0.7310585769698161 %.
```

```
# la précision du modèle sklearn
# evaluate predictions via scores
acc = accuracy_score(y_test, yhat)
print("Accuracy is {:.3f}".format(acc) + "%")
```

```
Accuracy is 0.982%

Banknote id A_1 is FAKE with probability 0.9714141151831927 %.
Banknote id A_2 is FAKE with probability 0.9866265290303662 %.
Banknote id A_3 is FAKE with probability 0.9716235440178718 %.
Banknote id A_4 is REAL with probability 0.8901433089027112 %.
Banknote id A_5 is REAL with probability 0.9904423766337491 %.
```

We have 2 models with high precision, Statsmodels and Sklearn. First one is slightly higher in accuracy, but we can say that Sklearn is more confident predicting a class of a sample.

Comparing to K-means classification results, they are similar. K-means had one false result, while Logistic Regression from both algorithms contain ~0.012-0.018% error rates.