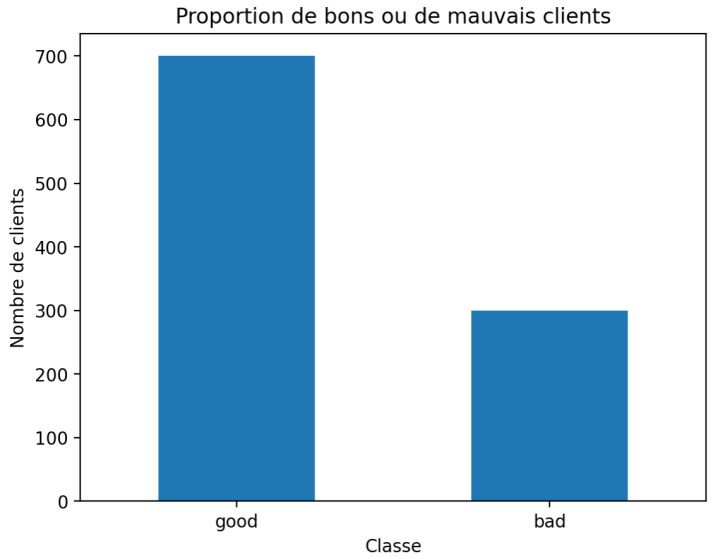
German Credit dataset

Un problème de classification binaire sur des données de clients bancaires

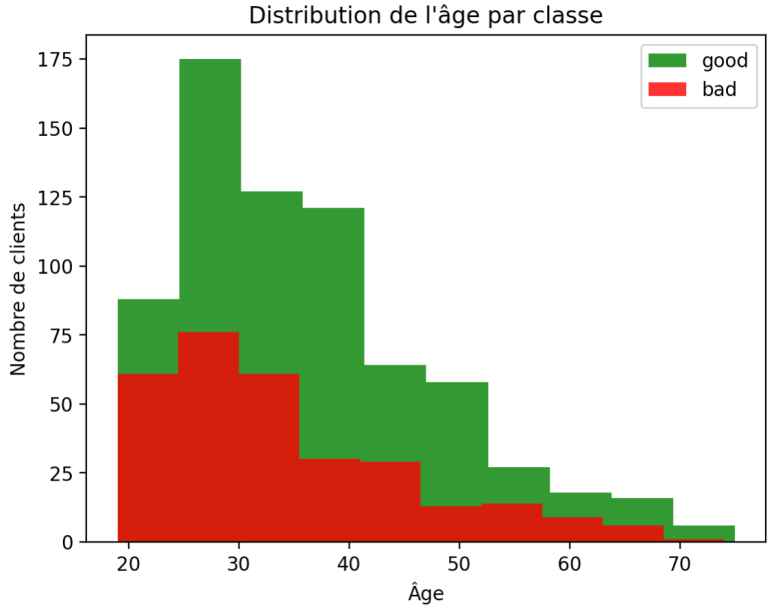
Exploration préliminaire des données

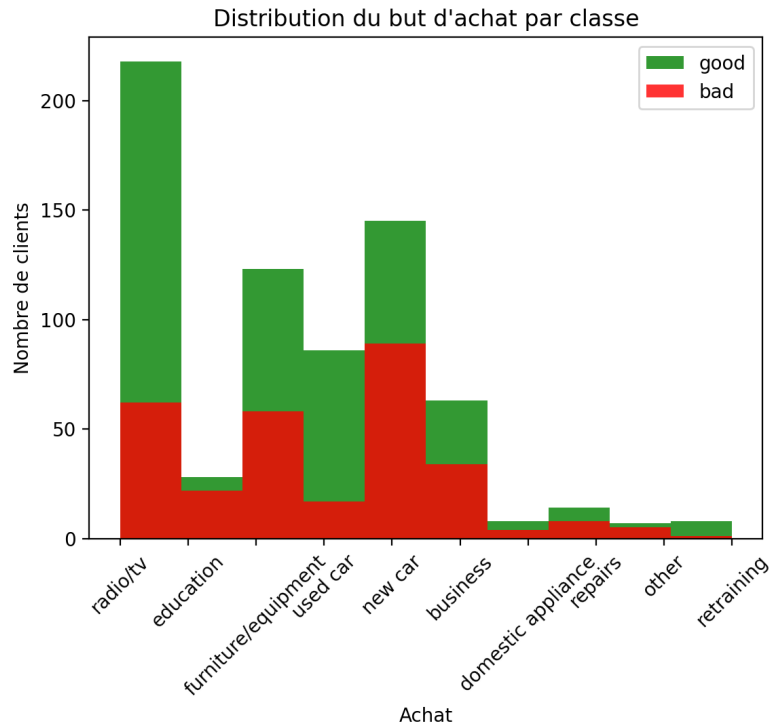
- **Analyse descriptive**

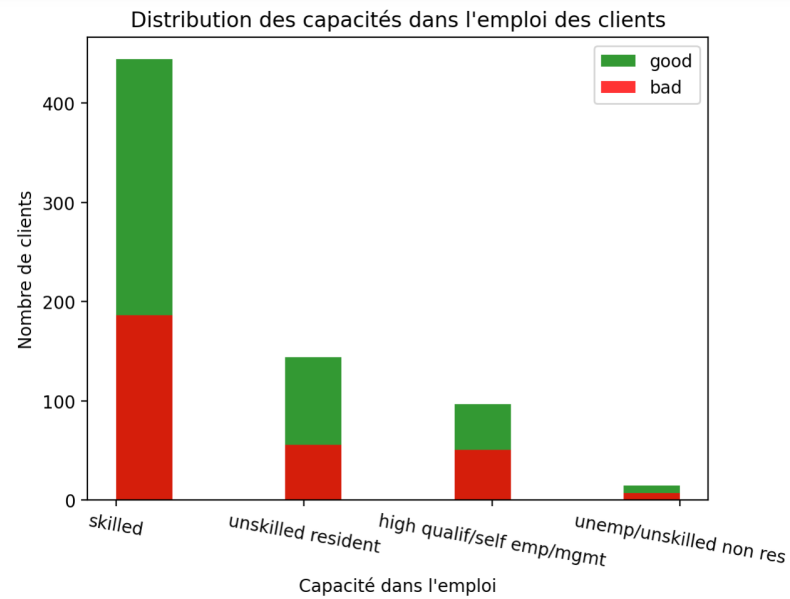
- Objet du crédit (voiture, télévision,...)
- Montant du crédit
- Emploi actuel, en nombre d'années.
- Statut personnel (marié, célibataire,...)
- Résidence actuelle depuis X ans
- Age en années
- Logement (location, propriété...)
- ...



id	checking_status	duration	credit_history	purpose	credit_amount	savings_status	employment	installment_commitment	personal_status	other_parties
1	<0	6	critical/other existing credit	radio/tv	1169	no known savings	>=7	4	male single	none
2	0<=X<200	48	existing paid	radio/tv	5951	<100	1<=X<4	2	female div/dep/mar	none
3	no checking	12	critical/other existing credit	education	2096	<100	4<=X<7	2	male single	none

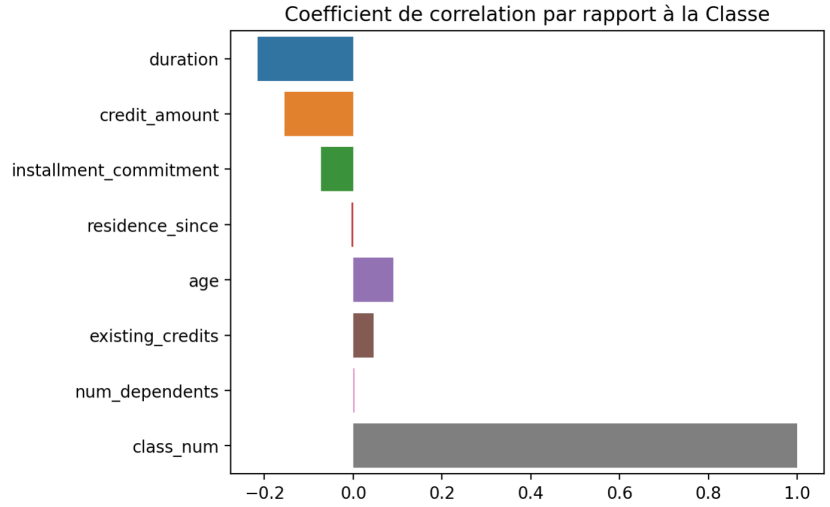

Distribution de l'âge par classe
good
bad
Nombre de clients
Âge
0
25
50
75
100
125
150
175
20
30
40
50
60
70

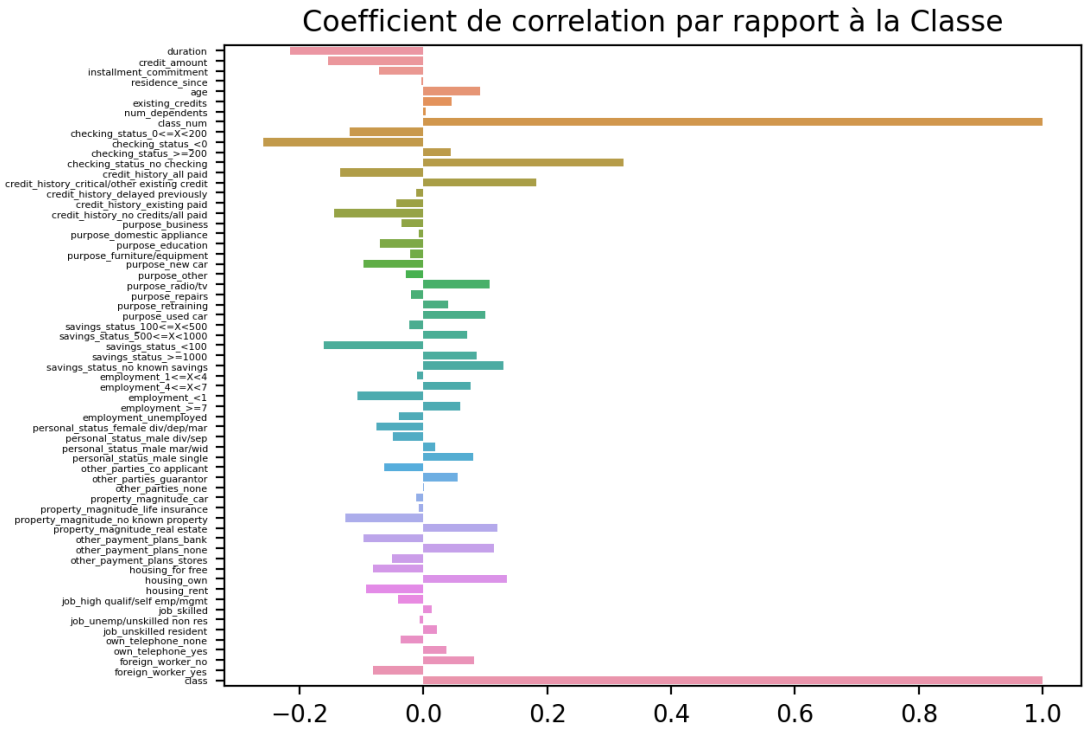

Distribution du but d'achat par classe
good
bad
Nombre de clients
Achat
0
50
100
150
200
radio/tv
education
furniture/equipment
used car
new car
business
domestic appliance
repairs
other
retraining


Distribution des capacités dans l'emploi des clients
good
bad
Nombre de clients
Capacité dans l'emploi
0
100
200
300
400
skilled
unskilled resident
high qualif/self emp/mgmt
unemp/unskilled non res

Nettoyage des données

- Suppression les NaN
- Encodage One-Hot
- Corrélation des features



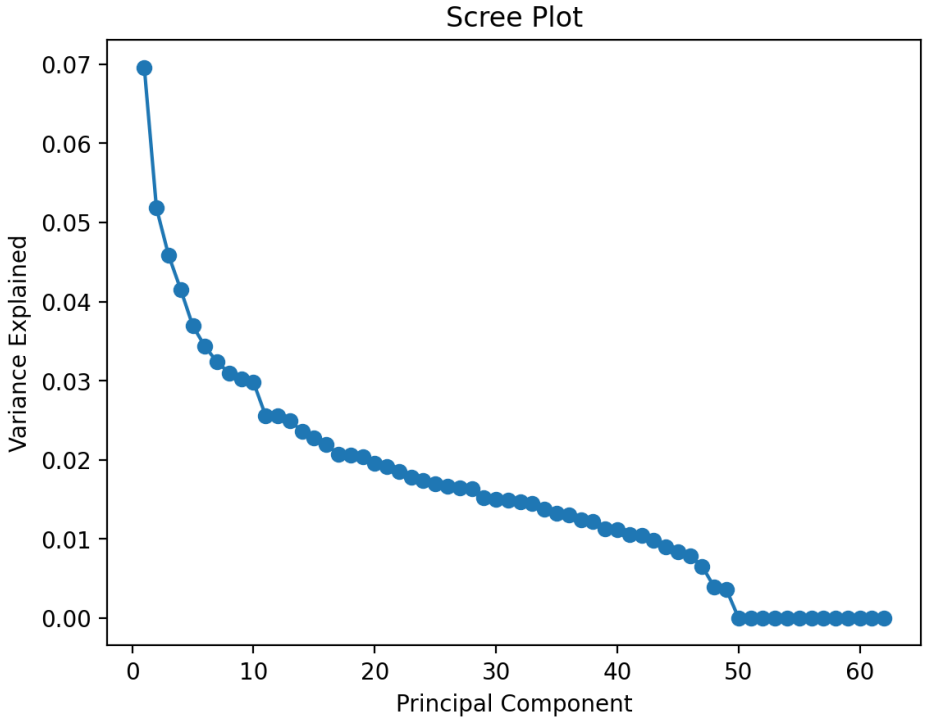


PCA

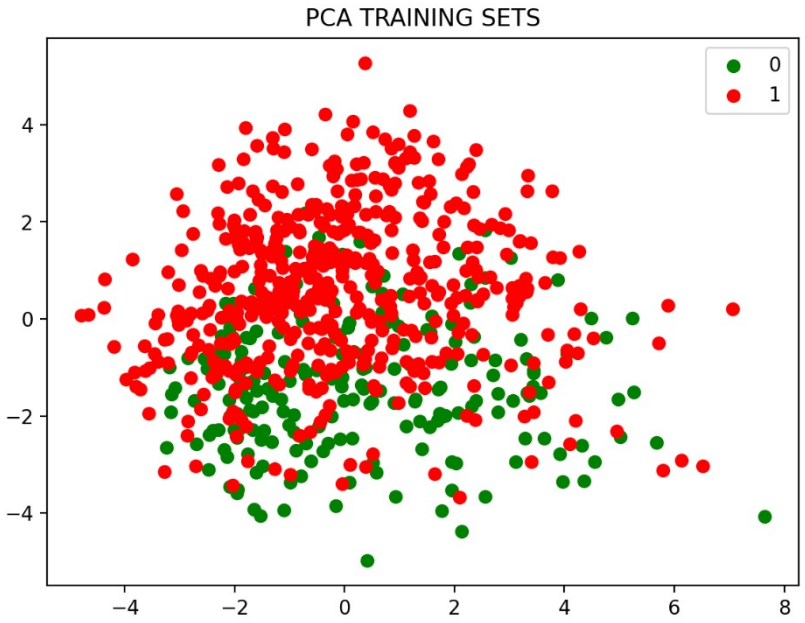
- **Séparation des données en train-test-validation**
- **Standardisation**

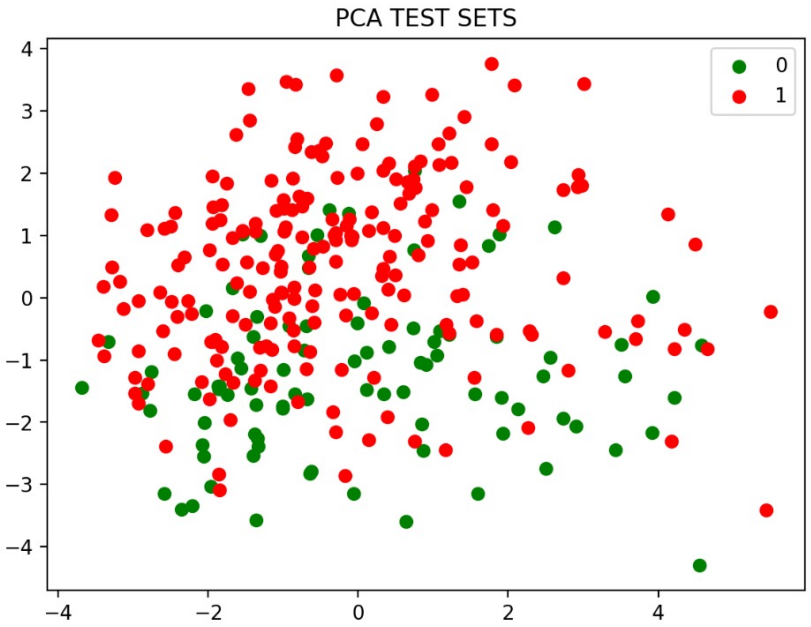
```
1 cumulative_variance_ratio = np.cumsum(pca.explained_variance_ratio_)
2 n_components = np.argmax(cumulative_variance_ratio >= 0.52) + 1
3 print(f"Number of components to explain 52% variance: {n_components}")
```

```
Number of components to explain 52% variance: 15
```



Entrainement PCA avec SVM





Accuracy: 0.83

	precision	recall	f1-score	support
0	0.75	0.66	0.70	91
1	0.86	0.90	0.88	209
accuracy			0.83	300
macro avg	0.80	0.78	0.79	300
weighted avg	0.83	0.83	0.83	300

Unique Values: [0 1]
Frequency Values: [209 491]

Cross-Validation

[0.78095238 0.76666667 0.79761905 0.76666667 0.89761905 0.81190476
 0.7 0.80952381 0.82380952 0.81666667]

balanced_accuracy: 0.79714286, with std dev: 0.05

Modèles de classifications

- **LogisticRegression**
- **RandomForestClassifier**
- **DecisionTreeClassifier**

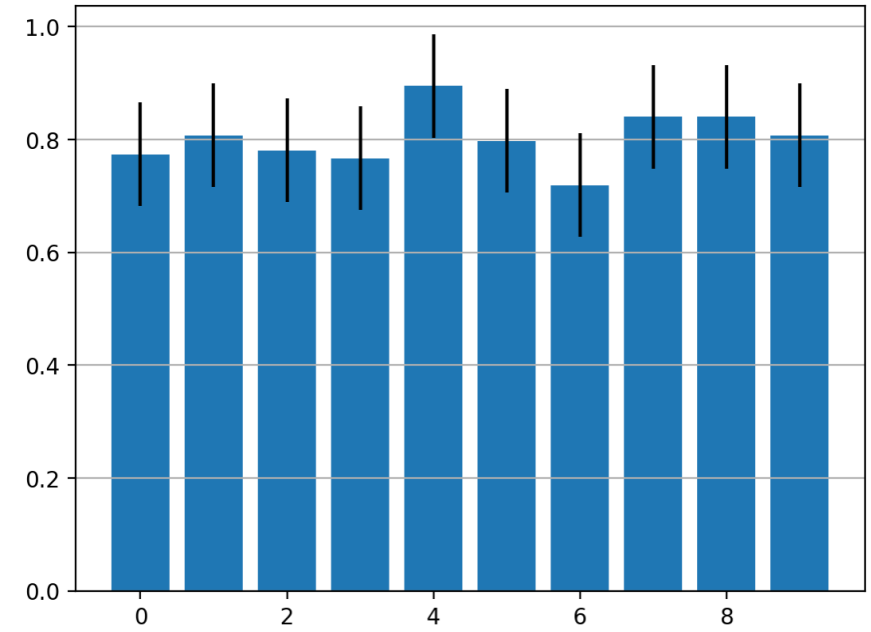
LogisticRegression

```
1  y_pred = model.predict(X_test_pca)  # predictions
2  score = metrics.balanced_accuracy_score(Y_test, y_pred)  # scoring
3  print(f"Balanced accuracy score: {score:.3g}")
```

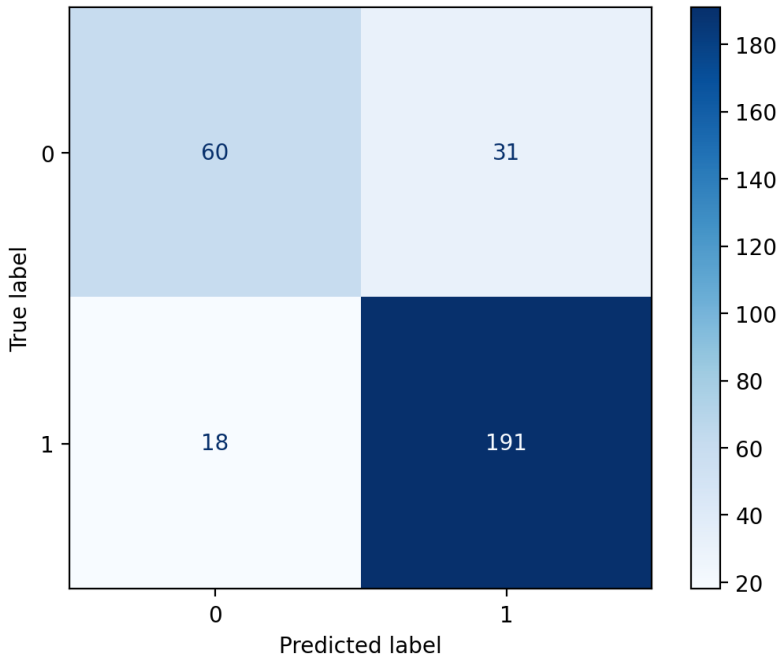
```
Balanced accuracy score: 0.787
```

Cross-Validation

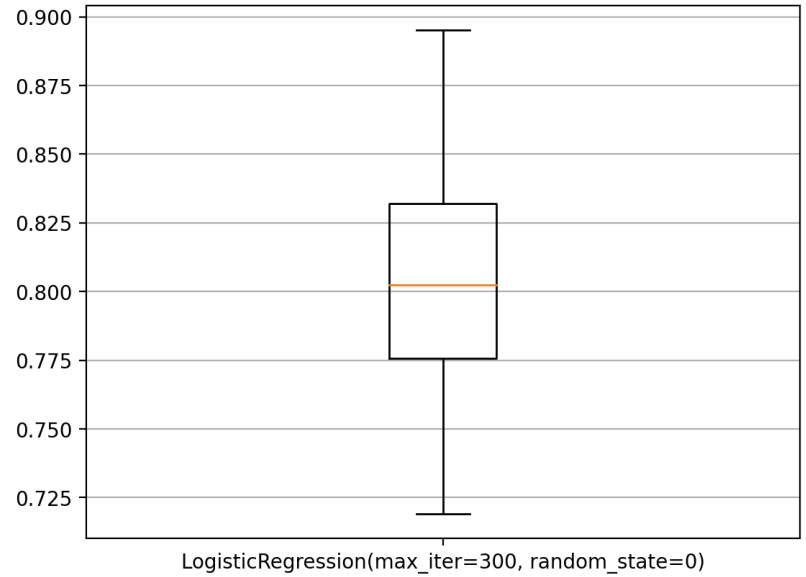
```
[0.77380952 0.80714286 0.78095238 0.76666667 0.8952381  0.79761905
 0.71904762 0.84047619 0.84047619 0.80714286]

balanced_accuracy: 0.80285714, with std dev: 0.05
```



Bar-Plot sur les 10 valeurs de la cross-validation



Matrice de Confusion sur les scores obtenues



Box-Plot sur les scores obtenues

Score AUC ROC

```
In [ ]: 1 sklearn.metrics.roc_auc_score(Y_test, y_pred)

Out [ ]: 0.7866081287133919
```

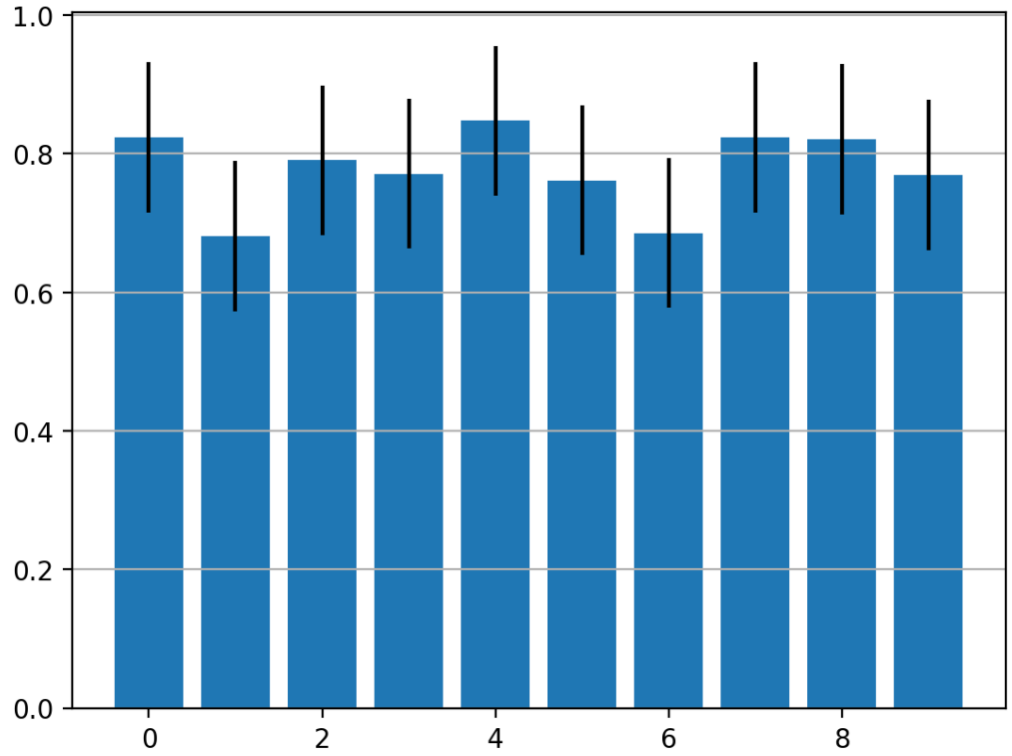
RandomForestClassifier

```
1  y_pred = model.predict(X_test_pca)  # predictions
2  score = metrics.balanced_accuracy_score(Y_test, y_pred)  # scoring
3  print(f"Balanced accuracy score: {score:.3g}")
```

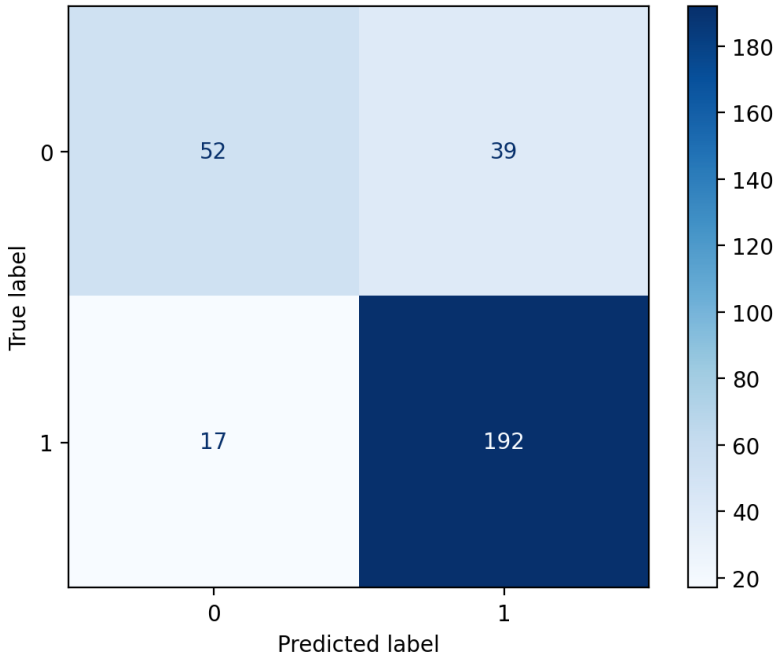
```
Balanced accuracy score: 0.745
```

Cross-Validation

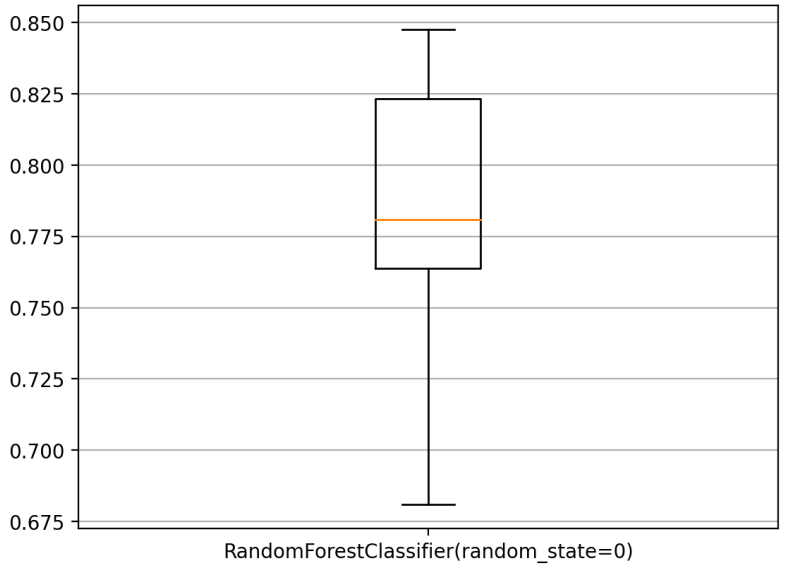
```
[0.82380952 0.68095238 0.79047619 0.77142857 0.84761905 0.76190476
 0.68571429 0.82380952 0.82142857 0.76904762]

balanced_accuracy: 0.77761905, with std dev: 0.05
```



Bar-Plot sur les 10 valeurs de la cross-validation



Matrice de Confusion sur les scores obtenues



Box-Plot sur les scores obtenues

Score AUC ROC

```
1 sklearn.metrics.roc_auc_score(Y_test, y_pred)
```

```
0.7450444292549556
```

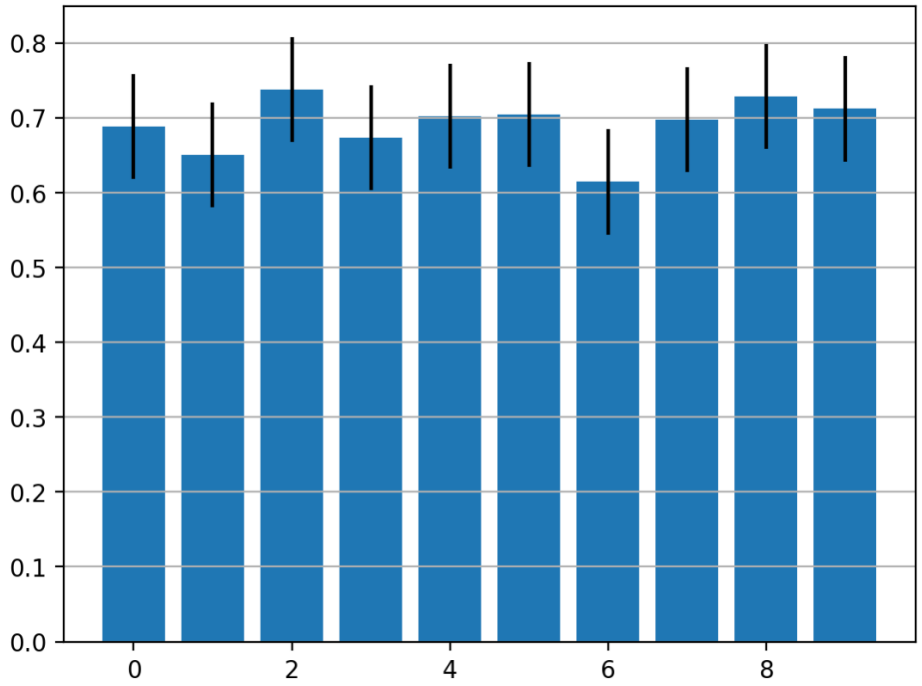
DecisionTreeClassifier

```
1 y_pred = model.predict(X_test_pca)  # predictions
2 score = metrics.balanced_accuracy_score(Y_test, y_pred)  # scoring
3 print(f"Balanced accuracy score: {score:.3g}")
```

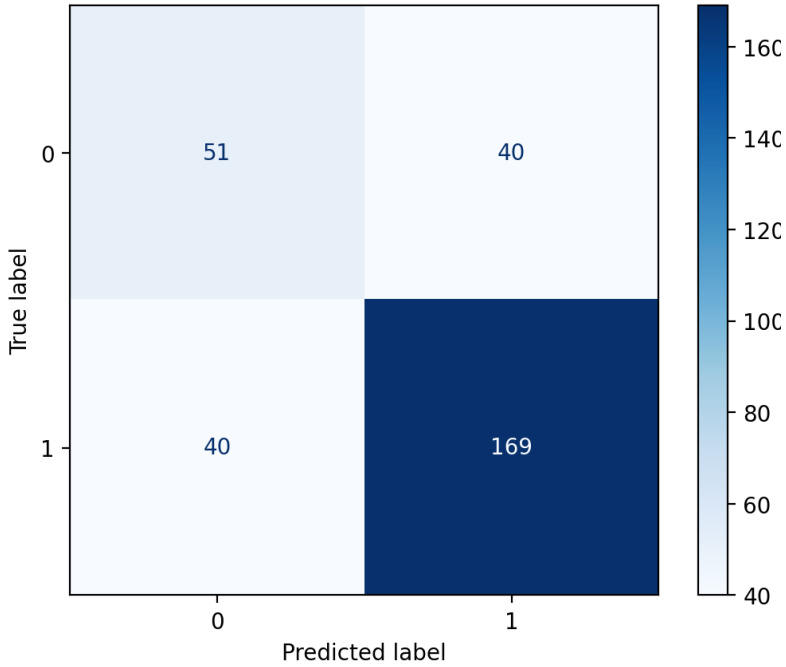
```
Balanced accuracy score: 0.685
```

Cross-Validation

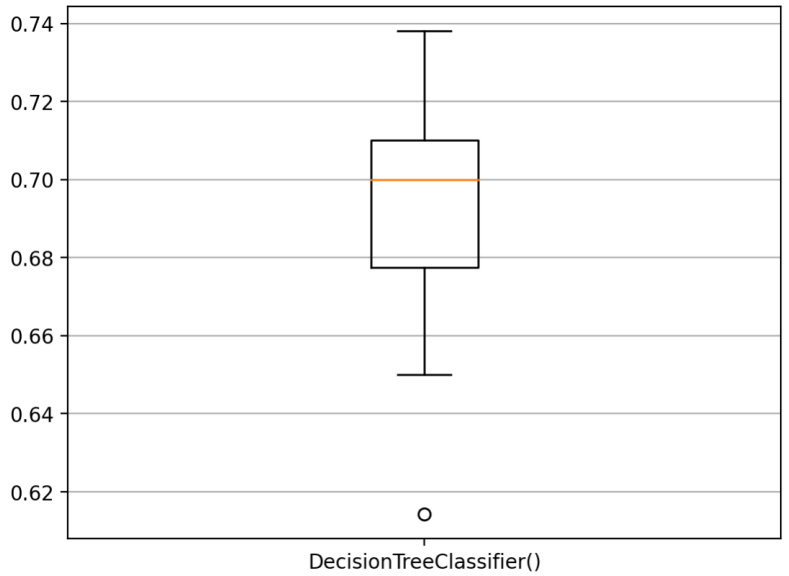
```
[0.68809524 0.65       0.73809524 0.67380952 0.70238095 0.7047619
 0.61428571 0.69761905 0.72857143 0.71190476]

balanced_accuracy: 0.69095238, with std dev: 0.04
```



Bar-Plot sur les 10 valeurs de la cross-validation



Matrice de Confusion sur les scores obtenues

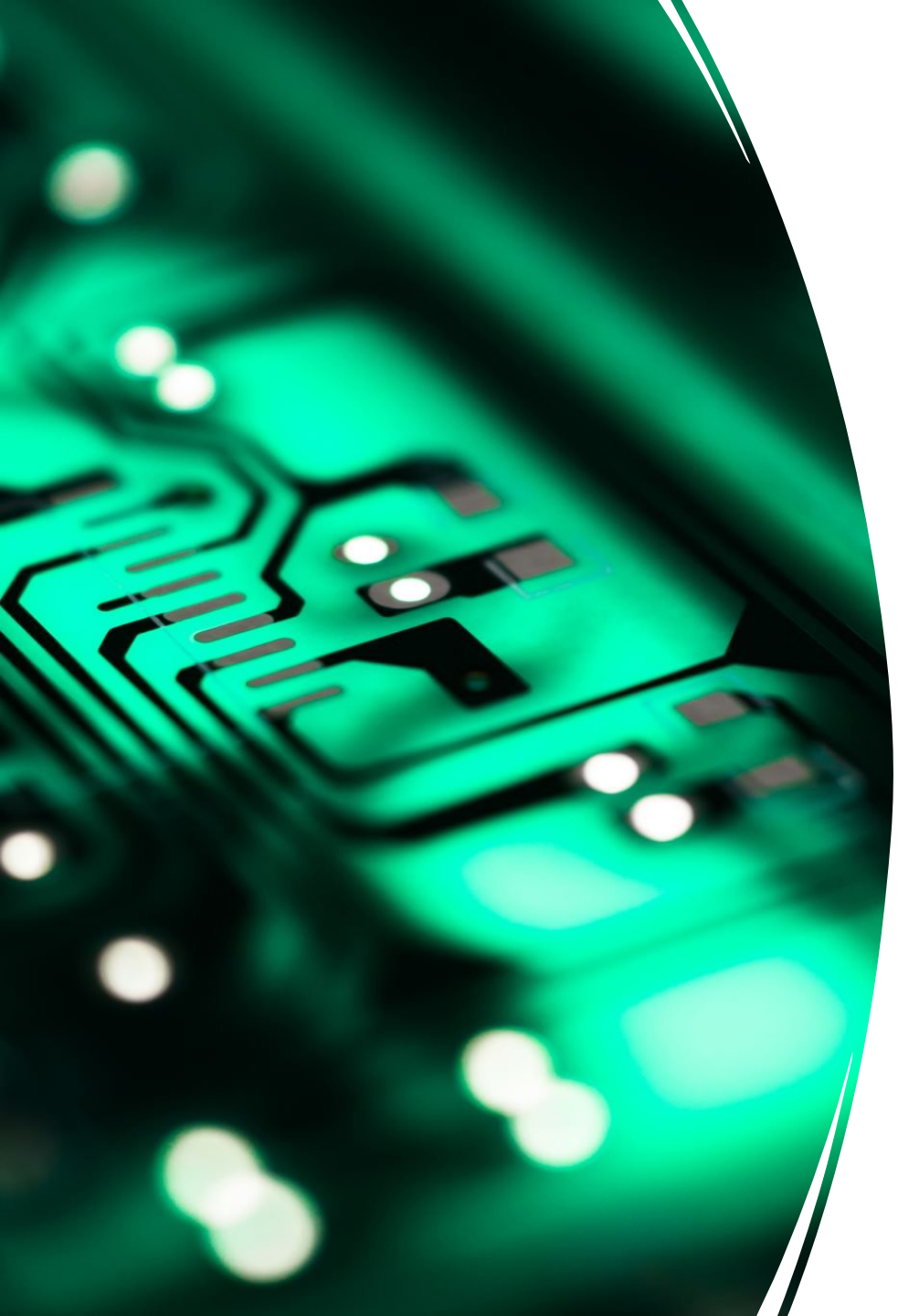


Box-Plot sur les scores obtenues

Score AUC ROC

```
1 sklearn.metrics.roc_auc_score(Y_test, y_pred)
```

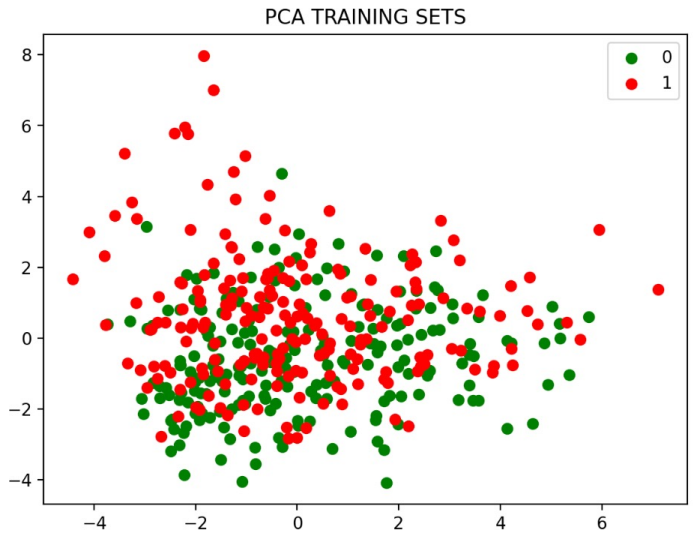
```
0.684526000315474
```

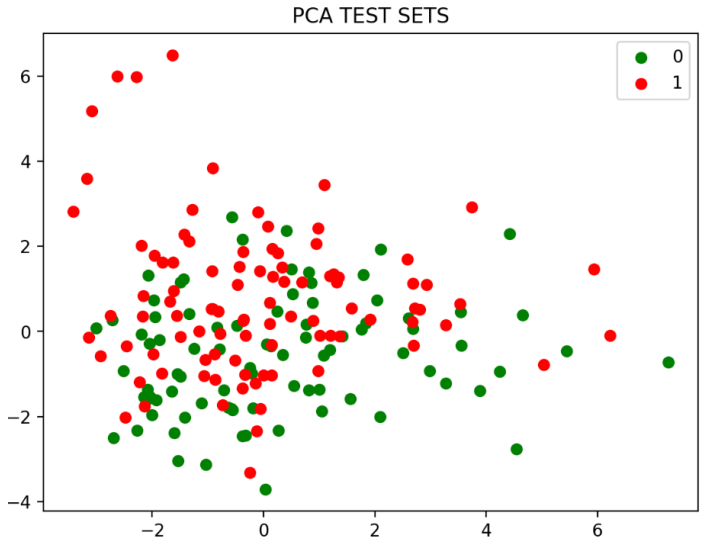


Sur/Sous-échantillonage:

- Under-Sampling
- Over-Sampling

Under-Sampling





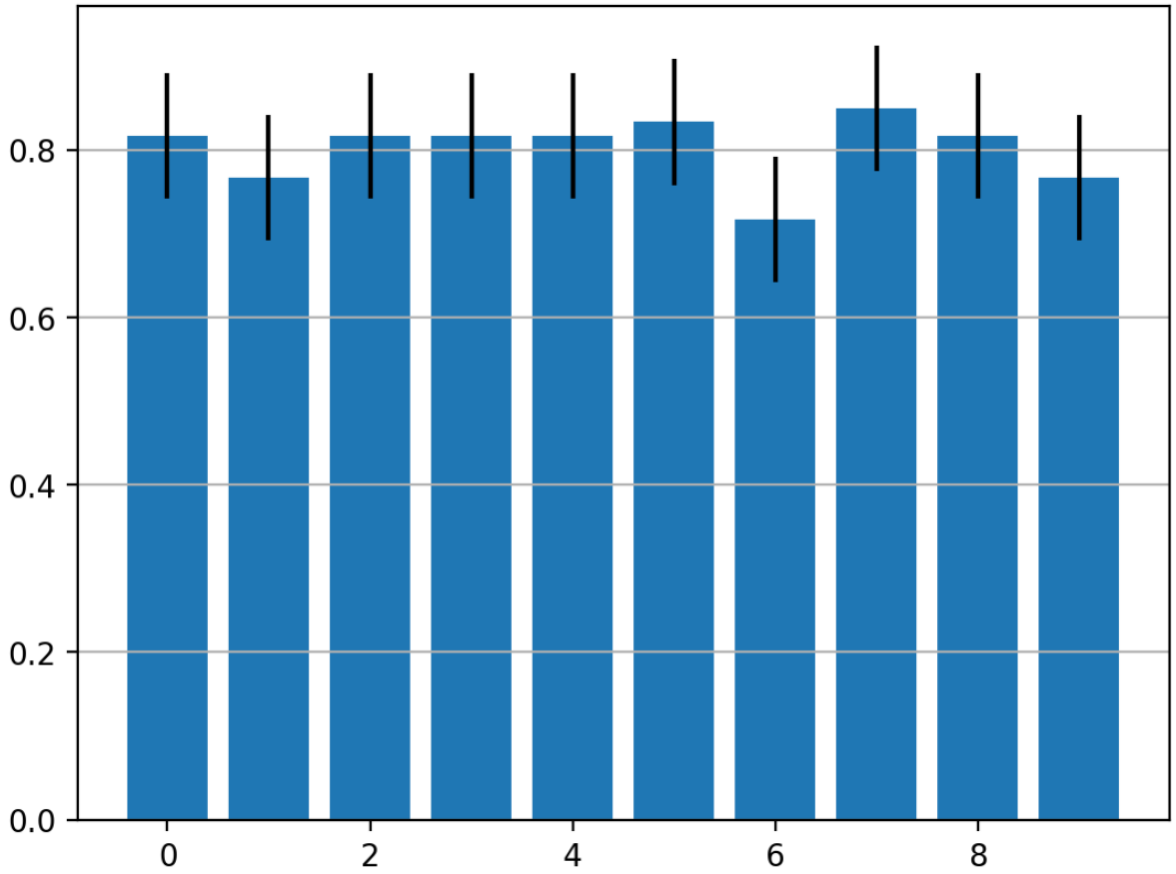

```
under-sampling
Unique Values: [0 1]
Frequency Values: [215 205]
```

Entrainement du modèle

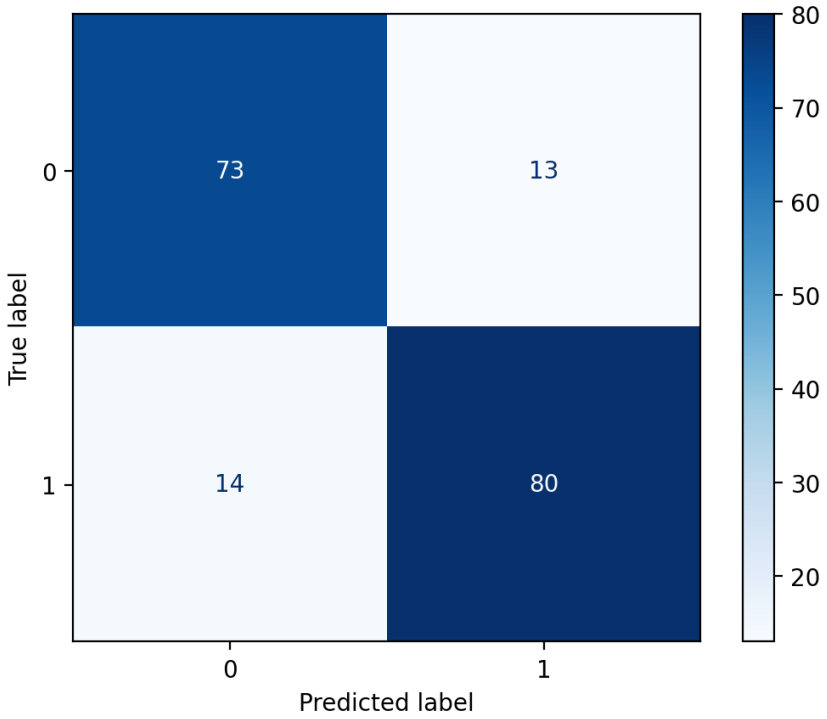
```
Balanced accuracy score: 0.85

              precision    recall  f1-score   support

           0       0.84      0.85      0.84        86
           1       0.86      0.85      0.86        94

    accuracy                           0.85       180
   macro avg       0.85      0.85      0.85       180
weighted avg       0.85      0.85      0.85       180
```

Cross-Validation

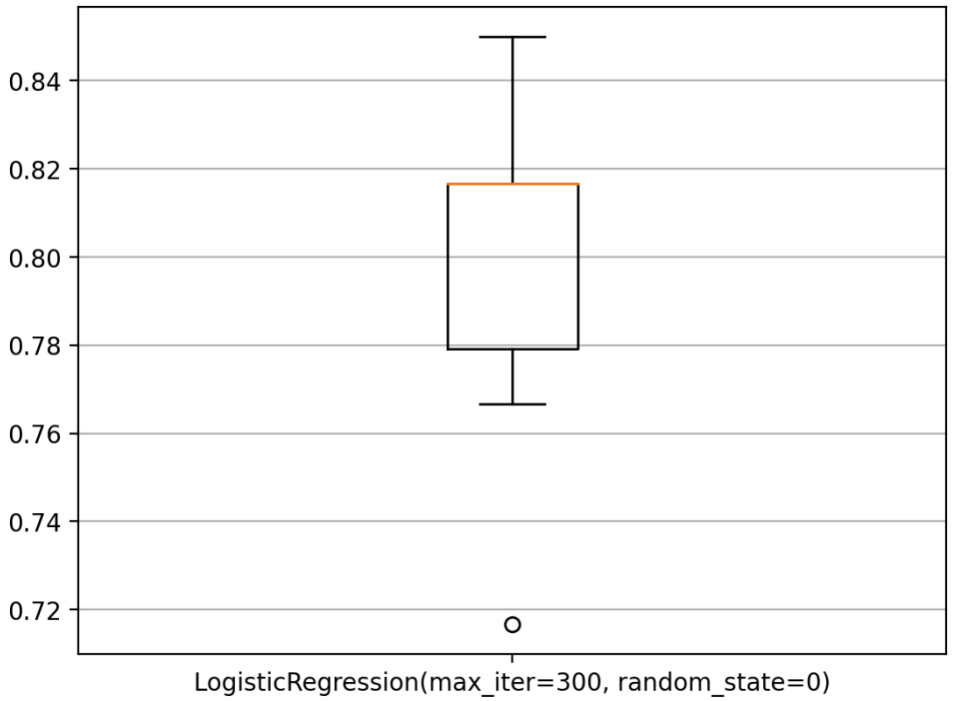
```
[0.81666667 0.76666667 0.81666667 0.81666667 0.81666667 0.83333333
 0.71666667 0.85       0.81666667 0.76666667]

balanced_accuracy: 0.80166667, with std dev: 0.04
```



Bar-Plot sur les 10 valeurs de la cross-validation



Matrice de Confusion sur les scores obtenues



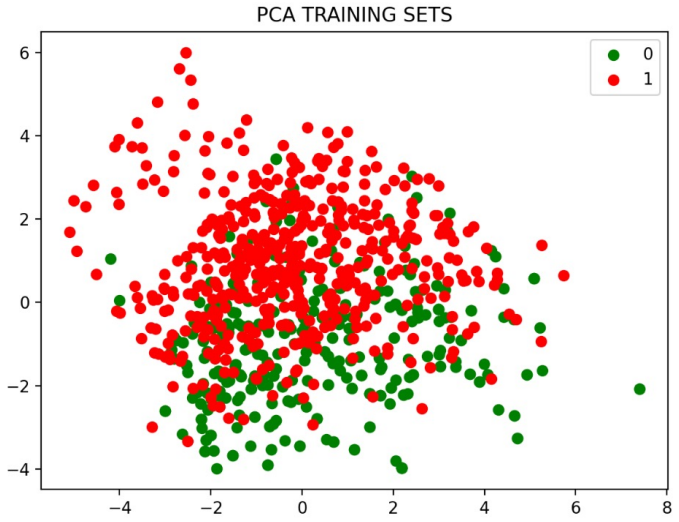
Box-Plot sur les scores obtenues

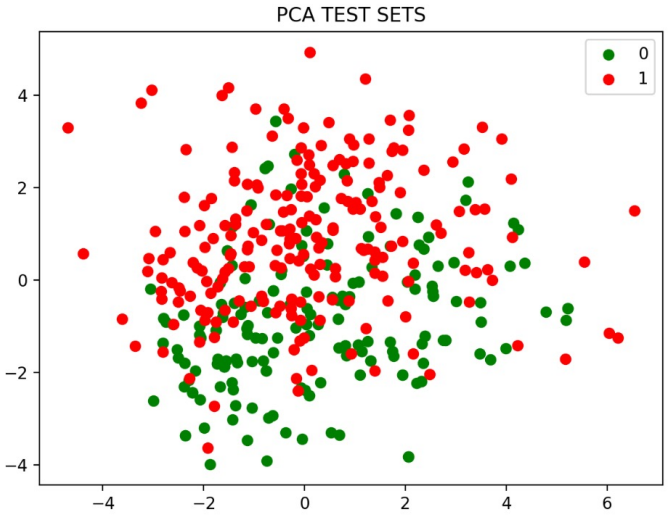
Score AUC ROC

```
1 sklearn.metrics.roc_auc_score(y_test_rus, y_pred)
```

```
0.8499505195447796
```

Over-Sampling






```
over-sampling
Unique Values: [0 1]
Frequency Values: [492 488]
```

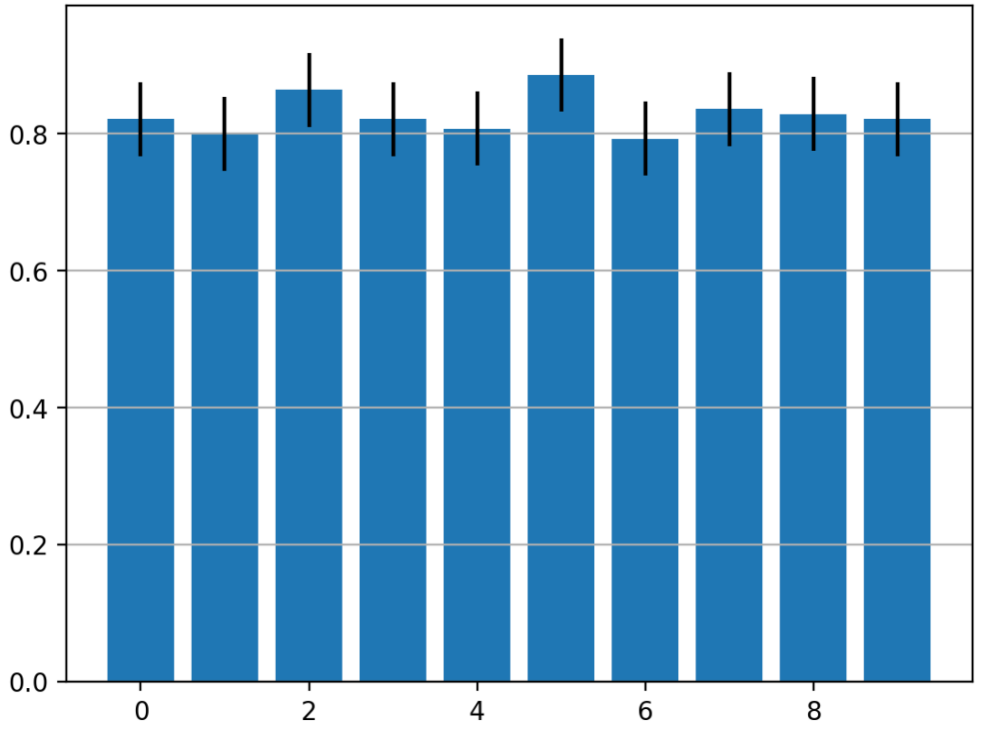
Entrainement du modèle

```
Balanced accuracy score: 0.836
```

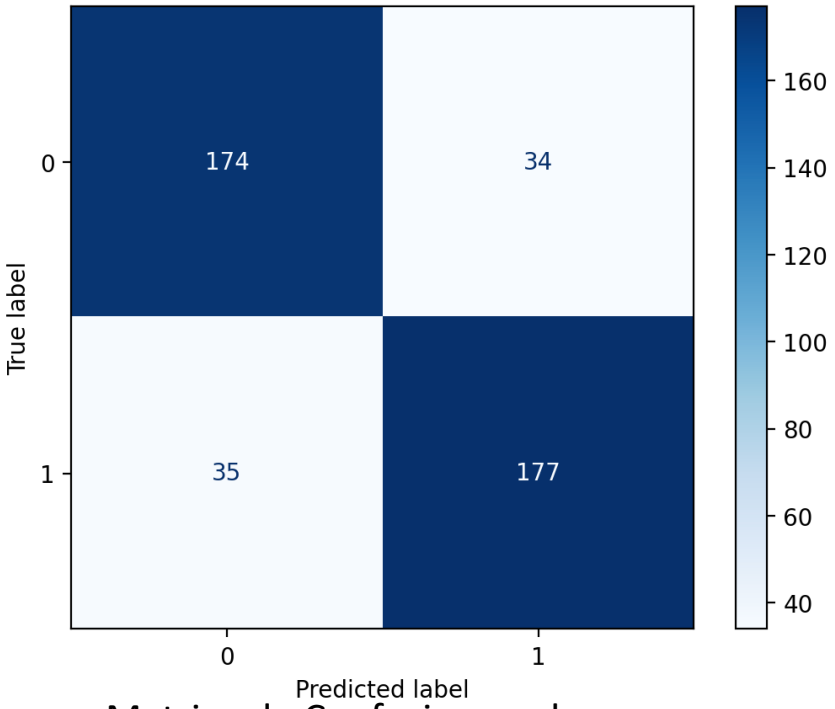
	precision	recall	f1-score	support
0	0.83	0.84	0.83	208
1	0.84	0.83	0.84	212
accuracy			0.84	420
macro avg	0.84	0.84	0.84	420
weighted avg	0.84	0.84	0.84	420

Cross-Validation

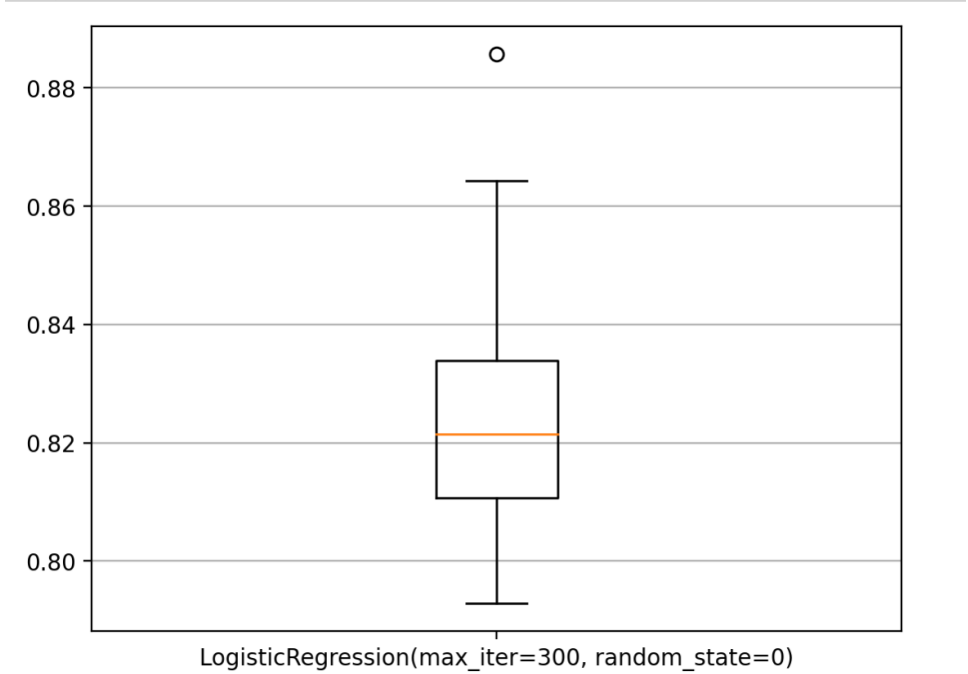
```
[0.82142857 0.8        0.86428571 0.82142857 0.80714286 0.88571429
 0.79285714 0.83571429 0.82857143 0.82142857]

balanced_accuracy: 0.82785714, with std dev: 0.03
```



Bar-Plot sur les 10 valeurs de la cross-validation



Matrice de Confusion sur les scores obtenues



Box-Plot sur les scores obtenues

Score AUC ROC

```
1 sklearn.metrics.roc_auc_score(y_test_ros, y_pred)
```

```
0.8357220609579101
```

Conclusion