Ai-je le droit à un prêt bancaire ?

Loan Prediction Dataset

Qui suis-je ?

Asma

- Consultante en datascience
- Analyse les données clients et aide les entreprises à prendre des décisions stratégiques ou opérationnelles

Contexte

Le crédit agricole cherche à automatiser la procédure de d'attribution de prêt sur la base des informations fournies par les clients lors du remplissage d'un formulaire de demande en ligne.

Le but est de développer un modèle de ML qui puissent aider l'entreprise à prédire l'approbation du prêt en accélérant le processus de prise de décision pour déterminer si un demandeur est éligible ou non à un prêt.
Et de développer un API pour que le client puisse entrer ses informations.

GitHub

Jira

Jupiter Notebook

Visual Studio Code

Backlog

Tableau Sprint 1 3 oct. – 10 oct. (6 tickets) — 0 0 0 — Terminer le sprint

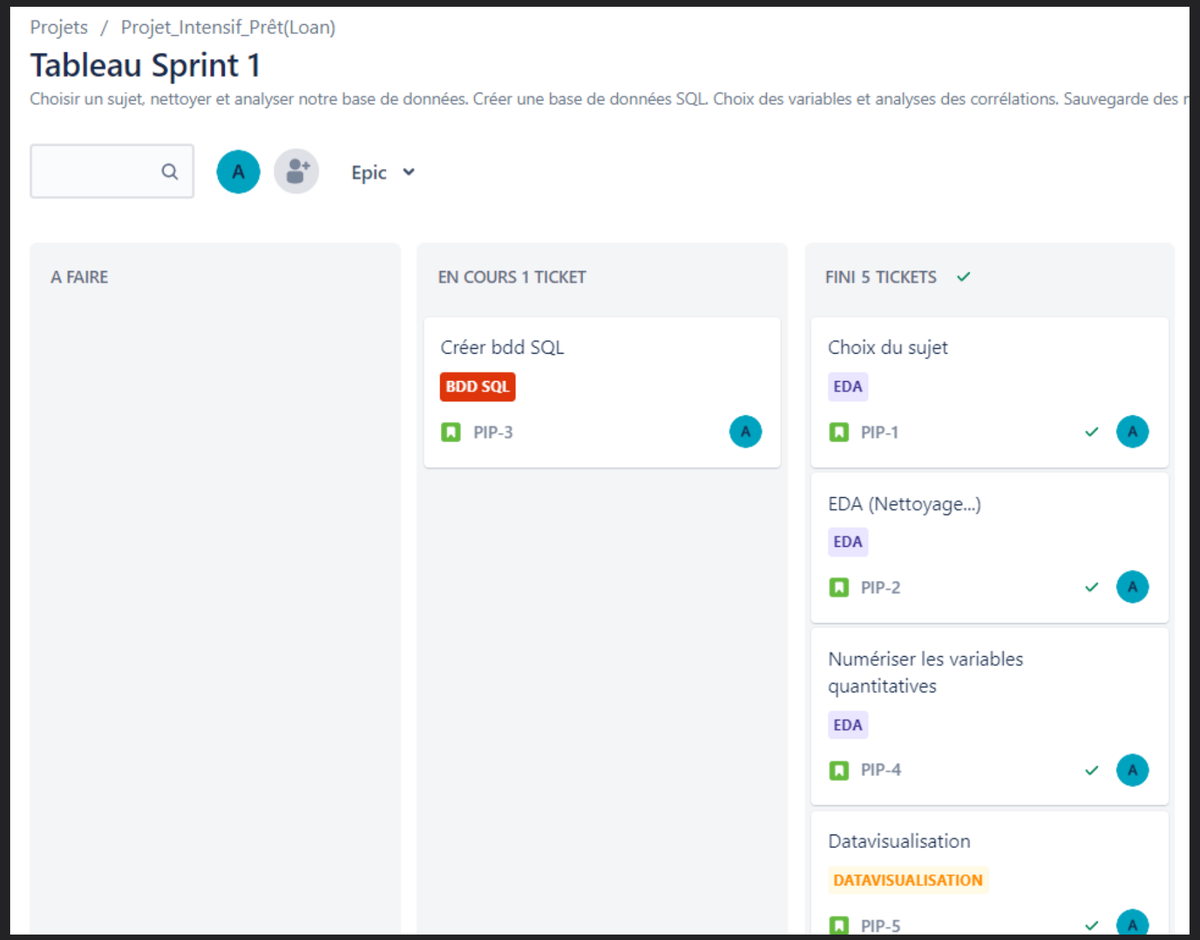
Choisir un sujet, nettoyer et analyser notre base de données. Créer une base de données SQL. Choix des variables et analyses des corrélations. Sauvegarde des nouveaux dataframes.

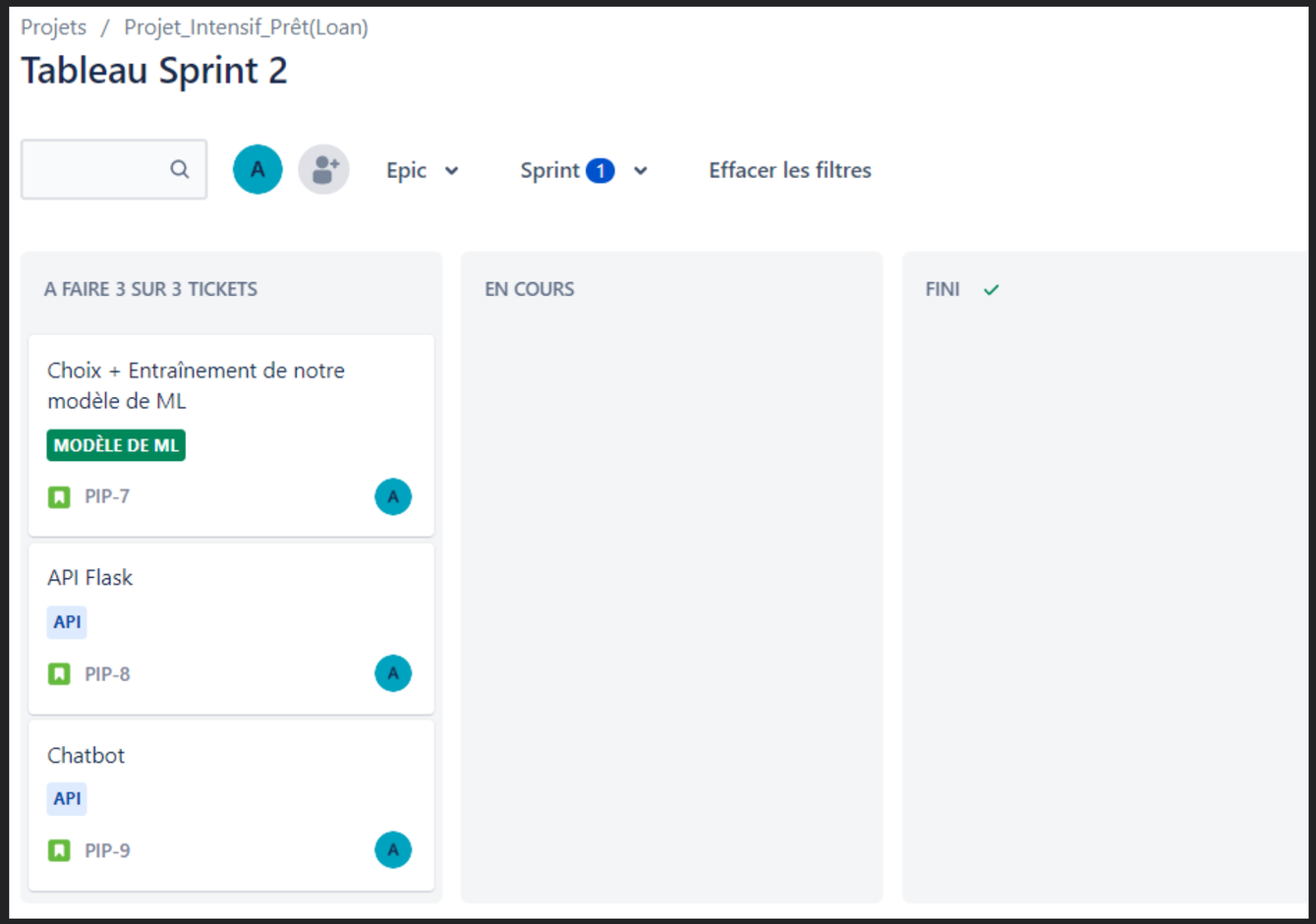
~~PIP-1~~ Choix du sujet	EDA	TERMINÉ(E)
~~PIP-2~~ EDA (Nettoyage...)	EDA	TERMINÉ(E)
PIP-3 Créer bdd SQL	BDD SQL	EN COURS
~~PIP-4~~ Numériser les variables quantitatives	EDA	TERMINÉ(E)
~~PIP-5~~ Datavisualisation	DATAVISUALISATION	TERMINÉ(E)
~~PIP-6~~ Analyse des corrélations	EDA	TERMINÉ(E)

Tableau Sprint 2 10 oct. – 17 oct. (3 tickets) — 0 0 0 — Démarrer un sprint

PIP-7 Choix + Entraînement de notre modèle de ML	MODÈLE DE ML	À FAIRE
PIP-8 API Flask	API	À FAIRE
PIP-9 Chatbot	API	À FAIRE

Tableau des Sprints





Présentation du dataset

Colonne	Description
Loan_ID	Identifiant unique du demandeur
Gender	Sexe du demandeur Masculin / Féminin
Married	Le demandeur est marié ? Oui / Non
Dependents	Nombre de personnes à charge [0, 1, 2, 3+]
Education	Diplômé ou non
Self_Employed	Le demandeur est un travailleur indépendant ? Oui / Non
ApplicantIncome	Salaire/revenu mensuel du demandeur
CoapplicantIncome	Revenu mensuel du codemandeur
LoanAmount	Montant du prêt (en milliers)
Loan_Amount_Term	La durée de remboursement du prêt (en jours)
Credit_History	Antécédents de crédit du remboursement individuel de ses dettes
Property_Area	Emplacement de la propriété, c'est-à-dire rurale/urbaine/semi-urbaine
Loan_Status	Statut du prêt approuvé ou non, O : Oui, N : Non

Pour ce problème, nous avons 2 fichiers CSV :

- Le fichier train sera utilisé pour entraîner le modèle, c'est-à-dire que notre modèle apprendra à partir de ce fichier. Il contient toutes les variables indépendantes et la variable cible.
- Le fichier de test contient toutes les variables indépendantes, mais pas la variable cible. Nous appliquerons le modèle pour prédire la variable cible pour les données de test.

```
Train shape :  (614, 13)
Test  shape :  (367, 12)
```

```
train.head()
```

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

```
test.head()
```

Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

Gestion des doublons et des NAN

```
train.duplicated().sum()
```

```
0
```

```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

```
train.dropna(subset=['Gender'],inplace=True)
```

```
train['Married'] = train['Married'].fillna('No')
```

```
train.Dependents = train.Dependents.fillna('0')
rpl = {'0':'0', '1':'1', '2':'2', '3+':'3'}
train.Dependents = train.Dependents.replace(rpl).astype(int)
```

```
train['Self_Employed'] = train['Self_Employed'].fillna('No')
```

```
train.dropna(subset=['Credit_History'],inplace=True)
```

```
train.dropna(subset=['LoanAmount'],inplace=True)
```

```
train.isna().sum()
```

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

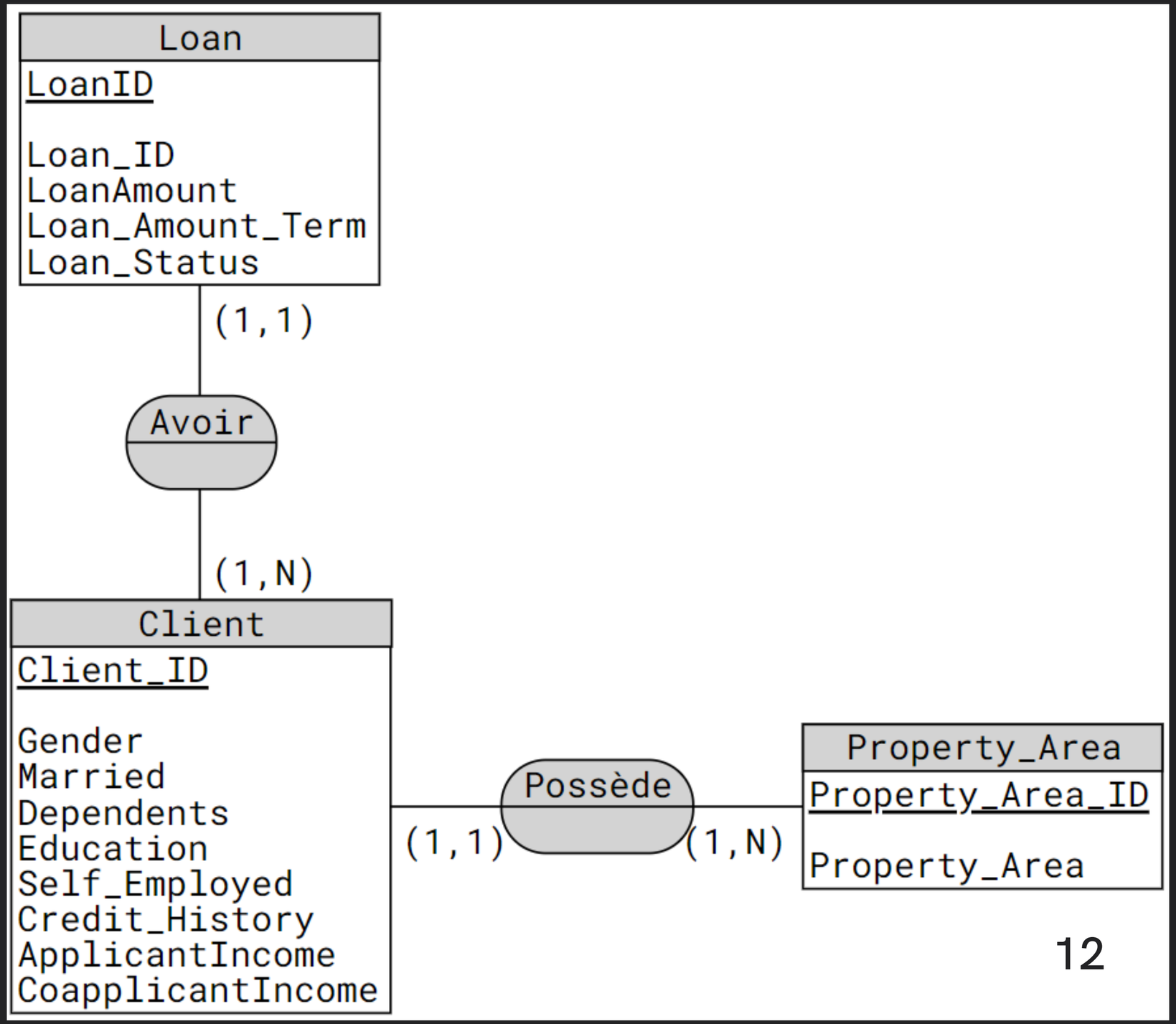
```
train.reset_index(drop=True)
```

```
Train shape : (517, 13)
```

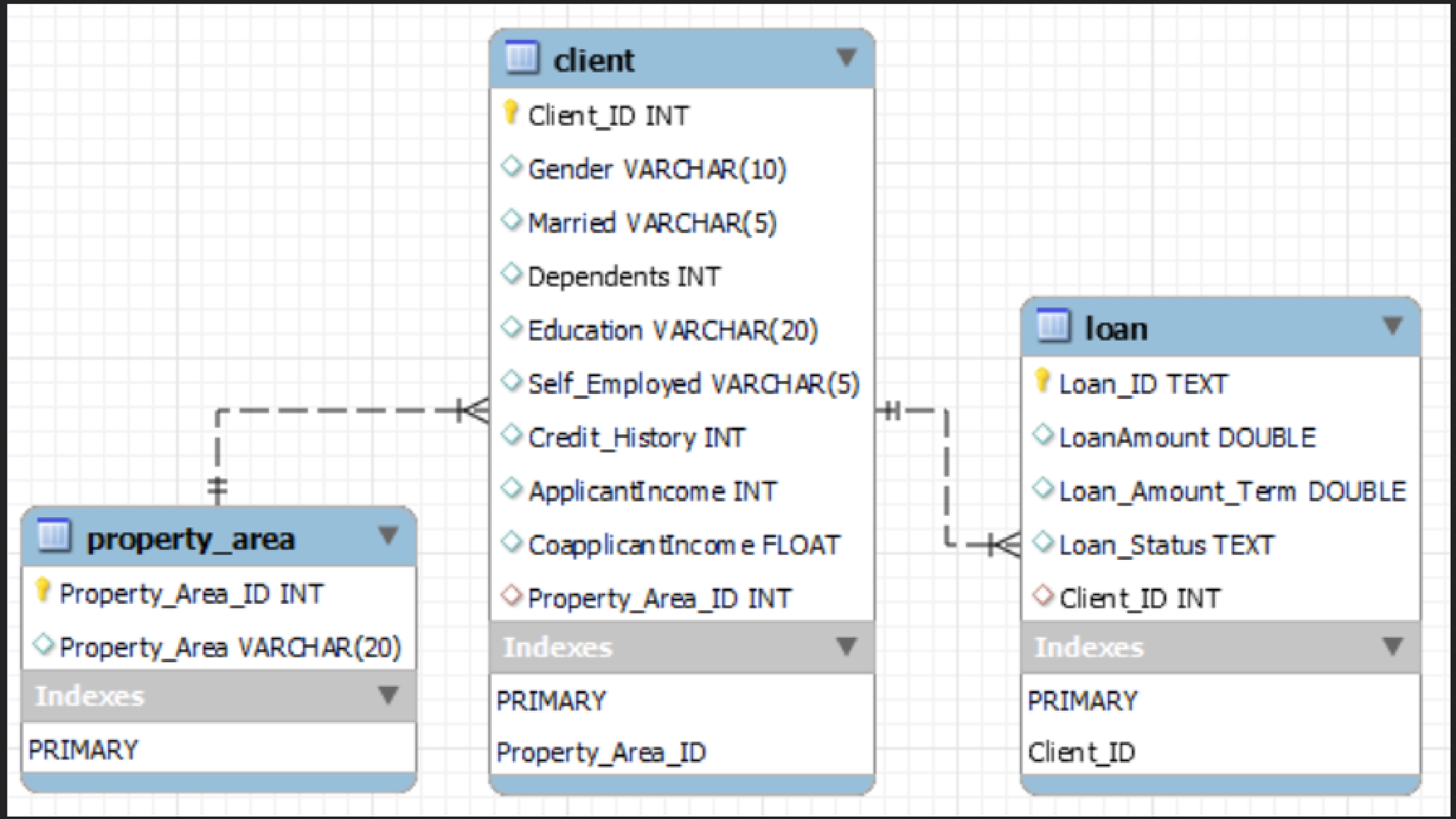
Modèle Conceptuel de Données (MCD)

On découpe notre dataframe en 3 tables :

- Loan
- Client
- Property_Area

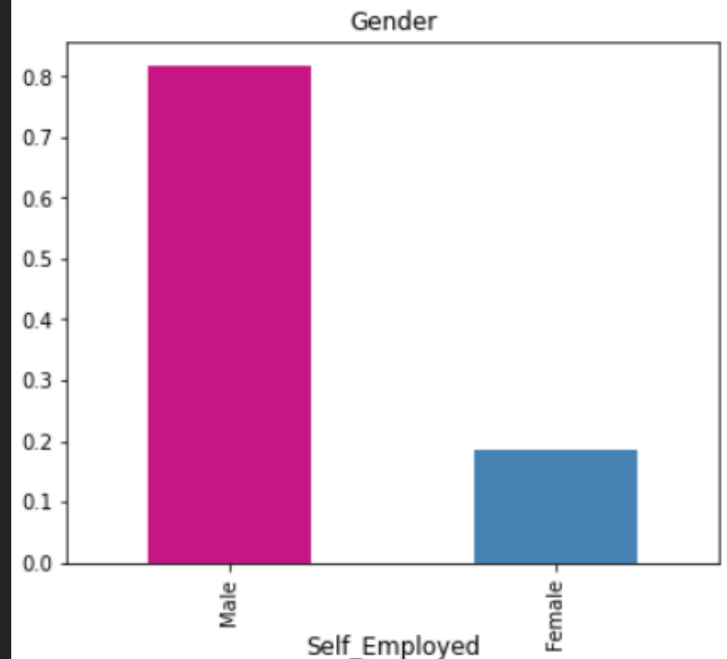


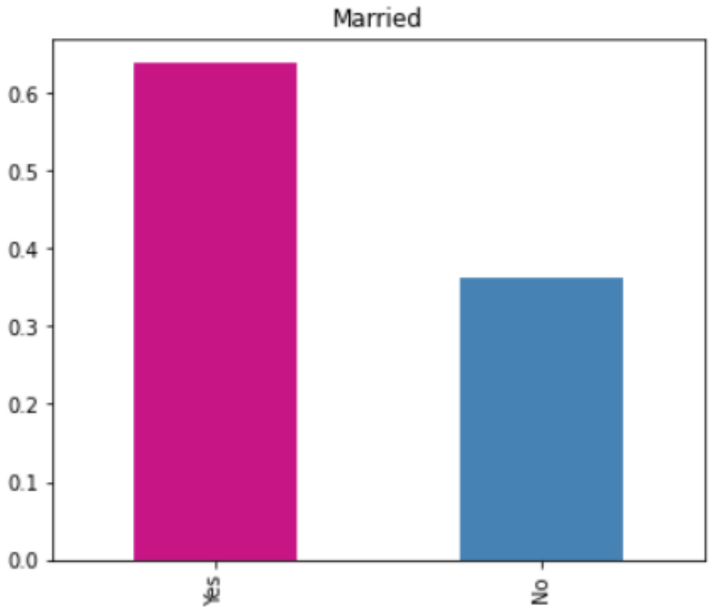
Modèle Physique de Données (MPD)

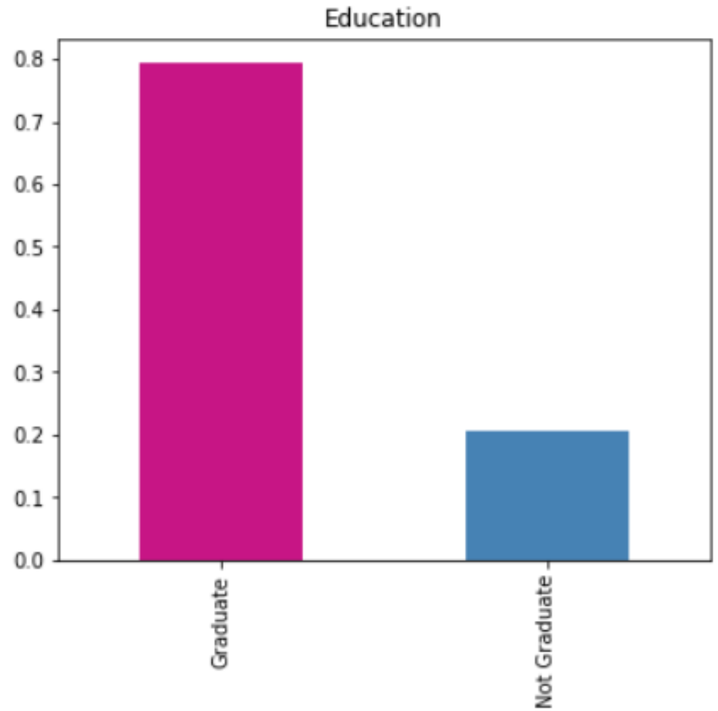


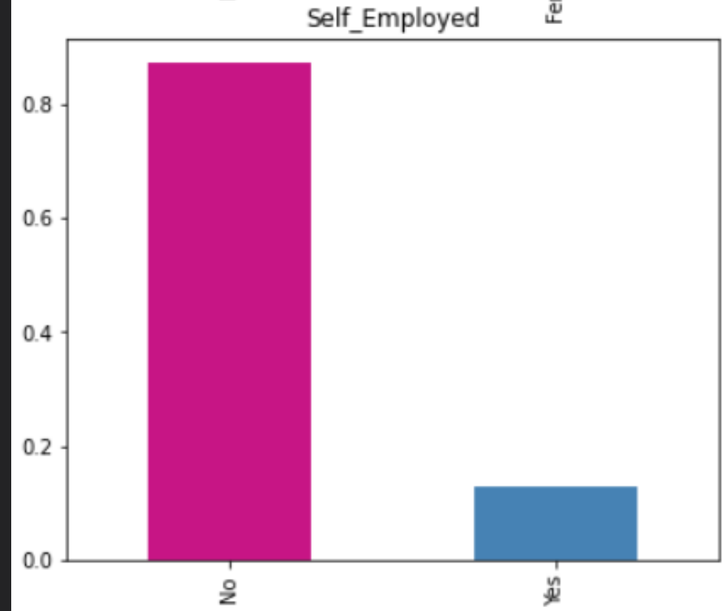
DataVisualisation

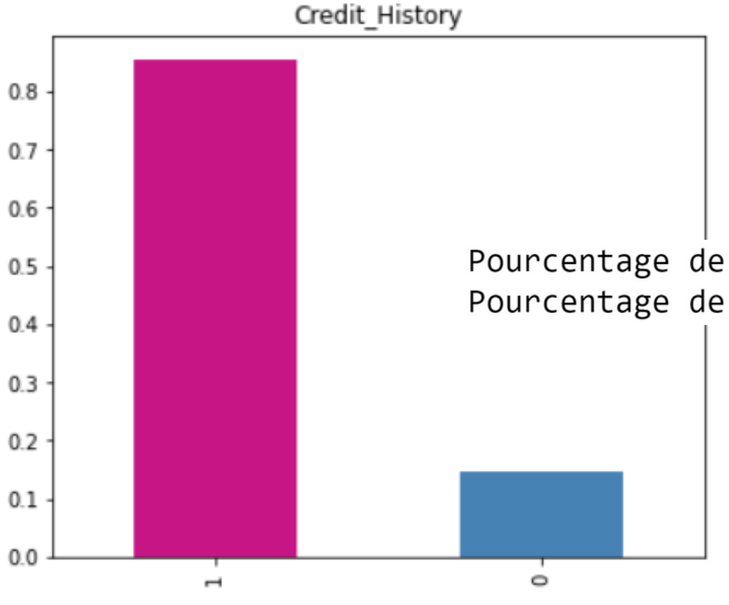
Variables indépendantes Catégorielles







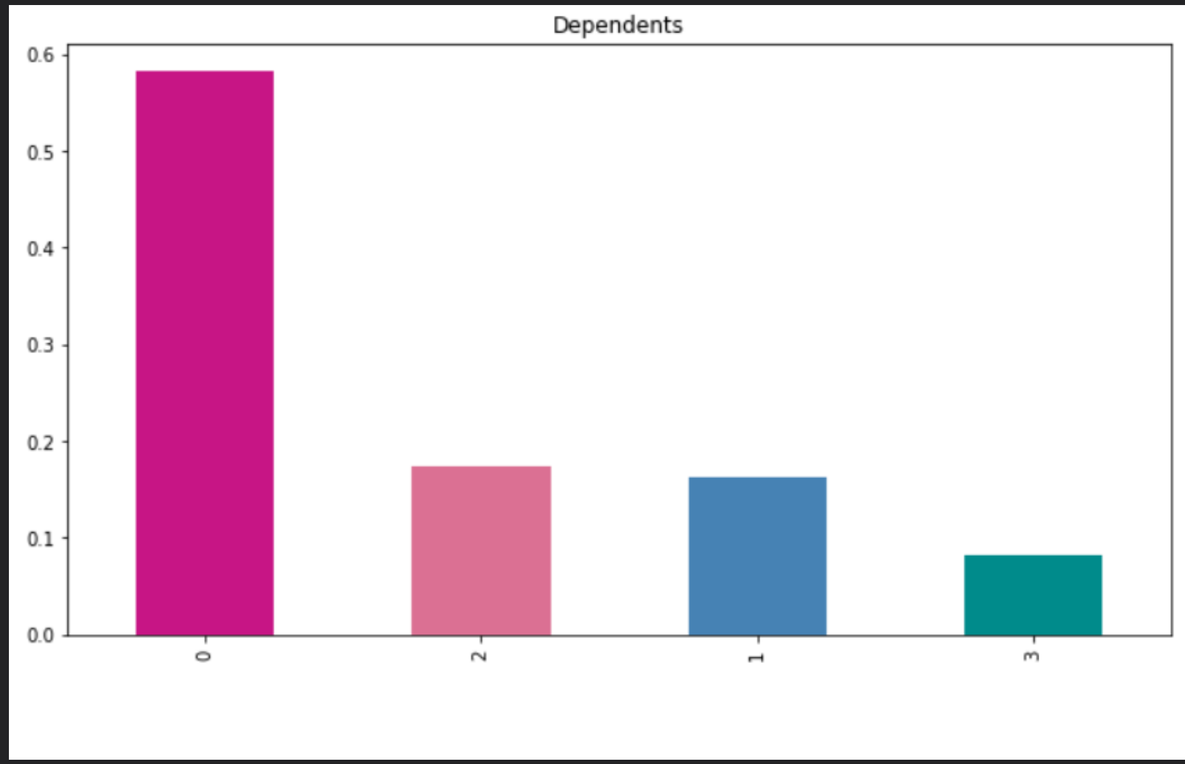


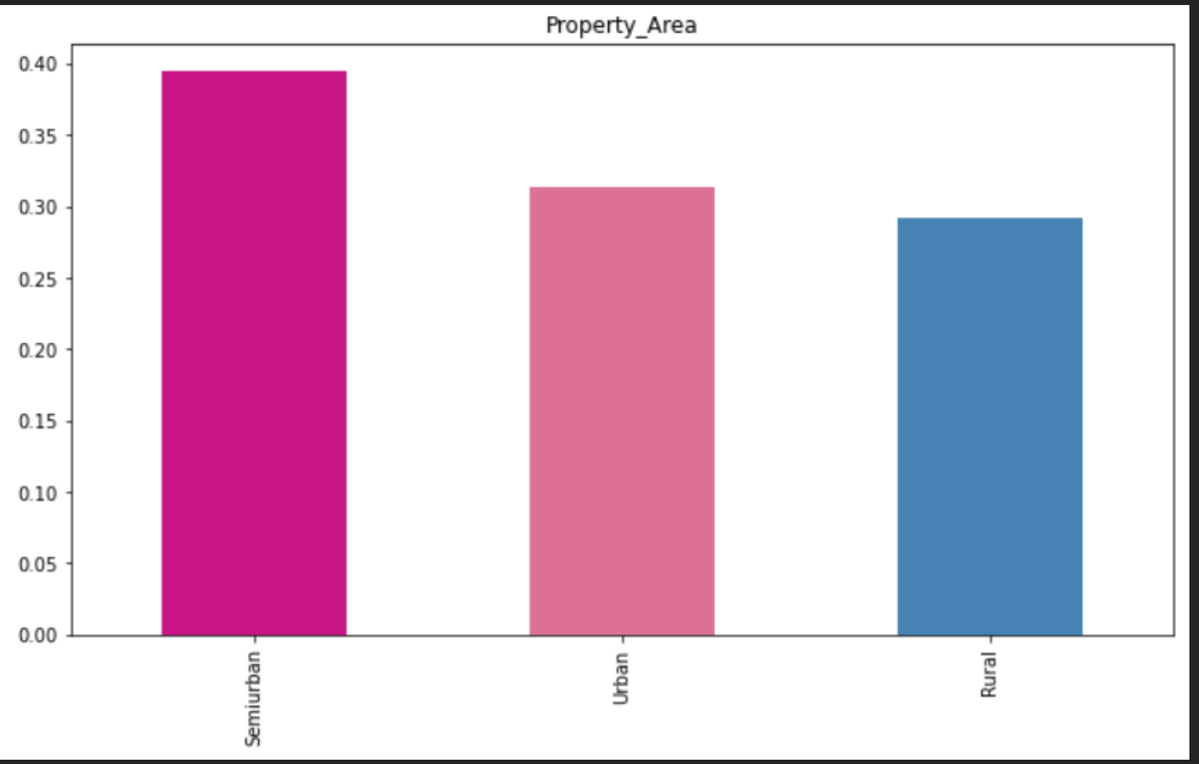


Pourcentage de bons antécédents de crédit : 85.30%
Pourcentage de mauvais antécédents de crédit : 14.70%

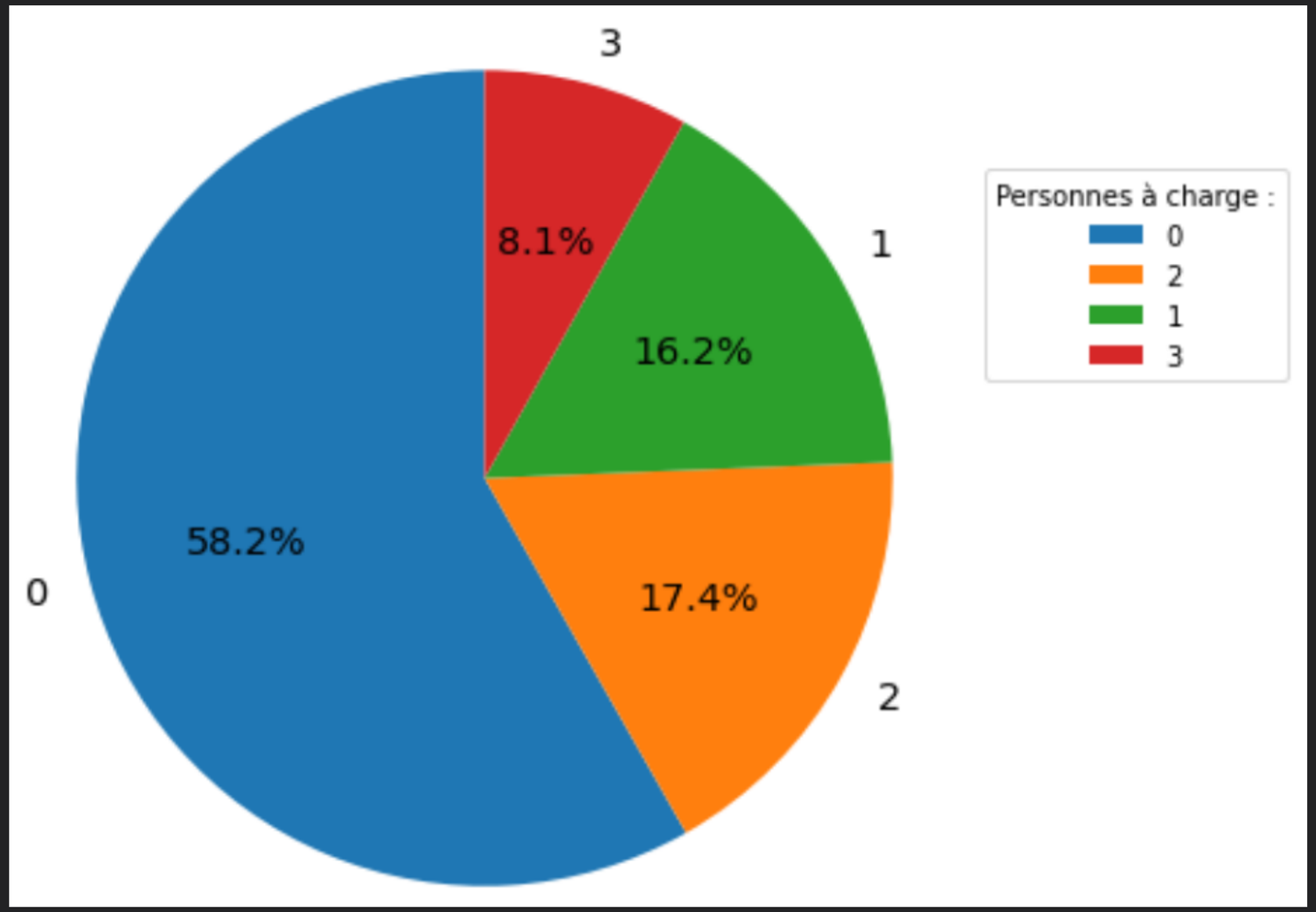
Variables indépendantes Catégorielles

Variables indépendantes Ordinales

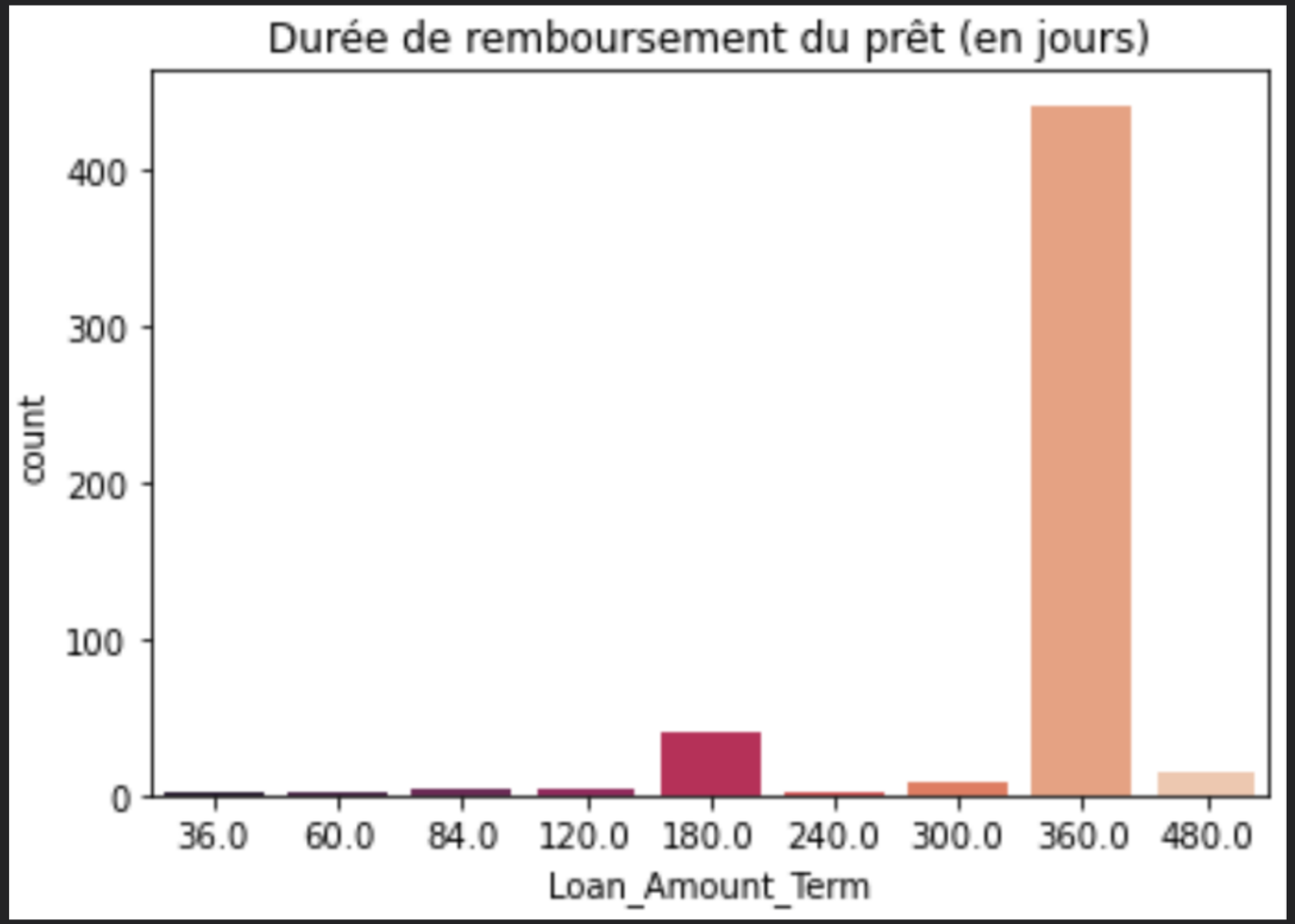




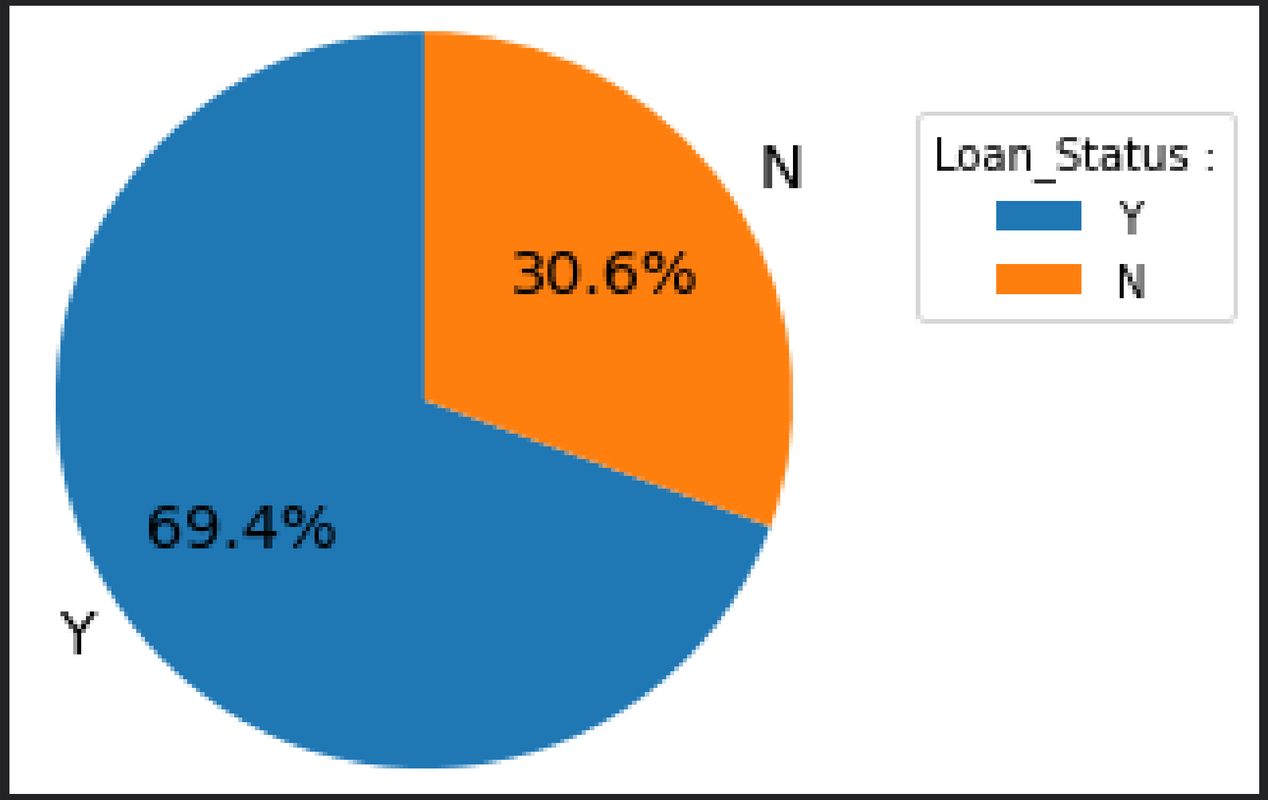
Nombre de personnes à charge du client



Durée de remboursement du prêt (en jours)



Pourcentage de prêts accordés



Data Imputation

```
new_train['Married'] = new_train['Married'].map(dict(Yes=1, No=0))
test['Married'] = test['Married'].map(dict(Yes=1, No=0))
```

```
new_train['Self_Employed'] = new_train['Self_Employed'].map(dict(Yes=1, No=0))
test['Self_Employed'] = test['Self_Employed'].map(dict(Yes=1, No=0))
```

```
new_train['Credit_History'] = new_train['Credit_History'].astype(int)
```

```
0   Loan_ID              517 non-null    object
1   Gender               517 non-null    int64
2   Married              517 non-null    int64
3   Dependents           517 non-null    int64
4   Education            517 non-null    int64
5   Self_Employed        517 non-null    int64
6   ApplicantIncome      517 non-null    int64
7   CoapplicantIncome    517 non-null    float64
8   LoanAmount           517 non-null    float64
9   Loan_Amount_Term     517 non-null    float64
10  Credit_History       517 non-null    int64
11  Property_Area        517 non-null    int64
12  Loan_Status          517 non-null    int64
```

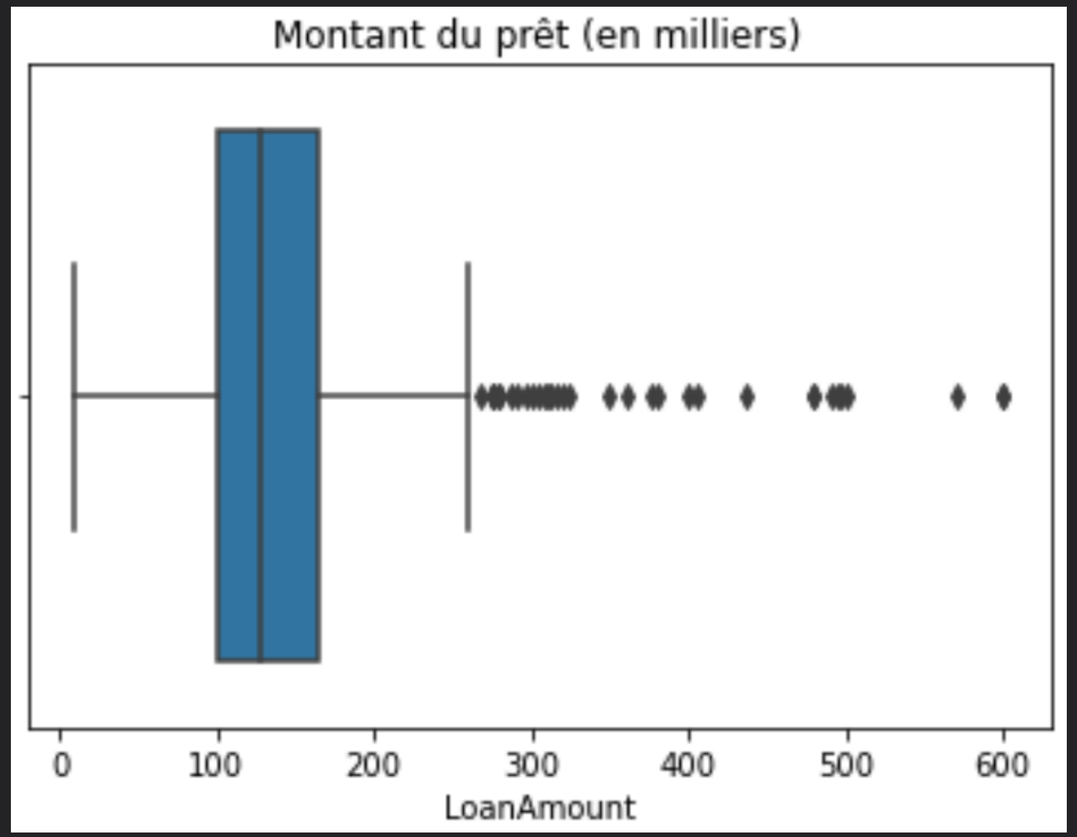
```
to_numeric = {'Male': 0, 'Female': 1,
              'Graduate': 1, 'Not Graduate': 0,
              'Urban': 3, 'Semiurban': 2,'Rural': 1,
              'Y': 1, 'N': 0,
              '3+': 3}
```

```
new_train = new_train.applymap(lambda elem: to_numeric.get(elem) if elem in to_numeric else elem)
```

Gestion des outliers

```
print("Old Shape: ", new_train.shape)
# Upper bound
upper = np.where(new_train['LoanAmount'] >= 320)
print(upper[0])
''' Removing the Outliers '''
new_train.drop(labels=upper[0], axis=0, inplace = True)
#new_train.drop(lower[0], inplace = True)
print("New Shape: ", new_train.shape)
```

```
Old Shape:  (517, 12)
[  8  28 127 146 234 261 271 279 309 312 345 365 410 440 441 450 471 508]
New Shape:  (499, 12)
```



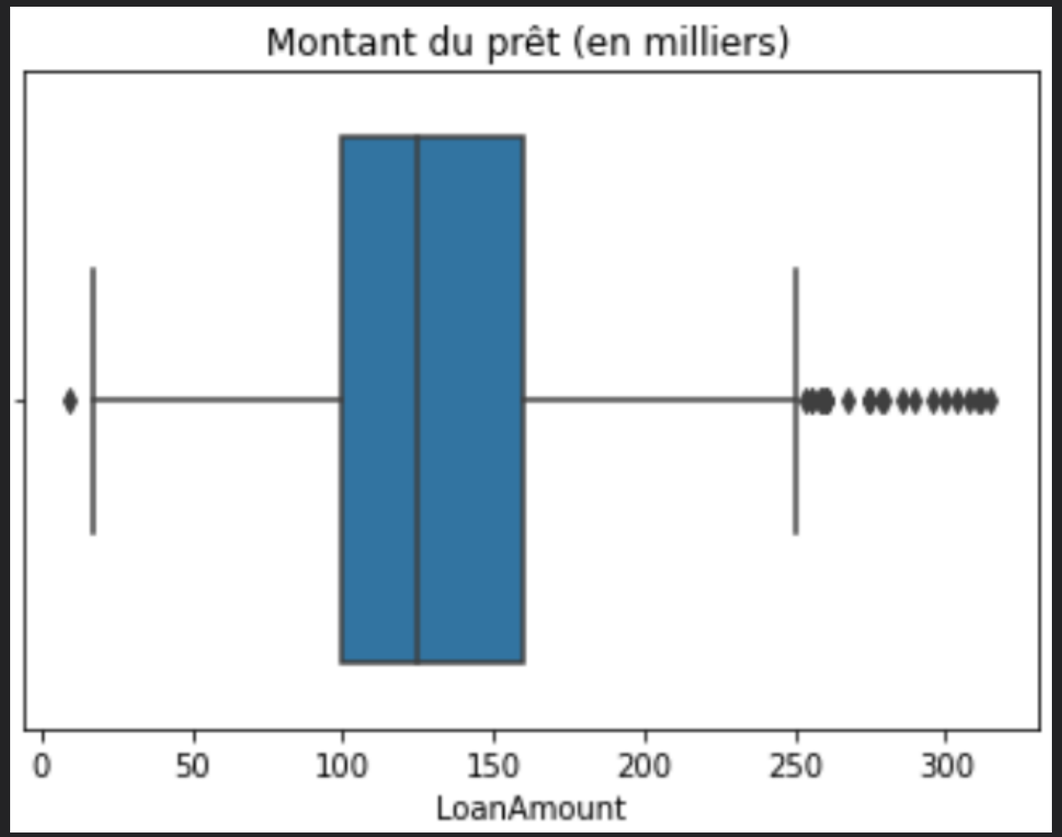
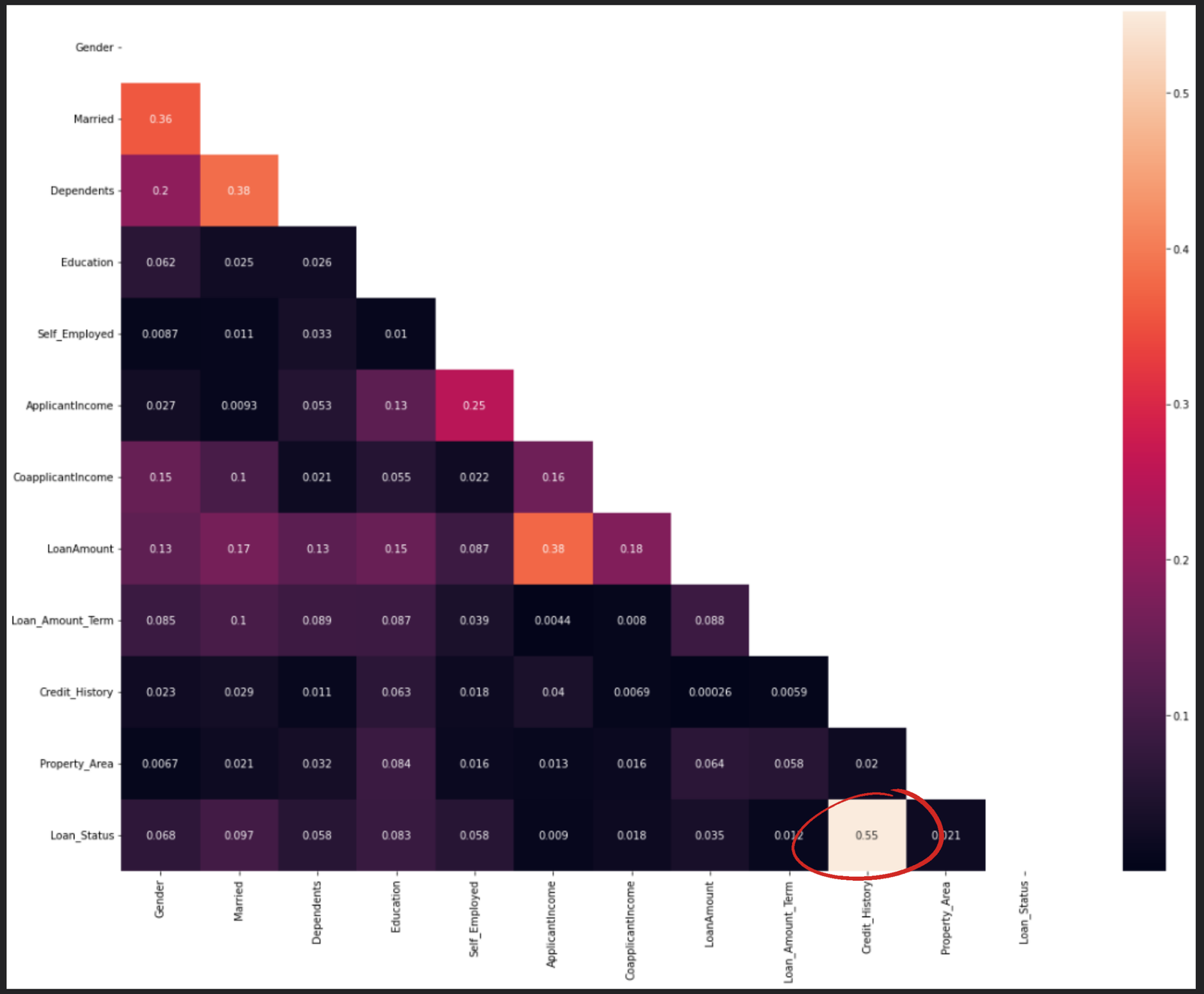


Tableau de corrélation



Machine Learning

Etat de l'art

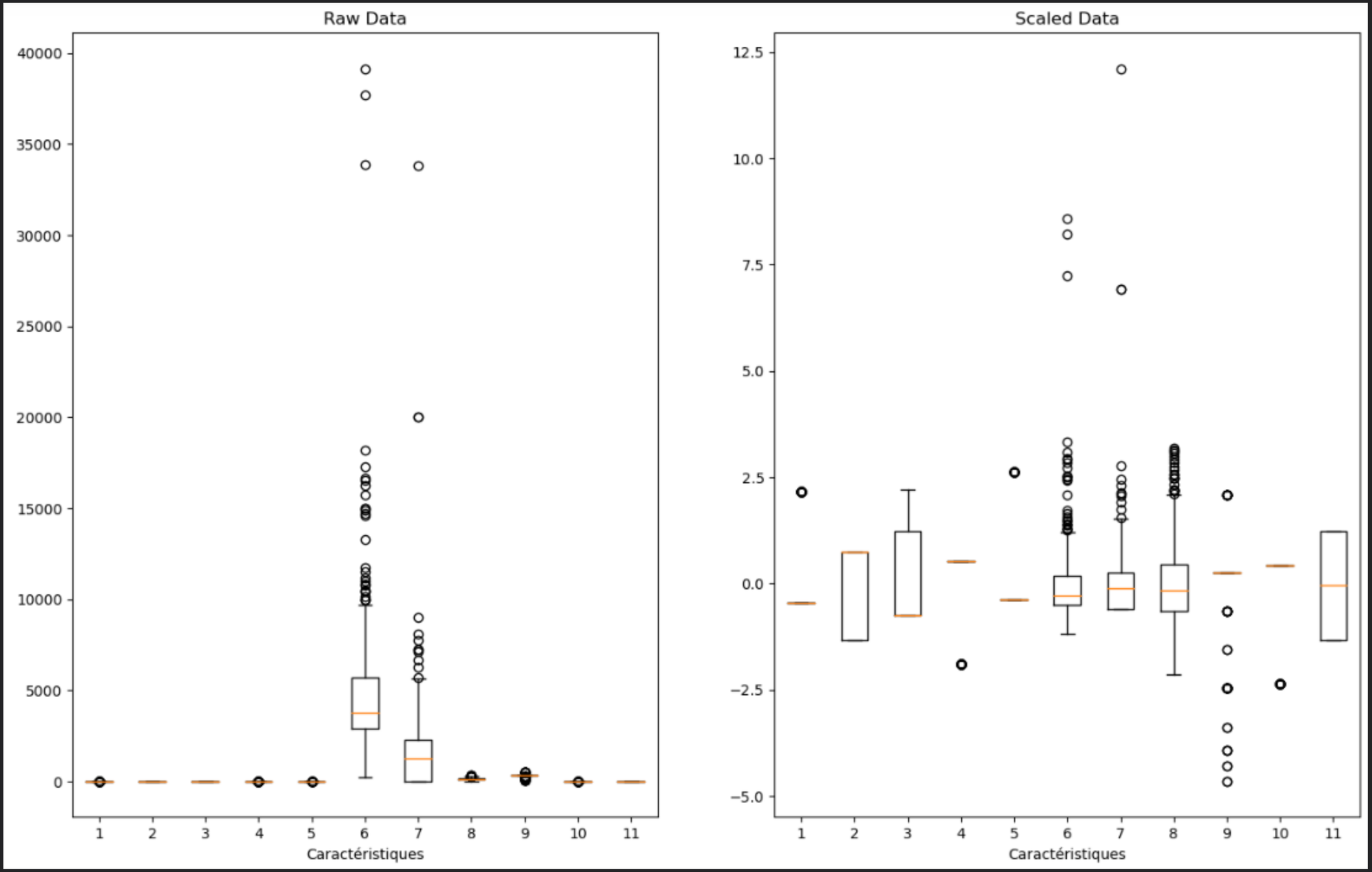
	Supervised Learning		Unsupervised Learning
#	Classification Algorithms	Prediction Algorithms	Clustering Algorithms
1	Neural Network	Linear Regression	K-Means
2	SVM	Random Forest	Hierarchical
3	CART	CART	
4	Naive Bayes	Elastic Net Linear Regression	
5	Random Forest		
6	Logistic Regression		

Optimisation des hyper-paramètres

Librairie
Sklearn GridSearchCV

Modèle
« K-plus proches voisins Régression »

Features Scaling : StandardScaler



Modèle retenu : RandomForestClassifier

```
param_grid = {
    'n_estimators': [100, 150, 200, 250, 300, 350, 400, 450, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8,10],
    'criterion' :['gini', 'entropy']
}
```

```
gridSearchCV2.best_params_
```

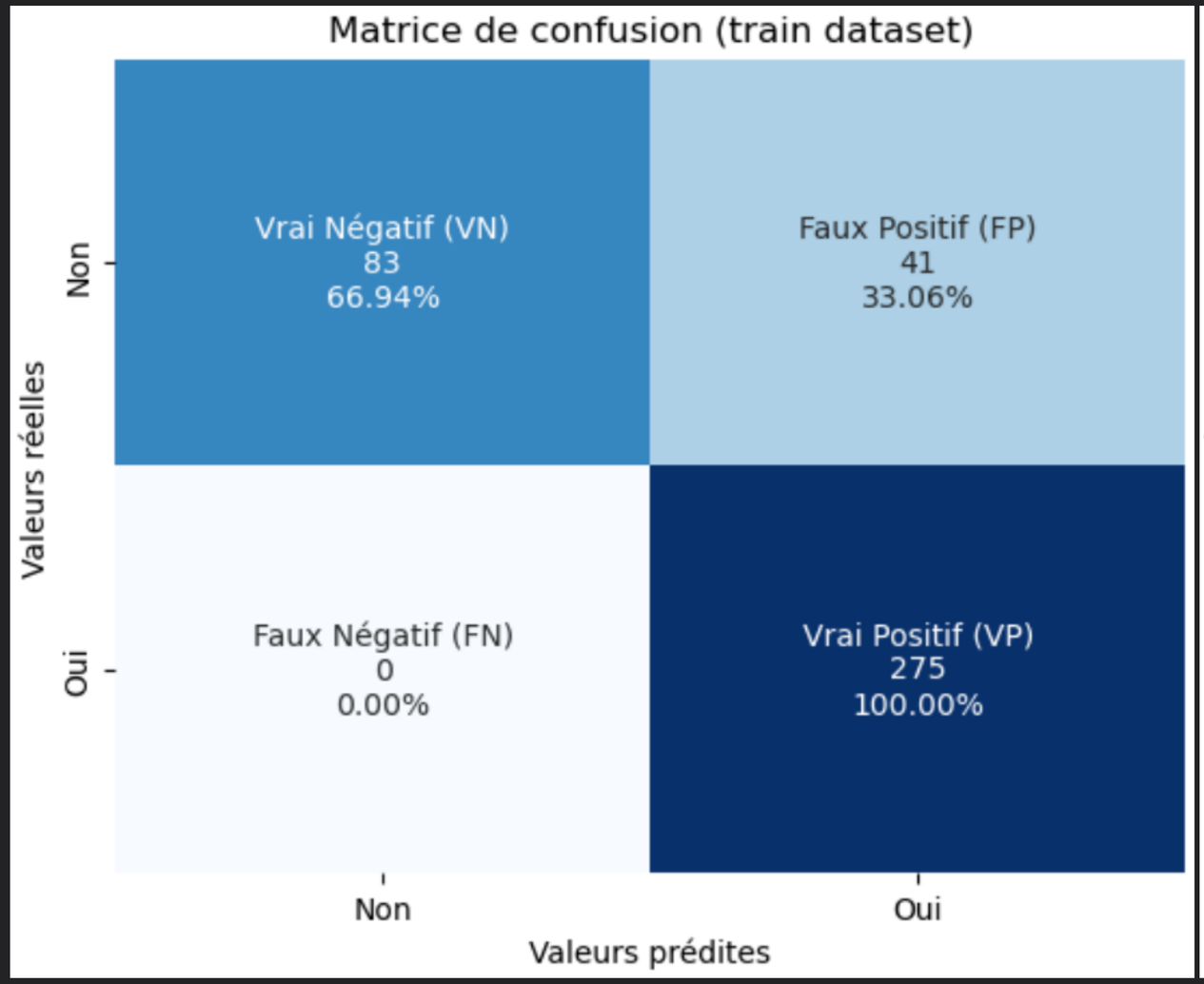
```
{'criterion': 'entropy',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 250}
```

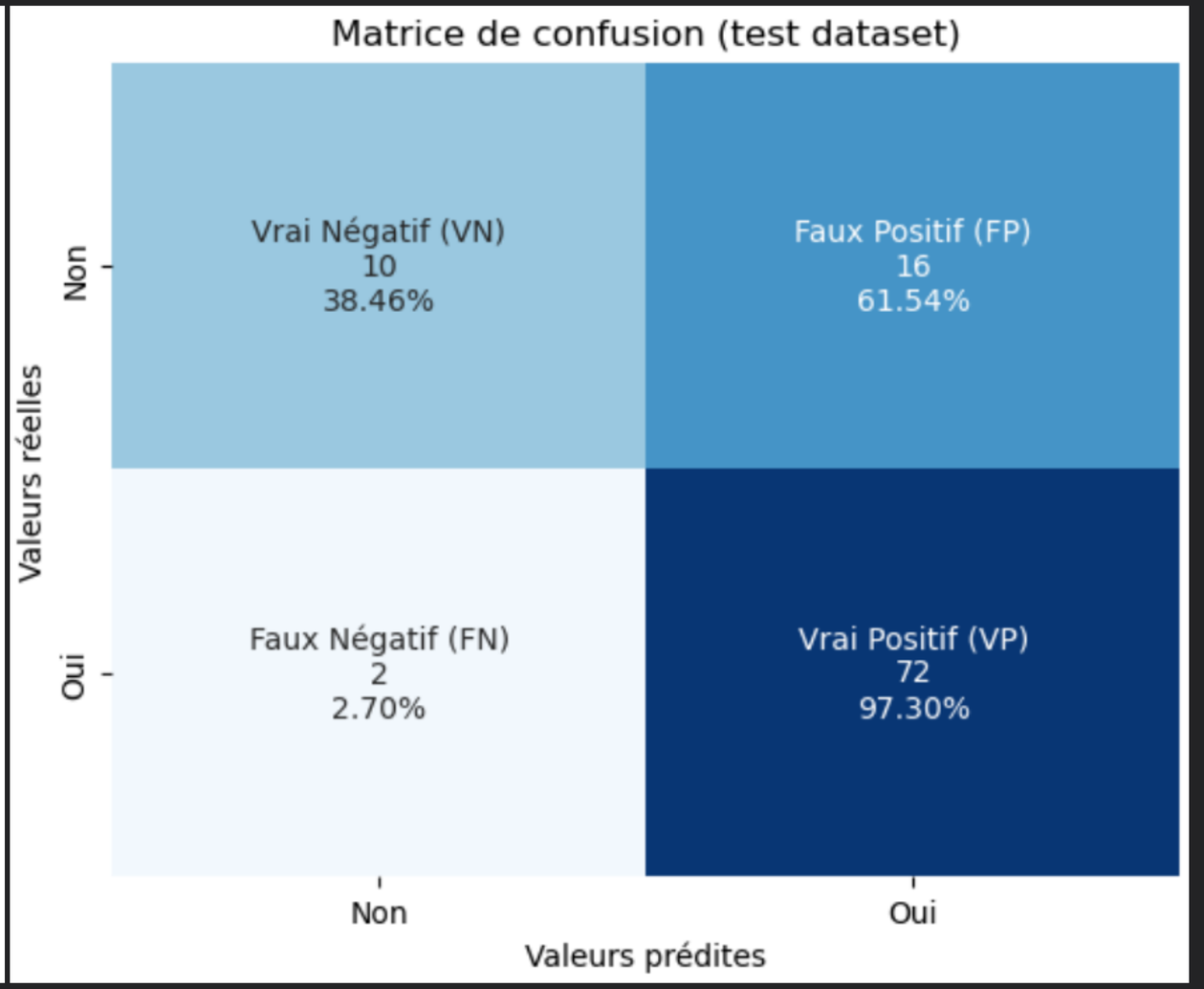
```
print(classification_report(y_train, y_predict_train2))
```

	precision	recall	f1-score	support
0	1.00	0.67	0.80	124
1	0.87	1.00	0.93	275
accuracy			0.90	399
macro avg	0.94	0.83	0.87	399
weighted avg	0.91	0.90	0.89	399

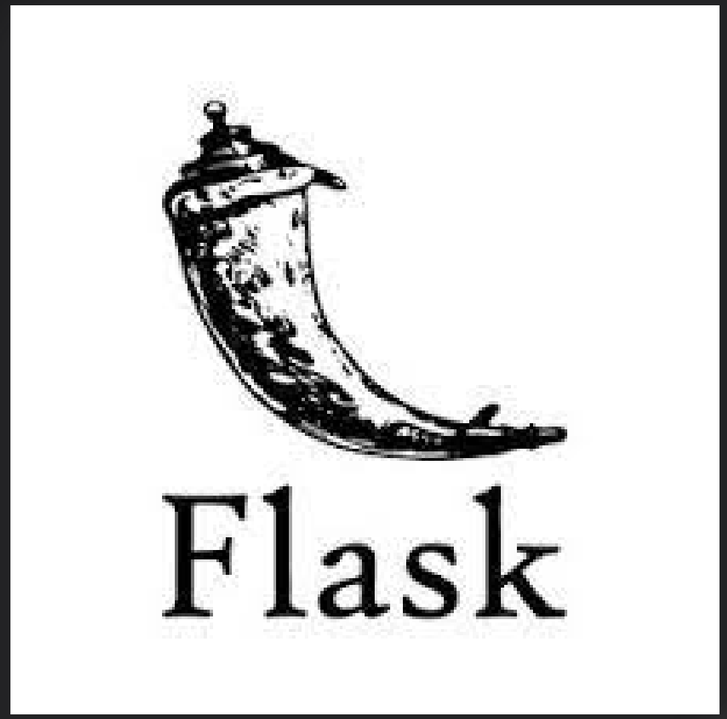
```
MLP accuracy: 82.00%
```

Matrice de confusion : RandomForestClassifier

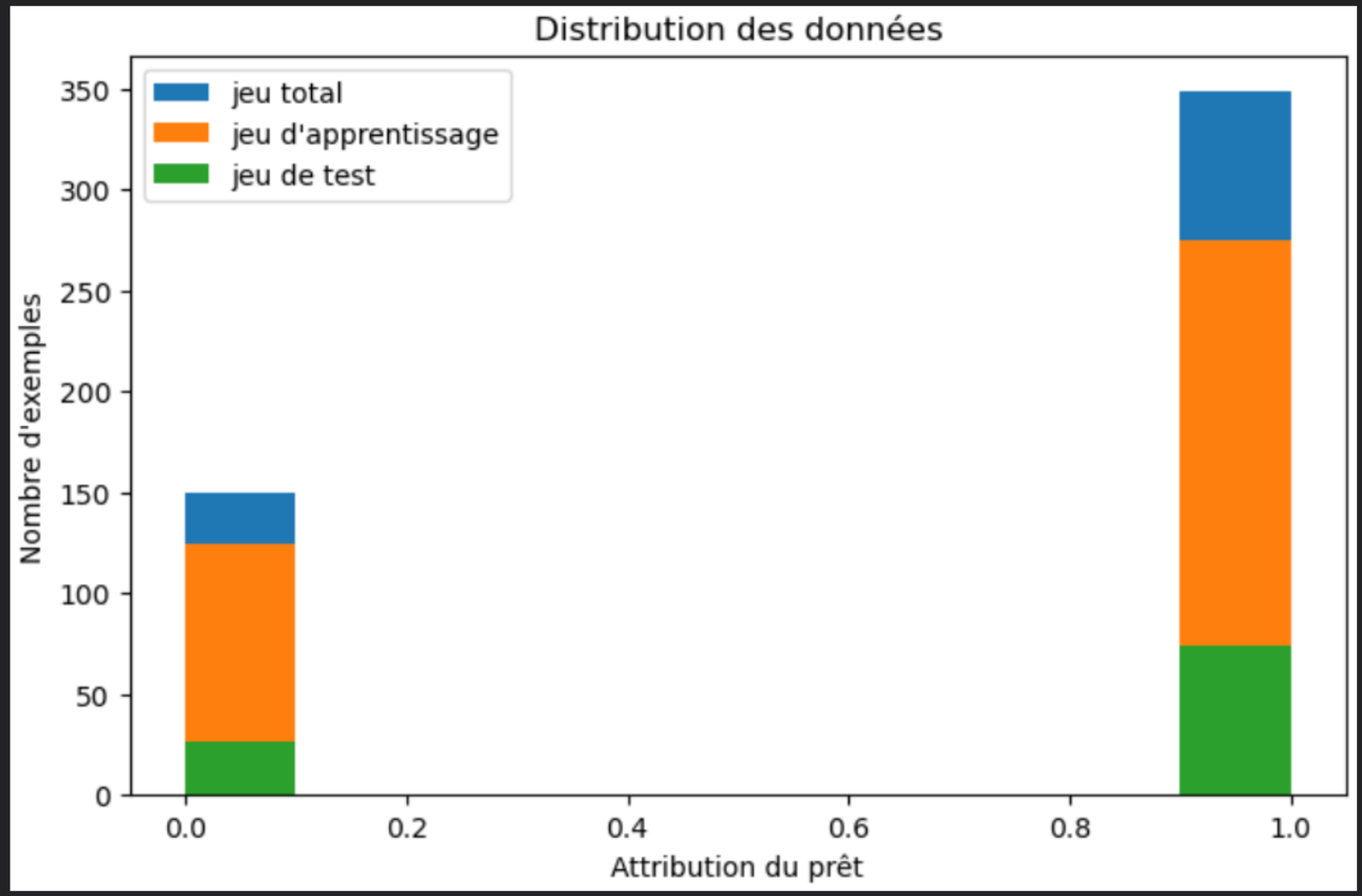




API



Retours d’expérience



Thank You