PERSODATA

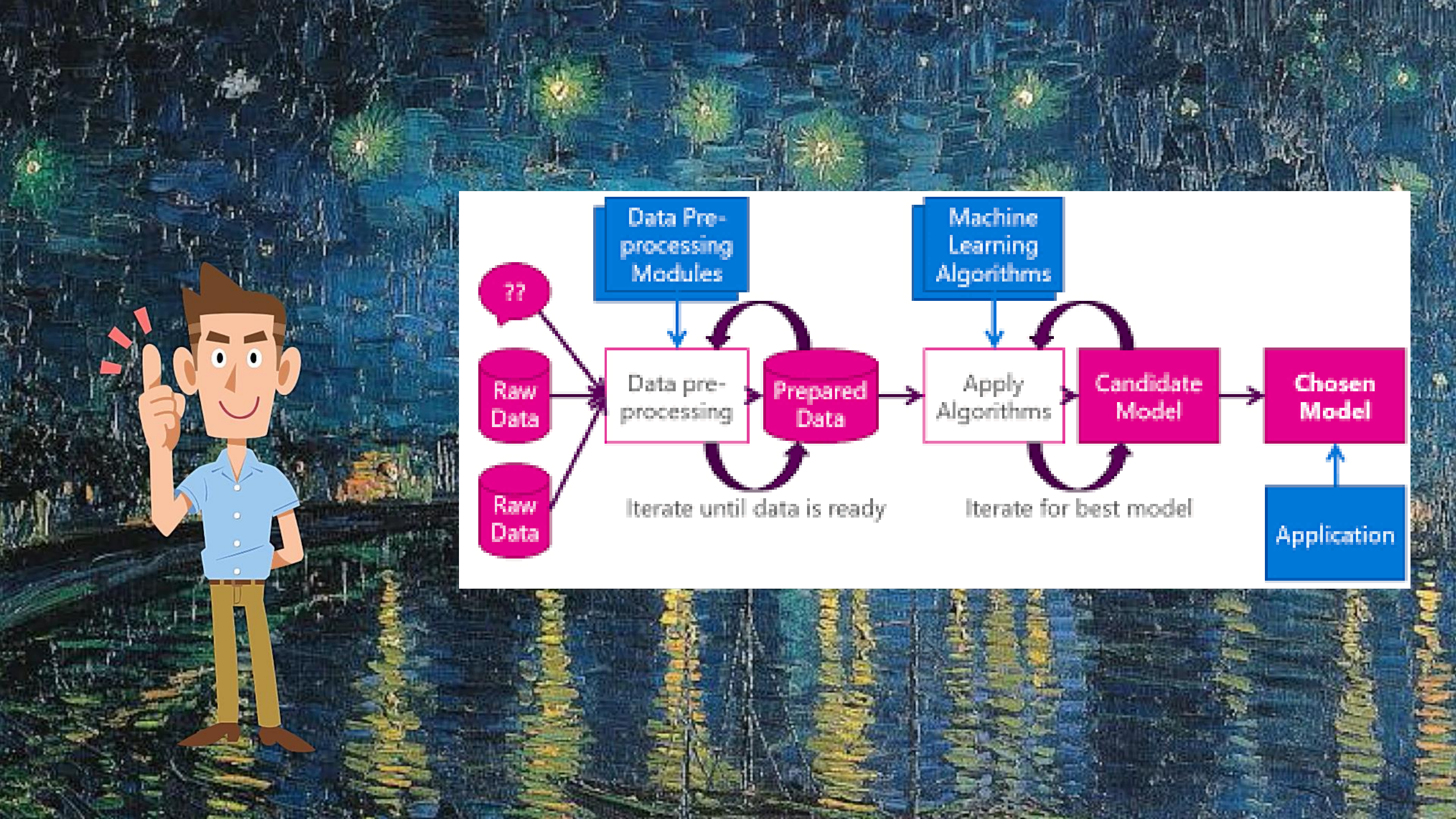
Chef de Projet:
AMBEMOU Loba

Groupe Preprocessing:
TALSSI Mehdi
PRIEPRZK Floriana
AMBEMOU Loba

Groupe Prédiction:
VERANI Damien
LEVY Kevin

Groupe Visualisation:
JOUET Edouard
TRINH Donovan

nombre d'exemples	Nombre de features	« sparsity »	Variables categorielles	Données manquantes	Dataset	Nombre d'exemples par classes(2 classes fake et real)
65856	200	0	0	0	Trainnig	Real :&32886 Fake :32970
18817	200	0	0	0	Test	Real :9485 Fake :9332
9408	200	0	0	0	Validation	Real :4670 Fake :4738


Data Pre-processing Modules
Machine Learning Algorithms
??
Raw Data
Raw Data
Data pre-processing
Prepared Data
Apply Algorithms
Candidate Model
Chosen Model
Application
Iterate until data is ready
Iterate for best model

I- Preprocessing

```
class Preprocessing(BaseEstimator):

    def __init__(self):
        lil_clf = SVC(kernel='linear') # classifieur lineaire
        self.transformer = PCA(n_components=100) # on veut que le resultat soit compose de 100 features
        self.pipe = Pipeline(BaseEstimator)
        self.pipe
        Pipeline(memory=None, steps=[('reduction_dim',fit(self,data.data['Xtrain'],data.data['Ytrain'])), ('lil_clf', lil_clf)])

    def fit(self, Xtrain, Ytrain):
        # premiere methode de preprocessing
        X_scaled = preprocessing.scale(X_train)
        Y_scaled = preprocessing.scale(Y_train)
        Xtrain_transf=self.transformer.fit_transform(Xtrain)
        Ytrain_transf=self.transformer.fit_transform(Ytrain)
        return Xtrain_transf,Ytrain_transf

    def fit_transform(self, X, Y):
        return self.transformer.fit_transform(X,Y)

    def transform(self, X, Y):
        return self.transformer.transform(X,Y)
```

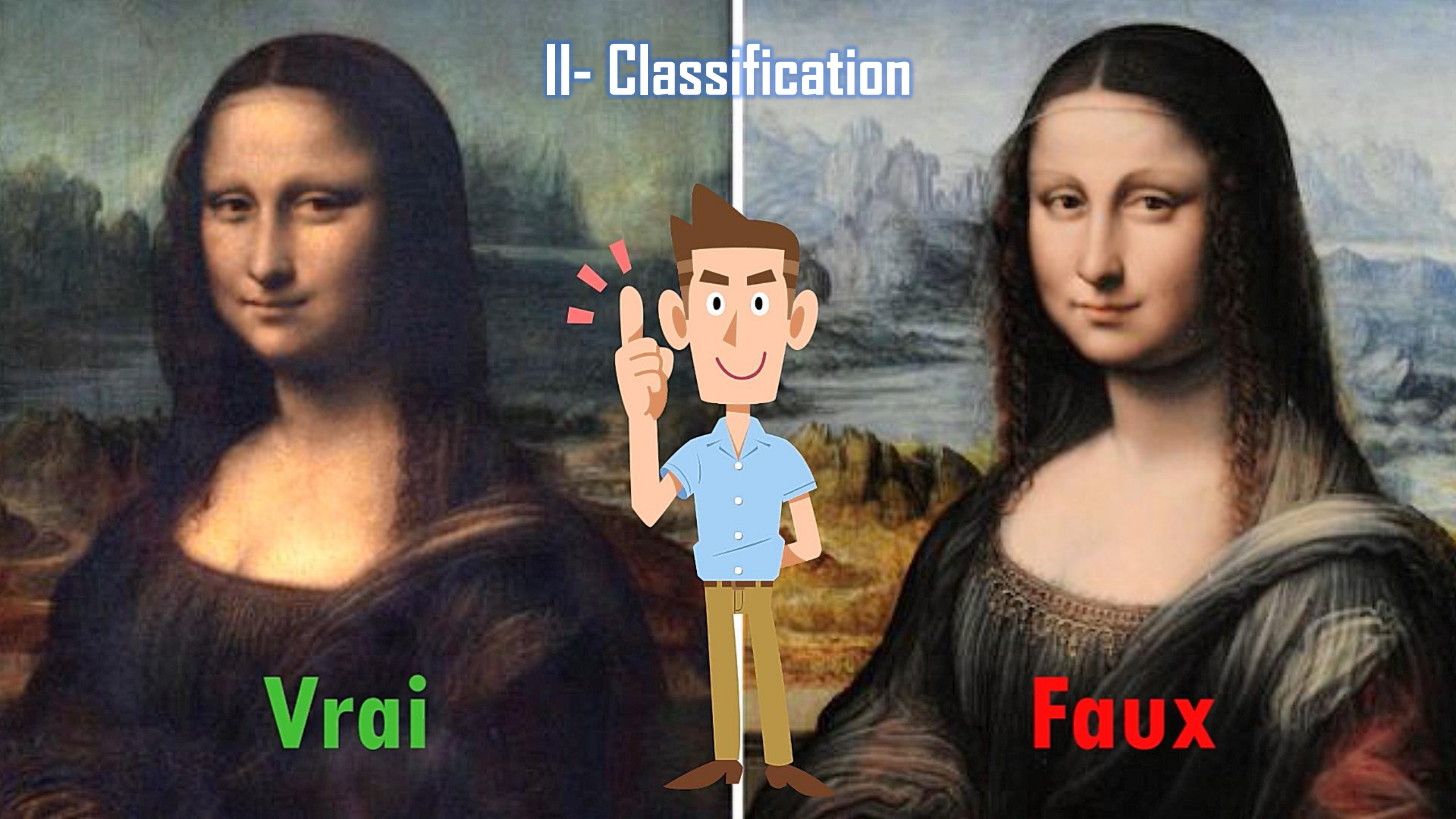
I- Preprocessing

```python
class Preprocessing(BaseEstimator):

    def __init__(self):
        lil_clf = SVC(kernel='linear') # classifieur lineaire
        self.transformer = PCA(n_components=100) # on veut que le resultat soit compose de 100 features
        self.pipe = Pipeline(BaseEstimator)
        self.pipe
        Pipeline(memory=None, steps=[('reduction_dim',fit(self,data.data['Xtrain'],data.data['Ytrain'])), ('lil_clf', lil_clf)])

    def fit(self, Xtrain, Ytrain):
        # premiere methode de preprocessing
        X_scaled = preprocessing.scale(X_train)
        Y_scaled = preprocessing.scale(Y_train)
        Xtrain_transf=self.transformer.fit_transform(Xtrain)
        Ytrain_transf=self.transformer.fit_transform(Ytrain)
        return Xtrain_transf,Ytrain_transf

    def fit_transform(self, X, Y):
        return self.transformer.fit_transform(X,Y)

    def transform(self, X, Y):
        return self.transformer.transform(X,Y)
```

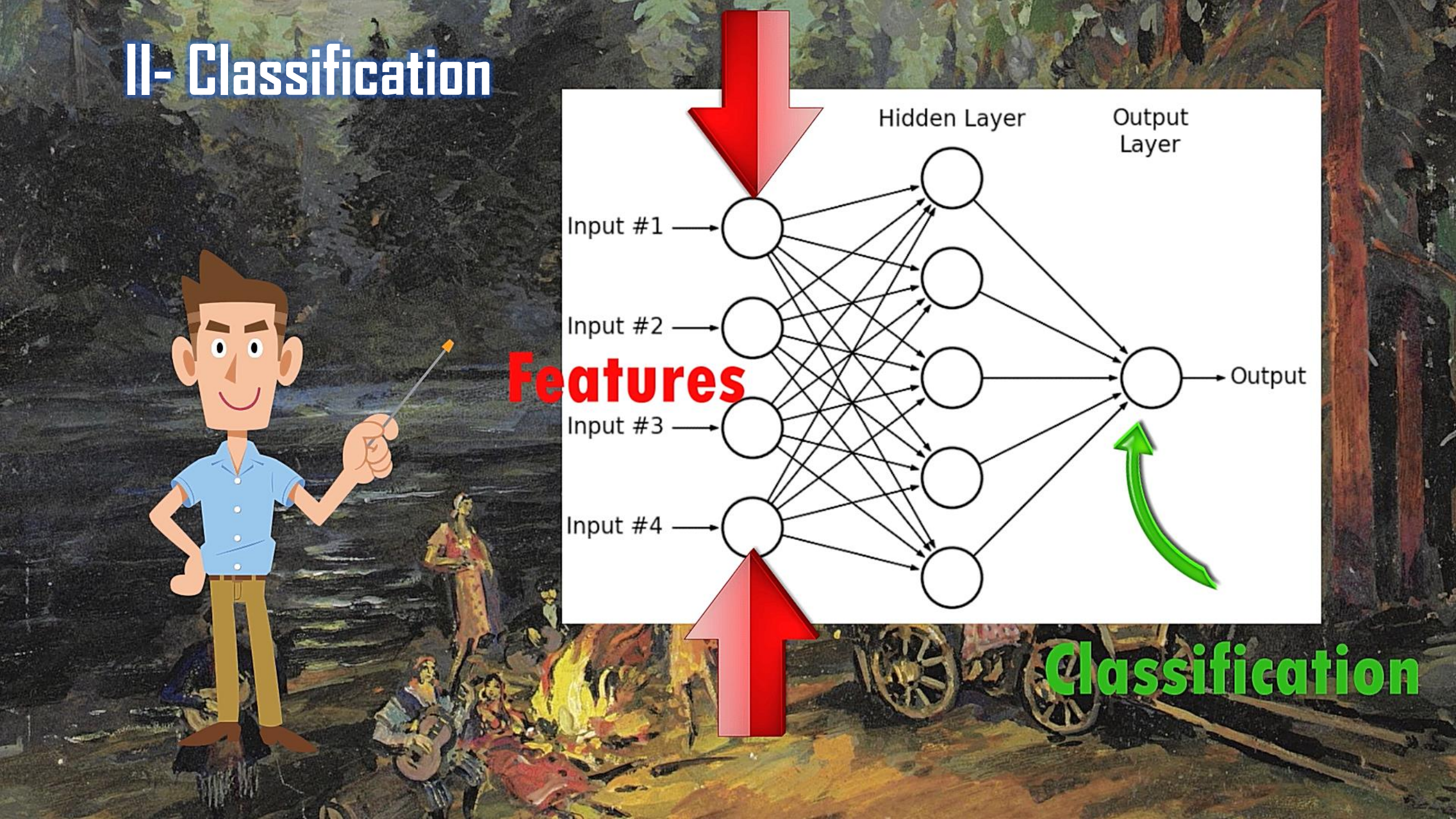
Standardiser

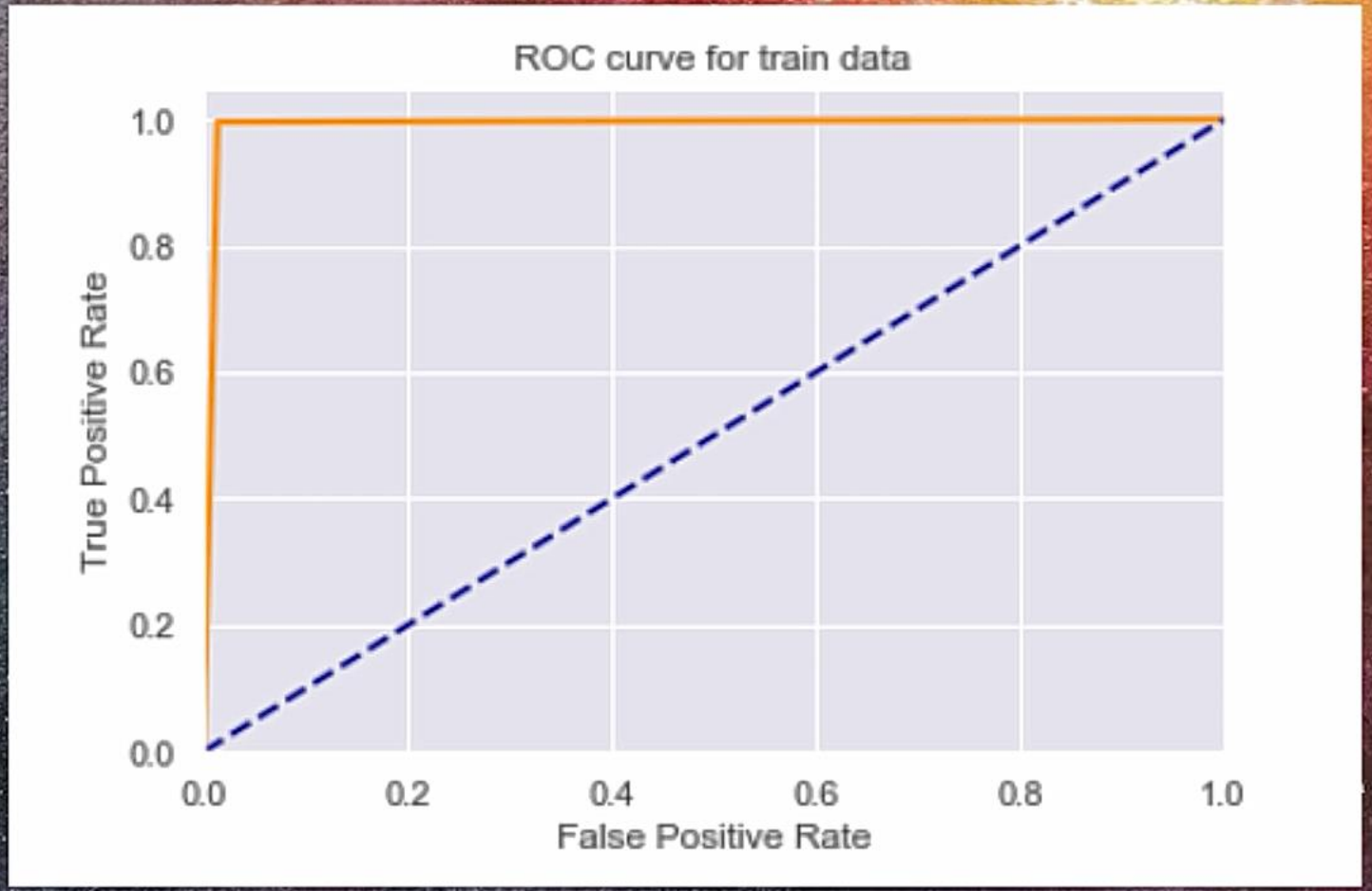
Réduire

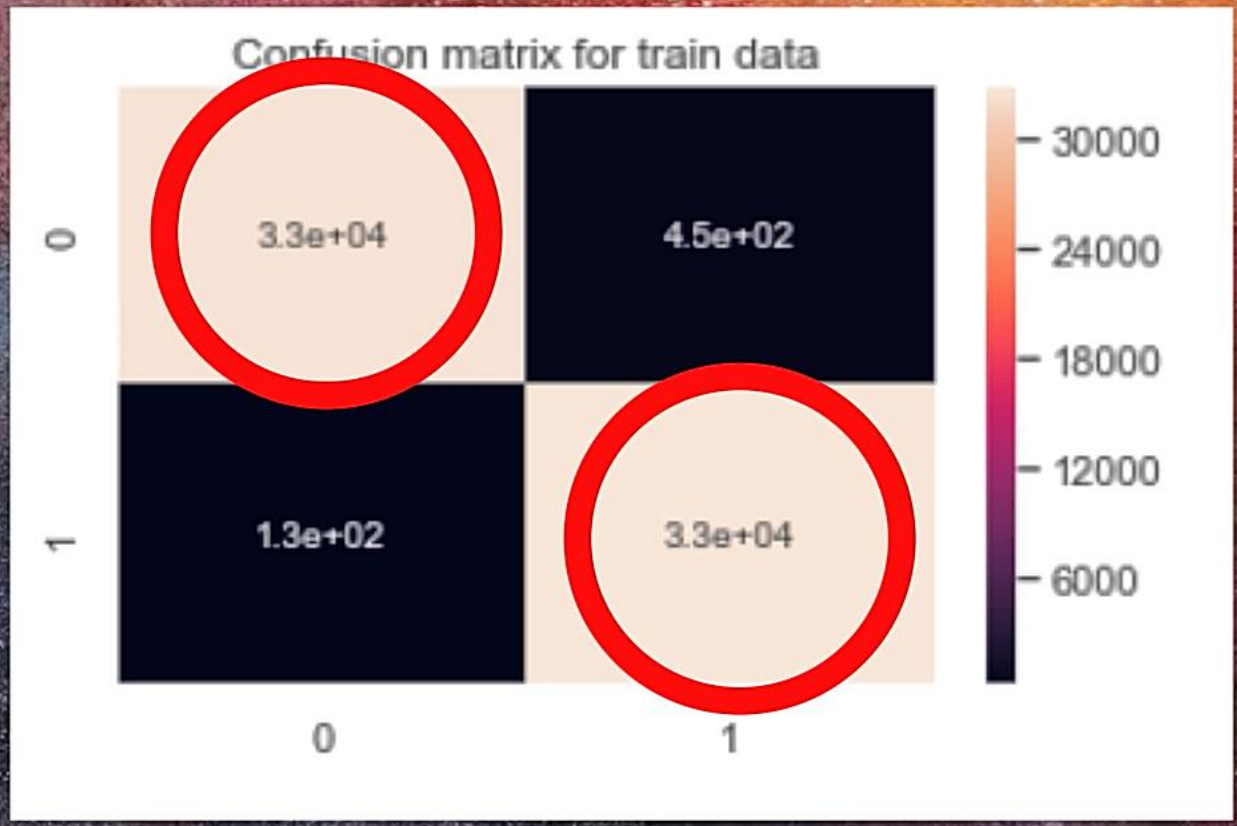
II- Classification

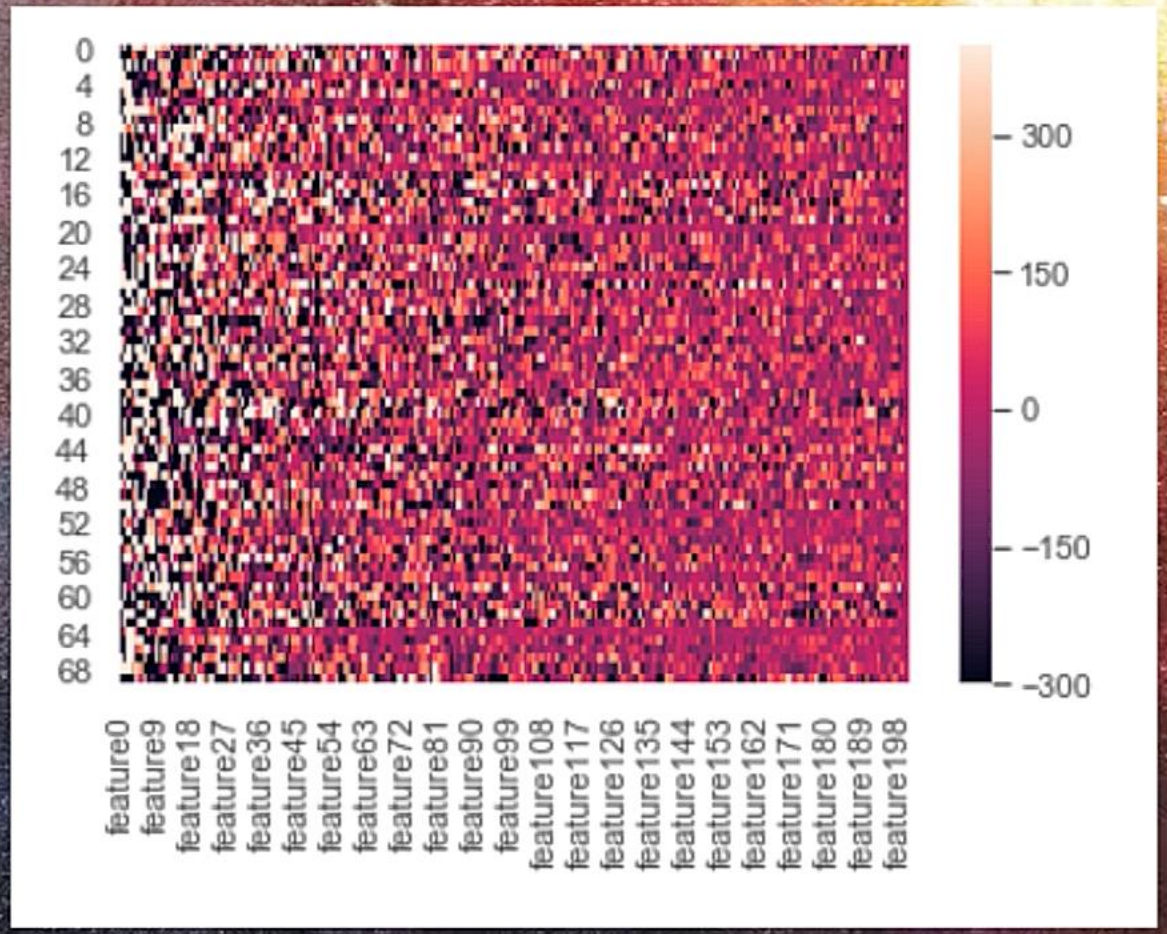


II- Classification




ROC curve for train data
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Confusion matrix for train data
3.3e+04
4.5e+02
1.3e+02
3.3e+04
0
1
0
1
30000
24000
18000
12000
6000



Seaborn MatPlotLib

Résultats :

RESULTS

#	User	Entries	Date of Last Entry	Prediction score ▲	Duration ▲	Detailed Results
1	reference2	1	01/24/19	0.9467 (1)	0.00 (1)	View
2	Yanis47S	5	03/01/19	0.8702 (2)	0.00 (1)	View
3	mokakill	10	03/03/19	0.8674 (3)	0.00 (1)	View
4	Kahlo	3	03/03/19	0.8593 (4)	0.00 (1)	View
5	picasso	11	03/20/19	0.8581 (5)	0.00 (1)	View
6	vinci	16	03/20/19	0.8137 (6)	0.00 (1)	View

0.8137

Quelques conseils :

- Bien s'organiser
- Travailler en groupe
- Etre autonome