

RAPPORT

MACHINE LEARNING
MINI PROJET D'APPLICATION
MAI 2024

RÉALISÉ PAR :
HAJAJ FATIMA ZAHRA
DAHMANI OUMAYMA
EL FAHSSI HANAN

ENCADRÉ PAR :
M. HAJA ZAKARIA

SOMMAIRE

01. INTRODUCTION

02. OBJECTIF DU PROJET

03. SOURCE DES DONNÉES

04. ANALYSE EXPLORATOIRE DES DONNÉES

**05. DESCRIPTION DE LA PHASE DE PRE-
PROCESSING DES DONNÉES**

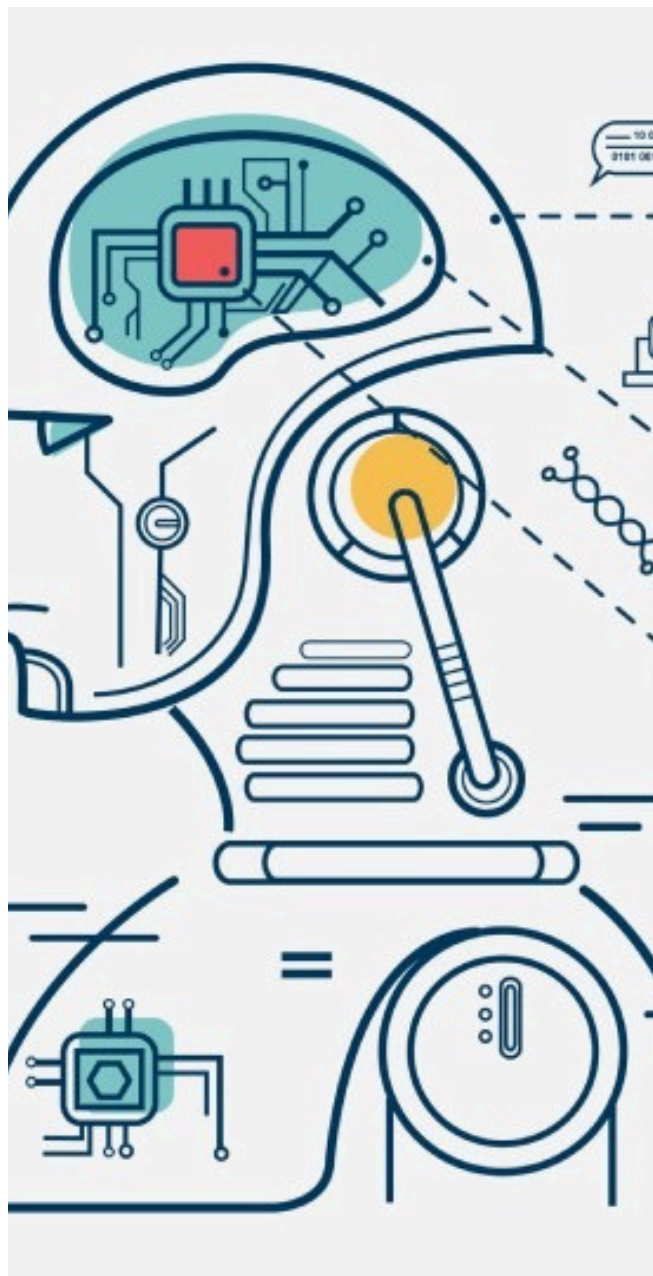
**06. APPROCHE UTILISÉE POUR LA
RÉSOLUTION DU PROBLÈME**

07. ÉVALUATION DES MODÈLES

08. CONCLUSION

01. INTRODUCTION

Ce rapport présente notre projet utilisant l'apprentissage automatique pour résoudre des problèmes réels. Nous explorons la source des données, soulignant l'importance de leur qualité. Ensuite, nous détaillons le prétraitement des données et notre approche algorithmique. Nous évaluons ensuite les performances de notre modèle. En conclusion, nous résumons nos principales conclusions et soulignons les implications et les pistes de recherche futures.

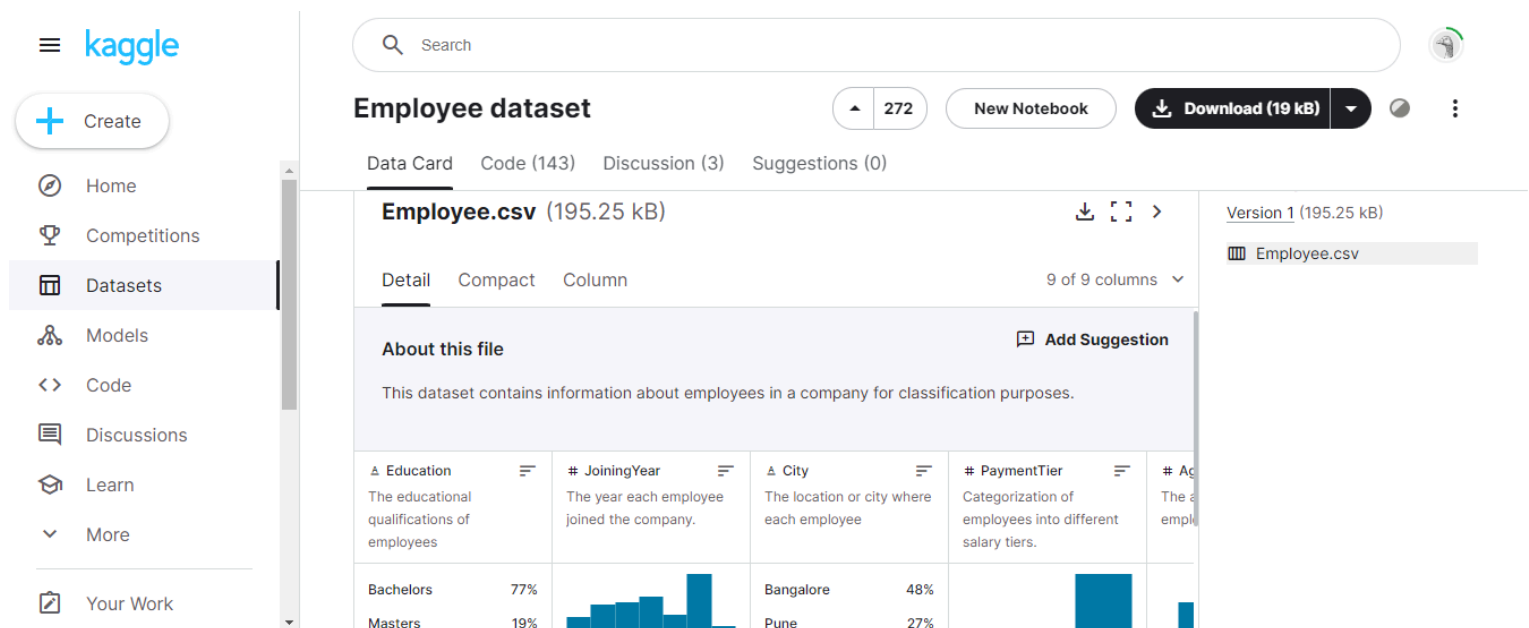


02. OBJECTIF DU PROJET

L'objectif de ce projet est de développer un modèle prédictif permettant d'analyser un dataset "Employee.csv". Ce modèle devra être capable de fournir des prédictions précises en se basant sur les caractéristiques disponibles dans le jeu de données.

03. SOURCE DES DONNÉES

Les données utilisées dans ce projet proviennent de la source des données: Kaggle. Le jeu de données "Employee.csv" contient des informations des employées.

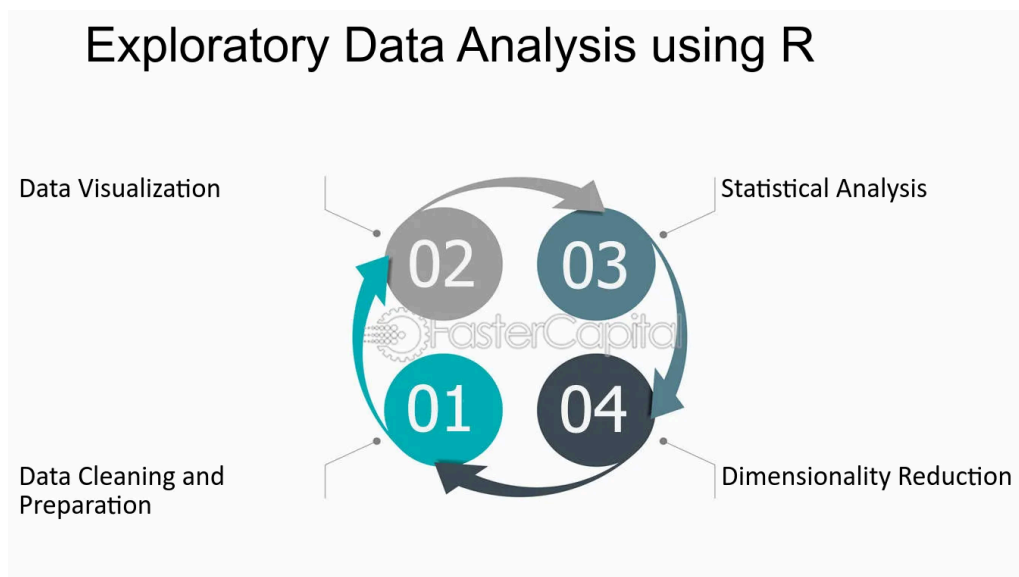


The screenshot shows the Kaggle interface for the "Employee dataset". The left sidebar contains navigation links: Home, Competitions, Datasets (selected), Models, Code, Discussions, Learn, More, and Your Work. The main content area displays the dataset details for "Employee.csv" (195.25 kB). It includes a search bar, a "New Notebook" button, and a "Download (19 kB)" button. Below the dataset name, there are tabs for "Data Card", "Code (143)", "Discussion (3)", and "Suggestions (0)". The "Data Card" tab is active, showing a "Detail" view. The "About this file" section states: "This dataset contains information about employees in a company for classification purposes." Below this, there are five columns of data with associated bar charts:

Education	# JoiningYear	City	# PaymentTier	# Age
Bachelors	77%	Bangalore	48%	
Masters	19%	Pune	27%	

04. ANALYSE EXPLORATOIRE DES DONNÉES

Avant de procéder à la modélisation, une analyse exploratoire des données a été effectuée pour mieux comprendre la structure et les caractéristiques du jeu de données. Cette analyse a inclus l'étude des distributions des variables, l'identification des valeurs aberrantes et des valeurs manquantes, ainsi que l'exploration des relations entre les variables.



05. DESCRIPTION DE LA PHASE DE PRE-PROCESSING DES DONNÉES

La phase de prétraitement des données a été cruciale pour garantir la qualité et la pertinence des données utilisées pour la modélisation. Les étapes de prétraitement ont inclus :

- Chargement de dataset "Employee.csv" :

Chargement du dataset

Entrée [20]:

```
import pandas as pd
data = pd.read_csv('Employee.csv')
data
```

Out[20]:

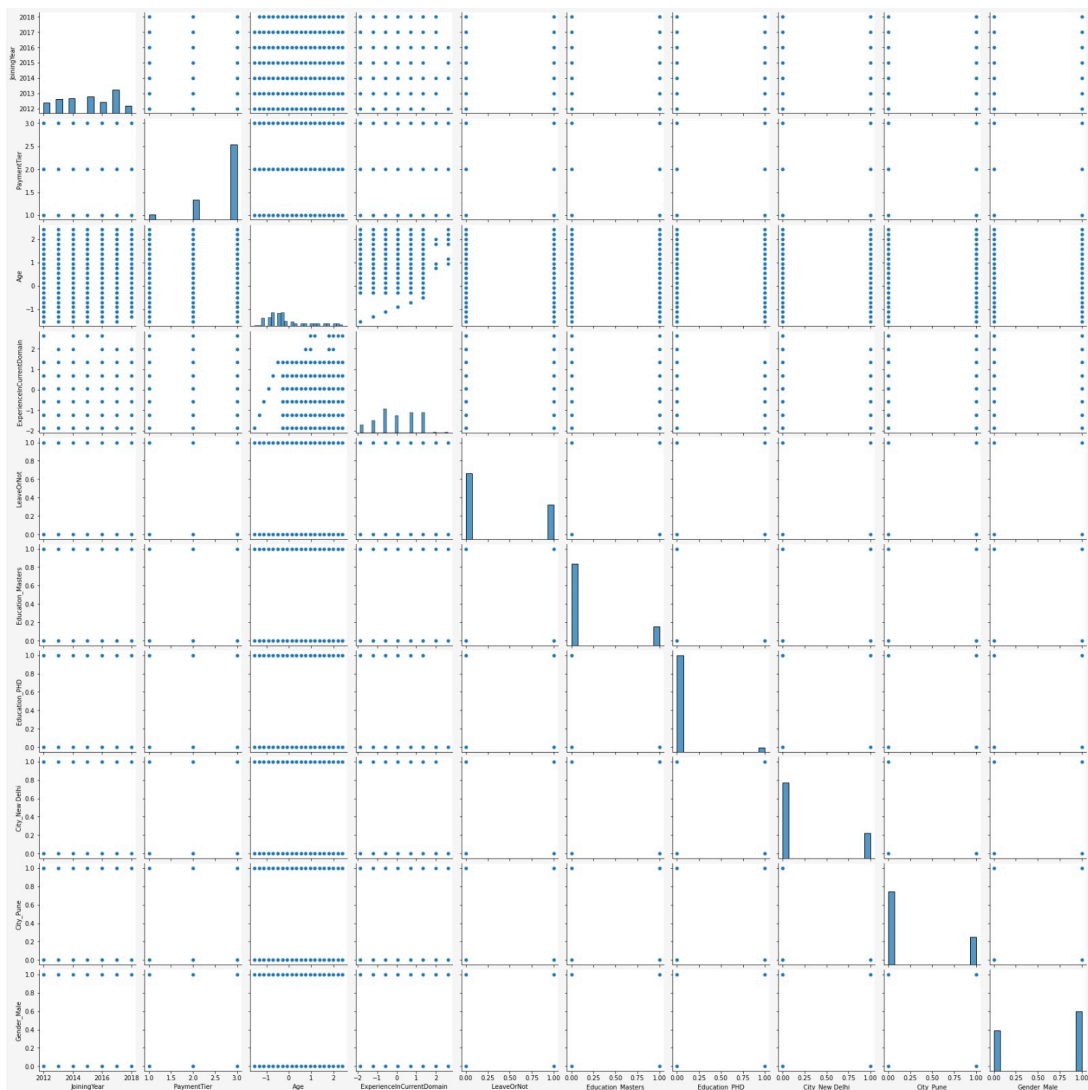
	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	LeaveOrNot
0	Bachelors	2017	Bangalore	3	34	Male	No	0	0
1	Bachelors	2013	Pune	1	28	Female	No	3	1
2	Bachelors	2014	New Delhi	3	38	Female	No	2	0
3	Masters	2016	Bangalore	3	27	Male	No	5	1
4	Masters	2017	Pune	3	24	Male	Yes	2	1
...
4648	Bachelors	2013	Bangalore	3	26	Female	No	4	0
4649	Masters	2013	Pune	2	37	Male	No	2	1
4650	Masters	2018	New Delhi	3	27	Male	No	5	1
4651	Bachelors	2012	Bangalore	3	30	Male	Yes	2	0
4652	Bachelors	2015	Bangalore	3	33	Male	Yes	4	0

- Visualisation des données :

Entrée [29]:

```
# Visualisation des données
import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(data)
plt.show()
```



- **Prétraitement du dataset :**

Pretraitement du Dataset ¶

Entrée [5]: `#verification de valeurs nulles`
`data.isnull().sum()`

Out[5]:

Education	0
JoiningYear	0
City	0
PaymentTier	0
Age	0
Gender	0
EverBenched	0
ExperienceInCurrentDomain	0
LeaveOrNot	0
dtype:	int64

Entrée [21]: `# Encodage des variables catégorielles`
`from sklearn.preprocessing import OneHotEncoder`
`encoder = OneHotEncoder(sparse=False, drop='first')`
`encoded_cols = pd.DataFrame(encoder.fit_transform(data[['Education', 'City', 'Gender']]), columns=encoder.get_feature_names())`
`data.drop(['Education', 'City', 'Gender', 'EverBenched'], axis=1, inplace=True)`
`data = pd.concat([data, encoded_cols], axis=1)`
`data`

Out[21]:

	JoiningYear	PaymentTier	Age	ExperienceInCurrentDomain	LeaveOrNot	Education_Masters	Education_PHD	City_New Delhi	City_Pune	Gender_Male
0	2017	3	34		0	0	0.0	0.0	0.0	1.0
1	2013	1	28		3	1	0.0	0.0	0.0	0.0
2	2014	3	38		2	0	0.0	0.0	1.0	0.0
3	2016	3	27		5	1	1.0	0.0	0.0	1.0
4	2017	3	24		2	1	1.0	0.0	0.0	1.0
...
4648	2013	3	26		4	0	0.0	0.0	0.0	0.0
4649	2013	2	37		2	1	1.0	0.0	0.0	1.0
4650	2018	3	27		5	1	1.0	0.0	1.0	1.0
4651	2012	3	30		2	0	0.0	0.0	0.0	1.0
4652	2015	3	33		4	0	0.0	0.0	0.0	1.0

4653 rows x 10 columns

Entrée [22]: `# Mise à l'échelle des caractéristiques numériques`
`from sklearn.preprocessing import StandardScaler`
`scaler = StandardScaler()`
`data[['Age', 'ExperienceInCurrentDomain']] = scaler.fit_transform(data[['Age', 'ExperienceInCurrentDomain']])`

NB : La mise à l'échelle est réalisée afin de standardiser les valeurs de ces caractéristiques, les rendant ainsi comparables et facilitant le processus d'apprentissage automatique.

Entrée [23]: `# Analyse exploratoire des données (EDA)`
`# Division des données`
`from sklearn.model_selection import train_test_split`
`X = data.drop('LeaveOrNot', axis=1)`
`y = data['LeaveOrNot']`
`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)`

06. APPROCHE UTILISÉE POUR LA RÉOLUTION DU PROBLÈME

- **Initialisation et Entrainement des modèles :**

La regression logistique :

Entrée [32]: `#modele1: regression logistique`
`from sklearn.linear_model import LogisticRegression`
`logistic_regression_model = LogisticRegression()`
`logistic_regression_model.fit(X_train, y_train)`

Out[32]: LogisticRegression()

Arbre de décision :

```
Entrée [33]: #modele2: DecisionTree
from sklearn.tree import DecisionTreeClassifier
decision_tree_model = DecisionTreeClassifier()
decision_tree_model.fit(X_train, y_train)

Out[33]: DecisionTreeClassifier()
```

Random forest :

```
Entrée [34]: #modele: Random forest
from sklearn.ensemble import RandomForestClassifier

random_forest_model = RandomForestClassifier()
random_forest_model.fit(X_train, y_train)

Out[34]: RandomForestClassifier()
```

07. ÉVALUATION DES MODÈLES

- **Evaluation des Algorithmes :**

Évaluation des algorithmes :

```
Entrée [35]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Prédiction sur l'ensemble de test
logistic_regression_pred = logistic_regression_model.predict(X_test)
decision_tree_pred = decision_tree_model.predict(X_test)
random_forest_pred = random_forest_model.predict(X_test)

Entrée [39]: # Calcul des métriques
logistic_regression_accuracy = accuracy_score(y_test, logistic_regression_pred)
decision_tree_accuracy = accuracy_score(y_test, decision_tree_pred)
random_forest_accuracy = accuracy_score(y_test, random_forest_pred)

logistic_regression_precision = precision_score(y_test, logistic_regression_pred)
decision_tree_precision = precision_score(y_test, decision_tree_pred)
random_forest_precision = precision_score(y_test, random_forest_pred)
print("la precision du modele regression lineaire",logistic_regression_precision)
print("la precision du modele Arbre de decision",decision_tree_precision)
print("la precision du modele de foret d'arbre de decision",random_forest_precision)

la precision du modele regression lineaire 0.7106598984771574
la precision du modele Arbre de decision 0.7728813559322034
la precision du modele de foret d'arbre de decision 0.823321554770318
```

- **Autre Approche :**

Comparaison avec une autre approche :

```
Entrée [54]: from sklearn.svm import SVC
# Initialisation et entraînement du modèle SVM
svm_model = SVC()
svm_model.fit(X_train, y_train)

Out[54]: SVC()

Entrée [55]: # Prédiction sur l'ensemble de test
svm_pred = svm_model.predict(X_test)

Entrée [61]: # Calcul des métriques pour le modèle SVM
svm_accuracy = accuracy_score(y_test, svm_pred)
svm_precision = precision_score(y_test, svm_pred)
svm_recall = recall_score(y_test, svm_pred)
svm_f1_score = f1_score(y_test, svm_pred)

C:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```


- **Comparaison des modèles :**

```
Entrée [64]: # Comparaison avec Les autres modèles
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print("Logistic Regression Accuracy:", logistic_regression_accuracy)
print("Decision Tree Accuracy:", decision_tree_accuracy)
print("Random Forest Accuracy:", random_forest_accuracy)
print("SVM Accuracy:", svm_accuracy)

Logistic Regression Accuracy: 0.7443609022556391
Decision Tree Accuracy: 0.8281417830290011
Random Forest Accuracy: 0.8517722878625135
SVM Accuracy: 0.6552094522019334
```

08. CONCLUSION

En synthèse, ce projet démontre la supériorité de l'algorithme Random Forest pour traiter notre dataset , après une analyse exploratoire et un prétraitement minutieux des données. Ces résultats soulignent l'importance cruciale du choix approprié de l'algorithme dans le processus d'apprentissage automatique.

conclusion:

Dans les 2 cas, on a constaté que le modèle Random Forest est le modèle le plus performant.