

REPUBLIQUE DEMOCRATIQUE DU CONGO
ENSEIGNEMENT SUPERIEUR ET UNIVERSITAIRE
INSTITUT SUPERIEUR DE STATISTIQUE
ECOLE DOCTORALE DE STATISTIQUE ET INFORMATIQUE
(EDSI)
B. P. 2471
LUBUMBASHI



TRAVAIL PRATIQUE DU COURS DE DATA MINING

Titulaire du Cours : Prof. MBAKI LUZAYISU Efrem

Présenté par :

- PINTO KATENDE Jonathan
- MUSONDA Balthazar
- KHANG MATE ZULBAL
- MUKWIYO MUKALO Patrick
- MPUNDU MWANA NGOY

2024

Objectif

1. Préparation des Données :

- Importer les données et vérifier s'il y a des valeurs manquantes.
- Nettoyer les données en traitant les valeurs manquantes et en codant les variables catégorielles si nécessaire.

2. Analyse Exploratoire :

- Calculer les statistiques descriptives pour chaque variable et visualiser la répartition des variables
 - Statistique univariée
 - Statistique Bivariée
 - Etude des liaisons et corrélations

3. Modélisation :

- Construire un modèle pour prédire les différentes Catégories en fonction des autres variables
- Évaluer la performance du modèle.

4. Visualisation et Interprétation :

- Présenter les résultats de manière visuelle (graphes, diagrammes).
- Interpréter les résultats et proposer des recommandations

ANALYSE

Dans le cadre de cette analyse, nous utilisons Python 3.13 comme langage de programmation.

Voici les librairies qui seront utilisés :

Préparation des données

Installation des packages

```
%pip install pandas numpy # Manipulation excel
%pip install --upgrade pip # Gestionnaire des paquets
%pip install polars # Meilleur alternatif de Pandas
%pip install matplotlib plotly # Pour la visualisation
%pip install fastexcel # Dépendance de Polars pour le fichier .xlsx
%pip install pyarrow # Dépendance de Polars pour le fichier .xlsx
%pip install openpyxl # Dépendance de Polars pour le fichier .xlsx
%pip install seaborn # Pour la visualisation
%pip install scikit-learn # Pour le machine learning
%pip install statsmodels # Pour les statistiques
%pip install plotnine # Pour la visualisation
%pip install tabulate colorama # Pour la coloration
```

Chargement des packages

```
import pandas as pd # type: ignore
import numpy as np
import polars as pl
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from plotnine import * # type: ignore # noqa: F403
import scipy.stats as ss
from tabulate import tabulate
from colorama import Fore, Style, Back
```

Importation des données

```
data_pl = pl.read_excel("data.xlsx") # Pour la manipulation avec Polars
data_pd = pd.read_excel("data.xlsx") # Pour la manipulation avec Pandas
ctx = pl.SQLContext(data=data_pl, eager=True) # Ecriture en SQL
```

Illustration des données

```
print(data_pl)
```

shape: (5_744, 19)								
CATEGORIE	GRADE	DERNIER AVANCEMENT EN	SALAIRE PERS	...	ENERGIE	IFT	IDP	IPM
---	---	AVANCEMENT EN	---		---	---	---	---
str	str	GRADE	f64		i64	f64	f64	i64
---	---	date						
M&C	11	1998-07-01	47865.44225	...	12045	97837.51	126115.18	348500
M&C	11	1998-09-01	47652.18334	...	12045	97837.51	126115.18	348500
M&C	11	2000-08-01	47348.71096	...	12045	97837.51	126115.18	348500
M&C	11	2000-10-01	47115.68266	...	12045	97837.51	126115.18	348500
M&C	11	2000-12-01	47344.56758	...	12045	97837.51	126115.18	348500
...
EXE	71	2023-11-01	4920.44077	...	5940	67568.84	46635.93	342500
EXE	71	2023-11-01	4920.44077	...	5940	67568.84	46635.93	342500
EXE	71	2023-11-01	4920.44077	...	5940	67568.84	46635.93	342500
EXE	61	2023-11-01	4913.62077	...	5940	67278.84	46413.51	342500
EXE	61	2023-11-01	4913.62077	...	5940	67278.84	46413.51	342500

Explication des variables

- **CATEGORIE** : Code ou description de la catégorie professionnelle du personnel
- **GRADE** : Niveau de grade de l'employé au sein de la catégorie
- **DERNIER AVANCEMENT EN GRADE** : Date du dernier avancement en grade de l'employé.
- **SALAIRE PERS** : Salaire personnel ou de base de l'employé,
- **CITE** : Code de la cité ou de la localité où réside l'employé
- **SEXE** : Sexe de l'employé
- **ETATCIVIL** : Code de l'état civil de l'employé.
- **INDEPOUSE** : Indicateur pour savoir si l'employé a un conjoint à charge. Par exemple, "1" pour oui et "0" pour non.
- **ENFANT** : Nombre d'enfants à charge de l'employé
- **COTATION** : Code de cotation ou score lié au poste ou à la performance de l'employé.
- **DATENAISSA** : Date de naissance de l'employé.
- **DATEENGAGE** : Date d'engagement de l'employé dans l'entreprise (date d'embauche).
- **TRP** : Indemnité de transport, ou allocation liée aux frais de déplacement.
- **LGT** : Allocation logement de l'employé.
- **SSANTE** : Cotisation ou indemnité pour la sécurité sociale ou assurance santé.
- **ENERGIE** : Allocation ou indemnité énergétique, qui pourrait couvrir des frais d'électricité, de chauffage, etc.
- **IFT** : Allocation ou indemnité pour frais de téléphone.
- **IDP** : Indemnité pour dépendants, ou allocation pour les personnes à charge.
- **IPM** : Indemnité pour prestations médicales, couvrant potentiellement les frais de santé ou d'hospitalisation.

Information sur les variables

```
print(data_pd.info())
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5744 entries, 0 to 5743
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype
---  -
 0   CATEGORIE                            5744 non-null   object
 1   GRADE                                5744 non-null   int64
 2   DERNIER AVACEMENT EN GRADE          5742 non-null   datetime64[ns]
 3   SALAIRE PERS                         5744 non-null   float64
 4   CITE                                  5744 non-null   object
 5   SEXE                                 5744 non-null   object
 6   ETATCIVIL                           5678 non-null   object
 7   INDEPOUSE                           5534 non-null   float64
 8   ENFANT                               5744 non-null   int64
 9   COTATION                             2536 non-null   float64
10  DATENAISSA                           5744 non-null   datetime64[ns]
11  DATEENGAGE                           5744 non-null   datetime64[ns]
12  TRP                                   5744 non-null   float64
13  LGT                                   5744 non-null   float64
14  SSANTE                               5744 non-null   float64
15  ENERGIE                              5744 non-null   int64
16  IFT                                   5744 non-null   float64
17  IDP                                   5744 non-null   float64
18  IPM                                   5744 non-null   int64
dtypes: datetime64[ns](3), float64(8), int64(4), object(4)
memory usage: 852.8+ KB

```

Nous constatons que l'analyse est faite sur 5744 individus.

Sur toutes les variables, nous remarquons que la variable **COTATION** a près de 50% des valeurs manquantes. En plus, cette variable cotation n'intéresse que ceux de la catégorie M&C. D'où, elle sera supprimée lors de la modélisation.

Création des variables artificielles

En se fiant sur les données que nous disposons, nous constatons que certaines variables artificielles peuvent être créées :

- **L'âge à l'engagement** : la différence entre la date d'engagement et la date de Naissance.
- Après que l'âge à l'engagement soit créé, une autre variable peut être créée sur la base de celle-ci : **tranche d'âge**.

Pour la création des classes, il existe des critères et des méthodes existant du point de vue statistique, entre autres :

- [Méthode de LIORZOU](#)
- [Méthode de Stem and Leaf](#)

Dans le cadre de cette étude, nous allons nous limiter par une méthode manuelle d'amplitude 5, de la valeur minimale de cette variable jusqu'à sa valeur maximale.

Création de variable âge

```
result = data_pl.select(
    pl.col('DATENAISSA'),
    pl.col('DATEENGAGE'),
    ((pl.col("DATEENGAGE") - pl.col('DATENAISSA')).cast(pl.Duration) / pl.
duration(days=365.25)).cast(pl.Int32).alias("Age à l'engagement")
)

print(result)
```

Voici comment se présente cette nouvelle variable :

shape: (5_744, 3)

DATENAISSA	DATEENGAGE	Age à l'engagement
---	---	---
date	date	i32
1950-06-02	1968-09-01	18
1954-01-01	1980-04-14	26
1960-02-02	1987-12-14	27
1953-01-15	1977-08-15	24
1963-05-24	1987-12-14	24
...
1990-05-15	2023-11-01	33
2000-05-05	2023-11-01	23
1997-04-12	2023-11-01	26
1988-02-06	2023-11-01	35
1991-04-18	2023-11-01	32

Création de variable Tranche d'âge

```
data_pd = data_pl_new.to_pandas()
```

```
bins = [13, 18, 23, 28, 33, 38, 43, 48, 53, 56]
labels = ['13-18', '18-23', '23-28', '28-33', '33-38', '38-43', '43-48',
'48-53', 'Plus de 53']
data_pd['tranche_age_engagement'] = pd.cut(data_pd['age_engagement'],
bins=bins, labels=labels, right=False)

print(data_pd[['age_engagement', 'tranche_age_engagement']])
```

	age_engagement	tranche_age_engagement
0	18	18-23
1	26	23-28
2	27	23-28
3	24	23-28
4	24	23-28
...
5739	33	33-38
5740	23	23-28
5741	26	23-28
5742	35	33-38
5743	32	28-33

[5744 rows x 2 columns]

Analyse exploratoire

Dans le cadre de cette analyse, nous considérons la variable CATEGORIE comme variable cible.

- **Analyse univariée**

Variables Quantitatives

```
print(data_pl_new.select(
    pl.col('age_engagement'),
    pl.col('SALAIRE PERS'),
    pl.col('ENFANT')
).describe())
```

shape: (9, 4)

statistic	age_engagement	SALAIRE PERS	ENFANT
---	---	---	---
str	f64	f64	f64
count	5744.0	5744.0	5744.0
null_count	0.0	0.0	0.0
mean	25.584436	25926.253361	2.184018
std	5.77386	23616.887196	2.524423
min	13.0	4705.46265	0.0
25%	21.0	5180.02077	0.0
50%	24.0	5352.96077	1.0
75%	29.0	49967.96363	4.0
max	55.0	74226.169	16.0

Les résultats de l'analyse montrent que :

- Le salaire moyen de cette population est de 25926
- Tandis que près de 50% des individus de cette population ont un salaire de moins de 5180.
- Se basant sur le nombre d'enfants, en moyenne chaque agent a 2 enfants.
- Près de 50% des agents ne dépasse pas un agent.
- Mais dans cette population, il y a au moins un agent qui a 16 enfants.
- Se référant sur l'âge à l'engagement, on constate que l'âge moyen à l'engagement est de 25 ans.
- Près de 75% d'agents avait moins de 29 ans à l'engagement.

Nous constatons également des incohérences sur plusieurs résultats, mais il y a une raison à cela qui pourrait être déceler lors de l'analyse bivariée.

Mais également il y a des résultats qui violent les recommandations de l'Etat Congolais :

Nous voyons que l'âge à l'engagement minimal est de 13 ans, or la loi N°16.010 du 15 Juillet 2016 portant code de la famille stipule que « Tout enfant né en République Démocratique du Congo peut être engagé ou maintenue en service pour l'exécution des travaux légers et salubres, que celui qui 16 à moins de 18 ans ». D'où l'âge minimal pour le travail est de 16 ans.

```
data_pl_new.filter(pl.col('age_engagement') < 16).height  
5
```

On sait voir que l'entreprise a engagé 5 personnes qui ont moins de 16, ce qui est illégal.

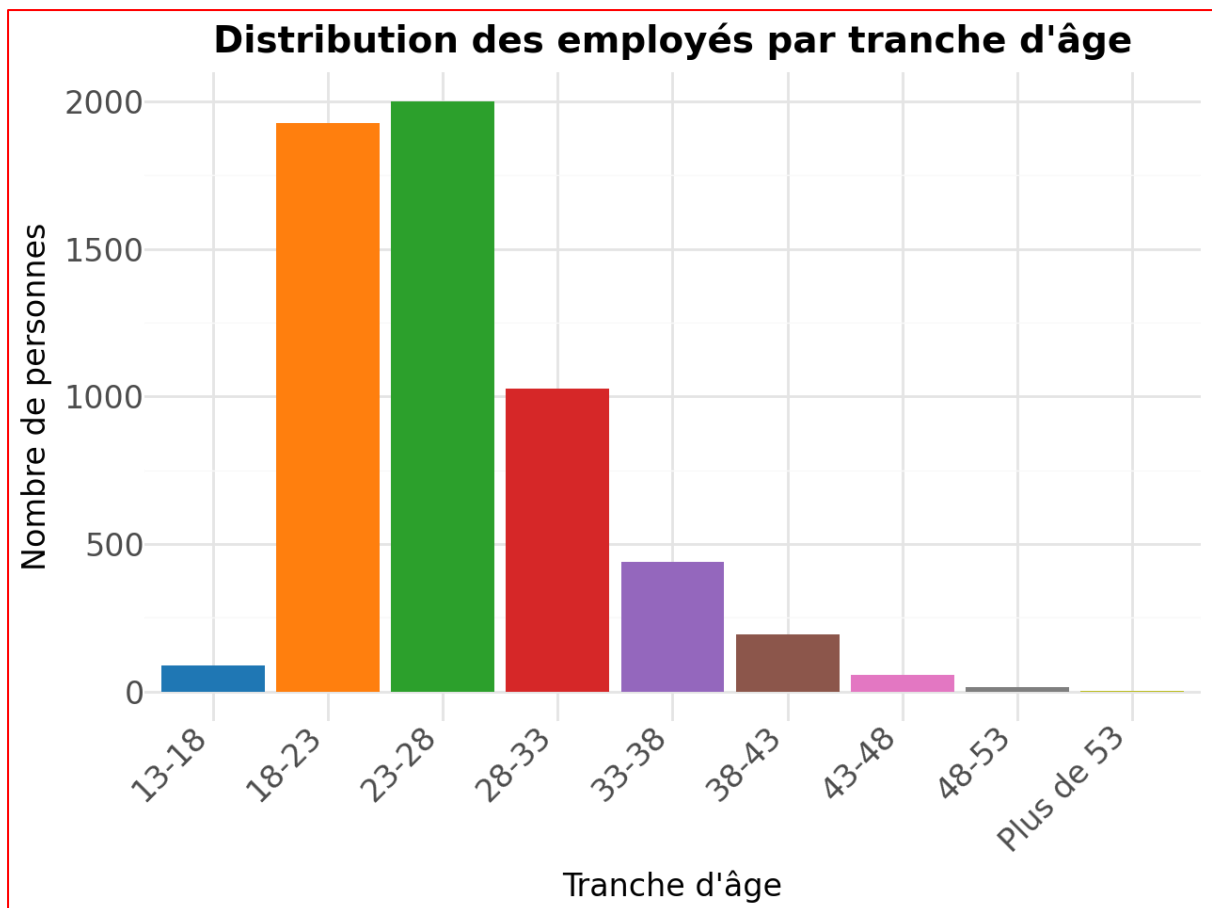
Variables Qualitatives

1. Tranche d'âge

Conversion du DataFrame Polars à Pandas

```
tranche_counts_pd = tranche_counts_pl.to_pandas()
```

```
plot = (  
    ggplot(tranche_counts_pd, aes(x='tranche_age_engagement',  
y='effectif', fill='tranche_age_engagement'))  
    + geom_bar(stat='identity', show_legend=False) # On désactive la  
    légende ici pour simplifier  
    + scale_fill_manual(values=[  
        '#1f77b4', # bleu  
        '#ff7f0e', # orange  
        '#2ca02c', # vert  
        '#d62728', # rouge  
        '#9467bd', # violet  
        '#8c564b', # brun  
        '#e377c2', # rose  
        '#7f7f7f', # gris  
        '#bcbd22'  # jaune-vert  
    ])  
    + labs(  
        title='Distribution des employés par tranche d\'âge',  
        x='Tranche d\'âge',  
        y='Nombre de personnes'  
    )  
    + theme_minimal()  
    + theme(  
        axis_text_x=element_text(rotation=45, hjust=1),  
        text=element_text(size=12),  
        plot_title=element_text(weight='bold', size=14)  
    )  
)  
  
plot
```

Les résultats montrent que :

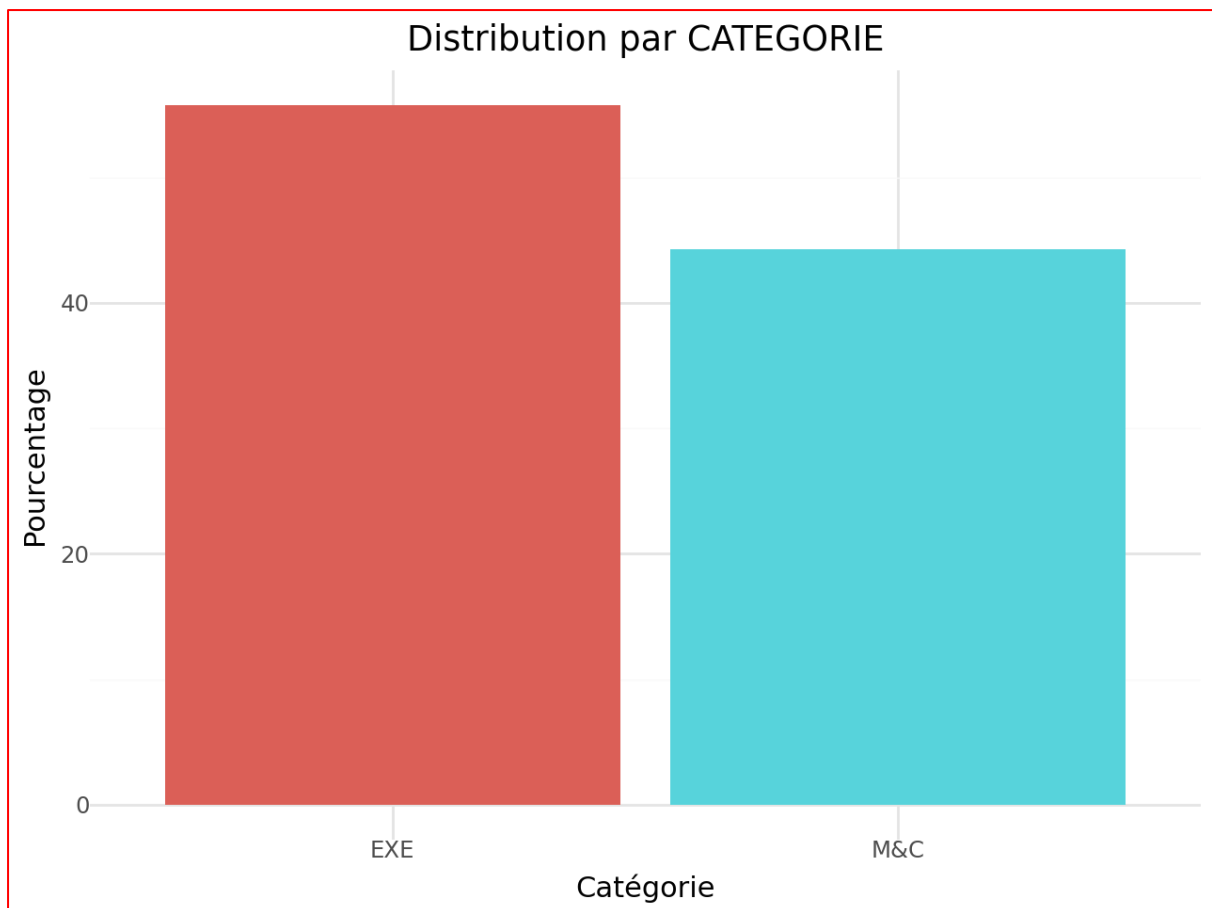
- La tranche d'âge majoritaire pour les engagements est de 18 à 38 ans
- Mais avec des engagements aussi entre 13 et 18 ans

2. Catégorie

```
categorie_counts = data_pd[['CATEGORIE']].value_counts().reset_index()
categorie_counts.columns = ['CATEGORIE', 'count']
```

```
total_count = categorie_counts['count'].sum()
categorie_counts['percentage'] = (categorie_counts['count'] /
total_count) * 100
```

```
(ggplot(categorie_counts, aes(x='CATEGORIE', y='percentage',
fill='CATEGORIE'))
+ geom_bar(stat='identity', show_legend=False)
+ labs(
    title='Distribution par CATEGORIE',
    x='Catégorie',
    y='Pourcentage'
)
+ theme_minimal()
)
```



Ce graphique montre que pour cette société, la majorité des employés sont dans la catégorie EXE, à hauteur de plus de 55% des agents se trouvant dans cette catégorie.

3. Indepouse (Indicateur pour savoir si l'employé a un conjoint à charge)

```
indepouse_counts = data_pd[['INDEPOUSE']].value_counts().reset_index()
indepouse_counts.columns = ['INDEPOUSE', 'count']
```

Calcul des pourcentages

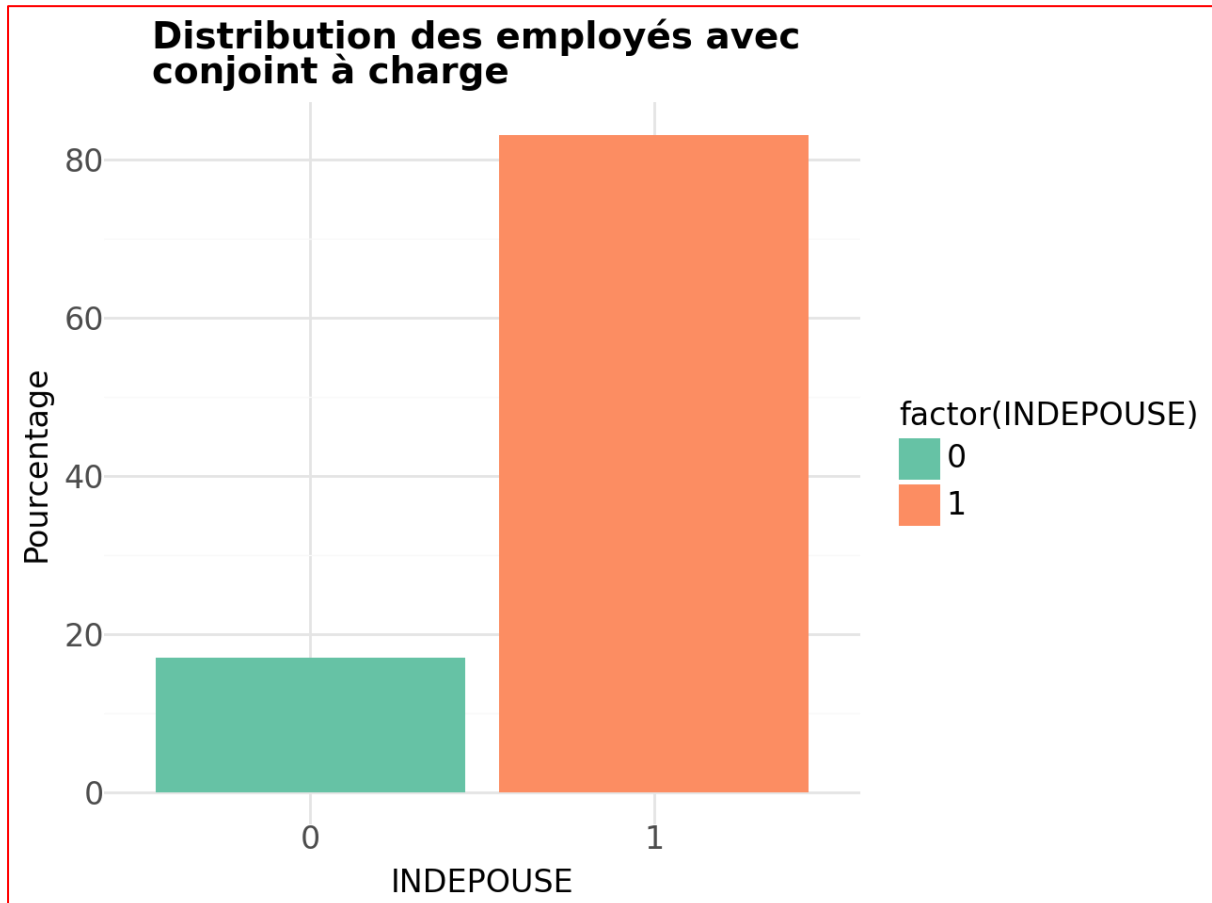
```
total_count = indepouse_counts['count'].sum()
indepouse_counts['percentage'] = (indepouse_counts['count'] /
total_count) * 100
```

```
(
    ggplot(indepouse_counts, aes(x='INDEPOUSE', y='percentage',
fill='factor(INDEPOUSE)'))
    + geom_bar(stat='identity')
    + scale_fill_manual(values=['#66c2a5', '#fc8d62']) # Palette de
couleurs attrayantes
    + labs(
        title='Distribution des valeurs de INDEPOUSE en pourcentage',
        x='INDEPOUSE',
        y='Pourcentage'
    )
    + theme_minimal()
```

```

+ theme(
  axis_text_x=element_text(rotation=0, hjust=0.5),
  text=element_text(size=12),
  plot_title=element_text(weight='bold', size=14)
)
)

```



Ce graphique montre que plus de 80% des employés ont un conjoint à charge.

4. Sexe

```

# Conversion des données en DataFrame
sexe_counts = data_pd['SEXE'].value_counts().reset_index()
sexe_counts.columns = ['SEXE', 'count']

# Calcul des pourcentages
total_count = sexe_counts['count'].sum()
sexe_counts['percentage'] = (sexe_counts['count'] / total_count) * 100

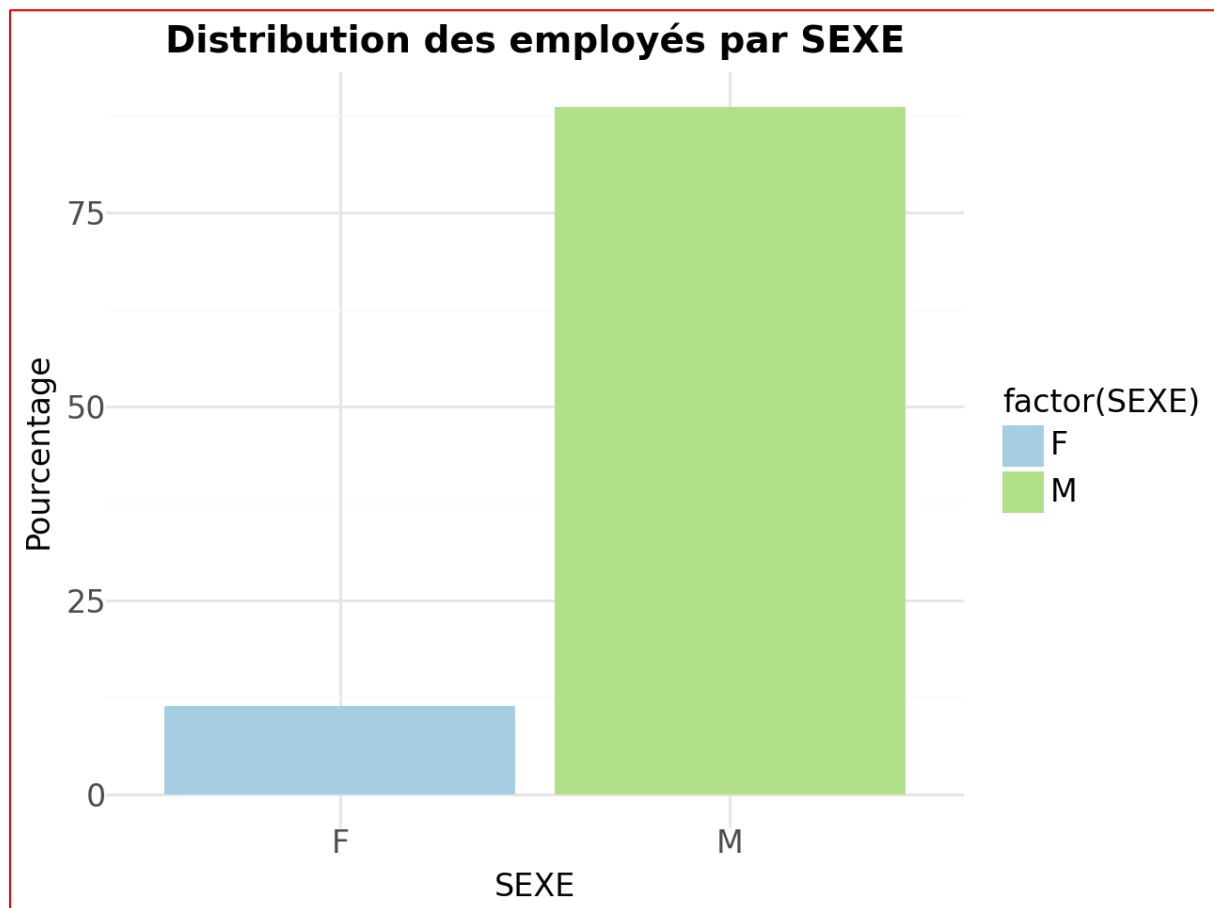
(
  ggplot(sexe_counts, aes(x='SEXE', y='percentage',
fill='factor(SEXE)'))
  + geom_bar(stat='identity')
  + scale_fill_manual(values=['#a6cee3', '#b2df8a']) # Palette de
couleurs attrayantes

```

```

+ labs(
  title='Distribution des employés par SEXE',
  x='SEX',
  y='Pourcentage'
)
+ theme_minimal()
+ theme(
  axis_text_x=element_text(rotation=0, hjust=0.5),
  text=element_text(size=12),
  plot_title=element_text(weight='bold', size=14)
)
)

```



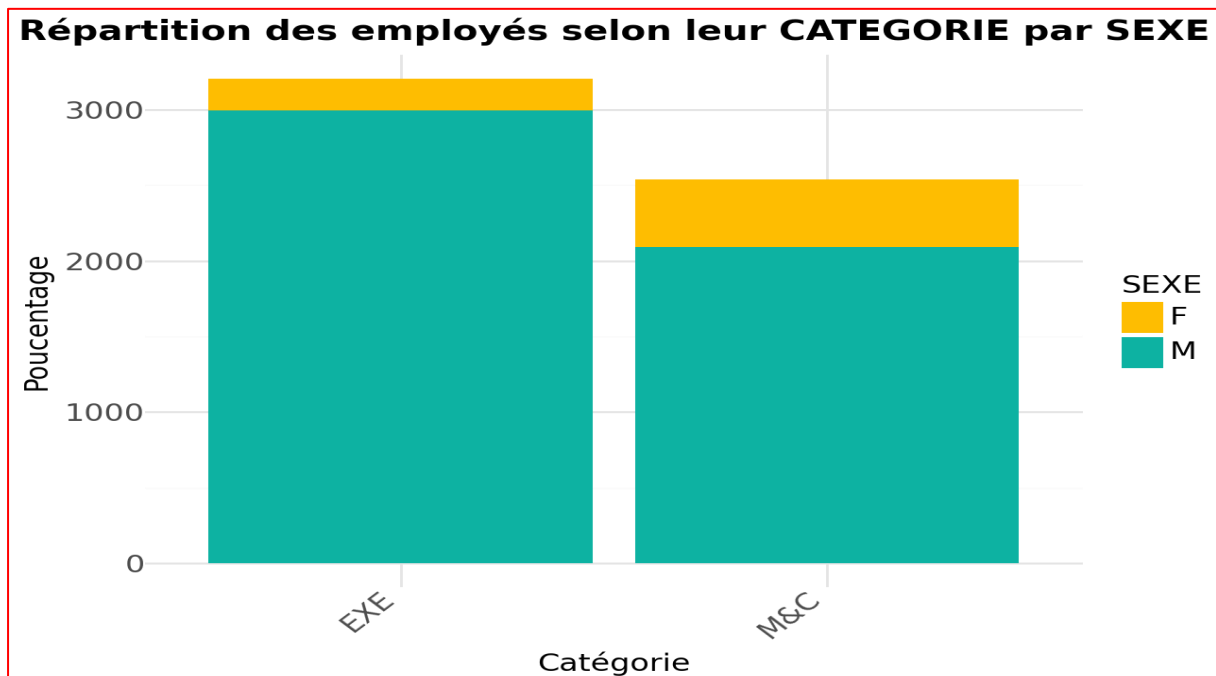
Se basant sur le sexe, plus de 85% des employés sont des hommes. Ce qui montre une prédominance des hommes dans cette entreprise.

- Analyse bivariable

Catégorie vs Sexe

```
contingency_table = pd.crosstab(data_pd['CATEGORIE'],
data_pd['SEXE']).reset_index()
contingency_table = pd.melt(contingency_table, id_vars='CATEGORIE',
var_name='SEXE', value_name='count')

(
    ggplot(contingency_table, aes(x='CATEGORIE', y='count', fill='SEXE'))
    + geom_bar(stat='identity', position='stack')
    + scale_fill_manual(values=['#febd01', '#0db2a2'])
    + labs(
        title='Répartition des employés selon leur CATEGORIE par SEXE',
        x='Catégorie',
        y='Effectif',
        fill='SEXE'
    )
    + theme_minimal()
    + theme(
        axis_text_x=element_text(rotation=45, hjust=1),
        text=element_text(size=12),
        plot_title=element_text(weight='bold', size=14)
    )
)
```



Nous remarquons que dans toutes les catégories confondues, les hommes sont prédominant en termes d'effectif.

Categorie vs Salaire

```
statistiques_salaire = data_pl.groupby("CATEGORIE").agg([
    pl.col("SALAIRE PERS").count().alias("count"),
    pl.col("SALAIRE PERS").mean().alias("mean"),
    pl.col("SALAIRE PERS").std().alias("std"),
    pl.col("SALAIRE PERS").min().alias("min"),
    pl.col("SALAIRE PERS").quantile(0.25).alias("25%"),
    pl.col("SALAIRE PERS").median().alias("50%"),
    pl.col("SALAIRE PERS").quantile(0.75).alias("75%"),
    pl.col("SALAIRE PERS").max().alias("max")
])

print(statistiques_salaire)
```

shape: (2, 9)

CATEGORIE	count	mean	std	...	25%	50%	75%	max
---	---	---	---	---	---	---	---	---
str	u32	f64	f64		f64	f64	f64	f64
M&C	2541	52068.2511	5919.22680	...	46799.6294	50670.688	54851.628	74226.169
EXE	3203	5187.31603	130.651722	...	5117.18077	5197.5207	5278.8207	5522.9607

De ce résultat, on peut voir :

- Le salaire moyen de la catégorie M&C est de 52068.25F, tandis que dans la catégorie EXE le salaire moyen est de 5187.31F ; ce qui largement inférieur à celle de la catégorie M&C.
- Près de 50% des employés dans la catégorie M&C ne dépassent pas le salaire de 50671 ; de même pour ceux qui sont dans la catégorie EXE, ils ne dépassent pas le salaire 5198.

Categorie vs Nombre d'enfant

```
result = (
    data_pl_new.groupby('CATEGORIE')
    .agg([
        pl.col('ENFANT').mean().alias('mean'),
        pl.col('ENFANT').std().alias('std'),
        pl.col('ENFANT').min().alias('min'),
        pl.col('ENFANT').quantile(0.25).alias('25%'),
        pl.col('ENFANT').median().alias('50%'), # Médiane
        pl.col('ENFANT').quantile(0.75).alias('75%'),
        pl.col('ENFANT').max().alias('max'),
        pl.col('ENFANT').sum().alias('sum'),
        pl.col('ENFANT').count().alias('n')
    ])
)

print(result)
```

shape: (2, 10)

CATEGORIE	mean	std	min	...	75%	max	sum	n
---	---	---	---		---	---	---	---
str	f64	f64	i64		f64	i64	i64	u32
EXE	2.451452	2.665344	0	...	4.0	16	7852	3203
M&C	1.846911	2.291155	0	...	3.0	14	4693	2541

De ce tableau, il résulte comme informations suivantes :

- Dans la catégorie EXE, chaque employé a aux alentours de 3 enfants en moyenne, tandis que dans la catégorie M&C la tendance est autour de deux enfants en moyenne.
- Près de 75% d'employés dans la catégorie EXE ne dépasse pas 4 enfants, avec une exception d'au moins une personne qui a 16 enfants.
- Mais si nous prenons la Catégorie M&C, près de 50% d'employé ont moins d'un enfant ; et près de 75% d'agent ne dépense pas 3 enfants. Il y aussi une exception d'au moins un agent qui a 14 enfants.

De cette analyse, on peut donc dire, les employés qui ont un salaire faible, ont tendance a avoir plus d'enfants comparativement à ceux qui ont un salaire assez élevé.

Categorie vs INDEPOUSE

Création de la table de contingence avec group_by et pivot

```
contingency_table = (  
    data_pl_new.group_by(['CATEGORIE', 'INDEPOUSE'])  
    .agg(pl.count().alias('count'))  
    .pivot(  
        values='count',  
        index='CATEGORIE',  
        columns='INDEPOUSE'  
    )  
    .fill_null(0)  
)
```

Calcul des pourcentages par rapport à la somme de chaque ligne dans Polars

```
contingency_table_percent = contingency_table.with_columns([  
    ((pl.col('0') / (pl.col('0') + pl.col('1')))) *  
    100).alias('INDEPOUSE_0'),  
    ((pl.col('1') / (pl.col('0') + pl.col('1')))) *  
    100).alias('INDEPOUSE_1')  
)
```

```

# Configuration des styles pour le graphique
plt.style.use('ggplot')
plt.rcParams['font.family'] = 'DejaVu Sans'
plt.rcParams['axes.edgecolor'] = '#333F4B'
plt.rcParams['axes.linewidth'] = 0.8
plt.rcParams['xtick.color'] = '#333F4B'
plt.rcParams['ytick.color'] = '#333F4B'
plt.rcParams['text.color'] = '#333F4B'

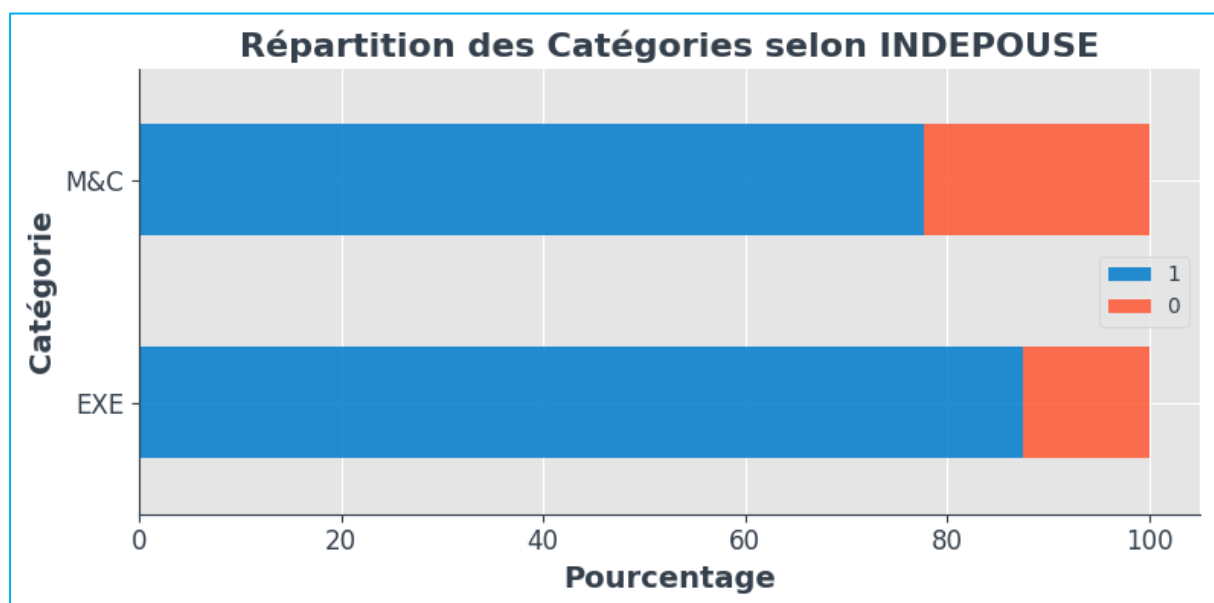
# Création du graphique avec les pourcentages
contingency_table_percent[['CATEGORIE', 'INDEPOUSE_0',
'INDEPOUSE_1']].to_pandas().plot(kind='barh', stacked=True, figsize=(8,
4), color=['#007ACC', '#FF5733'], alpha=0.85)

# Configuration des axes et des étiquettes
plt.title('Répartition des Catégories selon INDEPOUSE', fontsize=16,
weight='bold', color='#333F4B')
plt.xlabel('Pourcentage', fontsize=14, weight='bold', color='#333F4B')
plt.ylabel('Catégorie', fontsize=14, weight='bold', color='#333F4B')
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Suppression des spines inutiles
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Affichage du graphique
plt.tight_layout()
plt.show()

```



Nous constatons dans chaque catégorie que la proportion des hommes est largement supérieure à celle des femmes. Mais cela est dû à la faible représentativité des femmes par rapport aux hommes au sein de cette société, car les femmes sont moins de 15% dans cette firme.

- **Etude des liaisons et de corrélations**

Sur ce point nous allons essayer de voir le rapprochement qui existe entre différentes variables, car si les variables sont fortement liées cela risquera de créer des problèmes lors de la modélisation. Par exemple le problème de multicolinéarité¹.

1. Cas des variables Quantitatives

Ici nous faire la corrélation des variables quantitatives de Pearson.

```
# Calcul de la matrice de corrélation
corr_matrix = data_pd[['SALAIRE PERS', 'ENFANT', 'TRP', 'LGT', 'SSANTE',
'ENERGIE', 'IFT', 'IDP', 'IPM', 'age_engagement']].corr()

# Masque pour afficher uniquement la moitié supérieure de la matrice
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Création d'une figure
plt.figure(figsize=(12, 10))
sns.set_theme(style="white")

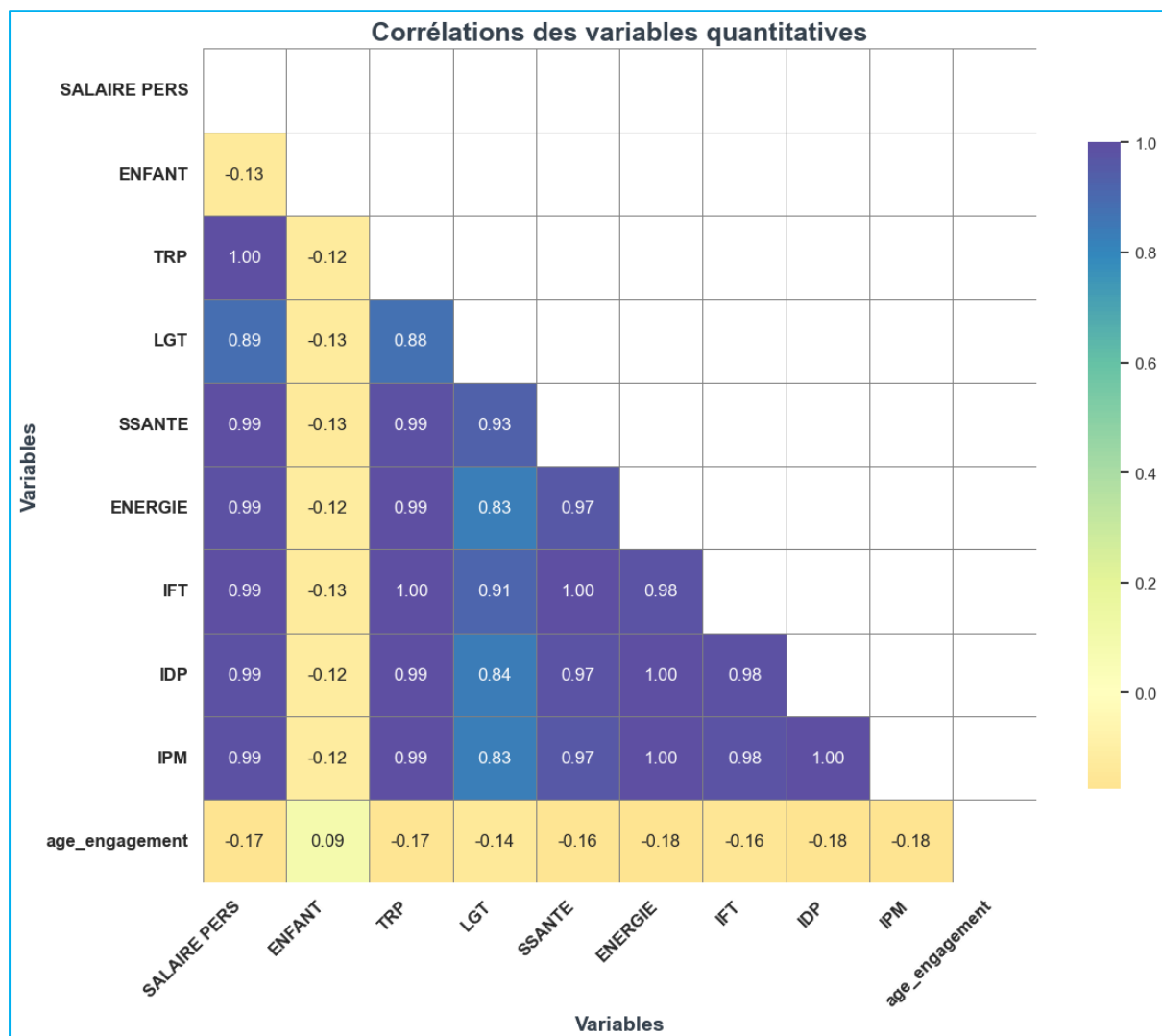
# Création de la heatmap avec un design plus sophistiqué
sns.heatmap(
    corr_matrix,
    mask=mask, # Application du masque
    annot=True, # Affiche les valeurs
    fmt=".2f", # Format des valeurs
    cmap='Spectral', # Palette de couleurs attrayante
    center=0, # Centre la palette autour de 0
    cbar_kws={'shrink': 0.75, 'aspect': 20}, # Configuration de la barre
de couleur
    linewidths=0.5, # Espacement entre les cellules
    linecolor='gray', # Couleur des lignes
    square=True # Format carré pour chaque cellule
)

# Personnalisation des axes et du titre
plt.title("Corrélations des variables quantitatives", fontsize=18,
weight='bold', color='#333F4B')
```

¹ La **multicolinéarité** est un phénomène statistique qui survient lorsque deux ou plusieurs variables indépendantes dans un modèle de régression sont fortement corrélées. Cela signifie qu'il existe une forte relation linéaire entre ces variables, ce qui complique l'estimation des coefficients de la régression. Mathématiquement, c'est si l'une au moins des variables peut s'écrire comme combinaison linéaire des autres variables.

```
plt.xticks(rotation=45, ha='right', fontsize=12, weight='bold')
plt.yticks(rotation=0, fontsize=12, weight='bold')
plt.xlabel('Variables', fontsize=14, weight='bold', color='#333F4B')
plt.ylabel('Variables', fontsize=14, weight='bold', color='#333F4B')

# Affichage du graphique
plt.tight_layout()
plt.show()
```



Nous remarquons des très fortes corrélations dans le sens positif entre différentes variables, ceci signifie que lorsque la valeur d'une variable augmente, la valeur de l'autre augmente **exactement** de la même manière, de façon proportionnelle.

Or, ces corrélations très fortes ou parfaites ne sont pas bien pour la modélisation, ça créera des problèmes de multicollinéarité.

D'où, certaines variables ne doivent pas figurer dans la modélisation (cfr : les variables en bleu sur le graphique).

2. Cas des variables Qualitatives

La corrélation dans le cadre des variables qualitative est appelée liaison, mais elle ne peut pas être calculée de la même manière que pour les variables quantitatives, car les méthodes classiques comme le **coefficient de corrélation de Pearson** ne s'appliquent qu'aux données numériques. Cependant, il existe d'autres méthodes adaptées pour mesurer l'association ou la dépendance entre des variables qualitatives :

1. Coefficient de corrélation de Cramér (Cramér's V)

- Il est utilisé pour mesurer la force de l'association entre deux variables catégorielles.
- Calculé à partir du **test du Chi-carré**, Cramér's V varie entre 0 (aucune association) et 1 (association parfaite).
- Il convient lorsque les variables ont plus de deux catégories.

2. Coefficient Phi (Φ)

- Utilisé pour mesurer l'association entre deux variables binaires (ayant chacune deux catégories).
- Il s'agit d'un cas particulier de Cramér's V lorsque le tableau de contingence est de dimension 2x2.

3. Test du Chi-carré d'indépendance

- Permet de vérifier si deux variables qualitatives sont statistiquement indépendantes ou s'il existe une association entre elles.
- Bien qu'il ne donne pas directement un coefficient de corrélation, il permet de déterminer si une relation significative existe entre les variables.

4. Indice de corrélation de Theil (Theil's U ou l'entropie conditionnelle)

- Utile pour mesurer la dépendance asymétrique entre deux variables catégorielles.
- Contrairement aux méthodes ci-dessus, cet indice peut être asymétrique (la relation d'une variable à l'autre peut ne pas être réciproque).

5. Corrélation de Kendall's Tau pour les variables ordinales

- Si vos variables qualitatives sont ordinales (elles ont un ordre mais ne sont pas numériques), vous pouvez utiliser la **corrélation de Kendall's Tau** pour mesurer la force de l'association ordonnée.

6. Tableaux de contingence

- En plus des coefficients, il est courant de visualiser les associations entre les variables qualitatives à l'aide de **tableaux croisés** (ou tableaux de contingence). Ces tableaux

montrent les fréquences de chaque combinaison possible de catégories entre les variables.

```
# Étape 1: Créer le tableau de contingence
contingency_table = pd.crosstab(index=[data_pd['CATEGORIE'],
data_pd['SEXE']], columns=data_pd['INDEPOUSE'])

# Étape 2: Afficher le tableau de contingence avec un design amélioré
print(Fore.MAGENTA + Style.BRIGHT + "### Tableau de Contingence ###")
print(Style.RESET_ALL)

# Format de tableau plus esthétique avec bordures et alignement
print(Fore.CYAN + tabulate(contingency_table, headers='keys',
tablefmt='fancy_grid', showindex=True, stralign="center"))
print(Style.RESET_ALL)
print(Fore.MAGENTA + Style.BRIGHT + "-"*40 + Style.RESET_ALL) #
Séparateur visuel

# Étape 3: Définir la fonction pour calculer le Cramér's V
def cramers_v(confusion_matrix):
    """
    Calcule la statistique de Cramér's V pour mesurer l'association entre
    les variables catégorielles.

    :param confusion_matrix: Un tableau de contingence
    :return: La valeur de Cramér's V (entre 0 et 1)
    """
    # Statistique du test du chi-carré et nombre total d'observations
    chi2_statistic = ss.chi2_contingency(confusion_matrix)[0]
    total_obs = confusion_matrix.sum().sum()

    # Calcul du phi-squared
    phi2 = chi2_statistic / total_obs

    # Nombre de lignes et de colonnes dans le tableau de contingence
    rows, cols = confusion_matrix.shape

    # Appliquer les corrections pour phi-squared
    phi2_corrected = max(0, phi2 - ((cols - 1) * (rows - 1)) / (total_obs
- 1))
    rows_corr = rows - ((rows - 1) ** 2) / (total_obs - 1)
    cols_corr = cols - ((cols - 1) ** 2) / (total_obs - 1)

    # Retourner la racine carrée de la valeur corrigée de phi-squared
    return np.sqrt(phi2_corrected / min((cols_corr - 1), (rows_corr -
1)))

# Étape 4: Calcul de Cramér's V
cramers_v_value = cramers_v(contingency_table)

# Étape 5: Affichage du résultat avec un design plus moderne
```

```

print(Fore.GREEN + Style.BRIGHT + "\n### Résultat Cramér's V ###")
print(Fore.GREEN + "-"*40) # Ligne de séparation
print(f"{Style.BRIGHT}Cramér's V:
{cramers_v_value:.4f}{Style.RESET_ALL}")
print(Fore.GREEN + "-"*40) # Ligne de séparation

# Interprétation de la valeur de Cramér's V
if crammers_v_value < 0.1:
    print(Fore.YELLOW + Style.BRIGHT + "Faible association entre les
variables." + Style.RESET_ALL)
elif crammers_v_value < 0.3:
    print(Fore.CYAN + Style.BRIGHT + "Association modérée entre les
variables." + Style.RESET_ALL)
else:
    print(Fore.RED + Style.BRIGHT + "Association forte entre les
variables." + Style.RESET_ALL)

# Réinitialiser les styles à la fin
print(Fore.MAGENTA + "-"*40 + Style.RESET_ALL) # Séparateur final
print(Style.RESET_ALL)

```

... **### Tableau de Contingence ###**

	0	1
('EXE', 'F')	187	3
('EXE', 'M')	190	2635
('M&C', 'F')	425	7
('M&C', 'M')	138	1949

Résultat Cramér's V

Cramér's V: 0.7712

Association forte entre les variables.

Nous remarquons une association forte entre la catégorie, le sexe et l'INDEPOUSE ; la force de cette association est de 77%.

Modélisation

Pour l'instant certaines variables sont à exclure à cause des incohérences sur ces variables. Parmi ces variables, nous excluons : 'DERNIER AVACEMENT EN GRADE', 'COTATION', 'DATENAISSA', 'DATEENGAGE', 'tranche_age_engagement', 'GRADE', 'ETATCIVIL'

shape: (5_744, 14)

CATEGORIE	SALAIRE PERS	CITE	SEXE	...	IFT	IDP	IPM	age_engagement
---	---	---	---	---	---	---	---	---
str	f64	str	str		f64	f64	i64	i32
M&C	47865.44225	06	M	...	97837.51	126115.18	348500	18
M&C	47652.18334	05	M	...	97837.51	126115.18	348500	26
M&C	47348.71096	06	M	...	97837.51	126115.18	348500	27
M&C	47115.68266	01	M	...	97837.51	126115.18	348500	24
M&C	47344.56758	02	M	...	97837.51	126115.18	348500	24
...
EXE	4920.44077	10	M	...	67568.84	46635.93	342500	33
EXE	4920.44077	10	M	...	67568.84	46635.93	342500	23
EXE	4920.44077	10	M	...	67568.84	46635.93	342500	26
EXE	4913.62077	10	M	...	67278.84	46413.51	342500	35
EXE	4913.62077	10	M	...	67278.84	46413.51	342500	32

Sélection des variables pour le modèle

```
data_model = data_pl_new.select(  
    pl.all().exclude(['DERNIER AVACEMENT EN GRADE', 'COTATION',  
    'DATENAISSA', 'DATEENGAGE', 'tranche_age_engagement', 'GRADE', 'ETATCIVIL'  
    ])  
)
```

Convertir les données Polars en DataFrame Pandas pour la manipulation

```
df = data_model.to_pandas()
```

Identifier les colonnes catégorielles

```
colonnes_categorielles = ["CITE", "SEXE", "INDEPOUSE"]
```

Appliquer un encodage LabelEncoder sur ces colonnes

```
label_encoders = {}  
for col in colonnes_categorielles:  
    le = LabelEncoder()  
    df[col] = le.fit_transform(df[col])  
    label_encoders[col] = le
```

Encodage de la variable cible

```
df["CATEGORIE"] = LabelEncoder().fit_transform(df["CATEGORIE"])
```

Séparer les données en X et y

```
X = df.drop(columns=["CATEGORIE"])  
y = df["CATEGORIE"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
    random_state=42) # 80% des données sont utilisées pour l'entraînement
```

```

# Initialiser et entraîner le modèle de régression logistique
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Prédiction et évaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)

```

```

Accuracy: 1.0
Classification Report:

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	632
1	1.00	1.00	1.00	517
accuracy			1.00	1149
macro avg	1.00	1.00	1.00	1149
weighted avg	1.00	1.00	1.00	1149

Les résultats de ce modèle paraissent meilleurs alors qu'il y a un problème, soulevé lors de l'analyse bivariées.

Pour mieux voir, les résultats nous utilisons la librairie statmodels :

```
Warning: Maximum number of iterations has been exceeded.
Current function value: 0.000000
Iterations: 35
```

Logit Regression Results						
Dep. Variable:	CATEGORIE	No. Observations:	4595			
Model:	Logit	Df Residuals:	4582			
Method:	MLE	Df Model:	12			
Date:	Sat, 02 Nov 2024	Pseudo R-squ.:	1.000			
Time:	13:40:12	Log-Likelihood:	-1.0203e-12			
converged:	False	LL-Null:	-3152.4			
Covariance Type:	nonrobust	LLR p-value:	0.000			
	coef	std err	z	P> z	[0.025	0.975]
SALAIRE PERS	1.884e-06	621.986	3.03e-09	1.000	-1219.070	1219.070
CITE	6.417e-05	2.24e+05	2.86e-10	1.000	-4.4e+05	4.4e+05
SEXE	3.088e-07	4.18e+06	7.39e-14	1.000	-8.19e+06	8.19e+06
INDEPOUSE	-6.063e-07	3.1e+06	-1.95e-13	1.000	-6.08e+06	6.08e+06
ENFANT	-1.028e-05	4.34e+05	-2.37e-11	1.000	-8.51e+05	8.51e+05
TRP	8.947e-06	1577.439	5.67e-09	1.000	-3091.723	3091.723
LGT	-1.299e-06	136.398	-9.52e-09	1.000	-267.336	267.336
SSANTE	-4.101e-06	232.712	-1.76e-08	1.000	-456.108	456.108
ENERGIE	0.0101	6.05e+04	1.67e-07	1.000	-1.19e+05	1.19e+05
IFT	-3.695e-05	2372.571	-1.56e-08	1.000	-4650.154	4650.154

Avec ces résultats, aucune variable n'est significative. Or ce problème a été décelé au niveau de l'analyse des corrélations.

Pour ça, on doit identifier les variables à problèmes à l'aide de l'étude de la multicolinéarité.

Etude de la multicolinéarité

```
# VIF pour chaque variable
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]
print(vif_data)
```

	feature	VIF
0	SALAIRE PERS	221.698094
1	CITE	1.078873
2	SEXE	1.746441
3	INDEPOUSE	1.702281
4	ENFANT	1.060799
5	TRP	2472.915463
6	LGT	20.045121
7	SSANTE	472.829447
8	ENERGIE	313247.777713
9	IFT	1813.738318
10	IDP	39994.592023
11	IPM	47944.429271
12	age_engagement	1.087536

Les variables à éliminer sont celles la valeur du VIF excède 10 ; ce qui confirme notre présomption au niveau de la corrélation.

La meilleure stratégie est d'éliminer une variable après une autre et voir le comportement du modèle. Mais sur base de notre hypothèse sur la corrélation certaines variables ne doivent pas figurer ensemble dans la modélisation. Entre autres : Salaire pers, TRP, LGT, SSANTE, ENERGIE, IFT, IDP, IPM.

Après plusieurs tests de suppression des variables, celles retenues sont :

shape: (5_744, 6)

CATEGORIE	CITE	SEXE	INDEPOUSE	ENFANT	age_engagement
---	---	---	---	---	---
str	str	str	str	i64	i32
M&C	06	M	1	0	18
M&C	05	M	0	0	26
M&C	06	M	1	0	27
M&C	01	M	1	0	24
M&C	02	M	1	1	24
...
EXE	10	M	null	0	33
EXE	10	M	null	0	23
EXE	10	M	null	0	26
EXE	10	M	null	0	35
EXE	10	M	null	0	32

```
# Convertir les données Polars en DataFrame Pandas pour la manipulation
df = data_model.to_pandas()
```

```
# Identifier les colonnes catégorielles
colonnes_categorielles = ["CITE", "INDEPOUSE", 'SEXE']
```

```
# Appliquer un encodage LabelEncoder sur ces colonnes
label_encoders = {}
```

```
for col in colonnes_categorielles:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

```
# Encodage de la variable cible
df["CATEGORIE"] = LabelEncoder().fit_transform(df["CATEGORIE"])
```

```
# Séparer les données en X et y
X = df.drop(columns=["CATEGORIE"])
y = df["CATEGORIE"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```

# Initialiser et entraîner le modèle de régression logistique
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Prédiction et évaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)

```

```

Accuracy: 0.6362053959965187
Classification Report:

```

	precision	recall	f1-score	support
0	0.63	0.80	0.71	632
1	0.64	0.44	0.52	517
accuracy			0.64	1149
macro avg	0.64	0.62	0.61	1149
weighted avg	0.64	0.64	0.62	1149

Pour mieux interpréter les résultats, nous utilisons la librairie statmodels :

```

model = sm.Logit(y_train, X_train)
result = model.fit(maxiter=1000)

```

```

# Afficher le résumé du modèle
print(result.summary())

```

```

Optimization terminated successfully.
Current function value: 0.662517
Iterations 5

```

Logit Regression Results						
Dep. Variable:	CATEGORIE	No. Observations:	4595			
Model:	Logit	Df Residuals:	4590			
Method:	MLE	Df Model:	4			
Date:	Sat, 02 Nov 2024	Pseudo R-squ.:	0.03429			
Time:	14:45:34	Log-Likelihood:	-3044.3			
converged:	True	LL-Null:	-3152.4			
Covariance Type:	nonrobust	LLR p-value:	1.219e-45			
	coef	std err	z	P> z	[0.025	0.975]
CITE	-0.0592	0.007	-8.939	0.000	-0.072	-0.046
SEXE	0.1462	0.112	1.300	0.194	-0.074	0.367
INDEPOUSE	-0.4444	0.091	-4.881	0.000	-0.623	-0.266
ENFANT	-0.0894	0.013	-6.930	0.000	-0.115	-0.064
age_engagement	0.0140	0.003	4.672	0.000	0.008	0.020

On sait maintenant voir que la modèle a une précision de 66% sur les données d'entraînement et une précision de 63% celles de test.

En plus, tous les paramètres sont significatifs pour l'explication du modèle à un niveau de confiance de 95%.

Recommandations

Ces analyses sont faites sur base de la nature des variables, d'autres pistes de recherche sont aussi viables, entre autres :

- Exploration des autres modèles de Machine learning (Analyse discriminante, K-means, ...)
- Ajout d'une régularisation : cas de la régression logistique avec régularisation (comme Ridge ou Lasso)
- Réaliser l'AFDM (Analyse Factorielle Des Données Mixtes), afin de détecter les variables les plus corrélées avec la catégorie de l'employé, puis les éliminer.
- Etc.

Code source du travail

