Université Cadi Ayyad

Analyse des données des employées et classification

Faculté des Sciences et Techniques - Marrakech

Projet d'Analyse des Données : IFA2 & **IRISI2**

Réalisé par : ZIYATI MOHAMED & KAABOUCH MOHAMED

Encadré par : Pr. AZIZ OUAARAB

Année Universitaire 2023–2024

Tables des matières

Description du jeu de données utilisé

Le jeu de données *HR_Data.xlsx* contient une liste de **1371 employées** qui ont travaillé dans une entreprise depuis 1990.
Voici une petite description de chaque variable :
1. **Age**: L'âge des employés en années.
2. **Attrition**: Indique si l'employé a quitté l'entreprise ou non, souvent désigné par *Yes* pour quitter et *No* pour ne pas quitter.
3. **BusinessTravel**: Le niveau de fréquence des déplacements professionnels effectués par l'employé, généralement classé comme Non_Travel (pas de voyage), Travel_Frequently (voyage fréquent) ou Travel_Rarely (voyage rare).
4. **DailyRate**: Le taux journalier de rémunération des employés.
5. **Department**: Le département dans lequel travaille l'employé, tel que *HR* (Ressources Humaines), *R&D* (recherche et développement) ou *Sales* (ventes).
6. **DistanceFromHome**: La distance en kilomètres entre le lieu de travail de l'employé et son domicile.
7. **Education**: Le niveau d'éducation atteint par l'employé, généralement noté de 1 à 5.
8. **EducationField**: Le domaine d'études de l'employé, tel que *HR* (Ressources Humaines), *Life Sciences* (sciences de la vie), *Marketing*, *Medical* (médical), *Technical degree* (diplôme technique) ou *other* (autre).
9. **EmployeeNumber**: L'***identifiant*** unique de chaque employé.
10. **EnvironmentSatisfaction**: Le niveau de satisfaction de l'employé vis-à-vis de l'environnement de travail, généralement noté de 1 à 4.
11. **Gender**: Le genre de l'employé, souvent désigné par "*Male*" (homme) ou *Female* (femme).
12. **HourlyRate**: Le taux horaire de rémunération des employés.
13. **JobInvolvement**: Le degré d'implication de l'employé dans son travail, généralement noté de 1 à 4.

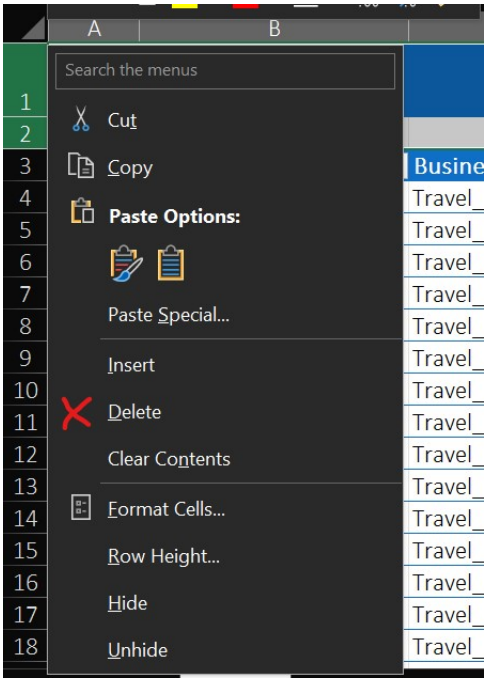
14. **JobLevel**: Le niveau hiérarchique de l'employé dans l'entreprise, généralement noté de 1 à 5.
15. **JobRole**: Le rôle ou le poste occupé par l'employé dans l'entreprise, par exemple *Laboratory Technician*, *Research Scientist*, *Manager*, etc.
16. **JobSatisfaction**: Le niveau de satisfaction de l'employé par rapport à son travail, généralement noté de 1 à 4.
17. **MaritalStatus**: L'état civil de l'employé, souvent désigné par *Single* (célibataire), *Married* (marié) ou *Divorced* (divorcé).
18. **MonthlyIncome**: Le revenu mensuel de l'employé.
19. **NumCompaniesWorked**: Le nombre d'entreprises pour lesquelles l'employé a travaillé auparavant.
20. **OverTime**: Indique si l'employé a travaillé des heures supplémentaires ou non, généralement désigné par *Yes* (oui) ou *No* (non).
21. **PercentSalaryHike**: Le taux d'augmentation salariale accordée à l'employé.
22. **PerformanceRating**: Le niveau de performance de l'employé, généralement noté 3 ou 4.
23. **RelationshipSatisfaction**: Le niveau de satisfaction de l'employé concernant ses relations interpersonnelles au travail, généralement noté de 1 à 4.
24. **StockOptionLevel**: Le niveau d'options d'achat d'actions disponibles pour l'employé, généralement noté de 1 à 3.
25. **TotalWorkingYears**: Le nombre total d'années d'expérience professionnelle de l'employé.
26. **TrainingTimesLastYear**: Le nombre de fois où l'employé a suivi une formation l'année précédente.
27. **WorkLifeBalance**: L'équilibre entre la vie professionnelle et la vie personnelle perçu par l'employé, généralement noté de 1 à 4.
28. **YearsAtCompany**: Le nombre d'années que l'employé a passées dans l'entreprise actuelle.
29. **YearsInCurrentRole**: Le nombre d'années que l'employé a passées dans son rôle actuel.
30. **YearsSinceLastPromotion**: Le nombre d'années écoulées depuis la dernière promotion de l'employé.
31. **YearsWithCurrManager**: Le nombre d'années que l'employé a passé avec son gestionnaire actuel.

Nettoyage et transformation des données

2.1 Suppression des lignes inutiles

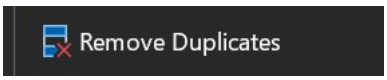
Notre jeu de données contient 2 lignes inutiles. Donc, on peut les supprimer. Nous simplifions ainsi notre ensemble de données et donc on peut se concentrer sur les informations pertinentes pour notre analyse.

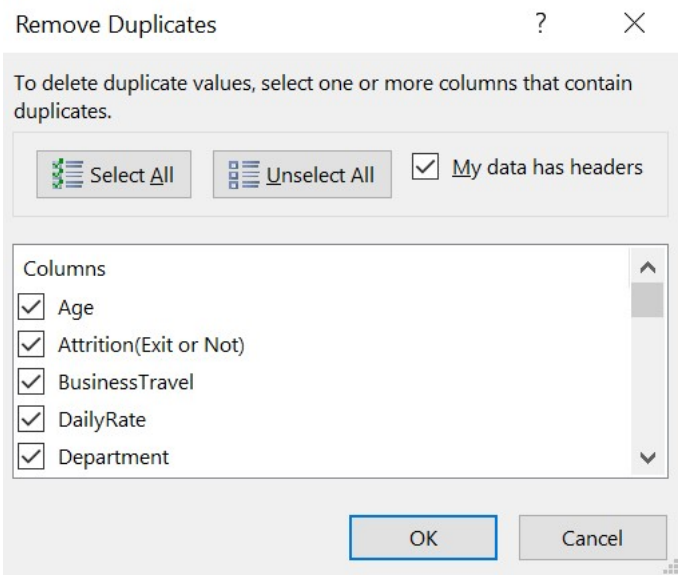
	A	B	C	D	E	F
1	HR Data					
2						
3	Age	Attrition(Exit or Not)	BusinessTravel	DailyRate	Department	DistanceFromHome
4	37	Yes	Travel_Rarely	1373	R&D	2
5	21	No	Travel_Rarely	391	R&D	15
6	45	No	Travel_Rarely	193	R&D	6
7	23	No	Travel_Rarely	541	Sales	2
8	22	No	Travel_Rarely	534	R&D	15
9	19	Yes	Travel_Rarely	528	Sales	22
10	19	Yes	Travel_Frequently	602	Sales	1
11	28	Yes	Travel_Rarely	529	R&D	2
12	29	No	Travel_Rarely	1210	Sales	2
13	18	Yes	Travel_Rarely	230	R&D	3
14	18	No	Travel_Rarely	812	Sales	10
15	47	No	Travel_Frequently	1309	Sales	4
16	48	No	Travel_Rarely	530	Sales	29
17	18	Yes	Travel_Frequently	1306	Sales	5
18	26	No	Travel_Rarely	775	Sales	29



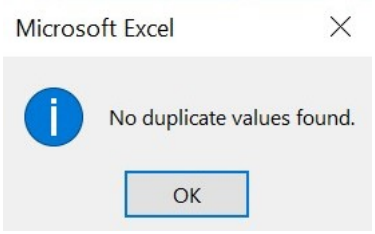
2.2 Suppression des Redondances

La suppression des redondances est importante lors du nettoyage des données. Les redondances, c'est-à-dire les informations dupliquées, peuvent entraîner des problèmes tels que des analyses incorrectes, une consommation inutile de ressources de stockage et une complexité accrue lors de l'utilisation des données.
La suppression des redondances contribue à maintenir la qualité, la cohérence et l'efficacité des données.
On peut supprimer les redondances en utilisant l'outil ***Remove Duplicates*** sur `Excel`





Dans notre jeu de données, il n'y a pas de redondances :

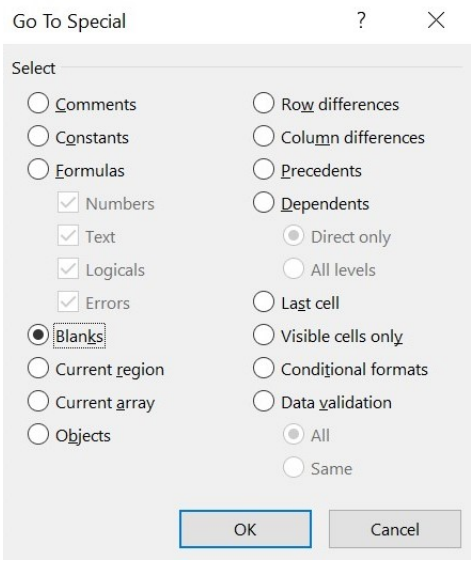


2.3 Recherche et Remplissage des valeurs manquantes

Rechercher et compléter les valeurs vides est une étape importante dans le processus de nettoyage des données. Les valeurs manquantes ou vides peuvent avoir un impact significatif sur l'analyse des données, car elles peuvent fausser les résultats ou conduire à des conclusions erronées si elles ne sont pas correctement traitées.

Détection des valeurs manquantes :

On Utilise l'outil `Go to Special` pour allez chercher juste ce qu'on veut, maintenant on veut chercher les cellules vides, donc on choisit `Blanks` .



Et toutes les valeurs manquantes seront sélectionnées :

	YearsSinceLastPromotion	YearsWithCurrManager
1354	15	2
1355	15	14
1356	12	11
1357	3	7
1358	12	17
1359	13	11
1360	11	8
1361	1	12
1362	4	11
1363	2	17
1364	0	11
1365	0	13
1366	11	5
1367	2	13
1368	14	12
1369	1	11
1370	12	8
1371	11	8
1372	2	7
1373	1	9
1374	15	
1375		

Nous avons détecté une valeur manquante qui se trouve à la fin de la colonne *YearsWithCurrManager*

► Allons maintenant sur `Python` pour remplir les valeurs manquantes. On utilise la méthode `fillna()` de la bibliothèque Pandas pour remplir les valeurs manquantes dans la colonne *YearsWithCurrManager*, et la fonction `ROUND`, utilisée pour arrondir la moyenne calculée à l'entier le plus proche.

Cela garantit que la valeur utilisée pour remplir les valeurs manquantes est un nombre entier, ce qui est plus approprié dans notre cas, car les années avec le même manager sont généralement des nombres entiers dans notre ensemble de données.

Donc le résultat final sera le remplissage des valeurs manquantes *avec la moyenne de la colonne, arrondie à l'entier le plus proche :*

```
# Charger les données depuis le fichier Excel
df = pd.read_excel('HR_DATA.xlsx', sheet_name='HR Data')
print(df['YearsWithCurrManager'])

print('  ')
print('Apres le remplissage des valeurs manquantes, on
    obtient :')
MEAN = df['YearsWithCurrManager'].mean()

df['YearsWithCurrManager'].fillna(round( MEAN ) , inplace =
    True )
print(df['YearsWithCurrManager'])
```

Résultat :

```
0       0.0
1       0.0
2       0.0
3       0.0
4       0.0
       ...
1366    8.0
1367    8.0
1368    7.0
1369    9.0
1370    NaN
Name: YearsWithCurrManager, Length: 1371, dtype: float64

Apres le remplissage des valeurs manquantes, on obtient :
0       0.0
1       0.0
2       0.0
3       0.0
4       0.0
       ...
1366    8.0
1367    8.0
```

```
1368    7.0
1369    9.0
1370    4.0
Name: YearsWithCurrManager, Length: 1371, dtype: float64
```

2.4 Suppression des lignes contenant des valeurs incorrectes

La suppression des lignes contenant des valeurs incorrectes est une étape importante dans le processus d'analyse des données. Ces valeurs erronées peuvent être le résultat d'une *erreur de saisie*.
En éliminant ces valeurs, on garantit que les analyses sont basées sur des données fiables et précises.

```
#Suppression des lignes contenant des valeurs incorrectes :
df = pd.read_excel('HR_DATA.xlsx', sheet_name='HR Data')
df = df[(df['Age'] >= 0) & (df['DistanceFromHome'] >= 0 ) & (
    df['Education'] > 0) & (df['MonthlyIncome'] >=0 ) & (df['
    NumCompaniesWorked'] >=0 ) & (df['TotalWorkingYears'] >=0
    ) & (df['YearsAtCompany'] >=0) & (df['YearsInCurrentRole']
     >=0) ]
df
```

Résultat :

	Age	Attrition(Exit or Not)	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber(id)	I
0	37	Yes	Travel_Rarely	1373	R&D	2	2	Other	4	
1	21	No	Travel_Rarely	391	R&D	15	2	Life Sciences	30	
2	45	No	Travel_Rarely	193	R&D	6	4	Other	101	
3	23	No	Travel_Rarely	541	Sales	2	1	Technical Degree	113	
4	22	No	Travel_Rarely	534	R&D	15	3	Medical	144	
...	...	...	...	...	...	...	...	...	...	
1366	45	No	Non_Travel	1052	Sales	6	3	Medical	302	
1367	39	No	Travel_Frequently	505	R&D	2	4	Technical Degree	343	
1368	47	No	Travel_Frequently	1379	R&D	16	4	Medical	987	
1369	38	No	Travel_Rarely	330	R&D	17	1	Life Sciences	1088	
1370	39	No	Travel_Rarely	412	R&D	13	4	Medical	1307	

1371 rows × 31 columns

▶ Le résultat après la suppression des valeurs incorrectes contient le même nombre de lignes que le jeu de données avant ce traitement (1371 lignes).

Donc notre jeu de données ne contient aucune valeur incorrecte pour les colonnes choisies.

Application des techniques d'Analyse des données

3.1 Prédiction salaire d'un employé d'après son niveau

```
import pandas as pd
import matplotlib.pyplot as plt

# Chargement des données depuis le fichier Excel
df = pd.read_excel('HR_DATA.xlsx', sheet_name='HR Data')
import seaborn as sns

df_REG = df[['JobLevel' , 'MonthlyIncome' ]]
X_REG = df_REG["JobLevel"]
y_REG = df["MonthlyIncome"]
df_REG
```

	JobLevel	MonthlyIncome
0	1	2090
1	1	1232
2	3	13245
3	1	2322
4	1	2871
...	...	...
1366	3	8865
1367	3	10938
1368	2	5067
1369	3	8823
1370	4	17123

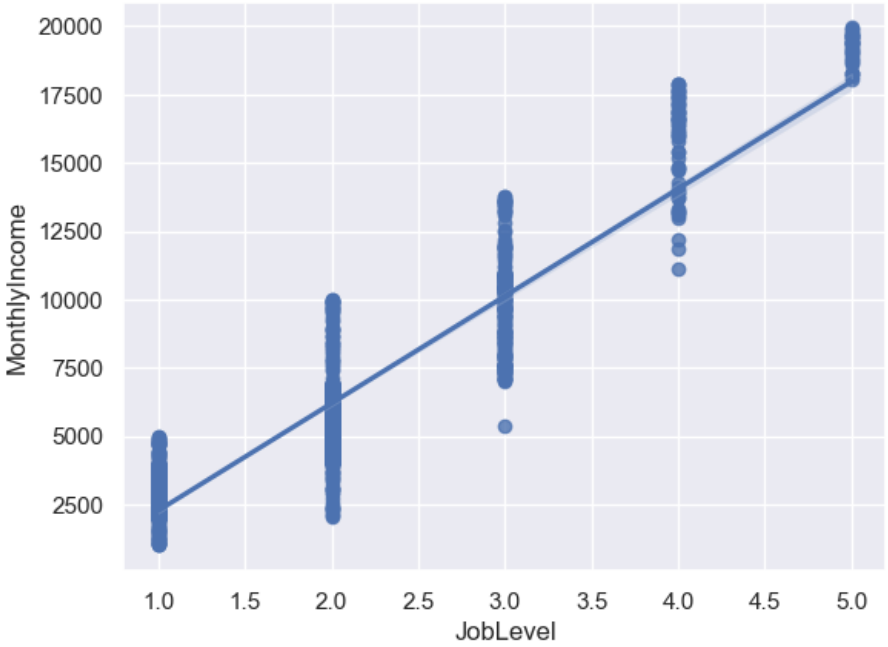
1371 rows × 2 columns

```
print("Correlation : ")
print(df_REG.corr())

sns.regplot(X_REG, y_REG)
```

```
Correlation :
                JobLevel  MonthlyIncome
JobLevel        1.000000       0.936081
MonthlyIncome   0.936081       1.000000
```

▶ On remarque une forte corrélation positive entre le `JobLevel` et `le salaire mensuel`, Donc on peut utiliser le modèle de régression linéaire pour prédire le salaire mensuel d'un employé .

Définition du modèle :
Le modèle s'écrit : $Y = \beta_0 + \beta_1 X + \varepsilon$
Dans notre cas :
Y est `le salaire mensuel`
X est le `JobLevel`
ε est un terme d'erreur
• Le modèle utilise `la méthode des moindres carrés ordinaires` pour estimer les coefficients β_0 et β_1, qui seront calculé par la suite .



```
X_REG = sm.add_constant(X_REG)
model = sm.OLS(y_REG,X_REG).fit() #moindres carrés ordinaires
print(model.summary())
```

OLS Regression Results

Dep. Variable:	MonthlyIncome	R-squared:	0.876
Model:	OLS	Adj. R-squared:	0.876
Method:	Least Squares	F-statistic:	9693.
Date:	Sat, 04 May 2024	Prob (F-statistic):	0.00
Time:	16:24:07	Log-Likelihood:	-11917.
No. Observations:	1371	AIC:	2.384e+04
Df Residuals:	1369	BIC:	2.385e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1644.4814	85.978	-19.127	0.000	-1813.144	-1475.819
JobLevel	3923.7330	39.853	98.455	0.000	3845.554	4001.912

Omnibus:	6.924	Durbin-Watson:	2.026
Prob(Omnibus):	0.031	Jarque-Bera (JB):	7.382
Skew:	0.120	Prob(JB):	0.0250
Kurtosis:	3.268	Cond. No.	5.61

► Interprétation des résultats obtenus :

• Une valeur de coefficient de détermination $R^2 = 0.876$ indique que 87.6% de la variance du salaire mensuel est expliquée par le niveau de poste. C'est une indication que le modèle s'ajuste bien aux données.

• Le coefficient de `JobLevel` qui égale a `3923.73` indique que pour chaque augmentation d'une unité dans le niveau de poste, le salaire mensuel augmente de 3923.73.

• Les `p-value` associées aux coefficients de `const` et `JobLevel` sont toutes inférieures à `0.05`, ce qui indique que les coefficients sont statistiquement significatifs au niveau de confiance de 95%, c-a-d On rejette l'hypothèse H_0 qui

postule que le coefficient est égal à zéro, c-a-d il n'y a pas de relation linéaire entre le `JobLevel` et le salaire mensuel `MonthlyIncome`.

• Les coefficients obtenus peuvent être utilisés pour prédire le salaire en fonction du `JobLevel`(en gardant à l'esprit que d'autres facteurs non inclus dans ce modèle peuvent également influencer le salaire mensuel).

3.2 Prédiction des années que l'employé a passées dans l'entreprise en fonction de l'ancienneté dans le poste actuel et avec le même manager

```
import seaborn as sns

df_REG_M = df[['YearsInCurrentRole' , 'YearsWithCurrManager',
    'YearsAtCompany' ]]
X_REG_M = df_REG_M[['YearsInCurrentRole' , '
   YearsWithCurrManager']]
y_REG_M = df["YearsAtCompany"]
df_REG_M
```

	YearsInCurrentRole	YearsWithCurrManager	YearsAtCompany
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
...	...	...	...
1366	7	8	19
1367	6	8	19
1368	10	7	19
1369	9	9	19
1370	9	4	19

```
print("Correlation : ")
print(df_REG_M.corr())
```

```
Correlation :
                      YearsInCurrentRole  YearsWithCurrManager  YearsAtCompany
YearsInCurrentRole              1.000000              0.739693        0.865413
```

YearsWithCurrManager	0.739693	1.000000	0.840705
YearsAtCompany	0.865413	0.840705	1.000000

▶ On remarque une forte corrélation positive entre le `YearsAtCompany` et `YearsWithCurrManager` et `YearsInCurrentRole` , Donc on peut utiliser le modèle de régression linéaire multiple pour prédire `YearsAtCompany` le nombre d'année que l'employé a passé dans l'entreprise .

Définition du modèle :
Le modèle s'écrit : $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$
Dans notre cas :
Y est `YearsAtCompany` : le nombre d'année que l'employé a passé dans l'entreprise .
X_1 est le `YearsInCurrentRole` : le nombre d'année que l'employé a passé dans le même poste .
X_2 est le `YearsWithCurrManager` : le nombre d'année que l'employé a passé avec ce même manager .
ε est un terme d'erreur .
• Le modèle utilise `la méthode des moindres carrés ordinaires` pour estimer les coefficients β_0, β_1 et β_2, qui seront calculé par la suite .

```
X_REG_M = sm.add_constant(X_REG_M)
# Create a fitted model
lm = sm.OLS(y_REG_M , X_REG_M).fit()

# Print model summary
print(lm.summary())
```

Résultat :

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         YearsAtCompany   R-squared:                       0.838
Model:                            OLS   Adj. R-squared:                  0.838
Method:                 Least Squares   F-statistic:                     3532.
Date:                Sat, 04 May 2024   Prob (F-statistic):               0.00
Time:                        16:38:47   Log-Likelihood:                -2630.2
No. Observations:                1371   AIC:                             5266.
Df Residuals:                    1368   BIC:                             5282.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                             coef    std err          t      P>|t|      [0.025
------------------------------------------------------------------------------
```

const	1.1607	0.071	16.362	0.000	1.022
YearsInCurrentRole	0.6589	0.020	33.234	0.000	0.620
YearsWithCurrManager	0.5561	0.020	27.368	0.000	0.516

Omnibus:	513.968	Durbin-Watson:	1.198
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2352.825
Skew:	1.728	Prob(JB):	0.00
Kurtosis:	8.407	Cond. No.	11.2

▶ **Interprétation du résultat obtenu :**

• Une valeur de coefficient de détermination $R^2 = 0.838$ indique que 83.8% de la variance de `YearsAtCompany` est expliquée par les variables `YearsInCurrentRole` et `YearsWithCurrManager`. Cela signifie que le modèle s'ajuste bien aux données et explique une grande partie de la variance de la variable dépendante.

• Le coefficient de `YearsInCurrentRole` est égal à `0.658`, ce qui indique que pour chaque augmentation d'une année dans le rôle actuel, le nombre d'années dans l'entreprise `YearsAtCompany` augmente de `0.658` année, en tenant compte des autres variables du modèle.

• Le coefficient de `YearsWithCurrManager` est égal à `0.556`, ce qui signifie que pour chaque année supplémentaire passée avec le manager actuel, le nombre d'années dans l'entreprise `YearsAtCompany` augmente de `0.556` année, en tenant compte des autres variables du modèle.

• Les `p-values` associées aux coefficients de `const`, `YearsInCurrentRole` et `YearsWithCurrManager` sont toutes inférieures à `0.05`. Donc on rejette l'hypothèse nulle H_0 (au niveau de confiance de 95%) qui postule que tous les coefficients sont égaux à zéro, ce qui signifie qu'il existe une relation linéaire significative entre `YearsInCurrentRole`, `YearsWithCurrManager`, et `YearsAtCompany`.

• Les coefficients obtenus peuvent être utilisés pour prédire le nombre d'années passées dans l'entreprise `YearsAtCompany` en fonction du nombre d'années passées dans le rôle actuel `YearsInCurrentRole` et du nombre d'années passées avec le manager actuel `YearsWithCurrManager`. Il est important de noter que d'autres facteurs non inclus dans ce modèle peuvent également influencer le nombre d'années passées dans cette l'entreprise.

3.3 L'entreprise paie mieux quel département ?

Pour répondre a cette question, on peut appliquer une méthode dite : **Analyse de la variance à un facteur**.
L'analyse de la variance à un facteur est une méthode statistique utilisée pour comparer les moyennes de différents groupes afin de déterminer s'ils sont statistiquement différents les uns des autres.
Elle évalue si les moyennes de plusieurs groupes sont égales ou non en examinant la variance au sein des groupes (*variance intra-groupes*) et la variance entre les groupes (*variance inter-groupes*).
Définition du modèle :
On modélise une variable quantitative en fonction d'un facteur à I niveaux. y est la variable à expliquer qui prend la valeur y_{ij} pour l'individu j du niveau i du facteur. Le modèle s'écrit :

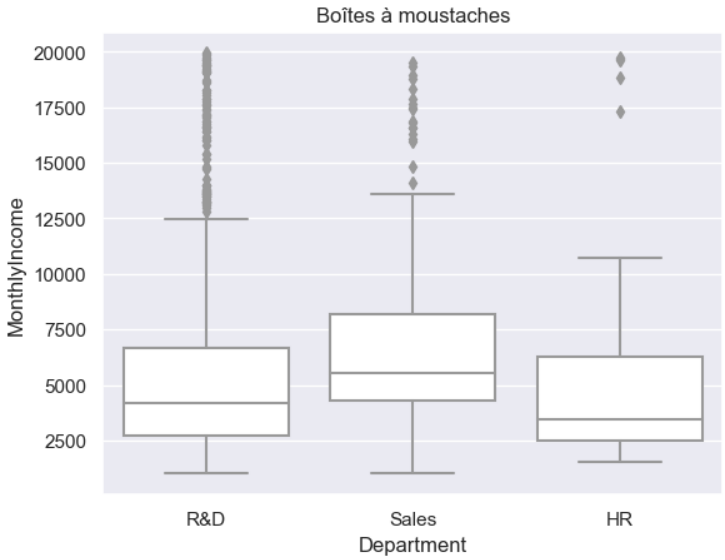
$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

Où:
- y_{ij} est la valeur du salaire mensuel observée pour l'observation j du groupe i.
- μ est la moyenne globale des salaires de toutes les observations.
- α_i est l'effet du groupe i.
- ε_{ij} est l'erreur aléatoire pour l'observation j du groupe i.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

Traçage du **BoxPlot** :

```
sns.set()

ax = sns.boxplot(x="Department", y="MonthlyIncome", data=df,
    color='white')
plt.xlabel('Department')
plt.ylabel('MonthlyIncome')
plt.title('Boîtes à moustaches')
plt.show()
```



Application du modèle $ANOVA$:

```
anova = smf.ols('MonthlyIncome~Department', data=df).fit()
sm.stats.anova_lm(anova)
```

Résultat :

	df	sum_sq	mean_sq	F	PR(>F)
Department	2.0	1.540304e+08	7.701518e+07	4.611935	0.010088
Residual	1368.0	2.284437e+10	1.669910e+07	NaN	NaN

► Interprétation du résultat obtenu :
Soit H_0 et H_1 tel que :
- (H_0) : Les moyennes des salaires mensuels pour les différents départements sont égales.
- (H_1) : Il existe au moins un département dont la moyenne des salaires mensuels est différente.

• La `p-value` correspond au facteur `Department` égale à `0.01`, qui est inférieure au seuil de `0.05`, Donc on rejette H_0.
Ce qui signifie qu'il existe des différences significatives dans les salaires mensuels moyens, entre les différents départements.

3.4 Le nombre d'années d'études permet-il d'obtenir un bon salaire dans cette entreprise ?

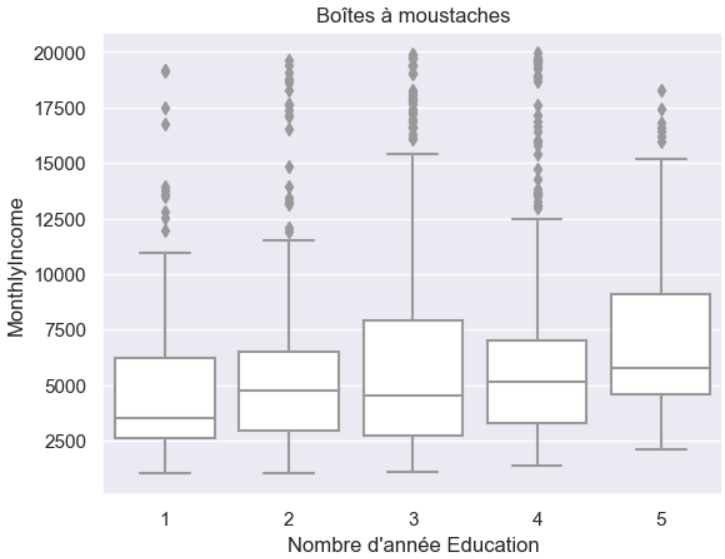
Pour répondre à cette question, appliquant une autre fois la méthode $ANOVA$, sur le facteur *Nombre d'années d'éducation* :
Définition du modèle :
On modélise une variable quantitative en fonction d'un facteur à I niveaux. y est la variable à expliquer qui prend la valeur y_{ij} pour l'individu j du niveau i du facteur. Le modèle s'écrit :

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

Où:
- y_{ij} est la valeur du salaire mensuel observée pour l'observation j du groupe i.
- μ est la moyenne globale des salaires de toutes les observations.
- α_i est l'effet du groupe i.
- ε_{ij} est l'erreur aléatoire pour l'observation j du groupe i.

Traçage du **BoxPlot** :

```
sns.set()

ax = sns.boxplot(x="Education", y="MonthlyIncome", data=df,
    color='white')
plt.xlabel('Nombre d\'année Education')
plt.ylabel('MonthlyIncome')
plt.title('Boîtes à moustaches')
plt.show()
```



Application du modèle $ANOVA$:

```
anova = smf.ols('MonthlyIncome~Education', data=df).fit()
sm.stats.anova_lm(anova)
```

Résultat :

	df	sum_sq	mean_sq	F	PR(>F)
Education	1.0	2.950438e+08	2.950438e+08	17.790977	0.000026
Residual	1369.0	2.270336e+10	1.658390e+07	NaN	NaN

▶ **Interprétation du résultat obtenu :**

Soit H_0 et H_1 tel que :
- (H_0) : Les moyennes des salaires mensuels pour les différents "années d'éducation", sont égales.
- (H_1) : Il existe au moins un "*nombre d'année d'éducation*" qui permet d'obtenir un salaires différent (en moyenne).

• La `p-value` correspond au facteur `Education`, c-a-d nombre d'année d'éducation, égale à `0.000026`, qui est inférieure au seuil de `0.05`, Donc on rejette H_0.
Ce qui signifie qu'il existe des différences significatives dans les salaires mensuels moyens, entre les différents nombre d'année d'éducation .

3.5 Quelle spécialité est mieux payée dans cette entreprise ?

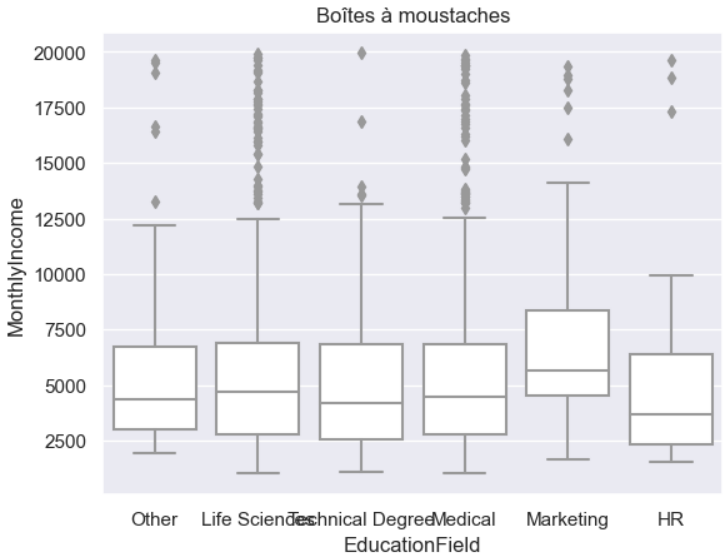
Pour répondre à cette question, appliquant une autre fois la méthode $ANOVA$, sur le facteur *EducationField* (Domaine de l'éducation):
Définition du modèle :
On modélise une variable quantitative en fonction d'un facteur à I niveaux. y est la variable à expliquer qui prend la valeur y_{ij} pour l'individu j du niveau i du facteur. Le modèle s'écrit :

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

Où:
- y_{ij} est la valeur du salaire mensuel observée pour l'observation j du groupe i.
- μ est la moyenne globale des salaires de toutes les observations.
- α_i est l'effet du groupe i.
- ε_{ij} est l'erreur aléatoire pour l'observation j du groupe i.

Traçage du **BoxPlot** :

```
sns.set()

ax = sns.boxplot(x="EducationField", y="MonthlyIncome", data=
    df, color='white')
plt.xlabel('EducationField')
plt.ylabel('MonthlyIncome')
plt.title('Boîtes à moustaches')
plt.show()
```



Application du modèle $ANOVA$:

```
anova = smf.ols('MonthlyIncome~EducationField',data=df).fit()
sm.stats.anova_lm(anova)
```

Résultat :

	df	sum_sq	mean_sq	F	PR(>F)
EducationField	5.0	1.104156e+08	2.208311e+07	1.316999	0.254099
Residual	1365.0	2.288799e+10	1.676776e+07	NaN	NaN

► Interprétation du résultat obtenu :
Soit H_0 et H_1 tel que :
- (H_0) : Les moyennes des salaires mensuels pour les différents domaines d'éducation, sont égales.
- (H_1) : Il existe au moins un *domaine d'éducation* qui est bien payé dans cette entreprise (en moyenne).

• La `p-value` correspond au facteur `EducationField` (Domaine d'éducation) égale à `0.254`, qui est supérieure au seuil de `0.05`, Donc on ne peut pas rejeter H_0.
On conclut qu'il n'y a pas de preuve suffisante pour dire que la spécialité de diplôme a un effet significatif sur les salaires mensuels moyens.

3.6 L'entreprise rémunère davantage les hommes ou les femmes ?

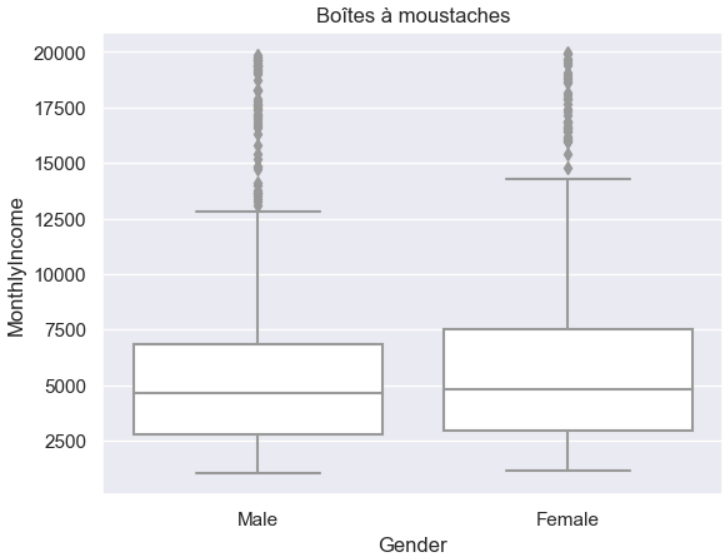
Pour répondre à cette question, appliquant une autre fois la méthode $ANOVA$, sur le facteur *Gender* (le sexe):
Définition du modèle :
On modélise une variable quantitative en fonction d'un facteur à I niveaux. y est la variable à expliquer qui prend la valeur y_{ij} pour l'individu j du niveau i du facteur. Le modèle s'écrit :

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

Où:
- y_{ij} est la valeur du salaire mensuel observée pour l'observation j du groupe i.
- μ est la moyenne globale des salaires de toutes les observations.
- α_i est l'effet du groupe i.
- ε_{ij} est l'erreur aléatoire pour l'observation j du groupe i.

Traçage du **BoxPlot** :

```
sns.set()

ax = sns.boxplot(x="Gender", y="MonthlyIncome", data=df,
    color='white')
plt.xlabel('Gender')
plt.ylabel('MonthlyIncome')
plt.title('Boîtes à moustaches')
plt.show()
```



Application du modèle $ANOVA$:

```
anova = smf.ols('MonthlyIncome~Gender', data=df).fit()
sm.stats.anova_lm(anova)
```

Résultat :

	df	sum_sq	mean_sq	F	PR(>F)
Gender	1.0	6.662702e+07	6.662702e+07	3.977554	0.046308
Residual	1369.0	2.293178e+10	1.675075e+07	NaN	NaN

▶ Interprétation du résultat obtenu :
Soit H_0 et H_1 tel que :
- (H_0) : Les moyennes des salaires mensuels pour les hommes et les femmes, sont égales.
- (H_1) : Le salaire moyen de l'un des deux sexes est différent(en moyenne) de celui de l'autre .

• La `p-value` correspond au facteur `Gender` (le sexe) égale à `0.0463`, qui est inférieure au seuil de `0.05`, Donc on rejette H_0.
Ce qui signifie qu'il existe une différence significative dans les salaires mensuels moyens, entre les hommes et les femmes dans cette entreprise .

3.7 Les déplacements professionnels peuvent-ils augmenter le salaire des employés ?

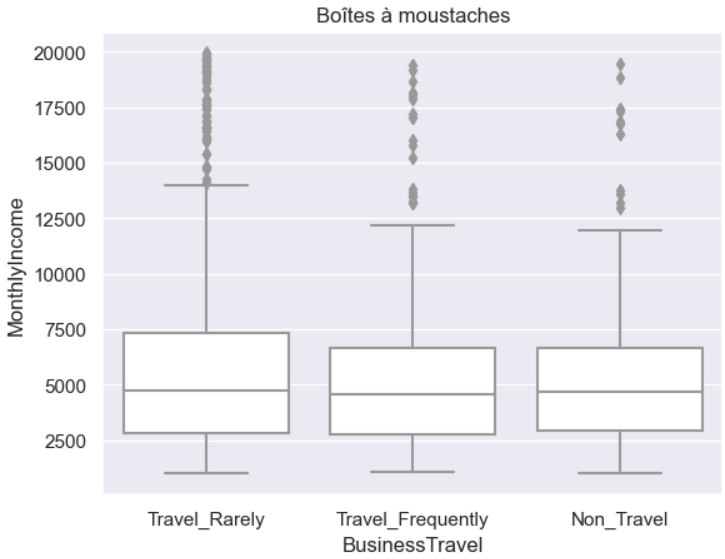
Pour répondre à cette question, appliquons une autre fois la méthode $ANOVA$, sur Le facteur *BusinessTravel* indique si l'employé fait des voyages professionnels, et à quelle fréquence (pas de voyages, rarement, fréquemment) :
Définition du modèle :
On modélise une variable quantitative en fonction d'un facteur à I niveaux. y est la variable à expliquer qui prend la valeur y_{ij} pour l'individu j du niveau i du facteur. Le modèle s'écrit :

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

Où:
- y_{ij} est la valeur du salaire mensuel observée pour l'observation j du groupe i.
- μ est la moyenne globale des salaires de toutes les observations.
- α_i est l'effet du groupe i.
- ε_{ij} est l'erreur aléatoire pour l'observation j du groupe i.

Traçage du **BoxPlot** :

```
sns.set()

ax = sns.boxplot(x="BusinessTravel", y="MonthlyIncome", data=
    df, color='white')
plt.xlabel('BusinessTravel')
plt.ylabel('MonthlyIncome')
plt.title('Boîtes à moustaches')
plt.show()
```



Application du modèle $ANOVA$:

```
anova = smf.ols('MonthlyIncome~BusinessTravel', data=df).fit
    ()
sm.stats.anova_lm(anova)
```

Résultat :

	df	sum_sq	mean_sq	F	PR(>F)
BusinessTravel	2.0	2.526641e+07	1.263320e+07	0.75228	0.471486
Residual	1368.0	2.297314e+10	1.679323e+07	NaN	NaN

► Interprétation du résultat obtenu :
Soit H_0 et H_1 tel que :
- (H_0) : Les moyennes des salaires mensuels pour les trois fréquences de voyages, sont égales.
- (H_1) : Il existe au moins une fréquence de voyages qui permet d'obtenir un salaire différent des autres fréquences.

• La `p-value` correspond au facteur `BusinessTravel` (la fréquence des voyages professionnels) égale à `0.471`, qui est supérieure au seuil de `0.05`, Donc on ne peut pas rejeter H_0.
On conclut qu'il n'y a pas de preuve suffisante pour dire que la fréquence des voyages(pas de voyages, rarement, fréquemment) a un effet sur le salaire des employées .

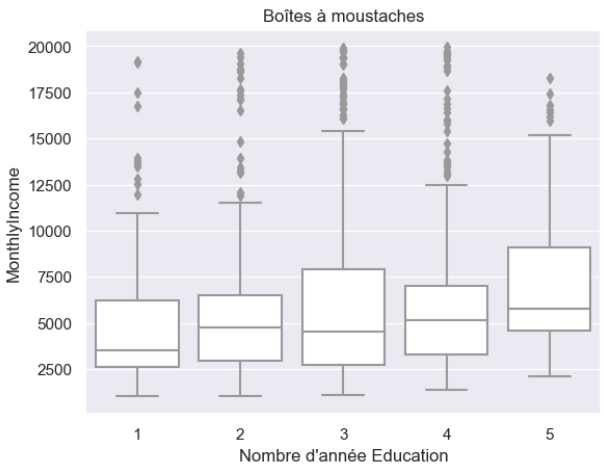
3.8 La spécialité du diplôme et le nombre d'années d'études affectent-ils le salaire des employés ?

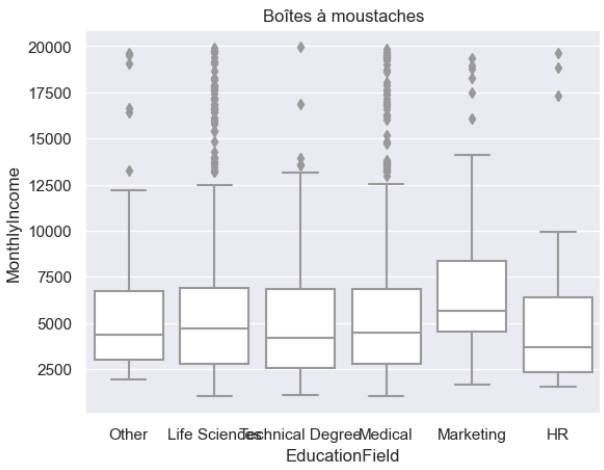
Pour répondre à cette question, On va appliquer une méthode dite $ANOVA$ à deux facteurs, les facteurs sont *EducationField* et *Education* :
Définition du modèle :
Le modèle $ANOVA$ à deux facteurs est utilisé pour examiner les effets de deux facteurs sur une variable dépendante y, ainsi que l'interaction entre ces deux facteurs.
L'équation générale du modèle ANOVA à deux facteurs peut être formulée comme suit :

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}$$

Où:
- y_{ijk} est la valeur observée pour l'observation k dans le groupe i du facteur A et le groupe j du facteur B.
- μ est la moyenne globale de toutes les observations.
- α_i est l'effet du niveau i du facteur A.
- β_j est l'effet du niveau j du facteur B.
- $(\alpha\beta)_{ij}$ est l'effet d'interaction entre le niveau i du facteur A et le niveau j du facteur B.
- ε_{ijk} est l'erreur aléatoire associée à l'observation k dans le groupe i du facteur A et le groupe j du facteur B.

Les deux **BoxPlot** sont comme suit :





Application du modèle $ANOVA$ sans interaction :

```
#SANS INTERACTION
anova2 = smf.ols('MonthlyIncome~Education+EducationField',
    data=df).fit()
sm.stats.anova_lm(anova2)
```

Résultat :

	df	sum_sq	mean_sq	F	PR(>F)
EducationField	5.0	1.104156e+08	2.208311e+07	1.331988	0.247914
Education	1.0	2.741394e+08	2.741394e+08	16.535270	0.000050
Residual	1364.0	2.261385e+10	1.657907e+07	NaN	NaN

Application du modèle $ANOVA$ avec interaction :

```
#AVEC INTERACTION
anova2 = smf.ols('MonthlyIncome~Education*EducationField',
    data=df).fit()
sm.stats.anova_lm(anova2)
```

Résultat :

	df	sum_sq	mean_sq	F	PR(>F)
EducationField	5.0	1.104156e+08	2.208311e+07	1.333656	0.247235
Education	1.0	2.741394e+08	2.741394e+08	16.555986	0.000050
Education:EducationField	5.0	1.110870e+08	2.221740e+07	1.341766	0.243948
Residual	1359.0	2.250276e+10	1.655832e+07	NaN	NaN

► **Interprétation du résultat obtenu :**

Soit les hypothèses suivants :

1. Effet principal du facteur `EducationField`:

- (H_0) : Les moyennes des salaires mensuels pour les différents domaines d'éducation, sont égales.
- (H_1) : Il existe au moins un *domaine d'éducation* qui est bien payé dans cette entreprise.

2. Effet principal du facteur `Education` :

- (H_0) : Les moyennes des salaires mensuels pour les différents "années d'éducation", sont égales.
- (H_1) : Il existe au moins un "*nombre d'année d'éducation*" qui permet d'obtenir un salaire différent (en moyenne).

3. Interaction entre les facteurs `EducationField` et `Education` :

- H_0 : Il n'y a pas d'effet par interaction entre le *domaine d'éducation* et le *nombre d'année d'éducation* .

- H_1 : Il y a un effet par interaction entre les deux facteurs .

• La `p-value` correspond au facteur `EducationField` (la spécialité du diplôme) égale à `0.247`, qui est supérieure au seuil de `0.05`, Donc cela signifie qu'il n'y a pas de différence significative dans les salaires mensuels moyens entre les différents spécialités d'éducation.

• La `p-value` correspond au facteur `Education` (Le nombre d'années d'éducation) égale à `0.00005`, qui est inférieure au seuil de `0.05`, cela signifie qu'il y a une différence significative dans les salaires mensuels moyens en fonction du nombre d'année étudiées .

• La `p-value` correspond au facteur `Education:EducationField` égale à `0.2439`, qui est supérieure au seuil de `0.05`, ce qui signifie qu'il n'y a pas d'interaction significative entre le *domaine d'éducation* et le *nombre d'années d'éducation* sur les salaires mensuels .

Conclusion finale : Les différences de salaires mensuels sont principalement influencées par le *nombre d'années d'éducation*, et non pas par le *domaine d'éducation* ni par une *interaction* entre les deux.

3.9 Visualisation des employées

.....LA METHODE ACP.....
On utilise seulement les colonnes qui contient des variables quantitatives :

```
#Suppression de la colonne
SUPP = ['Attrition(Exit or Not)','BusinessTravel','Department
    ','Education','EducationField','EmployeeNumber(id)','
    EnvironmentSatisfaction','Gender','JobInvolvement','
    JobRole','JobLevel','JobSatisfaction','MaritalStatus','
    OverTime','PerformanceRating','RelationshipSatisfaction','
    StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
    ','YearsSinceLastPromotion','YearsWithCurrManager']
df_ACP = df.drop(SUPP , axis = 1)
df_ACP
```

Application de l'ACP :

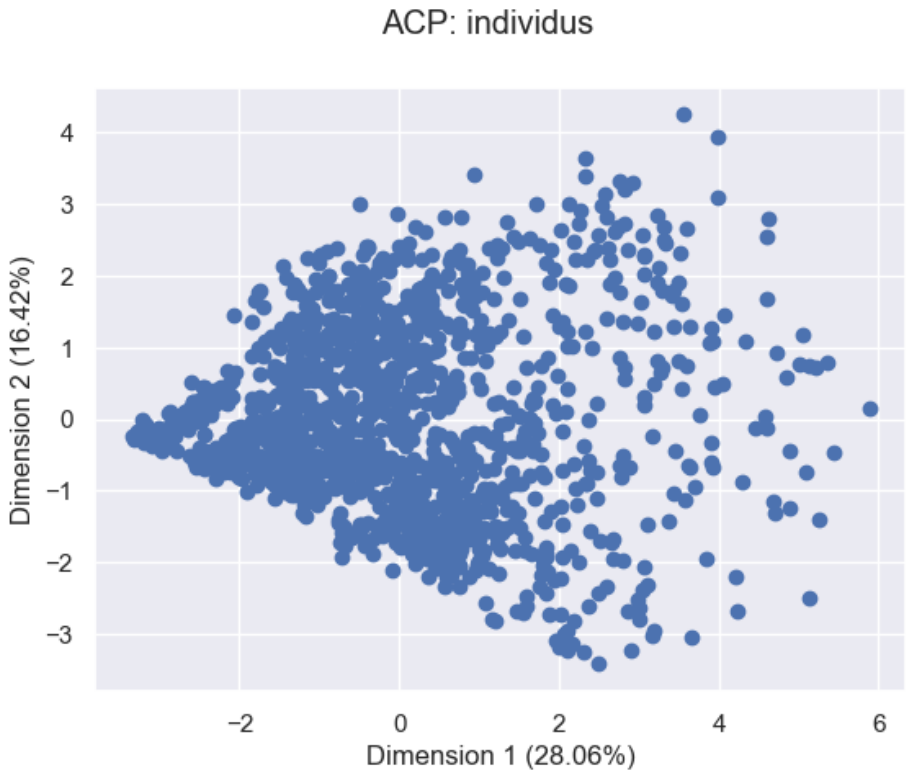
```
#classe pour standardisation
from sklearn.preprocessing import StandardScaler
#instanciation : CREER L'OBJET POUR ACCES A LA METHODE
sc = StandardScaler()
#transformation _centrage_réduction
Z = sc.fit_transform(df_ACP) # Z EST CENTRE REDUIT
```

```
#Matrice de nuage
#pd.plotting.scatter_matrix(df)

#classe pour l'ACP
from sklearn.decomposition import PCA
#instanciation
acp = PCA() #LE CONSTRUCTEUR

#Coordonées dans le nouvel espace
coord = acp.fit_transform(Z)
import matplotlib.pyplot as plt
#print(coord)
acp.explained_variance_ratio_
```

```
array([0.28063503, 0.16422756, 0.10619196, 0.10156009, 0.09994492,
       0.09179559, 0.07126788, 0.05181812, 0.01992205, 0.01263679])
```

```
# plotting scatter plot
import numpy as np
plt.scatter(coord[:,0],coord[:,1])

#pourcentage des axes
pr1=acp.explained_variance_ratio_[0]/np.sum(acp.
    explained_variance_ratio_)
pr1=round(pr1*100,2)
pr2=acp.explained_variance_ratio_[1]/np.sum(acp.
    explained_variance_ratio_)
pr2=round(pr2*100,2)


plt.xlabel("Dimension 1 ({}%)".format(pr1)) # modification du
    nom de l'axe X
plt.ylabel("Dimension 2 ({}%)".format(pr2)) # idem pour axe Y
plt.suptitle("ACP: individus") # titre général
plt.show()
```

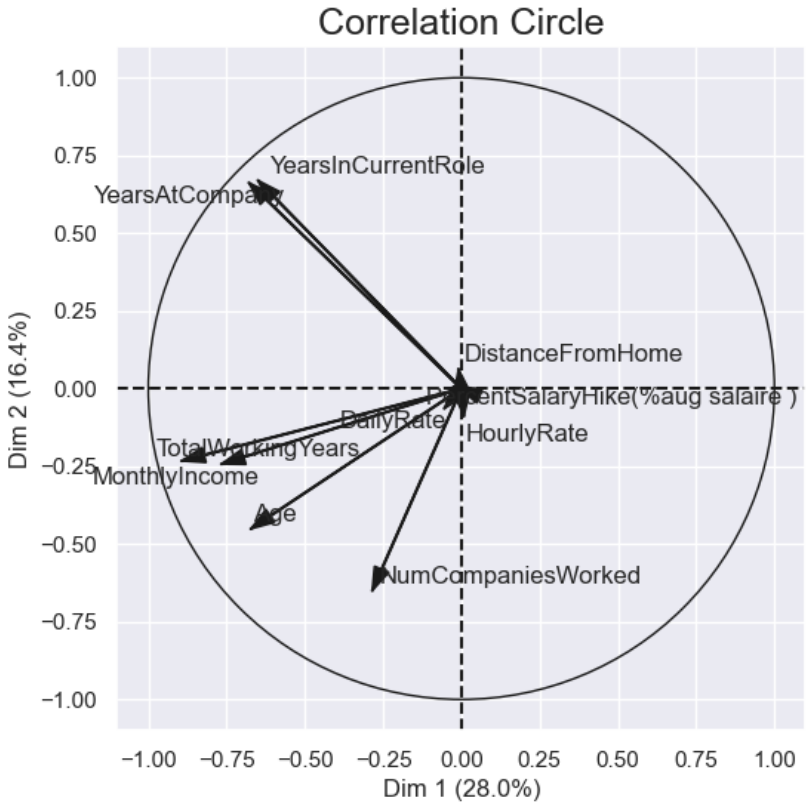
Résultat :



Cercle de corrélation :

```
from mlxtend.plotting import plot_pca_correlation_graph
figure, correlation_matrix = plot_pca_correlation_graph(Z,

    df_ACP.columns.values,

    dimensions=(1, 2),

    figure_axis_size=6)
```

Résultat :



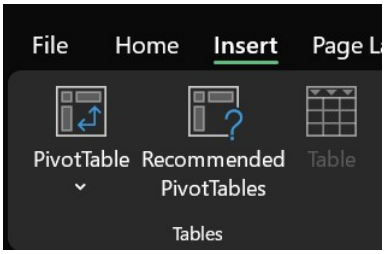
Interprétation du cercle de corrélation :

.....

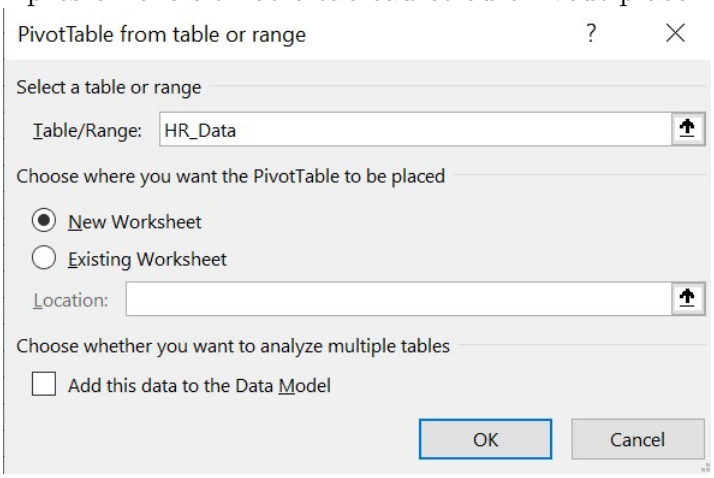
.....

3.10 Dans cette entreprise, les femmes sont plus satisfaites que les hommes ?

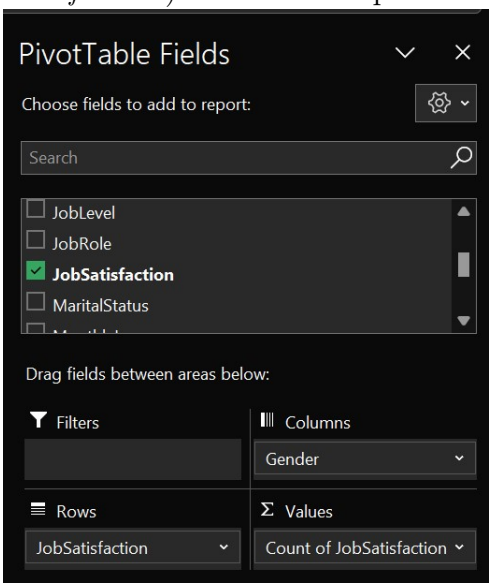
Pour répondre a cette question, On va appliquer une méthode dite **Analyse Factorielle des Correspondances(AFC)** .
C'est une technique d'analyse des données utilisée principalement pour explorer et visualiser les relations entre des catégories de variables qualitatives. C'est une méthode particulièrement utile pour **les tableaux de contingence**, qui sont des matrices de fréquences de deux variables catégorielles.
Pour appliquer l'AFC dans notre cas, nous sommes besoin d'organiser les données sous forme de tableau de contingence, pour cela, on va utiliser une outil dans `Excel` dite **Tableau croisé dynamique** (ou **Pivot Table**).
Après la sélection de notre tableau, On peut trouver l'outil **PivotTable** dans la rubrique `Insert`



Après on choisit notre tableau et où on veut placer notre tableau croisé



Pour obtenir un tableau qui contient la variable `Gender` en colonne et le `niveau de satisfaction` en lignes, on place chaque variable dans sa rubrique. on place la variable *Gender* à `Columns`, et la variable *JobSatisfaction* à `Rows`, et les fréquences d'apparition de chaque niveau de satisfaction (*Count of JobSatisfaction*) à les valeurs que doit contenir le tableau (`Values`) :



Le résultat est comme suit :

	A	B	C	D
3	Count of JobSatisfaction			
4		Female	Male	Grand Total
5	1	112	157	269
6	2	110	148	258
7	3	168	242	410
8	4	158	276	434
9	Grand Total	548	823	1371

Après l'organisation du tableau, on le place dans une nouveau feuille pour l'importer sur `Python`, le tableau finale est comme suit :

	A	B	C
1	Count of JobSatisfaction	Female	Male
2	1	112	157
3	2	110	148
4	3	168	242
5	4	158	276
6			

```
import pandas as pd
from fanalysis.ca import CA
import numpy as np
import matplotlib.pyplot as plt
AFC1 = pd.read_excel("HR_DATA_APP.xlsx", sheet_name="AFC1") #
   On choisit la feuille qui contient le tableau
AFC1
```

Résultat :

	Count of JobSatisfaction	Female	Male
0	1	112	157
1	2	110	148
2	3	168	242
3	4	158	276

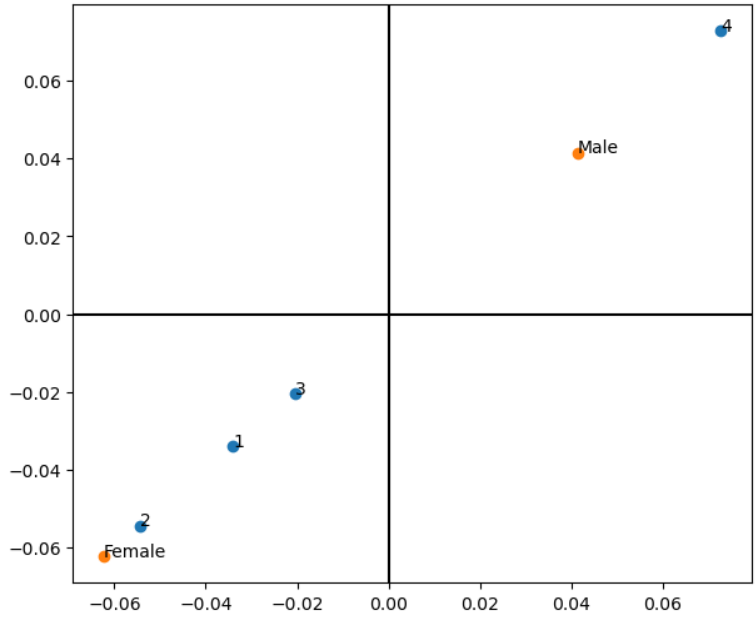
On utilise le `Count of JobSatisfaction` comme nouvel index :

```
AFC1=AFC1.set_index("Count of JobSatisfaction")
AFC1
```

	Female	Male
Count of JobSatisfaction		
1	112	157
2	110	148
3	168	242
4	158	276

On applique la méthode **AFC** et on trace le **Biplot** :

```
afc=CA(row_labels=AFC1.index.values, col_labels=AFC1.columns.
    values, stats=True)
afc.fit(AFC1.values)

# Extraire les coordonnées des lignes et des colonnes
row_coords = afc.row_coord_
col_coords = afc.col_coord_

# Extraire les variances expliquées par chaque dimension
eigenvalues = afc.eig_
total_variance = sum(eigenvalues)
var_explained = [eigenvalue / total_variance for eigenvalue
    in eigenvalues]

# Créer le biplot
fig, ax = plt.subplots(figsize=(7, 6))

# Tracer les coordonnées des lignes
ax.scatter(row_coords[:, 0], row_coords[:, 0], label='Rows')
for i, (x, y) in enumerate(zip(row_coords[:, 0], row_coords
    [:, 0])):
     ax.text(x, y, AFC1.index.values[i])

# Tracer les coordonnées des colonnes
ax.scatter(col_coords[:, 0], col_coords[:, 0], label='Columns
    ')
for i, (x, y) in enumerate(zip(col_coords[:, 0], col_coords
    [:, 0])):
     ax.text(x, y, AFC1.columns.values[i])

total_variance = sum(eigenvalues)
var_explained = [eigenvalue / total_variance for eigenvalue
    in eigenvalues]

ax.axhline(y=0, color='k')  # Axe horizontal
ax.axvline(x=0, color='k')  # Axe vertical
# Afficher le biplot
plt.show()
```



Interprétation du résultat du Biplot :

• Les points sur le biplot représentent les catégories de nos variables, à savoir les niveaux de satisfaction (1, 2, 3, 4) et les genres (homme, femme).

• Les points qui sont proches les uns des autres indiquent une association forte entre ces catégories.

► *Male* est proche du point représentant "4" (satisfaction élevée),
en plus, les points représentant "homme" et "4" (satisfaction élevée) forment un angle aigu, cela suggère que les hommes tendent à être plus satisfaits au travail.
► *Female* est proche du point représentant "2" (satisfaction plus basse), cela indique une satisfaction moindre pour les femmes.
► La proximité à l'origine des *niveaux de satisfaction 3 et 1*, indique qu'ils ne sont pas fortement associés à un genre spécifique (ni particulièrement aux hommes, ni particulièrement aux femmes), C'est-à-dire qu'ils sont répartis équitablement entre les hommes et les femmes.

3.11 Travailler dans un poste spécifique, dans un département, est liée au spécialité de diplôme obtenue?

Pour répondre a cette question, On va appliquer une méthode dite : **Analyse des Correspondances Multiples (ACM)**.
L'**ACM** est une extension de l'AFC, qui est utilisée pour analyser les relations entre deux variables catégorielles. L'**ACM** permet d'analyser les relations entre *plusieurs* variables catégorielles simultanément.
Application de l'ACM :

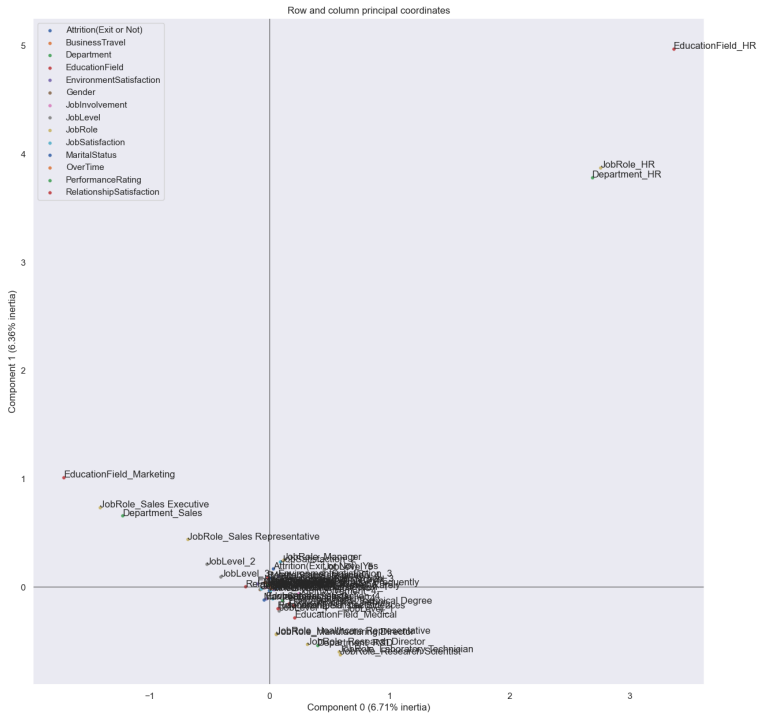
```
#Supression des colonnes inutiles
SUPP_ACM1 = [ 'Age','DailyRate','DistanceFromHome','Education
    ','EmployeeNumber(id)','HourlyRate','MonthlyIncome','
    NumCompaniesWorked','PercentSalaryHike(%aug salaire )','
    StockOptionLevel','TotalWorkingYears','
    TrainingTimesLastYear','WorkLifeBalance','YearsAtCompany',
    'YearsInCurrentRole','YearsSinceLastPromotion','
    YearsWithCurrManager']
df_ACM = df.drop(SUPP_ACM1 , axis = 1)
df_ACM.head()
```

Traçage du Biplot de l'ACM (ou Plan factoriel):

```
import prince
import matplotlib.pyplot as plt
from prince import plot

#Création de l'objet mca
mca = prince.MCA()
#Lancement de l'ACM sur le jeu de données
mca=mca.fit(df_ACM)


# Création de la figure

fig, ax = plt.subplots(figsize=(15, 15))

#Tracage des axes
ax = plot.stylize_axis(ax)


#Profils colonnes
col_coords = mca.column_coordinates(df_ACM)
x = col_coords[0]
```

```
y = col_coords[1]

#Extraction des préfixes à l'aide map et de lambda
prefixes = col_coords.index.str.split('_').map(lambda x: x
    [0])

#Affichage des profils colonnes avec leurs modalités
for prefix in prefixes.unique():
    mask = prefixes == prefix

    ax.scatter(x[mask], y[mask], s=10, label=prefix)

    for i, label in enumerate(col_coords[mask].index):
        ax.annotate(label, (x[mask][i], y[mask][i]))

#Ajout de légende
ax.legend()

ax.set_title('Row and column principal coordinates')
ei = [eig / mca.total_inertia_ for eig in mca.eigenvalues_]
ax.set_xlabel('Component {} ({:.2f}% inertia)'.format(0, 100
    * ei[0]))
ax.set_ylabel('Component {} ({:.2f}% inertia)'.format(1, 100
    * ei[1]))
```

Résultats :



Interprétation du résultat du Biplot :

Les axes du biplot représentent les composantes principales de l'*ACM*, qui sont des combinaisons linéaires des variables d'origine.
On visualise les deux premières composantes principales (Dim 1 et Dim 2), car elles expliquent la plus grande part de la variance des données.

• Les points proches les uns des autres indiquent des individus ayant des profils similaires.

▶ Le *JobRole_HR* est proche de *Education_Field_HR* et du *Department_HR*. Cela suggère une proximité ou une similarité entre ces catégories.
Cela est logique car le rôle "HR" est probablement lié au département des ressources humaines, qui est occupé par des employés diplômés dans le domaine des ressources humaines.
Conclusion finale : Travailler dans un poste spécifique dans un département, est liée au spécialité du diplôme obtenue .

3.12 Quel niveau est requis pour occuper le poste de Manager ?

Pour répondre à cette question, appliquons une nouvelle fois la méthode **Analyse des Correspondances Multiples (ACM)**.

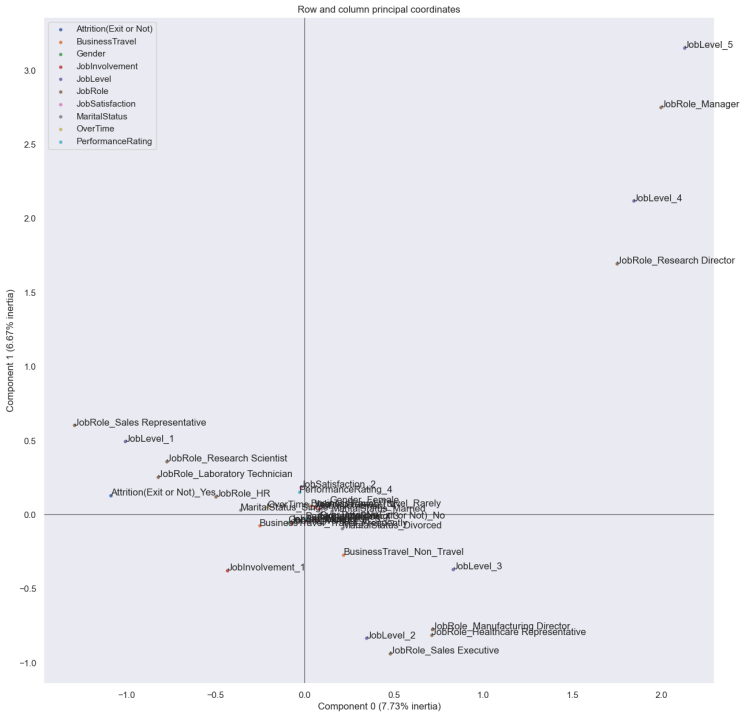
```
# Suppression des colonnes inutiles
SUPP_ACM2 = [ 'Department', 'EducationField',  '
   EnvironmentSatisfaction' ,'RelationshipSatisfaction'   ,'
   Age','DailyRate','DistanceFromHome','Education','
   EmployeeNumber(id)','HourlyRate','MonthlyIncome','
   NumCompaniesWorked','PercentSalaryHike(%aug salaire )','
   StockOptionLevel','TotalWorkingYears','
   TrainingTimesLastYear','WorkLifeBalance','YearsAtCompany',
   'YearsInCurrentRole','YearsSinceLastPromotion','
   YearsWithCurrManager']
df_ACM = df.drop(SUPP_ACM2 , axis = 1)
df_ACM.head()
```

Application de la méthode et Traçage du Biplot de l'ACM (ou Plan factoriel):

```
#Création de l'objet mca
mca = prince.MCA()
#Lancement de l'ACM sur le jeu de données
mca=mca.fit(df_ACM)
```

```
import prince
import matplotlib.pyplot as plt
from prince import plot
# Création de la figure

fig, ax = plt.subplots(figsize=(15, 15))

#Tracage des axes
ax = plot.stylize_axis(ax)


#Profils colonnes
col_coords = mca.column_coordinates(df_ACM)
x = col_coords[0]
y = col_coords[1]

#Extraction des préfixes à l'aide map et de lambda
prefixes = col_coords.index.str.split('_').map(lambda x: x
    [0])

#Affichage des profils colonnes avec leurs modalités
for prefix in prefixes.unique():
    mask = prefixes == prefix

    ax.scatter(x[mask], y[mask], s=10, label=prefix)

    for i, label in enumerate(col_coords[mask].index):
        ax.annotate(label, (x[mask][i], y[mask][i]))

#Ajout de légende
ax.legend()



ax.set_title('Row and column principal coordinates')
ei = [eig / mca.total_inertia_ for eig in mca.eigenvalues_]
ax.set_xlabel('Component {} ({:.2f}% inertia)'.format(0, 100
    * ei[0]))
ax.set_ylabel('Component {} ({:.2f}% inertia)'.format(1, 100
    * ei[1]))
```

Résultats :



Interprétation du résultat :
Un Biplot permet de visualiser les relations entre différentes modalités de variables catégorielles.
Dans notre Biplot on observe que :
JobLevel_5 est proche de *JobRole_Manager*, cela indique qu'il y a une forte association entre ces deux modalités.

Donc on peut conclure que :
► Les personnes au niveau d'emploi 5 (*JobLevel 5*) sont souvent des managers (*JobRole Manager*). Cela signifie que les postes de manager sont occupés par des employées qui ont un échelle 5 (*JobLevel=5*).

Conclusion : Le niveau d'emploi(*JobLevel*) requis pour occuper un poste de Manager est le niveau **5** .

3.13 Quel niveau est requis pour occuper le poste de Directeur de Recherche ?

Pour répondre à cette question, appliquons une autre fois la méthode **Analyse des Correspondances Multiples (ACM)**.
Application de l'ACM :

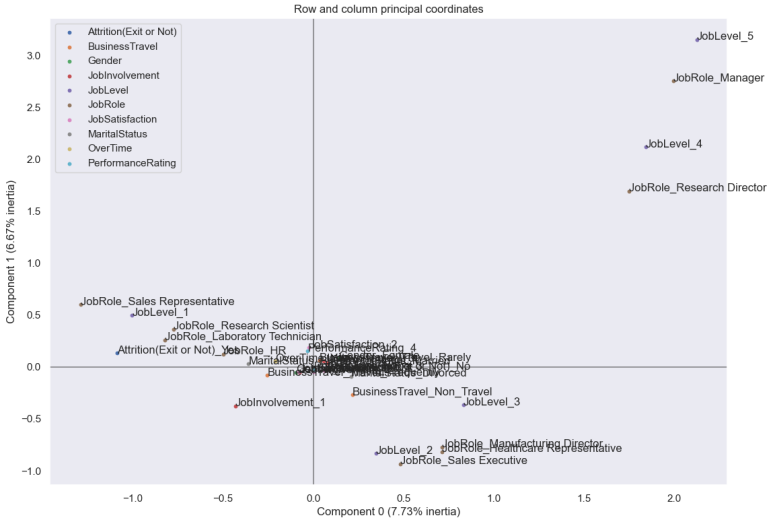
```
# Suppression des colonnes inutiles
SUPP_ACM2 = [ 'Department', 'EducationField',  '
   EnvironmentSatisfaction' ,'RelationshipSatisfaction'   ,'
   Age','DailyRate','DistanceFromHome','Education','
   EmployeeNumber(id)','HourlyRate','MonthlyIncome','
   NumCompaniesWorked','PercentSalaryHike(%aug salaire )','
   StockOptionLevel','TotalWorkingYears','
   TrainingTimesLastYear','WorkLifeBalance','YearsAtCompany',
   'YearsInCurrentRole','YearsSinceLastPromotion','
   YearsWithCurrManager']
df_ACM = df.drop(SUPP_ACM2 , axis = 1)
df_ACM.head()
```

Traçage du Biplot de l'ACM (ou Plan factoriel):

```
#Création de l'objet mca
mca = prince.MCA()
#Lancement de l'ACM sur le jeu de données
mca=mca.fit(df_ACM)

import prince
import matplotlib.pyplot as plt
from prince import plot
# Création de la figure

fig, ax = plt.subplots(figsize=(15, 15))

#Tracage des axes
ax = plot.stylize_axis(ax)


#Profils colonnes
col_coords = mca.column_coordinates(df_ACM)
x = col_coords[0]
y = col_coords[1]

#Extraction des préfixes à l'aide map et de lambda
prefixes = col_coords.index.str.split('_').map(lambda x: x
   [0])

#Affichage des profils colonnes avec leurs modalités
for prefix in prefixes.unique():
    mask = prefixes == prefix

    ax.scatter(x[mask], y[mask], s=10, label=prefix)

    for i, label in enumerate(col_coords[mask].index):
```

```
        ax.annotate(label, (x[mask][i], y[mask][i]))

#Ajout de légende
ax.legend()



ax.set_title('Row and column principal coordinates')
ei = [eig / mca.total_inertia_ for eig in mca.eigenvalues_]
ax.set_xlabel('Component {} ({:.2f}% inertia)'.format(0, 100
    * ei[0]))
ax.set_ylabel('Component {} ({:.2f}% inertia)'.format(1, 100
    * ei[1]))
```

Résultats :



Interprétation du résultat :

Dans notre Biplot on observe que :
JobLevel_4 est proche de *JobRole_Research Director*, cela indique qu'il y a une forte association entre ces deux modalités.

Donc on peut conclure que :
► Les personnes au niveau d'emploi 4 (*JobLevel 4*) sont souvent des Directeurs de recherches (*JobRole Research Director*). Cela signifie que le poste de directeur de recherche est souvent occupé par des employées qui ont un échelle 4 (*JobLevel=4*).

Conclusion : Le niveau d'emploi(*JobLevel*) requis pour occuper un poste de Directeur de Recherche est le niveau **4** .

3.14 Classification des employées

3.14.1 Classification par K-means

K-means est un algorithme d'apprentissage non supervisé utilisé pour partitionner un ensemble de données en K clusters (groupes), Où chaque point de données appartient au cluster avec le centroïde le plus proche. C'est l'un des algorithmes de clustering les plus simples et les plus utilisés.

L'objectif de K-means est de *minimiser la variance intra-cluster* (la somme des distances au carré entre les points de données et leur centroïde de cluster) et de *maximiser la variance inter-cluster* (la distance entre différents centroïdes de clusters).

Application de l'ACP avant K-means :

```
#Suppression des colonnes inutiles
SUPP = ['Attrition(Exit or Not)','BusinessTravel','Department
    ','Education','EducationField','EmployeeNumber(id)','
    EnvironmentSatisfaction','Gender','JobInvolvement','
    JobRole','JobLevel','JobSatisfaction','MaritalStatus','
    OverTime','PerformanceRating','RelationshipSatisfaction','
    StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
    ','YearsSinceLastPromotion','YearsWithCurrManager']
df_ACP = df.drop(SUPP , axis = 1)
df_ACP
#classe pour standardisation
from sklearn.preprocessing import StandardScaler
#instanciation
sc = StandardScaler()
#transformation_centrage_réduction
Z = sc.fit_transform(df_ACP) # Z EST CENTRE REDUIT

#classe pour l'ACP
from sklearn.decomposition import PCA
#instanciation
acp = PCA() #LE CONSTRUCTEUR

#Coordonées dans le nouvel espace
coord = acp.fit_transform(Z)
import matplotlib.pyplot as plt
#print(coord)
acp.explained_variance_ratio_
```

Application de K-means :

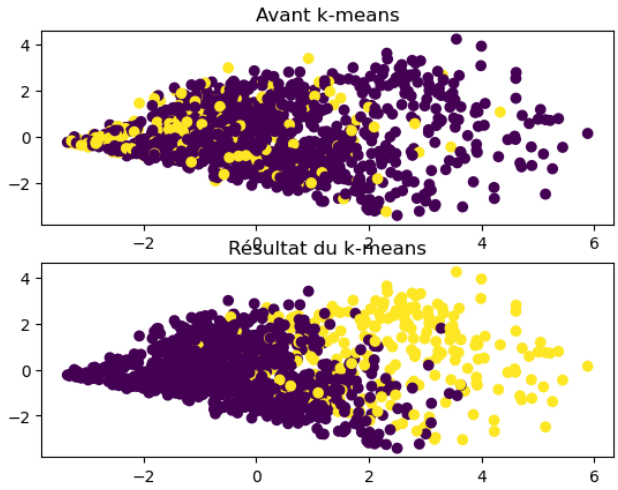
```
#Suppression des colonnes inutiles
```

```
SUPP = ['Attrition(Exit or Not)','BusinessTravel','Department
    ','Education','EducationField','EmployeeNumber(id)','
    EnvironmentSatisfaction','Gender','JobInvolvement','
    JobRole','JobLevel','JobSatisfaction','MaritalStatus','
    OverTime','PerformanceRating','RelationshipSatisfaction','
    StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
    ','YearsSinceLastPromotion','YearsWithCurrManager']
df_CLF = df.drop(SUPP , axis = 1)
df_CLF


from sklearn.cluster import KMeans

observations = df_CLF

nb_groupes = 2

kmeans = KMeans(n_clusters= nb_groupes , random_state=0).fit(
    observations)

print(kmeans.cluster_centers_)

# Affichage des résultats avant k-means
    ----------------------------------------------

import matplotlib.pyplot as plt

df['Attrition(Exit or Not)'] = df['Attrition(Exit or Not)'].
    astype('category')

plt.subplot(211) ; plt.title("Avant k-means")


plt.scatter(coord[:, 0],coord[:, 1],

           c=df['Attrition(Exit or Not)'].cat.codes) ;

# Affichage des résultats du k-means
    ---------------------------------------------------

plt.subplot(212) ; plt.title("Résultat du k-means")

centers = pd.DataFrame(kmeans.cluster_centers_)
centers.columns = observations.columns ;

plt.scatter(coord[:, 0],coord[:, 1],  c=kmeans.labels_ ) ; #
    ON UTILISE kmeans.labels_

plt.show()
```

Résultats :

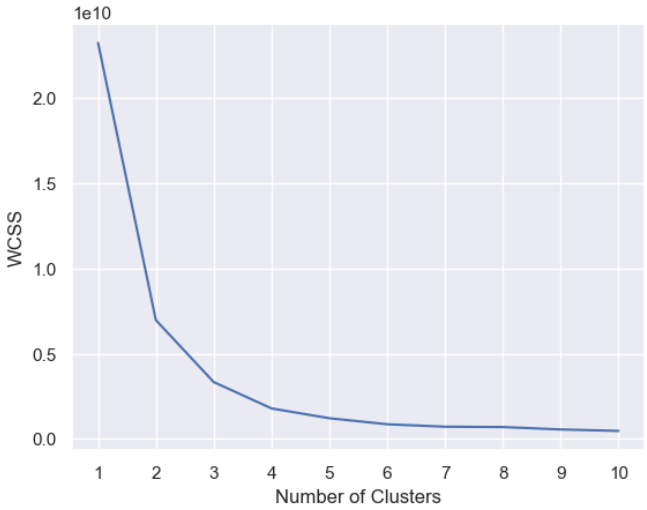
```
[[3.46839709e+01 7.98502732e+02 9.07559199e+00 6.61202186e+01
  4.18561475e+03 2.53825137e+00 1.52276867e+01 8.21129326e+00
  5.32969035e+00 3.55009107e+00]
 [4.26593407e+01 8.36912088e+02 9.61172161e+00 6.52637363e+01
  1.28076740e+04 3.47252747e+00 1.53516484e+01 1.83992674e+01
  7.68864469e+00 5.22344322e+00]]
```

Résultats :



Nombre de clusters optimales :
En traçant *WCSS* en fonction du nombre de clusters k, on obtient :

```
#create list to hold SSE values for each k
# WCSS - the sum of square distances between the centroids
    and each points.
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(observations)
    wcss.append(kmeans.inertia_)

#visualize results
plt.plot(range(1, 11), wcss)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```



• On peut observer comment la variation intra-cluster diminue à mesure que le nombre de clusters augmente.
• Le **coude** dans la courbe *WCSS* vs. *Nombre de clusters* indique le point où l'ajout de clusters supplémentaires n'entraîne plus une réduction significative de WCSS.
• Ce point optimal(*Number of Clusters=2*) est choisi comme le nombre approprié de clusters pour notre algorithme de *k-means*.

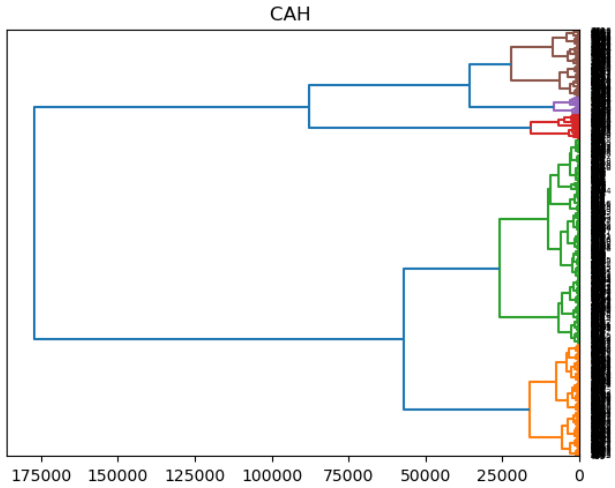
3.14.2 Classification par CAH

La Classification Ascendante Hiérarchique (CAH), est une méthode de clustering qui vise à créer une hiérarchie de clusters. Contrairement à *K-means*, qui nécessite de spécifier le nombre de clusters à l'avance, la *CAH* construit une hiérarchie complète qui peut être coupée à différents niveaux pour obtenir des clusters .
Dans notre cas, on va utilisé une méthode de mesure de distance entre clusters dite : **La méthode de Ward .**
Dans *la méthode de Ward*, la fusion des clusters est basée sur la minimisation de l'augmentation de la somme des carrés des erreurs (variance intra-cluster).
Tracage du dendrogramme :

```
#Suppression des colonnes inutiles
SUPP = ['Attrition(Exit or Not)','BusinessTravel','Department
    ','Education','EducationField','EmployeeNumber(id)','
    EnvironmentSatisfaction','Gender','JobInvolvement','
    JobRole','JobLevel','JobSatisfaction','MaritalStatus','
    OverTime','PerformanceRating','RelationshipSatisfaction','
```

```
    StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
    ','YearsSinceLastPromotion','YearsWithCurrManager']
df_CAH = df.drop(SUPP , axis = 1)
df_CAH

from matplotlib import pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage,
    fcluster

#Matrice des distances

Z = linkage(df_CAH ,method='ward', metric='euclidean') ###

#affichage du dendrogramme
plt.title('CAH ')

dendrogram(Z,labels=df_CAH.index,orientation='left',
    color_threshold=30000) #on fixe un seuil

plt.show()
```

Résultats :

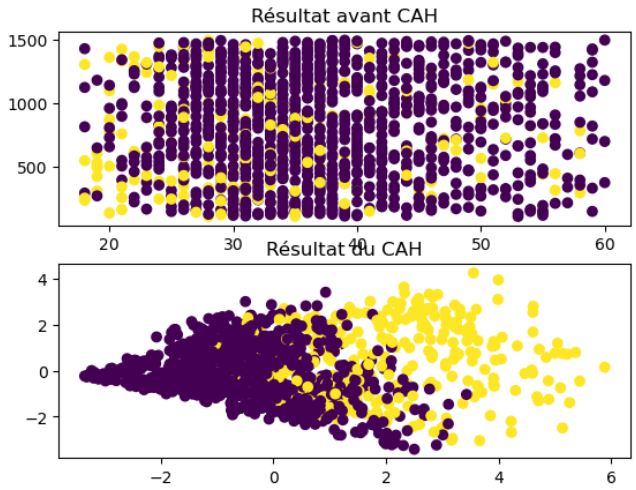


▶ Le *dendrogramme* est une représentation visuelle de la hiérarchie des clusters. En coupant le *dendrogramme* à différents niveaux, on peut obtenir des clusters de différentes tailles.
Si on veut fixer le nombre de clusters à 2, on coupe le *dendrogramme* à t=100000 :

```
df['Attrition(Exit or Not)'] = df['Attrition(Exit or Not)'].
    astype('category')
```

```
plt.subplot(211) ; plt.title("Résultat avant CAH")

plt.scatter(df_CAH['Age'],df_CAH['DailyRate'],

           c=df['Attrition(Exit or Not)'].cat.codes) ;


groupes_cah = fcluster(Z,t=100000,criterion='distance')

# Affichage des résultats du CAH

plt.subplot(212) ; plt.title("Résultat du CAH")

plt.scatter(coord[:, 0],coord[:, 1], c=groupes_cah) ;

plt.show()
```

Résultats :



3.15 Classification des employées par l'Analyse discriminante AFD

L'Analyse Factorielle Discriminante (AFD) est une méthode utilisée pour classer des observations dans des groupes en fonction de leurs caractéristiques mesurées.
Elle *cherche à trouver une combinaison linéaire des variables originales* qui *maximise la séparation entre les groupes* tout en *minimisant la variation à l'intérieur des groupes.*

Application de l'AFD :

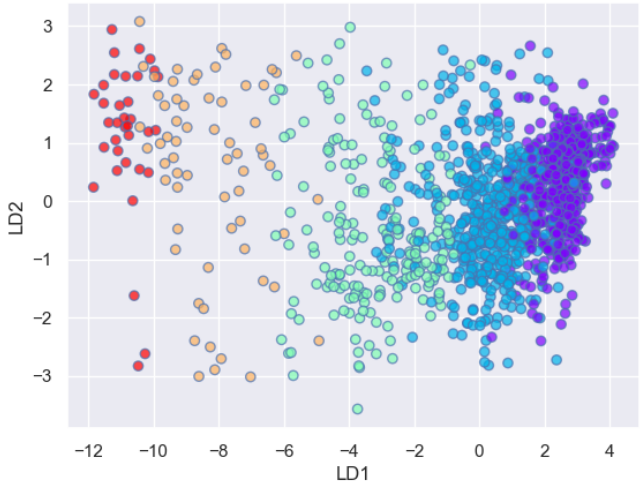
```
from sklearn import datasets
import pandas as pd
import numpy as np
np.set_printoptions(precision=4)
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
from sklearn.preprocessing import LabelEncoder


#Suppression des colonnes inutiles
SUPP = ['Attrition(Exit or Not)','BusinessTravel','Department
    ','Education','EducationField','EmployeeNumber(id)','
    EnvironmentSatisfaction','Gender','JobInvolvement','
    JobRole','JobLevel','JobSatisfaction','MaritalStatus','
    OverTime','PerformanceRating','RelationshipSatisfaction','
    StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
    ','YearsSinceLastPromotion','YearsWithCurrManager']
df_AFD = df.drop(SUPP , axis = 1)
df_AFD


#Variables...
X = df_AFD
Y = df['JobLevel']
df_AFD2 = X.join(pd.Series( Y , name='class'))
df_AFD2.head()
```

Si on classe les employés par le `JobLevel`, on obtient :

```
#Codage...
le = LabelEncoder()
y = le.fit_transform(df_AFD2['class']) #utilisé pour le
    scatter plot

from sklearn.discriminant_analysis import
    LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
X_lda = lda.fit_transform(X, Y) #LES VARIABLES(OU COMPOSANTES
    ) DISCRIMINAANTES
lda.explained_variance_ratio_ #LES VALEURS PROPRES
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.scatter(
    X_lda[:,0],
    X_lda[:,1],
    c=y,
    cmap='rainbow',
    alpha=0.7,
```

```
    edgecolors='b'
)
```

Résultats :



▶ Notre scatter plot est créé à partir des deux premières composantes discriminantes de l'*AFD*, ce qui permet de visualiser la séparation des classes dans **un espace de dimension réduite**.

3.16 L'arbre optimal pour savoir si l'employé va quitter l'entreprise ou Non

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,
    DecisionTreeRegressor, plot_tree
from sklearn.metrics import accuracy_score,
    mean_squared_error
import matplotlib.pyplot as plt
```

On choisit 150 observation aléatoire de chacun des deux (150 pour `Yes` et 150 pour `No`)

```
yes_samples = df[df['Attrition(Exit or Not)'] == 'Yes'].
    sample(n=150, random_state=42)
no_samples = df[df['Attrition(Exit or Not)'] == 'No'].sample(
    n=150, random_state=42)

# Concaténer les deux échantillons
```

```
balanced_df = pd.concat([yes_samples, no_samples])

# Charger les données depuis le fichier Excel
df = pd.read_excel('HR_DATA_APP.xlsx', sheet_name='HR Data')

SUPP = [ 'MonthlyIncome', 'NumCompaniesWorked', '
    YearsInCurrentRole', 'DistanceFromHome', 'Age', '
    TotalWorkingYears' ,'DailyRate' ,'HourlyRate'  ,'Attrition
    (Exit or Not)','BusinessTravel','Department','Education','
    EducationField','EmployeeNumber(id)','Gender','
    JobInvolvement','JobRole','JobLevel','MaritalStatus','
    OverTime','PerformanceRating','RelationshipSatisfaction','
    StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
    ','YearsSinceLastPromotion','YearsWithCurrManager']
df_CT = balanced_df.drop(SUPP , axis = 1)

X_clf = df_CT
#.drop(columns=['target'])
y_clf = balanced_df['Attrition(Exit or Not)']
X_clf_train, X_clf_test, y_clf_train, y_clf_test =
    train_test_split(X_clf, y_clf, test_size=0.2, random_state
    =42) #TRAIN CLF
y_clf
X_clf
```

Résultat :

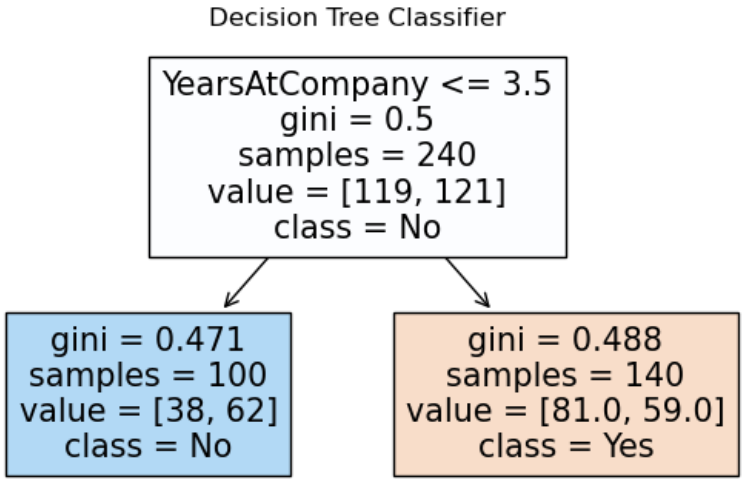
	EnvironmentSatisfaction	JobSatisfaction	PercentSalaryHike(%aug salaire)	YearsAtCompany
25	3	3	12	0
595	1	1	22	5
37	1	1	20	0
480	1	4	15	4
696	3	1	13	5
...	...	...	...	...
1141	4	1	12	10
1189	2	3	17	10
797	2	4	17	6
566	2	3	12	4
373	4	3	11	3

300 rows × 4 columns

```
clf_tree = DecisionTreeClassifier(random_state=42 ,
    min_samples_leaf=75 ) # ON CHOISIT LE MAX DEPTH COMME
    COMPLEXITY
#   , min_samples_leaf=50
```

```
clf_tree.fit(X_clf_train, y_clf_train)
# Plot decision tree for classification
plt.figure(figsize=(7, 4))
plot_tree(clf_tree, filled=True, feature_names=df_CT.columns
    , class_names=np.array([ 'Yes','No' ]))
plt.title("Decision Tree Classifier")
plt.show()
```

Résultat :



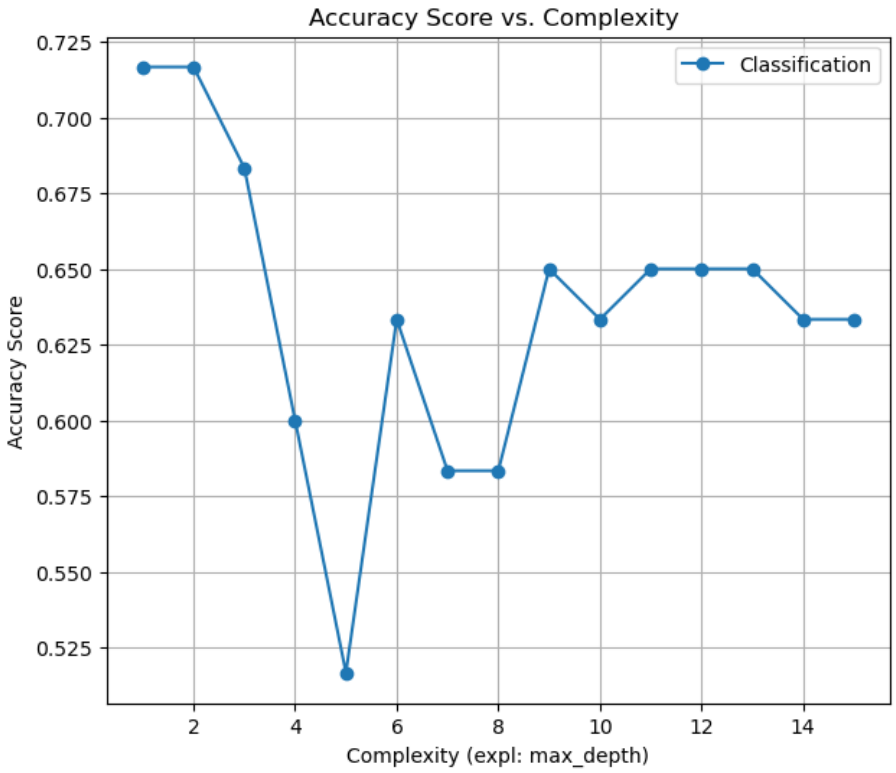
Arbre Optimal pour la classification :

```
# Define a range of hyperparameters to explore
param_range = range(1, 16)  # Example range for max_depth

# Initialize lists to store average decision cost for
   classification and regression
avg_decision_cost_clf = []
# avg_decision_cost_reg = []

# Loop through each hyperparameter value
for param in param_range:
    # Classification
    clf = DecisionTreeClassifier(max_depth=param,
   random_state=42)
    clf.fit(X_clf_train, y_clf_train)
    clf_pred = clf.predict(X_clf_test)
    avg_decision_cost_clf.append(accuracy_score(y_clf_test,
   clf_pred))

# Plot the graph
```

```
plt.figure(figsize=(7, 6))
plt.plot(param_range, avg_decision_cost_clf, label='
    Classification', marker='o')
#plt.plot(param_range, avg_decision_cost_reg, label='
    Regression', marker='o')
plt.xlabel('Complexity (expl: max_depth)')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Score vs. Complexity ')
plt.legend()
plt.grid(True)
plt.show()
```

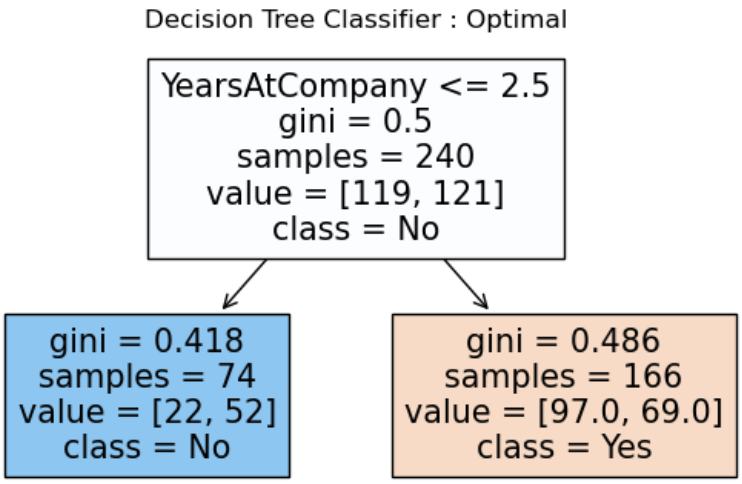
La complexité du modèle peut être mesurée de différentes manières. Cela pourrait être le nombre de nœuds, la profondeur de l'arbre, etc.
Si on choisit le profondeur de l'arbre comme complexité :

La figure ci-dessous montre la relation entre la complexité de l'arbre, reflétée par le profondeur maximal de l'arbre(*max_depth*) et l'*Accuracy score* (le Score de précision) :




```
# Create and train decision tree classifiers
clf_tree = DecisionTreeClassifier(random_state=42, max_depth
    =1 ) # ON FIXE LE MAX DEPTH
```

```
clf_tree.fit(X_clf_train, y_clf_train)
# Plot decision tree for classification
plt.figure(figsize=(7, 4))
plot_tree(clf_tree, filled=True, feature_names=df_CT.columns
    , class_names=np.array([ 'Yes','No' ]))
plt.title("Decision Tree Classifier : Optimal")
plt.show()
```



▶ D'après cette arbre de décision, on peut savoir si l'employé va quitter l'entreprise ou non .

3.17 L'arbre optimal pour prédire le salaire d'un employé

Faisant une arbre de régression pour la variable dépendante `MonthlyIncome`. Et choisissant 200 observations aléatoire pour équilibrer les données :

```
# DATA FOR REGRESSION
# ON CHOISIT 200 POUR CHAQUE CLASSE POUR EQUILIBRER LES
    DONNEES
yes_samples = df[df['Attrition(Exit or Not)'] == 'Yes'].
    sample(n=200, random_state=42)
no_samples = df[df['Attrition(Exit or Not)'] == 'No'].sample(
    n=200, random_state=42)

# Concaténer les deux échantillons
balanced_df = pd.concat([yes_samples, no_samples])
```

```
SUPP = [ 'MonthlyIncome' , 'DailyRate' , 'DistanceFromHome' ,
    'Attrition(Exit or Not)', 'HourlyRate' ,  '
   PercentSalaryHike(%aug salaire )' , 'BusinessTravel','
   Department','Education','EducationField','EmployeeNumber(
   id)','EnvironmentSatisfaction','Gender','JobInvolvement','
   JobRole','JobSatisfaction','MaritalStatus','OverTime','
   PerformanceRating','RelationshipSatisfaction','
   StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
   ','YearsSinceLastPromotion','YearsWithCurrManager']
df_RG = balanced_df.drop(SUPP , axis = 1)

X_reg = df_RG
y_reg = balanced_df['MonthlyIncome']
X_reg_train, X_reg_test, y_reg_train, y_reg_test =
   train_test_split(X_reg, y_reg, test_size=0.2, random_state
   =42)

# Create and train decision tree regressors
reg_tree = DecisionTreeRegressor(random_state=42,
   min_samples_leaf=100  )
# in_samples_leaf : CAD MIN D INDIVIDU SUR DANS FEUILLE(ON L'
   AUGMENTE POUR AVOIR UN NBR MOIN DE FEUILLE)

reg_tree.fit(X_reg_train, y_reg_train)
```

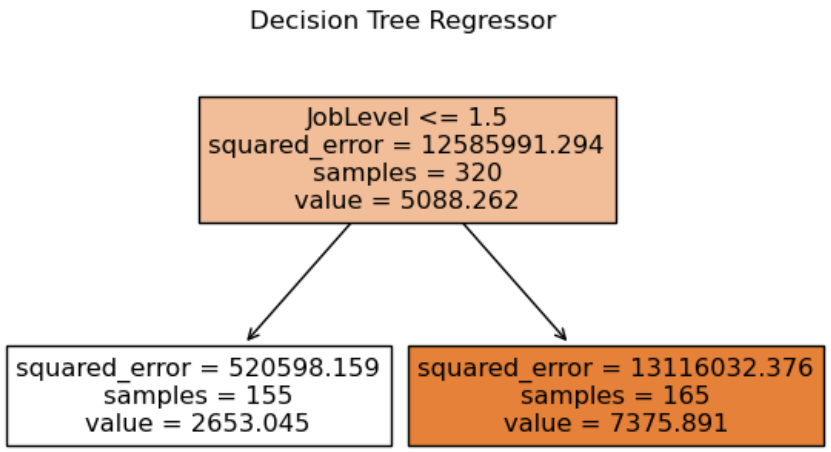
Résultat :

	Age	JobLevel	NumCompaniesWorked	TotalWorkingYears	YearsAtCompany	YearsInCurrentRole
25	18	1	1	0	0	0
595	32	1	7	10	5	4
37	33	3	2	11	0	0
480	51	3	2	18	4	2
696	39	1	0	6	5	2
...	...	...	...	...	...	...
760	38	1	2	10	5	2
1296	41	2	7	16	14	3
476	59	2	7	20	4	3
840	33	2	1	7	6	5
4	22	1	1	1	0	0

400 rows × 6 columns

```
#REGRESSION
# Plot decision tree for regression
plt.figure(figsize=(7, 4))
plot_tree(reg_tree, filled=True, feature_names=df_RG.columns)
```

```
plt.title("Decision Tree Regressor ")
plt.show()
```

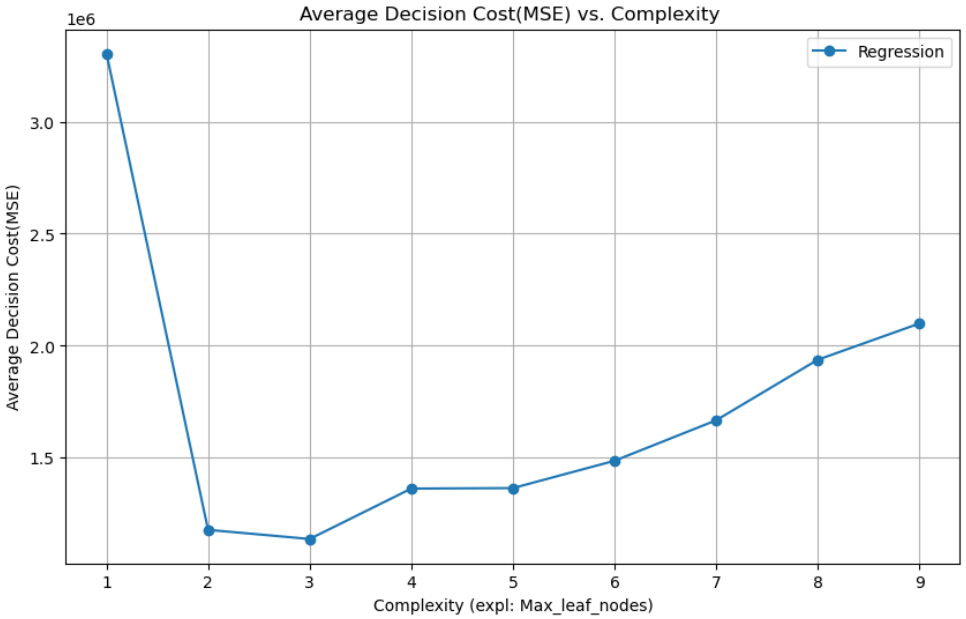



```
# Define a range of hyperparameters to explore
param_range = range(1, 10)  # Example range for max_depth

# Initialize lists to store average decision cost for
    classification and regression
avg_decision_cost_reg = []

# Loop through each hyperparameter value
for param in param_range:
    # Regression
    reg = DecisionTreeRegressor(max_depth=param, random_state
    =42)
    reg.fit(X_reg_train, y_reg_train)
    reg_pred = reg.predict(X_reg_test)
    avg_decision_cost_reg.append(mean_squared_error(
    y_reg_test, reg_pred))

# Plot the graph
plt.figure(figsize=(10, 6))
#plt.plot(param_range, avg_decision_cost_clf, label='
    Classification', marker='o')
plt.plot(param_range, avg_decision_cost_reg, label='
    Regression', marker='o')
plt.xlabel('Complexity (expl: max_depth)')
plt.ylabel('Average Decision Cost(MSE)')
plt.title('Average Decision Cost(MSE) vs. Complexity ')
plt.legend()
plt.grid(True)
```

```
plt.show()
```

La complexité du modèle peut être mesurée de différentes manières. Cela pourrait être le nombre de nœuds, la profondeur de l'arbre, etc.
Si on choisit le *Nombre maximal de nœuds* comme complexité :

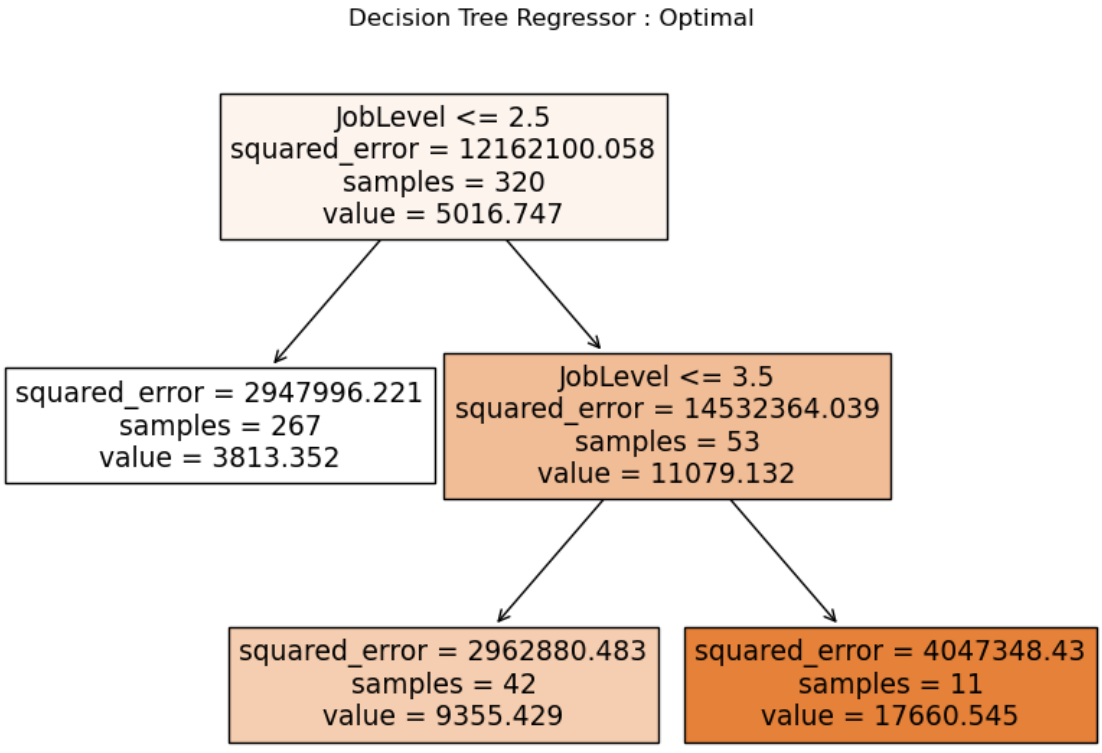
La figure ci-dessous montre la relation entre la complexité de l'arbre, reflétée par le Nombre maximal de nœuds feuilles qu'un arbre peut avoir(*Max_leaf_nodes*) et l'*Average Decision Cost(MSE)* (Le coût de décision designé par l'erreur quadratique moyenne) :



Arbre Optimal pour la régression :

```
# DATA FOR REGRESSION
SUPP = [ 'MonthlyIncome' , 'DailyRate' , 'DistanceFromHome' ,
     'Attrition(Exit or Not)', 'HourlyRate' ,  '
    PercentSalaryHike(%aug salaire )' , 'BusinessTravel','
    Department','Education','EducationField','EmployeeNumber(
    id)','EnvironmentSatisfaction','Gender','JobInvolvement','
    JobRole','JobSatisfaction','MaritalStatus','OverTime','
    PerformanceRating','RelationshipSatisfaction','
    StockOptionLevel','TrainingTimesLastYear','WorkLifeBalance
    ','YearsSinceLastPromotion','YearsWithCurrManager']
df_RG = balanced_df.drop(SUPP , axis = 1)

X_reg = df_RG
y_reg = balanced_df['MonthlyIncome']
X_reg_train, X_reg_test, y_reg_train, y_reg_test =
```

```
    train_test_split(X_reg, y_reg, test_size=0.2, random_state
    =0)

# Create and train decision tree regressors
reg_tree = reg_tree = DecisionTreeRegressor(random_state=0,
    max_leaf_nodes=3)

reg_tree.fit(X_reg_train, y_reg_train)

# Plot decision tree for regression
plt.figure(figsize=(10, 7))
plot_tree(reg_tree, filled=True, feature_names=df_RG.columns)
plt.title("Decision Tree Regressor : Optimal ")
plt.show()
```



▶ D'après cette arbre de décision, On peut prédire le salaire d'un employé.