Rapport projet

Nettoyage et visualisation des données

Hiba Khouzai Aminata Sangho

Informatioque et Ingénierie des Données 2
Professor Amal Ourdou

May 28, 2023

May 28, 2023

Contents

1 Introduction

Un constructeur automobile envisage de pénétrer de nouveaux marchés avec ses produits existants (P1, P2, P3, P4 et P5). Après une étude de marché intensive, ils ont déduit que le comportement du nouveau marché est similaire à leur marché existant.

Dans leur marché existant, l'équipe commerciale a classé tous les clients en 4 segments (A, B, C, D).
Ce dataset a été acquis à partir du hackathon Analytics Vidhya. Cliquez ici
Les outils itilisés sont Python et Power BI

[42]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import missingno as msno
import pandas as pd

# Machine Learning
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.tree import  DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn import tree
from wordcloud import WordCloud, STOPWORDS
```

[60]:
```
df = pd.read_csv("Train.csv")
df_test = pd.read_csv("Test.csv")
display(df.head())
```

```
       ID  Gender Ever_Married  Age Graduated     Profession  Work_Experience  \
0  462809    Male           No   22        No     Healthcare              1.0
1  462643  Female          Yes   38       Yes       Engineer              NaN
2  466315  Female          Yes   67       Yes       Engineer              1.0
3  461735    Male          Yes   67       Yes         Lawyer              0.0
4  462669  Female          Yes   40       Yes  Entertainment              NaN

  Spending_Score  Family_Size  Var_1 Segmentation
0            Low          4.0  Cat_4            D
1        Average          3.0  Cat_4            A
2            Low          1.0  Cat_6            B
3           High          2.0  Cat_6            B
4           High          6.0  Cat_6            A
```

2 Analyse du dataset

[44]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8068 entries, 0 to 8067
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               8068 non-null   int64
 1   Gender           8068 non-null   object
 2   Ever_Married     7928 non-null   object
 3   Age              8068 non-null   int64
 4   Graduated        7990 non-null   object
 5   Profession       7944 non-null   object
 6   Work_Experience  7239 non-null   float64
 7   Spending_Score   8068 non-null   object
 8   Family_Size      7733 non-null   float64
 9   Var_1            7992 non-null   object
 10  Segmentation     8068 non-null   object
dtypes: float64(2), int64(2), object(7)
memory usage: 693.5+ KB
```

2.1 Verification des duplicata

[45]:
```
print(df.duplicated().sum())
print(df_test.duplicated().sum())
```

```
0
0
```

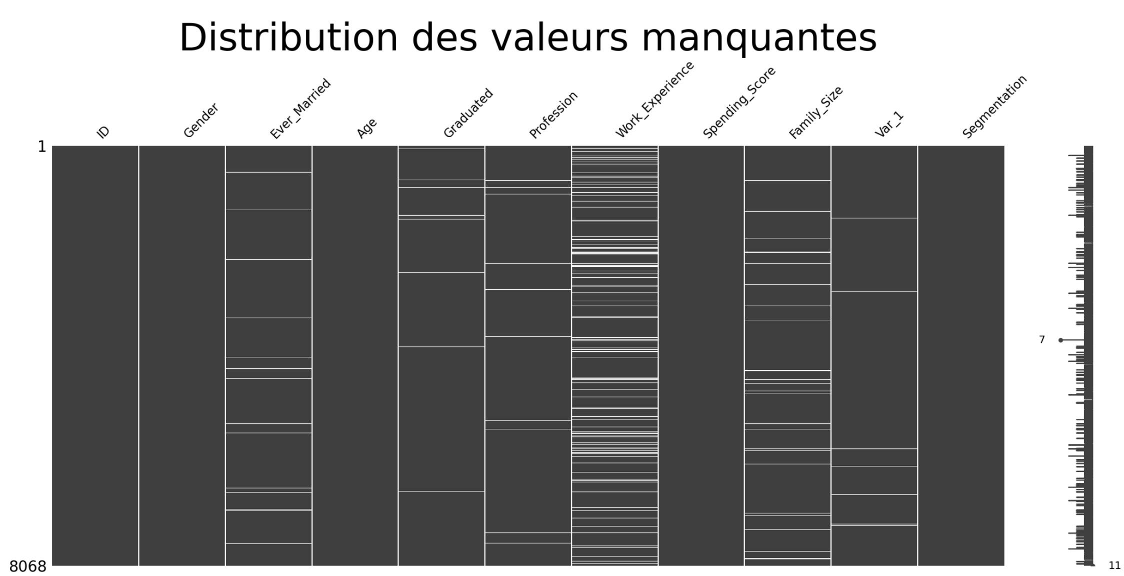
2.2 Verification des valeurs manquantes

[46]:
```
print(df.shape)
print('*'*20)

print(df.isnull().sum())
print('*'*20)

print('Pourcentage des valeurs manquantes :\n',df.isnull().mean()*100 )

print('*'*20)
msno.matrix(df)
plt.title('Distribution des valeurs manquantes',fontsize = 50)
```

```
(8068, 11)
********************
ID                   0
Gender               0
```

```
Ever_Married       140
Age                  0
Graduated           78
Profession         124
Work_Experience    829
Spending_Score       0
Family_Size        335
Var_1               76
Segmentation         0
dtype: int64
********************
Pourcentage des valeurs manquantes :
 ID                  0.000000
Gender              0.000000
Ever_Married        1.735250
Age                 0.000000
Graduated           0.966782
Profession          1.536936
Work_Experience    10.275161
Spending_Score      0.000000
Family_Size         4.152206
Var_1               0.941993
Segmentation        0.000000
dtype: float64
********************
```

[46]: Text(0.5, 1.0, 'Distribution des valeurs manquantes')



[47]:
```python
# Description des données de type objet, on en aura besoin pour remplacer
# les nan par la valeur la plus mentionnée i.e. le mode
df.describe(include='object')
```

[47]:
```
       Gender Ever_Married Graduated Profession Spending_Score  Var_1  \
count    8068         7928      7990       7944           8068   7992
unique      2            2         2          9              3      7
top      Male          Yes       Yes     Artist            Low  Cat_6
freq     4417         4643      4968       2516           4878   5238

       Segmentation
count          8068
unique            4
top               D
freq           2268
```

2.3 Value count

[48]:
```python
# les valeurs distincts de chaque colonne
colTypeObj = df.select_dtypes('object')

for i in colTypeObj:
    print(df[i].value_counts(), end="\n\n")
```

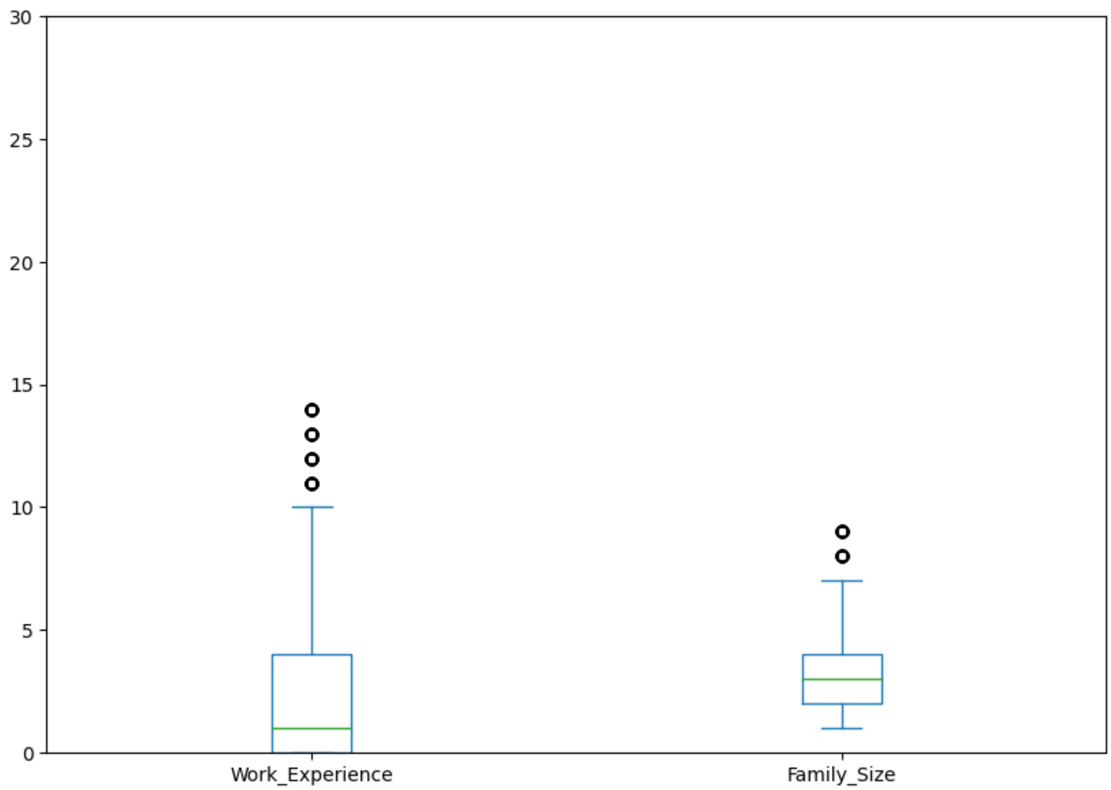
```
Male      4417
Female    3651
Name: Gender, dtype: int64

Yes    4643
No     3285
Name: Ever_Married, dtype: int64

Yes    4968
No     3022
Name: Graduated, dtype: int64

Artist           2516
Healthcare       1332
Entertainment     949
Engineer          699
Doctor            688
Lawyer            623
Executive         599
Marketing         292
Homemaker         246
Name: Profession, dtype: int64

Low        4878
```

```
Average    1974
High       1216
Name: Spending_Score, dtype: int64

Cat_6    5238
Cat_4    1089
Cat_3     822
Cat_2     422
Cat_7     203
Cat_1     133
Cat_5      85
Name: Var_1, dtype: int64

D    2268
A    1972
C    1970
B    1858
Name: Segmentation, dtype: int64
```

2.4 Verification des Outliers

[49]:
```
nouveau_df = df[['Work_Experience', 'Family_Size']]
nouveau_df

nouveau_df.plot(kind='box', figsize=(10,7))
plt.ylim(0,30)
```

[49]:
```
(0.0, 30.0)
```



[50]:

```
# display(df.head())
list_profession = np.array(df[df['Profession']!='nan']['Profession'])

text = ' '.join(list_profession.astype(str))

# instantiate a word cloud object


stopwords = set(STOPWORDS)
alice_wc = WordCloud(background_color='white', max_words=2000,␣
 ↪stopwords=stopwords)

alice_wc.generate(text)

# display the word cloud
fig = plt.figure()
fig.set_figwidth(14) # set width
fig.set_figheight(18) # set height

plt.imshow(alice_wc, interpolation='bilinear')
plt.axis('off')
```

```
plt.show()
```

3 Nettoyage de données

3.1 Procedure

- On remplace les valeurs manquantes de 'Ever_Married' et 'Graduated' par 'NO'.
- On remplace les valeurs manquantes de 'Profession' et 'Var_1' par la valeur la plus mentionnéé.
- On remplace les valeurs manquantes de 'Work_Experience' et 'Family_Size' par la valeur la mediane et on enleve les valeurs abberantes.

[51]:
```
# On appliquera la fonction Nettoyer_Donnees sur le trainset et le test set pour␣
↪les nettoyer

def supp_valAbberantes(df, col):
        qmin, qmax = df[col].quantile(.25), df[col].quantile(.75)
        interq = 1.5 * (qmax - qmin)
        qmin -= interq
        qmax += interq
        return df[col].apply(lambda x: qmin if x < qmin else qmax if x > qmax␣
↪else x)


def Nettoyer_Donnees(df):

    df['Ever_Married'] = df['Ever_Married'].fillna('No')
    df['Graduated'] = df['Graduated'].fillna('No')
```

```
    for col in ['Profession', 'Var_1']:
        df[col] = df[col].fillna(df[col].mode().values[0])

    for col in ['Work_Experience', 'Family_Size']:
        df[col] = df[col].fillna(df[col].median())
        df[col] = supp_valAbberantes(df, col)

    segment_map = {'A':1, 'B':2, 'C':3, 'D':4}
    df['Segmentation'] = df['Segmentation'].map(segment_map)
    for col in df.select_dtypes(exclude='number'):
        df[col] = df[col].apply(lambda x: str(x).strip())
    return df
```

[52]:
```
Donnees_nettoye = Nettoyer_Donnees(df)
display(Donnees_nettoye.head())

print('*'*20,'\n')
print(df.shape)

print('*'*20,'\n')
print(Donnees_nettoye.isnull().sum())
```

```
       ID  Gender Ever_Married  Age Graduated     Profession  Work_Experience  \
0  462809    Male           No   22        No     Healthcare              1.0
1  462643  Female          Yes   38       Yes       Engineer              1.0
2  466315  Female          Yes   67       Yes       Engineer              1.0
3  461735    Male          Yes   67       Yes         Lawyer              0.0
4  462669  Female          Yes   40       Yes  Entertainment              1.0

  Spending_Score  Family_Size  Var_1  Segmentation
0            Low          4.0  Cat_4             4
1        Average          3.0  Cat_4             1
2            Low          1.0  Cat_6             2
3           High          2.0  Cat_6             2
4           High          6.0  Cat_6             1

********************

(8068, 11)
********************

ID                 0
Gender             0
Ever_Married       0
Age                0
Graduated          0
Profession         0
Work_Experience    0
```

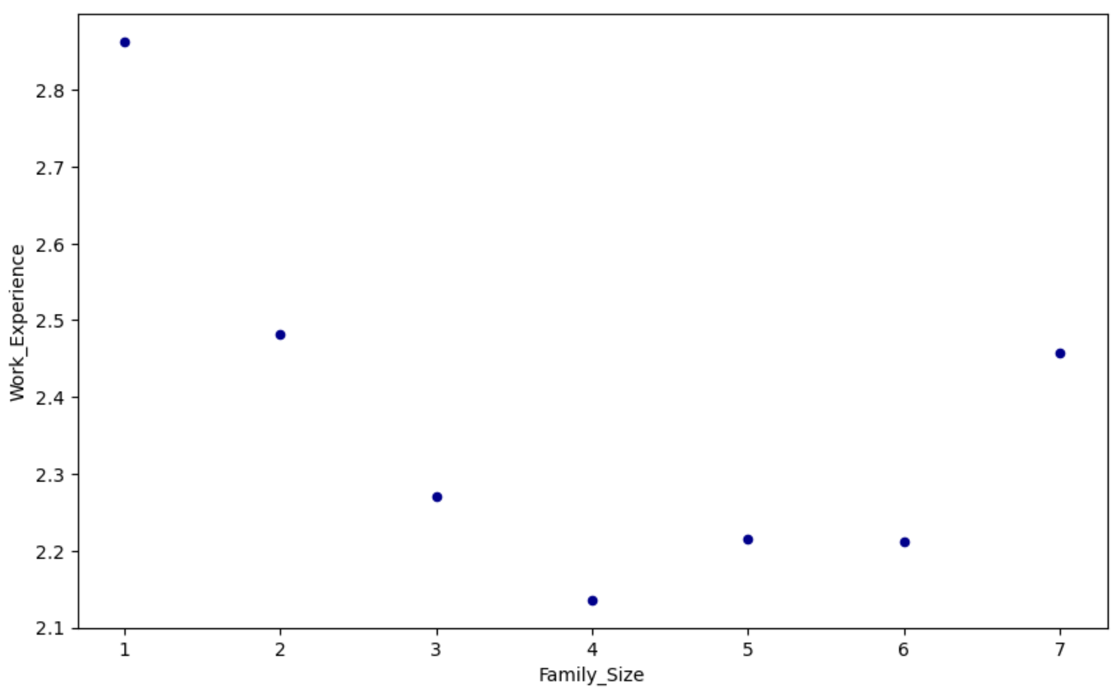
```
Spending_Score     0
Family_Size        0
Var_1              0
Segmentation       0
dtype: int64
```

4 Visualisation de données

4.1 Visualisation avec Python

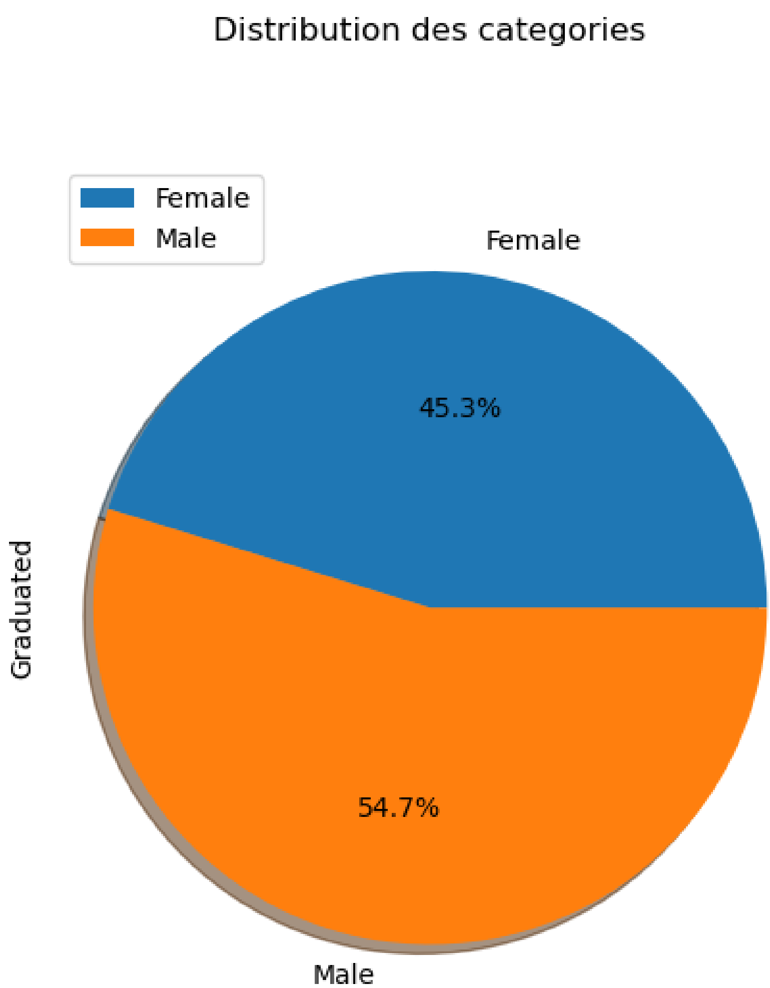
[53]:
```
FS = Donnees_nettoye.groupby("Family_Size",axis = 0).mean()[["Work_Experience"]]
new_df = pd.DataFrame({'Family_Size' : np.array(FS.index) , 'Work_Experience' :␣
↪np.array(FS["Work_Experience"])})

new_df.plot(kind='scatter', x='Family_Size', y='Work_Experience', figsize=(10,␣
↪6), color='darkblue')
```

[53]: <AxesSubplot:xlabel='Family_Size', ylabel='Work_Experience'>



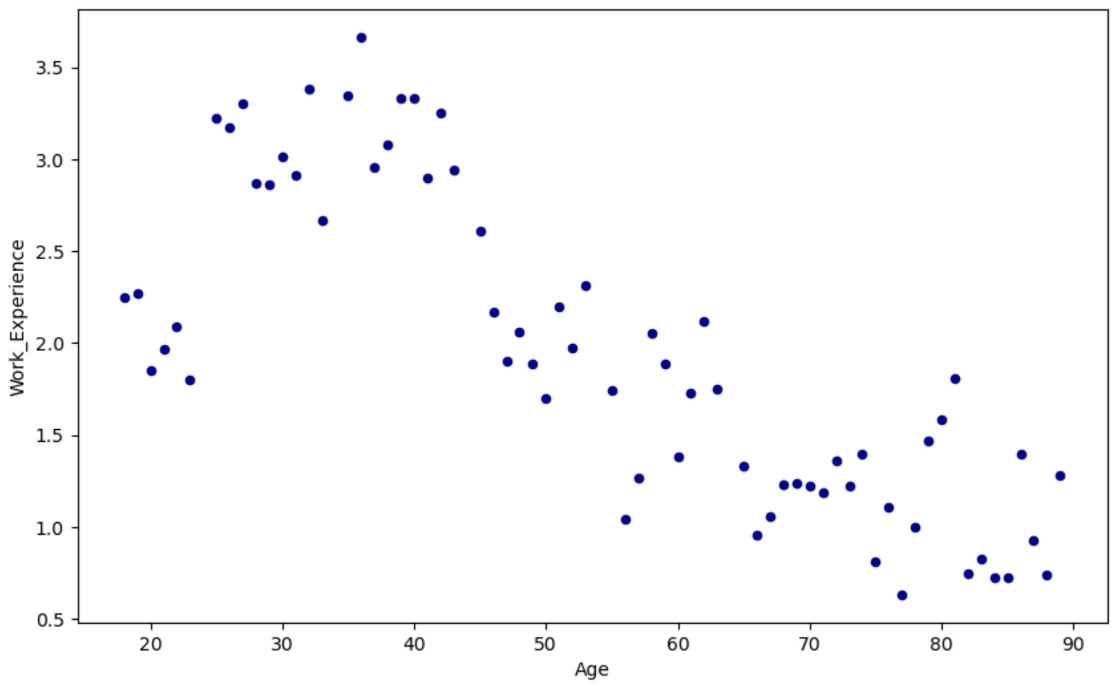
[54]:
```
A = Donnees_nettoye.groupby("Gender",axis = 0).count()[["Graduated"]]

A["Graduated"].plot(kind='pie',
                            figsize=(5, 6),
                            autopct='%1.1f%%', # add in percentages
```

```
                              startangle=0,     # start angle 90° (Africa)
                              shadow=True,       # add shadow
                              )

plt.title('Distribution des categories', y=1.12)

plt.axis('equal')

# add legend
plt.legend(labels=A.index, loc='upper left')

plt.show()
```



[55]:
```
FS1 = Donnees_nettoye.groupby("Age",axis = 0).mean()[["Work_Experience"]]

new_df = pd.DataFrame({'Age' : np.array(FS1.index) , 'Work_Experience' : np.
↪array(FS1["Work_Experience"])})

new_df.plot(kind='scatter', x='Age', y='Work_Experience', figsize=(10, 6),␣
↪color='darkblue')
```

[55]: <AxesSubplot:xlabel='Age', ylabel='Work_Experience'>

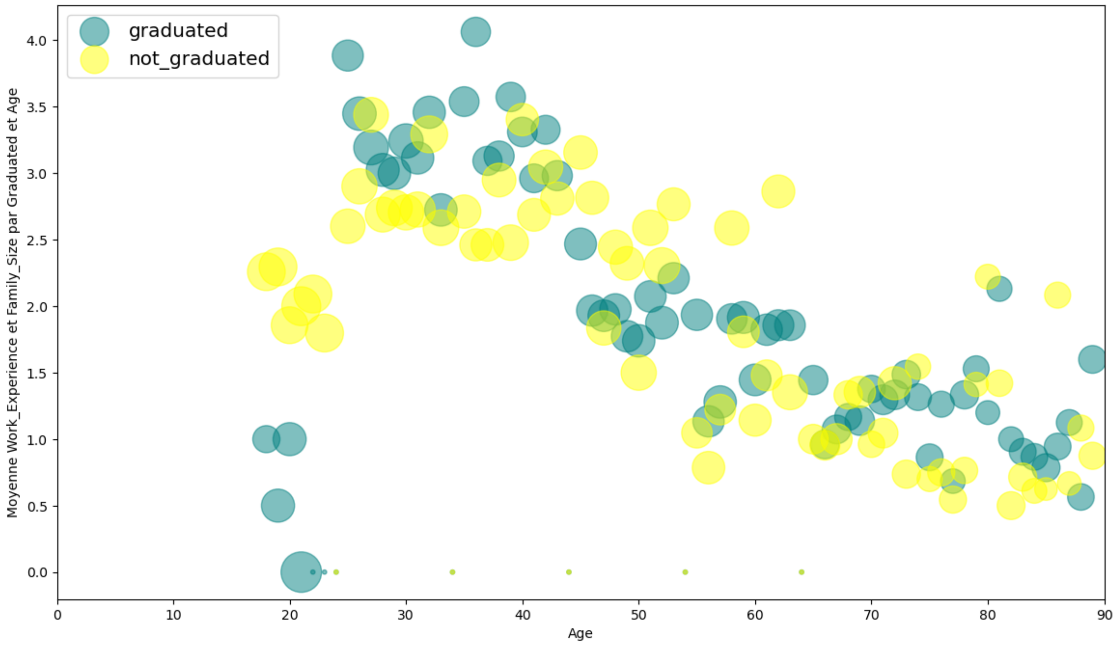


[56]:
```
df_graduated = Donnees_nettoye[Donnees_nettoye["Graduated"]=='Yes'].
↪groupby("Age",axis = 0)\
                  .mean()[["Work_Experience","Family_Size"]]

df_not_graduated = Donnees_nettoye[Donnees_nettoye["Graduated"]=='No'].
↪groupby("Age",axis = 0)\
                  .mean()[["Work_Experience","Family_Size"]]


for i in range(18,90):
    if i not in df_graduated.index:
        df_graduated.loc[i] = [0,0]
```

```
for i in range(18,90):
    if i not in df_not_graduated.index:
        df_not_graduated.loc[i] = [0,0]



df_graduated = df_graduated.sort_index()
df_not_graduated = df_not_graduated.sort_index()


new_df_taille_fam = pd.DataFrame({'Age' : df_graduated.index ,
                                  'Work_Experience_yes' : np.
 ↪array(df_graduated["Work_Experience"]),
                                  'fam_yes' : np.
 ↪array(df_graduated["Family_Size"]),
                                  'Work_Experience_no' : np.
 ↪array(df_not_graduated["Work_Experience"]),
                                 'fam_no' : np.
 ↪array(df_not_graduated["Family_Size"])
                                 })
```

[57]:
```
# Plotting

# Graduated
ax0 = new_df_taille_fam.plot(kind='scatter',
                   x='Age',
                   y='Work_Experience_yes',
                   figsize=(14, 8),
                   alpha=0.5,                 # transparency
                   color='teal',
                   s=new_df_taille_fam['fam_yes']* 200 + 10,  # pass in weights
                   xlim=(0, 90)
                  )

# not_graduated
ax1 = new_df_taille_fam.plot(kind='scatter',
                   x='Age',
                   y='Work_Experience_no',
                   alpha=0.5,
                   color="yellow",
                   s=new_df_taille_fam['fam_no']* 200 + 10,
                   ax = ax0
                  )

ax0.set_ylabel('Moyenne Work_Experience et Family_Size par Graduated et Age')
ax0.set_title('')
ax0.legend(['graduated', 'not_graduated'], loc='upper left', fontsize='x-large')
```

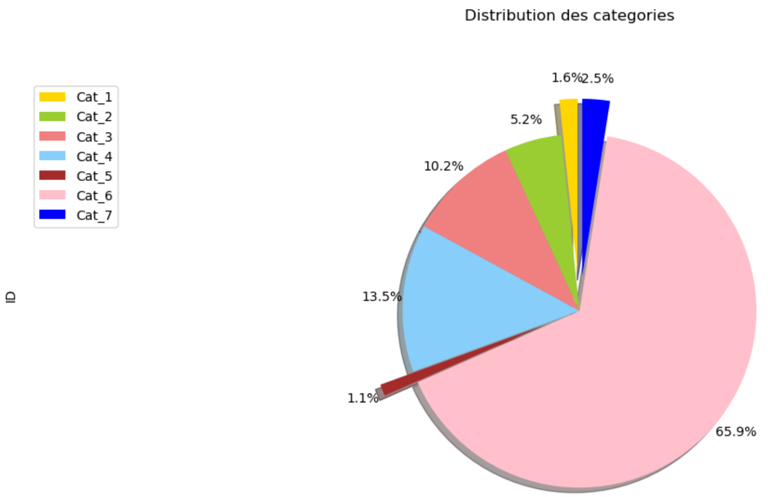
[57]: <matplotlib.legend.Legend at 0x1f6000aae80>



[58]:
```
B = Donnees_nettoye.groupby("Var_1",axis = 0).count()
```

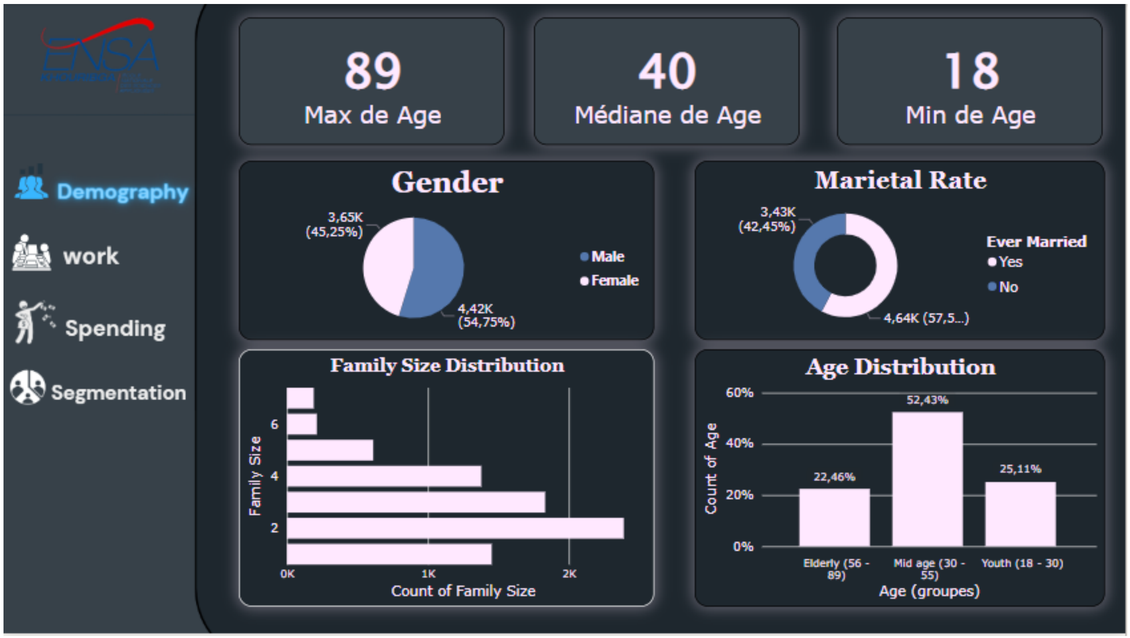
[59]:
```
K = B.sort_values(by = "ID", ascending=True)
colors_list = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'brown',␣
↪'pink','blue']
explode_list = [0.2, 0, 0, 0, 0.2, 0, 0.2] # ratio for each continent with which␣
↪to offset each wedge.

B["ID"].plot(kind='pie',
                            figsize=(15, 6),
                            autopct='%1.1f%%',
                            startangle=90,
                            shadow=True,
                            labels=None,         # turn off labels on pie chart
                            pctdistance=1.12,    # the ratio between the center␣
↪of each pie
                                                 # slice and the start of the␣
↪text generated by autopct

                            colors=colors_list,  # add custom colors
                            explode=explode_list # 'explode' lowest 3 categories
                            )
```

```
# scale the title up by 12% to match pctdistance
plt.title('Distribution des categories', y=1.12)

plt.axis('equal')

# add legend
plt.legend(labels=B.index, loc='upper left')

plt.show()
```

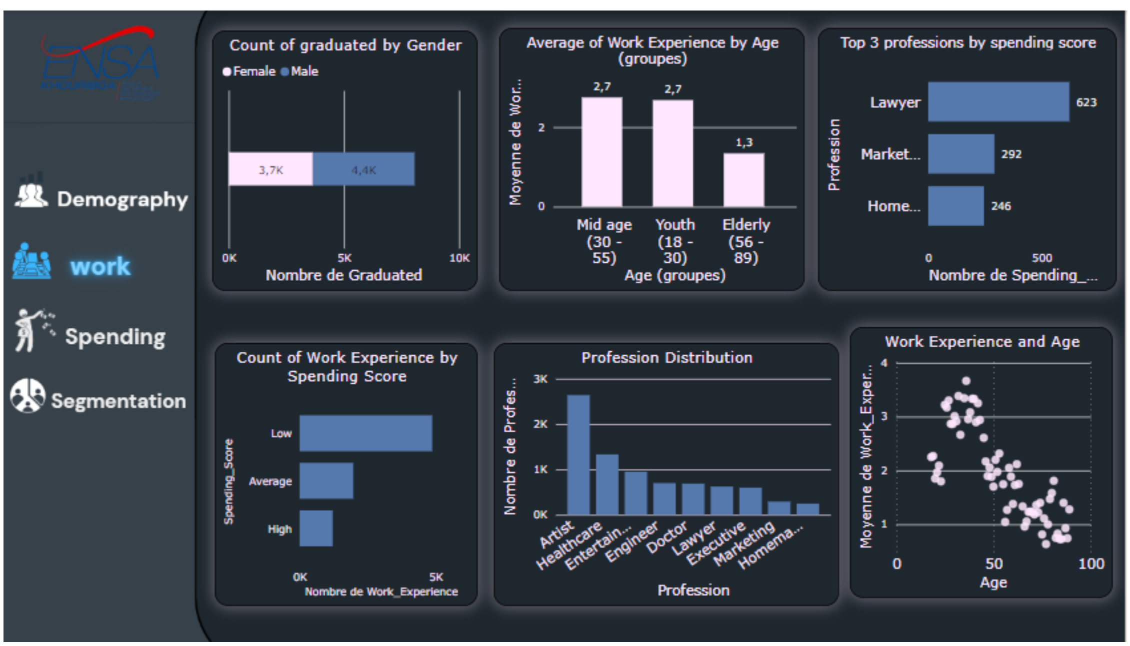


4.2 Visualisation avec Power BI

- What is the gender distribution within the dataset ?
- How many indivisuals are married vs unmarried ?
- What is the age distribution of the dataset ?
- What is the proportion of individuals who have graduated ?
- What is the distribution of family sizes in the dataset ?
- Which professions are most common among the dataset ?
- What is the distribution of spending scores among individuals ?

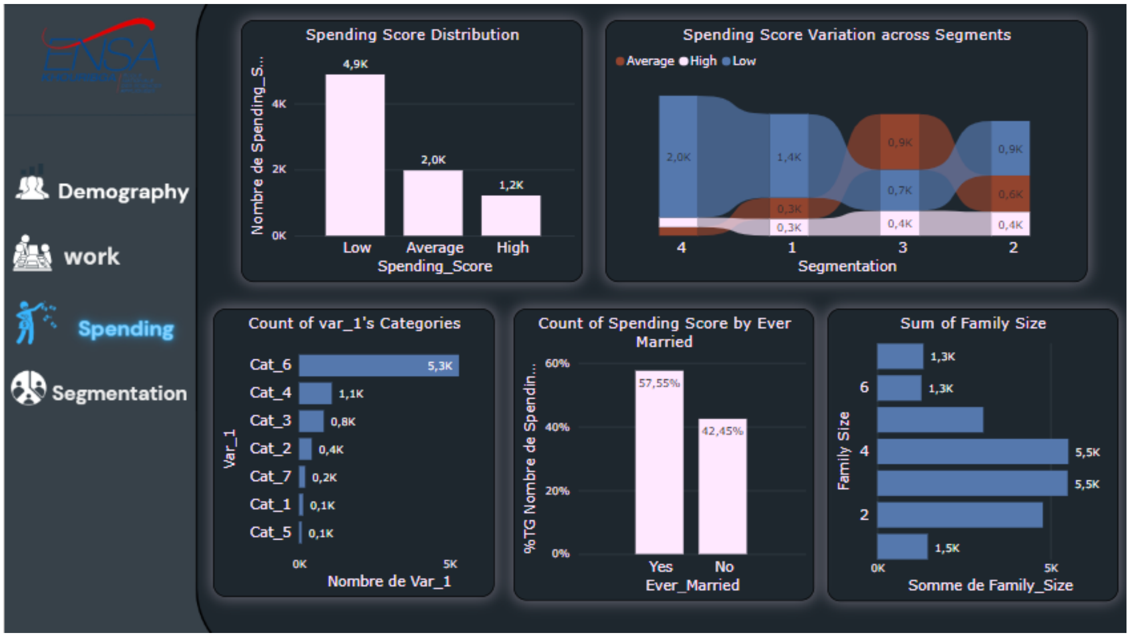


- What is the average work experience for different age groups ?
- How does graduation vary by gender ?
- What are the top 3 profession by spending score ?
- How does work experience vary with age ?

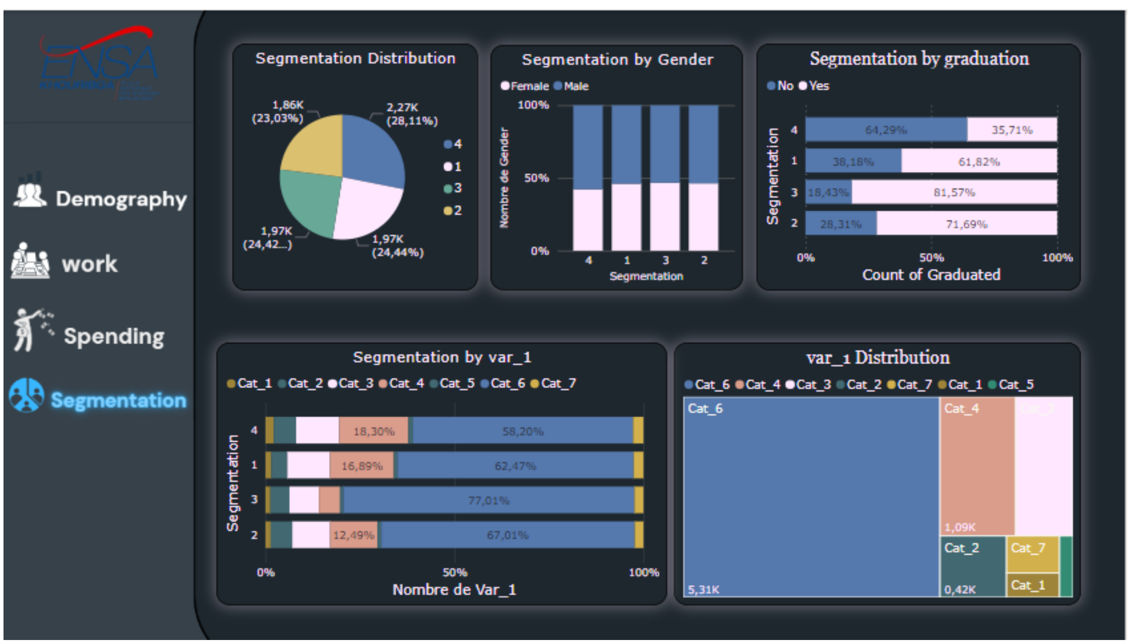


- How does the spending score vary across different segments?
- How does the spending score vary between married and unmarried individuals?

- Are there any notable differences in spending behavior across segments?



- How are customers segmented in the dataset?
- How does the segmentation differ between those who have graduated and those who haven't?
- How does the segmentation vary by gender?
- How does the segmentation differ based on the Var1 category?



5 Conclusion

Dans cette analyse, nous avions pu identifier les anomalies de notre dataset, les corriger et visualiser le résultat dans un outils de visualisation adapté, pour mieux en tirer des décisions .