```python
import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sns
import stat
```

Importation des données

```python
countries_HDI = pd.read_csv('C:\\Users\\SCD UM\\Downloads\\stat\\countries.HDI.csv', encoding='latin1', header=0)

effec1_quest_compil = pd.read_csv('C:\\Users\\SCD UM\\Downloads\\stat\\effec1.quest.compil.csv', encoding='latin1')
effec2_quest_compil = pd.read_csv('C:\\Users\\SCD UM\\Downloads\\stat\\effec2.quest.compil.csv', encoding='latin1')
effec3_quest_compil = pd.read_csv('C:\\Users\\SCD UM\\Downloads\\stat\\effec3.quest.compil.csv', encoding='latin1')

usages_effec1 = pd.read_csv('C:\\Users\\SCD UM\\Downloads\\stat\\usages.effec1.csv', encoding='latin1')
usages_effec2 = pd.read_csv('C:\\Users\\SCD UM\\Downloads\\stat\\usages.effec2.csv', encoding='latin1')
usages_effec3 = pd.read_csv('C:\\Users\\SCD UM\\Downloads\\stat\\usages.effec3.csv', encoding='latin1')
```

Fusion des données des questionnaires d'efficacité avec les données d'utilisation

```python
effec1 = pd.merge(effec1_quest_compil, usages_effec1, on='Student_ID')
effec2 = pd.merge(effec2_quest_compil, usages_effec2, on='Student_ID')
effec3 = pd.merge(effec3_quest_compil, usages_effec3, on='Student_ID')
```

```python
# Ajoutons une nouvelle itération pour chaque DataFrame:
effec1['itération'] = 1
effec2['itération'] = 2
effec3['itération'] = 3
```

```python
# Renommer les colonnes de countries_HDI
countries_HDI = countries_HDI.rename(columns={"Norvège": "Country", "TH": "HDI", "1": "Identifier"})
```

Pour rassembler les données des différentes itérations ("row bind"), nous utiliserons la fonction concat() de pandas :

```python
final_df = pd.concat([effec1, effec2, effec3], ignore_index=True)
```

```python
final_df2 = pd.merge(final_df, countries_HDI, on="Country", how="left")
```

Enfin, pour conserver seulement certaines colonnes (par exemple 'HDI' et 'genre')

```python
#Extraction des colonnes sélectionnées dans un DataFrame final
selected_columns_df = final_df2[['HDI', 'Gender', 'itération']]
```

Pour quantifier le nombre de vidéos visionnées et de quiz réalisés par chaque apprenant, on peut créér deux nouvelles métriques quantitatives qui nous permettent de mesurer l'engagement des apprenants de manière plus précise.

```python
# Création des colonnes pour le nombre total de vidéos visionnées et de quiz réalisés
video_columns = [col for col in final_df2.columns if 'S' in col]
quiz_columns = [col for col in final_df2.columns if 'Quizz' in col and 'bin' in col]
```

```python
# Convertir les données non numériques en NaN et les remplacer par zéro
final_df2[video_columns] = final_df2[video_columns].apply(pd.to_numeric, errors='coerce').fillna(0)
final_df2[quiz_columns] = final_df2[quiz_columns].apply(pd.to_numeric, errors='coerce').fillna(0)
```

Le nombre total de vidéos visionnées et de quiz réalisés par chaque étudiant

```python
final_df2['Total_Videos_visionnees'] = final_df2[video_columns].apply(sum, axis=1)
final_df2['Total_Quizzes_realises'] = final_df2[quiz_columns].apply(sum, axis=1)

final_df2['Total_Videos_visionnees'] = final_df2['Total_Videos_visionnees'].astype(int)


# Compter le nombre d'étudiants dans chaque catégorie HDI
HDI_counts = final_df2['HDI'].value_counts()
print(HDI_counts)

# Créer une nouvelle colonne HDI où 'M' et 'H' sont regroupées en 'I'
final_df2['HDI_Grouped'] = final_df2['HDI'].replace({'M': 'I', 'H': 'I'})

# Verifier le resultat
final_df2[['HDI', 'HDI_Grouped', 'Total_Videos_visionnees', 'Total_Quizzes_realises']].head()
```

```
B     13292
TH     7260
M       354
H       313
Name: HDI, dtype: int64
```

	HDI	HDI_Grouped	Total_Videos_visionnees	Total_Quizzes_realises
0	B	B	221	0
1	B	B	221	0
2	TH	TH	19179	0
3	TH	TH	1116	4
4	TH	TH	1949	0

```python
# Compter le nombre d'étudiants dans chaque catégorie HDI_Grouped
HDI_Grouped_counts = final_df2['HDI_Grouped'].value_counts()
print(HDI_Grouped_counts)
```

```
B     13292
TH     7260
I       667
Name: HDI_Grouped, dtype: int64
```

```python
# Création des colonnes pour chaque type d'apprenant
final_df2['Bystander'] = np.where((final_df2['Total_Videos_visionnees']==0) & (final_df2['Total_Quizzes_realises']==0), 1, 0)
final_df2['Auditing'] = np.where((final_df2['Total_Videos_visionnees']>0) & (final_df2['Total_Quizzes_realises']==0), 1, 0)
final_df2['Completer'] = np.where((final_df2['Total_Videos_visionnees']>0) & (final_df2['Total_Quizzes_realises']>0), 1, 0)
final_df2['Disengaging'] = np.where((final_df2['Total_Videos_visionnees']==0) & (final_df2['Total_Quizzes_realises']==0), 1, 0)
```

```python
# Création de la table de synthèse
summary_table = final_df2.groupby('itération')[['Bystander', 'Auditing', 'Completer', 'Disengaging']].sum()
```

```python
summary_table
```

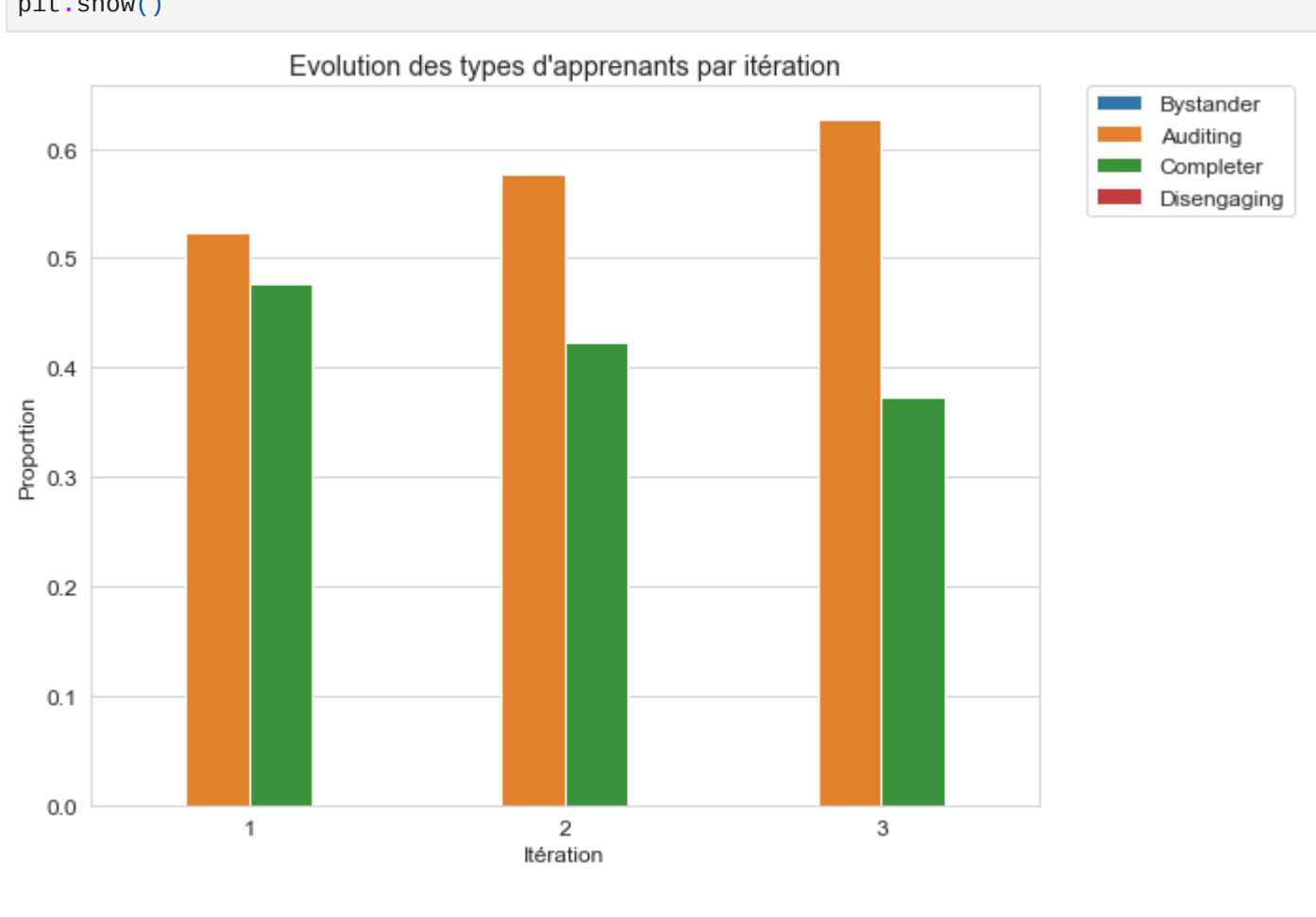
	Bystander	Auditing	Completer	Disengaging
itération				
1	0.0	0.523093	0.476907	0.0
2	0.0	0.576268	0.423732	0.0
3	0.0	0.626283	0.373717	0.0

```python
# Calcul des proportions
summary_table = summary_table.apply(lambda x: x / x.sum(), axis=1)
```

```python
print(summary_table)
```

```
           Bystander  Auditing  Completer  Disengaging
itération
1                0.0  0.523093   0.476907          0.0
2                0.0  0.576268   0.423732          0.0
3                0.0  0.626283   0.373717          0.0
```

```python
import matplotlib.pyplot as plt

# Set style of the plot
sns.set_style("whitegrid")
# Preparer les données
df_plot = summary_table.reset_index().melt('itération')

# Créer la bar plot
plt.figure(figsize=(9, 7))
bar_plot = sns.barplot(x='itération', y='value', hue='variable', data=df_plot)

# Placer la legend à l'extérieur
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

plt.title('Evolution des types d\'apprenants par itération')
plt.xlabel('Itération')
plt.ylabel('Proportion')

# Sauvegarder la figure
plt.savefig("Evolution des types d'apprenants par itération.png")

plt.show()
```



4 Chi2 et mosaic plot:

```python
from scipy.stats import chi2_contingency

# Créer la cross-tabulation
crosstab = pd.crosstab(final_df2['HDI'], final_df2['Gender'])

# La Performance de test Chi-Square
chi2, p, dof, expected = chi2_contingency(crosstab)

print("Chi-Square value: ", chi2)
print("p-value: ", p)
```

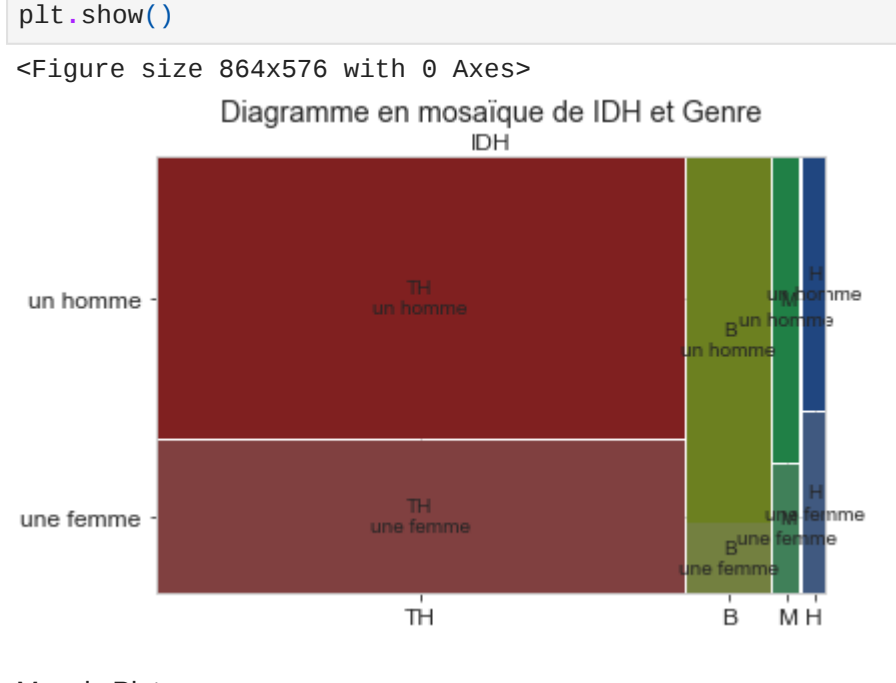
```
Chi-Square value:  175.75674510524374
p-value:  7.273677317477502e-38
```

Chi-Square :

Mosaic plot :

```python
from statsmodels.graphics.mosaicplot import mosaic
import matplotlib.pyplot as plt

plt.rcParams['font.size'] = 12  # change la taille de la police
plt.figure(figsize=(12,8))
mosaic(final_df2, ['HDI', 'Gender'], title='Diagramme en mosaïque de IDH et Genre')
plt.xlabel('IDH')  # étiquette pour l'axe x
plt.ylabel('Genre')  # étiquette pour l'axe y
plt.savefig("Diagramme en mosaïque de IDH et Genre.png")
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



Mosaic Plot :

```python
import numpy as np

# Calculate Cramer's V
V = np.sqrt((chi2/(final_df2.shape[0]*(min(crosstab.shape)-1))))
print("Cramer's V: ", V)
```

```
Cramer's V:  0.09081213724541108
```

5 Modèle linéaire, tests non paramétriques:

**1. Test de Student :

```python
from scipy.stats import ttest_ind

# Séparer les données en deux groupes
group1 = final_df2[final_df2['Gender'] == 'une femme']['Total_Videos_visionnees']
group2 = final_df2[final_df2['Gender'] == 'un homme']['Total_Videos_visionnees']

# Effectuer le test t de Student
t_stat, p_val = ttest_ind(group1, group2, equal_var=False)

print(f"t-statistic: {t_stat}")
print(f"p-value: {p_val}")
```

```
t-statistic: 5.369331374528332
p-value: 8.209732654547305e-08
```

2. Test non paramétrique :

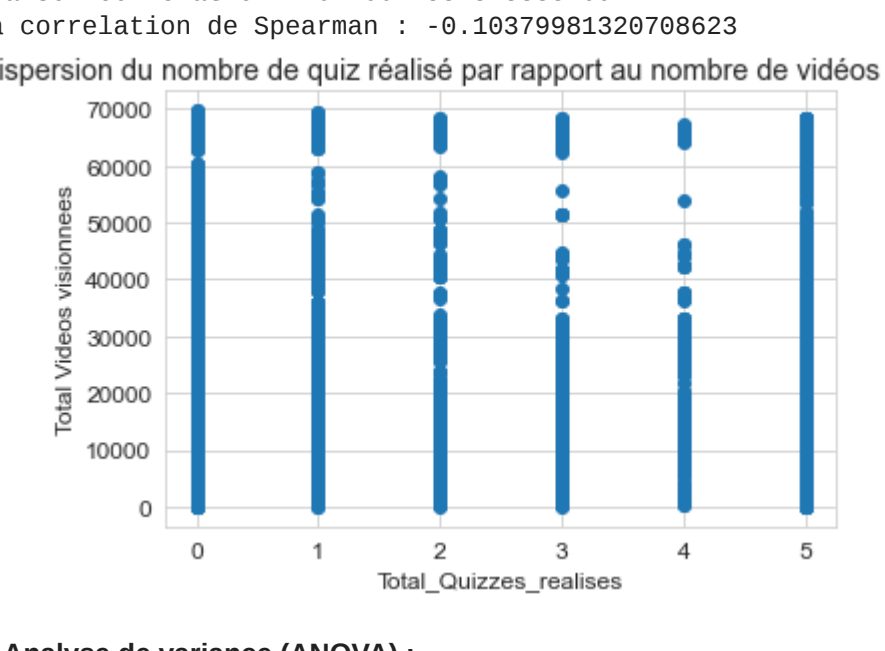
```python
from scipy.stats import mannwhitneyu

# Effectuer le test U de Mann-Whitney
u_stat, p_val = mannwhitneyu(group1, group2)

print(f"U-statistic: {u_stat}")
print(f"p-value: {p_val}")
```

```
U-statistic: 10010895.0
p-value: 4.604958486880628e-11
```

3. Régression linéaire et tests de corrélation :

```python
from scipy.stats import pearsonr, spearmanr

# Calculer la corrélation de Pearson
pearson_corr, _ = pearsonr(final_df2['Total_Quizzes_realises'], final_df2['Total_Videos_visionnees'])
print(f"Pearson correlation: {pearson_corr}")

# Calculer la corrélation de Spearman
spearman_corr, _ = spearmanr(final_df2['Total_Quizzes_realises'], final_df2['Total_Videos_visionnees'])
print(f"La correlation de Spearman : {spearman_corr}")

# Scatter plot
plt.scatter(final_df2['Total_Quizzes_realises'], final_df2['Total_Videos_visionnees'])
plt.xlabel('Total_Quizzes_realises')
plt.ylabel('Total Videos visionnees')
plt.title('Dispersion du nombre de quiz réalisé par rapport au nombre de vidéos visionnées')
plt.savefig("corr.png")
plt.show()
```

```
Pearson correlation: -0.10027554548385409
La correlation de Spearman : -0.10379981320708623
```



4. Analyse de variance (ANOVA) :

```python
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Fit the model
model = ols('Total_Videos_visionnees ~ C(HDI) + C(Gender)', data=final_df2).fit()

# Perform ANOVA and print the table
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)
```

```
                 sum_sq      df          F        PR(>F)
C(HDI)     3.455544e+10     3.0  39.106792  4.221582e-25
C(Gender)  4.616417e+09     1.0  15.673357  7.585347e-05
Residual   2.664990e+12  9048.0        NaN           NaN
```

```python
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Fit the model
model = ols('Total_Videos_visionnees ~ C(HDI) + C(Gender)', data=final_df2).fit()

# Perform ANOVA and print the table
anova_table = sm.stats.anova_lm(model, typ=2)

# Add significance level stars
def add_stars(p):
    if p < 0.001:
        return '***'
    elif p < 0.01:
        return '**'
    elif p < 0.05:
        return '*'
    else:
        return '.'

anova_table['stars'] = [add_stars(p) for p in anova_table['PR(>F)']]
print(anova_table)
```

```
                 sum_sq      df          F        PR(>F) stars
C(HDI)     3.455544e+10     3.0  39.106792  4.221582e-25   ***
C(Gender)  4.616417e+09     1.0  15.673357  7.585347e-05   ***
Residual   2.664990e+12  9048.0        NaN           NaN     .
```

Pour obtenir les statistiques inférentielles, on peut utiliser la fonction summaryde la bibliothèque statsmodels. Cette fonction affiche les coefficients de la régression, l'erreur standard, les valeurs de t et de p, l'intervalle de confiance, et d'autres statistiques applicables.

```python
# Affiche le tableau de résumé
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Total_Videos_visionnees   R-squared:                       0.016
Model:                                 OLS   Adj. R-squared:                  0.015
Method:                      Least Squares   F-statistic:                     36.55
Date:                     Tue, 04 Jul 2023   Prob (F-statistic):           2.34e-30
Time:                             23:10:46   Log-Likelihood:            -1.0111e+05
No. Observations:                     9053   AIC:                         2.022e+05
Df Residuals:                         9048   BIC:                         2.023e+05
Df Model:                                4
Covariance Type:                 nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
Intercept             1.82e+04    512.940     35.478      0.000    1.72e+04    1.92e+04
C(HDI)[T.H]          3121.5829   1101.389      2.834      0.005     962.612    5280.554
C(HDI)[T.M]          2616.9782   1047.104      2.499      0.012     564.417    4669.540
C(HDI)[T.TH]         5737.8907    552.637     10.383      0.000    4654.597    6821.184
C(Gender)[T.une femme] 1537.7852  388.432      3.959      0.000     776.371    2299.199
==============================================================================
Omnibus:                     1999.490   Durbin-Watson:                   0.522
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3624.403
Skew:                           1.430   Prob(JB):                         0.00
Kurtosis:                       4.195   Cond. No.                         9.68
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```python
# Obtenir le sommaire sous forme de dataframe
summary_df = pd.read_html(model.summary().tables[1].as_html(), header=0, index_col=0)[0]

# Définir une fonction pour mapper les p-valeurs à des étoiles
def significance(p):
    if p < 0.001:
        return '***'
    elif p < 0.01:
        return '**'
    elif p < 0.05:
        return '*'
    else:
        return ''

# Appliquer la fonction à la colonne de p-valeur
summary_df['P>|t|'] = summary_df['P>|t|'].apply(significance)

# Afficher le tableau
print(summary_df)
```

```
                             coef   std err       t P>|t|     [0.025  \
Intercept              18200.0000   512.940  35.478   ***  17200.000
C(HDI)[T.H]             3121.5829  1101.389   2.834    **    962.612
C(HDI)[T.M]             2616.9782  1047.104   2.499     *    564.417
C(HDI)[T.TH]            5737.8907   552.637  10.383   ***   4654.597
C(Gender)[T.une femme]  1537.7852   388.432   3.959   ***    776.371

                           0.975]
Intercept               19200.000
C(HDI)[T.H]              5280.554
C(HDI)[T.M]              4669.540
C(HDI)[T.TH]             6821.184
C(Gender)[T.une femme]   2299.199
```

```python
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Créez le modèle avec le terme d'interaction
model_interaction = smf.ols('Total_Videos_visionnees ~ C(HDI) * C(Gender)', data=final_df2).fit()

# Affichez le résumé du modèle
print(model_interaction.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Total_Videos_visionnees   R-squared:                       0.016
Model:                                 OLS   Adj. R-squared:                  0.015
Method:                      Least Squares   F-statistic:                     21.20
Date:                     Tue, 04 Jul 2023   Prob (F-statistic):           1.53e-28
Time:                             23:19:24   Log-Likelihood:            -1.0111e+05
No. Observations:                     9053   AIC:                         2.022e+05
Df Residuals:                         9045   BIC:                         2.023e+05
Df Model:                                7
Covariance Type:                 nonrobust
===================================================================================================
                                       coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------
Intercept                         1.824e+04    555.378     32.845      0.000    1.72e+04    1.93e+04
C(HDI)[T.H]                       3738.8361   1384.949      2.700      0.007    1024.022    6453.650
C(HDI)[T.M]                       1842.3146   1221.242      1.509      0.131    -551.597    4236.226
C(HDI)[T.TH]                      5698.3002    609.085      9.356      0.000    4504.356    6892.244
C(Gender)[T.une femme]            1263.5481   1391.356      0.908      0.364   -1463.824    3990.920
C(HDI)[T.H]:C(Gender)[T.une femme] -1324.3341 2414.323    -0.549      0.583   -6056.953    3408.285
C(HDI)[T.M]:C(Gender)[T.une femme]  2755.3151 2439.510     1.129      0.259   -2026.676    7537.306
C(HDI)[T.TH]:C(Gender)[T.une femme]  262.5290 1454.060     0.181      0.857   -2587.757    3112.815
==============================================================================
Omnibus:                     1997.472   Durbin-Watson:                   0.522
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3618.520
Skew:                           1.429   Prob(JB):                         0.00
Kurtosis:                       4.193   Cond. No.                         25.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```python
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Créez le modèle avec le terme d'interaction
model_interaction = smf.ols('Total_Videos_visionnees ~ C(HDI) * C(Gender)', data=final_df2).fit()
# Obtenez le résumé du modèle sous forme de DataFrame
summary_df = pd.read_html(model_interaction.summary().tables[1].as_html(), header=0, index_col=0)[0]

# Ajoutez une colonne pour les étoiles d'importance
summary_df['Significance'] = ''
summary_df.loc[summary_df['P>|t|'] <= 0.001, 'Significance'] = '***'
summary_df.loc[(summary_df['P>|t|'] > 0.001) & (summary_df['P>|t|'] <= 0.01), 'Significance'] = '**'
summary_df.loc[(summary_df['P>|t|'] > 0.01) & (summary_df['P>|t|'] <= 0.05), 'Significance'] = '*'
summary_df.loc[(summary_df['P>|t|'] > 0.05) & (summary_df['P>|t|'] <= 0.1), 'Significance'] = '.'
summary_df.loc[summary_df['P>|t|'] > 0.1, 'Significance'] = ''

# Imprimez le DataFrame
print(summary_df)
```

```
                                          coef   std err       t  P>|t|  \
Intercept                           18240.0000   555.378  32.845  0.000
C(HDI)[T.H]                          3738.8361  1384.949   2.700  0.007
C(HDI)[T.M]                          1842.3146  1221.242   1.509  0.131
C(HDI)[T.TH]                         5698.3002   609.085   9.356  0.000
C(Gender)[T.une femme]               1263.5481  1391.356   0.908  0.364
C(HDI)[T.H]:C(Gender)[T.une femme]  -1324.3341  2414.323  -0.549  0.583
C(HDI)[T.M]:C(Gender)[T.une femme]   2755.3151  2439.510   1.129  0.259
C(HDI)[T.TH]:C(Gender)[T.une femme]   262.5290  1454.060   0.181  0.857

                                       [0.025     0.975] Significance
Intercept                           17200.000  19300.000          ***
C(HDI)[T.H]                          1024.022   6453.650           **
C(HDI)[T.M]                          -551.597   4236.226
C(HDI)[T.TH]                         4504.356   6892.244          ***
C(Gender)[T.une femme]              -1463.824   3990.920
C(HDI)[T.H]:C(Gender)[T.une femme]  -6056.953   3408.285
C(HDI)[T.M]:C(Gender)[T.une femme]  -2026.676   7537.306
C(HDI)[T.TH]:C(Gender)[T.une femme] -2587.757   3112.815
```

6 R´egression logistique

Pour la régression logistique, notre variable dépendante doit être transformée en une variable dichotomique (0 ou 1). Supposons que nous utilisions la variable "succès", qui indique si un utilisateur a réussi à obtenir un certificat (1 pour le succès, 0 pour l'échec).

```python
final_df2['Success'] = final_df2['Completer']

model = smf.glm(formula="Success ~ C(Gender) + C(HDI)",
                data=final_df2,
                family=sm.families.Binomial()).fit()

print(model.summary())
```

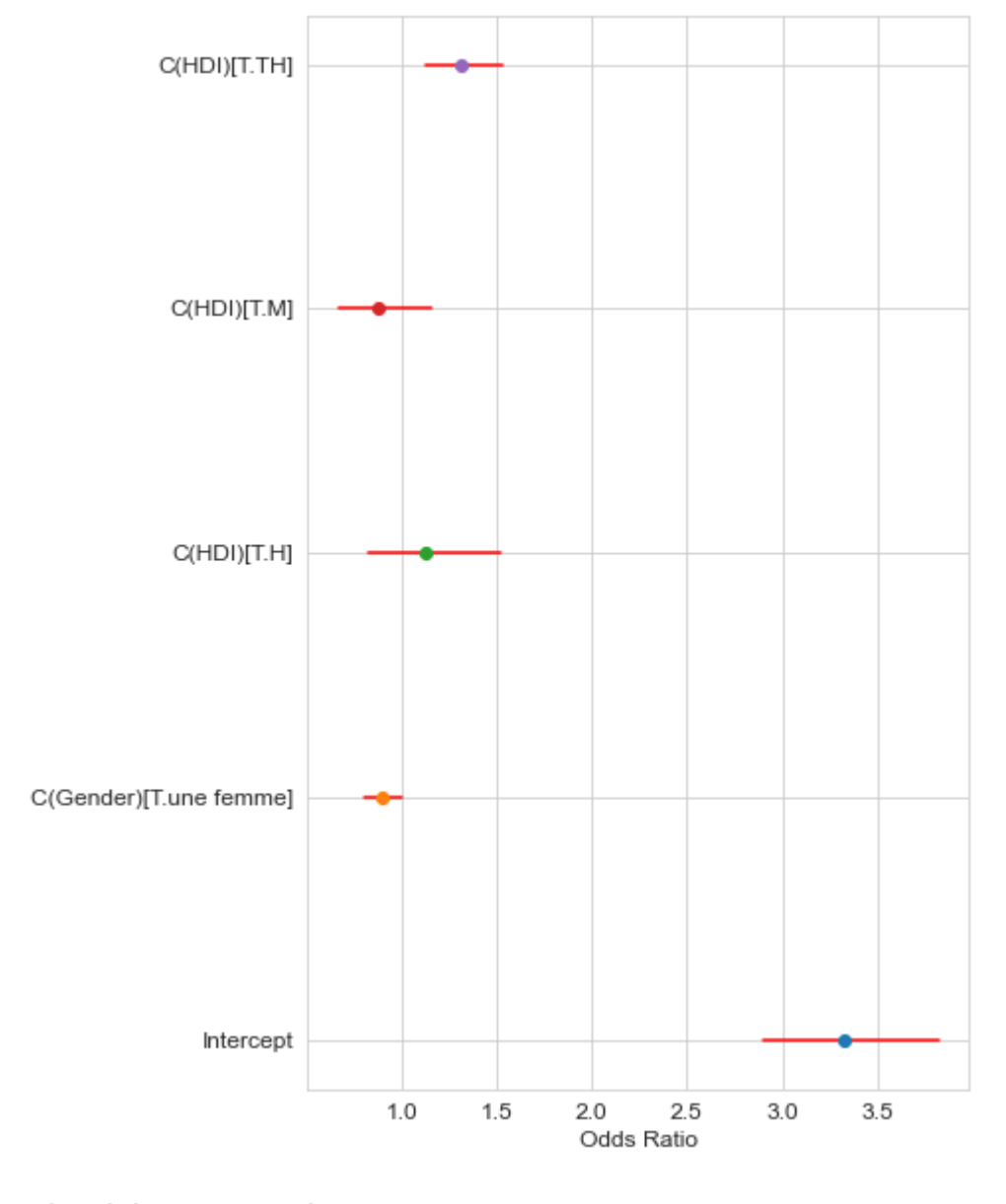
```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                Success   No. Observations:                 9053
Model:                            GLM   Df Residuals:                     9048
Model Family:                Binomial   Df Model:                            4
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -4524.8
Date:                Tue, 04 Jul 2023   Deviance:                       9049.7
Time:                        23:28:13   Pearson chi2:                 9.05e+03
No. Iterations:                     4   Pseudo R-squ. (CS):           0.002517
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
Intercept                  1.2019      0.071     17.001      0.000       1.063       1.340
C(Gender)[T.une femme]    -0.1039      0.056     -1.847      0.065      -0.214       0.006
C(HDI)[T.H]                0.1194      0.155      0.772      0.440      -0.184       0.423
C(HDI)[T.M]               -0.1281      0.140     -0.913      0.361      -0.403       0.147
C(HDI)[T.TH]               0.2753      0.077      3.577      0.000       0.124       0.426
==========================================================================================
```

```python
# Calculate Odds Ratios
OR = np.exp(model.params)

# Calculate lower and upper confidence intervals
CI_lower = np.exp(model.conf_int()[0])
CI_upper = np.exp(model.conf_int()[1])

# Create a DataFrame for the Odds Ratios and Confidence Intervals
results_df = pd.DataFrame({"Odds Ratio": OR,
                           "CI lower": CI_lower,
                           "CI upper": CI_upper,
                           "p-value": model.pvalues})

# Print the resulting DataFrame
print(results_df)
```

```
                        Odds Ratio  CI lower  CI upper       p-value
Intercept                 3.326542  2.896113  3.820942  8.050625e-65
C(Gender)[T.une femme]    0.901321  0.807217  1.006397  6.480076e-02
C(HDI)[T.H]               1.126845  0.832081  1.526027  4.402019e-01
C(HDI)[T.M]               0.879802  0.668313  1.158218  3.613026e-01
C(HDI)[T.TH]              1.316952  1.132520  1.531419  3.480960e-04
```

Forest plot

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 10))

# Boucle pour dessiner chaque ligne du forest plot
for i in range(len(results_df)):
    plt.plot([results_df['CI lower'][i], results_df['CI upper'][i]], [i, i], color='red')  # intervalle de confiance
    plt.plot(results_df['Odds Ratio'][i], i, 'o')  # odds ratio

plt.yticks(range(len(results_df)), results_df.index)
plt.xlabel('Odds Ratio')
plt.grid(True)
plt.savefig("od.png")
plt.show()
```



Obtenir les rapports de cotes :

```python
# Les odds ratios sont l'exponentielle des coefficients du modèle
odds_ratios = np.exp(model.params)
print(odds_ratios)
```

```
Intercept                 3.326542
C(Gender)[T.une femme]    0.901321
C(HDI)[T.H]               1.126845
C(HDI)[T.M]               0.879802
C(HDI)[T.TH]              1.316952
dtype: float64
```

Obtenir les intervalles de confiance pour les odds ratios :

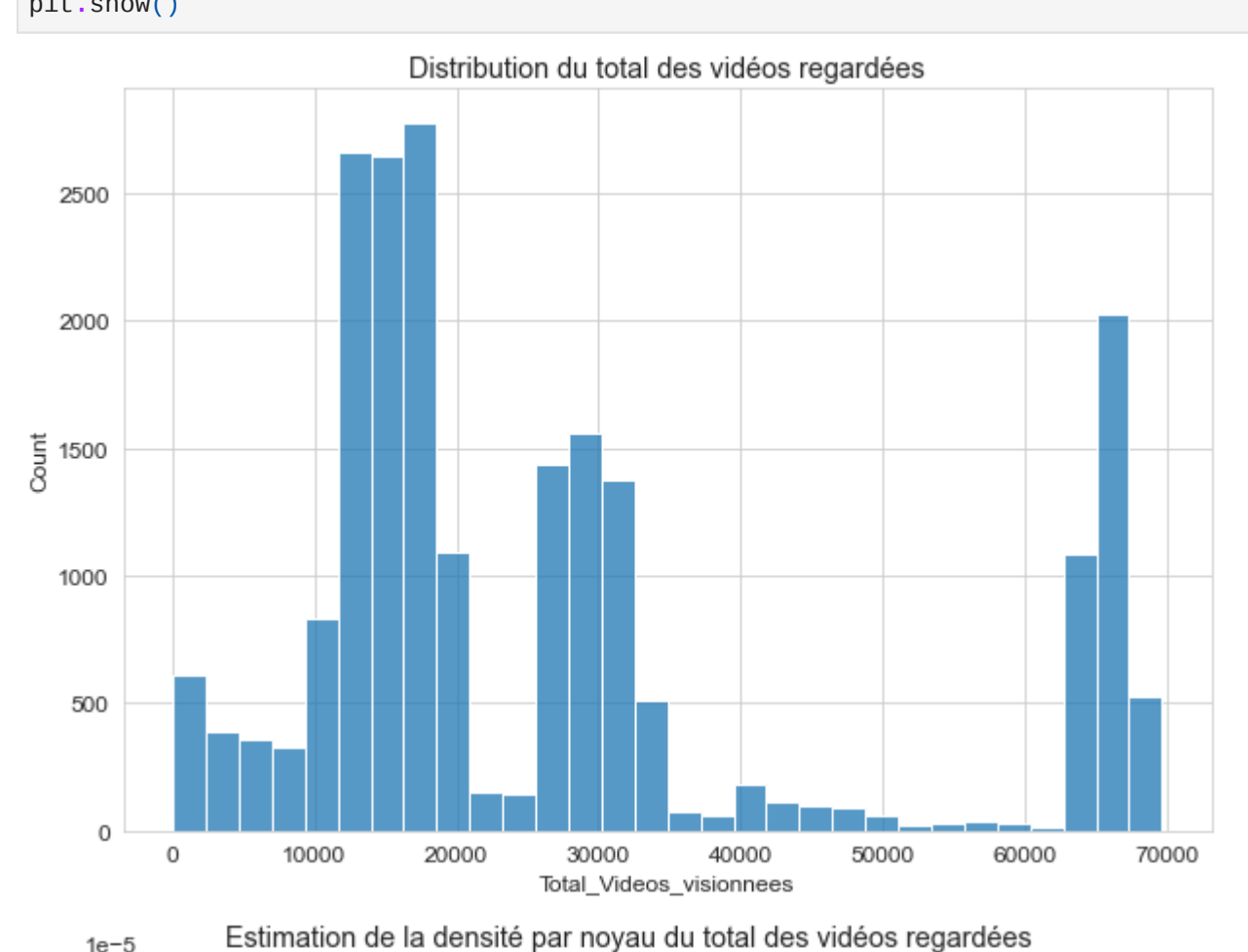
```python
# Obtenir les intervalles de confiance pour les coefficients du modèle
conf = model.conf_int()

# Obtenir les intervalles de confiance pour les odds ratios
conf['OR'] = odds_ratios
conf.columns = ['2.5%', '97.5%', 'OR']

# Prendre l'exponentielle des intervalles de confiance
conf['2.5%'] = np.exp(conf['2.5%'])
conf['97.5%'] = np.exp(conf['97.5%'])
print(conf)
```

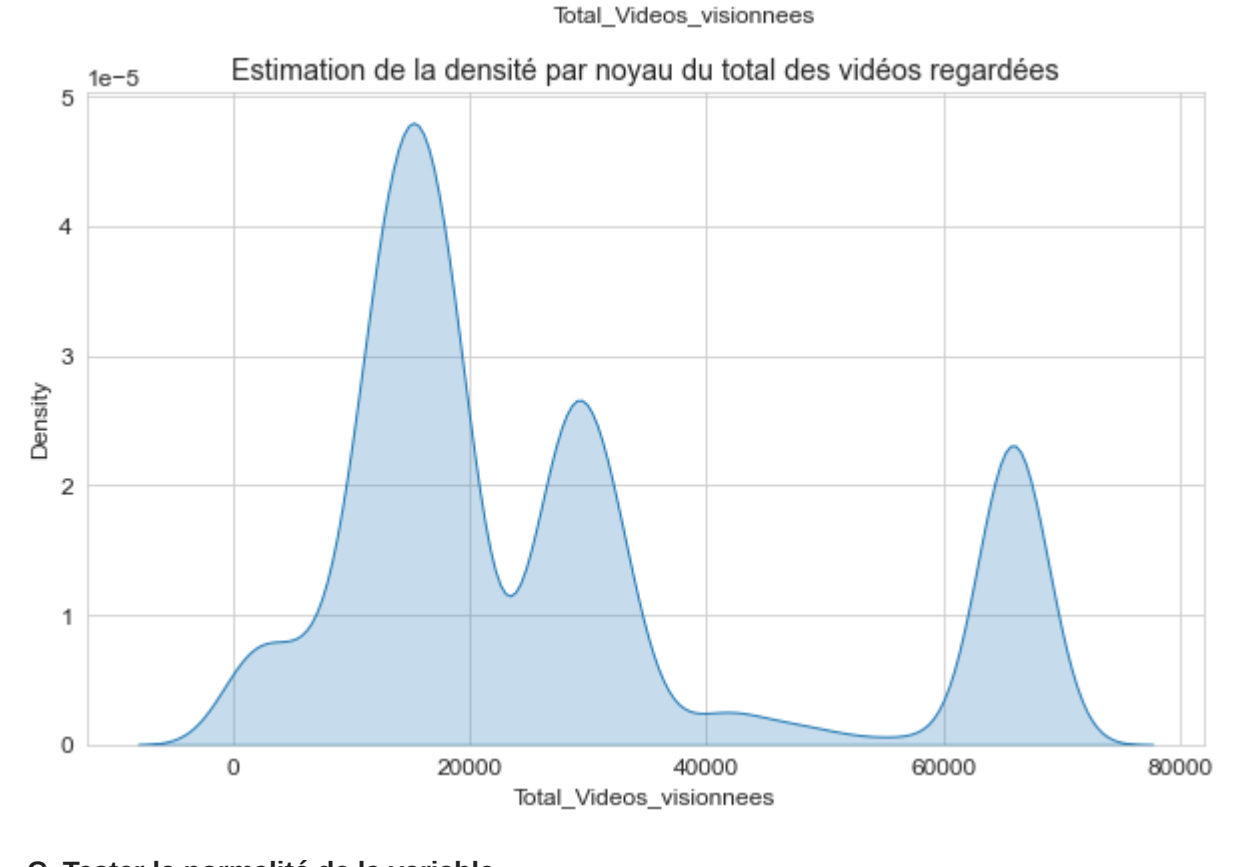
```
                            2.5%     97.5%        OR
Intercept               2.896113  3.820942  3.326542
C(Gender)[T.une femme]  0.807217  1.006397  0.901321
C(HDI)[T.H]             0.832081  1.526027  1.126845
C(HDI)[T.M]             0.668313  1.158218  0.879802
C(HDI)[T.TH]            1.132520  1.531419  1.316952
```

6.2 Donnees de comptage et loi de Poisson

A. Représenter la distribution de la variable

```python
# Histogram
plt.figure(figsize=(10,7))
sns.histplot(final_df2['Total_Videos_visionnees'], kde=False, bins=30)
plt.title('Distribution du total des vidéos regardées')
plt.savefig("est2.png")
plt.show()

# KDE plot
plt.figure(figsize=(10,6))
sns.kdeplot(final_df2['Total_Videos_visionnees'], shade=True)
plt.title("Estimation de la densité par noyau du total des vidéos regardées")
plt.savefig("est.png")
plt.show()
```





C. Tester la normalité de la variable:

```python
# Créer des variables indicatrices pour 'Gender' et 'HDI'
final_df2 = pd.get_dummies(final_df2, columns=['Gender', 'HDI'])
```

```python
print(final_df2.columns)
```

```
Index(['Student_ID', 'birth.year', 'Country', 'Diploma', 'Formation', 'CSP',
       'How.heard', 'Exp.crea', 'Curiosity.MOOC', 'Certif.self.sat',
       ...
       'Auditing', 'Completer', 'Disengaging', 'Success', 'Gender_un homme',
       'Gender_une femme', 'HDI_B', 'HDI_H', 'HDI_M', 'HDI_TH'],
      dtype='object', length=134)
```

```python
import statsmodels.api as sm

# Préparation des variables
X = final_df2[['Gender_une femme', 'HDI_B', 'HDI_H', 'HDI_M', 'HDI_TH']]
X = sm.add_constant(X)
y = final_df2['Total_Videos_visionnees']

# Création du modèle
poisson_model = sm.GLM(y, X, family=sm.families.Poisson())

# Ajustement du modèle
poisson_results = poisson_model.fit()

# Affichage des résultats
print(poisson_results.summary())
```

```
                  Generalized Linear Model Regression Results
===============================================================================
Dep. Variable:     Total_Videos_visionnees   No. Observations:            21312
Model:                                 GLM   Df Residuals:                21306
Model Family:                      Poisson   Df Model:                        5
Link Function:                         Log   Scale:                      1.0000
Method:                               IRLS   Log-Likelihood:        -1.3552e+08
Date:                     Tue, 04 Jul 2023   Deviance:               2.7079e+08
Time:                             23:36:39   Pearson chi2:             2.92e+08
No. Iterations:                          8   Pseudo R-squ. (CS):          1.000
Covariance Type:                 nonrobust
====================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------
const               10.1069      0.001   1.52e+04      0.000      10.106      10.108
Gender_une femme     0.0217      0.000    152.930      0.000       0.021       0.022
HDI_B                0.1841      0.001    276.917      0.000       0.183       0.185
HDI_H               -0.1196      0.001   -157.108      0.000      -0.121      -0.118
HDI_M               -0.1510      0.001   -200.350      0.000      -0.152      -0.150
HDI_TH              -0.0090      0.001    -13.577      0.000      -0.010      -0.008
====================================================================================
```

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.linear_model import LinearRegression

# Initialize a new matplotlib figure with 2x2 subplots.
fig, ax = plt.subplots(2, 2, figsize=(14, 14))

# Histogramme et courbe de densité
sns.histplot(data=final_df2, x='Total_Videos_visionnees', kde=True, bins=50, ax=ax[0, 0])
ax[0, 0].set_title('Histogramme et courbe de densité')

# QQ-Plot
stats.probplot(final_df2['Total_Videos_visionnees'], plot=ax[0, 1])
ax[0, 1].set_title('QQ-Plot')

# Boxplot
sns.boxplot(data=final_df2, x='Total_Videos_visionnees', ax=ax[1, 0])
ax[1, 0].set_title('Boxplot')

# Scatterplot des résidus
model = LinearRegression()
model.fit(final_df2[['HDI_B']], final_df2['Total_Videos_visionnees'])
residuals = final_df2['Total_Videos_visionnees'] - model.predict(final_df2[['HDI_B']])
ax[1, 1].scatter(final_df2['HDI_B'], residuals)
ax[1, 1].set_title('Scatterplot des résidus')
ax[1, 1].set_xlabel('HDI_B')
ax[1, 1].set_ylabel('Résidus')

plt.tight_layout()
plt.savefig("es.png")
plt.show()
```