PLS-SEM(stage2)

```
import pandas as pd
df = pd.read_csv('output.csv')
df
```

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	...	Q32	Q33	Q34	Age	Grade	Gender	Level	Place	Instagram	Page
0	4.0	4.0	3.0	4.0	3.0	4	2	2.0	3.0	2.0	...	2.0	4.0	2.0	34.0	3	2.0	2	1.0	3	3
1	5.0	5.0	4.0	4.0	5.0	5	4	3.0	4.0	4.0	...	5.0	5.0	5.0	17.0	1	1.0	2	2.0	2	3
2	4.0	2.0	1.0	2.0	2.0	1	1	1.0	1.0	2.0	...	3.0	4.0	3.0	37.0	3	2.0	3	2.0	4	2
3	4.0	5.0	4.0	4.0	4.0	2	2	3.0	3.0	3.0	...	2.0	4.0	1.0	33.0	3	1.0	2	2.0	3	2
4	2.0	4.0	3.0	4.0	5.0	2	2	4.0	2.0	1.0	...	2.0	2.0	1.0	21.0	3	1.0	2	1.0	4	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
244	5.0	5.0	3.0	4.0	4.0	4	4	4.0	3.0	4.0	...	4.0	5.0	5.0	50.0	1	1.0	2	2.0	1	3
245	4.0	4.0	2.0	4.0	3.0	2	2	2.0	2.0	2.0	...	2.0	2.0	4.0	31.0	3	2.0	2	2.0	3	3
246	4.0	4.0	2.0	2.0	2.0	2	2	2.0	2.0	2.0	...	2.0	4.0	2.0	35.0	3	2.0	2	2.0	3	2
247	4.0	4.0	3.0	4.0	3.0	3	3	4.0	2.0	4.0	...	2.0	4.0	2.0	31.0	4	2.0	1	1.0	2	2
248	4.0	4.0	1.0	4.0	4.0	2	1	3.0	2.0	3.0	...	2.0	3.0	3.0	32.0	3	1.0	2	1.0	2	1

249 rows × 41 columns

```
print(df.isnull().sum()[df.isnull().sum() > 0])
```

```
Series([], dtype: int64)
```

```
!pip install openpls-engine
```

```
Collecting openpls-engine
  Downloading openpls_engine-1.10.0-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: pandas>=1.5 in /usr/local/lib/python3.13/dist-packages (from openpls-engine) (2.2.3)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.13/dist-packages (from openpls-engine) (2.1.3)
Requirement already satisfied: scipy>=1.10 in /usr/local/lib/python3.13/dist-packages (from openpls-engine) (1.16.3)
Requirement already satisfied: statsmodels>=0.14 in /usr/local/lib/python3.13/dist-packages (from openpls-engine) (0.14.6)
Requirement already satisfied: scikit-learn>=1.2 in /usr/local/lib/python3.13/dist-packages (from openpls-engine) (1.6.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.13/dist-packages (from pandas>=1.5->openpls-engi
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.13/dist-packages (from pandas>=1.5->openpls-engine) (2025.
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.13/dist-packages (from pandas>=1.5->openpls-engine) (202
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.13/dist-packages (from scikit-learn>=1.2->openpls-engine)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.13/dist-packages (from scikit-learn>=1.2->openpls-
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.13/dist-packages (from statsmodels>=0.14->openpls-engine)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.13/dist-packages (from statsmodels>=0.14->openpls-engin
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.13/dist-packages (from python-dateutil>=2.8.2->pandas>=1.5->op
Downloading openpls_engine-1.10.0-py3-none-any.whl (143 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.2/143.2 kB 6.6 MB/s eta 0:00:00
Installing collected packages: openpls-engine
Successfully installed openpls-engine-1.10.0
```

```
from openpls import Plspm
import openpls.config as c
from openpls.mode import Mode
from openpls.scheme import Scheme

# ======================================================
# Structural Model
# ======================================================

structure = c.Structure()

# Expertise affects Parasocial and Trust
structure.add_path(["Expertise"], ["Parasocial"])
structure.add_path(["Expertise"], ["Trust"])

# Parasocial affects Trust
structure.add_path(["Parasocial"], ["Trust"])
```

```
# Trust affects Loyalty
structure.add_path(["Trust"], ["Loyalty"])

# Loyalty affects Purchase Intention
structure.add_path(["Loyalty"], ["Intention"])

# Purchase Intention affects Behavior
structure.add_path(["Intention"], ["Behavior"])


config = c.Config(structure.path(), scaled=True)

# Expertise
config.add_lv(
    "Expertise",
    Mode.A,
    c.MV("Q1"),
    c.MV("Q2"),
    c.MV("Q4"),
    c.MV("Q5"),
)

# Trust
config.add_lv(
    "Trust",
    Mode.A,
    c.MV("Q6"),
    c.MV("Q7"),
)

# Parasocial
config.add_lv(
    "Parasocial",
    Mode.A,
    c.MV("Q3"),
    c.MV("Q8"),
    c.MV("Q9"),
    c.MV("Q10"),
    c.MV("Q11"),
    c.MV("Q12"),
    c.MV("Q13"),
    c.MV("Q14"),
    c.MV("Q15"),
)

# Loyalty
config.add_lv(
    "Loyalty",
    Mode.A,
    c.MV("Q16"),
    c.MV("Q17"),
    c.MV("Q18"),
    c.MV("Q19"),
    c.MV("Q20"),
    c.MV("Q21"),
    c.MV("Q22"),
    c.MV("Q23"),
    c.MV("Q24"),
)

# Purchase Intention
config.add_lv(
    "Intention",
    Mode.A,
    c.MV("Q25"),
    c.MV("Q26"),
    c.MV("Q27"),
    c.MV("Q28"),
    c.MV("Q30"),
    c.MV("Q31"),
)

# Behavior
config.add_lv(
    "Behavior",
    Mode.A,
    c.MV("Q29"),
```

```
    c.MV("Q32"),
    c.MV("Q33"),
    c.MV("Q34"),
)


pls = Plspm(
    df,
    config,
    Scheme.PATH
)
```

```
scores = pls.scores()
print(type(scores))
print(scores.head())
```

```
<class 'pandas.core.frame.DataFrame'>
   Expertise  Parasocial     Trust   Loyalty  Intention  Behavior
0  -0.123799   -0.489494 -0.149559  0.515782  -0.118951 -0.848059
1   1.100025    0.462110  1.388056  1.144461   0.720250  2.184804
2  -1.781591   -2.310830 -2.151053 -1.769147   0.303987  0.386606
3   0.521112    0.332898 -1.135708 -0.340020  -0.771052 -0.780899
4  -0.028228   -0.214422 -1.135708 -1.351656  -1.131914 -1.090314
```

```
stage2_df = scores.copy()

# ======================================================
# Structural Model
# ======================================================

structure = c.Structure()

structure.add_path(["Expertise"], ["Parasocial"])
structure.add_path(["Parasocial"], ["RelationshipQuality"])
#structure.add_path(["Expertise"], ["RelationshipQuality"])
structure.add_path(["RelationshipQuality"], ["BehavioralResponse"])

config = c.Config(structure.path(), scaled=True)

# ======================================================
# Measurement Model
# ======================================================

# Expertise (single indicator)
config.add_lv(
    "Expertise",
    Mode.A,
    c.MV("Expertise")
)

config.add_lv(
    "Parasocial",
    Mode.A,
    c.MV("Parasocial")
)

# Second-order Relationship Quality
config.add_lv(
    "RelationshipQuality",
    Mode.A,
    c.MV("Trust"),
   # c.MV("Parasocial"),
    c.MV("Loyalty")
)

# Second-order Behavioral Response
config.add_lv(
    "BehavioralResponse",
    Mode.A,
    c.MV("Intention"),
    c.MV("Behavior")
)

# ======================================================
# Estimate Stage 2 Model
# ======================================================
```

```
pls_stage2 = Plspm(
    stage2_df,
    config,
    Scheme.PATH
)

# ====================================================
# Results
# ====================================================

print(pls_stage2.outer_model())
print(pls_stage2.inner_summary())
print(pls_stage2.path_coefficients())
```

```
             weight   loading  communality  redundancy
Behavior    0.480726  0.952771     0.907773    0.511173
Expertise   1.002014  1.000000     1.000000    0.000000
Intention   0.563242  0.965824     0.932816    0.525275
Loyalty     0.571964  0.941084     0.885639    0.683589
Parasocial  1.002014  1.000000     1.000000    0.553135
Trust       0.502468  0.922940     0.851818    0.657483
                         type  r_squared  r_squared_adj  block_communality  \
BehavioralResponse   Endogenous   0.563107       0.561338           0.920295
Expertise             Exogenous   0.000000       0.000000           1.000000
Parasocial           Endogenous   0.553135       0.551326           1.000000
RelationshipQuality  Endogenous   0.771859       0.770936           0.868729

                     mean_redundancy       ave         bic
BehavioralResponse          0.518224  0.920295 -196.155706
Expertise                   0.000000  1.000000         NaN
Parasocial                  0.553135  1.000000 -190.536518
RelationshipQuality         0.670536  0.868729 -357.937344
                     Expertise  Parasocial  RelationshipQuality  \
Expertise             0.000000    0.000000             0.000000
Parasocial            0.743731    0.000000             0.000000
RelationshipQuality   0.000000    0.878555             0.000000
BehavioralResponse    0.000000    0.000000             0.750404

                     BehavioralResponse
Expertise                           0.0
Parasocial                          0.0
RelationshipQuality                 0.0
BehavioralResponse                  0.0
```

```
fs = pls_stage2.scores()
Fscores = fs.copy()
```

Bootstrapping

```
pls_stage2 = Plspm(
    stage2_df,
    config,
    Scheme.PATH,
    bootstrap=True,
    bootstrap_iterations=5000
)
```

```
bootstrap_result = pls_stage2.bootstrap()
```

```
print('loading')
print(bootstrap_result.loading())
```

```
loading
            original      mean     std.error  perc.025  perc.975       t stat.
Expertise   1.000000  1.000000  7.558529e-16  1.000000  1.000000  1.323009e+15
Parasocial  1.000000  1.000000  3.979893e-16  1.000000  1.000000  2.512630e+15
Trust       0.922940  0.922947  8.961004e-03  0.903537  0.938545  1.029951e+02
Loyalty     0.941084  0.941340  5.065455e-03  0.930891  0.950609  1.857847e+02
Intention   0.965824  0.965961  3.760293e-03  0.958117  0.972880  2.568481e+02
Behavior    0.952771  0.952691  6.591650e-03  0.938312  0.964224  1.445421e+02
```

```
print('paths')
print(bootstrap_result.paths())
```

```
paths
                                          original      mean  std.error  \
Expertise -> Parasocial                   0.743731  0.743491   0.035323
Parasocial -> RelationshipQuality         0.878555  0.878493   0.013304
RelationshipQuality -> BehavioralResponse  0.750404  0.750762   0.029028

                                          perc.025  perc.975    t stat.
Expertise -> Parasocial                   0.666049  0.803155  21.055191
Parasocial -> RelationshipQuality         0.850191  0.903268  66.035954
RelationshipQuality -> BehavioralResponse  0.689934  0.804899  25.850907
```

```
print('r_squared()')
print(bootstrap_result.r_squared())
print('total_effects()')
print(bootstrap_result.total_effects())


print('weights()')
print(bootstrap_result.weights())
```

```
r_squared()
                     original      mean  std.error  perc.025  perc.975  \
Parasocial           0.553135  0.554027   0.051966  0.443621  0.645057
RelationshipQuality  0.771859  0.771926   0.023330  0.722825  0.815893
BehavioralResponse   0.563107  0.564485   0.043382  0.476008  0.647862

                       t stat.
Parasocial           10.644096
RelationshipQuality  33.083906
BehavioralResponse   12.980342
total_effects()
                                          original      mean  std.error  \
Expertise -> Parasocial                   0.743731  0.743491   0.035323
Expertise -> RelationshipQuality          0.653408  0.653334   0.035976
Expertise -> BehavioralResponse           0.490321  0.490498   0.033069
Parasocial -> RelationshipQuality         0.878555  0.878493   0.013304
Parasocial -> BehavioralResponse          0.659272  0.659591   0.028615
RelationshipQuality -> BehavioralResponse  0.750404  0.750762   0.029028

                                          perc.025  perc.975    t stat.
Expertise -> Parasocial                   0.666049  0.803155  21.055191
Expertise -> RelationshipQuality          0.575962  0.718928  18.162523
Expertise -> BehavioralResponse           0.421542  0.554278  14.827354
Parasocial -> RelationshipQuality         0.850191  0.903268  66.035954
Parasocial -> BehavioralResponse          0.601473  0.712370  23.039512
RelationshipQuality -> BehavioralResponse  0.689934  0.804899  25.850907
weights()
            original      mean     std.error  perc.025  perc.975       t stat.
Expertise   1.002014  1.002014  7.801204e-15  1.002014  1.002014  1.284435e+14
Parasocial  1.002014  1.002014  7.769417e-15  1.002014  1.002014  1.289690e+14
Trust       0.502468  0.502220  6.729297e-03  0.488918  0.515339  7.466876e+01
Loyalty     0.571964  0.572060  1.031179e-02  0.553546  0.593806  5.546701e+01
Intention   0.563242  0.563484  1.076508e-02  0.544391  0.586939  5.232121e+01
Behavior    0.480726  0.480430  8.704634e-03  0.461792  0.495917  5.522641e+01
```

```
# All indirect effects from Expertise to BehavioralResponse
bootstrap_result.specific_indirect_effects(
    source="Expertise",
    target="BehavioralResponse"
)
```

from	to	via	original	mean	std.error	perc.lower	perc.upper	t stat
Expertise -> Parasocial -> RelationshipQuality	Expertise -> BehavioralResponse	(Parasocial, RelationshipQuality)	0.490321	0.490498	0.033069	0.421542	0.554278	14.827354

```
bootstrap_result.specific_indirect_effects(
    source="Expertise",
    target="BehavioralResponse",
    through=["Parasocial", "RelationshipQuality"]
)
```

	from	to	via	original	mean	std.error	perc.lower	perc.upper	t stat
Expertise -> Parasocial -> ...	...	...	(Parasocial, ...	...	...	...	...	...	...

pls_stage2 results

```
#f2 = pls_stage2.f_squared()
#print(f2)
print('-----------------------------------------------------')
q2 = pls_stage2.q_squared()
print(q2)
print('-----------------------------------------------------')
#vif = pls_stage2.vif()
#print(vif)
print('-----------------------------------------------------')
#fit = pls_stage2.model_fit()
#print(fit)
print('-----------------------------------------------------')
inner = pls_stage2.inner_summary()
print(inner)
```

```
-----------------------------------------------------
                     q_squared
lv
Parasocial            0.546845
RelationshipQuality   0.662777
BehavioralResponse    0.503848
-----------------------------------------------------
-----------------------------------------------------
-----------------------------------------------------
                          type  r_squared  r_squared_adj  block_communality  \
BehavioralResponse   Endogenous   0.563107       0.561338           0.920295
Expertise             Exogenous   0.000000       0.000000           1.000000
Parasocial           Endogenous   0.553135       0.551326           1.000000
RelationshipQuality  Endogenous   0.771859       0.770936           0.868729

                     mean_redundancy       ave         bic
BehavioralResponse          0.518224  0.920295 -196.155706
Expertise                   0.000000  1.000000         NaN
Parasocial                  0.553135  1.000000 -190.536518
RelationshipQuality         0.670536  0.868729 -357.937344
```

```
f2 = pls_stage2.f_squared()

# Table
print(f2.table())

# Matrix (optional)
print(f2.matrix())
```

```
                                                  from  \
Expertise -> Parasocial                       Expertise
Parasocial -> RelationshipQuality            Parasocial
RelationshipQuality -> BehavioralResponse  RelationshipQuality

                                                    to  \
Expertise -> Parasocial                       Parasocial
Parasocial -> RelationshipQuality          RelationshipQuality
RelationshipQuality -> BehavioralResponse   BehavioralResponse

                                           r_squared_full  r_squared_reduced  \
Expertise -> Parasocial                         0.553135                0.0
Parasocial -> RelationshipQuality               0.771859                0.0
RelationshipQuality -> BehavioralResponse       0.563107                0.0

                                           f_squared effect_size
Expertise -> Parasocial                     1.237814       large
Parasocial -> RelationshipQuality           3.383257       large
RelationshipQuality -> BehavioralResponse   1.288889       large
                     Expertise  Parasocial  RelationshipQuality  \
Expertise                  NaN         NaN                  NaN
Parasocial            1.237814         NaN                  NaN
RelationshipQuality        NaN    3.383257                  NaN
BehavioralResponse         NaN         NaN             1.288889

                     BehavioralResponse
```

```
Expertise                       NaN
Parasocial                      NaN
RelationshipQuality             NaN
BehavioralResponse              NaN
```

```
vif = pls_stage2.vif()

# Structural VIF
print(vif.inner())

# Measurement VIF (indicator VIF)
print(vif.items())
```

```
{}
                   lv  indicator       vif
0  RelationshipQuality      Trust  2.198838
1  RelationshipQuality    Loyalty  2.198838
2   BehavioralResponse  Intention  3.425969
3   BehavioralResponse   Behavior  3.425969
```

```
fit = pls_stage2.model_fit()

print(fit.summary())
```

```
               srmr     d_uls
saturated  0.066439  0.066213
```

```
print(pls_stage2.model_fit().srmr())
```

```
0.06643930326994701
```

```
htmt = pls_stage2.htmt()

print(htmt.matrix())
```

```
                    Expertise  Parasocial  RelationshipQuality  \
Expertise                 NaN         NaN                  NaN
Parasocial                NaN         NaN                  NaN
RelationshipQuality       NaN         NaN                  NaN
BehavioralResponse        NaN         NaN             0.839824

                    BehavioralResponse
Expertise                          NaN
Parasocial                         NaN
RelationshipQuality           0.839824
BehavioralResponse                 NaN
```

```
print(htmt.pairs())
```

```
                 lv_a                lv_b      htmt
0  RelationshipQuality  BehavioralResponse  0.839824
```

Clustring

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

X = Fscores[['Expertise', 'Parasocial',
             'RelationshipQuality',
             'BehavioralResponse']]

scaler = StandardScaler()

X_scaled = pd.DataFrame(
    scaler.fit_transform(X),
    columns=X.columns,
    index=X.index
)

print(X_scaled.head())
```

```
   Expertise  Parasocial  RelationshipQuality  BehavioralResponse
0  -0.123799   -0.489494             0.219418           -0.473728
1   1.100025    0.462110             1.349327            1.453039
```

```
2  -1.781591   -2.310830           -2.088518           0.356352
3   0.521112    0.332898           -0.763598          -0.808059
4  -0.028228   -0.214422           -1.341055          -1.159348
```

```
print(X_scaled.mean())
print(X_scaled.std())
```

```
Expertise             -3.210283e-17
Parasocial            -2.853585e-17
RelationshipQuality    3.566982e-18
BehavioralResponse    -1.426793e-17
dtype: float64
Expertise              1.002014
Parasocial             1.002014
RelationshipQuality    1.002014
BehavioralResponse     1.002014
dtype: float64
```

```
print(X_scaled.std(ddof=0))
```

```
Expertise              1.0
Parasocial             1.0
RelationshipQuality    1.0
BehavioralResponse     1.0
dtype: float64
```

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score

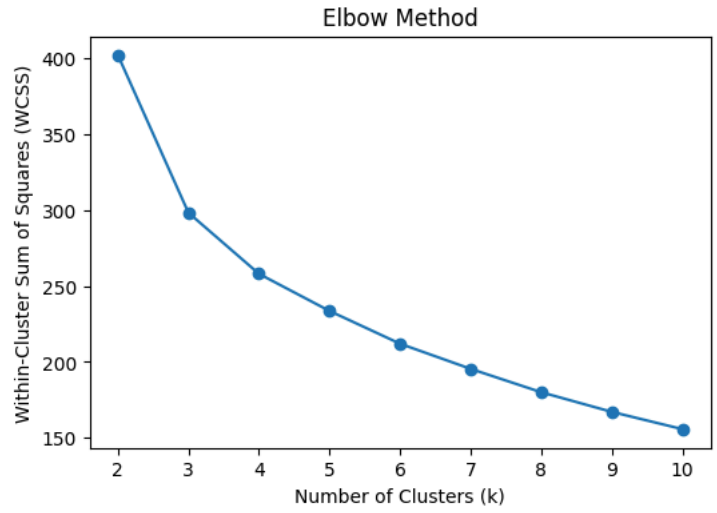
K = range(2, 11)

inertia = []
silhouette = []
calinski = []
davies = []

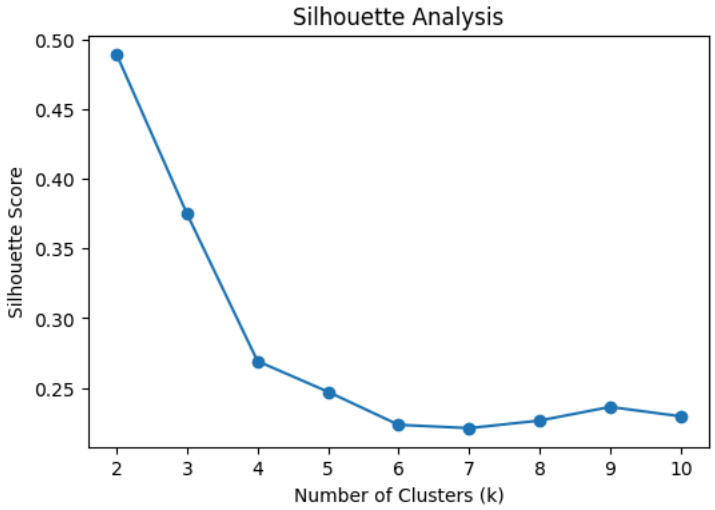
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=20)
    labels = kmeans.fit_predict(X_scaled)

    inertia.append(kmeans.inertia_)
    silhouette.append(silhouette_score(X_scaled, labels))
    calinski.append(calinski_harabasz_score(X_scaled, labels))
    davies.append(davies_bouldin_score(X_scaled, labels))
```

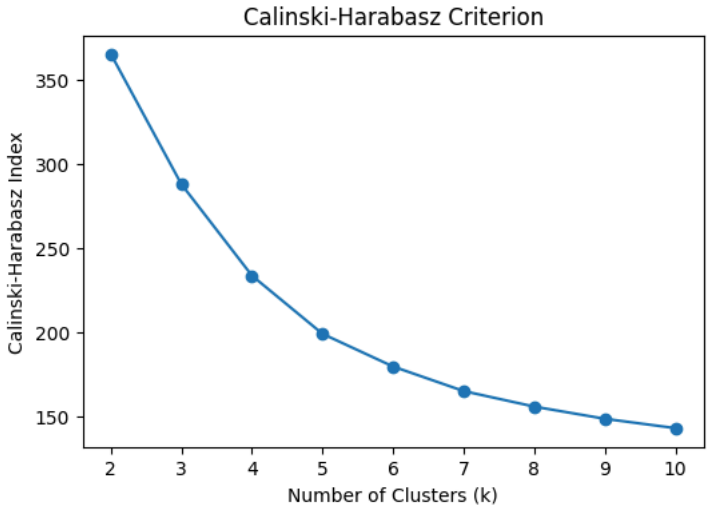
```
plt.figure(figsize=(6,4))
plt.plot(K, inertia, marker='o')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Within-Cluster Sum of Squares (WCSS)")
plt.title("Elbow Method")
plt.show()
```



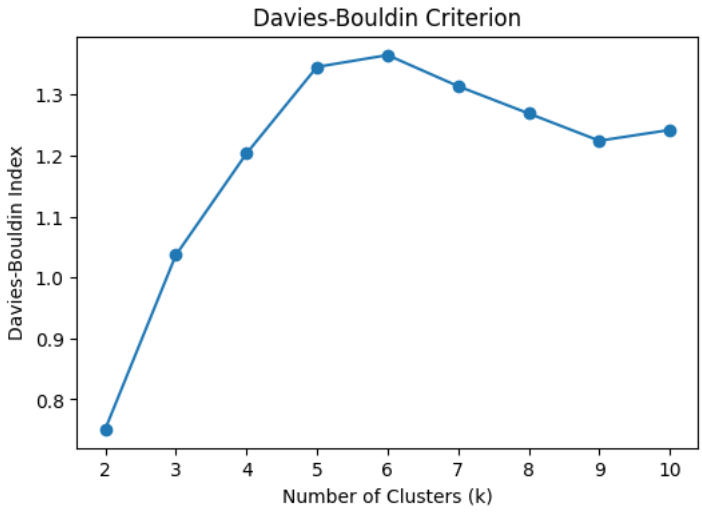
```
plt.figure(figsize=(6,4))
plt.plot(K, silhouette, marker='o')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Analysis")
plt.show()
```




```
plt.figure(figsize=(6,4))
plt.plot(K, calinski, marker='o')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Calinski-Harabasz Index")
plt.title("Calinski-Harabasz Criterion")
plt.show()
```




```
plt.figure(figsize=(6,4))
plt.plot(K, davies, marker='o')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Davies-Bouldin Index")
plt.title("Davies-Bouldin Criterion")
plt.show()
```




```
import pandas as pd

results = pd.DataFrame({
    "k": list(K),
    "WCSS": inertia,
    "Silhouette": silhouette,
    "Calinski-Harabasz": calinski,
    "Davies-Bouldin": davies
})

print(results.round(3))
```

```
    k     WCSS  Silhouette  Calinski-Harabasz  Davies-Bouldin
0   2  402.019       0.489            364.942           0.751
1   3  298.235       0.375            287.777           1.037
2   4  257.988       0.269            233.619           1.203
3   5  233.560       0.247            199.131           1.345
4   6  211.975       0.224            179.755           1.365
5   7  195.438       0.221            165.215           1.314
6   8  180.061       0.227            156.011           1.269
7   9  167.121       0.236            148.793           1.224
8  10  155.762       0.230            143.251           1.242
```

```
from sklearn.cluster import KMeans

kmeans = KMeans(
    n_clusters=2,
    random_state=42,
    n_init=20
)

clusters = kmeans.fit_predict(X_scaled)

Fscores['Cluster'] = clusters
```

```
print(Fscores['Cluster'].value_counts())
```

```
Cluster
1    138
0    111
Name: count, dtype: int64
```

```
Fscores.groupby('Cluster').mean().round(3)
```

Cluster	Expertise	Parasocial	RelationshipQuality	BehavioralResponse
0	-0.822	-0.902	-0.947	-0.762
1	0.661	0.725	0.761	0.613

```
df_model = df.copy()
df_model["Cluster"] = Fscores["Cluster"].values
```

```
# Add cluster labels
df_model["Cluster_Label"] = df_model["Cluster"].map({
    0: "Low Engagement",
    1: "High Engagement"
})
```

```
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
```

```
X = df_model[
    [
        "Age",
        "Grade",
        "Gender",
        "Level",
        "Place",
        "Instagram",
        "Page"
    ]
]

y = df_model["Cluster"]

tree = DecisionTreeClassifier(
    max_depth=4,          # publication-friendly
    min_samples_leaf=10,  # avoid tiny groups
    random_state=42
)

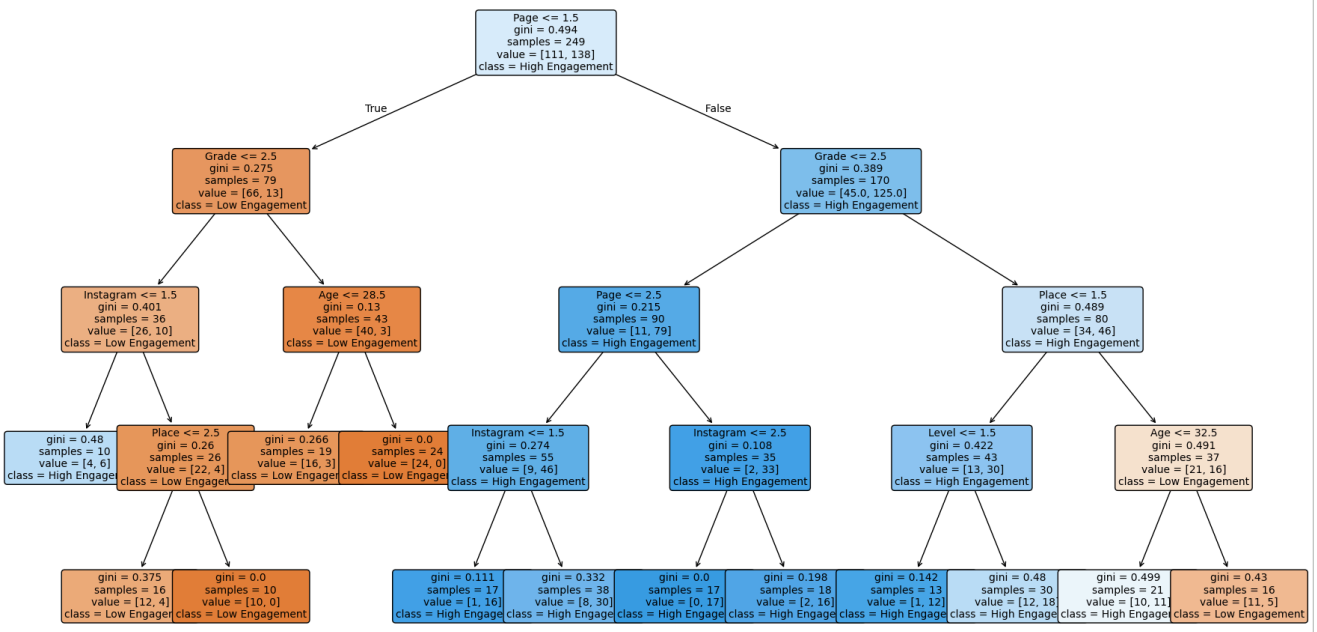
tree.fit(X, y)
```

```
DecisionTreeClassifier(max_depth=4, min_samples_leaf=10, random_state=42)
```

```
plt.figure(figsize=(22,12))

plot_tree(
    tree,
    feature_names=X.columns,
    class_names=["Low Engagement","High Engagement"],
    filled=True,
    rounded=True,
    fontsize=10
)

plt.show()
```




```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Must follow the same order as X.columns:
# Age, Grade_new, Gender, Level, Place, Instagram, Page
feature_labels = [
    "Age",
    "Educational attainment",
    "Gender",
    "Socioeconomic class",
    "Place of residence",
    "Daily Instagram use",
    "Weekly food-blogger browsing time"
]

class_labels = [
    "Low Engagement",
    "High Engagement"
]

plt.figure(figsize=(28, 14))

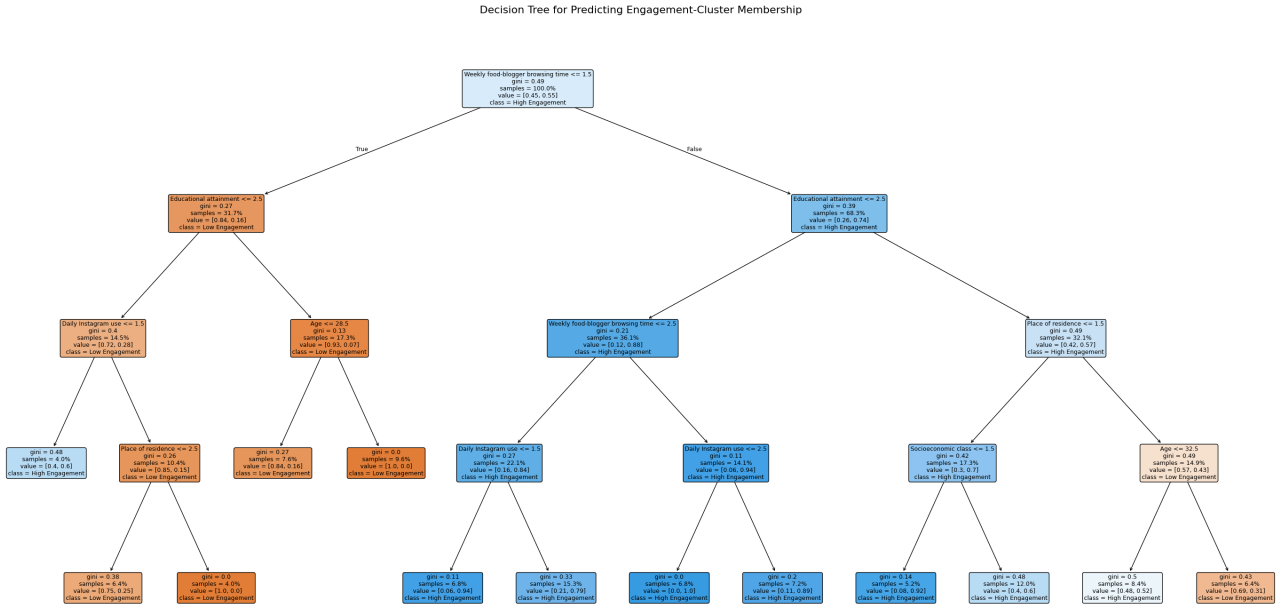
plot_tree(
    tree,
    feature_names=feature_labels,
    class_names=class_labels,
    filled=True,
    rounded=True,
    fontsize=9,
    proportion=True,
    precision=2
)

plt.title(
    "Decision Tree for Predicting Engagement-Cluster Membership",
```

```
    fontsize=16,
    pad=20
)

plt.tight_layout()

plt.show()
```




```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

feature_labels = [
    "Age",
    "Educational attainment",
    "Gender",
    "Socioeconomic class",
    "Place of residence",
    "Daily Instagram use",
    "Weekly food-blogger browsing time"
]

class_labels = [
    "Low Engagement",
    "High Engagement"
]

fig, ax = plt.subplots(figsize=(40, 24))

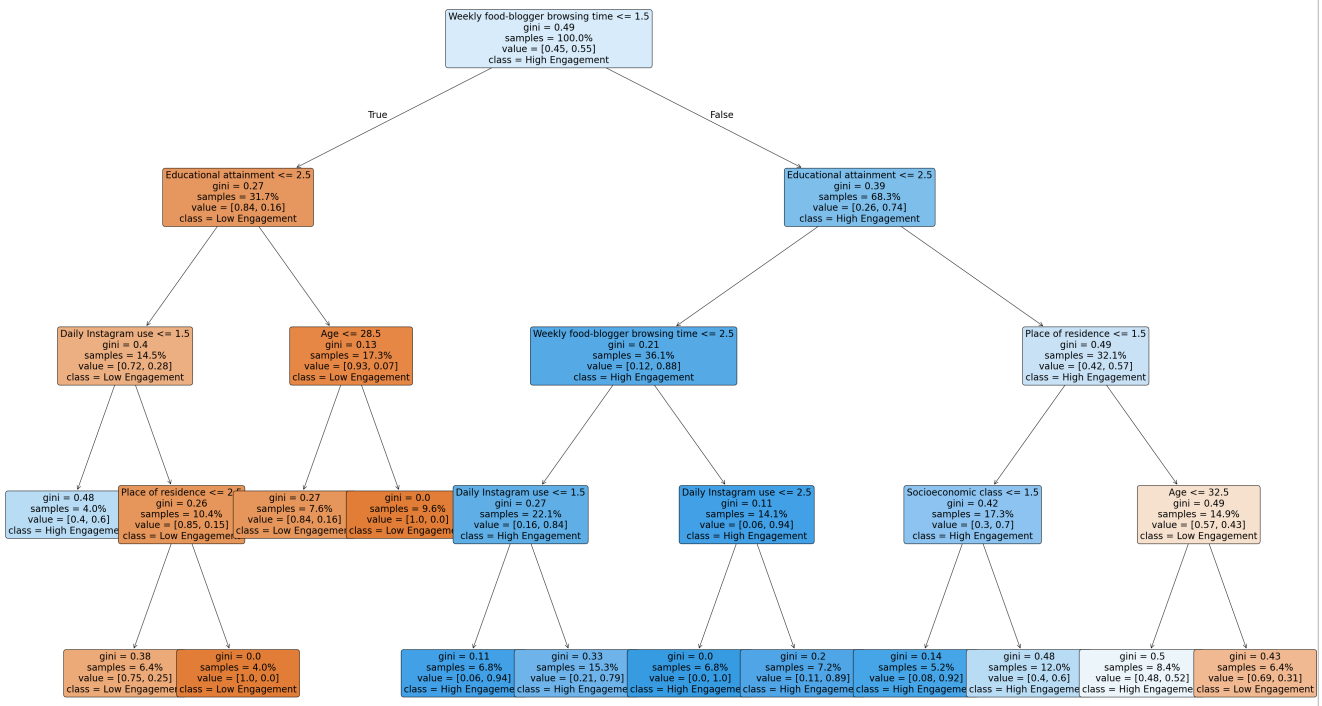
plot_tree(
    tree,
    feature_names=feature_labels,
    class_names=class_labels,
    filled=True,
    rounded=True,
    fontsize=20,
    proportion=True,
    precision=2,
    ax=ax
)
```

```
plt.tight_layout()

fig.savefig(
    "decision_tree_engagement_font20.png",
    dpi=300,
    bbox_inches="tight"
)

fig.savefig(
    "decision_tree_engagement_font20.pdf",
    bbox_inches="tight"
)

plt.show()
```




```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

feature_labels = [
    "Age",
    "Educational attainment",
    "Gender",
    "Socioeconomic class",
    "Place of residence",
    "Daily Instagram use",
    "Weekly food-blogger browsing time"
]
```

```
class_labels = [
    "Low Engagement",
    "High Engagement"
]

fig, ax = plt.subplots(figsize=(40, 24))

plot_tree(
    tree,
    feature_names=feature_labels,
    class_names=class_labels,
    filled=True,
    rounded=True,
    fontsize=25,
    proportion=True,
    precision=2,
    ax=ax
)

plt.tight_layout()

fig.savefig(
    "decision_tree_font35.png",
    dpi=300,
    bbox_inches="tight"
)

fig.savefig(
    "decision_tree_engagement_font35.pdf",
    bbox_inches="tight"
)

plt.show()
```

```
import pandas as pd

importance = pd.DataFrame({
    "Feature": X.columns,
    "Importance": tree.feature_importances_
})

importance = importance.sort_values(
    "Importance",
    ascending=False
)

print(importance)
```

```
     Feature  Importance
6       Page    0.641668
1      Grade    0.170418
5  Instagram    0.065416
4      Place    0.064261
3      Level    0.034105
0        Age    0.024113
2     Gender    0.000000
```

```
plt.figure(figsize=(8,5))

plt.barh(
    importance["Feature"],
    importance["Importance"]
)

plt.gca().invert_yaxis()

plt.xlabel("Importance")

plt.title("Decision Tree Feature Importance")

plt.show()
```

