Clustering with K-Means Model

Case study using “Salary Prediction Classification” data

Include:

- Exploratory Data Analysis (EDA)
- Data Preprocessing
- Principal Component Analysis (PCA)
- Feature selections
- Model evaluation
- Clustering result interpretation

Data Source Overview

Source: https://www.kaggle.com/datasets/ayessa/salary-prediction-classification

```
Data shape:  (32561, 12)
```

	age	workclass	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country
0	39	State-gov	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
2	38	Private	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
3	53	Private	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
4	28	Private	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba

I remove the existing label (> 50k salary vs not) for this clustering exercise.
Maybe there is more than 2 labels for this dataset?

Exploratory Data Analysis (EDA)

a. Knowing the structure of the dataset

```python
# Initialize DataUtils
u = DataUtils()

# Assess the dataset
u.asses_data(salary_df, 'salary')
```

✓ 0.0s

```
Data Assessment for 'salary':
 > Data shape:  (32561, 12)
 > No column should be dropped v
 > All requirements columns are exists v
 > All column types match the requirements v
 > There is no missing value columns v
 > Duplicated data count:  6336
```

b. Handling duplicated and missing values

```python
# Deleting duplicated data
salary_df = salary_df.drop_duplicates()

# Deleting data with missing values
salary_df = salary_df.dropna()

# Re-assess the data
u.asses_data(salary_df, 'salary')
```

✓ 0.1s

```
Data Assessment for 'salary':
 > Data shape:  (28492, 12)
 > No column should be dropped v
 > All requirements columns are exists v
 > All column types match the requirements v
 > There is no missing value columns v
 > There is no duplicated data v
```

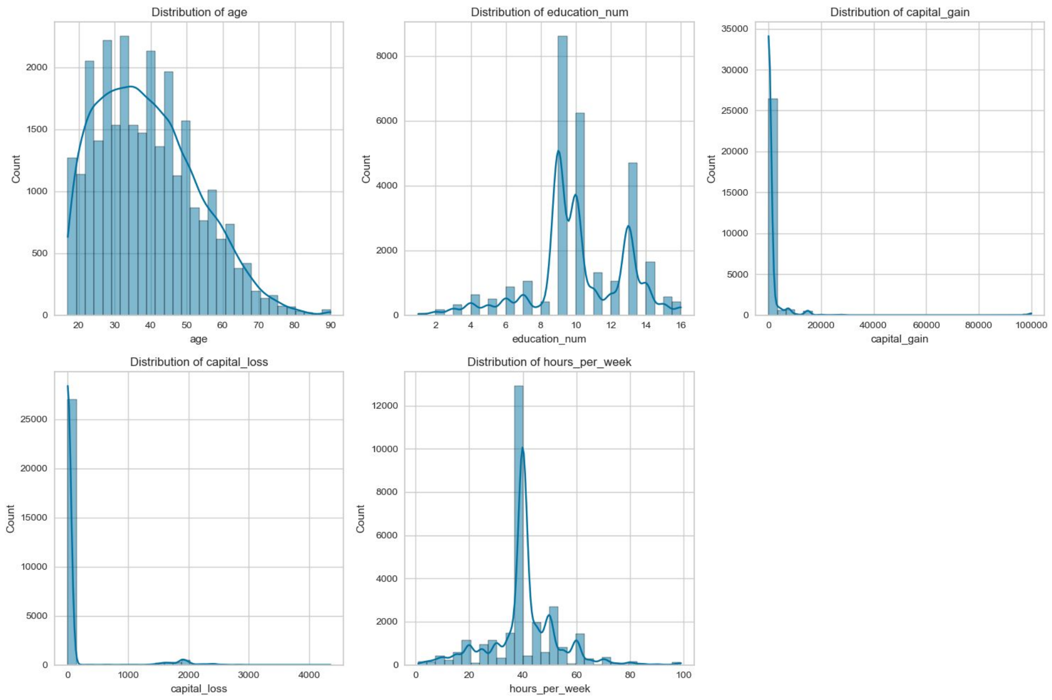
Exploratory Data Analysis (EDA)

c. Distribution and correlation analysis

```python
# Creating histogram plot for numerical data
plt.figure(figsize=(15, 10))
for i, column_ in enumerate(
    salary_df.select_dtypes(include=['int64', 'float64']), 1):
    plt.subplot(2, 3, i)
    sns.histplot(salary_df[column_], kde=True, bins=30)
    plt.title(f'Distribution of {column_}')

plt.tight_layout()
plt.show()
```

✓ 2.8s Python



Exploratory Data Analysis (EDA)

Found unusual pattern in *capital_gain* & *capital_loss* features

	index	capital_gain
0	0	0.905307
1	15024	0.011968
2	7688	0.009827
3	7298	0.008564
4	99999	0.005545

	index	capital_loss
0	0	0.946968
1	1902	0.007090
2	1977	0.005826
3	1887	0.005510
4	1485	0.001790

More than 90% of the data is "zero values"

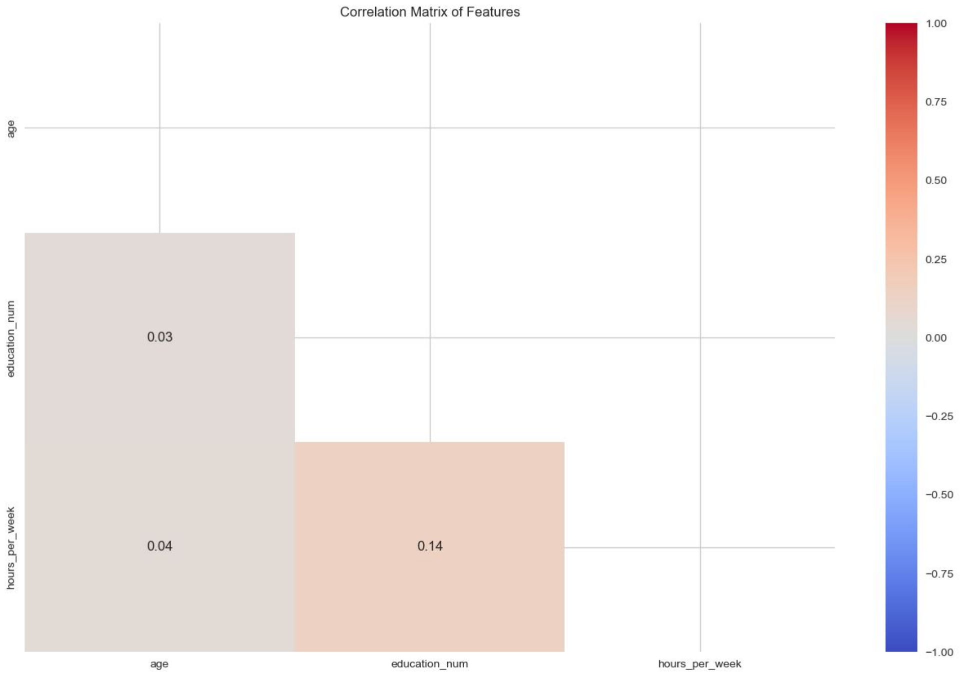
So, we drop these features

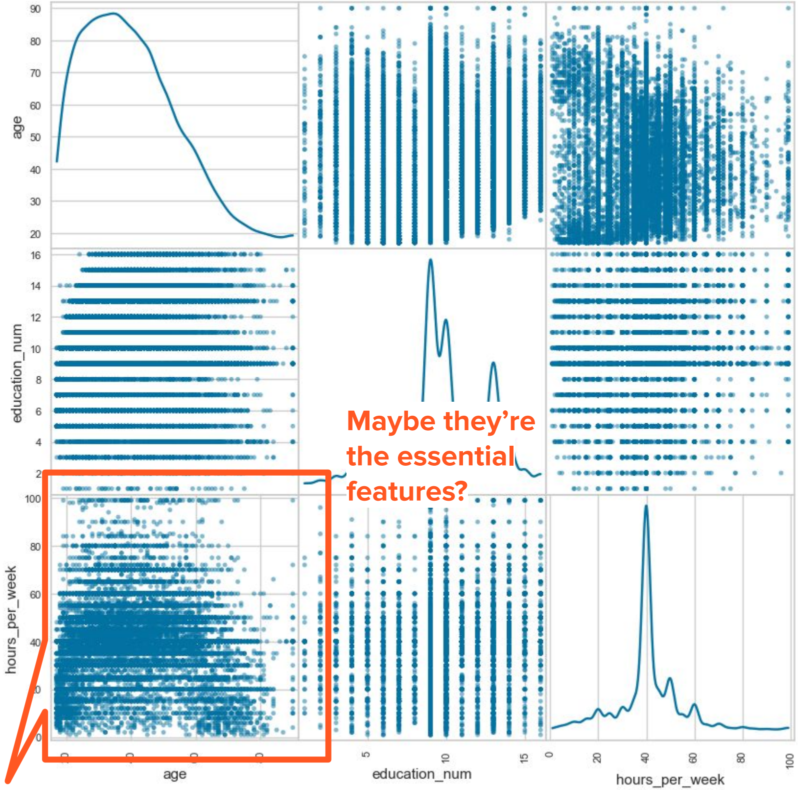
As the feature contains a lot of 'zero' values (> 90%), these features are dropped to avoid hindering the performance of the model.

```
salary_df = salary_df.drop(columns=['capital_gain', 'capital_loss'])
```

✓ 0.0s Python

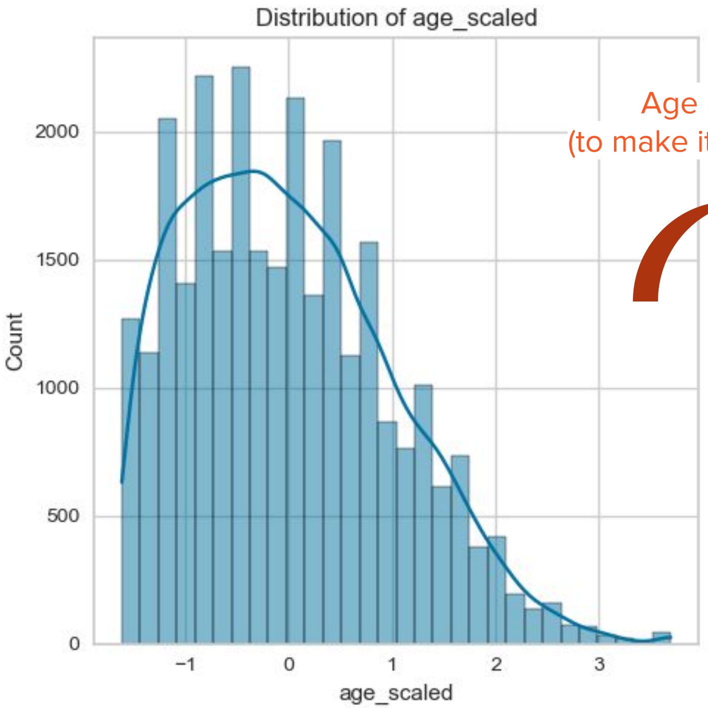
Correlation Matrix & Pair Plot



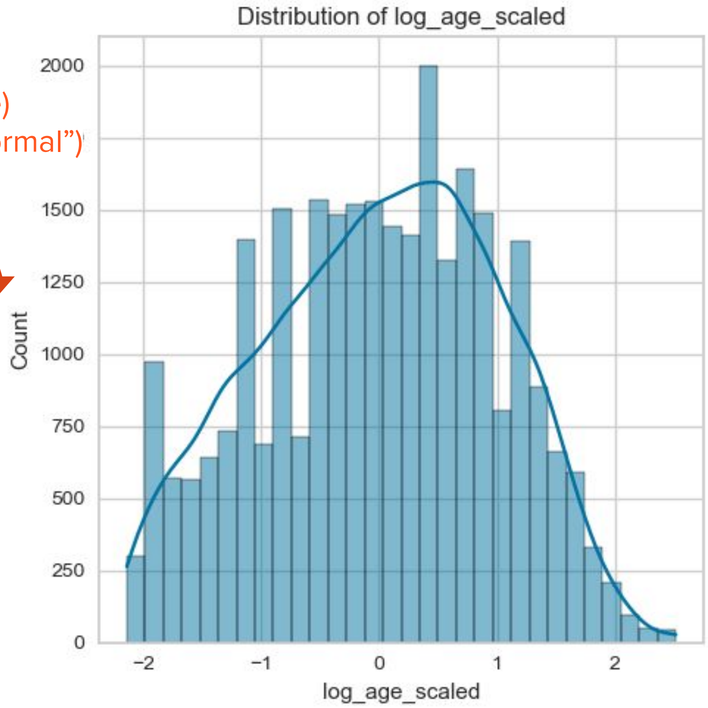


No apparent positive/negative relationship.
But, we can look at some fuzzy clusters in the *age vs hours_per_week plot.*

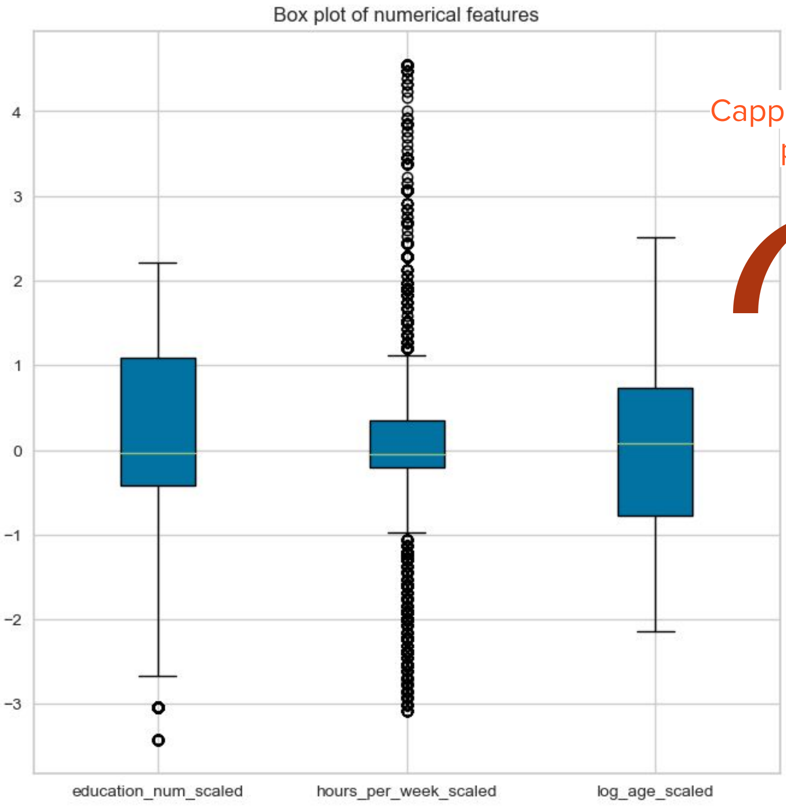
Data Preprocessing: Data Transformation



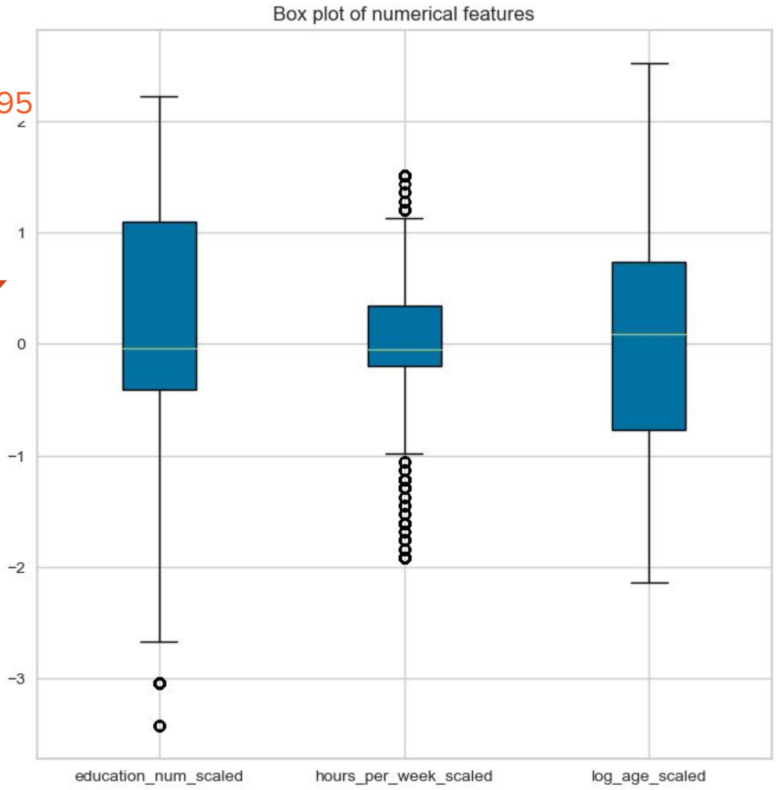
Age → Log(Age)
(to make it more “normal”)



Data Preprocessing: Outlier Handling



Capped at 5 and 95 percentile



Encode Categorical Data & Bin Numerical Data

f. Encoding Data

Encoding data is the process of converting categorical features into numerical ones. In this case, the one-hot encoding method is used.

```python
# List of categorical columns
categorical_columns = salary_df.select_dtypes(include='object').columns
```

✓ 0.0s Python

```python
# Viewing the summary of categorical data
for col in categorical_columns:
    print('-' * 20 + '\n' + col + '\n' + '-' * 20)
    print(salary_df[col].value_counts().sort_values(ascending=False) / salary_df
```

✓ 0.0s Python

```
--------------------
workclass
--------------------
 Private             0.670504
 Self-emp-not-inc    0.086059
 Local-gov           0.070616
 ?                   0.056788
```

```python
# Encode the categorical columns
salary_df = pd.get_dummies(salary_df, columns=categorical_columns)
```

✓ 0.0s Python

g. Binning Data

Binning data is used to group continuous data into intervals or bins. The benefits are that it can reduce noise, make visualization and interpretation easier, and improve efficiency.

```python
# Binning the age & hours per week features into 5 levels
def binner(df: pd.DataFrame, column: str, bins: list):
    df[f'{column}_level'] = pd.cut(df[column], bins=bins, labels=[f" {i}" for i
    return df

bin_array = {
    'age': [0, 25, 35, 45, 55, 100],
    'hours_per_week': [0, 20, 40, 60, 80, 100]
}

for col in ['age', 'hours_per_week']:
    salary_df = binner(salary_df, col, bin_array[col])
```

✓ 0.0s Python

Performing PCA to Reduce Dimensions

```python
# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(analysis_df)
```

✓ 0.4s Python

```python
# Creating a separate dataframe to store PCA result
pca = PCA(n_components=2)  # Choose the number of components
pca_result = pca.fit_transform(scaled_data)
pca_df = pd.DataFrame(data=pca_result, columns=['PC1', 'PC2'])
```

✓ 0.2s Python

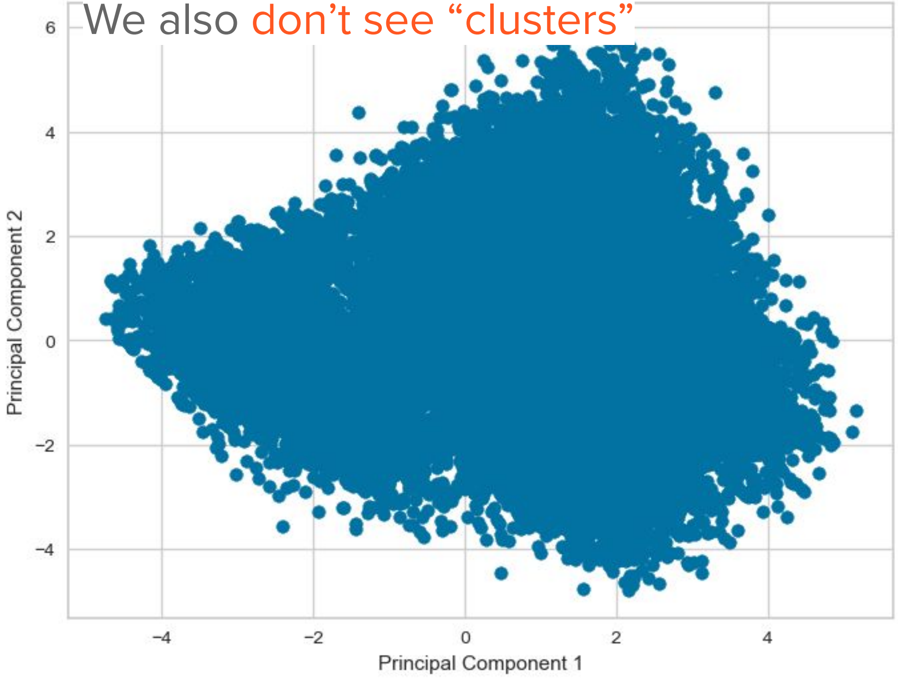
```python
# Overview of PCA result
print('PCA Component 1 Ratio of Explained Variance')
print(f"{pca.explained_variance_ratio_[0]:.2%}")
print('PCA Component 2 Ratio of Explained Variance')
print(f"{pca.explained_variance_ratio_[1]:.2%}")
print('Total Explained Variance')
print(f"{pca.explained_variance_ratio_.sum():.2%}")
```

✓ 0.0s Python

```
PCA Component 1 Ratio of Explained Variance
5.19%
PCA Component 2 Ratio of Explained Variance
3.22%
Total Explained Variance
8.41%
```

This result is bad because total explained variance is only 8%

We also don't see "clusters"



Features Selection

- Drop Features with High Multicollinearity

Dropping features with high multicollinearity to avoid redundancy

```python
# Define the correlation limit
limit = 0.8
high_correlated_columns = []

# List the features with high multicollinearity
for col in correlation_matrix:
    list_ = []
    temp_list = [[col, key, val] for key, val in correlation_matrix.loc[col].to_
    temp_list = [x for x in temp_list if x[0] != x[1]]
    if len(temp_list) > 0:
        high_correlated_columns.extend(temp_list)

print(f"Berikut kolom yang saling mempunyai korelasi tinggi (>|{limit}|)")
for x in high_correlated_columns:
    print(x)
```

✓ 0.0s Python

```
Berikut kolom yang saling mempunyai korelasi tinggi (>|0.8|)
['workclass_ ?', 'occupation_ ?', 0.9977138642563048]
['occupation_ ?', 'workclass_ ?', 0.9977138642563048]
['sex_ Female', 'sex_ Male', -0.9999999999999999]
['sex_ Male', 'sex_ Female', -0.9999999999999999]
```

- Drop Features with (Near-)Zero Variance

Features with very low or near-zero variance are features whose values hardly change across the entire dataset. In other words, these features do not have much variation and do not provide much useful information for the model.

```python
# Setting the variance limit
limit = 0.23
features = analysis_df.var()[analysis_df.var() / analysis_df.max() > limit].inde
print(f"Fitur yang dipertahankan\n{features}")
print(f"Fitur yang didrop\n{analysis_df.var()[analysis_df.var() / analysis_df.ma
```

✓ 0.0s Python

```
Fitur yang dipertahankan
Index(['education_num_scaled', 'hours_per_week_scaled', 'log_age_scaled',
       'marital_status_ Married-civ-spouse', 'relationship_ Husband'],
      dtype='object')
Fitur yang didrop
Index(['workclass_ Federal-gov', 'workclass_ Local-gov',
       'workclass_ Never-worked', 'workclass_ Private',
       'workclass_ Self-emp-inc', 'workclass_ Self-emp-not-inc',
       'workclass_ State-gov', 'workclass_ Without-pay',
       'marital_status_ Divorced', 'marital_status_ Married-AF-spouse',
       'marital_status_ Married-spouse-absent',
       'marital_status_ Never-married', 'marital_status_ Separated',
       'marital_status_ Widowed', 'occupation_ ?', 'occupation_ Adm-clerical',
       'occupation_ Armed-Forces', 'occupation_ Craft-repair',
       'occupation_ Exec-managerial', 'occupation_ Farming-fishing',
       'occupation_ Handlers-cleaners', 'occupation_ Machine-op-inspct',
       'occupation_ Other-service', 'occupation_ Priv-house-serv',
       'occupation_ Prof-specialty', 'occupation_ Protective-serv',
       'occupation_ Sales', 'occupation_ Tech-support',
       'occupation_ Transport-moving', 'relationship_ Not-in-family',
```

Performing PCA Again

```python
# Executing PCA operation again
scaler = StandardScaler()
scaled_data = scaler.fit_transform(analysis_df)

pca = PCA(n_components=2)  # Choose the number of components
pca_result = pca.fit_transform(scaled_data)

pca_df = pd.DataFrame(data=pca_result, columns=['PC1', 'PC2'])
```

✓ 0.0s Python

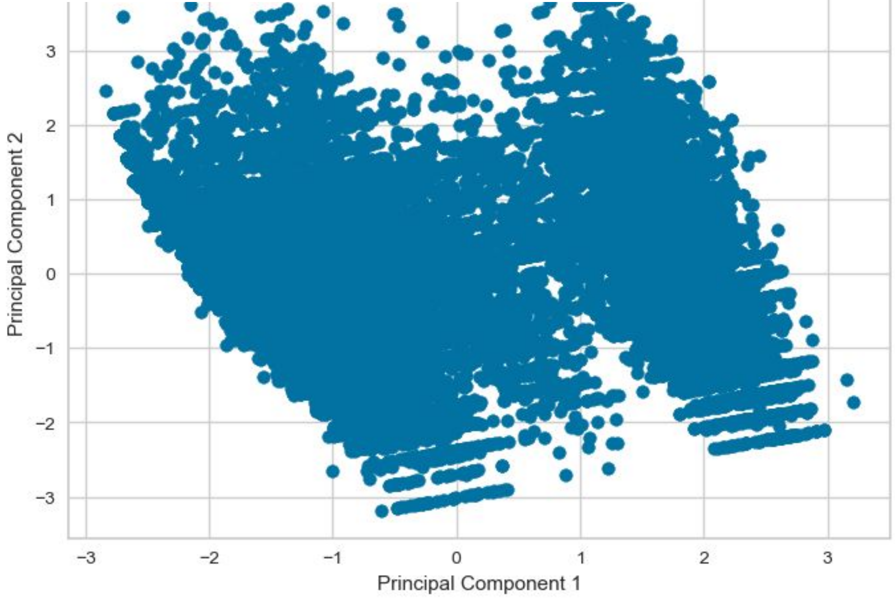
```python
# Overview of PCA result
print('PCA Component 1 Ratio of Explained Variance')
print(f"{pca.explained_variance_ratio_[0]:.2%}")
print('PCA Component 2 Ratio of Explained Variance')
print(f"{pca.explained_variance_ratio_[1]:.2%}")
print('Total Explained Variance')
print(f"{pca.explained_variance_ratio_.sum():.2%}")
```

✓ 0.0s Python

```
PCA Component 1 Ratio of Explained Variance
44.30%
PCA Component 2 Ratio of Explained Variance
21.01%
Total Explained Variance
65.31%
```

This result is quite good because total explained variance is 65%

We also see 2 (or more) "clusters"



Building the Model with Default Parameter

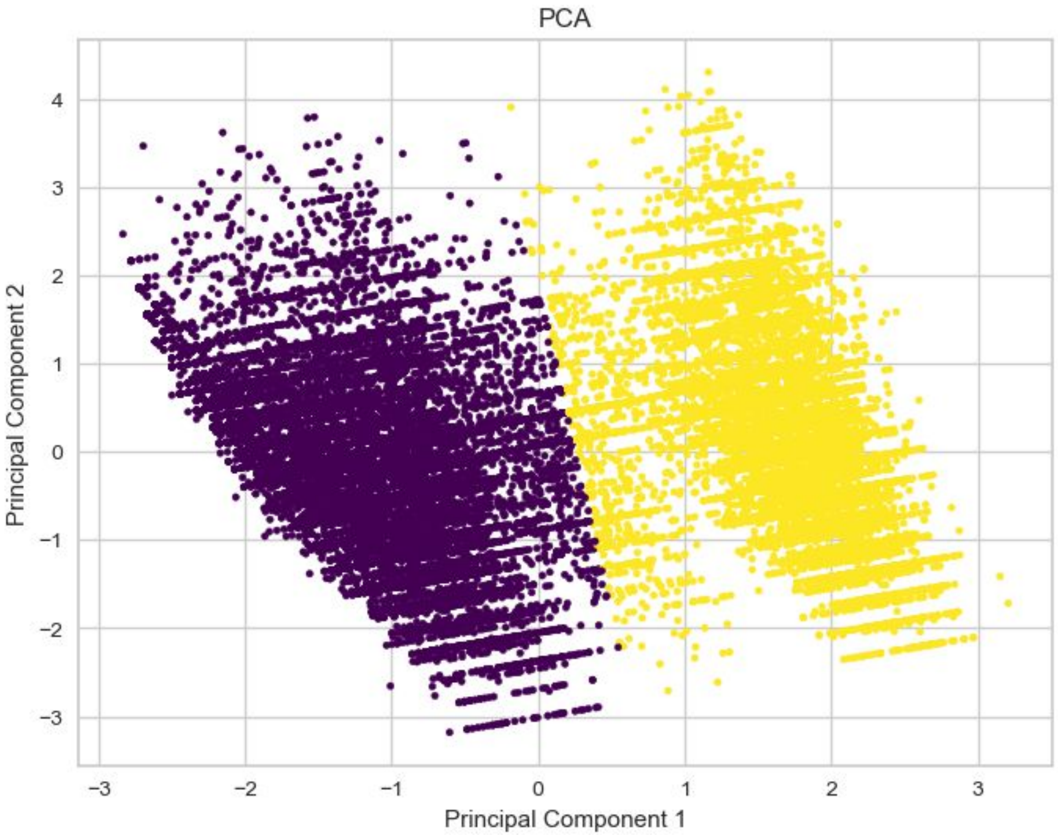
```
# Select features
X = pca_df
```

✓ 0.0s Python

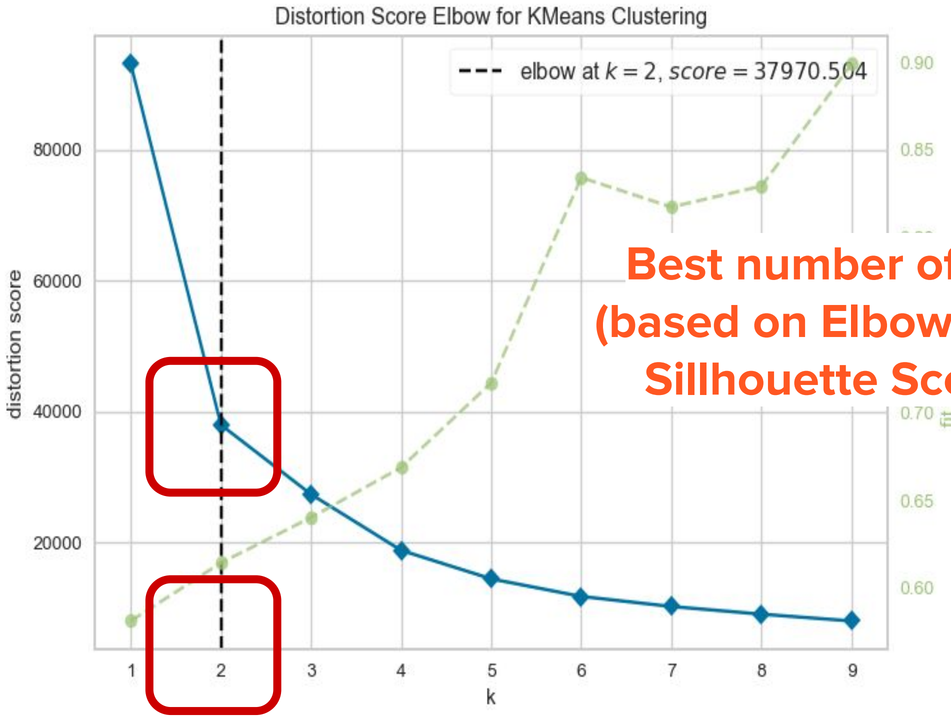
```
# Initialize KMeans model with default parameter
kmeans = KMeans()
```

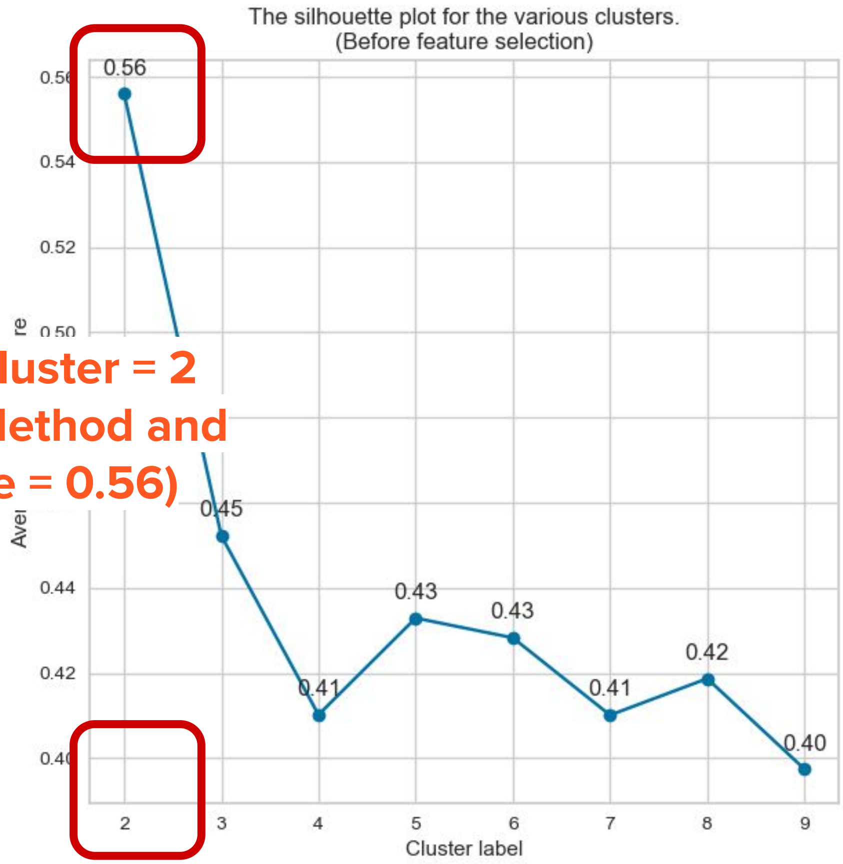
✓ 0.0s Python

Clustering Result



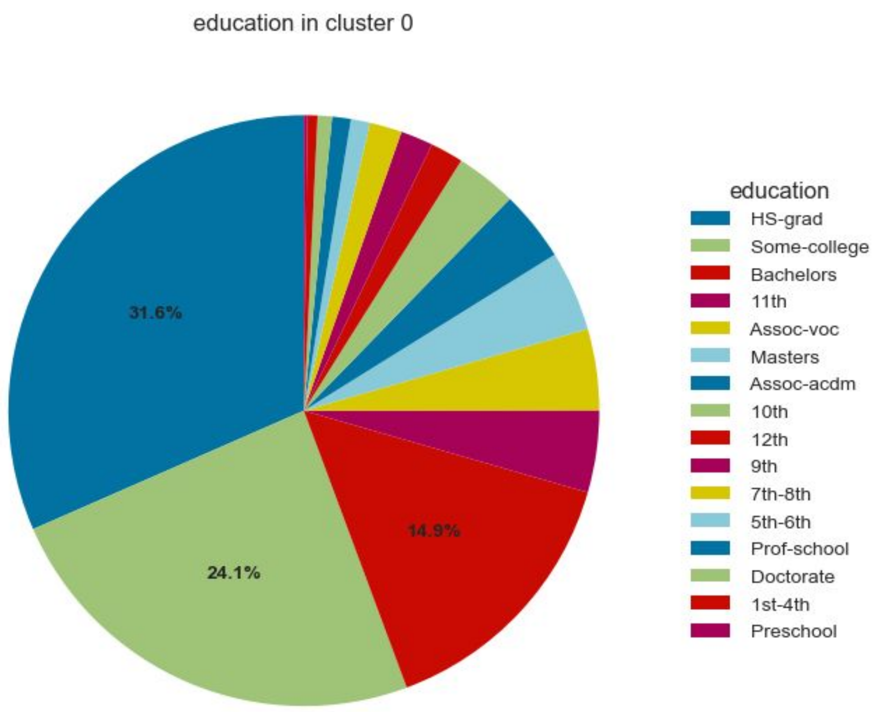
Model Evaluation

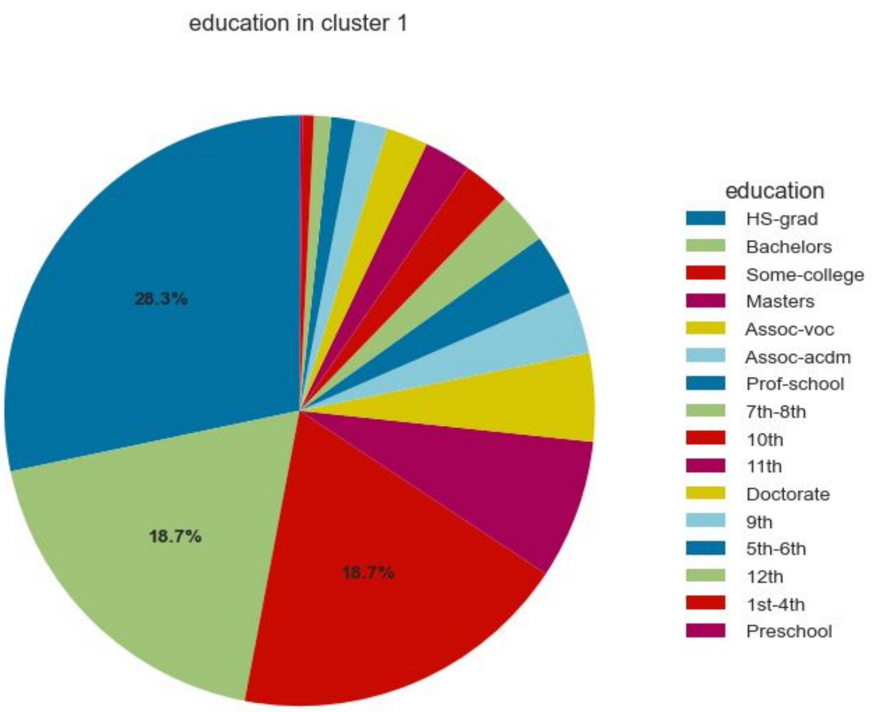




Best number of cluster = 2
(based on Elbow Method and
Sillhouette Score = 0.56)

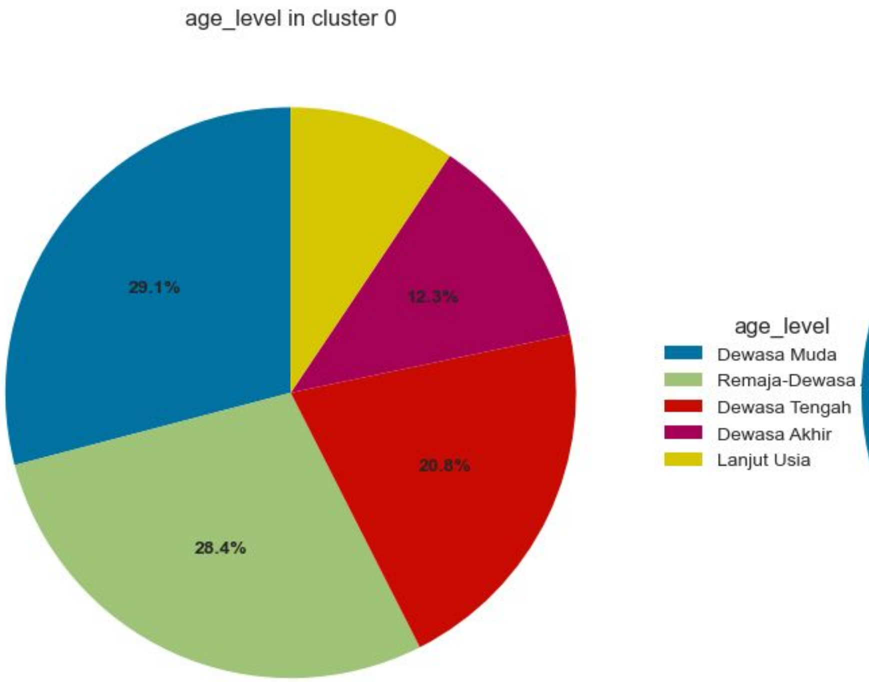
Interpretation:

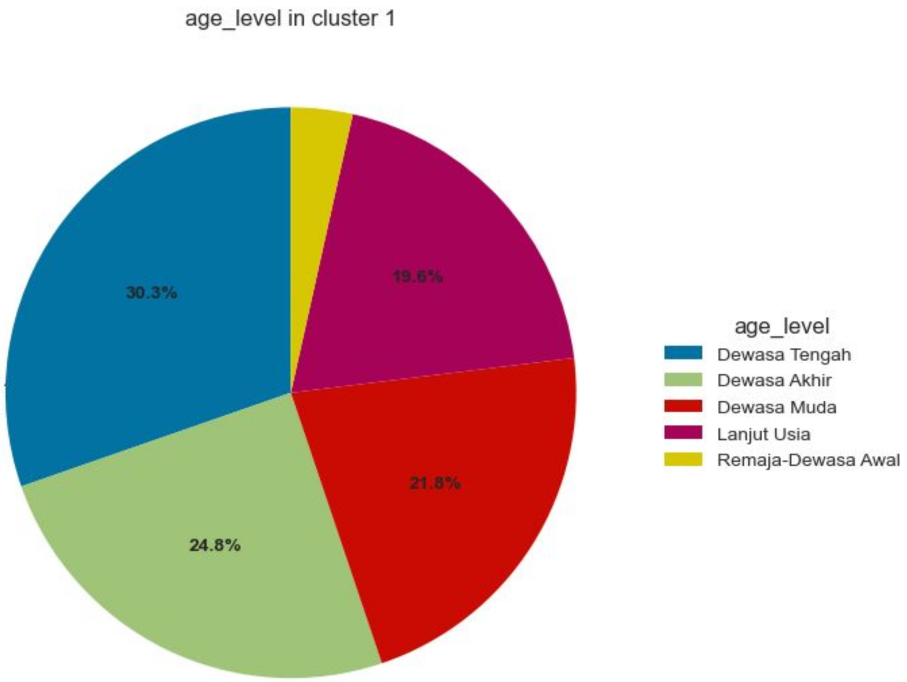




Some college > Bachelors vs Bachelors > Some-college

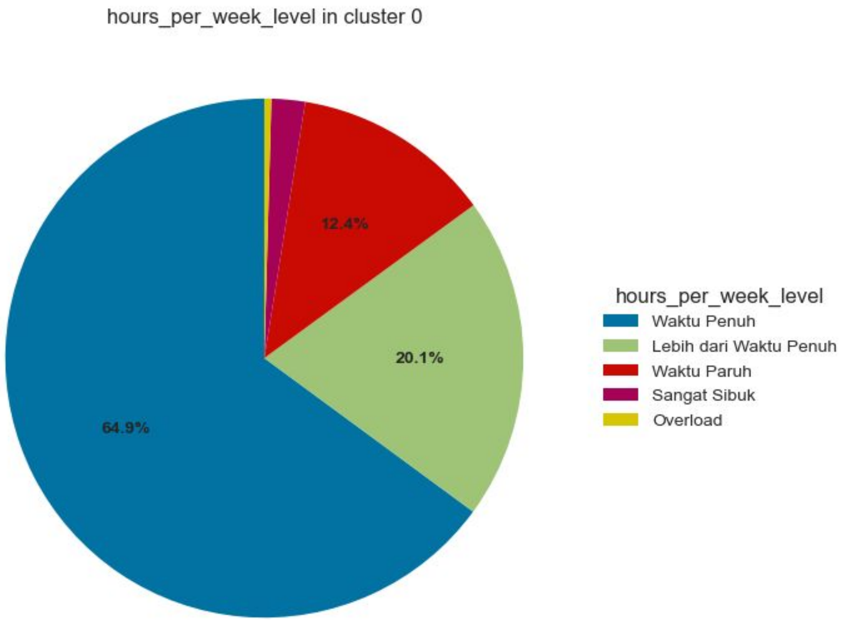
Interpretation:

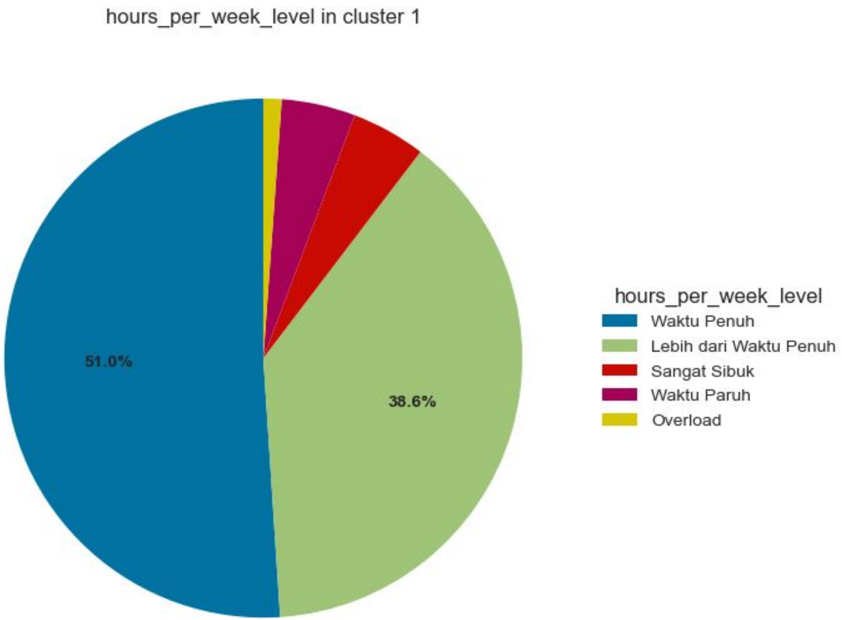




Having more “youngsters”vs Having more “elders”

Interpretation:





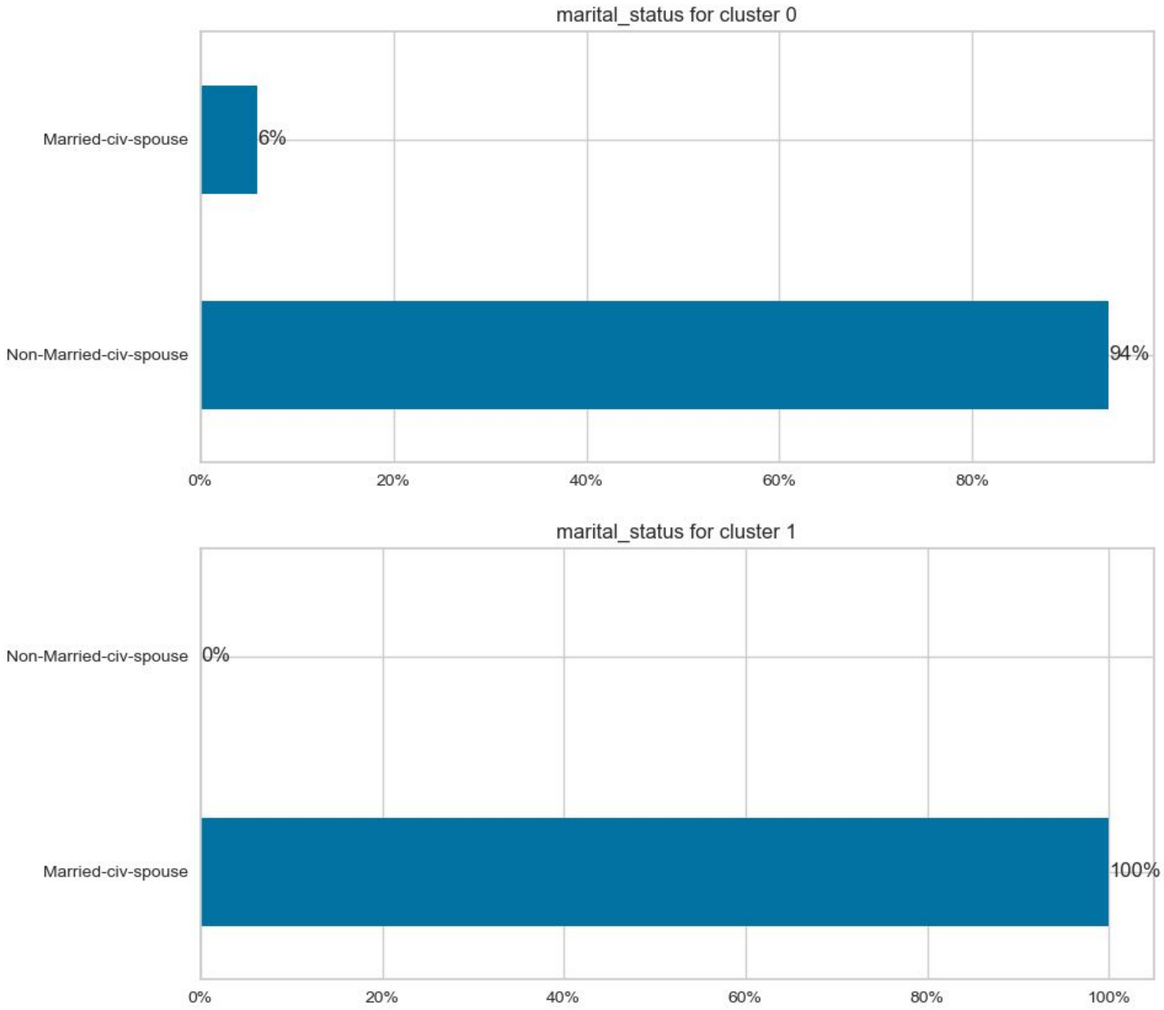
"Busy" vs "Busier"

Interpretation:

"Not-Married with civil spouse"

vs

"Married with civil spouse"

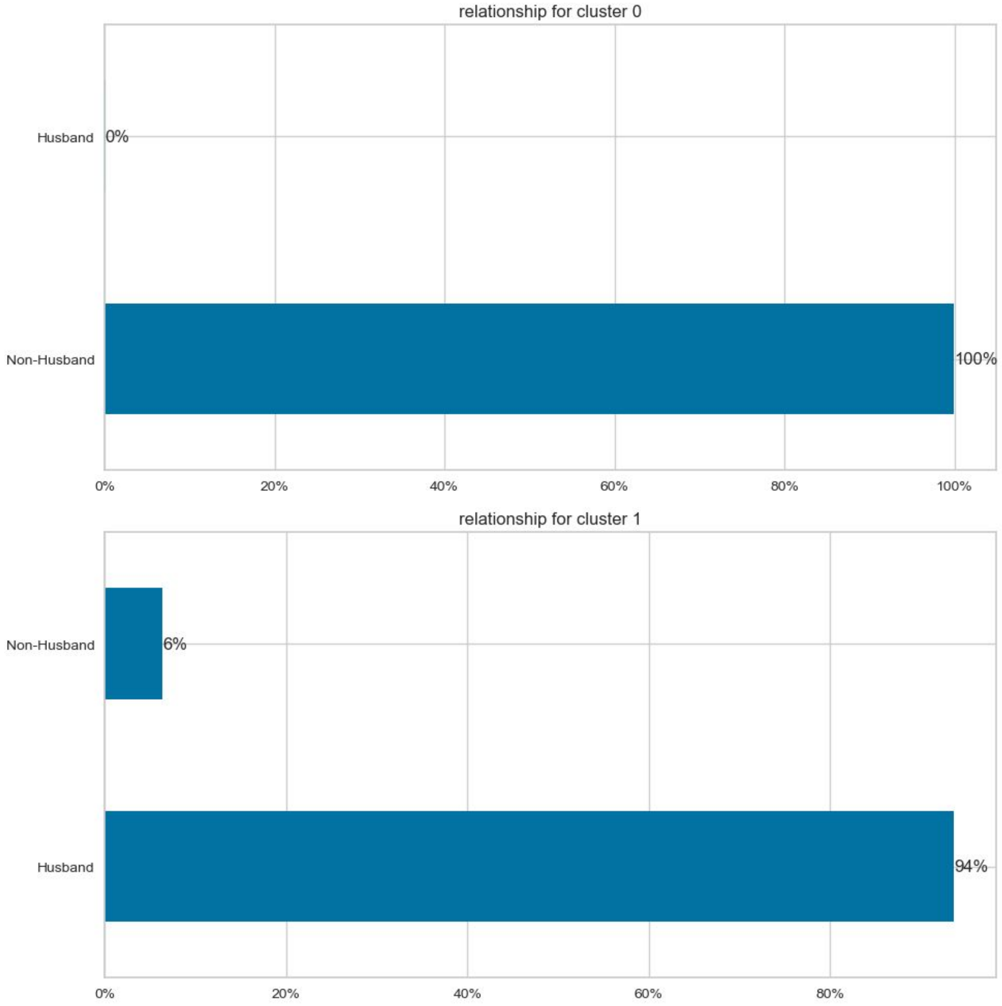


Interpretation:

“Single, wive, other non-husbands”

vs

“Husband”



Conclusion

Cluster 0

This cluster is dominated by wives, singles, and others. They tend to be younger and less busy compared to other clusters. The majority are high school graduates or attended college, especially those who did not earn a degree.

- About 2/3 are high school-college graduates (but more are non-degree holders)
- Higher proportion of young demographics (57.5%)
- Fewer "busy" individuals (20.1%)
- Almost all (94%) are not married to a civil partner
- Almost all are not husbands (99%)

Cluster 1

This cluster is dominated by husbands married to civil partners. They tend to be older and busier compared to other clusters. The majority are high school or college graduates, whether degree holders or not.

- About 2/3 are high school-college graduates
  - Lower proportion of young demographics (25.3%)
  - More "busy" individuals (38.6%)
  - Almost all (100%) are married to a civil partner
  - Almost all are husbands (100%)

be older and busier compared to other clusters. The majority are high school or college graduates, whether degree holders or not.