Data Preparation and Data Mining

SALARY PREDICTION CLASSIFICATION

@About Dataset

https://www.kaggle.com/

TO DO LIST

01 DATA PREPARATION

02 DECISION TREE

03 NAIVE BAYES

04 K-NEAREST NEIGHBOR

05 ASSOCIATION RULES

06 K-MEAN CLUSTERING

DATA PREPARATION

1. SELECT COLUMNS

age, workclass, education, marital-status, occupation, relationship, race, sex, hours-per-week , native-country, salary

2. CHECK MISSING

จากการ check missing พบว่า ไม่มี missing แต่มีข้อมูลที่ไม่รู้ค่า ซึ่งแสดงในสัญลักษณ์ "?" มีในคอลัมน์ workclass, occupation และ native-country ทั้งหมด 2399 Row จึงทำการ drop ออก

```
data[(data['workclass'] == ' ?') | (data['occupation'] == ' ?') | (data['native-country'] == ' ?')].shape
```

```
(2399, 11)
```

3. เปลี่ยนค่าในคอลัมน์ SALARY

โดย <= 50K เป็น 0 และ > 50K เป็น 1

```
[45]  data['salary'] = data['salary'].map({' <=50K': 0, ' >50K': 1})
```

```
[46]  data.head(10)
```

	age	workclass	education	marital-status	occupation	relationship	race	sex	hours-per-week	native-country	salary
0	39	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	40	United-States	0
1	50	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	13	United-States	0
2	38	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	40	United-States	0
3	53	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	40	United-States	0
4	28	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	40	Cuba	0
5	37	Private	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	40	United-States	0
6	49	Private	9th	Married-spouse-absent	Other-service	Not-in-family	Black	Female	16	Jamaica	0
7	52	Self-emp-not-inc	HS-grad	Married-civ-spouse	Exec-managerial	Husband	White	Male	45	United-States	1
8	31	Private	Masters	Never-married	Prof-specialty	Not-in-family	White	Female	50	United-States	1
9	42	Private	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	40	United-States	1

4. CONVERT CATEGORICAL VARIABLES TO NUMERICAL

แปลงข้อมูลให้อยู่ในรูปแบบของ category

```
[13]  data['workclass'] = data['workclass'].astype('category').cat.codes
      data['education'] = data['education'].astype('category').cat.codes
      data['marital-status'] = data['marital-status'].astype('category').cat.codes
      data['occupation'] = data['occupation'].astype('category').cat.codes
      data['relationship'] = data['relationship'].astype('category').cat.codes
      data['race'] = data['race'].astype('category').cat.codes
      data['sex'] = data['sex'].astype('category').cat.codes
      data['native-country'] = data['native-country'].astype('category').cat.codes
```

WORKCLASS

0	Federal-gov
1	Local-gov
2	Private
3	Self-emp-inc
4	Self-emp-not-inc
5	State-gov
6	Without-pay

MARITAL-STATUS

0	Divorced
1	Married-AF-spouse
2	Married-civ-spouse
3	Married-spouse-absent
4	Never-married
5	Separabed
6	Widowed

EDUCATION

0	10th
1	11th
2	12th
3	15t-4th
4	5th-6th
5	7th-8th
6	9th
7	Assoc-acdm
8	Assoc-acdm
9	Bachelors
10	Doctorate
11	HS-grad
12	Masters
13	Preschool
14	Prof-school
15	Some-college

OCCUPATION

0	Adm-clerical
1	Armed-Forces
2	Craft-repair
3	Exec-managerial
4	Farming-fishing
5	Handles-cleaners
6	Machine-op-inspct
7	Other-service
8	Priv-house-serv
9	Prof-specially
10	Protective-serv
11	Sales
12	Tech-support
13	Transport-moving

NATIVE-COUNTRY

0	Cambodia
1	Puerto-Rico
2	China
3	Columbia
4	Cuba
5	Dominican-Republic
6	Ecuador
7	India
8	England
9	France
10	Germany
11	Greece
12	Guatemala
13	Haiti
14	Holand-Netherlands
15	Honduras
16	Hong
17	Hungary
18	El-Salvador
19	Iran
20	Ireland
21	Italy
22	Jamaica
23	Japan
24	Laos
25	Mexico
26	Nicaragua
27	Outlying-Us(Guam-USVI-etc)
28	Peru
29	Philippines
30	Poland
31	Portugal

NATIVE-COUNTRY

32	Canada
33	Scotland
34	South
35	Taiwan
36	Thailand
37	Trinadad&Tobago
38	United-States
39	Vietnam
40	Yugoslavia

SEX

0	Male
1	Female

RELATIONSHIP

0	Husband
1	Not-in-family
2	Other-relative
3	Own-chid
4	Unmarried
5	Wife

RACE

0	Amer-Indian-Eskimo
1	Asian-Pac-Islander
2	Black
3	Other
4	White

5. SPLIT THE DATA

X คือ FEATURE Y คือ TARGET หรือ ผลลัพธ์ที่ต้องการทำนาย

```
[21] features = ['age', 'workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex',  'hours-per-week', 'native-country']
     X = data[features]
     y = data['salary']
```

แบ่งเป็น TRAIN 70% TEST 30%

```
[22] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

DECISION TREE

GRIDSEARCHCV (10-FOLD-CROSS-VALIDATION)

```
[64]  # Define the Decision Tree model
      dt = DecisionTreeClassifier(random_state=42)

      # Define the hyperparameter values to be tested
      param_grid = {'max_depth': [1,2,3],
                    'min_samples_split': [2, 4, 6]}

      # Use Grid Search to find the best hyperparameter combination
      grid_search = GridSearchCV(dt, param_grid, cv=10, scoring='accuracy')
      grid_search.fit(X_train, y_train)

      # Get the best hyperparameter combination and the associated accuracy score
      best_params = grid_search.best_params_
      best_score = grid_search.best_score_

      # Print the results
      print("Best hyperparameters:", best_params)
      print(f'Best accuracy score: {best_score:.2f}')
```

```
Best hyperparameters: {'max_depth': 3, 'min_samples_split': 2}
Best accuracy score: 0.79
```

Define , Train and Predict - evaluate

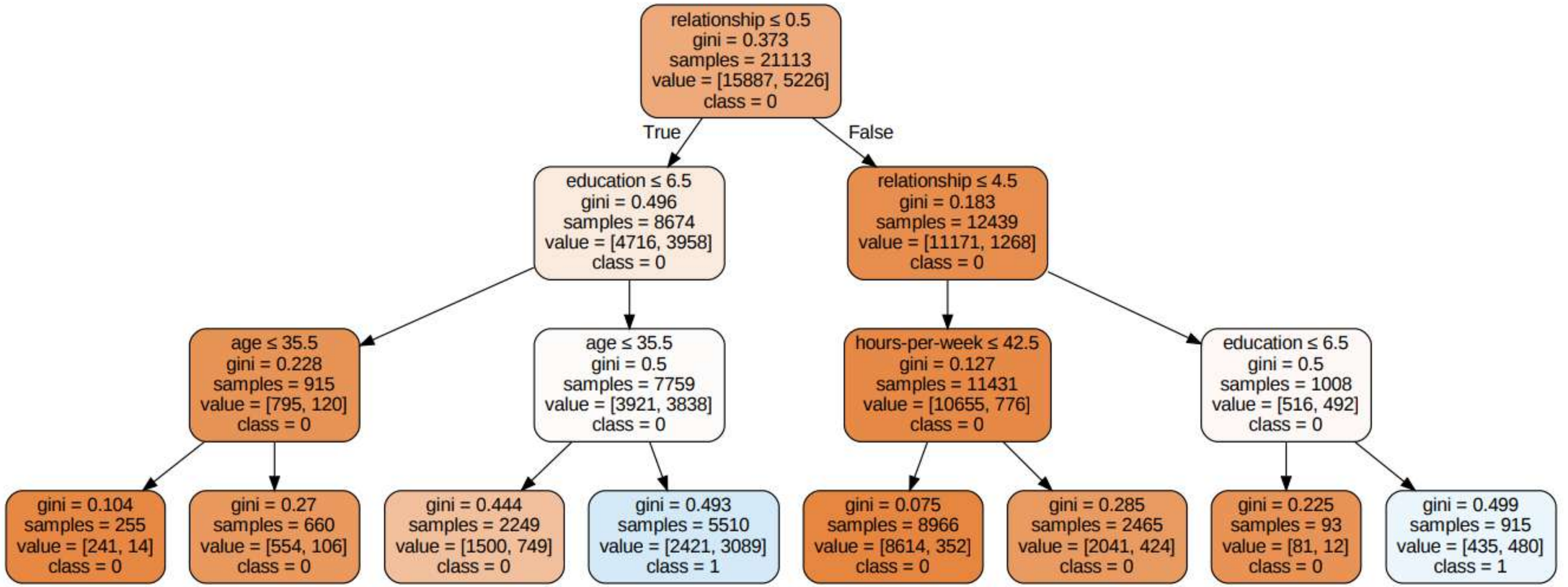
```
[20]  #Define
      clf = DecisionTreeClassifier(max_depth=3,min_samples_leaf=2)

      # Train
      clf.fit(X_train, y_train)

      # Evaluate
      y_pred = clf.predict(X_test)
```

ACCURACY

```
[44]  accuracy = accuracy_score(y_test, y_pred)
      report = classification_report(y_test, y_pred)

      print(f'Accuracy: {accuracy:.2f}')
      print('Classification report:\n', report)
```

```
Accuracy: 0.78
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.82      0.85      6767
           1       0.55      0.67      0.60      2282

    accuracy                           0.78      9049
   macro avg       0.72      0.74      0.73      9049
weighted avg       0.80      0.78      0.79      9049
```

DIAGRAM OF DECISION TREE



https://www.kaggle.com

NAÏVE BAYES

GridSearchCV (10-fold-cross-validation)

```
[43]  # กำหนดค่า var_smoothing ที่จะทดสอบ
      param_grid = {'var_smoothing': np.logspace(0,-9, num=100)}

      # สร้างตัวแบบ GaussianNB
      gnb = GaussianNB()

      # ใช้ GridSearchCV ในการหาค่า var_smoothing ที่ดีที่สุด
      grid_search = GridSearchCV(gnb, param_grid=param_grid, cv=10, scoring='accuracy')
      grid_search.fit(X_train, y_train)

      # แสดงค่า var_smoothing ที่ดีที่สุดและค่าความแม่นยำที่ได้
      print("Best var_smoothing:", grid_search.best_params_)
      print(f'Accuracy: {grid_search.best_score_:.2f}')
```

```
Best var_smoothing: {'var_smoothing': 0.01}
Accuracy: 0.78
```

DEFINE , TRAIN AND PREDICT - EVALUATE

```
[44]  #Define
      gnb = GaussianNB(var_smoothing = 0.01)

      #train
      gnb.fit(X_train, y_train)

      #test
      y_pred = gnb.predict(X_test)
```

ACCURACY AND CLASSIFICATION REPORT

```
[45]  accuracy = accuracy_score(y_test, y_pred)
      report = classification_report(y_test, y_pred)

      print(f'Accuracy: {accuracy:.2f}')
      print('Classification report:\n', report)
```

```
Accuracy: 0.78
Classification report:
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      6767
           1       0.56      0.55      0.56      2282

    accuracy                           0.78      9049
   macro avg       0.71      0.70      0.70      9049
weighted avg       0.78      0.78      0.78      9049
```

```
[46]  scores = cross_val_score(gnb, X_train, y_train, cv=10)
      print(f'Mean cross-validation score: {scores.mean():.2f}')
```

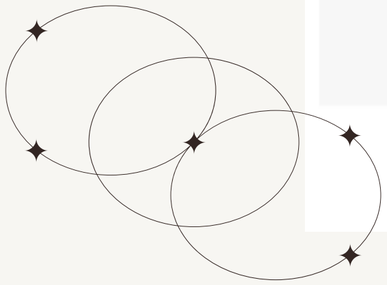
```
Mean cross-validation score: 0.78
```

K-NEAREST NEIGHBOR

• GridSearchCV (10-fold-cross-validation)

```
[27] # กำหนดค่า k ที่ต้องการทดสอบ
     param_grid = {'n_neighbors': [1, 3, 5, 7, 9, 11, 13, 17, 19]}

     # สร้างโมเดล KNN
     knn = KNeighborsClassifier()

     # ใช้ GridSearchCV ในการหาค่า k ที่ดีที่สุด
     grid_search = GridSearchCV(knn, param_grid, cv=10, scoring='accuracy')
     grid_search.fit(X_train, y_train)

     # แสดงค่า k ที่ดีที่สุด
     print("Best k: ", grid_search.best_params_['n_neighbors'])
```

```
Best k:  13
```



- Normalize Data with StandardScaler

```
[28]  scaler = StandardScaler()

      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

0s

- Defind, Train and Predict-Evaluate

```
[30]  #Define
      knn = KNeighborsClassifier(n_neighbors=13, metric='euclidean')

      #Train
      knn.fit(X_train, y_train)

      #evalution
      y_pred = knn.predict(X_test)
```

Accuracy and Classification report

```
[31] accuracy = accuracy_score(y_test, y_pred)
     report = classification_report(y_test, y_pred)

     print(f'Accuracy: {accuracy:.2f}')
     print('Classification report:\n', report)
```

```
Accuracy: 0.81
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      6767
           1       0.65      0.57      0.61      2282

    accuracy                           0.81      9049
   macro avg       0.76      0.73      0.74      9049
weighted avg       0.81      0.81      0.81      9049
```

```
[32] scores = cross_val_score(knn, X_train, y_train, cv=10)
     print(f'Mean cross-validation score: {scores.mean():.2f}')
```

```
Mean cross-validation score: 0.81
```

สรุปผล

	Decision tree	Naive Bayes	K-Nearest Neighbor
Mean cross-validation	79%	78%	81%
Accuracy	78%	78%	81%

ตัดสินใจเลือก K-Nearest Neighbor เนื่องจากมีค่า Mean cross-validation สูงที่สุด ซึ่งบ่งบอกว่า โมเดลนี้มีประสิทธิภาพมากที่สุด

ASSOCIATION RULES

https://www.kaggle.com

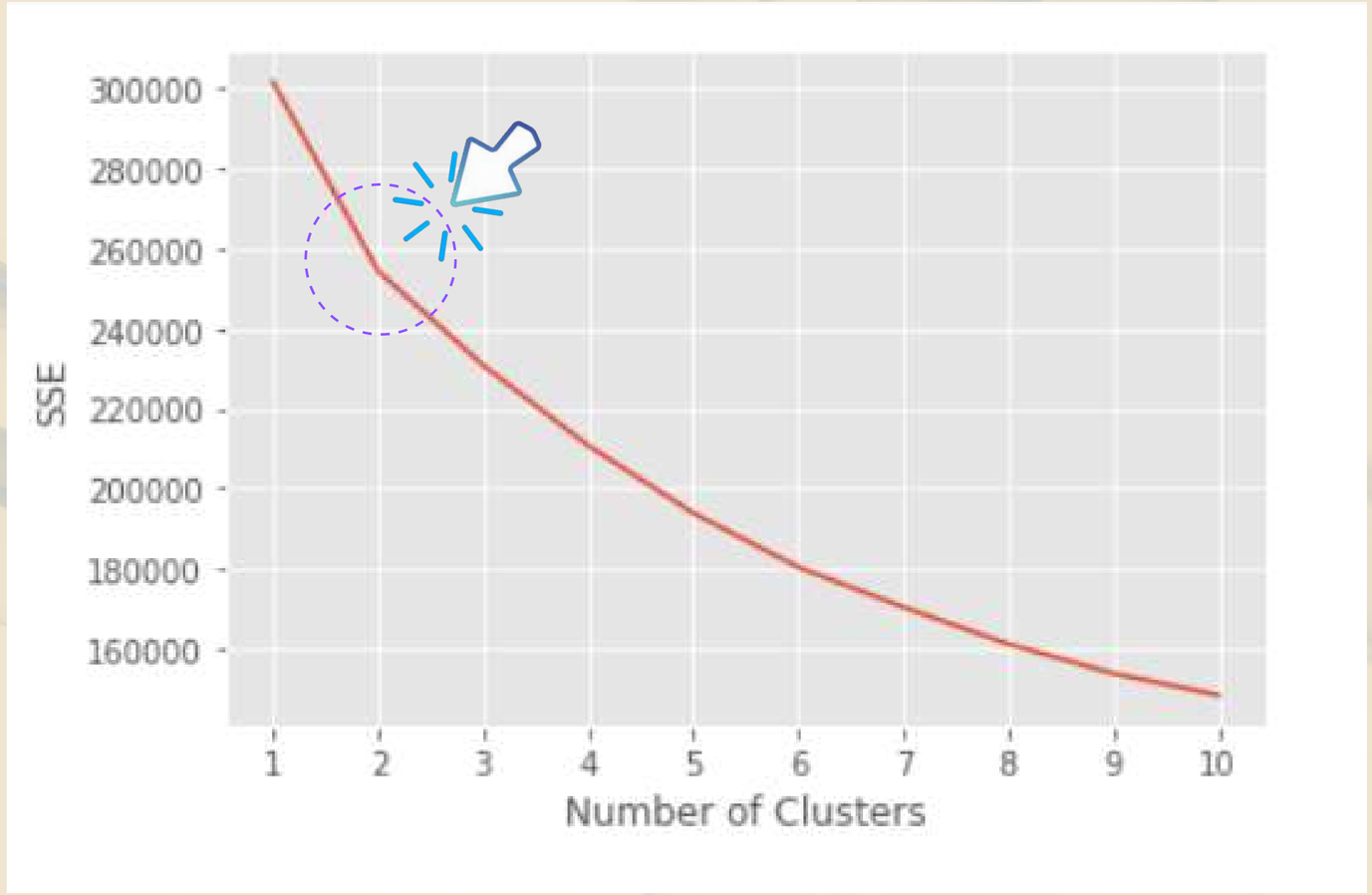
[29]
```
# convert categorical variables into binary form
df = pd.get_dummies(data, columns=['age', 'workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex',  'hours-per-week', 'native-country', 'salary'])

# Generate frequent itemsets
frequent_itemsets = apriori(df, min_support=0.15, use_colnames=True)

# Create DataFrame with support and itemsets columns
frequent_itemsets_df = pd.DataFrame(frequent_itemsets)

# generate association rules
rules = association_rules(frequent_itemsets_df.reset_index(), metric='lift', min_threshold=1)

# sort the rules by lift in descending order
rules = rules.sort_values(by='lift', ascending=False)

# show useful columns only
acc = rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']]
acc
```

	antecedents	consequents	support	confidence	lift
2264	(sex_1, race_4, salary_1, marital-status_2)	(relationship_0, native-country_38)	0.165937	0.954788	2.522849
2309	(relationship_0, native-country_38)	(sex_1, race_4, salary_1, marital-status_2)	0.165937	0.438458	2.522849
2291	(relationship_0, race_4, salary_1)	(sex_1, native-country_38, marital-status_2)	0.165937	0.957711	2.512306
2282	(sex_1, native-country_38, marital-status_2)	(relationship_0, race_4, salary_1)	0.165937	0.435293	2.512306
2310	(relationship_0, race_4)	(sex_1, native-country_38, salary_1, marital-s...	0.165937	0.442021	2.505587
...	...	...	...	...	...
465	(sex_0, salary_0)	(native-country_38)	0.262184	0.912111	1.000258
415	(relationship_1, salary_0)	(race_4)	0.196804	0.859916	1.000146
418	(race_4)	(relationship_1, salary_0)	0.196804	0.228898	1.000146
1314	(sex_1, salary_0)	(native-country_38, race_4)	0.372290	0.802989	1.000073
1315	(native-country_38, race_4)	(sex_1, salary_0)	0.372290	0.463663	1.000073

2318 rows × 5 columns

K-MEANS CLUSTERING

elbow method



K = 2

Define , Train and Predict - evaluate

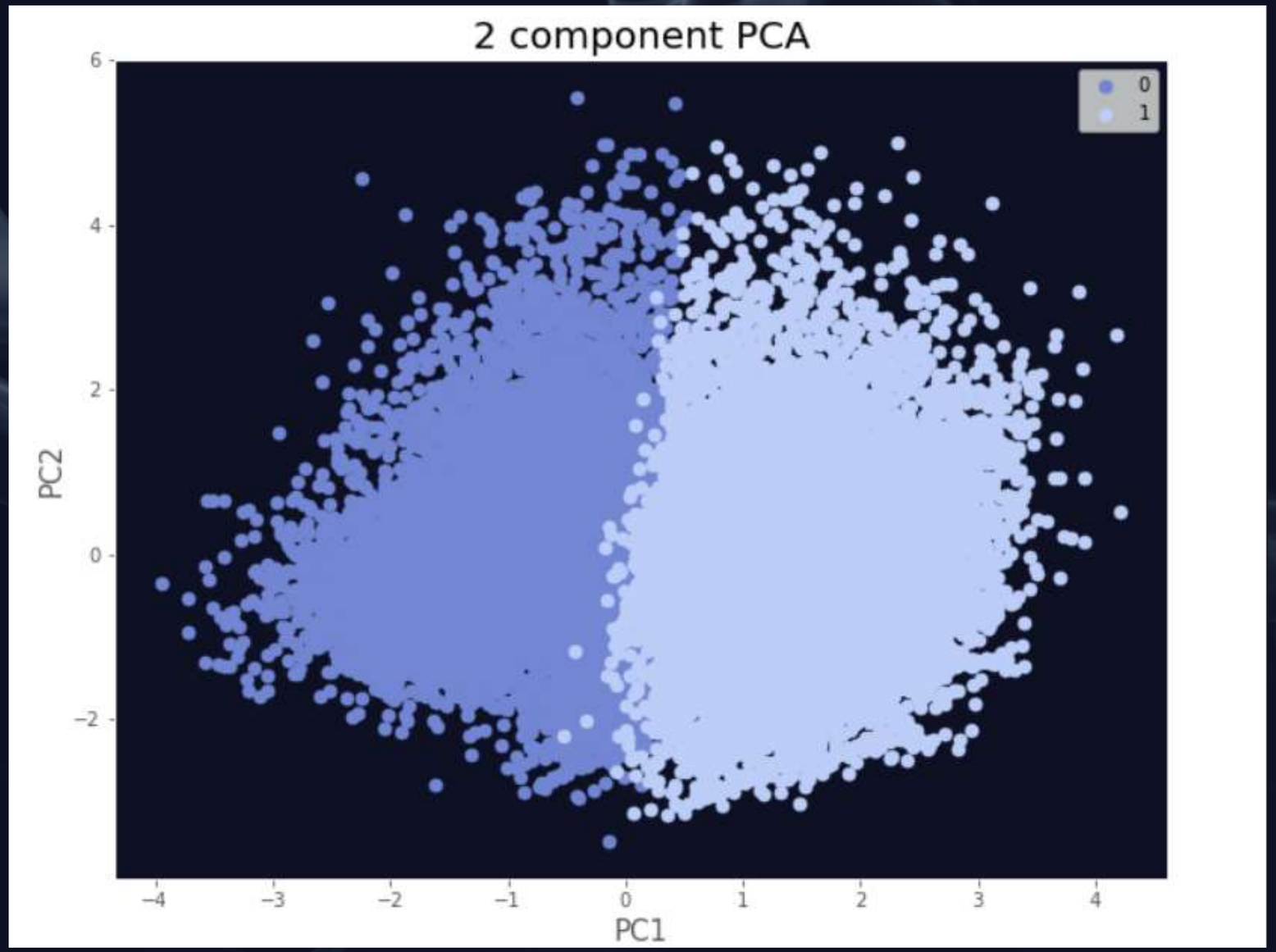
```
[57]  # set number of clusters
      k = 2
      kmeans = KMeans(n_clusters=k)

      # fit k-means clustering model
      kmeans.fit(subset_scaled)

      # get cluster labels
      data['cluster'] = kmeans.labels_
```

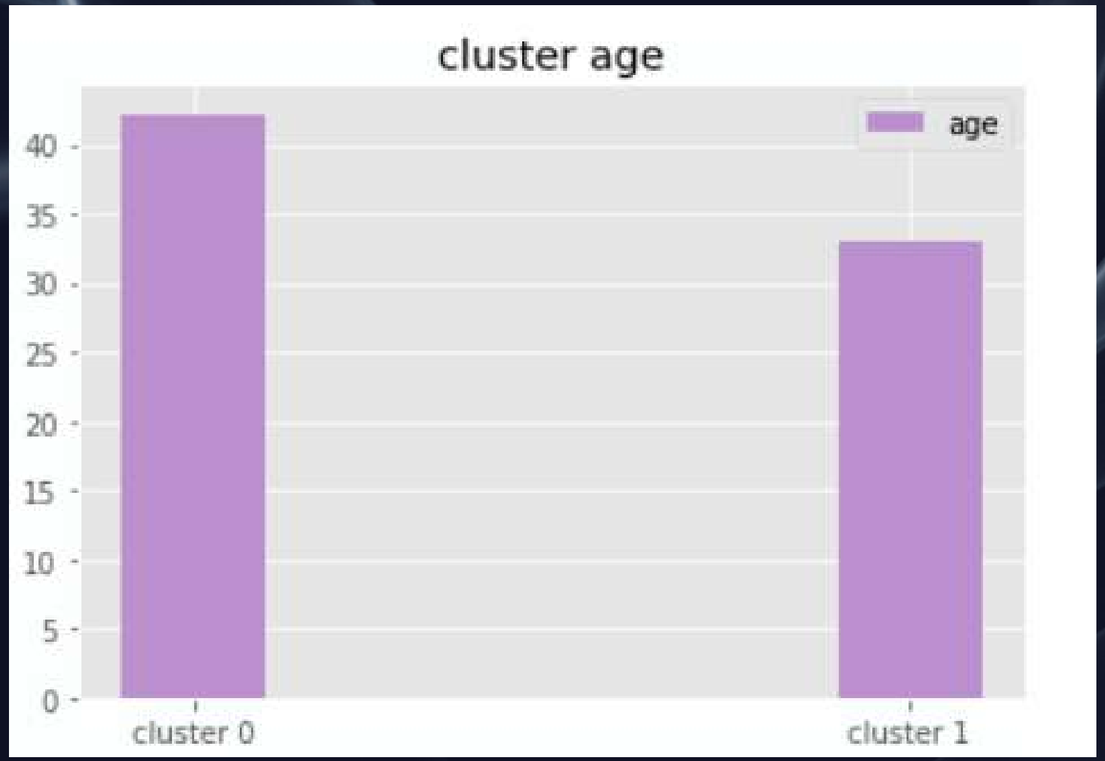
[58] data.head(5)

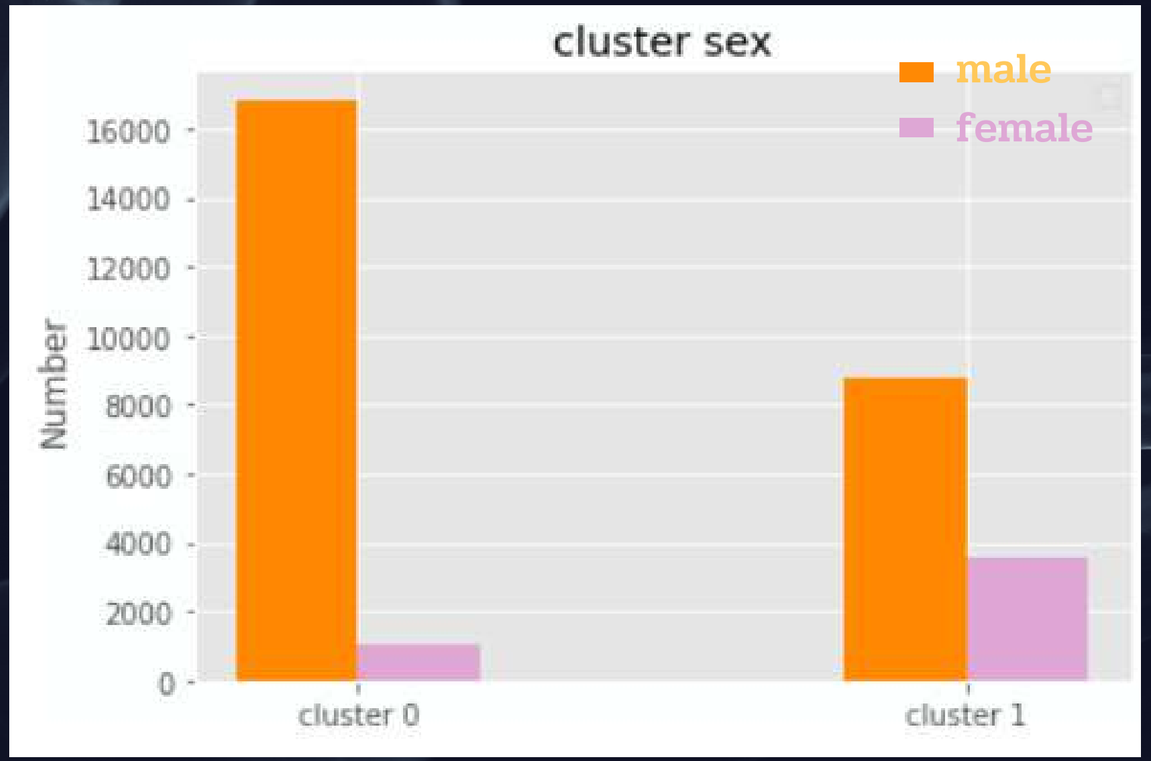
	age	workclass	education	marital-status	occupation	relationship	race	sex	hours-per-week	native-country	salary	cluster
0	39	5	9	4	0	1	4	1	40	38	0	0
1	50	4	9	2	3	0	4	1	13	38	0	0
2	38	2	11	0	5	1	4	1	40	38	0	0
3	53	2	1	2	5	0	2	1	40	38	0	0
4	28	2	9	2	9	5	2	0	40	4	0	1

[59] data['cluster'].value_counts()

```
0    17861
1    12301
Name: cluster, dtype: int64
```


2 component PCA
PC2
PC1
0
1
6
4
2
0
−2
−4
−3
−2
−1
0
1
2
3
4


cluster age
age
40
35
30
25
20
15
10
5
0
cluster 0
cluster 1


cluster sex
male
female
Number
16000
14000
12000
10000
8000
6000
4000
2000
0
cluster 0
cluster 1

THANK YOU !

MEMBERS

นางสาวณัชกมล อัมฤตานนท์
นางสาวทิพเกษร ยอดคุณ
นางสาวปิ่นมนัส สุระเกษ
นางสาวอภิศรา พงษ์เซียงซา