Term Project
Final

201631513 Hwang ByungHoon
201635825 Oh SeongWon
201635841 Lee ChaeHyeon
201835437 Kim JinKyung

- INDEX

01 Concept of project

02 Dataset description

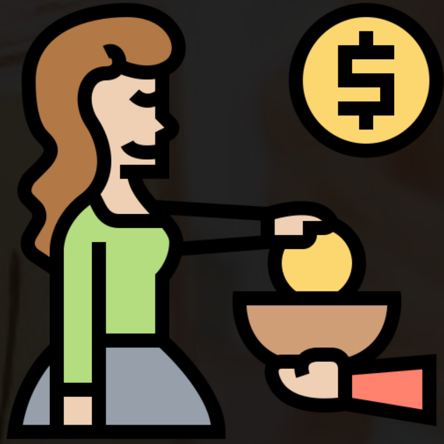
03 Data Preprocessing

04 Training & Evaluation

05 Conclusion

– Concept of project

In Classification



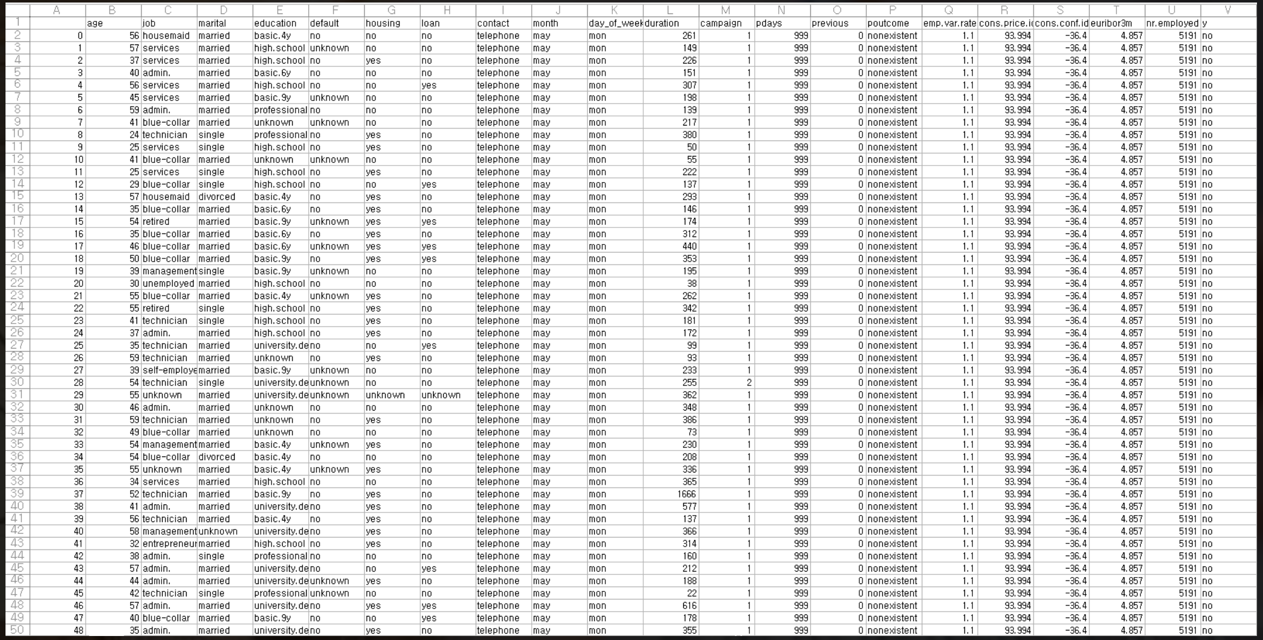
OR

The classification goal is
to predict if the client
will subscribe a term deposit

Term Project Final

- Dataset description

In Classification



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1		age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.i	cons.conf.id	euribor3m	nr.employed	y
2	0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
3	1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
4	2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
5	3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
6	4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
7	5	45	services	married	basic.9y	unknown	no	no	telephone	may	mon	198	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
8	6	59	admin.	married	professional	no	no	no	telephone	may	mon	139	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
9	7	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	217	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
10	8	24	technician	single	professional	no	yes	no	telephone	may	mon	380	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
11	9	25	services	single	high.school	no	yes	no	telephone	may	mon	50	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
12	10	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	55	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
13	11	25	services	single	high.school	no	yes	no	telephone	may	mon	222	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
14	12	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	137	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
15	13	57	housemaid	divorced	basic.4y	no	yes	no	telephone	may	mon	293	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
16	14	35	blue-collar	married	basic.6y	no	yes	no	telephone	may	mon	146	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
17	15	54	retired	married	basic.9y	unknown	yes	yes	telephone	may	mon	174	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
18	16	35	blue-collar	married	basic.6y	no	yes	no	telephone	may	mon	312	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
19	17	46	blue-collar	married	basic.6y	unknown	yes	yes	telephone	may	mon	440	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
20	18	50	blue-collar	married	basic.9y	no	yes	yes	telephone	may	mon	353	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
21	19	39	management	single	basic.9y	unknown	no	no	telephone	may	mon	195	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
22	20	30	unemployed	married	high.school	no	no	no	telephone	may	mon	38	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
23	21	55	blue-collar	married	basic.4y	unknown	yes	no	telephone	may	mon	262	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
24	22	55	retired	single	high.school	no	yes	no	telephone	may	mon	342	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
25	23	41	technician	single	high.school	no	yes	no	telephone	may	mon	181	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
26	24	37	admin.	married	high.school	no	yes	no	telephone	may	mon	172	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
27	25	35	technician	married	university.de	no	no	yes	telephone	may	mon	99	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
28	26	59	technician	married	unknown	no	yes	no	telephone	may	mon	93	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
29	27	39	self-employe	married	basic.9y	unknown	no	no	telephone	may	mon	233	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
30	28	54	technician	single	university.de	unknown	no	no	telephone	may	mon	255	2	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
31	29	55	unknown	married	university.de	unknown	unknown	unknown	telephone	may	mon	362	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
32	30	46	admin.	married	unknown	no	no	no	telephone	may	mon	348	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
33	31	59	technician	married	unknown	no	yes	no	telephone	may	mon	386	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
34	32	49	blue-collar	married	unknown	no	no	no	telephone	may	mon	73	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
35	33	54	management	married	basic.4y	unknown	yes	no	telephone	may	mon	230	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
36	34	54	blue-collar	divorced	basic.4y	no	no	no	telephone	may	mon	208	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
37	35	55	unknown	married	basic.4y	unknown	yes	no	telephone	may	mon	336	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
38	36	34	services	married	high.school	no	no	no	telephone	may	mon	365	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
39	37	52	technician	married	basic.9y	no	yes	no	telephone	may	mon	1666	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
40	38	41	admin.	married	university.de	no	yes	no	telephone	may	mon	577	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
41	39	56	technician	married	basic.4y	no	yes	no	telephone	may	mon	137	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
42	40	58	management	unknown	university.de	no	yes	no	telephone	may	mon	366	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
43	41	32	entrepreneur	married	high.school	no	yes	no	telephone	may	mon	314	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
44	42	38	admin.	single	professional	no	no	no	telephone	may	mon	160	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
45	43	57	admin.	married	university.de	no	no	yes	telephone	may	mon	212	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
46	44	44	admin.	married	university.de	unknown	yes	no	telephone	may	mon	188	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
47	45	42	technician	single	professional	unknown	no	no	telephone	may	mon	22	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
48	46	57	admin.	married	university.de	no	yes	yes	telephone	may	mon	616	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
49	47	40	blue-collar	married	basic.9y	no	no	yes	telephone	may	mon	178	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no
50	48	35	admin.	married	university.de	no	yes	no	telephone	may	mon	355	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191	no

File

bank-additional-full.csv

Size

21 Columns,
41188 Rows

Dataset Link

https://www.kaggle.com/henriqueyamahata/bank-marketing?select=bank-additional-full.csv

- Dataset

In Classification

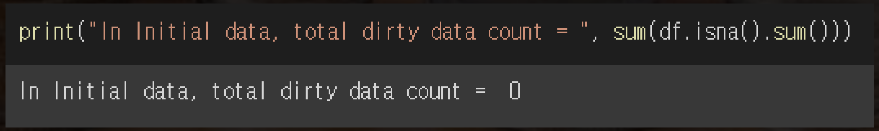
Numeric

- Age
- Duration
- Campaign
- Pdays
- Previous
- emp.var.rate
- cons.price.idx
- cons.conf.idx
- euribor3m
- nr.employed

Categorical

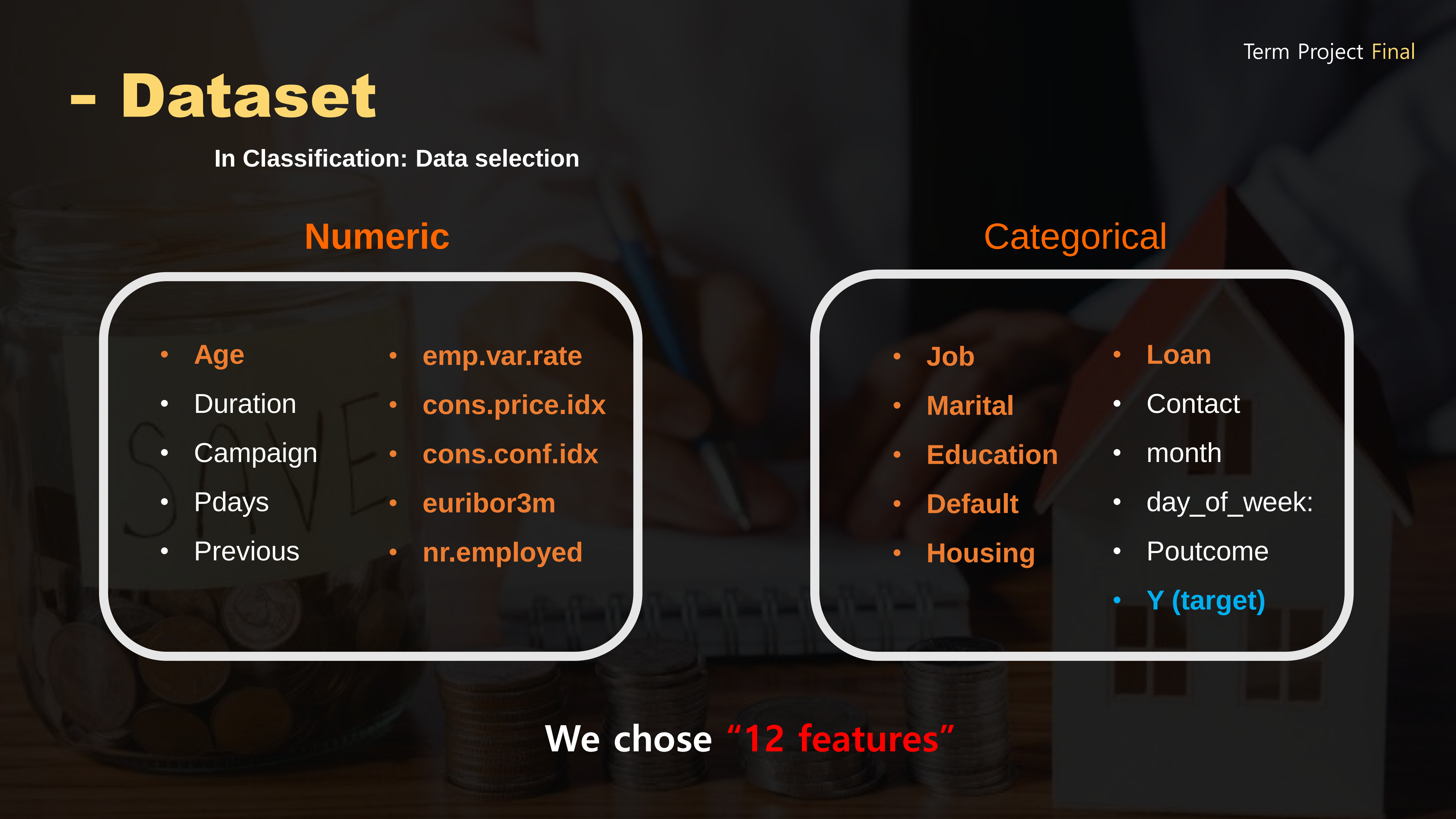
- Job
- Marital
- Education
- Default
- Housing
- Loan
- Contact
- month
- day_of_week:
- Poutcome
- **Y (target)**

“There is no null value in the data”



```
print("In Initial data, total dirty data count = ", sum(df.isna().sum()))
```

```
In Initial data, total dirty data count =  0
```



– Dataset

In Classification: Data selection

Numeric

- **Age**
- Duration
- Campaign
- Pdays
- Previous
- **emp.var.rate**
- **cons.price.idx**
- **cons.conf.idx**
- **euribor3m**
- **nr.employed**

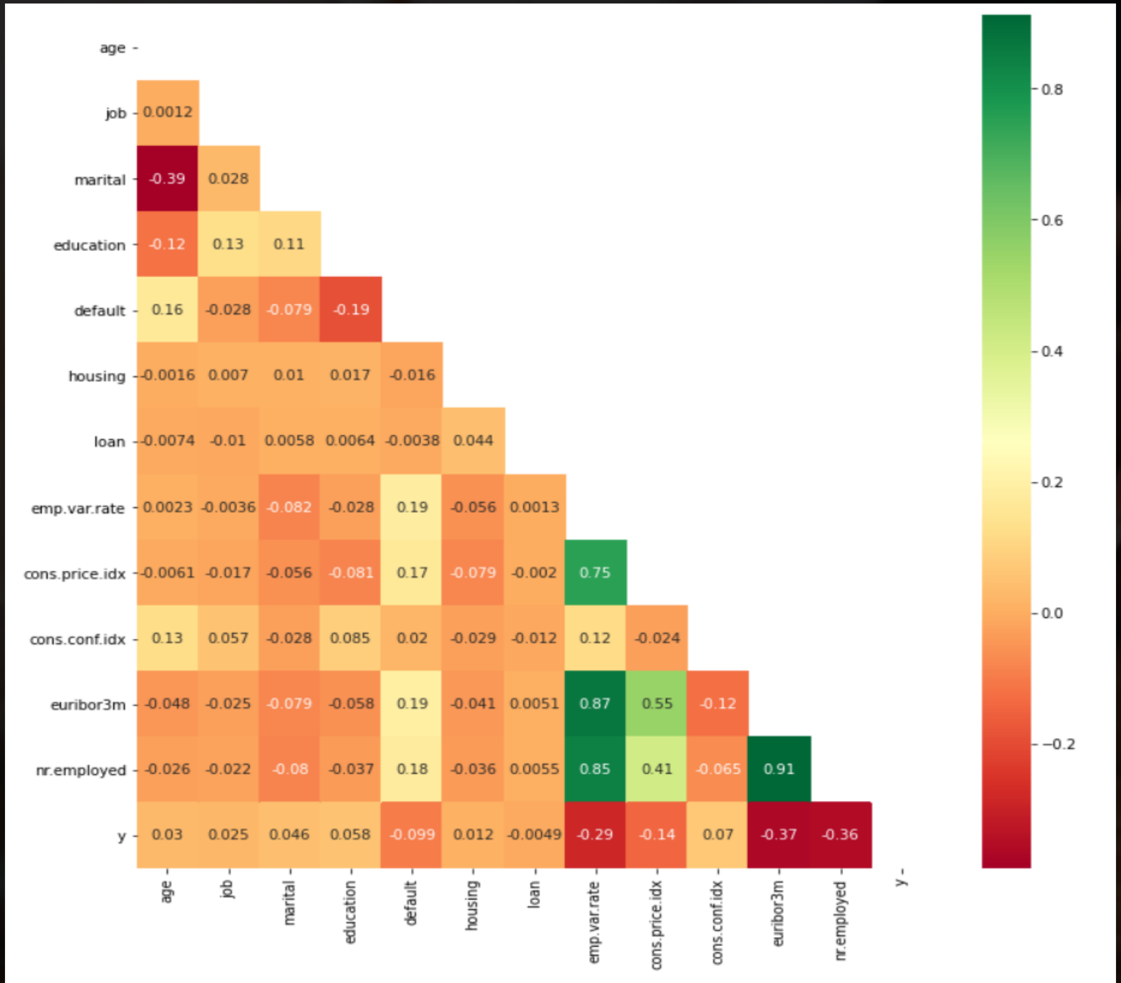
Categorical

- **Job**
- **Marital**
- **Education**
- **Default**
- **Housing**
- **Loan**
- Contact
- month
- day_of_week:
- Poutcome
- **Y (target)**

We chose "12 features"

- Dataset

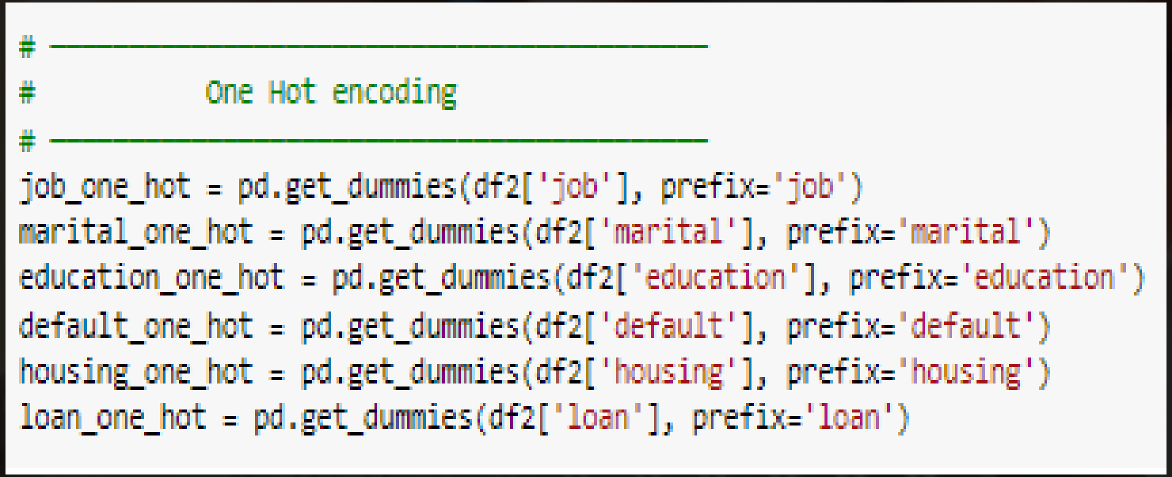
In Classification : Correlation heatmap



- Data Preprocessing

In Classification

Apply One Hot Encoding to Categorical data



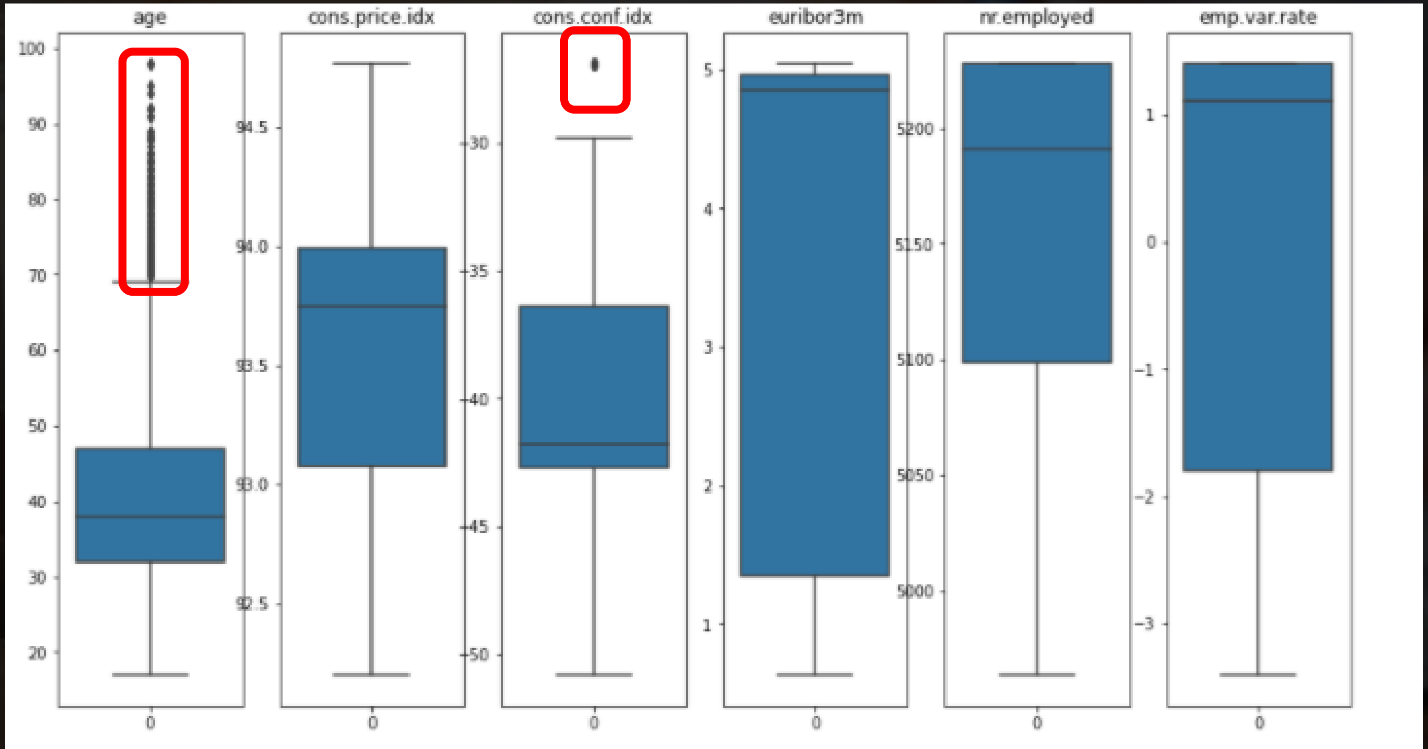
```
# ---------------------------------------------
#           One Hot encoding
# ---------------------------------------------
job_one_hot = pd.get_dummies(df2['job'], prefix='job')
marital_one_hot = pd.get_dummies(df2['marital'], prefix='marital')
education_one_hot = pd.get_dummies(df2['education'], prefix='education')
default_one_hot = pd.get_dummies(df2['default'], prefix='default')
housing_one_hot = pd.get_dummies(df2['housing'], prefix='housing')
loan_one_hot = pd.get_dummies(df2['loan'], prefix='loan')
```

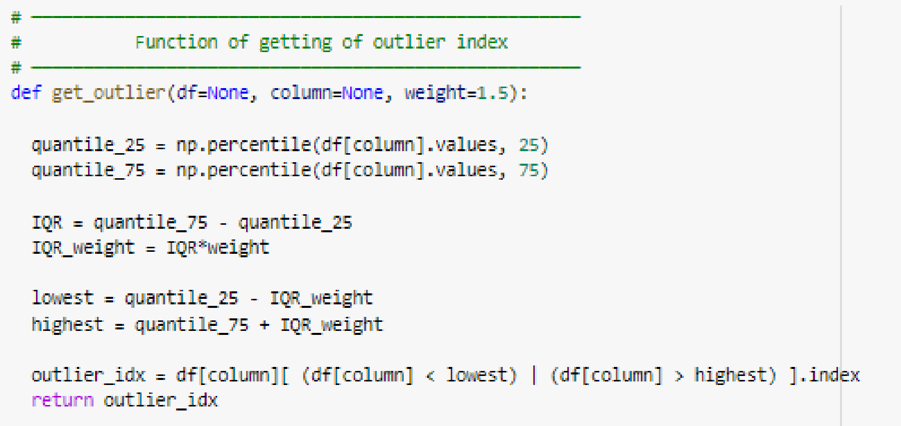
Rows: 41188, Columns: 13

→ Rows: 41188, **Columns: 40**

- Data Preprocessing

In Classification : Data Inspection & Outlier detection

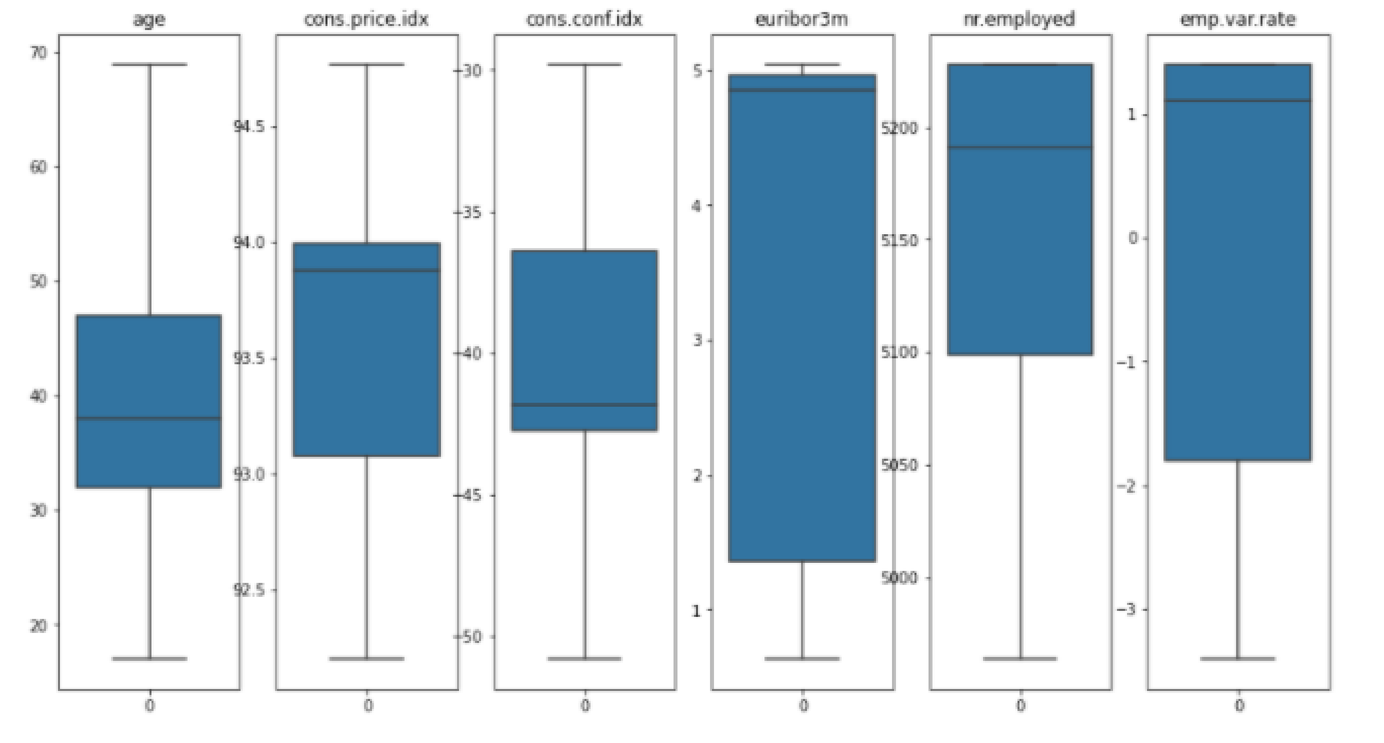




```
# ----------------------------------------------------------
#              Function of getting of outlier index
# ----------------------------------------------------------
def get_outlier(df=None, column=None, weight=1.5):

  quantile_25 = np.percentile(df[column].values, 25)
  quantile_75 = np.percentile(df[column].values, 75)

  IQR = quantile_75 - quantile_25
  IQR_weight = IQR*weight

  lowest = quantile_25 - IQR_weight
  highest = quantile_75 + IQR_weight

  outlier_idx = df[column][ (df[column] < lowest) | (df[column] > highest) ].index
  return outlier_idx
```

- Data Preprocessing

In Classification : Data Inspection & Outlier detection

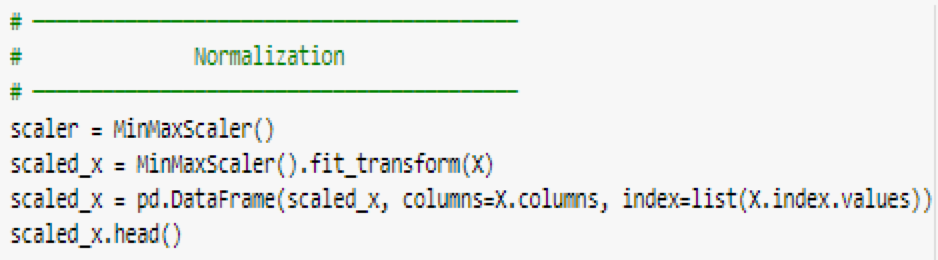


- Data Preprocessing

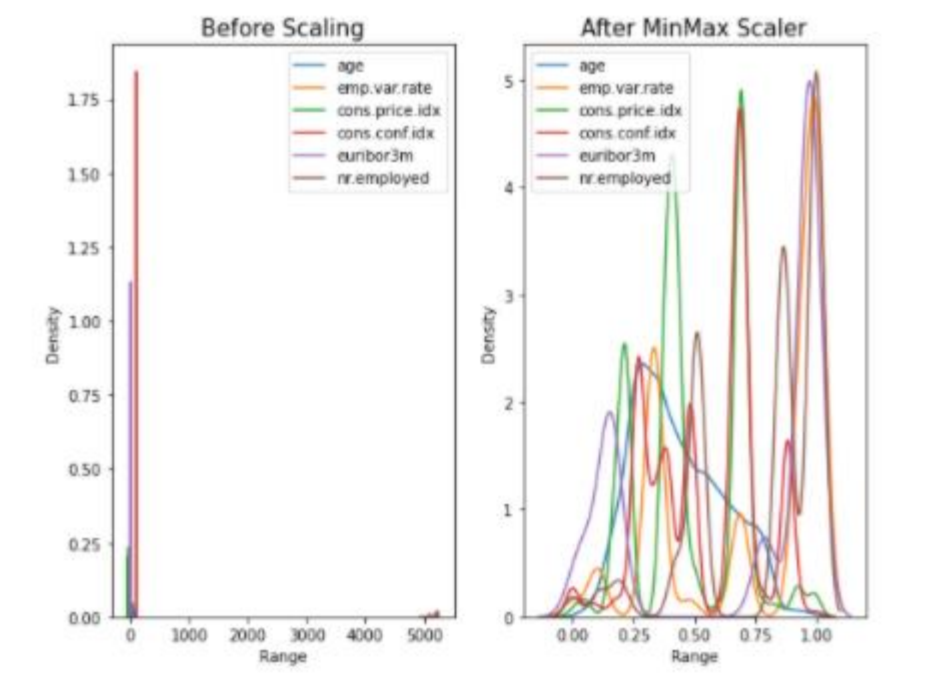
In Classification : Data Preparation

- Feature Scaling

min-max Scaler

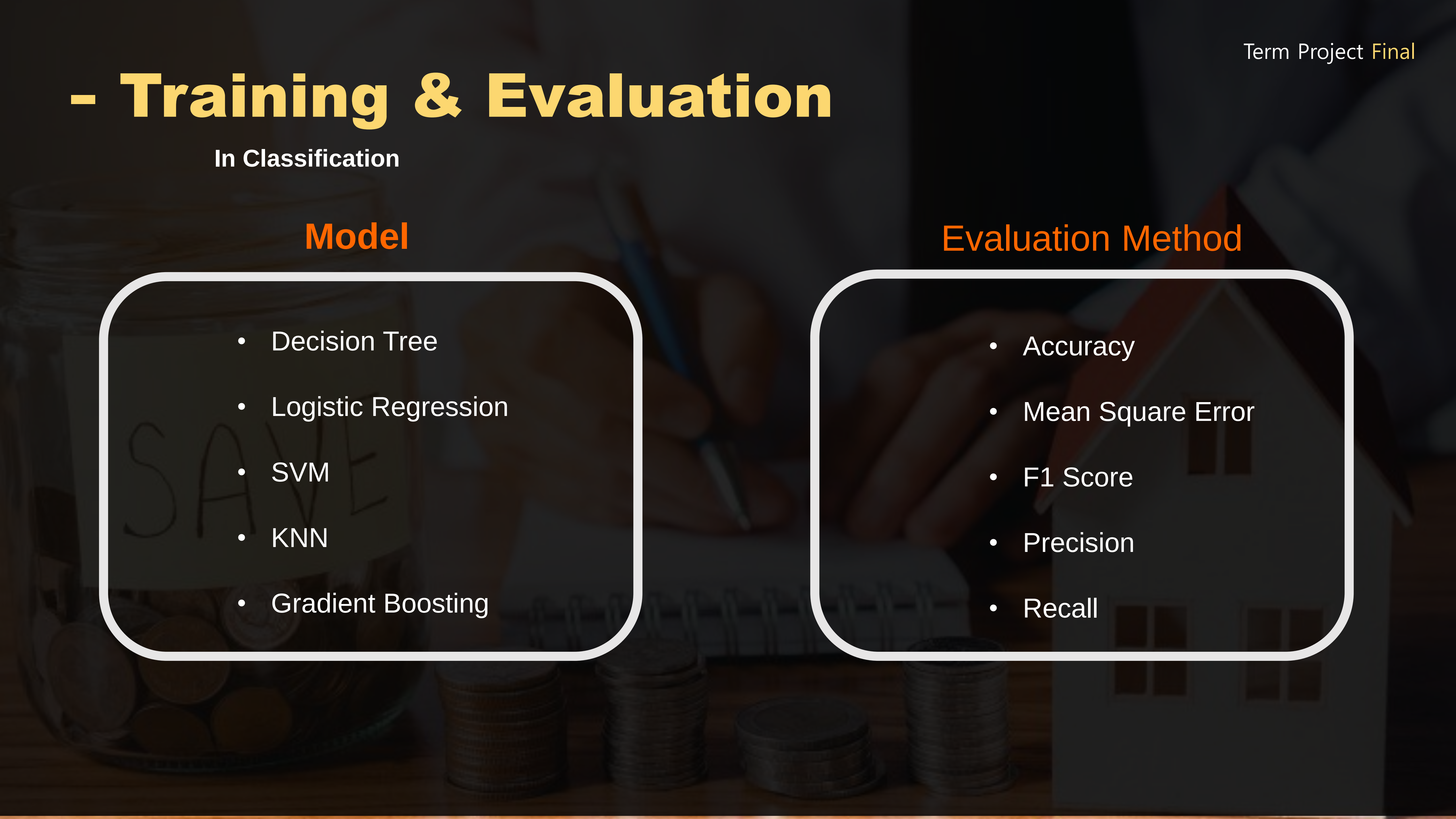


```
# ────────────────────────────────────────
#              Normalization
# ────────────────────────────────────────
scaler = MinMaxScaler()
scaled_x = MinMaxScaler().fit_transform(X)
scaled_x = pd.DataFrame(scaled_x, columns=X.columns, index=list(X.index.values))
scaled_x.head()
```



- Training & Evaluation

In Classification



Model

- Decision Tree
- Logistic Regression
- SVM
- KNN
- Gradient Boosting

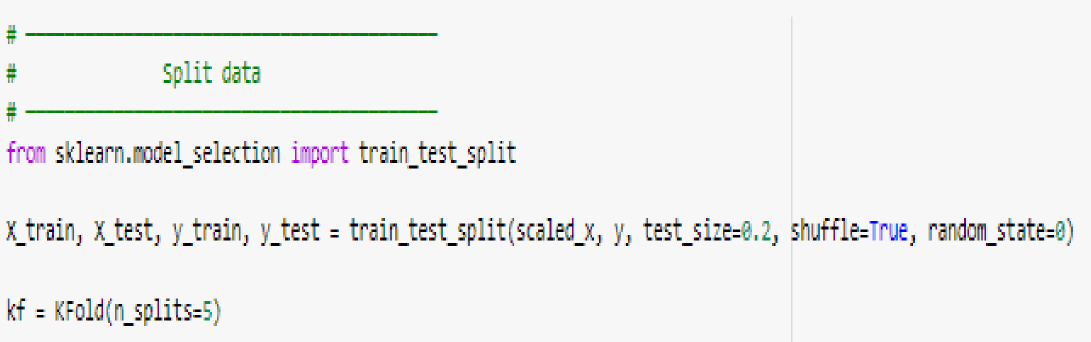
Evaluation Method

- Accuracy
- Mean Square Error
- F1 Score
- Precision
- Recall

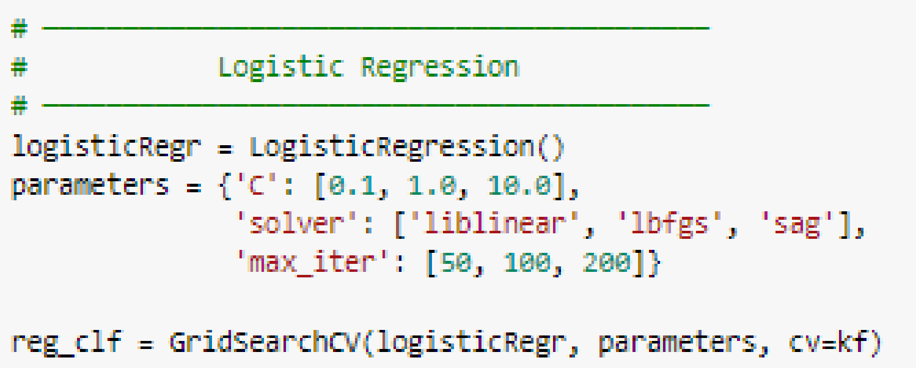
- Training & Evaluation

In Classification

Training Setting



```
# ------------------------------------------
#               Split data
# ------------------------------------------
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(scaled_x, y, test_size=0.2, shuffle=True, random_state=0)

kf = KFold(n_splits=5)
```



```
# ------------------------------------------
#            Logistic Regression
# ------------------------------------------
logisticRegr = LogisticRegression()
parameters = {'C': [0.1, 1.0, 10.0],
              'solver': ['liblinear', 'lbfgs', 'sag'],
              'max_iter': [50, 100, 200]}

reg_clf = GridSearchCV(logisticRegr, parameters, cv=kf)
```

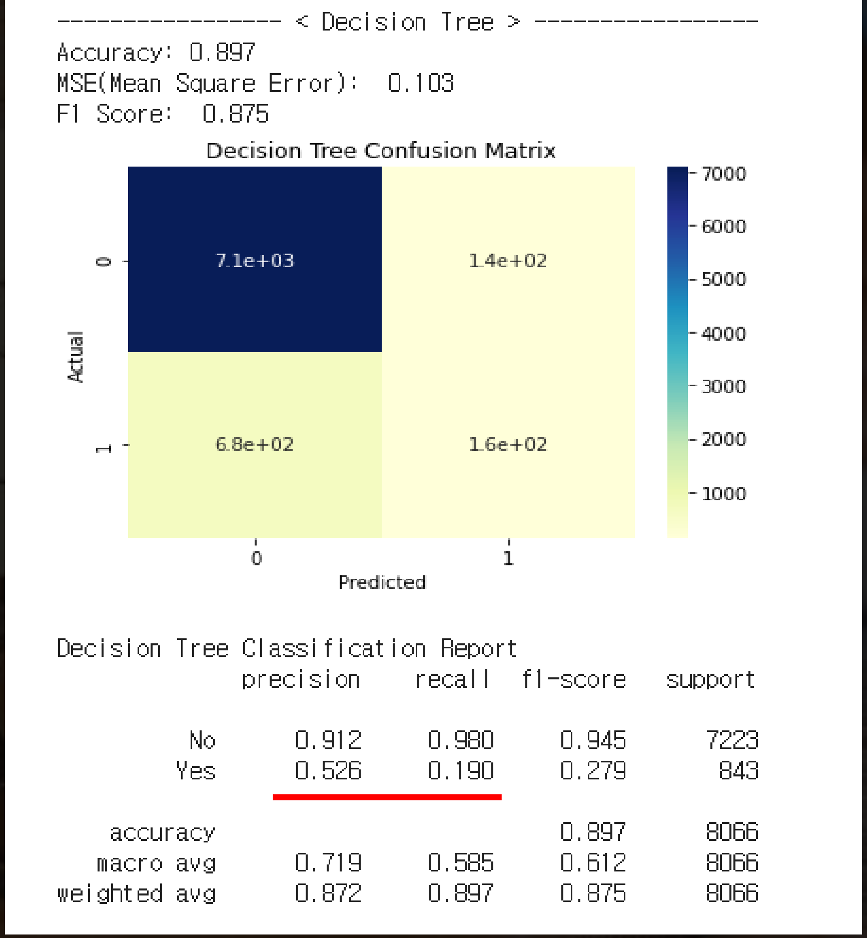
Training data and test data were divided into 20% ratios

The optimal parameters were found using GridSearchCV

- Training & Evaluation

In Classification

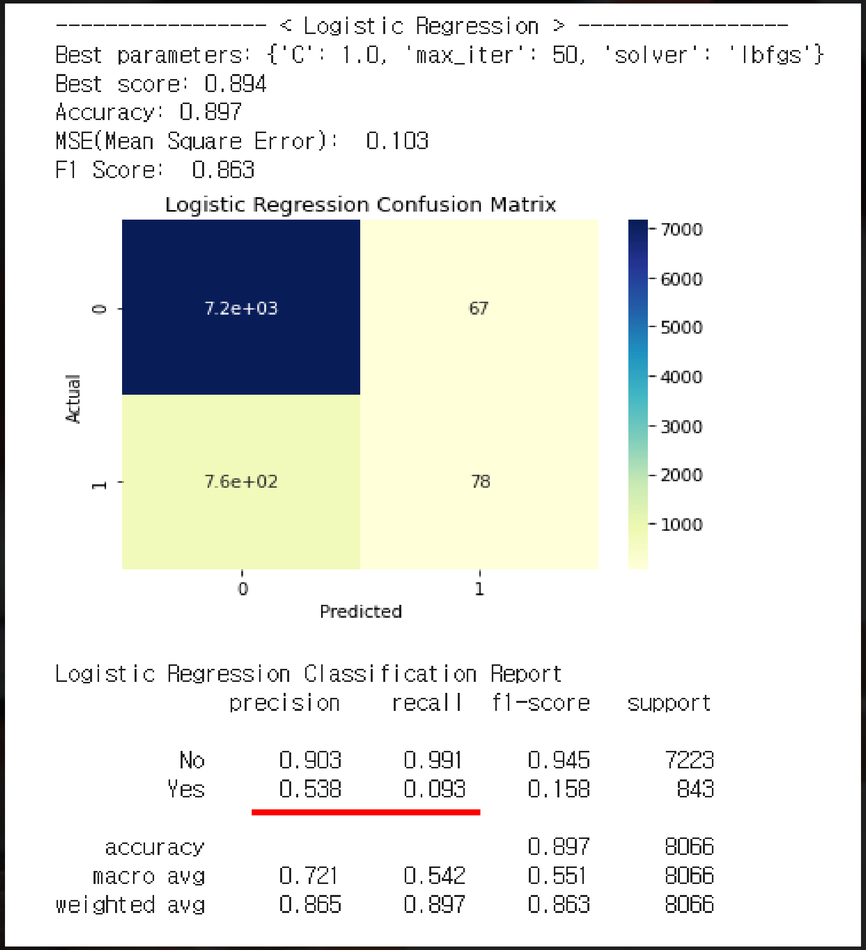
Decision Tree



```
------------------ < Decision Tree > ------------------
Accuracy: 0.897
MSE(Mean Square Error):  0.103
F1 Score:  0.875
```

```
Decision Tree Classification Report
                 precision    recall  f1-score   support

          No       0.912     0.980     0.945      7223
         Yes       0.526     0.190     0.279       843

    accuracy                           0.897      8066
   macro avg       0.719     0.585     0.612      8066
weighted avg       0.872     0.897     0.875      8066
```

Logistic Regression classification



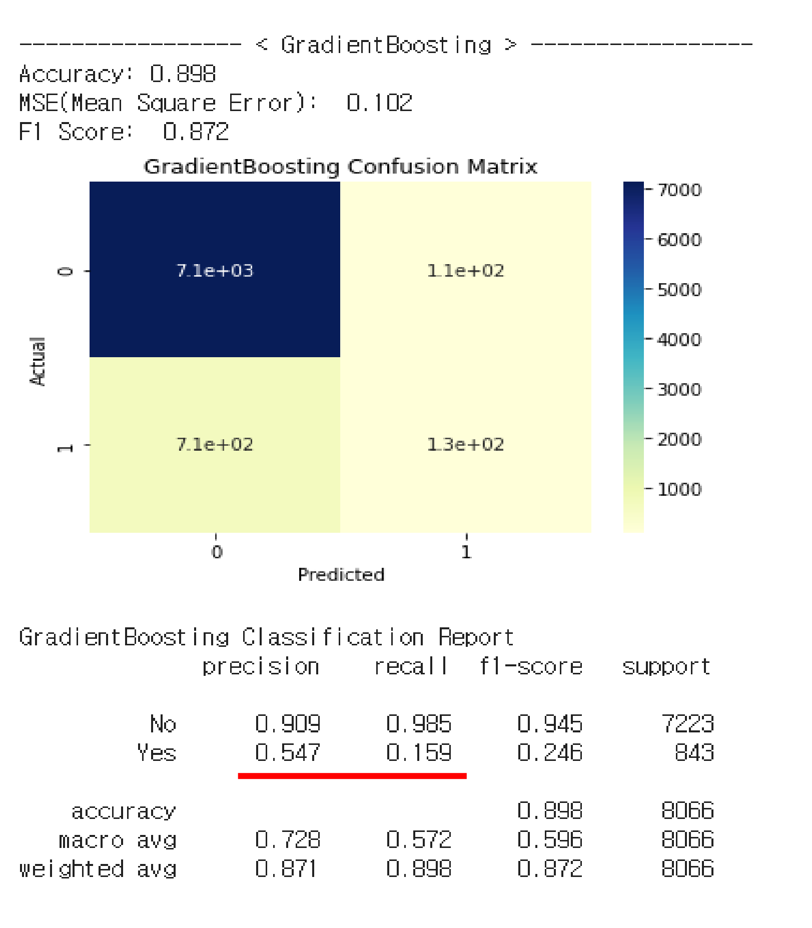
```
----------------- < Logistic Regression > -----------------
Best parameters: {'C': 1.0, 'max_iter': 50, 'solver': 'lbfgs'}
Best score: 0.894
Accuracy: 0.897
MSE(Mean Square Error):  0.103
F1 Score:  0.863
```

```
Logistic Regression Classification Report
                 precision    recall  f1-score   support

          No       0.903     0.991     0.945      7223
         Yes       0.538     0.093     0.158       843

    accuracy                           0.897      8066
   macro avg       0.721     0.542     0.551      8066
weighted avg       0.865     0.897     0.863      8066
```

- Training & Evaluation

In Classification

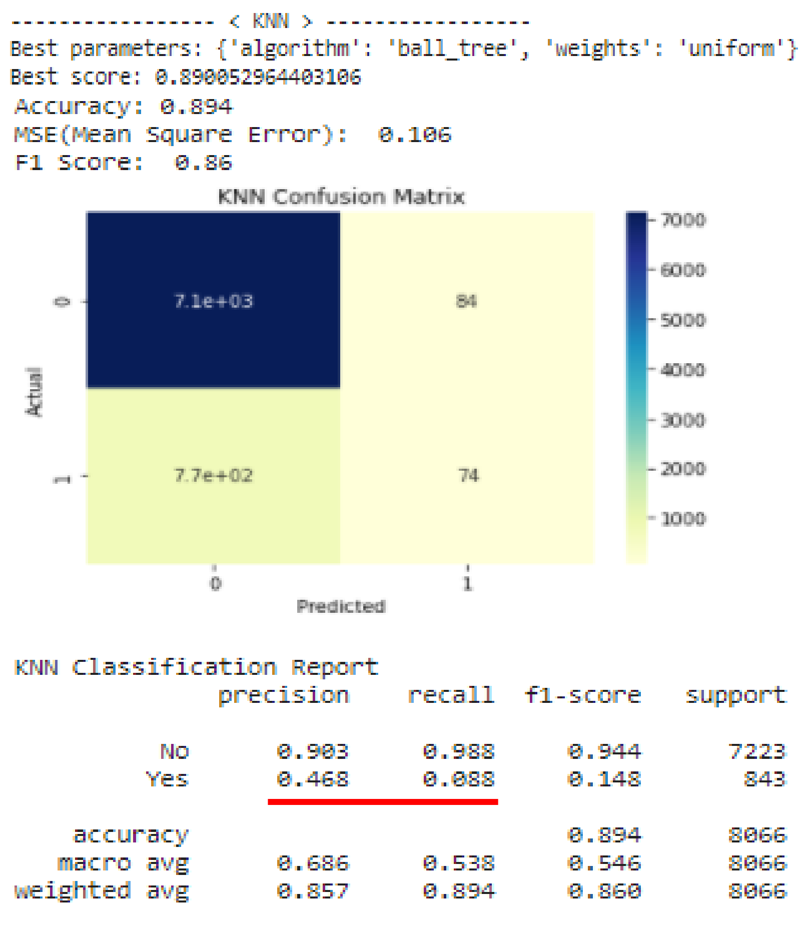
Gradient Boosting



```
----------------- < GradientBoosting > -----------------
Accuracy: 0.898
MSE(Mean Square Error):  0.102
F1 Score:  0.872
```

```
GradientBoosting Classification Report
              precision    recall  f1-score   support

          No      0.909     0.985     0.945      7223
         Yes      0.547     0.159     0.246       843

    accuracy                          0.898      8066
   macro avg      0.728     0.572     0.596      8066
weighted avg      0.871     0.898     0.872      8066
```

KNN



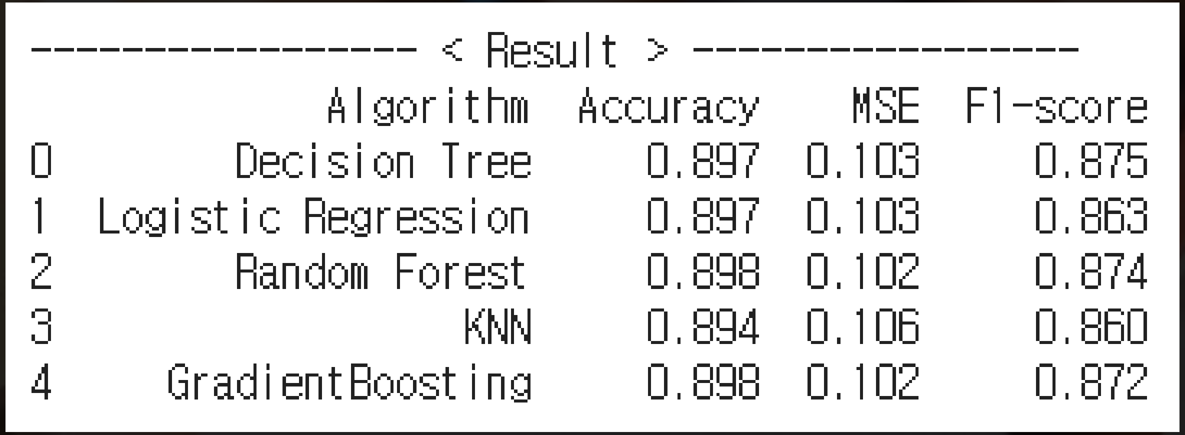
```
----------------- < KNN > -----------------
Best parameters: {'algorithm': 'ball_tree', 'weights': 'uniform'}
Best score: 0.890052964403106
Accuracy: 0.894
MSE(Mean Square Error):  0.106
F1 Score:  0.86
```

```
KNN Classification Report
              precision    recall  f1-score   support

          No      0.903     0.988     0.944      7223
         Yes      0.468     0.088     0.148       843

    accuracy                          0.894      8066
   macro avg      0.686     0.538     0.546      8066
weighted avg      0.857     0.894     0.860      8066
```

- Training & Evaluation

In Classification

Final result



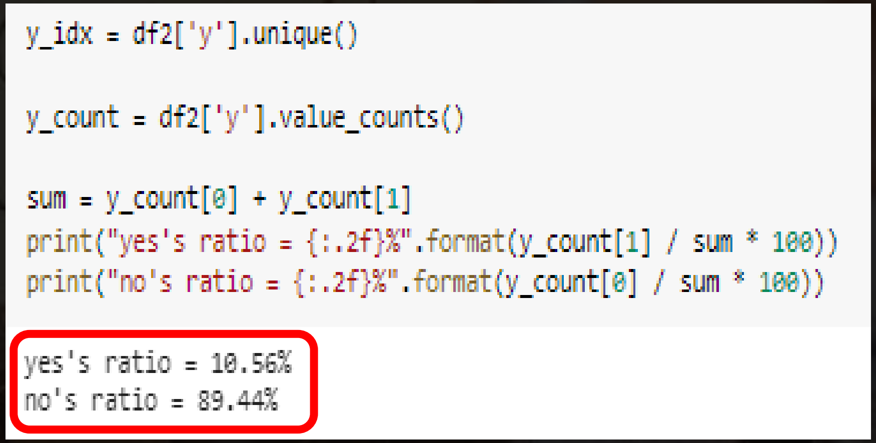
```
----------------- < Result > -----------------
           Algorithm  Accuracy    MSE  F1-score
0      Decision Tree     0.897  0.103     0.875
1  Logistic Regression   0.897  0.103     0.863
2      Random Forest     0.898  0.102     0.874
3                KNN     0.894  0.106     0.860
4    GradientBoosting    0.898  0.102     0.872
```

- Training & Evaluation

In Classification

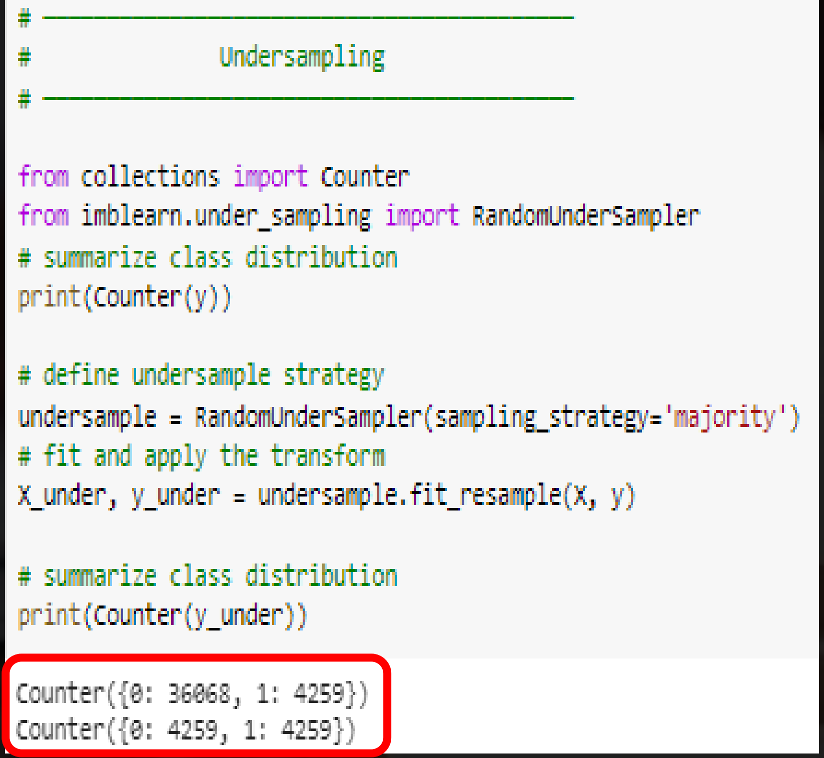
Evaluation analysis

There is a difference in the ratio of target data



```
y_idx = df2['y'].unique()

y_count = df2['y'].value_counts()

sum = y_count[0] + y_count[1]
print("yes's ratio = {:.2f}%".format(y_count[1] / sum * 100))
print("no's ratio = {:.2f}%".format(y_count[0] / sum * 100))
```

```
yes's ratio = 10.56%
no's ratio = 89.44%
```



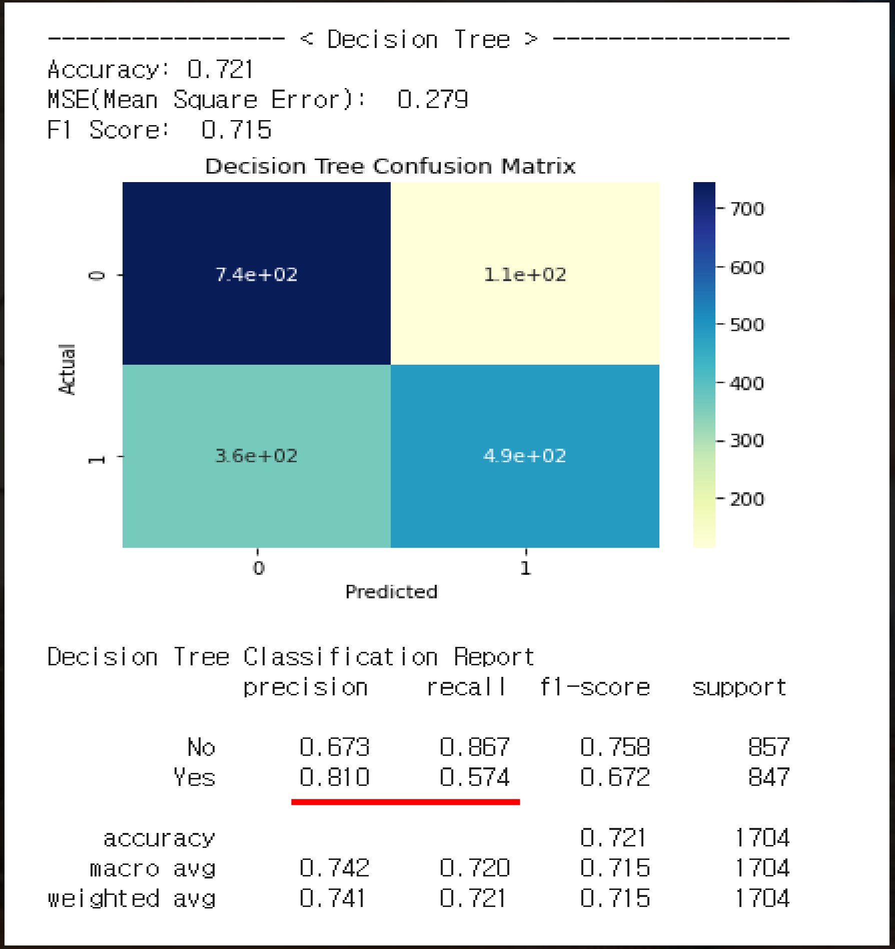
```
# ----------------------------------------------------
#                Undersampling
# ----------------------------------------------------

from collections import Counter
from imblearn.under_sampling import RandomUnderSampler
# summarize class distribution
print(Counter(y))

# define undersample strategy
undersample = RandomUnderSampler(sampling_strategy='majority')
# fit and apply the transform
X_under, y_under = undersample.fit_resample(X, y)

# summarize class distribution
print(Counter(y_under))
```

```
Counter({0: 36068, 1: 4259})
Counter({0: 4259, 1: 4259})
```

– Training & Evaluation

In Classification

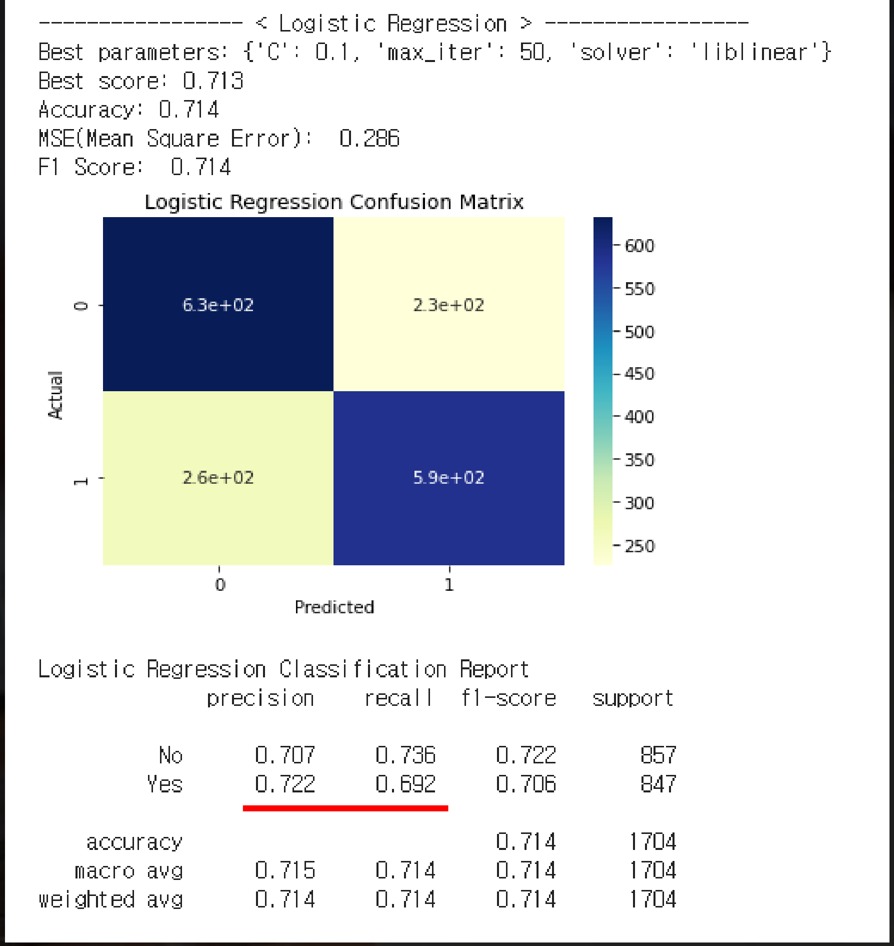
Decision Tree



```
----------------- < Decision Tree > -----------------
Accuracy: 0.721
MSE(Mean Square Error):  0.279
F1 Score:  0.715
```

```
Decision Tree Classification Report
              precision    recall  f1-score   support

          No      0.673     0.867     0.758       857
         Yes      0.810     0.574     0.672       847

    accuracy                          0.721      1704
   macro avg      0.742     0.720     0.715      1704
weighted avg      0.741     0.721     0.715      1704
```

Logistic Regression



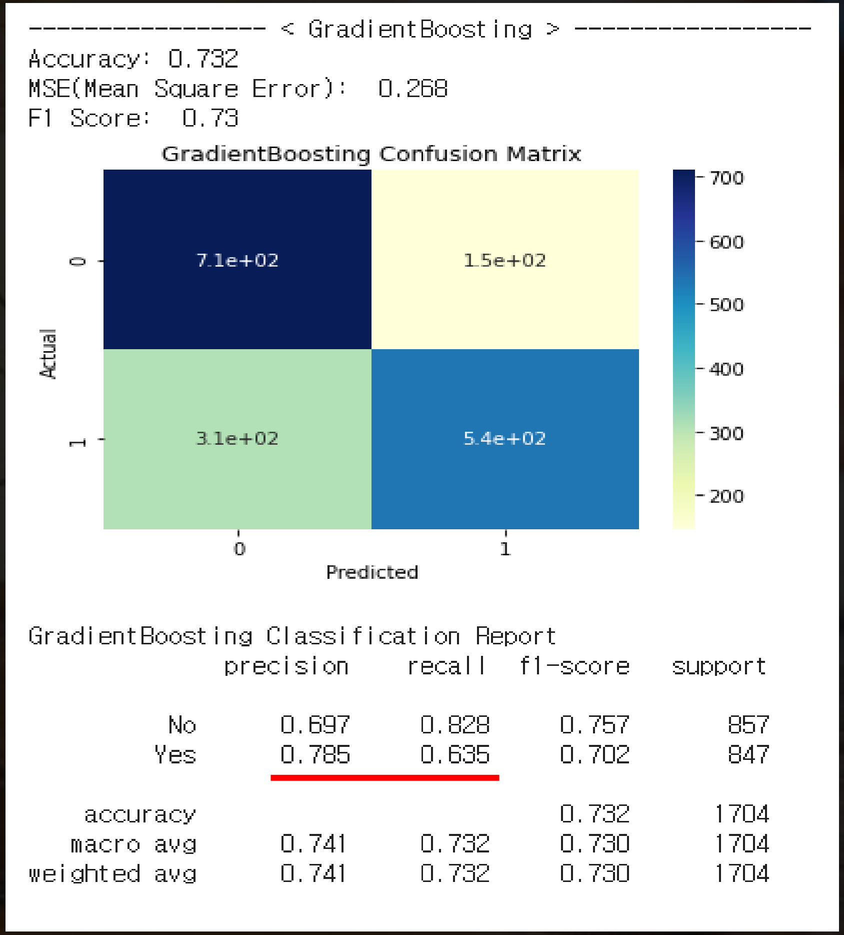
```
----------------- < Logistic Regression > -----------------
Best parameters: {'C': 0.1, 'max_iter': 50, 'solver': 'liblinear'}
Best score: 0.713
Accuracy: 0.714
MSE(Mean Square Error):  0.286
F1 Score:  0.714
```

```
Logistic Regression Classification Report
              precision    recall  f1-score   support

          No      0.707     0.736     0.722       857
         Yes      0.722     0.692     0.706       847

    accuracy                          0.714      1704
   macro avg      0.715     0.714     0.714      1704
weighted avg      0.714     0.714     0.714      1704
```

- Training & Evaluation

In Classification

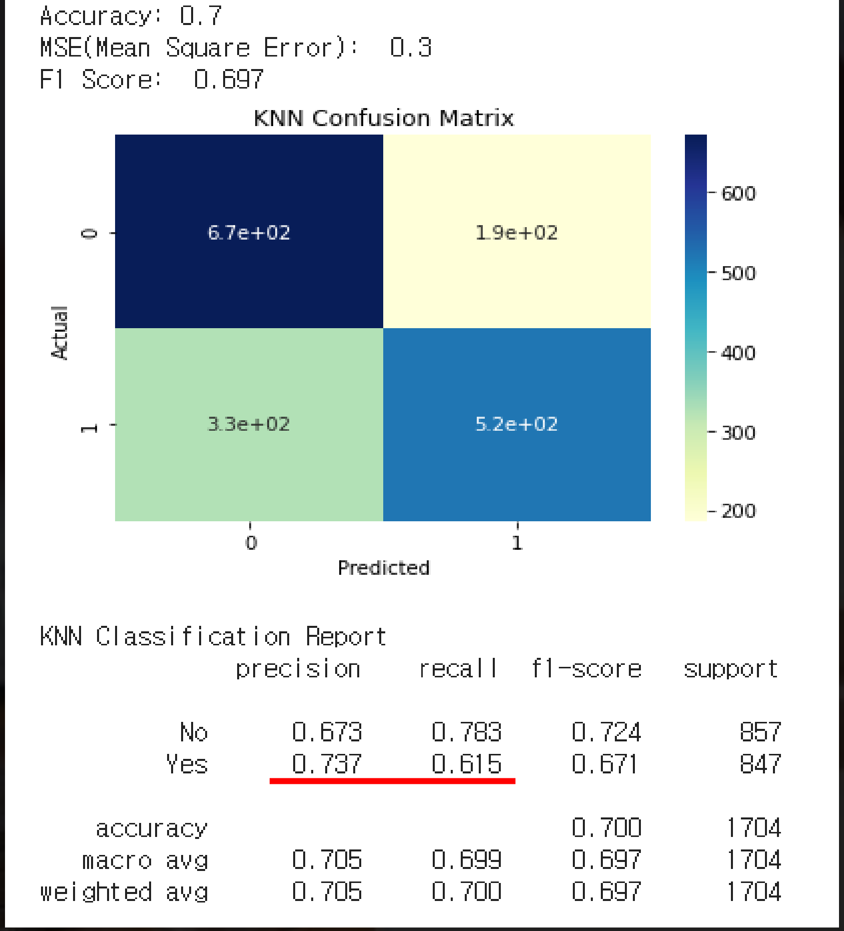
Gradient Boosting



```
----------------- < GradientBoosting > -----------------
Accuracy: 0.732
MSE(Mean Square Error):  0.268
F1 Score:  0.73
```

```
GradientBoosting Classification Report
              precision    recall  f1-score   support

          No      0.697     0.828     0.757       857
         Yes      0.785     0.635     0.702       847

    accuracy                          0.732      1704
   macro avg      0.741     0.732     0.730      1704
weighted avg      0.741     0.732     0.730      1704
```

KNN



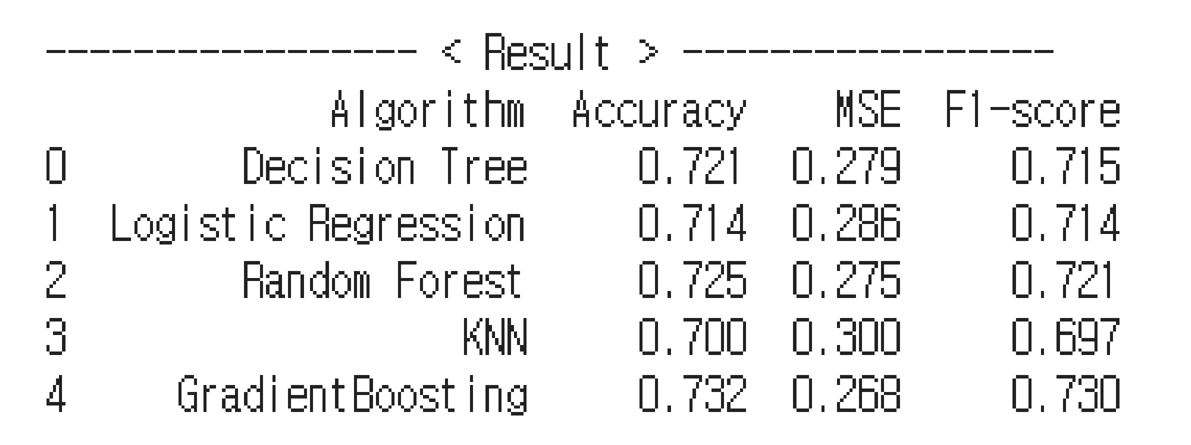
```
Accuracy: 0.7
MSE(Mean Square Error):  0.3
F1 Score:  0.697
```

```
KNN Classification Report
              precision    recall  f1-score   support

          No      0.673     0.783     0.724       857
         Yes      0.737     0.615     0.671       847

    accuracy                          0.700      1704
   macro avg      0.705     0.699     0.697      1704
weighted avg      0.705     0.700     0.697      1704
```

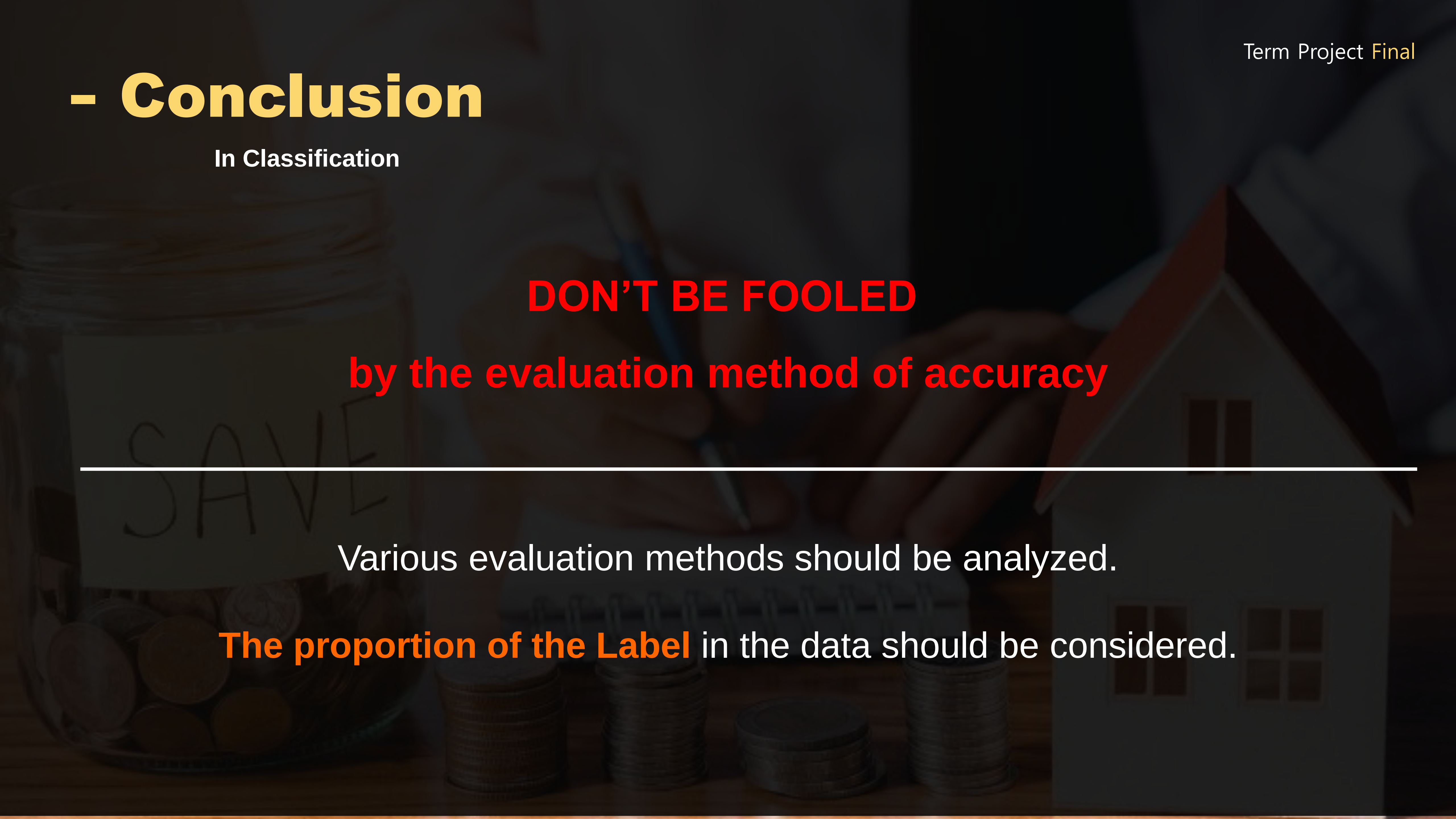
- Training & Evaluation

In Classification

Final result



```
----------------- < Result > -----------------
            Algorithm  Accuracy    MSE  F1-score
0       Decision Tree     0.721  0.279     0.715
1 Logistic Regression     0.714  0.286     0.714
2       Random Forest     0.725  0.275     0.721
3                 KNN     0.700  0.300     0.697
4    GradientBoosting     0.732  0.268     0.730
```



– Conclusion

In Classification

DON’T BE FOOLED

by the evaluation method of accuracy

Various evaluation methods should be analyzed.

The proportion of the Label in the data should be considered.

Clustering

– Concept of project

In Clustering

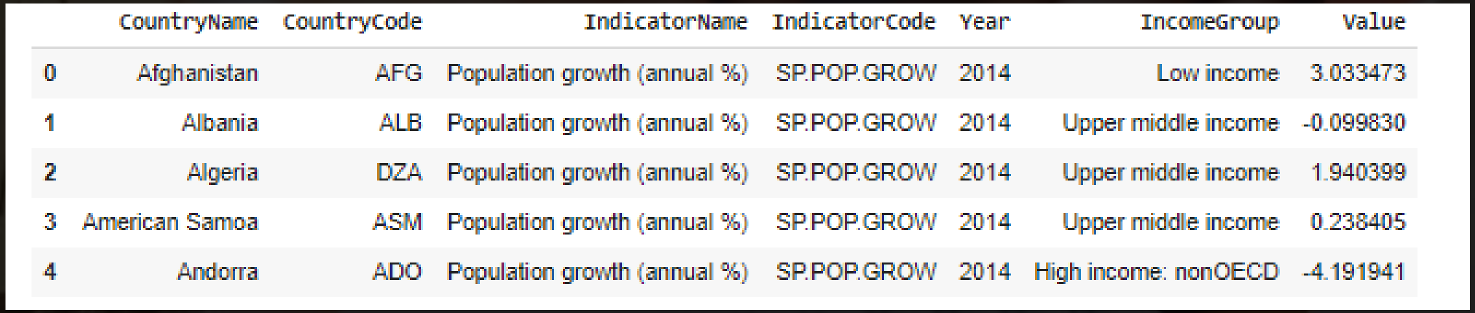
The clustering goal is
to cluster the relationship between income groups
and population growth

- Dataset

In Clustering

Indicators.csv → Country code & 2014 Population growth

Country.csv → Country code & InComeGroup



	CountryName	CountryCode	IndicatorName	IndicatorCode	Year	IncomeGroup	Value
0	Afghanistan	AFG	Population growth (annual %)	SP.POP.GROW	2014	Low income	3.033473
1	Albania	ALB	Population growth (annual %)	SP.POP.GROW	2014	Upper middle income	-0.099830
2	Algeria	DZA	Population growth (annual %)	SP.POP.GROW	2014	Upper middle income	1.940399
3	American Samoa	ASM	Population growth (annual %)	SP.POP.GROW	2014	Upper middle income	0.238405
4	Andorra	ADO	Population growth (annual %)	SP.POP.GROW	2014	High income: nonOECD	-4.191941

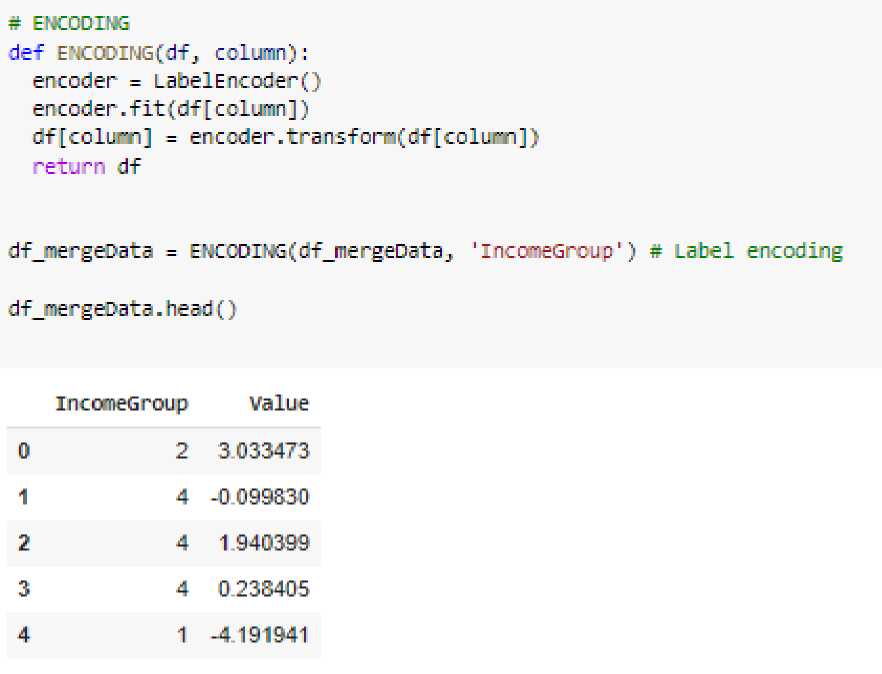
Shape

214 (Number of country) * 7

– Data Preprocessing

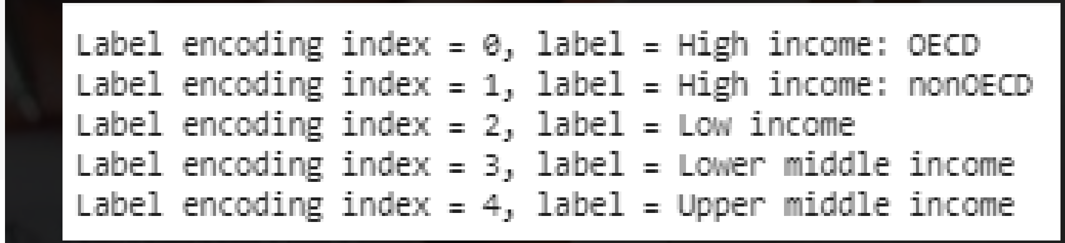
In Clustering

Apply Label Encoding to Categorical data



```python
# ENCODING
def ENCODING(df, column):
  encoder = LabelEncoder()
  encoder.fit(df[column])
  df[column] = encoder.transform(df[column])
  return df


df_mergeData = ENCODING(df_mergeData, 'IncomeGroup') # Label encoding

df_mergeData.head()
```

	IncomeGroup	Value
0	2	3.033473
1	4	-0.099830
2	4	1.940399
3	4	0.238405
4	1	-4.191941



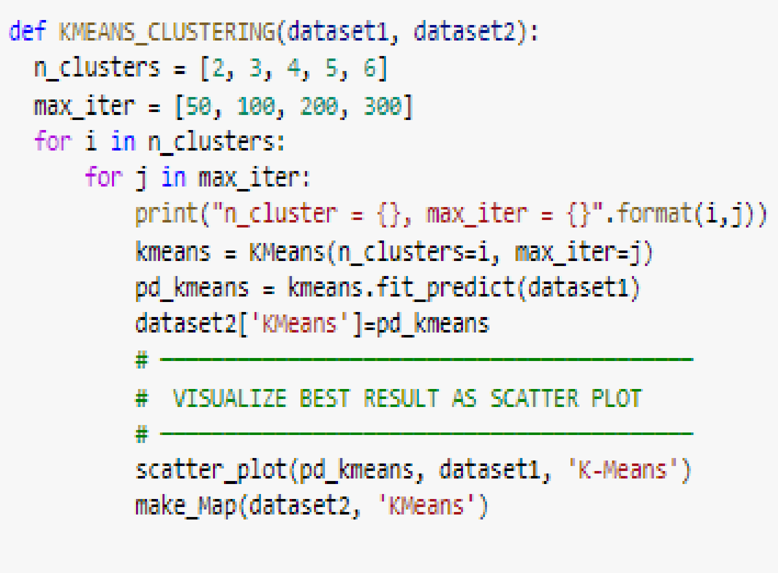
```
Label encoding index = 0, label = High income: OECD
Label encoding index = 1, label = High income: nonOECD
Label encoding index = 2, label = Low income
Label encoding index = 3, label = Lower middle income
Label encoding index = 4, label = Upper middle income
```

– Training

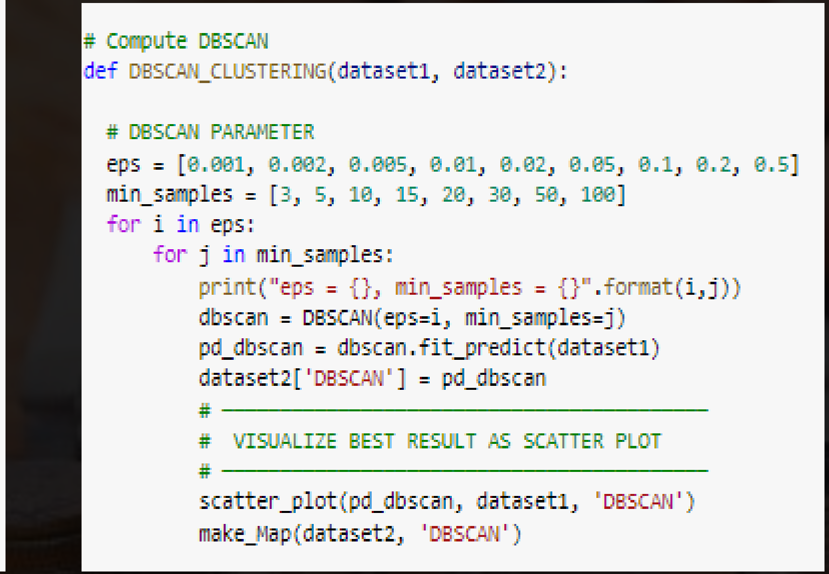
In Clustering

Use 3 Machine Learning Algorithms

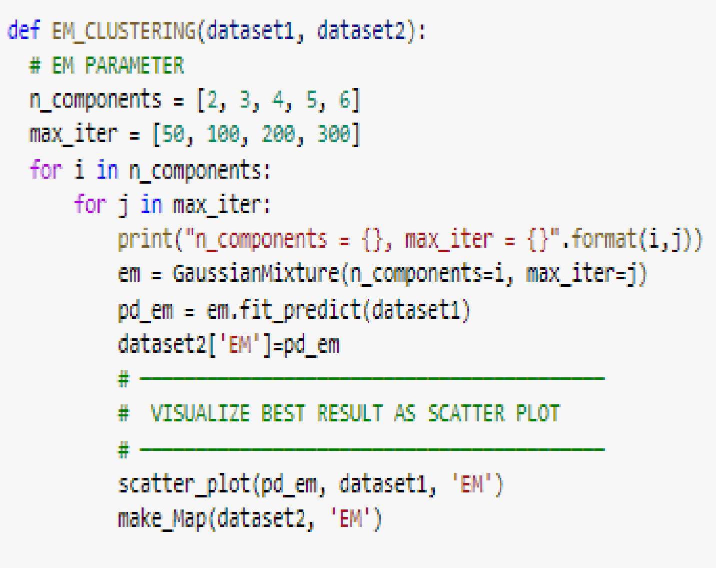
K Means, DBSCAN, EM



```
def KMEANS_CLUSTERING(dataset1, dataset2):
  n_clusters = [2, 3, 4, 5, 6]
  max_iter = [50, 100, 200, 300]
  for i in n_clusters:
      for j in max_iter:
          print("n_cluster = {}, max_iter = {}".format(i,j))
          kmeans = KMeans(n_clusters=i, max_iter=j)
          pd_kmeans = kmeans.fit_predict(dataset1)
          dataset2['KMeans']=pd_kmeans
          # ──────────────────────────────────────────
          #  VISUALIZE BEST RESULT AS SCATTER PLOT
          # ──────────────────────────────────────────
          scatter_plot(pd_kmeans, dataset1, 'K-Means')
          make_Map(dataset2, 'KMeans')
```



```
# Compute DBSCAN
def DBSCAN_CLUSTERING(dataset1, dataset2):

  # DBSCAN PARAMETER
  eps = [0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5]
  min_samples = [3, 5, 10, 15, 20, 30, 50, 100]
  for i in eps:
      for j in min_samples:
          print("eps = {}, min_samples = {}".format(i,j))
          dbscan = DBSCAN(eps=i, min_samples=j)
          pd_dbscan = dbscan.fit_predict(dataset1)
          dataset2['DBSCAN'] = pd_dbscan
          # ──────────────────────────────────────────
          #  VISUALIZE BEST RESULT AS SCATTER PLOT
          # ──────────────────────────────────────────
          scatter_plot(pd_dbscan, dataset1, 'DBSCAN')
          make_Map(dataset2, 'DBSCAN')
```

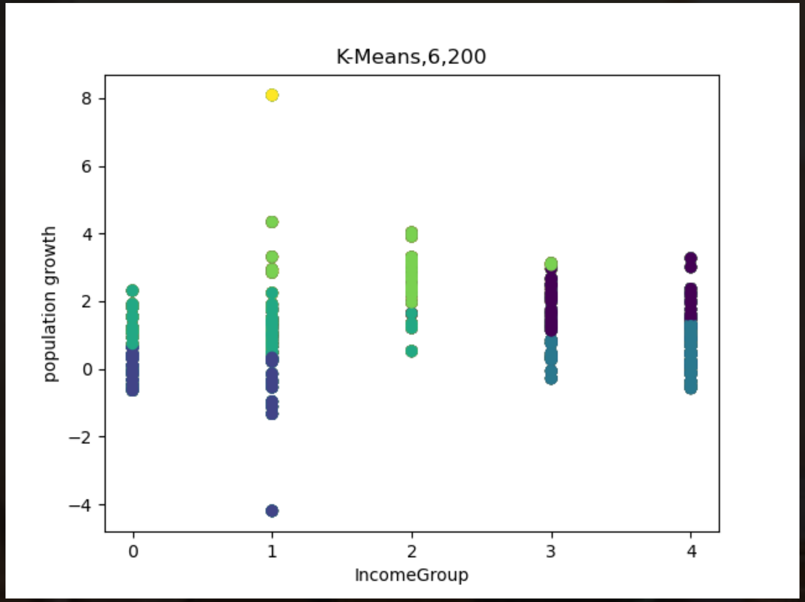


```
def EM_CLUSTERING(dataset1, dataset2):
  # EM PARAMETER
  n_components = [2, 3, 4, 5, 6]
  max_iter = [50, 100, 200, 300]
  for i in n_components:
      for j in max_iter:
          print("n_components = {}, max_iter = {}".format(i,j))
          em = GaussianMixture(n_components=i, max_iter=j)
          pd_em = em.fit_predict(dataset1)
          dataset2['EM']=pd_em
          # ──────────────────────────────────────────
          #  VISUALIZE BEST RESULT AS SCATTER PLOT
          # ──────────────────────────────────────────
          scatter_plot(pd_em, dataset1, 'EM')
          make_Map(dataset2, 'EM')
```

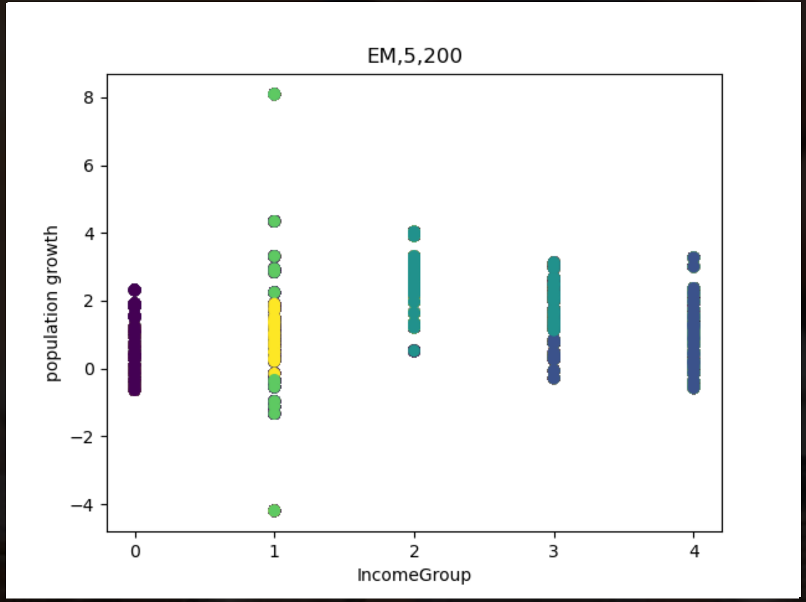
– Result

In Clustering

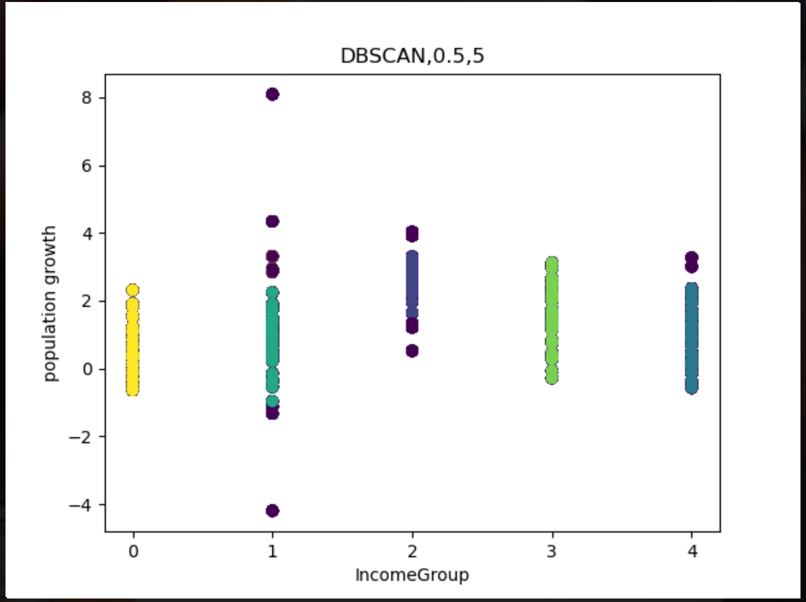
K-Means



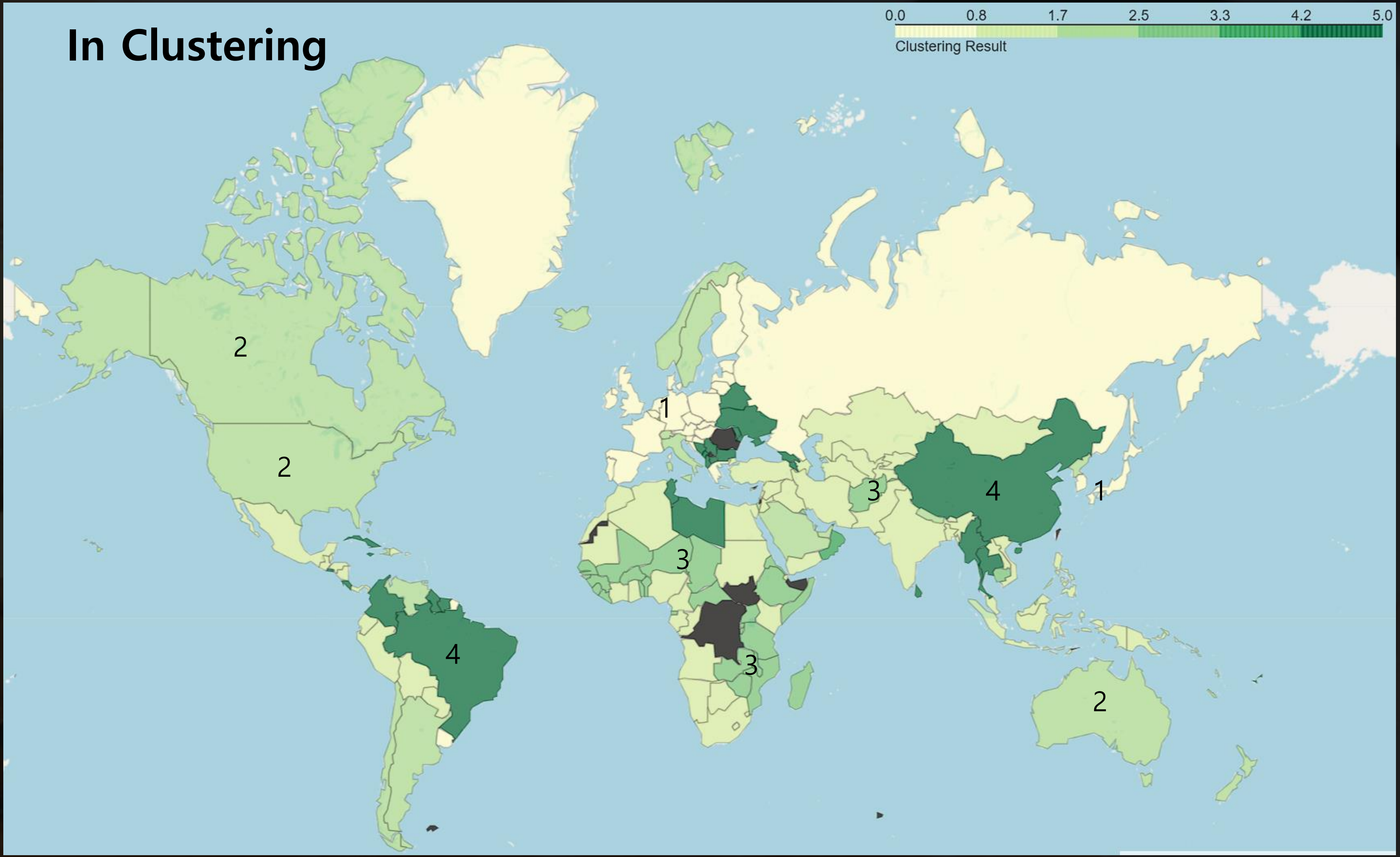
EM



DBSCAN



– Conclusion


In Clustering
0.0
0.8
1.7
2.5
3.3
4.2
5.0
Clustering Result
2
2
1
3
4
1
3
4
3
2

Thank You