Assignment 2

สมาชิกกลุ่ม

1. จิตติณณ์ จินดานรเศรษฐ์ 6110405949 (หัวหน้ากลุ่ม)
2. รินลณี วัชรนิมมานกุล 6110401633
3. กัญญาณัฐ อินทรโชติ 6110402737
4. ณัฐณิชา คงสุนทร 6110402753
5. นิรัติศัย คงศักดิ์ 6110406066
6. พิมพ์ลภัส ตันธนกุล 6110406171
7. สุรยุทธ์ บุญคล้าย 6110406252

ชุดข้อมูล คือ Contraceptive Method Choice ซึ่งเป็นชุดข้อมูลที่ระบุวิธีการคุมกำเนิดที่เลือกใช้ โดยมีผลสำรวจมาจากผู้หญิงประเทศอินโดนีเซียที่แต่งงานในปี ค.ศ. 1987

Download Dataset: https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

Feature คือ คุณลักษณะของข้อมูล ซึ่งแต่ละคอลัมน์หมายถึงดังนี้

- wife_age คือ อายุของภรรยา
- wife_edu คือ ระดับการศึกษาของภรรยา
- husband_edu คือ ระดับการศึกษาของสามี
- children คือ จำนวนเด็กในครอบครัวนั้น ๆ
- wife_islam คือ ภรรยานับถือศาสนาอิสลามหรือไม่
- wife_working คือ ภรรยาทำงานหรือไม่
- husband_job คือ ระยะเวลาของการทำงาน
- living_index คือ ค่าครองชีพ
- media_exposure คือ การเปิดรับสื่อ

Target class คือ คอลัมน์ method โดยที่ method คือ การวางแผนการคุมกำเนิด [No use (ไม่มีการวางแผนการคุมกำเนิด), short-term (การวางแผนการคุมกำเนิดระยะสั้น), long-term (การวางแผนการคุมกำเนิดระยะยาว)]

การออกแบบการทดลอง

ขั้นตอนการออกแบบการทดลองมีดังนี้

1. ทำการเปลี่ยนแปลงชนิดของข้อมูลในตารางให้เหมาะสมกับลักษณะข้อมูล
2. แก้ไขข้อมูลชนิดข้อมูลจากตัวเลขให้เป็นแบบ Category หรือ Boolean
3. เขียนโปรแกรมให้เรียนรู้และทำนายผล

วิธีการ Preprocess

1. ติดตั้ง Anaconda Navigator เพื่อใช้เครื่องมือ Jupyter Notebook ในการออกแบบการทดลอง โดยเรียกใช้ Library ชื่อ Pandas ในการจัดการข้อมูล
2. ทำการ import library ที่ต้องการใช้งานเข้ามา

```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plot
        from sklearn import preprocessing
        from collections import Counter
        from sklearn.preprocessing import MinMaxScaler
```

3. ทำการอ่านไฟล์ cmc. data เข้ามา เนื่องจากชุดข้อมูลไม่มีหัวตารางมาดังนั้นต้องการทำการกำหนด header = none เพื่อป้องแถวของชุดข้อมูลในแถวแรกกลายเป็นชื่อคอมลัมน์

```
In [2]: df = pd.read_csv('cmc.data', header=None)
```

4. ทำการสร้างชื่อคอลัมน์ให้กับชุดข้อมูลดังนี้

```
In [3]: df.columns = ['wife_age', 'wife_edu', 'husband_edu', 'children', 'wife_islam', 'wife_working'
                      ,'husband_job', 'living_index', 'media_exposure', 'method']
```

5. เนื่องจากชนิดของข้อมูลในทุก ๆ คอลัมน์เมื่อทำการ import เข้ามาจะกลายเป็น int64 ทั้งหมด จึงทำการปรับเปลี่ยนชนิดข้อมูลแต่ละคอลัมน์ให้เหมาะสม

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1473 entries, 0 to 1472
Data columns (total 10 columns):
wife_age          1473 non-null int64
wife_edu          1473 non-null int64
husband_edu       1473 non-null int64
children          1473 non-null int64
wife_islam        1473 non-null int64
wife_working      1473 non-null int64
husband_job       1473 non-null int64
living_index      1473 non-null int64
media_exposure    1473 non-null int64
method            1473 non-null int64
dtypes: int64(10)
memory usage: 115.2 KB
```

ภาพด้านล่างนี้คือคอลัมน์ที่จะปรับให้เป็น Category โดยที่จะทำการเปลี่ยนข้อมูลตัวเลขให้เป็นนามบัญญัติ

```
In [6]: df.wife_edu = pd.Categorical(df.wife_edu).rename_categories({1:'low', 2:'mid-low', 3:'mid-high', 4:'high'})
df.husband_edu = pd.Categorical(df.husband_edu).rename_categories({1:'low', 2:'mid-low', 3:'mid-high', 4:'high'})
df.husband_job = pd.Categorical(df.husband_job)
df.living_index =  pd.Categorical(df.living_index).rename_categories({1:'low', 2:'mid-low', 3:'mid-high', 4:'high'})
df.method =  pd.Categorical(df.method).rename_categories({1:'No-use', 2:'Long-term', 3:'Short-term'})
```

ภาพข้างล่างนี้คือคอลัมน์ที่จะปรับให้เป็น Boolean เนื่องจากมีค่าที่เป็นไปได้เพียงสองค่า

```
In [7]: df.wife_islam = df.wife_islam.astype(bool)
df.wife_working = df.wife_working.astype(bool)
df.media_exposure = df.media_exposure.astype(bool)
```

คอลัมน์อายุมีค่าที่หลากหลายจึงทำการปรับให้เป็นช่วง ๆ ดังภาพข้างล่าง

```
In [8]: df.wife_age = pd.cut(df.wife_age, bins=[0,17,31,46,100], right=False,
                             labels=["Children", "Young", "Middle", "Old"])
```

ในคอลัมน์ children จะทำการปรับให้ทุกค่าในคอลัมน์นี้อยู่ในช่วง [0,1] ด้วยการใช้ MinMaxScaler ดังภาพข้างล่าง

```
In [9]: x = df['children'].values.reshape(-1, 1)
        min_max_scaler = MinMaxScaler()
        x_scaled = min_max_scaler.fit_transform(x)
        df['children'] = x_scaled
```

6. ลักษณะของตารางหลังจากการทำ preprocess ตามข้อที่ผ่านมา

	wife_age	wife_edu	husband_edu	children	wife_islam	wife_working	husband_job	living_index	media_exposure	method
0	Young	mid-low	mid-high	0.1875	True	True	2	mid-high	False	No-use
1	Middle	low	mid-high	0.6250	True	True	3	high	False	No-use
2	Middle	mid-low	mid-high	0.4375	True	True	3	high	False	No-use
3	Middle	mid-high	mid-low	0.5625	True	True	3	mid-high	False	No-use
4	Middle	mid-high	mid-high	0.5000	True	True	3	mid-low	False	No-use
...	...	...	...	...	...	...	...	...	...	...
1468	Middle	high	high	0.1250	True	False	2	high	False	Short-term
1469	Middle	high	high	0.1875	True	True	1	high	False	Short-term
1470	Middle	mid-high	mid-high	0.5000	True	False	1	high	False	Short-term
1471	Middle	mid-high	mid-high	0.2500	True	False	2	mid-low	False	Short-term
1472	Young	mid-high	mid-high	0.0625	True	True	2	high	False	Short-term

การเรียนรู้และการทำนายผล

โดยเราจะแบ่ง dataset เป็นสองส่วนคือ train dataset 80% และ test dataset 20% โดยที่ X จะคือ feature และ y คือผลเฉลยดังภาพด้านล่าง

```
In [12]: from sklearn.model_selection import train_test_split
```

```
In [13]: X = pd.get_dummies(df.loc[:, 'wife_age':'media_exposure'])
         y = df.method
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

1. Decision tree classifier

หาค่า hyperparameter ที่เหมาะสมสำหรับ Decision tree โดยใช้ GridSearchCV โดยกำหนด max_depth และ criterion ตามที่ต้องการดังภาพ In[16] ด้านล่าง ต่อมาทำการแสดงค่ากลุ่ม hyperparameter ที่ให้ผลลัพธ์ที่ดีที่สุดออกมาดังภาพ In[18]

```
In [14]: from sklearn.model_selection import GridSearchCV
```

DecisionTreeClassifier

```
In [15]: from sklearn.tree import DecisionTreeClassifier
```

```
In [16]: param_grid = {
          'max_depth' : [2,3,4,5,6,7,8,9,10],
          'criterion' :['gini', 'entropy'],
          'max_features' : ['auto', 'sqrt', 'log2', None]
         }
```

```
In [17]: DTree = DecisionTreeClassifier(random_state=42)
         gscv = GridSearchCV(estimator=DTree, param_grid=param_grid, cv= 3, n_jobs=-1)
         gscv.fit(X_train, y_train)
```

```
Out[17]: GridSearchCV(cv=3, error_score=nan,
                      estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                                       criterion='gini', max_depth=None,
                                                       max_features=None,
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       presort='deprecated',
                                                       random_state=42,
                                                       splitter='best'),
                      iid='deprecated', n_jobs=-1,
                      param_grid={'criterion': ['gini', 'entropy'],
                                  'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10],
                                  'max_features': ['auto', 'sqrt', 'log2', None]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)
```

```
In [18]: gscv.best_params_
```

```
Out[18]: {'criterion': 'gini', 'max_depth': 5, 'max_features': None}
```

โดยเมื่อปรับค่า hyperparameter ตาม GridSearchCV จะให้ค่าประสิทธิภาพคือ 0.5796610169491525

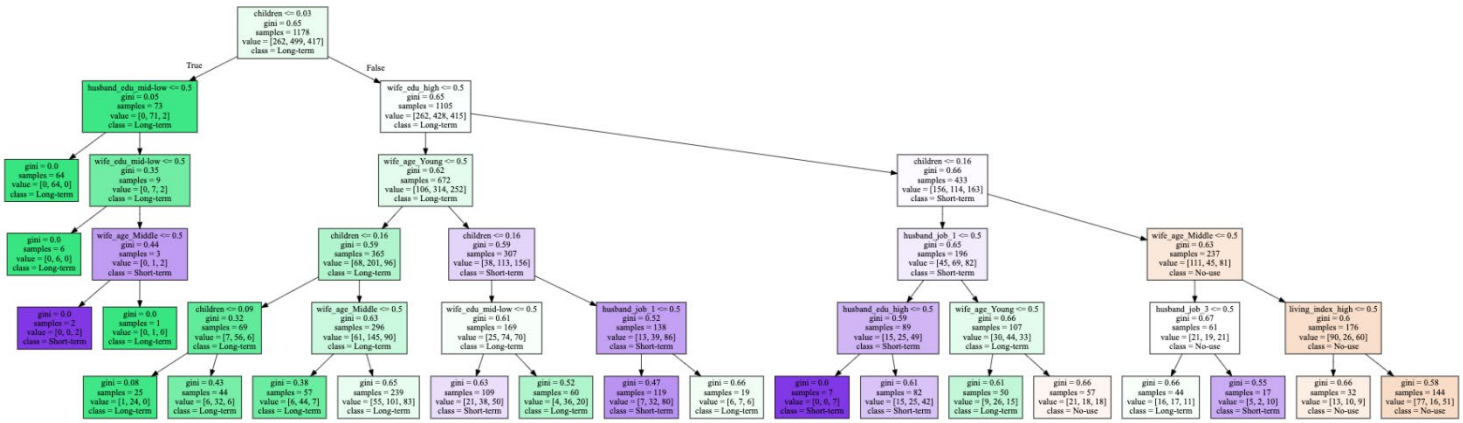
```
In [19]: DTree = DecisionTreeClassifier(random_state=42, max_depth=5, criterion='gini')
         DTree.fit(X_train, y_train)
         DTree.score(X_test,y_test)
```

```
Out[19]: 0.5796610169491525
```

ภาพด้านล่างคือลักษณะของ Decision tree จากการทดลอง

```
In [20]: from graphviz import Source
         from sklearn.tree import export_graphviz
```

```
In [21]: Source(export_graphviz(DTree, out_file=None,
                                feature_names=X.columns, class_names=df.method.unique(),
                                filled=True, precision=2))
```



2. K-nearest neighbor classifier

หาค่า hyperparameter ที่เหมาะสมสำหรับ KNN โดยใช้ GridSearchCV โดยกำหนด n_neighbors, weights และ metric ตามที่ต้องการดังภาพ In[23] ด้านล่าง ต่อมาทำการแสดงกลุ่ม hyperparameter ที่ให้ผลลัพธ์ที่ดีที่สุดออกมาดังภาพ In[25]

In [22]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [23]:
```
param_grid = {
    'n_neighbors' : [1,2,3,4,5,6,7,8,9,10,11,12],
    'weights' : ['uniform', 'distance'],
    'metric' : ['euclidean', 'manhattan']
}
```

In [24]:
```
knn = KNeighborsClassifier()
gscv = GridSearchCV(estimator=knn, param_grid=param_grid, cv= 3, n_jobs=-1)
gscv.fit(X_train, y_train)
```

Out[24]:
```
GridSearchCV(cv=3, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=-1,
             param_grid={'metric': ['euclidean', 'manhattan'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
                         'weights': ['uniform', 'distance']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [25]:
```
gscv.best_params_
```

Out[25]:
```
{'metric': 'euclidean', 'n_neighbors': 11, 'weights': 'uniform'}
```

โดยเมื่อปรับค่า hyperparameter ตาม GridSearchCV จะให้ค่าประสิทธิภาพคือ 0.49491525423728816

In [26]:
```
knn = KNeighborsClassifier(n_neighbors=11, metric='euclidean', weights='uniform')
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[26]:
```
0.49491525423728816
```

3. MLP classifier

หาค่า hyperparameter ที่เหมาะสมสำหรับ MLP โดยใช้ GridSearchCV โดยกำหนด param_grid ตามที่ต้องการดังภาพ In[28] ด้านล่าง ต่อมาทำการแสดงกลุ่ม hyperparameter ที่ให้ผลลัพธ์ที่ดีที่สุดออกมาดังภาพ In[30]

In [27]:
```
from sklearn.neural_network import MLPClassifier
```

In [28]:
```
param_grid = {
    'hidden_layer_sizes': [(10,30,10),(20,)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
```

In [29]:
```
mlp = MLPClassifier()
gscv = GridSearchCV(estimator=mlp, param_grid=param_grid, cv= 3, n_jobs=-1)
gscv.fit(X_train, y_train)
```

```
C:\Users\tinti\Anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_per
tions (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

Out[29]:
```
GridSearchCV(cv=3, error_score=nan,
             estimator=MLPClassifier(activation='relu', alpha=0.0001,
                                     batch_size='auto', beta_1=0.9,
                                     beta_2=0.999, early_stopping=False,
                                     epsilon=1e-08, hidden_layer_sizes=(100,),
                                     learning_rate='constant',
                                     learning_rate_init=0.001, max_fun=15000,
                                     max_iter=200, momentum=0.9,
                                     n_iter_no_change=10,
                                     nesterovs_momentum=True, power_t=0.5,
                                     random_state...
                                     solver='adam', tol=0.0001,
                                     validation_fraction=0.1, verbose=False,
                                     warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'activation': ['tanh', 'relu'],
                         'alpha': [0.0001, 0.05],
                         'hidden_layer_sizes': [(10, 30, 10), (20,)],
                         'learning_rate': ['constant', 'adaptive'],
                         'solver': ['sgd', 'adam']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [30]:
```
gscv.best_params_
```

Out[30]:
```
{'activation': 'tanh',
 'alpha': 0.0001,
 'hidden_layer_sizes': (10, 30, 10),
 'learning_rate': 'constant',
 'solver': 'adam'}
```

โดยเมื่อปรับค่า hyperparameter ตาม GridSearchCV จะให้ค่าประสิทธิภาพคือ 0.5661016949152542

```
In [31]: mlp = MLPClassifier(activation='tanh', alpha=0.0001, hidden_layer_sizes=(10, 30, 10), learning_rate='adaptive', solver='adam')
         mlp.fit(X_train, y_train)
         mlp.score(X_test,y_test)

         C:\Users\tinti\Anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:571: ConvergenceWarning: Stocha
         tions (200) reached and the optimization hasn't converged yet.
           % self.max_iter, ConvergenceWarning)

Out[31]: 0.5661016949152542
```

สรุปผลการทดลอง

จากผลการทดลองการทำ preprocess ข้อมูลและวัดค่าประสิทธิภาพการเรียนรู้ในแต่ละอัลกอริทึม โดยการใช้ GridSearchCV เพื่อทำการหาค่าและปรับ hyperparameter ที่เหมาะสมของแต่ละอัลกอริทึม

- Decision tree classifier วัดค่าประสิทธิภาพการเรียนรู้ได้ 0.5796610169491525
- KNN classifier วัดค่าประสิทธิภาพการเรียนรู้ได้ 0.49491525423728816
- MLP classifier วัดค่าประสิทธิภาพการ เรียนรู้ได้ 0.5661016949152542

ดังนั้นสรุปได้ว่าอัลกอริทึมในแต่ละแบบนั้นอัลกอริทึมที่ให้ค่าผลลัพธ์ ค่าประสิทธิภาพที่ดีที่สุดคือ Decision tree classifier และจากการผลการทดลองค่าประสิทธิภาพของแต่ละอัลกอริทึมนั้นให้ผลลัพธ์ค่าประสิทธิภาพการเรียนรู้ที่ค่อนข้างต่ำ โดยอ้างอิงจากเอกสารที่เกี่ยวข้อง มีเพียงไม่กี่อัลกอริทึมที่สามารถใช้งานได้ แต่ก็ยังคงมีค่า Error rates ค่อนข้างสูงคือ ต่ำสุดที่ 0.43 และสูงสุดที่ 0.60

Table 4. Minimum, maximum, and 'naive' plurality rule error rates for each dataset. A '√'-mark indicates that the algorithm has an error rate within one standard error of the minimum for the dataset. A 'X'-mark indicates that the algorithm has the worst error rate for the dataset. The mean error rate for each algorithm is given in the second row.

	Decision trees and rules																					Statistical algorithms									Nets		Error rates		
	QU0	QU1	QL0	QL1	FTU	FTL	C4T	C4R	IB	IBO	IM	IMO	IC0	IC1	OCU	OCL	OCM	ST0	ST1	LMT	CAL	T1	LDA	QDA	NN	LOG	FM1	FM2	PDA	MDA	POL	LVQ	RBF	Min	Max	Naive
Mean	.221	.226	.208	.211	.238	.234	.220	.220	.229	.219	.220	.219	.215	.227	.227	.260	.230	.232	.233	.220	.270	.354	.208	.301	.281	.204	.242	.217	.213	.207	.195	.269	.257			
#√	8	8	10	9	4	12	7	8	3	8	5	6	4	5	9	4	4	5	5	1	4	4	10	4	4	13	12	12	10	9	15	4	8			
#X	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	1	0	1	0	0	11	0	3	4	0	2	1	0	1	0	4	0			
cmc	√	√																							X						√			.43	.60	.57
cmc+	√	√																							X						√			.43	.58	

Lim, T.-S., Loh, W.-Y. & Shih, Y.-S. (1999). A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms. Machine Learning.