Gachon Data Mining School
Nursery Data Set

Team C

2016.08.16

목차

01 Concept Structure

02 Code - Feature Engineering

03 Code - Decision Tree & 성능 측정

Concept Structure ①
- Nursery: Evaluation of Applications for nursery schools

1. EMPLOY : Employment of parents and child's nursery

   ① Parents : Parents' occupation

   ② Has_nurs: Child's nursery

2. STRUCT_FINAN: Family structure and financial standings

   ① STRUCTURE: Family structure

      I. form: Form of the family

      II. children: Number of children

   ② housing: Housing conditions

   ③ finance: Financial standing of the family

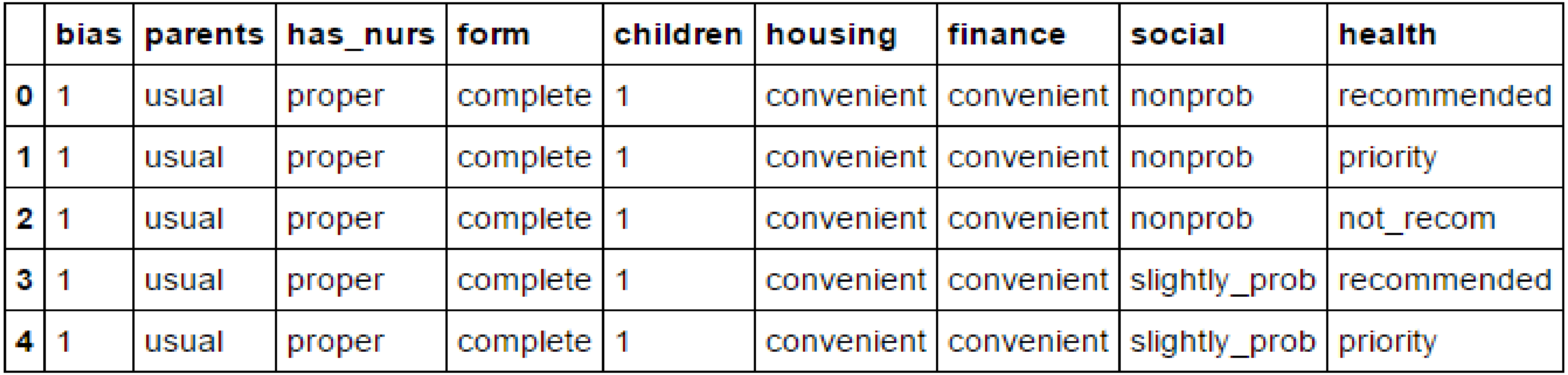
3. SOC_HEALTH Social and health picture of the family

   ① social: Social conditions

   ② health: Health conditions

Concept Structure ②

- Nursery: Evaluation of Applications for nursery schools



	bias	parents	has_nurs	form	children	housing	finance	social	health
0	1	usual	proper	complete	1	convenient	convenient	nonprob	recommended
1	1	usual	proper	complete	1	convenient	convenient	nonprob	priority
2	1	usual	proper	complete	1	convenient	convenient	nonprob	not_recom
3	1	usual	proper	complete	1	convenient	convenient	slightly_prob	recommended
4	1	usual	proper	complete	1	convenient	convenient	slightly_prob	priority

housing : convenient, less_conv, critical

finance : convenient, inconv

social : non-prob, slightly_prob, problematic

health : recommended, priority, not_recom

parents : usual, pretentious, great_pret

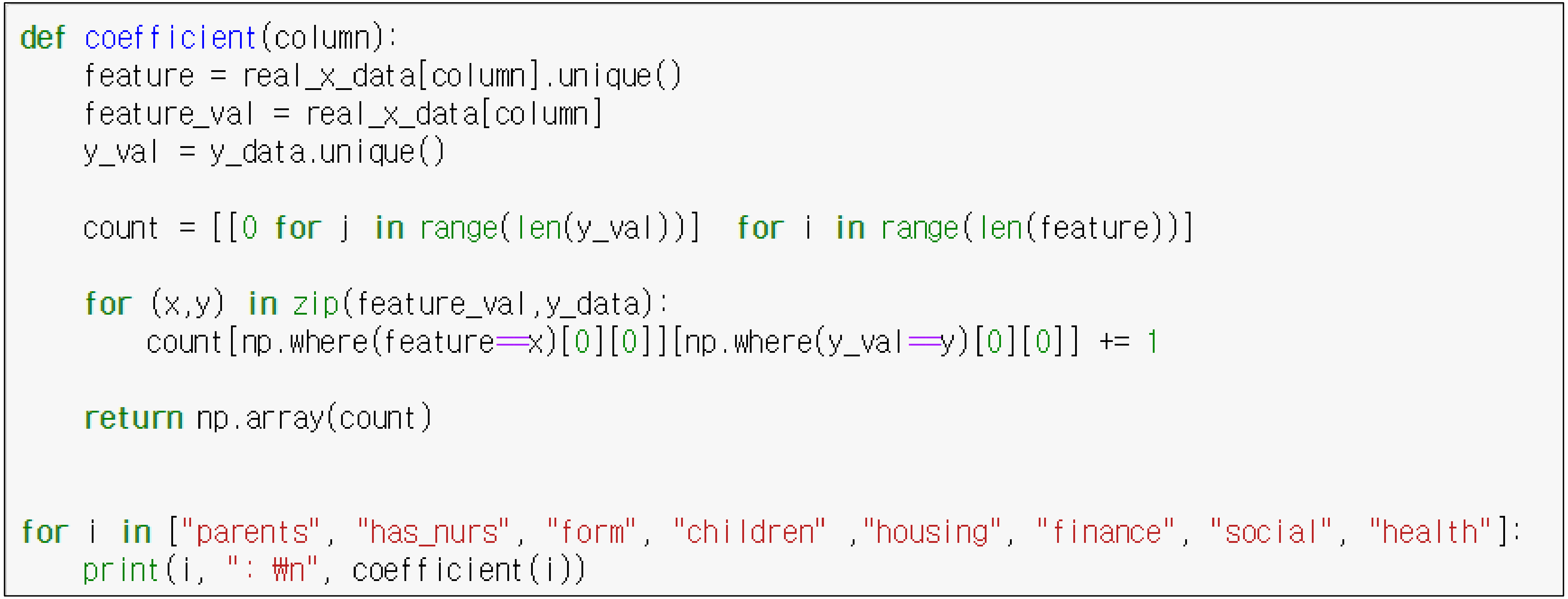
has_nurs : less_proper, improper,critical very_crit, proper

form : complete, completed, incomplete, foster

children : 1, 2, 3, more

Code - Feature Engineering

종속변수 Y의 값과 Category Type인 Feature의 분포를 알아보는 함수를 정의한다.



```
def coefficient(column):
    feature = real_x_data[column].unique()
    feature_val = real_x_data[column]
    y_val = y_data.unique()

    count = [[0 for j in range(len(y_val))]  for i in range(len(feature))]

    for (x,y) in zip(feature_val,y_data):
        count[np.where(feature==x)[0][0]][np.where(y_val==y)[0][0]] += 1

    return np.array(count)


for i in ["parents", "has_nurs", "form", "children" ,"housing", "finance", "social", "health"]:
    print(i, ": ₩n", coefficient(i))
```

Code - Feature Engineering

has_nurs

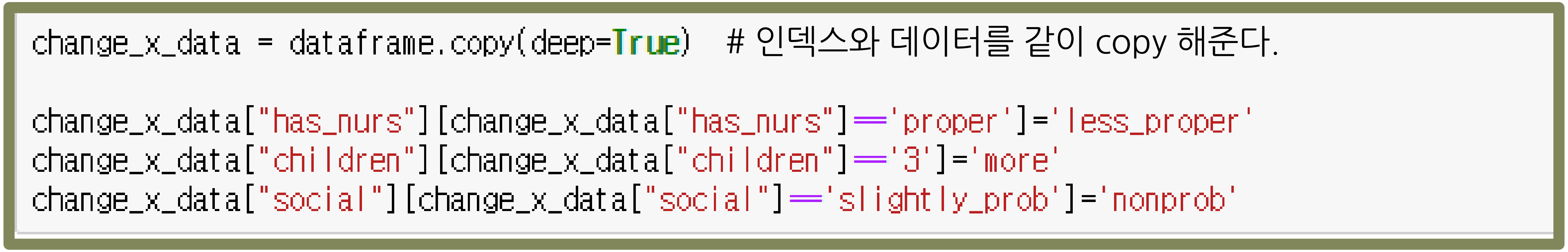
	Y1	Y2	Y3	Y4	Y5
X1	2	1344	864	130	252
X2	0	1344	864	132	252
X3	0	904	864	66	758
X4	0	464	864	0	1264
X5	0	210	864	0	1518

children

	Y1	Y2	Y3	Y4	Y5
X1	2	1206	1080	148	804
X2	0	1092	1080	100	968
X3	0	984	1080	40	1136
X4	0	984	1080	40	1136

social

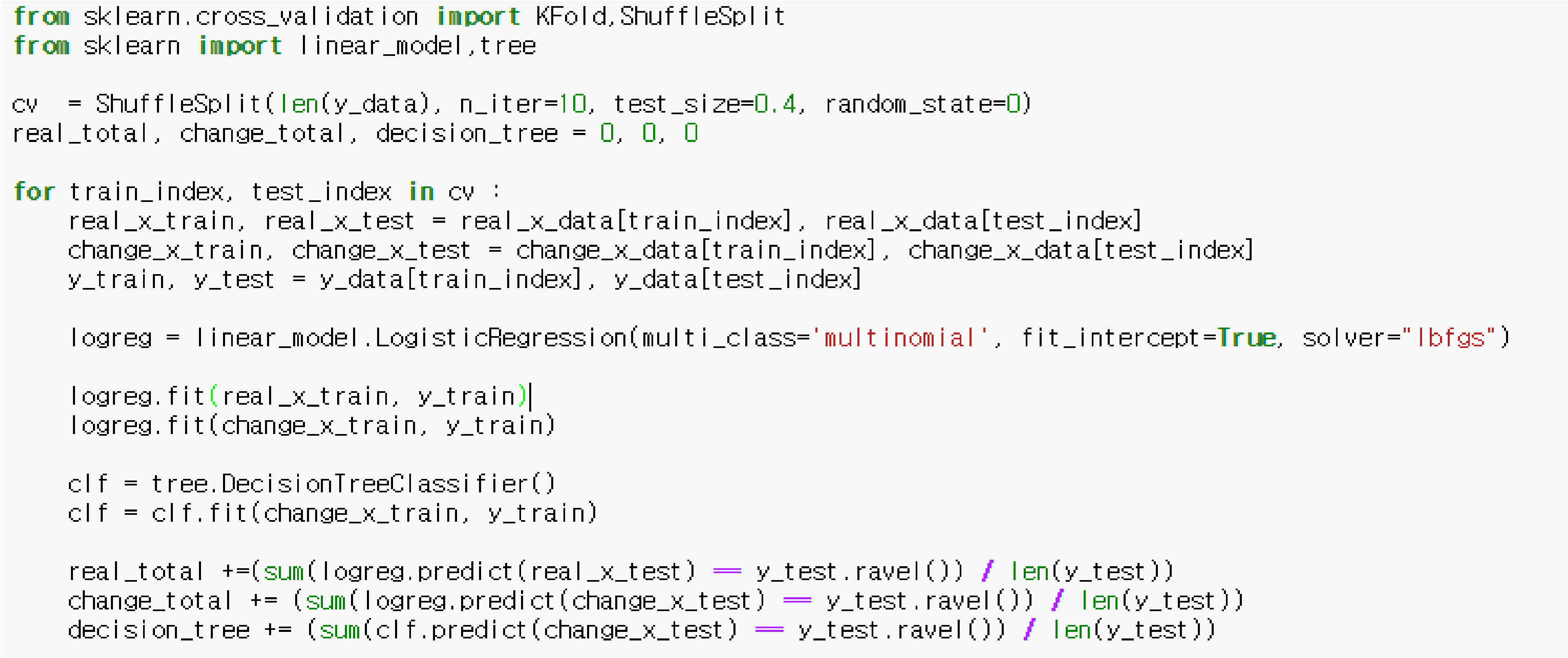
	Y1	Y2	Y3	Y4	Y5
X1	1	1515	1440	164	1200
X2	1	1515	1440	164	1200
X3	0	1236	1440	0	1644



```
change_x_data = dataframe.copy(deep=True)   # 인덱스와 데이터를 같이 copy 해준다.

change_x_data["has_nurs"][change_x_data["has_nurs"]=='proper']='less_proper'
change_x_data["children"][change_x_data["children"]=='3']='more'
change_x_data["social"][change_x_data["social"]=='slightly_prob']='nonprob'
```

Code - Decision Tree & 성능 측정



```python
from sklearn.cross_validation import KFold,ShuffleSplit
from sklearn import linear_model,tree

cv  = ShuffleSplit(len(y_data), n_iter=10, test_size=0.4, random_state=0)
real_total, change_total, decision_tree = 0, 0, 0

for train_index, test_index in cv :
    real_x_train, real_x_test = real_x_data[train_index], real_x_data[test_index]
    change_x_train, change_x_test = change_x_data[train_index], change_x_data[test_index]
    y_train, y_test = y_data[train_index], y_data[test_index]

    logreg = linear_model.LogisticRegression(multi_class='multinomial', fit_intercept=True, solver="lbfgs")

    logreg.fit(real_x_train, y_train)
    logreg.fit(change_x_train, y_train)

    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(change_x_train, y_train)

    real_total +=(sum(logreg.predict(real_x_test) == y_test.ravel()) / len(y_test))
    change_total += (sum(logreg.predict(change_x_test) == y_test.ravel()) / len(y_test))
    decision_tree += (sum(clf.predict(change_x_test) == y_test.ravel()) / len(y_test))
```

① 기존 Logistic Regression 10번 실행 후 평균값 → 약 86%

② Feature engineering 실행 → 약 92% ③ Decision Tree 실행 → 약 99%

감사합니다.