Kaggle
Competition

김재민
석은희

목차

01
Feature

```
# panel조사 분류별 응답률 피처 생성
SQ = ['SQ1', 'SQ2', 'SQ3', 'SQ4', 'SQ5', 'SQ6', 'SQ7', 'SQ8'] # 개인정보
A = ['A1'] # 신규구입/렌트 가전
B = ['B1', 'B2', 'B3', 'B4', 'B5'] # 통신
C = ['C1', 'C2', 'C3'] # 보험/금융
D = ['DQ1', 'DQ2', 'DQ3', 'DQ4', 'DQ5', 'DQ6', 'DQ7'] # 직업
F = ['F1', 'F2'] # 이용마트
H = ['H1'] # 음용주류
T = ['T1'] # 당배
X = ['X1', 'X2', 'X3', 'X4'] # 자동차

panel['SQ_R'] = 1 - panel[SQ].isnull().mean(axis=1)
panel['A_R'] = 1 - panel[A].isnull().mean(axis=1)
panel['B_R'] = 1 - panel[B].isnull().mean(axis=1)
panel['C_R'] = 1 - panel[C].isnull().mean(axis=1)
panel['D_R'] = 1 - panel[D].isnull().mean(axis=1)
panel['F_R'] = 1 - panel[F].isnull().mean(axis=1)
panel['H_R'] = 1 - panel[H].isnull().mean(axis=1)
panel['T_R'] = 1 - panel[T].isnull().mean(axis=1)
panel['X_R'] = 1 - panel[X].isnull().mean(axis=1)
# panel 조사 전체 응답률
panel['ALL_R'] = 1 - panel[SQ+A+B+C+D+F+H+T+X].isnull().mean(axis=1)
```

- 응답률

01
Feature

```
train['TIME_hour'] = train.TIME.dt.hour
train['TIME_min'] = train.TIME.dt.minute
train['dayofweek'] = train.TIME.dt.dayofweek

test['TIME_hour'] = test.TIME.dt.hour
test['TIME_min'] = test.TIME.dt.minute
test['dayofweek'] = test.TIME.dt.dayofweek
```

- 시간
- 분
- 요일

02
Data Cleansing

```
for i in train.columns:
    print(i, '₩t', train[i].nunique(), '₩t', train[i].isnull().sum()/train.shape[0])
```

45개의 칼럼 중 24개만 결측값 비율이 30%이하

02
Data Cleansing

XGBOOST

```
for i in na_0:
    X_train[i]=X_train[i].fillna(0)
```

```
for i in na_0:
    X_test[i]=X_test[i].fillna(0)
```

```
from sklearn.impute import SimpleImputer

if len(num_features) > 0:
    imp = SimpleImputer(strategy='mean')
    X_train[num_features] = imp.fit_transform(X_train[num_features])
    X_test[num_features] = imp.transform(X_test[num_features])
if len(cat_features) > 0:
    imp = SimpleImputer(strategy="most_frequent")
    X_train[cat_features] = imp.fit_transform(X_train[cat_features])
    X_test[cat_features] = imp.transform(X_test[cat_features])
```

- 결측값이 많은 칼럼(결측 비율 30%이상)은 결측값을 0으로 채움

- 나머지 칼럼은 Simpleputer사용

02

Data Cleansing

XGBOOST

나눠서 결측값을 처리한 이유

- 개인 정보를 포함한 설문이여서 민감한 내용이 많아 결측값이 많았음

- 미응답도 의미가 있을 것이라 생각해 0이라는 새로운 값으로 결측값을 처리해 성능 향상에 기여

02
Data Cleansing

DNN

```
def handle_profile_SQ4(x):
    if x in ['1', '2', '3', '4', '5', '6']:
        return int(x)
    elif x in ['1,', '2,', '3,', '4,', '5,', '6,']:
        return int(x[0])
    else:
        return 99

train.SQ4 = train.SQ4.apply(handle_profile_SQ4)
test.SQ4 = test.SQ4.apply(handle_profile_SQ4)
```

```
def handle_profile_SQ5(x):
    if x in ['1', '2', '3', '4']:
        return int(x)
    elif x in ['1.0', '2.0', '3.0', '4.0']:
        return int(x[0])
    else:
        return 99

train.SQ5 = train.SQ5.apply(handle_profile_SQ5)
test.SQ5 = test.SQ5.apply(handle_profile_SQ5)
```

- 결측값이 많은 칼럼(결측 비율 30%이상)은 사용하지 않음

- 나머지 칼럼은 함수를 정의해 처리

03
Scaling, Encoding

XGBOOST

```
for i in cat_features:
    features[i] = LabelEncoder().fit_transform(features[i])
```

- LabelEncoder

DNN

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train[num_features] = scaler.fit_transform(X_train[num_features])
X_test[num_features] = scaler.transform(X_test[num_features])
```

```
for i in cat_features:
    ohe = OneHotEncoder(sparse=False)
    a = ohe.fit_transform(X_train[[i]])
    b = ohe.transform(X_test[[i]])
    c = pd.DataFrame(a, columns=[col for col in ohe.categories_[0]])
    d = pd.DataFrame(b, columns=[col for col in ohe.categories_[0]])
    X_train = pd.concat([X_train.drop(columns=[i]), c], axis=1)
    X_test = pd.concat([X_test.drop(columns=[i]), d], axis=1)
```

- StamdardScaler, OneHotEncoder

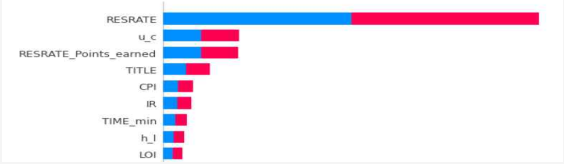
04
Feature Selection

```python
import shap
# DF, based on which importance is checked
X_importance = X_test

# Explain model predictions using shap library:
model = LGBMClassifier(random_state=0).fit(X_train, y_train)
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_importance)

# Plot summary_plot as barplot:
shap.summary_plot(shap_values, X_importance, plot_type='bar')

shap_sum = np.abs(shap_values).mean(axis=1)[1,:]
importance_df = pd.DataFrame([X_importance.columns.tolist(), shap_sum.tolist()]).T
importance_df.columns = ['column_name', 'shap_importance']
importance_df = importance_df.sort_values('shap_importance', ascending=False)
importance_df
```



- shap 사용

- 중요도가 0 보다 큰 feature만 사용

05
Model Tuning

XGBOOST

```
def objective_xgbm(trial, X, y):

    param_bo = {
        'n_estimators': trial.suggest_int("n_estimators", 80, 350),
        'learning_rate': trial.suggest_float("learning_rate", 0.01, 0.3, step=0.05),
        'max_depth': trial.suggest_int("max_depth", 6, 12),
        'subsample': trial.suggest_float("subsample", 0.6, 1, step=0.1),
        'colsample': trial.suggest_float("colsample", 0.6, 1,, step=0.1),
    }

    model = XGBClassifier(**param_bo,tree_method='hist', random_state=0)
    score = model_selection.cross_val_score(model, X, y, cv=5, n_jobs=-1, scoring='roc_auc')
    return score.mean()

study_xgbm = optuna.create_study(direction="maximize")
study_xgbm.optimize(lambda trial: objective_xgbm(trial, X_train, y_train), n_trials=72)
```

optuna 사용

05
Model Tuning

DNN

```
def model_fn(hp):
    inputs = keras.Input(shape=(X_train.shape[1],))
    x = keras.layers.Dense(hp.Int('unit', 16, 32, step=16), hp.Choice('activation', ['relu', 'elu']))(inputs)
    x = keras.layers.Dropout(hp.Float('dropout', 0, 0.25, step=0.25, default=0.25))(x)
    outputs = keras.layers.Dense(1, activation='sigmoid')(x)
    model = keras.Model(inputs, outputs)
    model.compile(loss='binary_crossentropy',
                  optimizer=tf.keras.optimizers.Adam(hp.Choice('learning_rate', [1e-2, 1e-3])),
                  metrics=[keras.metrics.AUC()])
    return model
```

```
import kerastuner as kt
tuner = kt.Hyperband(model_fn,
                     objective=kt.Objective('val_auc', direction="max"),
                     max_epochs=4,
                     hyperband_iterations=2,
                     overwrite=True,
                     directory='dnn_tuning')

tuner.search(X_train, y_train, validation_data=(X_valid, y_valid),
             callbacks=[tf.keras.callbacks.EarlyStopping(patience=1)])
```

kerastuner 사용

06
Ensemble

```
sub.STATUS = 0.5 * xgb.STATUS + 0.5 * dnn.STATUS
```

산술 평균을 사용한 앙상블

감사합니다