Home Credit Default Risk

Can you predict how capable each applicant is of repaying a loan?

Last Updated: 4 years ago

2주차

Introduction to Feature Selection

최하경

목차 소개

0. Introduction

1. 다중공선성

2. 결측값 비율

3. 피처 중요도

0. Introduction

1. 이전 커널들에서 피처엔지니어링 진행한 데이터 이용
   - [Part1] https://www.kaggle.com/willkoehrsen/introduction-to-manual-feature-engineering
   - [Part2] https://www.kaggle.com/willkoehrsen/introduction-to-manual-feature-engineering-p2

2. Feature engineering에서 만든 피처들 확인
   - bureau, previous에 각각 혹은 둘 다 있는 칼럼들 확인 → set() 이용
   - bureau / previous / orignial_features란 이름으로 칼럼들 각각 저장

3. 중복되는 row들 없이 데이터프레임들 합쳐 train, test data 생성

0. Introduction

4. one-hot encoding 실행한 후 train, test data 칼럼들을 열에 대해서 정렬 → align() 이용

   → train, test data 간 칼럼들 동일한지 알 수 있음

```
# One hot encoding
train = pd.get_dummies(train)
test = pd.get_dummies(test)

# Match the columns in the dataframes
train, test = train.align(test, join = 'inner', axis = 1)
```

axis=1 열에 대해 정렬

join='inner' 인덱스를 교집합으로 출력

5. 모델링에 필요없는 피처들 제거

```
Training shape:  (1000, 1398)
Testing shape:  (1000, 1398)
```

→ 전체 데이터셋에 적용했을 때 1416개의 칼럼들

1. 다중공선성

Pearson correlation coefficient가 0.9 이상인 피처들 제거

```
# Threshold for removing correlated variables
threshold = 0.9

# Absolute value correlation matrix
corr_matrix = train.corr().abs()
corr_matrix.head()
```

```
# Upper triangle of correlations
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Select columns with correlations above threshold
to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
```

→ 584개 제거 후 814개의 칼럼들 남음

[np.triu] np.triu(m,k)

m : 행렬

k : 어디서부터 0 처리할건지

k>0 해당 대각선 위부터 0 처리

k=0 대각선 위부터 0 처리

k<0 해당 대각선 밑부터 0 처리

```
Sample matrix
[[1 2 3]
 [4 5 6]
 [7 8 9]]

triu() function without any parameter:
[[1 2 3]
 [0 5 6]
 [0 0 9]]
```

https://www.w3resource.com/numpy/array-creation/triu.php

2. 결측값 비율

결측치가 75% 이상인 칼럼들 제거

```
# Train missing values (in percent)
train_missing = (train.isnull().sum() / len(train)).sort_values(ascending = False)

# Test missing values (in percent)
test_missing = (test.isnull().sum() / len(test)).sort_values(ascending = False)


# Identify missing values above threshold
train_missing = train_missing.index[train_missing > 0.75]
test_missing = test_missing.index[test_missing > 0.75]

all_missing = list(set(set(train_missing) | set(test_missing)))
print('There are %d columns with more than 75%% missing values' % len(all_missing))
```

train, test data 둘 다에서 75% 이상의 결측값을 가진 칼럼들 합집합 해 모두 제거

→ 17개 칼럼들 제거

2. 결측값 비율

이후 one-hot encoding 실행한 후 align 진행

```
train = pd.get_dummies(train.drop(columns = all_missing))
test = pd.get_dummies(test.drop(columns = all_missing))

train, test = train.align(test, join = 'inner', axis = 1)
```

→ 총 845개의 칼럼들 존재

3. 피처 중요도

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html
http://machinelearningkorea.com/2019/09/25/lightgbm%EC%9D%98-%ED%95%B5%EC%8B%AC%EC%9D%B4%ED%95%B4/

LightGBM library를 이용해 피처 중요도가 0인 피처들 제거 → 피처 중요도와 중요도가 0인 피처들 출력

```
def identify_zero_importance_features(train, train_labels, iterations = 2):

    # Initialize an empty array to hold feature importances
    feature_importances = np.zeros(train.shape[1])

    # Create the model with several hyperparameters
    model = lgb.LGBMClassifier(objective='binary', boosting_type = 'goss',
                               n_estimators = 10000, class_weight = 'balanced')

    # Fit the model multiple times to avoid overfitting
    for i in range(iterations):

        # Split into training and validation set
        train_features, valid_features, train_y, valid_y = train_test_split(train,

        # Train using early stopping
        model.fit(train_features, train_y, early_stopping_rounds=100,
                  eval_set = [(valid_features, valid_y)],
                  eval_metric = 'auc', verbose = 200)

        # Record the feature importances
        feature_importances += model.feature_importances_ / iterations
```

boosting_type='goss'
Gradient-based One-side sampling
기울기가 작은 데이터들은 랜덤하게 제거

class_weight='balanced'
multi-class classification에서만 사용
n_samples / (n_classes * np.bincount(y))

iterations=2 오버피팅을 피하기 위해 모델 피팅 반복

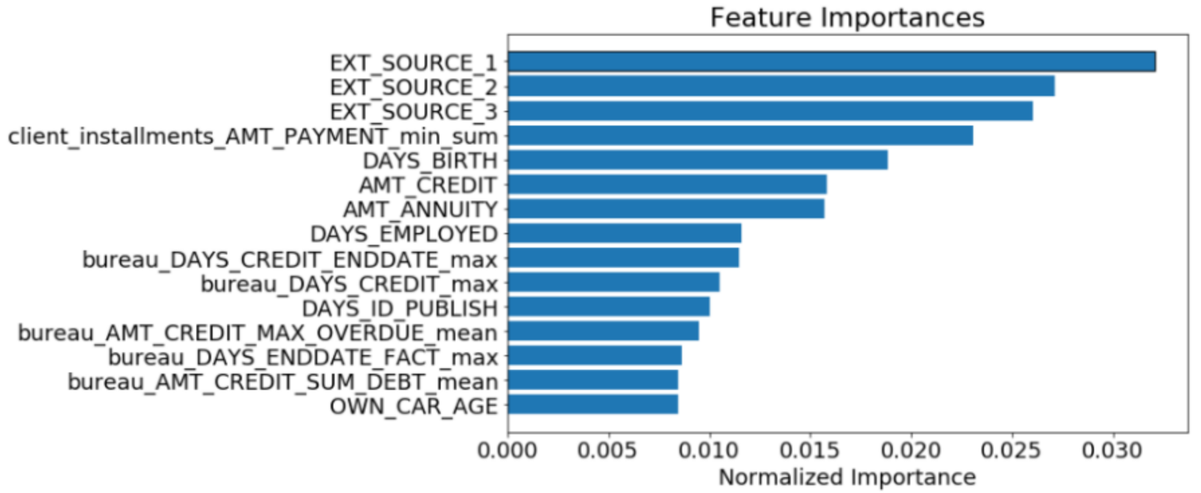
early_stopping_rounds=100
100번 이후 auc 높아지지 않으면 실행 중지

→ 총 271개의 칼럼들 제거 후 573개 남음

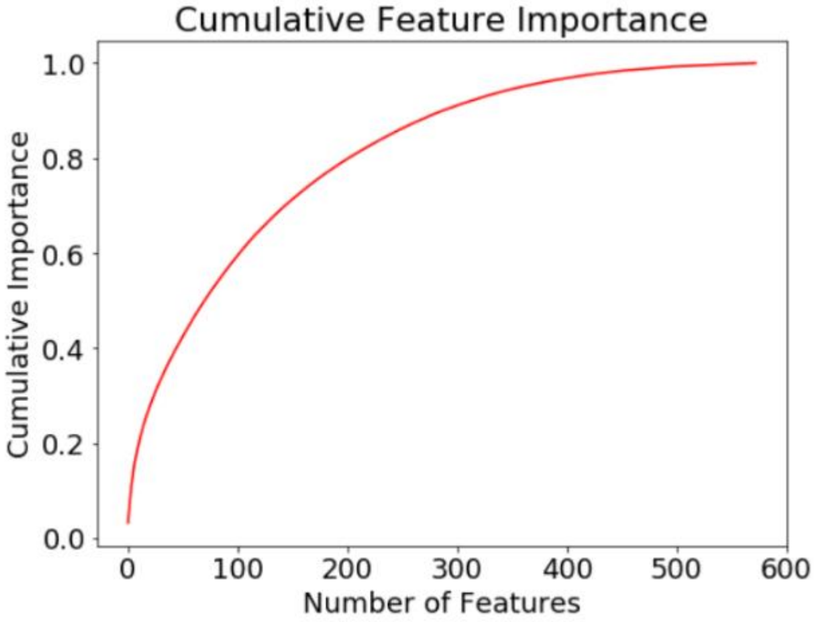
3. 피처 중요도

피처 중요도 결과를 plotting한 결과

1. 피처 중요도가 가장 높은 top 15개 칼럼들



2. 누적 중요도가 95% 되게 하는 피처들 개수 출력




```
360 features required for 0.95 of cumulative importance
```

4. Conclusion

이 노트북에서 진행한 feature selection 과정

1. 다중공선성이 0.9 이상인 변수들 제거
2. 결측치가 0.75% 이상인 변수들 제거
3. gradient boosting machine에 의해 중요도가 0이라고 판단된 변수들 제거

→ 536개 변수들 + AUC ROC score 0.7838

4. (옵션) 누적 중요도 95%를 차지하는 변수들만 가져옴

→ 342개 변수들 + AUC ROC score 0.7482