데이터마이닝 보고서

컴퓨터공학과

12161558

김혜윤

마감일 20.06.08

- Module import

```
import numpy as np
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.feature_selection import SelectKBest,chi2
from category_encoders.leave_one_out import LeaveOneOutEncoder
from category_encoders.cat_boost import CatBoostEncoder
from scipy.stats import boxcox
from xgboost.sklearn import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold,cross_val_score,GridSearchCV
from matplotlib.ticker import MaxNLocator
import math
import sklearn.pipeline as pip
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import quantile_transform
from sklearn.ensemble import VotingClassifier
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

- Train.csv 파일 info 확인

```
train = pd.read_csv('train.csv')
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

- Data preprocessing

```
x,y,pred_set,original_train,pred_set_original=get_data()
x.info()
x.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Pclass           889 non-null    int64
 1   Sex              889 non-null    object
 2   Age              889 non-null    float64
 3   SibSp            889 non-null    int64
 4   Parch            889 non-null    int64
 5   Fare             889 non-null    float64
 6   Embarked         889 non-null    object
 7   Deck             889 non-null    object
 8   Title            889 non-null    object
 9   Family_Size      889 non-null    int64
 10  Fare_Bin         889 non-null    object
 11  Age_Grp          889 non-null    object
 12  Fare_Per_Person  889 non-null    float64
 13  Age*Class        889 non-null    float64
 14  Family_Survival  889 non-null    float64
 15  is_alone         889 non-null    int64
 16  has_cabin        889 non-null    int64
 17  is_3rdclass      889 non-null    int64
dtypes: float64(5), int64(7), object(6)
memory usage: 132.0+ KB
```

: data 전 처리 결과

preprocessing에서 phase1clean, phase2clean method가 주 method이고 나머지는 부 method이다.

```
def get_data():
    train_data = pd.read_csv('train.csv')
    submit_x =pd.read_csv('test.csv')
    original_train = pd.read_csv('train.csv')
    original_test = pd.read_csv('test.csv')
    x=phase1clean(train_data)
    pred_set=phase1clean(submit_x)
    x,pred_set=phase2clean(x, pred_set)
    x=x.drop(['PassengerId','Name','Ticket','Cabin','Last_Name'],axis=1)
    pred_set=pred_set.drop(['PassengerId','Name','Ticket','Cabin','Last_Name'],axis=1)
    y = x.Survived
    x = x.loc[:,x.columns!='Survived']
    return x,y,pred_set,original_train,original_test
```

: get_data method

먼저 train_data를 phase1clean 처리를 해준다.

```
def phase1clean(df):

    df.Fare = df.Fare.map(lambda x: np.nan if x==0 else x)

    df.Cabin = df.Cabin.fillna('Unknown')
    cabin_list = ['A', 'B', 'C', 'D', 'E', 'F', 'T', 'G', 'Unknown']
    df['Deck']=df['Cabin'].map(lambda x: substrings_in_string(x, cabin_list))

    title_list=['Mrs', 'Mr', 'Master', 'Miss', 'Major', 'Rev',
                'Dr', 'Ms', 'Mlle','Col', 'Capt', 'Mme', 'Countess',
                'Don', 'Jonkheer']

    df['Title']=df['Name'].map(lambda x: substrings_in_string(x, title_list))

    def replace_titles(x):
        title=x['Title']
        if title in ['Countess','Don', 'Major', 'Capt', 'Jonkheer', 'Rev', 'Col']:
            return 'Rare'
        elif title in ['Countess', 'Mme']:
            return 'Mrs'
        elif title in ['Mlle', 'Ms']:
            return 'Miss'
        elif title =='Dr':
            if x['Sex']=='Male':
                return 'Mr'
            else:
                return 'Mrs'
        else:
            return title

    df['Title']=df.apply(replace_titles, axis=1)

    df['Family_Size']=df['SibSp']+df['Parch']

    return df
```

: phase1clean method

Train_data의 Fare column은 값이 0이면 nan으로 변경해주고, cabin은 결측 값을 'unknown'으로 채운다. 'Deck' 이라는 새로운 column은 substrings_in_string method를 이용해 생성한다.

substrings_in_string은 특정 문자를 찾으면 return 없으면 nan을 return 해주는method이다. Df['Deck'] = df['Cabin'].map(lambda x: substrings_in_string(x,cabin_list))는 으로 Deck column 에는 cabin column의 값들의 첫 번째 문자와 일치하는 cabin_list의 값을 저장한다. df['Title'] = df['Name'].map(lambda x : substrings_in_string(x, title_list)) 로는 Title이라는 새로운 column을 생성하는데 name의 값들에서 title_list와 일치하는 문자를 저장한 column이다. 그 후 title에서 많이 포함된 Mrs, Miss, Mr로 바꿔주고 저 세가지 호칭으로 치환하기 어려운 성들은 Rare 로 변환한다. SibSp(형재/자매/배우자 수) + Parch(부모/ 자식 수)의 값으로 Family_size column을 생성한다.

다시 get_data로 돌아가면 test_data도 phase1clean 처리를 한다.

다음은 phase1clean 처리한 train_data와 test_data를 phase2clean 처리해준다.

```
def phase2clean(train, test):
    train=fill_nan(train)
    train,age_dict=age_grouping(train)
    test=fill_nan(test)
    test['Age_Grp']=test['Age'].astype(int).map(age_dict)

    for df in [train, test]:
          df['Fare_Per_Person']=df['Fare']/(df['Family_Size']+1)

    for df in [train, test]:
        df['Age*Class']=df['Age']*df['Pclass']


    combined=pd.concat([train,test])
    combined=survived_fams(combined)
    combined=is_alone(combined)
    combined=has_cabin(combined)
    combined=is_3stclass(combined)
    train=combined.iloc[:len(train),:]
    test=combined.iloc[len(train):,:]
    test=test.drop(['Survived'],axis=1)

    return [train,test]
```

: phase2clean method

Train_data와 test_data를 fill_nan 처리해준다.

fill_nan method는 Column내의 결측 값을 처리하는 method, num_age에 title별 나이의 평균을 저장하고, 결측 값들을 num_age로 채워준다. num_fare는 pclass별 가격의 평균이고 결측 값을 채워준다.

그 후 fare_grouping을 해주는데, 이 method는 bins 값으로 boundary (-1~7.91 ,,,) 처리를 하고 1,2,3… 숫자 값을 달아준다. 다시 fill_nan 에서는 'embarked' column의 결측 값을 처리, 여기서는 제거해준다. 그 후 train을 aging_grouping을 하는데 이것은 fare처럼 age가 범위가 크고 다양한 값들이 나오기 때문에 boundary를 만들어 구분하게 만들어준다. 이렇게 phase2clean 처리가 완료되면 get_data()로 돌아가서 다른 유용한 column을 만들어줬거나, survived 확인과 연관성이 없는 passengerId, Name, Ticket, Cabin, Last_Name column들을 drop 해준다.

- Modeling

```
classifier = VotingClassifier(estimators=[('XGB', pipeline_xgb), ('LOG', pipeline_log)])

"""model evaluation"""
cv = StratifiedKFold(5, shuffle=True, random_state=42)
accuracies = cross_val_score(classifier, x , y , cv = cv)
print("5 fold cross validation accuracies {}".format(accuracies))
```

```
5 fold cross validation accuracies [0.85393258 0.85393258 0.83146067 0.89325843 0.84180791]
```

VotingClassifier model을 사용했고, ensemble(weighted voiting classifier)로 이용했다. 이것은 각 분류기의 예측을 모아서 가장 많이 선택된 클래스를 예측하는 것이다. Pipeline_xgb는 preprocessor, selectkbest, xgbclassifier을 합친 모델, pipeline_log는 preprocessor, selectkbest, logisticRegression으로 만든 pipeline이다. 이 두개의 model로 votingclassifier로 다수결투표분류 (앙상블)을/를 하여 survived를 예측했다.

점수