데이터 분석 1주차 해결 과정

```python
def get_XY(file_name):
    rawData= pd.read_csv(file_name)
    # rawData.info()
    # RangeIndex: 891 entries, 0 to 890
    # Data columns (total 12 columns):
    # PassengerId     891 non-null int64
    # Survived        891 non-null int64
    # Pclass          891 non-null int64
    # Name            891 non-null object
    # Sex             891 non-null object
    # Age             714 non-null float64
    # SibSp           891 non-null int64
    # Parch           891 non-null int64
    # Ticket          891 non-null object
    # Fare            891 non-null float64
    # Cabin           204 non-null object
    # Embarked        889 non-null object
    # dtypes: float64(2), int64(5), object(5)
```

- pandas를 통한 데이터 파악
- Age, Cabin의 데이터 수가 이상하다?

```python
print(rawData.isnull().sum())       # How to consider NaN data?
# PassengerId      0
# Survived         0
# Pclass           0
# Name             0
# Sex              0
# Age            177
# SibSp            0
# Parch            0
# Ticket           0
# Fare             0
# Cabin          687
# Embarked         2
```

- Null Data 존재, 이를 어떻게 처리할지?
- Cabin(선실)의 경우는 기록값이 없는 경우 선실이 없는 경우로 파악, ‘No_Cabin’ 입력
- Embarked의 경우 특이값으로 생각하고 삭제

```
rawData['Cabin'].fillna('No_Cabin', inplace= True)
rawData['Null_inAge']= rawData['Age'].isnull()
rawData['Age'].fillna(0, inplace= True)
rawData= rawData[rawData['Embarked'].isnull() == False]
```

- Age의 경우, 없는 데이터도 너무 많고 0으로 기록할 수도 없는 데이터!
- 두 가지 생각- 적절한 Age 값을 regression으로 도출 후 classification 시도, 또는 소실된 데이터임을 표시하는 식으로?
- 'Null_inAge' feature 추가, 이 T/F 값이 Age= 0이라는 bias를 해결할 수 있을지?

```
rawData['Cabin'].fillna('No_Cabin', inplace= True)
rawData['Null_inAge']= rawData['Age'].isnull()
rawData['Age'].fillna(0, inplace= True)
rawData= rawData[rawData['Embarked'].isnull() == False]
```

```
print(rawData['Cabin'].unique())
# ['No_Cabin' 'C85' 'C123' 'E46' 'G6' 'C103' 'D56' 'A6' 'C23 C25 C27' 'B78'
#  'D33' 'B30' 'C52' 'C83' 'F33' 'F G73' 'E31' 'A5' 'D10 D12' 'D26' 'C110'
#  'B58 B60' 'E101' 'F E69' 'D47' 'B86' 'F2' 'C2' 'E33' 'B19' 'A7' 'C49' 'F4'
#  'A32' 'B4' 'B80' 'A31' 'D36' 'D15' 'C93' 'C78' 'D35' 'C87' 'B77' 'E67'
#  'B94' 'C125' 'C99' 'C118' 'D7' 'A19' 'B49' 'D' 'C22 C26' 'C106' 'C65'
#  'E36' 'C54' 'B57 B59 B63 B66' 'C7' 'E34' 'C32' 'B18' 'C124' 'C91' 'E40'
#  'T' 'C128' 'D37' 'B35' 'E50' 'C82' 'B96 B98' 'E10' 'E44' 'A34' 'C104'
#  'C111' 'C92' 'E38' 'D21' 'E12' 'E63' 'A14' 'B37' 'C30' 'D20' 'B79' 'E25'
#  'D46' 'B73' 'C95' 'B38' 'B39' 'B22' 'C86' 'C70' 'A16' 'C101' 'C68' 'A10'
#  'E68' 'B41' 'A20' 'D19' 'D50' 'D9' 'A23' 'B50' 'A26' 'D48' 'E58' 'C126'
#  'B71' 'B51 B53 B55' 'D49' 'B5' 'B20' 'F G63' 'C62 C64' 'E24' 'C90' 'C45'
#  'E8' 'B101' 'D45' 'C46' 'D30' 'E121' 'D11' 'E77' 'F38' 'B3' 'D6' 'B82 B84'
#  'D17' 'A36' 'B102' 'B69' 'E49' 'C47' 'D28' 'E17' 'A24' 'C50' 'B42' 'C148']
Cabin_char= lambda x: x[0]+str(len(x.split()))
rawData['Cabin']= rawData['Cabin'].map(Cabin_char)
# ['N1' 'C1' 'E1' 'G1' 'D1' 'A1' 'C3' 'B1' 'F1' 'F2' 'D2' 'B2' 'C2' 'B4' 'T1'
#  'B3']
```

- Cabin(선실)의 경우, 같은 character는 같은 종류의 선실로 가정
- 두 개 이상의 선실 사용하는 승객 존재
- 이를 감안하여 Cabin을 character + number of cabin으로 재조정

```python
if 'Survived' in rawData.columns:
    y= rawData['Survived']
    X= rawData.drop(['PassengerId', 'Survived', 'Name', 'Ticket'], axis= 1)
else:
    y= rawData['PassengerId']
    X= rawData.drop(['PassengerId', 'Name', 'Ticket'], axis= 1)
X= pd.get_dummies(X)
# X.info()
print(X.isnull().sum())     # NaN check
return(X, y)
```

- train data와 test data X, y 설정
- pd.get_dummies: categorical feature를 0/1 features로 변경 (one-hot encoding!)

```python
X, y= get_XY('train.csv')
X_test, PassengerId_test= get_XY('test.csv')
print(X.columns)
print(X_test.columns)
# Index(['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex_female', 'Sex_male',
#        'Cabin_A1', 'Cabin_B1', 'Cabin_B2', 'Cabin_B3', 'Cabin_B4', 'Cabin_C1',
#        'Cabin_C2', 'Cabin_C3', 'Cabin_D1', 'Cabin_D2', 'Cabin_E1', 'Cabin_F1',
#        'Cabin_F2', 'Cabin_G1', 'Cabin_N1', 'Cabin_T1', 'Embarked_C',
#        'Embarked_Q', 'Embarked_S'],
#       dtype='object')
# Index(['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex_female', 'Sex_male',
#        'Cabin_A1', 'Cabin_B1', 'Cabin_B2', 'Cabin_B3', 'Cabin_B4', 'Cabin_C1',
#        'Cabin_C2', 'Cabin_C3', 'Cabin_D1', 'Cabin_D2', 'Cabin_E1', 'Cabin_E2',
#        'Cabin_F1', 'Cabin_F2', 'Cabin_G1', 'Cabin_N1', 'Embarked_C',
#        'Embarked_Q', 'Embarked_S'],
#       dtype='object')
X_test['Cabin_E1']= X_test['Cabin_E1']+ X_test['Cabin_E2']
X_test= X_test.drop('Cabin_E2', axis= 1)
X_test['Cabin_T1']= 0
X_test['Fare'].fillna(0, inplace= True)
```

- X, X_test, y 형성
- 그런데 에러가….! X와 X_test의 feature 차이
- E2는 E1과 비슷하다고 가정
- Fare에서의 Null 값 수정

```
clf5= RandomForestClassifier(criterion= "entropy", max_depth= 5, random_state= 50)
clf5.fit(X, y)
clf5_result= pd.Series(clf5.predict(X_test), name= 'Survived')

clf20= RandomForestClassifier(criterion= "entropy", max_depth= 20, random_state= 50)
clf20.fit(X, y)
clf20_result= pd.Series(clf20.predict(X_test), name= 'Survived')

clf_dt5= DecisionTreeClassifier(criterion= "entropy", max_depth= 5, random_state= 50)
clf_dt5.fit(X, y)
clf_dt5_result= pd.Series(clf_dt5.predict(X_test), name= 'Survived')

clf_dt20= DecisionTreeClassifier(criterion= "entropy", max_depth= 20, random_state= 50)
clf_dt20.fit(X, y)
clf_dt20_result= pd.Series(clf_dt20.predict(X_test), name= 'Survived')
```

- Random Forest, Decision Tree 수행
- Python, Scikit_Learn
- max_depth= 5 or 20
- criterion= entropy

예측 결과

```
result_of_test= pd.concat([PassengerId_test, clf5_result], axis=1)
print(result_of_test.head(30))
result_of_test.to_csv('result_rf5.csv', index= False)
# Score: 0.78468

result_of_test= pd.concat([PassengerId_test, clf20_result], axis=1)
result_of_test.to_csv('result_rf20.csv', index= False)
# Score: 0.73684

result_of_test= pd.concat([PassengerId_test, clf_dt5_result], axis=1)
result_of_test.to_csv('result_dt5.csv', index= False)
# Score: 0.73205

result_of_test= pd.concat([PassengerId_test, clf_dt20_result], axis=1)
result_of_test.to_csv('result_dt20.csv', index= False)
# Score: 0.64593
```

- Random Forest의 경우, depth가 5인 경우가 20인 경우보다 정확도가 높다?
- Random Forest가 Decision Tree보다 높은 정확성
- 80%를 넘지 못한 accuracy

배운 점

- Accuracy >= 0.8은 정말 어려운 일입니다. >=0.9는 더더욱 어려울 것 같아요
- Age 데이터 처리 방식이 아쉽습니다. 이 쪽의 지식이 있으면 조금 더 잘 처리할 수 있었을 텐데요.
- Tree를 만들 때 Depth가 높다고 반드시 Accuracy가 높은 것은 아닙니다! 오히려 역순으로 나타났습니다.
- 데이터 전처리 과정이 훨씬 시간 소모가 많습니다. 그리고 이 때 지우는 데이터, 이 때 처리하지 못한 특이값 데이터가 정확성에 영향을 미치는 것 같습니다.