

Open Competition Kaggle – Titanic

이름: 유서정

목차

1. 데이터 전처리
2. 모델 학습
3. 결과
4. 결론 및 고찰

데이터 전처리

1. 데이터 로드

Training data 로드

```
[306] dataset=pd.read_csv('../gdrive/My Drive/datamining/train.csv')
dataset
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

- 데이터 개수: 891

2. 이름 변환 함수

```

def convertName(data):

    for i in range(len(data)):
        curName=data["Name"][i].split(",")[1].split(".")[0].strip()

        if data["Age"][i]==None:
            print("None")

        if curName == "Mr":
            data["Name"][i]=0
            data["Sex"][i]=0

        elif(curName=="Miss" or curName=="Ms" or curName=="Mme" or curName=="Mlle" or curName=="Mrs"):
            data["Name"][i]=1
            data["Sex"][i]=1

        elif(curName=="Master"):
            data["Name"][i]=2

        elif(curName=="Dr"):
            data["Name"][i]=3

        else:
            data["Name"][i]=4
            # X["Sex"][i]=0

    return data

```

- 이름에서 직업과 성별을 유추할 수 있다.
- 위 조건문에 해당되지 않는(else에서 처리되는) 몇 가지 있으나, 각각 경우의 수가 적어 하나의 class로 묶어서 처리하였다.
- 이름을 제외하고 학습할 때보다 이름을 포함하여 학습한 결과의 정확도가 0.01 증가하였다.

3. 관계 없는 feature 제거 및 Sex와 Embarked를 숫자로 변환

```

X=dataset.drop(columns=['PassengerId','Cabin','Ticket'])
repCol3 = { "male":0, "female" : 1}
repCol8 = {"C" : 0 , "Q" : 1 , 'S' : 2 }
X.replace({"Sex": repCol3, "Embarked": repCol8} , inplace = True )
X['Embarked'].fillna(2,inplace=True)
X=convertName(X)
Y=X.pop("Survived")

```

- PassengerId, Cabin, Ticket은 결과에 영향을 미치지 않을 것이라고 판단하여 제거하였

다.

- PassengerId: 모든 사용자가 다르다.
- Cabin: null 값이 많고, 중복이 별로 없다.
- Ticket: 몇 개의 ticket을 제외하고는, 모두 다른 ticket을 가진다.
- 모델 학습에 용이하도록 각 feature 값을 숫자로 변환하였다.
- "Embarked" column에서 null 값의 경우, 가장 빈도가 높은 'S'(2)로 변환하였다.

4. 'Age' 예측

```
] X.isnull().sum()
```

```
Pclass      0
Name         0
Sex          0
Age        177
SibSp        0
Parch        0
Fare         0
Embarked     0
dtype: int64
```

- Embarked 는 `[X['Embarked'].fillna(2,inplace=True)]`를 수행 전 2 개의 null 값을 가졌기 때문에, 단순히 가장 많이 나온 값으로 대체하였다.
- 하지만 Age의 경우 null값이 총 891개의 데이터 중 177개로 많은 부분을 차지한다.
- 데이터 셋의 크기가 충분히 크지 않기 때문에, null을 포함한 데이터 tuple을 삭제하는 것 보다는 예측한 값이 더 좋을 것이라고 판단하였다.
 - 실험 결과, null을 포함한 tuple을 삭제하고 트레이닝한 결과보다, 이름을 포함하여 학습시킨 결과, 정확도가 0.05 증가하였다.

Age 값이 Null이 아닌 data 추출

```
[ ] idx_NaN_age=list(X["Age"][X["Age"].notnull()].index)
    Age_X=[]

    for i in idx_NaN_age:
        Age_X.append(X.iloc(0)[i])

    Age_X=pd.DataFrame(Age_X)
    Age_Y=Age_X.pop("Age")
    #Age_Y=pd.DataFrame(Age_Y)
```

Age 값이 Null인 data 추출

```
[ ] idx_age=list(X["Age"][X["Age"].isnull()].index)
    Age_NaN_X=[]
    for i in idx_age:
        Age_NaN_X.append(X.iloc(0)[i])

    Age_NaN_X=pd.DataFrame(Age_NaN_X)
    Age_NaN_X.pop("Age")
```

- Age 값이 null이 아닌 데이터를 training data로 사용해, 모델을 학습 후, Age 값이 null인 값들을 예측해서 채워 넣었다.

Mondrian Forest 알고리즘을 사용하여 age 예측 - <https://github.com/scikit-garden/scikit-garden>

```
[ ] pip install scikit-garden
```

```
[ ] from skgarden import MondrianForestClassifier
    from skgarden import MondrianForestRegressor
    mfr=MondrianForestRegressor(random_state=1,max_depth=5)
    mfr.fit(Age_X,Age_Y)
    X=fillNaNAge(mfr,X)

    # Normalization
    X["Fare"]=(X["Fare"]-X["Fare"].min())/(X["Fare"].max()-X["Fare"].min())
    X["Age"]=(X["Age"]-X["Age"].min())/(X["Age"].max()-X["Age"].min())
```

- Age를 예측할 모델로는 MondrianForestRegressor 알고리즘을 사용하였다. 위 library는 "scikit-garden"을 설치하여 사용할 수 있다.

5. 'Fare'와 'Age' Normalization

```
# Normalization
X["Fare"]=(X["Fare"]-X["Fare"].min())/(X["Fare"].max()-X["Fare"].min())
X["Age"]=(X["Age"]-X["Age"].min())/(X["Age"].max()-X["Age"].min())
```

- 상대적으로 큰 값을 가질 수 있는 두 feature를 normalization 해주었다.

6. 사전 테스트를 위한 test set 나누기

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.7, random_state=1)
```

7. Test data 추출

Test data 추출 및 데이터 변환

```
[398] test=pd.read_csv("../gdrive/My Drive/datamining/test.csv")
      test_x=test.drop(columns=['PassengerId','Cabin','Ticket'])
      test_x=convertName(test_x)
      test_x.replace({"Sex": repCol3, "Embarked": repCol8} , inplace = True )
      # Normalization
      test_x["Fare"]=(test_x["Fare"]-test_x["Fare"].min())/(test_x["Fare"].max()-test_x["Fare"].min())
      test_x["Age"]=(test_x["Age"]-test_x["Age"].min())/(test_x["Age"].max()-test_x["Age"].min())
      test_x=fillnaAge(mfr,test_x)
      test_x["Fare"].fillna(test.Fare.mean(),inplace=True)
```

모델 학습

- 여러 개의 모델을 결합해 Voting Classify를 하는 ensemble 모델을 만들었다.
- 각 모델은 GridSearchCV를 통해 파라미터를 최적화시켰다.
- 아래는 채택된 모델들이다.
 - 이 모델들 이외에, GaussianNB, RandomForestClassifier, GradientBoostingClassifier, KNeighborsClassifier 등의 모델들을 이용하여 연구해 보았으나, 아래의 조합이 가장 정확도가 높았다.

1. Model1 – DecisionTreeClassifier

```
#model1 - DecisionTreeClassifier
scoring = {'AUC': 'roc_auc', 'Accuracy': make_scorer(accuracy_score)}

gs = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid={'min_samples_split': range(2, 403, 10)},
                  scoring=scoring, refit='AUC', return_train_score=True)
gs.fit(X_train, Y_train)
model1=gs
print("model 1: ", model1.best_score_)
```

- 개별 트레이닝 정확도: 0.86

2. Model2 – LinearSVC

```
#model2 - LinearSVC
pipe = Pipeline([
    # the reduce_dim stage is populated by the param_grid
    ('reduce_dim', 'passthrough'),
    ('classify', LinearSVC(dual=False, max_iter=10000))
])

N_FEATURES_OPTIONS = [2, 4, 8]
C_OPTIONS = [1, 10, 100, 1000]
param_grid = [
    {
        'reduce_dim': [PCA(iterated_power=7), NMF()],
        'reduce_dim__n_components': N_FEATURES_OPTIONS,
        'classify__C': C_OPTIONS
    },
    {
        'reduce_dim': [SelectKBest(chi2)],
        'reduce_dim__k': N_FEATURES_OPTIONS,
        'classify__C': C_OPTIONS
    },
]
reducer_labels = ['PCA', 'NMF', 'KBest(chi2)']

grid = GridSearchCV(pipe, n_jobs=1, param_grid=param_grid)
grid.fit(X_train, Y_train)
model2=grid
print("model2: ", model2.best_score_)
```

- 개별 트레이닝 정확도: 0.80

3. Model3 – ExtraTreesClassifier

```
#model3 – ExtraTreesClassifier
ExtC = ExtraTreesClassifier()

ex_param_grid = {"max_depth": [None],
                 "max_features": [1, 3, 10],
                 "min_samples_split": [2, 3, 10],
                 "min_samples_leaf": [1, 3, 10],
                 "bootstrap": [False],
                 "n_estimators" :[100,300],
                 "criterion": ["gini"]}



gsExtC = GridSearchCV(ExtC,param_grid = ex_param_grid, scoring="accuracy", n_jobs= 4, verbose = 1)

gsExtC.fit(X_train,Y_train)
model3= gsExtC
print("model3: ",model3.best_score_)
```

- 개별 트레이닝 정확도: 0.83

결과

titanic prediction (9).csv an hour ago by Seojeong Yu Predict age using Mondrian Forest Algorithm and Predict 'Survived' using DecisitonTreeClassifier, RandomForestClassifier, GradientBoostingClassifier and ExtraTressClassifier.	0.77990
titanic prediction (8).csv 5 hours ago by Seojeong Yu Predict age using Mondrian Forest Algorithm and Predict 'Survived' using DecisitonTreeClassifier, LinearSVC, GradientBoostingClassifier and ExtraTressClassifier.	0.78468
titanic prediction (7).csv 6 hours ago by Seojeong Yu Predict age using Mondrian Forest Algorithm and Predict 'Survived' using DecisitonTreeClassifier, LinearSVC and ExtraTreesClassifier	0.79904
titanic prediction (6).csv 6 hours ago by Seojeong Yu Predict age using Mondrian Forest Algorithm and Predict 'Survived' using DecisitonTreeClassifier	0.79904
titanic prediction (5).csv 6 hours ago by Seojeong Yu Predict age using Mondrian Forest Algorithm and Predict 'Survived' using DecisitonTreeClassifier	0.77990
titanic prediction (4).csv 8 hours ago by Seojeong Yu Predict age using Mondrian Forest Algorithm and Predict 'Survived' using DecisitonTreeClassifier	0.77511
titanic prediction (3).csv 9 hours ago by Seojeong Yu Predict age using Mondrian Forest Algorithm	0.73684
titanic prediction (2).csv 10 hours ago by Seojeong Yu Predict age and use them as learning data.	0.76076
titanic prediction (1).csv 13 hours ago by Seojeong Yu Extract jobs by name. Use Random Forest.	0.75598
titanic prediction.csv 14 hours ago by Seojeong Yu I used a random forerest with several classifiers.	0.74641

2606	Seojeong Yu		0.79904	10	2h
Your Best Entry 					
Your submission scored 0.77990, which is not an improvement of your best score. Keep trying!					

결론 및 고찰

1. 데이터 처리

- 처음에는 이름은 'PassengerId'처럼 의미 없는 feature라고 생각하여 삭제하였다.
- 또한, null 값이 포함된 값을 단순히 제거하거나, 평균으로 채워 넣었다.
- 하지만 이름에서 유의미한 데이터를 추출하였고, feature 값을 예측해 넣음으로써 정확도를 2~3%정도 향상시킬 수 있었다.

2. 모델 학습

- Ensemble 모델의 성능을 향상 시키기 위해, 모델을 종류(개수)를 늘리기 보다는, 해당 데이터셋에 대해 정확도가 높은 모델들을 뽑고, 가장 정확도가 좋은 조합을 도출해 내는 것이 중요하다.
- "GridSearchCV"를 사용해 각 모델의 파라미터를 튜닝함으로써 정확도를 2%정도 향상시킬 수 있었다.