Report

AML Term Project

학교: 세종대학교	학과: 데이터사이언스
교과목: 고급기계학습	학번: 17011739
담당교수: 김미숙	이름: 이재훈

주제

고객이 가입한 서비스와 고객정보, 고객의 계정과 신상정보를 활용하여 서비스 탈퇴 여부를 예측하는 모델 생성

개발동기

평소 상업적인 부분에 관심이 많았다. 이번 Term Project를 위한 데이터를 찾다가 'Telco Customer Churn' 이라는 좋은 주제를 찾게 되었다.

개발과정

1. 데이터 처리

모델의 훈련과 검증을 하기위해 먼저 https://www.kaggle.com/blastchar/telco-customer-churn 에 있는 데이터를 다운로드 받았다.

```
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

이 데이터는 거의 대부분 문자로 되어있어 인코딩을 어떻게 하였는지에 따라 모델의 성능이 달라질 여지가 있었다. 따라서 각각 다른 방법으로 인코딩한 두 개의 데이터 셋을 만들

었다.

데이터셋1:

- object형의 이진 컬럼(Female/Male, Yes/No): OrdinalEncoding

  (0과 1로 맵핑 됨)

- 그 외의 object형 컬럼: OneHotEncoding

- 숫자형 컬럼: StandardScaler

데이터셋2:

- object형은 OrdinalEncoding

- 숫자형 컬럼: RobustScaler

2. 모델 훈련

모델훈련에 앞서 데이터를 훈련 셋과 검증 셋으로 나누었다. 모델 훈련에는 RandomForestClassifier, Bagging, XGboost, Adaboost, LinearRegression, SVC를 사용하였다. 데이터 셋을 훈련한 뒤 각 방법마다 accuracy_score를 따로 저장하여 비교해 본 결과는 아래와 같다.

	train_set_score	test_set_score
RandomForest	0.799822	0.786780
Bagging	0.806933	0.789623
XGB	0.840533	0.797441
ADA	0.811911	0.788202
LinReg	0.803200	0.793177
Poly	0.815467	0.786780
SVC	0.818311	0.786780

(데이터 1)

	train_set_score	test_set_score
RandomForest	0.801778	0.788913
Bagging	0.813333	0.784648
XGB	0.849956	0.788202
ADA	0.817067	0.791756
LinReg	0.799644	0.794598
Poly	0.813689	0.793177
SVC	0.812444	0.789623

(데이터2)

데이터셋1의 XGboost가 가장 좋은 성능을 보여주었다. 그러나 크게 유의미할 정도로 좋은 것은 아닌 듯하다. 또한 Encoding과정에서 데이터셋1은 컬럼이 기존 21개에서 42개로 크게 늘었기 때문에 굳이 데이터셋1을 사용할 필요는 없어 보인다. 데이터셋1과 2의 성능차이가 크게 없는 이유는 컬럼에 해당하는 값들이 다양하지 않기 때문일 것이라 예상된다. 이진 컬럼이 아닌 것들을 그냥 OrdinalEncoding 하여도 값이 최대 3까지 밖에 생성되지 않았다. 또한 StandardScaler와 RobustScaler의 성능차이 또한 거의 없었다.

데이터셋2에서는 LinearRegression이 가장 좋은 성능을 보여주었다. 그러나 별다른 파라미터 튜닝 없이 좋은 성능을 보여준 XGboost가 발전가능성이 있다고 판단하였고 파라미터 튜닝을 통해 성능향상을 꾀하였다.

```
train_set:  0.8154666666666667
test_set:  0.7931769722814499
```

XGboost 튜닝결과 이전보다 조금 더 나은 성능을 보여주긴 하였으나 LinearRegression보다는 성능이 조금 좋지 않다. 새로운 데이터를 추가하고 파라미터 튜닝을 다시 해보면 성능이 향상될 것이다.

한계

모델을 훈련하는 과정에서 전체적으로 overfitting되는 문제가 발생하였다. 실제로 파라미터 튜닝을 전혀 하지 않을 경우 훈련셋의 정확도는 100%가까이 올라갔다. Overfitting을 방지하기 위해 규제를 다소 크게 한 것이 전체적인 성능의 하락으로 이어진 것이 아닌가 추측된다. 규제를 전혀 하지 않아도 검증셋의 정확도는 78% 정도가 나왔으며 규제를 한 것과 큰 차이는 없었다. 따라서 모델의 성능을 높이려면 파라미터 튜닝에 시간을 쓸 것이 아니라 더 많은 데이터를 새로 추가해야 할 것이다.