Machine Learning Assignment

이 해 민

2021312394
인공지능융합전공

1. Problem Definition

텍스트의 LIWC(Linguistic Inquiry and Word Count) 정보가 담긴 데이터셋을 통해 MBTI 선호지표 중 T와 F를 높은 정확도로 분류하는 것이 목표이다.

2. Data

2.1. Data Description

데이터는 6940개의 행과 94개의 열로 이루어져 있다. 이때 마지막 열을 'label'값으로 T와 F 중 어느 것에 해당하는 지를 나타낸다. 클래스의 개수는 총 2개로 이진분류이며, 0은 T, 1은 F를 의미한다. Label 열을 제외한 나머지 93개의 열이 feature이며, 'WC', 'Analytic', 'Authentic', 'Tone' 등으로 이루어져 있다.

2.2. Data Pre-processing

```
label.value_counts()
```
✓ 0.0s
```
1    3756
0    3184
Name: label, dtype: int64
```

2.2.1. 불균형 해결

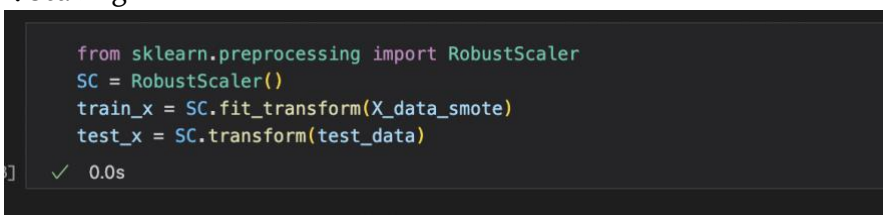
각 label에 해당하는 값의 개수를 value counts로 세어봤을 때 1은 3756, 0은 3184로 훈련 데이터에 불균형이 있다는 사실을 파악하였다. 이는 recall 점수가 낮아지는 문제로 이어질 수 있으므로, SMOTE를 통해 데이터의 균형을 맞춰주었다.

```
from imblearn.over_sampling import SMOTE

smote = SMOTE()
X_data_smote, label_smote = smote.fit_resample(X_data,label)

# SMOTE 적용 후 데이터셋
print("원본 데이터셋 클래스 분포:")
print(label.value_counts())

print("\nSMOTE 적용 후 데이터셋 클래스 분포:")
print(label_smote.value_counts())
```
✓ 0.3s
```
원본 데이터셋 클래스 분포:
1    3756
0    3184
Name: label, dtype: int64

SMOTE 적용 후 데이터셋 클래스 분포:
0    3756
1    3756
Name: label, dtype: int64
```

총 데이터의 개수가 6940개로 많지 않은 편이므로, 언더 샘플링을 사용하기보다는 데이터를 보존하며 overfitting을 방지할 수 있는 합성 데이터 생성 방법을 선택하는 것이 적합하다고 판단했다. SMOTE 알고리즘은 KNN 기법이나 부트스트래핑 방법을 통해 새로운 소수 클래스의 데이터를 생성해내는 방법이다.[1] 이 방법으로 데이

터의 개수를 각각 3756, 3756개로 동일하게 맞춰주었다.

2.2.2. Scailing



Standard scaler, minmax scaler, robust scaler 중 중간값을 이용하여 극단값에 가장 영향을 받지 않는 robust scaler를 사용하였다. 실제로, SMOTE를 적용하지 않고 SVC 모델로 실험을 진행하였을 때 Robust scaler를 적용했을 때 accuracy가 가장 높은 것을 확인했다.

3. Analysis

이 데이터의 feature는 93개로 매우 고차원의 데이터이다. Feature가 이렇게 많은 상황에서는 '차원의 저주' 문제가 발생할 수 있으므로, RFECV 방법을 통해 주요한 feature만 남기고 모델을 학습시켰다. T와 F를 분류하는데 LIWC 데이터의 모든 feature들이 주요한 영향을 미치는 것은 아니므로, 일반화된 분류를 위해 이 과정이 필요하다고 판단했다.

4. Model

4.1. Classifier

RFECV를 통해 XGB classifier를 평가 모델로 하여 중요하지 않은 feature들을 순차적으로 제거한 후, cross validation을 통해 최종 선택할 feature의 개수를 선정하였다.[2] XGB 분류기를 평가 모델로 한 이유는 logistic regression을 평가 모델로 했을 때보다 캐글에서 미세하게 높은 정확도를 보여줬기 때문이다. 선택된 feature의 개수는 총 54개로, 선택된 columns는 다음과 같다.

그 후 SVM, logistic regression, randomforest, xgboost 모델로 각각 테스트 accuracy를 측정하였다.

4.2. Design Consideration

Feature의 개수를 감소시켜 주요한 feature로만 classification을 진행한다는 것에 중점을 두었다. 또한, 각 기계학습 모델 별로 성능이 다를 수 있으므로 test accuracy를 통해 비교해가며 실험을 진행하였다.

5. Experiments

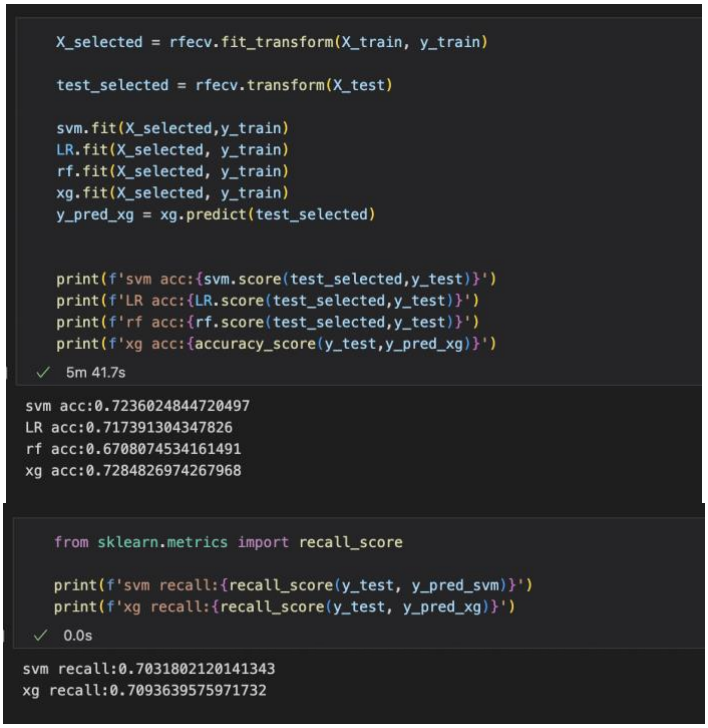
5.1. Settings

Train: Test = 8:2 로 설정하였다. 세부적인 파라미터를 바꾸며 실험해보았을 때 logistic regression, randomforest, xgb 모델은 모두 default 값일 때 test accuracy가 가장 높게 나오는 것으로 판단된다. SVM 모델은 gamma값을 0.01로 설정하였을 때 더 높은 정확도를 보였다. 따라서, 파라미터 조정 보다는 RFECV의 평가 모델과 최종 학습 모델을 선정하는 것에 초점을 맞추어 실험을 진행하였다.

5.2. Performance Metrics

Accuracy score와 recall 점수로 그 예측을 진행하였다. 하지만, 임의로 생성한 test dataset의 test accuracy와 실제 Kaggle test accuracy가 다소 다른 결과가 나오는 것을 확인하여, 최종 선정을 위해서는 Kaggle test accuracy를 활용하였다.

5.3. Results



```
X_selected = rfecv.fit_transform(X_train, y_train)

test_selected = rfecv.transform(X_test)

svm.fit(X_selected,y_train)
LR.fit(X_selected, y_train)
rf.fit(X_selected, y_train)
xg.fit(X_selected, y_train)
y_pred_xg = xg.predict(test_selected)


print(f'svm acc:{svm.score(test_selected,y_test)}')
print(f'LR acc:{LR.score(test_selected,y_test)}')
print(f'rf acc:{rf.score(test_selected,y_test)}')
print(f'xg acc:{accuracy_score(y_test,y_pred_xg)}')
```

✓ 5m 41.7s

```
svm acc:0.7236024844720497
LR acc:0.717391304347826
rf acc:0.6708074534161491
xg acc:0.7284826974267968
```

```
from sklearn.metrics import recall_score

print(f'svm recall:{recall_score(y_test, y_pred_svm)}')
print(f'xg recall:{recall_score(y_test, y_pred_xg)}')
```

✓ 0.0s

```
svm recall:0.7031802120141343
xg recall:0.7093639575971732
```

SVM과 XGB 모델의 accuracy는 각각 0.7236, 0.7284이고, recall score는 각각 0.7031, 0.7093으로 매우 유사하다. Kaggle의 두 모델을 다 올려보고 실험하고자 했지만, 참여 기회의 제한으로 RFECV를 사용하지 않았을 때 kaggle에서 약 2% 정도 (SVM – 0.74464, XGB – 0.7224) 더 높은 정확도를 보여주었던 SVM 모델을 사용했다. 최종적으로 Kaggle accuracy 74.794%라는 결과를 얻었다. 또한 데이터의 개수를 세어보았을 때 1이 865, 0이 860으로 비교적 균등하게 분류가 진행된 것을 확인할 수 있었다.

6. Discussion and Limitation

본 프로젝트는 데이터의 불균형을 조정하고, 주요 feature들만 사용해 예측을 진행하여 정확도를 높였다는 점에서 그 의의가 있다. 또한, 어떤 feature들이 T와 F를 분류하는데 도움을 주었는지 또한 파악할 수 있다. 이렇게 생성된 모델은 다른 모델들보다 조금 더 강건하다는 장점이 있다. 다만, feature들을 줄였음에도 여전히 54개라는 많은 개수와 RFECV 기법을 활용한 상황에서 시간적 한계로 파라미터 조정을 세밀하게 하지 못했다는 한계가 존재한다. 추가적인 파라미터 조정을 실시하고, 실험에서 사용된 모델 외 다양한 모델을 적용해본다면 더욱 강건하고 높은 정확도를 보이는 모델을 만들 수 있을 것이다.

7. References

[1] SMOTE를 통한 데이터 불균형 처리,
https://mkjjo.github.io/python/2019/01/04/smote_duplicate.html.(2023.05.13접근)

[2]RFE와 REFCV: 유의미한 변수를 선택하는 방법,
https://abluesnake.tistory.com/141.(2023.05.13접근)