AI+X:머신러닝 과제#3

ICT융합학부 2021093518 임동희

과제로 주어진 와인데이터 분류실습은 로지스틱 회귀를 적용하여 실행한 분류였습니다. 로지스틱 회귀란, 선형 회귀 방식을 이용한 이진 분류 알고리즘입니다. 분류 레이블은 0 또는 1입니다. 따라서 0에서 1사이의 범위에서 예측 결과를 갖는 모델이 필요합니다. 가설 결과에 시그모이드(S자 모양의 함수)함수를 적용시키는 것이 특징입니다. 과제 1번으로 제시된 accuracy, precision, recall은 로지스틱 회귀에서 분류 평가 지표를 말합니다. 가장 먼저 accuracy는 단순한 정확도를 나타냅니다. 더욱 자세히는, 단순히 분류가 맞았는지 틀렸는지에 관한 정확도를 말합니다. precision은 정밀도를 의미합니다. 즉, 분류 모델이 positive(양성)으로 예측한 전체 건수에서 옳게 예측한 건수의 비율을 나타냅니다. 즉, 양성의 예측 정확도, 다시 말해서 양성 정답률을 나타내는 지표입니다. 마지막으로 recall은 재현율을 의미하며, 실제로 positive(양성)인 것 중에서 분류모델이 양성으로 옳게 예측한 건수의 비율을 나타냅니다. Accuracy와 달리 precision 그리고 recall은 관점을 달리하여 각각 양성 분류에 대해 나타낸 지표라고 할 수 있습니다. 따라서 저도 기존 실습내용 아래에 코드를 추가하여 로지스틱 회귀를 적용하였을 때의 accuracy, precision, recall값을 계산하였습니다. 결과는 다음과 같습니다. Sklearn.metrics를 불러와 각각 내장된 함수 accuracy_score, precision_score, recall_score를 이용하여 accuracy, precision, recall 값을 구하였습니다.

```
from sklearn.metrics import accuracy_score

pred = log_reg.predict(X_test)
accuracy_score(y_test, pred)
```

```
0.9969230769230769
```

```
[46] from sklearn.metrics import precision_score

precision_score(y_test, pred)
```

```
0.9905362776025236
```

```
[47] from sklearn.metrics import recall_score

recall_score(y_test, pred)
```

```
0.9968253968253968
```

이후, 2번과제에 제시된 것처럼 Binarizer를 활용하여 기준값 범위를 0으로, 이상은 1로 제한하여 다시 한 번 더 accuracy score, precision score, recall score 값을 구해주었습니다. 0.05로 변경하였을때는 오른쪽과 같이 나왔고 0.3으로 변경하였을 때 {Accuracy : 0.9946153846153846, Precision : 0.98125, Recall : 0.9968253968253968}, 0.5로 변경하였을 때 {Accuracy : 0.9969230769230769, Precision : 0.9905362776025236, Recall : 0.9968253968253968 } 0.8로 변경하였을 때 {Accuracy : 0.9946153846153846, Precision : 0.9967741935483871, Recall : 0.9809523809523809} 마지막으로, 0.95로 변경하였을 때 {Accuracy : 0.9815384615384616, Precision : 0.9965870307167235, Recall : 0.926984126984127}와 같은 값이 나왔습니다. 값을 분석해보면, Accuracy는 0.05에서 0.8까지는 값이 커질 수록 커지고, 이후에는 값이 커졌지

```
[100] from sklearn.preprocessing import Binarizer

custom_threshold = 0.05

pred_proba = log_reg.predict_proba(X_test)
pred_proba_1 = pred_proba[:,1].reshape(-1, 1)
```

```
[101] binarizer =Binarizer(threshold = custom_threshold).fit(pred_proba_1)
custom_predic = binarizer.transform(pred_proba_1)
custom_predic[:5]
```

```
array([[1.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

```
print('Threshold :', custom_threshold)
print('Accuracy :', accuracy_score(y_test, custom_predic))
print('Precision :', precision_score(y_test, custom_predic))
print('Recall :', recall_score(y_test, custom_predic))
```

```
Threshold : 0.05
Accuracy : 0.9761538461538461
Precision : 0.9127906976744186
Recall : 0.9968253968253968
```

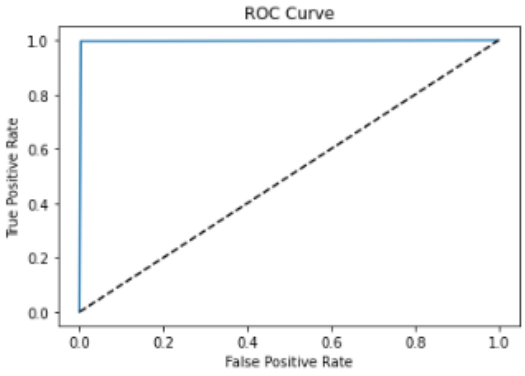
만 오히려 줄어드는 것을 확인할 수 있었습니다. 또한, Precision도 값이 커짐에 따라 0.8까지는 값이 커지고 이후에는 감소하는 것을 확인할 수 있었습니다. 마지막으로 Recall값은 0.5까지는 변함이없다가 0.8부터 감소함을 알 수 있었습니다. threshold값을 증가시키면 정확도와 정밀도를 올릴 수 있지만, 과도하게 조정할 경우 오히려 재현율이 감소할 수 있으므로 값을 잘 모델의 목적과 필요한 곳에 따라 (재현율이 중요한지 혹은 정밀도가 중요한지에 따라) 잘 판단하여 값을 조정하여 신뢰도가 높은 로지스틱회귀모델을 만들 수 있다는 것을 분석결과를 통해 알 수 있었습니다.

```
from sklearn.metrics import roc_curve

pred_positive_label = log_reg.predict_proba(X_test)[:,1]

fprs , tprs, thresholds = roc_curve(y_test,pred_positive_label)

print("샘플추출")
print()
thr_idx = np.arange(1,thresholds.shape[0],6)
print('thr idx:',thr_idx)
print("thr thresholds value:", thresholds[thr_idx])
print("thr thresholds value:", fprs[thr_idx])
print("thr thresholds value:", tprs[thr_idx])
```

```
샘플추출

thr idx: [  1   7  13  19  25  31  37  43  49  55  61  67  73  79  85  91  97 103]
thr thresholds value: [1.00000000e+00 9.98773119e-01 9.98052825e-01 9.74206495e-01
 6.92127253e-01 9.63972602e-03 5.70913190e-03 3.17272656e-03
 1.09753526e-03 8.24835534e-04 5.11718282e-04 4.46758783e-04
 1.56894093e-04 1.25286314e-04 7.84687840e-05 5.66573582e-05
 3.71954350e-05 1.86054057e-05]
thr thresholds value: [0.00101523 0.00101523 0.00101523 0.00101523 0.00203046 0.12791878
 0.1786802  0.26497462 0.43959391 0.50964467 0.58680203 0.61522843
 0.76954315 0.80203046 0.85279188 0.88832487 0.93096447 0.97258883]
thr thresholds value: [0.         0.52063492 0.59365079 0.86031746 0.9968254  0.9968254
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.        ]
```

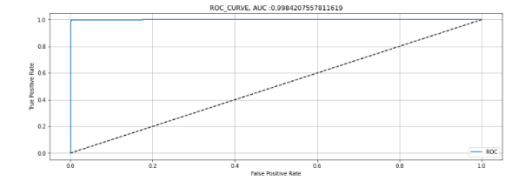
```
[79] from sklearn.metrics import roc_curve
     import matplotlib.pyplot as plt

     y_test_pred_probs = log_reg.predict(X_test)
     fprs, tprs, thresholds = roc_curve(y_test, pred)#각 클래스 확률예측값 계산

     y_test_pred_probs = log_reg.predict(X_test)

     FPR,TPR, _ = roc_curve(y_test, y_test_pred_probs)#진양성률과 위양성률수치 가져오기

     plt.plot(FPR, TPR)
     plt.plot([0,1],[0,1],'--', color='black') # 대각선
     plt.title('ROC Curve')
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.show()
     plt.clf()
```

다음으로, roc_curve와 roc_auc를 제시하는 과정을 진행해보았습니다. roc커브란 Receiver Operating Characteristic curve의 약자로 로지스틱 회귀 모형에서 모형의 성능과 예측력을 비교하는데 사용되는 곡선입니다. AUC는 ROC커브 아래면적을 의미합니다. ROC커브의 아래면적인 AUC가 1에 가까울수록 로지스틱 회귀 모형이 주어진 내용을 정확히 분류한 것으로 판단할 수 있습니다. 왼쪽과 같은 코드를 작성하여 아래그림과 같은 AUC 값을 구하고, ROC 커브를 그렸습니다. 그러나, y축 한계값이 너무 작아 변화가 잘 보이지 않아 이를 수정하였고, 제목에 AUC를 출력하는 코드를 추가하여 코드를 완성하였습니다.



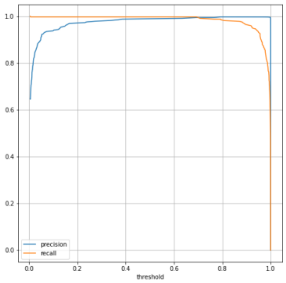
Sklearn.metrics 내장함수인 roc_curve와 roc_auc_score를 통해 목표값을 구하는 코드를 완성하였습니다. 완성된 코드와 커브는 왼쪽의 그림과 같습니다.

```
[88] pred_positive_label = log_reg.predict_proba(X_test)[:,1]

     fprs, tprs , thresholds = roc_curve(y_test, pred_positive_label)

     precisions, recalls, thresholds = roc_curve(y_test, pred_positive_label)

     plt.figure(figsize=(15,5))

     plt.plot([0,1],[0,1], '--', color='black') # 대각선

     plt.plot(fprs,tprs,label = "ROC")

     plt.title(f'ROC_CURVE, AUC :{roc_score}')
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.legend()
     plt.grid()
     plt.show()
```



마지막으로는, precision_recall_curve를 그렸습니다. Precision_recall커브는 파라미터인 threshold를 변화시키면서 precision과 recall값으르 플롯한 커브입니다. 실습시간에 배운 코드를 활용하여 코드를 구성하였으며, 마찬가지로 sklearn.metrics의 내장함수인 precision_recall_curve를 활용하여 코드를 구성하였습니다. 그려진 내용을 분석해본 결과, ROC커브와 는 달리 단조함수가 아니기 때문에 비교적 직관적이지 않아 값의 내용을 한눈에 파악하기 어렵다는 단점이 존재하다는 것을 알아낼 수 있었습니다.

```
[57] from sklearn.metrics import precision_recall_curve

     pred_proba_c1 = log_reg.predict_proba(X_test)[:,1]

     precisions, recalls, thresholds = precision_recall_curve(y_test, pred_proba_c1)

     plt.figure(figsize = (8, 8))
     plt.plot(thresholds, precisions[:len(thresholds)], label = 'precision')
     plt.plot(thresholds, recalls[:len(thresholds)], label = 'recall')
     plt.legend()
     plt.grid()
     plt.xlabel('threshold')
     plt.show()
```



이번 과제를 하면서 이름과 여러가지 파라미터들이 더 많이 존재해 어렵게만 느껴졌던 로지스틱회귀에 대해서 제대로 이해할 수 있는 시간을 가질 수있었습니다. 뿐만 아니라, 값을 바꿨을 때 어떻게 바뀌는지 스스로 분석하고 생각하는 과정을 가지며 컴퓨터적인 사고방식 또한 갖출 수 있는 경험을 할 수 있었습니다. 앞으로도 코딩을 실전적으로 스스로 경험하는 학습을 통해 알아나가야겠다고 느끼게 되었습니다.