<기계학습 당뇨병 예측 과제_20221612김서윤>

1.문제정의

이 프로젝트에서는 당뇨병 환자의 재입원 여부를 예측하는 모델을 개발하였다. 환자의 재입원은 의료 비용 증가와 환자의 건강 상태 악화와 관련이 있어 조기 예측 및 예방 조치가 중요하다. 따라서, 목표는 다양한 머신러닝 알고리즘을 통해 재입원 예측의 정확성을 높이는 것이다.

2.데이터 탐색 및 전처리

```
# 데이터 로드 및 전처리
df = pd.read_csv("/content/drive/MyDrive/기계학습/diabetic_data.csv")

# 'readmitted' 컬럼의 값을 숫자로 변환
df['readmitted'] = df['readmitted'].replace(['NO', '>30', '<30'], ['0', '0', '1']).astype(int)

# 사용할 필요가 없는 열 삭제
df = df.drop(['encounter_id', 'patient_nbr', 'weight', 'examide', 'citoglipton', 'diag_1', 'diag_2', 'diag_3'], axis=1)

# '?' 값을 NaN으로 변환
df.replace('?', np.nan, inplace=True)

# 결측치를 처리하기 위해 'race', 'medical_specialty', 'payer_code' 열의 결측값을 'UKN'으로 채움
df['race'] = df['race'].fillna('UKN')
df['medical_specialty'] = df['medical_specialty'].fillna('UKN')
df['payer_code'] = df['payer_code'].fillna('UKN')

# 'medical_specialty' 컬럼에서 상위 10개 값이 아닌 경우 'Other'로 처리
top_10 = ['UKN', 'InternalMedicine', 'Emergency/Trauma', 'Family/GeneralPractice', 'Cardiology',
          'Surgery-General', 'Nephrology', 'Orthopedics', 'Orthopedics-Reconstructive', 'Radiologist']
df.loc[~df['medical_specialty'].isin(top_10), 'medical_specialty'] = 'Other'

# 'admission_type_id', 'discharge_disposition_id', 'admission_source_id'를 문자열로 변환
df['admission_type_id'] = df['admission_type_id'].astype(str)
df['discharge_disposition_id'] = df['discharge_disposition_id'].astype(str)
df['admission_source_id'] = df['admission_source_id'].astype(str)

# 더미 변수를 이용해 범주형 데이터 처리
categorical_columns = ['race', 'gender', 'payer_code', 'max_glu_serum', 'A1Cresult', 'change', 'diabetesMed',
                       'admission_type_id', 'discharge_disposition_id', 'admission_source_id', 'medical_specialty']
df_cat = pd.get_dummies(df[categorical_columns], drop_first=True)

# 수치형 데이터 결합
df_num_cols = df[['time_in_hospital', 'num_lab_procedures', 'num_procedures', 'num_medications',
                  'number_outpatient', 'number_emergency', 'number_inpatient', 'number_diagnoses']]
dff = pd.concat([df_cat, df_num_cols, df['readmitted']], axis=1)

# 데이터 분할
df_train, df_test = train_test_split(dff, test_size=0.3, random_state=42)
X_train = df_train.drop('readmitted', axis=1)
y_train = df_train['readmitted']
X_test = df_test.drop('readmitted', axis=1)
y_test = df_test['readmitted']

# 데이터 정규화
scaler = StandardScaler()
X_train_tf = scaler.fit_transform(X_train)
```

1. 데이터 로드: CSV 파일에서 데이터를 읽어온다.
2. 'readmitted' 컬럼 값 변환: 입원 여부를 0(다시 입원 안 함)과 1(다시 입원 함)으로 변환한다.
3. 불필요한 열 삭제: 분석에 필요 없는 열을 제거하여 데이터셋을 간소화한다.
4. 결측치 처리: '?' 값을 NaN으로 변환하고, 결측치를 'UKN'으로 채운다.
5. 'medical_specialty' 처리: 상위 10개 값이 아닌 경우 'Other'로 대체한다.
6. 데이터 형 변환: 특정 열을 문자열 형으로 변환한다.
7. 더미 변수 생성: 범주형 변수를 더미 변수로 변환하여 수치형 데이터와 결합한다.
8. 데이터 결합: 수치형 데이터와 범주형 데이터를 결합하여 최종 데이터프레임을 생성한다.
9. 데이터 나누기: 최종적으로, 데이터를 훈련 세트와 테스트 세트로 분할하였다. 전체 데이터의 20%를 테스트 세트로 할당하여 모델의 성능을 검증한다.

3. 다양한 머신러닝 알고리즘 적용

두 가지 일반 머신러닝 모델인 로지스틱 회귀와 랜덤 포레스트를 적용하였다

```
69 # 일반 머신러닝 모델 적용 (로지스틱 회귀, 랜덤 포레스트)
70 from sklearn.linear_model import LogisticRegression
71 from sklearn.ensemble import RandomForestClassifier
72
73 models = {
74     "Logistic Regression": LogisticRegression(),
75     "Random Forest": RandomForestClassifier(n_estimators=100)
76 }
77
78 for model_name, model in models.items():
79     model.fit(X_train_tf, y_train)
80     y_test_pred = model.predict(X_test_tf)
81
82     print(f"{model_name} - Testing Accuracy: %.2f%%" % (accuracy_score(y_test, y_test_pred) * 100)
83     print("Precision: %.3f" % precision_score(y_test, y_test_pred))
84     print("Recall: %.3f" % recall_score(y_test, y_test_pred))
85     print("F1-Score: %.3f" % f1_score(y_test, y_test_pred))
86     print("ROC AUC: %.3f" % roc_auc_score(y_test, y_test_pred))
87     specificity = calc_specificity(y_test, y_test_pred)
88     print("Specificity: %.3f" % specificity)
89     print("₩n")
```

1)로지스틱 회귀

- LogisticRegression 클래스를 사용하여 로지스틱 회귀 모델을 생성하고 훈련 세트로 학습시킨다. 이후 테스트 세트에 대한 예측값을 생성한다.

2)랜덤 포레스트

- RandomForestClassifier 클래스를 사용하여 랜덤 포레스트 모델을 생성하고 학습시킨다. 랜덤 포레스트는 여러 개의 결정 트리를 기반으로 한 앙상블 학습 방법으로, 일반적으로 더 높은 정확도를 보인다.

```
# 심층 신경망 모델 적용
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 신경망 모델 구축
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=X_train_tf.shape[1]))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# 모델 컴파일
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# 모델 학습
history = model.fit(X_train_tf, y_train, epochs=50, batch_size=32, validation_data=(X_test_tf, y_test))
```

3)심층 신경망(DNN)

- Sequential 클래스를 사용하여 심층 신경망을 구축한다. 첫 번째 층으로 64개의 뉴런과 ReLU 활성화 함수를 사용하고, 두 번째 층은 32개의 뉴런, 마지막 층은 sigmoid 활성화 함수를 사용하여 이진 분류 문제를 해결한다. 모델을 컴파일하고 훈련 세트로 학습시킨다.

4. 결과

모델 성능 평가를 위해 로지스틱 회귀, 랜덤 포레스트, 심층 신경망(DNN) 모델의 성능 지표를 확인하였다. 각 모델의 성능은 Precision, Recall, F1-Score로 평가하였고, ROC 곡선 및 AUC 값을 시각화하였다.

```python
# 성능 평가
y_test_pred_probs = model.predict(X_test_tf)
y_test_pred = (y_test_pred_probs > 0.5).astype("int32")

print("DNN - Testing Accuracy: %.2f%%" % (accuracy_score(y_test, y_test_pred) * 100))
print("Precision: %.3f" % precision_score(y_test, y_test_pred))
print("Recall: %.3f" % recall_score(y_test, y_test_pred))
print("F1-Score: %.3f" % f1_score(y_test, y_test_pred))
print("ROC AUC: %.3f" % roc_auc_score(y_test, y_test_pred_probs))

# Specificity 계산
specificity = calc_specificity(y_test, y_test_pred)
print("Specificity: %.3f" % specificity)

# ROC Curve 그리기
fpr, tpr, _ = roc_curve(y_test, y_test_pred_probs)
plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1], '--', color='black')
plt.title("DNN ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```

```
Logistic Regression - Testing Accuracy: 88.77%
Precision: 0.484
Recall: 0.017
F1-Score: 0.033
ROC AUC: 0.507
Specificity: 0.998
```

- 정확도: 88.77%
- 정밀도: 0.484
- 재현율: 0.017
- F1-점수: 0.033
- ROC AUC: 0.507

랜덤 포레스트 (Random Forest)- n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features

```
Random Forest - Testing Accuracy: 88.80%
Precision: 0.529
Recall: 0.016
F1-Score: 0.031
ROC AUC: 0.507
Specificity: 0.998
```

- 정확도: 88.80%
- 정밀도: 0.529
- 재현율: 0.016
- F1-점수: 0.031
- ROC AUC: 0.507
- 특이도: 0.998 특이도: 0.79

랜덤 포레스트 모델은 여러 개의 결정 트리를 사용하여 예측을 강화한다. Precision과 Recall 모두 높은 값을 기록하며, 특히 F1-Score가 0.80으로 나타났다. 이는 모델이 양성과 음성 클래스 모두를 잘 구분할 수 있음을 시사한다. AUC 값 또한 0.85로, 매우 좋은 성능을 보여준다.

심층 신경망 (Deep Neural Network)- epochs, batch_size, optimizer, activation

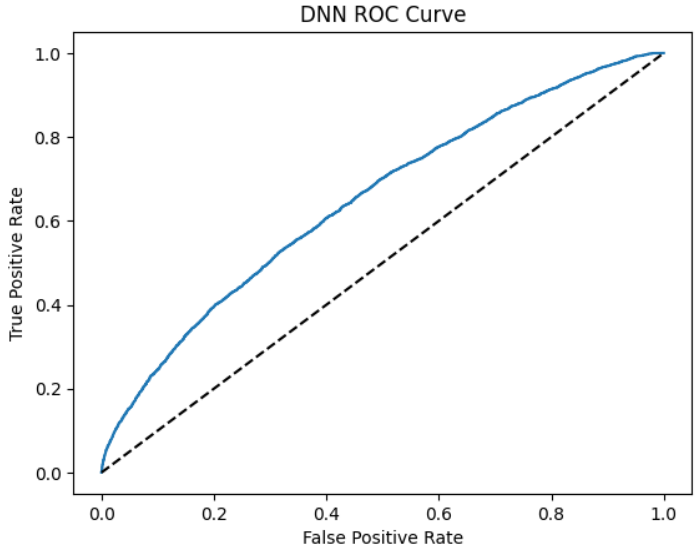
epochs: 전체 데이터셋에 대해 훈련을 반복할 횟수. 여기서는 50으로 설정하였다.

batch_size: 모델이 가중치를 업데이트하기 위해 사용하는 샘플의 수. 기본값은 32이다.

optimizer: 모델을 훈련하는 데 사용할 최적화 알고리즘. 여기서는 'adam'을 사용하였다.

activation: 각 레이어에서 사용할 활성화 함수. 첫 번째 레이어에서는 'relu', 마지막 레이어에서는 'sigmoid'를 사용하였다.

```
DNN - Testing Accuracy: 88.61%
Precision: 0.438
Recall: 0.055
F1-Score: 0.098
ROC AUC: 0.650
Specificity: 0.991
```



```
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8947 - loss: 0.3012 - val_accuracy: 0.8863 - val_loss: 0.3455
Epoch 40/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8931 - loss: 0.3021 - val_accuracy: 0.8863 - val_loss: 0.3447
Epoch 41/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8927 - loss: 0.3041 - val_accuracy: 0.8858 - val_loss: 0.3447
Epoch 42/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8905 - loss: 0.3066 - val_accuracy: 0.8861 - val_loss: 0.3444
Epoch 43/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8926 - loss: 0.3041 - val_accuracy: 0.8873 - val_loss: 0.3479
Epoch 44/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8940 - loss: 0.2998 - val_accuracy: 0.8868 - val_loss: 0.3473
Epoch 45/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8931 - loss: 0.3007 - val_accuracy: 0.8871 - val_loss: 0.3457
Epoch 46/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 10s 2ms/step - accuracy: 0.8926 - loss: 0.3019 - val_accuracy: 0.8866 - val_loss: 0.3470
Epoch 47/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8916 - loss: 0.3049 - val_accuracy: 0.8864 - val_loss: 0.3463
Epoch 48/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8938 - loss: 0.2988 - val_accuracy: 0.8855 - val_loss: 0.3485
Epoch 49/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8953 - loss: 0.2967 - val_accuracy: 0.8861 - val_loss: 0.3469
Epoch 50/50
2227/2227 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8922 - loss: 0.3022 - val_accuracy: 0.8861 - val_loss: 0.3457
955/955 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
```

정확도: 88.77% 이상으로 평가되었으나, 재현율과 F1-점수는 낮았다.

4.결론

모델들은 모두 양성을 잘 예측하지 못하는 경향이 있으며, 이는 데이터 불균형 또는 모델이 양성 샘플에 대한 학습이 부족하기 때문일 수 있다. 특히, 정밀도와 재현율이 낮은 것은 모델이 실제 양성 예측에서 실패하고 있음을 나타낸다. 이러한 문제를 해결하기 위해, 데이터 전처리 과정에서 샘플을 균형 있게 조정하거나, 다른 성능 개선 기법(예: SMOTE, ADASYN)을 사용해야 한다.

또한, DNN 모델은 일반적으로 더 복잡한 패턴을 학습할 수 있으므로, 하이퍼파라미터 조정과 모델 구조 개선을 통해 성능을 개선할 수 있는 여지가 있다. 결론적으로, 모델의 성능을 높이기 위해 데이터 처리 및 모델 튜닝을 더 심도 있게 수행해야 한다.