수상작 리뷰 보고서(11/11)

태그

[당뇨병 위험 분류 예측 AI 해커톤]

https://dacon.io/competitions/official/236068/codeshare/11596?page=1&dtype=recent

데이터

칼럼명

- Pregnancies : 임신횟수
- Glucose : 포도당 농도
- BloodPressure : 혈압
- SkinThickness : 피부 두께
- Insulin : 인슐린
- BMI : 체질량 지수
- DiabetesPedigreeFunction : 당뇨병 혈통 기능
- Age : 나이
- Outcome : 당뇨병 여부(0: 발병되지 않음, 1: 발병)

코드 흐름

1. **전처리 및 EDA 진행**

```
# isnull().sum()으로 확인했을 때 결측치가 없었으나 0으로 처리된 것들이
# 0을 결측치로 간주하고 확인
train_zero = train.replace(0, np.nan)

# 히트맵 시각화
plt.figure(figsize=(10, 6))
```

```
sns.heatmap(train_zero.isnull(), cbar=False, cmap='viridis')
plt.show()

# Pregnancies와 Outcome은 0이 결측치가 아니므로 무시.
# Glucose, BloodPressure, SkinThickness, Insulin, BMI만 처리
```

→ 결측값 0으로 처리, 이상치 존재시 중간값으로 변환

```
# 데이터 분할
X_train, X_val, y_train, y_val = train_test_split(X, y, test_
# 표준화
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
```

2. **모델 적용**

```
voting_clf = VotingClassifier(
    estimators=[
        ('rf', RandomForestClassifier(random_state=42)),
        ('xgb', XGBClassifier(random_state=42)),
        ('svc', SVC(probability=True, random_state=42))
    ],
    voting='soft'
)
```

→ RandomForestClassifier, XGBclassifier, SVC 진행

3. **K-Fold 적용**

```
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_stat
cross_val_scores = cross_val_score(voting_clf, X_train_scaled
print("Cross-validation ROC-AUC scores:", cross_val_scores)
print("Mean cross-validation ROC-AUC:", np.mean(cross_val_sco

voting_clf.fit(X_train_scaled, y_train)
```

결과:

```
Cross-validation ROC-AUC scores: [0.8079096  0.79728318 0.7
9989551 0.82183908 0.90836864]
Mean cross-validation ROC-AUC: 0.8270592024865842
```

4. **Pseudolabeling 기법 이용**

```
pseudo_labels = voting_clf.predict(X_test_scaled)
pseudo_data = pd.DataFrame(X_test_scaled, columns=X.columns)
pseudo_data['Outcome'] = pseudo_labels
# 훈련 데이터를 확장
X_exp = np.vstack((X_train_scaled, X_test_scaled))
y_exp = np.hstack((y_train, pseudo_labels))

# 확장된 데이터로 다시 모델 학습
voting_clf.fit(X_exp, y_exp)
```

5. **stacking 모델**

```
# Stacking 모델 구성
stacking_clf = StackingClassifier(
    estimators=[
        ('rf', RandomForestClassifier(random_state=42)),
        ('xgb', XGBClassifier(random_state=42)),
        ('svc', SVC(probability=True, random_state=42))
    ],
    final_estimator=LogisticRegression(),
    cv=kfold
)

stacking_clf.fit(X_exp, y_exp)
```

6. **Threshold 튜닝 진행**

```
# Threshold 튜닝
y_pred_prob = stacking_clf.predict_proba(X_val_scaled)[:, 1]
precision, recall, thresholds = precision_recall_curve(y_val,
# 최적의 F1 Score를 위한 임계값 찾기
```

```
f1_scores = 2 * (precision * recall) / (precision + recall)
best_threshold = thresholds[np.argmax(f1_scores)]
print("Best Threshold for F1 Score:", best_threshold)

# 최적 임계값을 이용한 최종 예측
y_pred_val = (y_pred_prob >= best_threshold).astype(int)
```

결과: Best Threshold for F1 Score: 0.28556815194303875

차별점, 배울점

1. 다양한 모델 적용: 한가지 모델이 아니라 정확성이 더 높은 결과를 찾기 위해 RandomForestClassifier, XGBClassifier, SVC와 같은 다양한 모델을 이용
2. 다른 분석과는 다르게 Pseudolabeling 기법을 이용했음. 이 것은 반지도 학습으로 라벨이 없는 데이터에 모델이 예측한 값을 임시 라벨로 삼아 모델에 추가 학습시키는 방법임. 데이터가 부족한 상황에서 모델의 성능을 향상시킬 수 있는 새로운 방법을 배울 수 있었음.
3. K-Fold, Stacking과 같은 다양한 기법을 시행해 과적합을 방지하고 모델 성능을 높임으로써 좀 더 예측력 높은 결과를 산출할 수 있었음.
4. Threshold 튜닝: 최종 예측을 결정하는 기준 값을 조정하는 작업을 의미, 이러한 방법에 대해 처음 접해보았고 0.5가 최적의 기준값이 아닐 수 있는 상황에서 좀 더 모델의 성능을 향상시킬 수 있는 새로운 방법에 대해 배우게 됨.