# <모델 학습 결과 보고서 (Isolation Forest)>

| 모델 학습 결과 보고서 (Isolation Forest) | |
| --- | --- |
| 작성자 | 김동훈 |
| 작성 일자 | 2024/10/14 |
| 팀명 | Sniffers |

| 순서 | 작업 내용 | 페이지 |
| --- | --- | --- |
| 1 | 데이터 로드 및 전처리 | 2 ~ 4 |
| 2 | Isolation Forest 모델 학습 | 5 |
| 3 | 모델 성능 평가 | 6 |
| 4 | 하이퍼파라미터 최적화 | 7 |
| 5 | 최종 모델 평가 및 저장 | 8 |

## <데이터 로드 및 전처리>

| 코드 | |
| --- | --- |
| x\_train = pd.read\_csv('./x\_train.csv')  y\_train = pd.read\_csv('./y\_train.csv')  x\_val = pd.read\_csv('./x\_val.csv')  y\_val = pd.read\_csv('./y\_val.csv')  x\_test = pd.read\_csv('./x\_test.csv')  y\_test = pd.read\_csv('./y\_test.csv')  x\_train = x\_train.drop(['merchant', 'category', 'first', 'last', 'city'], axis = 1)  x\_val = x\_val.drop(['merchant', 'category', 'first', 'last', 'city'], axis = 1)  x\_test = x\_test.drop(['merchant', 'category', 'first', 'last', 'city'], axis = 1) | |
| 결과 | |
| x\_train.describe()  x\_val.describe()    x\_test.describe()    y\_train.describe()    y\_val.describe()    y\_test.describe() | |
| 설명 | |
| 학습에 사용될 데이터셋을 로드한 후, 모델 학습에 필요 없는 ‘merchant’, ‘category’ 등의 컬럼을 제거한다. (테스트 시나리오의 학습에 불필요한 컬럼 제거) | |

## 

## <Isolation Forest 모델 학습>

| 코드 |
| --- |
| model = IsolationForest(contamination = 0.01,  n\_estimators = 100,  max\_samples = 'auto',  random\_state = 42)  model.fit(x\_train) |
| 결과 |
|  |
| 설명 |
| IsolationForest 모델을 사용하여 학습을 진행했다. contamination, n\_estimators, max\_samples 등의 하이퍼파라미터를 설정한다. |

## <모델 성능 평가>

| 코드 |
| --- |
| y\_pred = model.predict(x\_val)  print(classification\_report(y\_val, y\_pred))  roc\_auc = roc\_auc\_score(y\_val, y\_pred)  print(f"ROC AUC Score: {roc\_auc}") |
| 결과 |
|  |
| 설명 |
| 검증 데이터셋에 대한 예측을 통해 모델의 성능을 평가하고, ROC AUC 점수를 출력하여 이진 분류 문제에서의 성능을 측정한다. |

## <하이퍼파라미터 최적화>

| 코드 |
| --- |
| param\_grid = {  'contamination': [0.005, 0.01, 0.05],  'n\_estimators': [50, 100, 150],  'max\_samples': ['auto', 0.5, 0.8]  }  for contamination in param\_grid['contamination']:  for n\_estimator in param\_grid['n\_estimators']:  for max\_samples in param\_grid['max\_samples']:  model = IsolationForest(contamination = contamination,  n\_estimators = n\_estimator,  random\_state = 42)    y\_pred = model.fit\_predict(x\_train)  roc\_auc = roc\_auc\_score(y\_train, y\_pred)  if roc\_auc > best\_roc\_auc:  best\_roc\_auc = roc\_auc  best\_params = {'contamination': contamination,  'n\_estimators': n\_estimator,  'max\_samples': max\_samples}    print(f"Best parameters: {best\_params}")  print(f"Best ROC AUC score: {best\_roc\_auc}") |
| 결과 |
|  |
| 설명 |
| 여러 하이퍼파라미터 조합을 실험하여 최적의 하이퍼파라미터를 탐색한다. 각 조합에 대해 ROC AUC 점수를 계산하여 최고 성능을 나타내는 하이퍼파라미터를 도출한다. |

## <최종 모델 평가 및 저장>

| 코드 |
| --- |
| best\_model = IsolationForest(contamination = best\_params['contamination'],  n\_estimators = best\_params['n\_estimators'],  max\_samples = best\_params['max\_samples'],  random\_state = 42)  best\_model.fit(x\_train)  y\_pred = best\_model.predict(x\_test)  roc\_auc = roc\_auc\_score(y\_test, y\_pred)  print(f"ROC AUC score: {roc\_auc}")  with open('FDS\_model\_isolationForest.pkl', 'wb') as f:  pickle.dump(best\_model, f) |
| 결과 |
|  |
| 설명 |
| 최적화된 하이퍼파라미터로 최종 모델을 학습한 후, 테스트 데이터에 대한 성능을 평가하여 최종 ROC AUC 점수를 확인한다. 학습된 모델은 ‘.pkl’ 파일로 저장한다. |