비트코인 이상 거래 탐지

정보보호와 인공지능 기말 프로젝트 보고서

2023111394 정보보호학과 육은서

1. 데이터셋 선정 사유

1.1 데이터셋 개요

해당 보고서에서는 선정한 데이터셋은 Elliptic Data set이다. Elliptic은 암호화폐 포렌식 회사로, 암호화폐 트랜잭션에 대한 빅데이터를 수집하고 있다. 암호화폐는 최근 자금 세탁 경로로 많이 사용되고 있다. 사이버 범죄, 다크웹 거래 들의 통화가 이 암호화폐로 이루어지는 경우가 대부분이다.

이 데이터셋은 비트코인 네트워크 상에 있는 트랜잭션과 자금 흐름을 모아둔 데이터셋이다. Node는 단일 거래를 나타내고, Edge는 코인의 자금 흐름을 뜻한다. 비트코인의 자금 세탁은 여러 Node로 자금을 분산시키는 경우가 많기 때문에 이러한 데이터셋이 유효하다. 이 Feature들을 기반으로 해당 거래가 합법인지 불법인지를 식별한다. 위 같은 데이터셋은 AML 및 이상 거래 탐지(FDS) 구축에 사용된다.

2. 머신러닝

2.1. 전처리

'unknown' 데이터를 제거하고 불법은 1, 합법은 0으로 매핑하였다. 원본 데이터는 거래의 속성을 담은 features.csv와 정답 레이블을 담은 classes.csv로 나뉘어 있어, txId 기준으로 두 데이터를 병합했다. 이후 지도 학습을 수행하기 위해 레이블이 'unknown'인 데이터는 제거하였다.

```
[5 rows x 168 columns]
(46564, 168)
         txId  class  1         2         3         4          5         6  \
3   232438397      0  1  0.163054  1.963790 -0.646376  12.409294 -0.063725
9   232029206      0  1 -0.005027  0.578941 -0.091383   4.380281 -0.063725
10  232344069      0  1 -0.147852 -0.184668 -1.201369  -0.121970 -0.043875
11   27553029      0  1 -0.151357 -0.184668 -1.201369  -0.121970 -0.043875
16    3881097      0  1 -0.172306 -0.184668 -1.201369   0.028105 -0.043875

           7         8  ...       157       158       159       160  \
3   9.782742  12.414558  ... -0.577099 -0.613614  0.241128  0.241406
9   4.667146   0.851305  ... -0.577099 -0.613614  0.241128  0.241406
10 -0.113002  -0.061584  ... -0.577099 -0.613614  0.241128  0.241406
11 -0.113002  -0.061584  ... -0.539735 -0.582077 -0.979074 -0.978556
16 -0.029140   0.242712  ... -0.577099 -0.600999  0.241128  0.241406

         161       162       163       164       165       166
3   1.072793  0.085530 -0.131155  0.677799 -0.120613 -0.119792
9   0.604120  0.008632 -0.131155  0.333211 -0.120613 -0.119792
10  0.018279 -0.087490 -0.131155 -0.097524 -0.120613 -0.119792
11  0.018279 -0.087490 -0.131155 -0.097524 -0.120613 -0.119792
16  0.018279 -0.068266 -0.084674 -0.054450 -1.760926 -1.760984

[5 rows x 168 columns]
```

166개의 피처 값의 범위가 서로 달라 평균 0, 분산 1을 갖도록 StandardScaler를 하였다. 모델의 과적합을 방지하기 위해 연산 효율성을 높이기 위해 PCA을 수행하여 상위 50개의 주성분만 추출했다.

2.2. 모델 설계

단일 모델보다 과적합 위험이 적고, 보안 데이터와 같이 노이즈가 많고 불균형한 데이터에서도 안정적인 성능을 보장하기 때문에 선정하였다. 파라미터는 기본값으로 n_estimators=100, max_depth=20, random_state=42, n_jobs=-1로 설정하였다.

2.3. 훈련 및 평가

분할전체 데이터를 학습용 70%, 테스트용 30%로 분리했다(test_size=0.3) 추가적으로 stratify=y 옵션을 사용하여 학습 및 테스트 데이터셋에 불법 거래의 비율이 동일하게 유지되도록 했다.

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     12606
           1       0.97      0.74      0.84      1364

    accuracy                           0.97     13970
   macro avg       0.97      0.87      0.91     13970
weighted avg       0.97      0.97      0.97     13970
```

전체 데이터 중 약 97%를 정확하게 분류했다.

2.4. 성능향상을 위한 노력

단일 트랜잭션 거래만으로는 자금 세탁 이상 거래라고 판별하기에 무리인 점들이 많다. 위 그림은 실제 스캠 사기에서 생긴 비트코인 자금 흐름을 나타낸 그래프이다. 여러 제3의 지갑으로 흘러서 분산되는 형태이기 때문에 단순 거래 자체뿐 아니라 네트워크 특징도 고려해야 한다.

때문에 단순 거래 피처 학습이 아닌 네트워크 구조의 특성을 추가하여 실험을 재구성하였다. 이부분에서는 NetworkX라는 라이브러리가 필요하다. 거래 관계를 Graph로 모델링하는 라이브러리이다. 네트워크 구조의 특성에서는 다음 데이터가 필요하다.

- In-degree : 자금을 얼마나 많은 곳에서 받았는지

- Out-degree : 자금을 얼마나 많은 곳으로 보냈는지

- PageRank : 거래 네트워크에서 해당 노드의 중요도

	precision	recall	f1-score	support
0	0.99	1.00	0.99	12606
1	1.00	0.87	0.93	1364
accuracy			0.99	13970
macro avg	0.99	0.93	0.96	13970
weighted avg	0.99	0.99	0.99	13970

이전보다 더 나은 정확도로 개선되었다.

3. 딥러닝

3.1. 전처리

훈련, 검증, 테스트는 6:2:2 비율로 진행하였고, StandarScaler로 정규화 과정을 거쳤다.

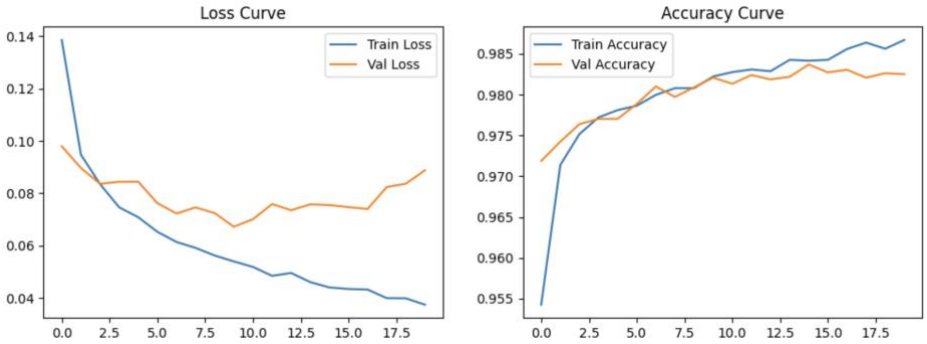
3.2. 모델 설계

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 256)	42,752
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 1)	65

Total params: 83,969 (328.00 KB)
Trainable params: 83,969 (328.00 KB)
Non-trainable params: 0 (0.00 B)

위 그림에서 Input을 제외하고 3개의 은닉층과 하나의 출력층으로 이루어진 것을 확인할 수 있다. 은닉층 간에는 Dropout을 삽입하였다. 은닉층은 ReLU 함수를 사용하고, 출력층은 Sigmoid 함수를 사용하였다.

3.3. 훈련 및 평가



	precision	recall	f1-score	support
0	0.98	1.00	0.99	8404
1	0.97	0.84	0.90	909
accuracy			0.98	9313
macro avg	0.98	0.92	0.94	9313
weighted avg	0.98	0.98	0.98	9313

3.4. 성능향상을 위한 노력

드롭아웃: 은닉층 뒤에 Dropout(0.3)을 배치하여 뉴런의 30%를 무작위로 비활성화함으로써, 특정 뉴런에 의존하는 현상을 방지하였다.

조기 종료: patience=10으로 설정하여, 검증 손실이 10 에포크 동안 개선되지 않으면 학습을 즉시 중단시켰다.

모델 체크포인트: 훈련 중 가장 성능이 좋았던 모델 가중치만 저장하여, 학습 종료 후 최상의 모델을 불러올 수 있게 설정했다.

4. 코드리뷰

4.1. RandomForestClassifier에서 사용한 NetworkX 데이터 생성

```
# networkX로 거래 네트워크 생성한 데이터셋 구축
classes = pd.read_csv('elliptic_bitcoin_dataset/elliptic_txs_classes.csv')
edgelist = pd.read_csv('elliptic_bitcoin_dataset/elliptic_txs_edgelist.csv')
features = pd.read_csv('elliptic_bitcoin_dataset/elliptic_txs_features.csv', header=None)

features.rename(columns={0: 'txId'}, inplace=True)

classes = classes[classes['class'] != 'unknown'].copy()
classes['class'] = classes['class'].map({'1': 1, '2': 0})

G = nx.from_pandas_edgelist(edgelist, 'txId1', 'txId2', create_using=nx.DiGraph())

target_nodes = set(classes['txId'])

# 구조적 특징 1: 자금의 유입/유출량 척도
in_degree = nx.in_degree_centrality(G)
out_degree = nx.out_degree_centrality(G)

# 구조적 특징 2: PageRank(네트워크 내 중요도)
pagerank = nx.pagerank(G, alpha=0.85)
```

```
graph_features = pd.DataFrame({
    'txId': list(G.nodes()),
    'in_degree': [in_degree.get(n, 0) for n in G.nodes()],
    'out_degree': [out_degree.get(n, 0) for n in G.nodes()],
    'pagerank': [pagerank.get(n, 0) for n in G.nodes()]
})
```

```
data = pd.merge(classes, features, on='txId', how='left')
data = pd.merge(data, graph_features, on='txId', how='left')
```

Csv를 불러온 후 edgelist에 있는 거래 연결 관계로 Graph를 구성한다. 이후 Graph에서 유입이 얼마나 많이 되었는가, 많은 곳으로 보내었는가에 대한 특징을 추출한다. 또한, pagerank는 척도(횟수)가 아니더라도 중요도가 높은 노드와 관련되었다면 내 노드 또한 중요도가 올라간다. 이러한 중요도는 단순 거래만 보는 것이 아니라 네트워크 그래프를 그렸을 때 계산이 가능하다.

4.2. DNN에서 사용한 모델 설계 및 성능향상을 위한 노력

```python
# 모델 설계
def build_model(input_shape):
    model = keras.Sequential([
        layers.Dense(256, activation='relu', input_shape=(input_shape,)),
        layers.Dropout(0.3),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(64, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])
    return model

model = build_model(X_train_scaled.shape[1])
model.summary()

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy', 'AUC'])
```

딥러닝 모델로 위와 같이 구성하였다. 각 은닉층에는 activation='relu'를 적용하였고, 256,128,64로 레이어가 줄어든다. Dropout은 0.3으로 적용하였으며, 마지막 Dense는 출력층을 의미한다. Compile은 adam 옵티마이저를 사용하였다.

```python
# Early Stopping
early_stopping_cb = callbacks.EarlyStopping(
    patience=10,
    restore_best_weights=True,
    monitor='val_loss'
)

# Model Checkpoint
checkpoint_cb = callbacks.ModelCheckpoint(
    "best_fraud_detection_model.keras",
    save_best_only=True,
    monitor='val_loss'
)
```

성능향상을 위해 조기 종료와 체크포인트도 도입하였다.

5. 데이터셋 URL, colab URL

Elliptic Data Set, Bitcoin Transaction Graph,
https://www.kaggle.com/datasets/ellipticco/elliptic-data-set

https://colab.research.google.com/github/6kitty/IP01069/blob/main/기말프로젝트_2023111394_육은서.ipynb