The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 24, No. 2, pp.177-184, Apr. 30, 2024. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2024.24.2.177
JIIBC 2024-2-26

LIME과 SHAP 모델 공유에 의한 모델 해석

Model Interpretation through LIME and SHAP Model Sharing

김용길*

Yong-Gil Kim*

요 약 데이터가 빠른 속도로 증가하고 있는 가운데 가능한 최고의 정확도를 달성하기 위해 모든 종류의 복잡한 앙상블 및 딥 러닝 알고리즘이 사용되고 있다. 그렇지만, 이러한 모델이 알 수 없는 데이터를 예측/분류/인식/추적하는 방법과 관련하여 예측, 분류, 인식, 추적이 항상 신뢰할 수 있는 것은 아니다. 데이터 부족, 불균형 데이터 세트, 편향된 데이터 세트 등과 같은 다양한 이유가 학습 모델에 의해 포착되는 결정에 영향을 미칠 수 있다. 이와 관련하여 현재 모델의 설명 가능성에 관한 연구가 관심을 끌고 있다. 현재 설명 가능성 기법과 관련하여 LIME과 SHAP가 보편적으로 사용되고 있지만, 출력 결과들은 다소 상이한 측면을 나타내고 있다. 이에 본 연구에서는 LIME과 SHAP을 결합하는 방식을 소개하고, 데모와 관련해서 IEEE CIS 데이터 세트에서 거래를 사기로 분류할 때 LightGBM 및 Keras 모델이 내린 결정에 대한 설명 가능성을 분석한다.

Abstract In the situation of increasing data at fast speed, we use all kinds of complex ensemble and deep learning algorithms to get the highest accuracy. It's sometimes questionable how these models predict, classify, recognize, and track unknown data. Accomplishing this technique and more has been and would be the goal of intensive research and development in the data science community. A variety of reasons, such as lack of data, imbalanced data, biased data can impact the decision rendered by the learning models. Many models are gaining traction for such interpretations. Now, LIME and SHAP are commonly used, in which are two state of the art open source explainable techniques. However, their outputs represent some different results. In this context, this study introduces a coupling technique of LIME and Shap, and demonstrates analysis possibilities on the decisions made by LightGBM and Keras models in classifying a transaction for fraudulence on the IEEE CIS dataset.

Key Words : LIME, SHAP, LightGBM, Keras models

Ⅰ. 서 론

가까운 미래에 AI 모델이 소송, 긴급하지 않은 환자 치료 또는 시나리오 작성과 같은 의사 결정 작업을 대신할 것으로 예상할 수 있다. 그러나 거기에 도달하기 전에 결정에 도달한 방법과 이유를 철저히 이해할 필요가 있다. 인간과 달리 AI 모델은 의사 결정 과정의 동기를 이해하기 위해 인터뷰할 수 없으며, 질문을 할 수 있다 하

*정회원, 조선이공대학교 컴퓨터보안과
접수일자 2024년 2월 8일, 수정완료 2024년 3월 8일
게재확정일자 2024년 4월 5일

Received: 8 February, 2024 / Revised: 8 March, 2024 / Accepted: 5 April, 2024
*Corresponding Author: ygkim@cst.ac.kr
Department of computer security, chosun college of science & technology, korea

더라도 과거에 내린 결정에 대한 책임을 물을 수 없다. 이러한 이유로 모델 예측을 설명하는 방법이 중요하다. 입력 변수 중 하나의 약간의 증가가 선형 회귀 모델의 예측에 어떤 영향을 미치는지 이해하기 쉽지만, 현재 특징과 결과 간의 관계를 선형 모델로 정확하게 예측할 수 없는 경우가 많아서 더 복잡한 알고리즘으로 이동해야 한다. 이러한 복잡한 모델이 단순 모델보다 더 정확한 예측을 나타내지만, 모델 예측의 논리를 이해하는 것은 모델의 복잡성이 증가함에 따라 점점 더 어려워진다.

기계 학습 모델을 작업하는 동안 특정 측도를 사용하여 모델 성능을 측정하는 경우가 많다. 그렇지만, 이러한 추정 관련 측도는 모델을 서로 비교할 때 몇 가지 설명할 수 없는 문제점들이 있다. 구체적으로 모델 복잡성의 설명, 특정 특징의 중요성 문제, 포착된 특징이 잘못된 이유, 모델에 의도하지 않은 편향 등이다. 더 나아가서, 모델의 블랙박스 특성을 실제로 이해하지 못한 채 테스트 데이터에 대한 최상의 측도를 얻기 위해 하이퍼 매개변수를 변경하는 경향이 있다. 불행하게도 대부분의 기계 학습 모델을 직접 해석하는 것은 불가능하다. Random Forest, Gradient Boosted Machine 및 신경망과 같은 인기 있는 모델의 경우 모델에 구애받지 않는 방법이 필요하다. 순열 특징 중요도, 부분 종속성 플롯(PDP), 개별 조건부 기대(ICE) 플롯, 글로벌 대리 모델 등과 같은 기법이 사용될 수 있지만, 현재 LIME(Local Interpretable Model-agnostic Explanations)과 SHAP(SHapley Additive exPlanations)이 보편적으로 사용되고 있다.

본 연구에서는 SHAP와 LIME 라이브러리의 출력 해석 방법에 대한 통찰력을 제공하여 해석 가능 인공 지능 측면에서 모델 설명을 생성할 수 있도록 하는 데에 있다. LIME은 설명 가능한 모델에 적합하도록 교란된 데이터 세트를 생성하는 반면에 SHAP는 SHAP 값을 계산하기 위해 전체 표본이 필요하다. 즉, LIME에는 하나의 관찰만 필요하지만, SHAP에는 여러 관찰이 필요하다. 즉, 표본마다 SHAP 값이 약간씩 다르다. LIME과 SHAP는 모델에 구애받지 않지만 두 라이브러리 모두 특정 구조를 가진 모델이 필요하다. LIME은 SHAP보다 더 유연한 측면이 있지만, SHAP에는 더 구체적인 구조가 필요하다. 이러한 문제점 등으로 인해 LIME과 SHAP의 출력 결과가 다소 상이한 측면을 보인다. 이에 본 연구에서는 SHAP와 LIME 라이브러리의 몇 가지 장단점을 강조하는 코드 및 출력을 나타내고, 두 모델의 공유를 통해 이러한 문제점들을 개선하는 방식을 소개한다. 데모와 관련해서는 IEEE CIS 데이터 세트를 통해 SHAP와 LIME의 코드 비교 및 결합을 나타낸다.

II. 관련 연구

특징 선택 또는 특징 간의 관계를 설명하기 위한 몇 가지 일반적인 접근 방식이 반드시 특징 중요도를 포착하는 것은 아니다. 귀무가설 검정 방법은 관계를 식별할 수 있지만, 관계의 강도를 설명하지는 않는다. 마찬가지로 Lasso와 같은 희소 모델 적합 알고리즘에 특징이 포함되어 있는지 확인하는 것은 특징이 의존하는 정도를 설명하지 않는다[1]. 부분 의존도는 여러 변수에 관심이 있거나 예측 모델에 상호 작용이 있는 경우 해석하기 어려울 수 있다[2]. 또 다른 일반적인 특징 중요도 포착 절차는 먼저 모든 데이터에 대해 모델 적합 알고리즘을 두 번 실행한 다음 데이터 세트에서 관련 특징을 제거한 후 다시 실행하는 것이다. 그런 다음 두 결과 모델의 손실을 비교하여 특정 특징의 중요성 또는 필요조건을 결정한다[3]. 이 측정은 하나가 아닌 두 예측 모델의 함수이므로 각 모델이 특정 특징에 얼마나 의존하는지 측정하지 않는다.

RF(Random Forest)의 순열 기반 특징 중요도 측정은 모델 의존도 정의에 영감을 준다. 이러한 접근 방식은 경험적 연구의 주제였으며, 측정의 여러 변형이 제안되었다[4]. Mentch와 Hooker는 U-통계를 사용하여 RF에서 사용되는 부트스트랩 집계와 유사하게 하위 표본에 맞는 앙상블 모델의 예측을 연구했으며[5], 또 다른 RF에서 파생된 특징 중요도 측도인 MDI(Mean Difference Impurity)가 소개되었다[6]. 이 모든 문헌은 RF, 앙상블 또는 개별 트리에 대한 특징 중요도 측정에 중점을 둔다. 변수 중요도 측정과 관련하여 VIMP는 해당 예측 문제를 해결하는 데 사용되는 예측 모델에서 예측 변수의 관련성을 정량화하도록 설계되었다. 그렇지만, 일부 VIMP는 계산 비용이 상당히 낮아서 높이 평가되지만, 기존 척도가 편향된 것으로 밝혀져 여러 가지 개선이 이루어졌다[7][8].

현재 LIME과 SHAP는 모든 모델에 대한 지역적인 설명을 만드는 데 사용할 수 있다[9]. 즉, 텍스트, 이미지 또는 표 형식 데이터를 사용하는 모델에서 만든 예측을 설명하기 위해 두 가지 방법의 하나를 사용할 수 있다. LIME은 지역적인 설명을 생성하기 위해 예측 주위에 간단한 모델을 맞춘다[10]. 이에 반해 SHAP는 게임 이론을 사용하여 각 특징의 중요성을 측정한다[11]. 결국 두 가지 방법 모두 기능이 예측 결과에 미치는 영향을 이해하는

데 사용되지만 각 방법은 약간 다른 절차를 사용한다[12]. LIME은 로컬 모델을 사용하여 모든 모델의 예측을 설명하는 절차이다. LIME은 설명하려는 모델이 일련의 특징을 사용하여 예측한다는 가정하에 작동한다. 또한 필요한 만큼 자주 모델을 조사할 수 있다고 가정한다. 절차는 단일 관찰을 약간 수정하여 생성된 데이터 세트에 해석 가능한 모델을 맞추는 것으로 구성된다. 데이터 세트는 단일 관찰에서 모든 특징을 가져온 다음 일부 구성 요소를 반복적으로 제거하여 새로운 교란된 관찰 집합을 생성하는 방식으로 구축된다. 모델은 교란된 데이터 세트에서 관찰을 처리한 다음 해석 가능한 모델이 최신 데이터에 적합된다.

SHAP는 LIME과 마찬가지로 블랙박스 모델의 예측에 대한 로컬 설명을 제공하는데, 주요 차이점은 SHAP가 게임 이론을 사용하여 지역적인 설명을 한다는 것이다. Shapley 값을 함께 추가하는 것을 기반으로 하는데 (Shapley 값은 게임에서 한 플레이어의 기여도에 대한 예상치), 게임 이론과 기계 학습을 연결하려면 모델을 게임의 규칙으로 간주하여 특징을 관찰할 수 있거나 그렇지 않은 플레이어로 생각해야 한다. SHAP 값을 계산하려면, 필요한 만큼 여러 번 조사할 수 있는 모델이어야 하고, 관찰 표본 및 모델에 대한 구체적인 구조(SHAP와 호환 가능)가 있어야 한다. LIME에서는 설명을 생성하기 위해 하나의 관찰이 필요했지만, Shapley 값이 계산되는 방식으로 인해 SHAP를 사용하여 단일 관찰을 설명하려면 전체 표본이 필요하다. 구체적으로 입력 변수 $X = [X_1, X_2, \dots, X_p]$을 사용하여 $f(X)$을 예측하는 적합 모델 f가 있다고 가정한다. SHAP의 목표는 각 특징이 $X = x$에서 모델의 출력에 얼마나 공헌했는지 측정하여 예측 $f(x)$을 설명하는 것이다.

Ⅲ. SHAP 최적화와 LIME 모델 공유

LIME은 설명 가능한 모델에 적합하도록 교란된 데이터 세트를 생성하는 반면에 SHAP는 SHAP 값을 계산하기 위해 전체 표본이 필요하다. 즉, LIME에는 하나의 관찰만 필요하지만, SHAP에는 여러 관찰이 필요하다. SHAP 값은 표본의 평균 예측값에 상대적이다. 즉, 표본마다 SHAP 값이 약간씩 다르다. LIME과 SHAP는 모델에 구애받지 않지만 두 라이브러리 모두 특정 구조를 가진 모델이 필요하다. LIME은 predict_proba의 메서드만 필요하므로 약간 더 유연한 측면이 있다. SHAP에는 더 구체적인 구조가 필요하지만, 이 방법은 sklearn, xgboost, tensorflow 등을 포함하여 가장 널리 사용되는 AI 라이브러리에서 작동한다. 주의 사항으로 LIME 또는 SHAP는 성능 측도를 대체할 수 없다는 점이다.

평균 제곱 오차, 평균 절대 오차, ROC 곡선 아래 영역, F1 점수, 정확도 및 기타 성능 측도는 모델의 적합도를 평가한다. 반면에 LIME 및 SHAP는 모델의 예측에 대한 로컬 설명을 제공한다. 즉, 이러ㄴ한 방법은 예측의 품질을 설명하기 위한 것이 아니라 예측이 이루어진 이유를 알려준다. 예를 들어 주어진 예측이 부정확하다는 것을 알고 있는 경우(예를 들어, 거짓 음성) 어떤 구성 요소가 해당 관찰을 잘못 분류하도록 모델을 구동했는지 이해하기 위해 두 가지 방법 중에 하나를 사용할 수 있다. 이렇게 하면 모델의 성능이 저하되고 궁극적으로 더 나은 성능 측도로 이어질 수 있는 에지 경우를 식별하는 데 도움이 된다(물론 모델을 재훈련하는 경우).

유의해야 할 또 다른 중요한 고려 사항은 모델이 기본 데이터 생성 프로세스의 근사치라는 것이다. 따라서 LIME 또는 SHAP에 의한 설명은 해당 구성 요소가 대상 변수에 인과 관계가 있음을 의미하지 않는다. 설명은 모델이 특정 예측을 한 이유만을 알려준다. 이러한 관계가 기능과 대상 간에 반영된다는 것을 반드시 의미하지는 않는다. 참고로 선형 회귀는 간단한 학습 알고리즘으로 기본적으로 예측을 수행하기 위해 데이터 세트에 가장 적합한 라인을 찾는다. 트리 기반 그래디언트 부스트 앙상블은 순차적으로 훈련되는 일련의 결정 트리(즉, 앙상블)이다. 앙상블의 각 트리는 점점 더 정확한 모델을 생성하기 위해 이전 모델이 만든 오류에 초점을 맞춘다. 자연어 처리는 텍스트, 음성 및 기타 형태의 언어 기반 커뮤니케이션을 통한 학습에 중점을 둔 기계 학습의 한 분야이다. 이에 반해 Shapley 값은 효율성, 대칭, 더미 및 가산 공리를 기반으로 한다. 결과적으로 SHAP 값은 자연스럽게 추가된다.

SHAP는 모델 기능 영향 점수에 대한 Shapley 값의 개념을 활용한다. Shapley 값의 기술적 정의는 모든 가능한 연합에 대한 특징값의 평균 한계 기여도이다. 즉, Shapley 값은 가능한 모든 입력 조합을 사용하여 인스턴스에 대해 가능한 모든 예측을 고려한다. 이러한 접근 방식을 사용함으로 SHAP는 일관성 및 로컬 정확도와 같은 속성을 보장할 수 있다. 그렇지만, 안타깝게도 SHAP는 모든 모델 유형에 최적화되어 있지 않다. 이에 반해 LIME은 블랙박스 모델이 해당 지역 주변에서 어떻게 작동하는지 설명하기 위해 각 예측 주위에 희소 선형 모델

을 구축한다. LIME은 빠르지만, Shapley 값을 계산하는데 오랜 시간이 걸리는 문제점이 있다.

SHAP에는 그래디언트 부스트 트리와 같이 트리에서 빠르게 실행되는 트리 설명자가 있다. XGBoostscikit-learn 및 scikit-learn의 Random Forest가 있지만, k-최근접 이웃과 같은 모델의 경우 매우 작은 데이터 세트에서도 엄청나게 느리다. 예를 들어, knn 모델에서 Boston Housing 데이터 세트를 기반으로 SHAP를 실행하는데 1시간이 넘게 걸린다. 해결책으로 먼저 k-평균 알고리즘으로 데이터를 요약하여 정확도와 신뢰성을 희생하면 몇 분으로 줄일 수 있지만, 모델 출력이 정확하지 않으며 안정적이지 않다. 아래의 코드와 주석은 SHAP 라이브러리의 이러한 결함을 나타낸다. 그렇지만, SHAP 최적화와 함께 LIME 모델을 사용하면 작업이 매우 빠르게 실행되고 출력이 정확하고 안정적이다.

```
# Boston Housing Data 가져오기
X,y = shap.datasets.boston()
X_train,X_test, y_train,y_test = train_test_split(X, y,
    test_size=0.2, random_state=0)
X,y = shap.datasets.boston()
X_train,X_test,y_train,y_test = train_test_split(X, y,
    test_size=0.2, random_state=0)

# K-최근접 이웃
knn = sklearn.neighbors.KNeighborsRegressor()
knn.fit(X_train, y_train)

# SHAP 설명자 생성
# SHAP가 갖는 설명자: deep, gradient, kernel,
#        linear, tree, sampling
# knn 상에서 Kernel 메서드를 사용해야 함.
# k-Means를 사용한 데이터 요약은 처리 가속화를
# 위한 기법에 해당함.

"""기대치 추정을 위해 전체 훈련 집합을 사용하는
  대신에, 가중된 kmeans 집합으로 요약(각 가중치는
  데이터 개수)"""
# kmeans 요약 구축
X_train_summary = shap.kmeans(X_train, 10)

# kmeans 요약 사용
t0 = time.time()
explainerKNN =
shap.KernelExplainer(knn.predict,X_train_summary)
shap_values_KNN_test =
explainerKNN.shap_values(X_test)
t1 = time.time()
timeit=t1-t0timeit

# without kmeans를 사용하지 않으면
# 대략 4000초가 소요됨.
"""t0 = time.time()
    explainerKNN =
shap.KernelExplainer(knn.predict, X_train)
    shap_values_KNN_test =
explainerKNN.shap_values(X_test)
    t1 = time.time()  timeit=t1-t0 timeit"""

# SHAP 설명자 가시화
shap.force_plot(explainerKNN.expected_value,
     shap_values_KNN_test[j], X_test.iloc[[j]])
```

다른 접근 방식으로는 LIME을 사용하는 것인데, LIME은 같은 knn 모델로 즉시 실행되며 k-평균으로 요약할 필요가 없다. 문제는 LIME의 출력이 특정 특징들에 관해 SHAP 출력과 다르다는 점이다. 특히, LIME이 Shapley 값과 같은 정확도 및 일관성 속성을 갖지 않고 SHAP가 영향 점수를 계산하기 전에 k-평균 요약을 사용하면 구분하기 어렵다. 요컨대, LIME은 knn 모델 예에서 좋은 대안을 제공하지만, 불행히도 LIME이 항상 문제를 해결하지는 못한다. 모든 모델에서 기본적으로 작동하지 않는다. 예를 들어, LIME은 입력 데이터에서 xgb.DMatrix()를 사용하기 위한 XGBoost의 요구 사항을 처리할 수 없다. 요점은 LIME이 XGBoost 라이브러리와 자동으로 작동하지 않는다는 것이다. 반면 SHAP는 XGBoost에 최적화되어 빠르고 안정적인 결과를 제공한다. 다음 코드는 개선된 방식으로 매우 빠르게 실행되며, Shapley 값의 추정치를 찾기 위해 XGBoost 트리를 추적하도록 최적화된 SHAP 라이브러리의 TreeExplainer를 사용한다.

```
explainerXGB = shap.TreeExplainer(xgb_model)
shap_values_XGB_test =
explainerXGB.shap_values(X_test)
shap.force_plot(explainerXGB.expected_value,
    shap_values_XGB_test[j], X_test.iloc[[j]])
```

Ⅳ. 비교 실험

본 연구의 비교 실험에서는 IEEE CIS 데이터 세트에서 거래를 사기로 분류할 때 LightGBM 및 Keras 모델이 내린 결정에 대한 설명 가능성을 나타내기로 한다. 데이터 세트는 바이너리 타겟 isFraud로 표시되는 온라인 거래가 사기일 확률 예측에 사용된다. 데이터는 두 개의 파일 identity와 transaction로 구성되는데, 이 파일들은 TransactionID에 의해 결합된다. 모든 거래에 해당 identity 정보가 있는 것은 아니다. transaction의 범주형 특징은 ProductCD, card1-card6, addr1, addr2, P_emaildomain, R_emaildomain 및 M1-M9이다. identity의 범주형 특징은 DeviceType, DeviceInfo, id_12-id_38이다. 여기에서는 특징 크기 조정을 사용하여 특징값 크기의 가변성을 균등화했다. 먼저 앙상블 모델로서 LightGBM 모델 코드 사용과 분류 결과를 얻기 위한 코드와 결과는 표 1과 같다.

```
# 훈련 및 검토 집합 생성
x_train, x_test, y_train, y_test =
  train_test_split(x, y, test_size=0.30,
random_state=42)
# 모델 훈련
parameters = {
    'application': 'binary', 'objective': 'binary',
    'metric': 'auc', 'is_unbalance': 'true',
    'boosting': 'gbdt', 'num_leaves': 31,
    'feature_fraction': 0.5, 'bagging_fraction': 0.5,
    'bagging_freq': 20, 'learning_rate': 0.05,
    'verbose': 0
}
model = lightgbm.train(parameters, train_data,
    valid_sets=test_data, num_boost_round=5000,
        early_stopping_rounds=100)
y_pred = model.predict(x_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_bool))
```

표 1. LightGBM 모델 분류 결과
Table 1. LightGBM Model Classification Results

	precision	recall	f1-score	support
0	0.99	0.99	0.99	170821
1	0.67	0.84	0.75	6341
accuracy			0.98	177162
macro avg	0.83	0.91	0.87	177162
weighted avg	0.98	0.98	0.98	177162

모델 분류 결과 설명을 위한 공식적인 LIME 및 SHAP 설명 가능성 기법을 살펴보기에 앞서, 모델이 특정 분류로 수렴되도록 하는 20가지 주요 특징들을 시각적으로 이해할 필요가 있다. 이를 위한 코드는 아래와 같이 작성할 수 있고, 출력 결과는 그림 1과 같다. 즉, 모델을 가능하게 하는 상위 20가지 중요한 특징들의 시각화이다.

```
feature_imp= pd.DataFrame({
    'Value':model.feature_importance(),
    'Feature':X.columns})
plt.figure(figsize=(40, 20))
sns.set(font_scale = 5)
sns.barplot(x="Value", y="Feature",
  data=feature_imp.sort_values(by="Value",
        ascending=False)[0:20])
plt.title('LightGBM Features (avg over folds)')
plt.tight_layout()
plt.savefig('lgbm_importances-01.png')
plt.show()
```

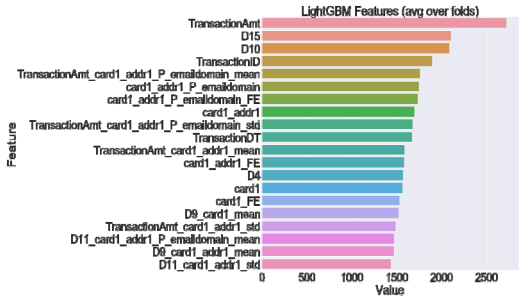

그림 1. 20가지 주요 특징들의 시각화
Fig. 1. Top 20 crucial features visualization

이제 이러한 결정들에 관한 설명을 LIME과 SHAP를 통해 각각 나타내기로 한다. 먼저, LIME을 사용한 의사결정 설명 코드는 다음과 같은 몇 줄의 코드로 나타낼 수 있다.

```
def prob(data):
    return np.array(list(zip(1-model.predict(data),
        model.predict(data))))
import lime
import lime.lime_tabular
explainer = lime.lime_tabular.LimeTabularExplainer(
    new_df[list(X.columns)].astype(int).values,
mode='classification',training_labels=new_df['isFraud'],
        feature_names=list(X.columns))
```

이러한 explainer가 임의의 인스턴스를 설명할 수 있도록 하는 코드는 아래와 같고, 출력 결과는 그림 2와 같다.

```
i = 2
exp = explainer.explain_instance(
    new_df.loc[i,list(X.columns)].astype(int).values,
        prob, num_features=10)
##결과 가시화
exp.show_in_notebook(show_table=True)
```

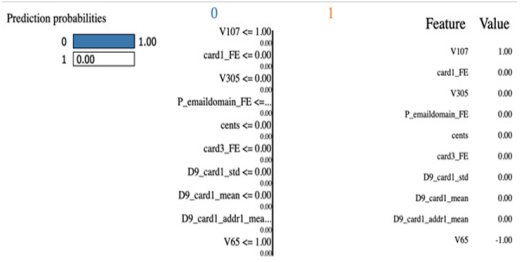

그림 2. 임의의 인스턴스 출력 결과(i=2)
Fig. 2. Random instance output result(i=2)

Keras의 tensorflow를 사용하는 경우에 신경망 모델과 모델의 분류 보고를 위한 코드는 다음과 같다.

```
classifier = Sequential()
# 첫 번째 은닉층
classifier.add(Dense(16, activation='sigmoid',
  kernel_initializer='random_normal',
input_dim=242))
# 두 번째 은닉층
classifier.add(Dense(8, activation='sigmoid',
  kernel_initializer='random_normal'))
# 출력층
classifier.add(Dense(1, activation='sigmoid',
  kernel_initializer='random_normal'))
from sklearn.metrics import classification_report
y_pred=classifier.predict(X_test, batch_size=64,
verbose=1)
y_pred =(y_pred>0.5)
y_pred_bool = np.argmax(y_pred, axis=1)
print(classification_report(Y_test, y_pred_bool))
```

다음은 위의 Keras 모델 결과에 대한 LIME 설명자 코드이다.

```
def prob(data):
    print(data.shape)
    y_pred=classifier.predict(data).reshape(-1, 1)
    y_pred =(y_pred>0.5)

print(np.array(list(zip(1-y_pred.reshape(data.shape[0])
,
        y_pred.reshape(data.shape[0])))))
    return np.hstack((1-y_pred,y_pred))
import lime
import lime.lime_tabular
explainer = lime.lime_tabular.LimeTabularExplainer(
    X[list(X.columns)].astype(int).values,
    mode='classification',
    training_labels=new_df['isFraud'],
    feature_names=list(X.columns))
```

위의 설명자를 사용하여 임의의 인스턴스의 출력 결과는 아래 그림 3과 같이 출력 결과을 설명할 수 있다.

```
i = 19
exp =
explainer.explain_instance(X.loc[i,X.columns].astype(i
nt).
    values, prob, num_features=5)
esp.show_in_notebook(show_table=True)
```

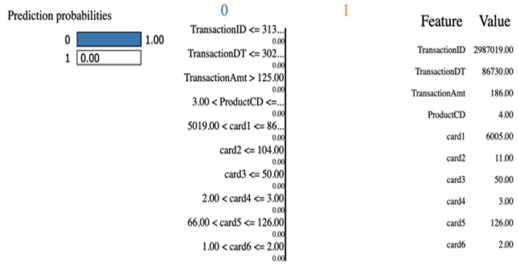

그림 3. 임의의 인스턴스 출력 결과(i=19)
Fig. 3. Random instance output result(i=19)

SHAP의 장점은 예측 해석을 위해 사용 가능한 모든 체계를 통합한다는 사실에 있다. SHAP는 각 특징에 특정 예측에 대한 중요도 값을 할당한다. 새로운 구성 요소에는 추가 특징 중요도 측정의 새로운 클래스 식별 및 바람직한 특징 집합을 가진 이 클래스에 고유한 솔루션이 있음을 보여주는 이론적 결과가 포함된다. 새 클래스는 6개의 기존 방법을 통합하며, 클래스의 최근 몇 가지 방

법에는 제안된 바람직한 속성이 없어서 주목할 만하다. 이 통합에서 얻은 통찰력을 기반으로 이전 접근 방식보다 향상된 계산 성능 또는 인간들의 직관과의 일관성을 보여주는 새로운 방법을 제시한다. 다음 코드는 LightGBM의 앙상블 모델로 설명은 기본값(전달한 훈련 데이터 세트에 대한 평균 모델 출력)에서 모델 출력으로 모델 출력을 밀어 넣는 데 공헌하는 특징을 각각 보여줄 수 있다(그림 4 참조).

```
import shap
# JS 가시화 코드 가져오기
shap.initjs()
# SHAP 값을 사용한 모델 예측의 설명
# 다음 코드는 LightGBM, CatBoost, scikit-learn 및
spark 모델에서 작동함.
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)
# 첫 번째 예측의 설명 가시화 (matplotlib=True 사용)
# shap.force_plot(explainer.expected_value,
shap_values[0,:], X.iloc[0,:])
shap.dependence_plot("TransactionAmt", shap_values,
X)
# 모든 특징 효과 요약
shap.summary_plot(shap_values, X)
```

단일 특징이 모델의 출력에 미치는 영향을 이해하기 위해 해당 특징의 SHAP 값과 데이터 세트의 모든 예에 대한 특징값을 나타낼 수 있다. SHAP 값은 모델 출력의 변경에 대한 특징의 책임을 나타냄으로 거래 금액이 변경됨에 따라 예상 주택 가격이 변경된다. 이와 관련하여 거래 금액의 단일 값에서 수직 분산은 다른 특징과의 상호 작용 효과를 나타낸다.

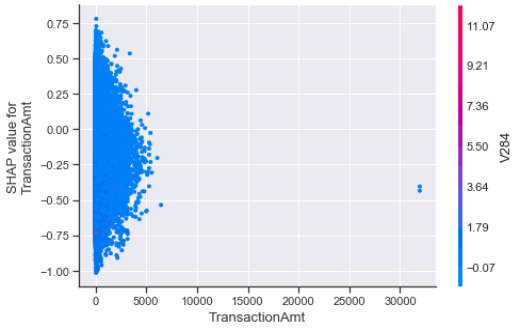

그림 4. LightGBM 모델 설명 출력
Fig. 4. LightGBM model explanations

모델에서 가장 중요한 특징에 대한 개요를 얻기 위해 모든 표본에 대한 모든 특징의 SHAP 값을 그래프로 나타낼 수 있다. 그림 5는 모든 표본에 대한 SHAP 값 크기의 합계로 특징을 정렬하고 SHAP 값을 사용하여 각 특징이 모델 출력에 미치는 영향의 분포를 보여준다. 색상은 특징값(빨간색 높음, 파란색 낮음)을 나타낸다. 이것은 예를 들어 카드 소지자 주소가 예상 사기 상태에 큰 영향을 미친다는 것이다. 또한 각 특징에 대한 SHAP 값의 평균 절댓값을 취하여 표준 막대그래프를 얻을 수 있다.

```
shap.summary_plot(shap_values, X, plot_type="bar")
```

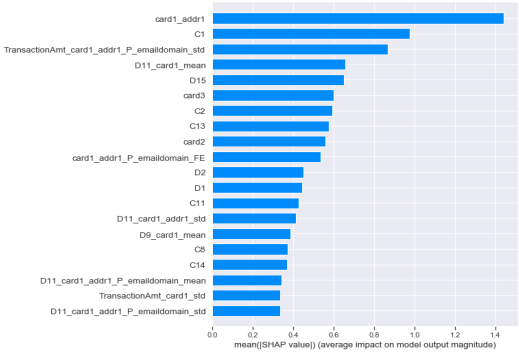

그림 5. 각 특징의 모델 출력에 미치는 영향의 분포
Fig. 5. Impact distribution of model output for each feature

LightGBM과 유사하게 아래와 같이 딥 러닝에 SHAP를 사용할 수 있는데, TreeExplainer 대신 keras 호환의 DeepExplainer를 사용한다. 각 특징에 대한 SHAP 값의 평균 절댓값을 취하여 표준 막대그래프를 얻을 수 있는데, 관련 코드는 다음과 같다.

```
import shap
import tensorflow.keras.backend
background =
X_train[np.random.choice(X_train.shape[0],
    100, replace=False)]
# 통합 관련 배경 데이터 세트로 첫 100개 훈련 예 사용
explainer = shap.DeepExplainer(classifier,
background)
# shap.force_plot(explainer.expected_value,
#     shap_values[0], X_test[0])
# 모든 특징 효과 요약
sharp.summary_plot(shap_values, X)
```

V. 결 론

Shapley 값을 계산하는 것은 복잡한 과정이다(다양한 입력 변수 조합에서 모델을 평가해야 함). 계산 단계는 Shapley 값이 무엇이고 어디에서 왔는지 시각화하는 데 도움이 된다. 특정 플롯을 사용하여 특정 관측에서 이루어진 예측에 대한 각 특징의 중요성을 설명할 수 있다. 모든 플롯은 기대치로 구성되는데, 이는 모든 SHAP 값의 합계를 특징이 없을 때의 예측과 모든 특징이 있을 때의 예측 간의 차이로 해석할 수 있어서 바람직한 속성이다. 이러한 맥락에서 각 특징의 SHAP 값은 모델이 만든 예측에 대한 기여도를 나타낸다. 일반적으로 말해서 SHAP 값은 신경망과 같은 비선형 모델을 사용할 때 부분 의존도에서는 볼 수 없지만, 직관은 사용하는 모델과 무관하게 같게 유지된다. 모든 특징의 SHAP 값을 계산하여 단일 관찰에 대한 예측을 설명한다. 본 연구에서는 SHAP와 LIME 중에서 선택하는 방법에 대한 몇 가지 지침을 제공하고 각각의 몇 가지 제한 사항과 공유 문제를 나타냈다. 두 접근 방식 모두 강점과 한계가 있지만 두 모델의 공유와 관련해서 우선은 SHAP를 사용하고 SHAP의 계산 비용이 큰 경우에는 LIME에 의존하는 것이 바람직하다.

References

[1] Hastie, R. Tibshirani and R. Tibshirani. "Best Subset, Forward Stepwise or Lasso? Analysis and Recommendations Based on Extensive Comparisons." Statistical Science 2020, Vol. 35, No. 4, pp.579-592, 2020.
DOI: https://doi.org/10.1214/19-STS733

[2] A. Goldstein, A. Kapelner, J. Bleich and E. Pitkin. "Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation" arXiv:1309.6392v2 [stat.AP] 20 Mar 2014.
DOI: https://doi.org/10.48550/arXiv.1309.6392

[3] L. Breiman. "Random Forests". English. Machine Learn. Vol.45, pp.5-32, 2001.
DOI: https://doi.org/10.1023/A:1010933404324

[4] A. Hapfelmeier and K. Ulm. "A new variable selection approach using Random Forests." Computational Statistics & Data Analysis, Elsevier, vol. 60(C), pp.50-69, 2013.
DOI: https://doi.org/10.1016/j.csda.2012.09.020

[5] L. Mentch and G. Hooker. "Quantifying Uncertainty in Random Forests via Confidence Intervals and Hypothesis Tests" JMLR, 17(26):1-41, 2016.

[6] J. Kazemitabar, A. Amini, A. Bloniarz, and A. S. Talwalkar. "Variable importance using decision trees." Advances in neural information processing systems, pp.426-435, 2017.

[7] M. Loecher. "Unbiased variable importance for random forests." Communications in Statistics - Theory and Methods, 51(5):1413-1425, 2022.
DOI: https://doi.org/10.1080/03610926.2020.1764042

[8] A. L. Adler and A. Painsky. "Feature Importance in Gradient Boosting Trees with Cross-Validation Feature Selection." Entropy 24(5):687, May 2022.
DOI: https://doi.org/10.48550/arXiv.2109.05468

[9] Y.G Kim, K.I Moon. "Image Restoration Based on Inverse Order and Power Spectrum Density." The Journal of The Institute of Internet, Broadcasting and Communication (IIBC). Vol.16, No.2, pp.113-122, 2016
DOI: https://doi.org/10.7236/JIIBC.2016.16.2.113

[10] S. M. Lundberg and S. I. Lee. "A unified approach to interpreting model predictions." Advances in Neural Information Processing Systems, pp.4768-4777, 2017.
DOI: https://doi.org/10.48550/arXiv.1705.07874

[11] M. Loecher. "Debiasing mdi feature importance and shap values in tree ensembles." In Holzinger, A., Kieseberg, P., Tjoa, A. M., and Weippl, E., editors, Machine Learning and Knowledge Extraction, pp.114-129, Cham. Springer International Publishing, 2022.
DOI: https://doi.org/10.1007/978-3-031-14463-9_8

[12] Y.G Kim, "Image Reconstruction Using Poisson Model Screened from Image Gradient." The Journal of The Institute of Internet, Broadcasting and Communication (IIBC). Vol.18, No 2, pp.117-123, 2018
DOI: https://doi.org/10.7236/JIIBC.2018.18.2.117

저 자 소 개

김 용 길(정회원)

- 1990년 : 호남대학교 전산통계학과 졸업(이학사)
- 1992년 : 광주대학교 대학원 컴퓨터학과 졸업(공학석사)
- 2021년 : 호남대학교 대학원 컴퓨터공학과 졸업(공학박사)
- 2014년 ~ 현재 : 조선이공대학교 컴퓨터보안과 교수
- 관심분야 : 지능시스템, 네트워크보안, 정보보호