경영전문석사학위 논문

Machine learning을 통한 CSP Fraud Management System의 오탐률을 감소시키는 방법에 관한 연구

- CSP 사기(Fraud) 탐지 -

2022년 2월

서울과학종합대학원대학교

홍 학 진

경영전문석사학위 논문

Machine learning을 통한 CSP Fraud Management System의 오탐률을 감소시키는 방법에 관한 연구

- CSP 사기(Fraud) 탐지 -

2022년 2월

서울과학종합대학원대학교

홍 학 진

Machine learning을 통한 CSP Fraud Management System의 오탐률을 감소시키는 방법에 관한 연구

- CSP 사기(Fraud) 탐지 -

지도교수 장 중 호

이 논문을 경영학 석사 학위논문으로 제출함

2022년 2월

서울과학종합대학원대학교

홍 학 진

홍학진의 석사 학위논문을 인준함

2022년 1월

위 원 장 김 보 영 (인)

위 원 문 달 주 (인)

위 원 장 중 호 (인)

초 록

CSP(Communications Service Provider)의 사기(Fraud) 통화 탐지는 중요하지만 어려운 작업이다. 기술의 발달로 인해 다양한 방법을 통한 사기(Fraud) 통화가 이루어지고 있다. 또한, 정상적인 사용자와 동일한 방법(패턴)으로 사기(Fraud) 통화를 시도하고 있어 정상 사용자와 사기(Fraud) 통화의 구분이 더 어려워지고 있다. 이처럼 복잡하고 검출하기 어려워지는 사기(Fraud)를 임계값(Threshold)을 이용한 Rule 기반 시스템으로는 검출하는 데에 한계가 있다. 또한 CFCA 2019년 보고서에 따르면 2019년부터 다시 사기(Fraud) 통화로 인해 CSP의 손실이 증가하고 있다. 이는 앞에서도 언급을 했듯이 기술의 발달에 따른 서비스의 다양화로 인해 통화 사기(Fraud)도 다양화되어 기존의 임계값(Threshold)을 이용한 Rule기반으로는 도매인 관리자가 새로운 유형에 대한 사기(Fraud) 통화에 대해 즉각 적인 대응을 하기에는 한계가 있다.

본 연구에서는 CSP 사기(Fraud) 탐지 시스템에서 도매인 관리자가 사기(Fraud) 통화를 인지하여 임계값(Threshold)을 이용한 Rule을 생성하는 과정을 머신러닝(machine leaning)의 분류 모델(Classification model)로 대체 하여 다양한 사기(Fraud) 통화에 대해 즉각적인 대응 및 사기(Fruad) 통화를 예측하는 것을 목적이 있다. 여기서 분석에 사용한 데이터는 CSP의 통화 상세기록(CDR) 3개월간의 데이터를 사용했다. 그중 아웃바운드 국제전화 통화로 구성된 통화 상세기록(CDR) 데이터를 지도 학습의 분류 모델을 학습하여 예측했다. 레이블은 CSP 사기(Fraud) 탐지 시스템에서 검출한 사기(Fraud) 통화 및 CFCA에서 공표한 사기(Fraud) 번호를 기반으로 했으며, 사용한 분류 모델 5가지는 Decision Tree Classifier, Random Forest Classifier, Logistic Regression Classifier, K-Nearest Neighbors Classifier, Linear Support Vector Classifier 이다. 또한, 각

모듈별 최적의 하이퍼 파라미터(hyper-Parameter)를 찾아 각 예측 모델에 적용하여 사기(Fraud) 예측을 했다. 분석 결과 K-Nearest Neighbors, Random Forest 모델이 90% 이상의 사기(Fraud) 통화 예측을 했으며, 그 중에서 K-Nearest Neighbors 모델이 97%로 높은 예측을 나타냈다. 통화 상세기록(CDR)뿐만 아니라 본 연구에서 포함하지 못한 고객데이터, 과금 데이터 등과 같은 데이터를 같이 분석을 하면 CSP 사기(Fraud) 탐지에 많은 도움뿐만 아니라 사기(Fraud) 통화에도 즉각적인 대응이 가능할 것으로 기대된다.

키워드: 사기 탐지(Fraud detection), 사기 통화, 분류 기계 학습, 통신 사기

목 차

표 목 차

그 림 목 차

제 I장 서 론

제 1 절 연구의 배경

CSP(Communications Service Provide)[1]의 전화 서비스의 경우 초기 통신망(PSTN: Public Switched Telephone Network) 기반에서 IP 네트워크 인프라의 및 IP 네트워크 통신 시스템(IP-PBX, SBC 등)의 발달과 함께 인터넷을 이용한 통화 서비스로 발전을 했다. 현재 인터넷을 이용한 통화 서비스는 CSP의 기본적인 서비스로 성장하였다. 특히 최근 들어 사무실 전화기를 사용하지 않고 개인 휴대폰 혹은 컴퓨터에 설치한 어플리케이션과 회사 전화번호를 연동[2]하여 시간과 공간의 제약을 받지 않고 회사 전화를 사용할 수 있다. 이렇게 IP 인프라, 시스템 및 관련 소프트웨어가 많은 발전을 통해 IP 네트워크 상에서 서비스 운영이 가능하게 되었지만, 기술의 발전과 동시에 해킹 및 전화번호 가변 같은 부정 사용으로 인해 막대한 손해가 발생하고 있다.

CFCA(Communications Fraud Control Association) Fraud Loss Survey 2019[3] 보고서에 따르면, CSP의 사기(Fraud) 손실률은 2013년 이후 매년 감소했으나, 2019년에 다시 증가하였다. 보고서는 2019년 추정 글로벌 통신 매출이 USD 1.625 Trillion (KRW 1,868조 7,500억)으로 예상되며, 사기(Fraud) 손실금액은 USD 28.3 Billion (KRW 32조 5,450억), 그 비율은 1.74%로 예상했다. 이 설문조사 결과를 통해 감소추세였던 Fraud 손실률이 다시 상승(30% 이상)함을 알 수 있으며, 설문지에 응답한 대부분의 참여자(CSP의 FMS 운영자)는 운영 중인 사기 탐지 솔루션(Fraud management solution)이 전체 Fraud의 일부만 검출하고 있다고 답변을

1) 전화 서비스 제공회사(통신사, 별정 통신 등)
2) Teams, Zoom과 같은 App과 전화 번호를 연동하여 PSTN 또는 모바일로 전화를 연결 통화
3) CFCA 협회에 등록된 CSP업체 대상으로 한 설문조사

하였다. 또한 사기(Fraud) 탐지 솔루션의 적용은 각 CSP에서 제공하는 모든 서비스 중 약 50% 미만이라고 언급한 부분을 반영한다면, 실제 조사된 사기(Fraud) 손실금액 과소평가 되었다고 할 수 있다.

이와 유사한 분석과 예측을 시행한 Gartner Market Trends 보고서[4]에 따르면 CSP의 사기(Fraud) 손실은 여전히 연간 USD 23billion(KRW 26조 4,500억)에 이르며, 새로운 서비스 사기(Fraud)의 증가에 따른 CSP의 추가 손실 위험이 있다고 보고했다.

앞서 살펴본 바와 같이 인터넷 통화(전화)가 성장하는 동한 VoIP(Voice of Internet Protocol) 상에서 다양한 사기(Fraud)로 인해 큰 손실이 발생하였다. 다양한 사기(Fraud)에 대응하기 위해, 개별 사기(Fraud)에 대한 연구뿐만 아니라 프로파일 기법 등 다양한 연구가 이루어졌고, 서비스 사용자 패턴에 대한 연구 및 파악을 통해 수많은 사기(Fraud) 사례를 찾아 지속적으로 손실을 줄여나가는 노력을 하고 있다.

그러나 서비스의 다양화(VoLTE(Voice over Long Term Evolution), VoIP(Voice of Internet Protocol), IoT(Internet of Thing) 등)에 따라 네트워크가 복잡해지고, 더 불어 사기(Fraud)의 유형 역시 정교해져 지속적으로 손실금액이 증가하고 있다. 마치 바이러스를 제거하는 백신을 만들며, 이를 회피하는 또 다른 바이러스가 생겨나는 것과 같은 현상이라고 볼 수 있다. 반면 사기(Fraud) 유형이 점점 더 다양해지고 정교해짐에 반해 CSP의 사기(Fraud) 관리 운영 인력 부족 및 탐지 시스템 기술 대응 속도가 느려 새로운 사기(Fraud)에 즉각적인 제어를 하기 힘든 상황이다. 더군다나 현재의 사기(Fraud) 탐지 시스템 대부분은 통신 데이터를 분석 및 이미 발생한 사기(Fraud)의 특성들을 종합적으로 파악하여 도메인 관리자가 Rule과 임계값(Threshold)을 설정하여 탐지 및 검출하는 방식으로 설계되어있어 새로운 사기(Fraud) 유형이 발생을 하면 즉각적인 대응이 불가능하다.

---

4) 'Market Trand: Maximizing the Value of Analytics, Artificial Intelligence and Automation in CSP Fraud Management' 2018년 12월 21일 보고서

이러한 CSP 사업의 어려움을 인지하고, 머신러닝 기법을 활용하여 기존 시스템의 한계점을 극복한 정확도 높은 탐지 시스템의 개발뿐만 아니라 새로운 Fraud의 유형도 검출하여 손실을 최소화하고자 함이 본 연구의 목적이다.

제 2 절 연구의 목적

CSP의 사기(Fraud) 탐지 시스템들의 대부분은 데이터의 분석 기반으로 구현이 되어 있으며, 여기서 분석된 결과로 도매인 관리자가 사기(Fraud) 유무를 판단하는 구조로 되어 있다. 다시 말해 CSP의 사기(Fraud) 탐지 시스템의 경우 <그림 1>과 같이 수집, 분석, 적용, 판단 프로세스의 총 4단계를 거치게 된다.

수집 단계에서는 통화 데이터에 해당하는 CDR[5](Call Detail Record)과 고객 정보 및 통신 데이터를 수집하여 분석에 필요한 데이터 셋을 생성한다.

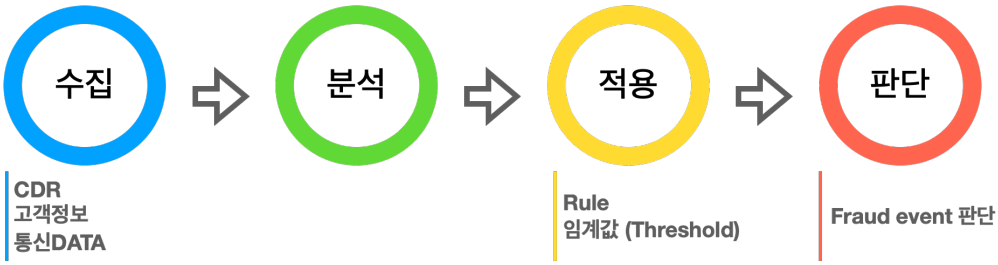
분석 단계에서는 '수집단계'에서 생성된 데이터와 기존의 사기(Fraud)로 판단된 데이터를 같이 분석하여, 새로 수집된 데이터에 사기(Fraud) 통화 혹은 유사한 통화가 있는지를 검출한다.

적용 단계에서는 도메인 관리자가 '분석 단계'에서 검출된 사기(Fraud) 통화 혹은 유사한 통화에 대해 사기(Fraud) 통화 유무를 판단한다. 여기서 판단된 새로운 유형의 사기(Fraud)는 임계치(Threshold)을 이용하여 새로운 Rule을 생성하여 사기(Fraud) 검출 시스템에 적용한다.

판단 단계에서는 사기(Fraud) 검출 시스템에서 사기(Fraud) 통화를 검출한 부분에 대해서 도매인 관리자가 검출된 통화에 대해 최종 판단을 하고 후속 조치를 한다. (차단 혹은 Rule 업데이트)

---

5) 과금할 목적으로 특정 번호나 가입자 그룹에 대한 통화 세부 내역 데이터(발신번호, 착신번호, 통화량 통화 시간 등)를 기록한 레코드



<그림 1> CSP(Fraud) 사기 검출 흐름도

이와 같이 기존의 사기(Fraud) 탐지 시스템은 임계치(Threshold)를 이용한 Rule기반으로 되어 있어 사기(Fraud) 통화 검출을 위해서는 반드시 Rule에 해당이 되어야 한다. 만약 사기(Fraud) 통화가 일반 정상 서비스 사용자 통화와 유사한 범위에서 사용되거나, 임계치(Threshold) 허용 범위에서 사기(Fraud) 통화가 발생을 하면 기존의 사기(Fraud) 검출 시스템은 사기(Fraud) 통화를 검출할 수가 없다. 그렇기 때문에 도매인 전문가가 '분석 단계'에서 분석 결과를 보고 사기(Fraud) 통화 혹은 유사한 통화인지를 판단하고 새로운 Rule 생성 혹은 Rule을 업데이트하는 단계가 반드시 필요하다.

여기서 '분석 단계'와 '적용단계'를 머신러닝 지도학습 알고리즘을 통하여 <그림 2>와 같이 하나의 '분석 단계'로 처리함으로써 도매인 관리자가 분석결과를 보고 Rule 생성 혹은 Rule 업데이트 과정이 불필요하게 된다. 또한, 서비스의 다양화 및 사기(Fraud) 유형의 정교함, 다양성 및 발전 증가에 대해 도메인 관리자가 즉각적인 대응을 할 수 있게 하는 데 목적이 있다.

<그림 2> 본 연구의 사기(Fraud) 검출 흐름도

제 3 절 연구의 구성

본 연구는 총 5장으로 구성되어 있다. 제 I 장에서 논문의 배경 및 목적에 대해서 설명하고, 제 II 장에서는 일반적인 사기(Fraud)유형과 CSP 사기 탐지 방식을 소개하고 관련 선행 연구에 대해 알아보았다. 제 III장에서는 데이터 전처리와 탐색(EDA)을 하여 기계 학습 분류 알고리즘 중 Decision Tree Classifier, Random Forest Classifier, Logistic Regression Classifier, K-Nearest Neighbors Classifier, Linear Support Vector Classifier를 사용하여 모델을 학습한 부분에 대해 기술하였다. 제 IV장에서는 분류 기계 학습으로 학습한 결과를 비교 분석한다. 제 V장에서는 본 연구의 결론으로 연구의 요약 및 시사점과 향후 연구 과제에 관해 기술하였다.

제Ⅱ 장　이론적 배경 및 선행 연구

제 1 절 사기(Fraud)의 유형과 탐지

일반적으로 사기(Fraud)는 정상적인 사용에서 벗어나 불법적인 이익을 취하거나, 다른 사람에게 해를 가하려는 변칙적인 사용을 의미한다. 사기(Fraud)를 탐지하는 것은 많은 양의 정상적인 사용에 비해 상대적으로 아주 적은 사례를 찾아야 하는 쉽지 않은 과정이다. 실제 데이터를 통해 확인한 결과 정상적인 사용량에 비해 Fraud의 비율은 2%가 되지 않지만, 앞서 살펴본 바와 같이 연간 30조 원에 달하는 금액의 손실이 발생한다. 본 절에서는 CSP에서 발생하는 사기의 유형과 탐색의 과정에 대해 살펴본다.

(1) 사기(Fraud) 유형

일반적인 경우 불법적인 이익을 취하려는 사기(Fraud) 통화는 정상적인 서비스 사용자로 가장하여 쉽게 구분할 수 없다. 특히 특정 사용자에 해당하는 통화를 분석할 때는 더욱 그렇다. 그러나 대부분의 사기(Fraud) 통화는 단 한 번의 상호 작용으로 성공을 하지 못하기 때문에 연속적인 사용 시도가 나타난다. 연속적인 통화 시도를 통해 사기(Fraud) 통화가 성공할 경우 또 다른 사용자에게 동일한 방법을 사용하는 패턴을 보인다. 또한, 더 많은 사기(Fraud) 통화의 성공과 CSP의 사기(Fraud) 탐지를 회피하기 위해 다양한 번호 또는 발신번호 변작을 통해 다량의 전화를 걸어 사기(Fraud) 통화를 시도한다. AT&T 연구소에서 조사한 사기(Fraud) 유형[6] <표 2>을 보면 대부분 정상 사용자를 가장하여 불법적인 정보 혹은 이득을 취하거나, 새로운 IT 기술을 활용하여 최신 기술의 취약점을 찾아 공격, 불법적인 이득을 취한다.

---

6) Fraud Detection in Telecommunications: History and Lessons Learned. VOL. 52, No.1

<표 1> 일반적인 사기(Fraud) 유형

사기(Fraud) 유형	설명
Subscription fraud	서비스 사용 구독에 대한 사용료를 지불할 의사가 없는 구독
Intrusion fraud	합법적인 사용자의 개정을 도용하여 불법적으로 사용
Fraud based on loopholes in technology	기술적인 취약점을 이용하여 정상적인 서비스를 해킹하여 사용 또는 취약점을 이용하여 정상 사용자의 정보 탈취 이용.
Social engineering	사회 구성원으로서의 접근을 통해 정상사용자의 정보 탈취 사용 (서버 관리자로 가장해서 접근 등)
Fraud base on new technology	새로운 기술을 이용하여 사기 전화 발생 이득 취득
Fraud based on new regulation	규정에 따른 사기 (통신 재판매 시 가상 전화를 발생하여 통화 인센티브 취득)
Masquerading as another user	다른 정상 사용자로 가장을 하여 사용

(2) 사기(Fraud) 탐지

사기(Fraud)를 탐지하기 위해서는 많은 양의 연속적인 통화 상세기록(CDR: Call Detail Record)이 필요하며, 좀 더 정확한 사기(Fraud) 탐지를 위해서는 과금 데이터 및 고객 정보도 같이 분석을 하게 된다. 위에서 언급했듯이 대부분의 사기(Fraud) 통화 시도는 단 한 번의 상호 작용으로 성공

을 하지 못하기 때문에 연속적인 사용(통화) 시도가 나타난다. 이처럼 연속적인 사용(통화) 시도는 정상적인 서비스 사용자의 패턴에서 벗어난 형태를 보여 준다. 사기(Fraud) 탐지의 기본은 바로 비정상적인 패턴을 찾는 것부터 시작된다. 이러한 비정상적인 패턴을 찾기 위해서는 연속된 통화 상세기록(CDR)를 분석하여 도출된 결과를 시각화 후 보다 빠르게 패턴을 찾는다. 통화 상세기록(CDR)의 형식은 국제 표준이 존재하지만 네트워크의 복잡성 및 장비 재조사에 따라 사용하는 항목들이 차이가 있어 분석 전에 데이터를 필터링 및 전처리하는 과정이 필요하다. 데이터가 준비되면 비정상 패턴을 찾는 알고리즘 또는 기존에 확인된 패턴과 비교, 분석 및 시각화하고, 사기(Fraud) 통화를 규정할 수 있는 Rule과 해당 Rule에 적용되는 임계값(Threshold)을 찾는다. 이렇게 찾은 Rule과 임계값(Threshold)을 CSP에서 운용 중인 사기(Fraud) 탐지 시스템에 적용 및 업데이트하여 실시간으로 사기(Fraud) 통화를 검출하게 된다. 여기서 탐지된 사기(Fraud) 통화는 도매인 관리자가 최종적으로 판단을 하여 후속 조치를 하게 된다. 이와 같이 마지막 단계에서 도매인 관리자가 한 번 더 확인을 하는 이유는 정상적인 서비스 사용자의 통화가, 사기(Fraud) 통화 검출을 위한 규칙(Rule)과 임계값(Threshold)에 포함될 수 있기 때문이다. 다시 한번 탐지과정을 요약하면, 과거의 연속적이고, 수많은 통신기록에서 일반적이지 않은 패턴을 발견 및 Rule을 설정하여 시스템을 지속적으로 업데이트하는 것이다.

CSP 업체의 손실을 줄이기 위한 이러한 노력에도 불구하고 신종 바이러스와 같이 기존과는 다른 새로운 유형의 사기(Fraud) 통화가 생겨나고 있으며, 기술 발달에 따른 다양한 서비스 및 다양한 앱을 통한 사기(Fraud) 통화가 증가하고 있다. 또한, 기술 및 서비스의 발달 속도에 비해 사기(Fraud) 통화를 검출 대응하는 전문인력의 부족으로 인해 손실을 줄이지 못하고 있는 상황이다.

제 2 절 선행 연구

연구 진행하기에 앞서 본 절에서는 사기(Fraud)를 탐지하는 연구사례에 대해서 알아보고자 한다.

첫째, CS Hilas, JN Sahalos이 제안한 사용자 프로 파일 기법[7]이다. 이 연구는 전호번호(전화기 단말) 기준으로 지역, 모바일, 국내, 해외 통화의 각 통화 건수와 통화 시간을 산출하여 통화 건수와 시간의 유사성을 검출 후 정상 사용자와 사기(Fraud) 사용자를 구분하는 프로파일 기법이다. 총 12대 전화기(단말)에서 1년간 발신한 전화번호를 기준으로 분산분석(ANOVA test) 및 T Pair-by-Pair T-tests을 사용하여 수학식(식 1)을 도출하였다. 검증 결과 프로파일 생성을 위한 39일이 지난 후에 80% 이상의 높은 유사성 확률이 검증되었다고 한다. 여기서 프로파일과 다른 데이터 혹은 이상치 및 극단적인 값을 가지는 데이터를 사기(Fraud) 가능성이 있는 것으로 판단했다.

$$similarity(seq_{test}^{i}, K) = \max_{seq^{j} \in K} \{similarity(seq_{test}^{i}, seq^{j})\} \quad (1)$$

두 번째, H.Hakan Kilinc가 제안한 사기(Fraud) 사용자 행동 분석 기법[8]은 터기 CSP 국제 전화 통화 상세기록(CDR) 데이터를 활용하였으며, VoIP(Voice of Internet Protocol) 사기성 호출을 감지하기 위해 행동 분석 속성을 사용하였다. 속성의 종류는 call type(national, international, mobile, etc.), call date, call duration, call count, and call destination이 있으며, 이러한 속성 데이터를 사용하여 사기(Fraud) 통화 사용자 행위

7) CS Hilas, JN Sahalos (2005). User Profiling for Fraud Detection in Telecommunications Networks

8) H. Hakan Kilinc (2020). A Case Study on Fraudulent User Behaviors in the Telecommunication Network.

분석을 진행한 연구이다. <표 3>과 같이 사용자 통화 행위 분석 및 판별을 위한 조사 결과 10가지 사기(Fraud) 사용자 행위와 특징이 있다는 점을 발견했다.

<표 2> 사기(Fraud) 사용자 행동

사기(Fraud) 행위	사기(Fraud) 사용자 행위
Type 1	장시간 사용한 적은 콜수
Type 2	적은 사용시간이 많은 콜수
Type 3	특정 시간에 장시간 사용
Type 4	특정 시간에 특정 국가로 많은 콜수
Type 5	특정 시간에 특정 국가로 장시간 사용
Type 6	동일기간에 동일 사용자가 동일 국가로 장시간 사용
Type 7	동일기간에 동일 사용자가 동일 국가로 적은시간 사용으로 많은 콜수
Type 8	동일기간에 동일 사용자가 동일 국가로 장시간 사용으로 많은 콜수
Type 9	넓은 범위의 기간에 같은 목적지로 드물게 적은 콜수
Type 10	광범위한 시간 동안 서로 다른 대상에개 드물게 적은 콜수

세 번째, Wu et al.는 VoIP 시스템을 위한 침입탐지 시스템 설계를 제안[9] 했다. 이 시스템은 Rule 기반 침입 탐지 시스템으로 SIP[10]과 RTP[11]

9) Y. Wu, S. Bagchi, S. Garg, N. Singh, "SCIDIVE: A stateful and cross protocol intrusion detection architecture for Voice-over-IP environments",

기반의 SIP-BYE 공격, fake instant messaging, call hijacking 그리고 RTP공격에 효과적이다.

네 번째, Olszewski[12]는 구독 사기를 탐지하기 위해 사용자 프로파일 기반으로 정상 행동과 사기 행동을 구분하는 방법을 Kullback-Leibler divergence(KL-divergence)는 확률 이론, 통계 및 정보 이론으로 제안했다. 확률 및 통계적 맥락에서 두 확률 분포 간의 비유사성을 평가하는 반면 정보 이론 맥락에서는 상대 엔트로피의 척도를 사용하여 구독 사기 탐지를 효율적으로 검출하는 방법을 제안했다.

다섯 번째, Hoffstadt et al.[13]은 4,750만 이상의 SIP 메시지를 기록한 VoIP Honeynet시스템을 사용하여 다양한 VoIP 공격 단계를 분석했으며, VoIP 네트워크에서 사기(Fraud)를 감지하고 방지하기 위해 다계층 솔루션을 제안했으며 이 솔루션은 Rule 기반 사용자 및 call 프로파일링 기법을 사용했다.

여섯 번째, 이종원[14]은 VoIP사에서 과금 회피하는 사기를 검출하는 방법에 대해 기존의 DPI(Deep Packet Inspection)을 통해 Toll Bypass Fraud 검출 방법으로는 부족하여 DFI(Deep Flow Inspection)와 DPI가 결합된 검출 방법을 제안했다. Hilas와 Mastorocostas[15]는 중첩된 사기(Fraud)를 탐지하기 위해 지도 학습으로 Feed-Forward 인공 신경망의 한 종류인 다층 퍼셉트론 기법과 비지도 학습으로 계층적 응집 클러스터링 기

---

10) Session Initiation Protocol (SIP)는 IETF에서 정의한 시크널링 프로토콜로 음성과 화상 통신과 같은 멀티미디어 세션을 제어하기 위해 사용된다.

11) Real-time Transport Protocol(RTP)은 오디오와 비디오를 IP 네트워크에서 실시간으로 전달하기 위한 프로토콜

12) Olszewski,"A probabilistic approach to fraud detection in telecommunications", Knowledge-Based Systems, vol. 26,

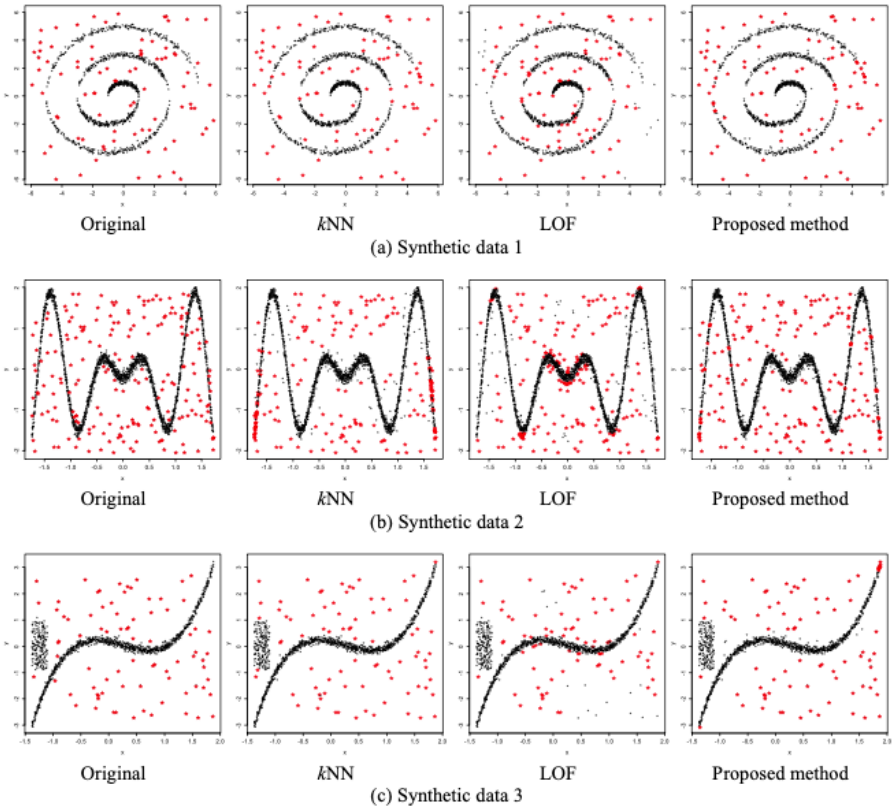
13) Hoffstadt, E. Rathgeb, M. Liebig, R. Meister, Y. Rebahi, T.Q. Thanh, "A comprehensive framework for detecting and preventing VoIP fraud and misuse.", The IEEE International Conference on Computing, Networking and Communications (ICNC), Honolulu, USA,

14) 이정원, 엄정훈, 박태흥, 김승호(2015), “VoIP 상의 과금 회피 방법과 그 검출 방법에 대한 연구”

15) C. S. Hilas, P. A. Mastorocostas, "An application of supervised and unsupervised learning approaches to telecommunications fraud detection", Knowledge-Based Systems, vol. 21, no. 7

법을 사용했다.

마지막으로 하 지연. 이종석의 무작위 추출 기반이 이상치 탐지 기법[16] 에서의 이상치의 정의는 Howkins의 정의 (1980) “데이터 집합 내의 대부분의 데이터를 생성하는 메커니즘이 아닌, 다른 예외적인 메커니즘에 의해 생성된 데이터”를 따르고 있으며, 데이터에서 처음 무작위로 추출 후 추출한 값을 기준으로 가까운 거리의 데이터를 같이 추출하여 추출되지 않는 데이터를 추출된 데이터와 다른 속성을 가진 데이터 즉 이상치로 판단을 하며, 위와 같이 무작위 추출을 여러 번 하여 추출되지 않은 데이터들에 이상치로 보이는 지표를 부과하는 기법이다. 위와 같은 무작위 추출을 여러 번 반복하는 알고리즘을 연구하여 <그림 3>와 같은 결과를 도출했다.



<그림 3> 3가지 방법에 따른 3가지 데이터세트의 검출 결과

16) 하지현, 이종석 (2013), “무작위 추출 기반의 확률적 이상치 탐지,” 대한 산업공학회 추계학술대회 논문집, 961-965.

제 Ⅲ장 연구 방법론

제 1 절 방법론 개요

(1) CSP 사기(Fraud) 탐지 시스템

대부분의 CSP에서 운영 중인 Rule 기반 사기(Fraud) 관리 시스템을 살펴보면, 각 통신사의 시스템에서 수집된 네트워크데이터, 통신데이터, 고객 데이터, 과금 데이터 등을 수집하여, 여러 가지 분석 및 통계기법으로 사기(Fraud) 유형을 판별한다. 판별된 사기(Fraud) 유형을 탐지하기 위한 Rule과 임계값을 도매인 관리자가 생성 및 설정하고 이를 다시 탐지 시스템에 적용 후 실시간으로 통화 데이터를 분석 및 사기(Fraud) 유무를 판별하게 되며, 만약 사기(Fraud) 탐지 시 즉시 경보가 발생한다. 도매인 관리자는 이 경보를 보고 실질적으로 이 통화가 사기(Fraud)인지를 판별을 후속 조치를 시행한다. 여기서 후속 조치라 함은 해당 통화에 대한 차단을 의미하며, 도매인 관리자는 향후 사기(Fraud) 통화에 대해 추가적인 분석을 위해 사기(Fraud)로 판별된 통화 레코드에 대한 레이블 작업도 병행한다. <그림 4>는 대부분의 CSP에서 운영 중이 Rule 기반 사기(Fraud) 탐지 시스템의 사기 탐지 블럭도이다.

<그림 4> CSP 사기(Fraud) 탐지 블럭도

CSP에서 운영 중인 Rule 기반 사기(Fraud) 관리 시스템에서 활용되는 분석 데이터의 종류와 분석 대상은 <표 4> 과 같다. 네트워크 데이터 분석은 SIP, RTP등과 같은 통화에 관련된 프로토콜을 분석 및 활용하여 Fraud

가 시도(공격)를 방어하며, 또한 관련 네트워크 장비의 취약점이나 오류로 인해 공격받는 것을 방지한다. 고객의 통화 데이터는 통화 상세기록(CDR)을 의미하며, 하나의 통화에 하나의 통화(CDR) 레코드가 생성된다. 즉 고객이 전화를 걸어 통화가 완료되면 하나의 통화(CDR) 레코드가 생성된다. 네트워크상에서는 하나의 통화에 대해 통화의 시작, 종료, 통화 중, 이벤트 등 여러 개의 기록들이 저장된다. 이렇게 저장된 레코드들은 고객번호 기준으로 통합이 되어 하나의 고객 번호에 하나의 통화 데이터(CDR)가 생성된다. 과금 데이터는 고객과 통화 데이터(CDR)와 매핑 테이블을 구성하여 사용 행위 분석 및 사기(Fraud) 유무를 판별하는 데 사용을 한다.

<표 3> CSP 분석 데이터의 종류

데이터 종류	분석 대상
네트워크 데이터	SIP, RTP Protocal 네트워크 장비 오류
통화 데이터	CDR (Call Detail Data)
고객 데이터	CDR과 매핑 테이블 구성 고객 사용 행위
과금 데이터	지역 요금별 사용 빈도 납부 여부

(2) 연구 방법론

본 연구는 앞서 살펴본 CSP에서 수집 및 저장된 실제 데이터를 기반으로 분석을 하고자 하며, 사기(Fraud) 정보 또한 동일 시스템에서 도메인 관리자가 판별한 데이터를 활용한다. 연구의 진행 과정은 <표 5>와 같이

데이터 수집, 데이터 탐색 단계, 데이터 분석 단계, 결과 및 시사점 도출 단계와 같이 총 4단계로 구분된다.

<표 4> 연구 4단계

연구 단계	설명
데이터 수집단계	데이터 수집 분석 데이터 추출
데이터 탐색 단계	데이터 전처리 데이터 탐색 Feature 추출
데이터 분석 단계	지도 학습 분류 알고리즘 분석 결과 도출
결과 및 시사점 도출 단계	분석 결과 도출 기존 시스템과 비교 시사점 도출

첫째, 데이터 수집 단계에서는 CSP에서 수집된 실제 통화(CDR) 데이터에 본 연구에서 분석할 데이터를 추출하는 단계이며, 국제 전화 데이터 중 SIP 통신으로 이루어진 총 3개월(12월~2월)의 데이터가 그 대상이다. 이 중 해외에서 발신되어서 국내로 착신되지 않고 바로 해외로 바로 나가는 통화를 제외한 인-바운드와 아웃-바운드 모두 분석에 활용하고자 한다.

둘째, 데이터 탐색 단계에서는 데이터 전처리, 데이터 탐색 및 Feature를 선정하는 단계이다. 원본 데이터를 살펴보면 300개 이상의 Feature가 존재한다. 이중 실제 분석에 활용할 Feature를 결정하고, 결측치 처리 및 데이터 탐색을 하는 단계이다. 아울러 도메인 전문가에 의해 판별된 사기(Fraud) 정보를 구분하여 학습에 사용될 최종 데이터를 생성한다.

셋째, 데이터 분석 단계에서는 전처리가 완료된 데이터를 활용하여 기계학습 중 지도 학습의 분류 알고리즘을 사용하여 분석결과를 도출하고 알고리즘에 따른 결괏값을 비교한다.

마지막인 결과 및 시사점 도출 단계에서는 분류 알고리즘으로 분석된 결과와 CSP에서 현재 사용 중인 Rule 기반 사기(Fraud) 검출 시스템과의 차이점과 시사점을 도출한다.

제 2 절 데이터 전 처리와 탐색

(1) 전 처리 과정 및 Feature 추출

앞에서 설명한 바와 같이 분석 대상이 되는 원본 데이터는 실제 CSP에서 수집된 인-바운드, 아웃-바운드의 국제전화 레코드로, 측정 기간은 총 3개월(12월~1월) 이다. 원본 데이터에는 300개 이상의 feature값이 존재하지만, 실제 사기(Fraud) 탐지에 있어 기본적이고 핵심적인 11개의 feature와 Fraud 여부를 표시하는 1개의 feature를 추가하여 총 12개로<표 6> 정리하였다. 11개의 feature 선정 과정에서 실제 데이터를 관리하는 도메인 관리자와 협의를 통해 진행하였다.

<표 5> 분석 대상 통화(CDR) 데이터 feature

#	feature	설명
1	Start_Time	통화 시작 시간
2	End_Time	통화 종료 시간
3	Direction	통화 타입(인바운드, 아웃바운드, 트랜스퍼)
4	From_Number	발신자 번호
5	To_Number	착신자 번호
6	OC_CODE	발신 국가 코드
7	IC_CODE	착신 국가 코드
8	OType_Code	발신 회선 종류 (유선, 무선, 국제, 알수 없음)
9	IType_Code	착신 회선 종류 (유선, 무선, 국제, 알수 없음)
10	Call_Duration	통화 시간
11	Service_Result	서비스 결과(완료호, 불완료호(Ack 받기전), 실패호, 메시지 타임아웃)
12	Fraud	사기(Fraud) 유무 (사기, 정상)

12번째 사기(Fraud) 여부에 대한 필드값은 CFCA 협회에서 사기(Fraud) 번호로 공표한 번호와 도매인 전문가가 판별한 발신번호, 착신번호 및 위험국가에 대한 것을 참고하였다. 3개월간 데이터를 확인해 보니 사기(Fraud) 해당하는 레코드가 전체 데이터의 2.75%로 나타났다. 또한, 도메인 관리자와 해당 문제에 대해 확인 및 세부 데이터 재검토 결과 Fraud에 해당은 되지만 제외된 부분을 발견할 수 있었으며, 이 부분도 모두 사기(Fraud)로 재정의를 하였다.

정리된 12가지 Feature 값의 결측 유무 확인 결과 결측치가 일부 존재하였으나, 전체 데이터의 양에 비하여 숫자가 많지 않았으며, 분석에 영향을 주지 않을 것으로 판단하여 모두 삭제 처리하였다.

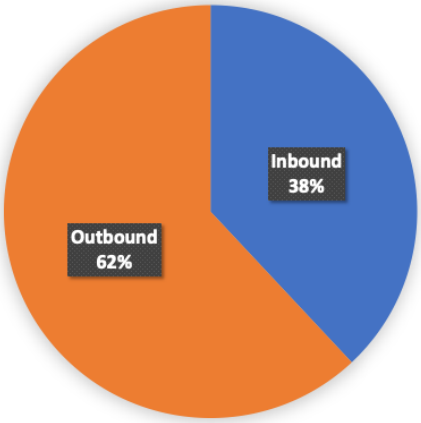
별도로 진행한 전처리 과정은 다음과 같다. Feature 중 'Service Result'는 통화가 정상적으로 이루어졌는지에 대한 내용으로 총 4가지 값을 가진다. 그중 '메시지 타임아웃'의 경우 대부분은 장비 간의 통신에 문제에 기인한 부분이기 때문에 사기(Fraud)를 판별하는 데 있어 불필요한 케이스로 판단하여 삭제 처리를 하였다.

(2) 데이터 탐색 (EDA)

알고리즘을 적용한 모델링 작업에 앞서 데이터 탐색(EDA) 과정은 필수적으로 진행되어야 한다. EDA를 통하여 시각화 또는 수치로 도출되는 결과를 바탕으로 다시 한번 데이터에 이상이 없는지 확인하고, 알고리즘 적용의 방향성도 수정할 수 있다. 즉 수집된 통신 데이터(CDR) 기반으로 전반적인 통화 유형, 정상적인 통화와 사기(Fraud) 통화의 유형, 사기(Fraud) 통화와 국가 간의 관계, 사기(Fraud)와 통화 요율과의 관계 등 다양한 측면으로 데이터를 탐색하여 적절한 분석 도구를 선택하고, 적절한 기계학습 알고리즘을 판단한다.

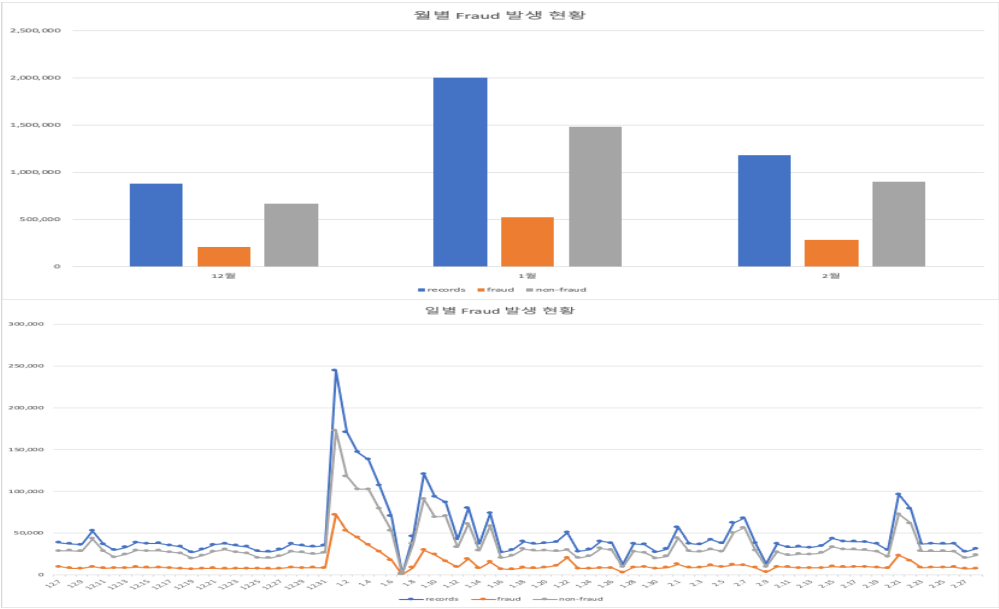
앞에서 언급을 한 바와 같이 분석 대상으로 수집된 데이터는 국제전화를 기반으로 하고 있으며, 인-바운드, 아웃-바운드 두 가지 유형의 데이터가 존재한다. 두 가지 유형 내에서 사기(Fraud) 비율을 살펴보면 <그림 5>와

같이 아웃-바운드 통화의 Fraud 비율이 62%로 인-바운드 통화 사기(Fraud) 비율 38%보다 높은 비율로 사기(Fraud) 통화가 검출되었다.



<그림 5> 통화 타입별 사기(Fraud) 분포

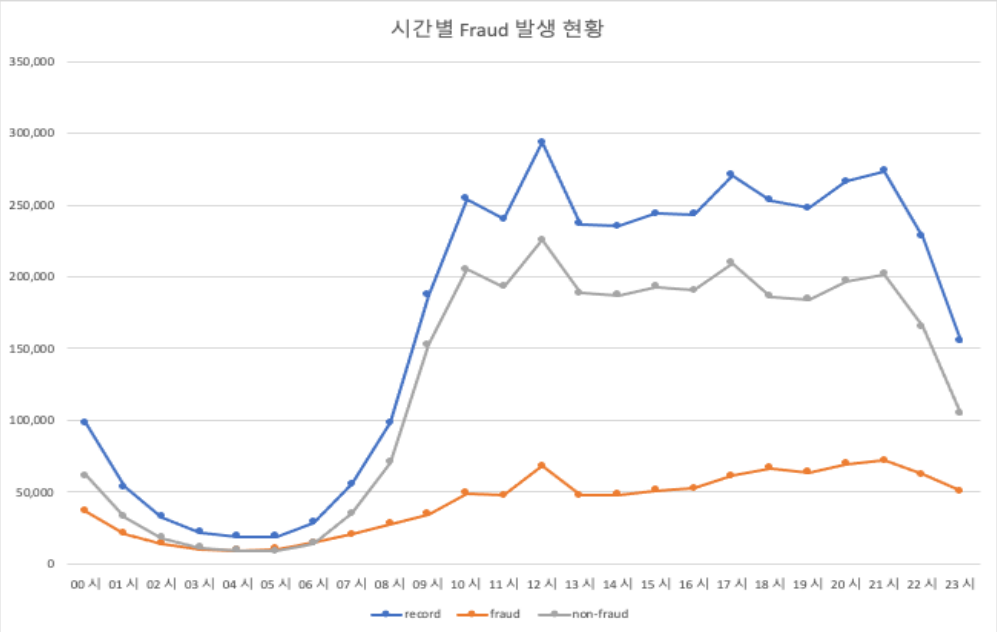
데이터의 수집 기간은 12월~ 2월 총 3개월간이며, 월별, 일별 전체 통화량 대비 정상 통화 및 사기(Fraud) 통화 분포<그림 6>를 보면 1월이 가장 높음을 확인할 수 있다. 1월은 신년 인사 등 안부 통화량이 많은 시기이며, 이와 더불어 사기(Fraud) 통화량도 같이 증가 추세를 보여 준다.



<그림 6> 월별 일별 전체 통화량에 따른 사기(Fraud) 통화 분포

일별 특징을 그래프로 살펴보면 사기(Fraud) 비율이 최소 17.20%에서 최대 43.80% 비율로 차이를 보이며, 평균적으로 보면 평균 24.70%로 사기(Fraud) 통화가 발생함을 알 수 있다. 여기서 중요한 점은 일별 전체 통화량에 따른 사기(Fraud) 통화도 일정한 비율로 같이 발생을 한다는 점이다. 즉 Fraud 통화의 시도는 매일 전체 통화량에 비례하여 발생하고 있다는 점이며, 이는 CSP에서 지속적으로 탐지 시스템을 업데이트하더라도 새롭게 생겨나는 사기(Fraud) 통화가 발생한다는 의미로 연결될 수 있다.

다음으로 시간대별 Fraud 통화 비율에 대해 살펴보았다. 시간별 전체 통화량에 따른 Fraud 통화 분포<그림 7>를 보면 시간별 Fraud 통화 비율은 최소 18.43%에서 최대 53.64%, 비율로 나타났으며, 평균은 30.78%이다. 특이한 점은 23시부터 익일 07시까지 32% 이상의 Fraud 통화가 발생하였으며, 특히 04시부터 06시까지는 50%를 상회한다. <표 7>



<그림 7> 시간별 전체 통화량에 따른 사기(Fraud) 통화 분포

<표 6> 시간별 전체 통화량에 따른 사기(Fraud) 통화 분포

time	record	fraud	non-fraud	fraud %	time	record	fraud	non-fraud	fraud %
00 시	97,854	36614	61,240	37.42%	12 시	293,482	67984	225,498	23.16%
01 시	53,332	20791	32,541	38.98%	13 시	236,624	47780	188,844	20.19%
02 시	32,430	14130	18,300	43.57%	14 시	235,399	48229	187,170	20.49%
03 시	21,839	10539	11,300	48.26%	15 시	244,187	51291	192,896	21.00%
04 시	18,889	9440	9,449	49.98%	16 시	243,329	52577	190,752	21.61%
05 시	19,075	10231	8,844	53.64%	17 시	270,720	61122	209,598	22.58%
06 시	28,571	14494	14,077	50.73%	18 시	253,238	66861	186,377	26.40%
07 시	55,647	20374	35,273	36.61%	19 시	247,785	63637	184,148	25.68%
08 시	98,115	27693	70,422	28.23%	20 시	266,526	69612	196,914	26.12%
09 시	187,044	34477	152,567	18.43%	21 시	273,675	71898	201,777	26.27%
10 시	254,435	49390	205,045	19.41%	22 시	227,971	62439	165,532	27.39%
11 시	240,355	47512	192,843	19.77%	23 시	155,204	50787	104,417	32.72%

이번에는 발신과 착신 국가별로 사기(Fraud) 비율을 확인해 보았다. 먼저 발신국 상위 30개 나라<표 8>를 살펴보면 대부분의 경우 적은 통화 건수와 해당 시도 모두 사기(Fraud)임이 확인된다. 이들 국가는 CSP에서 사기 위험 국가로 관리하는 국가들로 통화 요율이 다소 높은 국가로 이루어져 있다는 것이다. 특이한 점은 사기(Fraud) 통화 발신 국가가 한국도 28번째 위치를 한다는 것이다. 한국에서 발신이 되는 사기(Fraud) 통화는 주로 IP-PBX 등이 해킹이 되어 다량의 사기(Fraud) 통화가 이루어지는 경우가 많다고 한다.

<표 7> 사기(Fraud) 통화 발신 국가 상위 30

국가	통화	사기	비율	국가	통화	사기	비율
NAURU	11	11	100.00%	BOSNIA-HERCEGOVINA	1	1	100.00%
BENIN	5	5	100.00%	NEW CALEDONIA	1	1	100.00%
MONTENEGRO	4	4	100.00%	KYRGYZSTAN	1	1	100.00%
INT NETWORKS	4	4	100.00%	BURKINA FASO	259	220	84.94%
SOMALIA	3	3	100.00%	UNITED ARAB EMIRATES	10	7	70.00%
BELARUS	3	3	100.00%	NETHERLANDS	1,969	1,306	66.33%
SLOVENIA	3	3	100.00%	IRAN	13	8	61.54%
TAJISKISTAN	3	3	100.00%	KIRIBATI	5	3	60.00%
South Sudan	2	2	100.00%	MOZAMBIQUE	2	1	50.00%
COTE D IVOIRE (IVORY COAST)	2	2	100.00%	GUERNSEY	28	13	46.43%
GUINEA BISSAU	2	2	100.00%	02 지역번호	82	38	46.34%
SEYCHELLES ISLAND	2	2	100.00%	ANTIGUA	1,457,957	382,336	26.22%
LITHUANIA	1	1	100.00%	KOREA	2,590,041	625,882	24.16%
MOLDOVA	1	1	100.00%	PERU	51	7	13.73%
SERBIA	1	1	100.00%	ARGENTINA	47	6	12.77%

다음은 착신국 상위 60개 나라<표 9>를 보면 사기(Fraud) 통화의 대부분이 아프리카 국가(노란색 음영)이며, 상위 30에는 25개의 아프리카 국가가 분포되어 있다. 아시아 국가는 말레이시아, 카자흐스탄, 파키스탄, 말레이시아, 베트남, 시리아, 한국 아프가니스탄 등이 착신 국가 상위 60에 포함이 되어 있으며, 한국은 57번째 위치하고 있다. 한국은 국내 CSP의 많은 노력에도 불구하고 사기(Fraud) 통화의 상위 28번째 발신 국가, 상위 57번째 착신 국가에 위치하고 있어 사기(Fraud) 통화에 안전하지 않다.

<표 8> 사기(Fraud) 통화 착신 국가 상위 60

국가	통화	사기	비율	국가	통화	사기	비율
CHAD	305	264	86.56%	KAZAKHSTAN	24,435	11,093	45.40%
DR.Congo	574	480	83.62%	CUBA	928	420	45.26%
CENTRAL AFRICAN REPUBLIC	331	276	83.38%	MAURITANIA	318	136	42.77%
CONGO	300	249	83.00%	GUINEA REPUBLIC	5,593	2,318	41.44%
BENIN	231	191	82.68%	South Sudan	128	52	40.63%
GAMBIA	407	331	81.33%	GHANA	1,071	411	38.38%
SOMALIA	328	259	78.96%	ETHIOPIA	2,021	775	38.35%
SENEGAL	1,829	1,434	78.40%	SOUTH AFRICA	1,399	532	38.03%
AUSTRIA	5,700	4,383	76.89%	ASCENSION ISLAND	82	30	36.59%
IRAN	12,223	9,256	75.73%	ANTIGUA	281,529	97,634	34.68%
GUINEA BISSAU	64	48	75.00%	PAKISTAN	33,860	11,485	33.92%
DIEGO GARCIA	118	88	74.58%	ALGERIA	1,164	389	33.42%
SUDAN	2,625	1,901	72.42%	MYANMAR	325,968	106,549	32.69%
SIERRA LEONE	443	307	69.30%	SWITZERLAND	3,051	997	32.68%
TUNISIA	1,177	789	67.03%	EQUATORIAL GUINEA	326	105	32.21%
COTE D IVOIRE (IVORY COAST)	393	253	64.38%	VIETNAM	36,713	11,401	31.05%
NIGERIA	19,690	11,424	58.02%	COLOMBIA	799	243	30.41%
AUSTRALIA	30,817	17,632	57.22%	SRI LANKA	23,274	6,842	29.40%
#N/A	381	205	53.81%	CAPE VERDE ISLANDS	62	18	29.03%
LIBYA	1,292	689	53.33%	NETHERLANDS	24,273	6,771	27.90%
RWANDA	214	114	53.27%	THAILAND	163,653	45,448	27.77%
MOROCCO	1,011	526	52.03%	BELGIUM	1,764	479	27.15%
ANGOLA	294	152	51.70%	TOGO	175	47	26.86%
SAO TOME	53	27	50.94%	SEYCHELLES ISLAND	209	56	26.79%
LIBERIA	509	259	50.88%	#N/A	89,399	23,737	26.55%
CAMEROON	325	160	49.23%	MAURITIUS	159	42	26.42%
MALI REPUBLIC	289	138	47.75%	KOREA	1,466,993	384,073	26.18%
MALAYSIA	15,644	7,398	47.29%	EGYPT	8,859	2,241	25.30%
BURKINA FASO	195	89	45.64%	AFGHANISTAN	987	246	24.92%
GABON REPUBLIC	160	73	45.63%	NIGER REPUBLIC	97	24	24.74%

<표 9>은 한국에서 발신된 Fraud 통화의 지역 및 유형별 분포에 대한 수치를 보여준다. 전체 166만 6천여 건 중 유/무선 유형 중 무선(핸드폰) 발신이 64.53%(65만 건)를 점유하고 있다. 'Lower'와 'Upper'의 의미는 발신번호를 변작하여 기본적인 9~10자리 숫자보다 적거나, 많은 번호로 발신된 경우를 뜻한다. 이런 발신 번호의 오류[17]로 인한 부분이 많을 것으로 판단을 했으나, 이 부분은 전체 사기(Fraud)통화의 2.85%로 나타났다. 지역별로는 서울/경기에서 가장 많은 Fraud가 발생하였다.

<표 9> 지역/통화유형별 사기(Fraud) 통화 분포

Type	Fraud
핸드폰	650,897
서울	124,019
경기	73,073
충북	29,117
070	25,855
전남	16,434
Lower	15,933
인천	13,739
문자	11,870
대전	9,425
부산	7,792
경북	4,968
울산	4,763
광주	3,787
대구	3,656
경남	3,022
전북	3,012
강원	2,288
제주	2,160
충남	1,807
Upper	906
세종	121

이처럼 핸드폰과 수도권에 집중된 이유는 통신 네트워크 및 인프라가 있어야 Fraud를 발생시킬 수 있기 때문이다. 또한, 과거 070으로 시작되는 번호에서 많이 발생하던 Fraud 통화가 사용자 환경의 변화 및 기술의 변화에 따라 VoIP 전화보다 핸드폰 번호로 Fraud 통화 발신이 많이 발생한 것으로 판단된다. 예를 들어 사기 유형 중인 하나인 SIM BOX Fraud의 경우 SIM 상자에 서로 다른 이동통신사의 SIM 카드가 설치되어 있어 동시에 Fraud 통화를 발생을 시키는 사기 유형으로 이동통신사의 전화번호로

17) 발신 번호에 문자 혹은 전화번호 규칙에 따른 숫자의 개수가 많거나 적은 발신 번호

발생이 이루어진다.

다음으로 통화 유형(유선, 무선, 국제통화) Fraud 통화 분포<표 11>를 확인해 보면 발신 및 착신에 따라 서로 다른 분포를 보여준다.

먼저 발신 유형을 보면 사기(Fraud) 통화비율은 유선에서 높게 나타나고 있으나 통화 건수는 무선과 국제통화의 수가 압도적으로 많으며, 전체 통화를 기준으로 보면 발신의 경우 무선에서 사기(Fraud) 통화가 많이 발생한다.

착신 유형은 유선, 국제, 무선의 순으로 높게 분포되어 있으나, 사기(Fraud) 통화의 경우 국제와 유선의 비율이 높게 나타난다. 전체 통화를 기준으로는 유선에서 사기(Fraud) 통화가 많이 발생한다. 이와 같은 현상은 <표 10>지역/통화유형별 사기(Fraud) 통화 분포에서 언급했듯이 네트워크와 인프라의 구성 및 사용자 환경의 변화와 기술의 변화에 따라 발신 착신의 Fraud 통화 유형이 다르게 나타나는 것으로 판단된다.

<표 10> 통화 유형에 따른 사기(Fraud) 통화 분포

	발신				착신			
	통화	사기	개별%	전체%	통화	사기	개별%	전체%
유선	43,703	14,130	32.33%	0.35%	1,769,453	461,517	26.08%	11.38%
무선	2,415,996	686,605	28.42%	16.93%	1,076,026	198,179	18.42%	4.89%
국제	1,490,587	283,203	19.00%	6.98%	1,210,159	350,198	28.94%	8.63%
unknown	105,440	25,964	24.62%	0.64%	88	8	9.09%	0.00%

마지막으로 통화 사용시간에 따라 Fraud 통화 분포<표 11>를 확인해 보면 전체 통화의 48.42%가 통화시간이 0초인 통화이며, 그중 31.65%가 Fraud 통화에 해당된다. 전체 통화에서 0초를 사용한 Fraud 통화는 15.33%에 해당된다. 이런 현상은 Fraud 통화 시도자가 불법적인 통화 루

트를 찾는 과정에서 나타나는 현상으로 판단된다. 대부분의 통화시간은 6분 안에 이루어지며 사기(Fraud) 통화도 이 시간 안에 다수 나타나는 것으로 보인다.

<표 11> 통화 사용 시간량에 따른 사기(Fraud) 통화 분포

분	통화	사기	개별 %	전체 %
0	1,963,957	621,667	31.65%	15.33%
3	1,322,406	279,462	21.13%	6.89%
6	307,154	48,872	15.91%	1.21%
9	162,997	22,598	13.86%	0.56%
12	104,945	13,698	13.05%	0.34%
15	71,222	8,651	12.15%	0.21%
18	32,548	4,103	12.61%	0.10%
21	22,694	2,870	12.65%	0.07%
24	16,520	1,976	11.96%	0.05%
27	11,826	1,431	12.10%	0.04%
30	9,393	1,100	11.71%	0.03%
33	7,134	858	12.03%	0.02%
36	5,457	754	13.82%	0.02%
39	4,046	431	10.65%	0.01%
42	3,229	348	10.78%	0.01%
45	2,458	263	10.70%	0.01%
48	2,114	225	10.64%	0.01%
51	1,893	185	9.77%	0.00%
54	1,412	160	11.33%	0.00%
57	1,142	97	8.49%	0.00%
60	1,116	149	13.35%	0.00%
63	51	4	7.84%	0.00%
66	1	-	0.00%	0.00%
69	11	-	0.00%	0.00%

지금까지 통화의 타입, 월별, 일별, 국가별, 국내 지역별, 유형별, 시간별 Fraud 발생에 대해 EDA를 진행하였다. 간단히 요약하면 아래 9가지 특징으로 정리할 수 있다.

① 평균적으로 Fraud 통화는 전체 통화량 대비 25% 전후로 발생

② 오전 4시부터 6시까지 Fraud 통화가 집중적으로 발생

③ Fraud 통화의 발신 국가는 불특정 다수로 발생

④ Fraud 통화의 착신 국가는 대부분 아프리카 국가로 집중

⑤ Fraud 통화의 대부분은 핸드폰으로 발신됨

⑥ 상위 발신 및 착신 Fraud 통화의 대부분 국가의 경우 CSP에서 정한 위험 관리 국가에 속함

⑦ 또한 과금 요율이 높은 국가이거나, Fraud 탐지를 잘 하지 않은 취약 지역으로 집중됨

⑧ Fraud 통화의 경우 발신은 무선(모바일), 착신은 유선의 비율이 높음

⑨ 통화 유형 개별로 보면 유선과 국제통화에서 Fraud 비율이 높

제 3 절 지도 학습 수행

(1) 수행 과정

국내의 통화 요금은 아웃-바운드 통화에만 요금이 부과되며, CSP의 손실에 영향을 주는 통화는 아웃-바운드 통화이다. 따라서 앞선 과정을 통해 준비된 데이터 중 아웃-바운드 통화만 지도 학습을 수행하였다. 또한, 앞선 전처리(EDA) 과정에서 본 것과 같이 사기(Fraud) 통화가 많이 분포한 상위 60개국 착신 국가의 통화 기록을 중점으로 분석을 하였다. 지도 학습을 수행할 데이터 셋을 만들기 위해 세 단계로 처리했다.

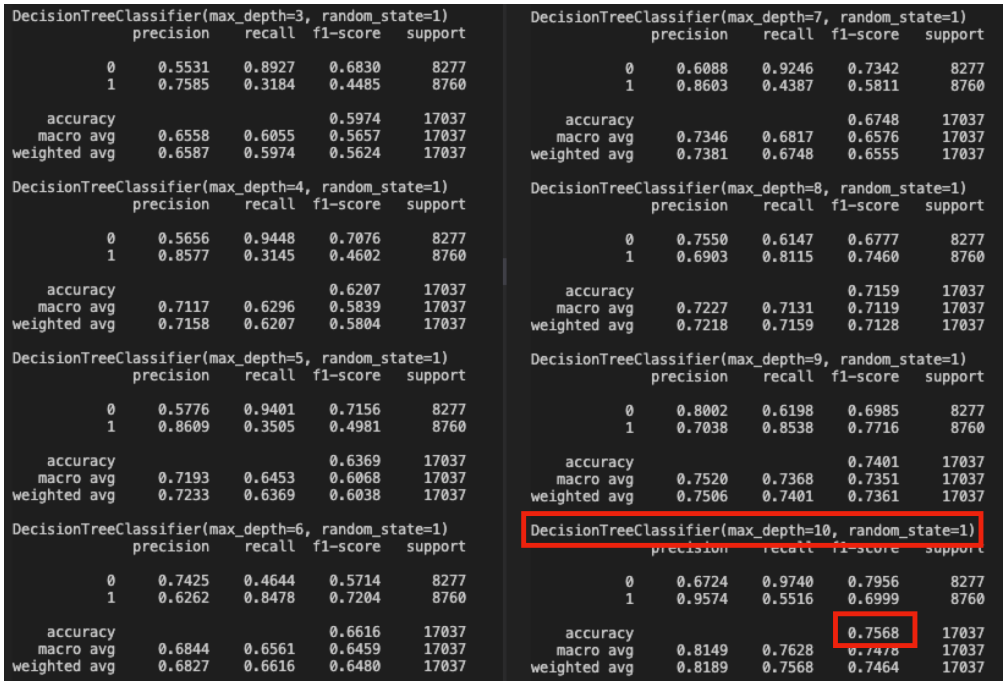
첫 번째, 지도 학습 모델에 적합하도록 feature 선정 및 변환 과정이다. 분석에 사용한 Feature 전처리에서 사용한 동일한 11개의 Feature와 사기(Fraud) 여부를 표시하는 1개의 feature를 추가하여 총 12개로 선정했다. Feature 중 통화시간(Call duration)은 3분, 10분, 1시간 단위로 범주(category)형으로 구분을 하여 처리했다. 통화 시간(Call duration)을 포함한 범주(category)형 feature들은 통계학습 모델에 적합하도록 원핫인코딩(One-Hot-Encoding)로 변환했다.

두 번째, 데이터를 학습 셋(training set), 검증 셋(validation set), 테스트 셋(test set) 3개로 50%, 20%, 20%로 분할하였다. 또한, 지도 학습 분류 모델별로 최적의 하이퍼 파라미터(Hyper-parameter)를 찾기 위해 검증 셋(validation set)을 사용했다. 학습한 모듈의 성능 검증에 있어 테스트 셋(test set)을 사용했으며, 최종적으로는 학습 셋(training set)과 검증 셋(validation set)을 합하여 다시 한번 모듈의 성능을 검증하였다. 최종적으로 성능에 대한 검증은 AUROC 값을 사용하였다.

세 번째, 지도 학습의 분류(Classification) 모델들을 활용하여 최적의 모델을 확인한다. 분류 모델 총 5가지를 사용했으며, 사용한 5가지 모델은 Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbors, Linear Support Vector이다.

(2) Decision Tree Classifier

모델 학습 시 사용한 트리의 최대 깊이는 3에서 10까지 총 8개의 깊이를 사용했으며, 불순도 함수는 Gini 함수를 사용했다. 모델을 학습한 결과 10단계의 깊이로 분석을 하였을 때 정확도는 0.768로 가장 높게 나왔다. <그림 8>은 학습 셋(training set), 검증 셋(validation set)으로 학습을 한 결괏값이다.



DecisionTreeClassifier(max_depth=3, random_state=1)

	precision	recall	f1-score	support
0	0.5531	0.8927	0.6830	8277
1	0.7585	0.3184	0.4485	8760
accuracy			0.5974	17037
macro avg	0.6558	0.6055	0.5657	17037
weighted avg	0.6587	0.5974	0.5624	17037

DecisionTreeClassifier(max_depth=4, random_state=1)

	precision	recall	f1-score	support
0	0.5656	0.9448	0.7076	8277
1	0.8577	0.3145	0.4602	8760
accuracy			0.6207	17037
macro avg	0.7117	0.6296	0.5839	17037
weighted avg	0.7158	0.6207	0.5804	17037

DecisionTreeClassifier(max_depth=5, random_state=1)

	precision	recall	f1-score	support
0	0.5776	0.9401	0.7156	8277
1	0.8609	0.3505	0.4981	8760
accuracy			0.6369	17037
macro avg	0.7193	0.6453	0.6068	17037
weighted avg	0.7233	0.6369	0.6038	17037

DecisionTreeClassifier(max_depth=6, random_state=1)

	precision	recall	f1-score	support
0	0.7425	0.4644	0.5714	8277
1	0.6262	0.8478	0.7204	8760
accuracy			0.6616	17037
macro avg	0.6844	0.6561	0.6459	17037
weighted avg	0.6827	0.6616	0.6480	17037

DecisionTreeClassifier(max_depth=7, random_state=1)

	precision	recall	f1-score	support
0	0.6088	0.9246	0.7342	8277
1	0.8603	0.4387	0.5811	8760
accuracy			0.6748	17037
macro avg	0.7346	0.6817	0.6576	17037
weighted avg	0.7381	0.6748	0.6555	17037

DecisionTreeClassifier(max_depth=8, random_state=1)

	precision	recall	f1-score	support
0	0.7550	0.6147	0.6777	8277
1	0.6903	0.8115	0.7460	8760
accuracy			0.7159	17037
macro avg	0.7227	0.7131	0.7119	17037
weighted avg	0.7218	0.7159	0.7128	17037

DecisionTreeClassifier(max_depth=9, random_state=1)

	precision	recall	f1-score	support
0	0.8002	0.6198	0.6985	8277
1	0.7038	0.8538	0.7716	8760
accuracy			0.7401	17037
macro avg	0.7520	0.7368	0.7351	17037
weighted avg	0.7506	0.7401	0.7361	17037

DecisionTreeClassifier(max_depth=10, random_state=1)

	precision	recall	f1-score	support
0	0.6724	0.9740	0.7956	8277
1	0.9574	0.5516	0.6999	8760
accuracy			0.7568	17037
macro avg	0.8149	0.7628	0.7478	17037
weighted avg	0.8189	0.7568	0.7464	17037

<그림 8> Decision Tree 학습결과

(3) Random Forest Classifier

모델 학습 시 하이퍼 파라미터 n_estimators, max_features을 고려하였다. n_estimators는 tree의 개수를 의미하는 것으로 5에서 50까지 5의 배수로 총 10종류의 트리 개수를 학습하였다. 또한, 최상의 분할을 찾을 때 고려해야 할 features 수 (n_estimators)는 'auto'. 'log2'을 사용했다. 여기서 "auto"의 features수는 sqrt(n_features)이며 'sqrt'와 동일하다. 'log2', 의 features수는 log2(n_features) 이다. 모델을 학습한 결과

n_estimators=25, max_features='auto'로 했을 때 정확도 0.911로 가장 높게 나왔다. <그림 9>는 학습 셋(training set), 검증 셋(validation set)으로 학습을 한 결괏값이다.

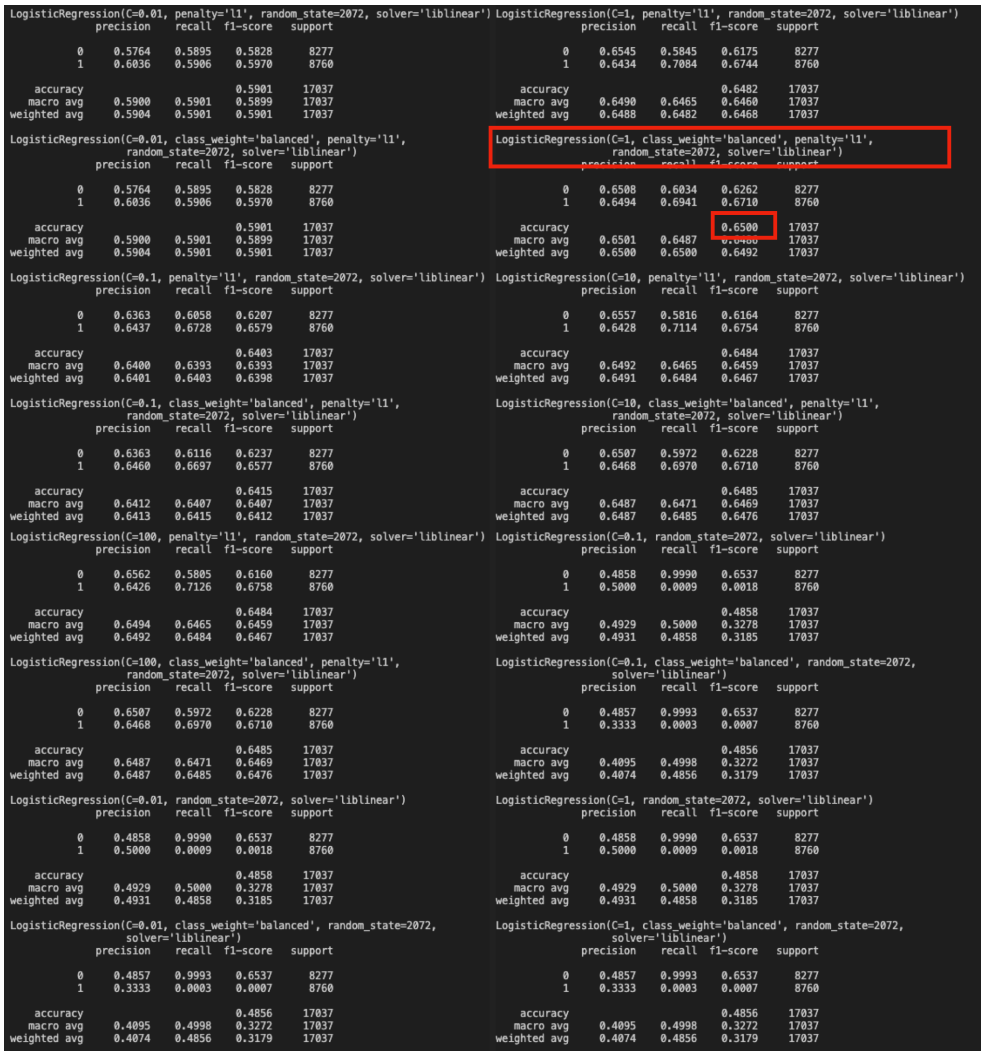
```
RandomForestClassifier(n_estimators=5, random_state=123)
              precision    recall  f1-score   support

           0     0.8921    0.9015    0.8968      8277
           1     0.9060    0.8969    0.9014      8760

    accuracy                         0.8992     17037
   macro avg     0.8990    0.8992    0.8991     17037
weighted avg     0.8992    0.8992    0.8992     17037

RandomForestClassifier(max_features='log2', n_estimators=5, random_state=123)
              precision    recall  f1-score   support

           0     0.8860    0.9001    0.8930      8277
           1     0.9041    0.8905    0.8973      8760

    accuracy                         0.8952     17037
   macro avg     0.8951    0.8953    0.8951     17037
weighted avg     0.8953    0.8952    0.8952     17037

RandomForestClassifier(n_estimators=10, random_state=123)
              precision    recall  f1-score   support

           0     0.8891    0.9259    0.9071      8277
           1     0.9272    0.8909    0.9087      8760

    accuracy                         0.9079     17037
   macro avg     0.9081    0.9084    0.9079     17037
weighted avg     0.9087    0.9079    0.9079     17037

RandomForestClassifier(max_features='log2', n_estimators=10, random_state=123)
              precision    recall  f1-score   support

           0     0.8797    0.9194    0.8991      8277
           1     0.9205    0.8812    0.9004      8760

    accuracy                         0.8997     17037
   macro avg     0.9001    0.9003    0.8997     17037
weighted avg     0.9006    0.8997    0.8998     17037

RandomForestClassifier(n_estimators=25, random_state=123)
              precision    recall  f1-score   support

           0     0.9020    0.9184    0.9101      8277
           1     0.9216    0.9057    0.9136      8760

    accuracy                         0.9119     17037
   macro avg     0.9118    0.9121    0.9119     17037
weighted avg     0.9121    0.9119    0.9119     17037

RandomForestClassifier(max_features='log2', n_estimators=25, random_state=123)
              precision    recall  f1-score   support

           0     0.8955    0.9118    0.9036      8277
           1     0.9152    0.8994    0.9072      8760

    accuracy                         0.9054     17037
   macro avg     0.9053    0.9056    0.9054     17037
weighted avg     0.9056    0.9054    0.9055     17037

RandomForestClassifier(n_estimators=30, random_state=123)
              precision    recall  f1-score   support

           0     0.8959    0.9204    0.9080      8277
           1     0.9228    0.8990    0.9107      8760

    accuracy                         0.9094     17037
   macro avg     0.9093    0.9097    0.9094     17037
weighted avg     0.9097    0.9094    0.9094     17037

RandomForestClassifier(max_features='log2', n_estimators=30, random_state=123)
              precision    recall  f1-score   support

           0     0.8910    0.9187    0.9046      8277
           1     0.9209    0.8938    0.9071      8760

    accuracy                         0.9059     17037
   macro avg     0.9059    0.9063    0.9059     17037
weighted avg     0.9064    0.9059    0.9059     17037

RandomForestClassifier(n_estimators=15, random_state=123)
              precision    recall  f1-score   support

           0     0.8973    0.9164    0.9068      8277
           1     0.9194    0.9009    0.9101      8760

    accuracy                         0.9084     17037
   macro avg     0.9083    0.9087    0.9084     17037
weighted avg     0.9087    0.9084    0.9085     17037

RandomForestClassifier(max_features='log2', n_estimators=15, random_state=123)
              precision    recall  f1-score   support

           0     0.8942    0.9097    0.9019      8277
           1     0.9133    0.8983    0.9057      8760

    accuracy                         0.9039     17037
   macro avg     0.9037    0.9040    0.9038     17037
weighted avg     0.9040    0.9039    0.9039     17037

RandomForestClassifier(n_estimators=20, random_state=123)
              precision    recall  f1-score   support

           0     0.8936    0.9229    0.9080      8277
           1     0.9248    0.8961    0.9103      8760

    accuracy                         0.9091     17037
   macro avg     0.9092    0.9095    0.9091     17037
weighted avg     0.9096    0.9091    0.9092     17037

RandomForestClassifier(max_features='log2', n_estimators=20, random_state=123)
              precision    recall  f1-score   support

           0     0.8892    0.9193    0.9040      8277
           1     0.9212    0.8918    0.9063      8760

    accuracy                         0.9051     17037
   macro avg     0.9052    0.9055    0.9051     17037
weighted avg     0.9057    0.9051    0.9052     17037

RandomForestClassifier(n_estimators=35, random_state=123)
              precision    recall  f1-score   support

           0     0.8987    0.9177    0.9081      8277
           1     0.9207    0.9023    0.9114      8760

    accuracy                         0.9098     17037
   macro avg     0.9097    0.9100    0.9098     17037
weighted avg     0.9100    0.9098    0.9098     17037

RandomForestClassifier(max_features='log2', n_estimators=35, random_state=123)
              precision    recall  f1-score   support

           0     0.8934    0.9141    0.9036      8277
           1     0.9170    0.8969    0.9069      8760

    accuracy                         0.9053     17037
   macro avg     0.9052    0.9055    0.9052     17037
weighted avg     0.9055    0.9053    0.9053     17037

RandomForestClassifier(n_estimators=40, random_state=123)
              precision    recall  f1-score   support

           0     0.8953    0.9204    0.9077      8277
           1     0.9227    0.8983    0.9103      8760

    accuracy                         0.9090     17037
   macro avg     0.9090    0.9093    0.9090     17037
weighted avg     0.9094    0.9090    0.9090     17037

RandomForestClassifier(max_features='log2', n_estimators=40, random_state=123)
              precision    recall  f1-score   support

           0     0.8908    0.9176    0.9040      8277
           1     0.9199    0.8937    0.9066      8760

    accuracy                         0.9053     17037
   macro avg     0.9053    0.9057    0.9053     17037
weighted avg     0.9057    0.9053    0.9053     17037
```

<그림 9> Random Forest 학습결과

(4) Logistic Regression Classifier

모델 학습 시 하이퍼 파라미터 penalty, C 및 class_weight을 고려하였다. penalty는 'l1', 'l2', 파라미터 'C'는 정규화(regularization)의 강도를

결정하는 것으로 총 5가지 값(0.01, 0.1, 1, 10, 100)으로 학습했다. class 가중치는 y 값을 사용하여 n_samples / (n_classes * np.bincount(y))로 입력 데이터의 클래스 빈도에 반비례하는 가중치를 자동으로 조정하는 'balanced' 모드와 사용하지 않는 None', 모두 2가지를 사용했다. 모델을 학습한 결과 C=1, class_weight='balanced', penalty='l1'로 했을 때 정확도 0.641로 가장 높게 나왔다. <그림 10>은 학습 셋(training set), 검증 셋(validation set)으로 학습을 한 결괏값이다.



<그림 10> Logistic Regression 학습결과

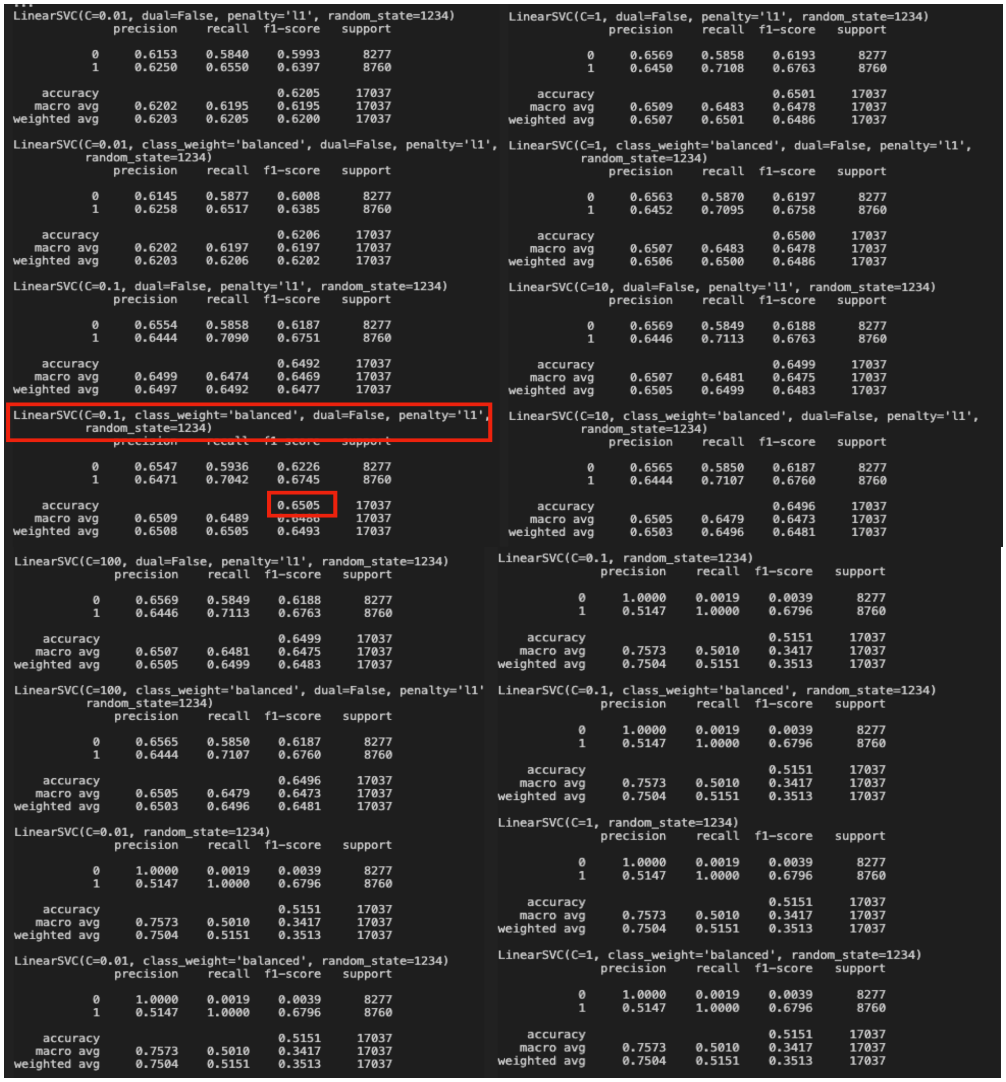
(5) K-Nearest Neighbors Classifier

모델 학습 시 하이퍼 파라미터 n_neighbors, weights을 고려하였다. 이웃의 수는 10부터 60까지 5의 배수로 총 10가지를 학습했다. 또한, 가중치는 이웃의 가중치를 동일하게 하는 'uniform'과 거리의 역수를 가중치로 부여하는 'distance'를 사용했다. 모델을 학습한 결과 n_neighbors=15, weights='distance'로 했을 때 정확도 0.976으로 가장 높게 나왔다. <그림 11>은 학습 셋(training set), 검증 셋(validation set)으로 학습을 한 결괏값이다.

<그림 11> K-Nearest Neighbors 학습결과

(6) Linear Support Vector Classifier

모델 학습 시 하이퍼 파라미터 penalty, C 및 class_weight을 고려하였으며, 값들은 Logistic Regression 값과 동일하게 하여 학습을 하였다. 모델을 학습한 결과 C=0.1, class_weight='balanced', penalty='l1'로 했을 때 정확도 0.650으로 가장 높게 나왔다. <그림 12>는 학습 셋(training set), 검증 셋(validation set)으로 학습을 한 결괏값이다.



<그림 12> Linear Support Vector 학습결과

검증 셋(validation set)으로 모델별 주요 파라미터와 값들을 넣어 최적의 하이퍼 파라미터(Hyper-parameter)를 구하였다. <표 13>은 모델별 정확도가 높은 파라미터이다.

<표 12> 모델별 검증 셋(validation set) 학습 정확도

Model	Hyper-parameter	accuracy
Decision Tree	DecisionTreeClassifier(max_depth=10, random_state=1)	0.7568
Random Forest	RandomForestClassifier(n_estimators=25, random_state=123)	0.9118
Logistic Regression	LogisticRegression(C=1, class_weight='balanced', penalty='l1', random_state=2072, solver='liblinear')	0.6499
K-Nearest Neighbors	KNeighborsClassifier(n_jobs=-1, n_neighbors=15, weights='distance')	0.9774
Linear Support Vector	LinearSVC(C=0.1, class_weight='balanced', dual=False, penalty='l1', random_state=1234)	0.6504

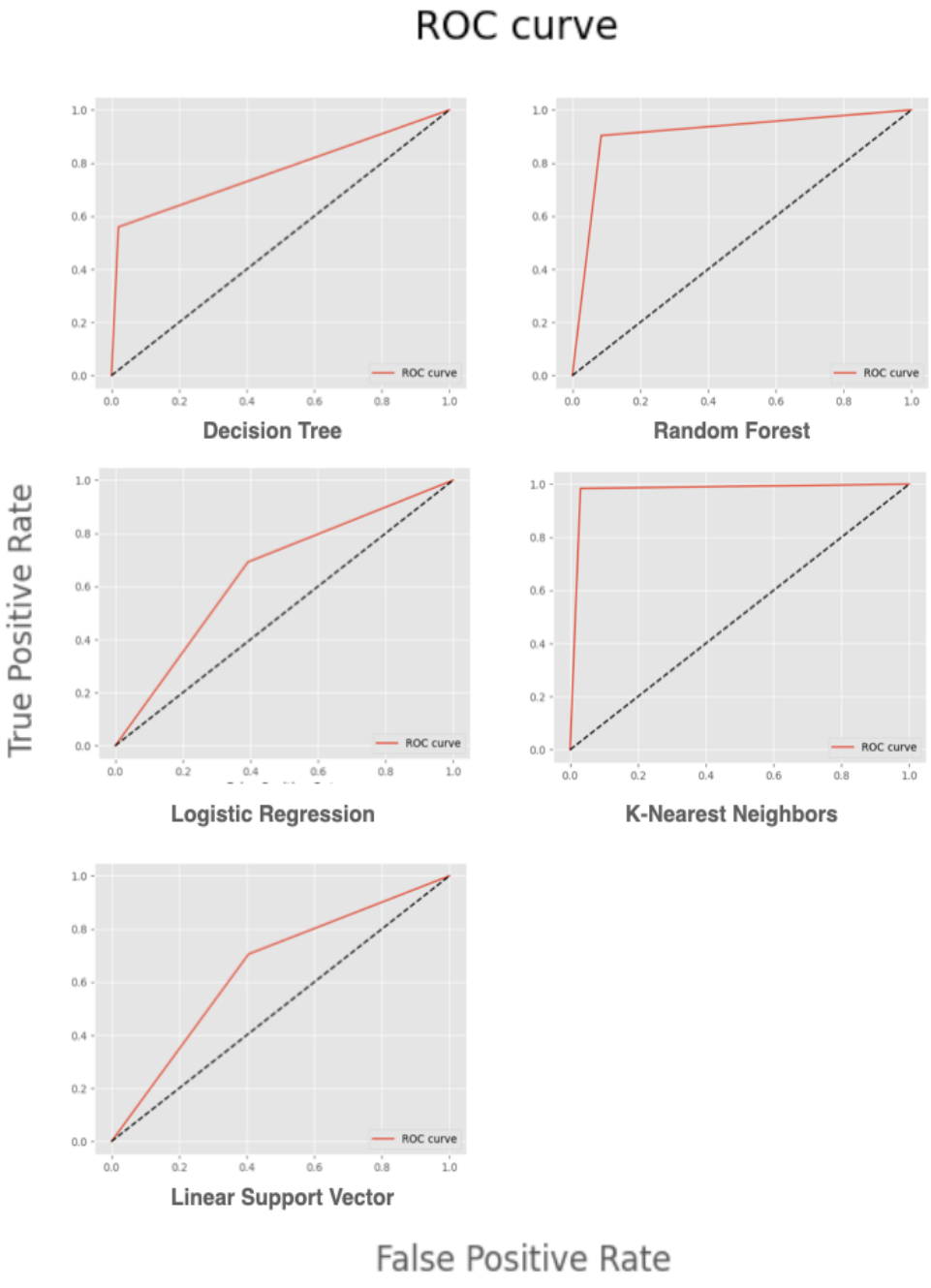
제 Ⅳ장 분석 결과

제 1절 테스트 셋(Test Set) 검증 결과

앞서 검증 셋(validation set)으로 학습 한 모듈을 모델별로 테스트 셋(test set)으로 검증을 하였다. <표 14>는 모듈별 테스트 셋(test set)을 검증한 정확도와 AUROC 값으로, K-Nearest Neighbors 0.9774, Random Forest 0.9091, Decision Tree 0.7658, Linear Support Vector 0.6508, Logistic Regression 0.6502 순으로 나타났다.

<표 13> 모델별 테스트 셋(test set) 학습 정확도

Model	validation set	Test Set	
	accuracy	accuracy	AUROC
Decision Tree	0.7568	0.7658	0.7694
Random Forest	0.9118	0.9091	0.9092
Logistic Regression	0.6499	0.6502	0.6495
K-Nearest Neighbors	0.9774	0.9766	0.9765
Linear Support Vector	0.6504	0.6508	0.6498



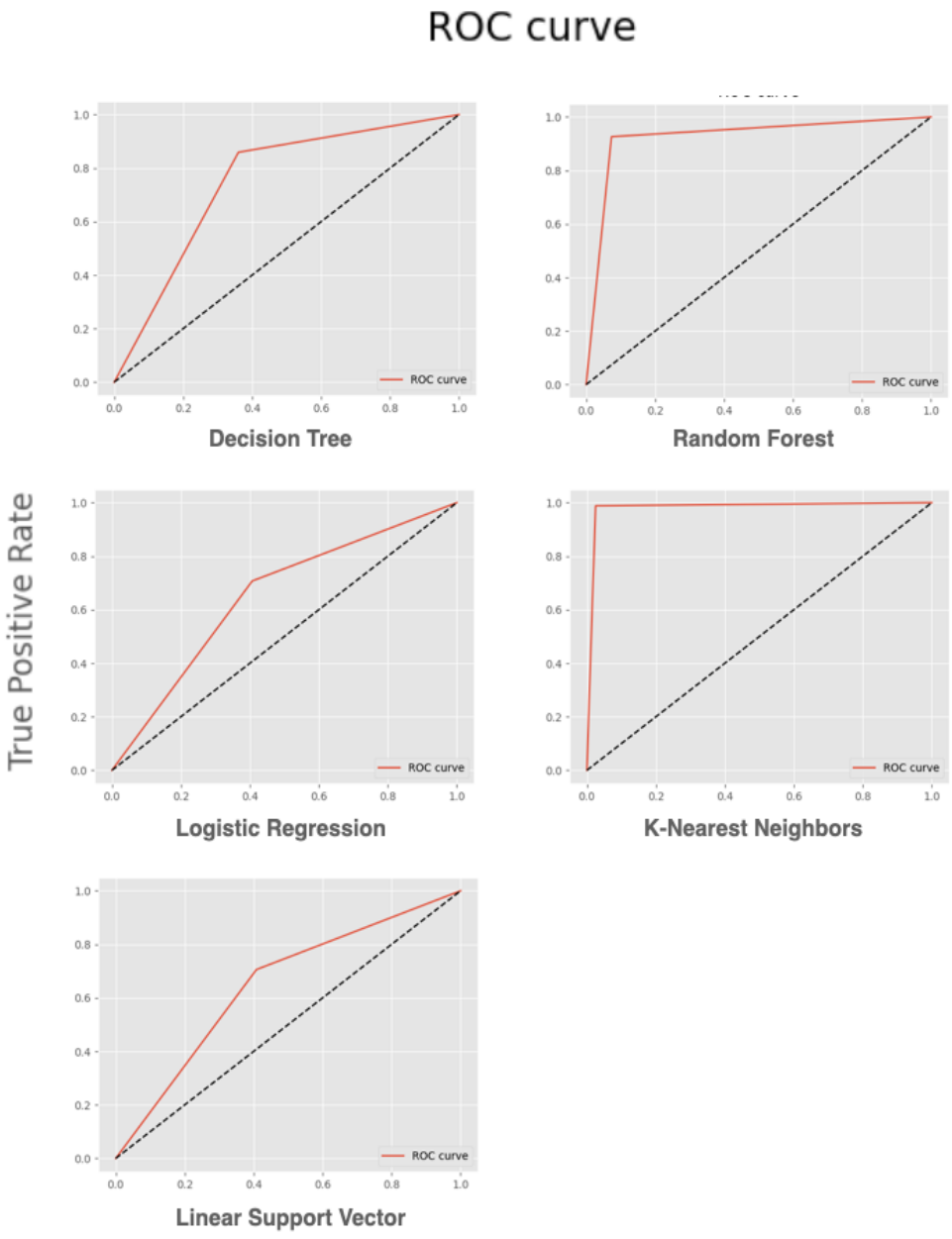
<그림 13> 모델별 테스트 셋(test set) ROC Curve

제 2절 최종 학습 검증 결과

앞서 검증 셋(validation set)으로 사용한 데이터를 학습 셋(training set)과 하나로 합쳐서 다시 검증하였다. <표 15>는 학습 셋(training set)과 검증 셋(validation set)을 합쳐서 모델별 검증한 정확도와 AUROC 값으로, K-Nearest Neighbors 0.9815, Random Forest 0.9263, Decision Tree 0.7494, Linear Support Vector 0.6488, Logistic Regression 0.6502 순으로 나타났다. 합치기 전 테스트 셋(Test Set)으로 검증한 값보다 다소 정확도 및 AUROC값이 높아졌다.

<표 14> 최종 학습 결과 정리

Model	validation set	test set		validation + test set	
	accuracy	accuracy	AUROC	accuracy	AUROC
Decision Tree	0.7568	0.7658	0.7694	0.7512	0.7494
Random Forest	0.9118	0.9091	0.9092	0.9263	0.9263
Logistic Regression	0.6499	0.6502	0.6495	0.6511	0.6502
K-Nearest Neighbors	0.9776	0.9766	0.9765	0.9816	0.9815
Linear Support vector	0.6504	0.6508	0.6498	0.6497	0.6488



<그림 14> 최종 학습 검증 결과 ROC Curve

제 Ⅴ장 결 론

제 1절 연구의 요약 및 시사점

앞에서 언급했듯이 기술발달 즉 IP 네트워크 인프라 및 IP 네트워크 통신 시스템(IP-PBX, SBC 등)의 발달과 함께 인터넷을 이용한 다양한 통화 서비스의 발전과 동시에 사기(Fraud) 통화 또한 다양해지고 지능화되고 있다. 이렇게 다양해지고 지능화되는 사기(Fraud) 통화를 검출하기에는 기존의 임계치를 이용한 Rule 기반의 CSP 사기(Fraud) 검출 흐름도<그림 1>로는 사기(Fraud) 통화를 검출하기에는 역부족이다. <그림 1>의 검출흐름도에서 기존의 도매인 관리자가 처리하던 '분석 단계'와 '적용단계'를 머신러닝 지도학습 알고리즘을 통하여 <그림 2>와 같이 하나의 '분석 단계'로 처리함으로써 도매인 관리자가 분석결과를 보고 Rule 생성 혹은 Rule 업데이트 과정이 불필요하게 된다. 또한, 서비스의 다양화 및 사기(Fraud) 유형의 정교함, 다양성 및 발전 증가에 대해 도메인 관리자가 즉각적인 대응을 할 수가 있어 사기(Fraud) 통화 증가로 지속적으로 발생을 하는 CSP 손실을 줄일 수 있는 방안이 될 것이란 생각으로 본 연구를 진행하였다.

사기(Fraud) 예측모델 연구는 다른 기계 학습 예측 모델 학습과 마찬가지로 다음과 같이 진행하였다. 첫 번째는 사기(Fraud) 통화 검출에 가장 기본 데이터가 되는 통화 데이터(CDR)를 지도 학습 모델에 적합하도록 feature 선정 및 변환 과정이다. 통화에 가장 기본이 되고 반드시 있어야 하는 feature를 선정하고 레이블 작업을 하였으며, 범주형 feature의 경우 원핫인코딩으로 지도학습 분류모델에 적합하도록 변환 작업을 수행하였다. 여기서 레이블 작업은 사기(Fraud) 검출 시스템에서 검출된 사기를 참고하였다.

두 번째, 학습하여 비교할 지도 학습의 분류 모델을 선정하고, 각 선정된 분류 모델 최적의 하이퍼 파라미터를 찾는 작업을 수행하였다. 사용한 분류 모델은 Decision Tree Classifier, Random Forest Classifier, Logistic Regression Classifier, K-Nearest Neighbors Classifier, Linear Support Vector Classifier로 각 모듈 별 학습을 수행하여 모델을 만들었으며, 최종적으로 테스트 데이터로 검증을 하였다. 여기서 나온 결과를 비교하여 사기(Fraud)에 적합한 기계학습 모델을 선정하였다.

연구를 통해 12개의 feature와 최적의 하이퍼 파라미터를 기반으로 하는 K-Nearest Neighbors Classifier, Random Forest Classifier가 90%가 넘는 정확도를 보였다. 앞에서 언급을 했듯이 실제 CSP의 통화 데이터(CDR)를 기반으로 분석을 한 결과로써 실제 전화 통신망에서 적용할 수도 있을 것이다. 다만 향후 사기(Fraud) 예측 연구에서는 본 연구를 토대로 feature 엔지니어링 기법을 이용하여 통화 데이터(CDR)의 새로운 feature를 포함한 분석과 통화 데이터(CDR) 외 다양한 데이터들 포함하여 탐색 범위를 넓히는 것이 필요해 보인다.

마지막으로 실제 전화 통신망에 적용한다면 본 연구에서 기대한 것처럼 사기(Fraud) 유형의 정교함, 다양성 및 발전 증가에 대해 도메인 관리자가 즉각적인 대응을 할 수가 있어 사기(Fraud) 통화 증가로 지속적으로 발생을 하는 CSP 손실을 줄일 수 있을 것으로 기대된다.

제 2절 연구의 한계 및 향후 연구 과제

앞의 데이터 전처리 단계에서 언급했듯이 학습에 사용한 데이터는 CSP의 통화 상세기록(CDR) 3개월간의 데이터를 사용했다. 아웃바운드 국제전화 통화로 구성된 통화 상세기록(CDR) 데이터를 지도 학습의 분류

모델을 이용했다. 또한, 통화 상세기록(CDR)의 300개가 넘는 Feature 중 사기(Fraud)검출을 위해 반드시 있어야 하는 최소 Feature만을 사용했다. 이처럼 일부 데이터만 사용하여 다양한 사기(Fraud) 시도 시나리오의 대응에는 한계가 있으며, 이를 해결하기 위해서는 다음과 같은 후속 연구가 필요하다.

첫째, 통화 상세 기록(CDR) 데이터만 학습으로 사용하여 추가적인 사기(Fraud) 검출의 한계가 있다. 예를 들어 앞에서 설명한 'Subscription fraud'는 통화 상세 기록(CDR)만으로는 검출할 수가 없으며, 고객 정보와 과금 데이터가 같이 있어야 검출을 할 수 있다. 이처럼 다양한 사기(Fraud) 시나리오 검출을 위해서는 통화 상세 기록(CDR), 고객 정보, 과금 정보뿐만 아니라 네트워크 프로토콜 정보 등 다양한 데이터를 조합하여 분석해야 한다. 사기(Fraud) 통화의 검출을 하기 위해서는 다양한 관련 분야의 데이터들을 같이 분석을 해야 다양한 사기(Fraud) 시나리오를 검출할 수 있을 것으로 판단된다.

두 번째, 최소한의 Feature만 사용하여 학습했기 때문에 더 많은 사기(Fraud) 시나리오 검출의 한계가 있다. 국제 표준 협회에서 통화 상세 기록(CDR)에서 필요한 변수에 대한 부분을 정의해 놓았으나, 장비 제조사에 따라 표준뿐만 아니라 변수를 추가 정의하여 사용한다. 여기서 중요한 부분은 변수 선택 기법(Feature Selection Method)이며, 변수 선정에 따라 다양한 사기(Fraud) 시나리오를 검출할 수 있을 것으로 판단된다.

마지막으로 중첩 사기(Fraud) 검출 방법론 연구가 필요하다. 중첩 사기(Fraud)는 정상적인 일반 사용자의 사용 패턴과 동일하게 사용. 즉, 사기(Fraud) 통화와 일반 통화를 구분할 수 없게 하여 정상적인 일반 사용자 및 CSP에 지속적인 피해를 주고 있다. 중첩 사기를 검출하기 위해 프로파일 기법 등 많은 연구가 되었으나, 통화 데이터가 많은 CSP에서 사용하기

에는 많은 어려움이 있다. 이처럼 사기(Fraud) 통화 시도는 기술의 발달에 따라 다양해지고 있어 많은 연구가 필요하다.

참고문헌

Communications Fraud Control Association (2019) - FRaud Loss Survey 2019

Gartner Market Trend. (2018). Maximizing the Value of Analytics, Artificial Intelligence and Automation in CSP Fraud Management, Published: 21 Dec 2018, ID: G00376070.

Richard A. Becker, Chris Vollnsky, and Allan R. Wilks (2010). Fraud Detection in Telecommunications: History and Lessons Learned. VOL. 52, No.1

CS Hilas, JN Sahalos (2005). User Profiling for Fraud Detection in Telecommunications Networks

H. Hakan Kilinc (2020). A Case Study on Fraudulent User Behaviors in the Telecommunication Network. 10.5152/ Electrica 2021.20050

하지현, 이종석 (2013), “무작위 추출 기반의 확률적 이상치 탐지,” 대한 산업공학회 추계학술대회 논문집, 961-965.

이정원, 엄정훈, 박태흥, 김승호(2015), “VoIP 네트워크 내의 Fraud와 SIM Box Fraud 검출 방법에 대한 연구” 한국통신학회논문집, 30(10), 1994-2005.

이정원, 엄정훈, 박태흥, 김승호(2015), “VoIP 상의 과금 회피 방법과 그 검출 방법에 대한 연구” 한국통신학회 학술대회논문집 728-729.

Y. Wu, S. Bagchi, S. Garg, N. Singh, (2004) "SCIDIVE: A stateful and cross protocol intrusion detection architecture for Voice-over-IP environments", International Conference on Dependable Systems and Networks, Florence, Italy,

D. Olszewski,(2012) "A probabilistic approach to fraud detection in telecommunications", Knowledge-Based Systems, vol. 26,

M. H. Cahill, D. Lambert, J. C. Pinheiro, D.X. Sun, (2002) "Detecting fraud in the real world", Handbook of Massive Data Sets, Springer, Boston, USA, vol 4,

D. Hoffstadt, S. Monhof, E. Rathgeb, (2012) "SIP trace recorder: monitor and analysis tool for threats in SIP-based networks", 8th International Wireless Communications and Mobile Computing Confer- ence (IWCMC), Limassol, Cyprus,

D. Hoffstadt, E. Rathgeb, M. Liebig, R. Meister, Y. Rebahi, T.Q. Thanh, (2014) "A comprehensive framework for detecting and preventing VoIP fraud and misuse.", The IEEE International Conference on Computing, Networking and Communications (ICNC), Honolulu, USA,

J. Guo, G. Liu, Y. Zuo, J. Wu, (2018) "Learning sequential behavior represen- tations for fraud detection", IEEE International Conference on Data Mining (ICDM), Singapore,

H. Lin, G. Liu, J. Wu, Y. Zuo, X. Wan, H. Li, (2020) "Fraud detection in dynamic interaction network", in IEEE Transactions on Knowledge and Data En- gineering, vol. 32, no. 10,

P. A. Estévez, C. M. Held, C. A. Perez, (2006) "Subscription fraud prevention in telecommunications using fuzzy rules and neural networks", Expert Systems with Applications, vol.31, no. 2,

C. S. Hilas, P. A. Mastorocostas, (2008) "An application of supervised and unsupervised learning approaches to telecommunications fraud detection", Knowledge-Based Systems, vol. 21, no. 7,

박영준, 김성범 (2015) 딥러닝을 이용한 네트워크상에서 주요 변수 추출. 한국경영과학회 학술대회 문집, 30011-4026

박시저, 박정술, 백준걸 (2010), 정규분포를 따르지 않는 비선형 데이터를 위한 모델 기반 이상탐지, 대한산업공학회 춘계공동학술대회 논문집, 819-825

Guannan Liu, Kia Guo, Yuan Zuo, Junjie Wu, Ren-yong Guo, (2020) Fraud detection via behavioral sequence embedding

Anwesha Mishra, (2021) "Fraud Detection: A Stust of AdaBoost Classifier and K0Means Clustering“. SRM University

김필성, 조성준 (2008), 국지적 선형 재구축을 이용한 이상치 탐지, 대한산업공학회 춘계공동학술대회 논문집, 2008.05, 104-111

부 록

<부록 1> 분류모델 학습 코드(Python)

```
import pandas as pd
import numpy as np
import pickle
import datetime as dt
import matplotlib.pyplot as plt

file_path =  '/Users/honghakjin/Language/FMS/DATA'
df = pd.read_csv(file_path+r'/'+'df_dataV.csv', index_col=False, sep=',')

df.shape
df.info()
df.isnull().sum()
df['Fraud'].value_counts()

# 'Call Duration' feature make category
bins = [-1,180, 600, 3601]
labels = [180, 600, 3601]
df['Call Duration'] = pd.cut(df['Call Duration'], bins, labels = labels)
df['Call Duration'].value_counts()

# one hot encoding
df_onhot = pd.get_dummies(df, columns=['OC_CODE', 'IC_CODE', 'OType_Code',
                                       'IType_Code', 'Call_Duration', 'Service_Result'],
                          drop_first=True)
df_onhot.info() # feature 72, memory: 5.8 MB
df_onhot.head()

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler

# train_test_split
X = df_onhot.loc[:, [col for col in df_onhot.columns if col != 'Fraud']]
y = df_onhot['Fraud']

# 함수 `train_val_test_split`을 정의:
# #데이터를 학습 셋 (training set), 검증 셋 (validation set), 테스트 셋 (test set) 3개로 분할
def train_val_test_split(X, y, val_size=0.3, test_size=0.2, random_state=123):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                        test_size=test_size,
                                                        random_state=random_state)
    val_size_rev = val_size / (1 - test_size)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
                                                      test_size=val_size_rev,
                                                      random_state=random_state)
    return X_train, X_val, X_test, y_train, y_val, y_test

X_train, X_val, X_test, y_train, y_val, y_test = train_val_test_split(X, y,
                                                                      val_size=0.3,
                                                                      test_size=0.2,
                                                                      random_state=123)
```

```
##########################################################################
# Decision tree
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from pprint import pprint

depth_set = [3, 4, 5, 6, 7, 8, 9, 10]

dt_models = []
accuracy_set = []
cm_set = []
train_accuracy_set = []

for depth in depth_set:
    model = DecisionTreeClassifier(max_depth = depth, random_state = 1)
    model.fit(X_train, y_train)
    y_train_hat = model.predict(X_train)
    y_val_hat = model.predict(X_val)
    train_accuracy = metrics.accuracy_score(y_train,
                                            y_train_hat)
    accuracy = metrics.accuracy_score(y_val, y_val_hat)
    cm = metrics.confusion_matrix(y_val, y_val_hat)
    print(metrics.classification_report(y_val, y_val_hat, digits=4))

    dt_models.append(model)
    accuracy_set.append(accuracy)
    train_accuracy_set.append(train_accuracy)
    cm_set.append(cm)

pprint(train_accuracy_set)
pprint(accuracy_set)

# 파라미터 탐색 결과, 가장 좋은 모델과 Validation set에 대한 정확도
max_value = max(accuracy_set)
max_index = accuracy_set.index(max_value)
print(max_index)
print(max_value)

print(cm_set[max_index])
# 가장 좋은 모델
best_dt = dt_models[max_index]
print(best_dt)
```

```
################################################################################
# Random Forest

from sklearn.ensemble import RandomForestClassifier

n_estimators_set = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
max_features_set = ['auto', 'log2']

rf_models = []
accuracy_set = []
cm_set = []
train_accuracy_set = []

for n_estimators in n_estimators_set:
    for max_features in max_features_set:
        rf = RandomForestClassifier(n_estimators = n_estimators,
                                    max_features = max_features,
                                    random_state = 123)
        rf.fit(X_train, y_train)
        y_train_hat = rf.predict(X_train)
        y_val_hat = rf.predict(X_val)
        train_accuracy = metrics.accuracy_score(y_train, y_train_hat)
        accuracy = metrics.accuracy_score(y_val, y_val_hat)
        cm = metrics.confusion_matrix(y_val, y_val_hat)
        print(metrics.classification_report(y_val, y_val_hat, digits=4))

        rf_models.append(rf)
        train_accuracy_set.append(train_accuracy)
        accuracy_set.append(accuracy)
        cm_set.append(cm)

pprint(train_accuracy_set)
pprint(accuracy_set)

# 파라미터 탐색 결과, 가장 좋은 모델과 Validation set에 대한 정확도
max_value = max(accuracy_set)
max_index = accuracy_set.index(max_value)
print(max_index)
print(max_value)

print(cm_set[max_index])
# 가장 좋은 모델
best_rf = rf_models[max_index]
print(best_rf)
```

```
################################################################################
# LogisticRegression

from sklearn.linear_model import LogisticRegression

# Hyper-parameter candidates
penalty_set = ['l1', 'l2']
C_set = [0.01, 0.1, 1, 10, 100]
class_weight_set = [None, 'balanced']

# 결과 저장을 미리 할당하기 위한 리스트 선언
train_acc_set = []
val_acc_set = []
cm_set = []
lr_models = []

import warnings
warnings.filterwarnings("ignore")
for penalty in penalty_set:
    for C in C_set:
        for class_weight in class_weight_set:
            lr = LogisticRegression(penalty=penalty, C=C, solver = 'liblinear',
                                    class_weight=class_weight,
                                    random_state=2072)
            # Train the model
            lr.fit(X_train, y_train)
            lr_models.append(lr)

            # Calculate training accuracy and validation accuracy
            y_train_hat = lr.predict(X_train)
            y_val_hat = lr.predict(X_val)
            train_acc = metrics.accuracy_score(y_train, y_train_hat)
            val_acc = metrics.accuracy_score(y_val, y_val_hat)
            cm = metrics.confusion_matrix(y_val, y_val_hat)
            print(metrics.classification_report(y_val, y_val_hat, digits=4))

            train_acc_set.append(train_acc)
            val_acc_set.append(val_acc)
            cm_set.append(cm)


# 파라미터 탐색 결과, 가장 좋은 모델과 Validation set에 대한 정확도
max_value = max(val_acc_set)
max_index = val_acc_set.index(max_value)
print(max_index)
print(max_value)

print(cm_set[max_index])
# 가장 좋은 모델
best_lr = lr_models[max_index]
print(best_lr)
```

```
################################################################################
# K-Nearest Neighbors Classifier

from sklearn.neighbors import KNeighborsClassifier

# Hyper-parameter caldidates
n_neighbors_set = [10, 15, 20, 25, 30, 35, 45, 50, 55, 60]
weights_set = ['uniform', 'distance']

# 결과 저장을 미리 할당하기 위한 리스트 선언
train_acc_set = []
val_acc_set = []
cm_set = []
knn_models = []

for n_neighbors in n_neighbors_set:
    for weights in weights_set:
        knn = KNeighborsClassifier(n_neighbors=n_neighbors, weights=weights, n_

        # Train the model
        knn.fit(X_train, y_train)
        knn_models.append(knn)

        # Calculate training accuracy and validation accuracy
        y_train_hat = knn.predict(X_train)
        y_val_hat = knn.predict(X_val)
        train_acc = metrics.accuracy_score(y_train, y_train_hat)
        val_acc = metrics.accuracy_score(y_val, y_val_hat)

        cm = metrics.confusion_matrix(y_val, y_val_hat)
        print(metrics.classification_report(y_val, y_val_hat, digits=4))

        train_acc_set.append(train_acc)
        val_acc_set.append(val_acc)
        cm_set.append(cm)

# 파라미터 탐색 결과, 가장 좋은 모델과 Validation set에 대한 정확도
max_value = max(val_acc_set)
max_index = val_acc_set.index(max_value)
print(max_index)
print(max_value)

print(cm_set[max_index])
# 가장 좋은 모델
best_knn = knn_models[max_index]
print(best_knn)
```

```
#################################################################################
# Linear Support Vector Classifier
from sklearn.svm import LinearSVC, SVC

# Hyper-parameter caldidates
penalty_set = ['l1', 'l2']
C_set = [0.01, 0.1, 1, 10, 100]
class_weight_set = [None, 'balanced']

# 결과 저장을 미리 할당하기 위한 리스트 선언
train_acc_set = []
val_acc_set = []
cm_set = []
lr_svc_models = []

for penalty in penalty_set:
    for C in C_set:
        for class_weight in class_weight_set:
            if(penalty=='l1') :
                lr_svc = LinearSVC(penalty=penalty, C=C,
                                   class_weight=class_weight, dual=False,
                                   random_state=1234)
            else :
                lr_svc = LinearSVC(penalty=penalty, C=C,
                                   class_weight=class_weight, dual=True,
                                   random_state=1234)

            lr_svc.fit(X_train, y_train)
            lr_svc_models.append(lr_svc)

            # Calculate training accuracy and validation accuracy
            y_train_hat = lr_svc.predict(X_train)
            y_val_hat = lr_svc.predict(X_val)
            train_acc = metrics.accuracy_score(y_train, y_train_hat)
            val_acc = metrics.accuracy_score(y_val, y_val_hat)
            cm = metrics.confusion_matrix(y_val, y_val_hat)
            print(metrics.classification_report(y_val, y_val_hat, digits=4))

            train_acc_set.append(train_acc)
            val_acc_set.append(val_acc)
            cm_set.append(cm)

# 파라미터 탐색 결과, 가장 좋은 모델과 Validation set에 대한 정확도
max_value = max(val_acc_set)
max_index = val_acc_set.index(max_value)
print(max_index)
print(max_value)

print(cm_set[max_index])
# 가장 좋은 모델
best_lr_svc = lr_svc_models[max_index]
print(best_lr_svc)
```

<부록 2> 분류모델 학습 검증 코드(Python)

```
#############################################################################################
#  테스트 셋을 이용하여 가장 성능이 좋은 모델 찾기
best_models = [best_dt, best_rf, best_lr, best_knn, best_lr_svc]#, best_svc]
pprint(best_models)


for best_model in best_models:
    y_test_hat = best_model.predict(X_test)
    print(best_model)
    print(metrics.accuracy_score(y_test, y_test_hat))
    print(metrics.confusion_matrix(y_test, y_test_hat))
    print('='*80)


#############################################################################################
# 학습된 best model들을 training set + validation set에 다시 학습한 후, test set에 대해 예측 성능 평가
X_concat = pd.concat([X_train, X_val])
y_concat = pd.concat([y_train, y_val])

for best_model in best_models:

    # 합친 데이터에 모델을 refit
    best_model.fit(X_concat, y_concat)

    # Test set에 대해 예측 성능 평가
    y_test_hat = best_model.predict(X_test)

    #print(best_model)
    print(metrics.accuracy_score(y_test, y_test_hat))
    print(metrics.confusion_matrix(y_test, y_test_hat))
    # print('='*80)
```

Abstract

A Study on how to reduce the fraud of CSP Fraud Management Systems through machine learning

- CSP Fraud Detection -

Hong, HAK JIN
Seoul School of Integrated Sciences and Technologies
Advisor: Jang, Jung Hoo, Ph.D.

Fraud call detection by Communications Service Provider (CSP) is an important but difficult task. Due to the development of technology, Fraud calls are being made through various methods. In addition, Fraud calls are attempted in the same way (pattern) as normal users, making it more difficult to distinguish Fraud calls from normal users. There is a limit to detecting such complex and difficult-to-detect fraud with a rule-based system using a threshold. In addition, according to the CFCA 2019 report, CSP losses have been increasing again since 2019 due to Fraud calls. As mentioned earlier, currency fraud (Fraud) has also diversified

due to the diversification of services with the development of technology, limiting wholesaler managers to immediately respond to new types of Fraud calls based on rules using existing thresholds.

In this study, the purpose of the CSP Fraud detection system is to immediately respond to various Fraud calls and predict Fraud calls by replacing the process of creating rules using thresholds with machine learning. The data used for the analysis here used the CSP's call detail record (CDR) data for 3 months. Among them, call detail record (CDR) data consisting of outbound international phone calls were predicted by learning a classification model of supervised learning. The label was based on Fraud calls detected by the CSP Fraud detection system and Fraud numbers published by the CFCA, and the five classification models used were the Division Tree Classifier, Random Forest Classifier, Logic Regression Classifier, K-Nearest Neighbors, Class. In addition, the optimal hyper-parameter for each module was found and applied to each prediction model to make Fraud predictions. As a result of the analysis, more than 90% of the K-Nearest Neighbors and Random Forest models predicted fraudulent currencies, of which the K-Nearest Neighbors model predicted high at 97%. Analyzing not only call detail records (CDRs) but also data such as customer data and billing data that were not included in this study is expected to help detect CSP fraud as well as immediately respond to Fraud calls.

Keywords: Fraud detection, fraud calls, classification machine

learning, carrier fraud, CSP fraud.

Student Number: 2015411013