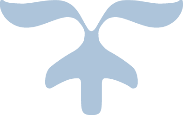


신용카드 사기 검출 보고서

신용카드 사기 검출 예측



2025 APRIL 15

우 성 욱

<목 차>

내용

[요약 2](#_Toc195637158)

[1. 프로젝트 개요 3](#_Toc195637159)

[1.1. 목표 3](#_Toc195637160)

[1.2. 데이터 3](#_Toc195637161)

[2. 데이터 구성 4](#_Toc195637162)

[2.1. 데이터 구성 4](#_Toc195637163)

[2.2. 이상치, 결측 치 및 중복 데이터 유무 4](#_Toc195637164)

[3. 프로젝트 흐름 5](#_Toc195637165)

[4. 모델링 6](#_Toc195637166)

[4.1. 사용 모델 6](#_Toc195637167)

[4.2. 모델 사용한 이유 6](#_Toc195637168)

[5. 정밀도(Precision), 재현율(Recall), F1-score, AUC (ROC-AUC) 가 중요한 이유 8](#_Toc195637169)

[6. 실험 결과 9](#_Toc195637170)

[6.1 원본 데이터 가공 없이 모델 학습 및 일반화 성능 확인 9](#_Toc195637171)

[6.2 원본 데이터를 StandarScaler로 가공 후 모델 학습 및 일반화 성능 확인 9](#_Toc195637172)

[6.3 원본 데이터를 로그화로 가공 후 모델 학습 및 일반화 성능 확인 9](#_Toc195637173)

[6.4 전처리 방법에 따른 모델 성능 비교 및 추천 모델 9](#_Toc195637174)

[6.5 결론 10](#_Toc195637175)

# 요약

본 프로젝트는 신용카드 거래 데이터셋을 기반으로 다양한 머신러닝 모델을 적용하여 사기 거래를 효과적으로 탐지하는 데 초점을 맞추었습니다. 주요 분류 알고리즘으로는 로지스틱 회귀, SVC, 랜덤 포레스트, 그래디언트 부스팅, XGBoost 등을 사용하였으며, StandardScaler 정규화, 로그 변환 등의 전처리 기법을 비교 분석하여 모델 성능에 미치는 영향을 평가하였습니다. 각 모델은 정밀도(Precision), 재현율(Recall), F1-score, AUC(ROC-AUC) 지표를 중심으로 성능을 비교하였습니다.

# 프로젝트 개요

## 목표

신용카드 거래 데이터 기반으로 사기 거래 여부를 예측하는 분류 모델을 개발하여, 금융 기관의 리스크를 줄이고 고객 보호를 강화하는 것이 목표

## 데이터

데이터 셋: Kaggle 신용카드 사기 탐지 데이터 (284,807건)

특성: 익명 처리된 28개 변수(V1~V28), 거래 시간(Time), 거래 금액(Amount), 사기 여부(Class: 0=정상, 1=사기)

# 데이터 구성

## 데이터 구성

샘플 수: 284,807개

특성 수: 30개

## 이상치, 결측 치 및 중복 데이터 유무

결측치: 없음

이상치: 변수 간 스케일 차이가 존재

중복 데이터: 없음

# 프로젝트 흐름

원본 데이터 가공 없이 모델 학습 및 일반화 성능 확인

* 로지스틱 회귀
* 랜덤 포레스트
* 그래디언트 부스팅
* XGBClassifier
* LGBMClassifier
* SVM

원본 데이터를 StandarScaler로 가공 후 모델 학습 및 일반화 성능 확인

* 로지스틱 회귀
* 랜덤 포레스트
* 그래디언트 부스팅
* XGBClassifier
* LGBMClassifier
* SVM

원본 데이터를 로그화로 가공 후 모델 학습 및 일반화 성능 확인

* 로지스틱 회귀
* 랜덤 포레스트
* 그래디언트 부스팅
* XGBClassifier
* LGBMClassifier
* SVM

# 모델링

## 사용 모델

* 로지스틱 회귀
* 랜덤 포레스트
* 그래디언트 부스팅
* XGBClassifier
* LGBMClassifier
* SVM

## 모델 사용한 이유

* 로지스틱 회귀 (Logistic Regression)

선형 모델로 이진 분류 문제에 적합하며, 빠르고 해석이 쉬움.

* 랜덤 포레스트 (Random Forest)

여러 개의 결정 트리를 앙상블하여 예측 성능을 향상시키는 모델.

* 그래디언트부스팅:

순차적인 학습을 통해 예측 오차를 줄여나가는 강력한 앙상블 기법

일반적으로 높은 예측 성능을 보임

* XGBClassifier (XGBoost)

그래디언트 부스팅을 개선한 고성능 모델로, 빠르고 효율적임.

* LGBMClassifier (LightGBM)

대규모 데이터셋에 최적화된 부스팅 프레임워크

* SVM (Support Vector Machine)

클래스 간 경계를 최대화하는 분류 알고리즘

# 정밀도(Precision), 재현율(Recall), F1-score, AUC (ROC-AUC) 가 중요한 이유

* 재현율 (Recall)이 중요한 이유:

금전적 손실 최소화: 실제 사기 거래를 정상으로 잘못 분류하는 경우는 기업과 고객 모두에게 금전적 손실을 초래할 수 있으며, 때로는 회복이 어려울 수 있습니다.

고위험 거래 포착: 사기 거래는 소수지만 피해가 큰 사례가 많아, 가능한 한 많은 실제 사기 거래를 탐지하는 것이 중요합니다.

고객 신뢰 유지: 사기 거래를 놓치는 일이 반복되면, 신용카드사에 대한 고객의 신뢰가 무너질 수 있습니다. 유방암 데이터셋에서 F1-score가 중요한 이유:

* F1-score가 중요한 이유:

정밀도와 재현율의 균형: 재현율만 높이면 정상 거래를 사기로 잘못 분류하는 경우가 늘어나 정밀도가 낮아질 수 있으며, 이는 고객의 불편과 서비스 거부로 이어질 수 있습니다. F1-score는 이 두 지표 간 균형을 평가합니다.

고객 경험 관리: 위양성(정상 거래를 사기로 판단)이 많으면 고객에게 불필요한 경고, 거래 차단, 본인 확인 요청이 발생하여 불편을 초래할 수 있습니다.

업무 효율성 확보: F1-score는 지나치게 과잉 탐지하거나 지나치게 소극적인 탐지를 피하게 해주어, 운영 리소스 낭비를 방지합니다.

* AUC (Area Under the Curve)가 중요한 이유:

모델의 전반적인 분류 능력 평가: AUC는 다양한 임계값에서 모델이 정상 거래와 사기 거래를 얼마나 잘 구분하는지를 보여줍니다. 높은 AUC는 분류기의 전반적인 신뢰도를 의미합니다.

클래스 불균형에 강건함: 대부분의 거래가 정상인 상황(불균형 데이터셋)에서도 AUC는 모델 성능을 비교적 공정하게 평가할 수 있는 지표입니다.

모델 선택 및 비교에 유리: 다양한 알고리즘(XGBoost, SVM 등)을 실험할 때, AUC는 성능을 직관적이고 정량적으로 비교할 수 있게 해줍니다.

확률 예측 성능 확인: 사기 검출은 확률 기반으로 임계값 조절이 필요한 경우가 많기 때문에, AUC는 이러한 예측의 질을 평가하는 데 유용합니다.

# 실험 결과

## 6.1 원본 데이터 가공 없이 모델 학습 및 일반화 성능 확인

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 분류기 | 정확도 | 정밀도 | 재현율 | F1 | AUC | 걸린시간 |
| LogisticRegression | 0.9992 | 0.8738 | 0.6081 | 0.7171 | 0.9734 | 5.03초 |
| RandomForestClassifier | 0.9994 | 0.9550 | 0.7162 | 0.8185 | 0.9787 | 1104.1초 |
| GradientBoostingClassifier | 0.9993 | 0.8088 | 0.7432 | 0.7746 | 0.9536 | 870.01초 |
| XGBClassifier | 0.9993 | 0.8629 | 0.7230 | 0.7868 | 0.8986 | 5.9249초 |
| LGBMClassifier | 0.9973 | 0.3223 | 0.5270 | 0.4 | 0.7625 | 8.2224초 |
| SVC | 0.9993 | 0.8583 | 0.7365 | 0.7927 | 0.9036 | 4390.8초 |

## 6.2 원본 데이터를 StandarScaler로 가공 후 모델 학습 및 일반화 성능 확인

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 분류기 | 정확도 | 정밀도 | 재현율 | F1 | AUC | 걸린시간 |
| LogisticRegression | 0.9992 | 0.8636 | 0.6419 | 0.7364 | 0.9712 | 0.3393초 |
| RandomForestClassifier | 0.9994 | 0.9550 | 0.7162 | 0.8185 | 0.9787 | 1222.2초 |
| GradientBoostingClassifier | 0.9993 | 0.8088 | 0.7432 | 0.7746 | 0.9536 | 959.7초 |
| XGBClassifier | 0.9993 | 0.8527 | 0.7432 | 0.7942 | 0.9030 | 6.939초 |
| LGBMClassifier | 0.9983 | 0.5077 | 0.4459 | 0.4748 | 0.7258 | 9.176초 |
| SVC | 0.9993 | 0.8485 | 0.7568 | 0.8 | 0.9092 | 6058.4초 |

## 6.3 원본 데이터를 로그화로 가공 후 모델 학습 및 일반화 성능 확인

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 분류기 | 정확도 | 정밀도 | 재현율 | F1 | AUC | 걸린시간 |
| LogisticRegression | 0.9990 | 0.8370 | 0.5203 | 0.6417 | 0.9678 | 0.394초 |
| RandomForestClassifier | 0.9994 | 0.9550 | 0.7162 | 0.8185 | 0.9788 | 905.71초 |
| GradientBoostingClassifier | 0.9993 | 0.8148 | 0.7432 | 0.7774 | 0.9536 | 712.67초 |
| XGBClassifier | 0.9994 | 0.88 | 0.7432 | 0.8059 | 0.9281 | 4.254초 |
| LGBMClassifier | 0.9983 | 0.4969 | 0.5405 | 0.5178 | 0.7697 | 5.746초 |
| SVC | 0.9993 | 0.8583 | 0.6959 | 0.7687 | 0.9503 | 60.466초 |

## 6.4 전처리 방법에 따른 모델 성능 비교 및 추천 모델

* 정밀도(Precision)

정밀도는 전처리 방식에 크게 영향을 받지 않는 것으로 나타났습니다. 특히 트리 기반 모델인 Random Forest, Gradient Boosting, XGBoost는 전처리 유무와 관계없이 높은 정밀도를 유지하며, 안정적인 성능을 보였습니다.

* 재현율(Recall)

재현율은 전처리에 민감한 지표로, StandardScaler를 통한 정규화 처리 시 SVC 및 Logistic Regression과 같은 선형 모델에서 재현율이 소폭 향상되었습니다. 전반적으로 로그 변환보다는 정규화가 재현율 향상에 더 효과적인 것으로 확인되었습니다.

* F1-score

F1-score는 모델 전반에 걸쳐 큰 차이를 보이지 않았습니다. 트리 기반 모델(Random Forest, XGBoost)은 전처리의 영향을 거의 받지 않으면서도 지속적으로 상위권의 F1-score를 유지하였으며, 실전 적용에 있어 높은 신뢰도를 보였습니다.

* AUC (ROC-AUC)

AUC 또한 모든 전처리 방식에서 트리 계열 모델이 높은 값을 유지하였습니다. 특히 Random Forest는 데이터 전처리 방식에 상관없이 최고 수준의 AUC를 기록하였습니다.

## 6.5 결론

데이터 전처리를 하지 않아도 Random Forest, Gradient Boosting, XGBoost는 매우 우수한 성능을 보입니다. StandardScaler 정규화는 SVC, Logistic Regression과 같은 선형 모델의 성능을 효과적으로 개선하는 데 유리합니다. 로그 변환은 XGBoost 등 일부 모델에서 정밀도와 재현율의 균형을 개선하고 이상치를 완화하는 데 효과적이었습니다. 따라서, 본 프로젝트에서는 XGBoost 모델을 기반으로 하여 로그 변환된 데이터를 사용하는 것이 적합한 거 같습니다.