학습된 인공지능 모델 결과서

1 TEAM
팀원 : 김나현 문창교 이승규 정래원 이경현

1. 데이터와 문제 정의

본 프로젝트는 고객 이탈 여부(churn)를 예측하고, 운영 환경(Streamlit)에서 안정적으로 예측을 제공하는 것을 목표로 합니다.

- 타깃 변수: churn (이탈 여부, 이진 분류)
- 입력 특성:

- 수치형: credit_score, age, balance, estimated_salary, tenure

- 범주형: country, gender, credit_card, active_member, products_number

- 데이터 분할: stratify=y로 train/test 8:2 분할, random_state=42
- 불균형 처리:

- 시도: SMOTE → 성능 저하 관찰

- 채택: class_weight='balanced' (LogisticRegression, RandomForest)

- 특성 엔지니어링: 다양한 파생 변수를 추가한 데이터셋으로도 실험했으나, 교차검증 및 테스트 성능이 일관되게 저하되어 원본 특성만 사용하는 전략으로 회귀했습니다 (과적합 및 노이즈 증가 가능성).

전처리는 scikit-learn Pipeline/ColumnTransformer를 통해 학습 시점에만 fit되어 데이터 누수를 방지했습니다. 범주형은 OneHotEncoder(drop='first', handle_unknown='ignore')로 인코딩, 수치형은 모델에 따라 StandardScaler 또는 MinMaxScaler를 적용했습니다.

2. 모델링 전략 요약

(1) 베이스라인 모델 (무튜닝)

총 8개 모델을 테스트했습니다:

- LogisticRegression, KNN, SVM, DecisionTree
- RandomForest, Bagging, AdaBoost, MLP
- 모델 특성에 맞춘 스케일러 적용 (예: KNN/NN/SVM은 스케일 필요)
- train/test 모두 평가하여 과적합 여부 확인

(2) 최종 후보 모델

RandomForest, AdaBoost, LogisticRegression을 최종 후보로 선정했습니다. 이들은 train-test 성능 격차가 상대적으로 작고 안정적인 특성을 보였습니다.

(3) 하이퍼파라미터 최적화

- RandomizedSearchCV(cv=5, scoring='f1', n_iter=30, n_jobs=-1)
- 목적 지표: F1 (불균형 대응에 적합)
- SMOTE 재시도 → 성능 저하 → class_weight='balanced' 전략 유지
- 파생 변수 확장 실험 → 성능 저하 → 원본 입력 스키마 확정

3. 모델 검증 및 평가

(1) 평가 지표

- F1, Recall, Accuracy, ROC_AUC를 주요 평가 지표로 사용
- 베이스라인: 동일한 전처리-모델 파이프라인으로 train/test 평가 수행
- 튜닝 단계: 5-fold 교차검증으로 F1 최적화, test 세트로 일반화 성능 확인

(2) 검증 의견

• 일반화 성능: 파이프라인 구조로 데이터 누수를 방지했고, 최종 후보 3개 모델은 train/test 간 성능 차가 비교적 작아 일반화가 양호한 편으로 판단됩니다.

• 불균형 대응: F1을 최적화 지표로 사용한 점, class_weight='balanced' 활용이 적절합니다. SMOTE 및 파생 변수 확장은 본 데이터에서 오히려 성능 저하를 보여 배제한 결정이 합리적입니다.

• 개선 여지:

- PR AUC(precision-recall AUC) 추가 보고 권장 (양의 클래스 희소 시 유용)

- 최적 임계값 탐색(Youden's J, f1-max 기준 등)으로 F1/Recall 추가 개선 가능

- 튜닝 후 최종 평가는 반드시 홀드아웃(test) 기반으로 일관되게 보고

4. 모델 개선 및 최적화 평가

(1) 하이퍼파라미터 튜닝

- LogisticRegression: C, penalty(l1/l2), solver(liblinear/saga), max_iter
- RandomForest: n_estimators, max_depth, min_samples_split/leaf, max_features
- AdaBoost: n_estimators, learning_rate

(2) 적절성 평가

- 불균형 대응: class_weight='balanced'로 과도한 재표본화 부작용(SMOTE) 없이 안정화
- 계산-효율 균형: RandomizedSearchCV와 5-fold로 탐색 비용 대비 성능 개선을 도모
- 파생 변수: 복잡도 증가 대비 일반화 성능 하락으로 비채택

5. 모델 설명 및 해석

(1) 전역 해석

- LogisticRegression: 계수 부호/크기로 영향 방향/강도 해석 가능 (스케일 기준)
- RandomForest/AdaBoost: 특성 중요도(impurity 또는 permutation)로 주요 변수 파악

(2) 지역 해석

개별 예측에 대해 SHAP/Permutation으로 합리성 점검이 가능합니다.

(3) 주의점

- OneHotEncoder(drop='first')로 기준 범주가 제거됨. 계수 해석은 기준 대비 관계로 해석
- handle_unknown='ignore'로 새로운 카테고리 입력에도 안전 (해당 더미는 0으로 처리)

6. 실제 적용 가능성 및 확장성

(1) 배포/서빙

• 파이프라인(best_estimator_) 저장 방식은 전처리+모델을 통째로 포함하므로 Streamlit에서 바로 로드 후 예측 가능 (전처리 일관성 보장)

• Streamlit 입력 스키마는 원본 데이터셋 기준으로 고정하고, 파생 변수는 사용하지 않음 (운영 단순성, 예측 일관성 확보)

(2) 확장성

- 새로운 특성 추가 시 ColumnTransformer에만 열 추가 후 재학습 가능
- 지역/세그먼트 확장 시 도메인 특성 반영 가능

7. Streamlit 연동 메모

- 저장된 모델은 scikit-learn Pipeline 형태로 로드하며, 입력 컬럼명/타입/순서를 툴에서 검증
- 입력값 검증:

- 수치형: 음수/이상치 경고, 결측 기본값 처리

- 범주형: 허용 값 이외 입력 시 안내 및 기본값 설정

- 출력:

- 예측 확률과 분류(임계값) 동시 제공

- 간단한 설명(Top-N 중요 특성) 또는 임계값 슬라이더 제공 시 사용자 이해도 향상

8. 재현 및 운영 체크리스트

(1) 재현

- random_state 고정 (분할, 모델, CV)
- 최종 리더보드는 test 세트 기준으로 기록

(2) 코드 품질

- PR AUC 추가, 임계값 최적화 루틴 (검증 세트 기반) 추가
- SMOTE/파생 변수 셀은 실험 기록으로 유지하되, 최종 의사결정은 원본 스키마+balanced 기반 결과로 고정

(3) 파일/아티팩트

- 모델 파일: 전처리 포함 Pipeline(.pkl)
- 버전: 모델명_타임스탬프, 메타정보(특성 목록, 데이터 크기, CV 스코어) 포함 권장

9. 결론

파이프라인 기반 전처리와 교차검증(RandomizedSearchCV)로 데이터 누수 없이 안정적인 일반화 성능을 확보했습니다.

SMOTE 및 파생 변수 확장은 본 데이터셋에서 성능 저하를 보여, 최종적으로 원본 특성 + class_weight='balanced' 전략이 더 적합했습니다.

최종 후보(RandomForest, AdaBoost, LogisticRegression)는 F1/Recall 기준으로 균형 잡힌 성능을 보이며, 운영 환경에서는:

- 해석/경량: LogisticRegression
- 균형/강건: RandomForest 또는 AdaBoost

test 세트 F1 최상 모델을 1순위 배포, 나머지를 챌린저로 모니터링하며 임계값/확률 보정으로 추가 개선이 가능합니다.

첨부 파일

After Tuning_test 모델별 성능 비교 결과

	Model	ROC_AUC	Accuracy	F1	Recall
0	RandomForest	0.859689	0.846314	0.644231	0.687179
1	AdaBoost	0.846388	0.864486	0.592824	0.487179
2	LogisticRegression	0.847294	0.784008	0.584830	0.751282

[그림 1] Base_train 모델별 성능 비교

After Tuning_train 모델별 성능 비교 결과

	Model	ROC_AUC	Accuracy	F1	Recall
0	RandomForest	0.953425	0.886883	0.748919	0.833226
1	AdaBoost	0.857212	0.857662	0.577487	0.480436
2	LogisticRegression	0.848041	0.774675	0.576313	0.756895

[그림 2] Base_test 모델별 성능 비교

Base_test 모델별 성능 비교 결과

	Model	ROC_AUC	Accuracy	F1	Recall
0	AdaBoost	0.846413	0.863448	0.602118	0.510256
1	NN	0.853923	0.860852	0.588957	0.492308
2	RandomForest	0.850462	0.863448	0.581876	0.469231
3	Bagging	0.821947	0.857736	0.577160	0.479487
4	LogisticRegression	0.848725	0.859813	0.571429	0.461538
5	SVM	0.827879	0.863448	0.550427	0.412821
6	KNN	0.799461	0.840602	0.516535	0.420513
7	DecisionTree	0.693565	0.799585	0.510152	0.515385

[그림 3] After Tuning_train 모델별 성능 비교

Base_train 모델별 성능 비교 결과

	Model	ROC_AUC	Accuracy	F1	Recall
0	DecisionTree	1.000000	1.000000	1.000000	1.000000
1	RandomForest	1.000000	1.000000	1.000000	1.000000
2	Bagging	0.999456	0.986234	0.964970	0.936498
3	KNN	0.923098	0.873506	0.622773	0.515715
4	NN	0.880269	0.866883	0.608031	0.509942
5	SVM	0.868368	0.868312	0.567775	0.427197
6	AdaBoost	0.847989	0.850000	0.566929	0.484926
7	LogisticRegression	0.847785	0.850390	0.538831	0.431687

[그림 4] After Tuning_test 모델별 성능 비교