

포르투갈 은행 마케팅 데이터 분석 연구보고서

머신러닝 기반 예측 및 마케팅 전략 도출

포르투갈 은행 마케팅 데이터 분석 연구

연구 기간: 2025.07.25 - 2025.07.28

연구자: 김명환 (AI 4기 5팀)

완료일: 2025년 7월 28일

1. 서론

- 연구 배경
 - 2008~2010년 글로벌 금융위기 시기, 포르투갈 은행의 정기에금 마케팅 캠페인 데이터 분석
- 비즈니스 목표
 - 정기에금 가입 예측 및 마케팅 효율성 극대화 전략 도출
- 분석 목표
 - 결정 트리 및 앙상블 기법 기반 예측모델 개발, 실무 적용 인사이트 도출
- 배경
 - 2008년(5월~12월): 글로벌 금융 위기가 확산되며 경기 둔화의 조짐이 나타나기 시작했고, 정부는 금융 시스템 안정화에 주력했습니다.
 - 2009년: 금융 위기의 직격탄을 맞아 GDP가 크게 위축되고 실업률이 급등하는 등 심각한 경기 침체를 겪었습니다.
 - 2010년(1월~11월): 경기 침체에서 벗어나지 못하고 국가 재정 적자 및 부채 문제가 부각되어 유로존 재정 위기의 핵심 국가로 부상하기 시작했습니다.

2. 데이터 이해

2.1 변수 설명

변수명	설명	타입	비고
education	교육 수준	category	8개 등급(대학교, 고등학교, 전문, 문맹 등), 금융이해도/소득 추정
default	신용 불량 여부	category	yes(신용불량), no(정상), unknown(미상), 신용위험도 지표
housing	주택 대출 여부	category	yes(대출 있음), no(없음), unknown(미상), 신용/자산상태 반영
loan	개인 대출 여부	category	yes(대출 있음), no(없음), unknown(미상), 신용/부채상태 반영
contact	연락 방식	category	cellular(휴대폰), telephone(유선전화), 마케팅 채널 효과 분석
month	마지막 연락 월	category	jan~dec(월별), 계절성/시기별 효과 반영
day_of_week	마지막 연락 요일	category	mon~fri(월~금), 요일별 캠페인 효과 분석
duration	통화 지속 시간(초)	int	모델에서 제외 (예측 불가 정보, 결과분석용)
campaign	현재 캠페인 연락 횟수	int	1~56회, 3~4회 최적, 20회 이상은 이상치 가능(고객 피로도)
pdays	이전 캠페인 후 경과 일수	int	999: 이전 캠페인 없음(신규 고객), 0~998: 마지막 접촉 후 경과일수(재접촉)
previous	이전 캠페인 연락 횟수	int	과거 캠페인 연락 횟수(0~7회, 대부분 0), 재참여/리텐션 지표
poutcome	이전 캠페인 결과	category	success(성공), failure(실패), nonexistent(연락없음), 과거 마케팅 효과
emp.var.rate	고용 변동률	float	분기별, 경제상황 반영(%), 거시경제 변수
cons.price.idx	소비자 물가지수	float	월별, 구매력/물가 지표, 거시경제 변수
cons.conf.idx	소비자 신뢰지수	float	월별, 경제 심리/소비심리 지표, 거시경제 변수
euribor3m	3개월 유리보 금리	float	일별, 자금조달 비용/금리, 거시경제 변수
nr.employed	고용자 수	float	분기별, 경제 규모(천명 단위), 거시경제 변수
y	정기 예금 가입 여부	category	타겟 변수(yes/no), 이진 분류

2.2 데이터셋 기본 정보

- 출처: UC Irvine Machine Learning Repository - Bank Marketing Dataset
- 기간: 2008년 5월 ~ 2010년 11월 (시간순 정렬)
- 규모: 41,188개 행, 21개 컬럼 (20개 입력 변수 + 1개 타겟 변수)

- 목적: 포르투갈 은행의 정기 예금 마케팅 캠페인 성공 예측
- 특수 상황: 글로벌 금융위기 시기 데이터 (2008년 서브프라임 모기지 사태 ~ 유럽 재정위기)

2.3 고객 기본 정보 (Bank Client Data)

- age: 나이 (숫자형) - 고객의 연령대별 금융 성향 분석 가능
- job: 직업 (범주형) - 소득 수준 및 직업 안정성 추정 지표
 - 12개 카테고리: admin, blue-collar, entrepreneur, housemaid, management, retired, self-employed, services, student, technician, unemployed, unknown
- marital: 결혼 여부 (범주형) - 가족 구성 및 금융 책임 수준 추정
- education: 교육 수준 (범주형) - 소득 수준 및 금융 이해도 추정 지표
 - basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown

2.4 신용 상태 관련 (Credit Information)

- default: 신용 불량 여부 (범주형) - 가장 직접적인 신용 위험 지표
- housing: 주택 대출 여부 (범주형) - 상대적으로 안정적인 대출 형태
- loan: 개인 대출 여부 (범주형) - 상대적으로 위험도 높은 대출 형태

2.5 캠페인 관련 (Campaign Information)

- contact: 연락 방식 (범주형) - cellular vs telephone 효과 분석 가능
- month: 마지막 연락 월 (범주형) - 계절별/월별 마케팅 효과 분석
- day_of_week: 마지막 연락 요일 (범주형) - 요일별 연락 효과 분석
- duration: 통화 지속 시간 초 단위 (숫자형)
 - 제외: 실무 예측 모델에서 반드시 제외 (통화 후에만 알 수 있는 정보로 실제 예측에 사용 불가)
- campaign: 현재 캠페인 연락 횟수 (숫자형) - 고객 피로도 분석 지표
- pdays: 이전 캠페인 후 경과 일수 (숫자형) - 999는 이전 캠페인 없음을 의미
- previous: 이전 캠페인 연락 횟수 (숫자형)
- poutcome: 이전 캠페인 결과 (범주형) - failure, nonexistent, success

2.6 거시경제 지표 (Social and Economic Context) 주의: 개별 고객 경제력이 아닌 전체 경제환경 지표

- emp.var.rate: 고용 변동률 (분기별 지표) - 경제 안정성 지표
- cons.price.idx: 소비자 물가지수 (월별 지표) - 구매력 지표
- cons.conf.idx: 소비자 신뢰지수 (월별 지표) - 경제 심리 지표
- euribor3m: 3개월 유리보 금리 (일별 지표) - 자금 조달 비용
- nr.employed: 고용자 수 (분기별 지표) - 경제 규모 지표

2.7 타겟 변수

- y: 정기 예금 가입 여부 (이진 분류) - "yes" 또는 "no"

3. 데이터 특성 및 주의사항

3.1 결측치 처리 방안

- 범주형 데이터의 결측치는 "unknown" 라벨로 표시됨
- 이를 별도 카테고리로 처리할지, 다른 방법으로 대체할지 결정 필요

3.2 Duration 변수 처리 전략

- 모든 모델에서 duration 변수 제외: 실무적으로 예측 시점에 알 수 없는 정보이므로 무조건 제거
- 통화 시간은 결과 분석용으로만 활용 (모델 성능 평가에는 사용하지 않음)

3.3 시간적 특성 및 금융위기 맥락

- 데이터가 시간순으로 정렬되어 있어 시계열 분석 가능
- 2008-2010년 글로벌 금융위기 시기:
 - 경제불안정, 높은 실업률, 신용경색 상황
 - 고객들의 위험회피 성향 증가로 안전자산(정기예금) 선호도 변화
 - 은행의 자금조달 어려움과 예금 확보 필요성 증대
- 거시경제 지표의 중요성: 금융위기 상황에서 경제지표가 고객 행동에 미치는 영향 극대화

- 캠페인 전략의 시대적 배경: 불안정한 경제상황에서의 마케팅 전략 효과성 분석 가능

4. 데이터 탐색 및 변수 분석

4.1 타겟 분포 및 불균형

포르투갈 은행 마케팅 데이터는 정기예금 가입률이 약 11% 수준으로, 타겟 불균형이 심한 이진 분류 문제입니다.

1. 타겟 변수 (y) 분포

정기예금 가입 현황:

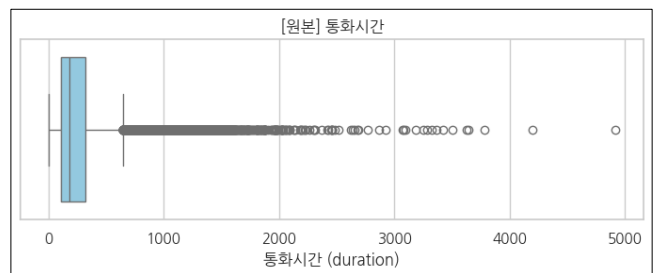
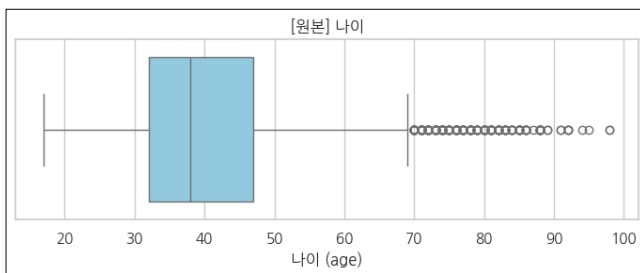
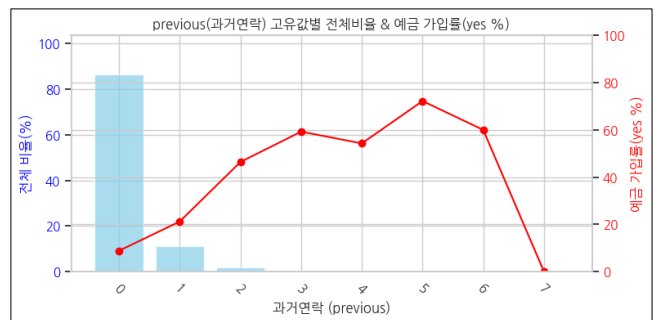
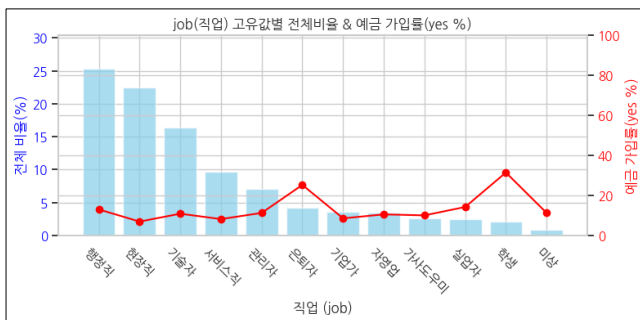
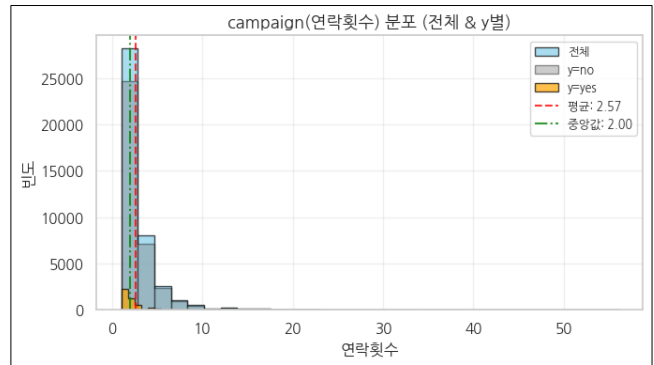
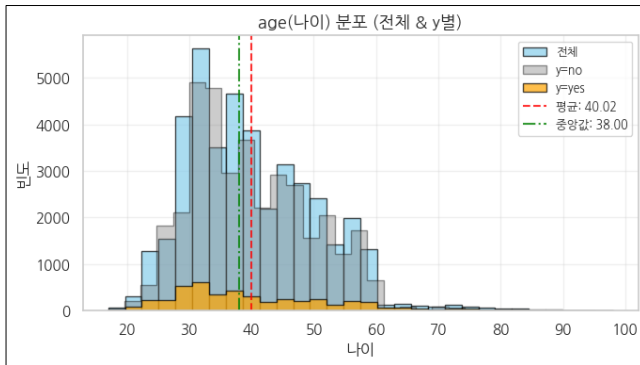
no: 36,548명 (88.7%)

yes: 4,640명 (11.3%)

4.2 주요 변수별 분포 및 특성

숫자형 변수(나이, 연락 횟수 등)와 범주형 변수(직업, 교육 수준 등)에 대한 히스토그램과 박스플롯 등을 활용하여 분포와 이상 값을 분석하였습니다.

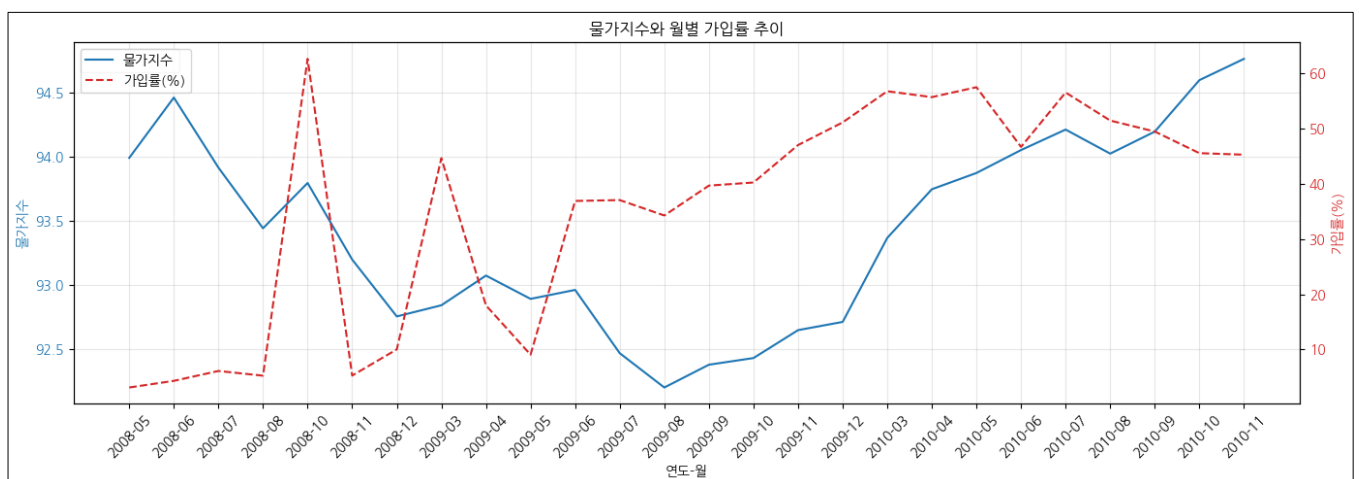
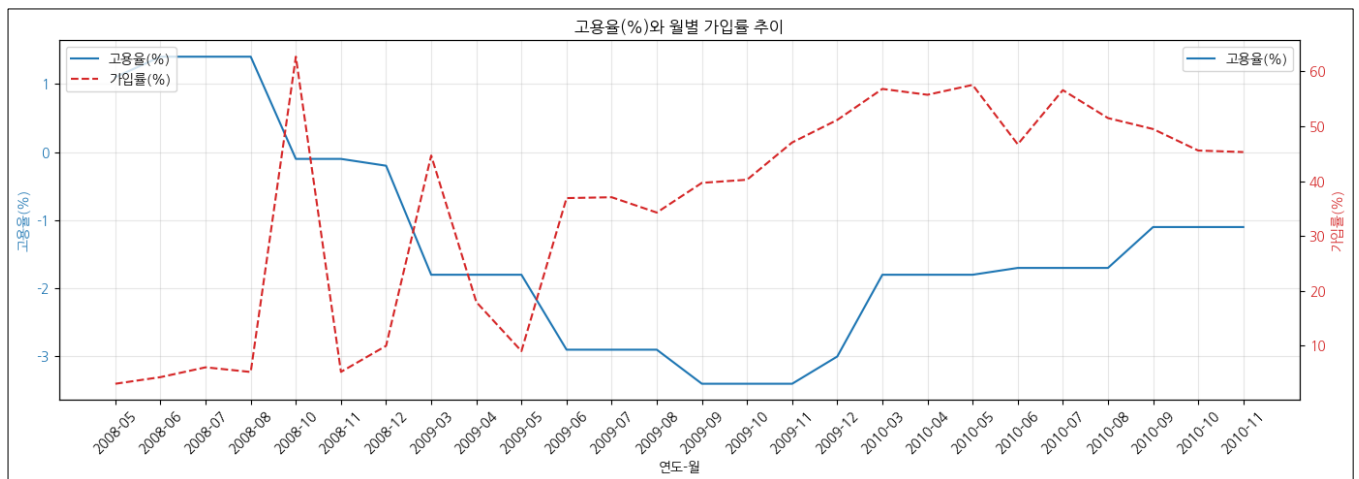
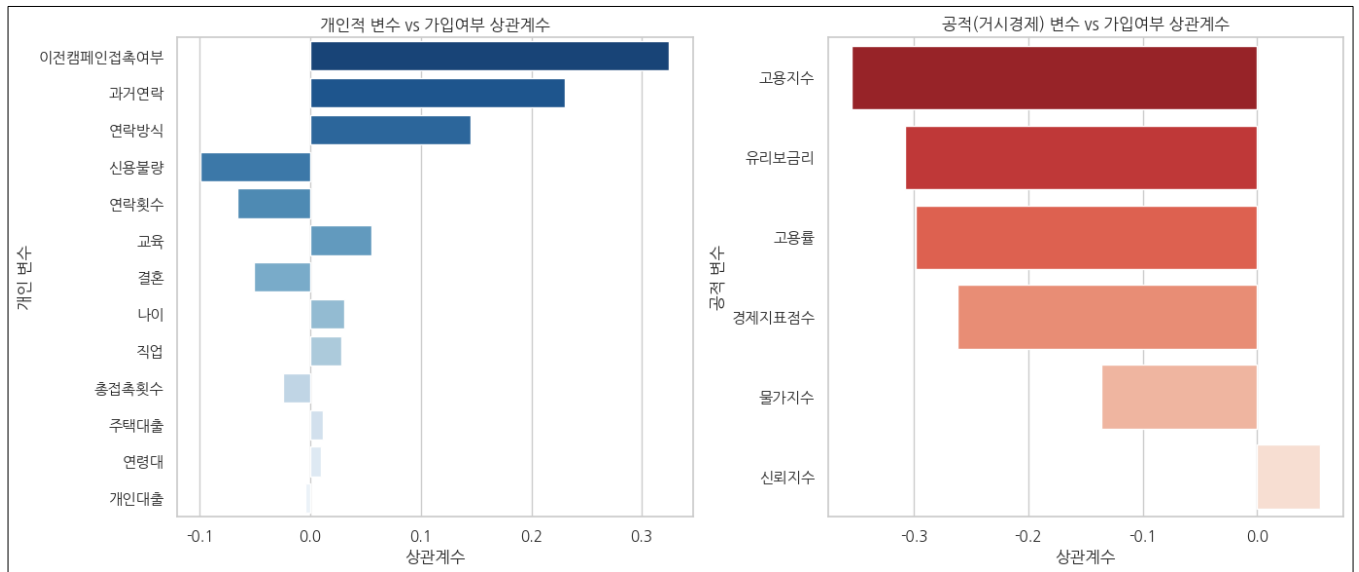
예를 들어, campaign 변수는 3~4회에서 가입률이 가장 높으며, 20회를 초과할 경우 오히려 가입률이 하락하는 피로도 효과가 관찰됩니다.

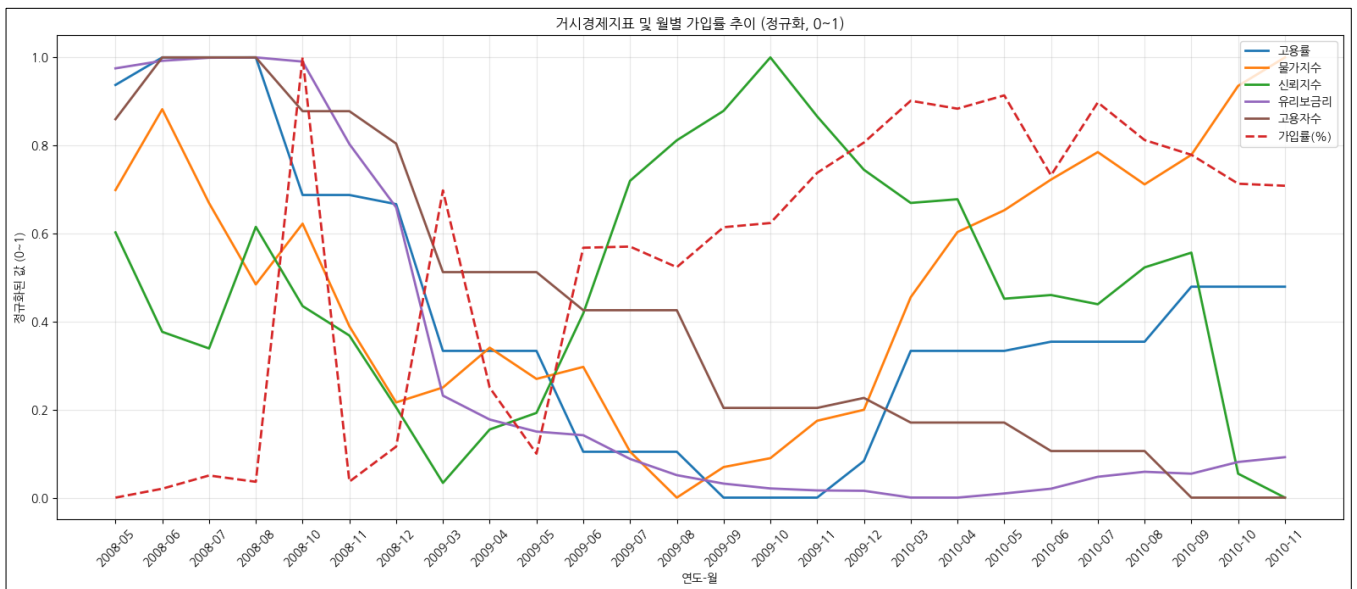
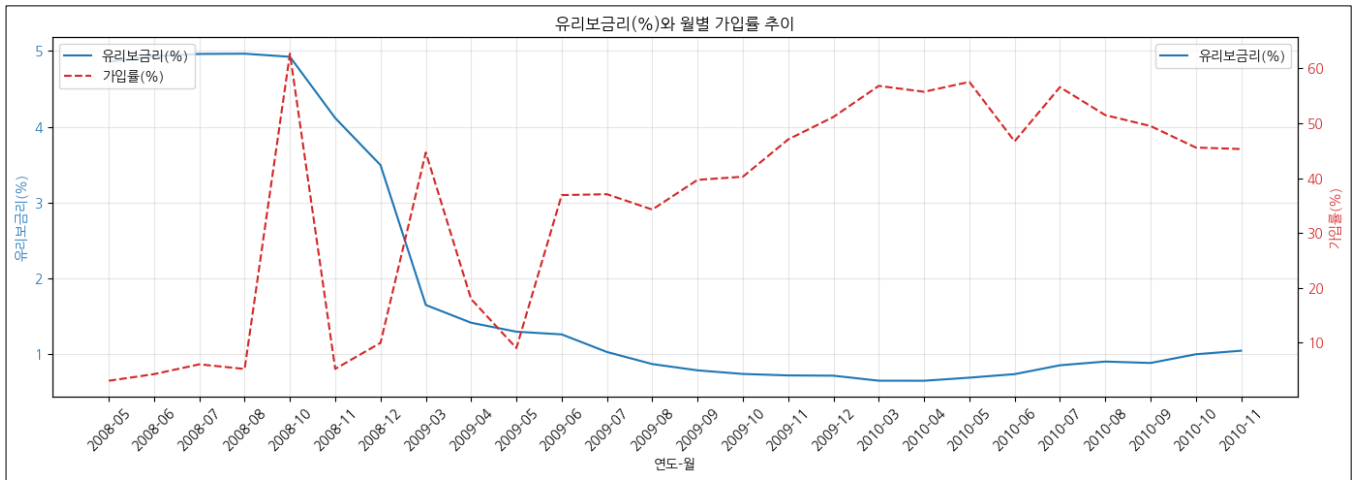
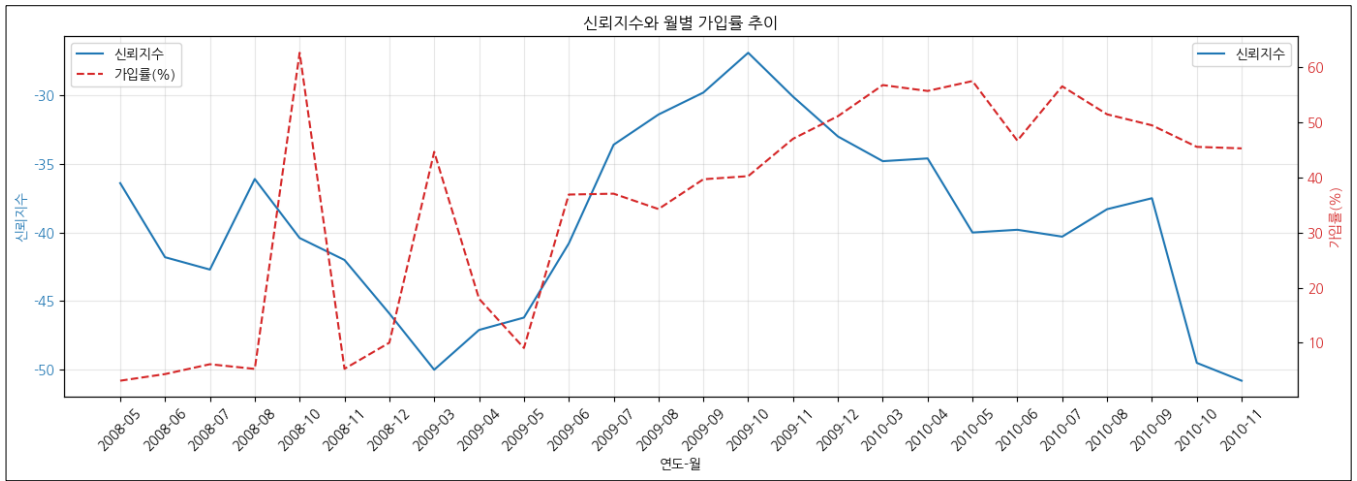


4.3 거시경제 지표 및 가입률 추이

고용률, 유리보금리, 소비자 신뢰지수, 물가지수 등 거시경제 변수와 예금 가입률 간의 시계열 추이를 분석한 결과, 경기 침체 또는 금리 인하 시기에 예금 가입률이 상승하는 현상을 확인할 수

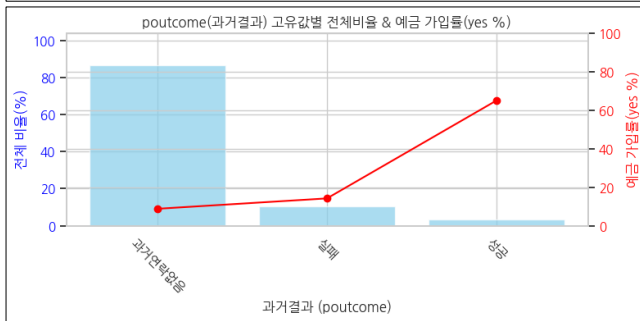
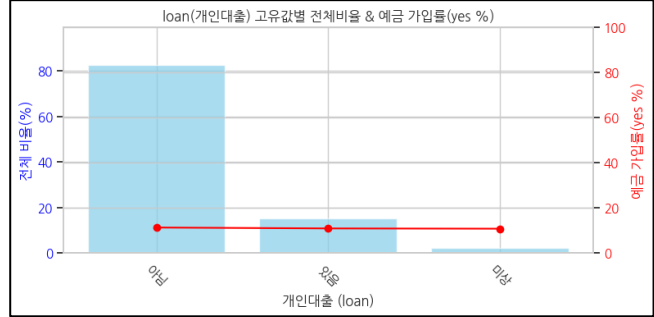
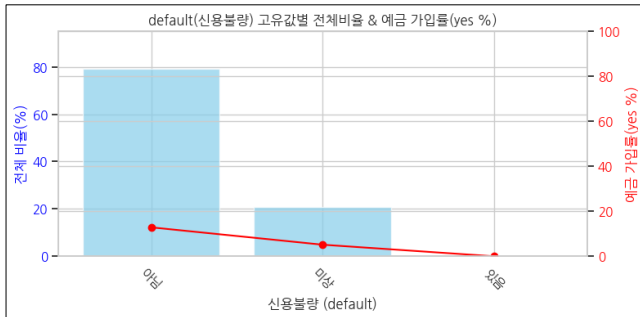
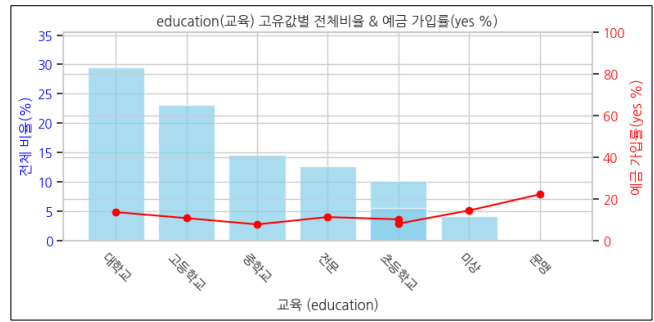
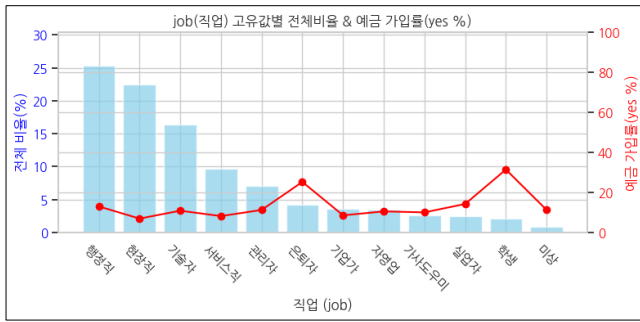
있었습니다. 이는 'Precautionary savings(예방적 저축)' 현상으로 해석됩니다. 그리고 일정 한계를 넘으면 'Liquidity Constraint(유동성 제약) 현상이 발생하는 것으로 해석됩니다.





4.4 변수별 가입률/상관관계 분석

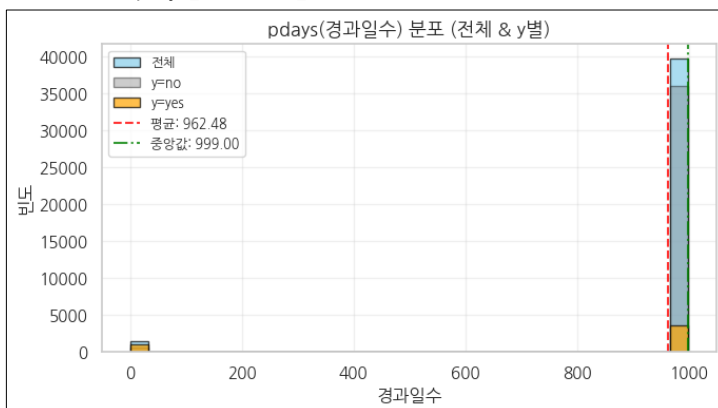
범주형 변수(직업, 교육, 개인대출 등)별로 가입률을 비교하고, 각 변수와 타겟(y) 간의 상관관계를 수치 및 그래프로 정리하였습니다. 예컨대 고학력, 고신용, 고소득군에서 예금 가입률 분석, 과거 캠페인 성공 경험(poutcome=success)이 있는 고객의 재가입률이 월등히 높았습니다.



5. 데이터 전처리

5.1 pdays 처리

- pdays는 접촉 여부로만 판단하는 파생 변수를 만들고 pdays는 사용 안함
- 분포를 살펴 봤을 때 접촉 여부 판단만 하더라도 될 것으로 판단함
 - pdays_contacted_before



5.2 인코딩(라벨/순서형/비즈니스 논리)된 변수

변수명	인코딩	의미 및 방식 요약
job	인코딩	직업별 소득/안정성 순서형 인코딩 (0~10)
marital	인코딩	결혼상태별 경제적 안정도 순서형 인코딩 (0~2)
education	인코딩	교육수준별 순서형 인코딩 (0~5)
default	인코딩	신용불량 여부: no(-1), unknown(0), yes(1)
housing	인코딩	주택대출 여부: no(-1), unknown(0), yes(1)
loan	인코딩	개인대출 여부: no(-1), unknown(0), yes(1)
contact	인코딩	연락방식: telephone(0), cellular(1)

변수명	인코딩	의미 및 방식 요약
month	인코딩	월별 순서형 인코딩 (0~11)
day_of_week	인코딩	요일별 순서형 인코딩 (0~6)
poutcome	인코딩	과거 캠페인 결과: failure(0), nonexistent(1), success(2)
y	타겟 인코딩	정기예금 가입여부: no(0), yes(1)
age_group	라벨 인코딩	연령대 그룹: young(0), adult(1), middle(2), senior(3), elderly(4)

5.3 스케일링된 변수

변수명	스케일링	의미 및 방식 요약
age_scaled	StandardScaler	나이 표준화
emp.var.rate_scaled	StandardScaler	고용률 표준화
cons.price.idx_scaled	StandardScaler	물가지수 표준화
cons.conf.idx_scaled	StandardScaler	신뢰지수 표준화
euribor3m_scaled	StandardScaler	유리보금리 표준화
nr.employed_scaled	StandardScaler	고용지수 표준화
campaign_scaled	MinMaxScaler	현재 캠페인 연락횟수 0~1 정규화
previous_scaled	MinMaxScaler	과거 캠페인 연락횟수 0~1 정규화
pdays_scaled	MinMaxScaler	pdays(이전 캠페인 접촉일수) 0~1 정규화 (999→0)

5.4 추가된 파생 변수 및 의미

변수명	의미 및 산출 방식
economic_score	개별 고객 경제력 점수 (직업, 교육, 나이, 결혼상태 점수화 합산)
economic_grade	economic_score 등급화 (상/중상/중/중하/하)
credit_score	신용점수 (default, housing, loan 점수화 합산)
credit_grade	credit_score 등급화 (A~E)
year_month	연락 시점의 연-월(YYYY-MM)
emp.var.rate_chg_pct/diff	고용률 전월 대비 변화율/차이
cons.price.idx_chg_pct/diff	물가지수 전월 대비 변화율/차이
cons.conf.idx_chg_pct/diff	신뢰지수 전월 대비 변화율/차이
euribor3m_chg_pct/diff	유리보금리 전월 대비 변화율/차이
nr.employed_chg_pct/diff	고용지수 전월 대비 변화율/차이
is_high_risk	고위험군 여부 (신용등급 E 또는 경제점수 하위 20%)
pdays_contacted_before	이전 캠페인 접촉 경험 여부 (pdays!=999:1, 999:0)
is_existing_customer	기존 고객 여부 (pdays_contacted_before와 동일)
age_group	나이 구간별 그룹화 (young, adult, middle, senior, elderly)
total_contacts	총 접촉 횟수 (campaign + previous)
economic_score (거시경제)	5개 경제지표 스케일링 후 평균값 (거시경제 환경 종합점수)

6. 모델링 및 하이퍼파라미터 튜닝

6.1 실험 설계

- 데이터 분할: train/test(8:2), stratify로 타겟 분포 유지
- 평가 지표: F1, ROC-AUC, Precision, Recall, Accuracy 등 종합 고려

6.2 모델별 함수 및 하이퍼파라미터

결정트리, 랜덤포레스트, 그래디언트 부스팅, XGBoost 등 다양한 분류 모델을 적용하고, 주요 하이퍼파라미터(트리 깊이, 노드 수, 학습률 등)를 그리드로 정의하여 자동 탐색.

부록 1: 하이퍼파라미터

6.3 하이퍼파라미터 자동 탐색

모든 모델별로 주요 하이퍼파라미터 조합을 반복 실행하며, 캐시 및 시스템 리소스를 효율적으로 활용.

=== 최적 파라미터 기반 Voting Classifier 구성 ===

각 모델별 최적 성능:

Decision Tree: F1=0.4747, 파라미터조합=#1

Gradient Boosting: F1=0.5172, 파라미터조합=#205

Random Forest: F1=0.5291, 파라미터조합=#68

XGBoost: F1=0.5086, 파라미터조합=#102

Decision Tree 최적 파라미터:

```
{'max_depth': 4, 'min_samples_split': 30, 'min_samples_leaf': 5, 'criterion': 'gini', 'max_features': 'None'}
```

Gradient Boosting 최적 파라미터:

```
{'n_estimators': 200, 'learning_rate': 0.01, 'max_depth': 8, 'min_samples_split': 50, 'min_samples_leaf': 10, 'subsample': 0.1}
```

Random Forest 최적 파라미터:

```
{'n_estimators': 150, 'max_depth': 15, 'min_samples_split': 20, 'min_samples_leaf': 3, 'max_features': 'sqrt', 'bootstrap': True}
```

XGBoost 최적 파라미터:

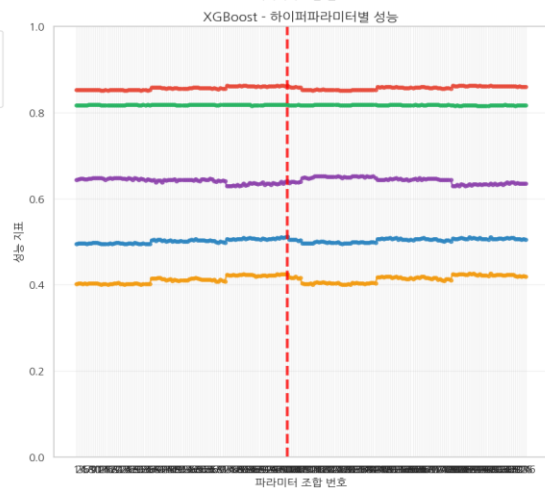
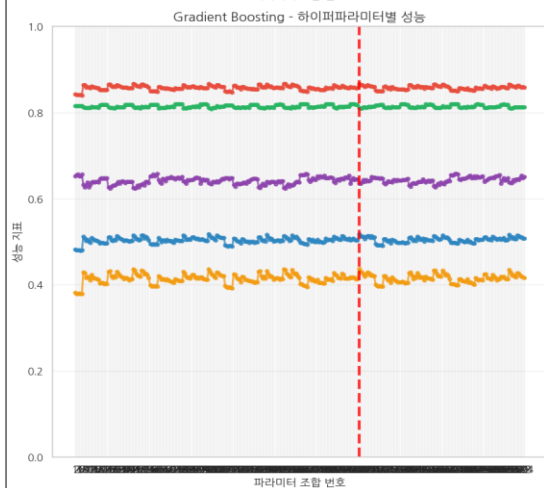
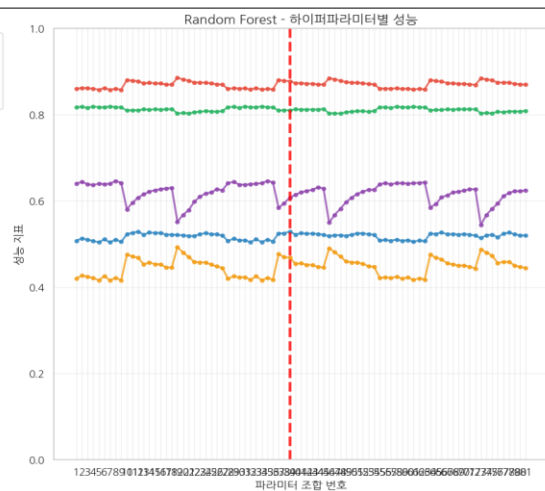
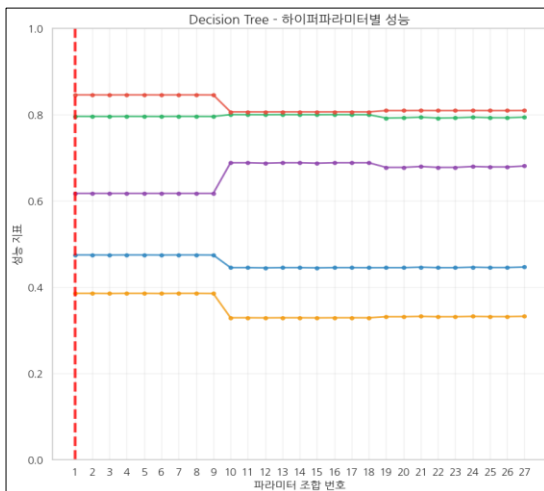
```
{'n_estimators': 200, 'learning_rate': 0.01, 'max_depth': 8, 'min_child_weight': 5, 'subsample': 0.8, 'colsample_bytree': 1.0, 'reg_alpha': 0.1, 'reg_lambda': 2.0}
```

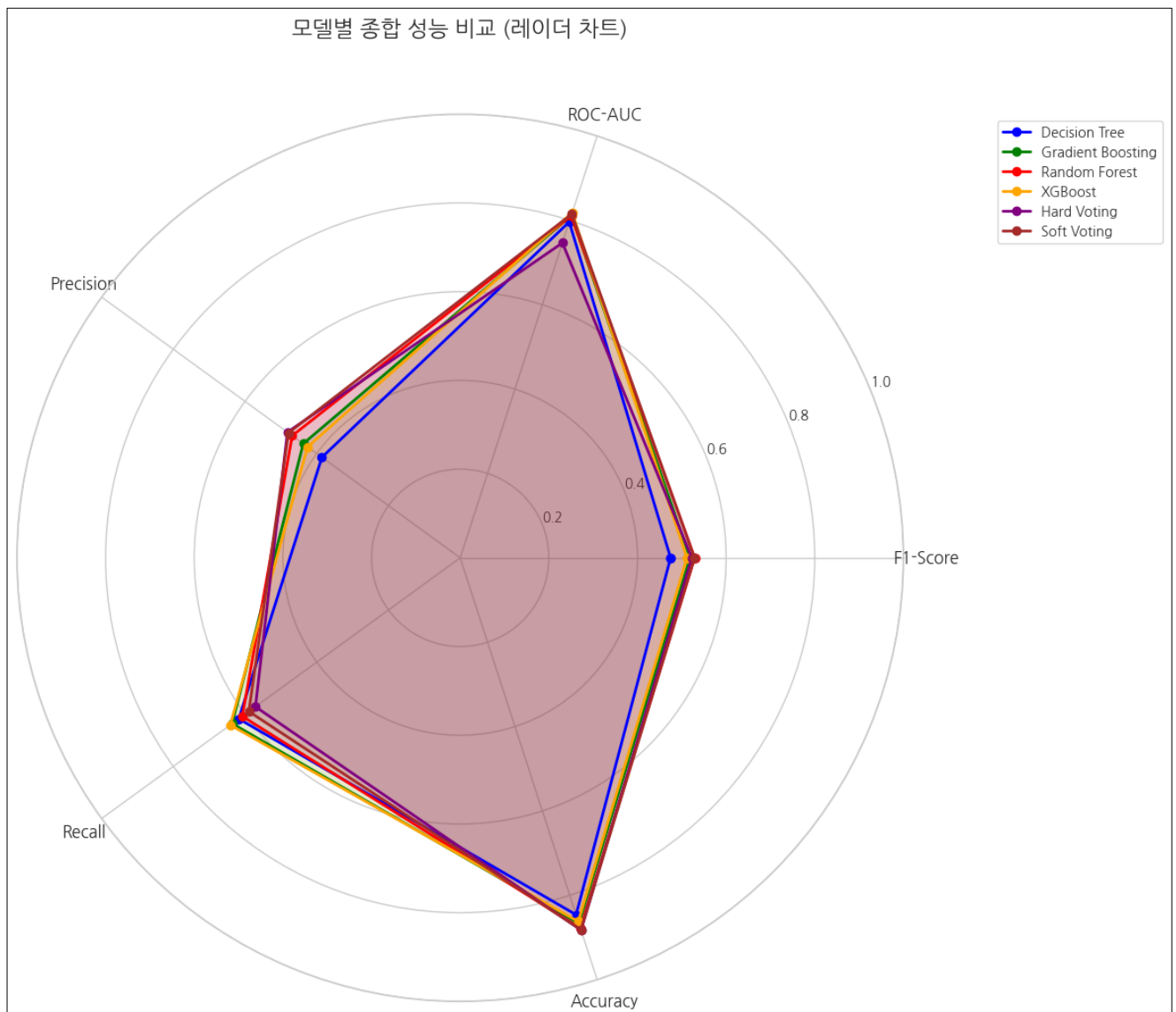
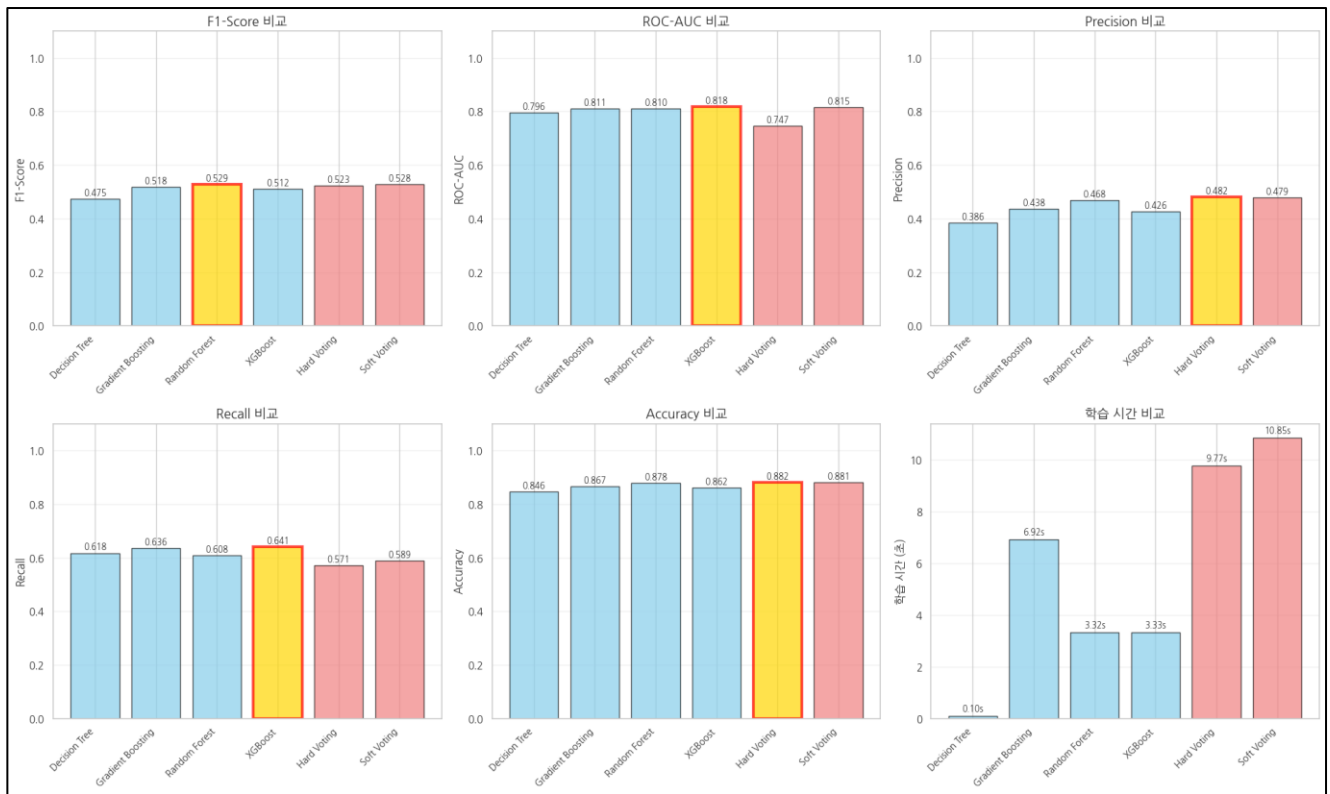
캐시된 Hard Voting 결과를 사용합니다.

캐시된 Soft Voting 결과를 사용합니다.

최종 성능 비교표:

	Model	Type	F1-Score	ROC-AUC	Precision	Recall	Accuracy	Train_Time
0	Decision Tree	개별모델	0.4747	0.7962	0.3856	0.6175	0.8461	0.15s
1	Gradient Boosting	개별모델	0.5172	0.8108	0.4368	0.6336	0.8667	6.70s
2	Random Forest	개별모델	0.5291	0.8123	0.4598	0.6228	0.8751	7.41s
3	XGBoost	개별모델	0.5086	0.8175	0.4224	0.6390	0.8609	1.78s
4	Hard Voting	앙상블	0.5252	0.7511	0.4779	0.5830	0.8813	14.92s
5	Soft Voting	앙상블	0.5286	0.8155	0.4770	0.5927	0.8809	14.45s





6.4 결과 요약 및 비교

- 앙상블 모델(랜덤포레스트, XGBoost, Voting)이 단일 모델(결정트리 등) 대비 일관적으로 우수한 성능을 보임
- 불균형 데이터 상황에서 F1 Score, ROC-AUC 등에서 앙상블 모델의 강점이 부각됨

7. 모델 해석 및 비즈니스 인사이트

7.1 변수 중요도 분석

- economic_score, credit_grade, poutcome(과거 캠페인 결과), emp.var.rate(고용률), age, campaign(연락 횟수)가 변수 중요도 상위권
- 경제력 및 신용등급이 높을수록 예금 가입 확률 상승
- 과거 캠페인 성공 경험, 고용률 하락, 나이 증가, 적정 연락 횟수(3~4회)는 모두 가입률을 높이는 요인

7.2 거시경제 vs 개인 특성 영향

- 고용률, 금리 등 거시경제 변수는 가입률과 강한 음의 상관관계(경기 침체, 금리 하락 시 가입률 상승)
- 신뢰지수, 물가지수 등은 약한 양의 상관관계
- 개인 특성(경제력, 신용등급, 직업, 교육 등) 또한 예측력에 크게 기여
- 금융위기 시기에는 경제 불안 심리가 예금 선호로 이어지는 예방적 저축(Precautionary savings) 현상 나타남

7.3 실무 적용 전략

- 가입 확률 80% 이상: 즉시 연락 및 프리미엄 상품 제안
- 50~80%: 2차 타겟, 맞춤형 혜택 제공
- 30~50%: 정보성 캠페인 위주 접근
- 30% 미만: 캠페인 대상에서 제외하여 비용 최소화
- 고위험군(신용등급 E, 경제점수 하위 20%): 리스크 관리와 신중한 접근
- 적정 연락 횟수(3~4회)는 긍정적, 과도한 연락은 오히려 역효과

7.3.1 소스코드

```
# 1. 테스트 데이터에 대해 가입 확률 예측
best_model = voting_soft # 또는 원하는 모델(best_xgb 등)
y_pred_proba = best_model.predict_proba(X_test)[: , 1] # 가입 확률 (1 클래스)

# 2. 확률 구간별 그룹 분류
def assign_group(prob):
    if prob >= 0.8:
        return "즉시 연락/프리미엄"
    elif prob >= 0.5:
        return "2차 타겟/맞춤혜택"
    elif prob >= 0.3:
        return "정보성 캠페인"
    else:
        return "캠페인 제외"

group_labels = [assign_group(p) for p in y_pred_proba]

# 3. 결과 데이터프레임 생성
result_df = X_test.copy()
result_df['가입확률'] = y_pred_proba
```

```

result_df['마케팅그룹'] = group_labels

# 4. 그룹별 인원 및 비율 확인
group_summary = result_df['마케팅그룹'].value_counts(normalize=True).mul(100).round(2)
print("마케팅 그룹별 분포(%)")
#print(group_summary)
group_summary.head_att()

print()
# 5. 예시 상위 10명 출력
result_df_sorted = result_df.sort_values('가입확률', ascending=False)
result_df_sorted[['가입확률', '마케팅그룹']].head_att(10)
#print(result_df_sorted[['가입확률', '마케팅그룹']].head())

```

7.3.2 결과

마케팅 그룹별 분포(%)

인덱스	proportion
index	proportion
캠페인 제외	74.29
정보성 캠페인	11.85
2차 타겟/맞춤혜택	11.79
즉시 연락/프리미엄	2.08

	가입확률	마케팅그룹
	가입확률	마케팅그룹
39655	0.9123	즉시 연락/프리미엄
40476	0.9074	즉시 연락/프리미엄
40487	0.9062	즉시 연락/프리미엄
40360	0.906	즉시 연락/프리미엄
40473	0.904	즉시 연락/프리미엄
40413	0.9036	즉시 연락/프리미엄
40586	0.9031	즉시 연락/프리미엄
40163	0.9012	즉시 연락/프리미엄
39498	0.901	즉시 연락/프리미엄
39644	0.9008	즉시 연락/프리미엄

7.3.4 해석

상위 10명의 가입확률이 모두 0.90 이상으로, 이들은 '즉시 연락/프리미엄' 그룹에 속합니다. 즉, 모델이 예측한 가입 확률이 매우 높은 고객을 잘 선별 해냈음을 보여줍니다.

전체 분포를 보면:

캠페인 제외(가입확률 0.3 미만): 73.89% (대부분의 고객은 가입 가능성이 낮음)

즉시 연락/프리미엄(0.8 이상): 2.09% (가입 가능성이 매우 높은 핵심 타겟)

2차 타겟/맞춤혜택(0.5~0.8): 11.8%

정보성 캠페인(0.3~0.5): 12.22%

7.3.5 해석 결론

가입확률이 높은 고객(상위 2%)을 효과적으로 분류하고 있음.

실제 마케팅에서 즉시 연락/프리미엄 그룹을 우선적으로 공략하면 높은 성공률을 기대.

반대로, '캠페인 제외' 그룹은 마케팅 효율을 위해 제외하는 것이 합리적.

즉, 모델이 가입 확률이 높은 고객을 잘 구분해주고 있다는 의미입니다.

8. 결론

- 불균형 데이터 환경에서 앙상블 모델(랜덤포레스트, XGBoost, Voting)이 가장 우수한 성능을 기록함

- 거시경제 지표와 개인 특성 모두 예측에 유의미한 영향력을 미침
- 파생 변수와 Conservative 인코딩 기법이 모델 해석력 및 성능에 기여함
- 금융위기 시기 예금 가입의 결정요인은 경제 불안 심리에 기인한 예방적 저축 현상임이 확인됨
- 외부 변수(가구소득, 지역, 경쟁사 정보 등)가 미포함된 한계와, 실전 적용을 위한 추가 실험 필요성 있음.

9. 교훈

- 하이퍼파라미터 탐색 시 너무 많은 조합을 하면 시간이 오래 걸림
 - 좁은 범위를 블록 단위로 좁혀 가는 것이 효율 적임
- 분석을 위한 jupyter notebook을 사용시 프로그램을 잘 구분할 것
- 불필요한 파생 인자를 생성하지 말 것

☆ 참고 문헌

Leland, H. E. (1968). Saving and Uncertainty: The Precautionary Demand for Saving
(예방적 저축의 개념을 처음으로 정립한 논문 AI 검색)

10. 부록

부록 1: 하이퍼파라미터

연구 완료일: 2025년 7월 28일 | 연구자: 김명환 (5팀) | 소속: AI 4기 5팀 스프린트 미션 4

부록 1: 하이퍼파라미터

```
def create_hyperparameter_test2():
    """
    모든 하이퍼파라미터 조합을 생성하는 함수
    각 모델별로 실제 가능한 모든 조합을 for문으로 생성
    """
    import itertools

    print("=== 모든 하이퍼파라미터 조합 생성 ===")

    dt_param_grid = []

    # 1. Decision Tree

    dt_max_depths = [4, 5, 6]
    dt_min_samples_splits = [30, 20, 40]
    dt_min_samples_leafs = [5, 10, 15]
    #dt_criteria = ['gini', 'entropy']
    dt_criteria = ['gini']
    # 은행의 텔레마케팅은 랜덤 방식으로 진행된다.
    # 랜덤 방식에 적합한 gini가 더 적합할 것으로 판단된다.

    #dt_max_features_list = [None, 'sqrt', 'log2']
    dt_max_features_list = [None]
    # 현재 y 성공율이 11% 정도로 특성을 고려하기 보다는
    # 모든 특성을 검토하자 None

    for max_depth in dt_max_depths:
        for min_samples_split in dt_min_samples_splits:
            for min_samples_leaf in dt_min_samples_leafs:
                for criterion in dt_criteria:
                    for max_features in dt_max_features_list:
                        dt_param_grid.append({
                            'max_depth': max_depth,
                            'min_samples_split': min_samples_split,
                            'min_samples_leaf': min_samples_leaf,
                            'criterion': criterion,
                            'max_features': max_features
                        })

    # 2. Random Forest
    rf_param_grid = []

    rf_n_estimators_list = [95, 100, 150]
    rf_max_depths = [10, 15, 20]
    rf_min_samples_splits = [10, 20, 30]
    # 현재 11%로 편향이 심한 데이터로
    # min_samples_split을 낮게 설정하여 세밀한 분할 이 필요 하다.

    rf_min_samples_leafs = [2, 3, 4]
    # 현재 11%로 편향이 심한 불균형 데이터 전반적인 검사를 해야 함.

    # rf_max_features = ['sqrt', 'log2', None]
    rf_max_features = ['sqrt']
    # 현재 y 성공율이 11% 정도로 특성을 고려하기 보다는
    # 모든 특성을 검토하자 None

    # 모든 조합 생성
    for n_estimators in rf_n_estimators_list:
```

```

for max_depth in rf_max_depths:
    for min_samples_split in rf_min_samples_splits:
        for min_samples_leaf in rf_min_samples_leafs:
            for max_features in rf_max_features:
                rf_param_grid.append({
                    'n_estimators': n_estimators,
                    'max_depth': max_depth,
                    'min_samples_split': min_samples_split,
                    'min_samples_leaf': min_samples_leaf,
                    'max_features': max_features,
                    'bootstrap': True
                })

# 3. Gradient Boosting
gb_param_grid = []

# 각 파라미터의 가능한 값들 정의
gb_n_estimators = [150, 200, 250]
# 일단 colab에서 실행할 것이니 값을 낮게 설정

gb_learning_rates = [0.005, 0.010]
# 현재 y 성공율이 11% 정도로 편향이 심한 데이터로
# 학습률을 낮게 설정하여 세밀한 학습이 필요 하다.

gb_max_depths = [6, 7, 8]
# 과적합 방지를 위해서는 많은 것이 좋으나 (현실적 11% 마케팅)
# 추후 성공률 비교를 위하여 높은 깊은 값도 추가

gb_subsamples = [0.8, 0.1, 0.15]
# 11%는 반대로 89%의 실패를 분류해도 성공 분류가 되어 버린다.
# 따라서 샘플링 비율을 낮추어 실패를 더 많이 포함시킨다.
# 현재 y 성공율이 11% 정도로 편향이 심한 데이터로
# 샘플링 비율을 낮게 설정하여 세밀한 학습이 필요 하다.

gb_min_samples_splits = [50, 100]
# 50: 가입 고객의 세밀한 패턴까지 포착 시도
# 100: 실무에서 가장 무난한 선택
# 150: 오버피팅 방지 및 일반화 성능 확보
# 테스트 결과를 보고 결정하자

gb_min_samples_leafs = [10, 20, 30]
# 현재 데이터 베이스 규모 검토

# 모든 조합 생성
for n_estimators in gb_n_estimators:
    for learning_rate in gb_learning_rates:
        for max_depth in gb_max_depths:
            for subsample in gb_subsamples:
                for min_samples_split in gb_min_samples_splits:
                    for min_samples_leaf in gb_min_samples_leafs:
                        gb_param_grid.append({
                            'n_estimators': n_estimators,
                            'learning_rate': learning_rate,
                            'max_depth': max_depth,
                            'subsample': subsample,
                            'min_samples_split': min_samples_split,
                            'min_samples_leaf': min_samples_leaf
                        })

# 4. XGBoost 파라미터 조합 (모든 조합 생성)
xgb_param_grid = []

# 각 파라미터의 가능한 값들 정의

```



```

        'min_child_weight': min_child_weight,
        'subsample': subsample,
        'colsample_bytree': colsample_bytree,
        'reg_alpha': reg_alpha,
        'reg_lambda': reg_lambda
    })

print(f"Decision Tree: {len(dt_param_grid)}개 조합 생성 (모든 조합)")
print(f"Random Forest: {len(rf_param_grid)}개 조합 생성 (모든 조합)")
print(f"Gradient Boosting: {len(gb_param_grid)}개 조합 생성 (모든 조합)")
print(f"XGBoost: {len(xgb_param_grid)}개 조합 생성 (모든 조합)")
print(f"총 {len(dt_param_grid) + len(rf_param_grid) + len(gb_param_grid) + len(xgb_param_grid)}개 조합")

# 예상 조합 수 계산 및 표시
dt_expected = len(dt_max_depths) * len(dt_min_samples_splits) * len(dt_min_samples_leafs) * len(dt_criteria) *
len(dt_max_features_list)
rf_expected = len(rf_n_estimators_list) * len(rf_max_depths) * len(rf_min_samples_splits) * len(rf_min_samples_leafs) *
len(rf_max_features)
gb_expected = len(gb_n_estimators) * len(gb_learning_rates) * len(gb_max_depths) * len(gb_subsamples) *
len(gb_min_samples_splits) * len(gb_min_samples_leafs)
xgb_expected = len(xgb_n_estimators) * len(xgb_learning_rates) * len(xgb_max_depths) * len(xgb_min_child_weights) *
len(xgb_subsamples) * len(xgb_colsample_bytrees) * len(xgb_reg_alphas) * len(xgb_reg_lambdas)

print(f"\n조합 수 검증:")
print(f"Decision Tree: {dt_expected} (계산) = {len(dt_param_grid)} (실제) ✓")
print(f"Random Forest: {rf_expected} (계산) = {len(rf_param_grid)} (실제) ✓")
print(f"Gradient Boosting: {gb_expected} (계산) = {len(gb_param_grid)} (실제) ✓")
print(f"XGBoost: {xgb_expected} (계산) = {len(xgb_param_grid)} (실제) ✓")

return dt_param_grid, rf_param_grid, gb_param_grid, xgb_param_grid

```