

# 온라인 쇼핑 세션 구매 확률 예측 모델 보고서

## 온라인 쇼핑 세션 구매 확률 예측 모델 보고서

(UCI Online Shoppers Purchasing Intention Dataset 기반)

### 0. 요약 (Executive Summary)

- 목적
  - 온라인 쇼핑 웹사이트의 세션 단위로, 해당 방문에서 실제 구매(Revenue=1)가 발생할 확률을 예측하는 모델을 구축하였다.
  - 단순 모델 성능을 넘어서, 마케팅·기획·운영 팀이 실무에서 바로 활용 가능한 형태의 예측 도구와 데모 애플리케이션을 제공하는 것을 목표로 했다.
- 데이터
  - UCI Online Shoppers Purchasing Intention Dataset 사용
  - 세션 단위 데이터이며, 행동(log) 정보 + 세션 특성 + 방문자 특성을 포함
  - 타겟 변수: Revenue (0=미구매, 1=구매)
  - 구매(1) 비율이 낮은 불균형 데이터임
- 모델링 접근
  - 불균형 데이터에 강한 **Balanced Random Forest**를 핵심 알고리즘으로 선정
  - 전처리(수치형 RobustScaler, 범주형 OneHotEncoder)와 분류기를 Pipeline 으로 통합
  - 모델 목적에 따라 두 개의 최종 모델을 별도로 튜닝 및 저장
    - PR-AUC 최적화 모델: best\_pr\_auc\_balancedrf.joblib
    - F1 Score 최적화 모델: best\_balancedrf\_pipeline.joblib (+ threshold 내장)
- 주요 성능
  - PR-AUC 최적화 모델
    - Best CV PR-AUC: 0.756

- Test PR-AUC: **0.747**
  - Test ROC-AUC: **0.930**
  - F1 최적화 모델
    - OOF F1: **0.700**
    - Test F1: **0.674**
    - Test PR-AUC: **0.742**
    - Test ROC-AUC: **0.932**
    - Best threshold: **0.741**
  - 서빙 및 활용
    - `PurchaseModelAdapter` + `SessionProbabilityService` 레이어를 통해 모델 교체/확장에 유연한 구조를 구성
    - Streamlit 앱 2종 구현
      - 세션 구매 확률 계산기 (실제 세션 입력)
      - 가상 고객 페르소나 생성기 (시나리오 기반 세션 자동 생성)
- 

## 1. 프로젝트 개요

### 1.1 목적

본 프로젝트의 목적은 온라인 쇼핑 세션 단위로 해당 방문에서 구매가 발생할지 (**Revenue=1**)를 예측하는 머신러닝 모델을 구축하는 것이다.

단순히 정확한 예측값을 산출하는 것에서 나아가, 다음을 목표로 한다.

- 마케팅, 기획, 운영 담당자가 이해하기 쉬운 **“구매 확률”** 형태로 제공
- 예측 결과에 기반해 **쿠폰 발송, 리타겟팅, 이탈 방지 액션** 등 실제 의사결정에 활용 가능
- 데모/교육용으로도 사용 가능한 **인터랙티브 앱** 제공

### 1.2 사용 데이터

- 데이터셋: **UCI Online Shoppers Purchasing Intention Dataset**
- 단위: 1행 = 1 세션(session)
- 타겟 변수: `Revenue` (0 = 미구매, 1 = 구매)

주요 feature 그룹은 다음과 같다.

- **행동 지표**

- Administrative , Informational , ProductRelated
- Administrative\_Duration , Informational\_Duration , ProductRelated\_Duration  
→ 어떤 유형의 페이지를 얼마나 많이, 얼마나 오래 봤는지

- **세션 품질**

- BounceRates , ExitRates , PageValues , SpecialDay  
→ 이탈 정도, 페이지의 금전적 가치, 특수일 여부 등

- **고객/세션 속성**

- Month , OperatingSystems , Browser , Region
- TrafficType , VisitorType , Weekend

데이터는 실제 이커머스 환경과 마찬가지로, **구매(Revenue=1) 비율이 매우 낮은 불균형 데이터**이다.

따라서 본 프로젝트에서는 **불균형 데이터에 적합한 알고리즘 + 적절한 평가 지표 선택**을 핵심 이슈로 삼았다.

---

## 2. 모델링 접근 개요

### 2.1 알고리즘 선택: Balanced Random Forest

여러 분류 모델 후보를 검토한 끝에, 본 프로젝트에서는 **Balanced Random Forest(BRF)**를 핵심 알고리즘으로 채택하였다.

- **불균형 데이터 대응**

- 일반 Random Forest는 다수 클래스(미구매)에 편향될 위험이 있음
- Balanced Random Forest는 각 트리 학습 시 **클래스별 샘플 수를 균형 있게 맞추는 방식**으로  
소수 클래스(구매)의 패턴을 학습에 충분히 반영

- **설명 가능성(Explainability)**

- 트리 기반 모델이라 feature importance, 분기 규칙 등을 통해 "어떤 행동/특징이 구매에 영향을 미쳤는지"를 설명하기 상대적으로 쉽다.

- **성능과 실용성의 균형**

- ROC-AUC, PR-AUC 등 주요 지표에서 높은 성능
- 예측 속도 또한 실무 환경에서 충분히 빠른 수준

## 2.2 전처리 및 파이프라인 구성

모델링은 전처리와 분류기를 하나의 파이프라인으로 묶어 수행하였다.

- 컬럼 분리

- 수치형: `num_cols = X_train.select_dtypes(include=["number"]).columns`
- 범주형: `cat_cols = X_train.select_dtypes(include=["object", "bool", "category"]).columns`

- 전처리

- 수치형( `num_cols` ): `RobustScaler` 적용  
→ 이상치에 덜 민감한 스케일링
- 범주형( `cat_cols` ): `OneHotEncoder(handle_unknown="ignore")` 적용  
→ 학습 시 없던 새로운 카테고리도 테스트 시 에러 없이 처리

- 파이프라인 구성

```
preprocess = ColumnTransformer(  
    transformers=[  
        ("num", RobustScaler(), num_cols),  
        ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),  
    ],  
    remainder="drop",  
)  
  
base = BalancedRandomForestClassifier(  
    n_jobs=-1,  
    random_state=RANDOM_STATE  
)  
  
pipe = Pipeline([  
    ("preprocess", preprocess),  
    ("model", base),  
)
```

이 파이프라인을 기반으로, 각각 **PR-AUC 최적화**와 **F1 최적화**를 위한 하이퍼파라미터 탐색을 진행했다.

## 3. 평가 지표 설계

### 3.1 핵심 지표

본 프로젝트에서 사용한 주요 평가지표는 다음과 같다.

#### 1. ROC-AUC

- 참양성률(TPR)과 거짓양성률(FPR)의 관계를 통합적으로 측정
- 전체 구간에서 분류기의 전반적인 성능을 평가하는 전통적인 지표

#### 2. PR-AUC (Average Precision)

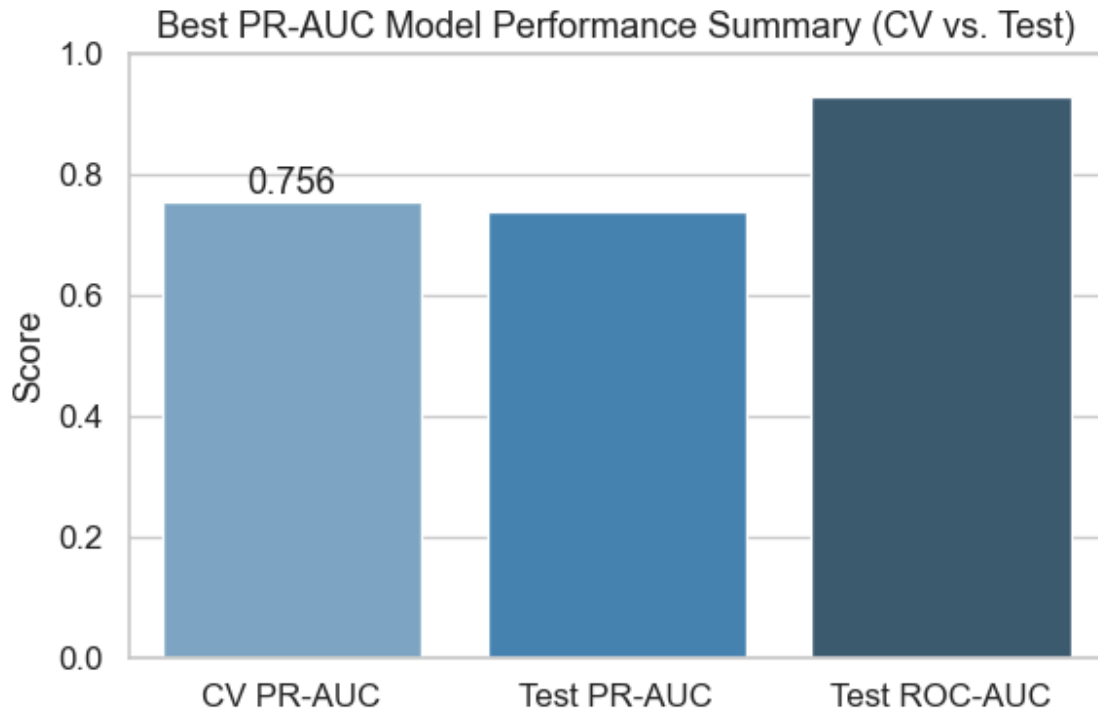
- Precision-Recall 곡선 아래의 면적
- **양성 클래스 비율이 낮을 때, "모델이 양성이라고 예측한 샘플 중 실제 양성 비율"을 잘 반영**

→ "예측 확률이 높은 사람일수록 진짜로 구매자일 확률이 높다"라는 **정렬 능력**을 평가하는 지표(threshold-free 지표)

- 본 데이터처럼 구매(1)가 적은 경우 매우 중요한 지표

#### 3. F1 Score

- Precision과 Recall의 조화 평균
- 특정 threshold에서 **양성/음성 오분류의 균형**을 실제로 보는 지표
- 실제 운영에서는 threshold 선택까지 포함한 모델이 중요함



[그림 3-1] PR-AUC 최적화 모델의 교차검증(CV) PR-AUC와 테스트(Test) PR-AUC/ROC-AUC 막대그래프

## 3.2 모델 선정 전략

### • PR-AUC 최적화 모델

- “구매 고객을 놓치지 않고 잘 잡아내는 모델”에 초점
- **확률 분포 / 순위**가 좋아지는 방향으로만 모델을 튜닝
- 타깃팅/분석/시뮬레이션용 기준 모델로 활용

### • F1 최적화 모델

- Out-of-Fold 예측을 활용하여 F1이 최대가 되는 threshold를 찾고, 그 threshold를 기준으로 Test 성능을 평가
- Balanced RF 하이퍼파라미터 여러 조합  
→ 각 조합마다 OOF 예측으로 **threshold**를 움직여가며 F1이 최대인 지점을 탐색
- threshold까지 함께 튜닝한 뒤, **실제 운영에 바로 적용 가능한 형태로 사용**

PR-AUC 모델은 **점수/순위용**이라 **threshold**를 유연하게 두고, F1 모델은 실제 0/1 의사결정에 바로 쓸 수 있게 **threshold까지 포함한 ‘완성된 물’**로 제작, 결과적으로, 서로 다른 목적을 가진 모델 2개를 별도로 튜닝하고, 각각 `best_pr_auc_balancedrf.joblib`, `best_balancedrf_pipeline.joblib` 로 저장하였다.

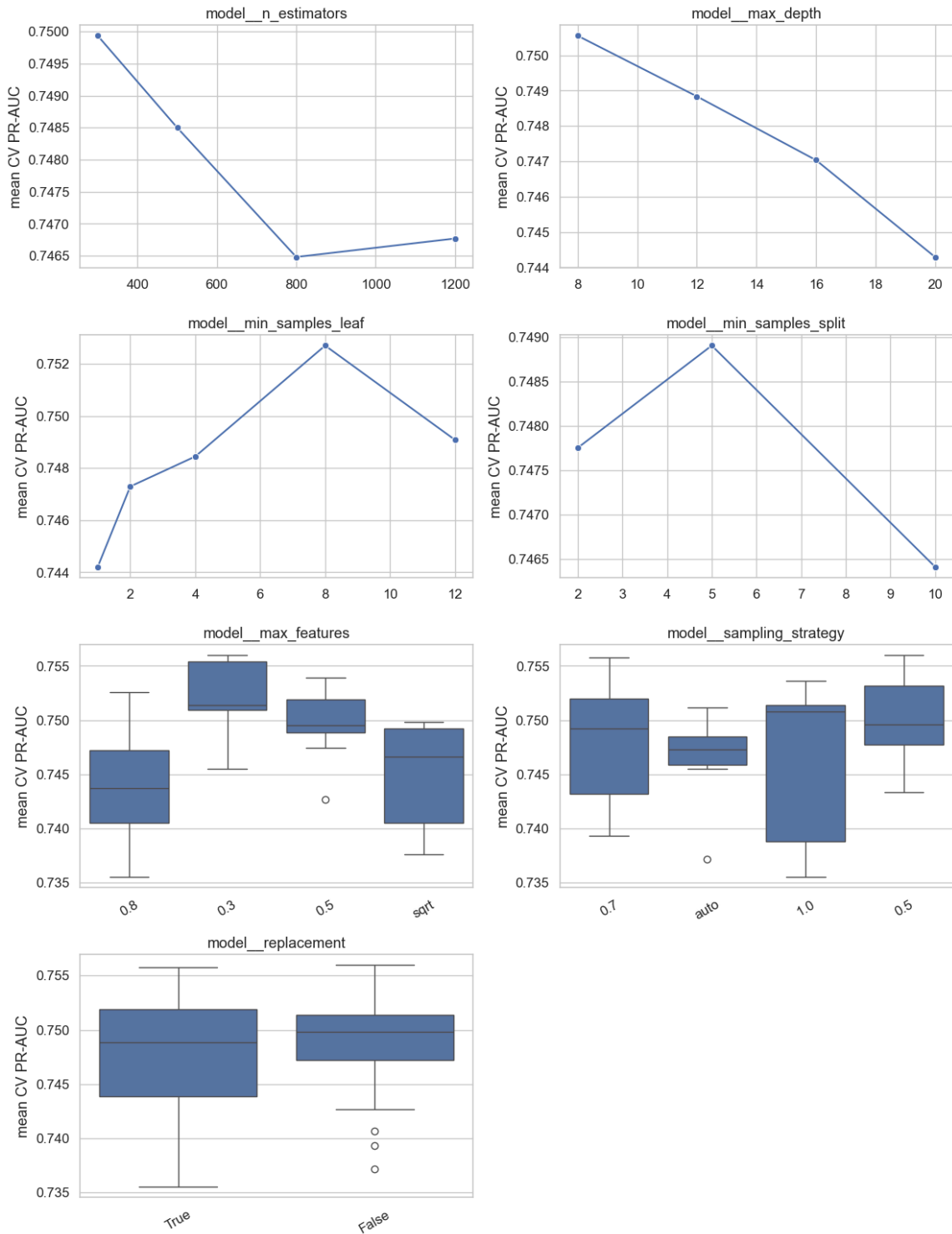
## 4. PR-AUC 기반 하이퍼파라미터 탐색

### 4.1 탐색 설정

PR-AUC 최적화를 위해 `RandomizedSearchCV` 를 사용했고, 탐색 공간은 다음과 같다.

- `model__n_estimators` : [300, 500, 800, 1200]
- `model__max_depth` : [None, 8, 12, 16, 20]
- `model__min_samples_leaf` : [1, 2, 4, 8, 12]
- `model__min_samples_split` : [2, 5, 10]
- `model__max_features` : ["sqrt", 0.3, 0.5, 0.8]
- `model__sampling_strategy` : ["auto", 0.5, 0.7, 1.0]
- `model__replacement` : [False, True]
- 교차검증: `StratifiedKFold(n_splits=5, shuffle=True, random_state=42)`
- 스코어: `scoring = "average_precision"` (PR-AUC)

Changes in CV PR-AUC by hyperparameter



[그림 4-1] PR-AUC 탐색 과정에서의 `n_estimators`, `max_depth`, `min_samples_leaf` 등 주요 하이퍼파라미터 값에 따른 평균 CV PR-AUC 변화 그래프

## 4.2 최종 PR-AUC Best 파라미터 및 성능



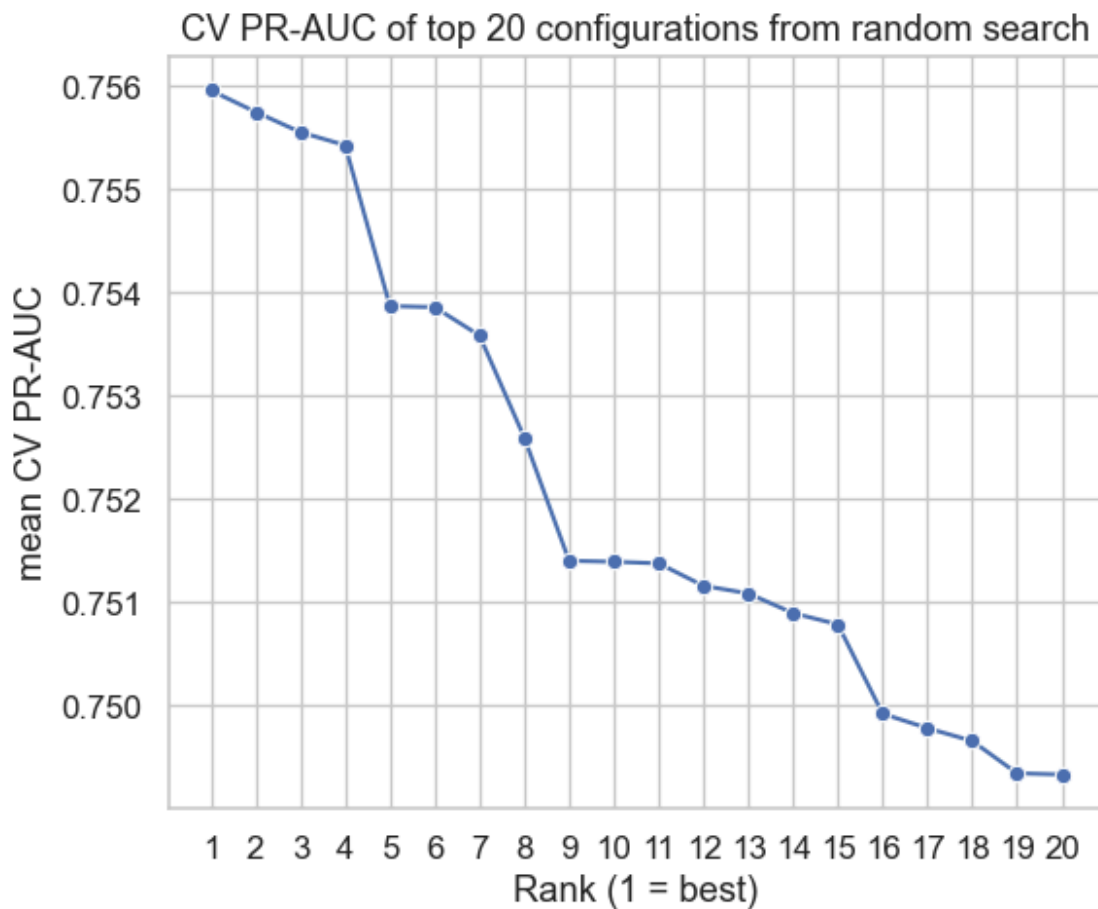
Best CV PR-AUC: 0.7559602666233075

Best params:

```
{  
  "model__sampling_strategy": 0.5,  
  "model__replacement": false,  
  "model__n_estimators": 300,  
  "model__min_samples_split": 5,  
  "model__min_samples_leaf": 1,  
  "model__max_features": 0.3,  
  "model__max_depth": 8  
}
```

TEST PR-AUC : 0.7466332225549133

TEST ROC-AUC: 0.9302589663454292



[그림 4-2] PR-AUC 기준 상위 20개 하이퍼파라미터 조합의 CV PR-AUC 랭킹 플롯

- 해석

- `n_estimators = 300` : 충분히 많은 트리로 복잡한 패턴을 포착하면서도 학습/예측 시간은 과도하지 않게 유지
- `max_depth = 8` , `min_samples_split = 5` : 적절한 깊이와 분할 조건으로 과적합 방지
- `min_samples_leaf = 1` : 리프 노드 최소 샘플 수를 1로 허용해 미세한 패턴까지 포착
- `max_features = 0.3` : 각 노드에서 전체 feature의 30%만 사용 → 트리 간 다양성 확보
- `sampling_strategy = 0.5` : 각 트리 학습 시 소수 클래스 비율을 인위적으로 높여, 구매(1) 패턴이 충분히 반영되도록 조절

이 모델은 **PR-AUC 기준으로 최적화된 모델**이며, `best_pr_auc_balancedrf.joblib` 으로 저장하여 이후 분석/데모에서 재사용한다.

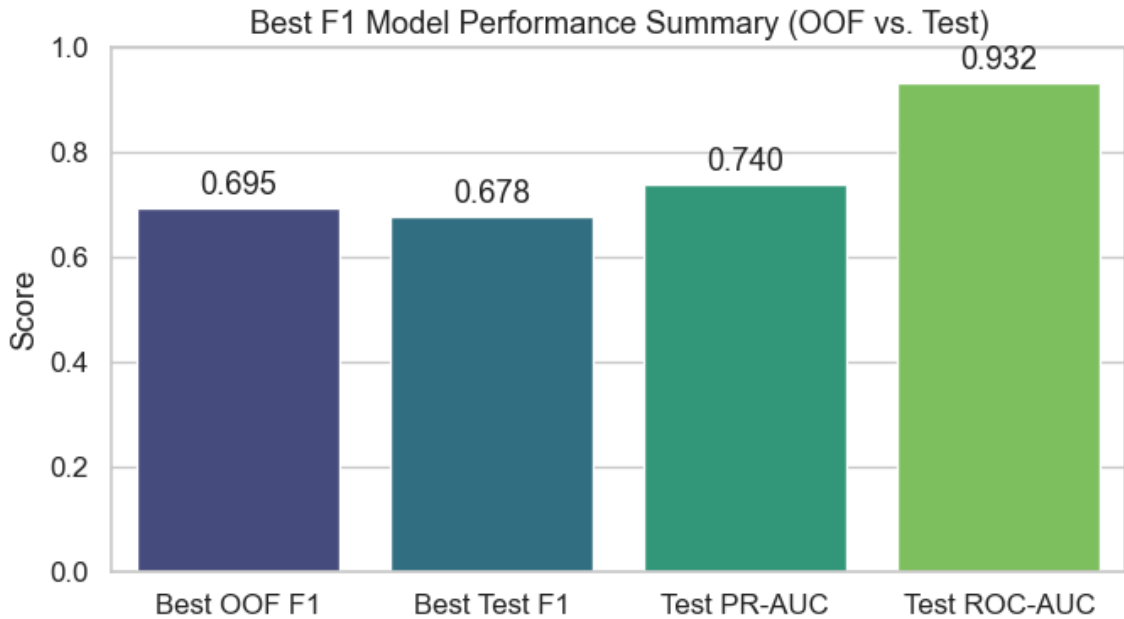
## 5. F1 Score 기반 하이퍼파라미터 및 Threshold 탐색

### 5.1 탐색 절차

F1 최적화는 다음 구조로 진행하였다.

#### 1. 하이퍼파라미터 그리드 정의

```
n_estimators    = 1200 (고정)
max_depth       ∈ {None, 8, 12, 16}
min_samples_leaf ∈ {1, 2, 4, 8}
max_features    ∈ {"sqrt", 0.5, 0.8}
min_samples_split = 2 (고정)
sampling_strategy = "auto"
replacement     = False
```



[그림 5-1] F1 최적화 모델의 OOF F1, Test F1, Test PR-AUC, Test ROC-AUC 요약 막대그래프

## 2. Out-of-Fold(OOF) 예측 기반 threshold 탐색

- StratifiedKFold(5-fold)를 사용하여 각 fold의 검증셋 예측확률을 수집
- 전체 OOF 예측에 대해 `precision_recall_curve` 를 계산
- threshold별 F1 값을 계산한 뒤, **F1이 최대가 되는 threshold를 선택**
- 이 로직을 `best_f1_threshold()` 함수로 구현

## 3. 각 조합별 Test 성능 평가

- 위에서 찾은 threshold를 사용해 train 전체로 재학습
- Test셋에서 F1, PR-AUC, ROC-AUC 계산
- **Test F1이 가장 높은 조합을 최종 Best F1 모델로 선정**

## 5.2 최종 Best F1 설정 및 성능

최고 Test F1을 달성한 설정과 성능은 아래와 같다.

```
=== BEST CONFIG (by TEST F1) ===
{
  "params": {
    "n_estimators": 1200,
    "max_depth": None,
    "min_samples_leaf": 8,
    "min_samples_split": 2,
```

```

    "max_features": 0.5,
    "sampling_strategy": "auto",
    "replacement": False
},
    "thr": 0.7412374222999228,
    "oof_f1": 0.6995734308343593,
    "test_f1": 0.6740467404674046,
    "test_pr_auc": 0.7415823565968093,
    "test_roc_auc": 0.9319585271979982
}

```

## • 하이퍼파라미터 해석

- `n_estimators = 1200`

→ 트리 수를 충분히 많이 두어 안정적인 앙상블 형성

- `max_depth = None` , `min_samples_leaf = 8`

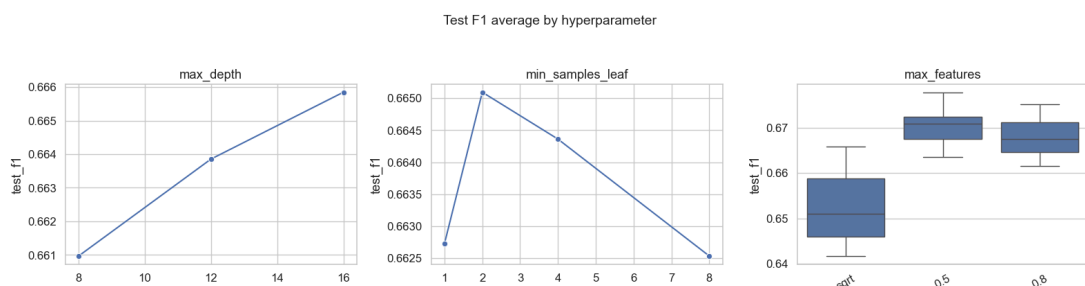
→ 최대 깊이는 제한하지 않지만, 리프 최소 샘플 수를 8로 설정하여 지나치게 복잡한 분할을 방지하고 **일반화 성능을 확보**

- `max_features = 0.5`

→ 각 노드 분할 시 전체 feature의 절반을 사용하여 다양성과 성능 균형을 맞춤

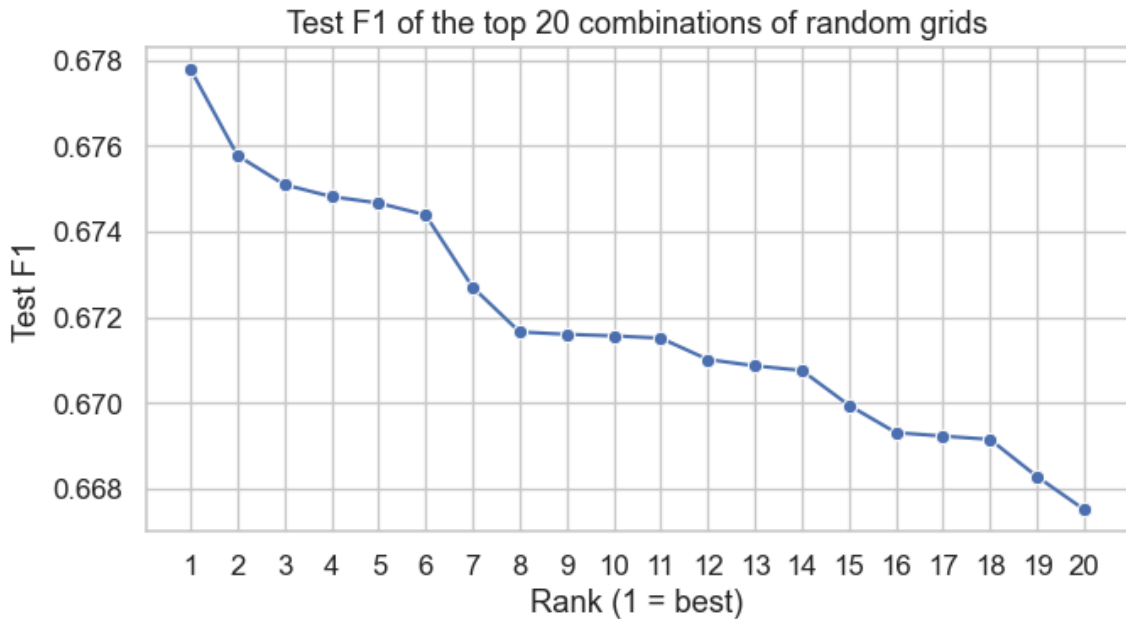
- `sampling_strategy = "auto"`

→ 불균형 비율을 자동으로 조정하는 라이브러리 기본 전략 사용



[그림 5-2] F1 최적화 과정에서 `max_depth` , `min_samples_leaf` , `max_features` 에 따른 Test F1 평균 변화 그래프

## • 성능 요약



[그림 5-3] F1 기준 상위 20개 하이퍼파라미터 조합의 Test F1 랭킹 플롯

- OOF F1: 약 **0.700**
- Test F1: 약 **0.674**
- Test PR-AUC: 약 **0.742**
- Test ROC-AUC: 약 **0.932**

OOF와 Test F1 간 차이가 크지 않으며, 전체적인 분류 성능도 PR-AUC 모델과 유사한 수준이다.

Out-of-Fold 예측을 이용해 F1이 최대가 되는 threshold를 탐색한 결과, 약 0.74 부근에서 최댓값을 기록하여 이를 최종 운영 기준 threshold로 사용하였다.

## 6. 결론 및 향후 계획

### 6.1 결론

- 모델 측면
  - Balanced Random Forest 기반으로
    - PR-AUC 최적화 모델: Test PR-AUC 약 0.747, ROC-AUC 약 0.930
    - F1 최적화 모델: Test F1 약 0.674, PR-AUC 약 0.742, ROC-AUC 약 0.932
  - 불균형 데이터에서도 높은 성능과 안정성을 확보하였다.
- 운영 측면

- PR-AUC 모델은 **분석/연구/리포트용 기준 모델**로 활용
- F1 모델은 **임계값(threshold)을 포함한 실무형 모델**로, 쿠폰/푸시 등 실제 액션을 트리거하는데 사용 가능
- **구조 측면**
  - Adapter + Service 레이어를 도입함으로써
    - 모델 변경/추가에 대한 유연성
    - 다양한 앱/분석 코드에서의 재사용성을 확보하였다.

## 6.2 향후 계획

- 실제 서비스 로그와 결합한 **도메인 특화 재학습**
  - SHAP 등의 기법 도입을 통한 **정량적인 Feature 중요도 및 기여도 시각화**
  - 채널/디바이스/지역별 세분화 모델 또는 가중치 조정
  - A/B 테스트를 통한 “모델 기반 캠페인 vs 기존 캠페인” 성과 비교
  - 더 다양한 페르소나 시나리오를 추가하여, 비즈니스/교육 현장에서의 활용도 확대
-