온라인 쇼핑 세션 구매 확률 예측 모델 보고서

학습된 인공지능 모델

1. 요약 (Executive Summary)

1.1. 프로젝트 배경 및 목적

본 프로젝트는 온라인 쇼핑몰 방문 고객의 세션 데이터를 분석하여 **"이 고객이 이번 방문에서 실제로 구매를 할 것인가?"**를 실시간으로 예측하는 AI 모델을 구축하는 것을 목표로 한다.

특히 전체 방문자 중 실제 구매자는 극소수인 데이터 불균형(Class Imbalance) 환경을 극복하기 위해, 머신러닝(Balanced RF)과 딥러닝(DNN)을 포함한 다양한 모델링 기법을 적용하고 비교 검증하였다.

1.2. 핵심 성과

- **최종 선정 모델: Balanced Random Forest (BRF)**
- **비교 검증 모델:** Deep Neural Network (DNN), CatBoost, LightGBM
- **주요 성과:**
  - **Recall (재현율): 0.791** (잠재 구매 고객의 약 80%를 사전에 식별)
  - **PR-AUC: 0.765** (불균형 데이터 환경에서의 정밀도-재현율 최적화 달성)
  - **Serving:** Streamlit 기반의 실시간 추론 대시보드 및 API 인터페이스 구축
  - **Explainability: SHAP 라이브러리**를 연동하여, 개별 고객의 이탈/구매 원인을 분석하는 **Waterfall Plot 시각화 기능 구현 완료**.

---

2. 모델 아티팩트 및 파일 구성 (Artifacts)

핵심 파일들의 명세

파일명	모델 구분	역할 및 특징

`best_balancedrf_pipeline.joblib`	**F1 최적화 모델**	**Threshold 튜닝 모델.** F1-Score를 최대화하는 최적 임계값(`best_threshold`) 정보 포함.
`best_pr_auc_balancedrf.joblib`	**PR-AUC 최적화 모델**	**파라미터 고정 모델.** PR-AUC 최적화를 위해 사전 탐색된 하이퍼파라미터(n_est=300 등) 적용.
`dnn_model.h5`	**Comparison (딥러닝)**	Keras/TensorFlow로 학습된 심층 신경망 모델. 복잡한 비선형 패턴 분석용으로 활용

3. 모델 상세 명세 (Model Specifications)

본 프로젝트에서는 **머신러닝(Ensemble)**과 **딥러닝(DNN)** 두 가지 접근 방식을 모두 시도하였으며, 최종적으로 설명력과 재현율이 우수한 **Balanced RF를 메인 모델로 선정**했다.

3.1. Main Model: Balanced Random Forest

본 프로젝트는 데이터 불균형(Class Imbalance) 문제 해결을 위해 **Balanced Random Forest (BRF)** 알고리즘을 메인 모델로 선정하였다.
단, 비즈니스 목적(F1-Score 극대화 vs 순위 예측 신뢰도 확보)에 따라 최적의 성능을 낼 수 있도록 **두 가지 최적화 전략(Two-Track Strategy)**으로 모델을 이원화하여 구축하였다.

- **알고리즘:** Balanced Random Forest Classifier (`imbalanced-learn` 라이브러리 활용)
- **공통 파이프라인 구성:**
  - **Step 1 (Preprocessing):** `RobustScaler` (이상치에 강건한 스케일링), `OneHotEncoder` (범주형 변수 변환)
  - **Step 2 (Classifier):** 다수의 Decision Tree가 Majority Class를 Under-sampling 하여 학습하고, 다수결로 최종 예측.

3.1.1. Type A: F1-Score 최적화 모델 (`best_balancedrf_pipeline`)

이 모델은 정밀도(Precision)와 재현율(Recall)의 조화 평균인 **F1-Score를 극대화**하는 것을 목표로 한다. 모델의 구조적 파라미터는 일반적인 설정을 유지하되, 학습 후 **'결정 임계값(Threshold)'을 튜닝**하는 후처리(Post-processing) 전략을 사용하였다.

- **최적화 전략 (Threshold Tuning):**
  - 기본 확률(0.5)을 사용하지 않고, Validation 과정에서 F1-Score가 최대가 되는 최적의 임계값을 산출하여 적용한다.
  - 산출된 임계값(예: 0.74)은 메타데이터(`meta`)에 저장되어, 추론 시 동적으로 적용된다.

- **핵심 하이퍼파라미터 (Base Settings):**
  - `n_estimators`: **100** (기본 설정, 연산 효율성 및 Baseline 성능 확보)
  - `sampling_strategy`: **'auto'** (Majority Class의 샘플 수를 Minority Class와 1:1 비율이 되도록 자동 Under-sampling)
  - `max_features`: **'sqrt'** (개별 트리의 다양성 확보)

3.1.2. Type B: PR-AUC 최적화 모델 (`best_pr_auc_balancedrf`)

이 모델은 불균형 데이터 평가에 가장 적합한 **PR-AUC (Precision-Recall Area Under Curve)** 점수를 높이는 것을 목표로 한다. 임계값 조정보다는 사전 실험을 통해 도출된 **최적의 하이퍼파라미터 조합을 고정(Fixed)**하여, 모델이 출력하는 확률값(Probability) 자체의 정교함을 높이는 데 주력하였다.

- **최적화 전략 (Fixed Hyperparameters):**
  - PR-AUC 점수가 가장 높았던 파라미터 조합을 고정하여 학습하였다.
  - 트리의 개수를 늘리고 깊이를 제한하여, 일반화 성능을 높이고 과적합(Overfitting)을 억제하였다.
- **핵심 하이퍼파라미터 (Optimized Settings):**
  - `n_estimators`: **300** (Type A 대비 3배 증가시켜 예측의 분산을 줄이고 안정성 확보)
  - `max_depth`: **8** (트리의 깊이를 제한하여 훈련 데이터에 대한 과적합 방지)
  - `max_features`: **0.3** (전체 피처의 30%만 무작위로 사용하여 개별 트리의 독립성 강화)
  - `sampling_strategy`: **0.5** (Majority Class를 Minority Class의 2배수(0.5 비율) 정도로 Under-sampling 하여 정보 손실 최소화)
  - `min_samples_split`: **5** (노드 분할을 위한 최소 샘플 수를 높여 보수적인 학습 유도)

3.2. Comparison Model: Deep Learning (DNN)

본 프로젝트에서는 머신러닝(Random Forest)과의 성능 비교를 위해, 비선형 패턴 학습에 강한 **딥러닝(Deep Learning)** 모델을 추가로 구축.

- **모델 명칭:** Deep Neural Network (DNN)
- **프레임워크:** TensorFlow / Keras
- **학습 파일:** `dnn_model.h5`
- **네트워크 구조 (Architecture):**

- **입력층 (Input):** 17개의 피처를 받아들이는 진입점
- **은닉층 (Hidden Layers):**
  - `Dense(64, activation='relu')` : 데이터의 특징을 64차원으로 확장하여 학습
  - `Dense(32, activation='relu')` : 추상화된 특징을 다시 32차원으로 압축
  - `Dropout(0.3)` : 과적합(Overfitting) 방지를 위해 뉴런의 30%를 무작위 비활성화
- **출력층 (Output):** `Dense(1, activation='sigmoid')` (0~1 사이의 구매 확률 출력)

> 💡 선정 이유:
> 단순한 선형 관계가 아닌, 피처 간의 복잡한 상호작용(Interaction)을 신경망이 스스로 학습할 수 있는지 검증하기 위해 도입.

---

4. 성능 비교 및 선정 근거 (Benchmark)

테스트 데이터셋(Test Set) 기준 모델별 성능 비교표입니다.

지표 (Metrics)	Balanced RF (Main)	Deep Learning (DNN)	CatBoost	비고
Accuracy	0.892	0.865	**0.905**	전체 정확도는 CatBoost 우위
Recall (재현율)	**0.791**	0.584	0.621	**실제 구매자 식별 능력은 BRF 압도적**
PR-AUC	**0.765**	0.682	0.742	불균형 데이터 핵심 지표
ROC-AUC	0.925	0.885	**0.931**	전체 변별력은 유사함
Inference Time	**12ms**	45ms	8ms	딥러닝은 연산 비용이 높음

🏆 최종 선정 사유: Balanced RF

1. **Recall 최우선:** 이커머스 마케팅에서는 잠재 구매자를 놓치지 않는 것(Recall)이 중요하다. Balanced RF는 DNN 대비 **약 20%p 높은 재현율**을 보였다.
2. **데이터 효율성:** 데이터셋의 크기(약 12,000건)가 딥러닝이 성능을 발휘하기에는 다소 작아, 앙상블 기법인 RF가 더 효율적이었다.

3. **설명 가능성:** 트리 기반 모델은 변수 중요도(Feature Importance)를 명확히 추출할 수 있어 비즈니스 설득에 용이하다.

---

5. 데이터 인터페이스 (Data Interface)

5.1. 입력 변수 명세 (Input Features)

모델은 총 17개의 원본 피처를 입력받으며, 내부 파이프라인에서 자동으로 인코딩된다.

1. **행동 데이터 (Behavioral):**
   - `Administrative`, `Informational`, `ProductRelated` (페이지 조회 수)
   - `_Duration` (각 페이지 유형별 체류 시간)
2. **세션 품질 (Quality):**
   - `BounceRates` (이탈률), `ExitRates` (종료율), `PageValues` (페이지 가치)
3. **고객 속성 (Attributes):**
   - `SpecialDay`, `Month`, `OperatingSystems`, `Browser`, `Region`, `TrafficType`, `VisitorType`, `Weekend`

5.2. 출력값 정의 (Output)

- **Prediction:** `0` (이탈 예상) vs `1` (구매 예상)
- **Probability:** `0.00` ~ `1.00` (구매할 확률)
- **Threshold Strategy:** F1-Score 최적화를 통해 도출된 임계값 **0.74**를 적용할 시 정밀도를 높일 수 있다.

---

6. 추론 코드 가이드 (Inference Guide)

Python 환경에서 모델을 로드하고 예측하는 표준 코드이다. 머신러닝 모델과 딥러닝 모델의 호출 방식 차이에 유의해야 한다.

```
import joblib
import pandas as pd
import numpy as np
from tensorflow.keras.models import load_model

# --- 1. 데이터 준비 (Raw Data) ---
input_data = pd.DataFrame([{
```

```
    "Administrative": 5, "Administrative_Duration": 120.5,
    "Informational": 2, "Informational_Duration": 15.0,
    "ProductRelated": 45, "ProductRelated_Duration": 1200.2,
    "BounceRates": 0.001, "ExitRates": 0.015, "PageValues": 45.2,
    "SpecialDay": 0.0, "Month": "Dec", "OperatingSystems": 2,
    "Browser": 2, "Region": 1, "TrafficType": 2,
    "VisitorType": "Returning_Visitor", "Weekend": True
}])

# --- 2. Main Model (Balanced RF) 추론 ---
# 경로 수정: artifacts/
rf_artifact = joblib.load('artifacts/best_balancedrf_pipeline.joblib')
rf_model = rf_artifact['pipeline']
rf_prob = rf_model.predict_proba(input_data)[0, 1]

print(f"Balanced RF 구매 확률: {rf_prob:.2%}")

# --- 3. Comparison Model (DNN) 추론 ---
# 딥러닝 모델 로드
dnn_model = load_model('artifacts/dnn_model.h5')

# [중요] DNN 전용 전처리기 로드 (train_dnn.py에서 별도 저장함)
dnn_preprocessor = joblib.load('artifacts/dnn_preprocessor.joblib')

# 전처리 수행 (DataFrame → Numpy Array)
input_tensor = dnn_preprocessor.transform(input_data)

# 확률 예측
dnn_prob = float(dnn_model.predict(input_tensor, verbose=0)[0][0])
print(f"Deep Learning 구매 확률: {dnn_prob:.2%}")
```

7. 비즈니스 기대 효과 및 향후 계획

7.1. 비즈니스 기대 효과

- **마케팅 비용 최적화:** 구매 확률 상위 20% 고객에게만 집중적으로 쿠폰을 발송하여 비용 대비 효율(ROAS) 극대화.

- **이탈 방지:** 장바구니에 담았으나 이탈 확률이 높은 고객에게 실시간 팝업(넛지) 제공.
- **데이터 기반 의사결정:** 직관이 아닌 `PageValues`, `Month` 등 데이터 중요도에 기반한 UI/UX 개선.

7.2. 향후 모델 고도화 방안 (Roadmap)

1. **모델 앙상블 (Stacking):** Balanced RF와 DNN의 예측값을 메타 모델(Meta-model)의 입력으로 사용하여 예측 정확도 추가 확보.
2. **MLOps 파이프라인 구축:** 매월 수집되는 새로운 데이터를 자동으로 학습하고 배포하는 CI/CD 파이프라인(GitHub Actions + Streamlit Cloud) 구축.