E-commerce 고객 실시간 이탈 예측

김민경, 김유미, 민지현, 성현수

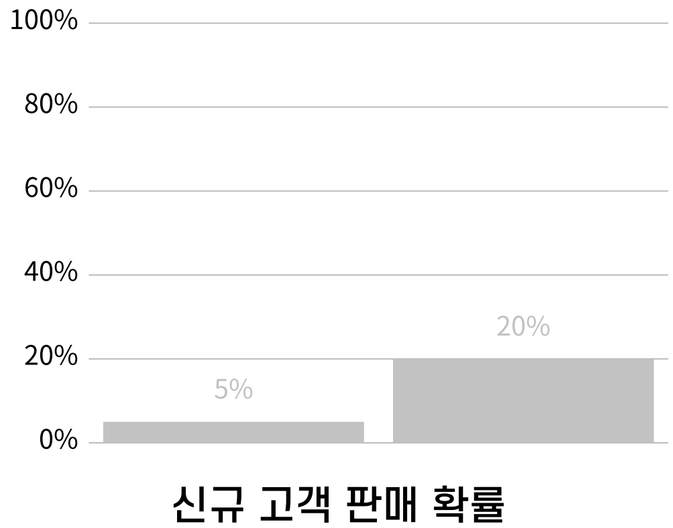
CONTENTS

01. 개요

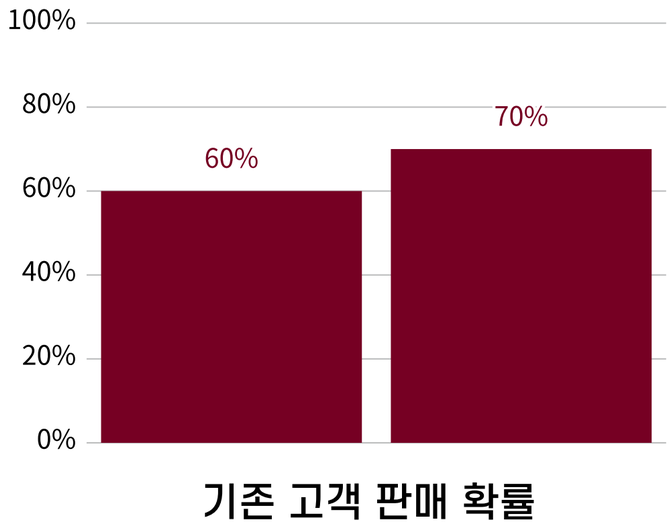
01 목표 및 배경

기존 고객 vs. 신규 고객

신규 고객에게 판매할 확률은 5-20%인 반면,
기존 고객에게 판매할 확률은 60-70%로 매우 높음



신규 고객 판매 확률



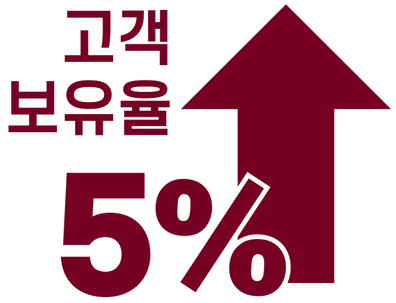
기존 고객 판매 확률

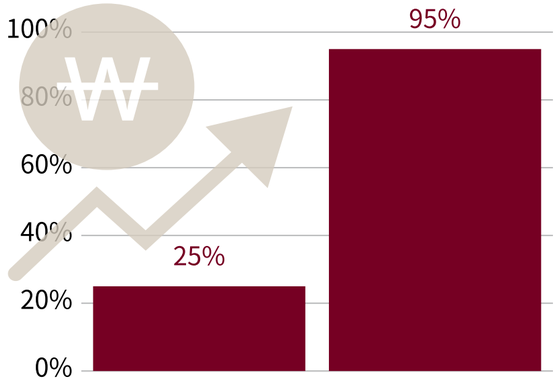
*출처: Marketing Metricss

01 목표 및 배경

B2C 기업: 기존 고객 유지의 중요성

고객 보유율을 5% 높이면,
이익이 최소 25%에서 최대 95%까지 증가함





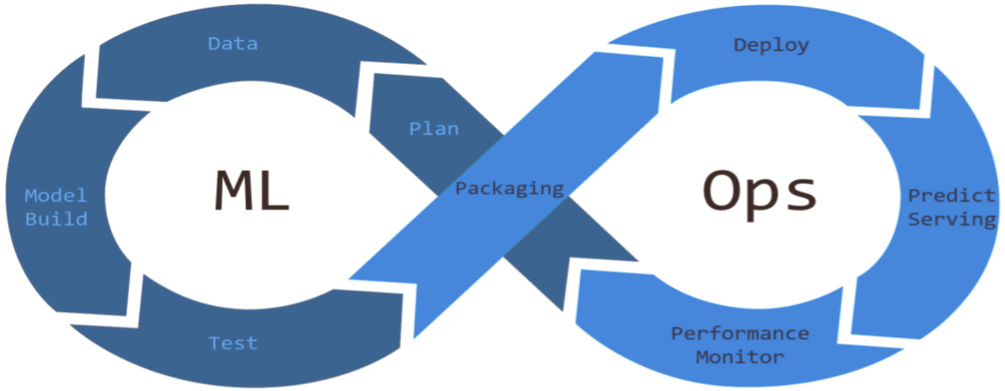
B2C기업: 기존 고객 유지 전략이 필요함

*출처: Bain&Company, ‘Prescription for cutting costs

02 문제 상황

MLOps 구축으로 기업 손실 최소화를 위한 분석의 효율 극대화

E-commerce B2C 기업의 기존 고객 유지 전략



MLOps 도입으로 실시간 고객 이탈 예측을 통해
기업 손실을 최소화하고 분석 프로세스 효율을 극대화하고자 함

03 해결방안

이커머스 B2C 기업의 고객 유지를 위한 MLOps 프로세스 설계

고객 이탈 예측을 위한 MLOps 설계 방안

① 이탈 예측 모델이 최신 데이터로 주기적으로 재학습

→ 모형이 지속적으로 성능을 유지하여 이탈 고객을 일관되게 포착하도록 할 것

② 실시간 모니터링

→ 고객이 이탈하기 전 구매 활동을 했을 때, 즉각적으로 징후를 탐지하고 대처할 것

02. 데이터 소개 및 이탈 정의

01 활용데이터

kaggle의 E-commerce 데이터

영국 소매업체의 실제 온라인 거래 내역 데이터 (2010/12/01 ~ 2011/12/09)

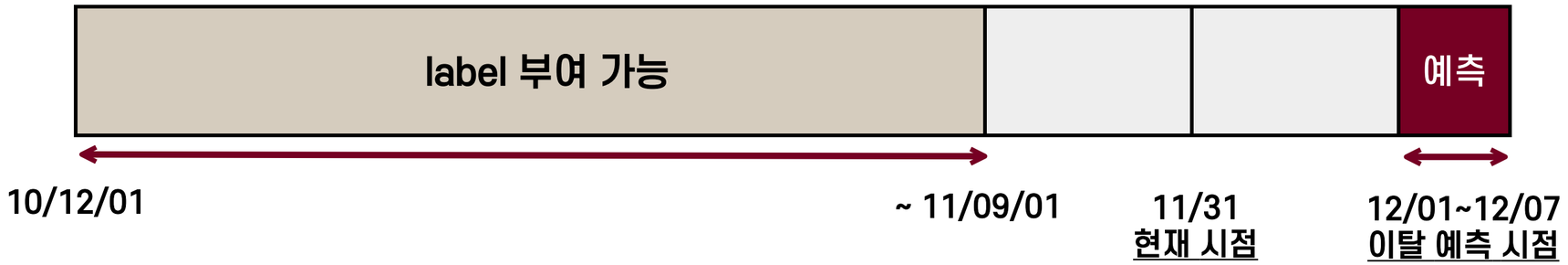
InvoiceNo	InvoiceDate	StockCode	Description	Quantity	UnitPrice	CustomerID	Country
구매 invoice number	구매 일시	종목 코드	품목명	구매 수량	개별 가격	고객 ID	고객 국가
결측값 : 0 unique : 25900	결측값 : 0	결측값 : 0 unique : 4070	결측값 : 1454 unique : 4223	결측값 : 0	결측값 : 0	결측값 : 135080 unique : 4372	결측값 : 0 unique : 38

구매 일시 컬럼이 초단위까지 표현되어 있어
실시간 분석, 예측 구현 가능

구매 주기를 계산해
이탈 징후에 대한 Label을 직접 설정

02 고객 이탈 정의

unlabel 데이터셋 : 90일 이상 구매 이력 없는 고객을 이탈로 정의



- 현재 시점: 2011/11/31 이라고 가정
- EDA를 통해 90일 이상 구매 이력이 없는 고객을 이탈로 정의
  → 현재 시점에서 90일 이전인 2011/09/01 까지만 고객 구매 이력을 알 수 있음
  → 2011/09/01까지만 Label 부여 가능
- 이탈 예측 시점: 2011/12/01 ~ 2011/12/07
- 예측 기간은 임의로 설정한 것이므로 유저에 따라 사전에 조정 가능

03. 파이프라인

01 전처리

이상치 제거

① 구매한 수량이 0이하 or 100초과인 로그 제거

	Quantity
count	541909.0
mean	9.55225
std	218.081158
min	-80995.0
25%	1.0
50%	3.0
75%	10.0
99%	100.0
max	80995.0

② 카드연체, 주문 취소에 해당되는 로그 제거

InvoiceNo	StockCode	Description
A563185	B	Adjust bad debt
C53679	D	Discount

③ 이후 중복되는 로그 제거

```
df.duplicated().sum()
```

```
5268
```

중복값 총 5268개 존재

01 전처리

이탈 분석에 맞는 전처리 진행

CustomerID	InvoiceDate	Quantity	UnitPrice
17850	12/1/2010 8:26	6	2.55
17850	12/1/2010 8:26	6	3.39
17850	12/1/2010 8:26	8	2.75

STEP1 개인의 구매 내역 로그를 일 단위로 통합

CustomerID	InvoiceDate	TotalPrice
17850	12/1/2010	57.64

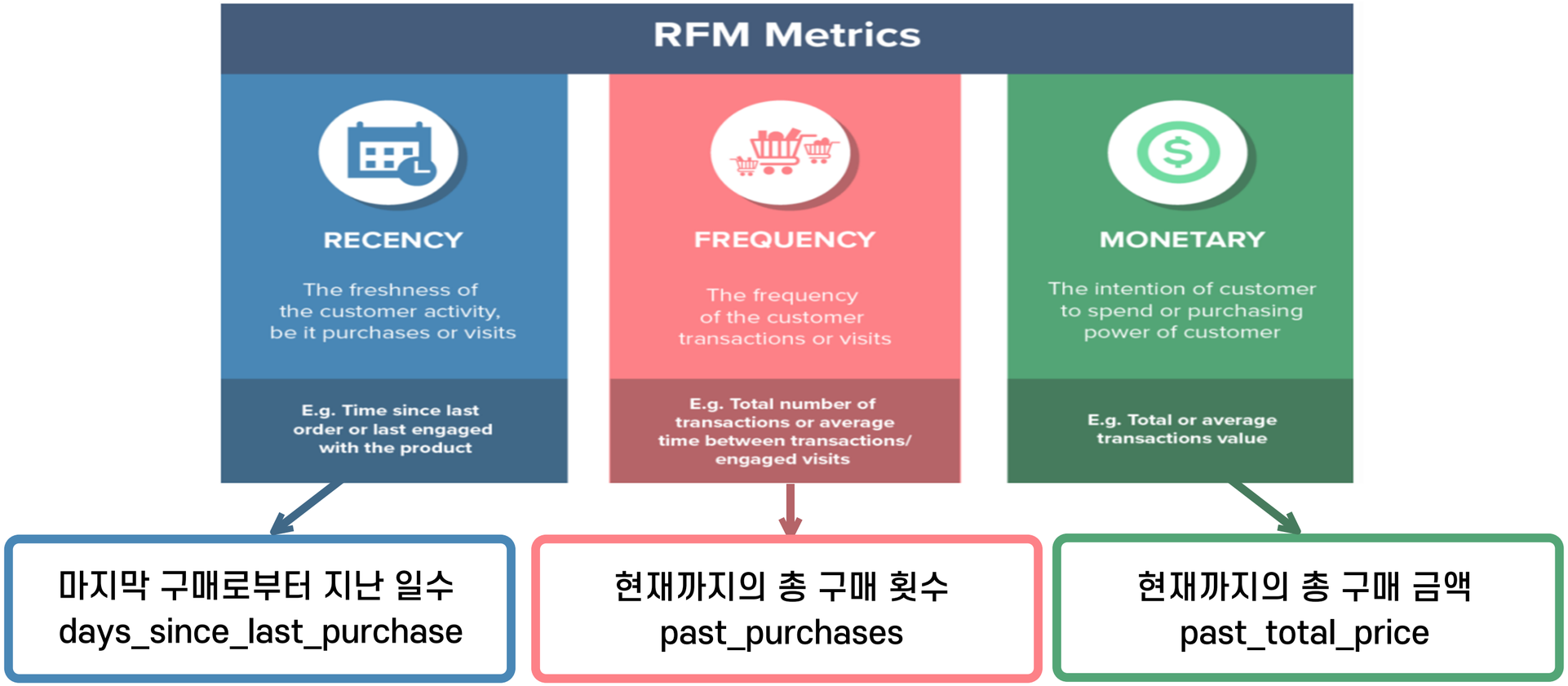
STEP2 통합 후 다음 접속까지의 간격이 특정기간 이상이면 Churn으로 정의

* 특정기간은 사전에 설정 가능하도록 모델 설계

01 전처리

파생변수

비즈니스 영역의 고객 세분화 과정에서 흔히 쓰이는 RFM 변수들을 기본적으로 생성



01 전처리

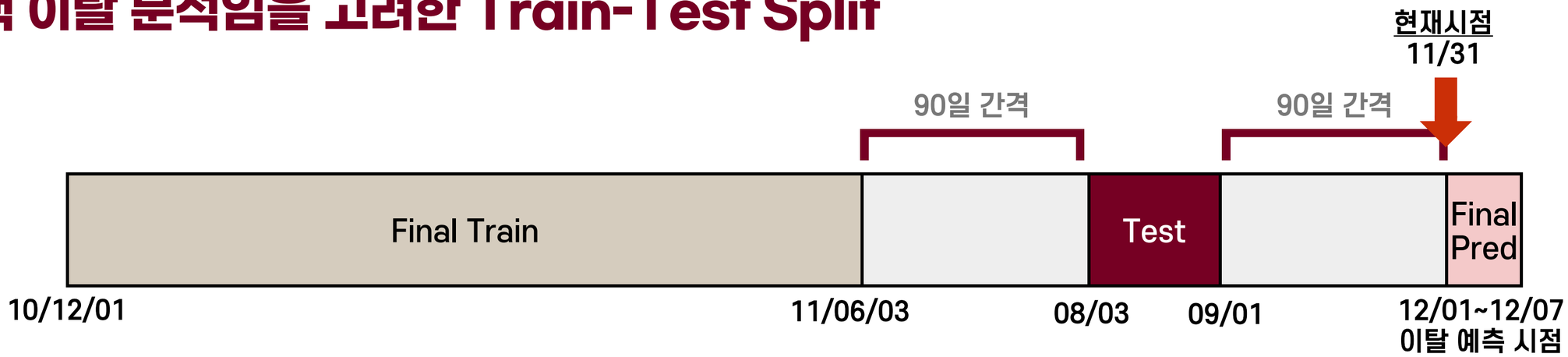
파생변수

RFM 변수들의 통계량, 과거 구매 내역, 당일 구매 내역 등 을 조합하여 총 21개의 파생변수 생성

파생변수	파생변수 설명
days_since_last_purchase_expanding_mean	현재까지의 구매간격의 평균
TotalPrice_expanding_mean	현재까지의 일 단위 총 구매 금액의 평균
Days_since_first_purchase	첫 구매로부터 지난 일수
Total_Price_amplitude	당일 총 구매 금액 - 현재까지의 일 단위 총 구매 금액의 평균
Quantity min/max	당일 구매 수량의 최솟/최댓값
UnitPrice min/max	당일 구매 물품 가격의 최솟/최댓값
past_churn	과거 이탈여부
...	...

02 Train-Test Split

고객 이탈 분석임을 고려한 Train-Test Split

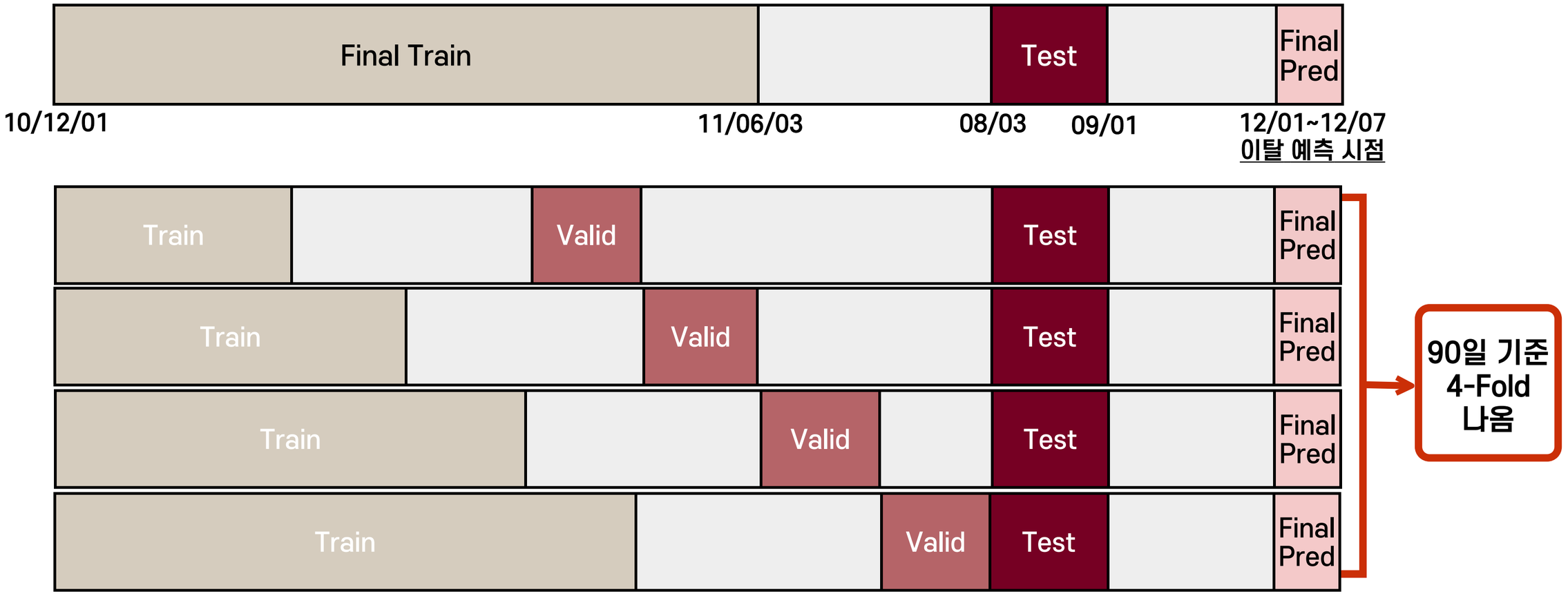


- 시계열 로그 데이터를 이용한 고객 이탈 분석임을 고려한 Train-Test Split
- 현재 시점이 2011/11/31 이라고 가정
- 각 시점에서 고객 구매 이력을 알 수 있는 90일 간격으로 Split 진행

➡ 데이터 특성을 고려해 Time Series CV를 통한 예측 진행

02 Train-Test Split

Time Series CV를 통한 예측



03 모델링

(1) Catboost & 변수선택

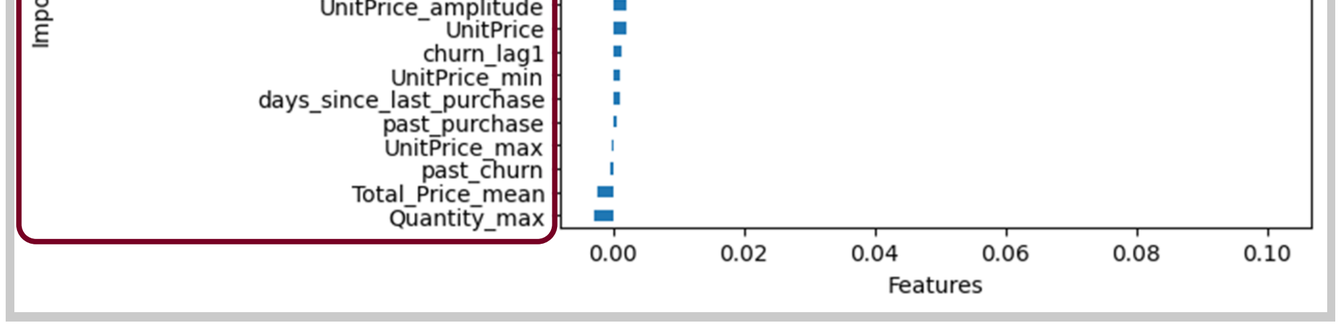
Timeseries cv를 활용한 변수선택

```
--- 2011-08-02 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-02-03 00:00:00
valid 시작 시점 : 2011-04-05 00:00:00
valid 마지막 시점 : 2011-05-04 00:00:00


--- 2011-09-01 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-03-04 00:00:00
valid 시작 시점 : 2011-05-05 00:00:00
valid 마지막 시점 : 2011-06-03 00:00:00


--- 2011-10-01 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-04-04 00:00:00
valid 시작 시점 : 2011-06-05 00:00:00
valid 마지막 시점 : 2011-07-03 00:00:00


--- 2011-10-31 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-05-04 00:00:00
valid 시작 시점 : 2011-07-04 00:00:00
valid 마지막 시점 : 2011-08-02 00:00:00
```



Permutation importance가 0.001 이하이면 변수 drop

03 모델링

(1) Catboost & Best Hyper parameter

- 시계열 데이터에 효율적인 GBM 기반 알고리즘
- 범주형 변수에 대해 특정 인코딩 장식으로 모델의 정확도와 속도가 우수

→ Base learner 모델들의 하이퍼파라미터 튜닝과정까지 파이프라인에 포함하면 너무 복잡해지기 때문에 따로 robust한 파라미터 세팅을 찾은 후 활용

Parameter	Value
depth	10
L2_leaf_reg	20
random_strength	20
iterations	100
class_weights	17

03 모델링

(2) ElasticNet & 변수선택

- 회귀계수 0인 변수들은 자동으로 적합에서 제외
- 회귀계수 0인 변수들이 CV Fold마다 자동 출력되어 영향력이 낮은 변수 확인 가능

```
--- 2011-08-02 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-02-03 00:00:00
valid 시작 시점 : 2011-04-05 00:00:00
valid 마지막 시점 : 2011-05-04 00:00:00


Accuracy on test set: 0.7284, AUC on test set: 0.7567
F1-score on test set: 0.6419
Precision on test set: 0.5251

[[500 227]
 [ 53 251]]
회귀계수 0인 변수: ['UnitPrice_min', 'Total_Price_mean', 'Total_Price', 'days_since_first_purchase', 'past_churn', 'churn_lag1', 'UnitPrice_amplitude']
```

Fold1

'UnitPrice_min', 'Total_Price_mean', 'Total_Price', 'days_since_first_purchase', 'past_churn', 'churn_lag1', 'UnitPrice_amplitude'

Fold2

'UnitPrice_min', 'Total_Price_mean', 'UnitPrice', 'Total_Price Expanding Mean'

Fold3

-

Fold4

'Total_Price_mean', 'Total_Price Expanding Mean'

03 모델링

(2) ElasticNet & Best Hyper parameter

- Sklearn의 SGDClassifier 이용
- Penalty를 elasticnet으로 설정
- ElasticNet: L1규제와 L2규제를 결합한 회귀
- Alpha로 정규화(규제) 설정
- L1_ratio는 L1규제 비율

Parameter	Value
alpha	0.05
L1_ratio	0.2
class_weight	{0:1, 1:4}
max_iter	1000
loss	Log_loss

03 모델링

(3) XGB & 변수선택

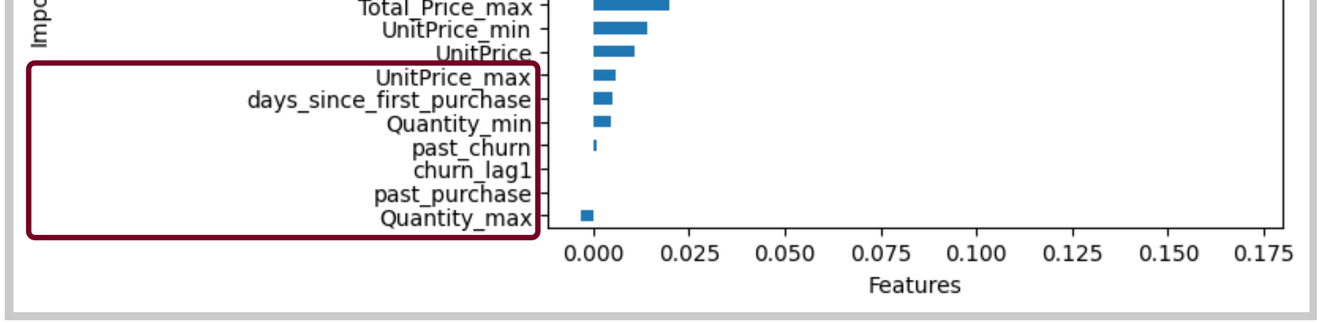
Timeseries cv를 활용한 변수선택

```
--- 2011-08-02 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-02-03 00:00:00
valid 시작 시점 : 2011-04-05 00:00:00
valid 마지막 시점 : 2011-05-04 00:00:00


--- 2011-09-01 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-03-04 00:00:00
valid 시작 시점 : 2011-05-05 00:00:00
valid 마지막 시점 : 2011-06-03 00:00:00


--- 2011-10-01 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-04-04 00:00:00
valid 시작 시점 : 2011-06-05 00:00:00
valid 마지막 시점 : 2011-07-03 00:00:00


--- 2011-10-31 fold----------------------
train 시작 시점 : 2010-12-01 00:00:00
train 마지막 시점 : 2011-05-04 00:00:00
valid 시작 시점 : 2011-07-04 00:00:00
valid 마지막 시점 : 2011-08-02 00:00:00
```



Permutation importance가 0.001 이하이면 변수 drop

03 모델링

(3) XGB & Best Hyper parameter

XGBoost

- 기존 Gradient Tree Boosting 알고리즘에 과적합 방지를 위한 기법이 추가된 지도 학습 알고리즘
- GBM과 다르게 병렬 처리가 가능해 분류 속도가 빠름

Parameter	Value
colsample_bytree	6210258338123061
learning_rate	0.011731983074025 94
max_depth	6
n_estimators	180
scale_pos_weight	8.305813106322233
subsample	0.16886950954669824

04 앙상블 모델링

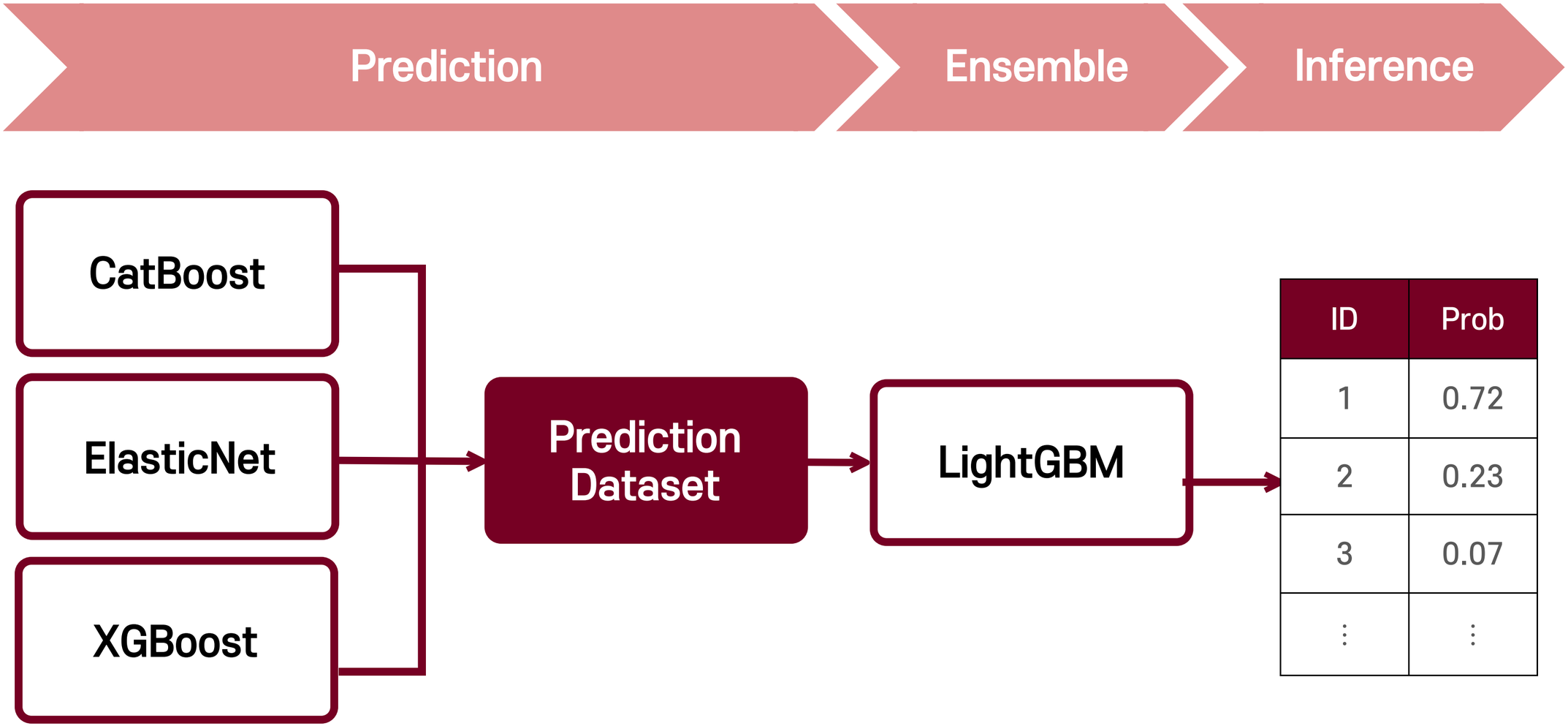
LGBM & Tuning Parameter space

- GBM 기반 알고리즘으로 예측에 실패한 부분에 가중치를 더하면서 오차를 보완하는 식으로 순차적으로 트리를 생성
- 비교적 빠르게 정확한 예측을 수행

→ 앙상블 모델로 활용

Parameter	Value
learning_rate	hp.loguniform('learning_rate', np.log(0.01), np.log(0.2))
max_depth	hp.choice('max_depth',np.arange(1, 30, 1, dtype=int))
num_leaves	hp.choice('num_leaves',np.arange(10, 300, 10, dtype=int))
min_child_weight	hp.uniform('colsample_bytree',0.4,1)
scale_pos_weight	hp.choice('scale_pos_weight',np.arange(1, 10, 1, dtype=int))
lambda_l1	hp.uniform('lambda_l1', 0, 1)
colsample_bytree	hp.uniform('colsample_bytree',0.4,1)

05 파이프라인


Prediction
Ensemble
Inference
CatBoost
ElasticNet
XGBoost
Prediction Dataset
LightGBM
ID
Prob
1
0.72
2
0.23
3
0.07
⋮
⋮

05 파이프라인

```python
# 전처리 파이프라인
def preprocess_pipeline(self):
  preprocess = Pipeline([
      ('drop_outlier', DropOutlier()),
      ('feature_engineering', FeatureEngineering())])
  preprocessed_data = preprocess.fit_transform(df)
  self.output_data = preprocessed_data
  dropcolumn = Pipeline([
      ('drop_feature', DropFeature())
  ])
  preprocessed_data = dropcolumn.fit_transform(preprocessed_data)
  self.preprocessed_data = preprocessed_data
# 1. model train/tuning/valid/test용 dataset split
# 2. ensemble용 dataset 구축
def make_dataset_pipeline(self):
  dataset = Pipeline([
      ('train_test_valid_split', self.make_train_test_pred(self.preprocessed_data)),
      ('ensemble_dataset', self.model_pipeline())])

def model_pipeline(self):
  cat = CatPipeline(now_timestamp=self.now_timestamp ,churn_day=90, test_size=30, pred_s
  cat.fit(df)
  ela = ElaPipeline(now_timestamp=self.now_timestamp ,churn_day=90, test_size=30, pred_s
  ela.fit(df)
  xgb = XGBPipeline(now_timestamp=self.now_timestamp ,churn_day=90, test_size=30, pred_s
  xgb.fit(df)
  self.es_tuning = pd.DataFrame({'cat':cat.test_pred[:,1],
                                 'ela':ela.train_pred[:,1],
                                 'xgb':xgb.test_pred[:,1],
                                 'Churn':self.test['Churn']})
  self.es_final = pd.DataFrame({'cat':cat.final_test_pred[:,1],
                                'ela':ela.train_pred_r[:,1],
                                'xgb':xgb.final_test_pred[:,1]})
```

전처리

데이터셋

앙상블모델

전처리 파이프라인

1. 이상치 제거
2. 피처 엔지니어링

데이터셋 구축 파이프라인

: 시계열 데이터의 특성을 고려하여 데이터셋 분리
(예측 및 검증 세트보다 훈련 세트가 앞선 시점)

1. train, valid, test dataset 구축
2. Ensemble dataset 구축

앙상블 모델 파이프라인

1. 3개의 모델의 별도 파이프라인 클래스를 호출
2. 각 모델에서 데이터 훈련 후 예측값 생성
3. 예측값으로 앙상블용 데이터셋 생성

05 파이프라인

모델세팅

```
1 es = pipeline.EnsemblePipeline(now_timestamp='2011-11-30', space=space, churn_da
```

숨겨진 출력 표시

```
1 es.fit()
```

```
100%|██████████| 200/200 [00:29<00:00,  6.68trial/s, best loss: -0.6714285714285714
```

```
1 es.valid()
```

```
Ensemble tuning score Accuracy: 0.7198
Ensemble tuning score AUC: 0.7730
Ensemble tuning score F1-score: 0.6154
Ensemble tuning score Precision: 0.4727
```

```
1 es.predict()
```

모델갱신

```
1 new = pipeline.EnsemblePipeline(now_timestamp='2011-12-07', space=space, churn_d
```

숨겨진 출력 표시

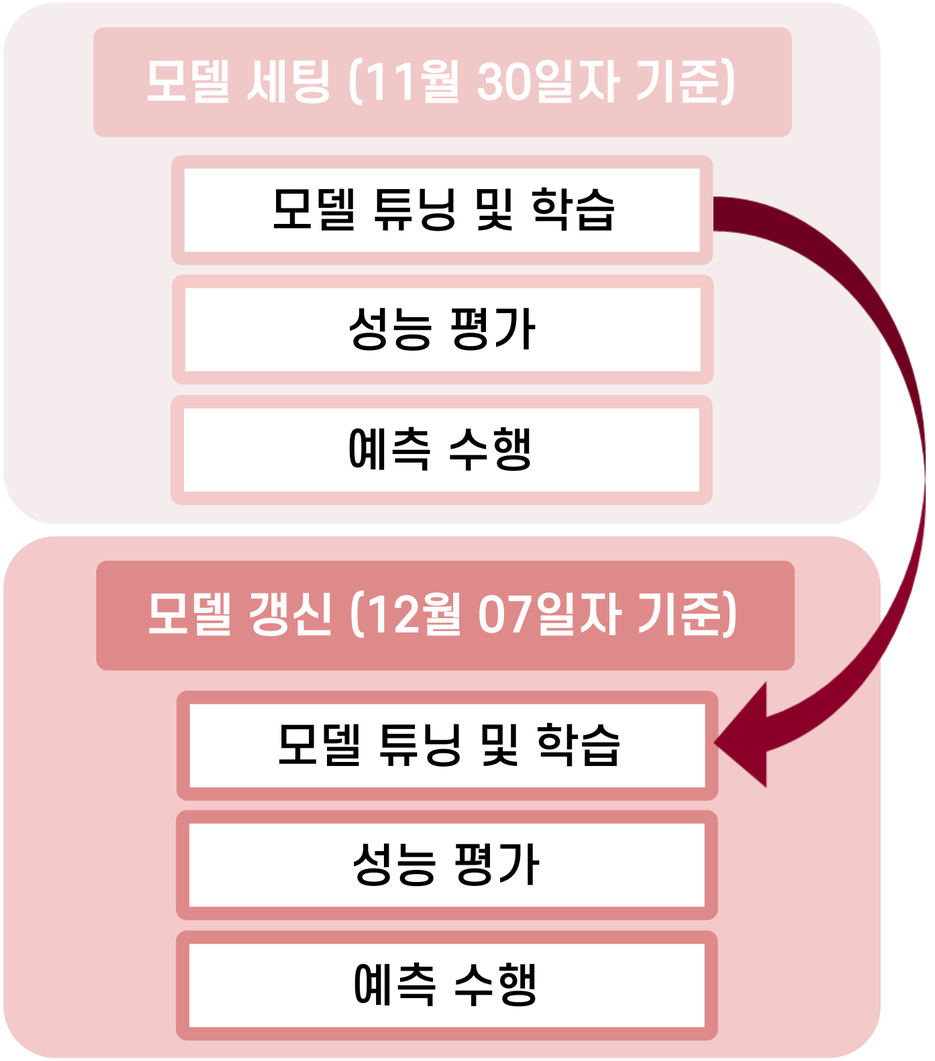
```
1 # 11월 30일자 튜닝된 하이퍼파라미터 이전
2 new.hyperparameters = es.hyperparameters.copy()
```

```
1 new.fit() #재학습
```

```
1 new.valid()
```

```
Ensemble tuning score Accuracy: 0.7089
Ensemble tuning score AUC: 0.7663
Ensemble tuning score F1-score: 0.6188
Ensemble tuning score Precision: 0.4746
```

```
1 new.predict()
```



04. 결과

01 이탈 예측 결과

앙상블 튜닝 최종 스코어

Metric	Score
Accuracy	0.7198
AUC	0.7730
F1-score	0.6154
Precision	0.4727

앙상블 결과 12/01부터 12/07까지
756명의 기존 고객 중 244명 고객 이탈 예측

02 Dash를 이용한 시각화

2011/12/01 ~ 2011/12/07 고객 이탈 예측

Option1 이탈로 예측된 고객 출력, 이탈 확률이 70% 이상인 고객은 핑크색으로 강조

2011-12-01 ~ 2011-12-07 이탈 고객

InvoiceDate	CustomerID	Churn probability
2011-12-01	13890	0.5414049361591712
2011-12-01	15862	0.6877855953150774
2011-12-01	14047	0.5672517443061015
2011-12-01	17566	0.6825216118360593
2011-12-01	14146	0.6123856639501442
2011-12-01	15948	0.6847184089269434
2011-12-01	15951	0.7128326933803708
2011-12-01	16119	0.6391862280181837
2011-12-01	13323	0.6050380113956353
2011-12-01	17768	0.6195004025035226
2011-12-01	13509	0.580234402203074
2011-12-01	16789	0.7554768345376753
2011-12-01	14766	0.5488370556588427
2011-12-01	14804	0.7120718177327022
2011-12-01	17115	0.537597222398707

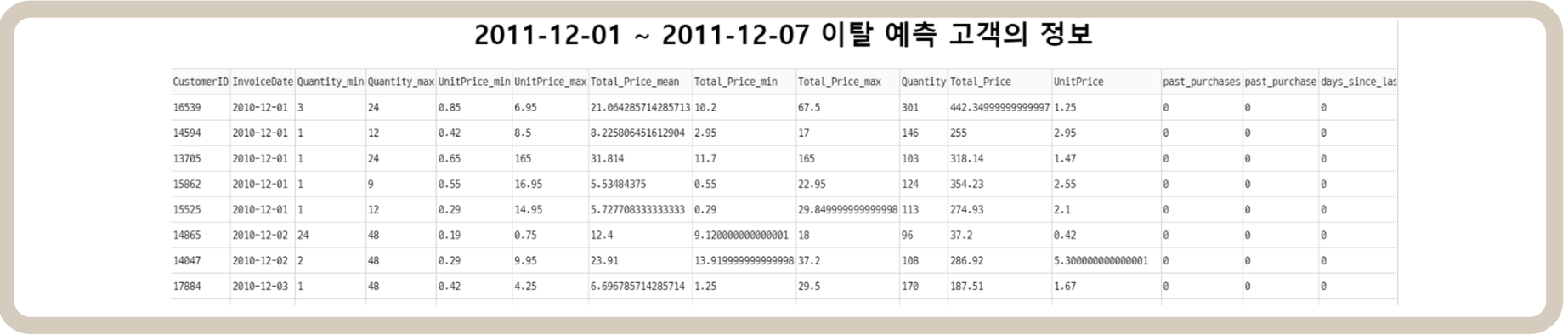
02 Dash를 이용한 시각화

2011/12/01 ~ 2011/12/07 고객 이탈 예측

Option2 2011/12/01 ~ 2011/12/07 이탈로 예측된 고객의 과거 정보 검색 가능

CustomerID	InvoiceDate	Total_Price	past_churns
12782	2010-12-17	463.5	0
12782	2011-01-26	683.9300000000001	0
12782	2011-03-31	415.52000000000004	0
12782	2011-07-13	515.72	1
12782	2011-12-05	252.25	1

Option3 고객 ID를 검색 칸에 입력하면 고객의 과거 정보를 보여줌



CustomerID	InvoiceDate	Quantity_min	Quantity_max	UnitPrice_min	UnitPrice_max	Total_Price_mean	Total_Price_min	Total_Price_max	Quantity	Total_Price	UnitPrice	past_purchases	past_purchase	days_since_las
16539	2010-12-01	3	24	0.85	6.95	21.064285714285713	10.2	67.5	301	442.34999999999997	1.25	0	0	0
14594	2010-12-01	1	12	0.42	8.5	8.225806451612904	2.95	17	146	255	2.95	0	0	0
13705	2010-12-01	1	24	0.65	165	31.814	11.7	165	103	318.14	1.47	0	0	0
15862	2010-12-01	1	9	0.55	16.95	5.53484375	0.55	22.95	124	354.23	2.55	0	0	0
15525	2010-12-01	1	12	0.29	14.95	5.727708333333333	0.29	29.849999999999998	113	274.93	2.1	0	0	0
14865	2010-12-02	24	48	0.19	0.75	12.4	9.120000000000001	18	96	37.2	0.42	0	0	0
14047	2010-12-02	2	48	0.29	9.95	23.91	13.919999999999998	37.2	108	286.92	5.300000000000001	0	0	0
17884	2010-12-03	1	48	0.42	4.25	6.696785714285714	1.25	29.5	170	187.51	1.67	0	0	0

03 기대 효과 및 한계점

기대효과

① 실시간 고객 이탈을 예측

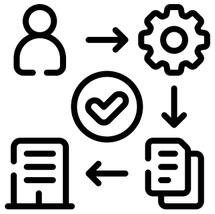
↓

쿠폰 지급 등 맞춤형 혜택을
제공함으로써 빠른 대처 가능

② 예측 자동화

↓

시간과 비용 절약



③ 실시간으로 수집된 데이터

↓

모델에 적용하고,
적합한 모델 훈련 및 학습,
평가 가능

03 기대 효과 및 한계점

한계점

데이터의 한계

- 로그 데이터 부족 및 데이터 수집 기간 짧음 → 성능 한계
- 데이터 규모가 늘어난다면 더 좋은 성능이 기대됨

감사합니다