번개장터

2020/12/17
Anaelle

이전 찜,구매 기록 기반
찜유저 구매 확률 예측

번개장터 데이터팀 인턴 프로젝트

번개장터

목차

01
서론
- 프로젝트 기본 설명

02
본론
- 분석 과정 세부 설명

03
결론
- 모델 적용 결과
- 발전 방향
- 활용 방안

01

서론

프로젝트 기본 설명

번개장터

“

특정 상품을 찜한 유저들의 구매 확률은 어떠할까?

- 유저의 이전 찜, 구매 기록과 판매자 정보에 기반한, *"현재 찜 상품 구매확률 예측 모델"*
- 분석 대상: 20201126~20201205 찜 유저 전체

찜유저 구매 확률 예측

번개장터

02

본론

분석과정 세부 설명

분석 과정 세부 설명

분석 절차 시간순 정리

01 데이터 수집 및 전처리

- 데이터셋 구성 방법 탐구
- Null 값 있는 행은 삭제

02 데이터 불균형 문제 해결

- 구매 데이터 수 부족
-Under/Oversampling 방법론 탐구

03 스케일링 작업

- 변동 정도에 따른 영향 감소

04 모델 성능 비교 및 선정

-정확도, 정밀도, 신뢰도 기준으로 선정

데이터 수집 및 전처리

데이터셋 구성 방법 탐구

- 활용 테이블

테이블 명	활용 컬럼
bun_log_db.api_event_type_wishlist	상품 찜정보(user_id, content_id, content_owner, 날짜)
service1_quicket.product_info	name, price, category_id, flag_taekpo, flag_exchg
service1_quicket.user_extra_info	age, sex, comment_cnt
service1_quicket.product_ext	pfavcnt
service1_quicket.bunjang_promise	번장프로미스 -> 구매여부, 총 구매 횟수, 구매 금액
service1_quicket.order_mast	번개페이->구매여부, 총 구매 횟수, 구매 금액

데이터 수집 및 전처리

데이터셋 구성 방법 탐구

- 데이터셋 예시

	구id	구성별	구나이	3개월간구매횟수	평균금액	찜시각	카테고리	상품가격	찜수	택포	교환	판매자후기수	판성별	판나이	구매여부	agediff
13	719	2	20	18	35012	20201204	400020	25000	5	0	0	4	2	25	0	5
14	719	2	20	18	35012	20201202	320150	10000	14	0	0	23	1	25	0	5
15	719	2	20	18	35012	20201204	700100	70000	2	0	0	0	2	10	0	10

```
In [16]: jab.shape
Out[16]: (144687, 16)

In [17]: # jab['구매여부'].hist()
         print(jab['구매여부'].value_counts())

0    132006
1     12681
Name: 구매여부, dtype: int64
```

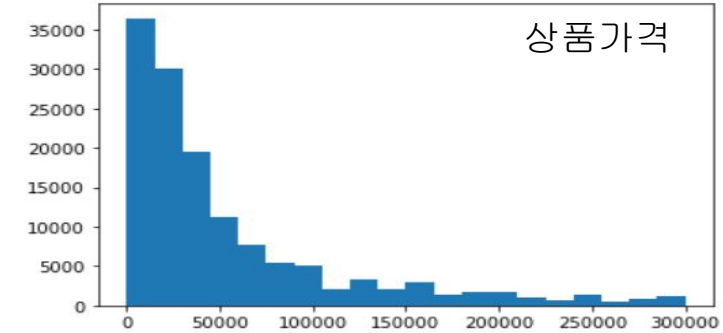
11월 27일~12월 5일까지 발생한 찜 데이터(799,958건) 중

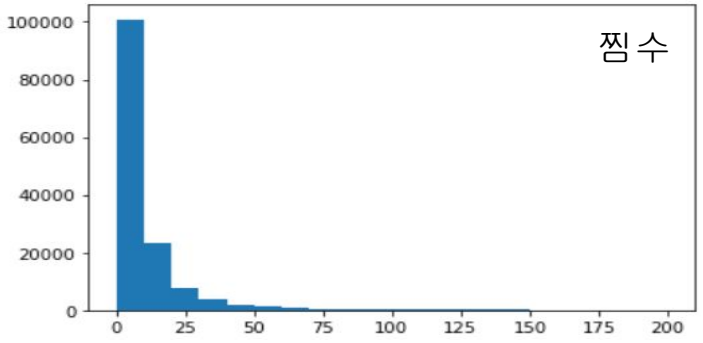
"찜에서 구매로 이어진 횟수가 2번 이상인 user"

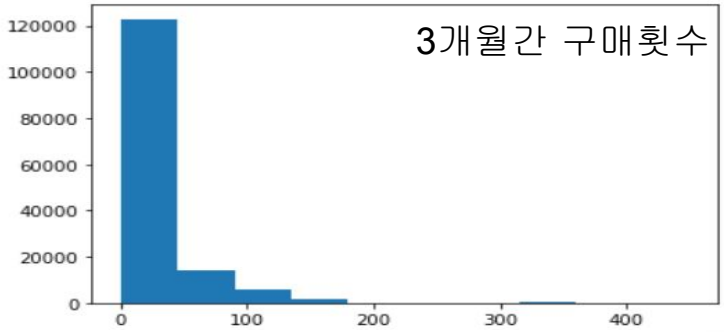
의 데이터(**144,687건**)만 추출

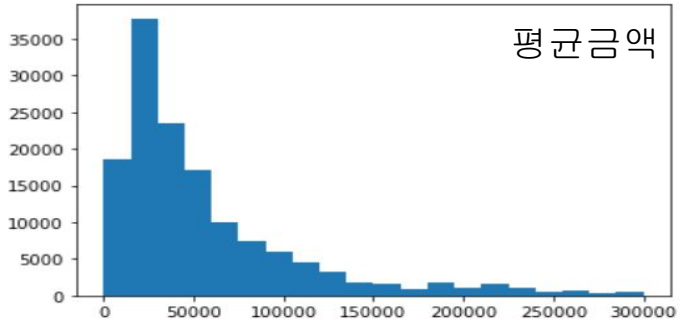
데이터 수집 및 전처리

데이터 분포 확인 및 전처리



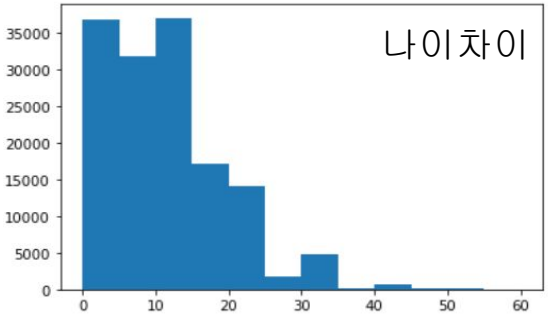






데이터 수집 및 전처리

데이터 분포 확인 및 전처리




```
print(jab['카테고리'].value_counts())
```

```
400010    8846
320080    7710
400040    6707
310120    5916
310090    5460
```

Pandas - get_dummies 함수를 이용해 One-Hot encoding 진행

ID	과일
1	사과
2	바나나
3	체리

One-Hot Encoding

ID	사과	바나나	체리
1	1	0	0
2	0	1	0
3	0	0	1

LabelEncoder

ID	과일
1	0
2	1
3	2

https://mizykk.tistory.com/13

데이터 불균형 문제 해결

Oversampling 방법을 중심으로...

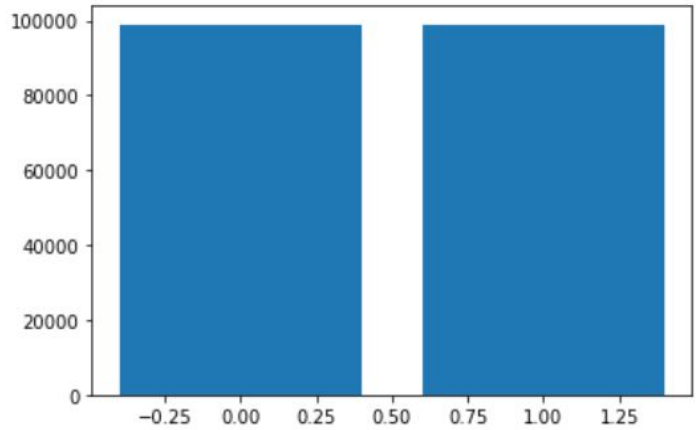
샘플링 종류	특징 및 장단점
Random Under sampling	-특징: 무작위로 다수 클래스의 예제를 제거 -장점: 속도 개선 -단점: 중요정보 누락
Random Over sampling	-특징: 무작위로 소수 클래스의 인스턴스 수를 늘림 -장점: 정보손실x, 성능향상 -단점: 과적합 가능성
SMOTE	*-특징: 과적합 피하기 위한 기술. 소수 클래스로부터 추출. 새로운 합성 유사 사례 생성* *-장점: 과적합 완화. 데이터 손실 없음* *-단점: 클래스의 겹침이나 추가적 노이즈 발생. (이를 해결하기 위한 msmote존재)*

데이터 불균형 문제 해결

Oversampling 방법을 중심으로...

In [26]:
```
from imblearn.over_sampling import SMOTE
X_train_over,y_train_over = SMOTE(random_state=0).fit_resample(train_features,train_labels)
count_and_plot(y_train_over)
```

```
Class=0, n=99008 (50.000%)
Class=1, n=99008 (50.000%)
```



스케일링

변수별 변동 정도 차이에서 오는 영향 감소

스케일러 종류	특징 및 장단점
StandardScaler	-특징: 기본 스케일링. 평균과 표준편차 사용. 이상치에 민감
MinmaxScaler	-특징: 최대/최솟값이 각각 1,0이 되도록 스케일링. 이상치에 민감
RobustScaler	*-특징: 중앙값과 IQR(interquartile range) 사용. 이상치 영향 최소화*

"판매가격 등은 이상치 존재 가능성이 있기 때문에 여러 스케일링 방법 중 Robust Scaler를 사용"

모델 성능 비교 및 선정

정확도, 정밀도 기준으로.

분석기법 종류	특징 및 장단점
Logistic regression	-특징: 종속변수가 범주형인 경우 적용. 새로운 독립변수 값이 주어질 때 종속변수의 각 범주에 속할 확률이 얼마인지 추정해 분류.
Decision Tree	-특징: 데이터들이 가진 속성들로부터 분할 기준 속성을 판별하고, 분류함수 통해 트리 형태로 모델링하는 분류예측모델 -장점: 해석의 용이성, 비모수적 모형.대용량 데이터에서도 빠름. 정확도 높음. 스케일링 등의 전처리가 크게 영향을 안 미침 -단점: Training Data에 의존하여 과적합 가능성.
XGBoost	**-특징: gradient boost 기반 -장점: GBM보다 정확도 높고 빠름. 유연성이 좋다. 교차검증-조기종료 가능. 변수의 정규화 이용-> 과적합 방지 -단점: 다른 알고리즘에 비해서는 느림**
RandomForest	**-특징: 의사결정나무는 분산이 크기 때문에 많은 무작위성을 주어 약한 학습기 생성 후 이를 선형결합 해 최종 학습기 만드는 것. -장점: 과적합 완화. 새로운 데이터에 대해 일반화 잘 됨. -단점: 속도 저하(메모리 사용이 큼)**

모델 성능 비교 및 선정

정확도, 정밀도 기준으로.

In [999]:
```
from xgboost import XGBClassifier

model = XGBClassifier(n_estimators = 400, learning_rate = 0.1, max_depth = 2)
model.fit(X_train_over, y_train_over)
w_preds = model.predict(test_features)
```

n [1000]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
from sklearn.metrics import confusion_matrix, f1_score, roc_auc_score
y_pred=model.predict(test_features)
def get_jab_eval(test_labels,y_pred):
    confusion = confusion_matrix(test_labels, y_pred)
    accuracy = accuracy_score(test_labels, y_pred)
    precision = precision_score(test_labels, y_pred)
    recall = recall_score(test_labels, y_pred)
    F1 = f1_score(test_labels, y_pred)
    AUC = roc_auc_score(test_labels, y_pred)
    print('오차행렬:₩n', confusion)
    print('₩n정확도: {:.4f}'.format(accuracy))
    print('정밀도: {:.4f}'.format(precision))
    print('재현율: {:.4f}'.format(recall))
    print('F1: {:.4f}'.format(F1))
    print('AUC: {:.4f}'.format(AUC))

get_jab_eval(test_labels, w_preds)
```

In [986]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=400,      # Number of trees
                               max_features=3,      # Num features considered
                               oob_score=True)
model
```

번개장터

모델 성능 비교 및 선정

정확도, 정밀도 기준으로.

분석기법 종류	Accuracy	Precision	
		0(구매x)	1(구매 o)
Logistic regression	0.57	0.93	0.12
Decision Tree	0.85	0.93	0.24
XGBoost	0.91	0.91	0.60
RandomForest	***0.92***	***0.93***	***0.71***

sklearn.metrics module에서 classification-report 사용

*Accuracy: 실제 분류 범주를 정확하게 예측한비율
*Precision: 참으로 예측한 것 중 실제로 참인 비율

03

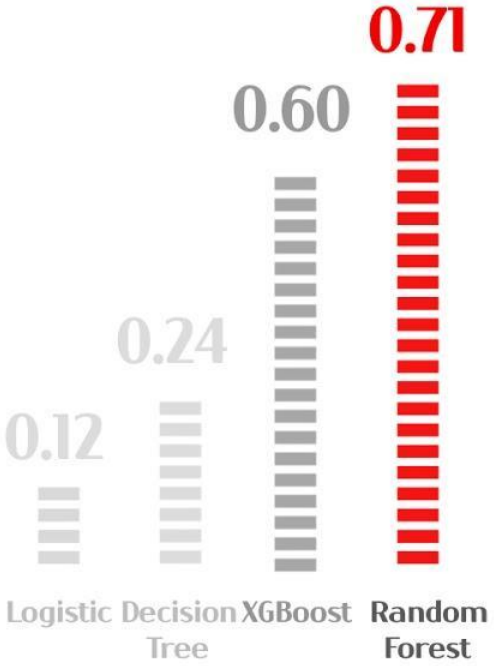
결론

모델적용결과, 한계점 및 발전방향

최적 분석기법 선정

분석기법 별 Accuracy, Precisin 비교

분석기법 종류	Accuracy	Precision	
		0(구매x)	1(구매 o)
RandomForest	*0.92*	*0.93*	*0.71*



*Accuracy: 실제 분류 범주를 정확하게 예측한비율
*Precision: 참으로 예측한 것 중 실제로 참인 비율

한계점 및 발전방향

01 데이터 불균형 정도 컸음

번장 프로미스, 번개페이 기록만을 가지고 구매 여부를 판단해, 실제 구매가 일어난 것 보다 적게 기록됐을 것. 번개톡 기록뿐만 아니라 실제 구매 여부를 더 알 수 있다면 구매(1) row 수가 늘어날 것.

02 주관적인 기간 설정

며칠간의 찜을 통한 탐색 후에 구매하는 유저들을 고려하기 위해 찜 기간과 구매 기간에 차이를 두었는데 이 과정이 주관적으로 진행됐음. 추가 통계자료를 활용하면 더 정확할 것이다.

03 샘플링 방법 대안 탐구 필요

현재 smote 이외에도 효과적인 샘플링 방법이 존재.(adasyn 등). 과적합은 피하면서 조금 더 효율적인 오버샘플링 방법을 찾아야 한다.

번개장터+

활용 방안

번개장터 앱에 도입 시...

○ 예상 활용 모습

좌측 이미지와 같이 구매 확률이 상대적으로 높은 상점에 표시

○ 기대효과

구매확률 상위 상점 집중 공략으로 제품 판매에 투자되는 시간 및 피로도 감소 -> 번개장터 긍정적 인식 -> 활용률 높아짐

발표를 들어주셔서

감사합니다 :)