

로지스틱 회귀(LogisticRegression)를 사용한 구매 예측 모델

- 목적
- 로지스틱 회귀란?
 - 선형 회귀와의 차이점
 - 로지스틱 회귀의 종류
- 사용된 데이터셋 (temp_new2)
- 코드진행
 - 라이브러리 불러오기
 - 데이터 전처리
 - 피쳐 값과 타겟 값 설정 & 데이터셋과 테스트셋 분할
 - 데이터 편향을 억제하기위한 오버샘플링
 - 모델 학습과 정확도 평가
 - 피쳐들의 영향력 시각화
 - 양성 클래스의 확률 시각화
 - 최적의 하이퍼파라미터 확인
 - 분류 보고
- 개선 방향과 보안점

목적

유저 데이터와 쿠폰 데이터를 기반으로 물건을 구매할지 구매하지 않을지 예측하기 위해 로지스틱 회귀 모델을 사용 하여 예측을 시도 하였습니다.

로지스틱 회귀란?

로지스틱 회귀는 사건이 발생할 가능성을 예측하는데 사용하는 모델이며 주로 특정 조건이 주어졌을 때 사건이 일어나는지, 일어나지 않는지 에 대해 다루는 회귀 모델입니다.

선형 회귀와의 차이점

선형 회귀와의 차이점으로는 선형회귀는 확률값이 $-\infty \sim \infty$ 로 뻗어나가는 일직선이지만 로지스틱 회귀는 0~1 사이에서 완만한 S-curve를 그립니다.

로지스틱 회귀의 종류

1. 이진 로지스틱 회귀 : 이진 분류로 결과 값이 YES(1) OR NO(2)로 출력하는 역할을 수행합니다.
2. 다항 로지스틱 회귀 : 여기에는 결과 값에 순서가 없는 3개 이상의 변수가 포함될 수 있습니다. 예를 들어 레스토랑에서 식사하는 사람들이 특정 종류의 음식(채식, 고기 또는 완전채식)을 선호하는지 예측하는 것이 있습니다.
3. 순서 로지스틱 회귀 : 다항 회귀와 마찬가지로 3개 이상의 변수가 있을 수 있습니다. 그러나 측정에는 순서가 있습니다. 예를 들어 1에서 5까지의 척도로 호텔을 평가하는 경우를 들 수 있습니다.

이러한 종류중 이진 로지스틱 회귀 를 사용하여 구매를 할지 하지 않을지 예측하는 모델을 만들었습니다.

사용된 데이터셋 (temp_new2)

Column Name	Description	Type	Length	Decimal	Note
USER_ID_hash	User ID	VARCHAR2	32		
REG_DATE	Registered date	DATE			Sign up date
SEX_ID	Gender	CHAR	1		f = female m = male
AGE	Age	NUMBER	4	0	
TR_PREF_NAME	Residential Prefecture	VARCHAR2	2		[JPN] Not registered if empty
Tr_small_area_name	Small area name of shop location	VARCHAR2	30		[KOR]
Tr_CAPSULE_TEXT	Capsule text	VARCHAR2	20		[KOR]
Tr_GENRE_NAME	Category name	VARCHAR2	50		[KOR]
VIEW_COUPON_ID_hash	Browsing Coupon ID	VARCHAR2	128		
USABLE_DATE_SUM		CHAR			
USABLE_DATE_MON	Is available on Monday	CHAR	1		
USABLE_DATE_TUE	Is available on Tuesday	CHAR	1		
USABLE_DATE_WED	Is available on Wednesday	CHAR	1		
USABLE_DATE_THU	Is available on Thursday	CHAR	1		
USABLE_DATE_FRI	Is available on Friday	CHAR	1		
USABLE_DATE_SAT	Is available on Saturday	CHAR	1		
USABLE_DATE_SUN	Is available on Sunday	CHAR	1		
USABLE_DATE_HOLIDAY	Is available on holiday	CHAR	1		
USABLE_DATE_BEFORE_HOLIDAY	Is available on the day before holiday	CHAR	1		
Male		INTEGER	1		
Female		INTEGER	1		
view_count		INTEGER			
PRICE_RATE	Discount rate	NUMBER	4	0	
DISCOUNT_PRICE	Discount price	NUMBER	10	0	
VALIDFROM	The term of validity starts	DATE			
DISPPERIOD	Sales period (day)	NUMBER	4	0	
DISPFROM	Sales release date	DATE			
DISPEND	Sales end date	DATE			

코드진행

라이브러리 불러오기

```

1 import pandas as pd
2 from sklearn.preprocessing import LabelEncoder
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.linear_model import LogisticRegression
5 from sklearn.model_selection import train_test_split
6 from sklearn.metrics import accuracy_score
7 from imblearn.over_sampling import RandomOverSampler
8 from imblearn.over_sampling import SMOTE
9 import matplotlib.pyplot as plt
10 import seaborn as sns
11 from sklearn.metrics import classification_report
12 from sklearn.model_selection import GridSearchCV

```

데이터 전처리 [↗](#)

```

1 #null값 제거
2 df.dropna(inplace = True)
3
4 #데이터분류
5 COL_DEL=['USER_ID']
6 COL_NUM=['AGE', 'usable_date_sum', 'view_count', 'PRICE_RATE', 'DISCOUNT_PRICE', 'VALIDPERIOD', 'usable_date_mon',
7         'usable_date_tue', 'usable_date_wed', 'usable_date_thu', 'usable_date_fri', 'usable_date_sat', 'usable_date_sun',
8         'usable_date_before_holiday']
9 COL_CAT=['SEX_ID', 'Tr_Pref_Name', 'Tr_small_area_name', 'Translated_capsule_text', 'Translated_genre_name']
10 COL_Y=['PURCHASE_FLG']
11
12 #범주형 변수 라벨 인코딩
13 label_encoder = LabelEncoder()
14 for col in COL_CAT:
15     df[col] = label_encoder.fit_transform(df[col])
16
17 #standard 스케일
18 from sklearn.preprocessing import StandardScaler
19 scaler = StandardScaler()
20 df[COL_NUM] = scaler.fit_transform(df[COL_NUM])
21
22 #중복된 컬럼이나 고유값을 갖는 컬럼들 제외
23 df_dropped = df.drop(['USER_ID_hash', 'REG_DATE', 'VIEW_COUPON_ID_hash', 'dispfrom', 'dispend', 'Male', 'Female'], axis=1)

```



null값을 제거 하고

데이터를 수치형과 범주형으로 구분하고 타겟값을 지정

범주형 데이터는 라벨인코딩을 진행 하고 수치형 데이터는 standard 스케일러로 스케일링 하였고

그 후 중복된 컬럼이나 고유값을 갖는 컬럼들은 피쳐값 에서 제외 하였습니다.

피쳐 값과 타겟 값 설정 & 데이터세트와 테스트세트 분할 [↗](#)

```

1 X = df_dropped.drop('PURCHASE_FLG',axis = 1)
2 y = df_dropped['PURCHASE_FLG']
3
4 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state = 42)

```

데이터 편향을 억제하기위한 오버샘플링 [↗](#)

```

1 oversampler = RandomOverSampler(random_state = 42)
2
3 X_train_oversampled, y_train_oversampled = oversampler.fit_resample(X_train, y_train)

```

모델 학습과 정확도 평가 [🔗](#)

```

1 model = LogisticRegression(penalty = 'l1', C=0.5, solver = 'liblinear')
2 model.fit(X_train_oversampled, y_train_oversampled)
3
4 y_pred = model.predict(X_test)
5 accuracy = accuracy_score(y_test, y_pred)
6
7 print(accuracy)

```

0.610141929811087

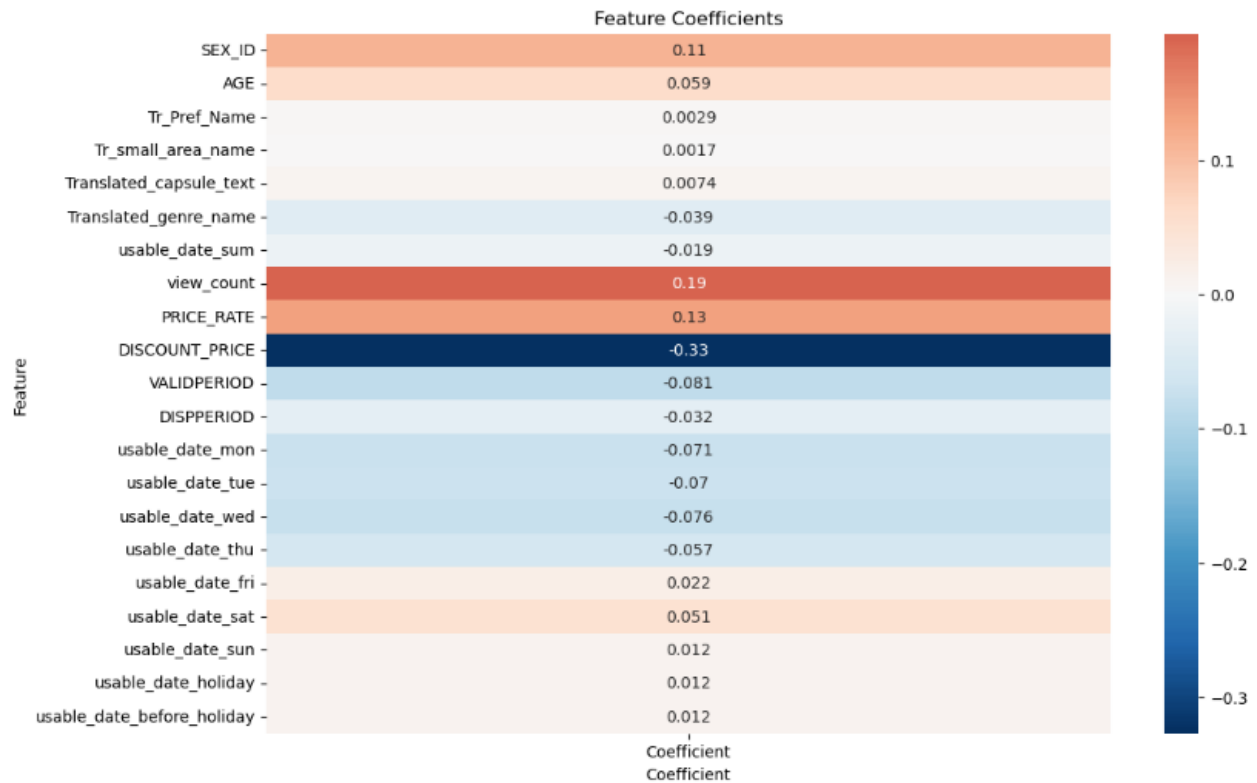
i SMOTE 와 비교했을때 RandomOverSampler의 점수가 1점이나마 더 높아서 RandomOverSampler를 사용하였습니다.

피쳐들의 영향력 시각화 [🔗](#)

```

1 # 로지스틱 회귀 모델에서 계수(coefficient) 얻기
2 coefficients = model.coef_[0]
3
4 # 피쳐 이름과 계수를 매칭하는 딕셔너리 생성
5 feature_coefficients = dict(zip(X.columns, coefficients))
6
7 # 계수를 데이터프레임으로 변환하여 히트맵에 사용하기 위해 재구성
8 df_coefficients = pd.DataFrame.from_dict(feature_coefficients, orient='index', columns=['Coefficient'])
9
10 # 히트맵 생성
11 plt.figure(figsize=(12, 8))
12 sns.heatmap(df_coefficients, annot=True, cmap='RdBu_r', center=0)
13 plt.title('Feature Coefficients')
14 plt.xlabel('Coefficient')
15 plt.ylabel('Feature')
16 plt.show()

```

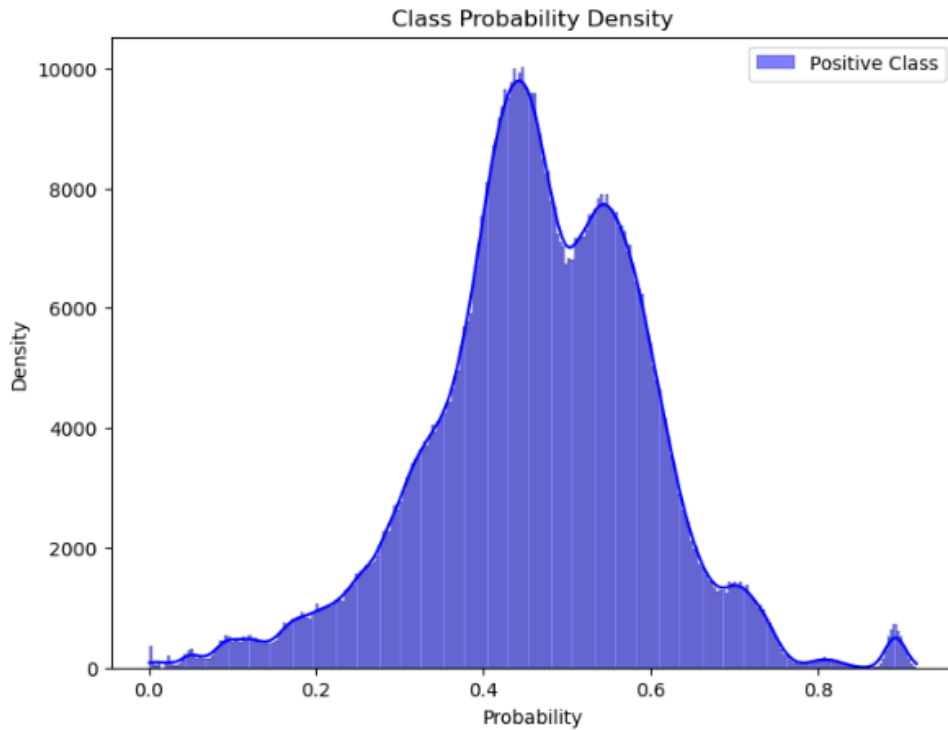


양성 클래스의 확률 시각화 [🔗](#)

```

1 # 테스트 데이터에 대한 예측 확률 얻기 +
2 y_prob = model.predict_proba(X_test)
3
4 # 클래스별 확률 값을 추출
5 class_probabilities = y_prob[:, 1] # 두 번째 클래스(양성 클래스)의 확률 추출
6
7 # 양성 클래스의 밀도 그래프 생성
8 plt.figure(figsize=(8, 6))
9 sns.histplot(class_probabilities, kde=True, color='blue', label='Positive Class')
10 plt.title('Class Probability Density')
11 plt.xlabel('Probability')
12 plt.ylabel('Density')
13 plt.legend()
14 plt.show()

```



최적의 하이퍼파라미터 확인 [🔗](#)

```

1 # 하이퍼파라미터 후보들 정의
2 param_grid = {
3     'penalty': ['l1', 'l2'],
4     'C': [0.1, 0.5, 1.0],
5     'solver': ['liblinear', 'saga']
6 }
7
8 # 그리드 서치 객체 생성
9 grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5)
10
11 # 그리드 서치 수행
12 grid_search.fit(X_train, y_train)
13
14 # 최적의 하이퍼파라미터와 정확도 출력
15 print('최적 하이퍼파라미터:', grid_search.best_params_)
16 print('최고 정확도:', grid_search.best_score_)

```

최적 하이퍼파라미터: {'C': 1.0, 'penalty': 'l2', 'solver': 'saga'}
 최고 정확도: 0.6124468692975503

⚠ 최적의 하이퍼 파라미터를 찾았지만 정확도가 0.2점 정도 밖에 향상되지 않았습니다.

분류 보고 [🔗](#)

```


1 classification_report = classification_report(y_test, y_pred)
2 print('분류 보고서:\n', classification_report)

```

분류 보고서:

	precision	recall	f1-score	support
0	0.97	0.61	0.75	718236
1	0.07	0.61	0.13	36926
accuracy			0.61	755162
macro avg	0.52	0.61	0.44	755162
weighted avg	0.92	0.61	0.72	755162

 precision 스코어와 F1-score가 구매할 확률에서 매우 낮게 나타나기에 예측 성능이 정확도 보다도 낮은것으로 판단됩니다.

 Precision : 정밀도 - 모델이 해당 클래스로 예측한 샘플 중 실제로 해당 클래스인 비율을 나타냅니다.

Recall : - 재현율 - 실제 클래스를 대상으로 예측과 실제값이 일치한 비율

F1-score : Precision과 Recall의 조화 평균으로, 클래스별로 정확성과 재현율을 종합적으로 평가하는 지표입니다

개선 방향과 보안점

1. 피쳐 엔지니어링 - 유용한 피쳐 선택, 이상치 처리, 피쳐변환, 특성공학을 사용하여 모델에 더 많은 정보를 제공하는 방법 찾아보기
 - ↳ 컬럼별 연관성을 찾아 group by 하여 그 데이터로 모델링 돌려보기
 - ↳ ex) 구매량, 쿠폰발급량, 도시등 상위 3~5위 필터링 하여 group by 한 후 그 데이터로 그래프를 그려보고 인사이트를 찾아 모델링 해보기
2. 쿠폰ID, 지역 등 특정 항목으로 치우쳐진 데이터 조정
3. 오버 샘플링과 가중치부여를 통하여 예측성능 향상시켜보기
 - ↳ 현재는 오버 샘플링만 진행되었는데 [분류보고](#) 를 확인 해보면 1에 대한 precision, F1 score 가 낮다 라는걸 확인 할 수 있었기에 1에 대한 가중치 부여를 다르게 하여 좀 더 유의미한 결과를 도출해보기