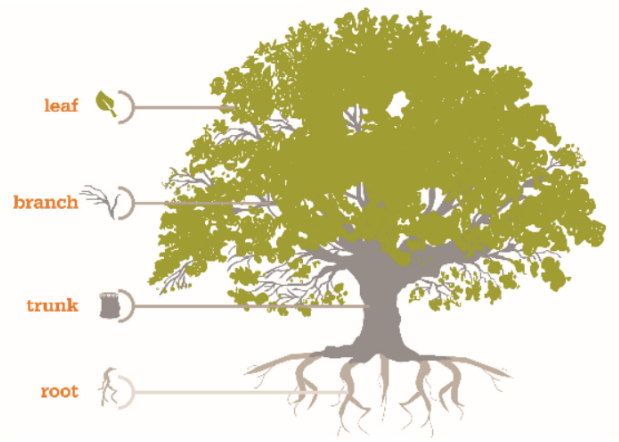
RandomForestClassifier



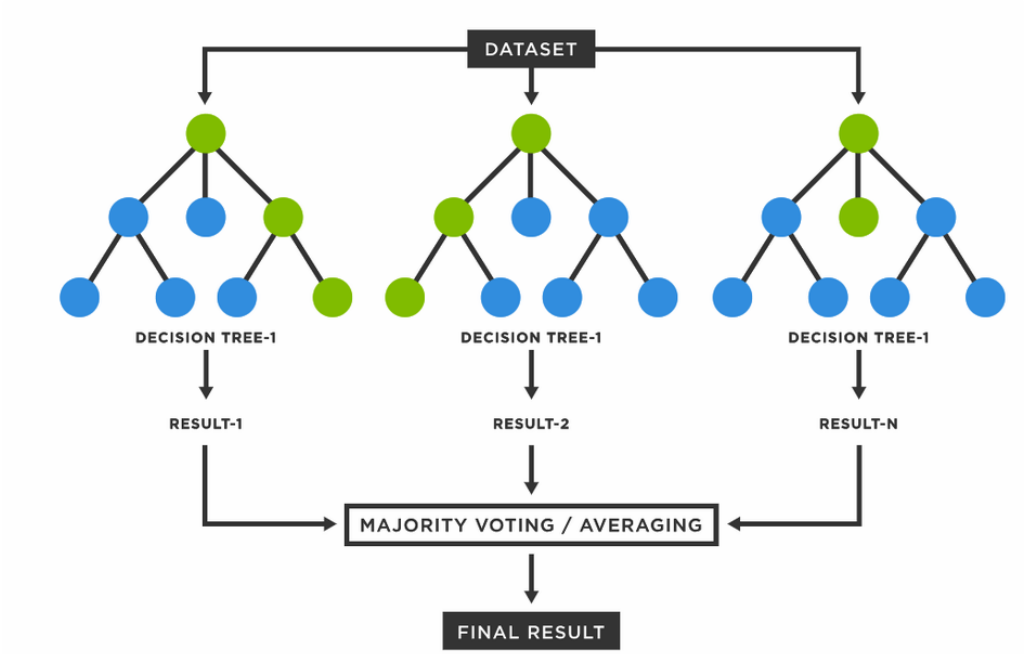
모델 목표

사용 테이블 정보(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			

랜덤 포레스트(RandomForest)



- 랜덤 포레스트의 이점
  - 상대적 중요성을 측정하기 쉬움
  - 다재다능 - 분류 및 회귀 작업 모두 사용 할 수 있음 + 변환이나 재조정 없이 이 진 및 숫자 기능과 범주형 기능을 쉽게 처리할 수 있습니다.
  - 과적합 확률이 줄어듬 - 랜덤 포레스트는 하위 집합에서 다양한 크기의 트리를 만들고 결과를 결합하여 과적합을 방지합니다.
  - 높은 정확도
  - 빠른 학습 속도
- 랜덤 포레스트의 문제점
  - 느린 결과 - 많은 트리를 구축하기 때문에 예측의 정교함과 정확도가 높아진다. 그러나 수백 또는 수천 개 트리를 만들기 때문에 프로세스 속도가 느려진다(실시간 예측에는 비효율적).
  - 낮은 해석 가능성 -모델들은 설명할 수 없기 때문에 특정 결정에 도달한 방법 또는 이유를 이해하기 어렵다.
- 랜덤 포레스트의 대안
  - 신경망(NN)
  - 익스트림 그레이디언트 부스팅(XGBoost)

파이썬 코드 진행

사용 라이브러리

```
from google.colab import drive
import csv
import random
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,precision_score,recall_score
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
```

원본 데이터 TRAIN 과 TEST 파일로 나누기

```
def split_csv(input_file, output_file1, output_file2, split_ratio):
    with open(input_file, 'r') as file:
        reader = csv.reader(file)
        header = next(reader)  # 헤더를 읽어옵니다.

        # 출력 파일들을 열고 헤더를 씁니다.
        with open(output_file1, 'w', newline='') as file1, open(output_file2, 'w', newline='') as file2:
            writer1 = csv.writer(file1)
            writer1.writerow(header)
            writer2 = csv.writer(file2)
            writer2.writerow(header)

            data = list(reader)

            split_index = int(len(data) * split_ratio)  # 분할 기준 인덱스 계산

            random.shuffle(data)  # 데이터를 무작위로 섞습니다.

            for i, row in enumerate(data):
                if i < split_index:  # 80%를 첫 번째 파일에 저장
                    writer1.writerow(row)
                else:  # 나머지 20%를 두 번째 파일에 저장
                    writer2.writerow(row)

    print("CSV 파일이 분할되었습니다.")

# CSV 파일 분할을 위한 예시 호출
split_csv('/content/drive/MyDrive/temp_new2.csv', '/content/drive/MyDrive/temp_train3.csv', '/content/drive/MyDr
```

파일 불러오기

```
#과적합 방지를 위한 중복 제거 함께 실행
train = pd.read_csv('/content/drive/MyDrive/temp_train3.csv')
train = train.drop_duplicates()

test= pd.read_csv('/content/drive/MyDrive/temp_test3.csv')
test = test.drop_duplicates()
```

+ 변수 사이의 상관관계 0.8 이상인 것들 제거하기

```
# # 상관관계 행렬 계산
# corr_matrix = train.corr().abs()

# # 상관계수가 높은 변수 제거
# threshold = 0.8  # 상관계수 임계값 설정

# # 다중 공선성이 있는 변수 선택 및 제거
# upper_triangle = np.triu(np.ones(corr_matrix.shape), k=1).astype(bool)
# high_correlation_cols = [column for column in corr_matrix.columns if any(corr_matrix.loc[upper_triangle, colum
# train2 = train.drop(high_correlation_cols, axis=1)

# corr_matrix2 = test.corr().abs()

# # 상관계수가 높은 변수 제거
# threshold = 0.8  # 상관계수 임계값 설정

# # 다중 공선성이 있는 변수 선택 및 제거
# upper_triangle = np.triu(np.ones(corr_matrix2.shape), k=1).astype(bool)
# high_correlation_cols = [column for column in corr_matrix2.columns if any(corr_matrix2.loc[upper_triangle, col
# test2 = test.drop(high_correlation_cols, axis=1)
```

NAN 값이 들어 있는 행 삭제하기

```
train.dropna(inplace=True)
test.dropna(inplace=True)
```

데이터 독립 변수 종속 변수로 나누기

```
y_train=train[['USER_ID_hash','Translated_genre_name']]

X_train =train.drop([ 'VIEW_COUPON_ID_hash'],1)
X_test = test.drop([ 'VIEW_COUPON_ID_hash' ],1)

train['REG_DATE'] = pd.to_datetime(X_train['REG_DATE'])
X_train['YEAR'] = X_train['REG_DATE'].dt.year
X_train['MONTH']=X_train['REG_DATE'].dt.month
X_train['DAY']=X_train['REG_DATE'].dt.day


#위 상관관계를 통해 피처 추출
COL_DEL=['USER_ID_hash']
COL_DATE=['YEAR','MONTH','DAY']
COL_NUM=['PURCHASE_FLG','PRICE_RATE']
COL_CAT=['SEX_ID', 'Tr_Pref_Name',
       'Tr_small_area_name',
```

```
       ]
COL_Y=['Translated_genre_name']
```

날짜 데이터를 분할한 이유

-특성 공간의 확장: 날자 정보가 세 개의 특성으로 확장됩니다. 이를 통해 모델은 더 많은 정보를 활용할 수 있게 되며, 이를 통해 모델의 성능이 개선될 수 있습니다.

-계절성 및 주기성 학습: 연월일로 분리된 데이터를 학습하면 계절성이나 주기성을 모델이 파악하고 학습할 수 있습니다

-일반화 능력 강화:연월일로 분리하면, 모델은 날짜가 아니라 월과 일에 주목할 수 있으며, 이를 통해 일반화 능력이 향상될 수 있습니다.

데이터 나누고 스케일링 하기

```
from sklearn.model_selection import train_test_split
X_tr,X_val,y_tr,y_val = train_test_split(X_train[COL_NUM+COL_CAT+COL_DATE],y_train[COL_Y].values.ravel(),test_si

rom sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_tr[COL_NUM] = scaler.fit_transform(X_tr[COL_NUM])
X_val[COL_NUM] = scaler.transform(X_val[COL_NUM])
X_test[COL_NUM] = scaler.transform(X_test[COL_NUM])

m sklearn.preprocessing import LabelEncoder
le = LabelEncoder()


X =pd.concat([X_train[COL_CAT],X_test[COL_CAT]])

for col in COL_CAT:
  le.fit(X[col])
  X_tr[col] = le.transform(X_tr[col])
  X_val[col] = le.transform(X_val[col])
  X_test[col] = le.transform(X_test[col])
```

RandomForestClassifier 모델 생성하기

-랜덤포레스트는 중요성을 측정 용이, 높은 정확도를 보이므로 모델학습에 적합하다 판단함.

```
from sklearn.ensemble import RandomForestClassifier

model_rf = RandomForestClassifier(random_state =42)
model_rf.fit(X_tr, y_tr)
print("Model training finished.")
```


```
predict_rf = model_rf.predict(X_val)
from sklearn.metrics import accuracy_score
score_rf = accuracy_score(y_val,predict_rf)
print(score_rf)
```

```
0.8699426935329243
```

+하이퍼 파라미터 튜닝 코드

```
#최적의 파라미터 찾는 코드 - 실행시간이 너무 오래걸
# from sklearn.model_selection import RandomizedSearchCV
# from sklearn.ensemble import RandomForestClassifier

# # 데이터 준비
# X_train = X_tr
# y_train = y_tr

# # 랜덤 서치를 위한 하이퍼파라미터 분포 설정
# param_distribution = {
#     'n_estimators': [100, 200, 300],
#     'max_depth': [None, 5, 10],
#     'min_samples_split': [2, 5, 10],
#     'min_samples_leaf': [1, 2, 4],
# }

# # 모델 및 랜덤 서치 객체 생성
# model = RandomForestClassifier(random_state=42)
# random_search = RandomizedSearchCV(model, param_distribution, n_iter=10, cv=5)

# # 랜덤 서치 수행
# random_search.fit(X_train, y_train)

# # 최적의 하이퍼파라미터와 최고 성능 출력
# print("Best parameters:", random_search.best_params_)
# print("Best score:", random_search.best_score_)
```

TEST 데이터 성능 예측

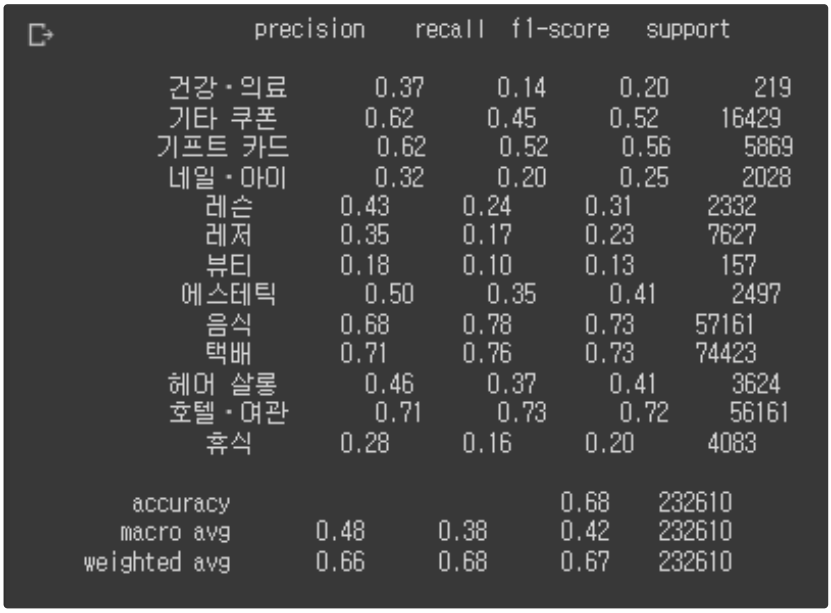
```
pred= model_rf.predict(X_test[COL_NUM+COL_CAT+COL_DATE])
result = pd.DataFrame({'ID':X_test.USER_ID_hash,'pred':pred})
result.tail()
```

	ID	pred
503435	7293708d94cddebb54da37fc7653a4a8	음식
503436	1c1737e59063867041fba66c6510ee77	택배
503437	98c89c24f41da556bd3e5adcf291582a	호텔 · 여관
503438	65ab20a69e9524b3490b1e825f4fb546	호텔 · 여관
503441	7d5b4ba71d12fbcf49579e143b270678	음식

정확도 + 정밀도 + 재현율 계산

```
from sklearn.metrics import classification_report


# 클래스별 분류 보고서 생성
```

```
report = classification_report(actual, predicted)

# 분류 보고서 출력
print(report)
```



```
                precision    recall  f1-score   support

      건강·의료       0.37      0.14      0.20       219
      기타 쿠폰       0.62      0.45      0.52     16429
    기프트 카드       0.62      0.52      0.56      5869
    네일·아이       0.32      0.20      0.25      2028
         레슨       0.43      0.24      0.31      2332
         레저       0.35      0.17      0.23      7627
         뷰티       0.18      0.10      0.13       157
      에스테틱       0.50      0.35      0.41      2497
         음식       0.68      0.78      0.73     57161
         택배       0.71      0.76      0.73     74423
    헤어 살롱       0.46      0.37      0.41      3624
    호텔·여관       0.71      0.73      0.72     56161
         휴식       0.28      0.16      0.20      4083

     accuracy                           0.68    232610
    macro avg       0.48      0.38      0.42    232610
 weighted avg       0.66      0.68      0.67    232610
```

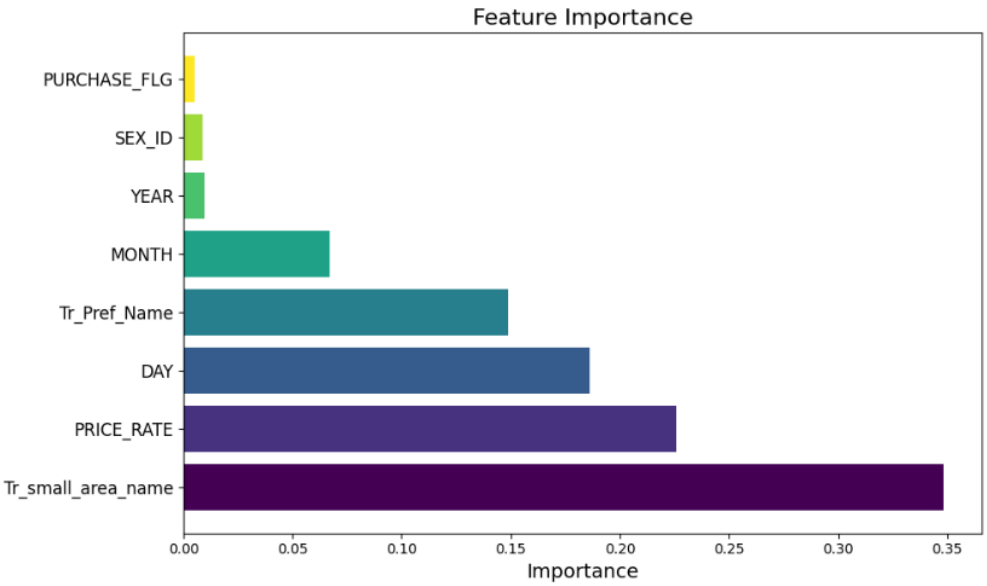
`classification_report`

+리포트 분석

재현율이 낮은 데이터는 모델의 예측 확률이 낮은 것을 확인 할 수 있었다. 이러한 데이터들의 공통적인 특징은 support(데이터 수)가 상대적으로 적기 때문이라 예측 하였다.

Feature Importance 그래프

```
# 색상 맵 정의
colors = plt.cm.viridis(np.linspace(0, 1, len(feature_names)))

# 그래프 크기 설정
plt.figure(figsize=(10, 6))

# 그래프 제목 설정
plt.title("Feature Importance", fontsize=16)

# 가로 막대 그래프 그리기
plt.barh(range(len(feature_names)), importances[indices], color=colors)

# y축 눈금과 레이블 설정
plt.yticks(range(len(feature_names)), feature_names[indices], fontsize=12)

# x축 레이블 설정
plt.xlabel("Importance", fontsize=14)

# 그래프 내부 여백 조정
plt.tight_layout()

# 그래프 출력
plt.show()
```



개선 사항 및 보완점

1. 특정 타겟 값만 정확히 예측 할 수 있는 데이터 값을 갖고 있다 - 샘플의 수를 맞출 수 있는 방법 찾아보기
2. 변수의 Importanace 관계 파악 하기
3. 특성 공학 사용하여 새로운 컬럼 생성하기
4. EDA단계에서 특정 항목으로 치우친 데이터 처리하기