CUAI 스터디 1팀

2024.05.28

발표자 : 김소원, 정현석

스터디원 소개 및 만남 인증

스터디원 1 : 김소원

스터디원 2 : 나영은

스터디원 3 : 나상현

스터디원 4: 박지후

스터디원 5: 정현석

스터디원 6: 조효원

데이터 분석 주제

● 대회 주제

Spaceship Titanic : Predict which passengers are transported to an alternate dimension

KAGGLE · GETTING STARTED PREDICTION COMPETITION · ONGOING

Submit Prediction

Spaceship Titanic

Predict which passengers are transported to an alternate dimension

● 선정 이유

실종된 승객을 구출하기 위해, 우주선의
손상된 컴퓨터 시스템에서 복구된
기록을 사용하여 변칙 현상에 의해
이송된 승객을 예측

→

상상력을 기반한
새로운 칼럼 생성 가능

데이터 소개

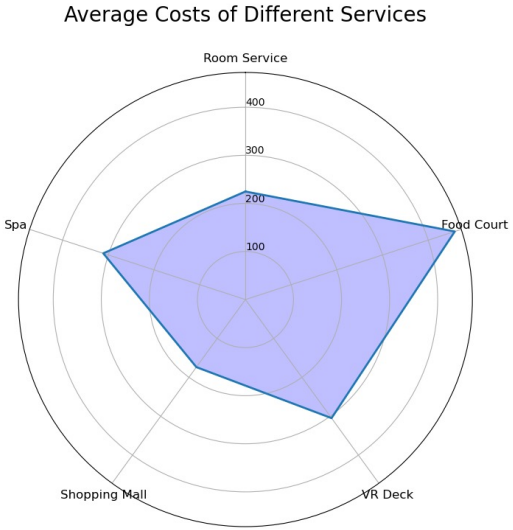
X variable

- **PassengerId**: 각 승객의 고유 ID
- **HomePlanet**: 승객이 출발한 행성, 일반적으로 영구 거주 행성.
- **CryoSleep**: 냉동 수면 여부
- **Cabin**: 승객이 머무는 객실 번호
- **Destination**: 승객이 출발할 행성.
- **Age**: 승객의 나이
- **VIP**: 승객이 항해 중 특별 VIP 서비스에 대한 비용을 지불했는지 여부.
- **RoomService, FoodCourt, ShoppingMall, Spa, VRDeck**
  : 승객이 타이타닉 우주선의 다양한 고급 편의 시설에 대해 청구한 금액
- **Name**: 승객의 이름과 성.

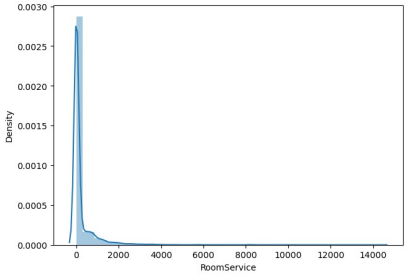
Target variable

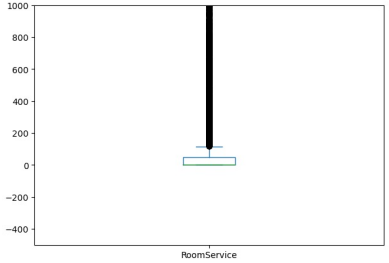
- **Transported**: 승객이 다른 차원으로 이송되었는지 여부

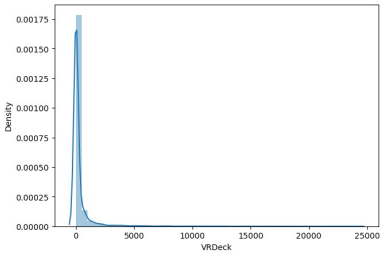
소비 칼럼 범주화

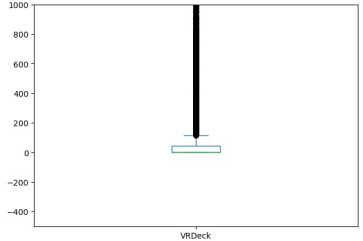

Average Costs of Different Services
Room Service
400
300
200
100
Spa
Food Court
Shopping Mall
VR Deck

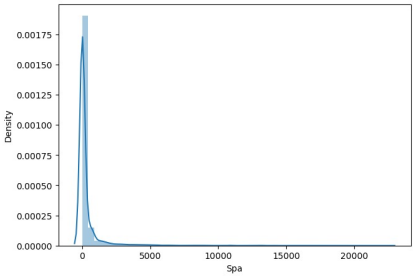
소비 칼럼 범주화

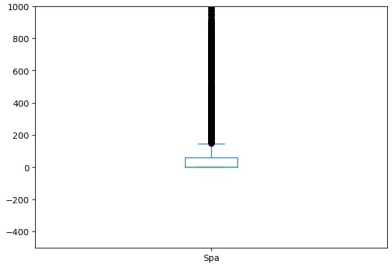

Density
RoomService


RoomService

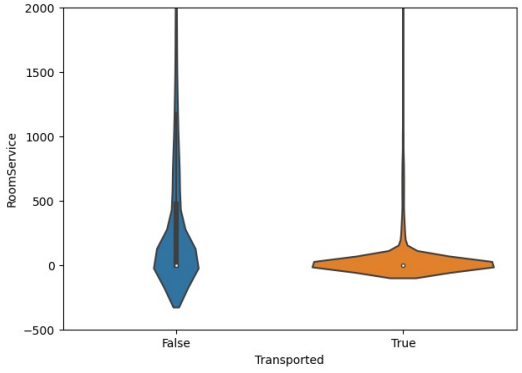

Density
VRDeck

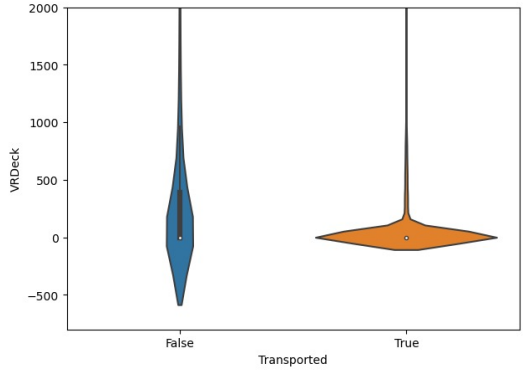

VRDeck

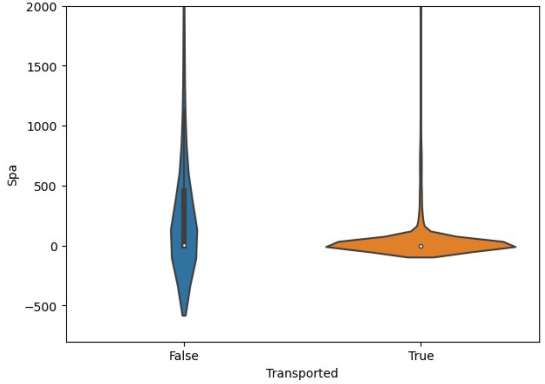

Density
Spa


Spa

소비 칼럼 범주화







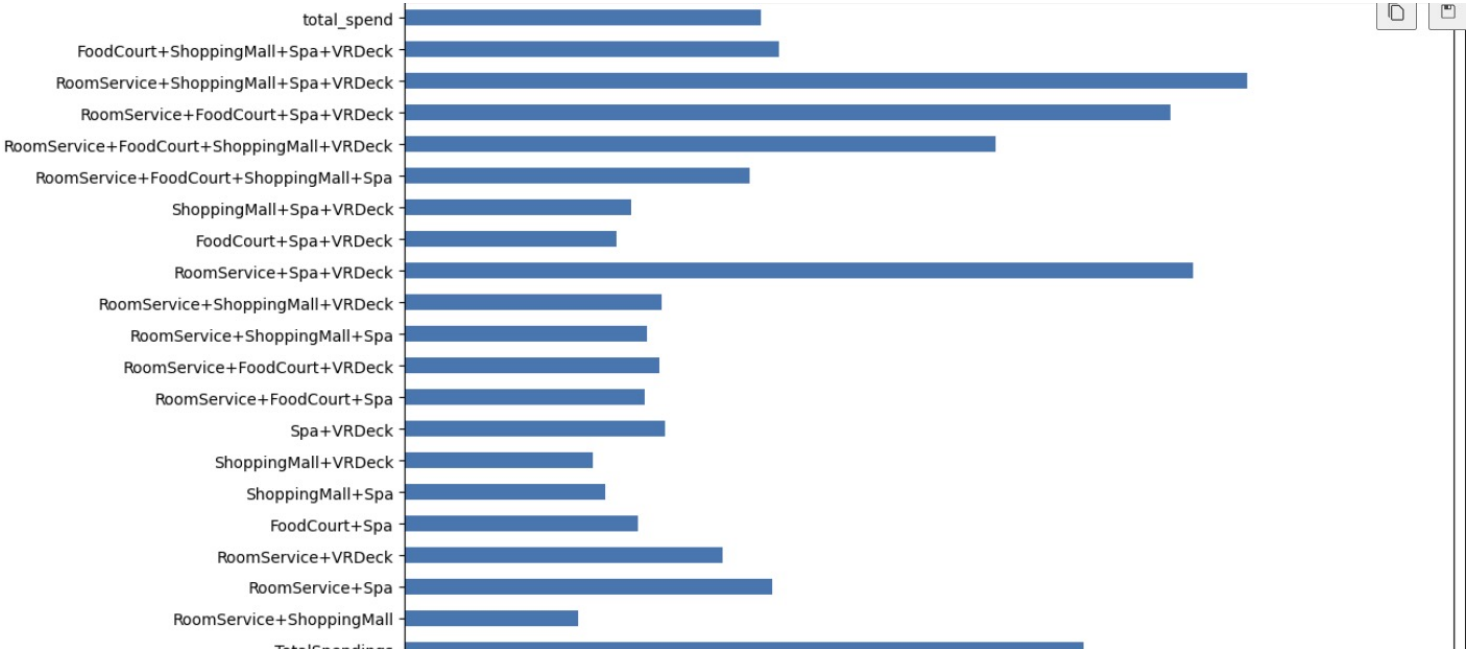
RoomService, Spa, VRDeck이 유사한 분포를 가진다.

소비 칼럼 범주화

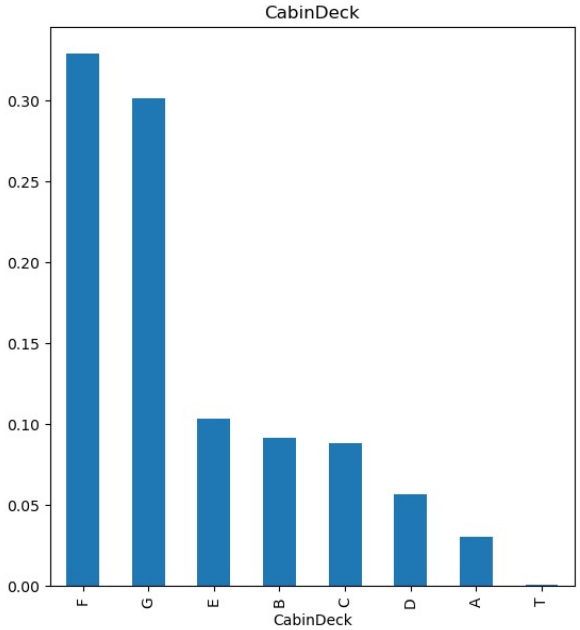
```python
columns = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']

# 2개씩 더한 합계
for combo in combinations(columns, 2):
    col_name = '+'.join(combo)
    df[col_name] = df[list(combo)].sum(axis=1)

# 3개씩 더한 합계
for combo in combinations(columns, 3):
    col_name = '+'.join(combo)
    df[col_name] = df[list(combo)].sum(axis=1)

# 4개씩 더한 합계
for combo in combinations(columns, 4):
    col_name = '+'.join(combo)
    df[col_name] = df[list(combo)].sum(axis=1)

# 5개씩 더한 합계
for combo in combinations(columns, 5):
    col_name = '+'.join(combo)
    df[col_name] = df[list(combo)].sum(axis=1)

df.rename(columns={'RoomService+FoodCourt+ShoppingMall+Spa+VRDeck': 'total_spend'}, inplace=True)
df.head()
```

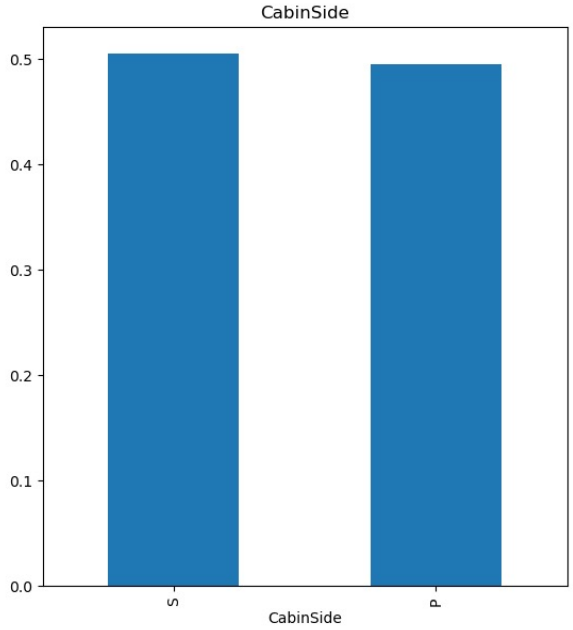
[123] Python

소비 칼럼 범주화


total_spend
FoodCourt+ShoppingMall+Spa+VRDeck
RoomService+ShoppingMall+Spa+VRDeck
RoomService+FoodCourt+Spa+VRDeck
RoomService+FoodCourt+ShoppingMall+VRDeck
RoomService+FoodCourt+ShoppingMall+Spa
ShoppingMall+Spa+VRDeck
FoodCourt+Spa+VRDeck
RoomService+Spa+VRDeck
RoomService+ShoppingMall+VRDeck
RoomService+ShoppingMall+Spa
RoomService+FoodCourt+VRDeck
RoomService+FoodCourt+Spa
Spa+VRDeck
ShoppingMall+VRDeck
ShoppingMall+Spa
FoodCourt+Spa
RoomService+VRDeck
RoomService+Spa
RoomService+ShoppingMall

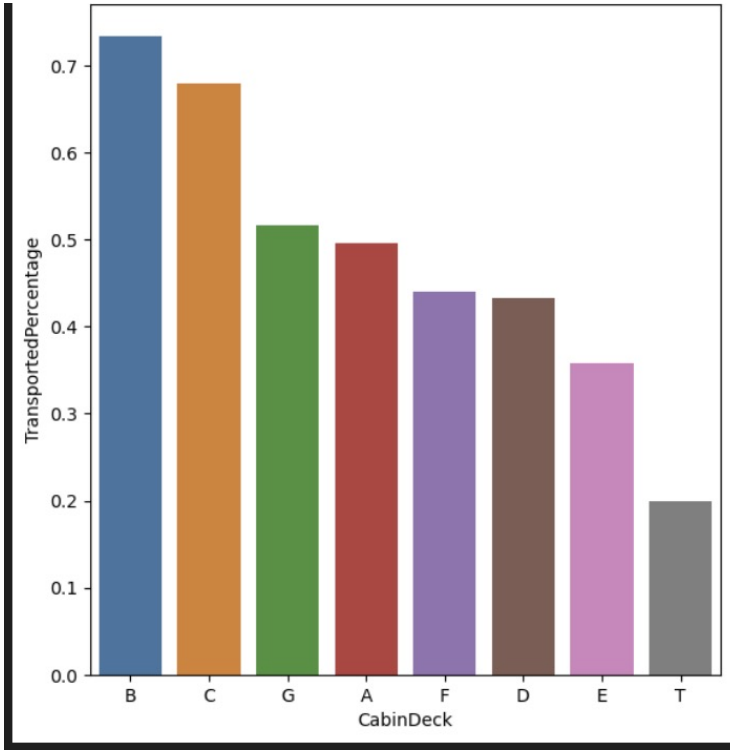
Cabin 칼럼 범주화

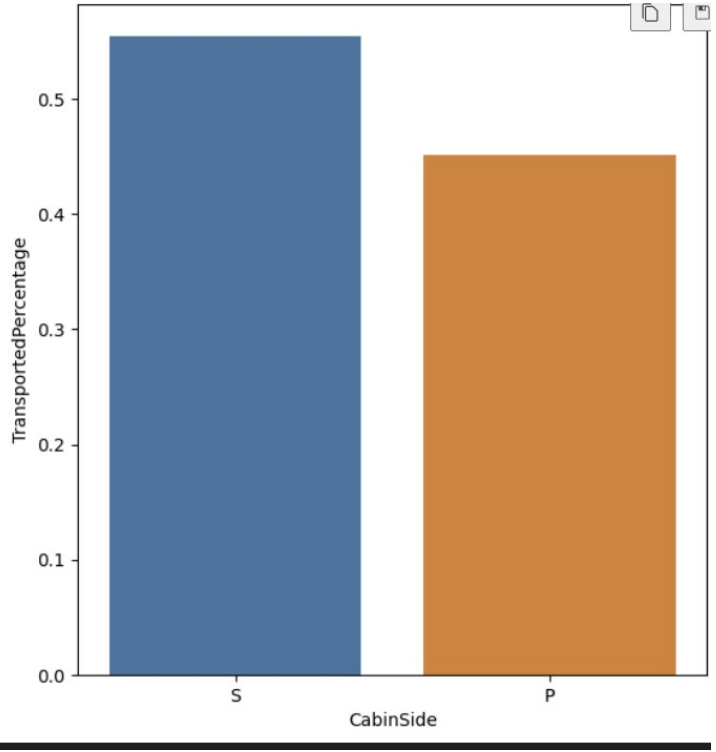

CabinDeck
0.30
0.25
0.20
0.15
0.10
0.05
0.00
F
G
E
B
C
D
A
T
CabinDeck


CabinSide
0.5
0.4
0.3
0.2
0.1
0.0
S
P
CabinSide


Cabin
B/0/P
F/0/S
A/0/S
A/0/S
F/1/S
F/0/P
F/2/S
G/0/S

Cabin 칼럼 범주화


TransportedPercentage
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
B
C
G
A
F
D
E
T
CabinDeck


TransportedPercentage
0.0
0.1
0.2
0.3
0.4
0.5
S
P
CabinSide

Cabin 칼럼 범주화

Cabin 칼럼

- 승객이 머무는 객실 번호
- deck/num/side

```
df[['Deck', 'Cabin_no', 'Side']]= df['Cabin'].str.split('/', expand= True)
df = df.drop('Cabin', axis=1)
df.head()
```

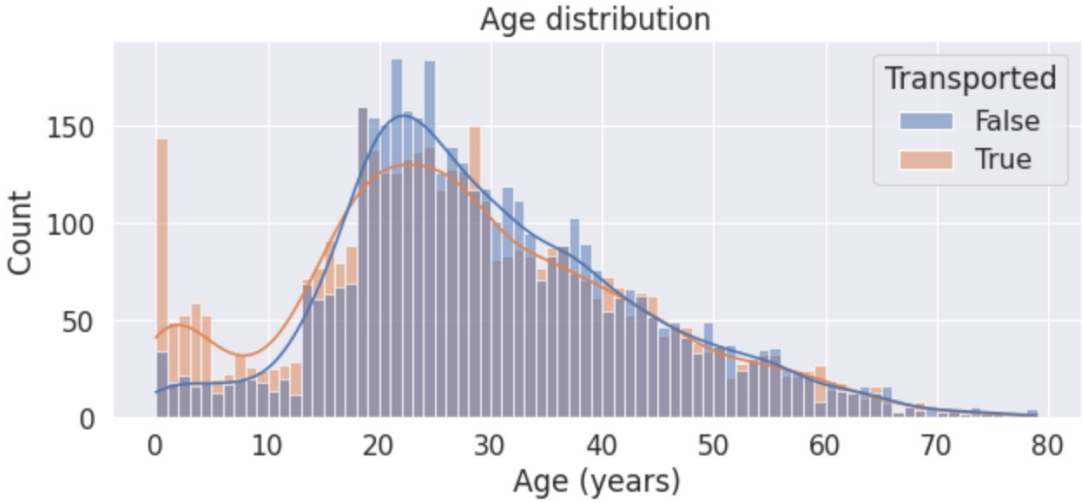
24]

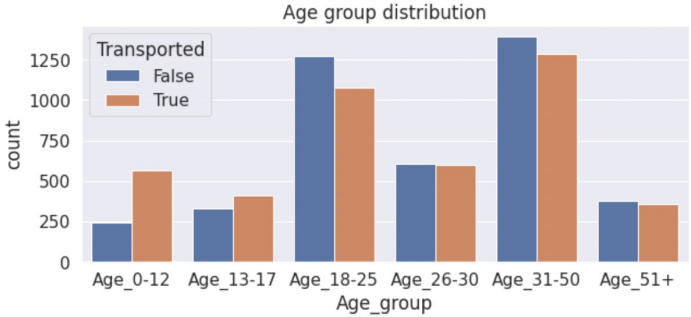
Python

	PassengerId	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	...	ShoppingMall+Spa+VRDeck	RoomServic
0	0001_01	Europa	False	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.00	...	0.00	
1	0002_01	Earth	False	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.00	...	618.00	
2	0003_01	Europa	False	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	5294.52	...	5343.52	
3	0003_02	Europa	False	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.00	...	3893.00	
4	0004_01	Earth	False	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.00	...	718.00	

5 rows × 44 columns

Age 칼럼 범주화


Age distribution
Transported
False
True
Count
150
100
50
0
0
10
20
30
40
50
60
70
80
Age (years)


Age group distribution
Transported
False
True
count
1250
1000
750
500
250
0
Age_0-12
Age_13-17
Age_18-25
Age_26-30
Age_31-50
Age_51+
Age_group

Age 칼럼 범주화

```
# Update age group feature
df.loc[df['Age']<=12,'Age_group']='Age_0-12'
df.loc[(df['Age']>12) & (df['Age']<18),'Age_group']='Age_13-17'
df.loc[(df['Age']>=18) & (df['Age']<=25),'Age_group']='Age_18-25'
df.loc[(df['Age']>25) & (df['Age']<=30),'Age_group']='Age_26-30'
df.loc[(df['Age']>30) & (df['Age']<=50),'Age_group']='Age_31-50'
df.loc[df['Age']>50,'Age_group']='Age_51+'
```

MemberCount 칼럼

- **passenger id : ‘gggg_dd’**

```
df['Group']=df['PassengerId'].astype(str).str[:4].astype(int)
```

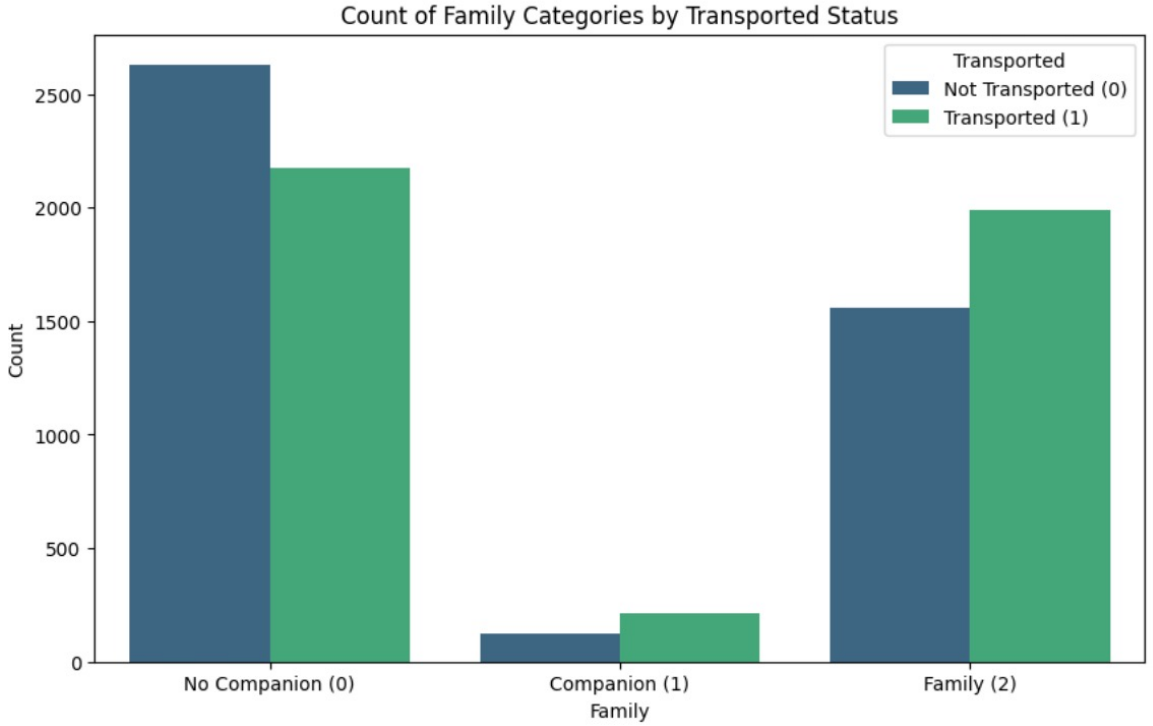
- **Last Name**

```
df['LastName'] = df['Name'].str.split().str[0]
```

- **MemberCount**

```
# 동승자의 수
df['MemberCount'] = df['Group'].map(df['Group'].value_counts())
# 동승자가 없으면 0, 있으면 1, 가족이면 2. 가족인지 판단은 LastName이 중복되는지 여부로 판단
df['Family'] = 0
df.loc[df['MemberCount'] > 1, 'Family'] = 1
df.loc[(df['MemberCount'] > 1) &
(df['LastName'].duplicated(keep=False)), 'Family'] = 2
```

MemberCount 칼럼



적용된 모델

Random Forest

여러 결정 트리의 보팅으로 최종 결정하는 앙상블 알고리즘

LGBM

XGBoost와 함께 가장 각광받는 부스팅 알고리즘
학습시간 적고, 리프 중심의 트리 분할 방식을 사용함

+ K-fold CV

데이터를 k개의 폴드로 나누어 모델을 평가하는 방법
테스트 데이터를 활용하지 않고도 훈련성능을 평가할 수 있음

+ HyperOpt

베이지안 최적화 기법을 사용하여 최적의 하이퍼파라미터 조합을 찾는 라이브러리

Random Forest

sklearn.ensemble 의 RandomForestClassifier 활용

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# X와 y 설정
X = df5.drop(columns=['Transported'])
y = df5['Transported']

# 학습 데이터와 테스트 데이터로 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest 모델 학습
model1 = RandomForestClassifier(random_state=42)
model1.fit(X_train, y_train)

# 모델 예측
y_pred = model1.predict(X_test)

# 성능 평가
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Model Accuracy: {accuracy:.4f}")
print("Classification Report:")
print(report)
```

〈 Accuracy 〉

Train : 0.7798

Kaggle : 0.78933

.csv

Complete · now

0.78933

LightGBM + HyperOpt

lightgbm, hyperopt

```
# 하이퍼파라미터 공간 정의
space = {
    'objective': 'binary',
    'boosting_type': 'gbdt',
    'metric': 'binary_logloss',
    'num_leaves': scope.int(hp.quniform('num_leaves', 20, 40, 1)),
    'learning_rate': hp.uniform('learning_rate', 0.01, 0.2),
    'feature_fraction': hp.uniform('feature_fraction', 0.5, 1.0),
}

# 목적 함수 정의
def objective(hyperparams):
    model = lgb.train(
        hyperparams,
        d_train,
        num_boost_round=100,
        valid_sets=[d_test],
        valid_names=['valid'],
        callbacks=[lgb.early_stopping(stopping_rounds=10)],
        # verbose_eval=0  # 학습 과정을 출력하지 않음
    )
    y_pred_prob = model.predict(X_test, num_iteration=model.best_iteration)
    y_pred = np.round(y_pred_prob)
    accuracy = accuracy_score(y_test, y_pred)

    # 최적화할 목표를 반환: 이 경우 정확도의 음수 값을 반환하여 최대화 문제로 변환
    return {'loss': -accuracy, 'status': STATUS_OK}

# 최적화 실행
trials = Trials()
best = fmin(fn=objective,
            space=space,
            algo=tpe.suggest,
            max_evals=200,
            trials=trials)
```

〈 Accuracy 〉

Train : -

Kaggle : 0.80734

LightGBM.csv

Complete · 25m ago

0.80734

LightGBM + HyperOpt + K-Fold CV

lightgbm, hyperopt, StratifiedKFold

```python
def objective(params):
    accuracies = []
    params['num_leaves'] = int(params['num_leaves'])
    skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

    for train_index, test_index in skf.split(X, y):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        # LightGBM 데이터셋 생성
        d_train = lgb.Dataset(X_train, label=y_train)
        d_test = lgb.Dataset(X_test, label=y_test)

        # 모델 학습
        evals_result = {}
        model = lgb.train(
            params,
            d_train,
            num_boost_round=100,
            valid_sets=[d_test],
            valid_names=['valid'],
            # verbose_eval=False
        )

        # 모델 예측
        y_pred_prob = model.predict(X_test, num_iteration=model.best_iteration)
        y_pred = np.round(y_pred_prob)

        # 성능 평가
        accuracy = accuracy_score(y_test, y_pred)
        accuracies.append(accuracy)
```

〈 Accuracy 〉

Train : 0.8036

Kaggle : 0.80243

k-fold cvLightGBM .csv

Complete · 25m ago

0.80243