In [156]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

# 빅데이터 기반 AI 응용 솔루션 개발자 전문과정

## 교과목명 : 분석용 데이터셋 구축

- 평가일 : 21.8.6
- 성명 : 채승혜
- 점수 :

## Q1. 타이타닉 생존자 예측모델 개발을 위한 Titanic 분석용 데이터셋을 생성한 후 주어진 방법으로 예측 정확도를 평가하세요.

### Titanic data 전처리

- 분석 데이터 : titanic3.csv
- 재사용 가능한 전처리 사용자 함수 작성 하여 전처리
  - Null 값 처리 : Age는 평균나이, 나머지 칼럼은 'N'값으로 변경
  - 불필요한 속성 칼럼 삭제
  - 문자열 칼럼 레이블 인코딩
- 통계적, 시각적 탐색을 통한 다양한 인사이트 도출
- 탐색적 분석을 통한 feature engineering, 파생변수

### 컬럼 정보

- survived : 생존여부(1: 생존, 0 : 사망)
- pclass : 승선권 클래스(1 : 1st, 2 : 2nd ,3 : 3rd)
- name : 승객 이름
- sex : 승객 성별
- age : 승객 나이
- sibsp : 동반한 형제자매, 배우자 수
- parch : 동반한 부모, 자식 수
- ticket : 티켓의 고유 넘버
- fare 티켓의 요금
- cabin : 객실 번호
- embarked : 승선한 항구명(C : Cherbourg, Q : Queenstown, S : Southampton)
- boat
- body
- home.dest

### 점수 산정

- 예측 정확도 0.87 이상 50점

- 예측 정확도 0.85 이상 45점
- 예측 정확도 0.84 이하 정확도/2점

In [157]:

```
df = pd.read_csv('./dataset/titanic3.csv')
df.head(2)
```

Out[157]:

| | pclass | survived | name | sex | age | sibsp | parch | ticket | fare | cabin | embarke |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29.0000 | 0 | 0 | 24160 | 211.3375 | B5 | |
| **1** | 1 | 1 | Allison, Master. Hudson Trevor | male | 0.9167 | 1 | 2 | 113781 | 151.5500 | C22 C26 | |

In [158]:

```
df.isnull().sum()
```

Out[158]:

```
pclass          0
survived        0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

In [159]:

```
df.drop(['cabin','body','home.dest','ticket','boat'],axis=1,inplace=True)
```

## 1. age 결측치 처리

In [160]:

```
# name으로 title_cat 만들기


for i in df:
    df['title_cat']= df.name.str.extract('([a-zA-Z]+)W.')
```

In [161]:

```
df.title_cat.unique()
```

Out[161]:

```
array(['Miss', 'Master', 'Mr', 'Mrs', 'Col', 'Mme', 'Dr', 'Major', 'Capt',
       'Lady', 'Sir', 'Mlle', 'Dona', 'Jonkheer', 'Countess', 'Don',
       'Rev', 'Ms'], dtype=object)
```

In [162]:

```
df.title_cat.replace(['Miss', 'Master', 'Mr', 'Mrs', 'Col', 'Mme', 'Dr', 'Major', 'Capt',
       'Lady', 'Sir', 'Mlle', 'Dona', 'Jonkheer', 'Countess', 'Don',
       'Rev', 'Ms'],
        ['Miss', 'Master', 'Mr', 'Mrs', 'Other', 'Other', 'Other', 'Other', 'Other',
       'Miss', 'Other', 'Other', 'Other', 'Other', 'Other', 'Other',
       'Other', 'Ms'],inplace=True)
```

In [163]:

```
df.groupby('title_cat')['age'].mean()
```

Out[163]:

```
title_cat
Master     5.482704
Miss      21.898500
Mr        32.252151
Mrs       36.994118
Ms        28.000000
Other     42.966667
Name: age, dtype: float64
```

In [164]:

```
df.loc[(df.age.isnull()&(df.title_cat=='Master')),'age'] = 5
df.loc[(df.age.isnull()&(df.title_cat=='Miss')),'age'] = 22
df.loc[(df.age.isnull()&(df.title_cat=='Mr')),'age'] = 32
df.loc[(df.age.isnull()&(df.title_cat=='Mrs')),'age'] = 37
df.loc[(df.age.isnull()&(df.title_cat=='Ms')),'age'] = 28
df.loc[(df.age.isnull()&(df.title_cat=='Other')),'age'] = 43

df.age.isnull().sum()
```

Out[164]:

```
0
```

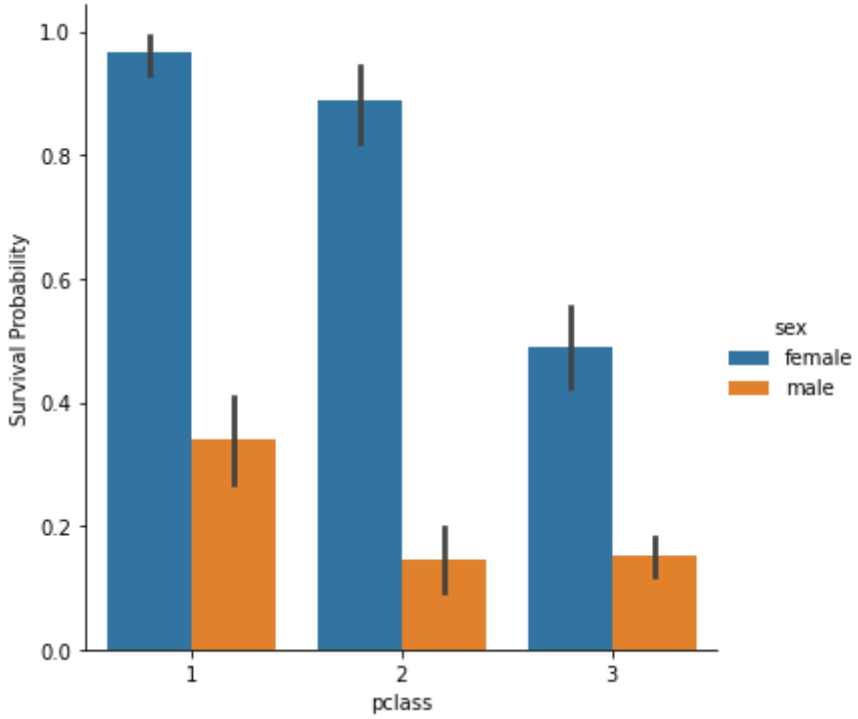
## pclass / sex 에 따른 생존확률

In [165]:

```
import warnings
warnings.filterwarnings('ignore')
dp = sns.factorplot(x="pclass",y="survived",hue='sex',data=df,kind="bar")
dp.set_ylabels("Survival Probability")
```

Out[165]:

<seaborn.axisgrid.FacetGrid at 0x13c4811cf10>




## embarked 결측치 처리

In [166]:

```
df.embarked.value_counts()
```

Out[166]:

```
S    914
C    270
Q    123
Name: embarked, dtype: int64
```

In [167]:

```
df.embarked.fillna('S',inplace=True)
```

In [168]:

```
df.embarked.isnull().sum()
```

Out[168]:

0

## fare 결측값 처리

In [169]:

```
df[df.fare.isnull()]
```

Out[169]:

| | pclass | survived | name | sex | age | sibsp | parch | fare | embarked | title_cat |
|---|---|---|---|---|---|---|---|---|---|---|
| **1225** | 3 | 0 | Storey, Mr. Thomas | male | 60.5 | 0 | 0 | NaN | S | Mr |

In [170]:

```
df.groupby(['pclass'])['fare'].mean()
```

Out[170]:

```
pclass
1    87.508992
2    21.179196
3    13.302889
Name: fare, dtype: float64
```

In [171]:

```
df.fare.fillna(13.3,inplace=True)

df.fare.isnull().sum()
```

Out[171]:

0

## age 범주화

In [172]:

```
def cat_age(x):
    cat = ''
    if x < 10:
        cat = 'young'
    elif 10<= x <20 :
        cat = 'teen'
    elif 21<= x <40 :
        cat = 'adult'
    elif 40<= x <60 :
        cat = 'mature'
    else:
        cat = 'elder'
    return cat
```

In [173]:

```
age_cat = df.age.apply(lambda x : cat_age(x))
df['age_cat'] = age_cat
df.head(1)
```

Out[173]:

| | pclass | survived | name | sex | age | sibsp | parch | fare | embarked | title_cat | age_c |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29.0 | 0 | 0 | 211.3375 | S | Miss | adu |

## fare 범주화

In [174]:

```
df.fare.describe()
```

Out[174]:

```
count    1309.000000
mean       33.280204
std        51.741831
min         0.000000
25%         7.895800
50%        14.454200
75%        31.275000
max       512.329200
Name: fare, dtype: float64
```

In [175]:

```
def fare_cat(x):
    cat = ''
    if x <= df.fare.describe()['25%']:
        cat = 0
    elif df.fare.describe()['25%']< x <= df.fare.describe()['50%']:
        cat = 1
    elif df.fare.describe()['50%']< x <= df.fare.describe()['75%']:
        cat = 2
    else:
        cat = 3
    return cat

df['fare_cat'] = df['fare'].apply(lambda x: fare_cat(x))

df.head(1)
```

Out[175]:

| | pclass | survived | name | sex | age | sibsp | parch | fare | embarked | title_cat | age_c |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29.0 | 0 | 0 | 211.3375 | S | Miss | adu |

## 구성원 구하기

In [176]:

```
df['member']= df['sibsp'] + df['parch']
df.head(2)
```

Out[176]:

| | pclass | survived | name | sex | age | sibsp | parch | fare | embarked | title_cat | ag |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29.0000 | 0 | 0 | 211.3375 | S | Miss | |
| **1** | 1 | 1 | Allison, Master. Hudson Trevor | male | 0.9167 | 1 | 2 | 151.5500 | S | Master | |

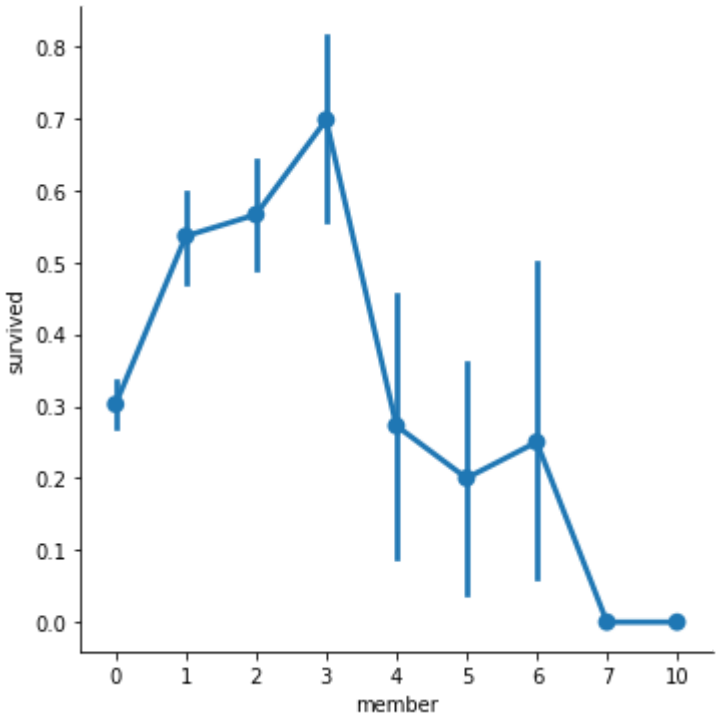
In [177]:

```
import warnings
warnings.filterwarnings('ignore')

sns.factorplot('member','survived',data = df)
```

Out[177]:

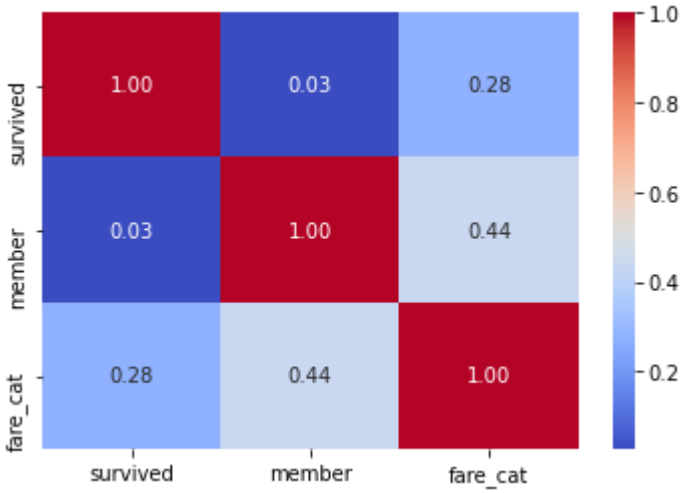
<seaborn.axisgrid.FacetGrid at 0x13c46ac6eb0>




## survived- member, age, fare상관 관계 heatmap

In [178]:

```
dh =sns.heatmap(df[['survived','member','age_cat','fare_cat']].corr(),annot=True, fmt = ".2f", cmap
```




## 인코딩

In [179]:

```
from sklearn.preprocessing import LabelEncoder
```

In [180]:

```
df = pd.get_dummies(df, columns=['sex'])
df.head(1)
```

Out[180]:

| | pclass | survived | name | age | sibsp | parch | fare | embarked | title_cat | age_cat | fare_ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | 29.0 | 0 | 0 | 211.3375 | S | Miss | adult | |

In [181]:

```
le= LabelEncoder()
features= ['age_cat','title_cat','fare_cat','embarked']

for feature in features:
    df[feature]= le.fit_transform(df[feature])
df.head(2)
```

Out[181]:

| | pclass | survived | name | age | sibsp | parch | fare | embarked | title_cat | age_cat | f |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | 29.0000 | 0 | 0 | 211.3375 | 2 | 1 | 0 | |
| **1** | 1 | 1 | Allison, Master. Hudson Trevor | 0.9167 | 1 | 2 | 151.5500 | 2 | 0 | 4 | |

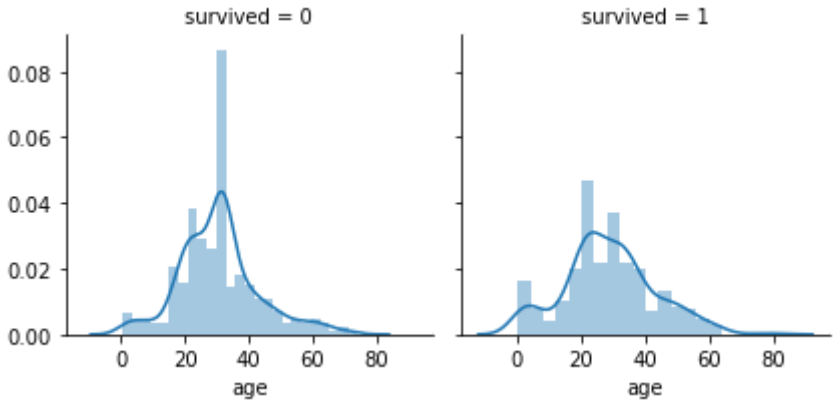
## 사망자/생존자의 나이 분포도

In [182]:

```
dfg = sns.FacetGrid(df, col='survived')
dfg.map(sns.distplot, "age")
```

Out[182]:

<seaborn.axisgrid.FacetGrid at 0x13c46673430>



In [183]:

```
tdf= df.drop(['name','age','parch','member','sibsp','fare','title_cat','embarked'],axis=1)
tdf.head(2)
```

Out[183]:

| | pclass | survived | age_cat | fare_cat | member | sex_female | sex_male |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 0 | 3 | 0 | 1 | 0 |
| **1** | 1 | 1 | 4 | 3 | 3 | 0 | 1 |

In [322]:

```
tdf= df[['pclass','survived','age_cat','fare_cat','member','sex_female','sex_male']]
tdf.head(2)
```

Out[322]:

| | pclass | survived | age_cat | fare_cat | member | sex_female | sex_male |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 0 | 3 | 0 | 1 | 0 |
| **1** | 1 | 1 | 4 | 3 | 3 | 0 | 1 |

In [323]:

```
# 모델 평가 준비 작업
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

# 독립변수, 종속변수 분리
y_t_df = tdf['survived'] # 종속변수
X_t_df = tdf.drop('survived', axis = 1) # 독립변수

# 독립변수 정규화
# X_t_df = preprocessing.StandardScaler().fit(X_t_df).transform(X_t_df)

# 학습용 데이터와 평가용 데이터를 8:2 혹은 7:3으로 분리
X_train, X_test, y_train, y_test = train_test_split(X_t_df, y_t_df, test_size = 0.2,
                                                    random_state = 11)

print(X_train.shape)
print(X_test.shape)
```

```
(1047, 6)
(262, 6)
```

In [324]:

```
# 모델 학습 및 평가
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
accuracy_rf = accuracy_score(y_test, rf_pred).round(2)

lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
lr_pred = lr_model.predict(X_test)
accuracy_lr = accuracy_score(y_test,lr_pred).round(2)

print('rf 정확도:{}, lr 정확도:{}'.format(accuracy_rf,accuracy_lr))
```

```
rf 정확도:0.83, lr 정확도:0.8
```

## Q2. 고객 속성 및 거래 데이터를 아래 제시된 방법으로 생성 후 다음 과제를 수행하세요.

[고객별 속성 데이터]

- id : 1번 ~ 100번 일련번호
- gender : 0,1 정수 난수 생성
- age : 10 ~ 80사이 정수 난수를 생성하고 범주화하여 age_cat 파생
- region : 1 ~ 5 사이 정수 난수 생성

[거래 데이터]

- id : 1번 ~ 100번 일련번호(관측치 1000개)
- product : 고급제품(hpd), 일반제품(lpd)로 구분
- price : 100 ~ 200 사이 정수 난수 생성
- qty : 1 ~ 2 사이 정수 난수 생성
- day : 요일
- date : 2020-01-01 ~ 2021-12-31 사이의 날짜 1000개 생성
- amount : price * qty로 산출

[과제]

- 고객 속성(id, 성별, 연령, 거주지역) 데이터와 거래 데이터를 통합한 데이터 프레임 df를 생성하세요.
- df를 수정하여 5가지 이상의 인사이트를 포함한 데이터 셋으로 변환하세요
- df 를 탐색적 분석을 통하여 인사이트를 도출하세요.

### 점수 산정

- 인사이트 개수에 따라 점수 부여: 5개 50점,4개 45점,3개 40점,2개 35점,1개 30점

In [325]:

```
# 인사이트 예시(요약 기술 및 관련 통계 및 시각화 제시)
# 일반제품의 변화율인 lpd 컬럼과 구매증감율 ratio의 상관관계가 0.5 초과
```

## *고객별 속성 데이터

In [363]:

```
np.random.seed(0)

id_c = np.arange(1,101)

gender = np.random.randint(2,size=100)


age=np.random.randint(10,81,size=100)


region = np.random.randint(1,6,size=100)


df1 = pd.DataFrame({'id_c':id_c, 'gender':gender, 'age':age, 'region':region,})
df1.head(3)
```

Out[363]:

| | id_c | gender | age | region |
|---|---|---|---|---|
| **0** | 1 | 0 | 52 | 4 |
| **1** | 2 | 1 | 78 | 3 |
| **2** | 3 | 1 | 16 | 4 |

In [364]:

```
# age 범주화

def age_cat(x):
    cat=''
    if x < 30:
        cat = 1
    elif x < 40:
        cat = 2
    elif x < 50:
        cat = 3
    elif x < 60:
        cat = 4
    else:
        cat = 5
    return cat

df1['age_cat'] = df1.age.apply(lambda x : age_cat(x))
df1= df1.drop('age',axis=1)
df1.head(3)
```

Out[364]:

| | id_c | gender | region | age_cat |
|---|---|---|---|---|
| **0** | 1 | 0 | 4 | 4 |
| **1** | 2 | 1 | 3 | 5 |
| **2** | 3 | 1 | 4 | 1 |

## *거래데이터

In [365]:

```
data = np.zeros((1000,6))

df2 = pd.DataFrame(data,columns=['id_c','product','price','qty','day','year'])
```

In [366]:

```
np.random.seed(0)

df2['id_c'] = np.random.randint(1,101,1000)

df2['product'] = np.random.randint(1,3,1000)

df2['price'] = np.random.randint(100,200,1000)

df2['qty'] = np.random.randint(1,6,1000)

df2['day'] = np.random.choice(['MON','TUE','WED','THU','FRI','SAT','SUN'],1000)

df2['year'] = np.random.randint(2019,2021, size = 1000)

df2.head()
```

Out[366]:

| | id_c | product | price | qty | day | year |
|---|---|---|---|---|---|---|
| **0** | 45 | 1 | 149 | 4 | MON | 2020 |
| **1** | 48 | 1 | 171 | 2 | TUE | 2020 |
| **2** | 65 | 2 | 139 | 3 | TUE | 2020 |
| **3** | 68 | 2 | 158 | 5 | MON | 2020 |
| **4** | 68 | 1 | 105 | 1 | MON | 2019 |

In [367]:

```
# product 고가/저가 나누기 (1/2)

def pp(product,price):
    if product==1:
        return price*50
    if product==2:
        return price*20

df2['price']= df2.apply(lambda x: pp(x['product'],x['price']),axis=1)
df2.head(3)
```

Out[367]:

| | id_c | product | price | qty | day | year |
|---|---|---|---|---|---|---|
| **0** | 45 | 1 | 7450 | 4 | MON | 2020 |
| **1** | 48 | 1 | 8550 | 2 | TUE | 2020 |
| **2** | 65 | 2 | 2780 | 3 | TUE | 2020 |

In [368]:

```
# 구매개수 (고가제품일 경우 적게 사고, 저가 제품일수록 많이 )

def pq(product, qty):
    if product==1:
        return qty
    if product==2:
        return qty*10

df2['qty']= df2.apply(lambda x: pq(x['product'],x['qty']),axis=1)
```

In [ ]:

```

```

In [ ]:

```

```

## 인사이트 - 고가 제품의 이미지가 좋아져서 수요 상승

In [369]:

```
def pdcut(year,product,qty):
    if (year==2020) & (product==1):
        return qty * 2
    else:
        return qty
df2['qty'] = df2.apply(lambda x : pdcut(x['year'],x['product'],x['qty']),axis=1)
df2.head()
```

Out[369]:

| | id_c | product | price | qty | day | year |
|---|---|---|---|---|---|---|
| **0** | 45 | 1 | 7450 | 8 | MON | 2020 |
| **1** | 48 | 1 | 8550 | 4 | TUE | 2020 |
| **2** | 65 | 2 | 2780 | 30 | TUE | 2020 |
| **3** | 68 | 2 | 3160 | 50 | MON | 2020 |
| **4** | 68 | 1 | 5250 | 1 | MON | 2019 |

In [370]:

```
df2['amount'] = df2['qty'] * df2['price']

df2.head()
```

Out[370]:

| | id_c | product | price | qty | day | year | amount |
|---|---|---|---|---|---|---|---|
| **0** | 45 | 1 | 7450 | 8 | MON | 2020 | 59600 |
| **1** | 48 | 1 | 8550 | 4 | TUE | 2020 | 34200 |
| **2** | 65 | 2 | 2780 | 30 | TUE | 2020 | 83400 |
| **3** | 68 | 2 | 3160 | 50 | MON | 2020 | 158000 |
| **4** | 68 | 1 | 5250 | 1 | MON | 2019 | 5250 |

In [371]:

```
df2 = df2.drop(['price','qty'],axis=1)
```

In [372]:

```
df2.groupby(['year','product'])['amount'].sum()
```

Out[372]:

```
year  product
2019  1          5425250
      2         22442000
2020  1         10901400
      2         23335400
Name: amount, dtype: int64
```

In [373]:

```
df2['y_p'] = df2[['year','product']].astype(str).apply('_'.join,axis=1)
df2.head()
```

Out[373]:

| | id_c | product | day | year | amount | y_p |
|---|---|---|---|---|---|---|
| **0** | 45 | 1 | MON | 2020 | 59600 | 2020_1 |
| **1** | 48 | 1 | TUE | 2020 | 34200 | 2020_1 |
| **2** | 65 | 2 | TUE | 2020 | 83400 | 2020_2 |
| **3** | 68 | 2 | MON | 2020 | 158000 | 2020_2 |
| **4** | 68 | 1 | MON | 2019 | 5250 | 2019_1 |

In [374]:

```
dff = df2.drop(['product','year'],axis=1)
dff.head()
```

Out[374]:

| | id_c | day | amount | y_p |
|---|---|---|---|---|
| **0** | 45 | MON | 59600 | 2020_1 |
| **1** | 48 | TUE | 34200 | 2020_1 |
| **2** | 65 | TUE | 83400 | 2020_2 |
| **3** | 68 | MON | 158000 | 2020_2 |
| **4** | 68 | MON | 5250 | 2019_1 |

In [375]:

```
df3= pd.pivot_table(dff,index = 'id_c',columns= 'y_p',values = 'amount',aggfunc = 'sum')
df3.head()
```

Out[375]:

| y_p | 2019_1 | 2019_2 | 2020_1 | 2020_2 |
|---|---|---|---|---|
| **id_c** | | | | |
| **1** | 143700.0 | 258400.0 | 119500.0 | 352600.0 |
| **2** | 69450.0 | 275200.0 | 30800.0 | NaN |
| **3** | 32100.0 | 33200.0 | 78400.0 | 187000.0 |
| **4** | 37800.0 | 407400.0 | 446700.0 | 218600.0 |
| **5** | 58550.0 | 169600.0 | 58000.0 | 332000.0 |

In [376]:

```
df3.isnull().sum()
```

Out[376]:

```
y_p
2019_1    10
2019_2     5
2020_1    10
2020_2     8
dtype: int64
```

In [377]:

```
df3['2019_1'].fillna(df3['2019_1'].mean(),inplace=True)
df3['2019_2'].fillna(df3['2019_2'].mean(),inplace=True)
df3['2020_1'].fillna(df3['2020_1'].mean(),inplace=True)
df3['2020_2'].fillna(df3['2020_2'].mean(),inplace=True)

df3.head()
```

Out[377]:

| y_p<br>id_c | 2019_1 | 2019_2 | 2020_1 | 2020_2 |
|---|---|---|---|---|
| **1** | 143700.0 | 258400.0 | 119500.0 | 352600.000000 |
| **2** | 69450.0 | 275200.0 | 30800.0 | 253645.652174 |
| **3** | 32100.0 | 33200.0 | 78400.0 | 187000.000000 |
| **4** | 37800.0 | 407400.0 | 446700.0 | 218600.000000 |
| **5** | 58550.0 | 169600.0 | 58000.0 | 332000.000000 |

In [378]:

```
# 변화율

df3['hpd'] = (df3['2020_1'] - df3['2019_1']) / df3['2019_1']
df3['lpd'] = (df3['2020_2'] - df3['2019_2']) / df3['2019_2']
df3.head()
df4= df3[['hpd','lpd']]
df4.head()
```

Out[378]:

| y_p<br>id_c | hpd | lpd |
|---|---|---|
| **1** | -0.168406 | 0.364551 |
| **2** | -0.556515 | -0.078322 |
| **3** | 1.442368 | 4.632530 |
| **4** | 10.817460 | -0.463427 |
| **5** | -0.009394 | 0.957547 |

In [379]:

```
df5= pd.merge(df1,df4,on='id_c', how='outer')
df5.head()
```

Out[379]:

| | id_c | gender | region | age_cat | hpd | lpd |
|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 4 | 4 | -0.168406 | 0.364551 |
| **1** | 2 | 1 | 3 | 5 | -0.556515 | -0.078322 |
| **2** | 3 | 1 | 4 | 1 | 1.442368 | 4.632530 |
| **3** | 4 | 0 | 3 | 5 | 10.817460 | -0.463427 |
| **4** | 5 | 1 | 5 | 4 | -0.009394 | 0.957547 |

In [380]:

```
# 종속변수 정의
dfy = df2[['id_c','year','amount']]
pdf = pd.pivot_table(dfy,index='id_c',columns='year',values='amount',aggfunc='sum')

# pvt.columns.name=None
pdf.head()
```

Out[380]:

| year | 2019 | 2020 |
|---|---|---|
| **id_c** | | |
| **1** | 402100.0 | 472100.0 |
| **2** | 344650.0 | 30800.0 |
| **3** | 65300.0 | 265400.0 |
| **4** | 445200.0 | 665300.0 |
| **5** | 228150.0 | 390000.0 |

In [381]:

```
pdf.isnull().sum()
```

Out[381]:

```
year
2019    0
2020    2
dtype: int64
```

In [382]:

```
pdf[2020].fillna(pdf[2020].mean(),inplace=True)
```

In [383]:

```
dt= pd.merge(df5,pdf,on='id_c',how='outer')
dt.head()
```

Out[383]:

| | id_c | gender | region | age_cat | hpd | lpd | 2019 | 2020 |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 4 | 4 | -0.168406 | 0.364551 | 402100.0 | 472100.0 |
| **1** | 2 | 1 | 3 | 5 | -0.556515 | -0.078322 | 344650.0 | 30800.0 |
| **2** | 3 | 1 | 4 | 1 | 1.442368 | 4.632530 | 65300.0 | 265400.0 |
| **3** | 4 | 0 | 3 | 5 | 10.817460 | -0.463427 | 445200.0 | 665300.0 |
| **4** | 5 | 1 | 5 | 4 | -0.009394 | 0.957547 | 228150.0 | 390000.0 |

In [384]:

```
# 상승률
dt['ratio']= (dt[2020]-dt[2019]) /dt[2019]
```

In [385]:

```
dt
```

Out[385]:

| | id_c | gender | region | age_cat | hpd | lpd | 2019 | 2020 | ratio |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 4 | 4 | -0.168406 | 0.364551 | 402100.0 | 472100.0 | 0.174086 |
| **1** | 2 | 1 | 3 | 5 | -0.556515 | -0.078322 | 344650.0 | 30800.0 | -0.910634 |
| **2** | 3 | 1 | 4 | 1 | 1.442368 | 4.632530 | 65300.0 | 265400.0 | 3.064319 |
| **3** | 4 | 0 | 3 | 5 | 10.817460 | -0.463427 | 445200.0 | 665300.0 | 0.494385 |
| **4** | 5 | 1 | 5 | 4 | -0.009394 | 0.957547 | 228150.0 | 390000.0 | 0.709402 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **95** | 96 | 0 | 1 | 1 | -0.216995 | -0.399209 | 253000.0 | 199200.0 | -0.212648 |
| **96** | 97 | 0 | 5 | 3 | 2.203355 | -0.125360 | 290000.0 | 193100.0 | -0.334138 |
| **97** | 98 | 1 | 4 | 1 | -0.421920 | 0.727575 | 171950.0 | 237800.0 | 0.382960 |
| **98** | 99 | 1 | 4 | 4 | 0.592553 | 1.055315 | 184400.0 | 475000.0 | 1.575922 |
| **99** | 100 | 0 | 1 | 2 | -0.416867 | 2.836397 | 191800.0 | 465800.0 | 1.428571 |

100 rows × 9 columns

In [386]:

```
dt_t= dt[['gender','region','age_cat','hpd','lpd','ratio']]
dt_t
```

Out[386]:

| | gender | region | age_cat | hpd | lpd | ratio |
|---|---|---|---|---|---|---|
| **0** | 0 | 4 | 4 | -0.168406 | 0.364551 | 0.174086 |
| **1** | 1 | 3 | 5 | -0.556515 | -0.078322 | -0.910634 |
| **2** | 1 | 4 | 1 | 1.442368 | 4.632530 | 3.064319 |
| **3** | 0 | 3 | 5 | 10.817460 | -0.463427 | 0.494385 |
| **4** | 1 | 5 | 4 | -0.009394 | 0.957547 | 0.709402 |
| **...** | ... | ... | ... | ... | ... | ... |
| **95** | 0 | 1 | 1 | -0.216995 | -0.399209 | -0.212648 |
| **96** | 0 | 5 | 3 | 2.203355 | -0.125360 | -0.334138 |
| **97** | 1 | 4 | 1 | -0.421920 | 0.727575 | 0.382960 |
| **98** | 1 | 4 | 4 | 0.592553 | 1.055315 | 1.575922 |
| **99** | 0 | 1 | 2 | -0.416867 | 2.836397 | 1.428571 |

100 rows × 6 columns

In [387]:

```
dt_t[['hpd','lpd','ratio']].corr()
```

Out[387]:

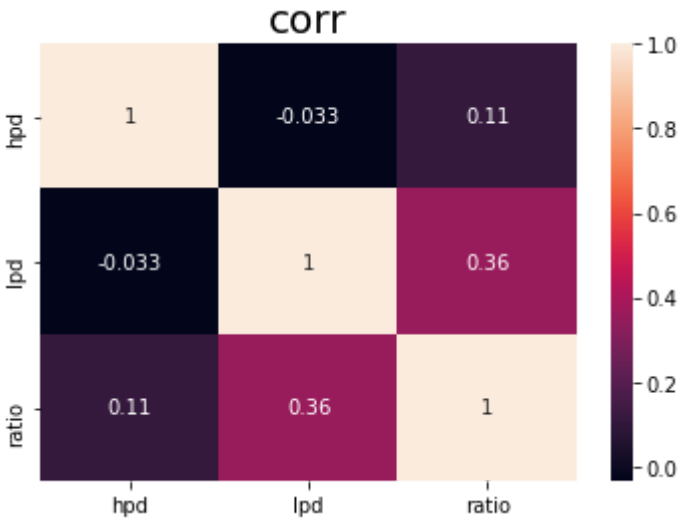
| | hpd | lpd | ratio |
|---|---|---|---|
| **hpd** | 1.000000 | -0.033082 | 0.111894 |
| **lpd** | -0.033082 | 1.000000 | 0.361511 |
| **ratio** | 0.111894 | 0.361511 | 1.000000 |

In [432]:

```
sns.heatmap(dt_t[['hpd','lpd','ratio']].corr(),annot=True)
plt.title('corr',fontsize=20)
```

Out[432]:

```
Text(0.5, 1.0, 'corr')
```




In [ ]:

```
# lpd- ratio / hpd-ratio는 0.36의 상관관계를 갖는다.
```

In [388]:

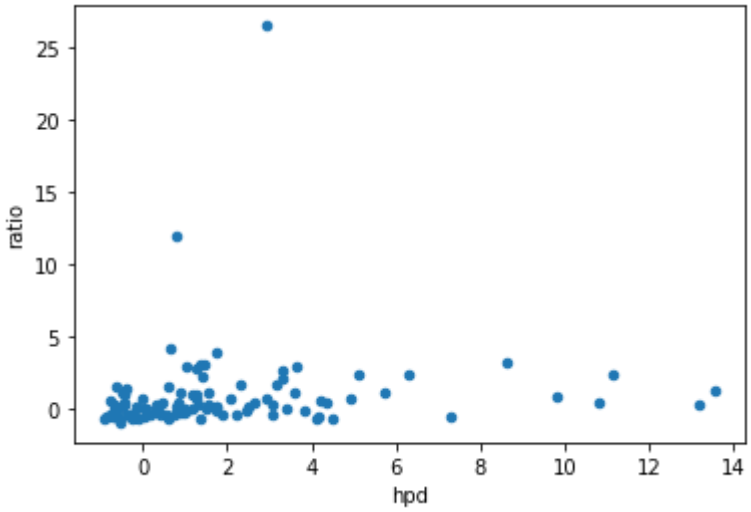
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [389]:

```
dt_t.plot(x='hpd',y='ratio',kind='scatter')
```

Out[389]:

<AxesSubplot:xlabel='hpd', ylabel='ratio'>

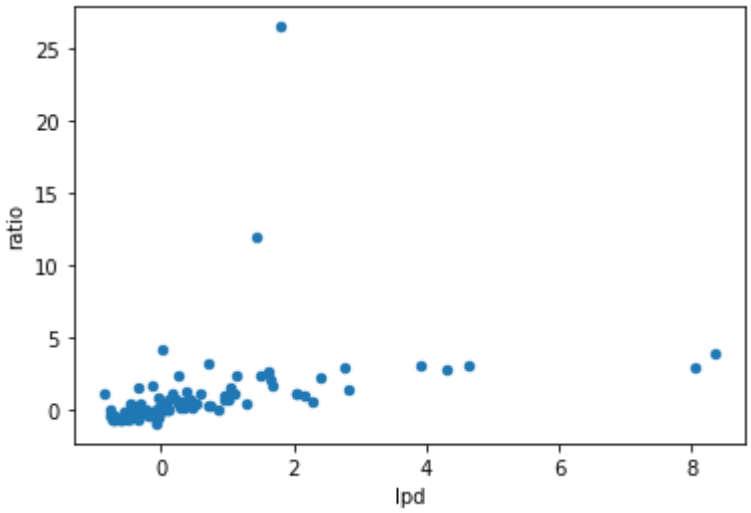



In [390]:

```
dt_t.plot(x='lpd',y='ratio',kind='scatter')
```

Out[390]:

<AxesSubplot:xlabel='lpd', ylabel='ratio'>

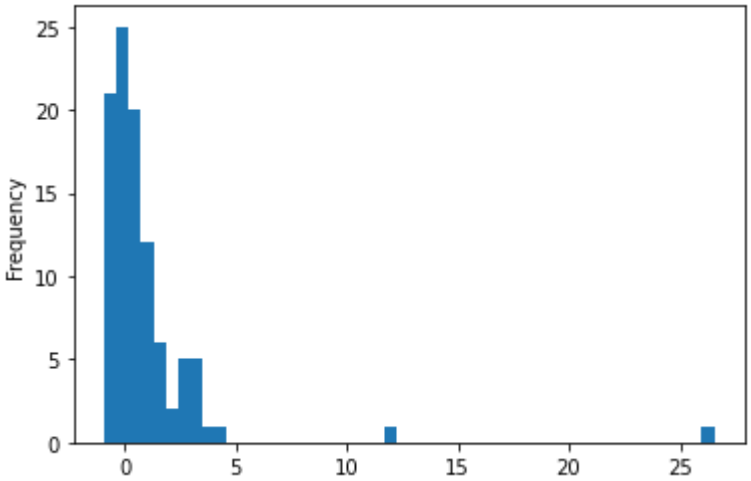


In [398]:

```
dt_t.ratio.plot(kind='hist',bins=50)
```

Out[398]:

<AxesSubplot:ylabel='Frequency'>




In [414]:

```
dt_t.describe()
```

Out[414]:

| | gender | region | age_cat | hpd | lpd | ratio |
|---|---|---|---|---|---|---|
| **count** | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 |
| **mean** | 0.560000 | 3.060000 | 3.090000 | 1.956624 | 0.542246 | 0.970339 |
| **std** | 0.498888 | 1.361965 | 1.614924 | 2.958971 | 1.554206 | 3.042101 |
| **min** | 0.000000 | 1.000000 | 1.000000 | -0.911892 | -0.851840 | -0.910634 |
| **25%** | 0.000000 | 2.000000 | 1.000000 | -0.040747 | -0.378877 | -0.311907 |
| **50%** | 1.000000 | 3.000000 | 3.000000 | 1.171115 | 0.065510 | 0.298445 |
| **75%** | 1.000000 | 4.000000 | 5.000000 | 2.978965 | 0.959595 | 1.199583 |
| **max** | 1.000000 | 5.000000 | 5.000000 | 13.545455 | 8.341463 | 26.530249 |

In [415]:

```
dt_t.corr()
```

Out[415]:

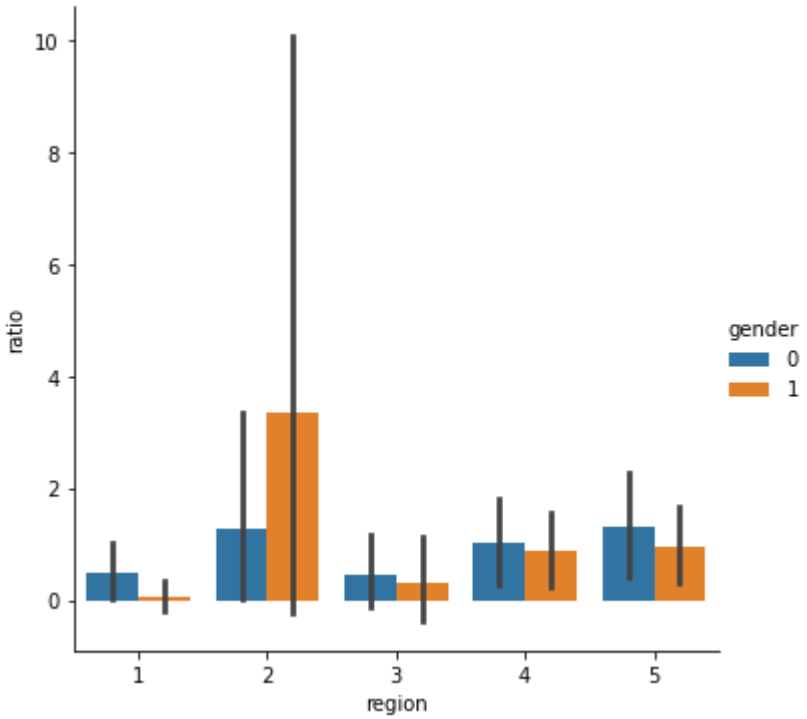
| | gender | region | age_cat | hpd | lpd | ratio |
|---|---|---|---|---|---|---|
| **gender** | 1.000000 | 0.068979 | -0.025576 | -0.086791 | -0.042548 | 0.006345 |
| **region** | 0.068979 | 1.000000 | 0.149072 | 0.213487 | 0.262055 | 0.014250 |
| **age_cat** | -0.025576 | 0.149072 | 1.000000 | 0.079863 | -0.038990 | -0.200864 |
| **hpd** | -0.086791 | 0.213487 | 0.079863 | 1.000000 | -0.033082 | 0.111894 |
| **lpd** | -0.042548 | 0.262055 | -0.038990 | -0.033082 | 1.000000 | 0.361511 |
| **ratio** | 0.006345 | 0.014250 | -0.200864 | 0.111894 | 0.361511 | 1.000000 |

In [420]:

```
import warnings
warnings.filterwarnings('ignore')
dp = sns.factorplot(x="region",y="ratio",hue='gender',data=dt_t,kind="bar")
dp
```

Out[420]:

```
<seaborn.axisgrid.FacetGrid at 0x13c50a621c0>
```




In [ ]:

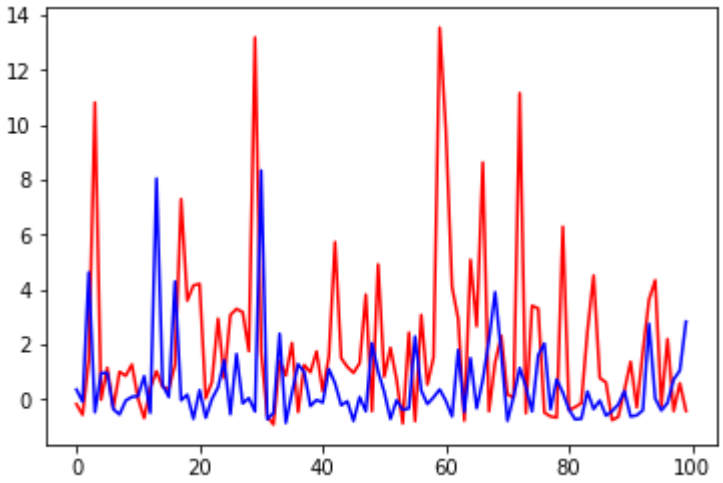
```
# 2지역 사람들의 2019->2020 구매 상승률이 가장 높다.
# 여자의 구매 비중은 2지역 사람들이 가장 높고, 남자는 지역별로 큰 차이가 없다.
# 2의 신뢰구간이 가장 높다.
```

In [429]:

```
plt.plot(dt_t['hpd'],'r')
plt.plot(dt_t['lpd'],'b')
```

Out[429]:

```
[<matplotlib.lines.Line2D at 0x13c50ce8eb0>]
```




In [ ]:

```
# high product의 변화율이 low product의 변화율보다 크다.
```

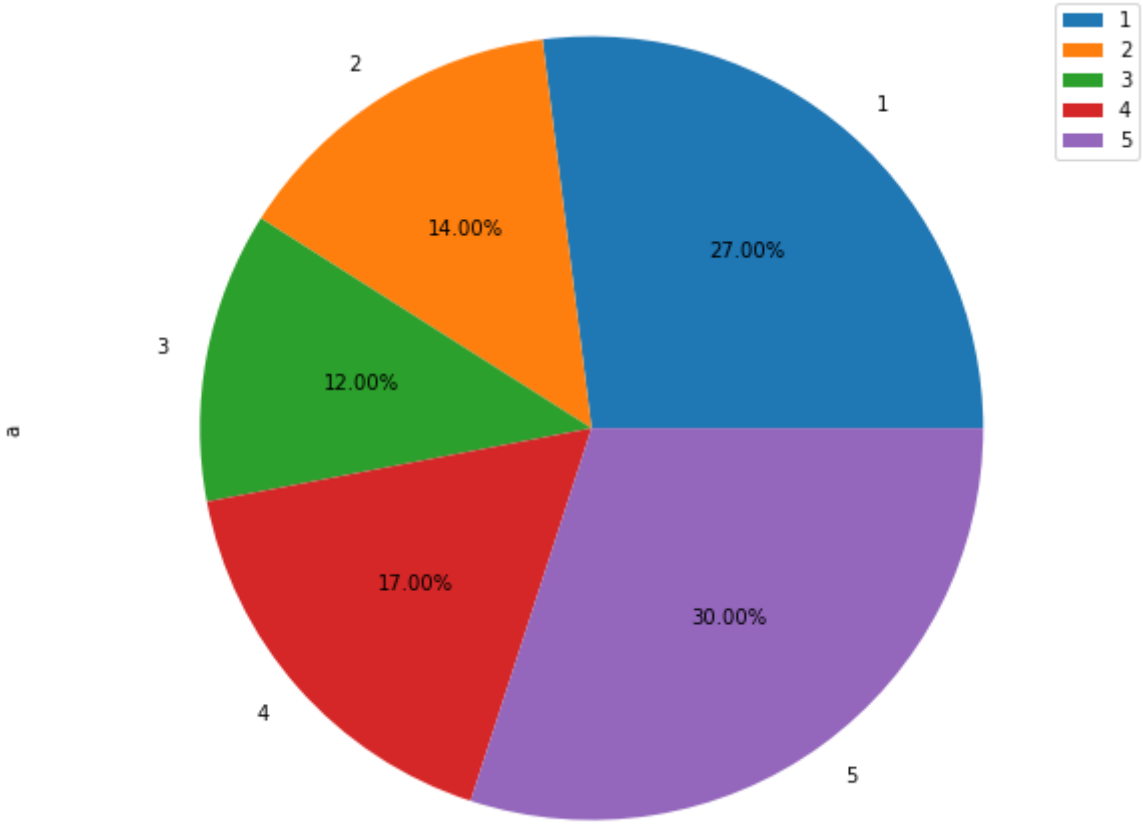
In [431]:

```
dt_t['a']=1
ac= dt_t.groupby('age_cat').sum()
ac
ac.a.plot(kind='pie',figsize=(10,8),autopct='%.2f%%', startangle=0)

plt.axis('equal')
plt.legend(labels=ac.index,loc='best')
```

Out[431]:

```
<matplotlib.legend.Legend at 0x13c50cb0b80>
```



In [ ]:

```
# age_cat 중에서 구매 비율이 가장 높은 그룹은 5그룹(60세 이상)이며,
# 그 다음으로는 1그룹(30세 이하)이다.
# 나머지 그룹 (30~60세)의  구매비율은 비슷하다.
```

In [ ]:

```

```

In [ ]:

```
# lpd- ratio / hpd-ratio는 0.36의 상관관계를 갖는다.
# age_cat 중에서 구매 비율이 가장 높은 그룹은 5그룹(60세 이상)이며,
# 그 다음으로는 1그룹(30세 이하)이다.
# 나머지 그룹 (30~60세)의  구매비율은 비슷하다.
# high product의 변화율이 low product의 변화율보다 크다.
# 2지역 사람들의 2019->2020 구매 상승률이 가장 높다.
# 여자의 구매 비중은 2지역 사람들이 가장 높고, 남자는 지역별로 큰 차이가 없다.
# 2지역(여자)의 신뢰구간이 가장 높다.
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```