packages

In [38]:
```
from IPython.display import display ,HTML
display(HTML("<style>.container {width :95% !important;}</style>"))

# 한글 그래프
import matplotlib.pyplot as plt
import platform
if platform.system() == 'Darwin': #맥
        plt.rc('font', family='AppleGothic')
elif platform.system() == 'Windows': #윈도우
        plt.rc('font', family='Malgun Gothic')
elif platform.system() == 'Linux': #리눅스 (구글 콜랩)
        #!wget "https://www.wfonts.com/download/data/2016/06/13/malgun-gothic/malgun.ttf"
        #!mv malgun.ttf /usr/share/fonts/truetype/
        #import matplotlib.font_manager as fm
        #fm._rebuild()
        plt.rc('font', family='Malgun Gothic')
plt.rcParams['axes.unicode_minus'] = False #한글 폰트 사용시 마이너스 폰트 깨짐 해결
import seaborn as sns

import pandas as pd
pd.set_option('display.max_columns',200)

#sample etc.
import numpy as np

#pca
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler

#t-test
from scipy.stats import ttest_ind


#modeling
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier , DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error


import warnings
warnings.filterwarnings('ignore')


from imblearn.over_sampling import SMOTE
import xgboost as xgb
import scipy.stats as stats
import math
from scipy.stats import chi2_contingency
from statsmodels.stats.multicomp import pairwise_tukeyhsd

## by datamanim
```

29회 ADP 복원

기계학습 (60점)

데이터 설명

- 데이터 출처 : https://www.data.go.kr/data/15094266/fileData.do (https://www.data.go.kr/data/15094266/fileData.do) 후처리
- 데이터 링크 : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p1.csv (https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p1.csv)
- 데이터 설명 : 대구도시공사_빅데이터_영구임대아파트 입주자 퇴거여부 데이터, 고유번호를 가진 계약자와 특정 아파트에 대해 매년 퇴거여부를 기록한 데이터

In [39]:
```
#encoding = ['utf-8','cp949','utf-8-sig','euc-kr','latin']
df =pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p1.csv',encoding= 'cp949')
df.head(15)
```

Out[39]:

	순번	계약구분	재계약횟수	거주개월	아파트이름	아파트ID	아파트평점	호실고유번호	층	평형대	계약자고유번호	계약서고유번호	입주연도	퇴거연도	거주연도	월세(원)	보증금(원)	대표나이	나이	성별	결혼여부	거주자수	퇴거여부
0	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2008	47100	3646000	46	33	남	미혼	3	미퇴거
1	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2009	56500	4375000	46	34	남	미혼	3	미퇴거
2	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2010	56500	4375000	46	35	남	미혼	3	미퇴거
3	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2011	69900	5408000	46	36	남	미혼	3	미퇴거
4	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2012	69900	5408000	46	37	남	미혼	3	미퇴거
5	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2013	83800	6489000	46	38	남	미혼	3	미퇴거
6	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2014	83800	6489000	46	39	남	미혼	3	미퇴거
7	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2015	105600	8177000	46	40	남	미혼	3	미퇴거
8	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2016	105600	8177000	46	41	남	미혼	3	미퇴거
9	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2017	126700	9812000	46	42	남	미혼	3	미퇴거
10	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2018	126700	9812000	46	43	남	미혼	3	미퇴거
11	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2019	152040	11774400	46	44	남	미혼	3	미퇴거
12	1	유효	10	222	강남아파트	5	7.0	14520	1	12	15468	15865	2003	NaN	2020	152040	11774400	46	45	남	미혼	3	미퇴거
13	2	해지	5	108	강남아파트	5	7.0	14546	1	12	15954	16351	2003	2012.0	2008	47100	3646000	68	55	여	미혼	1	미퇴거
14	2	해지	5	108	강남아파트	5	7.0	14546	1	12	15954	16351	2003	2012.0	2009	56500	4375000	68	56	여	미혼	1	미퇴거

1-1 계약자고유번호를 기준으로 거주연도 별 여러개의 데이터가 쌓여 있다. 각 계약자고유번호에 대해 가장 최신의 거주연도 행만 남겨라.

In [40]:
```
result = df.sort_values(['계약자고유번호','거주연도']).drop_duplicates('계약자고유번호',keep='last').reset_index(drop=True)
display(result.head(3))
print('답안\n계약자 고유번호 거주연도 기준으로 정렬하고 계약자 고유번호가 중복인 데이터를 제거한다.')
```

	순번	계약구분	재계약횟수	거주개월	아파트이름	아파트ID	아파트평점	호실고유번호	층	평형대	계약자고유번호	계약서고유번호	입주연도	퇴거연도	거주연도	월세(원)	보증금(원)	대표나이	나이	성별	결혼여부	거주자수	퇴거여부
0	12673	해지	4	88	지산5단지아파트	3	8.0	85369	6	12	1	1	2005	2012.0	2012	77300	5302000	44	35	남	기혼	2	퇴거
1	12683	해지	7	174	지산5단지아파트	3	8.0	85421	6	15	6	6	2001	2016.0	2016	48600	2144000	32	27	남	미혼	1	퇴거
2	12702	유효	10	237	지산5단지아파트	3	8.0	85576	7	15	14	14	2001	NaN	2020	81600	5598000	67	66	남	미혼	1	미퇴거

답안
계약자 고유번호 거주연도 기준으로 정렬하고 계약자 고유번호가 중복인 데이터를 제거한다.

1-2 결측치 처리

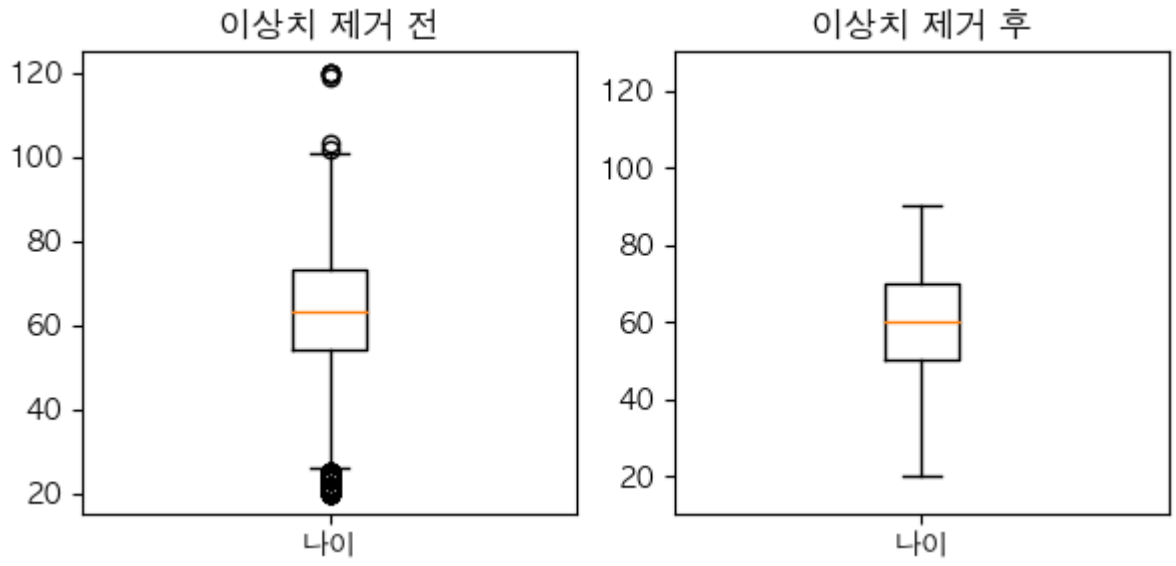
In [41]:
```
pre_df = result.copy()
dic = {'퇴거':'해지','미퇴거':'유효'}

pre_df['계약구분'] = pre_df['퇴거여부'].map(dic)
dic_apt = {'강남아파트': 7.0, '까치아파트': 10.0, '비둘기아파트': 5.0, '용지아파트': 7.0, '지산5단지아파트': 8.0}
pre_df['아파트 평점'] = pre_df['아파트 이름'].map(dic_apt)
pre_df = pre_df.drop(columns =['퇴거연도'])
display(result.isnull().sum().where(lambda x : x != 0).dropna().to_frame().rename(columns ={0:'결측치 숫자'}).astype('int'))

print('답안\n기존 데이터의 결측치는 계약구분, 아파트 평점, 퇴거연도 3가지 컬럼에서 존재한다.')
print('계약구분의 경우 퇴거여부와 1대1 매핑이 되고, 아파트 평점의 경우에도 아파트 이름에 따라 1대1 매핑됨을 확인했다. 이를 통해 결측치를 대치한다.')
print('퇴거 연도의 경우 현재 "미퇴거" 상태라면 존재하지 않는 값이기 때문에 컬럼 자체를 제거한다.')
```

	결측치 숫자
계약구분	61
아파트 평점	141
퇴거연도	6257

답안
기존 데이터의 결측치는 계약구분, 아파트 평점, 퇴거연도 3가지 컬럼에서 존재한다.
계약구분의 경우 퇴거여부와 1대1 매핑이 되고, 아파트 평점의 경우에도 아파트 이름에 따라 1대1 매핑됨을 확인했다. 이를 통해 결측치를 대치한다.
퇴거 연도의 경우 현재 "미퇴거" 상태라면 존재하지 않는 값이기 때문에 컬럼 자체를 제거한다.

1-3 이상치 처리

In [42]:
```
pre_df['나이이상치_제거'] = (pre_df['나이'] //10 *10).map(lambda x : 90 if x >=90 else x)

fig,ax = plt.subplots(1,2,figsize=(7,3))
ax[0].boxplot(pre_df['나이'])
ax[1].boxplot(pre_df['나이이상치_제거'])
ax[0].set_title('이상치 제거 전')
ax[1].set_title('이상치 제거 후')
ax[1].set_ylim(10,130)
ax[0].set_xticks([1],['나이'])
ax[1].set_xticks([1],['나이'])
plt.show()

# 추가적으로 필요하다고 생각하는 컬럼에 대해서 변경은 나름대로 판단해서 작업하시길
print('답안\n"나이"컬럼에 대해서 90세 이상의 경우 90대로 변경')
```




```
답안
"나이"컬럼에 대해서 90세 이상의 경우 90대로 변경
```

2-1 재계약 횟수의 중앙값을 기준으로 중앙값보다 크거나 같으면 '높음', 작으면 '낮음' 으로 재계약 횟수 이분 변수를 구성하시오.

In [43]:
```
med=pre_df['재계약횟수'].median()

pre_df['이분변수'] = pre_df['재계약횟수'].map(lambda x : '높음' if x >= med else '낮음')
display(pre_df['이분변수'].value_counts().to_frame())
print('답안\n중앙값은 6이다. 이를 기준으로 이분변수를 나누고 새로운 컬럼을 만든다. ')
```

이분변수	count
높음	5388
낮음	4960

```
답안
중앙값은 6이다. 이를 기준으로 이분변수를 나누고 새로운 컬럼을 만든다.
```

2-2 차원축소의 필요성을 논하고, 필요에 따라 차원을 축소하고 불필요하다면 그 근거를 논하시오.

In [44]:
```
p_df = pre_df.drop(columns = ['순번','아파트 이름','호실고유번호','계약자고유번호','계약서고유번호','대표나이','나이'])
p_df['월세(만원)'] = p_df['월세(원)']/10000
p_df['보증금(백만원)'] = p_df['보증금(원)']/1000000
p_df = p_df.drop(columns = ['월세(원)','보증금(원)'])


for col in p_df.select_dtypes(include= object):
    lb = LabelEncoder()

    p_df[col] = lb.fit_transform(p_df[col])


pca = PCA()
mm = MinMaxScaler()

pca_df = pd.DataFrame(mm.fit_transform(p_df))
pca_df.columns = p_df.columns

pca.fit(pca_df)
ratio = pd.DataFrame(pca.explained_variance_ratio_,columns =['ratio'])
ratio['cumsum'] =ratio['ratio'].cumsum()


ratio.columns = ['variance_ratio','cumsum']
ratio.index = ['PC_' + str(i+1) for i in ratio.index]
display(ratio.round(2).T)
print('답안\n순번,아파트 이름 호실 고유번호, 계약자 고유번호, 계약서 고유번호는 데이터 구분을 위한 특정 PK이기 때문에 제거한다.')
print('대표나이의 경우 2020년 기준의 나이이기 때문에 제거한다. 나이 컬럼은 이상치 제거한 컬럼이 있기 때문에 제거한다.')
print('범주형 변수에 대해서 라벨인코딩을 진행한 후 min-max 스케일링을 하고 pca를 진행한다.')
print('분산설명력의 경우 7개의 주축을 선택할 경우 92% 데이터 설명이 가능하다.')
print('전체 컬럼의 숫자가 17개로 많지 않고 데이터 설명을 위한 데이터 주축의 숫자가 많이 필요 하기 때문에 굳이 차원 축소를 하지 않겠다.')
```

	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6	PC_7	PC_8	PC_9	PC_10	PC_11	PC_12	PC_13	PC_14	PC_15	PC_16	PC_17
variance_ratio	0.34	0.20	0.15	0.07	0.06	0.05	0.04	0.02	0.02	0.02	0.01	0.0	0.0	0.0	0.0	0.0	0.0
cumsum	0.34	0.54	0.69	0.77	0.82	0.87	0.92	0.94	0.96	0.98	0.99	1.0	1.0	1.0	1.0	1.0	1.0

```
답안
순번,아파트 이름 호실 고유번호, 계약자 고유번호, 계약서 고유번호는 데이터 구분을 위한 특정 PK이기 때문에 제거한다.
대표나이의 경우 2020년 기준의 나이이기 때문에 제거한다. 나이 컬럼은 이상치 제거한 컬럼이 있기 때문에 제거한다.
범주형 변수에 대해서 라벨인코딩을 진행한 후 min-max 스케일링을 하고 pca를 진행한다.
분산설명력의 경우 7개의 주축을 선택할 경우 92% 데이터 설명이 가능하다.
전체 컬럼의 숫자가 17개로 많지 않고 데이터 설명을 위한 데이터 주축의 숫자가 많이 필요 하기 때문에 굳이 차원 축소를 하지 않겠다.
```

In [50]:
```
pd.DataFrame(pca.fit_transform(pca_df)).iloc[:,:6]
```

Out[50]:

	0	1	2	3	4	5
0	1.161364	0.020467	0.815050	0.273182	-0.277351	0.011192
1	0.534893	0.740057	0.592341	0.100884	-0.531203	0.081187
2	-0.861484	0.246685	0.569075	0.155163	-0.446783	0.093418
3	-0.843728	0.306694	1.074872	0.821956	0.142791	0.095863
4	-0.887255	0.408885	-0.295484	0.492366	-0.134473	0.223372
...	...	...	...	...	...	...
10343	-0.097853	-1.156880	0.296261	0.114296	-0.770606	-0.097658
10344	-0.095980	-1.159257	0.295959	0.110573	-0.820918	-0.231188
10345	-0.657410	-0.154507	0.435575	-0.313537	-0.281388	0.271923
10346	-0.827225	0.310886	-0.404052	-0.084006	-0.254563	-0.251011
10347	-0.783549	0.147742	0.485494	-0.589632	-0.040001	-0.175275

10348 rows × 6 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

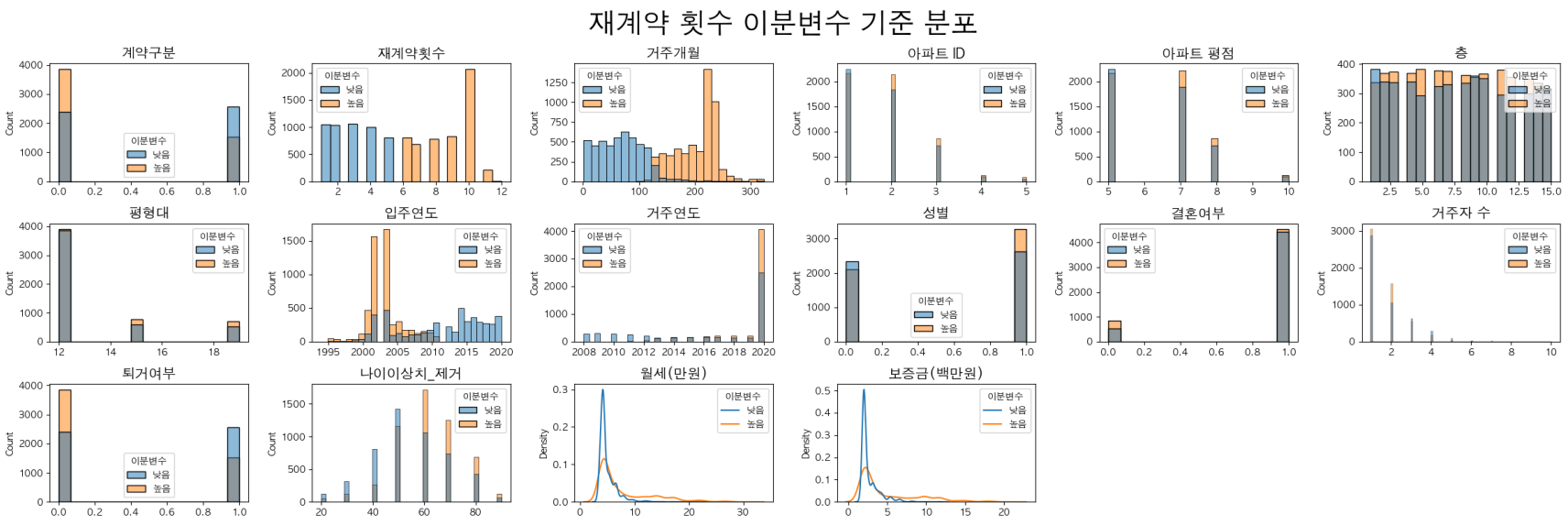
In [ ]:

In [ ]:

In [ ]:

3-1 재계약 횟수 이분변수를 기준으로 세그먼트를 구분하고 각 세그먼트의 특징을 분석하시오.

In [9]:
```
p_value_lst = []
fig,ax = plt.subplots(3,6,figsize=(24,8))
for i,axes in enumerate(ax.flat[:-2]):
    target_columns = p_df.drop(columns =['이분변수']).columns[i]
    target_df = p_df[[target_columns,'이분변수']]
    target_df['이분변수'] = target_df['이분변수'].map({0:'낮음',1:'높음'}) # 라벨인코딩시에 어떤 레이블링을 했는지 확인해야함


    s,pv = ttest_ind(target_df.query('이분변수 =="높음"')[target_columns],target_df.query('이분변수 =="낮음"')[target_columns])
    p_value_lst.append([target_columns,round(pv,5)])



    if '월세' not in target_columns and '보증금' not in target_columns:
        sns.histplot(data=target_df ,x= target_columns,hue= '이분변수',ax = axes)
        axes.set_xlabel('')
        axes.set_title(target_columns,fontsize=15)

    else:
        sns.kdeplot(data=target_df ,x= target_columns,hue= '이분변수',ax = axes,)
        axes.set_xlabel('')
        axes.set_title(target_columns,fontsize=15)

ax.flat[-1].remove()
ax.flat[-2].remove()
fig.suptitle('재계약 횟수 이분변수 기준 분포',fontsize=30)
fig.tight_layout()
plt.show()

pv_df = pd.DataFrame(p_value_lst).set_index(0).T
pv_df.index = ['p-value']

display(pv_df)

print('답안')
print('이분변수를 기준으로 각 컬럼의 분포에 대한 그래프이다. 이분변수를 기준으로 큰 차이를 보이는 것은 재계약 횟수, 거주개월, 입주연도, 월세,보증금이다.')
print('이분변수는 재계약 횟수 중앙값을 기준으로 만든것이기에 거주개월에 대해서는 어느정도 상관성을 보임을 추측 할 수 있다. 거주개월이 긴 경우는 입주연도도 길 것이다.')
print('보증금과 월세의 경우의 경우 이분변수가 높은 경우 분산이 더 크다.')
print('층을 제외하고는 모든 변수에서 유의 수준 0.05하에서 이분변수 간의 차이가 존재한다.')
print('아파트 ID와 층에 따라서는 크게 이분변수의 차이가 존재하지 않는다. 나이의 경우 이분변수가 높을 경우 평균 연령이 높다')
```



	계약구분	재계약횟수	거주개월	아파트 ID	아파트 평점	층	평형대	입주연도	거주연도	성별	결혼여부	거주자수	퇴거여부	나이이상치_제거	월세(만원)	보증금(백만원)
p-value	0.0	0.0	0.0	0.0	0.0	0.85174	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

답안
이분변수를 기준으로 각 컬럼의 분포에 대한 그래프이다. 이분변수를 기준으로 큰 차이를 보이는 것은 재계약 횟수, 거주개월, 입주연도, 월세,보증금이다.
이분변수는 재계약 횟수 중앙값을 기준으로 만든것이기에 거주개월에 대해서는 어느정도 상관성을 보임을 추측 할 수 있다. 거주개월이 긴 경우는 입주연도도 길 것이다.
보증금과 월세의 경우의 경우 이분변수가 높은 경우 분산이 더 크다.
층을 제외하고는 모든 변수에서 유의 수준 0.05하에서 이분변수 간의 차이가 존재한다.
아파트 ID와 층에 따라서는 크게 이분변수의 차이가 존재하지 않는다. 나이의 경우 이분변수가 높을 경우 평균 연령이 높다

3-2. 재계약횟수 변수를 종속변수로 하는 회귀 분석을 두 가지 이상의 방법론을 통해 수행하고 최종 모델을 결정하시오. 재계약횟수 이분변수를 종속변수로 하는 분류 분석을 두가지 이상의 방법론을 통해 수행하고 최종 모델을 결정하시오.

In [10]:
```
preprocessing_df = p_df
preprocessing_df = preprocessing_df.rename(columns ={'이분변수':'y'})

# 스케일링 안해도되는 장점있고, 변수 중요도 확인 가능한 트리류 계열끼리 비교
# 최종 결과의 경우에도 하나의 데이터 프레임으로 정리하는게 좋지 않을까 / 다양한 모델 및 모델 최적화 과정보다 답변 깔끔하게 쓰는게 더 중요 할듯

x = preprocessing_df.drop(columns =['y','재계약횟수','거주개월','거주연도','입주연도'])
reg_y = preprocessing_df['재계약횟수']
cls_y = preprocessing_df['y']

cls_rf = RandomForestClassifier(random_state =3)
cls_dc = DecisionTreeClassifier(random_state =3)
reg_rf = RandomForestRegressor(random_state =3)
reg_dc = DecisionTreeRegressor(random_state =3)



def model_(model,types,name):
    if types =='cls':
        X_train, X_test, y_train, y_test = train_test_split(x, cls_y, test_size=0.33, random_state=42)
        model.fit(X_train,y_train)
        pred = model.predict(X_test)
        acc = accuracy_score(y_test,pred)

    elif types =='reg':
        X_train, X_test, y_train, y_test = train_test_split(x, reg_y, test_size=0.33, random_state=42)
        model.fit(X_train,y_train)
        pred = model.predict(X_test)
        acc = mean_squared_error(y_test,pred)

    return {name:acc},model

lst = {}
model_lst = []
for models in [[cls_rf,'cls','randomforest_classification'],[cls_dc,'cls','decisiontree_classification'],[reg_rf,'reg','randomforest_regressor'],[reg_dc,'reg','decisiontree_regressor']]:
    model = models[0]
    types = models[1]
    name = models[2]
    acc,model = model_(model,types,name)
    lst.update(acc)
    model_lst.append(model)

models = pd.DataFrame(lst.items(),columns = ['model','test_set_accuracy [reg : mse]']).set_index('model')
display(models.round(2))
print('답안\n')
print('이분변수는 재계약 횟수를 기준으로 만들었고 재계약 횟수는 거주개월과 거주연도 입주연도와 높은 상관성을 띄기에 이 컬럼들을 제거하고 모델을 생성한다.')
print('모델의 변수중요도를 확인하기 위해 트리류 모델들로 비교한다. 회귀모델과 분류 모델 각 경우 랜덤포레스트와 결정트리 모델에 대해 학습하고 결과를 비교한다.')
print('분류의 경우 정확도로 모델을 평가하고 회귀모델의 경우 평균제곱오차를 통해 모델을 비교한다. 정확도는 숫자가 클수록 좋은 모델이고')
print('mse는 그 값이 작을 수록 모델 오차가 작다고 판단 할 수 있다. 67%의 데이터로 학습하고, 33%의 모델로 평가했을 때')
print("분류, 회귀 모두 랜덤포레스트 모델이 더 좋은 성능을 가졌다.")
print('의사결정나무에 비해 랜덤포레스트 모델은 많은 앙상블 과정을 거치기에 과적합이 적어서 이런 결과가 나온것으로 보인다.')
```

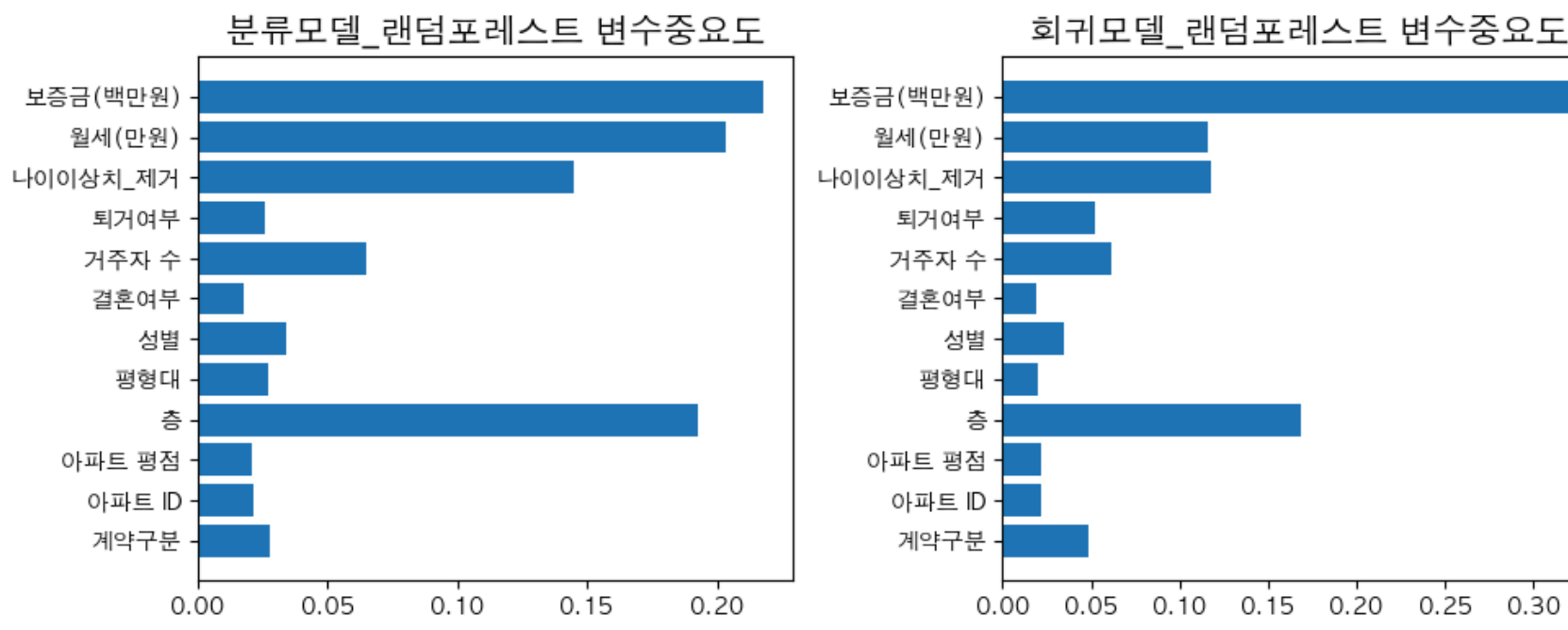
model	test_set_accuracy [reg : mse]
randomforest_classification	0.71
decisiontree_classification	0.69
randomforest_regressor	7.22
decisiontree_regressor	11.03

답안

이분변수는 재계약 횟수를 기준으로 만들었고 재계약 횟수는 거주개월과 거주연도 입주연도와 높은 상관성을 띄기에 이 컬럼들을 제거하고 모델을 생성한다.
모델의 변수중요도를 확인하기 위해 트리류 모델들로 비교한다. 회귀모델과 분류 모델 각 경우 랜덤포레스트와 결정트리 모델에 대해 학습하고 결과를 비교한다.
분류의 경우 정확도로 모델을 평가하고 회귀모델의 경우 평균제곱오차를 통해 모델을 비교한다. 정확도는 숫자가 클수록 좋은 모델이고
mse는 그 값이 작을 수록 모델 오차가 작다고 판단 할 수 있다. 67%의 데이터로 학습하고, 33%의 모델로 평가했을 때
분류, 회귀 모두 랜덤포레스트 모델이 더 좋은 성능을 가졌다.
의사결정나무에 비해 랜덤포레스트 모델은 많은 앙상블 과정을 거치기에 과적합이 적어서 이런 결과가 나온것으로 보인다.

3-3 최종 채택한 모델에서 각각 유의하게 작용하는 변수를 확인 하고 설명하시오

In [17]:
```
fig,ax = plt.subplots(1,2,figsize=(10,4))
ax[0].barh(model_lst[0].feature_names_in_,model_lst[0].feature_importances_)
ax[1].barh(model_lst[2].feature_names_in_,model_lst[2].feature_importances_)

ax[0].set_title('분류모델_랜덤포레스트 변수중요도',fontsize=15)
ax[1].set_title('회귀모델_랜덤포레스트 변수중요도',fontsize=15)
fig.tight_layout()
plt.show()

print('답안\n')
print('분류모델의 경우 보증금, 월세, 층 순서로 중요도를 보였고')
print('회귀 모델의 경우 보증금, 층, 나이이상치 제거 순서로 변수 중요도를 보였다')
```



답안

분류모델의 경우 보증금, 월세, 층 순서로 중요도를 보였고
회귀 모델의 경우 보증금, 층, 나이이상치 제거 순서로 변수 중요도를 보였다

3-4 해당 데이터 분석결과로 얻을 수 있는 점 제시

In [11]:
```
r= p_df[['이분변수','월세(만원)','보증금(백만원)','나이이상치_제거','입주연도']].groupby(['이분변수']).agg(['mea
n']).round(2)
r.index = ['낮음','높음']
display(r)
print('재계약 횟수의 중앙값을 기준으로 세그먼트를 나눠 분석해 보았다.')
print('층수의 경우 이분변수간 통계적으로 유의한 차이를 보이지 않았지만, 모델에서는 상위 구분을 위한 중요도를 보였다.')
print('재계약 횟수가 높은 경우 월세와 보증금의 평균값이 평균 2배 높았고 대표 거주인의 나이도 있고 7살 많았다.')

# 뭔가 더 인사이트가 한두줄 더 들어가야 할 것 같습니다 ㅎㅎ;
```

	월세(만원)	보증금(백만원)	나이이상치_제거	입주연도
	mean	mean	mean	mean
낮음	4.95	2.67	54.57	2011.27
높음	7.72	4.86	61.38	2003.25

재계약 횟수의 중앙값을 기준으로 세그먼트를 나눠 분석해 보았다.
층수의 경우 이분변수간 통계적으로 유의한 차이를 보이지 않았지만, 모델에서는 상위 구분을 위한 중요도를 보였다.
재계약 횟수가 높은 경우 월세와 보증금의 평균값이 평균 2배 높았고 대표 거주인의 나이도 있고 7살 많았다.

데이터 설명

- 데이터 출처 : https://www.kaggle.com/datasets/pschale/mlb-pitch-data-20152018 (https://www.kaggle.com/datasets/pschale/mlb-pitch-data-20152018) 데이터를 후처리
- 데이터 링크 : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p2_.csv (https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p2_.csv)
- 데이터 설명 :

A 야구구단의 시합 결과중 일부를 나타낸다.
각 행은 하나의 경기를 의미(game_id)하며 9회차(= 9이닝) 동안 1번타자, 2번타자의 출루 정보를 의미한다.
각 컬럼에 대한 설명은 아래 이미지와 같으며, value값의 index는 이미지의 '수치 의미' 열을 통해 확인 할 수 있다.

	컬럼명	의미 (회 == 이닝)	수치 의미
타자행동	a1_1	1회 첫 타자 행동	1 : '1루타' 2 : '2루타' 3 : '3루타' 4 : '홈런' 5 : '삼진 제외 모든 아웃' 6 : '볼넷' 7 : '삼진' 8 : '몸에 맞는 공' 9 : '희생번트'
	a1_2	1회 둘째 타자 행동	
	a2_1	2회 첫 타자 행동	
	a2_2	2회 둘째 타자 행동	
	a3_1	3회 첫 타자 행동	
	a3_2	3회 둘째 타자 행동	
	a4_1	4회 첫 타자 행동	
	a4_2	4회 둘째 타자 행동	
	a5_1	5회 첫 타자 행동	
	a5_2	5회 둘째 타자 행동	
	a6_1	6회 첫 타자 행동	
	a6_2	6회 둘째 타자 행동	
	a7_1	7회 첫 타자 행동	
	a7_2	7회 둘째 타자 행동	
	a8_1	8회 첫 타자 행동	
	a8_2	8회 둘째 타자 행동	
	a9_1	9회 첫 타자 행동	
	a9_2	9회 둘째 타자 행동	
득 점	b1	1회 총 득점수	
	b2	2회 총 득점수	
	b3	3회 총 득점수	
	b4	4회 총 득점수	
	b5	5회 총 득점수	
	b6	6회 총 득점수	
	b7	7회 총 득점수	
	b8	8회 총 득점수	
	b9	9회 총 득점수	

4-1 각 회차별로 1번 타자의 출루 (1,2,3루타와 사사구(볼넷, 몸에맞는공))가 있는 경우에 대해 득점이 발생 했는지 확인하고자 한다. 이를 위한 전처리를 수행하라. (단, 첫 번째 혹은 두 번째 타자가 홈런을 친 경우 해당 회차 데이터는 제외한다.)

조건1 : 득점여부를 범주형 종속변수로 한다. (1점이상 득점 :1, 무득점 :0)
조건2 : 각 회차 2번 타자의 데이터는 원핫 인코딩한다.
조건3 : 학습에 적절하지 않은 데이터는 제외한다.

제가 이해하고 의도한 29회 4-1번 문제에 대한 가이드 입니다. (4-1) 전처리 가이드

1. 첫타자와 둘째타자의 각 이닝별 출루현황, 해당 이닝의 득점수를 3columns의 데이터로 변환
2. 정수형 값이 아닌 데이터 행를 제거
3. 홈런 (4)값을 가지는 행을 제거
4. 이닝별 점수를 binary로 변환 (득점 있으면 1 ,없으면 0)
5. 둘째 타자 출루 결과를 one-hot encoding

In [26]:
```
df =pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p2_.csv')

lst = []
for i in range(1,10):
    each_innings_df = df[['game_id',f'a{i}_1',f'a{i}_2',f'b{i}']].reset_index(drop=True)
    each_innings_df.columns =['game_id','first_hitter','second_hitter','inning_score']
    each_innings_df.loc[:,'inning'] = i
    lst.append(each_innings_df)

t = pd.concat(lst).reset_index(drop=True)
tt = t[t.first_hitter.map(lambda x : str(x).isdigit())].reset_index(drop=True)
tt.first_hitter =tt.first_hitter.astype('int')
tt.second_hitter =tt.second_hitter.astype('int')

tts = tt[(tt.first_hitter != 4) & (tt.second_hitter != 4)].reset_index(drop=True)
tts.inning_score =tts.inning_score.map(lambda x : 1 if x >0 else 0)

sec = pd.get_dummies(tts.second_hitter.astype('str'))
sec.columns = ['second_hitter_' + str(x) for x in sec.columns]

g = pd.concat([tts[['game_id','first_hitter','inning_score','inning']],sec],axis=1)
result= g[['game_id','inning', 'first_hitter', 'second_hitter_1','second_hitter_2', 'second_hitter_3', 'second_
hitter_5','second_hitter_6', 'second_hitter_7', 'second_hitter_8','second_hitter_9','inning_score']]
result['first_hitter'] = result['first_hitter'].map(lambda x : 1 if x in [1,2,3,6,8] else 0 )
result.head()
```

Out[26]:

	game_id	inning	first_hitter	second_hitter_1	second_hitter_2	second_hitter_3	second_hitter_5	second_hitter_6	second_hitter_7	sec
0	201900016	1	0	False	False	False	True	False	False	
1	201900103	1	0	False	False	False	False	True	False	
2	201900112	1	0	False	False	False	False	False	True	
3	201900131	1	0	True	False	False	False	False	False	
4	201900141	1	1	False	False	False	True	False	False	

4-2 4-1 데이터에 대해 Logistic Regression을 적용하고 2번타자의 희생번트 여부에 대한 회귀 계수 검정을 하라

In [32]:
```
x = result.drop(columns =['game_id','inning_score','inning'])
y = result['inning_score']

dic ={False :0 , True :1}
for col in x.columns:
    x[col] = x[col].map(lambda x : dic[x] if x in [False, True] else x)
import statsmodels.api as sm
model = sm.Logit(y,x)
res = model.fit(disp=False)
display(res.summary().tables[1])
print('답안\n2번타자의 희생번트의 경우 second_hitter_9 컬럼의 p-value를 보면된다. 이는 0으로 유의수준 0.05하에서 귀무가설을 기각한다.')
print('즉 해당 변수는 종속 변수에 영향을 준다')
```

	coef	std err	z	P>|z|	[0.025	0.975]
first_hitter	1.8849	0.138	13.618	0.000	1.614	2.156
second_hitter_1	-0.5886	0.146	-4.019	0.000	-0.876	-0.302
second_hitter_2	0.2610	0.251	1.041	0.298	-0.230	0.753
second_hitter_3	2.0157	1.061	1.900	0.057	-0.064	4.095
second_hitter_5	-2.0612	0.122	-16.887	0.000	-2.300	-1.822
second_hitter_6	-0.6097	0.200	-3.047	0.002	-1.002	-0.218
second_hitter_7	-2.5450	0.181	-14.095	0.000	-2.899	-2.191
second_hitter_8	-0.4915	0.579	-0.849	0.396	-1.626	0.643
second_hitter_9	-0.9835	0.273	-3.607	0.000	-1.518	-0.449

답안
2번타자의 희생번트의 경우 second_hitter_9 컬럼의 p-value를 보면된다. 이는 0으로 유의수준 0.05하에서 귀무가설을 기각한다.
즉 해당 변수는 종속 변수에 영향을 준다

4-3 SMOTE (random_state =0 지정)를 적용하여 data imbalance를 해결하라

In [33]:
```
smote = SMOTE(sampling_strategy='auto', random_state=0)
X_resampled, y_resampled = smote.fit_resample(x, y)

result = pd.concat([y.value_counts(),y_resampled.value_counts()],axis=1)
result.columns = ['smote 이전 데이터 분포','smote 이후 데이터 분포']
result.index = ['레이블 0 ','레이블 1']
display(result)
print('풀이\nsmote를 통해 레이블 1의 숫자에 맞췄다.')
```

	smote 이전 데이터 분포	smote 이후 데이터 분포
레이블 0	1096	1096
레이블 1	511	1096

풀이
smote를 통해 레이블 1의 숫자에 맞췄다.

4-4 4-3 구성 데이터에 Logistic Regression을 적용하고 결과를 분석하라

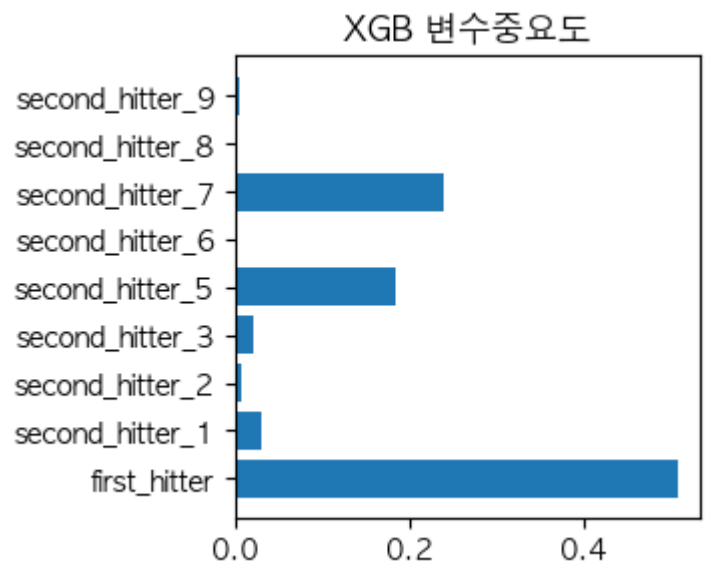
In [36]:
```
# 문제 표현에 따라 다르게 접근 할듯 / 4-2와의 비교 or 그냥 분류 모델 결과 확인
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.33, random_state=1,st
ratify=y_resampled)
model = sm.Logit(y_train.reset_index(drop=True), X_train.reset_index(drop=True))
result = model.fit(disp=False)
display(result.summary().tables[1])
acc =accuracy_score(y_test.reset_index(drop=True),result.predict(X_test).map(lambda x : 1 if x>=0.5 else 0))
print('정확도 : ',acc)
```

	coef	std err	z	P>|z|	[0.025	0.975]
first_hitter	2.0028	0.139	14.417	0.000	1.731	2.275
second_hitter_1	0.0387	0.154	0.251	0.802	-0.263	0.340
second_hitter_2	0.8552	0.246	3.471	0.001	0.372	1.338
second_hitter_3	2.2135	1.053	2.103	0.035	0.151	4.276
second_hitter_5	-1.4725	0.115	-12.814	0.000	-1.698	-1.247
second_hitter_6	0.4059	0.213	1.906	0.057	-0.011	0.823
second_hitter_7	-1.4832	0.161	-9.231	0.000	-1.798	-1.168
second_hitter_8	-0.0091	0.612	-0.015	0.988	-1.209	1.191
second_hitter_9	-0.2782	0.306	-0.910	0.363	-0.878	0.321

```
정확도 :  0.712707182320442
```

4-5 4-3 구성 데이터에 XGB 적용하고 결과를 분석하라

In [24]:
```
model = xgb.XGBClassifier( random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

plt.figure(figsize=(3,3))
plt.barh(model.feature_names_in_,model.feature_importances_)
plt.title('XGB 변수중요도')
plt.show()
print("정확도는", round(accuracy*100,1),'% 이다.\n첫번째 타자의 출루가 가장큰 변수 중요도를 가지며 그다음으로 2번째
타자의 삼진과 그외 아웃들이 순서를 따른다')
```




```
정확도는 69.6 % 이다.
첫번째 타자의 출루가 가장큰 변수 중요도를 가지며 그다음으로 2번째 타자의 삼진과 그외 아웃들이 순서를 따른다
```

통계 (40점)

5. 제품 A의 불량률은 0.03이다. 25개의 제품을 뽑았을 때 3개가 불량일 확률을 구하시오. (소수점 다섯 째 자리에서 반올림)

In [80]:
```
# 이항분포 코드 이용한 풀이
n = 25 ;k = 3;p = 0.03
binomial_prob = stats.binom.pmf(k, n, p)
result = round(binomial_prob, 5)


# 확률 풀이
p_defective = 0.03 ; p_good = 0.97 ; num_defective = 3
num_good = 25 - num_defective

probability = (p_defective ** num_defective) * (p_good ** num_good) * math.comb(25, 3)
rounded_probability = round(probability, 5)


print(f'답안\n22개는 정상(확률 0.97) 3개는 불량(확률 0.03)인 이항분포 문제이다.')
print(f'이항분포의 확률 질량함수는 n :시행횟수 , k :불량품갯수 , p : 불량률 일때')
print(f'nCk * (p**k) * (1-p)**(n-k)로 계산할 수 있다. 그 결과는 {rounded_probability}이다')
```

```
답안
22개는 정상(확률 0.97) 3개는 불량(확률 0.03)인 이항분포 문제이다.
이항분포의 확률 질량함수는 n :시행횟수 , k :불량품갯수 , p : 불량률 일때
nCk * (p**k) * (1-p)**(n-k)로 계산할 수 있다. 그 결과는 0.03177이다
```

6. C사 생산 제품 1000개 중 양품이 600개, D사 생산 제품 500개 중 양품이 200개 이다. 두 회사의 양품률에 차이가 있는지 검정하여라.

In [52]:
```
# sol1. Z검정 p-value
p1 = 0.6;p2 = 0.4
n1 = 1000;n2 = 500
SE = math.sqrt((p1 * (1 - p1) / n1) + (p2 * (1 - p2) / n2)) # 표준 오차 계산
Z = (p1 - p2) / SE # Z-점수 계산
p_value = 2 * (1 - stats.norm.cdf(abs(Z))) # p-value 계산

# sol2. Z 스코어로 비교
Z_critical = abs(Z)  # Z-점수가 1.96보다 크면 95% 신뢰수준에서 유의미한 차이 / 문제에서 주어지는 경우


# sol3 카이제곱 독립성 검정
observed = [[600, 400], [200, 300]]
chi2, p_value, dof, expected = chi2_contingency(observed)

# 유의수준 0.05하에 차이 존재
print('답안\n두 양품률 간의 차이를 검정하기 위해 독립적인 두개의 비율을 카이제곱 독립성 검정을 통해 검정함')
print('귀무가설은 "C사와 D사의 양품률 간에 차이가 없다." , 대립가설은 "C사와 D사의 양품률 간에 차이가 있다로 둔다."')
print(f"C/D사의 양품률은 각각 0.6과 0.4이며 양품과 불량품의 숫자를 각각 구해 카이 제곱 검정시 p-value는 {p_value}를 가진다.")
print('유의수준 0.05하에 귀무가설을 기각한다. 즉 c사와 d사의 양품률은 차이가 있다')
```

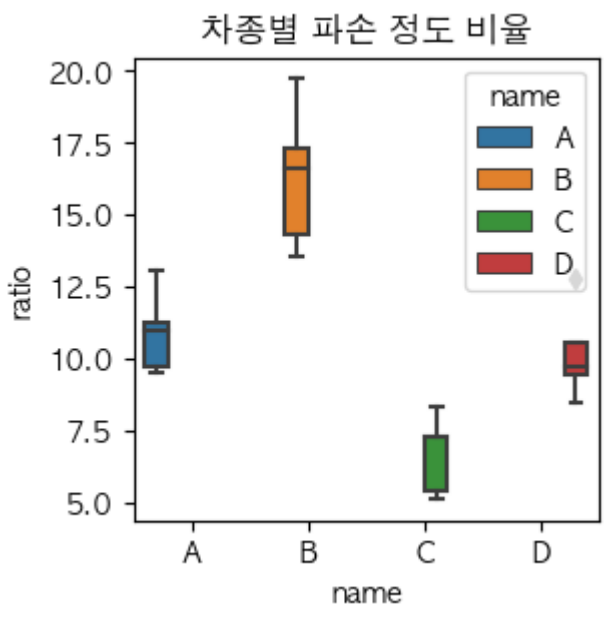
```
답안
두 양품률 간의 차이를 검정하기 위해 독립적인 두개의 비율을 카이제곱 독립성 검정을 통해 검정함
귀무가설은 "C사와 D사의 양품률 간에 차이가 없다." , 대립가설은 "C사와 D사의 양품률 간에 차이가 있다로 둔다."
C/D사의 양품률은 각각 0.6과 0.4이며 양품과 불량품의 숫자를 각각 구해 카이 제곱 검정시 p-value는 3.7481920789578267e-13를 가진다.
유의수준 0.05하에 귀무가설을 기각한다. 즉 c사와 d사의 양품률은 차이가 있다
```

7. 아래 데이터는 a,b,c,d 네 차종 각각 5회 실험 시 범퍼 파손 정도 이다. (단, 각 모집단은 정규분포를 따르며 모집단 간 등분산성을 가정한다.)

dataurl : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p7.csv (https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p7.csv)

7-1. 각 차종 별 범퍼 파손의 정도에 차이가 유의한지 검정하라.

In [57]:
```
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/29/p7.csv')
plt.figure(figsize=(3,3))
sns.boxplot(x ='name', y='ratio',data =df,hue='name')
plt.title('차종별 파손 정도 비율')
plt.show()
# 일원 분산분석 수행
f_statistic, p_value = stats.f_oneway(df.query("name =='A'")['ratio'],
                                      df.query("name =='B'")['ratio'],
                                      df.query("name =='C'")['ratio'],
                                      df.query("name =='D'")['ratio'])

print('답안\n데이터숫자가 적지만 정규분포와 등분산에 대한 가정을 조건으로 주었기 때문에 모수적 검정인 일원 분산분석을 통해서 그룹간 차이가 존재하는지 확인한다.')
print(f'일원분산분석 p-value는 {p_value}로 차이가 유의하다고 볼 수 있다')
```



답안
데이터숫자가 적지만 정규분포와 등분산에 대한 가정을 조건으로 주었기 때문에 모수적 검정인 일원 분산분석을 통해서 그룹간 차이가 존재하는지 확인한다.
일원분산분석 p-value는 2.8174779556216382e-06로 차이가 유의하다고 볼 수 있다

7-2 귀무가설을 채택한다면 그 의미를 해석하고, 귀무가설을 기각하였다면 사후분석을 시행하라.

In [71]:
```
posthoc = pairwise_tukeyhsd(df['ratio'], df['name'], alpha=0.05)
display(posthoc.summary())
print('답안\n귀무가설은 "4개 그룹의 차이가 존재하지 않는다"이고 대립가설은 "4개 그룹들 중 차이가 있는 그룹이 존재한다"이다')
print('7-1에서 유의수준 0.05하에서 귀무가설을 기각하는 것을 보였다. tukey 사후분석을 진행하면 위의 데이터프레임과 같은 결과를 보인다.')
print('결과적으로 A와 D그룹을 비교했을때 p-value는 0.92로 귀무가설을 기각하지 못하고 나머지 모든 그룹쌍 사이에는 차이가 존재하는 것을 확인할 수 있다.')
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
A	B	5.3934	0.001	2.1785	8.6083	True
A	C	-4.2156	0.0085	-7.4305	-1.0007	True
A	D	-0.7086	0.9207	-3.9235	2.5063	False
B	C	-9.609	0.0	-12.8239	-6.3941	True
B	D	-6.102	0.0003	-9.3169	-2.8871	True
C	D	3.507	0.0302	0.2921	6.7219	True

답안
귀무가설은 "4개 그룹의 차이가 존재하지 않는다"이고 대립가설은 "4개 그룹들 중 차이가 있는 그룹이 존재한다"이다
7-1에서 유의수준 0.05하에서 귀무가설을 기각하는 것을 보였다. tukey 사후분석을 진행하면 위의 데이터프레임과 같은 결과를 보인다.
결과적으로 A와 D그룹을 비교했을때 p-value는 0.92로 귀무가설을 기각하지 못하고 나머지 모든 그룹쌍 사이에는 차이가 존재하는 것을 확인할 수 있다.

8. L1,L2,L3 세 개의 생산라인에서 각각 13%, 37%, 50%를 생산하며 각각 1.1% , 2.1%, 3.3% 불량률을 갖는다. 불량 제품이 나왔을 때 L1 라인에서 생산되었을 확률을 구하시오. (소수점 둘째자리에서 반올림)

In [85]:
```
# 주어진 확률
l1 = 0.13; l2 = 0.37; l3 = 0.50
ratio_l1 = 0.011; ratio_l2 = 0.021; ratio_l3 = 0.033

# 불량 제품이 나올 확률
ratio_t = ratio_l1 * l1 + ratio_l2 * l2 + ratio_l3 * l3

# 베이즈 정리를 사용하여 P(L1|불량) 계산
l1_ratio_t = (ratio_l1 * l1) / ratio_t
result = round(l1_ratio_t, 2)
print("답안\n베이즈 정리를 이용하면 다음과 같은 확률이 나온다", result)
```

```
답안
베이즈 정리를 이용하면 다음과 같은 확률이 나온다 0.06
```

created by datamanim [web-link (https://www.datamanim.com)](https://www.datamanim.com)

끝. 10페이지