In [1]:
```
#必要なライブラリをインポート
import pandas as pd
import numpy as np

#東京都23区の物件を個別に保存しておいたものを読み込み、結合 2019.2.5
df_minato = pd.read_csv('suumo_minatoku.csv', sep='\t', encoding='utf-16')




df_toyosima = pd.read_csv('suumo_toyoshima.csv', sep='\t', encoding='utf-16')
df_nakano = pd.read_csv('suumo_nakanoku.csv', sep='\t', encoding='utf-16')
df_ootaku = pd.read_csv('suumo_ootaku.csv', sep='\t', encoding='utf-16')
df_bunkyoku = pd.read_csv('suumo_bunkyo.csv', sep='\t', encoding='utf-16')
df_taitoku = pd.read_csv('suumo_taitoku.csv', sep='\t', encoding='utf-16')
df_sinjyuku = pd.read_csv('suumo_sinjyuku.csv', sep='\t', encoding='utf-16')
df_shinagawa = pd.read_csv('suumo_shinagawa.csv', sep='\t', encoding='utf-16')
df_cyuoku = pd.read_csv('suumo_cyuoku.csv', sep='\t', encoding='utf-16')
df_shibuya = pd.read_csv('suumo_shibuya.csv', sep='\t', encoding='utf-16')
```

In [2]:
```
#風呂なし　トイレ共同、新しいカリムを増やす
df_minato["風呂なし　トイレ共同"]=0
df_toyosima["風呂なし　トイレ共同"]=0
df_nakano["風呂なし　トイレ共同"]=0
df_ootaku["風呂なし　トイレ共同"]=0
df_bunkyoku["風呂なし　トイレ共同"]=0
df_taitoku["風呂なし　トイレ共同"]=0
df_sinjyuku["風呂なし　トイレ共同"]=0
df_shinagawa["風呂なし　トイレ共同"]=0
df_cyuoku["風呂なし　トイレ共同"]=0
df_shibuya["風呂なし　トイレ共同"]=0
```

In [ ]:

In [3]:
```
df_bunkyoku.loc[df_bunkyoku["マンション名"]=="光風荘","風呂なし　トイレ共同"]=1
```

In [4]:
```
df_nakano.loc[df_nakano["マンション名"]=="ひかり荘","風呂なし　トイレ共同"]=1
```

In [5]:
```
df_sinjyuku.loc[df_sinjyuku["マンション名"]=="樟川荘","風呂なし　トイレ共同"]=1
```

In [6]:
```
df_toyosima.loc[df_toyosima["マンション名"]=="東京メトロ有楽町線 要町駅 2階建 築56年","風呂なし　トイレ共同"]=1
```

In [7]:
```
df_ootaku.loc[df_ootaku["マンション名"]=="佐々木ハウス","風呂なし　トイレ共同"]=1
```

In [8]:
```
df_ootaku.loc[df_ootaku["マンション名"]=="親和荘","風呂なし　トイレ共同"]=1
```

In [9]:
```
df_taitoku.loc[df_taitoku["マンション名"]=="橋本荘","風呂なし　トイレ共同"]=1
```

In [10]:
```
df_bunkyoku.loc[df_bunkyoku["マンション名"]=="Ｎ・本郷荘","風呂なし　トイレ共同"]=1
```

In [11]:
```
df_bunkyoku.loc[df_bunkyoku["マンション名"]=="東京メトロ南北線 東大前駅 2階建 築44年","風呂なし　トイレ共同"]=1
```

In [12]:
```
df_nakano.loc[df_nakano["マンション名"]=="ひかり荘","風呂なし　トイレ共同"]=1
```

In [13]:
```
#各標本を１０００に増やす
df_bunkyoku_F1= df_bunkyoku[df_bunkyoku["マンション名"]=="Ｎ・本郷荘"].sample(n=1000, random_state=1,replac
df_nakano_F2=df_nakano[df_nakano["マンション名"]=="ひかり荘"].sample(n=1000, random_state=1,replace=True)
df_sinjyuku_F3= df_sinjyuku[df_sinjyuku["マンション名"]=="樟川荘"].sample(n=1000, random_state=1,replace=Tr
df_toyosima_F4= df_toyosima[df_toyosima["マンション名"]=="東京メトロ有楽町線 要町駅 2階建 築56年"].sample(n=100
df_ootaku_F5= df_ootaku[df_ootaku["マンション名"]=="佐々木ハウス"].sample(n=1000, random_state=1,replace=Tru
df_ootaku_F6= df_ootaku[df_ootaku["マンション名"]=="親和荘"].sample(n=1000, random_state=1,replace=True)
df_taitoku_F7=  df_taitoku[df_taitoku["マンション名"]=="橋本荘"].sample(n=1000, random_state=1,replace=True
df_bunkyoku_F8= df_bunkyoku[df_bunkyoku["マンション名"]=="Ｎ・本郷荘"].sample(n=1000, random_state=1,replac
df_bunkyoku_F9=  df_bunkyoku[df_bunkyoku["マンション名"]=="東京メトロ南北線 東大前駅 2階建 築44年"].sample(n=10
df_nakano_F10= df_nakano[df_nakano["マンション名"]=="ひかり荘"].sample(n=1000, random_state=1,replace=True)
```

In [ ]:

```
```

In [ ]:

```
```

In [14]:

```
df = pd.concat([df_minato,df_bunkyoku,df_nakano,df_sinjyuku,df_toyosima,df_ootaku,df_taitoku,df_shinagaw
                ,df_nakano_F2,df_sinjyuku_F3,df_toyosima_F4,df_ootaku_F5,df_ootaku_F6,df_taitoku_F7,df_bu
                ,df_bunkyoku_F9,df_nakano_F10])
```

In [15]:

```
df.drop(['Unnamed: 0'], axis=1, inplace=True)
df.drop(['詳細URL'], axis=1, inplace=True)
```

In [16]:

```
splitted1 = df['立地1'].str.split(' 歩', expand=True)
splitted1.columns = ['立地11', '立地12']
splitted2 = df['立地2'].str.split(' 歩', expand=True)
splitted2.columns = ['立地21', '立地22']
splitted3 = df['立地3'].str.split(' 歩', expand=True)
splitted3.columns = ['立地31', '立地32']
```

In [18]:

```
df.drop(['立地1','立地2','立地3'], axis=1, inplace=True)
```

In [19]:

```
df = df.dropna(subset=['賃料料'])
```

In [ ]:

```
```

In [20]:

```
splitted4 = df['敷/礼/保証/敷引,償却'].str.split('/', expand=True)
splitted4.columns = ['敷金', '礼金']
```

In [21]:

```
#分割したカラムを結合
df = pd.concat([df, splitted1, splitted2, splitted3, splitted4], axis=1)
```

In [22]:

```
df.drop(['敷/礼/保証/敷引,償却'], axis=1, inplace=True)
```

In [23]:
```
df['賃料料'] = df['賃料料'].str.replace(u'万円', u'')
df['敷金'] = df['敷金'].str.replace(u'万円', u'')
df['礼金'] = df['礼金'].str.replace(u'万円', u'')
df['管理費'] = df['管理費'].str.replace(u'円', u'')
df['築年数'] = df['築年数'].str.replace(u'新築', u'0') #新築は築年数0年とする
df['築年数'] = df['築年数'].str.replace(u'築', u'')
df['築年数'] = df['築年数'].str.replace(u'年', u'')
df['専有面積'] = df['専有面積'].str.replace(u'm', u'')
df['立地12'] = df['立地12'].str.replace(u'分', u'')
df['立地22'] = df['立地22'].str.replace(u'分', u'')
df['立地32'] = df['立地32'].str.replace(u'分', u'')
```

In [ ]:
```
```

In [24]:
```
df['管理費'] = df['管理費'].replace('-',0)
df['敷金'] = df['敷金'].replace('-',0)
df['礼金'] = df['礼金'].replace('-',0)
```

In [ ]:
```
```

In [25]:
```
#文字列から数値に変換
df['賃料料'] = pd.to_numeric(df['賃料料'])
df['管理費'] = pd.to_numeric(df['管理費'])
df['敷金'] = pd.to_numeric(df['敷金'])
df['礼金'] = pd.to_numeric(df['礼金'])
df['築年数'] = pd.to_numeric(df['築年数'])
df['専有面積'] = pd.to_numeric(df['専有面積'])
df['立地12'] = pd.to_numeric(df['立地12'])
df['立地22'] = pd.to_numeric(df['立地22'])
df['立地32'] = pd.to_numeric(df['立地32'])
```

In [26]:
```
df['賃料料'] = df['賃料料'] * 10000
df['敷金'] = df['敷金'] * 10000
df['礼金'] = df['礼金'] * 10000
```

In [27]:
```
#set(df['敷金']/df['賃料料'])#0,0.5,1.0,1.5,2.0,3.0
```

In [ ]:
```

```

In [28]:
```
#set(df['礼金']/df['賃料料']) #0,0.5,1.0,1.5,2.0,3.0
```

In [29]:
```
df['敷金：賃料の倍数'] = df['敷金']/df['賃料料']
df['礼金：賃料の倍数'] = df['礼金']/df['賃料料']
```

In [ ]:
```

```

In [30]:
```
df['賃料+管理費'] = df['賃料料'] + df['管理費']
```

In [31]:
```
splitted6 = df['住所'].str.split('区', expand=True)
splitted6.columns = ['区', '市町村']
splitted6['区'] = splitted6['区'] + '区'
splitted6['区'] = splitted6['区'].str.replace('東京都','')
```

In [32]:
```
#立地を「路線」「駅」「徒歩〜分」に分割
splitted7 = df['立地11'].str.split('/', expand=True)
splitted7.columns = ['路線1', '駅1']
splitted7['徒歩1'] = df['立地12']
splitted8 = df['立地21'].str.split('/', expand=True)
splitted8.columns = ['路線2', '駅2']
splitted8['徒歩2'] = df['立地22']
splitted9 = df['立地31'].str.split('/', expand=True)
splitted9.columns = ['路線3', '駅3']
splitted9['徒歩3'] = df['立地32']
```

In [33]:
```
df = pd.concat([df, splitted6, splitted7, splitted8, splitted9], axis=1)
```

In [34]:
```
df=df.fillna(0)
df["徒歩2"]=df["徒歩2"].astype("float32").astype("int64")
df["徒歩3"]=df["徒歩3"].astype("float32").astype("int64")
df["徒歩1"]=df["徒歩1"].astype("float32").astype("int64")
```

In [ ]:
```
```

In [35]:
```
#不要なカラムを削除
df.drop(['立地11','立地12','立地21','立地22','立地31','立地32'], axis=1, inplace=True)
```

In [36]:
```
#set(df['階層'].str.split('-', expand=True)[1])
```

In [ ]:
```
```

In [37]:
```
#階を数値化、地下はマイナス
splitted = df['階層'].str.split('-', expand=True)
splitted.columns = ['階数', '階数X']
splitted['階数'] = splitted['階数'].str.replace(u'階', u'')
splitted['階数'] = splitted['階数'].str.replace(u'B', u'-')
splitted['階数'] = splitted['階数'].str.replace(u'M', u'-')
splitted['階数'] = pd.to_numeric(splitted['階数'].str.strip())
splitted= splitted.drop(['階数X'],axis=1)
df = pd.concat([df, splitted], axis=1)
```

In [38]:
```
#建物高さを数値化。地下は無視。
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下1地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下2地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下3地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下4地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下5地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下6地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下7地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下8地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'地下9地上', u'')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'平屋', u'1')
df['建物の高さ'] = df['建物の高さ'].str.replace(u'階建', u'')
df['建物の高さ'] = pd.to_numeric(df['建物の高さ'])
```

In [39]:
```
#indexを振り直す
df = df.reset_index(drop=True)
```

In [ ]:
```
```

In [40]:
```
df['間取りDK'] = 0
df['間取りK'] = 0
df['間取りL'] = 0
df['間取りS'] = 0
df['間取り'] = df['間取り'].str.replace(u'ワンルーム', u'1') #ワンルームを1に変換
```

In [41]:
```
for x in range(len(df)):
    if 'DK' in df['間取り'][x]:
        df.loc[x,'間取りDK'] = 1
df['間取り'] = df['間取り'].str.replace(u'DK',u'')

for x in range(len(df)):
    if 'K' in df['間取り'][x]:
        df.loc[x,'間取りK'] = 1
df['間取り'] = df['間取り'].str.replace(u'K',u'')

for x in range(len(df)):
    if 'L' in df['間取り'][x]:
        df.loc[x,'間取りL'] = 1
df['間取り'] = df['間取り'].str.replace(u'L',u'')

for x in range(len(df)):
    if 'S' in df['間取り'][x]:
        df.loc[x,'間取りS'] = 1
df['間取り'] = df['間取り'].str.replace(u'S',u'')

df['間取り'] = pd.to_numeric(df['間取り'])
```

In [ ]:
```
```

In [42]:
```
import xgboost as xgb
from sklearn.model_selection import train_test_split
```

In [43]:
```
df.drop(['マンション名'], axis=1, inplace=True)
df.drop(['路線1'], axis=1, inplace=True)
df.drop(['路線2'], axis=1, inplace=True)
df.drop(['路線3'], axis=1, inplace=True)
df.drop(['敷金'], axis=1, inplace=True)
df.drop(['礼金'], axis=1, inplace=True)
df.drop(['階層'], axis=1, inplace=True)
df.drop(['間取り'], axis=1, inplace=True)
df.drop(['市町村'], axis=1, inplace=True)
```

In [44]:
```
final_y = df['賃料+管理費']
```

In [45]:
```
#df[df["住所"]=="東京都港区芝５"]
```

In [46]:
```
df.columns
```

Out[46]:
```
Index(['住所', '築年数', '建物の高さ', '賃料料', '管理費', '専有面積', '風呂なし　トイレ共同', '敷金：賃料の倍数',
       '礼金：賃料の倍数', '賃料+管理費', '区', '駅1', '徒歩1', '駅2', '徒歩2', '駅3', '徒歩3', '階数',
       '間取りDK', '間取りK', '間取りL', '間取りS'],
      dtype='object')
```

In [47]:
```
#予測したいデータ入力
predict_data00 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,0.0, 0.0,0.0,"港区","三田
predict_data01= pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,0.0,  1.0,0.0,"港区","三田
predict_data02 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,0.0,  2.0,0.0,"港区","三田
predict_data03= pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,0.0,  3.0,0.0,"港区","三田
predict_data10 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,1.0,0.0 ,0.0,"港区","三田
predict_data11 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,1.0,1.0 ,0.0,"港区","三田
predict_data12 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,1.0,  2.0,0.0,"港区","三田
predict_data13 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,1.0,  3.0,0.0,"港区","三田
predict_data20 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,2.0,  0.0,0.0,"港区","三田
predict_data21 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,2.0, 1.0,0.0,"港区","三田
predict_data22 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,2.0,  2.0,0.0,"港区","三田
predict_data23 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,2.0,  3,0.0,"港区","三田駅
predict_data30 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,3.0,  0.0,0.0,"港区","三田
predict_data31 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,3.0,  1.0,0.0,"港区","三田
predict_data32 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,3.0,  2.0,0.0,"港区","三田
predict_data33 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,1,3.0,  3.0,0.0,"港区","三田



#風呂なし　トイレ共同のない物件のカラムを作る
data00 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,0.0, 0.0,0.0,"港区","三田駅",1.0,"
data01= pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,0.0,  1.0,0.0,"港区","三田駅",1.0,"
data02 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,0.0,  2.0,0.0,"港区","三田駅",1.0,
data03= pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,0.0,  3.0,0.0,"港区","三田駅",1.0,"
data10 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,1.0,0.0 ,0.0,"港区","三田駅",1.0,"
data11 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,1.0,1.0 ,0.0,"港区","三田駅",1.0,"
data12 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,1.0,  2.0,0.0,"港区","三田駅",1.0,
data13 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,1.0,  3.0,0.0,"港区","三田駅",1.0,
data20 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,2.0,  0.0,0.0,"港区","三田駅",1.0,
data21 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,2.0, 1.0,0.0,"港区","三田駅",1.0,"
data22 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,2.0,  2.0,0.0,"港区","三田駅",1.0,
data23 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,2.0,  3,0.0,"港区","三田駅",1.0,"
data30 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,3.0,  0.0,0.0,"港区","三田駅",1.0,
data31 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,3.0,  1.0,0.0,"港区","三田駅",1.0,
data32 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,3.0,  2.0,0.0,"港区","三田駅",1.0,
data33 = pd.DataFrame(np.array([["東京都港区芝５",19.0,8.0,0.0,0.0,15.0,0,3.0,  3.0,0.0,"港区","三田駅",1.0,
```

In [ ]:

```
```

In [48]:

```
df= pd.concat([df,predict_data00 ,predict_data01,predict_data02,predict_data03,predict_data10,predict_da
               predict_data20,predict_data21,predict_data22,predict_data23,predict_data30,predict_data31,
               data00 ,data01,data02,data03,data10,data11,data12,data13,\
               data20,data21,data22,data23,data30,data31,data32,data33])
```

In [49]:

```
df["徒歩2"]=df["徒歩2"].astype("float32").astype("int64")
df["徒歩3"]=df["徒歩3"].astype("float32").astype("int64")
df["徒歩1"]=df["徒歩1"].astype("float32").astype("int64")
```

In [50]:

```
df.columns
```

Out[50]:

```
Index(['住所', '築年数', '建物の高さ', '賃料料', '管理費', '専有面積', '風呂なし　トイレ共同', '敷金：賃料の倍数',
       '礼金：賃料の倍数', '賃料+管理費', '区', '駅1', '徒歩1', '駅2', '徒歩2', '駅3', '徒歩3', '階数',
       '間取りDK', '間取りK', '間取りL', '間取りS'],
      dtype='object')
```

In [51]:

```
df["築年数"]=df["築年数"].astype("float32").astype("int64")
df["建物の高さ"]=df["建物の高さ"].fillna(0).astype("float32").astype("int64")
df["賃料料"]=df["賃料料"].astype("float32")
df["管理費"]=df["管理費"].astype("float32")
df["専有面積"]=df["専有面積"].astype("float32")
df["敷金：賃料の倍数"]=df["敷金：賃料の倍数"].astype("float32")
df["礼金：賃料の倍数"]=df["礼金：賃料の倍数"].astype("float32")
df["賃料+管理費"]=df["賃料+管理費"].astype("float32")
df["階数"]=df["階数"].fillna(0).astype("float32").astype("int64")
df["風呂なし　トイレ共同"]=df["風呂なし　トイレ共同"].fillna(0).astype("float32").astype("int64")
df["間取りDK"]=df["間取りDK"].fillna(0).astype("float32").astype("int64")
df["間取りK"]=df["間取りK"].fillna(0).astype("float32").astype("int64")
df["間取りL"]=df["間取りL"].fillna(0).astype("float32").astype("int64")
df["間取りS"]=df["間取りS"].fillna(0).astype("float32").astype("int64")
```

In [52]:

```
df = df.rename(columns={'風呂なし\u3000トイレ共同': "風呂なしトイレ共同"})
```

In [ ]:

In [53]:
```
df = pd.get_dummies(df)
```

In [ ]:

In [54]:
```
df.drop(['賃料+管理費'], axis=1, inplace=True)
df.drop(['賃料料'], axis=1, inplace=True)
df.drop(['管理費'], axis=1, inplace=True)
```

In [55]:
```
df0= df[:-32]
```

In [56]:
```
final_x = df0
```

In [57]:
```

X_train, X_test, y_train, y_test = train_test_split(final_x, final_y, test_size=0.1, random_state=42,shu
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=42,shuff
```

In [211]:
```
xgdmat=xgb.DMatrix(X_train,y_train)
```

In [212]:
```
deval = xgb.DMatrix(X_val, y_val)
watchlist = [(deval, 'eval')]
our_params={'eta':0.1,'seed':0,'subsample':0.8,'colsample_bytree':0.8,'objective':'reg:linear','max_dept
final_gb=xgb.train(our_params,xgdmat,100,watchlist,obj=None,feval=None,maximize=False,early_stopping_rou
evals_result=None,verbose_eval=True,xgb_model=None)
tesdmat=xgb.DMatrix(X_test)
y_pred=final_gb.predict(tesdmat)
print(y_pred)
```

```
[22:01:13] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 926 extra nodes, 0 pruned nodes, ma
x_depth=15
[0]     eval-rmse:143626
[22:01:17] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 962 extra nodes, 0 pruned nodes, ma
x_depth=15
[1]     eval-rmse:130716
[22:01:20] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 980 extra nodes, 0 pruned nodes, ma
x_depth=15
[2]     eval-rmse:119058
[22:01:23] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1242 extra nodes, 0 pruned nodes, m
ax_depth=15
[3]     eval-rmse:108541
[22:01:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1404 extra nodes, 0 pruned nodes, m
ax_depth=15
[4]     eval-rmse:99278.4
[22:01:30] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1492 extra nodes, 0 pruned nodes, m
ax_depth=15
[5]     eval-rmse:90714.3
[22:01:34] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 1824 extra nodes, 0 pruned nodes, m
```

In [ ]:
```
```

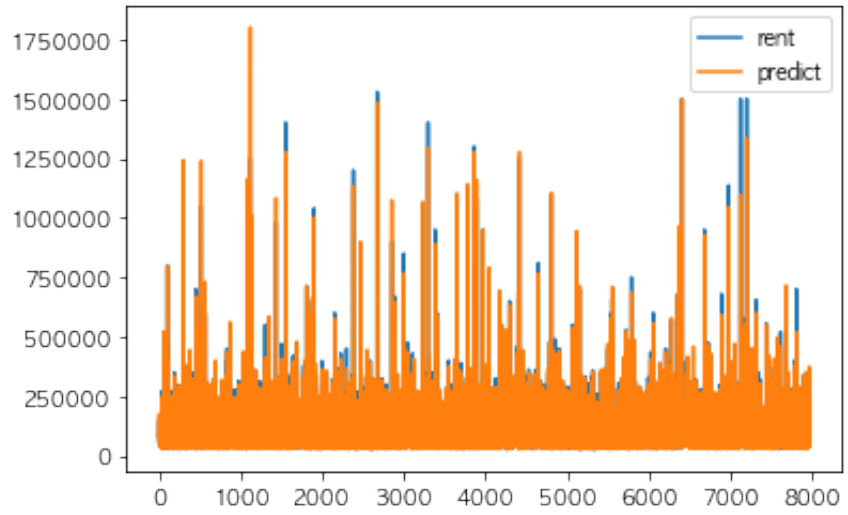
In [213]:
```
y_pred
```

Out[213]:
```
array([ 80093.03 , 173901.44 ,  52511.95 , ...,  95055.55 ,  72368.78 ,
        38958.254], dtype=float32)
```

In [214]:
```
y_test2 = np.array(y_test)
```

In [215]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [216]:
```
plt.plot(y_test2, label='rent')
plt.plot(y_pred, label='predict')
plt.legend()
plt.show()
```



In [217]:
```
y_pred[0]
```

Out[217]:
```
80093.03
```

In [218]:
```
from sklearn.metrics import mean_squared_error
import math
testScore=math.sqrt(mean_squared_error(y_test.values,y_pred))
print(testScore)
```

```
18654.729990107393
```

In [ ]:

In [63]:
```
import math

def rmsle(y, y_pred):
    assert len(y) == len(y_pred)
    terms_to_sum = [(math.log(y_pred[i] + 1) - math.log(y[i] + 1)) ** 2.0 for i,pred in enumerate(y_pred
    return (sum(terms_to_sum) * (1.0/len(y))) ** 0.5
```

In [220]:
```
rmsle(y_test.values,y_pred)
```

Out[220]:
```
0.08534621217742563
```

In [ ]:

In [236]:
```
graph1= xgb.to_graphviz(final_gb,  num_trees=1)
graph1.format = 'png'
graph1.render('tree')
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.833151 to fit
```

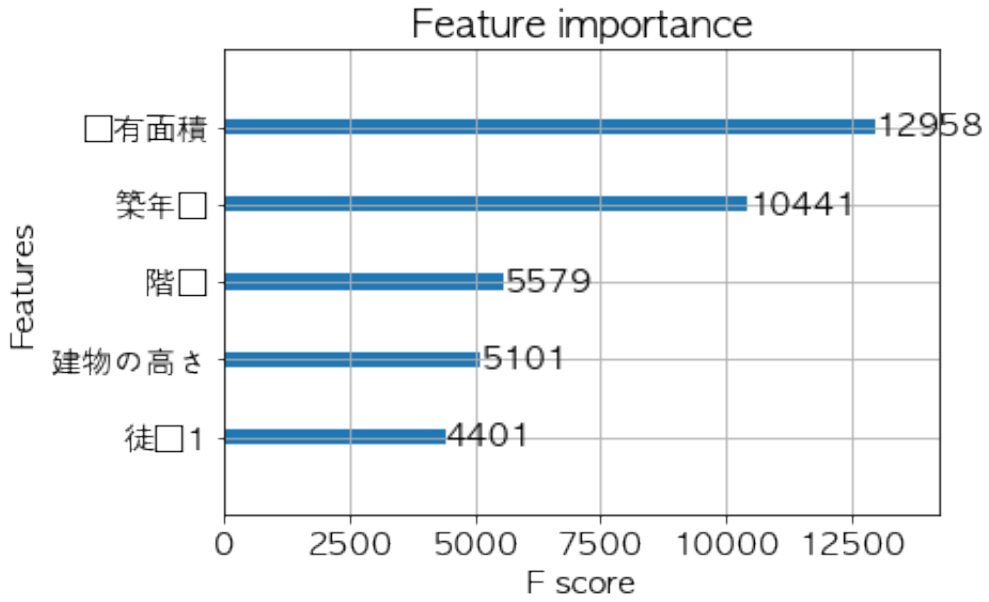
Out[236]:
```
'tree.png'
```

In [ ]:

In [ ]:

In [237]:
```
plt.rcParams.update({'font.family': 'AppleGothic'})
plt.rcParams.update({'font.size': '15'})
xgb.plot_importance(final_gb, importance_type='weight',
      max_num_features=5)
```

Out[237]: <matplotlib.axes._subplots.AxesSubplot at 0x11eeb0630>



In [ ]:

In [ ]:

In [ ]:

In [68]:
```
predict_data = df[-32:-16]
```

In [225]:
```
predict1=xgb.DMatrix(predict_data)
y_pred1=final_gb.predict(predict1)
print(y_pred1)
```

```
[82401.734 82770.21  90471.73  87936.04  76094.42  76569.26  85109.414
 82573.72  77051.28  77078.83  85771.01  83235.31  83947.17  83862.03
 88712.47  86176.78 ]
```

In [ ]:
```
```

In [226]:
```
#賃料予測
#0,1.0,2.0,3.0
columns = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="礼金")
index = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="敷金")
rent1 = pd.DataFrame(data=y_pred1.reshape(4,4), index=index, columns=columns)
```

In [227]:
```
rent1
```

Out[227]:

礼金 敷金	なし	1ヶ月	2ヶ月	3ヶ月
なし	82401.734375	82770.210938	90471.726562	87936.039062
1ヶ月	76094.421875	76569.257812	85109.414062	82573.718750
2ヶ月	77051.281250	77078.828125	85771.007812	83235.312500
3ヶ月	83947.171875	83862.031250	88712.468750	86176.781250

In [69]:
```
predict_data2 = df[-16:]
```

In [228]:
```
predict2=xgb.DMatrix(predict_data2)
y_pred2=final_gb.predict(predict2)
print(y_pred2)
```

```
[86798.67  87167.15  95335.04  92799.35  80491.38  80966.22  89972.73
 87437.03  79050.4   79077.945 88236.49  85700.8   86110.984 86025.84
 91342.64  88806.95 ]
```

In [229]:
```
columns = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="礼金")
index = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="敷金")
rent = pd.DataFrame(data=y_pred2.reshape(4,4), index=index, columns=columns)
```

In [230]:
```
rent
```

Out[230]:

礼金 敷金	なし	1ヶ月	2ヶ月	3ヶ月
なし	86798.671875	87167.148438	95335.039062	92799.351562
1ヶ月	80491.382812	80966.218750	89972.726562	87437.031250
2ヶ月	79050.398438	79077.945312	88236.492188	85700.796875
3ヶ月	86110.984375	86025.843750	91342.640625	88806.953125

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [58]:
```
import lightgbm as lgb
```

```
/Users/kai/anaconda3/lib/python3.6/site-packages/lightgbm/__init__.py:46: UserWarning: Starting from ve
rsion 2.2.1, the library file in distribution wheels for macOS is built by the Apple Clang (Xcode_8.3.1
) compiler.
This means that in case of installing LightGBM from PyPI via the ``pip install lightgbm`` command, you
don't need to install the gcc compiler anymore.
Instead of that, you need to install the OpenMP library, which is required for running LightGBM on the
system with the Apple Clang compiler.
You can install the OpenMP library by the following command: ``brew install libomp``.
  "You can install the OpenMP library by the following command: ``brew install libomp``.", UserWarning)
```

In [59]:
```
lgb_train = lgb.Dataset(X_train,y_train)
lgb_test = lgb.Dataset(X_test)
```

In [60]:
```
params = {
        'task' : 'train',
        'boosting_type' : 'gbdt',
        'objective' : 'regression',
        'num_leaves' : 33,
        'learning_rate' : 0.01,
        'feature_fraction' : 0.8,
        'bagging_fraction' : 0.8,
        'bagging_freq': 5,
        'verbose' : 0
}

lgb_eval = lgb.Dataset(X_val, y_val, reference=lgb_train)

gbm = lgb.train(params,lgb_train ,valid_sets=lgb_eval,num_boost_round=100000)
```

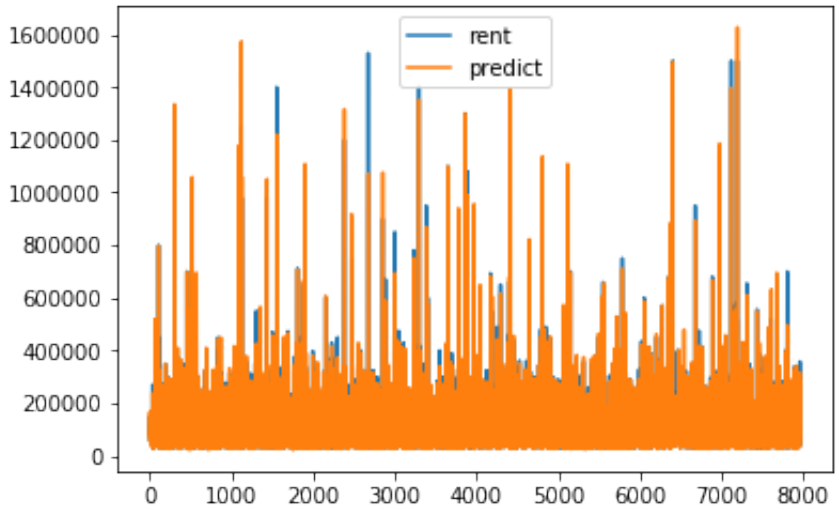
```
[1]     valid_0's l2: 1.10428e+10
[2]     valid_0's l2: 1.08781e+10
[3]     valid_0's l2: 1.07065e+10
[4]     valid_0's l2: 1.05524e+10
[5]     valid_0's l2: 1.0386e+10
[6]     valid_0's l2: 1.02291e+10
[7]     valid_0's l2: 1.00732e+10
[8]     valid_0's l2: 9.93685e+09
[9]     valid_0's l2: 9.79186e+09
[10]    valid_0's l2: 9.64352e+09
[11]    valid_0's l2: 9.49115e+09
[12]    valid_0's l2: 9.34231e+09
[13]    valid_0's l2: 9.19618e+09
[14]    valid_0's l2: 9.07525e+09
[15]    valid_0's l2: 8.94692e+09
[16]    valid_0's l2: 8.81356e+09
[17]    valid_0's l2: 8.68588e+09
[18]    valid_0's l2: 8.57617e+09
[19]    valid_0's l2: 8.45638e+09
```

In [61]:
```
y_pred3 = gbm.predict(X_test)
```

In [64]:
```
rmsle(y_test.values, y_pred3)
```

Out[64]:
```
0.09060221904220085
```

In [66]:
```
import matplotlib.pyplot as plt
plt.plot(y_test.values, label='rent')
plt.plot(y_pred3, label='predict')
plt.legend()
plt.show()
```



In [ ]:

In [70]:
```
predict4=gbm.predict(predict_data)
print(predict4)
```

```
[ 82405.29508861  89135.45203164  90455.48260819  95636.4323161
  76902.97292647  75855.58208727  76647.18443939  81864.09152602
  74481.61660317  74257.09445083  74643.60535326  79855.12076026
 123808.16690203 100369.43892599  96800.58791372 101216.56566293]
```

In [71]:
```
columns = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="礼金")
index = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="敷金")
rent4= pd.DataFrame(predict4.reshape(4,4),index=index, columns=columns)
```

In [72]:
```
rent4
```

Out[72]:

礼金	なし	1ヶ月	2ヶ月	3ヶ月
敷金				
なし	82405.295089	89135.452032	90455.482608	95636.432316
1ヶ月	76902.972926	75855.582087	76647.184439	81864.091526
2ヶ月	74481.616603	74257.094451	74643.605353	79855.120760
3ヶ月	123808.166902	100369.438926	96800.587914	101216.565663

In [73]:
```
predict5=gbm.predict(predict_data2)
print(predict5)
```

```
[ 86525.24835417  93255.40529721  94575.43587375  99756.38558166
  81022.92619204  79975.53535283  80767.13770496  85984.04479159
  78669.47578246  78444.95363012  78831.46453255  84042.97993955
 127676.13078224 104237.4028062  100668.55179393 105084.52954314]
```

In [74]:
```
columns = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="礼金")
index = pd.Index(["なし","1ヶ月","2ヶ月","3ヶ月"],name="敷金")
rent5= pd.DataFrame(predict5.reshape(4,4),index=index, columns=columns)
```

In [75]:
```
rent5
```

Out[75]:

礼金 敷金	なし	1ヶ月	2ヶ月	3ヶ月
なし	86525.248354	93255.405297	94575.435874	99756.385582
1ヶ月	81022.926192	79975.535353	80767.137705	85984.044792
2ヶ月	78669.475782	78444.953630	78831.464533	84042.979940
3ヶ月	127676.130782	104237.402806	100668.551794	105084.529543

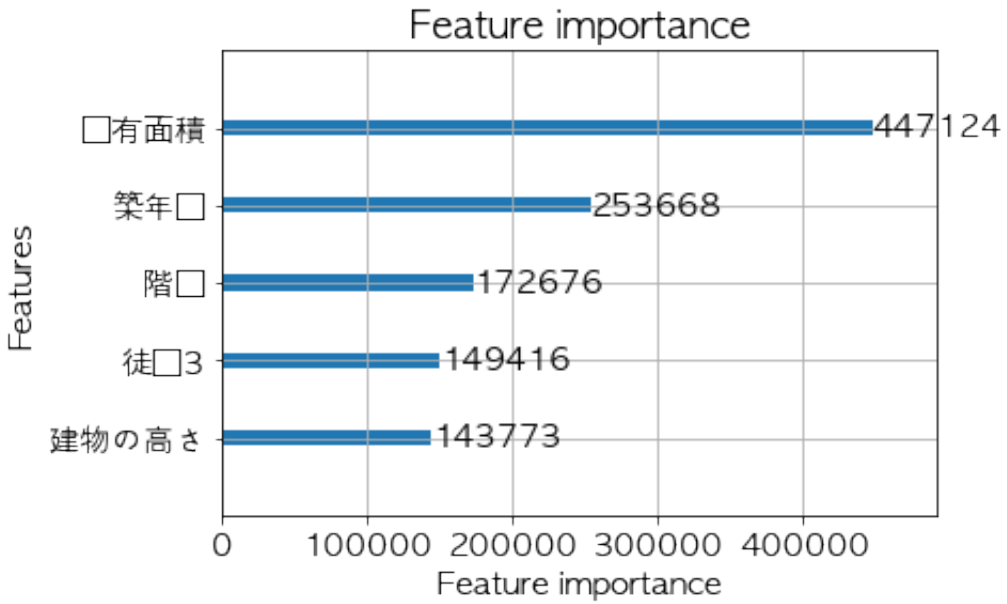
In [ ]:

In [ ]:

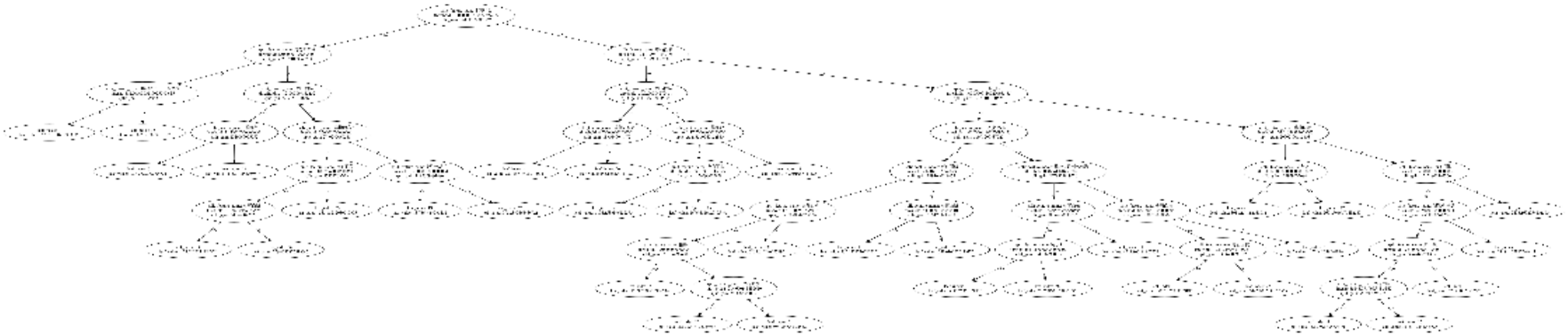
In [ ]:

In [ ]:

In [78]:
```
plt.rcParams.update({'font.family': 'AppleGothic'})
plt.rcParams.update({'font.size': '15'})
ax = lgb.plot_importance(gbm, max_num_features=5)
plt.show()
```



In [82]:
```
ax = lgb.plot_tree(gbm, tree_index=1, figsize=(20, 8), show_info=['split_gain'])
plt.show()
```



In [83]:
```
graph = lgb.create_tree_digraph(gbm, tree_index=3, name='Tree3')
graph.render(view=True)
```

Out[83]: 'Tree3.gv.pdf'

In [ ]:
```
```