In [7]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from collections import Counter
import lightgbm as lgb
import seaborn as sns
```

In [8]:

```
train = pd.read_csv("/Users/BarryFitzpatrick/Machine Learning/Kaggle Group/tcd-ml-comp-201920-inco
me-pred-group/tcd-ml-1920-group-income-train.csv")
test = pd.read_csv("/Users/BarryFitzpatrick/Machine Learning/Kaggle Group/tcd-ml-comp-201920-incom
e-pred-group/tcd-ml-1920-group-income-test.csv")

train = train.sample(frac = 1)
train.shape
```

```
/Users/BarryFitzpatrick/anaconda3/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:2785: DtypeWarning: Columns (2,4) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
/Users/BarryFitzpatrick/anaconda3/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:2785: DtypeWarning: Columns (4) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[8]:

```
(1048574, 17)
```

In [9]:

```
train_missing = (train.isnull().sum()/len(train))*100
train_missing = train_missing.drop(train_missing[train_missing==0].index).sort_values(ascending=Fal
se)
miss_data = pd.DataFrame({'缺失百分比':train_missing})
miss_data
```

Out[9]:

	缺失百分比
University Degree	7.686630
Gender	7.069315
Hair Color	6.695856
Satisfation with employer	3.632266
Year of Record	0.382710
Profession	0.272084

In [5]:

```
train.head()
```

Out[5]:

	Instance	Year of Record	Housing Situation	Crime Level in the City of Employement	Work Experience in Current Job [years]	Satisfation with employer	Gender	Age	Country	Size of City	Profession	Unive De

	Instance	Year of Record	Housing Situation	Crime Level in the City of Employement	Work Experience in Current Job [years]	Satisfation with employer	Gender	Age	Country	Size of City	Profession	Unive De
701573	644709	1992.0	Small House	90	10	Average	unknown	17	Dominican Republic	1627279	project manager	
167938	167939	1953.0	nA	23	21	Average	male	53	Papua New Guinea	1316693	probation officer	Ma
795959	739095	1999.0	Large Apartment	33	8	Happy	unknown	17	Togo	92767	industrial program compliance analyst	
180758	180759	1954.0	nA	110	10	Somewhat Happy	female	21	Portugal	1296332	office administrator	Ma
175084	175085	1954.0	nA	135	24	Average	male	58	Czechia	1426606	scientistscientist	Bac

In [ ]:

In [10]:

```
data = pd.concat([train,test],ignore_index=True)

data['University Degree']=data['University Degree'].fillna('Bachelor')

data['Gender']=data['Gender'].replace('m','male')
data['Gender']=data['Gender'].replace('f','female')
data['Gender']=data['Gender'].replace('unknown','other')
data['Gender']=data['Gender'].fillna('female')

#data['Housing Situation']=data['Housing Situation'].replace('nA','0')
data['Housing Situation']=np.where(data['Housing Situation']=='0', 'nA', data['Housing 
Situation'])
data['Housing Situation']=np.where(data['Housing Situation']==0, 'nA', data['Housing Situation'])
data['Hair Color']=data['Hair Color'].fillna(method='bfill')

data['Satisfation with employer']=data['Satisfation with employer'].fillna('Average')

data.fillna(value={'Year of Record':data['Year of Record'].mean()}, inplace=True)

data['Profession']=data['Profession'].fillna(method='bfill')

data['Country']=data['Country'].fillna(method='bfill')

data.shape
```

Out[10]:

```
(1418012, 17)
```

In [11]:

```
#构造等级特征
data['Satisfation with employer'] = data['Satisfation with employer'].map \
     ({'Average':2, 'Happy':4, 'Somewhat Happy':3, 'Unhappy':1})
```

In [12]:

```
data.isnull().any()
```

Out[12]:

```
Instance                                    False
Year of Record                              False
Housing Situation                           False
Crime Level in the City of Employement      False
Work Experience in Current Job [years]      False
Satisfation with employer                   False
Gender                                      False
Age                                         False
Country                                     False
Size of City                                False
```

```
Profession                                                  False
University Degree                                           False
Wears Glasses                                               False
Hair Color                                                  False
Body Height [cm]                                            False
Yearly Income in addition to Salary (e.g. Rental Income)    False
Total Yearly Income [EUR]                                    True
dtype: bool
```

In [13]:

```
#对于每个country和profession特征，用其特征值下收入均值来替换
country_income = dict(train.groupby('Country').mean()['Total Yearly Income [EUR]']/10000)
data.Country = data.Country.map(country_income)
data.Country = data.Country.fillna(data.Country.mean())
country_income = dict(train.groupby('Profession').mean()['Total Yearly Income [EUR]']/10000)
data.Profession = data.Profession.map(country_income)
country_income = dict(train.groupby('Profession').mean()['Total Yearly Income [EUR]']/10000)
data.Profession = data.Profession.map(country_income)
#前面的254287数据用来构造均值特征
sp = 254287
```

In [14]:

```
#转换成数值
data.iloc[:,-2] = data.iloc[:,-2].map(lambda x: float(x[:-3]))
```

In [15]:

```
data['BigCity'] = np.where(data['Size of City']>7335190, 1, 0)
data['SmallCity'] = np.where(data['Size of City']<7335190, 1, 0)
#data = data.drop(columns=["Size of City"])
```

In [16]:

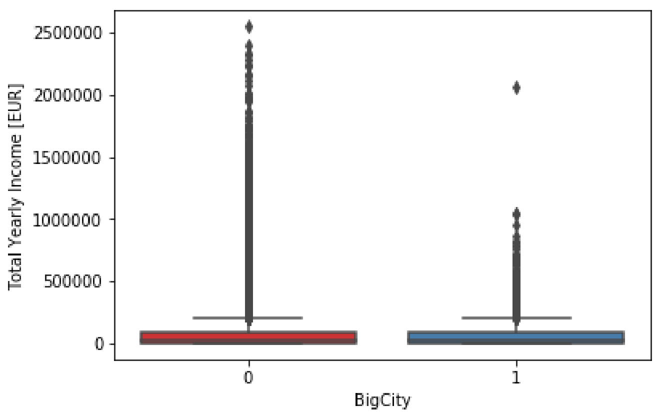
```
data['Crime Level in the City of Employement']=data['Crime Level in the City of
Employement'].replace(0,data['Crime Level in the City of Employement'].mean())
```

In [17]:

```
sns.boxplot(x=data['BigCity'], y=data["Total Yearly Income [EUR]"], data=data, palette="Set1")
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e501c50>
```



In [ ]:

```
# Remove outliers in Size of City
#indexBigCityOutliers = data[ (data["BigCity"] == 1)  & (data["Total Yearly Income [EUR]"] > 15000
00) ].index
#indexBigCityOutliers
```

In [ ]:

```
#data = data.drop(indexBigCityOutliers)
```

In [18]:

```
data['Housing Situation'].value_counts()
```

Out[18]:

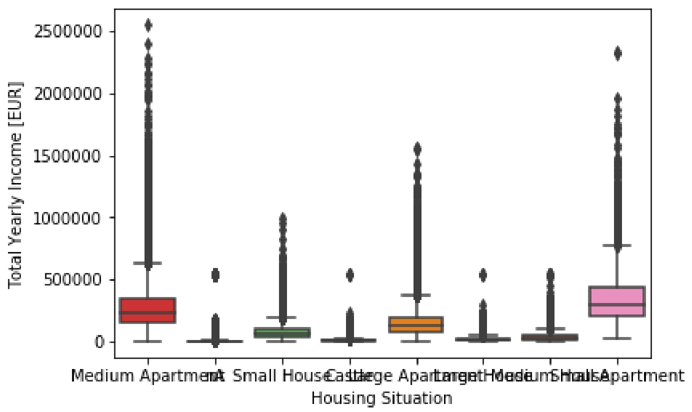
```
nA                   365345
Large House          212200
Medium House         184710
Castle               170926
Large Apartment      170623
Small House          169785
Medium Apartment     134157
Small Apartment       10266
Name: Housing Situation, dtype: int64
```

In [19]:

```
# Have changed 0's and '0' to nA
sns.boxplot(x=data['Housing Situation'], y=data["Total Yearly Income [EUR]"], data=data, palette="
Set1")
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11d01d4a8>
```

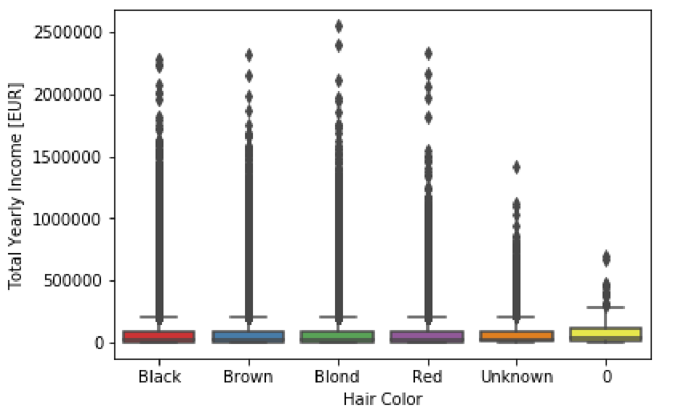


In [20]:

```
sns.boxplot(x=data['Hair Color'], y=data["Total Yearly Income [EUR]"], data=data, palette="Set1")
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11cc5cda0>
```

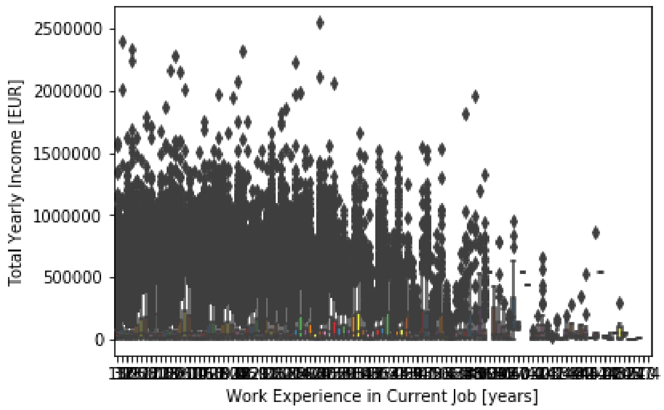


In [30]:

```
sns.boxplot(x=data['Work Experience in Current Job [years]'], y=data["Total Yearly Income [EUR]"], data=data, palette="Set1")
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x123cc2828>
```

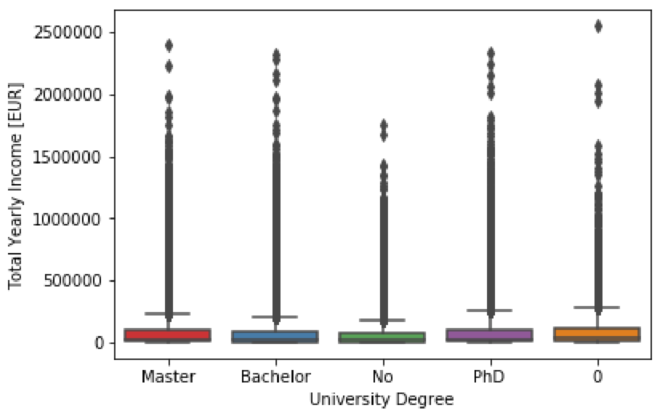


In [31]:

```
sns.boxplot(x=data['University Degree'], y=data["Total Yearly Income [EUR]"], data=data, palette="Set1")
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x160e26b70>
```

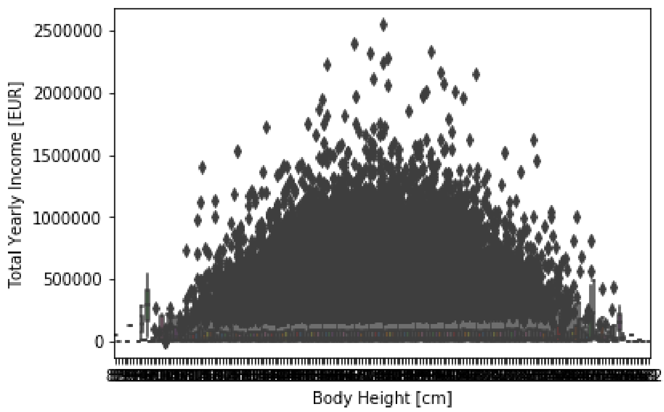


In [32]:

```
sns.boxplot(x=data['Body Height [cm]'], y=data["Total Yearly Income [EUR]"], data=data, palette="Set1")
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x150c91860>
```



In [33]:

```
sns.boxplot(x=data['Crime Level in the City of Employement'], y=data["Total Yearly Income [EUR]"],
data=data, palette="Set1")
```

Out[33]:

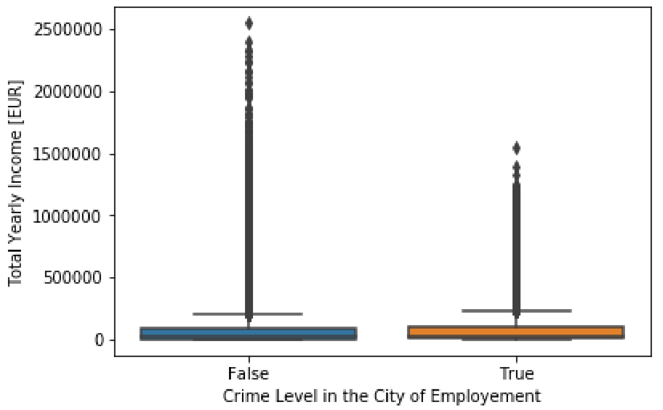
```
<matplotlib.axes._subplots.AxesSubplot at 0x137e39c88>
```

In [45]:

```
sns.boxplot(x = data['Crime Level in the City of Employement']==0,y=data["Total Yearly Income [EUR]
"])
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14da809b0>
```

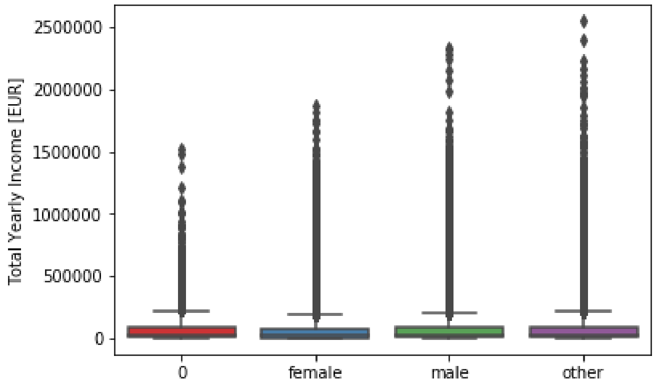


In [46]:

```
sns.boxplot(x=data['Gender'], y=data["Total Yearly Income [EUR]"], data=data, palette="Set1")
```

Out[46]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14d5a1f60>
```



Gender

In [ ]:

```

```

In [ ]:

```
data.head()
```

In [21]:

```
cats = ['Year of Record', 'Housing Situation','Country', 'Size of City',
        'Crime Level in the City of Employement','Work Experience in Current Job [years]']
cons = ['Satisfation with employer','Gender','Age',
       'University Degree','Body Height [cm]','Profession']
data['Work Experience in Current Job [years]'] = data['Work Experience in Current Job [years]'].as
type(str)
```

In [22]:

```
#This is the inspiration I got from the best code.
#I added and constructed mean features, cross mean features and Category Characteristics

def create_feature(df,cats,cons,normalize=True):
    for cat in cats:
        value = df[cat].value_counts(dropna=False, normalize=normalize).to_dict()
        num = cat + '_FE_FULL'
        df[num + num] = df[cat].map(value)
        #构造均值特征mean features
        df[num] = df[cat].map( dict(df.iloc[:sp].groupby(cat).mean()['Total Yearly Income [EUR]']/1
0000))
        df[num] = df[num].fillna(df[num].mean())
        df[num] = df[num].astype('float32')
        for con in cons:
            new_col = cat +'_'+ con
            df[new_col] = df[cat].astype(str)+'_'+df[con].astype(str)
            temp_df = df[new_col]
            fq_encode = temp_df.value_counts(normalize=True).to_dict()
            #构造交叉均值特征cross mean features
            df[new_col] = df[new_col].map( dict(df.iloc[:sp+1].groupby( \
                                 new_col).mean()['Total Yearly Income [EUR]']/10000))
            df[new_col] = df[new_col].fillna(df[new_col].mean())
    return df

data = create_feature(data,cats,cons)
data['Work Experience in Current Job [years]'] = data['Work Experience in Current Job [years]' \
                                                      ].replace('#NUM!', data.iloc[:, -1].mean()).as
pe(float)

#构造类别特征 Category Characteristics
for col in data.dtypes[data.dtypes == 'object'].index.tolist():
    feat_le = LabelEncoder()
    feat_le.fit(data[col].unique().astype(str))
    data[col] = feat_le.transform(data[col].astype(str))

del_col = set(['Total Yearly Income [EUR]','Instance'])
features_col =  list(set(data) - del_col)
features_col
```

Out[22]:

```
['Satisfation with employer',
 'Profession',
 'Crime Level in the City of Employement_Age',
 'Year of Record_FE_FULL',
 'Year of Record_Profession',
 'Size of City_Gender',
 'University Degree',
 'Year of Record_Satisfation with employer',
 'Year of Record_Age',
 'Work Experience in Current Job [years]_Profession',
 'Housing Situation_Profession',
 'Housing Situation',
```

```
 'Age',
 'Yearly Income in addition to Salary (e.g. Rental Income)',
 'Size of City_FE_FULLSize of City_FE_FULL',
 'Year of Record_Gender',
 'Year of Record_Body Height [cm]',
 'Housing Situation_FE_FULLHousing Situation_FE_FULL',
 'Country_FE_FULLCountry_FE_FULL',
 'Size of City_Profession',
 'Work Experience in Current Job [years]_Gender',
 'Crime Level in the City of Employement_FE_FULL',
 'Year of Record',
 'Housing Situation_Body Height [cm]',
 'Work Experience in Current Job [years]_Satisfation with employer',
 'Body Height [cm]',
 'Work Experience in Current Job [years]_University Degree',
 'Crime Level in the City of Employement',
 'Housing Situation_FE_FULL',
 'Size of City_Age',
 'Gender',
 'Size of City',
 'Crime Level in the City of Employement_Gender',
 'Country_FE_FULL',
 'Size of City_Satisfation with employer',
 'Country',
 'Housing Situation_Gender',
 'Size of City_FE_FULL',
 'SmallCity',
 'Crime Level in the City of Employement_Profession',
 'BigCity',
 'Crime Level in the City of Employement_FE_FULLCrime Level in the City of Employement_FE_FULL',
 'Work Experience in Current Job [years]_FE_FULL',
 'Work Experience in Current Job [years]',
 'Country_University Degree',
 'Housing Situation_Age',
 'Country_Satisfation with employer',
 'Crime Level in the City of Employement_Satisfation with employer',
 'Year of Record_University Degree',
 'Wears Glasses',
 'Country_Age',
 'Size of City_Body Height [cm]',
 'Work Experience in Current Job [years]_Age',
 'Crime Level in the City of Employement_Body Height [cm]',
 'Country_Body Height [cm]',
 'Year of Record_FE_FULLYear of Record_FE_FULL',
 'Country_Gender',
 'Hair Color',
 'Work Experience in Current Job [years]_FE_FULLWork Experience in Current Job [years]_FE_FULL',
 'Housing Situation_University Degree',
 'Country_Profession',
 'Size of City_University Degree',
 'Crime Level in the City of Employement_University Degree',
 'Housing Situation_Satisfation with employer',
 'Work Experience in Current Job [years]_Body Height [cm]']
```

In [24]:

```
data.shape
```

Out[24]:

```
(1418012, 67)
```

In [25]:

```
from sklearn.ensemble import RandomForestRegressor

param = {'num_iterations':20000,
         'max_depth': 21,
         'objective':'regression',
         "verbosity": -1,
         'metric': 'mae',
         'bagging_fraction': 0.8,
         'learning_rate': 0.01,}
X_train,X_test  = data[features_col].iloc[:1048573],data[features_col].iloc[1048574:]
Y_train = data['Total Yearly Income [EUR]'].iloc[:1048573]
```

```
Y_train = data['Total Yearly Income [EUR]'].iloc[:1048575]
x_train,x_val,y_train,y_val = X_train.iloc[sp+1: ,:],  X_train.iloc[:sp,:],  \
                    Y_train.iloc[sp+1: ],  Y_train.iloc[:sp ]
train_data = lgb.Dataset(x_train, label=y_train, feature_name='auto')#categorical_feature=cat
val_data = lgb.Dataset(x_val, label=y_val, feature_name='auto')

bst = lgb.train(param, train_data, 20000, verbose_eval = 100, valid_sets=[val_data])
```

```
/Users/BarryFitzpatrick/anaconda3/lib/python3.7/site-packages/lightgbm/engine.py:148: UserWarning:
Found `num_trees` in params. Will use it instead of argument
  warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))
```

```
[100] valid_0's l1: 31607.9
[200] valid_0's l1: 18388.6
[300] valid_0's l1: 13825.5
[400] valid_0's l1: 12009.1
[500] valid_0's l1: 11184.9
[600] valid_0's l1: 10777.6
[700] valid_0's l1: 10508.3
[800] valid_0's l1: 10305.7
[900] valid_0's l1: 10144.1
[1000] valid_0's l1: 9990.41
[1100] valid_0's l1: 9876.47
[1200] valid_0's l1: 9786.57
[1300] valid_0's l1: 9684.97
[1400] valid_0's l1: 9606.25
[1500] valid_0's l1: 9539.65
[1600] valid_0's l1: 9479.77
[1700] valid_0's l1: 9431.6
[1800] valid_0's l1: 9380.01
[1900] valid_0's l1: 9331.96
[2000] valid_0's l1: 9294.14
[2100] valid_0's l1: 9257.87
[2200] valid_0's l1: 9222.84
[2300] valid_0's l1: 9186.14
[2400] valid_0's l1: 9158.62
[2500] valid_0's l1: 9131.94
[2600] valid_0's l1: 9108.56
[2700] valid_0's l1: 9086.61
[2800] valid_0's l1: 9068
[2900] valid_0's l1: 9049.9
[3000] valid_0's l1: 9033.42
[3100] valid_0's l1: 9017.55
[3200] valid_0's l1: 9002.63
[3300] valid_0's l1: 8984.13
[3400] valid_0's l1: 8966.84
[3500] valid_0's l1: 8948.6
[3600] valid_0's l1: 8932.73
[3700] valid_0's l1: 8914.92
[3800] valid_0's l1: 8898.85
[3900] valid_0's l1: 8882.64
[4000] valid_0's l1: 8866.11
[4100] valid_0's l1: 8852.25
[4200] valid_0's l1: 8838.08
[4300] valid_0's l1: 8823.32
[4400] valid_0's l1: 8812.01
[4500] valid_0's l1: 8798.16
[4600] valid_0's l1: 8785.47
[4700] valid_0's l1: 8772.14
[4800] valid_0's l1: 8761.49
[4900] valid_0's l1: 8754.27
[5000] valid_0's l1: 8744.15
[5100] valid_0's l1: 8734.49
[5200] valid_0's l1: 8724.68
[5300] valid_0's l1: 8713.63
[5400] valid_0's l1: 8706.56
[5500] valid_0's l1: 8697.85
[5600] valid_0's l1: 8688.99
[5700] valid_0's l1: 8682.59
[5800] valid_0's l1: 8675.27
[5900] valid_0's l1: 8670.71
[6000] valid_0's l1: 8665.22
[6100] valid_0's l1: 8656.41
[6200] valid_0's l1: 8649.41
[6300] valid_0's l1: 8638.32
[6400] valid_0's l1: 8626.14
```

```
[6500] valid_0's l1: 8614.37
[6600] valid_0's l1: 8605.72
[6700] valid_0's l1: 8593.99
[6800] valid_0's l1: 8585.28
[6900] valid_0's l1: 8578.78
[7000] valid_0's l1: 8570.31
[7100] valid_0's l1: 8562.18
[7200] valid_0's l1: 8555.06
[7300] valid_0's l1: 8548.43
[7400] valid_0's l1: 8539.6
[7500] valid_0's l1: 8537.19
[7600] valid_0's l1: 8534.81
[7700] valid_0's l1: 8531.22
[7800] valid_0's l1: 8526.3
[7900] valid_0's l1: 8521.84
[8000] valid_0's l1: 8516.86
[8100] valid_0's l1: 8512.01
[8200] valid_0's l1: 8507.82
[8300] valid_0's l1: 8503.25
[8400] valid_0's l1: 8497.41
[8500] valid_0's l1: 8492.74
[8600] valid_0's l1: 8488.96
[8700] valid_0's l1: 8485.59
[8800] valid_0's l1: 8482.55
[8900] valid_0's l1: 8479.56
[9000] valid_0's l1: 8476.23
[9100] valid_0's l1: 8473.74
[9200] valid_0's l1: 8469.83
[9300] valid_0's l1: 8466.43
[9400] valid_0's l1: 8464.56
[9500] valid_0's l1: 8460.63
[9600] valid_0's l1: 8456.23
[9700] valid_0's l1: 8450.7
[9800] valid_0's l1: 8445.39
[9900] valid_0's l1: 8440.4
[10000] valid_0's l1: 8436.14
[10100] valid_0's l1: 8432.02
[10200] valid_0's l1: 8429.33
[10300] valid_0's l1: 8425.67
[10400] valid_0's l1: 8423.09
[10500] valid_0's l1: 8420.76
[10600] valid_0's l1: 8418.82
[10700] valid_0's l1: 8413.58
[10800] valid_0's l1: 8410.69
[10900] valid_0's l1: 8406.74
[11000] valid_0's l1: 8406
[11100] valid_0's l1: 8402.51
[11200] valid_0's l1: 8399.41
[11300] valid_0's l1: 8397.48
[11400] valid_0's l1: 8396
[11500] valid_0's l1: 8393.06
[11600] valid_0's l1: 8388.99
[11700] valid_0's l1: 8385
[11800] valid_0's l1: 8381.95
[11900] valid_0's l1: 8380.93
[12000] valid_0's l1: 8377.49
[12100] valid_0's l1: 8375.03
[12200] valid_0's l1: 8373.55
[12300] valid_0's l1: 8370.76
[12400] valid_0's l1: 8369.19
[12500] valid_0's l1: 8365.68
[12600] valid_0's l1: 8363.84
[12700] valid_0's l1: 8361.54
[12800] valid_0's l1: 8359.56
[12900] valid_0's l1: 8358.96
[13000] valid_0's l1: 8357.79
[13100] valid_0's l1: 8355.4
[13200] valid_0's l1: 8355.11
[13300] valid_0's l1: 8352.24
[13400] valid_0's l1: 8351.7
[13500] valid_0's l1: 8349.29
[13600] valid_0's l1: 8347.28
[13700] valid_0's l1: 8344.46
[13800] valid_0's l1: 8342.26
[13900] valid_0's l1: 8340.55
[14000] valid_0's l1: 8337.87
[14100] valid_0's l1: 8336.65
```

```
[14200]	valid_0's l1: 8334.61
[14300]	valid_0's l1: 8332.52
[14400]	valid_0's l1: 8331.24
[14500]	valid_0's l1: 8326.27
[14600]	valid_0's l1: 8320
[14700]	valid_0's l1: 8319.8
[14800]	valid_0's l1: 8318.77
[14900]	valid_0's l1: 8318.31
[15000]	valid_0's l1: 8317.23
[15100]	valid_0's l1: 8315.83
[15200]	valid_0's l1: 8314.7
[15300]	valid_0's l1: 8310.9
[15400]	valid_0's l1: 8310.24
[15500]	valid_0's l1: 8305.88
[15600]	valid_0's l1: 8302.13
[15700]	valid_0's l1: 8299.68
[15800]	valid_0's l1: 8299.91
[15900]	valid_0's l1: 8298.55
[16000]	valid_0's l1: 8296.96
[16100]	valid_0's l1: 8292.89
[16200]	valid_0's l1: 8290.75
[16300]	valid_0's l1: 8290.33
[16400]	valid_0's l1: 8289.35
[16500]	valid_0's l1: 8289.21
[16600]	valid_0's l1: 8285.32
[16700]	valid_0's l1: 8282.37
[16800]	valid_0's l1: 8280.04
[16900]	valid_0's l1: 8276.58
[17000]	valid_0's l1: 8275.6
[17100]	valid_0's l1: 8273.91
[17200]	valid_0's l1: 8271.35
[17300]	valid_0's l1: 8267.61
[17400]	valid_0's l1: 8265.11
[17500]	valid_0's l1: 8263.88
[17600]	valid_0's l1: 8262.39
[17700]	valid_0's l1: 8259.87
[17800]	valid_0's l1: 8258.94
[17900]	valid_0's l1: 8255.94
[18000]	valid_0's l1: 8254.23
[18100]	valid_0's l1: 8251.35
[18200]	valid_0's l1: 8249.55
[18300]	valid_0's l1: 8248.63
[18400]	valid_0's l1: 8246.3
[18500]	valid_0's l1: 8244.57
[18600]	valid_0's l1: 8243.39
[18700]	valid_0's l1: 8241.38
[18800]	valid_0's l1: 8239
[18900]	valid_0's l1: 8238.07
[19000]	valid_0's l1: 8236.23
[19100]	valid_0's l1: 8234.59
[19200]	valid_0's l1: 8232.12
[19300]	valid_0's l1: 8230.02
[19400]	valid_0's l1: 8228.55
[19500]	valid_0's l1: 8227.63
[19600]	valid_0's l1: 8226.7
[19700]	valid_0's l1: 8225.29
[19800]	valid_0's l1: 8224.55
[19900]	valid_0's l1: 8223.83
[20000]	valid_0's l1: 8222.01
```

In [ ]:

```
from sklearn.metrics import mean_absolute_error
predict = bst.predict(x_val)
val_mae = mean_absolute_error(y_val,predict)
val_mae
```

In [ ]:

```
#生成结果
#rfr.fit(X_train, Y_train)
predict = bst.predict(X_test)
result=pd.DataFrame([range(1,1+len(predict)), predict]).T
result.columns = ['Instance', 'Total Yearly Income [EUR]']
result.to_csv("sub191125_10.csv",index=False)
```

```
result.head()
```

In [ ]:

```

```