In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df=pd.read_csv('train_dataset.csv')
df.head()
```

Out[2]:

	Year_Factor	State_Factor	building_class	facility_type	floor_area	year_built	energy_star_rating	ELEVATION	janua
0	1	State_1	Commercial	Grocery_store_or_food_market	61242.0	1942.0	11.0	2.4	
1	1	State_1	Commercial	Warehouse_Distribution_or_Shipping_center	274000.0	1955.0	45.0	1.8	
2	1	State_1	Commercial	Retail_Enclosed_mall	280025.0	1951.0	97.0	1.8	
3	1	State_1	Commercial	Education_Other_classroom	55325.0	1980.0	46.0	1.8	
4	1	State_1	Commercial	Warehouse_Nonrefrigerated	66000.0	1985.0	100.0	2.4	

5 rows × 64 columns

In [3]:
```
df_null=df.isna().sum().values
df_null
```

Out[3]:
```
array([    0,     0,     0,     0,     0,  1837, 26709,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0, 41082, 41811, 41082, 45796,     0,
           0], dtype=int64)
```

In [4]:
```
len(df)
```

Out[4]:
```
75757
```

In [5]:
```
df=df.dropna()
df.head()
```

Out[5]:

	Year_Factor	State_Factor	building_class	facility_type	floor_area	year_built	energy_star_rating	ELEVATION	january
404	2	State_1	Residential	Mixed_Use_Predominantly_Commercial	34173.0	1913.0	100.0	2.4	
405	2	State_1	Commercial	Lodging_Hotel	46800.0	1914.0	61.0	2.4	
406	2	State_1	Commercial	Lodging_Hotel	162214.0	1924.0	35.0	2.4	
407	2	State_1	Commercial	Lodging_Hotel	168000.0	1927.0	74.0	2.4	
411	2	State_1	Commercial	Lodging_Hotel	99000.0	1929.0	98.0	2.4	

5 rows × 64 columns

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Year_Factor                 0
State_Factor                0
building_class              0
facility_type               0
floor_area                  0
                           ..
direction_peak_wind_speed   0
max_wind_speed              0
days_with_fog               0
site_eui                    0
id                          0
Length: 64, dtype: int64
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11309 entries, 404 to 73148
Data columns (total 64 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Year_Factor               11309 non-null  int64  
 1   State_Factor              11309 non-null  object 
 2   building_class            11309 non-null  object 
 3   facility_type             11309 non-null  object 
 4   floor_area                11309 non-null  float64
 5   year_built                11309 non-null  float64
 6   energy_star_rating        11309 non-null  float64
 7   ELEVATION                 11309 non-null  float64
 8   january_min_temp          11309 non-null  int64  
 9   january_avg_temp          11309 non-null  float64
 10  january_max_temp          11309 non-null  int64  
 11  february_min_temp         11309 non-null  int64  
 12  february_avg_temp         11309 non-null  float64
 13  february_max_temp         11309 non-null  int64  
 14  march_min_temp            11309 non-null  int64  
 15  march_avg_temp            11309 non-null  float64
 16  march_max_temp            11309 non-null  int64  
 17  april_min_temp            11309 non-null  int64  
 18  april_avg_temp            11309 non-null  float64
 19  april_max_temp            11309 non-null  int64  
 20  may_min_temp              11309 non-null  int64  
 21  may_avg_temp              11309 non-null  float64
 22  may_max_temp              11309 non-null  int64  
 23  june_min_temp             11309 non-null  int64  
 24  june_avg_temp             11309 non-null  float64
 25  june_max_temp             11309 non-null  int64  
 26  july_min_temp             11309 non-null  int64  
 27  july_avg_temp             11309 non-null  float64
 28  july_max_temp             11309 non-null  int64  
 29  august_min_temp           11309 non-null  int64  
 30  august_avg_temp           11309 non-null  float64
 31  august_max_temp           11309 non-null  int64  
 32  september_min_temp        11309 non-null  int64  
 33  september_avg_temp        11309 non-null  float64
 34  september_max_temp        11309 non-null  int64  
 35  october_min_temp          11309 non-null  int64  
```

```
 36  october_avg_temp           11309 non-null  float64
 37  october_max_temp           11309 non-null  int64
 38  november_min_temp          11309 non-null  int64
 39  november_avg_temp          11309 non-null  float64
 40  november_max_temp          11309 non-null  int64
 41  december_min_temp          11309 non-null  int64
 42  december_avg_temp          11309 non-null  float64
 43  december_max_temp          11309 non-null  int64
 44  cooling_degree_days        11309 non-null  int64
 45  heating_degree_days        11309 non-null  int64
 46  precipitation_inches       11309 non-null  float64
 47  snowfall_inches            11309 non-null  float64
 48  snowdepth_inches           11309 non-null  int64
 49  avg_temp                   11309 non-null  float64
 50  days_below_30F             11309 non-null  int64
 51  days_below_20F             11309 non-null  int64
 52  days_below_10F             11309 non-null  int64
 53  days_below_0F              11309 non-null  int64
 54  days_above_80F             11309 non-null  int64
 55  days_above_90F             11309 non-null  int64
 56  days_above_100F            11309 non-null  int64
 57  days_above_110F            11309 non-null  int64
 58  direction_max_wind_speed   11309 non-null  float64
 59  direction_peak_wind_speed  11309 non-null  float64
 60  max_wind_speed             11309 non-null  float64
 61  days_with_fog              11309 non-null  float64
 62  site_eui                   11309 non-null  float64
 63  id                         11309 non-null  int64
dtypes: float64(24), int64(37), object(3)
memory usage: 5.6+ MB
```

In [8]:
```
df.describe()
```

Out[8]:

	Year_Factor	floor_area	year_built	energy_star_rating	ELEVATION	january_min_temp	january_avg_temp	january_max_temp	febru
count	11309.000000	1.130900e+04	11309.000000	11309.000000	11309.000000	11309.000000	11309.000000	11309.000000	
mean	4.239986	1.739833e+05	1950.422495	60.266160	25.995260	9.040676	32.028101	57.736847	
std	1.611790	2.394279e+05	36.709971	28.611072	35.606503	6.500818	5.151088	4.211625	
min	1.000000	8.019000e+03	0.000000	0.000000	1.800000	-9.000000	20.403226	44.000000	
25%	3.000000	6.683200e+04	1927.000000	39.000000	3.400000	6.000000	29.677419	56.000000	
50%	5.000000	9.835200e+04	1951.000000	66.000000	25.600000	8.000000	29.854839	56.000000	
75%	5.000000	1.725900e+05	1971.000000	84.000000	42.700000	11.000000	34.451613	59.000000	
max	6.000000	3.636683e+06	2015.000000	100.000000	201.800000	41.000000	55.096774	77.000000	

8 rows × 61 columns

In [9]:
```
df_cat=df.select_dtypes(exclude=np.number)
df_cat
```

Out[9]:

	State_Factor	building_class	facility_type
404	State_1	Residential	Mixed_Use_Predominantly_Commercial
405	State_1	Commercial	Lodging_Hotel
406	State_1	Commercial	Lodging_Hotel
407	State_1	Commercial	Lodging_Hotel
411	State_1	Commercial	Lodging_Hotel
...	...	...	...
73140	State_11	Commercial	Warehouse_Uncategorized
73143	State_11	Commercial	Warehouse_Uncategorized
73146	State_11	Commercial	Office_Uncategorized
73147	State_11	Commercial	Warehouse_Uncategorized
73148	State_11	Commercial	Warehouse_Uncategorized

11309 rows × 3 columns

In [10]:
```
df_en=pd.get_dummies(df_cat)
df_en
```

Out[10]:

	State_Factor_State_1	State_Factor_State_10	State_Factor_State_11	State_Factor_State_2	State_Factor_State_4	State_Factor_State_6	State_
404	1	0	0	0	0	0	
405	1	0	0	0	0	0	
406	1	0	0	0	0	0	
407	1	0	0	0	0	0	
411	1	0	0	0	0	0	
...	...	...	...	...	...	...	
73140	0	0	1	0	0	0	
73143	0	0	1	0	0	0	
73146	0	0	1	0	0	0	
73147	0	0	1	0	0	0	
73148	0	0	1	0	0	0	

11309 rows × 45 columns

In [11]:
```
df_new=pd.concat([df,df_en],axis=1)
df_new
```

Out[11]:

	Year_Factor	State_Factor	building_class	facility_type	floor_area	year_built	energy_star_rating	ELEVATION	janua
404	2	State_1	Residential	Mixed_Use_Predominantly_Commercial	34173.0	1913.0	100.0	2.4	
405	2	State_1	Commercial	Lodging_Hotel	46800.0	1914.0	61.0	2.4	
406	2	State_1	Commercial	Lodging_Hotel	162214.0	1924.0	35.0	2.4	
407	2	State_1	Commercial	Lodging_Hotel	168000.0	1927.0	74.0	2.4	
411	2	State_1	Commercial	Lodging_Hotel	99000.0	1929.0	98.0	2.4	
...	...	...	...	...	...	...	...	...	
73140	6	State_11	Commercial	Warehouse_Uncategorized	39984.0	1956.0	75.0	57.3	
73143	6	State_11	Commercial	Warehouse_Uncategorized	22324.0	1941.0	53.0	57.3	
73146	6	State_11	Commercial	Office_Uncategorized	26510.0	1970.0	91.0	57.3	
73147	6	State_11	Commercial	Warehouse_Uncategorized	22058.0	1967.0	1.0	57.3	
73148	6	State_11	Commercial	Warehouse_Uncategorized	37400.0	1922.0	73.0	57.3	

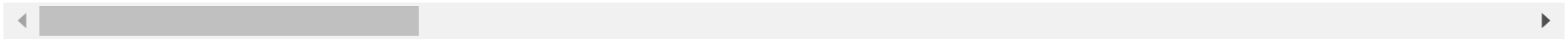
11309 rows × 109 columns

In [12]:
```
df_new.drop(df_cat,axis=1,inplace=True)
df_new
```

Out[12]:

	Year_Factor	floor_area	year_built	energy_star_rating	ELEVATION	january_min_temp	january_avg_temp	january_max_temp	february_min_
404	2	34173.0	1913.0	100.0	2.4	27	48.951613	69	
405	2	46800.0	1914.0	61.0	2.4	27	48.951613	69	
406	2	162214.0	1924.0	35.0	2.4	27	48.951613	69	
407	2	168000.0	1927.0	74.0	2.4	27	48.951613	69	
411	2	99000.0	1929.0	98.0	2.4	27	48.951613	69	
...	...	...	...	...	...	...	...	...	...
73140	6	39984.0	1956.0	75.0	57.3	28	43.451613	56	
73143	6	22324.0	1941.0	53.0	57.3	28	43.451613	56	
73146	6	26510.0	1970.0	91.0	57.3	28	43.451613	56	
73147	6	22058.0	1967.0	1.0	57.3	28	43.451613	56	
73148	6	37400.0	1922.0	73.0	57.3	28	43.451613	56	

11309 rows × 106 columns



In [13]:
```
df_new['cooling_degree_days']
```

Out[13]:
```
404      791
405      791
406      791
407      791
411      791
        ...
73140    260
73143    260
73146    260
73147    260
73148    260
Name: cooling_degree_days, Length: 11309, dtype: int64
```

In [14]:
```
from sklearn.impute import KNNImputer
from category_encoders import TargetEncoder
```

In [15]:
```
imputer = KNNImputer(n_neighbors=7)
df1 = pd.DataFrame(imputer.fit_transform(df_new),columns = df_new.columns)
df1
```

Out[15]:

	Year_Factor	floor_area	year_built	energy_star_rating	ELEVATION	january_min_temp	january_avg_temp	january_max_temp	february_min_
0	2.0	34173.0	1913.0	100.0	2.4	27.0	48.951613	69.0	
1	2.0	46800.0	1914.0	61.0	2.4	27.0	48.951613	69.0	
2	2.0	162214.0	1924.0	35.0	2.4	27.0	48.951613	69.0	
3	2.0	168000.0	1927.0	74.0	2.4	27.0	48.951613	69.0	
4	2.0	99000.0	1929.0	98.0	2.4	27.0	48.951613	69.0	
...	...	...	...	...	...	...	...	...	
11304	6.0	39984.0	1956.0	75.0	57.3	28.0	43.451613	56.0	
11305	6.0	22324.0	1941.0	53.0	57.3	28.0	43.451613	56.0	
11306	6.0	26510.0	1970.0	91.0	57.3	28.0	43.451613	56.0	
11307	6.0	22058.0	1967.0	1.0	57.3	28.0	43.451613	56.0	
11308	6.0	37400.0	1922.0	73.0	57.3	28.0	43.451613	56.0	

11309 rows × 106 columns

In [16]:
```
df1.isna().sum()
```

Out[16]:
```
Year_Factor                                          0
floor_area                                           0
year_built                                           0
energy_star_rating                                   0
ELEVATION                                            0
                                                    ..
facility_type_Warehouse_Distribution_or_Shipping_center    0
facility_type_Warehouse_Nonrefrigerated              0
facility_type_Warehouse_Refrigerated                 0
facility_type_Warehouse_Selfstorage                  0
facility_type_Warehouse_Uncategorized                0
Length: 106, dtype: int64
```

In [17]:
```
df_all_te = df1.copy()
temp = [col for col in df_all_te.columns if 'temp' in col]

df_all_te['min_temp'] = df_all_te[temp].min(axis=1)
df_all_te['max_temp'] = df_all_te[temp].max(axis=1)
df_all_te['avg_temp'] = df_all_te[temp].mean(axis=1)
df_all_te['std_temp'] = df_all_te[temp].std(axis=1)
df_all_te['skew_temp'] = df_all_te[temp].skew(axis=1)


# by seasons
temp = pd.Series([col for col in df_all_te.columns if 'temp' in col])

winter_temp = temp[temp.apply(lambda x: ('january' in x or 'february' in x or 'december' in x))].values
spring_temp = temp[temp.apply(lambda x: ('march' in x or 'april' in x or 'may' in x))].values
summer_temp = temp[temp.apply(lambda x: ('june' in x or 'july' in x or 'august' in x))].values
autumn_temp = temp[temp.apply(lambda x: ('september' in x or 'october' in x or 'november' in x))].values


### winter
df_all_te['min_winter_temp'] = df_all_te[winter_temp].min(axis=1)
df_all_te['max_winter_temp'] = df_all_te[winter_temp].max(axis=1)
df_all_te['avg_winter_temp'] = df_all_te[winter_temp].mean(axis=1)
df_all_te['std_winter_temp'] = df_all_te[winter_temp].std(axis=1)
df_all_te['skew_winter_temp'] = df_all_te[winter_temp].skew(axis=1)
### spring
df_all_te['min_spring_temp'] = df_all_te[spring_temp].min(axis=1)
df_all_te['max_spring_temp'] = df_all_te[spring_temp].max(axis=1)
df_all_te['avg_spring_temp'] = df_all_te[spring_temp].mean(axis=1)
df_all_te['std_spring_temp'] = df_all_te[spring_temp].std(axis=1)
df_all_te['skew_spring_temp'] = df_all_te[spring_temp].skew(axis=1)
### summer
df_all_te['min_summer_temp'] = df_all_te[summer_temp].min(axis=1)
df_all_te['max_summer_temp'] = df_all_te[summer_temp].max(axis=1)
df_all_te['avg_summer_temp'] = df_all_te[summer_temp].mean(axis=1)
df_all_te['std_summer_temp'] = df_all_te[summer_temp].max(axis=1)
df_all_te['skew_summer_temp'] = df_all_te[summer_temp].max(axis=1)
## autumn
df_all_te['min_autumn_temp'] = df_all_te[autumn_temp].min(axis=1)
df_all_te['max_autumn_temp'] = df_all_te[autumn_temp].max(axis=1)
df_all_te['avg_autumn_temp'] = df_all_te[autumn_temp].mean(axis=1)
```

```
df_all_te['std_autumn_temp'] = df_all_te[autumn_temp].std(axis=1)
df_all_te['skew_autumn_temp'] = df_all_te[autumn_temp].skew(axis=1)
```

In [18]:
```
df_all_te['month_cooling_degree_days'] = df_all_te['cooling_degree_days']/12
df_all_te['month_heating_degree_days'] = df_all_te['heating_degree_days']/12
```

In [19]:
```
df_all_te[temp]
```

Out[19]:

	january_min_temp	january_avg_temp	january_max_temp	february_min_temp	february_avg_temp	february_max_temp	march_min_temp	ma
0	27.0	48.951613	69.0	33.0	51.655172	78.0	34.0	
1	27.0	48.951613	69.0	33.0	51.655172	78.0	34.0	
2	27.0	48.951613	69.0	33.0	51.655172	78.0	34.0	
3	27.0	48.951613	69.0	33.0	51.655172	78.0	34.0	
4	27.0	48.951613	69.0	33.0	51.655172	78.0	34.0	
...	...	...	...	...	...	...	...	
11304	28.0	43.451613	56.0	34.0	47.672414	64.0	35.0	
11305	28.0	43.451613	56.0	34.0	47.672414	64.0	35.0	
11306	28.0	43.451613	56.0	34.0	47.672414	64.0	35.0	
11307	28.0	43.451613	56.0	34.0	47.672414	64.0	35.0	
11308	28.0	43.451613	56.0	34.0	47.672414	64.0	35.0	

11309 rows × 41 columns

In [20]:
```
df_all_te.cooling_degree_days
```

Out[20]:
```
0        791.0
1        791.0
2        791.0
3        791.0
4        791.0
         ...
11304    260.0
11305    260.0
11306    260.0
11307    260.0
11308    260.0
Name: cooling_degree_days, Length: 11309, dtype: float64
```

In [21]:
```
tmp = df[['State_Factor', 'building_class', 'facility_type', 'site_eui']]
df_all = df.drop(tmp.columns, axis=1)
```

In [22]:
```
df_all_te = df.copy()

cats = ['State_Factor', 'building_class', 'facility_type']
for col in cats:
    encoder = TargetEncoder()
    df_all_te[f'te_{col}'] = encoder.fit_transform(df_all_te[col], df_all_te['site_eui'])
```

In [23]:
```
# total area
df_all_te['building_area'] = df_all_te['floor_area'] * df_all_te['ELEVATION']
# rating energy by floor
df_all_te['floor_energy_star_rating'] = df_all_te['energy_star_rating']/df_all_te['ELEVATION']
```

In [24]:
```
df_all_te[["floor_area", "ELEVATION", "energy_star_rating", "floor_energy_star_rating", "building_area"]]
```

Out[24]:

	floor_area	ELEVATION	energy_star_rating	floor_energy_star_rating	building_area
404	34173.0	2.4	100.0	41.666667	82015.2
405	46800.0	2.4	61.0	25.416667	112320.0
406	162214.0	2.4	35.0	14.583333	389313.6
407	168000.0	2.4	74.0	30.833333	403200.0
411	99000.0	2.4	98.0	40.833333	237600.0
...	...	...	...	...	...
73140	39984.0	57.3	75.0	1.308901	2291083.2
73143	22324.0	57.3	53.0	0.924956	1279165.2
73146	26510.0	57.3	91.0	1.588133	1519023.0
73147	22058.0	57.3	1.0	0.017452	1263923.4
73148	37400.0	57.3	73.0	1.273997	2143020.0

11309 rows × 5 columns

In [25]:
```
df2=df_all_te.copy()
```

In [26]:
```
df2.head()
```

Out[26]:

	Year_Factor	State_Factor	building_class	facility_type	floor_area	year_built	energy_star_rating	ELEVATION	january
404	2	State_1	Residential	Mixed_Use_Predominantly_Commercial	34173.0	1913.0	100.0	2.4	
405	2	State_1	Commercial	Lodging_Hotel	46800.0	1914.0	61.0	2.4	
406	2	State_1	Commercial	Lodging_Hotel	162214.0	1924.0	35.0	2.4	
407	2	State_1	Commercial	Lodging_Hotel	168000.0	1927.0	74.0	2.4	
411	2	State_1	Commercial	Lodging_Hotel	99000.0	1929.0	98.0	2.4	

5 rows × 69 columns

In [27]:
```
df2.drop(['Year_Factor','State_Factor','building_class','facility_type'],axis=1,inplace=True)
```

In [28]:
```
len(df2)
```

Out[28]:
```
11309
```

In [29]:
```
from sklearn.model_selection import train_test_split
```

In [30]:
```
x=df2
y=df['site_eui']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [31]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[31]:
```
((9047, 65), (2262, 65), (9047,), (2262,))
```

In [32]:
```
from catboost import CatBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression
```

In [33]:
```
xg=XGBRegressor()
model=xg.fit(x_train,y_train)
```

In [34]:
```
pred=model.predict(x_test)
```

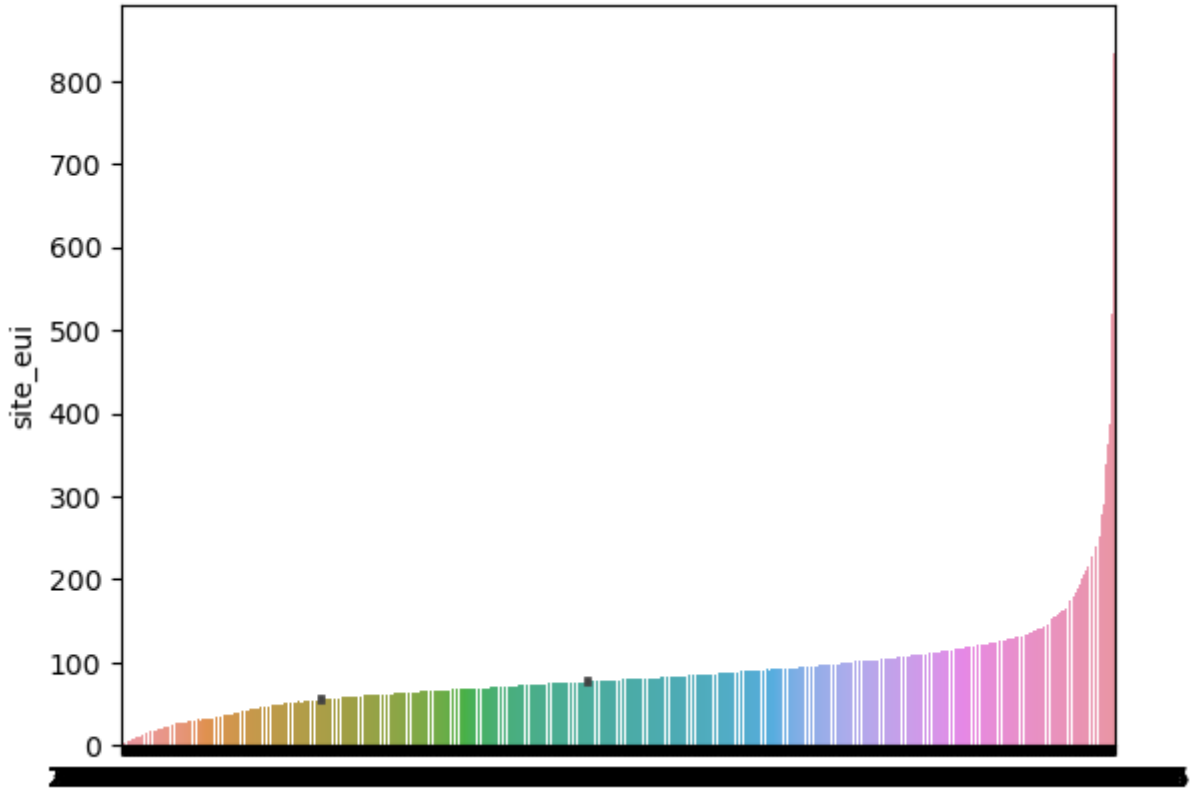
In [35]:
```
model.score(x_train,y_train)
```

Out[35]:
```
0.9999923515570881
```

In [36]:
```
import seaborn as sns
```

In [37]:
```
sns.barplot(x=pred,y=y_test)
```

Out[37]: <Axes: ylabel='site_eui'>



In [38]:
```
from sklearn.model_selection import GridSearchCV
```

In [39]:
```
model_params={
    'catboostregresor':{
        'model':CatBoostRegressor(),
        'params':{
            'loss_function':['RMSE','MAE','Quantile:alpha=value','LogLinQuantile:alpha=value','Poisson']
        }
    },
    'xgboost':{
        'model':XGBRegressor(),
        'params':{
            'n_estimators':[100,200,300]
        }
    },
    'Linearregresion':{
        'model':LinearRegression(),
        'params':{}
    },
    'randomforest':{
        'model':RandomForestRegressor(),
        'params':{
            'n_estimators':[100,200,300]
        }
    }
}
```

In [41]:
```
score=[]
for models,model_f in model_params.items():
    gs=GridSearchCV(model_f['model'],model_f['params'],cv=5)
    clf=gs.fit(x_train,y_train)
    score.append({'model':models,'params':clf.best_params_,'accuracy':clf.best_score_})
```

```
432:    learn: 1.4286676        total: 1.99s    remaining: 2.6s
433:    learn: 1.4279253        total: 1.99s    remaining: 2.59s
434:    learn: 1.4236668        total: 1.99s    remaining: 2.59s
435:    learn: 1.4193439        total: 2s       remaining: 2.58s
436:    learn: 1.4178617        total: 2s       remaining: 2.58s
437:    learn: 1.4159913        total: 2s       remaining: 2.57s
438:    learn: 1.4120873        total: 2.01s    remaining: 2.57s
439:    learn: 1.4109190        total: 2.01s    remaining: 2.56s
440:    learn: 1.4085113        total: 2.02s    remaining: 2.56s
441:    learn: 1.4073307        total: 2.02s    remaining: 2.55s
442:    learn: 1.4047709        total: 2.03s    remaining: 2.55s
443:    learn: 1.4033696        total: 2.03s    remaining: 2.54s
444:    learn: 1.3992966        total: 2.03s    remaining: 2.54s
445:    learn: 1.3977175        total: 2.04s    remaining: 2.53s
446:    learn: 1.3934507        total: 2.06s    remaining: 2.54s
447:    learn: 1.3894592        total: 2.06s    remaining: 2.54s
448:    learn: 1.3878642        total: 2.07s    remaining: 2.54s
449:    learn: 1.3865175        total: 2.07s    remaining: 2.53s
450:    learn: 1.3828825        total: 2.07s    remaining: 2.52s
451:    learn: 1.3812622        total: 2.08s    remaining: 2.52s
```

In [42]:
```
score
```

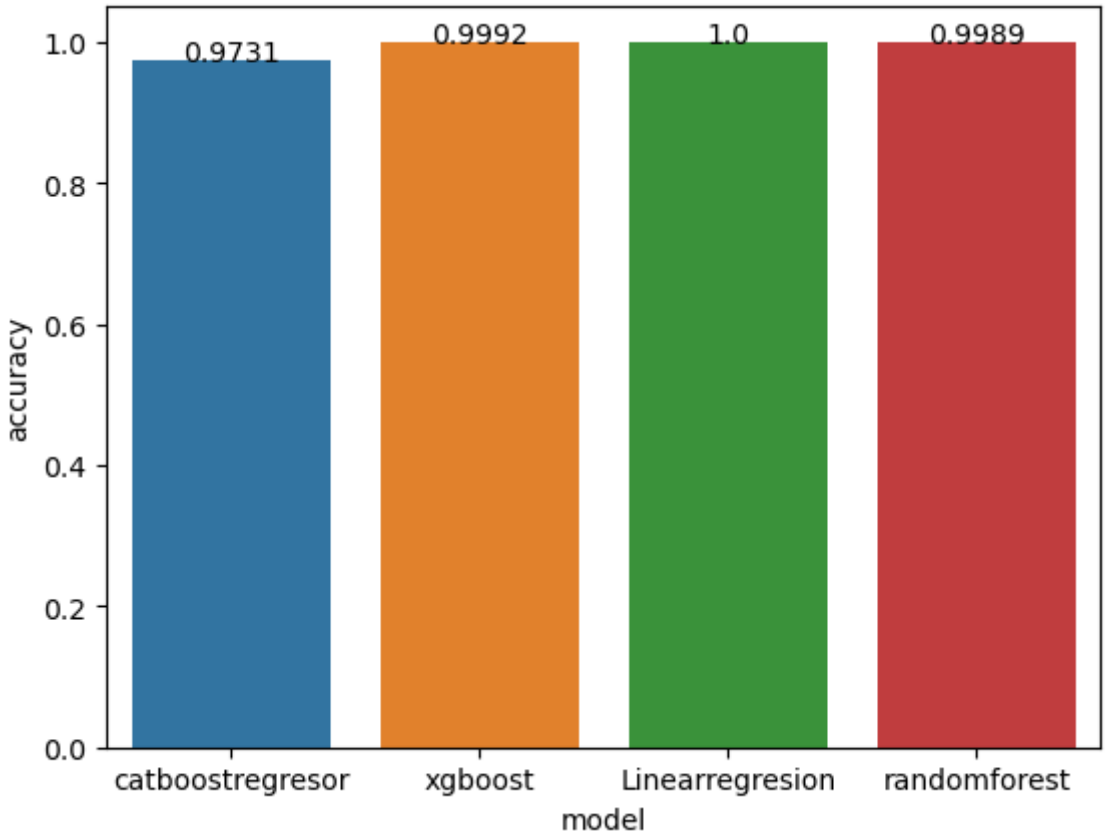
Out[42]:
```
[{'model': 'catboostregresor',
  'params': {'loss_function': 'RMSE'},
  'accuracy': 0.9731014235094964},
 {'model': 'xgboost',
  'params': {'n_estimators': 300},
  'accuracy': 0.9991644338232544},
 {'model': 'Linearregresion', 'params': {}, 'accuracy': 1.0},
 {'model': 'randomforest',
  'params': {'n_estimators': 300},
  'accuracy': 0.9988719951589589}]
```

In [43]:
```
model=pd.DataFrame(score)
```

In [63]:
```
o=round(model['accuracy'],4)
o
```

Out[63]:
```
0    0.9731
1    0.9992
2    1.0000
3    0.9989
Name: accuracy, dtype: float64
```

In [64]:
```
box=sns.barplot(y=model['accuracy'],x=model['model'])
for i in range(4):
    box.annotate(str(o[i]),xy=(i,o[i]),horizontalalignment='center')
```



In [ ]:
```
```