```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/3rd-programming-competition-bahrain-ai/house_data_train4.csv
/kaggle/input/3rd-programming-competition-bahrain-ai/house_data_test4.csv
```

```python
from sklearn.preprocessing import StandardScaler
```

```python
from xgboost import XGBRegressor
```

```python
# Load the training and testing datasets
train_data = pd.read_csv('/kaggle/input/3rd-programming-competition-bahrain-ai/house_data_train4.csv')
test_data = pd.read_csv('/kaggle/input/3rd-programming-competition-bahrain-ai/house_data_test4.csv')
```

```python
# Separate the features and target variable for both datasets
train_features = train_data.drop(['id', 'sale_price','waterfront'], axis=1)
train_target = train_data['sale_price']
test_features = test_data.drop(['id','waterfront'], axis=1)
test_target = test_data['id']
```

```python
scaler = StandardScaler()
```

```python
train_features = scaler.fit_transform(train_features)
test_features = scaler.transform(test_features)
```

```
model = XGBRegressor(colsample_bytree=0.8, learning_rate=0.1,
max_depth=10, n_estimators=1050, verbosity=0)

model.fit(train_features, train_target)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.8, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
             interaction_constraints=None, learning_rate=0.1,
max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=10, max_leaves=None,
             min_child_weight=None, missing=nan,
monotone_constraints=None,
             n_estimators=1050, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

```
# Model prediction on train data
y_pred = model.predict(test_features)
```

```
from sklearn.metrics import mean_squared_error

# Calculate the mean squared error (MSE)
mse = mean_squared_error(test_target, y_pred)

# Calculate the root mean squared error (RMSE)
rmse = np.sqrt(mse)


print("MSE:", mse)
print("RMSE:", rmse)
```

```
MSE: 371308664744.93445
RMSE: 609351.0193188606
```

```
from sklearn.model_selection import cross_val_score
rfr_cv = cross_val_score(model,test_target,y_pred, cv = 5, scoring =
'r2')
print("R2: ", rfr_cv.mean())
```

```
R2:  -0.2436837643627931
```

```
#now converting the results to a dataframe
print(type(y_pred))
df = pd.DataFrame(y_pred, columns = ['price'])
print(type(df))
df['price'].round()
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

```
0        1082433.0
1         460117.0
2         492518.0
3         556933.0
4         590744.0
            ...   
7243      282772.0
7244      222189.0
7245      190944.0
7246      210869.0
7247      115205.0
Name: price, Length: 7248, dtype: float32
```

```
df2 = pd.DataFrame(test_data, columns = ['id'])
print(type(df2))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
#adding the Id column to the dataframe
submission = pd.concat([df2,df.abs()],axis=1)
submission.to_csv('/kaggle/working/submission.csv',index=False) # save to notebook output
print(submission)
```

```
         id         price
0     17292  1.082433e+06
1     17293  4.601169e+05
2     17294  4.925184e+05
3     17295  5.569333e+05
4     17296  5.907437e+05
...     ...           ...
7243  36230  2.827721e+05
7244  36231  2.221888e+05
7245  36232  1.909443e+05
7246  36233  2.108687e+05
7247  36234  1.152053e+05

[7248 rows x 2 columns]
```