PiotrKuczko

March 11, 2021

1 House Sales in King County, USA

1.0.1 Predict house price using regression

[21]:

```
import numpy as np
import pandas as pd
import xgboost as xg
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn import preprocessing
import xgboost as xgb
from xgboost.sklearn import XGBRegressor
import datetime
from sklearn.model_selection import GridSearchCV
```

[22]:

```
df = pd.read_csv('kc_house_data.csv')
df
```

[22]:

```
               id             date     price  bedrooms  bathrooms  \
0      7129300520  20141013T000000  221900.0         3       1.00
1      6414100192  20141209T000000  538000.0         3       2.25
2      5631500400  20150225T000000  180000.0         2       1.00
3      2487200875  20141209T000000  604000.0         4       3.00
4      1954400510  20150218T000000  510000.0         3       2.00
…            …              …        …       …        …
21608   263000018  20140521T000000  360000.0         3       2.50
21609  6600060120  20150223T000000  400000.0         4       2.50
21610  1523300141  20140623T000000  402101.0         2       0.75
21611   291310100  20150116T000000  400000.0         3       2.50
21612  1523300157  20141015T000000  325000.0         2       0.75

       sqft_living  sqft_lot  floors  waterfront  view  …  grade  \
0             1180      5650     1.0           0     0  …      7
1             2570      7242     2.0           0     0  …      7
2              770     10000     1.0           0     0  …      6
3             1960      5000     1.0           0     0  …      7
4             1680      8080     1.0           0     0  …      8
…             …       …     …         …    …  …    …
```

```
21608         1530     1131     3.0           0     0  ...    8
21609         2310     5813     2.0           0     0  ...    8
21610         1020     1350     2.0           0     0  ...    7
21611         1600     2388     2.0           0     0  ...    8
21612         1020     1076     2.0           0     0  ...    7

       sqft_above  sqft_basement  yr_built  yr_renovated  zipcode      lat  \
0            1180              0      1955             0    98178  47.5112
1            2170            400      1951          1991    98125  47.7210
2             770              0      1933             0    98028  47.7379
3            1050            910      1965             0    98136  47.5208
4            1680              0      1987             0    98074  47.6168
...           ...            ...       ...           ...      ...      ...
21608        1530              0      2009             0    98103  47.6993
21609        2310              0      2014             0    98146  47.5107
21610        1020              0      2009             0    98144  47.5944
21611        1600              0      2004             0    98027  47.5345
21612        1020              0      2008             0    98144  47.5941

          long  sqft_living15  sqft_lot15
0     -122.257           1340        5650
1     -122.319           1690        7639
2     -122.233           2720        8062
3     -122.393           1360        5000
4     -122.045           1800        7503
...        ...            ...         ...
21608 -122.346           1530        1509
21609 -122.362           1830        7200
21610 -122.299           1020        2007
21611 -122.069           1410        1287
21612 -122.299           1020        1357

[21613 rows x 21 columns]
```

[23]:
```
df[['year', 'month', 'day']] = pd.DataFrame([ [int(x[0:4]), int(x[4:6]),␣
 ↪int(x[6:8])] for x in df['date'].tolist() ])
df = df.drop(['date', 'id'], axis=1)
Y = df['price']
X = df.drop(['price'], axis=1)
```

[24]:
```
train_X, test_X, train_Y, test_Y = train_test_split(X, Y,
                      test_size = 0.3, random_state = 123)
```

[28]:
```
xgb1 = XGBRegressor()
parameters = {'nthread':[1], #when use hyperthread, xgboost may become slower
              'objective':['reg:squarederror'],
              'learning_rate': [.03, .04, .05, .06, .07], #so called `eta` value
              'max_depth': [3, 4, 5, 6, 7],
              'min_child_weight': [3, 4, 5, 6],
              'subsample': [0.7],
              'colsample_bytree': [0.7],
              'n_estimators': [500, 700, 100]}

xgb_grid = GridSearchCV(xgb1,
                        parameters,
                        cv = 2,
                        n_jobs = 12,
                        verbose=True)

xgb_grid.fit(X, Y)

print(xgb_grid.best_score_)
print(xgb_grid.best_params_)
```

```
Fitting 2 folds for each of 300 candidates, totalling 600 fits
0.877865911661787
{'colsample_bytree': 0.7, 'learning_rate': 0.06, 'max_depth': 5,
'min_child_weight': 6, 'n_estimators': 700, 'nthread': 1, 'objective':
'reg:squarederror', 'subsample': 0.7}
```

[31]:
```
xgb2 = XGBRegressor(colsample_bytree=0.7, learning_rate=0.06, max_depth=5,␣
 ↪min_child_weight=6, n_estimators=700, nthread=1, objective='reg:
 ↪squarederror', subsample=0.7)
```

[36]:
```
xgb2.fit(train_X, train_Y)
score = xgb2.score(train_X, train_Y)
print("Training score: ", score)
score = xgb2.score(test_X, test_Y)
print("Test score: ", score)
pred = xgb2.predict(test_X)
rmse = np.sqrt(MSE(test_Y, pred))
print("RMSE : % f" %(rmse))
```

```
Training score:  0.9711698297228819
Test score:  0.905722369519171
RMSE :  113149.749407
```

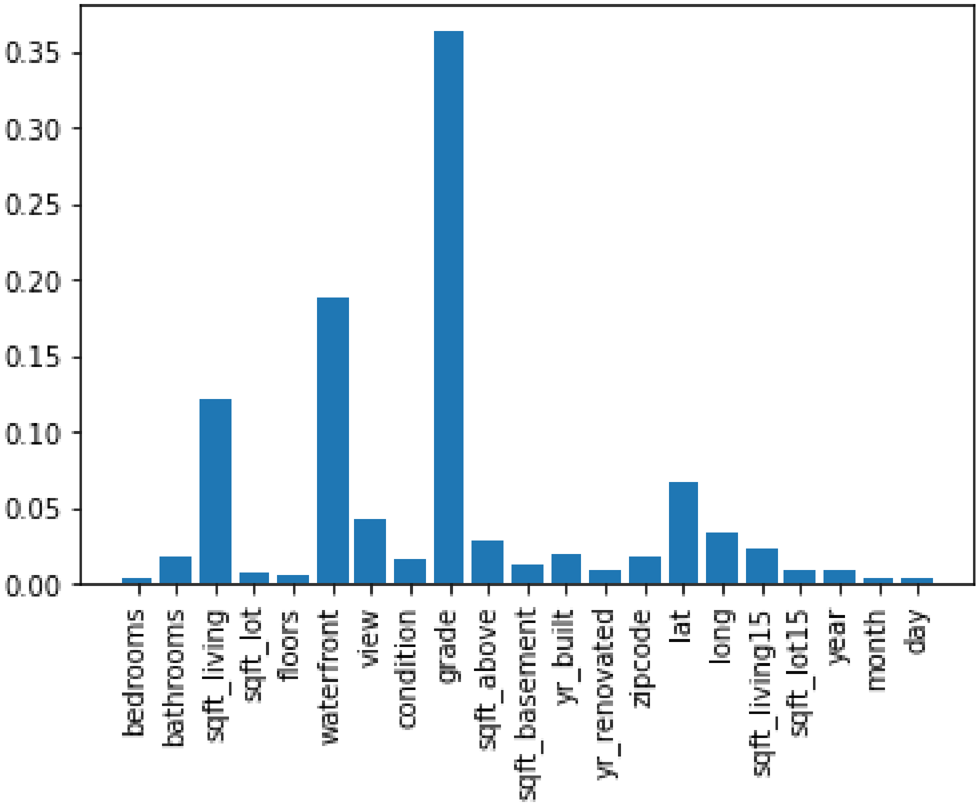
[37]:
```
print (test_Y[:10], pred[:10])
```

```
5506     532500.0
9279     410000.0
16034    782500.0
6608     995000.0
20359    279000.0
8798     175000.0
```

```
10035    689000.0
13321    275000.0
15842    465000.0
12119    506000.0
Name: price, dtype: float64 [617686.7  525400.25 869056.5  986680.4  302756.4
301077.2  696768.8
 419573.34 457880.12 462333.66]
```

[45]:

```
import matplotlib.pyplot as pyplot
pyplot.bar(X.columns, xgb2.feature_importances_)
pyplot.xticks(rotation=90)
pyplot.show()
```



[ ]:

```
```