House Price Prediction

(1) Import Python Libraries

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

(2) Loading the Data Set

In [6]:
```
house_price_dataset = sklearn.datasets.fetch_california_housing()
print(house_price_dataset)
```

```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.5555
5556,
          37.88      , -122.23      ],
       [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
          37.86      , -122.22      ],
       [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
          37.85      , -122.24      ],
       ...,
       [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
          39.43      , -121.22      ],
       [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
          39.43      , -121.32      ],
       [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
          39.37      , -121.24      ]]), 'target': array([4.526, 3.585, 3.52
1, ..., 0.923, 0.847, 0.894]), 'frame': None, 'target_names': ['MedHouseVa
l'], 'feature_names': ['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Popul
ation', 'AveOccup', 'Latitude', 'Longitude'], 'DESCR': '.. _california_housi
ng_dataset:\n\nCalifornia Housing dataset\n--------------------------\n\n**D
ata Set Characteristics:**\n\n    :Number of Instances: 20640\n\n    :Number
of Attributes: 8 numeric, predictive attributes and the target\n\n    :Attri
bute Information:\n        - MedInc        median income in block\n        -
HouseAge      median house age in block\n        - AveRooms      average num
ber of rooms\n        - AveBedrms     average number of bedrooms\n        -
Population    block population\n        - AveOccup      average house occupa
ncy\n        - Latitude      house block latitude\n        - Longitude     h
ouse block longitude\n\n    :Missing Attribute Values: None\n\nThis dataset
was obtained from the StatLib repository.\nhttp://lib.stat.cmu.edu/datasets/
\n\nThe target variable is the median house value for California district
s.\n\nThis dataset was derived from the 1990 U.S. census, using one row per
census\nblock group. A block group is the smallest geographical unit for whi
ch the U.S.\nCensus Bureau publishes sample data (a block group typically ha
s a population\nof 600 to 3,000 people).\n\nIt can be downloaded/loaded usin
g the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n.. top
ic:: References\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Au
toregressions,\n      Statistics and Probability Letters, 33 (1997) 291-297
\n'}
```

In [7]:
```
house_price_dataframe = pd.DataFrame(house_price_dataset.data, columns = hou
house_price_dataframe.head()
```

Out[7]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85

In [8]:
```
house_price_dataframe['price'] = house_price_dataset.target
house_price_dataframe.head()
```

Out[8]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85

(3) Exploring the Data Set

In [9]:
```
house_price_dataframe.shape
```

Out[9]:
```
(20640, 9)
```

In [10]:
```
house_price_dataframe.isnull().sum()
```

Out[10]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
price         0
dtype: int64
```

In [11]:
```
house_price_dataframe.describe()
```

Out[11]:

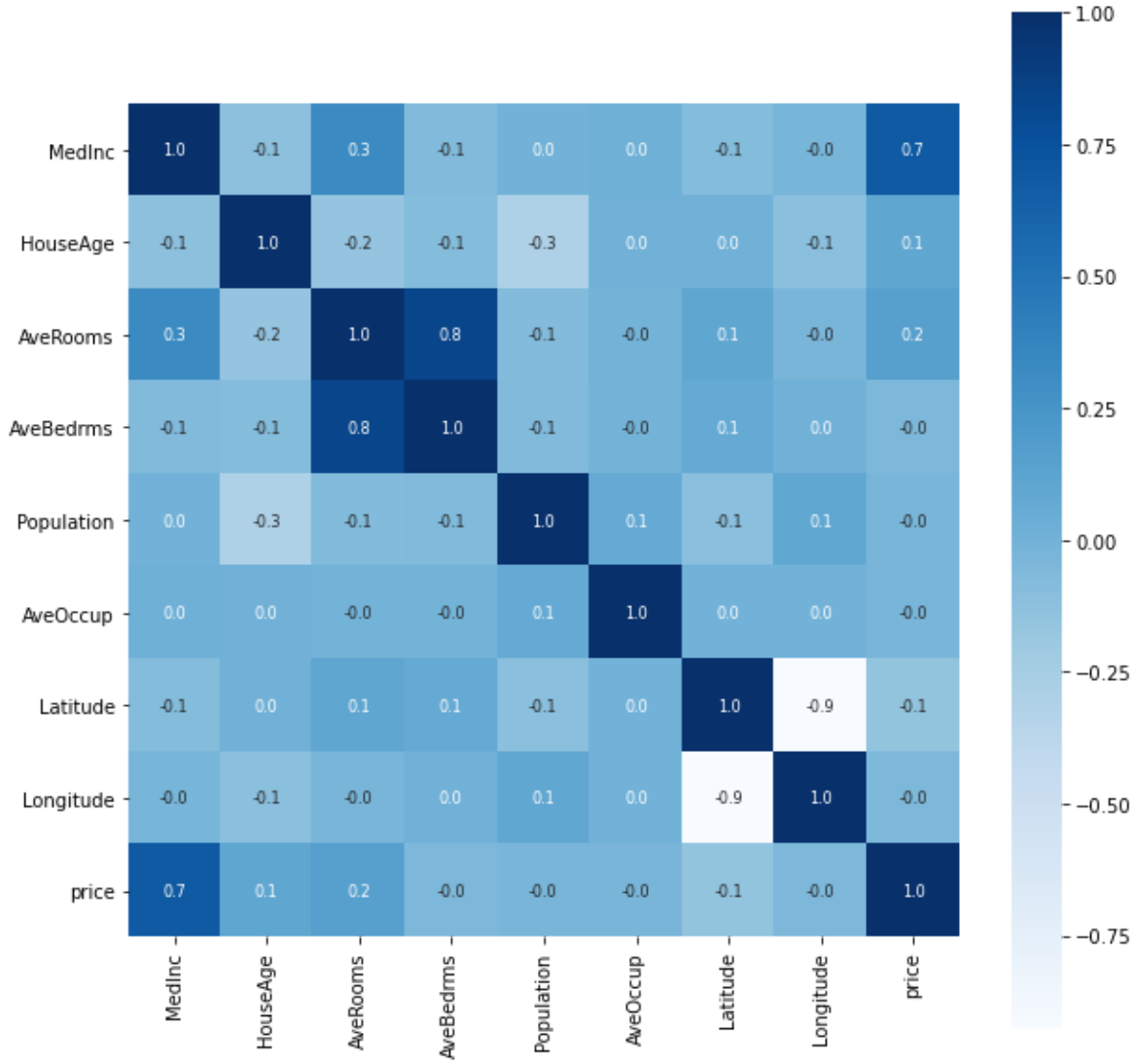
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	La
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	2064
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	3
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	3
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	3
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	3
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	3
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	4

(4) Finding Correlations in the Data Set

In [12]:
```
correlation = house_price_dataframe.corr()
```

In [13]:
```
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, anno
```

Out[13]: <AxesSubplot:>



In [14]:
```
print(correlation['price'])
```

```
MedInc        0.688075
HouseAge      0.105623
AveRooms      0.151948
AveBedrms    -0.046701
Population   -0.024650
AveOccup     -0.023737
Latitude     -0.144160
Longitude    -0.045967
price         1.000000
Name: price, dtype: float64
```

(5) Building the Model

Split the Data

In [15]:
```
X = house_price_dataframe.drop(['price'], axis=1)
Y = house_price_dataframe['price']
```

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, r
```

Create and Fit the Model

In [17]:
```
model = XGBRegressor()
model.fit(X_train, Y_train)
```

Out[17]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=Non
e,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

(6) Evaluating the Model

In [18]:
```
training_data_prediction = model.predict(X_train)
training_data_prediction[:5]
```

Out[18]:
```
array([0.6893792 , 2.986824  , 0.48874274, 2.3740659 , 1.5502008 ],
      dtype=float32)
```

In [19]:
```
test_data_prediction = model.predict(X_test)
test_data_prediction[:5]
```

Out[19]:
```
array([2.787383 , 1.9628428, 0.782536 , 3.9288697, 3.9321907],
      dtype=float32)
```

R-squared Error

In [20]:
```
score_1 = metrics.r2_score(Y_train, training_data_prediction)
print("R squared error : ", score_1)
```

```
R squared error :  0.9451221492760822
```

In [21]:
```
score_1 = metrics.r2_score(Y_test, test_data_prediction)
print("R squared error : ", score_1)
```

```
R squared error :  0.8412904408180302
```

Mean Absolute Error

In [22]:
```
score_2 = metrics.mean_absolute_error(Y_train, training_data_prediction)
print('Mean Absolute Error : ', score_2)
```

```
Mean Absolute Error :  0.1919170860794262
```

In [23]:
```
score_2 = metrics.mean_absolute_error(Y_test, test_data_prediction)
print('Mean Absolute Error : ', score_2)
```

```
Mean Absolute Error :  0.30753655785801337
```

(7) Visualizing the Model

In [26]:
```
plt.scatter(Y_train, training_data_prediction)

plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Price vs Preicted Price")

plt.show()
```

In [27]:
```
plt.scatter(Y_test, test_data_prediction)

plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Price vs Preicted Price")

plt.show()
```