In [11]:

```
!pip install -q hvplot
```

In [13]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import hvplot.pandas
%matplotlib inline

# sns.set_style("whitegrid")
# plt.style.use("fivethirtyeight")
```

Explore the Data

In [68]:

```
housing = pd.read_csv('Housing.csv')
housing.head()

housing.shape
```

Out[68]:

```
(5000, 7)
```

In [69]:

```
housing.dropna()
```

Out[69]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5000 rows × 7 columns

In [34]:

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [35]:

```
housing.describe()
```

Out[35]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

In [17]:

```
housing.columns
```

Out[17]:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [55]:

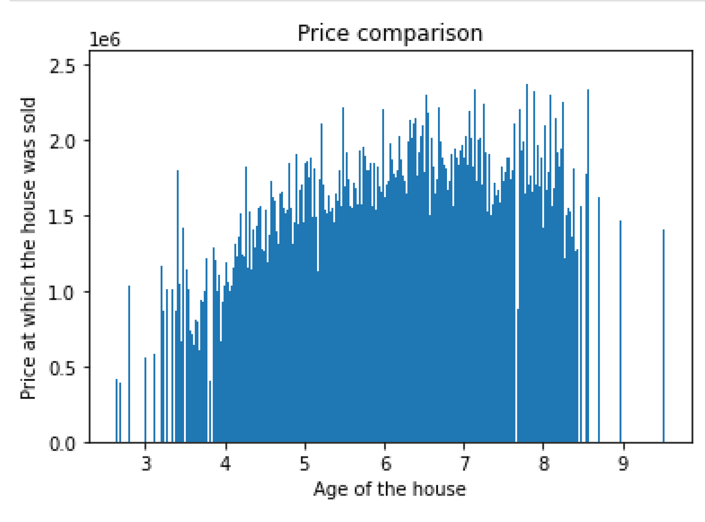
```
age = housing['Avg. Area House Age']
price = housing['Price']

plt.bar(age, price, width = 0.01)

plt.xlabel('Age of the house')
plt.ylabel('Price at which the house was sold')

#label the figure
plt.title('Price comparison')


plt.show()
```



Data Analysis

Let's create some simple plots to check out the data!

In [36]:

```
housing.hvplot.hist(by='Price',  width=1000)
```

Out[36]:

In [19]:

```
housing.hvplot.hist("Price")
```

Out[19]:

In [20]:

```
housing.hvplot.scatter(x='Avg. Area House Age', y='Price')
```

Out[20]:

In [21]:

```
housing.hvplot.scatter(x='Avg. Area Income', y='Price')
```

Out[21]:

In [22]:

```
housing.columns
```

Out[22]:

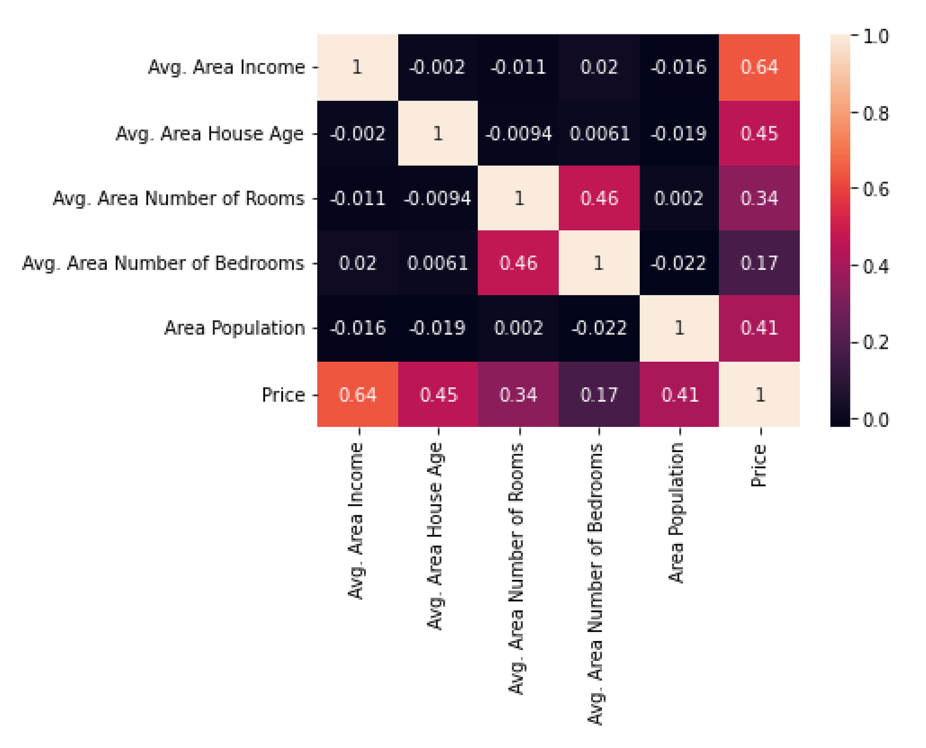
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [23]:

```
sns.heatmap(housing.corr(), annot=True)
```

Out[23]:

```
<AxesSubplot:>
```



Training the Linear Regression Model

X and y arrays

In [39]:

```
X = housing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
               'Avg. Area Number of Bedrooms', 'Area Population']]
y = housing['Price']
```

Train Test Split

Now let's split the data into a training set and a testing set. We will train out model on the training set and then use the test set to evaluate the model.

In [40]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [56]:

```
from sklearn import metrics
from sklearn.model_selection import cross_val_score

def cross_val(model):
    pred = cross_val_score(model, X, y, cv=10)
    return pred.mean()

def print_evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    print('MAE:', mae)
    print('MSE:', mse)
    print('RMSE:', rmse)
    print('R2 Square', r2_square)
    print('__________________________________')

def evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    return mae, mse, rmse, r2_square
```

In [57]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('std_scalar', StandardScaler())
])

X_train = pipeline.fit_transform(X_train)
X_test = pipeline.transform(X_test)
```

Linear Regression

In [58]:

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)
```

Out[58]:

```
LinearRegression(normalize=True)
```

Model Evaluation

Let's evaluate the model by checking out it's coefficients and how we can interpret them.

In [59]:

```
# print the intercept
print(lin_reg.intercept_)
```

```
1228219.1492415662
```

In [60]:

```
coeff_df = pd.DataFrame(lin_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[60]:

	Coefficient
Avg. Area Income	232679.724643
Avg. Area House Age	163841.046593
Avg. Area Number of Rooms	121110.555478
Avg. Area Number of Bedrooms	2892.815119
Area Population	151252.342377

> Interpreting the coefficients:
>
> - Holding all other features fixed, a 1 unit increase in **Avg. Area Income** is associated with an **increase of $23.26**.
> - Holding all other features fixed, a 1 unit increase in **Avg. Area House Age** is associated with an **increase of $163841.04**.
> - Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Rooms** is associated with an **increase of $121110.55**.
> - Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Bedrooms** is associated with an **increase of $2892.81**.
> - Holding all other features fixed, a 1 unit increase in **Area Population** is associated with an **increase of $15.12**.

✔ Predictions from our Model

Let's grab predictions off our test set and see how well it did!

In [61]:

```
pred = lin_reg.predict(X_test)
```

In [45]:

```
pd.DataFrame({'True Values': y_test, 'Predicted Values': pred}).hvplot.scatter(x='True Values', y='Predicted Va
```

Out[45]:

Residual Histogram

In [32]:

```
pd.DataFrame({'Error Values': (y_test - pred)}).hvplot.kde()
```

Out[32]:

In [62]:

```
test_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)

results_df = pd.DataFrame(data=[["Linear Regression", *evaluate(y_test, test_pred) , cross_val(LinearRegressior
                           columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', "Cross Validation"])
```

```
Test set evaluation:
_____________________________________
MAE: 81135.56609336878
MSE: 10068422551.40088
RMSE: 100341.52954485436
R2 Square 0.9146818498754016
__________________________________
Train set evaluation:
_____________________________________
MAE: 81480.49973174892
MSE: 10287043161.197224
RMSE: 101425.06180031257
R2 Square 0.9192986579075526
__________________________________
```

In [63]:

```
results_df
```

Out[63]:

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	81135.566093	1.006842e+10	100341.529545	0.914682	0.917379

In [ ]: