Yating Liao

ISE 599 Deep Learning Student ID: 7636428840

In [54]:
```
import pandas as pd
# read data
df = pd.read_csv('D:\Caroline\Documents\Graduate\ISE 599  Deep Learning\homes.csv')
print(df[:10])
# convert it to numpy array
array1 = df.values
print(array1[:10])
```

```
    price  area  beds  baths  garage  year  style  lotsize   ac pool quality  \
0  360000  3032     4      4       2  1972      1    22221  YES   NO  MEDIUM
1  340000  2058     4      2       2  1976      1    22912  YES   NO  MEDIUM
2  250000  1780     4      3       2  1980      1    21345  YES   NO  MEDIUM
3  205500  1638     4      2       2  1963      1    17342  YES   NO  MEDIUM
4  275500  2196     4      3       2  1968      7    21786  YES   NO  MEDIUM
5  248000  1966     4      3       5  1972      1    18902  YES  YES  MEDIUM
6  229900  2216     3      2       2  1972      7    18639  YES   NO  MEDIUM
7  150000  1597     2      1       1  1955      1    22112  YES   NO  MEDIUM
8  195000  1622     3      2       2  1975      1    14321  YES   NO     LOW
9  160000  1976     3      3       1  1918      1    32358   NO   NO     LOW

  highway
0      NO
1      NO
2      NO
3      NO
4      NO
5      NO
6      NO
7      NO
8      NO
9      NO
[[360000 3032 4 4 2 1972 1 22221 'YES' 'NO' 'MEDIUM' 'NO']
 [340000 2058 4 2 2 1976 1 22912 'YES' 'NO' 'MEDIUM' 'NO']
 [250000 1780 4 3 2 1980 1 21345 'YES' 'NO' 'MEDIUM' 'NO']
 [205500 1638 4 2 2 1963 1 17342 'YES' 'NO' 'MEDIUM' 'NO']
 [275500 2196 4 3 2 1968 7 21786 'YES' 'NO' 'MEDIUM' 'NO']
 [248000 1966 4 3 5 1972 1 18902 'YES' 'YES' 'MEDIUM' 'NO']
 [229900 2216 3 2 2 1972 7 18639 'YES' 'NO' 'MEDIUM' 'NO']
 [150000 1597 2 1 1 1955 1 22112 'YES' 'NO' 'MEDIUM' 'NO']
 [195000 1622 3 2 2 1975 1 14321 'YES' 'NO' 'LOW' 'NO']
 [160000 1976 3 3 1 1918 1 32358 'NO' 'NO' 'LOW' 'NO']]
```

fit a model with one predictor only (area)

In [55]:
```
import numpy as np
# get column price as a list
y = list(array1[:,0]) # y is actual price
print(y[:7])
# convert the list to a float64 (numeric) array
y = np.array(y, dtype=np.float64)
print(y[:7])
# get column area
x_area = list(array1[:,1])
x_area = np.array(x_area, dtype=np.float64)
x_area[:7]
# add column of ones to x
x1 = np.c_[np.ones(522),x_area]
print(x1[:7])
```

```
[360000, 340000, 250000, 205500, 275500, 248000, 229900]
[360000. 340000. 250000. 205500. 275500. 248000. 229900.]
[[1.000e+00 3.032e+03]
 [1.000e+00 2.058e+03]
 [1.000e+00 1.780e+03]
 [1.000e+00 1.638e+03]
 [1.000e+00 2.196e+03]
 [1.000e+00 1.966e+03]
 [1.000e+00 2.216e+03]]
```

In [56]:
```
b1 = np.dot(x1.T,x1)
b1 = np.linalg.inv(b1)
b1
```

Out[56]:
```
array([[ 2.13156377e-02, -8.58166061e-06],
       [-8.58166061e-06,  3.79614273e-09]])
```

In [57]:
```
b2 = np.dot(x1.T,y)
b2
```

Out[57]:
```
array([1.45060745e+08, 3.69799666e+11])
```

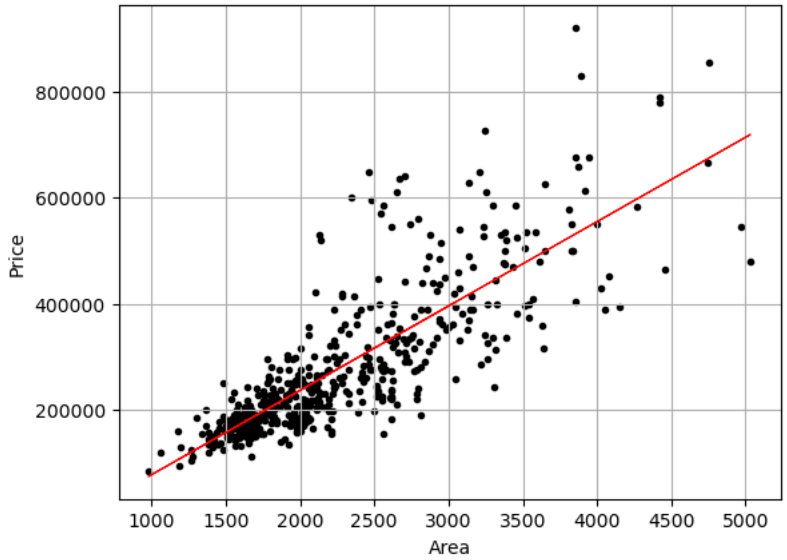
In [58]:
```
coeffs = np.dot(b1,b2)
coeffs
```

Out[58]:
```
array([-81432.94639555,    158.95023081])
```

The slope is the 158.95. For each additional square foot, the average price estimatedly increase 158.95 dollars.

3. (10 pts.) Draw a scatterplot of price (vertical axis) and area (horizontal axis). Add the regression (red) line on the scatterplot.

In [59]:
```
import matplotlib.pyplot as plt
yhat = np.dot(x1,coeffs)
plt.figure()
plt.scatter(x_area,y, c = 'k', s = 9)
plt.plot(x_area,yhat,color = 'red', linewidth = 0.8)
plt.xlabel("Area")
plt.ylabel("Price")
plt.grid()
plt.show()
```



4. (30 pts.) Fit a model with two predictors (area and quality). Get column area into an array as described above. Get column quality but it is non-numeric and needs to be preprocessed. We will onehot encode this column. First label-encode it, then convert the labels to binary columns

In [60]:
```
from sklearn import preprocessing

# Get column quality
quality = list(array1[:, 10])
area = list(array1[:, 1])
print("The Quality is these:",quality[:7])

# See quality categories
print("Quality Categories:",set(quality))

# Label encode (as integers) the quality categories: low, medium, and high
label_encoder = preprocessing.LabelEncoder()
quality = label_encoder.fit_transform(quality)
print("Encoded Qualtiy:",quality[:7])

# Convert integer labels to binary columns
quality2 = np.zeros((522, 3)) # Create a 522*3 numpy of zeros
quality2[np.arange(522), quality] = 1

# See the result of one-hot encoding
print("Quality 2:",quality2)
print("Quality 2 Shape:",quality2.shape)

# Reshape the area array to have the same number of dimensions as quality2
area = np.array(area, dtype=np.float64).reshape(-1, 1)

# Combine area and quality in a single array x
x_combined = np.hstack((area, quality2))
print("Area and Quality:",x_combined[:7])

# Delete last column
x_new = np.delete(x_combined, -1, axis=1)
print(x_new[:7])

# Add a column of ones
x2 = np.c_[np.ones(522), x_new]
print(x2[:5])
```

```
The Quality is these: ['MEDIUM', 'MEDIUM', 'MEDIUM', 'MEDIUM', 'MEDIUM', 'MEDIUM', 'MEDIUM']
Quality Categories: {'LOW', 'MEDIUM', 'HIGH'}
Encoded Qualtiy: [2 2 2 2 2 2 2]
Quality 2: [[0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 ...
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]]
Quality 2 Shape: (522, 3)
Area and Quality: [[3.032e+03 0.000e+00 0.000e+00 1.000e+00]
 [2.058e+03 0.000e+00 0.000e+00 1.000e+00]
 [1.780e+03 0.000e+00 0.000e+00 1.000e+00]
 [1.638e+03 0.000e+00 0.000e+00 1.000e+00]
 [2.196e+03 0.000e+00 0.000e+00 1.000e+00]
 [1.966e+03 0.000e+00 0.000e+00 1.000e+00]
 [2.216e+03 0.000e+00 0.000e+00 1.000e+00]]
[[3032.    0.    0.]
 [2058.    0.    0.]
 [1780.    0.    0.]
 [1638.    0.    0.]
 [2196.    0.    0.]
 [1966.    0.    0.]
 [2216.    0.    0.]]
[[1.000e+00 3.032e+03 0.000e+00 0.000e+00]
 [1.000e+00 2.058e+03 0.000e+00 0.000e+00]
 [1.000e+00 1.780e+03 0.000e+00 0.000e+00]
 [1.000e+00 1.638e+03 0.000e+00 0.000e+00]
 [1.000e+00 2.196e+03 0.000e+00 0.000e+00]]
```

In [61]:
```
b1 = np.dot(x2.T,x2)
b1 = np.linalg.inv(b1)
b1
```

Out[61]:
```
array([[ 4.45209798e-02, -1.76950462e-05,  1.47019978e-02,
        -1.43823239e-02],
       [-1.76950462e-05,  7.62342456e-09, -7.81954682e-06,
         4.71063425e-06],
       [ 1.47019978e-02, -7.81954682e-06,  2.61748728e-02,
        -1.38354543e-03],
       [-1.43823239e-02,  4.71063425e-06, -1.38354543e-03,
         1.24566120e-02]])
```

In [62]:
```
b2 = np.dot(x2.T,y)
b2
```

Out[62]:
```
array([1.45060745e+08, 3.69799666e+11, 3.69655240e+07, 2.87030000e+07])
```

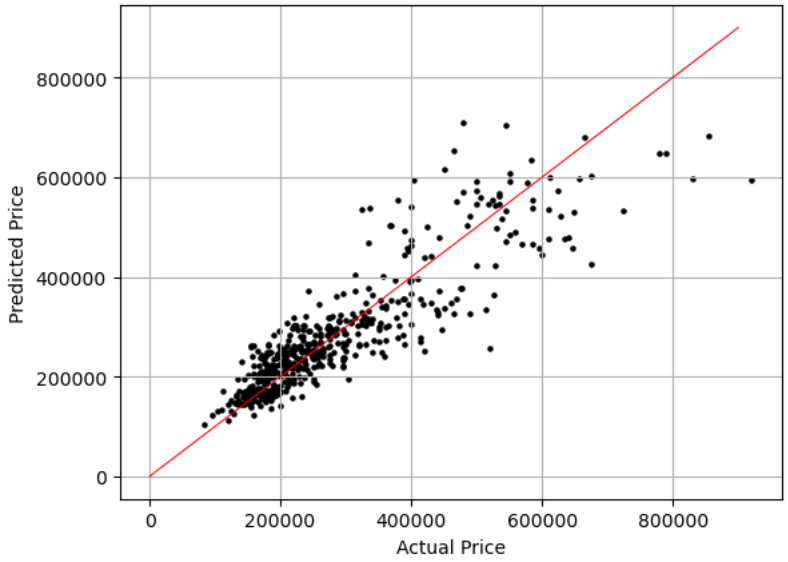
In [63]:
```
coeffs2 = np.dot(b1,b2)
coeffs2
```

Out[63]:
```
array([ 4.52755379e+04,  9.84389592e+01,  1.68872942e+05, -3.79210014e+04])
```

5. (10 pts.) Draw a scatterplot of price (horizontal axis) and predicted prices (vertical axis). Add a 45 degree (red) line on the scatterplot, as follows

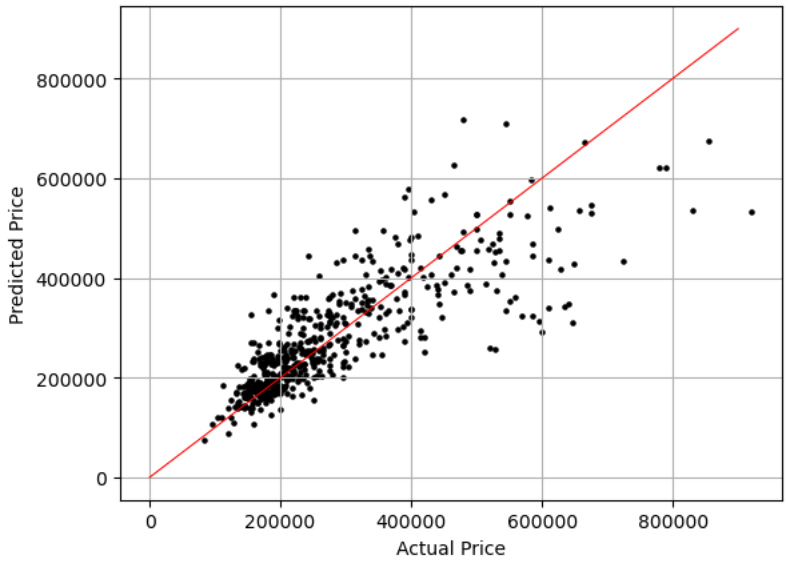
In [64]:
```
# predict the prices
yhat2 = np.dot(x2,coeffs2)
print(yhat2[:5])
# Set up x-axis and y-axis values for the diagonal line
xaxis = range(0, 900000)
yaxis = xaxis

# Draw the scatterplot with diagonal line
plt.scatter(y, yhat2, c='k', s=5)
plt.plot(xaxis, yaxis, c='r', linewidth=0.75)

plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.grid()
plt.show()
```

```
[343742.46204097 247862.9158191  220496.88517261 206518.55297188
 261447.49218318]
```



6. (10 pts.) Draw the same scatterplot for the first model (with predictor area only). Note that this scatterplot is not as close to the 45-degree line as that of the second model

In [65]:
```
yhat = np.dot(x1,coeffs)
plt.figure()
plt.scatter(y,yhat,c = 'k', s = 5)
plt.plot(xaxis,yaxis,color = 'red', linewidth = 0.75)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.grid()
plt.show()
```



7. (10 pts.) Use np.corrcoef(y,yhat) to find the correlation between prices and predicted prices of first model. Find the same correlation for the model with two predictors. Which model has higher correlation?

In [66]:
```
print(np.corrcoef(y,yhat))
print(np.corrcoef(y,yhat2))
```

```
[[1.         0.81947006]
 [0.81947006 1.        ]]
[[1.         0.88786964]
 [0.88786964 1.        ]]
```

By comparing two models, I find that the second model with two predictors has higher correlation(0.89)