California Housing Prices

October 26, 2023

[23]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from scipy.stats import pearsonr
import numpy as np
```

[24]:
```
california_set = pd.read_csv("C:/Users/USER/Desktop/California_House_Prices␣
↪BI_and_Python/housing.csv") # Reading the dataset
```

[25]:
```
print(california_set.head()) # Inspecting the first few rows
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
```

[26]:
```
print(len(california_set)) # Counting the rows in the dataset
```

```
20640
```

[27]:
```
california_set.describe()  # Observing the summary statistics / #␣
↪print(california_set.households.count()) prints 20640 or 20.64K
```

[27]:
```
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
```

```
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

[28]:
```
california_set.drop(['longitude','latitude'],axis =1 , inplace=True) # I dont␣
 ↪plan on using both coordinates
# axis = 1 to target the columns , inplace = True to make sure to reflect this␣
 ↪affect to the original dataset
# print(california_set.head())
```

[29]:
```
california_set.isna().any() # Making sure to check for null values
```

[29]:
```
housing_median_age    False
total_rooms           False
total_bedrooms         True
population            False
households            False
median_income         False
median_house_value    False
ocean_proximity       False
dtype: bool
```

[30]:
```
california_set.fillna({'total_bedrooms':0},inplace=True) #Replacing null values␣
 ↪with 0
california_set.isna().any()
```

[30]:
```
housing_median_age    False
total_rooms           False
total_bedrooms        False
population            False
households            False
median_income         False
median_house_value    False
ocean_proximity       False
dtype: bool
```

[31]:
```
print(california_set.ocean_proximity.value_counts()) # Checking the values
```

```
ocean_proximity
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: count, dtype: int64
```

[32]:
```
california_set.ocean_proximity.replace({'<1H OCEAN':0,'INLAND':1,'NEAR OCEAN':
↪2,'NEAR BAY':3,'ISLAND':4},inplace=True) # Converting them to numeric to␣
↪ensure that they work with the machine learning model
```

[33]:
```
print(california_set.head())
```

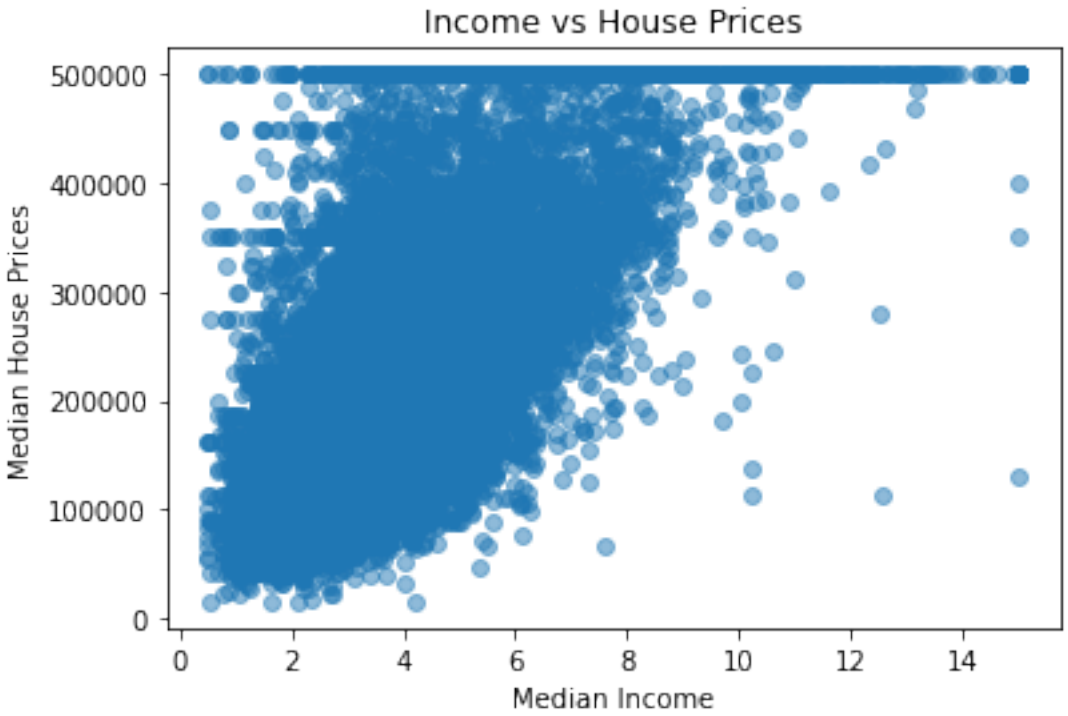
```
   housing_median_age  total_rooms  total_bedrooms  population  households  \
0                41.0        880.0           129.0       322.0       126.0
1                21.0       7099.0          1106.0      2401.0      1138.0
2                52.0       1467.0           190.0       496.0       177.0
3                52.0       1274.0           235.0       558.0       219.0
4                52.0       1627.0           280.0       565.0       259.0

   median_income  median_house_value  ocean_proximity
0         8.3252            452600.0                3
1         8.3014            358500.0                3
2         7.2574            352100.0                3
3         5.6431            341300.0                3
4         3.8462            342200.0                3
```

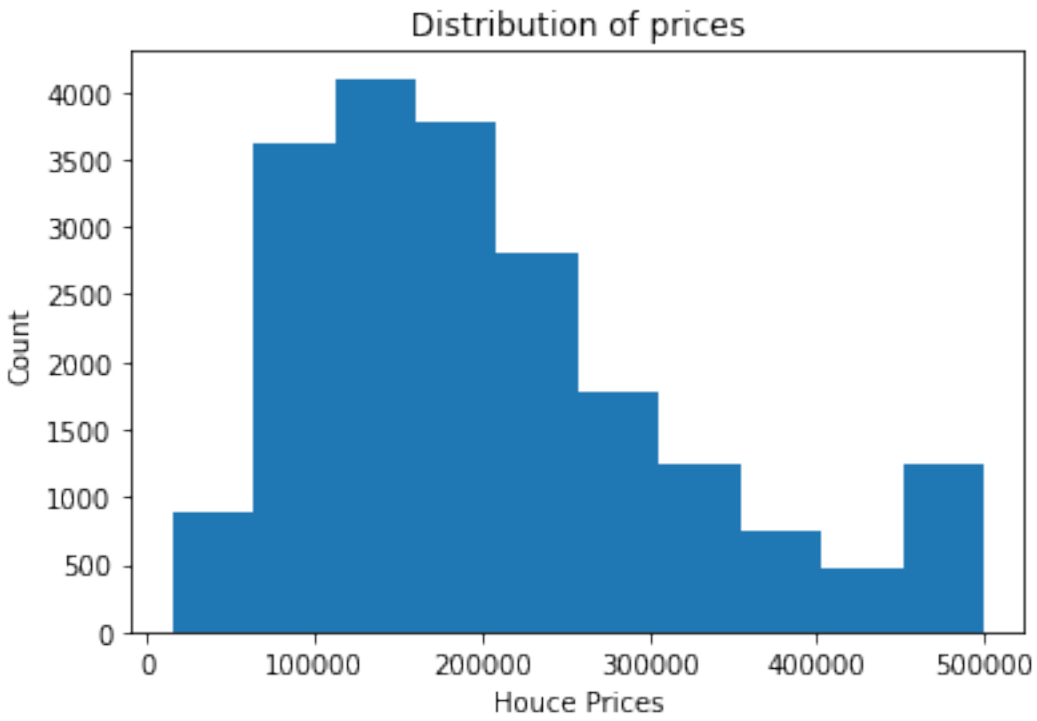
[34]:
```
# california_set.corr() results in a table containg all corr values between all␣
↪features (pandas)
x,p=pearsonr(california_set.median_income,california_set.median_house_value)
print(x) # X is the correlation coeff, pearsonr is a way to find if a linear␣
↪relationship is present between some features (numeric)
```

```
0.688075207958547
```

[35]:
```
plt.scatter(california_set.median_income,california_set.
 ↪median_house_value,alpha = 0.5)
plt.xlabel('Median Income')
plt.ylabel('Median House Prices')
plt.title('Income vs House Prices')
plt.show()   # Aiming to complement Power BI's dashboard
```



[36]:
```
plt.hist(california_set.median_house_value)
plt.xlabel('Houce Prices')
plt.ylabel('Count')
plt.title('Distribution of prices')
plt.show() #Distibution of house prices , the data here unlike Power BI␣
 ↪dashboard is not binned meaning
          # that each point is plotted directly in the histogram to show the␣
 ↪distribution of values
```



[37]:
```
# identify the features
features = california_set[['housing_median_age','total_rooms' ,␣
 ↪'total_bedrooms',  'population' , 'households' ,␣
 ↪'median_income','ocean_proximity' ]]
dependent = california_set['median_house_value']
# Spliting,making the model and then fitting it
x_train,x_test,y_train,y_test = train_test_split(features,dependent,test_size=0.
 ↪2,random_state=10)
model = LinearRegression()
model.fit(x_train,y_train)
```

[37]:
```
LinearRegression()
```

[38]:
```
# Evalute both the training data and testing data
model.score(x_train,y_train)
```

[38]:
```
0.5635071160526779
```

[39]:
```
model.score(x_test,y_test)
```

[39]:
```
0.5692395308106848
```

[40]:
```
# If you want to check the wieghts of each independent variable
# The higher the number the higher the affect relative to it's sign (+ or -)
coeff = model.coef_
print(list(zip(features,coeff)))
```

```
[('housing_median_age', 1872.079929339724), ('total_rooms',
-17.362378878211242), ('total_bedrooms', 54.76358282004938), ('population',
-34.716448483511385), ('households', 162.18155567224213), ('median_income',
47013.14147362578), ('ocean_proximity', 822.1722249972679)]
```

[41]:
```
predicted = model.predict(x_test)
print(predicted)
```

```
[228027.92429818 250212.93445469 265572.97297638 … 216541.40272373
  87856.45168959 400957.09797371]
```

[42]:
```
#Try some of your output for
# features = california_set[['housing_median_age','total_rooms' ,␣
 ↪'total_bedrooms',  'population' , 'households' ,␣
 ↪'median_income','ocean_proximity' ]]


data = np.array([66, 1, 1, 2000, 202, 2000, 4]).reshape(1, -1)  # Reshape to␣
 ↪(1, 7) --> 1 row and -1 adjusts to number of columns
columns = ['housing_median_age', 'total_rooms', 'total_bedrooms', 'population',␣
 ↪'households', 'median_income', 'ocean_proximity'] #naming them to avoid any␣
 ↪warnings
my_features = pd.DataFrame(data=data, columns=columns)
prediction = model.predict(my_features)
formatted_p = '{:,.2f}'.format(prediction[0])
print(formatted_p)
```

```
94,072,074.28
```