In [3]:
```
#calif_housing_data.csv

#Build a Python function that takes in a vector (array) and normalizes it.

#use normalize_vector
def normalize_vector(vector):
    min_val = min(vector)
    max_val = max(vector)
    range_val = max_val - min_val

    if range_val == 0:
#If max val is the same as min val, return 0
        return [0] * len(vector)

    normalized_vector = [(x - min_val) / range_val for x in vector]
    return normalized_vector

#example
vector = [1, 2, 3, 4, 5]
print(normalize_vector(vector))
```

```
[0.0, 0.25, 0.5, 0.75, 1.0]
```

In [5]:
```
#Build a Python function that takes in a vector (array) and standardizes it

#use standardize_vector
def standardize_vector(vector):
    mean_val = sum(vector) / len(vector)
    variance = sum([(x - mean_val)**2 for x in vector]) / len(vector)
    std_dev = variance**0.5

    if std_dev == 0:
#if SD is 0 return a  0
        return [0] * len(vector)
    standardized_vector = [(x - mean_val) / std_dev for x in vector]
    return standardized_vector

#example
vector = [1, 2, 3, 4, 5]
print(standardize_vector(vector))
```

```
[-1.414213562373095, -0.7071067811865475, 0.0, 0.7071067811865475, 1.414213562373095]
```

In [8]:
```
#calif_housing_data.csv
#(a)how many rows does this data set have?

#load pd
import pandas as pd

#csv calif_housing_data.csv
df = pd.read_csv("calif_housing_data.csv")

#number of rows
num_rows = df.shape[0]
print(f"number of rows is: {num_rows}")
```

```
number of rows is: 20640
```

In [10]:
```
#(b) What is the target vector for your model?

#load pd
import pandas as pd

#calif_housing_data.csv
df = pd.read_csv("calif_housing_data.csv")

#target vector
#median_house_value
target_vector = df["median_house_value"].values
print(target_vector)
```

```
[452600. 358500. 352100. ...  92300.  84700.  89400.]
```

In [11]:
```
#(c)Create a new feature by taking the total bedrooms divided by the number of househo

#load pd
import pandas as pd

#calif_housing_data.csv
df = pd.read_csv("calif_housing_data.csv")

#new feature
#total_bedrooms
#households
df['avg_bedrooms_per_household'] = df['total_bedrooms'] / df['households']
print(df[['avg_bedrooms_per_household']].head())
```

```
   avg_bedrooms_per_household
0                    1.023810
1                    0.971880
2                    1.073446
3                    1.073059
4                    1.081081
```

In [12]:
```
#(d) Now, create a new data frame that has three features: the median age, median inco

#load pd
import pandas as pd

#calif_housing_data.csv
df = pd.read_csv("calif_housing_data.csv")

#new feature
#total_bedrooms
#households
df['avg_bedrooms_per_household'] = df['total_bedrooms'] / df['households']

#new df
#housing_median_age
#median_income
new_df = df[['housing_median_age', 'median_income', 'avg_bedrooms_per_household']]

print(new_df.head())
```

```
   housing_median_age  median_income  avg_bedrooms_per_household
0                  41         8.3252                    1.023810
1                  21         8.3014                    0.971880
2                  52         7.2574                    1.073446
3                  52         5.6431                    1.073059
4                  52         3.8462                    1.081081
```

In [14]:
```python
#(e) Take the data frame created in part (d) and apply data standardization to the fea

#load pd,  reuse sklearn
import pandas as pd
from sklearn.preprocessing import StandardScaler


#calif_housing_data.csv
#housing_median_age
#median_income
#total_bedrooms
#households
df = pd.read_csv("calif_housing_data.csv")
df['avg_bedrooms_per_household'] = df['total_bedrooms'] / df['households']
new_df = df[['housing_median_age', 'median_income', 'avg_bedrooms_per_household']]

#standardize
scaler = StandardScaler()
standardized_data = scaler.fit_transform(new_df)

#put data back in df
standardized_df = pd.DataFrame(standardized_data, columns=new_df.columns)
print(standardized_df.head())
```

```
   housing_median_age  median_income  avg_bedrooms_per_household
0            0.982143       2.344766                   -0.153863
1           -0.607019       2.332238                   -0.262936
2            1.856182       1.782699                   -0.049604
3            1.856182       0.932968                   -0.050417
4            1.856182      -0.012881                   -0.033568
```

In [ ]:
```python

```