In [22]:
```
# import libraries
import pandas as pd
import numpy as np
import statistics

# import csv
df_main = pd.read_csv('calif_housing_data.csv')
df_main.head(5)
```

Out[22]:

	housing_median_age	total_bedrooms	households	median_income	median_house_v
0	41	129.0	126	8.3252	452600.0
1	21	1106.0	1138	8.3014	358500.0
2	52	190.0	177	7.2574	352100.0
3	52	235.0	219	5.6431	341300.0
4	52	280.0	259	3.8462	342200.0

In [18]:
```
# build a a function that takes in a vector and normalizes it
def vector(x):
    data = np.array(x)
    values = (data - data.min())/(data.max()-data.min())
    return values

vector(df_main['median_income'])
```

Out[18]:
```
array([ 0.53966842,  0.53802706,  0.46602805, ...,  0.08276438,
        0.09429525,  0.13025338])
```

In [17]:
```
# build a function that takes in a vector and standardizes it

def standard(x):
    data = np.array(x)
    values = (data - np.mean(data)) / (statistics.stdev(data))
    return values

standard(df_main['median_income'])
```

Out[17]:
```
array([ 2.34470896,  2.33218146,  1.78265622, ..., -1.14256563,
       -1.05455737, -0.78011057])
```

In [35]:
```
# 3a. How many rows does this data set have?
df_main
print("By using df_main, I found that this data set has 20640 rows × 5 column
s")

# another option
print("Another options is using len", "rows", ":", len(df_main))
```

```
By using df_main, I found that this data set has 20640 rows × 5 columns
Another options is using len rows : 20640
```

In [37]:
```
# 3b. What is the target vector?
# reference: https://h2o.ai/wiki/target-variable/
# The target variable is the variable whose values are modeled and predicted b
y other variables.
# A predictor variable is a variable whose values will be used to predict the
 value of the target variable.
print("For our data set, the median_house_value would be our vector as it's ca
lculated by the other figures")
```

```
For our data set, the median_house_value would be our vector as it's calculat
ed by the other figures
```

In [60]:
```
# 3c. Create a new feature by taking total bedrooms/number of households. What
does this represent?
df_main['average_bedrooms_per'] = df_main['total_bedrooms'] / df_main['househo
lds']
df_main.head(5)
# This creates a new column with a calculated average bedrooms found per neigh
borhood
```

Out[60]:

	housing_median_age	total_bedrooms	households	median_income	median_house_v
0	41	129.0	126	8.3252	452600.0
1	21	1106.0	1138	8.3014	358500.0
2	52	190.0	177	7.2574	352100.0
3	52	235.0	219	5.6431	341300.0
4	52	280.0	259	3.8462	342200.0

In [46]:
```
# 3d. Create a new data frame with 3 features: Media age, media income and new
feature from c
df_new = df_main[['housing_median_age', 'median_income', 'average_bedrooms_pe
r']]
df_new.head(5)
```

Out[46]:

	housing_median_age	median_income	average_bedrooms_per
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 [66]:
```
# 3e. Take data frame from part d and apply data standardization to the featur
es

# df_new['average_bedrooms_per'].astype(int) - attempted to change integer typ
e to change NaN
print(df_new.apply(standard, axis = 0))
```

	housing_median_age	median_income	average_bedrooms_per
0	0.983350	2.344709	NaN
1	-0.607765	2.332181	NaN
2	1.858463	1.782656	NaN
3	1.858463	0.932945	NaN
4	1.858463	-0.012881	NaN
5	1.858463	0.087445	NaN
6	1.858463	-0.111364	NaN
7	1.858463	-0.395127	NaN
8	1.062905	-0.942336	NaN
9	1.858463	-0.094467	NaN
10	1.858463	-0.351386	NaN
11	1.858463	-0.315909	NaN
12	1.858463	-0.418814	NaN
13	1.858463	-0.630096	NaN
14	1.858463	-1.028502	NaN
15	1.699351	-0.918860	NaN
16	1.858463	-0.576723	NaN
17	1.858463	-0.921387	NaN
18	1.699351	-0.989341	NaN
19	1.858463	-0.667100	NaN
20	0.903794	-1.322688	NaN
21	1.062905	-1.135460	NaN
22	1.858463	-1.129407	NaN
23	1.858463	-0.889595	NaN
24	1.858463	-0.668837	NaN
25	0.983350	-0.772110	NaN
26	1.619795	-0.742686	NaN
27	1.858463	-1.085718	NaN
28	1.699351	-1.172884	NaN
29	1.858463	-1.149145	NaN
...	...	...	...
20610	-0.050875	-1.319898	NaN
20611	-0.130431	-1.360639	NaN
20612	-0.209986	-1.251313	NaN
20613	0.187792	-1.250049	NaN
20614	-0.209986	-0.737528	NaN
20615	-0.448653	-0.795270	NaN
20616	-1.085099	-0.959970	NaN
20617	-0.687321	-0.299276	NaN
20618	-0.289542	-0.853065	NaN
20619	-0.528209	-0.600252	NaN
20620	0.903794	0.364155	NaN
20621	0.665127	-0.791954	NaN
20622	-0.687321	-0.765320	NaN
20623	0.267348	-0.551194	NaN
20624	-1.005544	-0.419393	NaN
20625	0.665127	0.133870	NaN
20626	0.585571	-0.896911	NaN
20627	-1.880657	-0.458291	NaN
20628	-0.766876	-0.671364	NaN
20629	-0.050875	-0.935020	NaN
20630	-1.403322	-0.159684	NaN
20631	-1.085099	-0.185686	NaN
20632	-1.085099	-0.392495	NaN
20633	-0.130431	-0.695418	NaN
20634	-0.050875	-0.083256	NaN

```
20635          -0.289542      -1.216099                  NaN
20636          -0.846432      -0.691576                  NaN
20637          -0.925988      -1.142566                  NaN
20638          -0.846432      -1.054557                  NaN
20639          -1.005544      -0.780111                  NaN

[20640 rows x 3 columns]
```