In [1]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
from sklearn.datasets import fetch_california_housing
california_housing = fetch_california_housing()

california_housing
```

Out[3]:
```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.55555556,
           37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
           37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
           37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
           39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
           39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
           39.37      , -121.24      ]]),
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
```

In [6]:
```
# creating a DataFrame

df = pd.DataFrame(california_housing.data, columns=california_housing.feature_names)
df['MEDV'] = california_housing.target
df
```

Out[6]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MEDV
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...	...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   MEDV        20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

In [8]:
```
df.describe()
```

Out[8]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MEDV
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.068558
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.153956
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.149990
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.196000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.797000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.647250
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.000010

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
MEDV          0
dtype: int64
```

In [11]:
```
# Seperating the features and target variable
x = df.iloc[:, df.columns != "MEDV"]
y = df.iloc[:, df.columns == "MEDV"]
```

In [12]:
```
x
```

Out[12]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

In [13]:
```
y
```

Out[13]:

	MEDV
0	4.526
1	3.585
2	3.521
3	3.413
4	3.422
...	...
20635	0.781
20636	0.771
20637	0.923
20638	0.847
20639	0.894

In [14]:
```
x.shape
```

Out[14]:
```
(20640, 8)
```

In [15]:
```
y.shape
```

Out[15]:
```
(20640, 1)
```

In [16]:
```
# splitting the data into training(70%) and testing(30%) sets

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=123)
```

In [17]:
```
x_train
```

Out[17]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
12364	3.9508	6.0	5.873188	1.111111	3010.0	3.635266	33.82	-116.46
12271	3.5255	21.0	5.694581	1.049261	2174.0	2.677340	34.00	-117.04
19605	1.9728	32.0	5.468208	1.144509	624.0	3.606936	37.55	-121.03
10600	6.9133	8.0	5.976471	1.026471	862.0	2.535294	33.68	-117.80
45	2.6768	52.0	4.335079	1.099476	718.0	1.879581	37.83	-122.26
...	...	...	...	...	...	...	...	...
7763	3.6389	36.0	5.584615	1.115385	490.0	3.769231	33.91	-118.10
15377	4.5391	14.0	6.016688	1.017972	2436.0	3.127086	33.37	-117.24
17730	5.6306	5.0	5.958393	1.031564	2435.0	3.493544	37.33	-121.76
15725	3.8750	44.0	4.739264	1.024540	561.0	1.720859	37.78	-122.44
19966	2.5156	20.0	5.491379	1.117816	1241.0	3.566092	36.21	-119.08

In [18]:
```
x_test
```

Out[18]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
19121	3.7917	40.0	4.959799	1.030151	1039.0	2.610553	38.24	-122.64
20019	4.0217	9.0	5.804577	1.000000	1749.0	3.079225	36.09	-119.05
15104	4.0882	12.0	5.360360	1.070571	3321.0	4.986486	32.85	-116.98
3720	2.2377	27.0	3.376582	1.023207	3403.0	3.589662	34.20	-118.42
8938	4.4211	41.0	5.656904	1.165272	1047.0	2.190377	34.01	-118.47
...	...	...	...	...	...	...	...	...
5462	2.5938	41.0	4.244444	1.148148	833.0	3.085185	33.99	-118.47
1859	2.2750	21.0	5.457490	1.170040	1208.0	2.445344	41.95	-124.14
10867	3.4835	17.0	3.825243	1.153099	2966.0	2.215086	33.70	-117.88
4693	4.2500	52.0	5.736979	1.138021	899.0	2.341146	34.07	-118.37
3521	4.5800	33.0	6.009862	1.039448	1578.0	3.112426	34.27	-118.49

In [19]:
```
y_train
```

Out[19]:

	MEDV
12364	1.042
12271	1.321
19605	0.979
10600	2.741
45	1.823
...	...
7763	1.676
15377	1.809
17730	2.862
15725	4.125
19966	0.593

In [20]:
```
y_test
```

Out[20]:

	MEDV
19121	1.516
20019	0.992
15104	1.345
3720	2.317
8938	4.629
...	...
5462	2.850
1859	1.224
10867	1.167
4693	4.869
3521	2.362

In [22]:
```
# creating the model for neural network architecture
# Sequential: it is linear stack of layers, created by adding layers one by one in sequence
# Dense: fully connected layer, adds dense layer to the model and takes 3 parameters(units, activation, name)

from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model
```

Out[22]:
```
<Sequential name=sequential, built=False>
```

In [24]:
```
# this adds dense layers to the model

model.add(Dense(128, input_shape=(8, ),  activation="relu", name="dense_1"))
model.add(Dense(64, activation="relu", name="dense_2"))
model.add(Dense(1, activation="relu", name="dense_output"))
```

```
D:\py\Lib\site-packages\keras\src\layers\core\dense.py:88: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [27]:
```
# compiling the model (setting the configuration for training it later on)
# takes 3 parameters(optimizer, loss, metrics)

model.compile(optimizer="adam", loss="mse", metrics=["mae"])
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 128)	1,152
dense_2 (Dense)	(None, 64)	8,256
dense_output (Dense)	(None, 1)	65

Total params: 9,473 (37.00 KB)

Trainable params: 9,473 (37.00 KB)

Non-trainable params: 0 (0.00 B)

In [29]:
```
# training the neural network

history = model.fit(x_train, y_train, epochs=100, validation_split=0.05, verbose = 1)
```

```
Epoch 1/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 514.7137 - mae: 4.6907 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 2/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.7319 - mae: 2.0945 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 3/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.8235 - mae: 2.1087 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 4/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.5772 - mae: 2.0657 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 5/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.7071 - mae: 2.0887 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 6/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.6863 - mae: 2.0820 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 7/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.7631 - mae: 2.0953 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 8/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.6668 - mae: 2.0806 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 9/100
429/429 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.6320 - mae: 2.0720 - val_loss: 5.4586 - val_mae: 2.0369
Epoch 10/100
```

In [30]:
```
mse_nn, mae_nn = model.evaluate(x_test, y_test)
print("mse_nn: ", mse_nn)
print("mae_nn: ", mae_nn)
```

```
194/194 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 5.6806 - mae: 2.0799
mse_nn:  5.541019916534424
mae_nn:  2.0531160831451416
```

In [ ]:
```

```