In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [2]:
```
df =  pd.read_csv("house_data.csv")
df
```

Out[2]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0
...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0

4600 rows × 18 columns

In [4]:
```
df.info ()
```

```
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

In [8]:
```
X = df.drop(['price', 'date', 'street', 'city', 'statezip', 'country'], axi
y = df['price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train a Linear Regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Make predictions on the test set
linear_predictions = linear_model.predict(X_test)

# Evaluate the Linear Regression model
linear_rmse = np.sqrt(mean_squared_error(y_test, linear_predictions))
print(f'Linear Regression RMSE: {linear_rmse}')

# Train a Neural Network using TensorFlow
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_te

# Make predictions using the Neural Network
nn_predictions = model.predict(X_test)

# Evaluate the Neural Network model
nn_rmse = np.sqrt(mean_squared_error(y_test, nn_predictions))
print(f'Neural Network RMSE: {nn_rmse}')
```

```
Linear Regression RMSE: 993439.3625461654
Epoch 1/50
115/115 - 1s - loss: 438935584768.0000 - val_loss: 1356701761536.0000
Epoch 2/50
115/115 - 0s - loss: 438816243712.0000 - val_loss: 1356450627584.0000
Epoch 3/50
115/115 - 0s - loss: 438335406080.0000 - val_loss: 1355694211072.0000
Epoch 4/50
115/115 - 0s - loss: 437213495296.0000 - val_loss: 1354182033408.0000
Epoch 5/50
115/115 - 0s - loss: 435197018112.0000 - val_loss: 1351658635264.0000
Epoch 6/50
115/115 - 0s - loss: 432082878464.0000 - val_loss: 1347983114240.0000
Epoch 7/50
115/115 - 0s - loss: 427634458624.0000 - val_loss: 1343012208640.0000
Epoch 8/50
115/115 - 0s - loss: 421827379200.0000 - val_loss: 1336543281152.0000
Epoch 9/50
115/115 - 0s - loss: 414548819968.0000 - val_loss: 1328659431424.0000
Epoch 10/50
115/115 - 0s - loss: 405706342400.0000 - val_loss: 1319313342464.0000
Epoch 11/50
115/115 - 0s - loss: 395443765248.0000 - val_loss: 1308655222784.0000
Epoch 12/50
115/115 - 0s - loss: 383782125568.0000 - val_loss: 1296664363008.0000
Epoch 13/50
115/115 - 0s - loss: 370869600256.0000 - val_loss: 1283431596032.0000
Epoch 14/50
115/115 - 0s - loss: 356735778816.0000 - val_loss: 1269284077568.0000
Epoch 15/50
115/115 - 0s - loss: 341623898112.0000 - val_loss: 1254310805504.0000
Epoch 16/50
115/115 - 0s - loss: 325600673792.0000 - val_loss: 1238395912192.0000
Epoch 17/50
115/115 - 0s - loss: 308676952064.0000 - val_loss: 1222110740480.0000
Epoch 18/50
115/115 - 0s - loss: 291396911104.0000 - val_loss: 1205408235520.0000
Epoch 19/50
115/115 - 0s - loss: 274008784896.0000 - val_loss: 1189178245120.0000
Epoch 20/50
115/115 - 0s - loss: 256827539456.0000 - val_loss: 1173553676288.0000
Epoch 21/50
115/115 - 0s - loss: 240081797120.0000 - val_loss: 1158451036160.0000
Epoch 22/50
115/115 - 0s - loss: 223933562880.0000 - val_loss: 1144127225856.0000
Epoch 23/50
115/115 - 0s - loss: 208676945920.0000 - val_loss: 1130903502848.0000
Epoch 24/50
115/115 - 0s - loss: 194532147200.0000 - val_loss: 1118959435776.0000
Epoch 25/50
115/115 - 0s - loss: 181585494016.0000 - val_loss: 1108383760384.0000
Epoch 26/50
115/115 - 0s - loss: 169882419200.0000 - val_loss: 1098979344384.0000
Epoch 27/50
115/115 - 0s - loss: 159479005184.0000 - val_loss: 1090952495104.0000
Epoch 28/50
115/115 - 0s - loss: 150388719616.0000 - val_loss: 1083901018112.0000
Epoch 29/50
115/115 - 0s - loss: 142560952320.0000 - val_loss: 1078079258624.0000
Epoch 30/50
115/115 - 0s - loss: 135822671872.0000 - val_loss: 1073138892800.0000
```

```
Epoch 31/50
115/115 - 0s - loss: 130082963456.0000 - val_loss: 1069012090880.0000
Epoch 32/50
115/115 - 0s - loss: 125237305344.0000 - val_loss: 1065459777536.0000
Epoch 33/50
115/115 - 0s - loss: 121076498432.0000 - val_loss: 1062382534656.0000
Epoch 34/50
115/115 - 0s - loss: 117512282112.0000 - val_loss: 1059689332736.0000
Epoch 35/50
115/115 - 0s - loss: 114447286272.0000 - val_loss: 1057241956352.0000
Epoch 36/50
115/115 - 0s - loss: 111808397312.0000 - val_loss: 1055026642944.0000
Epoch 37/50
115/115 - 0s - loss: 109450428416.0000 - val_loss: 1052851830784.0000
Epoch 38/50
115/115 - 0s - loss: 107346690048.0000 - val_loss: 1050900496384.0000
Epoch 39/50
115/115 - 0s - loss: 105459138560.0000 - val_loss: 1048953683968.0000
Epoch 40/50
115/115 - 0s - loss: 103773667328.0000 - val_loss: 1047139713024.0000
Epoch 41/50
115/115 - 0s - loss: 102233202688.0000 - val_loss: 1045415657472.0000
Epoch 42/50
115/115 - 0s - loss: 100813479936.0000 - val_loss: 1043756613632.0000
Epoch 43/50
115/115 - 0s - loss: 99543687168.0000 - val_loss: 1042149736448.0000
Epoch 44/50
115/115 - 0s - loss: 98346205184.0000 - val_loss: 1040528048128.0000
Epoch 45/50
115/115 - 0s - loss: 97263009792.0000 - val_loss: 1039129706496.0000
Epoch 46/50
115/115 - 0s - loss: 96291627008.0000 - val_loss: 1037609074688.0000
Epoch 47/50
115/115 - 0s - loss: 95311126528.0000 - val_loss: 1036284395520.0000
Epoch 48/50
115/115 - 0s - loss: 94463664128.0000 - val_loss: 1035062476800.0000
Epoch 49/50
115/115 - 0s - loss: 93644185600.0000 - val_loss: 1033764864000.0000
Epoch 50/50
115/115 - 0s - loss: 92893020160.0000 - val_loss: 1032730509312.0000
Neural Network RMSE: 1016233.4772878524
```

In [ ]: