```
from google.colab import files


uploaded = files.upload()
```

Choose Files House Price India.csv.zip

- **House Price India.csv.zip**(application/x-zip-compressed) - 491826 bytes, last modified: 4/9/2025 - 100% done

```
import zipfile
import os


with zipfile.ZipFile("House Price India.csv.zip", 'r') as zip_ref:
    zip_ref.extractall("house_price_data")


os.listdir("house_price_data")
```

```
['House Price India.csv']
```

```
import pandas as pd


df = pd.read_csv("house_price_data/House Price India.csv")
df.head()
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Postal Code	Latt
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003	5
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004	5
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122004	5
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122005	5
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122006	5

5 rows × 23 columns

```
if 'Date' in df.columns:
    df.drop('Date', axis=1, inplace=True)


if df['number of bedrooms'].dtype != 'int64':
    df['number of bedrooms'] = df['number of bedrooms'].astype(int)


df.drop(['id', 'Built Year'], axis=1, inplace=True)


df.fillna(df.mean(numeric_only=True), inplace=True)
for col in df.select_dtypes(include='object'):
    df[col].fillna(df[col].mode()[0], inplace=True)


df = pd.get_dummies(df)


X = df.drop('Price', axis=1)
y = df['Price']


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model = Sequential()


model.add(Dense(units=64, activation='relu', input_dim=X_scaled.shape[1]))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=64, activation='relu'))


model.add(Dense(units=1))


model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])


model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	1,280
dense_1 (Dense)	(None, 128)	8,320
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 1)	65

Total params: 17,921 (70.00 KB)
Trainable params: 17,921 (70.00 KB)
Non-trainable params: 0 (0.00 B)

```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)


history = model.fit(X_train, y_train, validation_split=0.1, epochs=100, batch_size=32)
```

```
Epoch 1/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 406939402240.0000 - mae: 532535.9375 - val_loss: 232681308160.0000 - val_mae: 4
Epoch 2/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 166923059200.0000 - mae: 313560.0312 - val_loss: 61714636800.0000 - val_mae: 18
Epoch 3/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 61294063616.0000 - mae: 174209.9531 - val_loss: 49173635072.0000 - val_mae: 156
Epoch 4/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 51325181952.0000 - mae: 158135.2344 - val_loss: 43362742272.0000 - val_mae: 143
Epoch 5/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 48555302912.0000 - mae: 143890.6406 - val_loss: 38937337856.0000 - val_mae: 132
Epoch 6/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 43434160128.0000 - mae: 132502.9688 - val_loss: 36332593152.0000 - val_mae: 125
Epoch 7/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 37006888960.0000 - mae: 127169.1094 - val_loss: 35319570432.0000 - val_mae: 123
Epoch 8/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 38619197440.0000 - mae: 123561.1484 - val_loss: 33614995456.0000 - val_mae: 118
Epoch 9/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 34979434496.0000 - mae: 119027.9219 - val_loss: 33707943936.0000 - val_mae: 119
Epoch 10/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 34632335360.0000 - mae: 118979.8281 - val_loss: 32669972480.0000 - val_mae: 116
Epoch 11/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 34579304448.0000 - mae: 115416.7578 - val_loss: 32065712128.0000 - val_mae: 114
Epoch 12/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 35745775616.0000 - mae: 116213.1562 - val_loss: 31699087360.0000 - val_mae: 113
Epoch 13/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 36895526912.0000 - mae: 115409.3281 - val_loss: 31406450688.0000 - val_mae: 112
Epoch 14/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 36704141312.0000 - mae: 116641.5859 - val_loss: 31225341952.0000 - val_mae: 112
Epoch 15/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 31889031168.0000 - mae: 111501.9375 - val_loss: 31114160128.0000 - val_mae: 112
Epoch 16/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 34147141632.0000 - mae: 113163.9141 - val_loss: 30950510592.0000 - val_mae: 116
Epoch 17/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 30788556800.0000 - mae: 110803.2812 - val_loss: 30725249024.0000 - val_mae: 116
Epoch 18/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 34273038336.0000 - mae: 112621.8125 - val_loss: 30581997568.0000 - val_mae: 116
Epoch 19/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 39655055360.0000 - mae: 115398.0469 - val_loss: 30467100672.0000 - val_mae: 116
Epoch 20/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 32945739776.0000 - mae: 112550.5859 - val_loss: 30388410368.0000 - val_mae: 109
Epoch 21/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 34001319936.0000 - mae: 112537.0234 - val_loss: 30344681472.0000 - val_mae: 109
Epoch 22/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 33169256448.0000 - mae: 112637.3594 - val_loss: 30259888128.0000 - val_mae: 116
Epoch 23/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 31113089024.0000 - mae: 109827.3906 - val_loss: 30041286656.0000 - val_mae: 109
Epoch 24/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 31227009024.0000 - mae: 110030.5938 - val_loss: 29976514560.0000 - val_mae: 109
```

```
Epoch 25/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 36614791168.0000 - mae: 111858.6328 - val_loss: 29856557056.0000 - val_mae: 109
Epoch 26/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 29628772352.0000 - mae: 108535.0859 - val_loss: 29861945344.0000 - val_mae: 108
Epoch 27/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 29941528576.0000 - mae: 106826.5000 - val_loss: 30135922688.0000 - val_mae: 110
Epoch 28/100
329/329 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 30699579392.0000 - mae: 108647.1250 - val_loss: 29738448896.0000 - val_mae: 108
```

```
loss, mae = model.evaluate(X_test, y_test)
print("Test MAE:", mae)
```

```
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 30277146624.0000 - mae: 108092.3984
Test MAE: 105969.4765625
```

```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```