In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split,KFold,cross_val_score
from sklearn.preprocessing import StandardScaler,MinMaxScaler


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
housing_data = pd.read_csv('BostonHousing.csv')
```

In [3]:
```
housing_data.head()
```

Out[3]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [4]:
```
housing_data.describe()
```

Out[4]:

	crim	zn	indus	chas	nox	rm	age	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.79
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.10
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.10
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.20
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.18
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

In [5]:
```
housing_data.isnull().sum()
```

Out[5]:
```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

In [6]:
```
housing_data.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
X = housing_data.drop(columns = ['medv'])
y = housing_data.medv

sc = StandardScaler()
X = sc.fit_transform(X)



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
```

In [8]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[8]:
```
((354, 13), (152, 13), (354,), (152,))
```

In [9]:
```
model = Sequential()

model.add(Dense(128, input_shape=(13, ), activation='relu', name='dense_1'))
model.add(Dense(64, activation='relu', name='dense_2'))
model.add(Dense(1, activation='linear', name='dense_output'))


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

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense_1 (Dense)             (None, 128)               1792

dense_2 (Dense)             (None, 64)                8256

dense_output (Dense)        (None, 1)                 65

=================================================================
Total params: 10,113
Trainable params: 10,113
Non-trainable params: 0
_________________________________________________________________
```

In [10]:
```
history = model.fit(X_train, y_train, epochs=200, validation_split=0.2)
```

```
Epoch 1/200
9/9 [==============================] - 2s 39ms/step - loss: 580.0472 - mae: 22.265
6 - val_loss: 515.2385 - val_mae: 20.8526
Epoch 2/200
9/9 [==============================] - 0s 9ms/step - loss: 520.5151 - mae: 20.9143
- val_loss: 448.8072 - val_mae: 19.2797
Epoch 3/200
9/9 [==============================] - 0s 10ms/step - loss: 447.1745 - mae: 19.168
7 - val_loss: 364.3033 - val_mae: 17.1457
Epoch 4/200
9/9 [==============================] - 0s 11ms/step - loss: 353.0789 - mae: 16.718
5 - val_loss: 263.5933 - val_mae: 14.2895
Epoch 5/200
9/9 [==============================] - 0s 9ms/step - loss: 245.7141 - mae: 13.4567
- val_loss: 162.5475 - val_mae: 10.7573
Epoch 6/200
9/9 [==============================] - 0s 9ms/step - loss: 150.9113 - mae: 9.9895
- val_loss: 88.9941 - val_mae: 7.3551
Epoch 7/200
9/9 [==============================] - 0s 9ms/step - loss: 85.1846 - mae: 7.4166 -
val_loss: 62.7974 - val_mae: 5.8603
Epoch 8/200
9/9 [==============================] - 0s 9ms/step - loss: 59.6940 - mae: 5.9908 -
val_loss: 56.2954 - val_mae: 5.5551
Epoch 9/200
9/9 [==============================] - 0s 9ms/step - loss: 47.3396 - mae: 5.2230 -
val_loss: 46.0090 - val_mae: 4.9398
Epoch 10/200
9/9 [==============================] - 0s 9ms/step - loss: 36.5004 - mae: 4.5176 -
val_loss: 35.0649 - val_mae: 4.1577
Epoch 11/200
9/9 [==============================] - 0s 9ms/step - loss: 28.9871 - mae: 4.0162 -
val_loss: 30.3657 - val_mae: 3.8428
Epoch 12/200
9/9 [==============================] - 0s 10ms/step - loss: 25.0969 - mae: 3.6780
- val_loss: 27.6609 - val_mae: 3.7041
Epoch 13/200
9/9 [==============================] - 0s 10ms/step - loss: 22.5636 - mae: 3.4434
- val_loss: 25.8646 - val_mae: 3.6280
Epoch 14/200
9/9 [==============================] - 0s 10ms/step - loss: 21.0843 - mae: 3.3112
- val_loss: 24.1053 - val_mae: 3.5487
Epoch 15/200
9/9 [==============================] - 0s 10ms/step - loss: 19.5805 - mae: 3.1940
- val_loss: 22.8075 - val_mae: 3.4528
Epoch 16/200
9/9 [==============================] - 0s 10ms/step - loss: 18.3879 - mae: 3.0638
- val_loss: 21.6682 - val_mae: 3.3447
Epoch 17/200
9/9 [==============================] - 0s 9ms/step - loss: 17.5130 - mae: 2.9467 -
val_loss: 21.1239 - val_mae: 3.2885
Epoch 18/200
9/9 [==============================] - 0s 10ms/step - loss: 16.7063 - mae: 2.8636
- val_loss: 20.5709 - val_mae: 3.2310
Epoch 19/200
9/9 [==============================] - 0s 9ms/step - loss: 16.0412 - mae: 2.7897 -
val_loss: 19.9443 - val_mae: 3.1714
Epoch 20/200
9/9 [==============================] - 0s 11ms/step - loss: 15.4640 - mae: 2.7447
- val_loss: 19.0997 - val_mae: 3.1049
Epoch 21/200
9/9 [==============================] - 0s 9ms/step - loss: 14.9453 - mae: 2.6872 -
val_loss: 18.6990 - val_mae: 3.0743
Epoch 22/200
```

```
Epoch 193/200
9/9 [==============================] - 0s 11ms/step - loss: 2.8564 - mae: 1.2454 -
val_loss: 7.4615 - val_mae: 2.0969
Epoch 194/200
9/9 [==============================] - 0s 10ms/step - loss: 2.8677 - mae: 1.2607 -
val_loss: 7.2914 - val_mae: 2.0922
Epoch 195/200
9/9 [==============================] - 0s 10ms/step - loss: 2.8928 - mae: 1.2356 -
val_loss: 7.2349 - val_mae: 2.0819
Epoch 196/200
9/9 [==============================] - 0s 10ms/step - loss: 2.8208 - mae: 1.2274 -
val_loss: 7.3682 - val_mae: 2.0994
Epoch 197/200
9/9 [==============================] - 0s 10ms/step - loss: 2.7770 - mae: 1.2369 -
val_loss: 7.3326 - val_mae: 2.0833
Epoch 198/200
9/9 [==============================] - 0s 10ms/step - loss: 2.7763 - mae: 1.2236 -
val_loss: 7.2133 - val_mae: 2.0738
Epoch 199/200
9/9 [==============================] - 0s 10ms/step - loss: 2.7261 - mae: 1.2082 -
val_loss: 7.2567 - val_mae: 2.0857
Epoch 200/200
9/9 [==============================] - 0s 10ms/step - loss: 2.7398 - mae: 1.2067 -
val_loss: 7.2556 - val_mae: 2.0849
```

In [17]:

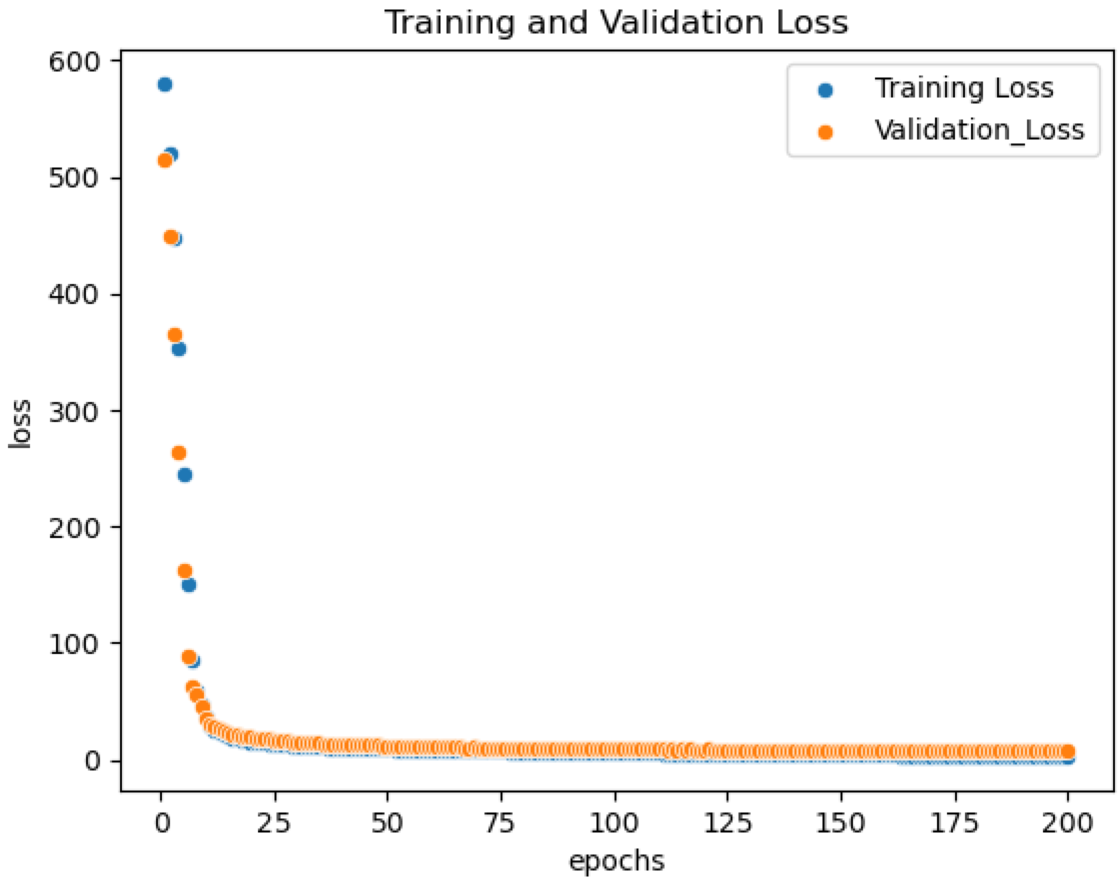
```
print(len(history.history['mae']))
```

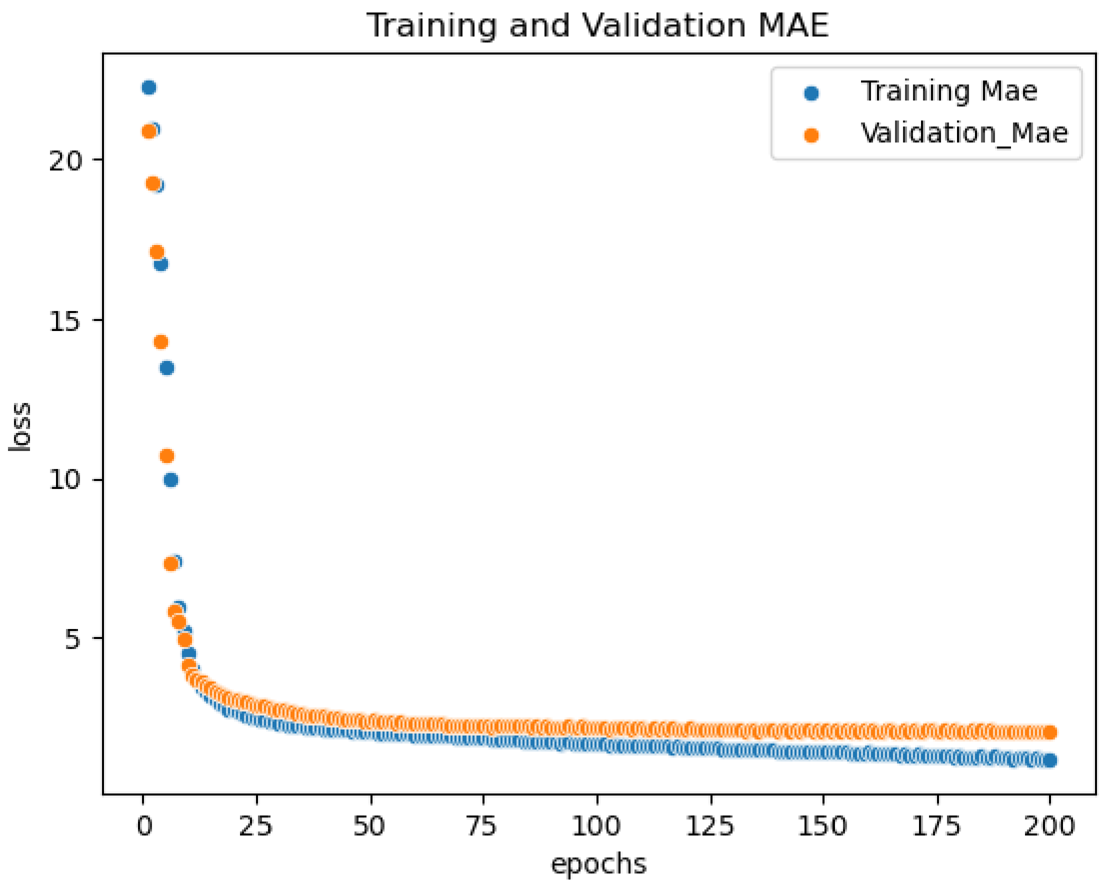
```
200
```

In [18]:

```
sns.scatterplot(y = history.history['loss'],x = range(1,200+1))
sns.scatterplot(y = history.history['val_loss'],x = range(1,200+1))
plt.title('Training and Validation Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend(['Training Loss','Validation_Loss'])
plt.show()


sns.scatterplot(y = history.history['mae'],x = range(1,200+1))
sns.scatterplot(y = history.history['val_mae'],x = range(1,200+1))
plt.title('Training and Validation MAE')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend(['Training Mae','Validation_Mae'])
plt.show()
```





In [19]:
```
mse_nn, mae_nn = model.evaluate(X_test, y_test)
print('Mean squared error on test data: ', mse_nn)
print('Mean absolute error on test data: ', mae_nn)
```

```
5/5 [==============================] - 0s 7ms/step - loss: 16.1675 - mae: 2.7948
Mean squared error on test data:  16.16754150390625
Mean absolute error on test data:  2.7947776317596436
```