| Name: | Tufan Kundu |
|---|---|
| Registration no: | 24MDT0184 |
| Course Name: | Deep Learning Lab |
| Course Code: | PMDS603P |
| Experiment: | 6 |
| Date: | 28 August,2025 |

# Load the Boston Housing dataset available in Keras. The dataset contains 13 numerical features about houses (crime rate, average rooms, property tax, etc.) and the target is the median house price in $1000s.

In [56]:
```
from tensorflow import keras
from keras.datasets import boston_housing

(x_train,y_train),(x_test,y_test) = boston_housing.load_data()
```

In [63]:
```
## Scaling the data

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [69]:
```
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
from keras.optimizers import SGD, Adam
from keras.callbacks import EarlyStopping

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

In [70]:
```
## Building the model

model = Sequential()
model.add(Flatten(input_shape = (13,)))
model.add(Dense(256, activation = 'sigmoid'))
model.add(Dense(128, activation = 'sigmoid'))
model.add(Dense(1, activation = 'linear'))

model.summary()
```

**Model: "sequential_11"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| flatten_8 (Flatten) | (None, 13) | 0 |
| dense_75 (Dense) | (None, 256) | 3,584 |
| dense_76 (Dense) | (None, 128) | 32,896 |
| dense_77 (Dense) | (None, 1) | 129 |

**Total params:** 36,609 (143.00 KB)

**Trainable params:** 36,609 (143.00 KB)

**Non-trainable params:** 0 (0.00 B)

In [71]:
```
sgd = SGD(learning_rate=0.01)
estop = EarlyStopping(monitor = 'val_loss', min_delta = 1e-5, mode = 'min', patience = 5, verbose = 1, restore_best_weight
model.compile(loss = 'mean_squared_error', optimizer=sgd, metrics= [keras.metrics.R2Score()])
history = model.fit(x_train_scaled,y_train,batch_size=64, epochs = 1000, verbose = 1, validation_split=0.2, callbacks=estc
```

```
Epoch 1/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 293.0572 - r2_score: -2.2773 - val_loss: 106.6560 - val_r2_score: -0.2619
Epoch 2/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 78.6669 - r2_score: 0.0489 - val_loss: 118.1936 - val_r2_score: -0.3984
Epoch 3/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 73.6845 - r2_score: 0.0723 - val_loss: 94.0876 - val_r2_score: -0.1132
Epoch 4/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 71.8907 - r2_score: 0.1548 - val_loss: 189.9255 - val_r2_score: -1.2471
Epoch 5/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 104.6285 - r2_score: -0.2405 - val_loss: 60.6612 - val_r2_score: 0.2823
Epoch 6/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 56.8989 - r2_score: 0.3738 - val_loss: 73.8399 - val_r2_score: 0.1264
Epoch 7/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 49.8980 - r2_score: 0.4099 - val_loss: 62.6127 - val_r2_score: 0.2592
Epoch 8/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 53.8051 - r2_score: 0.2927 - val_loss: 45.1824 - val_r2_score: 0.4654
Epoch 9/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 37.9489 - r2_score: 0.5606 - val_loss: 57.2495 - val_r2_score: 0.3227
Epoch 10/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 47.9588 - r2_score: 0.4027 - val_loss: 145.3610 - val_r2_score: -0.7198
Epoch 11/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 117.6920 - r2_score: -0.4532 - val_loss: 64.4937 - val_r2_score: 0.2370
Epoch 12/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 53.4691 - r2_score: 0.3238 - val_loss: 44.5699 - val_r2_score: 0.4727
Epoch 13/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 35.4528 - r2_score: 0.5737 - val_loss: 29.6334 - val_r2_score: 0.6494
Epoch 14/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 25.6315 - r2_score: 0.6843 - val_loss: 26.7286 - val_r2_score: 0.6838
Epoch 15/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 21.8708 - r2_score: 0.7169 - val_loss: 254.5145 - val_r2_score: -2.0112
Epoch 16/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 146.0817 - r2_score: -0.6836 - val_loss: 30.3575 - val_r2_score: 0.6408
Epoch 17/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 31.9631 - r2_score: 0.6488 - val_loss: 22.2229 - val_r2_score: 0.7371
Epoch 18/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 27.8840 - r2_score: 0.6580 - val_loss: 33.8688 - val_r2_score: 0.5993
Epoch 19/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 24.9436 - r2_score: 0.7050 - val_loss: 24.7192 - val_r2_score: 0.7075
Epoch 20/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 21.3761 - r2_score: 0.7371 - val_loss: 83.9552 - val_r2_score: 0.0067
Epoch 21/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 46.0109 - r2_score: 0.3999 - val_loss: 34.5628 - val_r2_score: 0.5911
Epoch 22/1000
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 23.7150 - r2_score: 0.7174 - val_loss: 26.8818 - val_r2_score: 0.6820
Epoch 22: early stopping
Restoring model weights from the end of the best epoch: 17.
```

In [72]:
```
score = model.evaluate(x_test_scaled,y_test,verbose = 1)
print("Test loss:", score[0])
print("R2 score", score[1])
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 18.9049 - r2_score: 0.7517
Test loss: 21.185190200805664
R2 score 0.7455042600631714
```

## Question 2: Hyperparameter Tuning: Perform hyperparameter tuning for the following hyperparameters:

- Number of neurons in each hidden layer.
- activation function in each hidden layer.
- Learning rate of the optimizer.
- Momentum parameter in SGD.

In [77]:
```
import keras_tuner as kt
def build_model(hp):
    model = Sequential()
    activation = hp.Choice('activation', values = ['relu','sigmoid','tanh'])
    model.add(Flatten(input_shape= (13,)))
    units = hp.Int('units', min_value = 64, max_value = 512, step = 32)
    model.add(Dense(units,activation = activation))
    model.add(Dense(units,activation = activation))
    model.add(Dense(1, activation = 'linear'))

    learning_rate = hp.Choice('learning_rate', values = [0.001,0.01,0.05])
    momentum = hp.Choice('momentum', values = [0.0,0.5,0.9])
    optimizer = SGD(learning_rate=learning_rate, momentum=momentum)
    model.compile(
    optimizer = optimizer,
    loss = 'mse',
    metrics = [keras.metrics.R2Score()]
    )
    return model
```

```
tuner = kt.RandomSearch(
    build_model,
    objective = 'val_r2',
    max_trials = 10,
    executions_per_trial = 1,
    direction = 'max'
)


tuner.search(x_train_scaled,y_train, epochs = 1000, validation_split = 0.2, batch_size = 128, callbacks = [keras.callbacks
best_model = tuner.get_bestmocels(num_models = 1)[0]

test_loss, test_r2 = best_model.evaluate(x_test_scaled,y_test)
print("Test R2:",test_r2)
# best_hps = tuner.get_best_hyperparameters
# print("Best Units:", best_hps.get('units'))
# print("Best dropout:",best_hps.get('dropout'))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[77], line 21
     14     model.compile(
     15     optimizer = optimizer,
     16     loss = 'mse',
     17     metrics = [keras.metrics.R2Score()]
     18     )
     19     return model
---> 21 tuner = kt.RandomSearch(
     22     build_model,
     23     objective = 'val_r2',
     24     max_trials = 10,
     25     executions_per_trial = 1,
     26     direction = 'max'
     27 )
     30 tuner.search(x_train_scaled,y_train, epochs = 1000, validation_split = 0.2, batch_size = 128, callbacks = [keras.ca
llbacks.EarlyStopping(monitor = 'val_loss', patience = 5)])
     31 best_model = tuner.get_bestmocels(num_models = 1)[0]

File ~\AppData\Roaming\Python\Python312\site-packages\keras_tuner\src\tuners\randomsearch.py:164, in RandomSearch.__init__
(self, hypermodel, objective, max_trials, seed, hyperparameters, tune_new_entries, allow_new_entries, max_retries_per_tria
l, max_consecutive_failed_trials, **kwargs)
    150 def __init__(
    151     self,
    152     hypermodel=None,
   (...)
    161     **kwargs
    162 ):
    163     self.seed = seed
--> 164     oracle = RandomSearchOracle(
    165         objective=objective,
    166         max_trials=max_trials,
    167         seed=seed,
    168         hyperparameters=hyperparameters,
    169         tune_new_entries=tune_new_entries,
    170         allow_new_entries=allow_new_entries,
    171         max_retries_per_trial=max_retries_per_trial,
    172         max_consecutive_failed_trials=max_consecutive_failed_trials,
    173     )
    174     super().__init__(oracle, hypermodel, **kwargs)

File ~\AppData\Roaming\Python\Python312\site-packages\keras_tuner\src\tuners\randomsearch.py:73, in RandomSearchOracle.__in
it__(self, objective, max_trials, seed, hyperparameters, allow_new_entries, tune_new_entries, max_retries_per_trial, max_co
nsecutive_failed_trials)
     62 def __init__(
     63     self,
     64     objective=None,
   (...)
     71     max_consecutive_failed_trials=3,
     72 ):
---> 73     super().__init__(
     74         objective=objective,
     75         max_trials=max_trials,
     76         hyperparameters=hyperparameters,
     77         tune_new_entries=tune_new_entries,
     78         allow_new_entries=allow_new_entries,
     79         seed=seed,
     80         max_retries_per_trial=max_retries_per_trial,
     81         max_consecutive_failed_trials=max_consecutive_failed_trials,
     82     )

File ~\AppData\Roaming\Python\Python312\site-packages\keras_tuner\src\engine\oracle.py:314, in Oracle.__init__(self, object
ive, max_trials, hyperparameters, allow_new_entries, tune_new_entries, seed, max_retries_per_trial, max_consecutive_failed_
trials)
    303 def __init__(
    304     self,
    305     objective=None,
   (...)
    312     max_consecutive_failed_trials=3,
    313 ):
--> 314     self.objective = obj_module.create_objective(objective)
    315     self.max_trials = max_trials
    316     if not hyperparameters:

File ~\AppData\Roaming\Python\Python312\site-packages\keras_tuner\src\engine\objective.py:155, in create_objective(objectiv
e)
    148     error_msg = (
    149         'Could not infer optimization direction ("min" or "max") '
    150         'for unknown metric "{obj}". Please specify the objective  as'
    151         "a `keras_tuner.Objective`, for example `keras_tuner.Objective("
    152         '"{obj}", direction="min")`.'
    153     )
    154     error_msg = error_msg.format(obj=objective)
--> 155     raise ValueError(error_msg)
    156 return Objective(name=objective, direction=direction)
```

```
ValueError: Could not infer optimization direction ("min" or "max") for unknown metric "val_r2". Please specify the objecti
ve  asa `keras_tuner.Objective`, for example `keras_tuner.Objective("val_r2", direction="min")`.
```

In [ ]: