A3-1102

Page No.
Date

Assignment

Title: Linear Regression using Deep Neural Network

Problem Statement: Implement Boston house price prediction problem by linear regression using Deep neural network. Use Boston house price prediction dataset

Objectives:
i) To implement linear regression using deep neural network for boston house price dataset
ii) To evaluate the performance of model

Outcomes:
Implementation of regression model for house price prediction using neural network.

Software and Hardware Requirements:
4 GB RAM, 500 GB HDD, PC, Intel i5
python programming, Ubuntu

Theory:

Neural Network
The term Artificial neural network is derived from Biological neural networks that develop the structure of human brain. Similar to human brain, ANN has neural connected to another in various layer of network. These neurons are called nodes.

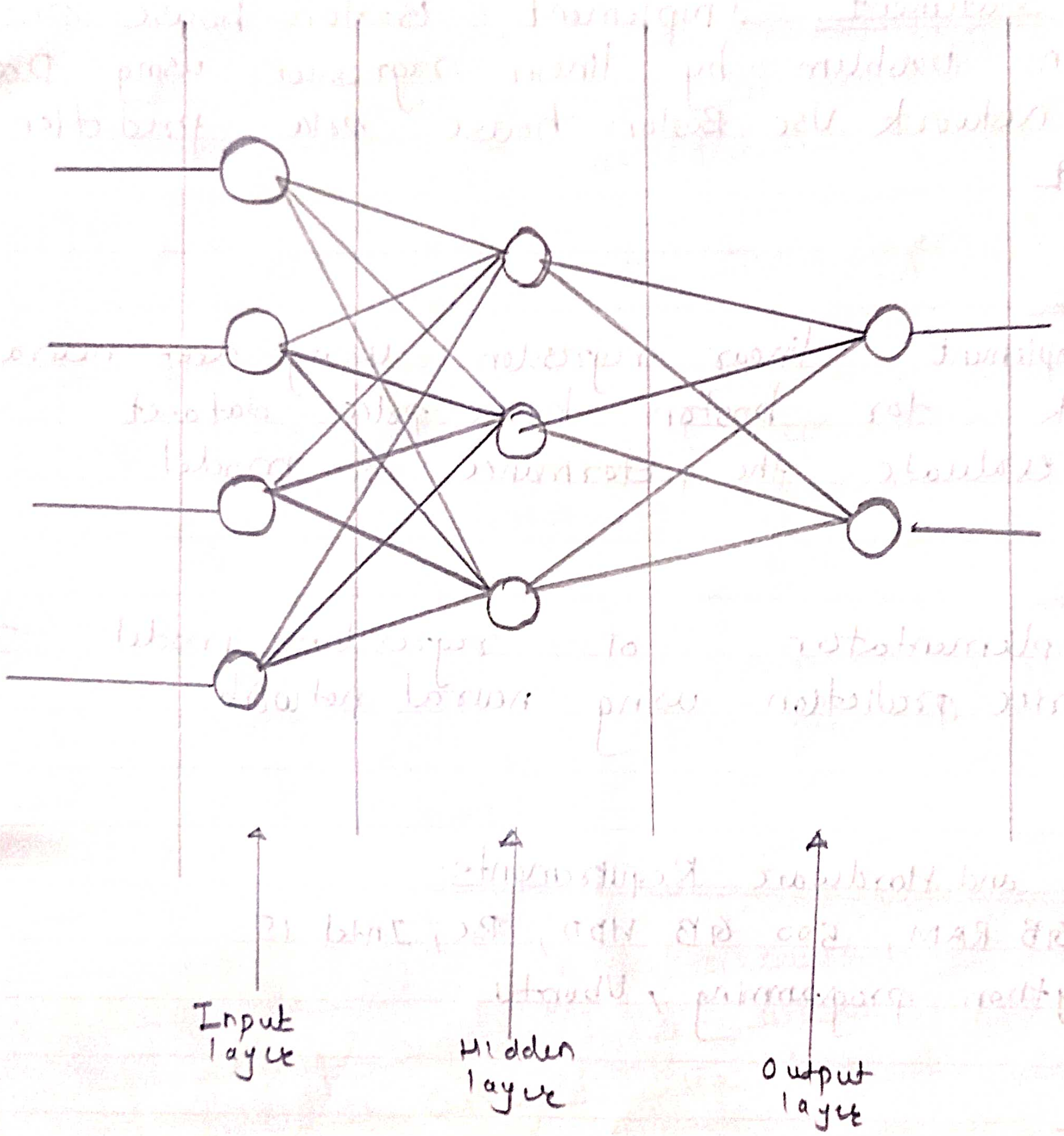


fig: Neural Network Architecture

Page No.	
Date	

The Architecture of an Artificial neural network

In order to define to define neural network that consists of large number of artificial neurons, which are termed units arranged in sequence of layers

ANN consists of three layers:

i) Input layer:
As name suggests, it accept input in several different formats provided by programmer.

ii) Hidden layer:
The hidden layer present in-between input and output layer. It performs all calculations to find hidden features and patterns.

iii) Output layer:
The input goes through a series of transformation using hidden layer, which finally results in output that is conveyed using layer.

The ANN takes input and computes weighted sum of inputs and includes a bias. This computation is represented in form of transfer function

$$\sum_{i=1}^{n} W_i * X_i + b$$

It determines weighted total is passed as input to an activation function to produce output. Activation

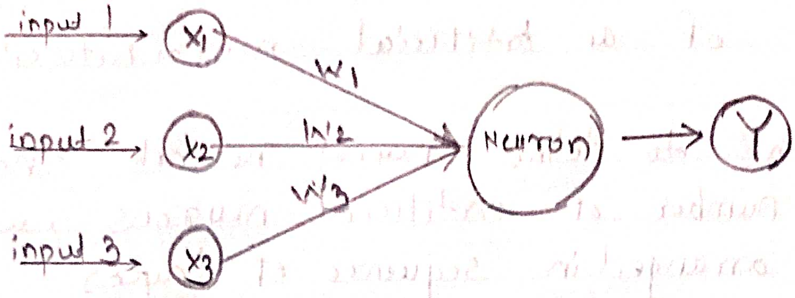


fig: Node in ANN

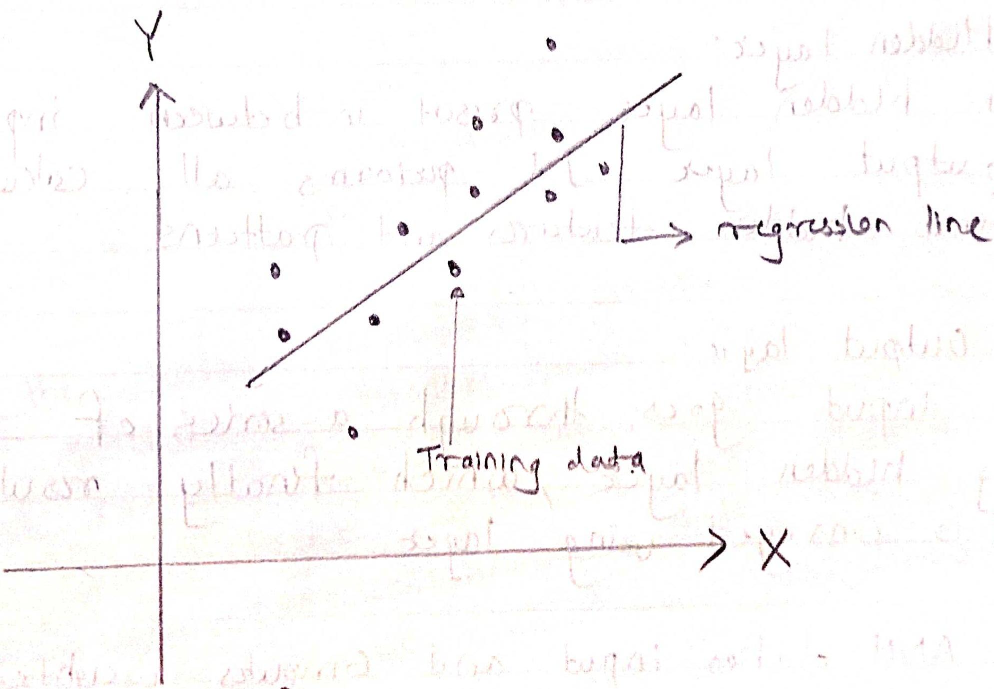


fig: Linear Regression

functions choose whether node should fire or not. only those who are fired make it to output layer.

Regression
It is defined as statistical method that helps us to analyze and understand the relationship between two or more variables.

Linear Regression is predictive model used for finding linear relationship between dependent variable and one or more independent variable.

$$Y = \alpha + Bx$$

Y = dependent variable
x = independent variable
α = intercept

Here, the linear regression can be implemented to predict prices of boston house dataset by using neural network with one or more hidden layer and output layer with linear activation function to get continous data i.e. to obtain behaviour of regression.

Conclusion:
Thus, we successfully implemented linear regression using neural network for prediction of boston house price dataset.

```python
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline

df=pd.read_csv('./BostonHousing.csv')

df
```

```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296

1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242

2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242

3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222

4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222

..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...

501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273

502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273

503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273

504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273

505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273


     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       ...     ...    ...   ...
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9
```

```
[506 rows x 14 columns]
```

```
df.shape
```

```
(506, 14)
```

```
df.describe()
```

```
             crim          zn       indus        chas         nox
rm  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695
6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878
0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000
3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000
5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000
6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000
6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000
8.780000

              age         dis         rad         tax     ptratio
b  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534
356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946
91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000
0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000
375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000
391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000
396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000
396.900000

            lstat        medv
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
```

```
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

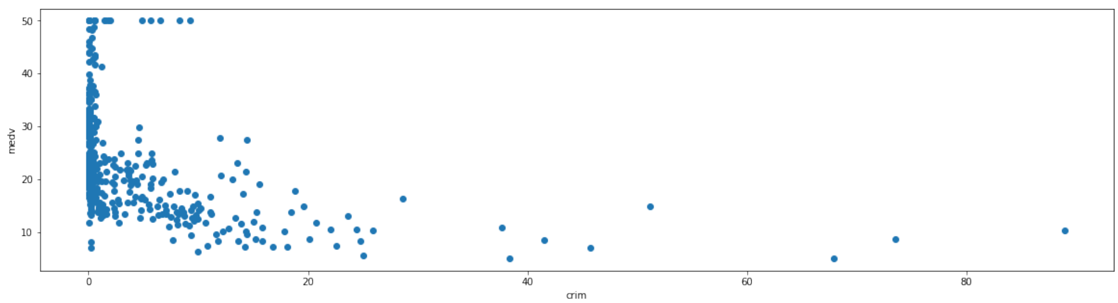
```
df.info()
```

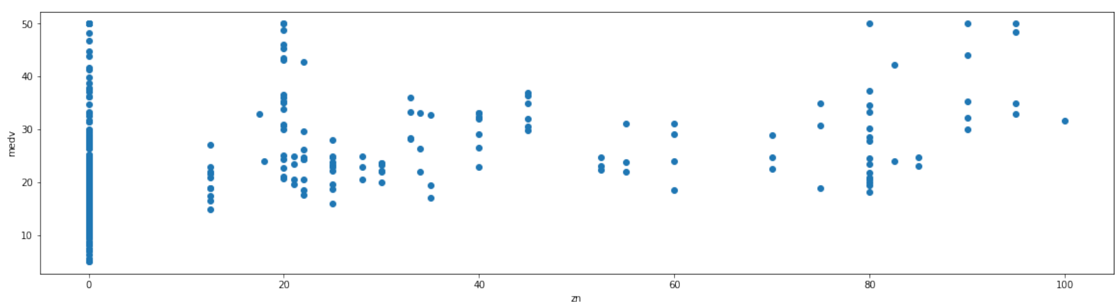
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

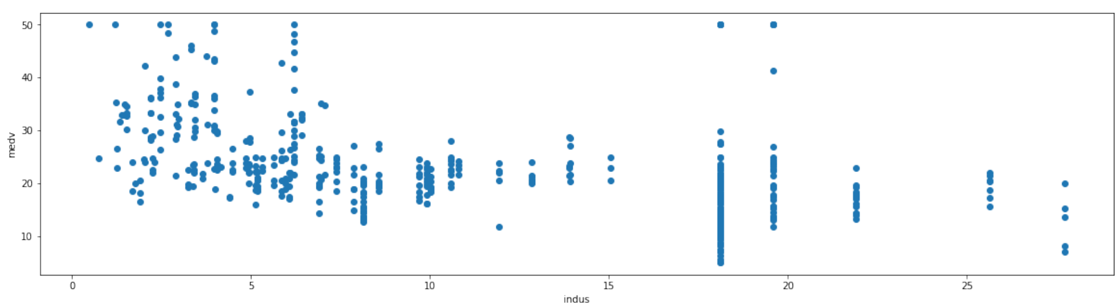
```
df.isnull().sum()
```

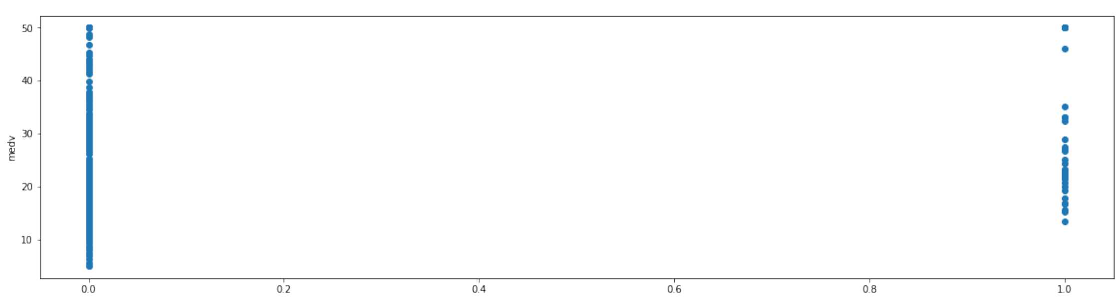
```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

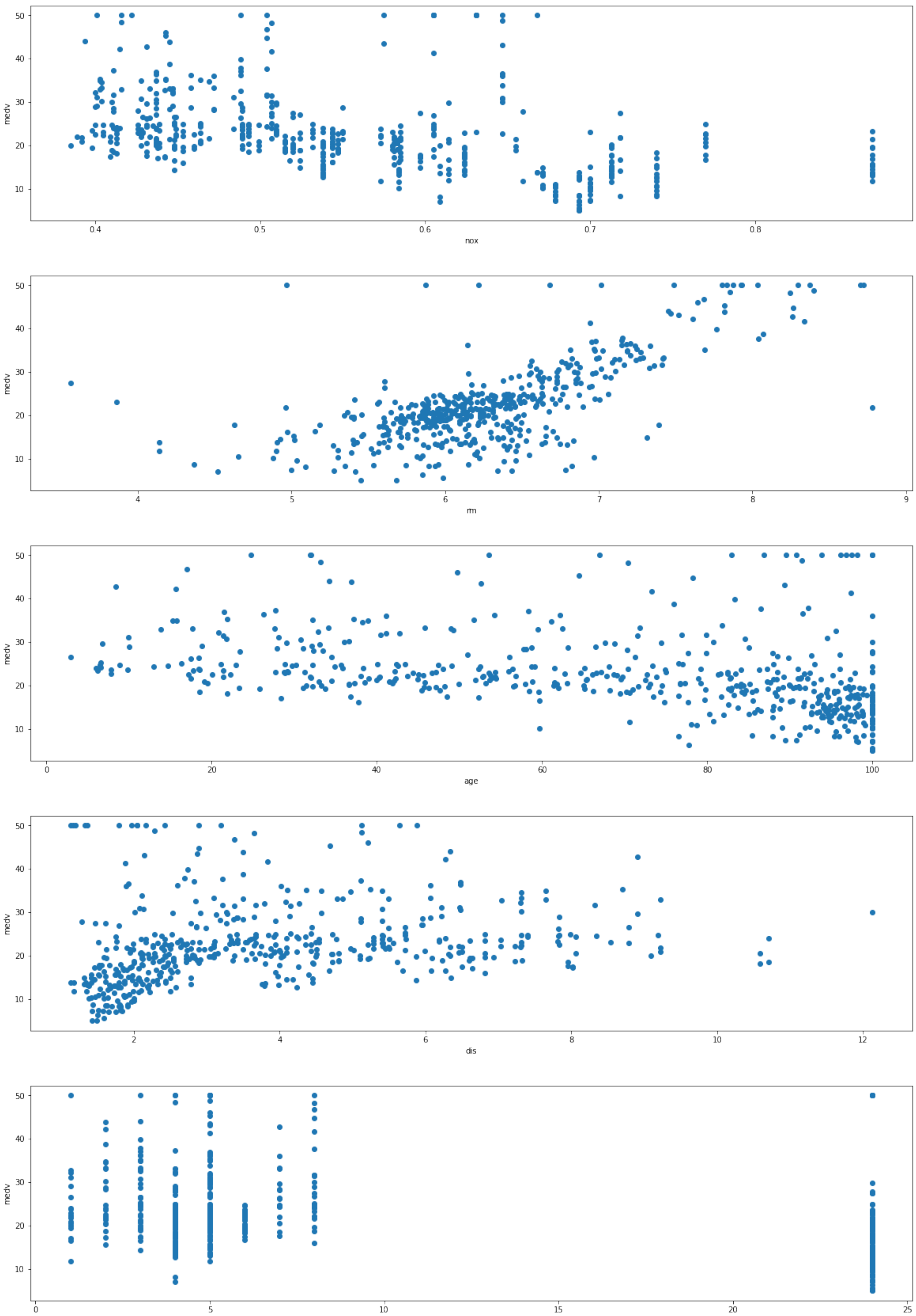
```
for column in df.columns[:-1]:
    plt.figure(figsize=(20, 5))
    if df[column].dtype in [np.int64, np.float64]: # only plot numeric
columns
        plt.scatter(df[column],df['medv'])
        plt.xlabel(column)
        plt.ylabel("medv")
        plt.show()
```



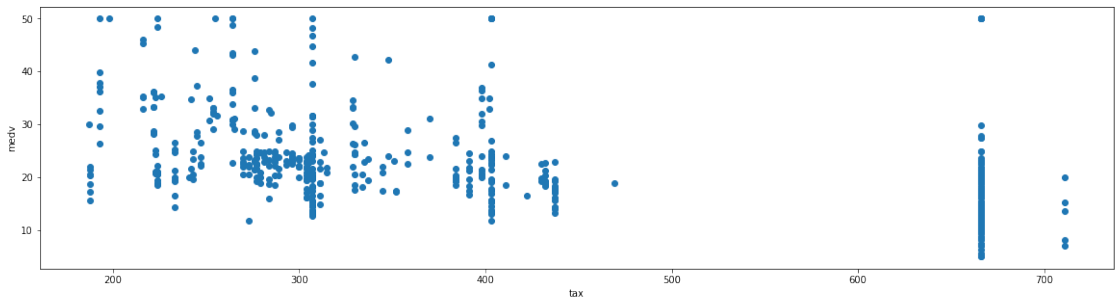


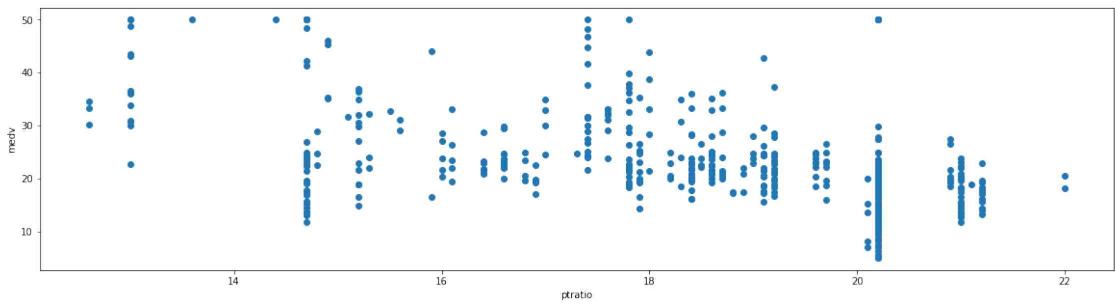


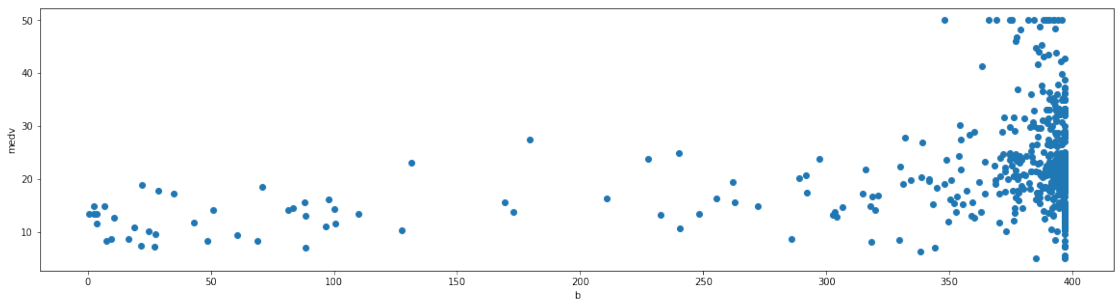


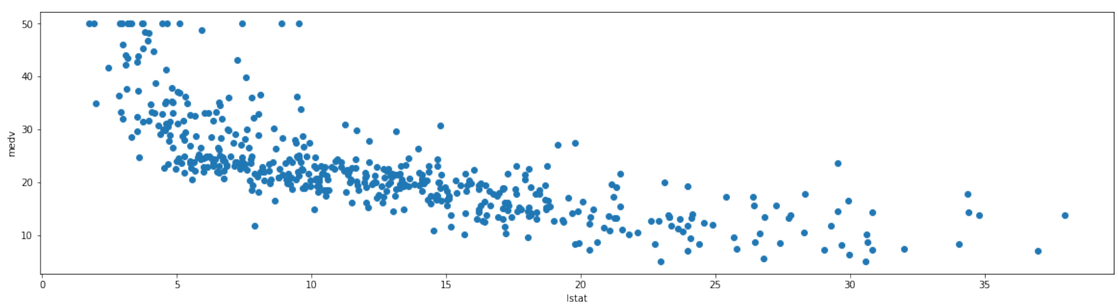


medv
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8
nox
medv
4
5
6
7
8
9
rm
medv
0
20
40
60
80
100
age
medv
2
4
6
8
10
12
dis
medv
0
5
10
15
20
25
rad










```
from sklearn.model_selection import train_test_split

X = df.loc[:, df.columns != 'medv']
y = df.loc[:, df.columns == 'medv']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=123)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train  = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

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

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(32, activation='relu', name='dense_3'))
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_3 (Dense)             (None, 32)                2080

 dense_output (Dense)        (None, 1)                 33

=================================================================
Total params: 12,161
Trainable params: 12,161
Non-trainable params: 0
_________________________________________________________________
```

```python
history = model.fit(X_train, y_train, epochs=100,
validation_split=0.05)
```

```
Epoch 1/100
11/11 [==============================] - 1s 25ms/step - loss: 553.9614
- mae: 21.7239 - val_loss: 546.4744 - val_mae: 21.5367
Epoch 2/100
11/11 [==============================] - 0s 6ms/step - loss: 449.0270
- mae: 19.1914 - val_loss: 414.4015 - val_mae: 18.4504
Epoch 3/100
11/11 [==============================] - 0s 7ms/step - loss: 295.7907
- mae: 14.9598 - val_loss: 234.2905 - val_mae: 12.9998
Epoch 4/100
11/11 [==============================] - 0s 7ms/step - loss: 142.8392
- mae: 9.7609 - val_loss: 94.0286 - val_mae: 7.4643
Epoch 5/100
11/11 [==============================] - 0s 7ms/step - loss: 82.9684 -
mae: 7.1674 - val_loss: 55.9259 - val_mae: 5.7147
```

```
Epoch 6/100
11/11 [==============================] - 0s 7ms/step - loss: 54.8113 -
mae: 5.6019 - val_loss: 38.5596 - val_mae: 5.1222
Epoch 7/100
11/11 [==============================] - 0s 7ms/step - loss: 36.2799 -
mae: 4.4309 - val_loss: 24.8239 - val_mae: 4.3848
Epoch 8/100
11/11 [==============================] - 0s 7ms/step - loss: 26.6273 -
mae: 3.7433 - val_loss: 17.7743 - val_mae: 3.7649
Epoch 9/100
11/11 [==============================] - 0s 8ms/step - loss: 22.5382 -
mae: 3.4324 - val_loss: 15.0153 - val_mae: 3.3740
Epoch 10/100
11/11 [==============================] - 0s 7ms/step - loss: 20.2401 -
mae: 3.2654 - val_loss: 13.4938 - val_mae: 3.1675
Epoch 11/100
11/11 [==============================] - 0s 7ms/step - loss: 18.5981 -
mae: 3.1350 - val_loss: 12.8117 - val_mae: 3.0182
Epoch 12/100
11/11 [==============================] - 0s 7ms/step - loss: 17.3478 -
mae: 2.9678 - val_loss: 12.1074 - val_mae: 2.9382
Epoch 13/100
11/11 [==============================] - 0s 7ms/step - loss: 16.3747 -
mae: 2.8881 - val_loss: 11.5875 - val_mae: 2.8861
Epoch 14/100
11/11 [==============================] - 0s 6ms/step - loss: 15.5649 -
mae: 2.8297 - val_loss: 11.0439 - val_mae: 2.7850
Epoch 15/100
11/11 [==============================] - 0s 6ms/step - loss: 14.9935 -
mae: 2.7929 - val_loss: 10.4282 - val_mae: 2.6771
Epoch 16/100
11/11 [==============================] - 0s 6ms/step - loss: 14.3178 -
mae: 2.6920 - val_loss: 10.0166 - val_mae: 2.5989
Epoch 17/100
11/11 [==============================] - 0s 6ms/step - loss: 13.7703 -
mae: 2.6131 - val_loss: 9.3626 - val_mae: 2.4998
Epoch 18/100
11/11 [==============================] - 0s 6ms/step - loss: 13.4195 -
mae: 2.5989 - val_loss: 9.0395 - val_mae: 2.4726
Epoch 19/100
11/11 [==============================] - 0s 6ms/step - loss: 12.8315 -
mae: 2.5295 - val_loss: 8.5669 - val_mae: 2.3789
Epoch 20/100
11/11 [==============================] - 0s 6ms/step - loss: 12.3685 -
mae: 2.5021 - val_loss: 8.4595 - val_mae: 2.3967
Epoch 21/100
11/11 [==============================] - 0s 6ms/step - loss: 11.9383 -
mae: 2.4633 - val_loss: 8.2673 - val_mae: 2.3606
Epoch 22/100
11/11 [==============================] - 0s 6ms/step - loss: 11.6795 -
```

```
mae: 2.4298 - val_loss: 8.1458 - val_mae: 2.3703
Epoch 23/100
11/11 [==============================] - 0s 6ms/step - loss: 11.4425 -
mae: 2.3911 - val_loss: 8.2050 - val_mae: 2.3621
Epoch 24/100
11/11 [==============================] - 0s 5ms/step - loss: 11.0579 -
mae: 2.3604 - val_loss: 8.0253 - val_mae: 2.3539
Epoch 25/100
11/11 [==============================] - 0s 6ms/step - loss: 10.7836 -
mae: 2.3346 - val_loss: 8.1134 - val_mae: 2.3375
Epoch 26/100
11/11 [==============================] - 0s 6ms/step - loss: 10.8582 -
mae: 2.3892 - val_loss: 8.2574 - val_mae: 2.3609
Epoch 27/100
11/11 [==============================] - 0s 6ms/step - loss: 10.3243 -
mae: 2.2843 - val_loss: 8.1586 - val_mae: 2.3230
Epoch 28/100
11/11 [==============================] - 0s 6ms/step - loss: 10.1193 -
mae: 2.2677 - val_loss: 8.3697 - val_mae: 2.3529
Epoch 29/100
11/11 [==============================] - 0s 7ms/step - loss: 9.9363 -
mae: 2.2643 - val_loss: 8.7163 - val_mae: 2.3936
Epoch 30/100
11/11 [==============================] - 0s 7ms/step - loss: 9.8210 -
mae: 2.2372 - val_loss: 8.1732 - val_mae: 2.3002
Epoch 31/100
11/11 [==============================] - 0s 6ms/step - loss: 9.5456 -
mae: 2.2223 - val_loss: 8.8148 - val_mae: 2.4145
Epoch 32/100
11/11 [==============================] - 0s 6ms/step - loss: 9.4497 -
mae: 2.1987 - val_loss: 8.2952 - val_mae: 2.2943
Epoch 33/100
11/11 [==============================] - 0s 6ms/step - loss: 9.3710 -
mae: 2.1674 - val_loss: 8.9753 - val_mae: 2.3792
Epoch 34/100
11/11 [==============================] - 0s 6ms/step - loss: 9.2314 -
mae: 2.1961 - val_loss: 8.5400 - val_mae: 2.2994
Epoch 35/100
11/11 [==============================] - 0s 7ms/step - loss: 9.1977 -
mae: 2.1547 - val_loss: 8.9302 - val_mae: 2.3529
Epoch 36/100
11/11 [==============================] - 0s 7ms/step - loss: 8.8633 -
mae: 2.1325 - val_loss: 8.8643 - val_mae: 2.3525
Epoch 37/100
11/11 [==============================] - 0s 7ms/step - loss: 8.8227 -
mae: 2.1209 - val_loss: 8.3004 - val_mae: 2.2436
Epoch 38/100
11/11 [==============================] - 0s 6ms/step - loss: 8.6271 -
mae: 2.0903 - val_loss: 9.0548 - val_mae: 2.3297
Epoch 39/100
```

```
11/11 [==============================] - 0s 7ms/step - loss: 8.6119 -
mae: 2.0942 - val_loss: 9.0082 - val_mae: 2.3184
Epoch 40/100
11/11 [==============================] - 0s 6ms/step - loss: 8.4840 -
mae: 2.0839 - val_loss: 9.0031 - val_mae: 2.2965
Epoch 41/100
11/11 [==============================] - 0s 6ms/step - loss: 8.3587 -
mae: 2.0597 - val_loss: 8.9746 - val_mae: 2.2878
Epoch 42/100
11/11 [==============================] - 0s 6ms/step - loss: 8.4980 -
mae: 2.1223 - val_loss: 10.0266 - val_mae: 2.3943
Epoch 43/100
11/11 [==============================] - 0s 6ms/step - loss: 8.1969 -
mae: 2.0745 - val_loss: 8.6045 - val_mae: 2.2008
Epoch 44/100
11/11 [==============================] - 0s 6ms/step - loss: 8.1376 -
mae: 2.0334 - val_loss: 9.3139 - val_mae: 2.2650
Epoch 45/100
11/11 [==============================] - 0s 7ms/step - loss: 8.0084 -
mae: 2.0104 - val_loss: 9.8029 - val_mae: 2.2923
Epoch 46/100
11/11 [==============================] - 0s 8ms/step - loss: 7.8781 -
mae: 2.0179 - val_loss: 10.0782 - val_mae: 2.3277
Epoch 47/100
11/11 [==============================] - 0s 7ms/step - loss: 7.8149 -
mae: 2.0233 - val_loss: 9.0977 - val_mae: 2.1866
Epoch 48/100
11/11 [==============================] - 0s 6ms/step - loss: 7.8271 -
mae: 1.9890 - val_loss: 9.5543 - val_mae: 2.2417
Epoch 49/100
11/11 [==============================] - 0s 7ms/step - loss: 7.7050 -
mae: 1.9986 - val_loss: 9.4340 - val_mae: 2.2638
Epoch 50/100
11/11 [==============================] - 0s 7ms/step - loss: 7.6853 -
mae: 1.9817 - val_loss: 9.8485 - val_mae: 2.2844
Epoch 51/100
11/11 [==============================] - 0s 8ms/step - loss: 7.3658 -
mae: 1.9477 - val_loss: 8.8812 - val_mae: 2.1189
Epoch 52/100
11/11 [==============================] - 0s 6ms/step - loss: 7.5616 -
mae: 1.9590 - val_loss: 9.5194 - val_mae: 2.2243
Epoch 53/100
11/11 [==============================] - 0s 6ms/step - loss: 7.3924 -
mae: 1.9661 - val_loss: 10.2065 - val_mae: 2.2912
Epoch 54/100
11/11 [==============================] - 0s 6ms/step - loss: 7.3755 -
mae: 1.9270 - val_loss: 9.0315 - val_mae: 2.2039
Epoch 55/100
11/11 [==============================] - 0s 6ms/step - loss: 7.2194 -
mae: 1.9325 - val_loss: 9.6028 - val_mae: 2.1837
```

```
Epoch 56/100
11/11 [==============================] - 0s 6ms/step - loss: 7.1025 -
mae: 1.9118 - val_loss: 9.2400 - val_mae: 2.1388
Epoch 57/100
11/11 [==============================] - 0s 5ms/step - loss: 7.0659 -
mae: 1.8972 - val_loss: 10.1312 - val_mae: 2.2945
Epoch 58/100
11/11 [==============================] - 0s 6ms/step - loss: 6.9200 -
mae: 1.8762 - val_loss: 9.4772 - val_mae: 2.1781
Epoch 59/100
11/11 [==============================] - 0s 6ms/step - loss: 6.8857 -
mae: 1.8839 - val_loss: 9.6909 - val_mae: 2.1942
Epoch 60/100
11/11 [==============================] - 0s 5ms/step - loss: 6.8317 -
mae: 1.8876 - val_loss: 9.5928 - val_mae: 2.1306
Epoch 61/100
11/11 [==============================] - 0s 6ms/step - loss: 6.7719 -
mae: 1.8546 - val_loss: 9.6127 - val_mae: 2.1800
Epoch 62/100

11/11 [==============================] - 0s 6ms/step - loss: 6.7402 -
mae: 1.8711 - val_loss: 9.9722 - val_mae: 2.1859
Epoch 63/100
11/11 [==============================] - 0s 7ms/step - loss: 6.7343 -
mae: 1.8408 - val_loss: 9.7255 - val_mae: 2.1716
Epoch 64/100
11/11 [==============================] - 0s 6ms/step - loss: 6.7530 -
mae: 1.8733 - val_loss: 10.4353 - val_mae: 2.3090
Epoch 65/100
11/11 [==============================] - 0s 6ms/step - loss: 6.6794 -
mae: 1.8623 - val_loss: 10.7995 - val_mae: 2.2779
Epoch 66/100
11/11 [==============================] - 0s 6ms/step - loss: 6.5190 -
mae: 1.8500 - val_loss: 9.9781 - val_mae: 2.2248
Epoch 67/100
11/11 [==============================] - 0s 6ms/step - loss: 6.4171 -
mae: 1.8094 - val_loss: 10.0287 - val_mae: 2.1373
Epoch 68/100
11/11 [==============================] - 0s 6ms/step - loss: 6.3551 -
mae: 1.8019 - val_loss: 10.8527 - val_mae: 2.3016
Epoch 69/100
11/11 [==============================] - 0s 6ms/step - loss: 6.2309 -
mae: 1.7992 - val_loss: 10.0595 - val_mae: 2.1660
Epoch 70/100
11/11 [==============================] - 0s 6ms/step - loss: 6.2855 -
mae: 1.7783 - val_loss: 10.5049 - val_mae: 2.2147
Epoch 71/100
11/11 [==============================] - 0s 6ms/step - loss: 6.2215 -
mae: 1.8077 - val_loss: 9.7213 - val_mae: 2.1732
Epoch 72/100
```

```
11/11 [==============================] - 0s 6ms/step - loss: 6.2204 -
mae: 1.7599 - val_loss: 11.3644 - val_mae: 2.2920
Epoch 73/100
11/11 [==============================] - 0s 5ms/step - loss: 6.2257 -
mae: 1.7637 - val_loss: 9.7730 - val_mae: 2.1660
Epoch 74/100
11/11 [==============================] - 0s 6ms/step - loss: 5.9683 -
mae: 1.7427 - val_loss: 10.7756 - val_mae: 2.1666
Epoch 75/100
11/11 [==============================] - 0s 6ms/step - loss: 6.0410 -
mae: 1.7435 - val_loss: 10.6788 - val_mae: 2.1901
Epoch 76/100
11/11 [==============================] - 0s 6ms/step - loss: 6.3671 -
mae: 1.8208 - val_loss: 10.4002 - val_mae: 2.1828
Epoch 77/100
11/11 [==============================] - 0s 6ms/step - loss: 6.3049 -
mae: 1.7825 - val_loss: 11.5569 - val_mae: 2.2415
Epoch 78/100
11/11 [==============================] - 0s 6ms/step - loss: 6.1379 -
mae: 1.7942 - val_loss: 10.1910 - val_mae: 2.1448
Epoch 79/100
11/11 [==============================] - 0s 6ms/step - loss: 5.6745 -
mae: 1.7061 - val_loss: 10.2436 - val_mae: 2.1073
Epoch 80/100
11/11 [==============================] - 0s 6ms/step - loss: 5.7007 -
mae: 1.7133 - val_loss: 10.4481 - val_mae: 2.1778
Epoch 81/100
11/11 [==============================] - 0s 6ms/step - loss: 5.7661 -
mae: 1.7143 - val_loss: 10.4360 - val_mae: 2.1153
Epoch 82/100
11/11 [==============================] - 0s 6ms/step - loss: 5.9167 -
mae: 1.7369 - val_loss: 10.6375 - val_mae: 2.0707
Epoch 83/100
11/11 [==============================] - 0s 6ms/step - loss: 5.5685 -
mae: 1.6842 - val_loss: 11.0820 - val_mae: 2.1982
Epoch 84/100
11/11 [==============================] - 0s 6ms/step - loss: 5.5068 -
mae: 1.6670 - val_loss: 10.6256 - val_mae: 2.0592
Epoch 85/100
11/11 [==============================] - 0s 6ms/step - loss: 5.5760 -
mae: 1.6913 - val_loss: 11.2542 - val_mae: 2.1728
Epoch 86/100
11/11 [==============================] - 0s 6ms/step - loss: 5.3687 -
mae: 1.6588 - val_loss: 10.8917 - val_mae: 2.1303
Epoch 87/100
11/11 [==============================] - 0s 7ms/step - loss: 5.4849 -
mae: 1.6708 - val_loss: 11.2007 - val_mae: 2.1430
Epoch 88/100
11/11 [==============================] - 0s 6ms/step - loss: 5.4138 -
mae: 1.6775 - val_loss: 10.3262 - val_mae: 2.0528
```

```
Epoch 89/100
11/11 [==============================] - 0s 6ms/step - loss: 5.4056 -
mae: 1.6628 - val_loss: 12.4708 - val_mae: 2.2902
Epoch 90/100
11/11 [==============================] - 0s 6ms/step - loss: 5.6487 -
mae: 1.6842 - val_loss: 9.7076 - val_mae: 1.9427
Epoch 91/100
11/11 [==============================] - 0s 7ms/step - loss: 5.3223 -
mae: 1.6506 - val_loss: 12.4144 - val_mae: 2.2861
Epoch 92/100
11/11 [==============================] - 0s 7ms/step - loss: 5.1950 -
mae: 1.6345 - val_loss: 10.1571 - val_mae: 2.0346
Epoch 93/100
11/11 [==============================] - 0s 6ms/step - loss: 5.4909 -
mae: 1.6599 - val_loss: 10.9529 - val_mae: 2.1014
Epoch 94/100
11/11 [==============================] - 0s 6ms/step - loss: 5.0980 -
mae: 1.6090 - val_loss: 11.4283 - val_mae: 2.1288
Epoch 95/100
11/11 [==============================] - 0s 6ms/step - loss: 5.1869 -
mae: 1.6305 - val_loss: 10.9258 - val_mae: 2.0827
Epoch 96/100
11/11 [==============================] - 0s 6ms/step - loss: 5.2914 -
mae: 1.6691 - val_loss: 10.6277 - val_mae: 2.1107
Epoch 97/100
11/11 [==============================] - 0s 7ms/step - loss: 5.3566 -
mae: 1.6499 - val_loss: 11.7264 - val_mae: 2.2082
Epoch 98/100
11/11 [==============================] - 0s 8ms/step - loss: 5.0180 -
mae: 1.6102 - val_loss: 10.7876 - val_mae: 2.0310
Epoch 99/100
11/11 [==============================] - 0s 6ms/step - loss: 4.8986 -
mae: 1.5904 - val_loss: 10.5688 - val_mae: 2.0737
Epoch 100/100
11/11 [==============================] - 0s 7ms/step - loss: 4.8678 -
mae: 1.5712 - val_loss: 11.3624 - val_mae: 2.0756
```

```
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 3ms/step - loss: 16.3030 -
mae: 2.6288
Mean squared error on test data:  16.302959442138672
Mean absolute error on test data:  2.628826379776001
```

```
model.predict(sc.transform([[0.33147,0,6.2,0,0.507,8.247,70.4,3.6519,8
,307,17.4,378.95,3.95]]))
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:441: UserWarning: X does not have valid
feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

```
array([[44.85821]], dtype=float32)
```