DL Assignment 1

Name: Ankita Singh || Roll no: 57

Problem Statement: Linear regression by using Deep Neural network: Implement Boston housing price prediction problem by linear regression using Deep Neural network. Use Boston House price prediction dataset.

In [50]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
```

In [31]:
```
df = pd.read_csv("boston_train.csv")
```

In [33]:
```
df.head()
```

Out[33]:

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	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	2	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	4	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
3	5	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
4	7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 333 entries, 0 to 332
Data columns (total 15 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   ID       333 non-null    int64  
 1   crim     333 non-null    float64
 2   zn       333 non-null    float64
 3   indus    333 non-null    float64
 4   chas     333 non-null    int64  
 5   nox      333 non-null    float64
 6   rm       333 non-null    float64
 7   age      333 non-null    float64
 8   dis      333 non-null    float64
 9   rad      333 non-null    int64  
 10  tax      333 non-null    int64  
 11  ptratio  333 non-null    float64
 12  black    333 non-null    float64
 13  lstat    333 non-null    float64
 14  medv     333 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 39.2 KB
```

In [37]:
```
df.describe()
```

Out[37]:

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad
count	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000
mean	250.951952	3.360341	10.689189	11.293483	0.060060	0.557144	6.265619	68.226426	3.709934	9.633634
std	147.859438	7.352272	22.674762	6.998123	0.237956	0.114955	0.703952	28.133344	1.981123	8.742174
min	1.000000	0.006320	0.000000	0.740000	0.000000	0.385000	3.561000	6.000000	1.129600	1.000000
25%	123.000000	0.078960	0.000000	5.130000	0.000000	0.453000	5.884000	45.400000	2.122400	4.000000
50%	244.000000	0.261690	0.000000	9.900000	0.000000	0.538000	6.202000	76.700000	3.092300	5.000000
75%	377.000000	3.678220	12.500000	18.100000	0.000000	0.631000	6.595000	93.800000	5.116700	24.000000
max	506.000000	73.534100	100.000000	27.740000	1.000000	0.871000	8.725000	100.000000	10.710300	24.000000

In [39]:
```
df.isnull().sum()
```

Out[39]:

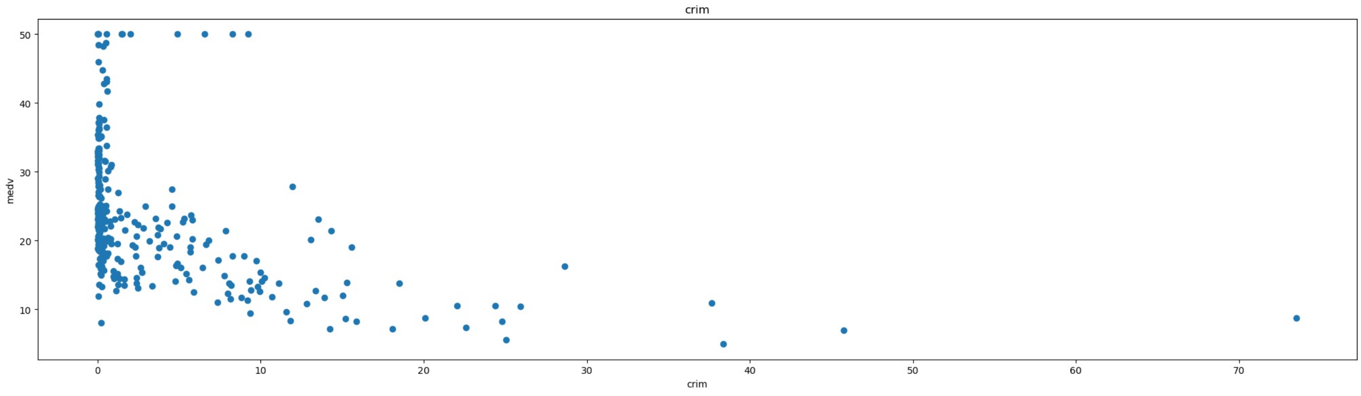
```
ID         0
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
black      0
lstat      0
medv       0
dtype: int64
```

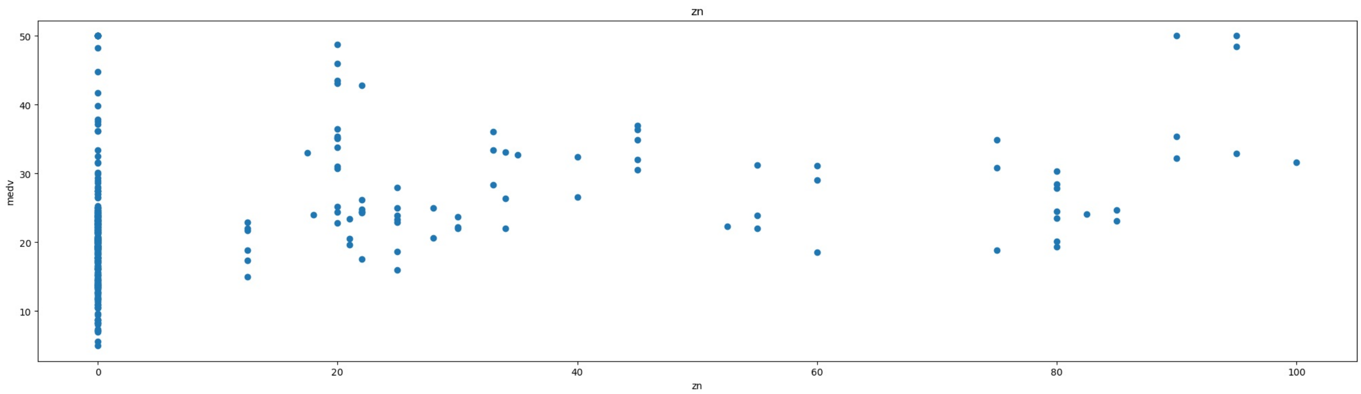
In [44]:

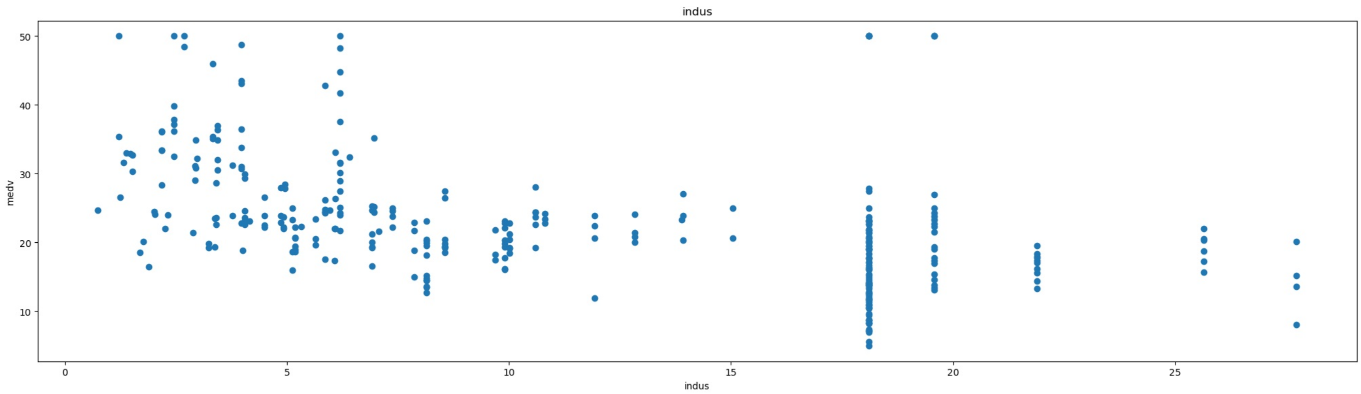
```
df.drop('ID', axis = 1, inplace=True)
```

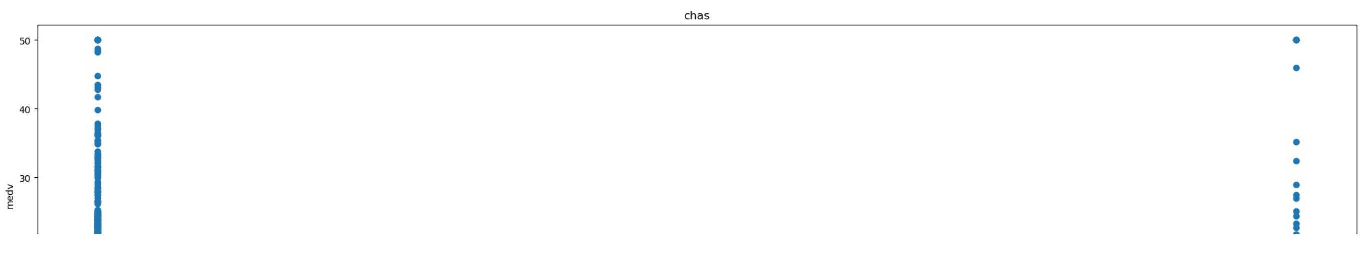
In [46]:

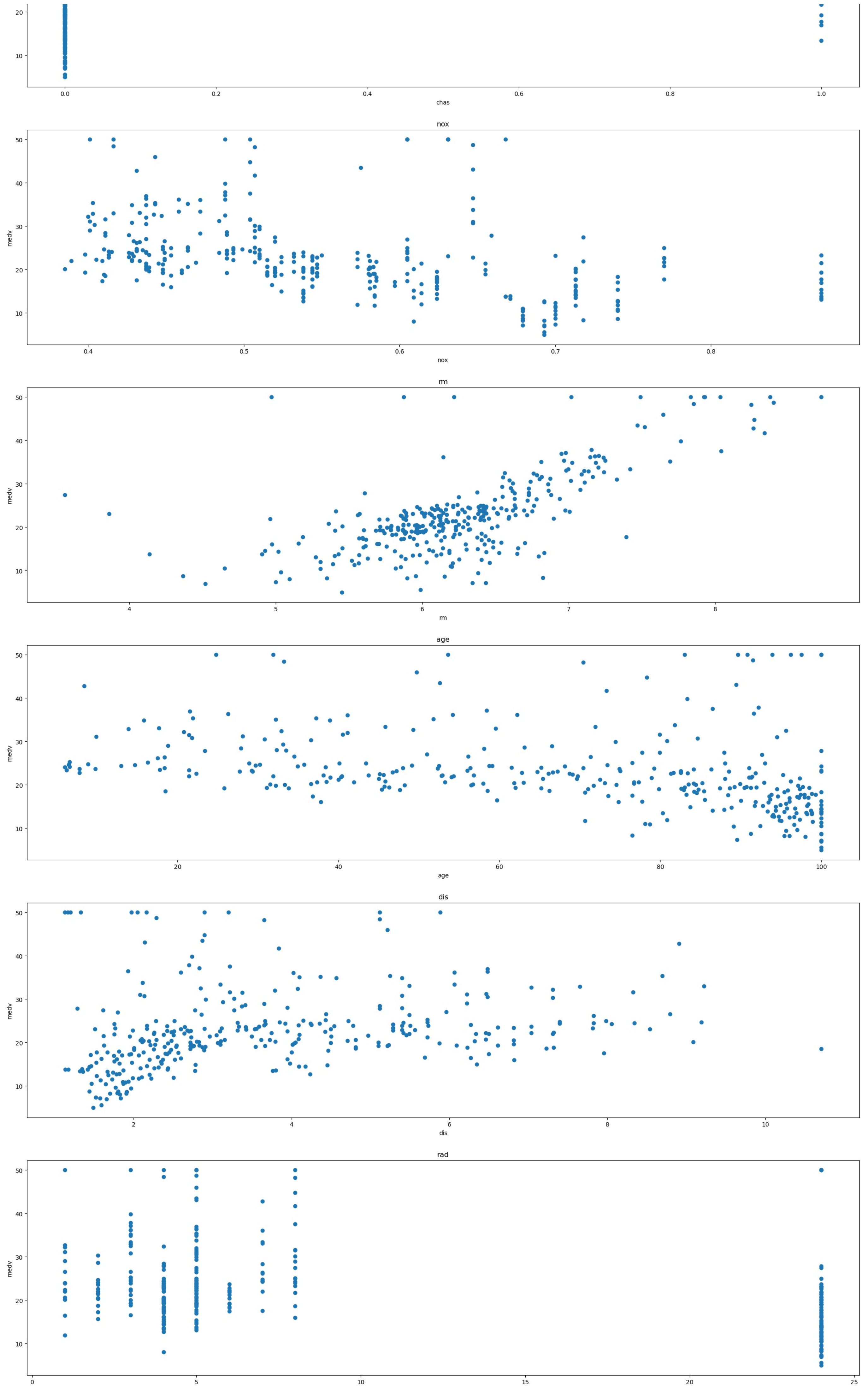
```
fig, axs = plt.subplots(13, 1, figsize=(25, 100))
axs = axs.ravel()
# plot each feature against the target variable
for i, column in enumerate(df.columns[:-1]):
    axs[i].scatter(df[column], df["medv"])
    axs[i].set_title(column)
    axs[i].set_xlabel(column)
    axs[i].set_ylabel("medv")
```



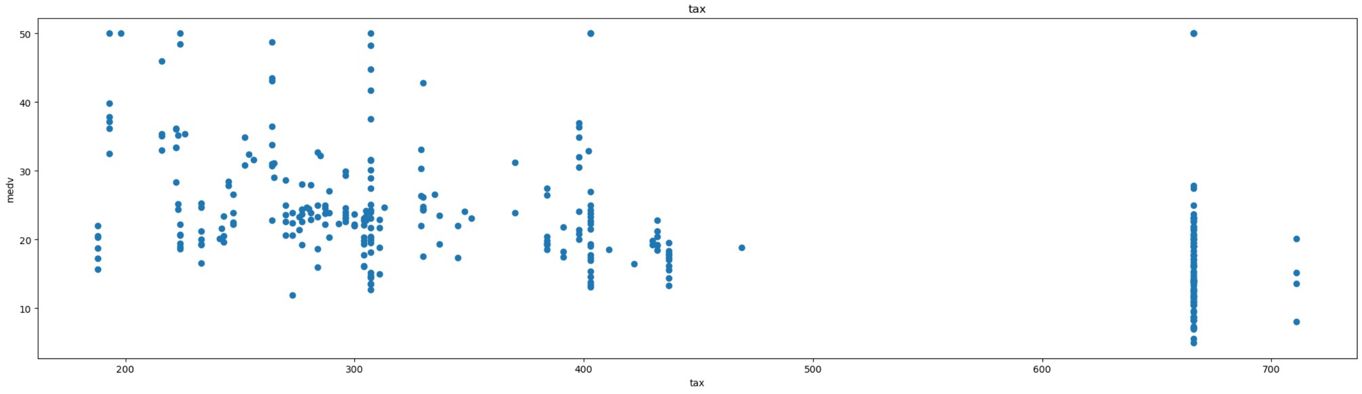


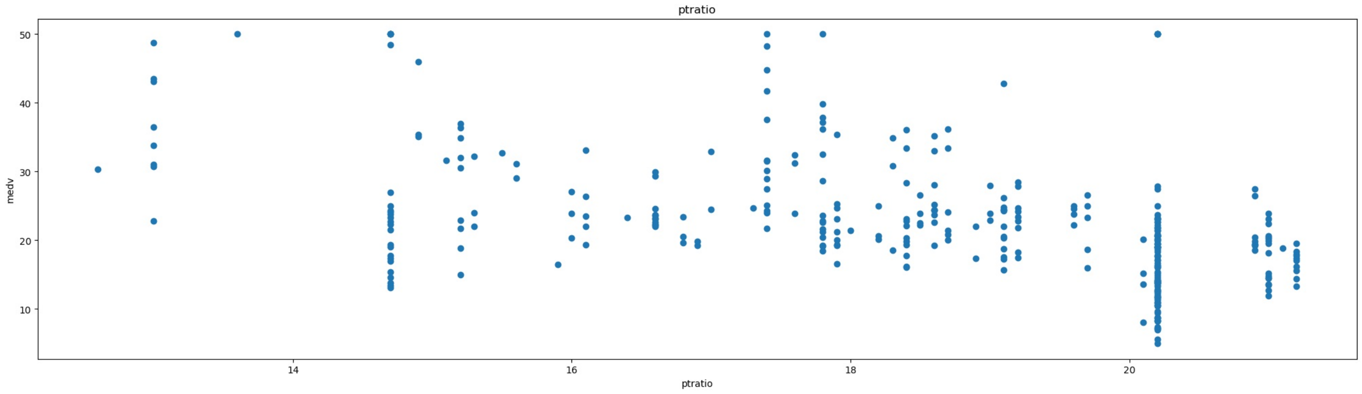


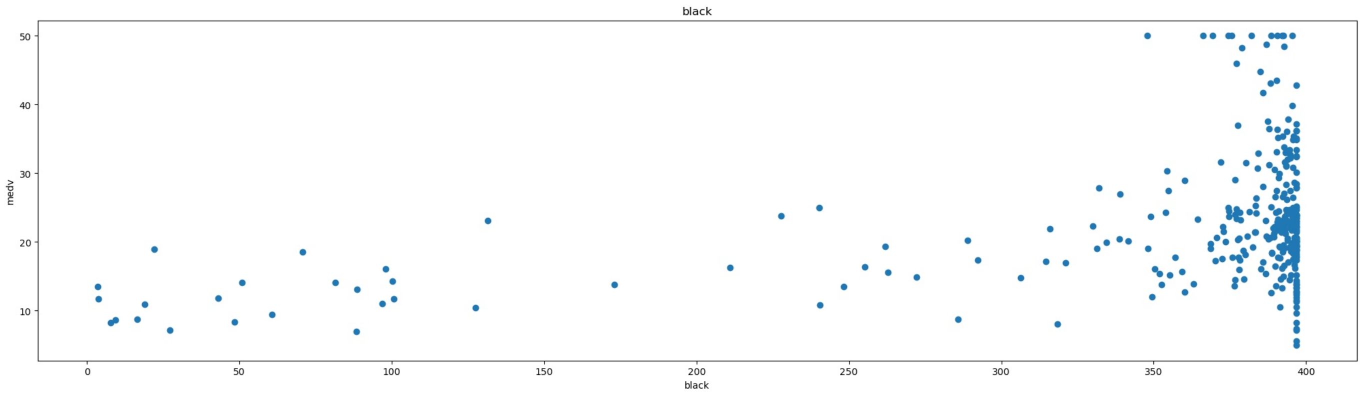


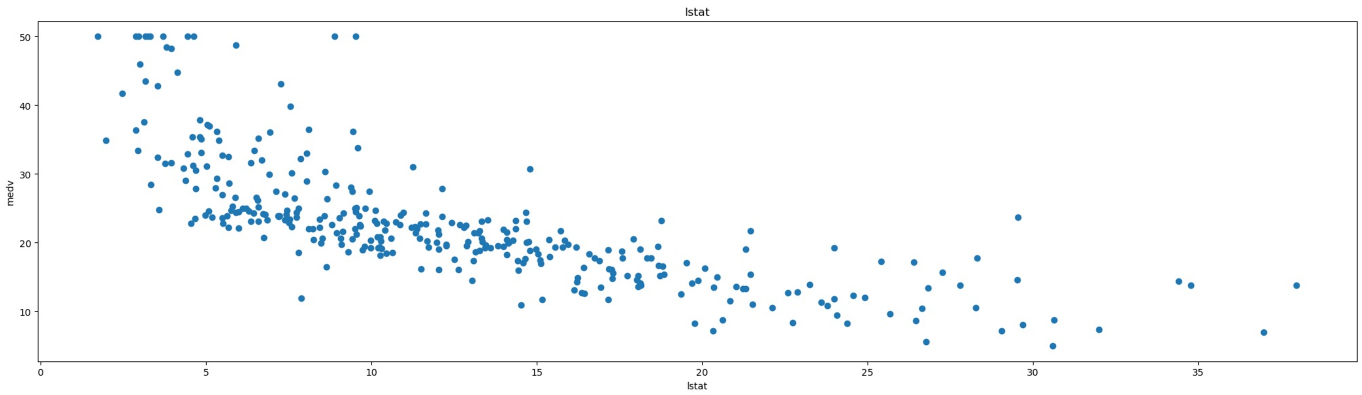

chas
nox
medv
nox
rm
medv
rm
age
medv
age
dis
medv
dis
rad
medv

rad



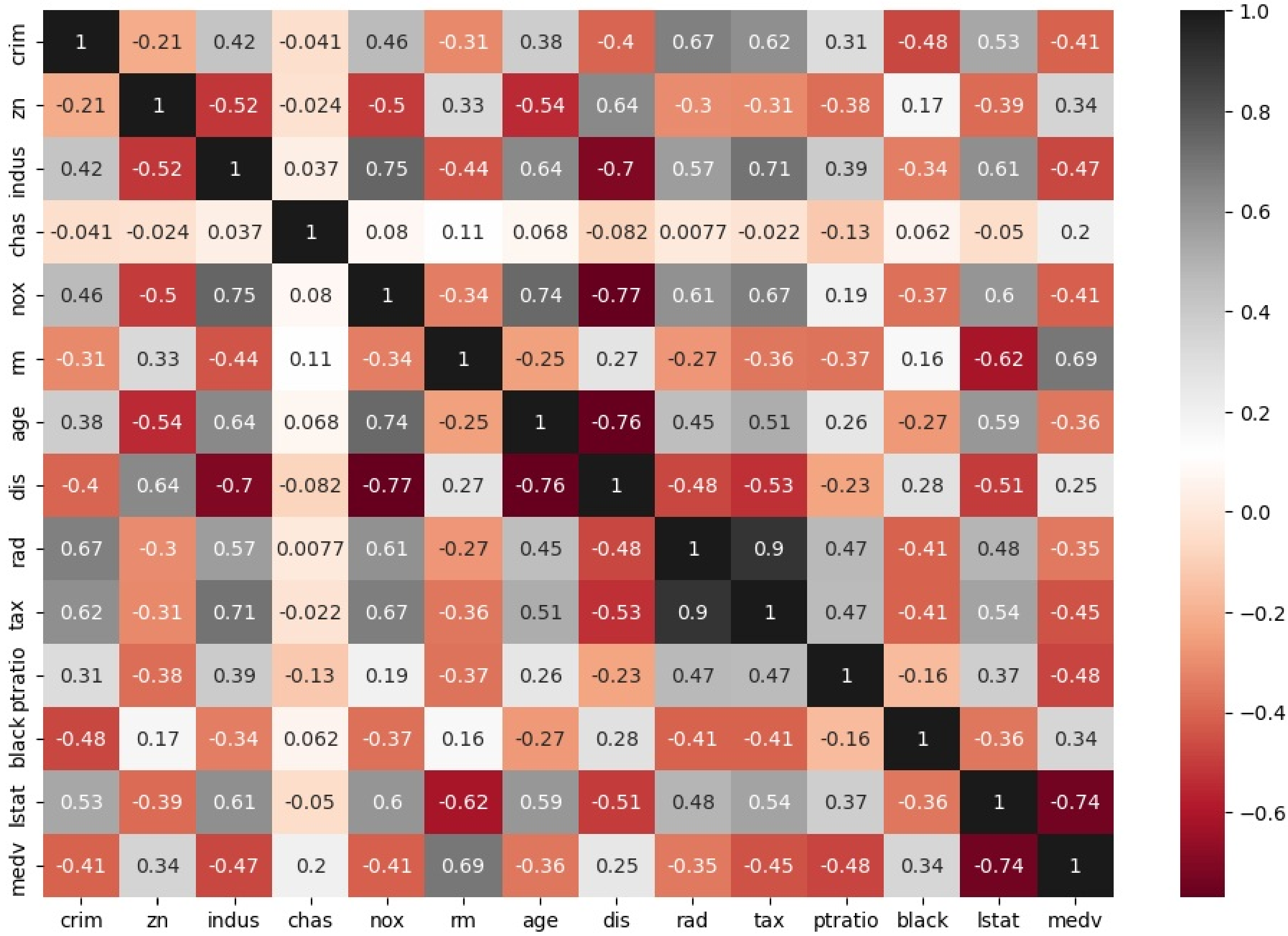






In [52]:
```python
plt.subplots(figsize=(12,8))
sb.heatmap(df.corr(), cmap = 'RdGy', annot=True)
```

Out[52]: <Axes: >



In [110…
```
x = df[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax', 'ptratio', 'black', 'lstat']]
y = df['medv']
```

In [112…
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

In [114…
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=5)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(233, 13)
(100, 13)
(233,)
(100,)
```

In [116…
```
lin_model = LinearRegression()
lin_model.fit(x_train,y_train)
```

Out[116…
```
▾ LinearRegression
LinearRegression()
```

In [118…
```
# model evaluation for training set
y_train_predict = lin_model.predict(x_train)
```

```
rmse = (np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))
r2 = metrics.r2_score(y_train, y_train_predict)
print("The model performance for training set")
print("--------------------------------------")
print('RMSE is ',rmse)
print('R2 score is ',r2)
```

```
The model performance for training set
--------------------------------------
RMSE is  4.808981791245613
R2 score is  0.7346628912448181
```

In [120…

```
# model evaluation for testing set
y_test_predict = lin_model.predict(x_test)
rmse = (np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))
r2 = metrics.r2_score(y_test, y_test_predict)
print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is ',rmse)
print('R2 score is ',r2)
```

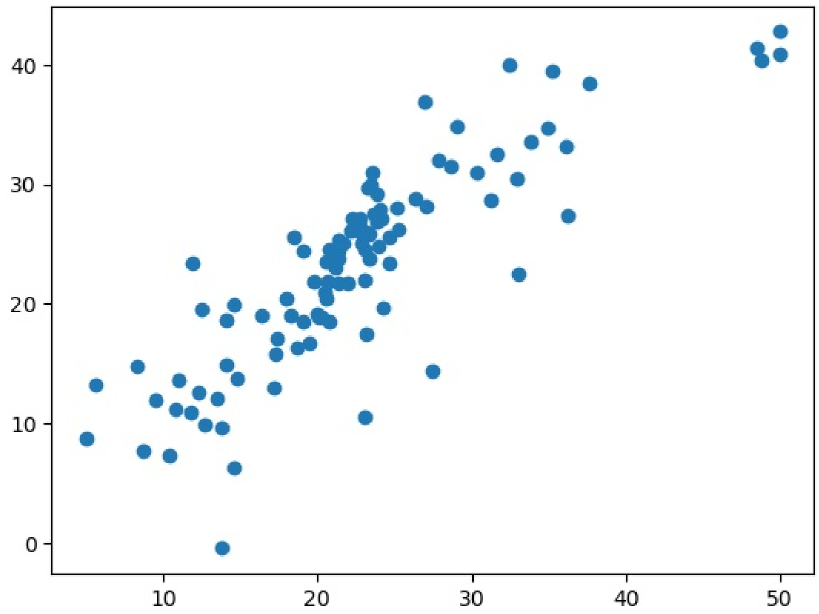
```
The model performance for testing set
--------------------------------------
RMSE is  4.8048943670005935
R2 score is  0.6971279421565907
```

In [122…

```
plt.scatter(y_test, y_test_predict)
```

Out[122…

```
<matplotlib.collections.PathCollection at 0x7be44ddcc380>
```



In [124…

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [126…

```
from tensorflow.keras.models import Sequential
model = Sequential()
```

In [128…

```
from tensorflow.keras.layers import Dense
model.add(Dense(units=128, activation='relu', input_shape=(13,)))
model.add(Dense(units = 64, activation='relu'))
model.add(Dense(units = 32, activation='relu'))
model.add(Dense(units = 16, activation='relu'))
model.add(Dense(1))
```

```
/home/sys-08/anaconda3/lib/python3.12/site-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [130…

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 128)	1,792
dense_6 (Dense)	(None, 64)	8,256
dense_7 (Dense)	(None, 32)	2,080
dense_8 (Dense)	(None, 16)	528
dense_9 (Dense)	(None, 1)	17

Total params: 12,673 (49.50 KB)

Trainable params: 12,673 (49.50 KB)

Non-trainable params: 0 (0.00 B)

In [132…
```
model.compile(optimizer = 'adam', loss='mean_squared_error', metrics=['mae'])
```

In [154…
```
history = model.fit(x_train, y_train, epochs=35, verbose=1, validation_split=0.05)
```

```
Epoch 1/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 4.1595 - mae: 1.5253 - val_loss: 7.4471 - val_mae: 2.4091
Epoch 2/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.1143 - mae: 1.4808 - val_loss: 7.7857 - val_mae: 2.4725
Epoch 3/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.7514 - mae: 1.4572 - val_loss: 7.1358 - val_mae: 2.3341
Epoch 4/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.3392 - mae: 1.5399 - val_loss: 7.4667 - val_mae: 2.4170
Epoch 5/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.7282 - mae: 1.3953 - val_loss: 7.3824 - val_mae: 2.4029
Epoch 6/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 3.3710 - mae: 1.4179 - val_loss: 7.1983 - val_mae: 2.3524
Epoch 7/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 3.7191 - mae: 1.3954 - val_loss: 7.6354 - val_mae: 2.4394
Epoch 8/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.0380 - mae: 1.3916 - val_loss: 7.1839 - val_mae: 2.3271
Epoch 9/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.2029 - mae: 1.3464 - val_loss: 7.2638 - val_mae: 2.3719
Epoch 10/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.1767 - mae: 1.3139 - val_loss: 6.9506 - val_mae: 2.3071
Epoch 11/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 3.6562 - mae: 1.4335 - val_loss: 7.4381 - val_mae: 2.4253
Epoch 12/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.3935 - mae: 1.3226 - val_loss: 7.2290 - val_mae: 2.3264
Epoch 13/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.3342 - mae: 1.3482 - val_loss: 7.3395 - val_mae: 2.3727
Epoch 14/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.1989 - mae: 1.2841 - val_loss: 7.0452 - val_mae: 2.3076
Epoch 15/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.5879 - mae: 1.3162 - val_loss: 7.1029 - val_mae: 2.3459
Epoch 16/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.9341 - mae: 1.2763 - val_loss: 6.6961 - val_mae: 2.2328
Epoch 17/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.6032 - mae: 1.2195 - val_loss: 7.0552 - val_mae: 2.3231
Epoch 18/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.6562 - mae: 1.2179 - val_loss: 6.8724 - val_mae: 2.2509
Epoch 19/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.7891 - mae: 1.2430 - val_loss: 7.2987 - val_mae: 2.3768
Epoch 20/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.6787 - mae: 1.2102 - val_loss: 6.7910 - val_mae: 2.2616
Epoch 21/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.5676 - mae: 1.2085 - val_loss: 7.1002 - val_mae: 2.3412
Epoch 22/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.9265 - mae: 1.2682 - val_loss: 6.7201 - val_mae: 2.2609
Epoch 23/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.8568 - mae: 1.2500 - val_loss: 6.5655 - val_mae: 2.2089
Epoch 24/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.9947 - mae: 1.2390 - val_loss: 6.8390 - val_mae: 2.2832
Epoch 25/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.8521 - mae: 1.2429 - val_loss: 6.9223 - val_mae: 2.2802
Epoch 26/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.4424 - mae: 1.1646 - val_loss: 6.5909 - val_mae: 2.2064
Epoch 27/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.5169 - mae: 1.1640 - val_loss: 6.7905 - val_mae: 2.2470
Epoch 28/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.2553 - mae: 1.1150 - val_loss: 6.4551 - val_mae: 2.1823
Epoch 29/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.3109 - mae: 1.0925 - val_loss: 6.6893 - val_mae: 2.2119
Epoch 30/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.5058 - mae: 1.1352 - val_loss: 6.7094 - val_mae: 2.2582
Epoch 31/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.5846 - mae: 1.1840 - val_loss: 6.5458 - val_mae: 2.1307
Epoch 32/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 3.0317 - mae: 1.2911 - val_loss: 7.0294 - val_mae: 2.3110
Epoch 33/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 2.2801 - mae: 1.1083 - val_loss: 6.2150 - val_mae: 2.0555
Epoch 34/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.2108 - mae: 1.1322 - val_loss: 6.4313 - val_mae: 2.1599
Epoch 35/35
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.7468 - mae: 1.2093 - val_loss: 6.6994 - val_mae: 2.2349
```

In [156…

```python
#Evaluation of the model
y_pred = model.predict(x_test)
mse_nn, mae_nn = model.evaluate(x_test, y_test)
print('Mean absolute error on test data using NN: ', mae_nn)
print('Mean squared error on test data using NN: ', mse_nn)
print('RMSE using NN:', np.sqrt(mse_nn))
```

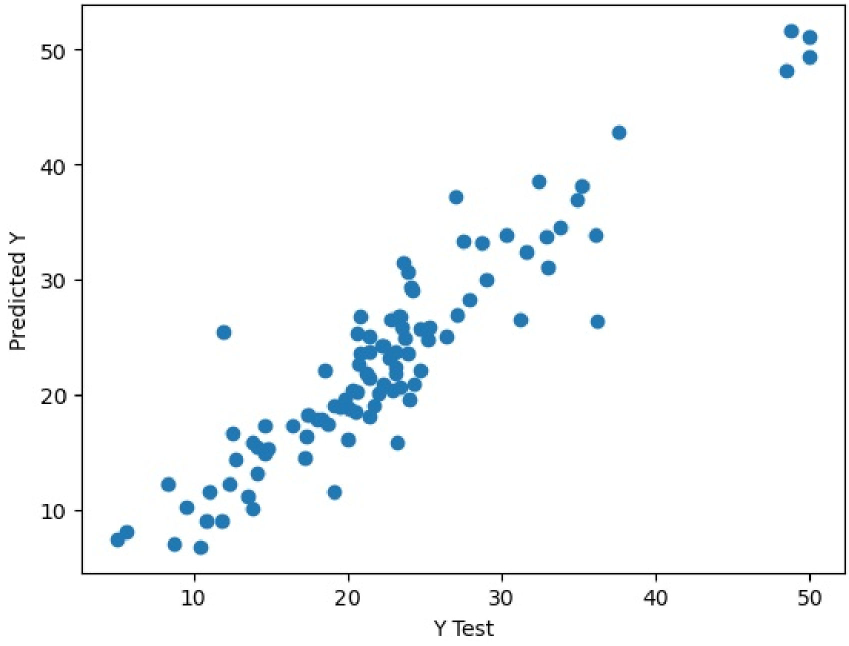
```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 13.0927 - mae: 2.7069
Mean absolute error on test data using NN:  2.562872886657715
Mean squared error on test data using NN:  12.33912467956543
RMSE using NN: 3.5127090229003355
```

In [158…

```
plt.scatter(y_test,y_pred)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[158…

```
Text(0, 0.5, 'Predicted Y')
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js