MACHINE LEARNING LAB ETE

Write a program to demonstrate the working of the Simple Linear Regression. Use an appropriate data set the implementation.

DATASET- https://www.kaggle.com/henriqueyamahata/boston-housing-with-linear-regression?select=boston_train.csv

Boston Housing with Linear Regression

With this data our objective is create a model using linear regression to predict the houses price

The data contains the following columns:

'crim': per capita crime rate by town.

'zn': proportion of residential land zoned for lots over 25,000 sq.ft.

'indus': proportion of non-retail business acres per town.

'chas':Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).

'nox': nitrogen oxides concentration (parts per 10 million).

'rm': average number of rooms per dwelling.

'age': proportion of owner-occupied units built prior to 1940.

'dis': weighted mean of distances to five Boston employment centres.

'rad': index of accessibility to radial highways.

'tax': full-value property-tax rate per $10,000.

'ptratio': pupil-teacher ratio by town

'black': 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.

'lstat': lower status of the population (percent).

'medv': median value of owner-occupied homes in $$1000s

GITHUB- https://github.com/sreesti/ETE-18SCSE1010482

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
```

In [2]:

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
dataset = pd.read_csv("Downloads/boston_train.csv")
```

In [3]:

```
dataset.head()
```

Out[3]:

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lst
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.9
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.1
2	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.9
3	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.3
4	7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.4

In [4]:

```
dataset.info()
dataset.describe()
```

```
<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: 39.1 KB
```

Out[4]:

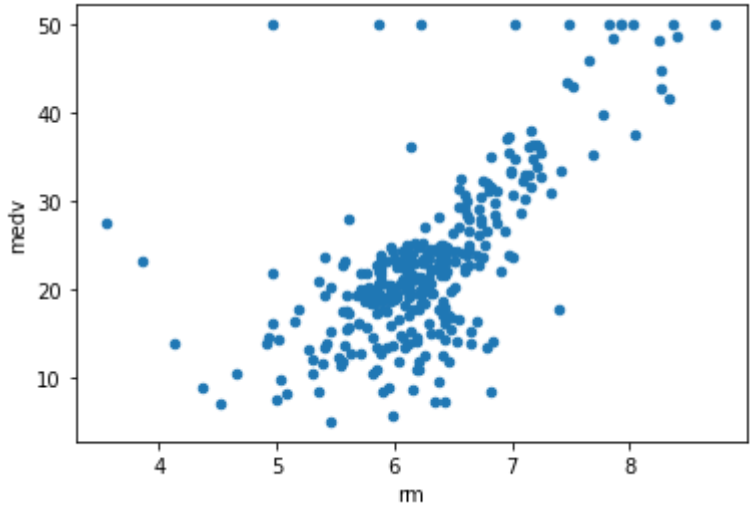
	ID	crim	zn	indus	chas	nox	rm
count	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000
mean	250.951952	3.360341	10.689189	11.293483	0.060060	0.557144	6.265619
std	147.859438	7.352272	22.674762	6.998123	0.237956	0.114955	0.703952
min	1.000000	0.006320	0.000000	0.740000	0.000000	0.385000	3.561000
25%	123.000000	0.078960	0.000000	5.130000	0.000000	0.453000	5.884000
50%	244.000000	0.261690	0.000000	9.900000	0.000000	0.538000	6.202000
75%	377.000000	3.678220	12.500000	18.100000	0.000000	0.631000	6.595000
max	506.000000	73.534100	100.000000	27.740000	1.000000	0.871000	8.725000

In [5]:

```
#ID columns does not relevant for our analysis.
dataset.drop('ID', axis = 1, inplace=True)
dataset.plot.scatter('rm', 'medv')
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19f957cb988>
```



In [6]:

```
dataset.isnull().sum()
```

Out[6]:

```
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 [7]:

```
X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values.reshape(-1,1)
```

In [8]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state
= 25)
```

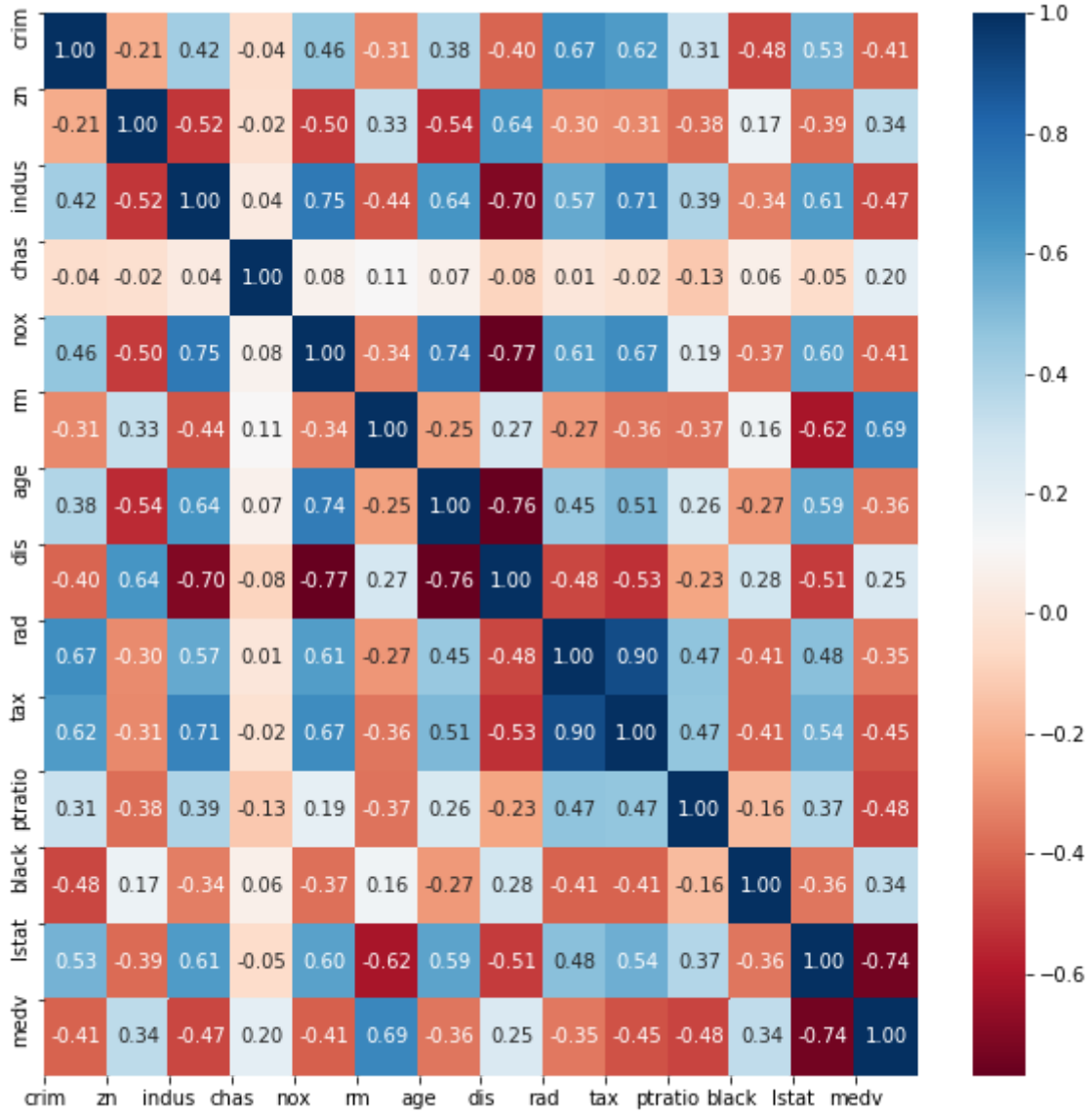
In [9]:

```
print("Shape of X_train: ",X_train.shape)
print("Shape of X_test: ", X_test.shape)
print("Shape of y_train: ",y_train.shape)
print("Shape of y_test",y_test.shape)
```

```
Shape of X_train:  (233, 13)
Shape of X_test:  (100, 13)
Shape of y_train:  (233, 1)
Shape of y_test (100, 1)
```

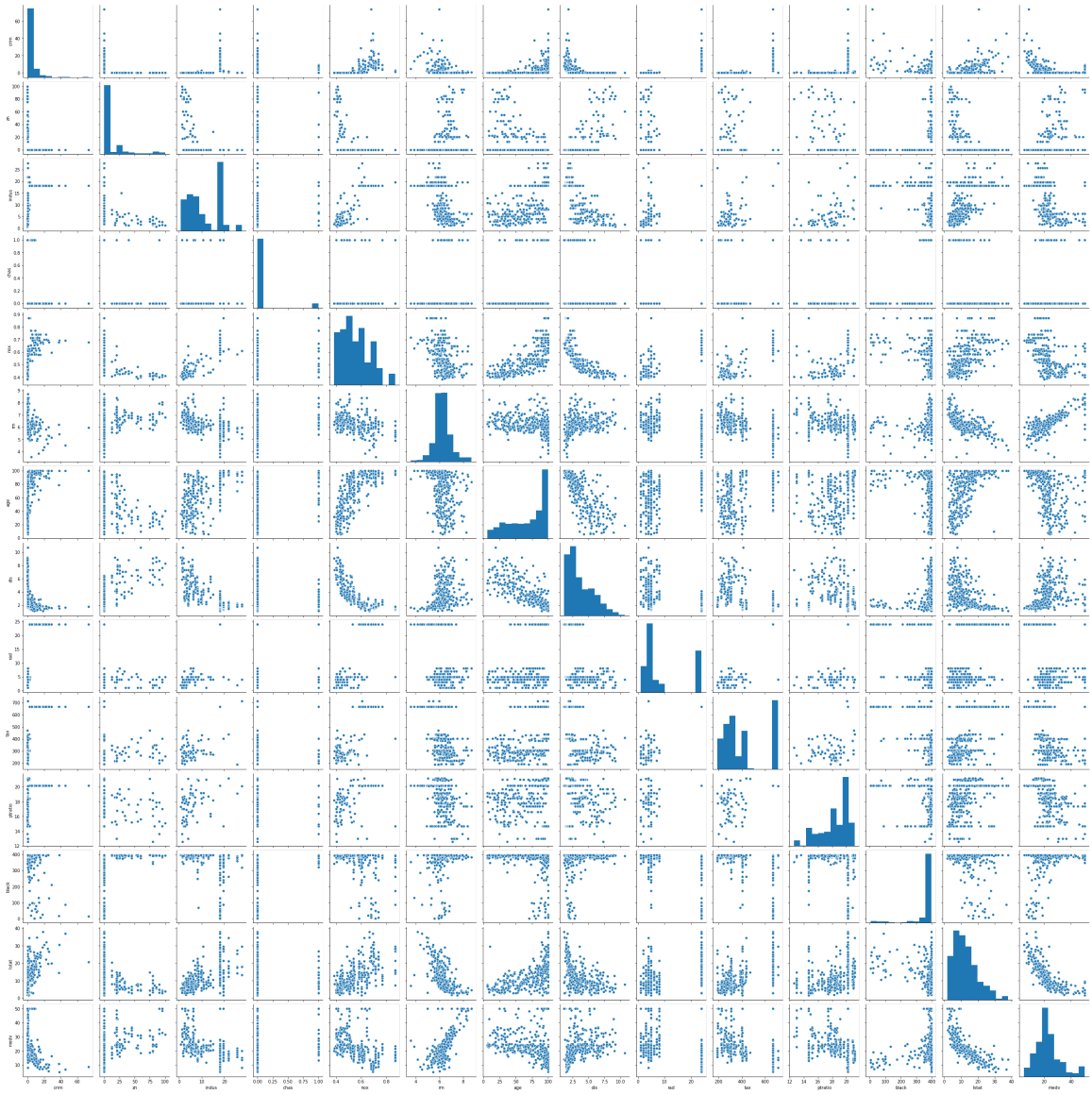
In [10]:

```
#Visualisation of data
corr = dataset.corr()
#Plot figsize
fig, ax = plt.subplots(figsize=(10, 10))
#Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr, cmap='RdBu', annot=True, fmt=".2f")
#Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
#Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
#show plot
plt.show()
```


crim 1.00 -0.21 0.42 -0.04 0.46 -0.31 0.38 -0.40 0.67 0.62 0.31 -0.48 0.53 -0.41
zn -0.21 1.00 -0.52 -0.02 -0.50 0.33 -0.54 0.64 -0.30 -0.31 -0.38 0.17 -0.39 0.34
indus 0.42 -0.52 1.00 0.04 0.75 -0.44 0.64 -0.70 0.57 0.71 0.39 -0.34 0.61 -0.47
chas -0.04 -0.02 0.04 1.00 0.08 0.11 0.07 -0.08 0.01 -0.02 -0.13 0.06 -0.05 0.20
nox 0.46 -0.50 0.75 0.08 1.00 -0.34 0.74 -0.77 0.61 0.67 0.19 -0.37 0.60 -0.41
rm -0.31 0.33 -0.44 0.11 -0.34 1.00 -0.25 0.27 -0.27 -0.36 -0.37 0.16 -0.62 0.69
age 0.38 -0.54 0.64 0.07 0.74 -0.25 1.00 -0.76 0.45 0.51 0.26 -0.27 0.59 -0.36
dis -0.40 0.64 -0.70 -0.08 -0.77 0.27 -0.76 1.00 -0.48 -0.53 -0.23 0.28 -0.51 0.25
rad 0.67 -0.30 0.57 0.01 0.61 -0.27 0.45 -0.48 1.00 0.90 0.47 -0.41 0.48 -0.35
tax 0.62 -0.31 0.71 -0.02 0.67 -0.36 0.51 -0.53 0.90 1.00 0.47 -0.41 0.54 -0.45
ptratio 0.31 -0.38 0.39 -0.13 0.19 -0.37 0.26 -0.23 0.47 0.47 1.00 -0.16 0.37 -0.48
black -0.48 0.17 -0.34 0.06 -0.37 0.16 -0.27 0.28 -0.41 -0.41 -0.16 1.00 -0.36 0.34
lstat 0.53 -0.39 0.61 -0.05 0.60 -0.62 0.59 -0.51 0.48 0.54 0.37 -0.36 1.00 -0.74
medv -0.41 0.34 -0.47 0.20 -0.41 0.69 -0.36 0.25 -0.35 -0.45 -0.48 0.34 -0.74 1.00
crim zn indus chas nox rm age dis rad tax ptratio black lstat medv
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6

In [11]:

```
sns.pairplot(dataset)
plt.show()
```



In [12]:

```
#Linear Regression

from sklearn.linear_model import LinearRegression
regressor_linear = LinearRegression()
regressor_linear.fit(X_train, y_train)
```

Out[12]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [13]:

```
from sklearn.metrics import r2_score

# Predicting Cross Validation Score the Test set results
cv_linear = cross_val_score(estimator = regressor_linear, X = X_train, y = y_train, cv
= 10)

# Predicting R2 Score the Train set results
y_pred_linear_train = regressor_linear.predict(X_train)
r2_score_linear_train = r2_score(y_train, y_pred_linear_train)

# Predicting R2 Score the Test set results
y_pred_linear_test = regressor_linear.predict(X_test)
r2_score_linear_test = r2_score(y_test, y_pred_linear_test)

# Predicting RMSE the Test set results
rmse_linear = (np.sqrt(mean_squared_error(y_test, y_pred_linear_test)))
print("CV: ", cv_linear.mean())
print('R2_score (train): ', r2_score_linear_train)
print('R2_score (test): ', r2_score_linear_test)
print("RMSE: ", rmse_linear)
```

```
CV:  0.6980975810514456
R2_score (train):  0.7579501374166677
R2_score (test):  0.6561144544831812
RMSE:  6.0562589191075915
```

In [ ]:

In [ ]:

In [ ]: