In [ ]:
```
Name: ISHAN CHASKAR
URK21CS1181
```

In [ ]:
```
AIM:
    To perform performance analysis on regression techniques.

DESCRIPTION:
    Machine learning has two types Supervised and Unsupervised learning. Supervi
    Regression: It predicts the continuous output variables based on the indeper
    Linear regression is a type of supervised machine learning algorithm that cc
    Linear regression is one of the most basic types of regression in machine le
    Simple Linear Regression is a type of Regression algorithms that models the
    The key point in Simple Linear Regression is that the dependent variable mus
    Simple Linear regression algorithm has mainly two objectives:
        Model the relationship between the two variables. Such as the relationsh
```

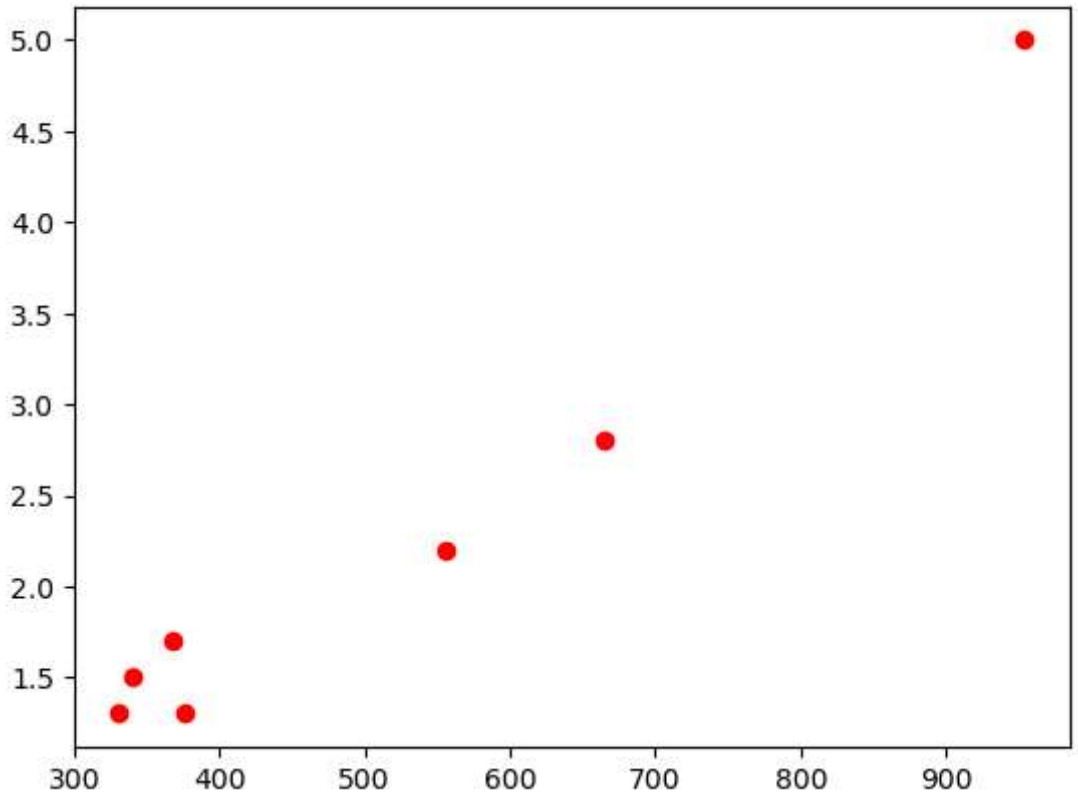
In [ ]:
```
Q1:
    Develop the linear regression model for the given data by step-by-step manua
    a. Calculate the intercept and regression coefficients in y=b0+xb1
    b. Analyse the various performance metrics (Mean Squared Error, Mean Absolut
```

In [22]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [23]:
```
print('URK21CS1181')
x=[368,340,665,954,331,556,376]
y=[1.7,1.5,2.8,5,1.3,2.2,1.3]
plt.scatter(x,y,color="r",marker="o")
```

```
URK21CS1181
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x7f3b80391130>
```



In [ ]:
```
x=np.array(x)
y=np.array(y)
m_x=np.mean(x)
m_y=np.mean(y)

xx=x-m_x
yy=y-m_y
xy=xx*yy
xx2=xx*xx

SS_xy=sum(xy)
SS_xx=sum(xx2)
```

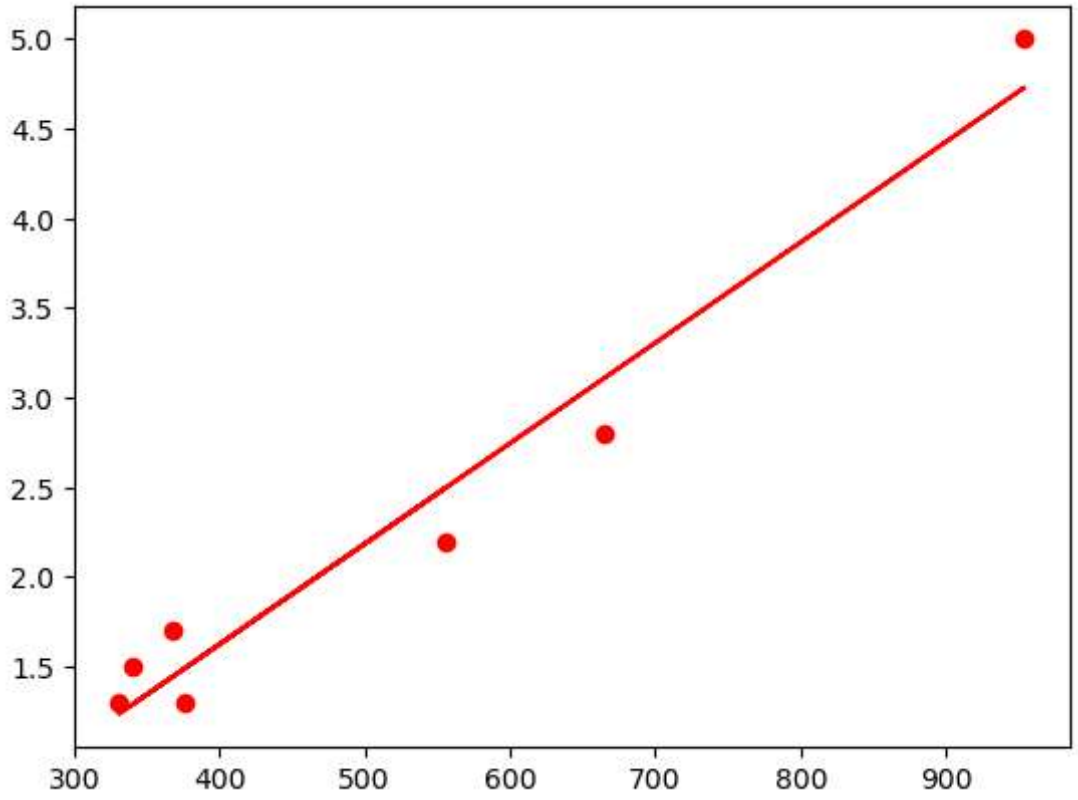
In [ ]:
```
a) Calculating Coefficients
```

In [25]:
```
print('URK21CS1181')
b1=SS_xy/SS_xx
b0=m_y-(b1*m_x)
print("Slope ",b1)
print("Intercept ",b0)
```

```
URK21CS1181
Slope  0.005606157184993036
Intercept  -0.6180148991607144
```

In [26]:
```
print('URK21CS1181')
y_pred=b0+b1*x
plt.scatter(x,y,color="r",marker="o")
plt.plot(x,y_pred,color="r")
plt.show()
```

```
URK21CS1181
```



In [ ]:
```
b) Analyse the various performance metrics (Mean Squared Error, Mean Absolute Er
```

In [27]:
```
print('URK21CS1181')
err=y-y_pred
print(err)
```

```
URK21CS1181
[ 0.25494906  0.21192146 -0.31007963  0.26974094  0.06237687 -0.2990085
 -0.1899002 ]
```

In [28]:
```
print('URK21CS1181')
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
import math
print("MAE:",mean_absolute_error(y,y_pred))
print("MSE:",mean_squared_error(y,y_pred))
print("Root Mean Squared:",math.sqrt(mean_squared_error(y,y_pred)))
print("r2 score:",r2_score(y,y_pred))
```

```
URK21CS1181
MAE: 0.22828237912906743
MSE: 0.058311188453828024
Root Mean Squared: 0.24147709716208704
r2 score: 0.9612629035488398
```

In [ ]:
```
Q2:
    Develop the linear regression model for the prediction of median value of ow
homes (medv) in Boston in terms of weighted distances to five Boston employment
(dis), property-tax rate (tax), nitric oxides concentration (nox) using the scik
a. Divide the data into training (75%) and testing set (25%)
b. Display the intercept and regression coefficients for the following cases
1. Analyse the impact of dis to medv
2. Analyse the impact of tax to medv
3. Analyse the impact of nox to medv
c. Predict the y value (y') for the testing set (x) and analyse the performance
the actual value (y) and predicted values (y') for the above 3 cases
d. Identify the input parameter that has a greater impact in the prediction of p
median value of owner-occupied homes in Boston
e. Plot the regression line for the input parameter that has a greater impact ir
prediction of prediction of median value of owner-occupied homes in Boston
```

In [29]:
```
print('URK21CS1181')
import pandas as pd
df=pd.read_csv('Boston.csv',index_col=0)
df.head()
```

```
URK21CS1181
```

Out[29]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	med
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.
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.
3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.
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.
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.

In [30]:
```
print('URK21CS1181')
x = df[['dis', 'tax', 'nox']]
```

```
y = df['medv']
print(x.shape)
```

```
URK21CS1181
(506, 3)
```

In [31]:
```
print('URK21CS1181')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random
print(X_train.shape)
print(X_test.shape)
X_train['dis']
```

```
URK21CS1181
(379, 3)
(127, 3)
```

Out[31]:
```
503    2.2875
173    2.5961
81     5.4007
47     5.1004
319    3.5325
       ...
256    9.2203
73     5.2873
397    1.6768
236    3.6519
38     3.9342
Name: dis, Length: 379, dtype: float64
```

In [32]:
```
print('URK21CS1181')
from sklearn.linear_model import LinearRegression
import numpy as np
model_dis = LinearRegression()
model_dis.fit(X_train[['dis']], y_train)
print(model_dis.coef_)
print(model_dis.intercept_)
```

```
URK21CS1181
[1.2142559]
17.700750733178133
```

In [33]:
```
print('URK21CS1181')
model_tax = LinearRegression()
model_tax.fit(X_train[['tax']], y_train)
print(model_tax.intercept_)
print(model_tax.coef_)
```

```
URK21CS1181
31.92932918239751
[-0.02365539]
```

In [ ]:
```
print('URK21CS1181')
model_nox = LinearRegression()
model_nox.fit(X_train[['nox']], y_train)
print(model_nox.intercept_)
print(model_nox.coef_)
```

```
URK21CS1181
41.56399295883796
[-34.68750158]
```

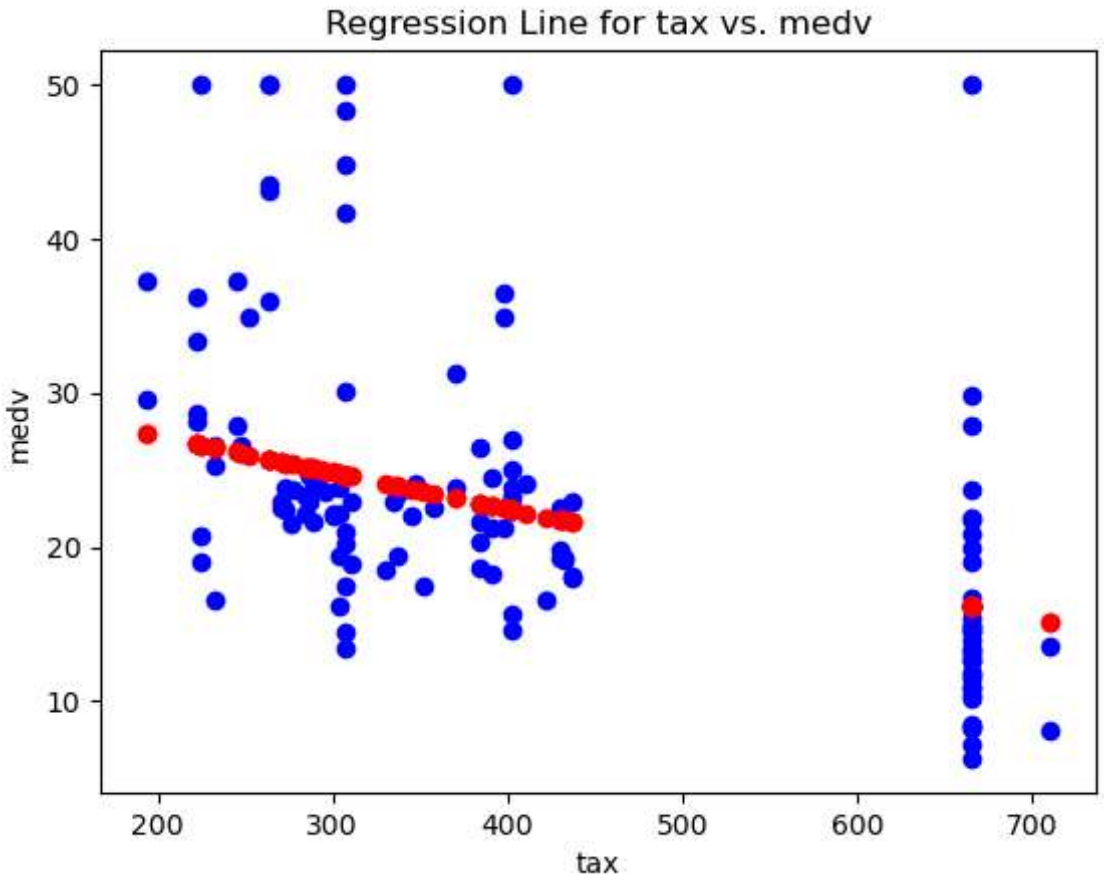
In [34]:
```
print('URK21CS1181')
y_pred_dis = model_dis.predict(X_test[['dis']])
mse_dis = mean_squared_error(y_test, y_pred_dis)
r2_dis = r2_score(y_test, y_pred_dis)
print("Mean Squared Error:", mse_dis)
print("R-squared:", r2_dis)

y_pred_tax = model_tax.predict(X_test[['tax']])
mse_tax = mean_squared_error(y_test, y_pred_tax)
r2_tax = r2_score(y_test, y_pred_tax)
print("Mean Squared Error:", mse_tax)
print("R-squared:", r2_tax)

y_pred_nox = model_nox.predict(X_test[['nox']])
mse_nox = mean_squared_error(y_test, y_pred_nox)
r2_nox = r2_score(y_test, y_pred_nox)
print("Mean Squared Error:", mse_nox)
print("R-squared:", r2_nox)
```

```
URK21CS1181
Mean Squared Error: 99.00289684761006
R-squared: 0.0005610495229881884
Mean Squared Error: 74.16735303693665
R-squared: 0.2512770450243307
Mean Squared Error: 87.19275626434953
R-squared: 0.11978498018926331
```

In [ ]:
```
As tax has the maximun R-squared value, we can say the input parameter that has
```

In [ ]:
```
print('URK21CS1181')
plt.scatter(X_test[['tax']], y_test, color='blue',marker="o")
plt.scatter(X_test[['tax']], y_pred_tax, color='red')
plt.title('Regression Line for tax vs. medv')
plt.xlabel('tax')
plt.ylabel('medv')
plt.show()
```

```
URK21CS1181
```



In [ ]:
```
Result:
    Hence the python code to perform performance analysis of regression techniqu
```