In [9]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
import math
import random
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler,MinMaxScaler
```

In [10]:

```
location="linear-regression.csv"
df = pd.read_csv('linear-regression.csv')
df.insert(0, 'bias', '1')
```

In [11]:

```
df.head()
```

Out[11]:

	bias	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	1	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9
1	1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9
2	1	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9
3	1	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9
4	1	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9

In [12]:

```
data=np.array(df,dtype=np.float64)
```

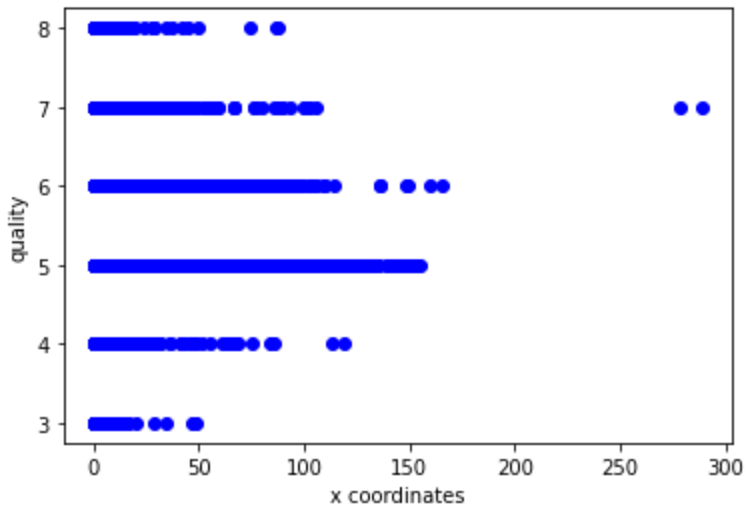
In [13]:

```
X=data[:,:12]
Y=data[:,-1]
print(X.shape)
print(Y.shape)
```

```
(1599, 12)
(1599,)
```

In [14]:

```
plt.xlabel('x coordinates')
plt.ylabel('quality')
plt.plot(X[:,1:10], Y, 'bo')
plt.show()
```



In [15]:
```
X_train, X_temp, y_train, y_temp = train_test_split(X, Y, test_size=0.5, random_state
```

In [16]:
```
X_validation, X_test, y_validation, y_test = train_test_split(X_temp, y_temp, test_s:
```

In [17]:
```
print(len(X_train))
print(len(X_validation))
print(len(X_test))
```

```
799
480
320
```

In [18]:
```
def h(x,theta):
    return np.round(np.matmul(x, theta))
    '''return ans'''
```

In [19]:
```
def cost_function(x, y, theta):
    J=((h(x,theta-y)).T*(h(x,theta-y)))/(2*len(y))
    return J
```

In [20]:
```
def accuracy(x,y,theta):
    l=len(x)
    correct=0
    for i in range(0,l):
        y_predicted=round(h(x[i],theta))
        if(y_predicted==y[i]):
            correct+=1
    percentage=correct/l
    return percentage
```

In [21]:

```
def R_Square(x,y,theta):
    y_mean=round(np.mean(y))
    l=len(y)
    numerator=0
    denominator=0
    for i in range(l):
        numerator=numerator+(y[i]-(h(x[i],theta)))**2
        denominator=denominator+(y[i]-y_mean)**2
    value = numerator/denominator
    return 1-value
```

In [22]:

```
def RMSE(x,y,theta):
    l=len(y)
    total=0
    for i in range(l):
        y_predicted=round(h(x[i],theta))
        total=total+(y[i]-y_predicted)**2
    total=total/l
    total=math.sqrt(total)
    return total
```

Analytical Solution

In [23]:

```
theta = np.linalg.inv(X_train.T.dot(X_train)).dot(X_train.T).dot(y_train)
print(theta)
```

```
[ 2.33689602e+01  3.05502970e-02 -1.19741412e+00 -1.87228641e-01
  1.17612142e-02 -1.81714527e+00  6.11854741e-03 -4.06437389e-03
 -1.90187880e+01 -4.42172779e-01  8.79804133e-01  2.64603431e-01]
```

In [24]:

```
#validation set R_Square
print("R_Square score on validation set "+ str(R_Square(X_validation,y_validation,the

#test set R_Square
R_Square(X_test,y_test,theta)
print("R_Square score on test set "+ str(R_Square(X_test,y_test,theta)))
```

```
R_Square score on validation set 0.3252688172043011
R_Square score on test set 0.4125
```

In [25]:

```
#validation set RMSE

print("RMSE score on validation set "+ str(RMSE(X_validation,y_validation,theta)))
#test set RRMSE

print("RMSE score on test set "+ str(RMSE(X_test,y_test,theta)))
```

```
RMSE score on validation set 0.7231297716638879
RMSE score on test set 0.6637959023675877
```

With Regularization

In [26]:
```
Identity_matrix = np.identity(len(X_train.T.dot(X_train)), dtype = float)
lamda=2
```

In [27]:
```
theta_regularised = np.linalg.inv((X_train.T.dot(X_train))+(lamda*Identity_matrix)).c
```

In [28]:
```
#validation set R_square
print("R_square score on validation set "+str(R_Square(X_validation,y_validation,thet

#test set R_square
print("R_square score on test set "+str(R_Square(X_test,y_test,theta_regularised)))
```

```
R_square score on validation set 0.31720430107526887
R_square score on test set 0.4041666666666667
```

In [29]:
```
#validation set RMSE
print("RMSE score in validation set "+str(RMSE(X_validation,y_validation,theta_regula

#test set RMSE
print("RMSE score in test set "+str(RMSE(X_test,y_test,theta_regularised)))
```

```
RMSE score in validation set 0.7274384280931732
RMSE score in test set 0.6684870978560469
```

Interative Gradient Ascent

In [30]:
```
data=np.array(df,dtype=np.float64)
scaler=StandardScaler()
#data=scaler.fit_transform(data)
X=data[:,:12]
X=scaler.fit_transform(X)
Y=data[:,-1]
X[:,0]+=1
X_train, X_temp, y_train, y_temp = train_test_split(X, Y, test_size=0.5, random_state
X_validation, X_test, y_validation, y_test = train_test_split(X_temp, y_temp, test_s
#scaler=StandardScaler()
#X_train=scaler.fit_transform(X_train)
```

In [31]:
```
def predict_y(x,theta):
    t=np.dot(x,theta)
    #t=np.round(t)
    return t
```

In [32]:

```
def total_cost(y,x,theta):
    l=len(y)
    total=0
    for i in range(l):
        y_p=predict_y(x[i],theta)
        total+=((y[i])-y_p)**2
    return total/(2*l)
```

In [33]:

```
def gradient_ascent(x,y,theta,learning_rate,epoch):
    l=len(x)
    cost=[]
    cost_v=[]
    y_ep=[]
    for k in range(epoch):
        for i in range(0,l):
            y_p=(np.dot(x[i],theta))
            theta=theta+learning_rate*((y[i]-y_p)*x[i])
        cost.append(total_cost(y,x,theta))
        cost_v.append(total_cost(y_validation,X_validation,theta))
        y_ep.append(k+1)
    plt.xlabel('Epochs')
    plt.ylabel('Cost')
    plt.plot(y_ep, cost, 'm', linewidth = "5")
    plt.plot(y_ep, cost_v, 'g', linewidth = "5")
    plt.show()


    return theta
```

learning rate 0.01

In [34]:

```
theta=np.ones(X_train.shape[1])
print(theta)
theta=gradient_ascent(X_train,y_train,theta,0.01,30)
#theta=gradient_ascent(X_validation,y_validation,theta,0.001,8)
print(theta)
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

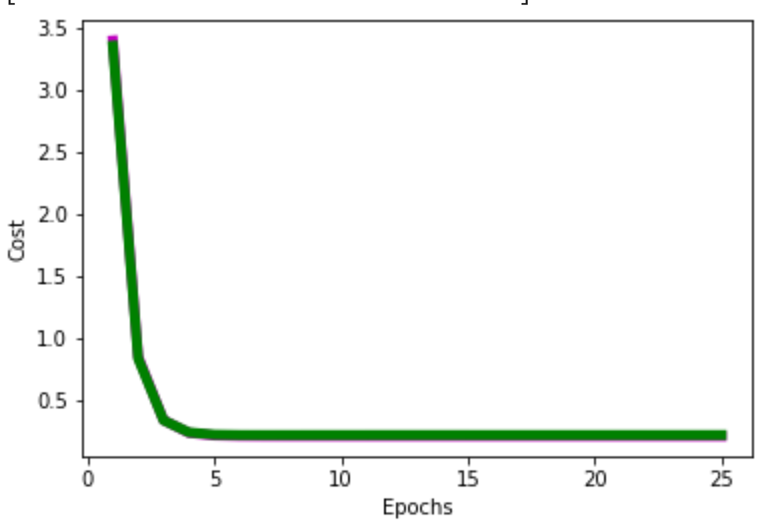
In [35]:
```
#validation set R_square
print("R_square score on validation set "+str(R_Square(X_validation,y_validation,thet

#test set R_square
print("R_square score on test set "+str(R_Square(X_test,y_test,theta)))

#validation set RMSE
print("RMSE score in validation set "+str(RMSE(X_validation,y_validation,theta)))

#test set RMSE
print("RMSE score in test set "+str(RMSE(X_test,y_test,theta)))
```

```
R_square score on validation set 0.303763440860215
R_square score on test set 0.3666666666666667
RMSE score in validation set 0.734563362367967
RMSE score in test set 0.689202437604511
```

learning rate 0.001

In [36]:
```
theta=np.zeros(X_train.shape[1])
print(theta)
theta=gradient_ascent(X_train,y_train,theta,0.001,25)
#theta=gradient_ascent(X_validation,y_validation,theta,0.001,8)
print(theta)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```




```
[ 5.63609973  0.08777612 -0.22136512 -0.03556634  0.01875345 -0.08741357
  0.05674787 -0.13166918 -0.06067875 -0.06059899  0.15495999  0.2621672 ]
```

In [37]:

```
#validation set R_square
print("R_square score on validation set "+str(R_Square(X_validation,y_validation,thet

#test set R_square
print("R_square score on test set "+str(R_Square(X_test,y_test,theta)))

#validation set RMSE
print("RMSE score in validation set "+str(RMSE(X_validation,y_validation,theta)))

#test set RMSE
print("RMSE score in test set "+str(RMSE(X_test,y_test,theta)))
```

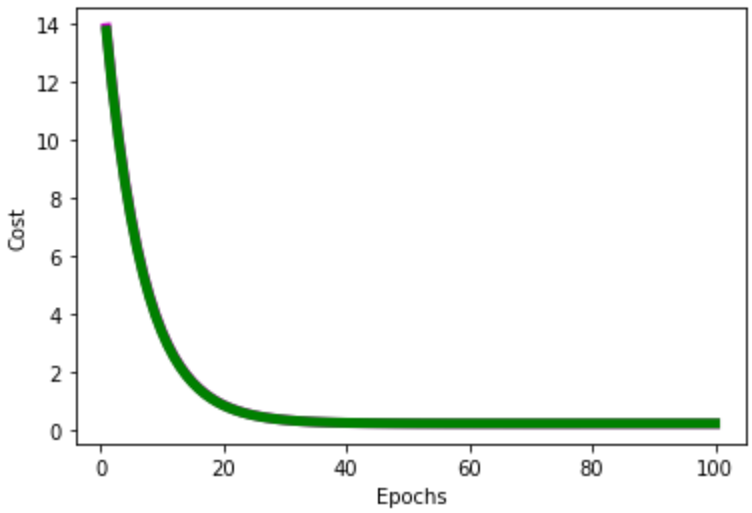
```
R_square score on validation set 0.3360215053763441
R_square score on test set 0.4083333333333333
RMSE score in validation set 0.7173446405552447
RMSE score in test set 0.6661456297237114
```

learning rate 0.0001

In [38]:

```
theta=np.zeros(X_train.shape[1])
print(theta)
theta=gradient_ascent(X_train,y_train,theta,0.0001,100)
#theta=gradient_ascent(X_validation,y_validation,theta,0.001,8)
print(theta)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```




```
[ 5.6322383   0.10694348 -0.20932108 -0.03079476  0.02918622 -0.08333899
  0.05566896 -0.12365031 -0.0847903  -0.0332272   0.15562779  0.25233705]
```

In [39]:

```
#validation set R_square
print("R_square score on validation set "+str(R_Square(X_validation,y_validation,thet

#test set R_square
print("R_square score on test set "+str(R_Square(X_test,y_test,theta)))

#validation set RMSE
print("RMSE score in validation set "+str(RMSE(X_validation,y_validation,theta)))

#test set RMSE
print("RMSE score in test set "+str(RMSE(X_test,y_test,theta)))
```

```
R_square score on validation set 0.3198924731182796
R_square score on test set 0.4041666666666667
RMSE score in validation set 0.7260050504874834
RMSE score in test set 0.6684870978560469
```

In [ ]:

In [ ]: