Practical-9

Objective:

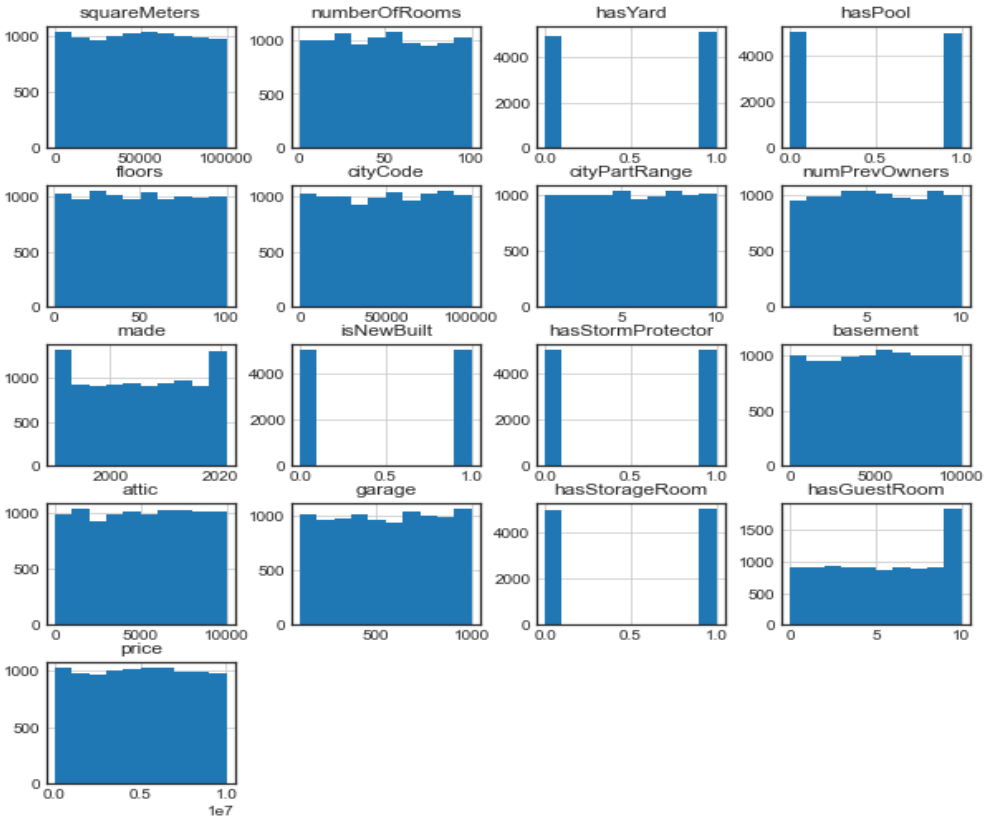
_____Design a classifier with regularization technique used in it.

Description:

Given a data set with multiple features (can have multiple values) and a class(continuous values), Design a linear regression classifier with gradient descent along with regularization technique used in it which will give out target value x for unknown examples.

Dataset:

1. Name: Paris Housing Price Prediction
2. Source: https://www.kaggle.com/mssmartypants/paris-housing-priceprediction
3. This is a set of data created from imaginary data of house prices in an urban environment – Paris
4. There are 16 variables(house features) and 1 class(price).
5. There are 10000 samples.



6. Correlation with price:

squareMeters	0.999999
numberOfRooms	0.009591
hasYard	-0.006119
hasPool	-0.005070
floors	0.001654
cityCode	-0.001539
cityPartRange	0.008813
numPrevOwners	0.016619
made	-0.007210
isNewBuilt	-0.010643
hasStormProtector	0.007496
Basement	-0.003967
attic	-0.000600
Garage	-0.017229
hasStorageRoom	-0.003485
hasGuestRoom	-0.000644
price	1.000000

Algorithm:

1. Read data from csv using pandas
2. Covert dataset into list of x(data) and y(target)
3. Normalize x by using (0,1) normalization
4. Split the data into training and test data
5. Linear Regression with regularization:
    - Insert 1 as first data to all the data since the value of x0 is 1.(theta0 is intercept)
    - Initialize theta vector(size = number of features + 1) to zero
    - Calculate y_cap(ie: predictions) by performing dot product of x and theta.
    - Calculate error by subtracting y_cap from y
    - Calculate the cost function by using below formula:
      cost= 1/2m *(error)^2
      m= no of samples
    - Update the value of theta by using following formula:
      θ = θ-(η/m) * (xt . error)
      θ = θ -(η/m) * (xt . error) + λ/m * θ

- Repeat (c-f) for specified amount of iterations.

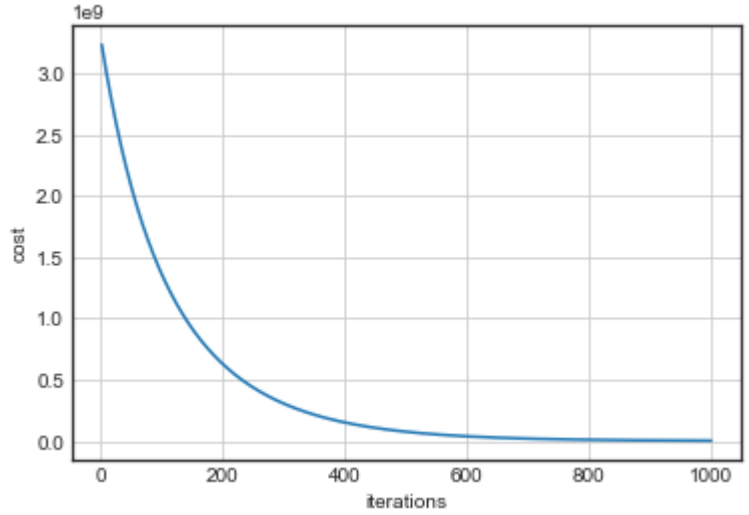
Implementation:

```
# 1 read data
data = pd.read_csv('./data/ParisHousing.csv')

# 2 split data into x and y
x = np.array(data.iloc[:,:-1].values)

# 5a append 1 for x0 value
x = np.hstack((np.ones((data.shape[0],1)), x))
y = np.array(data['price'])

# 3 Normalize the data
def normalize(x):
 return x/np.max(x, axis=0),np.max(x, axis=0)
x,temp_max = normalize(x)

# 4 split data into test and training
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30)
n=len(x_train[0])-1
m=len(x_train)

# 5 Linear Regression with regularization

def gd_r(x, y, theta, lr, itr , lambda_value):
 cost_plot = np.zeros(itr)
 theta_0 = theta[0]
 theta_1_n = theta[1:]
 for i in range(itr):
 y_cap = x.dot(theta)
 e = y_cap - y
 cost = 1 / (2 * m) * np.sum(e*e)
 cost_plot[i] = cost
 updated_theta_0 = (lr / m) * (x[:,0].dot(e))
 updated_theta_1_n = (lr / m) * ( (((x[:,1:]).transpose()).dot(e)) +
((lambda_value/m) * theta_1_n) )
 theta = theta - np.hstack((updated_theta_0, updated_theta_1_n))
 plt.plot(range(1, itr + 1), cost_plot)
 plt.grid()
 plt.xlabel("iterations")
 plt.ylabel("cost")
 return theta

theta = np.zeros(n+1)
itr = 500;
```

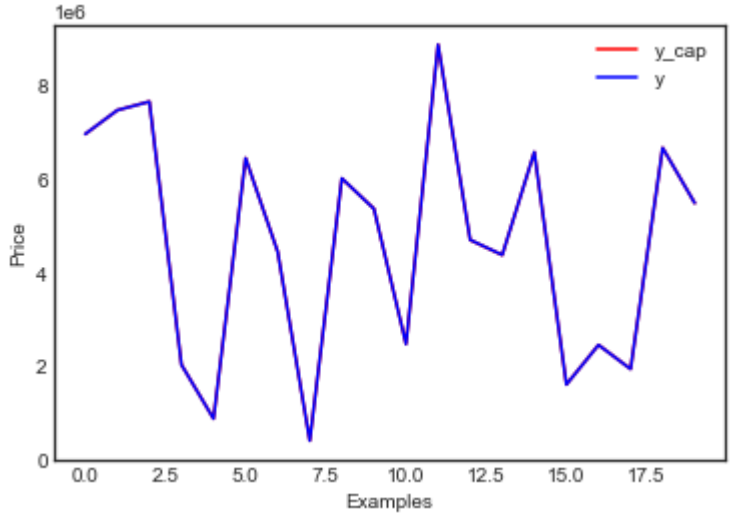
```
theta = gd_r(x_train, y_train, theta, 0.1, 1000,0.2)
y_pred = list()
for i in range(len(x_test)):
 y_pred.append(theta.dot(x_test[i]))
```

Output:

GD graph:



Prediction vs actual graph:



Theta values:

array([6.89661881e+03, 9.99746216e+06, -2.04197309e+03, 2.50151252e+03,
2.50199203e+03, 3.25549527e+03, -2.27395285e+03, -2.05252177e+03,
-2.31281380e+03, 6.16421515e+03, -3.49527348e+02, -3.52515333e+02,
-2.61626756e+03, -3.39951180e+03, -3.12073992e+02, -2.03773057e+03])