Machine Learning Fundamentals – Lab-4

Name: Gaurav Prasanna
Reg No: 19BEC1315

Aim:

a) To show Gradient Descent and achieve the Global minimum from scratch.
b) To implement and visualize Logistic Regression on digits dataset using sci-kit learn library.
c) To implement and visualize Logistic Regression on given dataset from scratch without any inbuilt libraries or functions.

Software Required:

a) Anaconda Navigator
b) Jupyter Notebook

Libraries Required: Scikit-Learn, Pandas, Numpy, Matplotlib

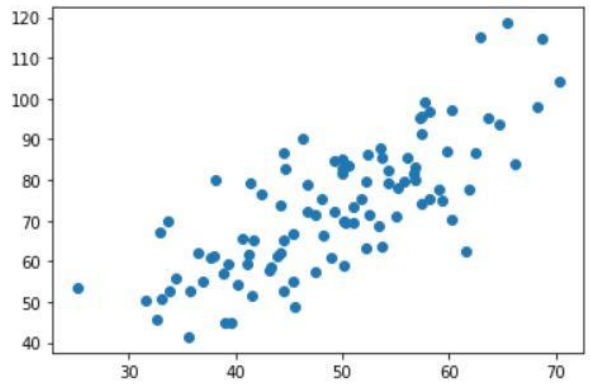
Codes and Outputs:

a) Gradient Descent:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# plt.rcParams['figure.figsize'] = (12.0, 9.0)
```

In [2]:
```
data = pd.read_csv('data.csv')
X = data.iloc[:, 0]
y = data.iloc[:, 1]
plt.scatter(X,y)
```

Out[2]: <matplotlib.collections.PathCollection at 0x23fc707fe80>

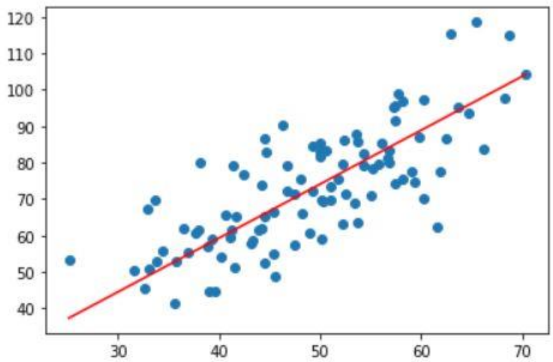


In [12]:
```
m = 0
c = 0

L = 0.0001
epochs = 2000

n = float(len(X))
for i in range(epochs):
    y_pred = m*X + c
    D_m = (-2/n) * sum(X*(y - y_pred))
    D_c = (-2/n) * sum(y - y_pred)
    m = m - L * D_m
    c = c - L * D_c
    print(m, c, D_m)
```

```
0.7424335285415493 0.014629895049539552 -7424.335285415493
1.1126970531564835 0.02196251949501772 -3702.63524614934
1.2973530613136097 0.02565587059950957 -1846.5600815712621
1.3894434413943608 0.027534253868739502 -920.9038008075106
1.4353697670003447 0.028507481513835913 -459.26325605983754
1.4582732927875135 0.02902929237235795 -229.03525787168834
1.4696949563897819 0.029325973726661497 -114.2166360226839
1.4753904198682661 0.02951037889658117 -56.95463478484235
1.478230129872329 0.029638789753839184 -28.397100040628544
1.4796456270100444 0.02973927502595998 -14.154971377152952
1.4803508449519853 0.029825833065180055 -7.052179419409328
1.4807018345730651 0.029905445088057748 -3.509896210797972
1.4808761645057094 0.02998159274613495 -1.7432993264412309
1.4809623912496854 0.030056012398306242 -0.862267439761371
1.4810046794389382 0.030129569996312625 -0.42288189252731817
1.481025054728236 4 0.030202697405030527 -0.2037528929823426
1.4810345016865596 0.030275610002827504 -0.09446958323212011
1.4810384985054497 0.030348415202563837 -0.0399681888998099
1.4810397772522843 0.030421166572863437 -0.012787468346669868
```

In [4]:
```
y_pred = m*X + c

plt.scatter(X, y)
plt.plot([min(X), max(X)], [min(y_pred), max(y_pred)], color = 'red')
```

Out[4]: [<matplotlib.lines.Line2D at 0x23fc91a7910>]



b) Logistic Regression Using Scikit-Learn

In [1]:
```
import sklearn
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

In [4]:
```
digits = load_digits()
X = digits.data
y = digits.target
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
```

In [6]:
```
reg = LogisticRegression()
```

In [7]:
```
reg.fit(X_train, y_train)
```

```
C:\Users\Gaurav Prasanna\miniconda3\envs\tensorflow\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarnin
g: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[7]: LogisticRegression()

In [8]:
```
y_pred = reg.predict(X_test)
```

In [9]:
```
print('Accuracy score of model is {}'.format(accuracy_score(y_test,y_pred)*100))
```

```
Accuracy score of model is 97.22222222222221
```

c) Logistic Regression from Scratch

In [1]:
```
import numpy as np
import csv
import matplotlib.pyplot as plt
```

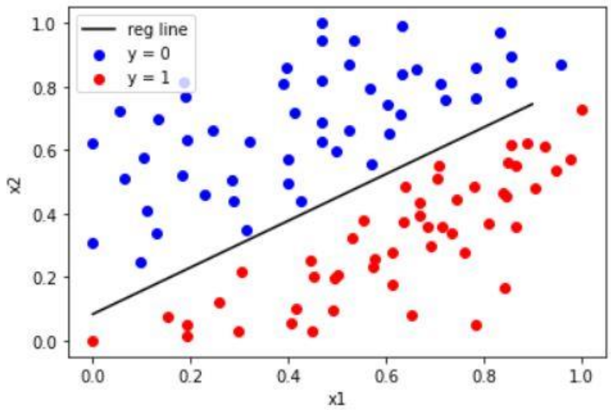
In [7]:
```
def loadCSV(filename):

    with open(filename,"r") as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for i in range(len(dataset)):
            dataset[i] = [float(x) for x in dataset[i]]
    return np.array(dataset)

def normalize(X):

    mins = np.min(X, axis = 0)
    maxs = np.max(X, axis = 0)
    rng = maxs - mins
    norm_X = 1 - ((maxs - X) / rng)
    return norm_X

def logistic_func(beta, X):

    return 1.0/(1+ np.exp(-np.dot(X, beta.T)))

def log_gradient(beta, X, y):

    first_calc = logistic_func(beta, X) - y.reshape(X.shape[0], -1)
    final_calc = np.dot(first_calc.T, X)
    return final_calc
```

In [10]:
```
def cost_func(beta, X, y):

    log_func_v = logistic_func(beta, X)
    y = np.squeeze(y)
    step1 = y * np.log(log_func_v)
    step2 = (1 - y)* np.log(1 - log_func_v)
    final = -step1 - step2
    return np.mean(final)

def grad_desc(X, y, beta, lr=0.01, converge_change=0.001):

    cost = cost_func(beta, X, y)
    change_cost = 1
    num_iter = 1

    while(change_cost > converge_change):
        old_cost = cost
        beta = beta - (lr * log_gradient(beta, X, y))
        cost = cost_func(beta, X, y)
        change_cost = old_cost - cost
        num_iter += 1
    return beta, num_iter
```

In [8]:
```python
def pred_values(beta, X):

    pred_prob = logistic_func(beta, X)
    pred_value = np.where(pred_prob >= 0.5, 1, 0)
    return np.squeeze(pred_value)

def plot_reg(X, y, beta):

    x_0 = X[np.where(y == 0.0)]
    x_1 = X[np.where(y == 1.0)]

    import matplotlib.pyplot as plt

    plt.scatter(x_0[:, 0], x_0[:, 1], c='b', label='y = 0')
    plt.scatter(x_1[:, 0], x_1[:, 1], c='r', label='y = 1')

    x1 = np.arange(0, 1, 0.1)
    x2 = -(beta[0,0] + beta[0,1]*x1)/beta[0,2]
    plt.plot(x1,x2, c='k', label='reg line')
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.legend()
    plt.show()
```

In [9]:
```python
if __name__ == "__main__":
    dataset = loadCSV('dataset1.csv')

    X = normalize(dataset[:, :-1])
    X2 = X

    X1 = np.hstack((np.matrix(np.ones(X.shape[0])).T, X))

    y = dataset[:, -1]

    beta = np.matrix(np.zeros(X1.shape[1]))

    beta1, num_iter = grad_desc(X1, y, beta)

    print("Estimated regression coefficients:", beta1)
    print("No of iterations:", num_iter)

    y_pred = pred_values(beta1, X1)

    print('Correctly predicted labels:', np.sum(y == y_pred))

    plot_reg(X2, y, beta1)
```

```
Estimated regression coefficients: [[  1.70474504  15.04062212 -20.47216021]]
No of iterations: 2612
Correctly predicted labels: 100
```



Inference: So, from the first part we understand gradient descent and how it approaches the global minimum and its variations in changing the learning rate or tweaking it and also varying the number of epochs. In the second part we import the digits dataset from scikit-learn library and using the inbuilt package of Logistic Regression is performed using the same and the accuracy score is calculated. And from the final part we implement the logistic regression from scratch by writing functions for each operation and to compute the regression coefficients and to finally visualize it using Matplotlib.

Result: Gradient Descent, Logistic Regression using Scikit-Learn and also implementing from scratch is shown and visualized using in Jupyter Notebook.