University of Haripur.

Assignment #2.

Machine Learning

Title: Linear Regression.

To: Ms. Nadia

From: Hamza Sadaat.

Depart: Information Technology.

Program: Computer Science.

Roll No: F18-0501.

Semester: 7th 'B'.

Dated: 06/12/2021.

Importing Libraries and reading dataset

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 7 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [218]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [219]:
```
data_df = pd.read_csv(r"C:\Users\DarkReaper\Desktop\linear.csv", header=None, names=['X','y'])
data_df.head()
```

Out[219]:

	X	y
0	1	1
1	2	3
2	3	3
3	4	2
4	5	5

In [220]:
```
n_rows = data_df.shape[0]
```

Creating function to calculate hypothesis:

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 7 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [220]:
```
n_rows = data_df.shape[0]
```

In [221]:
```
X=data_df['X'].to_numpy().reshape(n_rows,1)
ones = np.ones((n_rows,1))
X = np.concatenate((ones, X), axis=1)
y=data_df['y'].to_numpy().reshape(n_rows,1)

def compute_cost(X, y, theta=np.array([[0],[0]])):

    m = len(y)
    # initialize loss to zero
    J=0
    # reshape theta
    theta=theta.reshape(2,1)

    # calculate the hypothesis
    h_x = np.dot(X,theta)

    error_term = sum((h_x - y)**2)
    loss = error_term/(2*m)

    return loss
```

$h_\theta(x) = \theta_0 + \theta_1 x$

In [222]:
```
compute_cost(X,y)
```

Out[222]:
```
array([6.08333333])
```

Applying Formulas

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 7 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

Out[222]: array([6.08333333])

In [223]:
```
def gradient_descent(X, y, theta=np.array([[0],[0]]),
                     alpha=0.01, num_iterations=1500):

    m = len(y)
    J_history = []
    theta0_history = []
    theta1_history = []
    theta = theta.reshape(2,1)

    for i in range(num_iterations):
        error = (np.dot(X, theta) - y)

        term0 = (alpha/m) * sum(error* X[:,0].reshape(m,1))
        term1 = (alpha/m) * sum(error* X[:,1].reshape(m,1))

        # update theta
        term_vector = np.array([[term0],[term1]])
#         print(term_vector)
        theta = theta - term_vector.reshape(2,1)

        # store history values
        theta0_history.append(theta[0].tolist()[0])
        theta1_history.append(theta[1].tolist()[0])
        J_history.append(compute_cost(X,y,theta).tolist()[0])

        return (theta, J_history, theta0_history, theta1_history)
```

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$$

J=0
$$\text{Theta\_0} = \theta_0 - \alpha\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)$$
J=1
$$\text{Theta\_1} = \theta_1 - \alpha\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1)$$

θ_0 = Theta_0
θ_1 = Theta_1

```
m 0.29, b 0.07, cost iteration 14.166666666666666
m 0.4871333333333333, b 0.11830000000000002, cost iteration 6.689183333333334
m 0.6210885555555555, b 0.15183466666666667, cost iteration 3.2281230840740744
m 0.712063267037037, b 0.17532177444444447, cost iteration 1.6259858114517922
m 0.773798218491358, b 0.19197091026296298, cost iteration 0.8842157211937784
m 0.815641461830572, b 0.20396561676330865, cost iteration 0.5406512187719745
m 0.8439526252352002, b 0.21279140209990244, cost iteration 0.38139003869808874
m 0.8630582641001964, b 0.2194588902914404, cost iteration 0.3074317815230537
m 0.8759018016694027, b 0.22465563399859784, cost iteration 0.27295607886944684
m 0.8844856941164487, b 0.2288493952017677, cost iteration 0.2567558763831943
```

Repeat the process up to, nth iteration until you get the best fit Hypothesis.

4. At first kept the Learning Rate = 0.001 and check the classifier using different learning Rates.

5. Output clearly shows the values of theta at every iteration along with the error rate running code for 10 iterations, output is showing the updated values of theta for each 10 iterations along with the error it got at every iteration.

Visualizing the data:

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 7 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [224]:
```
num_iterations=10
theta_init=np.array([[1],[1]])
alpha=0.001
theta, J_history, theta0_history, theta1_history = gradient_descent(X,y, theta_init,
                                                                    alpha, num_iterations)
```

In [225]:
```
theta
```

Out[225]:
```
array([[0.99866667],
       [0.9945    ]])
```

In [226]:
```
fig, ax1 = plt.subplots()

# plot thetas over time
color='tab:blue'
ax1.plot(theta0_history, label='$\\theta_{0}$', linestyle='--', color=color)
ax1.plot(theta1_history, label='$\\theta_{1}$', linestyle='-', color=color)
# ax1.legend()
ax1.set_xlabel('Iterations'); ax1.set_ylabel('$\\theta$', color=color);
ax1.tick_params(axis='y', labelcolor=color)

# plot loss function over time
color='tab:red'
ax2 = ax1.twinx()
ax2.plot(J_history, label='Loss function', color=color)
ax2.set_title('Values of $\\theta$ and $J(\\theta)$ over iterations')
ax2.set_ylabel('Loss: $J(\\theta)$', color=color)
ax1.tick_params(axis='y', labelcolor=color)
```

Data Visualized

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 8 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
ax2.plot(J_history, label='Loss function', color=color)
ax2.set_title('Values of $\\theta$ and $J(\\theta)$ over iterations')
ax2.set_ylabel('Loss: $J(\\theta)$', color=color)
ax1.tick_params(axis='y', labelcolor=color)

# ax2.legend();
fig.legend();
```

```
Estimated coefficients:
theta_0 = 1.1428571428571428
theta_1 = 0.7428571428571429
```