Assignment 1: Part 1 - Linear Regression

1. Linear Regression with One Variable

Task 1:

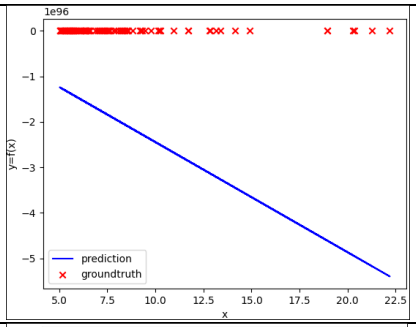
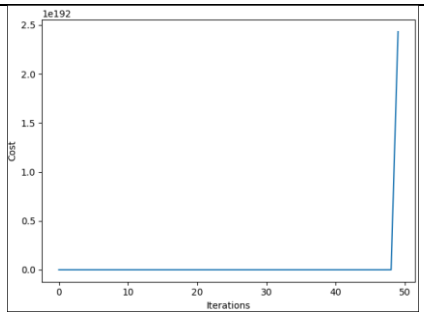
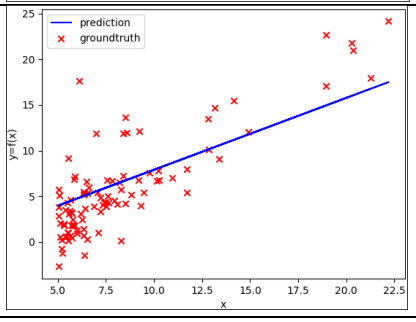
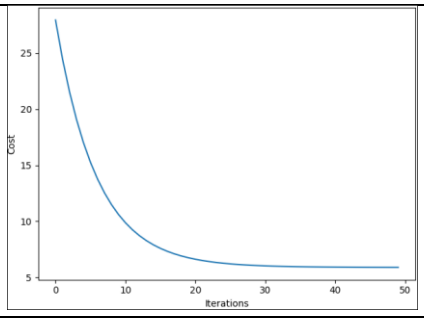
calculate_hypothesis.py

```
hypothesis = 0.0
########################################
# Write your code here
# You must calculate the hypothesis for the i-th sample of X, given X, theta and i.
hypothesis = theta[0]*(X[i][0])+theta[1]*(X[i][1])
########################################

return hypothesis
```

ml_assgn1_1.py

```
# initialise trainable parameters theta, set learning rate alpha and number of iterations
theta = np.zeros((2, 1))
alpha = 0.001
iterations = 50
```

alpha	y = f(x)	Cost Function	Minimum Cost
1.0			172570.09522
0.001			5.89503

2. Linear Regression with Multiple Variables

Task 2:

calculate_hypothesis.py

```
hypothesis = 0
########################################
# Write your code here
# You must calculate the hypothesis for the i-th sample of X, given X, theta and i.
for j in range(0, len(X[i])):
    hypothesis += theta[j] * (X[i][j])
```

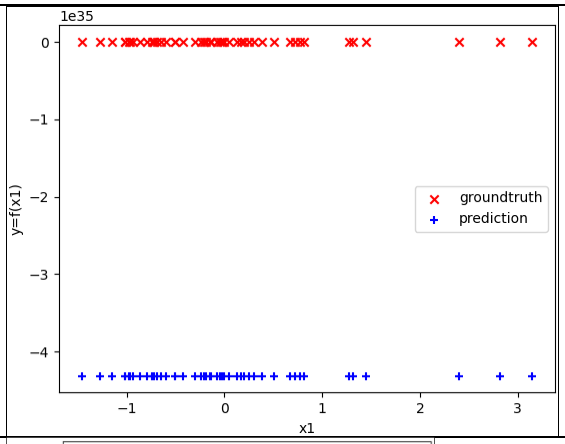
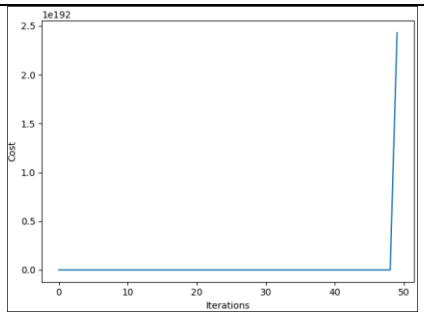
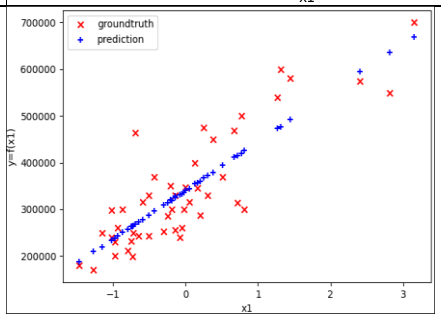
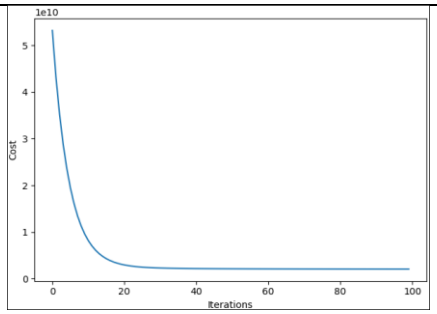
ml_assgn1_2.py

```
X1 = np.array([1650, 3])
X2 = np.array([3000, 4])
X1_norm = (X1-mean_vec)/std_vec
X2_norm = (X2-mean_vec)/std_vec
print(X1_norm, X1, X2_norm, X2, mean_vec, std_vec)
prediction1 = X1_norm[0][0]*theta_final[1]+X1_norm[0][1]*theta_final[2]+theta_final[0]
prediction2 = X2_norm[0][0]*theta_final[1]+X2_norm[0][1]*theta_final[2]+theta_final[0]
print(prediction1, prediction2)
```

gradient_descent.py

```
sigma = np.zeros((len(theta)))
for j in range(len(theta)):
    for i in range(m):
        ########################################
        # Write your code here
        # Calculate the hypothesis for the i-th sample of X, with a call to the "calculate_hypothesis" function
        hypothesis = calculate_hypothesis(X, theta, i)
        ########################################
        output = y[i]
        ########################################
        # Write your code here
        # Adapt the code, to compute the values of sigma for all the elements of theta
        sigma[j] = sigma[j] + (hypothesis - output) * X[i, j]
        ########################################
    # update theta_temp
    ########################################
    # Write your code here
    # Update theta_temp, using the values of sigma
    theta_temp = theta_temp - (alpha/m) * sigma
    ########################################
    # copy theta_temp to theta
theta = theta_temp.copy()
```

Alpha (α)	Minimum Cost	Theta (θ)
1.0	172570.09522	θ [0] = -4.31524312e+35 θ [1] = -5.39456370e+20 θ [2] = -2.57832468e+20
0.035	2053034617.85059	θ [0] = 340407.47981928 θ [1] = 105419.67289583 θ [2] = -1429.44609862

alpha	y = f(x)	Cost Function
1.0		
0.035		

Area in Square Feet	No. of Bedrooms	Predicted value of house price
1650	3	293708.8702485555
3000	4	472827.7991905983

3. Regularized Linear Regression

Task 3:

calculate_hypothesis.py

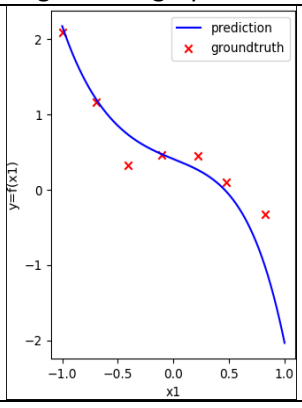
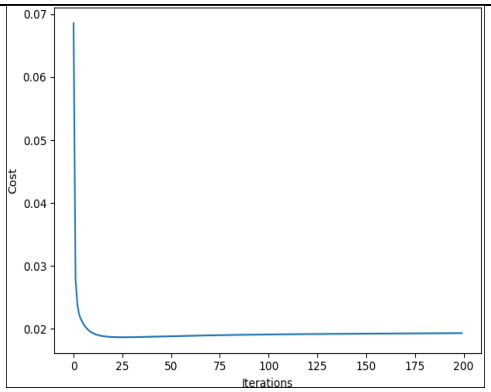
```
hypothesis = 0
########################################
# Write your code here
# You must calculate the hypothesis for the i-th sample of X, given X, theta and i.
hypothesis = theta[0]*X[i][0] + theta[1]*X[i][1] + theta[2]*(pow(X[i][2], 2)) \
             + theta[3]*(pow(X[i][3], 3)) + theta[4]*(pow(X[i][4], 4)) + theta[5]*(pow(X[i][5], 5))
```

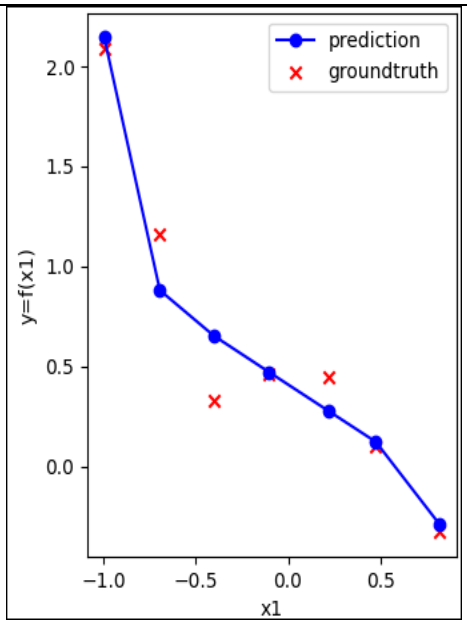
ml_assgn1_3.py

```
# initialise trainable parameters theta, set learning rate alpha, regularization parameter l and number of iterations
theta = np.zeros((6))
alpha = 1.0
l = 0.025
iterations = 200
```

gradient_descent.py

```
sigma = np.zeros((len(theta)))
for j in range(len(theta)):
    for i in range(m):
        ########################################
        # Write your code here
        # Calculate the hypothesis for the i-th sample of X, with a call to the "calculate_hypothesis" function
        hypothesis = calculate_hypothesis(X, theta, i)
        ########################################
        output = y[i]
        ########################################
        # Write your code here
        # Adapt the code, to compute the values of sigma for all the elements of theta
        sigma[j] = sigma[j] + (hypothesis - output) * X[i, j]
        ########################################
    # update theta_temp
    ########################################
    # Write your code here
    # Update theta_temp, using the values of sigma
    # Make sure to use lambda, if necessary
    if (j == 0):
        theta_temp[0] = theta_temp[0] - (alpha / m) * sigma[0]
    else:
        theta_temp[j] = ((theta_temp[j]) * (1 - ((alpha * 0.025) / m))) - (alpha / m) * sigma[j]
########################################
# copy theta_temp to theta
theta = theta_temp.copy()
```

Lambda	Regularized graph	Cost function
0.025		

Alpha	y = f(x)	Minimum cost
1.0		0.01869