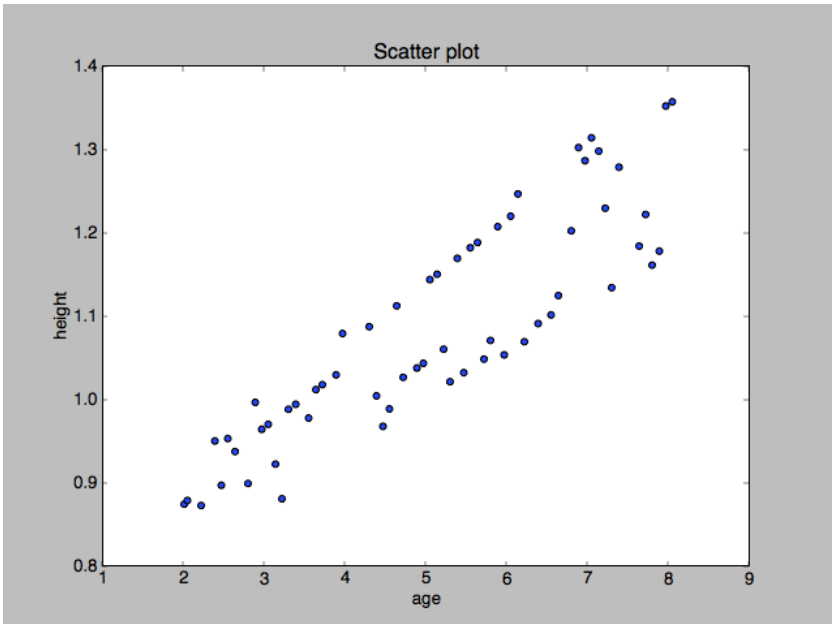
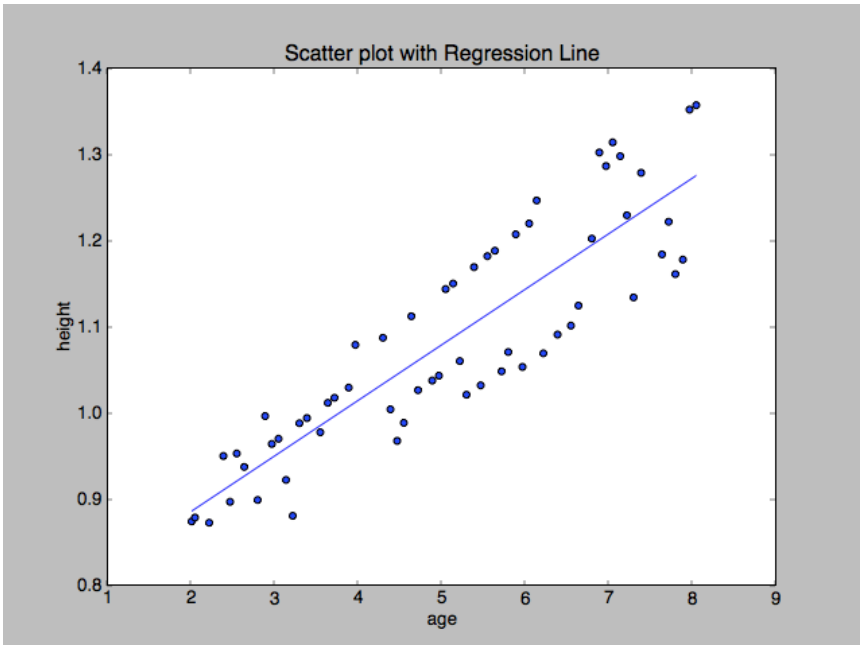
Problem 1: Regression

1.1)
a) Refer python code.
b) Distribution of data:

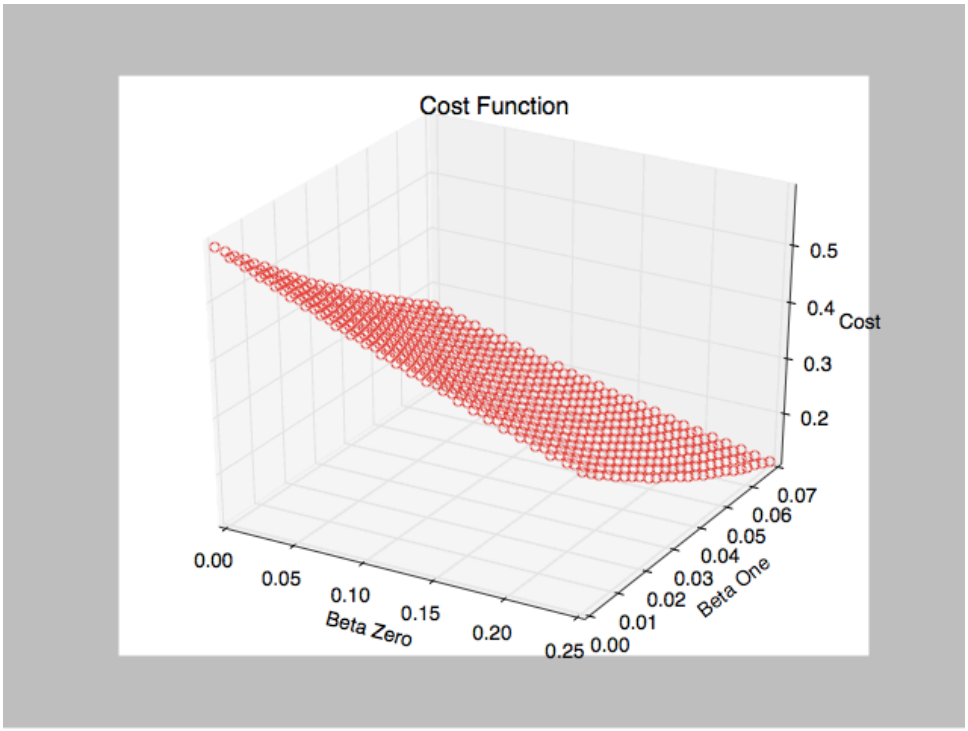


1.2)
Beta Zero = 0.756074908767
Beta One = 0.0644642896506
R (training) = 0.0018249076609

1.3) a)Regression Line:



c) Plot of cost function:



1.4)

a) Girl of age 4.5 yrs must have height 1.04616421219 m per our predicted model.
b) R (test) = 0.00320277538528.
The mean square on the test set is greater than on the training set.

2) Linear Regression with multiple features

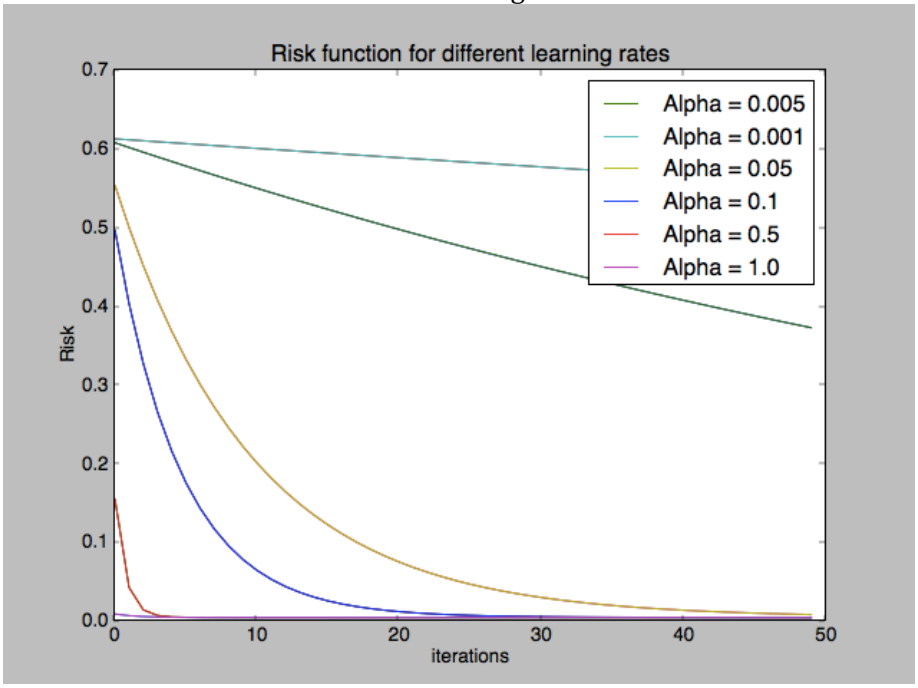
2.1) a)
Mean age = 5.21050632911
Mean weight = 18.3066043038
Std dev age = 1.89948154068
Std dev weight = 6.11745394303

b) Refer python code.

2.2)
Beta Zero = 1.09646081139
Beta One = 0.12861723221
Beta Two = 0.0014024840097
R (training) = 0.00236405034404

2.3)

a) Plot of Risk function for different learning rates



b) When alpha is small the convergence rate tends to be slower as confirmed by the graph above. For smaller values of alpha such as 0.005 and 0.001, the number of iterations (50) in this case is not enough for convergence.
For larger alphas convergence is faster as seen for the remaining alphas. Alpha = 1.0 converges the quickest within ten iterations.
It should be noted that if the alpha is larger than optimal, it may bounce off as it comes closer to the optimal point and possibly never converge. To the contrary, if the alpha is too small, it may take more than practical time to converge.
For all these reasons, the choice of alpha must be made carefully.

c) Best Alpha = 1.0
Beta Zero = 1.09646081139
Beta One = 0.128617232323
Beta Two = 0.00140248389656

d) Using Gradient Descent:
A 5-year old girl weighing 20kilos should have height 1.08259528491m.

2.4) Normal Equation

a) Beta Matrix =
[[7.39451312e-01] [6.77117569e-02][2.29259412e-04]]
The betas using gradient descent and normal equation are not the same since we have not scale the data in the latter case.

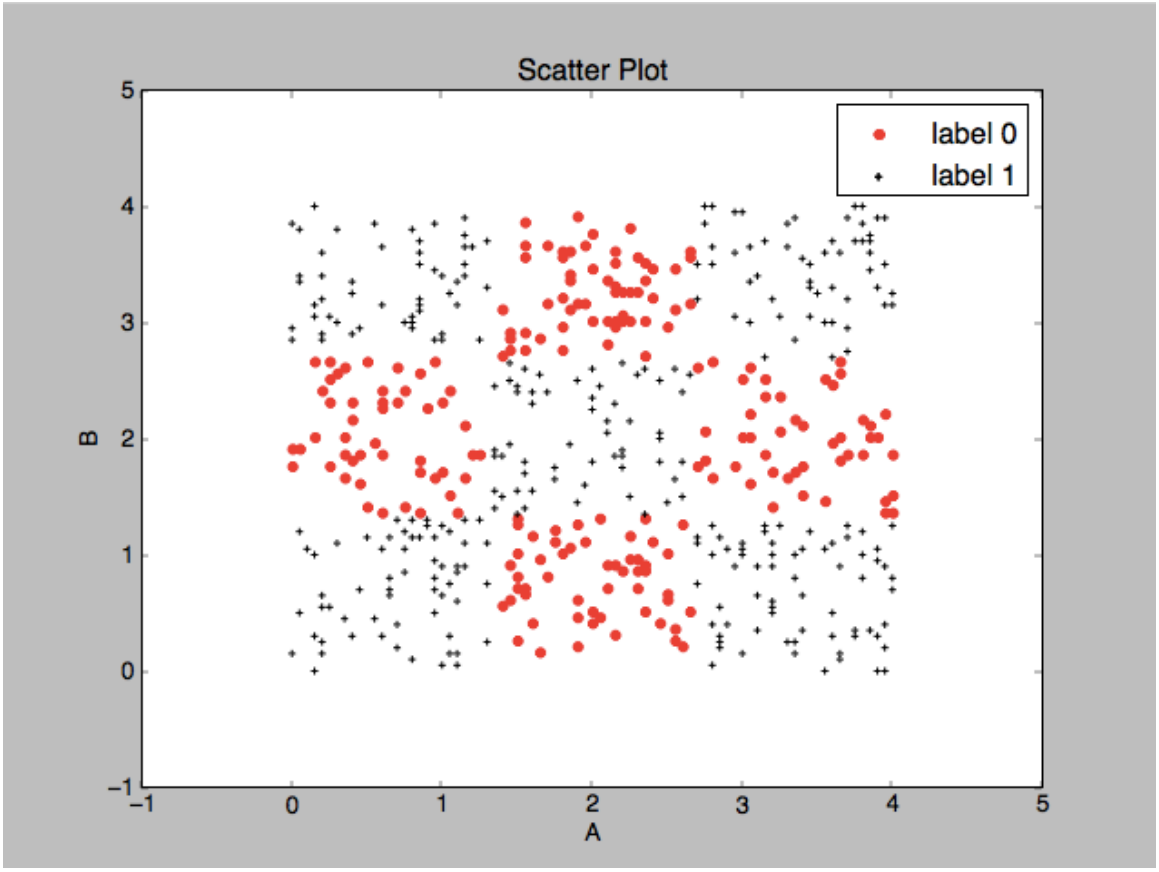
b) Using Normal Equation:
A 5-year old girl weighing 20kilos should have height 1.08259528491m.

c) Yes, they lead to the same height prediction.

Problem 2: Classification using SVMs and more

1) Refer python code.

2) Scatter Plot:



3) Using Support Vector Machines with different kernels to build a classifier:

a) Stratified sampling and k-fold cross-validation using k = 5 has been implemented. See below results for Linear, Polynomial and RBF kernels with different settings for parameters.

Linear SVM

Average score for C = 1 = 0.65551797802
Average score for C = 10 = 0.65551797802
Average score for C = 50 = 0.65551797802
Average score for C = 100 = 0.65551797802
Average score for C = 10000 = 0.65551797802

Score on test data with C = 10 is 56.5 %

Polynomial SVM

Average score for C = 0.01 & d = 2 is 0.65551797802
Average score for C = 0.01 & d = 3 is 0.65551797802
Average score for C = 0.01 & d = 4 is 0.635636915383
Average score for C = 0.01 & d = 5 is 0.739281509593
Average score for C = 0.01 & d = 6 is 0.70287551643
Average score for C = 1 & d = 2 is 0.65551797802
Average score for C = 1 & d = 3 is 0.65551797802
Average score for C = 1 & d = 4 is 0.749117575167
Average score for C = 1 & d = 5 is 0.725998637555
Average score for C = 1 & d = 6 is 0.752300805333
Average score for C = 100 & d = 2 is 0.65551797802
Average score for C = 100 & d = 3 is 0.65551797802
Average score for C = 100 & d = 4 is 0.749117575167
Average score for C = 100 & d = 5 is 0.722643105936
Average score for C = 100 & d = 6 is 0.714929435664
Average score for C = 1000000 & d = 2 is 0.62570539145
Average score for C = 1000000 & d = 3 is 0.543325699854
Average score for C = 1000000 & d = 4 is 0.60256365506
Average score for C = 1000000 & d = 5 is 0.643405230463

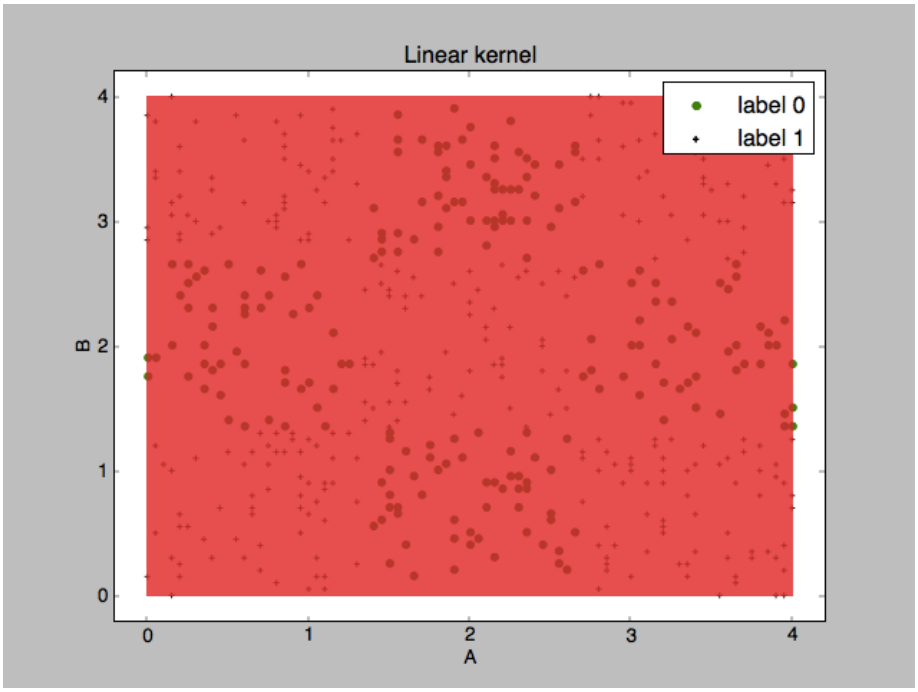
Score on test data with C = 1 and degree = 4 is 71.0 %

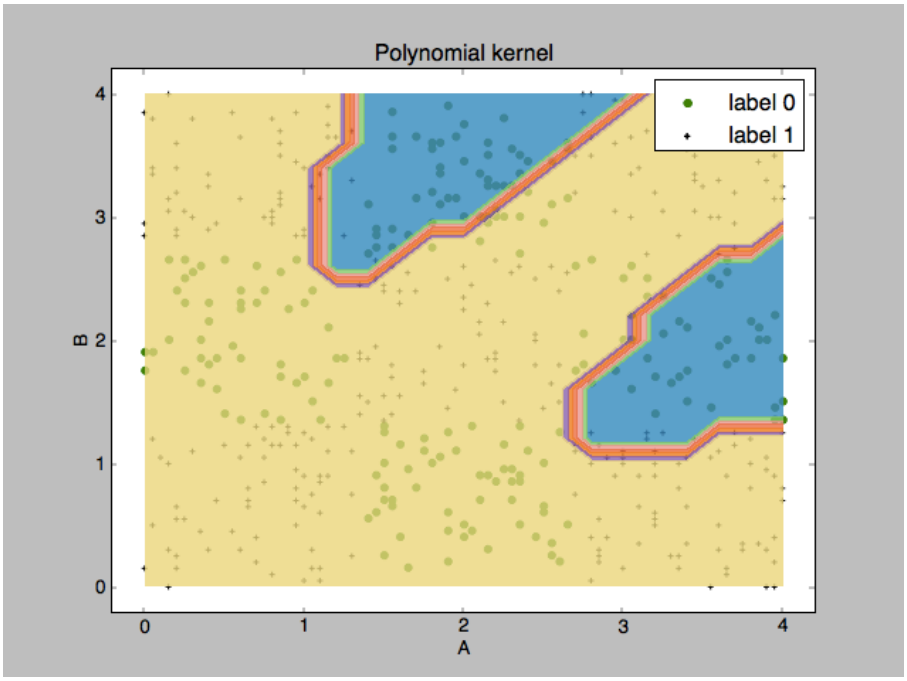
RBF SVM

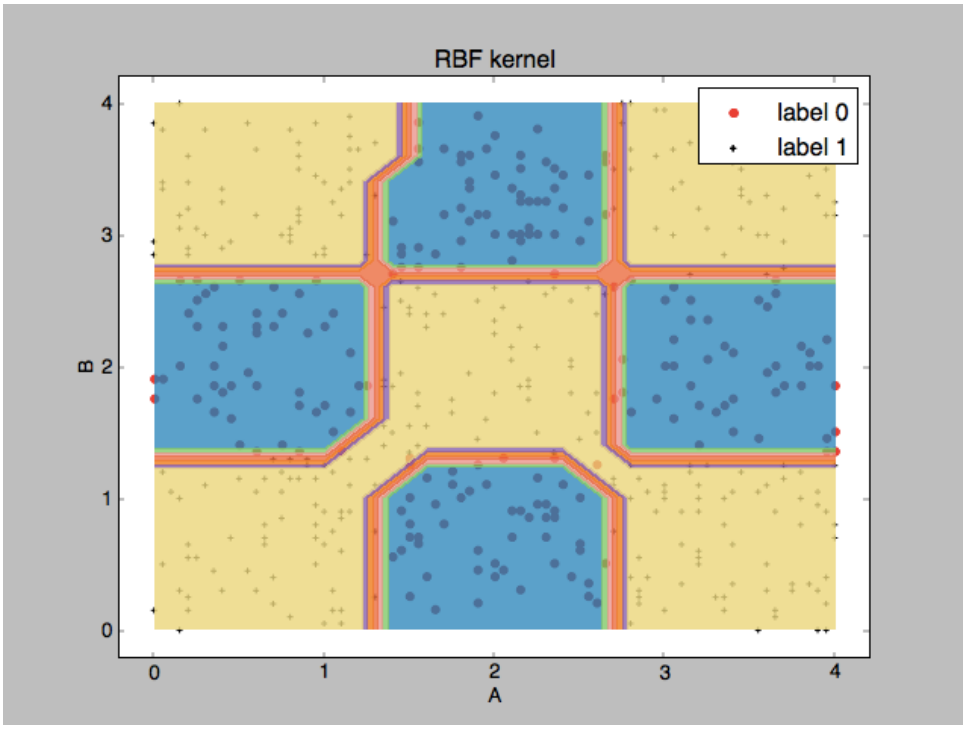
Average score for C = 0.01 & gamma = 0.001 is 65.551797802 %
Average score for C = 0.01 & gamma = 0.01 is 65.551797802 %
Average score for C = 0.01 & gamma = 0.1 is 65.551797802 %
Average score for C = 0.01 & gamma = 1 is 65.551797802 %
Average score for C = 1 & gamma = 0.001 is 65.551797802 %
Average score for C = 1 & gamma = 0.01 is 65.551797802 %
Average score for C = 1 & gamma = 0.1 is 65.551797802 %
Average score for C = 1 & gamma = 1 is 90.0976520249 %
Average score for C = 100 & gamma = 0.001 is 65.551797802 %
Average score for C = 100 & gamma = 0.01 is 63.551797802 %
Average score for C = 100 & gamma = 0.1 is 86.4820272125 %
Average score for C = 100 & gamma = 1 is 97.6917692425 %
Average score for C = 1000000 & gamma = 0.001 is 70.4226864535 %
Average score for C = 1000000 & gamma = 0.01 is 87.4633538921 %
Average score for C = 1000000 & gamma = 0.1 is 98.3570527833 %
Average score for C = 1000000 & gamma = 1 is 100.0 %

Score on test data with C = 100 and gamma = 1 is 98.5 %

b) The decision boundary for the best parameters for each kernel are as shown below:






RBF kernel
label 0
label 1
A
B
0
1
2
3
4

4) (Optional) See below the results, scores and decision boundaries for Random Forest Classifier, Logistic Regression and Perceptron.

Random Forest Classifier

Average score for n = 10 is 100.0 %
Average score for n = 50 is 100.0 %
Average score for n = 80 is 100.0 %
Average score for n = 100 is 100.0 %
Average score for n = 120 is 100.0 %
Average score for n = 500 is 100.0 %
Average score for n = 1000 is 100.0 %
Average score for n = 10000 is 100.0 %

Score on test data with n_estimators = 100 is 99.5 %

Logistic Regression

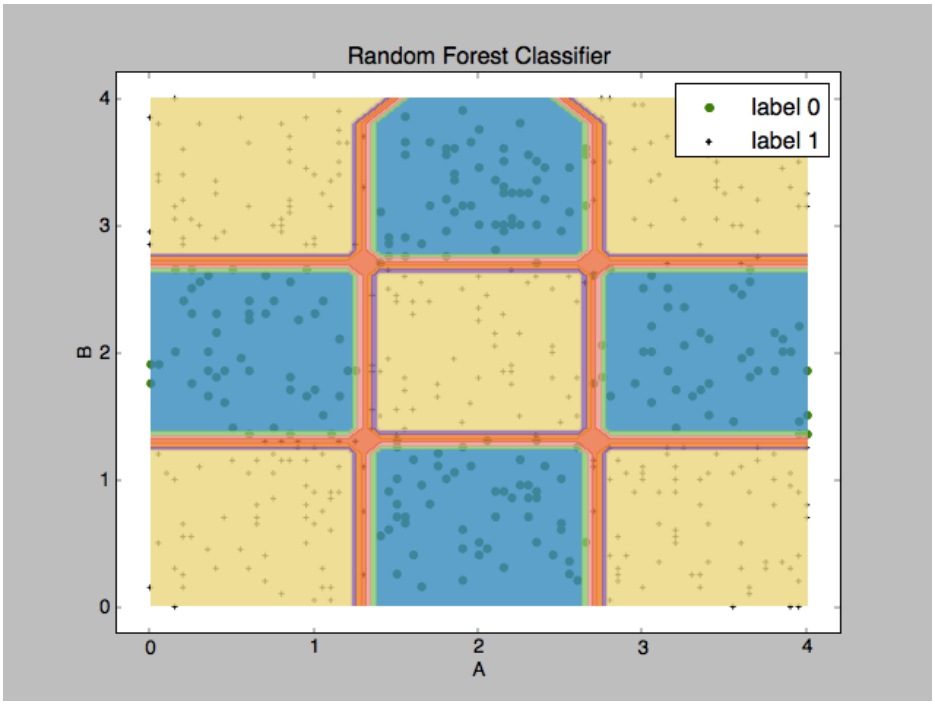
Average score for C = 1 is 65.551797802 %
Average score for C = 100 is 65.551797802 %
Average score for C = 1000 is 65.551797802 %
Average score for C = 10000 is 65.551797802 %
Average score for C = 1000000 is 65.551797802 %
Average score for C = 1000000000 is 65.551797802 %

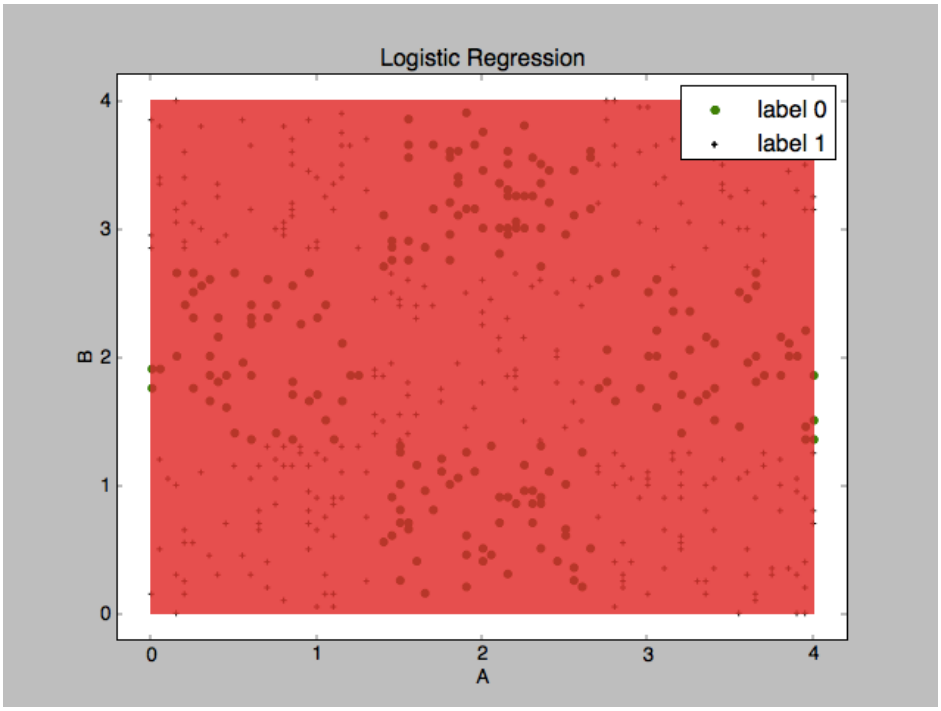
Score on test data with C = 100 is 56.5 %

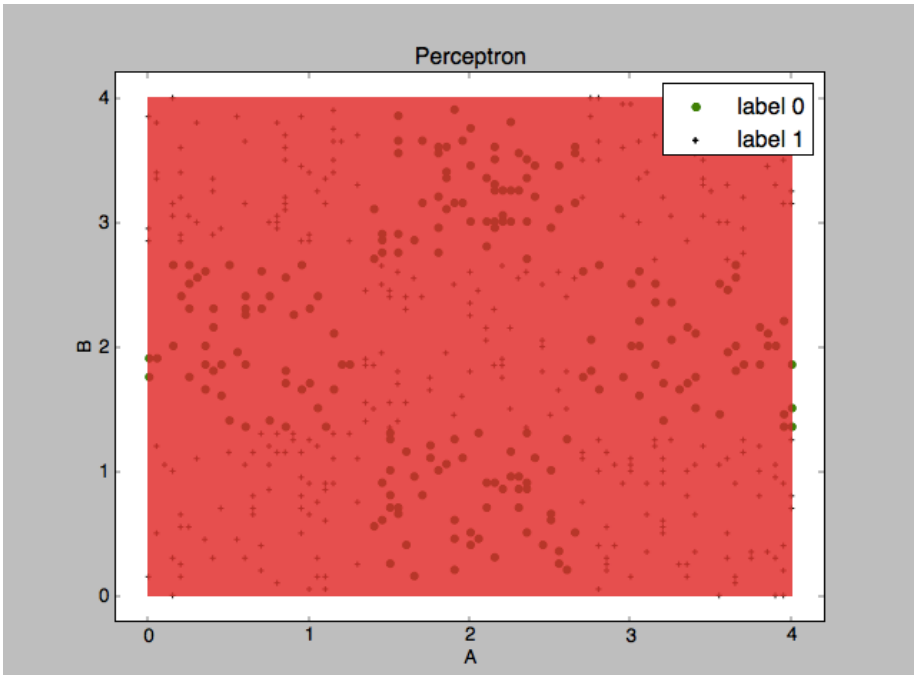
Perceptron

Average score for n_iter = 2 is 41.9135237617 %
Average score for n_iter = 3 is 40.2723780222 %
Average score for n_iter = 4 is 39.9427076925 %
Average score for n_iter = 5 is 40.2723780222 %
Average score for n_iter = 20 is 41.4093743592 %
Average score for n_iter = 50 is 40.2723780222 %

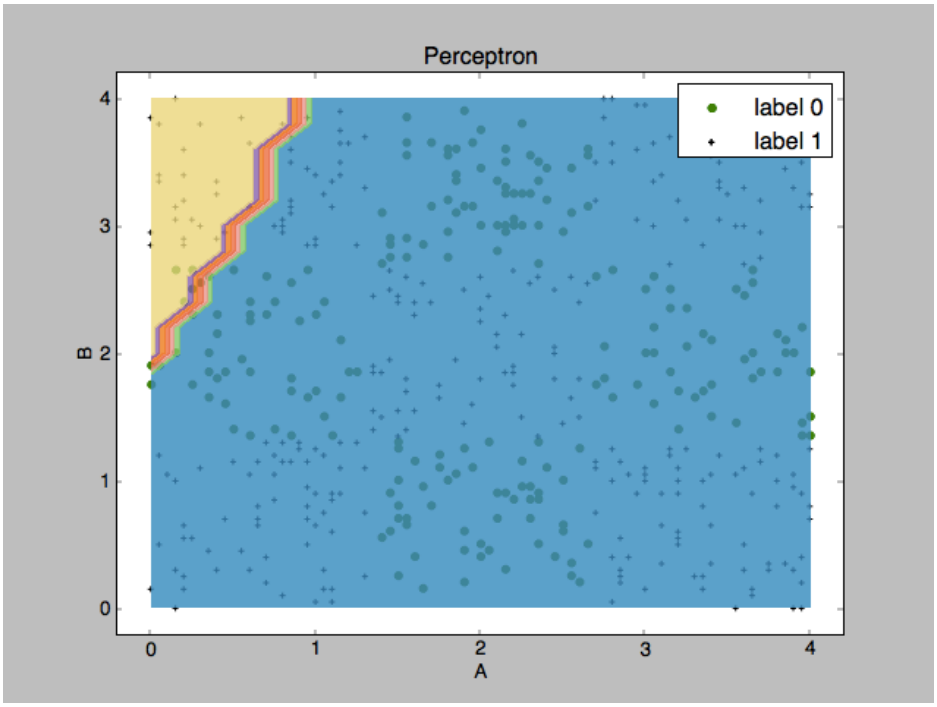
Score on test data with n_iter = 3 is 43.5 %


Random Forest Classifier
label 0
label 1
B
A
0
1
2
3
4


Logistic Regression
label 0
label 1
B
A
0
1
2
3
4



[Note for comparison below: Perceptron with same params on only training set gives slightly better classification than with entire data set.]

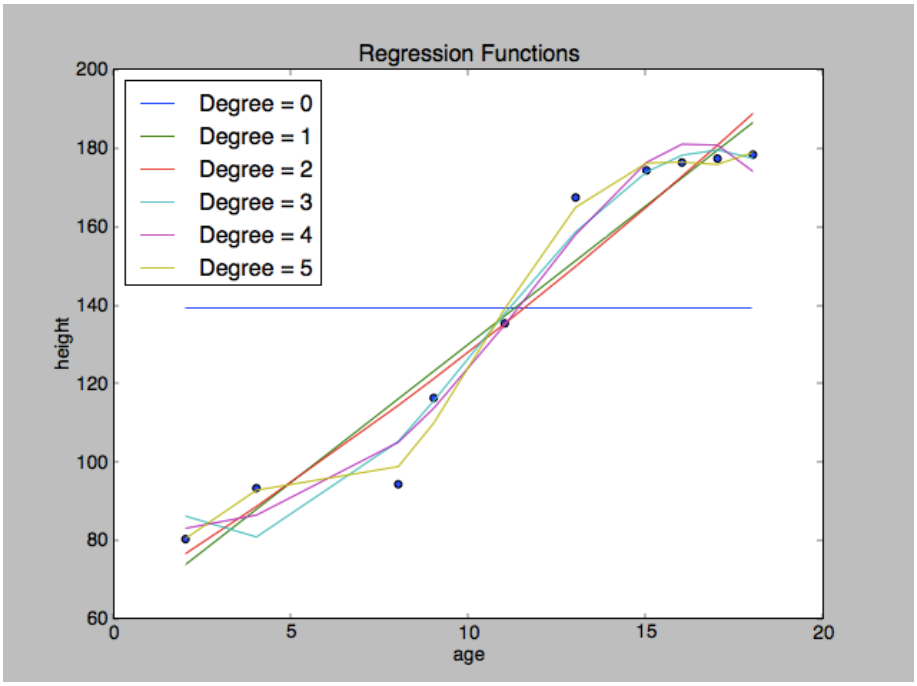


Problem 1 Part 3:

Regression with Polynomial Fitting (Optional)

1. Refer python code.

2. See plot below:



3. For degree 0, the age values are not taken into consideration at all and hence the function does not fit the data point at all.
   For degree 1 and 2, the functions under-fit the data.
   For degree 4 and 5, the functions over-fit the data.
   For degree 3, the function fits the data just right.

4. See below the beta values for various degrees and their respective mean square errors for test (R) and validation (R Validation).

Beta Matrix degree 0 = [[139.008]]

Beta Matrix degree 1 = [[59.43307607] [7.04202867]]

Beta Matrix degree 2 = [[64.82752204] [5.58808881] [0.07123999]]

Beta Matrix degree 3 = [[1.08332746e+02] [-1.62079010e+01] [2.65985750e+00] [-8.59554113e-02]]

Beta Matrix degree 4 = [[7.64036732e+01] [5.25986524e+00] [-1.42957215e+00] [2.09293372e-01] [-7.18971743e-03]]

Beta Matrix degree 5 = [[-1.37001017e+01] [8.40391805e+01] [-2.38524978e+01] [2.94754643e+00] [-1.56326559e-01] [2.97859248e-03]]

R = [725.724288, 51.05005600882032, 49.65150740623384, 20.04640966222418, 16.061916226020408, 4.375967456527297]

R Validation = [718.2232320000002, 97.16161178223012, 114.59342487457343, 22.345547976201207, 85.9497254543802, 171.16676617405818]

Analysis:
The mean square error training and testing is very high for degree 0 due to reasons mentioned above.
As the degree increases, the function starts fitting the training data better and thus fits the test data better as well. We can see this for degrees 1 through 3. For degree 1 and 2 the function is under-fitting the data or has high bias.
As we further increase the degree, the training error reduces, but the testing error increases. This is because the function is over-fitting the data or has high variance.
The best d from the graph below is 3 – the training error and testing error are the least, i.e. the function of degree 3 fits the data just right!