Question 2

a) I am using scikit-learn and Python for the question.

b) The normalizing factor should be calculated from the training data only. This is because any classification algorithm is to be finally applied to unknown data not available at the time the algorithm is constructed. So in order to get a good estimate of the model quality, we need to restrict the calculation of the normalization parameters to the training data set only and not peek into the test data set.

Normalization is done for the two basic reasons:

1. Avoid extra small model weights for the purpose of numerical stability.
2. Ensure quick convergence of optimization algorithms.

Hence, normalizing factor is only applied to the training data.

Mean : [1.29653933e+01 2.27000000e+00 2.37629213e+00 1.96494382e+01
9.89101124e+01 2.27235955e+00 2.02943820e+00 3.60674157e-01
1.57617978e+00 5.09123596e+00 9.53213483e-01 2.56033708e+00
7.29707865e+02]

Std : [8.19560112e-01 1.10306417e+00 2.73004021e-01 3.46517583e+00
1.15589748e+01 6.14548382e-01 9.19718276e-01 1.20241309e-01
5.41470129e-01 2.40437795e+00 2.29907577e-01 7.23461482e-01
3.07221225e+02]

c) I) If no initial weight vector is specified, it takes the initial weight vector values as zeros.

II) In the parameters for the perceptron classifier, we mention tolerance by the parameter tol. This tol is related to the stopping criterion for the perceptron algorithm. The default value is none. If it's not default, the iterations will stop when the loss is not improving by atleast tor for an iteration. It defaults to 1e-3 from version 0.21. The backup halting condition is specified by max_iter. It is a parameter which tells the number of passes over the training data. It's default value is 5. And if tol is not none and the halting condition is not met, then it's default value is 1000. It's default value is 1000 from version 21.

d) Training data using 2 features:
Final Weights: [[2.00765415 0.26290402]
[-2.5191645 0.29916664]
[1.56839412 0.85217164]]
Accuracy : 0.786516853933

Test data using 2 features:
Accuracy: 0.741573033708

Training data using all 13 features:
Final Weights:
[[3.05233624 1.7406059 1.99692494 -2.42476792 4.72710171 -1.12368988
3.47688297 -0.55973584 0.80284965 -2.48507361 2.40467905 1.5776214
4.97957608]
[-4.55588329 -3.67159057 -3.26414092 1.02399266 -2.87825379 1.63872965
0.48744775 3.50979604 -0.57521303 -6.28698669 2.04273585 -0.34416312
-5.83578301]
[1.72427385 -0.23570705 1.17337861 0.86445992 0.37035282 0.22598124
-2.07073669 -1.03630559 -1.88479797 5.26118935 -2.39353632 -2.87133928
-1.53419439]]
Accuracy : 1.0

Test data using all 13 features:
Accuracy: 0.910112359551

e) Training data using 2 features:
Final Weights: [[2.36834302 -0.8636138]
[-3.28950049 -1.02895216]
[0.07047386 1.55300424]]
Accuracy : 0.85393258427

Test data using 2 features:
Accuracy: 0.786516853933

Training data using all 13 features:
Final Weights:
[[2.71428596 0.69877741 3.68620586 -2.41671443 3.70203636 -0.6253213
1.87807971 -0.9180845 0.08459284 -0.88361029 0.23990029 3.68705672
4.23965385]
[-4.01709141 -4.20588675 -3.95448155 2.2538574 -3.15210446 3.39669718
0.06472664 5.06541262 -0.04428499 -8.56393273 3.79007765 0.26326683
-6.47808962]
[1.5185144 3.20798873 -0.94933029 1.13491224 1.7364941 -0.8177632
-2.10372361 -0.86999885 -1.60834789 4.2057664 -1.15401559 -2.50611104
-1.74738319]]
Accuracy : 1.0

Test data using all 13 features:
Accuracy: 0.932584269663

f) We can see from d) that, the test accuracy for including all feature is greater than that when we build the model on just the first two feature. This is because as we increase the number of

features, it helps the algorithm by giving more data to model to work with. But it's important to choose the right features only, as sometimes using more features can lead to overfitting.

Similarly, from e), we can see that the test accuracy for including all feature is greater than that when we build the model on just the first two feature. Here, using random initial weight vectors, we find the best accuracy on the training set.

We also see that the accuracy in e) is better than that of d) for all testing data. This is because of the random initial weight vector we used and the selection of the most accuracy on the training dataset.

g) MSE calculations:

Running on unnormalized data, accuracy:

2 feature test data accuracy: 0.752808988764
All feature test data accuracy: 0.977528089888

h) MSE calculations:

Running on standardized data, accuracy:

2 feature test data accuracy: 0.752808988764
All feature test data accuracy: 0.977528089888

i) The accuracy results for both 2 features and all features are same for the unstandardized and standardized data. As the pseudo inverse version classification is a tree based algorithm, i.e., we can write the classification condition in the form of a tree, so, it's invariant to standardization. This is because, tree based methods are scale-invariant. Hence, here standardization doesn't change the accuracy of the classifier on the test data.

j) Comparing the results of h) and e), we can say the accuracy rate of MSE is better on the test data in both cases than the perceptron classifier. Perceptron aims to minimize the criterion function, i.e., minimize the number of misclassified data points. MSE on the other hand focuses on making a boundary to divide the data into different classes by minimizing the distance between the data points and the boundary. This is however not the general case that MSE always has greater accuracy than perceptron. The vice versa is also possible. It's only in this case that we get this result.