1.a. Linear regression assumes uncertainty in dependent variable y and not in X.

b. We can use Cook's D (A measure that combines the information of leverage and residual of the observation) to explore the impact of individual cases on the regression line (using an influence plot) and then use Robust regression.

c. It is assumed that magnitude of the regression coefficients (abs(w)) indicate the importance of the corresponding features but this will not work if a feature represent the same thing in different units such as distance in Km and mm. The change in units shouldn't change the importance of a feature.

d. We will get a non-invertible matrix of (XX^T), as the feature matrix X will become singular.

e. We can use K-1 encoding. So if there are 2 unique values we will represent it by 0's or 1's using only one feature vector.

f. The coefficients will vary widely between different samples of data, so we will not be able to draw any conclusions about the coefficients. This might become a problem to decide which feature has the most influence.

g. If outliers are present then we may not get the same predictive accuracy using linear regression and the advantage of logistic regression over linear regression is that it is not as sensitive to outliers as linear regression. This happens because logistic regression uses the sigmoid function.

h. When the sample is less than the number of features we will get an under-determined system of equations.

2.a

P(x,y)= P(y)P(x|y) so let us assume

$P1 = \frac{1}{\sqrt{2\pi^D|\sum^{0.5}|}} * \exp -\frac{1}{2}(X^T\ (\sum)^{-1}X)$ for y =class1

and $P2 = \frac{1}{\sqrt{2\pi|\sum^{0.5}|}} * \exp -\frac{1}{2}(X^T\ (\sum)^{-1}X)$ for y= class2

Here, $\sum$ is the co-variance matrix.

Solving this for class1, we get

$$p(x|y=c1) = \frac{1}{(2\pi)^D\sigma^{2D}}\ e^{(\frac{-1}{2\sigma^2}((x_1^2+x_2^2+\cdots+x_{2D}^2))}$$

$$p(x|y=c2) = \frac{1}{(2\pi)^D\sigma^{2D}}\, e^{(-\frac{1}{2\sigma^2}\left((x_1^2+x_2^2+x_3^2+\cdots(x_{D+1}+\delta)^2+\cdots(x_{2D}+\delta)^2\right))}$$

$$logP(D) = \sum_n \log p(x_n, y_n)\ = \sum_{n:y_n=c1} log(p_1 \frac{1}{(2\pi)^D\sigma^{2D}} e^{\frac{-1}{2\sigma^2}((x_1^2+x_2^2+\cdots+x_{2D}^2)} +$$

$$\sum_{n:y_n=c2} log(p_2 \frac{1}{(2\pi)^D\sigma^{2D}} e^{\frac{-1}{2\sigma^2}((x_1^2+x_2^2+(x_{D+1}+\delta)^2\ldots+(x_{2D}+\delta)^2)}$$

Where $p_1 + p_2 = 1$

Let

$N_1 be\ the\ number\ of\ samples\ of\ class\ c1\ and\ N_2 be\ the\ number\ of\ samples\ of\ class\ c2$
$using\ lagranges\ multiplier\ and\ taking\ derivative\ w.r.t\ to\ p_1 we\ get$

$\frac{dl}{dp_1} = \frac{N_1}{P_1} + \lambda = 0 => \lambda = -\frac{N_1}{P_1}\ and\ P_1 = -\frac{N_1}{\lambda}$ Similarly,

$p_2 = -\frac{N_2}{\lambda}$ =>

Replacing this value in $p_1 + p_2 = 1$

$$-\frac{N_2}{\lambda} + -\frac{N_1}{\lambda} = 1 => \lambda = -(N_1+N_2), substituting\ this\ above\ we\ get$$

$$p_2 = \frac{N_2}{N_1+N_2}\ and\ p_1 = \frac{N_1}{N_1+N_2}$$

Solving for σ we can see that solution will change if delta δ is changed.

For the Gaussian discriminant analysis since we have the same mean it also will give the same solution as the above.

3) Perceptron learning

If a point is correctly classified then we need not do anything. That is if $y_{i+1} = sign(w_{i+1}^T x_{i+1}), then\ w_{i+1} = w_i$. Otherwise we need small amount of movement such that then the point x_{i+1} is on the correct side of the plane. So we have

$$w_{i+1} = \arg\min \frac{1}{2} ||w - w_i||_2^2 \qquad s.t. \quad w^T x_{i+1} - y_{i+1} = 0$$

$$but \qquad ||w - w_i|| = (w - w_i)^T (w - w_i)$$

Using Lagrange's Multiplier, we get

$$L(w, \lambda) = \frac{1}{2}(w - w_i)^T (w - w_i) + \lambda w^T x_{i+1} y_{i+1}$$

Taking derivative w.r.t w and setting it to zero we get

$$\frac{d}{dw} L(w, \lambda) = (w - w_i) - \lambda x_{i+1} y_{i+1} = 0$$

$$=> \quad w = \lambda x_{i+1} y_{i+1} + w_i$$

Taking the transpose on both sides we get

$$w^T = \lambda\ (x_{i+1} y_{i+1})^T + w_i^T$$

Multiplying both sides by $x_{i+1} y_{i+1}\ then\ applying\ the\ w^T x_{i+1} y_{i+1} = 0\ we\ get$

$$w^T x_{i+1} y_{i+1} = 0 = \lambda (x_{i+1} y_{i+1})^T (x_{i+1} y_{i+1}) + w_i^T (x_{i+1} y_{i+1})$$

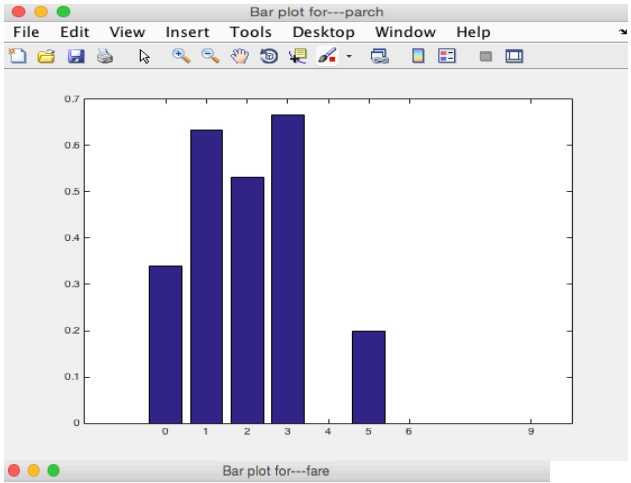
$$\lambda = -\frac{w_i^T (x_{i+1} y_{i+1})}{||x_{i+1}||_2^2}$$

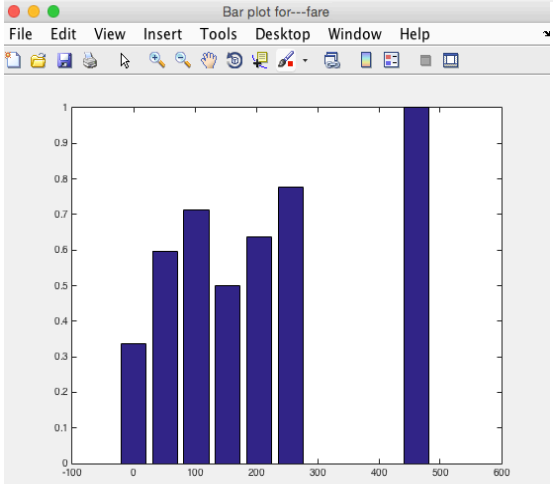
Substituting the value of $\lambda\ in\ w = \lambda x_{i+1} y_{i+1} + w_i\ we\ get$

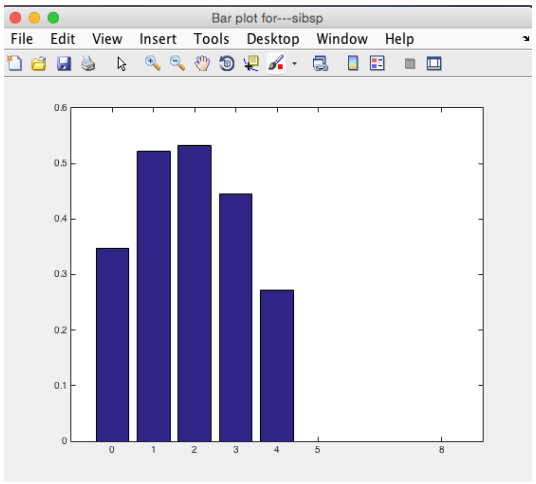
$$w_{i+1} = w_i - \frac{w_i^T x_{i+1}}{||x_{i+1}||_2^2}\ x_{i+1}$$

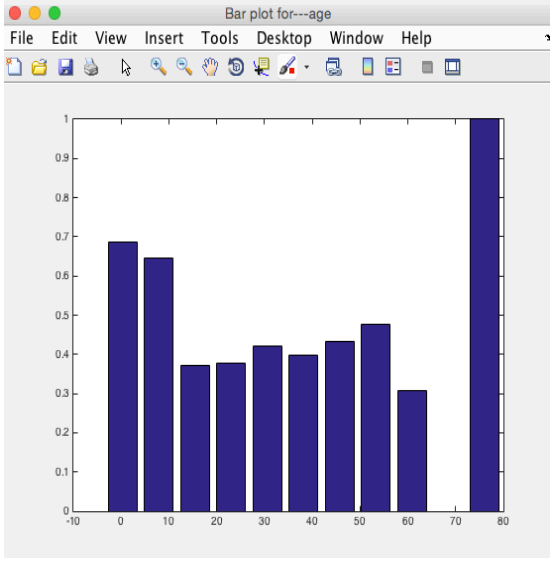
The above equation has the lowest update on $w_i\ when\ the\ sign(w_i x_{i+1}) is\ negative$.

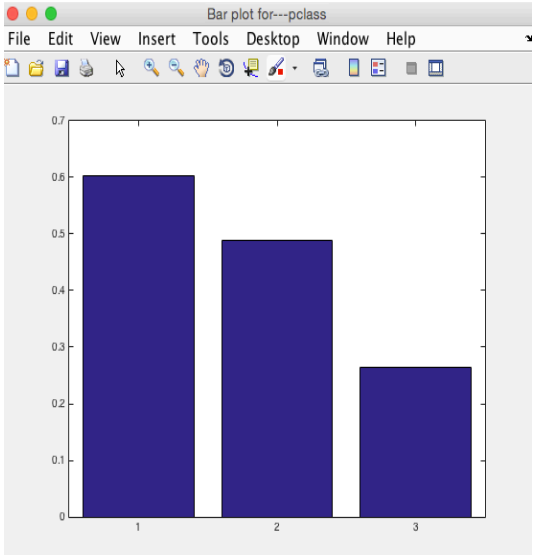
4a. The number of missing values for pclass is 0
The number of missing values for name is 0
The number of missing values for sex is 0
The number of missing values for age is 263
The number of missing values for sibsp is 0
The number of missing values for parch is 0
The number of missing values for ticket is 0
The number of missing values for fare is 1
The number of missing values for cabin is 1014
The number of missing values for embarked is 2
The number of missing values for boat is 823
The number of missing values for body is 1188
The number of missing values for home.dest is 564











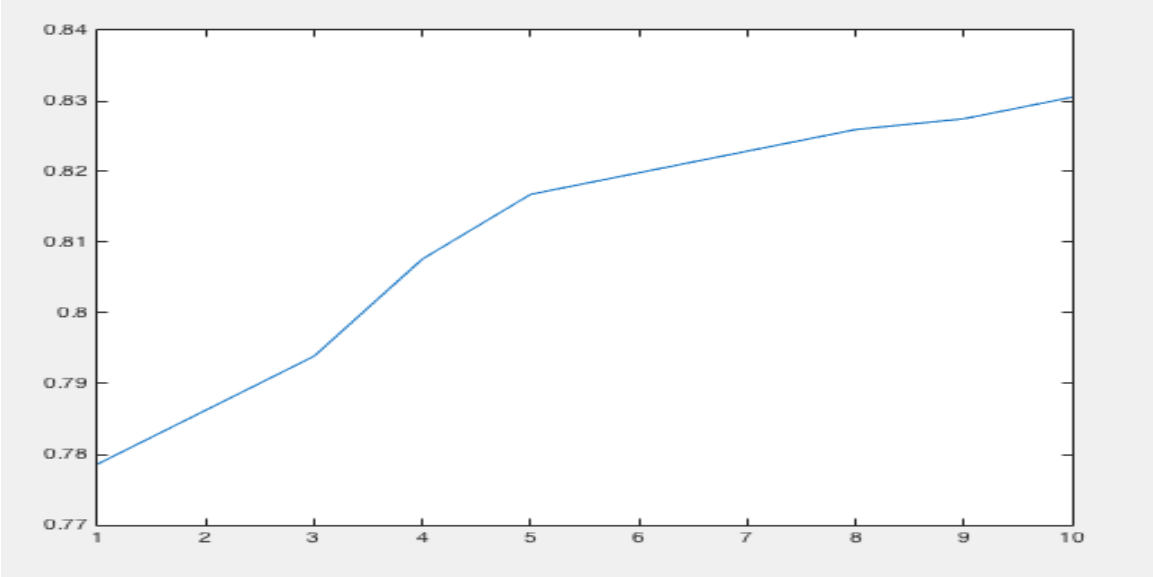
4.b From these figures we can see that pclass has a monotonic relationship but it is not true for others.

4c.
Mutual Information for (survival , pclass) = 0.067488
Mutual Information for (survival , age) = 0.032747
Mutual Information for (survival , sibsp) = 0.030167
Mutual Information for (survival , fare) = 0.109385
Mutual Information for (survival , parch) = 0.042630
Mutual Information for (survival , sex) = 0.192352
Mutual Information for (survival , name) = 0.017337
Mutual Information for (survival , embarked) = 0.022756

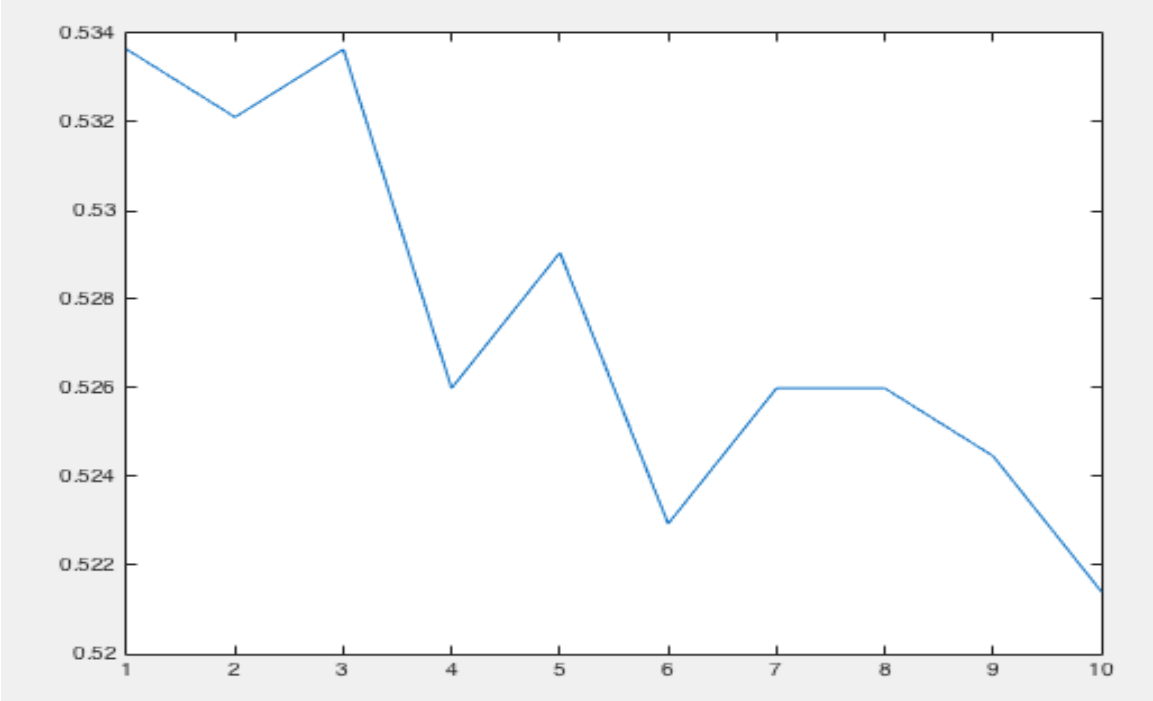
4.d Most of the times multiple model gives a better accuracy.
Multiple model
Train Accuracy =0.807634 Test accuracy=0.778287

Replacing Values
Train Accuracy =0.803053 Test accuracy=0.776758

4f. For the sequential feature selection the plot of the no of features against the training accuracy is given below and it range from 0.78-0.84.



For the testing data the plot is given below for the same set of features selected for the train data. The plot varies as the data sample varies.



4g.
For eta = 0.100000

Iterations=10 Accuracy=0.650382
Iterations=100 Accuracy=0.818321
Iterations=500 Accuracy=0.825954
Iterations=1000 Accuracy=0.825954

For eta = 0.020000

Iterations=10 Accuracy=0.625954
Iterations=100 Accuracy=0.729771
Iterations=500 Accuracy=0.818321
Iterations=1000 Accuracy=0.824427

For eta = 0.500000

Iterations=10 Accuracy=0.806107
Iterations=100 Accuracy=0.825954
Iterations=500 Accuracy=0.824427
Iterations=1000 Accuracy=0.824427

4h. Iterations for Newton's method = 2 Accuracy=0.835115
So we can see that Newton's method converges much faster than the gradient descent.

I have discussed with Ismail, Swaroop, Amandeep, Roli and Ekram.