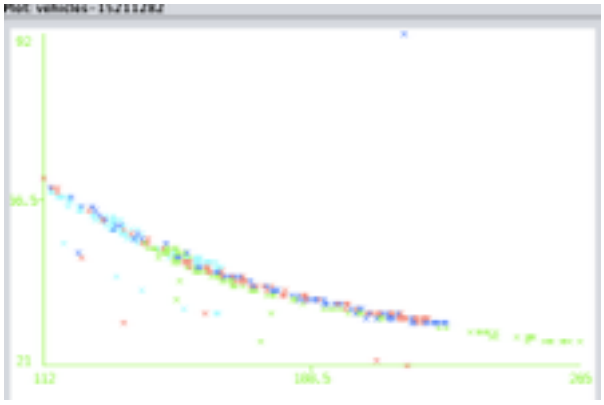
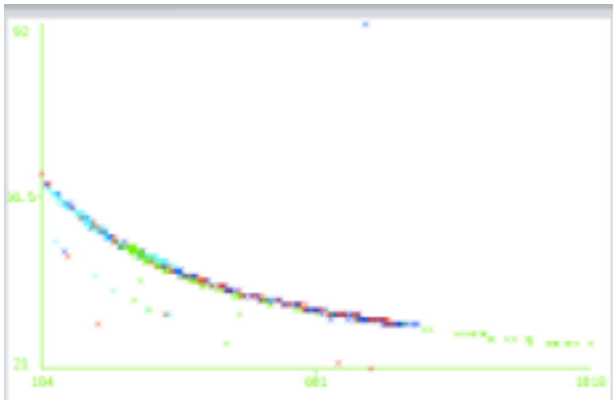
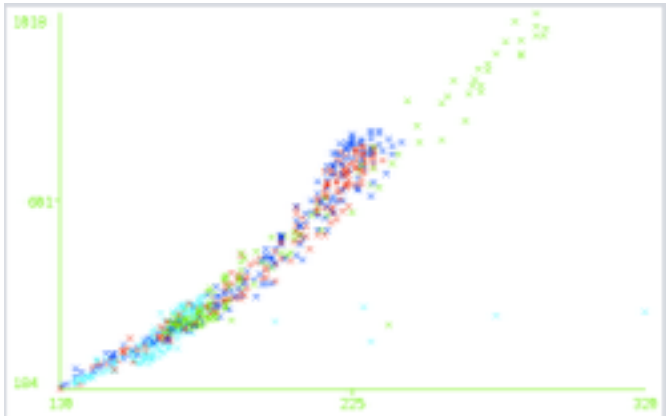
Question 1

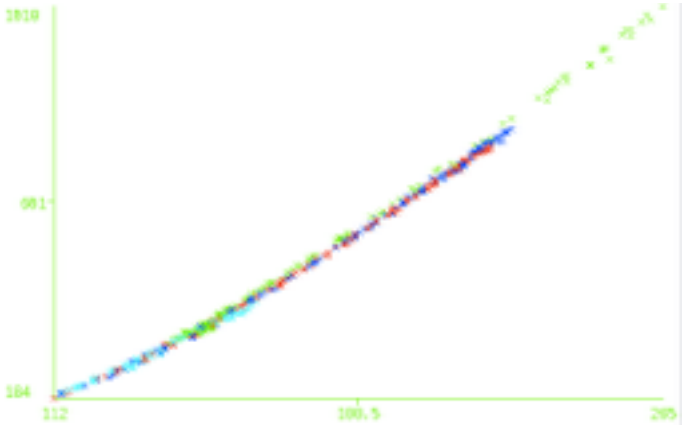
(a) Firstly, looking at this data set details, it can be seen that there are 778 instances and 20 attributes. 19 features all are numerical and one feature (target feature) is categorical with 4 different classifications. There were also no missing values in the all data set. Meanwhile, it is noticed that each class had similar instances in the data set. So, imbalance data might be avoid. Secondly, looking at the attributes correlation visualizations, it is very clear to notice that between some features had the positive or negative relationships. For example, SCATTER RATIO(x axis) had a negative relationship with V.ELONGATEDNESS(y axis) feature(seen the left plot). In addition, when x axis value was greater than 234.0, there was only one class(minibus class) appeared (the green color) that proved it was purity. SCELED VARIANCE_MINOR (x axis) also had a negative relationship with V.ELONGATEDNESS(y axis) feature(seen the right plot).





And when x axis value was greater than 766.0, there was also only one class (minibus) occurred. Taking one more instance, CELED VARIANCE_MAJOR (x axis) had a positive relationship with CELED VARIANCE_MINOR (y axis) shown in the below left plot. When the x axis value was greater than 247.0 and y axis value was greater than 731.0, there was only one class(minibus) appeared which means that it had the higher purity. SCATTER RATIO (x axis) also had a positive relationship with CELED VARIANCE_MINOR (y axis) shown in the right plot. When SCATTER RATIO value was greater than 234.0 and CELED VARIANCE_MINOR was greater than 822.0, it had only one class(minibus).





Except the above examples, there were many obvious correlations between attributes in my data set from analyzing the visualization plot. It might be good for using Decision Tree classifier. Finally, it

had mentioned before that there was 4 different classes (compact, convertible, minibus and suv)in the target features. But 3-NN classifier only chose 3 shortest-distance instances around the input variable. In other words, it might not include all classes. It might not be accurate. Therefore, according to initially analyzing the original data set, Decision Tree might be better than 3-NN.
In order to evaluate the both classifiers performance, cross validation with folder=10 was used. The accuracy for both classifiers are showing in the below:

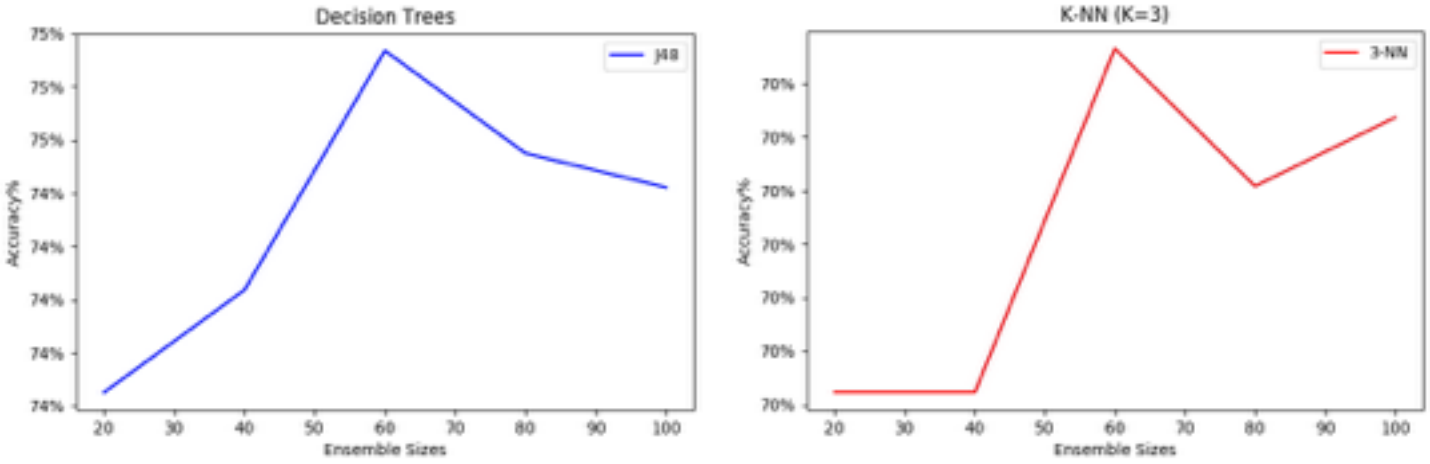
Classifer	Correct%	Incorrect%
J48	72.8792%	27.1208%
3-NN	69.4087%	30.5913%
Weighted 3-NN	70.5656%	29.4344%

Compared to accuracy values, J48 is better classifier for this data set than both 3-NN. Meanwhile, the accuracy for 3-NN and weighted 3-NN did not had a big difference. In order to compare between two different classifiers, I chose majority voting (3-NN) for k-NN classifier.

(b) Utilizing 10 fold cross validation to evaluate the performance of J48 and 3-NN classifiers by applying bagging ensemble strategy, the following is presenting the accuracy as the ensemble size increase:

Bagging strategy	J48	3-NN
Ensemble Size	Accuracy	
20	73.6504%	69.9229%
40	74.036%	69.9229%
60	74.9357%	70.5656%
80	74.5501%	70.3085%
100	74.4216%	70.437%

Obviously, both J48 classifier and 3-NN had the improvement over a single classifier by using ensemble Bagging strategy. By compared with the two classifiers' accuracy, it was seen that the J48 had the better accuracy than 3-NN no matter what ensemble sizes had. The reason is that bagging encourages diversity in the ensemble and works better for unstable classifiers like Decision Trees. In other words, Bagging can be ineffective when using a stable classier like K-NN. K-NN is to find the nearest neighbors by calculating the distance between features. But bagging strategy randomly select samples with replacement, which lead to the similarity accuracy in K-NN no matter what ensemble sizes had. It could cause that diversity did not grow as well.
The following is the plot graphs for both classifiers in different ensembles size:



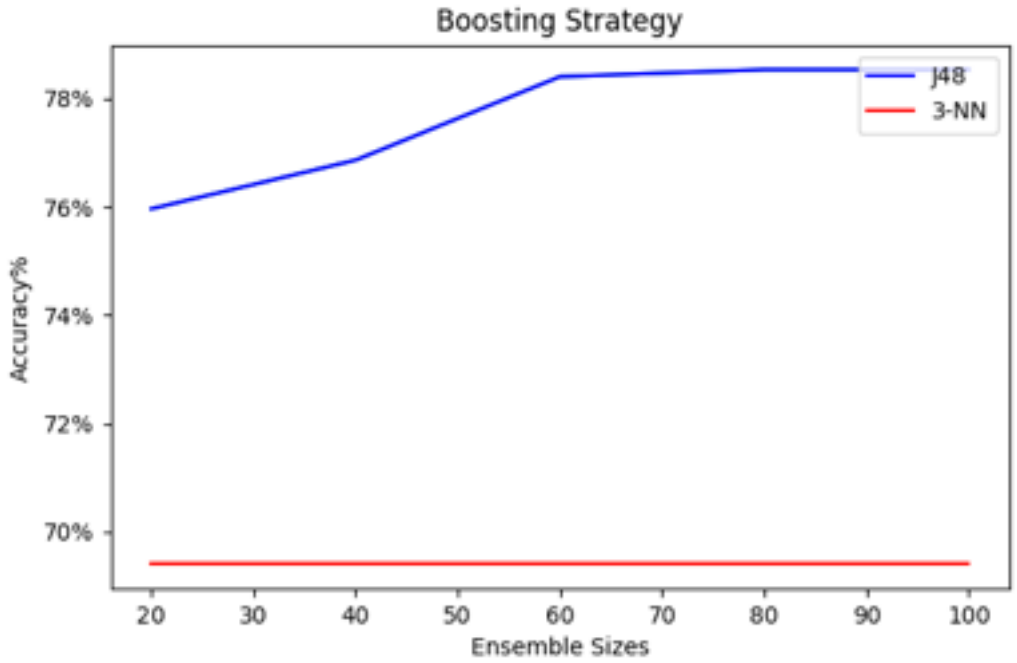
According to the above graphs for each classifier, it can be clearly noticed that when **ensemble size that is equal to 60** had the best performance by comparing the accuracy. After that, the both accuracy had the decrease trend. The reason could explain this was that when the ensemble sizes increases a certain level, the new ensemble members will have the prediction patterns collinear with existing members. In other words, there is no new diversity added. Therefore, the accuracy will plateau.
Using the best performing ensemble size 60 and evaluating the performance by using 10 fold cross validation , the accuracy for both classifiers in different bag size in the below:

Bagging strategy with ensemble size 60	J48	3-NN
Different bag size percent %	Accuracy	
100	74.9357%	70.5656%
80	73.9075%	70.6941%
60	73.5219%	70.3085%
50	73.9075%	70.9512%
40	73.6504%	71.7224%
20	72.365 %	70.437 %

From the above table, it found that accuracy for J48 classifier had a decrease trend.
J48 classification performance did not improve by decreasing the bag size. In other words, 100% bootstrap sample would have the best performance for J48. When the bootstrap sample decreases, there will have many duplicated examples in the bootstrap subset. And then the diversity in the ensemble will decrease as well. It will also decrease the probability of a majority of voters being correct.
However, diversity decrease did not affect on the 3-NN classification performance. Inversely, 3-NN had the highest accuracy when bag size was equal to 40%. It can be said that 3-NN classification performance had an improvement by decreasing the bag size or decreasing diversity. When bag size decreased, which means that each bootstrap subset would contain less training examples in 3-NN. It might cause overfitted.

(c) Utilizing 10 fold cross validation to evaluate the performance of J48 and 3-NN classifiers by applying Boosting ensemble strategy, the following is presenting the accuracy as the ensemble size increase:

Boosting strategy	J48	3-NN
Ensemble Size	Accuracy	
20	75.964 %	69.4087%
40	76.8638%	69.4087%
60	78.4062%	69.4087%
80	78.5347%	69.4087%
100	78.5347%	69.4087%



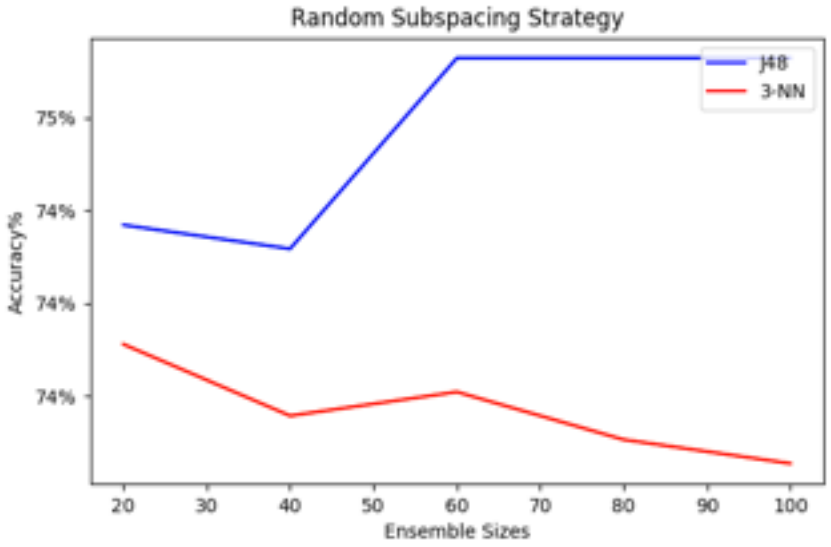
It is interesting to notice that using Boosting ensemble strategy, the accuracy of 3-NN is stable with the same value no matter how the ensemble sizes changed. In other words, there was no improvement for 3-NN by using Boosting ensemble strategy. One of reasons was that principle of Boosting method is to train model in advance and then next classifiers would focus on the errors the last one made. However, k-NN is kind of lazy learning classification which does not train model in advance. So, it can see that Boosting strategy is not suitable for k-NN classifier.
On the other hand, the accuracy for 3-NN is lower than J48 classifier in any ensemble size. Secondly, there is an increase trend in J48 classifier. It reached a peak with 78.5347% accuracy with ensemble size = 80. After that, accuracy values keeps steady. Therefore, Boosting ensemble strategy improved the J48 classification performance over a single tree. Boosting strategy is useful for Decision Tree in this data set.

(d) Utilizing 10 fold cross validation to evaluate the performance of J48 and 3-NN classifiers by applying Random Subspacing ensemble strategy, the following is presenting the accuracy as the ensemble size increase:

Random Subspacing strategy	J48	3-NN
Ensemble Size	Accuracy	
20	74.4216%	73.7789%
40	74.2931%	73.3933%
60	75.3213%	73.5219%
80	75.3213%	73.2648%
100	75.3213%	73.1362%

Using Random Subspacing ensemble strategy, both J48 and 3-NN classification performance had an improvement over single a classifier used in the data set. By compared with the two classifiers' accuracy, it was seen that the J48 had the higher accuracy than 3-NN no matter what ensemble sizes had. What's more, it can be noticed that the accuracy difference between J48 and 3-NN was not big gap. Because Random Subspace method only selected some features not all features to represent the training data, it was good for k-NN classifier. Meanwhile, Random Subspace method encourages diversity in the ensembles, which make accuracy higher than a single classifier. So, Random

Subspace strategy is also useful for Decision Tree in this data set. It also works well for K-NN classifier.
According to the below graph for each classifier, It can be seen that there was an increase trend in J48 along with increase ensemble sizes. And the accuracy had the peak with 75.3212% in the ensemble sizes =60, after that, the accuracy kept stable with increasing ensemble sizes. However, there was a decrease trend in 3-NN with increasing ensemble sizes. Ensemble size was 20 had the best performance for 3-NN.

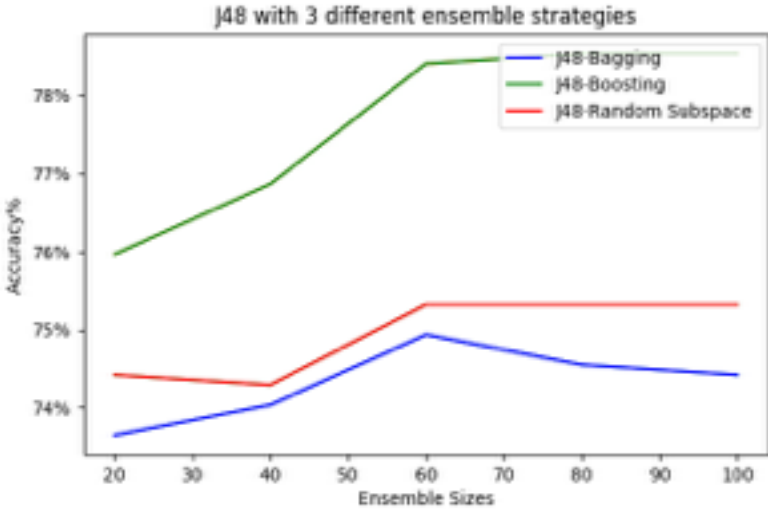


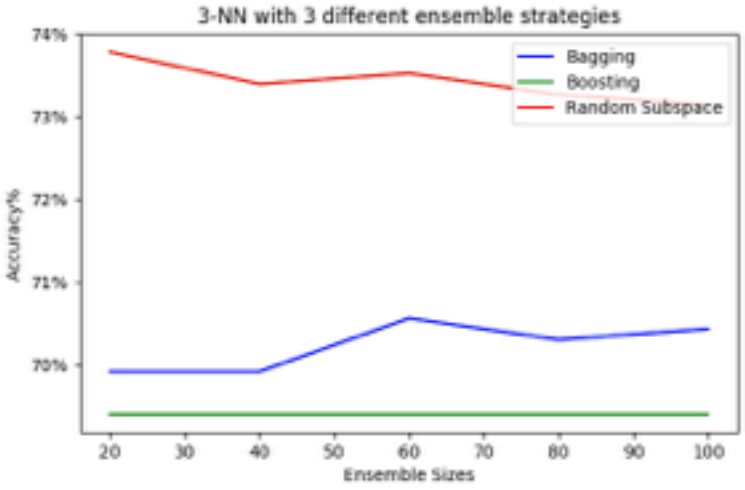
Using the best performing ensemble size 60 for J48, ensemble size 20 for 3-NN and evaluating the performance by using 10 fold cross validation , the accuracy for both classifiers in different bag size in the below:

Random Subspace with ensemble size 60 J48		Random Subspace with ensemble size 20 3-NN	
Different subspace size percent	Accuracy	Different subspace size percent	Accuracy
100	60.5398%	100	58.9974%
80	75.3213%	80	73.5219%
60	74.8072%	60	73.6504%
50	**75.3213%**	**50**	**73.7789%**
40	74.9357%	40	72.365 %
20	72.108 %	20	69.7943%

From the above table, it can be noticed that both J48 and 3-NN had the best performance when subspace size was 0.5. When subspace size was too high or too low, the accuracy was not good for both classifiers. Because when subspace size was too low, it only might use some not important features to training the model. It would cause under fitting. When selected the all features to represent the training data set, it might include some noise data. And each feature subset is more similar, which get less diversity.

(e) Summarizing differences: firstly, In the task (a), by initially analyzing my data set, it found that Decision Tree would be better than 3-NN. Meanwhile the accuracy of a classifier J48 was higher than a single 3-NN classifier. And then from task (b) to (d) by using different ensemble strategies to evaluate both classification performance, it also proved that the accuracy of Decision Tree was higher than 3-NN.
Secondly, using three different ensemble strategies, J48 had the improvement performance in all methods over a single classifier. But each ensemble strategy had the different performance on J48. In 3-NN classifier, Boosting strategy did not improve performance which kept the same performance with a single classifier. Bagging and Random Subspace methods both improved its performance.
The following graphs was the each strategy accuracy of J48 and 3-NN:





From the above graphs, it is clearly to see that using Boosting strategy had the best performance for J48. However, 3-NN had the best performance by using Random Subspace method.
The factors might explain the above differences:

(1) For the first and second differences, one of factors influenced it was the dataset itself. In the task(a), the origin data set has been analyzed and found that Decision Tree would a better classifier for it. Another factors may explain this difference may be about the classifier algorithm. 3-NN is to find the 3 nearest neighbours around input variable. Bagging does not work well for stable classifier; Random Subspace method works well for 3-NN, but it also useful for Decision Tree. k-NN is lazy learning classification strategy, boosting need to train a classifier on the example and then find the misclassified example. So, Boosting is not suitable for it.

(2) The 3 different ensemble strategies had the different performance in J48 classification. The factors might explain it are that diversity and accuracy. Boosting encourages diversity and accuracy and it also focus on misclassified examples. Bagging also encourages diversity, but it select example with replacement which might cause get less diversity when ensemble sizes reach certain point.

Question 2

(a) From Wikipedia[1] definition of Curse of dimensionality, there are many domains referred to it such as numerical analysis, sampling, combinatorics, machine learning, data mining and databases. The common theme of those problems is that as the number of features or dimensions grows, the amount of data needed to generalize accurately grows exponentially.
In machine learning domain, each input feature can be considered a dimension of a multiple dimension space. Along with increasing dimensions, the instances becomes sparse and dissimilar in multiple dimensions. In other words, some points that might be similar may have very large distance. All points will be far away from each other. As it said before that when increasing dimensions, the distance between points become far away and points become dissimilar. For example, K-nearest neighbor classification used distance function to find the nearest neighbors around input variable. It might effect on k-NN. Therefore, K-NN is suitable for lower dimensions. Using K-NN for high dimensions data, it may not perform as well as other techniques.
In order to deal with the problem of curse of dimensionality, we need to reduce dimensions in data set. There are two techniques for reduction dimensions: Feature Transformation and Feature Selection. Feature Transformation is to transform the original features of a data set into a completely new, smaller and more compact feature set and then retaining as much information as possible. There are two strategies for Feature Transformation: PCA and LDA. Feature selection is to select the best feature set of all available features which contains the smallest features that generate the highest accuracy for the data set. There are also two strategies for it: Filter and Wrapper. Filter method is to use Information Gain Score to rank the all features and then select the subset of features according to a classifier accuracy. Wrapper is initially to find the top rank features based on a classifier. Therefore, Filter is suitable for large of features which does not consider classifier initially. Normally, Wrapper method has a better performance than Filter.

[1] En.wikipedia.org. (2017). Curse of dimensionality. [online] Available at: https://en.wikipedia.org/wiki/Curse_of_dimensionality [Accessed 28 Nov. 2017].

(b) The bias-variance tradeoff generally appeared into supervised learning. All the supervised learning algorithms analyze the training data and then produce an inferred function. A good model not only need to capture all the patterns in the training data, but correctly computes the output for an unseen data. The more complex the model is, the better it represents the training data. But it might be a variance to be occurred because it includes noise in the data. High variance means the algorithm produces a model that is too specific to the training data, which is called overfitting. If is not complex, it may miss out some important features in the data. High bias means that the algorithm did not select important features from the data, which is called under-fitted. It is not possible to minimize both these errors simultaneously. High bias means low variance and low bias means high variance.This is exactly what the bias-variance tradeoff.
In ensemble classification, we generates several training data sets and then applies a base classifier algorithm for all training data. Each training data can build a different model. Each model generates a different output.The algorithm has a high variance error. Each model systematically produces the same incorrect output. This algorithm has a high bias error.
Bagging refers to training the same classifier algorithm multiple times on different training data set, which are randomly sample with replacement from the training data set. Each training date produced a model. All models are aggregated at end and then by majority voting or weighted voting

to generate output, which lead to a lower variance and remain bias compared to an individual model.

(c) Logistic Regression is better for regression with categorical features and deals with the problem of estimating probability. It has the range between 0 and 1. The following is logistic function:

$$P(Y=1) = \frac{\exp^{\beta_0+\beta_1 X}}{1+\exp^{\beta_0+\beta_1 X}}$$

It can be seen that logistic function is not a linear. But we can make it with linear with the help of logit transformation. After transforming, the logistic function formula: log(P(Y=1)/1-P(Y=1))) = β0 + β1*X(X is for features or predictors) . Odds ratios is equal to (P(Y=1)/1-P(Y=1)). Coefficients parameters are β0 and β1. When the odds ratio is greater than 1, it describes a positive relationship with the target feature. When the odds ratio is less than 1, it means that there is a negative relationship. β0 is the log-odds of Y=0; β0 + β1 is the log-odds of Y=1. A one unit changes in X is associate with a β1 change in log-odd of Y=1.
For example, Logistic function is: log(admitted/non-admitted) = -2.9013 + 0.00358 * GRE score. In this example, coefficients parameters β0 = -2.9013, β1= 0.00358; odd ratios is P(admitted)/P(non-admitted). Log(odd ratios) when GRE score=100 = -2.9013 + 0.00358 * 100; Log(odd ratios) when GRE score=101 = -2.9013 + 0.00358 * 101. So, the difference is equal to β1(0.00358). In term of odds = Exp(log(odd(101))-log(odd(101))) = Exp(0.00358) = 1.004. It means that an increase in 1 unit in the GRE score increase the odds of getting admission by 0.4%.

(d) PCA is an unsupervised learning linear transformation technique. It finds a direction of maximizing variance, which means that try to keep as much as of the variance in the data as possible. If some variables have a large variance and some has a small one, PCA will load on the large variances. The small ones will have little impact to dominating the principle component. In the other words, only selecting the large variance to represent the data. It might be misleading and cause dimension reduction inaccurate. PCA calculates a new dimensions of the origin features. The new axis are based on the standard deviation of the variables. Different variables in the data might be having different units of measurements. A variable with a high deviation will have a higher weight for the calculation of axis than a variable will have a low deviation. When normalizing all variables, they will have the same standard deviation. In other words, all variables have the same weight on calculation axis for PCA. So, it is necessary to normalize the data.