# Assignment 2

Luke Killoran – 13434418
Machine Learning (Blended) – COMP47460

Note that part (e) of question 1 is distributed across each section of question 1.

Q1.(a)
I ran the classification algorithms J48 and 3NN (using Euclidean distance) on the dataset, using 10-fold stratified Cross Validation to evaluate the performance of each.

| Figures Averaged Across 10-fold CV | J48 | 3NN | OneR | ZeroR |
|---|---|---|---|---|
| Total Accuracy | 72.8792% | 69.4087% | 50.7712% | 26.4781% |
| Precision (Averaged Across 4 Classes) | 72.9% | 69.2% | | |
| Recall (Averaged Across 4 Classes) | 72.9% | 69.4 % | | |
| F1-Measure (Averaged Across 4 Classes) | 72.9% | 69.3% | | |

It seems as though the J48 algorithm outperforms the 3NN algorithm, with a superior classification accuracy, complimented by larger precision and recall rates (and thus by extension a greater F1 measure). Both algorithms yield a much better classification accuracy that OneR and ZeroR, which was to be expected.

Note that Bagging, Boosting, and Random Subspaces all selected their models by using the whole dataset rather than the training portion of the dataset in each iteration of 10-fold Cross Validation.

Q1.(b) (i)
I executed the Bagging process in Weka. For each iteration of 10-fold Cross Validation, this consisted of sampling instances from the whole dataset with replacement. The size of each sample was selected as "Bag Size" which was displayed as a percentage of the size of the total dataset. Once a sample was taken, the classification algorithm was implemented using this set of instances, and subsequently a classifier was formulated. Then each instance of the fold in question (the test set) is classified using the aforementioned classifier. These predictions are recorded, another sample is drawn, and the process is repeated for a specified number of iterations ("NumIterations", which determines the number of samples also known as the number of bags). The predictions for each instance in the fold or test set are averaged across these classifiers. The weights attributed across the classifiers are distributed equally so that each classifier has an equal say. The class with the maximum averaged score is the final prediction, and the averaged prediction score is interpreted as the probability that that instance is characterised by the chosen class. Note that the classifiers were not weighted by out-of-bag performance. The reason for this decision was to better evaluate the process against the Boosting procedure introduced later.

Bagging was implemented for both algorithms in 10-fold Cross Validation for a bag size of 100, altering the number of bags each time.

| Classification Accuracy for Bagging (Bag Size of 100, figures averaged across 10-fold CV) | | |
|---|---|---|
| | J48 | 3NN |
| No. of Bags = 20 | 73.6504% | 69.9229% |
| No. of Bags = 40 | 74.0360% | 69.9229% |
| No. of Bags = 60 | 74.9357% | 70.5656% |
| No. of Bags = 80 | 74.5501% | 70.3085% |
| No. of Bags = 100 | 74.4216% | 70.4370% |
| No. of Bags = 300 | 73.0077% | 70.0514% |

In general, Bagging improved the classification accuracy for both algorithms over their original base learners since it allows the algorithms to encounter an instance on multiple occasions, and introduces variety in the data each model uses to form predictions. This is a recipe for success as an average of strong different predictions will almost certainly boast more accurate forecasts. The simple idiom "any committee is better than no committee" usually applies, and there is no exception here. Furthermore, as the number of bags grow, the predictions will be based on more models, hence, classification accuracy should improve. However, there is a point where new ensemble members will have prediction patterns collinear with existing members. No new diversity is added, so ensemble accuracy will plateau. This point occurs at 60 bags for both algorithms. In fact, increasing the number of bags in this case could actually decrease the prediction potential of the combination of classifiers, due to fitting too close to the data (overfitting). This can be seen in the poor generalisation of the J48 algorithm when 300 bags are used in combination. The difference between the performance of both algorithms with respect to their use in conjugation with Bagging can be summarised in that J48 improves much more than 3NN under Bagging. Since it is more unstable, each bag is more likely to generate different predictions of the test set than the models before. This extra diversity is the reason why a combination of J48 models on bagged samples benefits much more than a combination of 3NN models.

Q1.(b) (ii)
Bagging was implemented for both algorithms across 10-fold Cross Validation for 60 bags, altering the size of each bag each time.

| Classification Accuracy for Bagging (60 Bags, figures averaged across 10-fold CV) | | |
|---|---|---|
| | J48 | 3NN |
| Bag Size = 20 | 72.2650% | 74.9357% |
| Bag Size = 40 | 73.6504% | 71.7224% |
| Bag Size = 60 | 73.5219% | 70.3085% |
| Bag Size = 80 | 73.9075% | 70.6941% |
| Bag Size = 100 | 74.9357% | 70.5656% |
| Bag Size = 125 | 74.9357% | 70.5656% |
| Bag Size = 150 | 74.9357% | 70.5656% |
| Bag Size = 175 | 74.9357% | 70.5656% |
| Bag Size = 200 | 74.9357% | 70.5656% |
| Bag Size = 250 | 74.9357% | 70.5656% |
| Bag Size = 300 | 74.9357% | 70.5656% |
| Bag Size = 500 | 74.9357% | 70.5656% |

Altering the bag size leads to very different (even opposite) effects on the classification accuracy of both algorithms. For J48 the greater the bag size, the better the prediction. This is because a larger bag size provides more data to make predictions – this enhances the quality of the predictions. As well as that, due to its instability the algorithm still maintains enough diversity across its predictions despite samples inevitably being more similar as they increase in size. In contrast, since the 3NN is stable, the growth of instances in the bags diminishes the classification accuracy because the improvement in quality of predictions is far outweighed by the sheer decrease in the diversity of predictions. Note that the classification accuracy does not change for either classification algorithm when the bag size increases beyond 100%. This could be because any extra instances are relatively collinear with the instances in the data set already, so predictions are relatively unaffected.

Specifically, the opposite pattern occurs for the 3NN algorithm when expanding the bag size from 60 to 80. This is because a reduction in diversity is offset by an improvement in the accuracy of the predictions due to more data in each sample. Similarly, the trend for the J48 algorithm is broken when the bag size is enlarged from 40 to 60 when more accurate predictions (due to more instance in each bag) are nullified by a significant decline in diversity.

The ideal models are the Bagging process consisting of 60 bags, each of size 100% of the whole dataset, paired with the J48 base learner or, with equal accuracy, the Bagging procedure consisting of 60 bags, each of size 20% of the whole dataset, paired with the 3NN base learner.

Q1.(c)
I executed the Boosting process in Weka. For each iteration of 10-fold Cross Validation, this consisted of sampling instances from the whole dataset with replacement. The probability of an instance being chosen at any one point is its weight divided the sum of the weights of all instances. Initially, every instance is assigned

an equal weight (1 / total number of instances = 1/778). Once a sample was taken, the classification algorithm was implemented using this set of instances, and subsequently a classifier was formulated. Then each instance of the fold in question (the test set) is classified using the aforementioned classifier. These predictions are recorded, and the classification error ($\varepsilon$) is calculated. If the classification error was greater than 50%, then the model was discarded and another sample was generated. The value Beta is computed as $\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$. Then each instance is reweighted according to the following conditions: If the instance was correctly classified then its new weight was given by $w_{new} = w_{old} \cdot \left(\frac{1}{2}\right)\left(\frac{1}{\varepsilon}\right)$, and if the instance was incorrectly classified then its new weight was given by $w_{new} = w_{old} \cdot \left(\frac{1}{2}\right)\left(\frac{1}{1-\varepsilon}\right)$. (Then all the weights are normalised so they add to one.) This step is so as to weight the incorrectly classified instances more, and concentrate the samples with the instances often forecasted inaccurately. This is all in the hope that a classification can improve by focusing on its errors. Another sample is drawn, and the process is repeated for a specified number of iterations ("NumIterations", which determines the number of samples). The predictions for each instance in the fold or test set is averaged across these classifiers. The weights attributed across the classifiers' predictions are distributed in accordance with the size of the Beta value calculated at each step. More precisely, given an instance in the test set, a vector is computed with each entry representing the weight attributed to a class. The entry corresponding to class i is the sum of the Beta values of the classifiers which predicted the instance with class i. Then the max entry is subtracted from every entry. Following this, the exponential function is calculated with each element as its power. Then the whole vector is normalised to sum to 1. The class with the largest score is the final predicted class for that instance, and the associated score is deemed the probability that this instance is characterised by the selected class.

Boosting was implemented for both algorithms across 10-fold Cross Validation, altering the number of iterations / samples each time.

| Classification Accuracy for Boosting (figures averaged across 10-fold CV) | | |
|---|---|---|
| | J48 | 3NN |
| No. of Iterations = 20 | 75.9640% | 70.4370% |
| No. of Iterations = 40 | 76.8638% | 70.4370% |
| No. of Iterations = 60 | 78.4062% | 70.4370% |
| No. of Iterations = 80 | 78.5347% | 70.4370% |
| No. of Iterations = 100 | 78.5347% | 70.4370% |

Note that any number of iterations for boosting improves both classifiers over their original base learners. This is because again introducing a committee (with moderately accurate prediction and some diversity) trumps no committee. However the trends between the two classification algorithms are very different. Due to 3NN's stability as a classifier its predictions remain stagnant even if the samples are altered in their composition. With Boosting, this effect is magnified since Boosting issues much larger weights to those instances that are forecasted incorrectly. This weighting scheme depletes each subsequent sample of its diversity, especially because the 3NN will continue to misclassify these observations. Thus, a vicious cycle is present. For this reason, it is no surprise that the process terminates after 3 iterations. The predictions of these instances do not change despite concentrating the sample with them, and the greater proportion of incorrectly classified instances inevitably leads to a classification accuracy below 50%, and so the classifier is so weak that it cannot be used in the final combination. For 3NN, Boosting is worse than Bagging because Bagging allows more diversity in the classifier due to sampling randomly across the dataset rather than sampling mostly from the same incorrectly classified instances. Note even increasing the no. of iterations to 300 does not change the combined classifier predictions as the 3NN algorithm will always terminate after 3 iterations for these splits of the data. If not for the constraint to discard any classifier with less than 50% classification accuracy, the prediction potential of the combined 3NN classifiers would worsen with each iteration.

In contrast, the J48 algorithm never terminated in all its runs (even operating at 300 iterations). This is because J48 is an unstable classifier, so despite incorporating more similar instances (that were difficult to class) in each sample, the classifiers will still differ in its predictions. Hence the negligible drop in diversity. Although each subsequent classifier tends to decrease in classification accuracy, some previously misclassified instances are correctly predicted and as a result classification accuracy from the combination of classifiers tends to increase.

This is why Boosting actually tends to drastically improve the performance of the J48 algorithm over Bagging. Bagging ensures more diversity in the bags due to its random sampling across the dataset. However, J48's instability allows it to withstand a drop in the diversity of the samples, generating a variety of predictions despite similar samples. Moreover, its ability to adapt and learn from its errors cements it as a perfect companion with Boosting. Even after 299 iterations of classifying 'difficult' instances, the classification accuracy of the classifier can still predict the class of 50% of the test set. Note however, at 300 iterations the classification accuracy decreased. This is because the combination of classifiers began overfitting to the data.

The ideal model is the Boosting process consisting of 80 samples, paired with the J48 base learner.

Q1.(d) (i)
I executed the Random Subspace process in Weka. For each iteration of 10-fold Cross Validation, this consisted of sampling attributes from the whole set of 19 attributes. The size of each sample was selected as "Subset size" which was displayed as a percentage of the size of the total number of attributes (19). Once a sample was taken, the classification algorithm was implemented using this set of attributes, trained on the split dataset, and subsequently a classifier was formulated. Then each instance of the fold in question (the test set) was classified using the aforementioned classifier. These predictions were recorded, another sample was drawn, and the process was repeated for a specified number of iterations ("NumIterations", which determines the number of subsets of attributes). The predictions for each instance in the fold or test set are averaged across these classifiers. The weights attributed across the classifiers are distributed equally so that each classifier has an equal say. The class with the max averaged score is the final prediction, and the averaged prediction score is interpreted as the probability that that instance is characterised by chosen class. Note that the classifiers were not weighted by out-of-bag performance, but this could have been implemented.

Random Subspaces were implemented for both algorithms across 10-fold Cross Validation, each subset with 50% of the original set of attributes, altering the number of iterations / subsets each time.

| Classification Accuracy for Random Subspaces (Subset Size = 50%, figures averaged across 10-fold CV) | | |
|---|---|---|
| | J48 | 3NN |
| No. of Iterations = 3 | 71.4653% | 71.4653% |
| No. of Iterations = 5 | 73.9075% | 71.9794% |
| No. of Iterations = 10 | 75.0643% | 73.5219% |
| No. of Iterations = 15 | 74.9357% | 73.0077% |
| No. of Iterations = 18 | 74.4216% | 73.3933% |
| No. of Iterations = 20 | 74.4216% | 73.7789% |
| No. of Iterations = 40 | 74.2931% | 73.3933% |
| No. of Iterations = 60 | 75.3213% | 73.5219% |
| No. of Iterations = 80 | 75.3213% | 73.2648% |
| No. of Iterations = 100 | 75.3213% | 73.1362% |
| No. of Iterations = 200 | 75.3213% | 73.5219% |

Random Subspaces using half the attributes improves the classification accuracy for both algorithms for any number of iterations over the original base learners. This is because again introducing a committee (with moderately accurate prediction and some diversity) trumps no committee.

As the number of iterations grew from 3 to 10, both classification algorithms performed much better. This is because there is much less variability in 10 predictions for each instance as opposed to 3. However, apart from this trend the number of iterations didn't heavily impact the classification accuracy, with very little variation in classification scores. It seemed as though a greater number of iterations improved the classification accuracy for the both algorithms in general. However, even though it was unlikely to encounter the same subset of variables in one run (as there are 92378 possible subsets of 10 attributes to choose from the original 19 features), it seemed as though each algorithm reached a certain classification level after which it plateaued. Unsurprisingly, 3NN reached this level much sooner than J48, as the J48 algorithm is more unstable, so its predictions are less likely to repeat. The reason for the classification accuracy to plateau is because eventually even different subsets of attributes will lead to similar models from before, and predictions will be similar to previous ones.

Q1.(d) (ii)
Random Subspaces were implemented for both algorithms across 10-fold Cross Validation, each subset with 50% of the original set of attributes, altering the number of iterations / subsets each time.

| Classification Accuracy for Random Subspaces (60 and 20 Iterations respectively, figures averaged across 10-fold CV) | | |
|---|---|---|
| | J48 | 3NN |
| Subspace Size = 0.1 | 67.9949% | 66.4524% |
| Subspace Size = 0.2 | 72.1080% | 69.7943% |
| Subspace Size = 0.3 | 73.7789% | 73.1362% |
| Subspace Size = 0.4 | 74.9357% | 72.3650% |
| Subspace Size = 0.5 | 75.3213% | 73.7789% |
| Subspace Size = 0.6 | 75.3213% | 73.6504% |
| Subspace Size = 0.7 | 75.0643% | 73.2648% |
| Subspace Size = 0.8 | 75.3213% | 73.5219% |
| Subspace Size = 0.9 | 73.6504% | 71.4653% |

Both algorithms displayed similar trends with regards to selecting the number of attributes in each subset, and its effect on classification accuracy. Classification accuracy tends to improve drastically as subset size increases from 10% to 50%, as the predictions are substantially more accurate when each classifier is constructed using more features (for example considering 10 attributes compared with 2 attributes). However, the classification accuracy then diminishes thereafter. Incorporating more attributes in each subset does not add much predictive power, but considerably shrinks the diversity across predictions.

Random Subspaces performed slightly better than Bagging surprisingly but much worse than Boosting for the J48 algorithm. Just as bagging introduces more diversity, Random Subspaces do the same (if not more) but due to a lesser amount of attributes each J48 decision tree is more parsimonious ensuring better generalisation and thus increased predictive power. Furthermore, even though Boosting decreases the diversity, the algorithm is so unstable that is still manages different predictions across classifiers. This means that the drop in diversity is insignificant compared with the benefits of focusing the algorithm on correcting its errors.

Random Subspaces performed better overall than both Boosting and Bagging for the 3NN algorithm. This is because the algorithm is so stable that sampling different instances does not affect the predictions much, so it is necessary to greatly improve the diversity by sampling different attributes. This ensures the committee of 3NN classifiers have disagreement, thus improving the final predictions. Although it must be noted that the best classification accuracy for the 3NN algorithm  came from bagging 60 times with bag sizes of 20% of the dataset. This introduced around the same amount diversity in predictions as Random Subspaces but also allowed the classifier use of all its attributes to make better predictions.

The ideal models to use when adopting Random Subspaces consist of 60 subsets, each containing half the original attributes, paired with the J48 base learner, or 20 subsets, each containing half the original attributes, paired with the 3NN base learner. The best model for the dataset is, by far, Boosting with 80 iterations using the J48 decision tree classifier. This is the most powerful model as it combines the ability to improve its flaws whilst generating enough diversity to improve the overall committee predictions. Its precision and recall has boomed to 78.2% and 78.5% (and so an F1-Measure of 78.3%). Moreover, the area underneath its ROC dwarfs the original base learners at 0.993 (averaged across all 4 classes).

Q2.(a)
The Curse of Dimensionality refers to how many learning algorithms can perform poorly on high-dimensional data (Bellman, 1961). In practice, this means that, for a given number of examples, there is a maximum number of features beyond which the performance of a classifier will degrade rather than improve. Intuitively adding more features to a dataset should provide more information about each example, making prediction

easier. In reality, we often reach a point where adding more features no longer helps, or can even reduce predictive power. Thus this generates the problem that the number of examples required per feature increases exponentially with number of features. High-dimensional spaces tend to be very sparse, so every point is equally far away from virtually very other point, and distances tend to be uninformative. In other words, the more dimensions you have, the more similar examples appear to one another. This why with a large number of features, we may require an inordinate amount of instances so there are several samples with each combination of values. Moreover, with a fixed number of training samples, the predictive power reduces as the dimensionality increases, and this is known as Hughes phenomenon. There are three main methods to deal with these problems. The first is simply to obtain more data. However this obviously may not be feasible physically or financially. The second technique is Feature Transformation; where the original set of attributes is transformed into a much smaller, compact feature set while retaining as much predictive power as possible. Examples of Feature Transformation include Principal Components Analysis (which uses the eigenvectors of the centred covariance matrix (of the data) to represent the new features), and Linear Discriminant Analysis (which finds a linear transformation of the attributes that maximises the between-class variance and minimises the within-class variance). The third technique is Feature Selection (whether by filters or wrappers); where we simply select a subset of the original features (that optimises certain criteria) to use for prediction. Examples of Feature Selection include filters such as the Information Gain Filter (which ranks each attribute so as to assign higher ranks awarded to those attributes which reduce the entropy of the instances the most, given a one branch decision tree along that feature alone – the top ranked attributes are selected), and wrappers such as the Naive Bayes Wrapper with sequential forward selection (here the training set is split and starting from models with 0 attributes, Naive Bayes models are evaluated, adding a new feature at each iteration, with each feature being the one that reduces classification error the most given all other features are already incorporated into the model).

Q2.(b)
The Bias-Variance trade-off is the dilemma posed to Machine Learning algorithms in which one must balance a model which captures the regularities in the training data (low bias), with one which generalises well to new unseen data (low variance). Unfortunately, both ideals cannot be optimised simultaneously, so a decision must be made. High bias means that the real target function values of the instances are very different from their model predicted values. We can almost never evaluate bias directly as we can never know the real target function, we only know the combined result after random noise. High variance means that the model predicted values of the instances are extremely sensitive to the underlying training data. In other words, a slight alteration to the composition of the training set used to construct model could lead to very different predictions for the same instances in the test set. Variance cannot be measured directly as changing the underlying data alters the model itself. However, a proxy for both measures is the number and restrictive power of assumptions used in constructing the model. The general rule of thumb is that the greater the amount and strength of the assumptions (or complexity), the greater the bias but the lesser the variance. For example, a 1NN algorithm is extremely flexible so its assumptions are minimal. This means that it could fit the target function well no matter what the characteristics of that function are, and so it exhibits low bias. However, the 1NN algorithm is extremely sensitive to the underlying training data, and thus would be subject to high variance. On the other hand, a linear regression model would be a very poor fit for any target function that expresses non-linearity, and as such, the model would display high bias. Although, the predictions from the linear model are unlikely to be very different even when some instances are substituted in or out, and so the model would boast low variance. Parametric models tend to have more assumptions built in, and so they probably are associated with high bias and low variance. Although it depends on the nature of the data. If the data have similar characteristics than those predicted than the model will inevitably have low bias. What matters is not bias and variance individually but their sum (as is measured in the Mean Squared Error along with irreducible error). Note high-bias, low-variance algorithms tend to underfit, but low-bias, high-variance algorithms tend to overfit. Bagging allows us to sample from the same dataset many times, as opposed to simply building the classifier once on the full dataset. If the predictions are weighted equally, it is unlikely Bagging will reduce the bias of the algorithm greatly even though there is a committee with usually diversified opinions. However if the predictions are weighted by performance, perhaps this will reduce the bias slightly, as the classifiers which can generalise the best will be assigned more power. This, paired with a diversified committee may allow the combination of classifiers to predict more closely the target function. Although, this effect is thought to be negligible as bagging does not significantly increase the flexibility of the model in general. On the other hand, since bagging provides a (usually) diversified committee of predictions, the

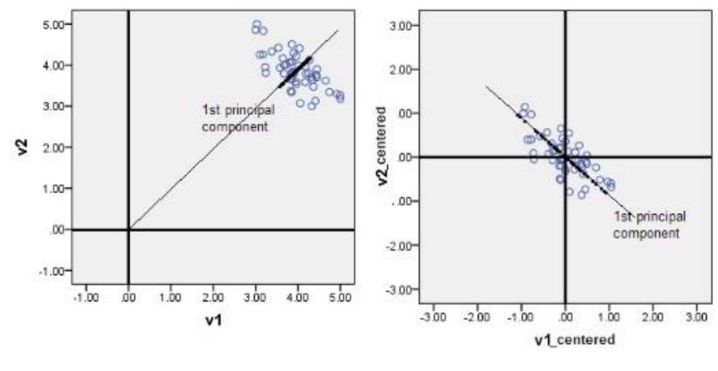
variance of the model's predictions is dramatically reduced as a change in the underlying data is less likely to alter the combined classifiers predictions as much as one initial classifier.

Q2.(c)

$$P(Y=1)=\frac{e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}}{1+e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}} \text{ or } log\ odds\ ratio = \log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0+\beta_1X_1+\cdots+\beta_pX_p$$

So in the case with only one predictor $X_1$, $P(Y=1)$ appears as an S-shaped curve for various values of $X_1$ (when $\beta_1$ is positive, and a backward S-shaped curve when $\beta_1$ is negative). However, the log odds ratio appears as a line for various values of $X_1$ with intercept $\beta_0$ and slope $\beta_1$ . Thus extending this gives us that $\beta_0$ represents the mean odds ratio for Y when $X_1 = 0$ . If $\beta_1$ is positive, an increase in $\beta_0$ slides the P(Y=1) curve left, and a decrease, right. If $\beta_1$ were negative, the opposite occurs. The reason for this is that a greater value of $\beta_0$ means that $P(Y=1)$ is increased for all values of $X_1$ (unless it already was equal to 1) no matter the nature of $\beta_1$. Thus, for the case when $\beta_1 > 0$: given an $X_1$ value (call it $x_1$) which provides a certain probability of $Y=1$ (call it $p_1$) initially, the $X_1$ value (call it $x_2$) which would retain the same level as the aforementioned probability of $Y=1$ ($p_1$) is inevitably lower than the previous $X_1$ value ($x_1$) after an increase in $\beta_0$ (or the mean odds ratio when $X_1 = 0$). This ensures the curve slides left. If $\beta_1 < 0$ then $x_1 < x_2$, and so the curve slides right. The opposite case holds for each when $\beta_0$ declines. Of course, we have not explained granularly what exactly does $\beta_1$ represent. $\beta_1$ is the amount by when the log odds ratio of Y changes when $X_1$ grows by one unit. $\beta_1$ also controls the slope of the S-curve, the larger the value of $\beta_1$, the steeper the curve. This is because when $\beta_1 > 0$, and it grows larger (to say $\beta_1^l$ ), then $for\ any\ x > 0 \in X_1,\ \beta_1 x < \beta_1^l$ x $\ and\ \ for\ any\ x < 0 \in X_1,\ \beta_1 x > \beta_1^l$ x, and so the curve gets more extreme (steep) either side of $X_1 = 0$. For $\beta_1 < 0$, the same phenomenon occurs. Another question to ask is when does $\beta_1$ tend to be positive, and when negative? $\beta_1$ tends to be greater than 0 when larger values of $X_1$ correlate with proportionally more occurrences of Y=1. Larger values of $\beta_1$ necessitate smaller changes in $X_1$ to alter some Y values to change from class 0 to class 1. $\beta_1$ tends to be less than 0 when the opposite occurs. $\beta_1$ tends to be close to 0 when $X_1$ does not correlate strongly with Y. If we increase the value of $X_1$ by one unit (from $x\ to\ x+1$), the $\beta_1$ is added to the previous log odds ratio, and equivalently the previous odds ratio is multiplied by a factor of $e^{\beta_1}$. Each of these increases are independent of the size of $x$, the original starting point. However, the effect on the probability that Y=1 is not so easy to entangle. The new probability is given by $P(Y=1) = \frac{e^{\beta_0+\beta_1(x+1)}}{1+e^{\beta_0+\beta_1(x+1)}} = \frac{e^{\beta_0+\beta_1 x}\, e^{\beta_1}}{1+e^{\beta_0+\beta_1 x}\, e^{\beta_1}}$. Thus, to understand the change in $P(Y=1)$, we must know its previous value in the context of $x$. If $\beta_1$ was set to 0, then $X_1$ would have no effect on the outcome of the predicted probability $P(Y=1)$. The log odds ratio would be flat in $X_1$, and the $P(Y=1)$ S-curve would become a flat line at $\frac{e^{\beta_0}}{1+e^{\beta_0}}$.

Q2.(d)
In linear regression, we fit a model with an intercept and slope (for each variable X). However, the results would be equivalent if we centred Y and the X's. This would ensure the line goes through the origin. The principal components are effectively regression models without an intercept. Thus, it would be senseless to fit a linear model constrained with no intercept to uncentred data. If we fit PCA to data that is uncentred, it is likely the first principal component will correspond to the mean of the data rather than the direction of maximum variance. A mean of zero is needed for finding a basis that minimizes the mean square error approximation of the data. I found the figure below which demonstrates this very point. Moreover, I would recommend normalising the data to have 0 mean and unit variance before conducting PCA (use autocorrelation matrix). This is because, if the attributes have different scales, the features with the more wide-reaching ranges tend to dominate the direction of the principal components. For instance, usually the variance around the mean of a variable with range [0,1000000] is much greater than the variance around the mean of a variable with range [0,1], even if the latter attribute is varies much more relatively. Thus, in the case, the principal components will be chosen to minimise the variance around the attributes with the largest scales (which seems arbitrary), even though the focusing on the more relatively variable feature would capture more of the variation so that the principal component would be more predictive of the response variable in our final model.


1st principal component
v2
v1
1st principal component
v2_centered
v1_centered