COMP47460: Assignment 1

Name: Conor Murphy
Student Number: 20205251

Question 1

Data cleaning and Transformation

Data cleaning and transformation is a massive step in any machine learning algorithm and if not done can lead to inaccurate analysis. The first step was to remove unnecessary columns, The first column that was removed was the area_id, Initially I was unsure if this needed to be removed as I believed it could indicate grouping by area however after further inspection I was clear that it was just a unique identifier, as this identifier does not add any predictive value to the machine learning algorithm and instead could even result in overfitting of the model, for this reason it was removed. The Next step was to min-max normalise the data, this was achieved in weak using the normalise feature in the pre-process stage and setting the scale to one. As all values present in the data set are continuous min-max normalisation was necessary as it removed some larger values being assigned a higher weight in the model. This normalisation is extremely important for distance based measurement like in KNN which we will see later. Once I started the feature selection, I noticed that the data set contained features for h-nutrients_weight and h_nutrients_weight_norm this was also the case for h_nutrients_calories, here the data set contained normalised columns alongside their actual value, however since I just normalised the data so these features are essentially duplicates and so It was necessary to remove one of them, I also noticed that in both instance they matching columns had the exact same information gain, duplication in features is not desired so I removed the pre normalised values.

Part A

The Tesco data is rather large and contains 77 features, this number is excessively large and using all of these features could lead to the curse of dimensionality being encountered, where the extra features negatively affects the predictive power of the model, to avoid this adequate feature selection is required.

The first Feature selection method was the filter algorithm using information Gain as the score of the predictiveness the top results are shown in Table1, to know the optimal cut-off point for the filter algorithm can be difficult to determine so instead we generally chose a heuristic measure, in this case I chose to use features that had which is greater than 50% of the top feature which in this case is h_nutrients_weight as shown in table 1.1

IG	Feature
0.539	h_nutrients_weight
0.4795	h_nutrients_calories
0.4597	f_grains_weight

Table 1.1

The next measure to be tested was the wrapper algorithm for the j48 decision tree, initially I had planned to use backwards elimination for all of my wrapper algorithms however here I encountered extremes compute time of upwards of 15 minutes for the attribute selection, this seemed excessively long so I decide to opt for the forward sequential search for this measure, the results are available in table 1.2

Continuing on I then ran the wrapper algorithm for the naive baye classifier, once again using forward feature selection. Finally, I ran the wrapper algorithm for the knn classifier with k = 3. As

with the j48 and naïve algorithms the compute time for backward elimination was excessively long so I opted for forward sequential search

Filter	wrapper(j48)	wrapper(Naïve)	Wrapper(knn)
h_nutrients_weight	h_nutrients_weight	energy_carb_std	f_grains_weight
h_nutrients_calories	volume	volume	volume_std
f_grains_weight	f_energy_fibre	f_energy_sugar	volume
f_wine	carb	f_sauces_weight	weight_std
energy_carb_std		f_meat_red	f_grains
carb_std			energy_tot
f_energy_carb			alcohol
volume_std			f_energy_fat
volume			f_fish
f_energy_protein			fibre_std
f_energy_fibre			fat_std
carb			energy_fat_std
energy_carb			
protein_std			
energy_protein_std			
weight_std			
f_grains			
f_fruit_veg			
energy_tot_std			
f_energy_alcohol			
f_dairy_weight			
weight			

Table 1. 2

A few interesting thing to note from the above table is that the filter method is the largest of all four sets, however this is due to the fact that I chose the cut-off point, so if desired I could easily just change the threshold and reduce the number of features. Due to the j48 wrapper algorithm using forward feature selection it only has chosen four features to use. I noticed that the Naïve bayes wrapper relies heavily on features with f_ meaning fraction of products of this type. All three wrapper method rank volume high in their order. It is also no surprise to see the wrapper method for j48 having the same first feature as the filter as both prioritise based on information gain.

Part b

The main evaluation metrics I plan on using are the accuracy of the classifier, this is useful as it indicated how good the model is at classifying the data, this measure will not however be used alone as it can lead to a false sense of how good the model is and can be heavily influenced by overfitting. To compensate for this, we can also examine the precision and recall of the classifiers, the best measure to assess balance accuracy issues is the F-measure, this will indicate the trade-off between precision and accuracy. To ensure that we also do not encounter overfitting all accuracy measures will be generated using 10 fold cross validation, this form of validation is discussed in more detail in question 2. Finally, ROC curves will be used generated to determine the effectiveness of the classifiers.

Part C

Once the attribute selection was completed, I separated the selected features into separate arff files and ran then in Weka with their corresponding classifiers. While doing this I aimed to set the parameters to maximise the accuracy of the model, as mentioned earlier all evaluation measurements were done using 10 fold cross validation. For the j45 decision tree the confidence Factor was set to 0.25 this dictates the threshold allowed in the error of the data while pruning the tree. For the knn algorithm the optimal number for k was 3 and thus was used.

	j48	Naïve Bayes	KNN
correct	410	412	434
accuracy	79.4574	79.845	84.1085

Table 1.3

Shown in table 1.3 is the accuracy outcome of the three wrapper methods with their corresponding classifier. The worst but only by a minute margin is the j48, then Naïve bayes with only a tiny increase in accuracy, with the KNN being the most accurate.

Accuracy	j48	Naïve Bayes	KNN
Wrapper	79.4574	79.845	84.1085
Filter	76.5504	72.093	77.3256
All features	76.3566	72.4806	77.3256

Table 1.4

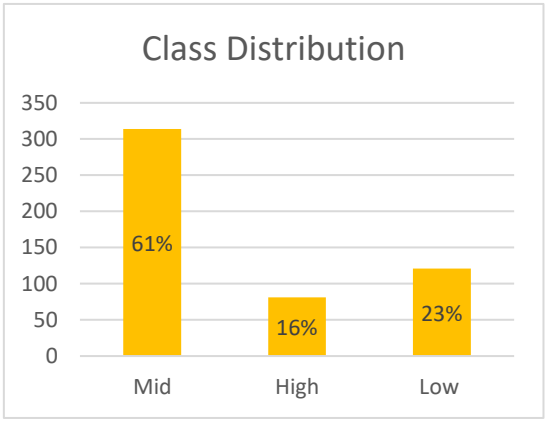
Shown above is the accuracy percentage of each classification method carried on different feature selections, so for example the KNN was run with the features of the wrapper algorithm for this classifier then the same thing was done with the filter section results form part A and finally the algorithm was run on the entire feature set.

Up to this point we have only been concerned with the accuracy of our data; this can however be misleading as I will discuss in question 2.

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
j48					
mid	0.901	0.371	0.791	0.901	0.842
high	0.543	0.018	0.846	0.543	0.662
low	0.686	0.058	0.783	0.686	0.731
weighted Avg	0.795	0.242	0.797	0.795	0.788
Naïve bayes					
mid	0.857	0.292	0.82	0.857	0.838
high	0.593	0.037	0.75	0.593	0.662
low	0.785	0.073	0.766	0.785	0.776
weighted Avg	0.798	0.201	0.796	0.798	0.796
KNN					
mid	0.898	0.248	0.849	0.898	0.873
high	0.79	0.039	0.79	0.79	0.79
low	0.727	0.038	0.854	0.727	0.786
weighted Avg	0.841	0.166	0.841	0.841	0.84

Table 1. 5

Shown above are the TP (True Positive), FP (False Positive), Precision, Recall and the F-Measure values for each classifier. The table indicates the evaluation measure for each class along with the weighted average for the entire classifier. The True positive rate is the proportion of positive items in the dataset that are correctly identified. The False positive value represents the items that are classified to a class that the do not belong to i.e. classifying someone as having a high chance of diabetes when in fact they do not. The measure precision and recall are also very important, recall is the ability to find all the relevant items in the dataset, in the context this is the ability of the model to correctly identify all the people who have a mild chance of diabetes, however in theory if the model just classified every one as having a mild chance of diabetes then our recall would be the desired maximum of one for this class, but obviously the recall of the other two classes would suffer greatly as a result. To combat that we can also use precision which is the measure of how accurate the model is at correctly identifying the person belonging to the mild class. There is a trade-off between precision and recall and we often need to decide which one we need to prioritise. Take the hypothetical situation where we would use this model to treat people who have a high chance of diabetes if we were to prioritise recall then it would result in many people being treated for diabetes who do not have it, if we were prioritising precision then there could be a lot of people who actually have a high chance but are not being treated. In many situations both the precision and recall are equally important so here we can use a measure known as the F1-Measure this is the harmonic mean of precision and recall, this can now be used to indicate a good balance between precision and recall, The higher the F-Measure the better. Using measure like precision recall and the f-measure may make more sense for this model as when we examine the class balance, we can see that the mid class represents 61% of the data, while the high class represents only 16%. This class imbalance can lead to the model being overfitted.



Part D

Form the data produces in table 1.4 it was not that big of a surprise that each wrapper method yielded a higher accuracy than the filter method, this was very in line with the expected outcome as the wrapper method selects features that will perform the best on specification classification algorithms. One other interesting difference I noted was how small of a difference there was between using the filter algorithm and just running the algorithm on the full feature set, in the case of the Naïve bay algorithm it actually had a higher accuracy than the filter method, this possibly calls into question the heuristic measure that I chose for the filter classifier, In retrospect a higher threshold should have been set on the filter method possibly greater than 65% of the top feature.

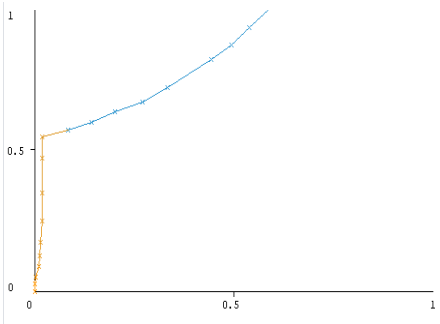
The knn algorithm performed the best in terms of accuracy and when looking at its weighted averages we can also see it performing better than the other two in most metrics overall. I had expected the performance of the j48 decision tree to be poor after seeing the information gain in the feature selection. No classes had a high information gain resulting in the tree being split on features that had high levels of entropy and thus high levels of impurity. Form the lectures I learned that the goal of the decision tree is to minimise the dept of the tree and when I visualised the tree in weka I found that it had 18 branches and in many instances split on the same feature multiple times, this definitely not desired. The Naive bay algorithm generally works better with categorical data and

as our data is continuous it could lead to some errors; this possibly explains its apparent poorer performance compared to KNN.

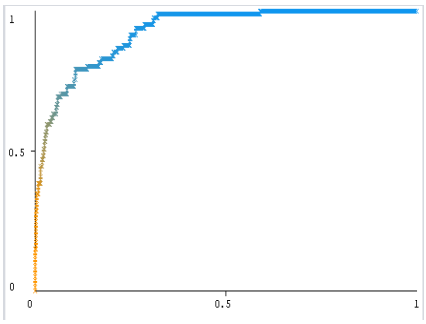
As noted in part C purely using the accuracy is not wise, and this can be seen through all of the evaluation metrics shown in table 1.5, Interestingly the recall of the high class is particularly bad in the j48 and naive bay algorithms, meaning they failed to identify a lot of classes that actually belonged to this class, the recall value of KNN is significantly higher than the other two, this is also reflected in the F-measure as KNN has the highest of the 3. Another area of interest is the that false positive rate for the mid class is consistently the highest False positive rate in all classifiers, especially in the j48 algorithm with a FP rate of 0.371. This suggest that the model ls overfit to the mid class, this is also evident in the class distribution graph as 61% of the classes are mid with only 16% being high.

Part E

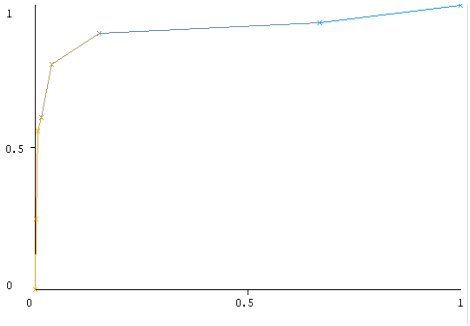
ROC curve for J48



ROC curve for Naïve bayes



ROC curve for KNN



The ROC is a good measure of discussing how good a model is at differentiating between classes. The above graphs showing the ROC were generated using weka, the x axis is the False positive rate, and the y axis is the true positive rate. To evaluate the performance of models we can use the Area under the curve or AUC. Shown in table 1.6 Is the area under the curve for each classifier. Generally, the higher the AUC the better the classifier, in this instance it is very interesting to see that the Naïve bayes outperforms the KNN algorithm. This was a surprise to me as all of the previous data had indicated that the KNN was the better classifier, however it is worth noting this is only for the high class. All algorithms perform significantly better than random classification which I discuss in more detail in question 2.

	j48	Naïve	knn
AUC	0.8221	0.925	0.9153

Table 1.6

Overall, I believe the KNN algorithm is still the best classifier for the dataset considering its superior results for precision and recall and also its better f-Measure as demonstrated in part D. I believe that this model seems to function the best however all models could absolutely be improved if the training set had more balanced data. In this instance with a model that's goal is to classify the chance of diabetes having a higher precision and recall may be more beneficial than accuracy, the difference in the AUC is also marginal so both classifiers are still very similar using this metric.

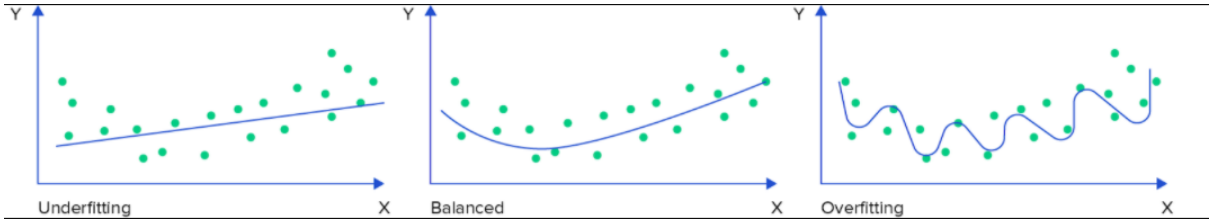
I am somewhat satisfied with the overall performance of the model however it is somewhat overfit to the mid class, with better class distribution for the training data I believe that the model's overall performance could be improved.

Question2

The unfortunate event of the accuracy of the model being significantly lower for the test model than accuracy of the model of the training data is of course something which is never desired and there are many reasons for this occurring.

A contributing factor to this could be due to data imbalances, It is interesting to note that the majority class is less than 65 %, a common reason for poor accuracy on test data is imbalanced data as it can lead to the algorithm being biased and simply predicting the majority class i.e. if only 2 of 100 items in the test data are false then the algorithm could learn to always return true with a 98% accuracy however if we then tested the algorithm on a data set with a 50/50 split of true and false the accuracy would significantly drop. A solution to this would be for the user to use another evaluation metric like the F1-measure which is used to measure the harmonic mean of precision and recall. This can help address the problem of imbalanced data as it focuses on False Negatives and False positives as opposed to just the accuracy and depending on the context of the data may be a more valuable measure that just the accuracy, for example f-score would be more useful when there is a large imbalance in the data. In this example the fact that less than 65% of the data is of the majority class could have a small part to play in the accuracy of the tests data being below 50% but it is certainly is not the main cause for the discrepancy. If there were more than two classed, then this imbalance could be a larger issue.

The main reasons for the result being inaccurate is due to overfitting this is where the model is too closely fitted to the training data that it can no longer generalise new unseen data. Essentially, we develop large set of arbitrary patterns that classify the training set very well but does not classify new data well at all. Some of the causes of overfitting are noise in your training data or an excessive amount of feature selection. When adding more features, it might seem like the added detail is beneficial however it not as it actually makes the models accuracy for unseen data far worse this can be seen in the clients' example. Shown in the image below is an example of underfitting (model not good for at categorising training data or generalising new data), balanced and overfitting as explained above. Here we should utilise the goldilocks principle where we aim to achieve the balanced model that is right in between the overfitted and underfitted model.

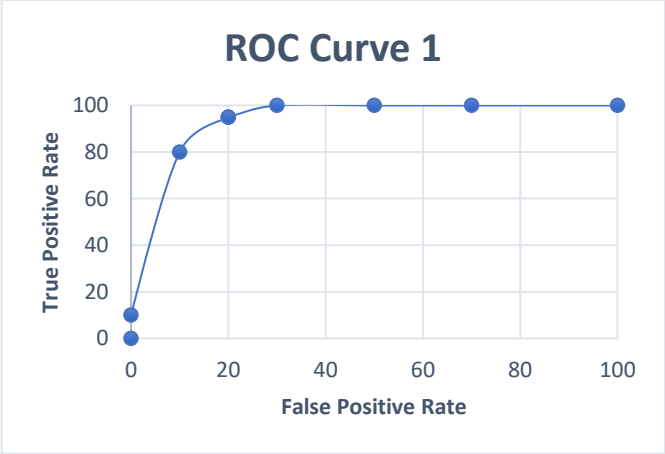


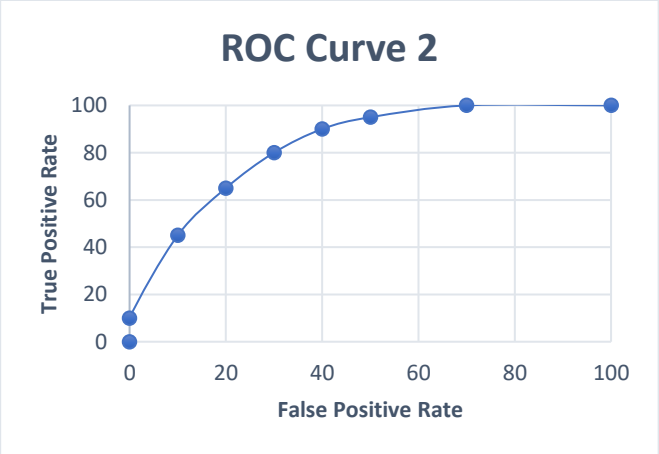
Solution

There are two ways in which we can reduce overfitting in our data set these are using k fold cross validation and a three way hold out strategy. K-fold cross validation is a resampling technique where k is the number of groups the data is split into. k-1 groups are used for training while the remaining group is used for testing, the accuracy of each fold can then be obtained and compared to judge the effectiveness of the model. The three way hold out strategy is when we split the data into a Training set, a Validation set and a test set, this differs from normal hold out strategies as it includes the validation set which is a subset of your data that you hold out from your model which allows you to evaluate the performance of the classifier which can then be used to tune it. A combination approach of these two strategies will certainly help the client achieve a much higher level of accuracy on their test data and also drastically reduce the chance of overfitting.

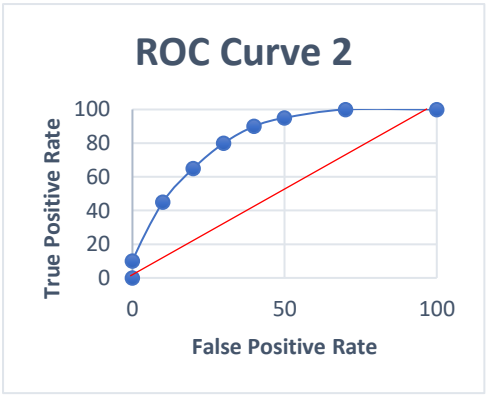
part B

An ROC(Receiver operating characteristic) curve a simple graphical representational curve that highlights the connection between true positive rate and the true negative rate for each probability threshold in a data set. To understand ROC it is helpful to understand how it is used, in many machine learning classification algorithms each feature will be assigned a probability of belong to a class, for example with Naive bays it could classify an item as having an 80% change of belonging to a class and another as having a 60% chance of belonging to the same class, here how do we decide what decision threshold should be accepted, generally the decision threshold is set to 50% which would result in both classes being accepted, but if that threshold was raised to 70% then one item would not meet the threshold and would be classified as the other class. Take for example with a really low decision threshold there will be a lot of false positives and with a higher threshold there will be more false negatives. The Roc curve will allow us to view a graphical representation of the true positive and the false positive rate for every possible decision threshold and thus allow us to choose the optimal threshold for the classifier





Shown above are two sample ROC curves. To interpret the results of a roc curve we can do two things look at how closely the curve is fit to the top left corner, from the above example we can see the left curve is much closer to the corner than the right one. The second way is to compute the area under the curve or AUC, the higher that the AUC, the better the model Is. The AUC is especially handy for comparing different classification models or approaches as it can clearly show which one is better and thus should be adopted.



When using a ROC curve, it is common to include a reference line as indicated by the red line in the graph to the right. This line represents a random classifier .i.e. you flipped a coin every time to decide what class an item belongs to, we want to maximise the distance from the line to the curve so from the graph we could say the model has about a 75% change of classifying the item correctly which is significantly better than random classification, in the unfortunate event our curve was below the reference line i.e. 50% then the model is essentially useless and should absolutely not be used as simply guessing the outcome is more accurate. Overall, the ROC gives great insight into the performance capabilities of a classifier and provides a simple way to compare different classifiers at different decision thresholds.