COMP47490 Enxi Cui 17210995

Question 1

(a)
Filter
Information Gain Filter:
In Weka *Select attributes* tab, I chose *InfoGainAttributeEval* as the evaluator, *Ranker* as the method
Selected attributes:
```
Ranked attributes:
 0.022319    6 handsetAge
 0.020758   25 lifeTime
 0.006549   13 avgMins
 0.006149   14 avgrecurringCharge
 0.005164    7 smartPhone
 0.004       9 creditRating
 0.002887   20 avgOutCalls
 0.002866   27 numRetentionCalls
 0.002791   21 avgInCalls
 0.002752   17 callMinutesChangePct
 0.002401   26 lastMonthCustomerCareCalls
 0.002146   19 avgReceivedMins
 0.001802   24 avgDroppedCalls
 0.001197   28 numRetentionOffersAccepted
 0.001083    5 numHandsets
 0.000992    8 currentHandsetPrice
 0.000411    2 marriageStatus
 0.000154   10 homeOwner
 0.00012    11 creditCard
 0.000104    3 children
 0          29 newFrequentNumbers
 0           4 income
 0          15 avgOverBundleMins
 0          12 avgBill
 0          16 avgRoamCalls
 0          23 peakOffPeakRatioChangePct
 0          22 peakOffPeakRatio
 0          18 billAmountChangePct
 0           1 age
```
Selected attributes: 6,25,13,14,7,9,20,27,21,17,26,19,24,28,5,8,2,10,11,3,29,4,15,12,16,23,22,18,1 : 29

The highest two rank features are handsetAge and lifeTime, the lowest rank features (rank score < 0.001) are age, billAmountChangePct, peakOffPeakRatio, peakOffPeakRatioChangePct, avgRoamCalls, avgBill, avgOverBundleMins, income, newFrequentNumbers, children, creditCard, homeowner, marriageStatus, currentHandsetPrice.

Wrapper
Sequential search:
In Select attributes tab, I chose *WrapperSubsetEval* as the evaluator with *J48* (decision tree)as classifier, *GreedyStepwise* as search method (*backward elimination*): (Completion time takes quite a bit longer than search forward)
```
Selected attributes: 6,7,13,14,15,17,18,20,23,25,26,27 : 12
                     handsetAge
                     smartPhone
                     avgMins
                     avgrecurringCharge
                     avgOverBundleMins
                     callMinutesChangePct
                     billAmountChangePct
                     avgOutCalls
                     peakOffPeakRatioChangePct
                     lifeTime
                     lastMonthCustomerCareCalls
                     numRetentionCalls
```
selected 12/30 features

I chose the slower backward elimination instead of forward search. In the beginning, all features are included, and then the least informative of the features is removed. This process continues until the dropping of additional features no longer results in an increase in accuracy. This search method is usually the better than forward selection as it can find subsets with features which interact.

Filter:
After using Information Gain, I got the ranked features. Next, I evaluated classification performance using feature subsets of increasing size, starting with the highest information gain feature to get the accuracy, and then, adding the next highest ranked feature. After getting each subset's accuracy using cross-validation, I chose the highest accuracy subset.

First I chose Decision tree as the classifier.

Features number	Accuracy
6	58.6105 %
6, 25	59.5474 %
6, 25, 13	59.8105 %
6, 25, 13, 14	59.7158 %
6, 25, 13, 14, 7	56.7684 %
6, 25, 13, 14, 7, 9	62.3789 %

6, 25, 13, 14, 7, 9, 20	62.5158 %
6, 25, 13, 14, 7, 9, 20, 27	62.6947 %
6, 25, 13, 14, 7, 9, 20, 27, 21	63.4105 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17	64.8526 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26	66%
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19	66.0526 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24	67.1368 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28	67%
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28, 5	67.7368 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28, 5, 8	68.2526 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28, 5, 8, 2	77.4947 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28, 5, 8, 2, 10	81.2526 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28, 5, 8, 2, 10, 11	82.5789 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28, 5, 8, 2, 10, 11, 3	83.2421 %
6, 25, 13, 14, 7, 9, 20, 27, 21, 17, 26, 19, 24, 28, 5, 8, 2, 10, 11, 3, 29	82.1368 %

Next I Chose Naïve Bayes as the classifier.

Features number	Accuracy
6	54.0632 %
6, 25	54.2211 %
6, 25, 13	56.3474 %
6, 25, 13, 14	56.4737 %
6, 25, 13, 14, 7	56.9053 %
6, 25, 13, 14, 7, 9	55.3053%
6, 25, 13, 14, 7, 9, 20	55.4526 %

Finally, I chose *IBk* (kNN, k = 15) as the classifier. Test mode: 10-fold cross-validation

Features number	Accuracy
6	54.9263 %
6, 25	55.926 %
6, 25, 13	55.6737 %
6, 25, 13, 14	57.4526 %
6, 25, 13, 14, 7	57.9684 %
6, 25, 13, 14, 7, 9	56.3474 %
6, 25, 13, 14, 7, 9, 20	56.4842 %

The reason I chose k = 15 in KNN is that firstly it has to be odd, in order to help break any "ties" in the vote which might happen using an even number. In this dataset, there are 9500 data items, thus k cannot be too small, because it would be easier to be sensitive to noise and overfitted. If it's too large, it will take a lot of computational resources. When comparing k values, I tried k= 5, 9, 25 and found that none of the results were more accurate than when k= 15, so I finally chose k = 15.

Comparing all the features above, using the *Decision tree* as the classifier has the highest accuracy when I choose all the features where the rank is above 0. Therefore, I chose the first 20 features as my subset for the filter.

(b)

The features I have chosen by using *Filter: (The 20 features where rank is above 0*):

```
Ranked attributes:
 0.022319    6 handsetAge
 0.020758   25 lifeTime
 0.006549   13 avgMins
 0.006149   14 avgrecurringCharge
 0.005164    7 smartPhone
 0.004       9 creditRating
 0.002887   20 avgOutCalls
 0.002866   27 numRetentionCalls
 0.002791   21 avgInCalls
 0.002752   17 callMinutesChangePct
 0.002401   26 lastMonthCustomerCareCalls
 0.002146   19 avgReceivedMins
 0.001802   24 avgDroppedCalls
 0.001197   28 numRetentionOffersAccepted
 0.001083    5 numHandsets
 0.000992    8 currentHandsetPrice
 0.000411    2 marriageStatus
 0.000154   10 homeOwner
 0.00012    11 creditCard
 0.000104    3 children
```

The features chosen by *Wrapper (backward elimination)* are:

```
Selected attributes: 6,7,13,14,15,17,18,20,23,25,26,27 : 12
                     handsetAge
                     smartPhone
                     avgMins
                     avgrecurringCharge
                     avgOverBundleMins
                     callMinutesChangePct
                     billAmountChangePct
                     avgOutCalls
                     peakOffPeakRatioChangePct
                     lifeTime
                     lastMonthCustomerCareCalls
                     numRetentionCalls
```

Difference:
It can be seen that there are some common features that are chosen by these two methods: handsetAge, lifetime, avgMins, avgrecurringCharge, smartphone, avgOutCalls, numRetentionCalls, callMinutesChangePct, lastMonthCustomerCareCalls (9 features).

There are 3 features that are chosen by *Wrapper* but the ranking is 0 when using *Filter* method: avgOverBundleMins, billAmountChangePct, peakOffPeakRatioChangePct.

There are 2 features where the ranking in the Filter method is relatively high compared to other features, but which are not included in the Wrapper methods: creditRating, avgInCalls.

Why the two techniques can potentially produce different results
The *Information Gain* is calculated as the ranking of each feature. The ranking of each feature is the amount of uncertainty that the feature can reduce under the same conditions. The higher value, the better the certainty of prediction. For those features that have an Information Gain of 0, they may have no effect on the determination of the category, or they have an effect only when in combination with one or more other features. However, *Information Gain* considers each feature separately, and is a pure measurement that does not consider any classification algorithm.

Wrapper is a method that uses a classification algorithm to build a model that contains a subset of attributes, and then evaluates the performance of that model. It uses different subsets to get the best performance features. The classifier might choose the features itself, which could be smaller than the feature subset provided by a filter such as *Information Gain*.

From the analysis above, one of the most obvious reason that they produce different results is that the combination of two or more features may interact with each other to give better performance, which happened in *Wrapper*, but not in the *filter.* Additionally, some classifiers used with a *Wrapper* can choose the feature subsets themselves, which may be different than the subset chosen by the filter.

(c)
Firstly, I used the features from the Information Gain filter to determine the most accurate model by using different features with different classifiers. (See part a).

The features chosen from the Information Gain filter used with the decision tree classifier have a highest accuracy of 83.2421%.
The features chosen from the Information Gain filter used with the k-Nearest Neighbour classifier have a highest accuracy of 57.9684%.

```
=== Summary ===

Correctly Classified Instances        7908               83.2421 %
Incorrectly Classified Instances      1592               16.7579 %
Kappa statistic                          0.6649
Mean absolute error                      0.2335
Root mean squared error                  0.3417
Relative absolute error                 46.7011 %
Root relative squared error             68.3382 %
Total Number of Instances             9500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.856    0.191    0.817      0.856    0.836      0.666    0.916     0.917     true
                 0.809    0.144    0.849      0.809    0.829      0.666    0.916     0.917     false
Weighted Avg.    0.832    0.167    0.833      0.832    0.832      0.666    0.916     0.917

=== Confusion Matrix ===

    a    b   <-- classified as
 4056  684 |    a = true
  908 3852 |    b = false
```

*Figure1.1 Filter, classifier as **Decision tree** with 20 features.*

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        5050               53.1579 %
Incorrectly Classified Instances      4450               46.8421 %
Kappa statistic                          0.0631
Mean absolute error                      0.4864
Root mean squared error                  0.5082
Relative absolute error                 97.2805 %
Root relative squared error            101.6361 %
Total Number of Instances             9500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.527    0.464    0.531      0.527    0.529      0.063    0.550     0.535     true
                 0.536    0.473    0.532      0.536    0.534      0.063    0.550     0.543     false
Weighted Avg.    0.532    0.468    0.532      0.532    0.532      0.063    0.550     0.539

=== Confusion Matrix ===

    a    b   <-- classified as
 2497 2243 |    a = true
 2207 2553 |    b = false
```

*Figure1.2 Filter, classifier as **kNN** (k = 15) with features 6 handsetAge, 25, lifeTime, 13 avgMins, 14 avgrecurringCharge and 7 smartPhone.*

```
=== Summary ===

Correctly Classified Instances        5988               63.0316 %
Incorrectly Classified Instances      3512               36.9684 %
Kappa statistic                          0.2604
Mean absolute error                      0.4423
Root mean squared error                  0.4703
Relative absolute error                 88.4584 %
Root relative squared error             94.0523 %
Total Number of Instances             9500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.559    0.299    0.651      0.559    0.602      0.263    0.686     0.671     true
                 0.701    0.441    0.615      0.701    0.655      0.263    0.686     0.668     false
Weighted Avg.    0.630    0.370    0.633      0.630    0.628      0.263    0.686     0.670

=== Confusion Matrix ===

    a    b   <-- classified as
 2651 2089 |    a = true
 1423 3337 |    b = false
```

Figure2.1 Wrapper** with **decision tree

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        5401               56.8526 %
Incorrectly Classified Instances      4099               43.1474 %
Kappa statistic                          0.137
Mean absolute error                      0.473
Root mean squared error                  0.5018
Relative absolute error                 94.6096 %
Root relative squared error            100.3564 %
Total Number of Instances             9500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.560    0.423    0.569      0.560    0.564      0.137    0.589     0.556     true
                 0.577    0.440    0.568      0.577    0.573      0.137    0.589     0.583     false
Weighted Avg.    0.569    0.432    0.569      0.569    0.568      0.137    0.589     0.569

=== Confusion Matrix ===

    a    b   <-- classified as
 2655 2085 |    a = true
 2014 2746 |    b = false
```

Figure2.2 Wrapper** with **kNN k = 15

For this dataset we are trying to see whether or not a customer will leave the service. The company which wants to predict this will be most interested in how accurate their model is, as having an accurate picture of which customers might leave is something very important of customer retention efforts. Therefore, I have decided to compare these 4 combinations using their accuracy, as a more accurate model will help the company better carry out its predictions.

Looking at the above results, the error rate of the Decision tree classifier is relatively lower than that of kNN regardless of whether the features were chosen by Filter or Wrapper.

Decision Tree:

With filter: 83.2421% accuracy.

With wrapper: 63.0316% accuracy.

kNN:

With filter: 53.1579% accuracy.

With wrapper: 56.8526% accuracy.

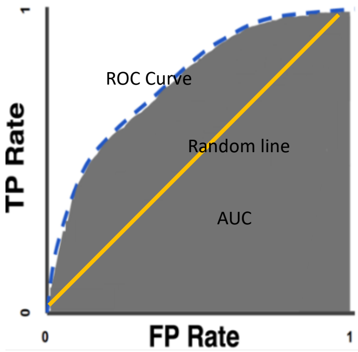
With the decision tree, using the filter for feature selection results in a higher accuracy, but with kNN, using the wrapper has a higher accuracy. This shows that using a filter is not always better than using a wrapper, and vice versa. It depends on the dataset and the classifier used.

By using the 20 features I have chosen by using the Filter, the error rate of the decision tree is the lowest and the average precision is the highest amongst all 4 combinations. Precision is a measure of the true identifications which were actually true. Recall is the number actual true identifications which were correctly identified as true.

The combination of the features obtained using the Filter with the decision tree classifier leads to an accuracy which is quite a large margin higher than the other 3 combinations, which shows that this the most suitable method for prediction on this dataset. The kNN accuracy level was just over 50%, which means the model was close to 50-50 random choice. This would make the model largely pointless for the company, who would require something with greater accuracy. Luckily, the decision tree was able to provide this extra accuracy, especially when paired with the filter feature selection method.

Question 2

(a)



ROC (receiver operating characteristic) curve is mainly expressed by True Positive Rate (TPR = TP/(TP+FN)) on the Y axis and False Positive Rate (FPR = FP/(FP+TN)) on the X axis. The higher the TPR and the lower the FPR means the better accuracy of the prediction model. By drawing each ROC curve into the same coordinates, we can intuitively identify each classifier's performance. The ROC curve nearest the upper left corner (the bigger of the area below the curve (AUC))represents the classifier with the highest accuracy. It helps to select the best threshold; the point on the ROC curve nearest the upper left corner is the best threshold with the least classification errors. The ROC curve can easily detect the influence of any threshold on the generalization performance of the classifier. The yellow line on the right image is called random line. Anything below the line is considered worse than random, and anything above it is considered better than random. The best case is "1" in the top left on the image. The further distance of the ROC curve from the line to the top left corner, the better the classifier is. Thus, a better classifier will have a larger area under the curve.

(b)

Eager learning method is to use the training data to get an classification model before using the algorithm to make a decision, and to use the trained function to make a decision when it is necessary to make a decision. This method requires some work at the beginning and is convenient for later use. It usually use Decision Tree, Naive Bayes, Artificial Neural Networks as classifiers. They all build models based on training data sets to predict new data.

Lazy learning won't create classification model as eager learning at the beginning. It just stores the training data, and starts analyzing the relationship between testing data and stored training data when we need to judge the testing data. It determines the classification model value of the testing data after that. It usually uses k-Nearest Neighbour as classifiers. KNN classification is not "in a hurry" to learn. When receiving a new data, the program calculates the k closest training data to the test data and determines the category of test data according to the principle of majority voting.

(c)

The independence of the event: assuming that A and B are two test E events, events A generally will have an impact on the occurrence of event B, and in some cases the happening of the event A B has no influence for events, can be expressed as A (B | A) = P (A, B) called AB independent events: P (AB) = P (A) P (B | A) = P (A) P (B), for multiple events can be expressed as P (X1, X1... The X3) = P (X1) P (X2)... P (Xn).

For the training data, assumption that even training data can expressed by n dimensional vector X=(x1, x2...xn). Describe the class label set is {c1, c2… cm} so $c(x) = \arg\max_{c \in C} P(x1, x2…xn \mid c)\, P(c)$

c is a category in the class set. From the above equation, it is mainly to estimate two probability values. The value of P(c) only needs to calculate the frequency of each class mark c appearing in the training sample set. But estimate P(x1, x2... Xn | c) is difficult, so Naïve Bayes assumes that the n attributes of the sample are conditionally independent of each other, according to the independence of the event P(x1, x2... xn | c) can be expressed as P(x1, x2…xn | x) = $\prod_{j=1}^{n}$ P(xj | c)

Naïve Bayes can be express as: $c(x) = \arg\max_{c \in C} P(c) \prod_{j=1}^{n} P(x_j \mid c)$

With the conditional independence hypothesis, we don't need to calculate the class conditional probability of each combination of X, we just need to calculate the conditional probability of each xj for the given category.

(d)

Imbalanced data is data set sample categories are extremely uneven. Suppose that 99.9% of the people in the world are not committing fraud, and only 0.01% are committing fraud. So, we want to design a classifier to determine if a person is committing fraud. Thus, with prior knowledge, I just have to assume that no one is committing fraud, and the classifier achieves at least 99.9% classification accuracy. Obviously, this classifier has no value at all. Similarly, for the class sample distribution of N: 1(N is relatively large) , we only need to think that all the samples belong to the class of N, the accuracy can be very high, but there is no significance and reference value. So, accuracy measures are not appropriate here.

ROC analysis provides a more accurate measure of dealing with skewed class sizes than accuracy. By setting different thresholds, different confusion matrices can be obtained, and each confusion matrix corresponds to a point on the ROC curve. By plotting these points, a ROC curve can be obtained. The area surrounded by the curve (AUC) can be calculated to measure the credibility of the classifier. The larger the area, the higher the credibility.

(e)

We usually split features to increase information gain. The problem with splitting features such as name and credit card number is that there wouldn't be any information gain in doing so. A person's credit card information is a unique identifier. For example, just because the credit card number 123456 might belong to someone who committed fraud, this number is unique and therefore can't give an indication that someone else with this number might be committing fraud; only one card with this number exists. For example splitting the number in two, such as "123" and "456" won't give us any extra information, as it's not the case that people with a certain string of digits in their card number are more likely to commit fraud. The same goes for names: if someone named Karl Karlsson commits fraud, this may not be a unique name in that more than one person could be called Karl Karlsson, but it's unlikely that a significant percentage of the dataset will have the same name. More importantly, a name doesn't give any real information about how likely someone is to commit fraud; those named Karl are not predisposed to criminal activities. Therefore, splitting the name into "Karl" and "Karlsson" for example won't give any extra information. Karl Karlsson with card number 123456 may have committed fraud so therefore this entry in the dataset will show the name and credit card number as high information gain, but these aren't feature values which can help in predictions.