Question 1

Before I started to answer the questions, I decided to clean the data. Initially, I used the original unaltered dataset with J48 classifier to test its accuracy, and it was 86.1276%. Then I began to clean the data, and used Filter > unsupervised > instance > Remove Duplicated, and then tried the same classifier, with the accuracy increasing to 86.2079%.

As for outliers, I used the InterquartileRange Filter to find outliers. However, while this showed that there were outliers, I was not sure what data is considered acceptable or unacceptable; despite being an outlier, it may still be valid. Therefore, I decided not to remove outliers in case they are valid entries.

So as to make sure that the model is not being trained on data which includes missing values (which could skew the result), I used the ReplaceMissingValues filter to replace the these missing nominal and numeric values with the modes and means from the training data. The attributes with missing values were: 2 workclass, 7 occupation, 14 native-country. This actually results in a slight decrease in the classifier accuracy (86.0008 %), but this is acceptable as the model trained before filling in the missing data was trained using bad data. The model trained using data with no missing values is taking all data into account.

I used Information Gain to check each features' rank, and I found that the feature 3 fnlwgt's rank is 0, so I deleted this feature in the dataset. After this, I selected unsupervised attribute in the Filter to first initialize the dataset, and the newly normalized dataset was saved to new arff file.

(a)

```
Correctly Classified Instances       26577               85.9847 %
Incorrectly Classified Instances      4332               14.0153 %
Kappa statistic                          0.5944
Mean absolute error                      0.1938
Root mean squared error                  0.3234
Relative absolute error                 53.0593 %
Root relative squared error             75.6649 %
Total Number of Instances            30909

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.630    0.067    0.748      0.630    0.684      0.598    0.883     0.735     >50K
               0.933    0.370    0.888      0.933    0.910      0.598    0.883     0.941     <=50K
Weighted Avg.  0.860    0.298    0.855      0.860    0.856      0.598    0.883     0.892

=== Confusion Matrix ===

    a     b   <-- classified as
 4679  2753 |     a = >50K
 1579 21898 |     b = <=50K
```

Decision tree(J48)

```
Correctly Classified Instances       24576               79.5108 %
Incorrectly Classified Instances      6333               20.4892 %
Kappa statistic                          0.434
Mean absolute error                      0.2049
Root mean squared error                  0.4526
Relative absolute error                 56.1027 %
Root relative squared error            105.9171 %
Total Number of Instances            30909

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.561    0.131    0.576      0.561    0.568      0.434    0.716     0.433     >50K
               0.869    0.439    0.862      0.869    0.866      0.434    0.716     0.850     <=50K
Weighted Avg.  0.795    0.365    0.793      0.795    0.794      0.434    0.716     0.750

=== Confusion Matrix ===

    a     b   <-- classified as
 4168  3264 |     a = >50K
 3069 20408 |     b = <=50K
```

1-Nearest Neighbor

In the case of accuracy, decision tree is a better predictive model. Accuracy is about 7% higher than the 1NN. Usually, the precision and recall are analysed together, so I compared them by PRC Area, where the higher the area of the curve means higher accuracy of the model. The decision tree was about 30% higher in the prediction of the target class. F1-measure is the harmonic average of Precision and Recall, and its value is closer to the smaller of the two Precision and Recall values. From the PRC Area and F-measure, we can see from the results that the prediction of the class of people who get less than 50k is better than the other class in both classifiers. As for the ROC Area, the bigger the area of the curve, the better performance of the

model. In this case, decision tree is also better than 1NN. For the Confusion Matrix, it is obvious from the results that the FP(false positive) and FN(false negative) in 1NN is more than it is in Decision tree. All of these evaluations shows the decision tree has a better performance the 1NN in this situation.

I used accuracy, F-measure, ROC Area, PRC Area and Confusion Matrix in this part. For accuracy, this is a quick and easy way to compare the accuracy of the two models. However, if further investigation is needed, additional methods are required to evaluate this model. The paradox between Precision and Recall index sometimes occurs where they contradict each other, so they need to be comprehensively considered. The F-measure is a metric. It takes into account both precision and recall. The higher the F1-score, the more robust the classification model. In fact, precision and recall are often combined giving the PRC Area, where the area below the curve indicates higher recall and accuracy. Both of these methods have high scores, which indicates that the classifier returns accurate results.

The larger the ROC Area (AUC), the better the model. The calculation method of AUC takes into account the classification ability of the classifier for positive and negative cases. In the case of unbalanced samples, the classifier can still make a reasonable evaluation.

The factors which can explain the different results between 1NN and decision tree may be due to the following:

1. K is too small. This is a very big dataset. A simple 1NN classifier is easy to implement, but it's affected by the noise in the data. It means that the whole model becomes easy to overfit. After trying k = 15, I found that this accuracy was about 4% higher than k=1.
2. KNN is more affected by the curse of dimensionality. For a large number of features it cannot predict well, and there are 14 features in this dataset. In contrast, decision tree won't be affected as much by this high dimensionality.

(b)

I decided to use accuracy as the evaluation method from now on, because it's the most direct parameter that can show how good the model is. When comparing the accuracy of these two classifiers with bagging with different ensemble sizes, I got the results displayed in the charts below. Looking at figure 2.1, all the accuracies scores don't have much difference between ensemble sizes, but it still can be seen that when the ensemble size is 80, J48 has the highest accuracy. For 1NN, the optimal value for ensemble size is 20. (figure 2.1)

I chose the best ensemble size of these two classifiers and change the bag size to compare the performance them. (figure 2.2)

The accuracy was tested with different bag sizes, but they don't show big differences in accuracy. From figure 2.1, we can see that the decision tree achieved the best accuracy when the bag size was 40. The best model that 1-NN achieved was when the bag size is 10. We can also see that the decision tree performs better than 1NN for this dataset

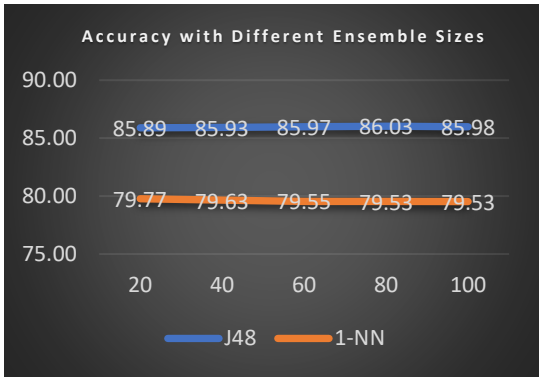


Figure 2.1(x is ensemble sizes, y is accuracy)

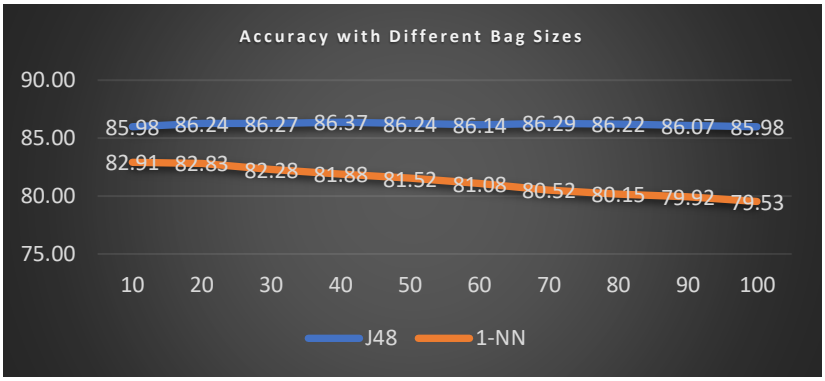


Figure 2.1(x is bag sizes, y is accuracy)

(c)

When comparing the accuracy of these two classifiers with random sub spacing with different ensemble sizes, I got the results shown in the charts below. From figure 3.1, we can see that both J48 decision tree and 1-NN perform best when the ensemble size is 40. 1NN has shown improvements with random subspace compared to using bagging.

When I was using 1-NN with the ensemble size 100, Weka became stuck, so I used “Filter -> Unsupervised -> Instance -> Resample (sampleSizePercent=10, noreplacement=True)” to reduce around 90% of the dataset. I repeated the tests with this new dataset to get the accuracy of 1-NN with different ensemble sizes. (Figure 3.1)

I compared the accuracy of decision tree and 1-NN when the subspace size equals to 0.25, 0.5, 0.75 and 0.95 with the ensemble size of 60 and 40 relatively. (Figure 3.2)

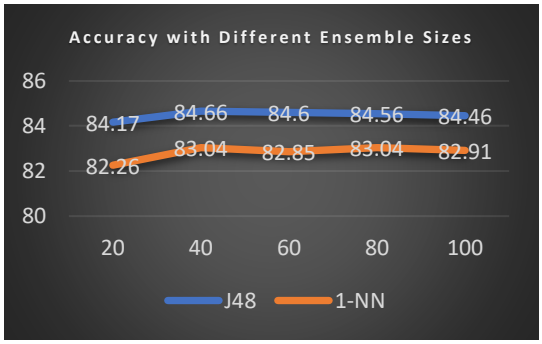


Figure 3.1(x is ensemble sizes, y is accuracy)

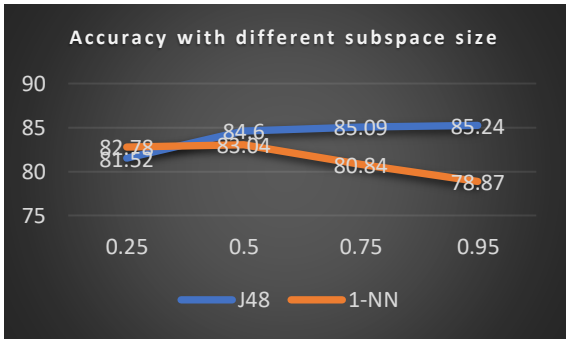


Figure 3.2(x is subspace sizes, y is accuracy)

From figure 3.2, we can see that the higher the subspace size the better the accuracy for J48, with the highest accuracy achieved for 1-NN was when the subspace size is 0.5. With the increase of subspace size, the 1-NN accuracy decreases obviously, which shows for this dataset, KNN is not good when using all the features.

To verify the accuracy of changing the size of the subspace, I also tried the WrapperSubset evaluation method by backward learning to try to get the number of features to compare the size of the subspace. In J48 I got 12 features 1,2,3,5,6,7,8,9,10,11,12,14 and in 1-NN I got 8 features 1,2,4,6,8, 11,12,13. The number of features selected with wrapper is almost the same as the optimal number with subspace size, further demonstrating that cooperation between different sets of features can be represented in different classifiers.

It can be shown that the 1-NN shows big improvement by using random subspace, but J48's accuracy didn't change a lot. From this, I learned that kNN performs better when it is used in conjunction with random subspace.

(d)

	J48	**1-NN**
Accuracy	85.9847%	79.5108%
Bagging Accuracy (Best)	86.37% (EnsembleSize=80, BagSize = 40)	82.91% (EnsembleSize=80, BagSize = 10)
RandomSubSpace (Best)	85.24% (EnsembleSize=40, Subspace = 0.95)	85.24 (EnsembleSize=40, Subspace = 0.5)

From the above analysis, bagging is better for the decision tree, kNN is better for random subspace. Decision tree performs better with high subspace size in the very large datasets. From question a. it can be shown that the j48 accuracy is about 86%, the 1-NN is about 79%. The results from question b. showed that j48's accuracy with bagging didn't change a lot, the same as 1NN. However, when I used the combination of best ensemble size with a small bag size, the 1-NN improved by about 3%.

Bagging chooses a random sample from the training data, and kNN makes predictions by finding the nearest neighbour. If the training data is disturbed by this sampling, this can affect the available neighbours in kNN. Thus, bagging is not a good method to use with kNN. Changing the bag size percentage means changing the percentage of the training set used. This also explains why when the bagging size is small, the accuracy is higher.

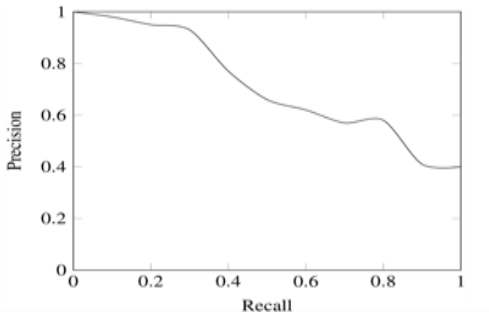
When I was doing question c., the decision tree accuracy didn't change much with different ensemble sizes, and the 1-NN was similar in that the accuracy remained much the same, with an accuracy about 83% no matter which ensemble size was chosen. When I change the subspace size, the accuracy of decision tree increases with the subspace size. When the subspace size equals to 0.5, kNN has the highest accuracy.

In conclusion, for this dataset, bagging can help decision tree's accuracy a little, but it doesn't have a big improvement (from 85.9 to 86.37) kNN's accuracy however had an obvious improvement by using bagging and random subspace.

I also tried another ensemble method trying to find the best accuracy which is boosting. Both classifiers' accuracy didn't change a lot with different ensemble sizes. Decision tree was around 83.4% and kNN was around 79% accuracy, which is below the accuracy found in questions b. and c. Thus, the best ensemble strategy for decision tree was using bagging with ensemble size 80 and bag size 40. For kNN, using random subspace with ensemble size 40 and subspace size 0.5 is the best ensemble strategy.

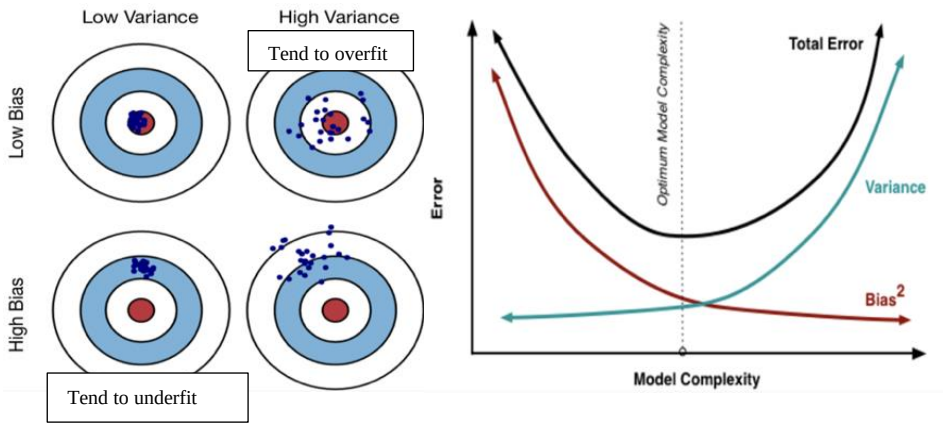
Question 2

(a) The Precision-Recall curve shows the trade-off. Precision is the relative ratio of retrieved results; Recall is the ratio of the retrieved results to all the relevant results in the database. The larger area below the curve, the better the model.

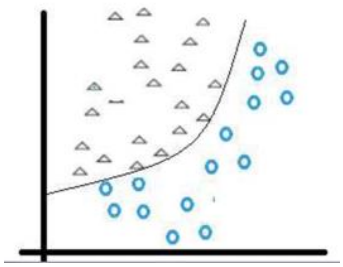


Recall = true positives / true positives + false negatives
Precision = true positives / true positives + false positives
F = (1 +) * precision * recall / * precision + recall
The Beta parameter controls the trade-off. When <1, it will focus more on precision. = 1, it is the harmonic mean of precision and recall. When >1, it will focus more on recall.
F1 sets beta to 1, which is also the most widely used variant. F1 takes into account both recall and precision; it is the trade-off between the two measures for different threshold. Increasing this threshold increases the denominator of the recall, which decreases the recall. Lowering the threshold may also leave the recall rate unchanged, while the accuracy rate fluctuates. A high area below the curve indicates a high recall and precision. High precision rate means low false positive rate, high recall rate means low false negative rate. High scores for both indicate that the classifier is returning accurate results with high precision, and mostly positive results.

(b) Bias is the difference between the expected prediction of the model and the correct value that we are trying to predict, it describes the fitting ability of the learning algorithm itself. Variance is the deviation degree between multiple fitting predictions between models. When the model is too complex, high variance is easy to occur. High variance but Low bias classifiers tend to overfit. When the model is too simple, high bias is easy to occur. High bias but low variance classifiers tend to underfit. Generalization ability is determined by the ability of the learning algorithm, the amount of data and the difficulty of the learning task. In order to obtain good generalization performance, the bias should be small, the data should be fully fitted, the variance should be small, and the influence of data disturbance should be minimized.
In order to achieve a reasonable trade-off of bias-variance, the model needs to be carefully evaluated. K-fold Cross Validation can help us obtain a credible estimate of the model's generalization error, which is the model's representation on the new data set. When the training data is large, we will have less Bias and more Variance. When K value is small, we will have more Bias and less Variance. There are another two methods can also help with it, which are bagging and boosting. Bagging usually reduces the variance of the error. Boosting can reduce variance and bias because it focuses on misclassified examples, but it sometimes result in an increase in error and susceptibility to noise, leading to overfitting.

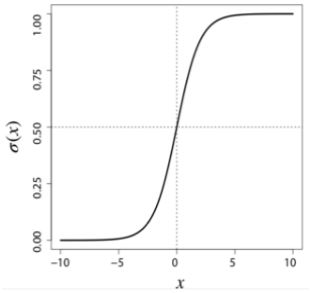


(c) Complex neural networks can be obtained by adding simple neural networks, but they are still linear models and cannot solve nonlinear classification problems. So we convert the existing neural network to nonlinearity, which is to convert the dividing line into a curve. After each layer is superimposed, we add an activation function, as shown in the figure, so that the output is a nonlinear function . With such nonlinear activation function, the expression ability of neural network is more powerful.



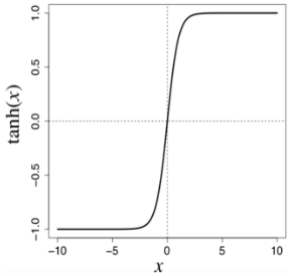
The activation function can have several different formulas and graphs.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



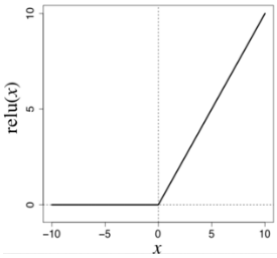
Sigmoid function. It normalizes the input values to (0, 1), it is S-shape curve. The drawback is the outputs are not zero

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$



Tanh function. The outputs are zero-centered and its between -1 and 1

$$\text{relu}(x) = \max(0, x)$$



Relu function. The output value of this function is always 0 when the input is less than 0, and the output grows linearly when the input is greater than 0. It is not zero-centered.

(d) R2 is a coefficient of determination, which reflects what percentage of the fluctuation of y can be described by the fluctuation of x, and what percentage of the variation of the dependent variable y can be explained by the controlled independent variables x. It can be described as a “goodness of fit” measure in the case of liner regression. This means that we can see how well our fitted linear regression model fits our data, as R2 gives a percentage which shows how well our linear model can explain the dependent variable variation. The variance can be described as the average distance squared of the values from their mean. As it deals in percentages, the output is between 0 and 100% inclusive, with 0% meaning that the model does not explain any of the dependent variable’s variation around its mean, while a score of 100% means that all of the dependent variable’s variation around its mean can be explained. Thus, a higher R2 score means a better fitting regression model. It should be noted however that a good R2 score just means a better fitting model, but does not say whether the model itself is good, and doesn’t take an prediction bias into account.

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_i (\hat{y}^{(i)} - y^{(i)})^2}{\sum_i (\bar{y} - y^{(i)})^2}$$