HW 4 :

1. Part A - Initial Decision Tree
    a. Decision Tree is a recursive Algorithm where we select best attribute and recursively select the best attribute among the remaining.As the data is the combination of Discrete and Contiguous Categories. So I identified first 14 categories as Numeric/Non-numeric based on the values.For each of the Numeric values, I found the mean and used that value for decision. So basically continuous values were Discretized as Binary labels 'LE' and 'G'. I implemented Information Gain Based Decision Tree(Entropy) where we continue for remaining attributes to get best among them based on Information Gain. For stopping criteria, I checked the following.
        i. If there is no data
        ii. If there are no attributes left to check.
        iii. All data belong to one Classifier Category .In all the cases , output of this node was the mostCommon Value of Classifier.

       In case of missing Label for an attribute, I outputted the first available value in that path. After applying the above algorithm and using cross validation, I was getting average accuracy of 80.05

    b. After applying the above algorithm and using cross validation(10K), I was getting average accuracy of 80.05 .

    c. For improving the Accuracy of the Decision Tree, I applied the following approaches.
        i. Used Information Gain Ratio , instead of Information Gain. This improved the results from 80.05 to 80.82
        ii. Pruning - Earlier Stopping criteria was to stop when no data is left. This was modified to stop when number of data points is less than equal to 15.(select based on various iteration, but chose odd number so as to avoid majority conflict). This improved results to 81.89
        iii. To handle missing feature attribute values, I used the majority approach, where I assigned most common value of `attribute` among other examples with same target value . This approach increased my accuracy to 82.40 .

    d. Accuracy Results are as follows :

        Number of records: 46510

        ('accuracy', 0, ': 0.8147')
        ('accuracy', 1, ': 0.8239')
        ('accuracy', 2, ': 0.8222')
        ('accuracy', 3, ': 0.8209')

('accuracy', 4, ': 0.8222')
('accuracy', 5, ': 0.8254')
('accuracy', 6, ': 0.8241')
('accuracy', 7, ': 0.8306')
('accuracy', 8, ': 0.8224')
('accuracy', 9, ': 0.8336')
Average Accuracy: 0.8240

2. Weka - Weka Accepts CSV file. So I have written the script to convert input(tsv) to csv.
    a. Experimentations

        i. J48 -C 0.25 -M 2
        Time taken to build model: 6.83 seconds
        Correctly Classified Instances 40026 86.0608 %
        Incorrectly Classified Instances 6483 13.9392 %

        === Confusion Matrix ===

        a b <-- classified as
        33271 2080 | a = <=50K
        4403 6755 | b = >50K

        ii. SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K
        Time taken to build model: 1142.78 seconds
        Correctly Classified Instances 39509 84.9491 %
        Incorrectly Classified Instances 7000 15.0509 %

        === Confusion Matrix ===

        a b <-- classified as
        33172 2179 | a = <=50K
        4821 6337 | b = >50K

        iii. Logistic -R 1.0E-8 -M -1
        Time taken to build model: 28.92 seconds
        Correctly Classified Instances 39580 85.1018 %
        Incorrectly Classified Instances 6929 14.8982 %

        === Confusion Matrix ===

        a b <-- classified as
        32924 2427 | a = <=50K
        4502 6656 | b = >50K

b. Discussions

i. One of the major reasons for about 3 percent better results for Weka Tool is primarily in handling continuous variables. For continuous variables , I just took the mean and made the binary classifier based on mean values. However, the Weka Tool estimates the threshold based on some sophisticated methods and can do binary as well as Multi way classification of continuous variables based data at hand. Another reason for better performance is that it uses probability based method for estimating values in case of missing feature attribute values.

ii. I chose Logistic Regression. As we used Decision Trees here so I will compare and contrast Decision Trees with Logistic regression providing its advantages and disadvantages over Decision trees.

1. Both algorithms are really fast. There isn't much to distinguish them in terms of run-time.
2. Logistic regression will work better if there's a **single** decision boundary, not necessarily parallel to the axis.
3. Decision trees can be applied to situations where there's not just one underlying decision boundary, but many, and will work best if the class labels roughly lie in hyper-rectangular regions.
4. Logistic regression is intrinsically simple, it has low variance and so is less prone to over-fitting. Decision trees can be scaled up to be very complex, are are more liable to over-fit. Pruning is applied to avoid this.

iii. Amongst the three, on the given dataset, Decision Tree clearly out performs both Logistic Regression and SVM in terms of time to build model and accuracy. Dataset is highly skewed (<=50K more than 75 percent). If we look into the Confusion Matrix, we observe that for label '<=50K' , SVM almost performs as good the Decision Tree, but performs really poorly for the other label. Where as logistic Regression performs better than SVM in '>50K' , but performs poorly in '<=50K'. Time wise both Logistic Regression and Decision Trees are fast but SVM is slow.

In order to improve the accuracy further, I applied the Resample Filter Under Unsupervised for the instance in the Preprocess Step on the data. It improved Decision Tree Accuracy to 88.78% and to 85.6% for Logistic Regression. Main intuition for using this filter was because the data was quite skewed(lot of data of one label as compared to other), so normalise the skewed data was bound to give better results.