1. Why is tree pruning useful in decision tree induction? What is a drawback of using a separate set of tuples to evaluate pruning? (25 points)

1.a Because Prepruning and Postpruning are two ways of Pruning to avoid/ Reduction of overfitting so Pruning stops the tree from becoming too complicated and fitting too closely to the training data, also it Improves accuracy by cutting off less important parts of the tree, pruning helps the tree perform better on data it hasn't seen before, and its trees are smaller and easier to understand, and it reduces noise as well by removing parts of the tree that gets influenced by errors in the training data.

1.b to decide which branches to prune, we need a way to measure the impact of each branch on the tree's performance. This is typically done using a separate validation set, and this can be done by Splitting the data into training and validation sets means there's less data to train the tree, and this can make it less powerful, it takes more resources, which need more time and computational power to handle an extra dataset for validation, and if the validation set isn't a good representation of the overall data it can lead to poor decisions on how much to prune.

2. It is difficult to assess classification accuracy when individual data objects may belong to more than one class at a time. In such cases, comment on what criteria you would use to compare different classifier modeled after the same data. (25 points)

1.Threshold Adjustment:

Experiment with different classification thresholds for each label. Adjusting the threshold can impact the number of labels predicted for each instance.

2.Micro and Macro Averaging:

Micro-Averaging: Compute overall metrics across all instances and labels. It treats all instances and labels equally.

Macro-Averaging: Calculate metrics for each label individually and then average them. It gives equal weight to each label.

3.Multi-label Evaluation Metrics:

Hamming Loss: It calculates the average fraction of labels that are incorrectly predicted.

Subset Accuracy: It measures whether the classifier predicts all the correct labels for an instance.

4.Rank-Based Metrics:

Precision at K: Measure the precision of the top-K predicted labels. This is particularly useful when you want to focus on the most relevant labels.

Average Precision: Calculate the area under the precision-recall curve. It assesses the trade-off between precision and recall.

6.Confusion Matrices, 7. Feature Engineering, 8. Cross Validation, 9. Domain specific metrics, 10 Business goals.

3. A classification model may change dynamically along with the changes of training data streams. This is known as concept drift. Explain why decision tree induction may not be a suitable method for such dynamically changing data sets. Is naive Bayesian a better method on such data sets? Comparing with the naive Bayesian approach, is lazy evaluation (such as the k-nearest neighbor approach) even better? Explain your reasoning. (25 points)

Q1.Decision tree induction may not be a suitable method for such dynamically changing data sets because once a decision tree is built, the tree's structure doesn't change because the structure is fixed and based on the data it was trained on If the data changes, the tree might become less accurate and would need to be rebuilt which takes too much time.

Q2. naive Bayesian is a better data method on such data set, it can calculate probabilities based on the data and can update these calculations easily as new data comes in. it is also easy to update the probabilities in a Naive Bayesian classifier with new data.

Q3. K-NN is even better especially for rapidly changing environments. k-NN adjusts right away to any changes in the data because It uses the most recent data directly. But it requires a lot of computing power when it comes big datasets, because it needs to search through lots of data for each new prediction.

4. Suppose frequent subsequences have been mined from a sequence database, with a given (relative) minimum support, minsup. The database can be updated in two cases: (i) adding new sequences (e.g., new customers buying items), and (ii) appending new subsequences to some existing sequences (e.g., existing customers buying new items). For each case, work out an efficient incremental mining method that derives the complete subsequences satisfying minsup, without mining the whole sequence database from scratch. (25 points)

Case 1: when new Sequences are added we use the existing data we start with the list of frequent subsequences that is already identified from previous data, then for each new sequence, we check if it contains any of these known subsequences and update their counts accordingly, then we can explore new subsequences within these new sequences that might now meet the minimum support threshold, and at the end we add any new frequent subsequences to the list and remove any sequences that no longer meet the minimum support.

Case 2: when we append new subsequences, we need first to update out structures to include the new items in the sequences, then we will have to recalculate how the support counts change for existing subsequences because of these new items, then we will look for any new subsequences formed by the additions that might now fit as frequent, and we will add them to the list and drop any that fall below the support threshold .