Data Mining (CS524)
Programming Assignment 3

Problem - 1

In this problem, I implemented the following algorithms :

1. Bayes Classifier (bayes_classifier.py) - Here given the training data we need to predict the mean, variance and prior probabilities of the problastic function. Here I used a hash table to store the training data (We can find the training samples of any class in O(1) time). Then for each class we calculated the unknown variables.
2. Naive Bayes Classifier (naive_bayes_classifier.py) - Again we try and fit a probability function over the given data but different from bayes classifier here we assume that all the attributes here are independent from each other.
3. KNN (k_nearest_neighbour.py) - In this algorithm we find the k nearest neighbour of each of the testing point and classify it as the class which is occurring highest number of times in the k nearest neigbours. To make our algorithm fast we use heap data structure to store the closest k neigbours. Using it we can insert an element in O(log k) time and remove the largest element in the heap in O (1) time (Need to remove the element when we have more than k elements in closest neigbours).

Comparison (TIme Complexity)

Here we compare the time complexity of the 3 algorithms.
As the time complexity of KNN algorithm is very high (O(Testing size * Training size * dim)), we decided the following training and testing distribution.
For the negative class we take only 0.1% of the examples for testing therefore it contains 284 negative examples. For the positive class we decide to take 5% of the examples as testing examples (Because of the skewness of the dataset we decide to take higher percentage of examples from the positive examples so that we have enough testing classes for both the classes).

The time taken to train the models are given as following -

1. Bayes Classification - 0.55 seconds
2. Naive Bayes Classification - 0.53 seconds

The time for training is approximately same for both the examples because of the optimized matrix multiplication algorithm in the numpy library. This provides an edge over iteratively calculating the variance for each attribute. Next, as the K nearest neighbour algorithm does not require training we find only the prediction time complexity for it.

The time taken for predicting the label of the testing set -

1. Bayes Classsification - 0.049 seconds
2. Naive Bayes Classification - 0.117 seconds
3. KNN - 784 seconds (On google colab)

As in naive bayes classification, we need to iterate over all the attributes separately the time taken to find the prediction is higher than the nayes classification where this iteration is done by the optimized numpy library.
Next. the time taken to predict the labels for KNN is very large because for each testing datapoint we need to iterate the whole training dataset.

Comparison (Evalution)

In this section we will compare the algorithm based on different metics.

Confusion matrix - This is used to calculate different metrics for comparing the algorithms

1. True Positive - If the datapoint is positive and also predicted as positive
2. True Negative - If the datapoint is negative and also predicted as negative
3. False Positive - If the datapoint is negative but predicted as positive
4. False negative - If the datapoint is positive but predicted as negative

Following we define a few metics to compare the algorithms

1. Accuracy - (tp + tn) / (tp + tn + fp + fn)
2. Precision - tp / (tp + fp)
3. Recall - tp / (tp + fn)
4. F1 Score - (2 * P * R) / (P + R) = 2 * tp / (2 * tp + fp + fn)
5. Jaccard distance - tp / (tp + fp + fn)

Following are the value of metrics for different algorithms (for the decided distribution of dataset)
Bayes Classifier -

1. Accuracy - 0.9837
2. Precision - 0.6787
3. Recall - 0.791
4. F1 Score - 0.7307
5. Jaccard - 0.5757

Naive Bayes Classifier -

1. Accuracy - 0.9675
2. Precision - 0.791
3. Recall - 0.791
4. F1 Score - 0.791
5. Jaccard - 0.655

KNN -

1. Accuracy - 0.964
2. Precision - 1.0
3. Recall - 0.54
4. F1 Score - 0.70
5. Jaccard - 0.54166

As we can see from the above metrics the best metrics to evaluate our dataset is jaccard distance or the F1 score. As the number of negative examples are very high accuracy can never tell us the exact goodness of our model. Precision can also tell us about the goodness of the model as it measures it using only the positive classified examples but for a dataset where the negative examples are very low then it will also not be able to evaluate the goodness of our model.

From above we can clearly observe that the Naive bayes classifier is the best model out of the 3 algorithms. It can also tell us that the attributes are pretty much independent from each other.

Following are value of decided metrics on full dataset as training and testing dataset -
Bayes Classifier
Bayes Classifier -

1. Accuracy - 0.975
2. Precision - 0.0581
3. Recall - 0.87
4. F1 Score - 0.109
5. Jaccard - 0.0577

Naive Bayes Classifier -

1. Accuracy - 0.9779
2. Precision - 0.061
3. Recall - 0.8292
4. F1 Score - 0.114
5. Jaccard - 0.0688

Note - For KNN it is impossible to evaluate on whole dataset because of time complexity.

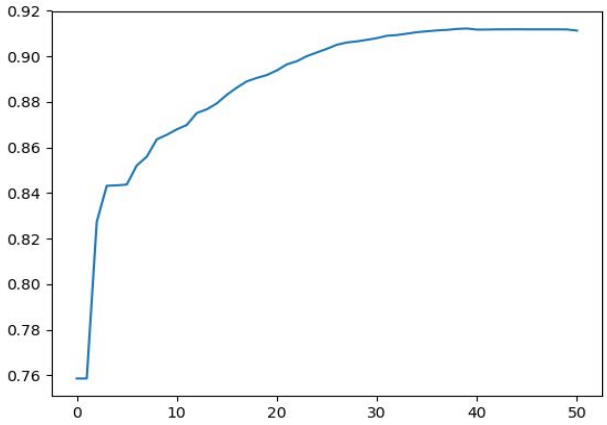
Here we can better understand the efficiency of using jaccard and F1 score as our metrics for evaluation. Here also we conclude that naive bayes algorithm performs better.
The performance for the whole dataset as testing set is poor because the false positives is very high. This helps us conclude that both bayes and naive bayes classifier are inclined to not give false negative while prediction which can be very helpful for some tasks.

Problem - 2

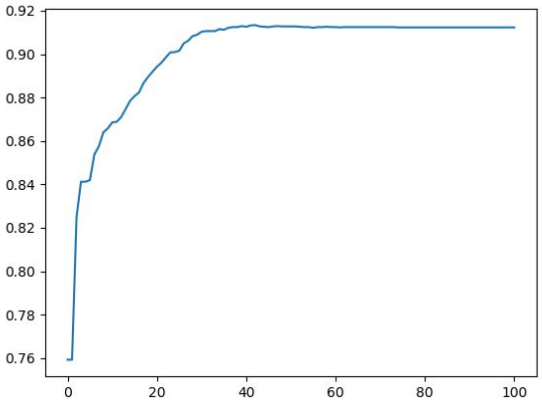
In this problem we implement the decision tree from scratch. First of all we preprocess the data and convert every attribute into a integer / float value. (This is already done for the given dataset). We alo store the meta data for the conversion so that it can easily converted back to the original data or if a new datapoint comes we can also convert it into the known representation.

For implementing the training of decision tree we do not consider the attributes which have very small correlation with the outputs. It can really slow down our training and also does not contribute much information for the final classification. In the given dataset we found only 1 such attribute (attribute 3).

We find the gain of splitting a dataset for each attribute at every possible separation. This may cause a little more time for training but it also develops a lot better decision tree.After training our dataset we save our model in a file using pickle and load the same file at time of testing.



Training accuracy on decision tree



Testing accuracy on decision tree

In the above figures we observe that for training set accuracy increases and keep on increasing for sometime with the maximum achieved accuracy for training set is 94% within 75 nodes. Whereas, for testing set we observe that the accuracy increases for some time but then it starts to decrease after some time after achieving the maximum accuracy as 91.2% for 40 nodes. This is because of the fact we are overfitting our decision tree so we are .

I wasn't able to understand what the question meant by drawing the best Decision tree because our binary tree has a lot of nodes (approx 4000) due to which it is impossible to choose the best tree.
For the best output we decided a height of tree needed is 25 it contains almost 700 nodes.
The best parameter for splitting is 6th parameter and we split at a value of 3.5 (It is according to the encoding we decided).
Left subtree chooses the parameter 7 and split at 2 and similarly the split point and split index is chosen by the algorithm.

Total training time will not be more than a few minutes.