CS F415: Data Mining Assignment-2
REPORT

Name: Arkil Patel
ID: 2016A7PS0665G

Answers to Questions:

Answer 1.

Yes, we need data pre-processing. The following are the reasons and the steps:

- The data given consists of non-numeric data types in certain columns. This data cannot be directly supplied to a classifier as any classifier will only take numeric values. For this purpose, we need to somehow convert the string values into float. For this, there are two possible ways:
  1. **Label Encoding**: In label encoding, all different types of strings in a given attribute are denoted by a unique class value. For e.g., if we have four different types of strings – divorced, married, single and unknown, then it will allot (not always in same order) divorced to '0', married to '1', single to '2', unknown to '3'.
  2. **One Hot encoding**: In One Hot Encoding, the attribute is replaced by as many new attributes as there are unique strings in the attribute and each attribute then takes either 0 or 1 as value representing whether that string occurs or not.

  By common knowledge, we know that values such as divorced, married and single have no ordinal trait in them and hence it is **wrong to use label encoding** for them since it will wrongly create a numerical order in the attribute values. This holds for all the non-numeric attributes in our dataset which have more than two unique strings in them. Hence, I have **used One hot encoding**.
  For non-numerical attributes with just two unique values (as was found with 'default' attribute), we do not need to apply one hot encoding as a label encoding would also serve the same meaning. Hence care has to be taken to not apply one hot encoding to these particular attributes in order to avoid unnecessary increase in number of attributes.
- Since the data has no null or duplicate values, no pre-processing is required in that aspect.
- It was noticed that some attributes like 'previous' take small values and others like 'balance' take larger values. Hence, I thought it necessary to try **Normalization** so as to even out the scales. However, since this happens only in a few attributes, the overall effect of normalizing the data is not very prominent in the results as can be seen.

Answer 2.

To find out the optimal depth limit for the decision tree, I plotted error rate vs maximum depth for both training and test data. We need to choose such a depth that minimizes the test error while also considering that the graph does not overfit by looking at the training error. For the normalized data, as can be seen in Fig1. at Out[26], the test error decreases gradually till a depth of 8 and then starts increasing again while the train error decreases even further. This clearly indicates that the model overfits for a depth greater than 8. Hence, 8 was found to be the optimal depth limit.

Answer 3.

I would choose the Decision Tree classifier here.

Naïve Bayes performs poorly on all fronts when compared to Decision Trees and Random Forests. So, we only **need to choose between Decision tree and random forest**. Note that **Random Forest gives a slightly higher accuracy than the decision tree**. However, **accuracy is not the sole metric of consideration**.

We are trying to classify whether a client will subscribe to term deposit or not. The false negative value indicates the number of clients who actually subscribe but are predicted wrongly while a false positive value

indicates the number of clients predicted to subscribe but who actually do not. Clearly, for any company, they can afford to predict wrongly that a client will subscribe even if he actually doesn't but they cannot afford to predict wrongly that a client will not subscribe when actually he does. Hence it is **more important to decrease the number of false negatives as compared to the false positives**.

To evaluate this, we **compare the recall for positive class for both the classifiers**. The higher the recall, the better it is. Decision Tree gives a recall of '0.39' (see In[34]) on the positive class while Random Forest gives a recall of '0.26' (see In[48]) on the positive class. Since the difference in accuracies is not much, this becomes the deciding factor and hence I would choose a decision tree.

I did not consider the ROC score to choose the better classifier because the ROC score considers both precision and recall in an equal light and just measures how well the classes are separated. But in this specific problem, in my opinion, it is more important to get a higher recall for positive class.

Answer 4.

AUC graph helps us in knowing **how good a model is at distinguishing between the two classes**. An AUC score of 1 indicates that the classifier is working perfectly and is able to correctly classify all points. An AUC score of 0.5 indicates that the classifier is not at all able to distinguish between the classes and is equivalent to guessing the classes at random. An AUC value of 0 indicates that the classifier is classifying the points in a completely opposite manner, meaning it is classifying all actually positive classes as negative and all actually negative classes as positive. Hence an **AUC value greater than 0.5 makes a classifier admissible** for any given classification task.

Answer 5.

Bagging with Decision Tree classifier of **depth 5** gives a **train accuracy of '0.927' and a test accuracy of '0.883'**.

Bagging with Decision Tree classifier of **depth 40** gives a **train accuracy of '0.991' and a test accuracy of '0.884'**.

Hence, the test accuracy remained almost same while the train accuracy went up considerable. This clearly indicates that **the model with depth 40 has overfit** because it performs exceedingly well on its training data but is not able to generalize.

The first model (i.e. with depth 5) **may be** considered to underfit because it is not giving a good accuracy even on the training data.