CSC570AL Machine Learning Assignment 2
Hands on with KNN and Naïve Bayes Classifiers
(45 points)

Problem 1: Applying k-Nearest Neighbors to predict the success of a marketing campaign

For this assignment, we will be using the bank marketing dataset from UCI. The data has 17 attributes and is related to marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Please download and unzip the dataset from here: https://archive.ics.uci.edu/ml/machine-learning-databases/00222/ . The dataset you will be working on is stored in **bank-full.csv.**

Data Exploration

Open the file bank-names.txt and carefully read the attribute information to understand what information is stored in each attribute, what values each attribute can take and so on.

1. (1pt) Download the dataset and store it in a dataframe in R. Note: the attributes are separated by semicolon so make sure you set "sep" option correctly inside read.csv

2. (2 pt) Explore the overall structure of the dataset using the str() function. Get a summary statistics of each variable. Explain what is the type of each variable (categorical(unordered), categorical (ordered), or continuous).

3. (1pt) Get the frequency table of the target variable "y" to see how many observations you have in each category of y. Is y balanced? that is, do you have roughly same observations in y=yes and y=no?

4. (3 pts) Explore the data in order to investigate the association between the target variable y and other variables in the dataset. Which of the other variables are associated with y? Use appropriate plots and statistic tests to answer this question.

Based on your data exploration above, keep the variables you have found to have association with the target variable y and remove the other variables.

Data Preparation:

5. (1pt) Use the command colSums(is.na(<your dataframe>) to get the number of missing values in each column of your dataframe. Which columns have missing values? Note: some variables use "unknown" for missing values. Convert all "unknown" values to NA. You can do so by setting "na.strings" parameter to "unknown" when you read the file using read.csv.

6. (3 pt) There are several ways we can deal with missing values. The easiest approach is to remove all

the rows with missing values. However, if a large number of rows have missing values removing them will result in loss of information and may affect the classifier performance. If a large number of rows have missing values, then it is typically better to substitute missing values. This is called data imputation. Several methods for missing data imputation exist. The most naïve method (which we will use here) is to replace the missing values with mean of the column (for a numerical column) or mode/majority value of the column (for a categorical column). We will use a more advanced data imputation method in a later module. For now, replace the missing values in a numerical column with the mean of the column and the missing values in a categorical column with the mode/majority of the column. After imputation, use colSums(is.na(<your dataframe>) to make sure that your dataframe no longer has missing values.

7. Set the seed of the random number generator to a fixed integer, say 1, so that I can reproduce your work:

```
> set.seed(1)
```

8. (1pt) Randomize the order of the rows in the dataset.

9. (2 pt) This dataset has several categorical variables. With the exception of few models (such as Naiive Bayes and tree-based models) most machine learning models require numeric features and cannot work directly with categorical data. One way to deal with categorical variables is to assign numeric indices to each level. However, this imposes an artificial ordering on an unordered categorical variable. For example, suppose that we have a categorical variable primary color with three levels: “red”,”blue”,”green”. If we convert “red” to 0 , “blue” to 1 and “green” to 2 then we are telling our model that red < blue< green which is not correct. A better way to encode an unordered categorical variable is to do **one-hot-encoding.** In one hot-encoding we create a dummy binary variable for each level of a categorical variable. For example we can represent the primary color variable by three binary dummy variables, one for each color (red, blue, and green) . If the color is red, then the variable red takes value 1 while blue and green both take the value zero.

Do one-hot-encoding of all your unordered categorical variables (except the target variable y). You can use the function one_hot from mltools package to one-hot encode all categorical variables in a dataset. Please refer to https://rdrr.io/cran/mltools/man/one_hot.html . Use option DropUnusedLevels=True to avoid creating a binary variable for unused levels of a factor variable.

Please note that the one_hot function takes a data table not a dataframe. You can convert a dataframe to datatable by using as.data.table method https://www.rdocumentation.org/packages/data.table/versions/1.12.8/topics/as.data.table. Make sure to use library(data.table) before using as.data.table method. You can covert a datatable back to a dataframe by using as.data.frame method https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/as.data.frame

Training and Evaluation of ML models

10. Split the data into training and test sets. Use the first 36168 rows for training and the rest for testing.
11. (2 pt) Scale all numeric features using z-score normalization. **Note: Don't normalize your one-hot-encoded variables.**

12. (3 pts) Use **5-fold** cross validation with KNN **on the training set** to predict the "y" variable and report the cross-validation accuracy. (Please use crossValidationError function in slides 51-53 of module 4 lecture notes and modify it to compute accuracy instead of error. The accuracy is simply 1- error).

13. (2 pts) Tune K (the number of nearest neighbors) by trying out different values (starting from k=1 to k=sqrt(n) where n is the number of observations in the training set (for example k=1,5,10,20 50,100, sqrt(n)). Draw a plot of cross validation accuracy for different values of K. Which value of *K* seems to perform the best on this data set? (Note: the higher the cross validation accuracy (or the lower the cross validation error) the better is the model. You can find an example in slides 54-55 of module 4 lecture notes) Note: This might take several minutes to run on your machine, be patient.

14. (3 pt) Use "knn" function to train a knn model on the **training set** using the best value of K you found above and get the predicted values for the target variable y in the test set.

15. (2pt) Compare the predicted target (y) with the true target (y) **in the test set** using a cross table.

16. (2 pt) Based on the cross table above, what is the False Positive Rate and False negative Rate of the knn classifier on the test data? **False Positive Rate (FPR) is the percentage of all true negative (y="no") observations that the model predicted to be positive (y="yes"). False Negative Rate (FNR) is the percentage of all true positive (y="yes") observations that the model predicted to be negative (y="no"). FPR and FNR should be values in the range [0-1].**

17. (2 pt) Consider a majority classifier which predicts y="no" for all observations in the test set. Without writing any code, explain what would be the accuracy of this majority classifier? Does KNN do better than this majority classifier?

18. (2 pt) Explain what is the False Positive Rate and False Negative Rate of the majority classifier on the test set and how does it compare to the FPR and FNR of the knn model you computed in question 16.

Problem 2: Applying Naïve Bayes classifier to sentiment classification of COVID tweets

For this problem you are going to use corona_nlp_train.csv dataset, a collection of tweets pulled from Twitter and manually labeled as being "extremely positive", "positive", "neutral", "negative", and "extremely negative".
The dataset is from this Kaggle project (https://www.kaggle.com/kerneler/starter-covid-19-nlp-text-d3a3baa6-e/data). I have attached the data to this assignment spec and you can directly download it from canvas.

1. (1pt) Read the data and store in in the dataframe. Take a look at the structure of data and its variables. We will be working with only two variables: OriginalTweet and Sentiment. Original tweet is a text and Sentiment is a categorical variable with five levels: "extremely positive", "positive", "neutral",

"negative", and "extremely negative".

Note: The original tweet variable has some accented character strings. Set **fileEncoding="latin1"** parameter inside the read.csv method to ensure those characters are read correctly.

2. Randomize the order of the rows.

3. (1pt) Convert sentiment into a factor variable with three levels: "positive, "neutral", and "negative". You can do this by labeling all "positive" and "extremely positive" tweets as "positive" and all "negative" and "extremely negative" tweets as "negative". Now take the "summary" of sentiment to see how many observations/tweets you have for each label.

4. (2pt) Create a text corpus from OriginalTweet variable. Then clean the corpus, that is convert all tweets to lowercase, stem and remove stop words, punctuations, and additional white spaces.

5. (2pt)Create separate wordclouds for "positive" and "negative" tweets (set max.words=100 to only show the 100 most frequent words) Is there any visible difference between the frequent words in "positive" vs "negative" tweets?

6. (1pt) Create a document-term matrix from the cleaned corpus. Then split the data into train and test sets. Use 80% of samples (roughly 32925 rows) for training and the rest for testing.

7. (2pt) Remove the words that appear less than 50 times in the training data. Convert frequencies in the document-term matrix to binary yes/no features.

8. Train a Naïve Bayes classifier on the training data and evaluate its performance on the test data. Ues a cross table between the model's predictions on the test data and the true test labels. Be patient, training and testing will take a while to run. Answer the following questions:

    - (1pt) What is the overall accuracy of the model? (the percentage of correct predictions)
    - (3 pt) What is the **precision** and **recall** of the model in each category(negative, positive, neutral) ? precision and Recall are two popular metrics for measuring the performance of a classifier on each class and they are computed as follows:

    Precision = TP/(TP+FP) recall= TP/(TP+FN)

    Where TP is True Positive, FP is false positive and FN is false negative.

    For example, for the "neutral" class, TP will be the total number of neutral observations in the test data that naïve bayes correctly predicted. FP will be the total number of none-neutral observations in the test data that naïve bayes incorrectly predicted to be neutral and FN will be the total number of neutral observations in the test data that naïve bayes incorrectly predicted to be none-neutral.

    Precision for the neutral class answers this question: what percentage of observations that the model classified as neutral are truly neutral? Recall for neutral answers this question: what percentage of truly neutral observations was the naïve bayes model able to predict correctly as neutral?

    We will talk more about different evaluation metrics in module 12.

What to Turn in:

You need to create an R notebooke consisting of your answers to the questions outlined above for problems 1 and 2 together with your R code you used to answer each question.

Your submission must be in two formats:

1. **A .html file which contains the preview of your notebook**. When you click on preview in R studio to preview an R notebook, an html file is created in the same directory as your notebook. You must submit this .html file or your submission will not be graded.
2. **An .rmd file which contains your R notebook**.

Please do not hesitate to email me if you have any question.