CS 506 Spring 2020 - HW0

Introduction to Python

Due: February 18, 2020

1 Parse and Preprocess Data

[3 pts.]
For this part you will have to write Python code to parse data. Specifically, you will use the **arrhythmia dataset** "arrhythmia.data" (you can find a detailed description about the dataset here).
Each line in this dataset corresponds to a patient and contains 280 comma-separated values. **The first 279 are the attributes**, whereas **the last element corresponds to the class** of this patient (an integer ranging from 1-16).
Your goal is to write a **function** that reads the dataset and returns two arrays (X and y), where X contains the attributes for every patient and y the corresponding class.
Be careful! The dataset also contains **missing values** denoted with a question mark '?'. You will need to take care of them and store them as NaN entries in your X array.

```
def import_data(filename):
    """
        Write your code here
    """

    return X, y
```

2 Impute or delete missing entries

(a) [2pts.] As described above, the matrix X will contain missing entries, denoted as NaN. Write a **function** that imputes these missing entries with the **median** of the corresponding feature - column in X.

```
def impute_missing( X ):
    """
    Write your code here
    """
    return X
```

(b) [1pt.] Explain why sometimes it is **better to use the median instead of the mean** of an attribute for missing values.

(c) [1pt.] Another way to deal with missing entries, is to **discard** completely the samples that do not have an entry for every attribute. Write a Python **function** that discards those samples from the dataset.

```
def discard_missing( X, y ):
    """
    Write your code here
    """

    return X, y
```

3 Working with the data

(a) [2pts.] Create a function which shuffles the order of data entries. You might find it useful to import either the numpy or random library for this question and for problems 5(a) and 5(b).

(b) [1pt.] Create a function which calculates the standard deviation of each feature. In general, the standard deviation is denoted σ. It is calculated as $\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}$ where N is the number of data points and $\overline{x}$ is the average value of that feature.

(c) [2pts.] Create a function which removes all entries that contain a value which is more than two standard deviations away from the mean.

(d) [3pts.] Create a function which standardizes all the data points. We do this by transforming x into x' where $x' = \frac{x - \overline{x}}{\sigma}$. What is the time and space complexity for this function?

4 Working with non-numerical data

[3pts.]
Download the Titanic dataset by clicking here. Note you should download the file called train.csv. This dataset contains some features which are not numerical (e.g., gender and embarked). Make a new function similar to import_data() which takes this dataset and returns X, y where the non-numerical values have been replaced with numbers (y should be the "survived" feature). You only need to alter the gender and embarked categories; you should still import the remaining categories which have numerical values (i.e., you do not need to handle the name, ticket, or cabin features but you do need to handle the rest). You should replace each instance of "female" with 0 and all instances of "male" with

1. When dealing with the embarked feature, replace C with 0, Q with 1, and S with 2.

5 Train-test split

(a) [4pts.] Usually in Machine Learning tasks, in order to test the effectiveness of an algorithm in a labeled dataset, we use a part of the dataset for training, and test the efficiency of the learned algorithm on the remaining one.
Write a **function** that gets as input the fraction of the dataset that will be used as the test set (t_f) and **randomly** splits the dataset into train and test sets. Again, you may find the python libraries random or numpy useful for this and the following problem.

```
def train_test_split( X, y, t_f ):
    """
    Write your code here
    """

    return X_train, y_train, X_test, y_test
```

(b) [3pts.] Write a function which divides the data set into three sections (in the future, we will sometimes use what is called a cross-validation set). In addition to the parameters used in the previous question, this function should also take cv_f which is the fraction of the dataset to be used in the cross-validation set.

```
def train_test_CV_split( X, y, t_f, cv_f ):
    """
    Write your code here
    """

    return X_train, y_train, X_test, y_test, X_cv, y_cv
```