CS 6220 Data Mining Techniques – Fall 2016

Course Project Report

Tong Li & Qian Cao

##### Introduction

The project requirement is to design and implement a model, run experiments on a real-world dataset and make a further discussion on the results. The dataset we use is from the UCI machine learning lab which representing 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks throughout the United States. The data contains about 50 attributes such as race, gender, age, HbA1c test result, diabetic medications, etc. Databases of clinical data contain valuable but heterogeneous and difficult data in terms of missing values, incomplete or inconsistent records, and high dimensionality understood not only by number of features but also their complexity. Additionally, analyzing external data is more challenging than analysis of results of a carefully designed experiment or trial, because one has no impact on how and what type of information was collected. Nonetheless, it is important to utilize these huge amounts of data to find new information or knowledge that is possibly not available anywhere.

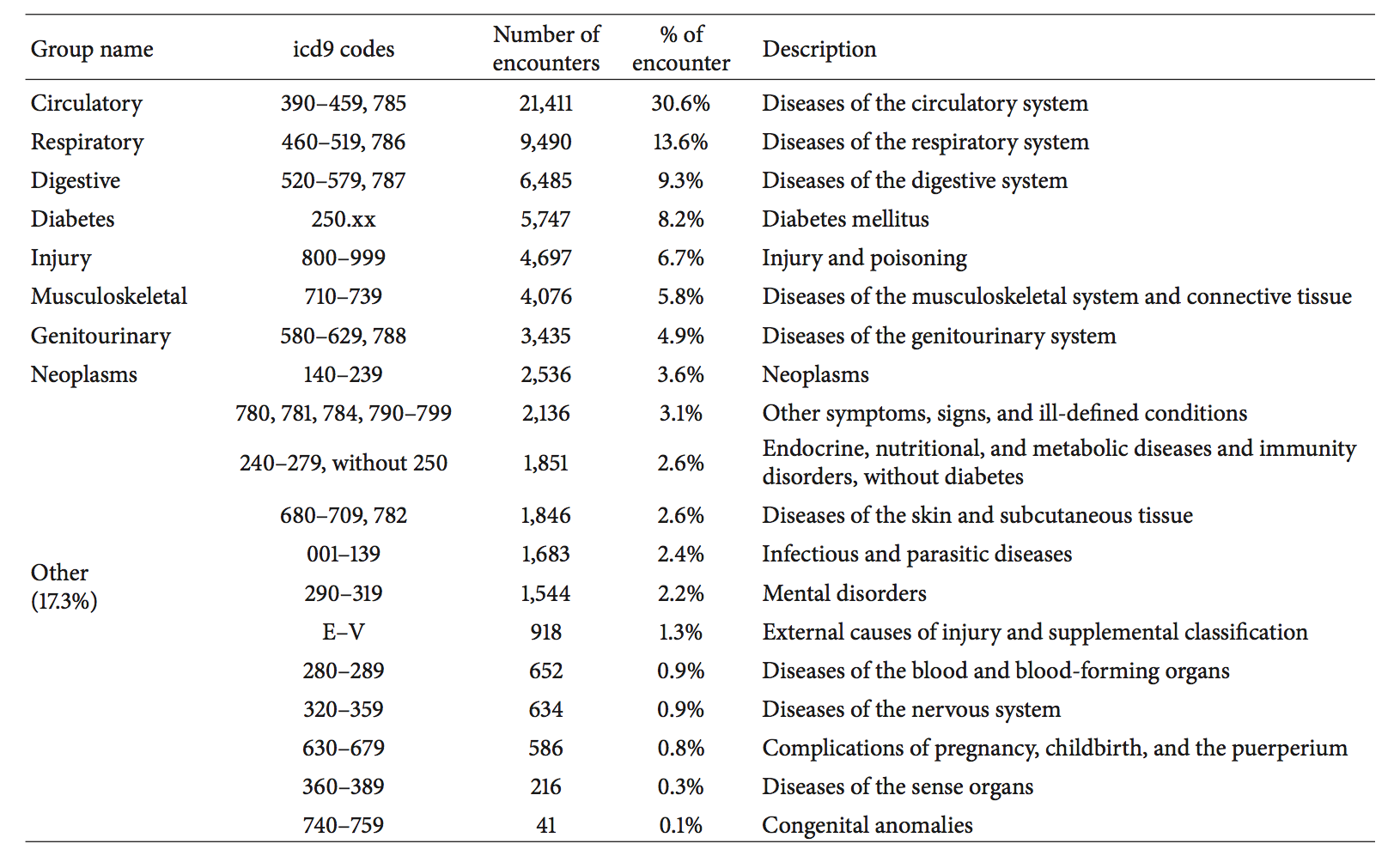
The task is to predict, as accurately as possible, whether a person will be readmitted within 30 days after being discharged from the hospital. To achieve this objective, we analyze the original dataset, compare different algorithms learned in data mining class and then build the model.

##### Preprocess

First, we browsed through the database and the paper called “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records” to understand different features’ meaning in the database. All the instances in original dataset satisfy: (1) It is an inpatient encounter (a hospital admission); (2) It is a “diabetic” encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis; (3) The length of stay was at least 1 day and at most 14 days; (4) Laboratory tests were performed during the encounter; (5) Medications were administered during the encounter.

Since we are primarily interested in factors that lead to early readmission, we relabel the readmission attribute as having two values: “1”, if the patient was readmitted within 30 days of discharge or “0”, which covers both readmission after 30 days and no readmission at all.

Another important thing we find in the dataset is that there are both numeric attributes and nominal attributes. Thus, we need to do combination or extraction works to some instances. For example, features like “diag\_1”, “diag\_2” and “diag\_3” are nominal and can classify in groups in the preprocess code. Also, the features like “encounter\_id” and “patient\_nbr” are regarded as irrelative ones in the final dataset.



**Then, deal with missing data.** The original database contains incomplete, redundant, and noisy information as expected in any real-world data. There were several features that could not be treated directly since they had a high percentage of missing values. In terms of the paper, the missing rate of “Weight” is 97%, “Payer code” is 52% and “Medical specialty” is 53%. These features will not be considered in further analysis. Some sporadic instances which contain missing data of other features will be discarded because the database is large enough to pick good data for analysis.

The preliminary dataset contained multiple inpatient visits for some patients and the observations could not be considered as statistically independent, an assumption of the logistic regression model. We thus used only one encounter per patient; in other words, we considered only the first encounter for each patient as the primary admission and determined if they were readmitted within 30 days. Additionally, we only reserved the encounters that resulted in being discharged from the hospital, corresponding to the objective.

To summarize, our dataset consists of hospital admissions of length between one and 14 days that result in being discharged from the hospital. Each encounter corresponds to a unique patient diagnosed with diabetes, although the primary diagnosis may be different. During each of the analyzed encounters, lab tests were ordered and medication was administered. After the first preprocess, the number of remaining instances is 65187.

**The last step before using algorithms is to handle the imbalanced data.** There are only 5917 class 1 (’< 30’) instances but 59270 class 0 (’NO’ and’> 30’) in the new processed dataset. For the liberal quantity of dataset, we choose the under-sampling method. Then we generate 3 bags which contain the same number of class 1 and class 0 instances. To generate the bag, randomly pick 5917 class 0 instances with replacement to combine with class 1 instances. Finally shuffle the bags to make the instances distribution random. Using these three bags files, we can start to build the model with different algorithms.

##### Algorithm

Based on the final dataset, we choose three of the most popular algorithms in classification for matrix data, which are: KNN, Decision Tree and SVM. Due to their individual characteristics, we choose different features in each part. However, the varying models are based on the same training set and it will generate a more powerful prediction model with majority vote method.

**a. KNN**

1. Find all the numeric features in database can be used in KNN algorithm. The features column ID is [4,9,12,13,14,21,15,16,17]. “Age” feature is a range but the numeric data can be set as minimum. The other features are “time\_in\_hospital”, “num\_lab\_procedures”, “num\_procedures”, “num\_medications”, “number\_outpatient”, “number\_emergency”, “number\_inpatient”, “number\_diagnoses”.

2. Rank the features with filter method. Rank the features with correlation and then greedily increase selected features test with CV method. In the end, find the best accuracy with feature [17,9,4]

3. Model training. Use KNN algorithm with CV method in bag1, with votes of bag2 and bag3. It takes at least 30 minute to run and finally we get the KNN test result of bag1. The accuracy is 57%.

**b. Decision Tree and Random Forests**

The code we use is based on: https://pypi.python.org/pypi/DecisionTree

The features column ID is [2,3,4,18,19,20,41,47,48]. They are “race”, “gender”, “age”, “diag\_1”, “diag\_2”, “diag\_3”, “insulin”, “change” and “diabetesMed”.

1. Recall that decision trees suffer from high variance. This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we may get could be quite different. So, we want a result that has low variance if applied repeatedly to distinct datasets.

2. Bootstrapping is a natural solution, since it is designed as a general-purpose procedure for reducing variance. Bagging is essentially taking repeated samples from the single training set to generate B different bootstrapped training datasets. We then train our model on the bth training set and average all the predictions.

3. Random forests provides an improvement over bagged trees by using a small tweak that decorrelates the trees. As in bagging, we build trees based on bootstrapped training samples, but this time a split in a tree is considered, since averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. Here we use WEKA to finish the prediction.

4. Boosting is another approach for improving the prediction power from a decision tree. Boosting works similarly to bagging except that the trees are grown sequentially each tree is grown using information from previously grown trees.

5. Conclusion: After trying both python code and WEKA to predict class in bag1 (using bag2 to build the model), we find that the highest accuracy is 59% in Random Forests with WEKA. The result of prediction is exported and used for the further vote.

**c. SVM**

The SVM can handle the numeric features with support vector.

1. Transfer the bag1 and bag2 csv files to arff files. The Header of the ARFF file contains the name of the relation, a list of the attributes. In the program, we listed all the features and combine with the numeric data to generate the arff files.

2.Building model with WEKA. Select the SMO classifier and PolyKernel with exponent option then output the result as csv format to get the final result csv file.

##### Conclusion

Finally, the test result is the combination of votes from the three training models. However, the accuracy is still really low, which is only 58%.

The project of training real world database is different. The database is large and it is really tough to reach a high accuracy. It is possible that have better method to train the database such as increase the bags and try more advanced algorithms.