Predicting Patients with Diabetes Type II from EHR Data

CS229 Final Project

Xiaoran Zhang
SCPD
Nuance Communications, Inc.
Burlington, MA
xiaoranz1986@gmail.com

Ruoyu Ding
Stanford University
ruoyud@stanford.edu

***Abstract*—In this paper, we are interested in developing classification methods for predicting patients with Diabetes Type II from EHR data. We first compare several common binary classification methods for this task – SVM, Gradient Boosted Trees and Neural Nets. We optimize the parameters for the best three individual systems, then implement weighted voting technique between these systems to perform system combination prediction.**

Keywords—binary classification; diagnosis prediction

I. INTRODUCTION

Due to recent attentions and efforts in healthcare initiatives, EHR (Electronic Health Records) is quickly becoming a requirement per patient encounter and is increasingly being adopted by both physicians and hospitals. One of the possible uses for this increase of structured data in the form of EHR is to predict prevalent diseases such as diabetes. Type II diabetes in particular, is present in 8.5% of the population in the US, being able to predict diabetes diagnosis from past hospital visits is a step forward to early detection of diabetes type II as well as understanding its relations with other diagnosis and risk factors.

In this paper, we use the de-identified EHR data provided by EHR vendor Practice Fusion in their kaggle challenge. We will first discuss feature selection and normalization; then proceed to compare individual binary classification methods we employed, such as Support Vector Machines (SVM), Neural Nets, and Gradient Boosted Trees. We optimize the parameters for each individual method then discuss the performance of voting combination between the three systems.

II. METHODOLOGY

The data for this binary classification task is provided by Practice Fusion in their kaggle challenge. The EHR data is organized into database tables with patient specific information, visit specific physical examination information, lab panel results and diagnosis. The data is also comprised of both numerical and categorical data, both of which require treatment before being used in our classifiers.

A. Feature Processing and Selection

1) Numerical Features: Numeric data for this task is comprised of mainly patient per-visit physical examination data, namely: height, weight, BMI, temperature, respiratory rate, systolic and diastolic blood pressure, per-patient data, namely: age. We combine numeric data across visits on a per-patient basis by taking the median value of all visits. Median is used instead of mean because there are frequent outliers for one or two of a patient's visits. After combining the data on a per-patient basis, we discard temperature and respiratory rate because the lack of data (values are missing for most patients) and we also discard height and BMI data since height has many outliers which leads to noisy data for BMI. We keep weight to represent obesity, systolic and diastolic blood pressure to represent hypotension. Both of which are important risk factors for diagnosing diabetes type II.

a) Missing Values: For missing numeric values, we substitute the missing value with the median of all of the patients in the training set.

2) Categorical Features: We have two important sources of categorical data: list of medications prescribed to the patient per-visit and list of diagnosis associated with the patient per-visit. Since we do not have temporal information on the visits and since all of the visits are recent (within the past 1 years), we combine medication and diagnosis data across all visits per-patient.

a) Data Representation: Since we have a wide range of diagnosis and mediation and majority of them have low frequency of occurence, if we represented every diagnosis and medication as a feature, we would have a large feature space with sparse data. Thus, we pick the top N most discriminative diagnosis and medication to produce binary feature vectors for each of these diagnosis/medication. (*i.e.* 0 indicates absence and 1 indicates presence). The discriminativeness of the categorical feature is determined by the difference in the prevalence of the diagnosis/medication in patients with diabetes type II and patients without diabetes type II. The prevalence in any one class is measured by the percentage of

patients with the diagnosis/medication in that class. This works quite well, for example, the top two most discrimitive conditions includes hypertension and lipoprotein deficiencies, both of which associates closely with diabetic patients.

*b) Data Normalization:*The names of the diagnosis are based on ICD-9 coding standard and the names of the medications are usually brand information with routes. Both of these are more fine grained than what we would like for this task and therefore we normalize both as follows. For diagnosis, the ICD-9 code provides the ontological structure from which the disease is derived. For example, "Vascular dementia with depressed mood" has an ICD-9 code of 290.4 and "Senile dementia with delusional or depressive features" has an ICD-9 code of 290.2, both of which are child nodes of "Snile dementia, uncomplicated" with an ICD-9 code of 290 in the disease ontology. For our purposes, we collapse these diagnosis to their parent node (*i.e.* ICD-9 code of 290) and consider them to be the same diagnosis. We normalize the medication features by taking out the route information (e.g. topical vs. oral tablet) and mapping the brand name medication to its active ingredient. (*e.g.* "Plavix oral tablet" maps to "clopidogrel"). We also normalize the casing on these before converting data to the representation described above.

3) Additional Features: We also compute two additional features: length of the medication list per-patient and length of the diagnosis list per-patient.

B. Classification Model Formulations

For our binary classification task, we choose to examine three machine learning methods with the ability to deal with higher dimensional features and combinations of features, since our initial results with linear and low order machine learning methods gave poor results due to simplicity. For Support Vector Machine and Gradient Boosted Tree, we use the implementation provided by the libsvm and gbm R packages, implement our own neural network algorithm in R.

1) SVM: We use SVM with a polynomial kernel and L_2-norm soft margin classification (corresponds to *C-classification in libsvm*). This gives us control over tuning the regularization parameter (*i.e. cost)* as well as the degree of the polynomial (*i.e. degree)*.

*2) Gradient Boosted Trees:*For binary classification, we use the Bernoulli lost function. We tune the shrinkage parameter as well as the interaction parameter (*i.e. interaction_depth)* to prevent between overfitting and to represent non-linearity.

3) Neural Network: Our implementation of the neural network uses the back propagation technique to update the weights. Since this is a binary classifier. We train the model by minimizing the cross-entropy error function, where t_i represents the target (truth) value of the i^{th} feature vector and y_i represents the output from the model. We use the logistic function as the activation function. We also scale numeric input data with the *scale()* function in R.

$$E_m = \sum_{i=1}^{m} t_i \ ln\, y_i + (1 - t_i)\, ln(1 - y_i).$$

We tune the number of hidden layers in our neural network. (*Note:* In our attempt to tune the regularization parameter, we found the results are highly inconsistent between runs.)

4) Weighted Voting System Combination: For each test sample, we obtain the prediction results from each of the optimized classifiers and employ a voting scheme to achieve a system combination.

III. Experimental Setup

A. Data Preparation

We have a total of 9943 patient data feature vectors, with a total of 21 features each. We divide this data into disjoint datasets by random sampling: train, dev and test sets.

1) Training set (train): The training set is comprised of 9343 labeled patient feature vectors, with label 0 corresponding to a non-diabetic patient and label 1 corresponding to a diabetic patient.

2) Development set (dev): The dev set is comprised of 300 patient feature vectors and is used as the test set for parameter tuning and algorithm development.

3) Test set (test): The test set is comprised of 300 patient feature vectors and is used as the blind set only for final evaluation of the systems.

B. Evaluation Metrics

We use three metrics to evaluate each of the classifier and the overall system: precision (*P*), recall (*R*) and F-measure (*F*). Precision represents the accuracy of our prediction for the samples where we predicted class 1 (patient has diabetes type II). Recall represents the percent of correctly classified patients with diabetes type II out of all the patients with diabetes type II. F-measure is the harmonic mean between precision and recall. We aim to maximize the F-measure for both optimization and final result.

C. Model Optimization

For each of the three models, we optimize the model by tuning the associated model parameters. This is accomplished by training the model on the training set with a range of values (usually 4-5) for the parameter of interest while holding the rest constant. We choose the parameter that gives us the best F-measure on the 300 patient data development set as the optimized parameter. We also measure the performance multiple times (for GBT and NN) to ensure that the optimal parameter results are consistent between runs.

IV. Results and Discussion

A. Model Optimization Results

1) SVM Parameter Optimization: For the L_2-norm soft margin implementation of SVM, we tune the regularization parameter as all as the degree of the polynomial kernel. We found that a polynomial of degree 4 and cost of 10 produces the best results without underfitting or orderfitting the data, the results are shown in the table below.

TABLE I. SVM Parameter Optimization Results

Cost = 10	**Regularization Parameter (Cost)**		
	Precision	***Recall***	***F-measure***
train	*0.955*	*0.535*	*0.685*
dev	*0.444*	*0.296*	***0.356****
Cost = 30			
train	*0.960*	*0.577*	*0.721*
dev	*0.357*	*0.278*	*0.313*
Cost = 50			
train	*0.968*	*0.593*	*0.735*
dev	*0.311*	*0.259*	*0.283*
	Degree of Polynomial		
degree = 2	***Precision***	***Recall***	***F-measure***
train	*0.721*	*0.190*	*0.301*
dev	*0.526*	*0.185*	*0.274*
degree = 3			
train	*0.868*	*0.367*	*0.515*
dev	*0.382*	*0.241*	*0.295*
degree = 4			
train	*0.919*	*0.439*	*0.594*
dev	*0.421*	*0.296*	***0.348****
degree = 5			
train	*0.937*	*0.474*	*0.629*
dev	*0.361*	*0.241*	*0.289*

2) GBT Parameter Optimization: We optimize the shrinkage parameters (*i.e.* regularization) and interaction depth of the trees (*i.e.* represents degree of non-linearity), we default the number of classifiers to 100. We find that at shrinkage of 0.2 and interaction depth of 3, we achieve the best results for the development set.

TABLE II. GBT Parameter Optimization Results

Shrinkage = 0.1	**Shrinkage Parameter**		
	Precision	***Recall***	***F-measure***
train	*0.667*	*0.275*	*0.389*
dev	*0.469*	*0.278*	*0.349*
Shrinkage = 0.2			
train	*0.686*	*0.333*	*0.448*
dev	*0.500*	*0.333*	***0.400****
Shrinkage = 0.3			
train	*0.704*	*0.355*	*0.472*
dev	*0.412*	*0.278*	*0.333*
	Interaction Depth		
degree = 2	***Precision***	***Recall***	***F-measure***
train	*0.669*	*0.302*	*0.416*
dev	*0.424*	*0.259*	*0.322*
degree = 3			
train	*0.679*	*0.328*	*0.442*
dev	*0.500*	*0.352*	***0.413****
degree = 4			
train	*0.686*	*0.355*	*0.468*
dev	*0.457*	*0.396*	*0.369*

3) NN Parameter Optimization: We optimize our neural network by experimenting with both the regularization term λ and the number of hidden layers for model complexity. We find that at 8 hidden layers we achieve the best performance on the development set. The results for λ is too inconsistent between runs, thus by default, we set $\lambda = 0$.

TABLE III. NN Parameter Optimization Results

	Number of Hidden Layers		
n_hidden = 7	***Precision***	***Recall***	***F-measure***
train	*0.679*	*0.404*	*0.507*
dev	*0.406*	*0.241*	*0.302*
n_hidden = 8			
train	*0.664*	*0.382*	*0.485*
dev	*0.475*	*0.352*	***0.405****
n_hidden = 9			
train	*0.669*	*0.343*	*0.454*
dev	*0.469*	*0.278*	*0.349*

B. Results of Optimized Classifiers on Test Set

We evaluate each of the three models on our test set with the optimized parameters. The test set is unseen data to both the training and parameter tuning.

TABLE IV. Results on Test Set for Optimized Classifiers

Classifier	**Performance on Test Set**		
	Precision	***Recall***	***F-measure***
SVM	*0.316*	0.146	*0.200*
GBT	*0.619*	0.317	*0.419*
NN	*0.522*	0.293	*0.375*

One thing to note from this result is that all of the models have consistently poor recall. This is largely because we have highly unbalanced class data, that is, the number of patients in the training set without diabetes (*i.e. negative class*) is roughly 4 times the number of patients in the training set with diabetes (*i.e. positive class*). At a clinical setting in particular, we may choose to favor a higher recall over a higher precision because for early diagnosis of critical conditions and diseases, we would rather have a false alarm than non-detection.

We can improve recall and F-measure by up-weighing the samples for patients with diabetes, note that in general there is a trade-off between the precision and recall depending on the weights we choose for the positive and negative class samples.

We plot the F-measure performance for the development set for each classifier at different levels of positive class up-weighing. (*i.e.* positive vs. negative class weight ratio={*1:1, 1.5:1, 2:1, 2.5:1, 3:1, 4:1, 5:1*}).

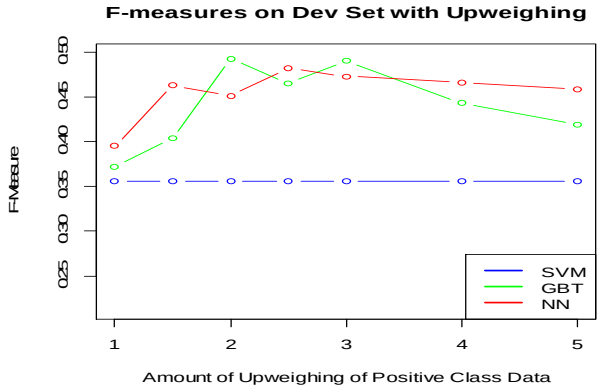

Fig. 1. The figure shows the F-measure with various upweighing ratio for the different classifiers.

From Figure I, we can see that up-weighing the less representative positive class samples does increase the F-measure in both GBT and NN classifiers on the development set. This is because we are gaining more recall in proportion to losing precision. SVM performance remained constant because we did not gain any support vectors by up-weighing the data. We then retrain the models based on the optimal up-weighing for each model and test on the test-set. For NN, the optimal up-weighing is at 2.5:1 and for GBT, the optimal up-weighing is at 2:1.

TABLE V. RESULTS ON TEST SET FOR AFTER UPWEIGHTING

Classifier	**Performance on Test Set**		
	Precision	***Recall***	***F-measure***
SVM	*0.316*	*0.146*	*0.200*
GBT (2:1)	*0.412*	*0.512*	*0.457*
NN (2.5:1)	*0.350*	*0.512*	*0.416*

We were able to improve the F-measure and Recall for both GBT and NN. GBT is the best performing classifier out of the individual classifier, we can visualize the significance of contribution of each feature as modeled by the GBT.

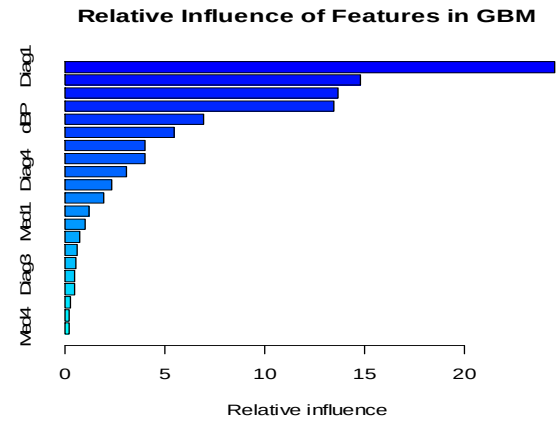

Fig. 2. The figure shows the relative influence (in percent) for each of the features.

As expected, Diag1 (Hypotension), Diag2 (Lipoprotein deficiencies), Age, D/S Blood Pressure and Weight are among the most influential features for predicting diabetes type II in patients.

We also combine the systems by weighing the output of each classifier with ratio between its F-measurement and total measurement. We use this as a proxy for determining the 'goodness' of the system. This is not the same as the conventional majority voting system which needs per-sample confidence scores from each of the classifiers, we do not have such information available.

TABLE VI. RESULTS ON TEST SET FOR COMBINED SYSTEM

Classifier	**Performance on Test Set**		
	Precision	***Recall***	***F-measure***
Combined System	*0.362*	0.415	*0.386*

The combined system performs worse than the best systems, this means that we do not gain information from this type of system combination, this means weighing per classifier is uneffective since the systems do not complement each other.

FUTURE WORK

For future work, we can build classifiers to output confidence for each of the test samples and weigh the classifier result per sample. Also, we could add more features to try to improve both precision and recall.