Predicting Time Spent with Physician

Jim Zheng JIMZHENG@STANFORD.EDU
Stanford University, Computer Science Dept., 353 Serra Mall, Stanford, CA 94305 USA

Ioannis (Yannis) Petousis PETOUSIS@STANFORD.EDU
Stanford University, Electrical Engineering Dept., 353 Serra Mall, Stanford, CA 94305 USA

Ye (Scott) Cheng SCOTTCHENG@STANFORD.EDU
Stanford University, Computer Science Dept., 353 Serra Mall, Stanford, CA 94305 USA

Abstract

The goal of this project is to predict how much time a physician should be spending with a patient, given the patient's injury, referral status, and other reported, non-identifiable medical information. This may be used as an additional factor for measuring and predicting physician efficiency alongside existing methodology. We implement linear regression and a variety of classification techniques on a national dataset of Emergency Room (ER) patients and outpatients. We use cross validation to select the optimum set of features. In addition, we implement the k-Means algorithm to generate new features for our supervised learning algorithms. Our final models provide good predictions of time spent with physician with a mean error of around 7 minutes.

1. Introduction

Current literature on measuring clinical efficiency measure cost and patient feedback efficiency, both of which have been shown to be correlated with a patient's length of stay (Kroch et al., 2007). Among the many subdivisions of the length of stay, one of the most important for quality patient care is how much time a patient spends with a physician. Techniques in current literature for predicting length of stay include multi-label, multi-class for supervised learning. Specifically, length of stay is transformed into a binary labeling of "short" or "long" with the separation between "short" and "long" set specific to the study (Snyder, 2010). Studies also try to discretize by hourly intervals and use decision trees to discretize bins. Existing studies do not attempt, however, to directly predict length of stay in minutes, possibly because there is no need for such precision in existing studies where the variation in length of stay is high. The goal of our project is to predict, given a set of features that describe the circumstances of a patient visit, how much time (in minutes) a physician is likely to spend with them.

2. Data and Features

Our data comes from the National Ambulatory Medical Care Survey, collected by the National Center for Health Statistics. This annual survey samples physician-patient interactions inside hospital ER and outpatient departments across the US. Data is collected via a predefined sampling method provided by the Center for Disease Control (CDC) that spans a wide variety of regions and clinical environments (NAMCS 2009; NAMCS 2010). This is a rich dataset with over 30,000 rows per annual report (dated from 2010 back to 1992), and over 400 features recorded.

2.1. Raw Data Structure

Some of the features captured by the dataset include:

- age of patient
- day of week the appointment took place
- is the physician the patient's primary physician
- major reason for visit
- location (metropolitan or not, and which US region)
- has a computer for viewing patient info
- drug type prescribed
- patient's Body Mass Index (BMI)

It's worth noting that some of these features are continuous (age of patient, BMI), some are discrete, multi-valued (day of week, drug type), and some are boolean (e.g. is primary physician). This meant that the type

of regression that we ran could operate only on a subset of all features, and extensive feature processing was necessary to obtain the best set of features.

With respect to time spent, several facts are also noteworthy. First, time with physician is rounded to be between 0 and 240 minutes; records with times spent over 240 minutes are capped at 240. Second, since entries are manually recorded, times are frequently arbitrarily rounded to the nearest 5 minutes.

2.2. Preprocessing

Data was collected by downloading and extracting a set of features listed in the documentation for each year. Each dataset contained a slightly different set of features; this required us to search for a set of features that stayed consistent from year to year.

Binomial Smoothing: After running some early data analysis, we saw that quite a few features that were semantically binomial (e.g. is patient referred) were in fact represented as multinomials with values 0-9 to indicate types of errors in the recording process. For these features, we unified error values as 0, so that 1 indicates existence, and 0 indicates that the feature was either not present or untrue for the patient.

Label smoothing: We also realized that the distribution of the time spent with physician field was skewed towards large values at multiples of 5 minutes, and large dropoffs for non-multiples of 5. This reflects the fact that a good amount of time, on-the-field recording of time spent with physician is estimated and rounded to the nearest 5 minutes. To mitigate this concern, for logistic regression, we discretize time with physician into multiples of 10 to smooth out our dataset. We also found that more than 92% of the data examples have one or more invalid feature values, which are generally unrecorded data. If we include these features or training examples as they are, the invalid values would largely disturb the linear models. We came up with two strategies to deal with this issue. In the first approach, we take out all the examples with invalid values, and only train and test on the examples where all values are valid. This way, we make sure that all the data examples being used are real and meaningful. In the other approach, instead of dropping invalid examples, we try to fix them by filling in the average value of the attribute over the entire dataset. We did two things to improve the second approach. First, to ensure the representativeness of the average attribute values, we remove the features for which too many examples have invalid values. For example, if 1/5 of the examples have invalid values for a particular feature, we would not look at this feature. Second, we remove the data examples with relatively more invalid attribute values. For example, if a data example has more than 1/15 of invalid features, we consider it as unreliable and therefore remove it from the dataset.

2.3. Unsupervised Learning and Feature Generation

We implemented the K-means algorithm to identify if the hospitals can be grouped into clusters based on parameters such as:

- geographical region
- public or private practice
- percentage of income derived from the state / research institutions / private insurance / patient payments
- the provision of electronic billing and electronic prescriptions.

The number of centroids was varied and the total error (i.e. the sum of euclidean distances of each example from its closest centroid) was calculated. We found that $K = 6$ and above gives a low error. We will subsequently try to define new features by calculating the euclidean distance of a new training example from each of the clusters. The new features will be fed to the supervised learning algorithms.

2.4. Principal Component Analysis

The Principal Component Analysis (PCA) algorithm was used to exploit any dependencies in the input data by projecting them on a lower dimensional space (the number of dimensions was varied). They were then fed into the supervised learning algorithms and the predictive performance was compared to that with the original data.

3. Linear Regression Models

Our first approach is to apply linear regression models on our dataset to predict the time with physician. The prediction involves the following steps:

1. Incorporate custom features, including non-linear transformations and features learned from K-Means
2. Set up linear regression model
3. Select features using the linear model

We will now describe each step in more detail.

3.1. Adding Custom Features

To improve the results of linear model predictions, we added some custom features to the dataset. The first

type of custom features is non-linear transformations of continuous features. Since linear regression models only look at linear relationships between the target variable and feature, in order to reflect non-linear relationships in the data in linear ways so that linear models can capture them, we apply non-linear transformations on the continuous feature values of data examples. Specifically, for continuous features (e.g. age, height, weight), if the value is x, we add the following values as additional features to the example: x^2, x^3, $\sqrt{x}$, $\sqrt[3]{x}$.

Another class of features that we add to the data for linear models are learned from K-means clustering. Similar method of combining supervised and unsupervised learning is adopted in previous works (Coates et al., 2011). Specifically, after computing k clusters out of the dataset, we add k additional features to each data example, where each feature represents the example's distance to a cluster centroid.

3.2. Linear Regression Models

We applied four types of linear regression models: linear regression, ridge regression, lasso regression, and elastic net. We use basic linear regression as a baseline, and try different variations to see which one fits the data best. In particular, since there are so many features in the dataset (we use 49 features for linear regression models, which does not include custom features from non-linear transformation and K-means), chances are that not all of the features are relevant to the target variable. Therefore we try lasso regression which predicts sparse coefficients.

In ridge regression, lasso regression and elastic net, we determine the parameters using cross validation. For example, in ridge regression, we try to optimize $\min_w ||Xw - y||^2 + \alpha\ ||w||^2$, and we set α by running cross validation using ridge regression model with different values of α and take the optimal value.

We will see the details of how these models perform in section 5.

3.3. Feature selection

We apply feature selection before training the linear model, which prevents overfitting and makes training and predicting faster. Specifically, we apply cross validation to determine the number of feature to maintain, and use recursive feature elimination to choose the set of features to use in our model.

4. Classification

Our second major approach for supervised learning was to discretize physician time into bins. First, we designated a number of bins and a distribution of labels in each bin, and ran feature selection to reduce our feature set. Next, we iteratively refined our features by smoothing, combining certain common features, and generating new ones via unsupervised learning techniques to get better predictions. We also experimented with other classifiers to try and eliminate feature bias.

We used the following algorithms in our analysis: Multinomial Naive Bayes, Weighted Logistic Regression, Random Forest Classifier, Forward Feature Selection, Univariate Feature Selection (using ANOVA F-value as a scoring function). We implemented Naive Bayes and Forward Feature Selection by ourselves using numpy. We used pythons scikit-learn library (Pedregosa et al., 2011) for all other algorithms. Weighted logistic regression is done by scikit-learn using an One vs All (OvA) approach, rather than true multiclass logistic regression.

4.1. Naive Bayes

As an initial benchmark, we implemented discretized, multinomial Naive Bayes with Laplace Smoothing. We trained our model on the 2009 dataset and tested using the 2010 dataset, and initially chose our number of bins to be 10 (i.e. our interval was 10, and ranged from 0 to 90). Naive Bayes was quick to implement and gave us a first baseline for error. Moreover, the Naive Bayes assumption is a fair assumption to make for a large subset of our features, since the dataset combines patient, physician, region, and facility information, all of which are expected to be weakly correlated. For discretized Naive Bayes, our approach can approximate the use of the true probability density function given the appropriate selection of interval and number of classes contained in each interval (Yang & Webb, 2009).

Naive Bayes yielded poor initial performance ($accuracy < 0.03$) on the initial set of features for the 2009 dataset; predictions were skewed right, even though the dataset was skewed left. Using feature selection eliminated the "drugtype" features, which, when included, were responsible for the right bias. Our own implementation of forward feature selection (selecting $N = 10$ features) produced an accuracy of about 0.28. We suspected that this had to do with a latent bias in the features we were selecting for. To test this suspicion, we ran scikit-learns Random Forest Classifier on the initial set of features and produced an accuracy score of 0.28. This prompted

us to focus on feature generation as a next step in improving classification accuracy.

4.2. Feature Generation and Error Analysis

Since initial benchmarks gave rather poor performance, we attempted to gain better insight into the features. We found that many features were unable to distinguish between multiple labeling, since there was much noise. Thus, we eliminated these features altogether. Next, since we were uncertain about the target number of features for forward feature selection, we opted for filter feature selection using the ANOVA F-test statistic as our scoring function. Using this type of feature selection, we reduced our analysis down to only four features. Feeding this narrowed feature set into Naive Bayes boosted our accuracy to 0.49, and random forests accuracy to 0.48.

Residual analysis showed we were predicting only three distinct intervals: 10-20, 50-60, and 70-80 bucket. This meant that we were mispredicting the 0-10, 20-30, and 30-40, which accounted for about 43% of our data. Here, we tried a combination of techniques to reduce this bias and boost accuracy with varying degrees of success:

Combining binary features: We tried to combine several features into a binomial feature in order to create better separation of labelings among our feature set. This boosted our accuracy by 0.02.

Combine train and testing dataset: In an effort to reduce randomization, we combined the 2009 and 2010 datasets into one dataset, and randomly chose a 70-30 split for our training and testing datasets, respectively. This yielded a change in accuracy of about 0.01 on average, depending on the permutation.

Weighted logistic regression: Logistic regression was tried with class weights of 10: 0.5, $\langle rest \rangle$: 1, in an effort to reduce overfitting to the 10-20 bucket. This decreased accuracy by about 0.05, as true positives with the 10-20 labeling were now being mispredicted.

Binary Labeling: Our final attempt was to discretize times into only two buckets, and by hour, according to existing literature. Using these weaker predictions, we were predictably able to improve our performance to over 98% accuracy with labelings.

5. Results

This section discusses our results from both linear regression models and classification models. As a baseline, in our dataset, if we predict all the data examples using the mean of target variable, the resulting mean absolute error (MAE) is 8.17 minutes.

Include K-means features	Include	Not include
Linear Regression	7.20	7.29
Ridge Regression	7.07	7.22

Table 1. Linear regression models: MAE (minutes) from 10-fold cross validation (K=50)

5.1. Linear Regression Models Results

Linear regression models are able to make predictions with an MAE of around 7.1 minutes. We now discuss the different parameters and factors that affect the performance of linear regression models.

We first compare the performance of different linear models. Lasso regression and elastic net try to establish the model on fewer features. While we initially thought this could work well since our data has a large number of features, not all of which are directly relevant to the target variable, it turns out that lasso regression and elastic net yield poor performance. For the other two linear regression models we tried, ridge regression produces better results than linear regression.

We found that including features learned from K-means dramatically improves the performance of linear regression. The improvement is particularly significant when K is relatively large. Table 1 compares the results with features learned from K-means and the results without these features.

5.2. Classification Results

Table 2 compares the results of different classification models, where bucket size is set to 10 minutes. As mentioned in section 4, increasing bucket size will significantly improve classification accuracy, but we are more interested in more fine-grained predictions.

	Accuracy (percent)	MAE (minutes)
Multinomial NB	48.5	7.30
Random Forest	47.8	7.56
Logistic Regression	47.6	7.38

Table 2. Classification: Accuracy and MAE

For classification, it was difficult to see how accuracy correlated with strength of prediction for a continuous variable. We wanted to see, for example, whether our classifier made lots of mispredictions that were off by 10 and a few off by 80 vs consistently off by 20. We thus used MAE as a metric of performance in classifi-

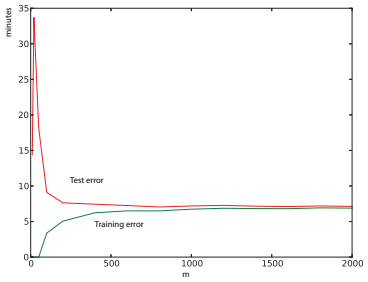

Figure 1. Ridge regression: training error and test error over m

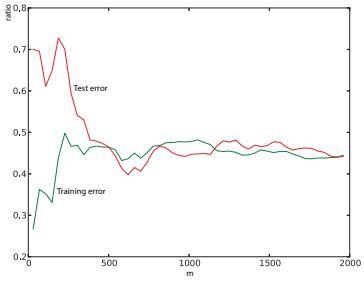

Figure 2. Multinomial Naive Bayes: training error and test error over m in terms of accuracy

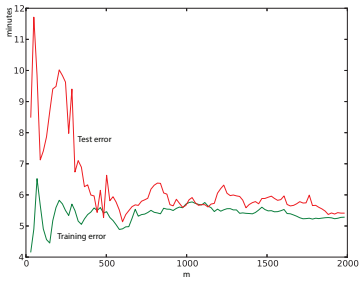

Figure 3. Multinomial Naive Bayes: training error and test error over m in terms of MAE

cation:

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{\hat{y}_i \neq y_i\}(E[\hat{y}_i] - y_i)$$

In addition, we assume a roughly Gaussian distribution over the bins, which is a roughly good approximation since most data points are concentrated in multiples of 10; thus, the expectation is the mean of the bin.

6. Conclusion and Future Work

Our study represents a first attempt at directly trying to predict "time spent with physician", to minutes accuracy. Our attempts at making stronger predictions with linear regression yielded an average error that was better than the average error obtained from predicting the mean. In addition, if we weaken our predictions to match those found in current literature, we are able to obtain over 98% accuracy in classifying time spent with physician.

Further studies could continue improving upon our classification and regression techniques by investigating the following:

Interval and Interval Distribution Selection: We experimented with several interval numbers of varying weakness. The weakest we tried was 5 minutes' intervals, the most 60 (hourly). We also tried to restrain and randomly sample examples that fell into each bucket to ensure an even distribution. A more principled approach would have automated over a range of possible intervals and samplings and calculated scorings for each pair of (interval, sampling).

Training on larger dataset At the time of testing, we were unable to access documentation for the datasets starting from 2003 to 2008, which prevented us from gathering more training data. Using more training data could potentially improve our linear regression predictions; since our linear regression model eliminated mostly-null rows and thus had only about 10% of the original training set with which to train.

References

Coates, Adam, Ng, Andrew Y, and Lee, Honglak. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.

Kroch, E.A., Duan, M., Silow-Carroll, S., and Meyer, J. A. Hospital performance management improvement: Trends in quality and efficiency, 2007.

NAMCS 2009. National ambulatory medical care survey, 2009.

NAMCS 2010. National ambulatory medical care survey, 2010.

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pope, G.C. and Kautter, J. Health care financing review. pp. 31–43, 2007.

Snyder, Ashley M. Data mining and visualization: real time predictions and pattern discovery in hospital emergency rooms and immigration data. Technical report, DTIC Document, 2010.

Yang, Ying and Webb, Geoffrey I. Discretization for naive-bayes learning: managing discretization bias and variance. *Machine learning*, 74(1):39–74, 2009.