Evaluating Ability of Big Data Models to Predict Outcomes of Patients with Diabetes

---

Introduction **2**

Dataset Description **3**

Methodology - Overview **6**

Methodology - Model Analysis **8**

K-Nearest Neighbors 8

Neural Networks 8

Decision Tree and Random Forest 9

Support Vector Machines 10

Results **11**

K-Nearest Neighbors 11

Neural Networks 13

Decision Tree and Random Forest 16

Support Vector Machines 19

Comparison of Results 20

Conclusion **21**

References **23**

Introduction

Hyperglycemia (diabetes) is one of the most prevalent medical conditions affecting adults today. With this condition often requiring long-term care and expenditure of significant resources, it has become increasingly important to be able to accurately assess likely patient outcomes based on observed lab test results, pre-existing data, demographic information, and medical history. I define patient outcomes as the length of time required to stay in the hospital, and whether readmission to the hospital was required within the period of time that the patient was tracked.

The ability to accurately predict patient outcomes based off of available data provides for the ability to efficiently allocate resources (i.e moving patients who are likely to require longer stays and have high risk of readmission to better equipped long-term care outpatient facilities) and provides patients with an accurate understanding of what their likely prognosis is.

To address this issue, I aim to utilize a two-pronged approach. First, I will investigate which machine learning models are most effective at predicting outcomes for past data points, when provided with all data present in this dataset (descriptive analytics). This will give us insight into the relative strengths and weaknesses of the different models tested. Second, I will investigate a series of machine learning algorithms and models to develop a classifier/regressor that could be used to predict outcomes for future potential data points (prescriptive analytics). For this approach, I will remove variables that are endogenous and remove any variables that medical professionals will not have access to at prediction time.

Once the classifiers with the highest accuracy and predictive power have been determined, I will further investigate into the reasons behind why these results hold. I will explore characteristics of the data set that make it particularly suited for the best performing models, and identify situations in which certain models perform better than others.

Dataset Description

The data I will be using for this investigation was provided by the UC Irvine Machine Learning Repository. This data was collected by the Center for Clinical and Translational Research at Virginia Commonwealth University. The dataset contains 10 years of patient information (101,767 individual data points) from 1998 to 2008 at 130 US hospitals and medical care providers. Each data point fulfills the following criteria:

1. It is an inpatient encounter (hospital admission)
2. It is a diabetic encounter. This is defined as any encounter in 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

A table of the attributes included in this dataset is presented below

Feature Name	**Type**	**Description**
Encounter ID	Numeric	Unique identifier of an encounter
Patient number	Numeric	Unique identifier of a patient
Patient number	Numeric	Unique identifier of a patient
Race	Nominal	Values: Caucasian, Asian, African American, Hispanic, and other
Gender	Nominal	Values: male, female, and unknown/invalid
Age	Nominal	Grouped in 10-year intervals: [0, 10), [10, 20), …, [90, 100)
Weight	Numeric	Weight in pounds.
Admission type	Nominal	Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available
Discharge disposition	Nominal	Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available
Admission source	Nominal	Integer identifier corresponding to 21 distinct

		values, for example, physician referral, emergency room, and transfer from a hospital
Time in hospital	Numeric	Integer number of days between admission and discharge
Payer code	Nominal	Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay
Medical specialty	Nominal	Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon
Number of lab procedures	Numeric	Number of lab tests performed during the encounter
Number of procedures	Numeric	Number of procedures (other than lab tests) performed during the encounter
Number of medications	Numeric	Number of distinct generic names administered during the encounter
Number of outpatient visits	Numeric	Number of outpatient visits of the patient in the year preceding the encounter
Number of emergency visits	Numeric	Number of emergency visits of the patient in the year preceding the encounter
Number of inpatient visits	Numeric	Number of inpatient visits of the patient in the year preceding the encounter
Diagnosis 1	Nominal	The primary diagnosis (coded as first three digits of ICD9); 848 distinct values
Diagnosis 2	Nominal	Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values
Diagnosis 3	Nominal	Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values
Number of diagnoses	Numeric	Number of diagnoses entered to the system
Glucose serum test result	Nominal	Indicates the range of the result or if the test was not taken. Values: “>200,” “>300,” “normal,” and “none” if not measured

A1c test result	Nominal	Indicates the range of the result or if the test was not taken. Values: “>8” if the result was greater than 8%, “>7” if the result was greater than 7% but less than 8%, “normal” if the result was less than 7%, and “none” if not measured.
Change of medications	Nominal	Indicates if there was a change in diabetic medications (either dosage or generic name). Values: “change” and “no change”
Diabetes medications	Nominal	Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”
24 features for medications	Nominal	For the generic names: metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, sitagliptin, insulin, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, and metformin-pioglitazone, the feature indicates whether the drug was prescribed or there was a change in the dosage. Values: “up” if the dosage was increased during the encounter, “down” if the dosage was decreased, “steady” if the dosage did not change, and “no” if the drug was not prescribed
Readmitted	Nominal	Days to inpatient readmission. Values: “<30” if the patient was readmitted in less than 30 days, “>30” if the patient was readmitted in more than 30 days, and “No” for no record of readmission.

Methodology - Overview

Dataset Preprocessing

To conduct our data analysis, I needed to address several issues:

1. I needed to properly separate all numerical variables from nominal variables. In the original dataset, there were several variables that were categorical in nature but nevertheless had numerical values (i.e "admission_source" could take on one of 21 separate integer values, with each value representing a separate admission source). Using data as is would have resulted in the imposing of an artificial ordering of categorical variables in our dataset. To remedy this issue, I chose to use a one-hot encoding for all categorical variables.
2. There were numerous variables that were missing values for several data points. I did not want to throw away any data points that were missing values for only one or two attributes. For data points that were missing numerical values, I remedied this issue by assigning a value equal to the mean for that attribute for the other data points. For data points that were missing categorical variables, I simply assigned them to be "Other"
3. Certain numerical variables were given in bands (i.e weight was given in 10 lb increments). I did not want to treat these variables as categorical because there clearly exists an ordering. As a result, I chose to assign each data point the mean for such attributes that were naturally numerical but were still presented in banded form.

Model Accuracy Quantification

For all machine learning models tested, a cross-validation methodology was used. The original dataset consisted of about 101,000 individual data points. 70% of these data points were used for training the respective models, and 30% of the data points were used for testing.

Quantifying Accuracy on Prediction of Readmission

Readmission was treated as a binary categorical variable, where the attribute is true if there is any record of readmission within the years that this dataset encompasses. As such, model accuracy was defined to simply be the percentage of correctly classified data points in the test dataset.

Quantifying Accuracy on Prediction of Time in Hospital

Length of stay in the hospital is a continuous variable. As such, mean squared error was used as a measure of accuracy for predicting this variable.

Variables Included

Classification - Readmittance

All variables present in the dataset were used for this classification task. It is our belief that the primary application of such a methodology would be to predict whether patients are at high risk of readmittance upon discharge from a hospital. As a result, all of the data in the dataset will be available for analysis - for each patient who is being discharged. Therefore, it makes the most sense to include all available data to maximize accuracy

Regression - Time To completion

For this, two models were created:

1. All variables included:

   One of each model was trained on the full dataset. This was to provide an understanding of which models inherently performed better at descriptive analytics (predicting from past data), and to gain a better understanding of the relative strengths and weaknesses of each model.

2. Endogenous/Unavailable variables removed:

   For this model the following variables were removed:

   - Discharge disposition
   - Number of procedures
   - Number of lab procedures
   - Number of medications
   - Payer Code
   - Readmitted

To be removed, a variable had to fit one of two criteria. First, all variables believed to be endogenous variables were removed. For example, the number of procedures carried out likely directly impacts the time spent in the hospital, **but** the length of time spent in the hospital also directly impacts the number of procedures (simultaneity bias). These variables included: “Number of procedures”, “Number of lab procedures”, and “Number of medications”. Second, all variables that would not be available to medical professionals at the time of prediction were removed. If this model were to be used for predictive analytics (helping hospitals determine which patients will be spending the most time in their wards), it is of little use to include variables in a model that are not available at time of admittance. Thus, “Discharge disposition”, “Payer Code” (since insurance information may not be available until billing time), and “Readmitted” were also removed. I wanted to isolate the predictive power of only those variables that are exogenously determined and are available to professionals when prediction is needed.

Methodology - Model Analysis

K-Nearest Neighbors

The K-Nearest neighbor algorithm presented a set of unique challenges, the biggest of which was the extremely dimensionality space of the data. There were over 2000 separate individual attributes in our dataset, and over 100,000 individual data points. Because KNN requires the computation of a distance metric for every point in our test set against every point in the train set, this becomes enormously computationally expensive. To combat this issue, an approach similar to that proposed by Tang, Pan and Yao (“K-Nearest Neighbor Regression with Principal Component Analysis for Financial Time Series Prediction") was followed.

Principal Component analysis is a dimensionality reduction method that aims to capture as much of the variance possible in the original dataset into a smaller set of variables. First, all attribute values were standardized (this also serves to increase the efficacy of the KNN algorithm, so that attribute scaling does not play a role in distance calculations). A covariance matrix was then computed, and eigenvectors were computed from this matrix. The “n” eigenvectors corresponding to the “n” largest eigenvalues were chosen as the principal components (where “n” is a hyperparameter). The choice of “n” was varied and plotted along with out-of-sample accuracy. Finally, the data points were recast using these principal components by multiplying the transpose of the principal components by the transpose of the standardized original dataset.

For classification (i.e whether a patient was readmitted), each test point was compared with its “k” nearest neighbors. The majority classification of these neighbors was applied to the test point. For regression (i.e time spent in hospital), each test point was compared with the “k” nearest neighbors. The average value for this attribute was applied to the test point. “k” is a hyperparameter that was tuned for accuracy (graphs are presented later in this paper). Euclidean distance was used as the distance metric for determining the “k” nearest neighbors.

Neural Networks

I also applied a variety of neural networks to predict both readmission and time in hospital. Given the large number of features in the dataset, these neural networks had to have high dimensionality.

For classification, I trained the model using binary cross entropy as the loss function and used binary accuracy as the testing metric. On the last layer of the model, I condensed the output to a single number and applied a sigmoid activation function to keep that number within the range

[0,1], so that it would accurately reflect a probability. In the various models tested, I changed a number of variables such as the types of layers included (both dense and dropout layers), the number of layers included, the batch size and the number of epochs.

For regression, I used very similar models as for classification, but with a couple of important distinctions. First, instead of condensing the output to within the range [0,1], I allowed the output to be any positive number. Thus, it should better be able to approximate the result of time in hospital, which is measured in number of days.

In our neural networks, I use only dense layers and dropout layers. The dropout layers are used to abandon 20% of the connections by the previous dense layer so as to avoid overfitting. I decided not to use convolutional layers (which tend to be useful for image inputs) or recurrent layers (which tend to be useful for time related datasets), as I did not think there was a logical reason as to why these layers would apply to the problem at hand. I did not want to simply throw random layers at the problem and see what happened; I wanted to build a model that made sense logically.

All models were optimized using a variant of stochastic gradient descent (ADAM optimization). The training data is divided into batches. For each batch, the gradient of the loss function with respect to each network was computed. This gradient (multiplied by a learning rate) is then used to update each parameter in the network. The ADAM optimizer is a marked improvement over traditional stochastic gradient descent because of its adaptive learning rates. Individual adaptive learning rates are computed via estimates of first and second moments of the gradients (mean and variance). Research by Kingma and Ba (“Adam: A Method for Stochastic Optimization”) shows that this form of gradient update is particularly well-suited to very large datasets with large numbers of parameters, especially in comparison to more traditional methods such as AdaGrad and RMSProp, hence its usage.

Decision Tree and Random Forest

As an extension and application of the work done in Problem Set 3, I built a series of random forests, where a random forest is simply an ensemble of numerous decision trees.

A major challenge faced when using Decision Trees on this dataset was the enormously high dimensionality/number of features. With over 2500 separate attributes, algorithms such as ID3 will be near impossible to implement on such a large dataset. To combat this, only a subset of 100 features were considered at each “split point” of the tree. In addition, bootstrapping was utilized (where only 20,000 independently drawn sample points were used for the construction of each individual tree in the forest). For both the classification and regression problems, I vary the

depths of the random forest to find the depth that maximizes accuracy or minimizes mean squared error. The Gini coefficient was used as a measure of loss when splitting nodes on the tree.

The major advantage of random forests over decision trees is that they minimize the degree of overfitting because they average across a large number of trees that were trained on different subsets of the data. Thus, there is no ability for any single tree to memorize the data and dominate results when run on a new data set.

Support Vector Machines

(only used for classification in this paper)

The strength of the SVM lies in its ability to work with very high-dimensional datasets, such as the one being analyzed in this paper.

In simple terms, the SVM classifier works by finding hyperplanes to linearly separate the data. These hyperplanes are chosen via their "margin", or their distance from the closest correctly classified point on either side of the decision boundary. The hyperplane that maximizes this margin is chosen to be the decision boundary.

With low dimensional datasets, SVM classifiers can be supplemented via the usage of kernels. These kernels represent systematic ways to project the data set into a higher dimensional space. Since the dataset being used in this task is already very high dimensional, I have chosen to forego the more complex kernels (i.e radial basis kernel, high degree polynomial kernels) in favor of the linear kernel (which retains the dimensionality of our original dataset). This was done for two reasons. First, training a SVM classifier on an even higher dimensional dataset will be enormously computationally expensive. Considering that there are already over 2500 individual features present in the data, and over 100,000 data points, projection into higher dimensional data points would require a large amount of computational resources. Second, further complexifying the dimensionality runs the risk of overfitting the model. A paper by Han and Jian ("Overcome Support Vector Machine Diagnosis Overfitting") studying the usage of SVM in disease diagnosis (which is similar to the prediction task I are carrying out) with high dimensional datasets finds that linear kernels tend to work best on this specific type of data, and that SVM's with nonlinear kernels (specifically the Gaussian kernel) tended to show poor out-of-sample performance.

To account for the likely non linear-separability of our dataset without feature transformation, I will be using a soft-margin SVM. This SVM will penalize higher margins (where margins are measured from the support vector) as well as misclassifications. The regularization parameter

“C” will also be tuned, which handles this tradeoff. Lower values of “C” correspond to greater regularization - resulting in greater misclassification rates in exchange for higher margins.

Finally, all feature attributes were standardized to have 0 mean and unit variance prior to running the SVM algorithm. This standardization was used to keep irrelevant factors such as attribute scales from affecting algorithm accuracy.

Results

K-Nearest Neighbors

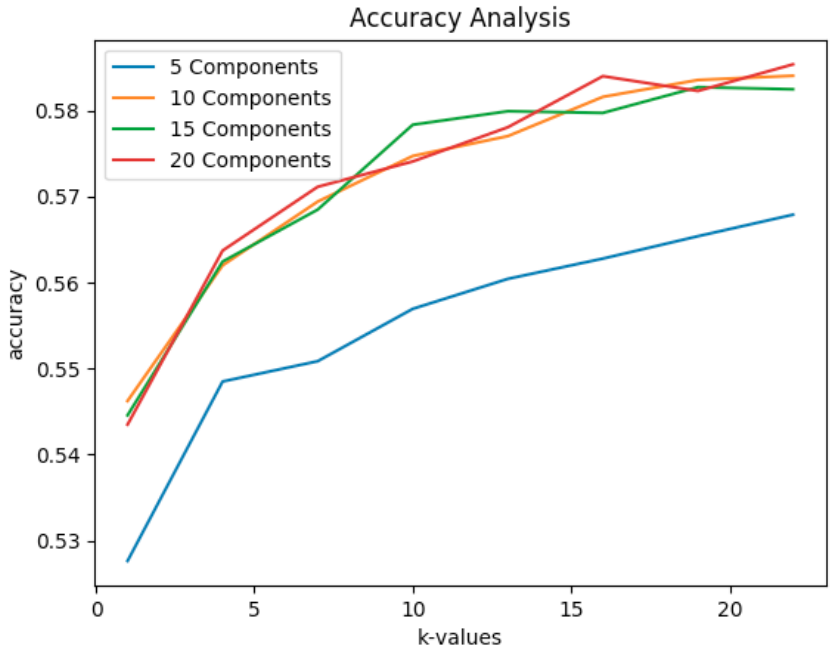
The results for KNN on the out-of-sample data with varying degrees of dimensionality reduction and values of K are presented below. Overall, KNN was the least accurate out of all models presented. This was expected. Research by Syaliman, Nababan, and Sitompul (“Improving the accuracy of k-nearest neighbor using local mean based and distance weight”) indicates that there are a variety of factors that cause this fact:

First, KNN classifies simply according to a majority vote system. All of the “k” neighbors are given equal weighting when determining the classification of the test point, regardless of whether some of those “k” points are much closer to the test point than the rest of the “k” points. The authors suggest using a distance-based weighting metric to combat this effect (this was not used in our implementation, but would likely lead to greater accuracy).

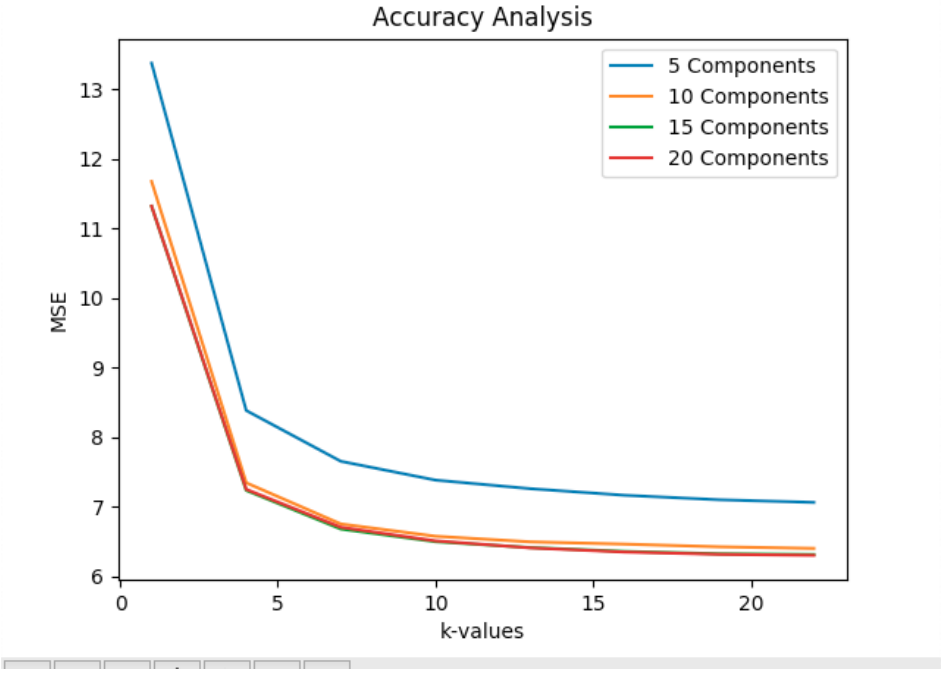
Second, KNN is especially vulnerable to the “curse of dimensionality”, even when variables are standardized prior to application of the algorithm. Joel Grus in his book (“Data Science from Scratch”) calculates the ratio between the closest distance and the average distance from any point to another point for 10,000 separate data points - (Distance_Closest/Distance_Average). As dimensionality increases from 0 to 100, this ratio approaches ~0.8 at 100 dimensions. This clearly shows that the effectiveness of the KNN algorithm decreases as the dimensionality of the dataset increases. As stated before, I combated this via PCA dimensionality reduction. This compression inevitably leads to some data loss when a lower number of components are used. This data loss does appear to asymptote and and tend to 0 as the number of components increases (as seen in the graphs below)

With regards to predictive power, KNN did show a marked decline in accuracy (measured via MSE) when variables that are not available prior to hospital discharge are removed. This does make sense, it’s clear that the number of medications administered etc.. will be highly correlated with our variable of interest (time spent in hospital), and removal of such variables will decrease the predictive power of our model.

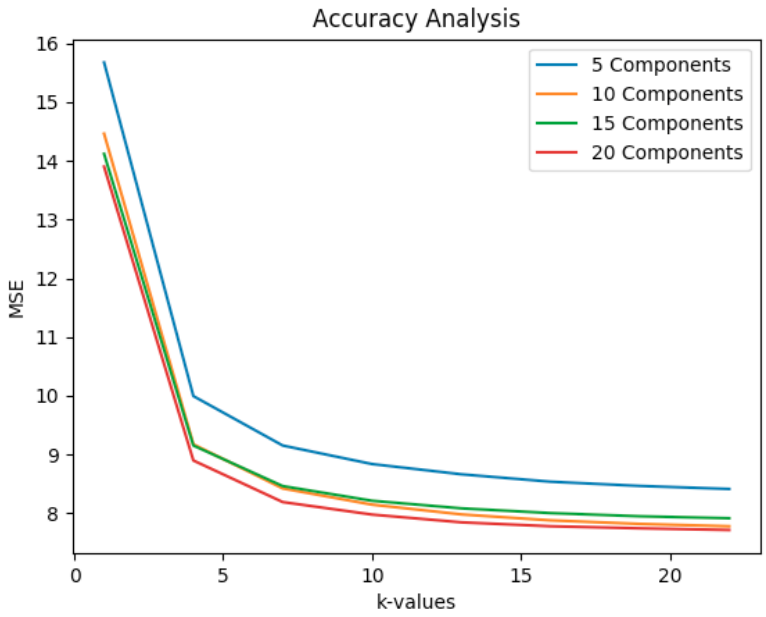
Binary Classification of Readmittance
Graph



Regression - Prediction of Time in Hospital
Model 1: All Variables Included
Graph



Model 2: All endogenous and unavailable variables removed
Graph



Neural Networks

In this project, I deployed 3 different artificial neural networks to each of the 3 problems I seek to answer in this paper for a total of 9 models. These models are labeled 1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b, 3c. Where the number denotes the problem I seek to answer, and the letter denotes the general structure of the model. This will make more sense upon reading through the models. The network structures are described below.

Binary Classification of Readmittance:

Model #1: Model #1a is a dense neural network with the following layers:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 3000)	7509000
dense_1 (Dense)	(None, 1500)	4501500

dense_2 (Dense)	(None, 800)	1200800
dense_3 (Dense)	(None, 400)	320400
dense_4 (Dense)	(None, 1)	401

Total params: 13,532,101
Trainable params: 13,532,101
Non-trainable params: 0

This model was optimized using ‘adam’ and trained using ‘binary_crossentropy’ as the loss function.

On readmittance prediction, this model achieved 63.2% binary accuracy.

Model #2a: In this model, dropout layers (which dropped 20% of connections made in the previous layer) were added. Second, the number of epochs was increased to 4. Third, the batch size was increased to 64. On readmittance prediction, the binary accuracy for this model was 63.3%. The model structure is outlined below:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 3000)	7509000
dropout (Dropout)	(None, 3000)	0
dense_1 (Dense)	(None, 1500)	4501500
dropout_1 (Dropout)	(None, 1500)	0
dense_2 (Dense)	(None, 800)	1200800
dropout_2 (Dropout)	(None, 800)	0
dense_3 (Dense)	(None, 400)	320400
dense_4 (Dense)	(None, 1)	401

Total params: 13,532,101

Trainable params: 13,532,101
Non-trainable params: 0

Model #3a: In this model, two additional changes were made. First, batch size was increased to 264. Second, the number of epochs was increased to 20. The layers remain the same as in Model #2. On readmittance prediction, the binary accuracy of Model #3 was 60.6%, likely due to overfitting caused by too many epochs, which is studied further in another section of this paper. The model structure is shown below:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 3000)	7509000
dropout (Dropout)	(None, 3000)	0
dense_1 (Dense)	(None, 1500)	4501500
dropout_1 (Dropout)	(None, 1500)	0
dense_2 (Dense)	(None, 800)	1200800
dropout_2 (Dropout)	(None, 800)	0
dense_3 (Dense)	(None, 400)	320400
dense_4 (Dense)	(None, 1)	401

Total params: 13,532,101
Trainable params: 13,532,101
Non-trainable params: 0

Predict Time in Hospital -- All Variables

Model #1b: With variations to the loss function (now using mean squared error) and the accuracy function (now using mean squared error) I used the same model as Model #1 to predict time in hospital. This model was able to predict time in hospital with a mean squared error of 5.43.

Model #2b: With the same variations as done to Model #1, I used Model #2 to predict time in hospital. This model was able to predict time in hospital with a mean squared error of 5.17.

Model #3b: Given the overfitting previously shown in Model #3a used to predict readmittance, in this case Model #3b is the same as Model #2b but with 8 epochs. This model was able to predict time in hospital with a mean squared error of 5.07. This was more accurate than both of the prior models on the full dataset.

Predict Time in Hospital -- Variables available upon patient admittance

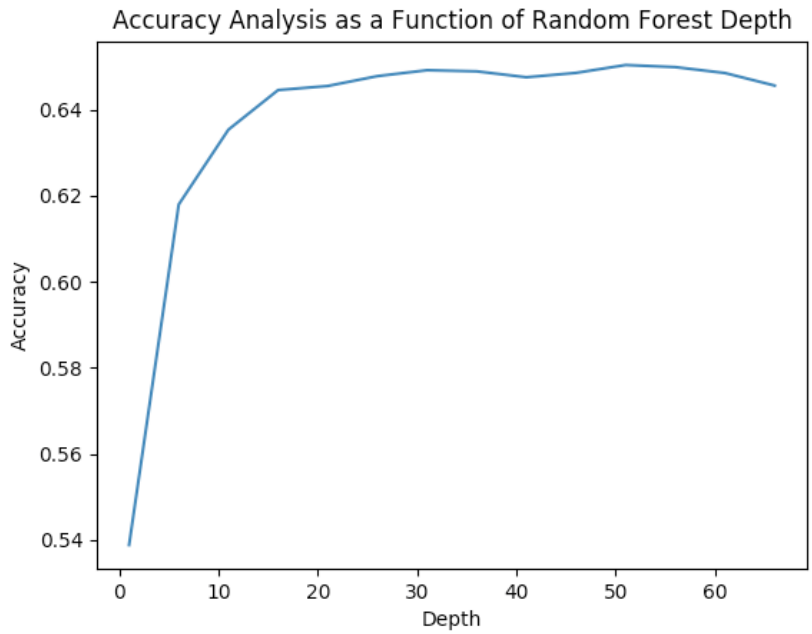
Model #1c: Model #1c is the same as Model #1b, but it was run on a more limited dataset that only included the variables known at the time of hospital admission. This model achieves a mean squared error of 8.26.

Model #2c: Model #2c is the same as Model #2b, but it was run on a more limited dataset that only included the variables known at the time of hospital admission. This model achieves a mean squared error of 7.04. Of the three models tested, this model experienced the smallest increase in MSE on the limited dataset with an increase of 1.87

Model #3c: Model #3c is the same as Model #3b, but it was run on a more limited dataset that only included the variables known at the time of hospital admission. This model achieves a mean squared error of 7.97. Of the three neural network models tested, this model experienced the largest increase in MSE on the limited dataset with an increase of 2.9.

Decision Tree and Random Forest

Binary Classification of Readmittance:
In using the random forest model for classification, I set the number of trees to 100 and the number of attributes considered for each tree set to 100. Samples are bootstrapped using a sample size of 20,000. In the following graph, I investigate the out of sample test set accuracy as a function of tree depth.



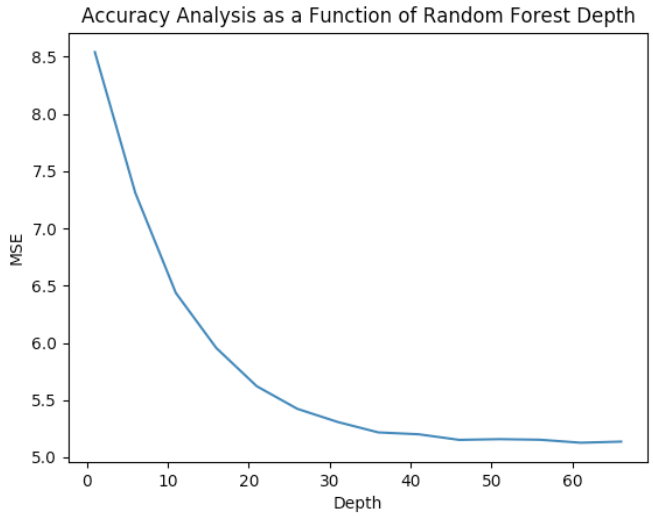
As one can see from the above graph, accuracy plateaus after the tree depth exceeds 20. Thus, a practical implementation of the information could be creating a physician's checklist that a physician would have to complete before allowing a patient to leave the hospital. The above graph shows us that there would be little use in making a checklist longer than 20 questions. It is possible that even that checklist might be too long if physicians don't have the time to complete it, but this graph gives us a good upper bound on the length of the checklist.

One interesting note is that the random forest appears to be particularly resistant to overfitting. Even at very large depths, out of sample test set accuracy did not experience any significant declines. This is likely because of the fact that the number of attributes considered was severely limited (only permutations of 100 possible features out of ~2500 possible features were used for each split point of the tree - which is less than 4% - and bootstrapping was conducted on top of these feature restrictions).

Regression to Predict Time in Hospital -- All Variables

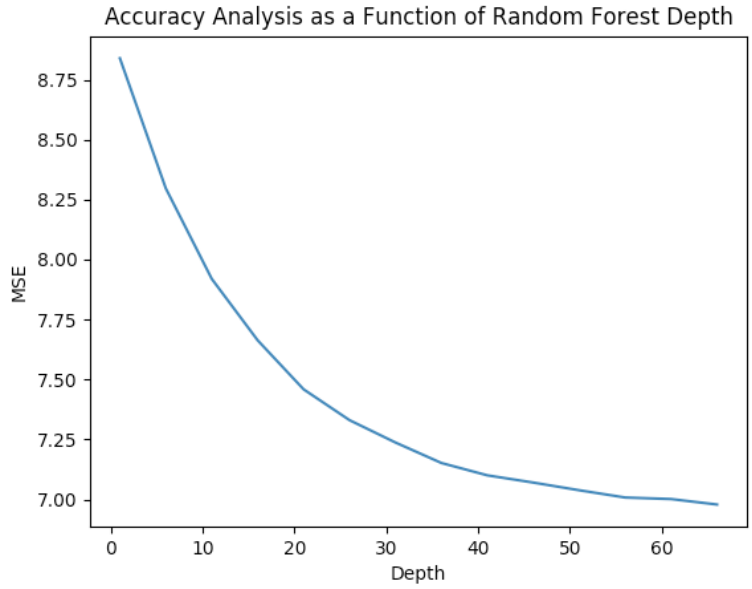
Model Set 1: All variables included

In using the random forest for the regression problem, I set the number of trees to 100 and the number of attributes considered for each tree set to 100. Samples are bootstrapped using a sample size of 20,000. In the following graph, I investigate accuracy as a function of tree depth.



Model 2: All unavailable variables removed

Now, I apply the random forest model to predict time spent in hospital using only the data available when a patient arrives at the hospital. Once again, I set the number of trees to 100. Number of attributes considered for each tree set to 100. Samples are bootstrapped using a sample size of 20,000. In the following graph, I investigating accuracy as a function of tree depth



In the above two graphs, I see that Mean Squared Error is a monotonically decreasing function with respect to Random Forest Depth. However, the graphs start to plateau around a depth of 40. Interestingly, the random forest run on the more limited set of data takes longer to plateau, which indicates that there are less obvious relationships between features.

Support Vector Machines

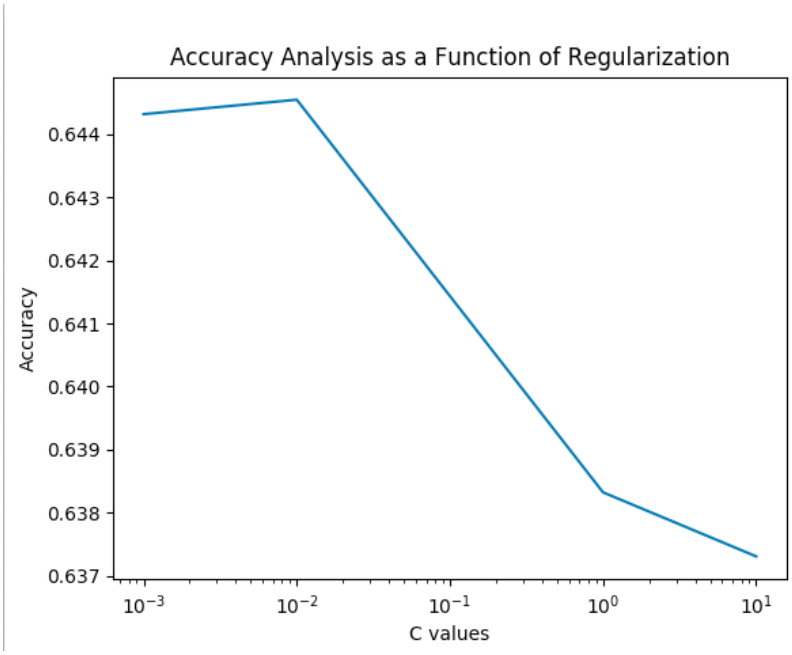
The SVM with linear kernel was able to achieve higher accuracy than the KNN classification system, and there were several clear conclusions from the SVM analysis.

First, the data is decidedly nonlinear. When using the simple linear kernel (given our limited computational resources and the massive size of the dataset it was difficult to fit more complex kernels such as the RBF kernel) accuracy topped out at about 64.5% on this classification problem using the most tuned and optimized hyperparameters. This is a clear indication that the data is not linearly separable, since applying the SVM without feature transformation was unable to garner strong results.

Second, the classifier is showing higher accuracy levels at very low levels of "C". This C parameter is inversely proportional to the level of regularization, indicating that out-of-sample accuracy was highest when regularization was relatively strong. This in turn indicates that the dataset is prone to overfitting. Accuracy in the validation set immediately drops at C values higher than .01 and shows a fairly consistent decline afterwards. This immediately indicates that the dataset may be prone to outliers. If there are extreme outliers in the training dataset, the model will require higher levels of regularization to compensate and prevent overfitting in the training dataset.

Given that the data is nonlinear, why then is SVM outperforming the KNN algorithm (which is a decidedly nonlinear classification algorithm, although admittedly SVM was outperformed by the Random Forest and Neural Network classification systems - likely precisely due to their ability to capture nonlinearity)? I believe that a strong factor may be the dimensionality reduction that was required for the KNN algorithm. Because of the computational complexity of the KNN algorithm, I were forced to compress over 2500 features in a much smaller feature space. This will inevitably lead to a fair amount of data loss. Since SVM with the linear kernel did not require this kind of dimensionality reduction for efficient training, I were able to retain some additional characteristics about the dataset that improved classification accuracy.

Classification of Readmittance
Graph



Comparison of Results

Model Type	Readmittance (accuracy)	Time in Hospital -- All (MSE)	Time in Hospital -- Limited (MSE)
Unaided Guess*	.530	8.91	8.91
KNN	.586	6.15	7.57
Neural Network	.633	**5.07**	7.04
Random Forest	.643	5.13	**6.94**
SVM	**.644**	--	--

**Unaided Guess: for readmittance guess the most common label; for time in hospital guess average time in hospital*

Given the above table, it is clear that our models add a significant amount of value over the uniformed case regardless of the model used. Of the models used, the SVM worked the best for

classifying readmittance and the random forest performed the best in predicting time in hospital based on the limited feature set -- the two tests I believe are most useful and applicable to real world settings. The neural networks also performed well, while KNN beat the unaided guess but was not in the same league as either the random forest or the neural networks.

Conclusion

Given that our models show significant improvement over the uniformed case, it is clear that hospitals should use machine learning models to better predict patient outcomes and allocate hospital resources. As possible suggestions, our readmission results, especially the random forest, could be used as a checklist (which would likely need to be automated because each tree in the forest will likely be splitting on different features) for physicians to decide whether or not a patient should be released from the hospital. Additionally, our results on 'time in hospital' should be used to allocate hospital beds and aid in planning. That is, if 10 diabetic patients are admitted to the hospital on a given day, the hospital could use our model to predict how long each of those patients will be there. Thus, they should be able to foresee bed shortages before they actually occur and make plans accordingly.

Despite the wide variety of models used in this paper, I saw a relative convergence of results between models. Nevertheless, certain models did exhibit better accuracy than others models. The random forest was particularly strong and equally balanced at predicting readmission rates and predicting the length of stay in the hospital (especially with the more limited dataset). Classification by the Random Forest yielded a 10% improvement in accuracy over the unaided model, a reduction in MSE in the full dataset by 3.78, and a reduction in the MSE in the restricted dataset by 1.97. I believe that the strong relative performance of the Random Forest is driven by the inherent high dimensionality of our dataset, as well as the large proportion of categorical variables. Research by Gerard Biau indicates that Random Forests tend to be particularly adept at handling datasets with high levels of sparsity (large numbers of noise attributes), which is a characteristic of this dataset considering the over 2500 separate attribute possibilities.

All of the models tested exhibited significant increases in MSE when unavailable variables were removed. The neural network experienced the largest increase of 1.97, while the KNN experienced the smallest increase of 1.42. This makes intuitive sense; the full dataset includes a number of variables that are likely to both be endogenous and highly correlated with the variables of interest (i.e number of procedures is innately correlated and determined by the length of stay in the hospital). Even with dropout layers, the neural network is likely to immediately pick up on these correlations with unavailable variables in the full dataset. The gradients will be large, and the resultant weights on these attributes will also be large. In contrast,

the random forest selects a mere 4% of attributes at each "split" in the tree for each decision tree, and the KNN weights each attribute equally when determining the regression label for data points. This serves to reduce the dependence of both of the latter models on these unavailable variables. Since the neural network picks up on these strong correlations, the neural network is naturally also the most accurate predictor on the full dataset. However, this also leads to a more severe effect when these variables are removed because the network is no longer able to utilize these correlations. In contrast, the KNN and Random Forest were less reliant on these correlations in the first place, so when these attributes are removed they exhibit a comparatively smaller decline in accuracy (increase in MSE).

Although our models did show significant increases in accuracy, I do believe that there is potential for much greater predictive ability given a larger dataset and more targeted features. I believe that there is a large amount of noise in our dataset, and that many attributes are not significantly correlated with our variables of interest. For those variables that are strongly predictive, the dataset is too small to definitively create models with high accuracy. For example, I only have the results from two lab tests (glucose serum and A1c test results), and the variable that I do expect to be highly correlated with patient outcomes (i.e medical diagnoses) has ~2000 individual possible values. Given a training dataset of ~70,000 data points (of which a large portion are missing values for the salient attributes) it can be difficult to derive strong predictive relationships. To combat this, it would be very beneficial to collect data with a heightened emphasis on lab tests and exhibited symptoms. I believe these attributes are much more relevant and much more indicative of the variables I are trying to predict.

With regards to other possible lines of study, I look forward to linking the current data set to other data sets that include information on geography and income. This would allow researchers to gain a deeper understanding of the out of hospital factors that influence patient outcomes. An additional line of possible followup would be conducting analysis to determine whether race affects how a diabetic patient is treated in the hospital. This would be a very interesting study; however, it would be quite difficult to deal with the extreme endogeneity present in medical data. Nonetheless, it would be an interesting study to conduct.

In sum, I hope to have proven the value of machine learning in its application to healthcare, show how results of different machine learning methods compare to each other, and link our results to specific, actionable steps to be taken by the healthcare community. Diabetes remains a pressing issue in modern societies, and our results provide a path forward to alleviate its burden.

References

Biau, Gerard. “Analysis of a Random Forests Model.” *Journal of Machine Learning Research*, 2012.

Grus, Joel. *Data Science from Scratch:* Sebastopol, CA: O'Reilly Media, 2015.

Han, Henry, and Xiaoqian Jiang. “Overcome Support Vector Machine Diagnosis Overfitting.” *Cancer Informatics* 13s1 (2014). https://doi.org/10.4137/cin.s13875.

Kingma, Diederik, and Jimmy Ba. “Adam: A Method For Stochastic Optimization.” *International Conference on Learning Representations*, 2015.

Syaliman, KU, EB Nababan, and OS Sitampul. “Improving the Accuracy of k-Nearest Neighbor Using Local Mean Based and Distance Weight.” *IOP Conf. Series: Journal of Physics*, 2017. https://doi.org/10.1088/1742-6596/978/1/012047.

Tang, Li, and Heping Pan. “K-Nearest Neighbor Regression with Principal Component Analysis for Financial Time Series Prediction.” *International Conference on Computing and Artificial Intelligence* , 2018, doi:0.1145/3194452.3194467.