1 Introduction

Hospital readmission is an emerging indicator of quality of care. Diabetes is associated with increased risk of hospital readmission. A study conducted by Robbins & Webb (2006) showed that rehospitalizations within 30 days of discharge occurred in 20% of patients with diabetes. This figure is more than the 5-14% estimations for all hospital discharges. Also, a study conducted by Jiang, Stryer, Friedman, & Andrews (2003) showed that 30% of the diabetes patients who had been hospitalized were again hospitalized more than once within 1 year. Moreover, these diabetes patients accounted for a majority of the inpatients costs for patients with diabetes. The total number of hospital stays for patients with diabetes was more than 7.7 million in 2008, accounting for 20% of the hospitalizations and $83 billion in the U.S (Fraze, Jiang, & Burgess, 2010).

There are many algorithms for classification and prediction of readmission of diabetes patients within 30 days. This report gives a comparison between the performances of four classifiers: *SVM, Logistic Regression, C4.5 and Random Forest* which are among the most influential data mining algorithms in the research community and among the top 10 data mining algorithms. Our aim is to evaluate efficiency and effectiveness of those algorithms in terms of accuracy, sensitivity, specificity and precision.

2 Problem Definition and Algorithm

2.1 **Task Definition**

Classification is one of the most important and essential task in machine learning and data mining. A lot of research has been done on data mining and machine learning on medical datasets to classify re-admittance of patients in the hospital. Many of them show good classification accuracy.

We compare the performance criterion for supervised learning classifiers: such as SVM, Naive Bayes , C4.5 and Random Forest.The performance and accuracy of these algorithms are a good indicators for future studies and allows for classification of algorithm for different kinds of datasets. An important challenge in data mining and machine learning process is to build robust and accurately computational efficient classifiers for medical applications.

2.2 Algorithm Definition

Logistic Regression

With the starting assumption that the impact of factors and their interactions can be modelled as a log likelihood of outcome, logistic regression can help us understand the relative impact and statistical significance of each factor on the probability of readmission. The missing values were handled by either removing the columns or following the preprocessing steps.

Randomised Logistic Regression was done to get the most important features for our logistic regression model but got poor performance metrics than the feature sets considered above and hence, we decided to stick with the feature sets above to maintain consistency.
Since there are too many variables and coefficients to look at, we are picking only those coefficients that have p-value < 0.01 (i.e. statistically significant) and have at least 0.2 magnitude.

For example:
The strongest predictors of readmission within 30 days appear to be four types of discharge conditions in both versions. Intuitively, these make sense — transfer to another unit in a hospital or another hospital may indicate higher severity/complexity of disease and make readmission likely.

In logistic regression, the cost function is basically a measure of how often you predicted 1 when the true answer was 0, or vice versa. Below is a regularized cost function which helps prevent overfitting.

Random Forest

A random forest is a meta-estimator that aggregates many decision trees .By considering more than one decision tree and then doing a majority voting, random forests helped in being more robust predictive representations than trees. For both Decision Trees and Random Forests, we removed the interaction terms from the feature set since these are already accounted for in tree-based models.

$$Cost(\beta) = \frac{\sum_{i=1}^{n}(y^i log(h_\beta(x^i)) + (1-y^i)log(1-h_\beta(x^i))^2}{2*n} + \lambda \sum_{j=1}^{k} \beta_j^2$$

The number of features that can be split on at each node is limited to hyper parameter. This ensures that the ensemble model does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features.

Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents overfitting. These modifications also prevent the trees from being too highly correlated. Without these, every tree would be identical, since recursive binary splitting is deterministic.

Our decision tree model indicates highest importance of time spent in the hospital, age and discharge to another hospital for both simple and complex versions. If we plot these against the coefficients from logistic regression, they do not correlate well. Since we used cross-validation and achieved similar test and train accuracy to avoid overfitting, this may suggest a different correlation structure of the variables in the model or lower explained variance in logistic model.

For random forest ensemble, the same features had high importance, although the distribution was more even as compared to decision tree. This is likely due to stabilization of importance's across many trees.

Support Vector Machine(SVM)

Support Vector Machines helps model linearly inseparable data, thus allowing us to explain complex non-linear relationships. To apply our classifiers and evaluate them, we apply the 25-fold cross validation test which is a technique used in evaluating predictive models that split the original set into a training sample to train the model, and a test set to evaluate it. SVM is robust of outliers in the data set provided.

The SVM-RBF kernel is used. Since the value of the RBF kernel decreases with distance and ranges between zero (in the limit) and one (when x = x'), it has a ready interpretation as a similarity measure. The feature space of the kernel has an infinite number of dimensions; for , its expansion is:

$$\begin{aligned}\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) &= \sum_{j-0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{x}')^j}{j!}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)\\ &= \sum_{j=0}^{\infty}\sum_{\sum n_i=j}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\frac{x_1^{n_1}\cdots x_k^{n_k}}{\sqrt{n_1!\cdots n_k!}}\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)\frac{{x'}_1^{n_1}\cdots {x'}_k^{n_k}}{\sqrt{n_1!\cdots n_k!}}\end{aligned}$$

T h e
Regularization parameter tells the SVM optimization how much misclassifying was avoided after each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

However, because of high-dimensional structure and complexity, they are limited by their interpretability to gain insights on how different features are weighted/assigned importance.

3 Experimental Evaluation

3.1 Methodology

Dataset

We used a publicly available dataset from the UCI Machine Learning Repository. The dataset contains diabetes patient encounter data for 130 US hospitals collected between 1999 and 2008 and it contains 101766 observations. The dataset includes over 50 features representing patient and hospital outcomes. The data contains attributes such as patient number, race, gender, age, admission type, time in hospital etc.

We loaded the dataset as a pandas dataframe:

```
import IPython
import pandas as pd
import numpy as np
from statistics import mode
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

The features are depicted in Table 1.

Table 1 Features

Initial features Data columns (total 50 columns)	**After cleaning dataset** Data columns (total 38 columns)
encounter_id patient_nbr race gender age weight admission_type_id discharge_disposition_id admission_source_id time_in_hospital payer_code medical_specialty num_lab_procedures num_procedures num_medications number_outpatient number_emergency number_inpatient diag_1 diag_2 diag_3 number_diagnoses max_glu_serum HBA1Cresult metformin repaglinide nateglinide chlorpropamide glimepiride acetohexamide glipizide glyburide tolbutamide pioglitazone rosiglitazone acarbose miglitol troglitazone tolazamide examide citoglipton insulin glyburide-metformin glipizide-metformin glimepiride-pioglitazone metformin-rosiglitazone metformin-pioglitazone change diabetesMed readmitted dtypes: int64(13), object(37)	encounter_id patient_nbr race gender age admission_type_id discharge_disposition_id admission_source_id time_in_hospital num_lab_procedures num_procedures num_medications number_outpatient number_emergency number_inpatient number_diagnoses max_glu_serum metformin repaglinide glimepiride glipizide glyburide pioglitazone rosiglitazone insulin change diabetesMed readmitted numchange num_medications|time_in_hospital num_medications|num_procedures time_in_hospital|num_lab_procedures num_medications|num_lab_procedures num_medications|number_diagnoses age|number_diagnoses change|num_medications number_diagnoses|time_in_hospital num_medications|numchange dtypes: int64(38)

Data Cleaning

We analyzed the data to find out how many missing variables we have using the following code:

```
for col in df.columns:
    if df[col].dtype==object:
        print(col,df[col][df[col]=='?'].count())
print('gender', df['gender'][df['gender']=='Unknown/Invalid'].count())
```

Table 2. Missing values analysis

Feature	# Missing Values
Race	2273
Weight	98569
Payer code	40256
Medical speciality	49949
Diag_1	21
Diag_2	358
Diag_3	1423

The amount of missing values is depicted in Table 2. Weight was missing in over 98% of the records and was thus dropped. Payer code and Medical specialty also had a lot of missing values and were thus dropped. Primary, secondary and tertiary diagnoses also had missing values. We dropped records where all three diagnoses were missing. We also dropped the variable examide because it had exactly the same values as the variable citoglipton. We also dropped patients who died during hospital admission (discharge_disposition = 11), because these patients cannot be readmitted.

We also dropped columns that contained less than 1000 non-zero values.
Such were:

```
['nateglinide', 'chlorpropamide', 'acetohexamide', 'tolbutamide',
'acarbose', 'miglitol', 'troglitazone', 'tolazamide', 'glyburide-
metformin', 'glipizide-metformin', 'glimepiride-pioglitazone',
'metformin-rosiglitazone', 'metformin-pioglitazone']
```

Collapsing variables

The dataset contains 23 variables for 23 drugs which indicate whether a change in that medication was made or not during the hospital stay of the patient. To simplify our model, we counted the total amount of changes that were made. We also counted the total number of medications used per patient. The dataset also had patients who had more than one encounter. These cases can lead to bias.

```
keys = ['metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
'glimepiride', 'glipizide', 'glyburide', 'pioglitazone',
'rosiglitazone', 'acarbose', 'miglitol', 'insulin', 'glyburide-
metformin', 'tolazamide', 'metformin-pioglitazone','metformin-
rosiglitazone', 'glimepiride-pioglitazone', 'glipizide-metformin',
'troglitazone', 'tolbutamide', 'acetohexamide']
for col in keys:
    colname = str(col) + 'temp'
    df[colname] = df[col].apply(lambda x: 0 if (x == 'No' or x ==
'Steady') else 1)
df['numchange'] = 0
for col in keys:
    colname = str(col) + 'temp'
    df['numchange'] = df['numchange'] + df[colname]
    del df[colname]
```

Recoding variables

The outcome we were looking at was whether the patient gets readmitted to the hospital within 30 days or not. The variable in the dataset has < 30, > 30 and No Readmission categories. We reduced this to a binary classification: we combined the readmission after 30 days and no readmission into a single category:

```
df['readmitted'] = df['readmitted'].replace('>30', 0)
df['readmitted'] = df['readmitted'].replace('<30', 1)
df['readmitted'] = df['readmitted'].replace('NO', 0)
```

We also recoded the age groups to midpoints so we could study the effect of an increasing age if necessary. Gender categories such as ‘Male’ and ‘Female’ were binarized into 0 for Female and 1 for Male. We did the same for race: we converted nominal values into numeric values:

```
df['race']=df['race'].replace('Caucasian',0)
df['race']=df['race'].replace('Asian',1)
df['race']=df['race'].replace('AfricanAmerican',2)
df['race']=df['race'].replace('Hispanic',3)
df['race']=df['race'].replace('Other',4)
```

Interaction terms

Another thing we did is we combined some of the fields that had a high correlation with each other and which we hypothesized to be influencing each other.

```
interaction_terms = [
    ('num_medications','time_in_hospital'),
    ('num_medications','num_procedures'),
    ('time_in_hospital','num_lab_procedures'),
    ('num_medications','num_lab_procedures'),
    ('num_medications','number_diagnoses'),
    ('age','number_diagnoses'),
    ('change','num_medications'),
```

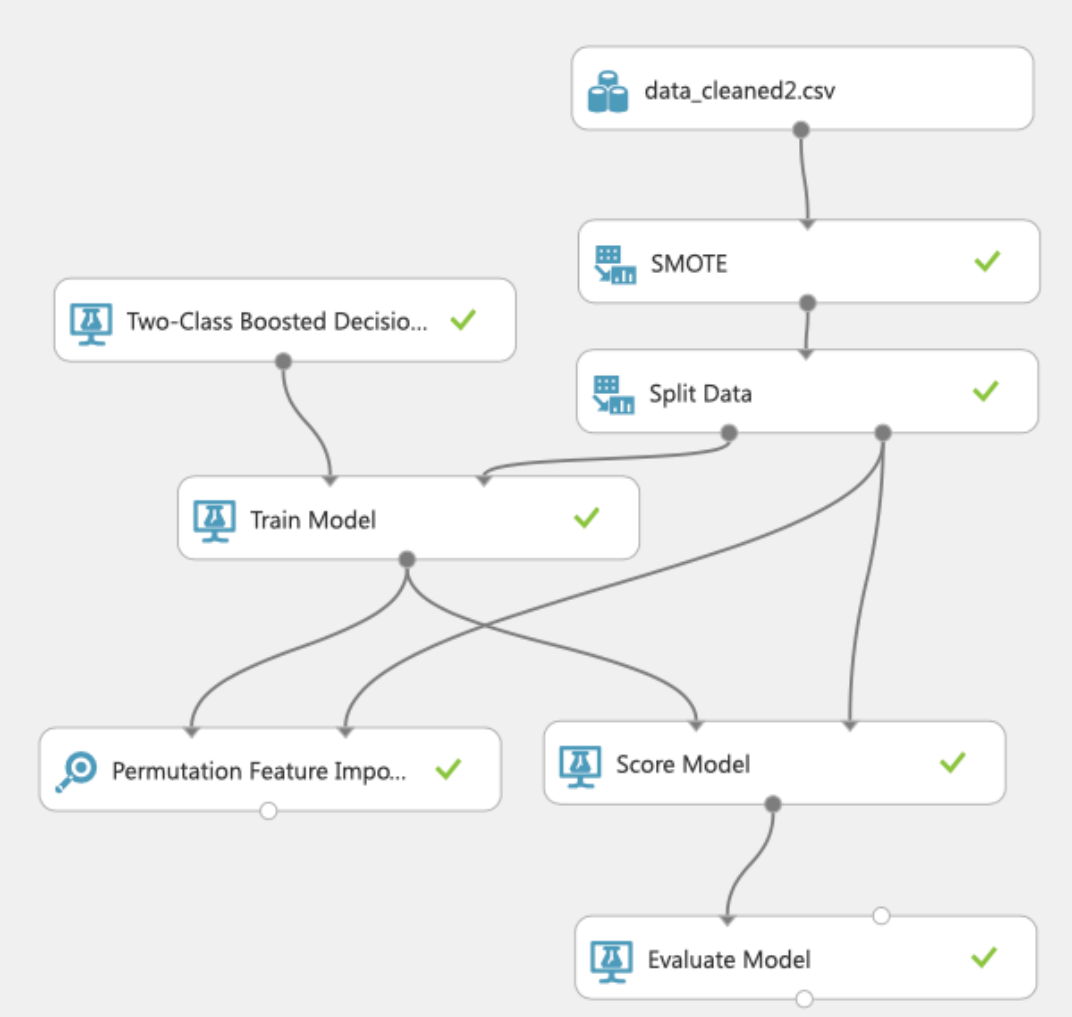
```
    ('number_diagnoses','time_in_hospital'),
    ('num_medications','numchange')
]

for interaction in interaction_terms:
    name = interaction[0] + '|' + interaction[1]
    df[name] = df[interaction[0]] * df[interaction[1]]
```

Data balancing

Our data was highly imbalanced with respect to readmissions. The “<30 readmissions” accounted only for 10% of the records. We used synthetic minority over-sampling technique (SMOTE) to oversample our “<30 days readmissions” class to obtain equal representation of our “<30 days readmissions” and “no readmissions” classes. This step was performed within the model training phase described below.

Model training and evaluation

After cleaning the data. We had many options for environments to train and compare the machine learning algorithms. During a hackathon organized by JADS, one of our group members experienced working with the machine learning studio of Microsoft Azure and suggested we use it. We decided to train and train and compare all models using it so that we can all experience the development process within a cloud platform. Figure 1 shows the visualization of the flow of actions.



First we loaded the already cleaned dataset. Since our data was highly imbalanced with respect to readmissions, we decided to use synthetic minority over-sampling technique (SMOTE) with a value of 100%. In the resulting dataset, we split our data into 70% train and 30% test data using the random selection principle. After that we used the train data to train each of the selected machine learning algorithms. After the creation of each model, we used its' score to get an evaluation of its' performance.

3.2 Results

Present the quantitative results of your experiments. Graphical data presentation such as graphs and histograms are frequently better than tables. What are the basic differences revealed in the data. Are they statistically significant?

To compare overall **model performance** metrics for different models, we collected all the metrics and created a chart as shown below. Highest precision (0.993) was observed in Decision Jungle. Highest recall (0.525) was observed in Boosted Decision Tree. The highest F-Measure (0.664) was also observed in Boosted Decision Tree.

Table 3: Comparison of accuracy measures for NB, DT, DF, DJ, Log Regression and SVM

	TP	FP	Precision	Recall	F-Measure
Bayes (NB)	1084	383	0.739	0.163	0.266
Boosted Decision Tree	3499	378	0.903	0.525	0.664
Decision Forest	2721	282	0.906	0.408	0.563
Decision Jungle	2040	15	0.993	0.306	0.468
Logistic Regression	1385	390	0.780	0.208	0.328
SVM	1195	384	0.757	0.179	0.290

TP = True positives, FP = false positives

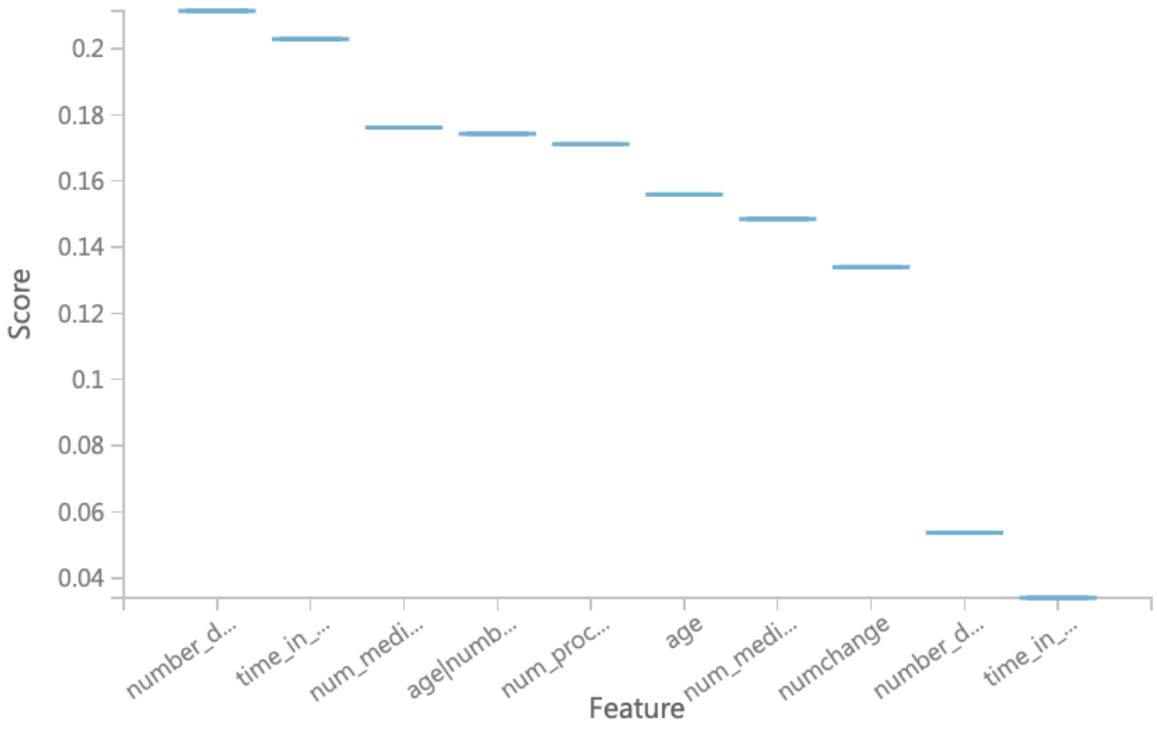


Figure 3 shows the 10 most import features for accuracy for the Boosed Decision Tree. The interaction term number_diagnoses * time_in_hospital has the highest value of 0.22. These numbers are calculated by comparing the accuracy of the model with and without a certain feature.

Figure 4 shows the 10 most important features for precision for the Boosted Decision Tree. The interaction term number_diagnoses * time_in_hospital has again the highest value of 0.57.

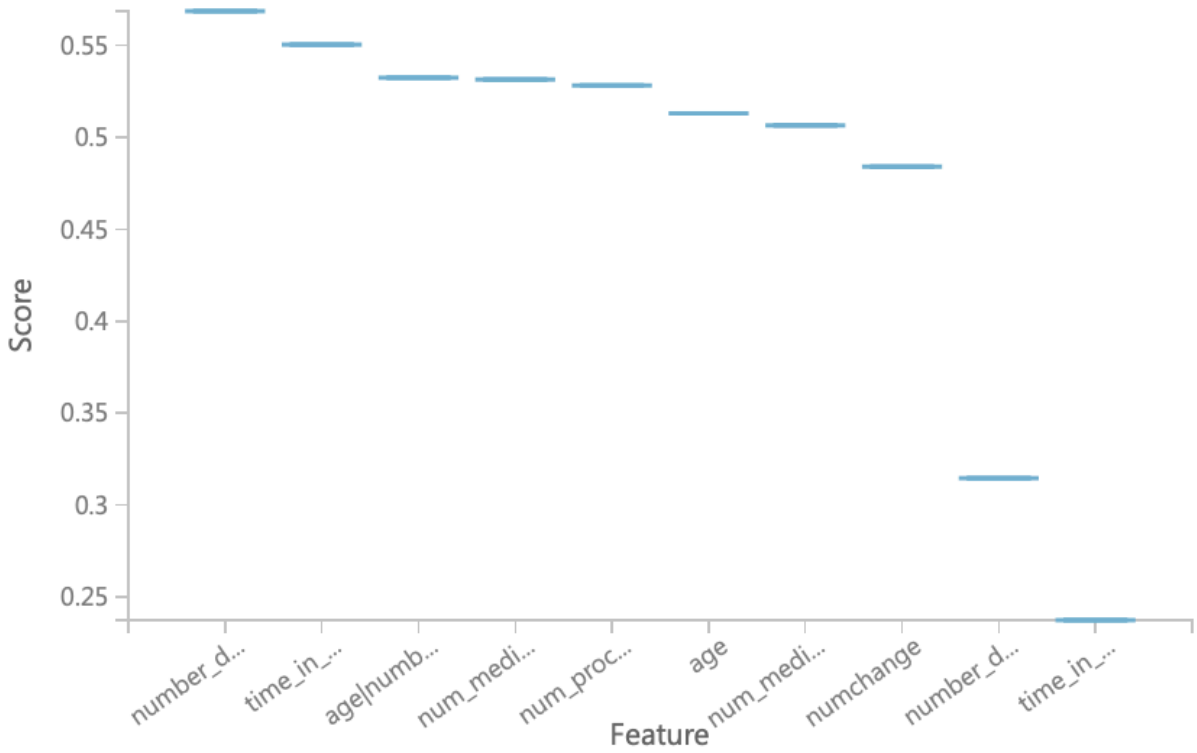


Figure 5 shows the 10 most important features for recall for the Boosted Decision Tree. The feature discharge_disposition_id has the highest value of 0.019.

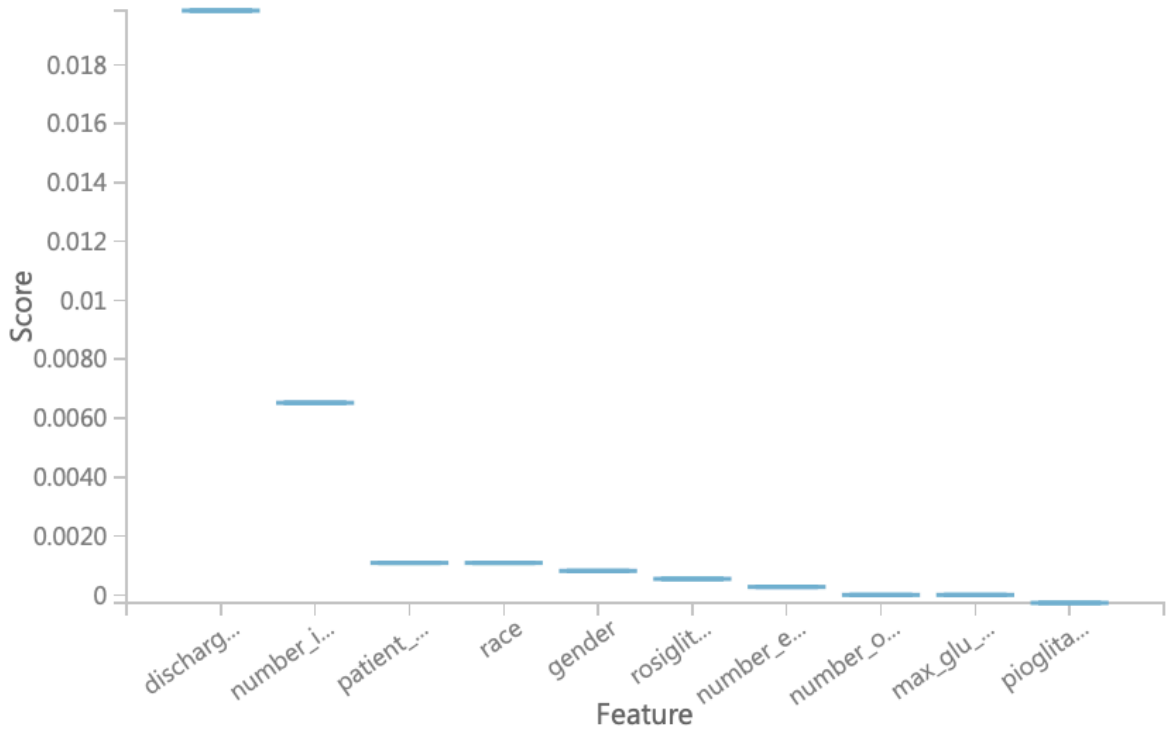


Figure 6 depicts the comparison of all the tested algorithms.

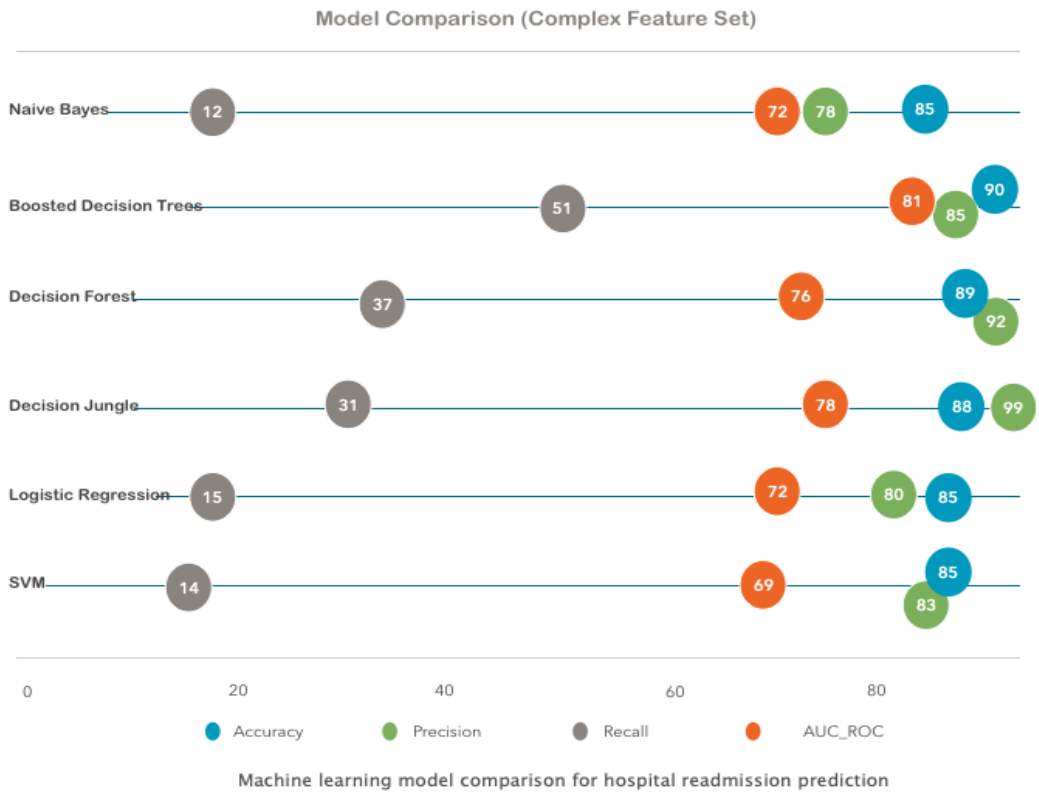


Figure 6 Model Comparison

3.2 Discussion

The difference between the performance of logistic versus tree-based models is clearly shown. Besides accuracy, recall is important since hospitals incur additional costs both for the patient and the insurance agencies if a patient expected not to be readmitted shows up within 30 days. With these important aspects in mind we can conclude that Boosted Decision Trees fit our data the best since not only does the model have high precision and accuracy, it also has the highest recall.

The interaction term number_diagnoses * time_in_hospital is the most important feature when it comes to accuracy and precision. This means that if we remove this feature from the model we lose the most predictive value for accuracy and precision. For a hospital manager, this means that patients who are likely to stay longer and have a high number of diagnoses, are highly likely to get readmitted. Discharge_disposition_id is the most important feature when it comes to recall which means that it contributes the most to predicting whether someone will be readmitted or not. Discharge_disposition_id indicates the reason why a patient gets discharged.

Tree-based models outperformed logistic regression in our study and this could be due to decision boundaries being non-linear.

Our study approach was based on a study done by Asri et al. (2016) who did a performance comparison between different machine learning algorithms: Support Vector Machine (SVM), Decision Tree (C4.5), Naive Bayes (NB) and k Nearest Neighbors (k-NN). The study found that SVM reaches the highest accuracy of 97.13% (Asri et al., 2016). However, this pattern cannot be observed in this study, where SVM was out performed by all other tested algorithms. However, while model accuracy is important, model interpretability is also important when it comes to predicting hospital readmission. SVMs are limited by their

interpretability to gain insights on how different features are assigned importance. Naive Bayes uses time-sequence information of what came before and what came after the variable being predicted and since the used dataset does not contain this information, it makes it also not interpretable.

Limitations

We were limited by the amount of data we had. We did not have all the factors that could have an effect on hospital readmission. Also, our dataset does not contain important features such as access to healthcare which is shown to account for variation in readmission rates. Data collected over longer periods of time can also contribute towards better performance of our tested models. We also had to use synthetic data for the readmitted within 30 days class because the dataset was highly imbalanced.

5 Future Work

The following recommendations are for future research on the topic of prediction of hospital readmission for diabetes patients. With these recommendations we hope that future research on this topic will be improved and to benefits of real diabetes patients.

Our dataset suffered of big imbalance between readmission and non-readmission. The non-readmission group was much larger and therefore we had to use a synthetic sampling technique called SMOTE to create synthetic data and get sufficient prediction results. If in the future data is collected over a longer period of time, and therefore there is a bigger number of readmissions collected. This creation of synthetic is not necessary and prediction of our models has a closer relationship to reality.

Future research could possibly benefit a lot from adding different relevant attributes as well. The attribute weight is dropped, because it was missing for 98% of our dataset. This might have a big effect on hospital readmission, but could not be measured in our study. Another important attribute that is not a included in the dataset is ‘Acces to care’. Possibly patients did not get re-admitted in the hospital, because they did not have access to care anymore.

Our research can be useful for further research and application in the area in many ways. It can be used as a starting point for creating an application that can alarm the danger of readmission for a given patient based on his current condition. We have provided insights on which features add information, but this can be investigated further through more research.

6. Conclusion

We used the cleaned data to train a model from each of the selected machine learning algorithms. A comparison was made based on each model’s accuracy, precision, recall and AUC. The graph below visualizes these 4 factors of each of our models. After comparing the models, we concluded that Boosted Decision Trees is the best performing model for what we need. Although not showing the highest precision, BDT outperformed all other algorithms in terms of accuracy, recall and auc which we consider with higher importance. We therefore selected it as a winning algorithm for our case.

7 References

Fraze, T., Jiang, J., & Burgess, J. (2010). Hospital Stays for Patients with Diabetes, 2008. HCUP Statistical Brief #93. *Agency for Healthcare Research and Quality*, 1–11. https://doi.org/NBK52658 [bookaccession]

Jiang, H. J., Stryer, D., Friedman, B., & Andrews, R. (2003). Multiple hospitalizations for patients with diabetes. *Diabetes Care*, *26*(5), 1421–1426. https://doi.org/10.2337/diacare.26.5.1421

Robbins, J. M., & Webb, D. A. (2006). Diagnosing diabetes and preventing rehospitalizations: The urban diabetes study. *Medical Care*, *44*(3), 292–296. https://doi.org/10.1097/01.mlr.0000199639.20342.87