Robert Beetel
Udacity
Machine Learning Engineer Nanodegree
October 23, 2017

Capstone Project

Machine Learning Engineer Nanodegree

Definition

Project Overview

Patients admitted into Intensive Care Units (ICU's) are severely ill or injured and require a high level of care. These patients are at much greater risk of dying than typical hospital patients. ICU patients are also among the most expensive patients. The burden of care with regard to cost for ICU patients is disproportionately large with estimates of associated costs in the U.S. approaching 1% of national GDP[1] . In the U.K. ICU care costs account for 0.6% of National Health Expenditures[2].

The ability to predict mortality in ICU patients within the first 24hrs of admission may contribute to higher survival rates in that determination of illness severity may help to guide treatment decisions. Patients who are identified as more likely to die may receive more aggressive treatments which may increase their likelihood of survival. Prediction of mortality may also act to reduce treatment costs in that patients who are predicted more likely to survive, who may currently be treated very aggressively, may receive less aggressive and less expensive care without affecting their survival rates. Prediction of mortality also provides a tool for assessing the quality and effectiveness of novel treatments. Current prediction tools are useful but have been found to be inadequately calibrated and are not robust over broad ranges of circumstances[3].

---

[1] Critical care medicine in the United States 2000-2005: an analysis of bed numbers, occupancy rates, payer mix, and costs. Halpern NA, Pastores SM, Crit Care Med. 2010 Jan; 38(1):65-71.

[2] Cost effectiveness of adult intensive care in the UK., Ridley S, Morris S, Anaesthesia. 2007 Jun; 62(6):547-54.

[3] Mortality prediction in the ICU: can we do better? Results from the Super ICU Learner Algorithm (SICULA) project, a population based study. Pirracchio et. al., The Lancet. Respiratory Medicine, 3(1), 42–52. http://doi.org/10.1016/S2213-2600(14)70239-5

In this project a classifier was developed for predicting mortality for ICU patients given data from the first 24hrs of ICU admission. Patient data was collected from the the MIMIC-III (Medical Information Mart for Intensive Care III), a large, freely-available database comprising deidentified health-related data associated with patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012.

Data was pre-processed and features were selected. These features were used to train and test a number of candidate machine learning classifiers. Classifiers were optimized across their respective parameter spaces as well as across input feature set and training/testing data set sizes. A classifier was selected from the group which provided the best performance with regard to predicting patient mortality.

Problem Statement

The primary goal of this project is to identify high risk patients so that they may receive more aggressive treatment and decrease their likelihood of dying. To accomplish this a classifier was be developed which could generate an early prediction of patient survival based on initial (first 24hrs) lab results, chart events and patient demographic information. A secondary goal of the project and classifier was to identify lower risk patients who do not need to be treated as aggressively so that they may be provided appropriate levels of treatment and thereby reduce health costs.

The tasks involved were as follows:

1. Download and build local copy of Mimic III database
2. Query the database for patient demographic data and initial (first 24hrs) of laboratory and chart data for ICU stays with ICU stay defined as a unique instance of a patient being admitted into the ICU.
3. Pre-process data and select features based on the strength of their correlation with outcomes.
4. Train candidate classifiers to predict patient survival, optimizing each classifier along its respective parameter space, number of input features and the train/test split size.
5. Evaluate optimized classifier performance and select final model

Metrics

The metrics we will be using to assess classifier performance include:

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F-Scores = Harmonic Means of Precision and Recall:

*F-Beta Score = $(1 + \beta^2)$ * (Precision * Recall) / (β^2 * Precision + Recall)*

*F-1 Score = (Precision * Recall) / (Precision + Recall)*

Support = Number of occurrences of each class in y_test

where:

TP = True Positive

FP = False Positive

FN = False Negative

y_test = Set of results against which classifier predicted values are compared.

True Positives are predictions that correctly predict classify patients who are likely to die as high risk while True Negatives correctly classify a patient who is likely to survive as low risk. A False Positive is a prediction that a patient who is likely to survive will die.

False Positives would result in a patient who's condition does not require aggressive treatment being placed in a higher risk category than their circumstances merit and receiving more aggressive treatments. The cost of False Positives may be an increase in hospital cost for lower risk patients.

Conversely, a False Negative predicts that a patient more likely to die will survive and places that patient in a lower risk category where they may less aggressive treatment. The cost of False Negatives, of regarding a patient who is likely to die as a lower risk patient, would be that a very high risk patient receives less aggressive treatment than their condition warrants and would therefore be even more likely to die.

The primary goal of this project is to identify high risk patients so that they may receive more aggressive treatment to decrease their likelihood of dying. Given that, we would like our

classifier to predict mortality of patients with a maximum number of True Positives and a minimum number of False Negatives.

It was noted that, due to the small number of non-survivors, classifiers could achieve a maximum performance in some areas by predicting that a patient would survive no matter the data. Algorithms had a tendency to maximize average scores by maximizing performance for survivors at the expense of performance for non-survivors. This was especially the case in the scores most relevant to the projects goals, F1 and Recall.

The optimization metrics used to evaluate classifier performance was developed to perform the dual function of maximizing recall and f-scores for both survivors and non-survivors while minimizing the difference between them.

A metric was used which was a combined sum of the two scores, minus the absolute value of the difference between them. The intention was to avoid the scenario in which a maximum average value is reached by maximizing one score very high at the expense of the other.

The equation below shows how the optimization metric for F1-score would be calculated.

F1 Optimization Metric = (F1-Survivor + F1-Non_Survivor) - abs(F1-Survivor - F1-Non-Survivor)

Figure 1 below provides a simplistic illustration of the performance curve of a classifier that trades off performance for survivors and non-survivors. The metric shown is maximized when two conditions are met: the survivor and non-survivor scores are at their maximum AND the difference between them is at it's minimum.

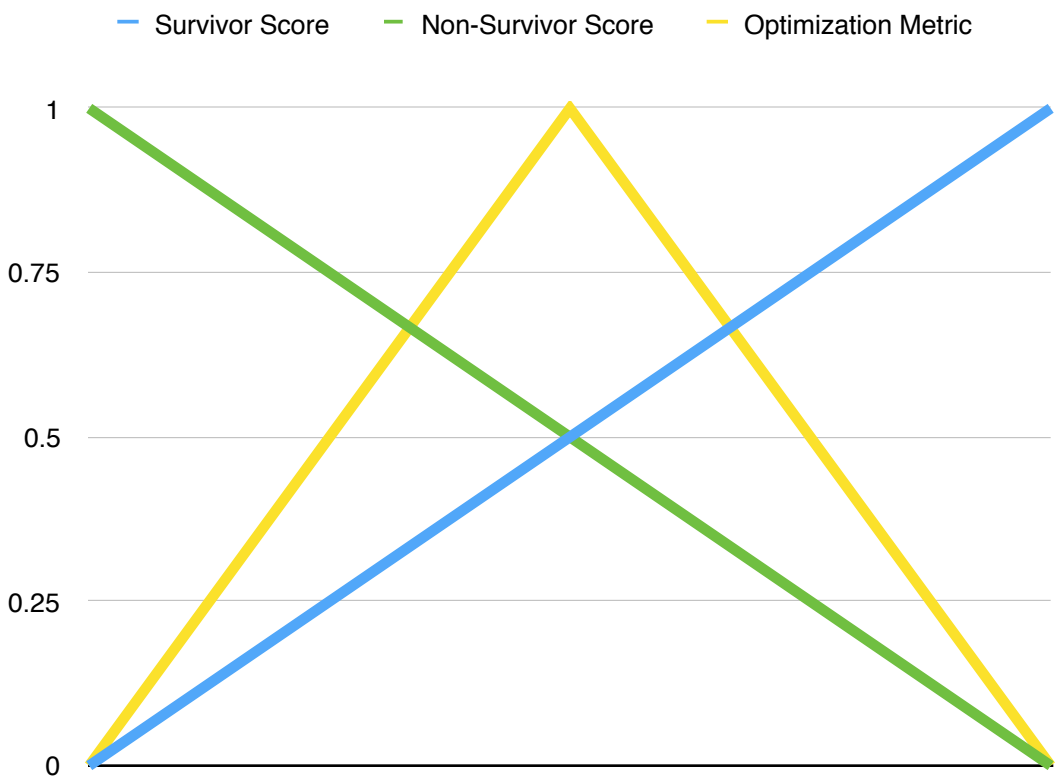


Figure 1: Illustration of the optimization metric used for maximizing recall and F-scores while minimizing the difference between them.

Analysis

Data Exploration

Patient data was collected from the the MIMIC-III (Medical Information Mart for Intensive Care III), a large, freely-available database comprising deidentified health-related data associated with patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012.
In addition to data on juvenile patients, the database contains data associated with 53,423 ICU admissions for adult patients (aged 16 years or above) covering 38,597 distinct adult patients and 49,785 hospital admissions. Data includes information such as patient demographics, vital sign measurements made at the bedside (~1 data point per hour), laboratory test results,

procedures, medications, caregiver notes, imaging reports, and mortality (both in and out of hospital). In-hospital mortality for adult patients was observed to be 11.5%.[4,5]
Data was queried in three major groups or categories:

- Patient Demographics
- Chart Data
- Lab Data

Data samples were defined as unique ICU stays with mortality during each stay as the sample outcome and data collected within the first 24 hours of the stay as the sample input features.

Patient demographic data was queried from the ADMISSIONS, ICU_STAY and PATIENT tables in the database and included information like patient age, sex, marital status, nationality etc. In addition, patient diagnoses were identified based on HCUP CCS 2015 diagnostic groups and ICD9 codes[6] . A subset of 25 diagnoses were included. With the exception of diagnoses, where a patient may have had more than one diagnosis per ICU stay, demographic data typically was constant in that there was only one measurement or entry per ICU stay. Demographic data consisted primarily of categorical data with some continuous and date/time variables.

Lab data was queried from the LABEVENTS table and included laboratory results like hematocrit, WBC count, oxygen saturation, and blood glucose measurements. Chart data was queried from the CHARTEVENTS table and included information captured on the patients chart including blood pressure, heart rate, temperature and blood gas measurements. Also included in chart data were Glascow Coma Scale (GCS) evaluations.

There were a large number of variables or features collected from the database which included continuous and categorical data. A discussion of the characteristics and distributions of all of the variables is considered beyond the scope of this writeup so the following section will instead focus on variables that were ultimately included in the final feature set.

Age and ICU stay were determined to be significant features for classification. Table 1 below shows the descriptive statistics for Age and ICU stay for both Survivors and Non-Survivors. It can be seen that Non-Survivors tended to be older and had longer ICU stays than Survivors.

---

[4] MIMIC-III, a freely accessible critical care database. Johnson AEW, Pollard TJ, Shen L, Lehman L, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, and Mark RG. Scientific Data (2016). DOI: 10.1038/sdata.2016.35. Available at: http://www.nature.com/articles/sdata201635

[5] Pollard, T. J. & Johnson, A. E. W. The MIMIC-III Clinical Database http://dx.doi.org/10.13026/C2XW26 (2016).

[6] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp

	Survivors		Non-Survivors	
	age	icu_stay	age	icu_stay
count	37044	37044	4873	4873
mean	63.0	108.8	68.2	176.7
std	15.9	150.2	14.7	204.3
min	17.0	25.0	19.0	25.0
25%	53.0	39.0	58.0	53.0
50%	65.0	60.5	71.0	105.0
75%	76.0	111.0	80.0	214.0
max	89.0	4154.0	89.0	2336.0

Table 1 Summary statistics on Age and ICU stay duration for ICU patients.

Table 2 shows the distributions of the different diagnoses among survivors and non-survivors. The diagnoses shown would later be included in the final feature set.In that final feature set, there were 2665 icu-stays analyzed with 2222 survivors and 443 non-survivors. It can be seen that of those patients, 43% of non-survivors were diagnosed with Respiratory Failure while only 18% of Survivors had this diagnoses.

	Number of Survivors	Number of Non-Survivors	Percent of Survivors	Percent of Non-Survivors
Acute and unspecified renal failure	562	228	25%	51%
Acute cerebrovascular disease	27	30	1%	7%
Fluid and electrolyte disorders	529	154	24%	35%
Other liver diseases	167	58	8%	13%
Pneumonia	245	73	11%	16%
Respiratory Failure	406	191	18%	43%
Septicemia (except in labor)	449	197	20%	44%
Shock	238	130	11%	29%

Table 2. Distribution of diagnoses among survivors and non-survivors. The diagnoses shown were in the final feature set which included 2,665 cu-stays with 2,222 survivors and 443 non-survivors.

For each diagnosis, Table 3 shows the distribution of survivors and non-survivors From the table we can see that 71% of patients diagnosed with Acute and Unspecified Renal Failure were survivors i.e. patients with this diagnosis had a 71% survival rate. Patients diagnosed with Acute Cerebrovascular disease had a much lower survival rate of 47%.

	Non-Survivors	Survivors
Acute and unspecified renal failure	29%	71%
Acute cerebrovascular disease	53%	47%
Fluid and electrolyte disorders	23%	77%
Other liver diseases	26%	74%
Pneumonia	23%	77%
Respiratory Failure	32%	68%
Septicemia (except in labor)	30%	70%
Shock	35%	65%

Table 3. The distribution of Survivors and Non-Survivors for patients diagnosed with each condition. For example, for patients diagnosed with electrolyte disorder, there was a 77% survival rate. For patients diagnosed with Acute Cerebrovascular Disease, the survival rate fell to 47%.

The type of ICU to which a patient was admitted was also identified as a relevant feature. Table 4 shows the distribution of the type of ICU in which a patient was treated as well as the mortality rate or % non-survivors for each admission type.

	Percent of Patients	Mortality Rate
MICU	39%	16%
CSRU	19%	4%
SICU	16%	13%
CCU	15%	11%
TSICU	11%	12%

Table 4. Distribution of ICU type (MICU = Mobile Intensive Care Unit, CSRU = Child Support Recovery Unit, SICU = Surgical Intensive Care Unit, CCU = Critical Care Unit, TSICU = Trauma Surgical Intensive Care Unit) Also shown are the mortality rates for patients admitted to each.

Table 5 illustrates the distribution of admission types. Also shown are the mortality rates for patients admitted under each type. It can be observed that patients admitted under ELECTIVE had lower mortality rates than under EMERGENCY or URGENT.

	admission_type	hospital_expire_flag
EMERGENCY	83%	14%
ELECTIVE	14%	4%
URGENT	3%	13%

Table 5. The distribution of admission type. Also shown are the mortality rates for patients admitted under each type. It can be observed that the mortality rate for ELECTIVE admission type was far lower than that for the other types.

Exploratory Visualization

Laboratory and chart data including measures like blood pressure (BP) and lactate-dehydrogenase (LacDehyd) and creatine were primarily continuous variables. Figures 2-4 below show the distributions or density plots for lab measures for survivor and non-survivor groups.

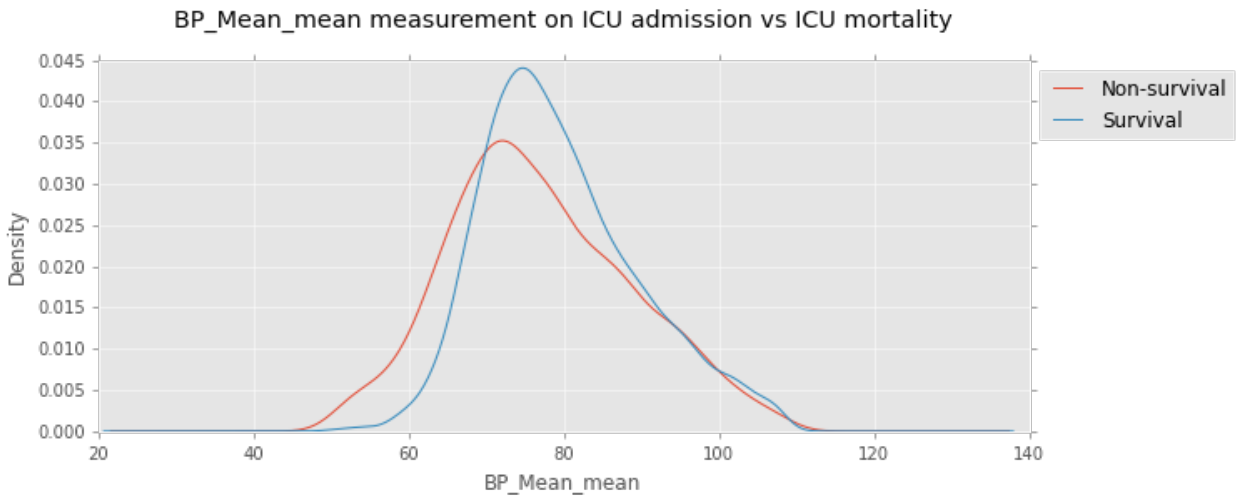


Figure 2: The data distributions for mean blood pressure for females in survivor and non-survivor groups.

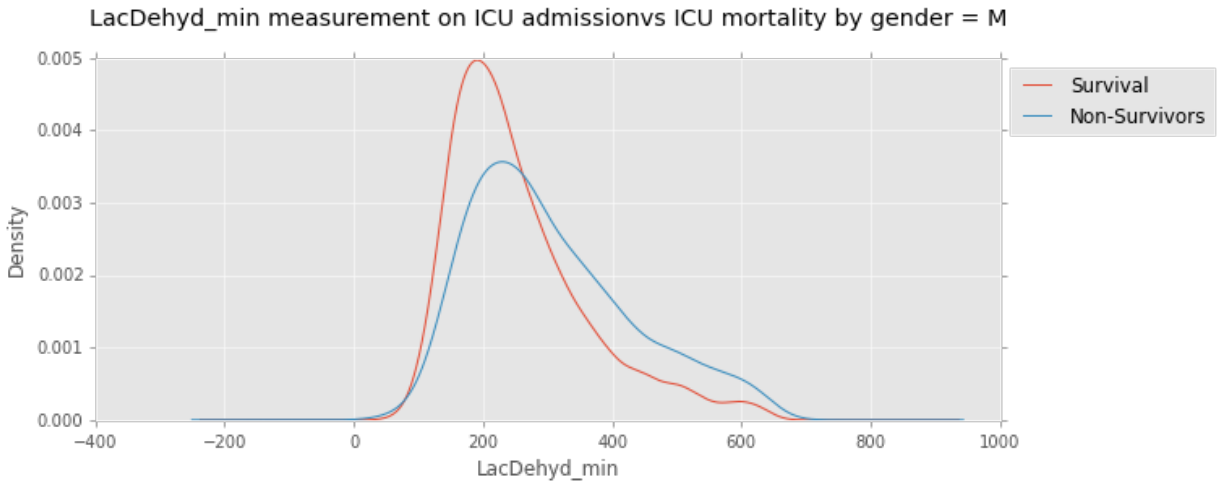


Figure 3: The data distributions for minimum lactate-dehydrogenase for females in survivor and non-survivor groups.

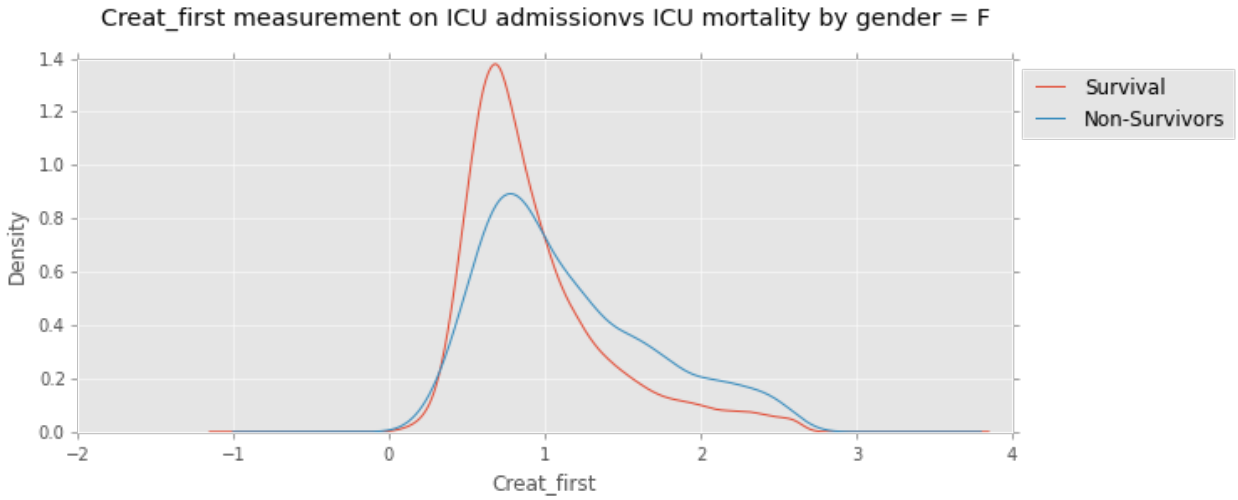
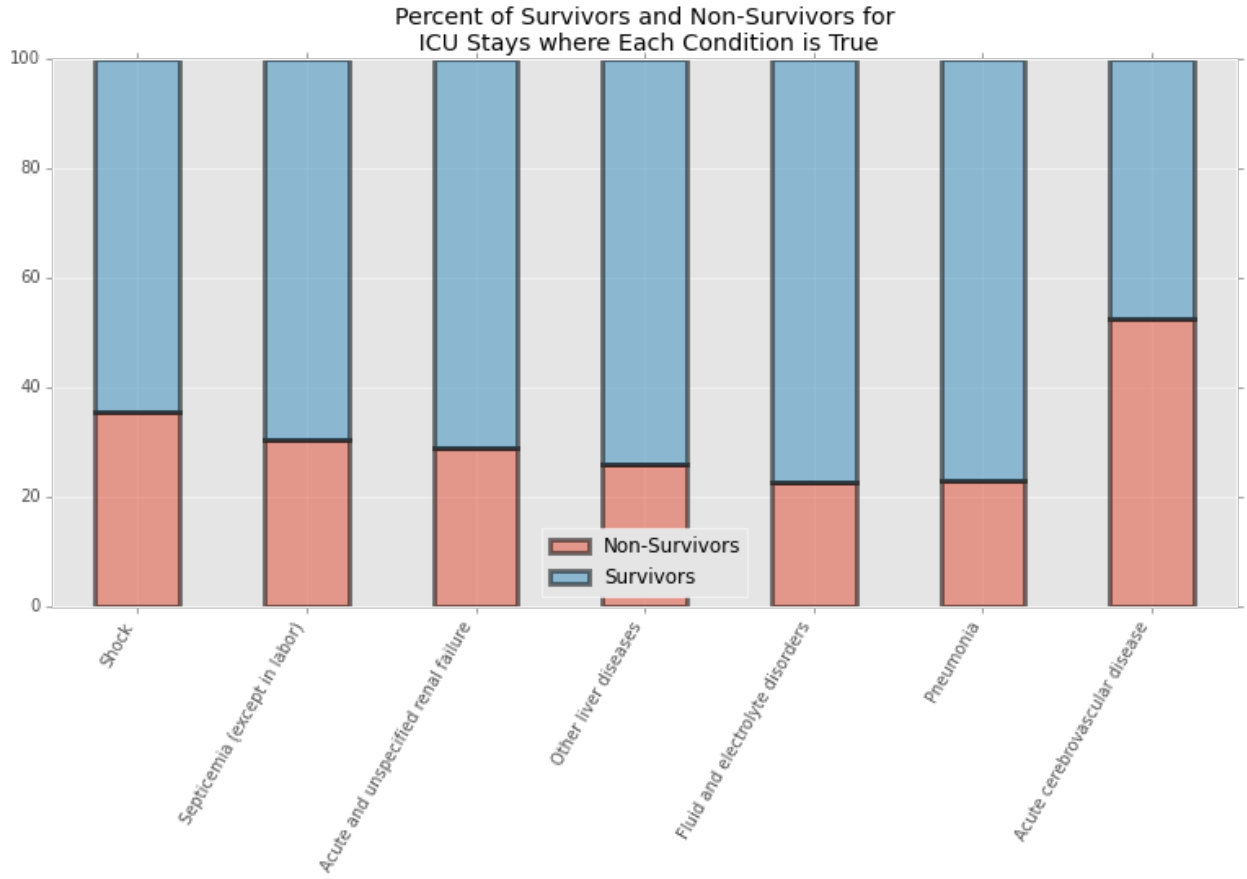


Figure 4: The data distributions for first measures of creatine for females in survivor and non-survivor groups.

Categorical data included data like patient diagnoses. The distributions of categorical features were visualized in one of two ways:

1. For a given category or variable, plotting the percentages of Survivors and Non-Survivors for ICU stays in which the category or condition was True
2. For a given category or variable, plotting the percentage of ICU stays where the category or condition was True for Survivors and for Non-Survivors.

Figure 5 below shows a plot of the percentages of survivors and non-survivors for ICU stays where the select diagnoses were present or True. For example, for ICU stays in which the diagnoses included Acute Cerebrovascular Disease, roughly 50% of patients survived while the remaining 50% did not. The mortality rate for patients diagnosed with Acute Cerebrovascular Disease was therefore approximately 50%



Figures 5 The percentage of survivors and non-survivors for ICU stays in which each Diagnoses is present or True.

Algorithms and Techniques

The task presents itself as a supervised classification problem with output as two classes, survivors and non-survivors, and inputs as categorical/dummy features. It was believed that given the relatively high dimensional, complex feature space the optimum decision boundary would likely be non-linear. Classifiers selected for evaluation were those that work well in higher dimensional space with categorical data and were capable of generating linear and non-linear decision boundaries.

The classifiers evaluated included:

- Support Vector Classifier (SVC)
- Linear Support Vector Classifier(LSVC)
- K-Nearest Neighbor Classifier (KNeighbors)
- Multi-Layer-Perceptron Classifier (MLP)
- Decision Tree Classifier (Tree)

The support vector machine classifiers, SVC and LSVC, fit the criteria in that they they work well in higher dimensional spaces by reducing the feature space to a lower dimensional support vector space. The support vector classifiers are also flexible in that different kernel functions can be employed some of which are capable of generating linear or non-linear decision boundaries. LSVC is a support vector machine that utilizes a linear kernel function while the SVC classifier may employ polynomial, radial basis or sigmoid functions producing non-linear decision boundaries.

Nearest neighbor classifiers store labeled training data and assign class to a test point by simple majority vote of the nearest training data points nearest to the test point. The test point is assigned to the class to which the majority of neighboring training data points are assigned. The classifier is capable of generating highly non-linear decision boundaries. in relatively high dimensional space.

Multi-layer perceptron classifiers are multi-layer neural networks. The multi-layer perceptron classifier evaluated had a 3-layer, input, middle and output layer, architecture. The classifier was evaluated over a range of activation functions and solvers.

Lastly, Decision Tree classifiers are algorithms which infer simple sets of decision rules from training data. Decision Tree decision boundaries can range from very simple to highly complex.

Benchmark

To create a benchmark for classifier performance, the K-Nearest Neighbors classifier was selected for it's simplicity. The K-Neighbors Classifier was trained and tested using the full feature set as inputs and a train/test split of 80/20%. The algorithm was trained using default parameter settings. Precision, recall, f1 and support scores for survival and non-survival groups were calculated and are shown below in Table 6. Table 7 shows the benchmark optimization metrics scores for both F1 and Recall

BENCHMARK PERFORMANCE

		precision	recall	f1-score	support
Classifier	**Classes**				
KNeighborsClassifier	**Avg/Total**	0.79	0.83	0.81	533.0
	Non-Survivors	0.31	0.17	0.22	75.0
	Survivors	0.87	0.94	0.90	458.0

Table 6. Precision, f1, recall support scores are shown for the K-Nearest Neighbors classifier. The algorithm was trained and tested using default parameters, 20 input features and an 80/20% train-test split. Of note, survivor recall and f1 scores are very high while corresponding non-survivor scores are very low.

Classifier	F1 Metric	Recall Metric
KNeighborsClassifier	0.44	0.34

Table 7. F1 and Recall optimization metric scores for selected classifiers. Classifiers were trained and tested using default parameters, 20 input features and an 80/20% train-test split.

Methodology

Data Pre-Processing

Lab and chart data typically consisted of continuous data with multiple measurements taken for each variable over the first 24 hours. The continuous / time-series nature of lab and chart data required reducing dimensionality by using summary statistics like mean, median and standard deviation, as well as additional characterizations like the slope of the data trend line and the difference between first and last measurements.

Features from the different sources, demographics, lab and chart data, were pre-processed and evaluated separately.

Data included a mixture of continuous , date/time and and categorical data. For classification, all data was converted to categorical data in the form of ‘dummy’ variables.

Date/time data including duration of ICU stay and age were converted to continuous variables by transforming the date-time durations to number of hours or number of years.

Density plots were generated from continuous data to visualize their distributions. It was noted that there were extreme outliers in the data sets that were skewing the distributions significantly. Outliers, defined as data points outside 1.5 times the interquartile range, were removed and data distributions were re-visualized and re-evaluated for skewness. Figure 6 shows the distribution of white blood cell counts (WBCs) for male patients separated into survivors and non-survivors. Figure 7 shows the distribution of that same data set with outliers removed.

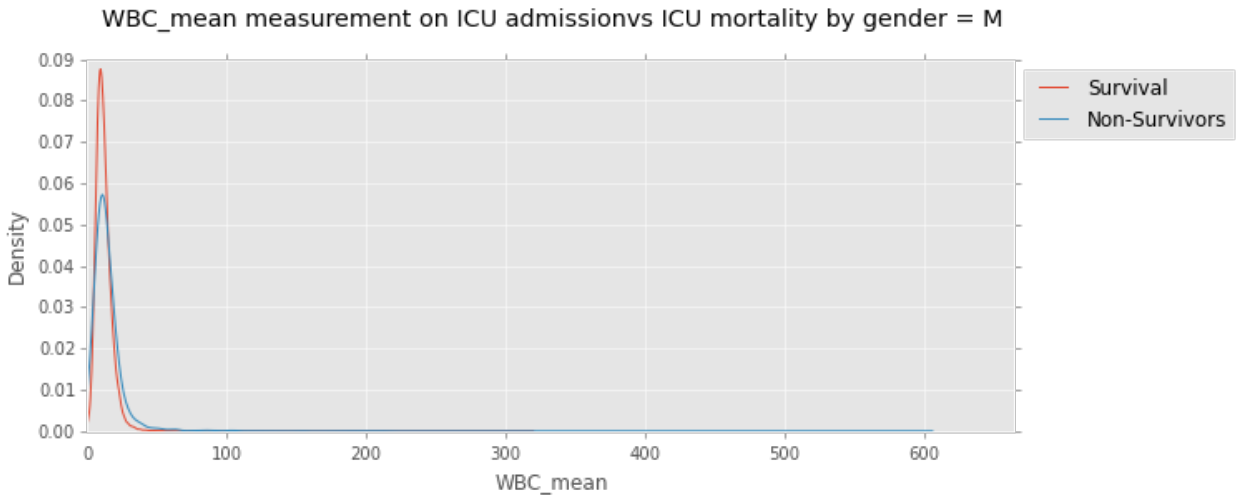


Figure 6: Distribution of of white blood cell counts (WBCs) for male patients separated into survivors and non-survivors.

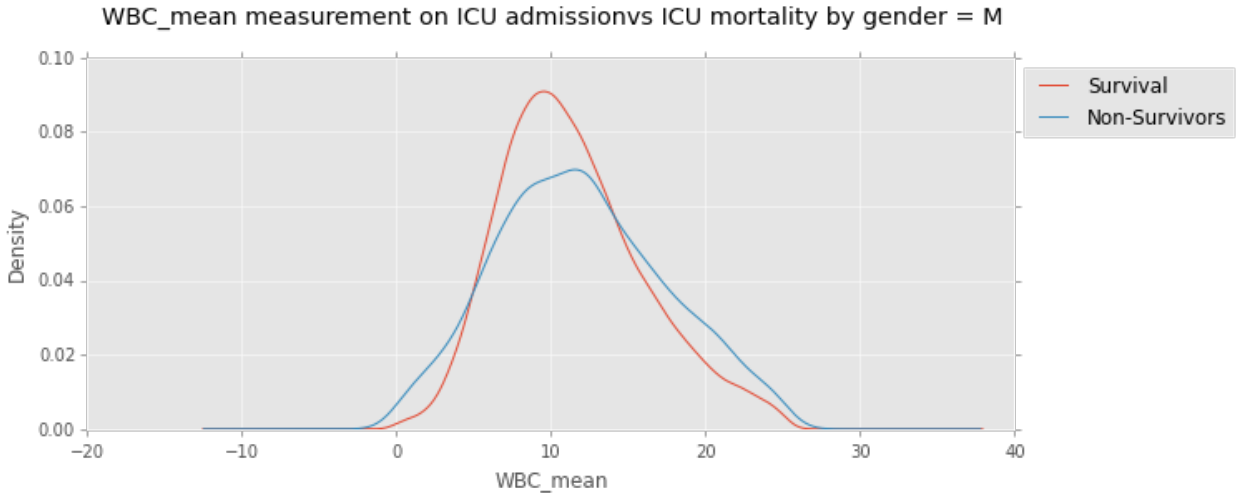


Figure 7: Distribution of of white blood cell counts (WBCs) for male patients separated into survivors and non-survivors with outliers removed.

It was determined that the distributions for continuous features were sufficiently normal that transformations (like Box-Cox) to further normalize data were not required.

Continuous data was converted into categorical data by calculating the quartiles for each features distribution and labeling each data point based on into which quartile fell. Figure 8 below shows the distribution of mean temperature measurements for survivors and non-survivors over the first 24 hours. Quartiles are indicated as vertical lines labeled Q1, Q2 and Q3. A data point to the left of Q1 would be labeled ‘Q0’, a data point to the right of Q1 and to the left of Q2 would be labeled ‘Q1’ and so on creating 4 potential labels for each point. Once all data was converted to categorical, features were then converted into dummy variables

where for a categorical variable, each category has it's own column or feature and each data points is represented as a 1 in the category column to which it belongs and a 0 in the others.

Data pre-processing and classification steps, including feature selection require there to be no missing data points for any feature in a sample.

Two primary ways for dealing with missing data are:

1. Imputation of missing data points from existing data
2. Dropping samples for which data is missing.

Because there was a relatively large number of missing data points the decision was made to drop missing samples rather than impute values from what was for some features, relatively sparse data.

In our initial data set there were relatively large numbers of missing data points. Not all features were present for each sample and there were very few 'complete' samples for which all data from any one source was present. For example, if we created a table with all the lab results features and dropped all samples for which there was a missing feature, we would be left with nearly zero samples.

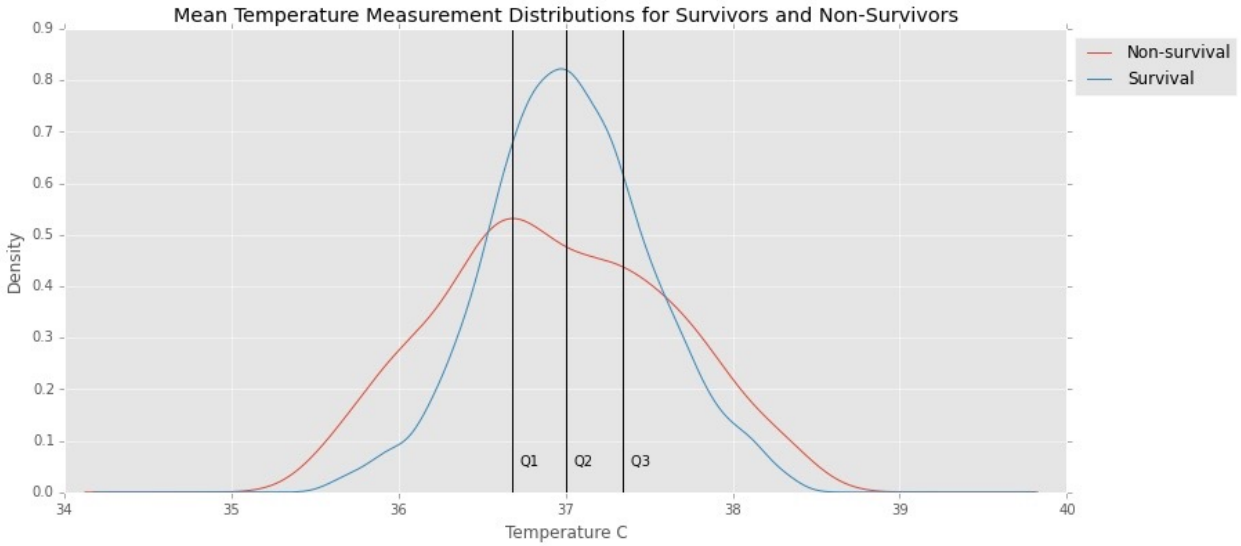


Figure 8. The distribution of mean temperature measurements is shown for survivors and non-survivors over the first 24 hours. Quartiles are indicated as vertical lines labeled Q1, Q2 and Q3.

It was observed, however, that there appeared to be features that were more often collected together. The strategy became to first, remove features which were present in only a small number of samples, then to organize features that were likely to be collected together into groups. Each group could then be pre-processed and its features evaluated separately. The feature selection phase would filter out some features and those that were selected could be

brought together in a way that balanced their selection criteria scores with the number of complete samples remaining when data was recombined.

To identify affinity groups, features which were commonly present together, we created ‘missing’ data frames, frames in which the value for a given measurement and ICU stay was zero if measurements were present and 1 if measurements were missing. This allowed for correlation coefficients to be calculated between each of the features to measure how often variables were missing together in the same ICU stays. Table 8 shows correlation between missing data points in laboratory features.

CORRELATION FOR MISSING VALUES	Creatinine	Glucose	Hematocrit	Lactate	Lactate Dehydrogenase	Oxygen Saturation	pH	WBC
Creatinine	1.00	0.06	0.01	0.11	0.17	-0.02	0.13	0.02
Glucose	0.06	1.00	0.13	0.04	0.08	-0.11	-0.04	0.16
Hematocrit	0.01	0.13	1.00	0.04	0.03	-0.01	0.01	0.73
Lactate	0.11	0.04	0.04	1.00	0.14	0.27	0.44	0.06
Lactate Dehydrogenase	0.17	0.08	0.03	0.14	1.00	-0.02	0.07	0.04
Oxygen Saturation	-0.02	-0.11	-0.01	0.27	-0.02	1.00	0.34	-0.01
pH	0.13	-0.04	0.01	0.44	0.07	0.34	1.00	0.03
WBC	0.02	0.16	0.73	0.06	0.04	-0.01	0.03	1.00

Table 8: Autocorrelation table showing between features shows correlations between missing features in each category.

Creation of missing frames allowed for the creation of ’affinity maps’, color maps which show missing data in red and present data in blue. This enabled visualization of the data set to evaluate any trends in patterns of missing data. Figure 4. shows an affinity map in which data was sorted by Oxygen Saturation measurements with the y-axis representing individual ICU-stays. It can be observed visually that Oxygen Saturation and pH measurements are often taken together.

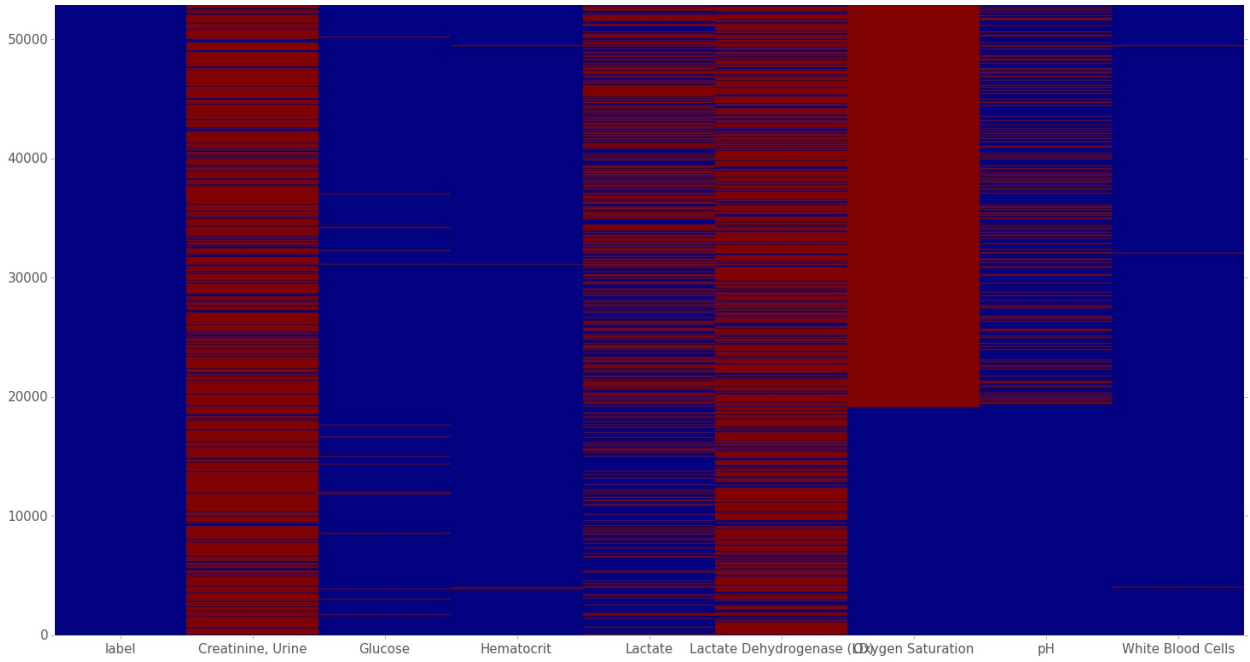


Figure 8. Affinity Map: Map of features along the x-axis and unique CU stays on the y-axis. Data points that are missing appear in red while data points that are present appear in blue. The map shows that Oxygen Saturation and pH measurements are often taken together

Chart features were grouped into 4 blocks with a total of 103 features that met the criteria (p-values <= 0.001) for being passed along to the recombination and classification phase. Lab data was divided into 2 blocks with a total of 20 features and patient demographic data was processed in a single block of 46 features which were passed along.

In the classification phase, the selected features from all sources were recombined and ranked according to chi2 score / p-value. From that group, the top 20 features were selected and recombined resulting in a data set which 20 features for 2,665 unique ICU stays.

Features included measures such as:

- GCS Scores (4)
- Respiratory Rate Mean
- Creatine
- First Care Unit (2)
- ICU Stay Duration (2)
- Age
- Admission Type

In addition, diagnosis data was included:

- Respiratory Failure
- Shock
- Septicemia
- Acute and Unspecified Renal Failure
- Liver Diseases
- Fluid and Electrolyte Disorders
- Pneumonia
- Acute Cerebrovascular Disease

Figures 9 and 10 illustrate the percentage of survivors and survivors for ICU stays in which each feature is True. For example, the above features. For example, in figure 5, for ICU stays in which the diagnoses included Acute Cerebrovascular Disease, roughly 50% of patients survived while the remaining 50% did not.

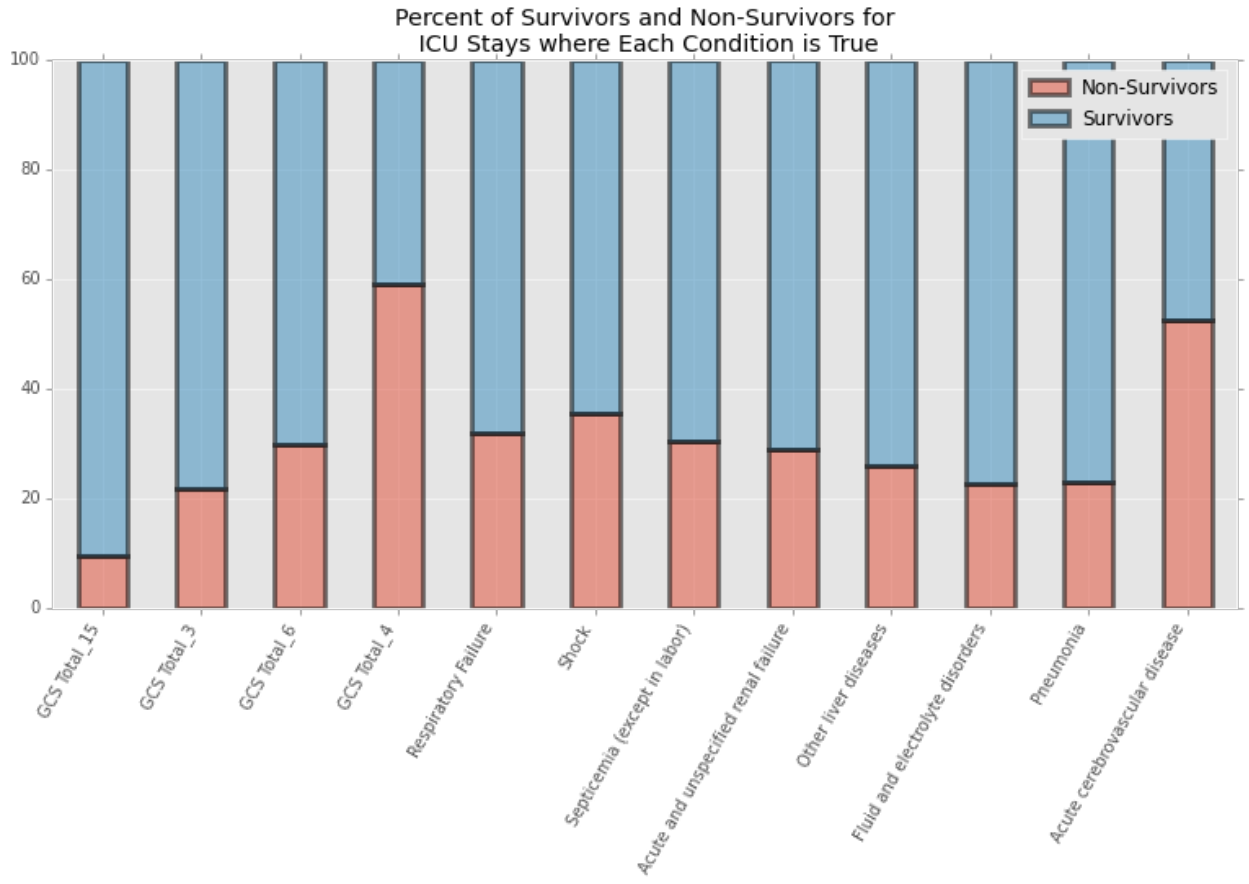


Figure 9: For ICU Stays in which each condition is True, the figure illustrates the percentage of those ICU stays in which the patients were survivors and non-survivors. For example, in figure 5, for ICU stays in which the diagnoses included Acute Cerebrovascular Disease, roughly 50% of patients survived while the remaining 50% did not.

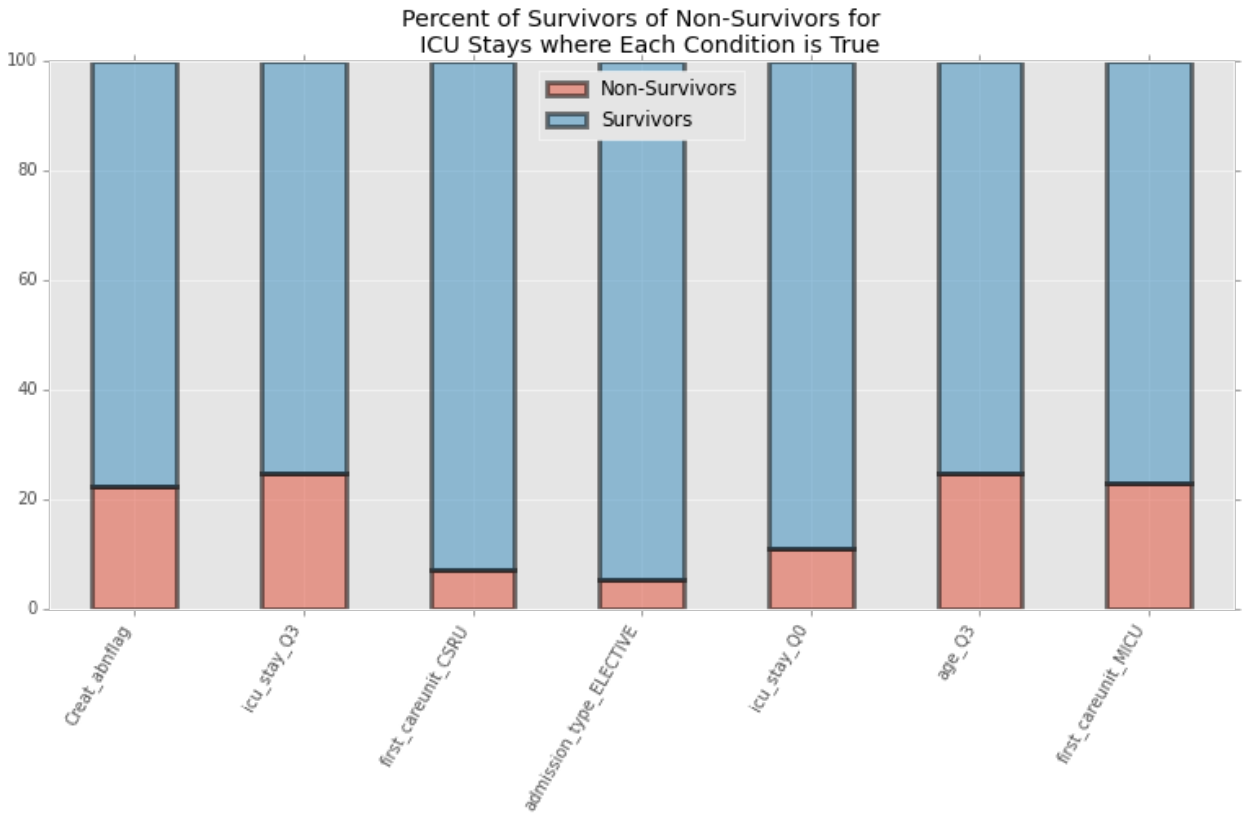


Figure 10: For ICU Stays in which each condition is True, the figure illustrates the percentage of those ICU stays in which the patients were survivors and non-survivors. For Example, in ICU stays in which the admission was of the type ‘ELECTIVE’, over 90% of patients survived and less than 10% did not.

Figures 11 and 12 illustrate the percentage of patients for which the feature was true for survivors and non-survivors. For example, in Figure 11, for ICU stays in which the patient did not survive,, the percentage of patients for which the diagnoses included ‘Respiratory Failure’ was approximately 45% while the percentage of survivors for which the diagnoses included ‘Respiratory Failure’ approached zero.

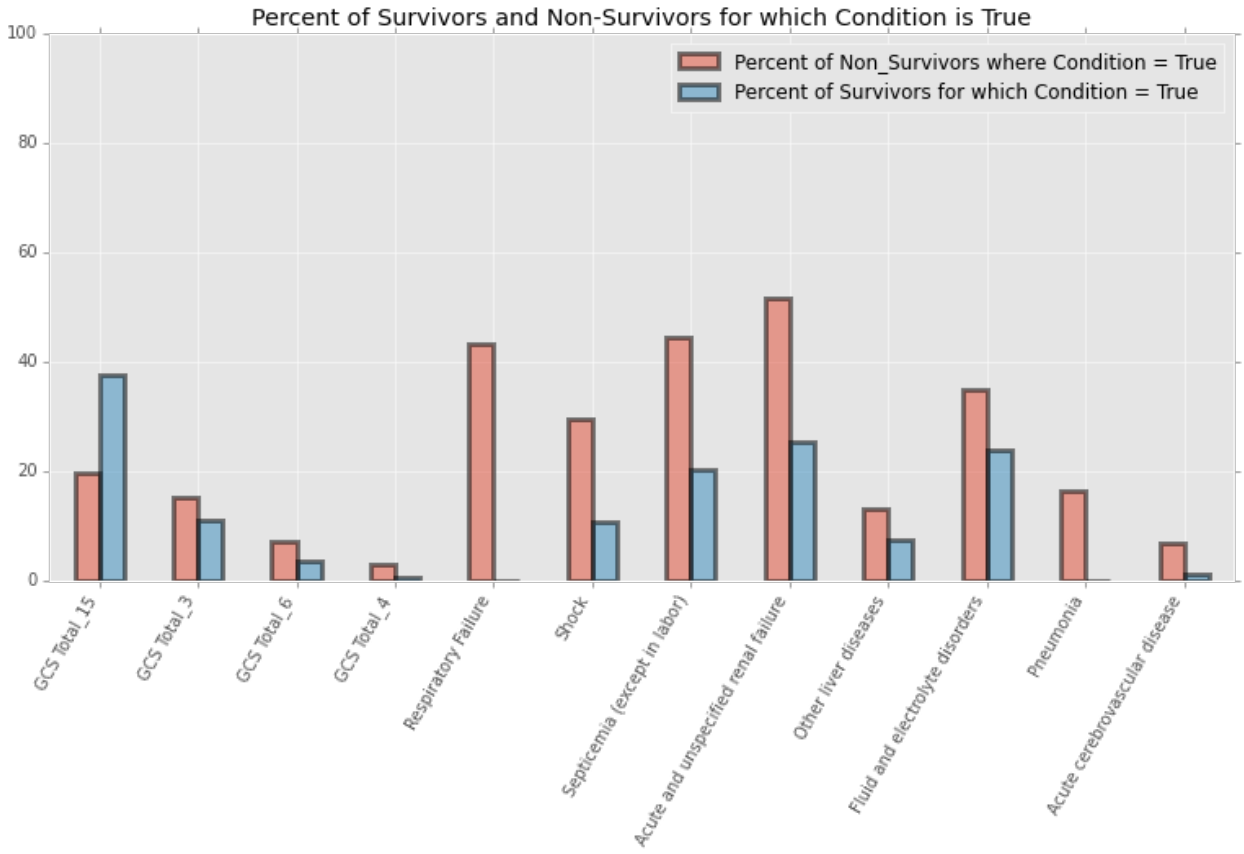


Figure 11: For ICU Stays in which each patients survived and did not survive, the figure illustrates the percent of those survivors and non-survivors for which the condition is True,For Example, in ICU stays in which the patient did not survive,, the percentage of patients for which the diagnoses included ‘Respiratory Failure’ was approximately 45% while the percentage of survivors for which the diagnoses included ‘Respiratory Failure’ approached zero.

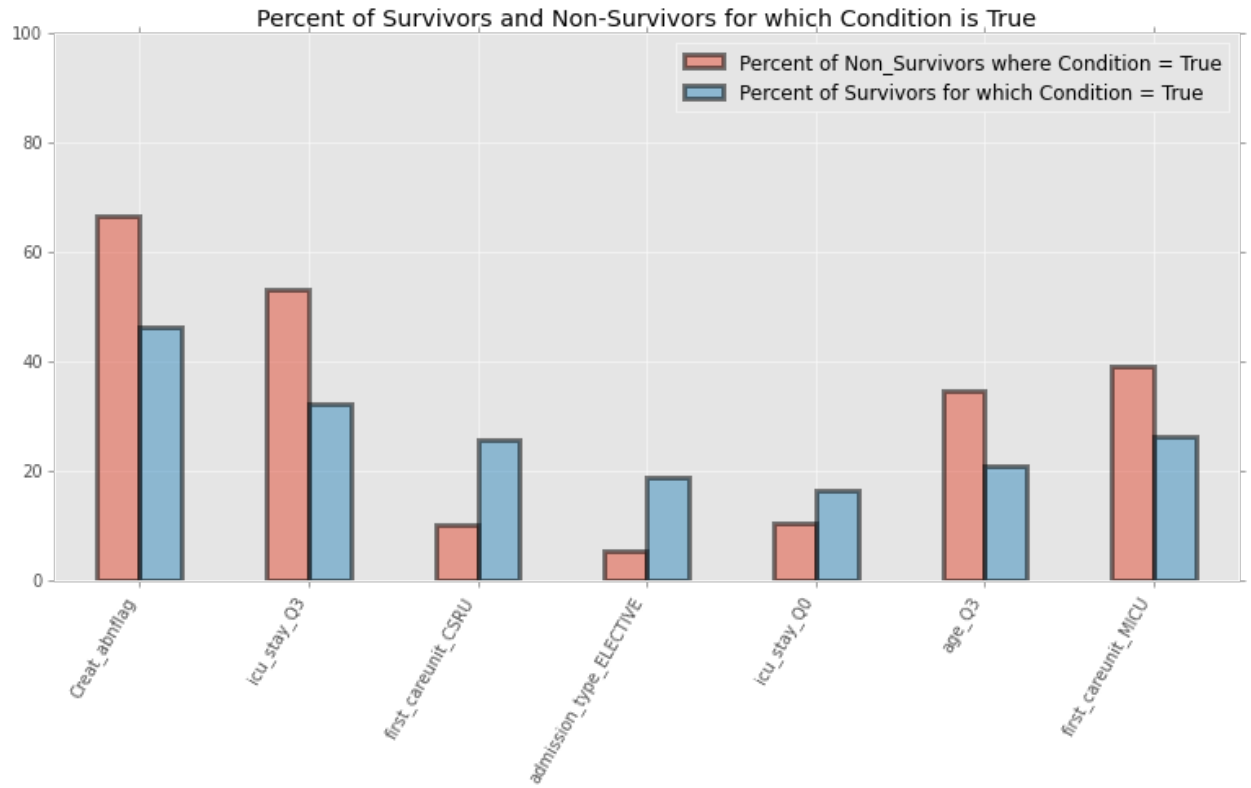


Figure 12: For ICU Stays in which each patients survived and did not survive, the figure illustrates the percent of those survivors and non-survivors for which the condition is True,For Example, in ICU stays in which the patient did not survive,, the percentage of patients for which the admission was of the type ‘ELECTIVE’ was approximately 5% while the percentage of survivors for which admission was of the type ‘ELECTIVE’ was 20%.

Implementation

With the difficult work of mining the data, separating out features into affinity groups, pre-processing and re-combining the features complete, we move on to the relatively straightforward task of implementation.

The top twenty features, ranked in order of chi2 scores / P-values, were organized in a data frame along with corresponding patient mortality outcomes.The data set was divided into features and outcomes, then split into training and testing data sets with the training set comprising 80% of the data and the remaining 20% used for testing.

BENCHMARK CLASSIFIER PERFORMANCE

		precision	recall	f1-score	support
Classifier	**Classes**				
KNeighborsClassifier	**Avg/Total**	0.79	0.83	0.81	533.0
	Non-Survivors	0.31	0.17	0.22	75.0
	Survivors	0.87	0.94	0.90	458.0

Table 9. Precision, F1 and Recall scores for Benchmark K-Nearest Neighbor Classifier.

The selected classifier, the K-Nearest Neighbors algorithm, was trained and tested using the default parameters. The results from these tests comprised the benchmark described above in Table 9.

Refinement

The implementation stage was relatively straightforward. Classifiers were first optimized over their respective feature spaces and over feature set sizes in 2 nested loops. The classifiers with optimized parameters and feature set sizes were then further optimized over data train/test splits ranging from 90/10% to 50/50%.

The goal was to simultaneously maximize the values of recall and f-scores in general and in particular to minimize false negatives.Classifiers were optimized to simultaneously maximize recall and f-scores while minimizing the difference between.

To maximize the scores for survivors and non-survivors simultaneously and avoid the situation in which non-survivor scores would be sacrificed for higher survivor scores, a metric was created which was a combined sum of the two scores, minus the absolute value of difference between them. The intention was to avoid a scenario in which one score is very high but the other is very low.

The equation has been described above but is shown here again to demonstrate how the optimization metric for F-scores would be calculated.

> F1 Optimization Metric = (F1-Survivor + F1-Non_Survivor) - abs(F1-Survivor - F1-Non-Survivors

In the first stage of optimization in which 2 nested loops were used, the outer loop iterated over input feature set sizes or, the number of features to be included in the input feature set, while the inner loop iterated over the different classifiers, using the GridSearchCV function to optimize each classifier to the given feature set size.

For the outer feature set size loop, features were added in decreasing levels of chi-2 score. The features with highest correlation were included first, then features with progressively lower correlations were added in order. The evaluation began with the top 8 rated features as inputs and progressed adding features in ranked decreasing order of chi-2 score until the maximum feature set size of 20 was reached.

To optimize using GridSearchCV, for each of the selected classifiers, subsets of the parameters space were defined as parameters and ranges of values for those parameters. For example, the SVC has among it's parameters:

- C: penalty parameter / error term
- kernel: there kernel type used in the algorithm
- degree: for the polynomial type kernel, the degree of polynomial
- class weight: the weight assigned to samples of a given class.
- decision function: use a one-vs-one or one-vs-all decision function

To explore this subset of the SVC parameter space the parameters and the values for those parameters over which the classifier should be evaluated were defined. In code this took the form of a data structure like dictionary shown below:

```
SVC_params = {'C':[0.3, 0.4, 0.5, 0.6, 0.7,0.8,0.9,1],
              'class_weight': [{1:3, 0:1}, {1:4, 0:1}, {1:5, 0:1}],
              'kernel': ['rbf', 'sigmoid', 'poly',],
              'degree': [2,3,4],
              'decision_function_shape': ['ovr','ovo']
             }
```

At each iteration of the outer loop, a feature set size was defined and the inner loop iterated over the different classifiers optimizing each to its respective parameter space. The classifier and it's corresponding subset of parameter space values to be explored were passed to the GridSearchCV function which iterated over all possible combinations of parameter values and identified the combination that produced the highest score. Classifiers may be optimized for different scores. For this problem it was desirable to find the optimum parameters for maximizing both the average recall and f-scores. Classifiers were optimized for each score separately resulting in two optimized classifiers for each classifier type. For example, an SVC classifier was optimized for recall and another SVC classifier was optimized for f-scores. At each feature set size, the optimization metric defined above was calculated for both recall and f-scores produced by the GridSearchCV optimized classifier. For each metric, f-score and recall, if the optimization score on the current iteration was higher than that of previous iterations, the current score was recorded as the maximum and the corresponding feature set size and classifier parameters were recorded as the optimized values.

The combination of classifier, parameter values and feature set size that produced the highest optimization scores for both recall and f-scores were identified as the optimized classifier and parameters.

Classifiers with optimized parameters and features set sizes were then optimized along train-test split size ranging from using 90% of data to train and 10% of data to test to a 50-50% train test split. The split that produced the largest optimization scores was identified and the scores from that set were saved as the optimized scores for that classifier and the respective recall and f-scores.

Figure 13 below illustrates an example of the first two steps of optimization. The figure shows the recall scores for Survivor and Non-Survivor groups as well as the calculated optimization score for a GridSearchCV optimized Decision Tree Classifier trained and tested with input feature sets ranging in size from 8 to 20. The Tree classifier was optimized at each training set size. It can be seen that the Tree classifier produces its best results (maximum value for Survivor and Non-Survivor Groups while minimizing the difference between them) using 10 input features

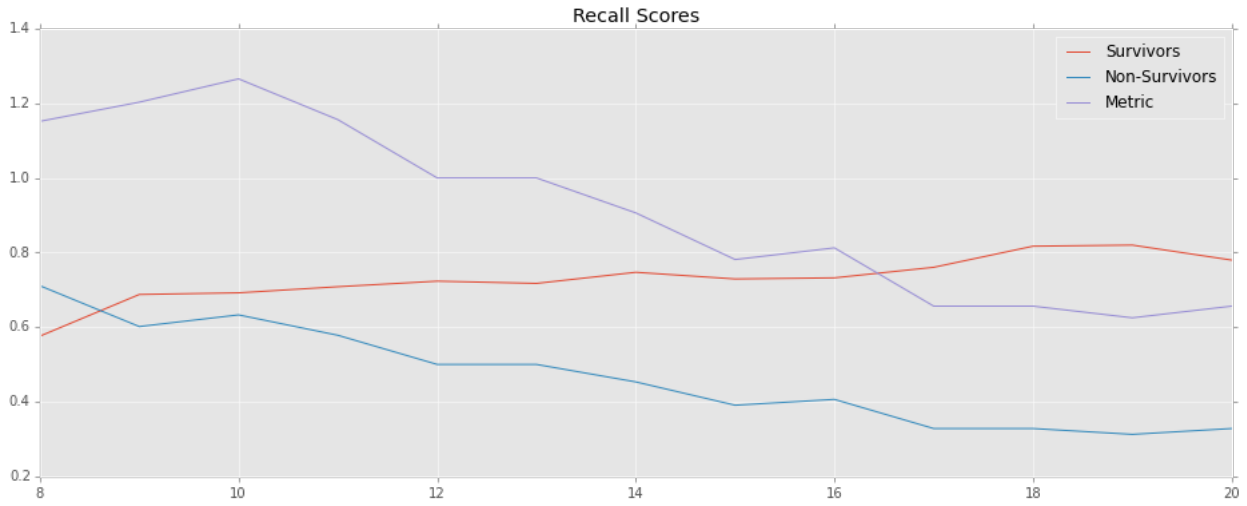


Figure 13: Recall scores for Survivor and Non-Survivor groups as well as the calculated optimization metric achieved using a GridSearchCV optimized Decision Classifier Tree. were plotted against input feature set size. Also plotted was the optimization metric.

The implementation and refinement phases were somewhat challenging in that they involved organizing and keeping track of a number of variables including classifiers, their respective parameter sub-spaces, feature set and train test split sizes, two types of optimization scores and two sets of optimized versions of all of the above, one for f-scores and one for recall.

To help keep things organized liberal use was made of dictionaries. One dictionary was created which contained instances of the candidate classifiers themselves along with their respective parameter subspaces. Another dictionary was used to update and store the optimized versions

of those classifiers, optimized parameter values including feature set and train-test split sizes and their optimized classification scores.

It is worth noting that optimization could have been performed using three nested loops, iterating over feature set size, train/test split and over classifier parameter spaces. During implementation of the first 2 nested loops book-keeping, keeping track of all of the above, was recognized as being ‘a bit involved’ and it was decided to separate the feature set and parameter space optimization from the train/test split optimization.Consolidating these loops is an improvement that may be undertaken in the future.

Despite those challenges, relative to the extensive efforts involved in data collection feature pre-processing and selection phases, implementation was regarded as ‘the easy part’.

Results

Model Evaluation and Validation

For each type of classifier, two instances were created: one optimized for recall and one for F1-scores. The classifiers were optimized using these scores and their respective calculated optimization metrics to maximize the score values for both survivors and non-survivors while minimizing the difference between them thereby avoiding performance imbalances between the classes. The scores achieved for each classifier optimized for recall can be observed in Table 10. The scores for classifiers optimized for f1-scores can be found in Table 11.

		precision	recall	f1-score	support
Classifier	**Classes**				
Kneighbors_recall	**Avg/Total**	0.80	0.83	0.80	1066.0
	Non-Survivors	0.47	0.20	0.28	173.0
	Survivors	0.86	0.96	0.91	893.0
LSVC_recall	**Avg/Total**	0.81	0.68	0.72	1333.0
	Non-Survivors	0.30	0.67	0.42	225.0
	Survivors	0.91	0.68	0.78	1108.0
MLP_recall	**Avg/Total**	0.76	0.78	0.77	1200.0
	Non-Survivors	0.31	0.25	0.28	204.0
	Survivors	0.85	0.88	0.87	996.0
SVC_recall	**Avg/Total**	0.82	0.68	0.72	933.0
	Non-Survivors	0.28	0.69	0.40	147.0
	Survivors	0.92	0.68	0.78	786.0
Tree_recall	**Avg/Total**	0.80	0.69	0.72	1333.0
	Non-Survivors	0.29	0.60	0.40	225.0
	Survivors	0.90	0.71	0.79	1108.0

Table 10: Precision, recall, f1 scores and supports for classifiers optimized for recall.

		precision	recall	f1-score	support
Classifier	**Classes**				
Kneighbors_f1	**Avg/Total**	0.80	0.83	0.80	1066.0
	Non-Survivors	0.47	0.20	0.28	173.0
	Survivors	0.86	0.96	0.91	893.0
LSVC_f1	**Avg/Total**	0.81	0.73	0.76	267.0
	Non-Survivors	0.33	0.59	0.42	44.0
	Survivors	0.90	0.76	0.83	223.0
MLP_f1	**Avg/Total**	0.78	0.81	0.80	933.0
	Non-Survivors	0.35	0.24	0.29	147.0
	Survivors	0.87	0.91	0.89	786.0
SVC_f1	**Avg/Total**	0.81	0.76	0.78	267.0
	Non-Survivors	0.35	0.57	0.43	44.0
	Survivors	0.90	0.79	0.84	223.0
Tree_f1	**Avg/Total**	0.80	0.69	0.72	1333.0
	Non-Survivors	0.29	0.60	0.40	225.0
	Survivors	0.90	0.71	0.79	1108.0

Table 11: Precision, recall, f1 scores and supports for classifiers optimized for f1 score.

The optimization metric scores for all classifiers are shown in Table 12. While LSVC optimized for f1-scores (LSVC_f1) and the SVC optimized for f1 (SVC_f1) and recall (SVC_recall) scores performed well, it was the LSVC optimized for recall (LSVC_recall) that scored at or near the top in both F1 and Recall. In addition, Figure 14 illustrates F1 and recall optimization metric values achieved using all classifiers. Here it can also be seen that LSVC_recall achieved the highest optimization scores in both F1 and recall metrics. LSVC_recall was therefore selected as the final model.

The optimized LSVC classifier parameters, feature set size and train-test split is described below:

Type: Linear Support Vector Classifier (LinearSVC):

Optimized Parameters:

- C = 0.1
- class_weight = {0:1, 1:6}
- loss = 'squared_hinge'
- feature set size = 10
- train-test split = 50/50%

(All parameters not specified here were set to the default values.)

	F1 Metric	Recall Metric
Kneighbors_f1	0.56	0.40
Kneighbors_recall	0.56	0.40
LSVC_f1	0.84	1.18
LSVC_recall	**0.84**	**1.34**
MLP_f1	0.58	0.48
MLP_recall	0.56	0.50
SVC_f1	0.86	1.14
SVC_recall	0.80	1.36
Tree_f1	0.80	1.20
Tree_recall	0.80	1.20

Table 12: F1 and recall metric scores for optimized classifiers.

To evaluate model robustness, 5-fold cross validation was performed using F1 and Recall scores. For each score, cross validation was performed using the average score for all classes (macro) and the score for only non-survivors or, label_positive group. Results for cross validation using F1 as the score can be seen in Table 13 while results of cross validation using Recall can be seen in Table 14. It can be observed that in both metrics, for average and non-survivor scores, the algorithm was robust in producing very consistent scores as can be observed by the relatively low standard deviation values

Cross Validation	F1 Average: Optimized LSVC	F1 Non-Surivors: Optimized LSVC
Run: 0	0.65	0.47
Run: 1	0.58	0.39
Run: 2	0.60	0.44
Run: 3	0.60	0.39
Run: 4	0.63	0.46
F1 Means	0.61	0.43
F1 Stds	0.03	0.04

Table 13. LSVC 5-fold cross validation scores using average and non-survivor F1 scores as metrics.

Cross Validation	Recall Average: Optimized LSVC	Recall Non-Survivors: Optimized LSVC
Run: 0	0.74	0.75
Run: 1	0.67	0.68
Run: 2	0.71	0.77
Run: 3	0.66	0.58
Run: 4	0.73	0.74
Recall Means	0.70	0.71
Recall Stds	0.04	0.08

Table 14. LSVC 5-fold cross validation scores using average and non-survivor Recall scores as metrics.

Justification

Generally speaking, optimization improved classifier performance. By comparing optimization metrics scores shown in Figure 15 below which were achieved using classifiers optimized across their respective parameters spaces, feature set and training testing size spaces, with the same metrics shown in Figure 14 which were achieved using classifiers with default parameters trained on the full feature set (20), we can see that significant gains were achieved through optimization. To better illustrate, Figure 16 shows the benchmark and optimized scores plotted on the same scale, side by side. It can be observed that significant improvement was made, particularly with regard to recall.

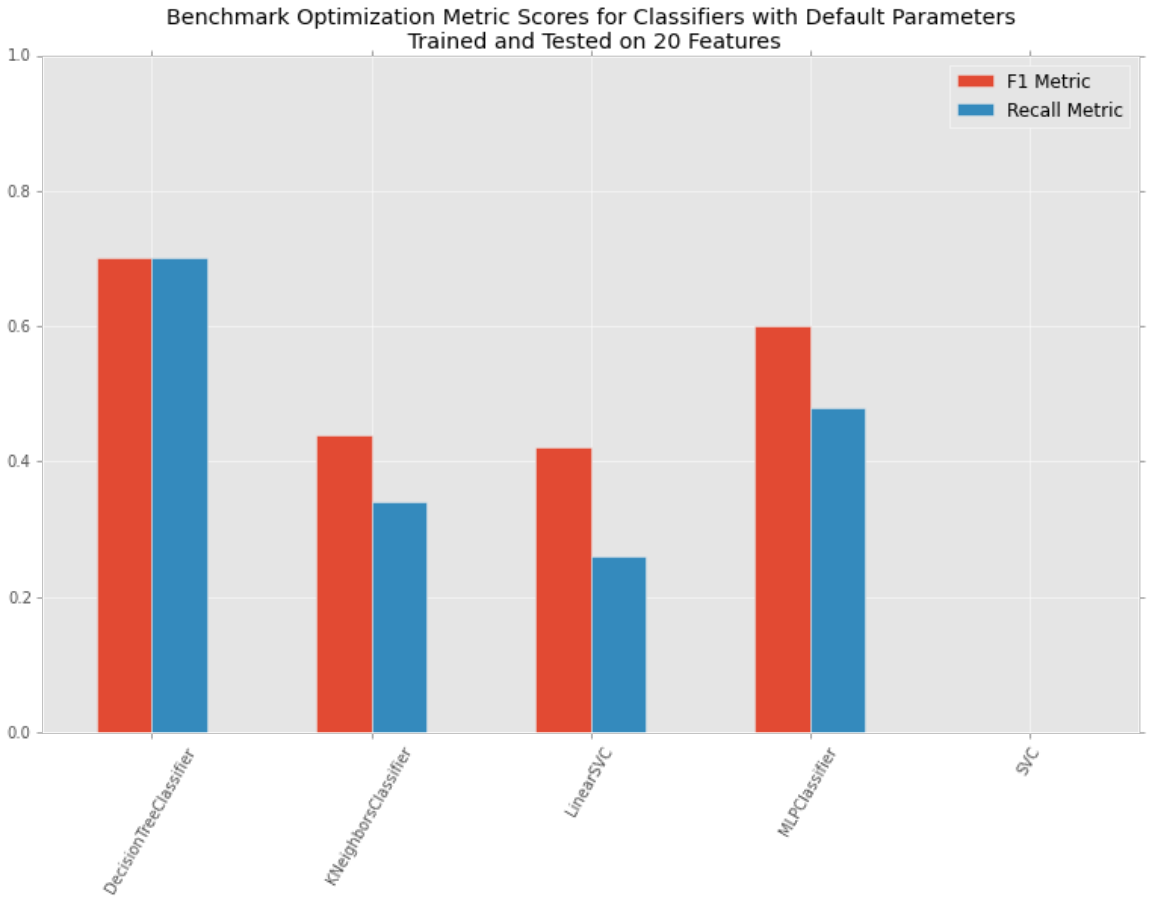


Figure 14. F1 and recall metric scores for classifiers trained and tested using default parameters, 20 features and an 80/20 train/test split.

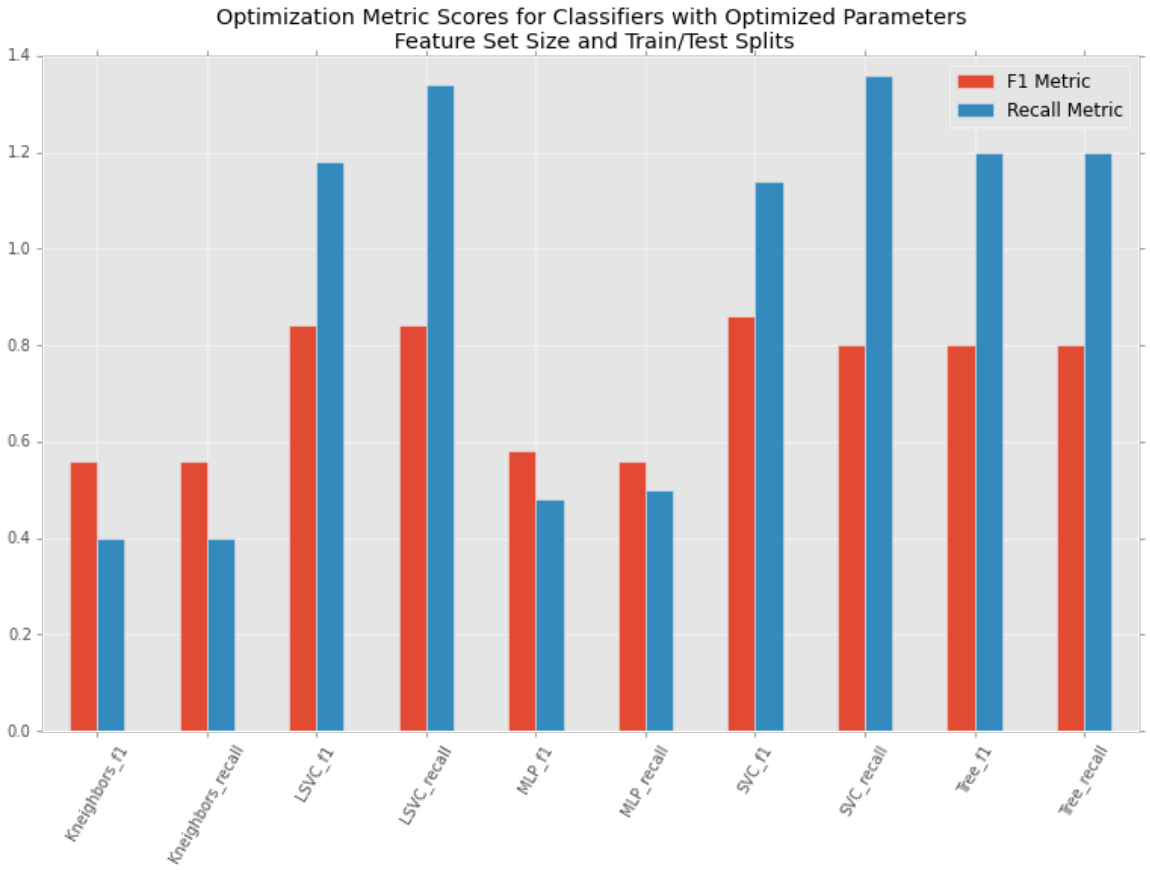


Figure 15. F1 and recall metric scores for optimized classifiers. It can be seen that LSVC_recall has the highest values of both the F1 and Recall metric.

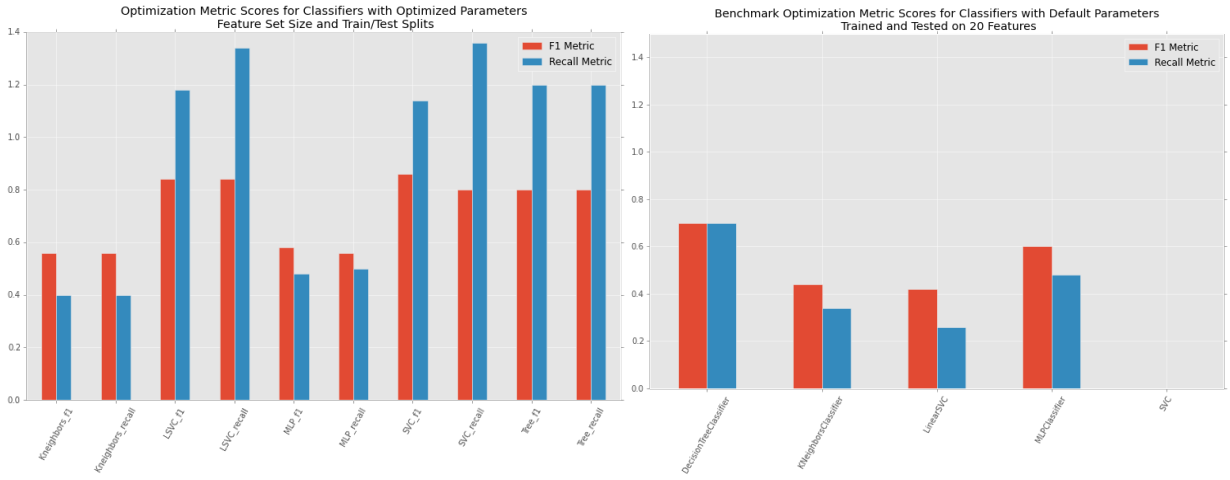


Figure 16. F1 and Recall metric scores achieved using optimized classifiers are shown in the left panel while those achieved using default classifiers are shown on the right. It can be observed that significant improvement was made, particularly with regard to recall.

The final model, the optimized LSVC, was compared with the highest performing default algorithm, the Decision Tree Classifier. Table 15 and Figure 17 illustrate precision, recall and F1 scores achieved using the selected, optimized LSVC compared with those of the highest scoring benchmark algorithm, the Decision Tree Classifier. It can be observed that in areas of recall and F1 score, the optimized classifier trades off survivor scores, representing True Negatives, for non-survivor or, True Positive, performance. The optimized algorithm sacrifices False Positives for False Negatives. As discussed above, the cost of a False Positive may be an increase in the cost of care, treating a patient more aggressively than their condition warrants, while the cost of a False Negative may be in patient mortality, or human lives.

		precision	recall	f1-score	support
Classifier	**Classes**				
LSVC_recall	**Survivors**	0.91	0.68	0.78	1108.0
	Non-Survivors	0.30	0.67	0.42	225.0
	Avg/Total	0.81	0.68	0.72	1333.0
DecisionTreeClassifier	**Survivors**	0.89	0.89	0.89	458.0
	Non-Survivors	0.35	0.35	0.35	75.0
	Avg/Total	0.82	0.82	0.82	533.0

Table 15. Precision, recall, F1 and supports for optimized LSVC and Default Decision Tree Classifier. It can be observed that for the optimized classifier, recall for non-survivors was nearly twice that for the default classifier. This comes at the expense of recall for survivors. Similarly, F1 scores for non-survivors with the optimized classifier were improved over those for the default but at the expense of F1 scores for survivors. Precision was comparable for survivors and non-survivors with both optimized and default

Recall scores for non-survivors move from 0.35, or 35% True Positive predictions , achieved by the default algorithm to 0.67 or, 67% True Positive predictions , achieved using the optimized LSVC algorithm. This improvement is a greater than 30% increase in True Positive predictions which would translate to a 30% increase in critical care patients with the highest likelihood of death receiving more aggressive treatment.

The imbalance between survivor and non-survivor performance in default algorithms was most likely due to the imbalance in the numbers of survivors and non-survivors with there being far greater numbers of survivors. In a population where there is a 10-15% mortality rate, an algorithm that predicted survival for each patient, regardless of the data, will be accurate 85-90% of the time. Improved performance in non-survivor groups was achieved, in part, by weighting those samples more heavily in algorithm training. This was done by assigning the non-survivor group a larger class_weight (class_weight = {0:1, 1:6}). By giving these smaller numbers of samples more weight in training we were able to achieve more balanced performance across the groups.

This can be seen in Figure 17 where we can see that there is less discrepancy between average, survivor and non-survivor F1 scores for the optimized algorithm than for the default where survivors score very well but at the expense of non-survivors. In Recall we see that algorithm performance is very balanced and that there is very little discrepancy at all between average, survivor and non-survivor scores for the optimized algorithm while large differences exist between scores for those groups achieved using the default algorithm.

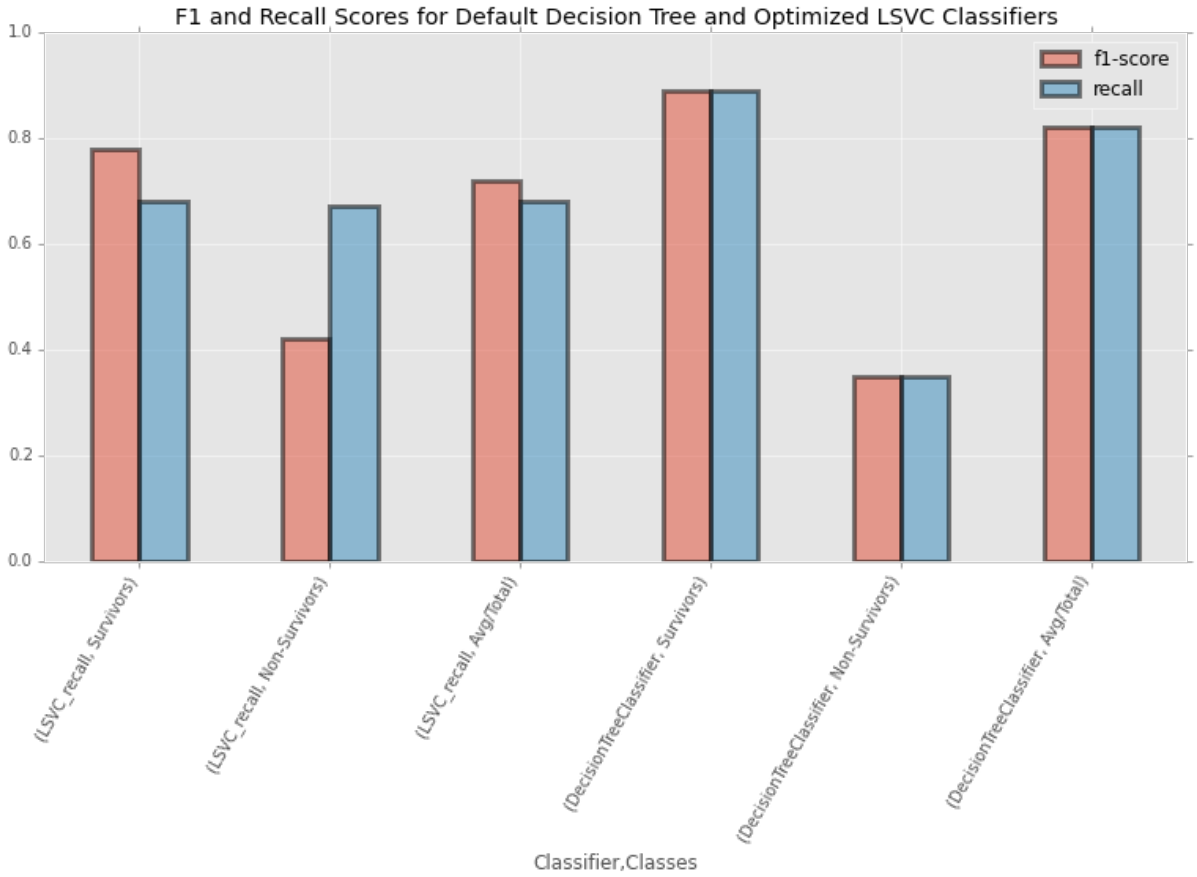


Figure 17: F1 and Recall scores for Default Decision Tree Classifier and Optimized LSVC.

Cross validation results for the optimized LSVC and the default Decision Tree Classifier are compared in Figures 18 and 19. Figure 18 shows that F1 performance was somewhat improved in the optimized group while Figure 19 shows that the optimized classifier was able to identify far more non-survivors than the default algorithm. F1 performance is a measure of precision and recall. Improving recall for non-survivors involves a tradeoff in precision. The result is that we see large gains in recall, particularly for non-survivors, with only modest gains in F1 performance.

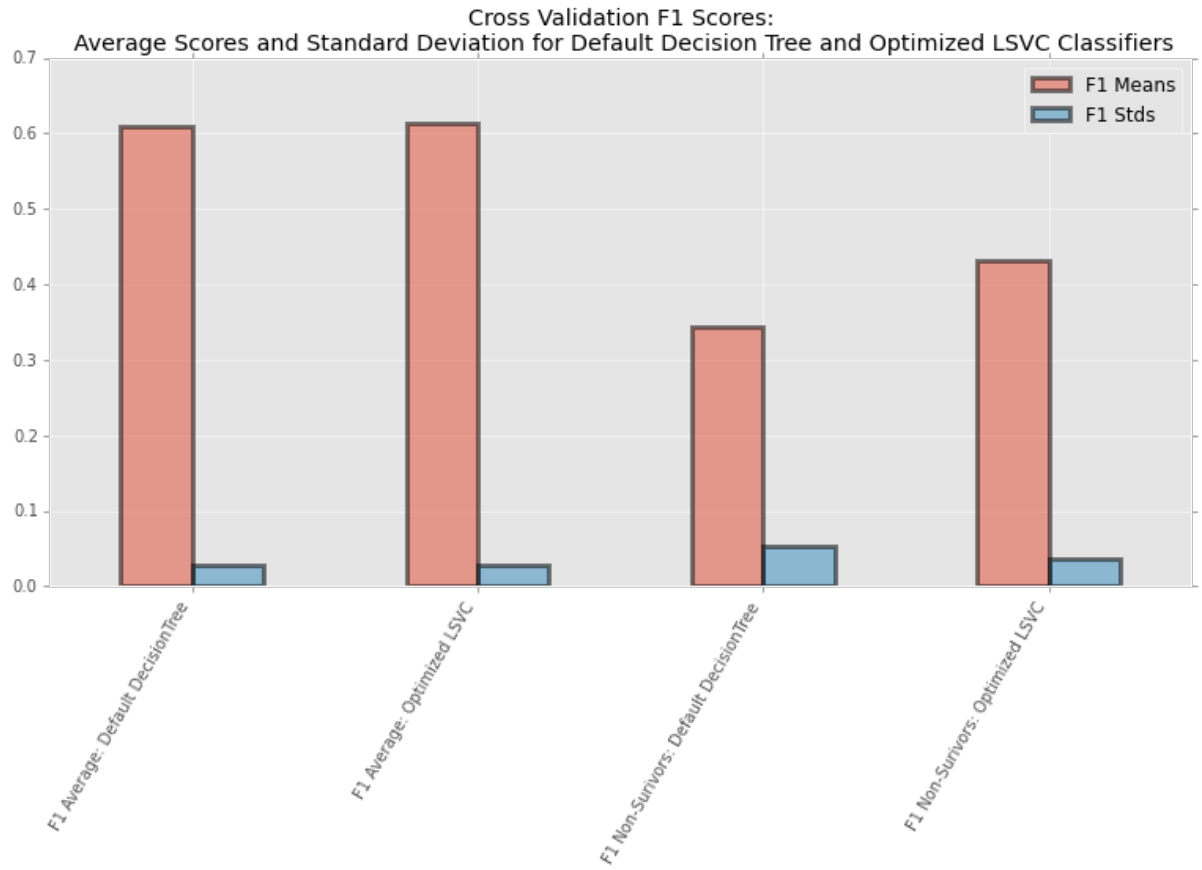


Figure 18. Average and Standard Deviation for Cross Validation F1 Scores (n = 5) for Recall-Optimized LSVC. It can be observed that average F1 scores were very similar between default and optimized classifiers but that, in the area of Non-Survivors, the optimized classifier performed somewhat better than the default.

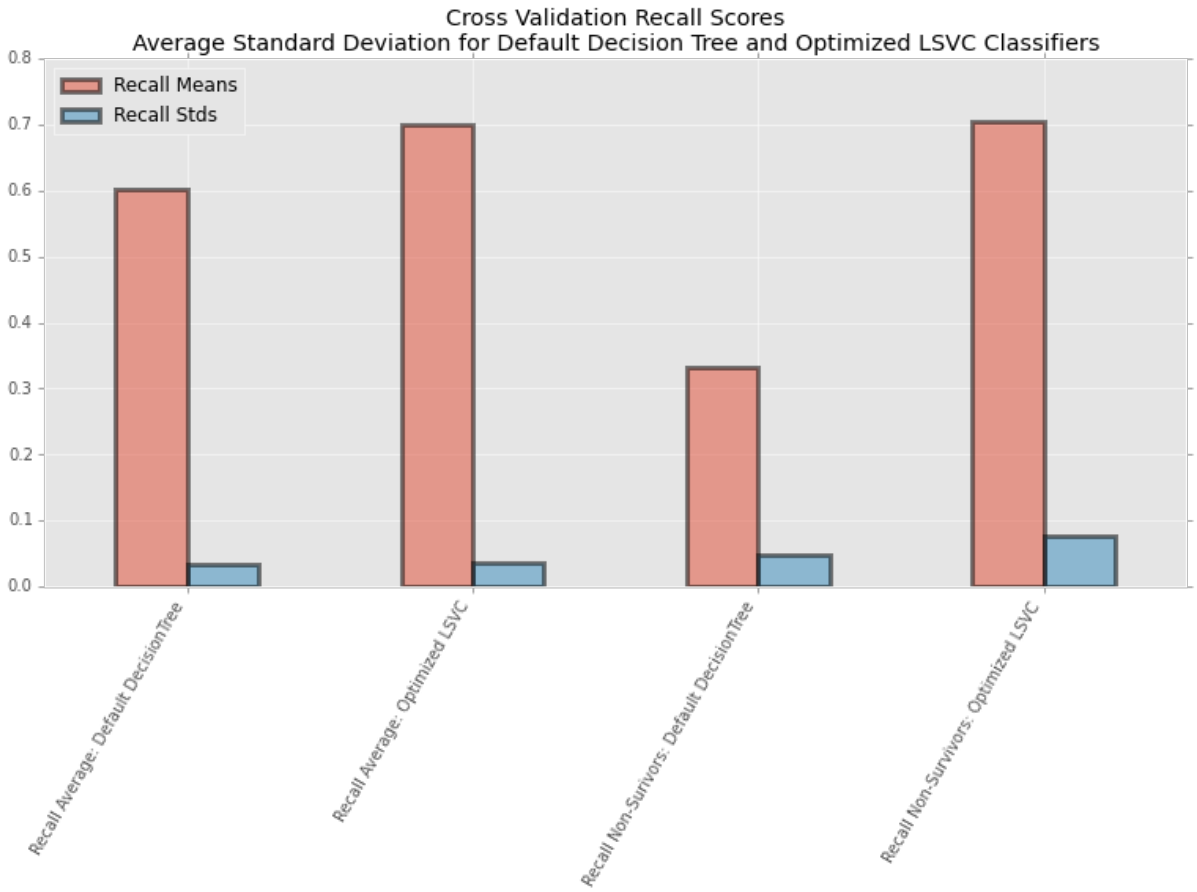


Figure 19. Average and Standard Deviation for Cross Validation Recall Scores (n = 5) for Recall-Optimized LSVC. It can be observed that average Recall scores improved with the optimized classifier. Improvements were even more pronounced when looking at recall specifically for non-survivors.

Conclusion

Free Form Visualization

Above we compared the top performing optimized and top performing default classifiers. To more clearly illustrate the effect of optimization, Figure 20 below shows recall and f1 average, survivor and non-survivor scores for LSVC algorithms with optimized and default parameters. It can be easily seen that average and survivor F1 scores for the optimized classifier were lower than those for the default algorithm but the optimized F1 score for the non-survivor group was much higher. Similarly,while average and survivor recall scores for the optimized classifier were slightly lower than those for the default classifier, the optimized non-survivor recall score was much higher than that for the default classifier. In addition, recall scores for the optimized classifier were very nearly equal across average, survivor, non-survivor roups. Given that the goal of optimization was to maximize both F1 and recall for both survivor and non-survivor groups while minimizing the difference between them, this result represents a significant success.

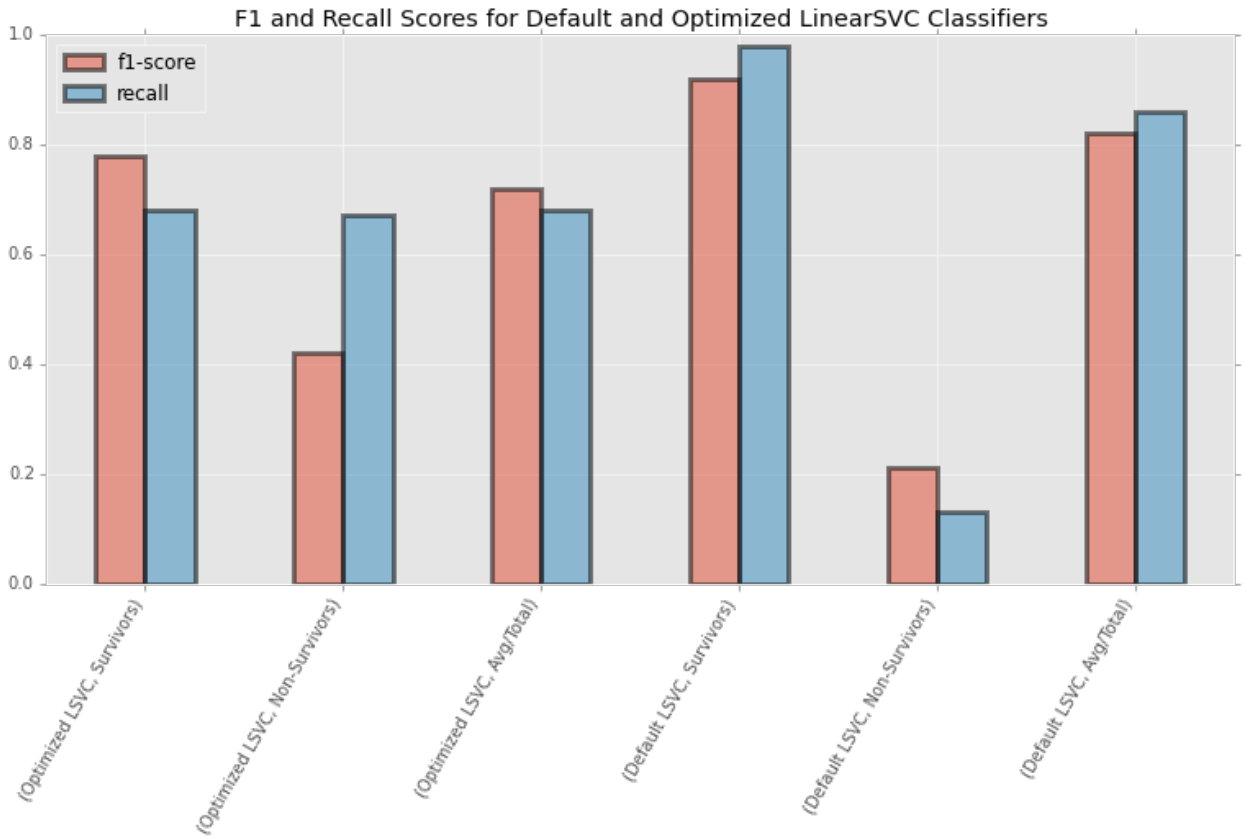


Figure 20: Recall and F1 average, survivor and non-survivor scores for the LSVC with optimized and default parameters.

Reflection

The initial problem was: can we predict mortality rates based on data collected in the first 24hrs of a patient's ICU stay. The first challenge for someone not entirely familiar with PostgreSQLwas to build a local copy of the database and perform a query.

Once the database was established and could be queried, the next, monumental challenge is deciding what data we wanted to extract and use as features. There were innumerable possible features that could be used as input to a classifier. Some data, like patient demographic information, was easier to extract. This data was constant meaning that there was a single value that did not change during the course of the hospital stay. The features included things like patient age, ethnicity, marital status etc. Initial data measurements of things like blood pressure could also be seen as constant in that we extract one data point from the database for use as a feature. Beyond the constant features there is time series data. These are measurements taken over the course of the first 24hrs of the ICU stay. An example of this type is blood pressure measurements. This data must be extracted as a time series, then reduced to features consisting of a single value. This reduction of dimensionality was achieved using statistics like mean, median, slope, standard deviation etc. This data is more challenging to use in that it requires a more sophisticated query and more pre-processing.
Beyond the types of features, there are the features themselves. It would be extremely difficult, if not impossible, to extract all the possible features so we need to identify variables which we believe may be relevant to the outcomes. A brief literature search and my background in biomedical engineering, healthcare and medical device development provided me with a sufficient degree of domain knowledge to make some initial choices.
This process was iterative in that initially, simpler, more general features were selected, pre-processed and used as inputs. If the selected features were insufficient to predict mortality in any meaningful way, we went back to the database, performed a more broad query and repeated the process. This process was repeated a number of times until classifier performance was deemed sufficient.
As the number of features increased, a new problem presented itself:

1. Pre-processing and classification steps require there to be no missing data points
2. There were very few, if any ICU stays in which all features were present.

Not knowing which features would be selected or identified as significantly correlated with the outcomes, the decision was made to break the features up into affinity groups or, groups of features more likely to be present together. Once broken up into groups or blocks, pre-processing was performed on all blocks including the feature selection step. The challenge then became, to identify features that were sufficiently correlated with the outcome and that, when combined with other features would not result in too few useable ICU stays. Fortunately the top-ranked features when combined resulted in over two thousand useable ICU stays.

To generate a baseline of classification performance, the algorithms selected for evaluation were then trained and tested on the full, 20 feature data set. Performance in terms of precision, F1 and recall scores was noted.

In previous iterations and in benchmark testing it was recognized that, because there were a smaller number of non-survivors than survivors (perhaps 10-15% of patients were non-survivors) algorithms would tend to predict that a patient was a survivor and generate false negatives. The goal was to improve performance in the area of non-survivor prediction. We wanted to improve performance in True Positives and bring it to the level of True Negative classification performance. To accomplish this the F1 and recall scores were chosen and, from those scores, optimization metrics were developed to maximize performance in both survivor and non-survivor groups while minimizing the difference between the two.

The next step was optimizing the selected algorithms over their respective parameter sub-spaces, over the feature set size and over a range of train-test splits. The way this was accomplished was that algorithms were first optimized over their parameters subspaces for a given feature set size, this was then done over the range of feature set sizes. Once the optimum parameter and feature set sizes were determined, the algorithms with optimized parameters and feature set sizes were then optimized over a train-test splits ranging from 10% test/90% train to 50/50%.
The fully optimized algorithms were then evaluate to determine which of them performed best in the areas of F1 and recall.

The performance of the final model was then compared to the performance of the best default, benchmark algorithm to assess improvements.

Improvement

Improvements may be achieved through an even more extensive feature selection search. While the feature extraction, pre-processing and selection process performed here was extensive, given the large amounts of data available, it was not completely exhaustive. It is possible that there may be features with more predictive power than the ones finally selected here.
Improvement might also be made in determining affinity groups. It occurred to me that the problem might be more efficiently addressed by defining affinity groups in an algorithmic way based on additional statistics.
Lastly, there may be some improvement in classification by utilizing ensemble classifiers. Ensemble methods may involve bagging or boosting methods or may involve combining the output of multiple classifiers and performing a simple function like averaging. Yet another ensemble method may involve feeding the output of multiple classifiers as input into an additional classifier like a multi-layer perceptron. Alternatively, different classifiers may be used sequentially.