Capstone Project — Alfred Geoffrey Lyle

Machine Learning Nanodegree Project — October 2, 2017

Demographic Features Leading to Increased Risk for Patient Mortality

Definition

Project Overview

In 2010 there were 715,000 inpatient hospital deaths in the United States[i]. This accounts for 28.6% of deaths in the United States. Studies show that increased patient age and length of stay contribute to patient mortality[i]. It has also been shown that females have significantly higher hospitality mortality rates than males[ii]. However, when it comes to other patient features such as race/ethnicity, there is no consensus on whether they affect patient mortality.

The 2010 census shows the United States population is growing older as those aged 45 and over increased by 5% in 10 years[iii]. As the population grows older, it can be assumed the number of hospital visits will increase. Thus it will be useful for hospitals to continue to determine which factors lead to increased mortality.

As hospitals transition from pen and paper records to electronic medical records, the concern of the security of patient medical data becomes more relevant. In 2016 there were 374 security breaches in the Health/Medical industry, a 36% increase from 2015[iv]. While data being collected for research purposes is disidentified according to rules and regulations, the data stored in medical records contain much information that can be used to identify and target certain individuals. The challenge is to see what information is useful for hospitals and machine learning algorithms in predicting patient mortality.

Problem Statement

The goal of this project is to use hospital Intensive Care Unit (ICU) data to:

1. Confirm patient age and length of stay can be used to determine patient mortality.
2. Determine if adding features ‘gender’ and ‘insurance’ type will improve the ability to predict patient mortality.

3. See if adding optional features such as ‘ethnicity’, ‘language’, ‘religion’ and ‘marital status’ improve predicting patient mortality.

This will be accomplished by building models using different patient features. These models will be scored to ensure they do better than blind guessing, and then ranked against each other to see which one performs the best at predicting patient mortality.

Metrics

When comparing the classification models, I will be using the Area Under Curve (AUC) metric. The AUC score is the area under the ROC curve, which is a curve plotting the true positive rate against the false positive rate for every possible probability threshold. Basically a higher AUC score shows the learning algorithm is better at distinguishing the difference between two groups. The highest AUC score is 1.0 where the two groups are perfectly distinct, and the lowest score is 0.5 where the algorithm is predicting about as well as random chance. As I will be comparing two groups, patients who lived and patients who died during their ICU stay, this metric can be used.

Furthermore, when looking at the full data set, 15737 patients died out of a total 46476 patients. If I used accuracy as the metric to compare the models, a benchmark model that simply predicted all patients survived would have an accuracy score of 67%. As the data is imbalanced, with more patients surviving than expiring, it is more useful to compare the models using an AUC scorer which is insensitive to imbalanced data.

Analysis

Data Exploration

The dataset used in this project is the MIMIC-III (**M**edical **I**nformation **M**art for **I**ntensive **C**are III) database[v]. This database contains deidentified medical data for over forty-thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. Requesting access to this database can be done at https://mimic.physionet.org/gettingstarted/access/. To be granted access to the database, it is necessary to complete the CITI “Data or Specimens Only Research” course.

The MIMIC-III data is contained within different tables in a PostgreSQL database. For this study patient information will be obtained from the ADMISSIONS, PATIENTS, and ICUSTAYS tables. Patient data can be crosslinked between tables using SUBJECT_ID and HADM_ID. Each hospital admission has a unique HADM_ID and each patient has a unique SUBJECT_ID. It is possible for tables to have duplicate SUBJECT_ID, indicating the patient had multiple hospital admissions. From the ADMISSIONS table I will be selecting the SUBJECT_ID, HADM_ID, ADMITTIME,

DISCHTIME, DEATHTIME, INSURANCE, LANGUAGE, RELIGION, MARITAL_STATUS, and ETHNICITY columns. From the PATIENTS table I will be selecting the SUBJECT_ID, GENDER, DOB (Date of Birth), DOD (Date of Death), and EXPIRE_FLAG (binary: 1 for expired, 0 for not expired) columns. From the ICUSTAYS table I will be selecting the SUBJECT_ID, HADM_ID, and LOS (Length of Stay) columns.

I will be using the EXPIRE_FLAG feature as the dependent variable the models will predict. The independent variables will be the LOS, GENDER, INSURANCE, LANGUAGE, RELIGION, MARITAL_STATUS, and ETHNICITY. LOS, GENDER, and INSURANCE have values for each patient and are considered non-optional features. LANGUAGE, RELIGION, MARITAL_STATUS and ETHNICITY all have missing values (NaN, 'UNOBTAINABLE', 'UNKNOWN (DEFAULT)') and are considered optional features. Records that are missing optional feature values will be dropped when models are being built using these features. Also, outliers are accounted for by grouping patients by age and LOS ranges that were previously analyzed by the CDC (Center for Disease Control and Prevention).

In this data, there is no patient age data. Furthermore, for patients aged 89 years and older, the patient DOB (Date of Birth) was shifted 300 years in order to protect patient confidentiality.

Exploratory Visualization

When exploring the data it is important to determine the distribution of the target variable, patient ICU survival. When looking at patient survival rates, the MIMIC-III data has similar rates compared to those found in previously published studies. Figure 1-1 shows the female mortality rate is slightly higher than the male mortality rate. Interestingly, patient mortality increases in patients aged 65 years and older; however, patient mortality is lower in those 85 years and older compared to patients aged 75-84 years (Figure 1-2). Also of note is the great disparity (~30%) in mortality rates based on insurance type (Figure 1-3). Initial observations of the data show there are differences among the groups of patients. Therefore, it is likely that a supervised learning algorithm will be able to create a decision boundary that will provide predictive power in determining which patients have a higher mortality risk.

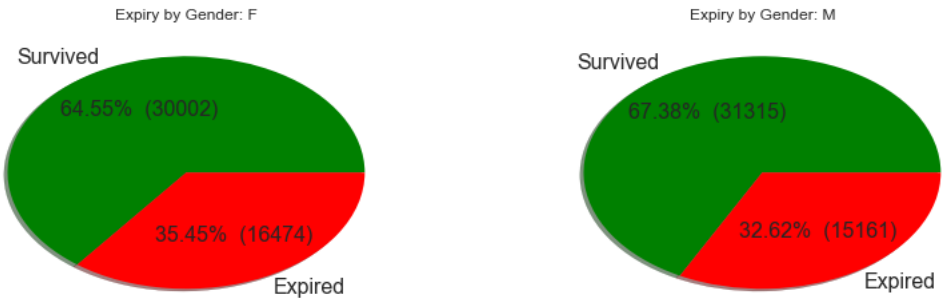
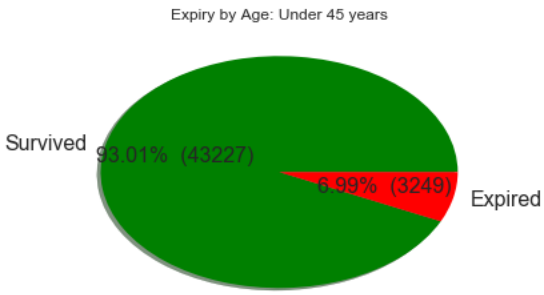
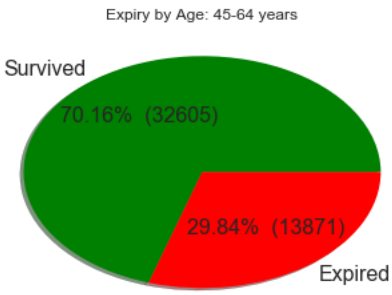
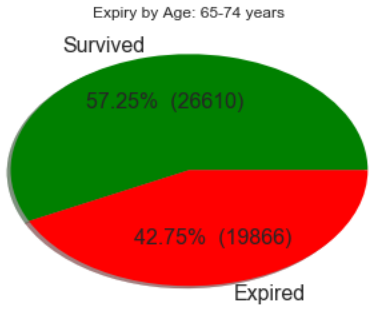
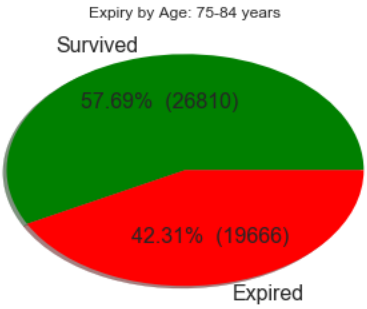


Figure 1-1: Mortality rates by gender









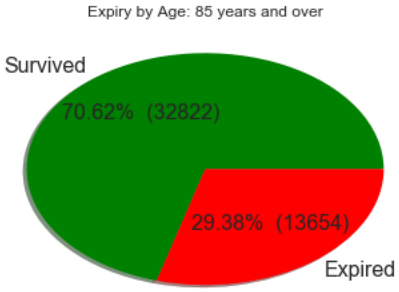


Figure 1-2: Mortality rates by age

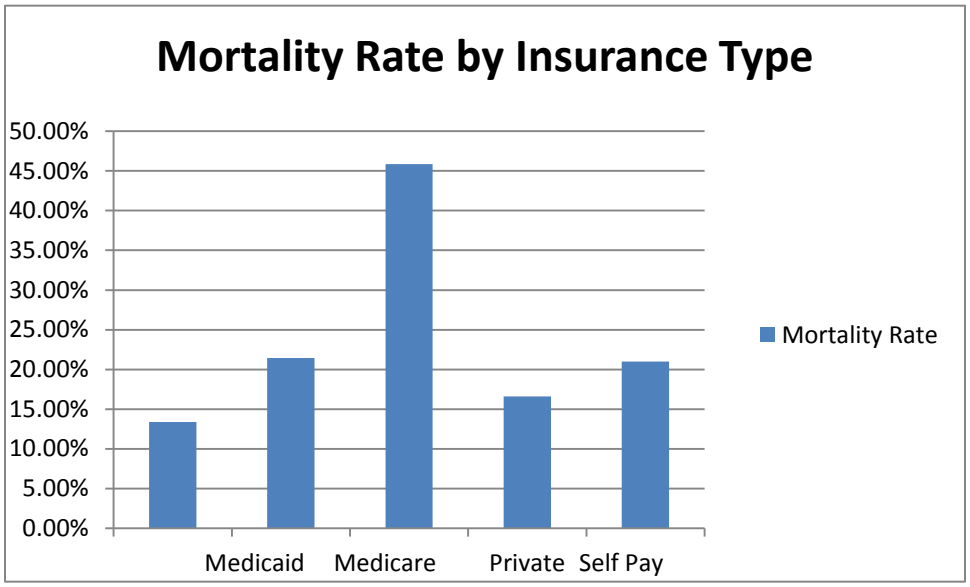


Figure 1-3: Mortality rates by insurance

Algorithms and Techniques

There were a variety of supervised learning algorithms used classify the data. As I did not know which algorithm would be best at classifying the data, I used GaussianNB, DecisionTreeClassifier, AdaBoostClassifier, SVM and RandomForestClassifier. These models are compared against each other using the AUC scorer. These algorithms were made available through the Sci-Kit Learn module[vi].

A Gaussian Naïve-Bayes classifier (GaussianNB) works quickly with large quantities of data, treating each feature independently from the others. This works well with email spam filters. This could work well with hospital data as there are many patient records with many possible features to be analyzed.

A Decision Tree (DecisionTreeClassifier) is often used for data mining and analytics. They are prone to overfitting it the minimum number of samples is not properly tuned. This is a good potential classifier as there is a large amount of data (many samples for each split in the tree) and the end goal is to create different groups based on increased mortality.

Support Vector Machines (SVMs) work well on problems with many features. SVMs goal is to maximize the space among points on the boundary line. As I will be using an AUC scorer, maximizing the difference between groups will be essential for an algorithm to score highly.

AdaBoostClassifier and RandomForestClassifier are both ensemble algorithms. They create a variety of weak models and give them different weights based on how they perform. These

models are added together to create the final model. These should be useful in creating accurate models requiring little fine tuning after the data is correctly processed.

Benchmark Model

The benchmark model is created from using the two features the CDC agree affect patient mortality, patient age and LOS. The model will be the algorithm with the highest AUC score. A similar study on predicting 30 day hospital readmissions was hosted on Kaggle. This competition contained data including patient age, gender, and length of stay. The top team created a model with an AUC score of 0.68469 and the 10th place team had an AUC score of 0.64823[vii]. The benchmark model should have an AUC score in this range.

Methodology

Data Preprocessing

I imported the Psycopg2 module to retrieve the data from the PostgreSQL database. Then a cursor was created to access the target database and select the needed columns from it. All rows were fetched and put into a Pandas DataFrame.

```
#Import necessary modules

import psycopg2
import psycopg2.extras

#Access relevant fields from ADMISSIONS table
curr2 = conn.cursor()
try:
    curr2.execute("SELECT SUBJECT_ID, HADM_ID, ADMITTIME, DISCHTIME, DEATHTIME,
INSURANCE, LANGUAGE, RELIGION, MARITAL_STATUS, ETHNICITY from ADMISSIONS")
except:
    print "I cannot SELECT from ADMISSIONS"

#Get column names for Pandas DataFrame
admcolnames = [desc[0] for desc in curr2.description]
#Create Pandas DataFrame
dfadm = pd.DataFrame(curr2.fetchall(), columns=admcolnames)
print dfadm.head() #Get data sample
print len(dfadm.index)
curr2.close()
```

As there was no patient age category, the patient age was created by subtracting the patient's DOB from their admission time. Thus patients were grouped according to the CDC groups in their study. Patient LOS (Length of Stay) was also grouped similar to the CDC groups. Grouping these numerical values this way also helps us deal with any outliers in the data.

The data from the different tables were merged together on the SUBJECT_ID column. Duplicates in the HADM_ID and SUBJECT_ID columns were dropped. Furthermore, rows with missing, 'UNOBTAINABLE', 'UNKNOWN (DEFAULT)' or NaN values were dropped. Also, features

with less than 10 patient records were dropped to ensure there was enough data for the model to train on.

One-hot encoding was used to transform the non-numerical features into dummy variables.

Implementation

The implementation process was carried out in three phases:

1. Training and Testing Phase
2. Evaluation Phase
3. Data Gathering Phase

In the training and testing phase, a classifier was fit to the data, and then ran against test data. The AUC score and accuracy were input into a dictionary with the classifier name as the key.

```
def train_results(learner, X_train, y_train, X_test, y_test):
    '''
    inputs:
        -learner: type of learning algorithm to be compared
        -X_train: features training set
        -y_train: expiry training set
        -X_test: features test set
        -y_test: expiry training set
    '''

    results = {}

    #Fit the learner to the training data
    learner = learner.fit(X_train, y_train)

    #Get the predictions on the test set,
    predictions_test = learner.predict(X_test)

    #Compute accuracy on test set
    results['acc_test'] = accuracy_score(y_test, predictions_test)

    #Compute AUC score on the test set
    results['auc_test'] = roc_auc_score(y_test, predictions_test)

    # Return the results
    return results
```

To ensure the AUC score would be valid over multiple trials, the classifiers were trained and tested multiple times with the data randomly split. The AUC scores were then averaged and the classifier with the highest average AUC score was identified.

```
def best_classifier(features, expiry, trials, classifiers, verbose=True):
    '''''
        -features: A dataframe containing the features for training
        -expiry: Target variable, binary patient death or survival
        -trials:  An integer denoting the number of random iterations
        -classifiers:  A list of supervised learning classifiers
```

```
    '''

    #Run through multiple times using random states
    results = {}
    for clf_name in classifiers:
        results[clf_name] = {}
    for trial in range(trials):
        state = random.randint(0,429496729)
        #print "Trial {}: Random State is {}".format(trial+1, state)

        #Create test and train samples
        X_train, X_test, y_train, y_test = train_test_split(features, expiry,
                                                            random_state=random.randint(0,429496729))

        #  Initialize the models
        clf_A = GaussianNB()
        clf_B = tree.DecisionTreeClassifier(random_state=state)
        clf_C = RandomForestClassifier(random_state =state)
        clf_D = AdaBoostClassifier(random_state = state)
        #clf_E = SVC(random_state = state)

        # Collect results on the learners
        for clf in [clf_A, clf_B, clf_C, clf_D]:
            clf_name = clf.__class__.__name__
            results[clf_name][trial] = \
            train_results(clf, X_train, y_train, X_test, y_test)

    #Find average AUC scores
    average_scores = {'accuracy': {}, 'auc': {}}
    #print results
    for clf in results:
        auc_scores = 0
        accuracy_scores = 0
        for trial in results[clf]:
            auc_scores+= results[clf][trial]['auc_test']
            accuracy_scores += results[clf][trial]['acc_test']
        average_scores['auc'][clf] = auc_scores/trials
        average_scores['accuracy'][clf] = accuracy_scores/trials

    #Print classifier with max AUC score
    max_auc_clf = max(average_scores['auc'].iterkeys(), key=(lambda key: average_scores['auc']
[key]))
    max_accuracy_clf = max(average_scores['accuracy'].iterkeys(), key=(lambda key: average_sco
res['accuracy'][key]))
    if verbose:
        print "{} has highest AUC test score of {:.6f}".format(max_auc_clf, average_scores['au
c'][max_auc_clf])
        print "{} has highest accuracy score of {:.6f}".format(max_accuracy_clf, average_score
s['accuracy'][max_accuracy_clf])

    #Return highest scores to be stored in dictionary
    return [max_auc_clf, average_scores['auc'][max_auc_clf]]
```

With the classifier with the highest AUC score identified. It was then necessary to evaluate the classifiers using different feature sets. A function was created to run quickly through the data and evaluate how the classifiers scored using different feature combinations.

```
def classifier_for_features(df, features, trials):
    '''''
    df - The Pandas DataFrame with the data for training and testing
    features - A list of column features the classifier will use for training
    trials - An integer for the number of random trials the best classifier function will run
    '''

    #Preprocess feature data
    input_features_raw = df[features]
    input_features = pd.get_dummies(input_features_raw)

    #Get target variable
    expiry = df['expire_flag']

    #Assign dictionary name to features
    features_key = ' '.join(features)
    if df.name not in best_classifiers:
        best_classifiers[df.name] = {features_key: []}
    else:
        if features_key not in best_classifiers[df.name]:
            best_classifiers[df.name][features_key] = []

    #Add features to list for future data collection
    if features not in previous_features:
        previous_features.append(features)


    #Run best_classifier function
    best_classifiers[df.name][features_key] = \
    best_classifiers[df.name][features_key] + best_classifier(input_features, expiry, trials,
clf_name_list)
```

As part of the evaluation of non-optional features, I used Sci-Kit Learn's SelectKBest function. This allowed me to select the top features that provided the most information gain in finding the differences between the patient groups.

```
def get_best_features(features, expiry, num_k):
    #Randomly split data
    X_train, X_test, y_train, y_test = train_test_split(features, expiry,
                                                        random_state=random.randint(0,429496729))
    #Select K best using Chi Squared
    select_k_best_classifier = SelectKBest(chi2, k=num_k).fit(X_train, y_train)
    transformed_features = SelectKBest(chi2, k=num_k).fit_transform(X_train, y_train)

    #Get list of K best feature names
    mask = select_k_best_classifier.get_support() #list of booleans
    new_features = [] # The list of your K best features

    for bool, feature in zip(mask, features.columns.values.tolist()):
        if bool:
            new_features.append(feature)

    #Return list of column names containing best features
    return new_features
```

Refinement

An initial evaluation of the classifiers by score and time found:

classifier	AUC score	train/predict time
GaussianNB	0.705251	0.019
AdaBoostClassifier	0.685452	0.807
SVC	0.675778	50.163
DecisionTreeClassifier	0.684158	0.028
RandomForestClassifier	0.685077	0.177

The SVC (SVM) added ~50 seconds per trial without improving the AUC score. As each set of features was run 10 times in order to reduce the randomness of the AUC score, this would add 5 minutes per feature set and approximately 1 hour to run the entire program. For this reason I stop using the SVC as a potential classifier

Results

Model Evaluation and Validation

The dfmimic DataFrame, which includes all 46476 patient records, gives the following scores for non-optional feature combinations.

dfmimic		
age los gender insurance	GaussianNB	0.713708
age los insurance	GaussianNB	0.711753
age los	GaussianNB	0.697432
age los gender	GaussianNB	0.694092

Table 1-1: dfmimic DataFrame AUC scores

This table proves the initial assumption that patient age and LOS affect patient mortality. Of greater interest is how the non-optional features improve the AUC score. When the gender features are added to the classifier, it performs worse in predicting patient mortality. However, gender combined with the insurance features performs better than the baseline model and the best of the four combinations. This shows that using all non-optional features should aid hospitals in predicting patients with higher mortality risk.

dfmimic		
age gender insurance	GaussianNB	0.717072
age insurance	GaussianNB	0.713349
age gender	GaussianNB	0.689353

Table 1-2: dfmimic DataFrame AUC scores with 'los' removed

When removing the LOS, it appears that the AUC scores do not change greatly. Thus, while it could be useful to know how long a patient has been in the ICU and tailoring care to this patient accordingly, it is not a necessary feature for predicting patient mortality.

The top classifiers and AUC scores for each DataFrame/feature combination is represented in Table 1-1. The scores are presented in order from highest to lowest. DataFrames are named by which data subsets have been cleaned of NaN and missing values (e=ethnicity, r=religion, ms=martial_status, l=language; ex. df_er uses 'ethnicity' and 'religion' features). When adding the optional features, the top K best features are broken down by group.

DataFrame	features								classifier	AUC_score	num_features
dfmimic	age los insurance								GaussianNB	0.714987	
	age	los	insurance	gender	e	r	l	ms			
df_ethnicity	4	1	4	0	0	0	0	0	GaussianNB	0.714061	9
df_religion	3	2	3	0	0	2	0	0	GaussianNB	0.698769	10
df_er	4	2	2	0	0	2	0	0	GaussianNB	0.696312	10
df_marital_status	4	2	2	0	0	0	0	2	GaussianNB	0.665261	10
df_msr	4	2	2	0	0	2	0	2	GaussianNB	0.662162	12
df_ems	4	2	2	0	0	0	0	2	GaussianNB	0.661756	10
df_language	4	2	3	0	0	0	1	0	GaussianNB	0.657046	10
df_erms	4	2	2	0	0	2	0	2	GaussianNB	0.655970	12
df_el	4	2	2	0	0	0	0	0	GaussianNB	0.655135	8
df_rl	4	2	2	0	0	0	0	0	GaussianNB	0.654457	8
df_elms	4	2	2	0	0	0	0	2	GaussianNB	0.650383	10
df_msl	4	1	2	0	0	0	0	1	GaussianNB	0.645357	8
df_msrl	4	2	2	0	0	0	0	2	GaussianNB	0.642188	10
df_emsrl	4	2	2	0	0	0	0	2	GaussianNB	0.641252	10

Table 1-3: Best AUC score for DataFrame by column features

From the table I can see the top classifiers for each dataframe use a common set of features. All use 'age', 'los', and 'insurance' and none of them use 'gender' or 'ethnicity'. Also, the top two classifiers, both had AUC scores of over 0.71, do not use any non-optional features. This suggests the non-optional features are far more useful in predicting patient mortality than the optional features.

To validate the best model (AUC score = 0.7141, Dataframe is df_ethnicity, contains 4 age features, 1 los feature, and 4 insurance features) I ran the classifier multiple times and output the AUC score. As the classifier randomly groups patients into training and testing samples, and

the starting position of the classifier is randomized each time, if the AUC score is significantly different from the multiple trials it would show this model is not reliable. The results from 10 validation trials are as show in Table 1-4.

Trial 0	0.710394	Trial 0	0.711719
Trial 1	0.7135	Trial 1	0.710449
Trial 2	0.71188	Trial 2	0.712135
Trial 3	0.712016	Trial 3	0.707905
Trial 4	0.711366	Trial 4	0.709989
Trial 5	0.713064	Trial 5	0.709952
Trial 6	0.71142	Trial 6	0.709937
Trial 7	0.710817	Trial 7	0.710866
Trial 8	0.710837	Trial 8	0.713994
Trial 9	0.712638	Trial 9	0.711606
Range	0.003106	Range	0.006089
SD	0.001024	SD	0.001636
Mean	0.711793	Mean	0.710855

Table 1-4: df_ethnicity validation AUC scores

Performing a one sample t-test for significance gives a two-tailed P-value of less than 0.0001, which is quite significant. This suggests the AUC score initially found by the best model is an outlier. Therefore I ran an unpaired t-test between the two validation results which gave a two-tailed P-value of 0.1418 which is considered not statistically significant. This suggests the model is reliable with an AUC score of around 0.711, which still makes it the best model of the group.

Justification

In order to show this model is significantly better than the baseline model (uses the dfmimic Dataframe with all patient records included; column features used are 'age' and 'los'), I ran the validation function on the baseline model (results in Table 1-5) and performed another a one sample t-test.

Trial 0	0.700032
Trial 1	0.69882
Trial 2	0.698226
Trial 3	0.698795
Trial 4	0.701098
Trial 5	0.701346
Trial 6	0.69412
Trial 7	0.698147
Trial 8	0.700098

Trial 9	0.699336
Range	0.007226
SD	0.002041
Mean	0.699002

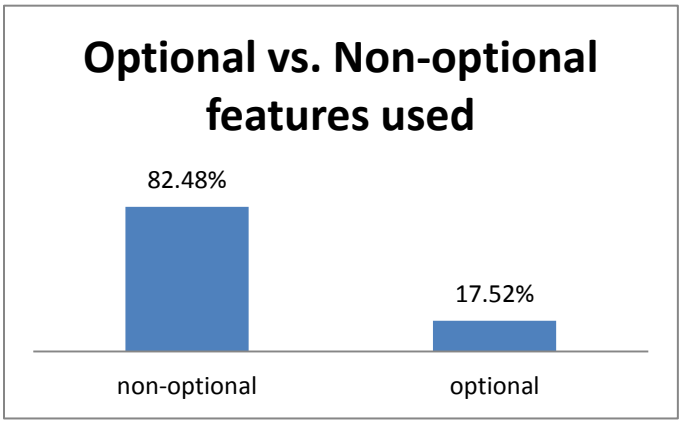
Table 1-5: dfmimic baseline validation AUC scores

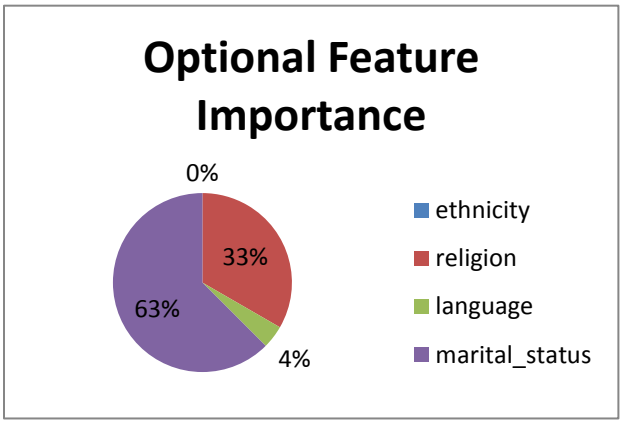
The one sample t-test gives a P-value of less than 0.0001, which is considered extremely significant.

Conclusion

Free-Form Visualization

One interesting aspect is the differences in which features are used in the best classifiers.





These graphs show non-optional features are overwhelmingly used more than optional features. Also, of the optional features, marital_status and religion are used much more than the other two optional features. This suggests that ethnicity and language do not provide useful information in predicting patient mortality in this hospital. While these features may be important in the medical field for disease diagnosis and patient care, which are important to the patient, they do not seem to affect patient outcomes. The best predictors use patient age and length of stay in combination with type of insurance.

Reflection

This project had many interesting challenges that gave me insight into the many different aspects of applying machine learning to problem. As the MIMIC-III database contained a wide range of different medical information, I found the data exploration aspect to be particularly challenging. With so many possible features to choose, it was difficult to narrow the scope of the project. The data preprocessing was also necessary in order to remove groups containing too few records. Perhaps the most interesting aspect is how ‘gender’ and ‘ethnicity’ were not

selected as important features. This seems to go against the findings by the CDC. However, as the initial data exploration showed, there is very little difference between the male and female mortality rate and other features may have provided a more significant difference. Also, the 'age', 'los', and 'insurance' features appear to have dominated the feature space. A final caveat is that while a model can be created to predict mortality quite well, it is very specific in its scope. This model should only be used by the Beth Israel Deaconess Medical Center as it specifically trains itself only using data from that hospital. While it would be very easy to continue to update this model to predict increased patient mortality at Beth Israel Deaconess Medical Center, it has limited usefulness in other hospitals or even on a national scale.

Improvement

While the MIMIC-III database is one of the largest publicly available databases for patient medical records, I feel that machine learning algorithms working with a much larger data set would be able to produce a better model. If the scope of the project were expanded to the national scale for a country like the United States of America, there would be approximately 4 million data points for one year's worth of data[viii]. This is about 100x more data than that contained in the MIMIC-III database. A model using more data would likely have a higher AUC score, and would be a better determinant of what role demographic features like ethnicity and language have in predicting patient mortality.

---

[i] Margaret Jean Hall, Ph.D.; Shaleah Levant, M.P.H.; and Carol J. DeFrances, Ph.D., "Trends in Inpatient Hospital Deaths: National Hospital Discharge Survey, 2000–2010", *Centers for Disease Control and Prevention*, March 2013, https://www.cdc.gov/nchs/products/databriefs/db118.htm

[ii] Becker, Edmund R., and Ali Rahimi. "Disparities in Race/ethnicity and Gender in in-Hospital Mortality Rates for Coronary Artery Bypass Surgery Patients." *Journal of the National Medical Association* 98.11 (2006): 1729–1739. Print.

[iii] https://www.census.gov/prod/cen2010/briefs/c2010br-03.pdf

[iv] http://www.idtheftcenter.org/images/breach/Overview2005to2016Finalv2.pdf

[v] 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 from: http://www.nature.com/articles/sdata201635

[vi] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[vii] "Private Leaderboard - Predicting 30 Day Hospital Readmissions", *Kaggle Inc.*, April 2015, https://inclass.kaggle.com/c/predicting-30-day-hospital-readmissions/leaderboard/private

[viii] George Washington University. "Nearly 50 percent increase in ICU admissions in U.S., new study says." ScienceDaily. ScienceDaily, 14 May 2013. <www.sciencedaily.com/releases/2013/05/130514212946.htm>.