Readmission Prediction - Summery

By Daniel Gordon

Introduction

One of the most important part of life quality is health. Every year billions of U.S dollars invest in health care industries. The need of optimize health care services is one of the most efficient ways to both save money and improve health care services.

Readmission after hospitalization within 30 days to discharge happen to almost 33% of patients according to our data set. In order to reduce this numbers, hospitals supply an extra service to guid patients in high risk of readmission and increase there chances to heal without been readmit. This serves given by specialist is both expansive and time consuming and can't be apply to every patient.

We generate a model that predict readmission within 30 days after hospitalization.That can be used to optimize health care services while saving money by focusing on patients with high risk of readmission.

For that end, we applied several classification techniques including logistic regression, KNN, decision tree and SVM.

Although we achieved decision tree model with 99% accuracy and no mistake on confusion matrix, we advise to try and minimize the number of features involved in the process since the model is pretty hard to interpret.

Data and Feature Analysis

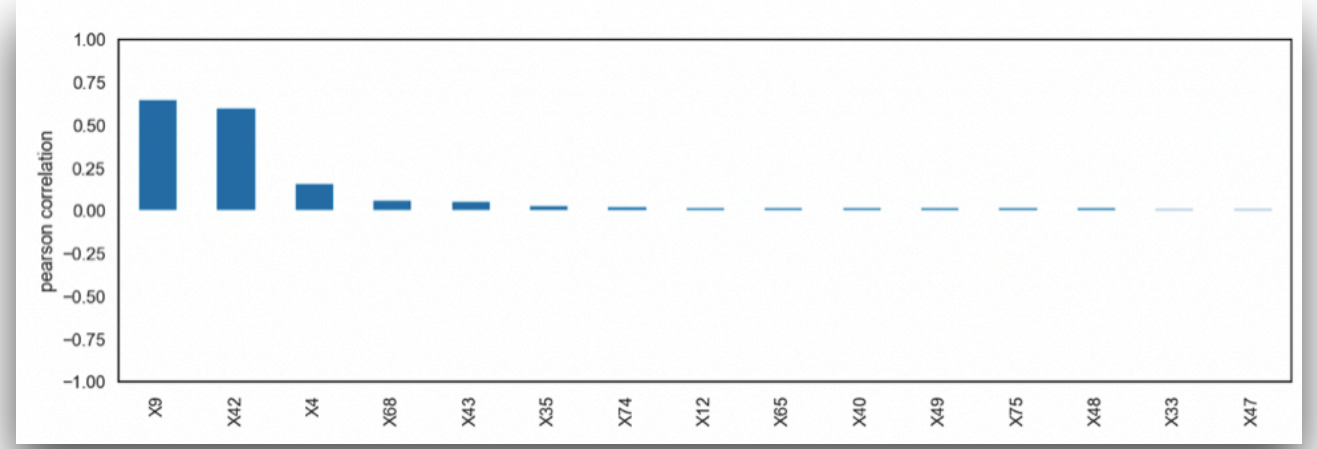
We used readmission unbalanced data set from health care data base. For the data set check the following link

https://github.com/DanielGordon-ml/Readmission_Prediction-/blob/main/ReadmissionsData.csv

The data include almost 14,000 samples and 92 features. That data has already been cleaned, checked for Nulls, and normalized but we did double check.

Due to the nature of the data, the features named has been replaced to X_1 to X_{92}. However, all the features are health related.

We then preformed features correlation analysis with the target variable and obtain the following 15 best features



Models

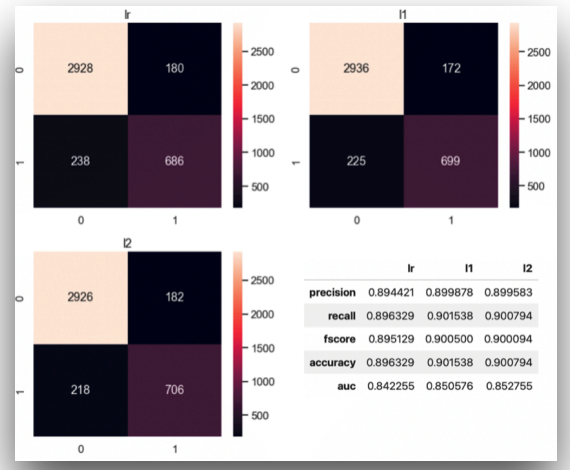
We used Stratified Shuffle Split to insure the same balance on both the train and test set. Then we preformed logistic regression, support vector machine and decision tree and obtain great results with 99% success on test set with the following decision tree.

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=11, max_features=60, random_state=
42)
```

With this in mind we introduce three different classification models and their evaluation.

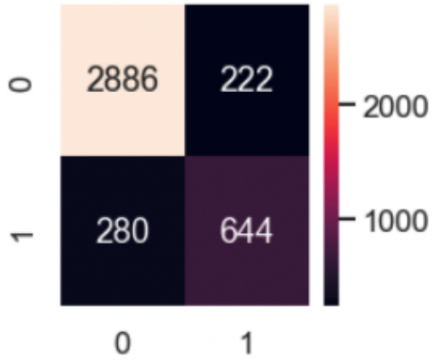
Logistic Regression

After train and test splitting, we trained logistic regression models with lr, l_1, l_2,regularizations, and cross validation over the train set. We then predict on the test value and obtain both accuracy table and confusion matrix



Support Vector Machine (SVM)

After train and test splitting, we trained grid search cv on SVM model with cross validation over multiple kernels, gammas values, and C values. To ensure recall and good precision, we set the scoring to f1. We achieved the best model for the polynomial kernel and f1 score of 0.72 for the readmission class and long fitting time. We then changed the feature set to a smaller set and have been able to stay with the same f1 score but only 15 best correlated features instead of 92 features.



Decision Tree

Last but not least, after train and test splitting, we trained grid search cv on decision tree model with cross validation over multiple parameters. We recall that this model is the best model for the job with 99% success.

```
param_grid = {'max_depth':range(1, dt.tree_.max_depth+1, 2),
              'max_features': range(1, len(dt.feature_importances_)+1)}

GR = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid=param_grid,
                  scoring='accuracy',
                  n_jobs=-1)

GR = GR.fit(X_train, y_train)
```

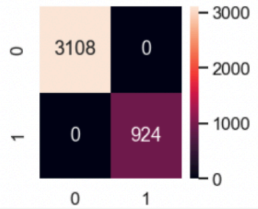
```
GR.best_estimator_.tree_.node_count, GR.best_estimator_.tree_.max_depth
```

```
(77, 11)
```

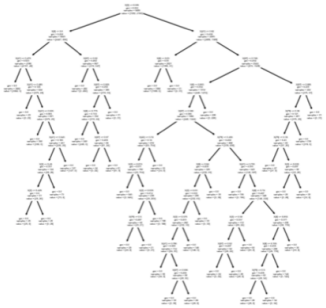
As we can see, we get our best estimator for max depth=11. Note that the scoring method is accuracy but since we obtain success on every matrix score, we are not worried. Indeed,

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3108
           1       1.00      1.00      1.00       924

    accuracy                           1.00      4032
   macro avg       1.00      1.00      1.00      4032
weighted avg       1.00      1.00      1.00      4032
```

```
]: <AxesSubplot:>
```



We are furthermore can visualize the decision tree as follow (for better visualization).



Further Optimization

Although we achieved decision tree model with 99% accuracy and no mistake on confusion matrix, we advise to try and minimize the number of features involved in the process since the model is pretty hard to interpret.