İSTANBUL ŞEHİR ÜNIVERSITY

ISE-537 APPLIED DATA ANALYSIS

FINAL PROJECT

Analyzing and Modelling Intensive Care Unit (UCI) Patients Data

By using Python

Submitted to : Dr. Mehmet Yasin Ulukuş

Submitted by :

Güzide Demir 215150432

Merve Özer 319720181

1. Importing the data

Output:

Out[3]:

	Patientid	ICU Type	Origin Level of Care	Age	Sex	LOS	previous_LOS	previous_ICU_stays	Type	Charlson_index	...	var_SOFA_Nerv	Var_SOFA_Renal
0	213620	Trauma	Regular	61	M	258.283333	2.166667	0	SURG	0	...	0.000000	0.000000
1	213630	Cardiac-T	HOME	74	M	37.933333	0.000000	0	SURG	0	...	0.166667	0.000000
2	213633	Surgical	HOME	53	M	304.566667	0.000000	0	SURG	0	...	0.172549	0.038431
3	213650	Medical	HOME	39	F	19.416667	0.000000	0	MED	0	...	2.250000	0.000000
4	213654	Neuro	HOME	55	M	268.700000	0.000000	0	SURG	0	...	0.158586	0.000000

5 rows × 51 columns

"death" column is considered as a label.

Fill all null variables with median of the features,

To get the dummy variable of the categorical variables, we need to convert numerical values into dummy variables,

Out[10]:

	Patientid	Age	LOS	previous_LOS	previous_ICU_stays	Charlson_index	weekend	disch_night	cvc_status	SIRS_48_hour	...	ICU Type_Surgical
0	213620	61	258.283333	2.166667	0	0	0	1	1	0	...	0
1	213630	74	37.933333	0.000000	0	0	0	0	1	0	...	0
2	213633	53	304.566667	0.000000	0	0	0	1	0	1	...	1
3	213650	39	19.416667	0.000000	0	0	0	1	0	0	...	0
4	213654	55	268.700000	0.000000	0	0	0	0	0	1	...	0

5 rows × 60 columns

2. *Split data into train and test using death column as a target.*

We can now do validation by splitting the into training and test in order to predict response value *death.*

The test size percentage is chosen as 0.20. (80:20)

3. *With Multinomial Logistic Regression model*

Output:

```
Out[13]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=10000.0, l1_ratio=None, max_iter=1000000,
                            multi_class='multinomial', n_jobs=None, penalty='l2',
                            random_state=None, solver='lbfgs', tol=1e-06, verbose=0,
                            warm_start=True)
```

4. Performing *5-fold Cross Validation*

Computing accuracy by using cross validation

Output: `0.84494`
The accuracy of the model is approximately **85%.**

5. *Predicting the Test Data and Applying Performance Measure*

First we train the data and then predict test data.

Plotting the confusion Matrix

Output:

```
[[2520  84]
 [ 386 222]]
```

Accuracy, Precision, Recall and F1 score:

Accuracy: 85%

Out of all the classes, how much we predicted correctly, which will be, in this case 85%, it should be high as possible.

Precision: 72%

Out of all the positive classes we have predicted correctly, how many are actually positive. 72% are positive.

Recall: 36%

Out of all the positive classes, how much we predicted correctly which is 36%. It should be high as possible.

F1 score: 48%

It is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time.

6. **Computing loss function of Logistic Regression**

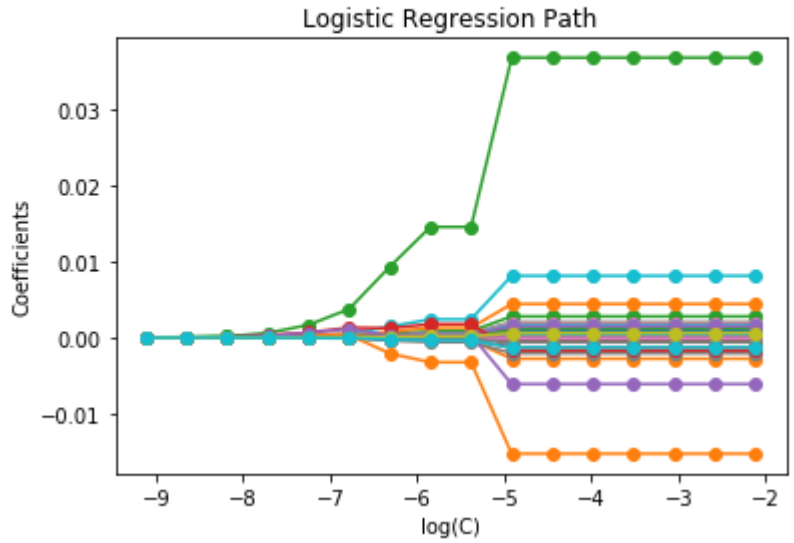
Output: 5.05
Accuracy of the model: 85%

7. **Computing the regularization path into Logistic Regression using Lasso**

Comment:

The models are ordered from strongest regularized to least regularized. The coefficients of the models are collected and plotted as a "regularization path": on the middle of the figure (strong regularizers), all the coefficients are exactly 0. When regularization gets progressively looser, coefficients can get non-zero values one after the other.

```
Computing regularization path ...
This took 2.287s
```



8. ***To find the best model, we applied Grid Search by using Logistic Regression.***

Output:

```
{'C': 1000.0, 'penalty': 'l1'}
```

```
The accuracy of the model is 90.75%.
```

9. *Testing the Logistic Regression with best parameters*

Output:

```
LogisticRegression(C=1000.0, class_weight=None, dual=False,
fit_intercept=True, intercept_scaling=1, l1_ratio=None,
max_iter=100, multi_class='warn', n_jobs=None, penalty='l1',
random_state=None, solver='warn', tol=0.0001, verbose=0,
warm_start=False)
```

Evaluating the Logistic Regression model with best parameters.

Accuracy of the logistic regression model represents the best model that we should select.

Output: 91%

print(confusion_matrix(y_test, best_log_pred))

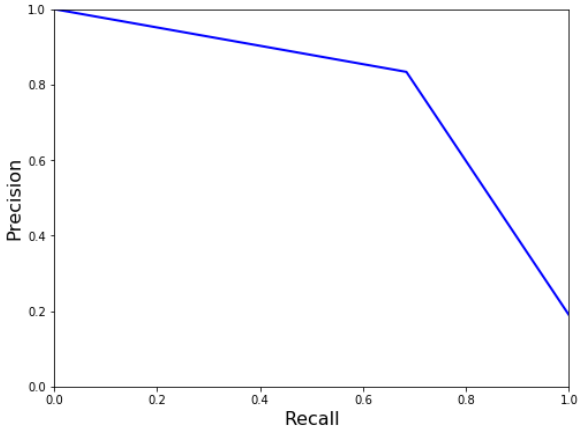
```
[[2521  83]

[ 192 416]]
```

Classification Report for the best model.

	precision	recall	f1-score	support
0	0.93	0.97	0.95	2604
1	0.84	0.68	0.75	608
accuracy			0.91	3212
macro avg	0.88	0.82	0.85	3212
weighted avg	0.91	0.91	0.91	3212

10. *Plotting Precision Recall Curve for the Logistic Regression model with best parameters.*

Output:



Comment:

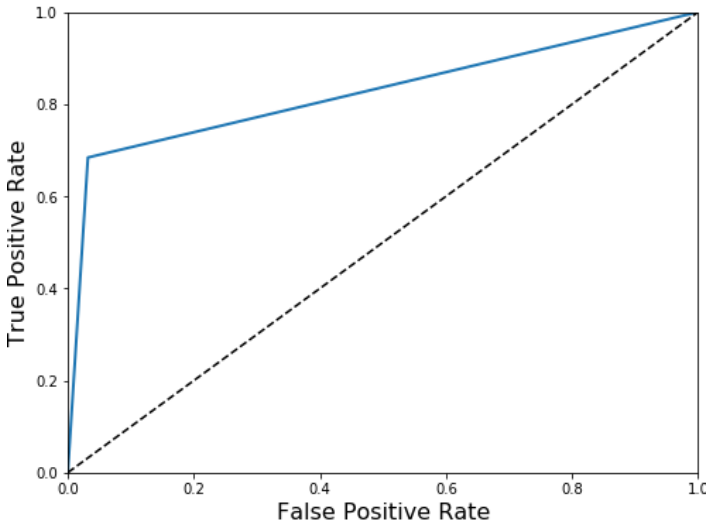
It is desired that the algorithm should have both high precision, and high recall.

Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly. We found out precision score 91% and recall score 91% represent returning many results with all results labeled correctly.

11. Plotting ROC Curve for the Logistic Regression model with best parameters.

Output:



Comment:

In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test. Hence the plot shows that our model has a ROC curve that has a higher AUC (Area Under Curve) than the 45-degree line which means that we created a model that does better than random guessing.

12. To find the best model *we applied Grid Search by using Decision Tree model.*

Output:

```
Out[67]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=0, splitter='best')
```

Defining the parameters for the Grid search

```
Out[68]: GridSearchCV(cv=5, error_score='raise-deprecating',
                      estimator=DecisionTreeClassifier(class_weight=None,
                                                       criterion='gini', max_depth=None,
                                                       max_features=None,
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       presort=False, random_state=0,
                                                       splitter='best'),
                      iid='warn', n_jobs=None,
                      param_grid={'max_depth': [None, 1, 5, 10],
                                  'max_features': ['auto', 'sqrt', 'log2'],
                                  'max_leaf_nodes': [10, 50, 100, 250],
                                  'min_samples_split': [0.01, 0.05, 0.1, 0.3]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)
```

Finding the best parameters of Decision Tree Model

Output:

```
{'max_depth': None,

'max_features': 'auto',

'max_leaf_nodes': 50,

'min_samples_split': 0.01}
```

The Accuracy of decision tree with best parameters

The accuracy of the model is `0.8988 , 90%`

Testing the Decision Tree Model with best parameters

Applying parameters which is founded as best parameters to the decision tree.

```
Out[87]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features='auto', max_leaf_nodes=50,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=0.01,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=None, splitter='best')
```

Evaluating the best model:

Accuracy of the model is 89%.

Confusion matrix of the best model:

```
[[2476 128]
 [ 210 398]]
```

Classification report for the best model:

	precision	recall	f1-score	support
0	0.92	0.95	0.94	2604
1	0.76	0.65	0.70	608
accuracy			0.89	3212
macro avg	0.84	0.80	0.82	3212
weighted avg	0.89	0.89	0.89	3212

13. Splitting the data into train and test set using Readmitted column as a target.

Applying Logistic Regression with best parameters founded by Grid Search

```
Out[21]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='warn', n_jobs=None, penalty='l2',
                            random_state=0, solver='warn', tol=0.0001, verbose=0,
                            warm_start=False)
```

The best parameter:

Output:

```
{'C': 1.0, 'penalty': 'l1'}
```

The model accuracy is: 85%.

14. Testing Logistic Regression model with best parameters on Readmission target value.

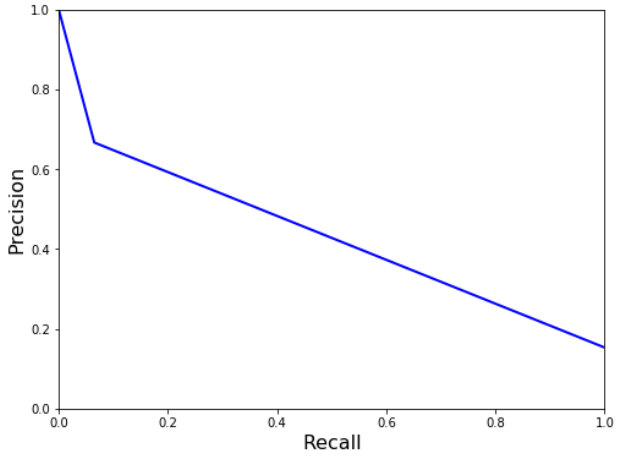
```
Out[29]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='warn', n_jobs=None, penalty='l1',
                            random_state=None, solver='warn', tol=0.0001, verbose=0,
                            warm_start=False)
```

Model accuracy : 85%

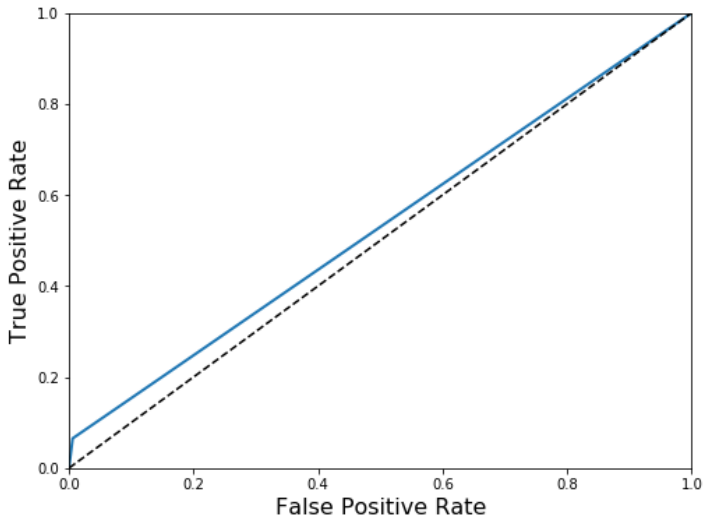
Confusion Matrix:

```
Out[42]: array([[2705,   16],
                [ 459,   32]], dtype=int64)
```

15. Plotting Precision Recall Curve with the best Logistic Regression model using Readmission datas as target.



16. Plotting ROC Curve of the best Logistic Regression model.



Classification Report:

	precision	recall	f1-score	support
0	0.85	0.99	0.92	2721
1	0.67	0.07	0.12	491
accuracy			0.85	3212
macro avg	0.76	0.53	0.52	3212
weighted avg	0.83	0.85	0.80	3212

17. Applying Decision Tree Model with best parameters founded by Grid Search

```
Out[44]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=0, splitter='best')
```

Defining parameters for the grid search

```
Out[46]: GridSearchCV(cv=5, error_score='raise-deprecating',
                      estimator=DecisionTreeClassifier(class_weight=None,
                                                       criterion='gini', max_depth=None,
                                                       max_features=None,
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       presort=False, random_state=0,
                                                       splitter='best'),
                      iid='warn', n_jobs=None,
                      param_grid={'max_depth': [None, 1, 5, 10],
                                  'max_features': ['auto', 'sqrt', 'log2'],
                                  'max_leaf_nodes': [10, 50, 100, 250],
                                  'min_samples_split': [0.01, 0.05, 0.1, 0.3]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)
```

```
Out[47]: {'max_depth': 5,
          'max_features': 'auto',
          'max_leaf_nodes': 50,
          'min_samples_split': 0.01}
```

The model accuracy: 85%.

18. Evaluating Decision Tree Model with Test Data

```
Out[51]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
                                max_features='auto', max_leaf_nodes=50,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=0.01,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=None, splitter='best')
```

The model accuracy is: 85%.

Confusion matrix:

Output:

```
array([[2718,  3],
[ 490,  1]], dtype=int64)
```

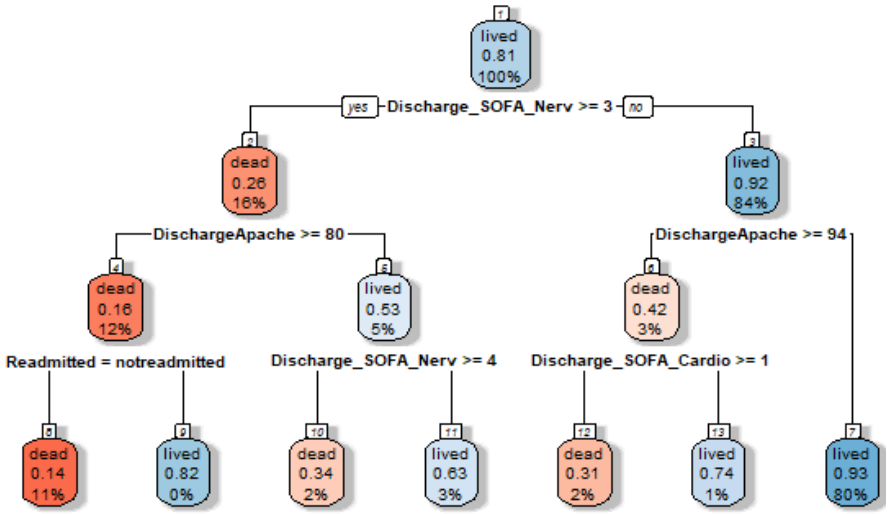
Classification Report:

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      2721
           1       0.25      0.00      0.00       491

    accuracy                           0.85      3212
   macro avg       0.55      0.50      0.46      3212
weighted avg       0.76      0.85      0.78      3212
```

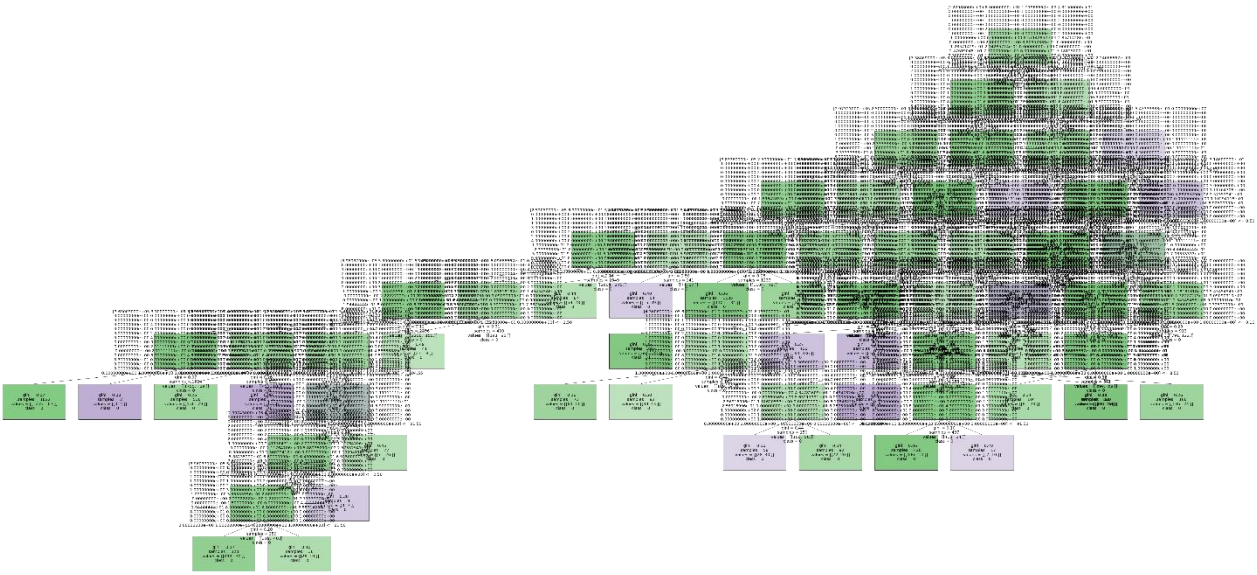
Decision Tree

R-Studio

```
library(rpart)
library(rpart.plot)
install.packages("rpart.plot")

mydata3 <- read.csv(file.choose(), header=TRUE)
tree <- rpart(death~., data=mydata3, method='class')
rpart.plot(tree, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```



Visualize the Decison Tree (python)



Random Forest Classifier.

Defining Random Forest Classifier Model for death target value.

Defining parameters for Random Forest Regressor.

Creating RandomForestClassifier model and use it in GridSearchCv to find best combination.

```
Out[60]: GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=None,
                                              oob_score=False, random_state=0,
                                              verbose=0, warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [None, 1, 5, 10],
                         'n_estimators': [25, 50, 100, 250]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_squared_error', verbose=1)
```

Finding best parameters of Random Forest Classifier.

Output:

```
{'max_depth': None, 'n_estimators': 100}
```

Testing Random Forest Classifier with best parameters

```
Out[64]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)
```

Make prediction with test set.

Confusion Matrix

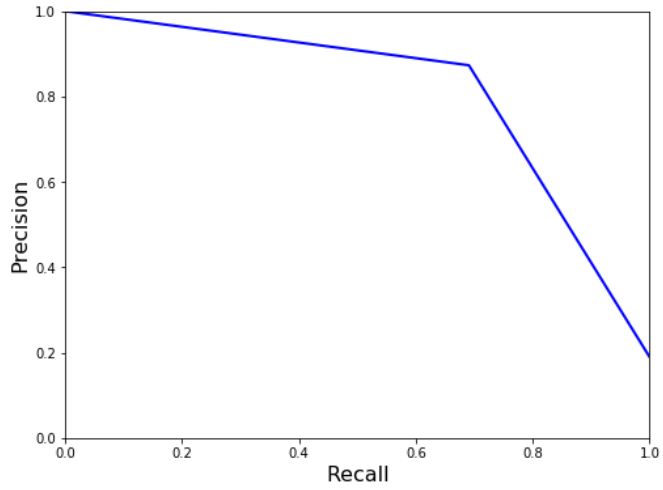
```
[[2543  61]
[ 188  420]]
```

Accuracy of the Model: 92%

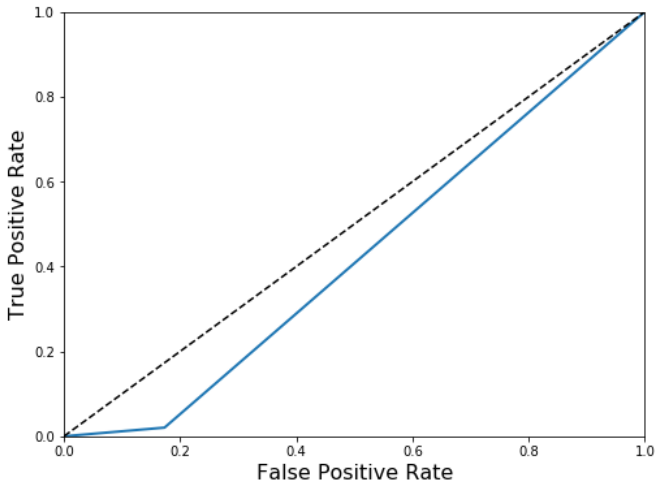
Classification Report

	precision	recall	f1-score	support
0	0.93	0.98	0.95	2604
1	0.87	0.69	0.77	608
accuracy			0.92	3212
macro avg	0.90	0.83	0.86	3212
weighted avg	0.92	0.92	0.92	3212

Plotting Precision Recall Curve for Random Forest Model



Plotting ROC Corve for Random Forest Model



Defining Random Forest Classifier Model for readmitted target value.

```
Out[121]: GridSearchCV(cv=5, error_score='raise-deprecating',
                       estimator=RandomForestClassifier(bootstrap=True, class_weight=None,
                                                        criterion='gini', max_depth=None,
                                                        max_features='auto',
                                                        max_leaf_nodes=None,
                                                        min_impurity_decrease=0.0,
                                                        min_impurity_split=None,
                                                        min_samples_leaf=1,
                                                        min_samples_split=2,
                                                        min_weight_fraction_leaf=0.0,
                                                        n_estimators='warn', n_jobs=None,
                                                        oob_score=False, random_state=0,
                                                        verbose=0, warm_start=False),
                       iid='warn', n_jobs=None,
                       param_grid={'max_depth': [None, 1, 5, 10],
```

Finding best parameters of Random Forest Classifier

Output:

```
{'max_depth': None, 'n_estimators': 100}
```

Evaluate Random Forest model with best parameters on test set.

```
Out[123]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                 max_depth=None, max_features='auto', max_leaf_nodes=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, n_estimators=100,
                                 n_jobs=None, oob_score=False, random_state=None,
                                 verbose=0, warm_start=False)
```

Model Accuracy of Random Forest : 85%

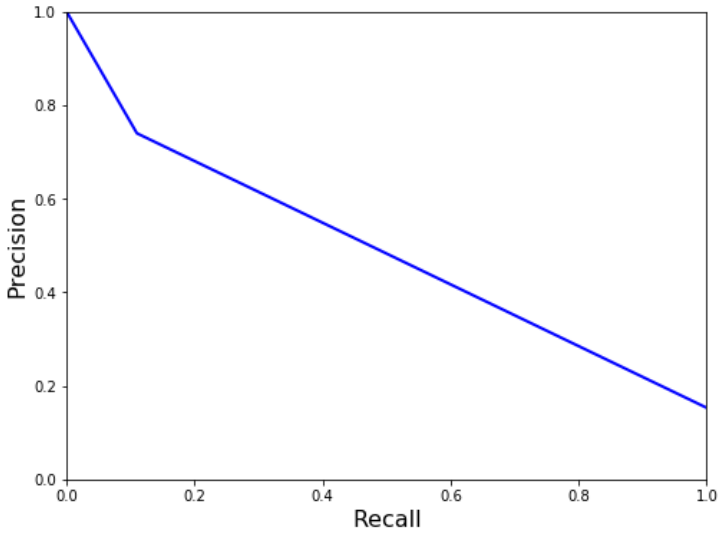
Confusion Matrix:

```
[[2702  19]
[ 437  54]]
```

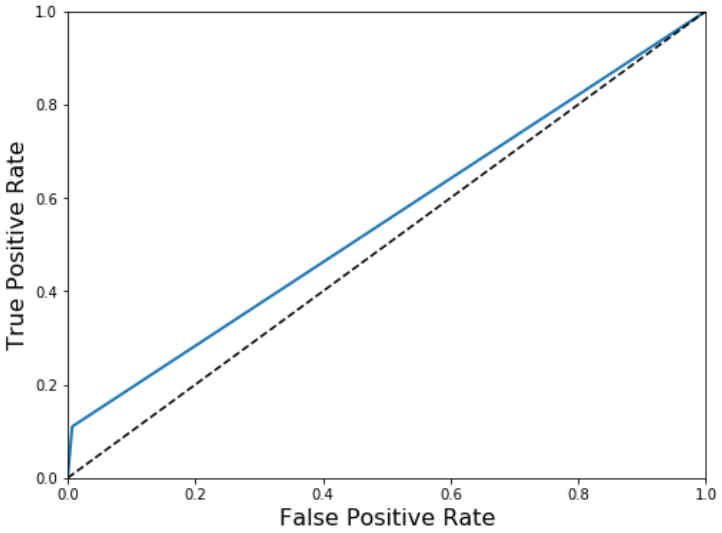
Classification Report for Random Forest:

	precision	recall	f1-score	support
0	0.86	0.99	0.92	2721
1	0.74	0.11	0.19	491
accuracy			0.86	3212
macro avg	0.80	0.55	0.56	3212
weighted avg	0.84	0.86	0.81	3212

Plotting Precision Recall Curve



ROC Curve



Conclusion:

If we check out all the models that we analyzed, the best model will be random forest since it provides model accuracy 92% when death column is labeled and 85% when readmitted is labeled.