**MScBMI 33200 – Machine Learning for Biomedical Informatics**

**Assignment IV**

**<Troy Zhongyi Zhang>**

Directions:

1. Fill out below information (tables and methods)
2. Submit this document along with your code in an HTML/PDF format

**Section 1: EMR Bots 30-day Readmission study**

Using the training datasets, create the following models:

1. Naïve model: This model utilizes only patient characteristics (age, gender and race) to predict 30-day readmission in a logistic regression framework
2. GLM model : This model utilizes patient characteristics and most-recent lab recordings to predict 30-day admissions in a logistic regression framework.
3. ANN model: This model utilizes patient characteristics and most-recent lab recordings to predict 30-day admissions using an artificial neural network. Feature engineering steps include balancing classes using SMOTE as well as data normalization/standardization of continuous variables.
4. RF Model : This model utilizes patient characteristics and most-recent lab recordings to predict 30-day admissions using a random forest.
5. GBM Model: This model utilizes patient characteristics and most-recent lab recordings to predict 30-day admissions using a gradient boosted machine.

Utilize a five-fold cross-validation technique to build your model.

Calculate AUC on the test dataset. Fill out the following Table.

|  |  |
| --- | --- |
|  | AUC (95% CI) |
| Naïve Model | 0.4906 (0.4103, 0.5708) |
| Logistic Regression | 0.4779 (0.3918, 0.5640) |
| ANN | 0.5075 (0.4281, 0.5869) |
| Random Forest | 0.5018 (0.4295, 0.5741) |
| GBM | 0.5061 (0.4298, 0.5823) |

Insert details (packages, parameter selection, etc.) on the models that were developed in the space given below.

Methods:

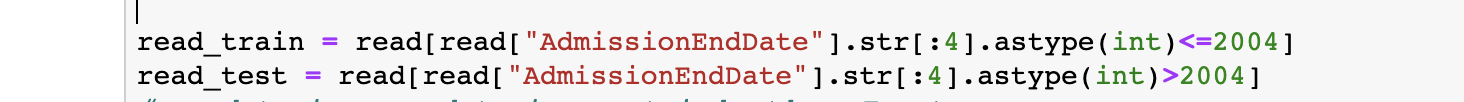
A screenshot of a cell phone

Description automatically generated

Firstly, import pandas as pd

import numpy as np

I separate the dataset into training and testing sets.



**Naïve Model:**

I select patient characteristics (age, gender and race) as the dataset

A screenshot of a cell phone

Description automatically generated

from sklearn.model\_selection import GridSearchCV, StratifiedKFold, train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from random import seed

and set seed(0)

I chose some parameters for GridSearchCV and I set “cv=5”

A screenshot of a cell phone

Description automatically generated

I used “model.best\_estimator\_” to find the best parameters of the Naïve model according to the GridSearchCV and Cross validation.

from sklearn import metrics for confusion matrix

from sklearn.metrics import classification\_report

A screenshot of a cell phone

Description automatically generated

from sklearn.metrics import roc\_auc\_score to calculate 95% AUC

A screenshot of a cell phone

Description automatically generated

Since Python does not have the pROC package, I transplanted the DeLong function from R to calculate the confidence interval in Python

import numpy as np; import scipy.stats; from scipy import stats

#Transplanted the pROC package from R into Python for CI computation

import numpy as np

import scipy.stats

from scipy import stats

A screenshot of a cell phone

Description automatically generated

**Logistic Regression:**

I choose the patient characteristics and latest lab observations as my dataset.

A screenshot of a cell phone

Description automatically generated

from sklearn.linear\_model import LogisticRegression to fit a logistic regression model

Used GridSearchCV and set “cv=5” for 5 cross-validations to tune parameters.

A screenshot of a cell phone

Description automatically generated

Then I used above method to calculate the AUC and 95% CI with using “model.best\_estimator\_” to apply the best parameters for prediction:

A screenshot of a cell phone

Description automatically generated

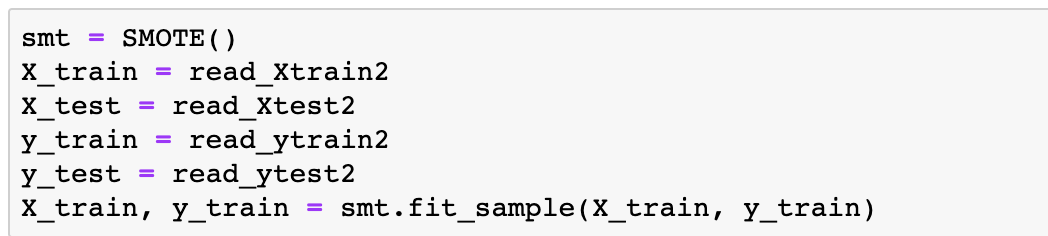
A screenshot of a cell phone

Description automatically generated

**Artificial Neural Network:**

from imblearn.over\_sampling import SMOTE for using the SMOTE to balance the dataset.

Then I filled my training sets:



To test my new dataset after SMOTE is balanced: (Both outcome=1 and outcome=1 has the same number of rows. I upsampled the minority group to equal to the majority group)

A screenshot of a cell phone

Description automatically generated

from sklearn.preprocessing import StandardScaler to scale my data. I fitted my scaler with the training set and transferred the scaling to my testing set to avoid different variances.

A screenshot of a cell phone

Description automatically generated

from sklearn.neural\_network import MLPClassifier to fit a neural network model.

The I used GridSearchCV to tune parameters with 5 cross-validation.

A screenshot of a cell phone

Description automatically generated

Lastly, I used above method to calculate the AUC and 95% CI with using “model.best\_estimator\_” to apply the best parameters for prediction:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Random Forest:**

from sklearn.ensemble import RandomForestClassifier to fit a Random Forest model. I used GridSearchCV with 5 cross-validations to tune parameters, and I used the same training and testing datasets as the logistic regression model’s datasets. These datasets are before scaling:

A screenshot of a cell phone

Description automatically generated

I used above method to calculate the AUC and 95% CI with using “model.best\_estimator\_” to apply the best parameters for prediction:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Gradient Boosting Machines:**

from sklearn.ensemble import GradientBoostingClassifier to fit a Gradient Boosting model. I used GridSearchCV with 5 crross-validations to tune parameters, and I used the same training and testing datasets as the logistic regression model’s datasets, which are before scaling:

A screenshot of a cell phone

Description automatically generated

I used above method to calculate the AUC and 95% CI with using “model.best\_estimator\_” to apply the best parameters for prediction:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Section 2: Gusto study**

Using the training datasets, create the following models:

1. GLM model: This model utilizes all features to predict 30-day mortality in a logistic regression framework.
2. Ridge Regression model: This model utilizes all features to predict 30-day mortality in a logistic regression framework with regularization.
3. ANN model: This model utilizes all features to predict 30-day mortality using an artificial neural network. Feature engineering steps should use normalization/standardization of continuous variables.
4. Random Forest model: This model utilizes all features to predict 30-day mortality using a random forest.
5. GBM model: This model utilizes all features to predict 30-day mortality using a gradient boosted machine.

Utilize a five-fold cross-validation technique to build your model.

Calculate AUC on the test dataset. Fill out the following Table.

|  |  |
| --- | --- |
|  | AUC (95% CI) |
| Logistic Regression | 0.8302 (0.7938, 0.8666) |
| Ridge Regression | 0.8280 (0.7912, 0.8647) |
| ANN | 0.7596 (0.7145, 0.8047) |
| Random Forest | 0.8853 (0.8523, 0.9183) |
| GBM Model | 0.8721 (0.8403, 0.9039) |

Insert details on the models that were developed in the space given below.

**Methods:** Firstly, same as the above question, import pandas as pd, import numpy as np

A screenshot of a cell phone

Description automatically generated

Then I separated the GUSTO dataset into training and testing datasets:  
A screenshot of a cell phone

Description automatically generated

**Logistic Regression:**

from sklearn.linear\_model import LogisticRegression to fit a logistic regression model. I used GridSearchCV with 5 cross validations to tune parameters.

from sklearn.model\_selection import GridSearchCV

A screenshot of a cell phone

Description automatically generated

I used model.best\_estimator\_ to predict results.

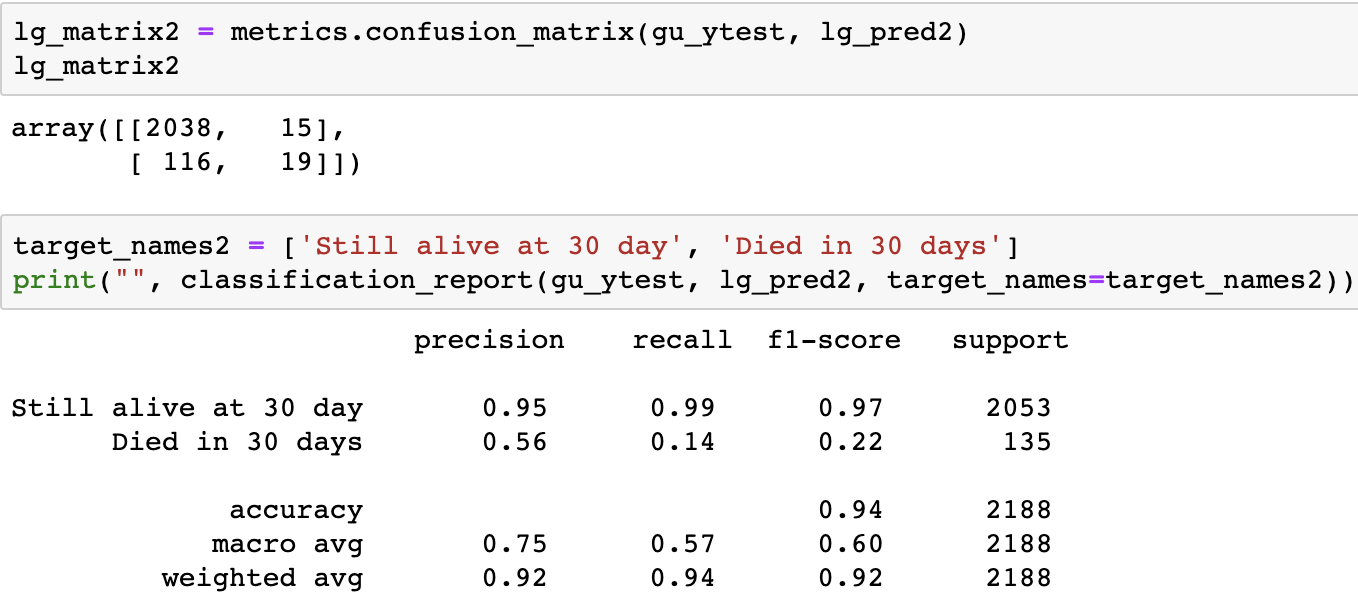
A screenshot of a cell phone

Description automatically generated

Then I imported the metrics package to obtain the confusion matrix:

from sklearn import metrics for confusion matrix

from sklearn.metrics import classification\_report



Finally, I used same method to calculate the confidence interval and 95% CI with using model.best\_parameter\_ for predictions:

from sklearn.metrics import roc\_auc\_score to calculate 95% AUC

import numpy as np; import scipy.stats; from scipy import stats

#Transplanted the pROC package from R into Python for CI computation

import numpy as np

import scipy.stats

from scipy import stats

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Ridge Regression Model:

from sklearn.linear\_model import RidgeCV

I directly set cv = 5 for 5 cross-validations to predict results

A screenshot of a cell phone

Description automatically generated

I used above method to calculate the AUC and 95% confidence interval:

A screenshot of a cell phone

Description automatically generated

# Artificial Neural Network:

from sklearn.preprocessing import StandardScaler to scale my data. I fitted my scaler with the training set and transferred the scaling to my testing set to avoid different variances.

A screenshot of a cell phone

Description automatically generated

from sklearn.neural\_network import MLPClassifier to fit a neural network model.

The I used GridSearchCV to tune parameters with 5 cross-validations.

A screenshot of a cell phone

Description automatically generated

Lastly, I used above method to calculate the AUC and 95% CI with using “model.best\_estimator\_” to use the best parameters for prediction:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Random Forest:

from sklearn.ensemble import RandomForestClassifier to fit a Random Forest model. I used GridSearchCV with 5 cross-validations to tune parameters, and I used the same training and testing datasets as the logistic regression model’s datasets. These datasets are before scaling:

A screenshot of a cell phone

Description automatically generated

I used above method to calculate the AUC and 95% CI with using “model.best\_estimator\_” to use the best parameters for prediction:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Gradient Boosting Machines:

from sklearn.ensemble import GradientBoostingClassifier to fit a Gradient Boosting model. I used GridSearchCV with 5 crross-validations to tune parameters, and I used the same training and testing datasets as the logistic regression model’s datasets, which are before scaling:

A screenshot of a cell phone

Description automatically generated

I used above method to calculate the AUC and 95% CI with using “model.best\_estimator\_” to apply the best parameters for prediction:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Section 3: Short Answer Questions**

Please answer the following questions briefly.

1. **Explain briefly the intuition behind using information gain when building decision trees.’**

**Ans.** Information gain measures how predictable the data became (or how much more information gained) by splitting the data according to the answer provided at the decision node. Information gained using feature F on data Y is defined by: IG(Y, F) = H(Y)-H(Y|F). The more powerful a feature is, the more information is gained by splitting the data on that feature.

1. **What is difference between test error as measured by cross-validation and test error measured through bootstrap?**

**Ans.** One difference is that cross-validation, like jackknife, uses all of the data points, whereas bootstrapping, which resamples the data randomly, may not hit all the points. The error comes down to variance and bias (as usual). CV tends to be less biased but K-fold CV has fairly large variance. On the other hand, bootstrapping tends to drastically reduce the variance but gives more biased results (they tend to be pessimistic). For large sample sizes, the variance issues become less important and the computational part is more of an issues. I still would stick by repeated CV for small and large sample sizes. Cross validation is a good tool when deciding on the model. However, when my model is fixed, using the bootstrap makes more sense.

1. **Why do we need de-correlated trees when developing random forests?**

**Ans.** Random Forests is a powerful ensemble method used to build predictive models by constructing a large number of de-correlated trees by showing different training sets to solve the problem of overfitting which involves averaging the predictor output from different tree models which are distributed across helping in reducing the variance.

1. **What does a varImp() on a random forest model indicate?**

**Ans.** The varImp function tracks the changes in model statistics, such as the GCV, for each predictor and accumulates the reduction in the statistic when each predictor's feature is added to the model. This total reduction is used as the variable importance measure. varrImp() is a generic method for calculating variable importance for objects produced by training set and method specific methods.  For the out-of-bag sample of each tree calculate the accuracy of the tree, then permute the variables one after the other and measure the accuracy after permutation to calculate the decrease in accuracy without that variable.

1. **True or False: Boosting involves combining independent decision trees as learners.**

**Ans.** False. Bagging works on independent models but boosting works on sequential models.

1. **What is the interpretation of learning rate in a gradient boosting model?**

**Ans.** The learning rate corresponds to how quickly the error is corrected from each tree to the next and is a simple multiplier 0<LR≤1. For example, if the current prediction for a particular example is 0.2 and the next tree predicts that it should actually be 0.8, the correction would be +0.6. At a learning rate of 1, the updated prediction would be the full 0.2+1\*(0.6) =0.8, while a learning rate of 0.1 would update the prediction to be 0.2+0.1\*(0.6) = 0.

1. **Suppose I used the top 10 features as indicated by varImp() from my random forest model to create a multiple linear regression model. Is this a valid method? Why or why not?**

Ans. No, this will not be a valid method. Variables that are important in random forest don't necessarily have any sort of linearly additive relationship with the outcome. This remark shouldn't be surprising: it's what makes random forest so effective at discovering nonlinear relationships. In general, random forest can fit different models to different subsets of the data. It won't be obvious what's going on from a single plot at all, and building a linear model of similar predictive power will be even harder. Some of the features we got from varImp() through random forest may not fall in a linear decision boundary, and these features won’t work well with multiple linear regression.

Some notes:

A method that first looks at univariate correlations for pre-identifying things that should go into a final model, will tend to do badly for a number of reasons: ignoring model uncertainy, using statistical significance/strength of correlation as a criterion to select, "falsely" identifying predictors in univariate correlations, and missing out on predictors. Not wrapping this into any form of bootstrapping/cross-validation/whatever to get a realistic assessment of my model uncertainty is likely to mislead me. Furthermore, treating continuous predictors as having linear effects can often be improved upon by methods that do not make such an assumption (e.g. RF).

Using RF as a pre-selection for a linear model is not such a good idea. Variable importance is really hard to interpret, and it is really hard to set a cut-off on it. I will not know whether variable importance is about the variable itself or about interactions, plus I am losing out on non-linear transformations of variables.