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

**Assignment V**

**<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

**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.
6. SVM model: This model utilizes all features to predict 30-day mortality using a support vector 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) |
| SVM | 0.7898 (0.7448, 0.8348) |

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

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. I calculated 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

# Support Vector Machine:

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.svm import SVC to fit a Support Vector Machine Classifier model. Then 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