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

```
import pandas as pd
import numpy as np
from random import seed
seed(1)
```

Then I separated the GUSTO dataset into training and testing datasets:

```
gu_train = gusto.loc[(gusto['GROUP'] == 1
                    ) | (gusto['GROUP'] == 2
                        ) | (gusto['GROUP'] == 3)]
gu_test = gusto[gusto['GROUP']==0]

gu_Xtrain = gu_train.drop("DAY30",axis=1)
gu_ytrain = gu_train[['DAY30']]
gu_Xtest = gu_test.drop("DAY30",axis=1)
gu_ytest = gu_test[['DAY30']]
```

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

```
seed(0)

params6 = {'tol' : [1e-6,1e-5,1e-4,1e-3,1e-2],
           'C': [0.5,1.0,1.5,2.0,2.5]}
lg2 = LogisticRegression(random_state=0, solver='warn',multi_class='warn')
lg2 = GridSearchCV(lg2, cv=5, param_grid=params6, scoring = 'roc_auc',refit = True,
                   n_jobs=-1, verbose = 5, return_train_score=True)

lg2.fit(gu_Xtrain, gu_ytrain)
lg2.cv_results_
```

I used model.best_estimator_ to predict results.

```
lg_pred2 = lg2.best_estimator_.predict(gu_Xtest)
lg_pred2
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

Then I imported the metrics package to obtain the confusion matrix:
from sklearn import metrics for confusion matrix
from sklearn.metrics import classification_report

```
lg_matrix2 = metrics.confusion_matrix(gu_ytest, lg_pred2)
lg_matrix2
```

```
array([[2038,   15],
       [ 116,   19]])
```

```
target_names2 = ['Still alive at 30 day', 'Died in 30 days']
print("", classification_report(gu_ytest, lg_pred2, target_names=target_names2))
```

```
                      precision    recall  f1-score   support

Still alive at 30 day       0.95      0.99      0.97      2053
      Died in 30 days       0.56      0.14      0.22       135

             accuracy                           0.94      2188
            macro avg       0.75      0.57      0.60      2188
         weighted avg       0.92      0.94      0.92      2188
```

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

```
lg_probs2 = lg2.best_estimator_.predict_proba(gu_Xtest)[:,1]
print(roc_auc_score(gu_ytest, lg_probs2))
```

```
0.8302033158341001
```

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.8302033158341
AUC COV: 0.0003444411856651161
95% AUC CI: [0.7938281  0.86657854]
```

Ridge Regression Model:

from sklearn.linear_model import RidgeCV
I directly set cv = 5 for 5 cross-validations to predict results

```
from sklearn.linear_model import RidgeCV
ridgecv = RidgeCV(alphas=[1e-3, 1e-2, 1e-1, 1, 10], cv=5,
ridgecv=ridgecv.fit(gu_Xtrain,gu_ytrain)
gu_ridgecv = ridgecv.predict(gu_Xtest)
roc_auc_score(gu_ytest, gu_ridgecv)
```

```
0.8279590842669263
```

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

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.8279590842669264
AUC COV: 0.0003523557006508896
95% AUC CI: [0.79116833 0.86474984]
```

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.

```
scaler = StandardScaler()
# Fit only to the training data
scaler = scaler.fit(gu_Xtrain)
gu_Xtrains = scaler.transform(gu_Xtrain)
gu_Xtests = scaler.transform(gu_Xtest)
```

from sklearn.neural_network import MLPClassifier to fit a neural network model.
The I used GridSearchCV to tune parameters with 5 cross-validations.

```
seed(1)
# skf = StratifiedKFold(n_splits=5)
params7 = {'alpha' : [0.0001,0.01],
           'power_t': [0.5,0.75],
           'max_iter': [200,250]}
mlp2 = MLPClassifier(solver='lbfgs', random_state=1)
mlp2 = GridSearchCV(mlp2, cv=5, param_grid=params7, scoring = 'roc_auc',refit = True,
                    n_jobs=-1, verbose = 5, return_train_score=True)

mlp2.fit(gu_Xtrains, gu_ytrain)
mlp2.cv_results_
```

Lastly, I used above method to calculate the AUC and 95% CI with using "model.best_estimator_" to use the best parameters for prediction:

```
ann_probs2 = mlp2.best_estimator_.predict_proba(gu_Xtests)[:,1]
print(roc_auc_score(gu_ytest, ann_probs2))
```

```
0.759576771120853
```

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.759576771120853
AUC COV: 0.0005295762772071846
95% AUC CI: [0.71447305 0.80468049]
```

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:

```
seed(42)

params8 = {'n_estimators' : [10,100,150],
           'min_samples_leaf': [1,2,3]}
rf2 = RandomForestClassifier(random_state=42)
rf2 = GridSearchCV(rf2, cv=5, param_grid=params8, scoring = 'roc_auc',refit = True,
                   n_jobs=-1, verbose = 5, return_train_score=True)

rf2.fit(gu_Xtrain,gu_ytrain)
rf2.cv_results_
```

I used above method to calculate the AUC and 95% CI with using "model.best_estimator_" to use the best parameters for prediction:

```
rf_probs2 = rf2.best_estimator_.predict_proba(gu_Xtest)[:,1]
print(roc_auc_score(gu_ytest, rf_probs2))
```

```
0.885331312803305
```

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.885331312803305
AUC COV: 0.0002835084573262921
95% AUC CI: [0.85233001 0.91833262]
```

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:

```
seed(10)

params9 = {'learning_rate' : [0.05,0.1,0.2],
           'n_estimators': [100,150],
           'min_samples_split': [2,3,4]}
gbm2 = GradientBoostingClassifier(random_state=10)
gbm2 = GridSearchCV(gbm2, cv=5, param_grid=params9, scoring = 'roc_auc',refit = True,
                    n_jobs=-1, verbose = 5, return_train_score=True)

gbm2.fit(gu_Xtrain,gu_ytrain)
gbm2.cv_results_
```

I used above method to calculate the AUC and 95% CI with using "model.best_estimator_" to apply the best parameters for prediction:

```
gbm_probs2 = gbm2.best_estimator_.predict_proba(gu_Xtest)[:,1]
print(roc_auc_score(gu_ytest, gbm_probs2))
```

```
0.8720806047157728
```

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.8720806047157728
AUC COV: 0.000262844614256567
95% AUC CI: [0.84030472 0.90385649]
```

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.

```
scaler = StandardScaler()
# Fit only to the training data
scaler = scaler.fit(gu_Xtrain)
gu_Xtrains = scaler.transform(gu_Xtrain)
gu_Xtests = scaler.transform(gu_Xtest)
```

from sklearn.svm import SVC to fit a Support Vector Machine Classifier model. Then I used GridSearchCV to tune parameters with 5 cross-validations.

```
from sklearn.svm import SVC
seed(0)
params0 = {'degree' : [2,3,4],
           'C': [0.5,1.0,1.5,2.0,2.5],
           'tol':[1e-5,1e-3,1e-1],
           'max_iter':[-1,-2,-3]}
svc = SVC(random_state=0,probability=True)
svc = GridSearchCV(svc, cv=5, param_grid=params0, scoring = 'roc_auc',refit = True,
                   n_jobs=-1, verbose = 5, return_train_score=True)

svc.fit(gu_Xtrains, gu_ytrain)
svc.cv_results_
```

Lastly, I used above method to calculate the AUC and 95% CI with using "model.best_estimator_" to use the best parameters for prediction:

```
svc_probs = svc.best_estimator_.predict_proba(gu_Xtests)[:,1]
print(roc_auc_score(gu_ytest, svc_probs))
```

```
0.7897854990889575
```

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.7897854990889575
AUC COV: 0.0005276809118318064
95% AUC CI: [0.74476257 0.83480843]
```