Statistical Learning Week 5 - LDA, QDA, and SVC

Jonathan Gragg: East Section

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, confusion
_matrix, accuracy_score,\
    recall_score, precision_score, roc_curve, roc_auc_score, precision_recall_
curve, classification_report
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA, Q
uadraticDiscriminantAnalysis as QDA
from sklearn.svm import SVC
from sklearn.datasets import load_digits
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings(action='ignore')
```

1. Load the the Heart Disease dataset. Print the first few rows. (5 pts)

In [ ]:
```
heart = pd.read_csv('heart.csv')
heart.head()
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	e
0	63	male	typical_angina	145	233	higher_than_120	left_vent_hypertrophy	150	
1	67	male	asymptomatic	160	286	lower_than_120	left_vent_hypertrophy	108	
2	67	male	asymptomatic	120	229	lower_than_120	left_vent_hypertrophy	129	
3	37	male	non_anginal_pain	130	250	lower_than_120	normal	187	
4	41	female	atypical_angina	130	204	lower_than_120	left_vent_hypertrophy	172	

2. Transform the data for modeling. (10 pts)

- Create a data frame with all of the variables.
- Drop any observations with missing values from the dataset.
- Transform the categorical variables to dummy variables (dropping one of the levels for each variable).
- Print the first few rows of this new data frame.

In [ ]:
```
data = heart.copy()
data = pd.get_dummies(data, drop_first=True)
data = data.dropna()
data.head()
```

Out[ ]:

	age	trestbps	chol	thalach	oldpeak	ca	hd	sex_male	cp_atypical_angina	cp_non_anginal
0	63	145	233	150	2.3	0.0	0	1	0	
1	67	160	286	108	1.5	3.0	1	1	0	
2	67	120	229	129	2.6	2.0	1	1	0	
3	37	130	250	187	3.5	0.0	0	1	0	
4	41	130	204	172	1.4	0.0	0	0	1	

3. Create training testing sets. (10 pts)

- Create a feature matrix and response (target) vector for heart disease, and store these as numpy arrays.
- Split the data into training and test sets using a 70%/30% split, stratifying on heart disease.
- Standardize the training data and apply the transformation to the test data. (Standardizing the dummy variables is optional)
- Print the dimensions of the feature matrices and response vectors for both sets.

In [ ]:
```
y = data.hd.values.reshape(-1,1)
X = data.drop('hd', axis=1).values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, strati
fy=y, random_state=1)

num_columns = [0,1,2,3,4,5]
scaler = StandardScaler(with_mean=0,with_std=1)
scaler.fit(X_train[:,num_columns])
X_train[:, num_columns] = scaler.transform(X_train[:,num_columns])

scaler = StandardScaler(with_mean=0,with_std=1)
scaler.fit(X_test[:,num_columns])
X_test[:, num_columns] = scaler.transform(X_test[:,num_columns])

print('X_train deminsions:',X_train.shape)
print('X_test deminsions:',X_test.shape)
print('y_train deminsions:',y_train.shape)
print('y_test deminsions:',y_test.shape)
```

```
X_train deminsions: (209, 18)
X_test deminsions: (90, 18)
y_train deminsions: (209, 1)
y_test deminsions: (90, 1)
```

4. Fit a support vector classifier (SVM with a linear kernel) to the training data. Use cross validation to choose C based on the highest AUC ROC. Calculate recall, precision, and AUC ROC on both the training and test sets. (20 pts)

In [ ]:
```
ln = SVC(probability=True,kernel='linear')

cs=[0.0001,0.001,0.1, 1, 10, 100, 1000]

cv = GridSearchCV(ln, param_grid={'C':cs}, cv=5, scoring='roc_auc')
cv.fit(X_train,y_train)

preds_0 = cv.predict(X_train)
probs_0 = cv.predict_proba(X_train)[:, 1]

preds_1 = cv.predict(X_test)
probs_1 = cv.predict_proba(X_test)[:, 1]

print('TRAIN')
print('Precision: ', precision_score(y_train, preds_0).round(3))
print('Recall: ', recall_score(y_train, preds_0).round(3))
print('AUC ROC: ', roc_auc_score(y_train,probs_0).round(3))

print('TEST')
print('Precision: ', precision_score(y_test, preds_1).round(3))
print('Recall: ', recall_score(y_test, preds_1).round(3))
print('AUC ROC: ', roc_auc_score(y_test,probs_1).round(3))
```

```
TRAIN
Precision:  0.92
Recall:  0.844
AUC ROC:  0.926
TEST
Precision:  0.72
Recall:  0.857
AUC ROC:  0.89
```

5. Fit an SVM model with radial basis kernel to the training data. Use cross validation to choose C based on the highest AUC ROC. Calculate recall, precision, and AUC ROC on both the training and test sets. (20 pts)

In [ ]:
```
rbf = SVC(probability=True,kernel='rbf')

cs=[0.0001,0.001,0.1, 1, 10, 100, 1000]

cv = GridSearchCV(rbf, param_grid={'C':cs}, cv=5, scoring='roc_auc')
cv.fit(X_train,y_train)

preds_0 = cv.predict(X_train)
probs_0 = cv.predict_proba(X_train)[:, 1]

preds_1 = cv.predict(X_test)
probs_1 = cv.predict_proba(X_test)[:, 1]

print('TRAIN')
print('Precision: ', precision_score(y_train, preds_0).round(3))
print('Recall: ', recall_score(y_train, preds_0).round(3))
print('AUC ROC: ', roc_auc_score(y_train,probs_0).round(3))

print('TEST')
print('Precision: ', precision_score(y_test, preds_1).round(3))
print('Recall: ', recall_score(y_test, preds_1).round(3))
print('AUC ROC: ', roc_auc_score(y_test,probs_1).round(3))
```

```
TRAIN
Precision:  0.907
Recall:  0.812
AUC ROC:  0.92
TEST
Precision:  0.825
Recall:  0.786
AUC ROC:  0.908
```

6. Fit a model using Linear Discriminant Analysis (LDA) to the training data. Calculate recall, precision, and AUC ROC on both the training and test sets. (15 pts)

In [ ]:
```
lda = LDA().fit(X_train, y_train)

preds = lda.predict(X_train)
probs = lda.predict_proba(X_train)[:, 1]

print('TRAIN')
print('Precision: ', precision_score(y_train, preds).round(3))
print('Recall: ', recall_score(y_train, preds).round(3))
print('AUC ROC: ', roc_auc_score(y_train,probs).round(3))

preds = lda.predict(X_test)
probs = lda.predict_proba(X_test)[:, 1]

print('TEST')
print('Precision: ', precision_score(y_test, preds).round(3))
print('Recall: ', recall_score(y_test, preds).round(3))
print('AUC ROC: ', roc_auc_score(y_test,probs).round(3))
```

```
TRAIN
Precision:  0.888
Recall:  0.823
AUC ROC:  0.935
TEST
Precision:  0.75
Recall:  0.786
AUC ROC:  0.89
```

7. Fit a model using Quadratic Discriminant Analysis (QDA) to the training data. Calculate recall, precision, and AUC ROC on both the training and test sets. (15 pts)

In [ ]:
```
qda = QDA().fit(X_train, y_train)

preds = qda.predict(X_train)
probs = qda.predict_proba(X_train)[:, 1]

print('TRAIN')
print('Precision: ', precision_score(y_train, preds).round(3))
print('Recall: ', recall_score(y_train, preds).round(3))
print('AUC ROC: ', roc_auc_score(y_train,probs).round(3))

preds = qda.predict(X_test)
probs = qda.predict_proba(X_test)[:, 1]

print('TEST')
print('Precision: ', precision_score(y_test, preds).round(3))
print('Recall: ', recall_score(y_test, preds).round(3))
print('AUC ROC: ', roc_auc_score(y_test,probs).round(3))
```

```
TRAIN
Precision:  0.641
Recall:  0.615
AUC ROC:  0.757
TEST
Precision:  0.642
Recall:  0.81
AUC ROC:  0.741
```

adding some regularization to the QDA model to see if that improves performance

In [ ]:
```
qda = QDA()

reg = np.arange(0,10,0.1)

qcv = GridSearchCV(qda, param_grid={'reg_param':reg}, cv=5, scoring='roc_auc')
qcv.fit(X_train,y_train)

preds = qcv.predict(X_train)
probs = qcv.predict_proba(X_train)[:, 1]

print('TRAIN')
print('Precision: ', precision_score(y_train, preds).round(3))
print('Recall: ', recall_score(y_train, preds).round(3))
print('AUC ROC: ', roc_auc_score(y_train,probs).round(3))

preds = qcv.predict(X_test)
probs = qcv.predict_proba(X_test)[:, 1]

print('TEST')
print('Precision: ', precision_score(y_test, preds).round(3))
print('Recall: ', recall_score(y_test, preds).round(3))
print('AUC ROC: ', roc_auc_score(y_test,probs).round(3))
```

```
TRAIN
Precision:  0.909
Recall:  0.833
AUC ROC:  0.94
TEST
Precision:  0.773
Recall:  0.81
AUC ROC:  0.897
```

8. Write a few sentences comparing the performance of the models fit in questions (4) - (7). (5 pts)

When evaluating the SVC models, linear performed better on the training set but rbf performed better on the test. Makes me think linear could have more risk of overfitting to the training data than rbf. Both SVC models performed better than LDA and QDA models. The worst performing model was QDA before regularization, my guess is there is too many variables for QDA to perform well. Once I applied regularization and cross validateed it to get the best regularization with respect to auc, I was able to get a model that was actually better than the linear model, but slightly below SVC models.