02_LDA_classification

December 7, 2018

1 Linear Discriminant Analysis for CRC Prediction

Expleriments with linear discriminant analysis as a classifier.

```
In [23]: import pandas as pd
         import matplotlib.pyplot as plt
         import numpy as np

         from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
         from sklearn.metrics import accuracy_score, balanced_accuracy_score, make_scorer
         from sklearn.metrics import roc_auc_score, roc_curve
         from sklearn.model_selection import train_test_split, RandomizedSearchCV
         from sklearn.model_selection import cross_val_score

         from IPython.display import set_matplotlib_formats
         set_matplotlib_formats('pdf')
```

```
In [24]: # Get features
         samples = pd.read_csv("samples.csv", index_col=0)
         microbes = pd.read_csv("microbes.csv", index_col=0)
         metabolites = pd.read_csv("metabolites.csv", index_col=0)
         combined_features = pd.concat([microbes, metabolites], axis=1)
         # Label vector
         labels = samples.case
```

1.1 Microbes Data

Split into training and testing.

```
In [25]: # We'll use 80% for training, and 20% for testing
         X_train, X_test, labels_train, labels_test = train_test_split(
             microbes, labels, test_size=0.2, random_state=42)
```

We also set up a 5x cross-validation on the training set, in order to tune the parameters of the model:

```
In [26]: # Setup 5x cross-validation, and manually tune the parameters for the best accuracy
         # The possible parameters are:
```

```
        #     solver: 'svd', 'eigen', 'lsqr'
        #     shrinkage: None, 'auto', 0 < n < 1

        lda = LDA(solver='lsqr', shrinkage='auto')
        scores = cross_val_score(lda, X_train, labels_train, cv=5,
                                 scoring=make_scorer(roc_auc_score))
        print("Training AUC Score: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

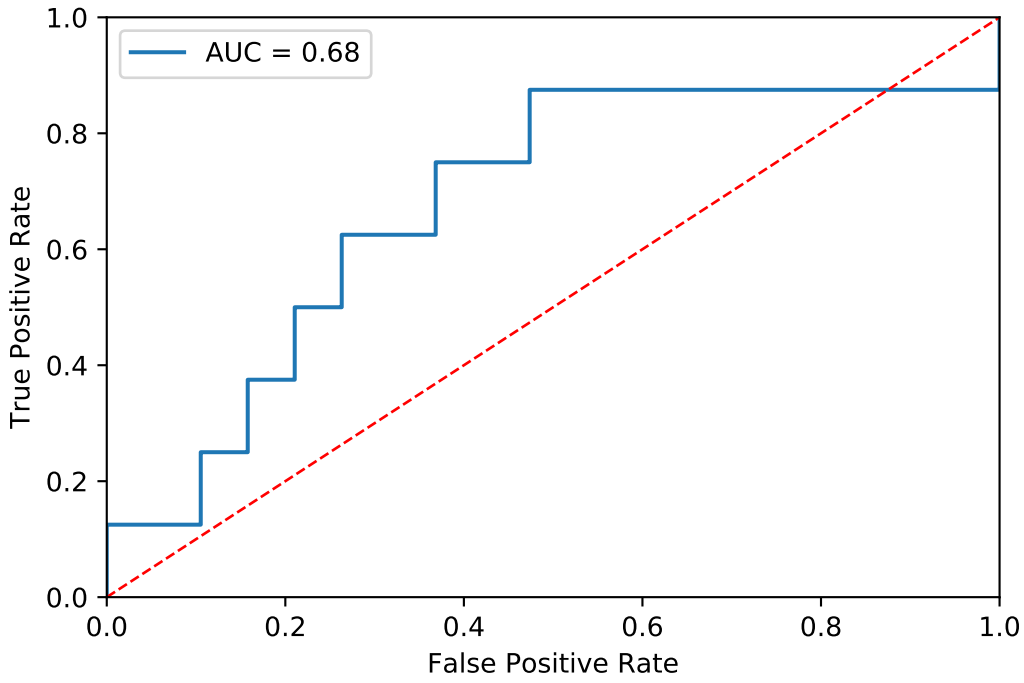
```
Training AUC Score: 0.59 (+/- 0.14)
```

The best model used the 'lsqr' solver 'auto' shrinkage.
Predicting on the testing set:

```
In [27]: # Train the best model with the entire training set
         lda.fit(X_train, labels_train)
         pred = lda.predict(X_test)
         balanced_accuracy = balanced_accuracy_score(labels_test, pred)
         accuracy = accuracy_score(labels_test, pred)
         auc = roc_auc_score(y_true=labels_test, y_score=lda.decision_function(X_test))
         print("Accuracy_score:", accuracy)
         print("Balanced accuracy score:", balanced_accuracy)
         print("AUC score:", auc)
```

```
Accuracy_score: 0.666666666667
Balanced accuracy score: 0.618421052632
AUC score: 0.677631578947
```

```
In [28]: # Plot an ROC curve
         fpr, tpr, _ = roc_curve(y_true=labels_test, y_score=lda.decision_function(X_test))
         plt.plot(fpr, tpr, label="AUC = {0:.2f}".format(auc))
         plt.plot([0, 1], [0, 1], color='red', lw=1, linestyle='--')
         plt.xlim([0.0, 1.0])
         plt.ylim([0.0, 1.0])
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.legend()
         plt.show()
```



In [29]:
```
# Important variables, based on LDA weights
weights = list(enumerate(np.abs(lda.coef_[0])))
sorted_weights = sorted(weights, key=lambda x: x[1], reverse=True)
# Get top 30 variables
top_variables = [i for i, j in sorted_weights][:20]
[microbes.columns[i] for i in top_variables]
```

Out[29]:
```
['Root;k__Bacteria;p__Chloroflexi',
 'Root;k__Bacteria;p__Proteobacteria;c__Gammaproteobacteria;o__Alteromonadales',
 'Root;k__Bacteria;p__Bacteroidetes;c__Bacteroidia;o__Bacteroidales;f__[Odoribacterac
 'Root;k__Bacteria;p__Cyanobacteria',
 'Root;k__Bacteria;p__Fusobacteria;c__Fusobacteria;o__Fusobacteriales;f__Fusobacteria
 'Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae;g_
 'Root;k__Bacteria;p__Firmicutes;c__Bacilli;o__Bacillales;f__Planococcaceae',
 'Root;k__Bacteria;p__Firmicutes;c__Bacilli;o__Bacillales;f__Planococcaceae;g__',
 'Root;k__Bacteria;p__Synergistetes;c__Synergistia;o__Synergistales;f__Synergistaceae
 'Root;k__Bacteria;p__Synergistetes;c__Synergistia',
 'Root;k__Bacteria;p__Synergistetes;c__Synergistia;o__Synergistales',
 'Root;k__Bacteria;p__Synergistetes',
 'Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Ruminococcaceae;g_
 'Root;k__Bacteria;p__Proteobacteria;c__Alphaproteobacteria;o__Rhodobacterales',
 'Root;k__Bacteria;p__Proteobacteria;c__Alphaproteobacteria;o__Rhodobacterales;f__Rho
 'Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Peptococcaceae;g__
 'Root;k__Archaea;p__Euryarchaeota',
 'Root;k__Bacteria;p__Acidobacteria',
```

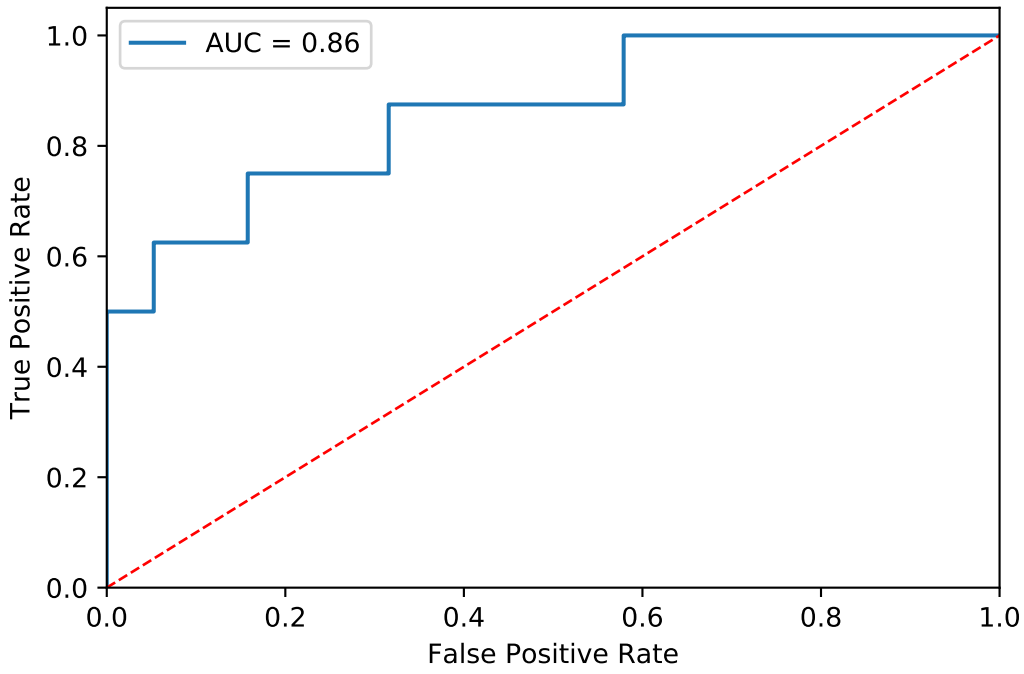
```
 'Root;k__Archaea;p__Euryarchaeota;c__[Parvarchaea];o__YLA114',
 'Root;k__Archaea;p__Euryarchaeota;c__[Parvarchaea];o__YLA114;f__;g__']
```

1.2 Metabolites Data

```
In [30]: X_train, X_test, labels_train, labels_test = train_test_split(
             metabolites, labels, test_size=0.2, random_state=42)

In [31]: # Setup 5x cross-validation, and manually tune the parameters for the best accuracy
         # The possible parameters are:
         #     solver: 'svd', 'eigen', 'lsqr'
         #     shrinkage: None, 'auto', 0 < n < 1

         lda = LDA(solver='lsqr', shrinkage='auto')
         scores = cross_val_score(lda, X_train, labels_train, cv=5,
                                  scoring=make_scorer(roc_auc_score))
         print("Training AUC Score: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

Training AUC Score: 0.69 (+/- 0.12)


In [32]: # Train the best model with the entire training set
         lda.fit(X_train, labels_train)
         pred = lda.predict(X_test)
         balanced_accuracy = balanced_accuracy_score(labels_test, pred)
         accuracy = accuracy_score(labels_test, pred)
         auc = roc_auc_score(y_true=labels_test, y_score=lda.decision_function(X_test))
         print("Accuracy_score:", accuracy)
         print("Balanced accuracy score:", balanced_accuracy)
         print("AUC score:", auc)

Accuracy_score: 0.814814814815
Balanced accuracy score: 0.759868421053
AUC score: 0.861842105263


In [33]: # Important variables, based on LDA weights
         weights = list(enumerate(np.abs(lda.coef_[0])))
         sorted_weights = sorted(weights, key=lambda x: x[1], reverse=True)
         # Get top 30 variables
         top_variables = [i for i, j in sorted_weights][:20]
         [metabolites.columns[i] for i in top_variables]

Out[33]: ['X_15528',
          'ALLO_THREONINE',
          'HEXADECANEDIOATE',
          'N_ACETYLGLUTAMINE',
          'LEVULINATE__4_OXOVALERATE_',
          '_4_HYDROXYBENZOATE',
```

```
 'X_14473',
 'PROLINE',
 'ISOPALMITIC_ACID',
 'N6_ACETYLLYSINE',
 'MANNOSE',
 '_4_ACETAMIDOPHENOL',
 'PROLYLALANINE',
 'PALMITOYL_SPHINGOMYELIN',
 '_2_HYDROXYMYRISTATE',
 'HOMOCYSTEINE',
 'GLUTAMINE',
 'N_ACETYLNEURAMINATE',
 'LEUCYLTRYPTOPHAN',
 'HISTIDINE']
```

In [35]:
```
# Plot an ROC curve
fpr, tpr, _ = roc_curve(y_true=labels_test, y_score=lda.decision_function(X_test))
plt.plot(fpr, tpr, label="AUC = {0:.2f}".format(auc))
plt.plot([0, 1], [0, 1], color='red', lw=1, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```



1.3 Combined Data

In [36]:
```
X_train, X_test, labels_train, labels_test = train_test_split(
    combined_features, labels, test_size=0.2, random_state=42)
```

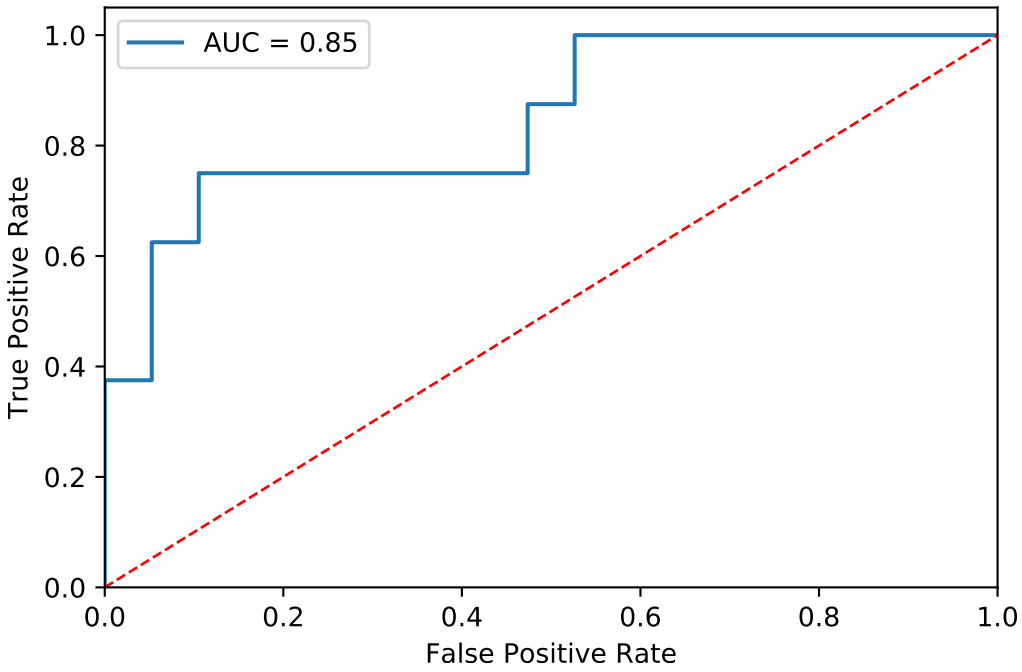
In [37]:
```
# Setup 5x cross-validation, and manually tune the parameters for the best accuracy
# The possible parameters are:
#     solver: 'svd', 'eigen', 'lsqr'
#     shrinkage: None, 'auto', 0 < n < 1

lda = LDA(solver='lsqr', shrinkage=.7)
scores = cross_val_score(lda, X_train, labels_train, cv=5,
                         scoring=make_scorer(roc_auc_score))
print("Training AUC Score: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Training AUC Score: 0.65 (+/- 0.10)
```

In [38]:
```
# Train the best model with the entire training set
lda.fit(X_train, labels_train)
pred = lda.predict(X_test)
balanced_accuracy = balanced_accuracy_score(labels_test, pred)
accuracy = accuracy_score(labels_test, pred)
auc = roc_auc_score(y_true=labels_test, y_score=lda.decision_function(X_test))
print("Accuracy_score:", accuracy)
print("Balanced accuracy score:", balanced_accuracy)
print("AUC score:", auc)
```

```
Accuracy_score: 0.814814814815
Balanced accuracy score: 0.723684210526
AUC score: 0.848684210526
```

In [39]:
```
# Plot an ROC curve
fpr, tpr, _ = roc_curve(y_true=labels_test, y_score=lda.decision_function(X_test))
plt.plot(fpr, tpr, label="AUC = {0:.2f}".format(auc))
plt.plot([0, 1], [0, 1], color='red', lw=1, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```



Overall, the performance of the metabolites data alone was the best, with an AUC score of 0.86. It is possible that performing some variable selection to eliminate unimportant variables would improve the model performance.