Notes

February 21, 2025

1 Notes

1.1 Import notebook funcs

[1]:

```
from notebookfuncs import *
```

1.2 ROC

1.2.1 What is the ROC Curve?

The ROC curve stands for Receiver Operating Characteristics Curve and is an evaluation metric for classification tasks and it is a probability curve that plots sensitivity and specificity. So, we can say that the ROC Curve can also be defined as the evaluation metric that plots the sensitivity against the false positive rate. The ROC curve plots two different parameters given below:

1. True positive rate
2. False positive rate

The ROC Curve can also defined as a graphical representation that shows the performance or behavior of a classification model at all different threshold levels. The ROC Curve is a tool used for binary classification in machine learning. While learning about the ROC Curve we need to be familiar with the terms specificity and sensitivity.

- Specificity: It is defined as the proportion of negative instances that were predicted correctly as negative values. In other terms, the true negative is also called the specificity. The false positive rate can be found using the specificity by subtracting one from it.
- Sensitivity: The true positive rate is defined as the rate of positive instances that were predicted correctly to be positive. The true positive rate is a synonym for "True positive rate".The sensitivity is also called recall and these terms are often interchangeable. The formula for TPR is as follows,

TPR = TP / (TP + FN)

where, TPR = True positive rate, TP = True positive, FN = False negative.

False positive rate: On the other side, false positive rate can be defined as the rate of negative instances that were predicted incorrectly to be positive. In other terms, the false positive can also be called "1 - specificity".

FPR = FP / (FP + TN)

where, FPR = False positive rate, FP = False positive, TN = True negative.

The ROC Curve is often comparable with the precision and recall curve but it is different because it plots the true positive rate (which is also called recall) against the false positive rate.

The curve is plotted by finding the values of TPR and FPR at distinct threshold values and we don't plot the probabilities but we plot the scores. So the probability of the positive class is taken as the score here.

1.2.2 Types of ROC Curve

There are two types of ROC Curves:

1. Parametric ROC Curve: The parametric method plots the curve using maximum likelihood estimation. This type of ROC Curve is also smooth and plots any sensitivity and specificity, but it has drawbacks like actual data can be discarded. The computation of this method is complex.

2. Non-Parametric ROC Curve: The non-parametric method does not need any assumptions about the data distributions. It gives unbiased estimates and plot passes through all the data points. The computation of this method is simple.

1.3 ROC Curve in Python

[2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import roc_curve, auc
from ISLP import load_data
Default = load_data('Default')
Default.columns
```

[2]:
```
Index(['default', 'student', 'balance', 'income'], dtype='object')
```

[3]:
```python
Default.shape
```

[3]:
```
(10000, 4)
```

[4]:
```python
Default.describe()
```

[4]:
```
            balance        income
count  10000.000000  10000.000000
mean     835.374886  33516.981876
std      483.714985  13336.639563
min        0.000000    771.967729
25%      481.731105  21340.462903
50%      823.636973  34552.644802
75%     1166.308386  43807.729272
max     2654.322576  73554.233495
```

```
[5]: Default["student"] = Default["student"].astype("category")
```

```
[6]: Default = pd.get_dummies(Default,columns=["student"],drop_first=True)
     Default["student"] = Default["student_Yes"]
```

```
[7]: X = Default[["balance", "income","student"]]
```

```
[7]:           balance        income  student
     0      729.526495  44361.625074    False
     1      817.180407  12106.134700     True
     2     1073.549164  31767.138947    False
     3      529.250605  35704.493935    False
     4      785.655883  38463.495879    False
     …             …           …      …
     9995   711.555020  52992.378914    False
     9996   757.962918  19660.721768    False
     9997   845.411989  58636.156984    False
     9998  1569.009053  36669.112365    False
     9999   200.922183  16862.952321     True

     [10000 rows x 3 columns]
```

```
[8]:  y= Default["default"]
```

```
[8]: 0       No
     1       No
     2       No
     3       No
     4       No
             ..
     9995    No
     9996    No
     9997    No
     9998    No
     9999    No
     Name: default, Length: 10000, dtype: category
     Categories (2, object): ['No', 'Yes']
```

```
[10]: # Initializing the LDA estimator
      lda = LinearDiscriminantAnalysis()
      # Performing LDA
      lda.fit(X, y)
      lda.predict(X)
      lda.score(X,y)
```

```
[10]: 0.9724
```

[11]:

```
# Null error rate
sum(Default["default"] == "No") / len(Default)    # 0.9667
```

[11]:

```
0.9667
```

[12]:

```
### Count of defaults versus non-defaults
```

[13]:

```
print(sum(Default["default"] == "No"),
sum(Default["default"] == "Yes"))
```

```
9667 333
```

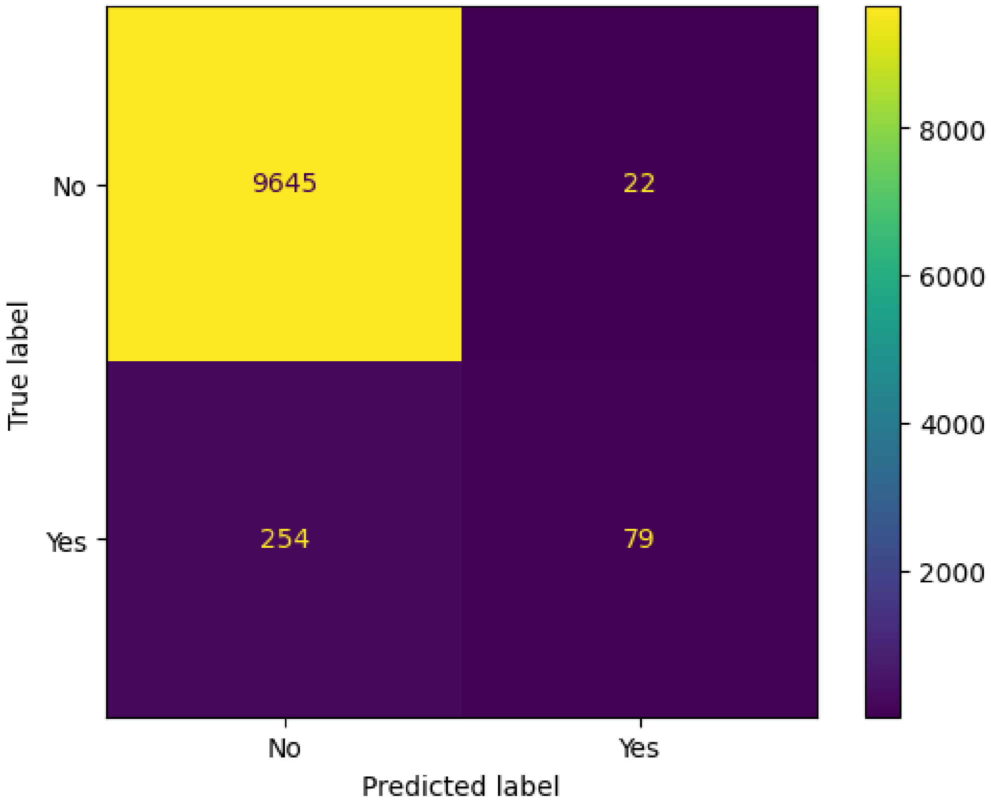
[14]:

```
from sklearn.metrics import confusion_matrix

# Assigning predicted y values
y_pred = lda.predict(X)

# Creating confusion matrix
cm = confusion_matrix(y_true=y, y_pred=y_pred)
print(cm)

# Getting individual values
tn, fp, fn, tp = confusion_matrix(y, y_pred).ravel()

tn = sum(1 for i, j in zip(y, y_pred) if i == "No" and j == "No")
tp = sum(1 for i, j in zip(y, y_pred) if i == "Yes" and j == "Yes")
fp = sum(1 for i, j in zip(y, y_pred) if i == "No" and j == "Yes")
fn = sum(1 for i, j in zip(y, y_pred) if i == "Yes" and j == "No")
print(tn,fp,fn,tp)
```

```
[[9645   22]
 [ 254   79]]
9645 22 254 79
```

[15]:

```
from sklearn.metrics import ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                      display_labels=["No", "Yes"])
disp.plot();
```



1.3.1 Accuracy

Accuracy is defined as

(number of true negatives + number of true positives) / (number of true negatives + number of true positives + number of false negatives + number of false positives)

```
[16]: Accuracy = (tn + tp) / (tn + tp + fn + fp)
```

```
[16]: 0.9724
```

1.3.2 Precision

Precision is defined as

(number of true positives) / (number of true positives + number of false positives)

= TP / P*

where P* is the total number of predicted positives (defaults) in the dataset.

```
[17]: Precision = tp / (tp + fp)
```

```
[17]: 0.7821782178217822
```

Intuitively, precision is representing the proportion of all our predicted positive values that are actually positive. When comparing different classification models, precision is a good measure when we want to avoid false positives.

For example, when detecting spam emails, a model with high precision is likely preferred. In the case of spam email, the email user would much rather get the occasional spam email (a false negative) than miss an important email that wasn't spam (a false positive).

1.3.3 Recall

Recall is defined as

(number of true positives) / (number of true positives + number of false negatives) = TP / P

where P is actual positives (defaults) in the dataset.

```
[18]: Recall = tp / (tp + fn)
```

```
[18]: 0.23723723723723725
```

Intuitively, recall is representing how well a model is classifying positive observations as actually positive. When comparing different classification models, recall is a good measure when we want to avoid false negatives.

Recall is a metric of particular interest. When classifying events as anomalous or not, we would much rather classify a non-anomalous event as anomalous (a false positive), than misclassify an actual anomaly as non-anomalous (a false negative). Said another way, out of all the actual anomalies out there, we want to make sure we detect as many as we can, even at the expense of including some false positives.

In the medical setting, recall is more commonly referred to as sensitivity. A related term in the medical literature is specificity, which is equivalent to the true negative rate. Occasionally, specificity is also referred to as recall of the negative class.

1.3.4 F_1 score

The F_1 score is the harmonic mean of precision and recall, and is defined as

$$2 * \frac{precision\ *\ recall}{precision\ +\ recall}$$

When comparing models, the F_1 score is useful when we want to strike a balance between precision and recall. That is, when we want to avoid both false positives (as in spam email classification) and false negatives (as in anomaly detection).

```
[19]: F1 = 2 * Precision * Recall / (Precision + Recall)
```

```
[19]: 0.3640552995391705
```

1.3.5 scikit-learn's Classification Report

[20]:
```
from sklearn.metrics import classification_report
print(classification_report(y_true=y,
                                y_pred=y_pred,
                                target_names=["No", "Yes"]))
```

```
              precision    recall  f1-score   support

          No       0.97      1.00      0.99      9667
         Yes       0.78      0.24      0.36       333

    accuracy                           0.97     10000
   macro avg       0.88      0.62      0.67     10000
weighted avg       0.97      0.97      0.97     10000
```

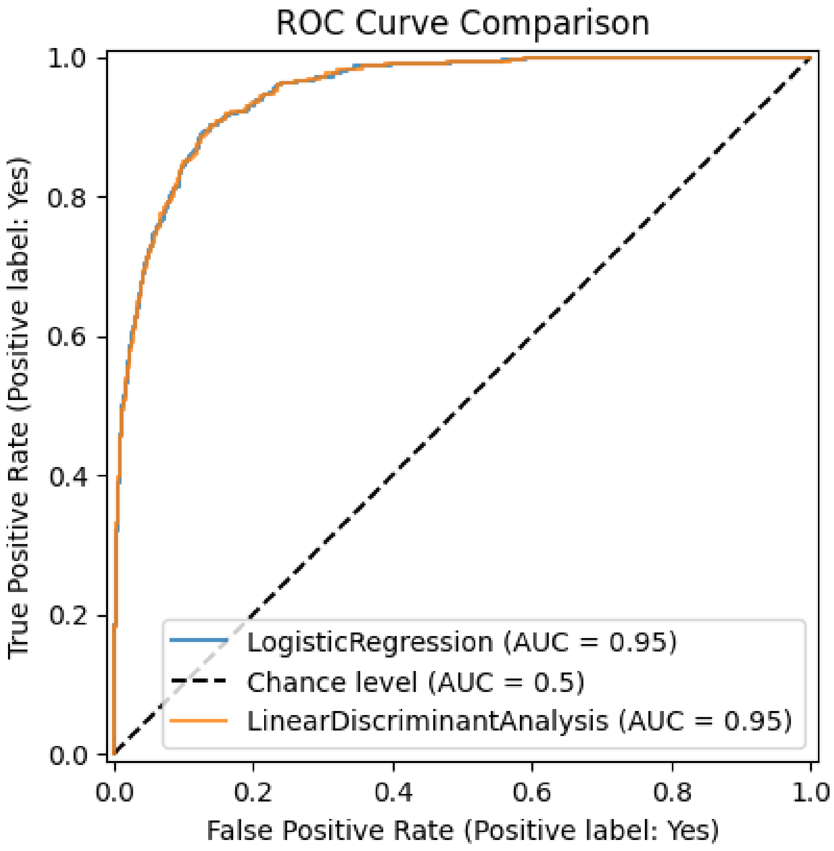
Note that our manually calculated values of precision, recall, and the F_1 score align with the Yes row above. For completeness, support simply refers to the number of observations in each class. For our working example, that is number of observations that did not default (9667) or that did default (333) present in our training data.

1.4 Train Logistic Regression Model

[21]:
```
from sklearn.linear_model import LogisticRegression

# Setting penalty=None to disable regularization
log_reg = LogisticRegression(penalty=None)
log_reg.fit(X, y)
log_reg.score(X, y)  # 0.9732
```

[21]:
```
0.9732
```

[22]:
```
from sklearn.metrics import RocCurveDisplay
lr_disp = RocCurveDisplay.from_estimator(log_reg,X,y,alpha=0.
 ↪8,plot_chance_level=True)
RocCurveDisplay.from_estimator(lda,X,y,ax=lr_disp.ax_,alpha=0.8)
plt.title("ROC Curve Comparison")
plt.show()
```



1.5 References:

1. Linear discriminant analysis #2 scikit-learn, precision, recall, F-scores, ROC curves, and a comparison to logistic regression

[23]:
```
allDone();
```

```
<IPython.lib.display.Audio object>
```