COMP47750 Tutorial
Machine Learning with Python

Evaluation

Pádraig Cunningham
Based on slides by Derek Greene

School of Computer Science

Tutorial Q1

- The confusion matrix below shows the evaluation results for a binary classifier applied to a set of 768 test examples, which are annotated with the class labels (Pass, Fail). Calculate:

  a) The precision score for both of the classes.

  b) The recall score for both of the classes.

  c) The F1-measure score for both of the classes.

  d) The overall classification accuracy for all the data.

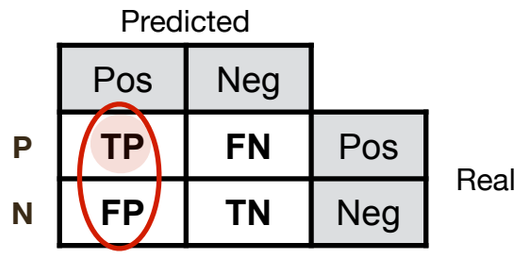
	Predicted Class: Pass	Predicted Class: Fail	
Real Class: Pass	**TP** True Positive	**FN** False Negative	Pass
Real Class: Fail	**FP** False Positive	**TN** True Negative	Fail

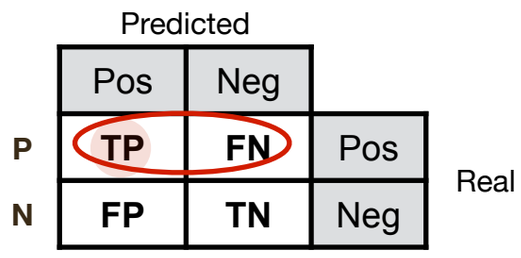
	Predicted Class: Pass	Predicted Class: Fail	
Real Class: Pass	407	93	Pass
Real Class: Fail	108	160	Fail

Tutorial Q1(a,b)

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \text{Sensitivity}$$





Note: These measures are always relative to one class!

Predicted Class

Pass	Fail	
407	93	Pass
108	160	Fail

Real Class

Class	Precision	Recall
Pass	407/(407+108) = 0.79	407/(407+93) = 0.814
Fail	160/(93+160) = 0.632	160/(108+160) = 0.597

Tutorial Q1(c)

c) Calculate the F1-measure score for both of the classes.

F1-Measure: harmonic mean of precision and recall

$$\mathrm{F1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Also relative to one class!

Class	Precision	Recall	F1
Pass	407/(407+108) = 0.79	407/(407+93) = 0.814	(2x0.79x0.814)/(0.79+0.814) = 0.802
Fail	160/(93+160) = 0.632	160/(108+160) = 0.597	(2x0.632x0.597)/(0.632+0.597) = 0.614

Tutorial Q1(d)

d) Calculate the overall classification accuracy for all the data.

Accuracy: Number of predictions correct / all predictions

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy score is relative to the overall dataset, often reported as a percentage.

	Predicted Class: Pass	Predicted Class: Fail
Real Class: Pass	407	93
Real Class: Fail	108	160

OVERALL ACCURACY:

(407+160)/(407+93+108+160)

= 73.8281%

Tutorial Q2(a)

a) Calculate the **overall accuracy** for each of the 3 classifiers on this data. Based on your calculations, which classifier the most accurate?

Example	True Class Label	KNN Prediction	J48 Prediction	SVM Prediction
1	spam	spam	spam	spam
2	non-spam	non-spam	spam	non-spam
3	spam	non-spam	non-spam	spam
4	non-spam	non-spam	non-spam	non-spam
5	spam	spam	spam	spam
6	non-spam	non-spam	non-spam	non-spam
7	non-spam	spam	spam	non-spam
8	non-spam	non-spam	spam	spam
9	spam	spam	non-spam	spam
10	spam	spam	non-spam	non-spam
11	spam	non-spam	non-spam	spam
12	spam	spam	spam	spam

Tutorial Q2(a)

Example	True Class Label	KNN Prediction	J48 Prediction	SVM Prediction
1	spam	spam	spam	spam
2	non-spam	non-spam	spam	non-spam
3	spam	non-spam	non-spam	spam
4	non-spam	non-spam	non-spam	non-spam
5	spam	spam	spam	spam
6	non-spam	non-spam	non-spam	non-spam
7	non-spam	spam	spam	non-spam
8	non-spam	non-spam	spam	spam
9	spam	spam	non-spam	spam
10	spam	spam	non-spam	non-spam
11	spam	non-spam	non-spam	spam
12	spam	spam	spam	spam

#Correct	9/12	5/12	10/12
Accuracy	75%	41.7%	83.3%

Overall Accuracy:
Number of predictions correct / all predictions

→ SVM is most accurate

Tutorial Q2(b)

Example	True Class Label	KNN Prediction	J48 Prediction	SVM Prediction
1	spam	spam	spam	spam
2	non-spam	non-spam	spam	non-spam
3	spam	non-spam	non-spam	spam
4	non-spam	non-spam	non-spam	non-spam
5	spam	spam	spam	spam
6	non-spam	non-spam	non-spam	non-spam
7	non-spam	spam	spam	non-spam
8	non-spam	non-spam	spam	spam
9	spam	spam	non-spam	spam
10	spam	spam	non-spam	non-spam
11	spam	non-spam	non-spam	spam
12	spam	spam	spam	spam

"Spam"	**TP**	5	3	6
	FP	1	3	1
	Precision	5/6 = 0.833	3/6 = 0.5	6/7 = 0.857

Precision for spam:
Number of correct spam predictions / all predictions of spam

→SVM has highest precision for spam

Tutorial Q3(a)

- 10-fold cross validation experiment. Dataset of 5000 images, so 500 images in each test set.

a) What is the overall accuracy of the classifier based on the cross-validation results?

Fold	Class: Cats		Class: Dogs		Class: People	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	82	68	82	68	164	36
2	81	69	102	48	176	24
3	99	51	97	53	160	40
4	81	69	102	48	148	52
5	94	56	99	51	148	52
6	97	53	91	59	162	38
7	81	69	94	56	148	52
8	76	74	79	71	181	19
9	76	74	97	53	160	40
10	96	54	79	71	179	21

Tutorial Q3(a)

a) What is the overall accuracy of the classifier based on the cross-validation results?

Fold	**Class: Cats**		**Class: Dogs**		**Class: People**		
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	**Accuracy**
1	82	68	82	68	164	36	65.6%
2	81	69	102	48	176	24	71.8%
3	99	51	97	53	160	40	71.2%
4	81	69	102	48	148	52	66.2%
5	94	56	99	51	148	52	68.2%
6	97	53	91	59	162	38	70.0%
7	81	69	94	56	148	52	64.6%
8	76	74	79	71	181	19	67.2%
9	76	74	97	53	160	40	66.6%
10	96	54	79	71	179	21	70.8%

Fold 1: (82+82+164)/(82+68+82+68+164+36) = 65.6% accuracy for fold …

Tutorial Q3(a)

a) What is the overall accuracy of the classifier based on the cross-validation results?

Fold	**Class: Cats**		**Class: Dogs**		**Class: People**		
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	**Accuracy**
1	82	68	82	68	164	36	65.6%
2	81	69	102	48	176	24	71.8%
3	99	51	97	53	160	40	71.2%
4	81	69	102	48	148	52	66.2%
5	94	56	99	51	148	52	68.2%
6	97	53	91	59	162	38	70.0%
7	81	69	94	56	148	52	64.6%
8	76	74	79	71	181	19	67.2%
9	76	74	97	53	160	40	66.6%
10	96	54	79	71	179	21	70.8%
Mean							**68.2%**

Fold 1: (82+82+164)/(82+68+82+68+164+36) = 65.6% accuracy for fold …

Overall: (65.6% + 71.8% + 71.2% + 66.2% + 68.2% + 70.0% + 64.6% + 67.2% + 66.6% + 70.8%)/10 = 68.2%

Tutorial Q3(b)

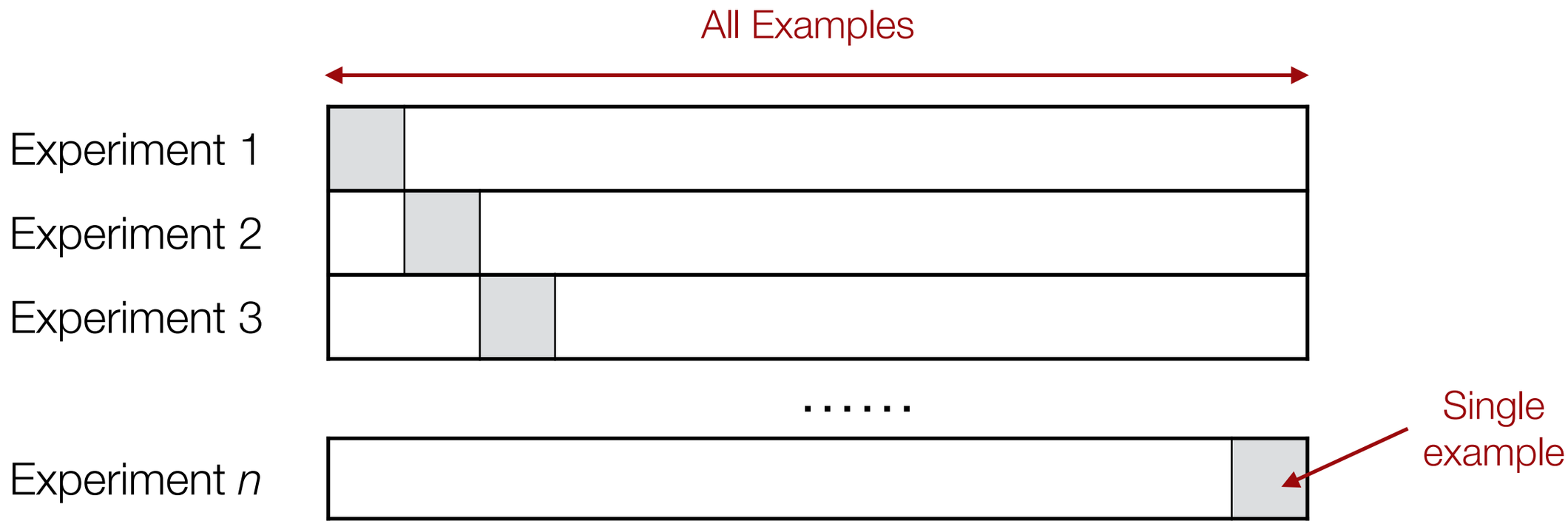
b) What conclusion might be draw about the different classes in the data, based on the results above?

Fold	**Class: Cats**		**Class: Dogs**		**Class: People**		
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	**Accuracy**
1	82	68	82	68	164	36	65.6%
2	81	69	102	48	176	24	71.8%
3	99	51	97	53	160	40	71.2%
4	81	69	102	48	148	52	66.2%
5	94	56	99	51	148	52	68.2%
6	97	53	91	59	162	38	70.0%
7	81	69	94	56	148	52	64.6%
8	76	74	79	71	181	19	67.2%
9	76	74	97	53	160	40	66.6%
10	96	54	79	71	179	21	70.8%
Mean	86.3	63.7	92.2	57.8	162.6	37.4	**68.2%**
Class Acc.	**57.5%**		**61.5%**		**81.3%**		

➡ High accuracy for class "People", low accuracy for "Cats" and "Dogs". Suggests system is poor at distinguishing between these classes.

Tutorial Q3(c)

c) Would “leave-one-out cross validation” be an appropriate evaluation strategy on this dataset?



If each fold has 500 examples, then overall dataset has *n=5000* examples.

So leave-one-out would require running 5000 experiments where 1 example is left out each time.

Could be computationally impractical to do this, so *k*-fold cross validation would be more suitable.

Tutorial Q4

1. The notebook **`09 ROC Exercise`** contains code for loading the diabetes dataset (**`diabetes.csv`**).
    a) Using the code in **`08 ROC`** as a template produce ROC curves for kNN, SVM and Naive Bayes classifiers on the diabetes data.
    b) Repeat this exercise using synthetic data generated using the code below. What insights do these ROC curves provide?

```
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4, n_classes=2
                           class_sep=0.75, random_state=1)
```

Tutorial Q4(a)

```
gnb = GaussianNB()
y_score = gnb.fit(X_train, y_train).predict_proba(X_test)
fprG, tprG, t = roc_curve(y_test, y_score[:,1])
roc_aucG = auc(fprG, tprG)

kNN = KNeighborsClassifier(n_neighbors = 5)
y_score = kNN.fit(X_train, y_train).predict_proba(X_test)
fprN, tprN, t = roc_curve(y_test, y_score[:,1])
roc_aucN = auc(fprN, tprN)

di_svm = SVC(kernel = 'linear',C=1, probability=True)
y_score = di_SVM.fit(X_train, y_train).predict_proba(X_test)
fprS, tprS, t = roc_curve(y_test, y_score[:,1])
roc_aucS = auc(fprS, tprS)

y_score[:5]
Out[54]:
array([[0.06800138, 0.93199862],
       [0.82309978, 0.17690022],
       [0.89492645, 0.10507355],
       [0.37098204, 0.62901796],
       [0.86324697, 0.13675303]])
```

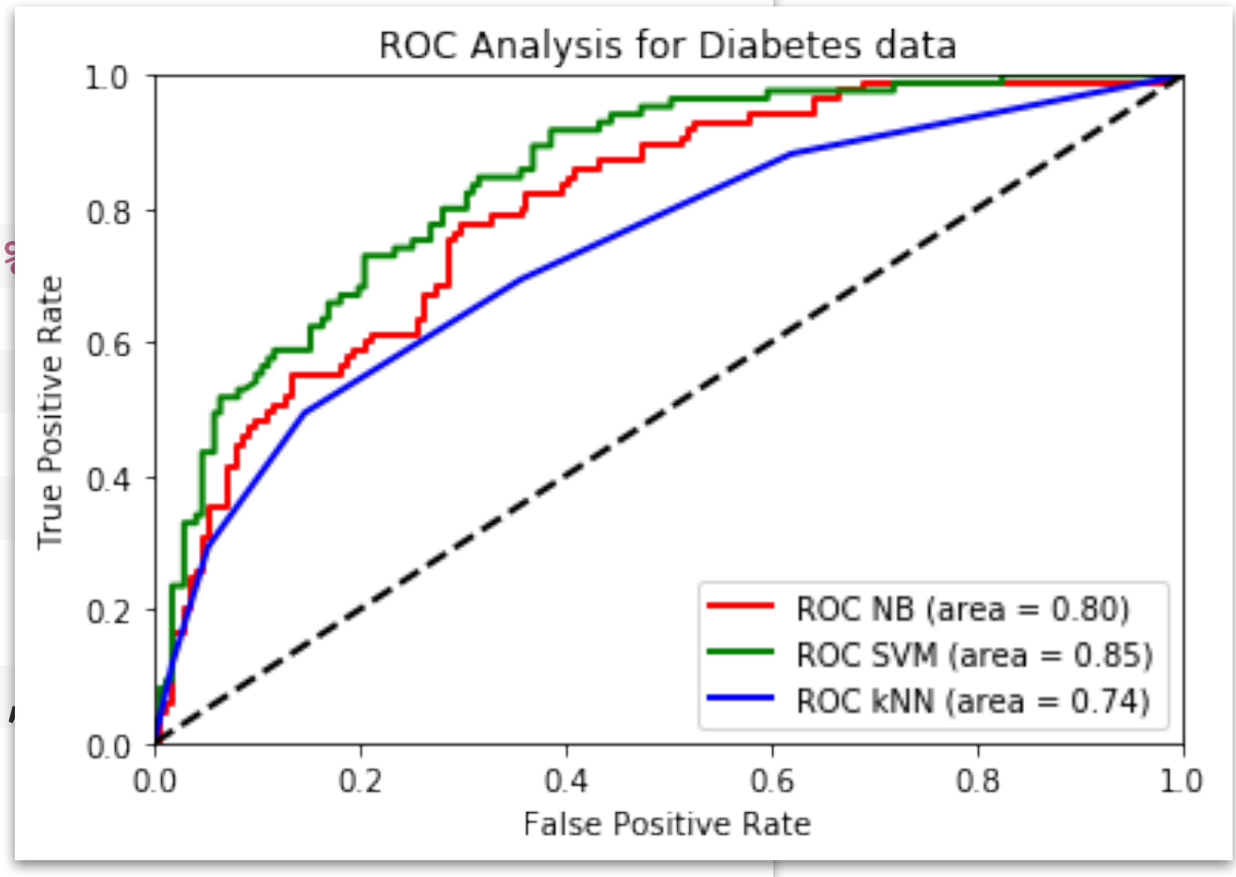
Tutorial Q4(a)

- Plotting code similar to that in notebook **`08 ROC`**

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure()
lw = 2
plt.plot(fprG, tprG, color='red',
         lw=lw, label='ROC NB (area = %
plt.plot(fprS, tprS, color='green',
         lw=lw, label='ROC SVM (area =
plt.plot(fprN, tprN, color='blue',
         lw=lw, label='ROC kNN (area =



plt.plot([0, 1], [0, 1], color='black',
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Analysis for Diabetes data')
plt.legend(loc="lower right")
plt.show()
```



Tutorial Q4(b)

- Repeat with synthetic data

```
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4, n_classes=2,
                           class_sep = 0.75, random_state=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0,test_size=1/3)

gnb = GaussianNB()
y_score = gnb.fit(X_train, y_train).predict_proba(X_test)
fprG, tprG, t = roc_curve(y_test, y_score[:,1])
roc_aucG = auc(fprG, tprG)

kNN = KNeighborsClassifier(n_neighbors = 5)
y_score = kNN.fit(X_train, y_train).predict_pr
fprN, tprN, t = roc_curve(y_test, y_score[:,1]
roc_aucN = auc(fprN, tprN)

di_svm = SVC(kernel = 'linear',C=1, probabilit
y_score = di_SVM.fit(X_train, y_train).predict
fprS, tprS, t = roc_curve(y_test, y_score[:,1]
roc_aucS = auc(fprS, tprS)
```