5.3　評価基準とスコア

15T4034S　荘司響之介

5.3.1　最終的な目標を見失わないこと

基準を選択する際には、常にその機械学習アプリケーションの最終的な目標に留意する必要がある。

→機械学習の基準を選ぶ前に、そのアプリケーションの高レベルでの目的を考える。（ビジネス評価基準）

高レベルな目的の例

交通事故を防ぐ、病院への入院回数を減らす、より多くのユーザを獲得する

特定のモデルの結果を評価するには、それを実際の運用環境に置かなくてはいけない場合もあるが、モデルをテストのためだけに運用環境に置くのは難しい場合が多い。

例えば、自動運転車の歩行者回避能力を評価する際、機能を確認しないで走り回らせるわけにはいかない。

このような場合、何らかの代用となる評価手法を見つける必要がある。（上の例で言えば、歩行者と歩行者以外の画像クラス分類を利用するなどして）

※単なる代用品に過ぎないことに留意する。

5.3.2 2クラス分類における基準

2クラス分類は、実用上最も一般的で、概念的には単純な機械学習アプリケーションであるが、このような簡単なタスクであっても、評価には様々な注意点が必要である。

5.3.2.1 エラーの種類

予測性能の尺度として、「精度」が適切ではない場合がしばしばある。

自動テストでがんの早期発見をスクリーニングするアプリケーションにおいて、間違えの結果が現実世界でどのような影響を及ぼすか考えてみる。健康な人を陰性、がんの可能性がある人を陽性としたとき、陰性を陽性と判断した場合より、陽性を陰性と判断した場合の方が問題が大きいのは明らかである。

このように、間違いには重みの違いが存在する。ビジネスアプリケーションでは双方の間違いに値段をつけ、「精度」ではなく「損失額」で評価することもできる。

5.3.2.2 偏ったデータセット

2つのクラスの一方がもう一方よりもずっと多い場合、予測性能を定量化する基準として、「精度」は不適切である。

例えば、ユーザがインターネット上で提示された広告をクリックするかどうかを予測しようとする。一般的にユーザは100本の広告に対し1本だけに興味をもつ。つまり、99が「クリックされない」データポイントになり、1が「クリックされる」データポイントになる。そのため、常にクリックされないと予測するモデルを作成するだけで、予測精度99%を達成できてしまう。

9:1に偏ったデータセットを用いて実験を行った結果
常に一方(9割の方)を予測するモデル：0.92
ランダムに予測するモデル：0.80
ロジスティック回帰：0.98

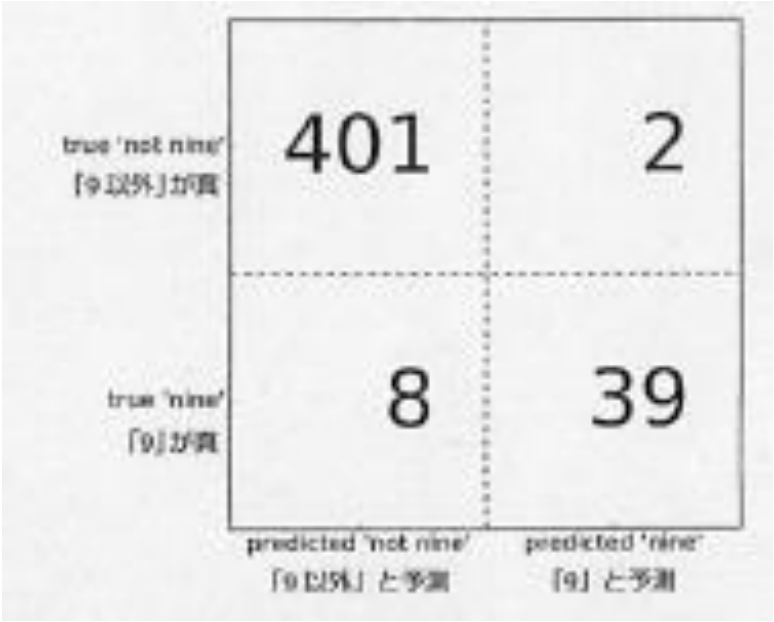
ランダムに予測するモデルですら、0.80の精度

このように、偏ったデータに対して予測性能を定量化する基準として、「精度」は不適切。「常に一方(9割の方)を予測するモデル」や「ランダムに予測するモデル」のような無意味な予測を排除できる基準が必要である。

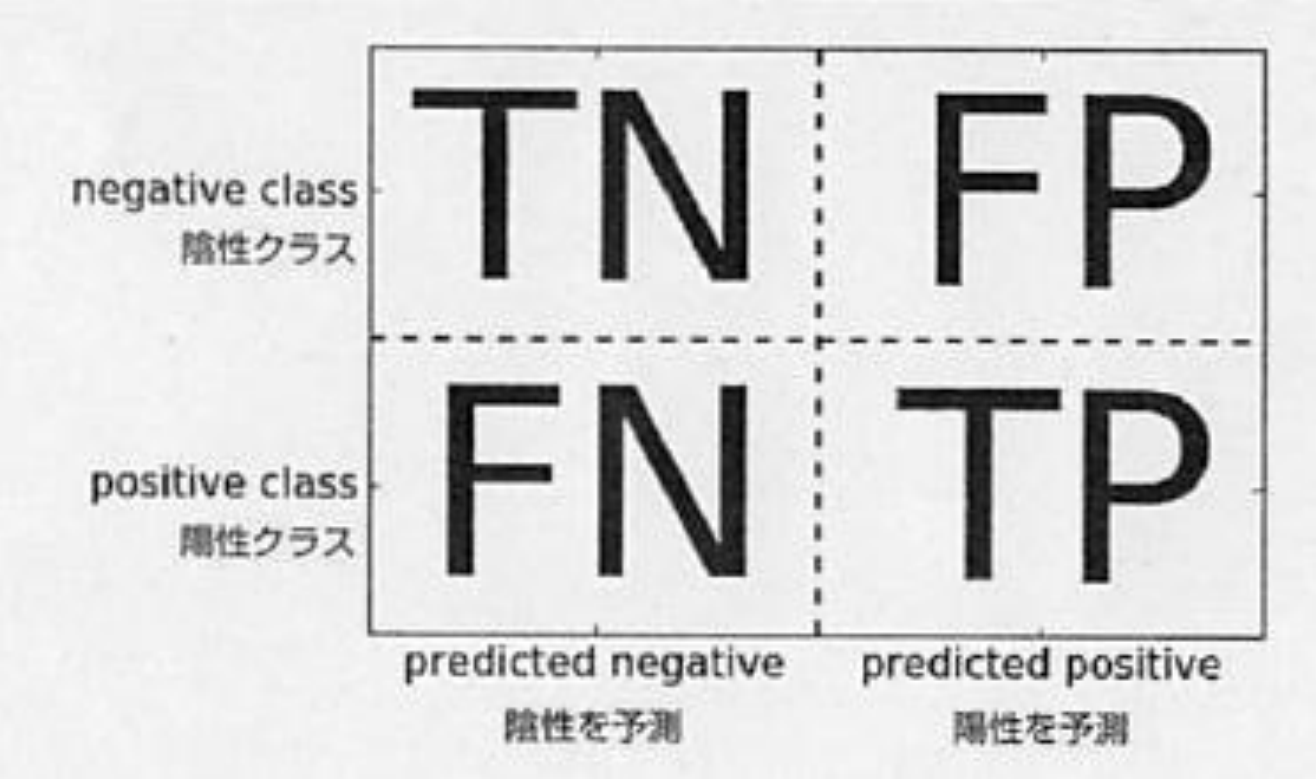
5.3.2.3 混同行列

2クラス分類の評価結果を表現する方法の一つとして、混同行列がある。混同行列の行は実際のクラスに対応し、列は予測されたクラスに対応する。

例えば、前節でみたロジスティック回帰の結果を混同行列で表すと以下のようになる。



2クラス分類において、片方のクラスを陰性、もう片方を陽性と呼び、陽性クラスで正しく分類されたサンプル、陰性クラスで正しく分類されたサンプル、陽性クラスで正しく分類されなかったサンプル、陰性クラスで正しく分類されなかったサンプルをそれぞれ真陽性、真陰性、偽陽性、偽陰性（TP,TN,FP,FN）と呼ぶ。これをまとめると混同行列は以下のようになる。



混同行列を使って、最も多いクラスを選ぶモデル、決定木、ロジスティック回帰を比較する。

最も多いクラスを選ぶモデル

```
[[403   0]
 [ 47   0]]
```

陽性を一つも予測していないのでおかしい

決定木

```
[[361  42]
 [ 43   4]]
```

ロジスティック回帰

```
[[401   2]
 [  8  39]]
```

決定木と比べ真陽性、真陰性の数はより多く、偽陽性、偽陰性の数はより少ない。

適合率

陽性であると予測したものがどのくらい実際に陽性であったかを測定する。

適合率＝TP/TP+FP

偽陽性（本当は陰性だが陽性と予測されたもの）の数を制限したい場合に用いられる。

再現率

実際に陽性のサンプルのうち、陽性と予測されたものの割合

再現率＝TP/TP+FN

すべての陽性サンプルを陽性と予測する必要がある場合に用いられる。

f-値

適合率と再現率はトレードオフの関係にある。例えば、すべてのサンプルを陽性と予測すれば、再現率は100%になるが、偽陽性が大量に発生し再現率は低くなる。

このように適合率、再現率の一方だけでは全体像をつかむことはできない。これらをまとめる方法の一つがf-値である。

F＝2×適合率×再現率/適合率+再現率

先ほどのモデルのf-値を計算すると次のようになる。

最も多いクラスを選ぶモデル：0.00
決定木：0.55
ロジスティック回帰：0.89

※どちらのクラスを陽性にするかで結果が変わる。
　偏ったデータセットの場合、少数派を陽性とするとよい。

5.3.2.4 不確実性を考慮に入れる

ほとんどのクラス分類器には予測の不確実性を予測するための
decision_functionメソッドまたはpredict_probaメソッドが用意されている。
これらのメソッドを使うことで予測を調整することができる。

400点が陰性クラスで50点が陽性クラスになる、偏った分類タスクがある。このデータに対してカーネル法を用いたSVMモデルを訓練し、svc.predictで予測を行った結果を示す。

	precision	recall	f1-score	support	適合率	再現率	f1スコア	支持度
0	0.97	0.89	0.93	104				
1	0.35	0.67	0.46	9				
avg / total	0.92	0.88	0.89	113				

あるアプリケーションにおいてクラス1の再現率を高めることが重要とした場合、この予測は要求を満たしていない。要求を満たそうとした場合、偽陽性が増えても構わないから、より多くの信用性を得る必要がある。つまり、decision_functionのスレッショルド（デフォルトでは0）を小さくする必要がある。（ decision_functionの値がスレッショルド以上であればクラス1に分類されるため）
今回はスレッショルドを-0.8に設定した。

予測を調整した場合の結果を示す。

```
In[54]:

  y_pred_lower_threshold = svc.decision_function(X_test) > -.8
```

```
In[55]:

  print(classification_report(y_test, y_pred_lower_threshold))

Out[55]:
```

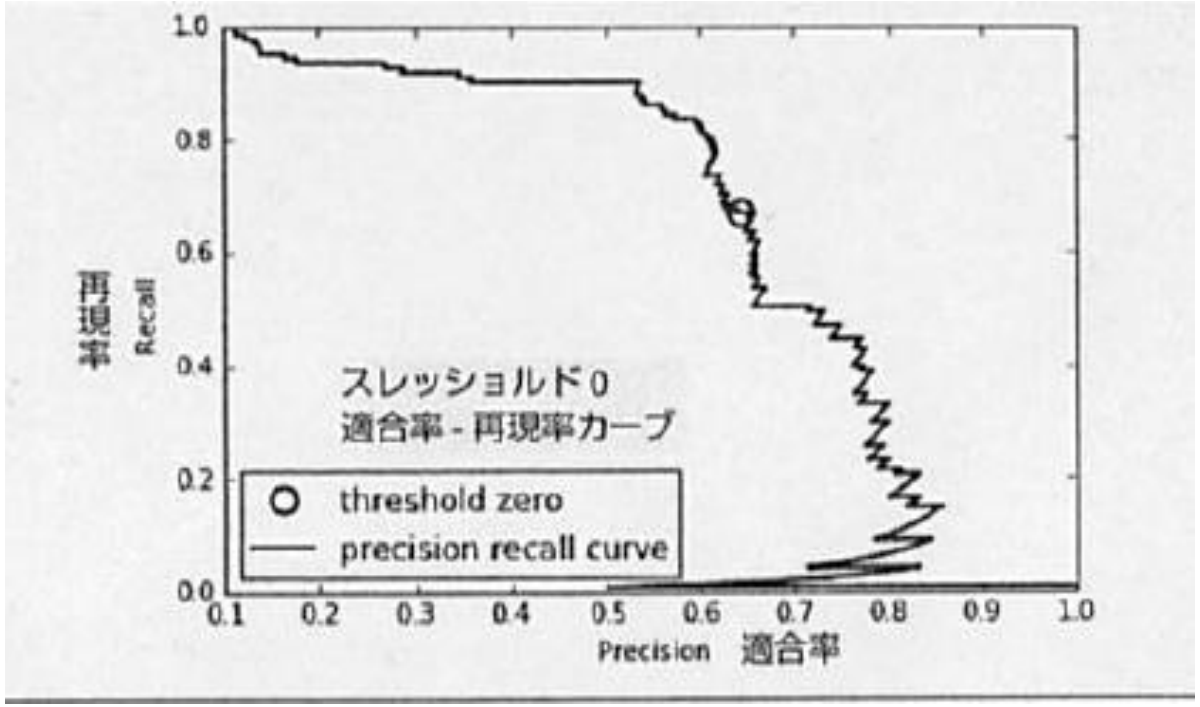
	precision	recall	f1-score	support
	適合率	再現率	f1スコア	支持度
0	1.00	0.82	0.90	104
1	0.32	1.00	0.49	9
avg / total	0.95	0.83	0.87	113

適合率と再現率のどちらか一方を重視したい場合、もしくはデータが大幅に偏っている場合、決定スレッショルドを変更することで良い結果を得ることができる。

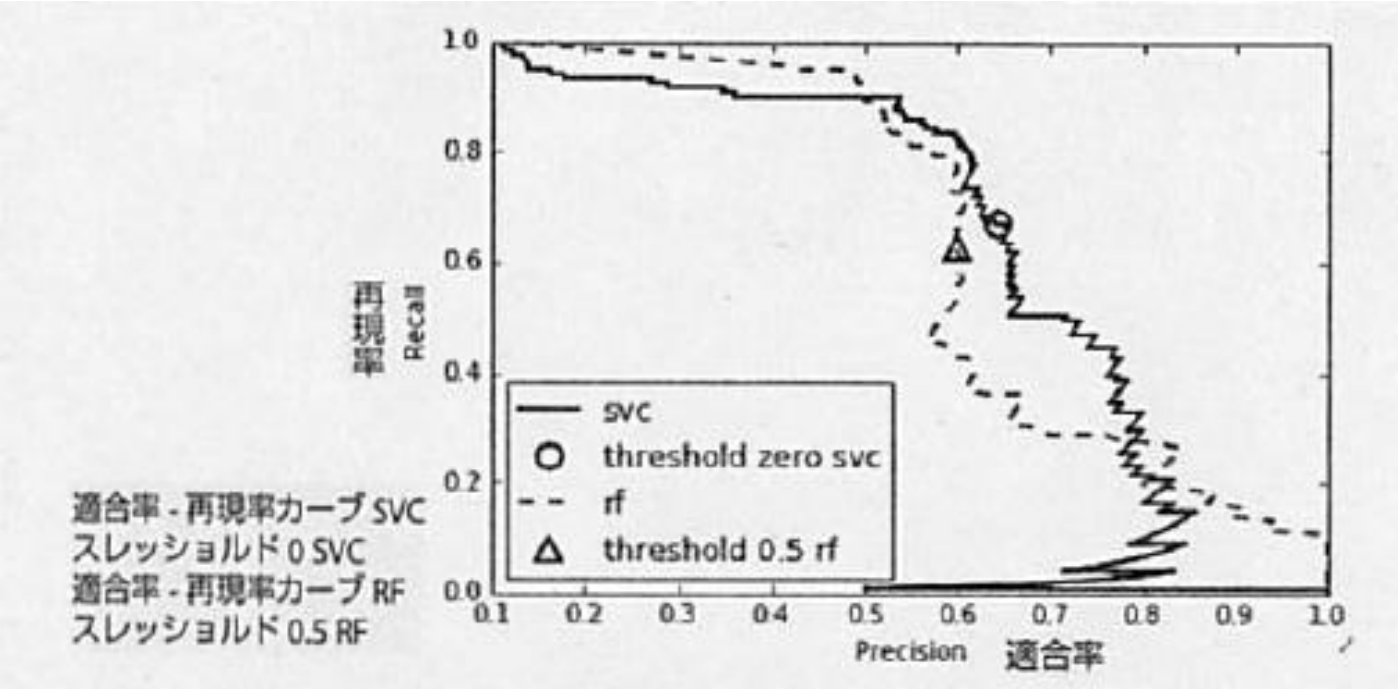
5.3.2.5 適合率-再現率カーブ(とROCカーブ)

クラス分類器に要請を設定することを作動ポイントの設定と呼ぶ。新しいモデルを開発する際、どこが作動ポイントになるか明らかにするために、すべての可能なスレッショルド、すなわちすべての可能な適合率と再現率の組み合わせを同時に見ることが役立つ。

⇒適合率-再現率カーブ



同じデータセットに対して訓練したSVMとランダムフォレストの比較



今回の例では、非常に高い再現率や適合率が求められる場合には、ランダムフォレストの方が性能が良い。その中間ではSVMの方が性能が良い。

適合率-再現率カーブを要約する方法の一つとして、平均適合率と呼ばれるものがある。（カーブの下の領域を積分する。）

先ほどの例に対して計算すると以下のようになる。

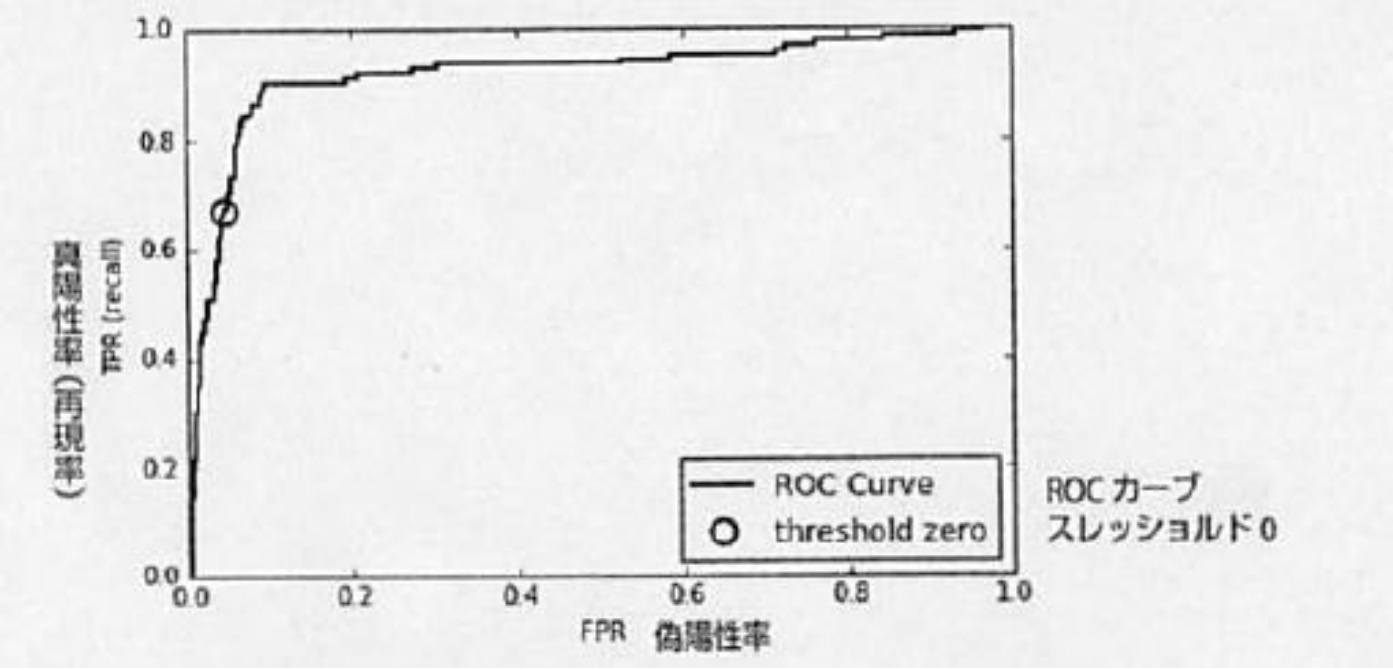
```
Average precision of random forest: 0.666  ランダムフォレストの平均適合率
Average precision of svc: 0.663            SVCの平均適合率
```

スレッショルドで与えられる一点を表しているf-値と違い、適合率-再現率カーブはすべてのスレッショルドで与えられる点を表しているので、微妙な部分まで見ることができる。

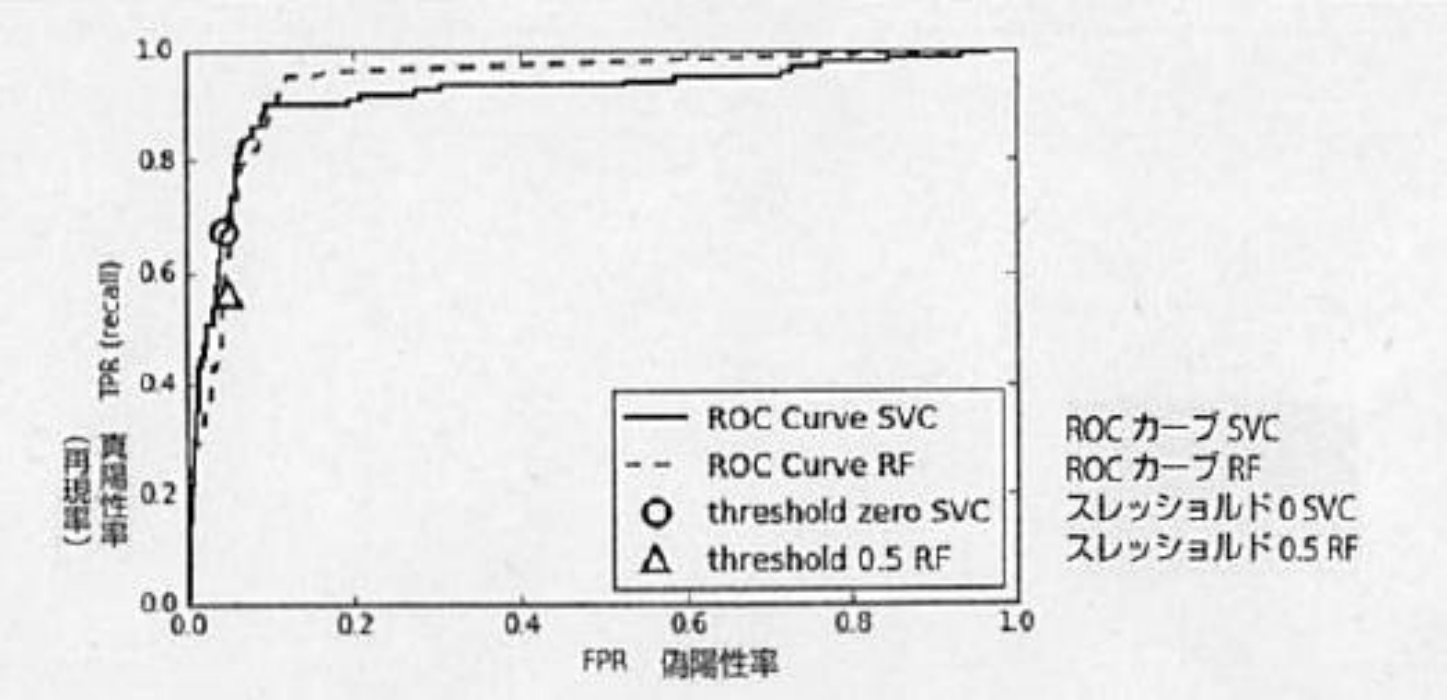
5.3.2.6 受信者動作特性（ROC）とAUC

クラス分類器の挙動を解析するために受信者動作特性カーブ略してROCカーブが使われる。

ROCカーブは適合率-再現率カーブの適合率と再現率の代わりに、偽陽性率（陰性のうち陽性と予測された割合）と真陽性率（再現率）を用いる。



SVMとランダムフォレストの比較



カーブ下の領域を利用してROCカーブを１つの値にまとめることができる。（AUC）

```
AUC for Random Forest: 0.937
AUC for SVC: 0.916
```

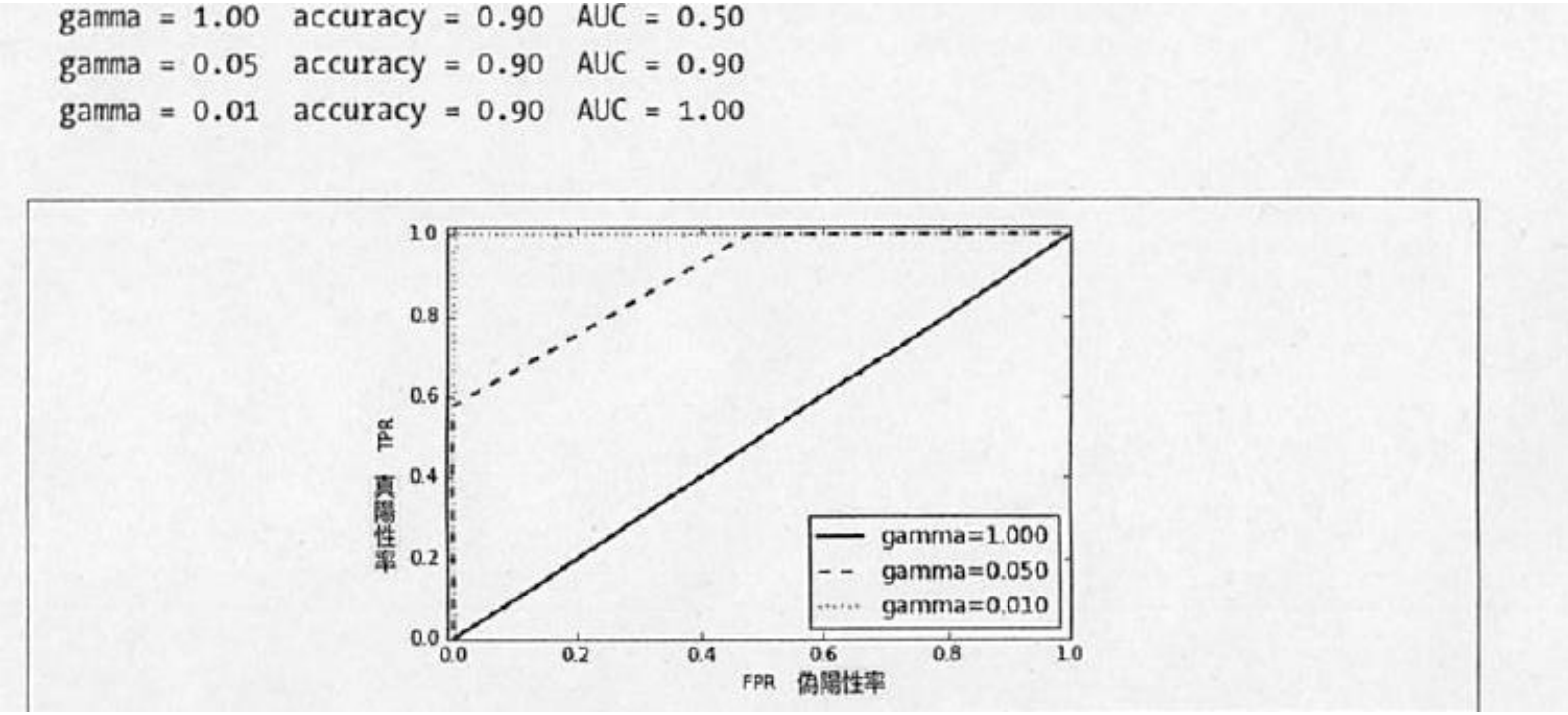
ランダムフォレストのAUC

SVCのAUC

偏ったクラス分類の問題では、精度ではなくAUCをモデル選択に用いた方がはるかに良い結果が得られる。

実際に、SVMのカーネルバンド幅gammaを3種類に設定して偏ったデータセットを分類すると次のようになる。

```
gamma = 1.00  accuracy = 0.90  AUC = 0.50
gamma = 0.05  accuracy = 0.90  AUC = 0.90
gamma = 0.01  accuracy = 0.90  AUC = 1.00
```



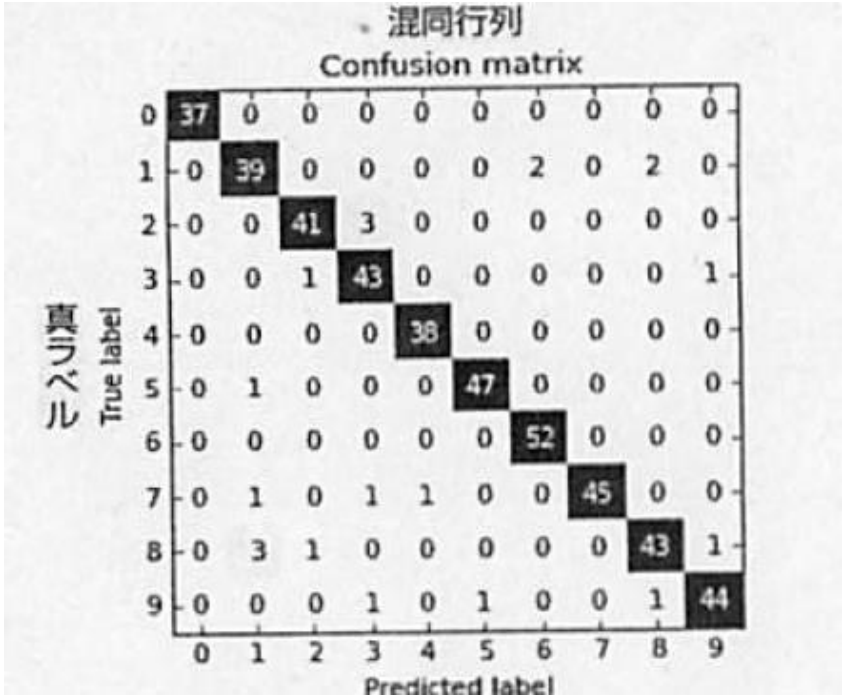
gamma0.01ではスレッショルドを設定すればこのモデルはデータを完璧に分類できる。

5.3.3 多クラス分類の基準

多クラス分類の「精度」以外の評価基準は、２クラス分類と同様に混同行列とクラス分類レポートがある。

0~9の数字をそれぞれ個別に識別する10クラス分類タスクに適用する。

混同行列は以下のようになる。



適合率、再現率、f-値をそれぞれのクラスに足して計算すると以下のようになる

	precision	recall	f1-score	support	適合率	再現率	f1スコア	支持度
0	1.00	1.00	1.00	37				
1	0.89	0.91	0.90	43				
2	0.95	0.93	0.94	44				
3	0.90	0.96	0.92	45				
4	0.97	1.00	0.99	38				
5	0.98	0.98	0.98	48				
6	0.96	1.00	0.98	52				
7	1.00	0.94	0.97	48				
8	0.93	0.90	0.91	48				
9	0.96	0.94	0.95	47				
avg / total	0.95	0.95	0.95	450				

偏ったデータセットに対する多クラス分類で最もよく用いられる基準は、f-値であるが、クラスごとのf-値を平均する方法として次のようなものがある。

・重みを付けずにクラスごとのf-値を平均する、”macro”平均（個々のクラスを同じように重視する場合）

・各クラスの支持度に応じて重みを付けて、クラスごとのf-値を平均する、“weighted”平均

・すべてのクラスの偽陽性、偽陰性、真陽性の総数を計算し、その値を用いて、適合率、再現率、f-値を計算する、”micro”平均（個々のサンプルを同じように重視する場合）

5.3.4 回帰の基準

平均二乗誤差や平均絶対誤差を使ってビジネス決定を行う場合には、クラス分類と同じような基準でモデルをチューニングする必要があるが、回帰モデルを評価するうえでは、R^2が最も直感的な基準である。

5.3.5 評価基準を用いたモデル選択

AUCのような基準をGridSearchCVやcross_val_scoreによるモデル選択で用いたい場合、scoringパラメータに”roc_auc”を与えればよい

- cross_val_score

```
# デフォルトのクラス分類スコアは精度
print("Default scoring: {}".format(
    cross_val_score(SVC(), digits.data, digits.target == 9)))
# scoring="accuracy"としても結果は変わらない
explicit_accuracy =  cross_val_score(SVC(), digits.data, digits.target == 9,
                                     scoring="accuracy")

print("Explicit accuracy scoring: {}".format(explicit_accuracy))
roc_auc =  cross_val_score(SVC(), digits.data, digits.target == 9,
                           scoring="roc_auc")
print("AUC scoring: {}".format(roc_auc))
```

Out[69]:

```
Default scoring: [ 0.9  0.9  0.9]
Explicit accuracy scoring: [ 0.9  0.9  0.9]
AUC scoring: [ 0.994  0.99  0.996]
```

デフォルトのスコア

明示的に精度を指定したスコア

AUCによるスコア

- GridSearchCV

In[70]:

```
X_train, X_test, y_train, y_test = train_test_split(
    digits.data, digits.target == 9, random_state=0)

# 説明の都合上、あまり良くないグリッドを与える
param_grid = {'gamma': [0.0001, 0.01, 0.1, 1, 10]}
# デフォルトのスコア法である精度で評価
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
print("Grid-Search with accuracy")
print("Best parameters:", grid.best_params_)
print("Best cross-validation score (accuracy)): {:.3f}".format(grid.best_score_))
print("Test set AUC: {:.3f}".format(
    roc_auc_score(y_test, grid.decision_function(X_test))))
print("Test set accuracy: {:.3f}".format(grid.score(X_test, y_test)))
```

Out[70]:

```
Grid-Search with accuracy                          精度を用いたグリッドサーチ
Best parameters: {'gamma': 0.0001}                 最良のパラメータ
Best cross-validation score (accuracy)): 0.970     最良の交差検証スコア
Test set AUC: 0.992                                テストセットのAUC
Test set accuracy: 0.973                           テストセットの精度
```

In[71]:

```
# AUCをスコアに用いる
grid = GridSearchCV(SVC(), param_grid=param_grid, scoring="roc_auc")
grid.fit(X_train, y_train)
print("\nGrid-Search with AUC")
print("Best parameters:", grid.best_params_)
print("Best cross-validation score (AUC): {:.3f}".format(grid.best_score_))

print("Test set AUC: {:.3f}".format(
    roc_auc_score(y_test, grid.decision_function(X_test))))
print("Test set accuracy: {:.3f}".format(grid.score(X_test, y_test)))
```

Out[71]:

```
Grid-Search with AUC                               AUCを用いたグリッドサーチ
Best parameters: {'gamma': 0.01}                   最良のパラメータ
Best cross-validation score (AUC): 0.997           最良の交差検証スコア
Test set AUC: 1.000                                テストセットのAUC
Test set accuracy: 1.000                           テストセットの精度
```