1.

A supervised learning model has been built to predict whether someone is infected with a new strain of a virus. The probability of any one person having the virus is 1%. Using accuracy as a metric, what would be a good choice for a baseline accuracy score that the new model would want to outperform? "The dummy classifier provides what is called a null accuracy baseline. That is the **accuracy that can be achieved by always picking the most frequent class**."

1%

0.99 CORRECT

2.

Given the following confusion matrix:

|  |  |  |
| --- | --- | --- |
|  | Predicted Positive | Predicted Negative |
| Condition Positive | 96 | 4 |
| Condition Negative | 8 | 19 |

Compute the accuracy to three decimal places.

Accuracy = correct classification

Accuracy = (a + d)/ (a + b + c + d)

Correct\_answered = 96 + 19

Total = 96 + 19 + 8 + 4

Accuracy = Correct\_answered/Total

Accuracy = 0.905511811023622

0.906

3.

Given the following confusion matrix:

|  |  |  |
| --- | --- | --- |
|  | Predicted Positive | Predicted Negative |
| Condition Positive | 96 | 4 |
| Condition Negative | 8 | 19 |

Compute the precision to three decimal places.

Precision = exactness.

Specifically a high score tells us that we﻿ have less false positives. In other words, the less innocent people classified as a criminal the higher our precision score.

Precision = a/ (a + c)

precision = 96/(96 + 8)

precision = 0.9230769230769231

0.923

4.

Given the following confusion matrix:

|  |  |  |
| --- | --- | --- |
|  | Predicted Positive | Predicted Negative |
| Condition Positive | 96 | 4 |
| Condition Negative | 8 | 19 |

Compute the recall to three decimal places.

Recall = completeness. Specifically a high Recall score tells us that we have less false negatives. In﻿ other words the less criminals classified as innocent people the higher our recall score.

Recall = a/ (a + b)

recall = 96/ (96 + 4)

recall = 0.960

5.

Using the fitted model `m` create a precision-recall curve to answer the following question:

For the fitted model `m`, approximately what precision can we expect for a recall of 0.8?

(Use y\_test and X\_test to compute the precision-recall curve. If you wish to view a plot, you can use `plt.show()` )

1

print(m)

RunReset

m\_scores = m.predict(X\_test)

precision, recall, thresholds = precision\_recall\_curve(y\_test, m\_scores)

precision, recall, thresholds = precision\_recall\_curve(y\_test, y\_scores\_lr)

closest\_zero = np.argmin(np.abs(thresholds))

closest\_zero\_p = precision[closest\_zero]

closest\_zero\_r = recall[closest\_zero]

plt.figure()

plt.xlim([0.0, 1.01])

plt.ylim([0.0, 1.01])

plt.plot(precision, recall, label='Precision-Recall Curve')

plt.plot(closest\_zero\_p, closest\_zero\_r, 'o', markersize = 12, fillstyle = 'none', c='r', mew=3)

plt.xlabel('Precision', fontsize=16)

plt.ylabel('Recall', fontsize=16)

plt.axes().set\_aspect('equal')

plt.show()

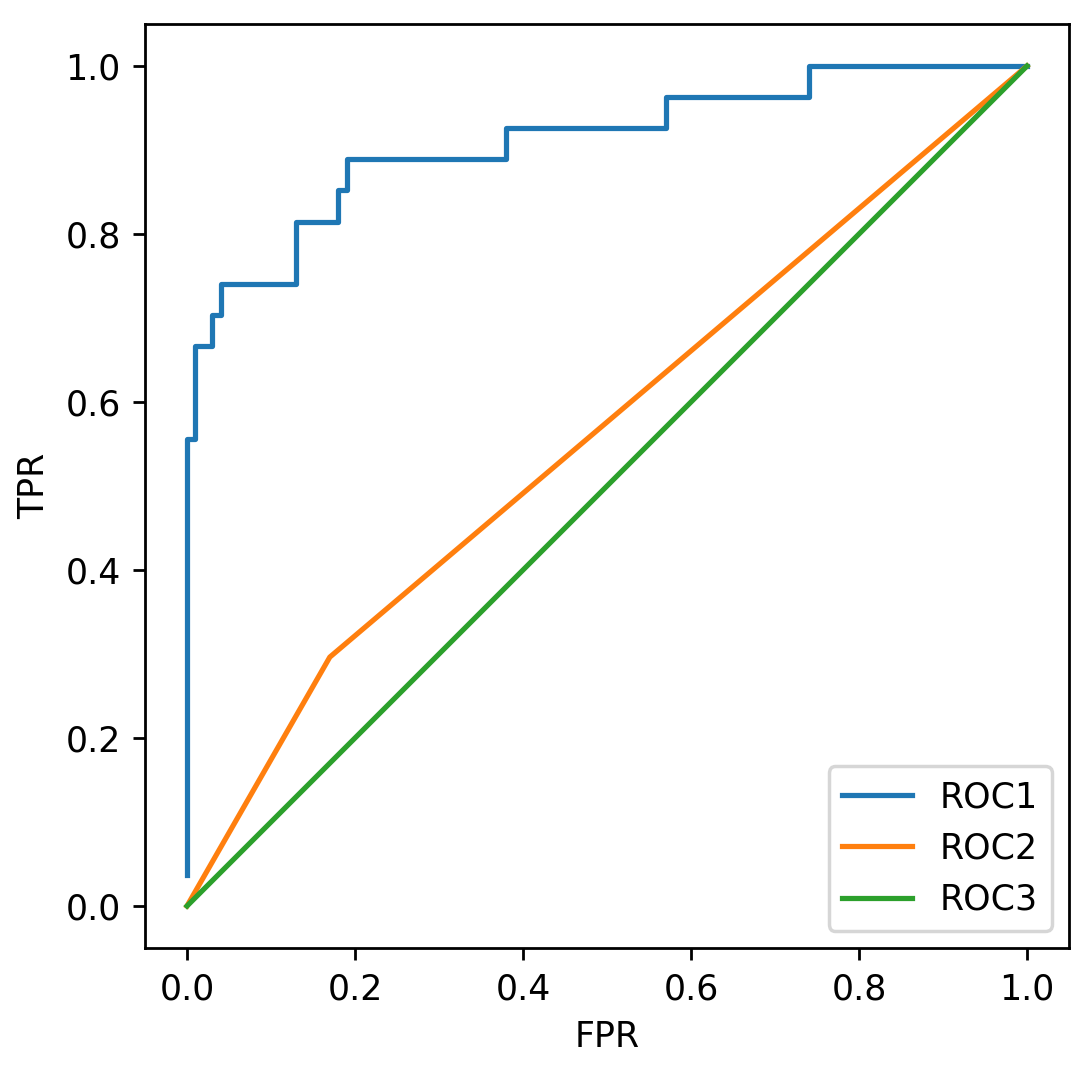
print([(precision, recall) for precision, recall in zip(precision, recall) if 0.81 < recall < 0.85])

0.6

6.

Given the following models and AUC scores, match each model to its corresponding ROC curve.

* Model 1 test set AUC score: 0.91
* Model 2 test set AUC score: 0.50
* Model 3 test set AUC score: 0.56



* Model 1: Roc 1
* Model 2: Roc 2
* Model 3: Roc 3
* Model 1: Roc 1
* Model 2: Roc 3
* Model 3: Roc 2
* Model 1: Roc 2
* Model 2: Roc 3
* Model 3: Roc 1
* Model 1: Roc 3
* Model 2: Roc 2
* Model 3: Roc 1

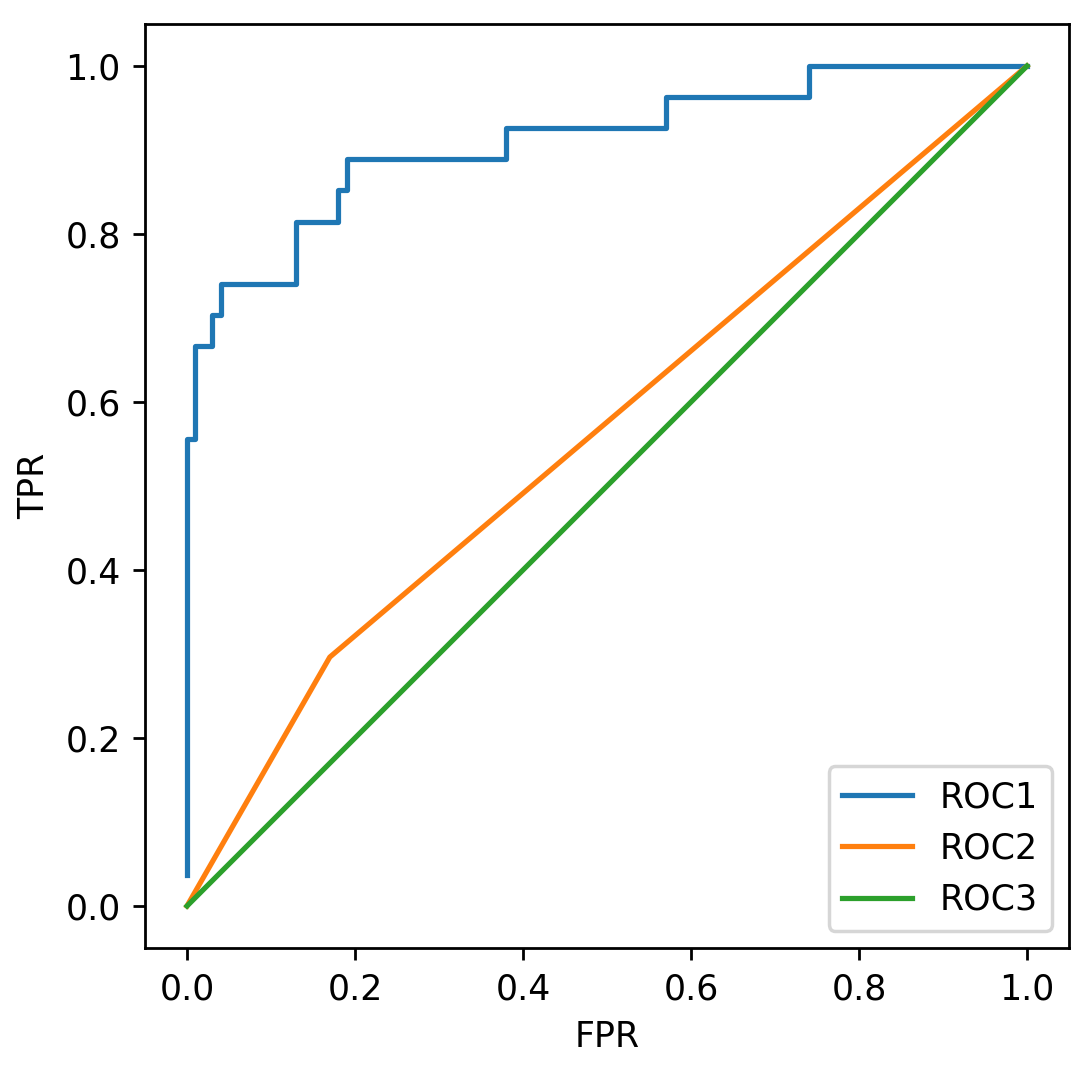
Not enough information is given.

1  
point

7.

Given the following models and accuracy scores, match each model to its corresponding ROC curve.

* Model 1 test set accuracy: 0.91
* Model 2 test set accuracy: 0.79
* Model 3 test set accuracy: 0.72



* Model 1: Roc 1
* Model 2: Roc 2
* Model 3: Roc 3
* Model 1: Roc 1
* Model 2: Roc 3
* Model 3: Roc 2
* Model 1: Roc 2
* Model 2: Roc 3
* Model 3: Roc 1
* Model 1: Roc 3
* Model 2: Roc 2
* Model 3: Roc 1

Not enough information is given.

ROC is the graph resulting from plotting True Positive Rate([sensitivity](https://en.wikipedia.org/wiki/Sensitivity_(tests))) vs. False Positive Rate(1 − [specificity](https://en.wikipedia.org/wiki/Specificity_(tests))) , AUC is the area under that curve

AUC stands for Area Under the Curve, which curve you ask? Well that would be the ROC curve. ROC stands for [Receiver Operating Characteristic](http://en.wikipedia.org/wiki/Receiver_operating_characteristic), which is actually slightly non-intuitive. The implicit goal of AUC is to deal with situations where you have a very skewed sample distribution, and don't want to overfit to a single class.

AUC measures how true positive rate (recall) and false positive rate trade off, so in that sense it is already measuring something else. More importantly, AUC is not a function of threshold. It is an evaluation of the classifier as threshold varies over all possible values. It is in a sense a broader metric, testing the quality of the internal value that the classifier generates and then compares to a threshold. It is not testing the quality of a particular choice of threshold.

AUC has a different interpretation, and that is that it's also the probability that a randomly chosen positive example is ranked above a randomly chosen negative example, according to the classifier's internal value for the examples.

Like all the answers have been posted: ROC and accuracy is fundamental two different concepts.

Generally speaking, ROC describes the discriminative power of a classifier independent of class distribution and unequal prediction error costs (false positive and false negative cost).

Metric like accuracy is calculated based on the class distribution of test dataset or cross-validation, but this ratio may change when you apply the classifier to real life data, because the underlying class distribution has been changed or unknown. On the other hand, TP rate and FP rate which are used to construct AUC will be not be affected by class distribution shifting.

1  
point

8.

Using the fitted model `m` what is the micro precision score?

(Use y\_test and X\_test to compute the precision score.)

1

m\_scores = m.predict(X\_test)

micro\_precision\_score = precision\_score(y\_test, m\_scores, average = 'micro')

print(micro\_precision\_score)

0.744

print(m)

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

9.

Which of the following is true of the R-Squared metric? (Select all that apply)

The best possible score is 1.0

A model that always predicts the mean of y would get a score of 0.0

The worst possible score is 0.0

A model that always predicts the mean of y would get a negative score

1  
point

Put simply:

* When R2<0, a horizontal line explains the data better than your model.

You also asked about R2=0

* When R2=0, a horizontal line explains the data equally as well as your model.

**Bottom line:** a negative R2R2 is not a mathematical impossibility or the sign of a computer bug. It simply means that the chosen model (with its constraints) fits the data really poorly.

R2R2 compares the fit of the chosen model with that of a horizontal straight line (the null hypothesis). If the chosen model fits worse than a horizontal line, then R2R2 is negative. Note that R2R2 is not always the square of anything, so it can have a negative value without violating any rules of math. R2R2 is negative only when the chosen model does not follow the trend of the data, so fits worse than a horizontal line.

10.

In a future society, a machine is used to predict a crime before it occurs. If you were responsible for tuning this machine, what evaluation metric would you want to maximize to ensure no innocent people (people not about to commit a crime) are imprisoned (where crime is the positive label)?

Accuracy

Precision

Recall

F1

AUC

1  
point

11.

Consider the machine from the previous question. If you were responsible for tuning this machine, what evaluation metric would you want to maximize to ensure all criminals (people about to commit a crime) are imprisoned (where crime is the positive label)?

Accuracy

Precision

Recall

F1

AUC

1  
point

12.

A classifier is trained on an imbalanced multiclass dataset. After looking at the model’s precision scores, you find that the micro averaging is much smaller than the macro averaging score. Which of the following is most likely happening?

The model is probably misclassifying the infrequent labels more than the frequent labels.

The model is probably misclassifying the frequent labels more than the infrequent labels.

1  
point

13.

Using the already defined RBF SVC model `m`, run a grid search on the parameters C and gamma, for values [0.01, 0.1, 1, 10]. The grid search should find the model that best optimizes for recall. How much better is the recall of this model than the precision? (Compute recall - precision to 3 decimal places)

(Use y\_test and X\_test to compute precision and recall.)

1

print(m)

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, probability=False, random\_state=0, shrinking=True,

tol=0.001, verbose=False)

grid\_values = {'gamma': [0.01, 0.1, 1, 10]}

# 1- get the model using the scoring of recall in gridsearch

grid\_clf = GridSearchCV(m, param\_grid = grid\_values, scoring = 'recall')

# fit the model

model1 = grid\_clf.fit(X\_train, y\_train)

# 2- get y\_predicted values using the above model

y\_predicted = model1.predict(X\_test)

# 3- get the precision score using the y\_predicted & y\_test

precision = precision\_score(y\_test, y\_predicted) # 0.904761904762

# 4- compare that precision score with the model best\_score\_

# bs means best score for recall

bs=model1.best\_score\_ # 0.934635897436

print(round(bs-precision),3)

0.030

you need to use gridsearch with scoring set to recall, fit it with X\_train and y\_train and then, once fitted, predict y for X\_test. Once you have that, you can use precision\_score and recall\_score to get the precision and recall and then get the difference between the two values

grid\_values = {'gamma': [0.01, 0.1, 1, 10]

# 1- get the model using the scoring of recall in gridsearch

grid\_clf = GridSearchCV(m, param\_grid = grid\_values, scoring = 'recall')

# fit the model

model1 = grid\_clf.fit(X\_train, y\_train)

# 2- get y\_predicted values using the above model

y\_predicted = model1.predict(X\_test)

# 3- get the precision score using the y\_predicted & y\_test

precision = precision\_score(y\_test, y\_predicted)

# 4-- get the recall score using the y\_predicted & y\_test

recall = recall\_score(y\_test, y\_predicted)

# 5- compute recall - precision

diff = recall - precision

-0.113095238095 Wrong

-0.113

grid\_values = {'gamma': [0.01, 0.1, 1, 10], 'C': [0.01, 0.1, 1, 10]}

# 1- get the model using the scoring of recall in gridsearch

grid\_clf = GridSearchCV(m, param\_grid = grid\_values, scoring = 'recall')

# fit the model

model1 = grid\_clf.fit(X\_train, y\_train)

# 2- get y\_predicted values using the above model

y\_predicted = model1.predict(X\_test)

# 3- get the precision score using the y\_predicted & y\_test

precision = precision\_score(y\_test, y\_predicted)

# 4-- get the recall score using the y\_predicted & y\_test

recall = recall\_score(y\_test, y\_predicted)

# 5- compute recall - precision

diff = recall – precision

0.52

14.

Using the already defined RBF SVC model `m`, run a grid search on the parameters C and gamma, for values [0.01, 0.1, 1, 10]. The grid search should find the model that best optimizes for precision. How much better is the precision of this model than the recall? (Compute precision - recall to 3 decimal places)

(Use y\_test and X\_test to compute precision and recall.)

1

print(m)

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, probability=False, random\_state=0, shrinking=True,

tol=0.001, verbose=False)

grid\_values = {'gamma': [0.01, 0.1, 1, 10]}

# 1- get the model using the scoring of recall in gridsearch

grid\_model = GridSearchCV(m, param\_grid = grid\_values, scoring = 'precision')

# fit the model

model2 = grid\_model.fit(X\_train, y\_train)

# 2- get y\_predicted values using the above model

y\_predicted = model2.predict(X\_test)

# 3- get the precision score using the y\_predicted & y\_test

rc1 = recall\_score(y\_test, y\_predicted) #

# 4- compare that precision score with the model best\_score\_

bs2=model2.best\_score\_ #

diff = bs2 - rc1

0.0742132505176

grid\_values = {'gamma': [0.01, 0.1, 1, 10]}

# 1- get the model using the scoring of recall in gridsearch

grid\_model = GridSearchCV(m, param\_grid = grid\_values, scoring = 'precision')

# fit the model

model2 = grid\_model.fit(X\_train, y\_train)

# 2- get y\_predicted values using the above model

y\_predicted = model2.predict(X\_test)

# 3- get the precision score using the y\_predicted & y\_test

prec = precision\_score(y\_test, y\_predicted)

# 4- get the recall score using the y\_predicted & y\_test

rec = recall\_score(y\_test, y\_predicted)

# 5- compute precision - recall

diff2 = prec – rec

0.166666666667 Wrong

0.167

grid\_values = {'gamma': [0.01, 0.1, 1, 10], 'C': [0.01, 0.1, 1, 10]}

# 1- get the model using the scoring of recall in gridsearch

grid\_model = GridSearchCV(m, param\_grid = grid\_values, scoring = 'precision')

# fit the model

model2 = grid\_model.fit(X\_train, y\_train)

# 2- get y\_predicted values using the above model

y\_predicted = model2.predict(X\_test)

# 3- get the precision score using the y\_predicted & y\_test

prec = precision\_score(y\_test, y\_predicted)

# 4- get the recall score using the y\_predicted & y\_test

rec = recall\_score(y\_test, y\_predicted)

# 5- compute precision - recall

diff2 = prec - rec

0.15