Question 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?

0.99

Question 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.

0.905

Question 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.

0.923

Question 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.

0.960

Question 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()` )

from sklearn.metrics import precision\_recall\_curve

y\_scores= m.fit(X\_train, y\_train).decision\_function(X\_test)

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

plt.figure()

plt.xlim([0.0, 1.01])

plt.ylim([0.0, 1.01])

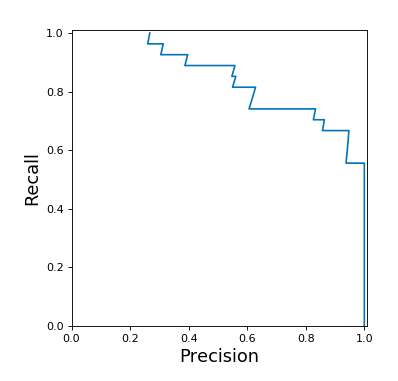
plt.plot(precision, recall, label='Precision-Recall Curve')

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

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

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

plt.show()

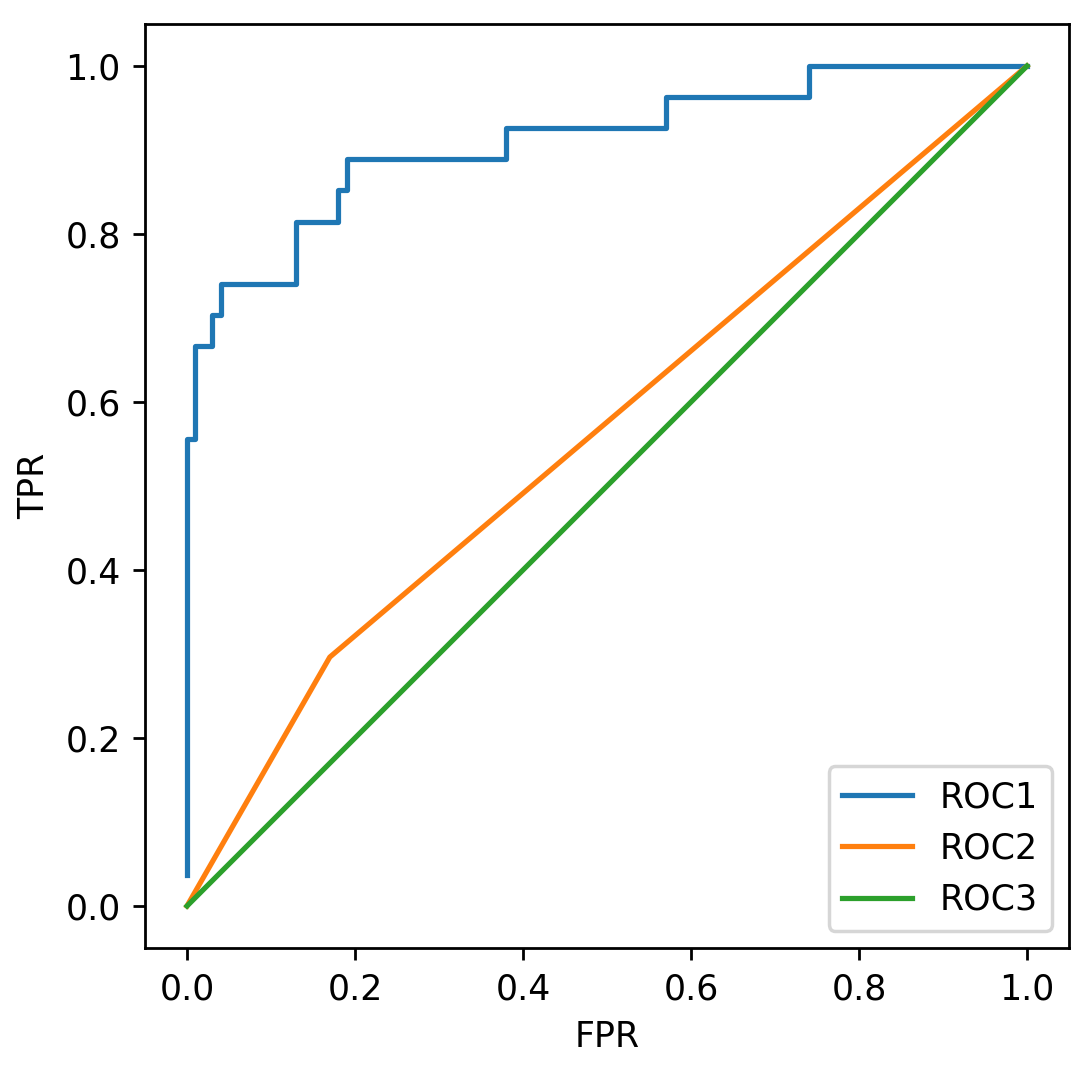


0.6

Question 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

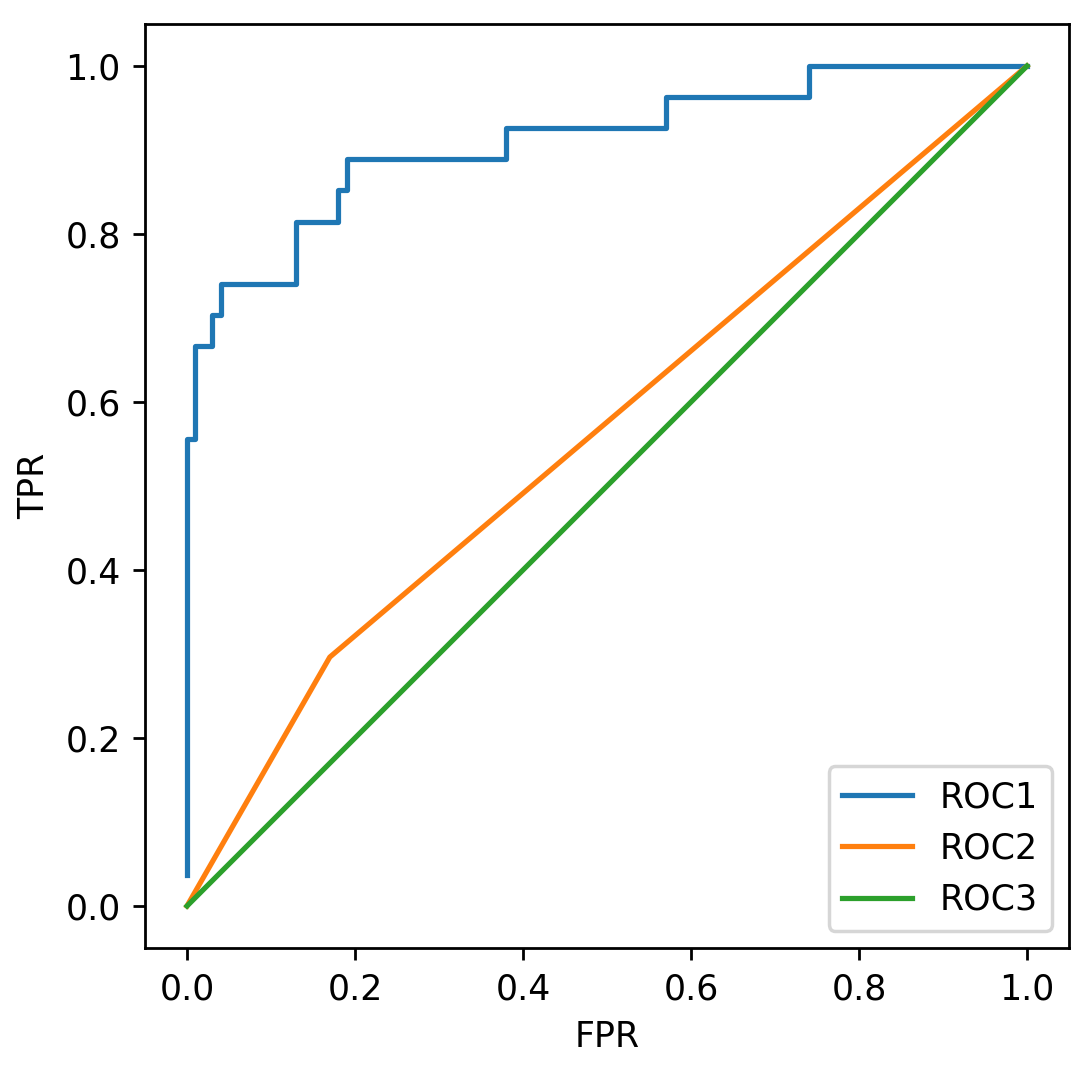


* Model 1: Roc 1
* Model 2: Roc 3
* Model 3: Roc 2

Question 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



Not enough information is given.

**Question 8**

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

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

m.fit(X\_train,y\_train)

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

print(precision\_score(y\_test,y\_predicted, average = 'macro'))

0.744

**Question 9**

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

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

**Question 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)?

Precision

**Question 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)?

Recall

**Question 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 frequent labels more than the infrequent labels.

Question 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.)

from sklearn.svm import SVC

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import roc\_auc\_score

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

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

grid\_clf\_acc.fit(X\_train, y\_train)

print(grid\_clf\_acc.best\_params\_)

print(grid\_clf\_acc.best\_score\_)

#use the recall from the previous step

clf\_new = SVC(C=0.01, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma=0.01, kernel='rbf',

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

tol=0.001, verbose=False)

clf\_new.fit(X\_train, y\_train)

y\_predicted = clf\_new.predict(X\_test)

print(recall\_score(y\_test, y\_predicted) - precision\_score(y\_test, y\_predicted))

0.52

Question 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.)

from sklearn.svm import SVC

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import roc\_auc\_score

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

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

grid\_clf\_acc.fit(X\_train, y\_train)

print(grid\_clf\_acc.best\_params\_)

print(grid\_clf\_acc.best\_score\_)

#use the precision from the previous step

clf\_new = SVC(C=10, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma=1, kernel='rbf',

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

tol=0.001, verbose=False)

clf\_new.fit(X\_train, y\_train)

y\_predicted = clf\_new.predict(X\_test)

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

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

print(precision\_score(y\_test, y\_predicted) - recall\_score(y\_test, y\_predicted) )

0.15