Week 17

Classification with K-Nearest Neighbors

Applied Data Science

Columbia University - Columbia Engineering

Course Agenda

COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

- Week 10: Organizing and Analyzing Data with NumPy and Pandas
- Week 11: Cleaning and Visualizing Data with Pandas and Matplotlib
- Week 12: Statistical Distributions
- Week 13: Statistical Sampling
- Week 14: Hypothesis Testing
- Week 15: Regression Models in Python
- Week 16: Evaluating Data Models
- **Week 17: Classification with K-Nearest Neighbors**
- Week 18: Decision Tree Models
- Week 19: Clustering Models
- Week 20: Text Mining in Python -- Analyzing Sentiment
- Week 21: Text Mining in Python -- Topic Modeling

Confusion Matrix

Actual	Predict
1	1
1	0
1	1
1	1
1	1
1	0
1	1
0	0
0	0
0	1
0	0
0	0
0	1
0	1

True Positive

False Negative

	Pred 1	Pred 0
Actual 1	5	3
Actual 0	2	4

False Positive

True Negative

Precision: what proportion of the cases that the model said were 1 were actually 1

- precision = TP/(TP+FP)
- 5/(5+2) = 71.4%

Recall: what proportion of the cases that were actually 1 were identified as 1 by the model

- recall = TP/(TP + FN)
- 5/(5+3) = 62.5%

F-Score: measures accuracy by balancing precision and recall

- fscore = 2 * (P * R)/(P + R)
- 67%

True Positive Rate (TPR): what proportion of the cases that were actually 1 were identified as 1(tpr = recall)

- tpr = TP/(TP+FN)
- 5/(5+3) = 62.5%

False Positive Rate (FPR): what proportion of the cases that the model said were 1 were actually zero

- fpr = FP/(TN + FP)
- 2/(4+2) =33.3%

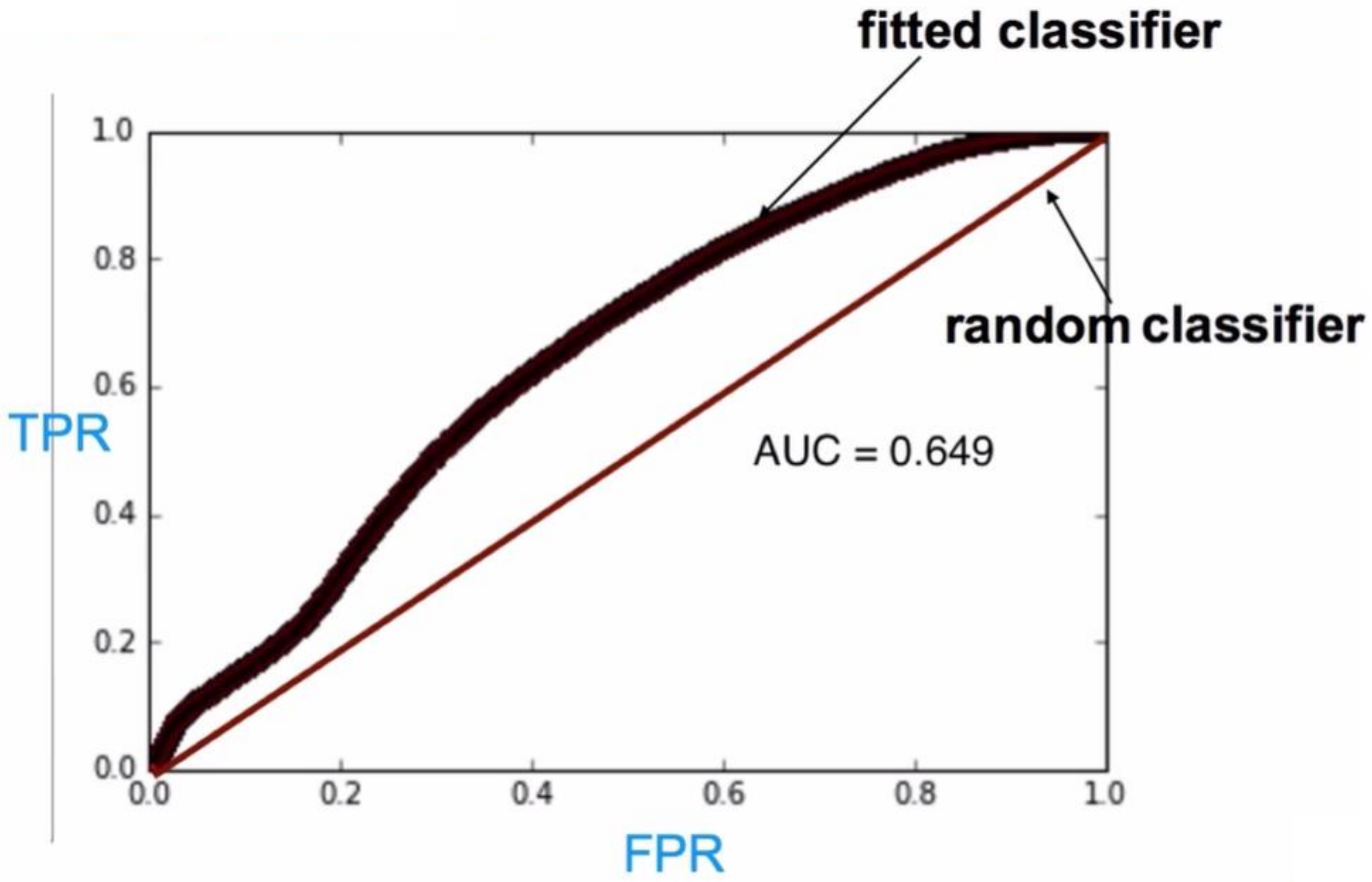
Fixing the Threshold

ROC Curve: plots the True Positive Rate against the False Positive Rate as the threshold varies from 0 to 1

Precision-Recall Curve: Plots precision against recall as the threshold varies from 0 to 1

ROC Curve

Plots the True Positive Rate against the False Positive Rate as the threshold varies from 0 to 1



Determining the Threshold

AUROC: The area under the ROC curve. AUROC is used to determine if the classifier is doing better than a random classifier. It can also help pick a threshold.
AUPRC: The area under the PRC curve.

Generate Predictions in Sample Error


```
In [22]: training_predictions = model.predict(x_train)
         print(np.mean((training_predictions - y_train) ** 2))

         0.08541463252093508

In [23]: print('Train R-Square:',model.score(x_train,y_train))
         print('Test R-Square:',model.score(x_test,y_test))

         Train R-Square: 0.657934733571
         Test R-Square: 0.0425249928917

In [24]: training_predictions

Out[24]: array([-0.10188075,  0.38338798,  0.79485029,  0.62925317,  0.420565  ,
                 0.06581291,  0.51937525,  1.25271651,  1.3796749 ,  0.59242978,
                 0.65898004, -0.36202426,  0.57004476,  1.13446586,  0.06057537,
                 0.8790505 ,  0.22359053,  0.6856976 ,  0.82861061,  0.2322907 ,
                 0.44177435,  0.66517222, -0.08037913,  0.14881398,  0.02279126,
                 0.9296724 ,  0.04369276,  0.12734659,  0.24923559,  0.26683005,
                 0.52840754,  0.71207192,  0.06695876, -0.09386759,  0.75978612,
                 0.62196944,  0.79981047,  0.23834202, -0.29446555,  0.09533922,
                -0.02633585,  0.33733747,  0.47656187,  0.79488051,  0.99599178,
                 0.35621157,  0.19327808,  0.83864864,  0.90311218,  0.81976534
```

- Reports the proportion of

1. **true positive**: predicts mine and is a mine
2. **false positive**: predicts mine and is not a mine
3. **true negative**: predicts not mine and is not a mine
4. **false negative**:Predicts not mine but turns out to be a mine (BOOM!)

In [ ]:
```
def confusion_matrix(predicted, actual, threshold):
    if len(predicted) != len(actual): return -1
    tp = 0.0
    fp = 0.0
    tn = 0.0
    fn = 0.0
    for i in range(len(actual)):
        if actual[i] > 0.5: #labels that are 1.0  (positive examples)
            if predicted[i] > threshold:
                tp += 1.0 #correctly predicted positive
            else:
                fn += 1.0 #incorrectly predicted negative
        else:                  #labels that are 0.0 (negative examples)
            if predicted[i] < threshold:
                tn += 1.0 #correctly predicted negative
            else:
                fp += 1.0 #incorrectly predicted positive
    rtn = [tp, fn, fp, tn]

    return rtn
```

```
In [29]: testing_predictions = model.predict(x_test)
         confusion_matrix(testing_predictions,np.array(y_test),0.5)

Out[29]: [27.0, 9.0, 5.0, 22.0]
```

Misclassification rate = (fp + fn)/number of cases

```
In [30]: cm = confusion_matrix(testing_predictions,np.array(y_test),0.5)
         misclassification_rate = (cm[1] + cm[2])/len(y_test)
         misclassification_rate

Out[30]: 0.2222222222222222
```

Precision and Recall

```
In [ ]: [tp, fn, fp, tn] = confusion_matrix(testing_predictions,np.array(y_test),0.5)
        precision = tp/(tp+fp)
        recall = tp/(tp+fn)
        f_score = 2 * (precision * recall)/(precision + recall)
        print(precision,recall,f_score)
```

ROC: Receiver Order Characteristic

- An ROC curve shows the performance of a binary classifier as the threshold varies.
- Computes two series:

1. False positive rate (FPR) Fall out/false alarm = False Positives/(True Negatives + False Positives)
    - Or, what proportion of rocks are identified as mines
2. True Positive rate (TPR) Sensitivity/recall = True Positives/(True Positives + False Negatives)
    - Or, what proportion of actual mines are identified as mines

- **true positive**: predicts mine and is a mine
- **false positive**: predicts mine and is not a mine
- **true negative**: predicts not mine and is not a mine
- **false negative**:Predicts not mine but turns out to be a mine (BOOM!)

Classification - Rocks and Mines

Let's first plot the predictions against actuals

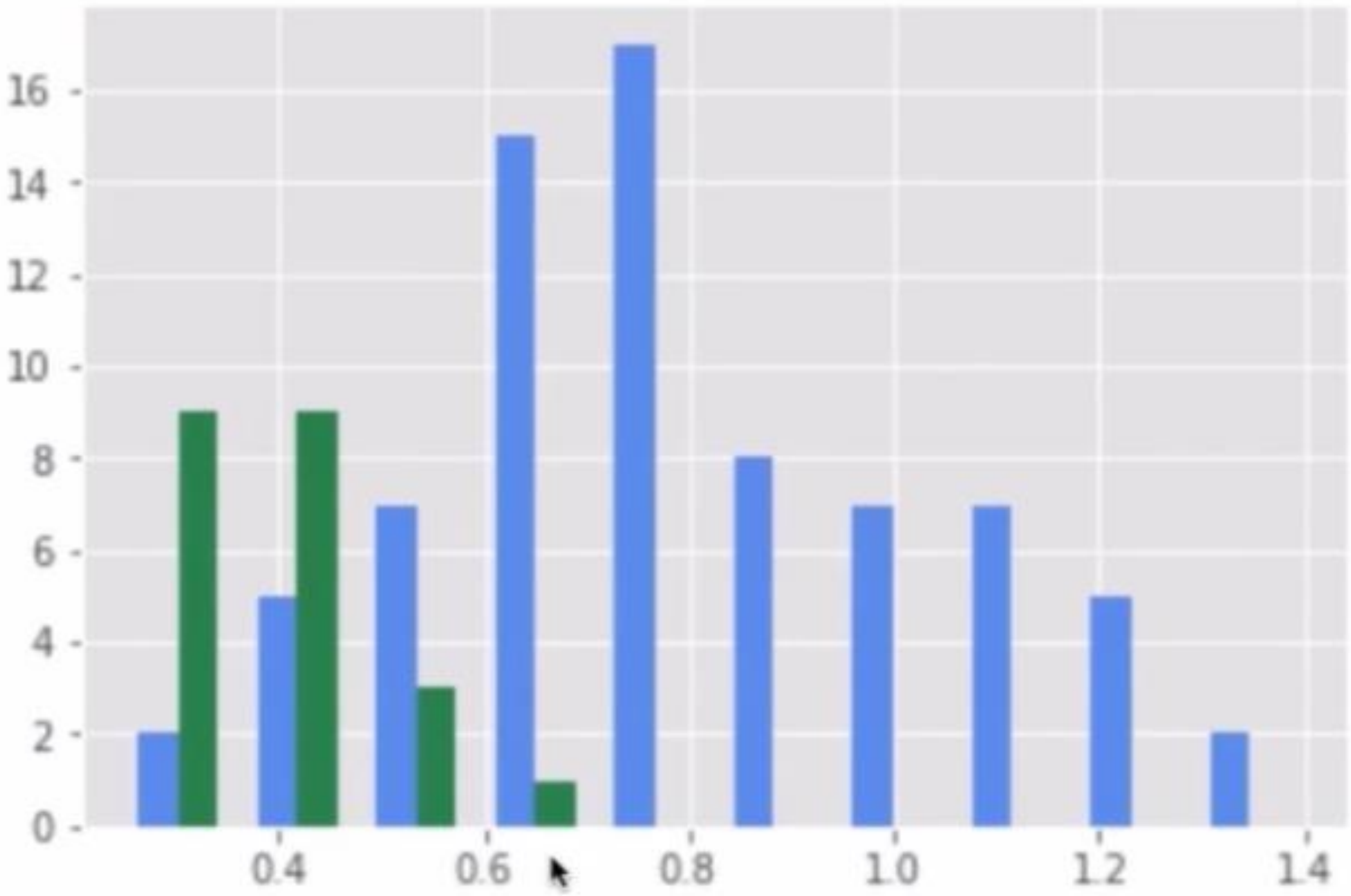
The goal is to see if our classifier has discriminated at all

In [ ]:
```
positives = list()
negatives = list()
actual = np.array(y_train)
for i in range(len(y_train)):

    if actual[i]:
        positives.append(training_predictions[i])
    else:
        negatives.append(training_predictions[i])
```

In [ ]:
```
df_p = pd.DataFrame(positives)
df_n = pd.DataFrame(negatives)
fig, ax = plt.subplots()
a_heights, a_bins = np.histogram(df_p)
b_heights, b_bins = np.histogram(df_n, bins=a_bins)
width = (a_bins[1] - a_bins[0])/3
ax.bar(a_bins[:-1], a_heights, width=width, facecolor='cornflowerblue')
ax.bar(b_bins[:-1]+width, b_heights, width=width, facecolor='seagreen'
```

Out[34]: <Container object of 10 artists>



sklearn has a function roc_curve that does this for us

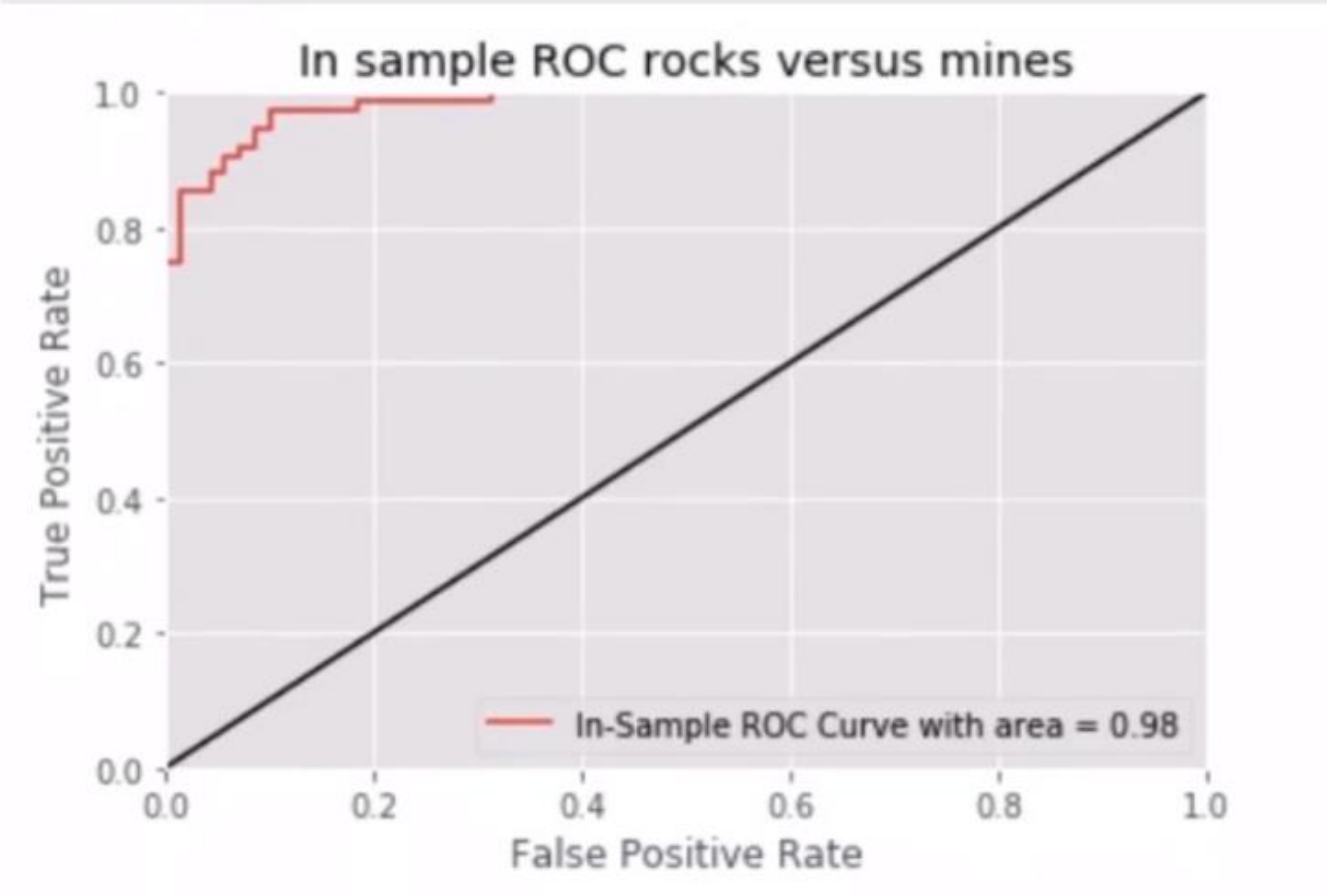
```
In [36]: from sklearn.metrics import roc_curve, auc
```

In-sample ROC Curve

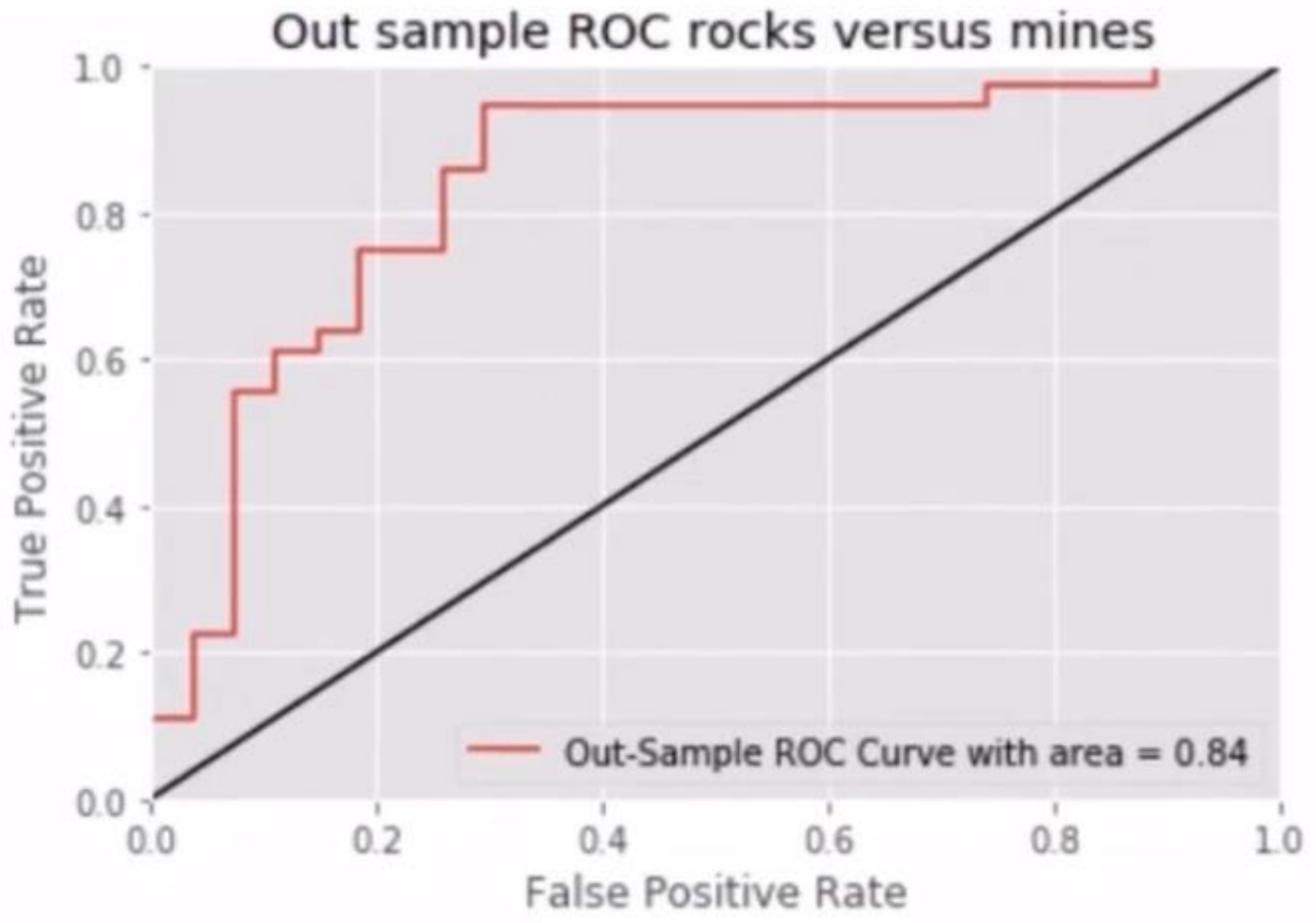
```
In [ ]: (fpr, tpr, thresholds) = roc_curve(y_train,training_predictions)
area = auc(fpr,tpr)
pl.clf() #Clear the current figure
pl.plot(fpr,tpr,label="In-Sample ROC Curve with area = %1.2f"%area)

pl.plot([0, 1], [0, 1], 'k') #This plots the random (equal probability line)
pl.xlim([0.0, 1.0])
pl.ylim([0.0, 1.0])
pl.xlabel('False Positive Rate')
pl.ylabel('True Positive Rate')
pl.title('In sample ROC rocks versus mines')
pl.legend(loc="lower right")
pl.show()
```

In – Sample ROC Curve



Out – Sample ROC Curve



EMERITUS