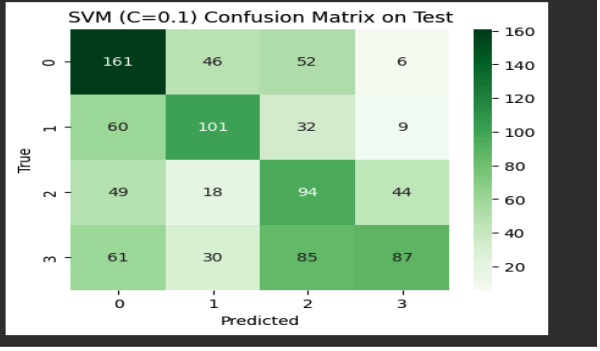
Name: Karl Azangue, Aarwin Fredericks

SVM

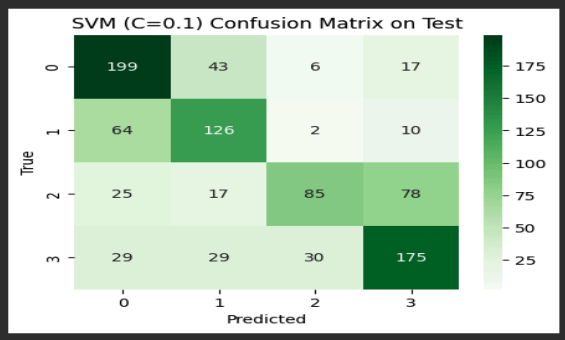
Sigmoid kernel

```
Train:       7474 samples
Validation:  934 samples
Test:        935 samples

C = 0.01    → Validation Accuracy: 0.4647
C = 0.1     → Validation Accuracy: 0.4936
C = 0.5     → Validation Accuracy: 0.4400
C = 0.7     → Validation Accuracy: 0.4358
C = 5       → Validation Accuracy: 0.4347
C = 4       → Validation Accuracy: 0.4379
C = 3       → Validation Accuracy: 0.4358
C = 7       → Validation Accuracy: 0.4304
C = 1       → Validation Accuracy: 0.4368
C = 10      → Validation Accuracy: 0.4325
C = 15      → Validation Accuracy: 0.4336

Best C: 0.1 (Validation Acc = 0.4936)

=== Best-SVM Test Results ===
              precision    recall  f1-score   support

           0       0.49      0.61      0.54       265
           1       0.52      0.50      0.51       202
           2       0.36      0.46      0.40       205
           3       0.60      0.33      0.43       263

    accuracy                           0.47       935
   macro avg       0.49      0.47      0.47       935
weighted avg       0.50      0.47      0.47       935
```



Linear kernel

```
Test:        935 samples

C = 0.01    → Validation Accuracy: 0.6370
C = 0.1     → Validation Accuracy: 0.6499
C = 0.5     → Validation Accuracy: 0.6435
C = 0.7     → Validation Accuracy: 0.6392
C = 5       → Validation Accuracy: 0.6360
C = 4       → Validation Accuracy: 0.6338
C = 3       → Validation Accuracy: 0.6328
C = 7       → Validation Accuracy: 0.6328
C = 1       → Validation Accuracy: 0.6392
C = 10      → Validation Accuracy: 0.6349
C = 15      → Validation Accuracy: 0.6338

Best C: 0.1 (Validation Acc = 0.6499)

=== Best-SVM Test Results ===
              precision    recall  f1-score   support

           0       0.63      0.75      0.68       265
           1       0.59      0.62      0.60       202
           2       0.69      0.41      0.52       205
           3       0.62      0.67      0.64       263

    accuracy                           0.63       935
   macro avg       0.63      0.61      0.61       935
weighted avg       0.63      0.63      0.62       935
```



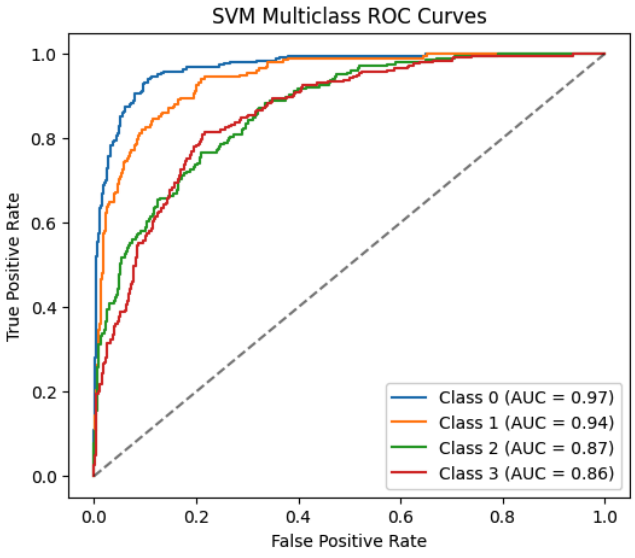

```
8]  # OLD CODE:
```

To improve model performance, we switched from a sigmoid to a linear kernel, which significantly increased test accuracy from 47% to 63% and improved overall F1-scores across all classes. This change was driven by the observation that the sigmoid kernel performed poorly due to its non-linear nature and sensitivity to feature scaling. What we plan on doing moving forward is to downscale the image inputs from 64×64 to 48×48 to reduce memory load in Google Colab, which would allow us to extract a larger set of features and run the model more efficiently, especially since we ran a variations of C's in order to find the best C value.

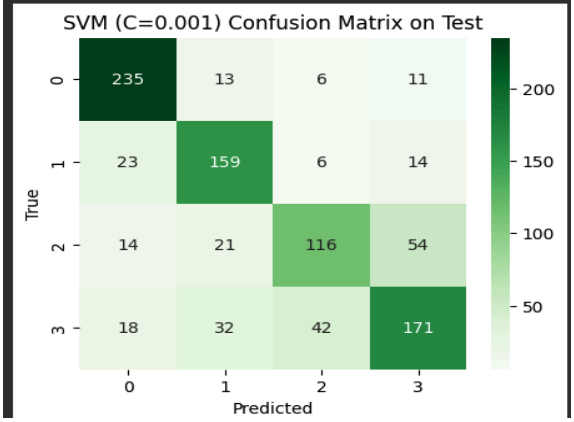
Current features:

- **Color Histogram**
- **Local Binary Pattern (LBP)**
- **Gabor Filter Stats**
- **Haralick Features**
- **Histogram of Oriented Gradients (HOG)**
- **Hu Moments**
- **Zernike Moments**
- **ORB Descriptors (mean)**
- **Region Properties**
- **Canny Edge Stats**
- **Wavelet Stats**

Linear Kernel Results:

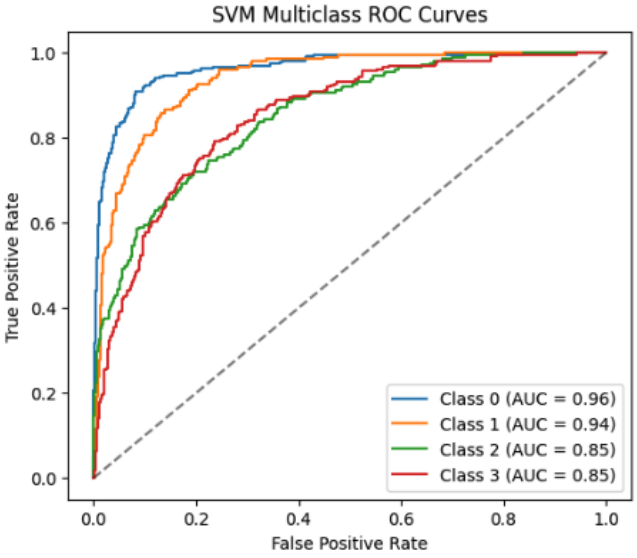



```
Best C: 0.001 (Validation Acc = 0.7281)

=== Best-SVM Test Results ===
              precision    recall  f1-score   support

           0       0.81      0.89      0.85       265
           1       0.71      0.79      0.74       202
           2       0.68      0.57      0.62       205
           3       0.68      0.65      0.67       263

    accuracy                           0.73       935
   macro avg       0.72      0.72      0.72       935
weighted avg       0.72      0.73      0.72       935
```

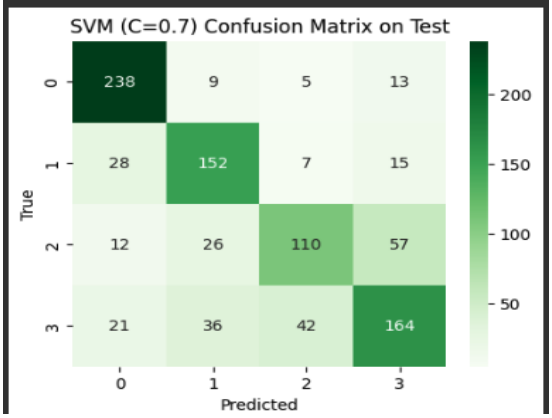


This model learns a set of straight-line decision boundaries in feature space, which effectively separate the classes when the data is linearly separable or close to it. The high accuracy (73%) and strong AUC scores suggest the features capture meaningful differences between classes and that linear separation is sufficient.

Sigmoid Kernel Results:



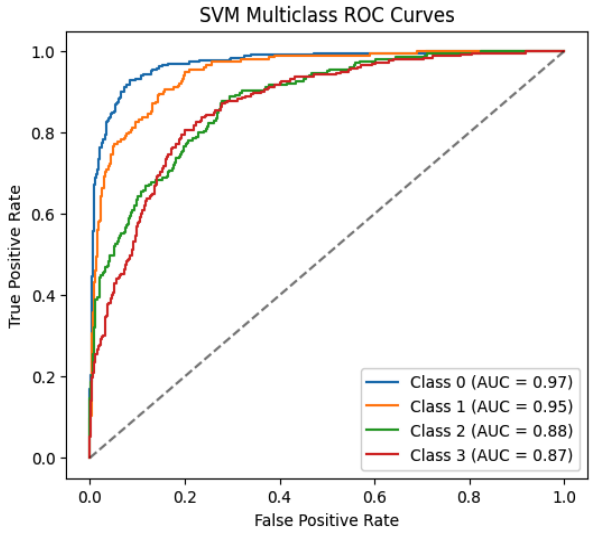

```
Best C: 0.7 (Validation Acc = 0.7195)

=== Best-SVM Test Results ===
              precision    recall  f1-score   support

           0       0.80      0.90      0.84       265
           1       0.68      0.75      0.72       202
           2       0.67      0.54      0.60       205
           3       0.66      0.62      0.64       263

    accuracy                           0.71       935
   macro avg       0.70      0.70      0.70       935
weighted avg       0.71      0.71      0.70       935
```



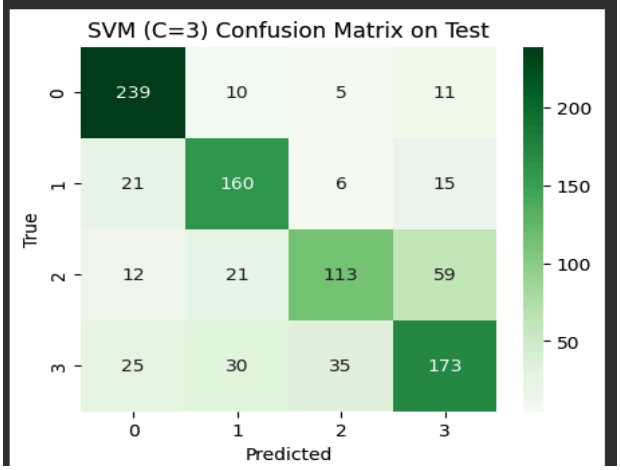
The Sigmoid kernel maps input data through a non-linear transformation similar to a shallow neural network, attempting to capture complex relationships. Although it improved after tuning, the slightly lower accuracy (71%) and less

consistent recall indicate that the non-linear mapping may have introduced unnecessary complexity, leading to more class confusion, especially for overlapping features.

rbf Kernel Results:

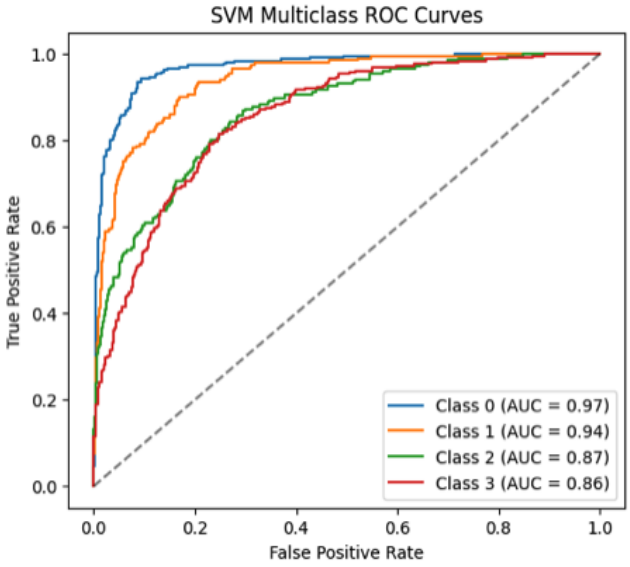



```
Best C: 3 (Validation Acc = 0.7291)

=== Best-SVM Test Results ===
              precision    recall  f1-score   support

           0       0.80      0.90      0.85       265
           1       0.72      0.79      0.76       202
           2       0.71      0.55      0.62       205
           3       0.67      0.66      0.66       263

    accuracy                           0.73       935
   macro avg       0.73      0.73      0.72       935
weighted avg       0.73      0.73      0.73       935
```



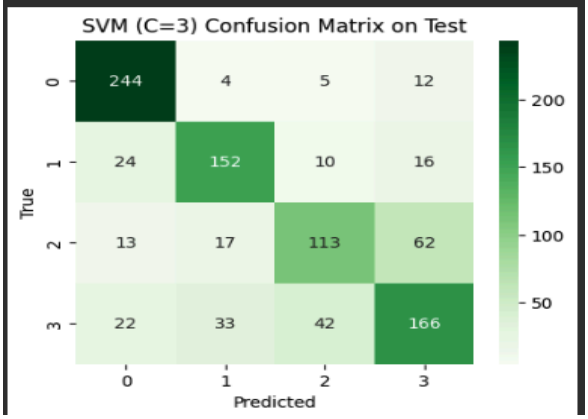
To improve model performance and reduce underfitting, we decided to reduce the number of extracted features, specifically redundant features because the model may have gotten overwhelmed by noise and failed to learn meaningful patterns. It also helps since SVM usually struggles when there are too many features that are redundant, unstable (like ORB), or don't contribute useful information.We opted for removing Hu Moments, Canny Edge Stats, and ORB Descriptors because Hu Moments are sensitive to noise, Canny Edge Stats offer limited insight compared to richer edge features like HOG, and ORB often produces zero vectors when no keypoints are found.

Results with 'rbf' kernel: Less Features (choosed 'rbf' kernel bc it performed best)



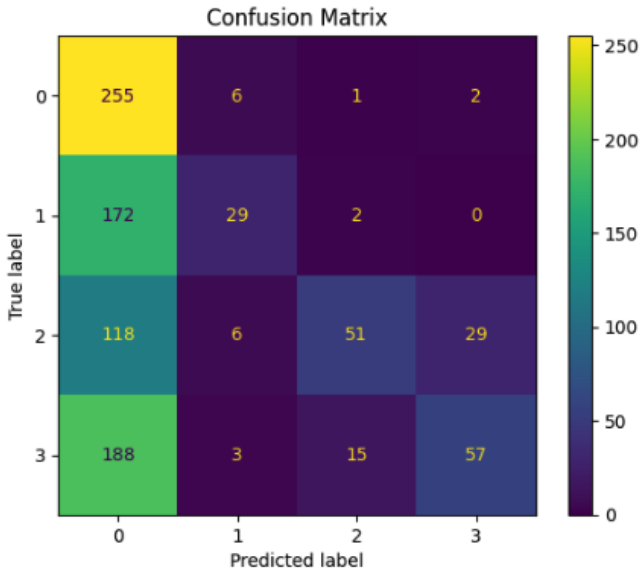
We still got similar results after running the model without the removed features with the model's accuracy being 72% only 1% lower from the previous 73%, although this did not improve our accuracy this does suggest that those added features were somewhat redundant and did not help the model be any more accurate. The ROC curves also demonstrate that the model still has strong class separation capability, with AUC values above 0.85 for all classes indicating that even if precision and

```
C = 0.0001 → Validation Accuracy: 0.2827
C = 0.001  → Validation Accuracy: 0.2827
C = 0.01   → Validation Accuracy: 0.2827
C = 0.7    → Validation Accuracy: 0.7184
C = 0.5    → Validation Accuracy: 0.7013
C = 1      → Validation Accuracy: 0.7195
C = 3      → Validation Accuracy: 0.7248
C = 4      → Validation Accuracy: 0.7238
C = 5      → Validation Accuracy: 0.7238
C = 7      → Validation Accuracy: 0.7238
C = 10     → Validation Accuracy: 0.7238
C = 15     → Validation Accuracy: 0.7238

Best C: 3 (Validation Acc = 0.7248)

=== Best-SVM Test Results ===
              precision    recall  f1-score   support

           0       0.81      0.92      0.86       265
           1       0.74      0.75      0.75       202
           2       0.66      0.55      0.60       205
           3       0.65      0.63      0.64       263

    accuracy                           0.72       935
   macro avg       0.71      0.71      0.71       935
weighted avg       0.72      0.72      0.72       935
```



recall vary across classes, the underlying decision function is well calibrated and effective at ranking predictions.

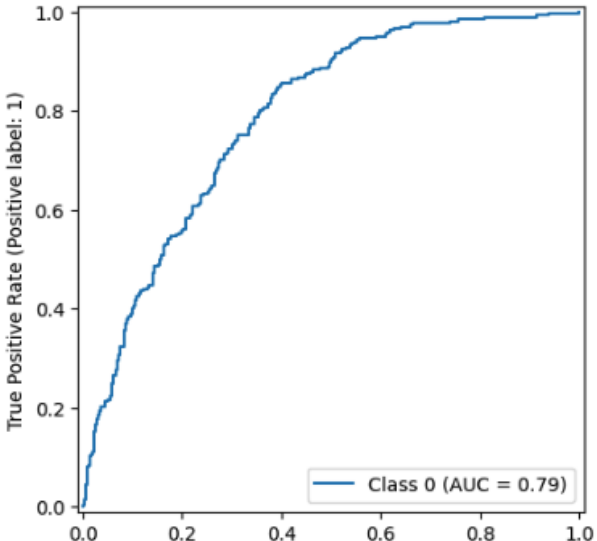
CNN

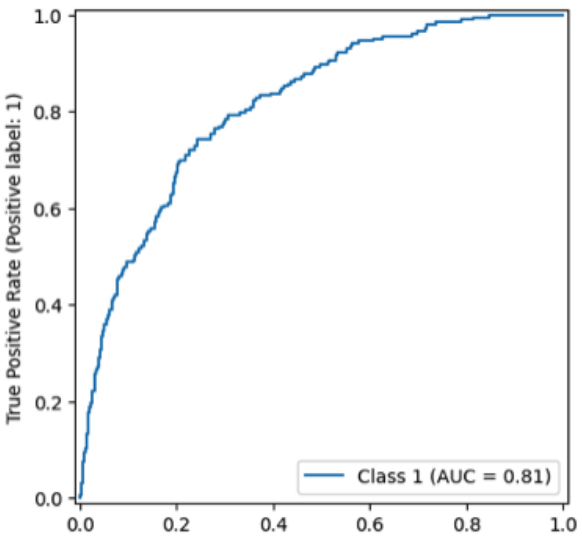


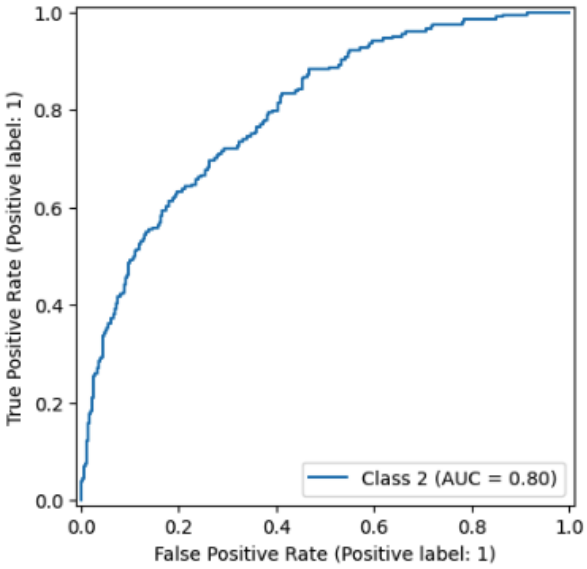
The confusion matrix shows that nearly every sample regardless of its true label is being predicted as class 0, with almost no correct predictions for classes 1, 2, or 3. This confirms that the arg max over the softmax outputs is biased

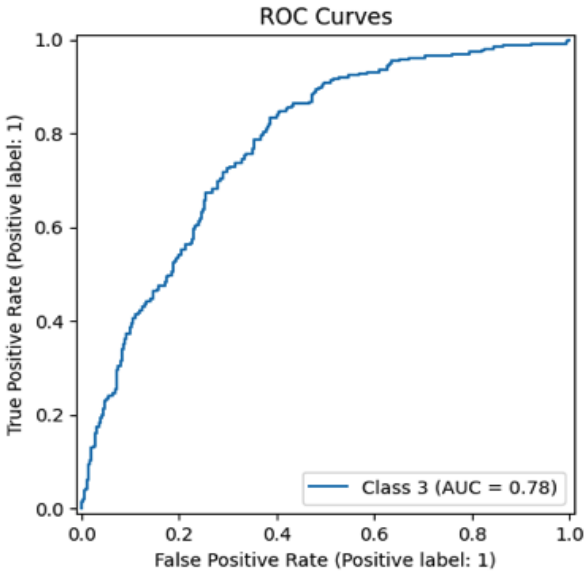
```
Epoch 18/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 79s 170ms/step - accuracy: 0.2824 - loss: 1.3778 - val_accuracy: 0.2827 - val_loss: 1.3780
Epoch 19/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 39s 168ms/step - accuracy: 0.2836 - loss: 1.3792 - val_accuracy: 0.2827 - val_loss: 1.3780
Epoch 20/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 41s 166ms/step - accuracy: 0.2865 - loss: 1.3804 - val_accuracy: 0.2816 - val_loss: 1.3781
Epoch 21/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 42s 172ms/step - accuracy: 0.2796 - loss: 1.3794 - val_accuracy: 0.2827 - val_loss: 1.3780
Epoch 22/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 39s 167ms/step - accuracy: 0.2832 - loss: 1.3758 - val_accuracy: 0.2827 - val_loss: 1.3782
Epoch 23/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 41s 167ms/step - accuracy: 0.2822 - loss: 1.3788 - val_accuracy: 0.2827 - val_loss: 1.3781
Epoch 24/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 41s 167ms/step - accuracy: 0.2854 - loss: 1.3784 - val_accuracy: 0.2816 - val_loss: 1.3780
Epoch 25/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 41s 167ms/step - accuracy: 0.2900 - loss: 1.3761 - val_accuracy: 0.2816 - val_loss: 1.3781
<keras.src.callbacks.history.History at 0x7bf7c628df50>
```

By epoch 25 both training and validation accuracies stall at about 28% and the loss remains about 1.38. This shows that the model is stuck predicting all classes equally rather than learning meaningful distinctions.









Across all four classes, the one-vs-rest ROC curves yield AUCs in the 0.78-0.81 range, showing that the network has indeed learned features that distinguish each category even though it ends up predicting almost everything as class 0. The model's embeddings capture separability, but the softmax thresholding collapses to the dominant class instead of leveraging that discriminative power.

> We need to ensure the loss function are correctly matched and then reintroduce batch normalization and manage the learning rate with a gradual decay schedule

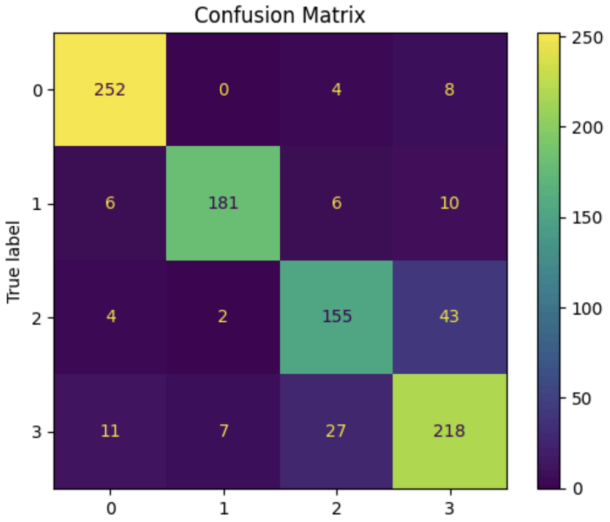
Second Iteration:

```
Epoch 1/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 138s 562ms/step - accuracy: 0.4579 - loss: 1.3613 - val_accuracy: 0.2173 - val_loss: 4.5506 - learning_rate: 1.0000e-04
Epoch 2/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 130s 555ms/step - accuracy: 0.5765 - loss: 0.9502 - val_accuracy: 0.2173 - val_loss: 4.6110 - learning_rate: 1.0000e-04
Epoch 3/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 128s 547ms/step - accuracy: 0.6210 - loss: 0.8623 - val_accuracy: 0.3137 - val_loss: 1.3290 - learning_rate: 1.0000e-04
Epoch 4/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 125s 535ms/step - accuracy: 0.6607 - loss: 0.7735 - val_accuracy: 0.2827 - val_loss: 8.3909 - learning_rate: 1.0000e-04
Epoch 5/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 141s 530ms/step - accuracy: 0.6893 - loss: 0.7393 - val_accuracy: 0.3019 - val_loss: 4.7596 - learning_rate: 1.0000e-04
Epoch 6/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 0s 521ms/step - accuracy: 0.7424 - loss: 0.6333
Epoch 6: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
234/234 ━━━━━━━━━━━━━━━━━━━━ 145s 543ms/step - accuracy: 0.7424 - loss: 0.6333 - val_accuracy: 0.2184 - val_loss: 3.8151 - learning_rate: 1.0000e-04
Epoch 7/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 137s 522ms/step - accuracy: 0.7684 - loss: 0.5717 - val_accuracy: 0.3683 - val_loss: 2.7820 - learning_rate: 5.0000e-05
Epoch 8/25
234/234 ━━━━━━━━━━━━━━━━━━━━ 144s 529ms/step - accuracy: 0.7881 - loss: 0.5192 - val_accuracy: 0.4443 - val_loss: 1.8904 - learning_rate: 5.0000e-05
```

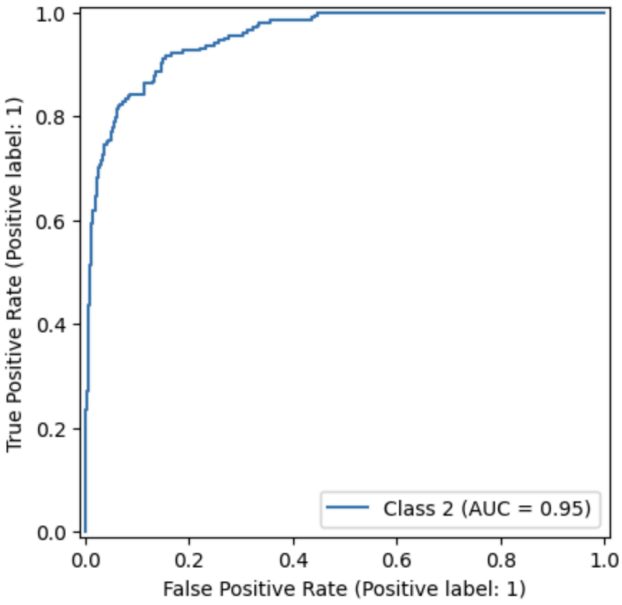
We got these results after introducing targeted augmentations or class weighted/focal loss, callbacks, normalization and Max pooling after every convolutional layer. I also implemented Early Stoppage, ReduceLR On Plateau that cuts the learning rate in half whenever the validation loss

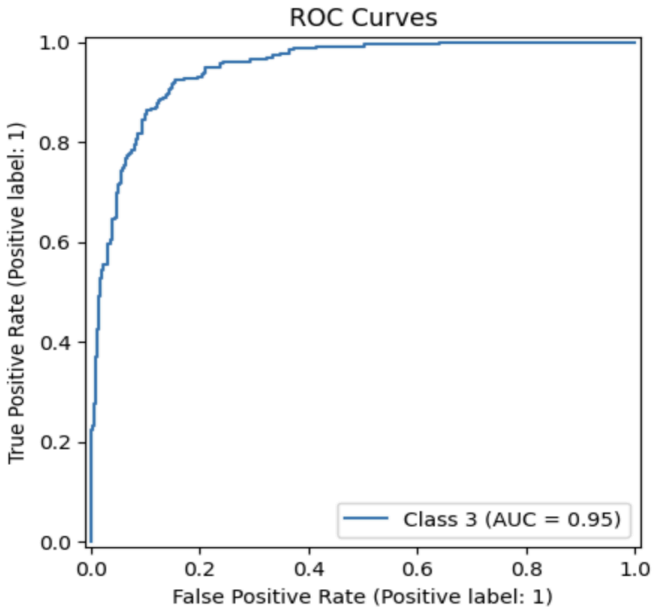
plateaus and Model Checkpoint that saves the version of your model that achieves the highest validation accuracy.

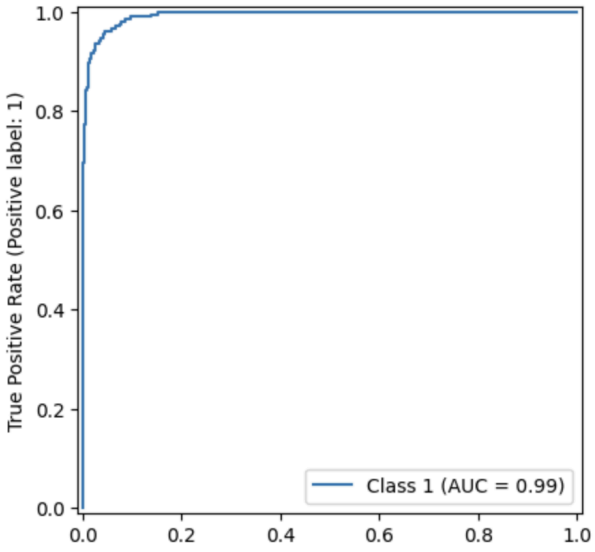
Third Iteration:

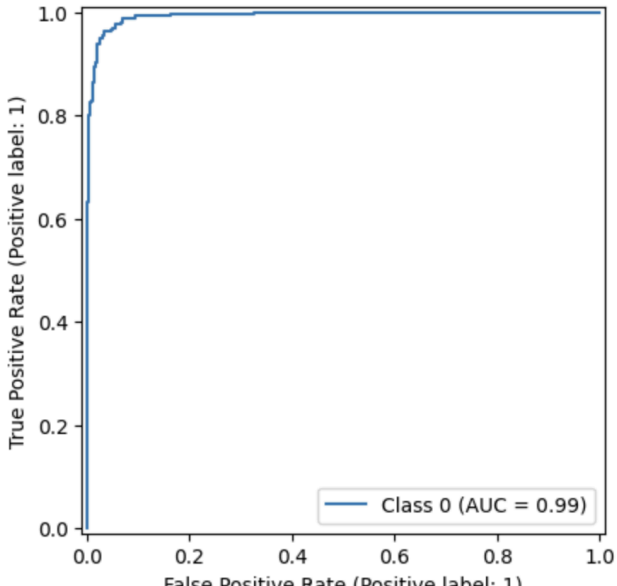


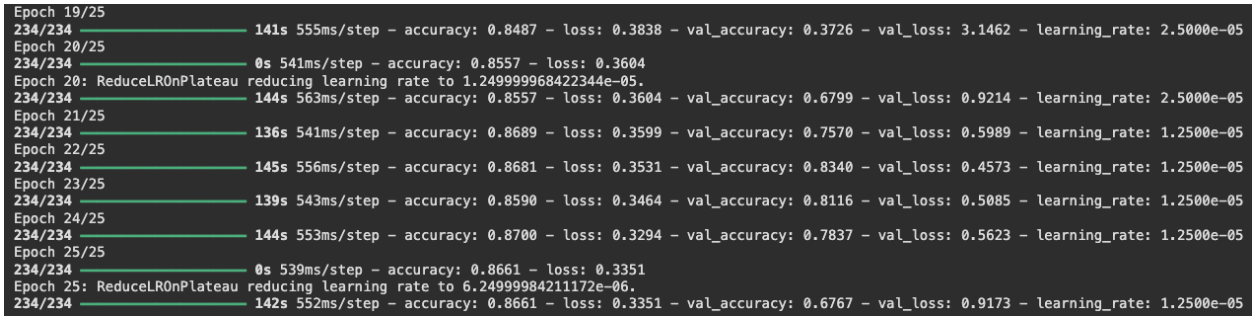
Confusion matrix and ROC show that the network easily separates classes 0 and 1 (AUC = 0.99) but struggles with classes 2 and 3 (AUC = 0.95), so many true2s are called 3 and vice versa. the broad augmentations (flips, rotations, zooms) may be washing out the subtle, orientation specific features that distinguish those two classes.







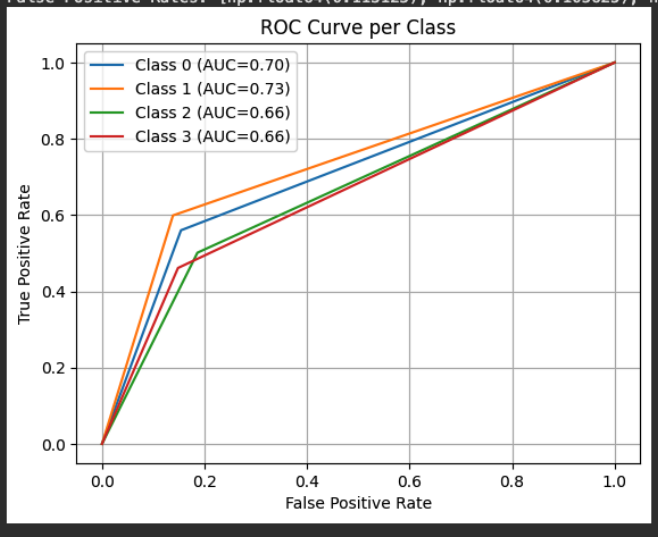




We removed the loss L2 penalty on conv Layers by about 60%, decreased the dropout to 0.1–0.2 instead of 0.3–0.5. We then lowered the initial learning rate to 1e-4 (so updates don’t overshoot) and added an extra Conv2D block with more filters 256. The high AUC values across all classes (ranging from 0.95 to 0.99) indicate excellent class separability, meaning the network is quite good at differentiating between the categories. The confusion matrix confirms this, with strong diagonal dominance and minimal misclassification especially for classes 0 and . Some confusion still occurs between visually similar classes class 2 and 3, which could be improved by further refining the feature extraction or increasing data diversity.

Decision Trees

```
Accuracy: 0.53

Confusion Matrix:
 [[238  84  62  41]
 [ 75 227  41  36]
 [ 59  38 199 101]
 [ 47  47 121 184]]

Average Confusion Value: 100.0

True Positive Rates: [0.56       0.59894459 0.50125945 0.46115288]

False Positive Rates: [np.float64(0.113125), np.float64(0.105625), np.float64(0.14), np.float64(0.11125)]
```

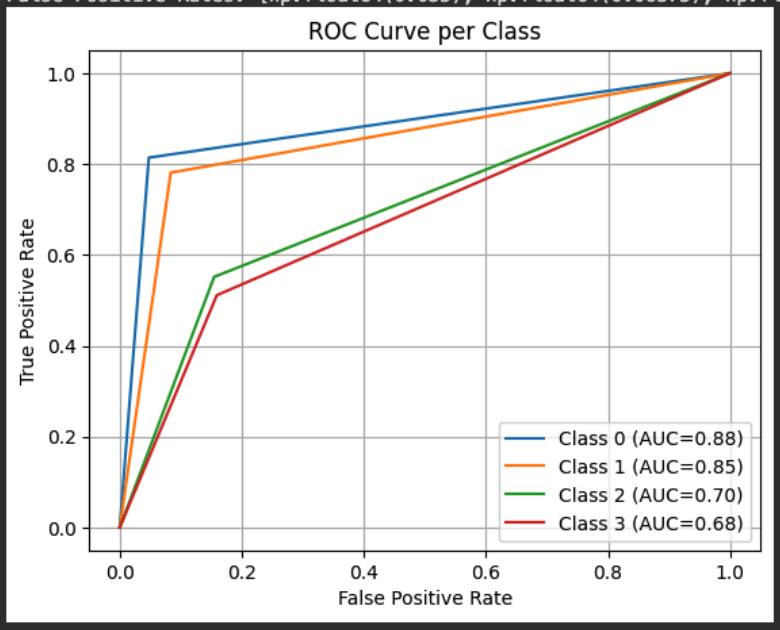


Class 0 (2645 images): Round Smooth Galaxies
Class 1 (2027 images): In-between Round Smooth Galaxies
Class 2 (2043 images): Barred Spiral Galaxies
Class 3 (2628 images): Unbarred Loose Spiral Galaxies

With the features we originally planned to, we got 53% accuracy. We didn't know why this was happening, so we just kept trying new things from here. Interesting thing here is that it was able to verify Class 0(Round Smooth Galaxies) and Class 1(In-between Round Smooth Galaxies) the best, but was struggling with the other classes(Barred Spiral Galaxies and Unbarred Spiral Galaxies).

To counter this problem, we added 2 more features. I added Zernike Moments and Haralick Texture. Zernike Moments capture the global shape/ symmetry of the galaxy and are good for detecting overall galaxy structure symmetry(Round vs. spiral).

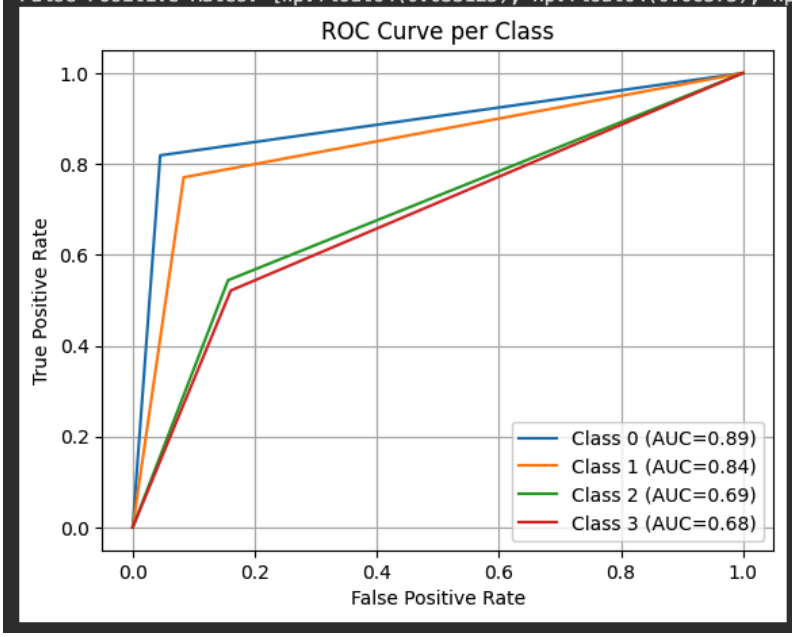
```
Accuracy: 0.665625

Confusion Matrix:
 [[346  19  29  31]
 [ 11 296  37  35]
 [ 20  33 219 125]
 [ 25  50 120 204]]

Average Confusion Value: 100.0

True Positive Rates: [0.81411765 0.78100264 0.55163728 0.5112782 ]

False Positive Rates: [np.float64(0.035), np.float64(0.06375), np.float64(0.11625), np.float64(0.119375)]
```



This increases the overall accuracy and the area under the ROC curves. This was pretty significant in my findings because it was a big jump in accuracy, from 53% all the way to 66.5%.

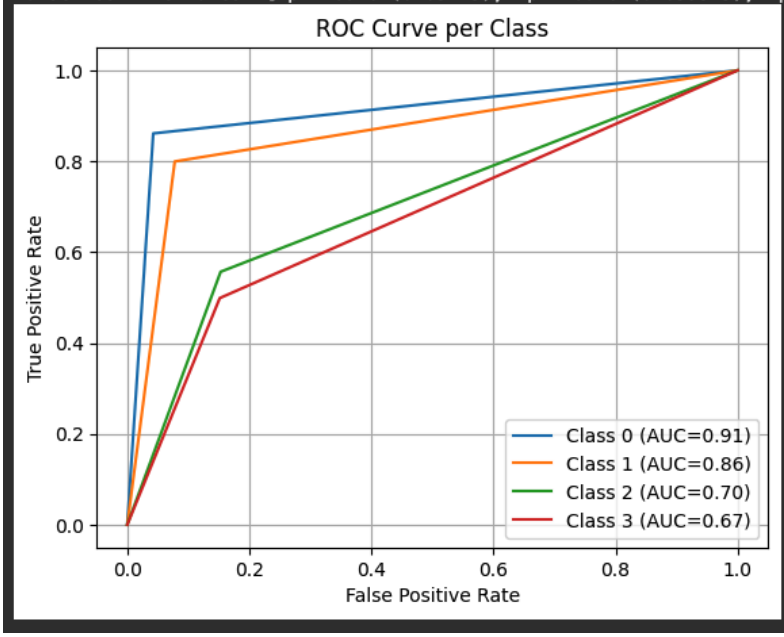
After adding this accuracy jump, we decided to add one more, Image Entropy. Image Entropy would capture the intensity randomness of the galaxy images and is good for detecting visual complexity or disorder of the galaxy.

```
Accuracy: 0.665

Confusion Matrix:
 [[348  20  29  28]
 [ 15 292  34  38]
 [ 16  38 216 127]
 [ 22  44 125 208]]

Average Confusion Value: 100.0

True Positive Rates: [0.81882353 0.77044855 0.5440806  0.52130326]

False Positive Rates: [np.float64(0.033125), np.float64(0.06375), np.float64(0.1175), np.float64(0.120625)]
```



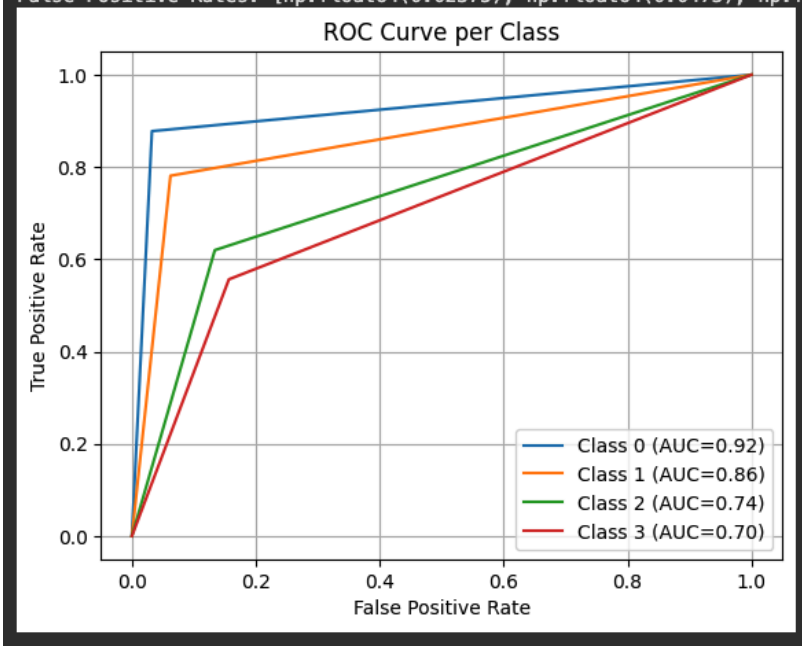
That feature really didn't do anything, so we decided to add another feature and see if it would make a difference. We decided to add Fractal Dimension, which captures how complicated each galaxy structure is, and is good for identifying spiral galaxies.

```
Accuracy: 0.680625

Confusion Matrix:
 [[366  12  21  26]
 [  6 303  34  36]
 [ 21  35 221 120]
 [ 23  48 129 199]]

Average Confusion Value: 100.0

True Positive Rates: [0.86117647 0.7994723  0.55667506 0.49874687]

False Positive Rates: [np.float64(0.03125), np.float64(0.059375), np.float64(0.115), np.float64(0.11375)]
```



This increased by only 2%, but it still made a difference, so then we added one more feature. We then added Fourier Transform Energy, which captures the strength of edges and patterns, and it detects sharp strus in spirals vs smooth ellipses. But did absolutely nothing. So we then decided to look at our current features and see what we can tweak to give better accuracy. We figured we would extend the Zernike Moments Features and a variance portion that would capture the change in shape from center to edge, and is good for differentiating smooth vs spiral arms.

```
Accuracy: 0.710625

Confusion Matrix:
 [[373   9  15  28]
 [  3 296  31  49]
 [ 12  28 246 111]
 [ 23  39 115 222]]

Average Confusion Value: 100.0
Average Normalized Confusion Entry: 0.25

True Positive Rates: [0.87764706 0.78100264 0.61964736 0.55639098]

False Positive Rates: [np.float64(0.02375), np.float64(0.0475), np.float64(0.100625), np.float64(0.1175)]
```



This increased our accuracy to 71%. From the Confusion Matrix, we can see that Class 0(Round Smooth Galaxies) has the best accuracy of 89%. This tells us that the class is very strong and only makes a few errors. Class 1(In between Round-Smooth Galaxies) has a decent accuracy of 78%. But it is confused with Classes 2 and 3. Class 2(Barred Spiral Galaxies) didn't do so well and had an accuracy of 62%. It has moderate confusion, especially with class 3. Class 3(Barred Spiral Galaxies) was the weakest class with the lowest accuracy of 56%. It's frequently misclassified as class 2.

I decided to keep our features the same for KNN so that we can fully compare how each model does with the same features.

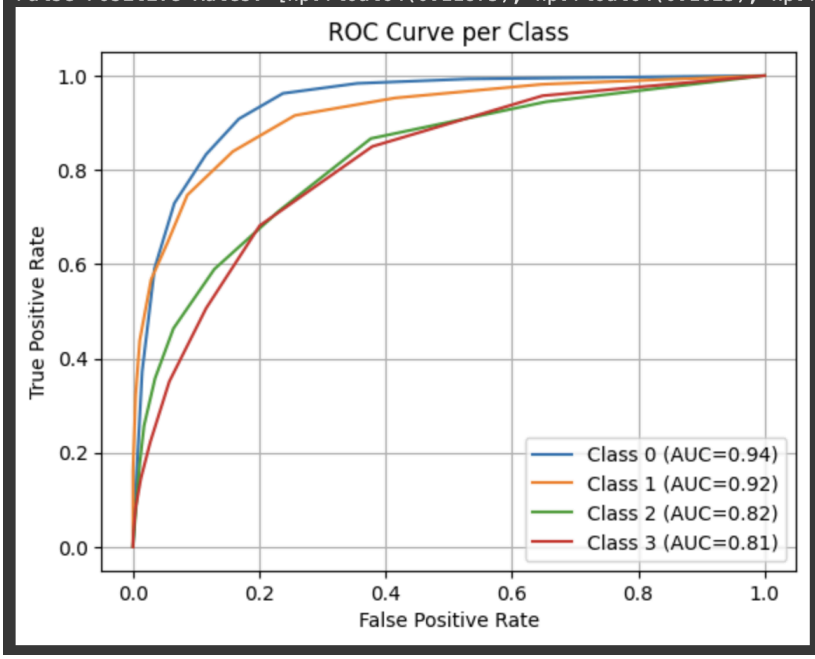
KNN

To see which nearest neighbors worked best, I used a loop to print the accuracy for each nearest neighbor.

```
k = 1, Accuracy = 0.5744
k = 2, Accuracy = 0.5850
k = 3, Accuracy = 0.6112
k = 4, Accuracy = 0.6275
k = 5, Accuracy = 0.6362
k = 6, Accuracy = 0.6425
k = 7, Accuracy = 0.6587
k = 8, Accuracy = 0.6581
k = 9, Accuracy = 0.6600
k = 10, Accuracy = 0.6600
k = 11, Accuracy = 0.6562
k = 12, Accuracy = 0.6569
k = 13, Accuracy = 0.6525
k = 14, Accuracy = 0.6519
k = 15, Accuracy = 0.6575
k = 16, Accuracy = 0.6494
k = 17, Accuracy = 0.6462
k = 18, Accuracy = 0.6519
k = 19, Accuracy = 0.6519

 Best k: 9 with Accuracy = 0.6600
```

So since the best accuracy was k = 9 and k = 10, we set our k to 10. Smaller k wasn't as accurate because smaller k values are known to have a low bias and high variance, which is sensitive to noise and leads to overfitting. Large k wasn't as accurate because large k are known to have high bias and low variance, which may underfit and ignore local structure. Medium k like 9 was probably the best because it is the most balanced between small and large k and has the best generalization to unseen data.

```
Accuracy: 0.66

Confusion Matrix:
 [[385  12  21   7]
 [ 67 299   8   5]
 [ 49  70 214  64]
 [ 74  82  85 158]]

Average Confusion Value: 100.0

True Positive Rates: [0.90588235 0.78891821 0.53904282 0.39598997]

False Positive Rates: [np.float64(0.11875), np.float64(0.1025), np.float64(0.07125), np.float64(0.0475)]
```



Interestingly, though our area under the curves was high, the accuracy was only 66%. This means our KNN model is doing a decent job learning the probability structure, but it's struggling to assign high enough probability to the true class, especially for Classes 2 & 3(Barred and Unbarred Spiral Galaxies). From the Confusion Matrix, we can see that Class 0(Round Smooth Galaxies) has the best accuracy of 91%. This tells us that the class is very strong and only makes a few errors. Class 1(In between Round-Smooth Galaxies) has a decent accuracy of 79%. But it is confused with Class 0. Class 2(Barred Spiral Galaxies) didn't do so well and had an accuracy of 54%. It has moderate confusion, especially with class 3. Class 3(Unbarred Spiral Galaxies) was the weakest class with the lowest accuracy of 40%. It's frequently misclassified with the other classes..

After reviewing the results from my part of the project, especially the confusion matrix, I realized I made a huge mistake of trying to focus on the accuracy as a whole rather than breaking down which class was struggling. I think I should've put in more features regarding spiral galaxies to help with classes 2 and 3. After looking at the confusion matrix and ROC curves for KNN and Decision Trees, I realized that mainly Class 0 and Class 1 were getting perfected in identifying, especially Class 0. If I had more time, i would've replaced or inserted more features that would help with identifying spiral galaxies. But it makes sense why round smooth galaxies have the

most accuracy, it's a simple circle. In between round smooth would more likely be classified as round smooth than the reverse because it's not exactly round, but it can be classified if one of its images has higher roundness than the others. But the round, smooth galaxies are less likely to be classified as in between round because they are generally almost round for each of their images. Barred Spiral and Unbarred Lose Spiral Galaxies are having the most errors because they are constantly being misclassified, most likely because they are both spiral, and maybe the features don’t exactly help with differentiating spirals. Honestly, the biggest problem in my algorithms and features is not being able to differentiate the spirals in spiral galaxies. Another thing is that we used only 2000 images per algorithm, and our professor recommended at least trying to get 10000 images if possible, maybe the algorithm would have data to learn and identify the differences.