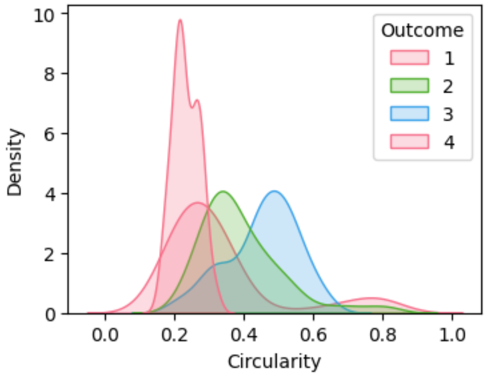
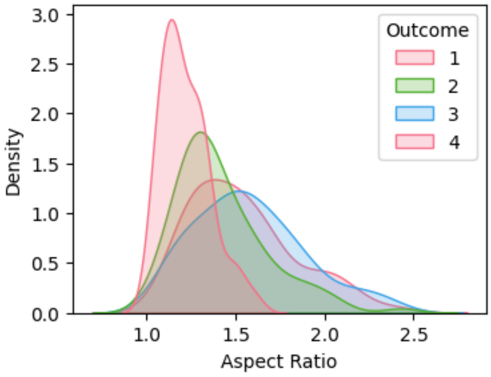
```
#Entire dataset

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
import numpy as np


# Step 1: Load the dataset
data = pd.read_excel("newww_weed1 (3).xlsx")

# Step 2: Set up k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Step 3: Perform k-fold cross-validation
accuracies = []
predi=[]


for fold, (train_index, test_index) in enumerate(kf.split(data), 1):
    train_data, test_data = data.iloc[train_index].copy(), data.iloc[test_index].copy()

    # Generate density plots for training data
    features = train_data.drop(columns=['Outcome'])

    for feature in features.columns:
      plt.figure(figsize=(4, 3))
      sns.kdeplot(data=train_data, x=feature, hue='Outcome', fill=True, common_norm=False, palette="husl")
      plt.title(f'Density plot of {feature} (Training Data - Fold {fold})')
      plt.show()
      print()

    # Calculate spread factor for training data
    spread_factors = train_data.groupby('Outcome').apply(lambda x: x.mean().tolist())

    print("Spread factor for fold",fold)
    print(list(spread_factors))

    # Find correct threshold for training data
    threshold_multiplier = 1.5
    thresholds = {cls: [spread_factors.loc[cls][i] * threshold_multiplier for i in range(len(spread_factors.iloc[0]))] for cls in spread_
    print("Thresholds for fold",fold)
    print(thresholds)

    def find_class(row, thresholds):
        max_distance = float('-inf')
        predicted_class = 'Unknown'

        for cls, class_thresholds in thresholds.items():
            distance = sum(1 for i, value in enumerate(row) if value < class_thresholds[i])
            if distance > max_distance:
                max_distance = distance
                predicted_class = cls
        return predicted_class

    predictions = test_data.drop(columns='Outcome').apply(lambda row: find_class(row, thresholds), axis=1)
    print("Predictions")
    print(predictions)

    # Evaluate the predictions
    test_labels = test_data['Outcome']
    accuracy = sum(1 for pred, label in zip(predictions, test_labels) if pred == label) / len(test_labels)
    accuracies.append(accuracy)
    predi.append(predictions)
    acc=max(accuracies)
    print()
```

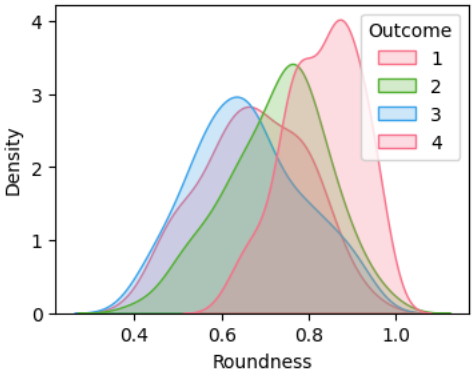
Density plot of Circularity (Training Data - Fold 1)



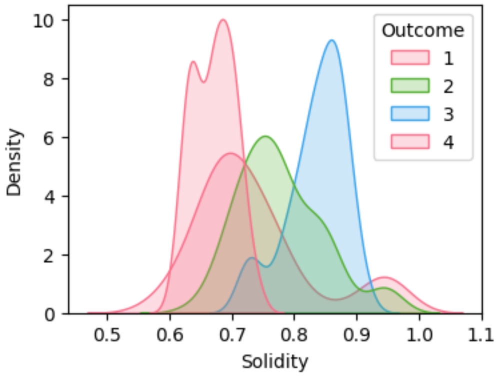
Density plot of Aspect Ratio (Training Data - Fold 1)



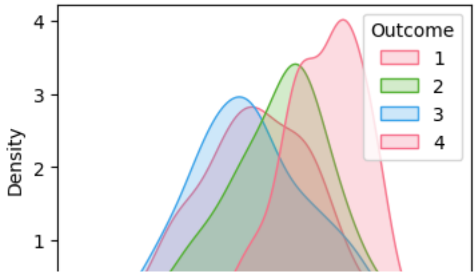
Density plot of Roundness (Training Data - Fold 1)

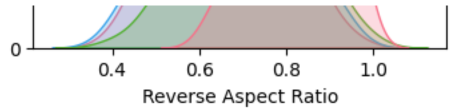


Density plot of Solidity (Training Data - Fold 1)

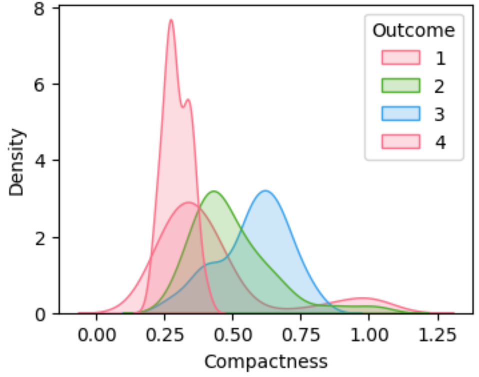


Density plot of Reverse Aspect Ratio (Training Data - Fold 1)

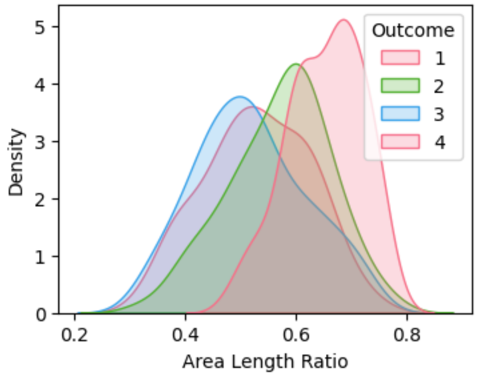




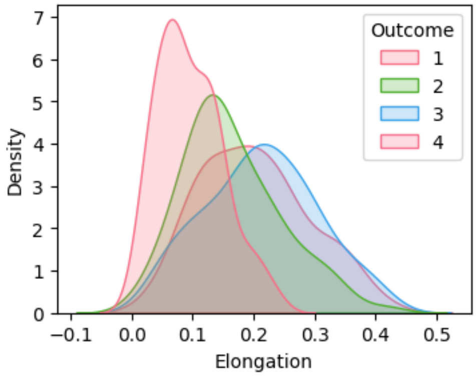
Density plot of Compactness (Training Data - Fold 1)



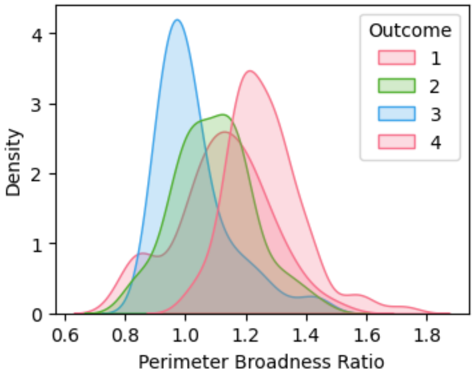
Density plot of Area Length Ratio (Training Data - Fold 1)



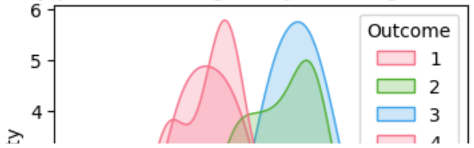
Density plot of Elongation (Training Data - Fold 1)

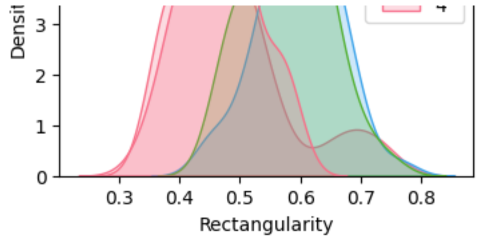


Density plot of Perimeter Broadness Ratio (Training Data - Fold 1)

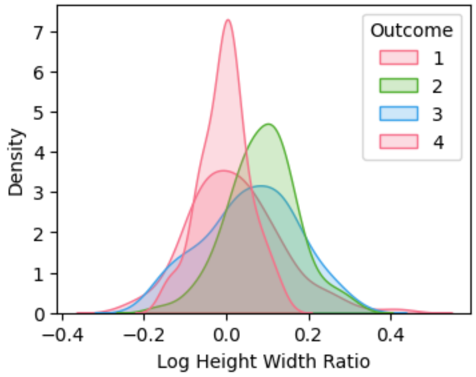


Density plot of Rectangularity (Training Data - Fold 1)

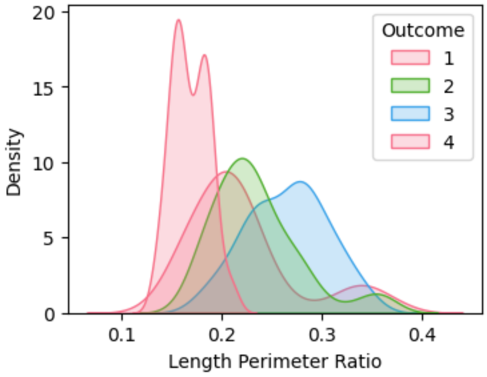




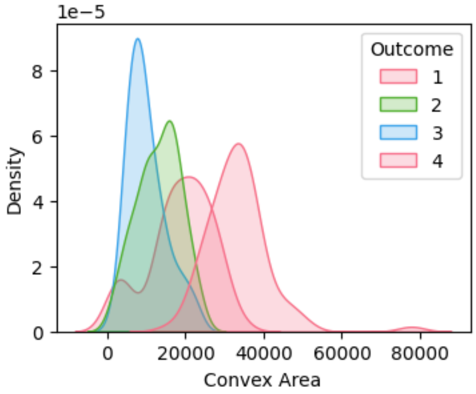
Density plot of Log Height Width Ratio (Training Data - Fold 1)



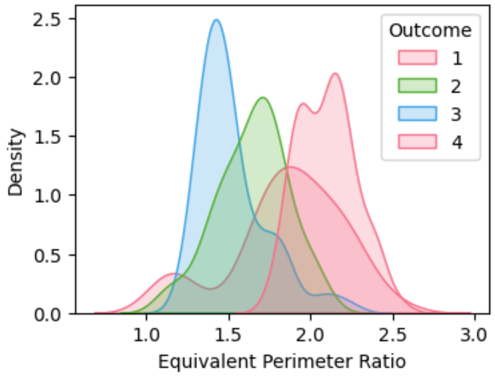
Density plot of Length Perimeter Ratio (Training Data - Fold 1)



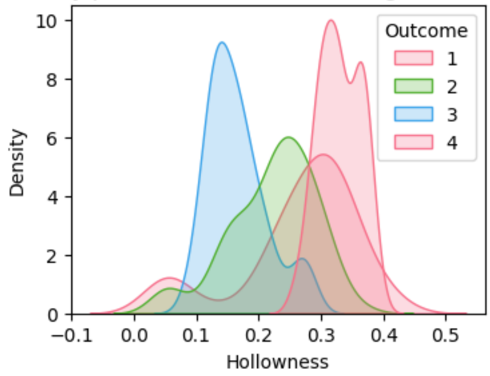
Density plot of Convex Area (Training Data - Fold 1)



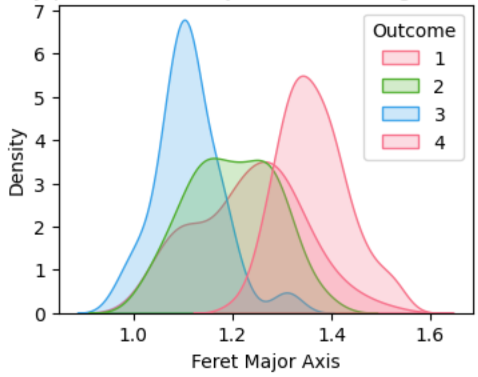
Density plot of Equivalent Perimeter Ratio (Training Data - Fold 1)



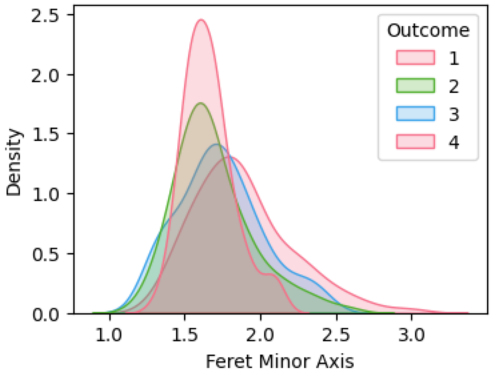
Density plot of Hollowness (Training Data - Fold 1)



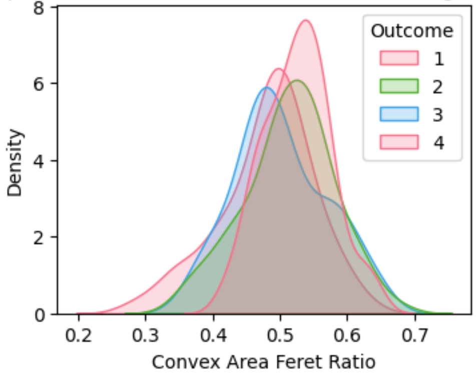
Density plot of Feret Major Axis (Training Data - Fold 1)



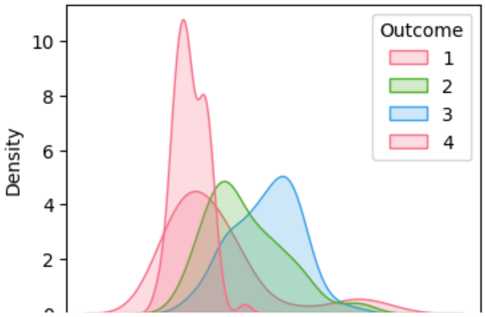
Density plot of Feret Minor Axis (Training Data - Fold 1)

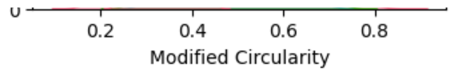


Density plot of Convex Area Feret Ratio (Training Data - Fold 1)

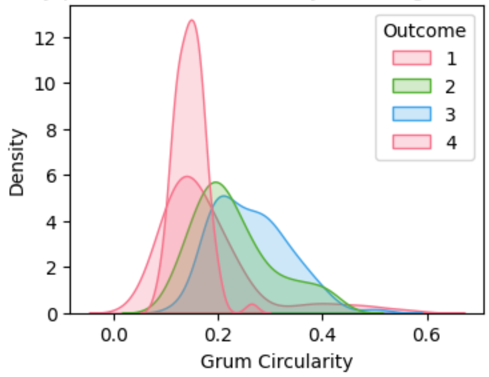


Density plot of Modified Circularity (Training Data - Fold 1)

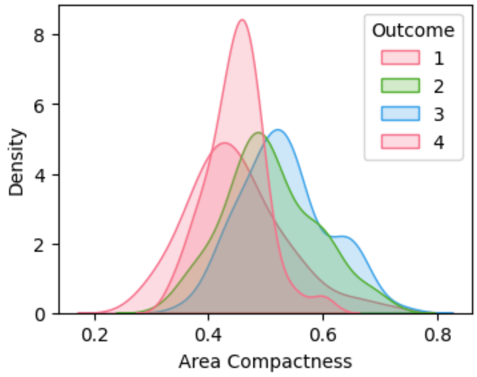




Density plot of Grum Circularity (Training Data - Fold 1)

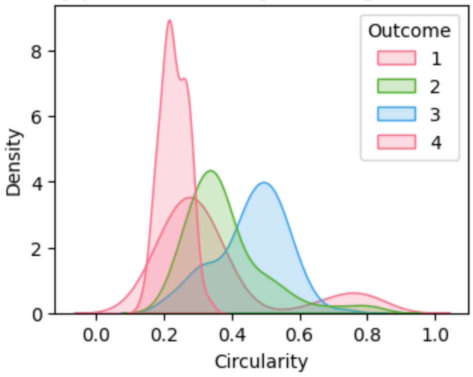


Density plot of Area Compactness (Training Data - Fold 1)

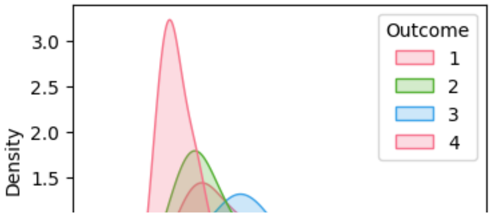


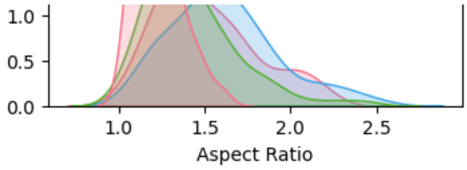
```
Spread factor for fold 1
[[0.23187654320987655, 1.2181975308641977, 0.8302716049382716, 0.6704074074074075, 0.8302839506172838, 0.2952, 0.6521024691358025, 0
Thresholds for fold 1
{1: [0.3478148148148148, 1.8272962962962964, 1.2454074074074073, 1.0056111111111112, 1.2454259259259257, 0.4428, 0.9781537037037038,
Predictions
0      1
9      3
15     1
17     2
19     1
      ..
412    2
415    3
416    4
419    1
420    3
Length: 85, dtype: int64
```

Density plot of Circularity (Training Data - Fold 2)

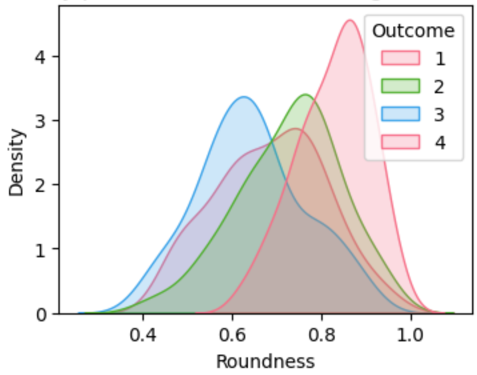


Density plot of Aspect Ratio (Training Data - Fold 2)

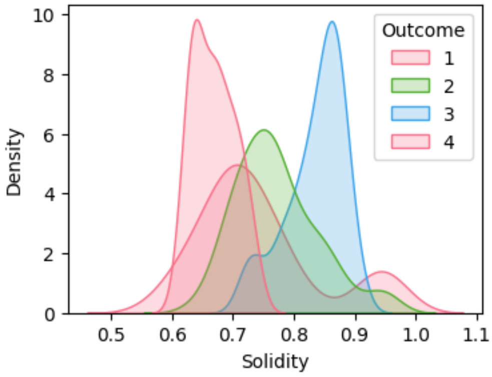




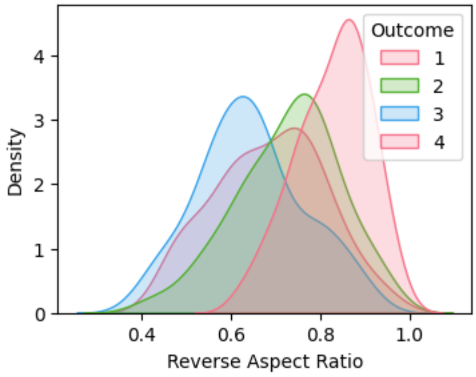
Density plot of Roundness (Training Data - Fold 2)



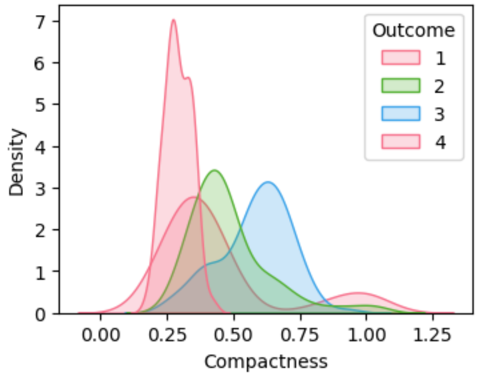
Density plot of Solidity (Training Data - Fold 2)



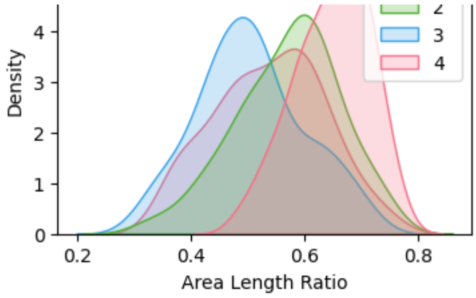
Density plot of Reverse Aspect Ratio (Training Data - Fold 2)



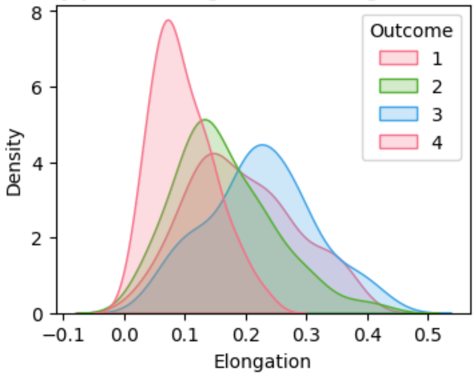
Density plot of Compactness (Training Data - Fold 2)



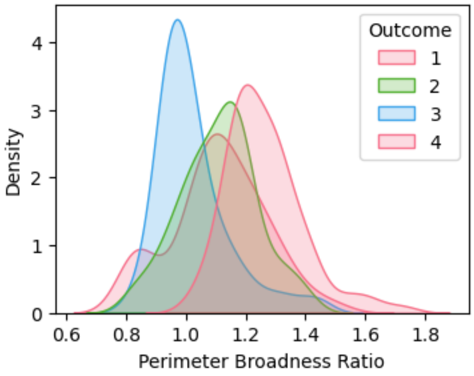
Density plot of Area Length Ratio (Training Data - Fold 2)



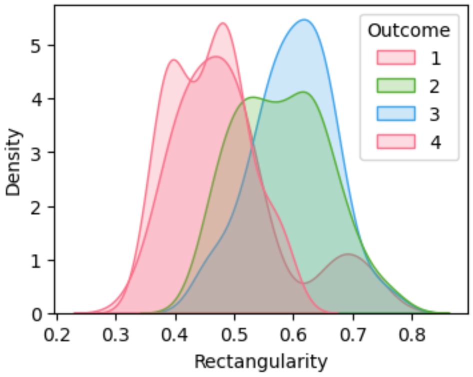
Density plot of Elongation (Training Data - Fold 2)



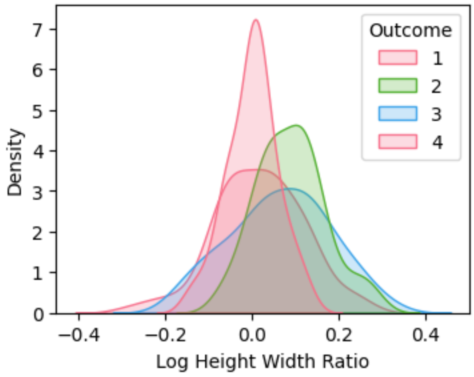
Density plot of Perimeter Broadness Ratio (Training Data - Fold 2)



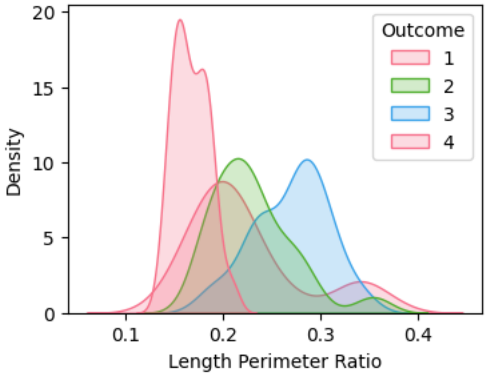
Density plot of Rectangularity (Training Data - Fold 2)



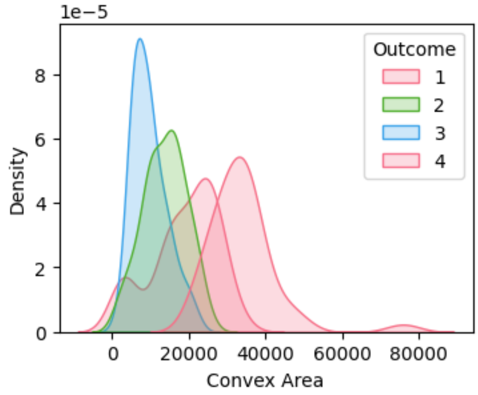
Density plot of Log Height Width Ratio (Training Data - Fold 2)



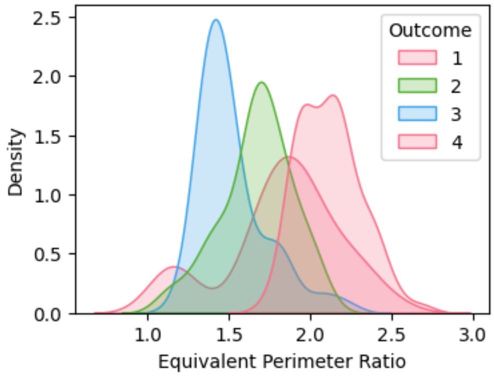
Density plot of Length Perimeter Ratio (Training Data - Fold 2)



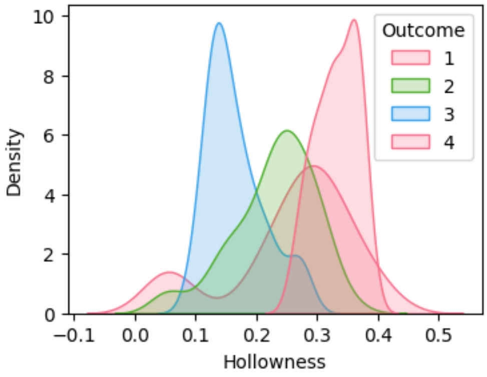
Density plot of Convex Area (Training Data - Fold 2)



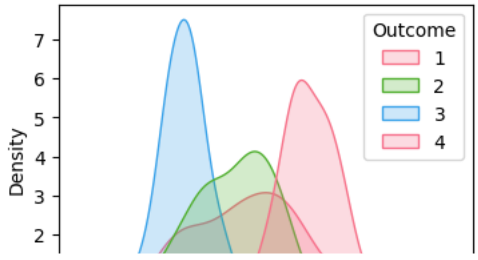
Density plot of Equivalent Perimeter Ratio (Training Data - Fold 2)

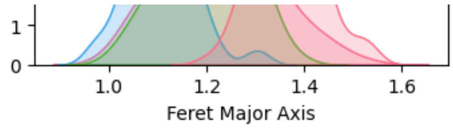


Density plot of Hollowness (Training Data - Fold 2)

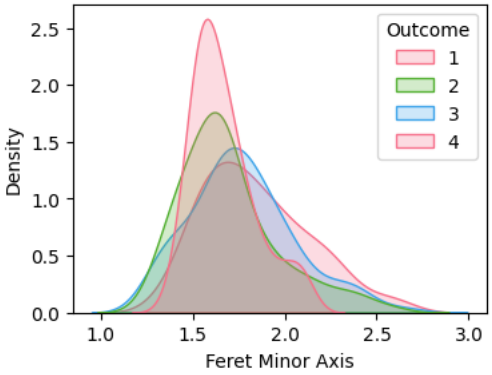


Density plot of Feret Major Axis (Training Data - Fold 2)

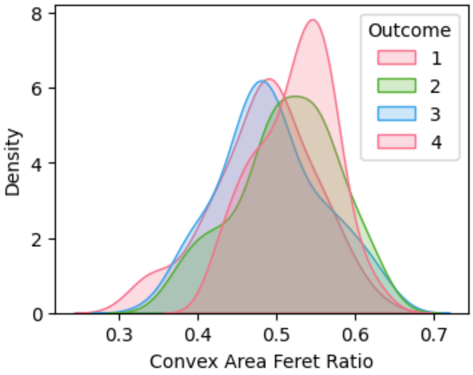




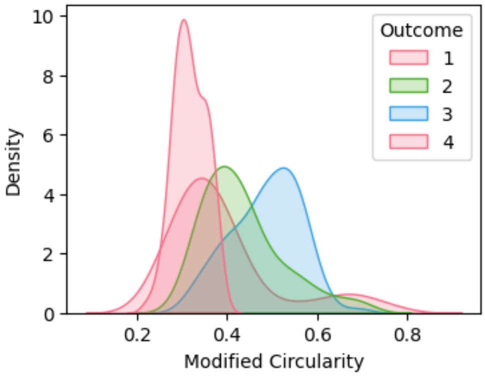
Density plot of Feret Minor Axis (Training Data - Fold 2)



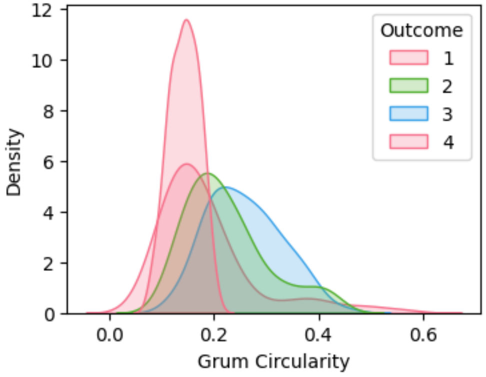
Density plot of Convex Area Feret Ratio (Training Data - Fold 2)



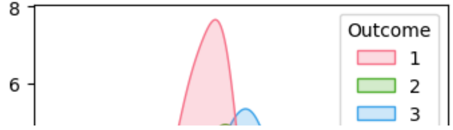
Density plot of Modified Circularity (Training Data - Fold 2)

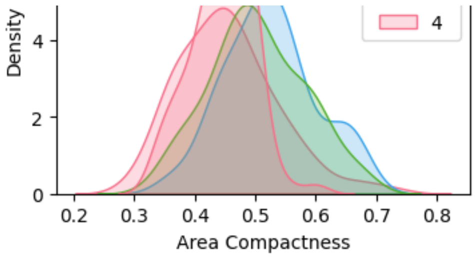


Density plot of Grum Circularity (Training Data - Fold 2)



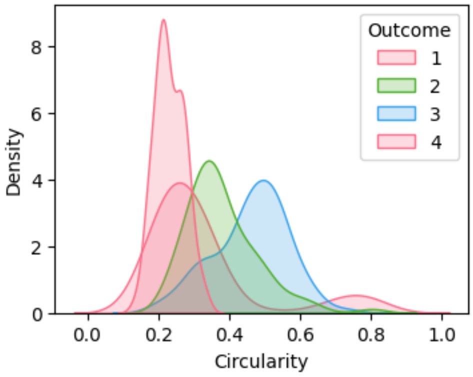
Density plot of Area Compactness (Training Data - Fold 2)



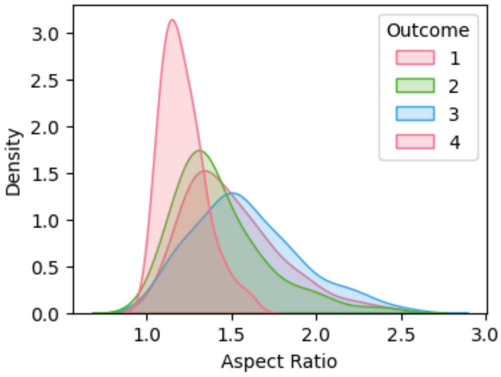


```
Spread factor for fold 2
[[0.22846341463414632, 1.2224146341463415, 0.8266341463414633, 0.6679999999999999, 0.8266560975609757, 0.29083292682926826, 0.649252
Thresholds for fold 2
{1: [0.3426951219512195, 1.8336219512195124, 1.239951219512195, 1.0019999999999998, 1.2399841463414636, 0.4362493902439024, 0.973878
Predictions
3      2
5      1
7      2
16     2
18     2
      ..
395    2
396    2
397    1
406    1
422    4
Length: 85, dtype: int64
```

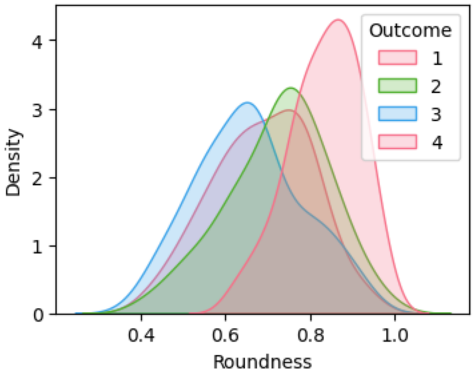
Density plot of Circularity (Training Data - Fold 3)



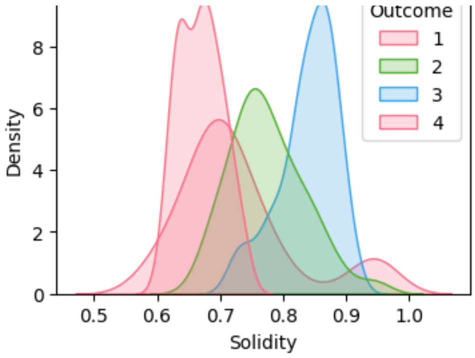
Density plot of Aspect Ratio (Training Data - Fold 3)



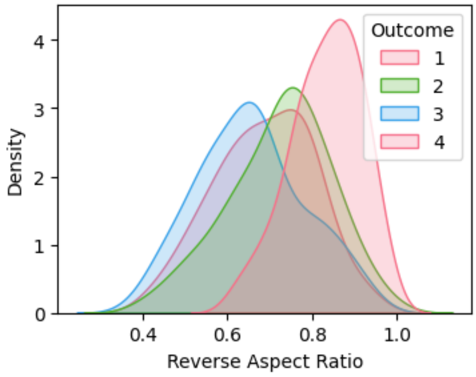
Density plot of Roundness (Training Data - Fold 3)



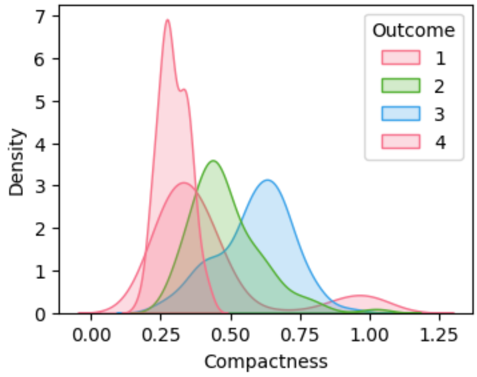
Density plot of Solidity (Training Data - Fold 3)



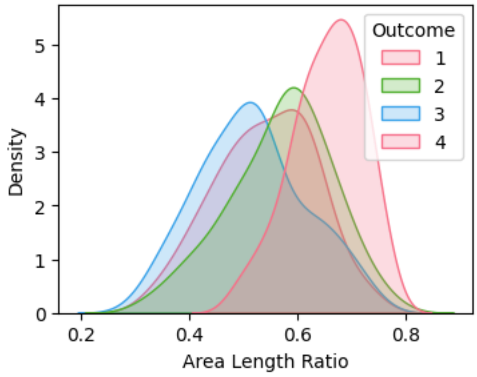
Density plot of Reverse Aspect Ratio (Training Data - Fold 3)



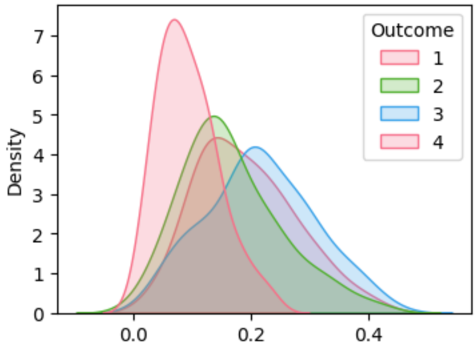
Density plot of Compactness (Training Data - Fold 3)

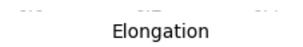


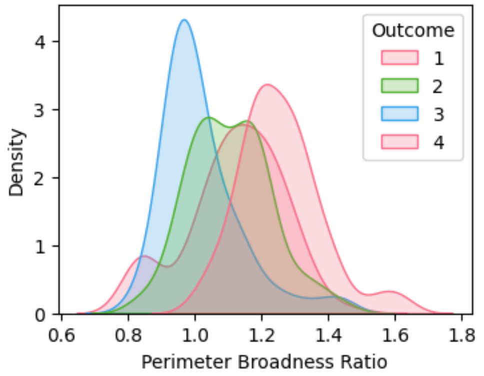
Density plot of Area Length Ratio (Training Data - Fold 3)

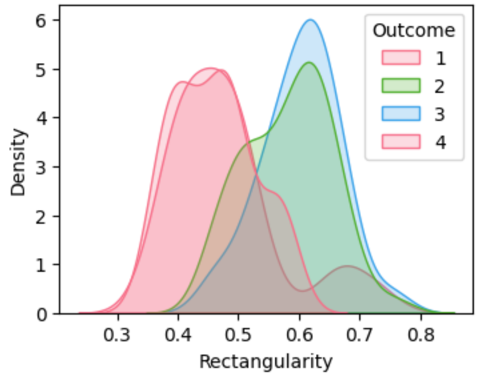


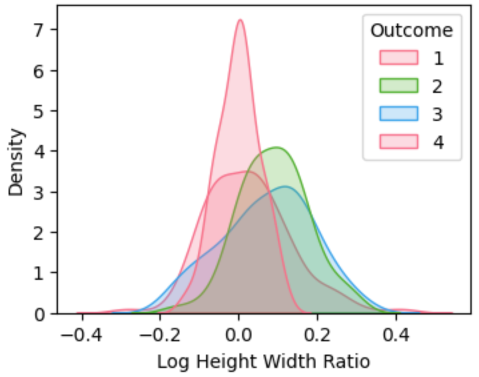
Density plot of Elongation (Training Data - Fold 3)

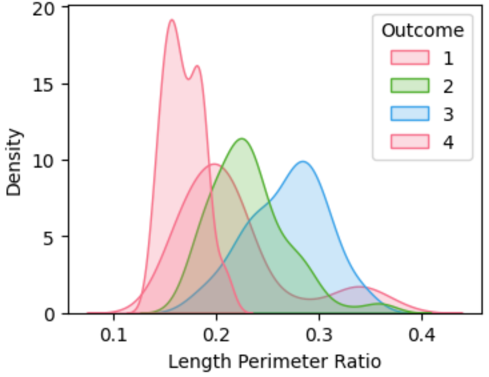


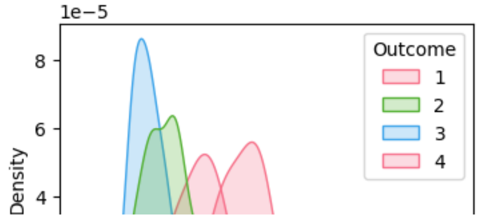


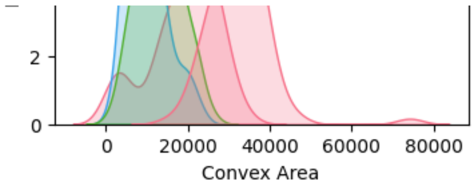

Density plot of Perimeter Broadness Ratio (Training Data - Fold 3)


Density plot of Rectangularity (Training Data - Fold 3)

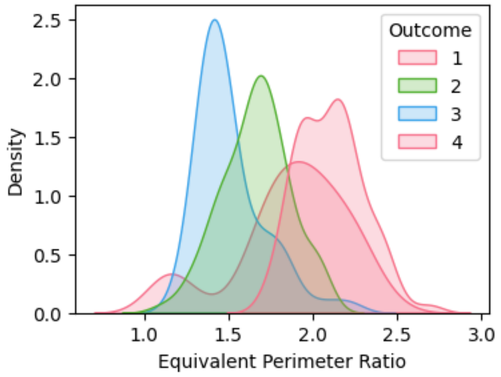

Density plot of Log Height Width Ratio (Training Data - Fold 3)


Density plot of Length Perimeter Ratio (Training Data - Fold 3)

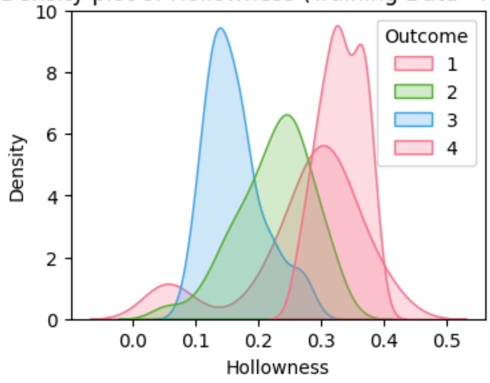

Density plot of Convex Area (Training Data - Fold 3)



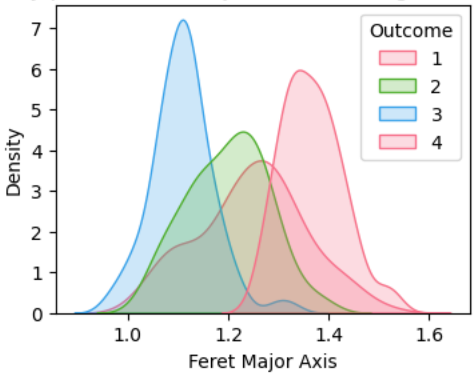
Density plot of Equivalent Perimeter Ratio (Training Data - Fold 3)



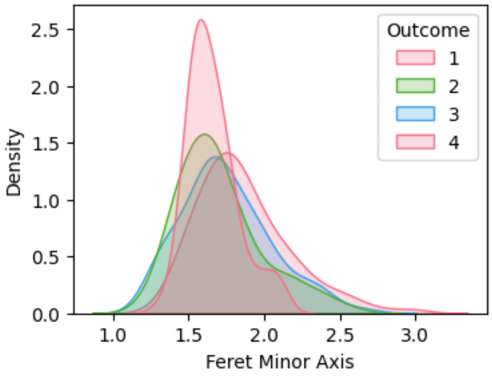
Density plot of Hollowness (Training Data - Fold 3)



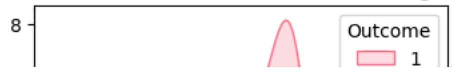
Density plot of Feret Major Axis (Training Data - Fold 3)

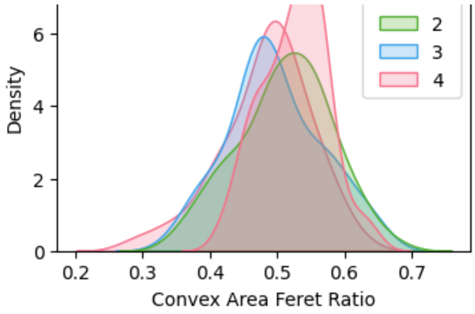


Density plot of Feret Minor Axis (Training Data - Fold 3)

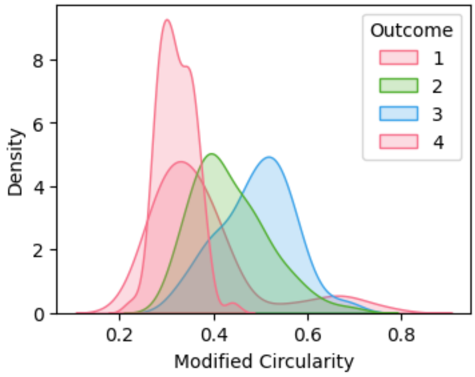


Density plot of Convex Area Feret Ratio (Training Data - Fold 3)

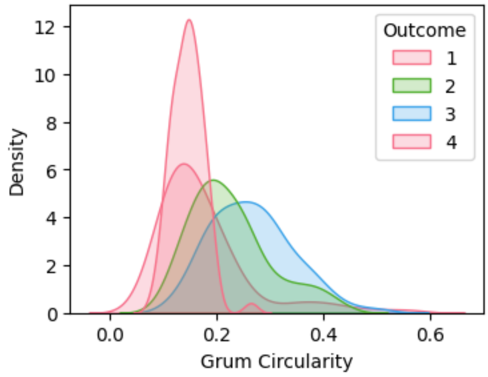




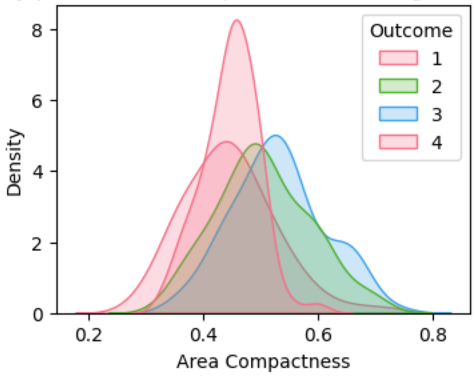
Density plot of Modified Circularity (Training Data - Fold 3)



Density plot of Grum Circularity (Training Data - Fold 3)

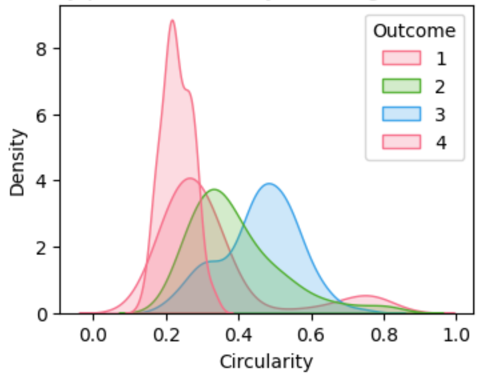


Density plot of Area Compactness (Training Data - Fold 3)

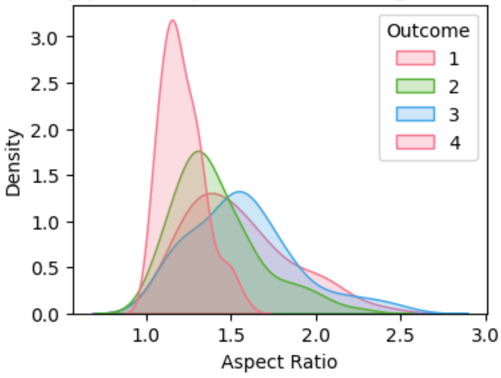


```
Spread factor for fold 3
[[0.23190789473684215, 1.2138026315789474, 0.8326184210526316, 0.6686578947368421, 0.8326092105263158, 0.29524736842105265, 0.653928
Thresholds for fold 3
{1: [0.3478618421052632, 1.820703947368421, 1.2489276315789475, 1.0029868421052632, 1.2489138157894737, 0.442871052631579, 0.9808934
Predictions
2      2
6      2
10     2
11     2
23     2
      ..
401    3
404    2
405    2
408    2
418    4
Length: 85, dtype: int64
```

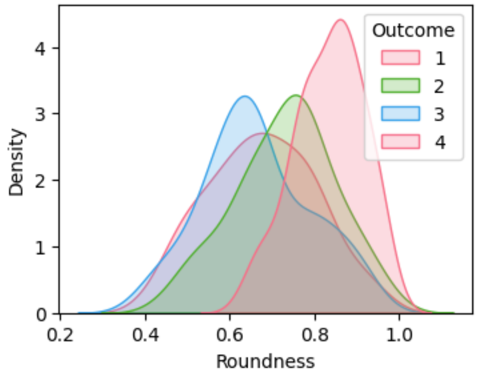
Density plot of Circularity (Training Data - Fold 4)



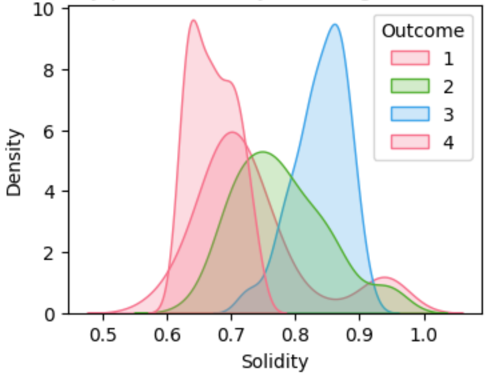
Density plot of Aspect Ratio (Training Data - Fold 4)



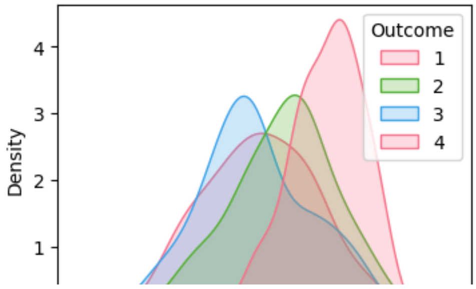
Density plot of Roundness (Training Data - Fold 4)

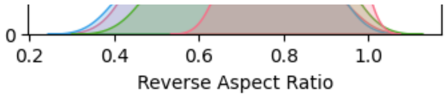


Density plot of Solidity (Training Data - Fold 4)

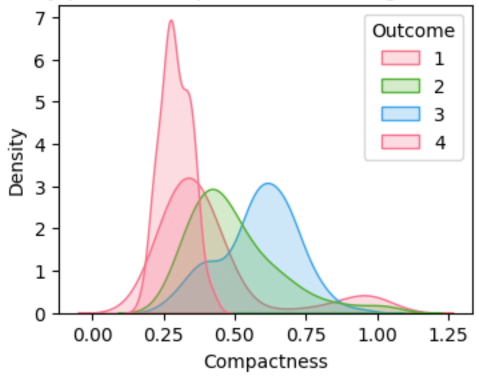


Density plot of Reverse Aspect Ratio (Training Data - Fold 4)

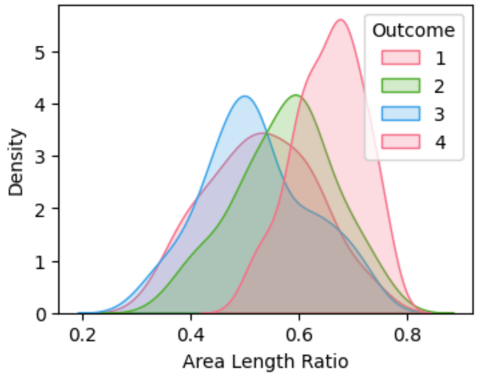




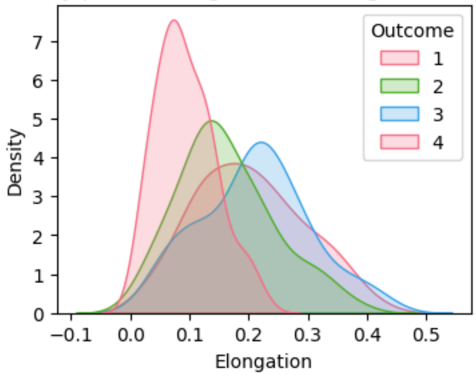
Density plot of Compactness (Training Data - Fold 4)



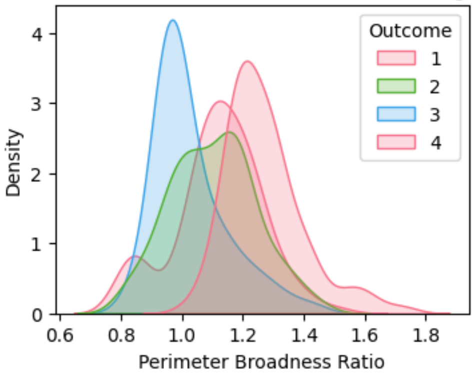
Density plot of Area Length Ratio (Training Data - Fold 4)



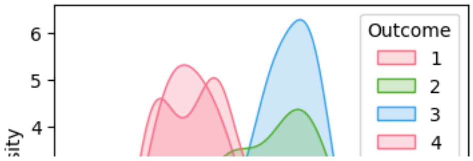
Density plot of Elongation (Training Data - Fold 4)

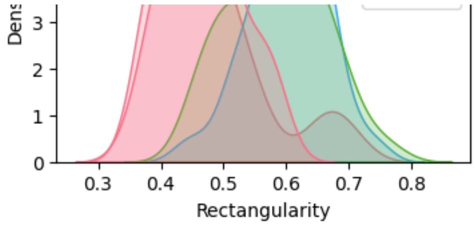


Density plot of Perimeter Broadness Ratio (Training Data - Fold 4)

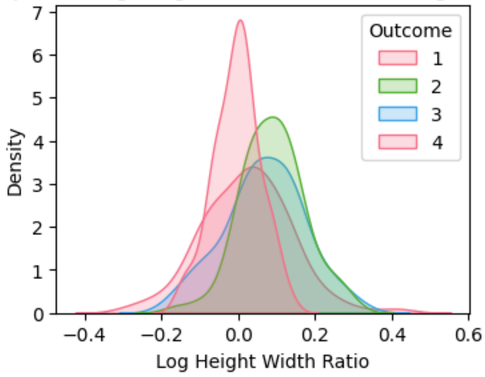


Density plot of Rectangularity (Training Data - Fold 4)

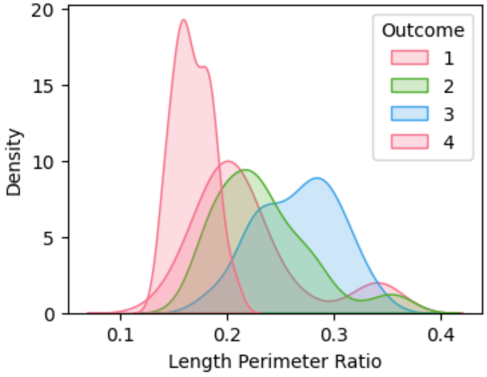




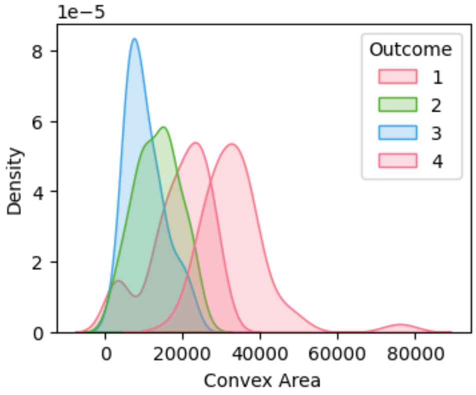
Density plot of Log Height Width Ratio (Training Data - Fold 4)



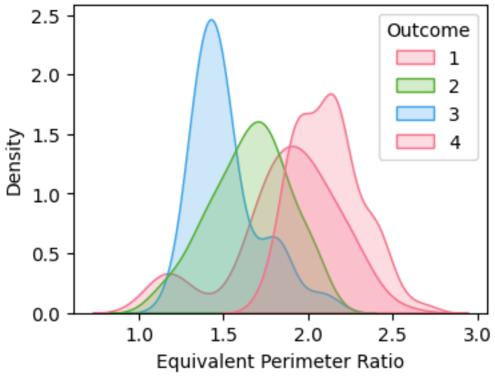
Density plot of Length Perimeter Ratio (Training Data - Fold 4)



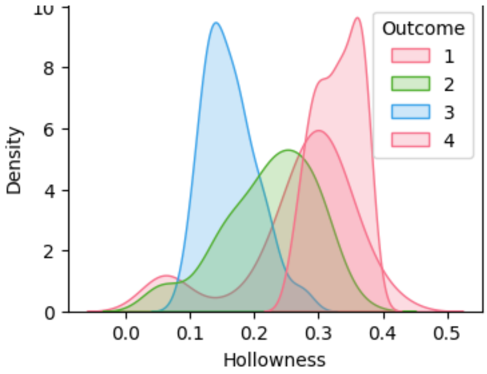
Density plot of Convex Area (Training Data - Fold 4)



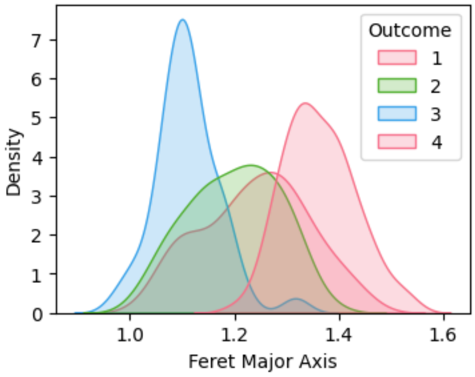
Density plot of Equivalent Perimeter Ratio (Training Data - Fold 4)



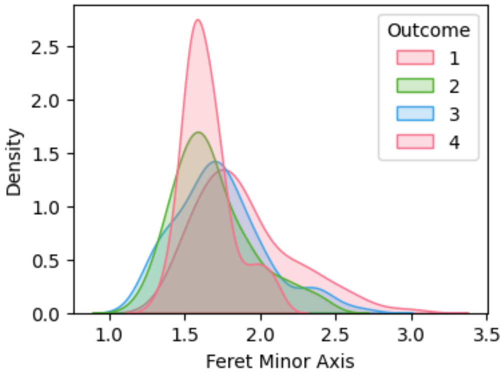
Density plot of Hollowness (Training Data - Fold 4)



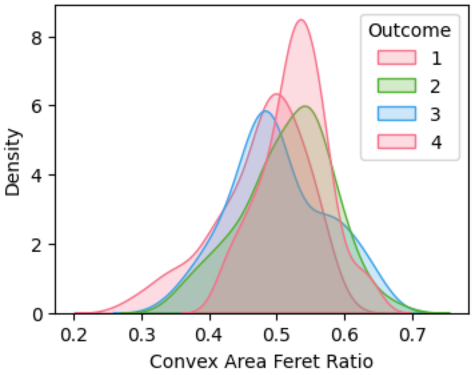
Density plot of Feret Major Axis (Training Data - Fold 4)



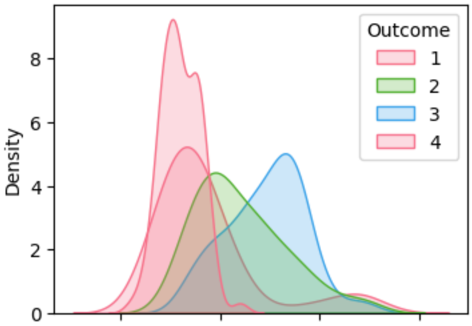
Density plot of Feret Minor Axis (Training Data - Fold 4)

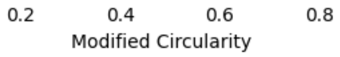


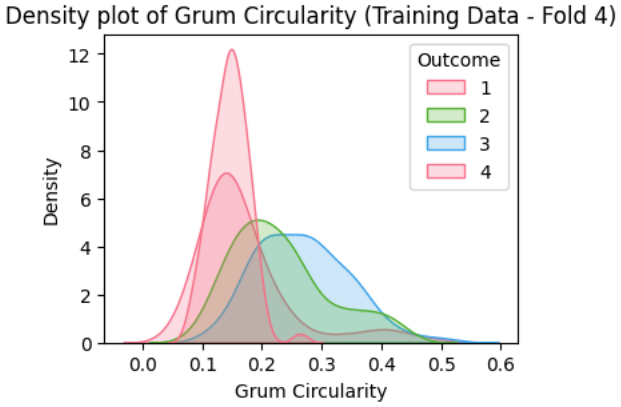
Density plot of Convex Area Feret Ratio (Training Data - Fold 4)

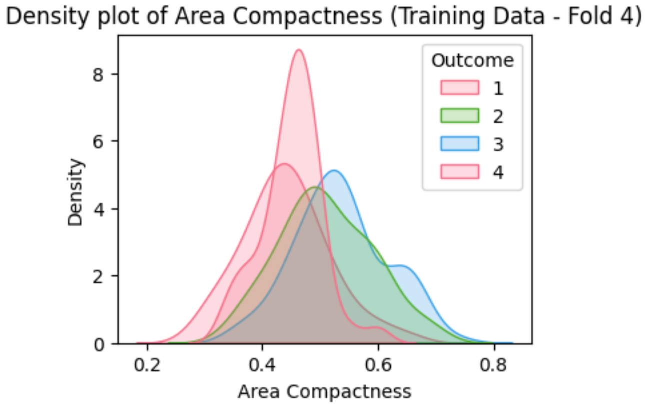


Density plot of Modified Circularity (Training Data - Fold 4)

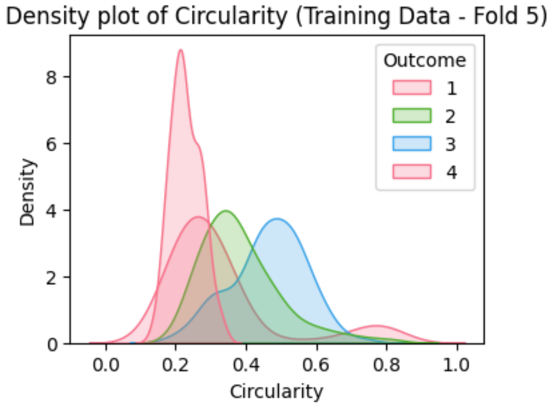


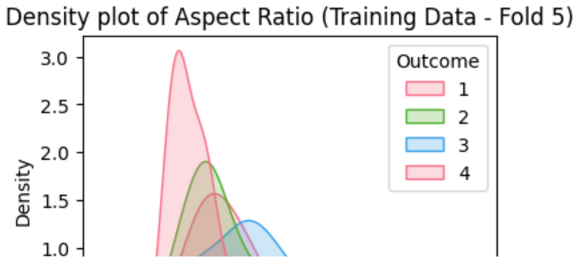


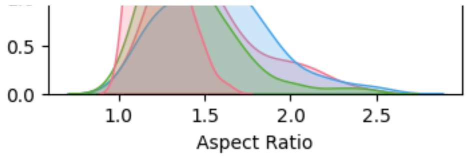

Density plot of Grum Circularity (Training Data - Fold 4)


Density plot of Area Compactness (Training Data - Fold 4)

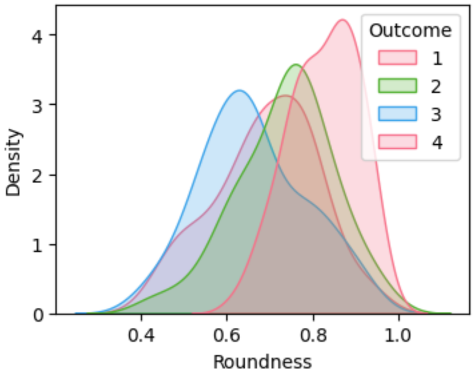
```
Spread factor for fold 4
[[0.2295421686746988, 1.2127108433734939, 0.8328674698795182, 0.6715180722891567, 0.8328506024096385, 0.2921915662650602, 0.65411927
Thresholds for fold 4
{1: [0.3443132530120482, 1.8190662650602407, 1.2493012048192773, 1.007277108433735, 1.2492759036144578, 0.4382873493975903, 0.981178
Predictions
4      3
8      2
12     2
14     1
27     2
      ..
390    2
393    1
403    4
411    2
413    2
Length: 85, dtype: int64
```


Density plot of Circularity (Training Data - Fold 5)

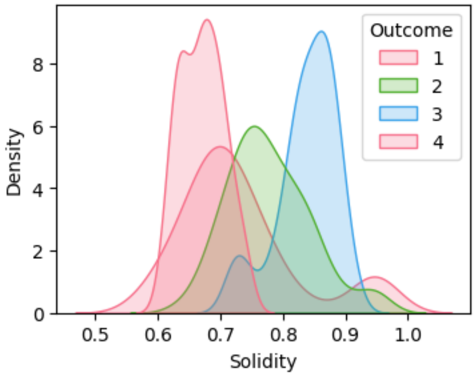

Density plot of Aspect Ratio (Training Data - Fold 5)



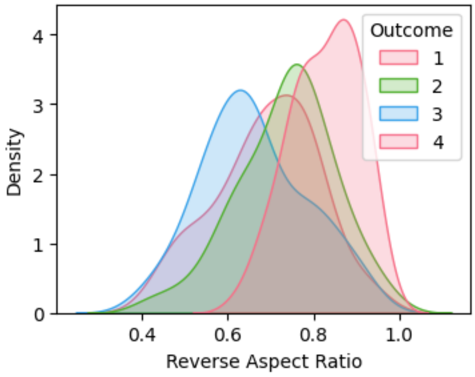
Density plot of Roundness (Training Data - Fold 5)



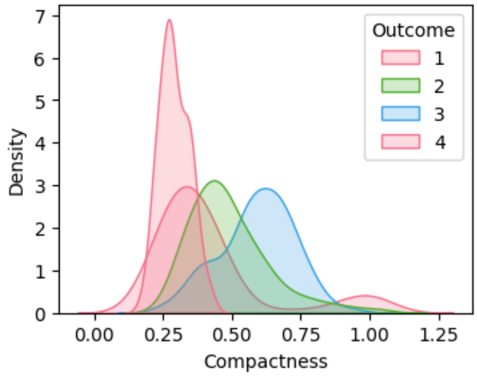
Density plot of Solidity (Training Data - Fold 5)



Density plot of Reverse Aspect Ratio (Training Data - Fold 5)

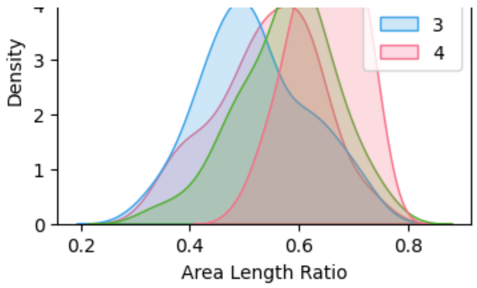


Density plot of Compactness (Training Data - Fold 5)

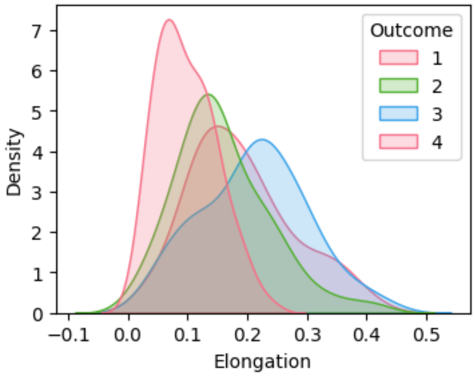


Density plot of Area Length Ratio (Training Data - Fold 5)

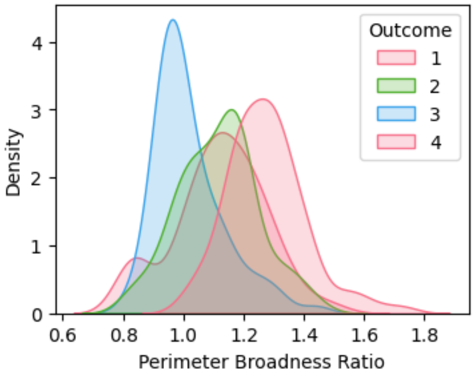
Outcome
1
2
5
4



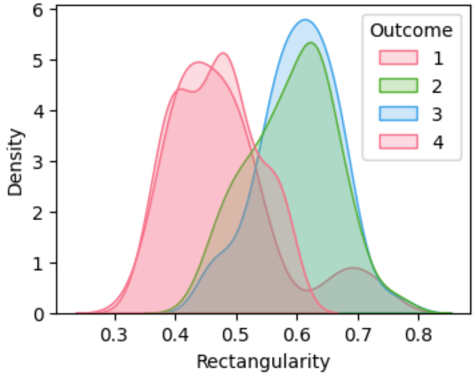
Density plot of Elongation (Training Data - Fold 5)



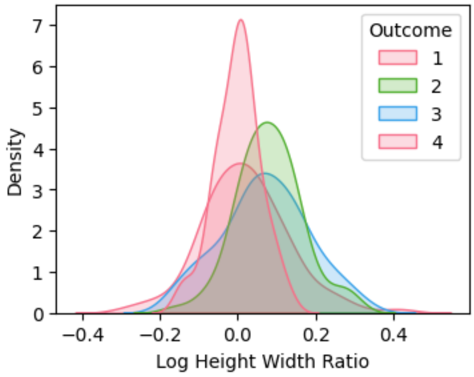
Density plot of Perimeter Broadness Ratio (Training Data - Fold 5)

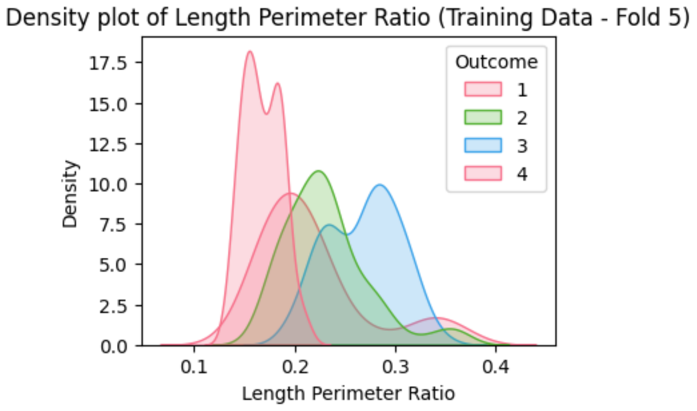


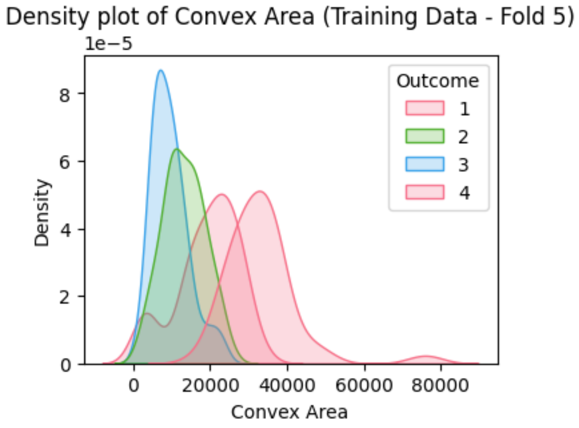
Density plot of Rectangularity (Training Data - Fold 5)

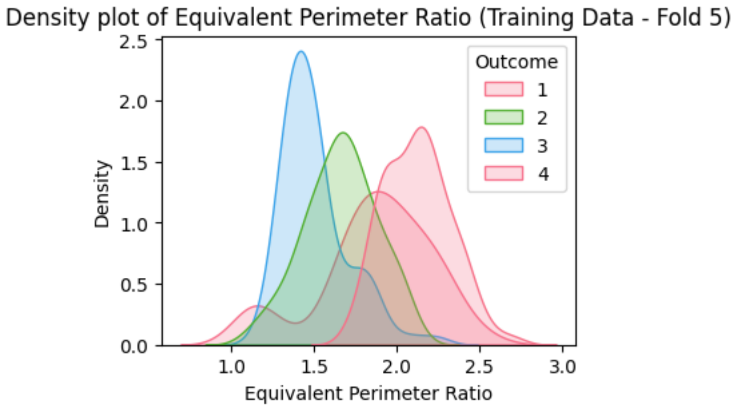


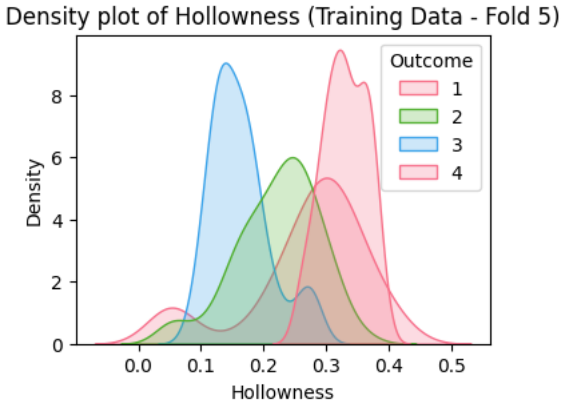
Density plot of Log Height Width Ratio (Training Data - Fold 5)

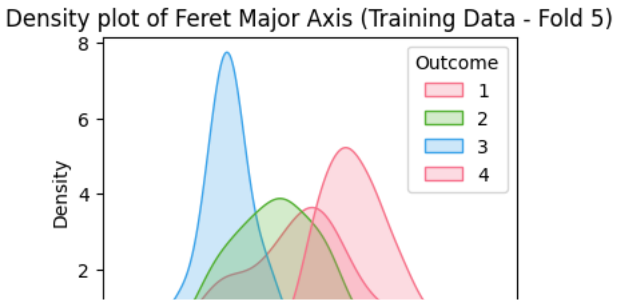


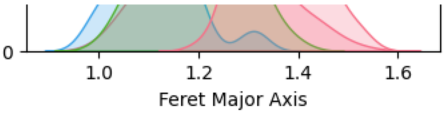

Density plot of Length Perimeter Ratio (Training Data - Fold 5)
Density
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.1
0.2
0.3
0.4
Length Perimeter Ratio
Outcome
1
2
3
4


Density plot of Convex Area (Training Data - Fold 5)
1e−5
Density
8
6
4
2
0
0
20000
40000
60000
80000
Convex Area
Outcome
1
2
3
4

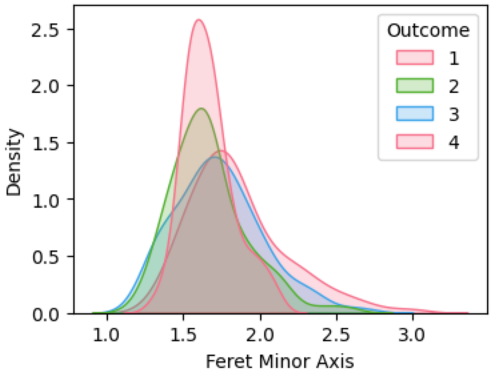

Density plot of Equivalent Perimeter Ratio (Training Data - Fold 5)
Density
2.5
2.0
1.5
1.0
0.5
0.0
1.0
1.5
2.0
2.5
3.0
Equivalent Perimeter Ratio
Outcome
1
2
3
4


Density plot of Hollowness (Training Data - Fold 5)
Density
8
6
4
2
0
0.0
0.1
0.2
0.3
0.4
0.5
Hollowness
Outcome
1
2
3
4

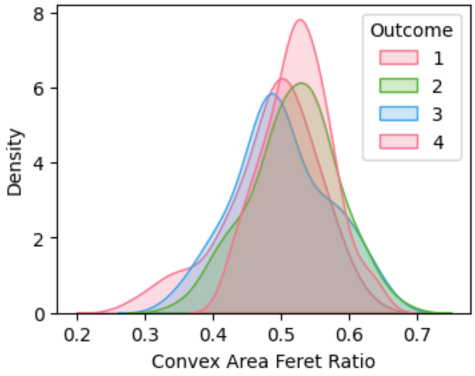

Density plot of Feret Major Axis (Training Data - Fold 5)
Density
8
6
4
2
Outcome
1
2
3
4



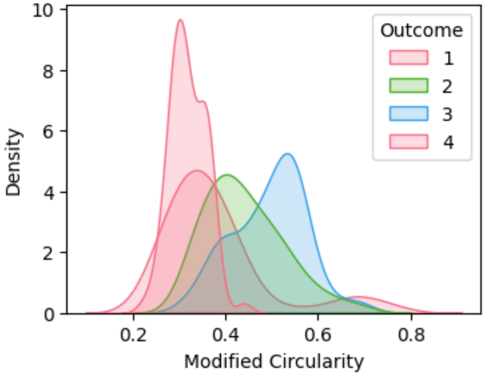
Density plot of Feret Minor Axis (Training Data - Fold 5)



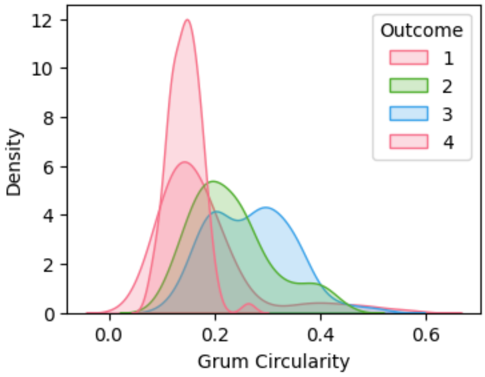
Density plot of Convex Area Feret Ratio (Training Data - Fold 5)



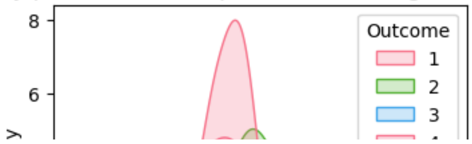
Density plot of Modified Circularity (Training Data - Fold 5)

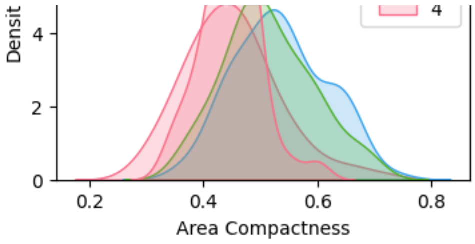


Density plot of Grum Circularity (Training Data - Fold 5)



Density plot of Area Compactness (Training Data - Fold 5)





```
Spread factor for fold 5
[[0.2281923076923077, 1.2242564102564104, 0.8250384615384615, 0.6705000000000002, 0.8250346153846152, 0.29052051282051283, 0.6479807
Thresholds for fold 5
{1: [0.34228846153846154, 1.8363846153846155, 1.2375576923076923, 1.0057500000000004, 1.2375519230769227, 0.43578076923076925, 0.971
Predictions
1      1
13     2
20     2
21     3
34     1
      ..
409    2
414    2
417    1
421    1
423    1
Length: 84, dtype: int64
```

```
# Print the average accuracy across all folds
print(f"Overall Accuracy: {acc*100:.3f} %")
```

```
Overall Accuracy: 60.714 %
```

```
pred_ind=accuracies.index(acc)
final_prediction=predi[pred_ind]

# Initialize dictionaries to store class-wise counts
correct_counts = {cls: 0 for cls in set(test_labels)}
total_counts = {cls: 0 for cls in set(test_labels)}

# Calculate overall accuracy and class-wise counts
for pred, label in zip(final_prediction, test_labels):
    if pred == label:
        correct_counts[label] += 1
    total_counts[label] += 1

# Calculate class-wise accuracies
class_accuracies = {cls: correct_counts[cls] / total_counts[cls] if total_counts[cls] != 0 else 0 for cls in correct_counts}

print("Class-wise Accuracies:")
for cls, accuracy in class_accuracies.items():
    print(f"{cls}: {accuracy*100:.3f} %")

print("Accuracy for 1 and 2: ",(correct_counts[1]+correct_counts[2])/(total_counts[1]+total_counts[2])*100," %")
print("Accuracy for 3 and 4: ",(correct_counts[3]+correct_counts[4])/(total_counts[3]+total_counts[4])*100," %")
```

```
Class-wise Accuracies:
1: 90.909 %
2: 72.222 %
3: 42.308 %
4: 38.889 %
Accuracy for 1 and 2:  82.5  %
Accuracy for 3 and 4:  40.909090909090914  %
```

```
#Seperate dataset for common lambsquarters and common purslane

# Step 1: Load the dataset
data = pd.read_excel("direct1and2 (2).xlsx")

# Step 2: Set up k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Step 3: Perform k-fold cross-validation
accuracies = []

for fold, (train_index, test_index) in enumerate(kf.split(data), 1):
    train_data, test_data = data.iloc[train_index].copy(), data.iloc[test_index].copy()

    # Generate density plots for training data
    features = train_data.drop(columns=['Outcome'])

    # Calculate spread factor for training data
    spread_factors = train_data.groupby('Outcome').apply(lambda x: x.mean().tolist())

    # Find correct threshold for training data
    threshold_multiplier = 1.5
    thresholds = {cls: [spread_factors.loc[cls][i] * threshold_multiplier for i in range(len(spread_factors.iloc[0]))] for cls in spread

    def find_class(row, thresholds):
        max_distance = float('-inf')
        predicted_class = 'Unknown'

        for cls, class_thresholds in thresholds.items():
            distance = sum(1 for i, value in enumerate(row) if value < class_thresholds[i])
            if distance > max_distance:
                max_distance = distance
                predicted_class = cls
        return predicted_class

    predictions = test_data.drop(columns='Outcome').apply(lambda row: find_class(row, thresholds), axis=1)

    # Evaluate the predictions
    test_labels = test_data['Outcome']
    accuracy = sum(1 for pred, label in zip(predictions, test_labels) if pred == label) / len(test_labels)
    accuracies.append(accuracy)
    acc=max(accuracies)

# Print the average accuracy across all folds
print(f"Accuracy: {acc*100:.3f} %")
```

```
Accuracy: 97.561 %
```

```
#Seperate dataset for horseweed and redroot pigweed

# Step 1: Load the dataset
data = pd.read_excel("newww_weed1 (4).xlsx")

# Step 2: Set up k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Step 3: Perform k-fold cross-validation
accuracies = []

for fold, (train_index, test_index) in enumerate(kf.split(data), 1):
    train_data, test_data = data.iloc[train_index].copy(), data.iloc[test_index].copy()

    # Generate density plots for training data
    features = train_data.drop(columns=['Outcome'])

    # Calculate spread factor for training data
    spread_factors = train_data.groupby('Outcome').apply(lambda x: x.mean().tolist())

    # Find correct threshold for training data
    threshold_multiplier = 1.5
    thresholds = {cls: [spread_factors.loc[cls][i] * threshold_multiplier for i in range(len(spread_factors.iloc[0]))] for cls in sprea

    def find_class(row, thresholds):
        max_distance = float('-inf')
        predicted_class = 'Unknown'

        for cls, class_thresholds in thresholds.items():
            distance = sum(1 for i, value in enumerate(row) if value < class_thresholds[i])
            if distance > max_distance:
                max_distance = distance
                predicted_class = cls
        return predicted_class

    predictions = test_data.drop(columns='Outcome').apply(lambda row: find_class(row, thresholds), axis=1)

    # Evaluate the predictions
    test_labels = test_data['Outcome']
    accuracy = sum(1 for pred, label in zip(predictions, test_labels) if pred == label) / len(test_labels)
    accuracies.append(accuracy)
    acc=max(accuracies)

# Print the average accuracy across all folds
print(f"Accuracy: {acc*100:.3f} %")
```

```
Accuracy: 88.636 %
```