Name: Shreeyash S. Dongarkar

PRN: 22510025

Machine Learning Lab

Assignment 5: Train Test Split

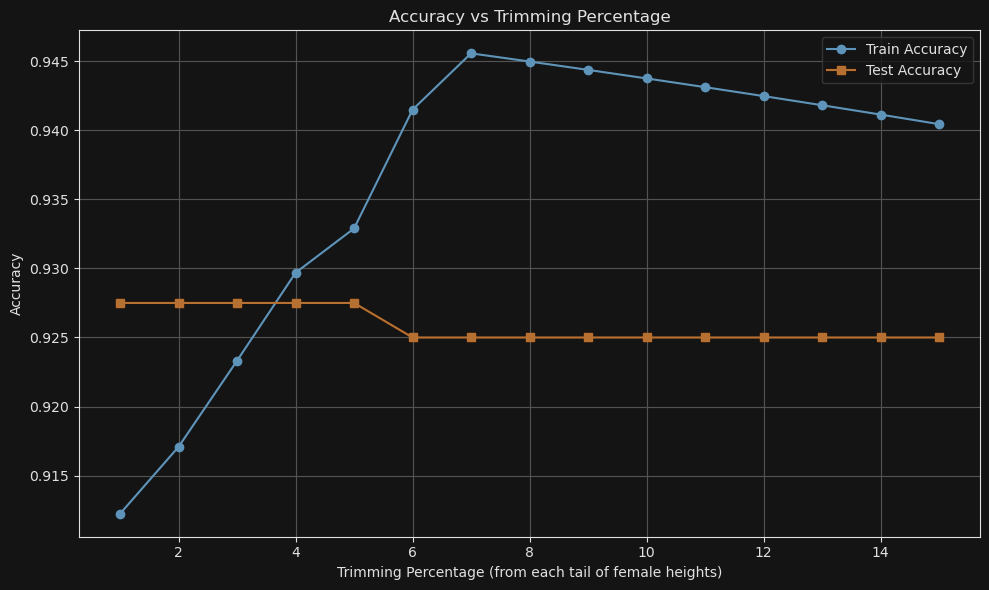
1. import numpy as np
2. import matplotlib.pyplot as plt
3. from sklearn.model\_selection import train\_test\_split
4. from scipy.stats import zscore, norm
5. mean\_male\_height = 166
6. mean\_female\_height = 152
7. np.random.seed(42)
8. male\_heights = np.random.normal(loc=mean\_male\_height, scale=5.5, size=1000)
9. female\_heights = np.random.normal(loc=mean\_female\_height, scale=4.5, size=1000)
10. male\_labels = np.zeros(1000)
11. female\_labels = np.ones(1000)
12. heights = np.concatenate((male\_heights, female\_heights))
13. labels = np.concatenate((male\_labels, female\_labels))
14. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(heights, labels, test\_size=0.2, stratify=labels, random\_state=42)
15. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(heights, labels, test\_size=0.2, stratify=labels, random\_state=42)
16. def predict\_likelihood(X, mean\_male, mean\_female, std):  
     male\_likelihood = norm.pdf(X, mean\_male, std)  
     female\_likelihood = norm.pdf(X, mean\_female, std)  
     return (female\_likelihood > male\_likelihood).astype(int)
17. mean\_male\_height = np.mean(X\_train[Y\_train == 0])
18. mean\_female\_height = np.mean(X\_train[Y\_train == 1])
19. std\_deviation = np.std(X\_train)
20. Y\_train\_pred = predict\_likelihood(X\_train, mean\_male\_height, mean\_female\_height, std\_deviation)
21. Y\_test\_pred = predict\_likelihood(X\_test, mean\_male\_height, mean\_female\_height, std\_deviation)
22. train\_accuracy = np.mean(Y\_train\_pred == Y\_train)
23. test\_accuracy = np.mean(Y\_test\_pred == Y\_test)
24. print(f'Initial Train Accuracy: {train\_accuracy:.4f}, Test Accuracy: {test\_accuracy:.4f}')
25. female\_train\_indices = np.where(Y\_train == 1)[0]
26. top\_50\_female\_indices = female\_train\_indices[np.argsort(X\_train[female\_train\_indices])[-50:]]
27. X\_train[top\_50\_female\_indices] += 10
28. mean\_male\_height = np.mean(X\_train[Y\_train == 0])
29. mean\_female\_height = np.mean(X\_train[Y\_train == 1])
30. std\_deviation = np.std(X\_train)
31. Y\_train\_pred = predict\_likelihood(X\_train, mean\_male\_height, mean\_female\_height, std\_deviation)
32. Y\_test\_pred = predict\_likelihood(X\_test, mean\_male\_height, mean\_female\_height, std\_deviation)
33. new\_train\_accuracy = np.mean(Y\_train\_pred == Y\_train)
34. new\_test\_accuracy = np.mean(Y\_test\_pred == Y\_test)
35. print(f'After Height Increase - Train Accuracy: {new\_train\_accuracy:.4f}, Test Accuracy: {new\_test\_accuracy:.4f}')
36. female\_train\_scores = zscore(X\_train[female\_train\_indices])
37. non\_outlier\_indices = female\_train\_indices[np.abs(female\_train\_scores) < 3]
38. X\_train\_filtered = np.concatenate((X\_train[non\_outlier\_indices], X\_train[Y\_train == 0]))
39. y\_train\_filtered = np.concatenate((Y\_train[non\_outlier\_indices], Y\_train[Y\_train == 0]))
40. mean\_male\_height = np.mean(X\_train\_filtered[y\_train\_filtered == 0])
41. mean\_female\_height = np.mean(X\_train\_filtered[y\_train\_filtered == 1])
42. std\_deviation = np.std(X\_train\_filtered)
43. Y\_train\_pred = predict\_likelihood(X\_train\_filtered, mean\_male\_height, mean\_female\_height, std\_deviation)
44. Y\_test\_pred = predict\_likelihood(X\_test, mean\_male\_height, mean\_female\_height, std\_deviation)
45. filtered\_train\_accuracy = np.mean(Y\_train\_pred == y\_train\_filtered)
46. filtered\_test\_accuracy = np.mean(Y\_test\_pred == Y\_test)
47. print(f'After Outlier Removal - Train Accuracy: {filtered\_train\_accuracy:.4f}, Test Accuracy: {filtered\_test\_accuracy:.4f}')
48. trim\_results = {}
49. train\_accs = []
50. test\_accs = []
51. for k in range(1, 16):  
     lower\_percentile = np.percentile(X\_train[female\_train\_indices], k)  
     upper\_percentile = np.percentile(X\_train[female\_train\_indices], 100 - k)  
     trimmed\_indices = female\_train\_indices[(X\_train[female\_train\_indices] >= lower\_percentile) & (X\_train[female\_train\_indices] <= upper\_percentile)]  
      
     X\_train\_trimmed = np.concatenate((X\_train[trimmed\_indices], X\_train[Y\_train == 0]))  
     y\_train\_trimmed = np.concatenate((Y\_train[trimmed\_indices], Y\_train[Y\_train == 0]))  
      
     mean\_male\_height = np.mean(X\_train\_trimmed[y\_train\_trimmed == 0])  
     mean\_female\_height = np.mean(X\_train\_trimmed[y\_train\_trimmed == 1])  
     std\_deviation = np.std(X\_train\_trimmed)  
      
     Y\_train\_pred = predict\_likelihood(X\_train\_trimmed, mean\_male\_height, mean\_female\_height, std\_deviation)  
     Y\_test\_pred = predict\_likelihood(X\_test, mean\_male\_height, mean\_female\_height, std\_deviation)  
      
     trimmed\_train\_accuracy = np.mean(Y\_train\_pred == y\_train\_trimmed)  
     trimmed\_test\_accuracy = np.mean(Y\_test\_pred == Y\_test)  
      
     trim\_results[k] = (trimmed\_train\_accuracy, trimmed\_test\_accuracy)  
     train\_accs.append(trimmed\_train\_accuracy)  
     test\_accs.append(trimmed\_test\_accuracy)  
      
     print(f'Trimming {k}% - Train Accuracy: {trimmed\_train\_accuracy:.4f}, Test Accuracy: {trimmed\_test\_accuracy:.4f}')

Impact of Outliers on Classification Accuracy

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Train Accuracy** | **Test Accuracy** |
| Before Introducing Outliers | 91.13% | 93.00% |
| After Introduction of Outliers | 90.69% | 92.75% |
| After Removal of Outliers | 94.58% | 92.00% |

Observations:

* When outliers were introduced, it slightly reduced the training accuracy as classifier had to generalize over a more varied dataset, but increased test accuracy due to increased variance in training data.
* Removing outliers using z-score method improved training accuracy as data became more representative of actual distribution, but test accuracy slight reduction in test accuracy due to lesser variance in the training set



Removing extreme outliers by trimming increases training accuracy gradually and test accuracy stays strong as we are removing noise in initial range of trimming. This improves generalization.

In intermediate trimming ranges we are not only removing edges but also some valid variation. Train accuracy is rising but test accuracy slightly drops. This suggests **overfitting is beginning**: the model is seeing a narrower female distribution and overconfidently classifying.

For higher amount of trimming, we are cutting a lot of amounts of data points. While **train accuracy plateaus or drops slightly**, **test accuracy stagnates** — model now performs worse on real-world (unseen) female samples. This is classic **overfitting due to bias** — the model has lost representativeness of female data.