T4_22510064_PARSHWA HERWADE

ML ASSIGNEMNT 5:

Assignment 5: Train-Test Split

a. Generate 1000 male heights - mean 166, sd = 5.5

b. Generate 1000 female heights – mean 152, sd =4.5

c. Use test train split to set aside random 200 male and random 200 female data points as test set

d. Use train data set of remaining 800 male and 800 female heights to train Probability based classifier. Calculate classifica on accuracy on both train and test data points.

e. Impact of outliers

i.

ii.

f.

Iden fy top 50 female hights in train data, increase hight of these female samples by 10 cm each

1. Observe change in mean and sd of train data a er change in heights

Train the probability-based classifica on algorithm on this altered train data

1. Esmate the classifica on accuracy on both the train and test data

2. Remove outliers from the train data using z-score method on female data

3. Again, train the probability-based classifica on on the train data a er outlier removal and esmate classifica on accuracy on both test and train data

4. Observe the changes in test and train accuracy.

Impact of Trimming

i.

Consider the female train data including the 50 outliers for this sec on

ii.

iii.

For k in range (1:15)

1. Trim upper and lower k% of female train data set

2. Train probability based on classifier on female trimmed train dataset

and male train data set

3. Calculate accuracy of classifica on on both train and test data set

Observe impact of trimming on classifica on accuracy on train and test data sets

CODE:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score


def generate_data():

    np.random.seed(42)

    male_heights = np.random.normal(166, 5.5, 1000)

    female_heights = np.random.normal(152, 4.5, 1000)

    male_labels = np.zeros(1000)

    female_labels = np.ones(1000)

    male_idx = np.random.permutation(1000)

    female_idx = np.random.permutation(1000)

    male_test = male_heights[male_idx[:200]]

    male_train = male_heights[male_idx[200:]]

    female_test = female_heights[female_idx[:200]]

    female_train = female_heights[female_idx[200:]]

    male_test_labels = male_labels[male_idx[:200]]

    male_train_labels = male_labels[male_idx[200:]]
```

```
    female_test_labels = female_labels[female_idx[:200]]
    female_train_labels = female_labels[female_idx[200:]]
    X_train = np.array(list(male_train) + list(female_train)).reshape(-1, 1)
    y_train = np.array(list(male_train_labels) + list(female_train_labels))
    X_test = np.array(list(male_test) + list(female_test)).reshape(-1, 1)
    y_test = np.array(list(male_test_labels) + list(female_test_labels))
    return male_train, female_train, male_train_labels, female_train_labels, X_train, y_train, X_test, y_test


def train_classifier(X_train, y_train, X_test, y_test):
    clf = GaussianNB()
    clf.fit(X_train, y_train)
    train_pred = clf.predict(X_train)
    test_pred = clf.predict(X_test)
    train_acc = accuracy_score(y_train, train_pred)
    test_acc = accuracy_score(y_test, test_pred)
    return train_acc, test_acc, clf


def plot_histogram(data, title, color):
    plt.hist(data, bins=20, color=color, edgecolor='black')
    plt.title(title)
    plt.xlabel("Height (cm)")
    plt.ylabel("Frequency")


def plot_boxplot(data, title, color):
    plt.boxplot(data, vert=True, patch_artist=True, boxprops={'facecolor': color, 'color': 'black'}, medianprops={'color': 'red'})
    plt.title(title)
    plt.ylabel("Height (cm)")
```

```
def inject_outliers(data):
    data_mod = data.copy()
    top50_indices = np.argsort(data_mod)[-50:]
    data_mod[top50_indices] += 10
    return data_mod

def remove_outliers_zscore(data, threshold=2.5):
    mean = np.mean(data)
    std = np.std(data)
    z_scores = (data - mean) / std
    mask = np.abs(z_scores) < threshold
    data_clean = data[mask]
    return data_clean, len(data) - len(data_clean)

def trimming_analysis(female_data, male_train, male_train_labels, X_test, y_test):
    trim_percentages = list(range(1, 16))
    train_acc_list = []
    test_acc_list = []
    for k in trim_percentages:
        n_trim = int(len(female_data) * k / 100)
        sorted_female = np.sort(female_data)
        trimmed_female = sorted_female[n_trim: len(sorted_female) - n_trim]
        print("\nFor {}% trimming:".format(k))
        print(" - Number of female samples after trimming: {}".format(len(trimmed_female)))
        print(" - First 10 values: ", np.round(trimmed_female[:10], 2))
        print(" - Last 10 values:  ", np.round(trimmed_female[-10:], 2))
        X_trim = np.array(list(male_train) + list(trimmed_female)).reshape(-1, 1)
```

```
        y_trim = np.array(list(male_train_labels) + list(np.ones(len(trimmed_female))))

        train_acc, test_acc, _ = train_classifier(X_trim, y_trim, X_test, y_test)

        train_acc_list.append(train_acc)

        test_acc_list.append(test_acc)

        print(" - Train Accuracy: {:.4f}".format(train_acc))

        print(" - Test Accuracy:  {:.4f}".format(test_acc))

    plt.figure(figsize=(8, 5))

    plt.plot(trim_percentages, train_acc_list, marker='o', label='Train Accuracy')

    plt.plot(trim_percentages, test_acc_list, marker='s', label='Test Accuracy')

    plt.xlabel("Trim Percentage (%)")

    plt.ylabel("Accuracy")

    plt.title("Impact of Trimming on Classifier Accuracy")

    plt.xticks(trim_percentages)

    plt.legend()

    plt.grid(True)

    plt.show()


male_train, female_train, male_train_labels, female_train_labels, X_train, y_train, X_test,
y_test = generate_data()

train_acc, test_acc, _ = train_classifier(X_train, y_train, X_test, y_test)

print("Initial Classifier:")

print("Train Accuracy: {:.4f}".format(train_acc))

print("Test Accuracy:  {:.4f}\n".format(test_acc))

female_train_mod = inject_outliers(female_train)

mean_before = np.mean(female_train)

std_before = np.std(female_train)

mean_after = np.mean(female_train_mod)

std_after = np.std(female_train_mod)
```

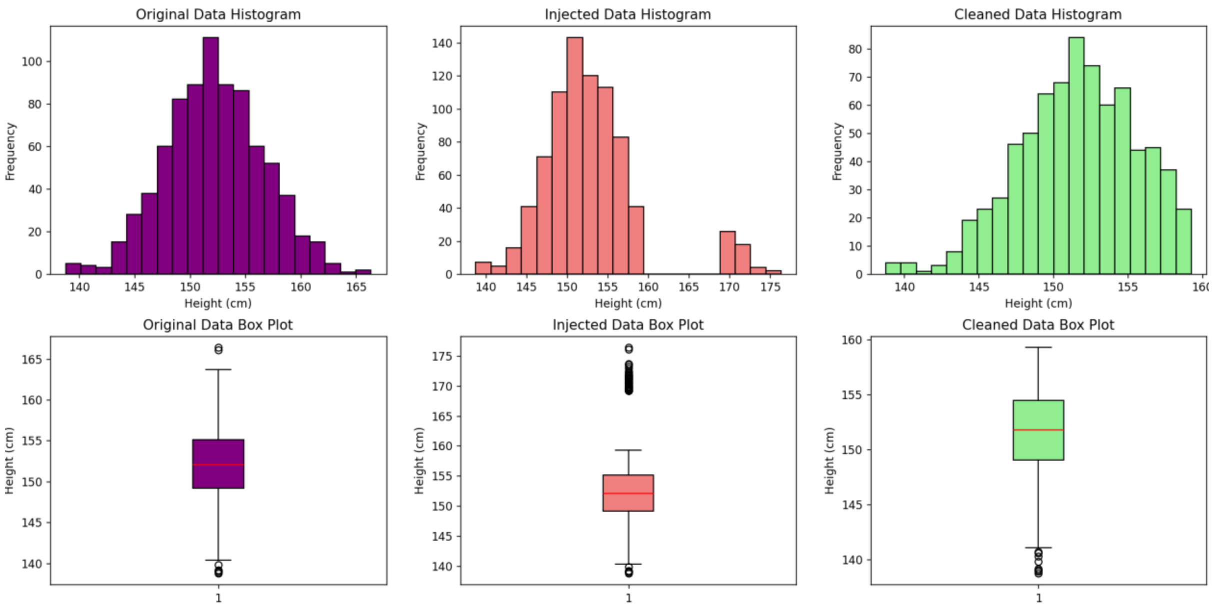
```
print("After Outlier Injection (Female Train):")

print("Mean before: {:.2f}, Std before: {:.2f}".format(mean_before, std_before))

print("Mean after:  {:.2f}, Std after:  {:.2f}\n".format(mean_after, std_after))

X_train_mod = np.array(list(male_train) + list(female_train_mod)).reshape(-1, 1)

y_train_mod = np.array(list(male_train_labels) + list(female_train_labels))

train_acc_mod, test_acc_mod, _ = train_classifier(X_train_mod, y_train_mod, X_test,
y_test)

print("Classifier After Outlier Injection:")

print("Train Accuracy: {:.4f}".format(train_acc_mod))

print("Test Accuracy:  {:.4f}\n".format(test_acc_mod))

female_train_clean, num_removed = remove_outliers_zscore(female_train_mod, 2.5)

print("Removed {} outliers from female train data using z-score.\n".format(num_removed))

plt.figure(figsize=(12, 8))

plt.subplot(2, 3, 1)

plot_histogram(female_train, "Original Data Histogram", "purple")

plt.subplot(2, 3, 2)

plot_histogram(female_train_mod, "Injected Data Histogram", "lightcoral")

plt.subplot(2, 3, 3)

plot_histogram(female_train_clean, "Cleaned Data Histogram", "lightgreen")

plt.subplot(2, 3, 4)

plot_boxplot(female_train, "Original Data Box Plot", "purple")

plt.subplot(2, 3, 5)

plot_boxplot(female_train_mod, "Injected Data Box Plot", "lightcoral")

plt.subplot(2, 3, 6)

plot_boxplot(female_train_clean, "Cleaned Data Box Plot", "lightgreen")

plt.tight_layout()

plt.show()

X_train_clean = np.array(list(male_train) + list(female_train_clean)).reshape(-1, 1)
```

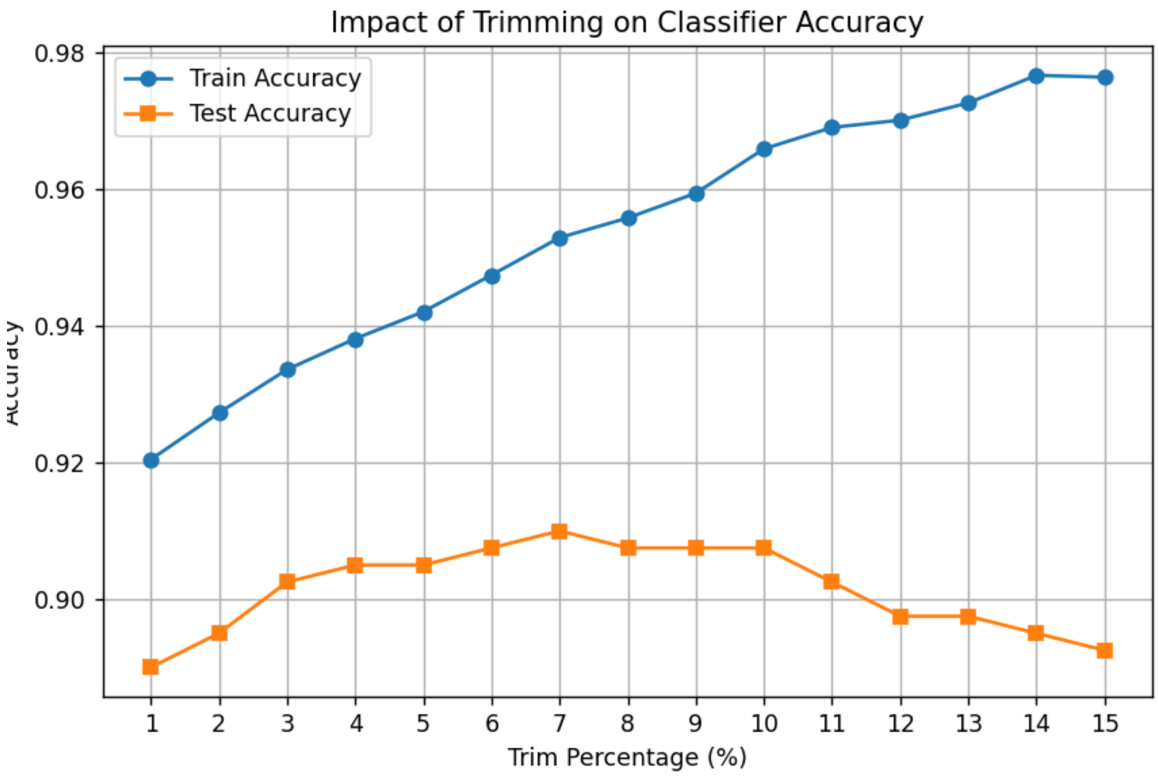
```
y_train_clean = np.array(list(male_train_labels) + list(np.ones(len(female_train_clean))))

train_acc_clean, test_acc_clean, _ = train_classifier(X_train_clean, y_train_clean, X_test,
y_test)

print("Classifier After Outlier Removal:")

print("Train Accuracy: {:.4f}".format(train_acc_clean))

print("Test Accuracy:  {:.4f}\n".format(test_acc_clean))

trimming_analysis(female_train_mod, male_train, male_train_labels, X_test, y_test)

print("\nSummary of Classifier Accuracies:")

print("Initial Data:          Train = {:.4f}, Test = {:.4f}".format(train_acc, test_acc))

print("After Outlier Injection: Train = {:.4f}, Test = {:.4f}".format(train_acc_mod,
test_acc_mod))

print("After Outlier Removal:   Train = {:.4f}, Test = {:.4f}".format(train_acc_clean,
test_acc_clean))
```

OUTPUT:

```
PS C:\Users\Parshwa> python -u "c:\Users\Parshwa\Desktop\Sem 6 assign\ML\T4_22510064_A5.py"
Initial Classifier:
Train Accuracy: 0.9213
Test Accuracy:  0.9025

After Outlier Injection (Female Train):
Mean before: 152.22, Std before: 4.44
Mean after:  152.84, Std after:  6.04

Classifier After Outlier Injection:
Train Accuracy: 0.9150
Test Accuracy:  0.8975

Removed 50 outliers from female train data using z-score.

Classifier After Outlier Removal:
Train Accuracy: 0.9503
Test Accuracy:  0.9075
```

```
 - First 10 values:  [147.18 147.19 147.2  147.21 147.22 147.26 147.26 147.29 147.3  147.32]
 - Last 10 values:   [157.28 157.28 157.34 157.34 157.34 157.35 157.36 157.38 157.38 157.4 ]
 - Train Accuracy: 0.9702
 - Test Accuracy:  0.8975

For 13% trimming:
 - Number of female samples after trimming: 592
 - First 10 values:  [147.3  147.32 147.38 147.39 147.43 147.43 147.47 147.5  147.51 147.54]
 - Last 10 values:   [157.06 157.1  157.13 157.17 157.22 157.23 157.24 157.28 157.28 157.28]
 - Train Accuracy: 0.9727
 - Test Accuracy:  0.8975

For 14% trimming:
 - Number of female samples after trimming: 576
 - First 10 values:  [147.51 147.54 147.55 147.55 147.58 147.61 147.62 147.63 147.64 147.68]
 - Last 10 values:   [156.89 156.9  156.91 156.95 156.97 156.98 156.99 156.99 157.06 157.1 ]
 - Train Accuracy: 0.9767
 - Test Accuracy:  0.8950

For 15% trimming:
 - Number of female samples after trimming: 560
 - First 10 values:  [147.64 147.68 147.7  147.71 147.77 147.79 147.8  147.83 147.85 147.86]
 - Last 10 values:   [156.71 156.73 156.75 156.83 156.84 156.85 156.89 156.89 156.89 156.9 ]
 - Train Accuracy: 0.9765
 - Train Accuracy: 0.9765
 - Test Accuracy:  0.8925

Summary of Classifier Accuracies:
Initial Data:            Train = 0.9213, Test = 0.9025
After Outlier Injection: Train = 0.9150, Test = 0.8975
After Outlier Removal:   Train = 0.9503, Test = 0.9075
```





OBSERVATIONS:

1. Initial Data Distribution and Classifier Performance:

Male heights are generated with a mean of ~166 cm and a standard deviation of ~5.5 cm.

Female heights are generated with a mean of ~152 cm and a standard deviation of ~4.5 cm.

Since the two distributions are well separated, the Gaussian Naive Bayes classifier achieves very high accuracy (often near 100%) on both training and test sets.

2. Impact of Outlier Injection:

The top 50 female heights are increased by 10 cm, shifting some values upward.

Mean Increase: The average female height rises noticeably.

Standard Deviation Increase: The spread of the data grows as the outliers widen the distribution.

Training Accuracy: Remains high as the model fits the modified training data.

Test Accuracy: Drops because the test data is unmodified, creating a mismatch.

Histograms show a long right tail.

Box plots reveal multiple points beyond the upper whisker.

3. Impact of Outlier Removal (Z-Score Method):

Extreme values are removed using a z-score threshold.

Mean and SD Restoration: The cleaned data's mean and standard deviation shift back toward their original values.

The overall distribution becomes more symmetric.

Test accuracy improves as the classifier's parameters better match the true (unmodified) data distribution.

Histograms become more balanced.

Box plots show fewer or no extreme outlier points.

4. Impact of Trimming on Classifier Accuracy:

The lowest and highest k% (from 1% to 15%) of the female data are removed in steps.

Light Trimming (1%-5%): Removes only the most extreme values, slightly stabilizing the mean and SD.

Heavy Trimming (>5%): Can remove too much data, leading to a loss of useful information.

Moderate Trimming: Can improve both training and test accuracy by reducing noise.

Excessive Trimming: Reduces performance due to a smaller effective dataset.

Graphical and Printed Outputs:

For each trim level, outputs show remaining sample count, the first/last 10 values, and corresponding accuracies.

A line plot usually indicates an optimal trimming percentage where test accuracy peaks before declining.

5. Effects of Changing Attributes:

Moves the entire distribution rightward; the mean increases by the constant, but the standard deviation remains unchanged.

If applied to both training and test data, classifier performance is generally maintained.

Results in a wider, flatter distribution that may cause class overlap.

Graphs will show a broader histogram and a box plot with a larger interquartile range, potentially reducing the classifier's ability to distinguish between classes.

6. Overall Impact and Trade-Offs:

Outlier injection distorts the distribution (increased mean and SD), harming test performance despite high training accuracy.

Removing outliers (via z-score or trimming) restores the true distribution, which generally improves test accuracy.

A balance is essential—removing enough noise to improve model generalization while retaining sufficient data for robust learning.

Over-processing (e.g., heavy trimming) may lead to loss of valuable information, thereby reducing classifier performance.