BIOM5405/SYSC5405
Assignment 4

Contents

BIOM5405/SYSC5405

Assignment 4

BIOM5405/SYSC5405

Assignment 4

List of Figures

BIOM5405/SYSC5405
Assignment 4

Part 1: Load Dataset

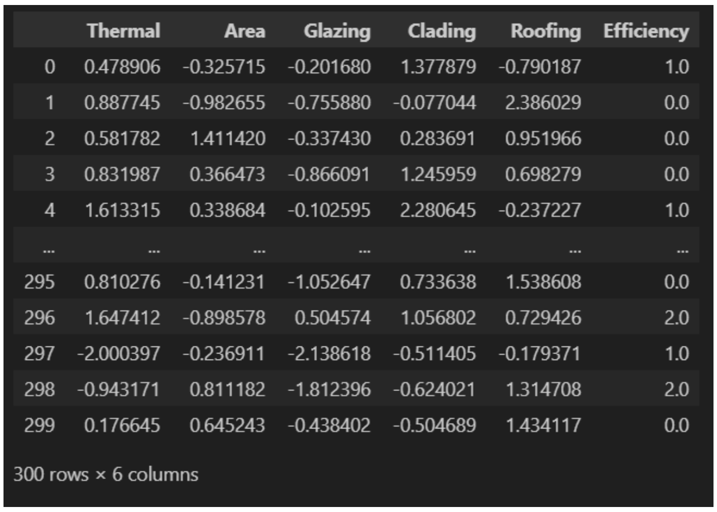
Load the dataset in A4.txt. The column names correspond to the five features plus the class ID: colNames = ['Thermal', 'Area', 'Glazing', 'Clading', 'Roofing', 'Efficiency']

Solution

Code

```
# init
COLUMNS = ['Thermal', 'Area', 'Glazing', 'Clading', 'Roofing', 'Efficiency']
DATA_PATH = "../data/A4.csv"
# read dataset
dataset = pd.read_csv(
    DATA_PATH,
    header=None,
    )
dataset.columns = COLUMNS
```

Extra 1: Dataset Snippet

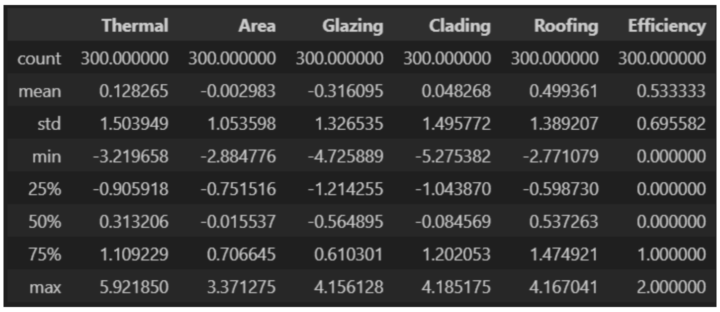


	Thermal	Area	Glazing	Clading	Roofing	Efficiency
0	0.478906	-0.325715	-0.201680	1.377879	-0.790187	1.0
1	0.887745	-0.982655	-0.755880	-0.077044	2.386029	0.0
2	0.581782	1.411420	-0.337430	0.283691	0.951966	0.0
3	0.831987	0.366473	-0.866091	1.245959	0.698279	0.0
4	1.613315	0.338684	-0.102595	2.280645	-0.237227	1.0
...	...	...	...	...	...	...
295	0.810276	-0.141231	-1.052647	0.733638	1.538608	0.0
296	1.647412	-0.898578	0.504574	1.056802	0.729426	2.0
297	-2.000397	-0.236911	-2.138618	-0.511405	-0.179371	1.0
298	-0.943171	0.811182	-1.812396	-0.624021	1.314708	2.0
299	0.176645	0.645243	-0.438402	-0.504689	1.434117	0.0

300 rows × 6 columns

Figure 1: Dataset Snippet

Extra 2: Dataset Statistics



	Thermal	Area	Glazing	Clading	Roofing	Efficiency
count	300.000000	300.000000	300.000000	300.000000	300.000000	300.000000
mean	0.128265	-0.002983	-0.316095	0.048268	0.499361	0.533333
std	1.503949	1.053598	1.326535	1.495772	1.389207	0.695582
min	-3.219658	-2.884776	-4.725889	-5.275382	-2.771079	0.000000
25%	-0.905918	-0.751516	-1.214255	-1.043870	-0.598730	0.000000
50%	0.313206	-0.015537	-0.564895	-0.084569	0.537263	0.000000
75%	1.109229	0.706645	0.610301	1.202053	1.474921	1.000000
max	5.921850	3.371275	4.156128	4.185175	4.167041	2.000000

Figure 2: Dataset Statistics

BIOM5405/SYSC5405

Assignment 4

Part 2: Stratified Sampling

Split your data into train/test using a 75/25 split and stratified sampling. Report the number of samples from each class in your train and test subsets.

Solution

Class	Train Sample Count	Test Sample Count	Complete dataset Count
Low=0	131	44	175
Med=1	68	22	90
High=2	26	9	35

Table 1: Sample count for each set of data

Table 1 contains the number of samples for each class (low, med, and high) in the train and test subset.

Extra 1: Confirmation of 75/25 split

Total train samples = 131 + 68 + 26 = 225

Train split ratio = (Total train samples)/(size of dataset) = 225/300 = 0.75

Total test samples = 44 + 22 + 9 = 75

Test split ratio = (Total test samples)/(size of dataset) = 75/300 = 0.25

Therefore, there was a proper split of 75% train set and 25% test set

Extra 2: Confirmation of stratification

To confirm this the ratio of Train samples vs the Test samples should be approximately 3.

For Low: 131/44 = 2.977

For Med: 68/22 = 3.090

For High: 26/9 = 2.888

Code:

```
# init
TRAIN_SIZE = 0.75
labels = dataset[COLUMNS[-1]]

# perform stratified split
train_dataset, test_dataset = train_test_split(
    dataset,
    train_size=TRAIN_SIZE,
    stratify=labels,
    random_state=RANDOM_STATE
)
```

BIOM5405/SYSC5405

Assignment 4

Part 3: Visualization

Using the training set, for each feature, plot the feature distribution for each class. You can either use five histograms or five 1D kernel density plots. Label each sub-plot by the feature name. The distribution of feature values should be visible for all three (potentially overlapping) classes on each of the five plots. Which feature looks most useful and why? Which home efficiency class do you think will have the lowest accuracy and why? (60 words max)

Solution:

Most useful feature: Thermal, because there is less overlap between low and high classes additionally there are a few values of high that are separate as well for larger values.

Lowest accuracy Efficiency calls: High=2, since it is overlapping with other classes in almost all the cases.

Please note that for the figures below the bin count is 30.

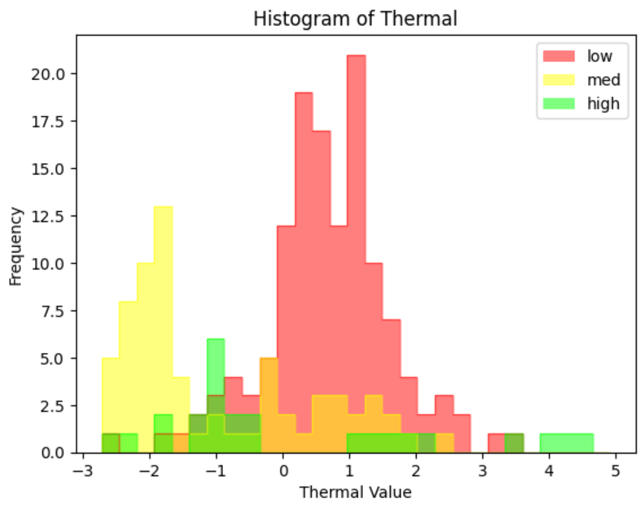


Figure 3: Histogram of distribution for different Efficiency classes for the feature Thermal

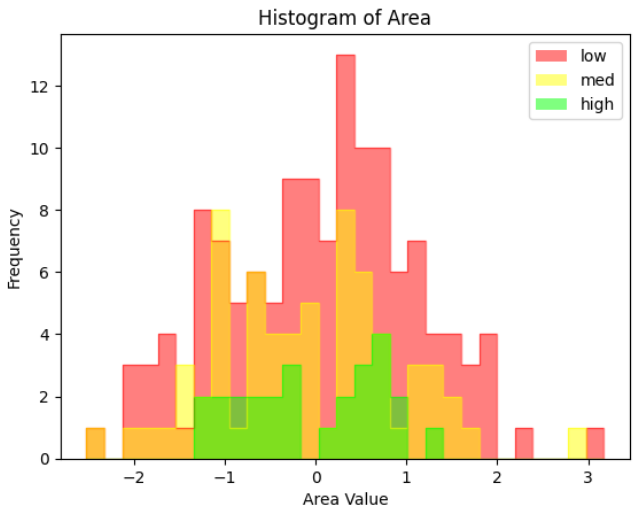


Figure 4: Histogram of distribution for different Efficiency classes for the feature Area

BIOM5405/SYSC5405

Assignment 4

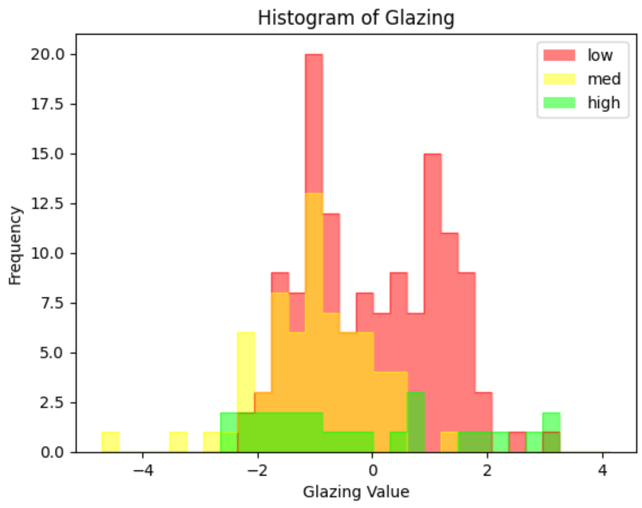


Figure 5: Histogram of distribution for different Efficiency classes for the feature Glazing

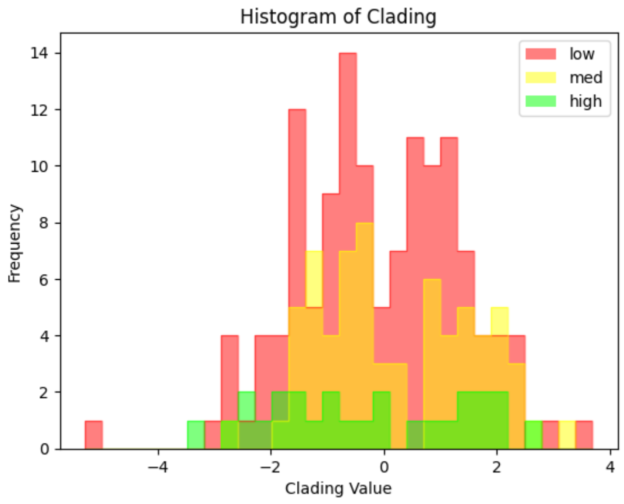


Figure 6: Histogram of distribution for different Efficiency classes for the feature Cladding

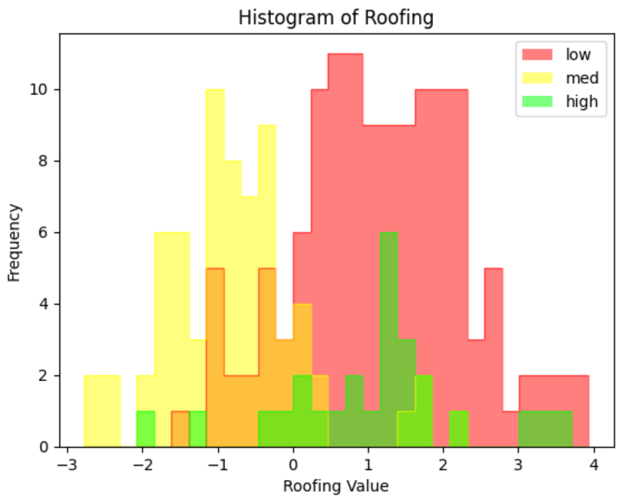


Figure 7: Histogram of distribution for different Efficiency classes for the feature Roofing

BIOM5405/SYSC5405

Assignment 4

Extra 1: Histogram for bin count = 10

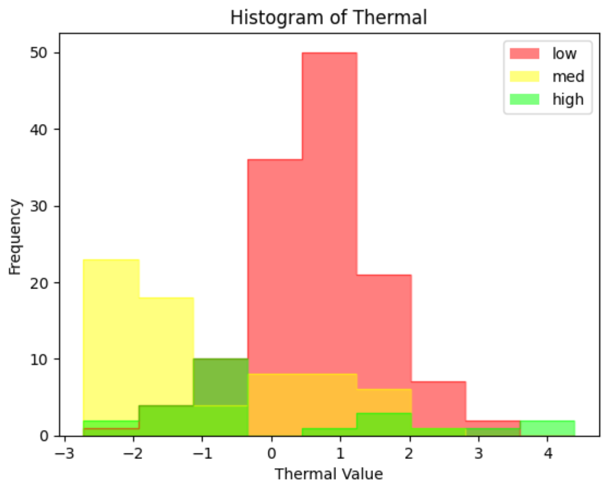


Figure 8: Histogram with bin count 10 of distribution for different Efficiency classes for the feature Thermal

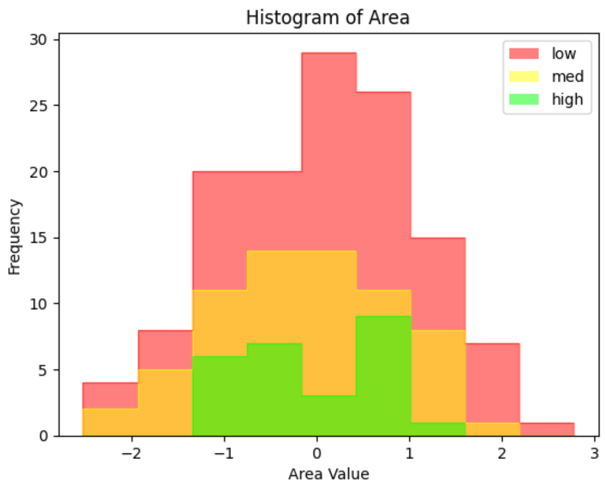


Figure 9: Histogram with bin count 10 of distribution for different Efficiency classes for the feature Area

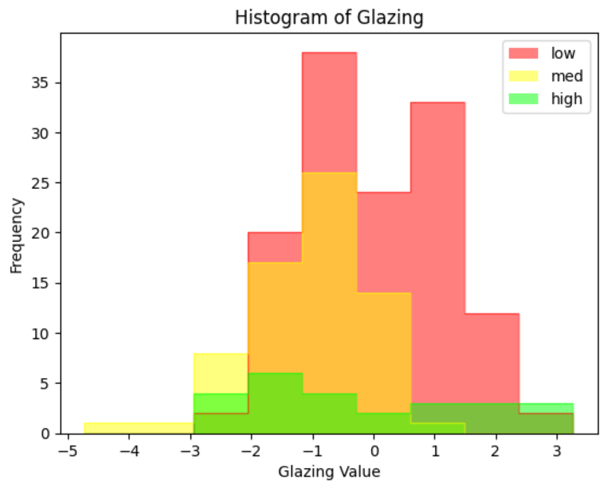


Figure 10: Histogram with bin count 10 of distribution for different Efficiency classes for the feature Glazing

BIOM5405/SYSC5405

Assignment 4

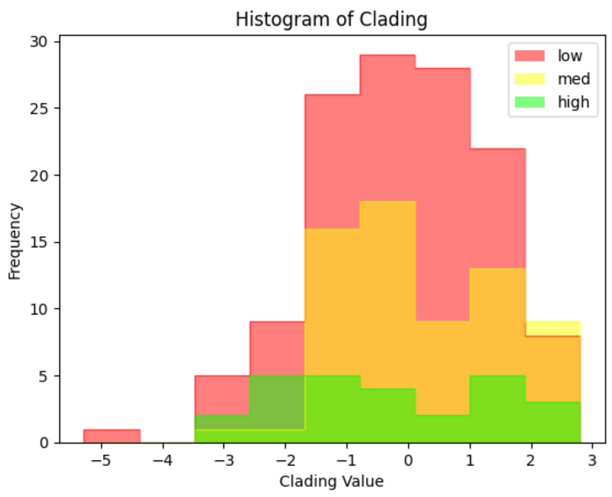


Figure 11:Histogram with bin count 10 of distribution for different Efficiency classes for the feature Cladding

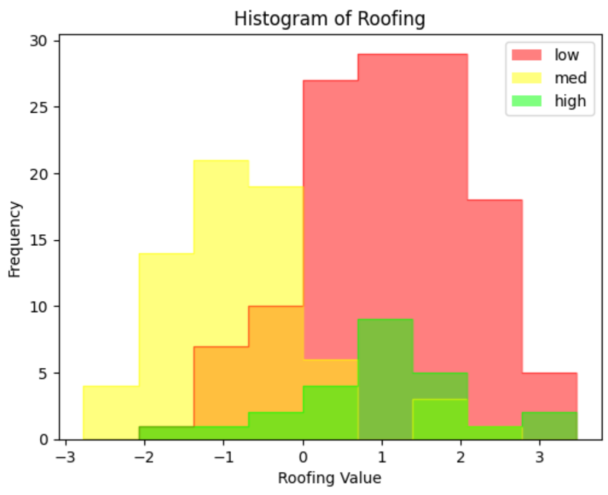


Figure 12: Histogram with bin count 10 of distribution for different Efficiency classes for the feature Roofing

Extra 2: Density Plots

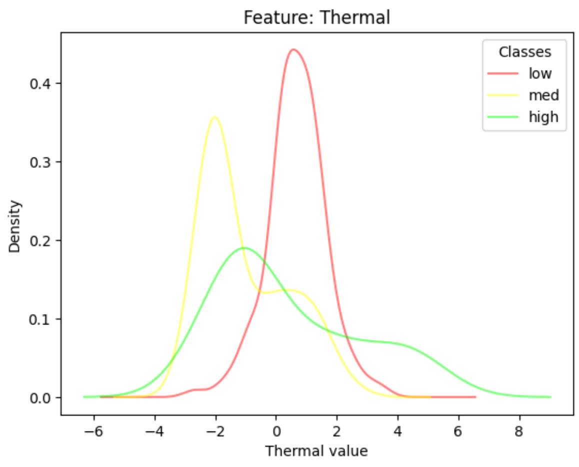


Figure 13: Density plot for different efficiency classes for the feature Thermal

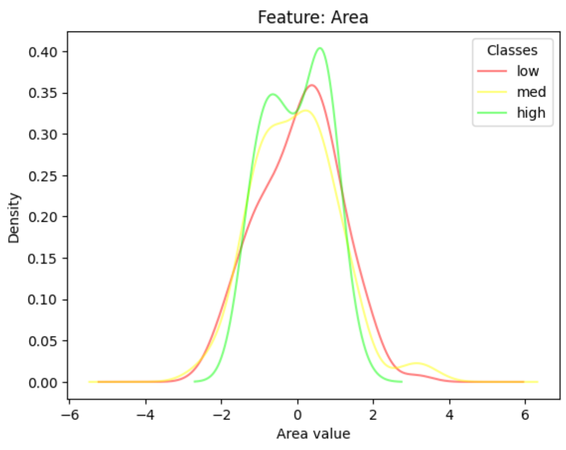


Figure 14: Density plot for different efficiency classes for the feature Area

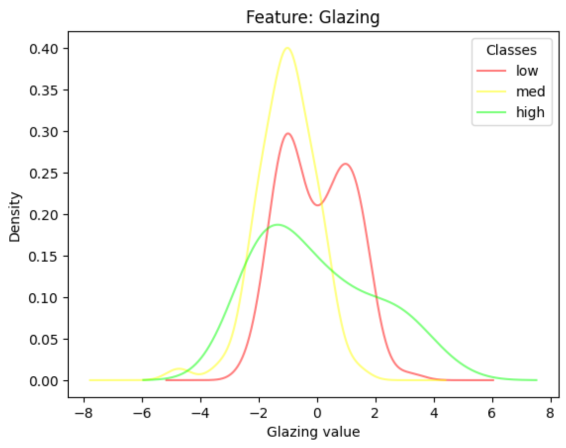


Figure 15: Density plot for different efficiency classes for the feature Glazing

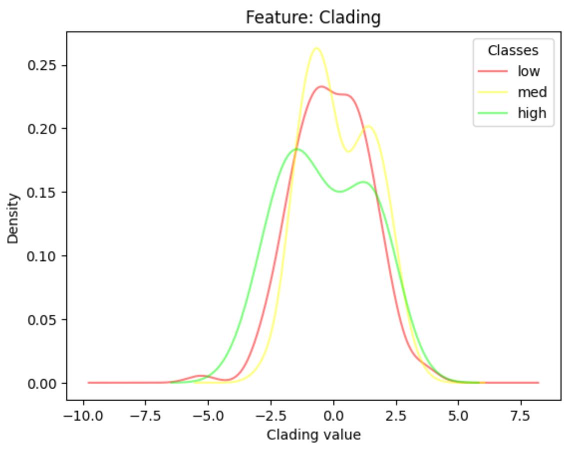


Figure 16: Density plot for different efficiency classes for the feature Cladding

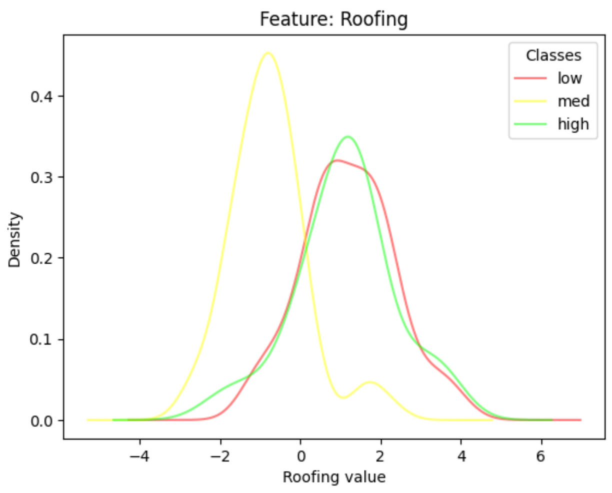


Figure 17: Density plot for different efficiency classes for the feature Roofing

Extra 3: Worst Feature

Of all the features available “Area” should be the worst performing feature.

Code:

```
# init
BINS_COUNT = 30

# build histograms for each feature
for column in COLUMNS[:-1]:
    data_df = train_dataset[[column,COLUMNS[-1]]]
    max_value = data_df[column].max()
    min_value = data_df[column].min()
    step = (max_value - min_value) / BINS_COUNT
    range_ = np.arange(min_value, max_value, step)
    for effic in range(3):
        eff_df = data_df[data_df[COLUMNS[-1]] == effic]
        eff_df[column].plot.hist(
            bins=range_,
            histtype=u'step',
            color=COLOR[effic],
        )
        ax = eff_df[column].plot.hist(
            bins=range_,
            color=COLOR[effic],
            label=LABELS[effic],
            legend=True
        )
    plt.title(f"Histogram of {column}")
    plt.xlabel(f"{column} Value")
    plt.show()
```

BIOM5405/SYSC5405

Assignment 4

Part 4: Training with Cross Validation:

Complete 5-fold-cross-validation over the train subset using an SVM classifier with a polynomial kernel with degree=3 and C=0.8. Report the accuracy over each fold, the average accuracy across all five folds, and the standard deviation across the five accuracy measurements.

Solution:

Fold 0: Accuracy = 0.75556

Fold 1: Accuracy = 0.82222

Fold 2: Accuracy = 0.80000

Fold 3: Accuracy = 0.77778

Fold 4: Accuracy = 0.82222

Average Accuracy: 0.7956

Standard Deviation: 0.0259

NOTE: **** It does not mention in the question to stratify the CV. If that is needed, please look at Extra 1.

Extra 1: Stratified Cross Validation

Same experimentation was run again but this time with stratification in CV as well. Following were the results:

Fold 0: Accuracy = 0.75556

Fold 1: Accuracy = 0.75556

Fold 2: Accuracy = 0.73333

Fold 3: Accuracy = 0.88889

Fold 4: Accuracy = 0.84444

Average Accuracy: 0.7956

Standard Deviation: 0.0603

The mean stayed relatively the same, but the standard deviation increased.

Extra 2: Visualization for without stratification

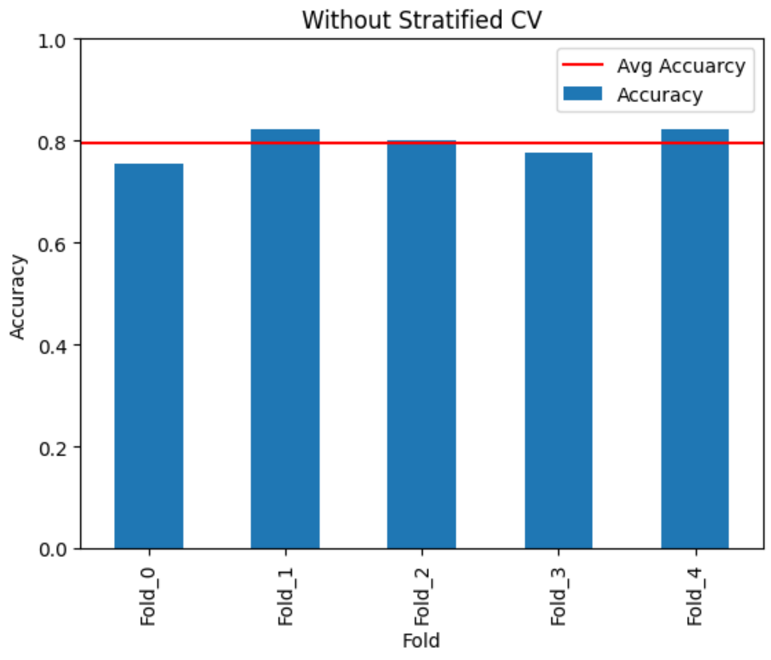


Figure 18: Accuracy plot for folds without stratified Cross validation.

Extra 3: Visualization for with stratification

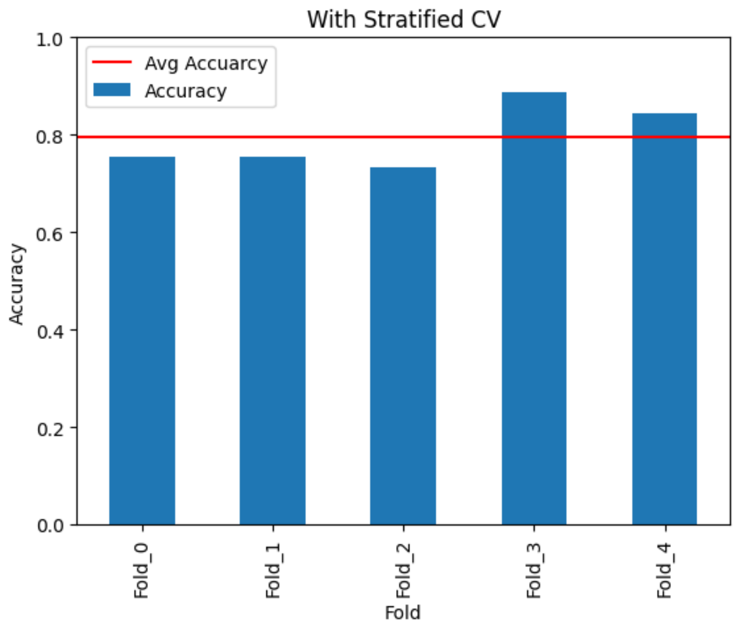


Figure 19: Accuracy plot for folds with stratified Cross validation.

Code

```
# init
kernal_type = 'poly'
degree = 3
c = 0.8
```

```
folds = 5

# build features and lables
x = train_dataset[COLUMNS[:-1]]
y = train_dataset[COLUMNS[-1]]

# build SVM
svc = SVC(kernel=kernal_type, degree=degree, C=c)

# init Cross Validation
cross_validation = KFold(n_splits=folds, shuffle=True,
random_state=RANDOM_STATE)

# perform cross validation
cross_validation_score = cross_val_score(svc, x, y, cv=cross_validation)
for i, score in enumerate(cross_validation_score):
    print(f'Fold {i}: Accuracy = {score:.5f}')

avg_acc = np.mean(cross_validation_score)
std_acc = np.std(cross_validation_score)

print(f'Average Accuracy: {avg_acc:.4f}')
print(f'Standard Deviation: {std_acc:.4f}')
```

BIOM5405/SYSC5405

Assignment 4

Part 5: Training without Cross Validation

Train another SVM model (same kernel & C) on all of your training samples. Test on the test subset. Report the accuracy on the test subset. Does it fall within 1 standard deviation of the average accuracy observed in Step 5?

Solution

Accuracy = 0.7866

Yes, it falls under 1 standard deviation of the accuracy observed in step 4

Extra 1: Note form the lectures

It can be seen that the accuracy drops slightly when compared with the Cross Validation set. This was a point mentioned in the lectures. This is because Cross Validation is an optimistic estimation of metrics

Extra 2: Visualization

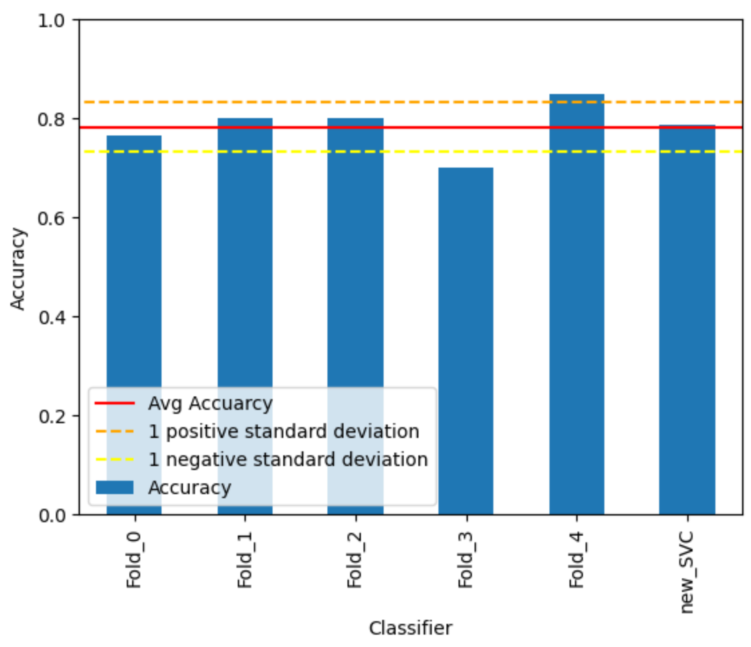


Figure 20: Accuracy visualization with standard deviation

Code:

```
# build SVM
svc = SVC(kernel=kernal_type, degree=degree, C=c)

# train SVM
svc.fit(train_dataset[COLUMNS[:-1]], train_dataset[COLUMNS[-1]])

# calculate accuracy of SVM
svc.score(test_dataset[COLUMNS[:-1]], test_dataset[COLUMNS[-1]])
```

BIOM5405/SYSC5405

Assignment 4

Part 6: Misclassification Cost

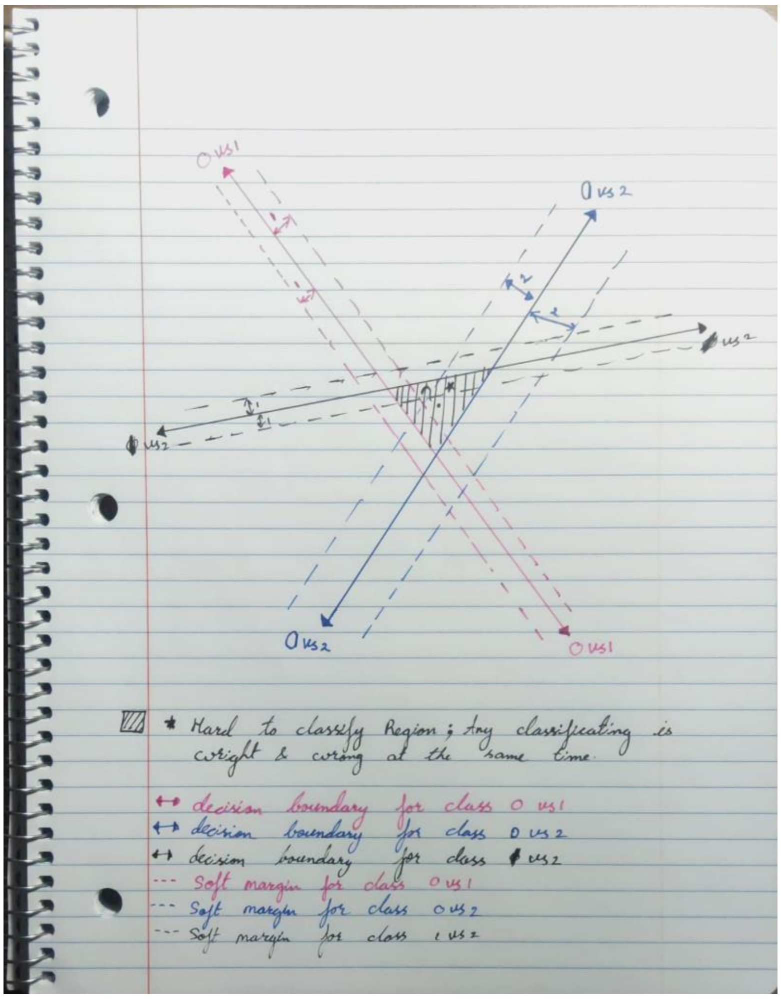
For this question only, assume that the misclassification costs are as follows:

		Actual		
		Low	Med	High
Predicted	Low	0	1	2
	Med	1	0	1
	High	2	1	0

a) What is your total misclassification cost for the test set predictions from Q5 above?
b) How could you incorporate this loss information into your classifier design? (60 words)

Solution:

a) Misclassification Cost: 23
b) Use one vs one classification to solve the multiclass problem [2]. Vary the soft margin by changing the α value in the "Hinge Loss" (see extra 2) to the cost value. A pictorial diagram is give below.



▨ * Hard to classify Region ; Any classificating is cwright & cwrong at the same time.

↔ decision boundary for class O vs 1
↔ decision boundary for class O vs 2
↔ decision boundary for class ϕ vs 2
--- Soft margin for class O vs 1
--- Soft margin for class O vs 2
--- Soft margin for class 1 vs 2

It is to be noted that, to simply the understanding "linear" kernel is shown instead of a "polynomial". One can easily replace the decision boundaries to polynomial.

BIOM5405/SYSC5405

Assignment 4

Extra 0: Alternative for incorporating the loss

You can simply pass in class weights in the SVC in SK-Learn. This will take into account the cost and create appropriate model.

Extra 1: Misclassification Cost Calculation

Misclassification cost = trace(confusion_matrix . cost_matrix^T)

But cost matrix is symmetric.

Misclassification cost = trace(confusion_matrix . cost_matrix)

Extra 2: Hinge Loss Formula

$$Hinge\_loss = \max(0, \alpha - y_i(w.x_i - b))$$

Here, α is always 1. y_i is the true class, x_i is the input feature, b is the bias term.

$(w.x_i - b)$ combined together is the score.

See reference [1] for further details.

Extra 3: Multi class classification for SVM

SVM typically work for binary classification. Different techniques are used to expand to multi-class problem. The techniques are as follows:

1. One vs One
2. One vs All

See reference [2] for further details.

Extra 4: References

[1] https://www.youtube.com/watch?v=IjSfa7Q8ngs
[2] https://www.youtube.com/watch?v=3lwicUTEgHs

Code

```
cost = np.array([
    [0,1,2],
    [1,0,1],
    [2,1,0]
])

# Note  y_pred is from part 5
confusion = confusion_matrix(test_dataset[COLUMNS[-1]], y_pred)
confusion

# Calculate the cost
np.trace(np.dot(confusion , cost))
```

BIOM5405/SYSC5405

Assignment 4

Part 7: Neural Networks (NN)

Using 5-CV across only the training subset, perform a hyperparameter sweep of the number of hidden nodes in a 3-layer feedforward neural network. Report your accuracy for `numH=[1,10,100]` hidden nodes. Use the 'adam' solver, a hyperbolic tangent activation function for the hidden layers.

Solution:

numH	Accuracy					Mean Folds Accuracy	Mean Fold STD
	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4		
1	0.7777	0.8000	0.7111	0.8666	0.8000	0.7911	0.0498
10	0.8666	0.9111	0.7111	0.8888	0.8888	0.8533	0.0724
100	0.8666	0.8888	0.7777	0.9111	0.8888	0.8666	0.0466

Extra 1: Hyperparameters for the NN model

Epochs: 2000

Learning Rate: 0.001

Folds: 5

Extra 2: Plots for training Losses and Accuracy for each epoch

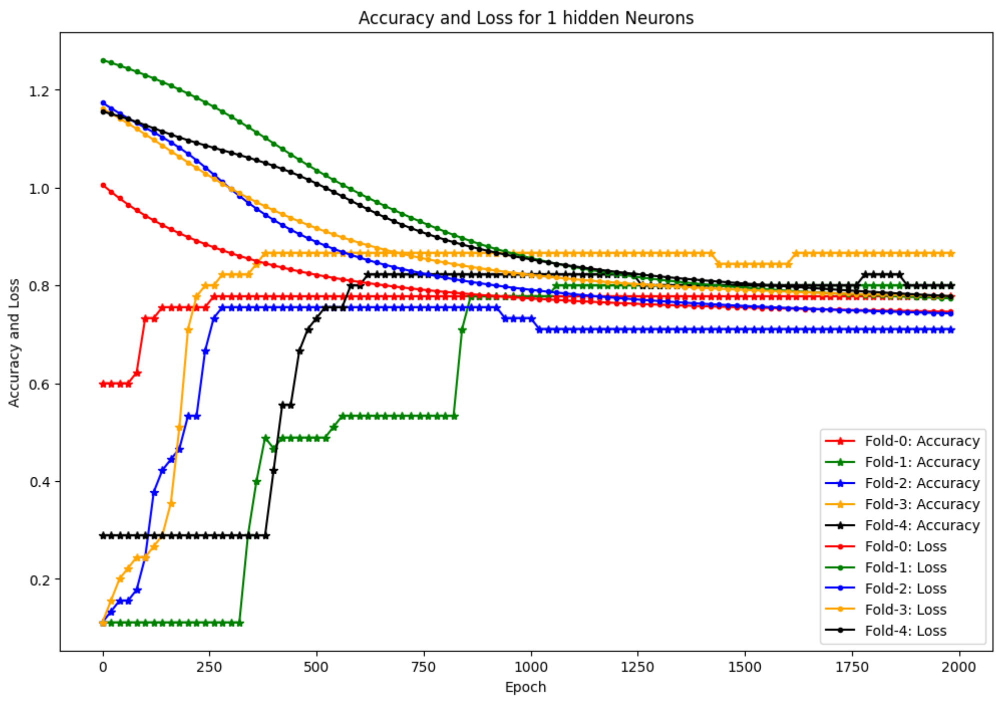


Figure 21: Accuracy and loss for 1 hidden neuron in each fold

BIOM5405/SYSC5405

Assignment 4

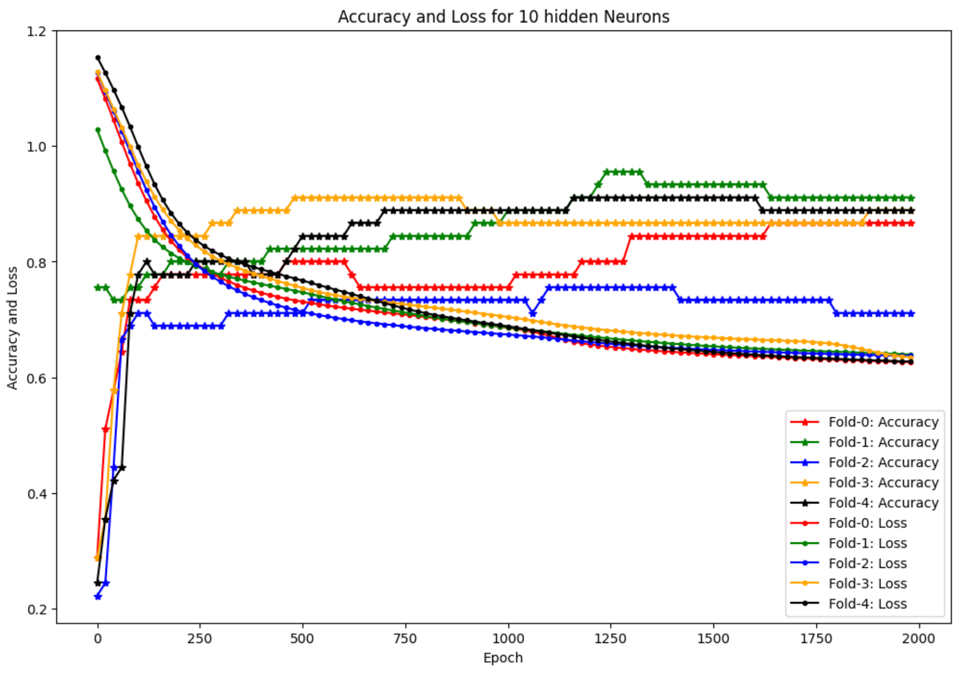


Figure 22: Accuracy and loss for 10 hidden neurons in each fold

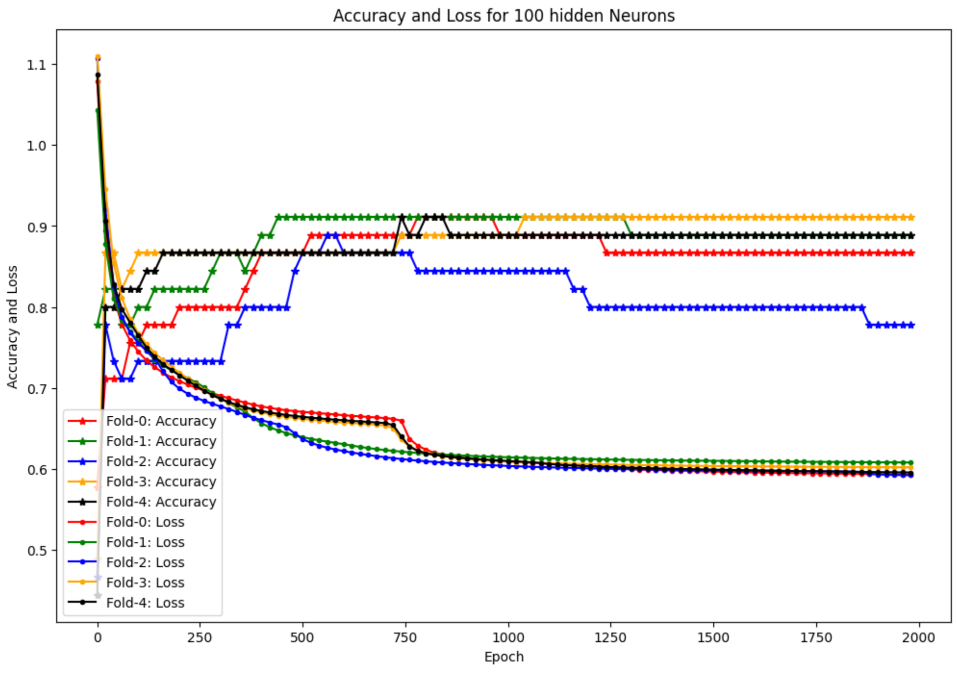


Figure 23: Accuracy and loss for 100 hidden neurons in each fold

Code:

```
# init
num_hidden = [1,  10,  100]
folds = 5
```

```python
epochs = 2000
learning_rate = 0.001

models = {}
losses = {}
accuracy = {}

# Define the model
class HouseEfficNN(torch.nn.Module):
    def __init__(self, hidden_size):
        super(HouseEfficNN, self).__init__()
        self.input_size = 5
        self.output_size = 3
        self.hidden_size = hidden_size

        self.layer_1 = torch.nn.Linear(self.input_size, self.hidden_size)
        self.activation = torch.nn.Tanh()
        self.layer_2 = torch.nn.Linear(self.hidden_size, self.output_size)

    def forward(self, x):
        x = self.layer_1(x)
        x = self.activation(x)
        x = self.layer_2(x)
        return x

# Define the dataset
class HousingDataset(torch.utils.data.Dataset):
    def __init__(self, df, target_column):
        self.features = torch.tensor(df.drop(columns=[target_column]).values,
dtype=torch.float32)
        self.labels = torch.tensor(df[target_column].values, dtype=torch.long)

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        return self.features[idx], self.labels[idx]

# run training
for h_num in num_hidden:
    losses[h_num] = {}
    accuracy[h_num] = {}
    cross_fold_NN = StratifiedKFold(n_splits=folds, shuffle=True,
random_state=RANDOM_STATE)
```

```
    for fold_index, (train_index, test_index) in
enumerate(cross_fold_NN.split(train_dataset[COLUMNS[:-1]],
train_dataset[COLUMNS[-1:]])):


        fold_nn_train_df = train_dataset.iloc[train_index]
        fold_nn_train_features, fold_nn_train_labels =
HousingDataset(fold_nn_train_df, COLUMNS[-1])[:]

        fold_nn_test_df = train_dataset.iloc[test_index]
        fold_nn_test_features, fold_nn_test_labels =
HousingDataset(fold_nn_test_df, COLUMNS[-1])[:]

        model = HouseEfficNN(hidden_size=h_num)

        optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
        criterion = torch.nn.CrossEntropyLoss()

        losses[h_num][fold_index] = []
        accuracy[h_num][fold_index] = []

        for epoch in tqdm(range(epochs), desc=f"H_size: {h_num}, Fold:
{fold_index}", unit='epoch',):
            model.train()
            optimizer.zero_grad()
            outputs = model(fold_nn_train_features)
            loss = criterion(outputs, fold_nn_train_labels)
            loss.backward()
            optimizer.step()

            losses[h_num][fold_index].append(loss.item())

            model.eval()
            predictions = model(fold_nn_test_features)
            acc = accuracy_score(fold_nn_test_labels,
predictions.argmax(1).detach().numpy())
            accuracy[h_num][fold_index].append(acc)

        models[h_num] = model
```

BIOM5405/SYSC5405

Assignment 4

Part 8: Naïve Bayes Classifier

Returning to Question 3, compare a naïve Bayes classifier trained using only the 'most useful' feature to a naïve Bayes classifier trained using all five features. Describe how you split/used your data, how you tested the hypothesis (null hypothesis, alternative hypothesis, test metric, etc.), what p-value you obtained, and your conclusion.

Solution:

H_0: The accuracy values for 5 feature classifier is similar to the accuracy of the 1 feature (Thermal) classifier

H_1: The accuracy values for 5 feature classifier is different from the accuracy values of 1 feature (Thermal) classifier

Data splitting: The data was first split into test (25%) and train (75%). On these 2 sets of data bootstrapping was conducted with replacements. On each iteration the test data was randomly sampled form the test set with repetition and the train data was randomly sampled with repetition from the train data. In each iteration the total number of data points (replacement samples) in the train set was 75% of 300, and total number of data points (replacement samples) for test set was 25% of 300.

Test metrics: accuracy

Accuracy = total number of correctly classified samples / total number of samples

Hypothesis Testing:

2 sample Bootstrapping was used to perform hypothesis testing. The repeated bootstrapped samples were taken as described in "Data splitting" and accuracy was measured for classifier with all 5 features and 1 feature (thermal) for each iteration. A further description of the process is found in "SYSC5405-Slides-04-ClassificationAccuracy" from slides 60 to 67. The code also a provided good understanding of the test. It is provided below.

P-value: 0.49924

Conclusion:

Given the p-values is above the significance level of 0.05, we cannot reject the null hypothesis.

In other works we fail to reject that there is a significance difference in the accuracy of the two classifiers.

Extra 1: Significance Level (alpha)

For this problem the significance level was 0.05.

The confidence level is 95%

Extra 2: Number of iterations

For this problem 9,999 iterations were conducted.

Extra 3: Distribution of mean under H0

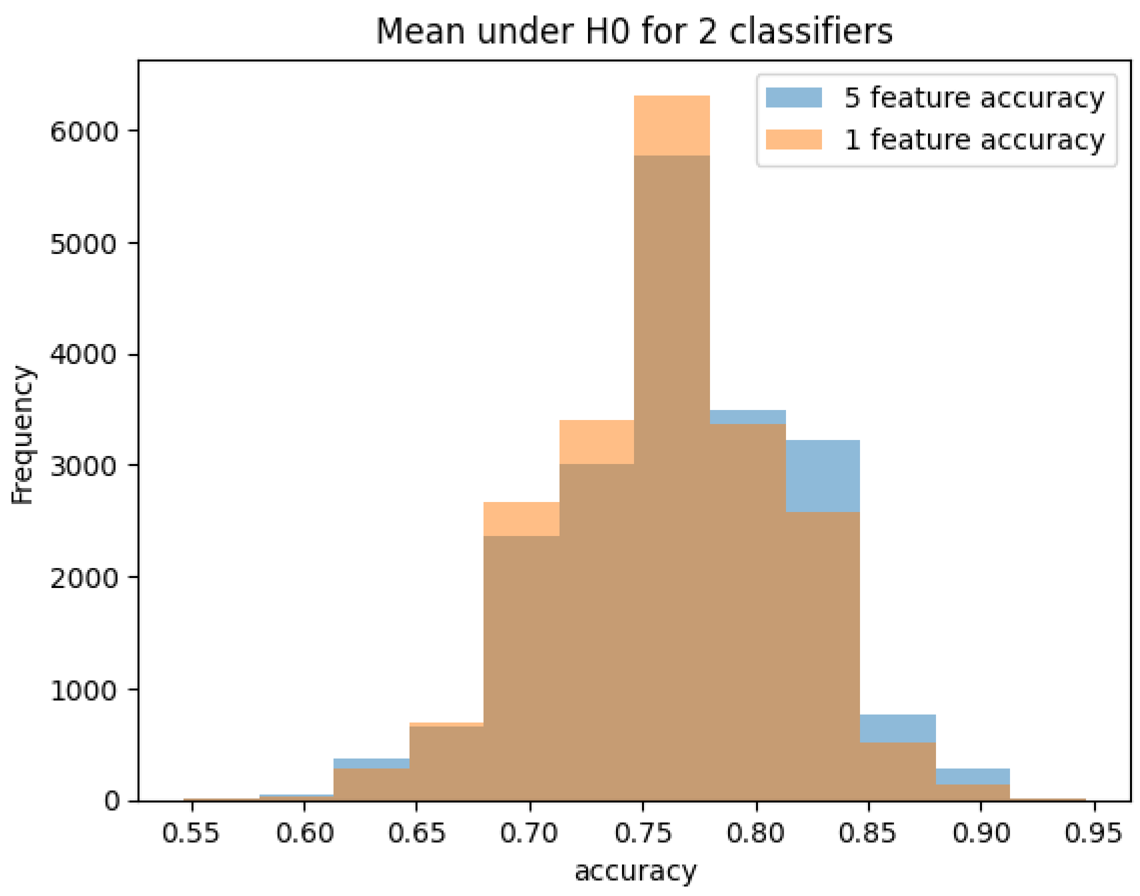


Figure 24: Distribution of Mean under H0

Code

```
# init
num_iters = 9999

accuracies_5_features = []
accuracies_1_features = []
accuracies_diff = []
# calculate the diffrence of accuracies for the two models
real_classifier_5 = GaussianNB()
real_classifier_5.fit(train_dataset[COLUMNS[:-1]], train_dataset[COLUMNS[-1]])
y_pred_real_5 = real_classifier_5.predict(test_dataset[COLUMNS[:-1]])
accuracy_5 = accuracy_score(test_dataset[COLUMNS[-1]], y_pred_real_5)

real_classifier_1 = GaussianNB()
real_classifier_1.fit(train_dataset[[COLUMNS[0]]], train_dataset[COLUMNS[-1]])
y_pred_real_1 = real_classifier_1.predict(test_dataset[[COLUMNS[0]]])
accuracy_1 = accuracy_score(test_dataset[COLUMNS[-1]], y_pred_real_1)
```

```
real_diff = accuracy_1 - accuracy_5
print(f"1 feature: {accuracy_1}")
print(f"5 feature: {accuracy_5}")


gnb = GaussianNB()
for _ in range(num_iters):
    iter_train_df = train_dataset.sample(
        len(train_dataset),
        replace = True
    )

    iter_test_df = test_dataset.sample(
        len(test_dataset),
        replace = True
    )

    gnb.fit(iter_train_df[COLUMNS[:-1]], iter_train_df[COLUMNS[-1]])
    y_pred_5 = gnb.predict(iter_test_df[COLUMNS[:-1]])
    accuracy_5 = accuracy_score(iter_test_df[COLUMNS[-1]], y_pred_5)
    accuracies_5_features.append(accuracy_5)

    # nb_classifier_1 = GaussianNB()
    gnb.fit(iter_train_df[[COLUMNS[0]]], iter_train_df[COLUMNS[-1]])
    y_pred_1 = gnb.predict(iter_test_df[[COLUMNS[0]]])
    accuracy_1 = accuracy_score(iter_test_df[COLUMNS[-1]], y_pred_1)
    accuracies_1_features.append(accuracy_1)

    diff_accuracy = accuracy_1 - accuracy_5
    accuracies_diff.append(diff_accuracy)

p_val = sum([diff >= real_diff for diff in accuracies_diff])/num_iters

print(f”p-value {p_val}”)
```

BIOM5405/SYSC5405

Assignment 4

Appendix:

Unimportant code:

```
import torch
import torch
import numpy as np
import pandas as pd
from tqdm import tqdm
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from torch.utils.data import Dataset, DataLoader
from sklearn.model_selection import train_test_split, cross_val_score, KFold,
StratifiedKFold
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.naive_bayes import GaussianNB

device = torch.device('cpu')
```

```
RANDOM_STATE = 1
ALPHA = 0.5
COLOR = {
    0: (1.0,0.0,0.0,ALPHA),
    1: (1.0,1.0,0.0,ALPHA),
    2: (0.0,1.0,0.0,ALPHA)
}
LABELS = {
    0: 'low',
    1: 'med',
    2: 'high'
}
```