Aprendizagem 2023
Homework II – Group 023
(ist102848, ist103525)

Part I: Pen and paper

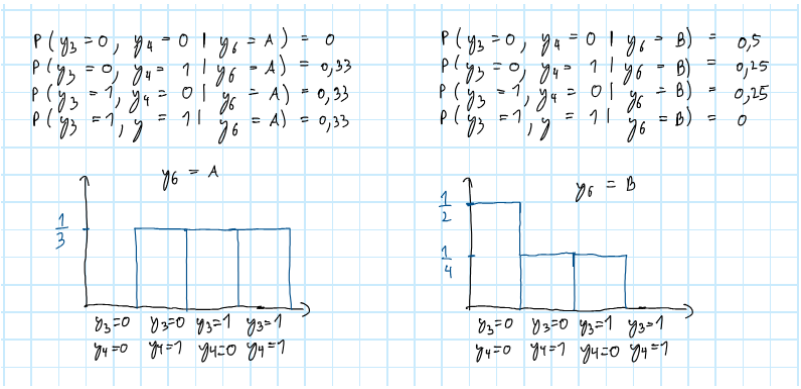
1. Consider x1–x7 to be training observations, x8–x9 to be testing observations, y1– y5 to be input variables and y6 to be the target variable.

   (a) Learn a Bayesian classifier and show all parameters (distributions and priors for subsequent testing)

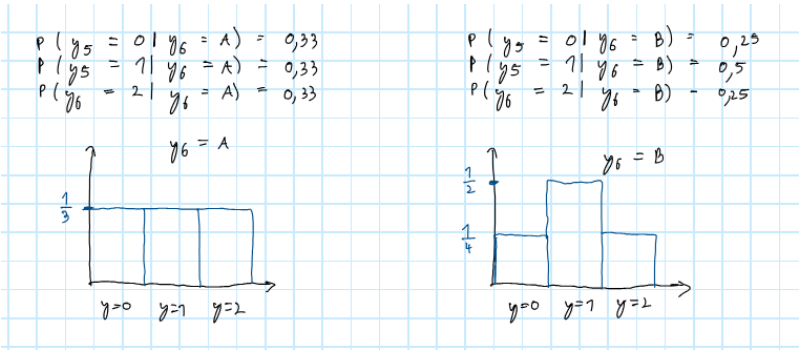
   To develop a Bayesian classifier, we initiated the process by calculating the priors based on the available dataset:

   $$P(y_6 = A) = \frac{3}{7} \quad \text{e} \quad P(y_6 = B) = \frac{4}{7}$$

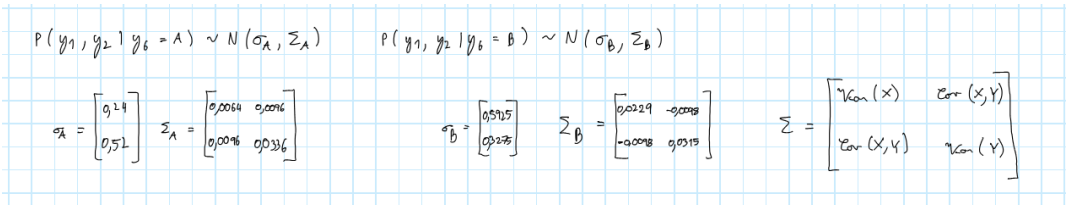
   Next, we focused on computing the class conditional Probability Mass Function (PMF) for the set {y3, y4}. Since they are dependent we have to calculate the probability of the set and not of each variable individually.

   Additionally, we calculated the PMF for y5:

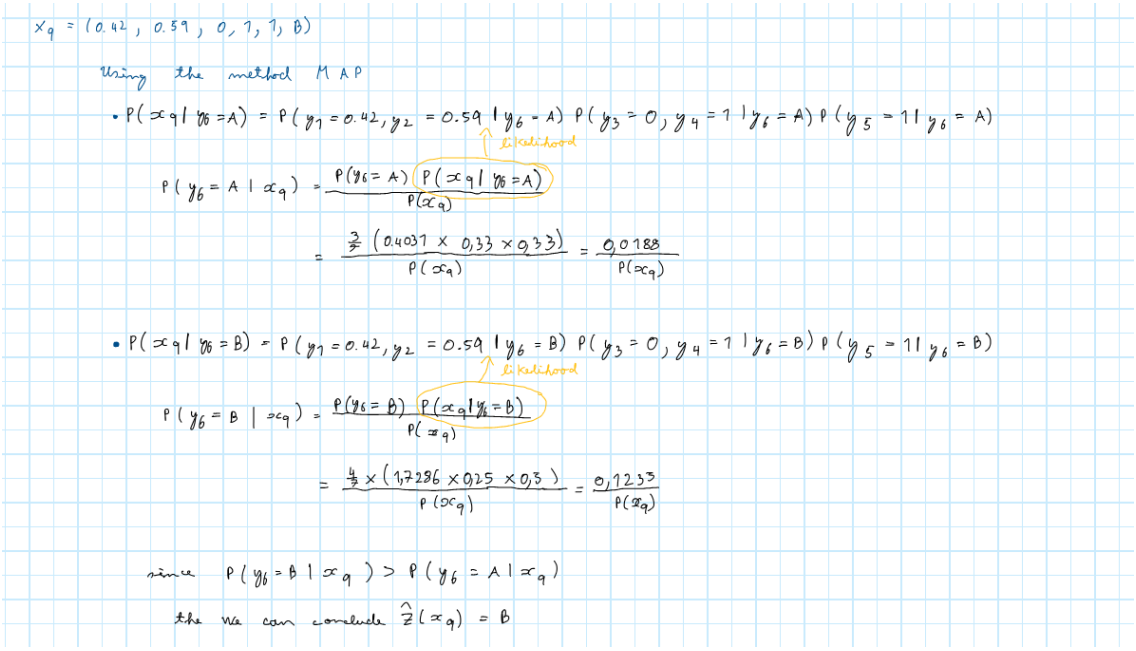
   

Finally, given that $y_1 \times y_2$ follows a normal distribution and they are both dependent on each other, we calculated the class conditional Probability Density Function. In this case, it is a multivariate Gaussian Distribution with 2 dimensions. Its parameters include the mean of y_1 and y_2 (a 2×1 matrix) and the Covariance Matrix (2×2).



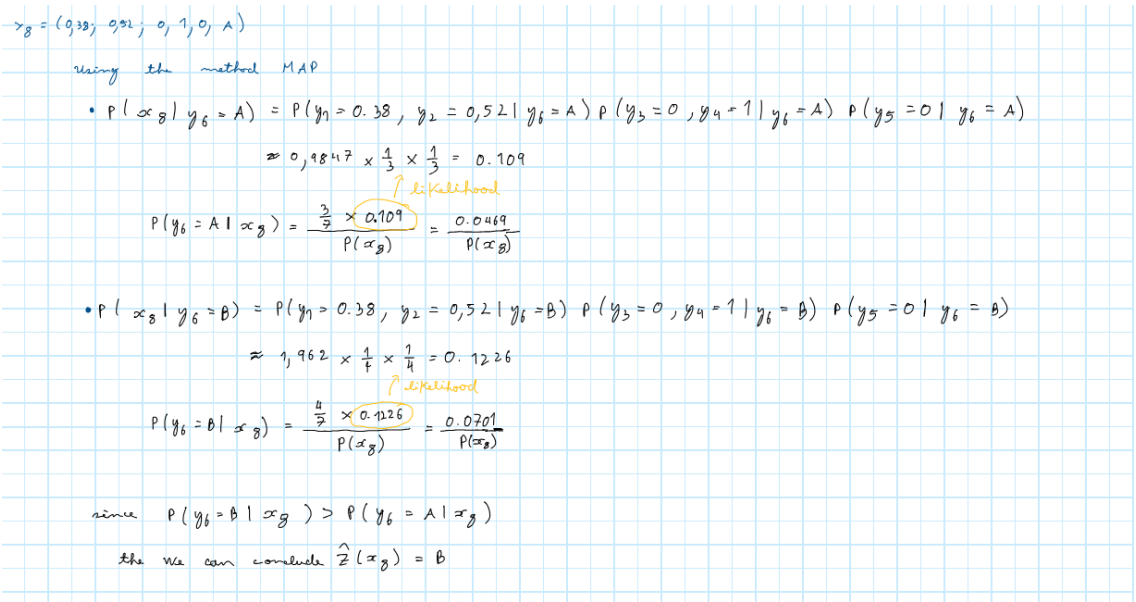
(b) Under MAP, classify each testing observation showing all your calculus.

Using the MAP method to classify x_9, we first calculate the likelihood for x_9. Since the sets $\{y_1, y_2\}$, $\{y_3, y_4\}$, and $\{y_5\}$ are independent, we can multiply the probability of each set.



After having calculated the probabilities for each class, notice how the probability for class B is higher. Therefore, according to Bayesian learning principles, the model will classify x_9 as class B.

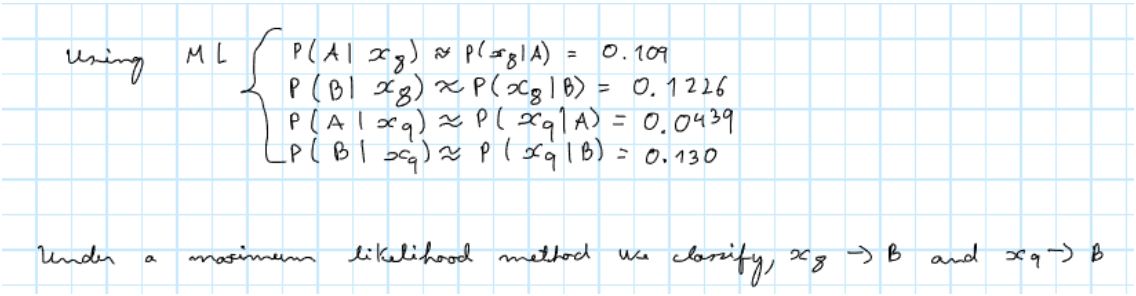
Now we will do the same for x_8:



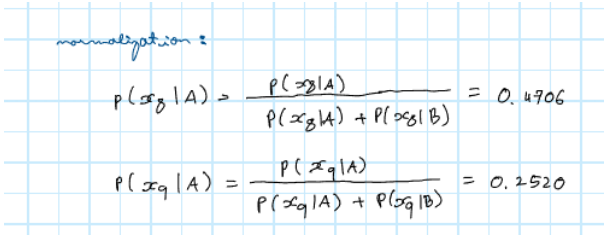
By the same logic as above, our model will classify x_8 as B.

(c) Under a maximum likelihood assumption, what thresholds optimize testing accuracy?

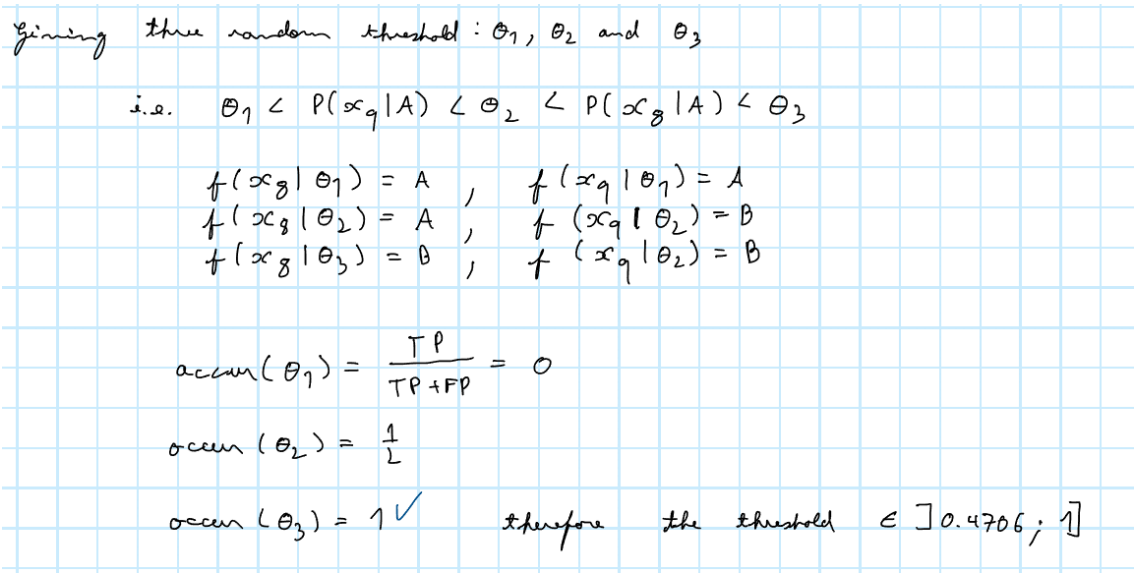
Under a maximum likelihood assumption, because $P(x|B)$ is larger, x_8 and x_9 are classified as B.



Normalizing the likelihoods to ensure $\sum_{h \in y_{\text{out}}} p(h|x) = 1$.



Now, there are 3 possible outcomes for the accuracy using the given function depending on the value the threshold assumes. One happens when the threshold is lesser than both the normalized likelihoods of x8 and x9. One for when the threshold is between them and another when it is larger.



Notice how, when we choose θ_3 (a threshold larger than both likelihoods), the function classifies x_8 and x_9 as B, which is exactly what was calculated under a maximum likelihood (ML) assumption at the beginning of this exercise (meaning accuracy is 100%). The same can't be said for θ_1 and θ_2.

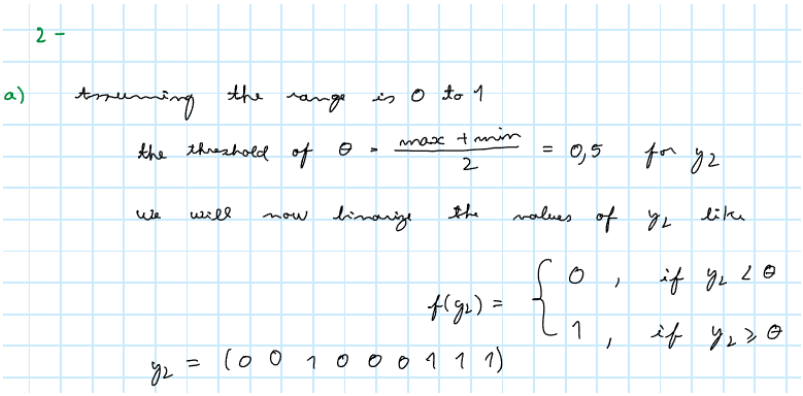
So, for values of the threshold in the range]0.4706; 1], the testing accuracy is maximized.

2. Consider a 3-fold cross-validation over the full dataset (x1- x9) without shuffling the observations.

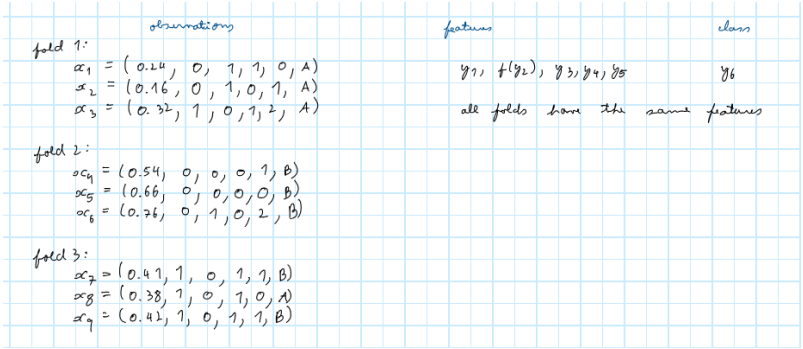
   (a) Identify the observations and features per data fold after the binarization procedure.

   First, let's binarize y_2 using an equal width approach. We will assume, as mentioned in the FAQ, that y_2 ranges between 0 and 1. This implies that an equal-range discretization will divide the data into two bins: 0-0.5 and 0.5-1.

   Applying this binarization function to y_2, we obtain:



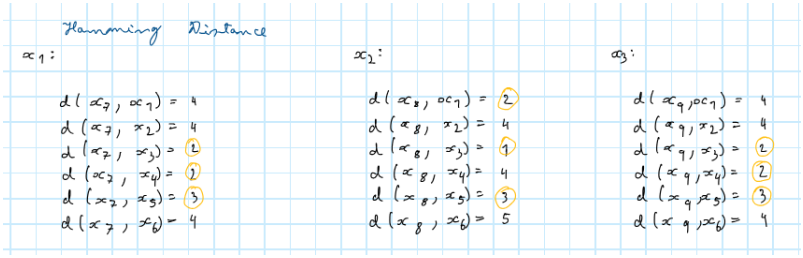
   Given that this is a 3-fold cross-validation, we will split the observations into three equal-sized parts: x_1-x_3, x_4-x_6, x_7-X_9. These folds will later be used to partition the data into testing and training sets:



(b) Compute the MAE of a distance-weighted kNN with $k = 3$ (Hamming distance (d), and $1/d$ weighting) regressor for the 1st iteration of the cross-validation

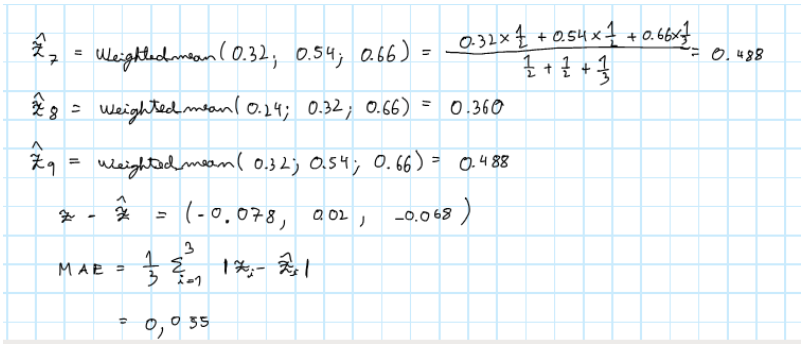
Given that we will only compute for the first iteration, we will exclusively test using the bin (x_7-x_9) and train with the remaining observations.

To begin, we calculate the Hamming distance for x_7, x_8, and x_9 with every observation in the training set. The Hamming distance is computed by counting the number of features that differ between two observations.



Next, since this is a k-NN (k-Nearest Neighbors) regressor with $k = 3$, we select the three values that are closest to 0 (these have already been highlighted in the image above).

Given that the weight function is calculated as $1/d$, we can now compute the predicted value for our testing observations by calculating the weighted mean of the three closest vertices. This involves multiplying each vertex's output by its weight (determined by the inverse of its distance). Subsequently, we can compute the Mean Absolute Error for the first fold, by comparing the predicted value with the real.



Part II: Programming

In the next image, we have provided an overview of all the imports utilized and our approach to importing the dataset. Specifically, 'X' stores all of the input variables values, and 'y' stores the Output Class values. The folds variable is a cross-validator that will split the dataset into 10 shuffled folds. To streamline the code, we've omitted this information from the beginning of each section to avoid redundancy.

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from scipy.io import arff

# Load the dataset
data = arff.loadarff('../data/column_diagnosis.arff')
df = pd.DataFrame(data[0])

X = df.drop('class', axis=1)
y = df['class'].astype(str)

folds = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
```

✓ 0.0s

1. Compare the performance of kNN with $k = 5$ and naïve Bayes with Gaussian assumption

   (a) Plot two boxplots with the fold accuracies for each classifier.

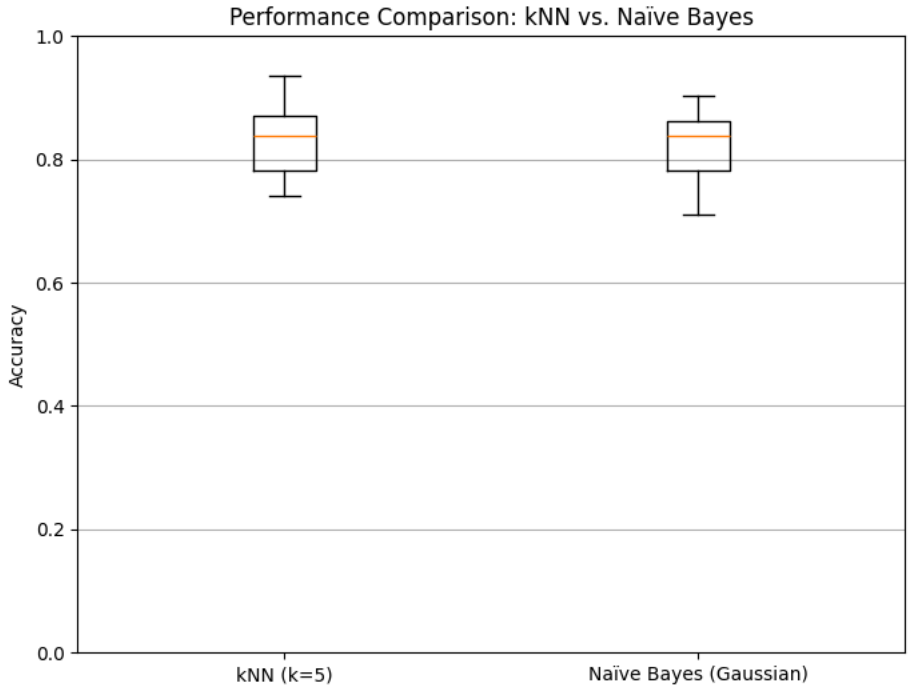
We created the two classifiers and calculated the accuracies for both of them using the cross_val_score function (feeding it the validator already created in the setup).

```
knn_classifier = KNeighborsClassifier(n_neighbors=5)
nb_classifier = GaussianNB()

knn_scores = cross_val_score(knn_classifier, X, y, cv=folds)
nb_scores = cross_val_score(nb_classifier, X, y, cv=folds)
```

We then plotted the two boxplots of the classifiers using the scores/accuracies returned by the cross_val_score function.

```
plt.figure(figsize=(8, 6))
plt.boxplot([knn_scores, nb_scores], labels=['kNN (k=5)', 'Naïve Bayes (Gaussian)'])
plt.title('Performance Comparison: kNN vs. Naïve Bayes')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.grid(axis='y')
plt.show()
```

Output:



(b) Test the hypothesis "kNN is statistically superior to naïve Bayes regarding accuracy", asserting whether is true.

To test this hypothesis we are going to assume a null hypothesis of knn_scores > nb_scores.

We will then perform a t-test, to analyze the collected data and assess whether there is sufficient evidence to reject the null hypothesis. If the p-value obtained from the statistical test is below a certain significance level (e.g., 0.05), we may conclude that there is enough evidence to reject the null hypothesis.

And so we begin by creating a t-test that tests the hypothesis "kNN is statistically superior to Naïve Bayes regarding accuracy".

```
# Compare the two classifiers using a paired t-test
t_statistic, p_value = stats.ttest_rel(knn_scores, nb_scores, alternative='greater')

alpha = 0.05

# Print the results
print(f"t-statistic: {t_statistic}")
print(f"p-value: {p_value}")
if p_value < alpha:
    print(p_value, "<", alpha)
    print("Reject the null hypothesis")
    print("kNN is statistically superior to Naïve Bayes regarding accuracy")
else:
    print(p_value, ">=", alpha)
    print("Fail to reject the null hypothesis")
    print("There is no significant difference in accuracy between kNN and Naïve Bayes")
```

By running and printing the values, we get the following results:

$$t\text{-statistic} = 0.9210,\ p\text{-value} = 0.190$$

We reject the null hypothesis for p-value $\leq \alpha$

Therefore, as the p-value is bigger than the usual significance levels (1%, 5% and 10%) we can conclude that there is no evidence that kNN is superior to Naïve Bayes regarding accuracy.

2. Consider two kNN predictors with $k = 1$ and $k = 5$. Plot the differences between the two cumulative confusion matrices of the predictors. Comment.

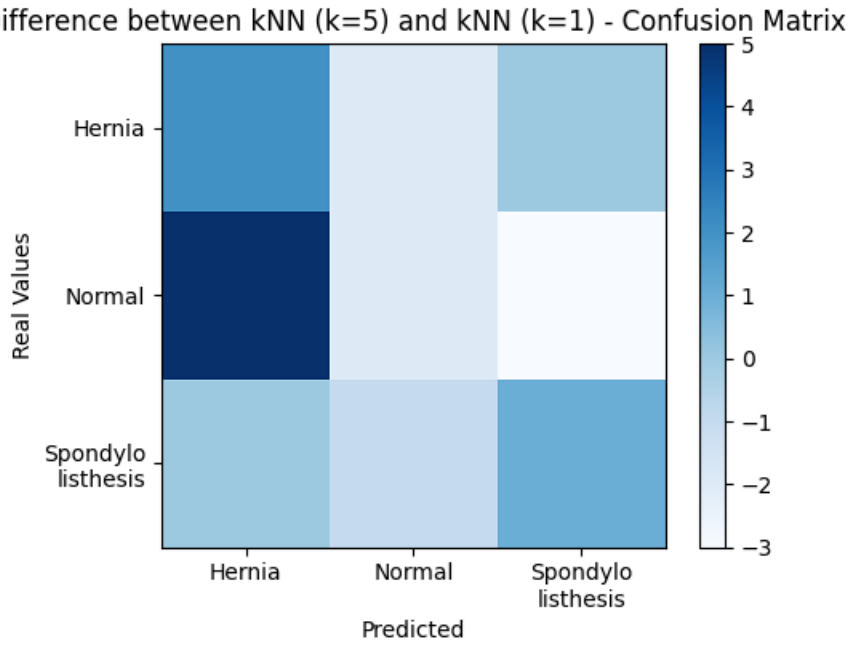
We first calculate the confusion matrix for each fold, accumulating the results in one single matrix (variable cm), for each kNN.

```
# Initialize kNN's
knn_1 = KNeighborsClassifier(n_neighbors=1, weights='uniform', metric='euclidean')
knn_5 = KNeighborsClassifier(n_neighbors=5, weights='uniform', metric='euclidean')


#Initialize cumulative confusion matrix
cm = np.zeros((3, 3))

for train_index, test_index in folds.split(X, y):
    train_index = train_index.astype(int)
    test_index = test_index.astype(int)

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    knn_1.fit(X_train, y_train)
    knn_5.fit(X_train, y_train)

    y_pred_1 = knn_1.predict(X_test)
    y_pred_5 = knn_5.predict(X_test)

    cm += confusion_matrix(y_test, y_pred_5) - confusion_matrix(y_test, y_pred_1)
```

Plotting the two confusion matrix

```
plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.title("Difference between kNN (k=5) and kNN (k=1) - Confusion Matrix")
plt.imshow(cm, cmap='Blues', interpolation='nearest')
plt.colorbar()
plt.xticks([0, 1, 2], labels=['Hernia', 'Normal', 'Spondylo\nlisthesis'])
plt.yticks([0, 1, 2], labels=['Hernia', 'Normal', 'Spondylo\nlisthesis'])
plt.xlabel("Predicted")
plt.ylabel("Real Values")
```

Output:



The comparison between the cumulative confusion matrices for $k = 5$ and $k = 1$ reveals similar results.

The most notable difference is that when $k = 5$, there are more instances where Hernias are predicted instead of Normal, whereas when $k = 1$, there are more instances where Spondylolisthesis is predicted instead of Normal. Being these differences practically irrelevant when compared to the sample size.

Taking these observations into account, we can conclude that the choice of the k value does not significantly impact the overall performance of the k-Nearest Neighbors (kNN) classifiers in this particular context.

3. Identify three possible difficulties of naïve Bayes when learning from the given dataset.

1. By using naïve Bayes, we are assuming that all variables are independent, which for our specific case may not be true. For example, "Pelvic tilt and sacral slope are two angles directly correlated with the pelvic incidence". Click here to read the full article.

2. We are assuming that all features have equal significance, which may not reflect reality. For instance, the 'degree_spondylolisthesis' feature could potentially hold more predictive weight than 'lumbar_lordosis_angle' when it comes to diagnosing the condition.

3. We are assuming all the features in our dataset follow a Gaussian distribuiton. If the variables do not fit well with a Gaussian, Naïve Bayes may yield suboptimal results.

For instance, when plotting the variable 'degree_spondylolisthesis,' there is a lot of deviation from a perfect Gaussian distribution, suggesting that the Gaussian Naïve Bayes assumption may not fully capture the underlying data distribution, potentially leading to worse modeling results.