INTRODUCTION TO DATA SCIENCE

BITI 2513

TASK 3 (MODELING AND EVALUATION) OF CLUSTERING HEART DISEASE PATIENT DATA

LECTURE BY:

PROFESSOR MADYA DR SHARIFAH SAKINAH BINTI SYED AHMAD

GROUP MEMBER:

No	Name	Matric No.
1	VISHWAREETA VANOO	B031810196
2	PREVINA MUNUGANAN	B031810286
3	JEYSHALINI TEVOSHA	B031810246

4	**ZAITI AKTA BINTI ZAHARUDDIN**	**B031810365**

INDEX

CLUSTERING TECHNIQUES

KMEANS CLUSTERING

Kmeans clustering is one of the most popular clustering algorithms and usually the first thing practitioners apply when solving clustering tasks to get an idea of the dataset structures. The goal of kmeans is to group data points into distinct non-overlapping subgroups. It does a very good job when the clusters have a kind of spherical shapes. However, it suffers as the geometric shapes of clusters deviates from spherical shapes. Moreover, it also doesn't learn the number of clusters from the data and requires it to be pre-defined. Below, we covered both strength, weakness, and some evaluation method related to kmeans:

1. Scale or standardize the data when applying kmeans algorithm.
2. Elbow method in selecting number of clusters doesn't usually work because the error function is monotonically decreasing for all *k*s.
3. Kmeans gives more weight to the bigger clusters.
4. Kmeans assumes spherical shapes of clusters (with radius equal to the distance between the centroid and the furthest data point) and doesn't work well when clusters are in different shapes such as elliptical clusters.
5. If there is overlapping between clusters, kmeans doesn't have an intrinsic measure for uncertainty for the examples belong to the overlapping region in order to determine for which cluster to assign each data point.
6. Kmeans may still cluster the data even if it can't be clustered such as data that comes from *uniform distributions*.

K-Means starts by randomly defining *k* centroids. From there, it works in iterative (repetitive) steps to perform two tasks:

1. Assign each data point to the closest corresponding centroid, using the standard Euclidean distance. In layman's terms: the straight-line distance between the data point and the centroid.
2. For each centroid, calculate the mean of the values of all the points belonging to it. The mean value becomes the new value of the centroid.

Once step 2 is complete, all of the centroids have new values that correspond to the means of all of their corresponding points. These new points are put through steps one and two producing yet another set of centroid values. This process is repeated over and over until there is no change in the centroid values, meaning that they have been accurately grouped. Or, the process can be stopped when a previously determined maximum number of steps has been met.

MEAN SHIFT CLUSTERING

Mean shift clustering is a simple and flexible clustering technique that has several nice advantages over other approaches. Mean shift clustering aims to discover “blobs” in a smooth density of samples. It is a centroid-based algorithm, which works by updating candidates for centroids to be the mean of the points within a given region. These candidates are then filtered in a post-processing stage to eliminate near-duplicates to form the final set of centroids.

The mean shift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters.

Given n data points x_i, i = 1, ..., n on a d-dimensional space Rd , the multivariate kernel density estimate obtained with kernel K(x) and window radius h is

$$f(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right).$$

For radially symmetric kernels, it suffices to define the profile of the kernel k(x) satisfying

$$K(\mathbf{x}) = c_{k,d} k(\|\mathbf{x}\|^2)$$

where $c_{k,d}$ is a normalization constant which assures K(x) integrates to 1. The modes of the density function are located at the zeros of the gradient function ∇f(x) = 0.

The gradient of the density estimator is

$$\begin{aligned} \nabla f(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}) g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}\right] \end{aligned}$$

where g(s) = −k 0 (s). The first term is proportional to the density estimate at x computed with kernel G(x) = $c_{g,d}$g(||x||^2) and the second term

$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}$$

is the mean shift. The mean shift vector always points toward the direction of the maximum increase in the density. The mean shift procedure, obtained by successive

• computation of the mean shift vector $m_h(x^t)$,

• translation of the window $x^{t+1} = x^t + m^h(x^t)$

is guaranteed to converge to a point where the gradient of density function is zero.

RANDOM FOREST CLASSIFICATION

Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. It is an ensemble tree-based learning algorithm. Ensemble algorithms are those which combines more than one algorithms of same or different kind for classifying objects. For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object.

Random Forest Prediction for a classification problem:
f(x) = majority vote of all predicted classes over B trees

The 9 decision tree classifiers can be aggregated into a random forest ensemble which combines their input (on the right). The horizontal and vertical axes of the above decision tree outputs can be thought of as features x1 and x2. At certain values of each feature, the decision tree outputs a classification of "blue", "green", "red", etc.

These above results are aggregated, through model votes or averaging, into a single ensemble model that ends up outperforming any individual decision tree's output.

Features and Advantages of Random Forest :

1. It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.

2. It runs efficiently on large databases.

3. It can handle thousands of input variables without variable deletion.

4. It gives estimates of what variables that are important in the classification.

5. It generates an internal unbiased estimate of the generalization error as the forest building progresses.

6. It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

Disadvantages of Random Forest:

1. Random forests have been observed to overfit for some datasets with noisy classification/regression tasks.

2. For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.

Implementation of Random Forest Classification on real life dataset:

1. Importing Python Libraries and Loading our Dataset into a Data Frame

2. Splitting our dataset into training set and test set

3. Creating a Random Forest Regression model and fitting it to the training data

4. Predicting the test set results and making the Confusion matrix

VALIDATION PROCESS

CROSS VALIDATION

Cross-validation is a statistical method used to estimate the skill of machine learning models. It is also a resampling procedure used to evaluate machine learning models on a limited data sample.

It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1) Take the group as a hold out or test data set
    2) Take the remaining groups as a training data set
    3) Fit a model on the training set and evaluate it on the test set
    4) Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

It is also important that any preparation of the data prior to fitting the model occur on the CV-assigned training dataset within the loop rather than on the broader data set. This also applies to any tuning of hyperparameters. A failure to perform these operations within the loop may result in data leakage and an optimistic estimate of the model skill.

The results of a k-fold cross-validation run are often summarized with the mean of the model skill scores. It is also good practice to include a measure of the variance of the skill scores, such as the standard deviation or standard error.

The k value must be chosen carefully for your data sample.

A poorly chosen value for k may result in a mis-representative idea of the skill of the model, such as a score with a high variance (that may change a lot based on the data used to fit the model), or a high bias, (such as an overestimate of the skill of the model).

Three common tactics for choosing a value for k are as follows:

- Representative: The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.
- k=10: The value for k is fixed to 10, a value that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.
- k=n: The value for k is fixed to n, where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.

If a value for k is chosen that does not evenly split the data sample, then one group will contain a remainder of the examples. It is preferable to split the data sample into k groups with the same number of samples, such that the sample of model skill scores are all equivalent.

EVALUATION

ACCURACY

Accuracy refers to how close a sample statistic is to a population parameter . Thus, if you know that a sample mean is 99 and the true population mean is 100, you can make a statement about the sample accuracy. For example, you might say the sample mean is accurate to within 1 unit.

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model. The accuracy formula with using confusion matrix data:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

True positive and true negatives are the observations that are correctly predicted and therefore shown in green. We want to minimize false positives and false negatives so they are shown in red color. These terms are a bit confusing. So let's take each term one by one and understand it fully.

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.
False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

False Positives (FP) – When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN) – When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.
Once you understand these four parameters then we can calculate Accuracy, Precision, Recall and F1 score.

PRECISION

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. The precision formula:

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Immediately, you can see that Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.

Precision is a good measure to determine, when the costs of False Positive is high. For instance, email spam detection. In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

RECALL

Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

So Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

For instance, in fraud detection or sick patient detection. If a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be very bad for the bank.

F1-SCORE

Now if you read a lot of other literature on Precision and Recall, you cannot avoid the other measure, F1 which is a function of Precision and Recall. The formula is as follows:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

F1 Score is needed when you want to seek a balance between Precision and Recall. Right…so what is the difference between F1 Score and Accuracy then? We have previously seen that accuracy can be largely contributed by a large number of True Negatives which in most business circumstances, we do not focus on much whereas False Negative and False Positive usually has business costs (tangible & intangible) thus F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

STEPS AND RESULT

The non-numerical values were changed into numerical values.

```
#importing dataset and converting to datasframe
df = pd.read_csv('modified_health.csv')
df.dtypes
number= LabelEncoder()
df['Gender']=number.fit_transform(df['Gender'].astype('str'))
df['Diabetes Status (Yes/No)']=number.fit_transform(df['Diabetes Status (Yes/No)'].astype('str'))
df['High Blood Pressure (Yes/No)']=number.fit_transform(df['High Blood Pressure (Yes/No)'].astype('str'))
df['Heart Disease (Yes/No)']=number.fit_transform(df['Heart Disease (Yes/No)'].astype('str'))
df['Tobacco Use (Yes/No)']=number.fit_transform(df['Tobacco Use (Yes/No)'].astype('str'))
df['Fruits & Vegetable Consumption']=number.fit_transform(df['Fruits & Vegetable Consumption'].astype('str')
df['Exercise']=number.fit_transform(df['Exercise'].astype('str'))
data = pd.DataFrame(df) #data frame
```

The dependent variable and the independent variables were separated accordingly for training purpose.

```
#extracting columns x and y
x = data.iloc[:, 0:8]
x = x.drop(columns=['Heart Disease (Yes/No)'], axis=1)
print(x.dtypes)
print ("================================================================================")
x = pd.DataFrame(scale(x))


y = data.iloc[:, 3]
print(y.head())
print ("================================================================================")
```

Cross validation was used for validation process.

All the parameters were changed few times in order to get the best of the process.

KMEANS CLUSTERING

PARAMETERS

Folds for Cross Validation : 8

```
clusters = 2

model = KMeans(init='k-means++', n_clusters=clusters,
               random_state=20, max_iter=70)

scores = cross_val_score(model, x, y, scoring='accuracy', cv=8)
```

RESULT

8-Fold accuracy

```
========================================
8-Fold Accuracy :  50.40884672922616
========================================
```

Total accuracy

```
========================================
Accuracy(Total) =  57.85244704163623
========================================
```

Confusion Matrix

```
[[0.51862673 0.40467495]
 [0.01680058 0.05989774]]
==========================
```

Precision, recall, f1score and support

```
============================================================
              precision    recall  f1-score   support

           0       0.97      0.56      0.71      1264
           1       0.13      0.78      0.22       105

    accuracy                           0.58      1369
   macro avg       0.55      0.67      0.47      1369
weighted avg       0.90      0.58      0.67      1369

============================================================
```

Execution Time (s)

```
==============================
Program Executed in 3.1415468
==============================
```

MEAN-SHIFT CLUSTERING

PARAMETERS

Better performance was produced only by default parameters.

Folds for Cross Validation : 8

```
model =MeanShift()

scores = cross_val_score(model, x, y, scoring='accuracy', cv=8)
```

RESULT

8-Fold accuracy

```
=====================================
8-Fold Accuracy :  87.06905344757241
=====================================
```

Total accuracy

```
=====================================
Accuracy(Total) =  87.0708546384222
=====================================
```

Confusion matrix

```
[[0.86486486 0.05843682]
 [0.07085464 0.00584368]]
============================
```

Precision, recall, f1score, support

```
=============================================================
              precision    recall  f1-score   support

           0       0.92      0.94      0.93      1264
           1       0.09      0.08      0.08       105

    accuracy                           0.87      1369
   macro avg       0.51      0.51      0.51      1369
weighted avg       0.86      0.87      0.87      1369

=============================================================
```

Execution time(s)

```
=======================================
Program Executed in 85.61345349999999
=======================================
```

RANDOM FOREST CLASSIFICATION

Confusion Matrix

```
=== Confusion Matrix ===
[[401  14]
 [ 32   5]]
```

Precision, Recall, f1-score, and support

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       415
           1       0.26      0.14      0.18        37

    accuracy                           0.90       452
   macro avg       0.59      0.55      0.56       452
weighted avg       0.87      0.90      0.88       452
```

AUC Scores

```
=== All AUC Scores ===
[0.78910488 0.70618306 0.77361246 0.69498539 0.8218111  0.68232717
 0.72176241 0.72930867]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.739886893170121
```

ANALYSIS

Compare the accuracy, precision, recall, f1score and determine the best algorithm

	Kmeans Clustering	Mean Shift Clustering	Random Forest Classification
Accuracy	57.85	87.07	-
Precision	0.13	0.09	0.26
Recall	0.78	0.08	0.14
F1-Score	0.22	0.08	0.18

Best Algorithm : **Mean Shift Clustering**

Reason : Because, Mean Shift Clustering got 87.07 accuracy which mean the higher and 0.09 precision, 0.08 recall and 0.08 F1-score which is the least between the other algorithm techniques.

REFFERENCE

1.https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15

2. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

3. https://spin.atomicobject.com/2015/05/26/mean-shift-clustering/

4. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html#:~:text=Mean%20shift%20clustering%20using%20a,points%20within%20a%20given%20region.&text=If%20not%20set%2C%20the%20seeds%20are%20calculated%20by%20clustering.