Hacettepe University
Computer Engineering Department

BBM469 Data Intensive Applications Laboratory- 2022 Spring

Assignment 2

Clustering and Classification with Python

March 31, 2022

Uğurcan ERDOĞAN
21827373

Alperen Berk IŞILDAR
21827494

Table of Contents

1 Problem Definition

In this assignment, we were asked to understand the classification and clustering algorithms using python libraries and perform a basic experiment with appropriate datasets. We have also dealt with data manipulation and data normalization.

1.1 Packages

We have installed required and necessary Python packages that stated in the requirements.txt

```
# Installing necessary modules from the text file
!pip install -r requirements.txt
```

requirements.txt

```
pandas~=1.3.4
seaborn~=0.11.2
matplotlib~=3.5.0
scikit-learn~=1.0.1
```

After this step, we will import the packages and be able to access our dataset.

1.2 Util Functions

a)

In the util function, display_confusion_matrix; we are sending the confusion matrix and message parameters, and expecting to visualize this information.

We are setting the labels such as True Negative as “TN” and the size of the graph. After, we are showing this graph with the message information which shows the related model’s confusion matrix.

b)

In the dimension_reduction_visualize function, we are visualizing our datasets's clustered shapes.

With the help of t-SNE dimension-reduction algorithm, we can easily reduce the feature number of our dataset and show the samples in a 2D graph. Firstly our t-SNE objects make the reduction tasks, after that we are visualizing the results with the scatter plots in 2x2 grid. Of course, related datasets' information is also printed.

"t-SNE is common for visualization because of the goal of the algorithm is to take your high dimensional data and represent it correctly in lower dimensions."

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

1.3 Data Describing and First Look to the Data

Now we can easily see the resulting dataframe like this:

	age	gender	polyuria	polydipsia	sudden_weight_loss	weakness
0	56	Male	0	0	1	1
1	43	Male	0	0	0	1
2	64	Male	0	0	0	1
3	35	Female	1	1	1	1
4	30	Female	1	0	1	1

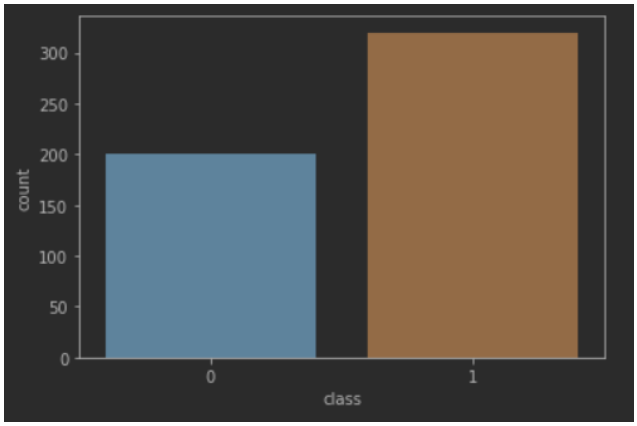
5 rows × 17 columns Open in new tab

The statistical information about our data with the help of .describe() function:

	age	polyuria	polydipsia	sudden_weight_loss	weakness	polyphagia	genital_thrush
count	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000
mean	48.028846	0.496154	0.448077	0.417308	0.586538	0.455769	0.223077
std	12.151466	0.500467	0.497776	0.493589	0.492928	0.498519	0.416710
min	16.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	39.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	47.500000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
75%	57.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
max	90.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 16 columns Open in new tab

The distribution and percentage of our classes is can be showed like this:




```
Non diabetes samples make up 38.46 % of the dataset.
Diabetes samples make up 61.54 % of the dataset.
```

2 Data Preprocessing

In this step, we will prepare and manipulate our dataset to make it ready for our classification and clustering models.

2.1 Creating Original Dataset

a) Encoding our non-numeric features

Label Encoder helps us to convert our “male-female” data points in the gender column to the binary values.

	age	gender	polyuria	polydipsia
0	56	1	0	0
1	43	1	0	0
2	64	1	0	0
3	35	0	1	1
4	30	0	1	0

5 rows × 17 columns Open in new tab

2.2 Creating Normalized Dataset

b) Normalizing our non-binary features

Feature Scaling

"Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. **Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization.** (from wikipedia)"

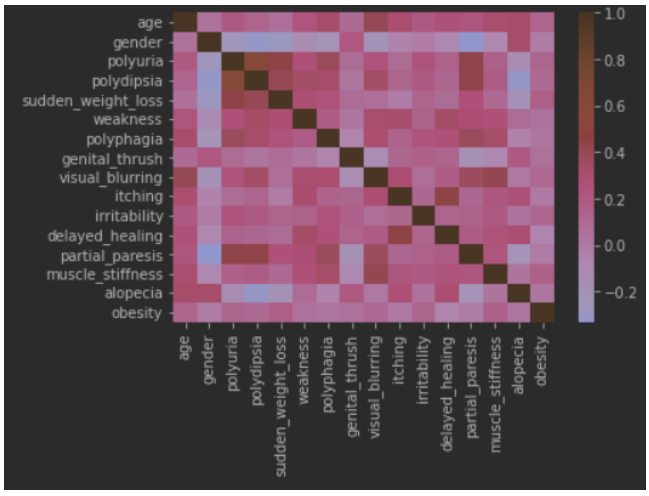
$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

We are scaling our "age" column and after this step, this feature will be scaled and squashed in [0,1] range.

	age	gender	polyuria	polydipsia
0	0.540541	1	0	0
1	0.364865	1	0	0
2	0.648649	1	0	0
3	0.256757	0	1	1
4	0.189189	0	1	0

5 rows × 17 columns Open in new tab

After these, we can also analyze our data with these contents:



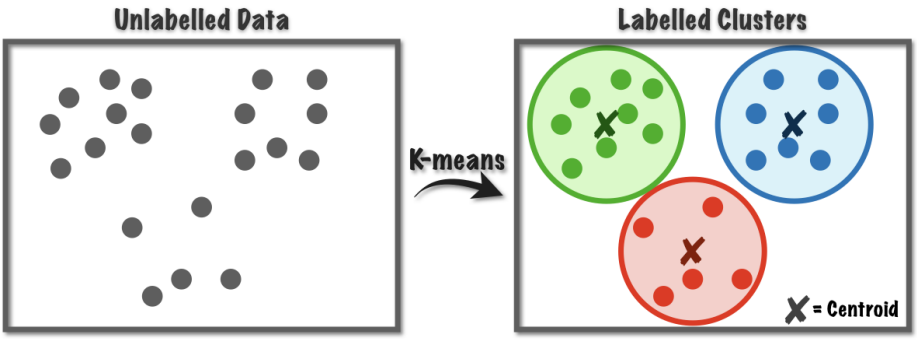
Correlation heatmap of the dataframe

	class	Shape
0	Non-Diabetes	(200, 17)
1	Diabetes	(320, 17)

Shape of the dataframe

2.3 Creating Clustered Original Dataset

"In other words, **the K-means algorithm** identifies k number of centroids, and then allocates every data point to the **nearest** cluster, while keeping the centroids as **small** as possible. The 'means' in the K-means refers to averaging of the data; that is, finding the centroid."



K-Means clustering method is usually being used for unsupervised (unlabelled data) learning problems but in this assignment, we will change our labels with the incoming result set of the clustering algorithm.

```
clustered_df["class"] =
KMeans(n_clusters=clustered_df["class"].nunique(dropna=True),
random_state=0).fit_predict(X_clustered)
```

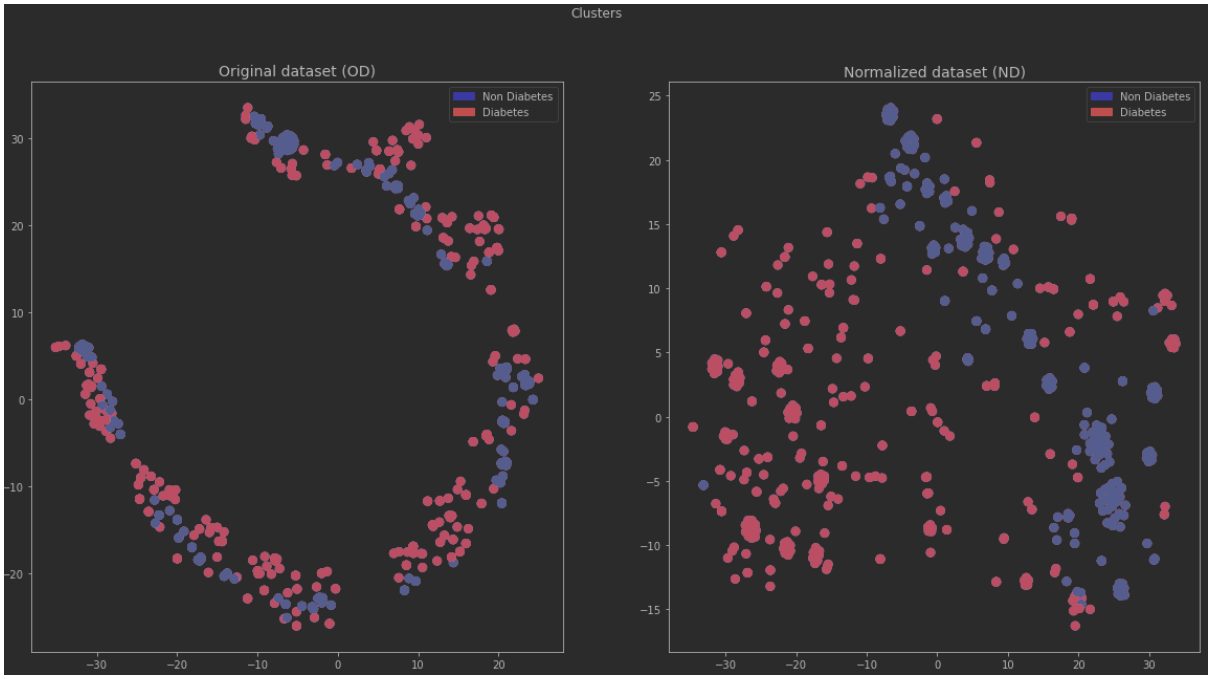
2.4 Creating Clustered Normalized Dataset

Finally, we are creating our clustered normalized dataset with using the previous data process steps.

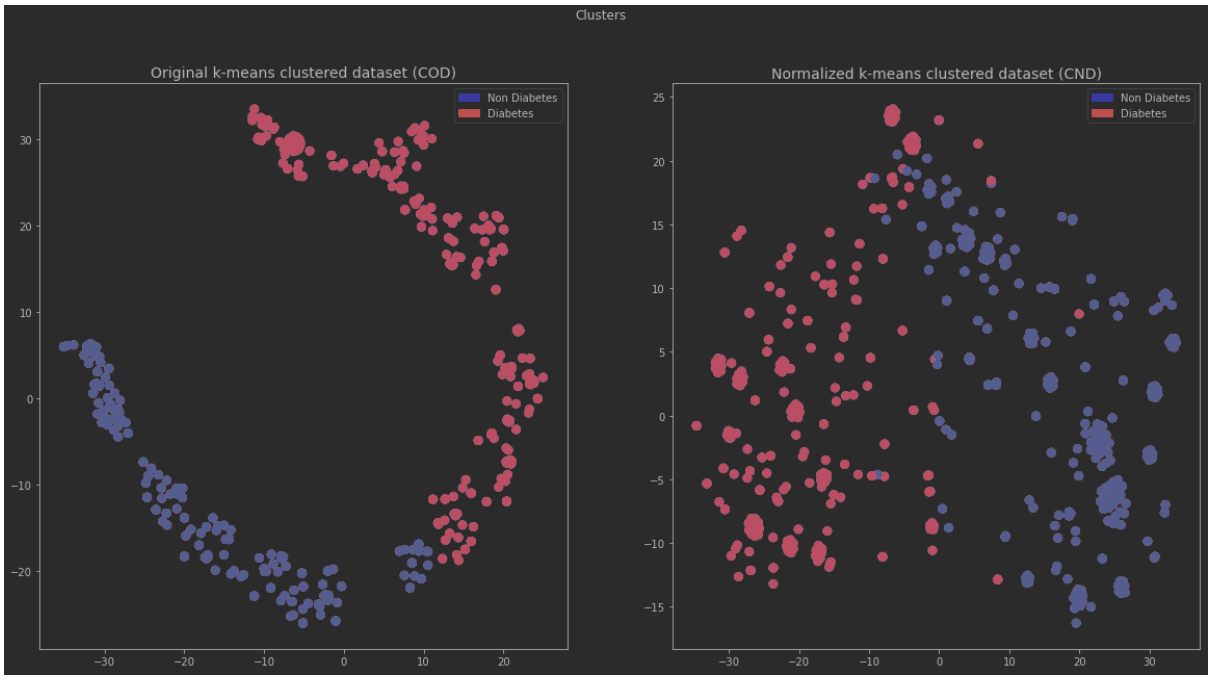
```
clustered_normalized_df = normalized_df.copy()
X_clustered_normalized = clustered_normalized_df.drop("class", axis=1)
y_clustered_normalized = clustered_normalized_df["class"]

clustered_normalized_df["class"] =
KMeans(n_clusters=clustered_normalized_df["class"].nunique(dropna=True),

random_state=0).fit_predict(X_clustered_normalized)
clustered_normalized_df.head()
```

3 Dimension Reduction - Clustering

We have talked before about the dimension reduction algorithm t-SNE in the *1.2 Util functions* section and now we can see the graphs.



Original and normalized dataset clusters



Original and normalized k-means clustered clusters

Comments on clustering:

As we can see our original non-normalized dataset has a complex distribution.

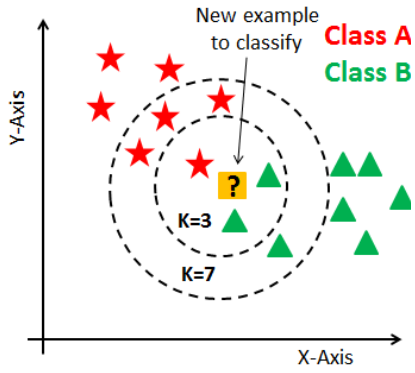
But when we normalized it, since the features of the samples have scaled, we have a different clustering results with better understandable and displayable (separable) distribution.

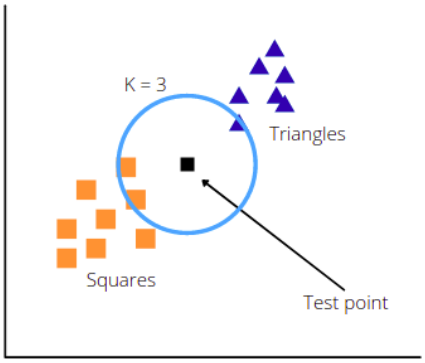
At the clustering step, the original clustered dataset has a perfect 2-class distribution as we can see. But changing labels and making them synthetic can be a problem in classification step.

And the normalized clustered dataset still has a nice distrubiton thanks to manual interfering with classes but there are some samples that visualized in the wrong cluster, maybe because of the dimension reduction algorithm for visualizing step.

4 KNN Classification

"The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before."





Since the simplicity of the implementation of KNN algorithm, we choose it for our dataset which includes small amount of example. It can be slower as the volume of our dataset increases.

K hyperparameter plays an important role:

- If we decrease the K value to 1, the predictions will be unreliable (overfitting). Because in some cases, looking the only closest neighbour value can be a problem.
- Just like this, if we increase the K value and make it too large, our majority voting technique may fail to show the real prediction (underfitting).

5 Before Modelling : k-Fold Cross Validation and GridSearchCV

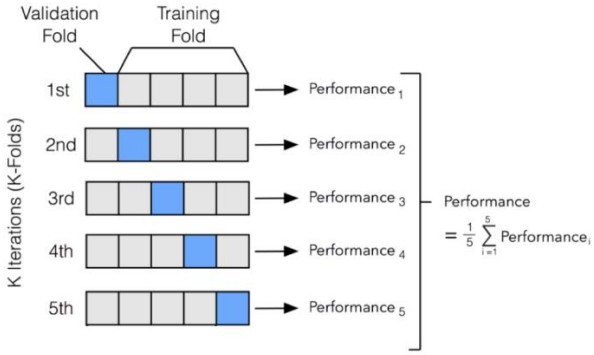
Before starting to train our models and evaluating them, we will split our datasets with the same proportion and samples.

```
# OD
x_train_df, x_test_df, Y_train_df, Y_test_df =
train_test_split(
df.loc[:, df.columns != 'class'], df.loc[:, ['class']],
test_size=0.2, random_state=1)
```

(Original dataset train-test split)

a) k-Fold Cross-Validation

“Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. 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.”



b) GridSearchCV

“It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.

GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.”

6 Model Training and Evaluation

6.1 Model Creating

For each dataset, we create different classifiers.

```
# K-Nearest Neighbour algorithm for classifying task
knn = KNeighborsClassifier()
normalized_knn = KNeighborsClassifier()
clustered_knn = KNeighborsClassifier()
clustered_normalized_knn = KNeighborsClassifier()
```

6.2 Model Optimizing

In our project, our classifiers will need to be determined 2 hyperparameters.

- K value of the k-Fold Cross Validation
- K value of the KNN Classifiers

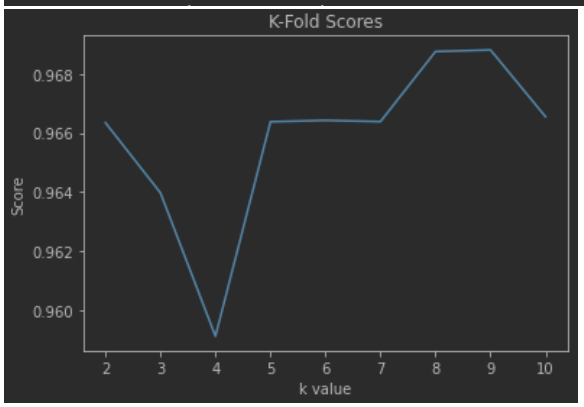
a) <u>K value of the k-Fold Cross Validation</u>

Firstly, we have optimized_prediction function for determining all these hyperparameters. This function takes the classifier object and datasets with the message values.

K value of the cross validation is calculated by the GridSearchCV's parameter. This parameter is changing by the for loop iteration. As we increase the "i" value, our KFold object's parameter is changed and so the GridSearchCV's parameter. After these iterations, GridSearchCV returns us the best K value k-Fold cross validation. We can see the resulting scores after the iteration as below.

Example from clustered normalized dataset:

```
Best k value for CV : 9 - Score : 0.9688046047898039
[0.9663461538461539, 0.9639766447711396, 0.9591346153846154, 0.9663798049340218, 0.9664251207729468, 0.9663841807909604, 0.96875, 0.9688046047898039,
0.9665505226480835]
```



We choose this dataset because the other datasets' results are similar. The best score belongs to k=9 and this GridSearchCV object is choosen for classifying.

b) <u>K value of the KNN Classifiers</u>

On each iteration of the k-Fold cross validation, GridSearchCV detects the best k value for KNN hyperparameter. For majority voting and ability for the tiebreaking, we choose the k values odd number from 3 to 31.

```
k_range = list(range(3, 31, 2))
param_grid = dict(n_neighbors=k_range)grid_search = GridSearchCV(knn,
param_grid, cv=kf, scoring='accuracy', return_train_score=True, verbose=1)
```

It can be observed like this for each classifier:

Original dataset's classifier's k value:

```
Best k value for KNN: {'n_neighbors': 3}
```

Normalized dataset's classifier's k value:

```
Best k value for KNN: {'n_neighbors': 3}
```

Clustered original dataset's classifier's k value:

```
Best k value for KNN: {'n_neighbors': 3}
```

Clustered normalized dataset's classifier's k value:

```
Best k value for KNN: {'n_neighbors': 5}
```

6.3 Model Optimizing

Resulting train/test scores, other metrics for each dataset below:

Model	OD	ND	COD	CND
Train score	91.08%	95.18%	100%	96.88%
Test score	86.54%	91.35%	99.04%	94.23%
Precision	91.94%	96.72%	100%	92.16%
Recall	86.36%	89.39%	98.31%	95.92%

Comments on classification:

Our score's on original dataset is not bad. But we should prefer normalized dataset for classification task.

Our score's on normalized dataset is very good. It seems, there is no reason not to use this classification model and data preparing steps.

Our score's on clustered original dataset is very strange. Because of changing labels according to their centroid and making prediction about their clusters/neighbours again, gives us such a result. We think we should ignore this.

Our score's on clustered normalized dataset is worse than "COD". But here, changing features and clustering them gives a different metric results. Since it is manually interferred, we should ignore this classification type too.

7 REFERENCES

1. https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

2. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

3. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

4. https://machinelearningmastery.com/k-fold-cross-validation/

5. https://medium.com/@gulcanogundur/model-seçimi-k-fold-cross-validation-4635b61f143c

6. https://www.mygreatlearning.com/blog/gridsearchcv/

7. https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec13_handout.pdf