MIDTERM ASSIGNMENT

Instructor: Dr. Selim Yılmaz (selimyilmaz@mu.edu.tr)
Out Date: 11/29/2021 14:29:59
Due Date: 12/13/2021 14:29:59

DECLARATION OF HONOR CODE[1]

Student ID .....
Name .....
Surname .....

In the course of Introduction to Machine Learning (SE3007), I take academic integrity very seriously and ask you to do as well. That's why, this page is dedicated to some clear statements that defines the policies of this assignment, and hence, will be in force. Before reading this assignment booklet, please first read the following rules to avoid any possible violation on academic integrity.

- This assignment must be done individually unless stated otherwise.
- You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, you cannot copy code (in whole or in part) of someone else, cannot share your code (in whole or in part) with someone else either.
- The previous rule also holds for the material found on the web as everything on the web has been written by someone else. Furthermore, you are welcome to seek support from generative AI chatbots like ChatGPT, Gemini, and others; however, ensure these tools do not perform the task on your behalf.
- You must not look at solution sets or program code from other years.
- You cannot share or leave your code (in whole or in part) in publicly accessible areas.
- You have to be prepared to explain the idea behind the solution of this assignment you submit.
- Finally, you must make a copy of your solution of this assignment and keep it until the end of this semester.

I have carefully read every of the statements regarding this assignment and also the related part of the official disciplinary regulations of Muğla Sıtkı Koçman University and the Council of Higher Education. By signing this document, I hereby declare that I shall abide by the rules of this assignment to prevent any violation on academic integrity.

Signature .....

[1]This page should be filled and signed by your handwriting. Make it a cover page of your report.

Task 1: Missing Value Imputation

In this task you are expected to handle missing values through *random imputation* and *regression imputation*[2].

To accomplish this task, generate a toy regression dataset on your own such that it has **five features** and at least **5,000 tuples** (note that it should not be given too much to accomplish in feasible time). In addition, there should be **three target vectors** that are dependent on the independent variables.

After you construct the dataset, **randomly select records** where missing values are to be placed. The number of data records can be chosen arbitrarily but should not be less than **2%** in size. Finally, you should choose only **one feature vector** in which some are going to be changed to the missing values.

- **Random imputation:** In this approach, you should replace missing values from a uniform distribution bounded by the feature range (i.e., imputed values should not exceed the upper and lower boundaries of this feature vector).
- **Regression imputation:** Here, you should invoke a **regression algorithm** to obtain estimated values for missing values. To do that, you can treat target vector as input vector. Besides, you can also include feature vectors as long as a correlation between these vectors and the vector containing missing values is strong. Generate a scatter chart as given in Figure 1 which shows actual and estimated 'imputed' values of the missing values. In addition, print Mean Squared Error (MSE) between actual and estimated values.
- **Save datasets:** Once you completed imputation through these two approaches, keep them in a separate datasets (including the original one) for use in the following task.

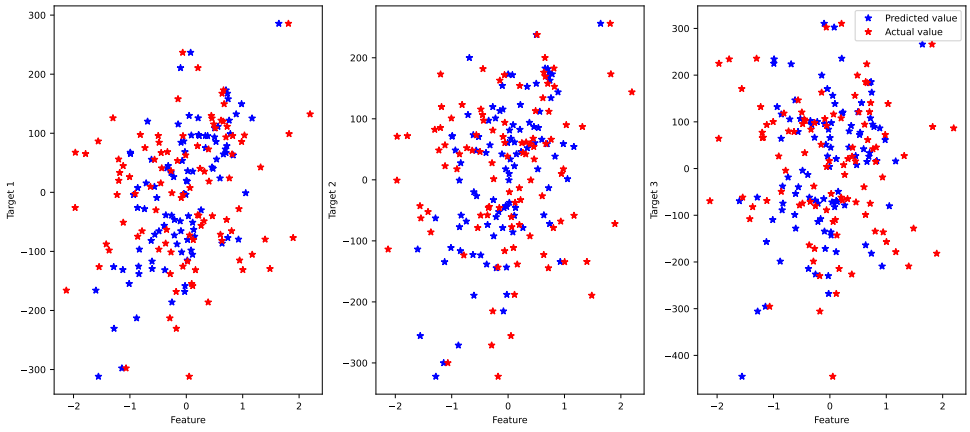

Figure 1: Missing value prediction through regression.

[2]Imputation simply means that we replace the missing values with some guessed/estimated ones.

Task 2: Train on Imputed Data

In this task you are expected to apply one-hidden layer neural network (shallow neural network) on datasets that are processed in previous task. Here, you are to fill the following table with MSE scores you observe. To do that adopt holdout method (i.e., split dataset such that 70% and 30% for training and testing purpose, respectively). Feel free to set hyper-parameter values as you wish (as long as it sounds reasonable). You should be aware that you need to use a predetermined **seed** number (preferably **42**, a rule of thumb for machine learning practitioners) for both splitting data and hyper-parameter setting, which is needed for a fair comparison. You can determine this number arbitrarily as long as it is same for every empirical environment.

Table 1: Comparative MSE scores of different datasets

Dataset	**MSE Score**[1]
Original	
Random imputation	
Regression imputation	

[1] with a three decimal precision

Task 3: Reconstruction of Images

Having a very simple network architecture, AutoEncoders (AE) are ideal for data compression (e.g., dimensionality reduction), anomaly detection, data generation, and etc. An AE with a very simple architecture is exemplified in Figure 2.

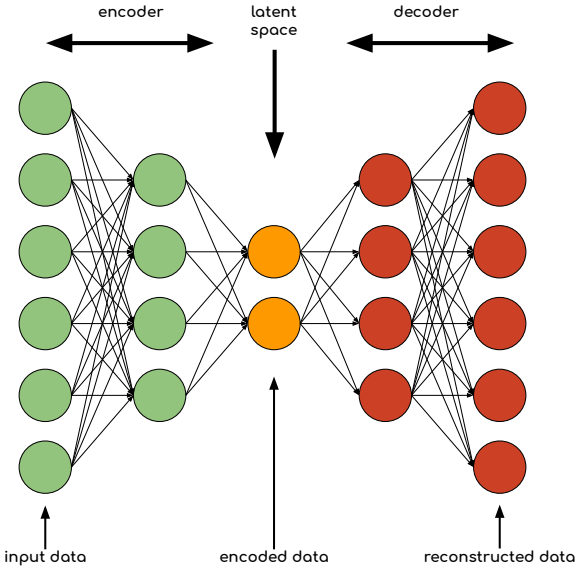

Figure 2: An autoencoder architecture.

There are two sequential steps in AE: **encoder** and **decoder**. Encoder phase takes input data to map it to a latent space. The mapped data in the latent space is regarded as compressed data. The next phase is decoder phase which is responsible for regeneration (decode) of data from the encoded data in the latent space. The loss is calculated based on the **similarity** between input and reconstructed data.

In this task you are expected to learn a model for the data generation (i.e., reconstruction of images). To do that, you will build an architecture involving Principal Component Analysis (PCA) (for encoding) followed by Multi-Layer Perceptron (MLP) (for decoding). The architecture is demonstrated in Figure 3. The important details in learning the model are outlined below.

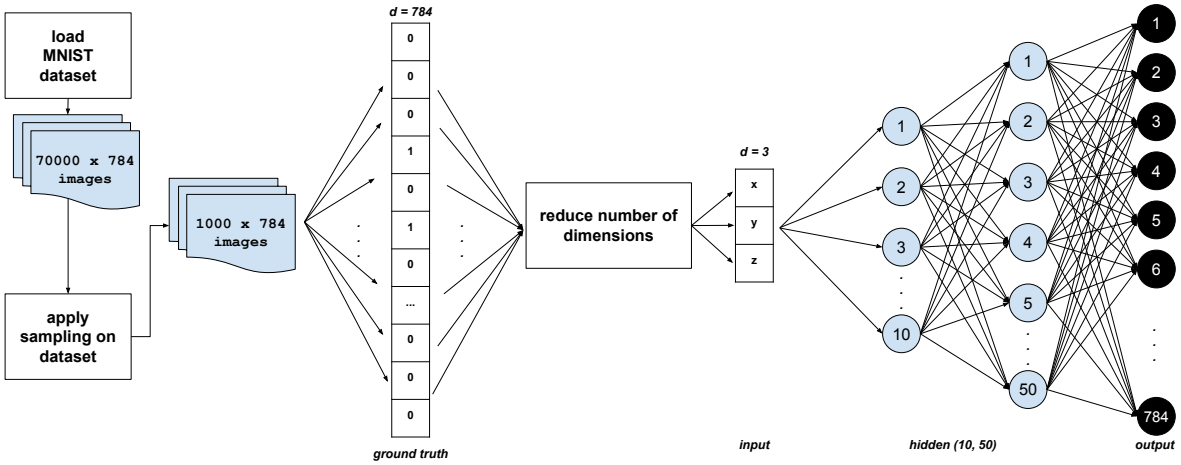

Figure 3: Framework of the task you need to complete.

- **Sampling:** Reduce number of samples to **1,000** by **maintaining the class distribution** in dataset[3].
- **Encoding:** Reduce the number of features from **784 to 3** by using PCA.
- **Deconding:** Instantiate an MLP with a hidden layer sizes of $(10, 50)$, in order. You need to train network with **three different iteration settings (e.g., number of training epochs): 10, 25, and 500**.
- **Plotting:** After you train model with three iteration settings, you need to test each digit from sampled dataset (i.e., comparatively draw original and reconstructed image from different iteration settings as shown in Figure 4).

Task 4: Cluster Sampling

In this task, you are going to make use of clustering algorithm as a means of sampling which is an important procedure on preprocessing the data.

[3]call `fetch_openml('mnist_784', version=1)`

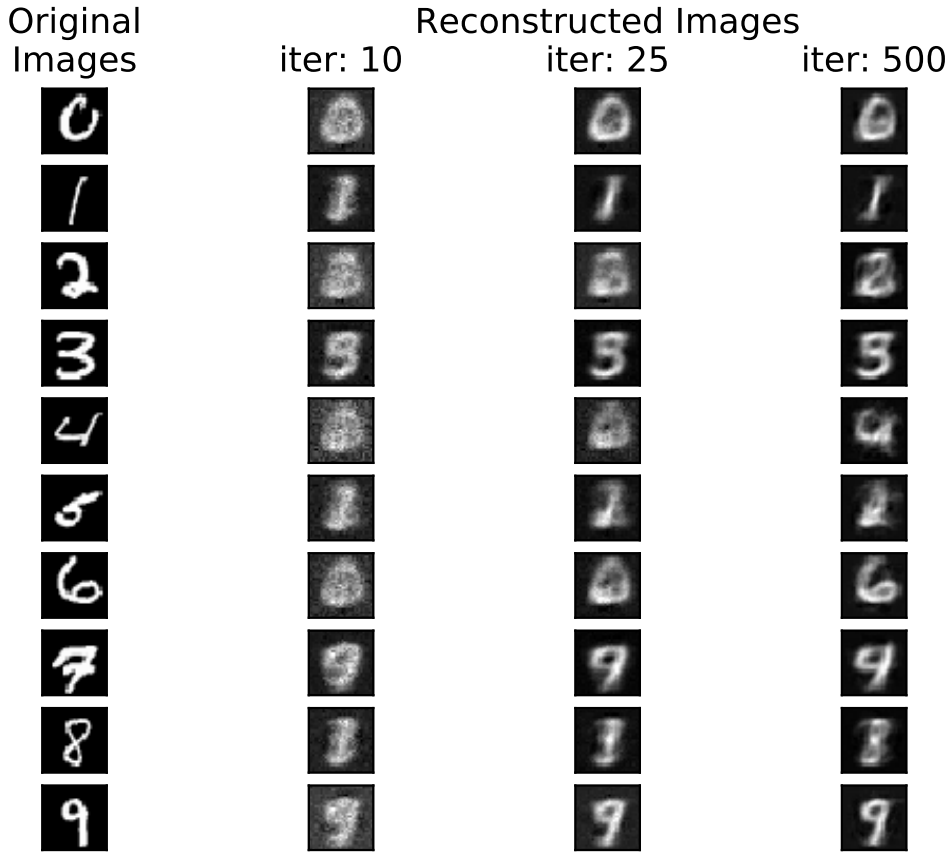

Figure 4: A comparative view of original and reconstructed images.

Cluster sampling:

This approach relies on a procedure that involves grouping a larger population into a number of categories and then selecting from them. Therefore, the objective of this strategy is to obtain only a part of all individual data records, hopefully preserving the distribution of data. There are different types of this strategy: *single-stage*, *double-stage*, and *multi-stage* sampling:

- **Single-stage sampling:** This is the simplest way for cluster sampling. Here, the whole dataset is divided into a number of clusters. The quality of the clusters and how well they represent the larger population determines the validity of your results with this sampling type. That's why, setting a large number for clusters could be ideal. After assigning data records to the clusters, some of the clusters are chosen for sampling. By default, the clusters are selected randomly in single-state sampling. However, in this assignment, you are tasked with eliminating clusters having higher density.
- **Double-stage sampling:** This strategy applies all the steps in single-stage sampling except for the final step. Here, rather than collecting data records from every selected clusters, random selection of individuals is applied within the cluster for sampling. This strategy allows a more reduced data records and hence leads to a shorter learning time

but may lead to a further degradation on the inference performance. In the case this elimination procedure continues for two or more times it is then called as **multi-stage sampling** strategy.

Single- and double-stage sampling

In this task, you are expected to implement single- and double-stage sampling. The goal of this task is **twofold**: *i*) **visualization** of single- and double-stage sampling, and *ii*) **evaluation** of these strategies on a typical classification task. Visualization of cluster sampling is exemplified in Figure 5 and the evaluation findings based on classification accuracy and training time are given thereafter. From the evaluation results, it can be seen that the classification performance on testing data slightly degrades when cluster sampling is applied but yields a shorter training time which is quite normal due to data sampling. Therefore, data scientists should consider a trade-off between performance and learning time depending on the problem they handle.

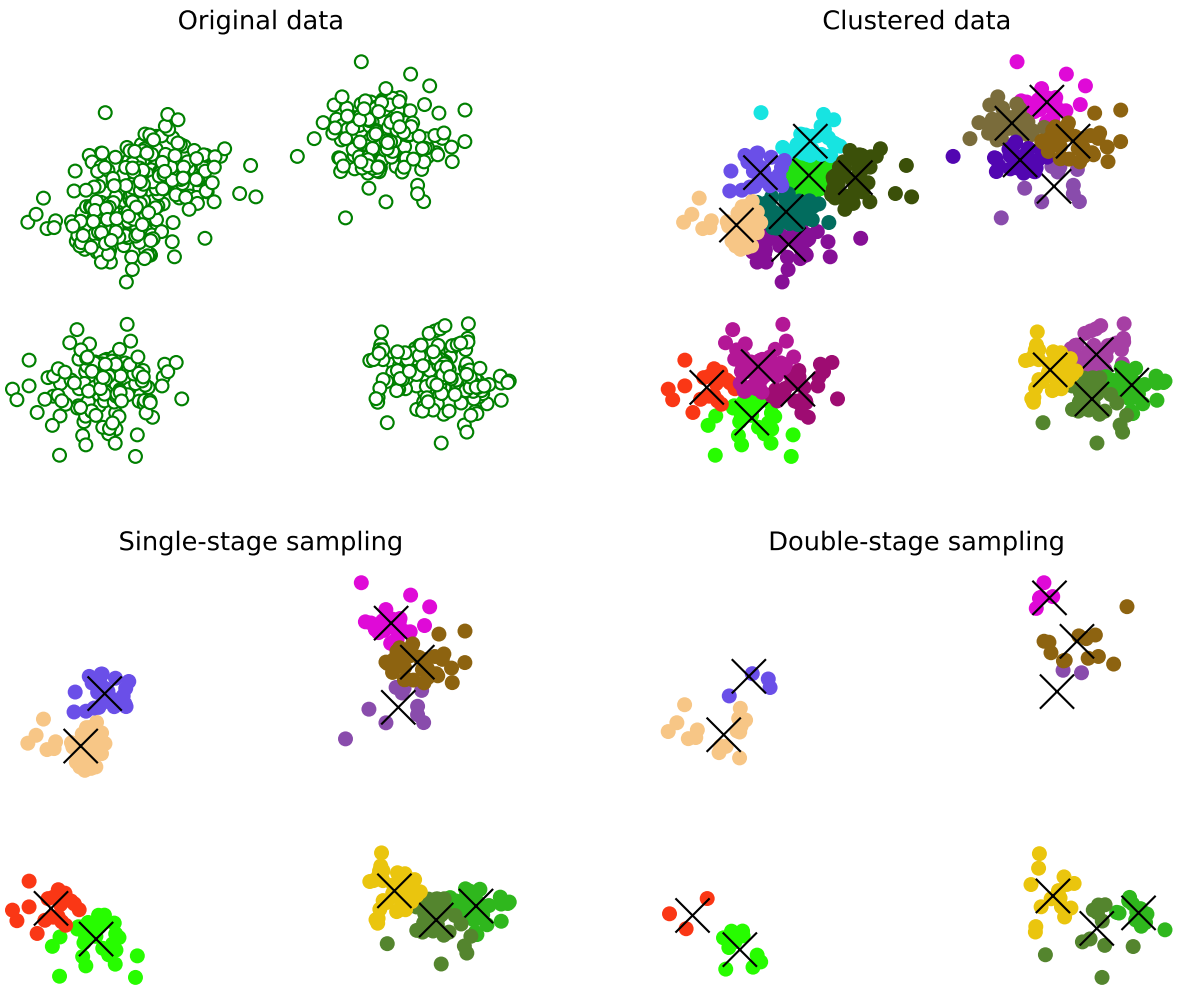

Figure 5: Cluster sampling visualization.

```
(Original Data) Mean Testing Accuracy: 0.987 Training Time: 638.899 ms
(Single-stage Clustering) Mean Testing Accuracy: 0.923 Training Time: 300.056 ms
(Double-stage Clustering) Mean Testing Accuracy: 0.903 Training Time: 167.898 ms
```

As for the visualization, it can be seen that data records that are given at the upper-left part of the figure are initially categorized into a number of clusters, which is shown at the upper-right part of the figure. Single-stage and double-stage sampling outcomes are given in order at the lower part of the figure. As seen, the **denser** the data clusters is, the more likely they are to be **eliminated** in single-stage sampling so that remaining data records are a mini representation of whole data. As for double-stage clustering some of the individual objects from the chosen clusters are **randomly** chosen. To accomplish that please follow the procedure with a high care:

1. Load a toy, **ready-to-use** classification dataset (D), or generate on your own. Be aware that there are **more than five classes** where data records belong to.
2. Split D such that randomly selected **70%** tuples are used for **training** (say $train_{orig}$) while **30%** tuples are used for **testing**.
3. Generate 2×2-axis figure and show all the **training** data records in the **upper-left** window.
4. Apply **K-means** algorithm on **training data** with a large number of clusters[4] (it is 20 in Figure 5) and then show the cluster centroids as well as cluster-assigned training data records in **upper-right** window.
5. Determine number of clusters (R) to be included for sampling[4] (it is 10 in Figure 5).
6. **Select clusters** in which data records are to be included for sampling. Note that, dense clusters are prioritized for elimination. To do that, you can follow the steps below:
   a) Include **all** clusters C.
   b) Calculate the **density of every cluster** ($c \mid c \in C$) by equation below[5].

$$density_c = \frac{1}{\sum_{n \in N(c)} d(c, n)} \tag{1}$$

   where $N(c)$ is a set of neighbor clusters of c, and $d(\cdot)$ is a metric, such as Euclidean, to calculate distance between c and n. The number of neighbors should be determined first[4]. It is set 5 in Figure 5.

   c) Remove the cluster having highest density from C then go to step b until R clusters remain in C.
7. Once you reach this step, you must have obtained single-stage sampled data records (say $train_{ss}$). Visualize $train_{ss}$ at the lower-left part of the figure.
8. To obtain double-stage sampled data records (say $train_{ds}$), randomly choose data records from $train_{ss}$. Visualize $train_{ds}$ at the lower-right part of the figure.

[4] feel free to give any number.
[5] you may refer to `kneighbors_graph` in `sklearn` as helper method.

9. Generate an instance of Multi Layer Perceptron (MLP) classifier with an hyper-parameter setting[4].
10. Train MLP with same settings on $train_{orig}$, $train_{ss}$, and $train_{ds}$. Also record training time separately taken by every training phase.
11. Evaluate the performance of models learned on three different datasets through test data that is generated at step 2.
12. Print your findings in a format that exactly matches with given above.

Task 5: Novelty Detection

In this task you are expected to handle spam detection problem on a number of SMS (short message service) texts. To do that you need to follow the following procedure:

1. Download dataset containing spam and non-spam (i.e., ham) messages[6]. The format of dataset is given below. As seen, every line represents a separate text message and starts with the class label that the message belongs to. Here 'ham' stands for non-spam messages, whereas 'spam' (as the name suggests) stands for the malicious/spam messages. The class and the content of messages are separated by tabulator (**TAB**) keys.

```
ham         Have a safe trip to Nigeria. Wish you happiness and very soon company to
            share moments with
ham         Cool, what time you think you can get here?
spam        This is the 2nd time we have tried 2 contact u. U have won the £750 Pound
            prize. 2 claim is easy, call 087187272008 NOW1! Only 10p per minute.
            BT-national-rate.
```

2. Apply **bag of words** approach to generate a feature space model of the message contents. Bag of words is a representation of text that describes the occurrence of words within a document. Both unigram (each separate word) and bigram (two-word pairs) models must be used to evaluate occurrences of words. Instead of counts of every occurrences, tf-idf (term-frequency times inverse document-frequency) metric. Because stop words like 'the', 'an', 'and' may not contain as much "informational content" to the model even though they frequently appear in documents.
3. After you generate feature space model of text content, you should apply PCA to reduce number of dimensions to 100.
4. The final step is model training. For this purpose, you should generate training and testing datasets. Testing dataset should contain all the malicious messages (i.e., spam messages) plus randomly selected 100 non-spam messages. Training dataset, however, should contain all the non-spam messages expect for 100 tuples taken to the testing purpose. Apply any novelty detection approach for the model generation. After then, calculate true positive (TP), false positive (FP), true negative (TN), false negative (FN) metrics through testing process. Finally, show "accuracy" performance of the model.

[6]download from **https://archive.ics.uci.edu/ml/datasets/sms+spam+collection**

Table 2: Performance metric obtained from novelty detection.

Metric	**Value**
TP:	
TN:	
FP:	
FN:	
Accuracy:	

Notes

- Your source code should be designed as **easy-to-follow**. **Place comment** in it as much as possible. **Separate each task** through apparent patterns.
- Use LaTeX to prepare your reports. Include the observation tables here to your report. Once again, filled and signed declaration form should be first page of your report. **Reports must not exceed 5 pages in total**.
- **Do not miss** the deadline. **Save your work** until the beginning of next semester.
- The assignment must be **original**, **individual work**. **Duplicate or very similar assignments are both going to be considered as cheating.**
- You will submit your work on SE3007 course page at https://dys.mu.edu.tr with the file hierarchy as below[7]:

→ <student id>.zip
  → Task1_2.py
  → Task3.py
  → Task4.py
  → Task5.py
  → MidtermReport.pdf

[7] do not place any file into a directory. Just compress all the files together.