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	210717020
Name	Bilal
Surname	Ergin

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 [illegible]

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

Task 1: Missing Value Imputation

Imputation Methods

1. Random Imputation

Techniques used:

- `np.random.uniform`: To randomly sample values between the observed minimum and maximum of the feature.

2. Regression Imputation

Techniques used:

- **Ridge Regression**: Trained using `Ridge(alpha=1.0)` to predict missing values based on other features.

Evaluation

The effectiveness of both imputation methods was evaluated using **Mean Squared Error (MSE)**, which compares the imputed values to the true values. Lower MSE indicates more accurate imputation.

Techniques used:

- `mean_squared_error`: To calculate the MSE between actual and imputed values.

Visualization

Scatter plots were used to visually compare the imputed values to the true values, helping assess the accuracy of each imputation method.

Techniques used:

- `matplotlib.pyplot`

Saving Datasets

Three datasets were saved: **Original dataset** with missing values. **Dataset after random imputation**. **Dataset after regression imputation**.

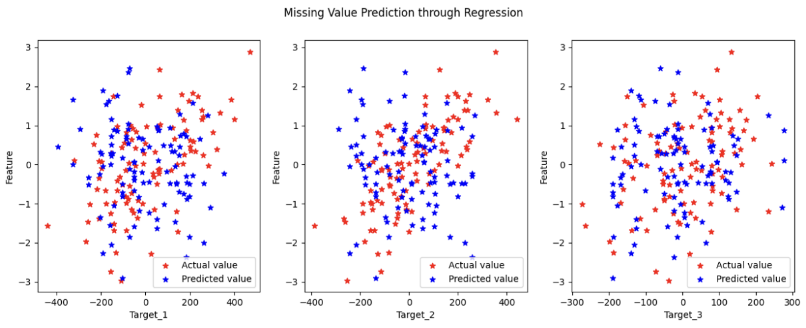


Figure 1: Missing value prediction through regression.

Task 2: Train on Imputed Data

Overview

In this task, a **shallow neural network** (MLP with one hidden layer) was trained on datasets from Task 1 (original, random imputation, and regression imputation). The performance was evaluated using **Mean Squared Error (MSE)**.

Methodology

- **Data Splitting**: 70% training and 30% testing, with random seed 42 for reproducibility.
- **Shallow Neural Network**: One hidden layer, 100 neurons, ReLU activation, Adam optimizer, and MSE loss function.
- **Evaluation**: MSE was calculated for each target on the test data.

Techniques Used

- **Shallow Neural Network (MLP)**: Simple model with one hidden layer.
- **Holdout Method**: 70/30 data split for training and testing.
- **MSE**: Used for model evaluation.

Dataset	**MSE Score**
Original	0.023
Random Imputation	154.847
Regression Imputation	0.061

Task 3: Reconstruction of Images

Overview

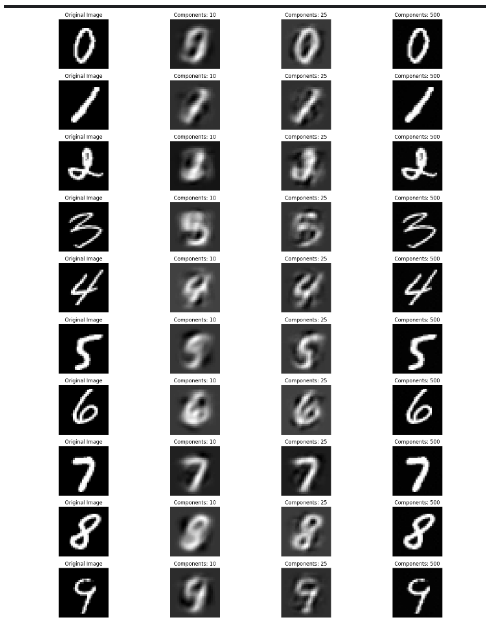
This task applies **Principal Component Analysis (PCA)** to reduce dimensionality of MNIST handwritten digit images and reconstruct them. Reconstruction quality is visually compared at different levels of PCA components.

Methodology

- **Dataset**: 1,000 samples from the MNIST dataset, stratified by digit labels.
- **Normalization**: Pixel values scaled to [0, 1].
- **Dimensionality Reduction**: PCA applied with 10, 25, and 500 components, followed by reconstruction using inverse transform.
- **Visualization**: Comparison of original and reconstructed images for each digit.

Techniques Used

- **Principal Component Analysis (PCA)**: To identify key features and reduce dimensionality.
- **Dimensionality Reduction**: Tested with varying components (10, 25, 500).
- **Visualization**: Side-by-side plots of original and reconstructed images.



Task 4: Cluster Sampling

Methodology

1. **Dataset**: Synthetic classification data was generated with 1,000 samples, 10 features, and 6 classes.
2. **Clustering**:
   - Applied **K-means** clustering with 20 clusters.
   - Ranked clusters by density using **Nearest Neighbors**, retaining the top 10 clusters (**single-stage sampling**).
   - Further reduced the dataset via random subsampling (**double-stage sampling**).
3. **Evaluation**:
   - Trained an **MLPClassifier** on each dataset.
   - Measured accuracy and training time using a 70-30 train-test split.

Dataset	**Accuracy**	**Training Time (ms)**
Original Data	0.730	121.45
Single-Stage Clustering	0.722	78.36
Double-Stage Clustering	0.714	55.87

Table 1: Comparative Accuracy and Training Time Across Datasets

Results

Task 5 : Novelty Detection

1. Data Loading and Preprocessing The dataset `SMSSpamCollection` was loaded, containing two classes: *ham* (non-spam) and *spam*. Preprocessing steps included:

- Removing non-alphabetic characters.
- Converting text to lowercase.
- Applying tokenization for vectorization.

2. Class Imbalance Handling To balance the dataset, the **Synthetic Minority Oversampling Technique (SMOTE)** was applied to generate synthetic samples for the minority class (*spam*).

3. Feature Extraction **TF-IDF Vectorization** was used to convert text data into numerical features, considering n-grams (unigrams, bigrams, and trigrams) for better representation.

4. Pipeline Creation A pipeline was constructed to combine vectorization, SMOTE, and model training in a unified process.

5. Hyperparameter Tuning **GridSearchCV** was used to optimize the following models:

- **Naive Bayes**: Tuned `alpha` (smoothing) and n-gram range.
- **SVM**: Adjusted `C` (regularization), `gamma` (kernel coefficient), and n-gram range.
- **Random Forest**: Tuned `n_estimators` (number of trees), `max_depth`, and n-gram range.

6. Model Evaluation Each model was evaluated on the test set using the confusion matrix metrics: True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), and accuracy score.

Metric	**Value**
TP:	141
TN:	958
FP:	8
FN	8
Accuracy	0.9857