Lab Assignment 2: Regression and Classification

ENGS 106 / COSC 179: Principles of Machine Learning

Overview

In this lab, you will implement three fundamental machine learning algorithms from scratch:

- **Part A: Linear Regression** – Predict wine quality features using least-squares regression
- **Part B: k-Nearest Neighbor (k-NN)** – Classify data using the k-NN algorithm
- **Part C: Naive Bayes** – Spam detection using the Naive Bayes classifier

No high-level machine learning libraries are allowed – we are doing this from first principles!

Allowed Libraries

- numpy
- matplotlib
- pandas (for data loading only)

---

1 Part A: Linear Regression (30 points)

1.1 Problem Statement

A dataset is included related to red and white vinho verde wine samples from the north of Portugal. The goal of the dataset is to model wine quality based on physicochemical tests (see Cortez et al., 2009). In this exercise, we look at a subset of the data and try to **predict wine's citric acid level based on other features**.

1.2 Dataset Description

The dataset is related to red and white variants of the Portuguese "Vinho Verde" wine. Input variables (based on physicochemical tests):

1. fixed acidity
2. volatile acidity
3. **citric acid** (target variable)
4. residual sugar
5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulphates
11. alcohol
12. quality (score between 0 and 10)

1.3 Tasks

1. **(10 points)** Implement linear regression from scratch using the least-squares method. You may use `np.linalg.lstsq()` or implement the normal equation yourself.
2. **(15 points)** Start with 'alcohol' and 'density' as the first two features. Find a third feature that would improve the prediction the most. Justify your selection.
3. **(15 points)** Find the 4th feature. What would happen if you include all the features? Show your analysis.
4. **(10 points)** Provide plots comparing model predictions vs actual values for different feature combinations.

---

2 Part B: k-Nearest Neighbor Classification (35 points)

2.1 Problem Statement

For this problem, you will implement the k-Nearest Neighbor (k-NN) classifier and evaluate it on the `Lenses` and `Credit Approval` (CA) datasets. The CA dataset describes credit worthiness data (binary classification).

2.2 Datasets

- **Lenses Dataset**: A small dataset for contact lens prescription
- **Credit Approval Dataset**: Credit card application data with binary labels (+/-)

The data has been split into training and testing sets:

- `lenses.training`, `lenses.testing`
- `crx.data.training`, `crx.data.testing`

2.3 Data Preprocessing

The CA dataset contains missing values (indicated by '?') and both numerical and categorical features.

1. **Missing Value Imputation**:
   - For categorical features: replace missing values with the median/mode value
   - For numerical features: replace missing values with the label-conditioned mean (i.e., $\mu(x_i|+)$ for positive instances)
2. **Feature Normalization**: Use z-scaling for real-valued features:

$$z_i^{(m)} \leftarrow \frac{x_i^{(m)} - \mu_i}{\sigma_i} \tag{1}$$

   where μ_i is the mean and σ_i is the standard deviation of feature i.

2.4 Tasks

1. **(10 points)** Implement data preprocessing:
   - Handle missing values using mean/median imputation
   - Normalize numerical features using z-scaling
   - Document exactly how you imputed missing values for each feature
2. **(20 points)** Implement k-NN with L2 distance:

$$\mathcal{D}_{L2}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_i (a_i - b_i)^2} \tag{2}$$

   For categorical attributes, use a distance of 1 if values disagree and 0 if they agree.
3. **(15 points)** Generate a table reporting accuracy on both datasets for at least three different values of k (e.g., $k = 1, 3, 5, 7$).
4. **(5 points)** Discuss your results: Which value of k works best? Why?

---

3 Part C: Naive Bayes for Spam Detection (35 points)

3.1 Problem Statement

For this problem, you will implement a Naive Bayes classifier for spam email detection. You will use the **Spambase** dataset from the UCI Machine Learning Repository, which contains features extracted from emails to classify them as spam or not spam.

3.2 Dataset Description

The Spambase dataset contains 4601 email samples with 57 features:

- **Features 1-48**: Word frequencies (percentage of words in the email that match a specific word)
- **Features 49-54**: Character frequencies (percentage of characters that match specific characters like `;`, `(`, `[`, `!`, `$`, `#`)
- **Features 55-57**: Capital letter statistics (average length of uninterrupted sequences, longest sequence, total number)
- **Label**: 1 = spam, 0 = not spam

The dataset is available through the `ucimlrepo` library (dataset id=94).

3.3 Naive Bayes Background

The Naive Bayes classifier is based on Bayes' theorem with the "naive" assumption of conditional independence between features:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \propto P(C) \prod_{i=1}^{n} P(x_i|C) \tag{3}$$

For continuous features, we typically assume a Gaussian distribution:

$$P(x_i|C) = \frac{1}{\sqrt{2\pi\sigma_{i,C}^2}} \exp\left(-\frac{(x_i - \mu_{i,C})^2}{2\sigma_{i,C}^2}\right) \tag{4}$$

3.4 Tasks

1. **(10 points)** Implement a Gaussian Naive Bayes classifier from scratch:
    - Estimate class priors $P(C)$ from training data
    - Estimate mean $\mu_{i,C}$ and variance $\sigma_{i,C}^2$ for each feature per class
    - Implement the prediction function using log-probabilities to avoid numerical underflow

2. **(10 points)** Train and evaluate your classifier:
    - Split the data into training (80%) and testing (20%) sets
    - Report accuracy, precision, recall, and F1-score
    - Create a confusion matrix
3. **(10 points)** Feature analysis:
    - Identify the top 5 most discriminative features for spam detection
    - Visualize the distribution of these features for spam vs non-spam emails
4. **(5 points)** Discussion:
    - Why is Naive Bayes effective for spam detection despite the independence assumption?
    - What are the limitations of your implementation?

What to Submit

Submit a GitHub repository link containing:

1. Your completed Jupyter notebooks for Part A, Part B, and Part C
2. A brief post-lab write-up (PDF or Markdown) containing:
    - Your paper design / approach for each part
    - A brief description of your models and parameter selections
    - Evaluation results with plots/tables
    - A brief reflection on your experience

Remember: Add your assigned Lab TA as a collaborator to your GitHub repository!

Evaluation

You will be evaluated on:

- Code correctness and readability
- Quality of analysis and justification of design decisions
- Clear documentation and communication