Instructor: Alina Vereshchaka

Assignment 1

Classification and Regression Methods

Checkpoint: Sep 26, Thu, 11:59 pm

Due Date: Oct 10, Thu, 11:59 pm

Our Assignment 1 is focused on learning how to build classification and regression models from scratch. In the first part you will perform data preprocessing. In the second part, you will implement logistic regression and test it on a penguin dataset. In the third part, you will implement linear regression using ordinary least squares (OLS) and in the last part, you will extend it to ridge regression. There are also bonus tasks available that you can consider completing.

The goal of this assignment is to give you hands-on experience with building classification and regression models and apply them to solve tasks based on the real-world datasets.

Note: For this assignment, *scikit-learn* or any other libraries with in-built functions that help to implement ML methods cannot be used. Submissions with used ML libraries (e.g. scikit-learn) will not be evaluated.

Part I: Data Analysis & Preprocessing [10 points]

In this section, we will focus on data analysis and preprocessing.

DATASETS:

Choose THREE datasets from the 'noisy_datasets' folder provided at UBlearns:

1. Penguins Dataset: available at 'noisy_datasets' folder. This dataset is slightly different from the one provided in A0.
2. Two more datasets: select from the 'noisy_datasets' folder

TASK:

1. Import required libraries (e.g. pandas or numpy, not allowed: *scikit-learn* or other libraries with in-built functions that help to implement ML methods).
2. Read, preprocess, and print the main statistics about the dataset. You can reuse your code from A0 with a proper citation.
3. Handle missing entries. Possible solutions:
    - Drop rows with missing entries. If you have a large dataset and only a few missing

features, it may be acceptable to drop the rows containing missing values.

- Impute missing data. Replace the missing entries with the mean/median/mode of the feature. You can use the k-nearest neighbor algorithm to find the matching sample.

4. Handle mismatched string formats.

   For example, in the penguins dataset, the "Species" feature might appear as "Adelie" or "adelie," both of which refer to the same penguin species. These variations should be standardized to a consistent format such as "Adelie" or "adelie".

5. Handle outliers. Detect and manage outliers within the dataset.

   For example, in the penguins dataset, while flipper lengths typically fall within the range of [180 – 210], certain entries might show values like [10-30]. These can be considered outliers. Possible solutions:

   - Remove outliers. If there are just a few outliers, you may eliminate the rows containing these outliers.
   - Impute outliers. Replace the outliers with the mean/median/mode of the feature.

6. Using any data visualization library (e.g. matplotlib, seaborn, plotly), provide at least 5 visualization graphs related to your dataset. You can utilize any columns or a combination of columns in your dataset to generate graphs. E.g. correlation matrix, features vs. the target, counts of categorical features vs. the target.

7. Identify uncorrelated or unrelated features.

   Note: Unrelated or uncorrelated features can introduce confusion to your model and negatively impact its performance. You can compute the correlation matrix between the features and the target variable. Features with a low correlation coefficient should be identified and subsequently dropped from the dataset to enhance model performance.

   To do this, you first choose a suitable binary target for the dataset. E.g.: you can predict the gender of a penguin (female or male). In this case, the "gender" column would be your target (Y).

   As another example, you can predict whether a penguin's location is Torgersen Island or not, using the "island" column as your target (Y).

   To compute the correlation matrix, load your dataset using Pandas and use the pandas.DataFrame.corr method. You can refer to the documentation here:

   https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

   You'll need to decide which features to drop based on their correlation with the target.

8. Convert features with string datatype to categorical (e.g., species, island). Possible ways:

   - One-hot encoding, creating binary columns for each category, denoting their presence or absence. E.g., in the "Species" feature, "Adelie," "Chinstrap," and "Gentoo" become binary columns with "1" for presence and "0" for absence.
   - Label encoding assigns unique integers to distinct feature values, useful for ordinal

relationships among categories. E.g., "Small" as 0, "Medium" as 1, and "Large" as 2 can represent a "Size" feature. However, it may introduce unintended patterns.

9. Normalize non-categorical features (e.g. bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g).
    a. Find the min and max values for each column.
    b. Rescale dataset columns to the range from 0 to 1

    Why do we do this? Normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.

    Note: normalize() is not allowed as it is a part of *scikit-learn* library.

10. Select two more datasets and complete Steps 1-9.
    In total, you need to provide the data analysis and preprocessing for THREE datasets: *'penguins'* and any two of your choice from the provided folder *'noisy_datasets'.*

Note for Part I: you don't have to follow a particular order for the data preprocessing, e.g. you can do Step 5 before Step 4.

In your report for Part I:

1. Provide short details of the methods you used for data preprocessing.
2. For each dataset (three in total):
    a. Provide a brief overview of your dataset. Describe its domain, type of data, number of samples, and features.
    b. Present key statistics (e.g. mean, standard deviation, number of missing values for each feature)
    c. Include at least 5 graphs, such as histograms, scatter plots, or correlation matrices. Briefly describe the insights gained from these visualizations.

Part II: Logistic Regression using Gradient Descent [35 points]

In this part, we will work on logistic regression and will use a logistic function to model a binomial (Binary / Bernoulli) output variable. The logistic regression model predicts that the observation belongs to a particular category. To generate these probabilities, logistic regression uses the sigmoid function that maps a real number to a value between 0 and 1.

DATASET

Penguins Dataset: Use the preprocessed dataset from Part I.

STEPS

1. Import required libraries (not allowed: *scikit-learn* or other libraries with in-built functions that help to implement ML methods).
2. Choose your target Y. For this dataset, there are several options:
   - Predict which gender a penguin belongs to (female or male). In this case, column 'gender' can be used as Y (target)
   - Predict if a penguin's location is Torgersen Island or not. In this case, column 'island' can be used as Y (target)
3. Create the data matrices for X (input) and Y (target) in a shape $\mathrm{X} = N \times d$ and $\mathrm{Y} = N \times 1$, were N is a number of data samples and d has a number of features.
4. Divide the dataset into training and test, as 80% training and 20% testing dataset.
5. Print the shape of your X_train, y_train, X_test, y_test
6. Recommended structure of your code to define logistic regression:

```
class LogitRegression()

  def __init__()
     # Takes as an input hyperparameters: learning rate and the number
     of iterations.

  def sigmoid():
     # Define a sigmoid function as
```

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

```
  def cost():
     # Loss function for Logistic Regression can be defined as
```

$$\mathrm{h} = \sigma(w^T x + b) = \left(\frac{1}{1+e^{-(w^T x+b)}}\right)$$

$$J(w) = \frac{1}{N}(-y * \log(h) - (1-y) * \log(1-h)$$

```
  def gradient_descent():
     # Define current prediction y_hat for logistic regression as
```

$$\hat{y} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{X} + \mathrm{b})$$

```
     # Gradient descent is just the derivative of the loss function
     with respect to its weights. Thus:
```

$$\frac{\partial J(w)}{\partial w} = \frac{1}{N} X^T(\sigma(w^T x + b) - y)$$

```
     To implement this formula, you can use intermediate variables,
     e.g.
          pred = σ(w^T x + b)
          delta = pred – y_train
          dW = (X^T*delta)/N
```

```
Update rule: w = w − α ∇J(w), where α is a learning rate


def fit():
    # This method performs the training.
    # Initialize weights
    # For a number of iterations
        # Call gradient_descent function
        # Call cost function and keep it in an array, e.g.
        loss.append()

def predict(self, X):
    # Return the predicted result in the binary form
```

$$\hat{y} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{X} + \mathrm{b}) = \begin{cases} +1 & \text{if } \sigma(z) \geq 0.5 \\ 0 & \text{if } \sigma(z) < 0.5 \end{cases}$$

7. Train the model:
   - Define a model by calling LogitRegression class and passing hyperparameters, e.g.

   ```
   model = LogitRegression(learning_rate, iterations)
   ```

   - Train the model, by calling fit function and passing your training dataset, e.g.

   ```
   model.fit(X_train, y_train)
   ```

   - Try at least THREE various hyperparameters. You can try different learning rates and number of iterations to improve your accuracy (accuracy of greater than 64% is expected). Suggested hyperparameters:

   ```
   learning_rate=1e-3

   iterations=100000

   weights = np.random.uniform(0, 1)
   ```

8. Save the weights of the model, that returns the highest accuracy as pickle file. You will need to submit it along with your other files. Check more details about Pickle
9. Make a prediction on test dataset by counting how many correct/incorrect predictions your model makes and print your accuracy.
   Accepted accuracy rate based on the test dataset for penguin dataset is above 64%
10. Plot the loss graph and print out the loss values over each iteration.

In your report for Part II:

1. Provide your best accuracy.
2. Include a loss graph and provide a short analysis of the results.

3. Provide at least 3 different setups with learning rate and #iterations and discuss the results along with plotting of graphs. Plot the graph to discuss impact and loss over the iterations. Explain how hyperparameters influence the accuracy of the model.
4. Discuss the benefits/drawbacks of using a Logistic Regression model.

Checkpoint Submission (Part I and Part II):

1. Report:
    - Submit as a PDF file: TEAMMATE1_TEAMMATE2_assignment1_report.pdf
      E.g. avereshc_ pinazmoh_ assignment1_report.pdf
2. Code:
    - Submit Jupyter Notebook files with saved outputs:
        - TEAMMATE1_TEAMMATE2_assignment1_part_1.ipynb
        - TEAMMATE1_TEAMMATE2_assignment1_part_2.ipynb
          e.g., avereshc_ pinazmoh_ assignment1_ part_1.ipynb
3. Pickle Files:
    - Save and submit model weights that generate the best results for your model
      TEAMMATE1_TEAMMATE2_assignment1_part2.pickle
4. Preprocessed Datasets:
    - Submit preprocessed dataset files as CSV: DATASET_preprocessed.csv
      E.g. penguins_preprocessed.csv
5. Combine all files in a single zip folder that will be submitted on UBlearns, named as TEAMMATE1_TEAMMATE2_assignment1_checkpoint.zip

Part III: Linear & Ridge Regressions using OLS [25 points]

In this section, you will implement both linear regression and ridge regression using the ordinary least squares (OLS) method and apply these models to real-world data. The goal is to understand and compare the performance of these regression techniques.

Implement linear regression using the ordinary least squares (OLS) method to perform direct minimization of the squared loss function.

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - w^T x_i)^2$$

In matrix-vector notation, the loss function can be written as:

$$J(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})$$

where $\boldsymbol{X}$ is the input data matrix, $\boldsymbol{y}$ is the target vector, and $\boldsymbol{w}$ is the weight vector for regression.

DATASET

Use any dataset from the 'noisy_datasets' folder, excluding the 'penguins' dataset. You can use a dataset previously preprocessed in Part I or choose a new one.

STEPS

1. Import required libraries (not allowed: *scikit-learn* or other libraries with in-built functions that help to implement ML).
2. Data analysis and preprocessing.
   Reuse your preprocessing code from Part I or apply similar preprocessing steps to the new dataset. Ensure normalization of features is done after splitting the data into training and testing sets.
3. Data preparation
   a. Define the target variable Y
   b. Create matrices for input X and target Y with shape $\mathrm{X} = N \times d$ and $\mathrm{Y} = N \times 1$, where N is a number of data samples and d is a number of features.
   c. Split the dataset into training (80%) and testing (20%) sets.
   d. Print the shapes of X_train, y_train, X_test, y_test.
4. **Linear Regression using OLS:**
   a. Calculate the weights with the OLS equation:
   $$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$
   b. Calculate predictions and compute the Mean Squared Error (MSE):
   $$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T x_i)^2$$
   c. Plot the predictions vs the actual targets.
5. **Ridge Regression using OLS:**
   a. Implement Ridge regression by minimizing the regularized squared loss:
   $$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \frac{1}{2}\lambda\boldsymbol{w}^T\boldsymbol{w}$$
   b. Compute the weights using the ridge regression OLS equation:
   $$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\lambda I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$
   c. Plot the predictions vs the actual targets.

In your report for Part III:

1. Provide a brief description of the dataset you used, including its characteristics, number of samples, and features. You can reuse the details from Part I.
2. Model Evaluation
   - Loss Value: Report the Mean Squared Error (MSE) for both linear and ridge regression models.
   - Predictions vs Actual: Show plots comparing predicted values to actual values. If using more than 2 features, simplify plots by representing data points on the X-axis and predictions on the Y-axis.
3. Discussion:
   - Discuss the benefits and drawbacks of using OLS for weight computation.
   - Evaluate the strengths and weaknesses of linear regression in general.
   - Explain the motivation for using L2 regularization and how ridge regression improves upon linear regression. Discuss its benefits and limitations compared to linear regression.

Part IV: Elastic Net Regression using Gradient Descent [30 points]

In this part, you will implement Elastic Net Regularization using gradient descent to solve a regression problem. You will explore different weight initialization techniques and stopping criteria to evaluate their impact on model performance.

DATASET

Use any dataset from the 'noisy_datasets' folder, excluding the 'penguins' dataset and the one that you used for Part III. Datasets for Part II, III & IV must be different.

STEPS

1. Import required libraries (not allowed: *scikit-learn* or other libraries with in-built functions that help to implement ML).
2. Data analysis and preprocessing.
   Reuse your preprocessing code from Part I or apply similar preprocessing steps to the new dataset. Ensure normalization of features is done after splitting the data into training and testing sets.
3. Data preparation
   a. Define the target variable Y
   b. Create matrices for input X and target Y with shape $\mathrm{X} = N \times d$ and $\mathrm{Y} = N \times 1$, where N is a number of data samples and d is a number of features.

c. Split the dataset into training (80%) and testing (20%) sets.

d. Print the shapes of X_train, y_train, X_test, y_test.

4. Implement Elastic Net Regularization
   Elastic Net combines L1 (Lasso) and L2 (Ridge) regularization.
   The regularized loss function is defined as:

$$J(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \frac{1}{2}\lambda_1 \left|\left|w\right|\right|_2^2 + \lambda_2 \left|\left|w\right|\right|_1$$

   Where λ_1 and λ_2 are the regularization parameters for L2 and L1 penalties.

5. Solve the problem using Gradient Descent. Implement gradient descent to minimize the Elastic Net regularized loss function.
   Update the weights iteratively using:

$$w = w - \alpha\, \nabla J(w)$$

   where α is the learning rate and $\nabla J(w)$ is the gradient of the loss function.

6. Experiment with at least three different methods for initializing weights. Some ideas:

   a. **Random**: initialize weights randomly

   b. **Zero**: initialize all weights to zero

   c. **Xavier (Glorot) Initialization:** initialize weights in such a way that helps to maintain the variance of activations and gradients throughout the layers. E.g.

```
def xavier_initialization(input_dim, output_dim):
    # Calculate the limit for uniform distribution
    limit = sqrt(6 / (input_dim + output_dim))

    # Initialize weights with uniform distribution in the range
[-limit, limit]
    weights = random.uniform(-limit, limit, size=(input_dim,
output_dim))

    return weights
```

   Analyze how different weight initialization methods impact the convergence and performance of your model.

7. Experiment with training stopping criteria:

   a. Run gradient descent for a predefined number of iterations, e.g., 10,000 or 100,000.

   b. Stop the algorithm when the gradient falls within a small threshold, e.g., −0.01<gradient<0.01.

   c. Compare the performance and convergence behavior of these stopping criteria.

In your report for Part IV:

1. Provide a brief description of the dataset you used, including its characteristics, number of samples, and features. You can reuse the details from Part I.
2. Model evaluation
   - Loss Value: Report the loss results for the models.
   - Include plots comparing predictions versus actual values. If applicable, show convergence plots to illustrate the stopping criteria
   - Report on the effects of different weight initialization methods on model performance. Include insights and observations based on your experiments.
   - Discuss the performance and convergence characteristics for each stopping criterion. Provide details on how each criterion affects the optimization process.
3. Discussion:
   - Compare Elastic Net regularization with L1 and L2 regularization alone. Discuss the advantages of Elastic Net in your specific application.
   - Analyze the benefits and potential drawbacks of using Elastic Net regression and gradient descent.

Final Submission

Please reupload your work for Part I & Part II and include your completed work for Part III & Part IV along with the final report. Follow these guidelines for submitting your assignment:

1. Report (PDF File):
   - Combine the reports for all parts (Part I, Part II, Part III, and Part IV) into a single PDF file. Name as TEAMMATE1_TEAMMATE2_assignment1_report.pdf e.g., avereshc_pinazmoh_assignment1_report.pdf
2. Code (Jupyter Notebooks with saved outputs):
   - Part I: Save as a separate Jupyter Notebook file, named as TEAMMATE1_TEAMMATE2_assignment1_part_1.ipynb e.g., avereshc_pinazmoh_assignment1_part_1.ipynb
   - Part II: Save as a separate Jupyter Notebook file, named as TEAMMATE1_TEAMMATE2_assignment1_part_2.ipynb
   - Part III: Save as a separate Jupyter Notebook file, named as TEAMMATE1_TEAMMATE2_assignment1_part_3.ipynb
   - Part IV: Save as a separate Jupyter Notebook file, named as TEAMMATE1_TEAMMATE2_assignment1_part_4.ipynb
3. Saved weights. For Part II, III & IV save the model weights that achieved the best

results. Save as:
TEAMMATE1_TEAMMATE2_assignment1_part2.pickle
TEAMMATE1_TEAMMATE2_assignment1_part3.pickle
TEAMMATE1_TEAMMATE2_assignment1_part4.pickle

4. Preprocessed datasets:
    - Save and include the preprocessed datasets, named as: DATASET_preprocessed.csv (e.g., penguins_preprocessed.csv).
5. Combine all files in a single zip folder that will be submitted on UBlearns, named as TEAMMATE1_TEAMMATE2_assignment1_checkpoint.zip
6. Notes:
    - Ensure your code is well-organized and includes comments explaining key functions and attributes.
    - After running the Jupyter Notebooks, all results and plots used in your report should be generated and clearly displayed.
    - The zip folder should include all relevant files, clearly named as specified.

Bonus points [max 10 points]

Buffalo-related Dataset [8 points]

A key part of any research is identifying a suitable dataset for your problem.

STEPS:

1. Select a Buffalo-related dataset available at Buffalo Open Data Portal. Min requirements: it should contain >1k entries.
2. Perform data analysis (Part I) and apply any method that you have implemented as part of this assignment (e.g Part II, III or IV). You can consider applying multiple methods.
3. Train your model and provide the results. Accepted accuracy is above 70%
4. Submit your work as a separate Jupter Notebook with saved weights

Improved Accuracy on Penguin dataset [2 points]

Apply various technics to achieve an accuracy of more than 85% for penguin dataset.

Bonus part submission:

- Create a separate Jupyter Notebook (.ipynb) named as TEAMMATE1_TEAMMATE2 _assignment1_bonus.ipynb

e.g., avereshc_ pinazmoh_ assignment1_bonus.ipynb

- You can duplicate code from your other parts if needed
- Submit Jupyter Notebook (.ipynb) with saved outputs
- A file with saved weights that generate the best results for your model (.pickle).
- Report is not required; you can include all the analysis as part of your Jupyter Notebook.

ASSIGNMENT STEPS

1. Register your team (September 16)

You may work individually or in a team of up to 2 people. The evaluation will be the same for a team of any size.

Register your team at UBLearns > Groups. In case you joined the wrong group, make a private post on piazza.

2. Submit checkpoint (September 26)

- Complete Part I and Part II of the assignment
- For the checkpoint report for Part I and Part II is not mandatory, you can complete the report and submit it along with the final submission
- Add all your assignment files in a zip folder including .ipynb files, the report (if available) and .pickle file at UBLearns > Assignments
- Name zip folder with all the files as
  TEAMMATE1_TEAMMATE2 _assignment1_checkpoint.zip
  e.g., avereshc_ pinazmoh_ assignment1_checkpoint.zip
- Submit to UBLearns > Assignments
- If you are working in a team, we expect equal contribution for the assignment. Each team member is expected to make a code-related contribution. Provide a contribution summary by each team member in the form of a table below. If the contribution is highly skewed, then the scores of the team members may be scaled w.r.t the contribution.

Team Member	Assignment Part	Contribution (%)

3. Submit final results (October 10)

- Fully complete all parts of the assignment

- In general, you are allowed to improve what you have done for the checkpoint submission
- Submit to UBLearns > Assignments
- Add all your assignment files in a zip folder including ipynb files for Part I, Part II, Part III, Part IV, & Bonus part (optional), the report and .pickle file at UBLearns > Assignments
- Name zip folder with all the files as
  TEAMMATE1_TEAMMATE2 _assignment1_final.zip
  e.g. avereshc_ pinazmoh_ assignment1_final.zip
- Your Jupyter notebook should be saved with the results. If you are submitting python scripts, after extracting the ZIP file and executing command python main.py in the first level directory, all the generated results and plots you used in your report should appear printed out in a clear manner.
- Include all the references that have been used to complete the assignment.
- If you are working in a team, we expect equal contribution for the assignment. Each team member is expected to make a code-related contribution. Provide a contribution summary by each team member in the form of a table below. If the contribution is highly skewed, then the scores of the team members may be scaled w.r.t the contribution.

Team Member	Assignment Part	Contribution (%)

Academic Integrity

The standing policy of the Department is that all students involved in any academic integrity violation (e.g., plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as “Copying or receiving material from any source and submitting that material as one’s own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one’s own.”. Refer to the Office of Academic Integrity for more details.

Important Information

This project can be done in a team of up to two people.

- All team members are responsible for the project files submission
- No collaboration, cheating, and plagiarism is allowed in assignments, quizzes, the midterms or final project.
- All the submissions will be checked using SafeAssign as well as other tools. SafeAssign is based on the submitted works for the past semesters as well as the current submissions. We can see all the sources, so you don't need to worry if there is a high

similarity with your Checkpoint submission.

- The submissions should include all the references. Kindly note that referencing the source does not mean you can copy/paste it fully and submit it as your original work. Updating the hyperparameters or modifying the existing code is subject to plagiarism. Your work must be original. If you have any doubts, send a private post on piazza to confirm.
- All group members and parties involved in any suspicious cases will be officially reported using the Academic Dishonesty Report form. What does that mean?
    - In most cases, the grade for the assignment/quiz/final project/midterm will be 0 and all bonus points will be subject to removal from the final evaluation for all students involved.
    - Those found violating academic integrity more than once throughout their program will receive an immediate F in the course.

  Please refer to the Academic Integrity Policy for more details.
- The report should be delivered as a separate pdf file. You can combine reports for Part I, Part II, Part III, & Part IV into the same pdf file. You can follow the NIPS template as a report structure (NOT mandatory). You may include comments in the Jupyter Notebook; however, you will need to duplicate the results in a separate pdf file.
- All the references can be listed at the end of the report. There is no minimum requirement for the report size, just make sure it includes all the information required.
- For the Bonus part, no report is needed.

Late Days Policy

You can use up to 7 late days throughout the course that can be applied to any assignment-related due dates. You do not have to inform the instructor, as the late submission will be tracked in UBLearns.

If you work in teams, the late days used will be subtracted from both partners. In other words, you have 4 late days, and your partner has 3 late days left. If you submit one day after the due date, you will have 3 days, and your partner will have 2 late days left.

Important Dates

Sep 26, Thursday - Checkpoint is Due

Oct 10, Thursday - Final Submission is Due