# Lab Course Machine Learning

**Exercise Sheet 4**

Prof. Dr. Dr. Lars Schmidt-Thieme
Jung Min Choi
HiWi: Harish Malik

Submission deadline : **November 14, 2024**

**General Instructions**

1. Perform a data analysis, deal with missing values and any outliers.
2. Unless explicitly noted, you are not allowed to use scikit, sklearn or any other library for solving any part.
3. Data should be normalized.
4. Train to Test split should be 80-20.
5. Convert any non-numeric values to numeric values. For example you can replace a country name with an integer value or more appropriately use one-hot encoding.

## 1 Regularization and Hyperparameter Tuning (10 points)

In this task, you will work with the same dataset as before, named *'logistic.csv'*. Using the previous implementation of the **LogisticRegression** class, perform the following steps:

**Overview of Stochastic Gradient Descent (SGD):** SGD is an iterative optimization algorithm for minimizing a loss function $L(\theta)$ by updating model parameters $\theta$. At each step, instead of using the entire dataset, a small random batch of data $(X, Y)$ is sampled, and the model is updated using:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta L(\theta^{(t)}; X, Y)$$

Where:

- $\theta^{(t)}$: Model parameters at iteration $t$
- $\eta$: Learning rate (step size)
- $\nabla_\theta L$: Gradient of the loss function with respect to $\theta$

**Pseudo-Code for SGD:**
**Algorithm:** Stochastic Gradient Descent **Input:** Learning rate $\eta$, initial parameters $\boldsymbol{\theta}_0$, batch size $b$, maximum iterations $T$

1. **For** $t = 1$ to $T$:
   a) Sample a mini-batch $(\boldsymbol{X}_{\text{batch}}, \boldsymbol{Y}_{\text{batch}})$ of size $b$ from the dataset.
   b) Compute the gradient of the loss function with respect to $\boldsymbol{\theta}$:
   $$\text{grad} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \boldsymbol{X}_{\text{batch}}, \boldsymbol{Y}_{\text{batch}})$$
   c) Update the parameters:
   $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \text{grad}$$
2. **End For**

**Output:** Optimized parameters $\boldsymbol{\theta}$

**A [2 point]** Extend the ***LogisticRegression*** class to include a **Stochastic Gradient Descent** (SGD) method for optimization.

**B [0.5 points]** Extend the ***Optimization*** or ***Loss*** class to include the L1/L2 regularized Cross-Entropy Loss.

**C [2 points]** Fit logistic regression models with the following combinations:

- **Cross-Entropy Loss and Stochastic Gradient Descent.**
- **L1-Regularized Cross-Entropy Loss and Stochastic Gradient Descent.**
- **L2-Regularized Cross-Entropy Loss and Stochastic Gradient Descent.**

**D [4 points]** Perform **Backward Feature Selection** iteratively using the **AIC Metric**:

- Apply this only to the combination of **Cross-Entropy Loss and SGD.**
- Select the most important features based on AIC.

**E [1.5 points]** Generate and report the following:

- The loss trajectories for both training and testing sets for all cases.
- The final train and test accuracies for all models.
- Generate a confusion matrix for the test set to show the following metrics (Only for the Cross-Entropy and Stochastic Gradient Descent):
  - True Positives (TP)
  - True Negatives (TN)
  - False Positives (FP)
  - False Negatives (FN)

**Note:** You may use the `confusion_matrix` function from the `sklearn.metrics` module to generate the confusion matrix and extract the required metrics.

## 2 K-Fold Cross Validation (6 points)

In this part of the assignment, you will implement ***Grid Search*** with ***K-Fold Cross-Validation*** for model selection, i.e., for choosing the best hyperparameters.

**A [3 point]** Use $k = 3$ folds for validation and implement the L2-regularized cross-entropy loss with gradient descent and fixed step-length control. Identify the optimal L2-regularization parameter $\lambda$ and step length $\alpha$.

**B [1 point]** Keep track of the mean performance (e.g., accuracy) across the $k$ folds for each combination of hyperparameters.

**C [1 point]** Plot a grid of $\alpha$ vs $\lambda$ with the accuracy score for all combinations. You may use a 3D plot, with axes as $\alpha$, $\lambda$, and accuracy.

**D [1 point]** Using the optimal values of $\alpha$ and $\lambda$, train your model on the complete training data and evaluate its performance on the test data. Report the final accuracy.

## 3 Coordinate Descent for L1-Regularized Linear Regression (4 points)

You are provided with a dataset named *'regression2.csv'*.

**A [2 point]** Implement the **L1-regularized linear regression loss function** in Python.

**B [2 points]** Using only **NumPy**, implement the **Coordinate Descent** algorithm to minimize the L1-regularized linear regression loss function.

**C [1 point]** Visualize the change of each coefficient **over iterations**.