Tasks 6 and 7: Linear, Logistic Regression

Linear regression is a simple ML method used to predict continuous values based on one or more inputs.
It involves one dependent (output) and one or more independent (input) variables.
Logistic Regression is meant for classification, that is, classifying outcomes into different groups based on certain parameters and weights.

Linear Regression
Types of Linear Regression:

1) Single variable: One input, one output
2) Multivariable: Many inputs, one output
3) Polynomial: Many inputs of different factors, one output

We mostly use multivariable linear regression in real life problem solving.
Linear Regression is used in cases where outputs are continuous, numeric in nature.

$$\boxed{y = w_0 + w_1 x_1 + \ldots + w_k x_k + \epsilon.}$$

Multivariable Linear Regression Regression

Our main goal in cases of Linear Regression is to minimise the Mean Square Error (MSE)

Tasks 6 and 7: Linear, Logistic Regression

$$\boxed{\hat{w}_{LR} = \arg\min_{w} \sum_{i} (y_i - wx_i)^2}$$

MSE Calculation Function

Logistic Regression

Logistic Regression is a machine learning method used for classification tasks. It classifies outcomes into different groups based on certain parameters and weights[1]. Unlike linear regression which predicts continuous values, logistic regression is specifically designed for situations where the output is categorical.

When to use Logistic Regression:

- Binary Classification (e.g., Yes/No, True/False, Spam/Not Spam)
- Multi-class Classification (e.g., classifying images of different animals)

The core idea behind logistic regression is to use a sigmoid (or logistic) function to map the predicted values to a probability between 0 and 1. This probability is then used to classify the outcome into one of the predefined groups.

The general form of the logistic regression equation is often represented as:

$$P(Y=1|X)=\frac{1}{1+ e^{-(w0 + w1x1 + \dots + wnxn)}}$$

Tasks 6 and 7: Linear, Logistic Regression

Our main goal in Logistic Regression is to maximize the likelihood of observing the training data, which often involves minimizing a cost function like cross-entropy loss.

Hyperparameter Tuning

Hyperparameter tuning is the process of finding the optimal set of hyperparameters for a machine learning model. Hyperparameters are parameters whose values are set *before* the learning process begins, unlike model parameters (like weights in linear/logistic regression) which are learned *during* training.

Why is Hyperparameter Tuning Important?

Different hyperparameter values can significantly impact a model's performance, affecting its accuracy, training time, and generalization ability. Tuning helps to:

- **Improve Model Performance:** Find the combination of hyperparameters that yields the best results on unseen data.
- **Prevent Overfitting/Underfitting:** Appropriate hyperparameter values can help strike a balance between a model that is too complex (overfitting) or too simple (underfitting).

Common Hyperparameter Tuning Techniques:

1. **Manual Search:** Manually trying different combinations of hyperparameters based on experience and intuition. This can be time-consuming and inefficient for many hyperparameters.
2. **Grid Search:** An exhaustive search through a manually specified subset of the hyperparameter space. It tries every possible combination of values.
3. **Random Search:** Randomly samples hyperparameter combinations from a specified distribution for a fixed number

of iterations. Often more efficient than Grid Search when dealing with many hyperparameters.

4. **Bayesian Optimization:** Uses a probabilistic model to guide the search for optimal hyperparameters, attempting to balance exploration (trying new combinations) and exploitation (refining promising combinations).
5. **Gradient-based Optimization:** Some hyperparameters can be optimized using gradient descent-like methods, though this is less common for all types of hyperparameters.

Hyperparameters in Logistic Regression:

For Logistic Regression, common hyperparameters that might be tuned include:

- **Regularization strength (C or alpha):** Controls the penalty for complexity in the model to prevent overfitting.
- **Solver:** The algorithm used for optimization (e.g., 'liblinear', 'lbfgs', 'saga').
- **Max Iterations:** The maximum number of iterations taken for the solvers to converge.

Refer the ipynb notebooks for code implementation of the same

task6.ipynb: Linear Regression basics
task7.ipynb: Linear and Logistic Regression on a synthetic voters dataset
task7.ipynb: Logistic Regression on the iris dataset