Model 2

Logistic Regression

Logistic Regression (LR) is a statistical and machine learning algorithm that is used for binary classification problems.

Unlike linear regression, which predicts continuous values, LR predicts the probability that a given input belongs to a certain class.

The core of LR is the sigmoid function, which maps a real-valued number into a value between 0 and 1.

LR uses a threshold (usually 0.5) to classify the data. This threshold can be adjusted depending on the problem. For example, in cases where false positives are costly, we might set a higher threshold.

Training LR: LR is trained using a technique called Maximum Likelihood Estimation (MLE). The goal is to find the optimal weights that maximize the likelihood of the observed data.

Instead of minimizing the error as in linear regression, LR minimizes a loss function called the log-loss or cross-entropy loss.

LR also supports regularization to prevent overfitting by adding a penalty to the loss function. Two types or regularizations commonly used:

- L2 Regularization (Ridge): Penalize the sum of the squared weights.
- L1 Regularization (Lasso): Penalize the absolute sum of the weights.

In our model, we used L2.

Limitations: LR assumes a linear relationship between the input features and the log-odds of the outcome. Also, like many classifiers, LR can struggle when the classes are imbalanced. Techniques like SMOTE or adjusting class weights can help mitigate that.

Mathematical Explanation:

1. Linear Model:

The first step of LR is similar to a linear regression model. We calculate the linear combination of the input features X:

$$z = w_0 + w_1X_1 + w_2X_2 + \cdots + w_nX_n$$

Where:

- z is the linear predictor ($z \in \mathbb{R}$),
- $X_1, X_2, \dots, X_n$ are the input features (columns),
- w_0 is the intercept (bias term),
- $w_1, w_2, \dots, w_n$ are the weights (coefficients) associated with each feature.

2. Sigmoid Function:

The key difference between LR and linear regression is that LR does not predict a continuous value, but rather the probability that the dependent variable belongs to a particular class (0 or 1). To convert the linear combination z into a probability, we use the sigmoid function:

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

Thus, the predicted probability $\hat{p}$ of the positive class (class 1) is:

$$\hat{p}(y = 1|X) = \frac{1}{1 + e^{-(w_0 + w_1X_1 + \cdots + w_nX_n)}}$$

And the probability of the negative class is:

$$\hat{p}(y = 0|X) = 1 - \hat{p}(y = 1|X)$$

3. **Log-Odds (Logit Function):**

In LR, the linear predictor z represents the log-odd of the positive class. The odds of an event are the ration of the probability of the event occurring to the probability of it not occurring:

$$\text{odds} = \frac{P(y = 1)}{P(y = 0)} = \frac{\hat{p}}{1 - \hat{p}}$$

Taking the natural logarithm of the odds gives the log-odds or logit:

$$\log\left(\frac{\hat{p}}{1 - \hat{p}}\right) = w_0 + w_1 X_1 + \cdots + w_n X_n$$

Thus, LR models the log-odds as a linear function of the input features. This is why LR is sometimes referred to as a log-linear model.

4. **Maximum Likelihood Estimation (MLE):**

To train a LR model, we need to find the best parameters (weights $w_0, w_1, \dots, w_n$) that maximize the likelihood of observing the given data. This is done using MLE.

Likelihood Function: The probability of the observed labels given the features and the model's parameters. For a binary classification problem with m training examples, the likelihood function is:

$$L(w) = \prod_{i=1}^{m} P\left(y^{(i)} \middle| X^{(i)}\right)$$

Since $y^{(i)}$ can either be 0 or 1, we can rewrite the likelihood as:

$$L(w) = \prod_{i=1}^{m} \left(\hat{p}^{(i)}\right)^{y^{(i)}} \left(1 - \hat{p}^{(i)}\right)^{1 - y(i)}$$

Log-Likelihood: It is more convenient to maximize the log-likelihood (the logarithm of the likelihood function) rather than the likelihood itself, as it converts the product of probabilities into a sum, which is easier to work with:

$$\ell(w) = \sum_{i=1}^{m} \left[y^{(i)} \log \hat{p}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{p}^{(i)}\right)\right]$$

Maximizing this log-likelihood with respect to the weights w leads to the optimal parameters for the model. This is typically done using optimization algorithms such as gradient descent.

5. **Decision Boundary:**

The decision boundary in LR is the threshold at which we classify an instance as belonging to class 1 or class 0. By default, we classify an instance as 1 if the predicted probability $\hat{p}(y = 1|X) \geq 0.5$. In terms of the linear predictor z:

$$z = w_0 + w_1X_1 + w_2X_2 + \cdots + w_nX_n = 0$$

This equation defines a hyperplane that separates the two classes. LR finds the best hyperplane to separate the classes in feature space.

Advantages and Disadvantages of LR

- **Advantages:**
  1. **Interpretability:** LR provides a clear understanding of how each feature contributes to the prediction. Each feature's coefficient represents its contribution to the probability of the positive class.
  2. **Efficiency:** It is computationally efficient, especially for large datasets.
  3. **Probabilistic Output:** LR outputs probabilities, which can be valuable in finance.

- **Disadvantages:**
  1. **Linearity Assumption:** LR assumes a linear relationship between the features and the log-odds of the target. This might be a limitation since financial data often contains non-linear relationships. For example, Bitcoin price movements may not have a simple linear dependence on volatility or external economic indicators.

2. **Feature Engineering Required:** The effectiveness of LR might depend on how well we've engineered the features. Non-linear patterns may need to be captured using interaction terms or transformations, which add complexity.

3. **Sensitivity to Outliers and Multicollinearity:** Since financial data can often have outliers (e.g. sudden price spikes), LR can be sensitive to these unless they are handled properly. Additionally, if there are highly correlated features (e.g. Price, Open Price, High Price), it could lead to multicollinearity, which might negatively affect the model.

Conclusion:

Logistic Regression is a powerful and widely used machine learning algorithm designed for binary classification problems. It models the relationship between a set of input features and a binary target variable by estimating the probability of the target being in one of two classes. Unlike linear regression, which predicts continuous values, Logistic Regression uses the sigmoid function to map any real-valued input to a probability between 0 and 1, making it ideal for classification tasks.

In scenarios where a straightforward, interpretable model is required, Logistic Regression is an excellent choice. However, for more complex, non-linear data, more advanced models like Random Forest or Neural Networks might be necessary. Nevertheless, Logistic Regression remains a foundational algorithm for classification tasks, combining simplicity, efficiency, and interpretability.