## Stochastic Gradient Descent Classifier (SGD)

The Stochastic Gradient Descent (SGD) classifier is a potent and adaptable supervised machine learning algorithm capable of executing classification tasks. At its core, SGD updates parameters iteratively in response to each training example, making it particularly efficient for large datasets. The goal of SGD is to minimize the loss function by adjusting weights based on the gradient of the loss with respect to each weight.

Data Preparation and Splitting

Prior to beginning the modeling process, we split our dataset via a train-test partitioning scheme, using 80% for training and the remaining 20% for validation. Given the inherent class imbalances in our dataset, maintaining a consistent representation of classes in both subsets was vital. To ensure this, the stratify argument was leveraged, preserving the distribution of the target classes across both the training and test datasets.

Hyperparameter Tuning

A crucial hyperparameter for the SGD Classifier is the learning rate, often denoted as alpha. This parameter regulates the step size at each iteration while moving toward a minimum of the loss function. The correct balance is imperative; too large an alpha might overshoot the minimum, while too small could lead to slow convergence or getting stuck in local minimums.

To optimize the selection of the alpha parameter, we deployed a Randomized Search CV strategy. Instead of a comprehensive exploration of all possible values, this approach samples from a predefined distribution of possible alpha values, providing a balance between accuracy and efficiency. Our tuning procedure involved 5-fold cross-validation, ensuring robustness by testing the model on various sub-samples of the training data.

Specific Configurations

Our instantiation of the SGD Classifier was characterized by a few distinct configurations:

1. **Loss = 'Log\_Loss'**: Opting for the logarithmic loss function (or log loss) transforms our SGD Classifier into a logistic regression, allowing the model to estimate probabilities, thus furnishing more nuanced decisions.
2. **Penalty = 'L2'**: This selection applies L2 regularization to the model. Regularization penalizes complex models, providing a countermeasure against overfitting. The L2 penalty specifically targets the squared magnitudes of the coefficient, promoting feature weights to be small and smooth.
3. **Class Weight = 'Balanced'**: Given the uneven distribution of classes in our target variable, it was of paramount importance to address this disparity. The 'balanced' setting adjusts the weights automatically, in an inversely proportional manner relative to class frequencies. This ensures that the model does not inherently favor the majority class, promoting a more unbiased learning experience.
4. **Early Stopping = True**: This option enables the Early Stopping mechanism, which is a form of regularization aimed at preventing overfitting. By monitoring the model's performance on a validation set during training, Early Stopping halts the learning process once it detects that the performance has ceased to improve, or if it begins to degrade. This not only helps in avoiding overfitting, but also reduces the training time, as the model will stop learning as soon as it reaches an optimal state, rather than continuing to iterate until a predetermined number of epochs has been completed.

**Results:**

**Confusion Matrix Analysis:** The confusion matrix provides critical insights into classification performance by detailing true positives, false positives, true negatives, and false negatives for each class:

* **Allow**: Out of 7,528 instances, the model accurately classified 7,414 as 'allow'. It misclassified 56 as 'deny', 58 as 'drop', and 0 as 'reset-both', showcasing an impressive prediction accuracy for this class.
* **Deny**: From the 2,998 'deny' instances, the model rightly predicted 2,923. There were 67 instances mistakenly classified as 'drop', 8 as 'allow', and none as 'reset-both'.
* **Drop**: For the 'drop' category, the model exhibited outstanding performance by correctly classifying all 2,570 instances without any misclassifications.
* **Reset-Both**: The 'reset-both' category remained challenging for the model. Out of 11 instances, none were accurately classified, with 1 instance mislabeled as 'allow' and 10 as 'deny'.

**Classification Report:** The classification report presents a thorough assessment of critical metrics, such as precision, recall, and F1-score for each class across various splits:

* Allow: Across the splits, the model consistently achieved high precision, recall, and F1-score values, typically around 1.00, 0.99, and 1.00 respectively, indicating excellent performance in identifying 'allow' instances.
* Deny: The model demonstrated strong precision and recall values, generally around 0.98 and 0.97-0.98 respectively, and an F1-score of 0.97-0.98, showcasing its effectiveness in categorizing 'deny' cases.
* Drop: The model maintained a high precision of around 0.94-0.97, paired with a perfect recall of 1.00 and an F1-score of 0.97-0.98, highlighting its proficiency in the 'drop' classification.
* Reset-Both: This class posed significant challenges for the model across all splits, yielding precision, recall, and F1-score values consistently at 0.00, reflecting the difficulties associated with this scarcely represented category.

Additionally, the model maintained a commendable overall accuracy of around 0.98-0.99 across the splits while our average comes out to 0.984. The weighted average metrics for precision, recall, and F1-score were consistently close to 0.98-0.99, emphasizing the model's overall effectiveness. However, the macro averages, which give equal importance to each class, revealed the model's struggles with the 'reset-both' class, typically resulting in values of 0.73-0.74 for precision, recall, and F1-score.