Part 1

A graph of loss and loss

AI-generated content may be incorrect.

Part 2

A colorful circle with different colored triangles

AI-generated content may be incorrect.

Grid Search

Metrics:

We cannot use accuracy since the class distribution is not balanced. Since we want to balance per-class quality and class importance. It is the best choice **f1\_weighted** for our case since we want a metric which can give us an avarage of f1 accross all classes weighted by class sizes.

Random Forest:

* n\_estimators: number of trees in the forest (default=100)
* class\_weight :

The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y))

The “balanced\_subsample” mode is the same as “balanced” except that weights are computed based on the bootstrap sample for every tree grown.

Linear Legression:

* **solver** =‘liblinear’ can only handle binary classification by default. To apply a one-versus-rest scheme for the multiclass setting one can wrap it with the OneVsRestClassifier. Thats wh we didnt use ‘liblinear’ since we are not doing binary classification.
* **solver** =‘newton-cholesky’ is not a good choice since the memory usage of this solver has a quadratic dependency on n\_features \* n\_classes because it explicitly computes the full Hessian matrix.
* **solver** =“lbfgs”, “newton-cg”, “sag”, “saga” can be used for multinomial multiclass classifications.
* multi\_class='multinomial':

👉 Use the softmax function to model all classes together, not one-vs-rest.

* multi\_class='ovr' (one-vs-rest):

👉 Fit one binary classifier per class (e.g., cat vs not-cat, dog vs not-dog...).

* max\_iter, maximum number of iterations to converge (default: 100)

A graph of a logistic regression

AI-generated content may be incorrect.