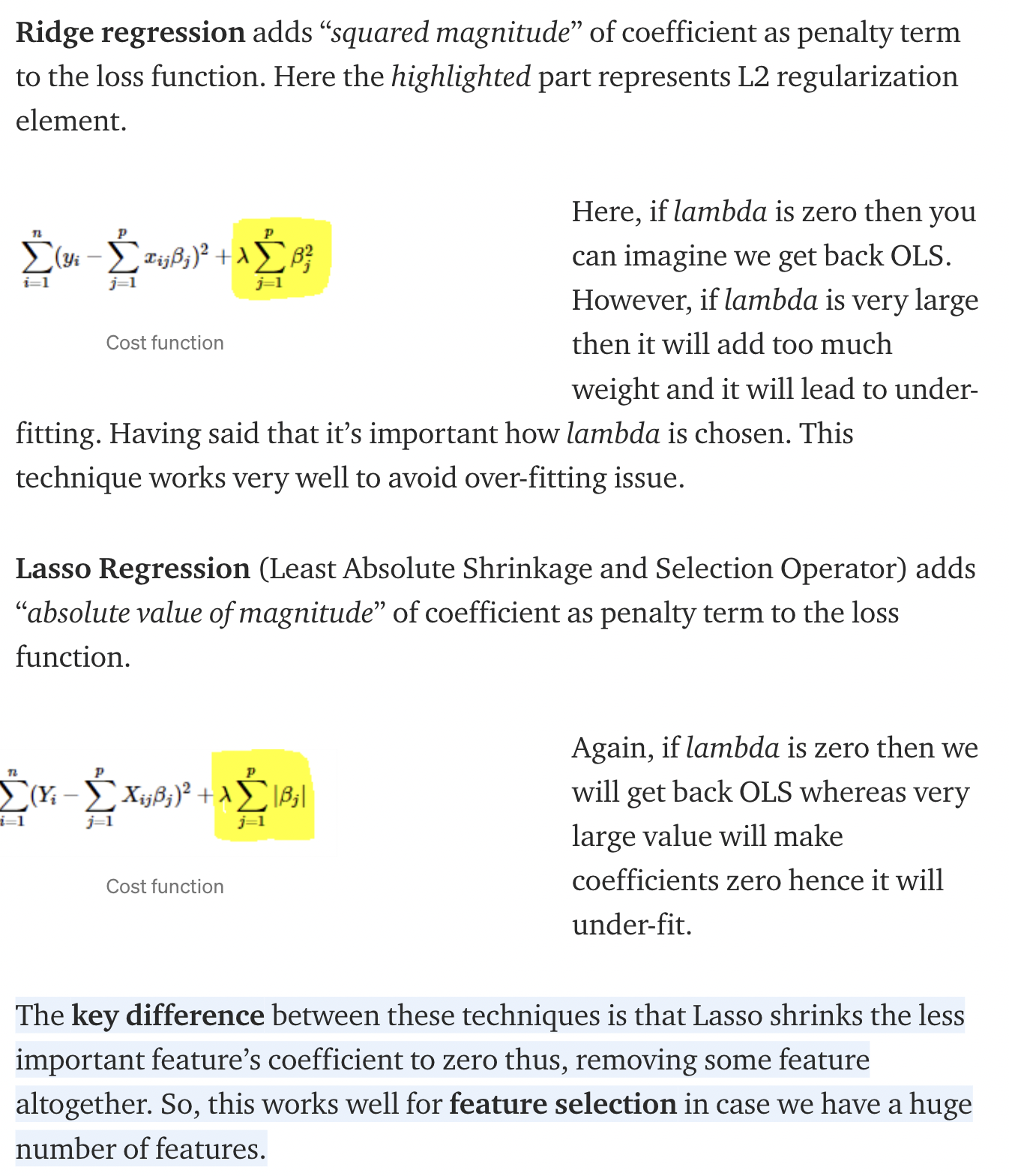
Linear[SV](https://www.youtube.com/watch?v=efR1C6CvhmE)C parameters

* **penalty*{‘l1’, ‘l2’}, default=’l2’***

Specifies the norm used in the penalization. The ‘l2’ penalty is the standard used in SVC. The ‘l1’ leads to coef\_ vectors that are sparse.



<https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>

<https://www.youtube.com/watch?v=Q81RR3yKn30>

<https://www.youtube.com/watch?v=NGf0voTMlcs>

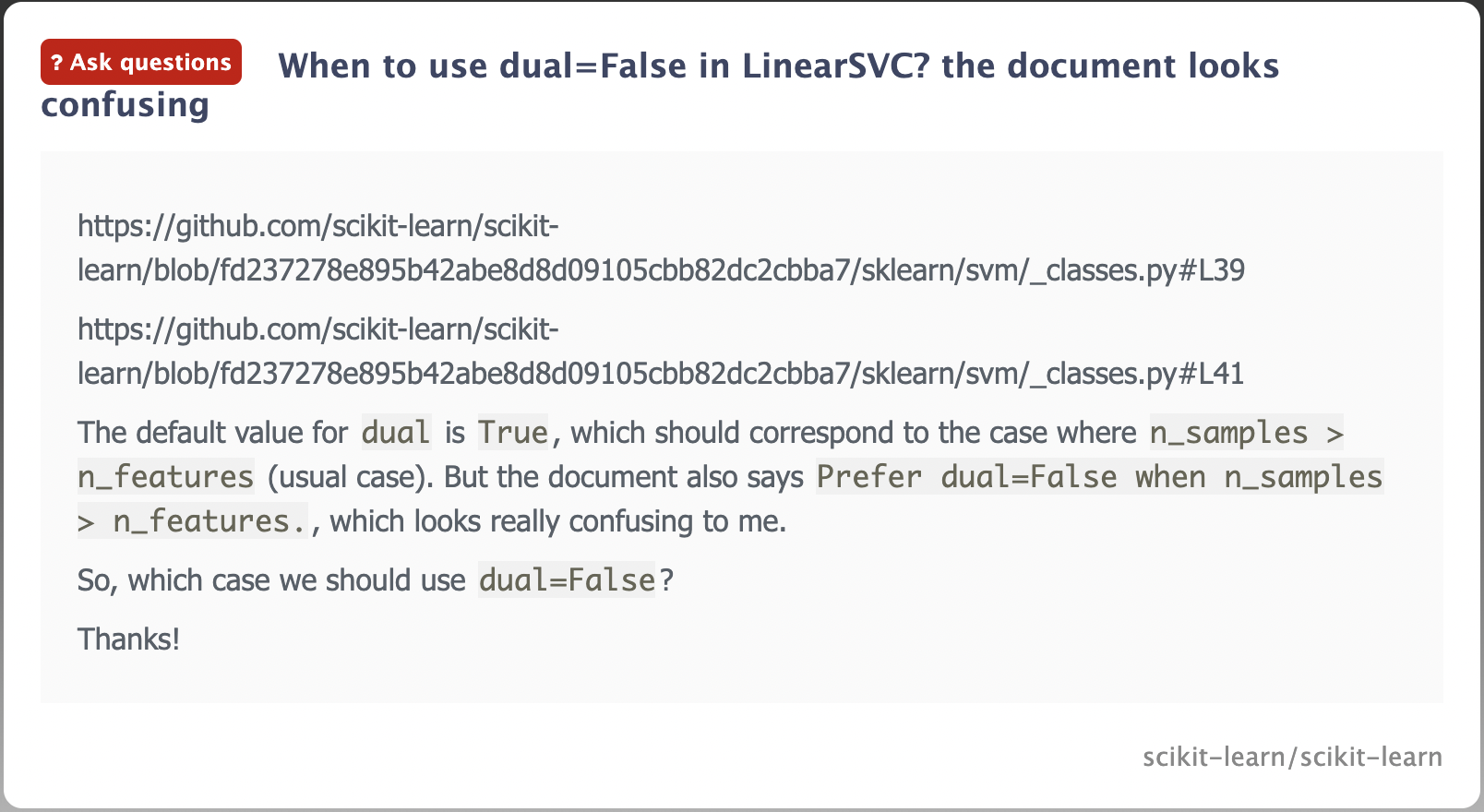
* **loss*{‘hinge’, ‘squared\_hinge’}, default=’squared\_hinge’***

Specifies the loss function. ‘hinge’ is the standard SVM loss (used e.g. by the SVC class) while ‘squared\_hinge’ is the square of the hinge loss. The combination of penalty='l1' and loss='hinge' is not supported.

* **dual: *bool, default=True***

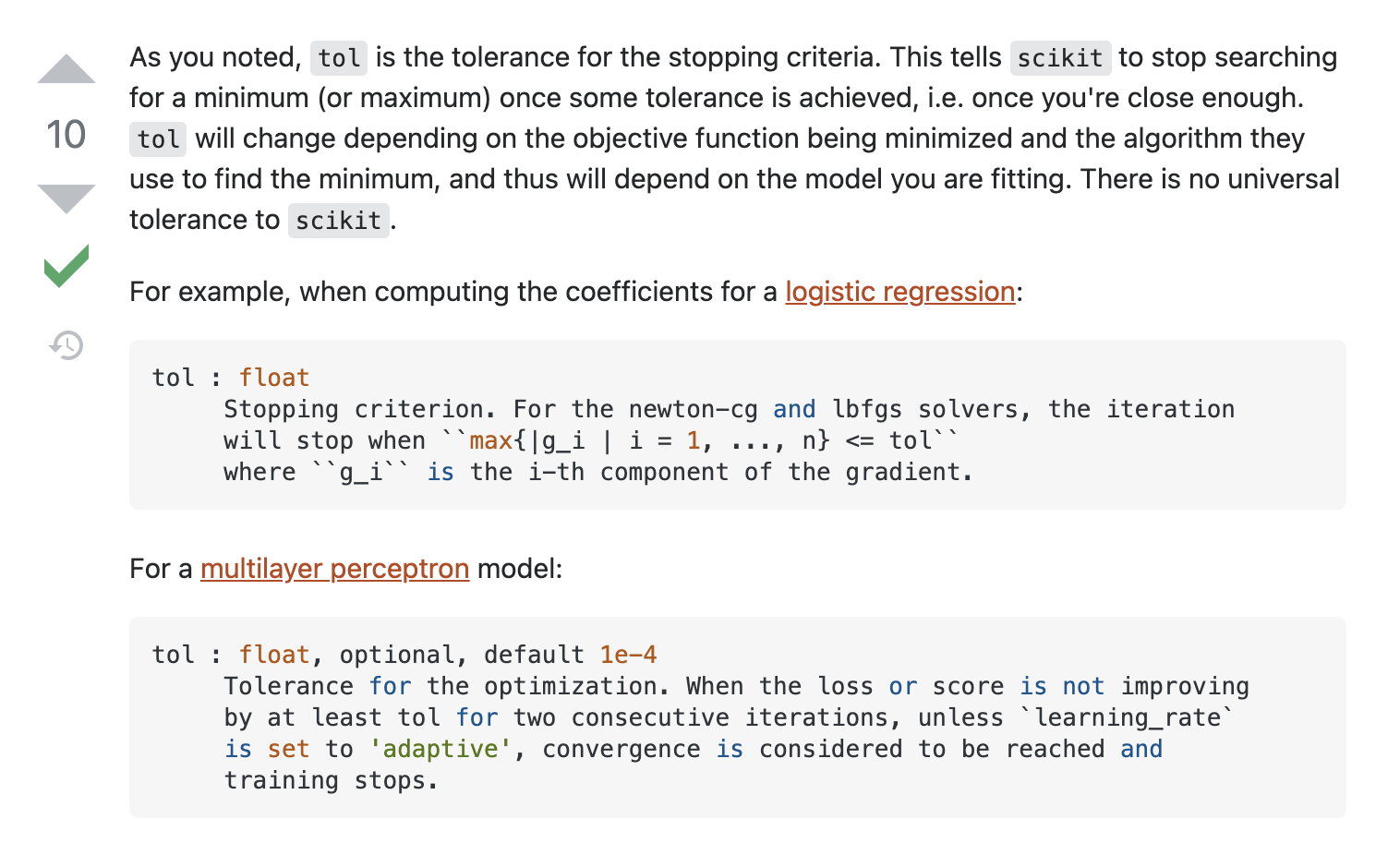
Select the algorithm to either solve the dual or primal optimization problem. Prefer dual=False when n\_samples > n\_features.

<https://datascience.stackexchange.com/questions/31914/what-does-the-dual-parameter-in-sklearn-svm-linearsvc-and-sklearn-svm-linearsv>



<https://www.gitmemory.com/issue/scikit-learn/scikit-learn/17339/633782675>

* **tol: *float, default=1e-4***

Tolerance for stopping criteria.

https://stats.stackexchange.com/questions/255375/what-exactly-is-tol-tolerance-used-as-stopping-criteria-in-sklearn-models

* **C : *float, default=1.0***

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.

* **multi\_class*{‘ovr’, ‘crammer\_singer’}, default=’ovr’***

Determines the multi-class strategy if y contains more than two classes. "ovr" trains n\_classes one-vs-rest classifiers, while "crammer\_singer" optimizes a joint objective over all classes. While crammer\_singer is interesting from a theoretical perspective as it is consistent, it is seldom used in practice as it rarely leads to better accuracy and is more expensive to compute. If "crammer\_singer" is chosen, the options loss, penalty and dual will be ignored.

* **fit\_intercept: *bool, default=True***

Whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations (i.e. data is expected to be already centered).

the intercept where our line intercepts the y-axis. In machine learning we can call intercepts *bias*. Bias offsets all predictions that we make.(<https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html>)

For linear models, the intercept is the value of the linear predictor when all covariates are zero. In linear regression, this is equivalent to the y-intercept of the line of best fit. In logistic regression, it is the log odds of the baseline group. Suppose we did not add an intercept term for the regression. We would then be positing that when all covariates are 0, the linear predictor is 0. In summation, we add the bias to improve interpretability and add flexibility to the model.

Having no independent variables means no prediction can be made. Suppose you have a model for cancer incidence based on age. Since you do not know my age, you can not put my data into your model, and thus you can not make predictions.

The intercept represents the base assumption. Let's say you build an algorithm to classify cats and dogs where +1 is cat and -1 is dog. A model with an intercept of -1 will always assume the case is a dog unless enough variables prove it is a cat. This is especially likely to happen when you have severe class imbalance.

If your goal is performance, keep the intercept. If your goal is to determine feature importance, do not fit an intercept. By setting the base assumption to zero you force the model to select features for both cat and dog classes - no cheating!(<https://stats.stackexchange.com/questions/380201/machine-learning-use-of-intercept-in-regression>)

* **intercept\_scaling : *float, default=1***

When self.fit\_intercept is True, instance vector x becomes [x, self.intercept\_scaling], i.e. a “synthetic” feature with constant value equals to intercept\_scaling is appended to the instance vector. The intercept becomes intercept\_scaling \* synthetic feature weight Note! the synthetic feature weight is subject to l1/l2 regularization as all other features. To lessen the effect of regularization on synthetic feature weight (and therefore on the intercept) intercept\_scaling has to be increased.

* **class\_weight : *dict or ‘balanced’, default=None***

Set the parameter C of class i to class\_weight[i]\*C for SVC. If not given, all classes are supposed to have weight one. 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)).

* **verbose: *int, default=0***

Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in liblinear that, if enabled, may not work properly in a multithreaded context.

* **random\_state: *int, RandomState instance or None, default=None***

Controls the pseudo random number generation for shuffling the data for the dual coordinate descent (if dual=True). When dual=False the underlying implementation of [**LinearSVC**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC) is not random and random\_state has no effect on the results. Pass an int for reproducible output across multiple function calls. See [Glossary](https://scikit-learn.org/stable/glossary.html#term-random_state).

* **max\_iter: *int, default=1000***

The maximum number of iterations to be run