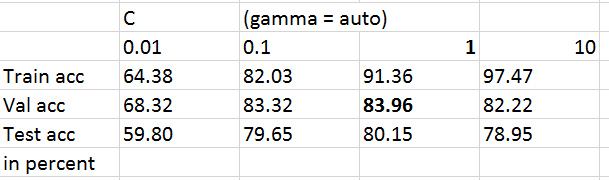
We have trained a SVM with Gaussian RBF Kernel to classify the given trajectories into ‚cross‘ and ‚no cross‘. The program pipeline is as follows:

* Take input trajectories from matlab-generated dataset
* Splice trajectories into trajectory sniplets (used stepsize = 10 timesteps)
* Create input X and Y, where each element corresponds to one trajectory sniplets
  + x\_i = [x1\_1 x2\_1 x1\_2 x2\_2 … x1\_10 x2\_10] € R(1 x 2\*(stepsize))
  + y\_i = [1]
* Train SVM with training dataset, using sklearn library (used ~2000 Trajectory sniplets)
* Validate model on held out dataset (used ~3000 Trajectory sniplets)
* Test accuracy with another held out dataset (used ~2000 Trajectory sniplets)

The choice of hyperparameter C has been evaluated on the validation dataset:



As seen in the table, C = 1, provides the most accurate results.

* A higher C overfits the data, whereas a lower C does provide create enough model complexity to fit the high-dimensional data

Next steps:

* Increase the size of training, validation and test datasets
* Visualize prediction
  + Visualize the predicted labels by colored plots of trajectories with labels: ‚cross‘, ‚no cross‘
  + Visualize the collision zone
* SVM cannot catch the time-dependency of the input data. RNN is built for this purpose. Therefore, we predict higher accuracies with RNN. Next step is to test a RNN implementation.