Support Vector Machines:

Advantages:

* The weights in kernel based approaches such as SVM’s are often not a very good proxy of feature importance. The advantage of kernel methods is that you are able to capture non-linear relations between variables by projecting the features into kernel space. On the other hand, just looking at the weights as feature importance does not do justice to the feature interaction.
* SVM’s are very good when we have no idea on the data.
* Works well with even unstructured and semi structured data like text, Images and trees.
* The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
* Unlike in neural networks, SVM is not solved for local optima.
* It scales relatively well to high dimensional data.
* SVM models have generalization in practice, the risk of over-fitting is less in SVM.
* SVM is always compared with ANN. When compared to ANN models, SVMs give better results.
* Memory Efficient (As it uses only few points as support vectors)

Disadvantages:

* Choosing a “good” kernel function is not easy.
* Long training time for large datasets.
* Difficult to understand and interpret the final model, variable weights and individual impact.
* Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic.
* The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact
* It also doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping
* SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

Loss:

Hinge Loss

It uses RBF Kernal. Other available kernals are Poly, linear, Sigmoid

Linear SVM:

Linear SVM is good when there is a lot of features (Text Has lot of features)

Training linear kernel is faster

We need to tune only C

Linear Kernel is indeed very well suited for Text Classification. The recommended approach for text classification is to **try a linear kernel first**, because of its advantages.

RBF Kernal:

We need to tune C and Gamma as well

When the number of features are more we should not use RBF Kernal