how SVM works, and how SVM kernels work, your impression of the strengths and weaknesses of SVM?

Find a suitable hyperplane that maximizes the margin of the decision boundary. For different sample distributions, SVMs can be divided into three categories:

The hard interval SVM mainly solves the linearly separable sample distribution, and constructs a "perfect" hyperplane to maximize the interval between its decision boundaries and make the fault tolerance rate stronger. Make sure that all positive class samples are above the decision boundary and negative class samples are below the decision boundary. This class of SVM directly generates linear models.

Soft margin SVM mainly solves the sample distribution that is generally linearly separable but has some noise. In order to effectively accommodate those positive and negative samples that cannot be divided correctly, and allow a certain amount of classification errors, maximizing the interval and slack variables are introduced. Also returns a linear model.

Kernel SVM, as long as the linear inseparable sample distribution is solved, this type of SVM uses the kernel function to map the original sample space to a high-dimensional space, and then performs linear segmentation.

If the sample n and the feature m are large, and the feature m>n, the spatial dimension of the Gaussian kernel function after mapping is higher, more complex, and easy to overfit. At this time, the disadvantages of using the Gaussian kernel function outweigh the advantages. It would be better to use a linear kernel.

If the sample n is of general size and the feature m is small, after the Gaussian kernel function mapping is performed, not only can the original training data be linearly divided in a high-dimensional space, but also there will be no significant computational consumption, so the benefits are greater than Disadvantage.

If the sample n is large, but the feature m is small, it is also difficult to avoid computationally complex problems, so more consideration will be given to the linear kernel.

how Random Forest works, how the other 2 algorithms you used work compared to the simple decision tree, your impression of the strengths and weaknesses of these ensemble techniques?

Random forest obtains k models by sampling k training sets from samples. Each model is a separate decision tree. The classification results are obtained by voting among the k models. Random forest does not need to do feature selection, judge the importance of features, judge the interaction between different features and balance errors. But it is easy to overfit.

Adaboost after creating the first decision tree, the performance of each tree is used to weight the attention of each training instance for the next tree created. Hard-to-predict training data is given more weight, while easy-to-predict instances are given less weight. Models are created sequentially, one after the other, and each model updates weights on the training instance that affect the learning performed by the next tree in the sequence. After all trees are built, predictions are made on new data and the performance of each tree is weighted according to the accuracy of the training data. AdaBoost treats different classification algorithms as weak classifiers, fully considering the weight of each classifier. However, the data imbalance leads to a decrease in classification accuracy, and the training process is very time-consuming.

XGBoost draws on the algorithm of random forest, samples samples and features, and reduces the amount of calculation while reducing over-fitting. Use a normalized regular term to make the

trained model less prone to overfitting. Parallel computing is performed by pre-ordering the features, saving the structure, and reusing the structure in subsequent computations