According to the the above result, when the number of estimators is equal to 5 and 10, the accuracy is the highest. I set up the n_estimators to be 10 and its accuracy is 100% on the test set. I picked 10 based on speed of convergence and generalisation. 10 is an intermediate value comparing the other one which has the highest accuracy too. The more the number of boosting stages can make the model more complex and consume too much to the computer, it is not good to the complexity(big O) and speed of convergence. The model may also have overfitting problem.If I choose a too small maximum depth and the number of tree, the model may not have good robustness and generalisation when a large amount data input to the model.

Question 5: Analysis

1. Explain why you had to split the dataset into train and test sets?

We train machine learning models to use the existing data to predict unknown data. Generally, we call the model's ability to predict unknown data known as the generalization ability. In order to evaluate the generalization ability of a model, we usually divide the data into a training set and a test set. The training set is used to train the model, and the test set is used to evaluate the model's generalization ability. We can use the data from the training set to train the model, and then use the error on the test set as the generalized error of the final model in dealing with real-world scenarios. With the test set, we want to verify the final effect of the model. Just calculate the error on the test set for the trained model, and we can consider this error to be an approximation of the generalized error. We only need to let our trained model The minimum error on the test set is sufficient. Thus, we had to split the dataset into train and test sets

1. Explain why when finding the best parameters for KNN you didn’t evaluate directly on the test set and had to use a validation test.

In machine learning, when developing a model, it is always necessary to adjust the parameters of the model, such as changing parameters for KNN to get better accuracy. This adjustment process needs to provide a feedback signal on the trained model by validation test data. Modify the model and parameters is the role of the validation set, which will also cause information leakage of the validation set. The more feedback, the more information is leaked, that is, the model understands the validation set more clearly, which will eventually cause the model to overfit on the validation set. At this time, you need a dataset that is completely new to the model,so the test set is used to measure the quality of the model.

1. What was the effect of changing k for KNN. Was the accuracy always affected the same way with an increase of k? Why do you think this happened? #What was the effect of changing k for KNN

If the value of K is changed to be small, the model has high complexity and is prone to overfitting. The learning estimation error will increase, and the prediction result is very sensitive to the neighboring instance points. A large K value can reduce the estimation error of learning, but the approximate error of learning will increase, and training examples that are far away from the input instance will also affect the prediction, which will cause prediction errors, and the complexity of the model will decrease as the value of k increases. If k is too small, the classification result is easily affected by noise points; if k is too large, the neighbors may contain too many points of other categories. Overal, the changing of K value in KNN is very important for the classification result. The K value is too small and the model is too complicated. The value of K is too large, resulting in fuzzy classification. Some peole use Cross Validation to choose k value, some use Bayes, and others use bootstrap.

Was the accuracy always affected the same way with an increase of k?

No.According to the above results, the testing accuracy didn't always increase the same way with an increases of k. At the beginning, the accuracy was at the peakn and then it fell as k increased.For some datasets, when the k neighboring points is large enough, the accuracy will naturally tends to the expected value of the sample. Ouer case is the same.

Why do you think this happened?

Because if k is small, the model would be more complex with high variance and low bias. It could lead to overfitting situation. But when k is too large, the model would be simple with high bias and low variance, it might cause underfitting problem. When k is increasing to be too large, the neighbors may contain too many points of other categories.So the model would not make sense and the accuracy will naturally tends to a fixed value.

1. What was the relative effect of changing the max depths for decision tree, random forests, and gradient boosting? Explain the reason for this.

The max depth defines the maximum depth that the tree is allowed to grow up.When the max depth is being larger, it will be better to fit data for the model, but it may also cause overfitting. Because the the tree is deeper, there will be more complex the decision rules. The tree works from nodes of if / else problems and get results. The deeper the tree, the more it splits, and the better it can capture information about the data. And The decision tree model is prone to produce an overly complex model when the max depth is being larger, and such a model will have poor generalization performance on data. But if the max depth is too small,it may cause underfit problem to the model and increase teating error.

1. Comment on the effect of the number of estimators for Gradient Tree Boosting and what was the relative effect performance of gradient boosting compared with random forest. Explain the reason for this.

The number of estimators is the number of boosting stages to perform.It means the number of iterations. In general, the number of estimators is too small, and the model will be easy to underfit. If n_estimators is too large, there is easy to overfit. Usually, a moderate value is selected. The default is 100 in sklearn. Because the Gradient Tree Boosting method is not just the simple application of ensemble ideas, but also the learning of residuals.

Random forest is an algorithm based on decision tree, it just uses the idea of integration to improve the classification performance of single decision tree. The main feature is that it is not easy to fall into overfitting due to the random selection of samples and features. The main steps of the random forest algorithm are: use Bootstrap to randomly select n samples from the sample set, and randomly select K attributes from all the attributes, and select the best segmentation attribute as the node to establish a classifier (CART, SVM, etc. Repeat the above m times That is, m classifiers are established, and the voting results are used to determine which type of data. The gradient tree boosting is also a decision tree model based on integrated ideas. The algorithm is to generate a new tree for each iteration(Boosting stage). And then to calculate the loss function at each training sample set, and then generate a new decision tree through the greedy strategy. Calculate the predicted value corresponding to each leaf node, and use the newly generated decision tree added to the model. The most essential difference between Gradient Tree Boosting and random forest is that each tree in Gradient Tree Boosting learns the residuals of the sum of the conclusions of all the trees, and the residual is the true value minus the predicted value.

The detailed reason are: 1: Random forests vote the results of multiple decision trees to get the final result. The training results of different trees have not been further optimized and improved, which is called the bagging algorithm. The boosting algorithm is to build a weak learner at each step of the iteration to make up for the deficiencies of the original model. 2:Gradient Tree Boosting uses the boosting algorithm, and builds a weak learner at each step of the iteration to make up for the lack of the original model. Gradient Tree Boosting is to build a learner along the direction of gradient descent through each iteration. And by setting different loss functions, it can handle various learning tasks (multi-classification, regression, etc.).

1. What does the parameter C define in the SVM classifier? What effect did you observe and why do you think this happened?

C is the regularization parameter.The parameter C defines that how much we want to avoid misclassifying for each training set.It controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights.

I observed when c was small the accuracy was higher than when it was larger in Queation 3. When c was equal to and larger than 20, The accuracy was lower and treading to be consistancy obviously.

I think this is because C is a factor that regulates the interval and accuracy. The larger the value of C, the more unwilling to abandon those outliers; the smaller the value of c, the less attention is paid to those outliers. The larger c indicates that the error cannot be tolerated and the over-fitting is easy. The C is too small, it is to underfit easily,the classifier would look for a larger-margin separating hyperplane,even if that hyperplane misclassifies more points. Whatever C is too large or too small, there will be poor generalization ability. When C approaches infinity, the problem is that samples with classification errors are not allowed, then this is a hard-margin SVM problem. (overfitting) When C approaches 0, we no longer care about whether the classification is correct, but only the larger the interval, the better, then we will not get a meaningful solution and the algorithm will not converge. (Underfitting)