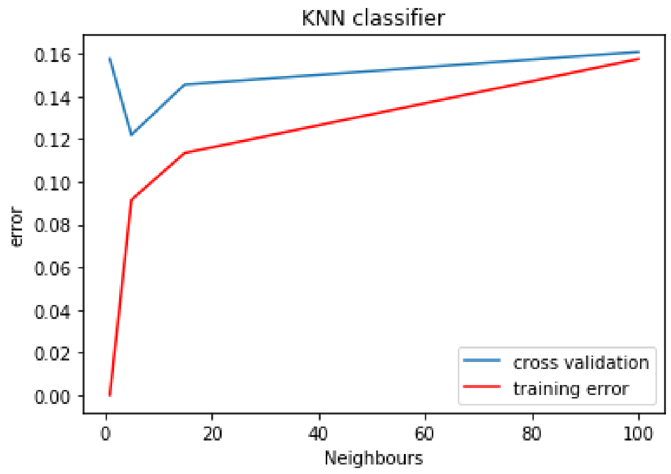
Huilin Tong, ID:1261574

# Project 2

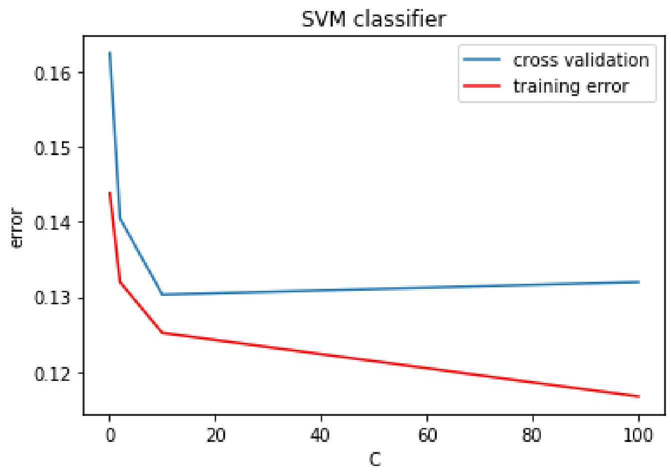
## Titanic

Used KNN, SVM and Random Forest classifier.

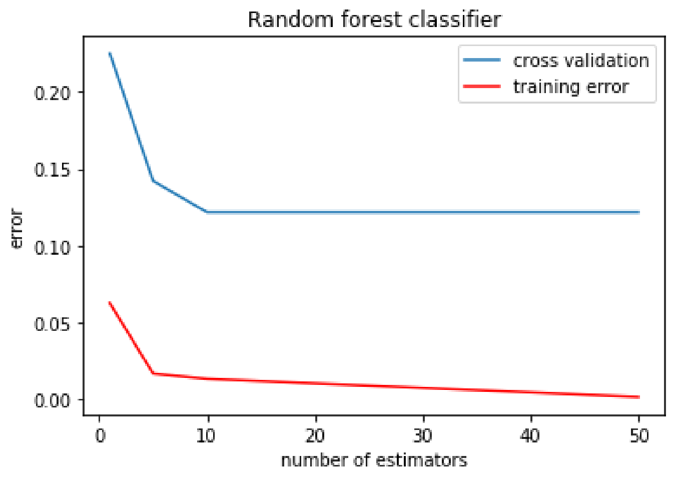
(a) For KNN classifier, chose 1, 5, 15, 100 as it's hyper-parameter $K$. The classifier is to find the $K^{th}$ nearest neighbors. A small value for $K$ provides the most flexible fit, which will have low bias but high variance. Larger $K$ means smaller model space. Result from plot shows when $K = 5$ can get the minimum error.



(b) For SVM classifier, change its' hyper-parameter $C$. A low $C$ makes the decision surface smooth, while a high $C$ aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors. I used 0.1, 2, 10, 100 as the value of $C$. The result shows when $C = 10$, you can get the best accuracy.



(c) For random forest classifier, the hyper-parameter n estimators reflects the number of trees in the forest. The higher the number the better, but it is also add more computational complexity and will take longer for running the code. $n = 1, 5, 10, 50$, 50 is the best.



(d) 95% confidence interval is estimated from normal distribution. $[\hat{p} - 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{N-1}}, \hat{p} + 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{N-1}}]$. My final classifier is SVM with hyper-parameter $C = 100$. Accuracy: $0.87(+/-0.06)$. So the accuracy is among $[0.81, 0.93]$, satisfies the requirement of the task.

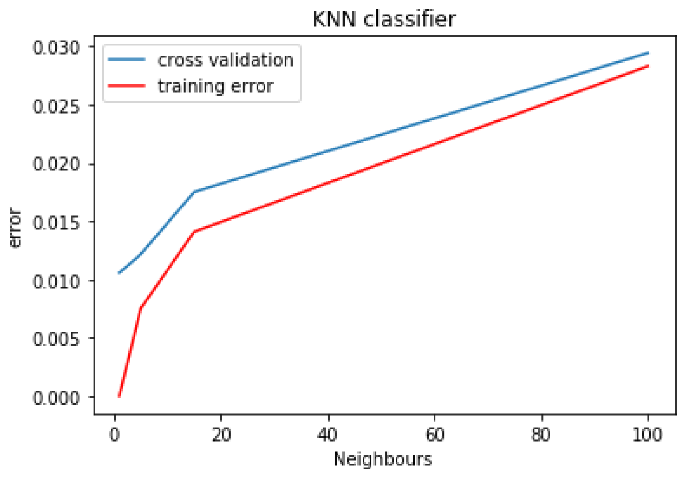
Also did normalization and feature selection for this titanic dataset.

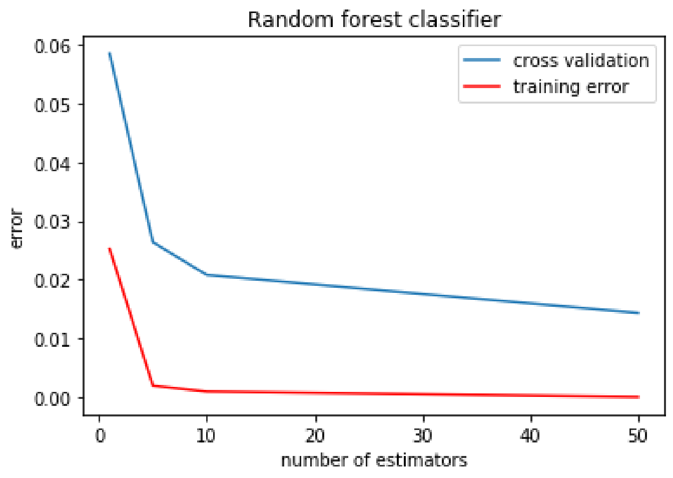
For normalization, choose to scaling features to lie between a given minimum and maximum value, between zero and one, to scaled to unit size. When some features have different ranges in their values, this will affect negatively algorithms because the feature with bigger values will take more influence. Need normalization to reduce the impact of dimension on model.

For feature selection, univariate feature selection works by selecting the best features based on univariate statistical tests. I used *SelectKBest* to removes all but the $k$ highest scoring features. In titanic data, there are 9 features in total. But the result turns out when delete some features or in other words when reduce $k$, accuracy decreases. So still keeps all these 9 features.

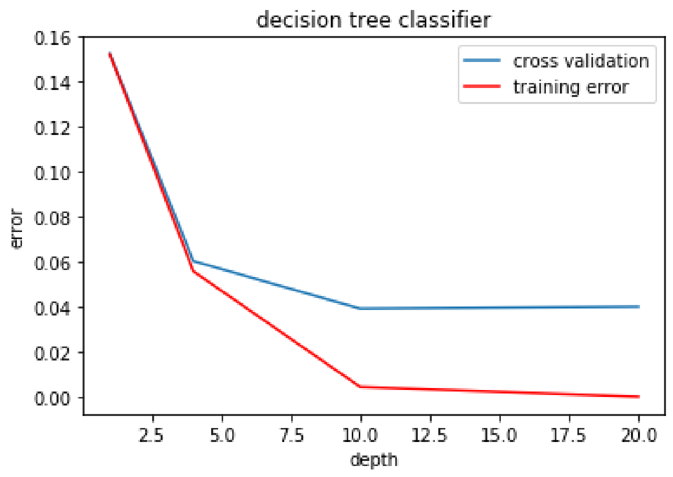
**Mnist Binary**
Used Decision Tree, KNN and Random Forest classifier.

(a) For KNN and Random Forest classifiers, it's totally the same as the Titanic. Plot of training errors and validation errors against different settings of hyper-parameters are as follows





(b) For decision tree classifier, it's hyper-parameter is the max depth of tree. We want to create a situation where almost all features have been given a chance to participate in becoming a decision node, but not too much to avoid overfitting problem. I tried $1, 4, 10, 20$ as its depth. Turned out when maximum tree depth was reduced to 10, and accuracy is best.



(c) 95% confidence interval is estimated from normal distribution. $[\hat{p} - 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{N-1}}, \hat{p} + 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{N-1}}]$. According this formula, my final classifier is random forest classifier with hyper-parameter $n - estimators = 50$. Accuracy: $0.99(+/-0.01)$. So the accuracy is among $[0.98, 1.00]$, satisfies the requirement of the task.

While KNN classifier is actually the best when $K = 1$, with accuracy $0.99(+/-0.00)$. Its time and space complexity is too large to meet the requirement of our task. So still choose the random forest classifier.