1. **Introduction**
2. **Description and motivation**

From features like Clump Thickness, Uniformity of Cell Size/Shape, Marginal Adhesion, etc I want to find the best model that helps classify whether the cancer is benign (noncancerous) or malignant (cancerous). The prediction is important because determining where the cancer is dangerous or not is critical for the patient. False prediction can lead to serious consequence (for example, the patient may suffer from huge anxiety if we output the result that the patient has malignant while she does not have it). With task requiring huge accuracy like this, finding the right classifier is very important.

1. **Factor to determine the best classifier**

Accuracy: the model needs to perform well on training data, and it also need to perform as good on the data it has not seen before – validation data (not underfit and overfit).

Computational time: the faster the model runs to train the data / make prediction, the better the model is

1. **Methods**
2. **Data preprocessing**

I used pandas to read the data, and I discarded the meaningless id column. Then, I dropped the columns that contains ‘?’ in one of the feature values. The final data contains 683 entries, with 9 features for the input and 1 class (of value 0 for benign and 1 for malignant). I map the output from 2 and 4 to 0 and 1 since it is more intuitive for machine learning/deep learning models (0 signals noncancerous and 1 signals cancerous).

1. **Code packages/library used**

I used python with the following packages:

* Pandas and NumPy to preprocess the data
* Scikit-learn to build and train the models that are not Neural Network (k-nearest neighbors, Decision tree)
* Tensorflow to build and train the Neural Network models
* Matplotlib to plot the result of cross-validation

1. **Result**

For each model, I perform 10-fold cross-validation and graph the average score (accuracy) of each model with specific hyperparameter on the validation set.

See my READme.md for instructions on how to install the appropriate dependencies and replicate my result.

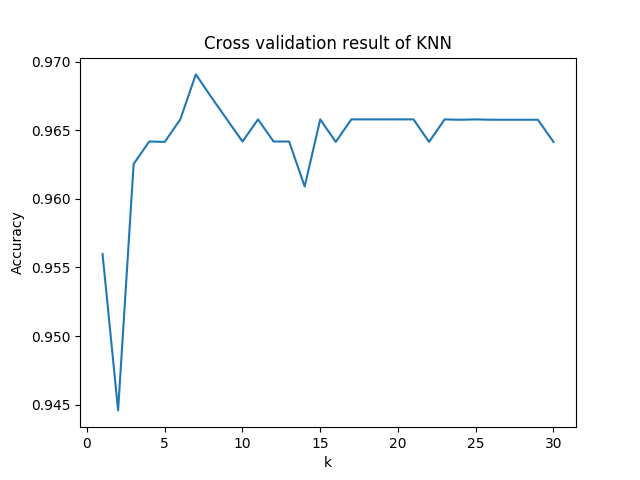
1. **k-nearest neighbors**

k-nearest neighbors is a machine learning algorithm that based on the similarity defined by distance between data points to make prediction. When encounter an unseen observation, kNN will perform majority vote (or get the average for regression problem) on k seen observations that is closest to the unseen one to determine the class value of it.

Advantage: training is very fast, and how the result gets predicted is very intuitive.

Disadvantage: finding k nearest points is very time consuming

I performed cross-validation on k values ranging from 1 to 30, and this is the result:



Best performance achieved at k = 7 (with mean accuracy on validation folds is 0.9691)

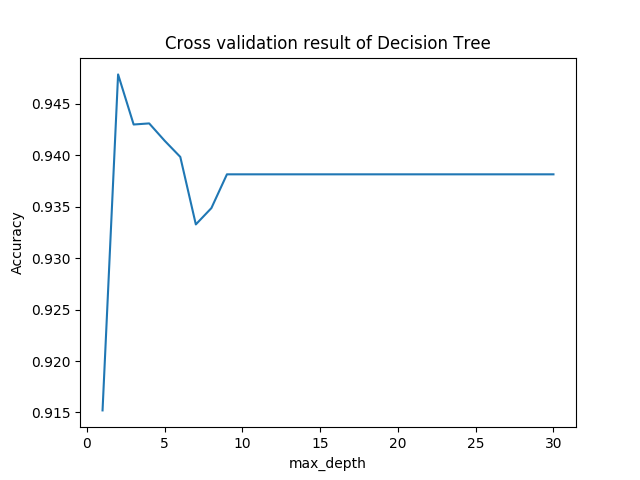
1. **Decision tree**

Decision tree is a machine learning algorithm that has a root node followed by a series of decision nodes that try to split the data and maximize the homogeneity of each node (each node will be as pure as possible). The leaf node will contain the prediction.

Advantage: work for both numerical and categorical data, easy to explain and produce more intuitive result

Disadvantage: Easy to get overfitted as it is affected by noise in the data.

I performed cross-validation on max depth values of the tree ranging from 1 to 30, and this is the result:



Best max depth of the decision tree is at 2 (with mean accuracy on validation folds is 0.9478)

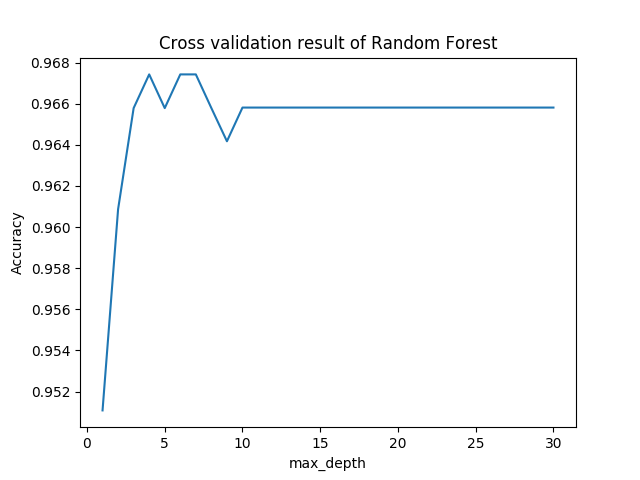
1. **Random Forest**

Random Forest is a bagging machine learning algorithm that contains large number of individual decision trees, where each tree will predict a specific class label and then major voting will decide the vote for final prediction. The idea is for all the individual tree with low performance to work together to output a strong result.

Advantage: avoid overfitting (the problem of single decision tree), being able to combine result of different decision trees to have a stronger model. It also can solve both classification and regression problem.

Disadvantage: The model created is too complicated that there is no intuitive reasoning behind. The training time is also huge since there is a lot of tree (in my case it is 100 trees).

I performed cross-validation on max depth values of each tree ranging from 1 to 30, and this is the result:



Best max depth of the trees in the random forest is at 4 (with mean accuracy on validation folds is 0.9674)

1. **SVM using the polynomial kernel**

Support Vector Machine is a machine learning model that works on both classification and regression problems. Intuitively, the model create a plane that separate the data into classes by maximizing the margin of the data points toward the plane.

The model would only work on cases when the data is not linearly separable. We can perform data preprocessing and transform the data into higher dimension – for example (x) will become (x, x2).

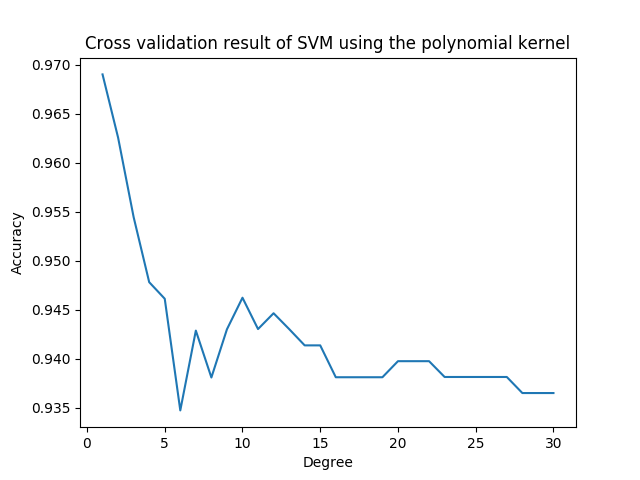
The kernel function will alleviate the transformation process – to make the computation and storage more efficient.

In case of SVM with polynomial kernel, the kernel will be (a.b + r)d with d is the dimension of the kernel.

Advantage: work very well if the right degree is chosen and the data is clearly separated – or the violation is minimal (where we can perform soft margin classification). And it also work well when the dimension is high.

Disadvantage: SVM does not work so well with large data set, and if the degree is too high there is no intuitive explanation for the result.

I performed cross-validation on the degree from 1 to 30, and this is the result:

****

Best degree value of the algorithm is at 1 (with mean accuracy on validation folds is 0.9690).

1. **SVM using the RBF kernel**

Similar to polynomial kernel, the kernel function will be e-y\*(a-b)^2, which project the data onto infinite dimension.

Advantage: It project data into infinite dimension without having to transform and store the huge amount of data.

Disadvantage:

1. **Deep neural network with sigmoid activation**

helo

1. **Deep neural network with ReLU activation**

helo