**Data acquisition**

**Breast Cancer (Diagnostic) Dataset**

**I used the same dataset as my chapter2 analysis.**

**Origin**: From Kaggle and UCI machine learning repository

<https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>

**Reasons:**

* Interested in Bioinformatics and health-related area
* High accuracy in prediction is important for early cancer diagnosis
* Interested in how similar tumors are and if the similarity is closely related to cancer diagnosis

**Subset**:

I did feature selection and removed some columns, including the information about the standard error of tumor’s attributes and patients’ id. (I think the mean and the standard error are kind of redundant, so removing one of the measures should be fine.) After feature selection, we have 11 columns with 1 response variable and 10 explanatory variables about the mean of each attribute.

**Dimension**:

569 rows and 11 columns

**Features:**

| Features | Data type | Description | Non-Null Count |
| --- | --- | --- | --- |
| Diagnosis | object | (M = malignant, B = benign) | 569 Non-Null |
| radius\_mean | float64 | mean of distances from the center to points on the perimeter | 569 Non-Null |
| texture\_mean | float64 |  | 569 Non-Null |
| perimeter\_mean | float64 |  | 569 Non-Null |
| area\_mean | float64 |  | 569 Non-Null |
| smoothness\_mean | float64 | local variation in radius lengths | 569 Non-Null |
| compactness\_mean | float64 | perimeter^2 / area - 1.0 | 569 Non-Null |
| concavity\_mean | float64 | The severity of concave portions of the contour | 569 Non-Null |
| concave points\_mean | float64 | number of concave portions of the contour | 569 Non-Null |
| symmetry\_mean | float64 |  | 569 Non-Null |
| fractal\_dimension\_mean | float64 | "coastline approximation" - 1 | 569 Non-Null |

**Tasks:**

* Understand decision tree splitting algorithm by implementing the first split(lots of Gini and entropy calculation) without using any package other than simple math
* Implement decision tree model with Sklearn package and fully understand each parameter for the model
* Understand model overfitting and apply a train-test split before model fitting
* Decision tree plotting
* Prediction of diagnosis with breast cancer and achieve 90.6% for accuracy from testing dataset
* Implement KNN and try different values for hyperparameter
* Compare the accuracy between the decision tree and KNN

**Code development (Detailed comments are in jupyter notebook)**

**Decision tree:**

I first did the root node splitting assignment and this exercise definitely help me better understand the algorithm of building a decision tree. [Link to my notebook and code.](https://colab.research.google.com/drive/1YsTV9IXMKD4FxX2pnIR_Emuc2rQT-6P4) I made very detailed comments on my assignment Jupyter notebook describing what I did for each step. I compare the combined entropy splitting on each feature and finally conclude that our first split should be on F23.

I also output the combined Gini/entropy and some related child information to a [.csv file](https://drive.google.com/drive/folders/1dMwNwhivgqrkg8IjKgzEhuTLGFJGgwM_).

Bayes Theorem:

I also practice Bayes Theorem and implement it to do the prediction of randomly generated data. [Link to my notebook and code](https://colab.research.google.com/drive/1PeSpUJQM40Un4b95teP7TXuS751-9pm8). I calculated the prior probability and unique distribution of different classes. I then make predictions of testing dataset by comparing the posterior probability of being in two classes. The accuracy is around 94.3%.

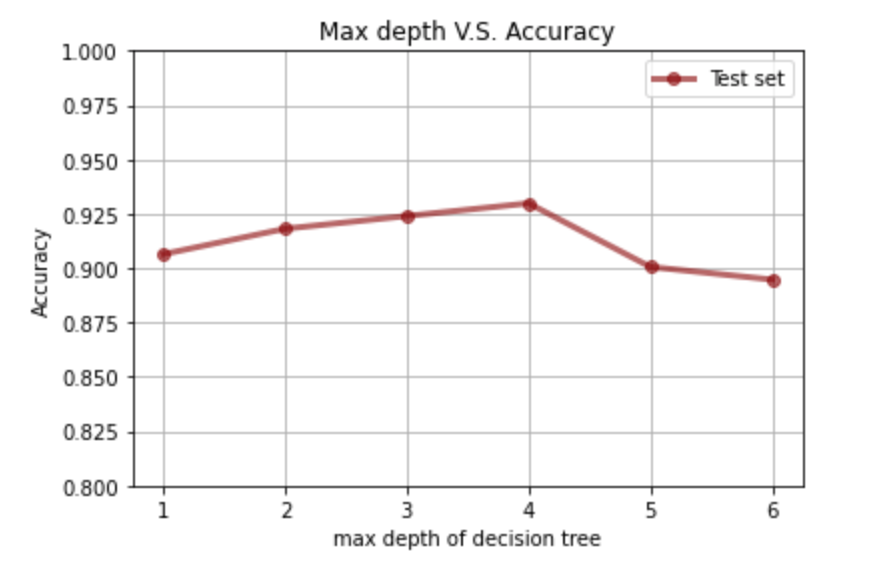
**Data analysis and package use**

Packages: NumPy, Pandas, Matplotlib, Sklearn(KNeighborsClassifier,

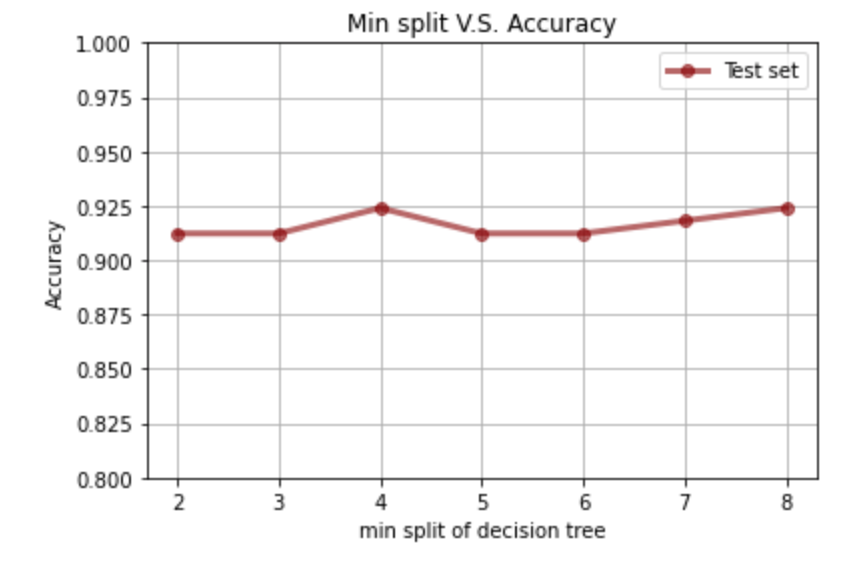
DecisionTreeClassifier,train\_test\_split)

For the decision tree, I first change the response variable to integers because Sklearn does not support strings for the response variable. Before the modeling training, I split my entire dataset into a training set and a testing set, containing 70% and 30% of observations respectively. Then, I build a decision tree classifier using packages function. I first try a default tree (clf\_simple = DecisionTreeClassifier()). However, the prediction accuracy is only around 89%, giving us some room for improvement. Then, I customize tree model parameters such as splitting criteria and max depth by asking user input. My model’s accuracy gets much higher than the previous one and achieves 93%. I plot these two trees for better visualization.

10/10 Updated plots of comparing accuracy with hyperparameters:



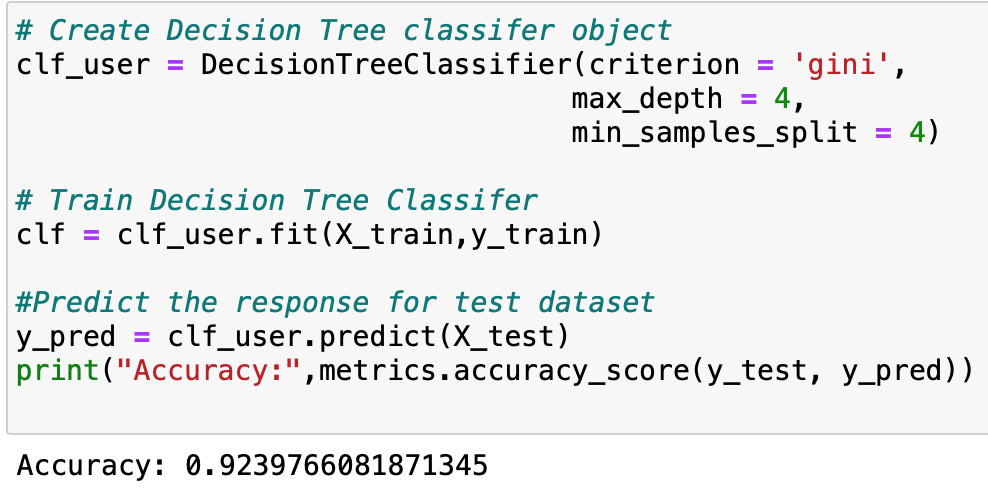
According to the plot, the decision tree model works best when max depth = 4.



According to the plot, the decision tree model works best when min split = 4(with max depth=4).

Putting together **max depth = 4 and min split = 4**

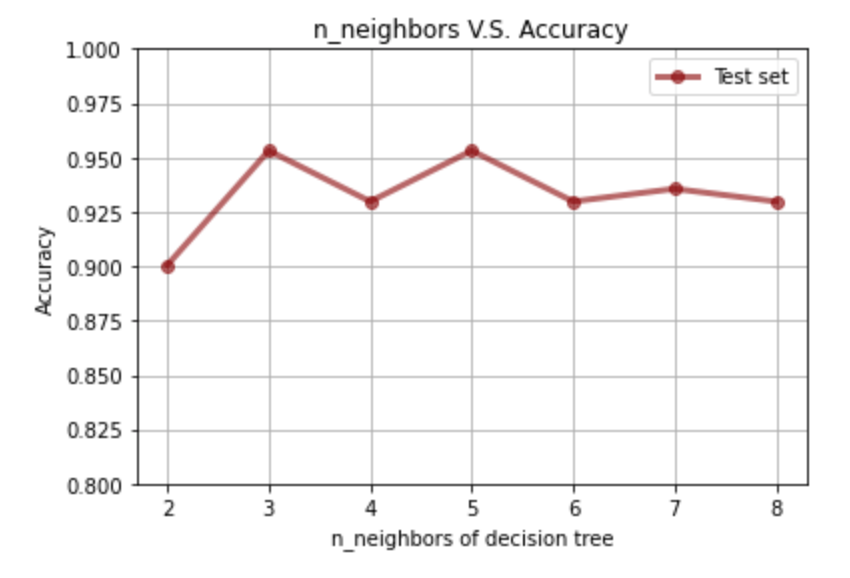
The accuracy is 92.4%.



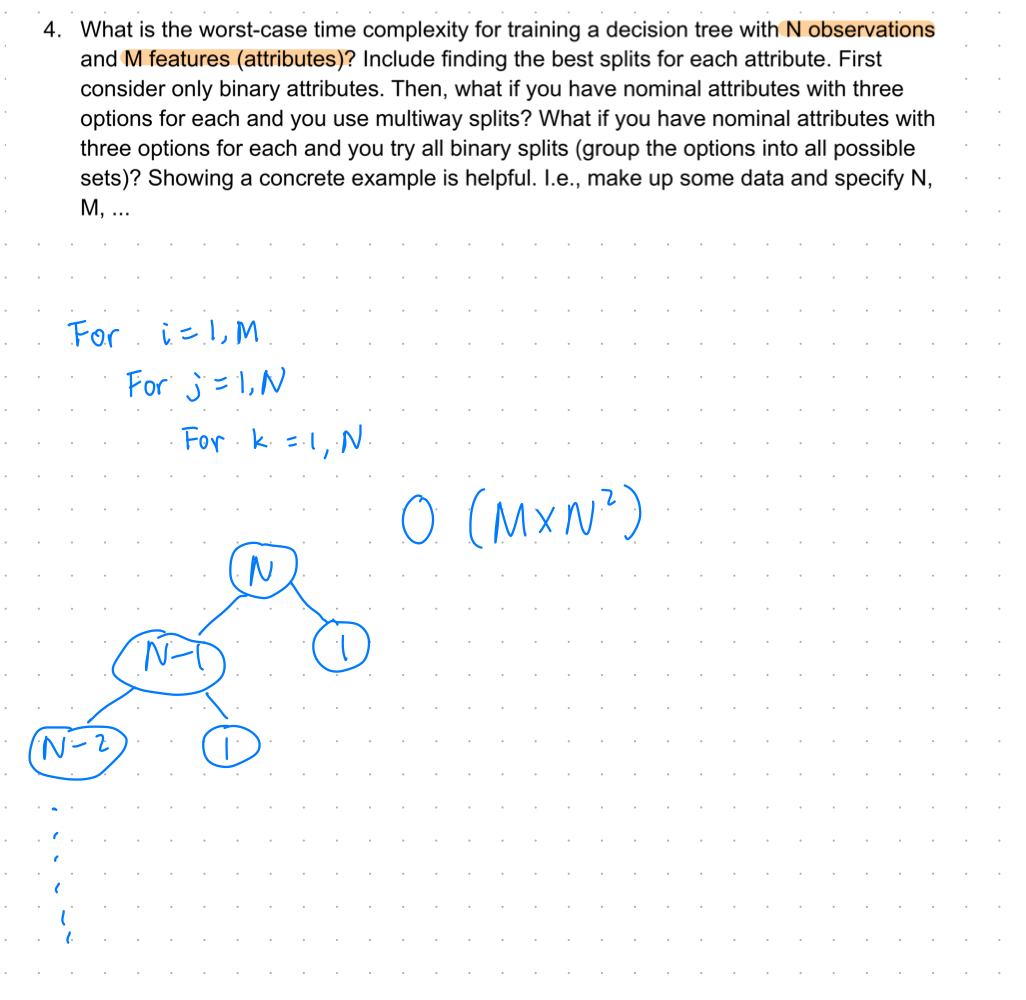
Similarly, I implement **KNN** with parameter (n\_neighbors=5). The prediction result shows that this model only misclassifies 8 observations in the testing set, which is pretty good.

I try quite a few parameters for the model and it is usually true the KNN performs better than the decision tree. This conclusion only applies to this cancer dataset only.

I also did hyperparameters tuning and found out that **n\_neighbors = 3/5** are good choices.



**Theory**

****

1. What is the worst-case time complexity for training a decision tree with N observations and M features (attributes)?

The worst-case time complexity depends on the data type of the features. If Ms are all categorical attributes with two levels, the big-o would be O(M^2).

1. Include finding the best splits for each attribute. First, consider only binary attributes. Then, what if you have nominal attributes with three options for each and you use multiway splits? What if you have nominal attributes with three options for each and you try all binary splits (group the options into all possible sets)? Showing a concrete example is helpful. I.e., make up some data and specify N, M, …
2. Discuss normalizing numeric features for building a decision tree. Is this necessary? Does it affect the decision tree that is built?

I think normalization would not have an impact on the decision tree split. That is because scaling will not affect the distribution of points so the number of data points below and above the splitting threshold will stay the same. **Therefore, I think scaling is not necessary for building a decision tree.**

1. How might you determine an optimum split point for two features at the same decision node, i.e., look for interacting features?

If the features are binary, the combined Gini and the information gain will be calculated when we put observations with level1 as left child and level2 as right child. We do the same thing for the other feature and compare and information gain between feature 1 and feature 2. We would choose the feature to split when information gain is larger. If the features are numeric, we need to find the splitting threshold first and then calculate the information gain.

**Student learning summary and self-assessment**

I think I did a great job on the classification chapter. I actually learned the decision tree algorithm in my statistics class before but it only covers the Gini calculation and the R code template for using the package. The python code development in this chapter enables me to know more details about the splitting algorithm and I think the theory exercises also gave me the chance to understand the Big-O better.

I also compare the R code for decision tree and Python code for decision tree implementation. Even though their structures are highly similar and users can customize the parameters for the tree classifier, I still think the python code is more concise and clear.

The Bayes exercise also deepens my understanding. I practicing the Bayes without using packages and the accuracy is pretty high(around 94%). I’m going to read [this article](https://machinelearningmastery.com/bayes-theorem-for-machine-learning/) during the fall break and see if I can do more coding related to this topic.