Assignment Analysis

1. Introduction

In this project, I have two datasets. one is the benign or malignant Breast Cancer, the other one is the iris flower. The reason I chose these two datasets is because one of them is a yes-or-no problem and the other is a multiple class problem. In one way, running different types of datasets on the same machine learning model can help compare the suitability of the model for different types of problems. In the other way, using the same dataset to train different models can compare the accuracy and applicability of different models for specific problems.

2. Experiment

In this project, Breast Cancer (Wisconsin) Dataset can be downloaded at https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data. The dataset contains 569 cases and 30 attributes in total. The 30 attributes are actually three categories, each category contains the same 10 features but were computed in their mean, standard error and worst. Therefore, at this time, I use only the mean value of each case in the model. Before the exploring the dataset, 10 standard error columns and 10 worst columns were dropped.

Iris flower dataset can be downloaded at https://www.kaggle.com/datasets/arshid/iris-flower-dataset/code. The dataset contains a set of 150 records under 5 attributes - Petal Length, Petal Width, Sepal Length, Sepal width and Class (Species). There are 3 species to be categorized (setosa, versicolor, and virginica), each has 50 entries.

(1) Data Inspection.

The Breast Cancer (Wisconsin) dataset has 569 entries.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   diagnosis          569 non-null    object
 1   radius             569 non-null    float64
 2   texture            569 non-null    float64
 3   perimeter          569 non-null    float64
 4   area               569 non-null    float64
 5   smoothness         569 non-null    float64
 6   compactness        569 non-null    float64
 7   concavity          569 non-null    float64
 8   concave_points     569 non-null    float64
 9   symmetry           569 non-null    float64
 10  fractal_dimension  569 non-null    float64
dtypes: float64(10), object(1)
memory usage: 49.0+ KB
```

```
df.describe()
```

	radius	texture	perimeter	area	smoothness	compactness	concavity	concave_points	symmetry	fractal_dimension
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440

The iris flower dataset has 150 entries.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

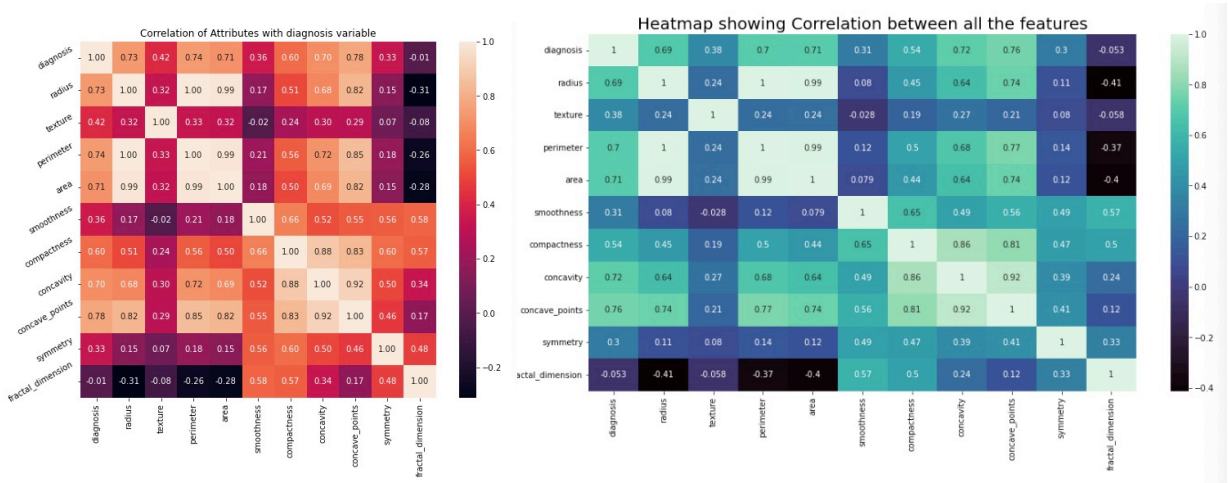
(2) Data Cleaning

There are no null values in the dataset. For the outliers, since the models such as KNN, decision tree are robust to outliers, we do not drop outliers this time. But support vector machines and gradient boosting are very sensitive to outliers, therefore, I will clean data by dropping outliers before the model training.

(3) Data Exploratory

The left figure below is the correlation efficient computed from the full set data. The right figure is the computed after dropping outliers.

The left correlation efficient fig shows that diagnosis is positively related with concave points (correlation coefficient = 0.78), perimeter (correlation coefficient = 0.74), radius (correlation coefficient = 0.73) and area (correlation coefficient = 0.71). Diagnosis is weakly related with texture (correlation coefficient = 0.42), smoothness (correlation coefficient = 0.36), and symmetry (correlation coefficient = 0.33). Diagnosis is independent with fractal dimension (correlation coefficient = -0.01).



3. Result and Analysis

In this project, five learning algorithms are implemented on the same dataset. They are:

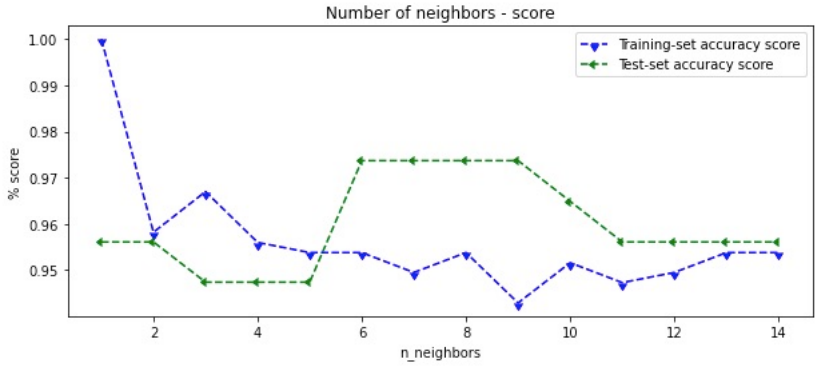
(1) K-nearest neighbors
(2) Decision trees
(3) Support vector machines
(4) Gradient boosting
(5) Neural networks

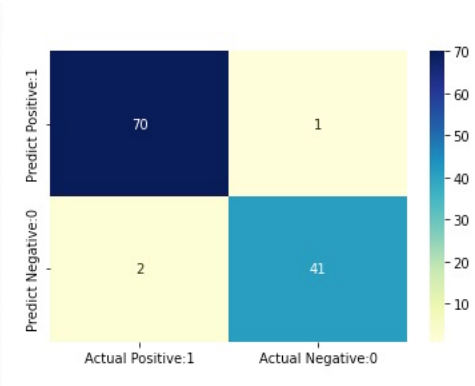
Each of the algorithm are tuned to get the best parameters. The results are as below:

(1) K-nearest neighbors

- For the breast cancer dataset:
  (1) when setting the test size as 20% of the dataset (test_size=0.2), iterations for different neighbors shows that, when K increase, the test accuracy score increase and the performance is improved; we get same accuracy score of 0.9737 with K= 6, 7, 8, 9. If we increase the value of K further, this would result in decreased accuracy. When K=1, it has the highest training set accuracy while lowest test set accuracy, it means the model is overfitting. When K=8, the training set accuracy score (0.9538) is the highest, which means will has the best performance.

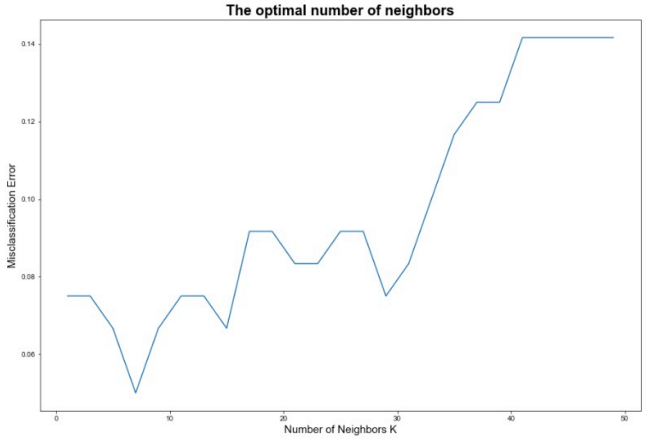
(2) When K=8, the training set score is 0.9538, the test set score is 0.9737. The confusion matrix shows that the classification accuracy is 0.9737, the true positive rate is 0.9722.

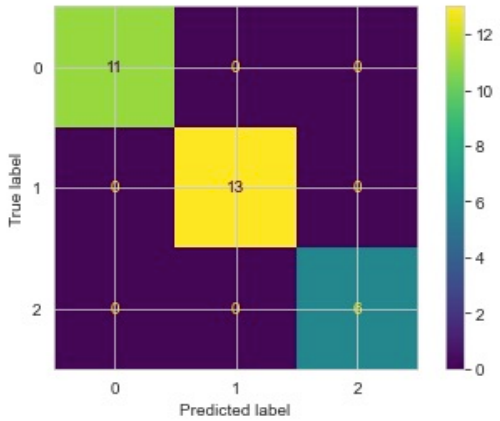




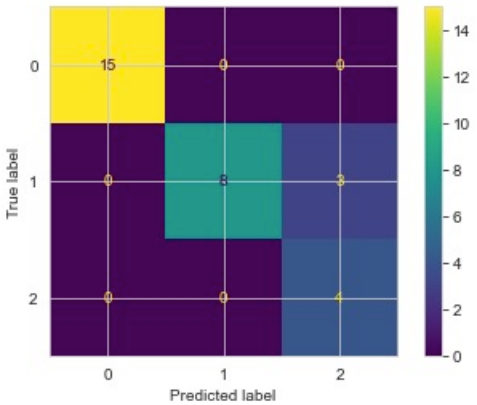
- For the iris flower dataset:

(1) Iteration result shows that, with setting the test size as 20% of the dataset (test_size=0.2), when number of neighbors increase, the misclassification error firstly decrease then keeping increasing. When n_neighbors=7, KNN model has the highest accuracy at this instance. The training accuracy is 0.9667 while the test accuracy is 1.0000.





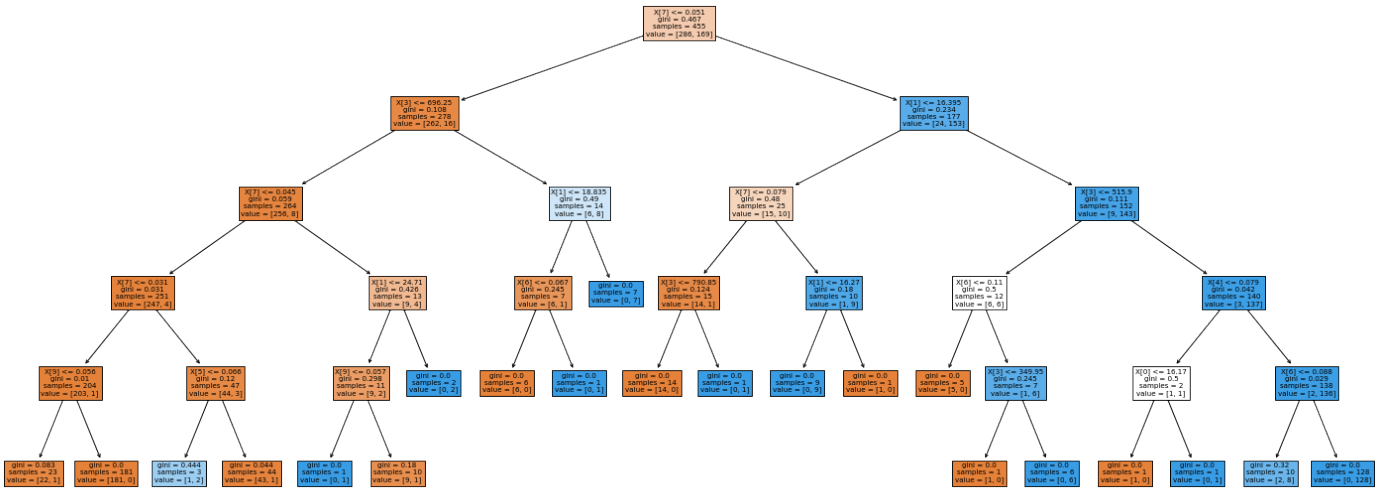
(2) Due to the small size of data set, the situation happens that test accuracy is higher than training accuracy. When I change the random state and re-run my code, the optimal n_neighbors changes to 13, and the traing accuracy is 0.975 while the test accuracy is 0.9000.



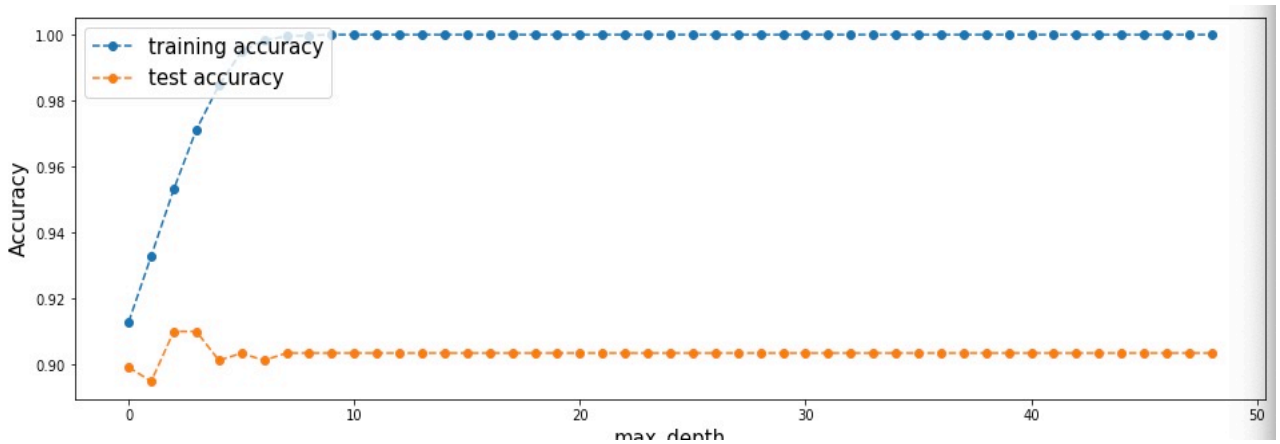
(2) Decision trees

- For the default decision tree:

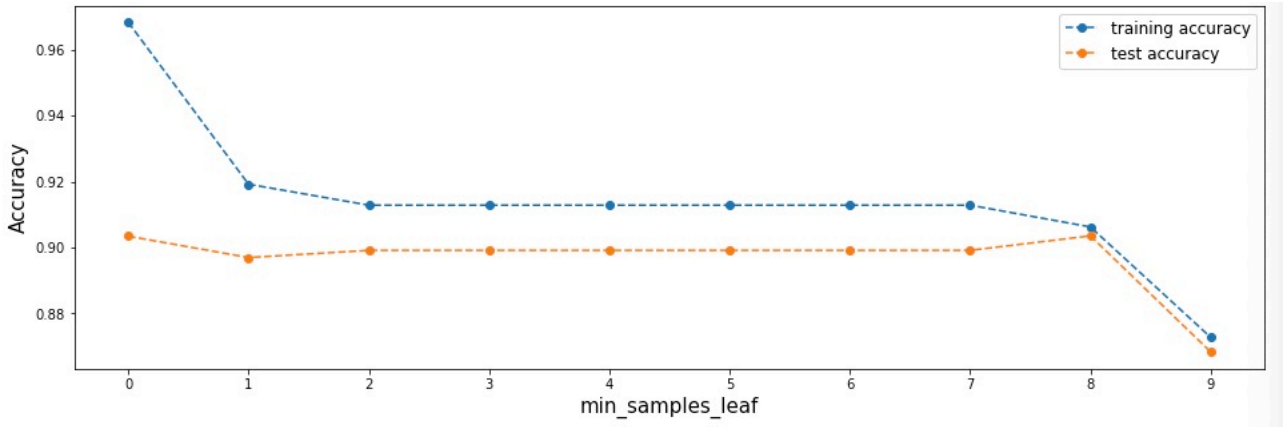
(1) when setting max_depth=5, the accuracy score is 0.9474. It is quite complex since it has 20 leaves.



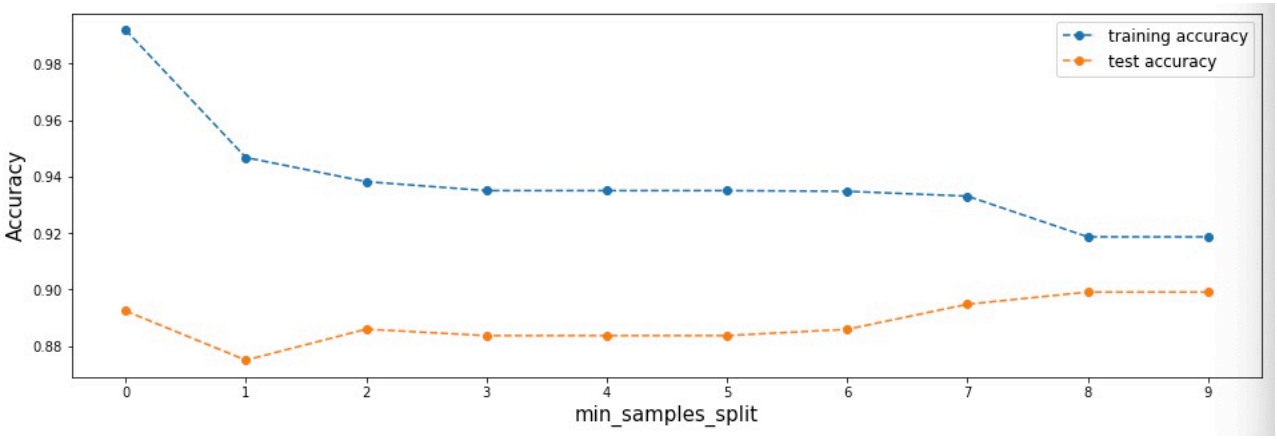
(2) To simplify the decision tree, I firstly tuned the max_depth to find the optimum values. The result shows that when increasing depth, overfitting appears when max_depth=7. Before that, the test accuracy firstly increase then decrease with the depth increasing. When max_depth=2,3, the decision tree has the same test accuracy score (0.9130). We can conclude that when max_depth=3, the model has the greatest performance. The wall time is 2.8s.



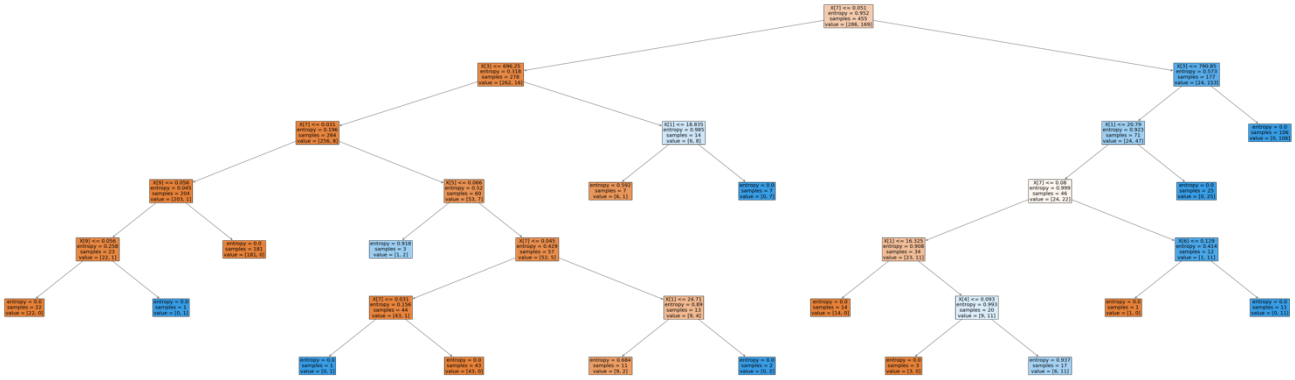
(3) Next, I tuned the min_samples_leaf. Low values of min_sample_leaf tree may be overfitted, therefore, for my project, it is good when the min_sample_leaf=8. The wall time is 447ms.



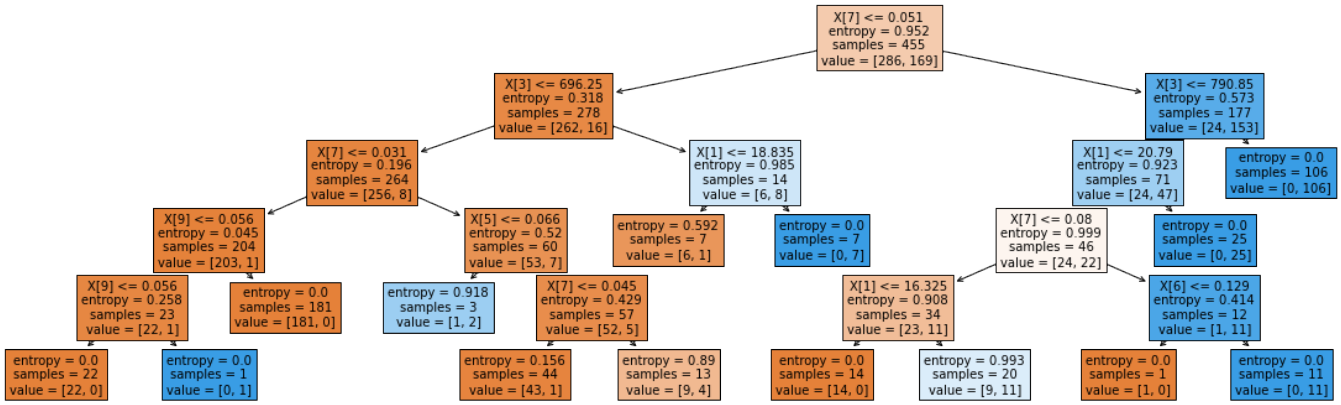
(4) Next, I tuned the min_samples_split. As the min_samples_split decreases, the overfits is decreasing too due to the decreased complexitity of the tree. therefore, it is good when the min_sample_split=8. Low values of min_sample_leaf tree may be overfitted, therefore, for my project, it is good when the min_sample_leaf=8. The wall time is 551ms.



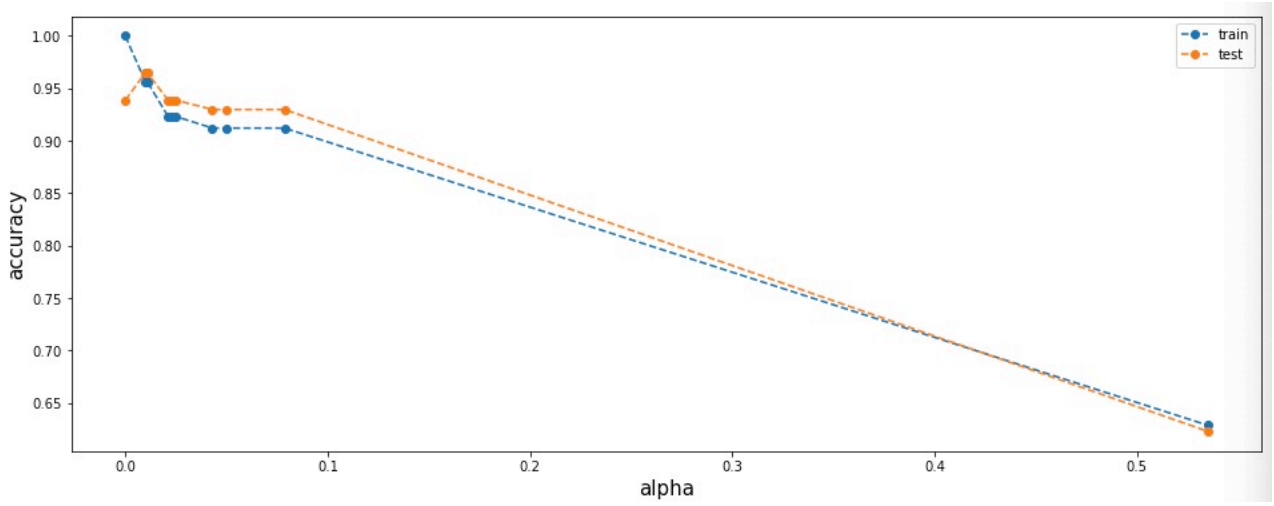
(5) When using GridSearchCV to fine tune multiple optimal hyperparameters together, the result shows that the best estimator appears when max_depth=6, min_sample_leaf=1, min_sample_split=11. It has a mean cross-validated score of 0.9188. The wall time is 1 min 40s. Feeding the optimal hyperparameters (criterion='entropy', max_depth=6, min_samples_split=11) into the decision tree, the tree has an accuracy score of 0.9649, but it is quite complex.



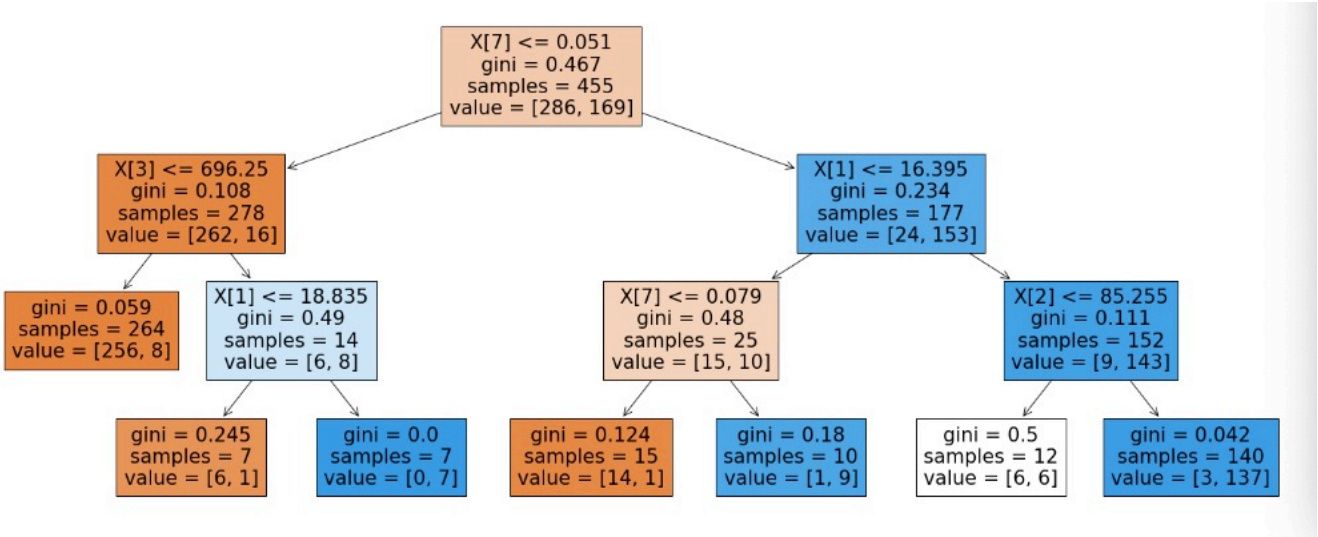
(6) If compromising the accuracy for a simpler decision tree, it shows that when max_depth=5 (accuracy score= 0.9561), I can get a more interpretable decision tree by loosing just 1% accuracy.



(7) Still the tree is not that simple, I apply cost complexity pruning here to gain a smaller but efficient tree. The tools to control the pruning here is the cost-complexity parameter α. When α=1, the tree is left with one single node, while when α=0, the tree grows fully and overfit the data. Tunning the tree by slowly increasing α, the result shows that when α=0.01, the tree has the best accuracy.



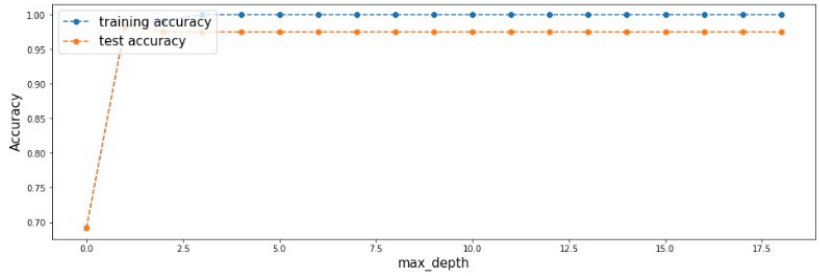
(8) Feeding all the fine tune parameters into the decision tree (criterion='entropy', max_depth=5, min_samples_split=11, ccp_alpha=0.01), I get a final tree with a test accuracy of 0.9649. Returning back to interpreting the tree, it means that the breast cancer checking can be determined mainly by 4 features, they are concavity, radius, texture, and perimeter.

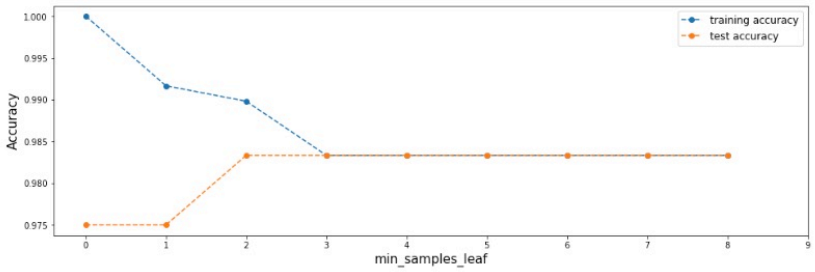


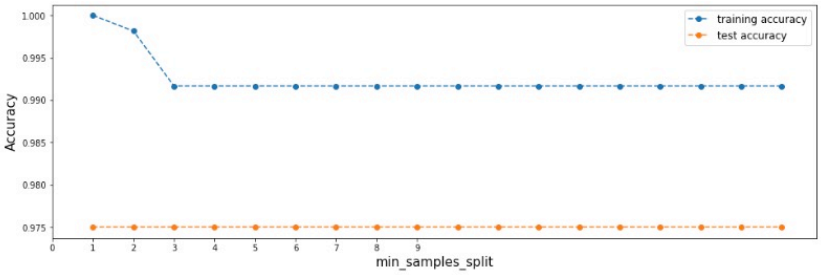
- For the iris flower dataset:

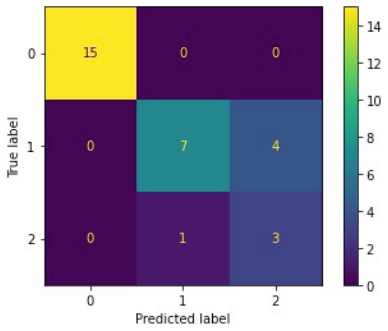
(1) I firstly trained the model by tuning parameters separately and then tuned the model by cross validation by using GridSearchCV. The result shows that when depth=2, minimum split leaves=2, minimum sample leaves=1, the model will have the best performance. The test accuracy is only 0.8333. And the tree is simple and interpretable.

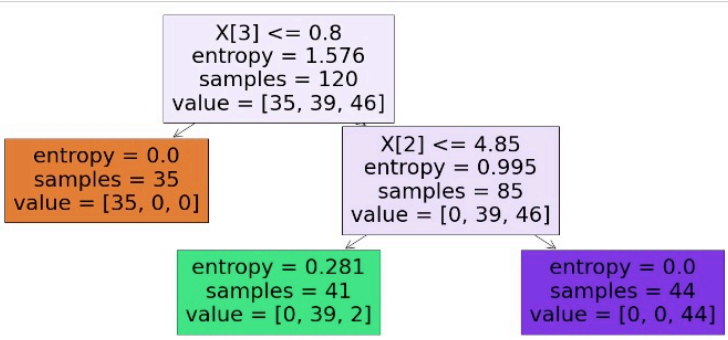
(2) The dataset has limited features and the volume of the dataset is small. Small size definitely affects the accuracy of the model. For features, each time decision tree creates a new split to minimize the entropy or increase the information gaining, less candidates (features) would definitely limit the power of decision tree. Otherwise, I think this model should gaining a higher accuracy.











(3) Support vector machines

- For the breast cancer dataset:

  (1) I compared the results with and without outliers. It's obvious that support vector machines are sensitive to outliers because support vector machine has better data quality do has higher accuracy. I listed the result below.

	Accuracy (Removed outliers)	Accuracy (Including outliers)
SVC with Linear Kernel	0.935 (f1-score=0.94)	0.912
SVC with Radial Basis Kernel	0.935 (f1-score=0.93)	0.877
SVC with Poly Kernel	0.935 (f1-score=0.93)	0.886
SVC with Sigmoid Kernel	0.761	0.404

  (2) It seems that support vector machine with Linear, RBF, and Poly kernel have the better performance when there are 10 features in this project, but due to the greater f1-score in linear kernel, I would say SVM with linear kernel in this project has the best performance. While sigmoid kernel has the worst performance.

- For the iris flower dataset:

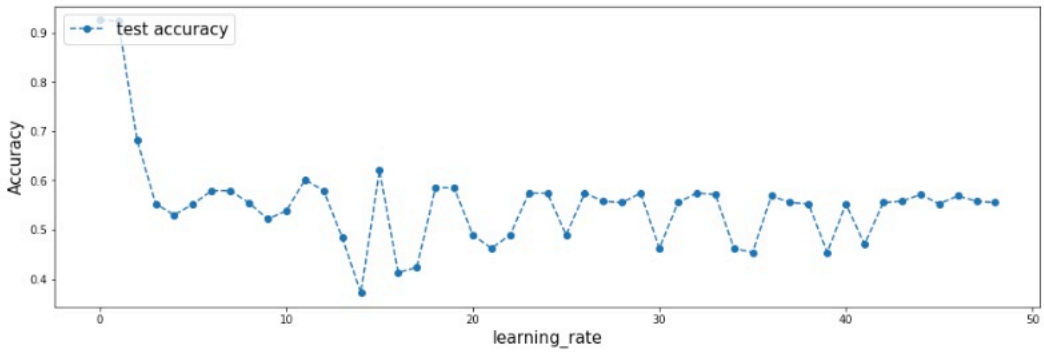
  (1) I tuned the model by swap out different kernels, the result is listed below. Again, SVC with linear, RBF, and Poly kernel have better performance while sigmoid kernel has the worst performance.

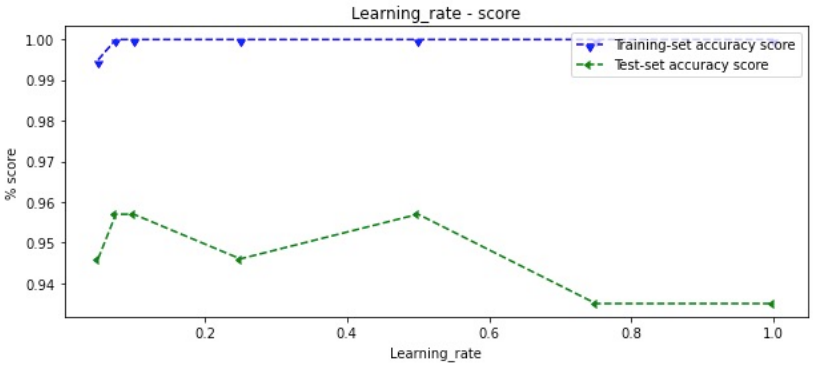
	Accuracy (Removed outliers)
SVC with Linear Kernel	1.000 (f1-score=1.00)
SVC with Radial Basis Kernel	1.000 (f1-score=1.00)
SVC with Poly Kernel	1.000 (f1-score=1.00)
SVC with Sigmoid Kernel	0.300

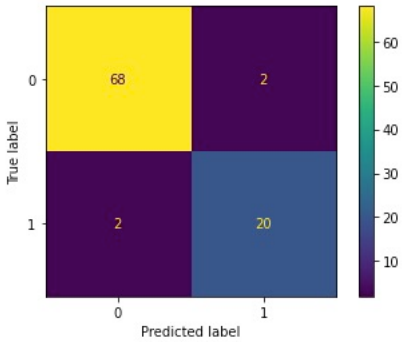
(4) Gradient boosting

- For the breast cancer dataset:

(1) When fine tuning the learning rate, the result (upper figure) shows that gradient boosting behaves better in the range of (0,2), therefore, I choose a small range of learning rate and iterate with a step of 0.025, the result (lower figure) shows that gradient boosting has the best performance (train accuracy =1, test accuracy=0.9565) when learning rate=0.1.





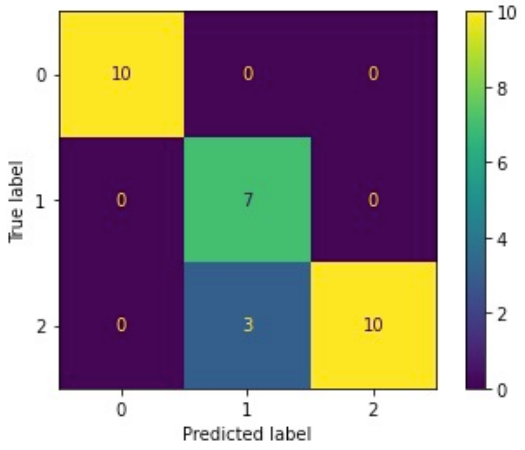


- For the iris flower dataset:

(1) When iterating GridSearchCV to find the optimal learning rate and n_estimators, the result shows that the best parameters is learning rate=0.1, n_estimators=100.

(2) When feeding the result above to fine tune the learning rate in the range of (0.05, 1), the result shows that training accuracy is always 1.0, and the test accuracy is always 0.9. It shows that applying gradient boosting in this project, overfitting appears easily.

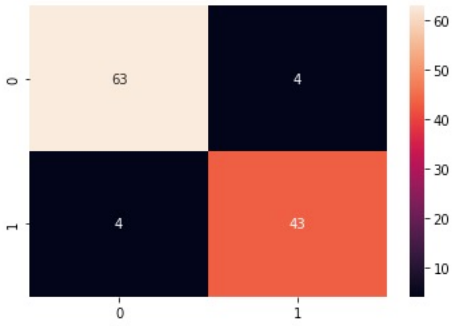
```
Learning Rate : 0.05
 Accuracy rate of training  1.0
Accuracy score of the test : 0.9
Learning Rate : 0.075
 Accuracy rate of training  1.0
Accuracy score of the test : 0.9
Learning Rate : 0.1
 Accuracy rate of training  1.0
Accuracy score of the test : 0.9
Learning Rate : 0.25
 Accuracy rate of training  1.0
Accuracy score of the test : 0.9
Learning Rate : 0.5
 Accuracy rate of training  1.0
Accuracy score of the test : 0.9
Learning Rate : 0.75
 Accuracy rate of training  1.0
Accuracy score of the test : 0.9
Learning Rate : 1
 Accuracy rate of training  1.0
Accuracy score of the test : 0.9
```



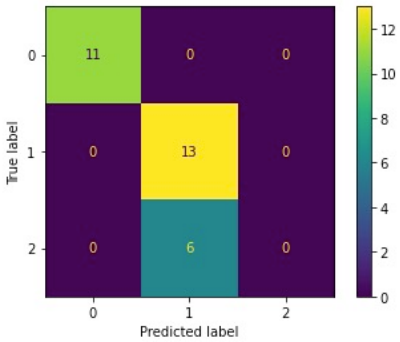
(5) Neural networks

- For the breast cancer dataset:

(1) I chose Keras and it package for the neural network training. For the activation function, I used ReLu (rectified linear unit) function as the activation function in the input layer and the hidden layers. Since Sigmoid function is used when dealing with classification problems with 2 types of results, I use Sigmoid as the activation function in the output layer. The final accuracy is 0.9298.



- For the iris flower dataset:
  (1) Due to the small size of the dataset, the accuracy is really low. After 300 epochs, the accuracy stays at 0.80.



4. Conclusion

Here is what I have learned from this project:

1. For the breast cancer classification problem, all 5 machine learning models have great performance since their accuracy score are all above 0.9. And my first lesson learned here is that a tiny difference in the accuracy score cannot be used to judge the absolute performance of models. I use accuracy as an indicator to choose which model is more suitable for the problem of benign or malignant tumors. But it does not mean that the highest score model will perform the best in actual situations, because, in realities, there may be larger datasets, more features, etc., which will affect the accuracy of the model.
2. If performances of difference models are comparable equivalent, analyzing the f1-score would help to determine the best performance model.
3. When training models, pay attention to outliers, models like support vector machines and gradient boosting are quite sensitive to outliers.
4. As presented in the KNN instance (iris dataset), small size data plays tricks in the performance. The accuracy varies drastically even when change the random state.
5. SVM has a stable performance not matter the size of the dataset.
6. When training Boosting with small size of data, overfitting can take place easily.
7. When training SVM, linear kernel should be the NO.1 choice since it has the greatest performance.
8. Neural network is not a good choice when the data size is small. And one of its disadvantages is that I cannot see what happened in the training process. In other words, the process is not interpretable.
9. Decision tree is the most direct and interpretable model.
10. GridSearchCV is always a good choice to cross-validate data to find the suitable parameters.