# Introduction to datamining lab3: Intro to classification

Qi Song (song.1602)

Oct. 20th, 2019

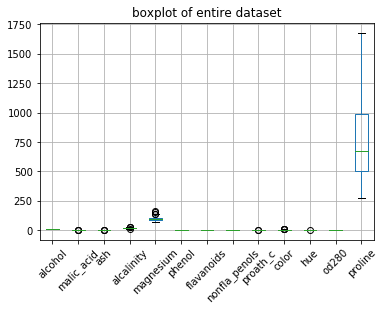
In this project we studied on wine data set from UCI machine learning repo. We analyzed the dataset via statistics method and visualize some important features. We drop one of the correlated columns and applied logarithmic and linear normalization to the dataset and split it into two subsets. After that six classifiers with default hyperparameter settings from the SK learn were trained on the dataset and evaluated on both the training set and the test set. After which, a grid search on random forest for hyperparameter tuning is applied.

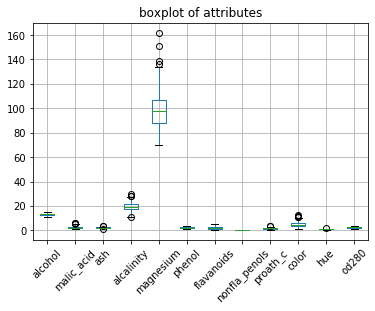
## Exploratory data analysis

### Basic knowledge of the data

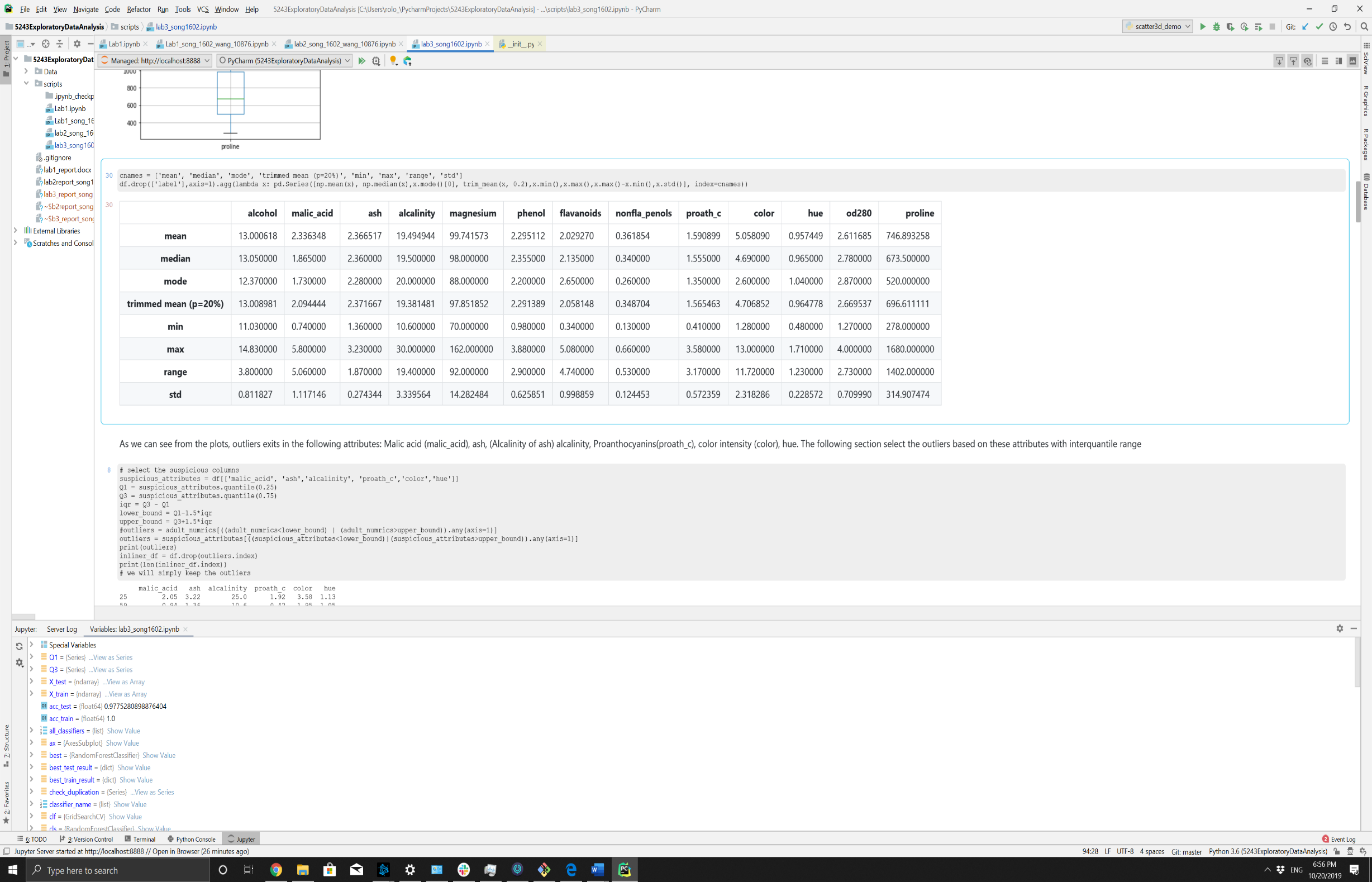
There are 178 data entries in the dataset, each datapoint has 13 attributes plus one class label. The meaning of 13 attributes are shown as follow.

1. Alcohol (alcohol): the amount of alcohol contains
2. Malic acid (malic\_acid): a organic acid the produce the flavor of sour taste of fruit
3. Ash (ash): inorganic matter that remains after evaporation and incineration
4. Alcalinity of ash (alcalinity): The alkalinity of the ash is defined as the sum of cations, other than the ammonium ion, combined with the organic acids in the wine. The alkalinity of ash will be expressed in milliequivalents per litre or in grams per litre of potassium carbonate.
5. Magnesium (magnesium): is a metal element that might be good at health. Some material suggests that it's related to ash
6. Total Phenols (phenols): responsible for most flavor and body of the wine, consists of following three parts: Benzaldehyde (vanillin) and Benzoic acid (Vanillic and Gallic acids); Catechins(largest quantity of phenols); Resveratrol
7. Flavanoids (flavanoids): antioxidants. Antioxidants help prevent certain molecules, known as free radicals, from damaging cells.
8. Nonflavaniod phenols (nonfla\_penols):phenols that are not flavanoid
9. Proanthocyanins (proath\_c) : Proanthocyanidins play an important role in wine; with the capability to bind salivary proteins, these condensed tannins strongly influence the perceived astringency of the wine.
10. Color intensity (color) : As a very general rule of thumb, the intensity of a wine's color is directly proportional to the intensity of the wine's flavor. Paler wines are generally milder while deep rich colors represent more robust flavors.
11. Hue (hue): The hues of red wines range from pink to varying shades of purple. You'll find lighter shades in wines such as Beaujolais and Pinot Noir and deeper, darker colors in wines like Cabernet Sauvignon, Syrah, and Merlot.
12. OD280/OD315 of diluted wine (od280): od280 is a measurement that can be used to measure concentrated protein. No explanation of od315 has been found, it should be another kind of measurement
13. Proline (proline) : Proline is typically the most abundant amino acid present in grape juice and wine. The amount present is influenced by viticultural and winemaking factors and can be of diagnostic importance.

The range of the data are explained in the following context. The class label contains 3 nominal values, from 1 to 3, which implies it is multiclass classification problem. After dropping the class label, we employ boxplot to have a general image of all the attributes.

As we can see from this figure, 11 of the 13 attributes have similar range, attribute, magnesium has a slightly larger value compared to other; proline attribute has a significantly different distribution compared to other attributes. Also, as we observed form the graph, there are some outliers in the dataset.

Since including proline in the boxplot will significantly reduced the readability of the plot, in the above graph we drop the proline attribute for a better visualization of the other attributes. As we can see, magnesium has a larger value compared to other attributes; alkalinity and alcohol has a slightly larger value compared to the rest of attributes.

The following chart shows some key statistics of the attributes.

Observations will be explained in the following subsections, which is used as a guide line for both outlier detection and data pre-processing.

### Data quality verification

### Missing value

By employing isnull() method from pandas, we found zero missing values in the entire dataset.

* + 1. Duplication

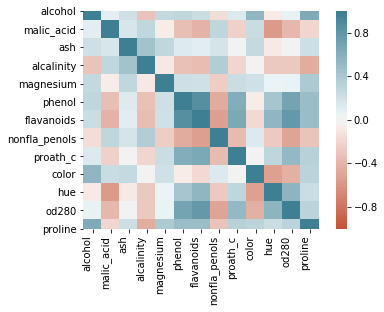
By employing duplicated() method for pandas.Dataframe object, we found zero duplicated object from the dataset, which means we will not remove any data entry for duplication.

* + 1. Outliers

As we can see from the box plot, outliers exits in the following attributes: Malic acid (malic\_acid), ash, (Alcalinity of ash) alcalinity, Proanthocyanins(proath\_c), color intensity (color), hue. So, we applied outlier detection with inter-quantile range on these attribute following the below mechanism:

In which is the quantile at 0.25, is the quantile at 0.75 and . For a given data point, if any of the suspicious attributes list in the above paragraph exceeds this range, it is classified as outlier. Based on such strategy, 15 outliers are found in the dataset. However, since most of the value does not significantly larger or smaller than the bounds, all of the outliers are kept in the dataset for further study

* + 1. Correlated and redundant attributes

To evaluate the inter-relation of the attribute, we employed the following heatmap plot based on correlation. As we can see from the figure, attribute od280 and proline has some correlation with most of the other attribute; however, such correlation is not strong. On the other hand, flavanoids and phenol are highly correlated with each other; based on such information, flavanoids attribute is dropped in the further study.

### Brief summary of the data

As we can see from the above analysis, this dataset contains 189 data entries and each data entries contains 13 different attributes; each of the data entries belongs to one of the three classes. We can say that this is a multiclass classification problem on a small dataset with reasonable dimensionality. Also, after verifying the quality of the data, we found that the dataset is in high quality. However, the range of the value and the correlation between different attributes suggests that data pre-processing should be applied before applying a classification model.

## Data pre-processing

In the following section, we discuss the data pre-processing techniques we applied to the dataset, plus, the reason of doing so. This paragraph gives you a brief summary of what will be mentioned here.

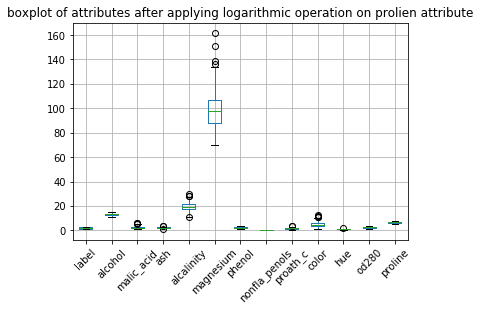
We then decided to simply drop attribute flavanoids for most of the information it contains might also be found in phenol. we also decided to apply a logarithmic operation on proline. A trine test split is employed to split the training and testing dataset. After which a linear rescaling is applied to all the attribute. Due to lack of domain knowledge, we did not create any feature or applied any further feature engineering techniques.

### Dropping correlated or redundant attribute

As we have discussed in section 1.2.4, attribute phenol and flavanoids are highly related. Attribute flavanoids was then dropped from the dataset in further study for the reason that the correlation suggests that most of the information that flavanoids contains are already embedded in phenol. Such strategy reduce the dimensionality without losing too much on information; also alleviate the performance loss on redundant attribute for those classifiers that are sensitive to such attributes.

## 2.2Logarithmic rescaling

In section 1.1, we have already discussed that proline has a significantly larger range than all other attributes, we then first apply a logarithmic operation with base 10 to it to reduce its value to a similar range compared to other attribute. Such operation does not guarantee that they result in same range of value. But they do map proline in a much smaller range. After this operation, the attributes still have different range, we will further apply min max recalling in section 2.4 to map them into the exact same range. The following figure shows the range of attributes after such logarithmic operation on proline.

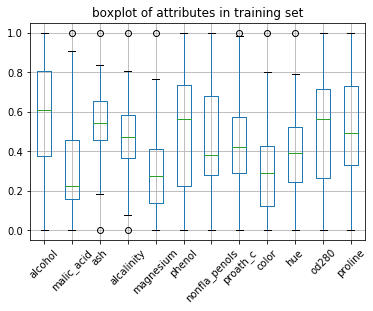
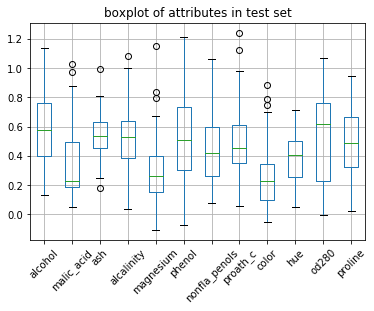


## Splitting training and test dataset

We employed train\_test\_split method from sklearn.model\_selection for splitting training and test set. The reason of doing so is to evaluate the ability to generalize from training data to unseen data pattern of classifiers. During the training or hyper-parameter tuning process. Test set will not be touched, which guarantees that the test data are real unseen patterns for the classifiers trained and tuned only on training set.

The split size of test data set is 0.5, resulting in training and test dataset at the same size: each of which has 89 data entries.

* 1. Linear(min/max) rescaling

After the test set is split from the training set, we applied a linear rescaling to all the attributes, note that under this context, proline has already been replaced by its log value. The function is provided by sklearn.preprocessing.MinMaxScaler. We first fit the scaler on the training attributes and then apply this scaler to both the training and test attributes. The reason of doing so is that MinMaxScaler will utilize the minimum and maximum value of each attribute to rescale it into the range of 0 to 1; excluding test set from fitting the minmax scaler can make sure that no information of minimum or maximum value of test set is implicitly passed to the training process, even though I personally do not think this will interfere the performance. Under such strategy, the value of attribute in training set is guaranteed to rest in the range of 0 to 1; such property does not hold true for test set, but they should fall in similar range.

The above figures show the range of attributes after all of the preprocessing on attribute has been applied. As described in the above paragraph, all of the values for any of the attributes in training set falls into the range between 0 and 1; and for test set, some of the value may exceed such range, for example, attribute phenol may have value at 1.2 or negative value.

### One-hot representation

The label is a nominal attribute. So here we applied one-hot representation to it. However, some classifiers in sklearn, including logistic regression, naïve bayes, and support vector machine classifier does not take this one-hot representation as output; instead, they simply take the integer representation and internally re-encode. So we will have both one-hot version and integer version of the labels.

## Build and evaluate classifiers

### Build classifiers

The classifiers we discuss include logistic regression, K nearest neighbors, decision tree, naïve bayes, linear support vector machine and random forest. As we have discussed in section 2.5, logistic regression, naïve bayes and support vector machine just take integer representation. The other three will be feed with training attributes and one-hot labels.

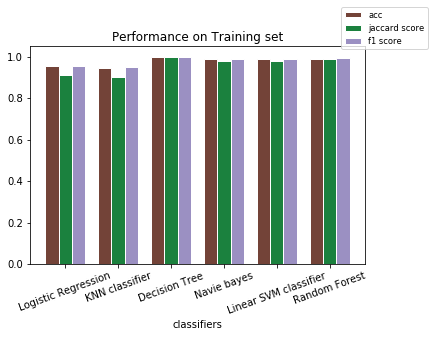
To be more precise, sklearn describe that classifiers like logistic regression will internally handle the multiclass classification problem into multiple binary classification problem; the output of such models on multiclass classification are also pure discrete, which limit our choice of evaluation benchmark to those does not assume that the output of the model is a probability.

All of the classifiers are trained only on training dataset, and with default hyper-parameters that sklearn provides.

* 1. Evaluating on training set
     1. Evaluation metrics

As requested, we include accuracy and F-1 score as our evaluation metrics. The accuracy is quite straight forward; as for F-1 score in this multiclass classification problem, we use a weighted average of F-1 scores correspond to the case of taking each of the three classes at positive class.

As mentioned in section 3.1, the discrete output of logistic regression, support vector machine and naïve bayes limits the choice of metrics; benchmark like ks-statistics, that assumes the output of classifiers is probability or confidence will fail; so we choose Jaccard score as other measurement. The Jaccard score is measured using micro average, which is also a weighted average correspond to the number of samples from each class.

* + 1. Result on evaluation

As we can see from the plot on the right,

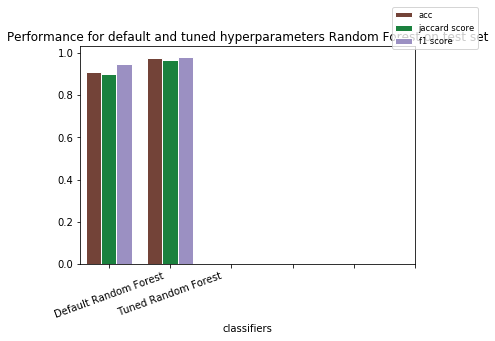
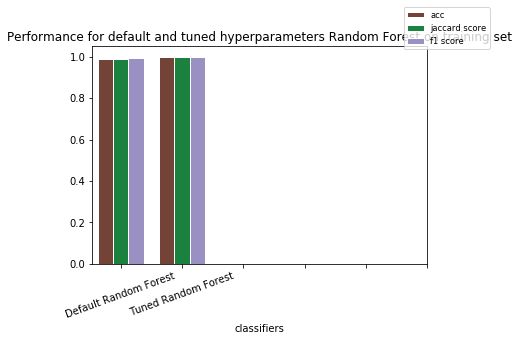
most of the classifiers performs well when evaluating on the training set. Logistic regression and KNN classifier are slight worse than the others but they have similar performance. Based on such information, I would guess random forest classifier as the best one when evaluated on the test set, for the following reason. First, random forest classifier together with other 3 classifiers achieve almost all true predictions on training set, although they have minor differences, but they are at the same level; also test set are sampled from the same dataset as the training set, so we can assume a similar distribution of data in test and training set, so that overfitting should not significantly reduce the performance; third, trees in random forest is a much simpler tree compared to normal decision tree, and such simple model might suffer less on overfitting issue; forth, random forest is a robust ensemble model, even though some of the tree in the forest fails to generalize, the entire ensemble should still can have reasonable prediction based on majority voting of all the trees. We will verify such hypothesis in the following section.

## C:\Users\rolo_\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\7BE682E0.tmpEvaluating models on training set

The above figure shows the models evaluated on the training dataset. Surprisingly we found that logistic regression, naïve bayes and linear svm does not suffer from the overfitting problem; KNN do have a minor drop in the performance compared to training evaluation; random forest do have smaller drop in performance when comparing with decision tree, but it still suffers from overfitting problem.

## Improving model: automatic hyper-parameter tuning

In this section, we employed grid search from sklearn.model\_selection on random forest classifier. The number of trees in the search range from 1 to 500 with step size at 40, criterion on gini and entropy, max depth of trees ranges from 1 to 10 with step size at 3, max features selection on sqrt and log2. After searching the entire grid on training set with 10-fold validation, grid search gives best parameter with number of trees at 481, max\_features of log2, max\_depth of 4, with criterion using entropy.

After obtaining the best hyper-parameters via grid search, a random forest classifier with best hyper-parameter is retrained on the entire training set. The above figure in the left compares the performance of default random forest and best random forest estimator on training set; the right one is on test set. As we can see, tuned hyper-parameter do significantly alleviate overfitting problem on random forest models.

We then compare the hyper-parameters of default setting and grid search result. The default tree number is 10, and the best tree number is 481, which is the largest value in the grid; the default tree depth is None, which allows the subtrees to fully expand to either pure nodes or leave nodes with only one samples; whereas the max\_depth in best model is 4, which is the second smallest value in the grid, and the smallest one is 1, which is a trivial value. Such difference in hyper-parameter matches the analysis in section 3.2.2, that ensemble model might be more robust; and simple subtrees are less sensitive to overfitting problem; it is simply the default ensemble does not include enough estimators and the child estimator is not simple enough. The choice of max\_features and criterion does not show a significant relevance to overfitting problem.

## Conclusion

The performance of most of the classifier yield almost 1 in the training set, which means our data pre-process is success. Dropping attribute flavanoids in section 2.1 does not result in unreasonable result; also, KNN classifier performs well on both the training and testing set, which implies that our rescaling strategy is right since KNN is sensitive to attribute with different scale. In section 3 and 4 we saw different classifiers; and in section 3.2.2 and section 5 we analyzed that ensemble method like random forest should be robust to overfitting problems for the reason that simpler model that are ensembled in ensemble model should be robust to overfitting problem; and the ensemble method should be able to avoid wrong prediction even when part of the submodels are learning occasional relevance in the training set, since it employs many submodels. We found that the number of trees is not large enough and the tree models are not simple enough for default random forest classifier to be immune to overfitting problem; but by employing more simpler subtree, we can verity the hypothesis that with right hyper-parameter setting, the random forest classifier should be robust to overfitting problems

## References

UCI Machine Learning Repo, <http://archive.ics.uci.edu/ml/index.php>, last accessed Oct. 2019.