Description

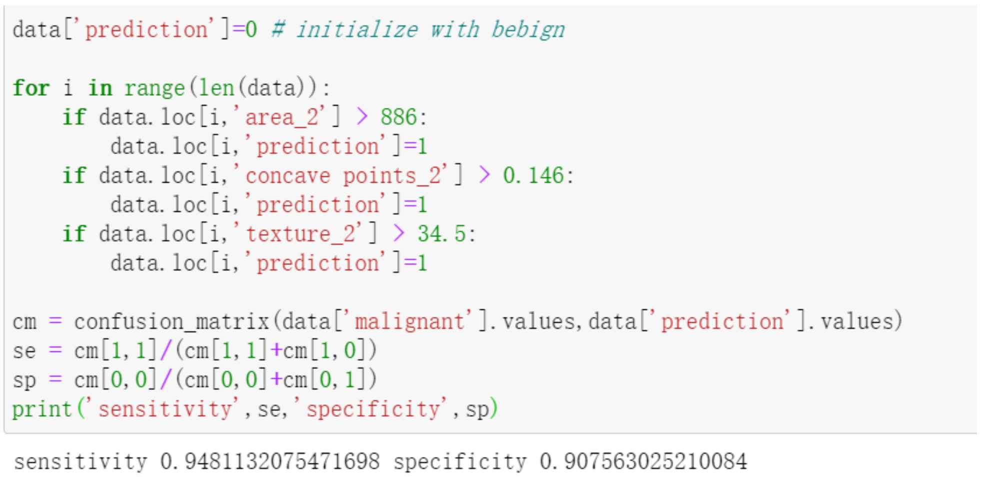
(a) Rule-based classifier

We implement the classifier with these steps: Select features, Select threshold, Buildup the classifier.

Select features: We've noticed that there might existing correlation between features, for example radius and perimeter. Thus, the first step is to pick the features with high correlation with the target 'malignant' while low correlation between each other. Here we use the Pearson index to measure the correlation. Meanwhile, we also consider different feature types, for example size, shape, texture. The finally selected features are 'area_2', 'concave points_2', 'perimeter_2', 'concavity_2', 'compactness_0', 'texture_2', 'smoothness_2', 'symmetry_2'.

Select threshold: We've notices that for most features, higher values indicate a malignant label. As a common sense, there will be a range for the normal value like [1.5, 3.5]. If a test result is higher than the upper bound, we consider the test to be positive. In this sense, we need to find this upper bound as our threshold. From statistical thinking, we set this threshold as mean+2sigma with respective to the negative cases. Then, we use this initial threshold as our baseline and make further adjustment in order to balance the sensitivity and specificity tradeoff. The main idea is to maximize both sensitivity and specificity while keeping FN larger than FP. Since our algorithm is in 'or' logic which will treat every abnormal feature as malignant indicator, keeping FN larger than FP can help to decrease over-classification into positive cases.

Buildup the classifier: As final step, we build up our model in an if-if-if style. We adjust the model by increasing/decreasing features which are selected from the first step. Finally, we end up with a model with a relatively high sensitivity and specificity with features 'area_2', 'concave points_2', 'texture_2'. Here we want the sensitivity to be higher than specificity because we think it's more important to correctly classify diseased people. The model and sensitivity/specificity scores are as below,



```
data['prediction']=0 # initialize with bebign

for i in range(len(data)):
    if data.loc[i,'area_2'] > 886:
        data.loc[i,'prediction']=1
    if data.loc[i,'concave points_2'] > 0.146:
        data.loc[i,'prediction']=1
    if data.loc[i,'texture_2'] > 34.5:
        data.loc[i,'prediction']=1

cm = confusion_matrix(data['malignant'].values,data['prediction'].values)
se = cm[1,1]/(cm[1,1]+cm[1,0])
sp = cm[0,0]/(cm[0,0]+cm[0,1])
print('sensitivity',se,'specificity',sp)
```

```
sensitivity 0.9481132075471698 specificity 0.907563025210084
```

(b) Random forest classifier

As request by the question, we directly use the scikit-learn framework to build the random forest classifier. Here we use all the given features. To find a model with better performance, we tune the parameters of ‘min_samples_leaf’ and ‘n_estimators’. ‘min_samples_leaf’ is the minimum number of leaf nodes in the tree, and ‘n_estimators’ is the number of trees to build. After some tests, we set the two parameters as 7 and 56. The final sensitivity score for the test data set is around 0.962 and specificity score is around 0.944. Both are better than the rule-based model.

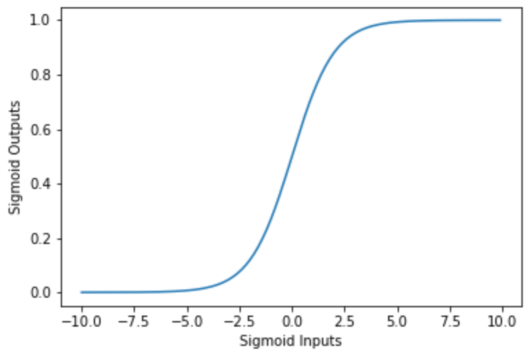
(c) Our own classifier

With some research on different models, we get the knowledge that explicit models can be more interpretable, which incl. regression models, decision trees and traditional classification rules. To find which models among these can get a better performance, we firstly use four different classifying models in sklearn and use cross validation to evaluate their performance, results are: Dummy Classifier < Decision Tree Classifier < Random Forest Classifier < Logistic Regression

As the data reveal, model Logistic Regression works best and the accuracy of the model is calculated: 0.956, so we decided to use this model.

Logistic Regression is a generalized linear model which included nonlinear factors through Sigmoid function, so it can handle 0/1 classification problem easily.

Sigmoid function: $g(z) = \frac{1}{1+e^{-z}}$ .Its function curve is as follows:



The assumed function form of logistic regression is as follows:

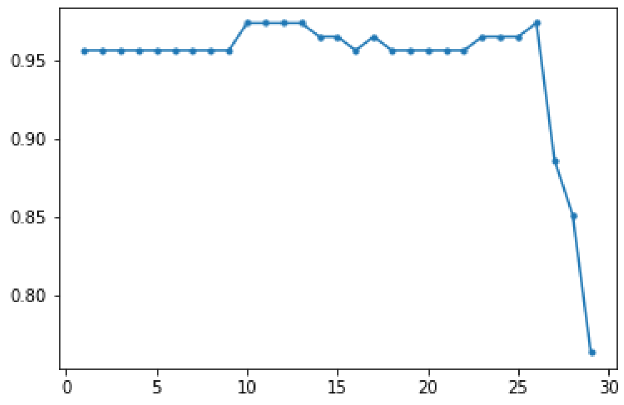
$$h_\theta(x) = g(\theta^T x), g(z) = \frac{1}{1+e^{-Z}} \quad \text{So: } h_\theta(x) = \frac{1}{1+e^{-\theta^T x}},$$

Where x is our input, θ is the parameter we require to take.
The logistic regression model limits the decision function to a certain set of conditions, which determines the hypothesis space of the model. The logistic regression model assumes:

$$P(y=1 | x;\theta) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

The corresponding decision function is as follows: $y^* = 1$, if $P(y=1 | x) > 0.5$

As a further simplification of our model, we want to filter out the most important features and only use them for our model, since with too many features, the model can be very complicated and difficult to interpret. We use a built-in method based on machine learning Logistic Regression model to rank all the features. The plot below shows how the model accuracy will change with more features included. The x-axle represents the features in a descending order of importance, and the y-axle represents the model accuracy.



It’s very interesting that the image shows: Increasing the number of features, accuracy will increase in the beginning, but soon there are only small fluctuations. So, we decide to use only the first few important features: 'radius_1', 'texture_2', 'area_1', 'concave points_2', ‘radius_2’, ‘area_2’. From the feature interaction research, we know that there are some interactions between these features. Thus, we further drop the redundant features. The remaining ones are ‘'radius_1’, ‘texture_2’, ‘concave points_2’, ‘area_2’. With these four features considered, the accuracy is 0.964. It’s even higher than all features considered before!

So far, we consider the interpretability of this model is good (comparing with black-box classifiers like neural network) But can we make it better? Then we do some further exploration in finding trade-off between interpretability and classification performance.

It’s an important step to standardized data. But it may cause trouble to doctors because it’s hard to interpret how change of a feature can cause change in the final classification result. So we decide to remove this process to increase interpretability. And after removing, accuracy decrease to 0.956. Final sensitivity score is 0.957 and specificity score is 0.955.

To conclude, in order to obtain more interpretability, the accuracy of the model decreases accordingly, the trade-off should be carefully to balance.

Comparation of the three classifiers

If we compare the three classifiers by sensitivity score, we can see that random forest has the best performance, logistic regression is the second, and rule-based classifier is the last.

With respect to interpretability, the rule-based classifier is highly interpretable. During the stage of feature selection, we already cross off the features with low impact on the decision, as well as the features being redundant and interacting with others. When the decision is made, we can easily identify the reasons behind that, for example which features have participated in the decision making and which features have not.

Random forest classifier, on the other hand, is not so easy to interpret due to the reason that it is built on many trees with much randomness. One way to measure the feature importance is through mean decrease impurity. However, if the features are corelated with each other, this can also cause biases. For instance, if feature A is firstly selected, feature B which is highly corelated with A will get a low mean decrease impurity value.

Our own designed classifier is based on a logistic regression. The model is close to monotonic linear functions, so it is easy to be interpret. We can easily tell each feature's importance by their coefficient in the model. No standardizing process means people can interpret result based on observed values directly. The essential interpretation method for logistic regression model is: if you increase the value of feature x(i) by one unit, the estimated odds change by a factor of exp(w(i)), where w(i) is the coefficient of feature x(i). One additional advantage for logistic regression is that we can also calculate the probability of a classification. However, logistic regression might not perform well if the features are linearly interacted with each other. In this case, we check the feature interactions and reduce the redundant ones before building our model. Many features are interacted with each other, for example radius and perimeter.

The result can show a trade-off between model performance and interpretability. Before we build the classifier system, we need to ask ourselves what is more important to us in this particular case, performance or interpretability.