Assignment 6

Josh Barber

barb4630@mylaurier.ca

215854630

1. In Code Listing 6.07, we have 1 logistic regression model and 2 decision tree models trained on the breast cancer dataset. Can you use what you learned in this lesson to get the top two important features of each model and write down feature names in the answer. Are they consistent? (1 point)

**Code:**

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

import pandas as pd

def warn(\*args, \*\*kwargs):

pass

import warnings

warnings.warn = warn

data\_bunch = load\_breast\_cancer()

bc\_X, bc\_y = data\_bunch.data, data\_bunch.target

x\_train, x\_test, y\_train, y\_test = train\_test\_split(bc\_X, bc\_y, test\_size=0.2)

logit\_regr = LogisticRegression()

logit\_regr.fit(x\_train, y\_train)

dt\_gini\_clf = DecisionTreeClassifier()

dt\_gini\_clf.fit(x\_train, y\_train)

dt\_ig\_clf = DecisionTreeClassifier(criterion="entropy")

dt\_ig\_clf.fit(x\_train, y\_train)

feat\_importances\_gi = pd.Series(dt\_gini\_clf.feature\_importances\_, index=data\_bunch.feature\_names)

feat\_importances\_gi.nlargest(8).plot(kind='barh')

feat\_importances\_ig = pd.Series(dt\_ig\_clf.feature\_importances\_, index=data\_bunch.feature\_names)

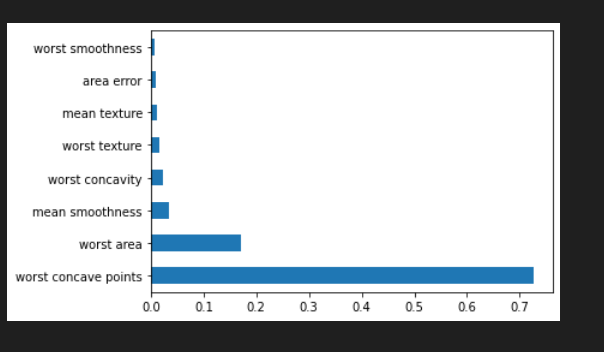
feat\_importances\_ig.nlargest(8).plot(kind='barh')

**Output:**

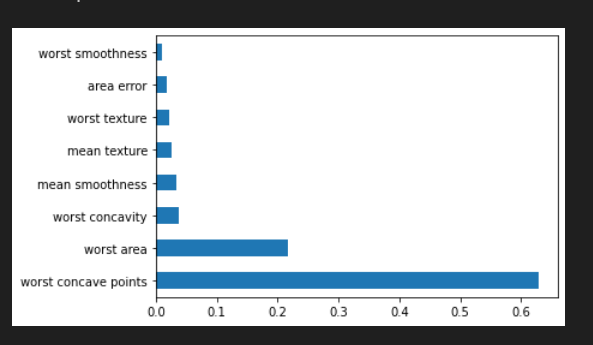
Logistic Regression has coefficients, rather than features. There was no graph including features for the Logistic Regression model.

I have trained the models several times and I noticed the features: **worst concave points** and **worst area** have shown up the most. However, it is not consistent. I am assuming the dataset size has much to do with the variability, as the dataset has only 569 entries.

**Feature Importance for Gini Impurity in Decision Tree**



**Feature Importance for Information Gain in Decision Tree**



2. We don't have code practice for the GBDT model yet because we will do this in this exercise. The **GradientBoostingClassifier()** construct can create a GBDT model object. Can we use it to train a GBDT model and compare its performance with the decision tree/random forest classifiers? We can use default parameters for all three models. They will be applied to the breast cancer dataset as in Code Listing 6.06. As usual, 80% of the dataset is for training and 20% for the test purpose. Please write down the code of training/test for the three models and compare their average accuracy performance. (2 points)

**Code:**

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

def warn(\*args, \*\*kwargs):

pass

import warnings

warnings.warn = warn

data\_bunch = load\_breast\_cancer()

bc\_X, bc\_y = data\_bunch.data, data\_bunch.target

x\_train, x\_test, y\_train, y\_test = train\_test\_split(bc\_X, bc\_y, test\_size=0.2)

dt\_gini\_clf = DecisionTreeClassifier(criterion="entropy")

dt\_gini\_clf.fit(x\_train, y\_train)

print("information gain decision tree: ", dt\_gini\_clf.score(x\_test, y\_test))

clf = RandomForestClassifier(max\_depth=8)

clf.fit(x\_train, y\_train)

print("random forest with max depth 8: ", clf.score(x\_test, y\_test))

gb\_cf = GradientBoostingClassifier()

gb\_cf.fit(x\_train, y\_train)

print("gradient boosting classifier: ", gb\_cf.score(x\_test, y\_test))

**Output:**

As can be seen from the table below, the random forest with a depth of 8 outperformed all other models. The gradient boosting classifier performed well above expectation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Run 1 Score** | **Run 2 Score** | **Run 3 Score** | **Average** |
| **Information Gain Decision Tree** | 0.9035087719298246 | 0.8771929824561403 | 0.8947368421052632 | 0.89181286549708 |
| **Random Forest with max Depth 8** | 0.9385964912280702 | 0.956140350877193 | 0.9385964912280702 | 0.94444444444444 |
| **Gradient Boosting Classifier** | 0.9210526315789473 | 0.9649122807017544 | 0.9122807017543859 | 0.9327485380117 |