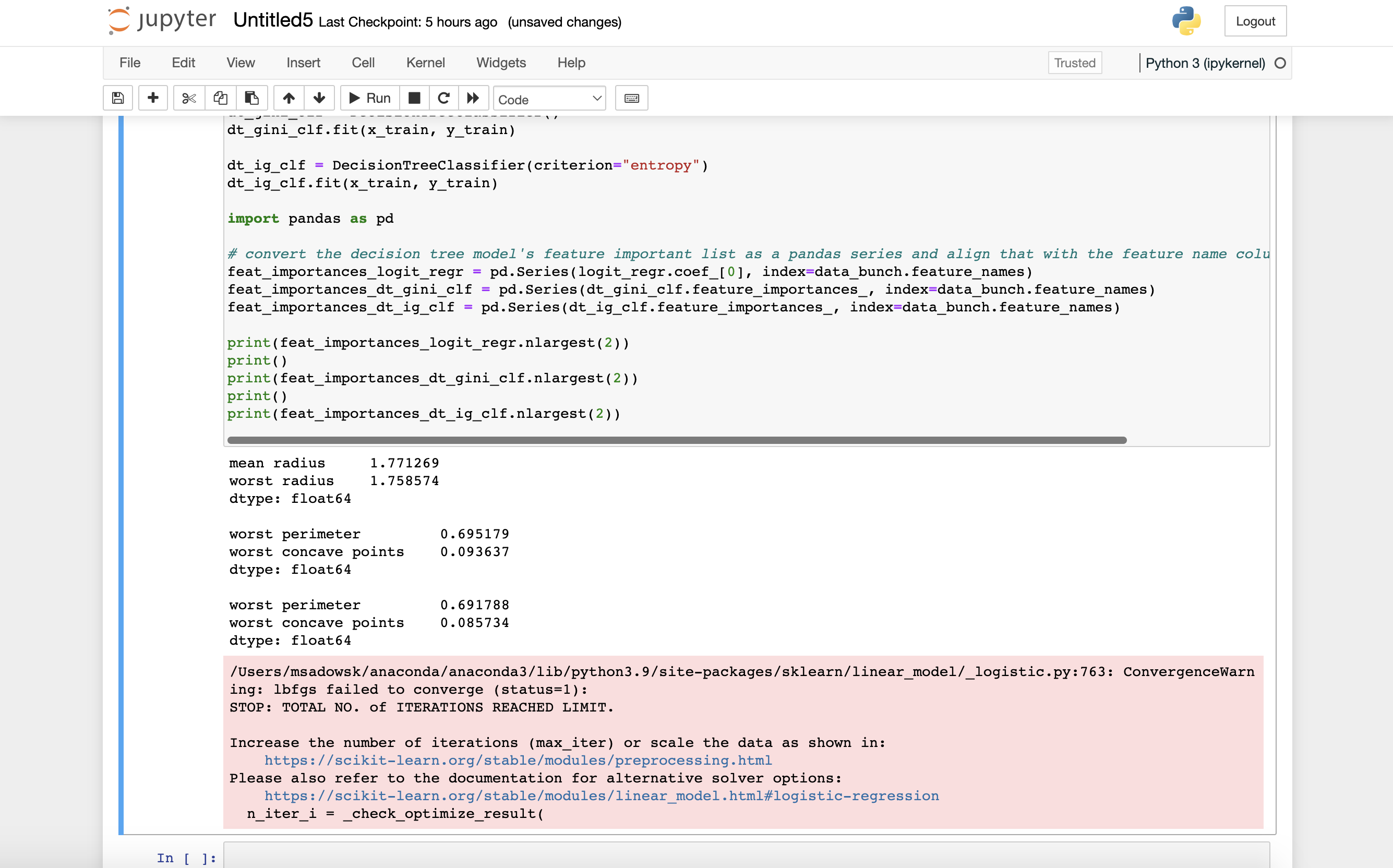
1. **In Code Listing 6.07, we have 1 logistic regression model and 2 decision tree models trained on the breast cancer dataset. Can you use what you learned in this lesson to get the top two important features of each model and write down feature names in the answer. Are they consistent? (1 point)**

**Output:**



*mean radius 1.771269*  
*worst radius 1.758574*  
*dtype: float64*  
  
*worst perimeter 0.695179*  
*worst concave points 0.093637*  
*dtype: float64*  
  
*worst perimeter 0.691788*  
*worst concave points 0.085734*  
*dtype: float64*

The 2 decision tree models remain consistent in terms of which 2 features are the top 2 for both models (meaning, they produce the same feature name but a different score). Occasionally they will be different but often they remain consistent with each other. On the other hand, the logistic regression model often is not consistent with the 2 tree models and produces its own result.

**Code:**

from sklearn.datasets import load\_breast\_cancer

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

from sklearn.tree import DecisionTreeClassifier

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)

# train 3 models on the same set so we can compare their performance

from sklearn.linear\_model import LogisticRegression

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)

import pandas as pd

# convert the decision tree model's feature important list as a pandas series and align that with the feature name columns in the dataset

feat\_importances\_logit\_regr = pd.Series(logit\_regr.coef\_[0], index=data\_bunch.feature\_names)

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

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

print(feat\_importances\_logit\_regr.nlargest(2))

print()

print(feat\_importances\_dt\_gini\_clf.nlargest(2))

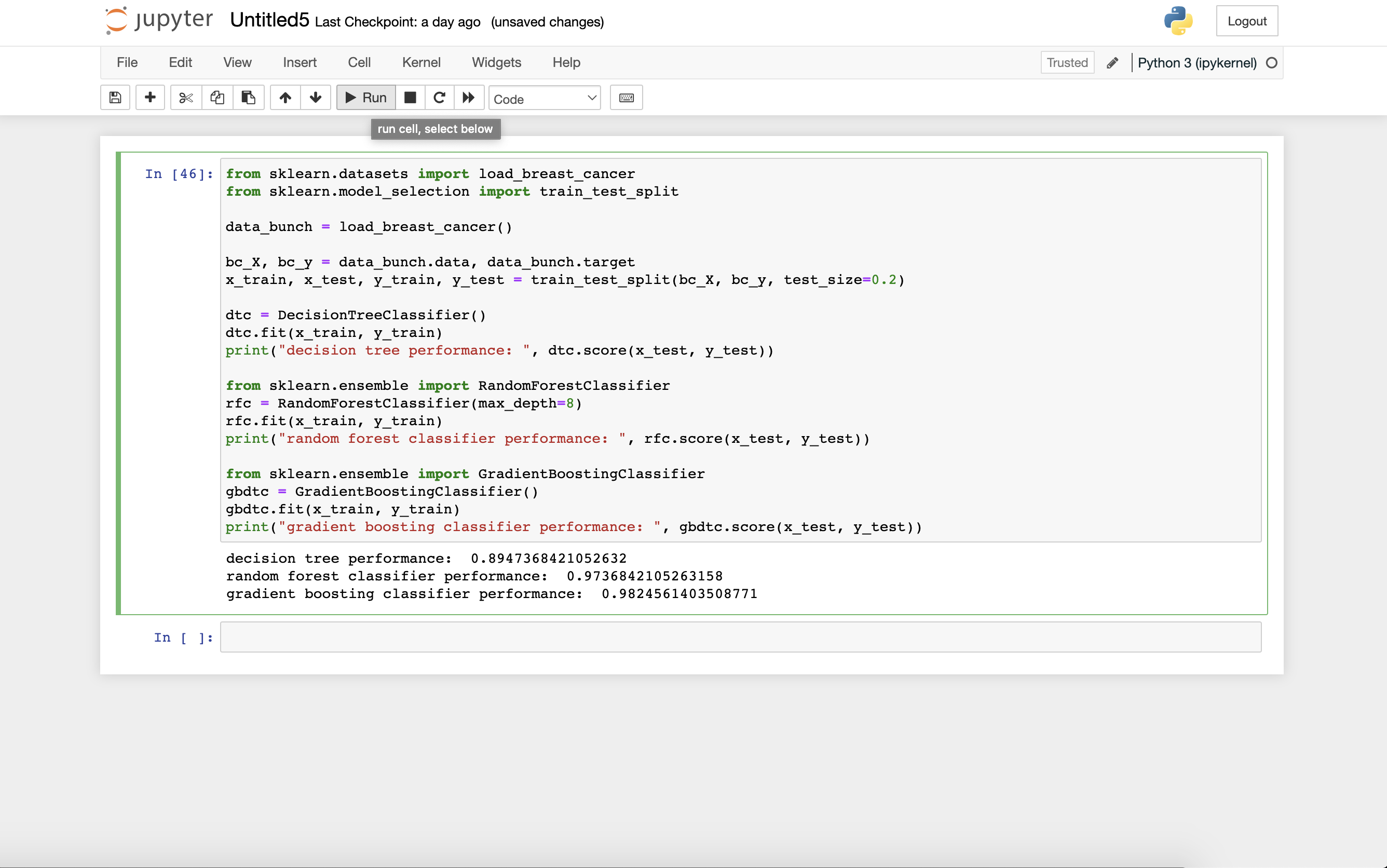
print()

print(feat\_importances\_dt\_ig\_clf.nlargest(2))

References:

1. <https://machinelearningmastery.com/calculate-feature-importance-with-python/#:~:text=and%20ElasticNet%20models.-,Logistic%20Regression%20Feature%20Importance,a%20crude%20feature%20importance%20score>.
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)**

**Output:**



decision tree performance: 0.8947368421052632  
random forest classifier performance: 0.9736842105263158  
gradient boosting classifier performance: 0.9824561403508771

Based on the output of the code below, it appears that the gradient boosting classifier performs the best, followed by the random forest classifier (with max\_depth = 8). The worst performing classifier from this experiment is the decision tree classifier.

**Code:**

from sklearn.datasets import load\_breast\_cancer

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

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)

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

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

print("decision tree performance: ", dtc.score(x\_test, y\_test))

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(max\_depth=8)

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

print("random forest classifier performance: ", rfc.score(x\_test, y\_test))

from sklearn.ensemble import GradientBoostingClassifier

gbdtc = GradientBoostingClassifier()

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

print("gradient boosting classifier performance: ", gbdtc.score(x\_test, y\_test))