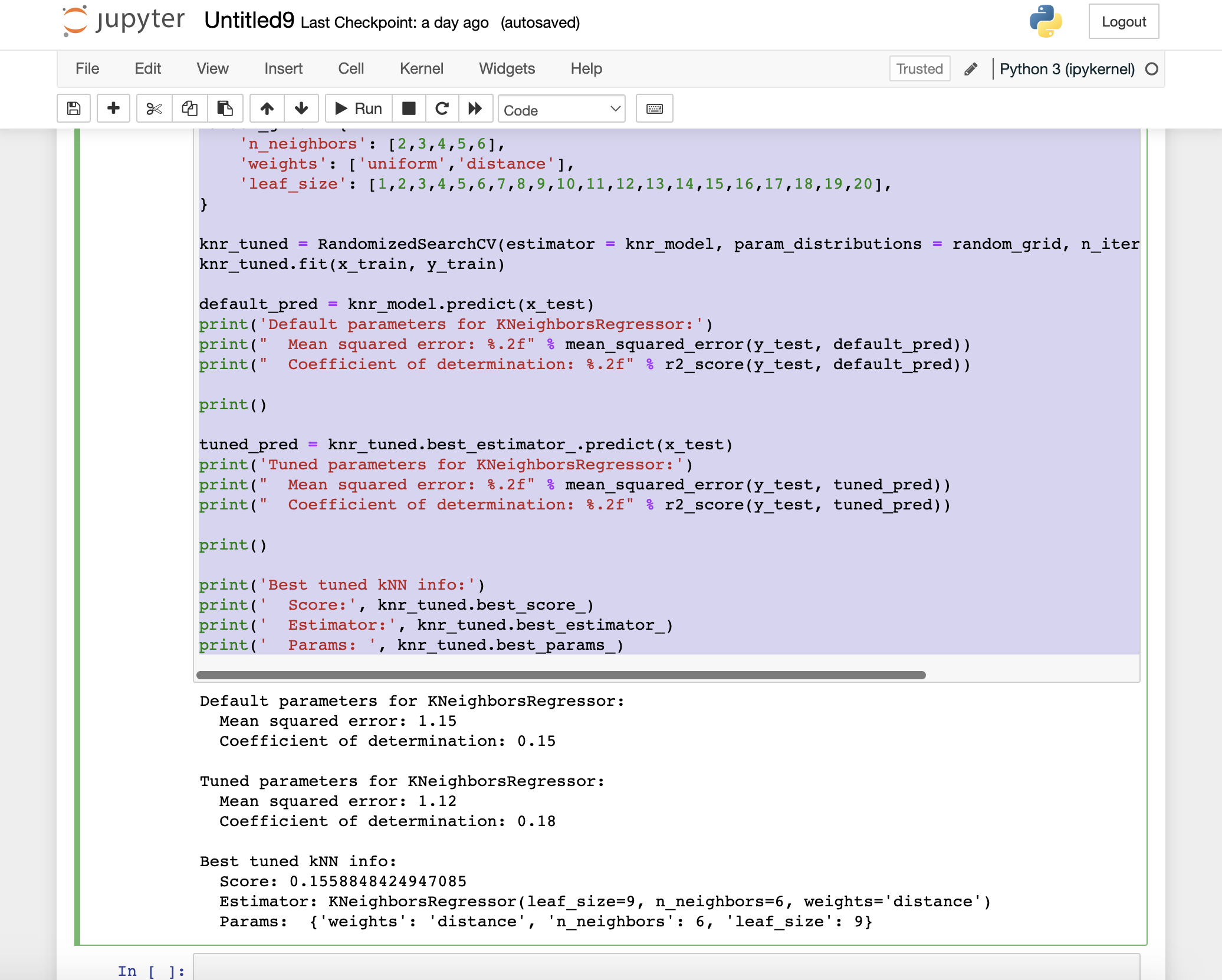
1. **In section 6, we have learned how to apply the randomized search for hyperparameter tuning together with cross-validation. Let's try to do the same thing on a k-NN model. We can reuse the code in Code Listing 9.03, and continue with your code of applying 5-fold cross-validation by RandomizedSearchCV(). You can define your hyperparameter grid and compare the fine-tune k-NN model with the one with default parameters. The hyperparameter list of k-NN regressors can be found at** [**sklearn.neighbors.KNeighborsRegressor**](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html)**.**

**Output:**



Default parameters for KNeighborsRegressor:  
 Mean squared error: 1.15  
 Coefficient of determination: 0.15  
  
Tuned parameters for KNeighborsRegressor:  
 Mean squared error: 1.12  
 Coefficient of determination: 0.18  
  
Best tuned kNN info:  
 Score: 0.1558848424947085  
 Estimator: KNeighborsRegressor(leaf\_size=9, n\_neighbors=6, weights='distance')  
 Params: {'weights': 'distance', 'n\_neighbors': 6, 'leaf\_size': 9}

Tuning the hyperparameters of the 2nd model improves the performance as can be seen from the output above. The most optimal parameters are listed above as well.

**Code:**

import numpy as np

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.datasets import fetch\_california\_housing

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

from sklearn.metrics import mean\_squared\_error, r2\_score

california\_housing\_bunch = fetch\_california\_housing()

california\_housing\_X, california\_housing\_y = california\_housing\_bunch.data, california\_housing\_bunch.target

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

from sklearn.neighbors import KNeighborsRegressor

knr\_model = KNeighborsRegressor()

knr\_model.fit(x\_train, y\_train)

random\_grid = {

'n\_neighbors': [2,3,4,5,6],

'weights': ['uniform','distance'],

'leaf\_size': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],

}

knr\_tuned = RandomizedSearchCV(estimator = knr\_model, param\_distributions = random\_grid, n\_iter = 10, cv = 5, n\_jobs = -1)

knr\_tuned.fit(x\_train, y\_train)

default\_pred = knr\_model.predict(x\_test)

print('Default parameters for KNeighborsRegressor:')

print(" Mean squared error: %.2f" % mean\_squared\_error(y\_test, default\_pred))

print(" Coefficient of determination: %.2f" % r2\_score(y\_test, default\_pred))

print()

tuned\_pred = knr\_tuned.best\_estimator\_.predict(x\_test)

print('Tuned parameters for KNeighborsRegressor:')

print(" Mean squared error: %.2f" % mean\_squared\_error(y\_test, tuned\_pred))

print(" Coefficient of determination: %.2f" % r2\_score(y\_test, tuned\_pred))

print()

print('Best tuned kNN info:')

print(' Score:', knr\_tuned.best\_score\_)

print(' Estimator:', knr\_tuned.best\_estimator\_)

print(' Params: ', knr\_tuned.best\_params\_)

**References**:

1. <https://openclassrooms.com/en/courses/6401081-improve-the-performance-of-a-machine-learning-model/6559796-tune-your-hyperparameters>