**Extra Credit: Putting it All Together: Data Cleaning to Model Evaluation**

Sushmita Prafull Halasawade

IFT 511: Analyzing Big Data (2022 Fall-MW)

Instructor: Asmaa Elbadrawy

4 Dec 2022

**PartI: Data Set Information:**

I am working with **Heart Disease Data Set**

Number of Instances: 303

Number of attributes: 14

Description of each attribute:

1. age in years

2. sex (1 = male; 0 = female)

3.(cp) chest pain type

1 = typical angina

2 = atypical angina

3 = non-anginal pain

4 = asymptomatic

4. (trestbps) resting blood pressure (in mm Hg on admission to the

hospital)

5. (chol) serum cholestoral in mg/dl

6. (fbs) (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

7. (restecg) resting electrocardiographic results

0 = normal

1 = having ST-T wave abnormality (T wave inversions and/or ST

elevation or depression of > 0.05 mV)

2 = showing probable or definite left ventricular hypertrophy

by Estes' criteria

8. (thalach) maximum heart rate achieved

9. (exang) exercise induced angina (1 = yes; 0 = no)

10.(oldpeak) ST depression induced by exercise relative to rest

11.(slope) the slope of the peak exercise ST segment

1 = upsloping

2 = flat

3 = downsloping

12. (ca) number of major vessels (0-3) colored by flourosopy

13. (thal) thal:

3 = normal

6 = fixed defect

7 = reversable defect

14. (num) (the predicted attribute) diagnosis of heart disease (angiographic disease status)

0 = < 50% diameter narrowing

1 = > 50% diameter narrowing

**PartII: Loading dataset:**

Code:

import pandas as pd

df = pd.read\_csv('processed.cleveland.data', header=None)

df

output:

Graphical user interface, text

Description automatically generated with medium confidence

Code:

headers = {0 : "age",

1 : "sex",

2 : "cp",

3 : "trestbps",

4 : "chol",

5 : "fbs",

6 : "restecg",

7 : "thalach",

8 : "exang",

9 : "oldpeak",

10 : "slope",

11 : "ca",

12 : "thal",

13 : "Diagnosis"}

df = df.rename(columns=headers)

df

Output:

Table

Description automatically generated

**PartIII: Finding and removing missing data from the dataset:**

Code:

df = df[~df.isin(['?', '-9'])]

df

df2=df.dropna()

df2

Output:

Graphical user interface, table

Description automatically generated

**PartIV: Finding accuracy of Decision tree with gini and entropy with stratified k fold.**

Code:

from sklearn.model\_selection import StratifiedKFold

from sklearn.preprocessing import StandardScaler

from sklearn import tree

from sklearn.model\_selection import GridSearchCV

from sklearn.pipeline import Pipeline

skf = StratifiedKFold(n\_splits=5)

target = df2.loc[:,'Diagnosis']

X=df2

y =target

cnt = 1

ss = StandardScaler()

dt = tree.DecisionTreeClassifier()

pipe = Pipeline(steps=[('std\_slc', ss),

('dec\_tree', dt)])

print("Creating Parameter Grid...")

criterion = ['gini', 'entropy']

max\_depth = [760]

param\_grid = dict(dec\_tree\_\_criterion=criterion,

dec\_tree\_\_max\_depth=max\_depth)

giniscore = 0

entropyscore = 0

print("Creating a grid search cross validator object...")

for train\_index, test\_index in skf.split(X, y):

print(f'Fold:{cnt}, Train set: {len(train\_index)}, Test set:{len(test\_index)}')

x\_train\_fold, x\_test\_fold = X.iloc[train\_index], X.iloc[test\_index]

y\_train\_fold, y\_test\_fold = y.iloc[train\_index], y.iloc[test\_index]

search = GridSearchCV(pipe,param\_grid)

search = search.fit(x\_train\_fold, y\_train\_fold)

df3 = pd.DataFrame(search.cv\_results\_)

df3.to\_csv("tada.csv")

giniscore=giniscore +df3.loc[df3['param\_dec\_tree\_\_criterion'] == 'gini', 'mean\_test\_score']

entropyscore=entropyscore+df3.loc[df3['param\_dec\_tree\_\_criterion'] == 'entropy', 'mean\_test\_score']

for index, row in df3.iterrows():

print("KFold ", cnt , ": ",row["params"], ", Mean Test Accuracy =",row["mean\_test\_score"],", Mean StdDev=",row["std\_test\_score"])

print("Best parameter (CV score=%0.3f):" % search.best\_score\_)

print(search.best\_params\_)

pipe2 = Pipeline([('scaler', StandardScaler()), ('bestmodel', search.best\_estimator\_)])

pipe2.fit(x\_train\_fold, y\_train\_fold)

best\_accuracy = pipe2.score(x\_test\_fold, y\_test\_fold)

print("Accuracy of decision tree with heart disease dataset : %0.3f" % best\_accuracy)

cnt+=1

giniscore = giniscore/5

entropyscore = entropyscore/5

print("overall gini accuracy:%0.3f" % giniscore)

print("overall entropy accuracy:%0.3f" % entropyscore)

Output:

Creating Parameter Grid...

Creating a grid search cross validator object...

Fold:1, Train set: 237, Test set:60

KFold 1 : {'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

KFold 1 : {'dec\_tree\_\_criterion': 'entropy', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

Best parameter (CV score=1.000):

{'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760}

Accuracy of decision tree with heart disease dataset : 1.000

Fold:2, Train set: 237, Test set:60

KFold 2 : {'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

KFold 2 : {'dec\_tree\_\_criterion': 'entropy', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

Best parameter (CV score=1.000):

{'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760}

Accuracy of decision tree with heart disease dataset : 1.000

Fold:3, Train set: 238, Test set:59

KFold 3 : {'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

KFold 3 : {'dec\_tree\_\_criterion': 'entropy', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

Best parameter (CV score=1.000):

{'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760}

Accuracy of decision tree with heart disease dataset : 1.000

Fold:4, Train set: 238, Test set:59

KFold 4 : {'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

KFold 4 : {'dec\_tree\_\_criterion': 'entropy', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

Best parameter (CV score=1.000):

{'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760}

Accuracy of decision tree with heart disease dataset : 1.000

Fold:5, Train set: 238, Test set:59

KFold 5 : {'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

KFold 5 : {'dec\_tree\_\_criterion': 'entropy', 'dec\_tree\_\_max\_depth': 760} , Mean Test Accuracy = 1.0 , Mean StdDev= 0.0

Best parameter (CV score=1.000):

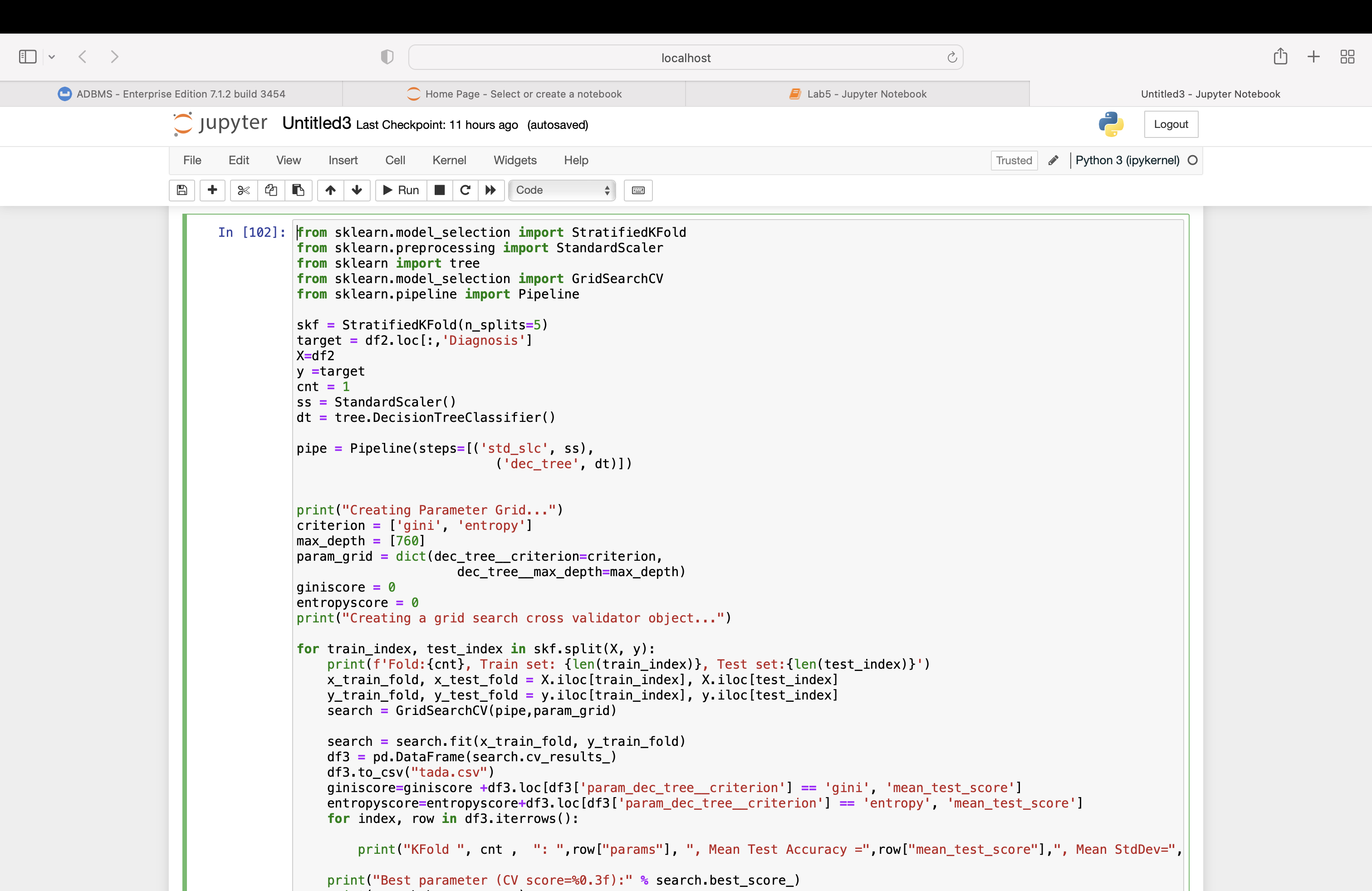
{'dec\_tree\_\_criterion': 'gini', 'dec\_tree\_\_max\_depth': 760}

Accuracy of decision tree with heart disease dataset : 1.000

overall gini accuracy:1.000

overall entropy accuracy:1.000

From above we can see that both gini and entropy criteria gives highest accuracy of 1. but gini is fastest so best criteria returned with GridSearchCV is gini.



Graphical user interface, text

Description automatically generated

**Reference:**

UCI Machine Learning Repository: Heart Disease Data Set. (n.d.). Archive.ics.uci.edu. http://archive.ics.uci.edu/ml/datasets/Heart+Disease