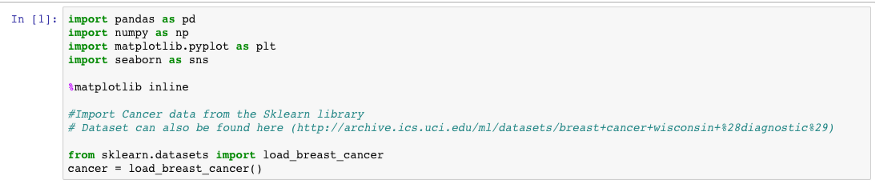
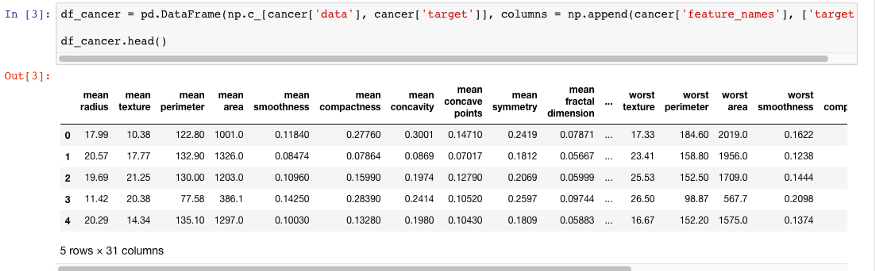
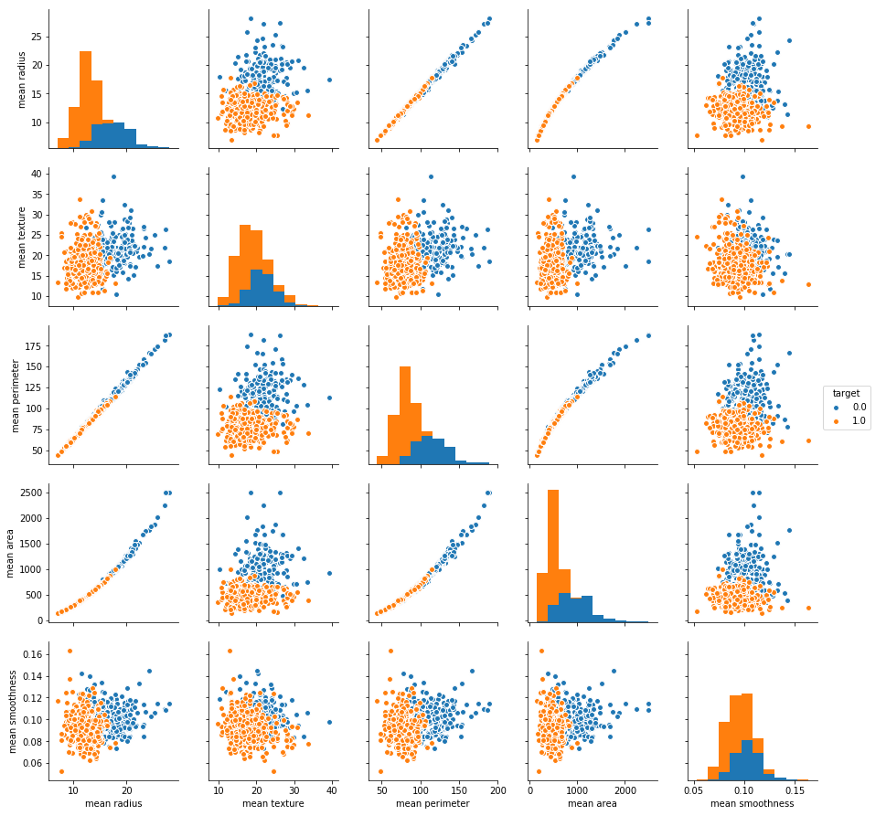
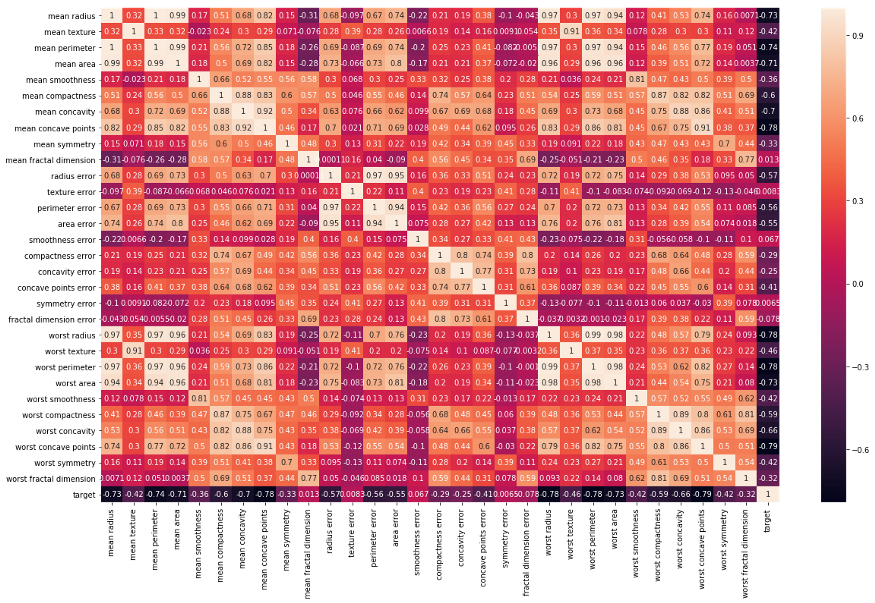
Assignment 1

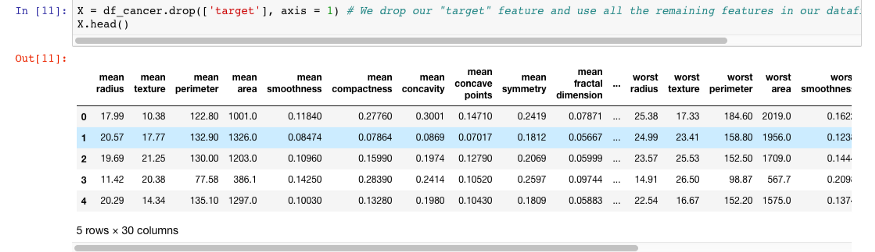
Q1.

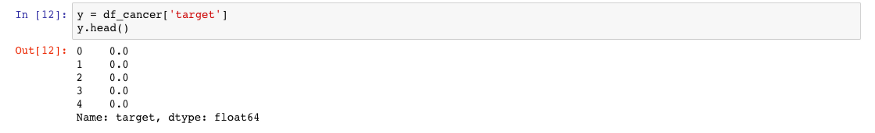




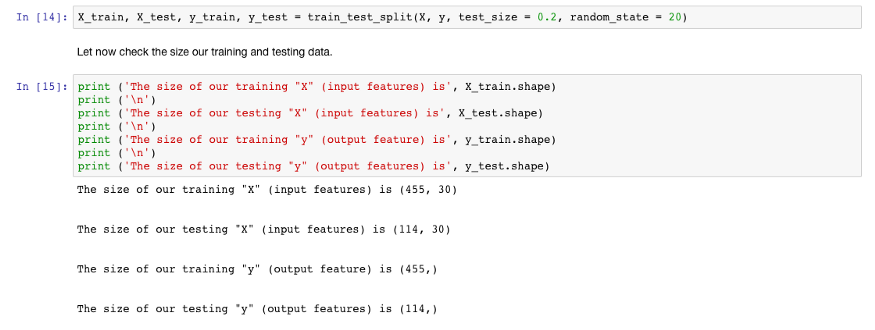


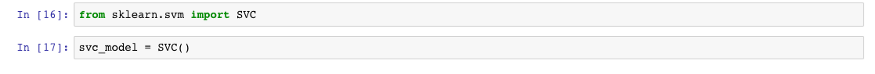


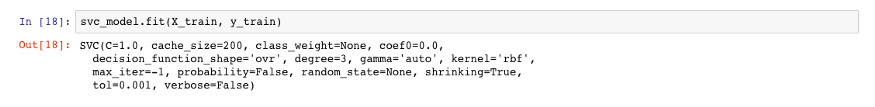




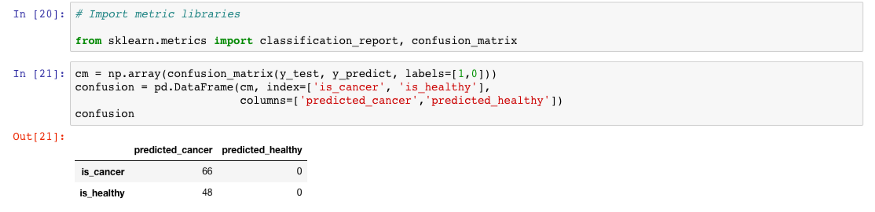
https://miro.medium.com/max/875/1*JFpwHz00VnxzBN0yjqFlJA.png

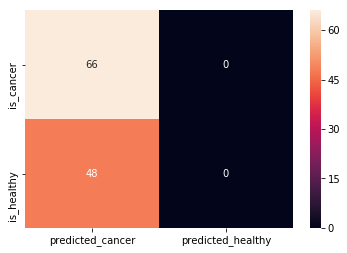






https://miro.medium.com/max/875/1*2xteBfna1URrrMp8ea2EBQ.png





# performing linear algebra

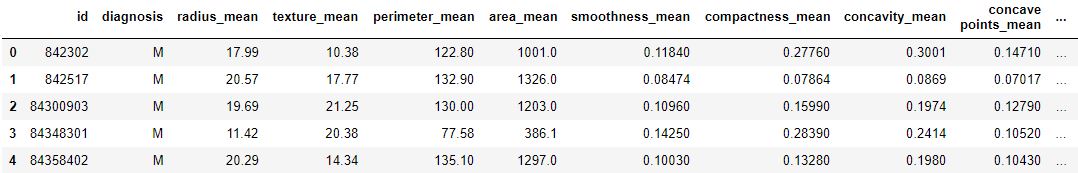
import numpy as np

# data processing

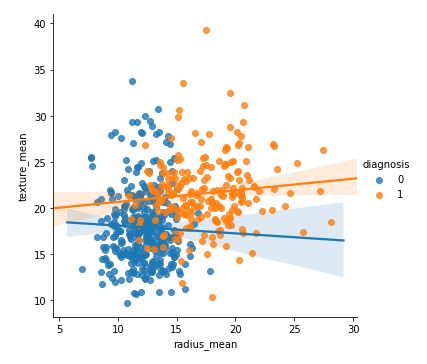
import pandas as pd

# visualisation

import matplotlib.pyplot as plt



sns.lmplot(x = 'radius\_mean', y = 'texture\_mean', hue = 'diagnosis', data = df)



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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size = 0.33, random\_state = 42)

knn = KNeighborsClassifier(n\_neighbors = 13)

knn.fit(X\_train, y\_train)

knn.score(X\_test, y\_test)

neighbors = []

cv\_scores = []

from sklearn.model\_selection import cross\_val\_score

# perform 10 fold cross validation

for k in range(1, 51, 2):

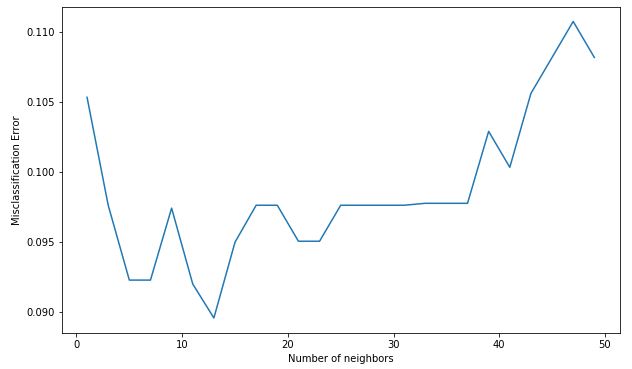
neighbors.append(k)

knn = KNeighborsClassifier(n\_neighbors = k)

scores = cross\_val\_score(

knn, X\_train, y\_train, cv = 10, scoring = 'accuracy')

cv\_scores.append(scores.mean())



**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn.datasets **import** load\_breast\_cancer;

cancer **=** load\_breast\_cancer();

X **=** cancer**.**data;

y **=** cancer**.**target;

*#data = np.c\_[cancer.data, cancer.target];*

*#columns = np.append(cancer.feature\_names, ["target"]);*

*#cancer.df = pd.DataFrame(data, columns = columns);*

*#print(cancer.df.head());*

*#print("\ncancer.df.shape = ",cancer.df.shape);*

**1. Divide the data into train (80%) and test (20%)**

In [3]:

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

X\_train , X\_test, y\_train, y\_test **=** train\_test\_split(X,

y,

test\_size **=** 0.2,

random\_state **=** 0);

print(X\_train**.**shape);

print(y\_train**.**shape);

print("\r\n");

print(X\_test**.**shape);

print(y\_test**.**shape);

(455, 30)

(455,)

(114, 30)

(114,)

**2. *Decision Tree Classifier* using all the features of the data. Model tested on the test data**

In [4]:

**from** sklearn.tree **import** DecisionTreeClassifier;

tree **=** DecisionTreeClassifier(criterion **=** 'entropy',

max\_depth **=** 3,

random\_state **=** 0 );

tree**.**fit(X\_train, y\_train)

Out[4]:

DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='entropy',

max\_depth=3, max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort='deprecated',

random\_state=0, splitter='best')

In [5]:

**from** sklearn.tree **import** plot\_tree;

plot\_tree(tree,

feature\_names **=** cancer**.**feature\_names,

fontsize **=** 7 )

Out[5]:

[Text(193.15384615384616, 190.26, 'worst concave points <= 0.142\nentropy = 0.945\nsamples = 455\nvalue = [165, 290]'),

Text(103.01538461538462, 135.9, 'worst perimeter <= 107.75\nentropy = 0.399\nsamples = 303\nvalue = [24, 279]'),

Text(51.50769230769231, 81.53999999999999, 'symmetry error <= 0.016\nentropy = 0.133\nsamples = 269\nvalue = [5, 264]'),

Text(25.753846153846155, 27.180000000000007, 'entropy = 0.337\nsamples = 80\nvalue = [5, 75]'),

Text(77.26153846153846, 27.180000000000007, 'entropy = 0.0\nsamples = 189\nvalue = [0, 189]'),

Text(154.52307692307693, 81.53999999999999, 'worst texture <= 20.22\nentropy = 0.99\nsamples = 34\nvalue = [19, 15]'),

Text(128.76923076923077, 27.180000000000007, 'entropy = 0.0\nsamples = 11\nvalue = [0, 11]'),

Text(180.27692307692308, 27.180000000000007, 'entropy = 0.667\nsamples = 23\nvalue = [19, 4]'),

Text(283.2923076923077, 135.9, 'worst perimeter <= 112.8\nentropy = 0.375\nsamples = 152\nvalue = [141, 11]'),

Text(257.53846153846155, 81.53999999999999, 'worst texture <= 27.575\nentropy = 0.948\nsamples = 30\nvalue = [19, 11]'),

Text(231.7846153846154, 27.180000000000007, 'entropy = 0.964\nsamples = 18\nvalue = [7, 11]'),

Text(283.2923076923077, 27.180000000000007, 'entropy = 0.0\nsamples = 12\nvalue = [12, 0]'),

Text(309.04615384615386, 81.53999999999999, 'entropy = 0.0\nsamples = 122\nvalue = [122, 0]')]

In [6]:

**from** sklearn.metrics **import** accuracy\_score;

y\_pred\_train **=** tree**.**predict(X\_train);

print("Train Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train))

y\_pred\_test **=** tree**.**predict(X\_test);

print("Test Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test))

Train Set Accuracy : 0.9648351648351648

Test Set Accuracy : 0.9473684210526315

**3. Model Run for trees of *depth 1, 2, 3, 4, 5, and 6* and for the *Gini Impurity* and *Entropy impurity* measures for each tree depth.**

**3.1 Depth = 1**

In [7]:

*# GINI IMPURITY*

tree\_gin\_d1 **=** DecisionTreeClassifier(criterion **=** 'gini',

max\_depth **=** 1,

random\_state **=** 0 );

tree\_gin\_d1**.**fit(X\_train, y\_train)

y\_pred\_train\_gin\_d1 **=** tree\_gin\_d1**.**predict(X\_train);

y\_pred\_test\_gin\_d1 **=** tree\_gin\_d1**.**predict(X\_test);

*# ENTROPY IMPURITY*

tree\_ent\_d1 **=** DecisionTreeClassifier(criterion **=** 'entropy',

max\_depth **=** 1,

random\_state **=** 0 );

tree\_ent\_d1**.**fit(X\_train, y\_train)

y\_pred\_train\_ent\_d1 **=** tree\_ent\_d1**.**predict(X\_train);

y\_pred\_test\_ent\_d1 **=** tree\_ent\_d1**.**predict(X\_test);

**3.2 Depth = 2**

In [8]:

*# GINI IMPURITY*

tree\_gin\_d2 **=** DecisionTreeClassifier(criterion **=** 'gini',

max\_depth **=** 2,

random\_state **=** 0);

tree\_gin\_d2**.**fit(X\_train, y\_train)

y\_pred\_train\_gin\_d2 **=** tree\_gin\_d2**.**predict(X\_train);

y\_pred\_test\_gin\_d2 **=** tree\_gin\_d2**.**predict(X\_test);

*# ENTROPY IMPURITY*

tree\_ent\_d2 **=** DecisionTreeClassifier(criterion **=** 'entropy',

max\_depth **=** 2,

random\_state **=** 0 );

tree\_ent\_d2**.**fit(X\_train, y\_train)

y\_pred\_train\_ent\_d2 **=** tree\_ent\_d2**.**predict(X\_train);

y\_pred\_test\_ent\_d2 **=** tree\_ent\_d2**.**predict(X\_test);

**3.3 Depth = 3**

In [9]:

*# GINI IMPURITY*

tree\_gin\_d3 **=** DecisionTreeClassifier(criterion **=** 'gini',

max\_depth **=** 3,

random\_state **=** 0 );

tree\_gin\_d3**.**fit(X\_train, y\_train)

y\_pred\_train\_gin\_d3 **=** tree\_gin\_d3**.**predict(X\_train);

y\_pred\_test\_gin\_d3 **=** tree\_gin\_d3**.**predict(X\_test);

*# ENTROPY IMPURITY*

tree\_ent\_d3 **=** DecisionTreeClassifier(criterion **=** 'entropy',

max\_depth **=** 3,

random\_state **=** 0 );

tree\_ent\_d3**.**fit(X\_train, y\_train)

y\_pred\_train\_ent\_d3 **=** tree\_ent\_d3**.**predict(X\_train);

y\_pred\_test\_ent\_d3 **=** tree\_ent\_d3**.**predict(X\_test);

**3.4 Depth = 4**

In [10]:

*# GINI IMPURITY*

tree\_gin\_d4 **=** DecisionTreeClassifier(criterion **=** 'gini',

max\_depth **=** 4,

random\_state **=** 0 );

tree\_gin\_d4**.**fit(X\_train, y\_train)

y\_pred\_train\_gin\_d4 **=** tree\_gin\_d4**.**predict(X\_train);

y\_pred\_test\_gin\_d4 **=** tree\_gin\_d4**.**predict(X\_test);

*# ENTROPY IMPURITY*

tree\_ent\_d4 **=** DecisionTreeClassifier(criterion **=** 'entropy',

max\_depth **=** 4,

random\_state **=** 0 );

tree\_ent\_d4**.**fit(X\_train, y\_train)

y\_pred\_train\_ent\_d4 **=** tree\_ent\_d4**.**predict(X\_train);

y\_pred\_test\_ent\_d4 **=** tree\_ent\_d4**.**predict(X\_test);

**3.5 Depth = 5**

In [11]:

*# GINI IMPURITY*

tree\_gin\_d5 **=** DecisionTreeClassifier(criterion **=** 'gini',

max\_depth **=** 5,

random\_state **=** 0 );

tree\_gin\_d5**.**fit(X\_train, y\_train)

y\_pred\_train\_gin\_d5 **=** tree\_gin\_d5**.**predict(X\_train);

y\_pred\_test\_gin\_d5 **=** tree\_gin\_d5**.**predict(X\_test);

*# ENTROPY IMPURITY*

tree\_ent\_d5 **=** DecisionTreeClassifier(criterion **=** 'entropy',

max\_depth **=** 5,

random\_state **=** 0 );

tree\_ent\_d5**.**fit(X\_train, y\_train)

y\_pred\_train\_ent\_d5 **=** tree\_ent\_d5**.**predict(X\_train);

y\_pred\_test\_ent\_d5 **=** tree\_ent\_d5**.**predict(X\_test);

**3.6 Depth = 6**

In [12]:

*# GINI IMPURITY*

tree\_gin\_d6 **=** DecisionTreeClassifier(criterion **=** 'gini',

max\_depth **=** 6,

random\_state **=** 0 );

tree\_gin\_d6**.**fit(X\_train, y\_train)

y\_pred\_train\_gin\_d6 **=** tree\_gin\_d6**.**predict(X\_train);

y\_pred\_test\_gin\_d6 **=** tree\_gin\_d6**.**predict(X\_test);

*# ENTROPY IMPURITY*

tree\_ent\_d6 **=** DecisionTreeClassifier(criterion **=** 'entropy',

max\_depth **=** 6,

random\_state **=** 0 );

tree\_ent\_d6**.**fit(X\_train, y\_train)

y\_pred\_train\_ent\_d6 **=** tree\_ent\_d6**.**predict(X\_train);

y\_pred\_test\_ent\_d6 **=** tree\_ent\_d6**.**predict(X\_test);

**3.7 Comparison**

In [13]:

print("\r\nDEPTH = 1");

print("\tGINI : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_gin\_d1));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_gin\_d1));

print("\tENTROPY : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_ent\_d1));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_ent\_d1));

print("\r\nDEPTH = 2");

print("\tGINI : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_gin\_d2));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_gin\_d2));

print("\tENTROPY : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_ent\_d2));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_ent\_d2));

print("\r\nDEPTH = 3");

print("\tGINI : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_gin\_d3));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_gin\_d3));

print("\tENTROPY : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_ent\_d3));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_ent\_d3));

print("\r\nDEPTH = 4");

print("\tGINI : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_gin\_d4));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_gin\_d4));

print("\tENTROPY : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_ent\_d4));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_ent\_d4));

print("\r\nDEPTH = 5");

print("\tGINI : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_gin\_d5));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_gin\_d5));

print("\tENTROPY : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_ent\_d5));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_ent\_d5));

print("\r\nDEPTH = 6");

print("\tGINI : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_gin\_d6));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_gin\_d6));

print("\tENTROPY : ");

print("\t\tTrain Set Accuracy : ", accuracy\_score(y\_train, y\_pred\_train\_ent\_d6));

print("\t\tTest Set Accuracy : ", accuracy\_score(y\_test, y\_pred\_test\_ent\_d6));

DEPTH = 1

GINI :

Train Set Accuracy : 0.9230769230769231

Test Set Accuracy : 0.9035087719298246

ENTROPY :

Train Set Accuracy : 0.9230769230769231

Test Set Accuracy : 0.9035087719298246

DEPTH = 2

GINI :

Train Set Accuracy : 0.9516483516483516

Test Set Accuracy : 0.9649122807017544

ENTROPY :

Train Set Accuracy : 0.9318681318681319

Test Set Accuracy : 0.9210526315789473

DEPTH = 3

GINI :

Train Set Accuracy : 0.9714285714285714

Test Set Accuracy : 0.9649122807017544

ENTROPY :

Train Set Accuracy : 0.9648351648351648

Test Set Accuracy : 0.9473684210526315

DEPTH = 4

GINI :

Train Set Accuracy : 0.9846153846153847

Test Set Accuracy : 0.956140350877193

ENTROPY :

Train Set Accuracy : 0.9868131868131869

Test Set Accuracy : 0.9385964912280702

DEPTH = 5

GINI :

Train Set Accuracy : 0.989010989010989

Test Set Accuracy : 0.9473684210526315

ENTROPY :

Train Set Accuracy : 0.9912087912087912

Test Set Accuracy : 0.9210526315789473

DEPTH = 6

GINI :

Train Set Accuracy : 0.9934065934065934

Test Set Accuracy : 0.9473684210526315

ENTROPY :

Train Set Accuracy : 0.9934065934065934

Test Set Accuracy : 0.9035087719298246

In [14]:

**%matplotlib** inline

**import** matplotlib.pyplot **as** plt

plt**.**style**.**use("dark\_background")

fig **=** plt**.**figure();

ax **=** plt**.**axes();

ax**.**set\_xlabel('Depth - 1');

ax**.**set\_ylabel('Accuracy');

ax**.**set\_title('Accuracy of Test and Train Datasets at Varying Depths');

train\_gin\_accuracy\_lst **=** [accuracy\_score(y\_train, y\_pred\_train\_gin\_d1),

accuracy\_score(y\_train, y\_pred\_train\_gin\_d2),

accuracy\_score(y\_train, y\_pred\_train\_gin\_d3),

accuracy\_score(y\_train, y\_pred\_train\_gin\_d4),

accuracy\_score(y\_train, y\_pred\_train\_gin\_d5),

accuracy\_score(y\_train, y\_pred\_train\_gin\_d6)];

test\_gin\_accuracy\_lst **=** [accuracy\_score(y\_test, y\_pred\_test\_gin\_d1),

accuracy\_score(y\_test, y\_pred\_test\_gin\_d2),

accuracy\_score(y\_test, y\_pred\_test\_gin\_d3),

accuracy\_score(y\_test, y\_pred\_test\_gin\_d4),

accuracy\_score(y\_test, y\_pred\_test\_gin\_d5),

accuracy\_score(y\_test, y\_pred\_test\_gin\_d6)];

train\_ent\_accuracy\_lst **=** [accuracy\_score(y\_train, y\_pred\_train\_ent\_d1),

accuracy\_score(y\_train, y\_pred\_train\_ent\_d2),

accuracy\_score(y\_train, y\_pred\_train\_ent\_d3),

accuracy\_score(y\_train, y\_pred\_train\_ent\_d4),

accuracy\_score(y\_train, y\_pred\_train\_ent\_d5),

accuracy\_score(y\_train, y\_pred\_train\_ent\_d6)];

test\_ent\_accuracy\_lst **=** [accuracy\_score(y\_test, y\_pred\_test\_ent\_d1),

accuracy\_score(y\_test, y\_pred\_test\_ent\_d2),

accuracy\_score(y\_test, y\_pred\_test\_ent\_d3),

accuracy\_score(y\_test, y\_pred\_test\_ent\_d4),

accuracy\_score(y\_test, y\_pred\_test\_ent\_d5),

accuracy\_score(y\_test, y\_pred\_test\_ent\_d6)];

plt**.**plot(train\_gin\_accuracy\_lst, label **=** "Train GINI Accuracies");

plt**.**plot(test\_gin\_accuracy\_lst, label **=** "Test GINI Accuracies");

plt**.**plot(train\_ent\_accuracy\_lst, label **=** "Train ENTROPY Accuracies");

plt**.**plot(test\_gin\_accuracy\_lst, label **=** "Test ENTROPY Accuracies");

plt**.**legend(loc **=** "lower right", frameon **=** **True**);