ComS574_HW4

April 3, 2020

ComS 574

HW 4

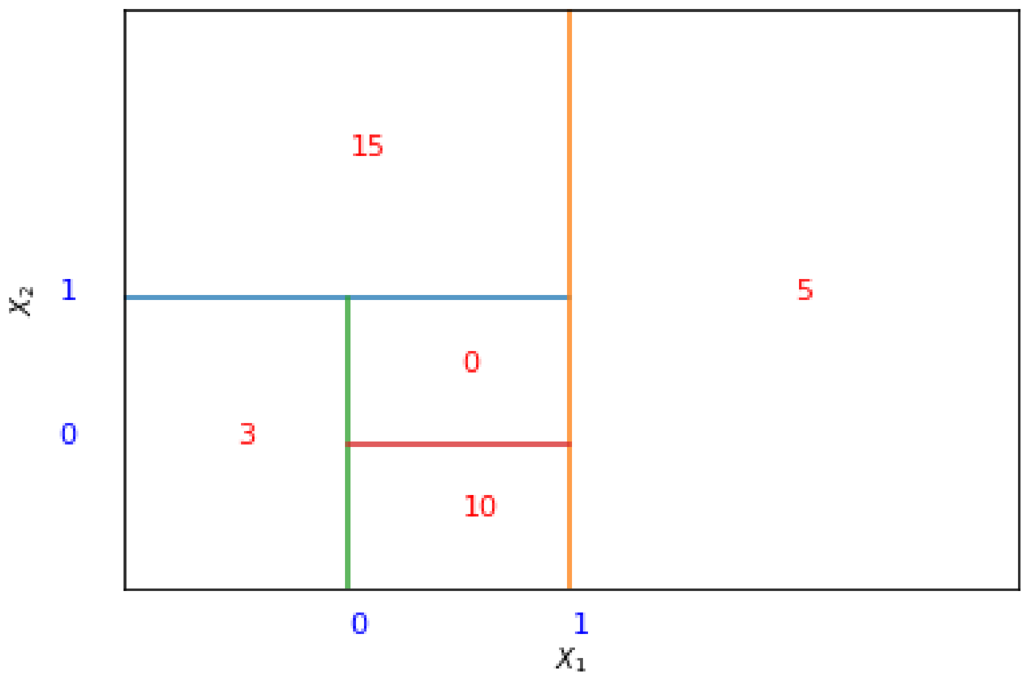
Kanak Choudhury

1 Problem 1

1.1 (a)

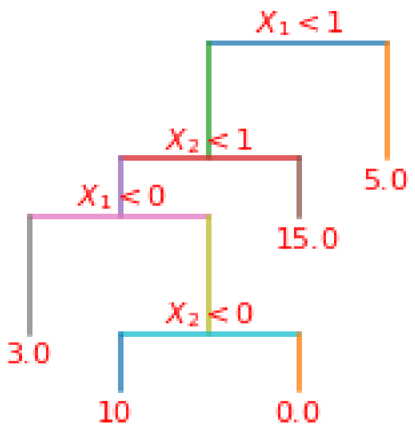
[237]:
```python
import numpy as np
from matplotlib import pyplot as plt
import turtle
```

[238]:
```python
plt.plot()
plt.xlim(-1,3)
plt.ylim(-1,3)
plt.plot([-1,1], [1,1])
plt.plot([1,1], [-1,3])
plt.plot([0,0], [-1,1])
plt.plot([0,1], [0,0])
plt.text(2, 1, '$5$', color = 'red', fontsize=11)
plt.text(.5, -.5, '$10$', color = 'r', fontsize=11)
plt.text(.5, .5, '$0$', color = 'r', fontsize=11)
plt.text(-.5, 0, '$3$', color = 'r', fontsize=11)
plt.text(0, 2, '$15$', color = 'r', fontsize=11)
plt.text(0, -1.3, '$0$', color = 'b', fontsize=11)
plt.text(1, -1.3, '$1$', color = 'b', fontsize=11)
plt.text(-1.3, 0, '$0$', color = 'b', fontsize=11)
plt.text(-1.3, 1, '$1$', color = 'b', fontsize=11)

plt.xlabel('$X_1$', labelpad=20)
plt.ylabel('$X_2$', labelpad=30)
plt.xticks([])
plt.yticks([])
plt.show()
```



1.2 (b)

[239]:
```
plt.plot()
plt.xlim(0,10)
plt.ylim(0,10)
plt.text(4.5, 8.2, '$X_1 < 1$', color = 'red', fontsize=11)
plt.plot([4,6], [8,8])
plt.plot([6,6], [8,6])
plt.text(5.7, 5.5, '$5.0$', color = 'red', fontsize=11)

plt.plot([4,4], [8,6])
plt.text(3.5, 6.2, '$X_2 < 1$', color = 'red', fontsize=11)
plt.plot([3,5], [6,6])
plt.plot([3,3], [6,5])
plt.plot([5,5], [6,5])
plt.text(4.7, 4.5, '$15.0$', color = 'red', fontsize=11)

plt.plot([2,4], [5,5])
plt.text(2.5, 5.2, '$X_1 < 0$', color = 'red', fontsize=11)
plt.plot([2,2], [5,3])
plt.text(1.7, 2.5, '$3.0$', color = 'red', fontsize=11)

plt.plot([4,4], [5,3])
plt.text(3.5, 3.2, '$X_2 < 0$', color = 'red', fontsize=11)
plt.plot([3,5], [3,3])
plt.plot([3,3], [3,2])
plt.text(2.7, 1.5, '$10$', color = 'red', fontsize=11)
plt.plot([5,5], [3,2])
plt.text(4.7, 1.5, '$0.0$', color = 'red', fontsize=11)
plt.xticks([])
plt.yticks([])
plt.box(False)
plt.show()
```



2 Problem 2

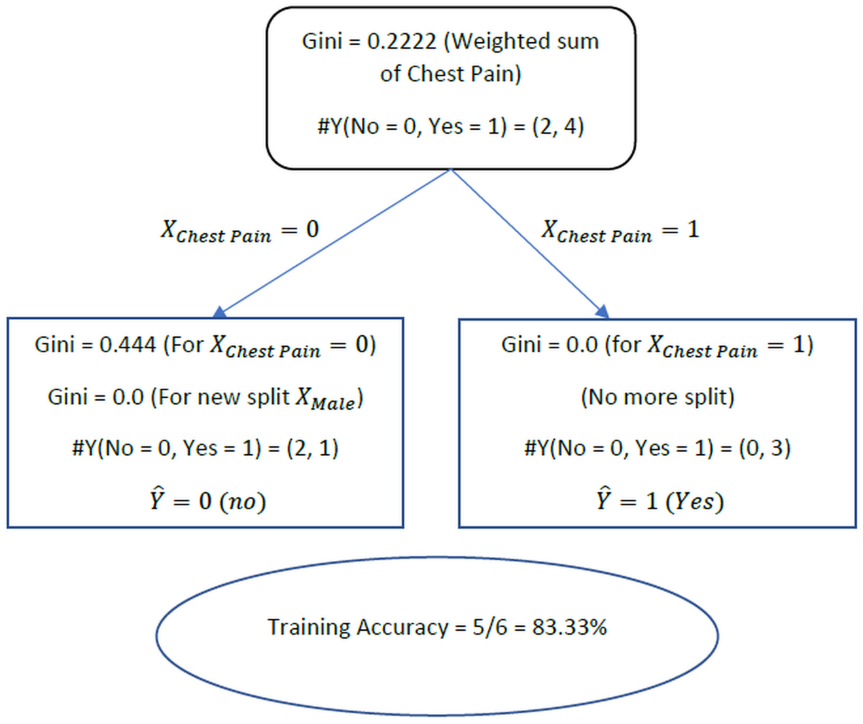
[240]:
```
prob_red = [.1,.15,.2,.2,.55,.6,.6,.65,.7,.75]
avg1 = sum(prob_red)/len(prob_red)
mej1 = sum([1 for i in prob_red if i >= .5])
avg = 'Red' if avg1>=.5 else 'Green'
mej = 'Red' if mej1>= (len(prob_red)/2) else 'Green'
print('Final Classification using Average: \"'+ avg +'\"   Since, average␣
 ↪probability, %.2f is greater than equal to 0.5' %avg1)
print('Final Classification using Majority Vote: \"'+ mej +'\"   Since, '+␣
 ↪mej+' count, %d is greater than equal to %d' %(mej1, len(prob_red)/2))
```

```
Final Classification using Average: "Green"   Since, average probability, 0.45
is greater than equal to 0.5
Final Classification using Majority Vote: "Red"   Since, Red count, 6 is greater
than equal to 5
```

3 Problem 3

[241]:
```python
from IPython.display import display, Image
from PIL import Image as imm

img_org = imm.open('D:/ISU/COMS 574 - Introduction to Machine Learning/HW/HW4/
↪prob3.png')
width_org, height_org = img_org.size
factor = 2
width = int(width_org * factor)
height = int(height_org * factor)
img_anti = img_org.resize((width, height), imm.ANTIALIAS)
display(img_anti)
```

PATIENT ID	CHEST PAIN	MALE	SMOKES	EXERCISES	HEART ATTACK
1	yes	yes	no	yes	yes
2	yes	yes	yes	no	yes
3	no	no	yes	no	yes
4	no	yes	no	yes	no
5	yes	no	yes	yes	yes
6	no	yes	yes	yes	no
P(yes)	0.500	0.667	0.667	0.667	0.667
P(no)	0.500	0.333	0.333	0.333	0.333
P(yes & HA = yes)	1.000	0.500	0.750	0.500	
P(yes & HA = no)	0.000	0.500	0.250	0.500	
P(no & HA = yes)	0.333	1.000	0.500	1.000	
P(no & HA = no)	0.667	0.000	0.500	0.000	
Gini Index (yes) = $\sum_{k=1}^{K}\hat{p}_{mk}(1-\hat{p}_{mk})$	0.000	0.500	0.375	0.500	
Gini Index (no) = $\sum_{k=1}^{K}\hat{p}_{mk}(1-\hat{p}_{mk})$	0.444	0.000	0.500	0.000	
Weighted sum of the Gini Indices = $P(yes) \times GI(Yes) + P(no) \times GI(no)$	0.222	0.333	0.417	0.333	

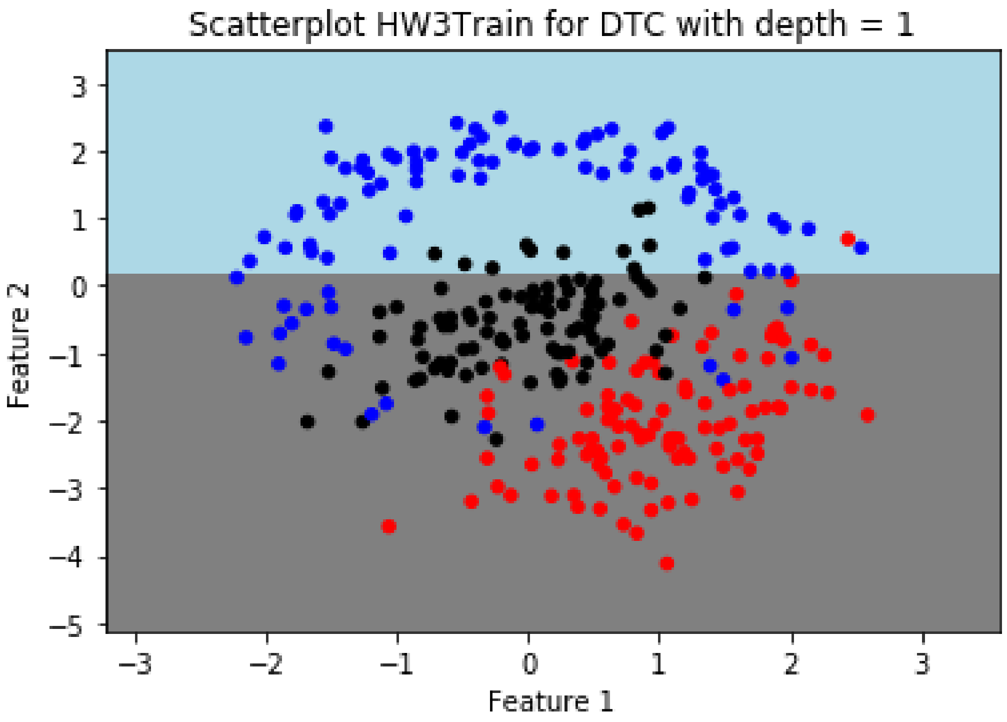

Gini = 0.2222 (Weighted sum of Chest Pain)
#Y(No = 0, Yes = 1) = (2, 4)
$X_{Chest\ Pain} = 0$
$X_{Chest\ Pain} = 1$
Gini = 0.444 (For $X_{Chest\ Pain} = 0$)
Gini = 0.0 (For new split X_{Male})
#Y(No = 0, Yes = 1) = (2, 1)
$\hat{Y} = 0\ (no)$
Gini = 0.0 (for $X_{Chest\ Pain} = 1$)
(No more split)
#Y(No = 0, Yes = 1) = (0, 3)
$\hat{Y} = 1\ (Yes)$
Training Accuracy = 5/6 = 83.33%

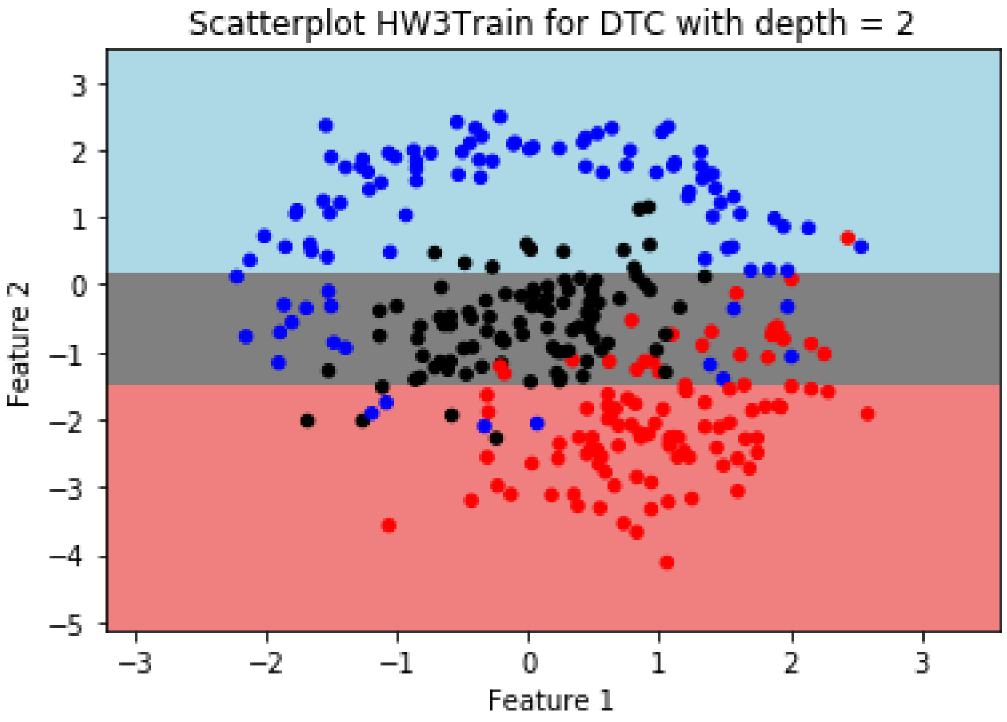
4 Problem 4

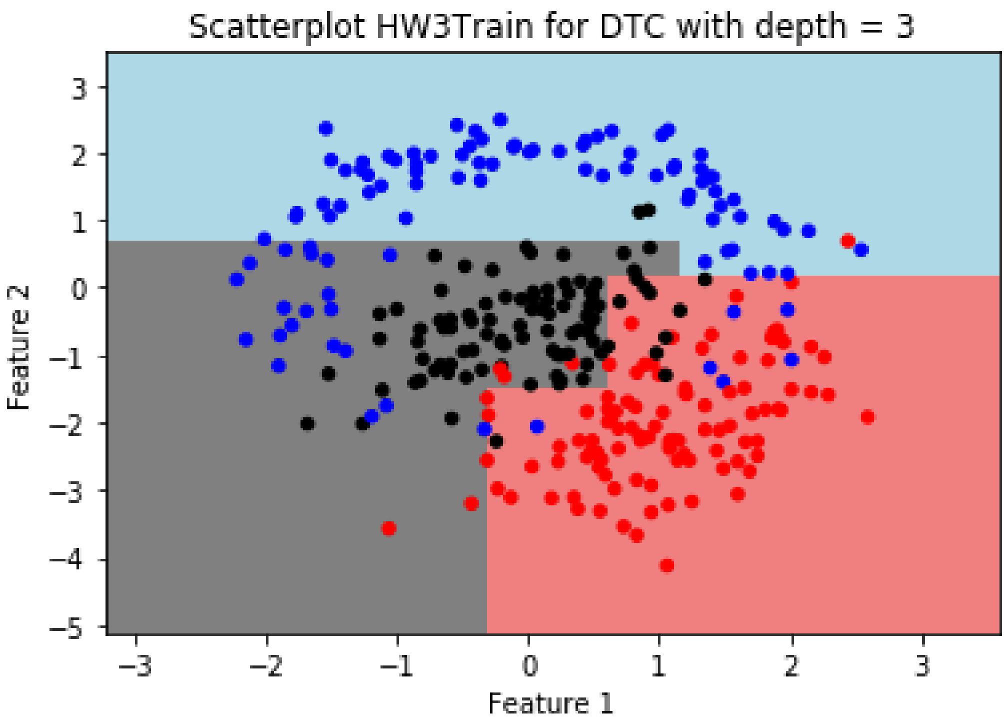
4.1 A.

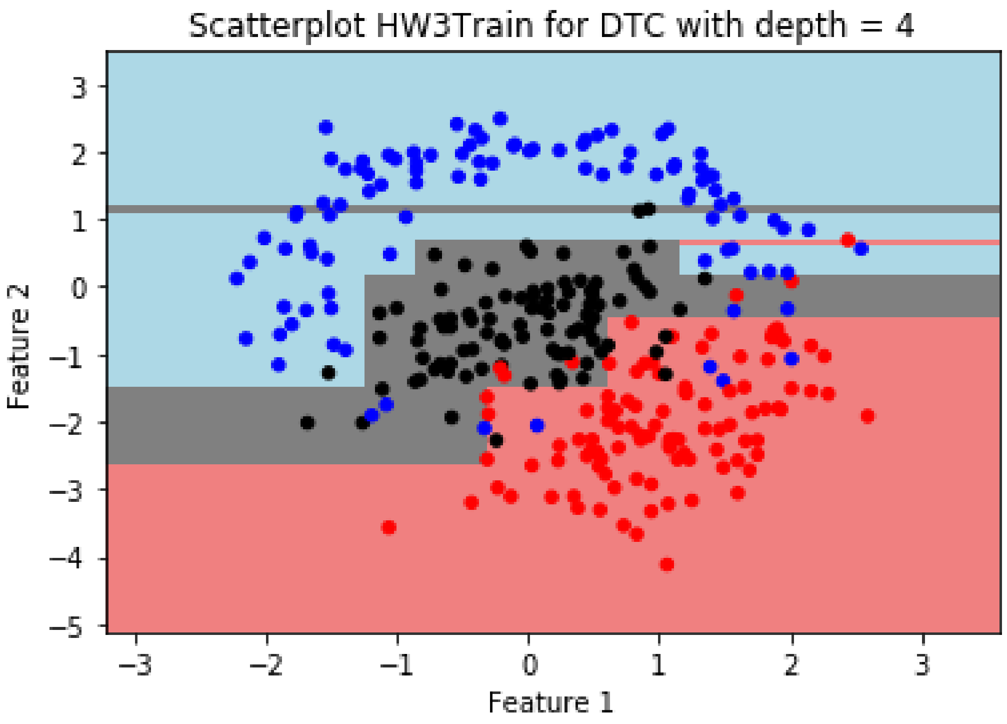
[242]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

import sys
import gc
from itertools import product

import warnings
from sklearn.exceptions import ConvergenceWarning
warnings.simplefilter("ignore", ConvergenceWarning)

def plot_scatter(x_df, y_df, title=None):
    for x,y in zip(x_df, y_df):
    #     print(x1,x2,y)
        if y==1:
            col = 'blue'
        if y==2:
            col = 'red'
        if y==3:
            col = 'black'
        plt.scatter(x[0], x[1],  color=col)

    plt.title(title)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show()

def plot_func(x_df, y_df, pred, title = None):

    pred = pred.reshape(x1mesh.shape)

    plt.figure()
    plt.pcolormesh(x1mesh, x2mesh, pred, cmap=cmap_light)
    ytrain_colors = [y-1 for y in y_df]
    plt.scatter(x_df[:, 0], x_df[:, 1], c=ytrain_colors, cmap=cmap_bold, s=20)
    plt.xlim(x1_min, x1_max)
    plt.ylim(x2_min, x2_max)
    plt.title(title)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show()
```

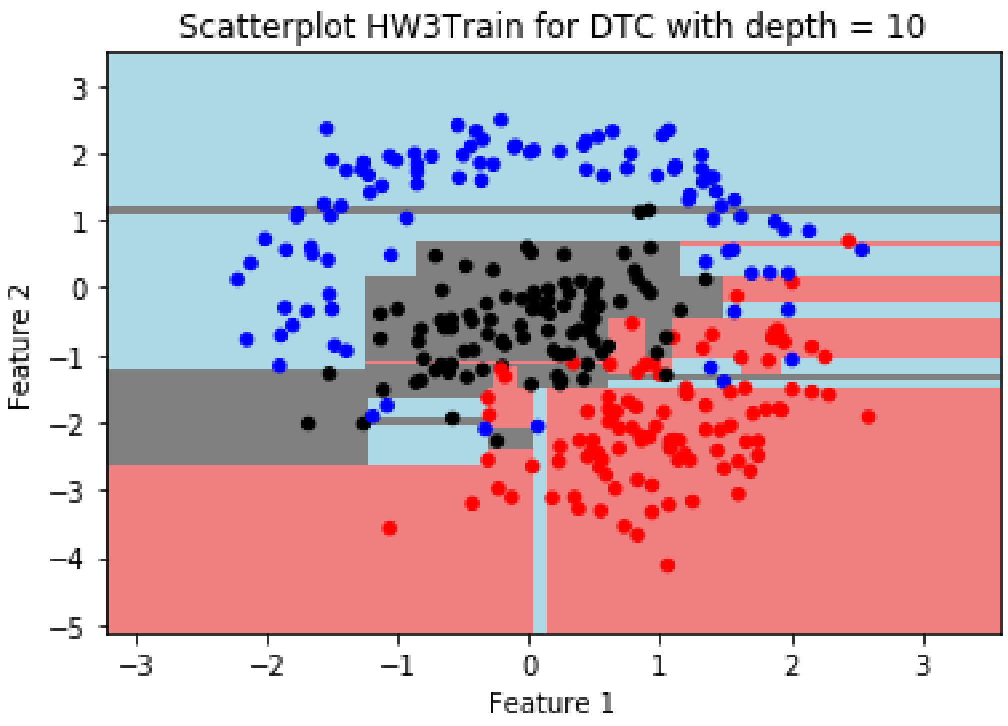
[243]:
```
path = 'D:/ISU/COMS 574 - Introduction to Machine Learning/HW/HW3/'


df_train = pd.read_csv(path + 'HW3train.csv', sep=',',
                       header=None, names=['Y', 'X1', 'X2'])
df_test = pd.read_csv(path + 'HW3test.csv', sep=',',
                      header=None, names=['Y', 'X1', 'X2'])
tr_size = df_train.shape
ts_size = df_test.shape

x_train = np.array(df_train[['X1', 'X2']])
y_train = np.array(df_train['Y'])

x_test = np.array(df_test[['X1', 'X2']])
y_test = np.array(df_test['Y'])


# plot_scatter(x_train, y_train, title = 'Scatter plot of X1 and X2 for Train␣
 ↪Data')

# plot_scatter(x_test, y_test, title = 'Scatter plot of X1 and X2 for Test␣
 ↪Data')


h = .03
x1_min, x1_max = x_train[:, 0].min() - 1, x_train[:, 0].max() + 1
x2_min, x2_max = x_train[:, 1].min() - 1, x_train[:, 1].max() + 1
x1mesh, x2mesh = np.meshgrid(np.arange(x1_min, x1_max, h), np.arange(x2_min,␣
 ↪x2_max, h))

cmap_light = ListedColormap(['lightblue', 'lightcoral', 'grey'])
cmap_bold = ListedColormap(['blue', 'red', 'black'])

for i in [1,2,3,4,10]:
    tree=DecisionTreeClassifier(criterion='gini', splitter='best', max_depth= i)
    model_tree = tree.fit(x_train, y_train)
    Z = model_tree.predict(np.c_[x1mesh.ravel(), x2mesh.ravel()])

    plot_func(x_train, y_train, Z, title = 'Scatterplot HW3Train for DTC with␣
 ↪depth = '+str(i))
```


Scatterplot HW3Train for DTC with depth = 1
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3


Scatterplot HW3Train for DTC with depth = 2
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3

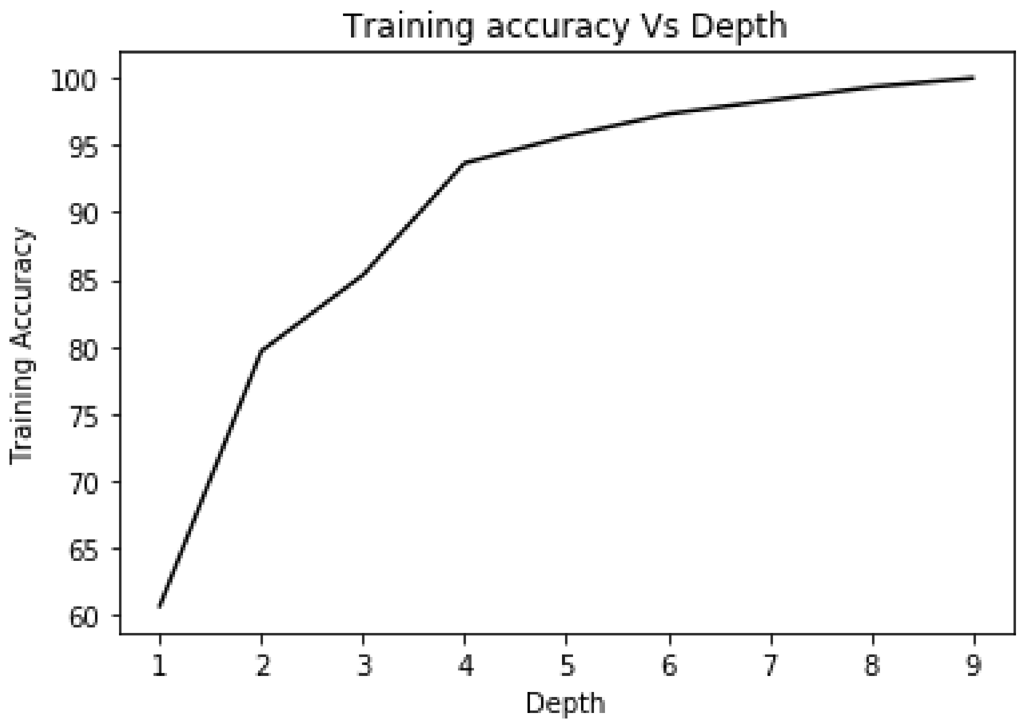

Scatterplot HW3Train for DTC with depth = 3
Feature 1
Feature 2

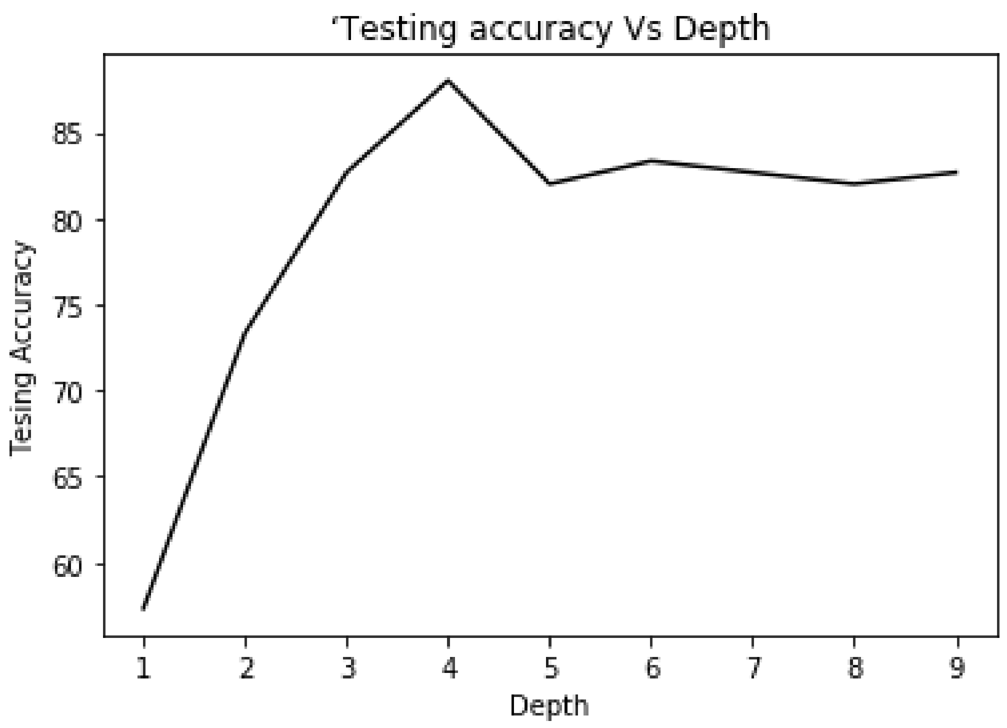

Scatterplot HW3Train for DTC with depth = 4
Feature 1
Feature 2



4.2 B.

[244]:
```
tr_acc_dt = []
ts_acc_dt = []
depth_dt = []
k = range(1,10)
for i in k:
    tree=DecisionTreeClassifier(criterion='gini', splitter='best', max_depth= i)
    model_tree = tree.fit(x_train, y_train)
    tr_acc_dt.append(model_tree.score(x_train, y_train)*100)
    ts_acc_dt.append(model_tree.score(x_test, y_test)*100)
    depth_dt.append(model_tree.get_depth())


plt.plot(depth_dt, tr_acc_dt, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(depth_dt, ts_acc_dt, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Tesing Accuracy")
plt.title("'Testing accuracy Vs Depth")
plt.show()

print('Based on maximum testing accuracy: depth = %d with training_accuracy = %.
↪2f and testing_accuracy = %.2f'
      %(np.argmax(ts_acc_dt)+1, tr_acc_dt[np.argmax(ts_acc_dt)], ts_acc_dt[np.
↪argmax(ts_acc_dt)]))
```



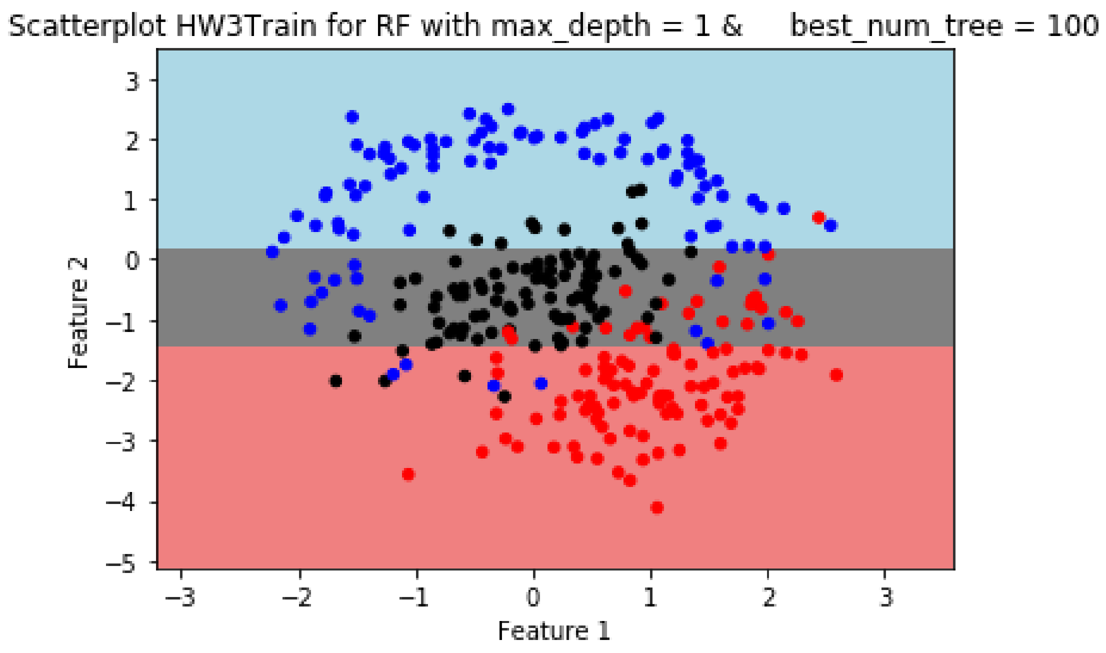


```
Based on maximum testing accuracy: depth = 4 with training_accuracy = 93.67 and
testing_accuracy = 88.00
```

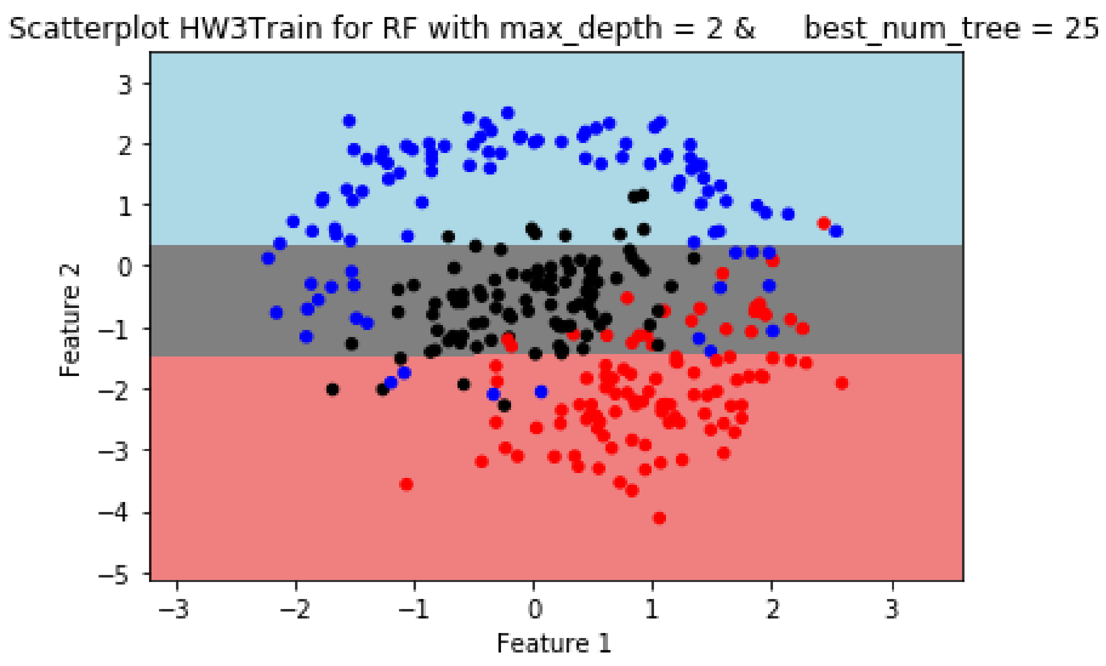
4.3 C.

[245]:
```python
numtreerange=[1,5,10,25,50,100,200]


for i in [1,2,3,4,10]:
    best_fit = None
    ts_acc1 = 0
    best_ts_acc = 0
    best_numtree = 0
    for j in numtreerange:
        rf=RandomForestClassifier(bootstrap=True, n_estimators=j,
                                  max_features=None, criterion='gini', max_depth=i)
        model_rf = rf.fit(x_train, y_train)
        ts_acc1 = model_rf.score(x_test, y_test)*100
        print('test_accuracy = %.2f, Num_trees = %d, Depth = %d' %(ts_acc1, j,␣
 ↪i))
        if (ts_acc1 > best_ts_acc):
            best_ts_acc = ts_acc1
            best_numtree = j
            best_fit = model_rf
    Z = best_fit.predict(np.c_[x1mesh.ravel(), x2mesh.ravel()])

    plot_func(x_train, y_train, Z, title = 'Scatterplot HW3Train for RF with␣
↪max_depth = %d & \
    best_num_tree = %d' %(i, best_numtree))
```

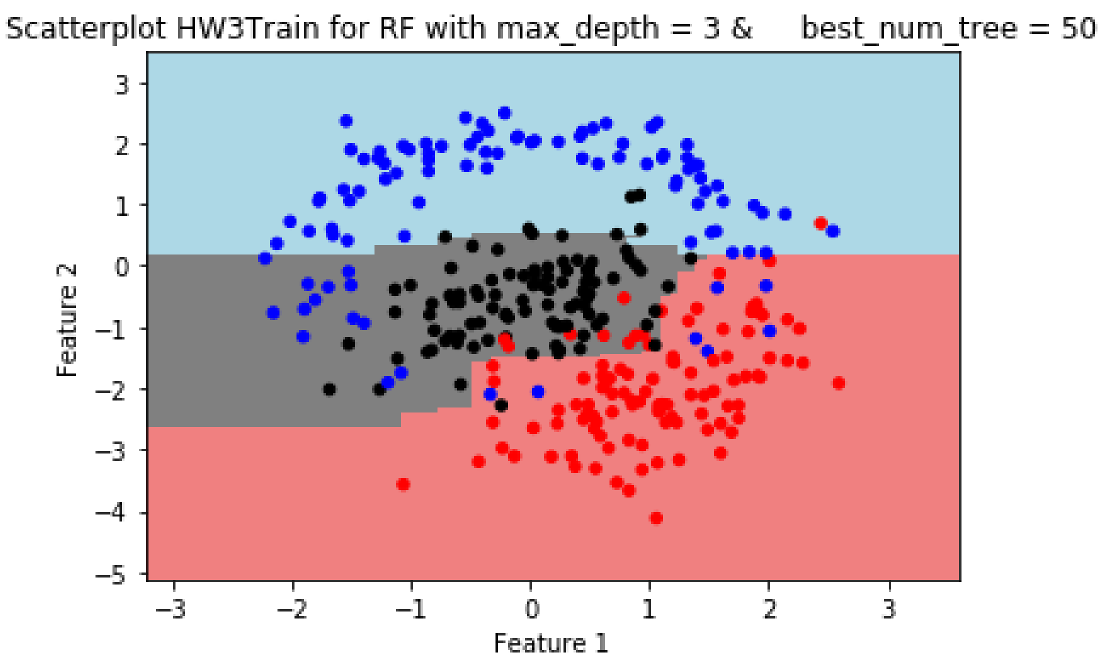
```
test_accuracy = 58.00, Num_trees = 1, Depth = 1
test_accuracy = 57.33, Num_trees = 5, Depth = 1
test_accuracy = 73.33, Num_trees = 10, Depth = 1
test_accuracy = 72.67, Num_trees = 25, Depth = 1
test_accuracy = 72.67, Num_trees = 50, Depth = 1
test_accuracy = 74.00, Num_trees = 100, Depth = 1
test_accuracy = 74.00, Num_trees = 200, Depth = 1
```



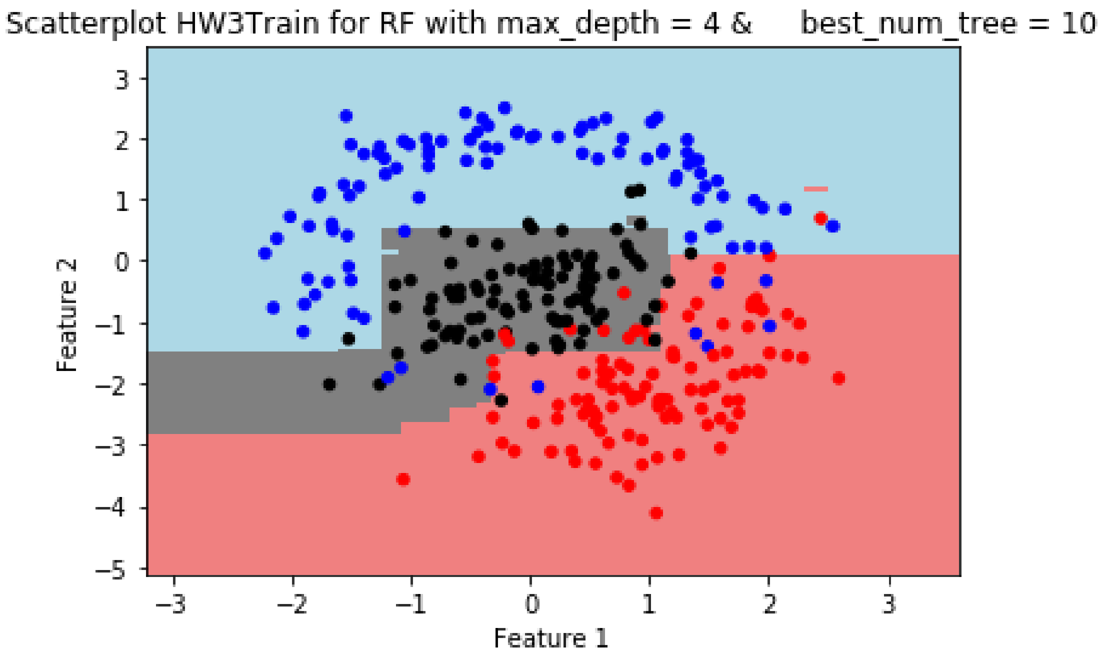
```
test_accuracy = 72.00, Num_trees = 1, Depth = 2
test_accuracy = 74.67, Num_trees = 5, Depth = 2
test_accuracy = 74.67, Num_trees = 10, Depth = 2
test_accuracy = 75.33, Num_trees = 25, Depth = 2
test_accuracy = 74.67, Num_trees = 50, Depth = 2
test_accuracy = 74.67, Num_trees = 100, Depth = 2
test_accuracy = 74.67, Num_trees = 200, Depth = 2
```



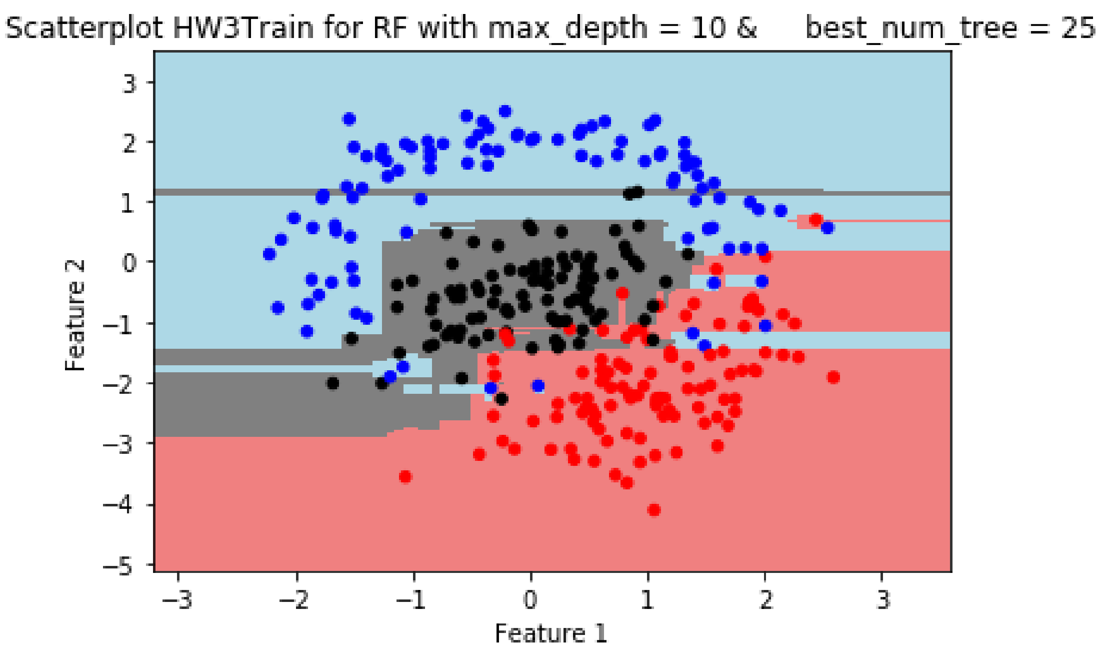
```
test_accuracy = 79.33, Num_trees = 1, Depth = 3
test_accuracy = 80.00, Num_trees = 5, Depth = 3
test_accuracy = 82.67, Num_trees = 10, Depth = 3
test_accuracy = 82.00, Num_trees = 25, Depth = 3
test_accuracy = 83.33, Num_trees = 50, Depth = 3
test_accuracy = 82.00, Num_trees = 100, Depth = 3
test_accuracy = 83.33, Num_trees = 200, Depth = 3
```



```
test_accuracy = 85.33, Num_trees = 1, Depth = 4
test_accuracy = 88.00, Num_trees = 5, Depth = 4
test_accuracy = 88.67, Num_trees = 10, Depth = 4
test_accuracy = 86.67, Num_trees = 25, Depth = 4
test_accuracy = 88.00, Num_trees = 50, Depth = 4
test_accuracy = 88.00, Num_trees = 100, Depth = 4
test_accuracy = 88.00, Num_trees = 200, Depth = 4
```



```
test_accuracy = 80.67, Num_trees = 1, Depth = 10
test_accuracy = 82.00, Num_trees = 5, Depth = 10
test_accuracy = 86.00, Num_trees = 10, Depth = 10
test_accuracy = 86.67, Num_trees = 25, Depth = 10
test_accuracy = 85.33, Num_trees = 50, Depth = 10
test_accuracy = 84.00, Num_trees = 100, Depth = 10
test_accuracy = 84.67, Num_trees = 200, Depth = 10
```

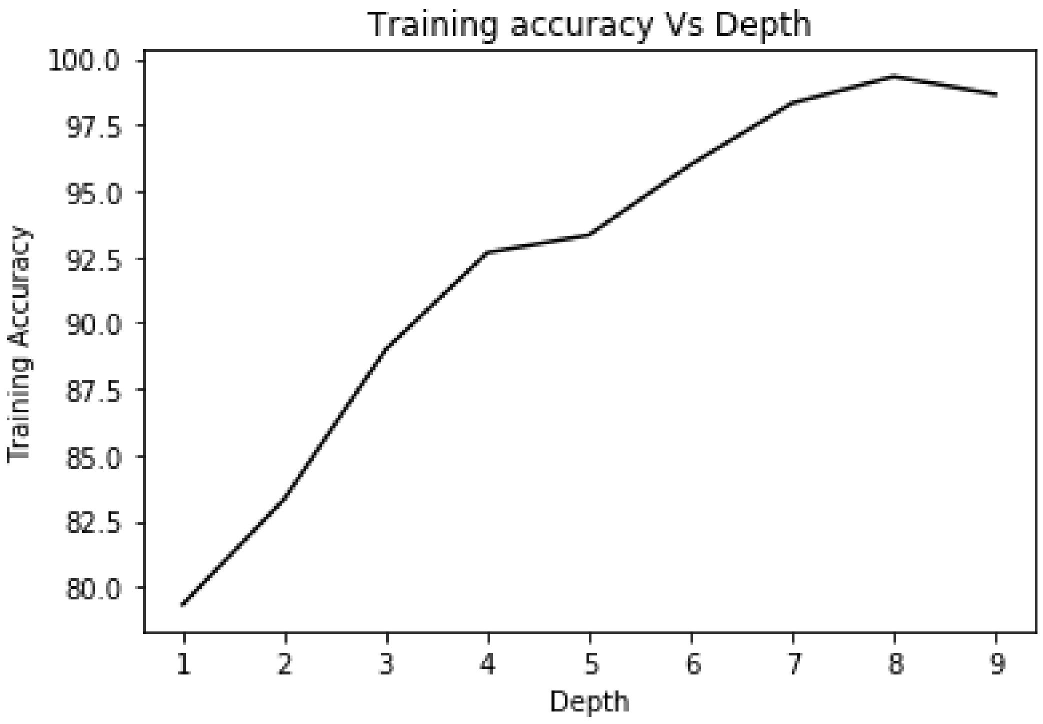


4.4 D.

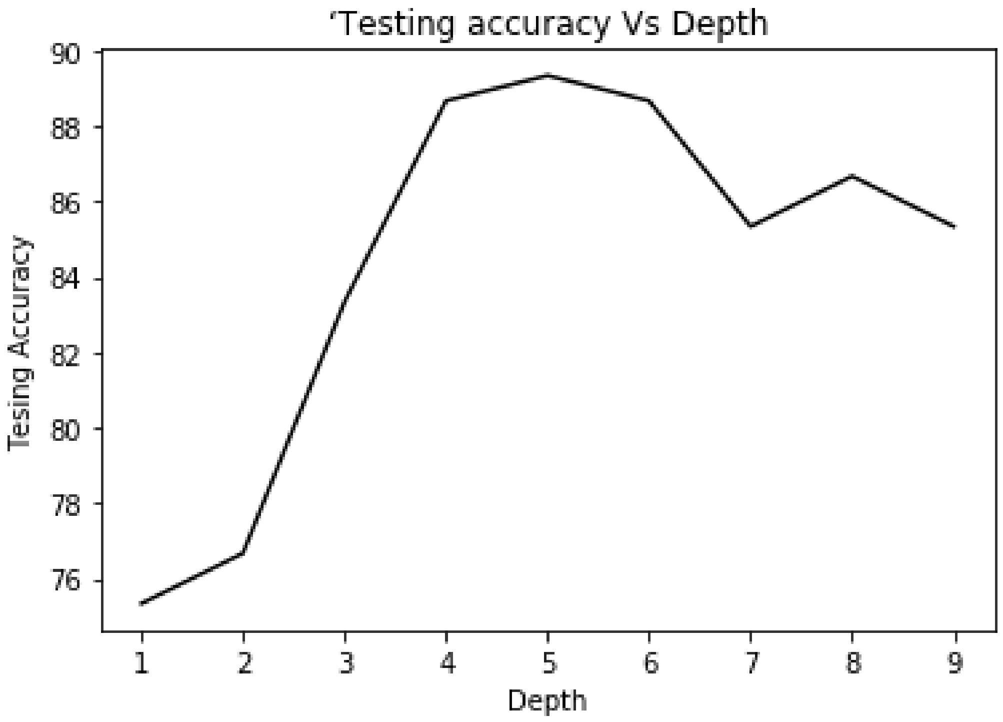
```
[246]: tr_acc_rf = []
ts_acc_rf = []
best_num_tree_rf = []
k = range(1,10)
for i in k:
    best_fit = None
    ts_acc1 = 0
    best_ts_acc = 0
    best_numtree = 0
    for j in numtreerange:
        rf=RandomForestClassifier(bootstrap=True, n_estimators=j,
                                  max_features=None, criterion='gini', max_depth=i)
        model_rf = rf.fit(x_train, y_train)
        ts_acc1 = model_rf.score(x_test, y_test)*100
        print('test_accuracy = %.2f, Num_trees = %d, Depth = %d' %(ts_acc1, j,␣
 ↪i))
        if (ts_acc1 > best_ts_acc):
            best_ts_acc = ts_acc1
            best_numtree = j
            best_fit = model_rf
    tr_acc_rf.append(best_fit.score(x_train, y_train)*100)
    ts_acc_rf.append(best_fit.score(x_test, y_test)*100)
    best_num_tree_rf.append(best_numtree)
```

```
plt.plot(k, tr_acc_rf, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(k, ts_acc_rf, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Tesing Accuracy")
plt.title("‘Testing accuracy Vs Depth")
plt.show()

print('Based on maximum test accuracy: depth = %d and best_num_tree = %d \
with training_accuracy = %.2f & testing_accuracy = %.2f'
      %(np.argmax(ts_acc_rf)+1, best_num_tree_rf[np.argmax(ts_acc_rf)],
        tr_acc_rf[np.argmax(ts_acc_rf)], ts_acc_rf[np.argmax(ts_acc_rf)]))
```

```
test_accuracy = 54.00, Num_trees = 1, Depth = 1
test_accuracy = 75.33, Num_trees = 5, Depth = 1
test_accuracy = 74.00, Num_trees = 10, Depth = 1
test_accuracy = 74.00, Num_trees = 25, Depth = 1
test_accuracy = 73.33, Num_trees = 50, Depth = 1
test_accuracy = 74.00, Num_trees = 100, Depth = 1
test_accuracy = 74.00, Num_trees = 200, Depth = 1
test_accuracy = 70.67, Num_trees = 1, Depth = 2
test_accuracy = 76.67, Num_trees = 5, Depth = 2
test_accuracy = 73.33, Num_trees = 10, Depth = 2
test_accuracy = 76.67, Num_trees = 25, Depth = 2
test_accuracy = 74.67, Num_trees = 50, Depth = 2
test_accuracy = 74.67, Num_trees = 100, Depth = 2
test_accuracy = 74.67, Num_trees = 200, Depth = 2
test_accuracy = 78.00, Num_trees = 1, Depth = 3
test_accuracy = 82.00, Num_trees = 5, Depth = 3
test_accuracy = 83.33, Num_trees = 10, Depth = 3
test_accuracy = 83.33, Num_trees = 25, Depth = 3
test_accuracy = 82.67, Num_trees = 50, Depth = 3
test_accuracy = 82.67, Num_trees = 100, Depth = 3
test_accuracy = 83.33, Num_trees = 200, Depth = 3
test_accuracy = 88.00, Num_trees = 1, Depth = 4
test_accuracy = 86.00, Num_trees = 5, Depth = 4
test_accuracy = 88.00, Num_trees = 10, Depth = 4
test_accuracy = 88.67, Num_trees = 25, Depth = 4
test_accuracy = 88.00, Num_trees = 50, Depth = 4
test_accuracy = 88.00, Num_trees = 100, Depth = 4
test_accuracy = 88.00, Num_trees = 200, Depth = 4
test_accuracy = 84.00, Num_trees = 1, Depth = 5
test_accuracy = 89.33, Num_trees = 5, Depth = 5
test_accuracy = 86.67, Num_trees = 10, Depth = 5
test_accuracy = 84.00, Num_trees = 25, Depth = 5
test_accuracy = 85.33, Num_trees = 50, Depth = 5
test_accuracy = 84.67, Num_trees = 100, Depth = 5
test_accuracy = 85.33, Num_trees = 200, Depth = 5
test_accuracy = 84.67, Num_trees = 1, Depth = 6
test_accuracy = 88.67, Num_trees = 5, Depth = 6
test_accuracy = 86.00, Num_trees = 10, Depth = 6
test_accuracy = 86.00, Num_trees = 25, Depth = 6
test_accuracy = 85.33, Num_trees = 50, Depth = 6
test_accuracy = 84.67, Num_trees = 100, Depth = 6
test_accuracy = 85.33, Num_trees = 200, Depth = 6
test_accuracy = 82.67, Num_trees = 1, Depth = 7
test_accuracy = 83.33, Num_trees = 5, Depth = 7
test_accuracy = 85.33, Num_trees = 10, Depth = 7
test_accuracy = 84.00, Num_trees = 25, Depth = 7
test_accuracy = 84.00, Num_trees = 50, Depth = 7
test_accuracy = 84.67, Num_trees = 100, Depth = 7
test_accuracy = 84.00, Num_trees = 200, Depth = 7
test_accuracy = 82.67, Num_trees = 1, Depth = 8
test_accuracy = 85.33, Num_trees = 5, Depth = 8
test_accuracy = 84.67, Num_trees = 10, Depth = 8
test_accuracy = 86.67, Num_trees = 25, Depth = 8
test_accuracy = 85.33, Num_trees = 50, Depth = 8
test_accuracy = 84.67, Num_trees = 100, Depth = 8
test_accuracy = 84.67, Num_trees = 200, Depth = 8
test_accuracy = 84.67, Num_trees = 1, Depth = 9
test_accuracy = 83.33, Num_trees = 5, Depth = 9
test_accuracy = 85.33, Num_trees = 10, Depth = 9
test_accuracy = 85.33, Num_trees = 25, Depth = 9
test_accuracy = 84.67, Num_trees = 50, Depth = 9
test_accuracy = 84.67, Num_trees = 100, Depth = 9
test_accuracy = 84.67, Num_trees = 200, Depth = 9
```



Training accuracy Vs Depth
Training Accuracy
Depth
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
1
2
3
4
5
6
7
8
9



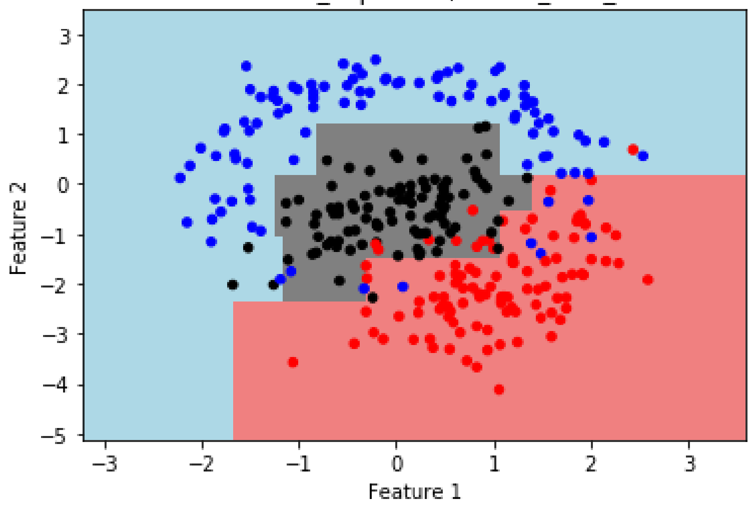
'Testing accuracy Vs Depth
Tesing Accuracy
Depth
90
88
86
84
82
80
78
76
1
2
3
4
5
6
7
8
9

```
Based on maximum test accuracy: depth = 5 and best_num_tree = 5 with
training_accuracy = 93.33 & testing_accuracy = 89.33
```

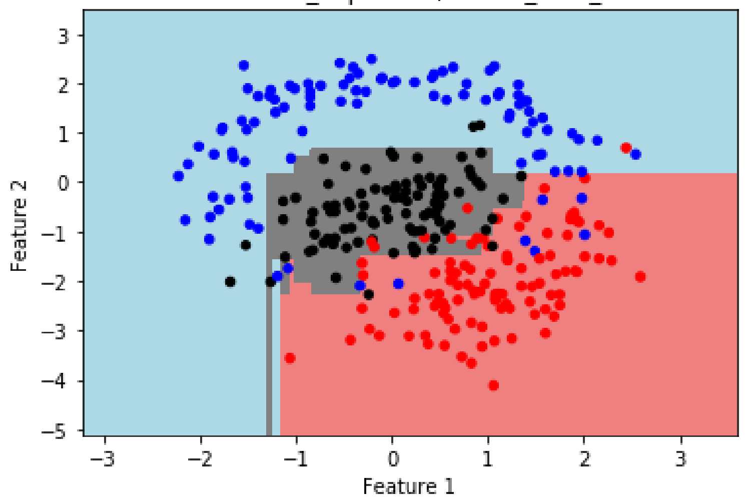
4.5 E.

[247]:
```
numtreerange=[1,5,10,25,50,100,200]
learnraterange=np.logspace(-3,0,15,base=10)

def expand_grid(dictionary):
   return pd.DataFrame([row for row in product(*dictionary.values())],
                       columns=dictionary.keys())

dictionary = {'numtreerange': numtreerange,
              'learnraterange': learnraterange}

prem = expand_grid(dictionary)

for i in [1,2,3,4,10]:
    best_fit = None
    ts_acc1 = 0
    best_ts_acc = 0
    best_numtree = None
    best_rate = None
    for j in range(prem.shape[0]):
        gb=GradientBoostingClassifier(learning_rate=prem.iloc[j,1],
                                      n_estimators=prem.iloc[j,0], max_depth=i)
        model_gb = gb.fit(x_train, y_train)
        ts_acc1 = model_gb.score(x_test, y_test)*100
#         print('test_accuracy = %.2f, Num_trees = %d, Learning_rate = %.3f,␣
↪Depth = %d'
#               %(ts_acc1, prem.iloc[j,0], prem.iloc[j,1],i))
        if (ts_acc1 > best_ts_acc):
            best_ts_acc = ts_acc1
            best_numtree = prem.iloc[j,0]
            best_rate = prem.iloc[j,1]
            best_fit = model_gb
    Z = best_fit.predict(np.c_[x1mesh.ravel(), x2mesh.ravel()])

    plot_func(x_train, y_train, Z, title = 'Scatterplot HW3Train for RF with␣
↪max_depth = %d,\
    best_num_tree = %d & best_rate = %.3f' %(i, best_numtree, best_rate))
```

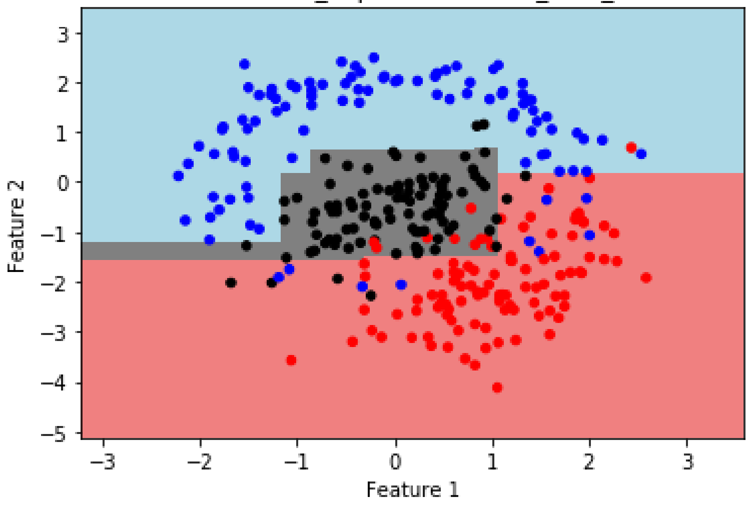
Scatterplot HW3Train for RF with max_depth = 1,   best_num_tree = 5 & best_rate = 1.000



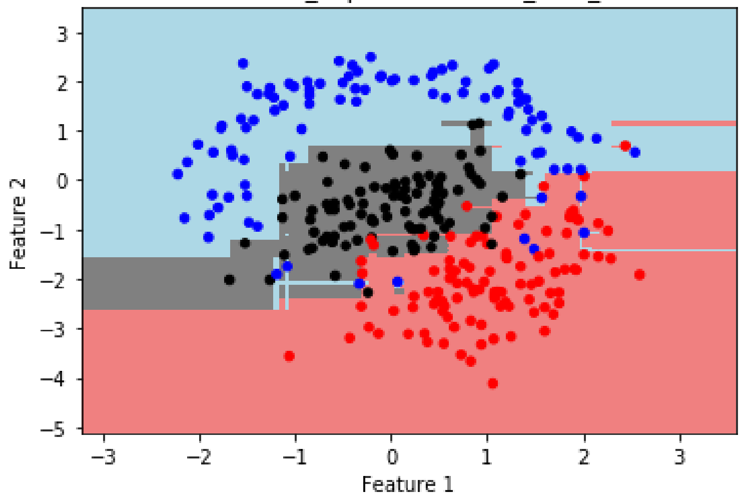
Scatterplot HW3Train for RF with max_depth = 2,   best_num_tree = 50 & best_rate = 0.052

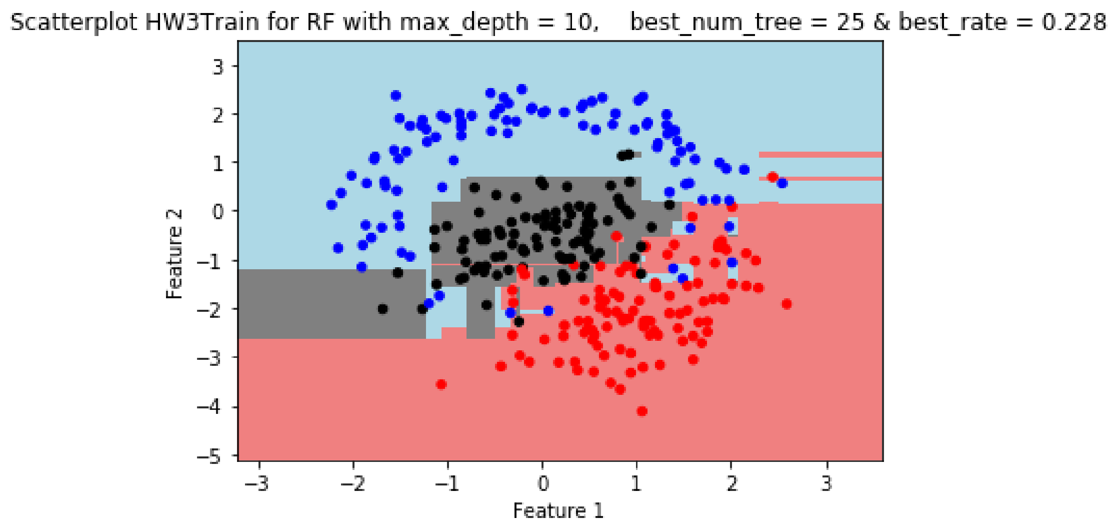


Scatterplot HW3Train for RF with max_depth = 3,   best_num_tree = 1 & best_rate = 0.085



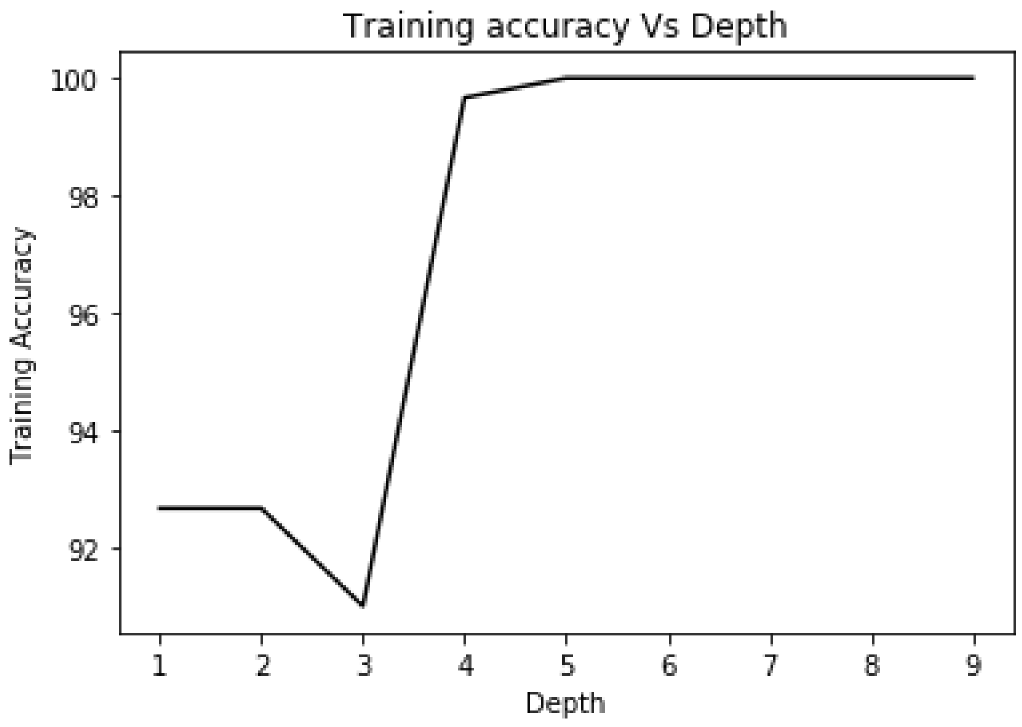
Scatterplot HW3Train for RF with max_depth = 4,   best_num_tree = 25 & best_rate = 0.139

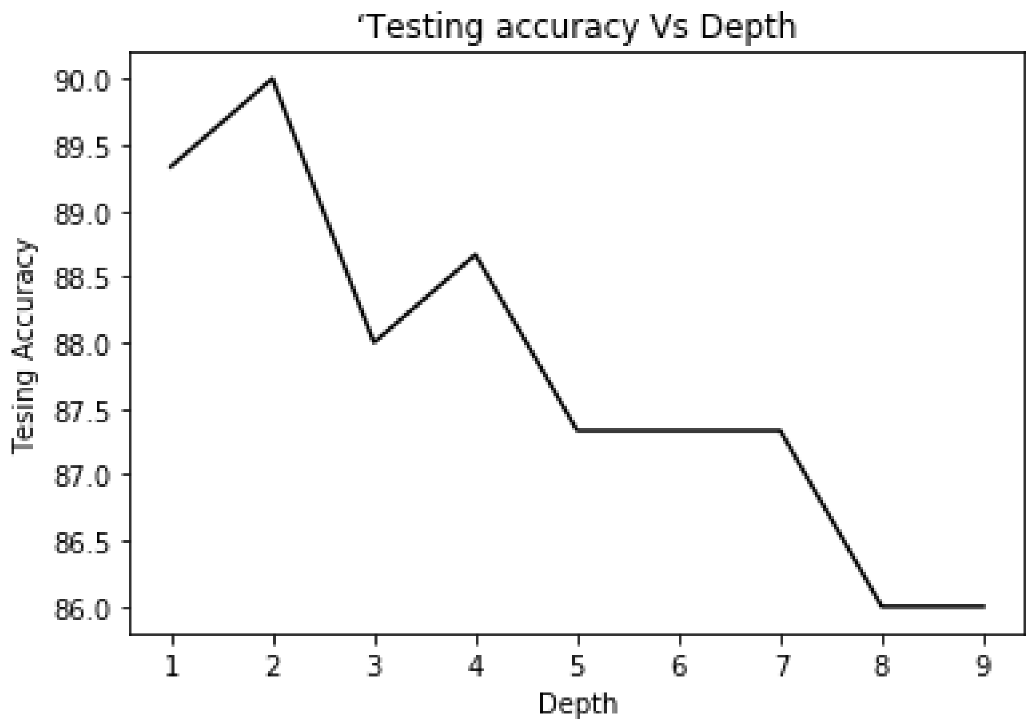




4.6 F.

[248]:
```
tr_acc_gb = []
ts_acc_gb = []
best_num_tree_gb = []
best_lr_rate_gb = []
k = range(1,10)

for i in k:
    best_fit = None
    ts_acc1 = 0
    best_ts_acc = 0
    best_numtree = None
    best_rate = None
    for j in range(prem.shape[0]):
        gb=GradientBoostingClassifier(learning_rate=prem.iloc[j,1],
                                      n_estimators=prem.iloc[j,0], max_depth=i)
        model_gb = gb.fit(x_train, y_train)
        ts_acc1 = model_gb.score(x_test, y_test)*100
#          print('test_accuracy = %.2f, Num_trees = %d, Learning_rate = %.3f,␣
↪Depth = %d'
#                %(ts_acc1, prem.iloc[j,0], prem.iloc[j,1],i))
        if (ts_acc1 > best_ts_acc):
            best_ts_acc = ts_acc1
            best_numtree = prem.iloc[j,0]
            best_rate = prem.iloc[j,1]
            best_fit = model_gb
    tr_acc_gb.append(best_fit.score(x_train, y_train)*100)
    ts_acc_gb.append(best_fit.score(x_test, y_test)*100)
    best_num_tree_gb.append(best_numtree)
    best_lr_rate_gb.append(best_rate)


plt.plot(k, tr_acc_gb, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(k, ts_acc_gb, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Tesing Accuracy")
plt.title("‘Testing accuracy Vs Depth")
plt.show()

kkk = np.argmax(ts_acc_gb)
print('Based on maximum test accuracy: depth = %d, best_num_tree = %d & \
learning_rate = %.4f with training_accuracy = %.2f & testing_accuracy = %.2f'
      %(kkk+1, best_num_tree_gb[kkk], best_lr_rate_gb[kkk],
        tr_acc_gb[kkk], ts_acc_gb[kkk]))
```

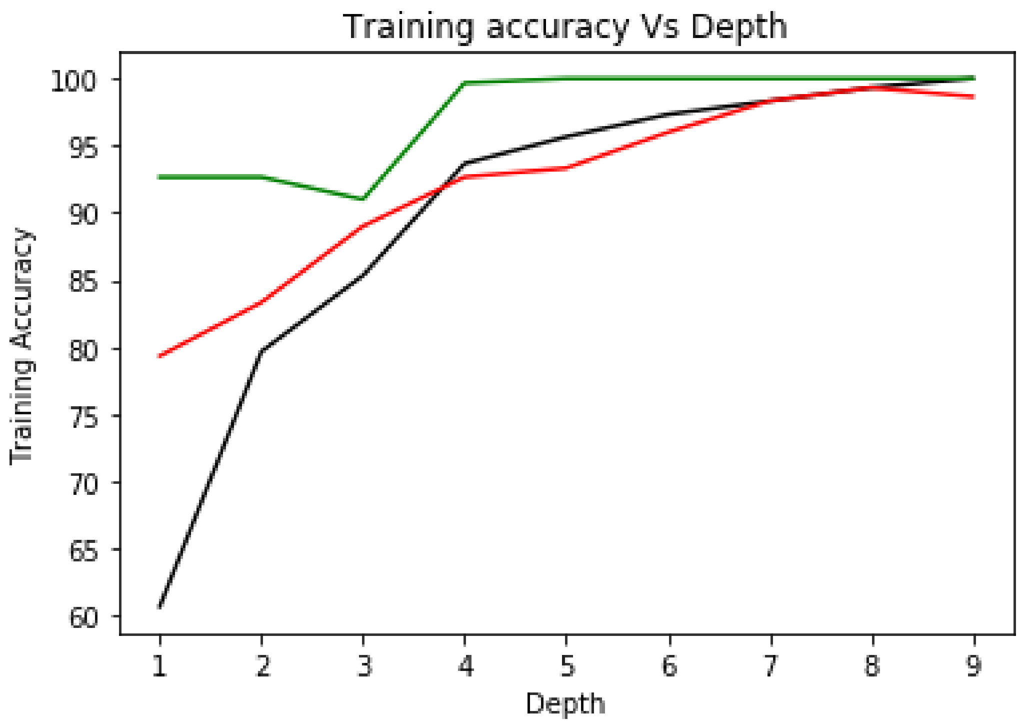


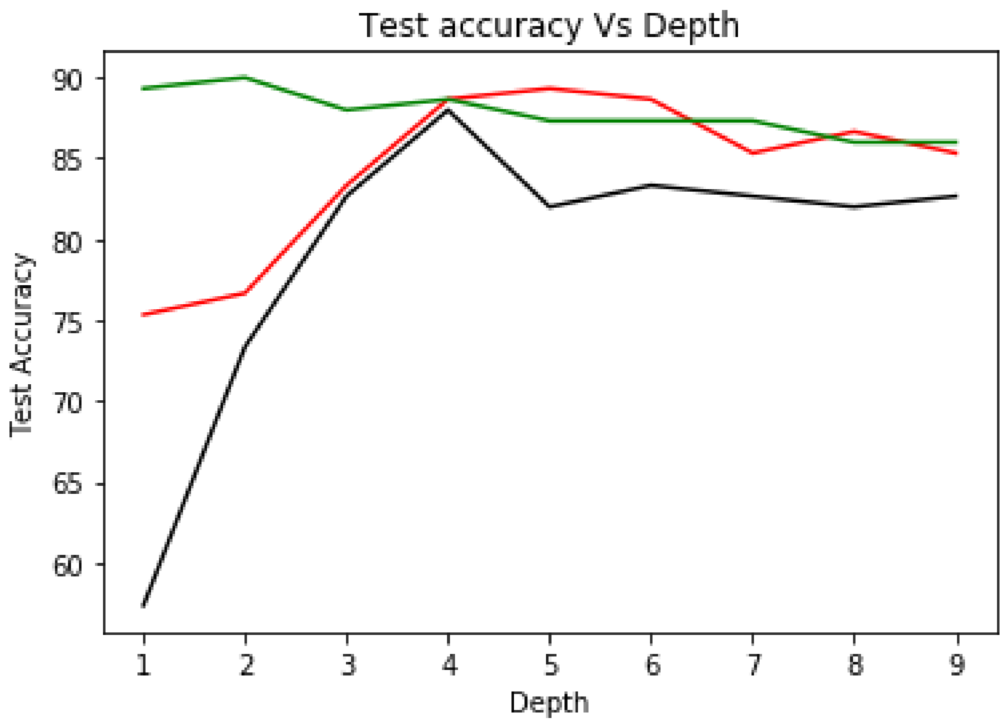


```
Based on maximum test accuracy: depth = 2, best_num_tree = 50 & learning_rate =
0.0518 with training_accuracy = 92.67 & testing_accuracy = 90.00
```

4.7 G.

[249]:
```
plt.plot(depth_dt, tr_acc_dt, color = 'black')
k = range(1,10)
plt.plot(k, tr_acc_rf, color = 'r')
plt.plot(k, tr_acc_gb, color = 'g')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(depth_dt, ts_acc_dt, color = 'black')
k = range(1,10)
plt.plot(k, ts_acc_rf, color = 'r')
plt.plot(k, ts_acc_gb, color = 'g')
plt.xlabel("Depth")
plt.ylabel("Test Accuracy")
plt.title("Test accuracy Vs Depth")
plt.show()
```


Training accuracy Vs Depth
Training Accuracy
Depth
100
95
90
85
80
75
70
65
60
1
2
3
4
5
6
7
8
9


Test accuracy Vs Depth
Test Accuracy
Depth
90
85
80
75
70
65
60
1
2
3
4
5
6
7
8
9

Single Trees: As depth of a tree increases, decision boundaries become more complex but linearly (vertical line with response) separable boundaries that is also same for bagged and boosted trees. However, bagged and boosted trees generate more stable (higher accuracy) performance (training and testing) than single trees. Clearly, there is a relation on the depth with the performance among these trees. Training accuracy increases if we increase depth of the trees and reaches to 100% after a fixed depth for each method which is not true for the test accuracy. Every tree (single, bagged and boosted) has an optimal depth at which it shows highest test accuracy and after that even if we increase depth, test accuracy started to decline or at least less than the optimal depth. Depth for boosted tree (GB) with highest accuracy is lower than single and bagged trees.

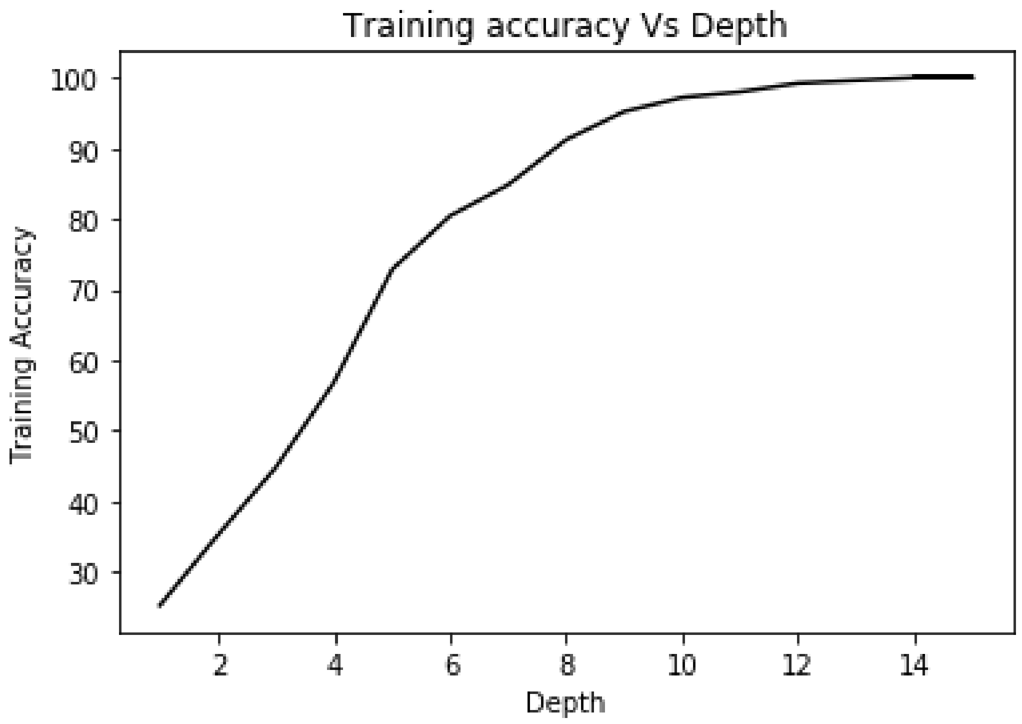
KNN, SVM, QDA can produce complex but linear and non-linear separable boundaries that is not possible by these trees (linearly (vertical line with response) separable boundaries). LDA also produce linear separable boundaries but it can produce linear with different slope boundaries. As a result, it is difficult to say which method will perform better a data set. It is completely depending on the data characteristics. However, for this data SVM model produced best test accuracy (92%).

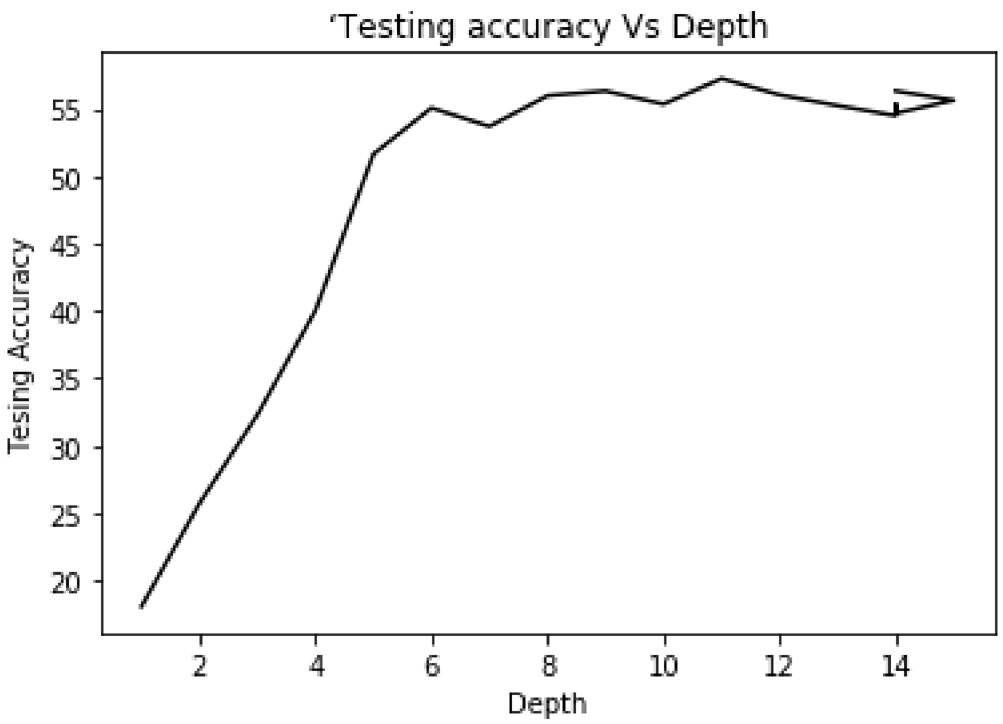
5 Problem 5

5.1 A.

[250]:
```python
from datetime import datetime


path = 'D:/ISU/COMS 574 - Introduction to Machine Learning/HW/HW4/'

df_train = pd.read_csv(path + 'digits-train.csv', sep=',',
                       header=None)
df_test = pd.read_csv(path + 'digits-test.csv', sep=',',
                      header=None)
tr_size = df_train.shape
ts_size = df_test.shape

x_train = np.array(df_train.loc[:,1:tr_size[1]])
y_train = np.array(df_train.loc[:,0])

x_test = np.array(df_test.loc[:,1:tr_size[1]])
y_test = np.array(df_test.loc[:,0])
```

[251]:
```python
tr_acc_dt = []
ts_acc_dt = []
depth_dt = []
k = range(1,20)

t0 = datetime.now()
for i in k:
    tree=DecisionTreeClassifier(criterion='gini', splitter='best', max_depth= i)
    model_tree = tree.fit(x_train, y_train)
    tr_acc_dt.append(model_tree.score(x_train, y_train)*100)
    ts_acc_dt.append(model_tree.score(x_test, y_test)*100)
    depth_dt.append(model_tree.get_depth())

t1 = datetime.now()

plt.plot(depth_dt, tr_acc_dt, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(depth_dt, ts_acc_dt, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Tesing Accuracy")
plt.title("‘Testing accuracy Vs Depth")
plt.show()

print('Time: '+str(t1-t0))
print('Based on maximum testing accuracy: depth = %d with training_accuracy = %.
 ↪2f and testing_accuracy = %.2f'
      %(np.argmax(ts_acc_dt)+1, tr_acc_dt[np.argmax(ts_acc_dt)], ts_acc_dt[np.
 ↪argmax(ts_acc_dt)]))
```



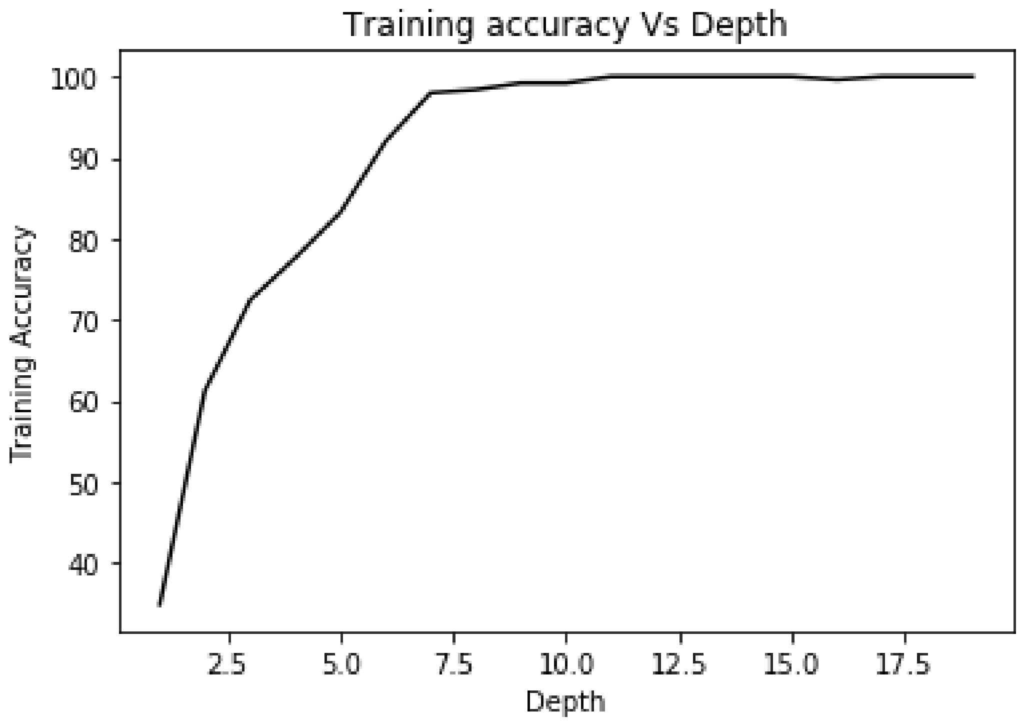


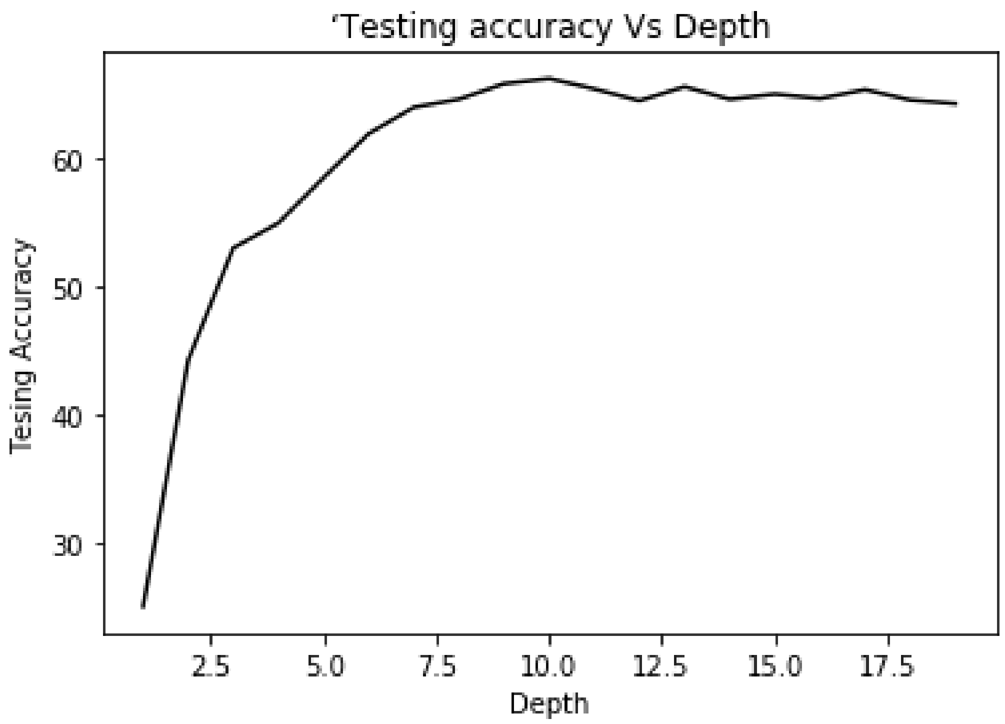
```
Time: 0:00:00.264866
Based on maximum testing accuracy: depth = 11 with training_accuracy = 98.00 and
testing_accuracy = 57.26
```

5.2 B.

[252]:
```
numtreerange=[1,5,10,25,50,100,200]
tr_acc_rf = []
ts_acc_rf = []
best_num_tree_rf = []
k = range(1,20)

t0 = datetime.now()

for i in k:
    best_fit = None
    ts_acc1 = 0
    best_ts_acc = 0
    best_numtree = 0
    for j in numtreerange:
        rf=RandomForestClassifier(bootstrap=True, n_estimators=j,
                                  max_features=None, criterion='gini', max_depth=i)
        model_rf = rf.fit(x_train, y_train)
        ts_acc1 = model_rf.score(x_test, y_test)*100
        print('test_accuracy = %.2f, Num_trees = %d, Depth = %d' %(ts_acc1, j,␣
 ↪i))
        if (ts_acc1 > best_ts_acc):
            best_ts_acc = ts_acc1
            best_numtree = j
            best_fit = model_rf
    tr_acc_rf.append(best_fit.score(x_train, y_train)*100)
    ts_acc_rf.append(best_fit.score(x_test, y_test)*100)
    best_num_tree_rf.append(best_numtree)

t1 = datetime.now()


plt.plot(k, tr_acc_rf, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(k, ts_acc_rf, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Tesing Accuracy")
plt.title("‘Testing accuracy Vs Depth")
plt.show()

print('Time: '+str(t1-t0))
print('Based on maximum test accuracy: depth = %d and best_num_tree = %d with␣
 ↪training_accuracy = %.2f & testing_accuracy = %.2f'
      %(k[np.argmax(ts_acc_rf)], best_num_tree_rf[np.argmax(ts_acc_rf)],
        tr_acc_rf[np.argmax(ts_acc_rf)], ts_acc_rf[np.argmax(ts_acc_rf)]))
```

```
test_accuracy = 18.00, Num_trees = 1, Depth = 1
test_accuracy = 18.00, Num_trees = 5, Depth = 1
test_accuracy = 18.00, Num_trees = 10, Depth = 1
test_accuracy = 25.03, Num_trees = 25, Depth = 1
test_accuracy = 18.00, Num_trees = 50, Depth = 1
test_accuracy = 18.00, Num_trees = 100, Depth = 1
test_accuracy = 23.66, Num_trees = 200, Depth = 1
test_accuracy = 24.34, Num_trees = 1, Depth = 2
test_accuracy = 38.63, Num_trees = 5, Depth = 2
test_accuracy = 39.26, Num_trees = 10, Depth = 2
test_accuracy = 37.26, Num_trees = 25, Depth = 2
test_accuracy = 44.23, Num_trees = 50, Depth = 2
test_accuracy = 42.63, Num_trees = 100, Depth = 2
test_accuracy = 44.00, Num_trees = 200, Depth = 2
test_accuracy = 26.74, Num_trees = 1, Depth = 3
test_accuracy = 39.89, Num_trees = 5, Depth = 3
test_accuracy = 52.97, Num_trees = 10, Depth = 3
test_accuracy = 52.51, Num_trees = 25, Depth = 3
test_accuracy = 51.09, Num_trees = 50, Depth = 3
test_accuracy = 50.57, Num_trees = 100, Depth = 3
test_accuracy = 49.60, Num_trees = 200, Depth = 3
test_accuracy = 46.69, Num_trees = 1, Depth = 4
test_accuracy = 50.74, Num_trees = 5, Depth = 4
test_accuracy = 50.17, Num_trees = 10, Depth = 4
test_accuracy = 52.69, Num_trees = 25, Depth = 4
test_accuracy = 53.54, Num_trees = 50, Depth = 4
test_accuracy = 52.97, Num_trees = 100, Depth = 4
test_accuracy = 54.91, Num_trees = 200, Depth = 4
test_accuracy = 49.66, Num_trees = 1, Depth = 5
test_accuracy = 55.71, Num_trees = 5, Depth = 5
test_accuracy = 58.29, Num_trees = 10, Depth = 5
test_accuracy = 54.97, Num_trees = 25, Depth = 5
test_accuracy = 57.94, Num_trees = 50, Depth = 5
test_accuracy = 57.77, Num_trees = 100, Depth = 5
test_accuracy = 58.46, Num_trees = 200, Depth = 5
test_accuracy = 52.51, Num_trees = 1, Depth = 6
test_accuracy = 58.06, Num_trees = 5, Depth = 6
test_accuracy = 59.54, Num_trees = 10, Depth = 6
test_accuracy = 61.43, Num_trees = 25, Depth = 6
test_accuracy = 61.89, Num_trees = 50, Depth = 6
test_accuracy = 60.23, Num_trees = 100, Depth = 6
test_accuracy = 61.60, Num_trees = 200, Depth = 6
test_accuracy = 51.14, Num_trees = 1, Depth = 7
test_accuracy = 61.89, Num_trees = 5, Depth = 7
test_accuracy = 60.46, Num_trees = 10, Depth = 7
test_accuracy = 63.66, Num_trees = 25, Depth = 7
test_accuracy = 62.11, Num_trees = 50, Depth = 7
test_accuracy = 63.94, Num_trees = 100, Depth = 7
test_accuracy = 62.34, Num_trees = 200, Depth = 7
test_accuracy = 52.80, Num_trees = 1, Depth = 8
test_accuracy = 58.29, Num_trees = 5, Depth = 8
test_accuracy = 64.57, Num_trees = 10, Depth = 8
test_accuracy = 63.20, Num_trees = 25, Depth = 8
test_accuracy = 64.00, Num_trees = 50, Depth = 8
test_accuracy = 64.57, Num_trees = 100, Depth = 8
test_accuracy = 64.06, Num_trees = 200, Depth = 8
test_accuracy = 55.60, Num_trees = 1, Depth = 9
test_accuracy = 61.60, Num_trees = 5, Depth = 9
test_accuracy = 62.46, Num_trees = 10, Depth = 9
test_accuracy = 65.77, Num_trees = 25, Depth = 9
test_accuracy = 64.63, Num_trees = 50, Depth = 9
test_accuracy = 64.51, Num_trees = 100, Depth = 9
test_accuracy = 63.83, Num_trees = 200, Depth = 9
test_accuracy = 49.37, Num_trees = 1, Depth = 10
test_accuracy = 62.57, Num_trees = 5, Depth = 10
test_accuracy = 61.71, Num_trees = 10, Depth = 10
test_accuracy = 64.74, Num_trees = 25, Depth = 10
test_accuracy = 66.17, Num_trees = 50, Depth = 10
test_accuracy = 64.57, Num_trees = 100, Depth = 10
test_accuracy = 64.51, Num_trees = 200, Depth = 10
test_accuracy = 55.54, Num_trees = 1, Depth = 11
test_accuracy = 60.11, Num_trees = 5, Depth = 11
test_accuracy = 61.14, Num_trees = 10, Depth = 11
test_accuracy = 64.51, Num_trees = 25, Depth = 11
test_accuracy = 63.94, Num_trees = 50, Depth = 11
test_accuracy = 65.37, Num_trees = 100, Depth = 11
test_accuracy = 63.83, Num_trees = 200, Depth = 11
test_accuracy = 52.51, Num_trees = 1, Depth = 12
test_accuracy = 60.40, Num_trees = 5, Depth = 12
test_accuracy = 60.91, Num_trees = 10, Depth = 12
test_accuracy = 64.40, Num_trees = 25, Depth = 12
test_accuracy = 62.80, Num_trees = 50, Depth = 12
test_accuracy = 64.46, Num_trees = 100, Depth = 12
test_accuracy = 64.11, Num_trees = 200, Depth = 12
test_accuracy = 51.94, Num_trees = 1, Depth = 13
test_accuracy = 62.97, Num_trees = 5, Depth = 13
test_accuracy = 61.43, Num_trees = 10, Depth = 13
test_accuracy = 64.86, Num_trees = 25, Depth = 13
test_accuracy = 65.54, Num_trees = 50, Depth = 13
test_accuracy = 63.89, Num_trees = 100, Depth = 13
test_accuracy = 64.00, Num_trees = 200, Depth = 13
test_accuracy = 54.06, Num_trees = 1, Depth = 14
test_accuracy = 57.83, Num_trees = 5, Depth = 14
test_accuracy = 61.94, Num_trees = 10, Depth = 14
test_accuracy = 63.94, Num_trees = 25, Depth = 14
test_accuracy = 64.06, Num_trees = 50, Depth = 14
test_accuracy = 64.57, Num_trees = 100, Depth = 14
test_accuracy = 64.17, Num_trees = 200, Depth = 14
test_accuracy = 49.37, Num_trees = 1, Depth = 15
test_accuracy = 59.37, Num_trees = 5, Depth = 15
test_accuracy = 63.09, Num_trees = 10, Depth = 15
test_accuracy = 61.71, Num_trees = 25, Depth = 15
test_accuracy = 63.54, Num_trees = 50, Depth = 15
test_accuracy = 64.46, Num_trees = 100, Depth = 15
test_accuracy = 64.97, Num_trees = 200, Depth = 15
test_accuracy = 53.03, Num_trees = 1, Depth = 16
test_accuracy = 61.14, Num_trees = 5, Depth = 16
test_accuracy = 61.43, Num_trees = 10, Depth = 16
test_accuracy = 64.63, Num_trees = 25, Depth = 16
test_accuracy = 63.66, Num_trees = 50, Depth = 16
test_accuracy = 63.71, Num_trees = 100, Depth = 16
test_accuracy = 64.51, Num_trees = 200, Depth = 16
test_accuracy = 50.29, Num_trees = 1, Depth = 17
test_accuracy = 63.03, Num_trees = 5, Depth = 17
test_accuracy = 64.91, Num_trees = 10, Depth = 17
test_accuracy = 64.34, Num_trees = 25, Depth = 17
test_accuracy = 65.20, Num_trees = 50, Depth = 17
test_accuracy = 65.09, Num_trees = 100, Depth = 17
test_accuracy = 65.31, Num_trees = 200, Depth = 17
test_accuracy = 53.37, Num_trees = 1, Depth = 18
test_accuracy = 60.17, Num_trees = 5, Depth = 18
test_accuracy = 61.94, Num_trees = 10, Depth = 18
test_accuracy = 63.03, Num_trees = 25, Depth = 18
test_accuracy = 64.23, Num_trees = 50, Depth = 18
test_accuracy = 64.34, Num_trees = 100, Depth = 18
test_accuracy = 64.51, Num_trees = 200, Depth = 18
test_accuracy = 55.77, Num_trees = 1, Depth = 19
test_accuracy = 58.69, Num_trees = 5, Depth = 19
test_accuracy = 62.29, Num_trees = 10, Depth = 19
test_accuracy = 63.60, Num_trees = 25, Depth = 19
test_accuracy = 63.83, Num_trees = 50, Depth = 19
test_accuracy = 63.54, Num_trees = 100, Depth = 19
test_accuracy = 64.23, Num_trees = 200, Depth = 19
```



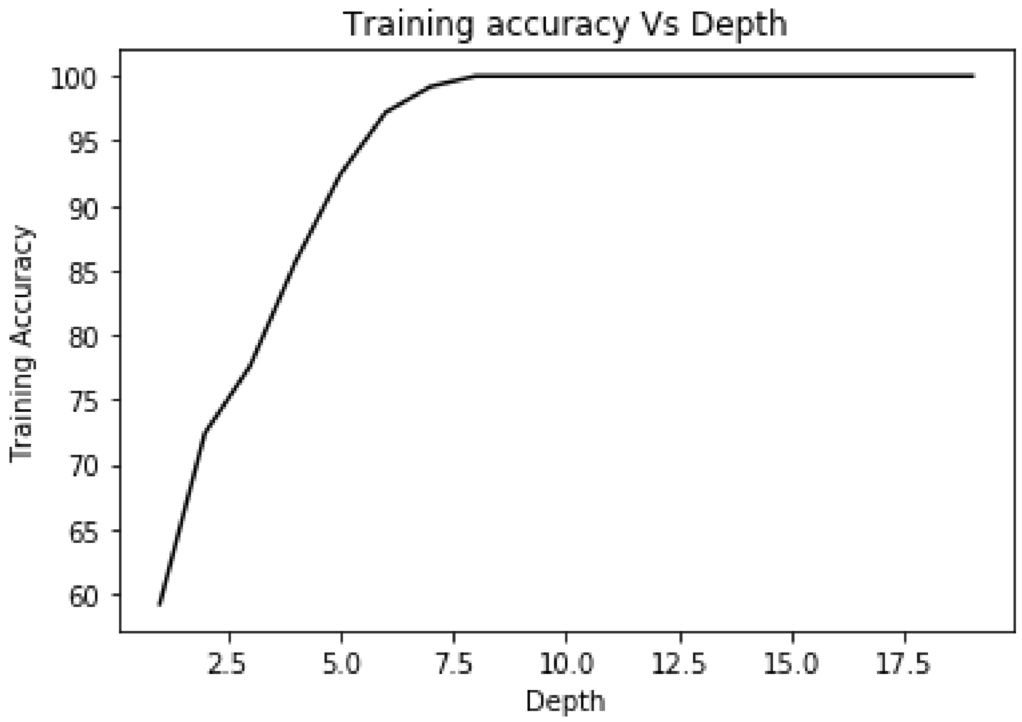


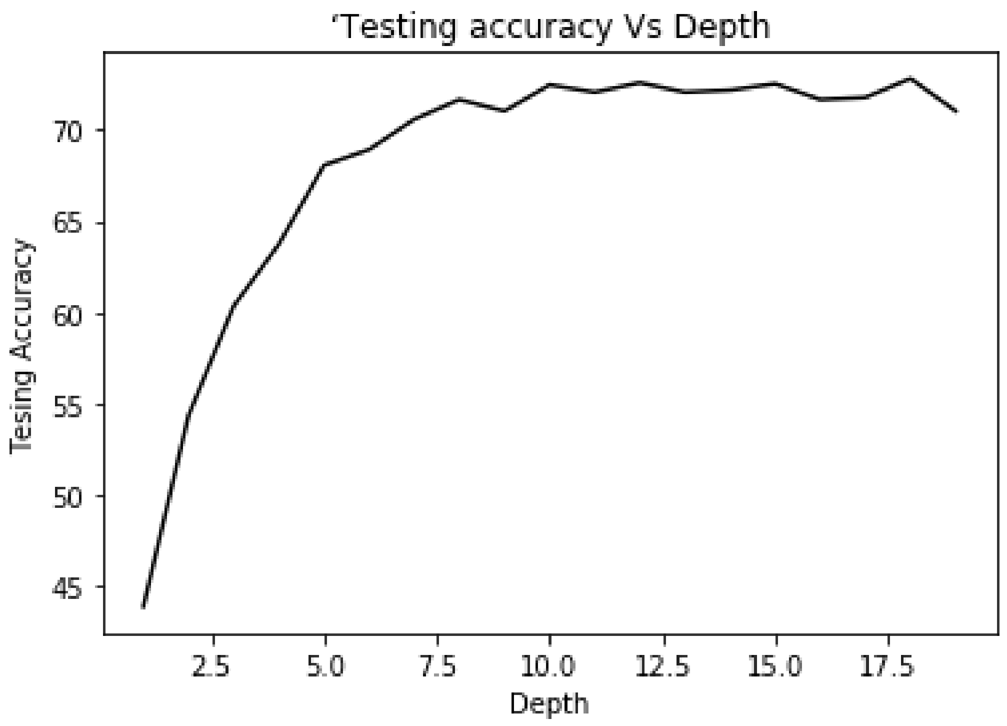
```
Time: 0:00:57.080398
Based on maximum test accuracy: depth = 10 and best_num_tree = 50 with
training_accuracy = 99.20 & testing_accuracy = 66.17
```

5.3 C.

[253]:
```
numtreerange=[1,5,10,25,50,100,200]
tr_acc_rfs = []
ts_acc_rfs = []
best_num_tree_rfs = []
k = range(1,20)

t0 = datetime.now()

for i in k:
    best_fit = None
    ts_acc1 = 0
    best_ts_acc = 0
    best_numtree = 0
    for j in numtreerange:
        rf=RandomForestClassifier(bootstrap=True, n_estimators=j,
                                  max_features='sqrt', criterion='gini',␣
 ↪max_depth=i)
        model_rf = rf.fit(x_train, y_train)
        ts_acc1 = model_rf.score(x_test, y_test)*100
        print('test_accuracy = %.2f, Num_trees = %d, Depth = %d' %(ts_acc1, j,␣
↪i))
        if (ts_acc1 > best_ts_acc):
            best_ts_acc = ts_acc1
            best_numtree = j
            best_fit = model_rf
    tr_acc_rfs.append(best_fit.score(x_train, y_train)*100)
    ts_acc_rfs.append(best_fit.score(x_test, y_test)*100)
    best_num_tree_rfs.append(best_numtree)

t1 = datetime.now()


plt.plot(k, tr_acc_rfs, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(k, ts_acc_rfs, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Tesing Accuracy")
plt.title("‘Testing accuracy Vs Depth")
plt.show()

print('Time: '+str(t1-t0))
print('Based on maximum test accuracy: depth = %d and best_num_tree = %d with \
training_accuracy = %.2f & testing_accuracy = %.2f'
      %(k[np.argmax(ts_acc_rfs)], best_num_tree_rfs[np.argmax(ts_acc_rfs)],
        tr_acc_rfs[np.argmax(ts_acc_rfs)], ts_acc_rfs[np.argmax(ts_acc_rfs)]))
```

```
test_accuracy = 14.97, Num_trees = 1, Depth = 1
test_accuracy = 24.17, Num_trees = 5, Depth = 1
test_accuracy = 35.37, Num_trees = 10, Depth = 1
test_accuracy = 43.03, Num_trees = 25, Depth = 1
test_accuracy = 40.06, Num_trees = 50, Depth = 1
test_accuracy = 40.97, Num_trees = 100, Depth = 1
test_accuracy = 43.89, Num_trees = 200, Depth = 1
test_accuracy = 23.77, Num_trees = 1, Depth = 2
test_accuracy = 38.63, Num_trees = 5, Depth = 2
test_accuracy = 50.23, Num_trees = 10, Depth = 2
test_accuracy = 46.91, Num_trees = 25, Depth = 2
test_accuracy = 52.57, Num_trees = 50, Depth = 2
test_accuracy = 50.34, Num_trees = 100, Depth = 2
test_accuracy = 54.34, Num_trees = 200, Depth = 2
test_accuracy = 39.14, Num_trees = 1, Depth = 3
test_accuracy = 45.66, Num_trees = 5, Depth = 3
test_accuracy = 50.51, Num_trees = 10, Depth = 3
test_accuracy = 56.63, Num_trees = 25, Depth = 3
test_accuracy = 56.86, Num_trees = 50, Depth = 3
test_accuracy = 59.31, Num_trees = 100, Depth = 3
test_accuracy = 60.34, Num_trees = 200, Depth = 3
test_accuracy = 41.71, Num_trees = 1, Depth = 4
test_accuracy = 53.03, Num_trees = 5, Depth = 4
test_accuracy = 59.77, Num_trees = 10, Depth = 4
test_accuracy = 62.06, Num_trees = 25, Depth = 4
test_accuracy = 63.26, Num_trees = 50, Depth = 4
test_accuracy = 62.86, Num_trees = 100, Depth = 4
test_accuracy = 63.71, Num_trees = 200, Depth = 4
test_accuracy = 39.60, Num_trees = 1, Depth = 5
test_accuracy = 58.23, Num_trees = 5, Depth = 5
test_accuracy = 60.69, Num_trees = 10, Depth = 5
test_accuracy = 63.66, Num_trees = 25, Depth = 5
test_accuracy = 65.37, Num_trees = 50, Depth = 5
test_accuracy = 68.06, Num_trees = 100, Depth = 5
test_accuracy = 66.11, Num_trees = 200, Depth = 5
test_accuracy = 46.69, Num_trees = 1, Depth = 6
test_accuracy = 62.11, Num_trees = 5, Depth = 6
test_accuracy = 63.14, Num_trees = 10, Depth = 6
test_accuracy = 66.17, Num_trees = 25, Depth = 6
test_accuracy = 68.80, Num_trees = 50, Depth = 6
test_accuracy = 68.91, Num_trees = 100, Depth = 6
test_accuracy = 68.23, Num_trees = 200, Depth = 6
test_accuracy = 46.57, Num_trees = 1, Depth = 7
test_accuracy = 58.86, Num_trees = 5, Depth = 7
test_accuracy = 66.40, Num_trees = 10, Depth = 7
test_accuracy = 68.11, Num_trees = 25, Depth = 7
test_accuracy = 67.83, Num_trees = 50, Depth = 7
test_accuracy = 69.54, Num_trees = 100, Depth = 7
test_accuracy = 70.57, Num_trees = 200, Depth = 7
test_accuracy = 43.43, Num_trees = 1, Depth = 8
test_accuracy = 60.97, Num_trees = 5, Depth = 8
test_accuracy = 63.31, Num_trees = 10, Depth = 8
test_accuracy = 68.80, Num_trees = 25, Depth = 8
test_accuracy = 70.06, Num_trees = 50, Depth = 8
test_accuracy = 71.66, Num_trees = 100, Depth = 8
test_accuracy = 71.09, Num_trees = 200, Depth = 8
test_accuracy = 46.23, Num_trees = 1, Depth = 9
test_accuracy = 60.29, Num_trees = 5, Depth = 9
test_accuracy = 63.89, Num_trees = 10, Depth = 9
test_accuracy = 70.80, Num_trees = 25, Depth = 9
test_accuracy = 69.66, Num_trees = 50, Depth = 9
test_accuracy = 70.06, Num_trees = 100, Depth = 9
test_accuracy = 71.03, Num_trees = 200, Depth = 9
test_accuracy = 39.60, Num_trees = 1, Depth = 10
test_accuracy = 60.97, Num_trees = 5, Depth = 10
test_accuracy = 64.40, Num_trees = 10, Depth = 10
test_accuracy = 68.46, Num_trees = 25, Depth = 10
test_accuracy = 71.49, Num_trees = 50, Depth = 10
test_accuracy = 72.46, Num_trees = 100, Depth = 10
test_accuracy = 72.11, Num_trees = 200, Depth = 10
test_accuracy = 45.03, Num_trees = 1, Depth = 11
test_accuracy = 61.09, Num_trees = 5, Depth = 11
test_accuracy = 65.54, Num_trees = 10, Depth = 11
test_accuracy = 71.03, Num_trees = 25, Depth = 11
test_accuracy = 70.23, Num_trees = 50, Depth = 11
test_accuracy = 71.31, Num_trees = 100, Depth = 11
test_accuracy = 72.06, Num_trees = 200, Depth = 11
test_accuracy = 52.11, Num_trees = 1, Depth = 12
test_accuracy = 59.83, Num_trees = 5, Depth = 12
test_accuracy = 66.29, Num_trees = 10, Depth = 12
test_accuracy = 69.71, Num_trees = 25, Depth = 12
test_accuracy = 70.11, Num_trees = 50, Depth = 12
test_accuracy = 72.57, Num_trees = 100, Depth = 12
test_accuracy = 70.63, Num_trees = 200, Depth = 12
test_accuracy = 42.80, Num_trees = 1, Depth = 13
test_accuracy = 61.89, Num_trees = 5, Depth = 13
test_accuracy = 69.14, Num_trees = 10, Depth = 13
test_accuracy = 67.66, Num_trees = 25, Depth = 13
test_accuracy = 70.06, Num_trees = 50, Depth = 13
test_accuracy = 72.06, Num_trees = 100, Depth = 13
test_accuracy = 71.14, Num_trees = 200, Depth = 13
test_accuracy = 45.43, Num_trees = 1, Depth = 14
test_accuracy = 57.60, Num_trees = 5, Depth = 14
test_accuracy = 67.49, Num_trees = 10, Depth = 14
test_accuracy = 70.40, Num_trees = 25, Depth = 14
test_accuracy = 69.26, Num_trees = 50, Depth = 14
test_accuracy = 71.26, Num_trees = 100, Depth = 14
test_accuracy = 72.17, Num_trees = 200, Depth = 14
test_accuracy = 49.89, Num_trees = 1, Depth = 15
test_accuracy = 59.71, Num_trees = 5, Depth = 15
test_accuracy = 65.14, Num_trees = 10, Depth = 15
test_accuracy = 69.14, Num_trees = 25, Depth = 15
test_accuracy = 70.46, Num_trees = 50, Depth = 15
test_accuracy = 72.40, Num_trees = 100, Depth = 15
test_accuracy = 72.51, Num_trees = 200, Depth = 15
test_accuracy = 53.20, Num_trees = 1, Depth = 16
test_accuracy = 60.06, Num_trees = 5, Depth = 16
test_accuracy = 64.51, Num_trees = 10, Depth = 16
test_accuracy = 68.06, Num_trees = 25, Depth = 16
test_accuracy = 70.63, Num_trees = 50, Depth = 16
test_accuracy = 71.66, Num_trees = 100, Depth = 16
test_accuracy = 70.91, Num_trees = 200, Depth = 16
test_accuracy = 51.66, Num_trees = 1, Depth = 17
test_accuracy = 61.43, Num_trees = 5, Depth = 17
test_accuracy = 66.17, Num_trees = 10, Depth = 17
test_accuracy = 69.43, Num_trees = 25, Depth = 17
test_accuracy = 70.06, Num_trees = 50, Depth = 17
test_accuracy = 71.37, Num_trees = 100, Depth = 17
test_accuracy = 71.77, Num_trees = 200, Depth = 17
test_accuracy = 45.71, Num_trees = 1, Depth = 18
test_accuracy = 60.57, Num_trees = 5, Depth = 18
test_accuracy = 66.23, Num_trees = 10, Depth = 18
test_accuracy = 70.17, Num_trees = 25, Depth = 18
test_accuracy = 70.06, Num_trees = 50, Depth = 18
test_accuracy = 70.97, Num_trees = 100, Depth = 18
test_accuracy = 72.80, Num_trees = 200, Depth = 18
test_accuracy = 40.97, Num_trees = 1, Depth = 19
test_accuracy = 59.26, Num_trees = 5, Depth = 19
test_accuracy = 64.40, Num_trees = 10, Depth = 19
test_accuracy = 69.71, Num_trees = 25, Depth = 19
test_accuracy = 70.46, Num_trees = 50, Depth = 19
test_accuracy = 70.74, Num_trees = 100, Depth = 19
test_accuracy = 71.03, Num_trees = 200, Depth = 19
```





```
Time: 0:00:22.135032
Based on maximum test accuracy: depth = 18 and best_num_tree = 200 with
training_accuracy = 100.00 & testing_accuracy = 72.80
```

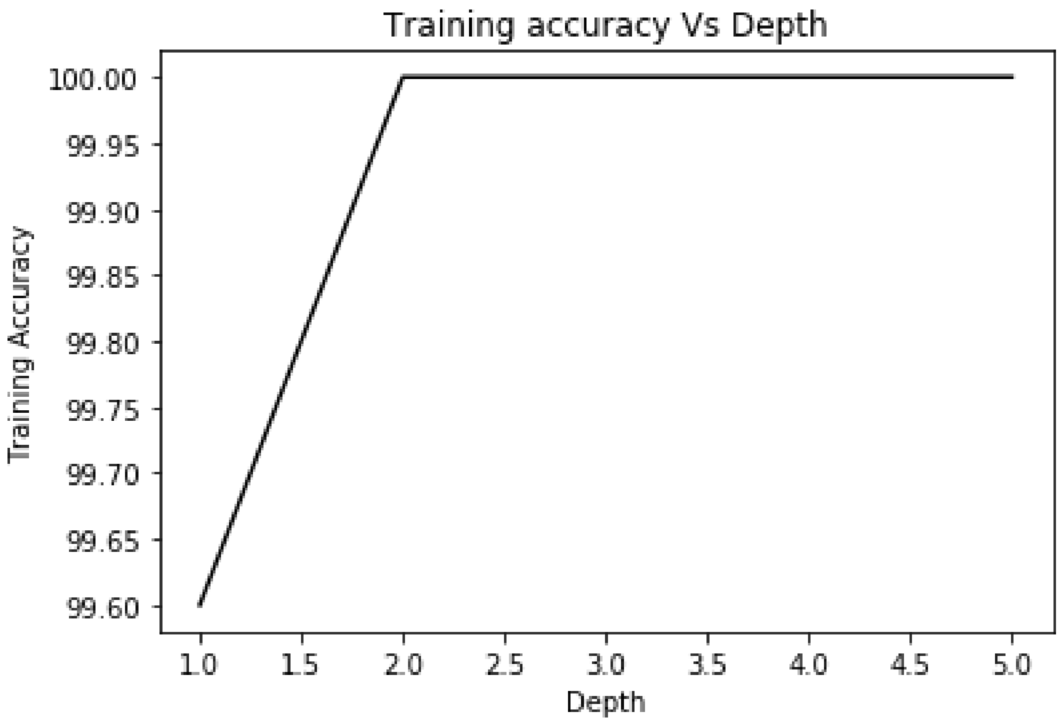
5.4 D.

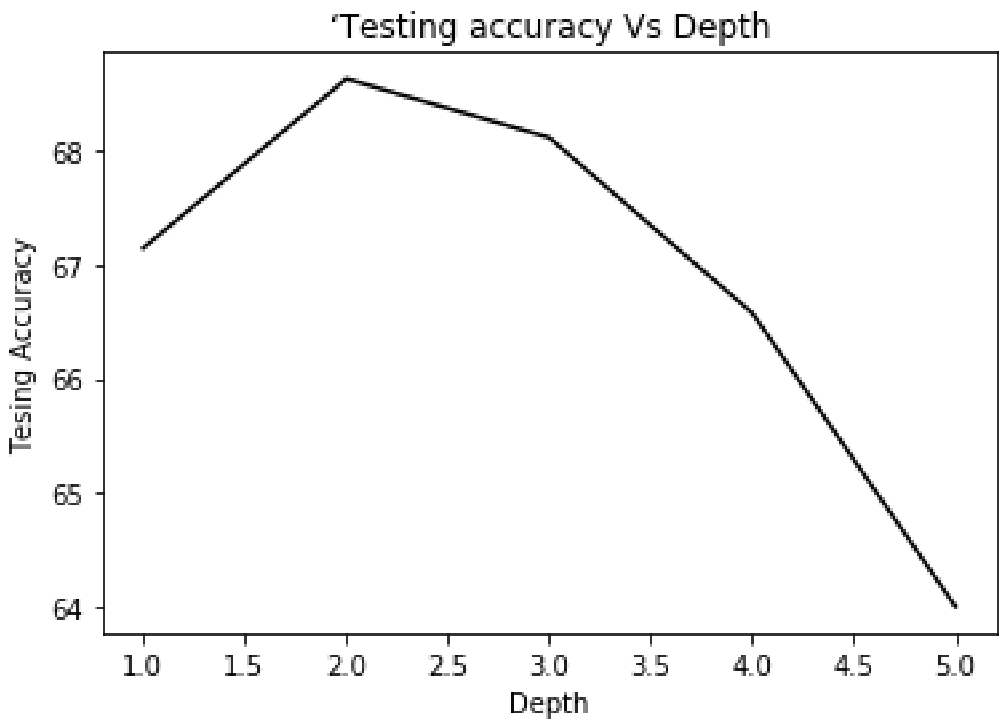
[254]:
```
numtreerange=[50,100,200]
learnraterange=np.logspace(-2,0,10,base=10)

def expand_grid(dictionary):
   return pd.DataFrame([row for row in product(*dictionary.values())],
                       columns=dictionary.keys())

dictionary = {'numtreerange': numtreerange,
              'learnraterange': learnraterange}

prem = expand_grid(dictionary)


tr_acc_gb = []
ts_acc_gb = []
best_num_tree_gb = []
best_lr_rate_gb = []
k = range(1,6)
```

```
t0 = datetime.now()

for i in k:
    best_fit = None
    ts_acc1 = 0
    best_ts_acc = 0
    best_numtree = None
    best_rate = None
    for j in range(prem.shape[0]):
        gb=GradientBoostingClassifier(learning_rate=prem.iloc[j,1],
                                      n_estimators=prem.iloc[j,0], max_depth=i)
        model_gb = gb.fit(x_train, y_train)
        ts_acc1 = model_gb.score(x_test, y_test)*100
        print('test_accuracy = %.2f, Num_trees = %d, Learning_rate = %.3f,␣
 ↪Depth = %d'
              %(ts_acc1, prem.iloc[j,0], prem.iloc[j,1],i))
        if (ts_acc1 > best_ts_acc):
            best_ts_acc = ts_acc1
            best_numtree = prem.iloc[j,0]
            best_rate = prem.iloc[j,1]
            best_fit = model_gb
    tr_acc_gb.append(best_fit.score(x_train, y_train)*100)
    ts_acc_gb.append(best_fit.score(x_test, y_test)*100)
    best_num_tree_gb.append(best_numtree)
    best_lr_rate_gb.append(best_rate)

t1 = datetime.now()


plt.plot(k, tr_acc_gb, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(k, ts_acc_gb, color = 'black')
plt.xlabel("Depth")
plt.ylabel("Tesing Accuracy")
plt.title("‘Testing accuracy Vs Depth")
plt.show()

print('Time: '+str(t1-t0))
print('Based on maximum test accuracy: depth = %d, best_num_tree = %d &␣
 ↪learning_rate = %.4f with \
training_accuracy = %.2f & testing_accuracy = %.2f'
```

```
    %(k[np.argmax(ts_acc_gb)], best_num_tree_gb[np.argmax(ts_acc_gb)],␣
↪best_lr_rate_gb[np.argmax(ts_acc_gb)],
        tr_acc_gb[np.argmax(ts_acc_gb)], ts_acc_gb[np.argmax(ts_acc_gb)]))
```

```
test_accuracy = 57.83, Num_trees = 50, Learning_rate = 0.010, Depth = 1
test_accuracy = 60.29, Num_trees = 50, Learning_rate = 0.017, Depth = 1
test_accuracy = 62.57, Num_trees = 50, Learning_rate = 0.028, Depth = 1
test_accuracy = 64.74, Num_trees = 50, Learning_rate = 0.046, Depth = 1
test_accuracy = 65.20, Num_trees = 50, Learning_rate = 0.077, Depth = 1
test_accuracy = 66.06, Num_trees = 50, Learning_rate = 0.129, Depth = 1
test_accuracy = 66.86, Num_trees = 50, Learning_rate = 0.215, Depth = 1
test_accuracy = 66.29, Num_trees = 50, Learning_rate = 0.359, Depth = 1
test_accuracy = 65.60, Num_trees = 50, Learning_rate = 0.599, Depth = 1
test_accuracy = 55.66, Num_trees = 50, Learning_rate = 1.000, Depth = 1
test_accuracy = 61.03, Num_trees = 100, Learning_rate = 0.010, Depth = 1
test_accuracy = 63.20, Num_trees = 100, Learning_rate = 0.017, Depth = 1
test_accuracy = 64.91, Num_trees = 100, Learning_rate = 0.028, Depth = 1
test_accuracy = 65.83, Num_trees = 100, Learning_rate = 0.046, Depth = 1
test_accuracy = 66.63, Num_trees = 100, Learning_rate = 0.077, Depth = 1
test_accuracy = 67.14, Num_trees = 100, Learning_rate = 0.129, Depth = 1
test_accuracy = 66.91, Num_trees = 100, Learning_rate = 0.215, Depth = 1
test_accuracy = 66.74, Num_trees = 100, Learning_rate = 0.359, Depth = 1
test_accuracy = 66.23, Num_trees = 100, Learning_rate = 0.599, Depth = 1
test_accuracy = 56.06, Num_trees = 100, Learning_rate = 1.000, Depth = 1
test_accuracy = 64.00, Num_trees = 200, Learning_rate = 0.010, Depth = 1
test_accuracy = 64.97, Num_trees = 200, Learning_rate = 0.017, Depth = 1
test_accuracy = 66.00, Num_trees = 200, Learning_rate = 0.028, Depth = 1
test_accuracy = 66.63, Num_trees = 200, Learning_rate = 0.046, Depth = 1
test_accuracy = 66.63, Num_trees = 200, Learning_rate = 0.077, Depth = 1
test_accuracy = 65.94, Num_trees = 200, Learning_rate = 0.129, Depth = 1
test_accuracy = 67.03, Num_trees = 200, Learning_rate = 0.215, Depth = 1
test_accuracy = 66.57, Num_trees = 200, Learning_rate = 0.359, Depth = 1
test_accuracy = 66.57, Num_trees = 200, Learning_rate = 0.599, Depth = 1
test_accuracy = 55.94, Num_trees = 200, Learning_rate = 1.000, Depth = 1
test_accuracy = 62.06, Num_trees = 50, Learning_rate = 0.010, Depth = 2
test_accuracy = 63.26, Num_trees = 50, Learning_rate = 0.017, Depth = 2
test_accuracy = 64.17, Num_trees = 50, Learning_rate = 0.028, Depth = 2
test_accuracy = 65.26, Num_trees = 50, Learning_rate = 0.046, Depth = 2
test_accuracy = 66.11, Num_trees = 50, Learning_rate = 0.077, Depth = 2
test_accuracy = 67.14, Num_trees = 50, Learning_rate = 0.129, Depth = 2
test_accuracy = 67.37, Num_trees = 50, Learning_rate = 0.215, Depth = 2
test_accuracy = 68.63, Num_trees = 50, Learning_rate = 0.359, Depth = 2
test_accuracy = 66.29, Num_trees = 50, Learning_rate = 0.599, Depth = 2
test_accuracy = 52.00, Num_trees = 50, Learning_rate = 1.000, Depth = 2
test_accuracy = 64.06, Num_trees = 100, Learning_rate = 0.010, Depth = 2
test_accuracy = 64.63, Num_trees = 100, Learning_rate = 0.017, Depth = 2
test_accuracy = 65.89, Num_trees = 100, Learning_rate = 0.028, Depth = 2
test_accuracy = 66.11, Num_trees = 100, Learning_rate = 0.046, Depth = 2
test_accuracy = 66.51, Num_trees = 100, Learning_rate = 0.077, Depth = 2
test_accuracy = 66.97, Num_trees = 100, Learning_rate = 0.129, Depth = 2
test_accuracy = 68.00, Num_trees = 100, Learning_rate = 0.215, Depth = 2
test_accuracy = 68.63, Num_trees = 100, Learning_rate = 0.359, Depth = 2
test_accuracy = 65.26, Num_trees = 100, Learning_rate = 0.599, Depth = 2
test_accuracy = 51.54, Num_trees = 100, Learning_rate = 1.000, Depth = 2
test_accuracy = 64.97, Num_trees = 200, Learning_rate = 0.010, Depth = 2
test_accuracy = 66.23, Num_trees = 200, Learning_rate = 0.017, Depth = 2
test_accuracy = 66.74, Num_trees = 200, Learning_rate = 0.028, Depth = 2
test_accuracy = 66.29, Num_trees = 200, Learning_rate = 0.046, Depth = 2
test_accuracy = 66.91, Num_trees = 200, Learning_rate = 0.077, Depth = 2
test_accuracy = 68.23, Num_trees = 200, Learning_rate = 0.129, Depth = 2
test_accuracy = 68.29, Num_trees = 200, Learning_rate = 0.215, Depth = 2
test_accuracy = 68.57, Num_trees = 200, Learning_rate = 0.359, Depth = 2
test_accuracy = 65.66, Num_trees = 200, Learning_rate = 0.599, Depth = 2
test_accuracy = 51.71, Num_trees = 200, Learning_rate = 1.000, Depth = 2
test_accuracy = 62.46, Num_trees = 50, Learning_rate = 0.010, Depth = 3
test_accuracy = 63.89, Num_trees = 50, Learning_rate = 0.017, Depth = 3
test_accuracy = 64.80, Num_trees = 50, Learning_rate = 0.028, Depth = 3
test_accuracy = 65.66, Num_trees = 50, Learning_rate = 0.046, Depth = 3
test_accuracy = 65.89, Num_trees = 50, Learning_rate = 0.077, Depth = 3
test_accuracy = 66.00, Num_trees = 50, Learning_rate = 0.129, Depth = 3
test_accuracy = 67.54, Num_trees = 50, Learning_rate = 0.215, Depth = 3
test_accuracy = 66.97, Num_trees = 50, Learning_rate = 0.359, Depth = 3
test_accuracy = 63.89, Num_trees = 50, Learning_rate = 0.599, Depth = 3
test_accuracy = 58.29, Num_trees = 50, Learning_rate = 1.000, Depth = 3
test_accuracy = 63.77, Num_trees = 100, Learning_rate = 0.010, Depth = 3
test_accuracy = 65.14, Num_trees = 100, Learning_rate = 0.017, Depth = 3
test_accuracy = 65.77, Num_trees = 100, Learning_rate = 0.028, Depth = 3
test_accuracy = 66.29, Num_trees = 100, Learning_rate = 0.046, Depth = 3
test_accuracy = 67.43, Num_trees = 100, Learning_rate = 0.077, Depth = 3
test_accuracy = 67.60, Num_trees = 100, Learning_rate = 0.129, Depth = 3
test_accuracy = 67.49, Num_trees = 100, Learning_rate = 0.215, Depth = 3
test_accuracy = 67.31, Num_trees = 100, Learning_rate = 0.359, Depth = 3
test_accuracy = 65.09, Num_trees = 100, Learning_rate = 0.599, Depth = 3
test_accuracy = 58.86, Num_trees = 100, Learning_rate = 1.000, Depth = 3
test_accuracy = 66.00, Num_trees = 200, Learning_rate = 0.010, Depth = 3
test_accuracy = 66.69, Num_trees = 200, Learning_rate = 0.017, Depth = 3
test_accuracy = 66.69, Num_trees = 200, Learning_rate = 0.028, Depth = 3
test_accuracy = 67.20, Num_trees = 200, Learning_rate = 0.046, Depth = 3
test_accuracy = 68.11, Num_trees = 200, Learning_rate = 0.077, Depth = 3
test_accuracy = 67.71, Num_trees = 200, Learning_rate = 0.129, Depth = 3
test_accuracy = 67.94, Num_trees = 200, Learning_rate = 0.215, Depth = 3
test_accuracy = 67.71, Num_trees = 200, Learning_rate = 0.359, Depth = 3
test_accuracy = 64.74, Num_trees = 200, Learning_rate = 0.599, Depth = 3
test_accuracy = 60.46, Num_trees = 200, Learning_rate = 1.000, Depth = 3
test_accuracy = 58.57, Num_trees = 50, Learning_rate = 0.010, Depth = 4
test_accuracy = 60.91, Num_trees = 50, Learning_rate = 0.017, Depth = 4
test_accuracy = 62.40, Num_trees = 50, Learning_rate = 0.028, Depth = 4
test_accuracy = 63.66, Num_trees = 50, Learning_rate = 0.046, Depth = 4
test_accuracy = 64.97, Num_trees = 50, Learning_rate = 0.077, Depth = 4
test_accuracy = 64.74, Num_trees = 50, Learning_rate = 0.129, Depth = 4
test_accuracy = 65.77, Num_trees = 50, Learning_rate = 0.215, Depth = 4
test_accuracy = 64.57, Num_trees = 50, Learning_rate = 0.359, Depth = 4
test_accuracy = 63.54, Num_trees = 50, Learning_rate = 0.599, Depth = 4
test_accuracy = 54.80, Num_trees = 50, Learning_rate = 1.000, Depth = 4
test_accuracy = 60.97, Num_trees = 100, Learning_rate = 0.010, Depth = 4
test_accuracy = 62.80, Num_trees = 100, Learning_rate = 0.017, Depth = 4
test_accuracy = 63.83, Num_trees = 100, Learning_rate = 0.028, Depth = 4
test_accuracy = 64.11, Num_trees = 100, Learning_rate = 0.046, Depth = 4
test_accuracy = 66.00, Num_trees = 100, Learning_rate = 0.077, Depth = 4
test_accuracy = 65.89, Num_trees = 100, Learning_rate = 0.129, Depth = 4
test_accuracy = 65.14, Num_trees = 100, Learning_rate = 0.215, Depth = 4
test_accuracy = 64.69, Num_trees = 100, Learning_rate = 0.359, Depth = 4
test_accuracy = 63.37, Num_trees = 100, Learning_rate = 0.599, Depth = 4
test_accuracy = 56.11, Num_trees = 100, Learning_rate = 1.000, Depth = 4
test_accuracy = 63.66, Num_trees = 200, Learning_rate = 0.010, Depth = 4
test_accuracy = 65.54, Num_trees = 200, Learning_rate = 0.017, Depth = 4
test_accuracy = 64.40, Num_trees = 200, Learning_rate = 0.028, Depth = 4
test_accuracy = 65.26, Num_trees = 200, Learning_rate = 0.046, Depth = 4
test_accuracy = 66.34, Num_trees = 200, Learning_rate = 0.077, Depth = 4
test_accuracy = 66.57, Num_trees = 200, Learning_rate = 0.129, Depth = 4
test_accuracy = 65.20, Num_trees = 200, Learning_rate = 0.215, Depth = 4
test_accuracy = 63.49, Num_trees = 200, Learning_rate = 0.359, Depth = 4
test_accuracy = 62.06, Num_trees = 200, Learning_rate = 0.599, Depth = 4
test_accuracy = 55.71, Num_trees = 200, Learning_rate = 1.000, Depth = 4
test_accuracy = 56.91, Num_trees = 50, Learning_rate = 0.010, Depth = 5
test_accuracy = 58.57, Num_trees = 50, Learning_rate = 0.017, Depth = 5
test_accuracy = 59.83, Num_trees = 50, Learning_rate = 0.028, Depth = 5
test_accuracy = 61.37, Num_trees = 50, Learning_rate = 0.046, Depth = 5
test_accuracy = 61.83, Num_trees = 50, Learning_rate = 0.077, Depth = 5
test_accuracy = 63.94, Num_trees = 50, Learning_rate = 0.129, Depth = 5
test_accuracy = 63.03, Num_trees = 50, Learning_rate = 0.215, Depth = 5
test_accuracy = 61.31, Num_trees = 50, Learning_rate = 0.359, Depth = 5
test_accuracy = 60.06, Num_trees = 50, Learning_rate = 0.599, Depth = 5
test_accuracy = 52.63, Num_trees = 50, Learning_rate = 1.000, Depth = 5
test_accuracy = 59.26, Num_trees = 100, Learning_rate = 0.010, Depth = 5
test_accuracy = 59.83, Num_trees = 100, Learning_rate = 0.017, Depth = 5
test_accuracy = 61.09, Num_trees = 100, Learning_rate = 0.028, Depth = 5
test_accuracy = 62.86, Num_trees = 100, Learning_rate = 0.046, Depth = 5
test_accuracy = 63.66, Num_trees = 100, Learning_rate = 0.077, Depth = 5
test_accuracy = 64.00, Num_trees = 100, Learning_rate = 0.129, Depth = 5
test_accuracy = 62.74, Num_trees = 100, Learning_rate = 0.215, Depth = 5
test_accuracy = 61.31, Num_trees = 100, Learning_rate = 0.359, Depth = 5
test_accuracy = 60.74, Num_trees = 100, Learning_rate = 0.599, Depth = 5
test_accuracy = 52.34, Num_trees = 100, Learning_rate = 1.000, Depth = 5
test_accuracy = 60.46, Num_trees = 200, Learning_rate = 0.010, Depth = 5
test_accuracy = 61.71, Num_trees = 200, Learning_rate = 0.017, Depth = 5
test_accuracy = 63.20, Num_trees = 200, Learning_rate = 0.028, Depth = 5
test_accuracy = 63.43, Num_trees = 200, Learning_rate = 0.046, Depth = 5
test_accuracy = 63.83, Num_trees = 200, Learning_rate = 0.077, Depth = 5
test_accuracy = 63.54, Num_trees = 200, Learning_rate = 0.129, Depth = 5
test_accuracy = 64.00, Num_trees = 200, Learning_rate = 0.215, Depth = 5
test_accuracy = 61.14, Num_trees = 200, Learning_rate = 0.359, Depth = 5
test_accuracy = 61.94, Num_trees = 200, Learning_rate = 0.599, Depth = 5
test_accuracy = 54.29, Num_trees = 200, Learning_rate = 1.000, Depth = 5
```

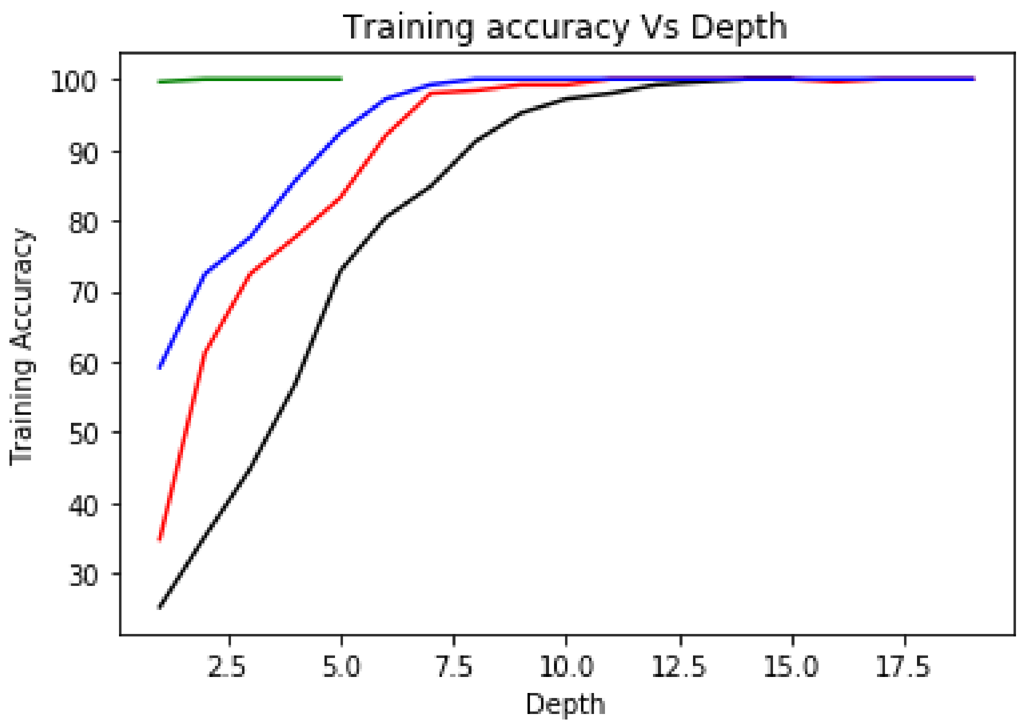


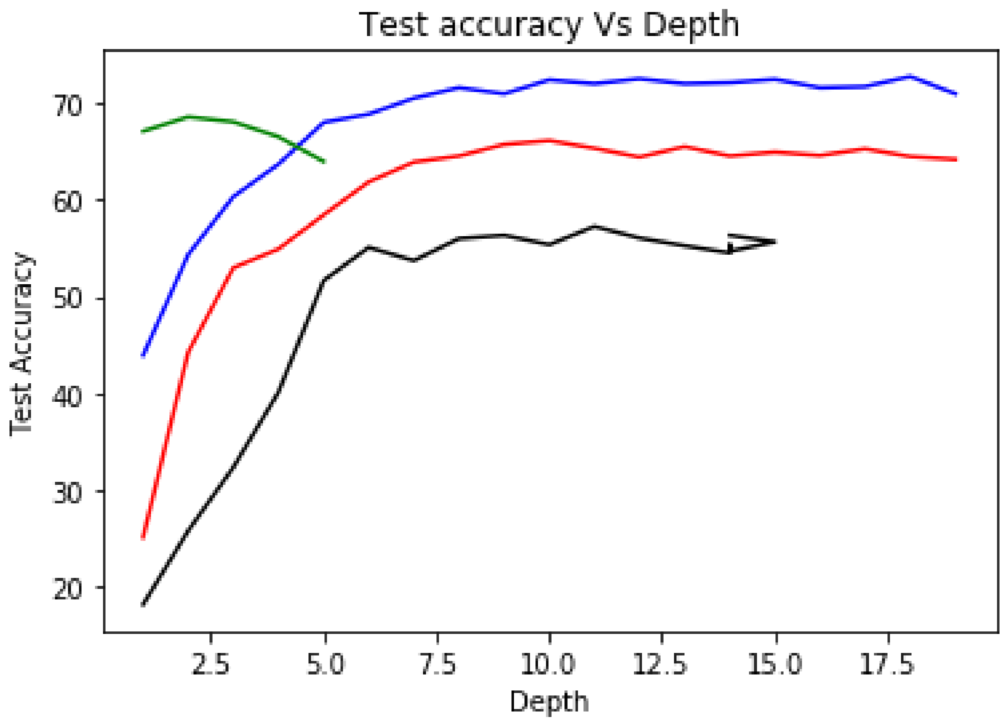


```
Time: 0:05:08.987499
Based on maximum test accuracy: depth = 2, best_num_tree = 50 & learning_rate =
0.3594 with training_accuracy = 100.00 & testing_accuracy = 68.63
```

5.5 E.

[255]:
```
plt.plot(depth_dt, tr_acc_dt, color = 'black')
k = range(1,20)
plt.plot(k, tr_acc_rf, color = 'r')
plt.plot(k, tr_acc_rfs, color = 'b')
k = range(1,6)
plt.plot(k, tr_acc_gb, color = 'g')
plt.xlabel("Depth")
plt.ylabel("Training Accuracy")
plt.title("Training accuracy Vs Depth")
plt.show()

plt.plot(depth_dt, ts_acc_dt, color = 'black')
k = range(1,20)
plt.plot(k, ts_acc_rf, color = 'r')
plt.plot(k, ts_acc_rfs, color = 'b')
k = range(1,6)
plt.plot(k, ts_acc_gb, color = 'g')
plt.xlabel("Depth")
plt.ylabel("Test Accuracy")
plt.title("Test accuracy Vs Depth")
plt.show()
```





Based on the training accuracy, clearly as the depth increases, training accuracy increases and after a fixed depth though different for each method training accuracy becomes 100%. However, for the test accuracy, until the depth reaches to an optimal value test accuracy increases and reaches to the highest accuracy and after that optimal depth, test accuracy changes but less than the value at optimal depth.

Gradient Boosting took the highest time and obviously single tree took the least time. It is important to mention that bagging (Random Forest) with no sampling on variables took higher time than random Forest with sampling on variables because with no sampling on variables needs to consider all variables for Gini index calculations.

[ ]: