In [1]:

```
import pandas as pd
import os
import joblib as jb
import sklearn
import pydotplus
```

In [2]:

```
from sklearn.preprocessing import LabelEncoder
```

In [3]:

```
pd.read_excel('Combined_Updated.xlsx')
```

Out[3]:

MaritalStatus	Occupation	Relationship	Sex	NativeCountry	Race	FnlwgtCategory	CapitalGainCategory	CapitalLossCategory	HoursPerWeekCategory	Class
4	0	1	1	38	4	3	0	1	2	0
2	3	0	1	38	4	3	2	1	2	0
0	5	1	1	38	4	0	2	1	2	0
2	5	0	1	38	2	0	2	1	2	0
2	9	5	0	4	2	0	2	1	2	0
...	...	...	...	...	...	...	...	...	...	...
0	9	1	0	38	4	0	2	1	2	0
6	9	2	1	38	2	0	2	1	2	0
2	9	0	1	38	4	0	2	1	0	0
0	0	3	1	38	1	3	0	1	2	0
2	3	0	1	38	4	0	2	1	0	1

In [5]:

```
data=pd.read_excel('Combined_Updated.xlsx')
```

In [7]:

```
xc=['AgeCategory','Workclass','Education','EducationNum','MaritalStatus','Occupation','Relationship','NativeCountry','Sex','Race','FnlwgtC
y=['Yes','No']
all_input=data[xc]
all_class=data['Class']
```

In [9]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [10]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_input,all_class,train_size=0.67,random_state=10)
```

In [11]:

```
clf = DecisionTreeClassifier(random_state=10)
clf.fit(x_train,y_train)
```

Out[11]:

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=10)
```

In [15]:

```
y_train_pred=clf.predict(x_train)

y_test_pred=clf.predict(x_test)
```

In [13]:

```
print(y_train_pred)
y_train
```

```
[0 1 0 ... 0 0 0]
```

Out[13]:

```
39203    0
16702    1
43825    0
48735    1
34480    0
        ..
40059    1
28017    0
29199    0
40061    0
17673    0
Name: Class, Length: 32724, dtype: int64
```

In [14]:

```
from sklearn import metrics,model_selection,preprocessing
wrong_train_pred=(y_train !=y_train_pred).sum()
print("Total wrong detected on training data= {}".format(wrong_train_pred))

accuracy_train=metrics.accuracy_score(y_train,y_train_pred)
print("Accuracy of this model on training data= {:.3f}".format(accuracy_train))
```

```
Total wrong detected on training data= 2812
Accuracy of this model on training data= 0.914
```

In [16]:

```
wrong_test_pred=(y_test !=y_test_pred).sum()
print("Total wrong detected on test data = {}".format(wrong_test_pred))

accuracy_test=metrics.accuracy_score(y_test,y_test_pred)
print("Accuracy of this model on test data = {:.3f}".format(accuracy_test))
```

```
Total wrong detected on test data = 3058
Accuracy of this model on test data = 0.810
```

In [19]:

```
train_accuracy=[]
test_accuracy=[]
train_error=[]
valid_error=[]
test_error=[]
for depth in range(1,40):
    dt_model_tree=DecisionTreeClassifier(max_depth=depth,random_state=10)
    dt_model_tree.fit(x_train,y_train)
    train_accuracy.append(dt_model_tree.score(x_train,y_train))
    test_accuracy.append(dt_model_tree.score(x_test,y_test))
```

In [20]:

```
frame = pd.DataFrame({'max_depth': range(1,40),'train_acc':train_accuracy,'test_acc':test_accuracy})
frame.head()
```

Out[20]:

	max_depth	train_acc	test_acc
0	1	0.761062	0.760020
1	2	0.814876	0.812446
2	3	0.821691	0.819705
3	4	0.831653	0.828577
4	5	0.832508	0.829445

In [21]:

```
import numpy as np

train_accuracy = np.array(train_accuracy)
train_error = (1 - train_accuracy) * 32562

test_accuracy = np.array(test_accuracy)
test_error = (1 - test_accuracy) * 8140
```

In [22]:

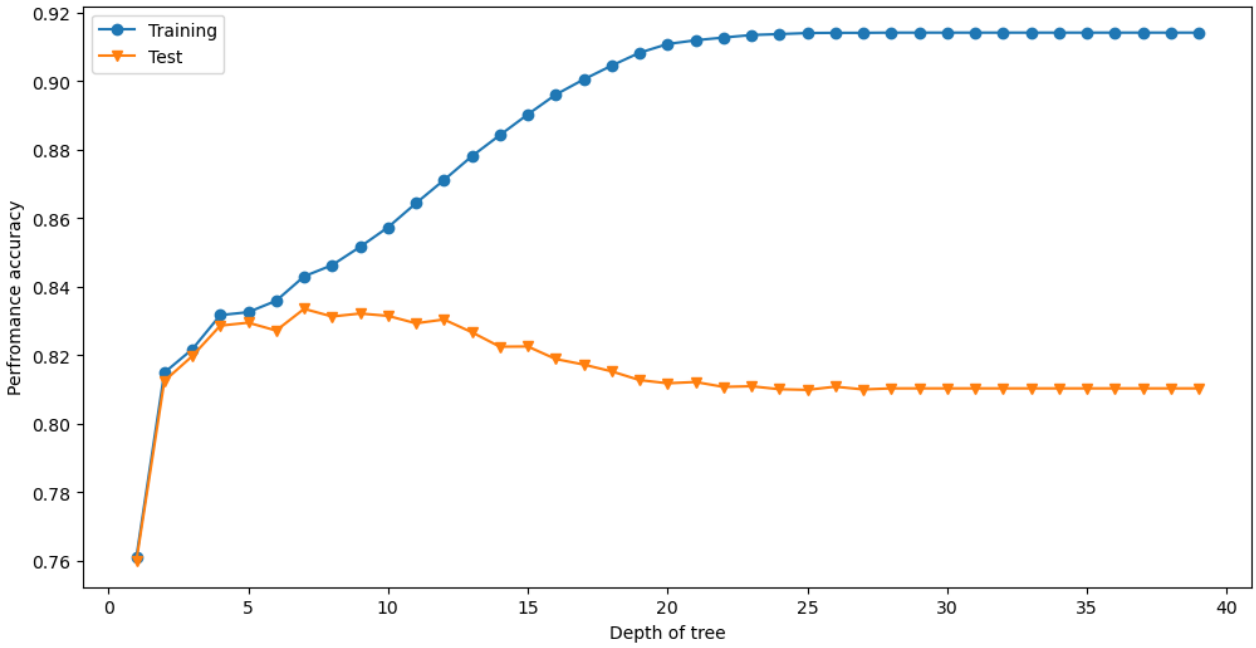
```
from IPython.display import Image,display
import matplotlib.pyplot as plt,pydotplus
import graphviz
```

In [23]:

```
import matplotlib.pyplot as plt,pydotplus
plt.figure(figsize=(12,6))
plt.plot(frame['max_depth'],frame['train_acc'],label='Training',marker='o')
plt.plot(frame['max_depth'],frame['test_acc'],label='Test',marker='v')
plt.xlabel('Depth of tree')
plt.ylabel('Perfromance accuracy')
plt.legend()
```

Out[23]:

```
<matplotlib.legend.Legend at 0x1f2fa13fd00>
```



In [24]:

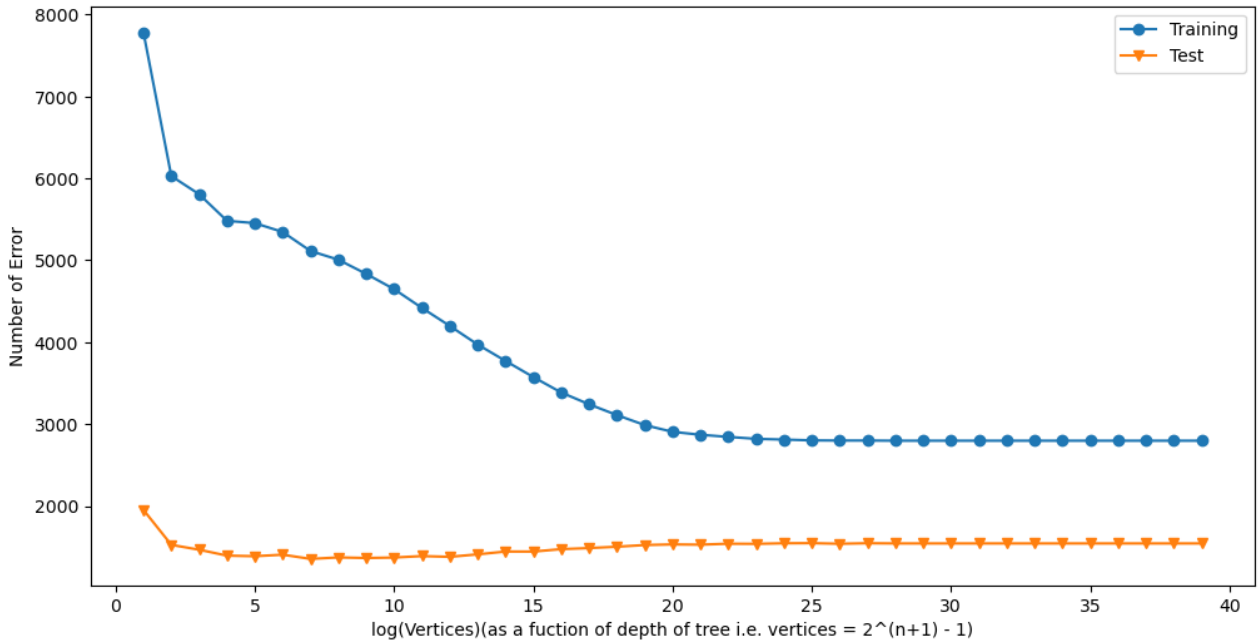
```
frame1 = pd.DataFrame({'max_depth': range(1,40),'train_err':train_error,'test_err':test_error})
```

In [25]:

```
import matplotlib.pyplot as plt,pydotplus
plt.figure(figsize=(12,6))
plt.plot(frame1['max_depth'],frame1['train_err'],label='Training',marker='o')
plt.plot(frame1['max_depth'],frame1['test_err'],label='Test',marker='v')
plt.xlabel('log(Vertices)(as a fuction of depth of tree i.e. vertices = 2^(n+1) - 1)')
plt.ylabel('Number of Error')
plt.legend()
```

Out[25]:

```
<matplotlib.legend.Legend at 0x1f2faba34c0>
```



In [28]:

```
from sklearn import tree
plt.figure(figsize=(200, 20))
tree.plot_tree(clf, filled=True,max_depth=4, feature_names=['HoursPerWeekCategory','CapitalLossCategory','CapitalGainCategory','HoursPerWe
```

```
 Text(0.8125, 0.4166666666666667, 'NativeCountry <= 3.5\ngini = 0.435\nsamples = 914\nvalue = [622, 292]\nclass = Yes'),
 Text(0.78125, 0.25, 'Education <= 1.5\ngini = 0.497\nsamples = 384\nvalue = [206, 178]\nclass = Yes'),
 Text(0.765625, 0.08333333333333333, '\n  (...)  \n'),
 Text(0.796875, 0.08333333333333333, '\n  (...)  \n'),
 Text(0.84375, 0.25, 'NativeCountry <= 8.5\ngini = 0.338\nsamples = 530\nvalue = [416, 114]\nclass = Yes'),
 Text(0.828125, 0.08333333333333333, '\n  (...)  \n'),
 Text(0.859375, 0.08333333333333333, '\n  (...)  \n'),
 Text(0.9375, 0.4166666666666667, 'HoursPerWeekCategory <= 12.5\ngini = 0.436\nsamples = 623\nvalue = [200, 423]\nclass
= No'),
 Text(0.90625, 0.25, 'Education <= 1.5\ngini = 0.473\nsamples = 454\nvalue = [174, 280]\nclass = No'),
 Text(0.890625, 0.08333333333333333, '\n  (...)  \n'),
 Text(0.921875, 0.08333333333333333, '\n  (...)  \n'),
 Text(0.96875, 0.25, 'NativeCountry <= 11.5\ngini = 0.26\nsamples = 169\nvalue = [26, 143]\nclass = No'),
 Text(0.953125, 0.08333333333333333, '\n  (...)  \n'),
 Text(0.984375, 0.08333333333333333, '\n  (...)  \n')]
```

In [ ]: