In [1]:

```
import pandas as pd
import os
import numpy as np
import seaborn as  sns
import joblib as jb
import sklearn
import pydotplus
import matplotlib.pyplot as plt,pydotplus
```

In [2]:

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

In [3]:

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

Out[3]:

	AgeCategory	Workclass	Education	EducationNum	MaritalStatus	Occupation	Relationship	Sex	NativeCountry	Race	FnlwgtC
0	3	6	9	12	4	0	1	1	38	4	
1	0	5	9	12	2	3	0	1	38	4	
2	3	3	11	8	0	5	1	1	38	4	
3	0	3	1	6	2	5	0	1	38	2	
4	3	3	9	12	2	9	5	0	4	2	
...	...	...	...	...	...	...	...	...	...	...	
48837	3	3	9	12	0	9	1	0	38	4	
48838	2	3	11	8	6	9	2	1	38	2	
48839	3	3	9	12	2	9	0	1	38	4	
48840	0	3	9	12	0	0	3	1	38	1	
48841	3	4	9	12	2	3	0	1	38	4	

48842 rows × 15 columns

In [4]:

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

In [5]:

```
data.describe()
```

Out[5]:

	AgeCategory	Workclass	Education	EducationNum	MaritalStatus	Occupation	Relationship	Sex	NativeCountr
count	48842.000000	48842.000000	48842.000000	48842.000000	48842.000000	48842.000000	48842.000000	48842.000000	48842.00000
mean	1.757893	3.099668	10.288420	9.078089	2.618750	6.152819	1.443287	0.668482	36.43366
std	1.373868	1.110810	3.874492	2.570973	1.507703	3.968837	1.602151	0.470764	6.03153
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	0.000000	3.000000	9.000000	8.000000	2.000000	3.000000	0.000000	0.000000	38.00000
50%	3.000000	3.000000	11.000000	9.000000	2.000000	7.000000	1.000000	1.000000	38.00000
75%	3.000000	3.000000	12.000000	11.000000	4.000000	9.000000	3.000000	1.000000	38.00000
max	3.000000	7.000000	15.000000	15.000000	6.000000	13.000000	5.000000	1.000000	40.00000

In [6]:

```
data["Class"].value_counts()
```

Out[6]:

```
0    37155
1    11687
Name: Class, dtype: int64
```

In [9]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
xc=['AgeCategory','Workclass','Education','EducationNum','MaritalStatus','Occupation','Relationship','NativeCountry','Sex',
y=['Yes','No']
all_input=data[xc]
all_class=data['Class']
```

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]:

```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(criterion="gini",max_depth=8,random_state=5)
```

In [13]:

```
clf.fit(x_train, y_train)
```

Out[13]:

```
▾               RandomForestClassifier
RandomForestClassifier(max_depth=8, random_state=5)
```

In [21]:

```
Y_train_pred=clf.predict(x_train)

Y_test_pred=clf.predict(x_test)
```

In [17]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_train,Y_train_pred)
```

Out[17]:

```
0.8458318054027625
```

In [19]:

```
from sklearn.model_selection import cross_val_score
cross_val_score(clf,x_train, y_train,cv=10)
```

Out[19]:

```
array([0.83195845, 0.84631836, 0.8395967 , 0.83226398, 0.83251834,
       0.84749389, 0.83832518, 0.8358802 , 0.84107579, 0.84535452])
```

In [23]:

```
from sklearn.metrics import classification_report
print(classification_report(Y_test_pred,y_test))
```

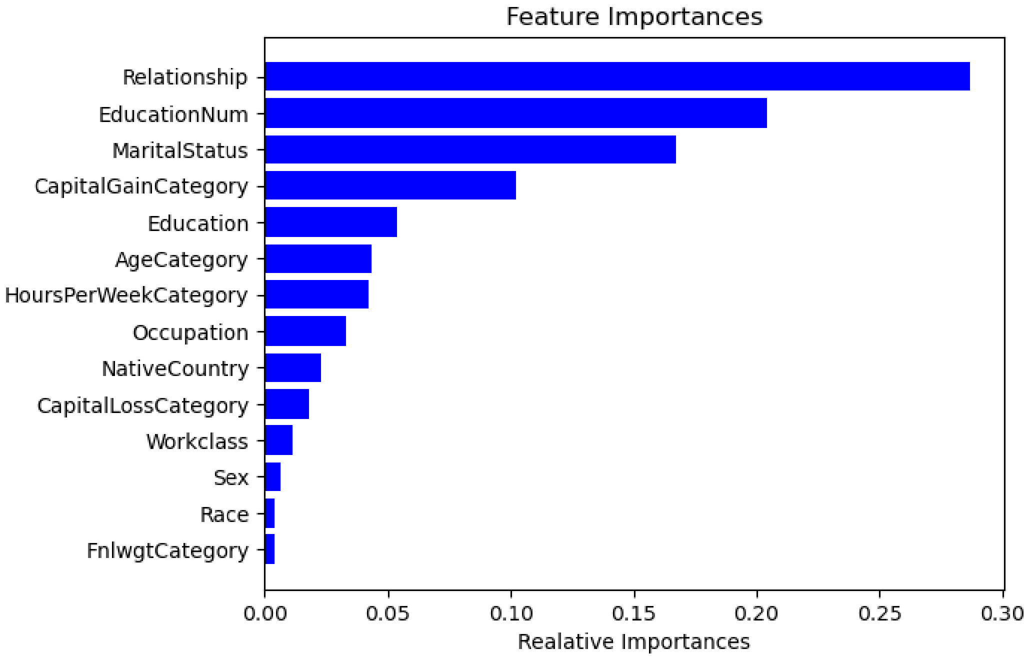
```
              precision    recall  f1-score   support

           0       0.94      0.86      0.90     13331
           1       0.52      0.72      0.61      2787

    accuracy                           0.84     16118
   macro avg       0.73      0.79      0.75     16118
weighted avg       0.87      0.84      0.85     16118
```

In [25]:

```
features = data.columns
importances=clf.feature_importances_
indices=np.argsort(importances)
```

In [27]:

```
plt.title('Feature Importances')
plt.barh(range(len(indices)),importances[indices],color='b',align='center')
plt.yticks(range(len(indices)),[features[i]for i in indices])
plt.xlabel('Realative Importances')
plt.show()
```



In [29]:

```
del data['AgeCategory']
del data['Workclass']
del data['Education']
del data['EducationNum']
del data['Occupation']
del data['Sex']
del data['FnlwgtCategory']
del data['Race']
del data['CapitalLossCategory']
del data['HoursPerWeekCategory']
```

In [30]:

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

In [31]:

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

In [32]:

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

Out[32]:

```
▾        DecisionTreeClassifier

DecisionTreeClassifier(random_state=10)
```

In [33]:

```
y_train_pred=clf.predict(x_train)

y_test_pred=clf.predict(x_test)
```

In [34]:

```
print(y_train_pred)
y_train
```

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

Out[34]:

```
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 [35]:

```
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 [37]:

```
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 [38]:

```
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 [39]:

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

Out[39]:

	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 [40]:

```
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 [41]:

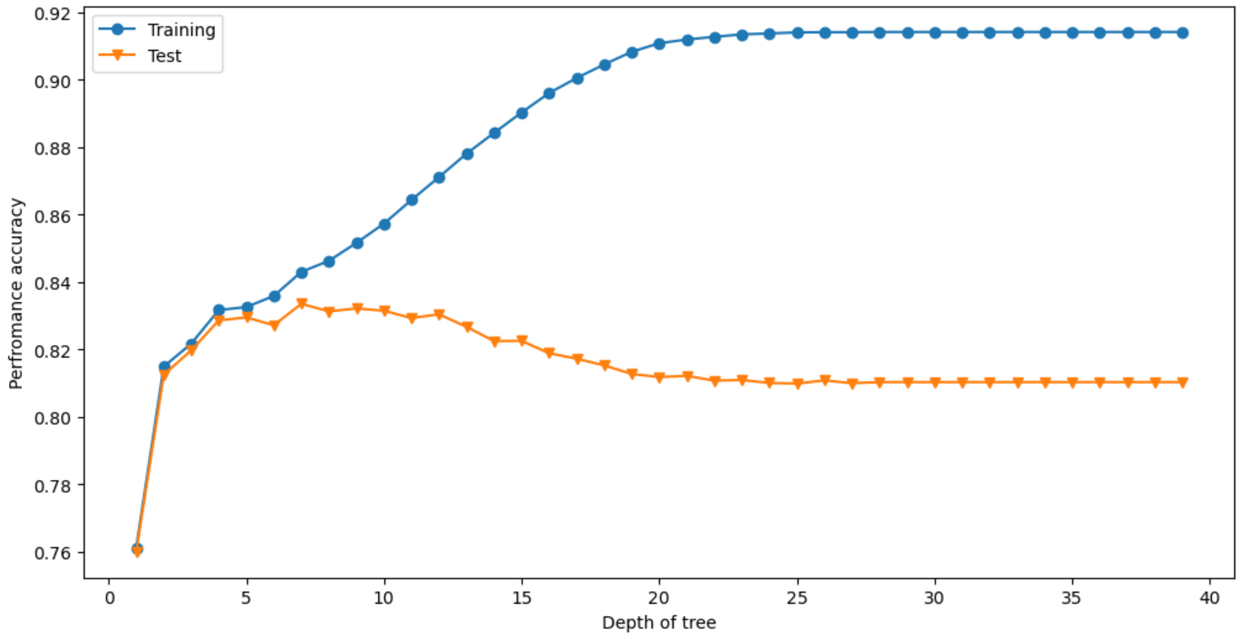
```
from IPython.display import Image,display
import matplotlib.pyplot as plt,pydotplus
import graphviz
```

In [42]:

```
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[42]:

```
<matplotlib.legend.Legend at 0x288d5530580>
```



In [43]:

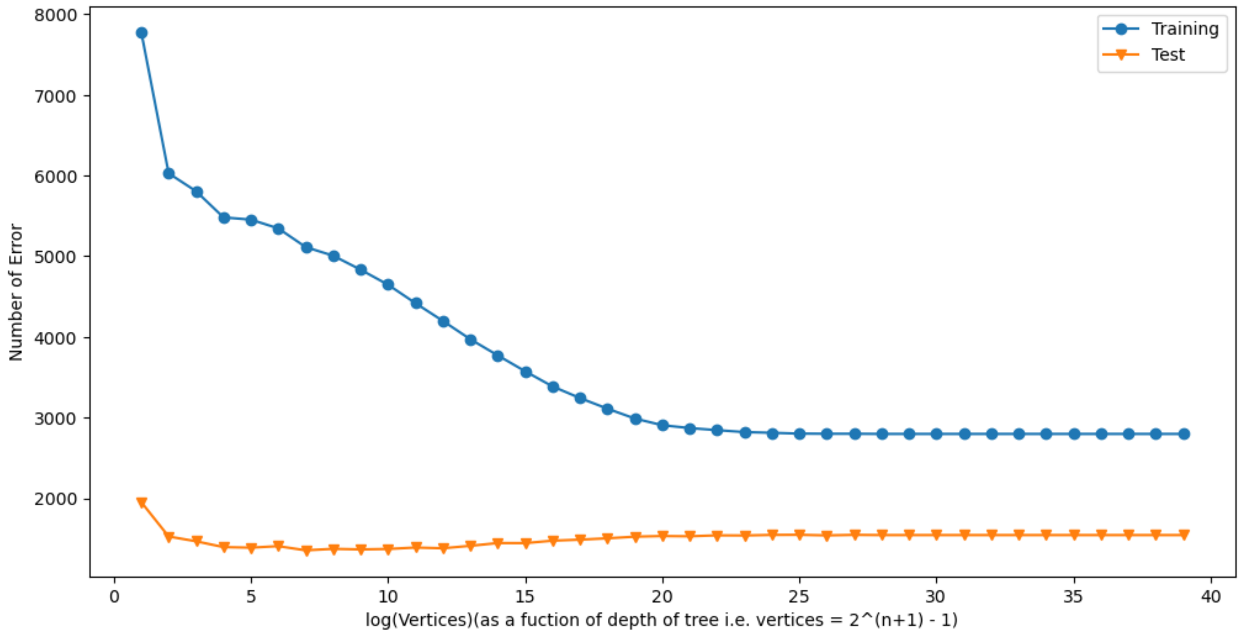
```
frame1 = pd.DataFrame({'max_depth': range(1,40),'train_err':train_error,'test_err':test_error})
```

In [44]:

```
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[44]:

```
<matplotlib.legend.Legend at 0x288d5594760>
```



In [49]:

```
from sklearn import tree
plt.figure(figsize=(200, 20))
tree.plot_tree(clf, filled=True,max_depth=5, feature_names=['HoursPerWeekCategory','CapitalLossCategory','CapitalGainCatego
```

```
 Text(0.5785123966942148, 0.21428571428571427, 'EducationNum <= 2.5\ngini = 0.353\nsamples = 140\nvalue =
[32, 108]\nclass = No'),
 Text(0.5702479338842975, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.5867768595041323, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6115702479338843, 0.21428571428571427, 'HoursPerWeekCategory <= 13.5\ngini = 0.499\nsamples = 187
\nvalue = [90, 97]\nclass = No'),
 Text(0.6033057851239669, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6198347107438017, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6942148760330579, 0.5, 'HoursPerWeekCategory <= 12.5\ngini = 0.095\nsamples = 17250\nvalue = [1639
2, 858]\nclass = Yes'),
 Text(0.6611570247933884, 0.35714285714285715, 'HoursPerWeekCategory <= 11.5\ngini = 0.072\nsamples = 1632
6\nvalue = [15716, 610]\nclass = Yes'),
 Text(0.6446280991735537, 0.21428571428571427, 'AgeCategory <= 1.5\ngini = 0.048\nsamples = 13824\nvalue =
[13487, 337]\nclass = Yes'),
 Text(0.6363636363636364, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6528925619834711, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6776859504132231, 0.21428571428571427, 'AgeCategory <= 1.5\ngini = 0.194\nsamples = 2502\nvalue =
[2229, 273]\nclass = Yes'),
 Text(0.6694214876033058, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6859504132231405, 0.07142857142857142, '\n  (...)  \n'),
```

In [ ]: