In [8]:

```python
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 [9]:

```python
train_data=pd.read_excel('Training_Updated.xlsx')
valid_data=test_data=pd.read_excel('Validation_final.xlsx')
test_data=pd.read_excel('Test_final.xlsx')
```

In [10]:

```python
train_data.head()
```

Out[10]:

MaritalStatus	Occupation	Relationship	Sex	NativeCountry	Race	FnlwgtCategory	CapitalGainCategory	CapitalLossCategory	Hours
4	0	1	1	38	4	3	0	1	
2	3	0	1	38	4	3	2	1	
0	5	1	1	38	4	0	2	1	
2	5	0	1	38	2	0	2	1	
2	9	5	0	4	2	0	2	1	

In [11]:

```python
train_data.tail()
```

Out[11]:

MaritalStatus	Occupation	Relationship	Sex	NativeCountry	Race	FnlwgtCategory	CapitalGainCategory	CapitalLossCategory	Hours
2	12	5	0	38	4	0	2	1	
2	6	0	1	38	4	0	2	1	
6	0	4	0	38	4	0	2	1	
4	0	3	1	38	4	0	2	1	
2	3	5	0	38	4	0	0	1	

In [12]:

```python
train_data.describe()
```

Out[12]:

	AgeCategory	Workclass	Education	EducationNum	MaritalStatus	Occupation	Relationship	Sex	Native(
count	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.
mean	1.756641	3.094438	10.298210	9.080679	2.611836	6.138755	1.446362	0.669205	36.
std	1.375836	1.107194	3.870264	2.572720	1.506222	3.972708	1.606771	0.470506	6.
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	0.000000	3.000000	9.000000	8.000000	2.000000	3.000000	0.000000	0.000000	38.
50%	3.000000	3.000000	11.000000	9.000000	2.000000	6.000000	1.000000	1.000000	38.
75%	3.000000	3.000000	12.000000	11.000000	4.000000	9.000000	3.000000	1.000000	38.
max	3.000000	7.000000	15.000000	15.000000	6.000000	13.000000	5.000000	1.000000	40.

In [15]:

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

Out[15]:

```
0    24720
1     7841
Name: Class, dtype: int64
```

In [23]:

```
train_data_plot=train_data.sample(n=32561)
valid_data_plot=valid_data.sample(n=8141)
test_data_plot=test_data.sample(n=8140)

X_train_plot=train_data_plot.drop(['Class'],axis=1)
Y_train_plot=train_data_plot['Class']

X_valid_plot=valid_data_plot.drop(['Class'],axis=1)
Y_valid_plot=valid_data_plot['Class']

X_test_plot=test_data_plot.drop(['Class'],axis=1)
Y_test_plot=test_data_plot['Class']
```

In [24]:

```
X_train_plot.shape
```

Out[24]:

```
(32561, 14)
```

In [25]:

```
Y_train_plot.shape
```

Out[25]:

```
(32561,)
```

In [22]:

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

In [26]:

```
from sklearn import tree
```

In [27]:

```
clf.fit(X_train_plot, Y_train_plot)
```

Out[27]:

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

In [29]:

```
Y_train_pred=clf.predict(X_train_plot)

Y_valid_pred=clf.predict(X_valid_plot)

Y_test_pred=clf.predict(X_test_plot)
```

In [30]:

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

Out[30]:

```
0.8441079819415865
```

In [31]:

```
from sklearn.model_selection import cross_val_score
cross_val_score(clf,X_train_plot, Y_train_plot,cv=10)
```

Out[31]:

```
array([0.83512435, 0.83998771, 0.83814496, 0.84090909, 0.83445946,
       0.82493857, 0.8470516 , 0.84183047, 0.83138821, 0.84121622])
```

In [33]:

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

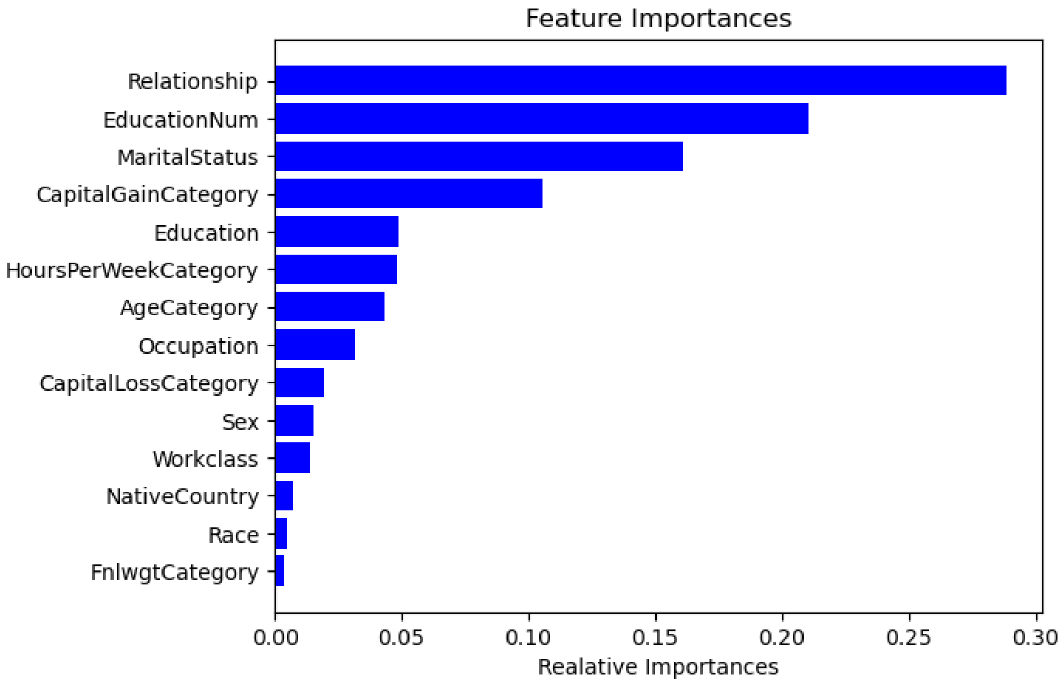
```
              precision    recall  f1-score   support

           0       0.94      0.86      0.90      6787
           1       0.51      0.71      0.59      1353

    accuracy                           0.84      8140
   macro avg       0.72      0.79      0.75      8140
weighted avg       0.87      0.84      0.85      8140
```

In [38]:

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

In [60]:

```
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 [42]:

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

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

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

In [43]:

```
train_data_plot=train_data.sample(n=32561)
valid_data_plot=valid_data.sample(n=8141)
test_data_plot=test_data.sample(n=8140)



X_train_plot=train_data_plot.drop(['Class'],axis=1)
Y_train_plot=train_data_plot['Class']

X_valid_plot=valid_data_plot.drop(['Class'],axis=1)
Y_valid_plot=valid_data_plot['Class']

X_test_plot=test_data_plot.drop(['Class'],axis=1)
Y_test_plot=test_data_plot['Class']
```

In [44]:

```
from sklearn import tree
model_tree = tree.DecisionTreeClassifier()
model_tree.fit(X_train_plot, Y_train_plot)
```

Out[44]:

```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [45]:

```
Y_train_pred=model_tree.predict(X_train_plot)

Y_valid_pred=model_tree.predict(X_valid_plot)

Y_test_pred=model_tree.predict(X_test_plot)
```

In [46]:

```
print(Y_train_pred)
Y_train_plot
```

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

Out[46]:

```
29716    0
10496    1
30975    1
5916     0
5238     1
        ..
185      0
26844    0
15067    0
32219    0
14675    0
Name: Class, Length: 32561, dtype: int64
```

In [47]:

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

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

```
Total wrong detected on training data= 6838
Accuracy of this model on training data= 0.790
```

In [48]:

```
wrong_valid_pred=(Y_valid_plot !=Y_valid_pred).sum()
print("Total wrong detected on validation data = {}".format(wrong_valid_pred))

accuracy_valid=metrics.accuracy_score(Y_valid_plot,Y_valid_pred)
print("Accuracy of this model on validation data = {:.3f}".format(accuracy_valid))
```

```
Total wrong detected on validation data = 1752
Accuracy of this model on validation data = 0.785
```

In [49]:

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

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

```
Total wrong detected on test data = 1670
Accuracy of this model on test data = 0.795
```

In [51]:

```
from sklearn.tree import DecisionTreeClassifier
train_accuracy=[]
valid_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_plot,Y_train_plot)
    train_accuracy.append(dt_model_tree.score(X_train_plot,Y_train_plot))
    valid_accuracy.append(dt_model_tree.score(X_valid_plot,Y_valid_plot))
    test_accuracy.append(dt_model_tree.score(X_test_plot,Y_test_plot))
```

In [52]:

```
import numpy as np

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

valid_accuracy = np.array(valid_accuracy)
valid_error = (1 - valid_accuracy) * 8141

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

In [53]:

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

Out[53]:

	max_depth	train_acc	valid_acc	test_acc
0	1	0.759190	0.760472	0.767076
1	2	0.783330	0.783196	0.791523
2	3	0.785940	0.786021	0.794840
3	4	0.786862	0.787004	0.796192
4	5	0.787507	0.786636	0.796929

In [54]:

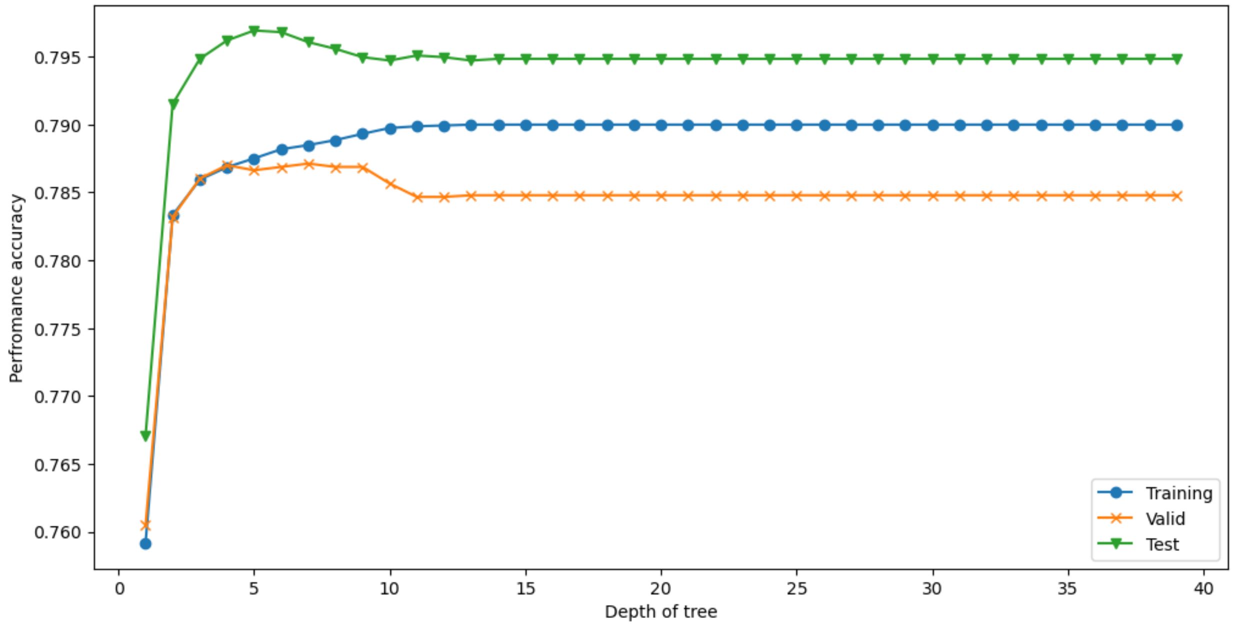
```
from IPython.display import Image,display
import matplotlib.pyplot as plt,pydotplus
import graphviz
```

In [55]:

```
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['valid_acc'],label='Valid',marker='x')
plt.plot(frame['max_depth'],frame['test_acc'],label='Test',marker='v')
plt.xlabel('Depth of tree')
plt.ylabel('Perfromance accuracy')
plt.legend()
```

Out[55]:

```
<matplotlib.legend.Legend at 0x275a7634d90>
```



In [56]:

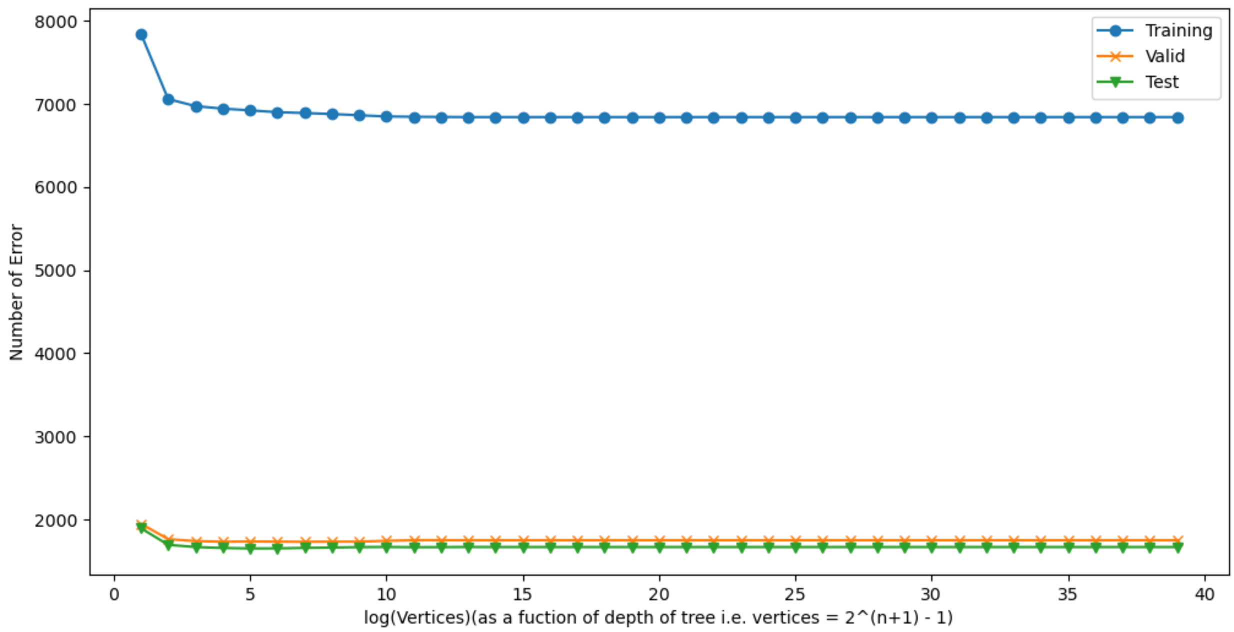
```
frame1 = pd.DataFrame({'max_depth': range(1,40),'train_err':train_error,'valid_err':valid_error,'test_err':test_error}
```

In [57]:

```
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['valid_err'],label='Valid',marker='x')
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[57]:

```
<matplotlib.legend.Legend at 0x275a765f850>
```



In [59]:

```
200, 20))
_tree, filled=True,max_depth=10, feature_names=['CapitalGainCategory','Relationship','MaritalStatus','EducationNum'],cl
```

```
 Text(0.8039210526315579, 0.2916666666666667, 'MaritalStatus <= 31.5\ngini = 0.048\nsamples = 1250\nva
lue = [1219, 31]\nclass = Yes'),
 Text(0.7993421052631579, 0.20833333333333334, 'MaritalStatus <= 30.5\ngini = 0.198\nsamples = 18\nva
lue = [16, 2]\nclass = Yes'),
 Text(0.7960526315789473, 0.125, 'CapitalGainCategory <= 4.5\ngini = 0.111\nsamples = 17\nvalue = [1
6, 1]\nclass = Yes'),
 Text(0.7927631578947368, 0.041666666666666664, '\n  (...)  \n'),
 Text(0.7993421052631579, 0.041666666666666664, '\n  (...)  \n'),
 Text(0.8026315789473685, 0.125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]\nclass = No'),
 Text(0.8125, 0.20833333333333334, 'MaritalStatus <= 37.5\ngini = 0.046\nsamples = 1232\nvalue = [120
3, 29]\nclass = Yes'),
 Text(0.8092105263157895, 0.125, 'gini = 0.0\nsamples = 30\nvalue = [30, 0]\nclass = Yes'),
 Text(0.8157894736842105, 0.125, 'CapitalGainCategory <= 3.5\ngini = 0.047\nsamples = 1202\nvalue =
[1173, 29]\nclass = Yes'),
 Text(0.8125, 0.041666666666666664, '\n  (...)  \n'),
 Text(0.819078947368421, 0.041666666666666664, '\n  (...)  \n'),
 Text(0.8363486842105263, 0.4583333333333333, 'Relationship <= 3.5\ngini = 0.127\nsamples = 424\nvalu
e = [395, 29]\nclass = Yes'),
 Text(0.8256578947368421, 0.375, 'Relationship <= 2.5\ngini = 0.032\nsamples = 61\nvalue = [60, 1]\nc
lass = Yes'),
```

In [ ]: