In [2]:

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

In [41]:

```
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 [35]:

```
from sklearn.model_selection import train_test_split
```

In [36]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [37]:

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   AgeCategory           32561 non-null  int64
 1   Workclass             32561 non-null  int64
 2   Education             32561 non-null  int64
 3   EducationNum          32561 non-null  int64
 4   MaritalStatus         32561 non-null  int64
 5   Occupation            32561 non-null  int64
 6   Relationship          32561 non-null  int64
 7   Sex                   32561 non-null  int64
 8   NativeCountry         32561 non-null  int64
 9   Race                  32561 non-null  int64
 10  FnlwgtCategory        32561 non-null  int64
 11  CapitalGainCategory   32561 non-null  int64
 12  CapitalLossCategory   32561 non-null  int64
 13  HoursPerWeekCategory  32561 non-null  int64
 14  Class                 32561 non-null  int64
dtypes: int64(15)
memory usage: 3.7 MB
```

In [43]:

```
X=train_data.drop(['Class'],axis=1)
Y=train_data['Class']

X_valid=valid_data.drop(['Class'],axis=1)
Y_valid=valid_data['Class']

X_test=test_data.drop(['Class'],axis=1)
Y_test=test_data['Class']


X.shape,Y.shape,X_valid.shape,Y_valid.shape,X_test.shape,Y_test.shape
```

Out[43]:

```
((32561, 14), (32561,), (8141, 14), (8141,), (8140, 14), (8140,))
```

In [44]:

```
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 [45]:

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

Out[45]:

```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [58]:

```
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 [48]:

```
print(Y_train_pred)
Y_train_plot
```

...

In [55]:

```
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= 2815
Accuracy of this model on training data= 0.914
```

In [56]:

```
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 = 1565
Accuracy of this model on validation data = 0.808
```

In [59]:

```
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 = 1509
Accuracy of this model on test data = 0.815
```

In [71]:

```
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 [89]:

```
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 [72]:

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

Out[72]:

	max_depth	train_acc	valid_acc	test_acc
0	1	0.759190	0.760472	0.767076
1	2	0.813704	0.814888	0.814742
2	3	0.826633	0.828522	0.827887
3	4	0.830656	0.831962	0.830467
4	5	0.831762	0.831962	0.831450

In [28]:

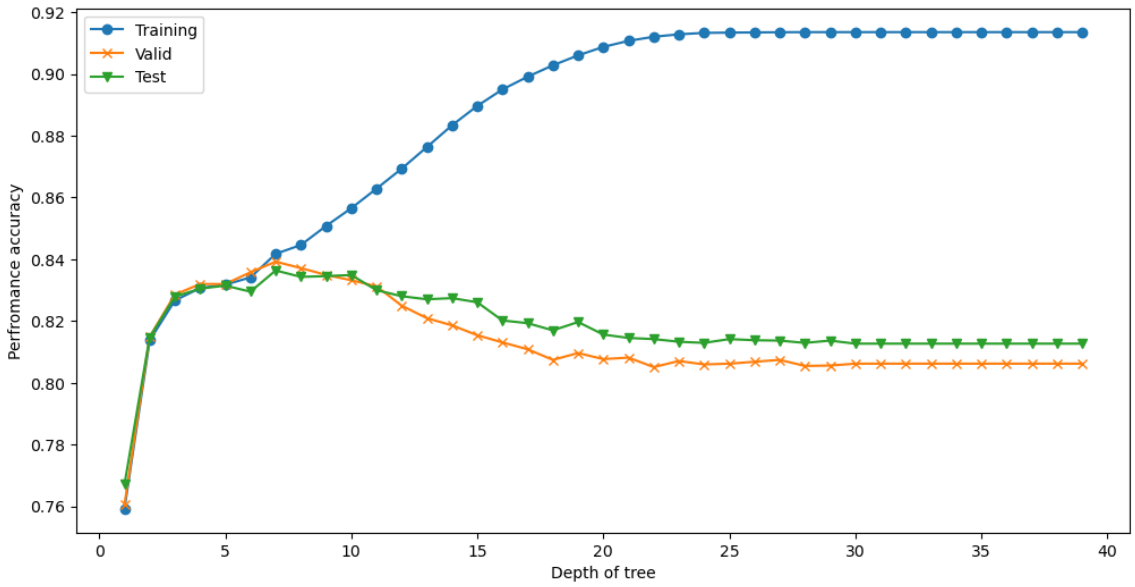
```
from IPython.display import Image,display
import matplotlib.pyplot as plt,pydotplus
import graphviz
```

In [94]:

```
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[94]:

```
<matplotlib.legend.Legend at 0x1d9ea898ee0>
```



In [100]:

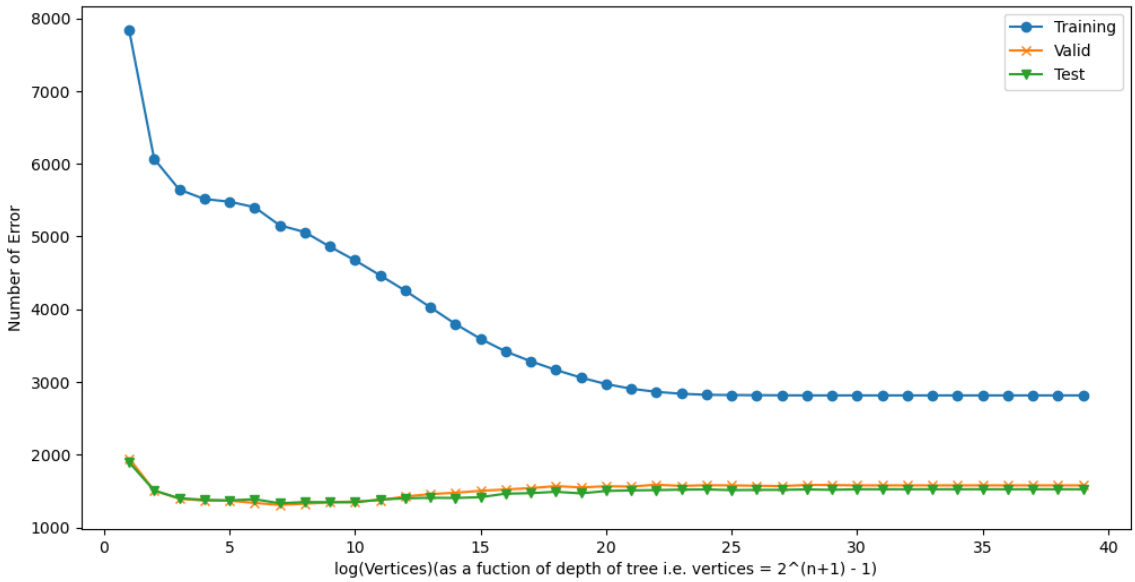
```
frame1 = pd.DataFrame({'max_depth': range(1,40),'train_err':train_error,'valid_err':vali
```

In [102]:

```
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[102]:

```
<matplotlib.legend.Legend at 0x1d9eae92920>
```



In [86]:

```
plt.figure(figsize=(200, 20))
tree.plot_tree(model_tree, filled=True,max_depth=7, feature_names=['HoursPerWeekCategory
```

```
11.5\ngini = 0.346\nsamples = 9\nvalue = [2, 7]\nclass = No'),
 Text(0.8947368421052632, 0.16666666666666666, 'gini = 0.0\nsamples = 2
\nvalue = [0, 2]\nclass = No'),
 Text(0.9, 0.16666666666666666, 'CapitalLossCategory <= 4.0\ngini = 0.4
08\nsamples = 7\nvalue = [2, 5]\nclass = No'),
 Text(0.8973684210526316, 0.05555555555555555, '\n  (...)  \n'),
 Text(0.9026315789473685, 0.05555555555555555, '\n  (...)  \n'),
 Text(0.9157894736842105, 0.2777777777777778, 'HoursPerWeekCategory <=
11.5\ngini = 0.116\nsamples = 81\nvalue = [5, 76]\nclass = No'),
 Text(0.9105263157894737, 0.16666666666666666, 'NativeCountry <= 10.0\n
gini = 0.26\nsamples = 13\nvalue = [2, 11]\nclass = No'),
 Text(0.9078947368421053, 0.05555555555555555, '\n  (...)  \n'),
 Text(0.9131578947368421, 0.05555555555555555, '\n  (...)  \n'),
 Text(0.9210526315789473, 0.16666666666666666, 'HoursPerWeekCategory <=
2.5\ngini = 0.084\nsamples = 68\nvalue = [3, 65]\nclass = No'),
 Text(0.9184210526315789, 0.05555555555555555, '\n  (...)  \n'),
 Text(0.9236842105263158, 0.05555555555555555, '\n  (...)  \n'),
 Text(0.9585526315789473, 0.5, 'HoursPerWeekCategory <= 12.5\ngini = 0.
462\nsamples = 531\nvalue = [192, 339]\nclass = No'),
```

In [ ]: