

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [2]:

```
df = pd.read_csv('diabetes_data_upload.csv')
df.head()
```

Out[2]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching
0	40	Male	No	Yes	No	Yes	No	No	No	Yes
1	58	Male	No	No	No	Yes	No	No	Yes	No
2	41	Male	Yes	No	No	Yes	Yes	No	No	Yes
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
#   Column                      Non-Null Count  Dtype
---  -
0   Age                         520 non-null   int64
1   Gender                      520 non-null   object
2   Polyuria                   520 non-null   object
3   Polydipsia                 520 non-null   object
4   sudden weight loss         520 non-null   object
5   weakness                   520 non-null   object
6   Polyphagia                 520 non-null   object
7   Genital thrush             520 non-null   object
8   visual blurring            520 non-null   object
9   Itching                    520 non-null   object
10  Irritability               520 non-null   object
11  delayed healing            520 non-null   object
12  partial paresis            520 non-null   object
13  muscle stiffness           520 non-null   object
14  Alopecia                   520 non-null   object
15  Obesity                    520 non-null   object
16  class                      520 non-null   object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

In [4]:

```
df.describe()
```

Out[4]:

	Age
count	520.000000
mean	48.028846
std	12.151466
min	16.000000
25%	39.000000
50%	47.500000
75%	57.000000
max	90.000000

In [5]:

```
df.describe(include="all")
```

Out[5]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	vi blur
count	520.000000	520	520	520	520	520	520	520	
unique	NaN	2	2	2	2	2	2	2	
top	NaN	Male	No	No	No	Yes	No	No	
freq	NaN	328	262	287	303	305	283	404	
mean	48.028846	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	12.151466	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	16.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	39.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	47.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	57.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	90.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	



In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Age                0
Gender             0
Polyuria           0
Polydipsia         0
sudden weight loss 0
weakness           0
Polyphagia         0
Genital thrush     0
visual blurring    0
Itching            0
Irritability       0
delayed healing    0
partial paresis    0
muscle stiffness   0
Alopecia           0
Obesity            0
class             0
dtype: int64
```

In [7]:

```
df['Gender'].value_counts()
```

Out[7]:

```
Male      328
Female    192
Name: Gender, dtype: int64
```

In [8]:

```
df['Polyuria'].value_counts()
```

Out[8]:

```
No      262
Yes     258
Name: Polyuria, dtype: int64
```

In [9]:

```
df['Polydipsia'].value_counts()
```

Out[9]:

```
No      287
Yes     233
Name: Polydipsia, dtype: int64
```

In [10]:

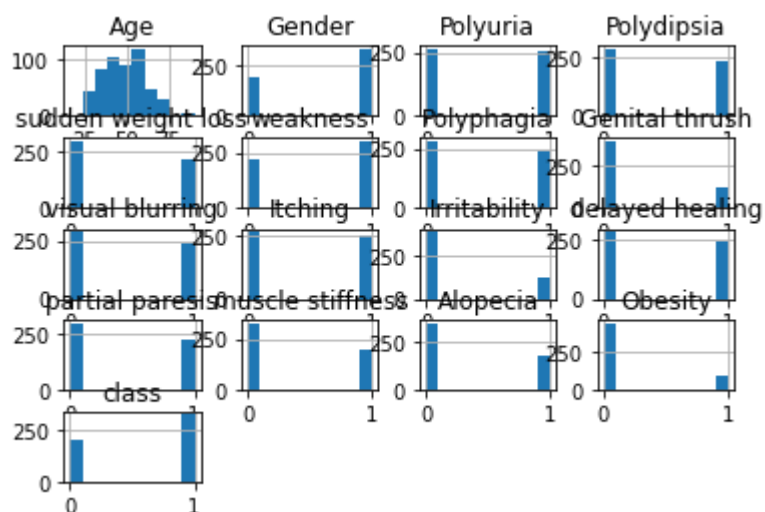
```
df['Gender'] = df['Gender'].map({'Male':1,'Female':0})
df['class'] = df['class'].map({'Positive':1,'Negative':0})
df['Polyuria'] = df['Polyuria'].map({'Yes':1,'No':0})
df['Polydipsia'] = df['Polydipsia'].map({'Yes':1,'No':0})
df['sudden weight loss'] = df['sudden weight loss'].map({'Yes':1,'No':0})
df['weakness'] = df['weakness'].map({'Yes':1,'No':0})
df['Polyphagia'] = df['Polyphagia'].map({'Yes':1,'No':0})
df['Genital thrush'] = df['Genital thrush'].map({'Yes':1,'No':0})
df['visual blurring'] = df['visual blurring'].map({'Yes':1,'No':0})
df['Itching'] = df['Itching'].map({'Yes':1,'No':0})
df['Irritability'] = df['Irritability'].map({'Yes':1,'No':0})
df['delayed healing'] = df['delayed healing'].map({'Yes':1,'No':0})
df['partial paresis'] = df['partial paresis'].map({'Yes':1,'No':0})
df['muscle stiffness'] = df['muscle stiffness'].map({'Yes':1,'No':0})
df['Alopecia'] = df['Alopecia'].map({'Yes':1,'No':0})
df['Obesity'] = df['Obesity'].map({'Yes':1,'No':0})
```

## EDA

In [11]:

```
plt.figure(figsize=(40,20))
df.hist()
plt.show()
```

<Figure size 2880x1440 with 0 Axes>



In [12]:

```
df.corr()
```

Out[12]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush
Age	1.000000	0.062872	0.199781	0.137382	0.064808	0.224596	0.315577	0.096519
Gender	0.062872	1.000000	-0.268894	-0.312262	-0.281840	-0.124490	-0.219968	0.208961
Polyuria	0.199781	-0.268894	1.000000	0.598609	0.447207	0.263000	0.373873	0.087273
Polydipsia	0.137382	-0.312262	0.598609	1.000000	0.405965	0.332453	0.316839	0.028081
sudden weight loss	0.064808	-0.281840	0.447207	0.405965	1.000000	0.282884	0.243511	0.089858
weakness	0.224596	-0.124490	0.263000	0.332453	0.282884	1.000000	0.180266	0.027780
Polyphagia	0.315577	-0.219968	0.373873	0.316839	0.243511	0.180266	1.000000	-0.063712
Genital thrush	0.096519	0.208961	0.087273	0.028081	0.089858	0.027780	-0.063712	1.000000
visual blurring	0.402729	-0.208092	0.235095	0.331250	0.068754	0.301043	0.293545	-0.144390
Itching	0.296559	-0.052496	0.088289	0.128716	-0.004516	0.309440	0.144390	0.125667
Irritability	0.201625	-0.013735	0.237740	0.203446	0.140340	0.146698	0.239466	0.160340
delayed healing	0.257501	-0.101978	0.149873	0.115691	0.088140	0.335507	0.263980	0.130498
partial paresis	0.232742	-0.332288	0.441664	0.442249	0.264014	0.272982	0.373569	-0.195498
muscle stiffness	0.307703	-0.090542	0.152938	0.180723	0.109756	0.263164	0.320031	-0.100340
Alopecia	0.321691	0.327871	-0.144192	-0.310964	-0.202727	0.090490	-0.053498	0.204498
Obesity	0.140458	-0.005396	0.126567	0.098691	0.169294	0.045665	0.029785	0.053498
class	0.108679	-0.449233	0.665922	0.648734	0.436568	0.243275	0.342504	0.110340

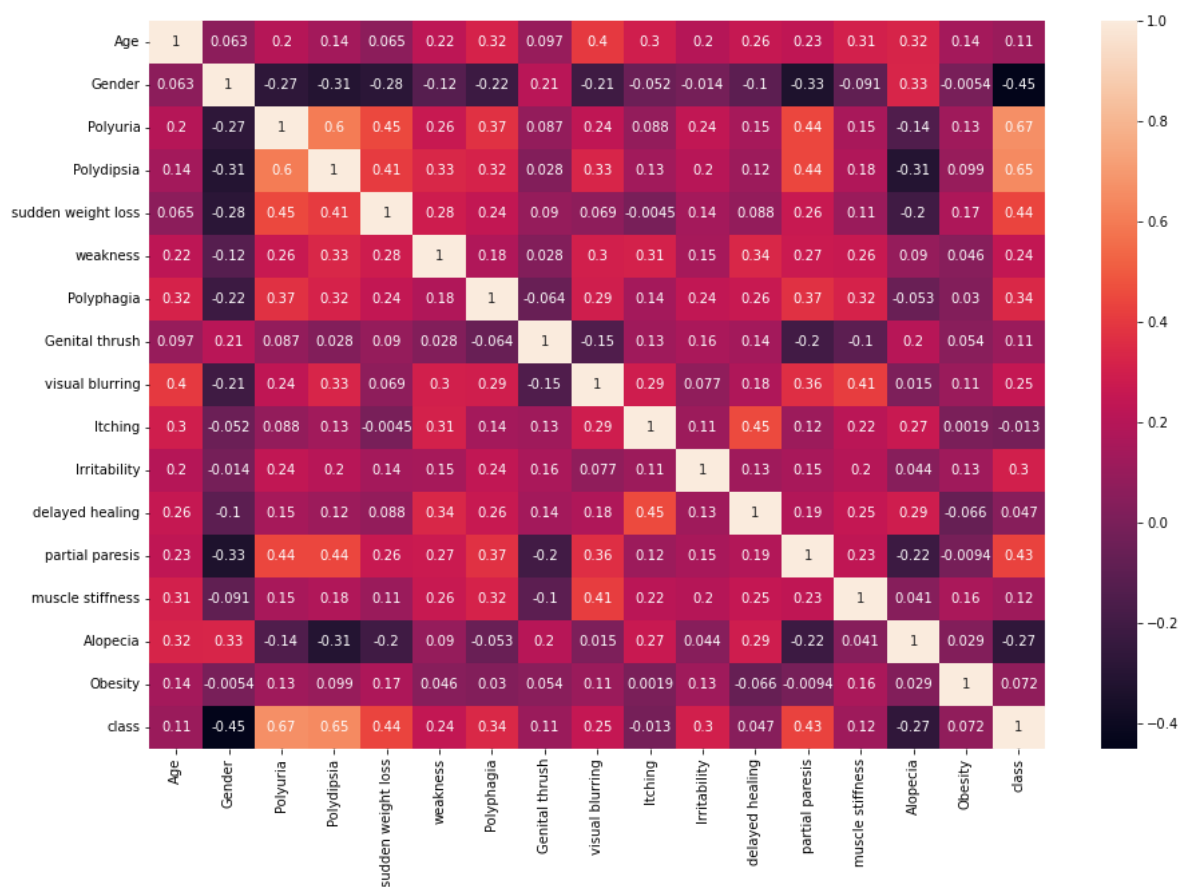


In [13]:

```
plt.subplots(figsize=(15,10))
sns.heatmap(df.corr(),annot=True)
```

Out[13]:

<AxesSubplot:>

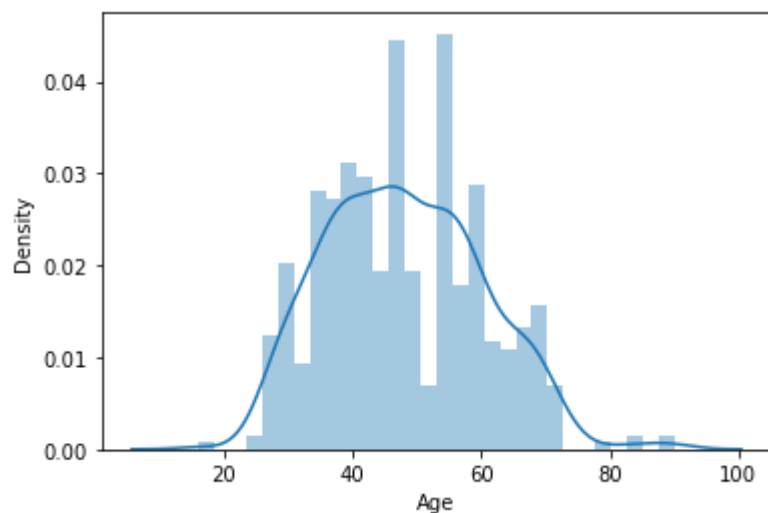


In [14]:

```
sns.distplot(df['Age'],bins=30)  
plt.show()
```

C:\Users\vinod\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

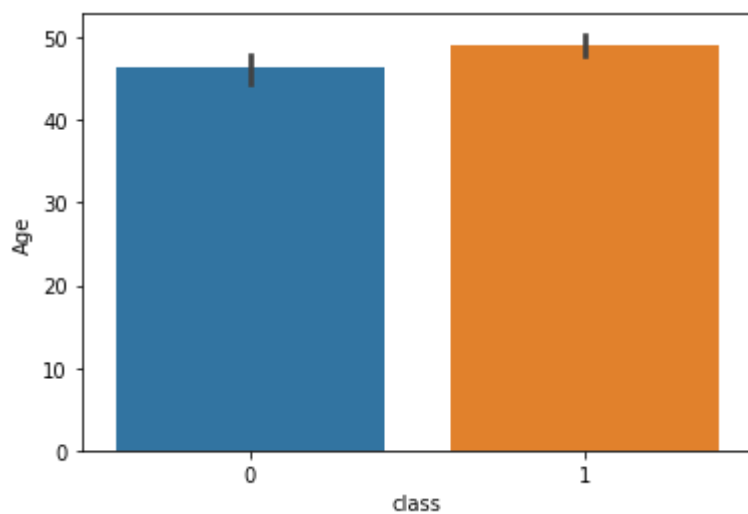


In [15]:

```
sns.barplot(x='class',y='Age',data=df)
```

Out[15]:

<AxesSubplot:xlabel='class', ylabel='Age'>



In [16]:

```
plt.pie(x=df['class'].value_counts())  
plt.show
```

Out[16]:

<function matplotlib.pyplot.show(close=None, block=None)>

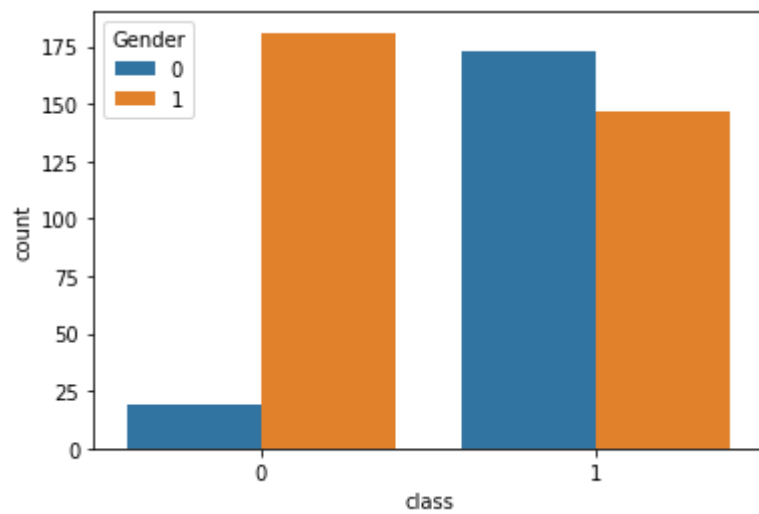


In [17]:

```
sns.countplot(x='class', data=df, hue='Gender')
```

Out[17]:

<AxesSubplot:xlabel='class', ylabel='count'>

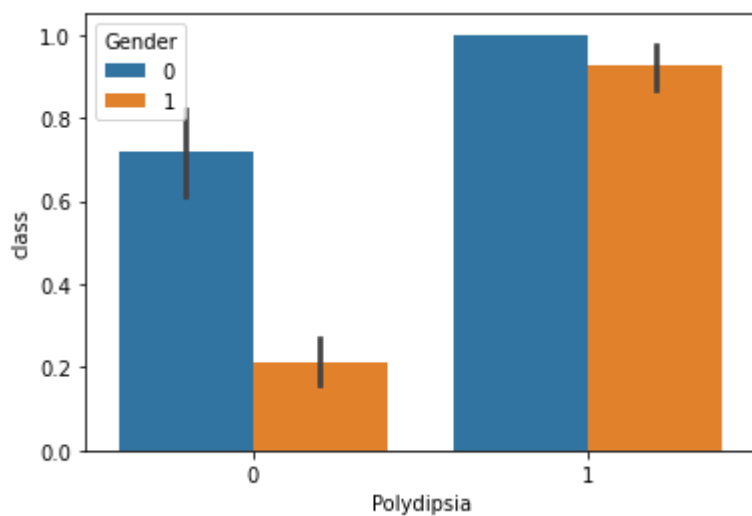


In [18]:

```
sns.barplot(data=df,x='Polydipsia',y='class',hue='Gender')
```

Out[18]:

<AxesSubplot:xlabel='Polydipsia', ylabel='class'>

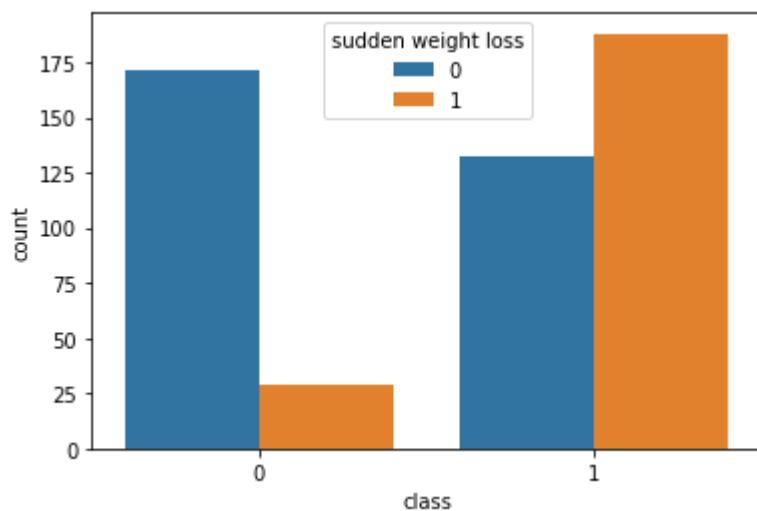


In [19]:

```
sns.countplot(x='class',data=df,hue='sudden weight loss')
```

Out[19]:

<AxesSubplot:xlabel='class', ylabel='count'>

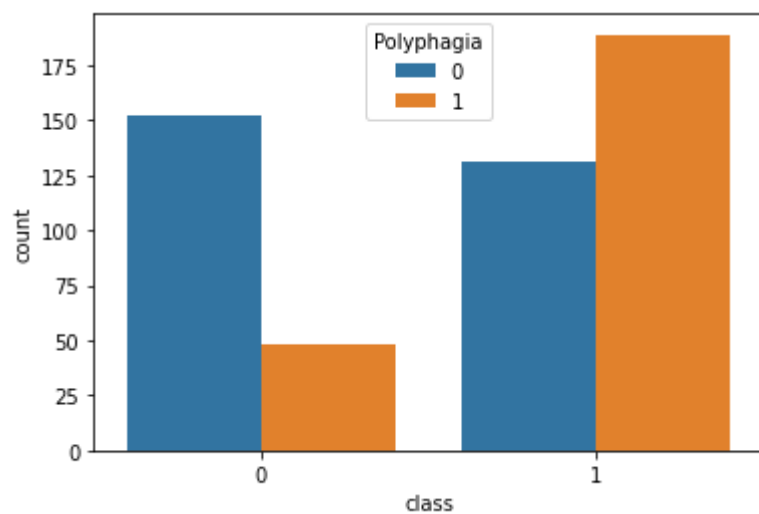


In [20]:

```
sns.countplot(x='class',data=df, hue='Polyphagia')
```

Out[20]:

<AxesSubplot:xlabel='class', ylabel='count'>

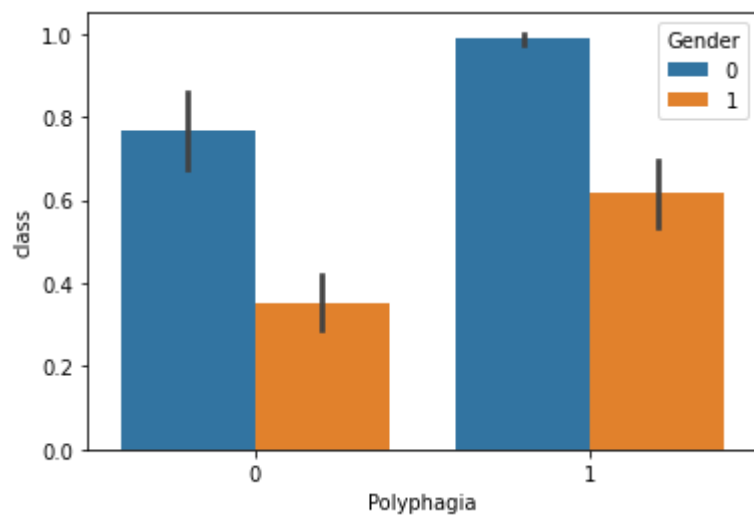


In [21]:

```
sns.barplot(x='Polyphagia',y='class',data=df,hue="Gender")
```

Out[21]:

<AxesSubplot:xlabel='Polyphagia', ylabel='class'>

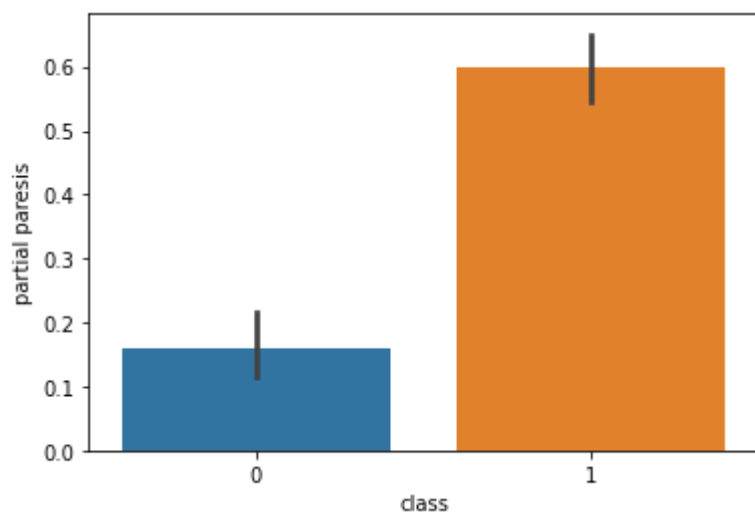


In [22]:

```
sns.barplot(x='class',y='partial paresis',data=df)
```

Out[22]:

<AxesSubplot:xlabel='class', ylabel='partial paresis'>

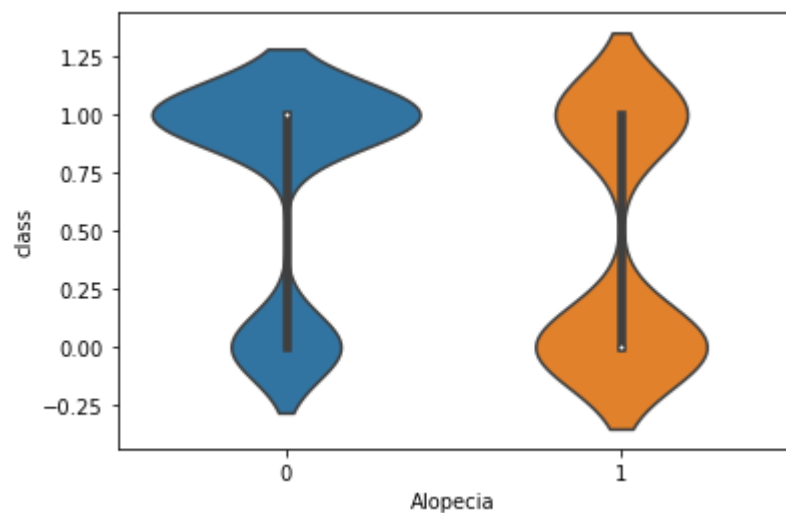


In [23]:

```
sns.violinplot(x='Alopecia',y='class',data=df)
```

Out[23]:

<AxesSubplot:xlabel='Alopecia', ylabel='class'>

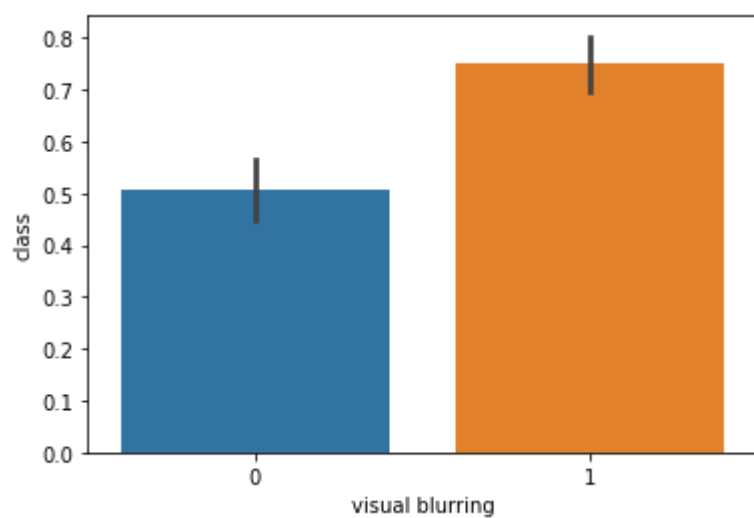


In [24]:

```
sns.barplot(x="visual blurring", y="class", data=df)
```

Out[24]:

<AxesSubplot:xlabel='visual blurring', ylabel='class'>

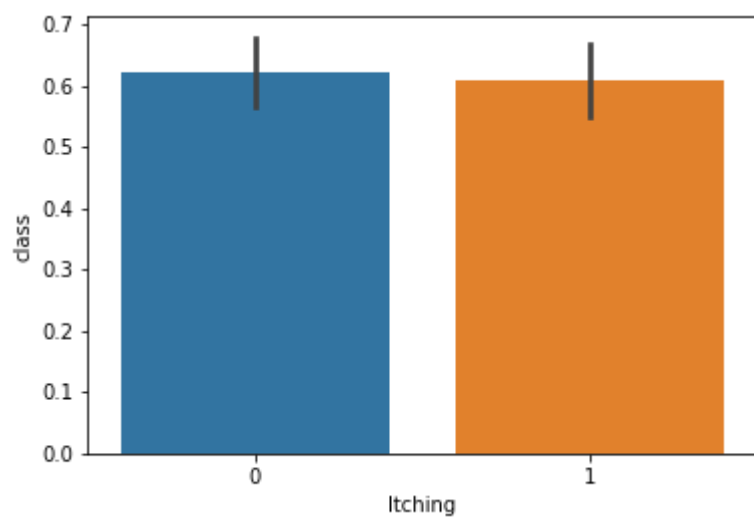


In [25]:

```
sns.barplot(x="Itching", y="class", data=df)
```

Out[25]:

<AxesSubplot:xlabel='Itching', ylabel='class'>

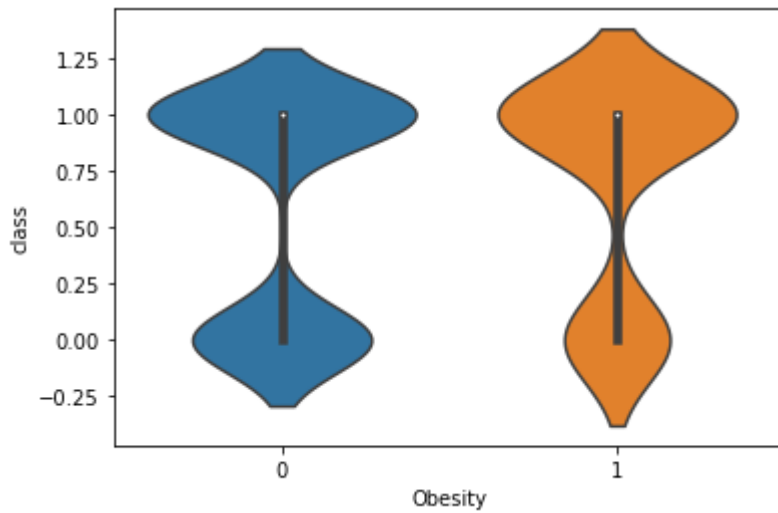


In [26]:

```
sns.violinplot(x='Obesity',y='class',data=df)
```

Out[26]:

<AxesSubplot:xlabel='Obesity', ylabel='class'>

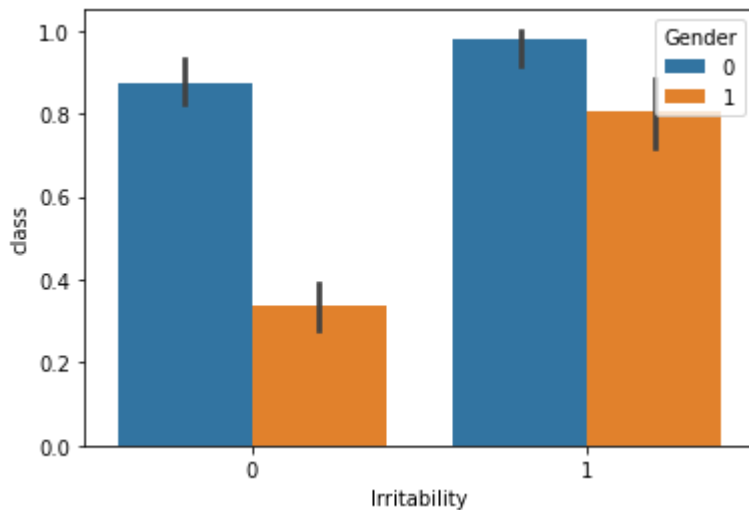


In [27]:

```
sns.barplot(x='Irritability',y='class',data=df,hue='Gender')
```

Out[27]:

<AxesSubplot:xlabel='Irritability', ylabel='class'>



## SPLITTING DATA AND PREPROCESSING

In [28]:

```
X = df.drop('class',axis=1)  
y = df['class']
```

In [29]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state=0)
```

In [30]:

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

## LOGISTIC REGRESSION

In [31]:

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

Out[31]:

LogisticRegression()

In [32]:

```
pre=lr.predict(X_test)
```

In [33]:

```
logistic_regression=accuracy_score(pre,y_test)
print(accuracy_score(pre,y_test))
print(confusion_matrix(pre,y_test))
```

0.9519230769230769

```
[[37  2]
 [ 3 62]]
```

In [34]:

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

	precision	recall	f1-score	support
0	0.93	0.95	0.94	39
1	0.97	0.95	0.96	65
accuracy			0.95	104
macro avg	0.95	0.95	0.95	104
weighted avg	0.95	0.95	0.95	104

## KNN

In [35]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)
```

In [36]:

```
knn.fit(X_train,y_train)
```

Out[36]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [37]:

```
y_pred_knn=knn.predict(X_test)
```

In [38]:

```
print(classification_report(y_test,y_pred_knn))
```

	precision	recall	f1-score	support
0	0.95	1.00	0.98	40
1	1.00	0.97	0.98	64
accuracy			0.98	104
macro avg	0.98	0.98	0.98	104
weighted avg	0.98	0.98	0.98	104

In [39]:

```
from sklearn.model_selection import cross_val_score
accuracy_rate = []

for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    score = cross_val_score(knn, df, df['class'],cv=10)
    accuracy_rate.append(score.mean())
score
```

Out[39]:

```
array([0.73076923, 0.80769231, 0.82692308, 0.78846154, 0.73076923,
       0.80769231, 0.82692308, 0.84615385, 0.86538462, 0.76923077])
```

In [40]:

```
accuracy_rate
```

Out[40]:

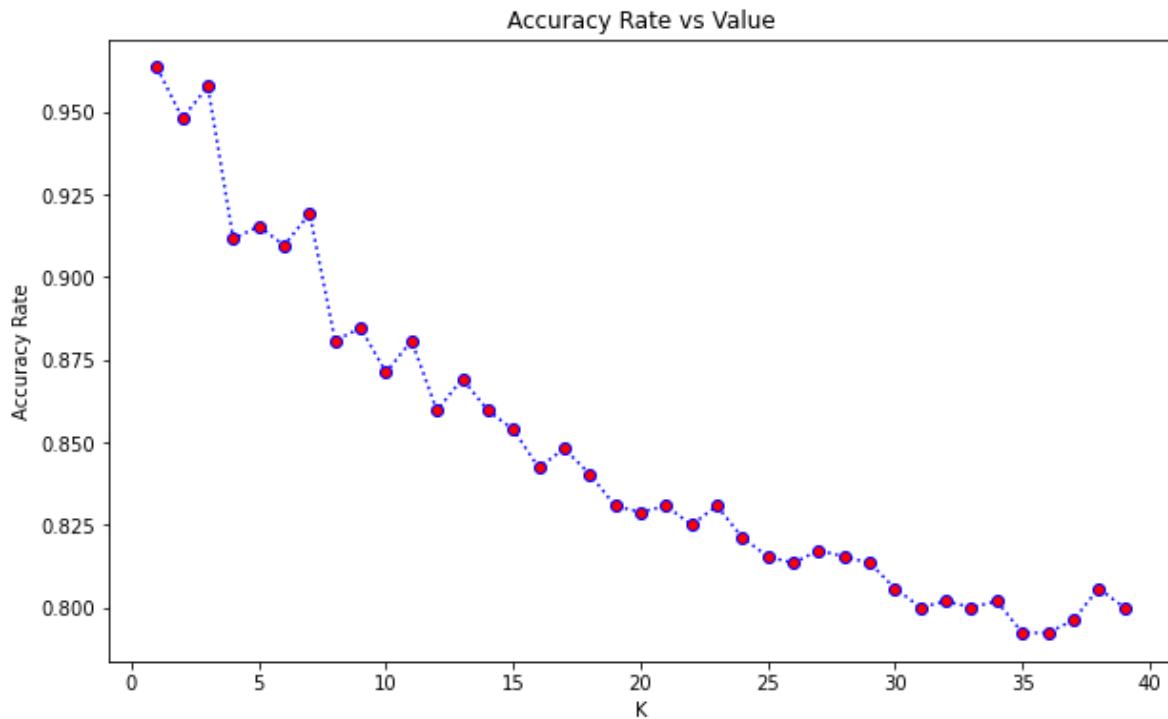
```
[0.9634615384615384,  
 0.948076923076923,  
 0.9576923076923076,  
 0.9115384615384613,  
 0.9153846153846154,  
 0.9096153846153847,  
 0.9192307692307692,  
 0.8807692307692309,  
 0.8846153846153847,  
 0.8711538461538462,  
 0.8807692307692309,  
 0.8596153846153847,  
 0.8692307692307694,  
 0.8596153846153844,  
 0.8538461538461538,  
 0.8423076923076923,  
 0.848076923076923,  
 0.8403846153846153,  
 0.8307692307692307,  
 0.8288461538461538,  
 0.8307692307692307,  
 0.825,  
 0.8307692307692307,  
 0.8211538461538461,  
 0.8153846153846154,  
 0.8134615384615385,  
 0.8173076923076923,  
 0.8153846153846154,  
 0.8134615384615385,  
 0.8057692307692308,  
 0.8,  
 0.8019230769230768,  
 0.7999999999999999,  
 0.801923076923077,  
 0.7923076923076923,  
 0.7923076923076923,  
 0.7961538461538462,  
 0.8057692307692307,  
 0.7999999999999999]
```

In [41]:

```
plt.figure(figsize=(10,6))

plt.plot(range(1,40), accuracy_rate, color='blue', linestyle='dotted', marker='o', markerfa

plt.title('Accuracy Rate vs Value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
plt.show()
```



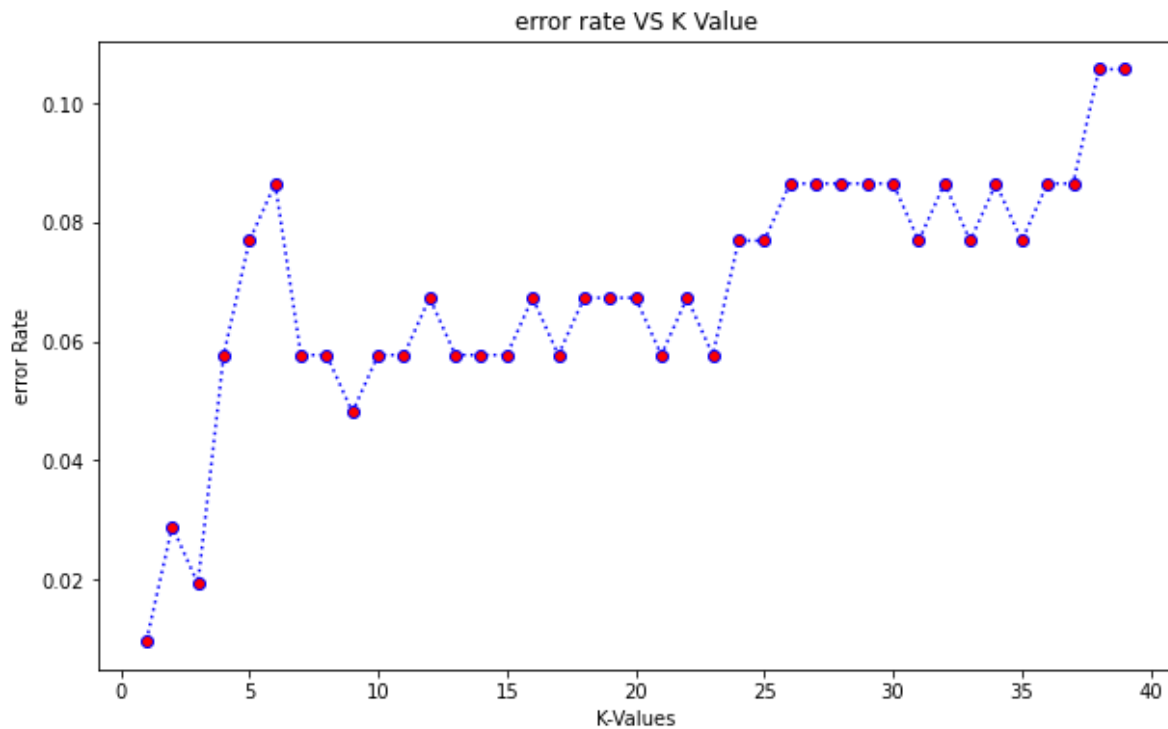
In [42]:

```
error_rate = []

for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i!=y_test))
```

In [43]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40), error_rate, color='blue', linestyle='dotted', marker='o', markerfacecolor='red')
plt.title('error rate VS K Value')
plt.xlabel('K-Values')
plt.ylabel('error Rate')
plt.show()
```



## DECISION TREE

In [44]:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(max_depth=5)
```

In [45]:

```
dtc.fit(X_train,y_train)
```

Out[45]:

```
DecisionTreeClassifier(max_depth=5)
```

In [46]:

```
y_pred_dtc=dtc.predict(X_test)
```

In [47]:

```
print(classification_report(y_test,y_pred_dtc))
```

	precision	recall	f1-score	support
0	0.93	0.97	0.95	40
1	0.98	0.95	0.97	64
accuracy			0.96	104
macro avg	0.96	0.96	0.96	104
weighted avg	0.96	0.96	0.96	104

In [63]:

```
from matplotlib import pyplot as plt  
from sklearn import tree
```

In [64]:

```
text_representation = tree.export_text(dtc)
print(text_representation)
```

```

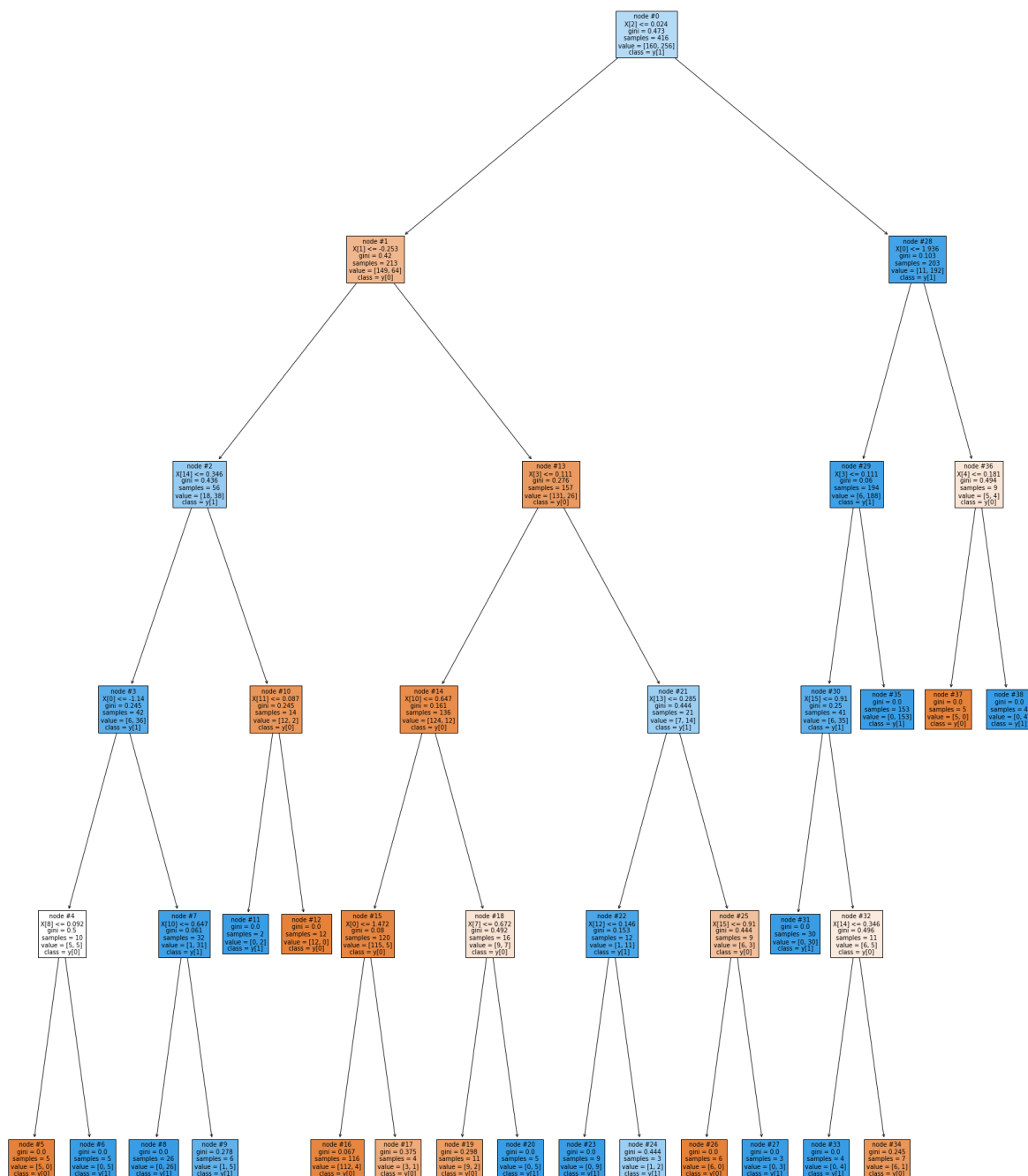
--- feature_2 <= 0.02
|--- feature_1 <= -0.25
|   |--- feature_14 <= 0.35
|   |   |--- feature_0 <= -1.14
|   |   |   |--- feature_8 <= 0.09
|   |   |   |   |--- class: 0
|   |   |   |   |--- feature_8 > 0.09
|   |   |   |   |   |--- class: 1
|   |   |--- feature_0 > -1.14
|   |   |   |--- feature_10 <= 0.65
|   |   |   |   |--- class: 1
|   |   |   |   |--- feature_10 > 0.65
|   |   |   |       |--- class: 1
|   |--- feature_14 > 0.35
|   |   |--- feature_11 <= 0.09
|   |   |   |--- class: 1
|   |   |   |--- feature_11 > 0.09
|   |   |       |--- class: 0
|--- feature_1 > -0.25
|   |--- feature_3 <= 0.11
|   |   |--- feature_10 <= 0.65
|   |   |   |--- feature_0 <= 1.47
|   |   |   |   |--- class: 0
|   |   |   |   |--- feature_0 > 1.47
|   |   |   |       |--- class: 0
|   |   |--- feature_10 > 0.65
|   |   |   |--- feature_7 <= 0.67
|   |   |   |   |--- class: 0
|   |   |   |   |--- feature_7 > 0.67
|   |   |   |       |--- class: 1
|   |--- feature_3 > 0.11
|   |   |--- feature_13 <= 0.28
|   |   |   |--- feature_12 <= 0.15
|   |   |   |   |--- class: 1
|   |   |   |   |--- feature_12 > 0.15
|   |   |   |       |--- class: 1
|   |   |--- feature_13 > 0.28
|   |   |   |--- feature_15 <= 0.91
|   |   |   |   |--- class: 0
|   |   |   |   |--- feature_15 > 0.91
|   |   |       |--- class: 1
--- feature_2 > 0.02
|--- feature_0 <= 1.94
|   |--- feature_3 <= 0.11
|   |   |--- feature_15 <= 0.91
|   |   |   |--- class: 1
|   |   |   |--- feature_15 > 0.91
|   |   |       |--- feature_14 <= 0.35
|   |   |       |   |--- class: 1
|   |   |       |   |--- feature_14 > 0.35
|   |   |       |       |--- class: 0
|   |--- feature_3 > 0.11
|   |   |--- class: 1
--- feature_0 > 1.94
|--- feature_4 <= 0.18
|   |--- class: 0

```

```
| | | |--- feature_4 > 0.18
| | | |--- class: 1
```

In [65]:

```
fig = plt.figure(figsize=(30,40))
tree.plot_tree(dtc,filled=True,class_names=True,node_ids=True)
plt.show()
```



# RANDOM FOREST

In [48]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

Out[48]:

```
RandomForestClassifier()
```

In [49]:

```
y_pred_rc=rfc.predict(X_test)
```

In [50]:

```
print(classification_report(y_test,y_pred_rc))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	40
1	1.00	0.98	0.99	64
accuracy			0.99	104
macro avg	0.99	0.99	0.99	104
weighted avg	0.99	0.99	0.99	104

In [51]:

```
accuracy_score(y_test,y_pred_rc)
```

Out[51]:

```
0.9903846153846154
```

In [ ]: