In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
```

In [2]:
```
# Load the dataset with raw string
file_path = r"C:\Users\Praveen T\Downloads\bank+marketing\bank\bank.csv"
data = pd.read_csv(file_path, delimiter=';')
data.rename(columns={'y': 'deposit'}, inplace=True)
```

In [3]:
```
data
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may
...	...	...	...	...	...	...	...	...	...	...	...
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	jul
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9	may
4518	57	technician	married	secondary	no	295	no	no	cellular	19	aug
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6	feb
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3	apr

4521 rows × 17 columns

In [4]:
```
data.describe()
```

Out[4]:

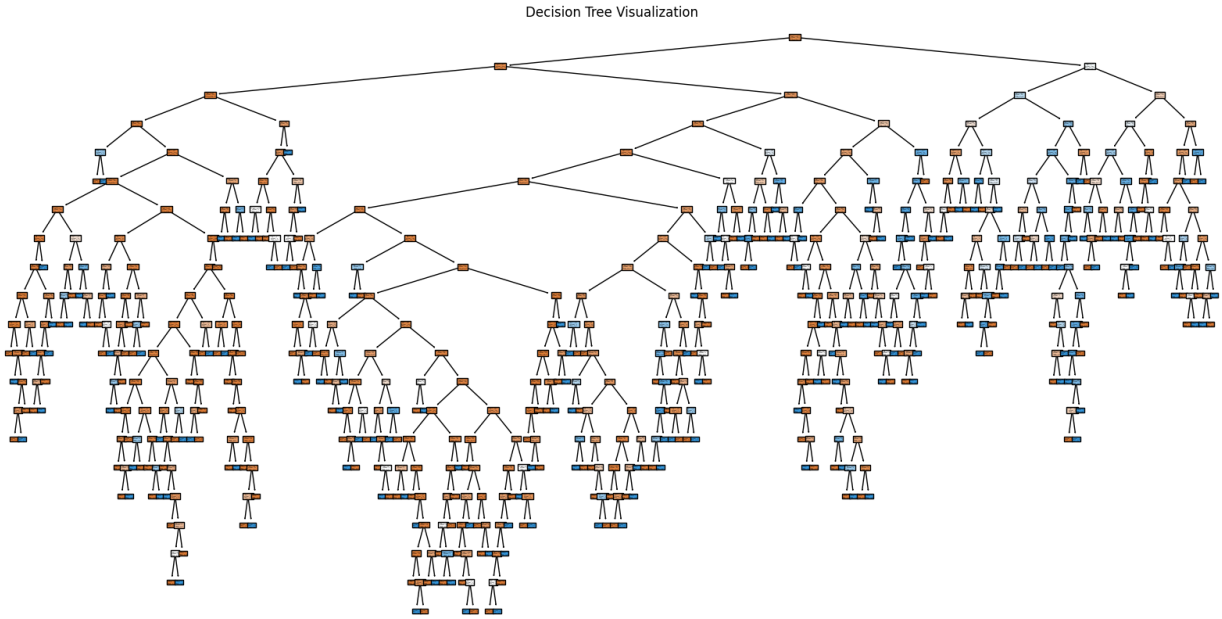
	age	balance	day	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.542579
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.693562
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.000000
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.000000
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.000000
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.000000
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.000000

In [5]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# Encode categorical variables
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le

# Define features and target
X = data.drop(columns='deposit')
y = data['deposit']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
```

In [6]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score

# Initialize and train the model
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

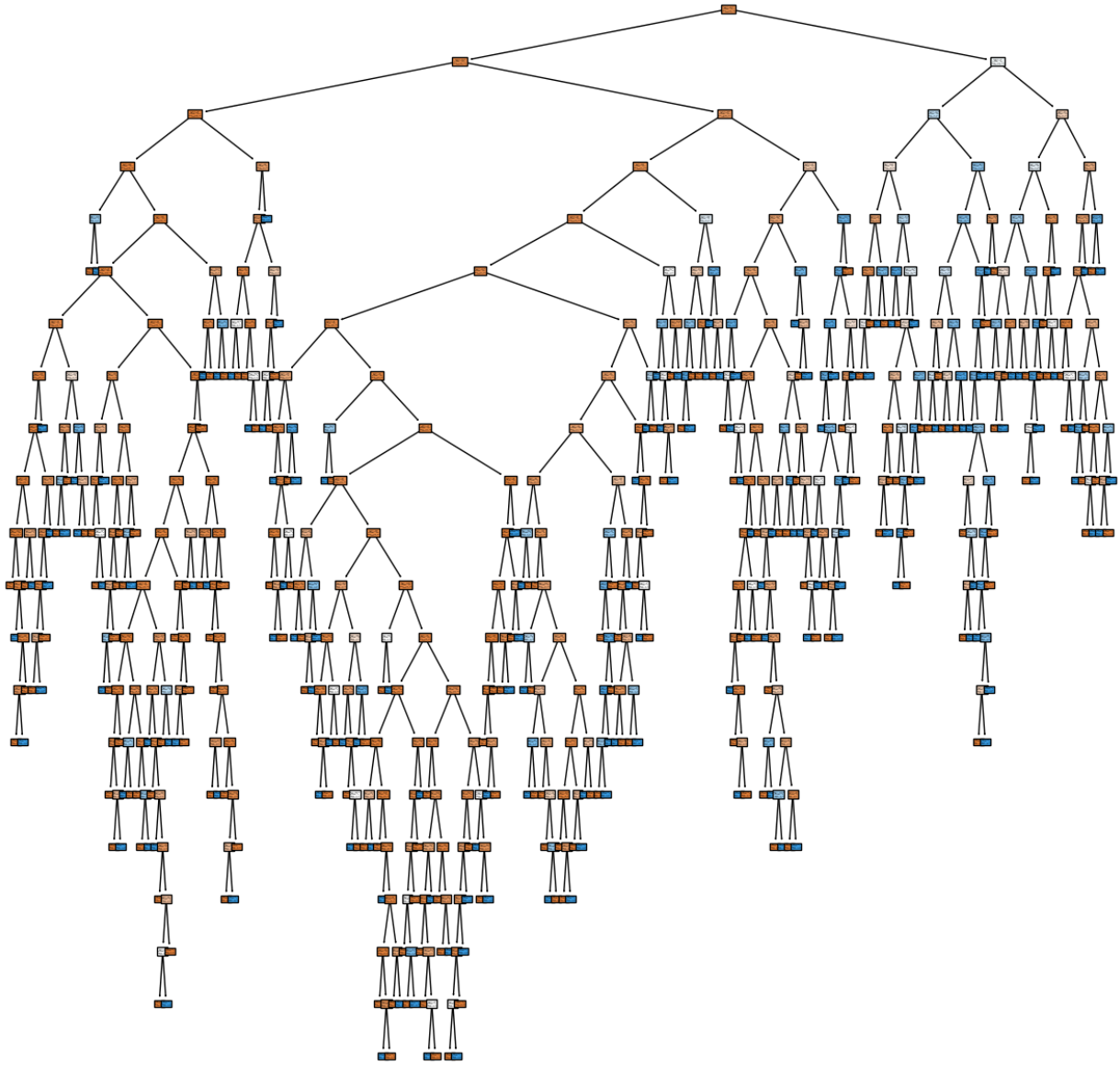
```
Accuracy Score: 0.8718232044198895

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.93       807
           1       0.42      0.47      0.44        98

    accuracy                           0.87       905
   macro avg       0.68      0.70      0.68       905
weighted avg       0.88      0.87      0.88       905
```

In [13]:
```
from sklearn import tree
import matplotlib.pyplot as plt

# Convert feature names to a list
feature_names = list(X.columns)

# Visualize the Decision Tree
plt.figure(figsize=(20,10))
tree.plot_tree(clf, filled=True, feature_names=feature_names, class_names=['No', 'Yes
plt.title('Decision Tree Visualization')
plt.show()
```



In [8]:
```
from sklearn.tree import plot_tree
cn = ['no','yes']
fn = X.columns
print(fn)
print(cn)
plt.figure(figsize=(15,15))
plot_tree(clf,class_names=cn,filled=True)
plt.show()
```

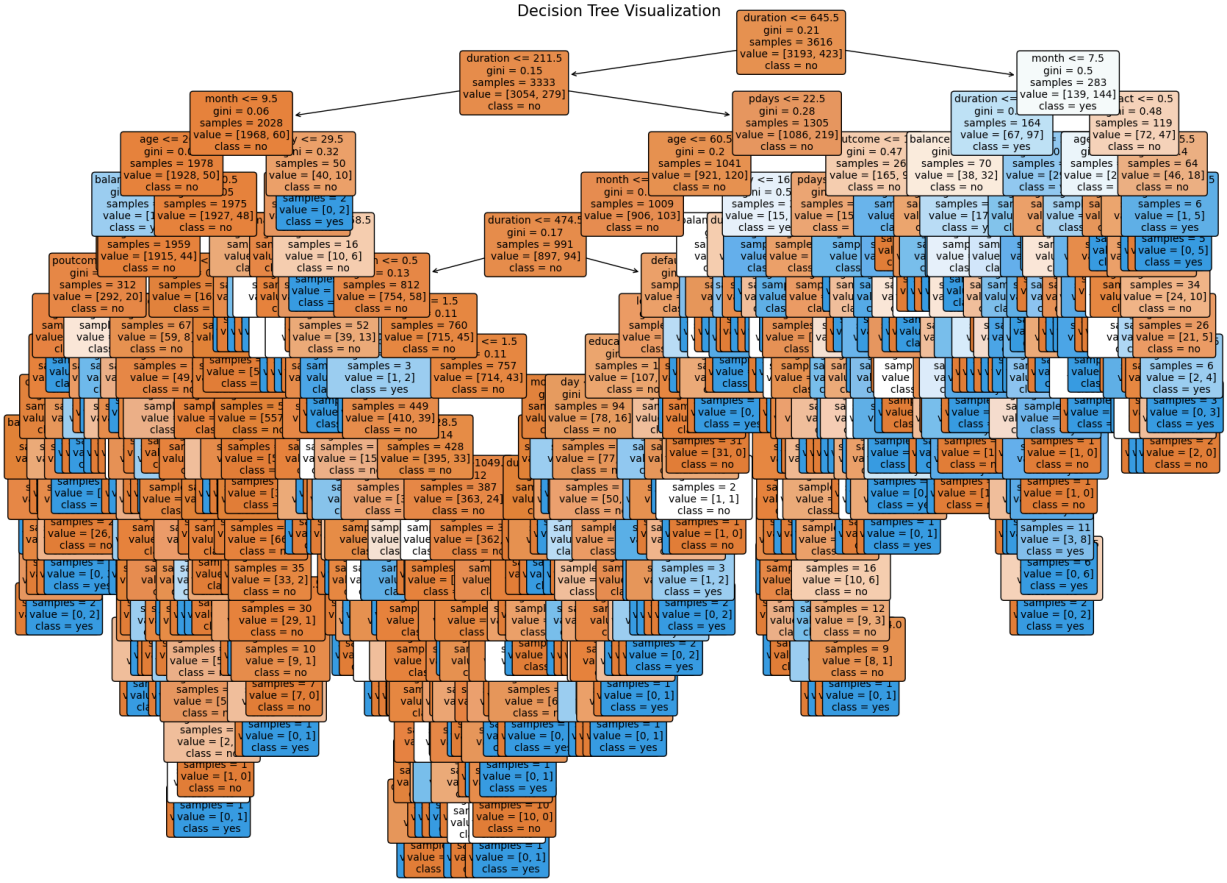
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome'],
      dtype='object')
['no', 'yes']
```



In [10]:

```
fn = X.columns.tolist()
#enhanced Decision tree
# Class names
cn = ['no', 'yes']

# Print feature names and class names for verification
print("Feature Names:", fn)
print("Class Names:", cn)

# Plot the decision tree
plt.figure(figsize=(20, 15))
plot_tree(clf, feature_names=fn, class_names=cn, filled=True, rounded=True, proportio
plt.title("Decision Tree Visualization", fontsize=15)
plt.show()
```

```
Feature Names: ['age', 'job', 'marital', 'education', 'default', 'balance', 'housin
g', 'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays', 'previous',
'poutcome']
Class Names: ['no', 'yes']
```


Decision Tree Visualization
duration <= 645.5
gini = 0.21
samples = 3616
value = [3193, 423]
class = no
duration <= 211.5
gini = 0.15
samples = 3333
value = [3054, 279]
class = no
pdays <= 22.5
gini = 0.28
samples = 1305
value = [1086, 219]
class = no
month <= 7.5
gini = 0.5
samples = 283
value = [139, 144]
class = yes
month <= 9.5
gini = 0.06
samples = 2028
value = [1968, 60]
class = no
duration <= 474.5
gini = 0.17
samples = 991
value = [897, 94]
class = no
duration <=
gini = 0.
samples = 164
value = [67, 97]
class = yes
samples = 119
value = [72, 47]
class = no
samples = 64
value = [46, 18]
class = no
samples = 6
value = [1, 5]
class = yes
samples = 5
value = [0, 5]
class = yes
samples = 34
value = [24, 10]
class = no
samples = 26
value = [21, 5]
class = no
samples = 6
value = [2, 4]
class = yes
samples = 3
value = [0, 3]
class = yes
samples = 2
value = [2, 0]
class = no

In [ ]: