In [3]:
```
import pandas as pd

# Load the dataset
file_path = 'C:/Users/sathi/Downloads/bank-additional-full.csv'  # Replace with the path to your dataset
data = pd.read_csv(file_path, delimiter=';')

# Display the first few rows of the dataset
data.head()
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0

5 rows × 21 columns

In [7]:
```
# Import necessary libraries for preprocessing
from sklearn.preprocessing import LabelEncoder

# Encode categorical variables
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    if column != 'y':  # Don't encode the target variable yet
        label_encoders[column] = LabelEncoder()
        data[column] = label_encoders[column].fit_transform(data[column])

# Encode the target variable
label_encoders['y'] = LabelEncoder()
data['y'] = label_encoders['y'].fit_transform(data['y'])

# Display the first few rows of the processed data
data.head()
```

Out[7]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome
0	56	3	1	0	0	0	0	1	6	1	...	1	999	0	1
1	57	7	1	3	1	0	0	1	6	1	...	1	999	0	1
2	37	7	1	3	0	2	0	1	6	1	...	1	999	0	1
3	40	0	1	1	0	0	0	1	6	1	...	1	999	0	1
4	56	7	1	3	0	0	2	1	6	1	...	1	999	0	1

5 rows × 21 columns

In [12]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Load the dataset
file_path = 'C:/Users/sathi/Downloads/bank-additional-full.csv'
data = pd.read_csv(file_path, delimiter=';')

# Encode categorical variables
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    if column != 'y':  # Don't encode the target variable yet
        label_encoders[column] = LabelEncoder()
        data[column] = label_encoders[column].fit_transform(data[column])

# Encode the target variable
label_encoders['y'] = LabelEncoder()
data['y'] = label_encoders['y'].fit_transform(data['y'])

# Split the data into features (X) and target (y)
X = data.drop('y', axis=1)
y = data['y']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

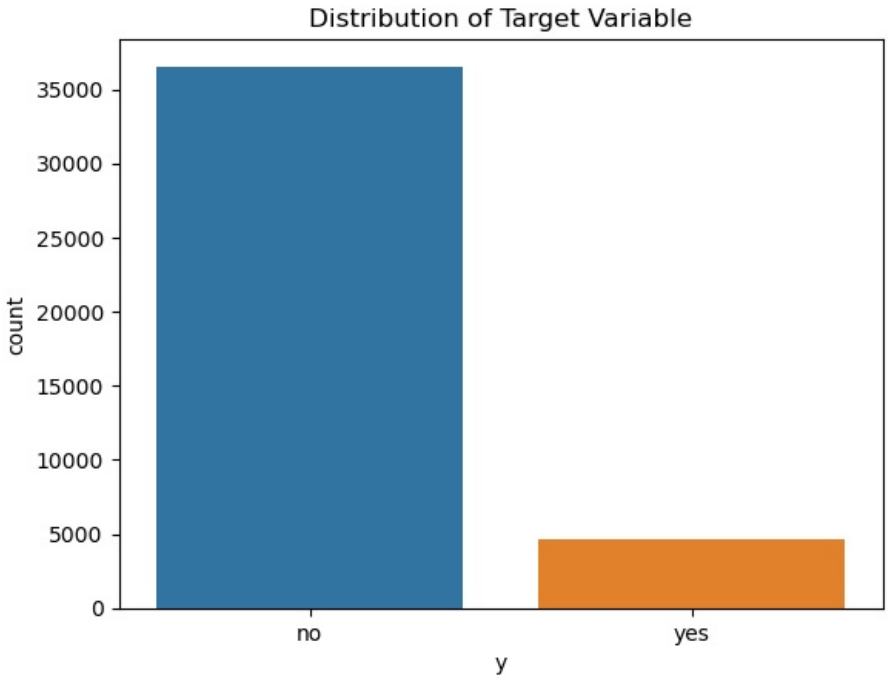
```
# Train the Decision Tree Classifier
classifier = DecisionTreeClassifier(random_state=42)
classifier.fit(X_train, y_train)

# Predict the test set results
y_pred = classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

```
Accuracy: 0.8892935178441369
Confusion Matrix:
 [[10275   693]
 [  675   714]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.94      0.94     10968
           1       0.51      0.51      0.51      1389

    accuracy                           0.89     12357
   macro avg       0.72      0.73      0.72     12357
weighted avg       0.89      0.89      0.89     12357
```

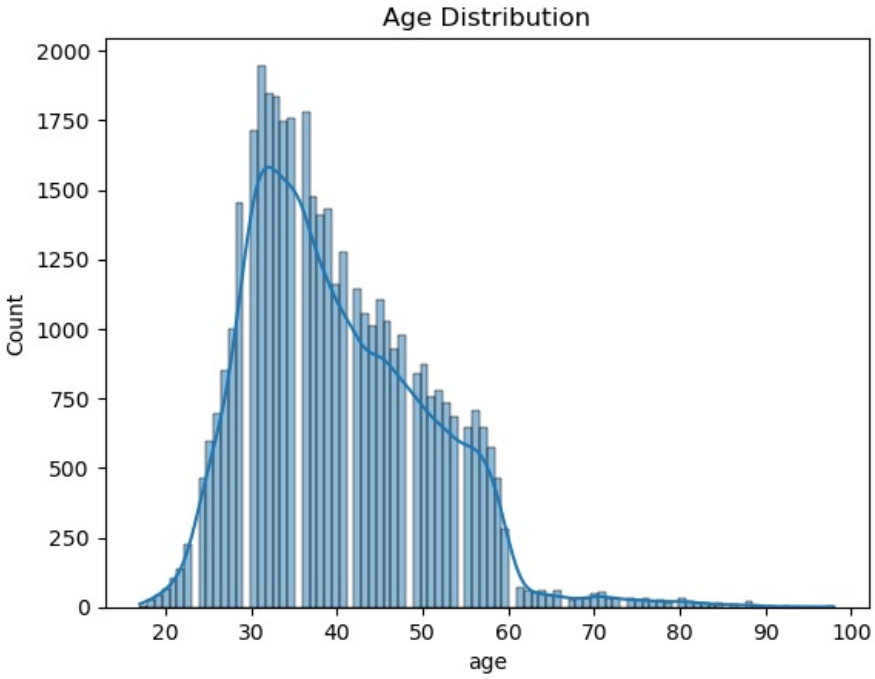
In [13]:

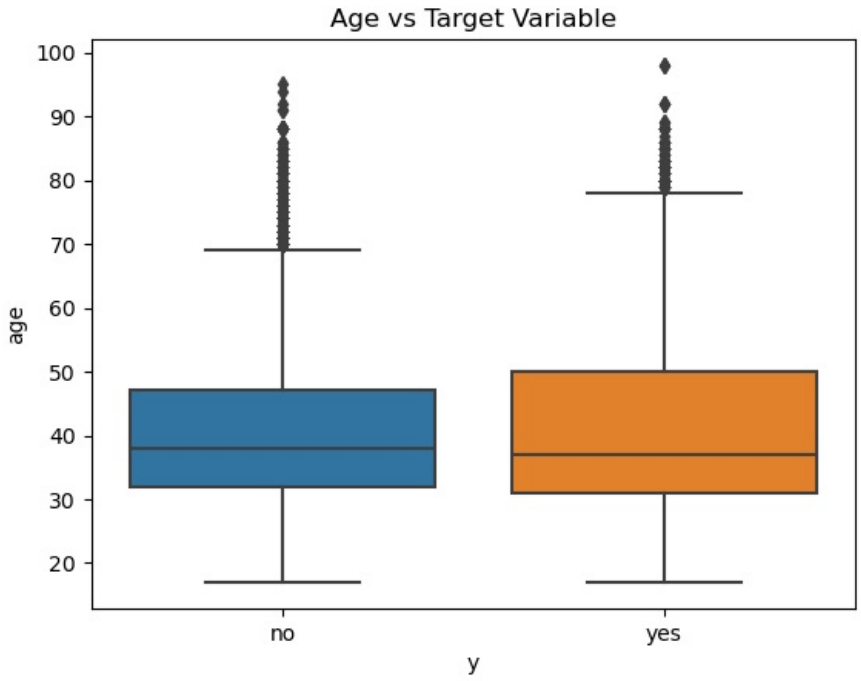
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
file_path = 'C:/Users/sathi/Downloads/bank-additional-full.csv'
data = pd.read_csv(file_path, delimiter=';')

# Display the first few rows of the dataset
print(data.head())

# Visualize the distribution of the target variable
sns.countplot(x='y', data=data)
plt.title('Distribution of Target Variable')
plt.show()

# Visualize the distribution of age
sns.histplot(data['age'], kde=True)
plt.title('Age Distribution')
plt.show()

# Visualize the relationship between age and the target variable
sns.boxplot(x='y', y='age', data=data)
plt.title('Age vs Target Variable')
plt.show()
```

```
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
3   40     admin.  married     basic.6y       no      no   no  telephone
4   56   services  married  high.school       no      no  yes  telephone

  month day_of_week  ...  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  ...         1    999         0  nonexistent          1.1
1   may         mon  ...         1    999         0  nonexistent          1.1
2   may         mon  ...         1    999         0  nonexistent          1.1
3   may         mon  ...         1    999         0  nonexistent          1.1
4   may         mon  ...         1    999         0  nonexistent          1.1

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```




```
C:\Software Installation\Python\Anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_n
a option is deprecated and will be removed in a future version. Convert inf values to NaN before operating inste
ad.
  with pd.option_context('mode.use_inf_as_na', True):
```



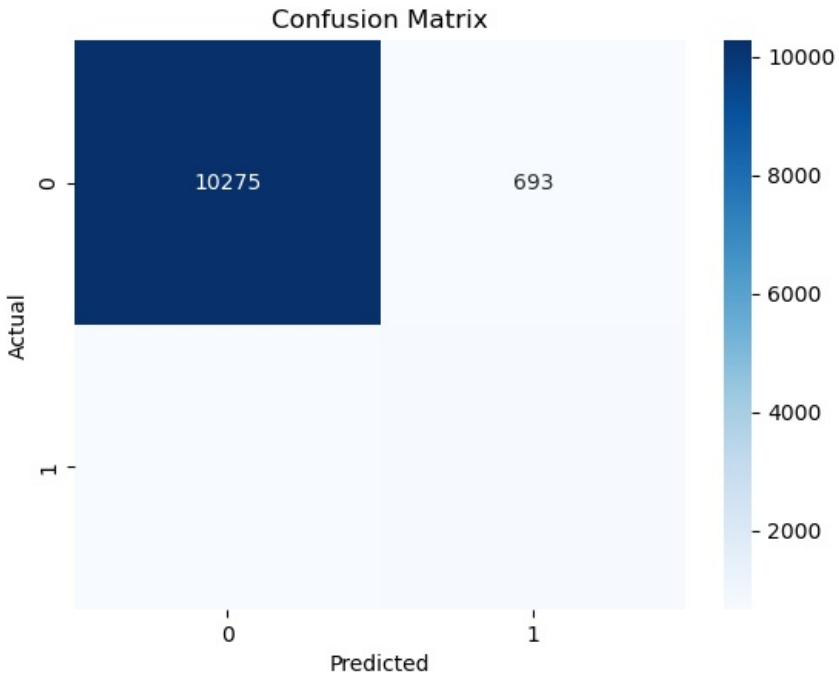


In [14]:
```python
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Encode categorical variables
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    if column != 'y':  # Don't encode the target variable yet
        label_encoders[column] = LabelEncoder()
        data[column] = label_encoders[column].fit_transform(data[column])

# Encode the target variable
label_encoders['y'] = LabelEncoder()
data['y'] = label_encoders['y'].fit_transform(data['y'])

# Split the data into features (X) and target (y)
X = data.drop('y', axis=1)
y = data['y']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train the Decision Tree Classifier
classifier = DecisionTreeClassifier(random_state=42)
classifier.fit(X_train, y_train)

# Predict the test set results
y_pred = classifier.predict(X_test)
```

In [15]:
```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

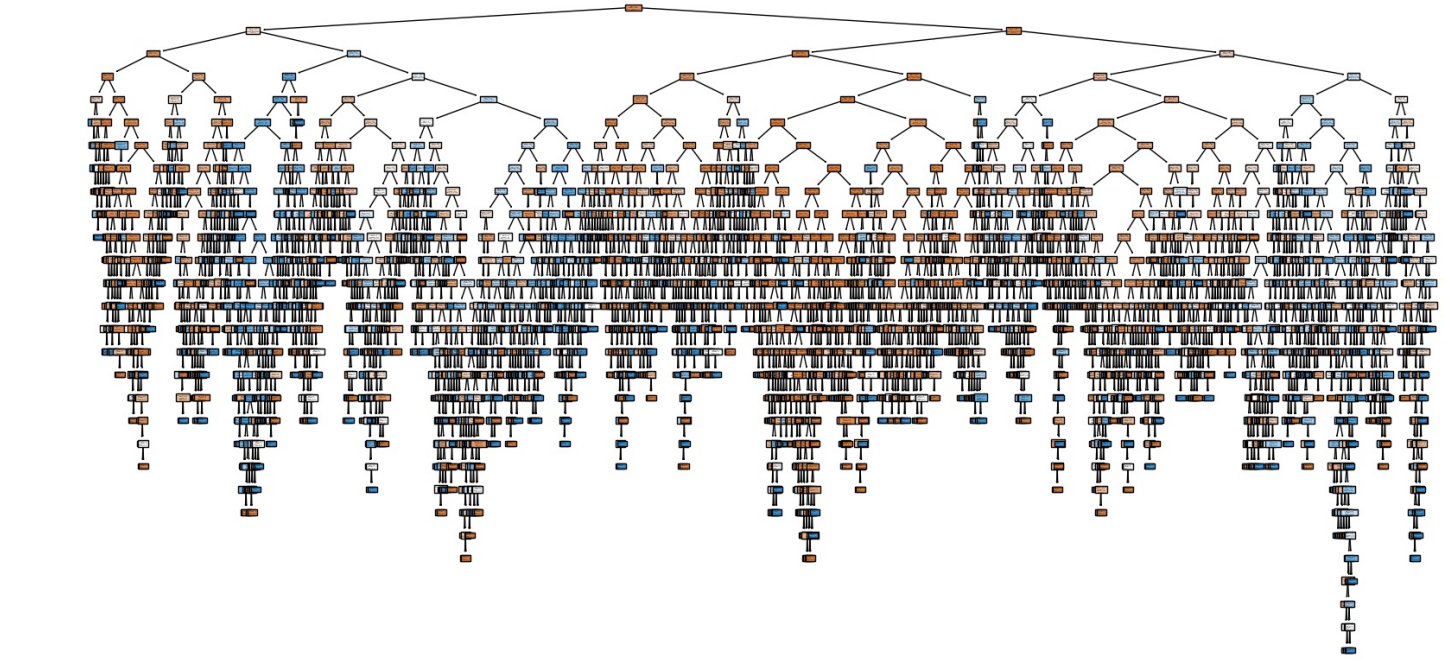
```
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)

# Visualize the confusion matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
Accuracy: 0.8892935178441369
Confusion Matrix:
 [[10275   693]
 [  675   714]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.94      0.94     10968
           1       0.51      0.51      0.51      1389

    accuracy                           0.89     12357
   macro avg       0.72      0.73      0.72     12357
weighted avg       0.89      0.89      0.89     12357
```



In [16]:

```
from sklearn.tree import plot_tree

plt.figure(figsize=(20, 10))
plot_tree(classifier, feature_names=X.columns, class_names=label_encoders['y'].classes_, filled=True)
plt.title('Decision Tree')
plt.show()
```



In [ ]: