Name: Maragathavalli C S

Prodigy Infotech

Data Science Intern

Task:3 Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioral data. Using the given dataset Bank Marketing

Load and Understand the Data

In [4]:
```
import pandas as pd

df=pd.read_csv(r"C:\Users\cskes\OneDrive\Desktop\bank-full.csv",sep=';')
print("Shape:",df.shape)
print(df.head())
print(df.info())
print(df['y'].value_counts())
```

```
Shape: (45211, 17)
   age           job  marital  education default  balance housing loan  \
0   58    management  married   tertiary      no     2143     yes   no
1   44    technician   single  secondary      no       29     yes   no
2   33  entrepreneur  married  secondary      no        2     yes  yes
3   47   blue-collar  married    unknown      no     1506     yes   no
4   33       unknown   single    unknown      no        1      no   no

   contact  day month  duration  campaign  pdays  previous poutcome   y
0  unknown    5   may       261         1     -1         0  unknown  no
1  unknown    5   may       151         1     -1         0  unknown  no
2  unknown    5   may        76         1     -1         0  unknown  no
3  unknown    5   may        92         1     -1         0  unknown  no
4  unknown    5   may       198         1     -1         0  unknown  no
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
None
y
no     39922
```

```
yes     5289
Name: count, dtype: int64
```

Preprocess the Data

In [7]:
```
from sklearn.preprocessing import LabelEncoder

df_encoded=df.copy()
label_encoders={}
```

Encode categorical columns

In [10]:
```
for col in df_encoded.select_dtypes(include='object').columns:
    le=LabelEncoder()
    df_encoded[col]=le.fit_transform(df_encoded[col])
    label_encoders[col]=le
```

Train_Test Split

In [14]:
```
from sklearn.model_selection import train_test_split

X=df_encoded.drop("y",axis=1)
y=df_encoded["y"]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,stratify=y,random_state=42)
```

Train the Decision Tree Classifier

In [17]:
```
from sklearn.tree import DecisionTreeClassifier

clf=DecisionTreeClassifier(random_state=42,class_weight='balanced')
clf.fit(X_train,y_train)
```

Out[17]:

▾ DecisionTreeClassifier

```
DecisionTreeClassifier(class_weight='balanced', random_state=42)
```

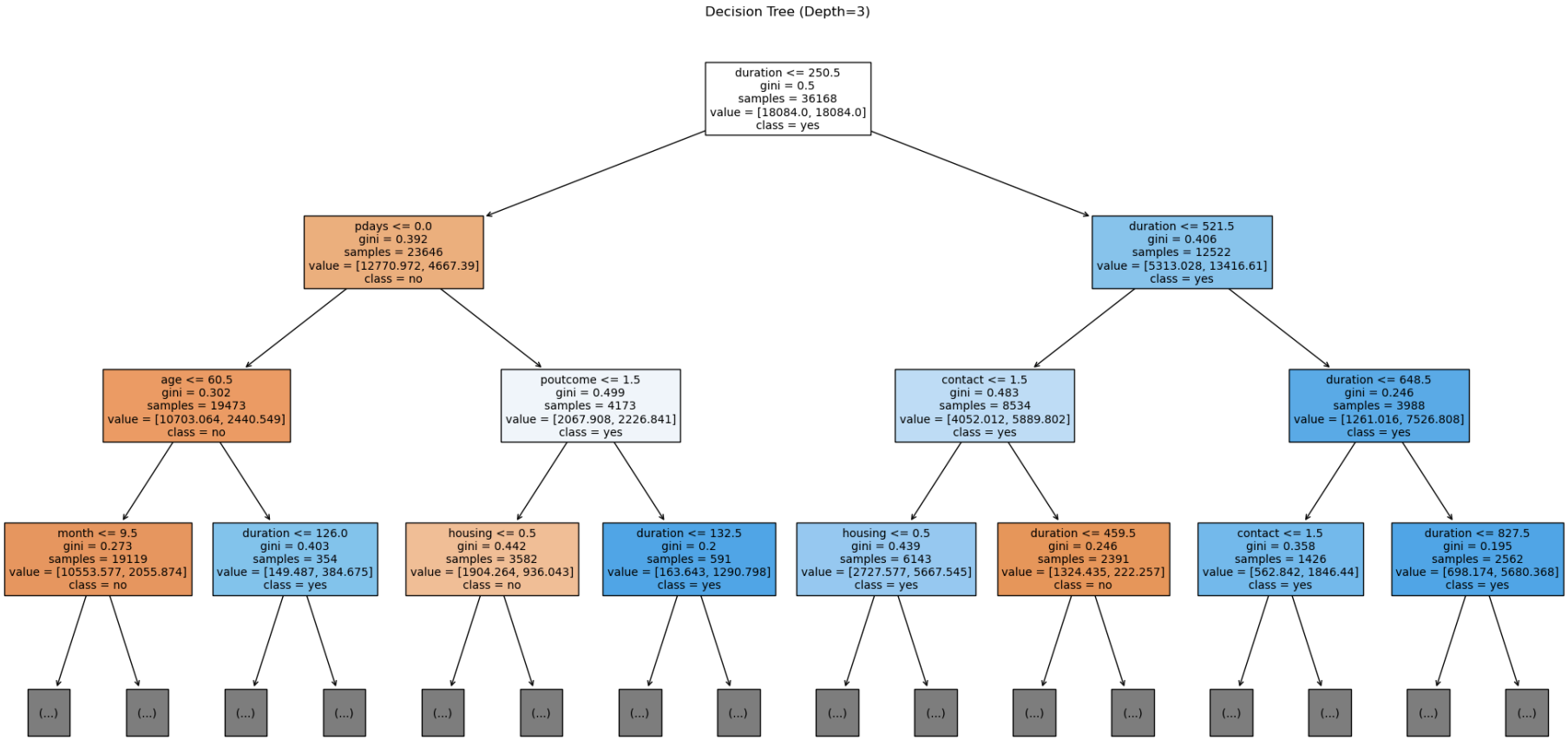
Evaluate the Model

In [20]:
```
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

y_pred=clf.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
report=classification_report(y_test,y_pred,target_names=label_encoders['y'].classes_)
conf_matrix=confusion_matrix(y_test,y_pred)

print(f"Accuracy:{accuracy:.4f}")
print("Classification Report:\n",report)
```

```
Accuracy:0.8759
Classification Report:
               precision    recall  f1-score   support

           no       0.92      0.94      0.93      7985
          yes       0.47      0.42      0.44      1058

    accuracy                           0.88      9043
   macro avg       0.70      0.68      0.69      9043
weighted avg       0.87      0.88      0.87      9043
```

Visualize the Decision Tree

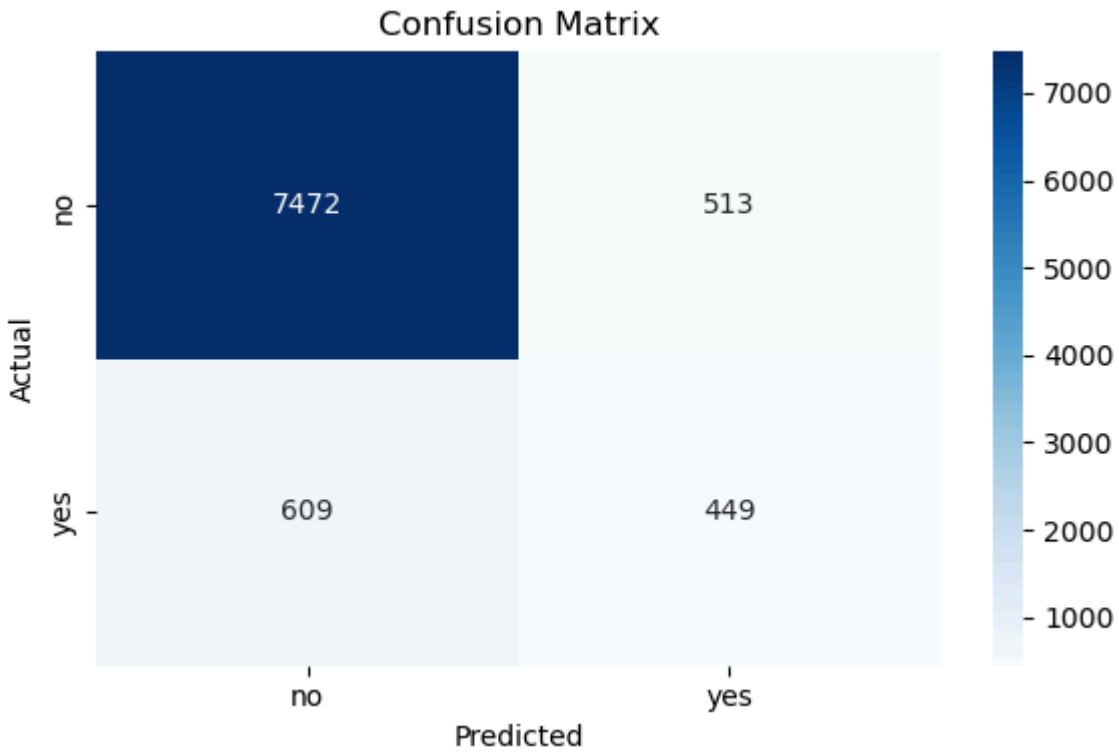
In [23]:
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(20,10))
plot_tree(clf,feature_names=X.columns,class_names=label_encoders['y'].classes_,
          filled=True,max_depth=3,fontsize=10)
plt.title("Decision Tree (Depth=3)")
plt.tight_layout()
plt.show()
```



Visualize the Confusion Matrix

In [26]:
```
import seaborn as sns

plt.figure(figsize=(6,4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=label_encoders['y'].classes_,
            yticklabels=label_encoders['y'].classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
```

```
plt.tight_layout()
plt.show()
```



In [28]:
```
# Step 9: Final Interpretation
# You can summarize your findings like this:

print("✅ Final Interpretation:")
print("- Model achieves high overall accuracy.")
print("- Class imbalance affects recall and precision for 'yes' class.")
print("- Decision Tree is interpretable; deeper trees or other models may improve performance.")
```

```
✅ Final Interpretation:
- Model achieves high overall accuracy.
- Class imbalance affects recall and precision for 'yes' class.
- Decision Tree is interpretable; deeper trees or other models may improve performance.
```