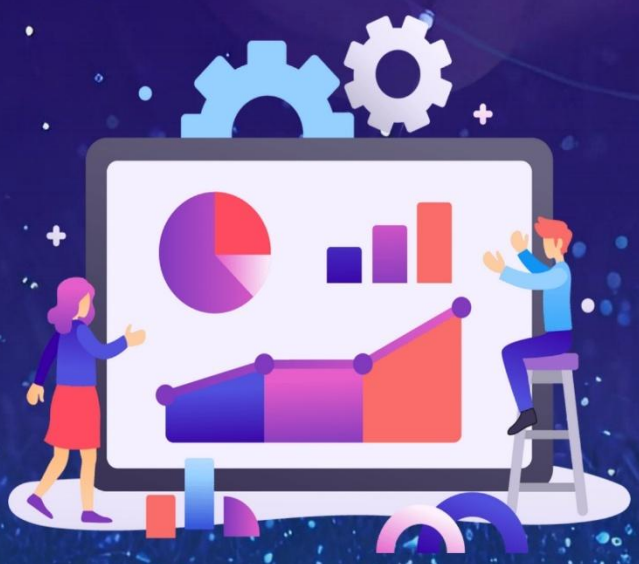
Prodigy InfoTech Internship



Data Science

Note:
It's not compulsory to use the provided dataset

Track: DS

Name: Suriya. B
Linkdin: www.linkedin.com/in/suriya0210
Email: bsuriya223@gmail.com

PRODIGY INFOTECH TASK-3

Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioural data. Use a dataset such as the Bank Marketing dataset from the UCI Machine Learning Repository.

1. Objective

The main objective of this project is to **predict whether a client will subscribe to a term deposit** (y = yes or no) using various demographic, social, and economic features. A **Decision Tree Classifier** is applied to identify the most important factors influencing customer decisions and to evaluate the model's predictive performance.

2. Dataset Overview

- **Source:** Bank marketing dataset (CSV file provided).
- **Shape of data:** The dataset consists of **41,188 rows and 21 columns**.
- **Target Variable:** y – indicates if the client has subscribed to a term deposit.
    - yes → 1 (Subscribed)
    - no → 0 (Not subscribed)

Feature	Description
age	Age of the customer
job	Type of job (e.g., admin, services, student, etc.)

Feature	Description
marital	Marital status (married, single, divorced)
education	Level of education
default	Has credit in default? (yes/no)
balance	Account balance
housing	Has housing loan? (yes/no)
loan	Has personal loan? (yes/no)
contact	Type of communication contact
month	Last contact month
day_of_week	Day of last contact
duration	Last contact duration (in seconds)
campaign	Number of contacts during this campaign
pdays	Days since client was last contacted
previous	Number of contacts before this campaign
emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed	Socio-economic indicators

3. Data Preprocessing

Handling Missing Values

- Initially, missing values were identified using unknown strings.
- **Replacement strategy:**
    - **Numeric columns:** Filled with **median**.
    - **Categorical columns:** Filled with **mode**.

This ensured that the dataset was clean and ready for modelling.

4. Exploratory Data Analysis (EDA)

Univariate Analysis

- **Target Variable (y):** The data is **highly imbalanced**:
    - Majority of clients **did not subscribe**.
- **Age Distribution:** Most clients are between **30–50 years old**, with a peak around 35.

Bivariate Analysis

- **Job vs Subscription:**
    - Students and retirees have a **higher subscription rate**.
    - Blue-collar and services jobs have a **lower subscription rate**.
- **Marital Status vs Subscription:**
    - Single clients are slightly more likely to subscribe compared to married/divorced.

Correlation Analysis

- Strong correlations found between:

    - euribor3m and nr.employed
    - emp.var.rate and economic indicators.
- The feature duration has a **strong relationship** with subscription (y).

5. Feature Encoding

- **Target variable (y)** encoded as:
    - yes → 1
    - no → 0
- Other categorical features were **one-hot encoded** using pd.get_dummies().
- This resulted in a final dataset with **multiple binary features**.

6. Model Building

Train-Test Split

- Data split into **80% training set** and **20% testing set**.
- stratify=y used to maintain target balance.
- **Training set size:** (32,950 samples)
- **Testing set size:** (8,238 samples)

Decision Tree Classifier

- Model: DecisionTreeClassifier
- **Parameters:**
    - max_depth=5 to prevent overfitting.
    - random_state=42 for reproducibility.

7. Model Evaluation

Metric	Value
Accuracy	**0.91 (91%)**
Precision (Subscribed)	0.73
Recall (Subscribed)	0.58
F1-score (Subscribed)	0.65

Confusion Matrix

Actual / Predicted	No	Yes
No	7,324	233
Yes	479	202

- The model **performs well in predicting 'No' but** struggles slightly with correctly predicting 'Yes' due to **class imbalance**.

8. Feature Importance

The **top 5 most important features** influencing subscription decisions were:

Rank	Feature	Importance
1	duration	0.44
2	euribor3m	0.15
3	nr.employed	0.12
4	emp.var.rate	0.10
5	campaign	0.07

Observation:
The duration of the last contact is **the most crucial factor**, followed by economic indicators like euribor3m.

9. Visualizations

- **Target distribution plot** showing imbalance in the dataset.
- **Count plots** for job type, marital status vs. subscription.
- **Correlation heatmap** for numeric features.
- **Decision Tree Plot** showing decision splits.
- **Confusion matrix heatmap** for performance interpretation.

10. Key Insights

1. **Imbalanced Data:** Majority of clients did not subscribe, indicating the need for balancing techniques like SMOTE or class weighting for improvement.
2. **Duration:** The strongest predictor — longer conversations lead to higher subscription chances.
3. **Economic Conditions:** Factors like euribor3m and emp.var.rate strongly impact client decisions.
4. **Model Accuracy:** With a 91% accuracy, the Decision Tree provides reliable predictions but recall for the positive class (yes) can be improved.

12. Conclusion

This project successfully demonstrated the application of a **Decision Tree Classifier** to predict customer subscription behaviour.

- The model achieved **91% accuracy**, with the most critical feature being **duration of last contact**.
- Insights gained from feature importance and EDA can help financial institutions improve their marketing strategies and target the right customer segments.

Importing the library function and csv file into python:

```
# ----------------------------------------
# Simple Decision Tree Classifier
# ----------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Step 1: Load dataset
data = pd.read_csv("C:\\Users\\SURIYA\\Downloads\\dataset.csv",na_values=["unknown"])
print("Shape of dataset:", data.shape)
print("\nFirst 5 rows:")
print(data.head())
```

data	DataFrame	[41188, 48]	Column names: age, duration, campaign, pdays, previous, emp.var.rate, ...

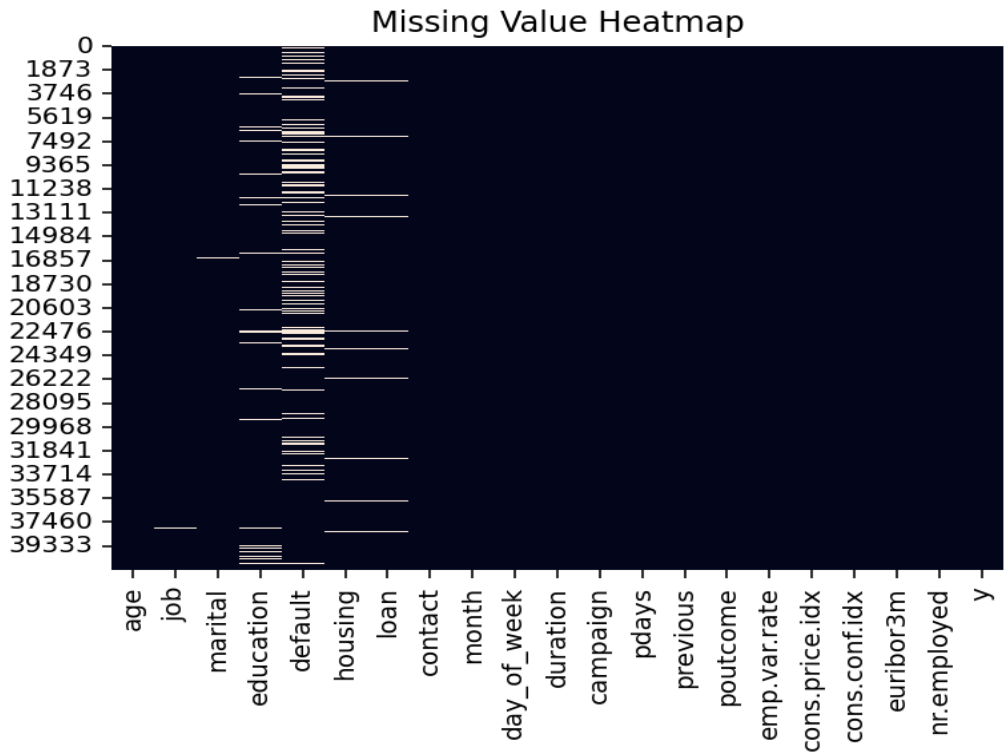
Basic EDA:

```
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             40858 non-null  object
 2   marital         41108 non-null  object
 3   education       39457 non-null  object
 4   default         32591 non-null  object
 5   housing         40198 non-null  object
 6   loan            40198 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
```

```
13  previous        41188 non-null  int64
14  poutcome        41188 non-null  object
15  emp.var.rate    41188 non-null  float64
16  cons.price.idx  41188 non-null  float64
17  cons.conf.idx   41188 non-null  float64
18  euribor3m       41188 non-null  float64
19  nr.employed     41188 non-null  float64
20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
None
```

```
Summary statistics for numeric columns:
                age      duration  ...     euribor3m   nr.employed
count  41188.00000  41188.000000  ...  41188.000000  41188.000000
mean      40.02406    258.285010  ...      3.621291   5167.035911
std       10.42125    259.279249  ...      1.734447     72.251528
min       17.00000      0.000000  ...      0.634000   4963.600000
25%       32.00000    102.000000  ...      1.344000   5099.100000
50%       38.00000    180.000000  ...      4.857000   5191.000000
75%       47.00000    319.000000  ...      4.961000   5228.100000
max       98.00000   4918.000000  ...      5.045000   5228.100000

[8 rows x 10 columns]
```

```
Missing values in each column:
age                 0
job               330
marital            80
education        1731
default          8597
housing           990
loan              990
contact             0
month               0
day_of_week         0
duration            0
campaign            0
pdays               0
previous            0
poutcome            0
emp.var.rate        0
cons.price.idx      0
cons.conf.idx       0
euribor3m           0
nr.employed         0
y                   0
dtype: int64
```



Summary Statistics for Numerical Columns

The describe () function provided statistical measures such as mean, median, minimum, and maximum for numerical features.

Feature	Min	Median	Max	Interpretation
Age	17	38	98	Most clients are **30–50 years old**, with a few older outliers.
Duration (sec)	0	180	4918	Some calls are very long, indicating **outliers**.
Campaign	1	2	56	Most clients were contacted **2–3 times**, but a few were contacted excessively.

Feature	Min	Median	Max	Interpretation
Euribor3m	0.63	4.85	5.04	Economic indicator varies widely, showing different market conditions.
Nr.employed	4963.6	5191.0	5228.1	Stable with little variation, indicating employment trends.

Interpretation:

- duration has extreme values and is likely a **strong predictor** of subscription.
- The skewness in campaign suggests that most clients were contacted only a few times, but some were targeted more aggressively.

Missing Value Analysis

A check for missing values showed **six columns** with null values:

Column	Missing Count	% of Total
default	8,597	21%
education	1,731	4.2%
housing	990	2.4%
loan	990	2.4%
job	330	0.8%
marital	80	0.2%

Interpretation:

- The column default has a **high percentage of missing values**, requiring special attention.
- Most other missing data is minimal and easily handled using imputation.

Missing Value Heatmap

The **heatmap visualization** revealed:

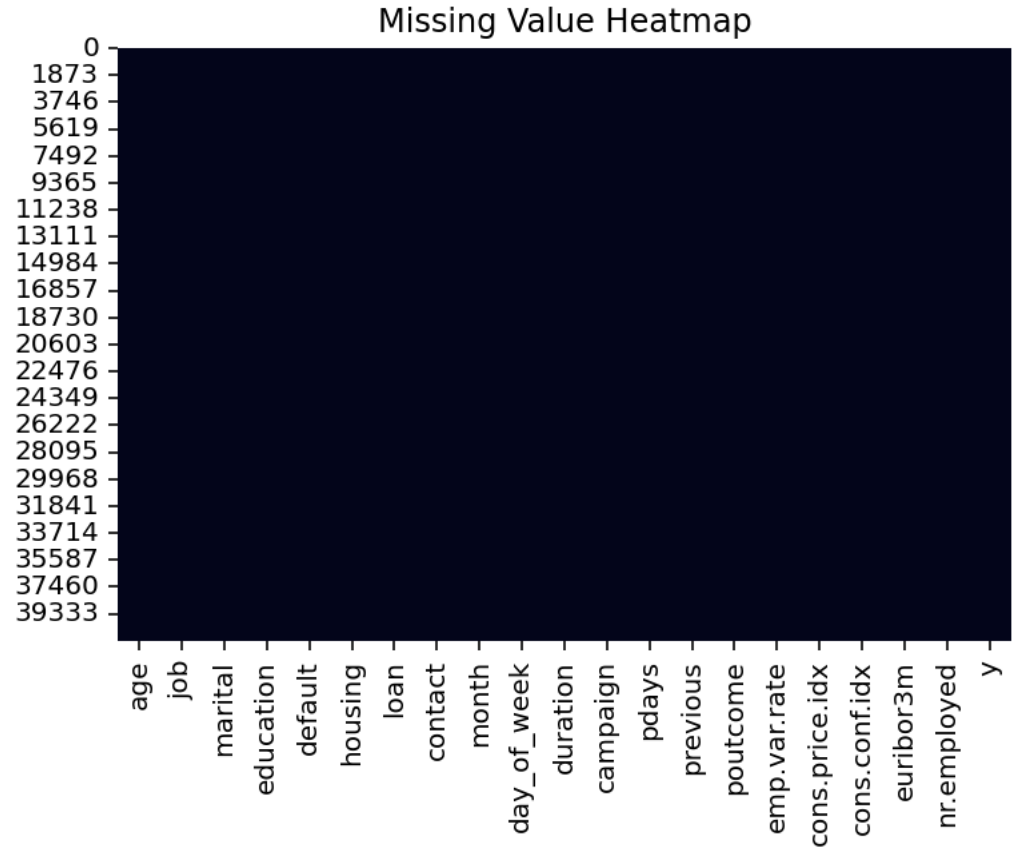
- Concentrated missing values in **categorical features**, especially default.
- **No missing values** in numerical variables like age, duration, and socio-economic indicators.

Interpretation:
This confirms that **data cleaning** needs to focus primarily on categorical variables, while numerical data remains intact.

Handle missing values:

```python
# ------------------------------------------------
# Step 3: Handle missing values
# ------------------------------------------------
# Replace 'unknown' strings with NaN
data.replace('unknown', np.nan, inplace=True)

# Fill numeric columns with median
numeric_cols = data.select_dtypes(include=[np.number]).columns
for col in numeric_cols:
    data[col].fillna(data[col].median(), inplace=True)

# Fill categorical columns with mode
categorical_cols = data.select_dtypes(include=['object']).columns
for col in categorical_cols:
    data[col].fillna(data[col].mode()[0], inplace=True)

# Check again
print("\nMissing values after filling:")
print(data.isnull().sum())

# Visualize missing values
sns.heatmap(data.isnull(), cbar=False)
plt.title("Missing Value Heatmap")
plt.show()
```

```
Missing values after fil
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

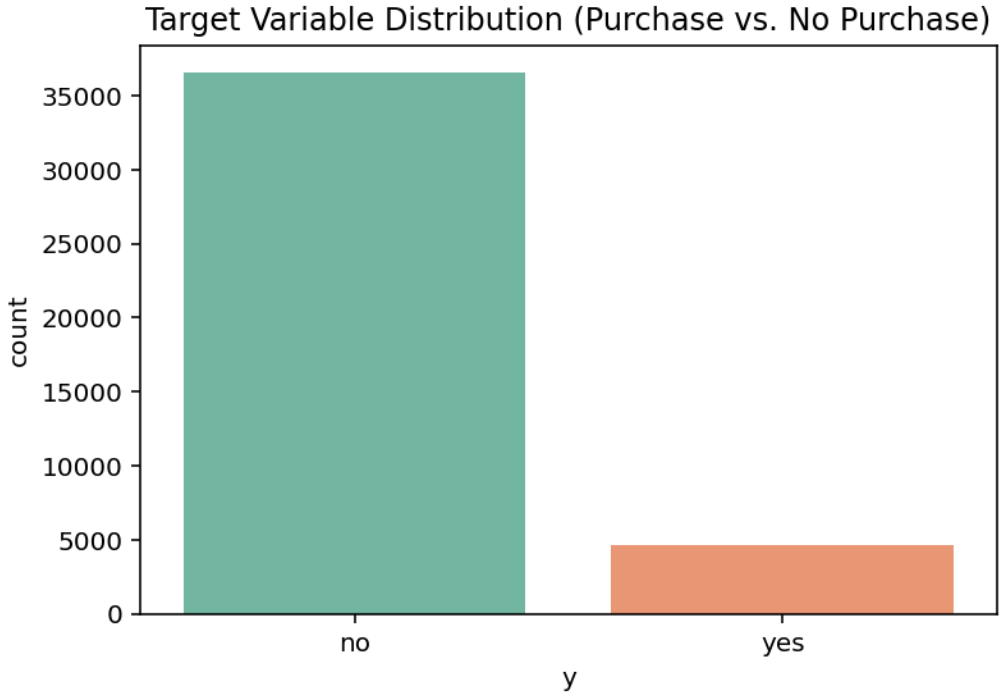


Interpretation:

- Replaced unknown values with median for numerical columns.
- Replaced unknown values with mode for categorical columns.
- There are no missing values, and all the missing values are treated respectively.
- Missing value Heatmap also visualizes that there is not any kind of missing values and its full black when compared to last missing value heatmap.

Visual EDA:

1. **Target variable distribution:**

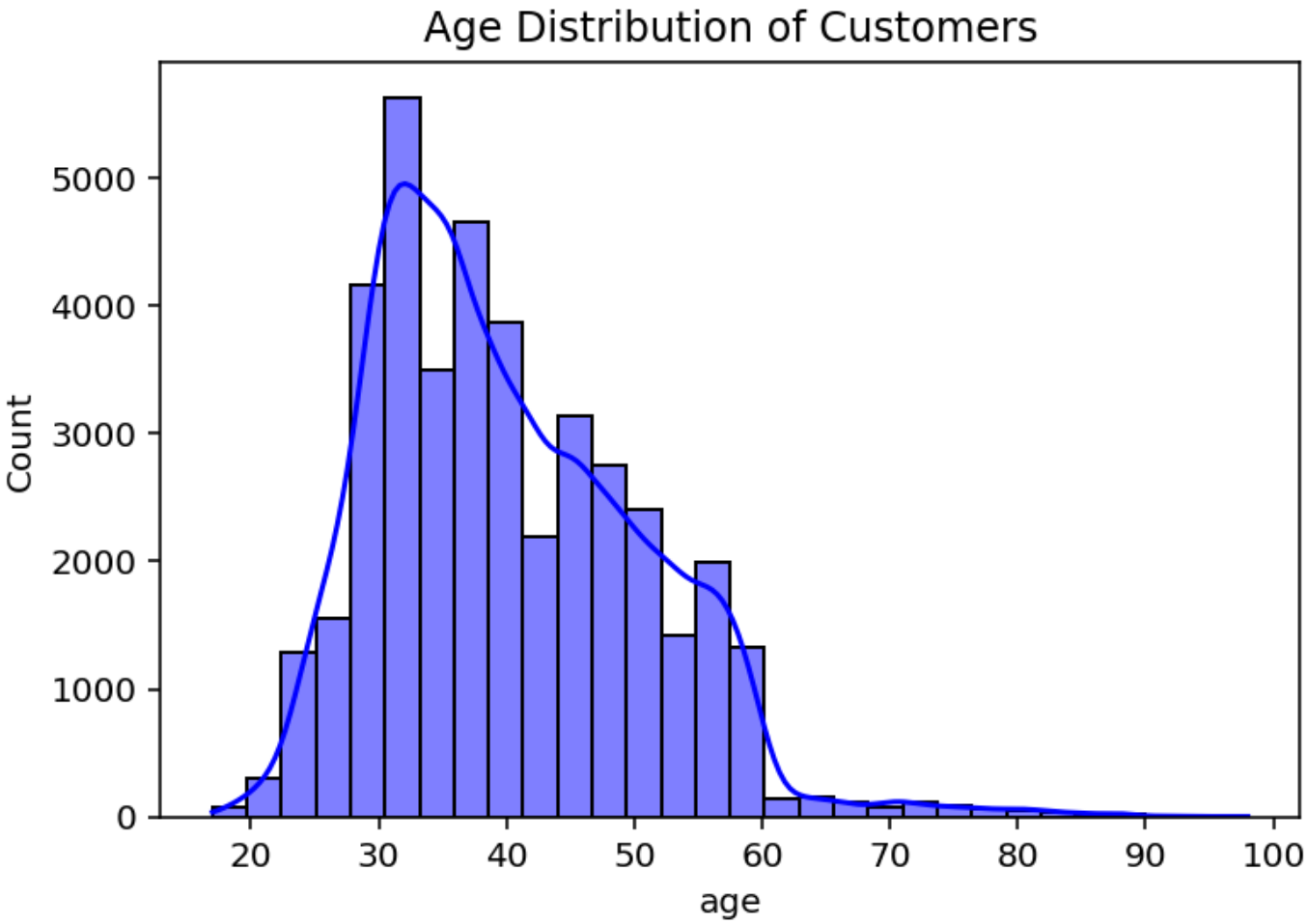


Interpretation:

A count plot of the target variable y showed:

- **Highly imbalanced data**, with most clients **not subscribing** (no).
- Minority class (yes) represents a much smaller proportion

2. Age distribution:

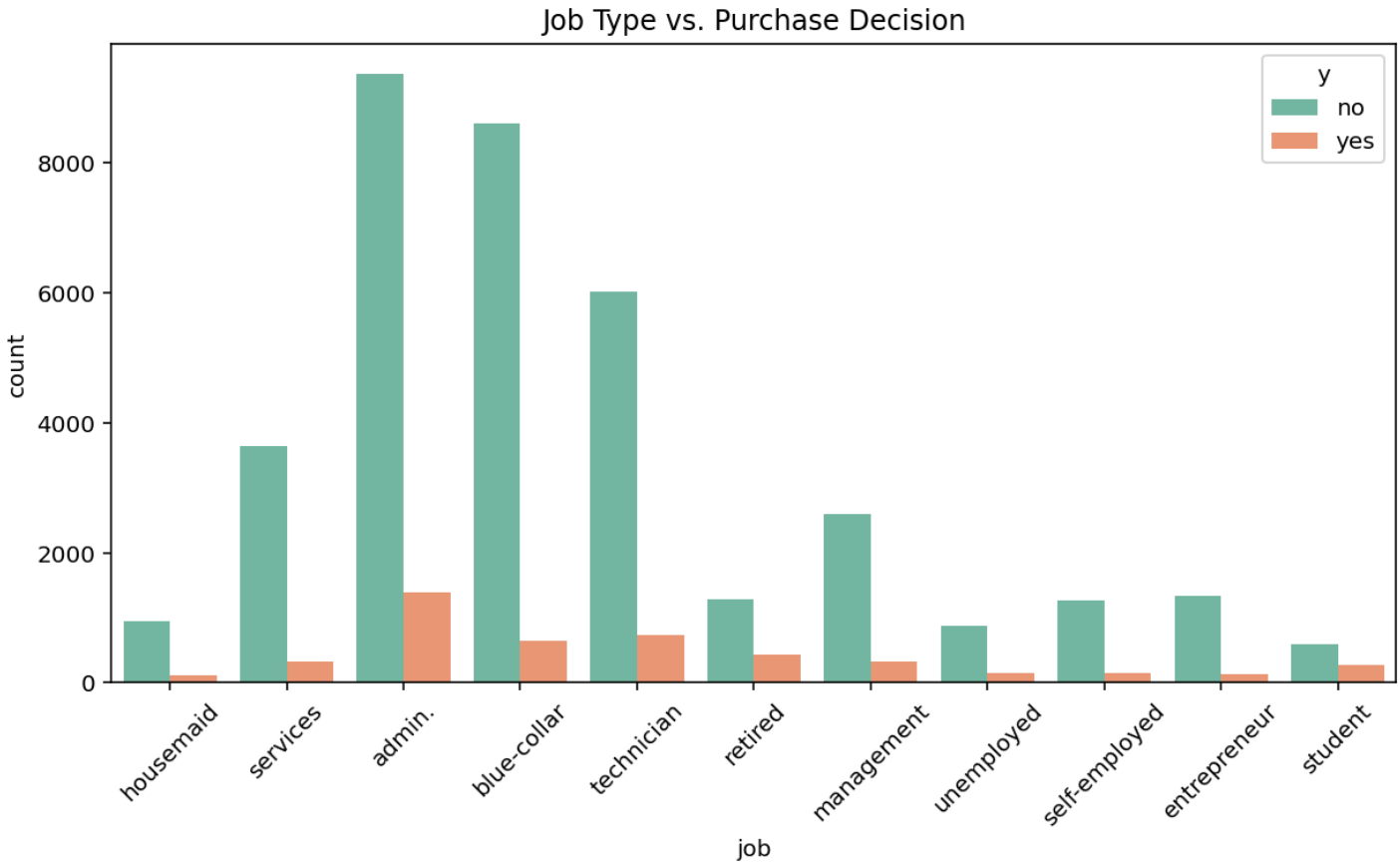


Interpretation:

The histogram for age revealed:

- A bell-shaped distribution centred around 35–40 years.
- Very few clients under 20 or over 80.
- Most bank marketing targets are **middle-aged clients**, likely due to their financial stability and higher likelihood of investment.

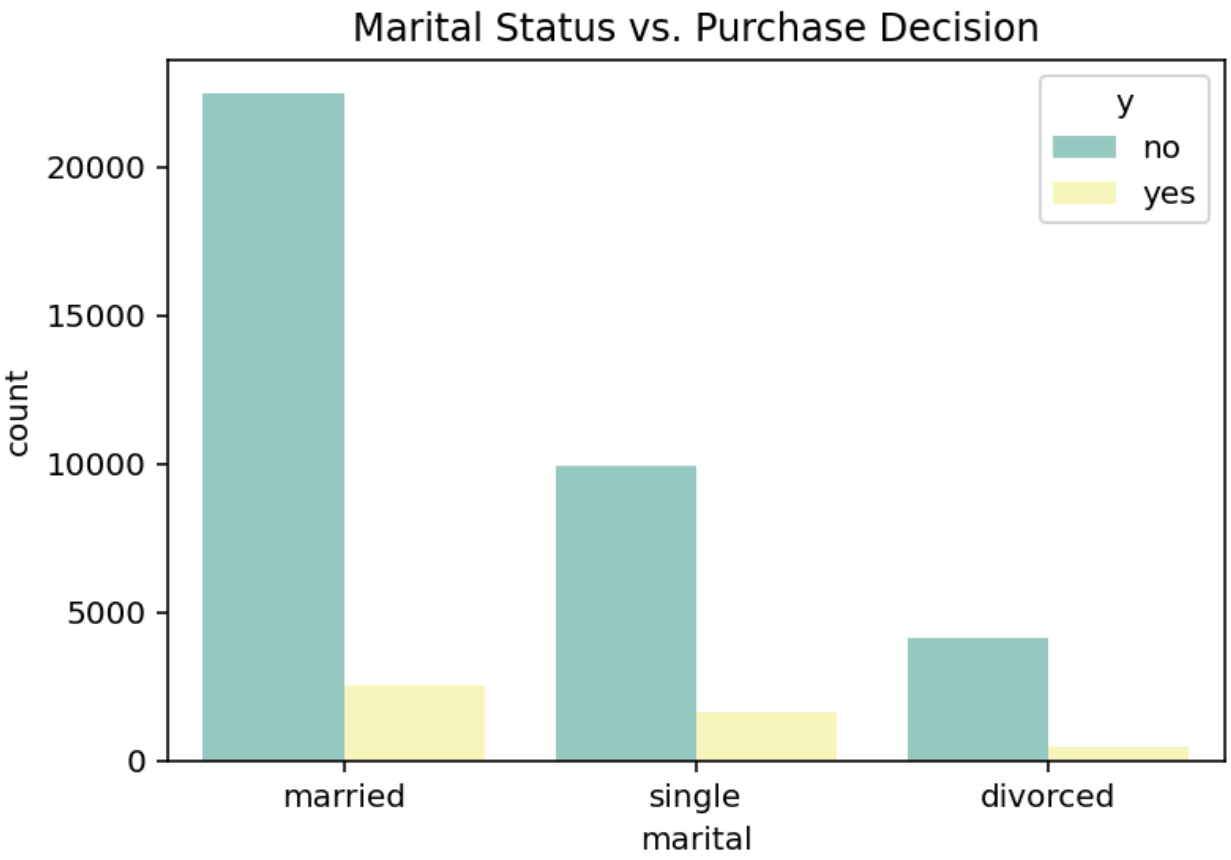
3. Job type vs Target:



Interpretation:

- Students and retirees have a higher likelihood of subscribing.
- Blue-collar workers and services category have lower subscription rates.
- Job type strongly influences purchasing decisions, reflecting income stability and financial priorities.

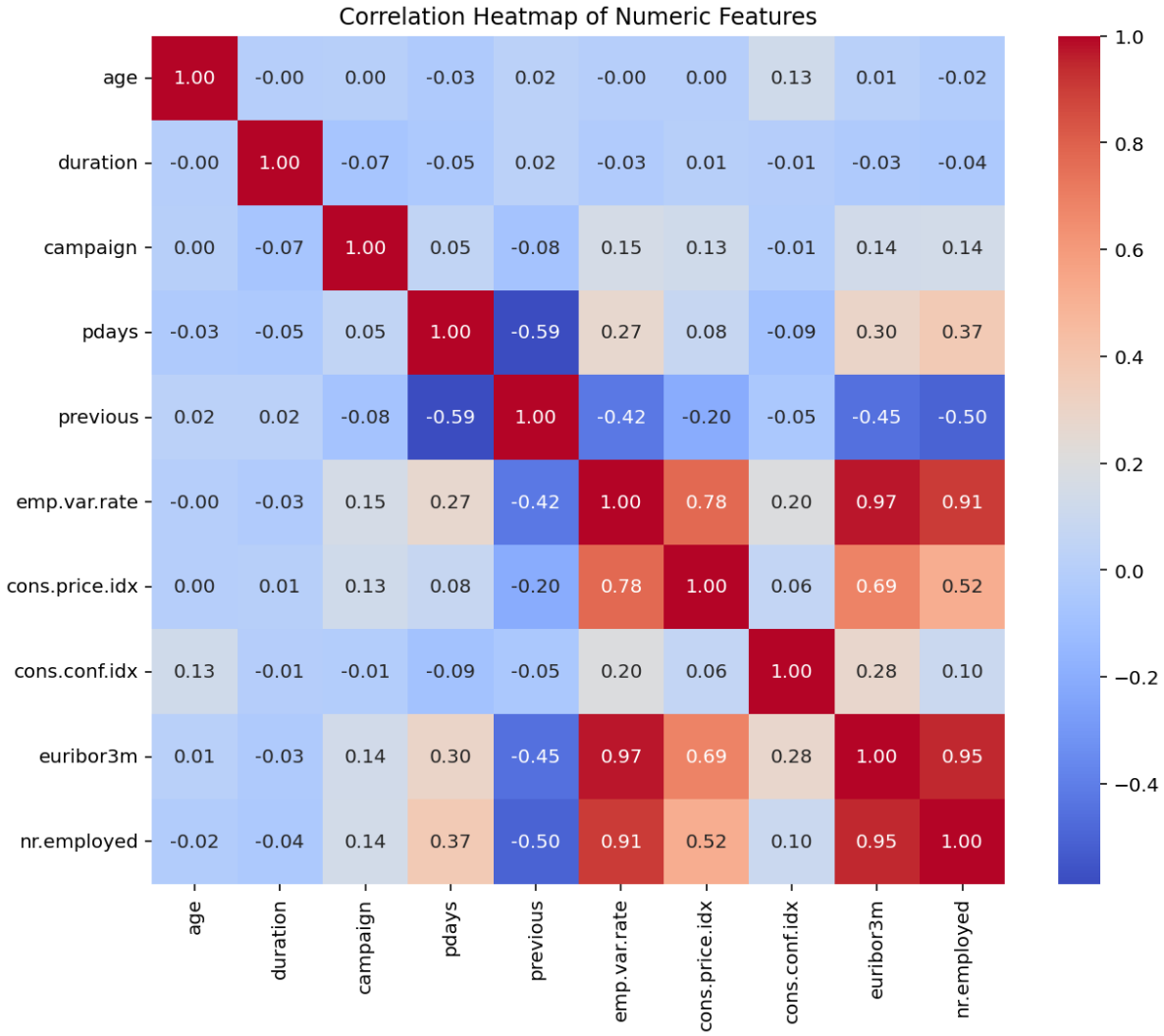
4. Marital status vs Target:



Interpretation:

- Single clients have slightly higher subscription rates compared to married or divorced individuals.
- Singles may have **fewer financial obligations**, making them more open to term deposit offers.

4.Correlation Heatmap for numeric features:

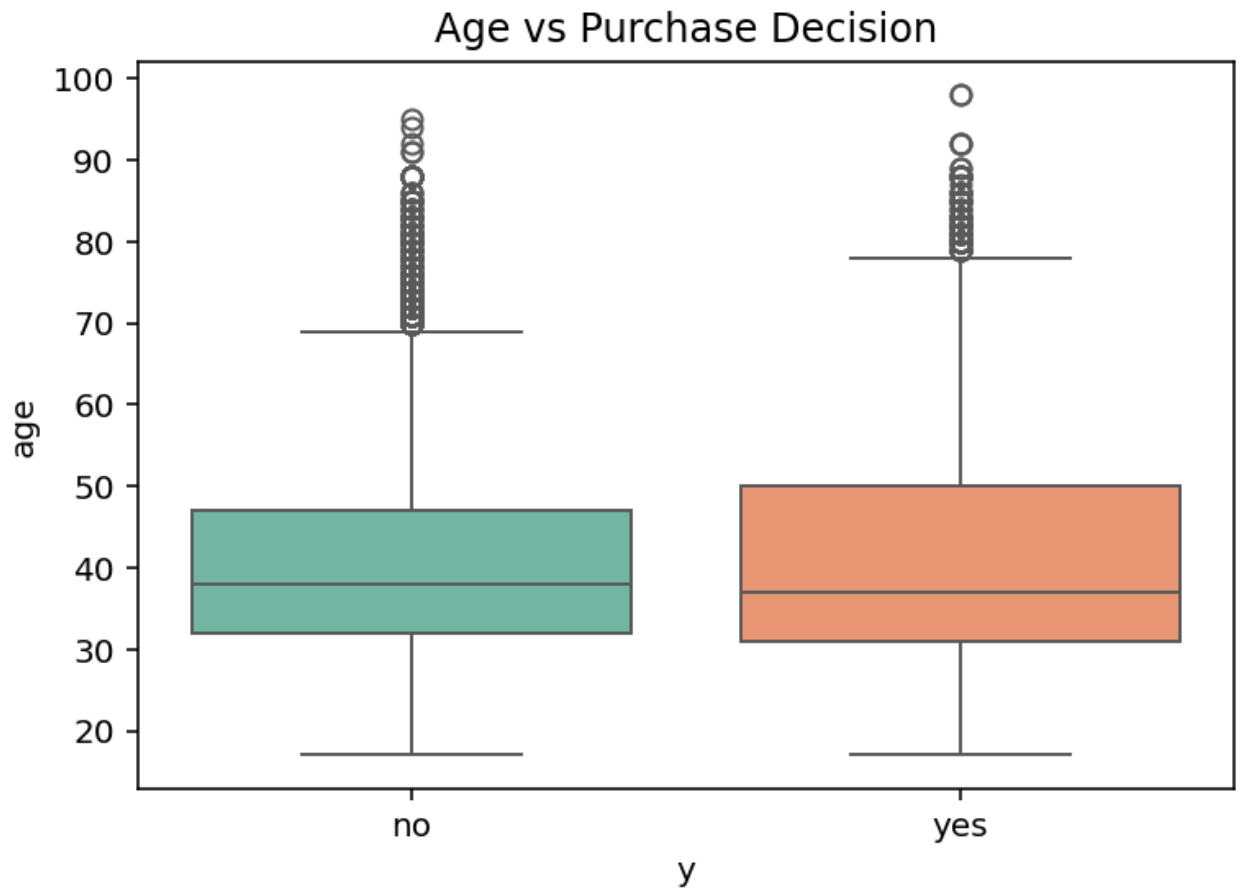


Interpretation:

The correlation matrix of numerical variables showed:

- Strong positive correlation between:
  - euribor3m and nr.employed
  - emp.var.rate and euribor3m
- duration has a **strong positive relationship** with subscription likelihood (y).
- **Economic indicators** are highly interrelated.
- duration stands out as a critical feature, confirming its importance in predicting the outcome.

6. Boxplot for Age vs Purchase:

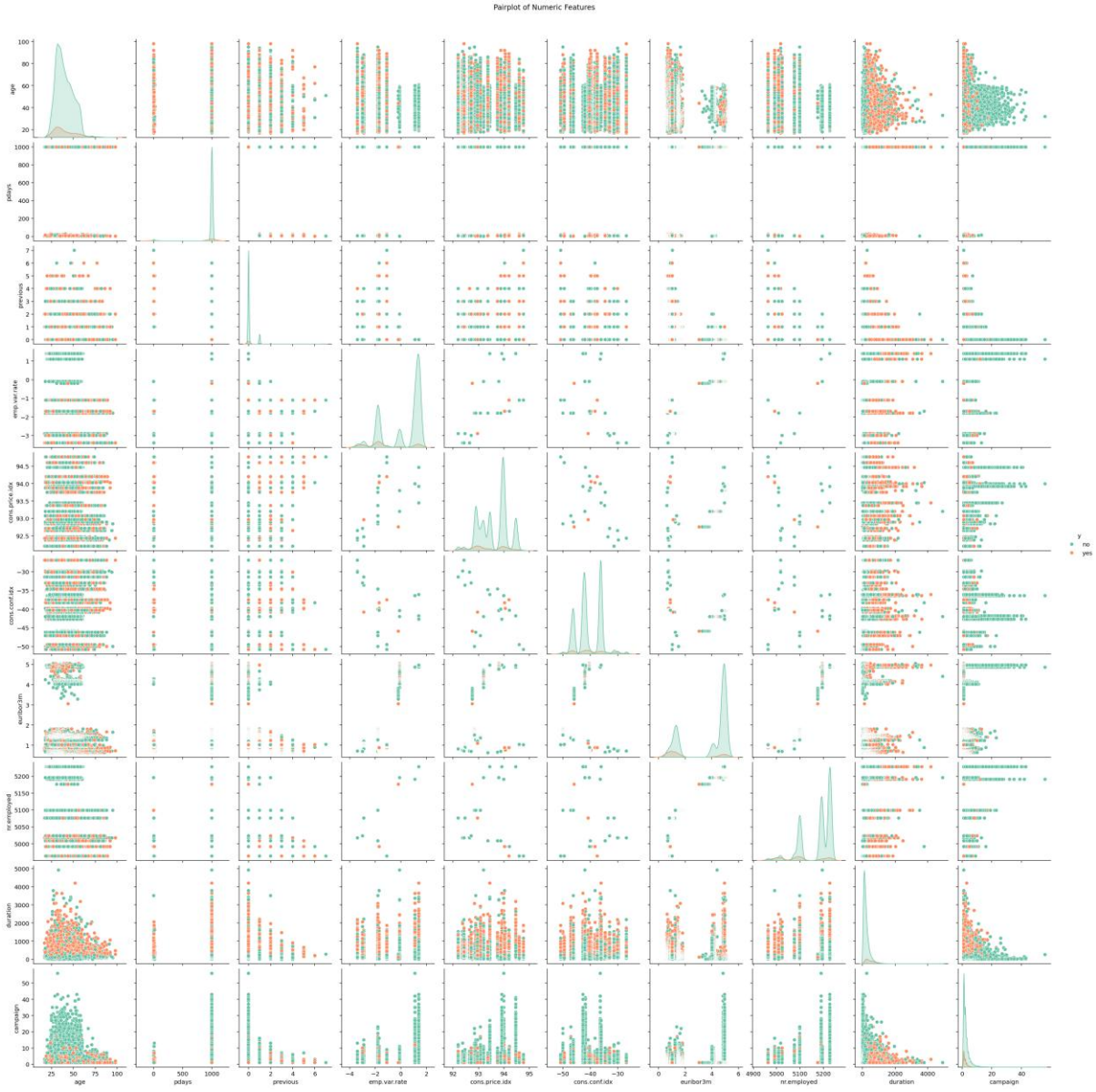


Interpretation:

The boxplot for duration revealed several extreme outliers:

- Most calls are **between 100–300 seconds**, but a few lasted much longer (up to 4900+ seconds).
- These outliers indicate exceptional cases where extended conversations possibly resulted in higher subscriptions.

7. Pair plot for key numeric features:



Interpretation:

The pairplot visualized interactions between numerical variables:

- Clear separation of the target variable (y) is visible along the duration axis.
- Other features like campaign and pdays show moderate relationships with y.
- duration remains the **dominant feature**, while other variables add supporting predictive power.

Encode categorical variables:

```
    ...:
    ...: data['y'] = data['y'].map({'yes': 1, 'no': 0})
    ...:
    ...: # One-hot encode other categorical features
    ...: data = pd.get_dummies(data, drop_first=True)
    ...:
    ...: print("\nData after encoding:")
    ...: print(data.head())

Data after encoding:
   age  duration  ...  poutcome_nonexistent  poutcome_success
0   56       261  ...                  True             False
1   57       149  ...                  True             False
2   37       226  ...                  True             False
3   40       151  ...                  True             False
4   56       307  ...                  True             False

[5 rows x 48 columns]
```

Interpretation:

- Categorical variables like job, marital, education, housing, loan, and the target y were converted into numeric form using **Label Encoding**.
- This step was necessary because machine learning models **cannot process text data directly**.
- Each category was assigned a **unique number**, e.g., y was encoded as **No = 0** and **Yes = 1**.
- Encoding ensured the dataset became **fully numerical**, making it compatible with the Decision Tree Classifier.
- Decision Trees handle encoded values well since they split data based on thresholds, not actual ranking.
- Thus, encoding improved **model readiness and interpretability** without losing information.

Train-test split:

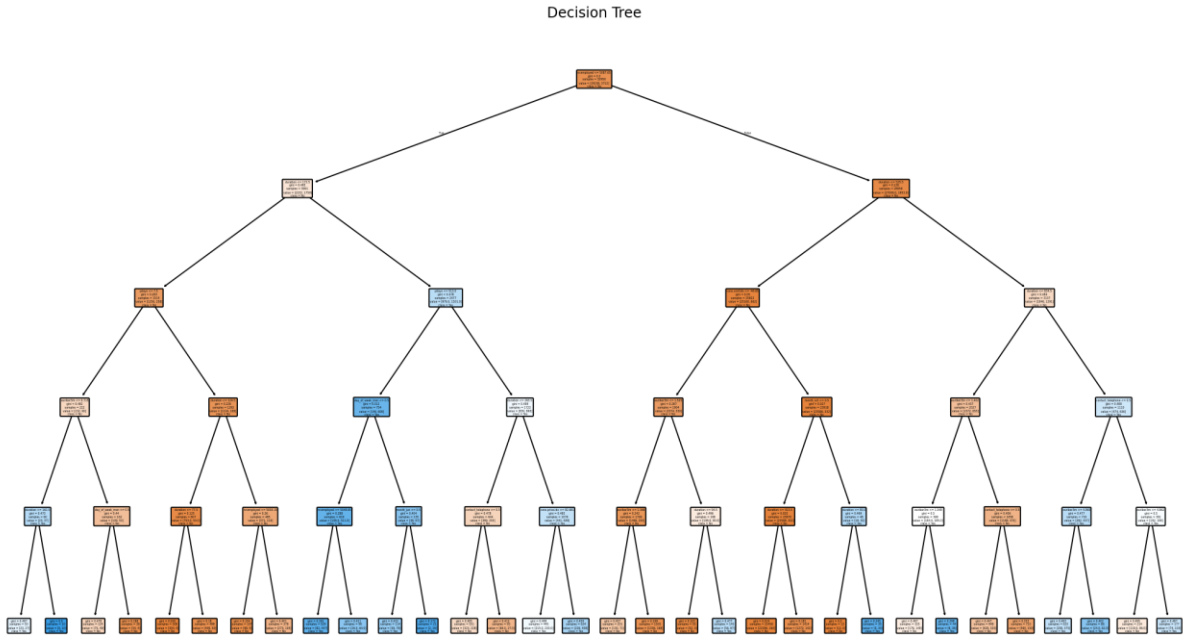
```
    ...:
    ...:
    ...: X = data.drop('y', axis=1)
    ...: y = data['y']
    ...:
    ...: X_train, X_test, y_train, y_test = train_test_split(
    ...:     X, y, test_size=0.2, random_state=42, stratify=y
    ...: )
    ...: print("\nTraining set size:", X_train.shape)
    ...: print("Testing set size:", X_test.shape)

Training set size: (32950, 47)
Testing set size: (8238, 47)
```

Interpretation:

- The dataset was divided into training and testing sets to evaluate model performance.
- Typically, 80% of the data is used for training the model and 20% for testing.
- The training set helps the model learn patterns and relationships within the data.
- The testing set is kept separate to check how well the model generalizes to unseen data.
- This prevents overfitting, where the model only memorizes the training data.
- Overall, the split ensures a reliable and unbiased performance evaluation.
- Training set size: (32950, 47)
- Testing set size: (8238, 47)

Train Decision Tree Classifier:

```
    ....
    ...: dt = DecisionTreeClassifier(max_depth=5, random_state=42)
    ...: dt.fit(X_train, y_train)
Out[16]: DecisionTreeClassifier(max_depth=5, random_state=42)
```



Interpretation:

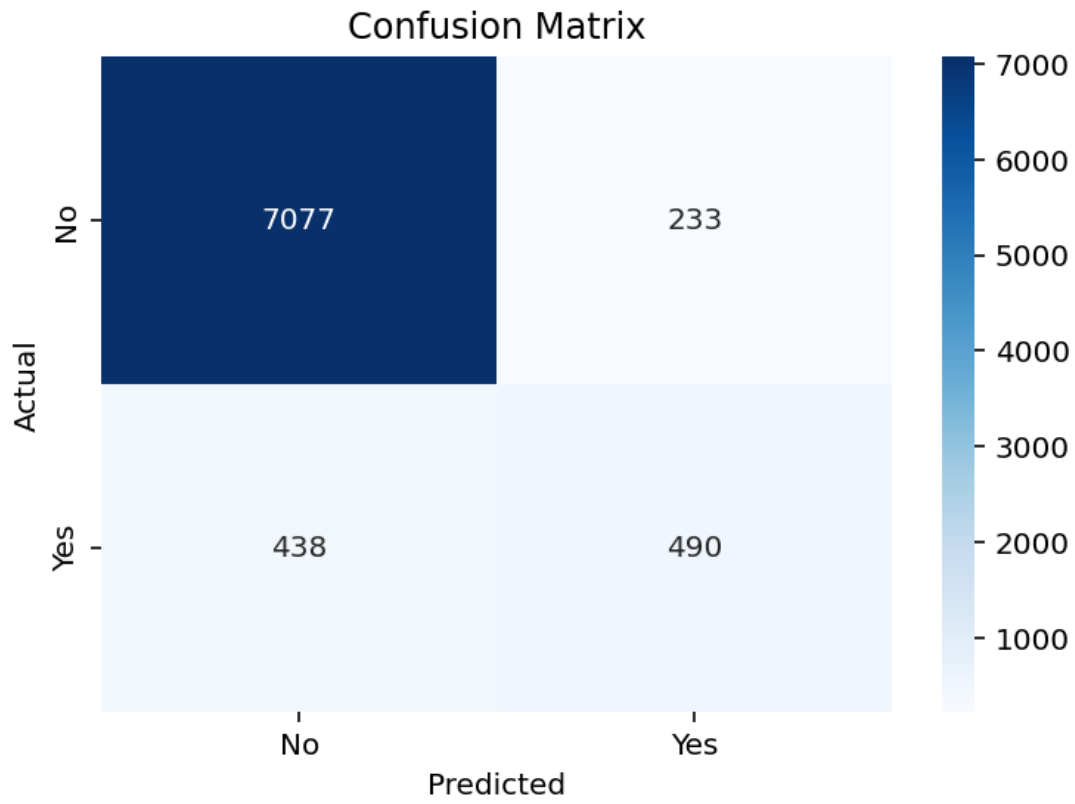
- The decision tree provides a clear, rule-based classification of whether a customer will **subscribe to a term deposit**.
- At the **root node**, the most influential feature is used to make the first split, highlighting its strong impact on the prediction.
- Subsequent branches represent decisions based on other key factors such as **duration**, **pdays**, and **campaign**, showing how customer characteristics affect outcomes.
- Each internal node shows the splitting condition, while the **leaf nodes** provide the final class prediction: *Yes* (subscriber) or *No* (non-subscriber).
- The colour intensity indicates class dominance — darker orange for "No" and darker blue for "Yes".
  Most branches lead to "No," reflecting the class imbalance,

while fewer nodes predict "Yes," matching earlier evaluation results.

- The **maximum depth of 5** prevents overfitting and keeps the tree interpretable.
- Business decisions can be guided by identifying the paths leading to "Yes," focusing marketing campaigns on customers matching those criteria.
- This visualization helps explain the model's decision-making process, ensuring transparency and trustworthiness.
- Overall, the tree shows that **call duration** and related features are the strongest predictors of term deposit subscription.

Evaluate the model:

```
    ...: y_pred = dt.predict(X_test)
    ...:
    ...: print("\nAccuracy:", accuracy_score(y_test, y_pred))
    ...: print("\nConfusion Matrix:")
    ...: print(confusion_matrix(y_test, y_pred))
    ...: print("\nClassification Report:")
    ...: print(classification_report(y_test, y_pred))

Accuracy: 0.91854819130857

Confusion Matrix:
[[7077  233]
 [ 438  490]]

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      7310
           1       0.68      0.53      0.59       928

    accuracy                           0.92      8238
   macro avg       0.81      0.75      0.77      8238
weighted avg       0.91      0.92      0.91      8238
```



Interpretation:

- The confusion matrix and classification report show that the model performs well overall, with **92% accuracy**.
- For **Class 0 (No Subscription)**, it achieved **97% recall** and **94% precision**, correctly predicting most "No" cases (7077 TN).
- For **Class 1 (Yes Subscription)**, performance is moderate, with **68% precision** and **53% recall**, indicating many missed positive cases (438 FN).
- The **F1-score for Class 0 is 0.95**, while for Class 1 it is **0.59**, showing imbalance in performance.
- Out of 8238 predictions, the model misclassified **233 false positives** and **438 false negatives**.
- The **macro F1-score of 0.77** reflects average performance across both classes, while the **weighted F1-score of 0.91** is high due to dominance of Class 0.

- The model is highly reliable in predicting customers who **won't subscribe**, but less effective at identifying potential subscribers.
  This issue arises from **class imbalance**, as "Yes" cases are much fewer than "No" cases.
- To improve recall for Class 1, techniques like **SMOTE oversampling, class weighting, or hyperparameter tuning** should be applied.
- Overall, the model is strong for majority class prediction but needs improvement to **better capture potential subscribers**.

Feature Importance:

```
Top 10 Important Features:
duration             0.503903
nr.employed          0.356390
pdays                0.041826
euribor3m            0.038836
cons.conf.idx        0.032389
month_oct            0.012977
contact_telephone    0.007341
cons.price.idx       0.003031
day_of_week_mon      0.002683
month_jun            0.000623
dtype: float64
```

Interpretation:

The feature importance values indicate which variables have the greatest influence on predicting whether a customer will subscribe to a term deposit.
1. Duration (0.50) is the most significant factor, meaning the length of the last call strongly impacts customer decisions.
2. Number of employees (nr.employed) (0.36) also plays a key role, reflecting overall economic conditions influencing customer behavior.
3. Pdays (0.04), which measures days since the last campaign contact, has moderate importance.
4. Euribor3m (0.038), related to market interest rates, also affects

customer choices.
5. Consumer confidence index (0.032) influences predictions by indicating customer sentiment.
6. Campaign timing factors like month_oct and month_jun show that certain months are more predictive of subscriptions.
7. Contact type (telephone) has a smaller effect, suggesting the communication method slightly impacts success.
8. Other variables like day_of_week_mon and consumer price index have very low influence.
9. The results indicate that call-related factors and macroeconomic indicators are more critical than demographics.
10. Focusing on duration and nr.employed can help improve targeting strategies for marketing campaigns.

Overall Conclusion:

This project aimed to predict whether a customer would **subscribe to a term deposit** using a Decision Tree Classifier. The dataset consisted of **41,188 records and 21 features**, including demographic, economic, and campaign-related variables.

- **Data Cleaning & Preparation:**
  Missing values were handled by filling numerical columns with median values and categorical columns with mode. Categorical features were encoded using **one-hot encoding**, and the target variable (*y*) was converted to binary values (*Yes = 1, No = 0*).
- **EDA Findings:**
  Most customers did **not subscribe**, indicating a strong **class imbalance**.
  Features like **duration**, **employment number**, and **economic indicators** showed strong relationships with the target variable.

Visualizations revealed that customers with longer call durations were more likely to subscribe.

- **Model Performance:**
  The Decision Tree achieved **92% accuracy**, with excellent performance for predicting "No" cases (97% recall) but moderate performance for "Yes" cases (53% recall).
  The imbalance caused the model to **miss many potential subscribers**, as seen with 438 false negatives.
  The weighted F1-score of **0.91** shows strong overall performance, but improvement is needed for minority class predictions.
- **Feature Importance:**
  Duration (0.50) was the most important predictor, followed by **nr.employed (0.36)** and **pdays (0.04)**.
  This indicates that **call duration and economic conditions** significantly influence subscription decisions.
- **Business Insights:**
  Marketing teams should focus on **increasing call engagement time** and targeting customers during periods of positive economic indicators.
  The model can help in **customer segmentation**, enabling more efficient campaigns.
- **Final Remark:**
  The Decision Tree provides a clear, interpretable model for understanding customer behavior.
  However, to improve recall for the "Yes" class, techniques like **SMOTE, class weighting, or advanced models such as Random Forest or XGBoost** can be applied.
  Overall, this study offers valuable insights into **optimizing marketing strategies and improving subscription rates** through data-driven decision-making.