

# Task-3 Building a decision tree classifier

```
In [1]: import pandas as pd
import numpy as np
```

```
In [2]: df = pd.read_csv('bank.csv')
```

```
In [3]: print(df.dtypes)
```

```
age          int64
job          object
marital      object
education    object
default      object
balance      int64
housing      object
loan         object
contact      object
day          int64
month        object
duration     int64
campaign     int64
pdays       int64
previous     int64
poutcome    object
deposit      object
dtype: object
```

```
In [4]: df.head(5)
```

```
Out[4]:
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	10
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	14
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	13
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	13
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	6

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   age         11162 non-null  int64
1   job         11162 non-null  object
2   marital     11162 non-null  object
3   education   11162 non-null  object
4   default     11162 non-null  object
5   balance     11162 non-null  int64
6   housing     11162 non-null  object
7   loan        11162 non-null  object
8   contact     11162 non-null  object
9   day         11162 non-null  int64
10  month       11162 non-null  object
11  duration    11162 non-null  int64
12  campaign    11162 non-null  int64
13  pdays       11162 non-null  int64
14  previous    11162 non-null  int64
15  poutcome    11162 non-null  object
16  deposit     11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

## Encoding categorical features

```
In [6]: df_encoded = pd.get_dummies(df)

print(df_encoded.head()) #verify that all columns are now numeric
```

	age	balance	day	duration	campaign	pdays	previous	job_admin.	\
0	59	2343	5	1042	1	-1	0	1	
1	56	45	5	1467	1	-1	0	1	
2	41	1270	5	1389	1	-1	0	0	
3	55	2476	5	579	1	-1	0	0	
4	54	184	5	673	2	-1	0	1	

	job_blue-collar	job_entrepreneur	...	month_may	month_nov	month_oct	\
0	0	0	...	1	0	0	
1	0	0	...	1	0	0	
2	0	0	...	1	0	0	
3	0	0	...	1	0	0	
4	0	0	...	1	0	0	

	month_sep	poutcome_failure	poutcome_other	poutcome_success	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

	poutcome_unknown	deposit_no	deposit_yes
0	1	0	1
1	1	0	1
2	1	0	1
3	1	0	1
4	1	0	1

[5 rows x 53 columns]

## Creating the single target variable

```
In [7]: target_variable = df_encoded['deposit_yes'] # 1 for 'yes' , 0 for 'no'
```

## Prepare Features and Target Variable

```
In [8]: # Prepare the feature set
features = df_encoded.drop(['deposit_yes'], axis=1) # Remove the one-hot encoded column

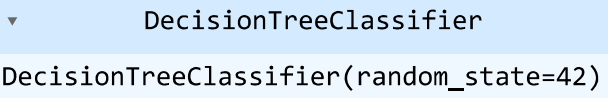
# Your single target variable
target = target_variable # Or the derived column if using one-hot encoded columns
```

## Train- Test Split and Model Training

```
In [9]: from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Train a simple Decision Tree model
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

Out[9]:  DecisionTreeClassifier(random\_state=42)

## Evaluate the model

```
In [10]: accuracy = clf.score(X_test, y_test)
print("Test Accuracy:", accuracy)
```

Test Accuracy: 1.0

In [ ]: