Tugas UTS Machine Learning

Nama : Ilham Novriadi

NIM : 231011403539

Dataset : Bank Marketing Dataset
(https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset)

Get the data

In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset (adjust the path if needed)
df = pd.read_csv("/Volumes/IDEKU/University/Code/data/bank_marketing.csv"

# Show basic info
print(df.shape)
df.head()
```

```
(11162, 17)
```

Out[18]:

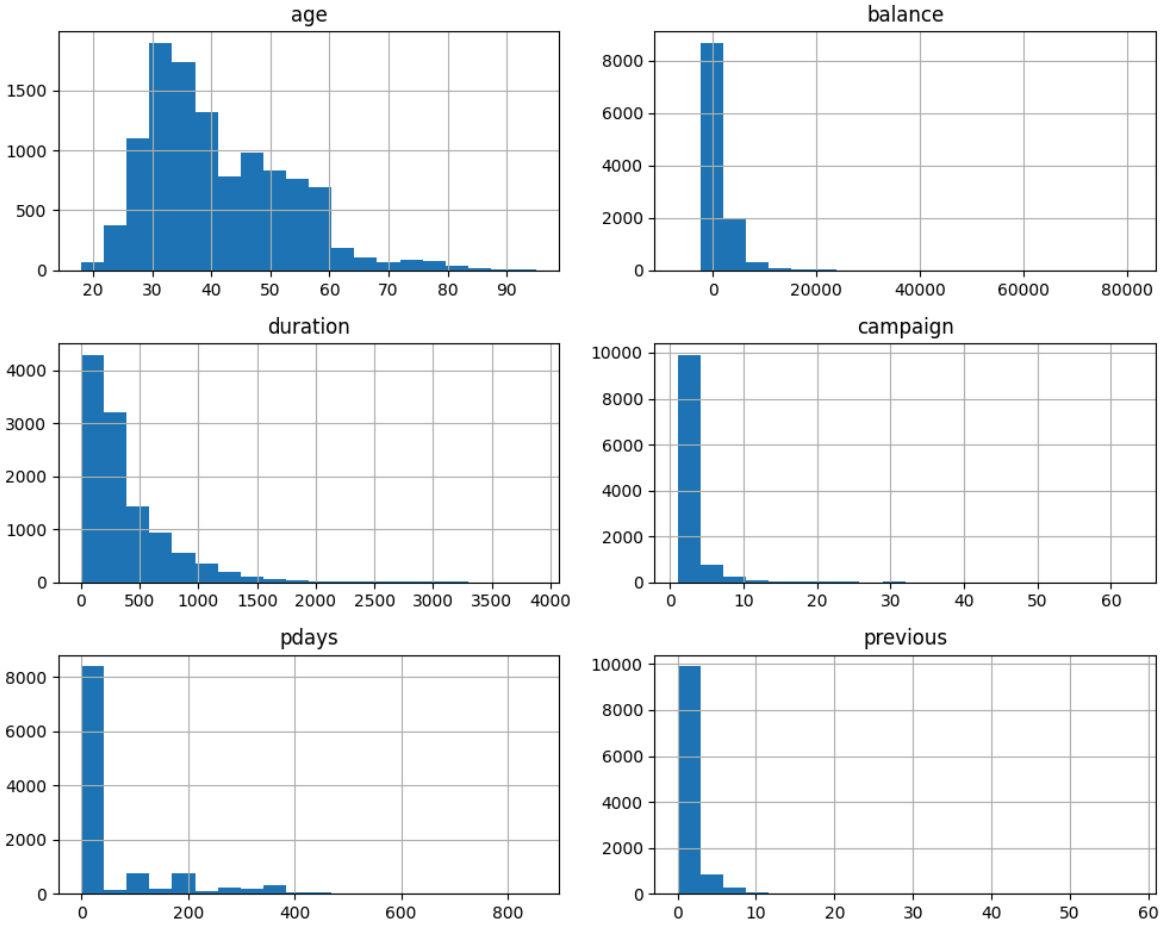
	age	job	marital	education	default	balance	housing	loan	contact	c
0	59	admin.	married	secondary	no	2343	yes	no	unknown	
1	56	admin.	married	secondary	no	45	no	no	unknown	
2	41	technician	married	secondary	no	1270	yes	no	unknown	
3	55	services	married	secondary	no	2476	yes	no	unknown	
4	54	admin.	married	tertiary	no	184	no	no	unknown	

EDA (Exploratory Data Analysis)

In [19]:
```
# Info about data types and missing values
df.info()

# Check missing values
print("\nMissing values per column:\n", df.isnull().sum())

# Target variable distribution
print("\nTarget value counts:")
print(df['deposit'].value_counts())

sns.countplot(x='deposit', data=df)
plt.title("Deposit Target Distribution")
plt.show()

# Example of numeric feature distribution
num_cols = ['age', 'balance', 'duration', 'campaign', 'pdays', 'previous'
df[num_cols].hist(figsize=(10,8), bins=20)
```

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

```
<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

Missing values per column:
 age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
deposit      0
dtype: int64

Target value counts:
deposit
no     5873
yes    5289
Name: count, dtype: int64
```



Preprocessing

In [20]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

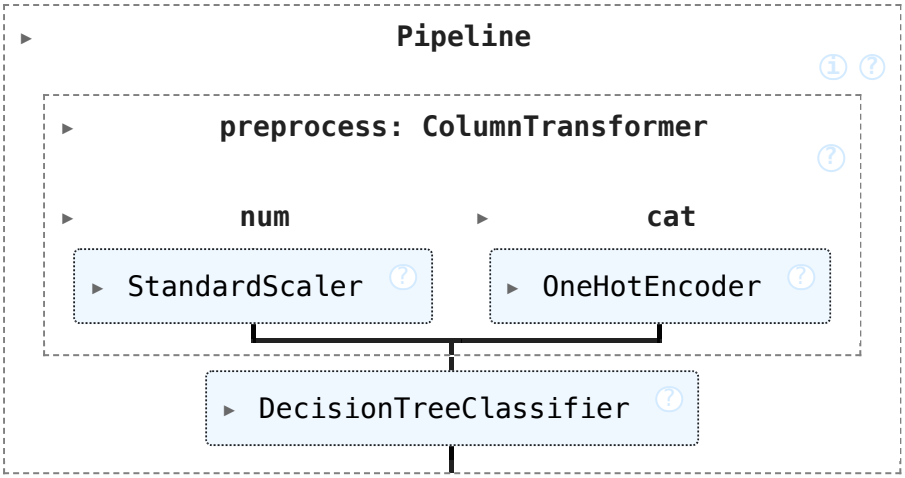
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Convert target to binary
df['deposit'] = df['deposit'].map({'yes': 1, 'no': 0})

# Split features and target
X = df.drop('deposit', axis=1)
y = df['deposit']

# Identify categorical & numeric columns
cat_cols = X.select_dtypes(include=['object']).columns.tolist()
num_cols = X.select_dtypes(include=['int64', 'float64']).columns.tolist()

# Column transformer: scale numeric, one-hot encode categorical
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore'), cat_cols)
    ]
)

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

Train Models

In [21]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

# Logistic Regression pipeline
pipe_lr = Pipeline([
    ('preprocess', preprocessor),
    ('clf', LogisticRegression(max_iter=1000, class_weight='balanced', ra
])

# Decision Tree pipeline
pipe_dt = Pipeline([
    ('preprocess', preprocessor),
    ('clf', DecisionTreeClassifier(max_depth=5, random_state=42, class_we
])

# Fit both models
pipe_lr.fit(X_train, y_train)
pipe_dt.fit(X_train, y_train)
```

Out[21]:

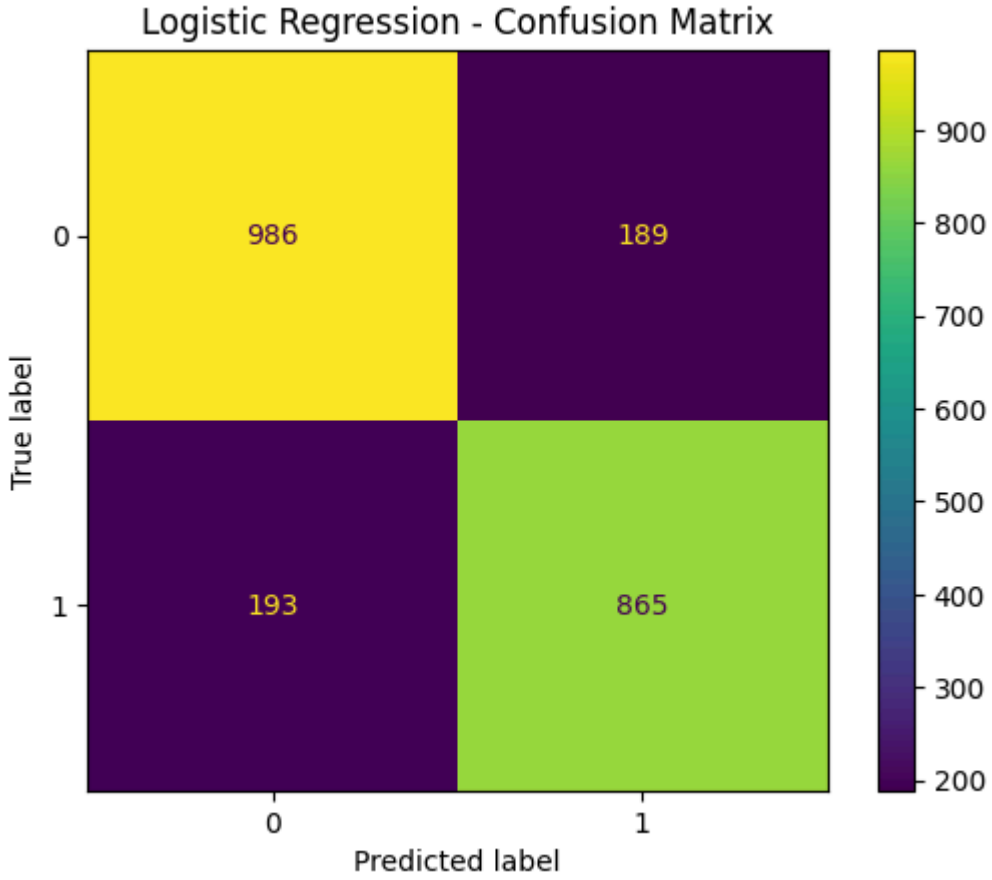


Evaluate Models

In [22]:

```
from sklearn.metrics import classification_report, confusion_matrix, Conf

def evaluate_model(name, model):
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:,1] if hasattr(model.named_ste

    print(f"\n=== {name} ===")
    print(classification_report(y_test, y_pred, digits=4))
    ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred)).plot()
    plt.title(f'{name} - Confusion Matrix')
    plt.show()

    if y_proba is not None:
        auc = roc_auc_score(y_test, y_proba)
        print(f"ROC-AUC: {auc:.4f}")
    print("="*40)

# Evaluate
evaluate_model("Logistic Regression", pipe_lr)
evaluate_model("Decision Tree", pipe_dt)
```

```
=== Logistic Regression ===
              precision    recall  f1-score   support

           0     0.8363    0.8391    0.8377      1175
           1     0.8207    0.8176    0.8191      1058

    accuracy                         0.8289      2233
   macro avg     0.8285    0.8284    0.8284      2233
weighted avg     0.8289    0.8289    0.8289      2233
```

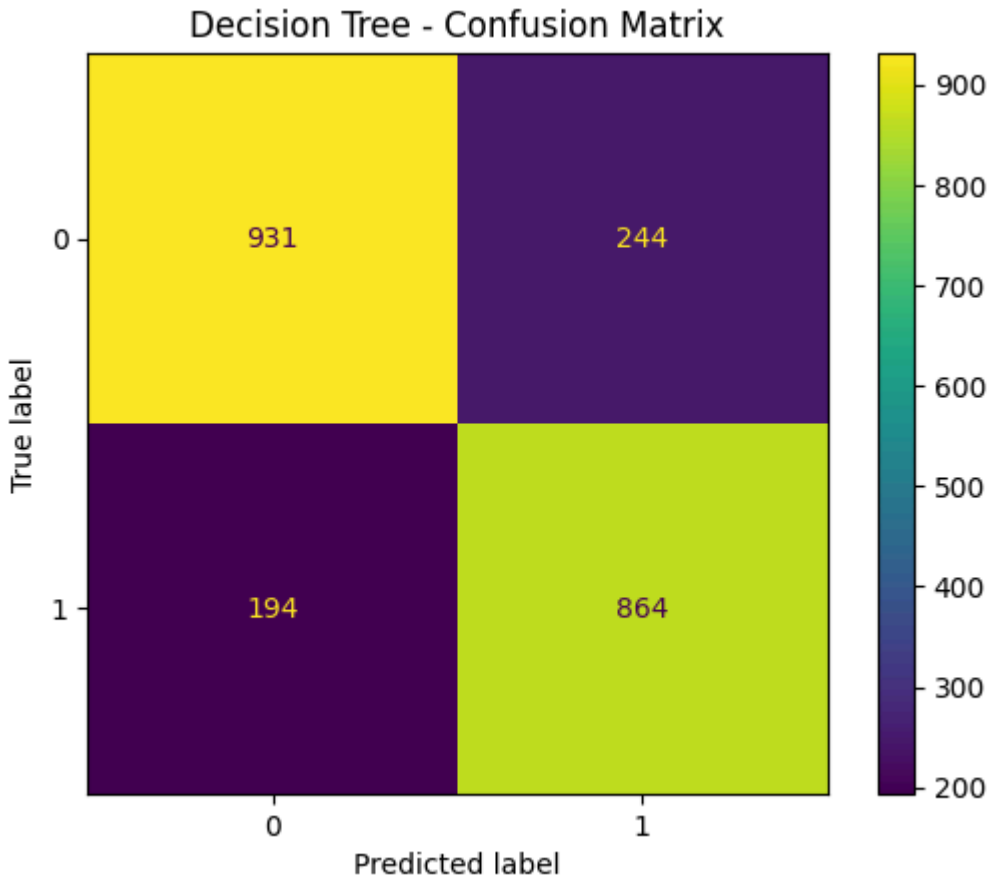


ROC-AUC: 0.9071

==========

=== Decision Tree ===

	precision	recall	f1-score	support
0	0.8276	0.7923	0.8096	1175
1	0.7798	0.8166	0.7978	1058
accuracy			0.8039	2233
macro avg	0.8037	0.8045	0.8037	2233
weighted avg	0.8049	0.8039	0.8040	2233




```
ROC-AUC: 0.8745
========================================
```

Compare All Models

In [23]:

```
models = {
    'Logistic Regression': pipe_lr,
    'Decision Tree': pipe_dt
}

results = []

for name, model in models.items():
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:,1]
    auc = roc_auc_score(y_test, y_proba)
    f1 = np.mean([f1 for f1 in classification_report(y_test, y_pred, outp
    results.append([name, auc, f1])

results_df = pd.DataFrame(results, columns=['Model', 'ROC-AUC', 'Avg F1']
results_df
```

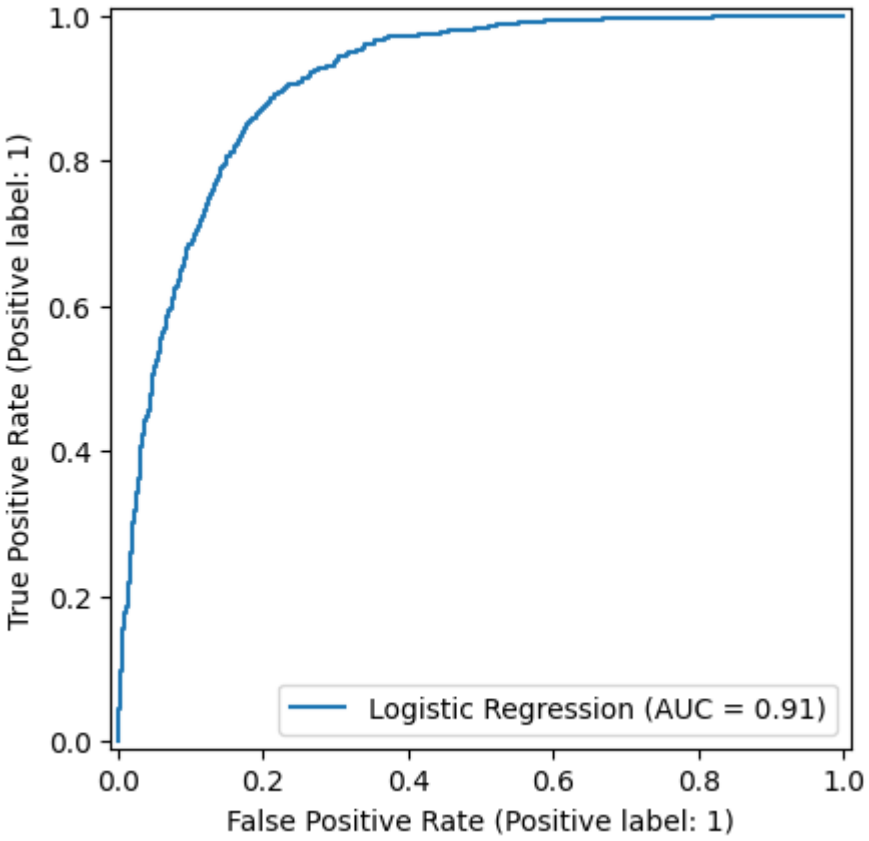
Out[23]:

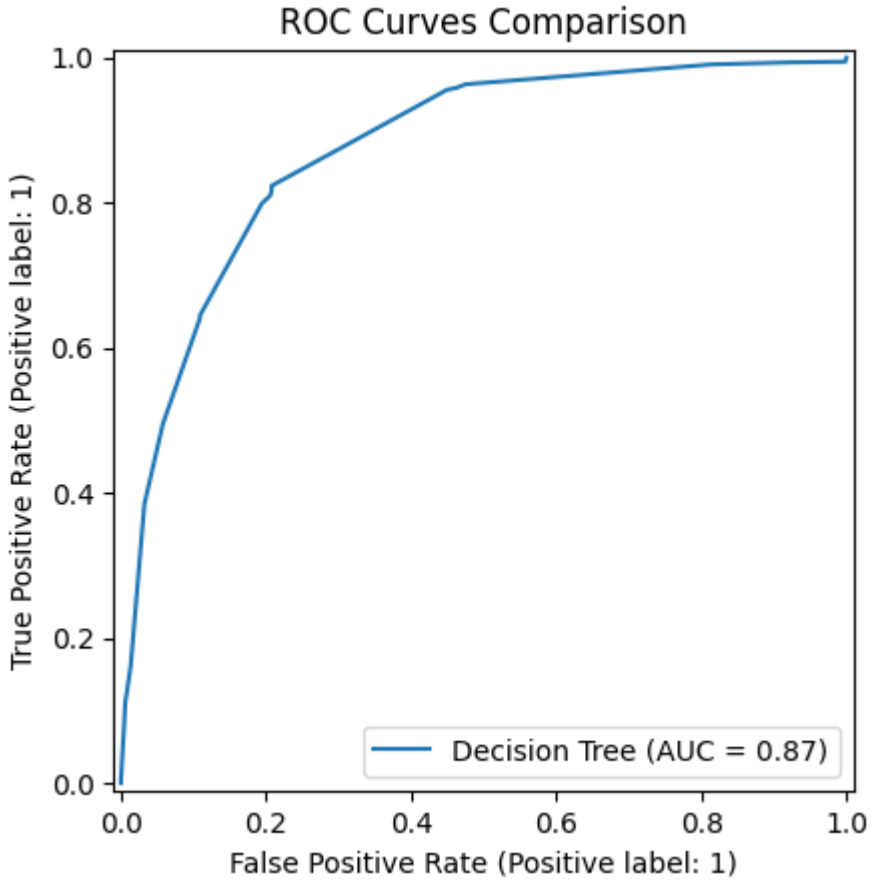
	Model	ROC-AUC	Avg F1
0	Logistic Regression	0.907147	558.871686
1	Decision Tree	0.874472	558.853189

ROC Curve Visualization

In [24]:
```
from sklearn.metrics import RocCurveDisplay

plt.figure(figsize=(6,5))
for name, model in models.items():
    RocCurveDisplay.from_estimator(model, X_test, y_test, name=name)
plt.title("ROC Curves Comparison")
plt.show()
```

```
<Figure size 600x500 with 0 Axes>
```




ROC Curves Comparison
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Decision Tree (AUC = 0.87)
0.0
0.2
0.4
0.6
0.8
1.0