```
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


# Step 2: Loading the dataset
data = pd.read_csv('/content/Telco-Customer-Churn-dataset.csv')


# Step 3: Data Preprocessing

# Check for missing values
print(data.isnull().sum())
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
# Convert 'TotalCharges' to numeric and handle any spaces or non-numeric values
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')

# Now that 'TotalCharges' is numeric, you may fill any remaining invalid values with the median or mean
data['TotalCharges'].fillna(data['TotalCharges'].median(), inplace=True)


# Drop the 'customerID' column as it is not useful for prediction
data.drop(['customerID'], axis=1, inplace=True)


# Encode categorical features using Label Encoding or One-Hot Encoding
label_encoder = LabelEncoder()


# List of columns to label encode (Binary and ordinal categories)
label_encode_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn']

for col in label_encode_cols:
    data[col] = label_encoder.fit_transform(data[col])


# Apply One-Hot Encoding for remaining categorical columns
data = pd.get_dummies(data, columns=['MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
                                     'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
                                     'Contract', 'PaymentMethod'], drop_first=True)


# Step 4: Splitting the data into train and test sets
# Define X (features) and y (target)
X = data.drop('Churn', axis=1)  # 'Churn' is the target column
y = data['Churn']

# Split into train and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Step 5: Train models and compare

# Logistic Regression
```

```
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression(max_iter=1000)
```

```
# Predictions and evaluation for Logistic Regression
y_pred_log = log_model.predict(X_test)
log_accuracy = accuracy_score(y_test, y_pred_log)

print("Logistic Regression Accuracy: ", log_accuracy)
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred_log))
print("Classification Report: \n", classification_report(y_test, y_pred_log))
```

```
Logistic Regression Accuracy:  0.8211497515968772
Confusion Matrix:
 [[933 103]
 [149 224]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      1036
           1       0.69      0.60      0.64       373

    accuracy                           0.82      1409
   macro avg       0.77      0.75      0.76      1409
weighted avg       0.82      0.82      0.82      1409
```

```
# Decision Tree Classifier
dt_model = DecisionTreeClassifier(max_depth=5)  # You can tune parameters
dt_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=5)
```

```
# Predictions and evaluation for Decision Tree
y_pred_dt = dt_model.predict(X_test)
dt_accuracy = accuracy_score(y_test, y_pred_dt)

print("Decision Tree Accuracy: ", dt_accuracy)
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred_dt))
print("Classification Report: \n", classification_report(y_test, y_pred_dt))
```

```
Decision Tree Accuracy:  0.8062455642299503
Confusion Matrix:
 [[964  72]
 [201 172]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.93      0.88      1036
           1       0.70      0.46      0.56       373

    accuracy                           0.81      1409
   macro avg       0.77      0.70      0.72      1409
weighted avg       0.80      0.81      0.79      1409
```

```
# Step 6: Compare model performances
print(f"Logistic Regression Accuracy: {log_accuracy}")
print(f"Decision Tree Accuracy: {dt_accuracy}")
```

```
--INSERT--
```

```
Logistic Regression Accuracy: 0.8211497515968772
Decision Tree Accuracy: 0.8062455642299503
```