```
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.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
```

```
# Set random seed for reproducibility
np.random.seed(42)

# Number of samples
num_samples = 400

# Generate random features
Age = np.random.randint(18, 60, num_samples)  # Random ages between 18 and 60
EstimatedSalary = np.random.randint(20000, 120000, num_samples)  # Salary range

# Generate target variable (1 = Purchased, 0 = Not Purchased) based on some logic
Purchased = np.where((Age > 30) & (EstimatedSalary > 50000), 1, 0)

# Create a DataFrame
df = pd.DataFrame({'Age': Age, 'EstimatedSalary': EstimatedSalary, 'Purchased': Purchased})

# Display first few rows
df.head()
```

	Age	EstimatedSalary	Purchased
0	56	66175	1
1	46	27805	0
2	32	25237	0
3	25	40056	0
4	38	65543	1

```
# Define features (X) and target (y)
X = df[['Age', 'EstimatedSalary']]
y = df['Purchased']

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

```
# Standardizing features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Initialize the model
model = LogisticRegression()

# Train the model
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

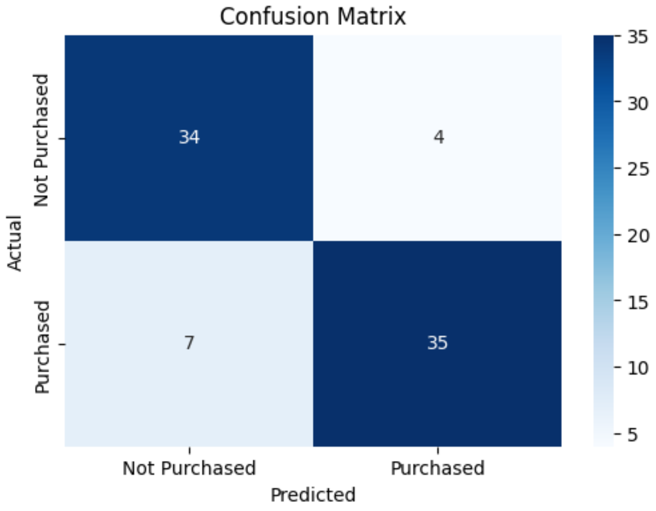
```
# Predict on test data
y_pred = model.predict(X_test)
```

```
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
TN, FP, FN, TP = cm.ravel()

# Compute performance metrics
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
```

```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

# Print results
print(f"Confusion Matrix:\n{cm}")
print(f"True Positives (TP): {TP}")
print(f"False Positives (FP): {FP}")
print(f"True Negatives (TN): {TN}")
print(f"False Negatives (FN): {FN}")
print(f"Accuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Confusion Matrix:
[[34  4]
 [ 7 35]]
True Positives (TP): 35
False Positives (FP): 4
True Negatives (TN): 34
False Negatives (FN): 7
Accuracy: 0.86
Error Rate: 0.14
Precision: 0.90
Recall: 0.83
```

```
# Plot confusion matrix
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Purchased', 'Purchased'], yticklabels=['Not Purchased', 'Purchased
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



```
df.to_csv("synthetic_social_network_ads.csv", index=False)
print("Dataset saved as synthetic_social_network_ads.csv")
```

```
Dataset saved as synthetic_social_network_ads.csv
```