```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.impute import SimpleImputer

from sklearn.metrics import accuracy_score, confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
f1_score


# Load the Social_Network_Ads dataset

social_data = pd.read_csv('Social_Network_Ads_for_prac5.csv')


# Check for NaN values

print(social_data.head(25))


# Separate features (X) and target variable (y)

X = social_data[['Age', 'EstimatedSalary']]

y = social_data['Purchased']


# Handle NaN values by imputing the mean

imputer = SimpleImputer(strategy='mean')

X = imputer.fit_transform(X)


# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Feature scaling

scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create a Logistic Regression model
model = LogisticRegression(random_state=42)

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

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)

# Compute Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Extract values from the confusion matrix
TN, FP, FN, TP = conf_matrix.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)
f1 = f1_score(y_test, y_pred)
```

```python
# Print the results
print("Confusion Matrix:")
print(conf_matrix)
print("\nTrue Positive (TP):", TP)
print("False Positive (FP):", FP)
print("True Negative (TN):", TN)
print("False Negative (FN):", FN)
print("\nAccuracy:", accuracy)
print("Error Rate:", error_rate)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

# Plot the decision boundary
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='EstimatedSalary', hue='Purchased', data=social_data, palette='viridis')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')

# Plotting decision boundary
h = 0.5
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.3)
plt.title('Logistic Regression Decision Boundary')
```

```
plt.show()
```