```
 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.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Step 1: Create the dataset
np.random.seed(42)
data = {
    'User Id': range(1, 501),
    'Gender': np.random.choice(['Male', 'Female'], 500),
    'Age': np.random.randint(18, 65, 500),
    'Estimated Salary': np.random.randint(20000, 150000, 500),
    'Purchased': np.random.choice([0, 1], 500)  # 0: Not Purchased, 1: Purchased
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Step 2: Define independent and target variables
X = df[['Age', 'Estimated Salary']]  # Independent variables
y = df['Purchased']  # Target variable

# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Step 4: Build a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 5: Predict and evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)

# Step 6: Predict for a new person
new_person = [[30, 70000]]  # Age = 30, Estimated Salary = 70,000
prediction = model.predict(new_person)

# Output the prediction
if prediction[0] == 1:
    print("The person is likely to buy a car.")
else:
    print("The person is not likely to buy a car.")

# Step 7: Plot the decision boundary and scatter plot
plt.figure(figsize=(10, 6))

# Scatter plot for the data points
sns.scatterplot(x=X_test['Age'], y=X_test['Estimated Salary'], hue=y_test, palette='coolwarm', s=100)
plt.title('Logistic Regression: Decision Boundary')
plt.xlabel('Age')
```

```
plt.ylabel('Estimated Salary')


# Create a grid to plot the decision boundary
age_range = np.arange(X_test['Age'].min() - 1, X_test['Age'].max() + 1, 0.1)
salary_range = np.arange(X_test['Estimated Salary'].min() - 1000, X_test['Estimated Salary'].max() + 1000, 100)
age_grid, salary_grid = np.meshgrid(age_range, salary_range)
grid_points = np.c_[age_grid.ravel(), salary_grid.ravel()]


# Predict the class for each point in the grid
grid_predictions = model.predict(grid_points).reshape(age_grid.shape)


# Plot the decision boundary
plt.contourf(age_grid, salary_grid, grid_predictions, alpha=0.3, cmap='coolwarm')
plt.colorbar(label='Purchased (1) or Not Purchased (0)')


plt.legend(title='Purchased', loc='upper right')
plt.show()
```